DATABASE CLOUDS - ACCOUNTING, FORECASTING AND WORKLOAD PLACEMENT FROM COMPLEX RDBMS ARCHITECTURES

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING, SCHOOL OF ENGINEERING,
DEPARTMENT OF COMPUTER SCIENCE

2022

By
Antony S. Higginson
Student id: 9582865
Under the Supervision of
Prof Norman W. Paton
and
Dr Suzanne Embury

Department of Computer Science

Contents

Word Count: 71119

List of Tables

List of Figures

Abstract

Most *on-premises* database architectures employ advanced database features such as replication (standby databases), consolidation and separation (pluggable databases), and high availability (HA) through clustered configurations, each with a dizzying number of metrics to help monitor, assess and notify if errors occur. These advanced database architectures are explicitly designed to enforce enterprise Service Level Agreements (SLAs), maximising stability while serving critical business functions. In Cloud architectures, a trade off between the cost of the infrastructure and the ability for custom configuration has become apparent, therefore knowing what size, configuration and where to place workloads is most important prior to any cloud adoption if we are to achieve the benefits that clouds profess to bring. It has been advocated by cloud service providers that cloud configurations reduce cost, speed up key business processes and improve support for agile development lifecycles. However, when enterprises suffer from server sprawl, the ability to individually assess each database system prior to cloud migration has resulted in *paralysis-by-analysis*. This can lead to guestimates of what resources are perceived as being used or required. In this thesis we investigate how to account for complex database architectures and their workloads in clouds, focusing on accounting, forecasting and capacity planning, workload placement, answering questions that are key to successful cloud adoption such as: What types of workloads are employed? Do those workloads exhibit complex data patterns such as trends, shocks or seasonality? What resources are the workloads consuming and require in the future? How should the workloads be placed to fully utilise database cloud architectures without compromising existing SLAs? Introducing new techniques and automation, we remove the high level of manual technical expertise and understanding currently employed by enterprises, which can be seen as error prone, cumbersome and time consuming.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to thank my supervisors Norman and Suzanne whom have been immense in their support and guidance especially when my wife, Laura, was tragically killed during the second year of my study. There were many times I simply wanted to give up and it was the support of them and the University of Manchester that gave me the strength to carry on.

I would also like to thank the Oracle Corporation that provided access to environments and software engineers to fully explore my ideas and test them in real life configurations on cutting edge technology. Without their support we could not have performed the experiments to the full potential we required.

Finally, I would like to thank my family, My darling wife Laura, whom I miss daily. I hope she is proudly looking down on me from heaven as it was her who supported me at the start of this study. My children, as they start on their own educational quest, who, hopefully can use my experience as inspiration for their own academic achievements.

Chapter 1

Introduction

The term 'Intergalactic Network' was a kind of intentionally grandiloquent way to express the idea, because we didn't really expect to get at that right away. It was all we could possibly do to make timesharing systems work

Dr. J. C. R. Licklider 1962

The adoption of cloud is now, arguably, an inevitability rather than a choice. When the idea of inter-connecting computers was first muted in the 1960's it was deemed one of fantasy or prophecy by DR J.C.R. Licklider. Dr Licklider had a vision of networked computers in his internal 1963 APRA memo that, arguably, provides the backbone or building blocks of how cloud computing has become a reality today, that is to say clouds are accessed via a network [Kit03]. The *'inter-galactic network'* in terms of cloud computing and its adoption is still a conundrum to solve as the paradigm of what was seen as an implementation problem in 1963 by Licklider, who cited that a time-sharing technology as the technology to connect a network of such centres [Ber05] can have similar parallels today by raising the question of, what do I need? Presently clouds store any type of data with an added complexion that access and its provision of resources is subscribed to as *everything-as-a-service* at all levels of the technological stack.

In 2020 Gartner® produced a competitor analysis [RGARHCDF20] highlighting the complex landscape of Cloud Service Providers and their offerings. Gartner defined a cloud database management system (DBMS) as being that from vendors that supply

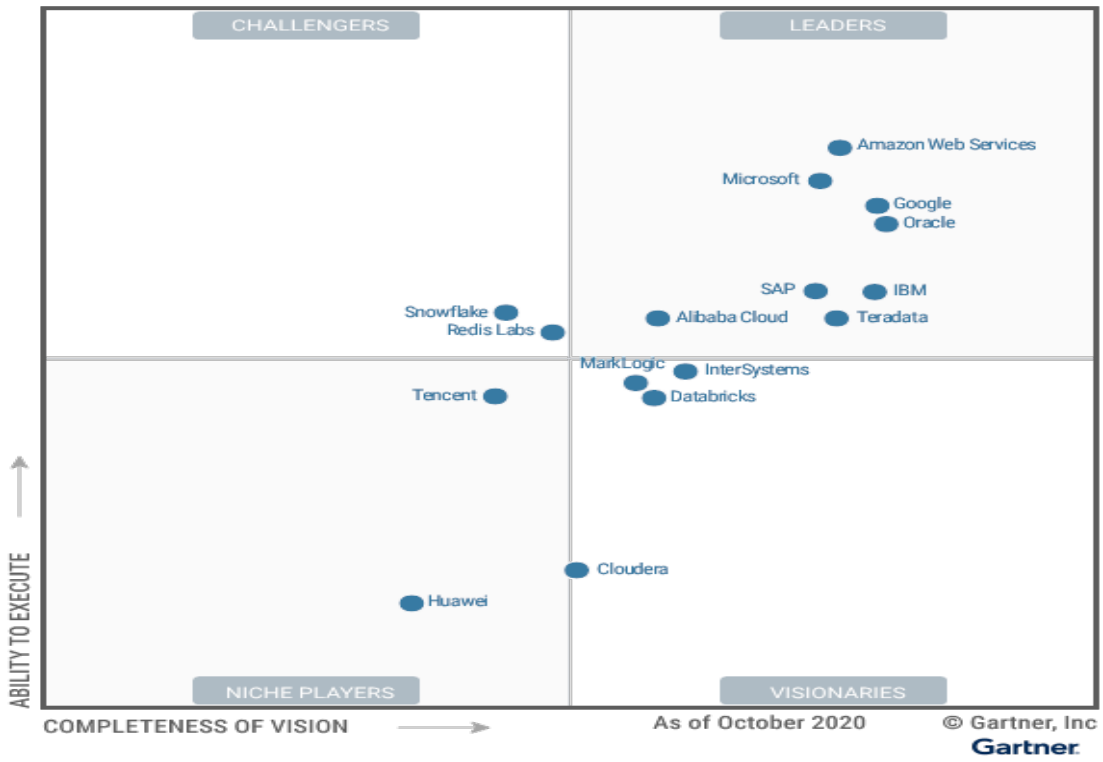


Figure 1.1: Gartner Magic Quadrant for Cloud Database Management Systems [RGARHCDF20]

fully provider-managed public or private cloud software systems that manage data in cloud storage (object, distributed data or other propriety storage infrastructure) that may use multiple data models such as relational, non-relational (document, key-value, wide-column, graph), geospatial, time series and others.

This market research performed by analysts Gartner, as shown in Figure 1.1, highlighted that the traditional DBMS offerings from the main vendors in the market place such as Microsoft, IBM, Oracle and SAP *et al* are all moving their database products to the cloud, thus joining new database vendors such as AWS and Google. This migration by the vendors has forced the users, who haven't yet fully adopted cloud, to also adopt if they are to continue their usage of the vendor software they rely on. Software vendors who are pushing their cloud prophecy, have an effect on software consumers having to adopt the paradigm. A feedback loop is created between the customers and their enterprises reliance on vendor software or technology to satisfy their own business needs. This has created a market shift that requires a brief historical synopsis.

1.1 What are Clouds?

Cloud computing, characterised by the National Institute of Standards and Technology (NIST) in 2011 [MG11] as: a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics (on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service), three service models (Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS)), and four deployment models (Private, Community, Public and Hybrid Clouds).

There is a notable omission of *Database-as-a-Service (DBaaS)* from this 2011 NIST classification, even though DBaaS is widely accepted as a cloud-service. DBaaS was first muted as far back as 2002 by Hacigumus *et al* [HIM02]. In-fact, there are also many other *as-a-service* terms missing from the NIST classification and for many years scientists struggled to classify *anything-as-a-service* in the early years of cloud computing, as depicted by Duan *et al* [DFZ+15], who did a study on the classifications of cloud. For the purpose of this thesis, it is important that we refer to Database Services provided by cloud architecture (stated by NIST) as DBaaS and Database Services that are provided by traditional architecture as on-premises. This is because we will be providing analysis from experiments with workloads executed on traditional on-premises architectures with a view to placing a *'shape'* of resources in a DBaaS cloud.

1.2 Monitoring and Provisioning in Clouds

Cloud provisioning requires a high level of orchestration and automation to allocate the Cloud Service Providers (CSP) resources and services to a customer. Provisioning is a key feature in the cloud computing model, because it concerns how a customer procures resource usage. Most cloud service providers aim to provide this functionality through an access interface such as a web portal via a *single-pane-of-glass*. A user can procure a shape of resources and configuration from a catalog of available resources, resulting in a slice of topology. For example, *Infrastructure* consisting of physical

server, storage and network (CPU, Storage, Memory and network Bandwidth), *Platform* such as a Virtual Machine (VM) with an Operating system or *Software* to execute or run an application. Cloud Provisioning is important because it allows the customer the ability to scale over a traditional model with on-premises, where the infrastructure is setup to last for years. Knowing what slice of cloud, involves knowing the size therefore a critical facet of provisioning is an understanding of capacity planning.

Provisioning through high orchestration such as the automatic installation of a *ready-made* functioning VM, is not an easy thing to achieve because the organisation may require complex management and monitoring of workloads residing on different clouds provided by different cloud vendors. Most CSPs provide cloud through something called compute, which is a combination of CPU, Memory, Networking and Storage. Other service offerings or ancillary services that improve capabilities such as machine learning or analytics are provided on top of this cloud configuration. Policy enforcement in a self-service provisioning model helps allow users to only procure what is available to them so not to *over-procure*. Introducing these functions of provisioning to the right teams all have one common feature, which is configuration and a facet of configuration is capacity and workload placement. Cost control is achieved through active monitoring of the resources being consumed and then offering the user notifications via alerts if the limit of thresholds of resources being consumed are breached. Effective monitoring of the resources allows the customer the scale by elasticising with the aim of maintaining service (SLAs)

1.2.1 DevOps

DevOps was spawned out of the lack of collaboration between development and operation teams with their constant friction over the processes or delivery practices when employing the Software Development Life Cycle (SDLC) as Mahanta *et al* describe [RPAM16]. Mahanta *et al* suggested that DevOps is not only a methodology but a mindset that has, through orchestration tooling, helped bridge a traditional gap between teams that often acted separately, which was often exaggerated by on-premises silo's. DevOps is also seen as an enabler for the Agile development methodology as it streamlines continuous delivery. DevOps provides this through pipelines optimisation, for example, software is committed to a software repository such as GitHub. A deployment pipeline creates the software artefacts, which are then deployed to Dev, Test and Staging. At each stage in the pipeline, the relevant notifications and approvals are issued before, finally, being deployed to the live platforms as a software update as

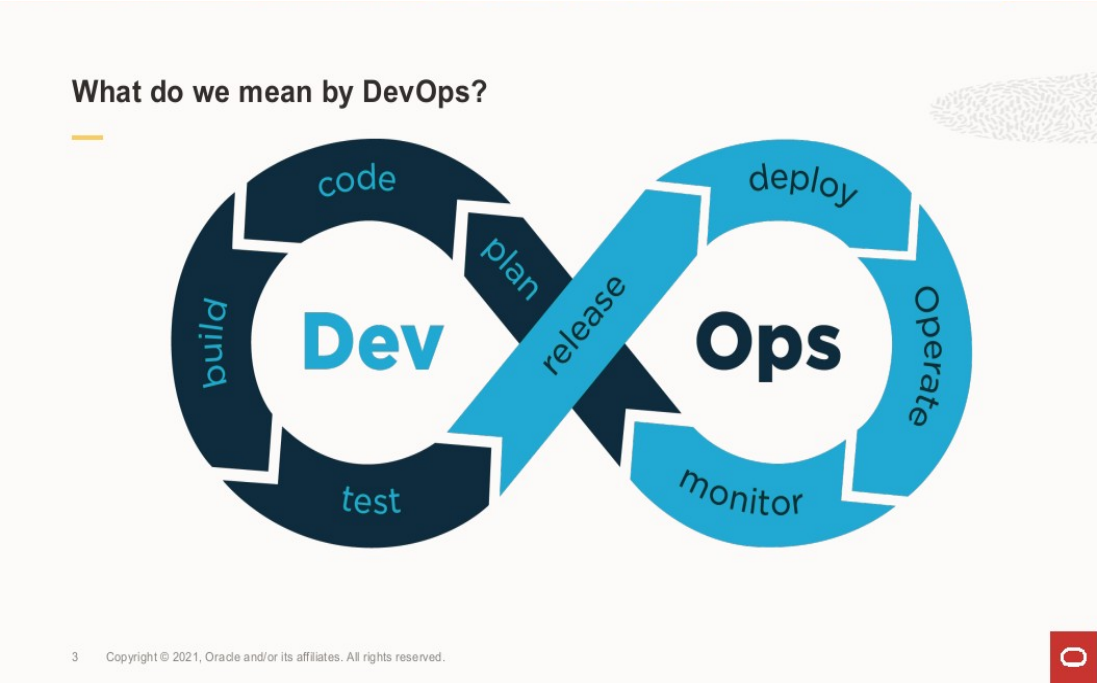


Figure 1.2: What do we mean by DevOps? [Mar21]

shown in Figure 1.2 ("Copyright Oracle and its affiliates. Used with permission").

DevOps is different to Agile even though they have correlations that people may assume are the same. Both work in conjunction with each other and are critical to organisations looking to adopt cloud, given cloud implies a high level of orchestration and automation. Agile seeks to improve efficiency for developers and release schedules, where DevOps brings together continuous delivery and implementation as described by Mohammed [Moh17].

1.2.2 AIOps

Artificial Intelligence (AI) for IT Operations (AIOps) was first coined by Gartner [Gar19] as the use of AI related technologies for traditional IT Operation activities and tasks. This may include correlation, anomaly detection, error processing and their resulting log files from said errors or metric consumption. Dang *et al* [DLH19] produced a survey into this emerging area as one that is attractive because it allows the customer or user to proactively monitor for faults rather than use threshold based reactive monitoring. This is an area that part of our research conducted in this thesis also explores. Utilising a supervised machine learning engine to perform a prediction on time series data to inform the user that there could be, *'potentially'*, a problem of

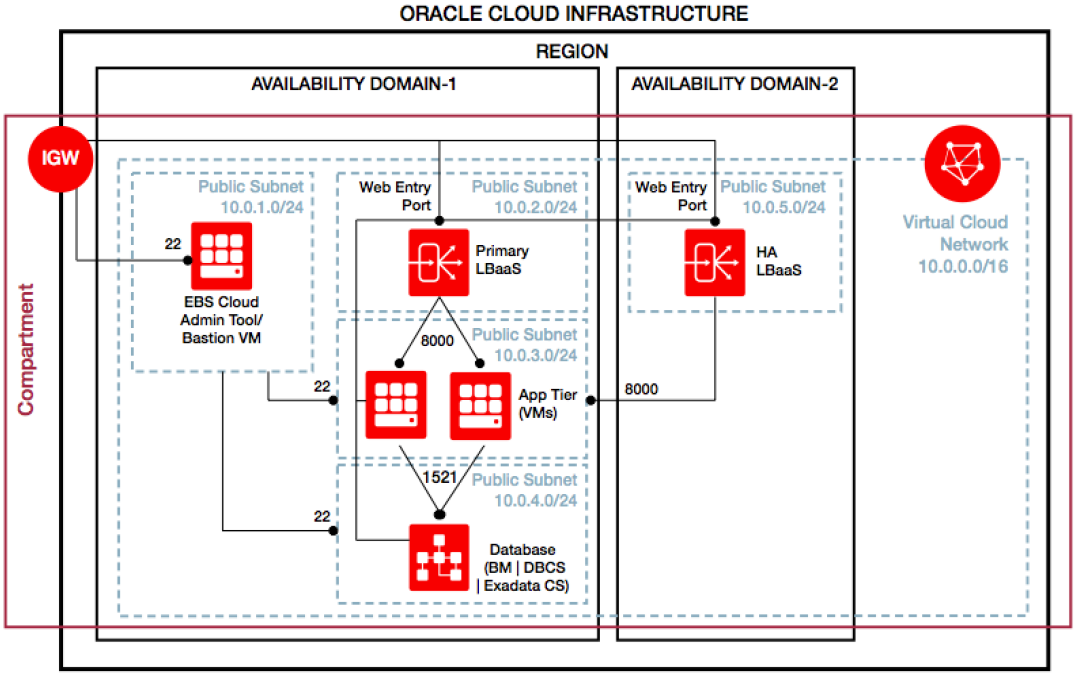


Figure 1.3: Sample OCI Database Configuration [Mar21]

resource exhaustion. This was a biproduct of our original intention to predict what a workload could consume prior to placement in a cloud. From a monitoring perspective, the prediction algorithm proposed and developed as part of this thesis could aid preventative monitoring for faults attributed to resource exhaustion by predicting them routinely.

1.3 Cloud Architecture

Understanding cloud architecture is complex because it involves lots of different concepts such as Compute, Storage, Network, Availability and Security, which can be sliced in different ways all working in harmony. All the dominant cloud service providers of Oracle, Microsoft, Google and AWS they have similar concepts but are different in name as shown in Tables 1.1, 1.2, 1.3, 1.4 and 1.5. All CSPs offer ancillary services such as Analytics, Big Data, Machine Learning and Artificial Intelligence (ML/AI), with specialised cloud configurations to fit particular industries. For the purposes of the thesis we will be focusing on Oracle Cloud Infrastructure (OCI) in a bare metal (without aa base operating system) configuration, which is particularly important for the final paper discussed in Section 5.3. We have listed AWS, Azure and

GCP as comparisons to support that our work could equally work in clouds other than that of Oracle, but through the support of Oracle who provided access to technology we have focused on Oracle Cloud Infrastructure.

An example of how cloud can be implemented from an architectural perspective is shown in Figure 1.3 ("Copyright Oracle and its affiliates. Used with permission"). Here we have *compartments* each of which is a *slice* of cloud consisting of network, storage, compute split over several Domains located in a Region. From a networking perspective LBaaS (*Load-Balancing-as-a-service*) helps provide a High Availability (HA) function with each Domain having their own subnets networking configurations. A region is a geographic location and all CSPs provide data centres geographically located throughout the world to serve customers geographically. In this example we can see that the compartment consists of enough compute to store databases and applications with the requisite private and public networks via ports that control access via accounts implemented through security policies. The database and its optimiser while reflect the number of CPUs, amount of memory and size of and speed of storage (IO) are not reflective in Figure 1.3. Intelligent Agents are installed locally on each entity with access to the server interrogating a workload. Access to the central repository that stores the metric data can be held remotely (via the agent) or in a data layer within the Cloud Architecture itself. This is not shown in Figure 1.3.

Regions and Availability Domains				
Concept	Amazon (AWS)	Google (GCP)	Microsoft (AZURE)	Oracle (OCI)
Cluster of Data Centres & Services	Region	Regions	Region	Region
Abstracted data Centre	Availability Zone	Zones	Availability Zone	Availability Domain
Hardware Groupings	-	Multiregional Resources, Global Services	Fault Domains	Fault domains

Table 1.1: Comparison of Cloud Regions and Availability Domains [Goo21b], [Cor21e], [Cor21f], [Cor21g]

Accounts, Tagging ad Organising				
Concept	Amazon (AWS)	Google (GCP)	Microsoft (AZURE)	Oracle (OCI)
Account organising resources	Account Resource Groups	Account	Account Subscriptions Resource Groups	Tenancy Compartments
Metadata to resources	Tags	Hashes and eTags	Tags	OCI Tagging
Multiple accounts management	AWS Organisations		NA	NA

Table 1.2: Comparison of Cloud Access Architecture [Goo21b], [Cor21e], [Cor21f], [Cor21g]

Compute Service Mapping				
Concept	Amazon (AWS)	Google (GCP)	Microsoft (AZURE)	Oracle (OCI)
Multi-tenant Virtual Machines	Elastic Compute Cloud (EC2)	Compute Engine	Azure VM's	OCI VM Instances
Single tenant Virtual Machines			Azure Dedicated Hosts	OCI Dedicated VM Hosts
Bare Metal hosts	AWS EC2 - I3.metal		Azure BareMetal Infrastructure	OCI Bare Metal Instances
Managed Kubernetes Service and Registry	Amazon Elastic Kubernetes Service (EKS) Amazon Elastic Container Registry	Kubernetes Engine Container registry	Azure Kubernetes Service (AKS) Azure Container Registry	Oracle Container Engine for Kubernetes OCI Registry
Serverless	Lambda	Cloud Functions	Azure Functions	Oracle Functions

Table 1.3: Comparison of Cloud Compute Architecture [Goo21b], [Cor21e], [Cor21f], [Cor21g]

1.4 Traditional Enterprise On-Premises Architecture

Prior to cloud computing, organisations utilised I.T. attached to data centres which the organisation was responsible for. Within these data centres physical infrastructure was installed causing organisations to spend vast amounts of money in procuring and maintaining this equipment though Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) financial models. This created financial strains because it was an up-front investment, thus architects were employed to create architectural *patterns* that were flexible to enable SDLC yet easy enough to maintain. *N-Tier* architecture emerged as the dominant architectural pattern that could achieve these goals. An example of legacy customer database architecture utilising an N-Tier approach can be

Storage Service Mapping				
Concept	Amazon (AWS)	Google (GCP)	Microsoft (AZURE)	Oracle (OCI)
Object Storage	AWS Simple Storage (S3)	Cloud Storage Cloud Storage Nearline	Blob Storage	Object Storage
Archival Storage	AWS S3 Glacier	Cloud Storage Coldline	Blob Storage (archive access tier)	Archive Storage
Block Storage	AWS Elastic Block Store (EBS)	Compute Engine Persistent Disks	Managed disks	Block Volumes
Shared File System	AWS Elastic File System		Azure Files	File Storage
Bulk Data Transfer	AWS Snowball	Transfer Appliance ; Transfer Service	Import/Export Azure Data Box	Data Transfer Appliance
Hybrid Data Migration	AWS Storage gateway	ZFS / Avere	StorSimple	Storage Gateway

Table 1.4: Comparison of Cloud Storage Architecture [Goo21b], [Cor21e], [Cor21f], [Cor21g]

found in Figure 1.4. In this diagram we see several configurations of database (Single and clustered) running on VMs. These databases then employ standby databases systems that are also running on VMs. The storage tier is accessed via mount points through Network Attached Storage (NAS) to a Storage Area Network (SAN).

1.4.1 N-Tier Architecture

N-tier can also be referred as Multi-tier architecture because software is separated out into different tiers both physically and logically. By enforcing separation it allows each tier to be delivered to the top of its capacity without necessarily impacting another tier. For example, the apps tier is separated from the database tier ensuring that the application does not take resources from the database located in the data tier. There are three main tiers but the term *N* meaning any number from 1, allowing the system to be further dissected into separation. For example the data tier being further disseminated into further tiers such as storage and database to incorporate Storage Area Network (SAN)/ Network Attached Storage (NAS) technology:

- Presentation tier - The top-most level for the application is the user interface with the role being to translate the task being requested by the application, into a result that the user understands.

Database Service Mapping				
Concept	Amazon (AWS)	Google (GCP)	Microsoft (AZURE)	Oracle (OCI)
Managed Relational Database systems	Amazon Relational Database Service (RDS) Amazon Aurora	Cloud SQL Cloud Spanner Cloud-SQL Cloud SQL (MySQL, Postgres)	SQL Database Database for MySQL Database for PostgreSQL	Oracle Autonomous Transaction Processing (ATP), Oracle MySQL Database Service
NoSQL	Amazon DynamicDB	Cloud Datastore, Cloud Bigtable	Table Storage Cosmos DB	Oracle NoSQL Database Cloud Service, Oracle Autonomous JSON Database (AJD)
Data Warehousing	Amazon Redshift	BigQuery	Synapse Analytics	Oracle Autonomous Data Warehouse (ADW), Oracle MySQL HeatWave

Table 1.5: Comparison of Cloud Database Services Architecture [Goo21b], [Cor21e], [Cor21f], [Cor21g]

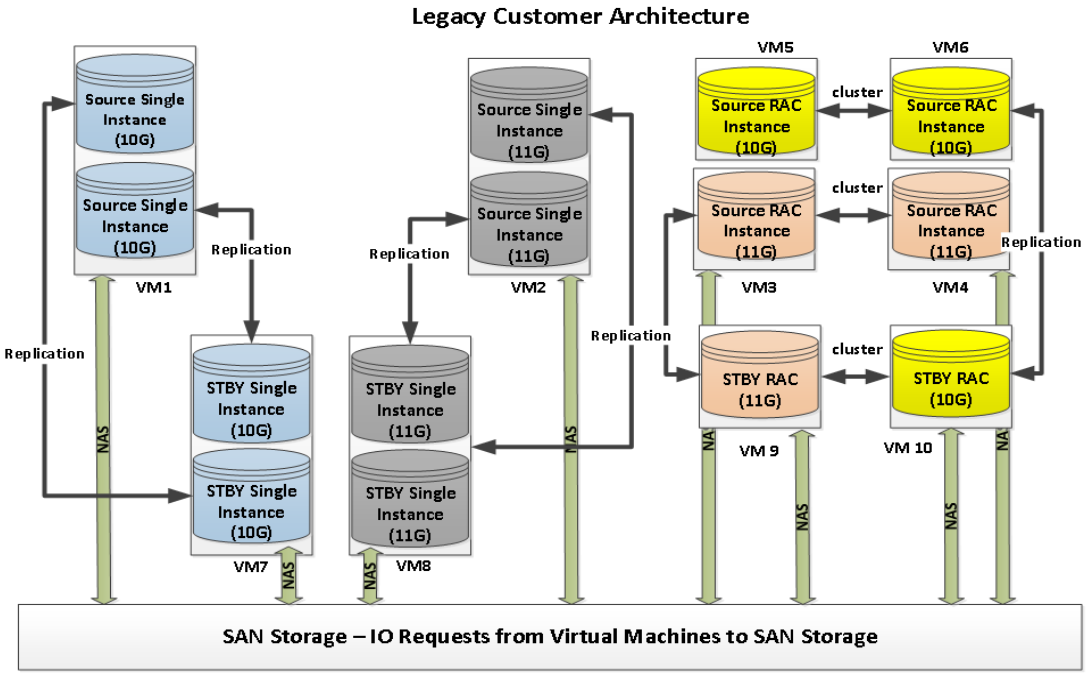


Figure 1.4: Legacy N-Tier Architecture

- Logic tier - This tier processes, coordinates and makes logical decisions such as calculations and evaluations on the data as it moves around or between the presentation and data tiers.
- Data tier - The information is stored and retrieved for a database or filesystem.

Since the advent of cloud one major problem is how to migrate or port these *N-Tier* architectures to cloud raising the key question of cloud portability. Kostoska *et al* [KDGR15] specifically looked at this area, proposing, that given a system portability problem, one may need to understand the relationships and dependencies between the services and application prior to cloud migration. There are several ways to do this such as Topology and Orchestration Services for Applications (TOSCA). In recent years all cloud vendors provide portability tooling or *lift-and-shift* capability by either supporting the application if it is open stack or migrating the data itself. This reverts the question back to one of capacity: what size is required and how do I place my application when porting or shifting my applications to cloud?

1.5 Metrics and Workloads

Science uses quantitative measurements by putting a value on something, for example, the rate of reaction by measuring how many seconds it takes for change to happen. Computer science also strives to provide software metrics as a way to measure a software system or process of some property. All operating systems and software allow administrators the ability to view how the system(s) are running from a *perspective*, with metrics providing the ability to do so. This enables administrators in aiding diagnoses should a process or system begin to slow down, they execute commands and identify the process responsible. CSP's provide clouds based on measurements of resources, namely storage and compute, with compute being mainly made up of CPU, local OS Storage and Memory. However, there are many metrics that one could be interested in if a system requires, for example, network *throughput*. Therefore if one is to map or compare one system to another, one must obtain the correct measurement from the right metric to correctly and accurately capacity plan a workload. In our first paper we focused on identifying the correct metrics at the Database layer of on-premises *N-tier* architecture. With a view to obtaining a trace of metric data from a database workload that can be mapped to a cloud architecture. Database Optimisers work by taking the configuration from their surroundings. For example, the number of CPUs,

amount of Memory and Storage, which influences the performance of the task they are performing (SQL). The larger and/or faster the number of machine resources, such as an increase or memory or CPU speed, the greater the number/speed of tasks that can be processed.

The larger and faster the machine the more tasks and efficiency of the database. In Section 5.1, our first paper, we tackle how some configurations in the Operating System can also influence the execution of workload or if the VM has been assigned more CPUs by the hypervisor.

1.6 Capacity Planning

IT capacity planning involves estimating the storage, computer hardware, software and connection infrastructure resources required over some future period of time. A common concern of enterprises is whether the required resources are in place to handle an increase in users or number of interactions. Capacity management is concerned about adding central processing units (CPUs), memory and storage to a physical or virtual server. This has been the traditional and vertical way of scaling up web applications, however IT capacity planning has been developed to include the goal of forecasting the requirements for this scaling approach.

A discrepancy between the capacity of an organization and the demands of its customers results in inefficiency, either in under-utilized resources or unfulfilled customer demand. The goal of capacity planning is to minimize this discrepancy. Demand for an organization's capacity varies based on changes in production output, such as increasing or decreasing the production quantity of an existing product, or producing new products. Better utilization of existing capacity can be accomplished through improvements in overall equipment effectiveness (OEE). Capacity can be increased through introducing new techniques, equipment and materials, increasing the number of workers or machines, increasing the number of shifts, or acquiring additional production facilities. For example, in Equation 1.1 capacity is a function of the number of machines *(M)* needed to satisfy the tasks *(T)* to be satisfied in the available utilisation *(U)* of the machines efficiency *(E)*. There are other definitions of capacity that could be independent of utilisation too. This equation (1.1) is only an example to highlight the problem of capacity planning.

$$Capacity = f(M,T,U,E) \tag{1.1}$$

Systems, arguably, metamorphose over time through the recursive nature of the SDLC as more functionality is added or features employed to enforce SLAs and improve QoS. This is an inevitability of all systems that are employed for long periods of time, coupled with traditional I.T. working in silos has created some notable gaps if one is to adopt cloud. For example, how to map key metrics that make-up a system at each layer of N-tier architecture to Cloud architectures. How to forecast all workloads in an advanced database system (*Cluster*, *Pluggable* or *Standby*) and place workloads together to maximise Total Cost of Ownership (TCO) while satisfying SLA, SLOs or QoS. We specifically tackle this question in the related work Section 3 of this thesis, especially if those systems employ advanced database features such as clustering, replication or pluggable databases. We provide actual results on any evaluations performed and outcomes of the algorithms tested on real world cloud configurations as we discuss in our papers in Section 5.

- How do we account for these complex architectures via metrics, measurements and workloads?
- How do we categorise or differentiate one system and its workload from another?
- How do we map one architecture (on-premises) to another (cloud)?

1.7 Forecasting or Predicting the Future

A critical function of any capacity planning is to stay ahead of growth and when viewed graphically, can be represented through time-series data that extends upward and to the right of any x,y linear chart as shown in Figure 1.5. Forecasting is used extensively in econometrics and is becoming more prominent in I.T. especially with the introduction of Machine Learning that can scale and speed up forecasting models, which we discuss in our second piece of work in Section 5.2. Forecasting is constructing new data points beyond what is already known, if the line goes up insinuating *trend*. All systems exhibit peaks and troughs that can be mapped to system usage such as surges in users. For example, an OLTP system may not be busy at night when, activity is light compared to the day and this can be reflected in the time-series data. Some systems also exhibit particular behaviours periodically. For example, consider a retail banking system that stores account information of its customers. Those customers are paid with a frequency of daily, weekly or monthly but mostly monthly. Therefore the system is more likely to be busier at the end or beginning of the month as that is when

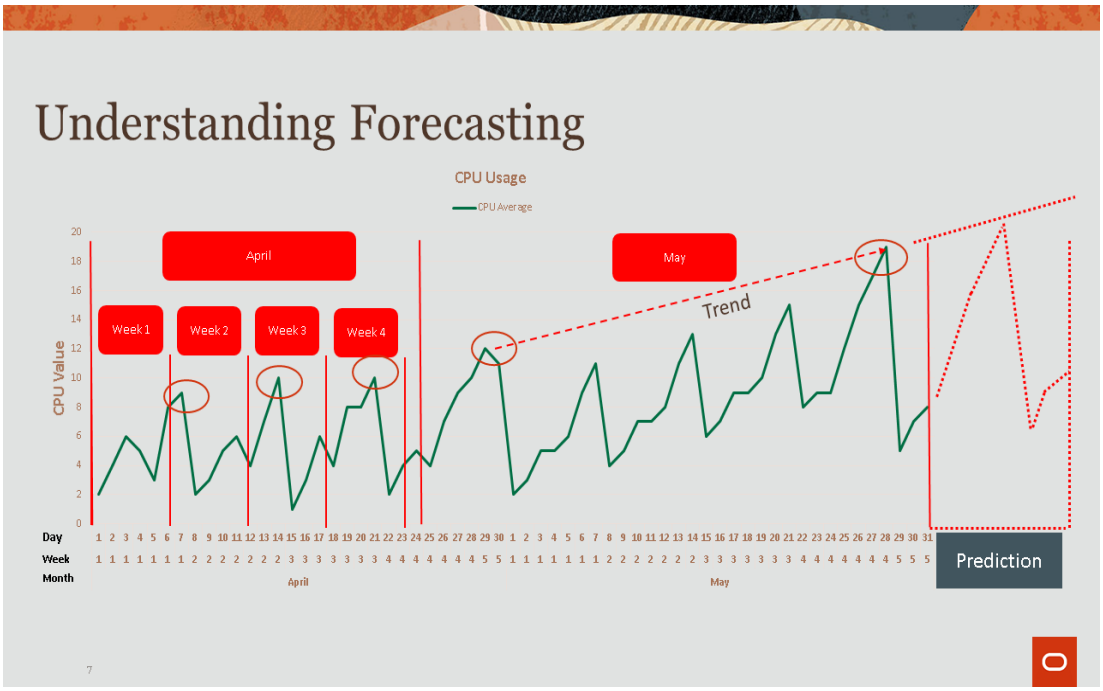


Figure 1.5: Forecasting CPU Metric Data

most people have most disposable income and bill payment activity takes place. Identifying these patterns in the system usage are crucial to forecasting, as shown in Figure 1.5.

Consider a metric such as CPU that has measurements captured on its usage every 15 minutes, producing 96 observations in any 24 hour period. A 15 minute capture period equates to 672 (weekly), 2688 (monthly) and over 8064 quarterly (3 months). Even if we are to reduce the capture rate to hourly it only reduces the data points to approximately 720 hourly data points over a 3 month period. If we are to understand the system usage and the complex exhibits in the data signal (patterns and trends) from the time-series data, one must ensure that we capture the correct measurements at the correct time window. Simply capturing a week will miss the surge in users of a financial accounting system if customers are paid monthly. Equally, the bank is required to submit financial information for tax purposes periodically, causing the analysis window to elongate beyond three months to potentially a year. Being able to provide a forecasting model that can complete in a timely manner, within a few minutes, is also key to user acceptance.

Consider one two-node database cluster that is an Online Transaction Preprocessing System (OLTP) of the financial accounting example, previously given. If we are

required to understand the CPU growth (the amount of CPU cycles consumed to complete a task), then an understanding of several things is required, such as the fact that, configuration of certain parameters at an operating system level can influence CPU. The Number of CPUs assigned to the workload, especially in clustered environments, and the size and model of the CPU. In a clustered environment, one CPU metric has now transposed into two CPU metrics, one from each node. Lots of systems employ multiple nodes of any number from 2, therefore a question of scale becomes a significant challenge, especially if we are to capture the historical picture of the system, through the system, multiple metrics that satisfy the compute aspect of cloud (CPU, Storage and Memory). Evaluating multiple models in a timely fashion was also part of the bigger problem of scale coupled with multiple metrics of a database can vary between one node and 32 nodes, although the standard node configuration is usually 2-4 nodes. It became apparent that there was a need to *filter* the number of models to be evaluated in a timely fashion. The key questions pertaining to Capacity Planning and performing a forecast are:

- How do we understand the historical usage of the system and its workload?
- How can we predict what the future resource consumption of the workload will be?
- How do we scale to forecast all the metrics that make up a system?
- How do we scale to forecast all the system that make up an estate?
- Short Term - How do we predict when resources will run out: proactive monitoring?
- Long Term - How do we perform capacity planning, whether that is on-premises, off-premises or involving non-cloud environments?

1.8 Research Context

Adopting clouds is non-trivial and one of the main barriers to cloud adoption is understanding costs, as Khajeh-Hosseini *et al* [KHGSS12] suggested in their 2010 analysis. They suggested that costs are complicated to meter in cloud, because resource usage is not easily accounted for and the lack of a cloud *toolkit* makes cloud adoption tricky. Understanding resource consumption accurately is part of capacity planning,

and a quick survey of AWS, Microsoft, Google and Oracle all show basic cloud costing tools yet stop short of proper capacity planning tooling or tooling that can accurately forecast future usage or place workloads together in an efficient manner. All CSP's provide a *lift-and-shift* capability, for example, moving a container or VM with the workload in its entirety. The responsibility to provision the correct size of target prior to a migration or cloud adoption lies with the consumer who requests resources.

The research context is to focus on capacity planning by:

1. In the first paper, we understand the approach to capacity planning by capturing the key metrics that identify resource consumption. This is important and relevant from on-premises complex database architectures with advanced features such as replication, clustering and pluggable databases, with or without containerisation.

2. In the first paper, we analyse the signal obtained from the metrics, so that we can understand how the architectures influence workloads. Understanding how workloads change depending on the task they are asked to fulfil, and accounting for these in advanced architectures, is paramount. This is often reflected in the signal

3. For any procurement of IT, a capacity planning exercise is performed and part of that exercise is an estimation of future resource consumption is. However, it is not well understood how the intricacies of complex usage patterns can influence future estimates, especially in systems that exhibit volatile data signals, which is discussed in detail in the second paper.

4. In the second paper we also propose that current techniques available do not scale when we are consolidating hundreds of database systems each with many metrics, thus a scaleable approach that leverages machine learning holds promise for providing an accurate forecast of future resource consumption at scale.

5. In the final paper, we propose that once the workloads are identified, how do we place them in the target architecture efficiently without compromising SLAs or clustered architectures. Current bin-packing algorithms do not account for advanced database features such as clustering or consolidated / isolated pluggable databases. This is a manual process taking expertise and time to accomplish.

1.9 Thesis Aims, Objectives and Contributions

Aim: To provide a firm foundation for capacity planning for databases with the aim of cloud adoption, including placing workloads in a cloud architecture without compromising advanced features such as clustering, replication (standby databases) or consolidation (pluggable databases) with the minimal resource wastage.

1.9.1 Objectives

- To identify the current footprint of the database including any advanced features such as Standby, Clustering and Pluggable databases by understanding the key metrics needed to account for advanced database workloads prior to cloud adoption.

- To perform a long-term capacity planning exercise (forecast) of what the future workload consumption will be, whether that is cloud, on-premisses, off-premises or non-cloud.

- To perform a short-term capacity planning exercise to predict when a workload will run out of resources to aid and prioritise workloads based on size or configuration prior to cloud adoption.

- To map and assign the advanced database workloads in a database cloud architecture (OCI) without compromising advanced database features such as Standby, Clustering and consolidated Pluggable configurations in such a way to maximise the target resources and their utilisation.

1.9.2 Empirical Evaluation

In terms of the empirical evaluation, the following contributions have been made:

- We report the results of an empirical analysis of the metrics for several representative workloads on diverse real-life configurations

- An investigation into the application of time series modelling techniques such as ARIMA and HES (defined in Section 3.3.1) to perform workflow predictions for OLTP and OLAP workloads.

- An empirical evaluation with controlled workloads at the database layers of the technological stack such as Database Instances that employ advanced database features.
- A demonstration of the applicability of the approach in real world workloads hosted by Oracle Advanced Customer Services.
- We identify the challenges and opportunities presented by advanced architectural features, when placing database workloads into complex cloud infrastructures from the empirical evaluation undertaken and the extra steps required prior to cloud adoption.
- We evaluate the algorithms in experiments and real world use cases that involve the placement of workloads into advanced target cloud architectures (OCI).

1.9.3 Contributions

The following contributions have been made

- We show in paper *'DBaaS Cloud Capacity Planning - Accounting for Dynamic RDBMS System that Employ Clustering and Standby Architectures'* in section 5.1, several contributions and lessons. A different approach to accounting for database architectures that employ advanced configurations is required as our experiments and analysis show.

    1. When Capacity Planning, one should obtain metrics at an *instance-by-instance* level and not for the global clustered database as a whole. The single most important reason for this is that a cluster can run in an uneven manner if QoS is applied, resulting in one node consuming more resource than another node in the cluster.
    2. Metrics need to be captured at different layers of the infrastructure in advanced configurations to avoid caching mechanisms which may skew true usage. For example, the storage layer employs caching to speed up IO intensive operations. Hypervisors and VMManagers assign CPU with some of the CPU being directed as support overhead, therefore the database may assume it has a full CPU core when in fact it has not.
    3. Operating System configuration has a profound effect when comparing one architecture to another as we observed in one experiment. For example,

Thread(s) per core influenced the execution of a OLTP workload on one VM to another resulting in increased efficiency. Accounting for operating system configurations is important when capacity planning, which may be overlooked.

4. Advanced database architectures such as Standby Databases present a completely different footprint to that of the live database and when capacity planning, one should not assume that a standby database will be a mirror reflection of resource consumption to the primary database as we show in our experiments. Also the way that standby databases operate means that certain metrics are not available, such as physical reads/writes, CPU and Memory, thus gathering metrics at the database layer is impractical. This results in the need to gather metrics at the host layer, which introduces a layer of complication should the host house more than one standby database.
5. In environments that employ advanced database architectures such as a clustering, if a workload running on one node fails over to other nodes in the cluster one should not assume that the properties of the composed workload being failed over will follow obviously from its constituents. Upon fail-over a new database footprint is created which will influence any capacity planning exercise.

- In our second piece of work titled *'Database Workload Capacity Planning using Time Series Analysis and Machine Learning'* in Section 5.2, we present techniques that are used for forecasting with a focus on short and long term capacity planning. Short-term capacity planning is the immediate usage and a forecast performed with a prediction over several days. Long-term capacity planning requires a forecast on data over many months with a prediction over many weeks. Several contributions and lessons where achieved were a novel approach was created.

  1. We are able to understand the complex data patterns exhibited by database architectures that employ advanced database features such as clustering and pluggable databases. We can identify how workloads can exhibit trend, seasonality and exogenous shocks that enable us to map them back to events in the system's behaviour
  2. We present a novel proposal for the self-selection and self-configuration of

forecasting models such as **S**easonal **A**utomated **R**egressional **I**ntegrated **M**oving **A**verages E**x**ogenous Variables (SARIMAX) and **H**olts **E**xponential **S**moothing (HES) extended to **T**rigonometric **S**easonality **B**ox-Cox **A**RMA **T**rend **S**easonal components (TBATS) for use with a given workload. We introduce Fourier Transforms on data that exhibit multiple seasonality (complex patterns) with the aim of increasing accuracy.

3. We leverage Supervised Machine Learning within the forecasting techniques that first learn the signal exhibited from the trace obtained from the metric data before performing the prediction. By utilising machine learning we reduce the amount of compute time taken to crunch the data and perform the prediction significantly from around 11-15mins without, Machine Learning, per metric to 1-2 mins with Supervised Machine Learning.
4. The Autocorrelate Function (ACF) and Partial Autocorrelate Functions (PACF) allow accounting of the time series into a stationary series that can be visualised through a correlogram. By automating this step we can significantly reduce the number of models per metric.
5. The work that we set out and cover in our the second paper, presents the novel approach to forecasting on volatile data exhibited from database architectures that employ advanced features. This work has resulted in a patent being filed through the Oracle Corporation in the United States and European Union.

- In our final piece of work titled *'Placement of Workloads from Advanced RDBMS architectures into Complex Cloud Infrastructure'* shown in Section 5.3, we tackle placement of workloads, which provided several contributions and lessons that were achieved resulting, in a novel approach being created.

  1. We identify the challenges and opportunities presented by advanced architectural features such as clustering and consolidated database environments employing pluggable databases, when placing workloads into complex cloud infrastructure such as Oracle Cloud Infrastructure (OCI).
  2. We present a new vector bin-packing algorithm for provisioning database workloads that takes into account fine-grained monitoring information and advanced architectures such as clustered databases that enable placement without compromising High Availability (HA) configurations.

3. We present an approach that identifies, once the workloads have been fitted, if further efficiencies can be obtained from the newly consolidated cloud workloads, advising if elastication is required.
4. We evaluate the algorithms in real world use cases of varying workload complexity and present the findings.
5. The work that we set out and cover in our the third paper resulted in a patent being filed through the Oracle Corporation in the United States and European Union.

1.10 Thesis Structure

This thesis is presented according to the guiding principles of a Journal Format. The core collection consists of published peer-reviewed papers and one manuscript submitted, which I have produced during my PhD studies: In all the first-authored publications, I contributed to the main ideas proposal, research development, research planning, literature review, writing, evaluation and analysis of the results. My supervisor, Norman Paton, also contributed to the ideas, proofread the papers and approved the results. To comply with the Journal Format policy, the self-contained papers are presented as they appear in print, with their respective abstracts, figures and references. Hence, there is a reasonable amount of repetition.

- **Title:** DBaaS Cloud Capacity Planning - Accounting for Dynamic RDBMS System that Employ Clustering and Standby Architectures

  Authors: Antony Higginson, Clive Bostock, Norman Paton, Suzanne Embury

  Published: In Proceedings of the 20th International Conference on Extending Database Technology (EDBT), Published by OpenProceedings,
  March 21-24 2017,
  Venice, Italy,
  Pages 687 - 699,
  ISBN: 978-3-89318-073-8,
  `https://10.5441/002/edbt.2017.01`.

- **Title:** Database Workload Capacity Planning using Time Series Analysis and Machine Learning

Authors: Antony S. Higginson, Mihaela Dediu, Octavian Arsene, Norman W. Paton, Suzanne M. Embury.

Published: In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, Published by ACM,
June 2020,
Pages 769–783,
https://doi.org/10.1145/3318464.3386140.

- **Title:** Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure

  Authors: Antony Higginson, Clive Bostock, Norman Paton, Suzanne Embury

  Published: In Proceedings of the 20th International Conference on Extending Database Technology (EDBT),

  Submitted to ACM,
  March 21-24 2017,
  Venice, Italy,
  Pages 687 - 699,
  ISBN: 978-3-89318-073-8,
  https://10.5441/002/edbt.2017.01.

1.11 Chapters

1.11.1 Chapter 2

In Chapter 2 we give a brief description of the historical journey to cloud from on-premises architecture and the differences, architecturally, of cloud. We discuss what makes the cloud proposition an attractive one to organisations, including the types of clouds organisations opt for and their distinctive configurations. We look at the challenges of cloud adoption that organisations face, including the steps required and why capacity planning is integral to cloud adoption. This chapter sets the scene for why the cloud proposition is one of inevitability rather than choice.

1.11.2 Chapter 3

In Chapter 3 we describe the experimental setup in detail that was used to execute, obtain, extract the metric data. We look at the available benchmarks that the industry use when comparing one hardware architecture to another and identifying or describing a workload. We describe the workloads in detail and the specific traits that differentiate one type of workload from another. This chapter sets the scene for the experiments we conduct to evaluate our work.

1.11.3 Chapter 4

Chapter 4 reviews the fundamental semantics of current techniques for capacity planning when accounting for databases and adopting cloud. The semantics refers to how to extract metric data from the correct level of the technological software stack such as the operating system, database and individual instances. We look at the existing techniques for forecasting and the various models such as time series analysis compared with other models such as stochastic and optimising solutions used today. When it comes to placement of the workloads we cover the various bin-packing algorithms such as First-fit decreasing (FFD) and optimal solutions.

1.11.4 Presentation of Published Work

In Chapter 5 we present a collection of published papers that covers specifics of the work from Chapter 4. These papers have been published in respected international conferences and form the bulk of the thesis. For each of the published works we also introduce the impact factor and journal ranking as covered by Resurchify [Met21].

1.11.5 Concluding Remarks

In Chapter 6 we provide an overall conclusion of the thesis commenting on the significance of the results from our experiments. We discuss limitations of the work and how future work can be undertaken in taking the research further.

1.11.6 Appendix

There are two appendices *A* and *B*. Appendix A is the license and permissions from the published pieces of work for use as part of this Thesis. Appendix B represents the

nuts and *bolts*, such as SQL queries, Python Libraries, OS, Database Versions and Products used in the experiments that supported the work.

Chapter 2

The Cloud Proposition - The Journey to Cloud Adoption

Invention comes in many forms and at many scales. The most radical and transformative of inventions are often those that empower others to unleash their creativity - to pursue their dreams.

Jeffrey P. Bezos, Founder
Amazon.com Inc

2.1 Historical Journey

Amazon Internet Solutions, started in 2006, described by Geoff Bezos, who was the CEO up until 2021, compared cloud to the utility company from the early 1900's. A factory needing electricity would build its own power plant; once the factories had the ability to get electrical energy from a utility, the demand for expensive private plants diminished. AWS had noticed that customers wanted to run Microsoft products on AWS. Microsoft, themselves had a conundrum too; in 2006 Windows Azure was created and Microsoft struck a deal with AWS in 2008 through Bob Muglia (Head of Servers and Tools Division at Microsoft). Muglia agreed to add Amazon to a program for companies that rent Microsoft tools to the their own customers and pay Microsoft at the end of each month. Although Amazon was not telling investors what they were

doing there was an indicator as the customer base of Amazon grew very quickly, notifying Microsoft of the potential for Cloud as a Business [Day16]. Arguably the dawn of the cloud arms race had started. Google and Microsoft launched competing services but Amazon was already capturing customers as they were first to launch. It wasn't until 2011 that Bob Muglia was removed for Satya Nadella (Future Microsoft CEO in 2014) did Microsoft truly push for Cloud Computing and the birth of Microsoft cloud (Azure). Google Cloud Platform (GCP) with its Google App Engine was launched in 2008 with the *modus operandi* of allowing developers to create web tools on google infrastructure. The goal being: *make it easier to get started and scale when the app has significant traffic and users*.

A common theme was beginning to surface and that was: the procurement of IT to satisfy business requirements efficiently and at scale had become slow and cumbersome, arguably because of N-Tier architectures. This resulted in a change of the status quo from the big traditional vendors and their on-premises software such as Microsoft, Oracle and IBM. Google [RGARHCDF20] have since become one of the dominant leaders in Cloud Computing along with Amazon. Not all vendors joined the arms race at the same time; Larry Ellison (Oracle CEO) was not an advocate of cloud to start with, and in 2009 dismissed cloud computing as a hype at first [LEwEZ09]. His criticism of some vendors stating their software is in a *'cloud'* when it was attached to a network, while true, arguably caused Oracle to be late in adopting cloud, which he was criticised for by the industry. Oracle have since become one of the dominant cloud vendors in recent years [RGARHCDF20]. Many commentators [Day16], [Til21] or analysts are now focusing on the battle or arms race that has surfaced between cloud vendors for cloud supremacy. Like every war, there is always collateral damage, which the customers feel. The affect on customers is the dreaded vendor *lock-in* that customers tried to avoid in the previous user license agreements of the 1980's, 1990's and early 2000's. Customers opting for multi or hybrid cloud solutions have created a new toxic mix of, arguably, the old problems of I.T. and Software development, but at a much faster pace.

