Holistic Cloud Computing Environmental Quantification and Behavioural Analysis

Peter Michael Garraghan

Submitted in accordance with the
requirements for the degree of Doctor of Philosophy

UNIVERSITY OF LEEDS

University of Leeds, School of Computing,
July 2014

Intellectual Property and Publication Statements

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly authored publications have been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

I. Solis Moreno, P. Garraghan, P. Townend, J. Xu "Analysis, Modelling and Simulation of Workload Patterns in a Large-Scale Utility Cloud", IEEE Transactions on Cloud Computing, 2014. In this paper, a comprehensive study of the characteristics and behaviour of users and tasks temporally and spatially is conducted. The method of analysis was proposed by Ismael Solis Moreno and myself equally. The cluster analysis and distribution analysis of users and tasks was conducted by myself, with Ismael Solis Moreno contributing towards the month-day analysis. The design, implementation and validation of the simulator was performed by Ismael Solis Moreno. The paper was reviewed and improved by Paul Townend and Jie Xu. The content of this paper is the base of the workload analysis in Chapter 4.

P. Garraghan, P. Townend, J. Xu "Real-Time Fault-Tolerance in Federated Cloud Environments" IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2012. This work develops and evaluations a real-time fault-tolerant mechanism for deployed in a federated Cloud. The system model, experiment, analysis and evaluation was conducted by myself. The paper was reviewed and improved by Paul Townend and Jie Xu. The conclusions of this work is used for defining workload assumptions presented in Chapter 4.

I. Solis Moreno, P. Garraghan, P. Townend and J. Xu, "An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models", in proceedings of The 7th International Symposium on Service-Oriented System Engineering (SOSE), San Francisco California, USA, 2013. For this paper,

the proposed method of analysis was proposed by Ismael Solis Moreno and myself. Ismael Solis Moreno conducted the implementation of the workload generator within the Cloud simulator. The conducted analysis of users and tasks was performed equally by Ismael Solis Moreno and myself. The paper was reviewed and improved by Paul Townend and Jie Xu. The content of this paper is the base of the coarse-grain analysis in Chapter 3 and the workload characterization in Chapter 4.

P. Garraghan, P. Townend, J. Xu "An Analysis of the Server Characteristics and Resource Utilization in Google Cloud" IEEE International Conference on Cloud Engineering (IC2E), USA, 2013. This paper analyzes the characteristics and inefficiencies of resource utilization in Cloud datacenter hardware. The data extraction, method of analysis and analysis results was conducted by myself. The paper was reviewed and improved by Paul Townend and Jie Xu. The analysis of server characteristics and resource utilization is presented in Chapter 5.

P. Garraghan, P. Townend, J. Xu "An Empirical Failure-Analysis of a Large-Scale Cloud Computing Environment" 15th IEEE International Symposium on High Assurance Systems Engineering (HASE), USA, 2014. This paper presents the temporal and spatial analysis of failures within a large-scale Cloud computing environment. The data extraction, analysis method, analysis results and applicability of work was conducted by myself. The paper was reviewed and improved by Paul Townend and Jie Xu. The contents of this work are presented in the failure-analysis section in Chapter 6.

P. Garraghan, I. Solis Moreno, P. Townend and J. Xu, “An Analysis of Failure-Related Energy Waste in a Large-Scale Cloud Environment”, IEEE Transactions on Emerging Topics in Computing, 2014. For this paper, the extraction of the data, methodology of analysis, failure identification, temporal and spatial failure analysis and the examples of applicability was performed by myself. The analysis of the impact of energy waste in the analyzed system was equally conducted by Ismael Solis Moreno and me. The proposed energy model was conducted by Ismael Solis Moreno. The paper was reviewed and improved by Paul Townend and Jie Xu. This method of analysis and analysis of failure-related energy waste is presented in Chapter 6.

Abstract

Cloud computing has been characterized to be large-scale multi-tenant systems that are able to dynamically scale-up and scale-down computational resources to consumers with diverse Quality-of-Service requirements. In recent years, a number of dependability and resource management approaches have been proposed for Cloud computing datacenters. However, there is still a lack of real-world Cloud datasets that analyse and extensively model Cloud computing characteristics and quantify their effect on system dimensions such as resource utilization, user behavioural patterns and failure characteristics. This results in two research problems: First, without the holistic analysis of real-world systems Cloud characteristics, their dimensions cannot be quantified resulting in inaccurate research assumptions of Cloud system behaviour. Second, simulated parameters used in state-of-the-art Cloud mechanisms currently rely on theoretical values which do not accurately represent real Cloud systems, as important parameters such as failure times and energy-waste have not been quantified using empirical data. This presents a large gap in terms of practicality and effectiveness between developing and evaluating mechanisms within simulated and real Cloud systems.

This thesis presents a comprehensive method and empirical analysis of large-scale production Cloud computing environments in order to quantify system characteristics in terms of consumer submission and resource request patterns, workload behaviour, server utilization and failures. Furthermore, this work identifies areas of operational inefficiency within the system, as well as quantifies the amount of energy waste created due to failures. We discover that 4-10% of all server computation is wasted due to Termination Events, and that failures contribute to approximately 11% of the total datacenter energy waste. These analyses of empirical data enables researchers and Cloud providers an enhanced understanding of real Cloud behaviour and supports system assumptions and provides parameters that can be used to develop and validate the effectiveness of future energy-efficient and dependability mechanisms.

Declaration

Some parts of the work presented in this thesis have previously appeared in the following papers:

P. Garraghan, I.S. Moreno, P. Townend, J. Xu, *"An Analysis of Failure-Related Energy Waste in a Large-Scale Cloud Environment",* IEEE Transactions on Emerging Topics in Computing, 2014.

I.S. Moreno, **P. Garraghan**, P. Townend, J. Xu, "*Analysis, Modelling and Simulation of Workload Patterns in a Large-Scale Utility Cloud*", IEEE Transactions on Cloud Computing, 2014

P. Garraghan, P. Townend, J. Xu "*An Empirical Failure-Analysis of a Large-Scale Cloud Computing Environment*" in the proceedings of the 15th IEEE International Symposium on High Assurance Systems Engineering (HASE), pp. 113-120, 2014.

P. Garraghan, P. Townend, J. Xu "*Using Byzantine Fault-Tolerance to Improve Dependability in Federated Cloud Computing*" in the International Journal of Software and Informatics**,** vol. 7, pp.221-237, 2013.

P. Garraghan, P. Townend, J. Xu, *"An Analysis of the Server Characteristics and Resource Utilization in Google Cloud"* in the proceedings of the IEEE International Conference on Cloud Engineering (IC2E), pp. 124-131, USA, 2013.

I.S. Moreno, **P. Garraghan**, P. Townend, J. Xu*, "An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models"* in the proceedings of the 7th IEEE International Symposium of Service-Oriented System Engineering (SOSE), USA. pp. 49-60, 2013.

P. Garraghan, P. Townend, J. Xu, "*Real-Time Fault-Tolerance in Federated Cloud Environments*" in the proceedings of 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), pp. 118-123, China, 2012.

P. Garraghan, P. Townend, J. Xu, "*Byzantine Fault-Tolerance in Federated Cloud Computing*" in the proceedings of the 6th IEEE International Symposium of Service-Oriented System Engineering (SOSE), pp. 280-285, USA, 2011.

P. Garraghan, P. Townend, J. Xu, "*Dependability in Federated Cloud environments*" in the proceedings of The UK e-Science All-Hands Meeting (AHM), York, UK, 2011.

Acknowledgement

As I write this acknowledgement for the last page of my thesis, I can't help but reflect on the road traversed to arrive at this destination, as well as the people who were part of journey.

First and foremost, I'd like to thank my supervisors Professor Jie Xu and Dr. Paul Townend: it's difficult to express my gratitude to them for giving me the opportunity to join the group and supporting me in my research pursuits. I'd also like to thank my thesis examiners Professor Santosh Shrivastava and Dr. Karim Djemame for their excellent feedback and comments given in the examination, as well as Professor John Davies for taking the time to sanity check my thesis.

I'd also like to thank immensely my colleague Dr. Ismael Solis Moreno who acted as a huge inspiration and mentor in my development as a researcher, and spent many late evenings with me staring at whiteboards. I also want to extend my gratitude to my colleagues and friends within the Distributed Systems and Services Group and the wider world, most notably Dr. David Webster and Renyu Yang.

Last but not least I'd like to thank my family for their eternal support and love, allowing me to reach this life milestone.

List of Acronyms

AD	Anderson-Darling
CDF	Cumulative Distribution Function
CMS	Cloud Management System
CPU	Central Processing Unit
CSV	Comma Separated Value
ER	Entity Relationship
ERP	Enterprise Resource Planning
FaaS	Failure-as-a-Service
GoF	Goodness of Fit
HaaS	Hardware-as-a-Service
HDD	Hard Disk Drives
HPC	High Performance Computing
HQL	Hive Query Language
IaaS	Infrastructure-as-a-Service
KPI	Key Performance Indicators
LXC	Linux Container
MTBF	Mean Time Between Failure
MTTR	Mean Time to Repair
NAS	Network Attached Storage
NIST	National Institute of Standards and Technology
OS	Operating System
P2P	Peer-to-Peer
PaaS	Platform-as-a-Service
PDF	Probability Distribution Function
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
SaaS	Software-as-a-Service
SECaaS	Security-as-a-Service
SoA	Service Oriented Architecture
SOAP	Simple Oriented Access Protocol
SQL	Structured Query Language
VM	Virtual Machine
VMM	Virtual Machine Manager
VO	Virtual Organization
VPN	Virtual Private Network

Contents

Figures

Tables

1 Introduction

1.1 Research Motivation

Modern computing systems can be characterized by their requirement for substantial computing power; this has been augmented by an exponential increase in the volume of data available for processing. One approach to fulfill this requirement is to establish large scale computing systems, which typically require significant time and financial effort. Such systems are usually designed with respect to a maximum, least or average usage requirement, making the resultant system either under-used or unable to deliver desired functionality due to shortage of resources. Furthermore, such systems do not exhibit the ability to grow dynamically, and require extensive design and development procedures to cope with evolving user requirements. This problem is aggravated when the user requirements are susceptible to unexpected changes. Therefore, contemporary computing systems are inflexible by nature and it is extremely difficult to guarantee on-demand availability for them in a cost effective manner.

Cloud computing has emerged as a new paradigm to facilitate the establishment of large scale, flexible computing infrastructures that are able to provide services that are encapsulated as workloads to customers on demand. However, Cloud computing faces new challenges that are not found in traditional distributed systems and require extensive and in-depth research to characterize and quantify real operational behaviour. This in turn will facilitate research into a large number of research domains including security, resource management, dependability and energy-efficient computing.

1.2 Research Context

Research into distributed systems is enhanced by the empirical analysis of real-world systems. This is required in order to understand and study behaviour within operational environments as well as quantify system characteristics such as consumer behaviour, workload utilization and server failures [1]. Such analysis is critical for both researchers and system providers; for providers, it enables a way to understand and study behavioural patterns and identify areas of operational inefficiency in terms of energy-waste within the system [2]. For

researchers, it enables deeper understanding of system behaviour and is fundamental to defining and justifying system assumptions; both of which can be exploited in order to improve Quality of Service (QoS) and aid in business decision making. Moreover, parameters derived from such analyses can be exploited in order to develop simulation models which accurately reflect the operational conditions of a system. Such parameters are critical when evaluating the practicality of mechanisms which aim to enhance the resource management [3] and dependability [1] of a system.

Cloud computing systems have been characterized as large-scale multi-tenant systems that are able to dynamically scale-up and scale-down computational resources to consumers who have diverse Quality of Service requirements [8][10]. Such system infrastructure are typically deployed in datacenters which are collocated systems within the same physical location due to common environmental and physical security requirements, forming the Cloud datacenter. In recent years, a number of dependability and resource management approaches have been studied for Cloud computing datacenters. However, there are a lack of real-world Cloud datasets that have been analysed and extensively modelled in order to study the characteristics of Cloud computing and quantify their effect on system dimensions such as resource utilization, user behavioural patterns and failure characteristics. This leads in two key research problems. First, without the holistic analysis of real-world Cloud systems characteristics, dimensions of interest such as workload behavioural patterns, user submission patterns, server resource utilization, failure rates and energy consumption cannot be quantified, resulting in inaccurate research assumptions of Cloud system behaviour. Second, the simulated parameters used in state-of-the-art Cloud mechanisms such as resource management and dependability currently rely on theoretical values which do not accurately represent real Cloud systems, as important parameters such as failure times and energy-waste are not quantified from empirical data. This leads to a large gap in terms of practicality and effectiveness between developing and evaluating mechanisms within simulated and real Cloud environments.

This thesis presents a general analysis method and an in-depth analysis of large-scale production Cloud computing environments in order to study system

behaviour and quantify system characteristics in terms of consumer submission and resource request patterns, workload resource utilization and execution length, server utilization characteristics, and failures. Furthermore, this work identifies areas of operational inefficiency within a system, as well as quantifies the amount of energy waste created due to failures. These analyses of empirical data provide researchers and Cloud providers with an increased understanding of real Cloud characteristics and behavioural patterns, and provide realistic system assumptions and experiment parameters. These parameters can be used in a large number of Cloud research domains that require accurate workload behaviour, and develop and validate the effectiveness of energy-efficient and dependability mechanisms.

1.3 Aims and Objectives

The aim of this research is to study and present an in-depth analysis of large-scale production Cloud computing environments. This is urgently needed as the characteristics and behavioural patterns of real Cloud computing environments need to be comprehensively studied in order to understand large-scale system behaviour as well as produce quantifiable parameters and model key components such as consumer behaviour, workload characteristics, server resource utilization and failure characteristics. Furthermore, this work also identifies and quantifies the operational inefficiencies within these systems in terms of wasted resource utilization of servers as well as energy waste due to failures within the system. The findings in this work can be leveraged by other researchers and Cloud providers in order to evaluate developed mechanisms based on realistic simulation parameters.

Specifically, the objectives of this research are as follows:

i) *To enable a more thorough understanding of the issues in accurately modelling Cloud computing environments and comprehensively study how Cloud behavioural characteristics impact the system.* Cloud computing has been stated to be a multi-tenant, heterogeneous, and flexible system that provisions computational services to diverse consumer requirements. However, there is a lack of in-depth studies that attempt to quantify fundamental system characteristics and their impact on the system environment. This is challenging due to the scale and

complexity of systems analysis when considering the relationships of components within a large-scale Cloud environment.

ii) *To provide an in-depth method of holistic analysis for operational trace logs to study and model Cloud behaviour*. This work proposes a general method of analysis that can be applied to Cloud datasets in order to study, model and quantify behavioural patterns and characteristics of system operation.

iii) *Empirical analysis and modelling of large-scale Cloud computing behavioural patterns and characteristics.* This research aims to study behavioural patterns and quantify key components within the Cloud environment, specifically within the areas of user behaviour, server resource utilization, workload execution length, resource utilization and failures. Such analysis is critical in understanding and building realistic research assumptions of the Cloud operational environment and deriving accurate simulation patterns derived from real systems.

iv) *To identify and quantify operational inefficiency in terms of wasted resource utilization and energy waste due to failures within large-scale Cloud environments.* This work for the first time quantifies and studies the root cause of energy waste due to failures within large-scale distributed systems. These results can be used by providers to identify areas and root causes of operational inefficiency within production systems, and by researchers to enhance dependable energy-efficient research that presently rely on theoretical values and assumptions for energy waste.

1.4 Research Methodology

The research methodology of this work consists of two core components:

- Identification of the challenges involved in studying and accurately modelling components of large-scale Cloud computing environments.
- An approach to address these challenges through the development of an analysis method, which leverages techniques such as cluster analysis, distribution modelling, temporal and spatial analysis of Cloud datasets to study system behaviour and construct accurate Cloud environment models

for user behaviour, workload classification and utilization, server resource utilization, failures and energy-waste due to failures.

This analysis is decomposed into four main areas to characterize and model the Cloud environment: Workload, Servers, failures and failure-related energy-waste. To facilitate this, it was necessary to model the lifecycle and relationships of components in a real production Cloud datacenter trace log, as well as construct an analysis infrastructure for data extraction and processing in a timely manner. Each area of analysis includes the study of statistical properties and characteristics in order to analyze and model the behaviour of the Cloud environment, as well as how findings can be used for practical application.

1.5 Major Contributions

The major contributions of this work are:

- *The identification of in-depth analysis of operational traces from Cloud computing datacenters as an effective means to comprehensively understand system behaviour and enhance system assumptions of Cloud research*. Current Cloud research presently use theoretical values, small-scale experiments or characteristics from non-Cloud systems to derive system assumptions. These assumptions are greatly enhanced by the empirical analysis of real-world Cloud datacenters in order to derive and quantify realistic system behaviour.
- *A solution to the challenges of analyzing large-scale Cloud environments to obtain meaningful results and identify relationships between components within the system*. Due to the massive scale and number of components within a Cloud environment, there are challenges in performing non-superficial analyses to model the relationships and behaviour of users, workload, servers, failures and energy-waste. Furthermore, there are additional technical challenges in extrapolating and processing relevant data in a timely manner due to data size and heterogeneity, as well as lack of suitable analysis infrastructure. Finally, there are a limited number of methods of Cloud component analysis and modelling which contain a sufficient degree of abstraction that can be applied to Cloud trace logs agnostically of the underlying trace log format. This work provides a comprehensive method of analysis which models and captures key

characteristics of Cloud components including workload, failures, servers and resource-inefficiency which can be applied to system trace logs when performing empirical system analysis.

- *The analysis of Cloud datasets to study and model realistic user behaviour, task classification, Cloud utilization models and failure characteristics and models*. Through the analysis of real world Cloud operational traces it is possible to empirically study actual Cloud computing behaviour. Specifically, we present a non-superficial and comprehensive spatial and temporal analysis of user submission rates and resource estimation, workload classification, resource utilization and execution length, and failure and repair characteristics of workloads and servers. Furthermore, we present the model parameters for these components that can be used by other researchers to construct their own simulations derived from realistic Cloud environment assumptions, as well as aid system architects when designing resource managers to improve Cloud workload reliability.
- *The study and quantification of operational inefficiencies within Cloud environment*. This work not only quantifies resource waste in terms of server utilization, but also presents the first analysis of failure-related energy waste produced within a Cloud environment. This is critical in not only identifying operational inefficiencies, but also quantifying their impact within large-scale systems in terms of energy cost.

1.6 Thesis Organization

The thesis is composed of seven chapters, of which this is the first:

Chapter 2 provides an introduction to the topics of Cloud computing and dependability. Existing work in analyzing and modelling Cloud computing environments specifically within the areas of workloads and servers is explored. Furthermore, the current state-of-the-art in failure analysis and energy waste in modern systems is discussed. This work is presented in order to understand the challenges associated with studying, quantifying and modelling large-scale Cloud environments.

Chapter 3 The case study of this research - the Google Cloud trace log - is presented. The trace log specification, data attributes, and lifecycle of datacenter components are discussed and described in detail. The modelling of relationships

between the system logs, the method of data extraction and construction of the analysis infrastructure required for data processing is presented and discussed in detail. This chapter concludes by analysis of the statistical parameters of coarse-grain statistics of the datacenter operation.

Chapter 4 presents the method, analysis and modelling of workload behaviour within large-scale Cloud computing environments, including user submission rates, resource estimation, task resource utilization and execution length. This chapter concludes by demonstrating how these results have been integrated within energy-efficient resource management mechanisms.

Chapter 5 presents the method and analysis of server characteristics within the Cloud datacenter, including resource utilization per server architecture type as well as quantifying the operational inefficiencies of servers in terms of wasted resource utilization.

Chapter 6 presents the analysis of failures and failure-related energy waste of both workload and servers. Specifically, the failure characteristics of tasks and servers are comprehensively studied and modelled. Furthermore, the amount of energy-waste created due to failures is quantified and studied in detail. This chapter concludes by discussion of practical application of these results.

Chapter 7 summarises findings and provides conclusions and outlines potential future research directions for this work.

2 Cloud Analytics

2.1 Evolution of Modern Computing Systems

This chapter describes the broad context of this research - i.e. analysis of Cloud computing datacenters to quantify system characteristics, behavioural patterns and system inefficiencies. The evolution of modern computing systems and relevant technologies are presented in order to better understand the emergence of Cloud computing. The background of Cloud computing is discussed and then defined within the context of this thesis. The concepts of dependability and systems analysis and how they are can be used to enhance system operation is defined and discussed in detail. Finally, the current state-of-the-art in Cloud analytics within the areas of workload, failures, servers and energy-waste are discussed, highlighting the importance of such work within this thesis.

2.1.1 Software System Model Definition

In order to better understand the emergence of Cloud computing presented in this thesis, it is necessary to present and discuss the conceptual design of a system and the evolution of modern distributed systems.

A *software system* is composed of a number of *components,* which work together to provide *functionality* and interact with entities in the *system environment* [4] as shown in Figure 2.1. These components can be humans, software, hardware or even other systems. This concept can be used recursively, in that individual components may be composed of multiple subcomponents operating within a system environment. In this model, a *consumer* (which can also be seen as a *user)* is defined as another system that exists within the system environment, interfacing at the system boundary.

Computing and communication systems can be characterized by five fundamental properties: functionality, performance, cost, security and dependability [5]. The *function* of a system, which is the intended purpose of the system, is usually described by the *functional specification* in terms of system functionality and performance. Systems which provide different properties of interest and pursue different functions will exhibit different system behaviour.

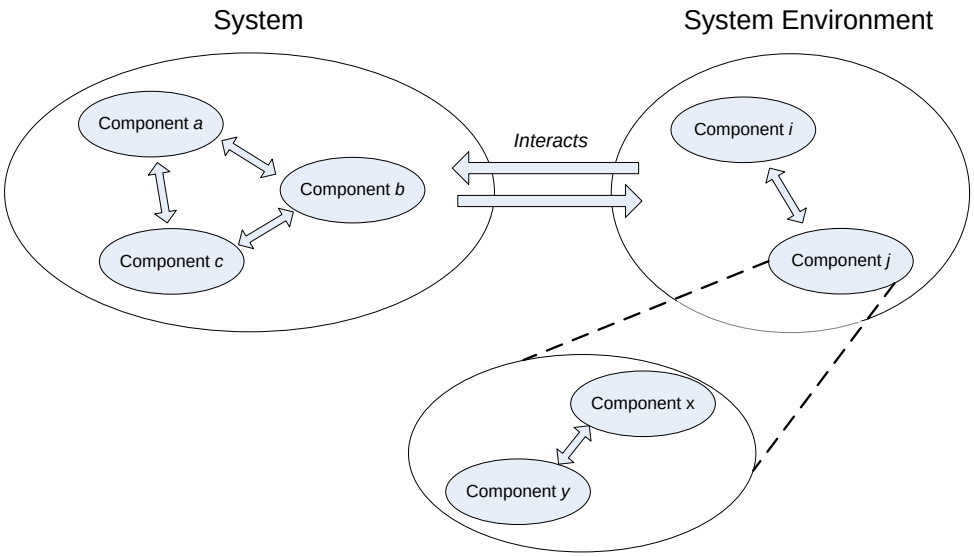


Figure 2.1 Conceptual model of a software system.

The *behaviour* of the system is what the system does in order to perform its defined function and can be described as a sequence of states [5].

2.1.2 Conceptual Design of Software Systems

The architecture of most software systems are designed by separation into three distinct layers: presentation, application logic and resource [6] as presented in Figure 2.2. These layers are defined as follows:

The *presentation layer* is responsible for presenting information to the system environment, and the interaction and communication with components that exist outside of the system. Examples of such entities include human users or other systems. A presentation layer can be implemented in a number of ways, such as a graphical user interface or a component that formats data into a given syntax. A typical example for this type of service is a program which drives an ATM screen.

The *application logic layer* is responsible for data processing and performing actual operation requested by a user through the presentation layer. This layer is also referred to as the services offered by the system. A typical example for this type of service is a program which implements a withdrawal operation from a bank account.

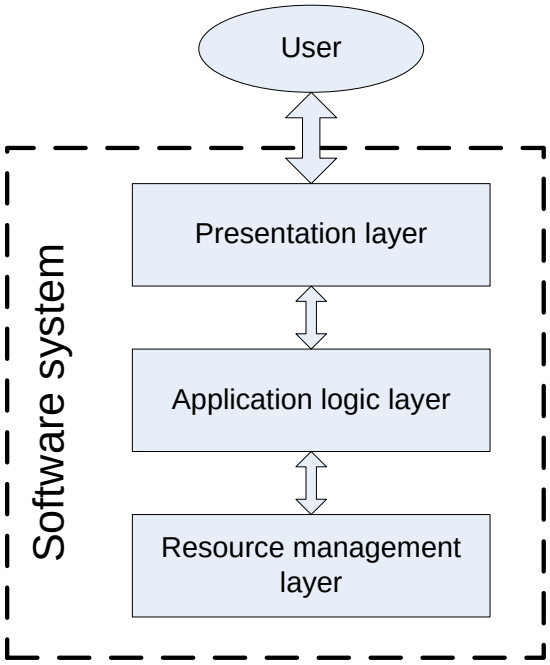


Figure 2.2 The layers of a software design system.

The *resource* management layer is responsible for the management and implementation of data sources in the system irrespective of nature of the data source. This includes data residing within databases, file systems and other data repositories.

2.1.3 Tiered software System Architectures

These three layers are conceptual, logically separating the functionality of the software system. In reality, implementing such layers within a system can be combined and distributed in a variety of ways, referred to as *tiers*. There are four fundamental types of software system architectures, dependant on tier organisation: 1-tier, 2-tier, 3-tier and *N*-tier [6].

1-tier architectures consist of all three layers (presentation, application logic and resource management) merged into a single tier as shown in Figure 2.3(a). These architectures were the result of the development of mainframe-based systems that can be interacted with through the use of dumb terminals which display information prepared by the mainframe. Such practises were necessary at the time of conception due to system CPU efficiency being high priority due to limited CPU resources available. These systems are monolithic and interaction with the system environment is limited through the use of dumb terminals; as a result, 1-tier systems are viewed today as legacy systems as they are difficult and expensive to maintain as they are essentially monolithic pieces of code.

2-tier architectures are an evolution of the 1-tier architecture, and emerged due to the development of the PC, which replaced dumb terminals of a mainframe with smaller computer systems that interacted with larger, more computationally powerful servers. Conceptually, this resulted in the separation of the presentation layer from the server, and instead merging with the client (i.e. a user's PC) as shown in Figure 2.3(b). This architecture is commonly known as 'Client-Server'.

This architecture presents two distinct advantages. First, as the presentation layer is separated from the server, the system can utilize computing resources from the client instead of the application logic and resource management layers, reducing resource utilization overhead. Second, it is possible to modify the presentation layer for different purposes without increasing system complexity, allowing different functionality for individual clients. However, there are limitations with this architecture in terms of scalability when increasing the number of clients interacting with the system causing increased server overhead. An additional issue is the legacy problem when 2-tier systems are used for objectives not originally intended; as the code written for clients is separate from the server, it is possible for clients to be developed to communicate with multiple servers enabling service integration. This results in increased system size, complexity and server dependency as modifications to a server also requires the client to be subsequently modified, causing clients to become larger and more complex than originally intended.

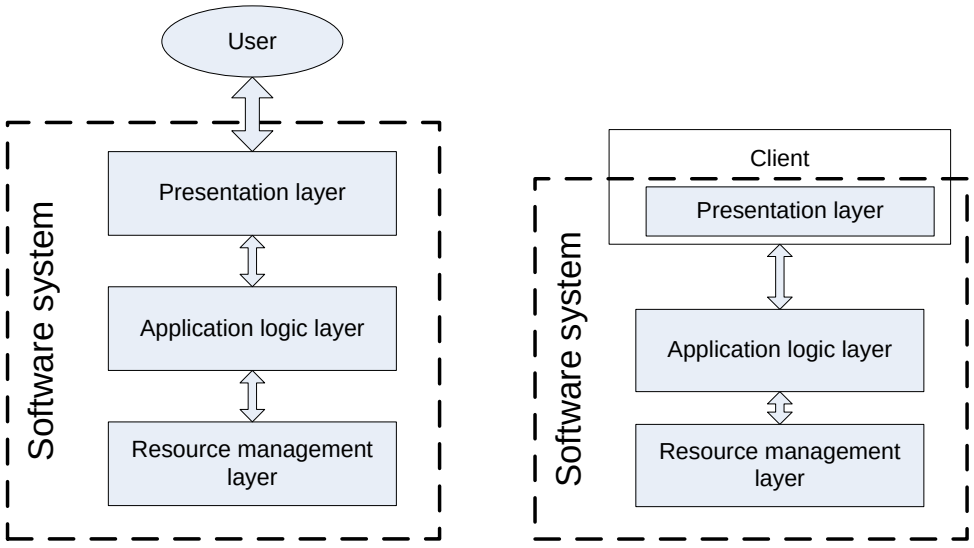


Figure 2.3 (a) 1-tier, (b) 2-tier software system architecture.

3-tier architectures overcome some of the problems within 2-tier architectures; these architectures, as shown in Figure 2.4(a), feature a distinct separation between the three layers. Similar to the 2-tier architecture, the presentation layer resides on the client, while the application logic resides within a middle tier which communicates between the client and back-end resources. The abstraction and infrastructure that is used to support the application logic is often referred to as the Middleware [7].