Broadly speaking, most enterprises follow a System-Development Life Cycle (SDLC) that has evolved from a five stage process muted by Redack ([Rad05]) to a seven stage process as described by Langer in 2008 [Lan08]. The SDLC enables a newly developed project to transition into an operational one supporting or implementing a business function. This, never ending, cycle has evolved over the years and while the

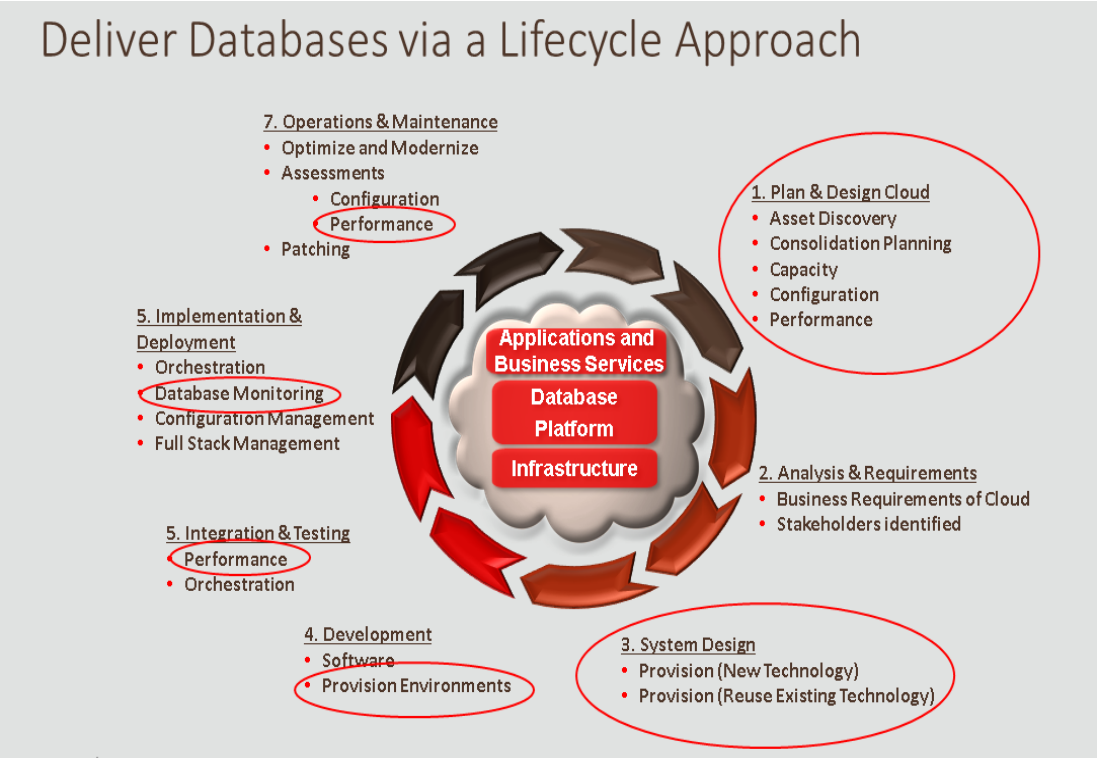


Figure 2.1: Software Development Lifecycle (SDLC) - Highlight focus

methods vary widely from *enterprise-to-enterprise* they have a common goal: to develop software as *cheaply, efficiently* and *effectively* as possible. Cloud aims to enable this goal with development methodologies such as *Agile* that tend to wind together these processes into a tight or rapidly repeating cycle. Waterfall type development methodologies tend to take each step in turn, where outputs from one become inputs to another. A high-level of the phases are as follows and are shown in Figure 2.1. In Figure 2.1 we adapt the SDLC where the work in this thesis is relevant as highlighted by the red ellipses. We discuss this is detail in Sections 1.9, 1.9.1 and 1.9.2 .

- Planning - Resource allocation (human and materials), capacity planning, project scheduling, cost estimation, provisioning

- System Analysis and Requirements - The business communicates with the IT teams to convey their requirements for new development and enhancement, usually involving stakeholders and subject matter experts.

- Systems Design - Architects and developers begin to design the software and architecture through established processes and frameworks such as The Open

Group of Architecture Framework (TOGAF). This may involve new technology or the promoting existing technology with the aim of employing standardisation.

- Development - This phase produces the software, and depending on the methodology employed, it could be *timed-boxed* sprints such as Agile or a single block of effort like the Waterfall method. The aim is to produce working software quickly

- Integration and Testing - Arguably the most import phase, this aims to test quality of software through unit, integration, performance and security testing.

- Implementation and Deployment - Usually this is a highly automated phase via continuous or application release tools.

- Operations and Maintenance - Software is continually monitored to ensure proper operation. Bugs and defects discovered in Production are reported and responded to, feeding back into the software development phase.

With the introduction of cloud and orchestration methodologies such as DevOps, some of the phases from SDLC are now highly automated. Achieving speed and efficiencies through environment replication, orchestration, effective monitoring and rapid deployment via environment provisioning, all result in a positive case for cloud adoption. Knowing *what, where* and by how *much*, arguably takes a prominent role.

However, simply adopting cloud is not as easy as one might believe because cloud has the added complexity of the infrastructure being remote or unknown, as it is accessed over a network *'somewhere'* and maintained by *'someone'*. Therefore, most adoptions of cloud infrastructure will involve some form of re-platform, upgrade, new architecture implementation and/or changing of financial models such as Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) into a new model of *pay-as-you-go* subscription based Cloud Expenditure (CLOUDEX). Software Vendors are, proverbially, asking their customers to not only move house but move country, adopt a new language and adhere to the new rules that implies. Cloud vendors have, however, gone some way to make adoption seem less problematic than perceived by introducing simple *lift-and-shift* models. The cloud arms race seems to be focusing of which vendor can migrate the greatest number of customers and their systems to their cloud platform quickest.

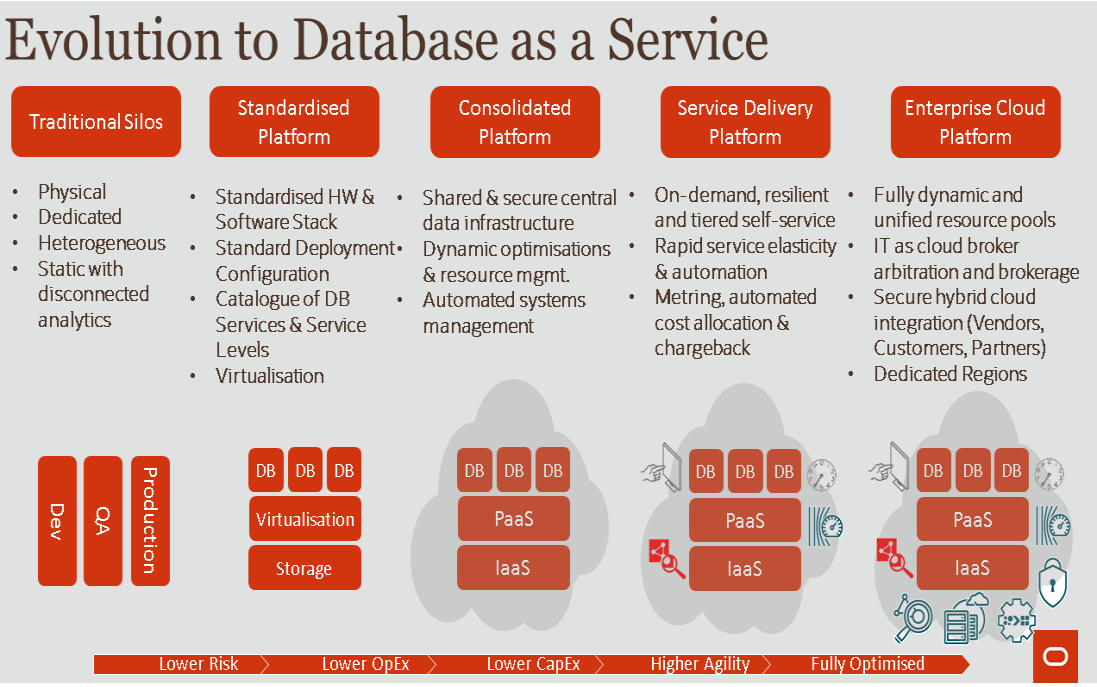


Figure 2.2: DBaaS evolution from traditional I.T.

2.2 Cloud Features and Market Forces

Traditionally, I.T. was consumed via data centres where applications or systems consisting of groups of applications interwoven or interoperating together were siloed. This silo'd approach allowed for separation between Dev, Test or Quality Assurance (Pre-production) and Live. Virtualisation allowed for the data centres and their silos to become more flexible through zones. For example a large physical piece of hardware (Server) could be virtualised into several servers in a zone to serve a particular group of applications, as shown in Figure 2.2. To protect data and, to a certain extent, create a business continuity plan, traditional failover, backup and recovery routines where then applied to these silos.

This created challenges for organisations that worked in this way specifically, it was expensive as the IT utilisation was directly funded from the business revenue via their financial models such as CAPEX or OPEX and servers were often under utilised. For example, consider a two-node cluster each node running at 48% utilisation. If one server fails then the traffic or users fail over to the single remaining node, which would increase to 98% utilisation. The moment the two-node cluster increases to 51% utilisation per node, then a third node had to be added, bringing down the total load. This complexity was often offset by introducing Quality of Service (QoS) that was assigned

to prioritise certain business functions *'in the event'* of failure, to the detriment of the organisation that argues that all functions are critical. This under utilisation of I.T. resources and the inflexibility of moving data from different silos created a low return of investment (RoI) and increased the total cost of ownership (TCO). Expanding I.T. in silos required bigger physical hardware, resulting in more servers, larger silos and more data centres. Organisations then had the added complexity of these data centres being close to their premises, to reduce latency issues.

Consider a small to medium sized enterprise (SME) such as a law firm with a few offices each with a team of solicitors. This added cost and complexity of employing a team of I.T. professionals to administer their I.T. could be the difference between success and failure of a business operating on tight margins. Cloud, as a concept, practically removed this worry that many SME's faced, one that Amazon arguably was first to grasp. AWS rented out the usage of Microsoft applications, removing the need to employ I.T. professionals or skill up software administrators because that risk, complexity and cost was shifted to AWS in the event of failure. As long as there was internet access and the use of Virtual Private Networks (VPNs), an SME could function *normally*.

Consider a larger organisation with its own data centre, working in a traditional siloed approach. The SDLC required that testing be done thoroughly to avoid outages yet because there was separation in silos this meant breaching networks to copy or restore data from backups. Elongated development times were often seen as restrictive to the development teams, with little or no *like-for-like* environments to develop against. The aim of the SDLC is to develop software as *cheaply, efficiently* and *effectively* as possible, yet the siloed approach created friction to this happening.

Cloud economics involves several facets but one of the biggest drivers is economies of scale. Economies of scale is an important one to understand, as with cloud it avoids the initial *'up-front'* investment often associated with a CAPEX financial model. In 2010 Microsoft produced a white paper specifically looking at the economics of cloud (Harms and Yamartino [HY10]). In their study they looked at Supply-side savings, Demand-side aggregation and Multi-tenancy efficiency, concluding there were three main drivers to economies of scale. These were larger data-centres can deploy computational resources at significantly lower cost than smaller ones; demand pooling improves the utilisation of these resources and multi-tenancy lowers application maintenance labour costs. Therefore knowing what slice or cloud to rent for how long is a requirement to fully maximise the resources, is a critical facet if cloud adoption is to

be successful or not.

2.3 Different Types of Clouds

2.3.1 Public Clouds

Public Clouds are the most common type of cloud computing environment. Microsoft Azure, Oracle Cloud Infrastructure, Google Cloud Platform and Amazon Web Services Cloud are all examples of a public cloud. In a public cloud you share the same Infrastructure, Storage, Network Devices and Compute with other organisations. Separation is managed through *'tenants'* and *'domains'* with a security layer that keeps the data secure, allowing users to manage their access via a web browser. Typically these types of configurations are used to provide web-based mail, online office applications or development and testing environments.

2.3.2 Private Clouds

A private cloud consists of computing resource that are exclusive to one business or organisation. A private cloud can be physically located locally such as on-premises of the organisation or remotely at the CSP. One of the key differences between a public and private cloud is that the cloud services and infrastructure are held within a private network therefore the cloud is dedicated solely to the organisation as shown in Figure 1.3. This type of cloud configuration allows for more flexibility, control or scalability, as the resources are not shared with other organisations.

2.3.3 Hybrid Clouds

Hybrid clouds are a combination of an organisation requiring computing resources that are shared and unshared. These types of configurations can be much more complicated especially with configurations requiring HA such as clustering and replication. Consider the example in Figure 2.3 provided as a Microsoft SQL Hybrid Server solution by Amazon AWS. It shows how database services can be shared between the data centre on-premises and remotely via Amazon Virtual Private Cloud. The connection is facilitated via a network connection and the creation of a *Distributed Availability Group* that spans both locations and configurations. These types of configurations are especially attractive for organisations that require advanced database features such as

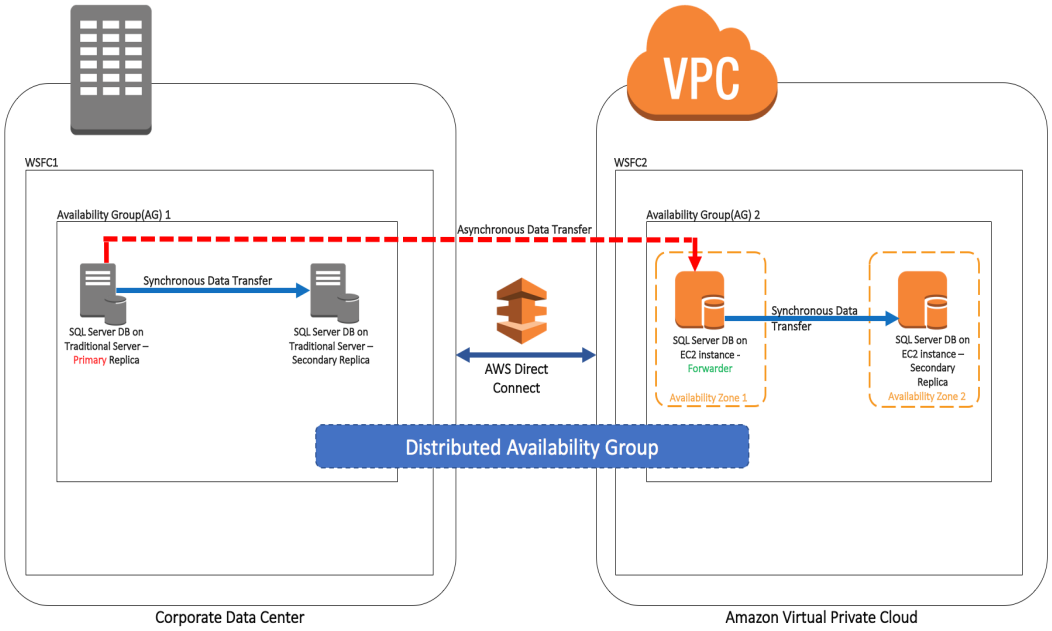


Figure 2.3: How to architect a hybrid Microsoft SQL Server solution using distributed availability groups [Siv18]

failover, clustering or replication. These types of systems are usually running core critical business functions.

There are several cloud service models described as IaaS, PaaS, DBaaS and SaaS shown in Figure 2.4. These service models are arguably increasing with the inclusion of Analytics and Machine Learning Layers or *serverless* as scientists struggle to classify services in the *anything-as-a-service* [DFZ+15]. NIST also do not classify DBaaS, however, for the purpose of the thesis, and given most software vendors offer different types of database services such as DataWarehousing, NoSQL and High Performance Database on top of *normal* database, we include DBaaS.

2.3.4 IaaS

Infrastructure as a service (IaaS) is the physical hardware needed to run essential computing services such as the Data Centre, Operations needed to maintain the assets within the data centre as well as the Physical Hardware, Servers, Storage, Networking and Security (Firewalls). IaaS allows the consumer to by-pass the cost and complexity of purchasing and maintaining the physical data centre infrastructure and physical

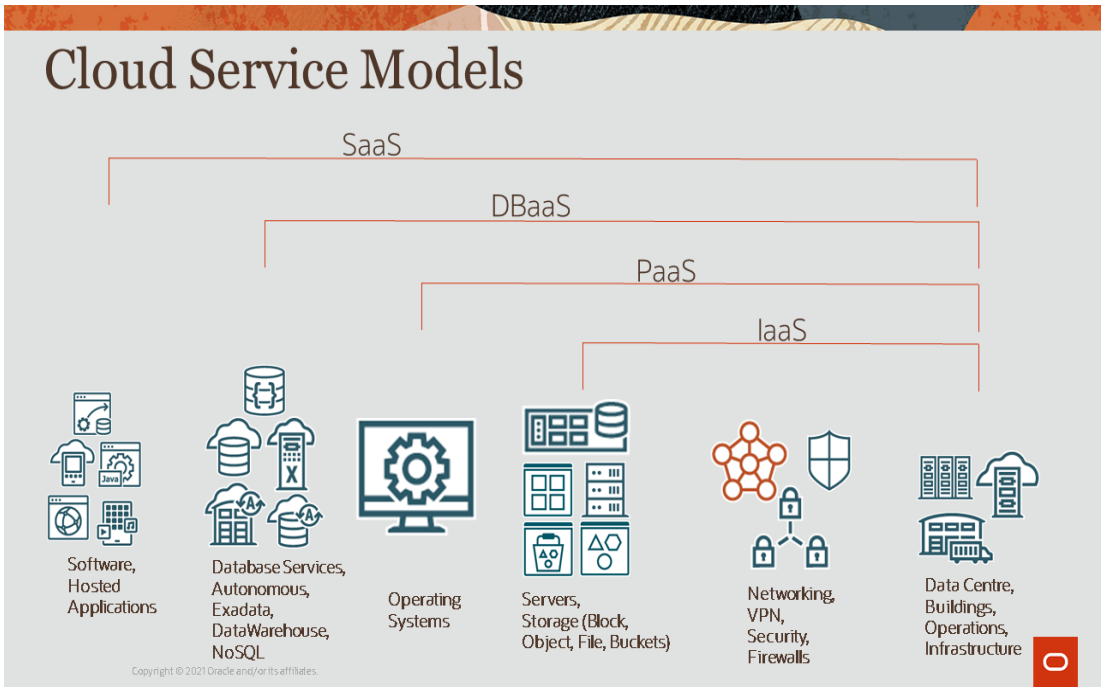


Figure 2.4: Cloud Service Models [Mar21]

servers. Each component of IaaS is offered as a service, for example, the storage services may encompass many different types of storage such as Volume, Object or Block. Different types of storages facilitate different types or volumes of files. For example, if database replication is employed the database logs may be stored in cheaper volume storage along with the database backups compared with pictures or videos for a streaming app that requires high speed block or object storage.

2.3.5 PaaS

Platform as a service (PaaS) is the operating systems, middleware and development tools organisations used to develop or deploy applications. Similar to IaaS, PaaS includes the servers, storage and networking with the aim of supporting the SDLC. For example, building, testing, deploying and updating applications using a development methodology such as Agile. PaaS allows organisations to avoid the expense of buying and maintaining software licenses or the underlying application infrastructure and middleware, containers or orchestration tools such as Kuberbetes, Docker etc. Typically the cloud consumer will manage the services and applications that the organisation develops with the CSP managing everything else.

2.3.6 DBaaS

Historically the Database Layer had been included in the PaaS layer, however given the plethora of different database services it has been separated for this thesis as its own cloud service type. Most cloud vendors have also taken this approach of separating DBaaS into its own service type in recent years [AWS21, Cor21m, Mic21, Goo21a]. Most applications that require the storing and retrieval of data require some form of database, for example if the data requires normalisation to serve financial information then a Relational Database Management System (RDBMS) is required. DBMS systems have evolved into some of the most advanced software platforms around and often incorporate replication, clustering, separation and consolidation (Pluggable databases) and security as they are often integral to the application and core business functions. There are specialist types of databases, depending on the application, for example in-memory, noSQL and Data Warehousing databases may focus on particular aspects of the application. A data Warehouse aggregates data to aid business intelligence type applications and is often busy in the evenings, crunching the data taken from daily business activity. Similarly, an OLTP database may require clustered high performance computing that satisfy Online Transaction Processing (OLTP) applications and run core business activity. NoSQL databases may satisfy systems that have large online stocks, where displaying the correct stock based on filters at speed is the requirement.

Database systems are often interwind with each other feeding information into one database from another. Applications may utilise different types of database depending on the workflow (tasks) the user(s) follows. For example, a user wishes to purchase online stock and may look at stock presented via a NoSQL database. When they chose to purchase an item and click through a checkout, a relational database is then accessed to ensure the correct financial information is created, accessed and updated. Similar to IaaS and PaaS, DBaaS allows the organisation to rent database services such as database storage in the form of a Schema, without the cost of user licenses, Infrastructure (IaaS), Operating systems or the highly skilled and talented professionals who maintain database environments.

2.3.7 SaaS

Software as a Service (SaaS) is a way of delivering applications over the internet; instead of installing and maintaining software, one simply accesses it via the internet. SaaS applications are commonly web-based such as internet-mail accessed via a thin

client. SaaS is attractive for customers who do not need to pay for perpetual licenses or have the need for administration teams to deploy and maintain the software. SaaS deployments are high in orchestration utilising tools such as Docker and Puppet via deployment pipelines to deploy code. In traditional N-tier architectures, software had to be replicated on web-servers, which presented a problem of scaling as the user activity (web activity) increased. Horizontal scaling is where the application is placed on multiple machines with vertical scaling being the increase of resources on existing machines. SaaS removes the cost and complexity of scaling as the consumer migrates to a *pays-as-you-go* model. If one needs to scale they simply request more resources, which was advocated by Google Cloud Platform for the development of web based apps.

2.4 Adopting Cloud

Cloud adoption for any organisation that already have advanced database architectures employed is non-trivial; it is complicated, and requires particular skills and expertise to plan, design, create, implement and migrate to cloud. A basic cloud installation for the purposes of quickly performing rudimentary development work is quick and easy as all cloud vendors provide a simple *pay-as-you-go* web-based functionality. Most organisations require or adhere to some form of cloud framework as Chang *et al* [CWW14] concluded in their work. They identified several risk factors, both strategic and operational, such as business, security and technical that are critical to cloud adoption. In an analysis of 11 Cloud Frameworks provided by the leading cloud vendors, they identified all have gaps when it comes to cloud adoption. One particular framework of relevance to capacity planning analysed was, the Performance Metrics Framework (PMF). Chang identified that PMF does not measure PaaS and SaaS and that this framework should included technical cost for IaaS, PaaS and SaaS, highlighting that there is a gap of capacity planning for cloud adoption. When it comes to adopting cloud for complex database architectures very little work has been completed as we cover in our first paper in section 5.1.

2.5 High-level Roadmap of a Cloud Implementation

Any cloud adoption involving a critical database with advanced features will invariably be treated as a migration project, simply because of the criticality of the business

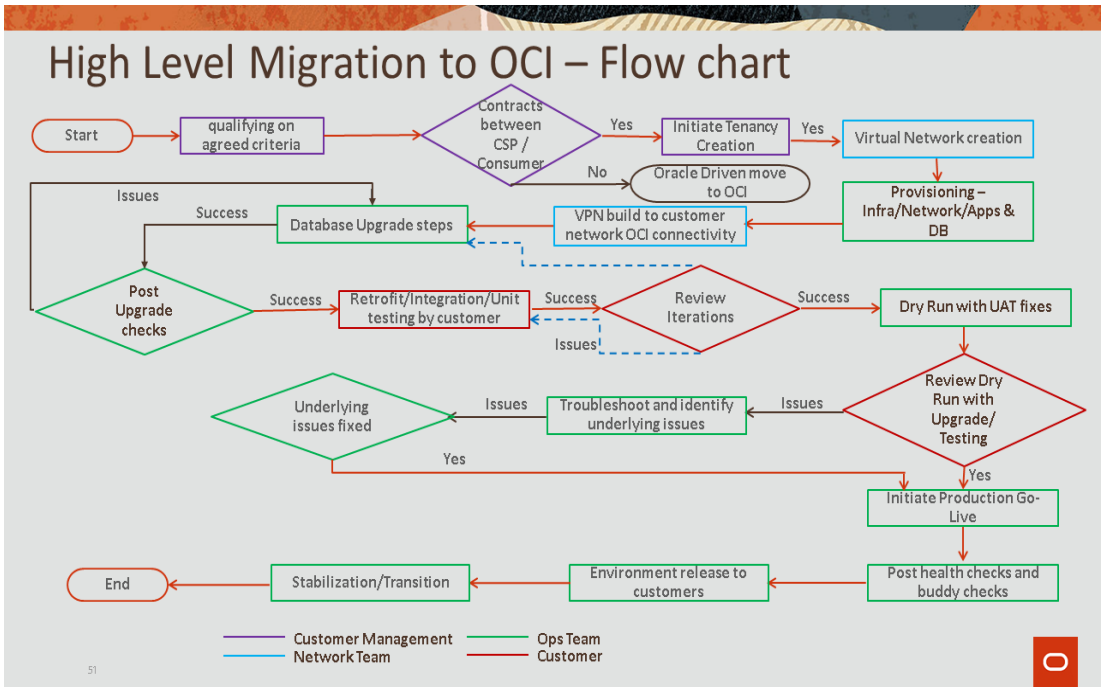


Figure 2.5: High Level DBaaS Cloud Adoption [Mar21]

function and because the database will reside on new architecture. There are four basic objectives when migrating a database to cloud: -

- Methods and Tools - List all the tools, methods and services available for database migration.
- Migration Strategy - Understand the parameters that help and design the right migration strategy.
- Advantages & Disadvantages - Articulate the trade-offs and advantages of the methods and tools, especially complexity, latency times, Recovery time objective (RTO).
- Demonstration of tooling - Building the reference Cloud architecture and testing the migration tooling to migrate database to cloud.

As each objective is expanded with further tasks identified, the level of complexity grows. Often, a cloud adoption exercise, involves the consulting services of the cloud vendor to aid the customer in achieving a successful database migration to cloud. In Figure 2.5 with Oracle Advanced Customer Services working with the customer, we show, at a high level how both parties work together to adopt cloud in complex database

environments. Often sharing tasks or responsibilities is not isolated to one party or the other. This diagram highlights, some of, the many critical tasks and functions required when adopting cloud.

2.6 Capacity Planning for Cloud Adoption

In the context of Section 2.5, Capacity Planning for Cloud Adoption has two major viewpoints to consider, which are the Consumer and Provider. These viewpoints are often treated as large projects because of their complexities. A user's perspective can be profoundly determined by SLAs whether those are throughput, availability or response times. A Provider's viewpoint can be determined by the ability to dynamically meet those SLAs. Menasce *et al* proposed an application that can take the SLAs, through the use of an optimisation algorithm, optimise the cloud vendor best suited to the application [MN09], viewing capacity planning as a optimisation problem. Others have also looked at capacity planning stochastically, utilising a wide range of models such as Markov Decision Chains or providing optimisation frameworks, but at the IaaS level of cloud as Ghosh *et al* show [GLX+14]. In our papers, as the work in this thesis will show, VMs or Physical Machines mask the true resource usage of the workload being executed in the databases layer (DBaaS). Therefore one must look at workloads, at a more granular detail, such as the specific database instance.

A major aspect of any migration or cloud adoption is understanding the impact of the newly adopted architecture on the resource consumption of the workload; the DML and SQL statements (Selects, Updates, Inserts and Deletes) being executed. Oracle databases work on a *cost-based* optimiser (CBO). The cost is a number obtained based on database statistics and calculations such as the number of physical I/O operations it thinks it will need to find the data. This is important as several factors can influence the CBO, for example disk performance (Solid State disk or Hard Disks Drives HDD), Data being stored in Memory, Indexes created on the database tables, Data composition stored in advanced configurations such as table partitioning, to name but a few. CPU also influences the CBO by means of the CPU *cycles per second* or the number of CPUs available. As we show in our first paper, one of the contributions made was the understanding of *threads per core* metric on a VM. This metric can be configured at a hypervisor level and can influence the execution of the CBO causing the efficiency of the workload execution to be positive or negative.

Capacity Planning at the moment is cumbersome and expert friendly, as it requires

talented professionals with a deep understanding of Database architecture and System Administration. For example, knowing how the internal database optimisers work, how to extract key performance and configuration data from the host or database and then interpreting those parameters into a meaningful value. Knowledge of benchmarks such as SPECInt (CPU) and TPC (Database Workloads) to enable or internally configure these metric values within the database, for comparison, is fundamental if one is to map N-Tier to a cloud architecture. Often this work is done via the use of spreadsheets, which build in complexity and are expert friendly.

2.6.1 Metric Data - Identifying Workloads from Metrics

Some metrics are only applicable to different tiers and therefore a further understanding of the system architecture is required. For example, a VM may have two applications consuming CPU yet when the measurement is taken it gives the value of CPU being consumed on the VM not indicating who or what application is consuming the CPU. This is a common problem when obtaining measurements from metrics on database servers. A further level of abstraction is required to interrogate the metrics further, by specifically looking at the database tier as well as the host. These metrics specifically look at Data Manipulation (DML) such as select, insert, updates and delete statements that are executed by the application. Databases that are configured to employ advanced features such as clustering and replication also behave differently because they may be in a different *state* such as recovery (Standby database) or have multiple nodes (Clustering). This is something we identify and tackle in the first paper in this thesis; how to account for these dynamic RDBMS systems and capture the correct metrics when different types of workloads are executed.

Obtaining the metric values is done via execution of a command(s) on the operating system or within the database, which provides the value at *that* particular time. To obtain a trace one must periodically execute the same command, for example, every 5, 10, 15 minutes or hourly. Execution of the commands also consumes resources so one must take a balanced approach of obtaining the correct metric at the right frequency to give enough data to perform an accurate analysis without placing the system under unnecessary strain. Obtaining the measurements between time periods to wide (daily) with cause a natural smoothing affect in the data, resulting in masking of the true values of the spikes, peaks and troughs. This can skew a prediction, something we identify in our second paper and also tackle in this thesis. Therefore obtaining the measurement from the metric at the correct frequency, capturing the traits that the system exhibits,

is crucial if one is to perform an accurate analysis of consumption with a view to performing a forecast of what the system requires prior to a migration. One can then map the system behaviours, for example surges in users logging on at particular times of the day, to the data signal obtained from the trace of metric data.

A workload is a collection of metrics and there have been several attempts to group together or identify and standardise workloads to answer particular questions when it comes to characterising I.T. systems. For example, throughput, as advocated by Zhan *et al* [ZZS+12] who specifically look at data centres that house High Volume Computing (HVC) and the different types of systems that serve business functions such as Online Transaction Processing (OLTP) and Data Warehousing (OLAP). This is a particular problem for customers with large data centres or cloud computing as they rely on using specific benchmarks such as Standard Performance Evaluation Corporation (SPEC) [Cor17c] and Transaction Performance Processing Council (TPC) [Cou21]. In this thesis we also rely on these standard benchmarks when capturing and obtaining measurements from metrics. Workloads are critical because they enable the formulation of a vector consisting of multiple metrics to create a *shape* of resource consumption. Given the complexity of cloud, as shown in Figure 1.3 this shape can be used to place a workload, in a target environment with other shapes. Metaphorically, similar to the tetris game of different shapes of resources being fitted together with the minimal amount of wastage, this is something we tackle in our final paper.

2.7 Database Metrics - Manually Obtaining Database Statistics

Within the Oracle database resides the Automatic Workload Repository (AWR) [Cor17b]. The AWR is a mechanism used to capture performance statistics namely:

- Wait events used to identify performance problems.
- Time model statistics indicating the amount of DB time associated with a process.
- Active Session History (ASH) statistics that capture specific statistics of a particular session or user.
- Some system and session statistics.

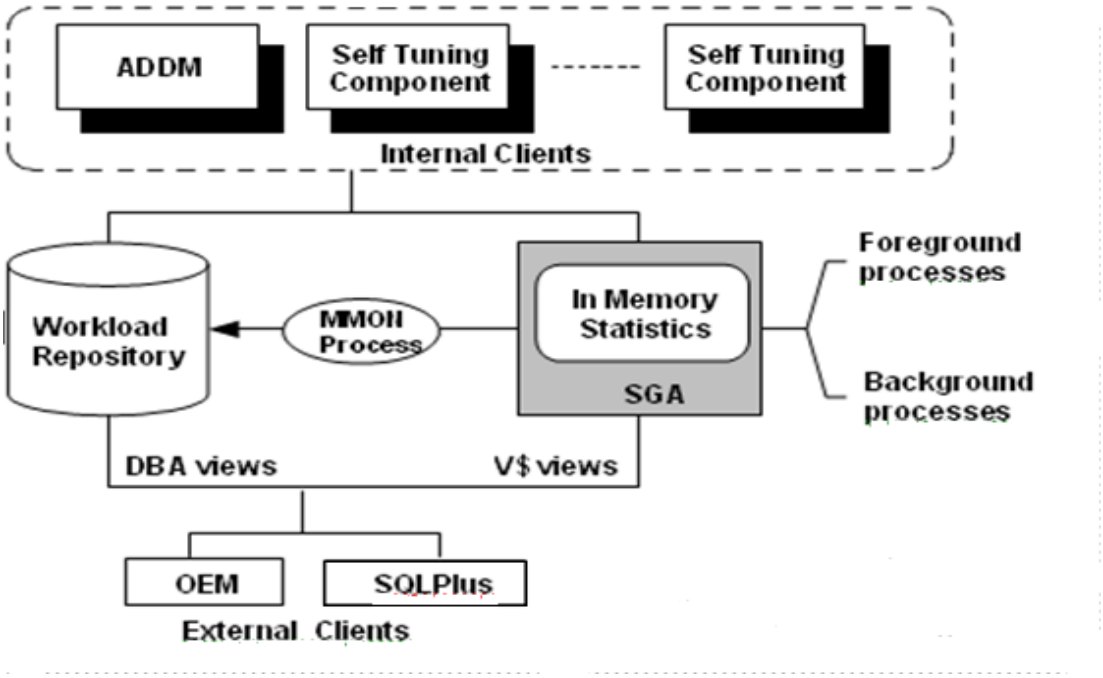


Figure 2.6: Basic Description of AWR Design for capturing Database Statistics [Dix12]

- Objects such as tables, sequences, views, indexes etc, usage statistics.
- Resource intensive SQL statements.

The repository that stores metric data also feeds into other tooling, for example tuning advisors such as the Automatic Database Diagnostic Monitor (ADDM) [Cor17a], that are used to aid diagnosis of performance problems. These tools aid the database administrator (DBA) in tuning or configuring the database to run at its most optimal. DBA's use these tools to provide insight into resource consumption and they can also be used to aid capacity planning. The AWR is configured to capture data hourly, and stores the data in a separate tablespace and schema. However, the length of time (retention period) is configurable, and DBA's often keep several months worth of data to aid performance problems that often degrade over time until a threshold is met that causes an outage. A detailed description has been provided in Appendix B.3.2. Extracting, understanding, interpreting and mapping key database metric data from AWR reports is cumbersome and time consuming, especially if there are many hundreds or thousands of databases.

DBA's often write custom SQL scripts that interrogate the AWR repository amongst other views within the database such as the internal database *DBA* and *V$* views. The

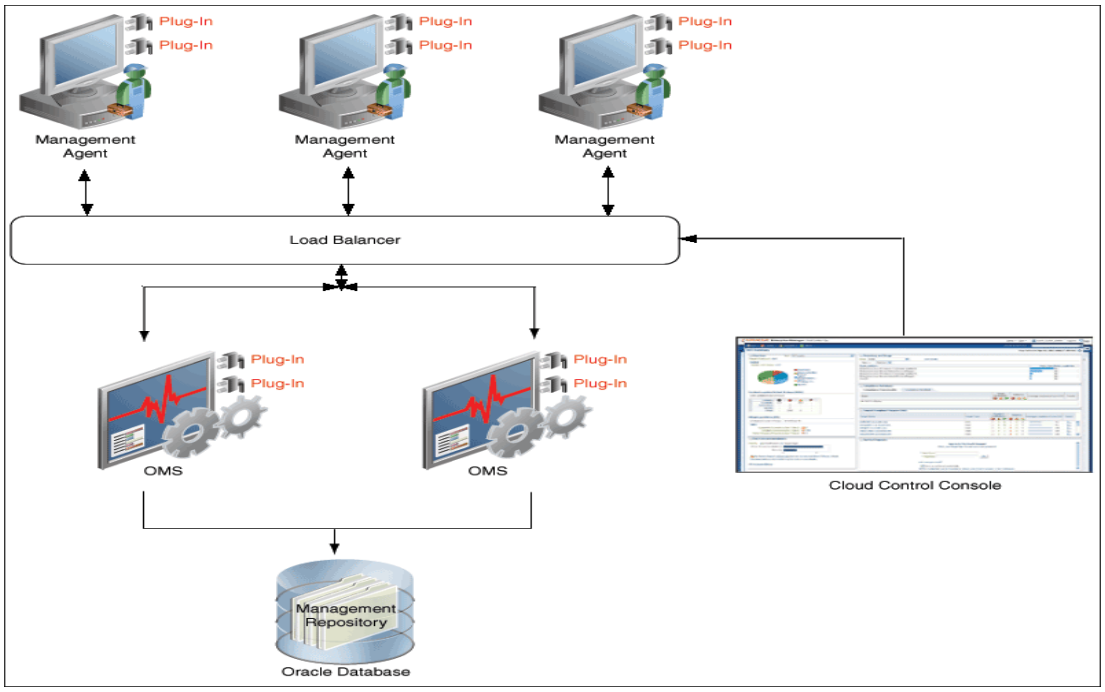


Figure 2.7: Enterprise Manager Cloud Control 12c Architecture [Cor21c]

AWR is part of a wider complicated mechanism for statistics gathering that DBA's interrogate to understanding the internal workings of the database, as shown in Figure 2.6. DBA's are talented professionals who also have knowledge of many areas of the technological stack such as Storage and Operating systems. Often they work in conjunction with other talented professionals such as System, Network or Storage Administrators and Developers to ensure that the application is designed to run at its most optimum.

2.7.1 Database Metrics - Intelligent Agent Based Software

Manual extraction of database metrics is cumbersome, time consuming and requires an elevated level of skill, however unless an agent based approach is adopted there is little an Administrator can do except to extract data manually. There are agent based management software packages available that can aid the Administrator in estates where there are many hundreds or thousands of databases. Intelligent agents are based on Monitor Analyse Plan and Execute (MAPE) as described by Arcaini *et al* [ARS15], and can execute commands periodically via a scheduler, storing those results in a central repository. Oracle Enterprise Manager (OEM) [Cor21h] is a licensed agent based management software that stores the results of said commands in

an database, as we show in Figure 2.7 (“Copyright Oracle and its affiliates. Used with permission”). Using an intelligent agent based application like OEM has its obvious advantages because the approach allows for the user to extract performance and configuration information from many databases centrally, rather than individually querying each database. However, not all information is extracted by the intelligent agent. For obscure or nuanced database parameters one must either specifically customise the agent or manually query the database AWR.

OEM is a licensed intelligent agent based application for managing databases. OEM has plug-ins that can do tasks such as stop and start databases, provision, perform backup and recoveries. Querying the central repository database and extracting the data centrally reduces the cumbersome and laborious effort of extracting data manually. We show how to do this in Appendix B.4.1 that shows a simple SQL query that can list a particular database and its unique identifier (GUID). That database GUID is then plugged into other queries to extract metric data into a suitable time series format that can be processed for forecasting and placement as described in Appendix B.

2.8 Conclusions

In this section we described how the cloud has now become an inevitability, and a natural progression from N-tier architecture to cloud is well under way. How to migrate these complex database architectures to cloud is fraught with difficulty and simple *lift-and-shift* will not suffice if one is to ensure minimum downtime and a successful migration. Extracting the data that is critical to understanding the system behaviours also requires a degree of expertise that is often *under-valued* or taken from granted. Once the data is extracted a high level of understanding is required to interpret the data into meaningful information that can aid decisions that greatly influence to design and procurement of cloud. In this thesis we aim to improve the understanding required to perform an accurate capacity planning exercise on complex architectures by; clearly defining the steps needed to performing an analysis, forecast and placement of workloads from complex database N-Tier architectures with the aim of a successful cloud adoption.

The focus of this thesis, and the work undertaken was several fold. Understanding the key configurations and architectural designs that can influence a workload such as standby and clustered configurations.

- Capture and extract meaningful metrics that reflect the true footing of a workload

into a meaningful trace.

- Analysing the trace of data to understand any patterns in the data that can influence a forecast is not trivial.
- Once the size of the workload is understood, place them together in a cloud is difficult if one is not to impact each of the workloads with another.
- A high level of automation is required with little human input to remove the manual exertion or mistakes that often accompany capacity planning exercises on large and complicated database estates.

Chapter 3

Related Work

Computer Science is no more about computers than astronomy is about telescopes.

Edsger W. Dijkstra

3.1 Capacity Planning

I.T. Capacity Planning is the art of providing enough resources to meet the demands of any SLAs, SLOs and QoS without bankrupting the organisation charged with meeting said SLAs, SLOs and QoS. Any form of I.T. procurement, provisioning or daily operations such as upgrades, migrations performance tuning, etc, involves an element of capacity planning. Capacity planning is aimed at answering consumption requirements *prior* to a deployment, while capacity management tends to deal with the management of resources *already* assigned. The aim of capacity planning, in practice, is to effectively reduce resource wastage and optimise resource consumption. Over provisioning can cause vast wastage to an organisation that may end up operating a data centre that is under utilised. Under provisioning can also give rise to significant consequences if outages occur due to applications or business functions failing due to a lack of resources. Thus a fine balance is required. Understanding Capacity Planning is complex because it involves many facets of resource consumption. In any form of cloud adoption, whether private, public or hybrid clouds, the basic concepts of capacity planning will remain the same as capacity planning for on-premises architecture with the exception that the cloud infrastructure configuration may be unknown.

The common theme with financial models (models where decisions are made on the basis of financial considerations) is that resource capacity planning, for example capturing resource usage and performing forecasts on future resource consumption, feeds into financial models that may determine or influence a particular cloud vendor or configuration in the cloud adoption process. This is also true for any framework, of which there are many ([SW13], [MCM13], [GDR15], [SMLB14]). Resource consumption and forecasting is an important facet of a framework, as it helps an organisation understand the shape of a new infrastructure. Frameworks often list *what* needs to be done rather than *how* to complete the task. The *how* is a much lower level of detail and expertise but there are still gaps in understanding the task of capturing metric consumption and to inform an accurate forecast. The proverbial of *'the right tools for the job'* is paramount when performing the *how*.

Most forms of capacity planning seem to focus on IaaS (VMs) and not the actual workload at a DBaaS or PaaS Layer ([KJA17], [SC16], [MK20]). In the literature, some works are multidisciplinary, for example, they are basing their work on a SLA approach utilising a machine learning technique to solve the conundrum they face. This allows for the work to be placed in multiple locations in Table 3.1. Yan *et al* [YLCL21] took a systemic approach in understanding how to optimize database workloads to achieve high-performance. Their analysis on available literature when looking at workload-aware performance tuning was separated into three main topics of *Workload Classification, Workload Forecasting* and *Workload-based Tuning*. They further separate these topics into categories and the literature of work done in these categories. I will be following this approach for this section as it clearly provides a systemic analysis to capacity planning as, shown in Table 3.1. In this table we have broken down the related work into several main topics and there is a reasonable amount of duplication from the papers and their related works. This table lists where researchers have used similar techniques, for example, Stochastic, Regression or Machine Learning techniques to answer the question of resource consumption.

3.1.1 Financial Models

Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) are viewed as financial models that can be a driver to cloud adoption as described by Kajiyama *et al* 2017 [KJA17] who conducted a survey into the decisions that affect cloud adoption. One of the biggest drivers for cloud adoption was the perceived benefit of *'Reducing spending on infrastructure'*, however they did not provide any models or frameworks

	Category	Comparable (Thesis)	Literature
Capacity Planning	Financial Models	Can aid reducing physical or projected costs	[SWJ+11] [KJA17] [PPN16]
	Frameworks	Can be leveraged as part of a framework but is not a framework	[SW13] [MCM13] [SMLB14] [GDR15]
	QoS, SLOs & SLAs	Can assist in placing workloads based on QoS, SLAs or SLAs but does not enforce	[SWJ+11] [Gro11] [KL12] [NXYJ17] [CBL+17] [BBB+17] [HBS+16] [AKY10] [PAT+20] [CMRB14]
Workload Classification	Benchmarks	Utilises existing benchmarks does not create new benchmarks	[Cou21] [HBS+16] [MB13]
	On-Premises	Works with workloads from architecture	[Cou21] [Cor17c] [MB13] [GKD+09] [KCK+13] [GRCK07]
	Cloud (IaaS, PaaS, DBaaS)	Works with workloads from any architecture or platform from any vendor	[XLZ+21] [AETEK13] [BBB+17] [GLX+13] [HBS+16] [PAT+20] [SMCFM18] [YQR+12] [MK20] [AbMl20] [MB13] [MKB+12]
Workload Forecasting	Time-Series	Extends and Introduces new techniques to deal with complex data structures	[XLZ+21] [Gro11] [CBL+17] [MKB+12] [Sko19] [CMRB14] [SMCFM18] [MK20]
	Machine Learning	Leverages Supervised Learning	[AETEK13] [GKD+09]
	Stochastic Processes	Not Applicable	[GLX+13] [AKY10] [PAT+20] [KCK+13] [MK20]
	Other	Not Applicable	[KL12] [BBB+17] [HS19]
Workload Placement	Bin Packing	Extends FFD to place complex RDBMS Workloads from complex architectures	[GRCK07] [YQR+12] [AbMl20] [DKJ11] [ACKS13] [PS21] [KGJ+21]
	Optimisation	Could aid evaluation after placement	[KL12] [NXYJ17] [BBB+17] [GLX+13] [GKD+09] [ZMPC18] [HS19]

Table 3.1: An Overview of Main Research Categories and Literature on Database Capacity Planning

to achieve cloud adoption. Shang *et al* 2011 [SWJ+11] seek to establish which cloud is *value-for-money*. They took the approach that based on an SLA performance they could derive if a cloud was financially viable. Pantelić *et al* 2016 [PPN16] used Multi-Attribute Utility Theory (MAUT) and created criteria with *weights*. They determined that *'The Monthly average price'* ranked second in importance behind Data Security in a ratio scale of 1-10. They then placed each cloud *as-a-service* in a hierarchy of importance. They concluded that using a MAUT method is a valid approach to select which cloud service is favourable, suggesting that the financial factors are not the sole drivers to cloud adoption but an important facet.

3.1.2 Frameworks

Sabharwal and Wali [SW13] concluded that capacity planning can actually be broken out into different sub-sections such as *capacity-in-use, redundant capacity* and *standby capacity* with the aim of answering key questions of Performance Requirements, Business Criticality and Future Growth. They surmise that the key goals of capacity planning are the reduction and efficient utilisation of resources, workload management and SLAs, Controlling VM Sprawl or Automatic Failures and Forecasting future growth. However, they stop short of providing solutions to answer these questions but provide a framework detailing a plan that can help to answer these challenges. Providing a framework to structure an approach is something addressed by Mozafari *et al* [MCM13] and Shari *et al* [SMLB14] using techniques such as MCDM (Multi Decision Criteria Decision Making) by weighing up attributes of a particular database by their importance in helping choose the right cloud. Gangwar *et al* 2015 [GDR15], looked at two frameworks called the Technology Acceptance Model (TAM) and Technology Organisational Environment (TOE). In their work, they identify several limitations, for example, TAM measures perceived adoption and self-reports on future behaviour rather than measuring of actual behaviour. TOE is a taxonomy for categorizing variables that make up the framework. For Cloud computing they proposed extending both TAM and TOE, integrating them together to create a single model given the complexities cloud brings. They specifically refer to capacity as not only physical resources but organisational capacity to adopt change that cloud brings.

3.1.3 Quality of Service (QoS), Service Level Agreements (SLAs) and Service Level Objectives (SLOs)

SLAs are paramount for critical systems that run 24*7. Most database features enforce redundancy to help meet critical SLAs and SLOs, for example, through clustering or standby databases. NEC Data Management Labs [Gro11] identified that one size does not fit all, and that customer SLAs and profit should be the main metrics for system management or optimisation. They created a piecewise linear function model based on profit and SLAs and applied this to individual SQL queries being executed. Kouki *et al* 2012 [KL12] identified that from a SaaS perspective of a Cloud Service Provider, to address how to maximise revenues without compromising QoS, SLAs or SLOs of their users. In their work they took a queuing theory approach to requests, and using Mean Value Analysis (MVA) they evaluate the performance of the request by assigning a label (Gold, Silver or Bronze). From this assignment they can capacity plan the resources needed to satisfy requirements, however, they do not place a workload. Noreikis *et al* 2017 [NXYJ17] took a novel approach to capacity planning based on QoS, focusing on hierarchical or edge clouds with Augmented and Virtual Reality applications. They determined QoS from a response delay of requests and what the demand in resources (CPU etc) will be with the aim of prioritising the task. Working at scale seems to be a problem with capacity planning based on SLAs, SLOs and QoS as different systems require bespoke solutions. From a PaaS or DBaaS perspective this could be very challenging for a CSP that has many customers with the responsibility of their systems. Carvalho *et al* 2017 [CBL$^+$17] extended their previous work of capacity planning based on QoS and SLAs but at an IaaS level (VMs). They prioritise requests and assign them to an available VM. This most recent work takes multi-dimensional resources and not just CPD (Earlier work). Other works such as Hwang *et al* 2016 [HBS$^+$16], Boukhelef *et al* [BBB$^+$17], Andrzejak *et al* 2010 [AKY10], Pereira *et al* 2020 [PAT$^+$20] and Calheiros *et al* 2014 [CMRB14] also provide solutions specifically focusing on SLAs, SLOs and QoS. Oracle Advanced Customer Services, in the *managed service* offerings, see SLOs, SLAs and QoS as critical drivers. Therefore being able to effectively capacity plan, short and long term, is critical to ensuring delivery of a high quality service.

3.2 Workload Classification

Classifying a workload is repetitive, that is to say a workload as a shape of resources can be obtained at each of the SaaS, DBaaS, PaaS and IaaS layers. One difficulty is knowing what to categorise the workload (OLTP, OLAP, DataMart) as and at what layer (IaaS, PaaS or DBaaS) if one is to enable successful SLAs, SLOs or QoS? Different types of workloads (OLTP, OLAP, DataMart) behave differently at different times and therefore depending on the type of workload will influence the SLAs, SLOs or QoS. The number of workloads can be expanded to any defined type. In this thesis we only used OLTP, OLAP and DataMart to keep the work simple. Capacity planning for DBaaS, must account for resource consumption of a type of workload that utilises advanced database features otherwise this gap is not accounted for. In our first paper we specifically tackle this gap where an OLTP DBMS workload behaves differently in a standby configuration, identifying the challenges of capacity planning these advanced configurations. It could be argued that it would be naive if one did not capacity plan for RDBMS systems that utilise these features or suggest that these features would not be available in the cloud (lift and shift capability), given users often run critical applications in a dual configuration while a cloud adoption takes place, especially if the database is very large or critical. For example, the RDBMS is many TB in size and/or has a *tight* SLAs, SLOs and QoS assigned to the system. Oracle Cloud Infrastructure has a product called Goldengate [Cor21i] that is a data mesh platform for replicating data in real-time between databases of different configurations for the purpose of redundancy and aid these types of issues. There is also an Oracle product called Data Guard for maintaining standby databases with the aim of addressing Recovery Time Objectives (RTO) and minimise data loss through recovery point objectives (RPO). Other vendors such as Microsoft also provide standby database capability in Azure as does GCP provide tooling in the form of Data Transfer and Data Mesh products. Knowing how and what techniques are available to help identify, classify and label a workload is critical, if one is to understand the nuances, characteristics and implications posed by the workload if it is to be executed in a cloud configuration.

3.2.1 Benchmarks

A characteristic of a workload can be assessed and replicated with the use of a Benchmark and there are several depending on what is deemed important. The challenge arises when one tries to compare an *apple* with a *pear* with the only surmise

being that they are fruit. A reputable benchmark can help address this problem. For example, system throughput that focuses on databases can be identified with the use of The Transaction Processing Performance Council (TPC) [Cou21], which, produces benchmarks based on the popular types of systems such as transactions processing (OLTP - TPC-C, TPC-E), Data Marts (TPC-H), or Big Data and Data Warehouse (OLAP TPCx-HS, TPCx-BB). These types of benchmark work on database requests and measure the throughput, providing a result that users can then use to measure performance. Different business tasks display different behaviours; for example, an OLTP workload tends to be short, concise transactions compared with an OLAP type workload that may have complicated tasks on larger data sets, often utilising machine learning algorithms. Another challenge is accounting for advanced features. TPC does provide clustering benchmarks however, it does not cover standby databases or consolidated environments such as Pluggable databases, which are required for Maximum availability architecture. TPC only focuses on throughput of workloads through particular configurations of RDBMS.

Another characteristic of a workload that is deemed critical and important to capacity planning is CPU. Standard Performance Evaluation Corporation (SPEC) is another suit of benchmarks with one of particular importance, which is CPU Integer. SPEC CPU 2017 [Cor17c] is organised into four main suites being; CPU Speed Integer 2017, 2017 Floating point, CPU 2017 Rate and CPU 2017 Rate Floating point, with optional metric for measuring energy consumption. This benchmark acts like a common denominator in comparing chip architecture by the number of SPECInts used when a particular workload (OLTP, OLAP, DataMart) are executed. This benchmark enables the user to determine if the CPU architecture on the cloud architecture is comparable to chip architecture from on-premises architecture and is the benchmark we use throughout this thesis.

Hwang *et al* 2016 [HBS+16] specifically looked at cloud benchmarks such as TPC [Cou21], Hi-Bench (Intel) [HHD+10], BenchCloud, CloudSuite [FAK+12] (Academia tools) and YCSB (Yahoo) [CST+10]. They categorised metrics for evaluating clouds such as Performance, Capability and Metrics with the aim of creating models capable of answering scaling problems such as scaling out, scaling up and elasticity at cloud (IaaS, PaaS and SaaS), in the absence of a benchmark that covers all three layers. They propose extensions to the these benchmarks using their approach. Using the right benchmark is critical to capacity planning as it allows comparable comparisons to be made depending on the type of workload executed. A user can determine if

the cloud architecture and/or its configuration will meet the demand of the workload, SLAs, SLOs and QoS.

3.2.2 On-Premises

Forecasting on-premises takes the form of either predicting the requirements of a data centre as Gmach *et al* 2007 [GRCK07], who used linear regression techniques (ARMA) to identify trends and use a bin packing algorithm to group VMs together in pools. Kraft *et al*, 2013, [KCK+13], also looked at particular storage metrics (such as IO) in the context of data centres. Focusing on how workloads influence each other. At an individual machine such as scaling out or scaling up as described by Ganapathi *et al* 2009 [GKD+09] focused on ML on-premises and is discussed in the machine learning Section 3.3.2.

3.2.3 Cloud (IaaS, PaaS, DBaaS)

Moussa and Badir 2013 [MB13] explained that the TPC-H and TPC-DS benchmarks are not designed for Data Warehouses in the cloud, further adding to the problem of developing and evaluating models. They identified what requirements are needed to create a cloud benchmark. Most research has focused on workloads at an IaaS level, which is mainly attributed to Virtual Machines (VMs). Mahambre and Chafe [MKB+12] look at the workload of a Virtual Machine to create patterns of workloads to understand how resources are being utilised, analysing the actual query being executed to predict if and when it is likely to exhaust the resources available. They use time-series analysis to form a prediction but do not disclose what particular models they use or how they deal with repeating patterns such as ARIMA or TBATS (Discussed in section 3.3.1). Ali-Eldin *et al* 2013 [AETEK13] focused on Amazon and Azure clouds through a Workload Analysis and Classification tool (WAC), with the aim of being able to assign workloads for candidacy to elasticise (Cloud burst) via controllers. A prediction if formed on the resource controllers and it is these controllers that then add additional resources to where the workload is being executed. They use a variety of techniques to do so, such as machine learning leveraging K-Nearest-Neighbours (KNN) and Support Vector Machines (SVMs) to classify the workload. When a new unknown workload arrives into their WAC tool a majority vote is taken with more *weight* being attributed to the nearest neighbours and then a class is assigned to the workload. This focuses on more capacity management and short term capacity

planning in the form of cloud bursting to manage sudden spikes of a workload once the workload is in the cloud.

Ghosh *et al* 2014 [GLX+13] focused on IaaS by assigning a cost to a VM workload, using stochastic techniques such as a stochastic reward net (SRN) and then based on probability, capacity plan the future resource requirements of the VM. Highlighting the need for specific solutions at the DBaaS and SaaS or actual database layer of the technological stack. Boukhelef *et al* 2017 [BBB+17] specifically looked at IO workloads in DBaaS clouds for hybrid storage for database objects (logical tables, etc,). They focussed on the Service Level Objective (SLO), and by solving an optimisation problem, place data based on the IO Speeds. Recently, Xia *et al* 2021 [XLZ+21] proposed a classification framework that classifies a workload as being *stable, peak* or *trough*, and then through a combination of statistical time series techniques are able to segment the time series to form a prediction. The aim of achieving long term cloud forecasting, however, it is not fully addressed as they have had challenges with volatile signals, hence the segmentation of the time series into periods. There seems to be a notable gap of accurately accounting and forecasting the actual database workload taking into consideration advanced database features such as Clustering, Standby databases or Pluggable (consolidated) databases on cloud (DBaaS, PaaS) unless it is a VM at the IaaS layer.

3.3 Workload Forecasting

Forecasting is a well understood problem and has been extensively used in econometrics to build models that form predictions. Most current work on cloud modelling focuses on VMs at an IaaS level. One of the key issues with modelling at this level is that it bundles several things together, for example CPU is taken as the sum of CPU used at a VM rather, than a VM may have a database running that is consuming CPU amongst other small tasks. As we show in our first paper, VMs mask the true usage of a database workload and it is possible to have multiple databases running in a VM or multiple pluggable (consolidated) instances running on a *master* container instance running in a VM. therefore, there is no easy way to determine where the CPU is being consumed when one forecasts at the VM level, without risking *over-provisioning* by including resources that are not necessarily being used. Performing a prediction of a workload in clouds is a relatively new topic, but essential from both sides of the relationship, whether that is the CSP that provisions the resources or the user who

consumes the resources. From our earlier work, in the paper Section 5.1, we show that extracting key metrics from advanced configurations over periods of time, several things became apparent. The data signal exhibits trend, patterns can be identified via peaks and troughs, and large spikes are shown when external activities are executed such as a backups, data aggregations. Therefore, a forecasting technique that can model accurately with the ability to take into consideration these traits is required.

3.3.1 Time-Series Analysis

Time-Series Analysis is a family of techniques that is widely used to perform forecasts, which is used extensively in Economics, Sales, Budgeting and Market Trading, to name a few. **A**uto-**R**egressive **M**oving **A**verage (ARMA) later expanded to **A**uto-**R**egressive **I**ntegrated **M**oving **A**verage (ARIMA), has been evolved further to incorporate **S**easonal variations (SARIMA) and finally, where we are today SARIMAX, which handles Exogenous variables or shocks. ARIMA is a class of models that capture the subtle structures held within the data signal. It was developed by Norbert Wiener *et al* in the 1930's and 1940's [W$^+$64]. Statisticians George Box and Gwilym Jenkins further developed these models for Business and Economic data in the 1970's, hence the name Box-Jenkins [BJRL15]. Time Series analysis is well suited to forecasting computational metric data because metric data exhibits complex structures such as trend, reoccurring patterns (Seasons) and/or complex seasons (seasons within seasons). Time series models are typically characterised by their parameters. For ARIMA, the parameters are as follows *p,d,q* and for seasonal components additional parameters are added *P,D,Q* and *F* to account for frequency thus creating the model *(p,d,q,P,D,Q),24*, where 24 would be a frequency of 24 hours. We explain the importance of these parameters in our second paper in Section 5.2. Skorupa, 2019 [Sko19] suggested that time series exhibiting multiple seasonality [Sko19], can be dealt with using the **T**rigonometric seasonality **B**ox-Cox transformation **A**RIMA errors **T**rend **S**easonal Components (TBATS) method, which uses a Fourier series to reduce the length of period or number of observations needed to perform a forecast. This is discussed in detail in our paper in Section 5.

Several academics have used Time Series Analysis to perform forecasting on computational workloads, however they tend to focus at the IaaS layer on VMs. Calheiros *et al* 2014 [CMRB14] did an in-depth analysis of the ARIMA technique, again focusing on the VM as a workload but for QoS and solving the problem of dynamic provisioning that CSPs currently experience. They predict the workload behaviour

and feed this into a queuing model for calculating the VM provisioning. They run ARIMA models against Web VMs at an hourly granularity, with data points taken over a month. Sousa *et al* [SMCFM18] looked at the MySQL database cluster in an Amazon EC2 cloud and used ARIMA to predict the workload. However, the environment is a simple one unlike our environment: an Oracle Exadata database cluster running several TPC workloads. Our aim is to evaluate enterprise workloads, and as our experiments will show we have evolved the time series technique to encompass ARIMA with Exogenous and Fourier terms to understand the complex structures within the time series data and make predictions

3.3.2 Machine Learning

As discussed in detail in our second paper titled *"Database Workload Capacity Planning using Time Series Analysis and Machine Learning"* Supervised Machine Learning is able to assist forecasting for several reasons, for example, taking historical data with known relationships between observations over historical time frequency such as hourly, daily or monthly, and the output is a numerical value (prediction) of what we think the future resource consumption of a metric is. Machine Learning can *learn* the historical data points and those traits exhibited by the data signal and lends itself well to regression problems. Also from our own perspective we wanted to use Machine Learning to help reduce the time to crunch the data in producing the forecasting model and address the problem of scale, i.e. crunching multiple models, metrics from many hundreds, if not thousands of customers each running many clustered or single databases. Here we consider a simple two-node clustered database and one metric (CPU). As we discuss in our paper, we measure data up to 30 lags. A time Lag is an interval of time between one event and another related event that happens after it. For example, a CPU data point taken hourly with each Lag relating to one hour (Lag=1) and the correlation between the values that are one time period apart. If we assume 30 Lags then we are considering the correlation between values that are 30 hours apart. ARIMA, SARIMA and SARIMAX models have many important parameters that make up the model, which we describe in detail in the second paper, in Section 4.1 titled *"Models"*. We also noticed that the time to crunch the data for an individual model was several minutes, therefore we looked to Machine Learning as a way to reduce this *elapsed* processing time of not having to crunch all the data each time the model is executed. The model only has to be executed on newly added data points, which is again discussed in detail in our second paper.

Using Machine Learning to aid the forming of a prediction is a relatively new idea, especially in cloud computing. Ganapathi *et al* 2009 [GKD+09] performed an empirical analysis of the different types of learning engines to tackle regression problems in a database optimiser for an IO metric. They looked at regression, clustering, component, canonical and correlation analysis, using supervised machine learning to train against the historical data. They concluded that Kernel Canonical Correlation Analysis (KCCA) was favourable, using nearest neighbours to form a prediction. They do not elaborate on how they deal with seasons, trend or external shocks (Exogenous). Tang *et al* 2019, [TYP19] used ensemble deep learning based on ARIMA to form predictions on Electrical loads as an important aspect of power system planning and operation. In this study they had issues such as non-linear patterns, external factors and volatile signals exhibiting seasons, which is interesting when compared with the data signal of an OLTP database instance for IO. An IO data signal is also influenced by external factors such as backups and data aggregations or loads that can cause an abnormal IO spike. If these loads are executed frequently this can create a season that ARIMA type models are best suited to deal with. Most recently, Fadda *et al* 2021 [FFPB21] have leveraged Machine Learning and ARIMA models to help answer the question of Tactical Capacity Planning (TCP) in logistics relating to delivery services and their demand. They created a Decision Support system to capture demand of daily orders taken from a city demographic, forecasting the capacity of the next time period based on the previous time period. This recent work reinforces our claim from 2020, that Machine Learning and time-series analysis models such as ARIMA modelling can be coupled together, performing forecasts to many facets of industry. Our aim is provide an accurate and efficient on the correct type of signal that is capable of being able to scale on hundreds of metrics for many customers important systems.

3.3.3 Stochastic Processes

Some academics have looked to stochastic processes to address capacity planning in clouds or I.T. in general. Stochastic processes take a collection of random variables as they evolve or are indexed in time. Arguably, capacity planning and resource consumption falls into this category as variables such as resource metrics or attributes relating to an SLA, SLO or QoS. These attributes and metrics have a strong relation to time in the form of historical observations to answer future consumption. The strong emphasis is to predict behaviour. Stochastic models have the advantage of being less

resource intensive in the calculations that form the prediction compared with time-series analysis that have very complex models. Stochastic models tend to *lean* on probability and confidence; for example a Markov chain represents different states that a system may follow with the transitions between the states indicating the occurrence of events such as the arrival of data from one location to another.

Andrzejak *et al* 2010 [AKY10] created a probabilistic model that enabled a user to optimise attributes associated to an SLA such as cost, performance and reliability. The model encapsulated parameters that a user could *tweak* in order to tune the SLA which focussed on spot instances that are idle and could be used to reduce wastage. Ghosh *et al* 2014 [GLX+13] introduced stochastic reward nets (SRN), similar to a Markov chain, based on sub-models that monitor resource pools. They categorised the pools as hot, warm and cold; the sub-models would interact with each other by assigning a cost, which relates a status to a physical machine. A probability calculation is performed on the Physical Machines to determine the likelihood of failure with the aim of moving the VM to a resource pool. More recently, Pereira *et al* 2020 [PAT+20] used stochastic techniques such as continuous Markov chains to model performance in cloud (fog) web servers. In this study they looked at the VM and potential bottlenecks occurring in a VM that have a detrimental affect on performance such as the network throughput. The Markov chains were used to identify the relationship between these components in the system and the probabilistic changes in state.

There are subtle differences between time series and stochastic processes even though it could be argued both techniques are somewhat related and arrive at the same outcome when performing a prediction on variables over time as highlighted by previous work. In stochastic processes they are random variables ordered in time. For example a random workload appears at a particular time and depending on some values, something is to be done with it. For example, add more resource or move the workload to somewhere where there is available resource, and based on probability the result of the workload being moved or having resource added will change for the better or worse (decision that could be aided by ML), which is a perfectly valid solution for that particular type of problem. In a time series, the *time* is indexed by integers continuously and discrete; discrete meaning, one value per hour. Which, is why I chose to focus on time series analysis as the metrics that make up a workload (CPU, Memory, IO) will be continuously indexed by a value in a sequence in time for as long as the workload is active. It is not a random variable per se, except that it may have random events exhibited in the data signal in the form of peaks, troughs and spikes, that may

have a pattern to them, that need to be accounted for.

3.4 Workload Placement

Placement of a workloads, when broken down, is multifaceted and inter-related. That is to say, all parts of the problem need to be addressed together, rather than just individual elements. In the context of dimensions, a one dimensional shape is one metric, yet the resources that make up a cloud compute consist of multiple metrics (CPU, IO and Memory), therefore, the dimensions are considered a vector. These vectors may increase in number to included other areas of the cloud technological stack such as network *throughput*, an element of scale when performing workload *placement*. Identifying the workload within the context of a database employing advanced features is also challenging. A clustered database can be seen as one database residing on multiple nodes, when in fact it is a database served by a number of memory structures, in the form of instances residing on each node, all consuming a *vector* of resources. These *clustered* instances may not run evenly, as QoS may dictate that one node is busier (consuming more resources) than another, forcing connections through a particular node rather than sharing them evenly. Databases that are consolidated in the form of a *pluggable* feature also pose a conundrum. A pluggable database can be detached from within a clustered database and attached elsewhere to another clustered database, each pluggable database consumes a vector of resources. The final problem with placement of workloads from advanced database configurations is the issue of High Availability (HA) from standby databases or clustered databases. If a clustered database resides on three nodes then it must be placed on three nodes or more otherwise we may reduce the QoS mentioned earlier.

Bin-packing algorithms to place or order tasks is a well understood problem, as analysed by Carter [Bay77], who conducted an analysis of the different types of bin-packing algorithms in 1977. Resource allocation or workload(s) placement in cloud environments is a well understood problem, for which bin-packing and optimisation solutions have been used. There have been extensive studies or surveys undertaken as Hameed, Khoshkbarforoushha *et al* [HKR$^+$16] and Bhavani and Guruprasad, [BG14] both identified in their survey's from 2014 and 2016. Furthermore Singh and Chana [SC16] produced an extensive survey in 2016 that concluded that resource provisioning is a challenging job and there is a need for more research into optimal resource usage as this leads to improving the resources consumed with the aim of reducing

wastage. This problem is ever more apparent in cloud computing, with users accessing any shape of resources (vectors) from anywhere, with the requirement of still being optimised. Most research has looked at consolidation from a Virtual Machine (VM) perspective or a Quality of Service (QoS) perspective and these are often single dimensions not vectors of dimensions. By focusing on bin-packing algorithms, we are evaluating if existing algorithms, such as first-fit decreasing are sufficient in placing workloads from complex database architectures that employ, for example, Clustered or Pluggable configurations.

3.4.1 Bin-Packing

The basic bin-packing problem is the process of taking items of differing volumes and *packing* them into a finite number of *bins* in a way that minimises the number of bins used. If one is to bin-pack on multiple metrics then one must use a vector approach to bin packing as Azar *et al* 2013 [ACKS13] described in their understanding. Gmach *et al* 2007 [GRCK07] provided a placement algorithm after forming a prediction on workloads and their demands at a Data Centre. However, they are not specific on the type of bin-packing algorithm used. For example, first, next or best-fit decreasing, or how they handle workloads that are *siblings* of each other, which clustered workloads are. As recently as 2018 several authors have found characterising and managing workloads a problem. Sen *et al* [SR18] concluded that sensitive workloads such as OLTP, analytical and hybrid may not be *optimally* provisioned, thus over-provisioning may be unavoidable. Zhang at al 2018 [ZMPC18] also viewed workload characterisation with a view to optimisation. Masari and Khoshnevis [MK20] in their 2019 survey identified techniques used to perform accurate forecasts of the resources being consumed as a precursor to provisioning. However, they stopped short of proposing how to place the workloads together once the forecasted future requirements are obtained. The most recent work covering vector bin-packing focused on IaaS and the placement of VM as the request is sent by the user as Pandiselvi and Sivakumar 2021 [PS21]. In their solution they looked at a 'Best-Fit' to provision VMs in space that has become available. Again looking at the data centre problem of the 'provider'. Ke *et al* 2021 [KGJ+21] recently, looked at Vector Bin Packing as a solution to Cloud Resource Managers, which they name as 'Fundy', in cloud when provisioning VMs. Fundy works as a cloud micro-service that farms out requests to multiple schedulers. This approach highlights the difficulty with dealing with large data centres that Cloud Vendors are having to provision for their customers.

Most placement of workloads seems to centre on provisioning of VMs (IaaS or PaaS) and not a workload (DBaaS or SaaS). Doddavula *et al* 2011 [DKJ11] suggested how to reduce server sprawl with the introduction of a vector packing algorithm. In their novel approach they classify vectors based on resource consumption and then through Matrix multiplication determine the possible combinations. By then applying rules, either the workload is full or a magnitude of full determine where the workload should reside with other workloads until the maximum of the target server threshold is reached. However in a clustered environment, this approach will face challenges as it is possible that several workloads, in a pluggable configuration, running on the same cluster are, classified as, full or a combination of classifications that could break their algorithm. Yu et al 2012 [YQR+12] looked at specifically first-fit decreasing bin-packing algorithms with a view to placement of workloads in DBaaS, however they do not address consolidated databases such as pluggable or clustered workloads. Azar *et al* 2013 [ACKS13] specifically used a vector (multiple metrics) to place virtual machines and suggested several algorithms based on lower or upper bounds of the resources consumed. Aydin *et al* 2019 [AbMl20], looked at placement using best-fit decreasing algorithm for the purpose of reducing the initial resource consumption of VM *fire-ups*. In this thesis we aim to plug the gap in placement of workloads from advanced database configurations into complex cloud architectures by creating new bin-packing algorithms.

3.4.2 Optimisation

Leading on from Bin-Packing as a solution to workload placement is the other side of the coin, and *view* placement of workloads as an optimisation problem, where priority is assigned to the most important workload. Kouki *et al* 2012 [KL12] perform a prediction based on SLAs with a view to establishing a capacity plan but they do not *place* or *migrate* a workload based on this analysis even though they view it as an optimisation problem. Boukhelef *et al* 2017 [BBB+17] took a heuristic approach to placing IO workloads in a DBaaS cloud but focused on the storage element. The challenge here is that workloads can have more than one metric that makes up compute (Memory and CPU) therefore a vector is required. Zhang *et al* 2018 [ZMPC18] provided a survey of database workloads and how they are managed and concluded that there are very few facilities to manage frequently changing workloads. They do not look at clustered workloads but mainly the resources consumed by a SQL query and

how this is managed within the database, viewing the work as an optimisation problem. Halfpap and Schlosser 2019 [HS19] used a heuristic linear programming model to solve a placement problem by dissecting a database and where appropriate, replicating the data across multiple nodes. In doing so, this placement technique *shared* the workload from being cumulative on one node to being distributed (replicated tasks) between multiple nodes while keeping response times of the requests from said database optimum, choosing the most optimum response time from the distributed group. However, replicating data could be wasteful in a cloud environment if the consumer is obliged to pay more than once (storage) for the same data in a trade-off against performance, which is not addressed by Halfpap and Schlosser 2019. In this thesis I have chosen not to view workload placement as an optimisation problem as the workloads have yet to move to cloud. They reside in N-tier architecture, on-premises therefore leading me to lean more towards a placement problem.

3.5 Conclusions

In summary, there are clearly gaps when performing a capacity planning exercise that requires housing databases in clouds, especially when advanced database features are employed. Whether that is collecting, extracting, accounting for the correct resources such as a CPU, IO and Memory from clustered, consolidated or separated and standby databases. These workloads change depending on the configuration or architecture, thus understanding which metrics to obtain is key. Categorizing the workloads into general types such as OLTP, OLAP or Data Marts with an aim to labelling the workloads that exhibit similar work patterns is also paramount as workloads and their data signal change in their behaviour, especially if the workloads fail-over or a standby is introduced. Once the data is extracted, a deeper understanding on the data signal is required from each of the labelled workloads prior to forming a forecast. Workloads exhibit complex data structures such as trend, seasonality and shocks, which must be accounted for if the forecast is to be accurate. Regardless of whether the question is one of *short* (Monitoring) or *long* (Capacity Planning) term resource consumption, accurate forecasting at speed and scale is required. One must couple multiple techniques together if one is to perform forecasts at scale (large estates). Once the forecast is obtained, placing the workloads into a target cloud configuration without compromising SLAs, SLOs or QoS presents a conundrum. One must identify the correct combination

of workloads to place in the correct target configuration without exceeding the available resources. Existing bin-packing algorithms, work on a *first*, *next* or *best*-fit basis, and, arguably, are too simplistic to place multiple *source* workloads and their vectors into multiple *target* cloud nodes at the same time. Therefore bin-packing algorithms also require further extensions.

Chapter 4

Experimental, Workload and Environmental Setup

What most experimenters take for granted before they begin their experiments is infinitely more interesting than any results to which their experiments lead.

Norbert Wiener

Experiments were conducted for each piece of work and were important to empirically evaluate new algorithms that were developed as part of this thesis. This section describes in detail the environmental setup, code or algorithms developed and the design and reasoning for the experiments. There will be a reasonable amount of duplication from what is described in the published works. It is important to understand that extracting data from live environments that customers utilise on Oracle technology could not be done due to contractual obligations; Oracle has a policy of not using customer's live data for their own means without customer approval. However, the environment was provided by the Oracle Corporation and was real world production ready physical Hardware and Software that customers utilise today. The workloads were developed to reflect *N-Tier* architecture and we will discuss this in more detail further on in this chapter. The work was conducted on Oracle technology. However, the algorithms produced in this thesis are agnostic of platform, vedor or configuration. As long as the relevant metrics are captured to reflect the architecture, and the forecast performed on a time series signal. Placement can take place using our bin-packing

algorithm in any target bin so long as the sizes of target bins are inputted into the templates.

The first piece of work as described in the paper titled *DBaaS Cloud Capacity Planning - Accounting for Dynamic RDBMS System that Employ Clustering and Standby Architectures* required experiments to investigate if a workload executed on one machine consumes similar resources when the workload is executed on another machine. The aim was to investigate what could cause the differences in resource consumption and we focused on three types of database configuration.

- Running workloads on a single instance - basic configuration.
- Running workloads on a single instance with a standby database - moderate configuration.
- Running workloads on two node clustered database - advanced configuration.

It was important to create experiments that satisfied the basic requirement to be *controllable*, *repeatable* and *observable*. The experiments and the environments created the foundations for all the work going forward, thus it was critical it was done correctly to ensure that errors, in the data, when obtained from experiments relating to forecasting where kept to a minimum given forecasts where expected to be performed on data extracted from the workloads being executed. If the metric data is inaccurate then it is reasonable to assume that any forecast will also be inaccurate. The first piece of work as detailed in the paper titled *DBaaS Cloud Capacity Planning - Accounting for Dynamic RDBMS Syste m that Employ Clustering and Standby Architectures* focused mainly on understanding workloads and how to capture the nuances of system configuration between different *types* of databases that can impact a workload, and how these impacts are reflected in the traces obtained from the metric data.

The second piece of work as detailed in paper *Database Workload Capacity Planning using Time Series Analysis and Machine Learning*, required a set of experiments involving extracting a trace of metric data from the workloads developed and executed from the first previous experiments. Once extracted, forecasts (predictions) where performed. However, to truly evaluate the forecast models, the workloads needed to exhibit complex data structures such as seasonality, trend and external shocks, hence the need for the workloads to be controllable. We achieved these complex execution patterns by executing the workload via Cron, which is an in-built scheduler that can execute tasks at a particular time as described in Section B.1.4 of Appendix B. Three

workloads were executed (OLTP, Data Mart and OLAP) and are also described in detail further on in this chapter in section 4.2. An online backup was executed via Oracle's internal back-up program Recovery Manager (RMAN) [Cor21l], daily to produce an external influence on the system that was seen in the IO metric. An online database backup executed in twilight hours reflects natural business operations of live environments. To produce trend, users where spawned in a controlled manner at daily time intervals, reflecting more general activity that was captured through the trace metric data. Seasonality (Patterns) were performed by spawning a surge of users at particular times of the day, reflecting users logging on at, for example, 09:00am of a working day. Other patterns produced where to reflect data being aggregated in twilight hours, highlighting data being crunched to form BI reporting in an OLAP schema, and backups executed at a uniform time intervals. These trends and seasonality where executed over many weeks to provide adequate data to perform predictions at regular frequency. The data signals observed produced complex data structures and therefore, for the experiments to be successful, the models had to be able to predict the following conditions and their ability to accurately assess:

1. Reoccurring Patterns (Seasonality).
2. Trends and Stationariness.
3. Mutliple patterns (Overlapping seasonality).
4. Shocks.

Time Series analysis lends itself to forming predictions more accurately, compared with stochastic processes, because of the complex patterns that are exhibited in volatile data signals, such as CPU and IO, as discussed in the *Related Work* section [Sko19]. To test the models properly we had to first execute the workloads to populate the central repository. Extracting the data from the database instances was done via an intelligent agent, to reflect monitoring practice in customer estates employed today. Data capture of the metrics is executed at 10-15 minute intervals and stored as raw data in a central repository. Aggregations are performed on the raw data to capture the *max_value* hourly, and stored in the central repository. By performing hourly aggregations, centrally in the repository, we remove any anomalies that may occur from performing aggregations on each instance at a time. Also, in clustered environments all metric data is required to give an accurate picture given data can be shared accross the cluster. We then ran the forecast models on the now, uniformly, hourly data for each instance,

to obtain a prediction. Testing the accuracy of the models was done by conducting the following:

1. Running the prediction and comparing it to the actual line by allowing the workloads to run forward.

2. Calculating the Root Mean Squared Error (RMSE) rate between the prediction line and the actual line.

3. Calculating the Mean Absolute Percentage Error (MAPE) rate between the prediction line and actual line.

4. Calculating the Mean Absolute Percentage Accuracy (MAPA) rate between the prediction line and actual line.

The final piece of work as described in the paper titled *"Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure"* was dealing with placement of database workloads into complex target cloud configurations. This was further broken down into several smaller evaluations of fitting workloads from, use cases of varying complexity into target cloud compute nodes, as shown in Table 4.1. In the experiments we are only testing the database placement algorithms as they are orthogonal to modelling; the placement algorithms do not know if the traces being inserted as inputs to the algorithms are actual or modelled. However, it is perfectly plausible that the inputs have first been predicted to obtain an estimate of future resource consumption to model what a future resource consumption may be prior to being placed, which is a common planning exercise in any estate migration. Once placed, a further experiment was conducted to evaluate if the *placed* workloads could be *fitted* tighter or more *dynamically* to achieve any efficiencies by elasticising the nodes. The use case was a consolidated environment where multiple pluggable databases can be detached and attached in a target node.

The experiments were designed to answer the following key questions relating to the algorithms and code developed.

1. Minimum targets needed - What is the minimum number of target nodes required to fit all workloads across all vectors (metrics)?

2. First Fit Decreasing Simple Placement - How do we place the workloads equally across equal sized bins?

Table 4.1: Workload Placement - Table of Experiments

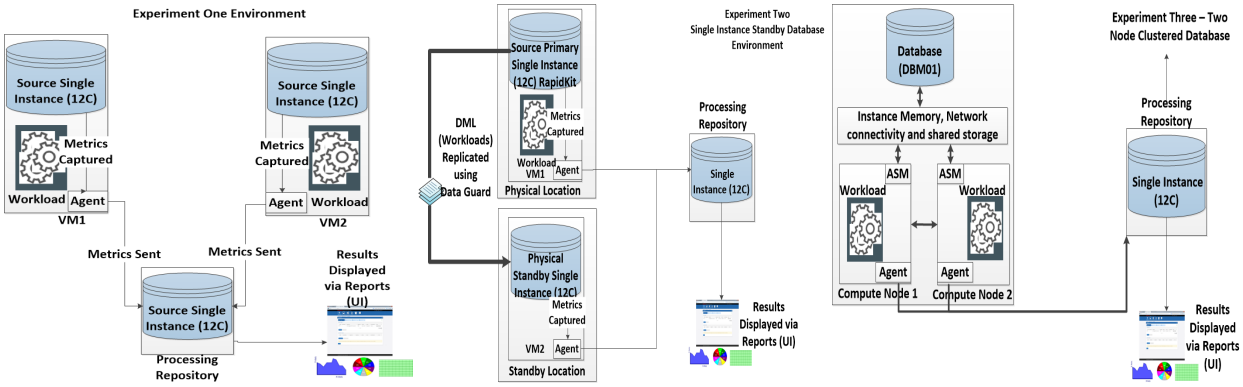
Experiment	Workloads	Target Bins
Basic Single Database Instance	10 Workloads (10 OLTP, 10 OLAP and 10 DM)	4 * OCI Bare Metal equal size
Basic Clustered Workloads	10 Workloads (10 RAC OLTP (5*2 Exadata nodes))	4 * OCI Bare Metal equal size
Basic different sized target bins	10 Workloads (10 OLTP, 10 OLAP and 10 DM)	4 * OCI Bare Metal unequal size
Moderate Combined (Clustered and Single Instance)	20 Workloads (4 * 2 node clustered + 5 OLTP, 6 OLAP and 5 DM)	4 * OCI Bare Metal unequal size
Moderate scaling	50 Workloads (10 * 2 node clustered + 10 OLTP, 10 OLAP and 10 DM	4 * OCI Bare Metal equal size
Moderate different sized target bins	20 Workloads (4 * 2 node clustered + 5 OLTP, 6 OLAP and 5 DM	6 * unequal OCI Bare Metal
Complex (Scaling & diffrent sized bins)	50 Workloads (10 * 2 node clustered + 10 OLTP, 10 OLAP and 10 DM	10 * unequal OCI Bare Metal

3. First Fit Decreasing Clustered Placement - If there are clustered workloads can we ensure that the all clustered workloads are placed without compromising High Availability?

4. Evaluating the placement - Once the workloads are placed (consolidated) together can we resize the target nodes, obtaining a tighter fit, reducing over provisioning?

4.1 Experimental Environmental Setup

As shown in Figure 4.1, several configurations of databases employing different advanced features where implemented to run the workloads. The workloads were executed via a Java Container, which spawned connections to the database. The storage layer was mounted and accessed via a network to a SAN. Intelligent agents executed various commands to poll metrics that were stored in a central repository. Oracle Automatic Storage Manager (ASM) [Cor21a] is an Oracle product that is a volume manager and file system for Oracle Database files. This is essential when running Oracle Real Application Clusters (RAC) [Cor21k] as ASM controls IO by managing the configuration across disks that are critical in the database storage architecture. This is shown in Figure 4.1c where ASM is a sub-component of the technological stack.



(a) Single Instance DB (b) Instances with Standby DB (c) Two Node Clustered DB

Figure 4.1: Experiment Architecture: different database combinations used for experiments.

4.2 Definition of a Database Workload

A workload can be described as the activity being performed on the database at a point-in-time, and essentially is broken down into the following:

- *Database* - An Oracle database is a set of physical files on disk(s) that store data. Data may be in the form of logical objects, such as tables, views and indexes, which are attached to those tables to aid the speed of access, reducing the resources consumed in accessing the data.

- *Instance* - An Oracle instance is a set of memory structures and processes that manage the database files. The instance exists in memory and a database exists on disk; an instance can exist without a database and a database can exist without an instance. For a database to be accessible the instance must be running.

- *Activity* - The DML (Data Modification Language)/DDL(Data Definition Language) i.e, SQL that is being executed on the database by the application, creates load consisting of CPU, memory and IOPS.

In the experiments, workloads are executed for a period of time via cron as shown in Table 4.2. For example, an OLTP workload will have 100 users if it is a small singular database, however for a larger RAC database the number of users is 2000 executed evenly across the cluster, running for 23:59 each day. We allow a 60 second gap before the workload resets to remove the risk of overlap between one workload finishing and another starting with the impact being, spawning 4000 users temporarily

that would create an artificial spike. This 60 second gap allows existing users to finish the execution of the queries which are essentially elongating in their elapsed execution times as the data set gets bigger. To create a surge, a further 100 users are spawned executing DML, against the Order Entry schema, which is described in Figure 4.2, for a period of 2 hours at 09:00am in the morning, 1 hour at lunch and 5 hours in the evening, which is increased to 1000 users if the database is a larger RAC configuration. For an OLAP workload different DML is executed against an Sales History schema, as described in Figure 4.3. These DML statements consist of more IO type queries and are executed for 5 hours in the evening. Data Marts are a combination of both OLTP transactions, short DML SQL during the day and larger more IO intensive DML SQL in the evenings. The workloads were executed daily over a period of several weeks, and where continuously monitored by a Database Administrator (DBA) should any alerts, errors or faults take place so that they could be rectified. If an error took place which rendered the workload or database in a state requiring a reboot, restart or session termination, the experiment was halted and rerun from scratch. In cases where a session needed to be restarted due to a spinning daemon process, that session was terminated and re-spawned automatically. The first few runs of the workloads flushed out many errors relating to memory configurations, and once stable ran for many weeks without error.

4.3 Swingbench

Swingbench [Gil10] is a, free to download, Java load generator based on the example schemas that come with the Oracle Database. Prior to running any workloads, there are some prerequisite configurations to be executed on the database as an administrator (DBA) with privileges and these are described in Appendix B.1.4. For example, the database type will determine specific parameters to be set, and the database must be in archivelog mode. There are essentially two main schemas that are used in the Swingbench load generator as shown in Figures 4.2 and 4.3 (“Copyright Oracle and its affiliates. Used with permission”).

- OE - Order Entry of customers purchasing items online (OLTP)
- SH - Sales History which is an OLAP schema

Workload Type	Workload Profile	DBNAME(S)	Workload Description	Number of Users	Duration (hh:mi)	Avg Transaction per sec
OLTP	General usage	RAPIDKIT RAPIDKIT2 DBM01	General Online Application with updates, inserts and deletes simulate working day	100 2000 (DBM01)	23:59	0.2
OLTP	Morning Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Morning Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 1000 (DBM01)	2:00	0.2
OLTP	Lunch Time Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Lunch Time Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 1000 (DBM01)	1:00	0.2
OLTP	Evening Time Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Evening Time Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 1000 (DBM01)	5:00	0.2
Daily OLTP Hot Backup taken at 23:00						
OLAP	Data Warehouse General Usage	RAPIDKIT RAPIDKIT2 DBM01	General Data Warehousing Application with heavy Selects taking place out of hours building Business Intelligence data	5 400 (DBM01)	8:00	0.4
Daily OLAP Hot Backup taken at 06:00						
Daily OLAP archivelog backups taken at 12:00,18:00,00:00						
DM	OLTP General Usage	RAPIDKIT RAPIDKIT2 DBM01	Combination of DML taking place during the business day and heavy DML taking out of ours	200 1000 (DBM01)	23:59	0.2
DM	OLTP Morning Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Morning Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 500 (DBM01)	2:00	0.2
DM	OLTP Lunch Time Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Morning Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 500 (DBM01)	2:00	0.3
DM	OLTP Evening Time Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Evening Time Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 500 (DBM01)	5:00	0.3
DM	OLAP Batch Loads Peak	RAPIDKIT RAPIDKIT2 DBM01	Evening Time Surge to simulate users logging on to the Online Application with updates, inserts and deletes	5 400 (DBM01)	8:00	0.3
Daily DM Hot Backup taken at 06:00						
Daily DM archivelog backups taken at 12:00,18:00,00:00						

Table 4.2: Database Workloads

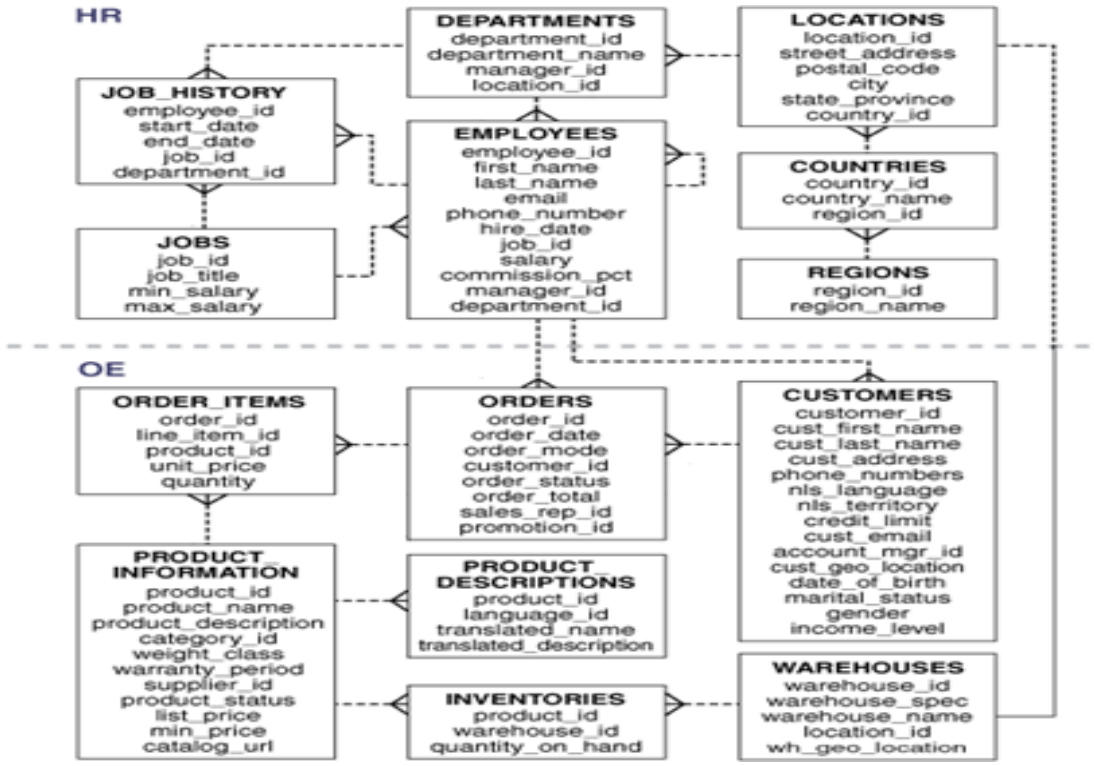


Figure 4.2: Order Entry - OLTP Schema [Cor13] [Gil10]

4.3.1 Execution

Swingbench has SQL statements embedded in the Java code executed via a Java container with connections spawned through the Oracle Network layer (TNS) on the database server via a JDBC thin client. The configuration used to run this type of application reflects enterprises that are employing on-premises N-Tier architectures. The Linux Operating System Scheduler cron is used to initiate the workload and the cron entries for the workload are shown in the appendix in Section B.1.3. Cron is configured in the following sample output, taking the example of a daily workload that runs for OLTP throughout the day for 24 hours for 200 users. Start at 3pm (0 15 * * *) execute the soe_bench.sh in the database identified as RapidKit for a run time (-rt) of 23 hours 59 minutes spawning 200 users (-uc) pipe the output logfile location to be run in the background (2&1)

```
0 15 * * * /u01/../soe_bench.sh RapidKit -rt 23:59 -uc 200 >> /tmp/daily.log 2>&1
|--------| |------------------| |------| |--------||------|  |-------------| |---|
```

4.4 Application Design to Generate Workloads

OrderEntry is based on the *'OE'* schema as shown in Figure 4.2. It can be run continuously, that is until you run out of either file space (Archivelogs) or tablespace (within the database). It introduces heavy contention on a small number of tables and is designed to stress interconnects, CPU and Memory based on a OLTP workload, TPC [Cou21]. It is installed using the wizard located in the bin directory and is discussed in detail in the appendix in Section B.1.

SalesHistory is based on the *'SH'* oracle sample schemas [Cor13] as shown in Figure 4.3, Like the *'OE'* schema. It is designed to test the performance of complicated queries when run against large tables. This type of schema is based on OLAP applications that are designed to test IO intensive queries.

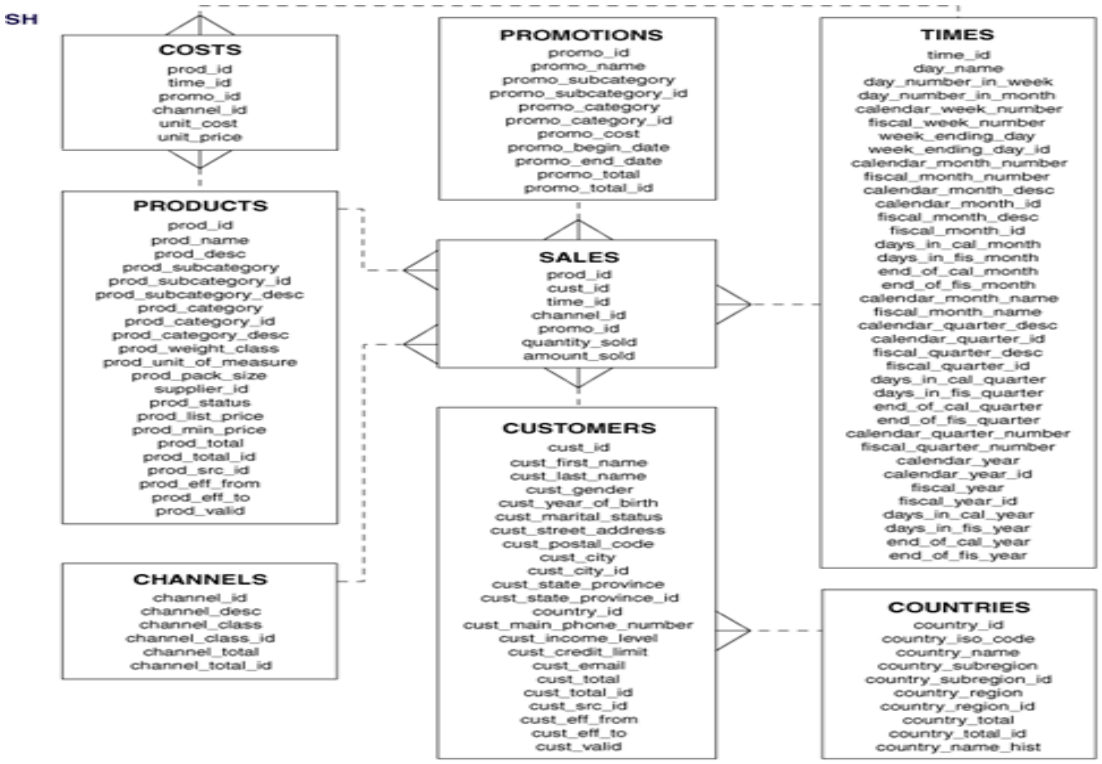


Figure 4.3: Sales History - OLAP Schema [Cor13] [Gil10]

Both schemas are executed via a JDBC and PL/SQL, and can be scaled over a number of default sizes from 1GB to 1TB. In our experiments we begin with the schema being small and over a period of weeks it grows to a significant size and thus trend, in the form of incline growth, is introduced. This can be seen in the appendix in Section B.1.3 under the paragraph titled *'RAC Instance OLTP - Linear Growth'*. As the data

size increases, the amount of resource consumed to execute the queries also increases as more data are returned through the database query processor. The SQL is working harder on a larger data set.

4.4.1 Physical Hardware and Operating System Configuration

The experiments involved an eclectic set of hardware, configured to run different types of database as shown in Table 4.3. The hardware used was of Oracle Exadata [Cor21d] of a 1/4 rack configuration with several VMs created of various physical requirements. The Operating System used was Oracle Enterprise Linux with various custom parameters set, which we will discuss further on in the chapter. The 2-node cluster also had specialist network configurations set to enable the database cluster to operate correctly, namely Virtual IP Addresses that manage the *heartbeat* function that keeps node integrity within the cluster [Cor21k]. The Exadata Machine comes with its own storage configuration, which ASM sits on top of, managing the IO generated from the RAC Database. The single instance databases where configured to have NAS attached SAN file system Storage that was not ASM managed. For the duration of the experiments no updates on the physical hardware or operating system were performed. This is because updates to the physical hardware or operating system can influence the behaviour of the workload. For example, the Kernel of the operating system may depreciate parameters that impact CPU, IO or Memory, between major releases resulting in the workload being processed differently. By keeping the configurations static we reduce the risk of these affects being reflected in the data signal, thus skewing any forecasts.

4.5 Data Capture and Processing - Enterprise Manager

Capturing the data is handled by an intelligent agent installed on each host where the experiment is conducted. The agent has the ability to identify *targets*, such as:

- Operating System and their configurations.
- Databases and their configurations.
- Database Instances and their configurations.
- Storage configurations.

VM Name	OS Type	CPU Details	Memory	Storage	Database Type	Products and Versions
Single Database Instance Configuration						
VM 1	OEL Linux 2.6.39	4 * 2.9 Ghz	32Gb	300Gb	Oracle Single Instance Database (RapidKit)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4),
VM 2	OEL Linux 2.6.39	4 * 2.9 Ghz	32Gb	300Gb	Oracle Single Instance Database (RapidKit2)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4),
Clustered Database Instance Configuration						
Clustered Compute Node 1	OEL Linux 2.6.39	24 * 2.9 Ghz	96Gb	14Tb	Oracle Clustered Multi-tenant Database Instance (DBM011)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4, • Grid Infrastructure (12.1.0.2), • Oracle Automatic Storage Manager (12.1.0.2),
Clustered Compute Node 2	OEL Linux 2.6.39	24 * 2.9 Ghz	96Gb	14Tb	Oracle Clustered Multi-tenant Database Instance (DBM012)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4, • Grid Infrastructure (12.1.0.2), • Oracle Automatic Storage Manager (12.1.0.2),
Standby Database Instance Configuration						
VM 3	OEL Linux 2.6.39	4 * 2.9 Ghz	32Gb	1Tb	Oracle Single Instance Standby Database (STBYRapidKit, STBYRapid-Kit2)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4),
Central Repository Details						
Storage Repository	OEL Linux 2.6.39	24 * 2.4 Ghz	32Gb	500Gb	Oracle Single Instance Database (EMREPCTA)	• Enterprise Edition (11.2.0.3), • Enterprise Manager R4 including Webserver and BIPublisher (12.1.0.4), • Enterprise Manager Agent (12.1.0.4),

Table 4.3: Platform Outline

- Performance Metrics of the Operating System.
- Performance Metrics of the Database and their instances.
- Performance Metrics of Storage software such as ASM.
- Performance Metrics of Storage Technology, such as NAS and SAN via Plug-ins

The agent must have the ability to login as a privileged user on the operating system (Oracle user) and database (DBSNMP user) with administrator privileges to enable the agent to execute commands and interrogate the database AWR repository. Data from all targets are then stored in a central repository in the SYSMAN schema, which houses a collection of tables, views and indexes, etc. These targets are stored as entities in the repository and a simple SQL query can extract the targets as described in section B.4.1. An aggregation DBMS task is executed nightly to create an hourly metric as described in Section B.4.4 of the appendix section. Parsing the data extracted from the OEM repository into Python via the Python database connector, Python libraries are used to perform forecasts and execute forecast or placement algorithms. Data is kept in the repository for 1 year, allowing a large number of data points to be extracted, providing the workloads are generating activity.

Should there be a break in the connection between the targets and the central repository, such as a network issues, then the agent stores the metric data in the form of xml files locally on the target and once connectivity is restored the data is then uploaded and processed in the repository retrospectively. The xml files are then cleared down once applied and the data stored in the SYSMAN schema. The central repository considered to be as important as any live critical database, and has its own RMAN backup routine executed nightly. If there is a gap in the time series data then an *interpolation* technique is used to estimate the data between two gaps. We do this if there are only 1 or two gaps, rather than reset a whole experiment to save time.

4.5.1 Data Capture and Processing - Python

Python [Fou21] was used to implement the algorithms and carry out the experiments that acted as a Proof of Concept (PoC). Commercial development, within Oracle is often conducted in Java, Javascript and follows a strict SDLC. There are several key libraries and techniques that were used and these are described in detail in Sections B.5 and B.5.1.

In the second piece of work titled *"Database Workload Capacity Planning using Time Series Analysis and Machine Learning"*; we used python as the program language to test the ARIMA, SARIMA and SARIMAX solutions of forecasting metric data. We utilised several packages that are listed in detail in Table B.3. Once we have forecasted the model, that model and its data are then stored in a repository. Root Mean Squared Error (RMSE) calculations are then calculated. If the RMSE drops to below 85% accuracy and/or a period of 7 days elapses, which creates new data points, then the algorithm is forced to re-learn. We do this to ensure consistency that the model is up to date and takes into consideration the latest data points. This mechanism was tested as part of the experiments to ensure that we could scale by processing many models on many metrics from many database instances. Scaling *out* was a problem for forecasting because we found that it was taking approximately 11-13 minutes for one forecast model to process 1000 hours of metric values to predict 24 hours in the future. we needed a solution that could reducing the elapsed time of 11-13 minutes to execute the model to less than 3 minutes. Supervised Learning solved this problem because it learnt the historical data. We trained on 60% and test on 40% split. The models that require relearning are then rerun as a batch job to keep resource utilisation to within acceptable boundaries. Supervised Machine Learning to perform forecasts is not new *per se*, however supervised learning on metric data *en masse* and on volatile data with complex data structures such as trend, seasonality, stationariness on metric data such as IO and CPU, is.

In the final piece of work titled *"Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure*", we, again, used Python as the program language, the motivation for this was much like the work on forecasting, it was quick and had lots of opensource libraries that could be readily used. There was a *'binpacking'* package, however, this package was simplistic and could not be used to deal with advanced workloads such as clustered environments or pluggable databases. We needed to write our own bin-packing algorithm that would also act as a PoC.

4.6 Conclusions

The motivation for the experiments being configured and conducted in the manner they where was several fold.

- Reflective of customers environments that are employed today.

- *Controllable*, *repeatable* and *observable* given we were performing forecasts based on metric data that required measurements to asses accuracy.
- Methodical in that Oracle Engineering Teams and Service Delivery Teams would accept as standard practice.
- Defensible in that the work undertaken in the thesis can be implemented into Oracle Products and Services.

It was important to focus on current *N-tier* architectures that enterprises and organisations employ today while undergoing cloud transformations. This meant that we had to utilise hardware of sufficient capacity to reflect live systems, which the Oracle Corporation kindly made available. Oracle Exadata hardware and Oracle Software is a license product that is expensive when it is licensed by the CPU under current license agreements. Some of the clustered nodes had 24 CPUs each, totalling 80 CPUs, as shown in Table 4.3, at several thousand dollars per CPU. Oracle allowed these experiments to be run without having to pay any license fees, and we had experiments running thousands of users over a period of weeks on over 50 database instances. There was a significant amount of work in installing, configuring hardware and software, creating the workloads, installing and configuring the agents and setting up the experiments to run error free for weeks at a time across multiple databases. Creating the experiments to be repeatable, controllable and observable also was a challenge especially for many pieces of work on different database configurations. It was important to corroborate a peak or trough in the data signal with a surge in users to ensure no spurious results skewed the predictions, thus initially a lot of data analysis was undertaken to scrutinise the metric data to obtain confidence in the workloads and agent capturing the data. We achieved confidence, by comparing the data from the AWR repository of the individual database instance against the metric data held in the OEM repository. We performed data reconciliation for each workload (OLTP, OLAP and Data Mart) and once correct we we could have confidence in the environment an experiments.

The second piece of work titled *"Database Workload Capacity Planning using Time Series Analysis and Machine Learning"*, was more challenging. From the experiments conducted in the work titled *"DBaaS Cloud Capacity Planning - Accounting for Dynamic RDBMS System that Employ Clustering and Standby Architectures"*; we obtained volatile data signals as we gathered the data at a granular frequency (Hourly data points), requiring the creation of new models. The work was carried out in consultation with Advanced Customer Services Engineering teams to ensure that any code

we produced followed their standards and adhere, to their SDLC but in the form of a PoC. Time Series Analysis is not a new concept, however, after the first slew of experiments it became apparent that the time to compute the models and perform a prediction on one metric from one database instance was taking a long time and was not scaleable. Rethinking the solution we focused on Machine Learning as potentially being able to reduce the *'crunch'* time by learning (training) 60% of the data signal therefore once learned the algorithm would only need to crunch the few data points added as the timeline rolled forward.

The final piece of work titled *"Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure"*, was also challenging as it required more new algorithms. We used the target architecture to be current Oracle Cloud Infrastructure Bare Metal configuration to keep with the mantra of *real world* system configurations. Obtaining a cloud architecture was not possible so we based the placement on target specifications via templates. The algorithms do not migrate the workloads; they provide a *plan* or *design* prior to a migration. Analysis of the architecture teams that Oracle Advanced Customer Services showed that providing an architectural diagram of workloads via a design diagram would be advantageous as it allows flexibility in moving workloads. We wrote the new algorithms based on templates reflecting the resource sizes of physical hardware, that a user would input or configure. For example, *selecting* the number of on-premises databases that are to be *placed* or *selecting* the available metrics that make up a vector, and configuring the available target resource, such as Memory, IO, CPU. This approach allowed for greater user interaction and could be automated by means of extracting the data from a central repository that holds metric data, such as the OEM repository from used to conduct the experiments throughout the thesis. In the final paper, a methodology is described in algorithm tables of how bin-packing works depending on the type of workload being placed. The bins are not dynamically changed by the algorithm but do take into consideration the amount of available headroom prior to placement. If the user wishes to manually increase the size of the target bins and then, for example, rerun the bin-packing routine with new sizes of target bins. The bin-packing algorithms will assume the new size of the bins and place the workloads accordingly. It is not intelligent enough to be dynamic.

Chapter 5

Presentation of Published Work

A drug is a substance which, if injected into a rabbit, produces a paper.

Otto Loewi

This chapter presents the core contributions of the thesis in the form of a collection published peer-reviewed papers. The published papers appear in International Conference Proceedings each with a high impact factor (Downloads and Citations). The contributions are presented in order as discussed in Section 1.10, and Appendix A shows the formal permissions for reusing them.

For each publication source, this chapter shows impact measurements in terms of ranking (if available), acceptance rate (if available in the publication year), impact factor, citation ranking (if there are citations) and awards (if any). A conference impact factor is estimated for the most recent (possible) year by analysing academic citations from Google Scholar. The rankings are obtained from resurchify [Met21], which assess major conferences and journals in Computer Science. The citation ranking is determined by retrieving and sorting paper citations also from Google Scholar or the University of Manchester's own publication bibliographical metadata or from the journal's own website such as Association of Computer Machinery (ACM) [DL21]. For the journal submitted, we obtained the most recent impact factor, the SCImago Journal Rank and (SJR) at the time of December 2022.