3-tier architectures separate the application logic from the resource management layer as well as reducing the complexity of clients by preventing data resources from being accessed directly by the presentation layer. This minimises bottlenecks, as the application logic layer does not need to transfer additional data to the client. Moreover, system maintainability is improved as changes to the Middleware do not require corresponding changes to the presentation layer. By separating the middleware tier and linking it across many nodes, the scalability and reliability of the system is enhanced. However, 3-tier architectures face issues when integrating clients to multiple systems or other three-tier systems. This is due to a lack of standards in terms of interfaces and communication protocols resulting in the requirement of significant development in order to integrate different 3-tier systems together.

N-Tier architectures are similar to 3-tier systems; the main difference however is that they are more capable of linking to other systems and are capable of connecting to the Internet. An *N*-Tier architecture can emerge from one of two situations: the resource layer of the system is composed of other complete 2-tier or 3-tier architecture systems, or an additional tier is created by deploying a Web Service within the presentation layer that is treated as an additional layer due to its complexity in comparison to the client as shown in Figure 2.4(b). However, N-Tier architectures face the same problems as 3-tier architectures in terms of lack of standards to enable interoperability between systems over the Internet, and consequently increased complexity and the amount of middleware required for system integration [151]. This is particularly true when application logic is distributed across multiple machines that each use heterogeneous middleware.

The evolution of software system architectures, from monolithic systems interacting through a dummy terminal, to many complex systems interconnected

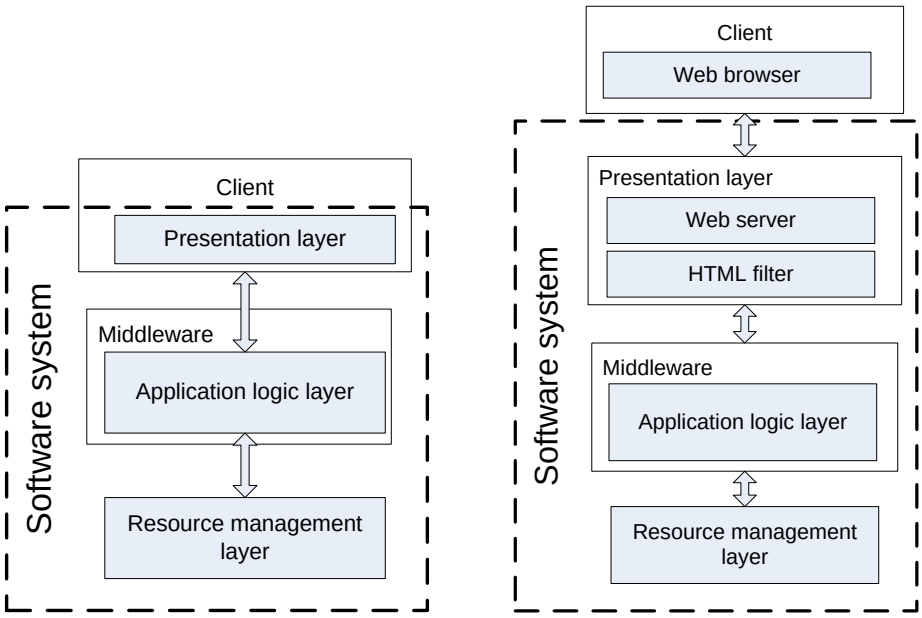


Figure 2.4 (a) 3-tier, (b) *N*-Tier software system architecture.

together which can be accessed over the Internet demonstrates several trends in the evolution of software systems:

The first trend is that software systems are becoming increasingly complex; from the above discussion is it observable that an increasing amount of tiers are required in order to build systems which are capable of handling increased number of users and resource pools. Unfortunately, the introduction of additional tiers results in increased challenges when developing and maintaining the system due to complexity in tier configuration and communication [5].

The second trend is that there is an increasing requirement for software systems to be integrated together. As systems grow more complex, with the integration of additional tiers as well as application logic and data resources residing across multiple machines, there is an increased challenge in effectively integrating systems together. Furthermore, these systems must be capable of integrating and communicating with legacy systems currently used by the organization [150].

The third trends entails that software systems described above have evolved to facilitate the needs of individual organisations; the Internet age has since resulted in the formation of organizations which are in constant state of evolution in order to compete and prosper within the global marketplace [151].

Such an environment has resulted in the need for systems that are capable of integration from different administrative domains, as well as potentially difference organisations. 3-tier and *N*-Tier system architectures are only capable of limited support in terms of system integration in such conditions. This is due to the lack of standards for communication and interactions between systems with heterogeneous architectures and administrative controls, resulting in increased complexity in middleware [5].

Service-oriented architectures have emerged as a means to address these challenges by enabling the development of applications composed of dynamic and loosely coupled applications capable of integrating across multi-organisational systems over the Internet in a standard way.

2.1.4 Service Computing

Service computing focuses on the connection between business processes and IT services so that business processes are seamlessly automated [8]. Service Oriented Architecture (SOA) is an architecture proposed for business to redefine processes to take advantage of 'formerly isolated component activities' [5]. Such architectures offer advantages over *N*-tier architectures in that complexity is reduced due to middleware decentralisation and modification less likely to affect existing services due to loose-coupling, in addition to well supported standards for developing cross-organizations systems over the Internet. A typical implementation of SOAs are through the use of Web Services, which are defined as self-contained, modular business applications that use standard interfaces over the Internet [23]. Web service developers make use of standards to enable interoperability between services. Two popular service based protocols for communication are SOAP [148] and REST [154].

The maturity of service computing has enabled the resurgence of a long sought concept: systems providing services to consumers as computing utilities.

2.1.5 Utility Computing

In modern day society, utilities such as water, gas, telecommunication and electricity are deemed as requirements for fulfilling routines in daily life [8]. In this context, we define *utility* as an essential service that can be easily obtained by the general population.

The idea of computing utility was realised as early as 1966 [9], where it was envisioned that computing networks would mature to reach a point where the idea of 'computer utilities' was made a reality and worked in similar principle to electrical and telephone utilities; able to provision computing service such as computing resources, development platforms or applications to consumers. There are two main actors within such environments; consumers and providers. *Providers* are defined as entities that own and maintain the underlying infrastructure to provision computing service. *Consumers* are entities (individuals or companies) that require computing power in order to achieve business objectives [10]. This service utility model for provisioning computing service provides a distinct advantage over traditional computing: Consumers pay providers for the amount of computer resources used over a given time frame, instead of investing, building and maintaining their own computing infrastructure which may be heavily underutilized for the majority of its life span if it is required for only handling large demand on a short-term basis.

There have been a number of technologies developed which increase the feasibility of computer utility. The first is the creation of communication protocols capable of forming distributed computer systems that are able to interact across the globe via the Internet, enabling the formation of potentially massive computer resource pools [8]. The second technology that has recently seen resurgence is Virtualization, which enables the abstraction of computing resources from the physical infrastructure enabling computing resources used by a consumer to be dynamically added and released on demand controlled by a virtual management system [11]. A typical use of this technology is the creation of Virtual Machines (VMs) which are self-contained environments which encapsulate state and virtual computing resources. The third technology is the evolutionary shift to service computing as discussed above, which allows computing utility and IT service to be provisioned to consumers as an automated business process through the use of standards.

As consumers move towards adopting such systems, the quality and reliability of the service provided becomes increasingly important. However, the level of service required by consumers can vary significantly depending on their business objectives. As a result, it may not be possible to fulfil all expectations for every

consumer from a service provider's perspective; hence a balance needs to be made via a negotiation process. At the end of the negotiation process, providers and consumers commit to a Service Level Agreement (SLA) [24]. SLAs provide a firm definition of the service agreement between service providers and consumers which includes defining the related service parties, service level objectives which formally express guarantee service conditions, and service parameters which are a measureable representation of service parties' obligations in order to measure whether service has been satisfactorily provisioned [153]. This typically includes service level guarantees, parameters and actions required in the case of violation [64]. One element provisioned and enforced through an SLA is the Quality of Service (QoS); QoS is a broad topic in the domain of Distributed Systems research, and can include a large variety of attributes ranging from geographical, economic, performance, real-time and security constraints of the service. Service providers use the SLA to optimize their infrastructures to meet the agreed terms of service, whilst service consumers use it to ensure that the agreed QoS has been fulfilled.

Modern day IT usage has grown at a substantial rate; consuming 1.8% of the global electricity consumption [25] and increased global data traffic more than fourfold within the past five years, and an expected further threefold increase by 2018 [26]. To meet this need, there has been a significant growth of large-scale distributed interconnected systems which are capable of providing computing service as a utility.

Through breakthroughs in research and technology, as well as the increased heterogeneity in consumer objectives, there have been a number of distributed systems with distinct characteristics that have emerged within the past few decades to pursue specific consumer objectives and attempt to realise the vision of computing utility.

Cluster computing [12][13][14] is a type of High-Performance Computing (HPC) paradigm that is capable of solving complex and huge-scale computing problems at reduced costs in comparison to traditional supercomputing systems. A cluster system is defined as a series of independent machines connected together by a network. Through the use of middleware, it is possible to create the illusion of a single system by abstracting the underlying infrastructure from users thereby

reducing system complexity [15]. Cluster systems have two principle actors; producers and consumers, which act in the same manner as the role for providers and consumers in utility computing [16].

Peer-to-Peer (P2P) systems are a class of systems which enables distributed resources to perform a function in a decentralized manner [17]. There is no division between client and servers since all nodes within the system are treated equally and can act as both clients and servers simultaneously. The objective of P2P systems is to aggregate resources across the system to improve system reliability and system scalability by reducing the dependency on centralized points within the system [18]. Such systems are highly effective for file sharing [19], security authentication and shared workspaces [20].

Grid computing systems coordinate resources not subject to centralized control using standard protocols and interfaces in order to deliver non-trivial service enabling sharing and selection of a large number of system types [21]. The motivation for Grid computing is to solve problems associated with "*coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations (VOs)*" [22]. Such systems are typically suited for solving research problems that require massive amounts of computation and data within the domains of science, engineering and commerce [22].

It is possible for these distributed systems to be deployed within Datacenters; co-located systems within the same physical location in order to satisfy common environmental and physical security requirements, as well as ease system maintenance [27]. Datacenters are traditionally used by consumers as co-locational facilities (i.e. consumers are provisioned physical space to purchase and configure their own IT equipment which is maintained by the datacenter provider). Datacenter systems continue to grow and an unprecedented rate, with Datacenter sales in 2013 totalling over $143 billion [28], with a forecasted growth of Datacenter installations between 2013-2016 at 15-20% [29], and number of server racks in the UK increasing from 7.7 million by 15% [25].

Unfortunately, all of the above systems fail to realise the vision of true computing utility; for cluster computing, such systems are typically tightly coupled, resulting in limitations in portability due to middleware modification.

Grids are project-oriented in nature, with VOs forming in order to complete mutually beneficial objectives [66]. Furthermore, the business model of Grids is composed of providing computing services and resources as units to be spent, allowing the provider to manage schedule resources and available allotted time prior to provisioning service to consumers. This characteristic is debilitating to its capability to provide true utility to consumers on an ad-hoc basis [66]. In the case of Datacenters, providers only provision physical space to consumers who purchase, install and maintain their own IT infrastructure while the datacenter provider is responsible the physical security and operational environment conditions. This ultimately leads to limitations in dynamically providing computing utility to consumers as described previously.

The most modern paradigm to emerge that addresses this problem and further realises utility computing provisioned as a service is Cloud computing; loosely coupled systems - typically deployed within datacenters - capable of dynamically providing computing service and utility to consumers.

2.2 Cloud Computing

2.2.1 Cloud Computing Definition

Cloud computing has emerged as an increasingly popular means of providing utility computing to consumers. There is currently no standard definition for Cloud computing, however there are a number of proposed definitions. A popular definition for Cloud computing is taken from the National Institute of Standards and Technology (NIST) [30], which states that Cloud computing is:

"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction".

Furthermore, Cloud computing from an implementation perspective is defined as [31]:

"parallel and distributed system consisting of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources base on service-level agreements established through negotiation between the service provider and consumers".

Cloud computing environments are typically deployed within datacenters, which is owned by a provider or a third party. In recent years, datacenters have been increasingly deployed to provide global IT services and have been a key factor in the increased formation and uptake of Cloud computing which is typically deployed using such systems. As a result, within this work the term Cloud datacenter and Cloud computing environment are synonymous.

2.2.2 Cloud Computing Characteristics

As systems are capable of being constructed and implemented by using a wide variety of architectures, technologies, designs, etc. it is advantageous to define the characteristics of the system in order to comprehensively understand the definition and behaviour agnostic of physical implementation. Identified by NIST [30], there are five essential characteristics of Cloud computing.

On-demand self-service: Cloud self-service interfaces provide mechanisms that support the management of the entire service delivery lifecycle. This allows consumers to acquire, manage and utilize computing resources and capability such as server resources and network storage automatically without needing direct human interaction from the service provider.

Broad network access: Cloud services are delivered over standard network protocols and can be accessed through standard interfaces to promote heterogeneous devices (i.e. Mobile phones, workstations, laptops). This grants providers the capacity to deliver a variety of Cloud services to a wide selection of devices used by consumers over networks.

Resource Pooling: The Cloud provider is able to pool resources to multiple consumers in the form of a multi-tenant model. These resources are assigned dependant on the demands of the consumer, who generally do not have fine-grained control or knowledge over the precise location of the resources provisioned. Instead, depending on the provider policy, consumers are capable of specifying the location of resource pooling at a higher level of abstraction such as country, state or datacenter.

Rapid Elasticity: Elasticity is the capability of a computing system to add or remove capacity from the system environment. Cloud resources are capable of being scaled-in and out dynamically in order to maintain expected QoS for

consumers as well as reduce costs; the latter is made possible as consumers are capable of adding and removing resources when required. These resources can be scaled in two directions: vertically (scale-up) by adding or removing resources from prior existing VMs or horizontally (scale-out) by adding or removing additional VMs.

Measured Service: Resource usage within Cloud systems can be monitored, managed and reported to both consumers and providers transparently for metering and billing. This enables consumers capable of being billed based on their usage of Cloud resources and services while providers are able to track billing patterns in order to enhance provisioned service.

As a result of these system characteristics, there are secondary characteristics that manifest within Cloud environments which have been observed and identified within the literature.

Massive scalable and complex architectures: With the increased market uptake of Cloud services [32][33][34], there has been a substantial growth in infrastructure [25][29], number of consumers [26][35] and Cloud services provisioned in datacenters; this growth is further augmented by the use of virtualisation technology and the ability to provision Cloud services on-demand. This has resulted in the increase in not just the number of Cloud datacenter facilities, but also the scale of such systems. Such increases have resulted in more complex management, architectures and applications [36][37][38], resulting in challenges when attempting to understand system characteristics as well as the relationship between components and their impact on system behaviour.

Wide diversity of dynamic workloads due to heterogeneous consumer demands. Clouds typically are multi-tenant nature, with consumers pursuing different business objectives and QoS requirements [10]. As a result, the properties and characteristics of workload deployed within the Cloud environment can vary substantially [39][40]. Due to the Cloud's ability to scale to consumer needs, as well as the diversity of consumer business objectives through the use of rapid elasticity, workload in Cloud can be substantially heterogeneous in terms of resource consumption and execution length.

2.2.3 Cloud Computing Taxonomy

It is important to define Cloud computing by its capabilities in order to formulate a concise taxonomy. Cloud computing implementations can be classified according to their service delivery model in terms of Software, Platform and Infrastructure [41]. Different service models result in different functionality and responsibility for the Cloud consumer and provider, respectively. Furthermore, Cloud systems can also be deployed in a number of ways to provision service classified as Private, Public, Hybrid, Community and Federated [41][42]. These concepts and their relation to each other are shown in Figure 2.5 and are described in subsequent sections.

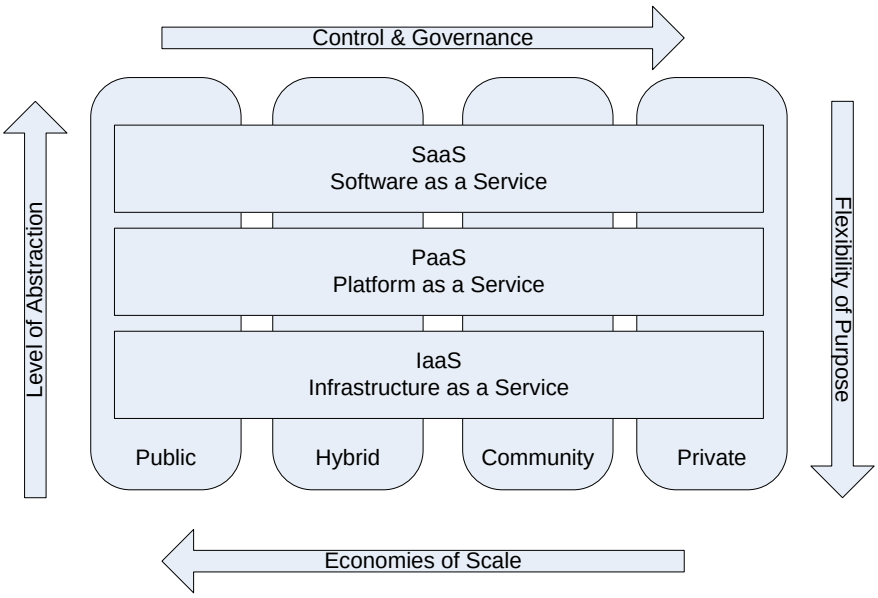


Figure 2.5 Cloud taxonomy of service deployment and service model.

2.2.4 Cloud Actors

It is necessary to define the principle actors within Cloud computing agnostic of service objective and physical deployment. According to the NIST Cloud Computing Reference Architecture [41] there are five principle actors within Cloud computing environments:

Cloud Provider: Entities that are responsible for management and administration of the physical Cloud infrastructure as well as the mechanisms to deliver service to the consumer [42]. Providers are generally responsible for the overall management of Cloud, which includes physical server maintenance, cooling systems and VM scheduling and resource management.

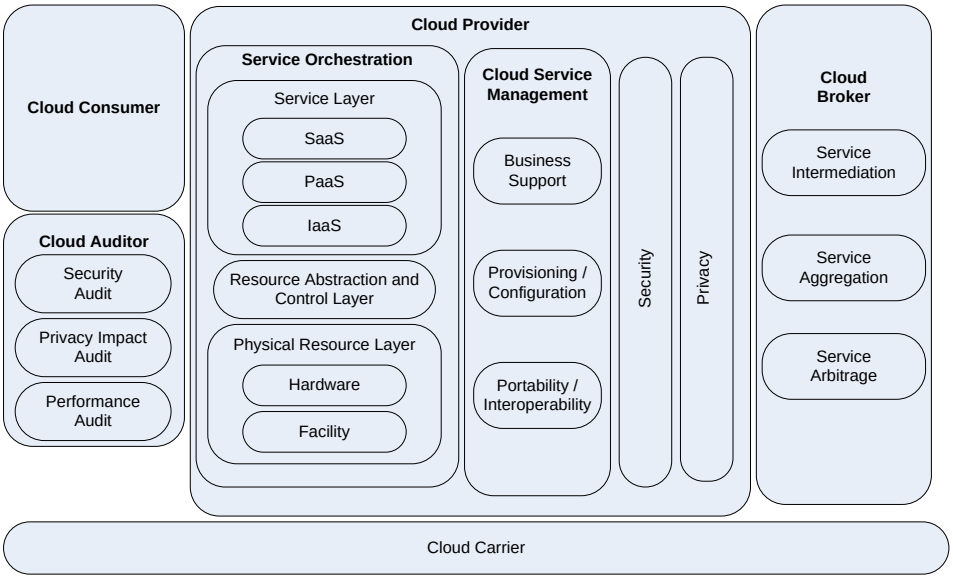


Figure 2.6 NIST Cloud Computing Model.

Cloud Auditor: Entities independent of the provider that assess Cloud services in order to evaluate whether services conform to standards such as system operation, performance and infrastructure security controls. The purpose of the auditor is to ensure that these controls are implemented and deployed correctly and that they produce the desired results for the system.

Cloud Broker: Responsible for the integration of Cloud services; Cloud brokers are perceived as consumers to Cloud providers, while perceived as providers themselves when interacting with consumers. The three main activities of a Cloud broker performs on Cloud service can be divided into intermediation, aggregation and arbitrage.

Cloud Carrier: Responsible for connecting, managing and transporting Cloud computational service through network, telecommunication and other devices between providers and consumers.

These actors within the context of the Cloud environment are presented in Figure 2.6.

2.2.5 Service Model

As described previously in Chapter 2.2.2, a key driving concept behind Cloud computing is provisioning service to consumers. There is a consensus held that Cloud computing can be categorized into one of three fundamental service models [41][43]. Definitions of these service models are as follows:

Software-as-a-Service (*SaaS*): Cloud providers supply remotely run software to consumers via the Internet. Specifically, consumers do not have access or permission to alter the configuration of the underlying operating platform and physical infrastructure running the software. The provider is responsible for software installation, maintenance and management. Examples of typical applications are word processors or project management tools. SAP, Salesforce.com and Oracle On Demand are examples of SaaS providers.

Platform-as-a-Service (*PaaS*): Cloud providers offer a software platform above virtualised infrastructure to consumers. Consumers are able to access a hosting environment (platform) and have complete control over deploying and configuring their own applications whilst having limited configuration over the operating platform. The consumer is unable to configure and access the underlying infrastructure such as the virtualized hardware and network. Examples of PaaS providers are Google App Engine, Flexiscale and Windows Azure.

Infrastructure-as-a-Service (*IaaS*): Providers provision computing resources to consumers, who are able to build their own operating platforms and software using virtualised resources. These resources include CPU, Memory, Storage and Network capability. Consumers are unable to manage the underlying infrastructure that provisions these resources, but are capable of configuring and deploying the operating platform and applications deployed within them. Providers on the other hand are responsible for the management and administration of the underlying physical infrastructure. Examples of IaaS providers are Amazon EC2, Rackspace and GoGrid.

These three deployment models can be deployed on top of each other. For example, a SaaS provider provisions service from a third party PaaS provider to host their applications, which use another third party IaaS for infrastructure. In such a scenario, tiers lower down the Cloud stack are obfuscated from the consumer and provided transparently as shown in Figure 2.7.

In addition, there exist a number of sub-categories that blur the boundaries of the three service models described:

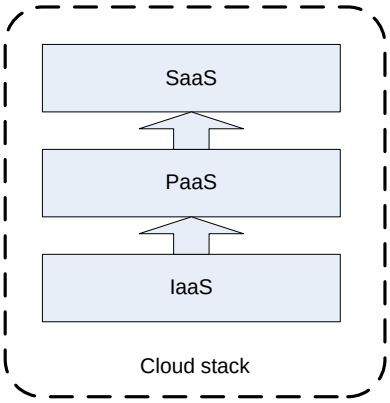


Figure 2.7 Cloud Service Stack.

Failure-as-a-Service (FaaS): A service model that routinely injects faults into Cloud services in order to evaluate service resiliency, identify risks and the impact of failures on the system holistically, and allow providers to schedule failures instead of waiting for unplanned failures to occur [44].

Hardware-as-a-Service (HaaS): This service is traditionally provided in co-locational datacenters, where providers are responsible for the network and physical environmental conditions of the datacenter and provide physical space and computing infrastructure to consumers who are given total control over the physical hardware. This service is best suited for consumers who wish to take advantage of scalable virtualized services whilst having total control over the resource management and security policies of the physical hardware [45].

Security-as-a-Service (SECaaS): This provisions security services such as intrusion detection systems, identity management and Virtual Private Networks (VPNs) to consumers, allowing them to attain desired levels of security and privacy. Example applications include shared datasets being protected from malicious alteration and configurable access control to protect service integrity [46].

2.2.6 Deployment Model

The Cloud service models described previously can be deployed in a number of different configurations as well as exhibit different security constraints, levels of system complexity and implementation cost [41]. There are five identified Cloud deployment models:

Private: The Cloud service is provisioned exclusively to a single organization or institution. The physical infrastructure may be owned or administrated by the

organization or a third party provider. This approach shares similarities to a traditional IT infrastructure, and brings similar advantages in terms of security and data privacy, as data and computation is kept within the boundaries of the organization. Private Clouds allow organizations to improve security due to restricted user access, boundaries and service optimization as the service and infrastructure is configured specifically to facilitate the business objectives of the organization. An example domain where such a deployment model is typically implemented is within the healthcare industry, where there are regulatory standards and jurisdictional constraints on data storage and management.

Public: Cloud services and infrastructure is provisioned to the general public who act as consumers, typically over the Internet. The Cloud infrastructure is hosted and fully managed by the Cloud provider, and consumers have no control over operational policy or location of the physical infrastructure. Public Clouds allow consumers to handle spikes in resources demanded by adding virtual resources to facilitate service load for their business objectives for a finite period of time (for reasons such as sharp increase in service demand). This is advantageous to consumers as they do not need to invest in extending their own infrastructure to handle the temporally high service load. As a result, the use of public Cloud can result in reduced expenditure and operational costs as well as improve resource utilization of a consumer's infrastructure.

Hybrid: Hybrid Clouds refers to the use of both Private and Public Clouds by an organization. Consumers typically outsource a section of their service to the public Cloud, typically non-critical data or services, whilst keeping mission critical data within the private Cloud. Similar to Public Clouds, this deployment allows additional resources to be quickly scaled-up for a finite amount of time to facilitate business objectives during peak demand of service. A common practise in such deployments is the use of Cloud bursting, which leverages a public Cloud for additional computing service when a private Cloud is momentarily unable to fulfil SLA due to high demand and system usage [30].

Community: Community Clouds are defined as multiple organizations with shared interests that share their infrastructure in order to complete their individual business objectives. An example of this would be sharing infrastructure

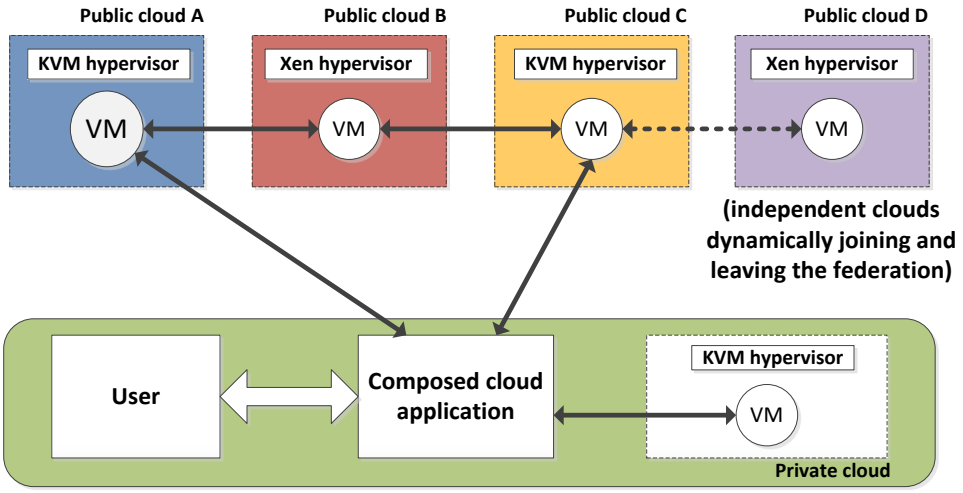


Figure 2.8 Cloud federation model.

and resources between governmental departments that each manages their own Cloud infrastructure.

Federated Cloud: Federated Cloud is an emerging deployment model defined as two or more independent Cloud providers that are capable of sharing resources and are "*able to scale applications across multiple domains to meet QoS targets of Cloud customers*" [10]. This differs from the definition of a community Cloud in that each system environment is not under the control of a central entity or administrative control [47], and are typically heterogeneous in terms of service and technologies as shown in Figure 2.8. Such service models are advantageous for consumers who desire to migrate workload based on desired consumer QoS [48], or a provider requiring additional computing resources [49].

2.2.7 Workload in Cloud

As discussed previously, the characteristics of Cloud computing result in highly dynamic and heterogeneous environments which have enabled a paradigm shift allowing consumers to dynamically request and utilise computational resources and services in order to pursue different business and QoS objectives. These resources and services are utilised through the concept of workload. We define workload as:

"*The amount of work assigned to, or done by, a client, workgroup, server, or system in a given time period.*" [50]

Within the context of Cloud computing, workloads are composed of two components; tasks and users. A *task* is defined as the basic unit of computation

assigned or performed in the Cloud and a *user* is defined as the actor responsible for creating and configuring the volume of tasks to be computed. Tasks are submitted by users via the Internet which are then executed within the Cloud computing environment. A workload can also be composed of multiple related tasks working towards a common objective known as a *job*.

Workload behaviour can be identified and characterized based on their attributes and properties; such properties include the execution length as well as the amount and type of resource utilized. Furthermore, workloads can be characterized by the constraints which limit where the task can be executed. Such constraints include requiring a specific server hardware architecture, or geographical location due to security and privacy constraints [51]

Table 2.1 Classification of Typical Cloud Workloads.

Workload Type	**Description/Examples**
Enterprise Resource Planning (ERP)	Business management software used for product planning, data mining.
WebMail	Typically free email services offered to consumers via a browser.
Storage	Storage services are used for backup and archiving purposes.
Analytics	Business analytics, data mining, temporal and spatial patterns within submitted datasets.
Development/test	Development and testing environments and services for software as well as developed products.
Collaboration	Multiple institutions collaborating together in order to complete project goals.
Gaming	Latency sensitive gaming applications.
Web Applications	eCommerce, java application, web searching, other well developed applications.
Desktop	Desktop computing, desktop management and desktop monitoring services.
Batch Processing	CPU intensive render farms for 3d modelling, visualization of large scale geo-data and performing large numerical calculations.

Workload can be classified by the type and amount of resource consumed such as CPU, Memory, Disk, Network, as well as its function. For example, workload which contain dependencies with each other are 'batch processing', typically used for computationally intensive scientific computing [52][53], while 'latency-sensitive' workload are typically used in real-time applications such as gaming, and financial analysis [54]. Table 2.1 lists common examples of workloads within Cloud environments, identified by IBM [55].

2.2.8 Hardware

Workload is deployed and executed in hardware of Cloud computing systems. As stated in Chapter 2.2.1, Cloud computing environments are typically implemented in datacenters, composed of hundreds or thousands of interconnected servers to provide computing utility. The servers in a Cloud datacenter are typically heterogeneous in nature, as different architectures are better suited to executing different types of workload submitted by consumers, or a workload requires a specific hardware architecture such as GPU [56].

At the fundamental physical infrastructure, there are minor differences between the infrastructure architecture between Cloud datacenters and other distributed systems such as Grids and Cluster systems, which are also interconnected network systems which are capable of leveraging technologies such as virtualization. Virtualization is a key technology which enables Cloud computing characteristics such as elasticity, scalability, resiliency and multi-tenancy [57].

2.2.9 Virtualization

As mentioned previously, a key technology used in many implementations of Cloud computing is virtualization. Virtualization is broadly defined as

"*separation of a service requested from the physical delivery of that service.*" [58]

and specifically defined as

"*the creation of substitutes for real resources, that is, substitutes that have the same functions and external interfaces, but differ in attributes such as size, performance and cost.*" [59]

These substitutes are known as virtual resources that are capable of creating multiple emulated computing environments defined as a Virtual Machine (VMs) on a single physical server known as the host.

Virtualization is typically used for consolidating hardware resources such as CPU, Memory, Disk and Network resulting in reduced hardware costs. These VMs are controlled through the use of a *Virtual Machine Manager* (VMM) (also known as a *Hypervisor*) which is responsible for the creation, deletion, migration and monitoring of the VM status [11]. Interactions between VMs and the underlying hardware are performed through the use of virtualization software using programs known as system calls.

There are several widely used techniques for virtualization: Full Virtualization, Para-virtualization and Operating System-level Virtualization.

Full Virtualization provides total abstraction from the underlying physical resources of the hardware in order to encapsulate VMs. Specifically, there is no modification made to the Operating Systems (OSs) and the VM which are unaware that it is a virtualized environment and that it shares the same physical infrastructure with other VMs. The VMM is responsible for providing access and allocation of resources between VMs and the physical hardware through the use of dispatching and paging [60]. Applications can be deployed within the platform of each VM, which are executed above the VMM as shown in Figure 2.9.

The concept of *para-virtualization* is similar to that of full virtualization, however the difference is that the OS is modified so that it is aware it is within a

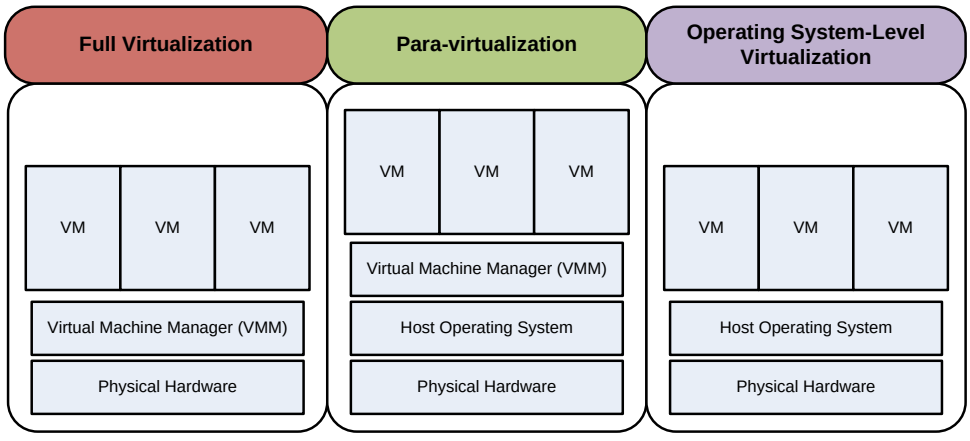


Figure 2.9 Virtualization techniques.

virtualized environment and consequently aware of resource demands from other VMs within the same physical server, resulting in less virtualization overhead. However, as OSs within the VMs require modification, this results in limitations to its compatibility and portability as it requires configuration and modification to run in different server architectures.

Operating System-Level Virtualization results in the host OS instead of the hardware being virtualized, resulting in multiple isolated user-space instances that share an identical OS. This allows system administrators the ability to assign resources upon VM creation as well as change them dynamically at runtime. It is also known as Single Kernel Image or container-based virtualization. This approach offers minimal overheard, however requires all VMs within the host to share a homogenous OS. A popular implementation of this type of virtualization is Linux containers (LXC) [61] which allows multiple isolated emulated Linux systems on a single host. Other implementations of Operating System-Level virtualization include OpenVZ [62] and HP-UX [63].

2.2.10 Cloud Computing Quality of Service

An important concept in the Cloud computing model is the ability for Cloud providers to guarantee negotiated levels of QoS through the use of SLAs (discussed in Chapter 2.1.5) to consumers.

Typical parameters of Cloud SLAs include availability, service performance, monitoring and service cost, security and reliability [8][65]. QoS parameters used by Cloud consumers are dependent on their business objectives, and more specifically the characteristics of the workload. For example, consumers whose workload consists of a database application might require high availability and geographical QoS constraints while a real-time application might emphasize high availability and a boundary on acceptable response time.

2.2.11 Differences between Cloud and Previous Distributed Systems

Cloud computing shares a number of its characteristics described in Chapter 2.2.2 with previous distributed paradigms; and there has been a sizable amount of work that discusses their differences, most notably Grid computing [66][67][68]. As discussed in Foster, et al. [66], Cloud shares similar system characteristics with service-oriented systems such as Grids which use similar

technologies such as virtualization in order to scale VMs. The primary difference between these two computing systems is the service model described in Chapter 2.1.5. Grid systems are typically more project-oriented with a number of institutes forming Virtual Organisations in order to complete mutually beneficial objectives; computing services and resources are provided as units to be spent, allowing the provider to manage scheduled resources and available allotted time prior to workload execution. In comparison, Cloud consumers are not required to "book" resources and are free to request resources on an ad-hoc basis, making it is possible for a consumer to request a large amount of computing resources without prior knowledge for the Cloud provider [66].

Moreover, due to these different system objectives, the type of workload applications deployed tend to also vary; Grid systems typically use scientific and computationally heavy jobs in order to solve research problems or further project goals [66]. In comparison, Clouds (and in particular Public Cloud datacenters) contain many different types of consumers pursuing varied business objectives not just limited to collaborative projects and scientific computing, but also workload applications as described previously and shown in Table 2.1.

Furthermore, Cloud datacenters are business critical systems which require high-assurance to provision service to potentially millions of consumers. As a result, it is necessary to draw attention towards a major property of software systems: dependability.

2.3 Cloud Dependability

2.3.1 Faults, Errors and Failures

Before the concept of dependability is discussed in detail, it is necessary to define the concepts of faults, errors and failures and their relationship to the state of a software system.

The *behaviour* of a system (as discussed in Chapter 2.1.1) is defined as its implementation in order to perform its function, and can be described as a sequence of states. The *total state* of a system is composed of five states: *Computation*, *communication*, *stored information*, *interconnection* and *physical condition* [5]. The *external state* of a system is any state which can be perceived at the service interface, while the remainder is the *internal state*.

Correct service is delivered when the service implements the system function, however it is possible for the system to experience service failure (abbreviated to failure) if the service deviates from correct service. Service failure within a system is a result of noncompliance with the functional specification of the service, or the specification does not describe adequately the system function.

Within normal system operation conditions, the delivered system service is the sequence of external state, and the system advances from one valid internal state to the next by means of valid transition. However, it is possible for this internal state to transition to an invalid state, known as an *error*. Errors occur due to presence of *faults* within the system. In this context, a fault is defined as "*the adjudged or hypothesized cause of an error*" [5]. Faults occur due to vulnerabilities that exist within the internal or external state; for example, a fault in the internal state of the system enables an external fault to damage the system. An error that results in the external state of the system becoming invalid causes a *failure*, which results in the system deviating from correct service as perceived by the user. While not all faults and errors within the system result in errors and failures, respectively all errors and consequentially all failures are the result of faults within the system. In the context of the system environment, the occurrence of a failure within a component as shown in Figure 2.10 results in a

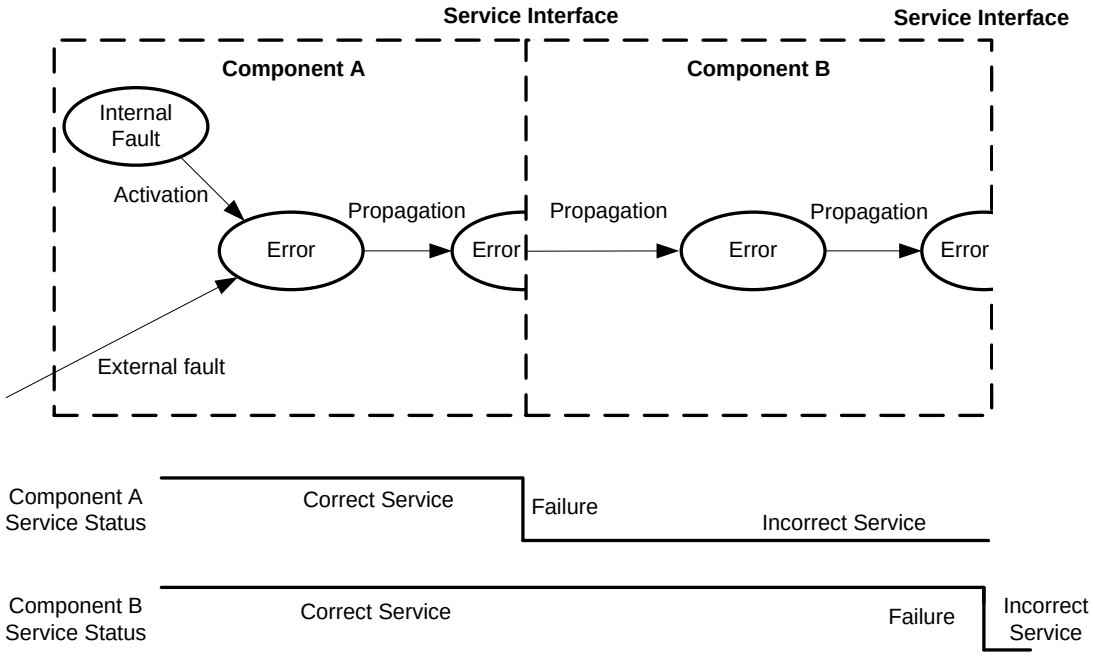


Figure 2.10 Fault, error and failure propagation between system components [5].

fault to become activated in other components within the system as well as components which interface with the failed component. As a result, a failure of a component becomes a fault for the components which interact with it.

2.3.2 Dependability

The term dependability is a key concept in provisioning software systems. Dependability in [69] is defined as "*that property of a computer system such that reliance can justifiably be placed on the service it delivers. The service delivered by a system is its behaviour as it is perceived by its users*." and similarly in [5], as "*a level of trust to be justifiably assigned to a system for it to avoid failures*".

Dependability is a global concept, and is subsumed by three main elements; attributes, means and threats as shown in Figure 2.11.

The *attributes* of dependability allow the properties expected from a system to be expressed, and are used to support the assessment of system quality [5][70].

Availability: The degree which a system or component is accessible when required for use and is typically expressed as a probability.

Reliability: The ability of a system to perform its system function under specified period of time, and is often expressed in time.

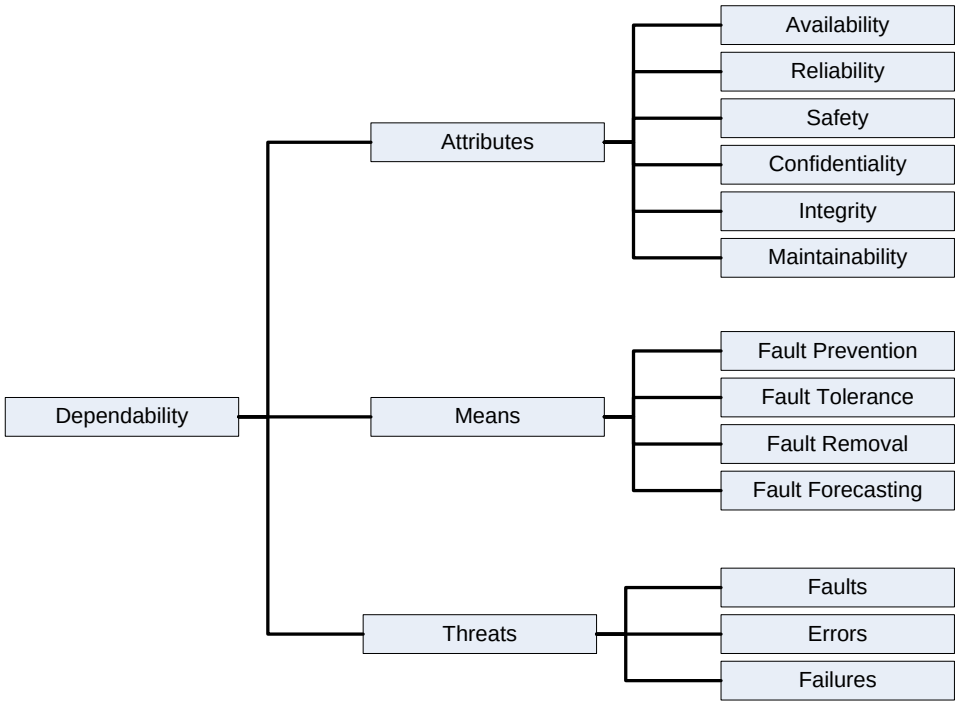


Figure 2.11 Dependability Tree.

Safety: The ability for a system to avoid catastrophic consequences on system users and the system environment.

Integrity: The degree which a system to run without alterations to correct service caused by unauthorized alterations being made to the system.

Maintainability: The ability for a system to undergo modification to correct faults, improve other attributes or adapt to changes within the system environment.

It is worth noting that reliability is not a synonym for dependability; rather, reliability is just one attribute of the overall concept. [5] distinguishes attributes of interest between the dependability and security research communities as shown in Figure 2.12. The attributes of dependability are included within the *dependability specification* of a system, which contains the requirements for the accepted frequency and severity of failures. It also includes the classes of faults which will be covered within the system. Furthermore, the specification only includes attributes which are of importance, resulting in some attributes to be more emphasized than others, or in some cases omitted.

The *means* of dependability are the methods by which the attributes of dependability are attained. They are classified as fault prevention, fault tolerance, fault removal and fault forecasting. The objective of fault prevention and tolerance is to deliver service which can be trusted, while fault removal and forecasting aim to obtain confidence in that ability.

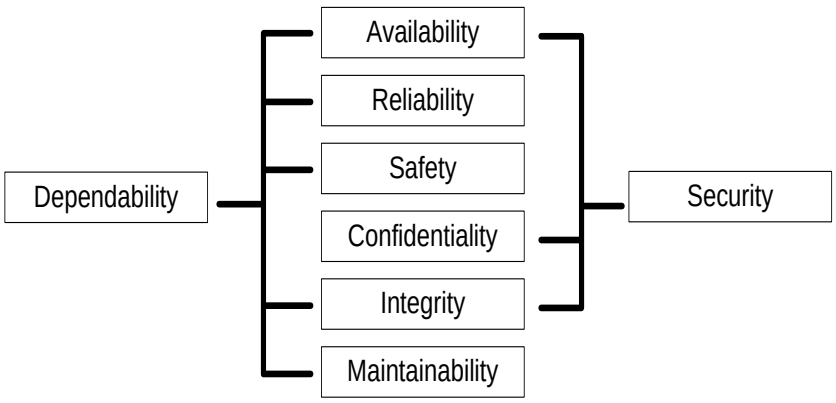


Figure 2.12 Dependability and Security attributes.

The *threats* to dependability aim to identify and classify threats to the system service. A failure within a system can manifest in a variety of different forms and characteristics (known as *failure modes*); ranging in terms of severity (minor to catastrophic), domain (content and timing) and consistency (consistent and inconsistent).

One way to classify faults in computing systems is to study behaviour upon failure. This classification defines what assumptions can be made about the behaviour of failed components. Failures are classified in [71][72][73] as:

Crash fault: The component proceeds to halt and to lose its internal state, resulting in a fail-stop failure.

Omission fault: The component does not to respond to a number of inputs.

Timing fault: The fault causes the component to deliver service too early or too late [74], resulting in either an early-timing or late-timing failure.

Byzantine fault: Faults that result in a component to behave arbitrarily. Faults can be caused from malicious attacks, operator errors, or software errors [75].

The above four faults form a hierarchy as shown in Figure 2.13, with crash and Byzantine being the simplest and most complex to identify and analyze, respectively. For crash faults, this is due to other components in the system are capable of identifying when a crash failure has occurred. For Byzantine faults however, different components may not only perceive whether the component is provisioning correct service or not, but may be exhibiting different types of faults such as crash, omission and timing [71].

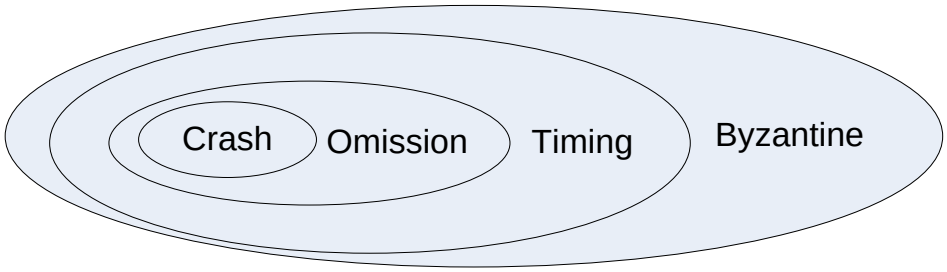


Figure 2.13 Fault Hierarchy.

Different types of faults are assumed for the nature of system component of interest. For example, a processor is frequently assumed to exhibit crash or Byzantine faults (in that either the processor stops executing, or no assumption is made about the nature of its failure). For communication networks, all type of faults can be assumed (crash faults if a message is not delivered, omission faults for lost messages, timing fault if there is a long delay for deliverance, etc.).

Faults result in the manifestation of different failure modes, which can be used to characterize failures in software systems from four perspectives as shown in Figure 2.14: Domain, Detectability, Consistency and Consequence.

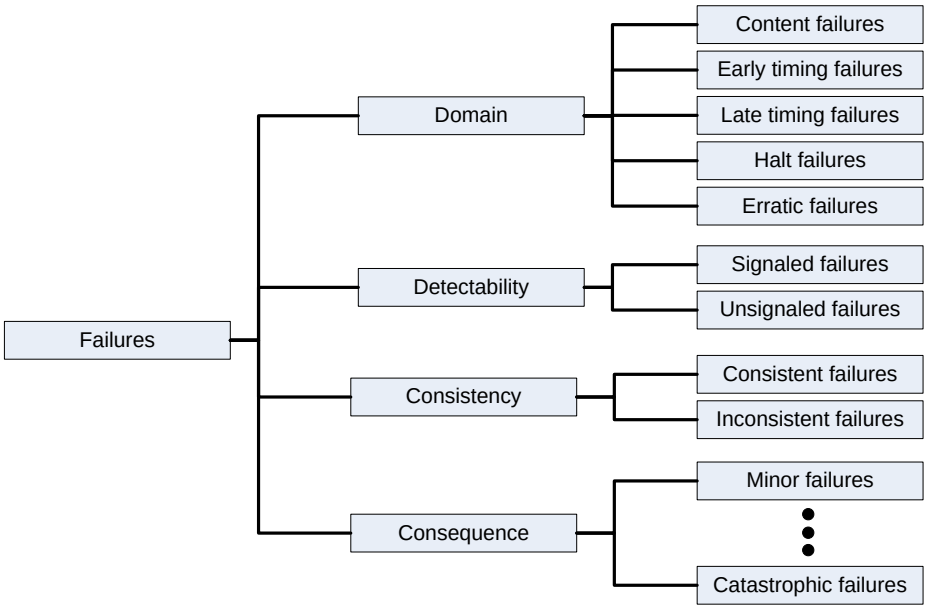


Figure 2.14 Failure characterization tree.

2.3.3 The Need for Dependability in Cloud Datacenters

Dependability, which is a fundamental property of computing systems, is a key concern in Cloud computing for the following reasons:

There are potentially great economic consequences for failure occurrence. Cloud environments are typically deployed in large-scale datacenters. As a result, there are potentially great economic consequence for any failures [78], ranging from minor to catastrophic failures. For minor failures, this results in systems either unable to fulfil QoS or requiring additional resources attempting to do so through means such as fault-tolerance and fault-recovery. For catastrophic failures, it has been reported by the International Group on Cloud Computing Resiliency (IWGCR) that the total downtime for 13 large-scale Cloud providers since 2007

equalled 569 hours, with an economic impact of approximately $71.7 million dollars [76].

Failures are becoming increasingly common and are now the norm, rather than the exception due to the large-scale and complexity of many Clouds datacenters [27][77]. As mentioned previously in Chapter 2.1.5, the system size and scale of state-of-the-art Cloud datacenters are growing by 7% and 17% per annum in terms of number of datacenter facilities and server racks, respectively. As of 2011, Amazon EC2, a popular large-scale Public Cloud, contained more than 449 billion objects and processed up to 290,000 requests per second at peak time [79]; it has been well documented in [1][80][81][82] that increased complexity and size of systems results in an increased number of failures to occur in both software and hardware.

Diversity of workload characteristics and behaviour result in different types of faults to activate. Cloud consumers have varied business objectives and QoS expectations. As a result, consumers will not only use different types of workloads such as Scientific Batch Processing, Latency Sensitive, Gaming, Databases, Video Streaming, etc. they can also use the same type of workload in different ways; this leads to different utilization patterns of resources as well as workload size and workload utilization intensity. It has been identified in past systems that the workload size and type of workload properties such as and computationally and I/O intensive workloads influence the types of faults that are activated [83][84]. Within the Cloud environment, due to the volume and diversity of workload resource utilization, there are a large variety of different types of faults with the potential to become activated.

A challenge for Cloud providers is deciding the degree of fault assumption coverage and failure mode coverage for the system. In this context, coverage is defined as the assumption of the types of faults and failures which occur within the system, and is typically used as a means to measure the effectiveness of fault-tolerance within a system [72]. Developing a system that is capable of large failure coverage results in substantial resource and development costs. Furthermore, depending on the frequency and severity of the failure, it is often viable to only consider a limited scope of failure coverage. For example Byzantine Failures, while more complicated and difficult to correct than crash-stop failures,

may be omitted from the fault specification due to complexity in fault identification and cost in terms of development and resources required for fault tolerance and correction [72].

Due to the large-scale and complexity of Cloud environments, it is inevitable that it is may not be possible to cover all types of faults and failures that occur. The authors of [85] state that building highly reliable and available Clouds is a "critical, challenging and urgently-required research problem" due to the large number of interacting components and their inherit complexity. Furthermore, due to the complexity and large-scale system size, analysis of the reliability of Cloud is difficult and challenging [86]. As a result, there is an urgent need to understanding how to build dependable Cloud environments.

2.3.4 Current Cloud Dependability Research

Research in Cloud computing is currently very active, with a number of initiatives to improve Cloud dependability by the development of practical mechanisms. A large body of current research is focused on well-defined areas. For example, improving Cloud availability and reliability by means of enhancing fault-tolerance in Cloud computing. A typical approach is to leverage the capability of Cloud elasticity in order to create a suitable number of VM replicas based on consumer QoS to create acceptable levels of fault-tolerance [87][88]. These works calculate the optimal number of replicas based on workload reliability [89] as well as additional parameters such as component usage [85], availability [90] and real-time constraints [91][92].

In addition, there are a wide variety of other areas which have received attention:

- Provisioning dependable storage within Cloud systems and developing solutions to improve the reliability of storage using a number of techniques including deduplication [93] and federated storage [94].
- Resource-provisioning which attempts to schedule workloads based on execution time and failure-characteristics of servers [95][96][97].
- Provisioning of federated Cloud workloads across multiple Cloud providers as a means to increase reliability [98], reduce migration time [48] and reduce costs [99].

The relevance and effectiveness of these works rely on whether the selected research assumptions and system parameters used for evaluation are correct. Unrealistic assumptions can hamper our understanding of Cloud computing environments and how their characteristics impact the physical environment. Furthermore, it can result in developing mechanisms which assume unrealistic environmental operation such as failure coverage, user behaviour and performance, ultimately resulting in reduce effectiveness when deployed within real Cloud datacenters. Correct assumptions and parameters can be derived from real large-scale production Cloud environments which is achieved through empirical analysis and modelling in order to study and quantify system characteristics and behavioural patterns.

2.4 Cloud Analytics

2.4.1 Systems Analysis

Studying computing systems is vital in order to further our understanding of system characteristics and behaviour, as well as create a set of assumptions about the operational behaviour and relationships between components. Such assumptions are typically created through either mathematical or logical relationships, forming a *model.* Models are used in order to understand how a system and its respective components behave [100]. In order to develop and deploy practical mechanisms for computing systems it is desirable to evaluate and validate their efficiency under different system environment states. There are a number of methods to study a system, as presented in Figure 2.15.

One of the most effective ways to study a system is to alter the physical system environment and monitor the new conditions of operation. This is particularly effective when validating the effectiveness of developed mechanisms, as they can simply be deployed and evaluated within the actual physical system. However, in many cases it is not feasible to perform such experiments as it could result in costly disruption to system operation.

As a result, it is more feasible to build and study a model that is representative of the system instead of using the actual physical system. Such models require validation to conclude whether the model accurately reflects the system for the particular study objective [101]. Model validation is defined as "*substantiation*

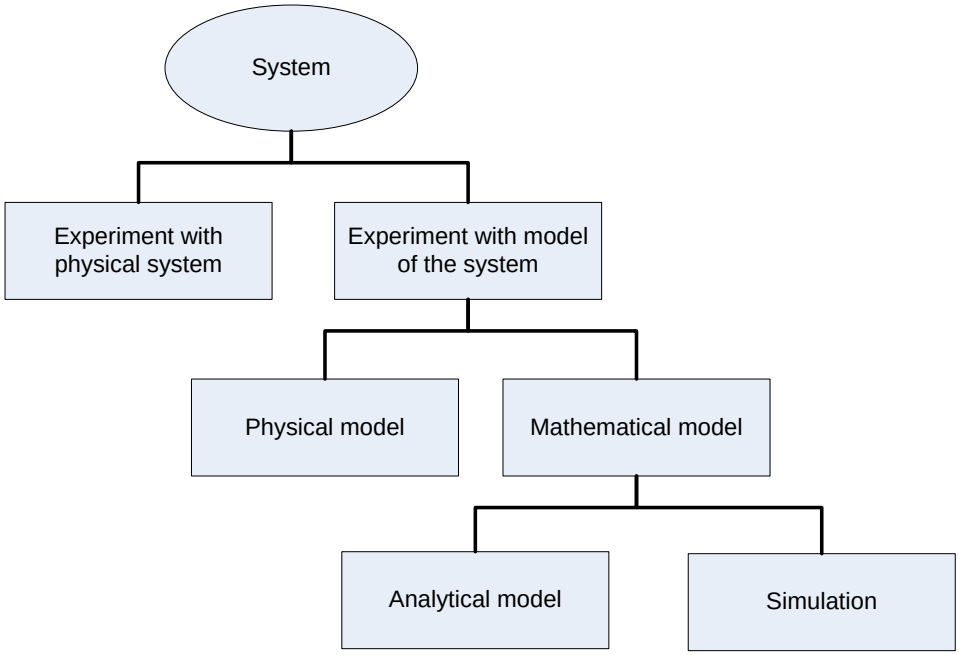


Figure 2.15 Methods of studying systems [100].

that a computerized model with its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model" [102]. In order to create models that accurately represent the physical system, it is necessary to analyze the physical system in order to produce system parameters which are used by the developed model.

Due to the large-scale and inherit complexity of modern day systems due to the number of component interactions, simulation is a favoured approach when studying behaviour and alterations to the system operation; such alterations can range from making components fail to study the impact on consumer QoS to modifying resource management algorithms to study changes in system resource utilization.

Analysis and simulation of Cloud environments significantly benefits both providers and researchers, as it enables a more in-depth understanding of the entire system as well as offering a practical way to improve datacenter functionality. For Cloud providers, it enables more in-depth understanding of their system environments, identifies areas of operational inefficiency, and enables a method to enhance a number of mechanisms to increase the productivity and QoS of their systems. For example, exploiting task heterogeneity to reduce contention for physical resources in servers or analyzing the correlation of failures to resource consumption. For researchers, analysis allows for more

realistic system assumptions for Cloud computing environments, enabling evaluation of theoretical mechanisms which are supported by empirical findings. Specifically, it supports researchers and providers in further understanding the actual status and conditions of the Cloud system and identifying Key Performance Indicators (KPI) necessary to improve operational parameters.

An important component of systems analysis is data analysis; used in order to study the system and conduct the analysis and simulation described above. There are a number of approaches for data analysis, which can be seen more as philosophies, as opposed to a set of techniques.