5.1 DBaaS Cloud Capacity Planning - Accounting for Dynamic RDBMS System that Employ Clustering and Standby Architectures

Antony Higginson, Clive Bostock, Norman Paton, Suzanne Embury

In Proceedings of the 20th International Conference on Extending Database Technology (EDBT),
Published by OpenProceedings,
March 21-24 2017,
Venice, Italy,
Pages 687 - 699,
ISBN: 978-3-89318-073-8,
https://10.5441/002/edbt.2017.01.

Summary: This paper sets the scene of how to account for advanced database configurations, obtaining the correct metrics at the right layer in N-Tier architectures. We empirically evaluate how the configurations and advance features impact the workloads and why this is important if we are to understand the target configuration of the same advanced database features to be created in a cloud.

Approaches: The approach taken was to empirically evaluate by using increasingly complex deployments of databases systems. Then execute a workload on each system and capture the usage, by means of an intelligent agent, periodically taking measurements of metrics such as, CPU, Memory and IO. These, relevant, metrics are stored in a central repository, which can then be mined to measure any differences the complex architectures have on the workload. The workloads are based on established benchmarks such as SPECInt (CPU) and TPC (IO). This paper highlights and answers the question where database deployments increase in complexity, what factors impact on the execution of the workload and by how much. Does this influence how a capacity planning exercise should be viewed prior to cloud adoption?

Comments on authorship: I proposed the main idea of the paper, developed and validated the main approach, conducted an empirical evaluation, analysed results, investigated related work, provided results, and all graphics, participated in the entire writing processes and addressed any comments. I created the environments, wrote any code in the form of shell scripts to execute the workloads. I configured any operating systems and databases as required and acted as the DBA to monitor the environments

as the experiments ran. Specific shell code was written to extend Swingbench for our purposes with the help of an Oracle Software Engineer (Clive Bostock) to ensure we followed Oracle SDLC. I presented the work at the EDBT Conference. My supervisors Norman Paton and Suzanne Embury also contributed to the idea, experiment design, literature research and analysis. They also proof read the paper and approved the final submission. They also guided the whole research process.

Key Contribution: See Sections 1.9.3 and 6.1

Impact Factor:(Dec 2022) Impact score - unavailable, Scimago shows the following h-index 20, SJR 0.31, Google Scholar shows citations 3

DBaaS Cloud Capacity Planning - Accounting for Dynamic RDBMS Systems that Employ Clustering and Standby Architectures

Antony S. Higginson
Oracle Advanced Customer Support Services
Didsbury, Manchester, UK
antony.higginson@oracle.com

Norman W. Paton
School of Computer Science
University Of Manchester
Manchester, M13 9PL, UK.
norman.paton@manchester.ac.uk

Suzanne M. Embury
School of Computer Science
University Of Manchester
Manchester, M13 9PL, UK.
suzanne.m.embury@manchester.ac.uk

Clive Bostock
Oracle Advanced Customer Support Services
Didsbury, Manchester, UK
clive.bostock@oracle.com

ABSTRACT

There are several major attractions that Cloud Computing promises when dealing with computing environments, such as the ease with which databases can be provisioned, maintained and accounted for seamlessly. However, this efficiency panacea that company executives look for when managing their estates often brings further challenges. Databases are an integral part of any organisation and can be a source of bottlenecks when it comes to provisioning, managing and maintenance. Cloud computing certainly can address some of these concerns when *Database-as-a-Service* (DBaaS) is employed. However, one major aspect prior to adopting DBaaS is Capacity Planning, with the aim of avoiding *under-estimation* or *over-estimation* of the new resources required from the cloud architecture, with the aim of consolidating databases together or provisioning new databases into the new architecture that DBaaS clouds will provide. Capacity Planning has not evolved sufficiently to accommodate complex database systems that employ advanced features such as Clustered or Standby Databases that are required to satisfy enterprise SLAs. Being able to efficiently capacity plan an estate of databases accurately will allow executives to expedite cloud adoption quickly, allowing the enterprise to enjoy the benefits that cloud adoption brings. This paper investigates the extent to which the physical properties resulting from a workload, in terms of CPU, IO and memory, are preserved when the workload is run on different platforms. Experiments are reported that represent OLTP, OLAP and Data Mart workloads running on a range of architectures, specifically single instance, single instance with a standby, and clustered databases.

This paper proposes and empirically evaluates an approach to capacity planing for complex database deployments.

Keywords

Cloud, DBaaS, Capacity Planning, Database, Provisioning, Standby, Clustering

1. INTRODUCTION

Traditionally, companies accounted for the cost of assets associated with their I.T. using Capex (Capital Expenditure) type models, where assets such as hardware, licenses and support, etc, were accounted for yearly. For example, a software license usually is based on an *on-premises* model that would be user based, or by the CPU if the application served many thousands of users. The advent of Cloud computing, with the *pay-as-you-go* subscription based model, has changed the way company executives look at the costing models of their I.T.

A similar paradigm unfolds when I.T. departments such as Development, Delivery and Support teams need to provision environments quickly to meet their business goals. Traditional project methodologies would request environments aiding development and testing with the goal of going live. Procurement and provisioning took time that was often added to the project lifecycle. Cloud computing addresses such issues so that a user can, with ease, request the rapid provision of resources for a period of time.

Once the system went live those Delivery and Support teams would then need to account for resources those particular systems consumed, reconciling with the Line Of Business (LOB). The results of that analysis would then feed back into next year's Capex model. This ongoing capacity planning to assess if they have enough resources is needed to ensure is that, as systems grow, there are enough resources to ensure that the system is able to meet QoS (Quality of Service) expectations. Cloud Computing has also made some advances here by enabling a metering or charge-back facility that can accurately account for the resources used (CPU, Memory, Storage). Cloud Computing can dynamically modify the cloud to reduce or increase those resources

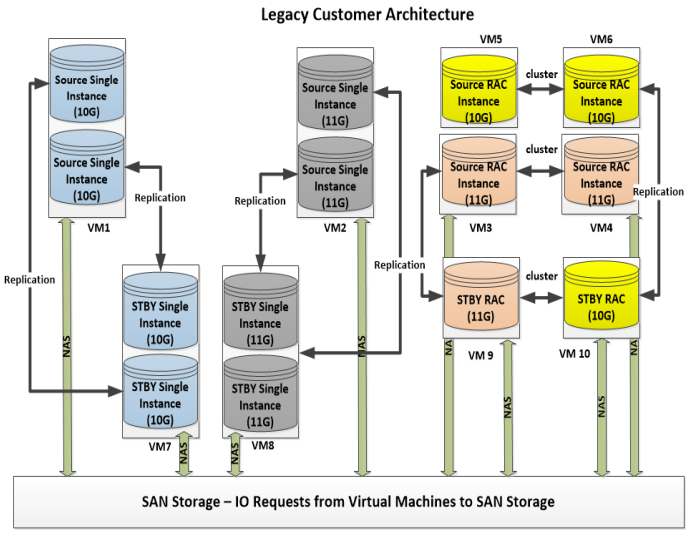


Figure 1: Example Architecture: Typical customer legacy database architecture.

as needed by the client.

However, companies with large estates have the additional challenge of having a plethora of database versions, for example, each database version offering a different feature that has a performance benefit over another database version. Similarly, the databases may be running on a eclectic set of operating systems and hardware, each affecting the workload in a subtle or major way. For example, the latest running version of a database may run on a highly configured SAN utilising the latest techniques in query optimization and storage. Comparing this footprint with an older version of software and infrastructure often leads to a *Finger-in-the-air* type approach.

A key feature of DBaaS is the ability to multi-tenant those databases where different workloads and database configurations can coexist in the shared resources, adding to the challenge of making effective capacity planning decisions. Determining the allocation is further complicated if the database utilises advanced features such as Clustering or Failover Technology, as workloads shift from one instance to another or are shared across multiple instances based on their own resource allocation managers. Furthermore, if a database employs a standby, this further complicates capacity planning decisions.

Cloud Computing is in its infancy, with incremental adoption within the industry as companies try and determine how to unpick their database estates and move them to cloud infrastructure. Databases often grow organically over many years in terms of their data and complexity, which often leads to major projects being derived when a major upgrade or re-platform exercise is required. With the introduction of cloud these exercises are becoming more prudent. This often leads to a series of questions on Capacity Planning.

- What is the current footprint of the database including any advanced features such as Standby or Clustering?
- What is the current configuration of the database?
- What type of DBaaS should I create?
- What size of DBaaS should I create?
- Can I consolidate databases that have similar configuration and utilisation foot-prints?
- Will my SLAs be compromised if I move to a cloud?

Such questions become very important prior to any provisioning or migration exercise.

The time taken to perform this analysis on databases also has a major impact on a company's ability to adopt cloud technologies often squeezing the bandwidth of the delivery and support teams. The departments suffer *paralysis-by-analysis*, and the migration to the cloud becomes more protracted to the frustration of all involved. If the analysis is not performed accurately then the risks of *over-estimation* and *under-estimation* increase. Being able to automate the gathering of data, analysing the data and then making a decision becomes ever more important in enterprises with large estates.

In this paper we look at the challenges of Capacity Planning for advanced database systems that employ clustering and standby databases, with a view to migration to a cloud. Our hypothesis is: "That a model based on physical measures can be used to provide dependable predictions of performance for diverse applications". We make two main contributions:

1. We propose an approach to workload analysis based on physical metrics that are important to capacity planning for database systems with advanced configurations.
2. We report the results of an empirical analysis of the metrics for several representative workloads on diverse real-life configurations.

The remainder of the paper is organised as follows. Section 2 introduces the Background and Related Work. In Section 3 we detail the environmental setup for conducting experiments outlining the database capacity planning problem. In Section 4 we introduce our solution in detail and

provide details on the experiments and analysis. Section 5 gives conclusions and future work.

2. BACKGROUND AND RELATED WORK

2.1 Background

Fig 1 shows an example environment of a company that is running different versions and configurations of databases on VM hardware. Physical machines are dissected into 10 VM's giving a level of separation. On these 10 VM's a total of 12 databases are run, of which 6 are primary databases and 6 are standby databases. This MAA *(Maximum Availability Architecture)* allows the company some comfort by running their primary *(Platinum)* SLA level applications on VM numbers 3, 4, 5 and 6, which host two clustered databases (offering a degree of resilience against node failure). In addition, these clustered databases have a physical standby database running on VM's 9 and 10 in case of database failure or corruption. Similarly, the 4 single instance stand alone databases that are running on VM's 1 and 2 also have a replicated standby database running on VM's 7 and 8, again offering the company some comfort that their secondary *(Gold)* level of applications will have a standby database for failover, should they need it.

The company also wish to increase their ROI *(Return on Investment)* with this environment and thus often open up the standby databases in *"Read Only"* mode during special times for applications that need to run year-end or month-end type BI *(Business Intelligent)* reports. This particular type of architectural pattern is a typical configuration companies use today to manage their database environments and applications that have 24*7 type SLAs. The difficulty becomes apparent when a new exercise is introduced that looks at consolidating, upgrading and migrating those environments listed in Fig 1 to a new cloud architecture, where resources can be tightly accounted and dynamically assigned. We are then faced with a capacity planning exercise.

2.2 Related Work

The objective of capacity planning is to provide an accurate estimate of the resources required to run a set of applications in a database cloud. Achieving this answer relies on the accurate capture of some base metrics, based on historical patterns, and applying some modelling techniques to form a prediction. There are two main viewpoints: the viewpoint of the CSP *(Cloud Service Provider)* in what they offer and their capabilities, i.e are there enough resources to provide services to consumers; and the viewpoint of the consumer, for example, can a customer capacity plan their systems against the CSP's capability? Indeed if the customer wishes to become a CSP but in a private cloud configuration, then the first viewpoint also becomes important.

A CSP offers resources, and existing models use various techniques to help customers assess the CSP capabilities. MCDM (Multi Criteria Decision Making) weighs the attributes of an individual database by their importance in helping to choose the right cloud (Mozafari et al 2013 [16] and Shari et al 2014 [19]. CSP's can also be assessed using a pricing model to validate their capability based on a consumers single systems workload as suggested by (Shang et al [20]); using this financial approach contributes to the *value-for-money* question that many enterprises seek when deciding on the right cloud.

If a consumer has a cloud, knowing where to place the workload based on utilisation to achieve the best fit is critical when beginning to answer the QoS (Quality of Service) question, and techniques such as *bin-packing* algorithms (Yu et al [21]) help achieve this answer. However systems may have dynamic workloads, which may evolve organically as datasets and/or numbers of users grow or shrink, as is especially common in internet based systems. There is a need for constant assessment of said workloads. Hacigumns et al [10] and Kouki et al [11] both look at the workload of an application or the query being executed, and then decide what type of database in a cloud would satisfy QoS. Mozafari et al [15] suggests using techniques that capture log and performance data over a period of time, storing them in a central repository, and modelling the workloads at a database instance level. With the advent of Virtualisation that enterprises utilise, including CSP's, when running their estates, several techniques such as coefficient of variation and distribution profiling are used to look at the utilisation of a Virtual Machine to try and capacity plan. Mahambre and Chafle [13] look at the workload of a Virtual Machine to create relationship patterns of workloads to understand how resources are being utilised, analysing the actual query being executed to predict if and when it is likely to exhaust resources available.

There seems to be a consensus among several academics (Shang et al [20], Loboz [12] and Guidolin et al 2008 [9]) on the need for long term capacity planning and the inadequacy of capacity planning in this new age of cloud computing using current techniques. The techniques used today assume that the architecture is simple, in that the architecture does not utilise virtualisation or advanced database features such as standby's and clustering technology, but in the age of consolidation and drive for standardisation, the architecture is not simple. Enterprises use combinations of technology in different configurations to achieve their goals of consolidation or standardisation. Most models use a form of linear regression to predict growth patterns. Guidolin et al 2008 [9] conducted a study of those linear regression models and came to the conclusion that as more parameters are added the models become less accurate, something also highlighted by Mozafari et al 2013 [15]. To mitigate against this inaccuracy more controls are added at the cost of performance of the model itself. For example, predicting the growth of several databases based on resource utilisation may become more inaccurate as the number of source systems being analysed increases, therefore requiring more controls to keep the accuracy. This is certainly interesting when trying to capacity plan several applications running on different configurations prior to a migration to a cloud. In addition, trying to simulate cloud computing workloads to develop new techniques is also an issue; Moussa and Badir 2013 [14] explained that the TPC-H [4] and TPC-DS [3] benchmarks are not designed for Data Warehouses in the cloud, further adding to the problem of developing and evaluating models.

3. EXPERIMENTAL SETUP

Given a description of an existing deployment, including the Operating System, Database and Applications running on that database (Activity), a collection of monitors on the existing deployment that report on CPU, Memory, IOPS's and Storage, the goal is to develop models of the existing configuration that contain enough information to allow reliable estimates to be made of the performance of a deploy-

Workload Type	Workload Profile	DBNAME(S)	Workload Description	Number of Users	Duration (hh:mi)	Avg Transaction per sec
OLTP	General usage	RAPIDKIT RAPIDKIT2 DBM01	General Online Application with updates, inserts and deletes simulate working day	100 2000 (DBM01)	23:59	0.2
OLTP	Morning Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Morning Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 1000 (DBM01)	2:00	0.2
OLTP	Lunch Time Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Lunch Time Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 1000 (DBM01)	1:00	0.2
OLTP	Evening Time Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Evening Time Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 1000 (DBM01)	5:00	0.2
Daily OLTP Hot Backup taken at 23:00						
OLAP	Data Warehouse General Usage	RAPIDKIT RAPIDKIT2 DBM01	General Data Warehousing Application with heavy Selects taking place out of hours building Business Intelligence data	5 400 (DBM01)	8:00	0.4
Daily OLAP Hot Backup taken at 06:00						
Daily OLAP archivelog backups taken at 12:00,18:00,00:00						
DM	OLTP General Usage	RAPIDKIT RAPIDKIT2 DBM01	Combination of DML taking place during the business day and heavy DML taking out of ours	200 1000 (DBM01)	23:59	0.2
DM	OLTP Morning Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Morning Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 500 (DBM01)	2:00	0.2
DM	OLTP Lunch Time Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Morning Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 500 (DBM01)	2:00	0.3
DM	OLTP Evening Time Peak Logon Surge	RAPIDKIT RAPIDKIT2 DBM01	Evening Time Surge to simulate users logging on to the Online Application with updates, inserts and deletes	100 500 (DBM01)	5:00	0.3
DM	OLAP Batch Loads Peak	RAPIDKIT RAPIDKIT2 DBM01	Evening Time Surge to simulate users logging on to the Online Application with updates, inserts and deletes	5 400 (DBM01)	8:00	0.3
Daily DM Hot Backup taken at 06:00						
Daily DM archivelog backups taken at 12:00,18:00,00:00						

Table 1: Database Workloads

ment when it is migrated to a cloud platform that may involve a Single Database, a Clustered Database or Standby Databases. To meet this objective, we must find out if a workload executed on one database is comparable to the same workload running on the same database on a different host.

Our approach to this question is by way of empirical evaluation. Using increasingly complex deployments, of the type illustrated in Fig 2, and representative workloads, we establish the extent to which we can predict the load on a target deployment based on readings on a source deployment. This section describes the workloads and the platforms used in the experiments.

3.1 Workloads

A Workload can be described as the activity being performed on the database at a point-in-time, and essentially is broken down into the following areas:

- *Database* - An Oracle database is a set of physical files on disk(s) that store data. Data may be in the form of logical objects, such as tables, Views and indexes, which are attached to those tables to aid speed of access, reducing the resources consumed in accessing the data.
- *Instance* - An Oracle instance is a set memory structures and processes that manage the database files. The instance exists in memory and a database exists on disk, an instance can exist without a database and a database can exist without an instance.
- *Activity* - The DML (Data Modification Language)/DDL (Data Definition Language) i.e. SQL that is being executed on the database by the application, creates load consisting of CPU, memory and IOPS/s.

.

The monitors used to capture the data report on IOPS's (Physical reads and Physical Writes), Memory (RAM assigned to a database or host) and CPU (SPECINT's). SPECInt is a benchmark based on the CINT92, which measures the integer speed performance of the CPU, (Dixit) [6]. The experiments involve controlled execution of several types of workloads on several configurations of database. Moussa and Badir 2013 [14] describe how running of controlled workloads using TPC has not evolved for clouds, therefore we will use a utility called *swingbench* (Giles)[8] to generate a controlled load based on TPC-C [5]. The workload is generated on several Gb's of sample data based on the Orders Entry (OE) schema that comes with Oracle 12C. The OE schema is useful for dealing with intermediate complexity and is based on a company that sells several products such as software, hardware, clothing and tools. Scripts are then executed to generate a load against the OE schema to simulate DML transactions performed on the database of a number of users over a period of Hour.

3.2 Outline of the Platforms

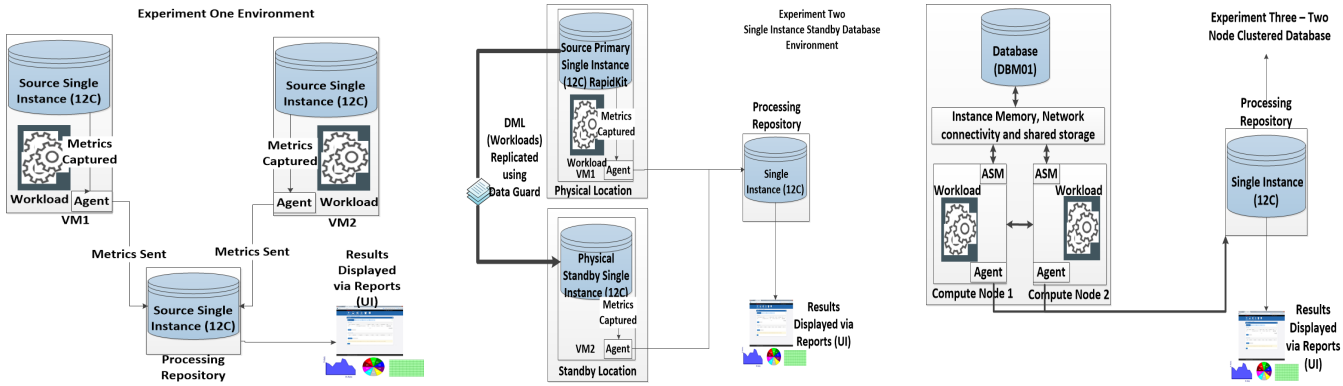
Three different types of workload were created *(OLTP, OLAP and Data Mart)* as shown in Table 1. The Database is placed in archivelog mode during each execution of the workload further creating IO on the Host and allowing for a hot backup to be performed on the database. The backup acts as a *'houskeeping'* routine by clearing down the archivelogs to ensure the host does not run out of storage space. This type of backup routine is normal when dealing with databases and each backup routine is executed periodically depending upon the workload.

4. EXPERIMENTS AND ANALYSIS

A number of experiments were conducted to investigate if a workload executed on one machine consumes similar resources when the workload is executed on another environment. The aim was to investigate what could cause dif-

VM Name	OS Type	CPU Details	Memory	Storage	Database Type	Products and Versions
Single Database Instance Configuration						
Virtual Machine 1	OEL Linux 2.6.39	4 * 2.9 Ghz	32Gb	300Gb	Oracle Single Instance Database (RapidKit)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4),
Virtual Machine 2	OEL Linux 2.6.39	4 * 2.9 Ghz	32Gb	300Gb	Oracle Single Instance Database (RapidKit2)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4),
Clustered Database Instance Configuration						
Clustered Compute Node 1	OEL Linux 2.6.39	24 * 2.9 Ghz	96Gb	14Tb	Oracle Clustered Multi-tenant Database Instance (DBM011)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4, • Grid Infrastructure (12.1.0.2), • Oracle Automatic Storage Manager (12.1.0.2),
Clustered Compute Node 2	OEL Linux 2.6.39	24 * 2.9 Ghz	96Gb	14Tb	Oracle Clustered Multi-tenant Database Instance (DBM012)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4, • Grid Infrastructure (12.1.0.2), • Oracle Automatic Storage Manager (12.1.0.2),
Standby Database Instance Configuration						
Virtual Machine 3	OEL Linux 2.6.39	4 * 2.9 Ghz	32Gb	1Tb	Oracle Single Instance Standby Database (STBYRapidKit, STBYRapidKit2)	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4),
Central Repository Details						
Storage Repository	OEL Linux 2.6.39	24 * 2.4 Ghz	32Gb	500Gb	Oracle Single Instance Database (EMREPCTA)	• Enterprise Edition (11.2.0.3), • Enterprise Manager R4 including Webserver and BIPublisher (12.1.0.4), • Enterprise Manager Agent (12.1.0.4),

Table 2: Platform Outline



(a) Single Instance (b) Single Instances with Standby Databases (c) Two Node Clustered Database

Figure 2: Experiment Architecture: different database combinations used for experiments.

ferences in the consumption of resources between workloads. The experiment focused on three types of database configuration:

- *Experiment 1* - Running three workloads *(OLTP, OLAP and DM)* on a single instance database.
- *Experiment 2* - Running the workloads *(OLTP, OLAP and DM)* on a single instance database with a Physical Standby Database.
- *Experiment 3* - Running the three workloads *(OLTP, OLAP and DM)* on a two node clustered database.

The database was always the same version between each host, the data set was always the same size to start, the workload was always repeatable in that the workload could be executed, stopped, the database reset and the same workload replayed.

4.1 Experimental Methodology

The experiments involve an eclectic set of hardware configured to run several different types of database as shown in Table 2. An agent polls the database instance every few minutes for specific metrics namely; Database Instance Memory, IOPS's (physical Reads/Writes) and CPU per sec. The metric results are stored in a central repository database, and are aggregated at hourly intervals. The configuration of the hardware, such as CPU Make model and SPECInt, and the database configuration are also stored in a central repository, which is then used as lookup data when performing comparisons between the performance of one workload on one database with the same workload on another database.

4.2 Experiment One - Single Database Instance

The first experiment was to execute three workloads on one single instance database on a virtual host (VM1) and

then execute the same three workloads on another single instance database on another virtual host (VM2) as shown in Fig 2a. The database configurations were the same in Instance Parameters, Software Version and Patch Level. The Hardware configurations were the same in OS Level, Kernel Version, and memory configuration. Some differences exist in the underlying architecture such as the Physical hardware and the Storage as these where VM's created on different physical machines. We capture the metrics for each workload and analyse the extent to which physical properties are consistent across platforms. This is shown graphically in Fig 3

4.3 Results and Analysis Experiment One - OLTP Workload

The results for OLTP, covering Memory, CPU and IOPS/s are shown graphically in Fig 3. These are simple line graphs from the OLTP workload shown in Table 1. It was observed that the OLTP workload from a CPU perspective had several distinguishing features. It clearly shows that the workload starts off low until the beginning of the experiment where a sudden jump takes place and the OLTP workload begins. Then there is a general plateau that relates to the 24 hour periods and at various times from there on in there are spikes.

- *CPU utilisation* - CPU over a 72 hour period was not the same between the two databases but at it largest peak (evening surge) there was a difference of approximately 300 SPECInts or +88% (day 24 hour 11) in its utilisation. The difference in utilization between the two workloads without the peaks was approximately +20%.
- *CPU Spikes (Backup)* - There were several spikes in CPU at 00:00 - 02:00 and relate to the daily hot RMAN backup that is taken for the databases.
- *CPU Spikes (Morning Surge)* - A large CPU spike was observed for several hundred users accessing the database at 08:00.
- *IOPS/s (general)* - There is a large difference in IOPS (day 23 hour 9) where the difference at peak is +88%. The difference in general usage (i.e. without the peaks) was +7%.

CPU, Memory and IOPS/s over a 72 hour period show similar traits in that the workload begins and there is a jump in the activity as the users logon. The first set of results show that even when executed on similar platforms, the metrics for the OLTP workloads can be substantially different, especially in the CPU and IOPS utilisation.

4.4 Results and Analysis Experiment One - OLAP Workload

The results for OLAP covering Memory, CPU and IOPS/s are shown graphically in Fig 4. The difference between the OLTP and OLAP workload is that the OLAP workload is high in Select statements and the result set is larger. The IO is representative of a Data Warehouse building cubes for interrogation by a Business Intelligence reporting tool. The execution times for the workload are also different; OLTP is fairly constant in its usage, whereas OLAP is more concentrated out of normal working hours. It was observed that the OLAP workload runs out of hours for a period of around five hours and this matches the description shown in Table 1.

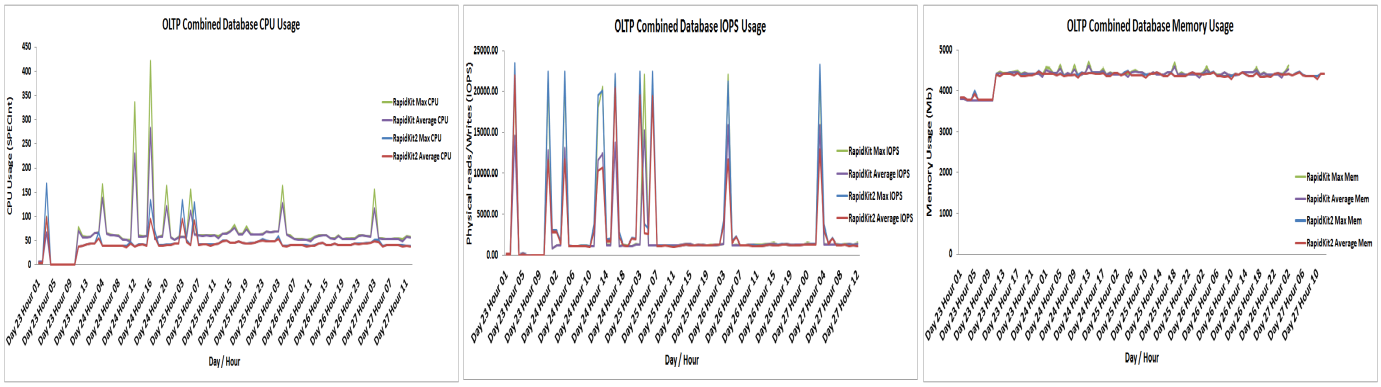
- *CPU Spikes (General Usage)* - CPU over a 72 hour period was not the same for the two databases, but at it largest peak there was a difference of only +1% (day 17 hour 05) in utilisation. Two workloads outside the peaks were essentially the same.
- *IOPS/s utilisation* - IOPS over a 72 hour period had a difference of approximately +50% in utilisation (Day 16 Hour 8); outside the peaks (Day 16 Hour 19) the utilisation is 0%.
- *IOPS/s Spikes (Backup)* - There are four backups that run during the 24 hours. Three of those backups are used as housekeeping routines that backup and delete the archivelogs; these backups are executed at 12:00, 18:00 and 00:00. One backup backs up the database (level-0) and the associated archivelogs, and this is executed at 06:00. There was no spike for 18:00 because the backup at 12:00 had removed the archivelogs and thus there was nothing to backup.

The OLAP Memory chart also showed the same characteristics as the IOPS/s and CPU charts in that there is a uniform pattern to there being a plateau and a spike over the 72 hours. Each of the databases had a memory configuration of 3.5Gb, given the OLAP workload would have had SQL requiring larger memory than 3.5Gb for sorting, thus sorts would have gone to disk rather than memory, accounting for the higher IOPS's readings in Fig 4 than in Fig 3.

4.5 Results and Analysis Experiment One - DataMart Workload

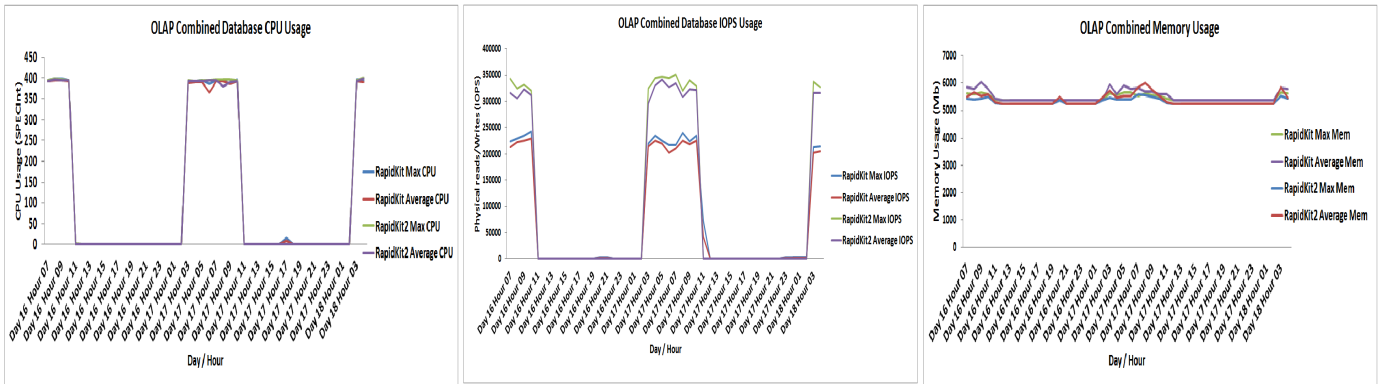
The results for the Data Mart covering Memory, CPU and IOPS/s are shown in Fig 5. It was observed that the Data Mart workload from a CPU perspective had several distinguishing features. It clearly shows that the workload starts off as the users connect and the workload is running, a sudden jump takes place at Day 10 Hour 3 as the Batch Loads are executed for approximately 6 hours, and this is repeated twice more throughout the 72 hours. There are also other peaks and troughs observed and these are consistent with the workload described in Table 1.

- *CPU utilisation* - CPU over a 72 hour period between the two databases and had a difference of approximately +64% during the normal day (Day 9 Hour 21). When the batch loads ran (Day 11 Hour 05) the difference in utilisation was +1%.
- *CPU Spikes (General)* - generally, the CPU utilisation between the two databases was the same, there is a difference of +1% at peak times.
- *IOPS/s Utilisation* - IOPS at peak (Day 9 Hour 21) had a difference of approximately +24%
- *Memory utilisation* - Memory was the same in general footprint however there were differences at peaks times of 300mb or +4%



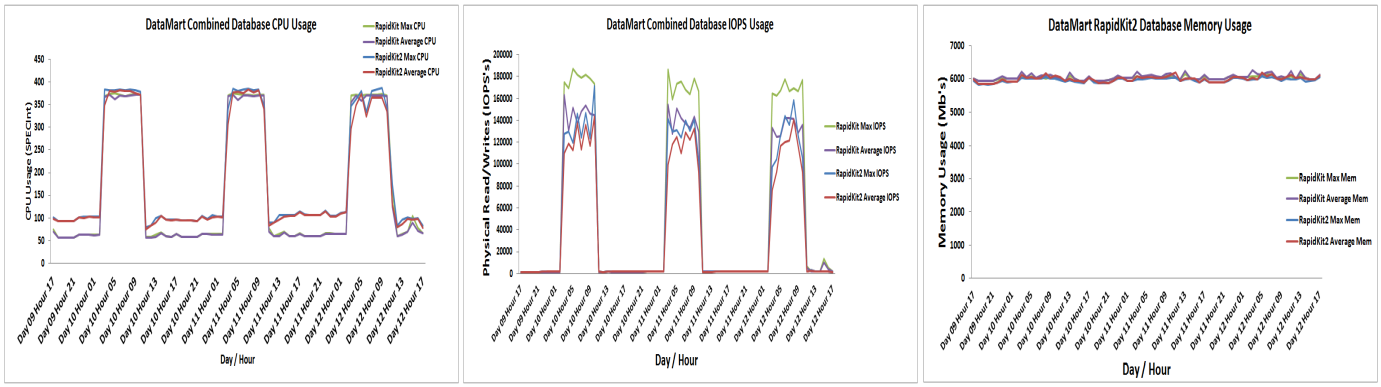
(a) CPU 72 hours (b) IOPS's 72 Hours (c) Memory 72 Hours

Figure 3: Results Single Instance OLTP: workload patterns for the 72 hour period.



(a) CPU 72 hours (b) IOPS's 72 Hours (c) Memory 72 Hours

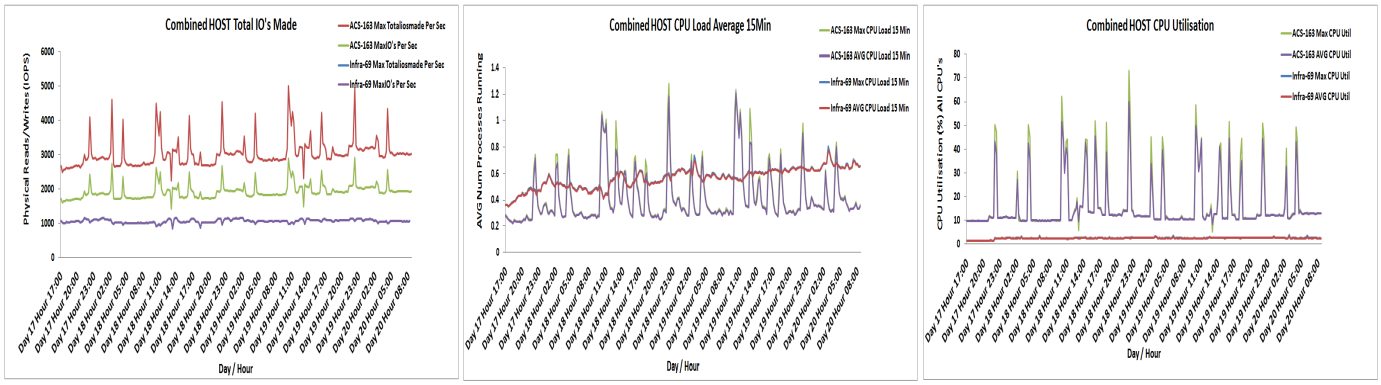
Figure 4: Results Single Instance OLAP: workload patterns for the 72 hour period.



(a) CPU 72 hours (b) IOPS's 72 Hours (c) Memory 72 Hours

Figure 5: Results Single Instance Data Mart: workload patterns for the 72 hour period.

In general there is a difference in the VM's at a CPU level. The VM named *acs-163* has a configuration of 16 Threads(s) per core (based on the lscpu command) from the VM *infra-69* which only has 1 thread per core. We believe this accounts for the difference in CPU for small concurrent transactions in the OLTP workload. Each of the databases had a memory *(SGA)* configuration of 3.5Gb, if the SQL statement executed in the workload requires a memory larger than 3.5Gb, which is more common in OLAP and Data Mart workloads then sorts will go to disk. Database memory configurations influence the database execution plans and optimisers and this sensitivity is reflected in the IOPS's charts shown in Fig's 3b, 4b and 5b.



(a) Host Total IO's Made 72 hours (a) Host CPU Load Avg (15Mins) 72 hours (b) Host CPU Utilisation 72 Hours

Figure 6: Results HOST Metrics OLTP: workload patterns for the 72 hour period.

4.6 Experiment Two - Single Instance Standby Configurations

The Second set of experiments was to introduce a more complicated environment executing one workload *(OLTP)* on a single instance *primary* database with a physical standby database kept in sync using the Data Guard technology (Oracle Data Guard [18]) across the two sites, as shown in Fig 2b. A key factor in this experiment is that the physical standby database is always in a recovering state and therefore is not opened to accept SQL connections in the same way as a normal *(primary)* database. Therefore the agent is unable to gather the instance based metrics, so we capture host based metrics to compare and contrast the workload:

- CPU load over 15mins - This is the output from the *"Top"* command executed in linux, this measurement is a number using or waiting for CPU resources. For example if there is a 1, then on average 1 process over the 15 min time period is using or waiting for CPU.
- CPU Utilisation Percentage - This is based on the *"MPSTAT -P ALL"* command and looks at the percentage of all cpu's being used .
- TotalIOSMade - This is the total physical reads and total physical writes per 15 minute interval on the host.
- MaxIOSperSec - This is the Maximum physical reads and physical writes per sec.

The two VM's are located within the same site but in different rooms, Data Guard is configured using Maximum Performance mode to allow for network drops in the connectivity between the two physical locations. The database configurations were the same in Instance Parameters, Software Version and Patch Level. The Hardware configurations were the same in OS Level, Kernel Version and memory configuration. We capture the metrics of each workload and analyse the consistency of the metrics, as shown graphically in Figure 6.

4.7 Results and Analysis Experiment Two - OLTP Workload

The results for OLTP covering CPU and IOPS/s are shown graphically in Figure 6. Relying on host based metrics has a profound effect in the ability to compare and contrast different CPU models, as there is no common denominator (SPECInt) calculated. It also becomes difficult if there are multiple standby databases existing in the same environment. When the workloads were compared between the hosts, due to the nature of the physical standby and the primary behaving, as designed, in a completely different way, the graphs clearly show that the standby database has a considerably lower utilisation of CPU and IO resources. This is for several reasons:

- A physical Standby Database is in recovery mode therefore is not open for SQL DML or DDL in the same manner as a primary database is opened in normal mode. Therefore processes are not spawned at OS level/Database level, consuming resources such as Memory, CPU.
- A Physical standby applies *"Archivelogs"* and therefore is much more dependent on Physical Writes as these logs *(changes)* are applied on the standby from the primary database, therefore less IO load is generated.
- The reduction in IOPS/s is also attributed to DML/DDL is not being executed on the standby database in the same manner as a primary database (e.g. rows are not being returned as part of a query result set).

It was clear after the first experiment *OLTP*, that the workloads would be profoundly different in their footprint regardless of the workload being executed, so we have not included the results of the other workloads namely, OLAP and Data Mart.

4.8 Experiment Three - Clustered Database (Advanced Configuration)

The final set of experiments was to execute three the workloads on a more advanced configuration, a two-node clustered database running in an Engineered system (Exadata X5-2 platform) [1], illustrated in Fig 2. During the experiment, compute nodes are closed down to simulate a fail-over. The database configurations were the same in Instance Parameters, Software Version and Patch Level. The hardware configurations were the same in OS Level, Kernel Version and memory configuration. A difference in this experiment

from the previous two is that the physical hardware and database are clustered. In this experiment we leverage the Exadata Technology in the IO substructure.

4.9 Results and Analysis Experiment Three - OLTP Workload

The results for OLTP covering Memory, CPU and IOPS/s are shown graphically in Fig 7. The OLTP workload was amended to run from node 1 for the second 24 hours and this is reflected in all three of the graphs, when the instance DBM012 is very much busier than instance DBM011. The workloads are then spread evenly for the following 48 hours.

- *CPU utilisation* - for the first 24 hours, the workloads were executed fairly evenly across the cluster with a workload of 2000 users connecting consistently with peaks of 1000 users at peak times, and the CPU showed similar patterns during the workload execution.

- *CPU utilisation* - When the workload ran abnormally and all users (3000 users) ran from one node, in the second 24 hours, then the CPU utilisation did almost double in usage as expected. The increase was approximately +99% (Day 7 Hour 15)

- *IOPS/s* - The IOPS's utilisation for the first 24 hours was similar, as expected, when the workloads were evenly spread. However when the workloads were run from node 2 in the second 24 hours the IOPS increase significantly, as expected. The IOPS during the failure period was as expected, an increase of +99% (Day 7 Hour 15).

- *IOPS/S Spike* - there are two major spikes occurring at Day 7 Hour 2 and Day 8 Hour 2, these are Level 0 database backups than only run from node 1 (DBM011)

- *Memory Consumption* - The maximum memory utilisation across both instances was consistent during the first 24 hours when the workload was evenly spread. The memory configuration on DBM012 is sufficient to handle the 3000 users during the failover period, although the increase in memory used on DBM012 was only +45%

In general, the conclusion from this experiment when executing the OLTP workloads was, it cannot be assumed that when a workload fails over from one node (database instance) to another node (database instance) the footprint will be double in terms of Memory. The workload did double for CPU and IOPS/s. The results show there is an increase in IOPS/s, Memory and CPU. The difference during normal running conditions (i.e. when workloads are evenly spread) was the following: +31% (Day 7 Hour 3) CPU, +2% Memory (Day 6 Hour 21) and +1% (Day 6 Hour 12) IOPS. When the workload failed over there was a difference of +97% (Day 7 Hour 9) CPU, +99% (Day 7 Hour 20) Memory and +99% (Day 08 Hour 10) IOPS. There are two large spikes at Day 7 Hour 2 and Day 8 Hour 2; these are Level 0 RMAN backups which account for the large IOPS readings. The database instance was sufficiently sized to handle both workloads otherwise we would of expected to see out of memory errors in the database instance alert file.

4.10 Results and Analysis Experiment Three - OLAP Workload

The results OLAP covering Memory, CPU and IOPS/s are shown graphically in Fig 8. The OLAP workload was amended to run from node 1 for the first 24 hours and this is clearly reflected in all three of the graphs, as the instance DBM011 is very much busier than instance DBM012 during this period. The workloads are then spread evenly for the following 48 hours.

- *CPU utilisation* - for the first 24 hours, node 1 ran the whole workload of 400 users and thus the DBM011 instance is busier compared with the workload across days two and three; as expected, utilization is effectively doubled, at +99%.

- *CPU utilisation* - when the workload ran normally (400 users) across both nodes then the utilisation was similar in its SPECint count with a difference of approximately +20%.

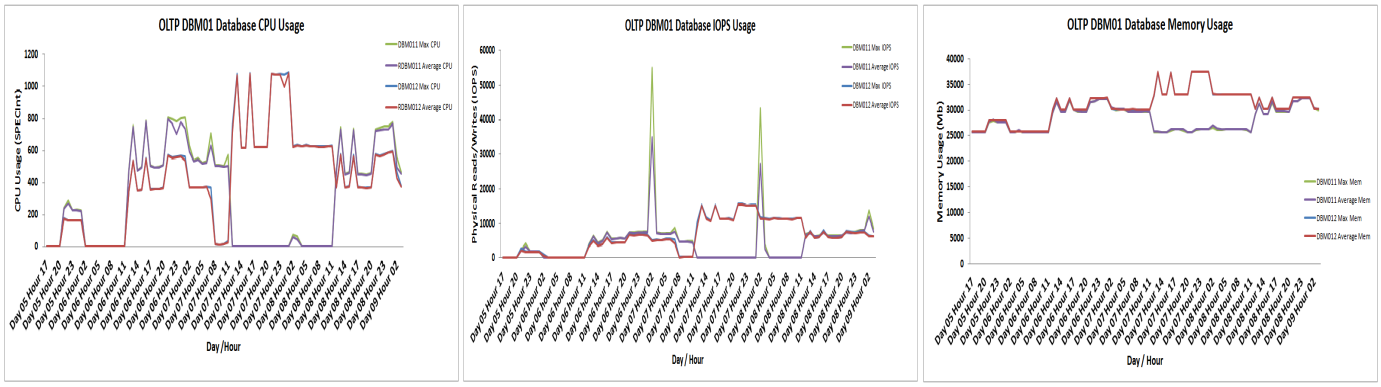
- *IOPS/s* - The IOPS's utilisation for the first 24 hours was busier on node 1, as expected, than node 2 given that both workloads were executed from DBM011 instance. The IOPS utilisation was almost double +99% (Day 25 Hour 05) the amount from the second period of time (Day 26 Hour 05) when the workloads were spread evenly across both instances.

- *Memory Consumption* - The maximum memory utilisation observed across both instances was consistent with the workload, the first 24 hours when the workload ran from node 1 is as expected in that there was sufficient memory to serve both workloads. However there is a difference of +55% (Day 25 Hour 04) in memory between nodes 1 and 2. For the second 24 hours, as the workloads reverted back to their normal hosts I.E. spread evenly across both nodes, their utilisation is similar with a difference of +1% (Day 26 Hour 04) between the nodes in memory utilisation.

In general, the conclusion from this experiment when executing the OLAP workloads was that it cannot be assumed that when a workload fails over from one node (database instance) to another node the footprint will be double in terms of Memory. For the metrics IOPS and CPU the increase was almost double; CPU had a difference of +99% (Day 25 Hour 04) and IOPS +99% (Day 24 Hour 04). When the workload was spread evenly across both nodes the differences between the nodes where CPU +20% (Day 26 Hour 3), Memory +2% (Day 26 Hour 3) and IOPS +1% (Day 26 Hour 4). The database instance was sufficiently sized to handle both workloads otherwise we would of expected to see out of memory errors in the database instance alert file.

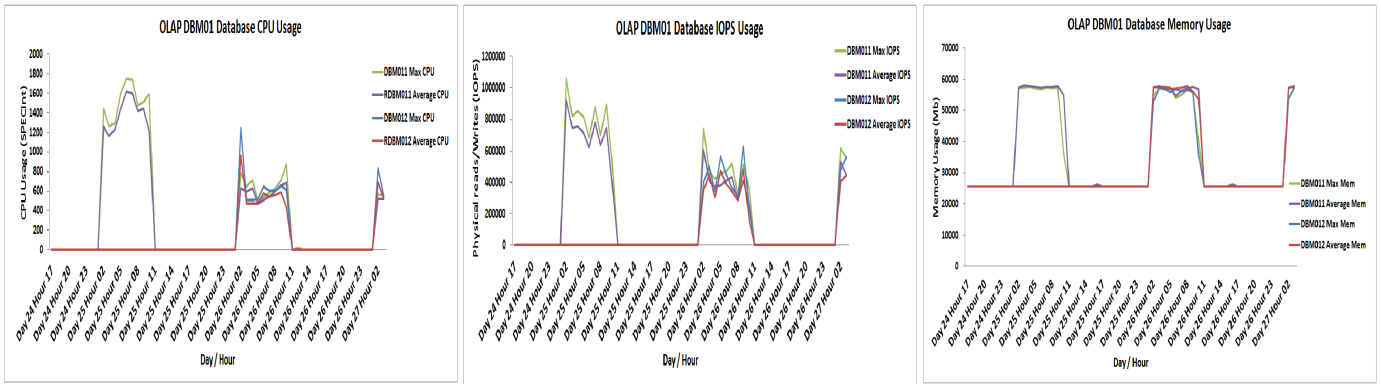
4.11 Results and Analysis Experiment Three - Data Mart Workload

The results are as follows for the Data Mart workloads covering Memory, CPU and IOPS/s, as shown graphically in Fig 9. The Data Mart workload was run normally for the first 24 hours, which is reflected in the workloads being similar for this period. A simulated failure of database instance DBM011 is then performed and all connections then failover to DBM012 on node 2 for the second 24 hours. This is



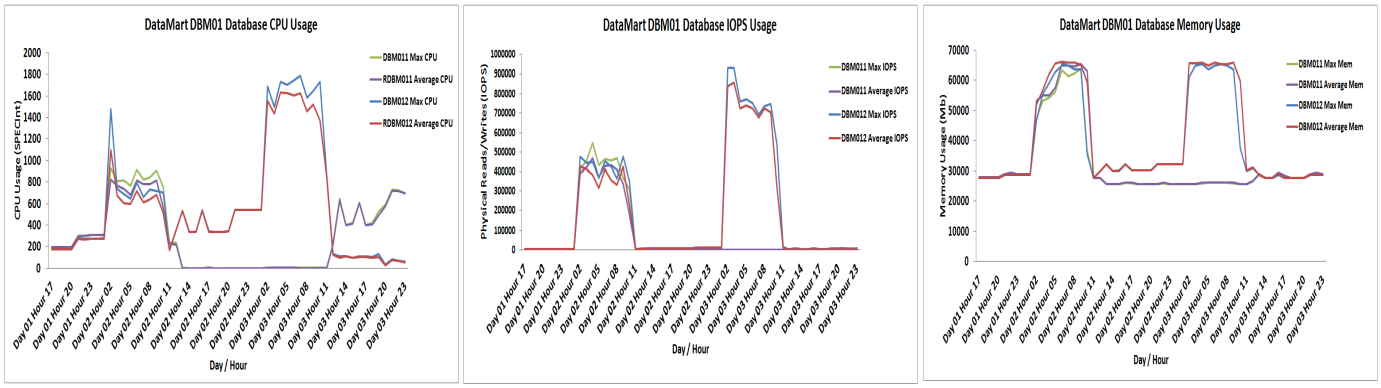
(a) CPU 72 hours (b) IOPS/s 72 Hours (c) Memory 72 Hours

Figure 7: Results RAC OLTP: workload patterns for the 72 hour period.



(a) CPU 72 hours (b) IOPS/s 72 Hours (c) Memory 72 Hours

Figure 8: Results RAC OLAP: workload patterns for the 72 hour period.



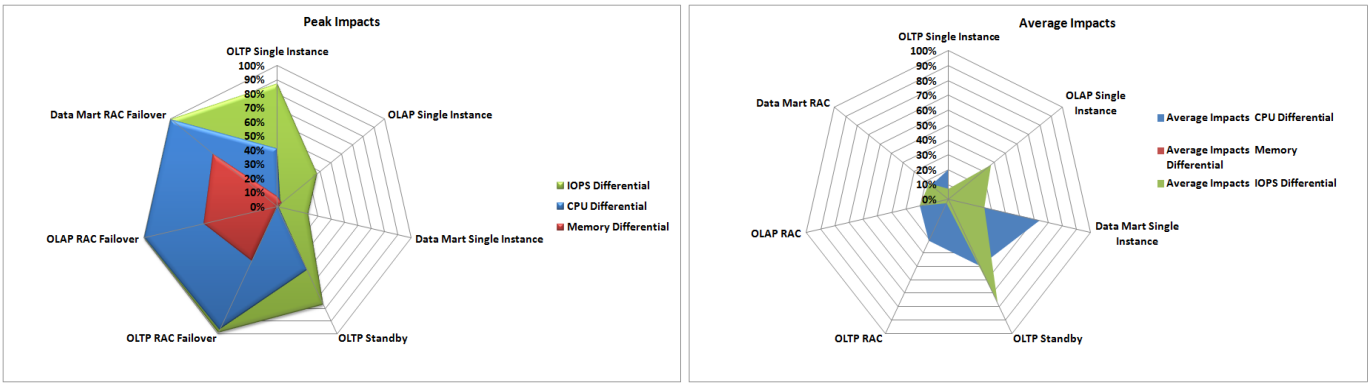
(a) CPU 72 hours (b) IOPS/s 72 Hours (c) Memory 72 Hours

Figure 9: Results RAC Data Mart: workload patterns for the 72 hour period.

reflected in all three of the graphs as the instance DBM012 becomes much busier than instance DBM011.

- *CPU utilisation* - For the first 24 hours, the workloads were executed fairly evenly across the cluster with a workload of 2700 users connecting at different times from the two nodes and the SPECInt count was similar with a average CPU difference of +15% (Day 2 Hour 04).

- *CPU utilisation* - When the workload ran abnormally and all users (2700 users) ran from one node, in the second 24 hours, then the CPU utilisation almost doubled in usage as expected +99% (Day 3 Hour 04).



(a) Volatility of workload Peak

(b) Volatility of workload Avg

Figure 10: Workload Impacts

- *IOPS/s* - The IOPS's utilisation for the first 24 hours was similar, as expected, when the workloads were evenly spread with a difference on average of +17% (Day 2 Hour 04). However, when the workloads were run from node 2 in the second 24 hours the IOPS increased significantly, rising to almost double at +99% (Day 3 Hour 04).

- *Memory Consumption* - The maximum memory utilisation across both instances was as expected during the first 24 hours, when the workloads were evenly spread, showing a difference of +9% (Day 2 Hour 04). This behaviour was not expected during the failover period when all users execute their workload on DBM012 as the utilisation difference is +60% (day 3 Hour 04). The memory configuration on DBM012 is sufficient to handle the 2700 users.