2.4.2 Data Analysis

The approach of data analysis used is dependent on the nature of the data being analysed, and whether there exist appropriate models for the data source [136]. There are a number of different approaches including Frequentist [137], Bayesian [138][139], and Exploratory Data Analysis (EDA) [140]. Each of these approaches follow different methods in regards to analysing and modelling a data population (For example, Bayesian models the data based on prior postulation using domain expertise and compares the theorized model against that of the true model, while EDA first performs the analysis and then models the system based on the prior analysis). Each approach offers advantages when considering the nature of the data, availability of accurately modelling the data, data size and degrees of statistical rigour required. In application however, analysts use such approaches freely dependant on the nature of the problem intuitively.

Moreover, there are a large variety of techniques which can be used in data analysis depending on the requirements and research objectives. These include classification [136], distribution modelling [141], correlation analysis [142], and other statistical analysis techniques. In the context of Cloud computing, data of interest for systems analysis includes operational trace logs which record resource usage and event occurrence of workload, servers and users. We describe two examples of statistical analysis techniques important within Cloud computing as well as other distributed systems which can be applied to identify component relationships and model system behaviour; Classification and Distribution Modelling.

2.4.3 Classification

In order to study the characteristics and behaviour of Cloud computing systems, it is advantageous to investigate whether it is possible to classify components based on their respective behaviour within the system. A practical example of this would be investigating how the behaviour of tasks (in terms of resource utilization and execution length) is affected by specific types of users. This leads to the ability of being able to study the distinct characteristics of users and tasks within a population and provide greater insight into their behaviour as well as quantify their impact on different aspects of the system such as server resource utilization and failure characteristics. Furthermore, it allows us to abstract general behaviours of workload, and enables modelling of the relationship between users and tasks.

One such classification technique is performed through the use of clustering. Cluster analysis is "the organization of a collection of patterns (usually represented as a vectors of measurements, or a point in a multidimensional space) into clusters based on similarity" [136]. A unique data point within a cluster pattern (such as a system component) is represented and composed by one or more attributes. Cluster analysis is also advantageous as it provides a visualization of the data, which can be leveraged when investigating specific

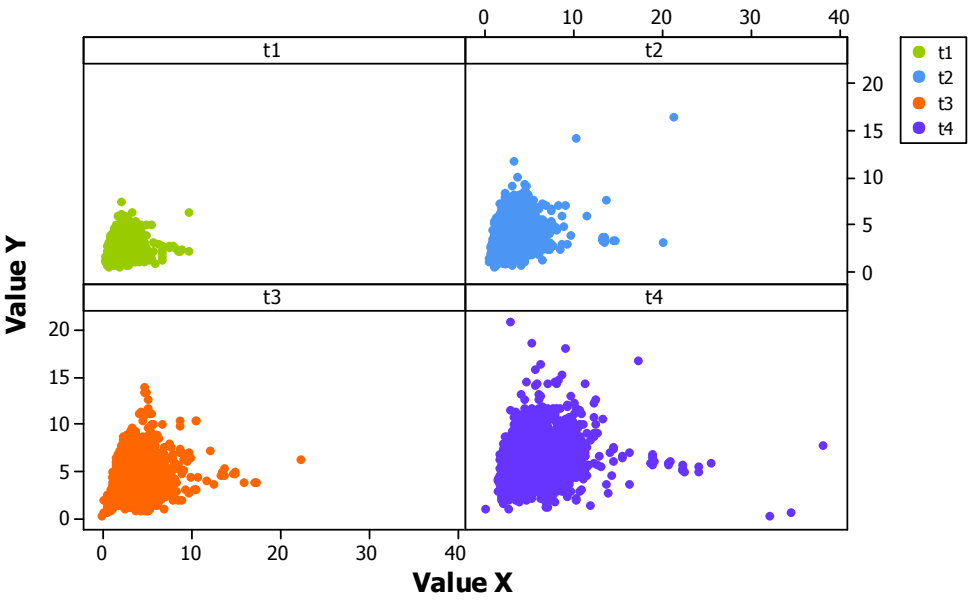


Figure 2.16 Example of Cluster analysis over four time periods *t*.

behaviours of a system over a time period. Figure 2.16 presents an example of this visualization, which depicts the growth of a clustered data population over four time periods *t*. Cluster analysis is used in a number of research domains including exploratory pattern-analysis, grouping and decision making.

2.4.4 Distribution Modelling

Another critical aspect of system analysis is modelling the behaviour of the system. Specifically, modelling the probability that a certain type of behaviour will occur. This behaviour can be represented as a value that is either continuous (a state transition within the system) or discrete (CPU resource utilization). Systems exhibit different degrees of stochasticity, which can be modelled using a probability distribution that calculates the probability of a specific measurement or value occurring within a component of the system [100].

There are different family types of probability distributions which each follow different shapes due to their respective parameters. Continuous distributions such as Weibull, Lognormal and Gamma are better suited for heterogeneous data populations, while examples of discrete distributions including binomial and Poisson are typically suited for categorical or homogenous data populations [100]. It is ideal to attempt to fit the probability of values occurring within a

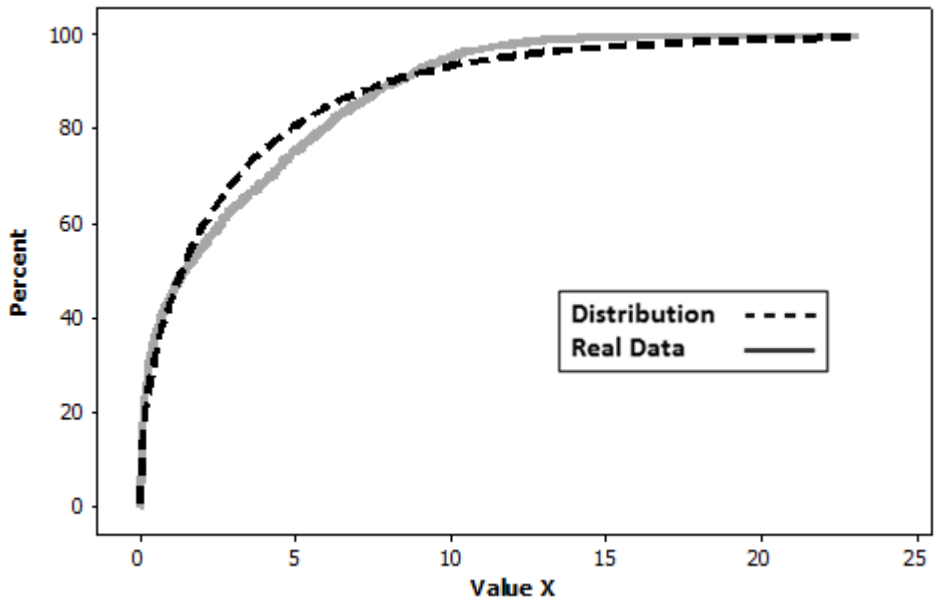


Figure 2.17 Example of distribution fitting of empirical data against a theoretical distribution.

system to a theoretical distribution; however in practise it is more common to develop an empirical distribution which fits the probability of a value occurring with an empirical data population in comparison to a theoretical distribution as shown in Figure 2.17. Distribution modelling has a number of applications in computing systems, including studying performance, probability of failures [1] and submission rates of users [143]. Distributions follow different parameters, which are used as inputs into a simulator when simulating behaviour of computing systems.

Attention is now drawn to application of analysis techniques described in this section to Cloud computing. These techniques can be used in order to greatly enhance the understanding of Cloud component behaviour and consequently Cloud datacenter system operation. Specifically, there are four domains of Cloud research which can be enhanced by the empirical analysis of Cloud datacenters: workload characterization, failure-analysis, server operation and system operational waste.

2.4.5 Workload Characterization in Cloud Computing

As discussed in Chapter 2.2.7, a key component within Cloud datacenters is workload. Specifically, users request computing services and resources for a task which is executed within a server. Analysis of workload characteristics and behaviour is critical when studying its affect on the performance and productivity of the overall system; such analysis allows the study of behavioural patterns of user submissions and task resource utilization over different time periods. Furthermore, it enables classification of tasks and users to study the relationship between user submissions and task resource utilization patterns at different time intervals and server architectures within the system.

Moreover, such analysis is required in order to model simulation parameters of the workload. Such models can be used in a number of research domains including resource optimization, security, dependability and energy-efficiency. In order to produce such realistic models, it is critical to derive their components and parameters from real-world production trace logs. Realistic workload models enable the simulation of Cloud environments whilst being able to control selected variables in order to study emergent system-wide behaviour, as well as support the estimation of accurate forecasting under dynamic system conditions

to improve QoS offered to users. Such modelling supports the enhancement of Cloud Management Systems (CMSs) as it allows providers to experiment with hypothetical scenarios and assess their decisions as a result of changes within the Cloud environment (i.e. Capacity planning for increased system size and users, alteration of the workload scheduling algorithm, performance tradeoffs, and service pricing models).

Wang, et al. [103] present an approach to characterize the workloads of Cloud Computing Hadoop ecosystems, based on an analysis of the first version of the Google tracelog [104]. The main objective of this work was to present coarse-grain statistical data of jobs and tasks in order to classify them by execution duration. Furthermore, they synthesized job submissions and execution duration which was used to evaluate design alterations to the Hadoop environment. The first case study focused on the evaluation of job scheduling of Hadoop tasks; the authors discovered that scheduling tasks based on the locality of task execution and data storage yields better overall performance. The second case study focused on studying how attaching Network-Attached Storage (NAS) affects the performance impact of Hadoop applications deployed within the cluster. A number of applications were chosen for evaluation including Terasort, Computationally heavy workloads, indexing and searching. The authors concluded that computationally heavy applications perform better when using NAS in comparison to locally attached storage. They concluded that for applications which replicate output data, NAS gives a substantial performance boost in comparison to local storage. This workload characterization focused on studying timing problems making it unsuitable for analyzing other Cloud computing issues related to resource usage patterns. Additionally, the analysis focused solely on tasks and ignores user behaviour, a crucial component in Cloud workload.

Zhang, et al. [105] presents a study to evaluate whether mean values for task waiting time, CPU, Memory, and disk consumption are suitable to accurately represent the performance characteristics of real system traces. The data used in their study is not publicly available and consists of the historical traces of 6 Google compute clusters spanning 5 days of operation. The experiments conducted suggest that mean values of run-time task resource consumption are a

promising way to describe overall task resource usage. However, it does not describe how the boundaries for task classification were defined.

Mishra, et al. [106] describes an approach to develop Cloud computing workload classifications based on task resource consumption patterns. The trace log analyzed is composed of 5 production Google compute clusters spanning 4 days. The proposed approach identifies workload characteristics, constructs the task classification, identifies the qualitative boundaries of each cluster and reduces the number of clusters by merging adjacent clusters. The authors present a number of observations from the analysis; first, tasks characteristics are primarily divided into two types: short-running and long-running tasks representing user-facing and compute intensive tasks, respectively. Furthermore, a small number of long-running tasks consume a large proportion of CPU and memory within a server. The approach presented is useful to create the classification of tasks, but does not perform an analysis of the characteristics of the formed clusters in order to derive a detailed workload model.

Kuvulya, et al. [107] present a statistical analysis of MapReduce traces. The analysis is based on ten months of MapReduce logs from the M45 supercomputing cluster [108]. Here, the authors present a set of coarse-grain statistical characteristics of the data related to resource utilization, job patterns, and source of failures. They discovered that users tend to execute the same type

Table 2.2 Comparison of Cloud Workload Analysis Research

Author	Trace size	Analysis Method	Analyzed Component	Analyzed Parameters	Workload Model & Validation
Wang [103]	7 Hours	Coarse-grain	Task	Task duration	No
Zhang [105]	30 Days *5 day sample*	Coarse-grain	Task	Task resource usage	No (parameters partially given)
Mishra [106]	4 Days	Cluster centroids	Task	Task resource usage	No
Kavulya [107]	10 Months	Coarse-grain	Task	Task duration	Yes (parameters given)
Aggarwal [109]	24 Hours	Cluster centroids	Task	Task disk usage	No

of job repeatedly, enabling accurate prediction of job completion times. Furthermore, 95% of successful jobs completed under 20 minutes and modelled using a Lognormal distribution. They compare two algorithms weighted by distance and locality, respectively for job prediction completion times. This work provides a detailed description of the distributions modelled for job completion times, but only provides very general information about the resource consumption and user behavioural patterns. Similar to [103], this characteristic limits the proposed approach mainly to the study of timing problems.

Aggarwal, et al. [109] describe an approach to characterize Hadoop jobs. The analysis was performed on a dataset spanning 24 hours from a Yahoo! production cluster comprised of 11,686 jobs. This dataset features attributes generated by the Hadoop framework such as execution time and Disk I/O. The main objective of this work is to group jobs with similar characteristics using clustering and then analyze the resultant centroids, enabling them to discover eight types of jobs. Furthermore, they present a comparison of actual production jobs and synthesized GridMix3 jobs. This work only focuses on the usage of the storage system, neglecting other critical resources such as CPU and Memory.

It is clear from the analysis of related work as shown in Table 2.2 that there are few production tracelogs available to analyze workload patterns in Cloud environments. Previous analyses contain three main gaps that need to be addressed in order to achieve greater insight into the impact of Cloud computing characteristics on the operational environment, as well as developing more realistic workload models. First, the analyzed system must be large-scale, with a sufficient operation time period more than a few days; failure to do so neglects the study of longer running tasks resulting in misinterpreting workload behaviour. Second, analyses need to explore more than coarse-grain statistics and cluster centroids; to capture the patterns of clustered tasks it is also necessary to analyze the parameters and trends of each respective cluster. Although previous approaches offer some insight into workload characteristics, they do not provide a structured model which can be used for conducting simulations. Finally, workload is always driven by the users, therefore analysis and realistic workload models must include user behavioural patterns linked to tasks in order to study the impact of user behaviour on the Cloud environment in terms of resource

utilization, task submission rates and resources requested. The approaches previously described completely focus on tasks, neglecting the impact of user behaviour on the overall environment workload.

Tasks submitted by users are executed on multi-tenant servers, which compose the physical infrastructure of the Cloud environment. As a result, another important component of Cloud computing is characterizing the physical servers in order to understand resource management of tasks and quantify the efficiency of resource utilization within a system.

2.4.6 Servers

Liu [129] present a technique to measure server CPU utilization and conducted a study of CPU utilization from two public Cloud providers; Amazon EC2 [130] and GoGrid [131] by submitting VMs into the Cloud environment and probing the network to discover whether VMs are scheduled onto the same physical server. Their results show that the average CPU utilization across 20 servers running m1.small VMs [132] over a week was 7.3%, with an average individual server utilization between 3.6% - 16.9%. Servers follow a diurnal pattern, with an average utilization between 8.09% and 6.76% between 08:00-20:00 and 20:00-08:00, respectively. Furthermore, utilization in "daytime" experiences not just higher utilization on average, but also higher utilization peaks compared to "night-time" - between 20-30% for short time periods. For computation heavy VMs, the average utilization across 10 servers is 7.8%, with individual server utilization between 4.15% - 16.6%. Finally, the author studied the utilization of servers on GoGrid using 10 small VMs across 10 servers, with average server CPU utilization at 2.2 - 4.5%. Although this work studies the utilization patterns of different VM types across two Cloud providers, it is limited to only studying a small number of servers which may not accurately represent the larger Cloud environment, composed of thousands of servers. Furthermore, while CPU utilization is an important value in studying server characteristics as highlighted in the work's thermal modelling, other resources such as disk, network and memory are also key parameters when characterizing and modelling severs within Cloud datacenters.

Kuvulya, et al. [107] presents the statistical properties of server utilization for MapReduce traces. The analysis is conducted on ten months worth of MapReduce logs from the M45 supercomputing cluster [108] composed of 400

servers, and includes the study of CPU, network and disk utilization. They observed that the average CPU resource utilization of the cluster rose within a 12 month period from 5-10%, and that network and disk usage increased by a factor of three. These changes are postulated to be the result of an upgrade to the map outputs of Hadoop, as well as changes from workload submitted by users although they observe that the cluster still operated below peak capacity. A limitation of this work is that the server utilization characterization and analysis does not explore more fine-grained characteristics and the degree of server architecture heterogeneity for the M45 cluster is unclear.

Birke, et al. [146] present the resource utilization statistics of several thousand servers from production datacenters between June 2009 - May 2011. These datacenters deploy a wide variety of workloads and pursue varying business objectives including Cloud platforms, banking and retail. From their analysis, the authors observe that enterprises and countries exhibit different utilization levels for utilization of CPU, memory, disk and Network, with CPU utilization between 13-42%, 7.25-24.91% and 12.15-23.77% for Enterprises, Country and physical or virtualization, respectively. The results from this analysis are used to explore the feasibility of predicting future CPU utilization behaviour within the datacenter. Furthermore, the authors demonstrate how CPU resource utilization is affected by seasonal patterns. A limitation of this work is that although the authors mention that studied datacenters contain varied workload, due to confidentiality reasons there is no breakdown between the type of resource different workloads leveraged, and whether the application type and user patterns impact resource utilization over different time periods. Finally, utilization patterns are taken as an aggregation, with little focus concerning the deviation of utilization between unique servers.

These studies are important to characterize utilization patterns of Cloud environments. However as discussed above, current work come with a number of limitations. Furthermore, Cloud datacenters deviate in terms of users, server architectures and resource utilization patterns due to different deployment and service models. Therefore, it is highly desirable to continuously characterize and analyze server behaviour in large-scale Cloud datacenters from different data sources, which is presently limited.

Both workload and servers are core components in defining and characterizing a physical computing system. As mentioned in Chapter 2.1.1, dependability is a fundamental concept of a system. As a result, it is important to study how, when and why failures manifest in workload and servers within large-scale Cloud computing environments.

2.4.7 Failure Analysis of Large-Scale Distributed Systems

Another research domain of interest is the failure characteristics of Cloud computing datacenters. Dependability research is greatly enhanced by the analysis of failures from real-world systems, as it enables researchers and practitioners to develop and tune highly effective dependability mechanisms based on realistic empirical data for a given domain. Without such analysis, the assumptions of failure characteristics for large-scale Cloud environments may be unrealistic, resulting in developed mechanisms to be less effective.

In order to research and create effective solutions for the problems faced by large-scale Cloud computing environments, it is imperative to analyze data from real-world sources; for example in the dependability field, a large portion of the state-of-the-art relies on statistical properties and accurate modelling of failure and repair characteristics. The failure characteristics of such environments are of particular concern as failures can result in degradation of Quality of Service (QoS), availability, reliability and energy-waste [110] that can ultimately lead to economic loss for both Cloud consumers and providers. In addition to facilitating research, accurate real-world large-scale failure and repair characteristics allow Cloud providers to create concrete failure scenarios to aid in system development and decision making. For example, such scenarios can assist in deciding what type of dependability mechanisms to use, and when/where to apply them in order to enhance system availability, reliability and energy-efficiency.

Cloud computing systems share a number of characteristics with other large-scale distributed paradigms, most notably Grid and HPC systems. As a result, due to the limited empirical analysis of Cloud datacenters, studying other distributed systems is a way to provide general insight into the failure characteristics of Cloud computing.

Schroeder, et al. [1] analyzed the failure data of the Los Alamos National Laboratory, comprising 22 HPC systems totalling to 4,750 and 24,101 servers and processors, respectively over the period of 9 years. They claim that a large body of commonly cited studies for failure analysis are taken from the late 80s to early 90s, and that modern computing systems are significantly different from modern systems [111][112][113]. Furthermore, a large body of system traces used in failure analysis are not made publically available, limiting their usefulness to other researchers. The authors analyzed the statistical properties of the systems for the root cause of failures and failure rates. Their results indicate that average failure rates vary across systems significantly, ranging from 20 to 1000 failures per year, and that the number of failures per system is heavily influenced by the system size as opposed to specific server architectures. In addition, they observe that the majority of server failures occur within a small proportion of the total server population, with 6% of nodes contributing to 20% of total failures. Furthermore, the authors present evidence that there exists positive correlation between the machine failure rate and workload intensity, and that failure times and repair times are well modelled by Weibull and lognormal distributions, respectively. Finally, they analyze the Mean Time Between Failure (MTBF) and Mean Time to Repair (MTTR) of the systems, and indicated that they are best modelled by Gamma or Weibull distributions and lognormal distribution for failure and repair times, respectively.

Liang, et al. [114] present a study of collected RAS (Reliability, Availability and Serviceability) events log from IBM's BlueGene/L system comprised of 128,000 processors over a span of 100 days. The objective is to study and characterize temporal and spatial failure events as well as the correlate fatal and non-fatal events in order to minimize negative impact on system performance. The authors focus on Network failures, application I/O and non-fatal events. They discover that different types of workloads exhibit different failure characteristics; network and application I/O failures occur in periodic bursts with short MTBF, while memory failures appear to be less variable and follows a daily temporal pattern. In addition, they observed that the occurrence of non-fatal events can be used to predict the occurrence of a fatal event, and three runtime failure prediction algorithms were developed and evaluated. However, this work does not explore

failures due to CPU, a key component in characterizing computing system behaviour and resource utilization.

Li, et al. [115] analyze job failures in a large-scale data-intensive Grid composed of 180 active sites and 34,121 servers over three time periods totalling 24 days. The objective was to characterize the interval times and life spans of failed jobs investigating cross-correlative structures and statistical models to fit the failure data and measure its impact on system performance. Their analysis shows that 5-8% of jobs executing on a server failed, while the majority of failures occurred during task scheduling or without commencing execution, ranging from 25-33% for all submitted jobs. Virtual Organizations play a substantial role in the occurrence of failures, with a minority possessing significant influence on the distribution of failures within the systems. The authors also observe that the occurrence of failures in job execution is periodic and follows a heavy tail distribution, and model failure inter-arrival times and life spans using MMPP4 and 2-phase hyper-exponential distributions for jobs that fail after execution starts. Moreover, the failure analysis was used to propose several failure-aware scheduling strategies which leverage the failure models derived from the trace analysis.

Sahoo, et al. [116], present a failure analysis of errors and failures from a cluster of 395 heterogeneous servers over a time period of 364 days. Using the failure event logs for both non-fatal events and critical events, the results demonstrate that failure rates are highly variable. Furthermore, server failure occurrence is non-uniform, with 4% of servers experiencing 70% of recorded failures, and the majority of failures being temporally correlated to specific time periods within a day. They postulate that the reason for this behaviour is due to the nature of the executing workload; with a large portion of hardware failures occurring within the I/O system and occurring on servers that are executing file/database servers [113].

Currently, limited work that has been attempted to empirically analyze and model Cloud failure characteristics:

Fiondella, et al. [117] present an empirical analysis of outages and incidents reported by Cloud providers and news outlets. The work models the growth of

annual incidents occurring in major Cloud datacenter vendors. They observe that outages were caused by a wide range of issues ranging from software, hardware, human error and power cooling systems. Moreover, the paper outlines that there is an urgent need for dependability models in order to quantify the impact of observed failures. The paper offers recommendations on how to reduce Cloud outages through the effective use of monitoring and analysis, fault diagnoses and modelling. While the work provides evidence of increased outages of Cloud datacenters, due to the coarse-grain nature of the dataset analyzed, it does not provide insight into how failures which do not cause complete system outages impact Cloud datacenters.

Vishwanath, et al. [118] present a failure analysis of hardware in large-scale datacenters across an observational period of 14 months. The trace consists of several thousand servers and presents the statistical properties and modelling of failures in disks, memory modules and RAID controllers. They discover that 78% of failures in servers occurred within the hard disk, which was the most commonly replaced component. Furthermore, the work presents hardware failure and repair patterns, and identifies a strong correlation between repairs frequency and the number of disks within a server.

Kavuyla, et al. [107] present workload failure characteristics from a production MapReduce supercomputing cluster, consisting of 171,000 jobs submitted into approximately 400 servers and 4000 processors over a time period of 10 months. They observe that 2.4% of total jobs fail within the system and that a large proportion of jobs fail within 150 seconds after the first task failure. They observe that job submission and failure is best modelled by lognormal distribution, however admit that the selected Goodness of Fit (GoF) test suggests that the distribution selected might not be optimal. Furthermore, the jobs within this system are limited to MapReduce type jobs and do not consider workload repair or server failure characteristics.

From studying the above works, it is clear that there exists a substantial gap in failure analysis of Cloud computing systems; the most obvious being that there are limited failure analysis of actual production Cloud systems. This is critical as such systems exhibit different characteristics compared to other distributed systems as discussed in Chapter 2.2.11. However, such claims need to be studied

and quantified in order to verify the existence and impact of such behaviours. In addition, each work focuses on specific objectives, ranging from modelling MTBF and MTTR, solely studying servers or workload, or presenting the statistical properties of failures. Furthermore, a large body of work fails to present model parameters, which are critical for other researchers when building simulators.

Table 2.3 Comparison of Different Failure Analysis

Authors	System and trace size	Analyzed Components	Analysis presented	Failure Modelling
Schroeder [1] Analysis failure rates/ root cause for failures and correlation between servers and workload intensity	HPC, 9 years	Server, Workload	Root Cause, failure time, repair time	Yes
Liang [114] Characterize temporal and spatial failure events; correlation of fatal and non-fatal events on system performance impact	Super Computer, 100 days	Server, workload	Spatial & temporal , non-fatal events	No
Li [115] Characterize life spans of failed jobs looking for cross-correlation structures and develop failure models to measure system performance impact.	Grid, 24 days	Workload	Failure modelling	Yes
Sahoo [116] Temporal and spatial analysis of error and failures.	Cluster, 364 days	Workload, server	Hazard rate, failure distributions.	Yes (No parameters)
Fiondella [117] Analysis of Cloud outages reported by vendors and news outlets.	Cloud, 9 years	High-level system availability	Statistical, distribution modelling	Yes (No parameters)
Vishwanath [118] Analysis of hardware failures in Cloud datacenters.	Cloud, 14 months	Servers	Temporal & spatial, failure modelling.	Yes
Kavuyla [107] Workload failure characteristics from a production MapReduce supercomputing cluster	Cluster, 10 months	Workload	Temporal & spatial failure modelling.	Yes

As mentioned in Chapter 2.3.3, failures in large-scale systems are now the norm rather than the exception. As a result, it is important to study not only how and when failures occur within Cloud computing environments, but also quantify their impact on system operation and cost in terms of operational waste.

2.4.8 System Operational Waste

Before presenting the related works within this area, it is necessary to define the key concepts of energy consumption, energy-efficiency and their impact in Cloud computing datacenters.

IT infrastructure and operations consume vast amounts of energy. In this context, energy is defined as "*the physical currency used for accomplishing a particular task*" [119]. Energy can be represented in a number of forms, including electrical, mechanical, kinetic, etc. Power is defined as the instantaneous rate of energy use. The energy used for a task is defined as the average power use and the total time taken for its completion, represented as:

$$Energy = Average\ Power \times Time$$

There is a growing trend for large-scale computing systems to not only improve performance and dependability, but also minimize energy requirements for computation [110][27][110]. As systems may vary in terms of workload as well as the user number and physical system size within its lifespan, such a requirement is typically measured in terms of energy-efficiency. Energy-efficiency is defined as the amount of energy required in order to produce an amount of work and is typically expressed as a ratio [120]. Energy is typically described in terms of electrical consumption within the context of computing systems, represented by Watt-hours (Wh) and Watts (W) for energy and power, respectively.

As discussed previously, according to the results reported in the 2012 DCD Intelligence Census Report: Energy [25] datacenters consumed 1.8% of the total global electricity and it is predicted that by 2020, the IT sector will become the world's most energy consuming industry [29]. Furthermore, according to the Environment Protection Agency [121], in 2011 the national energy consumption of datacenters within the United States consumed approximately 100 billion Kilowatt- Hours (KWh), the equivalent of $7.4 billion in its total annual energy bill, indicating significant economic cost nationally.

Datacenter energy consumption is not limited to just server resource utilization; there is a need for additional infrastructure in order to support and sustain the physical operational environment including room cooling, lighting and power supply systems. Figures 2.18(a) and 2.18(b) depict how energy is consumed within a datacenter, extracted from two separate reports [27] and [122]. It can be observed that IT equipment (including processors, power supply, other servers, networking, storage, etc.) constitute 31% and 52% of the total energy consumption, in Figures 2.17(a) and 2.17(b), respectively. Furthermore, it can also be observed that cooling systems contribute a significant amount of energy consumption at 34% and 38%, respectively.

As identified in [123], there are a number of causes for energy waste and energy-inefficiency in datacenters, including over-provisioning [124], inefficient legacy servers [125] and inefficient cooling [126]. Furthermore, a significant amount of energy waste is caused by software and hardware crashes [2][127]. These types of failures caused roll-backed computation of tasks, as well as additional energy overhead when performing rollback through mechanisms such as checkpointing and state recovery.

Recent theoretical analyses and studies have highlighted failures as a source of inefficiencies that increment energy consumption in large-scale datacenters, and have proposed a number of mechanisms to reduce energy waste. Furthermore, a

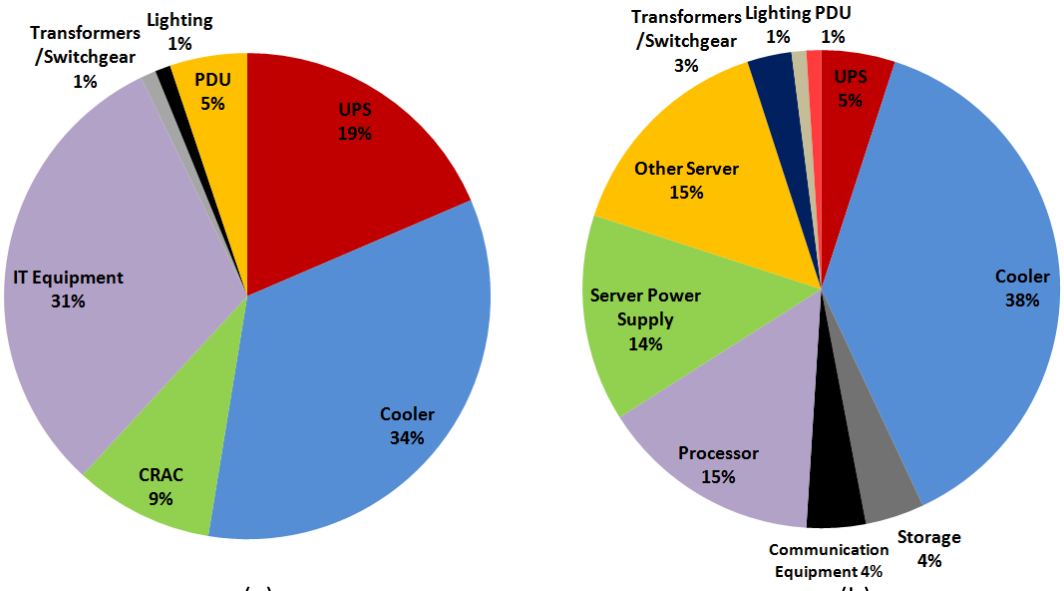


Figure 2.18 Study of datacenter energy usage from (a) DCD Census, (b) Emure Power

number of works have identified failures as a significant contributor to datacenter energy waste, and have proposed a number of solutions on mitigating their impact.

Ropars, et al. [128] highlight the amount of resources wasted with respect to computing power and energy due to computation rolled-back after failure. The main objective of this work is the design of hierarchical rollback-recovery protocols based on a combination of coordinated checkpointing and message logging in order to reduce the waste produced by redundant computation.

Nguyen, et al. [127] discuss the inefficiencies and increment of energy consumption due to failures in large-scale datacenters. The authors approach this energy waste problem by proposing a resource selection scheme to reduce the number of tasks resubmissions that result from failures during their life cycle.

Quiane-Ruiz, et al. [3] discusses the wasted resources and performance impact caused by task failures in MapReduce environments due to automatic rescheduling. The main objective of this work is to reduce the need for re-executing completed map tasks by re-computing intermediate data processed by local reducers and hence do not require migration to another node for processing through the use of fast tracking algorithms.

These works highlight an emerging requirement for Cloud mechanisms to not only improve the reliability of systems, but concurrently also minimize the total energy waste produced by failures. Unfortunately, all of these developed mechanisms rely on analyses that are derived from theoretical data and assumptions which provide no insight into the true characteristics and dimensions of energy waste caused by failures. It is important to validate these mechanisms in real operational environments, as well as identify the amount of energy wasted upon failure.

Reviewing the current state-of-the-art in Cloud analytics for the four respective analysis domains described in the previous subsections, it is clear that there are significant gaps that need to be addressed in order to further enhance the understanding of Cloud computing and its respective system components. However, there are a number of research and technical challenges that need to be overcome before this can be made reality.

2.5 Challenges in Cloud Analytics

2.5.1 Confidentiality and Business Concerns

Presently, there are an extremely limited number of data sources pertaining to large-scale production Cloud environments due to the confidentiality and business concerns of providers. Furthermore, existing trace logs such as those found in [107][146] are predominately released to select research institutions and are not publicly available. This in particular is a challenge to academics, who rely on selected institutes to perform extensive statistical analysis and modelling in order to quantify Cloud computing characteristics to derive realistic assumptions for system behaviour. Such assumptions and characteristics are critical to support evaluation of Cloud mechanism effectiveness and practicality proposed by researchers. Failure to do so can result in unrealistic behaviours and assumptions of Cloud system behaviour, leading to increase ineffectiveness of proposed mechanisms in real production environments.

2.5.2 Characterizing System Behaviour

A critical problem in Cloud analytics is the challenge of comprehensively analyzing and modelling Cloud components; production Clouds contain many different types of components discussed in Chapter 2.2. A typical Cloud datacenter is capable of producing vast quantities of data detailing user requests, failure events and server resource utilization. Furthermore, such data is composed of multiple attributes detailing different aspects of system operation. This results in increased complexity when studying the behaviour of components within the Cloud not only due to the volume of trace log data generated, but also the number of components and their respective interactions within the system.

Furthermore, it is necessary to understand the relationships that exist between components in order to study key aspects of system behaviour, including relationships between user submission patterns and specific types of tasks, server resource utilization inefficiencies and task energy waste generated due to failures. As a result, there is a requirement for computing infrastructure capable of extracting attributes of interest from Cloud trace logs and performing processing for statistical analysis in a timely manner. Unfortunately, such infrastructure may not be accessible to many academic institutions due to

economic cost and lack of sufficient technical knowledge in building large-scale analysis infrastructure.

Comprehensive study and analysis of datacenter trace logs is only feasible through deploying a Database environment and data processing to extract data from the trace log to perform calculations of interest. However, there are three main challenges that cause this process to be more complicated than simply uploading the data from the trace log into a database and then performing basic analytics to derive coarse-grain statistics of the system.

First, there is the challenge of identifying and studying attributes of interest relevant to the analysis objective. This is due to Cloud computing systems containing a large volume of data consisting of many attributes capturing various aspects of system operation. Failure to select relevant attributes can result in additional complexity in Cloud datacenter modelling and computation time required for analysis completion. This is due to increased data volume and more importantly, the potential to study irrelevant attributes which may result in misinterpretation of key system behaviour [140].

The second challenge is identifying and exploiting the relationships that exist between Cloud components. Component behaviour is captured through different system attributes recorded and distributed across trace logs. Trace logs generated by a Cloud datacenter not only contain attributes that capture different aspects of system operation, but can also be presented in different formats and contain little to no context to component relation. This introduces challenges studying system behaviour at a specific time or space (such as a specific observation period or server architecture, respectively). As a result, a significant amount of time and effort is required to understand how attributes relate to each other and the larger system environment. This requires extensive understanding of the datacenter operational environment including the lifecycle of components within the system and system operation in the event of failures. Moreover, it is a requirement that key attributes of interest are identified in order to understand and relate attributes recorded which can be used to comprehensively and holistically study Cloud system behaviour and quantify the relationships between components.

The third challenge is the requirement of expert domain knowledge of Cloud computing datacenters required to perform comprehensive and non-superficial analysis of the system. Although it is possible to extract coarse-grain statistics of the Cloud system from a trace log including the number of tasks, users and servers, there is a requirement for more complex analysis such as calculating the amount of resource utilization wasted per server due to system inefficiencies, analyzing task submission patterns and energy usage from different types of users, as well identifying the relationships between components. This requires not only the necessary technical knowledge of statistical analysis, but more importantly expert domain knowledge in the area of Cloud computing datacenters which is critical to comprehensively extract, interpret and leverage the findings from the analysis.

2.5.3 Analysis Method Abstraction

Cloud datacenter environments can be massively heterogeneous from one another; not only in terms of different deployment and service models as discussed in Chapter 2.2.5-2.2.6, but also in terms of user business objectives, infrastructure specification, system size, physical location and network topology. As a result, the effectiveness of Cloud analytics is greatly enhanced by the inclusion of not just extracting and studying the statistical properties of the system and its respective components, but also the development of a detailed analytics method with sufficient generality enabling its use for other Cloud trace logs agnostically. Failure to do so results in limitations in the practicality of research, as researchers are less likely to understand and interpret results, and recreate the analysis process for their desired research objectives.

2.5.4 The Need for Cloud Analytics

The application of systems analytics is critical in the uptake of Cloud computing. A large-portion of existing work within Cloud computing research currently leverages system assumptions and model parameters which are not derived from empirical analysis of Cloud systems. This consequently results in system assumptions which may not be representative of real system behaviour, and limits the usefulness and accuracy of modelling Cloud components.

Currently, empirical study of Cloud datacenters is limited as demonstrated from the literature review detailed in Chapters 2.4.5-2.4.8, resulting in a lack of

quantification of the inherit characteristics of Cloud computing environments. In other words, while it has been identified extensively in the literature that Cloud computing exhibit characteristics such as heterogeneity, workload diversity, and elasticity discussed Chapter 2.2.2, it is still unknown how these characteristics precisely manifest and impact the operational behaviour of the system. As a result, it is necessary to analyze real production systems in order to characterize Cloud components to measure and study how these characteristics affect the system environment in key areas of interest to business, research and engineering including resource utilization, consumer behaviour, dependability and energy-efficiency.

Furthermore, in the context of the four analysis domains discussed in Chapters 2.4.5-2.4.8, there exist specific challenges and gaps within state-of-the-art in Cloud analytics that require to be addressed.

2.5.5 Workload

As discussed in Chapter 2.4.5, from the existing Cloud workload characterization and analysis it is clear that there are few available production tracelogs. Furthermore, previous analyses contain gaps that need to be addressed in order to develop more realistic workload patterns. It is imperative to analyze large data samples as performed by [105][107]; small operational time frames as those used in [106][109][103] could lead to unrealistic models which may not reflect realistic task behaviour. This is particularly true when considering tasks that execute for longer time periods.

Moreover, analyses need to explore more than coarse-grain statistics and cluster centroids. To capture the patterns of workload it is necessary to analyze parameters such as resource utilization and execution length as well as study the trends of cluster characteristics over different time periods. Although previous approaches offer some insight about workload characteristics, they do not provide a structured model that can be used for conducting analysis and simulation.

Finally, as discussed in Chapter 2.2.7, workload is always driven by users, therefore realistic workload models must include user behavioural patterns linked to tasks. The current state-of-the-art approaches completely focus on tasks, neglecting the impact of user behaviour on the overall environment workload.

Within the context of classifying Cloud workload using *k*-means clustering, there has been work performed in [106] and [109]. However, a limitation in existing approaches is that the number of *k* clusters selected is based solely on the analyst's expertise and qualitative measurements. This is a problem in two specific aspects: First, the number of unique clusters may vary temporally at different time periods due to changes within the Cloud environment. Second, this approach introduces subjectivity when selecting which number of clusters is appropriate for the system, a typical challenge in *k*-means clustering [135]. Finally, it limits its applicability to different Cloud datacenters, with each system composed to varying behaviour.

2.5.6 Servers

In the context of characterizing servers in Cloud computing datacenters, there presently exists work that analyzes Cloud servers in terms of resource utilization. However, such works include distinct limitations due to the number of servers analyzed, omission of crucial resource utilization such as Disk and Memory [129], and analysis of coarse-grain statistical properties that do not explore the heterogeneity of server architectures [107]. Furthermore, there is a need for studying more than aggregated resource utilization of servers at different time frames such as those found in [146]. As a result, there is not only a requirement for the analysis of large-scale Cloud datacenters to further our understanding of system operation, but also studying the characteristics of server operation of different server architectures fine-grain detail across a substantial system operation time span.

Finally, for analysis of server characteristics and utilization, there are opportunities to perform in-depth analysis of large-scale systems to study temporal and spatial characteristics of resource utilization for CPU, Memory, Disk and Network for heterogeneous systems. This analysis can be used in order to develop greater insights into the Cloud environment operation. Furthermore, in conjunction with task behaviour, it is possible to outline server inefficiencies within servers.

2.5.7 Failures

Current failure analyses of large-scale systems focus on analyzing failure statistics in terms of Time Between Failures [1][116], root cause failure [133][134], and performance implications; but have been limited due to confidentiality concerns,

Furthermore, while these analyses are useful for characterizing failures in large-scale distributed systems, no analyses exist for in-depth failure characteristics of Cloud computing environments.

Cloud has been stated to contain a number of unique system characteristics, however there is an urgent need for quantifying such failure characteristics in order to build more realistic research assumptions about the environment. The most obvious problem is that there currently exists very limited failure statistics and models for Cloud computing. The current state-of-the-art either focuses on a specific component [118], application specific [107], or small-scale datasets [116]. Furthermore, many models derived from failure analysis of non-Cloud distributed systems focus on high-level system behaviour, agnostic of the type of workload executing, leading to potentially obfuscating specific failure characteristics of different types of workload and server architectures. Such analysis can be used in order to validate a number of mechanisms which aim to improve system reliability.

2.5.8 Failure-related Energy

Important work has been developed in power/energy modelling of large systems such as those presented by [124]. Additionally, a significant number of resource management techniques such as [54] have been proposed to reduce energy waste while maintaining acceptable performance levels. However, these approaches have relegated the impact of software and hardware crashes on the energy waste of systems. Furthermore, as mentioned in [128][147], another factor that increases resource waste in systems is the amount of rolled-back computation after the occurrence of a failure.

There have been recent theoretical analyses and studies which have highlighted failures as a source of inefficiencies that increase energy consumption in large-scale datacenters, and have proposed a number of mechanisms to reduce energy waste including resource selection [127], rollback [128] and resource scheduling [3]. Unfortunately, all of these analyses are derived from theoretical data and provide no real insight into the true characteristics and dimensions of energy waste caused by failures.

Current failure analyses of large-scale systems discussed in Chapter 2.4.7, limited due to confidentiality concerns, focus on analyzing failure statistics in terms of Time Between Failures [1][116], root cause failure [133][134] of systems, and performance implications. However, all of these practical analyses neglect quantifying the impact of failures in terms of energy. The gap between these two strands of (theoretical and practical) work emphasizes the urgent need for failure analyses that consider the impact of failures on energy consumption.

From the literature review and the current state-of-the-art, it is observable that there is a clear gap between failure analysis approaches and the design of mechanisms to reduce the energy waste caused by failures. On the one hand, the analyses are completely focused on how the failure characteristics are correlated to performance drawbacks and completely neglect the impact of those failures on the energy waste. On the other hand, theoretical approaches emphasize the importance of reducing the waste produced by failures without providing any insight about the characteristics and dimensions of the addressed problem. As a result, this creates the requirement for comprehensive failure analyses that also include the energy impact in real production environments.

Analyzing the impact of failures on energy consumption is critical for two main reasons. Firstly, it enables researchers and practitioners to understand the characteristics and dimensions of the problem in order to create concrete scenarios for decision-making. For example, in order to know what dependability mechanisms to use and when to apply them effectively, datacenter administrators require awareness about the variables and conditions under which waste is produced. Secondly, it is important for practitioners (e.g. Cloud service providers) to understand the impact that both task and server failures have on datacenter energy waste and operational efficiency.

2.6 Summary

This chapter has provided a detailed summary of the concepts of Cloud computing, dependability, systems analysis and how they can be used to better study and quantify systems behaviour in Cloud datacenters in order to enhance system dependability and reduce operational waste.

The abstraction of the software system model has been presented, and the

concepts of a system being composed of multiple components which interact with each other through the service interface within the system environment have been introduced. Furthermore, the evolution of the modern distributed system has been discussed, including the formation of different software tiers to facilitate changes to computing paradigms, as well as implementations of this abstraction in systems such as High-Performance Cluster Computing, Peer-to-Peer systems, Service Oriented Architectures and Grid computing.

The concept and definition of Cloud computing, as well as the key terminology and characteristics have been presented. Furthermore, the components of Cloud computing including workload, composed of tasks and users, as well as virtualization, QoS and servers have been discussed in detail.

The concept of dependability, a fundamental aspect of computing systems, has been described. Furthermore, the need of such research within the context of Cloud computing identified and discussed in detail. Moreover, current Cloud computing research in the dependability domain has been presented

The concept of systems analysis has been presented, and the critical need for empirical analysis and modelling of Cloud systems discussed. A literature review of the current state-of-the-art of analyzing and characterizing Cloud workloads, servers, failures, and failure-related energy has been presented and discussed in detail. Finally, current gaps in the state-of-the-art for these four domains as well as opportunities where Cloud analytics can be enhanced in order to improve the effectiveness and practicality of developed mechanisms has been highlighted.

The technical and research challenges in Cloud analytics, including confidentiality and business constraints of available Cloud trace logs, characterizing and studying system behaviour due to large volume of data generated and the inherit complexity of Cloud components, discovering meaningful relationships between Cloud component data attributes, and the need for analysis methods to be sufficiently abstract so that they can be applied generically Cloud datasets. Furthermore, we have highlighted the challenges in the current gaps in the state-of-the-art analysis of Cloud workload, failures, servers and failure-related energy.

The next chapter presents the case study trace log, analysis infrastructure used for data extrapolation, and coarse-grain statistical properties of the case study.

3 Cloud Datacenter Case Study

As discussed in the Chapter 2, there is an urgent requirement for in-depth empirical analysis of Cloud datacenters in order to quantify their behavioural characteristics, develop concrete research assumptions and system scenarios, and identify operational inefficiencies. This chapter introduces the case study used in this thesis: a real-world production Cloud datacenter trace log produced by Google. The case study trace log specification, attributes and system component lifecycle is presented, and are used to develop a model describing component relationships within the datacenter. Furthermore, the analysis infrastructure constructed in order to extract the data in a timely manner is presented and discussed. Finally, this chapter concludes by analysis of the statistical properties and coarse-grain statistics of high level operation of the case study.

3.1 Case study: Google Cloud Trace Log

3.1.1 Trace Log Description

The operational traces are taken from a Cloud datacenter owned by Google released in May, 2011 and is publicly available at [155]. This particular dataset was selected for a number of reasons: First, it provides a holistic view of data operation for the datacenter, including data concerning servers, users, tasks and their respective resource utilization and events. Analyzed trace logs described in Chapter 2.4.5-2.4.7 currently restrict insight of system behaviour to a limited scope, thus the inclusion of operational data of components within the same trace log presents the opportunity to study their relationships as well as characterize system operation in-depth. Second, the dataset is sufficiently large to conduct detailed analysis at different time observation periods; spanning 30 days of operation and consisting of over 12,500 heterogeneous servers interconnected through a high-bandwidth network.

A *server* is defined as the hardware and the system software managing the hardware (i.e. hypervisor, OS, software supporting communication and networking). As described in Chapter 2.2.2, users of Cloud datacenters pursuing different business objectives request resources for tasks which are submitted and executed within servers. A *user* can be a human entity, a machine or another software program. Within this system environment, a *task* is the most basic unit

Table 3.1 Description of Google Cloud data tables.

Component	Data table	Description
Server	Machine events	Deployment and removal of servers within the cluster.
	Machine attributes	Properties of the server including kernel version and clock speed.
Workload (Task & User)	Job events	Event records for job submissions and completion.
	Task events	Records of events within the trace log; includes times of submission/completion as well as erroneous events. Also includes resource estimation by users.
	Task constraints	Scheduling constraints of tasks due to architecture and security requirements.
	Task resource usage	Record of task resource utilization (CPU, Memory, Disk, Network, etc.) every 5 minutes.

of computation of software within a server. Within the context of the case study trace log, a task is represented as a Linux container (described in Chapter 2.2.9).

The trace log contains six data tables that capture different aspects of system operation as shown in Table 3.1. Data tables are separated into Comma Separated Value (CSV) files and are altogether 500GB in size. The data tables capture operational data including task execution constraints, server architecture specifications and task resource utilization. In addition, a number of data tables record events which occur within the system; an *event* is defined as an action that changes the state of a task or server in a specific time and place. Details about each attribute described in the data tables can be found in [156].

3.1.2 Server Events

Attributes recorded for server events include the server identifier, specification of the server architecture, the timestamp of event occurrence (recorded in microseconds) and the event type. Servers vary in terms of physical capacity of Memory and CPU (RAM size and CPU cores, respectively) as well as platform ID, which is a combination of micro-architectures and memory technology resulting in different clock rates and memory speeds. As a result, the combination of unique server CPU capacity, memory capacity and platform ID represents a unique server architecture type.

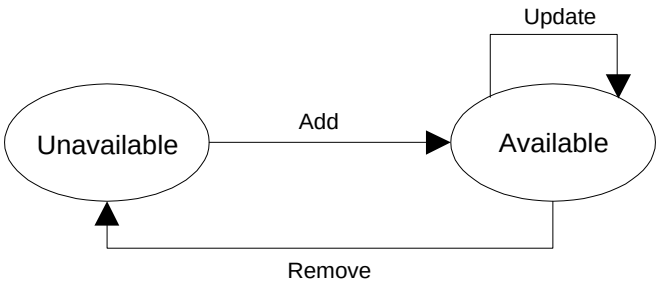


Figure 3.1 Server lifecycle

To further our understanding of system operation, it is advantageous to study the lifecycle of a server as shown in Figure 3.1 in order to observe all possible states a server can exhibit. Servers are capable of transitioning to different states as a result of an event occurrence. There are two possible states for servers; legible for task scheduling and executing (available) and disconnected from the cluster (unavailable). There are three possible events which will cause the server to transition between states: A server is made available to the system environment (ADD), the server is disconnected from the cluster due to failure or maintenance (REMOVE) and the available resources for a specific server are modified (UPDATE). Further details about server attributes can be found in the trace log specification [156]. A typical example of a server's lifecycle involves a) the server being connected to the Cloud datacenter (Server transitions from Unavailable to Available through the ADD event), b) modification of resources made available to the server such as CPU and memory (server receives an Update event), and c) server removal from the datacenter due to failure, maintenance of decommission (Server transitions from Available to Unavailable, through the Remove command).

3.1.3 Tasks Events

The task events data table contains a shared number of attributes found within the server event log. These include the task identifier (a combination of the task index and job identifier), timestamp of occurrence, and event type. Additionally, there are unique attributes, such as the Server Identifier where the task was allocated, the user identifier identifying the task owner, and the amount of CPU, Memory and Disk requested by the user.

Similar to servers, to better understand the role of task events and the possible states a task can exhibit within the system environment, it is necessary to study the task lifecycle. A task during its lifecycle can transition through a number of

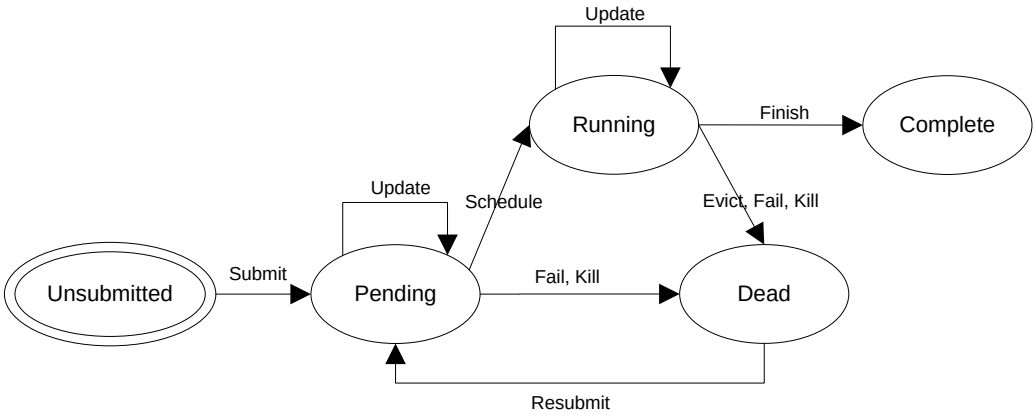


Figure 3.2 Task lifecycle

states as depicted Figure 3.2. A task will be assigned *Pending* status when it is waiting to be allocated after being initially submitted by the user or re-submitted by the task scheduler. Once the scheduler finds a suitable server to allocate the task and it is deployed, the status transitions to *Running*. When a task successful finishes execution, it transitions to *Complete* status and is subsequently removed from the system. A task that is descheduled without successfully completing execution transitions to *Dead* status.

There are three types of events that alter the state of a task to Dead without successful completion, from this point we refer to these events *as Termination Events* (TEs). First, it is possible for a task to be evicted from a server (EVICT) due to over commitment of the scheduler; the server on which the task is being executed becomes unstable, or the disk holding the data of the task fails. Second, it is possible for a task to terminate due to a software crash within the task (FAIL). Lastly, a task can be cancelled due to the user, loss of dependencies with other tasks, or unknown causes of termination (KILL). If any of the above three events occur, the task is resubmitted to the resource scheduler for server reallocation. An individual task can only run within a single server at any given time and can be allocated to another server upon rescheduling. Attributes such as the amount of resource requested by a user can be updated within the Pending and Running status.

To give a practical example, a task is submitted into the Cloud datacenter and is scheduled to the resource scheduler for server allocation (Task is assigned Pending status through the Submit event), once a suitable server is discovered or made available, the task is allocated to that server for execution (Task transitions to Running through the Schedule event). Moreover, it is possible within the task's

execution lifetime for the resource amount allocated to be altered (task experiences an Update event). In the best case scenario, a task successfully completes its execution and is consequently descheduled from a server (Task transitions to Complete status through the Finish event); however, it is possible for a task to experience a Termination Event resulting in unsuccessful completion (Evict, Kill and Fail resulting in Dead state), consequently causing the task to be resubmitted to the resource scheduler for server reallocation (Task is Resubmitted to Pending status).

The scheduling policy of the system is predominantly based on task priorities. The priority scale for tasks ranges from 0 to 11 to indicate the lowest and highest priority, respectively. When required, low-priority tasks are evicted to yield resources to higher tasks. According to [157][158], the trace log contains a mixture of different types of workload, composed of production tasks (priority 9), monitoring (10-11), batch job processing and latency sensitive tasks (2-8) and gratis tasks (0-1). Further details concerning task events and the task life cycle can be found in [156].

3.1.4 Task Resource Usage

The trace log also records the resource usage of individual tasks executing on a server. Resource usage is expressed as an aggregate utilization value calculated over a time frame of 300s (5 minutes). The task resource usage table contains a

Table 3.2 Attributes of the task resource usage data table

Identifier	**Attribute**
Timestamp	Resource monitoring start time
	Resource monitoring end time
Task identifier	Task index
	Jobid
Server ID	Server ID
CPU	CPU usage (mean and maximum)
	CPU rate
	Cycles per instruction
Memory	Memory usage
	Assigned memory
	Mapped and unmapped page cache memory usage
Disk	Disk IO time (mean and maximum)
	Local disk spaced used

large amount of attributes as shown in Table 3.2, including the task identifier, aggregated resource utilization of CPU, Memory and Disk, and server identifier where the task HAS executed. Values for resource utilization are normalized values ranging between 0-1, where 1 represents the largest capacity of a server within the Cloud system.

3.2 Datacenter Analysis Method

3.2.1 Datacenter Model

It is possible to extrapolate the data from datacenter trace logs in order to study and understand the relationship and interactions of components (users, tasks, and servers) within the Cloud as discussed in Chapter 2.5.2. To facilitate this, it is necessary to construct a model of the datacenter environment leveraging attributes found within the data tables and the lifecycle of tasks and servers in order to define datacenter entities and identify their respective characteristics. As shown in Figure 3.3, it is observable that there are a number of attributes which relate different entities together across data tables such as jobID, taskindex and serverID. Furthermore, we observe that the tables machine events, machine attributes, task events and task resource usage all contain the time stamp attribute, which can be leveraged when studying temporal behaviour over defined observational periods. On the other hand, it is possible to use the server ID to study spatial behaviour including task utilization and events that occur on specific servers.

Table 3.3 Summary of Catalogs

Catalog	**Attributes of interest**
User Catalog	Submission rates, resource requested, number of tasks submitted
Task catalog	Resource utilization, execution length, submission time, priority type
Server catalog	Resource utilization, useful utilization, wasted utilization
Task failure catalog	MTBF, MTTR, priority, resource utilization, server ID, failure time
Server failure catalog	MTBF, MTTR, tasks failed, failure time
Energy failure catalog	Energy waste (server included), energy waste (server omitted)
Platform catalog	Server Architecture ID, Machine ID, Platform ID

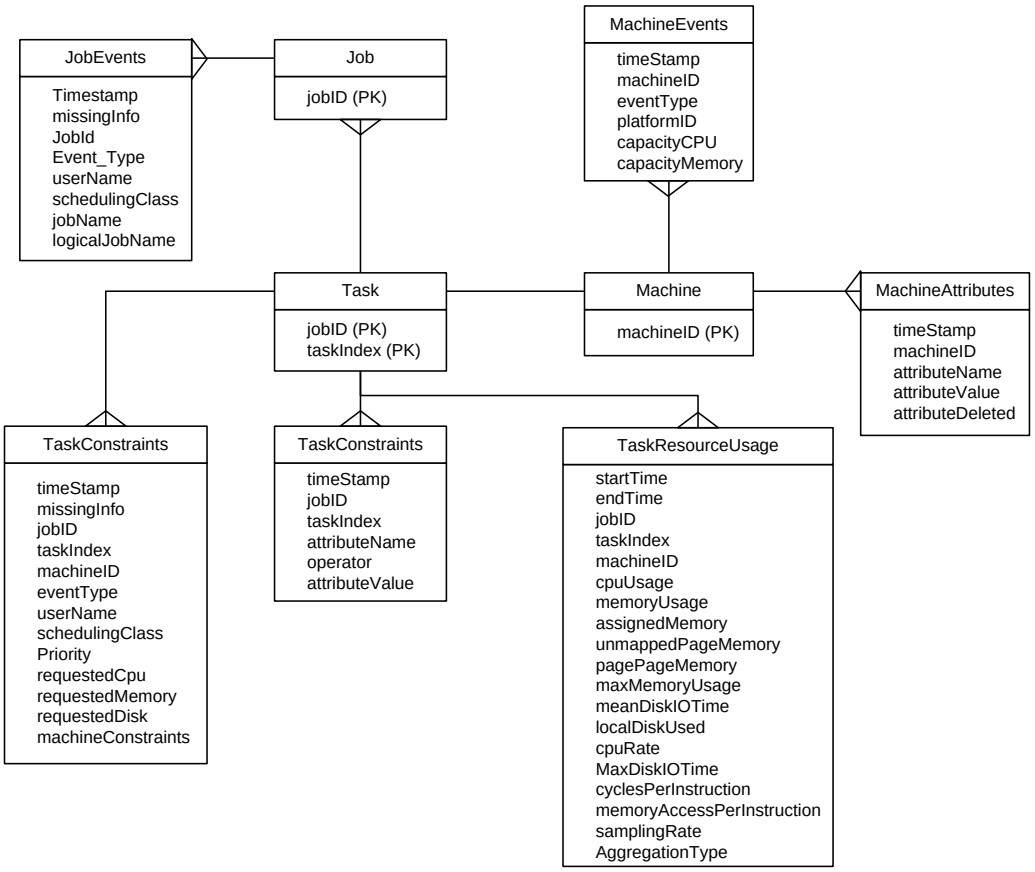


Figure 3.3 Entity Relationship Diagram of Google Cloud

This datacenter model (represented as an Entity Relationship (ER) model [159]) can be implemented within a database in order to extract and process data of interest. However, in order to perform data extrapolation data in a timely manner and to avoid and reduce query complexity and execution requiring multiple joins between data tables, it is necessary to create additional data tables which group together attributes of interest from the raw data tables for specific analysis objectives. These data tables are composed of attributes taken, or derived from the raw trace log data tables including data aggregation, counting and complex calculations. These created data tables are defined as *catalogs*, and are used to study specific system behaviour as shown in Table 3.3. Further details of each catalog, including how synthesized attributes were calculated is described in further detail within subsequent sections detailing the method of analysis for each Cloud analysis domain (workload, servers, failures, and failure-related energy).