In general, the conclusion from this experiment when executing the Data Mart workloads was, it cannot be assumed that when a workload fails over from one node (database instance) to another node the footprint will be double in terms of memory, as it only increased by approximately +60%. CPU and IOPS however, did double in its usage to approximately +99%. When the workload was spread evenly the average utilisation had a difference of CPU +15% (Day 2 Hour 04), Memory +9% (Day 2 Hour 04) and IOPS +17% (Day 2 Hour 04).

5. CONCLUSIONS AND FUTURE WORK

From the experiments conducted and the model we proposed, we conclude that capacity planning of databases that employ advanced configurations such as Clustering and Standby Databases is not a simple exercise. Taking the Average and Maximum readings for each metric (CPU, Memory Utilisation and IOPS) over a period of 72 hours, the outputs are volatile. One should not assume that a workload running on one database instance configured in one type of system will consume the same amount of resource as an another database instance running on another system, regardless of similarity; this is clearly shown in Fig 10 (a) (OLTP, OLAP, Data Mart RAC Failovers). These charts show us that as workloads become assimilated they completely change as the difference grows, sometimes considerably. The differences between the footprints based on configuration can vary between +10% (CPU OLAP RAC) in normal circumstances shown in Fig 10 (b) to 99% (CPU OLAP RAC) as shown in Fig 10 (a). Fig 10(a & b, OLTP Standby) also highlights that configuration has a big impact on capacity planning databases with advanced configurations, such as standby databases.

In this paper we highlighted the problems that organisations are faced with *over-estimation* and *under-estimation* when trying to budget on non-cloud compliant financial models such as capex or cloud compliant models, which are subscription based. Accurate capacity planning can help in reducing wastage when metrics are captured and the assumption of workloads being the same is not employed. Capturing and storing the data in a central repository, like the approach we proposed, allowed us to mine the data successfully without the labour intensive analysis that often accompanies a capacity planning exercise.

The main points from this work are.

1. When capacity planning DBaaS, it should be done on a *instance-by-instance* basis and not at a database level - this is especially the case in clustered environments where workloads can move between one database and another or fail-over technology is employed.

2. Metrics need to be captured at different layers of the infrastructure in advanced configurations, for example in the storage layer, caching can mask IOPS causing the workload to behave differently.

3. Hypervisors and VMManagers can influence capacity planning as these tools allocate resource. For example, a CPU can be dissected and allocated as a vcpu (Oracle VM) [2]. How does one know that the CPU assigned is a full CPU? The Oracle Software and the database itself may assume that a full CPU was made available, when in fact it was assigned 0.9 of a CPU due to overheads.

4. CPU configuration *(Thread(s) per core)* within a VM has a profound effect when capacity planning. We observed in experiment one (OLTP and Data Mart) that small concurrent transactions in the OLTP workload executed on VM acs-163 were a lot more efficient than

the same workload executed on another VM with lower thread(s) per core, and this is reflected in Figures 3, 4 and 5.

5. SPECInt benchmark is a valid benchmark when comparing one varient of CPU with another, especially when trying to capacity plan databases with a view to a migration or upgrade of the infrastructure.

6. Standby Databases presented a different footprint. A standby database is always in a *mounted* state and therefore is configured in a recovering mode by applying logs or changes from the primary. It should not be assumed that the footprints are the same.

7. In environments that employ standby database configurations, metrics that are available for collection on the primary database are not available on the standby, namely physical reads/writes, CPU and memory, thus gathering accurate metrics is impractical. Metrics can be gathered at a host level, however if multiple standby databases are running on the same host this makes reconciliation of which database is using what more challenging.

8. In environments that employ clustered databases, if a workload running on one node fails-over from another node within the cluster, one should not assume that the properties of the composed workload will follow obviously from its constituents. Upon failover, the workload from the failing node is assimilated, with the result being the formation of a completely new footprint.

Future work is to conduct the same type of experiments between different database versions, for example a workload running on Oracle Database Version 10G/11G and Oracle Database Version 12C, analysing if the internal database algorithms have any influence and by how much. However techniques already exist that go some way to answering this question through the use of a product called Database replay [7]. Being able to gather metrics from a standby database instance for CPU, IOPS and Memory is critical for our model as this would allow us to accurately analyse the CPU such as SPECInt, Memory and IOPS's. We could configure a custom metric to execute internal queries against the standby database, and this is now in the design phase, but until then capacity planning architectures with standby database will need to rely on host metrics.

6. REFERENCES

[1] U. Author. Oracle exadata database machine. *Security Overview*, 2011.

[2] O. Corporation. Oracle vm concept guide for release 3.3.

[3] T. P. P. Council. Tpc-ds benchmark.

[4] T. P. P. Council. Tpc benchmark h standard specification version 2.1. 0, 2003.

[5] T. P. P. Council. Tpc benchmark c (standard specification, revision 5.11), 2010. *URL: http://www. tpc. org/tpcc*, 2010.

[6] K. M. Dixit. Overview of the spec benchmarks., 1993.

[7] L. Galanis, S. Buranawatanachoke, R. Colle, B. Dageville, K. Dias, J. Klein, S. Papadomanolakis, L. L. Tan, V. Venkataramani, Y. Wang, and G. Wood. Oracle database replay. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1159–1170, New York, NY, USA, 2008. ACM.

[8] D. Giles. Swingbench 2.2 reference and user guide.

[9] M. Guidolin, S. Hyde, D. McMillan, and S. Ono. Non-linear predictability in stock and bond returns: When and where is it exploitable? *International journal of forecasting*, 25(2):373–399, 2009.

[10] H. Hacıgümüş, J. Tatemura, Y. Chi, W. Hsiung, H. Jafarpour, H. Moon, and O. Po. Clouddb: A data store for all sizes in the cloud. *Internet: http://www. neclabs. com/dm/CloudDBweb. Pdf,[January 25, 2012]*, 2012.

[11] Y. Kouki and T. Ledoux. Sla-driven capacity planning for cloud applications. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 135–140. IEEE, 2012.

[12] C. Loboz. Cloud resource usageĂŤtailed distributions invalidating traditional capacity planning models. *Journal of grid computing*, 10(1):85–108, 2012.

[13] S. Mahambre, P. Kulkarni, U. Bellur, G. Chafle, and D. Deshpande. Workload characterization for capacity planning and performance management in iaas cloud. In *Cloud Computing in Emerging Markets (CCEM), 2012 IEEE International Conference on*, pages 1–7. IEEE, 2012.

[14] R. Moussa and H. Badir. Data warehouse systems in the cloud: Rise to the benchmarking challenge. *IJ Comput. Appl.*, 20(4):245–254, 2013.

[15] B. Mozafari, C. Curino, A. Jindal, and S. Madden. Performance and resource modeling in highly-concurrent oltp workloads. In *Proceedings of the 2013 acm sigmod international conference on management of data*, pages 301–312. ACM, 2013.

[16] B. Mozafari, C. Curino, and S. Madden. Dbseer: Resource and performance prediction for building a next generation database cloud. In *CIDR*, 2013.

[17] J. Murphy. Performance engineering for cloud computing. In *European Performance Engineering Workshop*, pages 1–9. Springer, 2011.

[18] A. Ray. Oracle data guard: Ensuring disaster recovery for the enterprise. *An Oracle white paper*, 2002.

[19] S. Sahri, R. Moussa, D. D. Long, and S. Benbernou. Dbaas-expert: A recommender for the selection of the right cloud database. In *International Symposium on Methodologies for Intelligent Systems*, pages 315–324. Springer, 2014.

[20] S. Shang, Y. Wu, J. Jiang, and W. Zheng. An intelligent capacity planning model for cloud market. *Journal of Internet Services and Information Security*, 1(1):37–45, 2011.

[21] T. Yu, J. Qiu, B. Reinwald, L. Zhi, Q. Wang, and N. Wang. Intelligent database placement in cloud environment. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 544–551. IEEE, 2012.

5.2 Database Workload Capacity Planning using Time Series Analysis and Machine Learning

Antony S. Higginson, Mihaela Dediu, Octavian Arsene, Norman W. Paton, Suzanne M. Embury.

In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data,
Published by ACM,
June 2020,
Pages 769–783,
https://doi.org/10.1145/3318464.3386140.

Summary: This paper creates and evaluates valid solutions to performing accurate forecasts on metric data that exhibit complex data structures and leads on from previous work. We empirically evaluate the techniques on real world data from real world configurations and the novel idea created a patent filed by Oracle in the US and EU. This work was published in a top Database Technology journal.

Approaches: Our approach in this paper was to identify a particular type of model(s) that can form a prediction on complex data signals identified from the metrics captured (CPU, Memory and IO) from the first paper. There are many models and techniques available, such as Massive On-line Analysis (MOA) [oW22]. However, the data signal was complex in that it showed trend, patterns and exogenous variables (Shocks). Streaming, Online/Offline Analytical Tooling are provided by many vendors but we want to evaluate the actual time series techniques themselves and not just leverage an existing model. We evaluated several particular time series models such as ARIMA, SARIMA and SARIMAX. Due to the nature and complexity of the data signal we also evaluated TBATS as the latest technique, which combines Fourier Transforms. To evaluate the accuracy of the model(s) we used the established techniques called RMSE and MAPE to calculate the accuracy of the model as a mean over the whole data signal against the actual prediction line. The times series technique that displays the highest MAPE or RMSE is deemed the most accurate technique for that type of workload (OLTP, DM or OLAP). A key component of the evaluation was the ability of the model(s) to predict traits such as trend, shocks and patterns in their forecast. The model was evaluated to account for these traits?

Comments on authorship: I proposed the main idea of the paper, developed and validated the main approach, conducted an empirical evaluation, provided and analysed results, investigated related work, all graphics, participated in the entire writing processes and addressed any comments. I created the environments, wrote any code in the form of python forecast models. I Configuring any operating systems and databases as required and acted as the DBA to monitor the environments as the experiments ran. Specific python code was written to cover TBATS Models with the help of an Oracle Software Engineer (Octovian Arsene) and Fourier Transforms was helped by Oracle Software Engineer (Mihaela Dediu) to ensure we followed Oracle SDLC. I presented the work at the SIGMOD remote Conference. My supervisors, Norman Paton and Suzanne Embury also contributed to the idea, experiment design, literature research and analysis. They also proof read the paper and approved the final submission. They also guided the whole research process.

Key Contribution: See Sections 1.9.3 and 6.1

Impact Factor:(Dec 2022) Resurchify shows the Impact score - 4.41, h-index 166, SJR 2.273, Google Scholar shows citations 14, downloads 744

Database Workload Capacity Planning using Time Series Analysis and Machine Learning

Antony S. Higginson
Oracle Advanced Customer Services
Manchester, United Kingdom
antony.higginson@oracle.com

Mihaela Dediu
Oracle Advanced Customer Services
Manchester, United Kingdom
mihaela.dediu@oracle.com

Octavian Arsene
Oracle Advanced Customer Services
Bucharest, Romania
octavian.arsene@oracle.com

Norman W. Paton
University of Manchester
Manchester, United Kingdom
norman.paton@manchester.ac.uk

Suzanne M. Embury
University of Manchester
Manchester, United Kingdom
suzanne.m.embury@manchester.ac.uk

ABSTRACT

When procuring or administering any I.T. system or a component of an I.T. system, it is crucial to understand the computational resources required to run the critical business functions that are governed by any Service Level Agreements. Predicting the resources needed for future consumption is like looking into the proverbial crystal ball. In this paper we look at the forecasting techniques in use today and evaluate if those techniques are applicable to the deeper layers of the technological stack such as clustered database instances, applications and groups of transactions that make up the database workload. The approach has been implemented to use supervised machine learning to identify traits such as reoccurring patterns, shocks and trends that the workloads exhibit and account for those traits in the forecast. An experimental evaluation shows that the approach we propose reduces the complexity of performing a forecast, and accurate predictions have been produced for complex workloads.

ACM Reference Format:
Antony S. Higginson, Mihaela Dediu, Octavian Arsene, Norman W. Paton, and Suzanne M. Embury. 2020. Database Workload Capacity Planning using Time Series Analysis and Machine Learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3318464.3386140

SIGMOD'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3386140

1 INTRODUCTION

As enterprises strive to adopt cloud technologies, one of the key drivers is agility. However, organisations often underestimate the costs that can emerge with *pay-as-you-go* subscription based models. Prior to cloud, in organisations with on-premises architectures, the administration teams would provision environments. However, this was often seen as slow and cumbersome, which elongated development lifecycles to the detriment of the business. Cloud now addresses the need for speed and agility, while potentially opening the doors to unintended consequences in terms of cost. For example, there may be a lack of a regulatory framework within an organisation to oversee the acquisition of cloud resources. This lack of financial and regulatory discipline is not the sole cause of spiralling costs. For every environment provisioned, a proportion of *that* provisioned resource will probably never be used, therefore provisioning the correct shape (in terms of CPU, Memory and Storage) of cloud resource is paramount. When done correctly, Cloud Financial Management can deliver environments that scale in a way that shows promise for increasing business agility.

For Cloud Service Providers (CSPs), a burning issue becomes one of stability for short, medium and long term resource allocation. Having enough resources to satisfy consumer demand without compromising SLAs is paramount. The paradigm for creating and maintaining infrastructure has moved from the consumer (individual I.T. estates) to the CSP, where the I.T. estates become mega-estates with data centres dotted around the world that satisfy network delays. This shift in responsibility, although attractive to the consumer, comes at a revenue cost to the CSP's who, it could be argued, offset this cost back to the consumer. Therefore a

dance is created where capacity planning suddenly becomes a focal point.

Most current work on cloud modelling is at the infrastructure level, building on modelling of virtual machines (VMs). Forecasting at this level bundles a range of things together because a workload can mask what is really going on at the database level. For example, the sum of CPU being used will be shown, but only some of that will be assigned to the database. From a capacity planning perspective this means we are accounting for CPU that the database is not using, as it is being used at the VM level, and from a monitoring persepctive we cannot decipher how much CPU is assigned to the database and if the database CPU growth is specifically increasing. Provisioning for DBaaS presents short and long term challenges:

- *short term* - predicting when you will run out of resources: proactive monitoring.
- *long term* - capacity planning, whether that is on-premises, off-premises or non-cloud environments.

In this paper, we present techniques that are applicable to different prediction durations, but the experiments focus on short term capacity planning, with a view to minimizing over provisioning through timely allocation of the necessary resources.

There has been significant work on modelling and provisioning for VMs and DBs. We are particularly interested in time series analysis because recurring patterns and trends are key features in both short and long term decision making for DBaaS. Key features such as peaks, troughs and trends are all exhibited when workloads are analysed over time. In this paper we investigate the application of time series analysis techniques to represent database workloads. For example, OLTP and OLAP workloads tend to have different footprints and exhibit different behaviours. Specifically, we investigate the applicability of different time series modelling techniques with differing levels of seasonality and growth.

The contributions of this paper are as follows:

- An investigation into the application of time series modelling techniques such as ARIMA and HES to workflow predictions for OLTP and OLAP workloads.
- A proposal for the self-selection and self-configuration of models such as ARIMA and HES for use with a given workload.
- An empirical evaluation with controlled workloads at the database layers of the technological stack such as Database Instances.
- A demonstration of the applicability of the approach in real world workloads hosted by Oracle Advanced Customer Services.

2 RELATED WORK

In this section, we discuss related work of relevance to this paper under the headings of *workload modelling* and *prediction in clouds*.

Currently there is no standard for making predictions that try to answer *"how much resource do I need?/when will I require more resource?"* type questions. Some proposals suggest that predictions should be done at a particular "*layer*" of the technological stack such as a VM. Some propose that this question should be answered as part of the "*provisioning exercise*", when the size of the resource being requested prior to being provisioned is known. Others propose that predictions become important when monitoring or some form of interaction is required as part of a Quality of Service (QoS) agreement. Still others suggest that predictions are important to costings and design of priorities. Jennings *et al.* [16] performed an exhaustive study of 250+ papers on resource demand profiling and highlighted the different types of models and techniques that are used to profile such demand. They noted that the techniques used today model the relationship between the resources used by an application and that application's performance, thus highlighting gaps between the two.

2.1 Workload Modelling

Workload modelling provides a foundation for a variety of activities, including demand prediction. Kraft *et al.* [18] looked at particular storage metrics (such as IO) in the context of data centres, focusing on how workloads influence each other. This work was done in the context of consolidated environments but focused on the VM layers of the stack. Several techniques have been identified and borrowed from other fields. For example, Gmach *et al.* [10] used linear regression techniques to identify trends and use a bin packing algorithm to group VMs together. Lorido-Botran *et al.* [21] looked at how to scale workloads that require elasticising in cloud environments. They focussed on the implications at the IaaS level, but did use Time Series Analysis in the prediction algorithm. In autonomic computing, Jiang *et al* [17] used time series modelling techniques to dynamically adjust power, scheduled labour and auxiliaries at the IaaS level based on consumption, thus predicting power usage and adapting accordingly. They use an ensemble of techniques from Time Series Analysis such as Moving Average, Auto regression, neural networks, Support Vector Machines, etc., and apply this at hourly, daily and weekly granularities, with daily granularity giving favourable results.

Workload modelling can be used to inform short term adaptations or longer resource requirements. In terms of short term actions, several proposals have focused on QoS, using modelling techniques to identify problem workloads.

Carvalho *et al.* [6], Lorido-Botran and Tania [21] and Chaisiri *et al.* [7] all looked at approaches that identified VMs that *could be* a problem then taking action to adapt. Similarly, Sakr and Liu [22] looked at satisfying SLAs and applied rules to resources consumed by the workload. Gmach *et al.* [11] used modelling techniques at the provisioning stage as part of a capacity planning exercise with the aim of creating a just-in-time methodology by analysing the patterns and trends those VMs create and a weighting algorithm which moves the workload to a new set of VMs that are less contentious. Krompass *et al.* [19] and Schaffner *et al.* [23] also studied the impact of workloads causing outages by migrating them to a *less used* host (VM).

In terms of modelling at particular layers of the stack, as discussed previously, most work focusses on the VM. However, modelling at this layer (VMs) masks the true usage of a particular workload, as a natural smoothing of the data points occurs. Duggan *et al.* [8] took a novel approach of modelling the actual workload of a PaaS (Database) using Concurrent Query Performance Prediction (CQPP). They leveraged machine learning and linear regression techniques to predict the performance of a query. It is not clear if the regression techniques used were based on Time Series Analysis techniques.

Calheiros *et al* [5] used the ARIMA (Auto Regressive Integrated Moving Average) model to predict web requests at the VM layer, where a VM is a represented as a standard configuration of resources (CPU, memory and storage) and its expected performance is assumed. If the predicted load is likely to exhaust the expected performance, a new VM is created, thus solving the scaling problem. Tran *et al.* [27] focused on Time Series Analysis at an hourly level when analysing VM workloads. They used Seasonal ARIMA, but only at the VM level, and did not take into consideration seasons within seasons, which computational resources inevitably exhibit. Heuristics in existing work are generally based on greedy algorithms using simple rules. Carvalho *et al.* [6] proposed prediction based techniques for cloud environments by looking at admission control and assuming that capacity planning of resources has been done separately. They use a quota-based approach. The prediction is created based on heuristics and simple rules are applied to determine if a workload is behaving *predicted*, using techniques such as Exponential Smoothing (ETS) to exhaust the available resources.

In this paper we use time series analysis techniques to learn models of mainstream database workloads that include patterns such as trends and seasonality. We apply these techniques to different metrics that can be captured from databases. In the experiments, we evaluate the approach for short term predictions (of the order of days), but we also have some experience using them for longer term provisioning. In so doing, we provide the most comprehensive evaluation of time series based techniques for database workload modelling to date. By understanding the "*data*" and its structures well enough, it doesn't matter what layer of the technological stack or what part of the exercise; we should be able to make predictions based no time series data with complex characteristics.

2.2 Prediction in Clouds

Performing predictions of workloads in clouds is a relatively new idea, but essential from both sides of the relationship whether that is the CSP that provisions the resources or the user who consumes the resources. The argument here is one of elasticity and the *pay-as-you-go* nature of clouds. Jennings *et al.* [16] identify that VM placement is a problem but focus mainly on vector bin packing, such that the sum of the resources requested do not exceed the size of the bin in which they have been allocated. Jiang *et al.* [17] looked at finding a suitable balance between reducing the power cost and maintaining service levels by capacity planning VM workloads based on usage and then assigning a priority based on financial penalties to the provider.

There has been much work at the IaaS layer of clouds, namely VMs, but whether in terms of placement, provisioning, monitoring or where the database resides, work on the PaaS layer is scarce. Arguably the most resource consumptive layer of the stack is the database layer. Given that databases are multifaceted in the tasks they are asked to perform in feature rich DBMSs such as Oracle, IBM, Amazon and Microsoft, Calheiros *et al.* [5] did an in-depth analysis of the ARIMA technique, again focusing on the VM as a workload but for QoS and solving the problem of dynamic provisioning that CSPs currently experience. They predict the workload behaviour and feed this into a queuing model for calculating the VM provisioning. They run ARIMA models against Web VMs at an hourly granularity with data points taken over a month.

The goal of prediction in cloud environments can be to consolidate resources and this benefits both the consumer and the provider. There may be several reasons for this, such as cost, to meet SLA's or to save energy. For example, Dynamic Demand Response (D2R) is a project by the University of Southern California to utilise smart grid architectures specifically to address the supply and demand mismatch [24]. They utilise time series analysis, and specifically ARIMA, as an approach to predict power values based on time series data from a *kWh* metric. They apply this technique across their campus at the IaaS and PaaS layer. However, it is unclear if they apply it specifically to one or both. We make a case that the technique should be architecture independent such that

it should work for time series data regardless of architecture or metric.

There has been some work on prediction for cloud databases. For example, Sousa *et al.* [26] looked at the MySQL database cluster in an Amazon EC2 cloud and used ARIMA to predict the workload. However, the environment is a simple one unlike our environment: an Oracle Exadata database cluster running several TPC workloads. Our aim is to evaluate enterprise workloads that reflect our customers', and as our experiments will show we have evolved the time series technique to encompass ARIMA with Exogenous and Fourier terms to understand the complex structures within the time series data and make predictions.

3 PROBLEM DEFINITION

This section provides more detail on the problem to be solved. The problem can be described as follows.

> Given a time series m that provides monitoring information about a workload w, generate a prediction z for a period following on from that of w. The prediction should account for external influences on w.

The time series m is a trace or log of data in a time series format, as illustrated in Figures 2 and 3. The time series captures specific features of w, such as CPU, memory or logical IOs, but the techniques proposed are generic to any metric that has a time series format: $[x_1, ..., x_n]$. The time series is associated with the frequency of the monitoring, such as hourly, daily, weekly or monthly. The frequency of the prediction matches the frequency of m. The prediction z consists of the predicted values and associated error bars. The external influences on the prediction are events such as batch jobs and backups that routinely and sporadically occur in computational workloads, and which do not always follow a uniform pattern.

4 TIME SERIES ANALYSIS FOR DATABASE WORKLOADS

Time Series Analysis is a family of techniques that is widely used in applications that require forecasting, such as economics, sales, budgeting and market trading. Alysha *et al* [20] found that many time series exhibit complex seasonal patterns and applied a combination of Box-Cox transformation, Auto-Regressive Moving Average (ARMA) errors, Trend and Seasonal components (TBATS). Accounting for these complex data structures, Skorupa [1] suggests using a combination of varieties of the ARMA method to handle seasons within seasons to solve econometric problems—an interesting idea relative to our goal of making predictions from computational workloads that also show these traits. The main mode of operation for time series analysis is to understand the underlying forces that are reflected in the observed data, and to fit a model for forecasting. Both of these goals require that the pattern observed is identified. Once this takes place, we can then interpret and integrate the pattern with other data and then forecast if events will happen again. For example, does computational resource consumption over time have a trend? Is there is a pattern to these observations? Is it recurring?

Time Series are complex because each observation is dependant upon the previous observation that will likely be *influenced* by more than one observation. Applied to a computational problem where resources are assigned and workloads can be volatile, a workload can spike at particular times during a day and these spikes can repeat periodically or randomly. These influences are called *autocorrelation* — dependant relationships between successive observations. Time Series Analysis is broken down into two main areas with a view to uncovering patterns that can lead to the identification of a suitable model for predicting the future resource consumption of a workload:

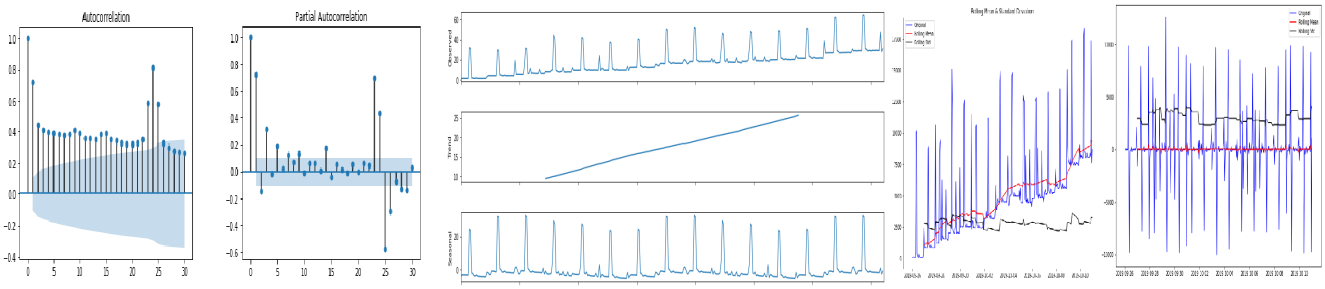
- *Time Domain* - ARIMA uses techniques such as Box-Jenkins and Dicky-Fuller to detect if the data is stationary, trending or requires an element of differencing.
- *Frequency Domain* - Techniques such as Fast Fourier Transform (FFT) to analyse data that is complex in a time domain.

This section reviews the main time series modelling and forecasting techniques, starting with techniques for stationary data and moving on to consider techniques for dealing with increasingly complex data. We also discuss how each approach can be used to capture features that are typical of database workloads.

4.1 Models

ARIMA is a class of models that capture the subtle structures of time series data. It was developed by Norbert Wiener *et al.* in the 1930's and 1940's [28]. Statisticians George Box and Gwilym Jenkins further developed these models for Business and Economic data in the 1970's, hence the name Box-Jenkins [2]. Computational resource utilisation is time series data and also exhibits structures such as trends, seasonality and/or complex multiple seasons, where a season within a season is exhibited. For example, variations may occur in the data at specific times, hourly, daily, weekly or monthly. These variations often repeat cyclically.

Such patterns are relevant to capacity planning for computational resources, where the goal is to answer the question *what resources are we likely to consume in the future?* We therefore investigate how to answer this question via ARIMA type models. Seasonal ARIMA (SARIMA) models also encapsulate seasonality, and are available in mathematical



(a) Correlograms (ACF/PACF) (b) SARIMAX Model Decomposition (c) Differencing Data (Trend)

Figure 1: Visualising Time Series Data

packages for R and Python. Here we cover multiple seasons, and have leveraged supervised Machine Learning to learn the past behaviours of a time series and forecast the future requirements for both long and short term predictions.

Time series models are typically characterized by their parameters. For ARIMA, the parameters are as follows:

- p is the number of autoregressive terms (AR) and can be found by using the Autocorrelation Function (ACF) or Partial Autocorrelation Function (PACF), as shown in Figure 1(a), and discussed in more detail in Section 4.3.
- d is the number of nonseasonal differences needed for stationarity, and
- q is the number of lagged forecast errors in the prediction equation.

The following additional parameters are required to handle the seasonal components:

- P is the order of the seasonal AR component (p),
- D is the seasonal differences needed for stationarity,
- Q is the seasonal order of the moving average terms (lagged forecast errors), and
- F is the frequency, for example 12 months, 24 hours.

Thus the SARIMA parameters are *(p,d,q,P,D,Q,F)*, which enables the model to handle both seasonal and non-seasonal workloads. We discover the seasonality of the data by decomposing it using library functions (in particular *statsmodels.tsa.seasonal* in python), as shown in Figure 1(b). When we decompose the data we visualise the traits as shown in Figure 1(a) and (b). For example, does the data have trend and seasonal patterns, and does it need to be differenced. If the data does have trend and seasonality then we can reduce the effects by differencing the data, as shown in Figure 1(c). This highlights the trend, and by differencing the data once we stablise it. Exogenous variables (Section 4.2) and Fourier Terms (Section 4.4) are leveraged to create more accurate forecasting models that can be executed on time series data at a server, database and transaction level.

From a simple regression model, as shown in formula (1), we see that $y(t) = \{y_t; t = 0, \pm1, \pm2, \ldots\}$ is a sequence of numbers (CPU, memory or IO), that is indexed by time t. We can see an observable signal sequence $x(t) = \{x_t\}$ and an unobservable noise sequence $\varepsilon(t) = \{\varepsilon_t\}$ that has independant variables that are random in nature.

$$y(t) = x(t)\beta + \varepsilon(t) \tag{1}$$

When the ARMA model is employed the (p,q) parameters are represented as:

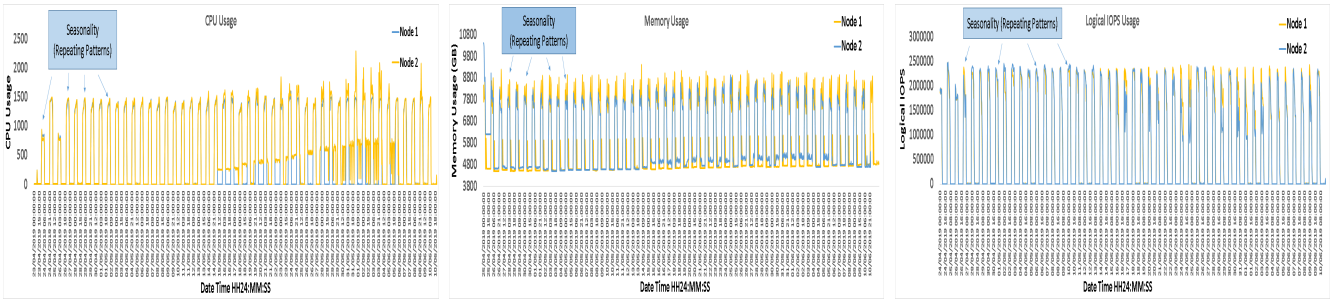
$$Y_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + a_t - \sum_{j=1}^{q} \theta_j a_{t-j} \tag{2}$$

which is often reduced to:

$$\phi_p(B)Y_t = \theta_q(B)a_t \tag{3}$$

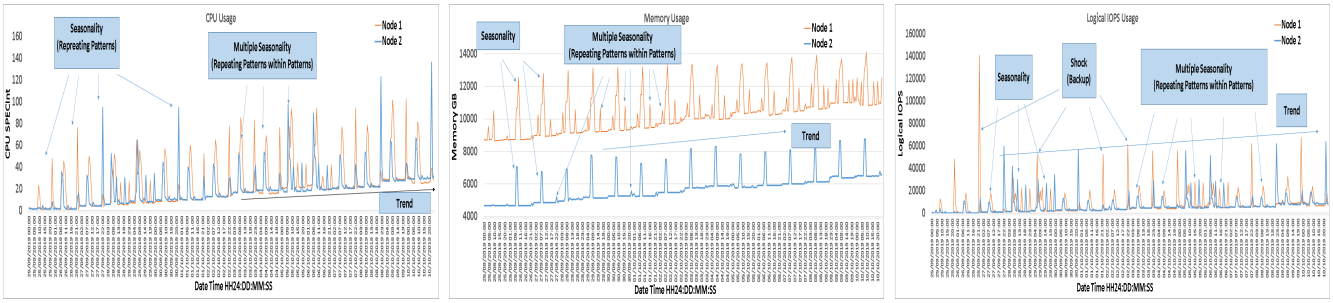
where $\phi_1, \ldots, \phi_p$ are the autoregressive parameters to be estimated, $\theta_1, \ldots, \theta_q$ are the moving average parameter to be estimated and $a_1, \ldots, a_t$ are a series of unknown random errors (or residuals) that are assumed to follow a normal distribution. This is commonly referred to as ARMA(p,q). B is the trend often found in time series data. However most computational time series data will probably introduce an element of stationariness. A stationary process is a stochastic process whose unconditional joint probability distribution does not change when shifted in time (i.e., in the future when performing a forecast). Consequently, parameters such as mean and variance also do not change in time.

When applied to our problem of capacity planning or short term monitoring, there is often an element of trend (incline or decline). Therefore, to make the data stationary we need to difference it, and for this we utilise Box-Jenkins to introduce parameter d:



(a) CPU Workload (b) Memory Workload (b) Logical IOPS Workload

Figure 2: Key Metrics: Workload Descriptions - Experiment One OLAP



(a) CPU Workload (b) Memory Workload (b) Logical IOPS Workload

Figure 3: Key Metrics: Workload Descriptions - Experiment Two OLTP

$$\phi_p(B)(1-B)^d Y_t = \theta_q(B)a_t \quad (4)$$

Thus (3) becomes (4) and creates the d that is the order of differencing. Replacing Y_t in the ARMA model with the differences creates the ARIMA(p,d,q) model. However, seasonality (patterns in the data) must be accounted for, to give:

$$\phi_p(B)\Phi(P)(B^s)(1-B)^d(1-B^s)^D Y_t = \theta_q(B)\Theta_Q(B^s)a_t \quad (5)$$

p, d and q are as defined earlier, and we introduce s as a known number of seasons (hourly, weekly, monthly, yearly). D is the order of differencing (this usually should not be greater than 2). P and Q are the autoregressive and moving averages when accounting for the seasonal shift. With analysis from the autocorrelation and partial autocorrelations (for which functions are available in python [4]) we can pre-populate the (p,d,q,P,D,Q,F) parameters that give a list of SARIMA models that are the most accurate for the data analysed.

4.2 Exogenous Variables

Exogenous variables are external parameters that convert the model ARIMA(p,d,q) to SARIMAX(p,d,q,P,D,Q,F) by including the linear effect that one or more external parameters has on the overall process; for example, a shock[1]. Computationally, examples could be a batch job, backup or fail-over that would seriously influence the computational resource consumption. Therefore Equation (2) extends to

$$Y_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{k=1}^{r} \beta_k X_{tk} + \varepsilon_t + \sum_{j=1}^{q} \theta_j a_{t-j} \quad (6)$$

Here we see the exogenous variable being held in r with time varying predictors t and coefficients denoted by β. It is possible to have multiple exogenous variables applied to a model, thus covering behaviours that many systems exhibit today. For example, a system that has a backup, batch jobs and that periodically fails over with trends and reoccurring patterns of usage could be covered by the SARIMAX model, as long as the exogenous variables (shocks) are understood and accounted for.

4.3 Exponential Smoothing Models

ARIMA type models work by transforming a time series to a stationary series, studying the nature of the stationary series

[1]SARIMAX is a variant of the SARIMA method that includes exogenous variables.

through autocorrelation (ACF) and partial autocorrelation (PACF) and then accounting for the auto-regressive or moving average if it is present. The key point with ARIMA is that the past observations are weighted equally. *Exponential Smoothing* is the other side of the coin, to accommodate missing values in the data or fixed drift. Fixed drift is where the forecast of tomorrow's values is today's values plus a drift term. Some cases can be confusing because computationally the data may be cyclical in behaviour (peaks and troughs) yet exhibit no trend. This is because the cycles are not of a fixed length, so before we observe the series we cannot be sure where the peaks and troughs of the cycles will be. In exponential smoothing, recent observations are given more weight than older observations, so forecasting is based on weighted averages of past observations. The weights decay exponentially as the observations get older. The exponential smoothing methods were proposed in the late 1950s by Hyndman *et al.* and Brown [3, 15]:

- Simple exponential smoothing (SES), suitable for data with no clear trend or seasonal pattern,
- Holt's linear trend (HLT) [13], extended HES to allow data with trend, and
- Holt-Winters seasonal method [30], extended to include seasonality.

Complex seasonal patterns, such as multiple seasonal periods, high frequency seasonality, non-integer seasonality and dual-calendar effects cannot be included in the above models. A new framework, incorporating Box-Cox transformations, Fourier representations with time varying coefficients and ARMA error correction, was introduced for data exhibiting complex patterns forecasting as suggested by Skorupa [25]. The new model is named TBATS, which stands for **T**rigonometric seasonality **B**ox-Cox **A**RMA **T**rend **S**easonal components. TBATS improved BATS by modeling seasonal effects using a Fourier series based trigonometric representation. This change allows non-integer length seasonal effects to be captured.

model :

$$y_t^{(\lambda)} = l_{t-1} + \Phi \cdot b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t \tag{7}$$

$$l_t = l_{t-1} + \Phi \cdot b_{t-1} + \alpha \cdot d_t \tag{8}$$

$$b_t = \Phi \cdot b_{t-1} + \beta \cdot d_t \tag{9}$$

$$d_t = \sum_{i=1}^{p} \varphi_i \cdot d_{t-i} + \sum_{i=1}^{q} \theta_i \cdot e_{t-i} + e_t \tag{10}$$

where,
T is the number of seasonalities
m_i is the length of the ith seasonal period
$y_t^{(\lambda)}$ is time series (Box-Cox transformed) at time t
$s_t^{(i)}$ is ith seasonal component
l_t is the level
b_t is the trend with damping effect
d_t is ARMA(p,q) process for residuals
e_t is Gaussian white noise
Φ - trend damping coefficient
α, β - smoothing coefficients
φ, θ are ARMA(p,q) coefficients

seasonal part :

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)} \tag{11}$$

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cdot cos(\lambda_i) + s_{j,t-1}^{*(i)} \cdot sin(\lambda_i) + \gamma_1^{(i)} \cdot d_t \tag{12}$$

$$s_{j,t}^{*(i)} = -s_{j,t-1}^{(i)} \cdot sin(\lambda_i) + s_{j,t-1}^{*(i)} \cdot cos(\lambda_i) + \gamma_2^{(i)} \cdot d_t \tag{13}$$

$$\lambda_i = \frac{2 \cdot \pi \cdot j}{m_i} \tag{14}$$

where,
k_i is the number of harmonics for the ith seasonal period
λ - Box-Cox transformation
$\gamma_1^{(i)}, \gamma_2^{(i)}$ - seasonal smoothing (two for each period)

The TBATS model has the following parameters: T, m_i, k_i, λ $\alpha, \beta, \phi, \varphi_i, \theta_i, \gamma_1^{(i)}, \gamma_2^{(i)}$. This leads to the question: how is the final model chosen? TBATS considers various alternatives and fits the models:

- with Box-Cox transformation and without it,
- with and without Trend,
- with and without Trend Damping,
- with and without ARMA(p,q) process used to model residuals,
- with a non-seasonal model, and
- with various amounts of harmonics used to model seasonal effects.

The final model will be chosen using the Akaike information criterion (AIC), which adds a penalty to the complexity of the model; the best model is the one having the lowest cost. The Python implementation was used in the experiments [25].

The reduced forms of TBATS are equivalent ARIMA models [20] generating the same forecast values. A TBATS model allows for dynamic seasonality whereas the ARIMA approach is that the seasonality is forced to be periodic as demonstrated by Hyndman [14].

4.4 Fourier Terms

SARIMAX deals with seasonality. However it has one major draw back, data with multiple seasonality. For example, data that exhibit trends and patterns that occur regularly such as every hour in a day or every day in a week or every week in a month. Such seasonal patterns are modeled through the introduction of Fourier terms, which are used as external regressors. In our analysis we look at computation data with hourly, daily, weekly or monthly seasons *(P1, P2, P3, Pm)*. Thus we have different Fourier series. Consider *Nt* as the ARIMA process:

$$yt = a + \sum_{i=1}^{M} \sum_{k=1}^{K_i} [\alpha sin(\frac{2\pi kt}{P_i}) + \beta cos(\frac{2\pi kt}{P_i})] + N_t \qquad (15)$$

In the example shown in equation (5) we look at two seasons, *P1* running over a 24 hours period and *P2* running over a weekly period. Thus for each of the periods, P_i, the number of Fourier terms (k_i) are chosen to find the best SARIMAX parameters, and then ordered appropriately and selected based on which gives the best root mean squared error (RMSE). We determine the granularity of data required from the Makridakis Competition results [29]; for example, for an effective hourly forecast 700 hourly data points (circa 29 days) are required. In our solution we apply Fourier analysis if we detect time series data with multiple seasonality.

5 APPROACH

This section describes how the time series analysis techniques from Section 4 are applied to support capacity planning for DBaaS. This section also details how we propose to use machine learning to automate the forecasts, and algorithmically how we are able to discover the models, removing the need for the user to have an intrinsic understanding of the complexities of time series analysis, which is a field in its own right.

5.1 Overview

Our approach was to execute is to capture key metrics (CPU, IOPS and Memory) that are applicable to monitoring and capacity planning via an agent. The Agent specifically executes commands on the hosts that retrieve the metric values from the database and polls these metrics at regular intervals. The values from the metrics are then stored, centrally, in a repository where they are aggregated into hourly values. This cycle continues for a period of 30 days in the experiments. The Algorithms shown in Figure 4 are then executed. The first stage of the algorithm gathers the data and checks for any missing values. It is possible that the agent may have been at fault and may not have executed or polled the value from the database target; this can happen in live environments due to maintenance cycles or faults. If this is the case, a linear interpolation exercise is carried out to fill in the gaps based on known data points. We then begin the Machine Learning task and split the data into a training set and a testing set. Depending on whether the user chooses Holt-Winters Exponential Smoothing (HES) as discussed in section 4.3 or SARIMAX, a different branch of the algorithm will be followed. If SARIMAX is selected the algorithm then analyses the time series data for the metric being predicted and computes the ACF/PACF to determine which models are probably a good fit for the data selected. As the flow continues, the data characteristics are understood, such as stationarity, seasonality, multiple seasonality and shocks, where each model is then computed to obtain an RMSE. The model with the best RMSE is the most accurate. That model is then stored in a central repository and used for a period of one week or until the model's RMSE drops to a point where it is rendered useless. The same approach is executed for the HES algorithm in that it is kept for one week and the RMSE is continually monitored.

5.2 Machine Learning Algorithm

Time Series Analysis lends itself to a supervised learning approach for several reasons. We take historical data with known relationships between the observations captured over historical time points. The output is a numerical value (prediction) of what we think the future resource consumption of a metric is, therefore it is a regression problem. There is no need to continuously compute the ACF/PACF (Lags), add exogenous or find the Fourier terms. We simply re-train on the data unless the number of observations increases significantly or the time since the last use of the models lengthens beyond a certain period. We have determined that a model becomes stale over time and thus will need to be recomputed after a week, as shown in Figure 4. Depending on the type of time series technique used (HES or SARIMAX), the future prediction length also has a bearing on how the data is treated in the ML algorithm, as shown in Table 1, which is based on Makridakis [29].

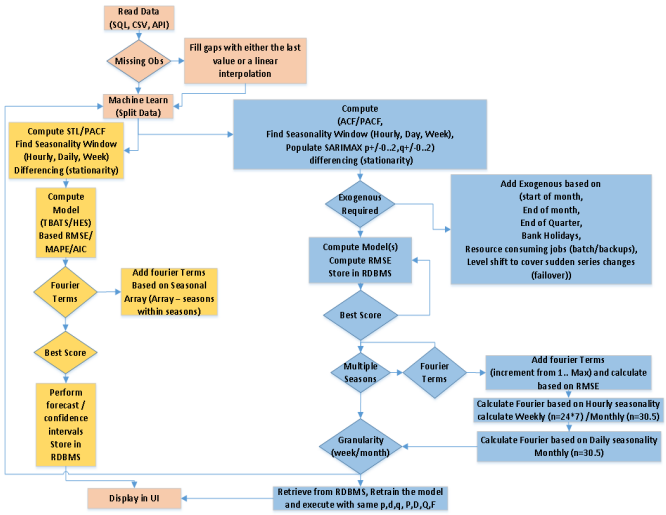


Figure 4: Algorithms: ESM/SARIMAX with FFT and Exogenous Workflow.

Table 1: Machine Learning Breakdown and Observations

Forecast	Obs	Train Set	Test Set	Prediction
SARIMAX Hourly	1008	984	24	24 (Hours)
SARIMAX Daily	90	83	7	7 (days)
SARIMAX Weekly	92	88	4	4 (Weeks)
HES [1] Hourly	1008	984	24	24 (Hours)
HES [1] Daily	90	83	7	7 (days)
HES [1] Weekly	92	88	4	4 (Weeks)

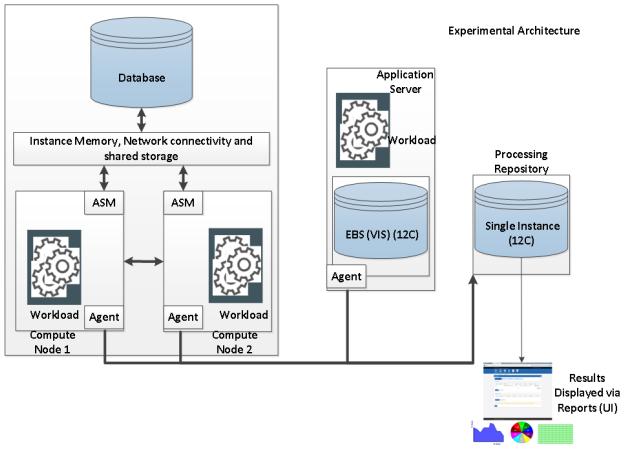


Figure 5: Experimental Architecture: Typical customer N-tier Architecture.

6 EXPERIMENTAL SETUP

6.1 Experimental Environment

The environment we created is based on an N-tier architecture, in that workloads are executed on an Oracle clustered database and through an application server that serves a web-based application; for example, transactions resulting from a sequence of *clicks* on a webpage. The load is shared between the nodes of the clustered database to keep an even balance of activity, as shown in Figure 5.

6.2 Experimental Workloads

Several Database workloads where used to run prediction models against as identified in our previous work [12]. To truly evaluate our full algorithm and the models, the workloads needed to exhibit several key characteristics, such as seasonality (repeating patterns). We produced two experiments running against a clustered database. Workload OLAP is a simple one that reflected seasonality, which can be depicted graphically via a chart as spikes in a workload that are surges in usage, for example users logging on at peak times. This is clearly identified for the key metrics in Figure 2. Trends (non-stationarity) that show the load growing—for example the data set becomes bigger and thus code execution times lengthen—or its resource utilisation increases and decreases again, are also visible in Figure 2. Workload two is a more complicated workload (OLTP), in that it exhibits a trend uniformly across all three metrics, as is shown in

[1]Table Footnote - Holt-Winters Exponential Smoothing

Figure 3. Users log on in surges at peak times to reflect typical usage of OLTP type systems, and this produced multiple seasonality. We also introduce shocks in the form of a backup which produces the large spike in logical IOPS, shown in Figure 3(c). The second experiment is to try and produce a prediction line that also grows in line with the trend, taking into consideration growth, multiple seasonality and shocks.

6.3 Experimental Models

Once the workloads had been executed, the relevant metrics captured and stored in a central repository, we ran the modelling routines. This consisted of exhaustively learning all the permutations of the models across the three techniques. As shown in the correlogram diagram examples in Figure 1(a), we measure the data over 30 lags, so each lag has a maximum of 22 models. For example, for Lag 1, the model combination is (1,0,0)(0,0,1,24) ,..., (1,1,2)(1,1,1,24), making the total number of models evaluated at over 6000 across the two experiments for one clustered database residing on two nodes.

The three techniques and the number of models are:

- ARIMA p,d,q = 180 models per instance (totalling 360 models)
- SARIMAX p,d,q,P,D,Q,F = 660 models per instance (totalling 1320 models)
- SARIMAX p,d,q,P,D,Q,F + Exogenous (4)+ Fourier Terms (2) = 666 models per instance (totalling 1332 models)

We measure the accuracy of every model against the RMSE and then choose the top model from each of the three methods. When we have the top model combination for each of the three methods, we then compare the accuracy of the results, as detailed in Table 2. There are several shocks in the form of backups that run every 6 hours (4 exogenous variables), and the FFT is made up of sine and cosine waves that are then added to the model with the best RMSE to see if it can be further improved in accuracy, again determined by the RMSE. This creates a huge number of models. In practice, we could reduce the number of models by tuning. We do this by looking at the correlogram shown in Figure 1(a), and looking at where the data points intersect with the shaded areas, as this gives an indication of a model that is likely to be suitable, thereby reducing the thousands of potential models considerably. For the purpose of empirical evaluation we compared over 6000 thousand models across two experiments, based on the RMSE.

7 EXPERIMENTS AND ANALYSIS

The experiments have been designed to investigate the extent to which the approach from Section 6 can address the following challenges in workoads:

C1 : Reoccurring patterns (seasonality).
C2 : Trends and Stationariness.
C3 : Multiple patterns (overlapping seasonality).
C4 : Shocks.

Our approach to this question is by way of empirical evaluation. Using increasingly complex deployments with representative workloads, we establish the extent to which we can predict the future load based on past load. This section describes the workloads and the platforms used in the experiments.

7.1 Experiment One - Simple OLAP Workload

Overview. Experiment one investigates Challenges *C1* and *C4*, it uses an OLAP workload with a modest number of 40 OLAP users connecting across the cluster. In this scenario, users connect to a clustered database and perform OLAP activities that are high in IO and execute for long periods of time; the tasks carried out are similar to those in TPC-H [9]. IO is generated via SQL activity and data manipulation language (DML) is executed via updates, inserts and deletes. This workload shows repeating patterns (seasonality), some growth (trend) and it did not exhibit multiple seasonality. A shock is introduced via a Backup that is run to perform housekeeping routines such as keeping the database logs (archivelogs) from filling up the disc drives in the operating system (OS). This is a routine that many database systems employ as some form of disaster recovery architecture, thus is a representative configuration. This produced a workload represented graphically in Figure 2. The dataset grew by several GB per hour.

Results. Examples of the predictions from Experiment one are shown in Figure 6, for CPU, for ARIMA, SARIMAX and SARIMAX with Exogenus Variables and Fourier Terms. The shaded area (blue) shows the data used by the algorithm for learning the behaviour of the workload. The recurring pattern (Challenge C1) is shown in each of the figures. The yellow section of the charts shows the prediction, which displays the continued pattern the algorithm has learnt. The results show that the peaks and troughs have been captured successfully by all three approaches to time series modelling. A similar behaviour is shown for the other metrics, namely Logical IO and Memory for each of the instances, and is not shown here for lack of space. Logical IO is a better metric for displaying complex data structures such as seasonality, and this is shown in Experiment 2, however, for the purpose of variety, forecasting a different metric such as CPU in Experiment 1 was chosen.

A deeper dive into the accuracy of the models is shown in Table 2(a). We tested the accuracy using three methods, which are Root Means Squared Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Percentage

Accuracy (MAPA). The results show that SARIMAX with Exogenous variables and FFT is consistently more accurate and reduces the error across the three metrics of CPU, Logical IO and Memory, as shown in Table 2(a). The results also show that there is a significant jump in accuracy when the seasonal component of the data is taken into consideration when modelling Logical IOPS's. We only focus on the RMSE which shows a considerably lower number between the three models. With an error of 52879.49 logical IOPS (for instance, cdbm012 in Table 2a), the RMSE is based on values of of 2.3 million logical IOPS per hour throughput at the workloads peak. Thus the error rate is acceptable.

Given the nature of OLAP workloads working in an N-Tier architecture, the results of Experiment One were promising given the added complexity of the structures the data produced. ARIMA models do well in predicting the pattern, but it is expected to do well given that the workload exhibits simple structures and patterns. The SARIMA and SARIMAX models took into consideration any traits, seasonality and trend and improved the scores a little more. This is the reason why the RMSEs are lower in SARIMA than ARIMA. There was a backup task (cbdm011) that was executed from Node 1 at midnight every night, which also contributed to IO, CPU and Memory, thus creating an *exaggerated pattern*. Overall the SARIMAX with FFT and Exogenous variables was the most accurate model. This raises the question as to whether SARIMAX with FFT and Exogenous would give similar results on workloads that are much more complicated, such as OLTP?

7.2 Experiment Two - Complicated OLTP Workload - Seasonality and Growth

Overview. Experiment Two introduces another layer of complexity with the introduction of trends and multiple seasonality via an OLTP workload, thus presenting challenges *C1, C2, C3* and *C4* in a single scenario. In this experiment a workload was executed with users connecting to a clustered database, mirroring an OLTP type system, and we allow the user base to grow per day. The workload is similar to TPC-E [9]. IO is generated via SQL, including updates, inserts and deletes. Memory is consumed as the user connects and executes the SQL, as is CPU via the normal database internal memory and optimiser management systems. This activity runs for a period of 30 days, and metrics are captured every 15 mins via an agent and stored in a central repository. Aggregation then takes place over the hour between the four captured metrics. Multiple seasonality is introduced because the number of users is growing every day and there is a pattern of growth. Trend is introduced by increasing the user base by 50 users per day across the database cluster, thus growing the dataset and the consuming more resources. Surges in users are introduced twice daily at 07:00am of 1000 users for a period of 4 hours and again at 9am for another 1000 users for a period of 1 hour. These introduce several factors of multiple seasonality and a shock. A further shock is introduced by means of a Recovery Manager backup that is employed as a database housekeeping routine that prevents the database redo logs from filling up the disc drives on the OS. This is standard procedure for any RDBMS architecture that employs disaster recovery and thus is a relevant configuration. This produced the workload depicted graphically in Figure 3 a, b and c.

Results. Results of the prediction are shown graphically in Figure 7. The results show that the prediction line grows with the trend line and it captures the seasonality, including multiple seaonsonality. In these charts we show that SARIMAX with Exogenous variables and Fourier terms across three metircs of CPU, memory and Logical IOPS for one database instance provides a prediction line that reflects the past behaviours. In the Logical IOPS, Figure 3(c), the model takes into consideration the introduction of a shock (Backup). This is encouraging as the shock is a backup which significantly increases in IO, therefore the model is able to handle aspects that reflect the true nature of computational workloads and still be accurate. The repeating patterns shown in each of the figures are also reflected in the prediction lines. We have not added all the charts (CPU and Memory) due to there not being enough space in the paper.

A deeper dive into the accuracy of the models is provided in Table 2(b). We tested the accuracy using three methods, which are Root Means Squared Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Percentage Accuracy (MAPA). The results show that SARIMAX with Exogenous variables and FFT is consistently more accurate, and reduces the error across the three metrics of CPU, Logical IO and Memory. The results also show that there is a significant jump in accuracy when the seasonal component of the data is taken into consideration when modelling Logical IOPS, but it also maintains the accuracy when we add in complex data structures such as multiple seasonality and shocks. Figure 7(c) clearly shows the multiple seasonality being predicted in the orange line as two large spikes (7:00am - 10am), reflecting a surge of 2000 users, and several smaller spikes (shocks) occurring at 00:00 in the form of a backup task. The trend (incline) is achieved by increasing the user base by 50 users per day. You can see the orange prediction line taking this complex workload structure into consideration in its prediction.

The results of Experiment Two were very promising, given the added complexity of the data introduced through multiple seasons, significant trend and shocks. Therefore, the models

(a) Experiment Results - OLAP

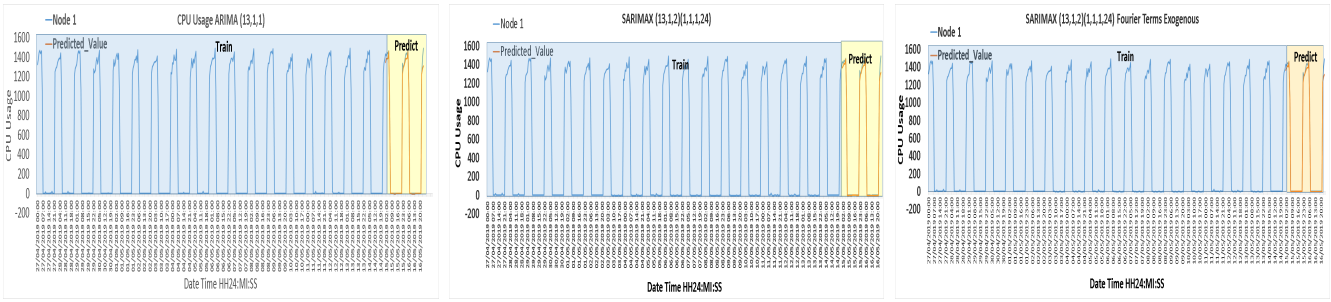
Forecast & Model	Metric	Root Mean Squared Error (RMSE)	MAPE %	Instance
ARIMA (13,1,1)	CPU	8.93	96.1	cdbm011
SARIMAX (13,1,2)(1,1,1,24)	CPU	8.4198	97	cdbm011
SARIMAX FFT Exogenous (13,1,2)(1,1,1,24)	CPU	8.4195	**97.06**	cdbm011
ARIMA (4,1,1)	CPU	44.78	97.28	cdbm012
SARIMAX (4,1,2)(1,1,1,24)	CPU	47.95	97.1	cdbm012
SARIMAX FFT Exogenous (4,1,2)(1,1,1,24)	CPU	38.89	**97.64**	cdbm012
ARIMA (13,1,2)	Memory	135.79	99.02	cdbm011
SARIMAX (1,1,1)	Memory	183.22	98.93	cdbm011
SARIMAX FFT Exogenous (1,1,2)(1,1,1,24)	Memory	130.42	**99.14**	cdbm011
ARIMA (13,1,2)	Memory	90.10	99.86	cdbm012
SARIMAX (1,1,1)(1,1,1,24)	Memory	61.30	98.91	cdbm012
SARIMAX FFT Exogenous (1,1,2)(1,1,1,24)	Memory	53.53	**99.92**	cdbm012
ARIMA (15,1,2)	Logical IOPS	39695	4533.01 -%	cdbm011
SARIMAX (4,1,1)(1,1,1,24)	Logical IOPS	15833.36	950.92 -%	cdbm011
SARIMAX FFT Exogenous (4,1,2)(1,1,1,24)	Logical IOPS	15112.06	**734.62 -%**	cdbm011
ARIMA (15,1,2)	Logical IOPS	151278.45	1375 -%	cdbm012
SARIMAX (4,1,1)(1,1,1,24)	Logical IOPS	52965.44	213.67 -%	cdbm012
SARIMAX FFT Exogenous (4,1,2)(1,1,1,24)	Logical IOPS	52879.49	**210.71 -%**	cdbm012

(b) Experiment Results - OLTP

Forecast & Model	Metric	Root Mean Squared Error (RMSE)	MAPE %	Instance
ARIMA (25,1,1)	CPU	10.89	88.8	cdbm011
SARIMAX (27,1,2)(1,1,1,24)	CPU	8.94	9.16	cdbm011
SARIMAX FFT Exogenous (27,1,2)(1,1,1,24)	CPU	6.02	**90.25**	cdbm011
ARIMA (22,1,1)	CPU	22.54	56.31	cdbm012
SARIMAX (2,1,1)(1,1,1,24)	CPU	12.22	86.08	cdbm012
SARIMAX FFT Exogenous (2,1,1)(1,1,1,24)	CPU	7.38	**86.54**	cdbm012
ARIMA (26,1,2)	Memory	551.07	97.08	cdbm011
SARIMAX (6,1,1)(1,1,1,24)	Memory	341.56	**98.02**	cdbm011
SARIMAX FFT Exogenous (6,1,1)(1,1,1,24)	Memory	359.44	97.66	cdbm011
ARIMA (21,1,1)	Memory	47.62	98.42	cdbm012
SARIMAX (27,1,2)(1,1,1,24)	Memory	0.12	**99.47**	cdbm012
SARIMAX FFT Exogenous (27,1,2)(1,1,1,24)	Memory	94.77	98.81	cdbm012
ARIMA (21,1,1)	Logical IOPS	8192.55	78.91	cdbm011
SARIMAX (14,1,2)(1,1,1,24)	Logical IOPS	6628.97	80.14	cdbm011
SARIMAX FFT Exogenous (14,1,2)(1,1,1,24)	Logical IOPS	4579.14	**82.1**	cdbm011
ARIMA (15,1,2)	Logical IOPS	8218.81	85.83	cdbm012
SARIMAX (1,1,1)(0,1,1,24)	Logical IOPS	1964.54	86.18	cdbm012
SARIMAX FFT Exogenous (4,1,2)(1,1,1,24)	Logical IOPS	2373.86	**87.33**	cdbm012

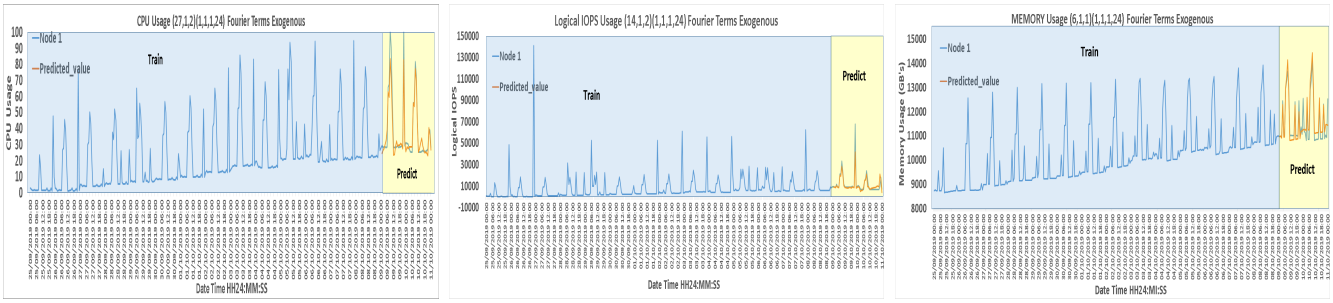
Table 2: Table of Results

were able to predict a trend and reflect all the traits of a complex computational model accurately.



(a) Exp 1 - ARIMA (b) Exp 1 - SARIMAX (c) Exp 1 - SARIMAX With Exog and Fourier Terms

Figure 6: Experiment 1: Prediction charts Comparing Three ARIMA Techniques



(a) Exp 2 - CPU (b) Exp 2 - Logical IOPS (c) Exp 2 - Memory

Figure 7: Experiment 2: Prediction Charts Using SARIMAX with Exogenous and Fourier Terms

8 DATABASE CAPACITY PLANNING IN PRACTICE

Oracle Advanced Customer Services has incorporated the work from this paper into its monitoring and assessment services that chart time series data. Figure 8 shows an early design of the UI and how the time series data is displayed across a clustered database instances. The user can select between SARIMAX or HES, as we have shown that these two models cover most nuances shown in computational workloads we studied.

The model has been found to be applicable in several use cases:

- *Short term monitoring:* within the next few days, what will resource usage look like across my technology estate?
- *Medium to Long Term Capacity Planning:* do I need to find extra capacity for my estate?
- *Migration:* If I need to migrate to a new platform, such as a Cloud architecture, what resource capacity do I need in the next 6 months to a year?

The approach is being applied across several thousand customers, covering 1000's of workloads involving different components in the technological stack. It has been applied to the following scenarios, as we can capture the metric usage via agents and store results in a central repository for time series analysis:

- Groups of *clicks* that make up a transaction in a web page.
- In conjunction with OATS, the Oracle Applications Testing Suite, we can predict if a transaction is beginning to slow down to aid pro-active monitoring of the application layer.
- Application containers such as weblogic can also be monitored as they are also a source of time series data.
- Network layers of storage, such as Network Attached Storage and SAN Volume Controllers, that are critical to the database instance are also monitored to display if the database is likely to suffer performance bottlenecks.
- Capacity Planning and mapping of different architectures to aid migrations to cloud technologies from on-premise.

We also intend to investigate the idea of applying this time series analysis approach with other learning techniques to act or make changes dynamically.

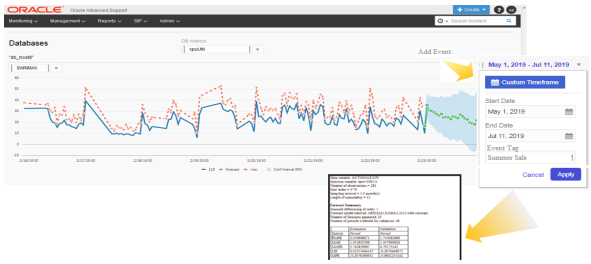
9 CONCLUSIONS

In conclusion, the use of machine learning has greatly increased the ability of these models to be utilised on computational workloads that exhibit diverse patterns over time. Manually creating a forecast is cumbersome, requires a degree of expertise in how the models work and understanding of the data to which the model is being applied. Forecasting is susceptible to mistakes, rendering the forecast inaccurate or wildly off when compared to actual findings because of these risks. Computational workloads are sometimes unsuitable for forecasting as the time series data is often unstable in that it has many variables (external and internal) that influence its behaviours; for example, system backups, batch jobs that aggregate data, or reports that consume vast amounts of resource infrequently or periodically. The data is also captured at different levels of granularity (polled every minute, hourly or daily) which also displays nuances or complex structures that often confuse forecasting models. When all of these challenges are aggregated, paired with the need for expertise and understanding from the person performing the forecast, manual forecasting of system workloads becomes impractical.

By automating the process and applying Machine Learning to continually assess the models performance, we reduce many of the costs and barriers. We reduce the risks of applying the wrong model as we continually assess the performance through Machine Learning to account for new behaviours the data (system) may adopt because the learning engine moves forward as the system moves forward, organically working together. We also don't *relearn* unless the model becomes unsuitable or the system (data) has changed significantly (shocks or new behaviours). Only at that point do we adjust to these new behaviours that all systems experience as new functionality is introduced throughout the systems life-cycle.

We also have performance tuned the algorithm by reducing the number of models we evaluate by automating the ACF and PACF functions. When we ran the experiments on a workload we evaluated several thousand models running on a two node database. If the clustered database resided on four nodes then the number of models needed to be evaluated across two experiments would be nearly 24000 and this is unmanageable, especially when faced with the challenge of providing analysis against thousands of workloads from thousands of customers routinely. Gains are also achieved by parallel processing the models. We only use HES or SARIMAX with Exogenous and Fourier terms.

Using established techniques such as SARIMAX with Exogenous variables to account for the nuances exhibited in computational workloads, we have improved the user experience of resource charts. Historically a chart would advise



(a) Single Instance With Exogenous Selection

Figure 8: Proposed User Interfaces: Model Selections and Predictions. ©Oracle Corporation

on past or present usage of a workload, in terms of metric based consumption such as CPU, IO, Memory or network bandwidth, and the user then had to interpret what that chart indicated about the future behaviour. By using this approach, we can now advise users on what we *think* may happen to their workloads. This gives a way to mitigate the responsibilities system administrators face when monitoring critical and important systems, rather than the "old" threshold-based monitoring approach, that often led to a reactive way of working when system outages took place. For example, consider a performance problem that begins weeks earlier but suddenly hits a threshold, becoming non-compliant relative to the SLA. The approach proposed in this paper could advise through a prediction that there is *likely* to be an issue soon, and therefore recommend refocussing some resources to avoid an outage. Providing this early warning capability to system administrators can only be a good thing.

Where this approach has its challenges is when a system is unstable or in a period of fault, for example frequent crashes as the Learning Engine then relearns to adopt new behaviours. In our algorithm we account for this by suggesting that the event needs to happen more then *3* times for it to be a behaviour, which can be changed manually. It is perfectly plausible that the system fails over to a new site to test disaster recovery. Therefore if a system crashes we discard it, however if the system continually crashes the learning engine will see it as a behaviour and account for it in its forecast. Live systems rarely continually crash but they do crash, therefore manual override is needed to accommodate systems that are *in-fault* as we suggest that forecasting will not be a true reflection of the system when stable.

There is still a need for threshold based monitoring. Utilising these techniques to predict when a threshold is likely to be breached is an advisable way to implement this approach for proactive monitoring and capacity based questions; we don't see our approach as a complete replacement for thresholds just yet.

REFERENCES

[1] 2019. *Forecasting Time Series with Multiple Seasonalities using TBATS in Python.* https://medium.com/intive-developers/forecasting-time-series-with-multiple-seasonalities-using-tbats-in-python-398a00ac0e8a

[2] George E. P. Box. 2008. *Time series analysis forecasting and control* (fourth edition. ed.). Hoboken, N.J.

[3] RG Brown. 1959. *Statistical forecasting for inventory control.* http://documents.irevues.inist.fr/handle/2042/28540.

[4] Jason Brownlee. 2014. Machine learning mastery. *URL: http://machinelearningmastery. com/discover-feature-engineering-howtoengineer-features-and-how-to-getgood-at-it* (2014).

[5] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya. 2015. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications in QoS. *IEEE Transactions on Cloud Computing* 3, 4 (Oct 2015), 449–458. https://doi.org/10.1109/TCC.2014.2350475

[6] M. Carvalho, D. MenascÃĺ, and F. Brasileiro. 2015. Prediction-Based Admission Control for IaaS Clouds with Multiple Service Classes. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. 82–90. https://doi.org/10.1109/CloudCom.2015.16

[7] S. Chaisiri, B. Lee, and D. Niyato. 2010. Robust cloud resource provisioning for cloud computing environments. In *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. 1–8. https://doi.org/10.1109/SOCA.2010.5707147

[8] Jennie Duggan, Olga Papaemmanouil, Ugur Cetintemel, and Eli Upfal. 2014. Contender: A Resource Modeling Approach for Concurrent Query Performance Prediction.. In *EDBT*. 109–120.

[9] Dominic Giles. 2019. *SwingBench 2.2 Reference and User Guide.*

[10] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. 2007. Capacity Management and Demand Prediction for Next Generation Data Centers. In *IEEE International Conference on Web Services (ICWS 2007)*. 43–50. https://doi.org/10.1109/ICWS.2007.62

[11] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. 2007. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. In *2007 IEEE 10th International Symposium on Workload Characterization*. 171–180. https://doi.org/10.1109/IISWC.2007.4362193

[12] Antony Higginson, Norman W. Paton, Suzanne M. Embury, and Clive Bostock. 2017. DBaaS Cloud Capacity Planning - Accounting for Dynamic RDBMS System that Employ Clustering and Standby Architectures. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT*. 687–698. https://doi.org/10.5441/002/edbt.2017.89

[13] CE Holt. 1957. *Forecasting seasonals and trends by exponentially weighted averages.*

[14] Rob. J. Hyndman. 2014. *TBATS with regressors.* https://robjhyndman.com/hyndsight/tbats-with-regressors.

[15] Rob J. Hyndman, George Athanasopoulos, and OTexts.com. 2014 2014. *Forecasting : principles and practice / Rob J Hyndman and George Athanasopoulos* (print edition. ed.). 291 pages ; pages.

[16] Brendan Jennings and Rolf Stadler. 2015. Resource Management in Clouds: Survey and Research Challenges. *Journal of Network and Systems Management* 23, 3 (01 Jul 2015), 567–619. https://doi.org/10.1007/s10922-014-9307-7

[17] Y. Jiang, C. Perng, T. Li, and R. Chang. 2012. Self-Adaptive Cloud Capacity Planning. In *2012 IEEE Ninth International Conference on Services Computing*. 73–80. https://doi.org/10.1109/SCC.2012.8

[18] Stephan Kraft, Giuliano Casale, Diwakar Krishnamurthy, Des Greer, and Peter Kilpatrick. 2013. Performance models of storage contention in cloud environments. *Software & Systems Modeling* 12, 4 (01 Oct 2013), 681–704. https://doi.org/10.1007/s10270-012-0227-2

[19] Stefan Krompass, Harumi Kuno, Umeshwar Dayal, and Alfons Kemper. 2007. Dynamic Workload Management for Very Large Data Warehouses: Juggling Feathers and Bowling Balls. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '07)*. VLDB Endowment, 1105–1115. http://dl.acm.org/citation.cfm?id=1325851.1325976

[20] Alysha M. De Livera, Rob J. Hyndman, and Ralph D. Snyder. 2011. Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing. *J. Amer. Statist. Assoc.* 106, 496 (2011), 1513–1527.

[21] Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A. Lozano. 2014. A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. *Journal of Grid Computing* 12, 4 (01 Dec 2014), 559–592. https://doi.org/10.1007/s10723-014-9314-7

[22] S. Sakr and A. Liu. 2012. SLA-Based and Consumer-centric Dynamic Provisioning for Cloud Databases. In *2012 IEEE Fifth International Conference on Cloud Computing*. 360–367. https://doi.org/10.1109/CLOUD.2012.11

[23] J. Schaffner, B. Eckart, D. Jacobs, C. Schwarz, H. Plattner, and A. Zeier. 2011. Predicting in-memory database performance for automating cluster management tasks. In *2011 IEEE 27th International Conference on Data Engineering*. 1264–1275. https://doi.org/10.1109/ICDE.2011.5767936

[24] Yogesh Simmhan, Saima Aman, Alok Kumbhare, Rongyang Liu, Sam Stevens, Qunzhi Zhou, and Viktor Prasanna. 2013-07. Cloud-Based Software Platform for Big Data Analytics in Smart Grids. *Computing in Science Engineering* 15, 4 (2013-07), 38,47.

[25] Grzegorz Skorupa. 2019. *TBATS implementation in Python.* https://github.com/intive-DataScience/tbats.

[26] Flavio R. C. Sousa, Leonardo O. Moreira, JosÃĺ S. Costa Filho, and Javam C. Machado. 2018. Predictive elastic replication for multi-tenant databases in the cloud. *Concurrency and Computation: Practice and Experience* 30, 16 (2018), e4437. e4437 cpe.4437.

[27] V. G. Tran, V. Debusschere, and S. Bacha. 2012. Hourly server workload forecasting up to 168 hours ahead using Seasonal ARIMA model. In *2012 IEEE International Conference on Industrial Technology*. 1127–1131. https://doi.org/10.1109/ICIT.2012.6210091

[28] Norbert Wiener. 1950. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications.* http://hdl.handle.net/2027/uc1.b4062686

[29] Wikipedia contributors. 2019. Makridakis Competitions — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Makridakis_Competitions&oldid=903376442 [Online; accessed 1-August-2019].

[30] PR Winters. 1960. Forecasting sales by exponentially weighted moving averages. In *Management Science*, Management Science (Ed.).

5.3 Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure

Antony Higginson, Clive Bostock, Norman Paton, Suzanne Embury

In Proceedings of the 25th International Conference on Extending Database Technology (EDBT),
Published by OpenProceedings.org,
March 29-1st April 2022,
Edinburgh, UK,
Pages 687 - 699,
ISBN: 978-3-89318-086-8,
https://10.5441/002/edbt.2017.01,

Summary: This final paper creates and evaluates vector bin-packing solutions following the First-Fit Decreasing (FFD) method. In this paper we execute the bin-packing algorithms against real world target cloud configurations from real world N-Tier architecture and real world use cases taken from Oracle ACS customers. This novel idea resulted in a Patent being filed in the US and EU.

Approaches: Given our understanding from papers one and two, our approach to bin-packing was several fold. Firstly, we chose the simplest and easiest technique, being First-Fit Decreasing (FFD) rather than more complex bin-packing algorithms, such as Best-Fit or Next-Fit. We had identified that existing vector bin-packing routines do not take into consideration clustered or pluggable workloads thus the question to be answered was: can the FFD technique, place workloads from complex database architectures, being clustered and pluggable workloads after our extension? Secondly, we must place in a current available cloud specification, which we chose to be Oracle OCI of a bare metal configuration and its specification. This involved creating a templated approach that can handle increasing the vector to cover metrics other IO, CPU and Memory. Finally, could we identify any wastage when the workloads were placed that could assist in elasticising the target cloud architecture to achieve savings in terms of cost (Resources or Monetary) that could be reassigned elsewhere to the cloud consumer.

Comments on authorship: I proposed the main idea of the paper, developed and validated the main approach, conducted an empirical evaluation, provided and analysed results, investigated related work, all graphics, participated in the entire writing processes and addressed any comments. I created the environments, wrote any code in the form of python bin-packing models. I configured any operating systems and databases as required and acted as the DBA to monitor the environments as the experiments ran. Specific python code was written to cover with the help of Oracle Software Engineer (Clive Bostock) to ensure we followed Oracle SDLC. My supervisors Norman Paton and Suzanne Embury also contributed to the idea, experiment design, literature research and analysis. Norman helped with the Mathematics and specific notation required to extend the FFD algorithm. They also proof read the paper and approved the final submission. They also guided the whole research process.

Key Contribution: From Section 1.9.3 and 6.1.

Reflection: One reflection of this particular piece of work that requires specific mention, was the awareness of other bin-packing techniques compared with the simple one we evaluated (FFD). Part of this research identified that *no* bin-packing routine placed workloads that were siblings of each other. In traditional vector bin-packing routines, each vector is treated separately or in isolation, for example a VM is a vector of resource. When in fact, under certain constraints, such as a clustered or pluggable deployments, they should, as we show in the paper be treated as *'related'*. That is, all *sibling* vectors must be placed before we say any one workload has been placed. Extending the algorithms to achieve this was considerable, and thus to then evaluate these extensions to other bin-packing routines, such as Best-Fit and Next-Fit or Harmonic fit algorithms would of been considerable. To evaluate our bin-packing extensions from First-Fit Decreasing to Next-Fit, Best-Fit, Worst-Fit or Harmonic (Harmonic-k, Refined or Modified) would have elongated the research of this thesis outside of the remit originally outlined. Bin-packing routines are extensively used, which we only touched on the simplest algorithm to evaluate complex clustered and pluggable workloads.

Impact Factor:(Dec 2022) Unavailable as still being computed...

Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure

Antony S. Higginson
Oracle Advanced Customer Services
Manchester, United Kingdom
antony.higginson@oracle.com

Clive Bostock
Oracle Product Development
Manchester, United Kingdom
clive.bostock@oracle.com

Norman W. Paton
University of Manchester
Manchester, United Kingdom
norman.paton@manchester.ac.uk

Suzanne M. Embury
University of Manchester
Manchester, United Kingdom
suzanne.m.embury@manchester.ac.uk

ABSTRACT

Capacity planning is an essential activity in the procurement and daily running of any multi-server computer system. Workload placement is a well known problem and there are several solutions to help address capacity planning problems of knowing *where*, *when* and *how much* resource is needed to place workloads of varying shapes (resources consumed). Bin-packing algorithms are used extensively in addressing workload placement problems, however, we propose that extensions to existing bin-packing algorithms are required when dealing with workloads from advanced computational architectures such as clustering and consolidation (pluggable), or workloads that exhibit complex data patterns in their *signals*, such as seasonality, trend and/or shocks (exogenous or otherwise). These extentions are especially needed when consolidating workloads together, for example, consolidation of multiple databases into one (*pluggable databases*) to reduce database server sprawl on estates. In this paper we address bin-packing for singular or clustered environments and propose new algorithms that introduce a time element, giving a richer understanding of the resources requested when workloads are consolidated together, ensuring High Availability (HA) for workloads obtained from advanced database configurations. An experimental evaluation shows that the approach we propose reduces the risk of provisioning wastage in *pay-as-you-go* cloud architectures.

1 INTRODUCTION

When employing I.T. to satisfy requirements, regardless of the type, combination or configuration, one thing has always been prevalent, which is the question of consumption, whether it is prior to a migration, re-platform, upgrade or installation. Understanding what resources are required over a period of time is key to the management of all I.T. systems. As hardware specifications increase, for example, in performance and capacity, the actual physical infrastructure decreases but with the adoption of virtualisation, the problem of server sprawl, arguably stays the same. The trade off between system separation conflicts with *true* consolidation that requires the workloads to be combined together or share the same infrastructure. Knowing how best to *fit* workloads together is a problem that has always been an important problem to solve.

Bin-packing is a well understood concept and used extensively in many fields of business. In computing, bin-packing can be used to place smaller workloads into larger infrastructure to establish how resources should be assigned to a set number of tasks. However, bin-packing requires additional constraints when dealing with advanced system architectures such as clustering and elasticising workloads once placed.

In this paper, we tackle placement of different workloads from advanced databases configurations into complex cloud architectures, and make the following contributions:

- We identify the challenges and opportunities presented by advanced architectural features, when placing database workloads into complex cloud infrastructrures.
- We present a new bin-packing algorithm for provisioning database workloads that takes into account fine-grained monitoring information and advanced architectures such as clustered Oracle databases that enable High Availability (HA).
- We evaluate the algorithms in experiments that involve the placement of diverse workloads into real world-cloud architectures.

2 PROBLEM DESCRIPTION

This section provides more detail on the problem to be solved, which is multifaceted. When the placement problem is broken down it becomes apparent that these facets are inter-related, that is to say, all parts of the problem need to be addressed together, rather than just individual elements. Before describing the problem, we provide some details on how workloads are executed on complex advanced database architectures and the relevent metrics.

Clustering. Clusters are groups of servers (also known as nodes) that are managed together, participating to act as one system, usually to provide high availability, as shown in Fig 1. These clusters reduce downtime and outages by allowing another server to take over should an outage occur or maintenance be required. Database clustering is the running of a database across multiple servers while accessing shared storage, for example, database datafiles that are stored on a SAN (Storage Area Network), NAS (Network Attached Storage) or a disk array. In Fig 1 we can see that the user's wish to access the *HR*, *Sales* or *Call Centre* applications from any web server. The Net Services layer of the technology stack, handles client access and directs users connections to the node where the service is running, for example, *Sales* is directed to run from node 2. A heartbeat between the nodes ensures cluster integrity, should one of the nodes no longer react or

produce a heartbeat, the service fails over and user connections are handled by the remaining nodes. This type of architecture facilitates a 24*7 SLA and is common in enterprises today.

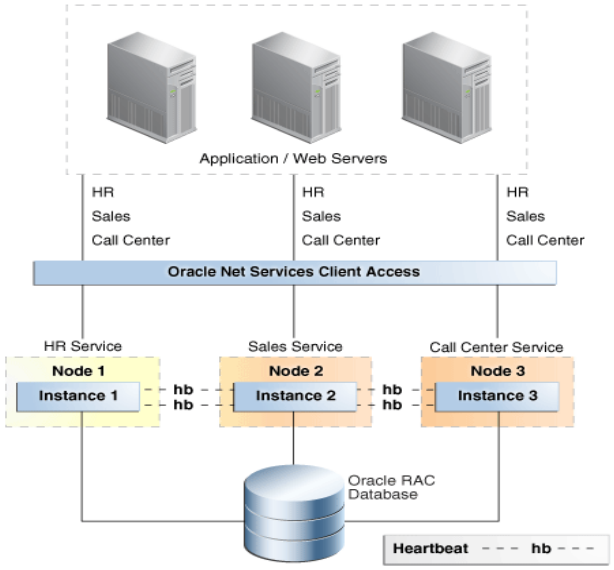


Figure 1: Oracle Database with Oracle RAC Architecture [6]

Workloads. Workloads consist of collections of tasks submitted by users. These tasks can be small units of work or individual pieces of SQL such as Data Modification Language statements (DML) that perform inserts, updates and deletes, that serve a web application, for example by way of an Online Transaction Processing system (OLTP). Other workloads consist of larger units of work such as batch jobs that aggregate information in the database periodically, for example, aggregation of sales data from hourly into daily, weekly or monthly, which is a common feature in data warehousing (OLAP). Another type of workload is a Data Mart that can be described as somewhere in-between OLTP and OLAP. Data marts consist of a combination of smaller DML OLTP units of work with medium OLAP type aggregations. Data Marts can be a subset of a large data warehouse that are subject-orientated such as sales, HR or Call Centre that are an aggregation of days and weeks rather than months or years.

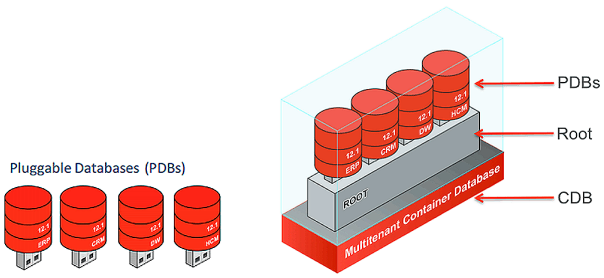


Figure 2: Oracle Database Multitenant Architecture [5]

These units of work vary in the amounts of resources consumed, and when analysed in a time series format, the task often dictates when the consumption of resources consumed to satisfy the task takes place. When these tasks are analysed via a trace we see those tasks exhibiting different patterns in their resource consumption. For example, in Fig 3 we have four workloads for CPU side by side. The first task, OLTP, shows a progressive trend with subtle repeating patterns (Seasonality). The next two tasks are OLAP showing a more definitive pattern of repeating tasks with little trend.

When placing workloads, we must treat each workload separately by extracting the *peak* values of each metric from each database instance on each node in each time interval, placing them on the target node. This is simple enough as long as the workloads are singular (independent of each other, and run on a single node). When the workload is clustered it becomes more complicated because, when placement commences, we must place the workloads while ensuring that we do not compromise HA. Therefore, for one clustered workload to be placed all workloads in the cluster must be placed. If we place a clustered workload without its sibling, we risk reducing a clustered workload to a singular workload, the consequence being that we lose high availability, thus compromising SLA's. The same understanding is required for pluggable databases [5], where a database may be detached from a singular database instance and plugged into another clustered database instance. This highlights that simple bin-packing routines are not suitable.

Consolidation. Consolidation can be described as running several workloads on a shared collection of nodes. There are several drivers for consolidation, such as system simplification or reducing server sprawl, whether that is the number of servers, clusters, databases or workloads. Server spawl can be described as a situation where servers are not being used to their full capacity, leading to significant wastage in terms of space, power and cooling, which can end up costing organizations substantial amounts of money.

Consolidation of databases has become easier with the development of pluggable databases where a database can be detached from a singular or Clustered Container Database (CDB), and plugged into another container DBMS that already has multiple plugged in databases. Detaching and attaching pluggable databases allows the pluggable database (and its associated data files) to be relocated to another server and be managed by another database instance (DBMS). This is shown in Fig 2, adding a further layer of abstraction when working in conjunction with HA. Each node in the cluster also houses a clustered container and within each clustered container there are pluggable databases. This architecture removes the support overhead of the database instance serving one database when one database instance can serve multiple plugged in databases while achieving HA. However, extracting the metric consumption on an instance with multiple pluggable databases residing together is challenging as the metric consumption is cumulative to the container. In this pluggable architecture, one must first separate the resource consumption for each pluggable, treating the pluggable database as a singular database workload.

Problem Statement. The problem to be solved can be considered as follows. Assume we are given a collection of *Workloads*, some of which are clustered, and a collection of computational *Nodes*. Each workload has a *time-varying* demand for resources defined using several metrics, and each server has a capacity defined using the same metrics. The task is to assign the Workloads to the Nodes, such that the demands placed by the workloads

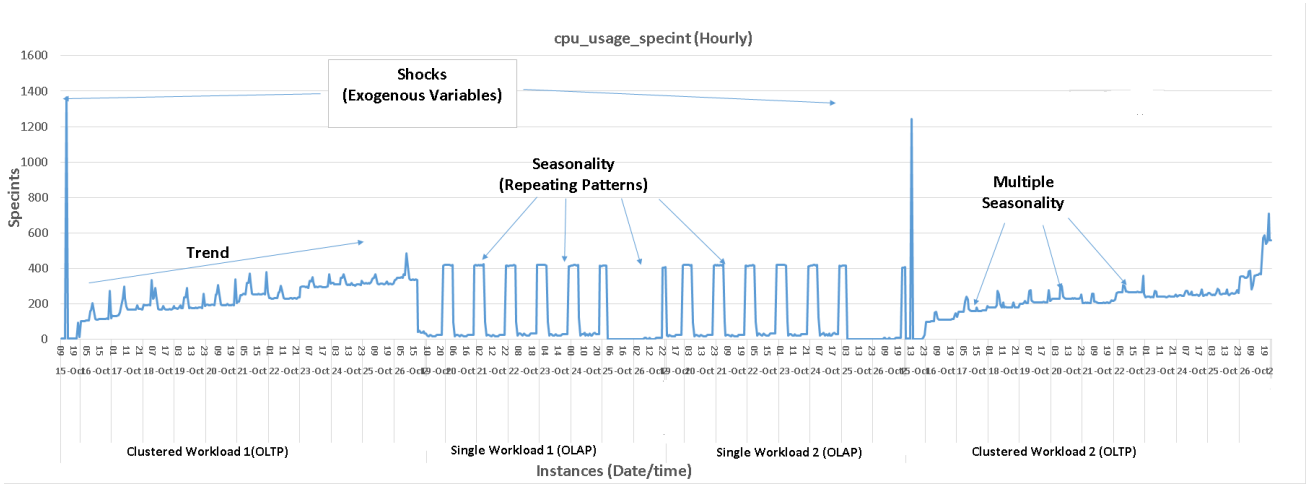


Figure 3: CPU Usage: Complex data structures.

are always within the capacity of the nodes, while respecting the constraints imposed by the clustered workloads.

3 RELATED WORK

Resource allocation or workload(s) placement in cloud environments is a well understood problem and there have been extensive studies and surveys undertaken as Hameed, Khoshkbarforoushha *et al* [16] and Bhavani and Guruprasad [4] both allude to in their survey's from 2014. Furthermore Singh and Chana [22] produced an extensive survey in 2016 that concluded that resource provisioning is a challenging job and there is a need for more research into optimal resource usage as this leads to improving the resources consumed with the aim of reducing wastage. This problem is ever more apparent in cloud computing with users accessing any shape of resources (vectors) from anywhere with the requirement of still being optimised. Vectors can be described as multiple metrics making up a shape rather than one singular metric. Most research has looked at consolidation from a Virtual Machine (VM) perspective to satisfy Quality of Service (QoS), and these are often single dimensions not vector dimensions. For example, placing workloads based on priority or assigning workloads to a VM via a migration to move problematic workloads should they exhaust a pool of resources as suggested by Yu, Wui *et al* [24]. In this survey, they identify several key problems of trying to figure out which applications can be consolidated together. Bin-packing algorithms do not take into consideration the divergent types of applications assigned to singular servers.

Other authors such as Sen, Ramachandra [21] and Zhang, Martin *et al* [25] view database workloads as controllable through the internal features of the database system via resource managers. However, they highlight the difficulties of the approach on cloud platforms where the resources are shared in the infrastructure. Halfpap and Schlosser [15] used an heuristic linear programming model to solve a placement problem by dissecting a database and where appropriate, replicating the data across multiple nodes. In doing so, this placement technique shared the workload from being cumulative on one node to being distributed between multiple nodes while keeping response times of the requests from said database optimum.

Most computing placement problems seem to centre on the provisioning of a VM (IaaS or PaaS) and not a workload SaaS or DBaaS. We identified in our previous work [18] that VM's mask the true signal of what the workload is actually consuming and that forecasting future resource requirements with a view to provisioning requires careful consideration of the nuances within the signal. Masdari and Khoshnevis in their 2019 survey [20] identified the techniques used to perform accurate forecasts of the resources being consumed as a precursor to provisioning. However, they stopped short of proposing how to *place* the workloads together once the forecasted future requirements are obtained.

When it comes to vector placement, using the bin-packing approach, several proposals have been put forward on prevalent customer use cases. For example, Wang, Hayat *et al* [23] looked at the scenario of a customer having multiple applications that make up a system. They propose a collection of algorithms based on a customers SLA as an important requirement to ensure business continuity. As application code or SQL is executed, and response times elongate this invariably lead to outages, provisioning the applications optimally by keeping these application response times as low as possible. Aydin, Muter *et al* [2], looked at another use-case for a VM placement problem with an added dimension of time. They looked for efficiencies in the power consumed by VM's fire-ups, with the aim of reducing energy consumption of data centres by minimising the number of servers and their fire-ups.

A current common theme with placement in a cloud setting is server sprawl and provisioning VM's. Doddavula, Kaushik and Jain [12], suggested reducing server sprawl with the introduction of a vector packing algorithm. In their novel approach, they classify vectors based on resource consumption, and then through Matrix multiplication determine the possible combinations. By then applying rules, either the workload is full or a magnitude of *full* determine where the workload should reside with other workloads until the maximum of the target server threshold is reached. However, in a clustered environment this approach will face challenges as it's possible that several workloads running on the same cluster are *full* or a combination of *classifications* that could break their algorithm. Clustered workloads that required enforced SLA's where workloads of differing priority may run from x node in the cluster as explained in Section 2.

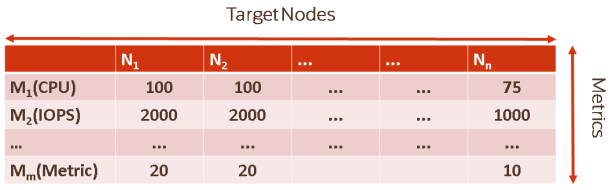


Figure 4: Nodes: Resource capacity.

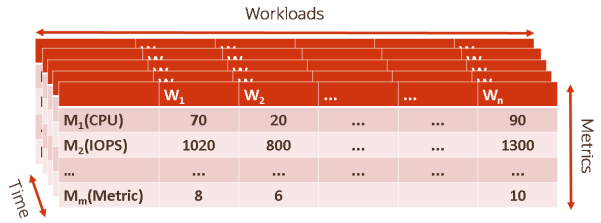


Figure 5: Workloads: Resource demand.

Notation	Interpretation
$Nodes = \{n_1, ..., n_n\}$	Computational Nodes.
$Workloads = \{w_1, ..., w_w\}$	Workloads.
$Metrics = \{m_1, ..., m_m\}$	Metrics (CPU, IOPS, etc).
$Times = \{t_1, ..., t_t\}$	Time intervals.
$Assignment(n_i) \rightarrow W$	The set of workloads $W \subseteq Workloads$ assigned to n_i.
$Capacity(n_i, m_j)$	The maximum capability of node n_i in relation to metric m_j.
$Demand(w_i, m_j, t_k)$	The peak demand that workload w_i has for resource of type m_j during time interval t_k.
$isClustered(w_i)$	True if w_i is in a clustered workload.
$Siblings(w_i)$	The set of workloads $s \subseteq Workloadss$ in the cluster of which w_i is a member.

Table 1: Notation for workload assignment.

4 BIN-PACKING

The basic bin-packing problem is the process of taking items of differing volumes and *packing* them into a finite number of *bins* in a way that minimises the number of bins used. The bin-packing problem is NP-complete as described by both Garey [13] and Korte *et al* [19], and thus approximate, heuristic, algorithms are often used in practice. There are many approaches to bin-packing, such as First-Fit Decreasing (FFD), Next-Fit (NF) and Best-Fit (BF) as discussed by Carter and Bays [3]. We look at First Fit Decreasing, and therefore, we treat all workloads being provisioned as having equal priority. Elastic Resource Provisioning (ERP) is assigning all workloads into one bin and elasticising the bin to fit around the workloads being placed, as described by Yu, Qiu *et al.* [24]. Our approach is to enhance FFD to tackle complex architectures such as clustered workloads and empirically evaluate them experimentally.

4.1 First Fit Decreasing Placement (FFD)

The notation used to describe bin-packing in this paper is listed in Table 1 and illustrated in Figures 4 and 5. The available resource is represented as a set of *Nodes*, each of which has a *Capacity* defined using a set of *Metrics*, that can include CPU, memory usage, logical IOs per second, etc. The bin-packing task is to assign a set of *Workloads* to the nodes. The *Demand* associated with each workload is defined in terms of the same metrics that are used to describe the nodes, but the *Demand* varies over a set of *Times*. The time-varying demand may be based on measured or predicted loads; our earlier work has explored the use of time series analysis techniques to model database workloads for capacity planning [18].

Sorting Workloads. First fit decreasing sorts workloads such that they can be assigned largest-first, and hence we need a notion of size; here we define order in terms of the demand across different metrics, normalising according to the total usage for each metric.

Overall demand for each metric (CPU, IOPS, MEMORY, STORAGE) is obtained from the demand of each workload:

$$overall\_demand(m) = \sum_{n_i \in Nodes} \sum_{w_j \in Workloads} \sum_{t_k \in Times} Demand(w_j, m, t_k) \quad (1)$$

Then the normalised demand of a workload w is the normalised sum of its demand over all metrics:

$$normalised\_demand(w) = (\sum_{m_j \in Metrics} \sum_{t_j \in Times} (Demand(w, m_j, t_k)/overall\_demand(m_k)) \quad (2)$$

Then the workloads can simply be sorted by their normalised demand. In practice, when assigning clustered workloads, clusters are considered in the order of the demand of their most demanding workloads, and then the workloads within a cluster are also sorted locally.

Node Capacity. The capacity of a node i for a metric m at time t is the original capacity of the node minus the resource usage of the workloads assigned to the node:

$$node\_capacity(n, m, t) = Capacity(n, m) - (\sum_{w_i \in Assignment(n)} Demand(w_i, m, t)) \quad (3)$$

Fitting. A workload w can be added into a node n if there is capacity for all the metrics at all times.

$$fits(w, n) = \forall m \in Metrics \; \forall t \in Times \; Demand(w, m, t) \leq node\_capacity(n, m, t) \quad (4)$$

A clustered workload, *isClustered* from Table 1, consists of database instances (workloads) that are siblings of each other as shown in Fig 1. The node is the number of nodes on which W_i is to be run, and $w \in ClusteredWorkload$ is the set of workloads that need to be assigned to the target nodes discretely. A rule is enforced where all *Siblings* must be packed into discrete target nodes before the cluster is said to be packed. If at any point one of the *Siblings* fails to pack into a discrete target node then all

siblings are rolled back and the resources are released back to *node_capacity*.

5 APPROACH

Using techniques based on First Fit Decreasing (FFD) and the definitions from Section 4, our aim is to place database workloads into a target Oracle Cloud Infrastructure (OCI) [7] that supports clustered workflows. The aim is to achieving savings in costs, both financial *(pay-as-you-go)* and to release resources back to the cloud pool for utilisation elsewhere.

5.1 Workload Placement Algorithm

A high level description is shown in Algorithm 1. One of the major challenges of workload placement in computing estates that have adopted clustered configurations, is accounting for clustered workflows, that must be placed in their entirety or not at all. Identifying the workload is clustered and on how many nodes is key to placing the workload in the target cloud OCI configuration.

Algorithm 1: FitWorkloads

```
Input: Workloads (from Table 1)
       Nodes (from Table 1)
Result: Assignment(n) (from Table 1)
        NotAssigned
Assignment(n) = ∅ for all n ∈ Nodes
NotAssigned = ∅
foreach w ∈ Workloads ordered by normalised_demand (Equation 2) do
    assigned = false
    foreach n ∈ Nodes do
        if isClustered(w_i) then
            if w_i not already added to Assignment with cluster or included in NotAssigned then
                assigned = FitClusteredWorkload(Siblings(w_i), Nodes, Assignment, NotAssigned)
            if assigned then
                break
        else if fits(w,n) then
            Add w to Assignment(n)
            assigned = true
            reduce node Capacity (Equation 3) by Assignment
            break
    if (not assigned) then
        Add w to NotAssigned
Report on Workloads Assigned, NotAssigned and Nodes Capacity
```

Firstly we extract key information as inputs, ordering workloads by demand. Key configuration data is stored in a central repository [8] that stores whether a workload is clustered or not. If a workload is part of a cluster then we set a flag for that particular workload (represented by *isClustered* in Table 1), and the full set of workloads with which it is clustered can be obtained using *Siblings* in Table 1.

When placing workloads, if the workload w is from a single database instance then we simply check if the workload $fits$ (Equation 4) into an available node, and if so, add it to *Assignment* for that node. We report back to the user all workloads that have been fitted (by way of *Assignment*), and any that have not (by way of *NotAssigned*).

If, however, the workload is clustered we extract the related *Sibling* workloads in the cluster from a central repository; as we use the Oracle Enterprise Manager system [8] and the OEM intelligent agent to capture all performance and configuration data relating to the database instances. OEM utilises a database schema to hold information relating to the workloads, and databases instances, and we handle this via a Global Unique Identifier (GUID).

5.2 Fitting Clustered Workloads

The fitting of clustered workloads, aims to enforce high availability by placing *all* workloads in a cluster before it can report back that the workloads are fitted. For example, if the clustered workload has three nodes with a database instance running on each node as described in Fig 1, then it must place the workloads on three discrete target nodes or it will roll back what has already been placed. We show this in Algorithm 2.

Firstly we understand how many nodes make up the cluster (1,2,..,N) nodes, which gives an indication of how many target nodes are required. We cannot fit a clustered workload from three nodes into two target nodes, therefore, we perform a test to ensure that the requisite number of target nodes are available. If there are not enough target nodes then we stop otherwise we loop through all of the workloads ensuring that the siblings of the cluster are assigned to discrete target nodes. Each time an assignment takes place the amount of resource of the target node is reduced by the vector of the workload. Finally, we report on what workloads are assigned to each target node.

5.3 Evaluating the Placement

Once the workloads from all database instances have been assigned and placed in their target nodes. We overlay each workload by the time frequency (Hourly), allowing an understanding of the existing data signals (peaks and troughs) when the workloads are consolidated together. A simple groupby ($\sum$) per hour and per metric shows the newly consolidated data signal. In traditional bin-packing exercises, the *max_value* of a metric is taken and then bin-packing is based on that value, however, if a peak is singular, for example, without pattern then the prospect of over provisioning becomes apparent. When the new trace is displayed over an *X,Y* (Stacked), which we show in Section 7.2 and Fig 7. We can clearly see the consolidated workloads exhibit their complex traits such as seasonality, trend and shocks against the threshold limit of the bin. This simple approach allows us to understand and where possible, perform or feed into further elastication exercises that can be performed on the bin to fit the consolidated workloads more tightly.

6 EXPERIMENTAL SETUP AND WORKLOADS

Firstly we execute a selection of workloads (OLTP, OLAP and Data Mart) on different Oracle database configurations of varying versions (10g, 11g and 12C) on a Oracle Enterprise Linux Operating System or Oracle Exadata (clustered workload) [9]. Executing the workloads for 30 days allows key database features such as optimisers and caching to be warmed up or routine backups to take place, which all consume resources and influence

Table 2: Table of Experiments

Experiment	Workloads	Target Bins
Basic Single Database Instance	10 Workloads (10 OLTP, 10 OLAP and 10 DM)	4 * OCI Bare Metal equal size
Basic Clustered Workloads	10 Workloads (10 RAC OLTP (5*2 Exadata nodes))	4 * OCI Bare Metal equal size
Basic different sized target bins	10 Workloads (10 OLTP, 10 OLAP and 10 DM)	4 * OCI Bare Metal unequal size
Moderate Combined (Clustered and Single Instance)	20 Workloads (4 * 2 node clustered + 5 OLTP, 6 OLAP and 5 DM)	4 * OCI Bare Metal unequal size
Moderate scaling	50 Workloads (10 * 2 node clustered + 10 OLTP, 10 OLAP and 10 DM	4 * OCI Bare Metal equal size
Moderate different sized target bins	20 Workloads (4 * 2 node clustered + 5 OLTP, 6 OLAP and 5 DM	6 * unequal OCI Bare Metal
Complex (Scaling & different sized bins)	50 Workloads (10 * 2 node clustered + 10 OLTP, 10 OLAP and 10 DM	16 * unequal OCI Bare Metal

Algorithm 2: FitClusteredWorkload

```
Input: ClusteredWorkload, including sibling data (from
       Algorithm 1)
Nodes (from Table 1)
Assignment(n) (from Table 1)
NotAssigned (from Algorithm 1)
Result: assigned
foreach w ∈ ClusteredWorkload ordered by
  normalised_demand (Equation 2) do
    assigned = false
    if target_nodes are <= source_nodes then
        foreach n ∈ Nodes do
            if fits(w,n) then
                Add w to Assignment(n)
                assigned = true
                reduce node Capacity (Equation 3) by
                  Assignment
                break
        if (not assigned) then
            Remove all members of ClusteredWorkload
              from Assigned
            Add all members of ClusteredWorkload to
              NotAssigned
            increase node Capacity (Equation 3) by
              Assignment
            break
        not enough nodes to fit
Report on Workloads Assigned and NotAssigned
return assigned
```

the bin-packing routines by way of providing a different metric values between a cold and warm database. We also monitor the workloads to ensure the workloads are running smoothly without error. The workloads are executing Data Manipulation Language (DML) statements such as inserts, updates and deletes while performing large data aggregations, for example Business Intelligence (BI) reports through a Java environment consisting of a web container, giving us an N-Tier architecture using the Oracle load generator Swingbench [14]. This environmental setup is reflective of Database Management systems in use today.

We utilise the Oracle Enterprise Manager [8] system to execute and capture all performance and configuration data via an Intelligent Agent, which captures metrics such as CPU, IOPS, Memory and Storage used. The agent executes commands to retrieve the *max_values* of key metrics such as *sar*, *iostat*, and *memory* on the host and metrics specifically from the database are also obtained such as storage used, memory used, CPU % used and logical read/writes. The agent captures these metrics at 15 minute intervals and stores the values in a central repository. Aggregations on the data captured every 15 minutes are then performed providing a *max_value* for each metric for each database instance and host hourly, daily, weekly or monthly.

Storing the values in a central repository this way, enables the ability to align the metrics uniformly over consistent observations such as hourly in an overlay manner, allowing an easy comparison of all database instances. Using python open source libraries such as numpy and pandas we then executed the algorithms and empirically evaluated the placement of the workloads into target bins. We could use *average_values* from the metrics captured but we choose *max_values* for the simple reason of provisioning on an average will usually be lower than a *max_value* and if a VM hits 100% utilised it will panic and may cause an outage. Therefore, we always place on a max_value from a metric.

Each workload generates complex data traces as shown in Fig 3, highlighting repeating patterns (seasonality), trend and shocks. Shocks are reflective of large IO operations, for example online database backups, and this can be seen in the metric IOPS. In Fig 3 we have displayed four workloads side-by-side for CPU usage that highlights these complex patterns, which are indicative of systems employed by most enterprises today. As workloads become larger in size, arguably the result is slower execution times and this is shown by the workloads exhibiting trend. Each experiment increases in complexity and scale as the number of workloads increase, which is described in Table 2. Each experiment and its results are discussed in detail in Section 7 where we produce charts that highlight the workloads consolidated together after being placed on a particular node. The target configuration is Oracle Cloud Infrastructure (OCI) [7] using the Bare Metal Configuration as shown in Table 3.

In the experiments we are only testing the database placement algorithms, as they are orthogonal to modelling. The placement algorithms do not know if the traces being inserted as inputs to the algorithms are actual or modelled, however, it is perfectly plausible that the inputs have first been predicted to obtain an

estimate of future resource consumption to model what a placement design may look like, which is a common planning exercise in any estate migration. Once the workloads are placed we also evaluate the consolidated workloads assigned to the target nodes to identify if there is any wastage that can be further eliminated by, for example, an elastication exercise.

Table 3: OCI Target Bin Configuration (BM.Standard.E3.128) [7]

Shape	Metric	Comments
Compute Shape	128 OCPU 2048 GB's Memory Block Storage	Equivalent to 980 SPECInts [10] per bin
Block Storage Shape	32 * 4TB Volumes 35,000 IOPS per vol	Equivalent to 1,120,000 IOPS per bin 128000GB Physical Storage
Network Shape	2* 50Gbps throughput Max 128 VNICS	65 per physical NIC

7 EXPERIMENTS AND ANALYSIS

We conducted several experiments that start off simple and then grow in complexity providing a detailed evaluation of the algorithms that reflect use cases of estates employed by customers today. For the purposes of saving space in the paper we have only included some of the most prominent charts and results that are interesting. Furthermore we do not describe all experiments listed in Table 2 to save space in the paper. The experiments we do show highlight the algorithms working to their full potential. We aim to answer the following questions:

(1) Minimum targets needed - What is the minimum number of target bins needed to fit all workloads across all vectors (metrics)?
(2) First Fit Decreasing Simple Placement - How do we place the workloads equally across equal sized bins?
(3) First Fit Decreasing Clustered Placement - If there are clustered workloads can we ensure that all clustered workloads are placed without compromising High Availability?
(4) Evaluating the placement - Once the workloads are placed (consolidated) together can we identify wastage with the aim to resize the target nodes, obtaining a tighter fit, reducing over provisioning?