3.2.2 Analysis Infrastructure

Each catalog and data table was deployed within a database for storage and querying. Due to the number of records a data table can contain (for example,

task resource usage contains over 1.2 billion rows) as well as the number of data table joins required for the data extraction and processing, it was discovered that deploying the database on a centralised machine was not feasible. Efforts to perform a simple join query between task events and task resource usage tables resulting in a query execution completion time greater than 160 hours.

Due to the number of queries required to explore the data as well as extract information of interest, it was necessary to develop and deploy a distributed analysis infrastructure to complete the analysis in a timely manner. To facilitate this, a distributed infrastructure composed of 50 machines (Intel Core 2 Quad CPU, 2.83 GHz, 8GB memory and 468 GB memory) was developed to provide the computational power and storage capacity required to perform the analysis. The cluster used Hadoop MapReduce [160], a programming model developed for processing datasets. MapReduce functions by dividing computation into two phases: Map and Reduce. The map phase divides a process into smaller sub-processes and distributes them across nodes. The reduce phase collects the output of each map and combines them together and outputs the result.

Furthermore, the analysis infrastructure uses HIVE [161], a data warehouse system for storage and query computation. HIVE is a database management system that uses Hive Query Language (HiveQL). HiveQL shares similar functionality to SQL and is capable of interfacing with MapReduce and converting HiveQL queries automatically into MapReduce processes as shown in Figure 3.4. As depicted, a HiveQL query is entered into the HIVE database management system, which is automatically translated into a MapReduce process, which then calculates the number of Map and Reduce nodes required to successful complete query execution. The Map phase is conducted across the nodes within the distributed analysis infrastructure, and is then collected and compiled together in the Reduce phase within a single node. After the Reduce phase is completed, the results of the query are outputted to a CSV file for conducting the desired analysis. Using this approach, it was possible to reduce the total execution time of queries by orders of magnitude, from 160 hours to approximately 15 minutes.

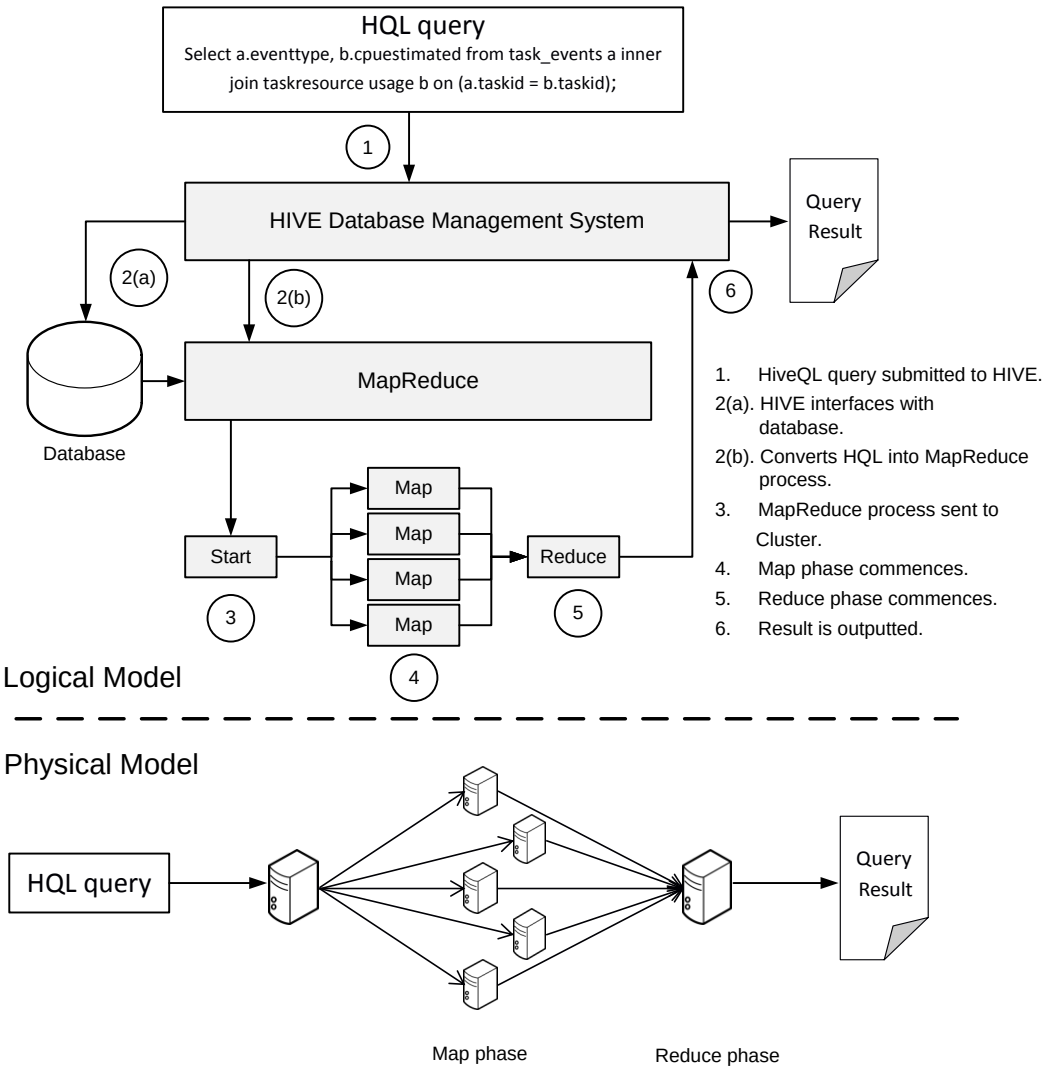


Figure 3.4 Analysis Infrastructure Model

3.2.3 Statistical Analysis Tools

After extrapolating data of interest for this research, it was necessary to perform complex analysis techniques using a suite of statistical software packages. The software packages Minitab [162] and R [163] were used for the statistical analysis while EasyFit [164], was used for distribution modelling. These software suites were selected over other technologies as they contain a large amount of functionality that can be exploited to support the analysis of the trace log including cluster analysis, sampling, data visualization, distribution modelling, general statistical analysis and correlation analysis.

3.3 Coarse-grain Analysis

This section presents the statistical properties of high-level operation for the Cloud computing datacenter derived from using the analysis infrastructure

described in Chapter 3.2. Table 3.4 depicts the general statistical properties of the trace log. Workload is created by 930 different users submitting 650,000 different jobs comprising just over 25 million submitted tasks. In this study, we focus on tasks instead of jobs as the vast majority of jobs are composed of a single task and to achieve fine granularity of analysis. We observe that there are over 47 million tasks scheduled which is due to the possibility that a submitted task can be rescheduled multiple times. Furthermore, there are 144,000,000 and 38,000 recorded events corresponding to tasks and servers, respectively.

Users have different submission patterns ranging from a single task to 3.5 million tasks over the trace log time span. From Figure 3.5(a), it can be seen that almost 94% of the users submit a small proportion of the total tasks within the Cloud environment, whilst the remaining 6% of users submit a large proportion of tasks into the system. This is further exemplified in Figure 3.6(a), which demonstrates that 1% of total users submit over 53.2% of the total submitted tasks within the system. This observation indicates that it is possible for a small number of users to have substantial impact on the system operational behaviour and resource consumption due to the sheer volume of tasks they submit, as well as the composition for the type of workload within the system.

Figure 3.5(c) depicts the variability of task submissions per day over the trace log time period, ranging from 678,929 to 4,940,423 tasks submitted daily. Each submitted task consumes CPU, memory and disk space at different amounts. As observed in Figure 3.5(b), approximately 98% of tasks incur small to moderate resource consumption while the remaining 2% incur large resource demands;

Table 3.4 General Trace Log Statistics.

Total Users	930
Total Jobs	650,000
Submitted Tasks	25,375,377
Scheduled Tasks	47,351,173
Task Events	144,010,768
Server Events	37,780

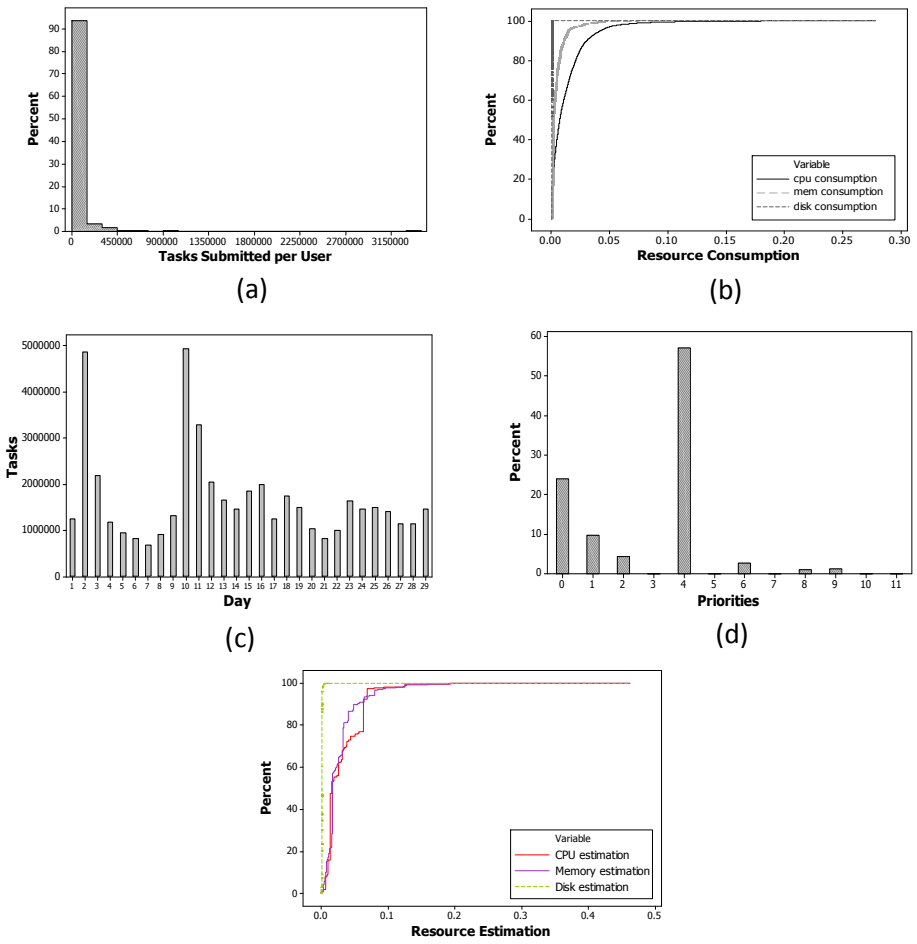


Figure 3.5 Coarse-grain analysis of task a) submission proportions, b) resource consumption, c) daily submissions, d) priority submissions, e) Resource estimation of users.

this is reflected by a mean and standard deviation for task execution time of 1945 seconds and 5751 seconds, respectively. The distribution of task execution time is right-skewed as demonstrated in Figure 3.6(b), reflecting similar characteristics to resource utilization of tasks; the Cloud datacenter studied appears to be composed of a large proportion of tasks with small execution times and resource utilization, and a small proportion of tasks which consume a large amount of resources and execute for a longer period of time. Such behaviour has also been observed in the analysis of other Cloud datacenters such as those in [107], which also observe a long-tail in job execution.

Figure 3.5(d) illustrates the distribution of task priorities in the analyzed datacenter, where it can be observed that lower priorities (0-8) represent 99% of the total submitted tasks, with task priorities 0 and 4 comprising 24% and 57%, respectively. Furthermore, we observe that higher or production task priorities

Table 3.5 Server proportions and submission rates

Server Platform	Server Type	CPU Capacity	Memory Capacity	Server %	Task submission %
A	1	0.25	0.25	1	0.65
B	2	1.00	1.00	6.34	6.92
	3	1.00	0.50	0.018	0.00409
C	4	0.50	0.25	30.76	25.93
	5	0.50	0.75	7.93	8.36
	6	0.50	0.50	53.46	57.89
	7	0.50	0.97	0.04	0.055
	8	0.50	0.12	0.43	0.19
	9	0.50	0.03	0.024	0.00060
	10	0.50	0.06	0.008	0.00031

(9-11) account for the remaining 1% approximately. These results imply that different type of users submit tasks of varying priority (as defined in Chapter 3.1.3), with development and batch-processing jobs within the case study being the most frequent.

Moreover, we observe that there exists a disparity between resource estimation patterns and actual resource consumption, as shown in Figures 3.5(e) and Figure 3.5(b), respectively. This is due to Cloud users typically overestimating the amount of resources actually required to provision their desired service [165]. Work in [40] discusses the resource disparity between estimation and actual utilization in further detail, and observe that users overestimate the requirements for resources for acceptable service. We can observe that Disk usage is relatively low in comparison to CPU and memory consumption; this is

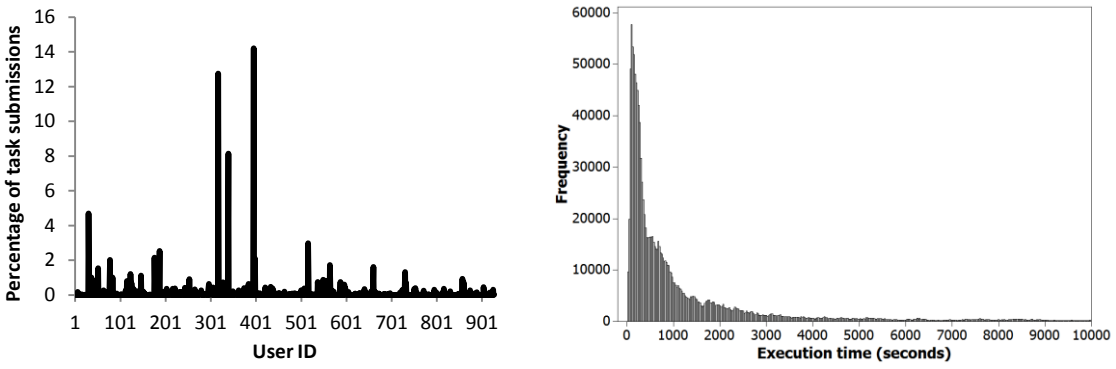


Figure 3.6 (a) Percentage of tasks submitted per User,
(b) Histogram of task execution time

due to the nature of the workload type executed within the Cloud environment, which appear to vary more in terms of CPU and memory consumption.

Table 3.5 presents the number of servers within the trace log categorized by server architecture type. There are a total of 10 unique server architecture types (as defined in Chapter 3.1.2), with server architecture 1 constituting 53.46% of the total number of servers within the trace log. Furthermore, we observe that there exist five server architecture types that represent less than 1% of the server population. We postulate that these servers are used for tasks which have specific constraints on the type of hardware architecture required for execution. We also observe that there exists a strong correlation between server population and the number of tasks submitted to a specific server type as shown in Figure 3.7 (a) and 3.7(b), respectively indicated by a Pearson Correlation Coefficient value of 0.994, representing strong correlation. This result presents insight into the type of scheduling algorithm deployed within the datacenter, which appears to be a type of load balancing of tasks across all servers, given the strong correlation between server utilization and number of tasks submitted.

Even from a coarse-grain analysis of the statistical properties of the case study Cloud datacenter, it is observable that there exists a high degree of diversity in terms of server heterogeneity, temporal and spatial patterns of user submission and resource request patterns, as well as different types of tasks in terms of resource utilization for CPU, Disk and Memory, execution time and priority type. Such evidence of heterogeneity has been identified individually in previous works

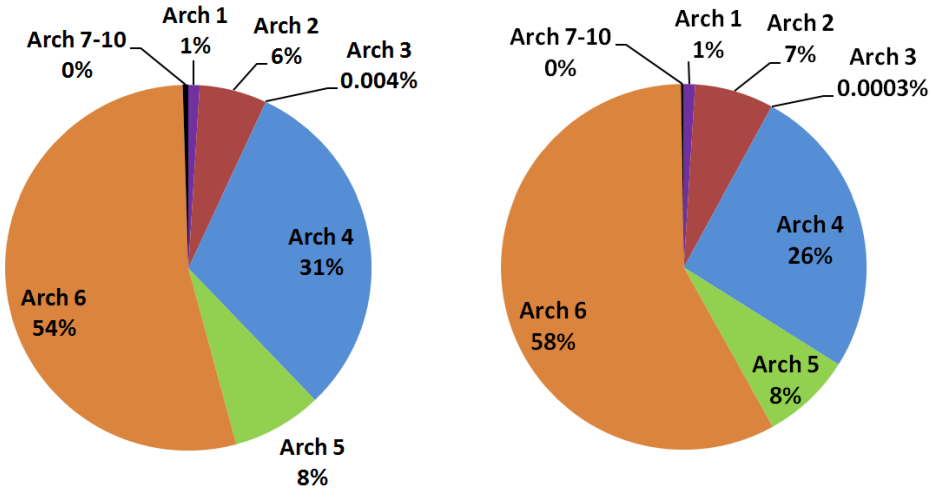


Figure 3.7 Proportions of (a) Server population, (b) Task submissions per server

focusing on analysis of specific Cloud components discussed in Chapter 2.4, and is for the first time demonstrated holistically at a large-scale across all Cloud components in this work. These findings demonstrate that the sum of the diverse utilization patterns and user submissions create and provide empirical evidence of a highly heterogeneous workload environment submitted onto heterogeneous servers architectures typical of multitenant Cloud datacenters that have been claimed to exist as discussed in Chapter 2.2.2.

The coarse-grain analysis is a good technique to provide an overview and insight into general system characteristics and behaviour. Attention is now drawn to in-depth analysis to explore the characteristics and behavioural patterns of workload - a key component in Cloud datacenter operation and modelling.

3.4 Summary

In this chapter, the case study for this thesis has been presented; a large-scale production Cloud environment containing over 12,500 servers spanning 30 days of operation. The system specification, the lifecycle of server and tasks, and the attributes within the trace log data tables are defined and discussed in detail. Furthermore, the process, challenges and solution to developing an analysis environment including the modelling the components within the Cloud datacenter as well as deploying analysis infrastructure capable of extrapolating and processing data within a timely manner has been presented and discussed in detail.

Using the analysis environment, we present coarse-grain statistical properties of the Cloud trace log, consisting of high-level information including daily submission patterns of users, the type and amount of resource requested and consumed, and composition of the server population. We observe and demonstrate that there exists a large degree of heterogeneity across the Cloud datacenter, with diverse submissions of tasks, different types of user submission patterns, resource estimation and utilization of tasks, and server architecture.

The remainder of this thesis is dedicated to the presentation, discussion and evaluation of in-depth analysis of workload, servers, failures and failure-related energy in Cloud datacenters, including method of extraction, research assumptions, analysis results and practical application.

4 Workload Characterization

4.1 Overview

As discussed in Chapter 2.2.7, workload is a core component in Cloud computing, comprised of users submitting tasks of heterogeneous execution length and resource utilization in order to pursue diverse business objectives. However, presently there is a substantial lack of empirical studies comprehensively quantifying workload characteristics and behavioural patterns in large-scale Cloud computing datacenters. Current work presently focuses entirely on tasks, neglecting user behavioural characteristics responsible for task submission patterns and the amount of requested resources required. Furthermore, existing approaches do not comprehensively study user and task diversity spatially of temporally, or provide a detailed method of workload analysis and modelling capable of quantifying and extracting empirical findings that is of practical use to researchers and Cloud providers.

This chapter presents the study and analysis of characteristics and behavioural patterns of Cloud workload: specifically, this chapter contains the following contributions:

- *A reusable method for characterizing and classifying Cloud workload based on users and tasks attributes*. This approach leverages rigorous statistical and modelling techniques to characterize and classify users and tasks to support Cloud computing research, providing realistic workload characteristics and parameters derived from large-scale production environments. The method can be applied to numerous Cloud computing trace logs in order to analyze workload behaviour of users and tasks.
- *An in-depth empirical analysis to quantify Cloud workload characteristics and behavioural patterns*. The method described above is applied to a production large-scale Cloud datacenter trace log. The analysis comprises the study and quantification of statistical and behavioural properties of users and tasks both temporally and spatially, as well as a study of their impact on the system environment. Furthermore, the diversity of workload over the entire trace log timespan and a number of

observational periods is explored to investigate and quantify the degree of diversity for user and task behaviour that manifests within the system.

- *An extensive distribution analysis to study the operational behaviour of users and tasks, and to extract model parameters for practical usage*. An extensive study of the internal characteristics of users and tasks is presented, and the distribution model parameters extracted and presented can be used by providers and researchers to capture Cloud environment behaviour to support research assumptions of the operational environment.

The outputs of these contributions can be used to quantify the real-behaviour of users and tasks in Clouds and to validate whether Cloud computing environments exhibit the characteristics of significant diversity and heterogeneity as claimed in Chapter 2.2.2. Specifically, this includes classifying the different types of users and tasks, changes in their behavioural patterns at different time frames, and the relationship that exists between each other. Furthermore, we present model parameters for users and tasks, which can be leveraged for practical use by researchers to develop realistic simulation environments as well energy-aware resource management mechanisms in the domains of overallocation and performance-interference.

4.2 Workload Characterization Method

This section presents the method for characterizing workload including the workload model, the method of classifying users and tasks, the sampling process for selecting the observation periods for analysis, and the assumptions of the workload and the system environment for data extraction and processing.

4.2.1 Workload Model

The first step of the analysis method is to define the workload model in order to conduct the analysis and modelling. In order to achieve this, it is necessary to identify the key attributes which are hypothesized to define the characteristics of users and tasks, as well significantly impact user and task behaviour when altered.

As described previously in Chapter 2.2.7, users are responsible for driving the volume and behaviour of tasks in terms of requested resource and task

submission patterns. In the context of the Google Cloud trace log, it is possible to identify three attributes that define key characteristics which are fundamental to describe user behaviour: the submission rate α, and requested amount of CPU β and Memory ϕ. The submission rate is defined as the frequency which users submit tasks into the Cloud datacenter. Submission rate (presented as task submissions per hour) is a quotient and is calculated by the sum of total submission events per user, and divided by the number of submissions over the trace log time span. Requested CPU and memory represent the amount of resources requested by a user upon task submission, and are extracted directly from the task event data table from the trace log.

Tasks are defined by the type and amount of work dictated by users, resulting in varying execution length and resource utilization patterns. As a result, attributes within the tracelog that define tasks are length χ, and the average resource utilization for CPU γ and Memory π. Length is defined as the total amount of work to be computed and is calculated based on the task duration and average CPU utilization; this is due to task duration (measured is seconds) being dependent on the architectural characteristics of the server where the task is allocated [167]. As a result, describing the task in terms of length allows us to perform an architecture-agnostic workload analysis, and is measured in Millions of Instructions (MI). CPU and Memory utilization is the average resource utilization of a task and is represented as the mean of all consumption measurements recorded in the tracelog for individual tasks. The attribute Disk utilization was not included within the workload model, as it was discovered from the exploratory analysis in Chapter 3.3 that 98% of disk usage was uniform and consumed miniscule amount of resources due to the nature of tasks within the studied Cloud environment. As a result, this attribute does not significantly impact task behaviour, and was consequently omitted from the task model.

Cloud workload is a composition of users and tasks and can be defined as a set of users with profiles *U* submitting tasks classified in profiles *T*, where each user profile u_i is defined by the probability functions of α, β and ϕ, and each task profile t_i by χ, γ and π. The expectation $E(u_i)$ of a user profile is given by probability $P(u_i)$, and the expectation E(ti) of a task profile is given by its

probability $P(t_i)$ conditioned to the probability of $P(u_j)$. The model components and their relationship are formalized in Equations 1 to 6.

$$U = \{u_1, u_2, u_3, ..., u_i\} \quad (1)$$

$$T = \{t_1, t_2, t_3, ..., t_i\} \quad (2)$$

$$u_i = \{f(\alpha), f(\beta), f(\phi)\} \quad (3)$$

$$t_i = \{f(\chi), f(\gamma), f(\pi)\} \quad (4)$$

$$E(u_i) = u_i P(u_i) \quad (5)$$

$$E(t_i) = t_i (P(t_i) | P(u_j)) \quad (6)$$

The defined workload model and its respective attributes can be leveraged in order to comprehensively study and analyze users and tasks within the Cloud datacenter.

4.2.2 Workload Classification

The second step of the analysis method is to classify tasks and users composed by their respective attributes. To facilitate this, *k*-means clustering was used for classification. *k*-means clustering is a popular data-clustering algorithm to divide *n* observations into *k* clusters, in which analyzed data sets are partitioned in relation to the selected attributes and grouped around cluster centroids [166]. One critical factor in the *k*-means clustering algorithm is determining the optimal number of clusters; this challenge in the context of workload characterization is highlighted in [145] where the number of *k* clusters is selected by the analyst based solely on visual interpretation and qualitative metrics, introducing subjectivity into subsequent analysis. As a result, we use the statistical method proposed by Pham, et al [145] which calculates a satisfactory number for *k* based on quantitative metrics, avoiding qualitative techniques that introduce subjectivity. This clustering method considers the degree of variability among all data points within the derived clusters in relation to the number of analyzed attributes. A number of clusters *k* is suggested when this variability represented by function $f(k)$ is lower than or equal to 0.85 according to extensive case studies conducted by the authors on a number of datasets. The clustering method is shown in Equations 7 and 8:

$$f(k) = \begin{cases} 1 & \text{If } k = 1 \\ \dfrac{S_k}{\alpha_k S_{k-1}} & \text{If } S_{k-1} \neq 0, \ \forall\ k > 1 \\ 1 & \text{If } S_{k-1} = 0, \ \forall\ k > 1 \end{cases} \quad (7)$$

$$\alpha_k = \begin{cases} 1 - \dfrac{3}{4N_d} & \text{If } k = 2 \text{ and } N_d > 1 \\ \alpha_{k-1} + \dfrac{1-\alpha_{k-1}}{6} & \text{If } k > 2 \text{ and } N_d > 1 \end{cases} \quad (8)$$

where S_K is the sum of cluster distortions, N_d is the number of parameters within the population and α_k is the weight factor based on the previous set of clusters.

For the analysis, the *k*-means clustering algorithm uses values ranging from 1-10. For each value of *k* we calculate *f*(*k*) using Equations 7 and 8. Based on the results we were able to statistically determine the number of clusters for U and T (Equations 1 and 2), respectively.

4.2.3 Sampling Process

One of the objectives of this chapter is to quantify the temporal diversity of Cloud workload. Specifically, how the characteristics and behavioural patterns of users and tasks change over different time periods. Such analysis allows the inspection of patterns that exist within the data and the exploration of the degree of variance over the system lifespan. As a result, apart from analysing the time span of an entire trace log, it is advantageous to study additional observation periods in order to study and quantify behavioural patterns at different times of operation, as well as create sample populations which can be used when validating derived distribution model parameters.

In the context of the Google Cloud trace log, three additional observational periods each spanning 24 hours have been selected in order to observe system operation at different time periods. We define the month observational period between time stamps 0 and 2505600000000 (representing 29 days in microseconds) within the trace log. Any records recorded after the latter timestamp were omitted from the analysis, as it only includes several hours of

additional recorded operation, which would result in misinterpretation of the results when analyzing daily behaviour. The four observational periods selected consist of the entire month trace, Day 2, Day 18 and Day 26. The latter three observational periods were selected for two reasons: First, they represent observational periods of low task length, high submission rate and an average of these two attributes, respectively. Secondly, the periods are temporally spread across the trace log period, and provide insight into system operation at different time periods.

4.2.4 Workload Analysis Assumptions

In order to produce a fair and comprehensive analysis to effectively characterize workload, it is necessary to define realistic assumptions about system operation. These assumptions are as follows:

- *A task is considered the basic element that consumes resources*. As the resource consumption is logged by tasks, and the majority of jobs are comprised of a single task, the analysis focuses on tasks for studying fine-grain behaviour while jobs are considered as the grouping element.
- *The task duration is considered from the timestamp of the latest submission event to successful completion*. This is due to the possibility of total task execution time being affected by system events such as resubmission caused by Termination Events. Analysis of Termination Event impact on the system is discussed in detail within Chapter 6.
- *Task length is calculated based on duration execution and average CPU utilization and is measured in Millions of Instructions (MI).* Duration is dependent on the specific architectural characteristics of the server where the task is allocated for execution [167]; as a result, describing the task in terms of length in MI allows us to perform workload analysis agnostic of architecture, as well as apply the model to datacenters of different server specifications.
- *Tasks that start before or finish after the tracelog time frame are not considered in the analysis*, as it is impossible to derive the length attribute for tasks which operate outside the trace log observational period.