We have included sample outputs from each experiment and the reader will note that these are not uniform in their outputs and this is because we wish to only highlight prominent outputs that answer questions the experiments pose. If we are to supply full sample outputs for each experiment we would quickly exhaust space in the paper. Also, we have used sample outputs as opposed to full UI screenshots, also saving space and showing the algorithms working. In our opinion, UI design although an important feature of any application is not as important, to this paper, as the algorithms working.

In all of our experiments we executed our algorithms multiple times with both adequate and inadequate target nodes and resources to ensure that the algorithms would place and reject workloads appropriately. In the experiments we report in this paper we took the average number of workloads we expect customers to provision per Oracle Cloud Architecture, i.e., customers

```
Can we fit all instances into minimum sized
bin for Vector CPU?

===== list

List of workloads
['DM_12C_1': 424.026, 'DM_12C_2': 424.026,
'DM_12C_3': 424.026, 'DM_12C_4': 424.026,
'DM_12C_5': 424.026,'DM_12C_6': 424.026,
'DM_12C_7': 424.026,'DM_12C_8': 424.026,
'DM_12C_9': 424.026, 'DM_12C_10': 424.026]

Target Bins 0
['DM_12C_1': 424.026, 'DM_12C_2': 424.026,
'DM_12C_3': 424.026, 'DM_12C_4': 424.026,
'DM_12C_5': 424.026,'DM_12C_6': 424.026]

Target Bins 1
['DM_12C_7': 424.026, 'DM_12C_8': 424.026,
'DM_12C_9': 424.026, 'DM_12C_10': 424.026]
```

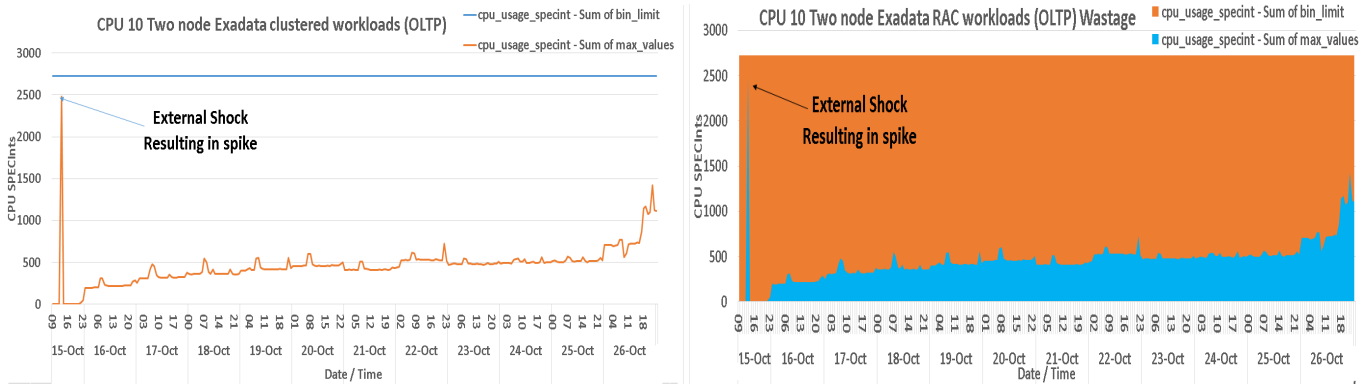
Figure 6: Sample output: Minimum Number of Nodes CPU

mostly provision a *1-to-1* relationship of an instance per VM. However, consolidation of workloads is rising as the technology allows; for example combining database workloads together is more achievable with the introduction of Pluggable databases [5] where one database can be detached from one instance and *'plugged'* into another instance. An instance includes the memory structures and optimiser that serves the database. The instance is then shared across multiple VM's that make up a physical cluster thus bin-packing multiple instances together is becoming more apparent.

7.1 Experiment (Basic) - Placement of Single Database Workloads (OLTP, OLAP & DM Workloads)

Overview. The first experiment involves placing 10 OLTP, 10 Data Mart and 10 Data Mart workloads from a source configuration of Oracle single database instances as described in Table 2, into four equally sized target OCI bins of a configuration described in Table 3. For the purpose of savings space in the paper have not shown these charts here. We show the sample outputs from the command line which we discuss in detail in *results*.

Results. In the sample outputs shown in Fig 6, we represent one metric (CPU) in the vector, although in our outputs, we cover all metrics in the vector. Each of the *max_values* taken from the hourly time period are listed as one list and then *placed* into the minimum number of bins. Each bin is represented within square brackets *'[]'*. In Fig 6, each workload is labelled by using a precursor to the value. For example workload DM_12C_1 identifies, DM representing the type of workload thus Data Mart, 12C representing the version of Oracle database the workload was executed with, and 1..10 being that actual workload. The values after the *':'* in Fig 6 is the *max_value*. As we show in Fig 6, we have successfully answered question 1, treating all workloads heuristically, what is the minimum number of target nodes required to fit my workloads?



(a) CPU OLTP RAC Workloads

(b) CPU OLTP RAC Wastage

Figure 7: RESULTS: Consolidated placed workloads & Potential Wastage

```
How many of the instances (Database Workloads)
can we get in 4 equal sized bins?

bin packed it looks like this
Target Bins 0
{'DM_12C_9': 424.026, 'DM_12C_5': 424.026,
'DM_12C_10': 424.026}
Target Bins 1
{'DM_12C_8': 424.026, 'DM_12C_4': 424.026,
'DM_12C_1': 424.026}
Target Bins 2
{'DM_12C_7': 424.026, 'DM_12C_3': 424.026}
Target Bins 3
{'DM_12C_6': 424.026, 'DM_12C_2': 424.026}
```

Figure 8: Sample output: Workloads placed equally across targets (CPU)

In Fig 8 we answer question 2, which is can we place the workloads equally across the target nodes? In the sample output shown in Fig 8 the target nodes are represented by brackets *'{}'*. Question 3 is not answered in this experiment as there are no clustered workloads however, we will answer this question in other experiments. Question 4 we ask in each experiment; evaluating the target nodes after placement can we resize the bins to obtain further savings? We have not shown these charts for this experiment due to the space available. We discuss this question in detail in Section 7.2 from charts a & b in Fig 7.

7.2 Experiment (Basic) - Placement of 10 Clustered workloads (RAC)

Overview. In this experiment, from Table 2, we focus on clustered workloads of a type OLTP, which are executed on an Oracle two node Exadata Machine [9]. There are 10 workloads, which equates to five two node clusters executed on an Oracle 11G version database. The workloads exhibit complex data structures as shown in Fig 3 such as seasonality, trend and exogenous shocks. In this experiment we are answering Question 3 of the first fit decreasing placement algorithm with the aim of enforcing High Availability. All workloads in a cluster must be placed or no workloads from the cluster will be placed as described in algorithm 2. The target bins are of type equal sized Oracle Cloud Infrastructure [7] and covered in Table 3.

Results. In the sample output shown in Fig 9, which we have cropped for ease of reading and space within the paper, we show the algorithm's command line output. This output is available in all experiments but we focus particularly on clustered workloads to highlight our placement algorithms working clearly. The target bin configurations are displayed along with the databases instances and their *max_values* for a given time period. The first block in Fig 9 titled *'Cloud Configurations'*, lists the target OCI bin vectors and their available space named OCI0,..,OCI6. The next block in Fig 9 titled *' Database instance / resource usage:'*, lists the source RAC database instances and their vectors, which we have also cropped for space within the paper by only showing four clustered databases instances (8 workloads).

We inform the user of the instances we are intending to place and as the algorithm is executed, providing a real-time decision of each instance being placed however, we have not shown this output here. Once executed we provide a summary of workloads placed, refused or rolled back, for example, if an workload from a clustered database instance has been placed but a sibling was not placed. It will require a rollback and the counter *'rollback count'* increments as shown in the block titled *'Summary'* in Fig 9. We observed in our tests that once an instance is rolled back, the resources are released and made available again, allowing a smaller vector size to be placed. We also provide a list of workloads that failed to fit due to lack of resources, reporting a list of failed instances to the user. In the block titled *'Cloud Target : DB Instance mappings:'*, we provide a mapping of the clustered workloads that were placed and their siblings, including which target node they are placed, note that no two instances from the same cluster are ever placed in the same target node; they are always placed discretely.

Evaluating the target nodes after placement, can we resize the bins to obtain further savings (Question 4)? Charts a & b in Fig 7 show several interesting points. Chart 7a shows an external shock that caused a spike and the FFD algorithm takes this *max_value* when placing workloads. When the workloads are consolidated together we can see trend as the line gradually rises. The available, target, resources are shown by the blue line and the large spike fits below the line. However, Chart 7b displays the potential CPU resources that will not be utilised (orange). Therefore,

```
Cloud configurations:
                     OCI0     OCI1 ...   OCI11 ...  OCI16
metric_column
cpu_usage_specint     2728     2728 ...    2364 ...  681.25
phys_iops          1120000  1120000 ...  560000 ...  280000
total_memory       2048000  2048000 ... 1024000 ...  512000

Database instances / resource usage:
RAC_1_OLTP_1 RAC_1_OLTP_2 RAC_2_OLTP_1 RAC_2_OLTP_2 RAC_3_OLTP_1 RAC_3_OLTP_2 RAC_4_OLTP_1 RAC_4_OLTP_2

metric_column
cpu_usage_specint  1,363.00   1,363.00    1,363.00    1,363.00   1,363.00    1,363.00    1,363.00   1,363.00
phys_iops         16,341.00  16,341.00   16,341.00   16,341.00  16,341.00   16,341.00   16,341.00  16,341.00
total_memory      13,822.00  13,822.00   13,822.00   13,822.00  13,822.00   13,822.00   13,822.00  13,822.00
USED_GB               53.47      53.47       53.47       53.47      53.47       53.47       53.47      53.47

SUMMARY                                 Cloud Target : DB Instance mappings:
=======                                 ====================================
Instance success: 8.                    OCI0 : RAC_1_OLTP_1, RAC_2_OLTP_2
Instance fails: 12.                     OCI1 : RAC_2_OLTP_1, RAC_3_OLTP_2
Rollback count: 0.                      OCI2 : RAC_3_OLTP_1, RAC_4_OLTP_2
Min OCI targets reqd: 10                OCI3 : RAC_4_OLTP_1, RAC_1_OLTP_2

Original vectors by bin-packed allocation:
                     OCI0    RAC_1_OLTP_1  RAC_2_OLTP_2
metric_column
cpu_usage_specint     2728     1,363.31     1,363.31
phys_iops          1120000    16,340.62     16,340.62
total_memory       2048000    13,822.21     13,822.21
```

Figure 9: Sample output: 10 RAC Workloads First Fit Decreasing High Availability Enforced

elasticising the target cloud node, and reassigning the resources would reduce wastage. In this experiment we have successfully placed clustered workloads proving our placement algorithms (Algorithm 2) and evaluated the consolidated workloads in their target nodes, successfully identifying potential wastage that may occur.

7.3 Experiment (Complex) - Placement of combined workloads (Clustered & Single Instance), varying sized bins at scale

Overview. This experiment is the most complex of all of the experiments we conducted. In this experiment we are answering the question of scale by placing a large number of workloads of varying size into different sized target nodes. The target nodes reduced in their available resources, which arguably, reflects the use case most customers face today when undertaking a migration exercise that involves procuring and placing workloads from on-premises advanced configurations such as clustering into cloud configurations. We tackle this experiment by running a combination of algorithms which are the following: -

- What is the minimum number of target nodes I require based on the size of my vectors?
- What is the maximum number of workloads I can fit into the available target nodes while keeping the integrity of the clustered workloads?
- Can we work at scale, which is a plethora of eclectic workloads into varying sizes of targets?

Firstly we obtained an estimate on the minimum number of target nodes based on the *max_values* obtained for each metric within the vector from each workload. Taking the configurations based in Table 3, the number of nodes needed to place 50 workloads was: -

- CPU - On this metric the advice was 16 target bins
- IOPS - On this metric the advice is 10 target bins
- Storage - On this metric the advice is 1 target bin
- Memory - on this metric the advice is 1 target bin

From analysis of the workloads we identified that some of our workloads are CPU and IOPS *heavy*, therefore, allowing the algorithms to utilise 16 available target nodes was key to meeting the demands of the experiment bearing in mind it is a question of scale. Utilising 16 target nodes of OCI configurations of varying sizes, these being 10 target bins 100%, 3 being 50% and 3 25% available resource from Table 3. In Fig 9 under Section *"Cloud Configurations"*, which we have cropped to aid viewing we show OCI11 (50%) and OCI16 (25%). Placing our workloads heuristically on a first fit decreasing method, all workloads are treated equally, but focusing on enforcing High Availability as there is a combination of both single and clustered workloads.

Results. The results of this experiment were very much the same as the previous experiments in that the algorithms (Algorithms 1 and 2) worked as expected. All the algorithms fitted their workloads in a First Fit Decreasing manner. All the Algorithms evaluated the nodes once placement of the workloads took place to identify further efficiencies. However, what we wish to focus

```
Rejected instances (failed to fit):
metric_column  cpu_usage_specint  phys_iops  total_mem
RAC_1_OLTP_1          1,363.31   47,982.17   13,882.21
RAC_7_OLTP_1          1,241.99   47,982.17   12,723.78
RAC_9_OLTP_2          1,241.99   47,982.17   12,723.78
RAC_9_OLTP_1          1,241.99   47,982.17   12,723.78
RAC_8_OLTP_2          1,241.99   47,982.17   12,723.78
RAC_1_OLTP_2          1,241.99   47,982.17   12,723.78
RAC_8_OLTP_1          1,241.99   47,982.17   12,723.78
RAC_10_OLTP_1         1,241.99   47,982.17   12,723.78
RAC_7_OLTP_2          1,241.99   47,982.17   12,723.78
RAC_10_OLTP_2         1,241.99   47,982.17   12,723.78
```

Figure 10: Sample output: Experiment 4 RAC workloads failed to fit

on in this experiment was the instances that failed to fit as shown in Fig 10.

From the sample output shown in Fig 10 it would suggest there is a random nature to the instances not being fitted and this is explained because of the following. When we first list the instances, their workloads, vectors and the amount of resource consumed we order them in descending order with the largest single instance being first then the largest RAC instances. By optimally sorting on size we avoid the algorithm rolling back already placed instances as the available target nodes exhaust their resources with siblings not been placed. We must treat the siblings of the clusters equally then sort order based on the size of the total cluster.

8 CONCLUSIONS AND FUTURE WORK

Summary. In this paper we evaluated the technique of Vector Bin-Packing utilising the First-Fit Decreasing algorithm against databases that employ advanced technologies such as clustering and pluggable databases with a view to fitting a variety of workloads into complex target cloud configurations such as Oracle Cloud Infrastructure with a Bare Metal configuration. When placing workloads into Cloud configurations because most cloud providers provision on multiple dimensions such as IOPS, Storage, CPU and Memory, a vector approach is required. If the *Cloud Consumer* is also a *Cloud Provider* then the vectors are likely to increase in number, covering other areas of cloud technology, for example Network throughput, Bandwidth or Virtual Network Interface Cards (VNIC) configuration to name but a few. The approach adopted provides the ability to place workloads on scaleable vectors, by increasing the number of metrics $[m_1, .., m_m]$.

We wanted to understand once the workloads were placed could we make further efficiencies? Given placement algorithms only take the *max_value* of a metric that is associated with its workload over time, once a workload migrates architectures, the *signal* changes, especially when analysed in a time series format as described from our earlier work [18]. Therefore, over-provisioning is possible without understanding if there are repeating patterns or trends within the signal. The charts in Fig's 7a and 7b, indicated by the colour orange shows that what, was initially, provisioned may not be used. Our approach identified this wastage.

- What is the maximum number of target nodes needed to consolidate my workloads?
- What size do I need those target nodes to be?
- How should those workloads be placed in the target nodes?
- Is the target node adequately sized once placement of the workloads takes place?
- Will placement of the workloads compromise my SLA's?

Given the popularity of advanced database features such as high availability and consolidation we had to extend the existing FFD bin-packing algorithms rather than simply mapping a database instance as a *1-to-1* mapping to a VM. By consolidating the workloads together, gave us additional complexities to take into consideration. For example, pluggable databases are still attached to a global database memory structure consuming resources. By treating a pluggable database as a single instance workload we were able to reduce complexity within the algorithms, allowing us to place pluggable databases. Our algorithms needed to be multi-faceted in that they can place simple, complex and very complex vectors attributed to any database workload regardless of the source database configurations. By treating pluggable and standby databases as a single instance workload allowed us to perform workload placement without introducing further notation in our formulas. A standby database will usually be in recovery mode applying all archivelogs from all nodes in the primary cluster therefore, a standby is a single instance which is more IO resource intensive than memory or CPU as we have shown in our earlier work [17].

Central Repository. Using an intelligent agent capable of Monitor Analyse Plan and Execute (MAPE) (Arcaini *et al* [1]) to identify, capture, store metric and configuration data centrally, allowed us to align the time series data of the workloads uniformly. An intelligent agent executes a command for example *sar* or *IOSTAT* at a particular time with the command results being stored in a central repository within a database schema. Aggregations are performed on the metric data to an hourly value and while this has the negative affect of smoothing the signal (averaging the time points) it allowed us to compare the workloads at any given time period easily, as shown in Fig 5, reducing the amount of data wrangling in the application layer by python libraries such as Pandas, Numpy etc.

Benchmarks. Comparing Servers with different performance speeds such as IOPS or CPU is a challenge and there we utilised benchmarks. SPECInt 2017 [10] was used to compare the workload consuming CPU on one architecture compared with another chip architecture. Storage benchmarks were also provided based on Transaction Processing Performance Council [11] benchmarks. However, RDBM systems utilise complex memory algorithms that often bypass fetch operations of the database therefore, logical reads were taken as the metric. However, given our approach and algorithms allows placement on a vector that is scaleable, other Metrics such as physical IOPS could be used if one chooses to do so.

Automation. With the manual approach of performing a workload placement exercise, technicians tend to adopt a spreadsheet approach when placing workloads into clouds. This approach can be cumbersome, for example, manually researching, converting the CPU (SPECint), IO speeds and Memory between the source and target architectures, so creating the spreadsheet is time consuming. Often these spreadsheets build in complexity and are bespoke to individual customers resulting in inflexibility, resulting in *'expert friendly'* analysis that only the author understands. We wanted to automate this process with the aim of reducing the

level of effort technicians spend manually building spreadsheets, reducing errors from miscalculations that may occur in bespoke spreadsheets and reduce the time to complete a placement plan from weeks/months to hours/days.

When we execute our algorithms we are effectively retrieving the configuration and performance data from a central repository. For example, extracting the CPU make, model and its SPECInt value that is obtained by the intelligent agent, therefore performing a comparison rather than manually researching this data. The Algorithm can then quickly place and store the placement design of the workloads as a *'plan'* in a normalised database schema, rather than having complex sets of bespoke spreadsheets. This approach worked well, allowing us to execute the placement algorithms in minutes rather than days or weeks.

Conclusion. In conclusion, we believe that there is a need for accurate workload placement especially when provisioning services such as IaaS, PaaS, DBaaS or SaaS, whether that is on-premise, remote or hybrid clouds. However, knowing which algorithm or collection of placement algorithms to use is key as one simply can not utilise a standard approach or an *off-the-shelf* technique when advanced workload configurations such as clustering are employed as our experiments show. We focused on the First-Fit Decreasing bin-packing method on advanced databases architectures such as Clustering and Pluggables and found that we needed to extend the FFD algorithm to accommodate sibling workloads within the cluster, especially when the cluster is consuming resources unevenly.

Future Work. During our experiments we found curious behaviour causing us to extend our new FFD Algorithm further, and this was attributed to ordering the workloads prior to placement, something we did not expect. The result was to insert steps 1 and 3 of algorithms 1 and 2 to include ordering in descending order with the largest workload being the first to be placed while also ordering the largest available resource target nodes being first. However, we did not account for the siblings when ordering clustered workloads. Therefore, we had to order the workloads *and* their siblings together. The reasoning for this is to account for the siblings in a cluster that consume resources unevenly. When clustered instance workloads are listed individually one workload within a cluster can be located considerably down the pecking order compared with its sibling if a simple ordering exercise takes place. Eventually the target nodes are exhausted of their resources and placement ceases. If the target node runs out of available resources before the sibling is placed then a rolling back exercised is performed.

Therefore, it is critical to order on the cluster and its siblings in descending order. This is more of a work around really rather than a solution. We are working on a solution to this problem but the likely answer will be to take the cumulative approach of the total amount of resources consumed per cluster and order based on number of nodes then resources consumed. We also intend to enable a user defined priority assignment function. Assigning a user defined priority to a workload allows for separation between live systems as some systems may be more critical than others. In our earlier work we leveraged Machine Learning (Supervised) coupled with Time Series Analysis [18] to predict future resource consumption of a workload and we see a combination of those techniques and placement to create a more informed choice when one is making decisions on what systems can be placed based on their future resource consumption.

REFERENCES

[1] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. 2015. Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 13–23. https://doi.org/10.1109/SEAMS.2015.10

[2] Nurşen Aydın, İbrahim Muter, and İlker Birbil. 2020. Multi-objective temporal bin packing problem: An application in cloud computing. *Computers and Operations Research* 121 (2020), 104959–.

[3] Carter Bays. 1977. A Comparison of Next-Fit, First-Fit, and Best-Fit. *Commun. ACM* 20, 3 (March 1977), 191–192. https://doi.org/10.1145/359436.359453

[4] Bhavani and Guruprasad. 2014. Resource Provisioning Techniques in Cloud Computing Environment: A Survey. *IJRCCT International Journal of Research in Computer and Communication Technology* 3, 7 (2014), 395 – 401.

[5] Oracle Corporation. 2017. *Oracle Multitenant Architecture.* https://www.oracle.com/uk/database/multitenant/.

[6] Oracle Corporation. 2020. *Oracle Database High Availability Overview and Best Practices, 19c.* https://docs.oracle.com/en/database/oracle/oracle-database/19/haovw/index.html.

[7] Oracle Corporation. 2021. *Oracle Cloud Infrastructure Compute Shapes.* https://docs.cloud.oracle.com/en-us/iaas/Content/Compute/References/computeshapes.htm.

[8] Oracle Corporation. 2021. *Oracle Enterprise Manager.* https://www.oracle.com/uk/enterprise-manager/technologies/.

[9] Oracle Corporation. 2021. Oracle Exadata Database Machine. *datasheet* (2021). https://www.oracle.com/database/technologies/exadata.html.

[10] Standard Performance Evaluation Corporation. 2017. *SPEC, CPU2017 Integer Result Intel Xeon Platinum 8168, 2.70 GHz.* https://www.spec.org/cpu2017/results/res2018q4/cpu2017-20181211-10244.html.

[11] Transaction Processing Performance Council. 2021. *Transaction Processing Performance Council.* http://www.tpc.org/information/benchmarks5.asp.

[12] S. K Doddavula, M Kaushik, and A Jain. 2011. Implementation of a Fast Vector Packing Algorithm and its Application for Server Consolidation. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science*. IEEE, 332–339.

[13] Michael R. Garey and David S. Jounson. 1979. *Computers and Intractability: A Guide to the theory of NP-Completeness.* Freeman.

[14] Dominic Giles. 2019. *SwingBench 2.2 Reference and User Guide.* http://www.dominicgiles.com/swingbench.html.

[15] S. Halfpap and R. Schlosser. 2019. Workload-Driven Fragment Allocation for Partially Replicated Databases Using Linear Programming. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1746–1749. https://doi.org/10.1109/ICDE.2019.00188

[16] Abdul Hameed, Abdul Hameed, Alireza Khoshkbarforoushha, Alireza Khoshkbarforoushha, Rajiv Ranjan, Rajiv Ranjan, Prem Prakash Jayaraman, Prem Prakash Jayaraman, Joanna Kolodziej, Joanna Kolodziej, Pavan Balaji, Pavan Balaji, Sherali Zeadally, Sherali Zeadally, Qutaibah Marwan Malluhi, Qutaibah Marwan Malluhi, Nikos Tziritas, Nikos Tziritas, Abhinav Vishnu, Abhinav Vishnu, Samee U Khan, Samee U Khan, Albert Zomaya, and Albert Zomaya. 2016. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing* 98, 7 (2016), 751–774.

[17] Antony S. Higginson, Norman Paton, Suzanne Embury, and Clive Bostock. 2017. DBaaS Cloud Capacity Planning: Accounting for Dynamic RDBMS Systems that Employ Clustering and Standby Architectures. In *: Proceedings of the 20th International Conference on Extending Database Technology.* https://doi.org/10.5441/002/edbt.2017.89

[18] Antony S. Higginson, Mihaela Dediu, Octavian Arsene, Norman W. Paton, and Suzanne M. Embury. 2020. Database Workload Capacity Planning Using Time Series Analysis and Machine Learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 769–783. https://doi.org/10.1145/3318464.3386140

[19] Bernhard Korte and Jens Vygen. 2006. Bin-Packing. In *Combinatorial Optimization: Theory and Algorithms. Algorithms and Combinatorics 21*. Springer, 426.

[20] Mohammad Masdari and Afsane Khoshnevis. 2019. A survey and classification of the workload forecasting methods in cloud computing. *Cluster computing* 23, 4 (2019), 2399–2424.

[21] Rathijit Sen and Karthik Ramachandra. 2018. Characterizing Resource Sensitivity of Database Workloads. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 657–669.

[22] Sukhpal Singh and Inderveer Chana. 2016. Cloud resource provisioning: survey, status and future research directions. *Knowledge and information systems* 49, 3 (2016), 1005–1069.

[23] Zhuoyao Wang, Majeed M Hayat, Nasir Ghani, and Khaled B Shaban. 2017. Optimizing Cloud-Service Performance: Efficient Resource Provisioning via Optimal Workload Allocation. *IEEE transactions on parallel and distributed systems* 28, 6 (2017), 1689–1702.

[24] Tao Yu, Jie Qiu, B Reinwald, Lei Zhi, Qirong Wang, and Ning Wang. 2012. Intelligent Database Placement in Cloud Environment. In *2012 IEEE 19th International Conference on Web Services*. IEEE, 544–551.

[25] Mingyi Zhang, Patrick Martin, Wendy Powley, and Jianjun Chen. 2018. Workload Management in Database Management Systems: A Taxonomy. *IEEE transactions on knowledge and data engineering* 30, 7 (2018), 1386–1402.

Chapter 6

Concluding Remarks

Boss: I just heard that light travels faster than sound. I'm wondering if I should shout when I speak, just so my lips appear to sync-up with my words.
Dilbert (thought): A little knowledge can be a ridiculous thing.

Scott Adams (Creator of Dilbert)

This chapter provides the overall conclusions of the thesis, outlines the most notable limitations of the work undertaken and discusses future work.

6.1 Conclusions

On paper, Capacity Planning seems a simple exercise - obtain and compare the resource estimates against the available or future resource demand. In practice, however, it is perhaps, considerably, a more challenging discipline to perform regardless of industry as Bakke and Hellberg [BH93] suggested as far back as 1993 in their study on the challenges of capacity planning. Resource Capacity Planning, when broken down, involves a deeper understanding of:

- The topology of technology layers that make-up a wider *system* and its architecture (N-Tier or cloud) either on-premises, remote or hybrid.
- Monitoring the appropriate attributes of the *system* at the correct frequency for

periods long enough to perform meaningful data analysis.

- Using the correct forecasting techniques and applicable models capable of performing accurate forecasts on the data, taking into consideration complex data patterns in their signals.
- Place the workloads together in target environments (Cloud) in such a manner as to maximise efficiency without overloading the nodes to starve other workloads or reducing any SLA, SLOs or QoS the system is legislated to adhere to.

In the absence of techniques, expertise and tooling available to perform capacity planning in an efficient and accurate manner, we broke the problem down into three main areas of work relating to database workloads, however, with an *eye* on the application of this work for other types of workload, such as *applications* (SaaS). We noticed, early on that a workload is the *consumption* of resources over a *period* of time that satisfies a set of tasks. Therefore a workload can be taken from any layer or component within a system where resources are consumed, such as applications, storage or a network switch.

DBaaS Cloud Capacity Planning - Accounting: Database systems are one of the most critical elements of many I.T. system as they communicate information related to the business readily. If the database is unavailable, arguably, the business, in whatever shape, rapidly diminishes or ceases to exist. Data, for example, in the form of customers, transactions, items and the criticality of decisions based on said data is a tangible asset in the value of a business. Therefore database management systems have grown in their complexity, employing a myriad of features to ensure they are running at their most optimum and are truly 24*7, 365 days per year with stringent SLAs, SLOs and QoS assigned to them. Migration, Upgrades and re-platforming of complex critical database architectures are decisions, C-level executives do not take lightly, and that require complete focus to ensure success.

The first piece of work as described in paper titled *"DBaaS Cloud Capacity Planning - Accounting for Dynamic RDBMS System that Employ Clustering and Standby Architectures"*, was understanding the make-up of a workload. *How* we account for complex database architectures, *what* metrics should be captured, *where* in the technology stack those metrics should be obtained and under *what* conditions do workloads change? The challenge was examined through the availability of benchmarks, therefore SPECInt and TPC were the logical choices to adhere to. The metrics chosen had

to map to metrics in a cloud environment, therefore *CPU per sec* can be interpreted to SPECInts, which is uniform regardless of architecture (on-premises N-Tier or Cloud), *Physical reads/writes* are the metric used by TPC and are also mappable to storage metrics of cloud environments as throughput, and Memory is RAM. Using an intelligent agent to execute and gather commands is advantageous as it stores the data centrally in a repository as opposed to manually extracting metric data from the individual internal database AWR repositories, however if no intelligent agents are available then one has no choice but to extract data this way.

Our work showed several things that provided a useful foundation for further work. Most recently, research has started to look at the SaaS layer specifically, as described by Stauffer *et al* 2021 [SMS21]. In their study they highlighted that cloud capacity planning tends to focus on the initial placement of a workload into/or a VM at an IaaS level. Their primary focus centred on elasticity management with the aim of adjusting capacity of instances of SaaS applications. They also highlight that key capacity metrics are to be obtained of the SaaS workload such as CPU, memory and disk storage. In their solution they use these metrics specifically to determine by *how* much the capacity should be increased by the consumption of the SaaS query. As we have also concluded, metrics need to be obtained at the right layer, for example at the database instance not the VM, especially if there are multiple instances running in a consolidated environment. Metrics obtained at the VM layer masks where the metric is utilised between multiple databases, therefore it is inappropriate for capacity planning to take place at a VM level without avoiding the inclusion of metrics consumed by all databases. Hypervisors mask what is assigned and what is actually provisioned, as some of the CPU is directed as support overhead and this needs to be accounted for. Operating system configurations have a profound affect on how a workload is executed, for example metrics such as *Thread(s) per core* improve the efficiency of an OLTP workload and thus OS configuration of CPU metrics needs to be accounted for. Advanced database features such as standby databases do not work as an active open running database. Standby databases are in a *recovery* mode applying activity (Redologs) from the primary and thus some key database metrics will not be available for extraction, resulting in metrics being obtained at the OS layer. In a clustered database environment, metrics should be obtained at the instance level of each node, as nodes can run unevenly with connections being routed through one node above another. Also, in the event of clustered node failover a workload should not assume that the properties of the workload on one node will follow obviously, it will assume a new *'consolidated'* workload resulting a

new database instance resource footprint. Arguably, it can be said given the current gaps, that analysis is taking place at the VM (IaaS) because it is so difficult to extract the key data from individual workloads.

One of the aims of this thesis, as described in Section 1.9.1, was to identify the current database footprint including any advanced features such as Standby, clustering and Pluggable databases. Identify any understandings that are required and the key metrics needed to assist that understanding. By performing experiments on a plethora of database configurations we have contributed to this aim by listing the key metrics that make up a workload, and under what conditions do those workloads change that can influence the accountability of metrics with a view to performing a capacity planning exercise.

Database Workload Capacity Planning using Time Series Analysis and Machine Learning: Building on the first piece of work, a critical function of any capacity planning exercise is the ability to form a prediction of future resource consumption. We extracted metric data stored from a central repository and prior to a forecast, one must understand the nuances and traits exhibited in the data signal. Upon analysis we noted from our first piece of work that particular metrics (IOS and CPU) are more granular, exhibiting more pounced peaks and troughs than other metrics (memory), which seem more static when displayed over a time series. We also observed from our first piece of work that depending upon the workload type (OLTP, OLAP and Data Mart), different nuances and intricacies where exhibited in the data signal such as reoccurring patterns reflecting surges in users logging on, or resource consumption growth as the data set gets bigger. A simple Ordinary Least Square Regression (OLS) algorithm would be insufficient, and more complex forecasting models were required. The only techniques available that could accommodate complex data patterns was time series analysis and specifically ARIMA, TBAT type models. However, these complex models require a deep understanding of the data via additional techniques such as Autocorrelation plots to determine the stationariness and trend of the data. Not only did this inflate the expertise need to execute and understand these models but it often influenced the parameters that make up a time series model. This presented us with a conundrum of how to filter the models that are more likely to be more accurate than others reducing the overall number. It became apparent that for one metric running on one clustered 8 node instance an issue of scale arose. Also, that the time to crunch a particular model was inordinately long. We looked to Machine Learning for a solution

and a higher level of automation in removing the understanding needed to correlate the models of the data from the ACF/PACF mechanism.

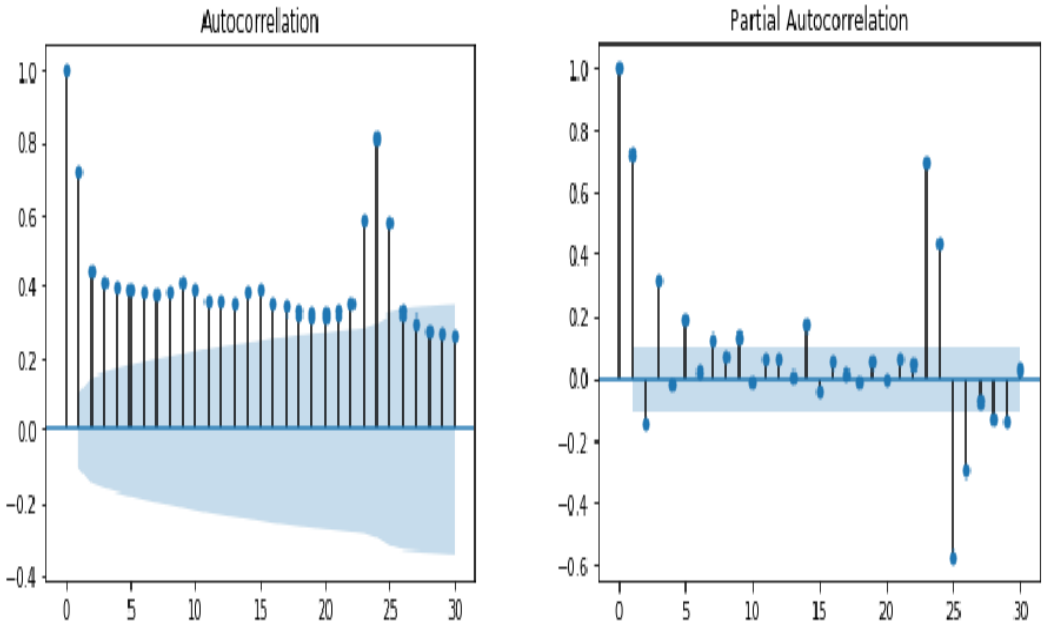


Figure 6.1: Correlograms (ACF/PACF)

Our contributions were several fold, if we remind ourselves from the objectives as described in Section 1.9.1, which was to perform short and long term capacity planning via a *forecast* of what the future resource consumption will be. We introduced and empirically evaluated multiple time series models and concluded that given the complex data patterns exhibited in volatile metric data, such as IO and CPU, and given that systems perform a myriad of tasks that can be influence externally via batch jobs or backups, time series analysis and SARIMAX is a suitable approach to perform a forecast. We chose this approach because there are very few random variables in the data otherwise a stochastic approach could of been more favourable. We also concluded that by adding a Fourier Transform we can handle the most complex data patterns identified in our study of real-life workloads, which consisted of over-lapping or multiple seasonality. We also automated the Autocorrelation function (ACF) and Partial Autocorrelation Function (PACF) to filter the number of SARIMAX models by predetermining the models that are an intersection of the '*dotted line*', resulting in great reduction in the number of models that needed to be processed. Consider Figure 6.1, which was discussed in the paper titled *"Database Workload Capacity Planning using*

Time Series Analysis and Machine Learning". This correlogram, shows a visualisation of both ACF and PACF of a time series data. We can see the number of models from 0 - 30 and that there is a shaded area. Where the shaded area best *intersects* with the models highlights the *p* and *P* of the SARIMAX group of models that are more likely to be accurate. In this example it would be model 26 (ACF) or models 3,8,..,29 for PACF.

Reducing the *'crunch'* time was achieved by leveraging supervised machine learning to learn the signal and after a period of 7 days or the Root Mean Squared Error was reduced to an unacceptable level; prompting the model to be reevaluated reduced the computational requirements, satisfying the *scale* question. To compute RMSE, we calculate the residual (difference between prediction and truth) for each data point, computing the norm of residual for each data point, compute the mean of residuals, and then take the square root of that mean. RMSE is commonly used in supervised learning applications, as RMSE uses and needs true measurements at each predicted data point. In machine learning, it is extremely helpful to have a single number to judge a model's performance, which we show in our results section within the paper. Root mean square error is one of the most widely used measures for producing a singular scoring number. It is a proper scoring rule that is intuitive to understand. One drawback of Mean Application Percentage Error (MAPE) is that forecasts that involve high numbers, which is routine when dealing with IO counts there is no upper limit, hence values greater than 100% are seen in the result table. One technique to counter this would be to use Mean Absolute Scaled Error (MASE), Symmetric Mean Absolute Percentage Error (sMAPE) or Mean Directional Accuracy (MDA).

This work resulted in a patent being filed by the Oracle Corporation for this work. It is currently road mapped to be introduced into the Oracle ACS Pulse Software. We also noticed that we have changed a user's experience of how they consume charts of metric information. Typically, when a user is presented with metric consumption over a time series format, the user is presented with *historical* data and the *present*, often prompting the user to formulate the *future* on their own. By providing the user with a prediction of the future we *advise* the user on the criticality of what is being shown. For example, if the resources to be consumed are predicted to result in a potential outage in the next 24 hours causing it to be flagged as critical, potentially feeding into anomaly detection routines.

Most recently, research has focused on using time series analysis as a prediction technique in federated clouds as described by Keshavarzi *et al* 2021 [KHB21]. In this

research they focus on the QoS needs of the user in multi-clouds and using a hybrid approach utilising ARIMA models in conjunction with minimum description length (MDL) algorithms to identify patterns in the QoS data. The ultimate goal is to predict resource consumption to determine the best QoS of a given SLA in multi-clouds. They do not focus on any layer of the cloud such as IaaS, PaaS, DBaaS or SaaS, instead opting to focus on the prediction technique. Kwilinksi *et al* 2021 [KLK$^+$21] also looked at time series analysis for prediction in the digital economy such as cryptocurrency, highlighting that predictions techniques such as ARIMA type models is valid in other industries other than Cloud Computing.

Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure: Obtaining a forecast of future consumption leads into workload placement for cloud architectures. Firstly, a vector of metrics are selected from a template, which the algorithm uses as parameters to place. Placement of all metrics must take place otherwise a *failure* to place is given. We took this approach because if the user wishes to place on a particular metric then it would suggest that it is of importance to the system or the application is particularity sensitive, for example *elapsed time* based metrics. Also, allowing the algorithm to place on user defined metrics allows the algorithm to scale for other parts of the cloud infrastructure, such as network *throughput* metrics. Secondly, an extraction of the *max_values* from the prediction, if one is performed, for each workload that has been selected as a candidate for placement. Max_values are obtained over average_values because if a system hits 100% utilised then it will panic. An average is a *middle* number in a set of numbers and is often lower than a max_value. The max_value is taken from a trace of metric data from the central repository which stores the data from the agent. We observed that, currently, bin-packing routines treat workloads as singular workloads as do vector bin-packing therefore an extension is required to place complex workloads obtained from clustered, standby or consolidated pluggable databases, which we provide and evaluate. We also have a requirement that advanced databases are complex, therefore by their very nature suffer from bottlenecks at various points, for example. We needed to utilise a vector approach and be able to place on multiple dimensions of metrics. Thus we extended the mathematic notation to include *'metrics'* as well as *instances*.

The problem to solve was multi-faceted, that is to say all parts of the problem needed to be tackled at the same time, rather than one individual piece; how to place complex database architectures into complex cloud architectures? A standby database

is always a single instance even if it is a clustered standby because it applies transaction logs, in the form of database redologs, from all nodes in the primary cluster. The difficulty seemed to be focused on placement of pluggable databases that are of a consolidated nature of many smaller, isolated databases within a container or *umbrella* instance. Treating a pluggable database in the same manner as a single instance allowed us to place it more easily. Because a pluggable can be singular database within a container instance or reside with other *sibling* databases. It can also be a Clustered Pluggable Database shared across multiple nodes adding further complexity. Treating a pluggable and standby database as single instances, reduced the complexity of the algorithm. Clustered databases also need to be treated as one entity yet all the instances in the cluster are treated separately; a cluster can run multiple clustered databases too, creating a *many-to-many* relationship. We treated clustered databases running on a node as a separate instance therefore the term instance acts like a logical *normalisation* key to remove the *many-to-many* relationships.

There was also one final complexity, which the issue of SLAs, SLOs and QoS enforced by High Availability configurations. We noted that current bin-packing solutions, because, they treat workloads in a singular fashion. Can give rise to a scenario where placement of all nodes in a cluster can be assigned to one target node as long as there is available resources. This effectively removes the QoS, SLA or SLO by placing all source clustered instances into one target cloud node. Therefore, we maintain High Availability by ensuring, through the algorithm, two things.

- The target cloud architecture must have the same as or more nodes than the source configuration.

- All workloads must be placed or no workloads are placed thus a rollback function is enforced.

During our experiments we found curious behaviour, causing us to extend the First-Fit Decreasing Algorithm further and this was attributed to ordering the workloads prior to placement, something we did not expect. The result was to insert an ordering routine (steps 1 and 3) of algorithms 1 and 2 listed in Section 5.1 of the paper titled *"Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure"*. This ordering routine ordered the instances in descending order with the largest workload being the first to be placed while also ordering the largest available resource target nodes being first. Therefore it is critical to order on the cluster and its

siblings in descending order. This novel approach to workload placement of complex workloads has resulted in another patent being filed by the Oracle Corporation.

Our final aim of the thesis from Section 1.9.1 was to place the workloads in a target cloud configuration without compromising SLAs, SLOs or QoS. Could the mechanism chosen, exercise high automation reducing the manual level of effort (LoE) currently experienced by technical professionals. Could the bin-packing routines work on advanced complex database architectures that are currently in play today, something that was achieved in our experiments and testing of our use cases. Recently bin-packing routines have been used in emerging technologies such as Internet Of Things (IOT) as described by Tyagi *et al* [TMBJ21]. In this research they looked to a Modified First-Fit decreasing (MFFD) to deal with network traffic throughput of a Cloud Radio Access Network (CRAN) from smart meters attached to households and businesses. The aim of the FFD algorithm is to place items based on meeting a deadline which they achieve with favourable results. Sridharan and Domnic [SD21] looked to an evaluation of several bin-packing algorithms such as First-Fit Decreasing and First-Fit Increasing to address a problem of elasticated VMs created in IaaSs of Data Centres. They developed a Policy and Elastic Aware Placement (PEAP) algorithm that is an extension of existing scheduling policies. As VM requests are received into the Data Centre they are labelled as new, migrating and incremental VM requests of elasticity each with a vector of resource. Utilising their PEAP algorithm, they influence the processing of requests by assigning a cost to the request, and then placement of the request takes place. When assessed against FFI and FFD they found that their PEAP was more optimal when it comes to processing VM vector requests in Data Centres, adding weight to our argument that heuristic placement algorithms such as FFD are too simplistic when dealing with vectors in complex cloud architectures.

Measurements where not taken on the performance of the algorithms themselves. This thesis does not evaluate the performance of the algorithms, in terms of speed, being executed. In the second paper we used a supervising learning algorithm to help reduce the execution time in producing the prediction. That is to say, being able to perform predictions on many metrics for customers, and their systems at the same time (scale). The algorithms produced, where new and required protection through legal patents thus were unique in solving the problem, and not how efficient it is in solving the problem. However, that could be future work to improve the efficiency of the algorithms to work at their most optimum.

This thesis does not claim that the challenges to capacity planning advanced database

architectures for cloud adoption are complete. It identifies gaps and weaknesses in existing techniques currently in practice, when performing a capacity plan exercise on complex database architectures. The thesis highlights issues such as, what are the key metrics that make up a workload, which metrics are mappable from on-premises to cloud architectures, and how workloads can be influenced depending on their system configuration, which are dependant upon the tasks these systems are asked to perform. The thesis identifies the challenges in forecasting the future resource consumption, short term and/or long term, of the workloads that exhibit complex patterns, which can be subtle but important. Finally, placement of complex workloads from advanced database architectures into complex target cloud architectures is an NP-Hard and NP-Complete problem that existing algorithms require significant extension or new algorithms to solve. This thesis potentially gives informed insight and tackles these issues on real-world configurations, for which there is no standard practice available. This thesis has shown several novel approaches such as forecasting and workload placement of which research shows is still a problem with academics utilising several approaches with varying combinations of techniques to address. As clouds become ever more complicated and widespread, as this thesis addresses, historic simplistic approaches are not applicable and new thinking is required, which we have gone some way in pursuing.

6.2 Limitations and Future Work

Some of this work (Forecasting and Workload Placement) has been placed on the roadmap of future Oracle Products, such as Oracle Pulse for the Oracle Managed Cloud Services portfolio that Oracle Advanced Customer Services offers their customers via the Oracle Pulse product. [Cor21j]. The work on creating, configuring and controlling workloads is used for testing monitoring software development in the Engineering function of Advanced Customer Services. Testing consumption based charting is difficult because it requires *load* to be placed on the database. By creating a repeatable, controllable and observable set of workloads we can properly test and populate graphs as workloads can be set off and increased/decreased by adding users to create spikes under a controlled environment, therefore testing the resource consumption charts is now much easier from this work. The Workload Placement algorithms are currently being evaluated with development teams within Oracle

Our approach to forecasting does have some challenges, such as when a system is

unstable or in a period of fault, for example, due to frequent crashes, as the Learning Engine then relearns to adopt those new behaviours. In our algorithm we account for this by suggesting that the event needs to happen more then 3 times for it to be a behaviour, which can be changed manually. It is perfectly plausible that the system fails over to a new site to test disaster recovery. Therefore if a system crashes we discard it, however if the system continually crashes the learning engine will see it as a behaviour and account for it in its forecast. Live systems rarely continually crash but they do crash, therefore manual override is needed to accommodate systems that are in-fault as we suggest that forecasting will not be a true reflection of the system when stable. There is still a need for threshold based monitoring. Utilising these techniques to predict when a threshold is likely to be breached is an advisable way to implement this approach for proactive monitoring and capacity based questions; we don't see our approach as a complete replacement for thresholds just yet.

Our approach to workload placement is also not without its challenges, especially when placing siblings of clusters, for which we implemented a work-around of ordering the workloads and their siblings in descending order. A solution to this problem could be to take the cumulative approach of the total amount of resources consumed per cluster and order based on number of nodes resources consumed. furthermore enabling a user defined priority assignment function would be advantageous. Assigning a user defined priority to a workload allows for separation between live systems, as some systems may be more critical than others. In our earlier work we leveraged Machine Learning (Supervised) coupled with Time Series Analysis to predict future resource consumption of a workload and we see a combination of those techniques and placement to create a more informed choice when one is making decisions on what systems can be placed based on their future resource consumption. Extending the approach to other algorithms such as best or next-fit seem a logical progression path including a search or optimisation strategy as we only focused on simple heuristic first-fit algorithms.

The work in this thesis was used on Oracle technology. It was not produced on Microsoft, Amazon, IBM or a hybrid combination of vendors. Arguably, one could question if the prediction or placement algorithms would work in agnostic environments rather than homogeneous to Oracle. However, the prediction utilises Python libraries on time series data. The data is extracted using MAPE [ARS15] agent technology, which is stored in a central repository. The placement algorithms also utilise Python too. Thus we are confident that our work is vendor agnostic even if it was not

tested on hybrid vendor cloud configurations. Future work could be to evaluate the algorithms on hybrid vendor and cloud configurations.

Bibliography

[AbMl20] Nurşen Aydın, İbrahim Muter, and Ş. İlker Birbil. Multi-objective temporal bin packing problem: An application in cloud computing. *Computers & Operations Research*, 121:104959, 2020.

[ACKS13] Yossi Azar, Ilan Reuven Cohen, Seny Kamara, and Bruce Shepherd. Tight bounds for online vector bin packing. STOC '13, page 961–970, New York, NY, USA, 2013. Association for Computing Machinery.

[AETEK13] Ahmed Ali-Eldin, Johan Tordsson, Erik Elmroth, and Maria Kihl. Workload classification for efficient auto-scaling of cloud resources. *Department of Computer Science, Umea University, Umea, Sweden, Tech. Rep*, 2013.

[AKY10] Artur Andrzejak, Derrick Kondo, and Sangho Yi. Decision model for cloud computing under sla constraints. In *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 257–266, 2010.

[ARS15] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. Modeling and analyzing mape-k feedback loops for self-adaptation. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 13–23, 2015.

[AWS21] Inc. Amazon Web Services. *AWS Cloud Databases*, 2021.

[Bay77] Carter Bays. A comparison of next-fit, first-fit, and best-fit. *Commun. ACM*, 20(3):191–192, mar 1977.

[BBB+17] Djillali Boukhelef, Kamel Boukhalfa, Jalil Boukhobza, Hamza Ouarnoughi, and Laurent Lemarchand. Cops: Cost based object placement strategies on hybrid storage system for dbaas cloud. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 659–664, 2017.

[Ber05] L.L Beranek. Bbn's earliest days: founding a culture of engineering creativity. *IEEE annals of the history of computing*, 27(2):6–14, 2005.

[BG14] Bhavani and Guruprasad. Resource provisioning techniques in cloud computing environment: A survey. *IJRCCT International Journal of Research in Computer and Communication Technology*, 3(7):395 – 401, 2014.

[BH93] Nils Arne Bakke and Roland Hellberg. The challenges of capacity planning. *International Journal of Production Economics*, 30-31:243–264, 1993. Special Issue Proceeding of the Seventh International Working Seminar on Production Economics.

[BJRL15] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[CBL+17] Marcus Carvalho, Francisco Brasileiro, Raquel Lopes, Giovanni Farias, Alessandro Fook, João Mafra, and Daniel Turull. Multi-dimensional admission control and capacity planning for iaas clouds with multiple service classes. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 160–169, 2017.

[CMRB14] Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications' qos. *IEEE transactions on cloud computing*, 3(4):449–458, 2014.

[Com17] Creative Commons. *Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0)*, 2017.

[Cor13] Oracle Corporation. *Database Sample Schemas*, 2013.

[Cor17a] Oracle Corporation. *Automatic Database Performance Monitoring*, 2017. https://docs.oracle.com/database/121/TDPPT/tdppt_auto.htm#TDPPT020.

[Cor17b] Oracle Corporation. *Real Application Clusters Administration and Deployment Guide, Automatiac Workload Repository*, 2017. https://docs.oracle.com/database/121/RACAD/GUID-C3CD2DCE-38BD-46BA-BC32-7A28CAC9A7FD.htm#RACAD951.

[Cor17c] Standard Performance Evaluation Corporation. *SPEC, CPU2017 Integer Result Intel Xeon Platinum 8168, 2.70 GHz*, 2017. https://www.spec.org/cpu2017/results/res2018q4/cpu2017-20181211-10244.html.

[Cor21a] Oracle Corporation. *Automatic Storage Management Administrator's Guide (ASM)*, 2021.

[Cor21b] Oracle Corporation. *Data Guard Concepts and Administration*, 2021.

[Cor21c] Oracle Corporation. *Enterprise Manager Cloud Control 12c Architecture*, 2021. https://docs.oracle.com/cd/E24628_01/doc.121/e25353/overview.htm#EMCON112.

[Cor21d] Oracle Corporation. *Exadata Database Machine Documentation*, 2021.

[Cor21e] Oracle Corporation. *Oracle Cloud Infrastructure for Amazon Web Services professionals*, 2021. https://docs.oracle.com/en/solutions/oci-for-aws-professionals/index.html#GUID-CD0E016D-AC8A-4A18-BE78-0FD7A3E1C64E.

[Cor21f] Oracle Corporation. *Oracle Cloud Infrastructure for Microsoft Azure professionals*, 2021. https://docs.oracle.com/en/solutions/oci-for-azure-professionals/index.html#GUID-1AB4955B-CBF3-437F-B0EE-A50B0F9EEF37.

[Cor21g] Oracle Corporation. *Oracle Cloud Service Mapping*, 2021. https://www.oracle.com/cloud/iaas/cloud-service-mapping.html.

[Cor21h] Oracle Corporation. *Oracle Enterprise Manager Cloud Control Documentation 12c Release 5*, 2021. https://docs.oracle.com/cd/E24628_01/index.htm.

[Cor21i] Oracle Corporation. *Oracle GoldenGate*, 2021. https://www.oracle.com/uk/integration/goldengate/.

[Cor21j] Oracle Corporation. *Oracle Pulse*, 2021.

[Cor21k] Oracle Corporation. *Oracle Real Application Clusters*, 2021.

[Cor21l] Oracle Corporation. *Oracle Recovery Manager (RMAN)*, 2021.

[Cor21m] Oracle Corporation. *Overview of the Database Service*, 2021.

[Cou21] Transaction Processing Performance Council. *Transaction Processing Performance Council*, 2021. http://www.tpc.org/information/benchmarks5.asp.

[CST+10] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, page 143–154, New York, NY, USA, 2010. Association for Computing Machinery.

[CWW14] Victor Chang, Robert John Walters, and Gary Wills. Review of cloud computing and existing frameworks for cloud adoption. In *Advances in Cloud Computing Research*. NOVA Publishers, March 2014.

[Day16] Matt Day. *How Microsoft emerged from the darkness to embrace the cloud*, December 12, 2016.

[DFZ+15] Yucong Duan, Guohua Fu, Nianjun Zhou, Xiaobing Sun, Nanjangud C. Narendra, and Bo Hu. Everything as a service (xaas) on the cloud: Origins, current and future trends. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 621–628, 2015.

[Dix12] Prashant Dixit. *ADDR & AWR Detailed Study (10g)*, 2012.

[DKJ11] Shyam Kumar Doddavula, Mudit Kaushik, and Akansha Jain. Implementation of a fast vector packing algorithm and its application for server consolidation. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, pages 332–339, 2011.

[DL21] ACM DL. *Association for Computing Machinery - Digital Library*, November, 2021. `https://dl.acm.org/`.

[DLH19] Yingnong Dang, Qingwei Lin, and Peng Huang. Aiops: Real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 4–5, 2019.

[FAK+12] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Clearing the clouds: A study of emerging scale-out workloads on modern hardware. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, page 37–48, New York, NY, USA, 2012. Association for Computing Machinery.

[FFPB21] Edoardo Fadda, Stanislav Fedorov, Guido Perboli, and Ivan Dario Cardenas Barbosa. Mixing machine learning and optimization for the tactical capacity planning in last-mile delivery. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1291–1296. IEEE, 2021.

[Fou21] Python Software Foundation. *What is Python? Executive Summary*, 2001 - 2021.

[Gar19] Gartner. *Everything you need to know about AIOp*, 2019. `https://www.moogsoft.com/resources/aiops/guide/everything-aiops/`.

[GDR15] Hemlata Gangwar, Hema Date, and R Ramaswamy. Developing a cloud-computing adoption framework. *Global Business Review*, 16(4):632–651, 2015.

[Gil10] Dominic Giles. *Swingbench*, 2010.

[GKD+09] Archana Ganapathi, Harumi Kuno, Umeshwar Dayal, Janet L. Wiener, Armando Fox, Michael Jordan, and David Patterson. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In *2009 IEEE 25th International Conference on Data Engineering*, pages 592–603, 2009.

[GLX+13] Rahul Ghosh, Francesco Longo, Ruofan Xia, Vijay K Naik, and Kishor S Trivedi. Stochastic model driven capacity planning for an infrastructure-as-a-service cloud. *IEEE Transactions on Services Computing*, 7(4):667–680, 2013.

[GLX+14] Rahul Ghosh, Francesco Longo, Ruofan Xia, Vijay K. Naik, and Kishor S. Trivedi. Stochastic model driven capacity planning for an infrastructure-as-a-service cloud. *IEEE Transactions on Services Computing*, 7(4):667–680, 2014.

[Goo21a] Google. *Google Cloud databases*, 2021.

[Goo21b] Google. *Google Cloud Overview*, 2021. `https://cloud.google.com/docs/overview`.

[GRCK07] Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. Capacity management and demand prediction for next generation data centers. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 43–50. IEEE, 2007.

[Gro11] Data Management Research Group. Data management research at nec labs. *SIGMOD Rec.*, 40(3):38–44, nov 2011.

[HBS+16] Kai Hwang, Xiaoying Bai, Yue Shi, Muyang Li, Wen-Guang Chen, and Yongwei Wu. Cloud performance modeling with benchmark evaluation of elastic scaling strategies. *IEEE Transactions on Parallel and Distributed Systems*, 27(1):130–143, 2016.

[HHD+10] Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang. The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, pages 41–51. IEEE, 2010.

[HIM02] H. Hacigumus, B. Iyer, and S. Mehrotra. Providing database as a service. In *Proceedings 18th International Conference on Data Engineering*, pages 29–38, 2002.

[HKR+16] Abdul Hameed, Alireza Khoshkbarforoushha, Prem Prakash Ranjan, Jayaraman, Joanna Kolodziej, Pavan Balaji, Sherali Zeadally, Qutaibah Marwan Malluhi, Nikos Tziritas, Abhinav Vishnu, Samee U Khan, and Albert Zomaya. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*, 98(7):751–774, 2016.

[HS19] Stefan Halfpap and Rainer Schlosser. Workload-driven fragment allocation for partially replicated databases using linear programming. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1746–1749, 2019.

[HY10] R. Harms and M. Yamartino. The economics of the cloud. November, 2010.

[KCK+13] Stephan Kraft, Giuliano Casale, Diwakar Krishnamurthy, Des Greer, and Peter Kilpatrick. Performance models of storage contention in cloud environments. *Software & Systems Modeling*, 12(4):681–704, 2013.

[KDGR15] Magdalena Kostoska, Aleksandar Donevski, Marjan Gusev, and Sasko Ristov. Porting an n-tier application on cloud using p-tosca: A case study. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 281–285, 2015.

[KGJ+21] Xiaodi Ke, Cong Guo, Siqi Ji, Shane Bergsma, Zhenhua Hu, and Lei Guo. Fundy: A scalable and extensible resource manager for

cloud resources. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 540–550, 2021.

[KHB21] Amin Keshavarzi, Abolfazl Toroghi Haghighat, and Mahdi Bohlouli. Online qos prediction in the cloud environments using hybrid time-series data mining approach. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, 45(2):461–478, 2021.

[KHGSS12] Ali Khajeh-Hosseini, David Greenwood, James W. Smith, and Ian Sommerville. The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. *Software: Practice and Experience*, 42(4):447–465, 2012.

[Kit03] C.I Kita. J.c.r. licklider's vision for the ipto. *IEEE annals of the history of computing*, 25(3):62–77, 2003.

[KJA17] Takahiko Kajiyama, Murray Jennex, and Theophilus Addo. To cloud or not to cloud: how risks and threats are affecting cloud adoption decisions. *Information & Computer Security*, 2017.

[KL12] Yousri Kouki and Thomas Ledoux. Sla-driven capacity planning for cloud applications. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pages 135–140. IEEE, 2012.

[KLK+21] Aleksy Kwilinski, Valeriya Litvin, Ekaterina Kamchatova, Julia Polusmiak, and Daria Mironova. Information support of the entrepreneurship model complex with the application of cloud technologies. *International Journal of Entrepreneurship*, 25(1):1–8, 2021.

[Lan08] Arthur M. Langer. *System Development Life Cycle (SDLC)*, pages 10–20. Springer London, London, 2008.

[LEwEZ09] FORA.tv Larry Ellison with Ed Zander. *The Churchill Clud, San Jose, CA - Objection to Cloud Computing*, September 21, 2009. `https://www.youtube.com/watch?v=UOEFXaWHppE`.

[Mar21] Oracle Corporation Brand Marketing. *Oracle Brand Marketing Icons and Graphics*, 2021. https://my.oracle.com/site/mktg/creative/Graphics/Icons/index.html.

[MB13] Rim Moussa and Hassan Badir. Data warehouse systems in the cloud: Rise to the benchmarking challenge. *Int. J. Comput. Their Appl.*, 20(4):245–254, 2013.

[MCM13] Barzan Mozafari, Carlo Curino, and Samuel Madden. Dbseer: Resource and performance prediction for building a next generation database cloud. In *CIDR*, 2013.

[Met21] Resurchify Journal Metrics. *Resurchify - Impact Factor*, November, 2021. https://www.resurchify.com/if/impact-factor-search.

[MG11] "Peter Mell and Tim Grance". The nist definition of cloud computing. SP 800-145, 2011.

[Mic21] Microsoft. *Types of databases on Azure - Fully managed, intelligent and flexible cloud database services*, 2021.

[MK20] Mohammad Masdari and Afsane Khoshnevis. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing*, 23(4):2399–2424, 2020.

[MKB+12] Shruti Mahambre, Purushottam Kulkarni, Umesh Bellur, Girish Chafle, and Deepak Deshpande. Workload characterization for capacity planning and performance management in iaas cloud. In *2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 1–7, 2012.

[MN09] Daniel A. Menasce and Paul Ngo. Understanding cloud computing: Experimentation and capacity planning. Dec, 2009.

[Moh17] Sikender Mohsienuddin Mohammad. Devops automation and agile methodology. volume 5(3):946–949, 2017.

[NXYJ17] Marius Noreikis, Yu Xiao, and Antti Ylä-Jaäiski. Qos-oriented capacity planning for edge computing. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6, 2017.

[oW22] University of Waikato. *AMassive On-line Analysis (MOA)*, 2022.

[PAT+20] Paulo Pereira, Jean Araujo, Matheus Torquato, Jamilson Dantas, Carlos Melo, and Paulo Maciel. Stochastic performance model for web server capacity planning in fog computing. *The Journal of Supercomputing*, 76(12):9533–9557, 2020.

[PPN16] Ognjen Pantelić, Ana Pajić, and Ana Nikolic. Analysis of available cloud computing models to support cloud adoption decision process in an enterprise. In *2016 6th International Conference on Computers Communications and Control (ICCCC)*, pages 135–139, 2016.

[PS21] C Pandiselvi and S Sivakumar. Performance of particle swarm optimization bin packing algorithm for dynamic virtual machine placement for the consolidation of cloud server. In *IOP Conference Series: Materials Science and Engineering*, volume 1110, page 012007. IOP Publishing, 2021.

[Rad05] Shirley Radack. The system development life cycle (sdlc). 27, 2005.

[RGARHCDF20] Merv Adrian Rick Greenwald Adam Ronthal Henry Cook Donald Feinberg. *Magic Quadrant for Cloud Database Management Systems*, 23 Novemeber 2020. `https://www.gartner.com/doc/reprints?id=1-24NITLSK&ct=201125&st=sb`.

[RPAM16] M Rajkumar, Anil Kumar Pole, Vittalraya Shenoy Adige, and Prabal Mahanta. Devops culture and its impact on cloud delivery and software development. In *2016 International Conference on Advances in Computing, Communication, Automation (ICACCA) (Spring)*, pages 1–6, 2016.

[SC16] Sukhpal Singh and Inderveer Chana. Cloud resource provisioning: survey, status and future research directions. *Knowledge and information systems*, 49(3):1005–1069, 2016.

[SD21] R Sridharan and S Domnic. Network policy aware placement of tasks for elastic applications in iaas-cloud environment. *Cluster Computing*, 24(2):1381–1396, 2021.

[Siv18] Anup Sivadas. How to architect a hybrid microsoft sql server solution using distributed availability groups. March, 2018.

[Sko19] Grzegorz Skorupa. *Forecasting Time Series with Multiple Seasonalites using TBATS in Python*, 2019.

[SMCFM18] Flávio RC Sousa, Leonardo O Moreira, José S Costa Filho, and Javam C Machado. Predictive elastic replication for multi-tenant databases in the cloud. *Concurrency and Computation: Practice and Experience*, 30(16):e4437, 2018.

[SMLB14] Soror Sahri, Rim Moussa, Darrell DE Long, and Salima Benbernou. Dbaas-expert: A recommender for the selection of the right cloud database. In *International Symposium on Methodologies for Intelligent Systems*, pages 315–324. Springer, 2014.

[SMS21] Jon M. Stauffer, Aly Megahed, and Chelliah Sriskandarajah. Elasticity management for capacity planning in software as a service cloud computing. *IISE Transactions*, 53(4):407–424, 2021.

[SR18] Rathijit Sen and Karthik Ramachandra. Characterizing resource sensitivity of database workloads. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 657–669, 2018.

[SW13] Navin Sabharwal and Prashant Wali. Cloud capacity management. In *Cloud Capacity Management*, pages 35–54. Springer, 2013.

[SWJ+11] Shifeng Shang, Bo Wang, Jinlei Jiang, Yongwei Wu, and Weimin Zheng. An intelligent capacity planning model for cloud market. *J. Internet Serv. Inf. Secur.*, 1(1):37–45, 2011.

[Til21] Aaron Tilley. *Battle for the Cloud, Once Amazon vs. Microsoft, Now Has Many Fronts*, July 25, 2021.

[TMBJ21] Sumarga Kumar Sah Tyagi, Amrit Mukherjee, Qu Boyang, and Deepak Kumar Jain. Computing resource optimization of big data in optical cloud radio access networked industrial internet of things. *IEEE Transactions on Industrial Informatics*, 17(11):7734–7742, 2021.

[TYP19] Lingling Tang, Yulin Yi, and Yuexing Peng. An ensemble deep learning model for short-term load forecasting based on arima and lstm. In *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6, 2019.

[W+64] Norbert Wiener et al. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*, volume 8. MIT press Cambridge, MA, 1964.

[XLZ+21] Bin Xia, Tao Li, Qifeng Zhou, Qianmu Li, and Hong Zhang. An effective classification-based framework for predicting cloud capacity demand in cloud services. *IEEE Transactions on Services Computing*, 14(4):944–956, 2021.

[YLCL21] Zhengtong Yan, Jiaheng Lu, Naresh Chainani, and Chunbin Lin. Workload-aware performance tuning for autonomous dbmss. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2365–2368. IEEE, 2021.

[YQR+12] Tao Yu, Jie Qiu, Berthold Reinwald, Lei Zhi, Qirong Wang, and Ning Wang. Intelligent database placement in cloud environment. In *2012 IEEE 19th International Conference on Web Services*, pages 544–551, 2012.

[ZMPC18] Mingyi Zhang, Patrick Martin, Wendy Powley, and Jianjun Chen. Workload management in database management systems: A taxonomy. *IEEE Transactions on Knowledge and Data Engineering*, 30(7):1386–1402, 2018.

[ZZS+12] Jianfeng Zhan, Lixin Zhang, Ninghui Sun, Lei Wang, Zhen Jia, and Chunjie Luo. High volume throughput computing: Identifying and characterizing throughput oriented workloads in data centers. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*, pages 1712–1721, 2012.

Appendix A

Licenses of Published Works

This Appendix presents the permission from Publishers for reusing the published papers that this thesis collects

A.1 Permission to Reuse Content from EDBT Publications 2017

Extending Database Technology (EDBT) 2017 Joint Conference was held in March 21-24, 2017 in Venice, Italy. Our work was part of the Industry Track of papers and is available with the following url `http://openproceedings.org/html/pages/2017_edbt.html`. The EDBT 2017 journal is covered by Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License [Com17]. The Creative Commons is a non-profit organisation that helps overcome legal obstacles to the sharing of knowledge and creativity to address the world's challenges. The license can be obtained via the url `https://creativecommons.org/licenses/by-nc-nd/4.0/`, which is also supplied via Figure A.1.

A.2 Permission to Reuse Content from Association of Computing Machinery (ACM) Publications 2020

The Association for Computer Machinery (ACM), provides specific Author Rights when submitting work through the ACM as part of their license agreement. This is available at this url `https://authors.acm.org/author-resources/author-rights`. Under their Reuse policy an Author can reuse their work for any dissertation:

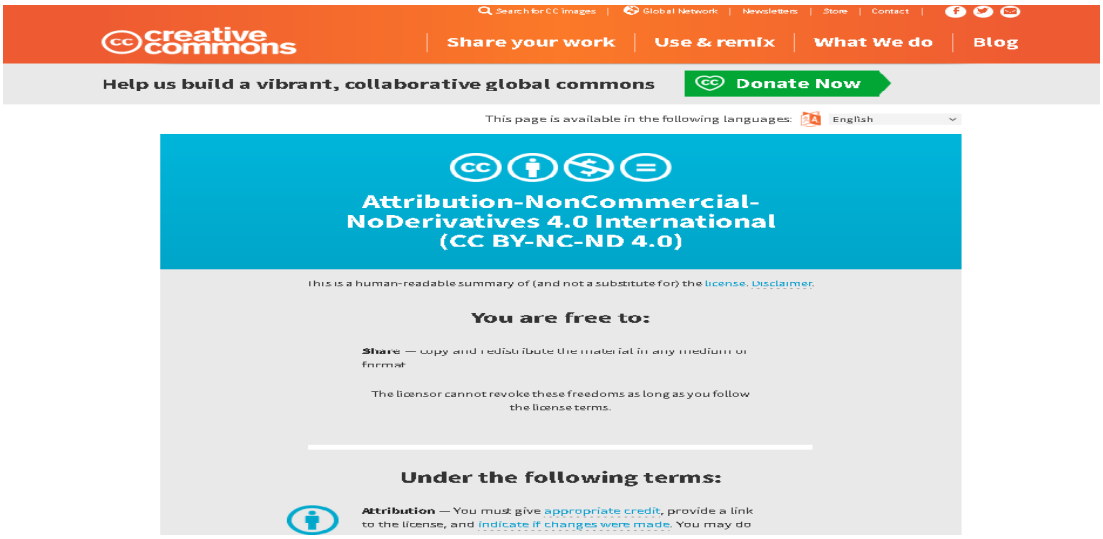


Figure A.1: EDBT 2017 Reuse License

Reuse

Authors can reuse any portion of their own work in a new work of their own (and no fee is expected) as long as a citation and DOI pointer to the Version of Record in the ACM Digital Library are included.

- Contributing complete papers to any edited collection of reprints for which the author is not the editor, requires permission and usually a republication fee.
- Authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected).
- Commercially produced course-packs that are sold to students require permission and possibly a fee.

A.3 Permission to Reuse Content from EDBT Publications 2022

ISBN 978-3-89318-086-8.

April, 2022, ISBN 978-3-89318-086-8 on OpenProceedings.org.

A.4 Permission to use Oracle Content

From: COPYRIGHT_US <copyright_us@oracle.com>
Sent: 05 January 2022 17:46
To: Antony Higginson
Cc: COPYRIGHT_US <copyright_us@oracle.com>
Subject: RE: Permission to use Oracle Content for PhD Thesis

Antony,

Thank you for checking with us. You may reproduce and distribute
the figures below (the "Works") in your thesis titled
"Database Clouds – Accounting, Forecasting and Workload Placement
From Complex RDBMS Architectures"
("the Publication").

Page 16 Figure 1.2 { DevOps
Page 17 Figure 1.3 { Sample OCI Diagram
Page 50 Figure 2.7 { Enterprise Manager Cloud Control 12c Architecture
Page 73 Figure 4.2 { Order Entry Schema Diagram
Page 75 Figure 4.3 { Sales History Schema Diagram

The Works shall appear as originally written without addition,
deletion, editing, or advertising appearing around it unless you obtain
our written approval in advance. Oracle retains all intellectual property
rights in the Works, and you may not do anything inconsistent with such
ownership rights. Oracle does not make any representations as to the
accuracy or completeness of any information contained in the Works or
as to its ownership rights in the Works and is not responsible for any
errors or omissions contained in the Works.

Please indicate that the Works are copyrighted by Oracle and used

with permission. Specifically, include this notice in your thesis when referencing the Works:

"Copyright Oracle and its affiliates. Used with permission."

If you have any questions, please don't hesitate to ask.

Regards,

Mark Schreier
Trademark & Copyright Legal

A.5 Permission to use Gartner® Content

Permission was sent to Gartner via email for use of the Magic Quadrant in Figure 1.1 and obtained via email for the purpose of this thesis. The following guidance was issued:

Gartner does not endorse any vendor, product or service depicted in its research publications and does not advise technology users to select only those vendors with the highest ratings or other designation. Gartner research publications consist of the opinions of Gartner's Research & Advisory organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

Appendix B

The Nuts and Bolts

This Appendix presents main aspects of software used to perform experiments, capture data, crunch data. It also provides snippets of SQL.

B.1 Workloads

The following section will describe in detail, workloads used and how they were configured and executed including the mechanisms to do so.

B.1.1 Swingbench Parameters for Swingbench

To avoid error *'Error occurred during initialization of VM Could not reserve enough space for 2097152KB object heap'* navigate to the *swingbench*
launcher
launcher.xml and change the following xml, however this is dependant upon the available memory on the server where Swingbench is to be executed. On a Server with 16Gb of memory we made this change

```
<jvmargset id="datagenerator.jvm.args">
\textbf{\textit{<jvmarg line="-Xmx2048m"/>}}
<jvmarg line="-Xms512m"/>
</jvmargset>

TO

<jvmargset id="datagenerator.jvm.args">
```

```
<jvmarg line="-Xmx1024m"/>
<jvmarg line="-Xms512m"/>
</jvmargset>
```

B.1.2 Swingbench - Creation Commands

Execute the following as the oracle user on the operating system, ensuring that the environment variables are set using the *'.oraenv'* command.

NOTE: The hostname (acs-163.us.oracle.com), port (1521) and instance (RapidKit) as the dba user (-dbap) this will create the Sales Order Entry (SOE) at an initial size of 10G (-scale 10) and use 64 threads to do so (-64) in a verbose (-v) manner to capture any details of failure errors.

```
./oewizard -cl -create -cs //host:1521/(SID) -dbap pmdev -scale 10 -tc 64 -v
```

```
./shwizard -cl -create -cs //host:1521/(SID) -dbap pmdev -scale 10 -tc 64 -v
```

B.1.3 Swingbench - Execution commands (Crontab Configurations)

Swingbench is initiated via the in-built Linux Scheduler cron

Single Instance OLTP : The following cron entry shows an OLTP entry with spikes and surges at particular times

```
##############################################################################
# OLTP - create a normal 8 hour load to simulate 200 users running on a
database and spikes of 300 users at busy times of the day
# activate this line if you want a OLTP load for 200 user for 8 hours
#
#0 8 * * * /u01/../soe_bench.sh RapidKit -rt 23:55 -uc 200 >> /../daily.log 2>&1
#
# create a spike that for 2 hours simulates a surge in users at
# special times of the day 08:00-10:00, 12:00-13:00 and 17:00-20:00
#
#0 8 * * * /u01/../soe_bench.sh RapidKit -rt 2:00 uc 100 >> /../morning.log 2>&1
#0 12 * * * /u01/../soe_bench.sh RapidKit -rt 1:00 uc 100 >> /../lunchtime.log 2>&1
```

```
#0 17 * * * /u01/../soe_bench.sh RapidKit -rt 5:00 uc 100 >> /../evening.log 2>&1
################################################################################
```

Backup and Housekeeping The following cron entry shows how an online RMAN backup of the database is conducted - the cron entry must be done on all nodes for general housekeeping, however the backup is executed from one node.

```
################################################################################
# backup the database with a level 0 compressed and remove
old backups and archivelogs generate IO on server
# perform a archivelog backup that will clear down redo every 6 hours
#
0 04 * * * /u01/../backup_all.sh >> /tmp/backup.log 2>&1
0 08 * * * /u01/../archivelog_all.sh >> /../archivelog_backup.log 2>&1
0 12 * * * /u01/../archivelog_all.sh >> /../archivelog_backup.log 2>&1
0 16 * * * /u01/../archivelog_all.sh >> /../archivelog_backup.log 2>&1
0 20 * * * /u01/../archivelog_all.sh >> /../archivelog_backup.log 2>&1
0 00 * * * /u01/../archivelog_all.sh >> /../archivelog_backup.log 2>&1
################################################################################
################################################################################
# housekeeping routine to keep on top of log files generated by workloads
#
0 8 * * * /u01/../housekeeping.sh >> /../housekeeping_routine.log 2>&1
0 16 * * * /u01/../housekeeping.sh >> /../housekeeping_routine.log 2>&1
#
################################################################################
```

RAC Instance OLTP - Linear Growth The following cron entry shows an OLTP entry with spikes and surges at particular times for a RAC database - the cron entry must be done on all nodes

```
################################################################################
# OLTP - create a normal 8 hour load to simulate x users running on a
database this create a trend we then add in some spikes to create
# seasonality  - i.e. there is a pattern that every day users spike -
to create seasonality within seasonality we then add in further surges
#
# linear growth every day by 25 users
```

```
59 23 08 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 50 >> /../day1.log 2>&1
59 23 09 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 75 >> /../day2.log 2>&1
59 23 10 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 100 >> /../day3.log 2>&1
59 23 11 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 125 >> /../day4.log 2>&1
59 23 12 * * /u01/v/soe_bench.sh pdb01  -rt 23:55 -uc 150 >> /../day5.log 2>&1
59 23 13 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 175 >> /../day6.log 2>&1
59 23 14 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 200 >> /../day7.log 2>&1
59 23 15 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 225 >> /../day8.log 2>&1
59 23 16 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 250 >> /../day9.log 2>&1
59 23 17 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 275 >> /../day10.log 2>&1
59 23 18 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 300 >> /../day11.log 2>&1
59 23 19 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 325 >> /../day12.log 2>&1
59 23 20 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 350 >> /../day13.log 2>&1
59 23 21 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 375 >> /../day14.log 2>&1
59 23 22 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 400 >> /../day15.log 2>&1
59 23 23 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 425 >> /../day16.log 2>&1
59 23 24 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 450 >> /../day17.log 2>&1
59 23 25 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 475 >> /../day18.log 2>&1
59 23 26 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 500 >> /../day19.log 2>&1
59 23 27 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 525 >> /../day20.log 2>&1
59 23 28 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 550 >> /../day21.log 2>&1
59 23 29 * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 575 >> /../day22.log 2>&1
# futher spikes to show seasonality wihtin seasonality
0 7 * * * /u01/../soe_bench.sh pdb01 -rt 4:00 -uc 100 >> /../morning1.log 2>&1
0 9 * * * /u01/../soe_bench.sh pdb01 -rt 1:00 -uc 100 >> /../morning2.log 2>&1
#
# create an OLTP spike every 09:00-10:00 12:00-14:00 and 17:00-21:00
with a general useage with a general usage throughout the day
#59 23 * * *  /u01/../soe_bench.sh pdb01  -rt 23:59 -uc 50 >> /../daily.log 2>&1
#0 7 * * * /u01/../soe_bench.sh pdb01 -rt 2:00 uc 500 >> /../morning.log 2>&1
#0 12 * * * /u01/../soe_bench.sh pdb01 -rt 2:00 uc 1000 >> /../lunch.log 2>&1
#0 17 * * * /u01/../soe_bench.sh pdb01 -rt 2:00 uc 1000 >> /../evening.log 2>&1
##############################################################################
```

RAC Instance OLAP The following cron entry shows an OALP entry with spikes and surges at particular times for a RAC database - the cron entry must be done on all nodes

```
##############################################################################
# OLAP - create a load that simulates a batch load running every night to do some
data crunching at 22:00
# activate this line if you want OLAP load for 500 users for 8 hours which
simulates OLAP (Data Warehouse)
#
#0 22 * * * /u01/../sh_bench.sh pdb01 -rt 8:00 uc 10 >> /tmp/daily.log 2>&1
##############################################################################
```

RAC Instance DataMart The following cron entry shows an DataMart entry with spikes and surges at particular times for a RAC database - the cron entry must be done on all nodes

```
##############################################################################
# OLAP Data Mart - create a load that simulates a data mart every day and night
for hours with spikes etc
#
#0 09 * * * /u01/../soe_bench.sh pdb01 -rt 23:55 -uc 200 >> /../daily.log 2>&1
#0 8 * * * /u01/../soe_bench.sh pdb01 -rt 2:00 uc 100 >> /../morning.log 2>&1
#0 12 * * * /u01/../soe_bench.sh pdb01 -rt 1:00 uc 100 >> /../lunchtime.log 2>&1
#0 17 * * * /u01/../soe_bench.sh pd01 -rt 5:00 uc 100 >> /../evening.log 2>&1
#0 22 * * * /u01/../sh_bench.sh pdb01 -rt 8:00 uc 5 >> /../daily_OLAP_dml.log 2>&1
##############################################################################

0 15 * * * /u01/../soe_bench.sh RapidKit -rt 23:59 -uc 200 >> /../daily.log 2>&1
|--------| |------------------| |------| |--------||------|  |------------| |---|
```

NOTE: Ensure that the first column is not an astrix '*' as this means execute the prevailing command every minute and you will consume all resources...

B.1.4 Database Parameters for Swingbench

Several configuration steps are required specifically for the execution of the Swingbench application prior to any experiments for example the correct number of processes, Memory configuration and Redo logs are required to ensure a smooth execution of workloads. Create a tablespace to store the data into:

```
sqlplus / as sysdba
```

```
alter system set processes=3000 scope=spfile;
alter system set job_queue_processes=64 scope=spfile;
alter system set db_recovery_file_dest_size=100G scope=spfile;
alter system set sga_max_size=3500M scope=spfile;
alter system set sga_target=3500M scope=spfile;
alter system set "_ORACLE_SCRIPT"=true;

set lines 200
column member format a100
SELECT b.MEMBER,a.GROUP#, a.THREAD#,a.SEQUENCE#,a.bytes/1024/1024,
a.status, b.type
FROM v$log a, v$logfile b
WHERE a.GROUP#=b.GROUP#;

Alter database add logfile group 4('<redo_location>/redo04.log') size 100m;
Alter database add logfile group 5('<redo_location>/redo05.log') size 100m;
Alter database add logfile group 6('<redo_location>/redo06.log') size 100m;

alter database drop logfile group <number_from_query>;

sqlplus / as sysdba
create tablespace soe
datafile '/<file_location>/soe01.dbf' size 10000M,
'/<file_location>/soe02.dbf' size 10000M autoextend on;

create tablespace sh
datafile '/<file_location>/sh01.dbf' size 10000M,
'/<file_location>/soe02.dbf' size 10000M autoextend on;
```

Note: It is advisable that a general DBA level of expertise is required in making these changes and thus it is probable that multiple datafiles be created and assigned to a tablespace. If the database is a clustered database then changes have to be made on each instance in the cluster and datafiles will be stored in a diskgroup of ASM.

Ensure the database is in archivelog mode as this is essential if the system is running a standby database. Archivelogs are shipped between the two instances via the

Oracle Product DataGuard [Cor21b].

```
SQL> shutdown immediate;

Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup mount;
ORACLE instance started.

Total System Global Area 5027385344 bytes
Fixed Size 2298736 bytes
Variable Size 1040190608 bytes
Database Buffers 3976200192 bytes
Redo Buffers 8695808 bytes
Database mounted.

SQL> alter database archivelog;

Database altered.

SQL> alter database open ;

Database altered.
```

Archivelogs will now be written to the server file system in the location of the *db_recovery_file_dest* parameter. The more activity is generated the more archivelogs are created (IO) thus it is imperative that housekeeping routines are employed to clear down the archivelogs routinely. Once they have been applied to any standby database. RMAN [Cor21l] archivelog backup is the preferred mechanism to perform this task and was employed in our experiments.

B.1.5 Workloads Dataset Sizes - Tables and Indexes

To find the size of the data set one must log into the database with DBA privileges and execute the following SQL

```
set lines 200
set pages 200
column owner format a5
column segment_type format a15
column segment_name format a35
compute sum of GB on report
break on report
select owner, segment_name, segment_type, bytes/1024/1024/1024 GB
  from dba_segments
 where owner='SOE'
   and segment_type in ('TABLE','INDEX')
order by segment_type, 4;
```

The above SQL provides a sample output of the following, **NOTE:** the sample output has been cropped for the purpose of saving space in the thesis thus the 'sum' does not reflect the correct value because a number of tables and indexes have not been added in the sample output.

```
OWNER SEGMENT_NAME                        SEGMENT_TYPE            GB
----- ----------------------------------- --------------- ----------
SOE   WAREHOUSES_PK                       INDEX           .000976563
SOE   PROD_CATEGORY_IX                    INDEX           .000976563
SOE   PROD_SUPPLIER_IX                    INDEX           .000976563
SOE   PROD_NAME_IX                        INDEX           .000976563
SOE   CUSTOMERS                           TABLE           1.29394531
SOE   ADDRESSES                           TABLE           1.32421875
SOE   ORDERS                              TABLE           1.55078125
SOE   ORDER_ITEMS                         TABLE           2.72167969
----------
sum                                                       14.7597656
```

B.2 Operating Software and Database Products

The following section will describe the operating system and database products used and their versions during the main pieces of work, which was to execute, run the workloads, gather and store the metric data.

OS Type	Products and Versions
Single Database Instance Configuration	
OEL Linux 2.6.39	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4),
OEL Linux 2.6.39	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4),
Clustered Database Instance Configuration	
OEL Linux 2.6.39	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4, • Grid Infrastructure (12.1.0.2), • Oracle Automatic Storage Manager (12.1.0.2),
OEL Linux 2.6.39	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4, • Grid Infrastructure (12.1.0.2), • Oracle Automatic Storage Manager (12.1.0.2),
Standby Database Instance Configuration	
OEL Linux 2.6.39	• Enterprise Edition (12.1.0.2), • Data Guard (12.1.0.2), • Enterprise Manager Agent (12.1.0.4),
Central Repository Details	
OEL Linux 2.6.39	• Enterprise Edition (11.2.0.3), • Enterprise Manager R4 including Webserver and BIPublisher (12.1.0.4), • Enterprise Manager Agent (12.1.0.4),

Table B.1: Operating System & Database Products

B.3 Automatic Workload Respository - (AWR)

Setting the retention policy of the AWR repository is done by executing the following procedure in the database:

```
BEGIN
DBMS_WORKLOAD_REPOSITORY.modify_snapshot_settings(
retention => 43200, -- Minutes (= 30 Days). Current value retained if NULL.
interval  => 60);   -- Minutes. Current value retained if NULL.
END;
/
```

The changes to the settings are reflected in the DBA_HIST_WR_CONTROL view. Typically the retention period should capture at least one complete workload cycle. If the system has monthly archive and loads a 1 month retention time would be more beneficial than the default 7 days. An interval of ”0” switches off snapshot collection, which in turn stops much of the self-tuning functionality, hence this is not recommended. Automatic collection is only possible if the STATISTICS_LEVEL parameter is set to TYPICAL or ALL. If the value is set to BASIC manual snapshots can be taken, but they will be missing some statistics.

B.3.1 AWR Execution

To gather a snapshot of database statistics one can execute the following command:

```
EXEC DBMS_WORKLOAD_REPOSITORY.create_snapshot;
```

To query which particular snapshots are available? One can query the *dba_hist_snapshot* view as a user with dba privileges execute the following SQL:

```
select snap_id,
instance_number,
snap_level,
cast(begin_interval_time as date) date_time
from dba_hist_snapshot
order by snap_id asc;
```

```
SNAP_ID INST_NUMBER  SNAP_LEVEL DATE_TIME
------- -----------  ---------- -------------------
26987         1          1      05/11/2021 01:00:01
26988         1          1      05/11/2021 02:00:06
26988         2          1      05/11/2021 02:00:06
26989         1          1      05/11/2021 03:00:10
26989         2          1      05/11/2021 03:00:10
26990         1          1      05/11/2021 04:00:15
26990         2          1      05/11/2021 04:00:15
26991         2          1      05/11/2021 05:00:19
26991         1          1      05/11/2021 05:00:19
```

There are two SQL scripts supplied that reside in the $ORACLE_HOME of the host. This is the location where the Database software is installed and located. In this example it is located /u01/app/oracle/product/12.1.0.2/dbhome_1

```
@\$ORACLE_HOME/rdbms/admin/awrrpt.sql
@\$ORACLE_HOME/rdbms/admin/awrrpti.sql
```

The AWR scripts will prompt the user to enter the report format (html or text), the start snapshot id, the end snapshot id and the report filename. The resulting report can be opened in a browser or text editor accordingly.

```
SQL> @$ORACLE_HOME/rdbms/admin/awrrpti.sql

Specify the Report Type
~~~~~~~~~~~~~~~~~~~~~~~
AWR reports can be generated in the following formats.
Please enter the name of the format at the prompt.
Default value is 'html'.

'html'          HTML format (default)
'text'          Text format
'active-html'   Includes Performance Hub active report
```

```
Enter value for report_type: text

Type Specified:  text

Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

DB Id     Inst Num DB Name      Instance     Host
------------ -------- ------------ ------------ ------------
3063321339        2 CDBM03       cdbm032      brmex2adm08v
m03.acs.orac
le.com
* 3063321339        1 CDBM03       cdbm031      brmex2adm07v
m03.acs.orac
le.com

Enter value for dbid: 3063321339
Using 3063321339 for database Id
Enter value for inst_num: 1
Using 1 for instance number

Specify the number of days of snapshots to choose from
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Entering the number of days (n) will result in the most recent
(n) days of snapshots being listed.  Pressing <return> without
specifying a number lists all completed snapshots.

Enter value for num_days: 1

Listing the last 2 days of Completed Snapshots

Snap
Instance     DB Name        Snap Id    Snap Started    Level
------------ ------------ --------- ------------------ -----
cdbm031      CDBM03           27058 08 Nov 2021 00:00      1
```

```
27082 09 Nov 2021 00:00      1
27083 09 Nov 2021 01:00      1
27084 09 Nov 2021 02:00      1

Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 27083
Begin Snapshot Id specified: 27083

Enter value for end_snap: 27084
End   Snapshot Id specified: 27084

Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is awrrpt_1_27083_27084.txt.
To use this name, press <return> to continue,
otherwise enter an alternative.

Enter value for report_name: cdb031_awrrpt.txt
```

B.3.2 AWR Report - Example

Understanding and interpreting an AWR report is the responsibility of the Database Administrator. This thesis is not intended to provide detailed interpretation of the AWR report or present performance tuning findings. The reason why this AWR example report is publish is to raise awareness of how database statistics on performance and configuration metrics that one would analyse for the purposes of monitoring or performing a capacity planning exercise.

```
WORKLOAD REPOSITORY report for

DB Name  DB Id      Instance  Inst Num Startup Time    Release     RAC
-------- ---------- --------  -------- --------------- ----------- ---
CDBM03   3063321339 cdbm031    1       14-Mar-21 16:17 12.1.0.2.0  YES

Host Name        Platform                 CPUs Cores Sockets Memory(GB)
```

```
--------------- ---------------- ------- ---- ----- ------- ----------
brmex2adm07vm03. Linux x86 64-bit            16    16       1      63.03

Snap Id         Snap Time                Sessions Curs/Sess Instances   CDB
---------       -------------------      -------- --------- --------- -----
Begin Snap:     27083 09-Nov-21 01:00:33      110       1.6         2 YES
End Snap:       27084 09-Nov-21 02:00:37      110       1.6         2 YES
Elapsed:                     60.08 (mins)
DB Time:                      7.04 (mins)
```

The above sample output displays basic information on the configuration of the host the database is running, and the period the rest of the report provides information for.

A good place to start to understand the daatabase workload is the section ”Top 10 Foreground Events by Total Wait Time”. This section describes how the database on the whole is performing for the snapshot period, which in this example is 1 hour.

```
Top 10 Foreground Events by Total Wait Time
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Total Wait       Wait   % DB Wait
Event                                Waits Time (sec) Avg(ms) time Class
------------------------------ --------- --------- -------- ---- --------
Failed Logon Delay                   240       240  1000.12 56.9 Other
DB CPU                                         126.6           30.0
CRS call completion                  480      17.7    36.80  4.2 Other
cell single block physical rea    54,736      13.5     0.25  3.2 User I/O
Disk file operations I/O           2,654       7.7     2.89  1.8 User I/O
GPnP Initialization                  480       7.6    15.90  1.8 Other
control file sequential read      33,962       6.6     0.20  1.6 System I
gc cr disk read                   25,256       2.3     0.09   .5 Cluster
PX Deq: Slave Session Stats       21,246       2.2     0.11   .5 Other
Disk file Mirror Read              8,556       2.1     0.25   .5 User I/O

Wait Classes by Total Wait Time
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Wait Class	Total Wait Waits	Avg Wait (ms)	% DB Time	Active (sec)	Avg time	Sessions
Other	713,143	0.40		287	68.0	0.1
DB CPU				127	30.0	0.0
User I/O	71,906	0.35		25	6.0	0.0
System I/O	51,751	0.19		10	2.4	0.0
Cluster	39,833	0.10		4	1.0	0.0
Scheduler	5	543.10		3	.6	0.0
Concurrency	9,828	0.07		1	.2	0.0
Application	1,231	0.31		0	.1	0.0
Commit	235	0.29		0	.0	0.0
Network	16,599	0.00		0	.0	0.0
Configuratio	16	0.13		0	.0	0.0

Wait Classes by total wait time as shown above, gives the DBA a birds eye view of, which database is suffering from excessive IO, CPU or other key areas the database maybe *waiting* on. This information allows the DBA to focus on where potential bottlenecks are, and if there is a monitoring role or capacity planing exercise, to extract the maximum number from the report.

The AWR report also lists specific area's relating to CPU, Memory and IO as listed in the sample output below. Where we can see the CPU extract that would be similar to the operating command *'TOP'* when execute on the host command line.

Host CPU

CPUs	Cores	Sockets	Load Average Begin	End	%User	%System	%WIO	%Idle
16	16	1	1.48	1.45	2.9	1.5	0.0	95.6

Instance CPU

% of total CPU for Instance:	1.7
% of busy CPU for Instance:	38.3

```
%DB time waiting for CPU - Resource Mgr:        0.6

IO Profile                  Read+Write/Second Read/Second  Write/Second
~~~~~~~~~~                  ---------------- -----------  -------------
Total Requests:                     29.7            28.3           1.5
Database Requests:                  15.8            15.2           0.6
Optimized Requests:                 28.0            28.0           0.0
Redo Requests:                       0.5             0.0           0.5
Total (MB):                          0.3             0.3           0.0
Database (MB):                       0.1             0.1           0.0
Optimized Total (MB):                0.3             0.3           0.0
Redo (MB):                           0.0             0.0           0.0
Database (blocks):                  16.0            15.2           0.8
Via Buffer Cache (blocks):          15.8            15.2           0.6
Direct (blocks):                     0.2             0.0           0.2



Memory Statistics
~~~~~~~~~~~~~~~~~                        Begin          End
                                    -----------    ----------
Host Mem (MB):                        64,547.7     64,547.7
SGA use (MB):                         12,928.0     12,928.0
PGA use (MB):                          5,264.4      5,270.4
% Host Mem used for SGA+PGA:             28.18        28.19
```

B.4 OEM SQL Queries

The main schema that houses configuration and performance data gathered by the Intelligent OEM agent, is held under the SYSMAN user, and are predominately the MGMT$ views. By extracting the GUID as the unique identifier we can create custom SQL statements that can vastly reduce the laborious cumbersome tasks of manually extracting metric data.

B.4.1 SQL Statements - Extracting Database Entities that are Monitored

This SQL query can extract the database entities currently configured to be monitored by the OEM Agents. Notice how the query brings back database entities that are of type rac_database (Clustered), oracle_pdb (Pluggable) and oracle_database (Normal-database)

```
column target_guid format a55
column target_name format a35
column target_type format a35
SELECT
rownum,
DBMS_LOB.substr(target_guid, 50) as target_guid,
target_name,
target_type
FROM sysman.mgmt$target
WHERE target_type in ('rac_database','oracle_database','oracle_pdb')
order by rownum, target_type asc
```

```
TARGET_GUID                      TARGET_NAME         TARGET_TYPE
-------------------------------   ---------------    ------------------
602974144292644FE0532A03410A7BCD E3DE0003           oracle_database
4B0109D9AAD0470EAFB30BACF6DF2079 GSAPRMY            oracle_database
8D72E86781E499FE0015AF16BB63AEA9 cdbm01_cdbm012     oracle_database
19AAD7189BFFC395002DDF5155DBA09E WIND12C1STBY1      oracle_pdb
8379BEDCA9921D42FE994B30394612A6 RAC1OLTP           rac_database
77DE4012A795D47A400197B18AC52B72 cdbm01             rac_database
5884F18AB3CA210DE0532A03410A1D50 dbm01              rac_database
```

By extracting the GUID for each database we wish to extract data from, we can then plug the relevant information

B.4.2 SQL Statements - Extracting particular Metrics

The following SQL statements lists the metric data that are collected for a particular instance. There are several things to note:

1. The target GUID is required
2. Metric_name is a list of the category of metrics

```
select
distinct(b.metric_column),
b.metric_name
from mgmt$target a, mgmt$metric_details b
where a.target_guid=b.target_guid
and b.target_guid in ('8D72E86781E499FE0015AF16BB63AEA9')
--and b.metric_name in ('DATABASE_SIZE','instance_throughput')
order by  metric_name asc;
```

The above query for extracting the list metric data is key because we can map this data to the metrics provided by the Cloud Vendor. This is also important for workload placement as we can place a workload on the max_value extract from the metric over a time series data. Sample output is provided below

```
METRIC_COLUMN                  METRIC_NAME
------------------             -------------------
ALLOCATED_GB                   DATABASE_SIZE
USED_GB                        DATABASE_SIZE
logons                         Database_Resource_Usage
max_tot_cpu_usage_ps           db_inst_cpu_usage
cpu_time_pct                   instance_efficiency
cpuusage_ps                    instance_efficiency
cpuusage_pt                    instance_efficiency
physreads_ps                   instance_throughput
physreads_pt                   instance_throughput
physwrites_ps                  instance_throughput
physwrites_pt                  instance_throughput
total_memory                   memory_usage
logreads_ps                    instance_throughput
logreads_pt                    instance_throughput
```

B.4.3 SQL Statements - Extracting Metric data at 10 minute intervals

The following query extracts metric data from the OEM repository in its most granular form at 10 minute intervals.

```
column target format a20
column metric format a25
column metric_type format a25
column value format a35
select
rownum,
a.target_name as target,
b.metric_name as metric,
b.metric_column as metric_type,
to_char(b.collection_timestamp, 'dd/mm/yyyy hh24:mi:ss') as sample_time,
b.value as value
from mgmt$target a, mgmt$metric_details b
where a.target_guid=b.target_guid
and b.target_guid in ('8D72E86781E499FE0015AF16BB63AEA9')
and b.metric_column ='logreads_ps'
and b.collection_timestamp between to_date('01/11/2021 17:00',
'dd/mm/yyyy hh24:mi:ss') and to_date('09/11/2021 17:00',
'dd/mm/yyyy hh24:mi:ss')
order by b.collection_timestamp
```

Producing the following sample output

```
instance       METRIC              METRIC_TYPE  SAMPLE_TIME         VALUE
-------------- ------------------- -----------  ------------------- ------
cdbm01_cdbm012 instance_throughput logreads_ps  01/11/2021 17:07:01 2339.7
cdbm01_cdbm012 instance_throughput logreads_ps  01/11/2021 17:17:01 1904.8
cdbm01_cdbm012 instance_throughput logreads_ps  01/11/2021 17:27:01 1807.2
cdbm01_cdbm012 instance_throughput logreads_ps  01/11/2021 17:37:01 2139.3
cdbm01_cdbm012 instance_throughput logreads_ps  01/11/2021 17:47:01 1824.2
cdbm01_cdbm012 instance_throughput logreads_ps  01/11/2021 17:57:01 1807.0
```

```
cdbm01_cdbm012 instance_throughput logreads_ps  01/11/2021 18:07:01 2316.8
cdbm01_cdbm012 instance_throughput logreads_ps  01/11/2021 18:17:01 1896.6
```

B.4.4 SQL Statements - OEM Hourly Data Roll-up Procedure

Aggregation batch processing jobs are managed in the Oracle Enterprise Manager repository (Oracle database) via the use of the database scheduler. Aggregating data can be done manually by executing the following command. Substituting the parameters for the correct job.

```
BEGIN
DBMS_SCHEDULER.RUN_JOB(
JOB_NAME            => 'em_rollup()',
USE_CURRENT_SESSION => FALSE);
END;
/
```

B.4.5 SQL Statements - Extracting Metric data at Hourly intervals

```
column target format a25
column metric format a25
select
rownum,
a.target_name as target,
b.metric_column as metric_type,
to_char(b.rollup_timestamp, 'dd/mon/yyyy) hh24:mi:ss') as sample_time,
b.maximum as value,
b.average as ave,
b.minimum as min
from mgmt$target a, mgmt$metric_hourly b
where a.target_guid=b.target_guid
and b.target_guid in ('4C9BC9D1BB2C73DAB9A286416DD337F4')
and b.metric_column ='logreads_ps'
and b.rollup_timestamp between to_date('01/11/2021 17:00',
```

```
'dd/mm/yyyy hh24:mi:ss') and to_date('10/11/2021 17:00',
'dd/mm/yyyy hh24:mi:ss')
order by b.rollup_timestamp
```

producing the following sample out

```
ROWNUM TARGET    METRIC_TYPE SAMPLE_TIME           VALUE     AVE         MIN
--------------  -----------  --------------------  --------  ----------  -----
1 cdbm01_cdbm011 logreads_ps  09/nov/2021 22:00:00  37409.616 8164.50867  24.165
2 cdbm01_cdbm011 logreads_ps  09/nov/2021 23:00:00  18803.573 3393.10217  19.097
```

B.5 Python

B.5.1 Python Libraries and Packages

There were several packages and libraries needed to build the algorithms programmatically and these are listed in tables B.2 and B.3. All packages are available as opensource and can be installed to a locally run IDE such as Pycharm Community as long as the pip installer is configured.

Library	Description	Specific Task
Pandas	Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. When the metric data is extracted using SQL it is passed into a Pandas DataFrame or Series to be processed. It is especially powerful when slicing and dicing metric data when the instances are joined in one dataframe	• series, • DataFrame,
Numpy	NumPy is a library used for working with arrays and is used to reshape or flatten any arrays when fitting values from workloads	
OS	OS Package provides functions for interacting with the Operating System. We store the target OCI configurations as templates (Excel) therefore OS allows us to read the path of where files are stored when reading and writing files. We also write the placement to a file as a report.	• listdir
Operator	Operator package constructs a callable that assumes an iterable object (e.g. list, tuple, set) as input, and fetches the n-th element out of it. If we need to extract the max_value from a list of metrics in a list of instances this is a library that facilitates that requirement.	• itemgetter
RE	Re is a Regular Expression package that can be used to search for specific patterns in data. This is used alongside Operator in searching for max_values of a metric from an instance	
Collections	The collection Module in Python provides different types of containers. A Container is an object that is used to store different objects and provide a way to access the contained objects and iterate over them. Some of the built-in containers are Tuple, List, Dictionary. It is used to convert list of lists to a dictionary	• defaultdict
binpacking	Bin Packing contains greedy algorithms to solve two typical bin-packing problems and is used to answer the minimum number of targets required and the consistent number of targets if all workloads are treated equally.	
cx_Oracle	cx_Oracle is a Python extension module that enables access to Oracle Database. It conforms to the Python database API specification. It is used to connect to the OEM repository and extract data.	

Table B.2: Table of Python Libraries Workload Placement

Library	Description	Specific Task
pickle	Serialising and de-serialising object structures to a byte-stream. once a model is created in the format E.G. SARIMAX(1,0,0)(1,1,1,24) we serialise and store the model against a metric and instance in a schema	
Matplotlib	Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. We use it to visualise data from adfuller, plot_acf, durbin_watson	• style, • dates,
statsmodels	This package was used to perform forecasts and perform particular statistical tests on the metric data. adfuller, durbin_watson and plot_acf are tests that determine stationariness, trend and seasonality. These tests determine if the data should be differenced and influence the p,d,q,P,D,Q parameters of ARIMA, SARIMA, SARIMAX. OLS is an Ordinary Least Squares Linear regression algorithm	• OLS, • adfuller, • ARIMA, • SARIMAX, • TBATS, • durbin_watson, • plot_acf
Scipy	SciPy is an opensource package to solve scientific and mathematical problems. fft, ifft and fftpack are related to Fast Fourier Transforms, which are used to filter and improve the accuracy of the forecast models. FFT works with the forecast models	• fftpack, • fft, • ifft
time	Time package is used to handle time-related tasks. Data extracted from the database may have a timestamp and thus time is a module that can change a timestamp into a date, character for processing	• datetime • timedelta
Multiprocessing	Multiprocessing supports the spawning of processes and is used to split a *job* of work into parallel threaded tasks. If the forecast models where taking a while with larger models, for example, (30,0,0),..,(30,2,30). Using the Pool attribute creates a pool of resources that can help speed up the forecasting time	• pool
sklearn.metrics	Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python and was used to learn the metric data. This package also calculated the RMSE of each model.	• LinearRegression, • preprocessing, • cross_validation, • mean_squared_error
codecs	Codecs is a package to translate bytes into strings and was used in conjunction with pickle. Once a forecast model is found and stored in the database it needs to be translated from or to a string within the python packages	
Pandas	Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. When the metric data is extracted using SQL it is passed into a Pandas DataFrame or Series to be processed. Autocorrelation_plot was called to visualise the data and used to automate the d,D of the SARIMAX models if trend or seasonality was detected.	• series, • autocorrelation_plot, • DataFrame,
Numpy	NumPy is a library used for working with arrays and is used in-conjunction with SciPy and Fourier Transforms. It is also used to plot graphs with matplotlib package and autocorrelation_plot package.	
cx_Oracle	cx_Oracle is a Python extension module that enables access to Oracle Database. It conforms to the Python database API specification. It is used to connect to the OEM repository and extract data.	

Table B.3: Table of Python Libraries Forecasting