- *Monitoring tasks are omitted from the analysis*. Task priorities 10-11 are responsible for monitoring the system to produce the trace log data, and are not consumers within the Cloud datacenter. Coupling this with the fact that they make up less than 1% of the total task population and have uniform resource consumption and behaviour, such tasks have been omitted from any subsequent analysis.

Furthermore, there is also a requirement to define explicit assumptions for the case study trace log when considering the task and server life cycle described in Chapter 3.1:

- *Every time a task terminates it is assumed that it restarts from the beginning of execution*. This is supported by the fact that a task failure is an interruption on a running task, requiring the system to re-execute the interrupted task [3][160]. Such behaviour and events are discussed in further detail within Chapter 6.
- *Server specification of the trace log is derived from empirical and widely used server benchmarks*. The server specification of different server architectures within the trace log are obfuscated due to confidentiality and business concerns. Consequently, server specifications are presented as Platform IDs and proportions of CPU and Memory as described in Chapter 3.3. As a result, we map servers to the specification of real physical servers taken from the SpecPower Benchmark 2008 [182]. The server Primergy RX200 S7 architecture was selected for server architecture 2, representing the largest capacity within the trace log. Consequently, resource capacity of CPU and Memory of the remaining server architectures are based on proportions of this server architecture. The processing power of this server was selected to calculate the length of tasks within the system.
- *Disk usage is not considered in task characterization due to uniform usage patterns*. This is due to 98% of tasks within the system share disk usage patterns as presented in Chapter 3.3 and confirmed in [158] resulting in this attribute to be unsuitable for classification and clustering purposes.

4.2.5 Analysis Method

The workload analysis is divided into two primary sections; cluster analysis and distribution analysis. The objective of the cluster analysis is to classify users and tasks, quantify their respective statistical properties, and study their temporal and spatial behavioural patterns across the entire system timespan as well as selected time observation periods as discussed in Chapter 3.3. Specifically, the cluster analysis studies the characteristics and behaviour of the created clusters and the statistical properties of each identified attribute within the workload model, including the Mean, Standard Deviation and Coefficient of Variation (*Cv*). In the context of the Google trace log, we investigate the variance of task and user clusters and their respective attributes over three additional observational periods of Day 2, Day 18 and Day 26 in order to inspect patterns that exist within the Cloud and to explore the variability and dynamicity over the system lifespan.

The objective of the distribution analysis is to study the internal data distributions of attributes in each task and user cluster in order to better understand intra-cluster behaviour and diversity, as well extract model parameters of practical use for researchers. This requires fitting the data to the closest theoretical distribution using a GoF test to obtain the parameters of their Probabilistic Distribution Functions (PDF). The data of each cluster is fitted to a parametric distribution by using the Anderson-Darling (AD) GoF statistical test, with the theoretical distribution with the lowest AD-value selected to represent the data distribution of each cluster attribute. The parameters of the PDFs for each workload model attribute described in Equations 3 and 4 are extracted and can be by other researchers in evaluating energy-efficient mechanisms and developing realistic simulation environments.

Now that the method for workload analysis and characterization and how it is applied to the Google trace log has been defined and discussed in detail, the remainder of this chapter is dedicated to the study and analysis of workload characteristics and behavioural patterns within the case study trace log.

4.3 Cluster analysis

4.3.1 Users

Figure 4.1 illustrates *k*-clusters partitioning that satisfies f(*k*) < 0.85 for users across the observational periods of the entire month, Day 2, Day 18 and Day 26. From Figure 4.1(a), which visualizes the cluster composition of users across the entire 29 trace log time span, we observe that the majority of users request similar portions of CPU and memory, and exhibit similar submission rates. Moreover, there exist three specific users whom exhibit a substantially high submission rate and request larger amounts of CPU and memory within clusters 2 (*U2*) and 3 (*U3*), respectively. When omitting these three users from the cluster analysis as depicted in Figure 4.1(b) it is clearer to observe that cluster

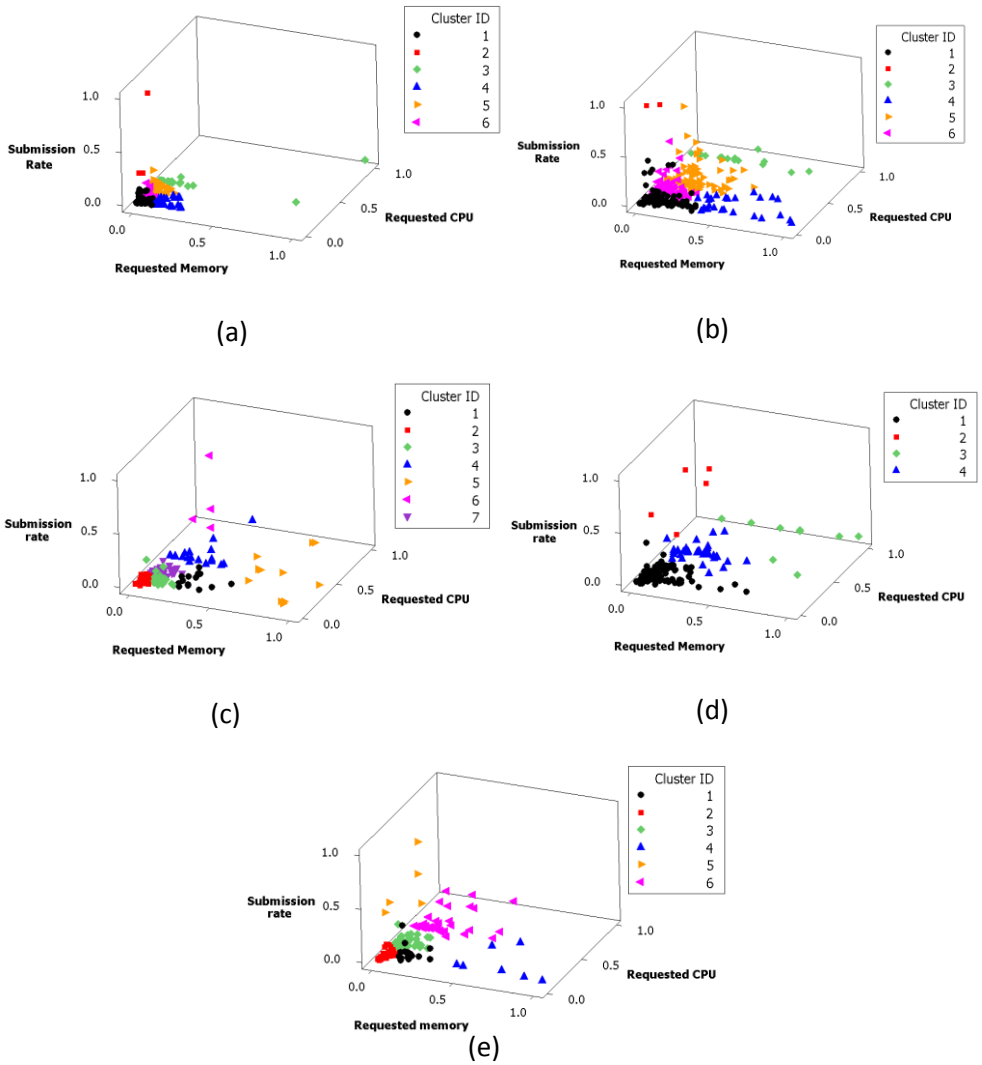


Figure 4.1 Clusterisation for users (a) Entire month, (b) Entire month (omitting outliers), (c) Day 2, (d) Day 18, (e) Day 26.

Table 4.1 Statistical properties of users for different time periods

Time period	Requested CPU			Requested Memory			Submission Rate		
	Mean	***Stdev***	***Cv***	***Mean***	***Stdev***	***Cv***	***Mean***	***Stdev***	***Cv***
Month	0.036	0.031	0.862	0.026	0.027	1.043	59.63	272.3	4.566
2	0.030	0.000	0.000	0.019	0.021	1.127	262.20	996.5	3.801
18	0.032	0.025	0.788	0.020	0.022	1.090	231.90	755.2	3.257
26	0.031	0.026	0.843	0.023	0.029	1.256	7.64	23.22	3.039

characteristics exhibit similar patterns across observational periods Day 2, 18 and 26 as demonstrated in Figure 4.1(c-e), respectively.

This is further exemplified in Table 4.1, which presents the statistical properties of all users for each time observational period. It is observable that parameters CPU and Memory requested exhibit similar values between approximately 0.030 - 0.036 and 0.19 - 0.26, respectively. Furthermore, we observe that task submission rates across different time periods are highly variable, with the standard deviation ranging from 59 - 262.

Table 4.2 shows the statistical properties of cluster attributes for the entire tracelog period. It is observable that users follow different resource utilization and submission patterns. For example, *U2* contains 0.71% of the total user population and has the highest submission rate at 2498 submissions per hour and relatively low CPU and memory requested compared to other clusters. In comparison *U3*, composing 6.37% of total users has a substantially high requested CPU and memory between 0.135 - 0.094, respectively, however exhibits the lowest submission rate, indicating this type of user infrequently submits more resource intensive tasks. *U1* which comprises the larger cluster with 37.03% of the population, submit tasks with low amount of resources requested at 0.01 and 0.016 for CPU and memory, respectively.

Table 4.2 Statistical Properties of User Clusters for Entire System.

		Requested CPU			Requested Memory			Submission Rate(Hourly)		
Cluster	**User%**	***Mean***	***Stdev***	***Cv***	***Mean***	***Stdev***	***Cv***	***Mean***	***Stdev***	***Cv***
U1	37.03	0.01	0.004	0.388	0.016	0.013	0.854	34.94	94	2.691
U2	0.71	0.016	0.011	0.689	0.019	0.013	0.658	2498	2034.6	0.814
U3	6.37	0.135	0.048	0.358	0.094	0.136	1.453	4.71	10.82	2.295
U4	6.37	0.025	0.018	0.718	0.092	0.031	0.342	13.49	19.47	1.444
U5	22.63	0.063	0.011	0.168	0.03	0.02	0.648	73.4	170.44	2.322
U6	26.89	0.032	0.006	0.197	0.014	0.01	0.752	43.63	105.18	2.411

We observe that requested CPU and memory across most clusters exhibits low variance, with an average *Cv* of 0.42 and 0.79, respectively (*U3* requested memory appears to have higher variance due to the strong influence of the three specific users requesting massive amounts of resources discussed above). The attribute submission rate exhibits highly variant behaviour across all user clusters, with an average *Cv* of 1.97. *U2* is the only user cluster whose *Cv* submission rate is less than 1, which is likely due to a small cluster population size of 3.

There are several reasons for the above observations. First, a key characteristic of Cloud computing is the ability for users to pursue diverse business objectives as discussed in Chapter 2.2.2. The results presented in this section substantiate this claim and demonstrate that user behavioural patterns not only vary by the number created clusters and statistical properties of their respective attributes, it also exemplifies that different types of users interact with the Cloud datacenter at different time periods. Second, the submission rate is outside the boundaries of the system and is entirely driven by user demands; such behaviour is reflective of the identified characteristics of Cloud computing, which provides the illusion of infinite resource to users [43] allowing them to submit as many tasks as required without conscious thought concerning system limitations. This is exemplified by *Cv* > 1 for clusters for the entire system as well as significant differences in submission rates across different observation periods as demonstrated in Tables 4.2 and 4.1, respectively. Users that interact with the system at different time periods will produce different volume of tasks in order to satisfy their business needs, while on the other hand due to constraints on physical resources consumed per task, requested resource of CPU and memory appears to be more stable reflected by a *Cv* =< 1 for the majority of clusters, as well as users in an observation period.

Table 4.3 Proportion of Task Clusters Population %

Cluster	**Month**	**Day 2**	**Day 18**	**Day 26**
T1	25.04	15.82	25.61	22.07
T2	1.38	1.8	1.84	1.99
T3	73.59	82.38	72.55	75.94

4.3.2 Tasks

Figure 4.2(a-d) presents the *k*-clusters for tasks across all observational periods. In contrast to users, it is observable that the cluster shapes are visually similar across all observational periods and demonstrate that it is possible to define three clusters for all observational periods where $f(k) < 0.85$. This similarity in cluster shapes is further exemplified when considering Cluster 2 (*T2*) introduces the largest variability to the overall cluster shape yet composes less than 2% of the total task population in comparison to *T3* which contains over 70% of the task population across all time periods as shown in Table 4.3. From Table 4.3, we can also observe that the proportion of tasks within the clusters stay relatively constant across the observational periods in comparison to user clusters, ranging between 15-25%, 1.3-2% and 72-74% for *T1*, *T2* and *T3*, respectively.

Table 4.4 presents the statistical properties of the attributes length, CPU and memory utilization for all clusters across the four observational periods. It is possible to make a more balanced comparison of tasks over different time periods in contrast to users due to the same number of clusters identified across all time periods for the latter; *T3* contains the lowest values for CPU, memory and length while *T2* contains the highest values while exhibiting more variant behaviour.

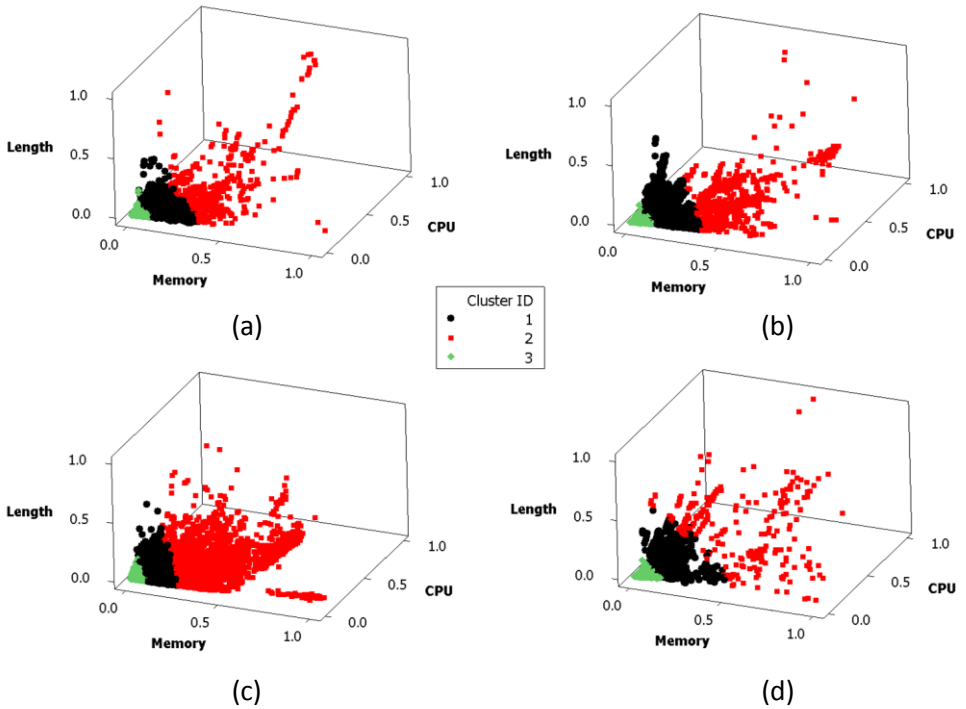


Figure 4.2 Clusterisation of tasks (a) Entire Month, (b) Day 2, (c) Day 18, (d) Day 26.

Table 4.4 Statistical Properties of Task Clusters

		Month			**Day 2**		
Parameter	**Cluster**	**Mean**	**Stdev.**	***Cv***	**Mean**	**Stdev.**	***Cv***
CPU	*T1*	0.029	0.028	0.966	0.029	0.025	0.862
	T2	0.095	0.088	0.926	0.071	0.071	1
	T3	0.006	0.012	2	0.007	0.012	1.714
Memory	*T1*	0.011	0.01	0.909	0.013	0.01	0.769
	T2	0.049	0.031	0.633	0.047	0.021	0.447
	T3	0.002	0.003	1.5	0.003	0.003	1
Length	*T1*	16,605,683	32,753,760	1.972	9,787,032	1,551,9963	1.586
	T2	123,974,450	250,146,799	2.018	30,932,490	40,683,248	1.315
	T3	739,117	4,056,404	5.488	245,445	655,190	2.669
		Day 18			**Day 26**		
		Mean	**Stdev.**	***Cv***	**Mean**	**Stdev.**	***Cv***
CPU	T1	0.028	0.014	0.492	0.006	0.006	1
	T2	0.076	0.051	0.667	0.065	0.04	0.615
	T3	0.005	0.005	0.984	0.026	0.012	0.462
Memory	*T1*	0.009	0.006	0.632	0.001	0.001	1
	T2	0.040	0.017	0.428	0.031	0.018	0.581
	T3	0.001	0.001	1.075	0.009	0.004	0.444
Length	*T1*	41,329,800	103,613,335	2.507	13,669,736	16,538,165	1.21
	T2	117,493,568	388,077,476	3.303	82300581	54,360,253	0.661
	T3	7,658,844	25,068,810	3.273	613,803	1,450,884	2.364

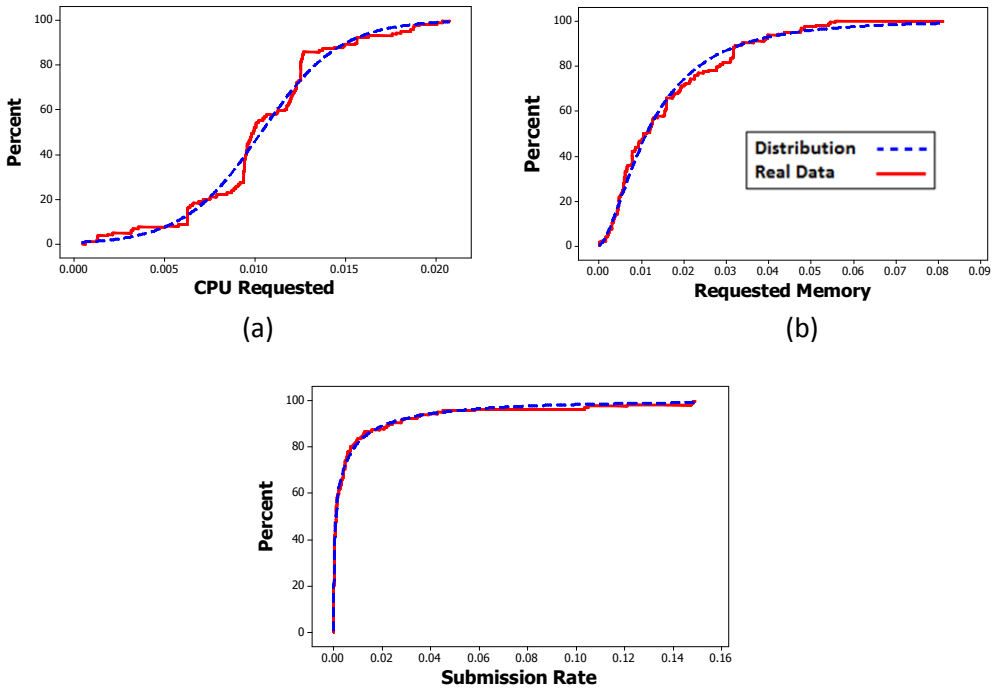
Similar to submission rates of users, we observe that task length is highly heterogeneous across all clusters and observational periods with an average *Cv* of 2.36, indicating high variation between values. This is due to similar reasons user submission rates variability; task length is an attribute that has less dependence on the Cloud infrastructure and exists outside the boundaries of the system environment, and is entirely dependent on the demands of the user (i.e. users execute tasks of different execution length in order to meet their QoS demands). In addition, similar to user resource estimation, CPU and memory utilization are less variable due to application domain constraints imposed by the system environment, reflected by an average *Cv* value of 0.93 and 0.83 for CPU and memory utilization, respectively.

4.4 Distribution Analysis

4.4.1 Users

Figures 4.3, 4.4 and 4.5 present the Cumulative Distribution Function (CDF) for *U1*, *U5* and *U6*, respectively to demonstrate the similarity between the theoretical distribution and the empirical data using the fitting process and AD test detailed in Chapter 4.2.5. The best fit distribution for each attribute within a user cluster, their corresponding AD as well as the distribution parameters which define the distribution shape are presented in Table 4.5.

Inspecting the different types of distributions and their respective parameter values, we observe further statistical evidence of inherit diversity of user behaviour. From Table 4.5, it is observable that the best-fit distribution for requested CPU varies between Logistic, 3-Parameter Weibull and Loglogistic and Wakeby. Memory is equally heterogeneous, ranging from 3-Parameter Lognormal, 3-Parameter Loglogistic and Weibull. Such results present insight into the nature of how different types of users request resources based on their business objectives and technical requirements. For example, the Wakeby distribution used for *U3* and *U5* (displayed in Figure 4.4(a)) demonstrates that a



(a) (b)

(c)

Figure 4.3 CDF of *U1* parameters (a) CPU requested, (b) Memory requested, (c) Submission Rate.

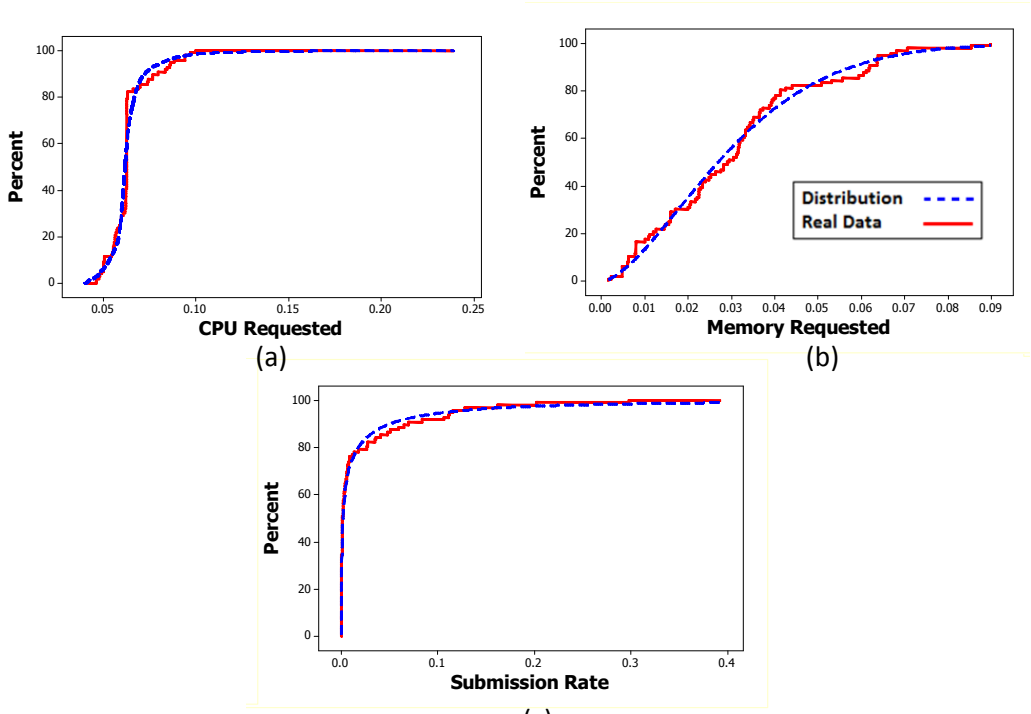


Figure 4.4 CDF of *U5* parameters (a) CPU requested, (b) Memory requested, (c) Submission Rate.

large portion of requested CPU is homogenous across these specific types of users. On the other hand, requested CPU and memory of *U4* is represented as 3-Parameter Weibull, signifying that a large portion of users in the analyzed

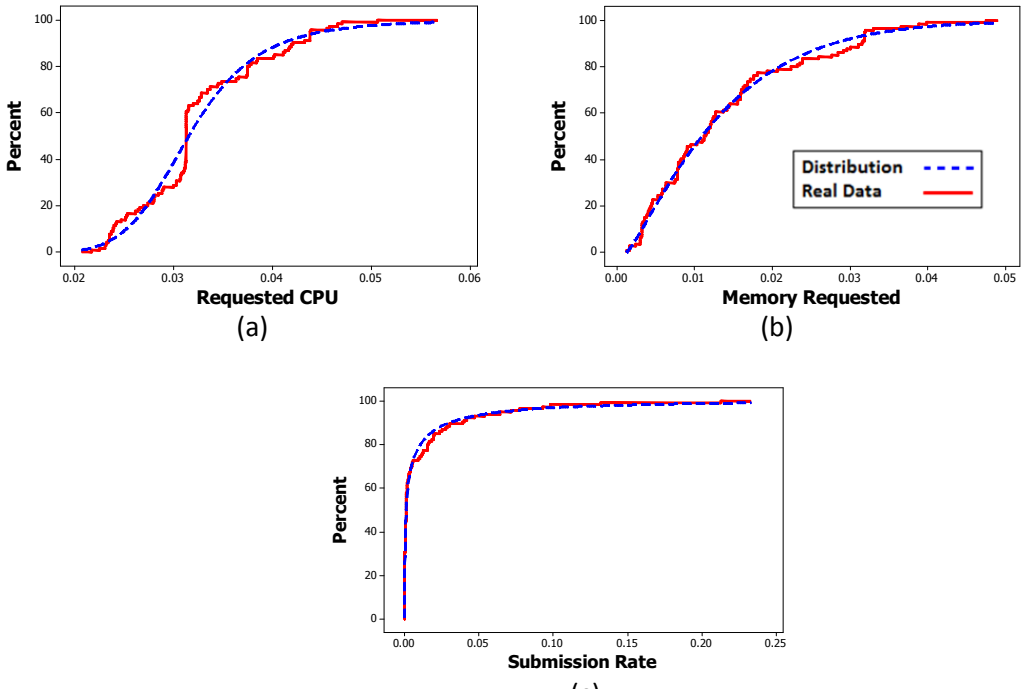


Figure 4.5 CDF of *U6* parameters (a) CPU requested, (b) Memory requested, (c) Submission Rate.

Table 4.5 Best Fit Distribution Parameters of User Clusters for Entire System

	Distribution (AD Value)	**Parameters**	**Distribution (AD Value)**	**Parameters**	**Distribution (AD Value)**	**Parameters**
Cluster	**Requested CPU**		**Requested Memory**		**Submission Rate (Hourly)**	
U1	Logistic (1.911)	μ = 0.0103 σ = 0.00216	3P Lognormal (0.875)	μ = -4.355 σ = 0.802 T = -1.60E-3	3P Weibull (0.278)	k = 0.372 λ = 0.0024 T = 3.86E-7
U2	Normal (0.431)	μ = 0.016 σ = 0.0109	Normal (0.191)	μ = 0.01916 σ = 0.01261	Lognormal (0.471)	μ = -0.5679 σ = 0.7496
U3	Wakeby (5.620)	α = 41.734 β = 334.62 δ = 0.973 ξ = 0.00 γ = 0.0003	3P Loglogistic (0.876)	α = 2.156 β = 0.06334 T = -6.1E-3	3P Weibull (1.591)	k = 0.2546 λ = 7.798E-4 T = 3.95E-7
U4	3P Weibull (0.742)	k = 1.190 λ = 0.02372 T = 2.903E-3	3P Weibull (0.342)	k = 1.095 λ = 0.0392 T = 0.0541	3P Lognormal (0.577)	μ = -6.757 σ = 1.779 T= -1.32E-4
U5	Wakeby (4.212)	α = 0.22515 β = 11.859 γ = 0.00383 δ = 0.38933 ξ = 0.0395	Weibull (0.629)	k = 1.570 λ = 0.03392	3P Weibull (0.367)	k = 0.338 λ = 0.0043 T = 3.6E-7
U6	3P Loglogistic (2.171)	α = 5.4452 β = 0.01896 γ = 0.01256	3P Weibull (0.563)	k = 1.186 λ = 0.0132 T = 1.20E-3	3P Weibull (0.523)	k = 0.034 λ = 0.0026 T = 3.86E-7

environment request smaller portions of CPU and memory with a small proportion of users requesting larger amounts.

The best-fit distributions for submission rates of user clusters predominantly follow 3-Parameter Weibull and 3-Parameter Lognormal. In conjunction with the parameter values, we observe that this data distribution is right-skewed as depicted in Figure 4.3(c) and 4.5(c), indicating that the Cloud environment is composed by the majority of users submitting small numbers of tasks, and a minority of users submitting a large proportion of tasks (for example, there exists one user that submits approximately 16% of the total tasks shown in Figure 3.3 within Chapter 3.3).

4.4.2 Tasks

Figures 4.6-4.8 and Table 4.6 present the CDF and statistical properties of the distributions for tasks clusters across the entire trace log period, respectively. We observe that CPU and memory utilization across the three clusters follows a number of distributions including General Extreme Value, Weibull, 3-Parameter

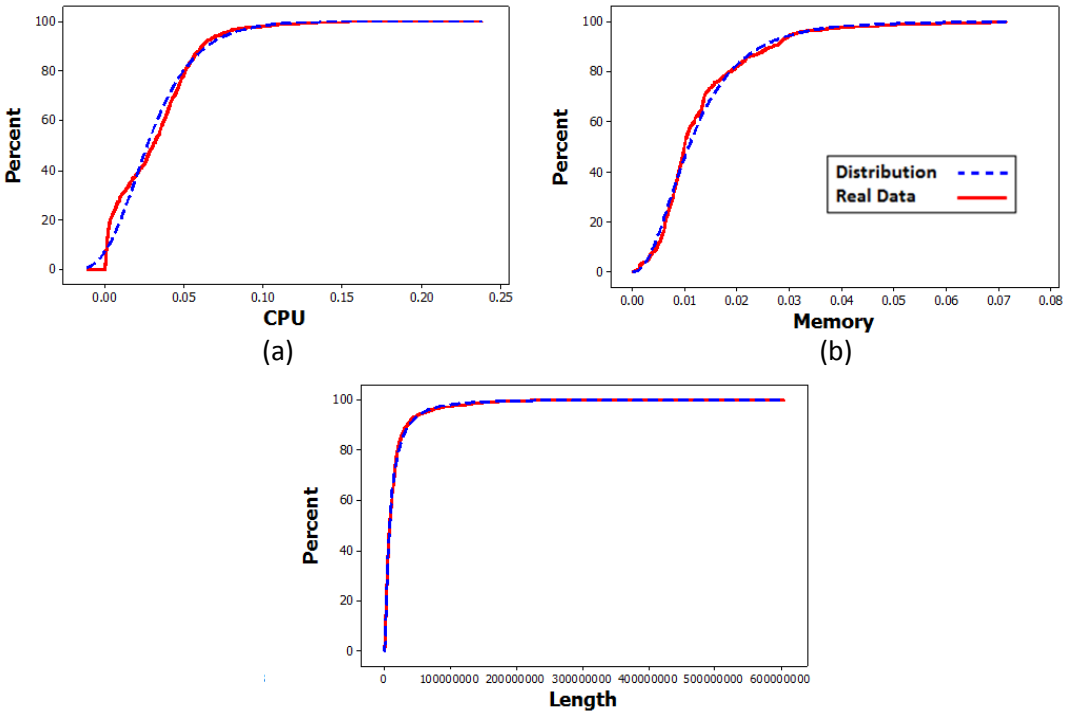
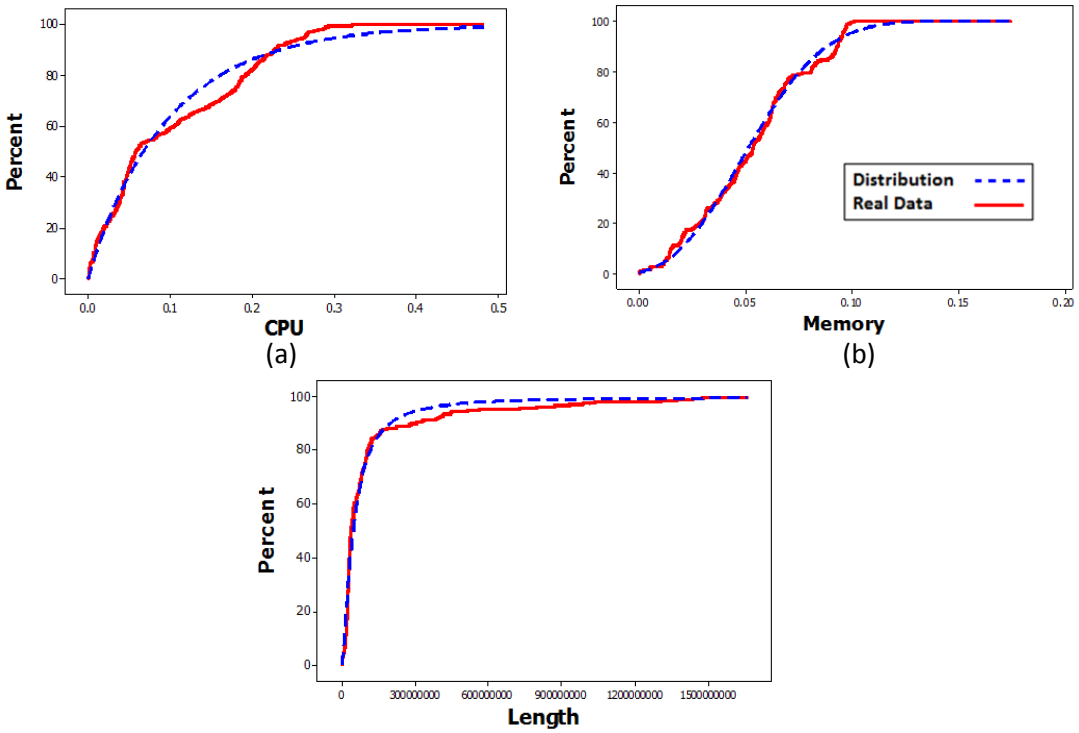


Figure 4.6 CDF of *T1* parameters (a) CPU, (b), Memory, (c) Length

Weibull and 3-Parameter Lognormal, indicating that a high proportion of tasks consume machine resources at lower rates as shown in Figure 4.8(a) and 4.6(b) for CPU and memory, respectively. Task length shares similar behaviour to user submission rates indicated by right-skewed values, signifying that most tasks exhibit short to medium execution duration as depicted in Figure 4.6-4.8(c).

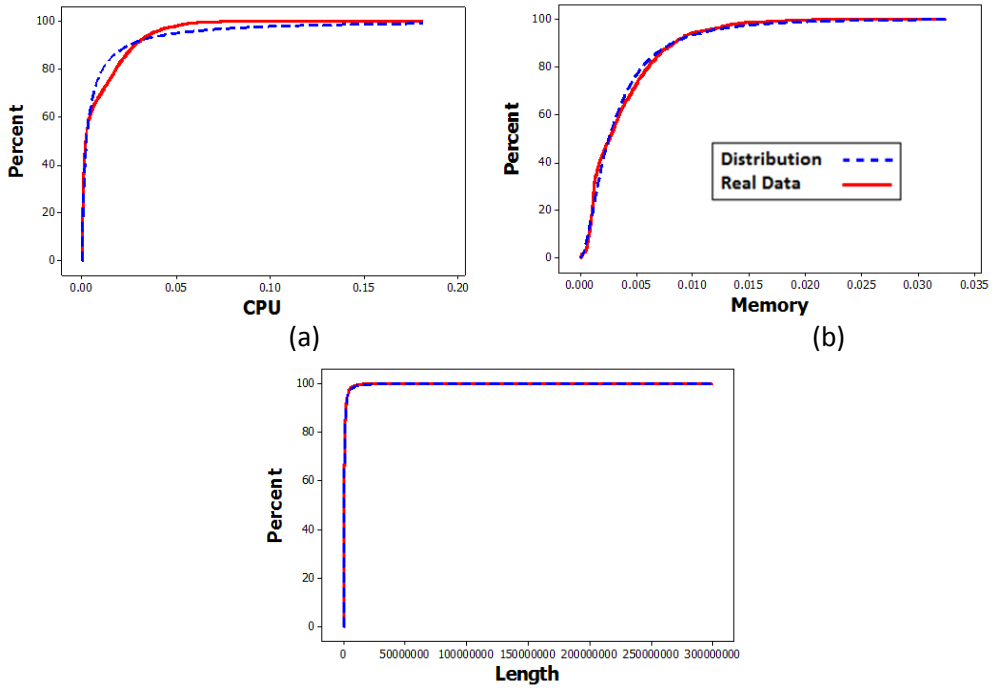
Moreover, we present a comparison of distributions for each cluster across the four selected time observational periods, as shown in Table 4.7; this is made feasible as every observational period is capable of producing three task clusters which satisfies $f(k) < 0.85$. An observation of interest is that individual time periods exhibit different best fit different distributions within that time frame. For example, Day 18 is composed of Loglogistic and Lognormal distributions for all parameters, while Day 26 is predominantly composed of 3-Parameter Lognormal and Gamma distributions. Moreover, we observe that task length appears to exhibit the most consistent distributions characteristics within selected time observations, predominately following Lognormal and 3-Parameter Lognormal. Furthermore, there exists homogeneity of distribution characteristics for task length across different observational periods, with time periods following the same distribution family (Lognormal and 3-Parameter Lognormal).



(a) (b) (c)

Figure 4.7 CDF of *T2* parameters (a) CPU, (b), Memory, (c) Length

When a task is executing within a server, it is possible to utilize no resources for a period of time due to the nature of its application. As a result, it is important to model the probability of no utilization for CPU or Memory utilization within the system. From the analysis, there exists a probability of 6-32% and 7-46% for CPU



(a) (b) (c)

Figure 4.8 CDF of *T3* parameters (a) CPU, (b), Memory, (c) Length

Table 4.8 Best Fit Distribution Parameters of Task Clusters for Entire System

Cluster	Distribution (AD Value)	Parameters	Distribution (AD Value)	Parameters	Distribution (AD Value)	Parameters
	CPU		Memory		Length	
T1	General Extreme Value (5.323)	ξ = -0.016 σ = 0.02098 μ = 0.01954	3P Lognormal (6.946)	μ = -4.342 σ = 0.569 T = -2.39E-4	Lognormal (12.048)	μ = 15.83 σ = 1.240
T2	Weibull (16.934)	k = 0.9594 λ = 0.09795	3P Weibull (3.203)	k = 2.528 λ = 0.0703 T = -9.29E-3	3P Loglogistic (10.692)	μ = 17.70 σ = 0.640
T3	3P Lognormal (15.934)	μ = -6.120 σ = 1.897 T = 6.41E-6	3P Lognormal (2.756)	μ = -5.907 σ = 0.877 T = -2.20E-4	3P Lognormal (8.045)	μ = 11.87 σ = 1.855 T = -255.9

Table 4.7 Best Fit Distribution Comparison for Task Clusters

Parameter	Cluster	Month	Day 2	Day 18	Day 26
CPU	*T1*	Gen. Extr. Value	Normal	Loglogistic	3P Lognormal
	T2	Weibull	Weibull	Lognormal	Lognormal
	T3	3P Lognormal	Lognormal	Lognormal	Gamma
Memory	*T1*	3P Lognormal	Lognormal	Lognormal	Gamma
	T2	3P Weibull	Normal	Loglogistic	3P Gamma
	T3	3P Lognormal	Lognormal	Loglogistic	3P Lognormal
Length	*T1*	Lognormal	Lognormal	Loglogistic	3P Lognormal
	T2	3P Loglogistic	Lognormal	Lognormal	3P Lognormal
	T3	3P Lognormal	Lognormal	Lognormal	3P Lognormal

Table 4.6 Probability of 0 for Task Resource Utilization

Cluster	CPU p(0)	Memory p(0)
T1	0.121	0.156
T2	0.069	0.073
T3	0.322	0.460

and memory, respectively to utilize no resources as shown in Table 4.8. We observe that there is a relationship between the probability of zero utilization and the cluster type; *T3* are characterized as small, low utilization tasks and exhibit a higher probability of zero utilization between 32-42% while *T2*, characterized as high utilization and execution length tasks have a probability of 6-7% zero utilization.

4.5 Impact of Workload Behaviour on Cloud Environment

These results highlight a number of important findings about the nature of Cloud workload in large-scale production systems and are summarised as follows:

Workload in Cloud computing datacenters are quantifiably heterogeneous and diverse. The results of the cluster and distribution analysis demonstrate statistical evidence of significant heterogeneity of Cloud workload in terms of resource estimation and submission patterns for users, and resource utilization and execution length for tasks. This heterogeneity does not only include different types of classifications of users and tasks with their own defined submission and utilization patterns, but also includes diversity of users and task behavioural patterns temporally across different time observational periods.

Users exhibit substantially more heterogeneous behaviour than tasks. From studying the characteristics of the clusters describing user behaviour across different time periods as well as their respective statistical and distribution properties, it is observable that users are more heterogeneous than tasks. This is represented by variance of the number of *k* clusters, as well as the distribution and statistical properties of attributes across different observation periods in comparison to tasks. As users are responsible for the characteristics of tasks, such findings signify that predicting future behaviour of workload requires not only the analysis and modelling of task behaviour, but must also include user submission and resource request patterns which currently are neglected in the current state-of-the-art.

Workload attributes which are defined outside the boundaries of the system environment introduce the highest level of heterogeneity. This is demonstrated by the attributes user submission rate and task execution length attributes exhibiting highly variant behaviour statistically in comparison to CPU and memory requested and utilization for users and tasks, respectively. The diversity of workload imposed by these two attributes introduces potential challenges in workload prediction which rely on the use of historical data, as the expiration time of the representativeness and accuracy of historical data is reduced. Therefore, the use of other techniques such as neural networks [174] that are capable of adapting to significant changes and evolution of Cloud workload behaviour may be more effective.

There exists two distinct types of task characteristics within clusters. From the intra-cluster analysis and studying the internal characteristics of the attributes within each task cluster, we observe that there predominantly exist two types of

tasks; A large proportion of tasks with small resource utilization and execution length, and a small proportion of tasks which utilize large amount of resources have larger execution lengths.

4.6 Results Validation

In order to characterize and analyze the performance of similar large-scale Cloud datacenters under a projected set of operating conditions, Solis Moreno [173] implemented the task and user model parameters derived from the analysis presented in this chapter as an extension to the CloudSim framework [168] [170][171]. CloudSim is a Java based framework that enables the simulation of complete Cloud Computing environments [169], providing abstraction for all the components within the Cloud computing model and their interactions. As discussed in Chapter 2.4.1, the quality and accuracy of simulation results are entirely dependent on how accurately the introduced parameters reflect the analyzed system in reality.

4.6.1 Simulation Validation

Model validation is defined as the "substantiation that a computerized model with its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model" [102]. In the situation where the analyst does not have access to the real system or to a different dataset sample from the same system, a common validation technique is to use a portion of the available data to construct a model and then use the remaining data to determine and validate whether the model behaves in the same manner as the real system. This is typically addressed by sampling the analyzed tracelog where both the input and the actual system response are collected from the same period of time [172]. According to Sargent, et al. [102], there are two approaches to compare the simulation model to the behaviour of the real system; the first is to use graphs to empirically evaluate the outputs while the second involves the application of statistical hypothesis tests to make an objective decision.

Table 4.9 Simulation Results for Proportions of Cloud Datacenter Components.

Component	Mean Sim.	Std Dev.	*Cv*	95% CI	Avg. Error
U1	36.981	2.943	7.958	(34.40,39.56)	0.047
U2	0.472	0.167	35.355	(0.32,0.61)	0.236
U3	5.613	0.840	14.974	(4.87,6.34)	0.755
U4	6.226	1.186	19.053	(5.18,7.26)	0.142
U5	23.538	1.086	4.614	(22.58,24.49)	0.896
U6	27.170	3.574	13.156	(24.04,30.30)	0.238
T1	24.127	1.159	4.803	(23.11,25.14)	0.909
T2	1.355	0.067	4.945	(1.29,1.41)	0.021
T3	74.518	1.120	1.503	(73.53,75.50)	0.930
S1	1.063	0.078	7.364	(0.99,1.13)	0.062
S2	6.312	0.310	4.912	(6.04,6.58)	0.006
S3	0.023	0.007	29.881	(0.01,0.03)	0.001
S4	30.502	0.199	0.651	(30.32,30.6)	0.198
S5	7.998	0.150	1.872	(7.86,8.12)	0.043
S6	53.553	0.282	0.527	(53.30,53.8)	0.053
S7	0.047	0.021	44.821	(0.02,0.06)	0.007
S8	0.455	0.021	4.597	(0.43,0.47)	0.042
S9	0.042	0.017	40.000	(0.02,0.05)	0.002
S10	0.005	0.007	149.071	(0.00,0.01)	0.003

The modelled parameters for users and tasks attributes derived from the analysis use both techniques. Furthermore, the proportion of the number of tasks, users, server population and task priorities within the trace log were also simulated and compared against the empirical data using the absolute error between the average output of the simulations and the empirical data.

The results from the simulation experiments which can be found in [173] demonstrate the accuracy of the models derived from the analysis and represent the operational characteristics of workload within the Cloud computing datacenter. Figure 4.9 illustrates the proportions of Cloud components (users, tasks and servers) generated by the simulator contrasted against the empirical data from the trace log. By comparing the average simulation outputs, we observe that the simulated proportions of Cloud components consistently match those from the trace log. Comparing the average simulation outputs with the real values, it is possible to observe that simulated proportions of components consistently match the proportions of the components in the actual system. Furthermore, Table 4.9 presents statistically generated proportions where it can be observed that for the generation of users, the average absolute error is calculated at 0.39% while for tasks and servers is calculated as 0.62% and 0.04%,

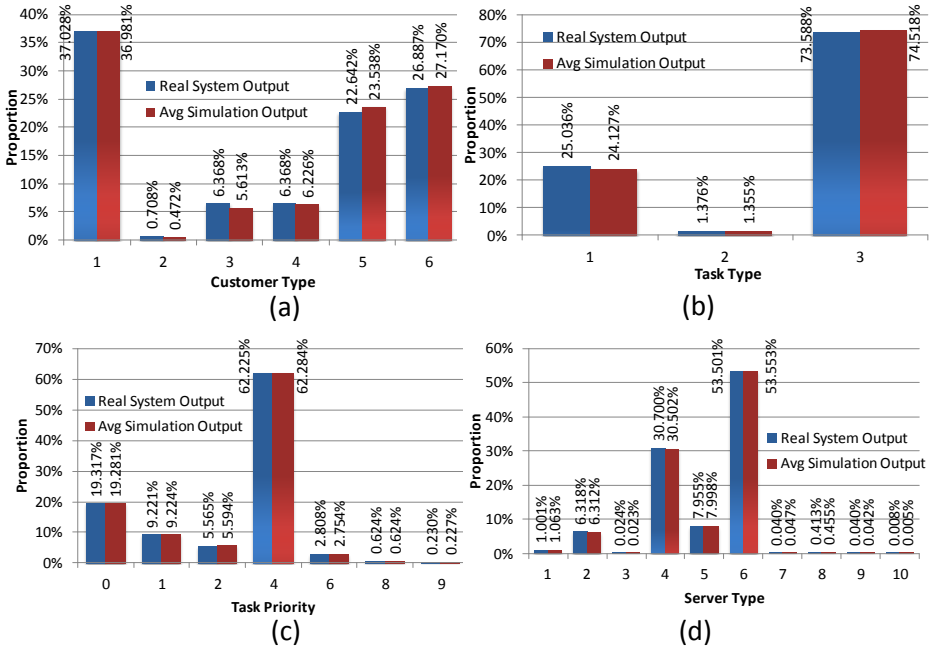


Figure 4.10 Comparison of proportions of real and simulated data for (a) users, (b) tasks, (c) task priority, (d) servers.

respectively. It is also observable that in all the cases the difference between the simulated and real system proportions is lower than 1%.

Furthermore, it is possible to compare simulated and real user and task patterns by using the empirical CDF of the real data for each cluster parameter compared against the empirical CDF of their simulated outputs. These distributions are

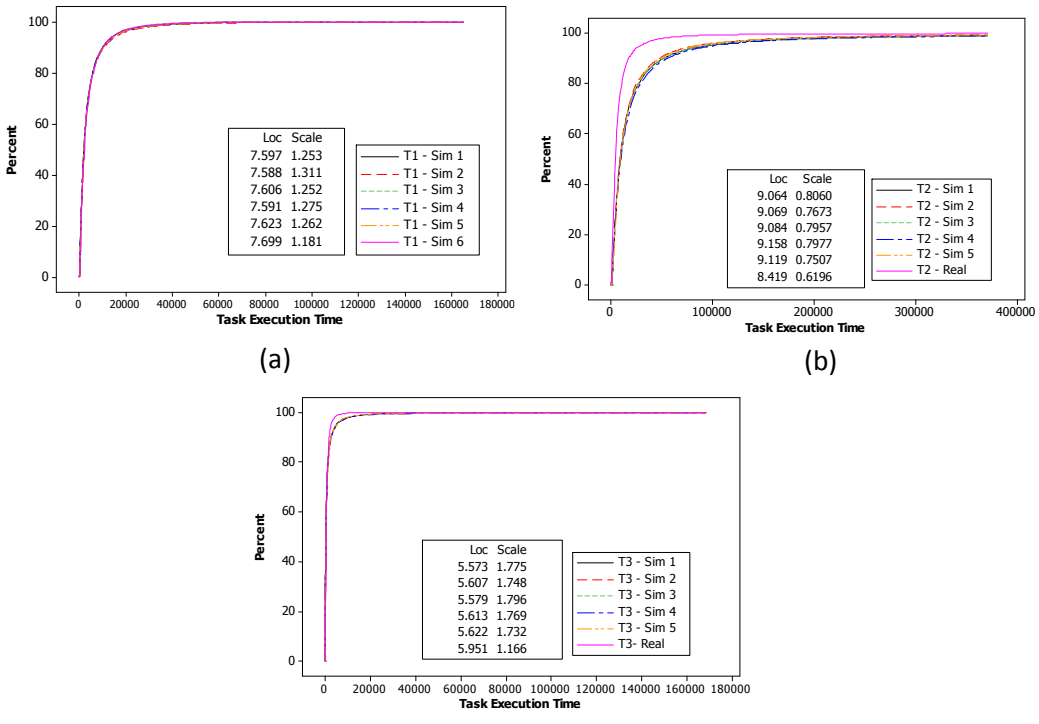


Figure 4.9 CDF of task patterns between real and simulated data of task execution time (seconds) for (a) CPU (s), (b) memory, (c) length.

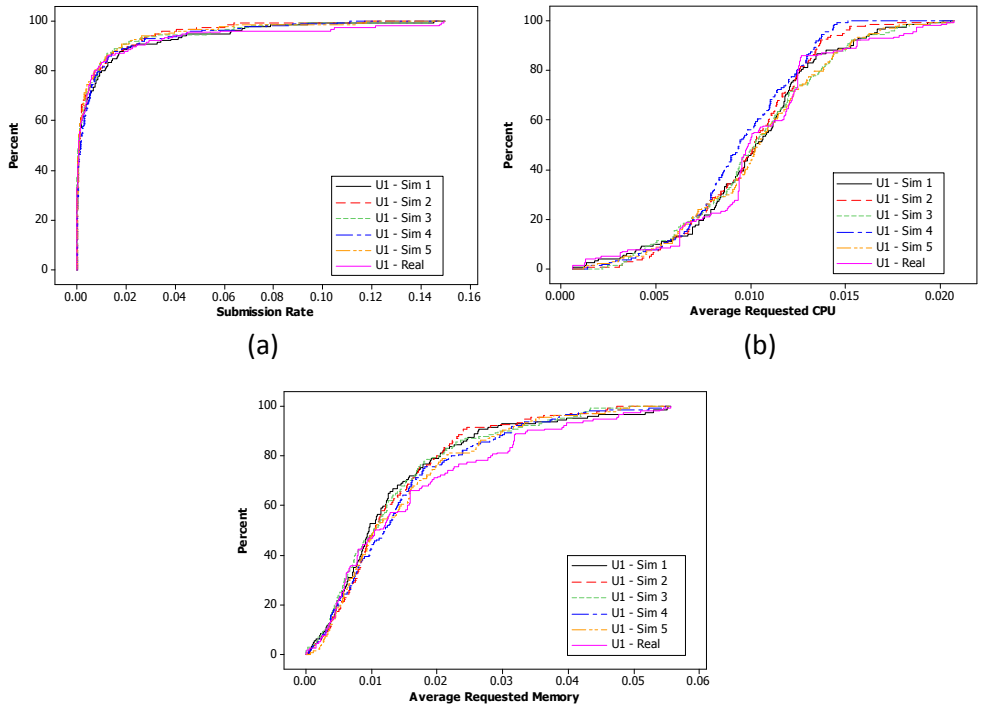


Figure 4.11 CDF of user patterns between real and simulated data for *U5*
(a) Requested CPU, (b) Requested memory, (c) Submission rate.

exemplified in Figures 4.10 and 4.11 with the parameters of *U1* and *T3*, respectively representing the largest populations for each element in the trace log. From these figures, it is observable that simulated component patterns are consistent with those observed within the real data.

As discussed in [173], further rigorous statistical testing on the statistical significance of the simulation outputs were performed using the *p*-Value test [178] and Mann Whitney [177] in order to compare two non-parametric populations and to verify whether the rejections are statistically significant given the variances reported, respectively.

4.6.2 Improvement of CPU Consumption Patterns

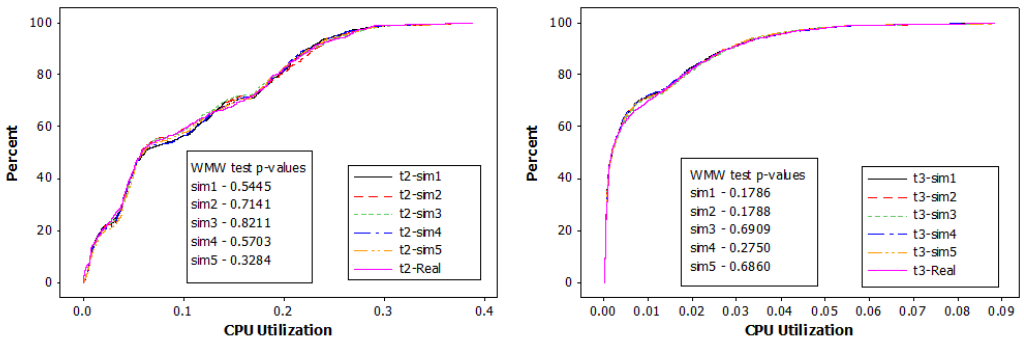
As observed from the distribution analysis discussed in Chapter 4.4.2, there are potential disparities between the empirical data and the theoretical distributions. This is most observable for Requested CPU and CPU utilization for *U1* and *T1*, respectively as shown in Figures 4.3(a) and 4.7(a), potentially leading to inaccuracies in simulating system behaviour. This is due to the nature of the real operational system behaviour of the system in the event of challenges fitting probability distributions of parameters sufficiently using the GoF test. The inaccuracies in the CPU utilization patterns observed in *T2* and *T3* are the result

Table 4.10 Sub-regions distribution fitting to improve CPU utilization for *T2* and *T3*

Cluster	Distributions	Parameters	Region Proportion
T2	Gen. Extreme Value	μ= 0.00593, σ=0.00583, ξ= -0.01822c	22.90%
	3-Parameter Lognormal	μ =-2.9072, σ =0.20621, T =-0.00888	32.44%
	Gen. Extreme Value	μ = 0.11193, σ =0.0242, ξ= -0.20605	16.10%
	3-Parameter Weibull	k=1.3318, λ=0.05718, T=0.16661	28.56%
T3	3-Parameter Lognormal	μ =-7.7268, σ =0.64993, T =-4.9626E-5	45.34%
	3-Parameter Weibull	k =0.89629, λ =0.00364, T=0.00136	28.21%
	3-Parameter Weibull	k =1.1097, λ =0.0152, T=0.01314	26.45%

of multimodal data distributions. Consequently, this makes fitting such datasets to a single theoretical distribution unsuitable and results in a significant gap between values found in simulated and real data as observed in Figure 4.10(b). To improve the accuracy of our model, we applied "multi-peak histogram analysis for region splitting" [181] and fitted the derived dataset sub-regions to new parametrical distributions. This is achieved by splitting the data based on the lowest points of the different valleys observed within a histogram of the parameter of interest. To identify the peaks and valleys of a multimodal dataset observed within a histogram, it is smoothed by applying the LOWESS [180] (Locally-Weighted Scatterplot Smoother) technique. The derived sub-regions are then fitted to new parametrical distributions following the same method described in Chapter 4.2.

Consequently, the CPU utilization patterns of affected clusters are comprised of a combination of different distribution families which are sampled by the model



(a) (b)

Figure 4.12 CPU utilization pattern improvement for (a)*T2* and (b)*T3*

simulator based on the proportional size of the derived sub-regions. The distribution parameters as well as the size of the obtained sub-regions are presented in Table 4.10.

The results of this method are illustrated in Figure 4.12 where it is observable that split distributions improve the fitting between the simulated and real datasets, with the error for execution time reduced from 8.07% to 0.42% and from 5.91% to 0.13% for *T2* and *T3*, respectively.

4.7 Application of Work

The proposed workload characterization model, analysis results and modelling of users and tasks presented in this paper can be used in a number of practical scenarios. First, the workload model described in Equations 1 and 2 can be extended to include any number of attributes of interest for the research objective without alterations to the analysis and modelling presented in Chapter 4.2. For example, attributes which identify the type of application executed (Database, video streaming, storage, etc.), security constraints of task placement and execution, and disk and network utilization of a task. Furthermore, the proposed method of workload characterization contains a sufficient level of generality that it can be applied to Cloud datacenter tracelogs of different specifications, as long as the system environment is understood sufficiently to create assumptions concerning workload identification.

Second, the derived validated models can be used by researchers to simulate request and consumption patterns considering attributes and patterns statistically close to those observed from a production environment. This is critical in order to improve resources utilization, reduce energy waste and in general terms support the design of accurate forecast mechanisms under dynamic conditions to improve the QoS provisioned to users. Two practical examples are outlined which use the results derived in this chapter to support the design and evaluation of two energy-aware mechanisms for Cloud computing environments.

The first is a resource overallocation mechanism that measures the user resource request patterns and the actual resource utilization of tasks submitted by users. Taking into account the attributes of requested and utilized resources it is

possible to estimate the resource overestimation patterns. From this, it is possible to exploit the resource overestimation patterns of each user type in order to smartly overallocate resources to the physical servers. This reduces the waste produced by frequent overestimations and increases datacenter availability and consequently allowing additional VMs in the same computing infrastructure, improving its energy-efficiency [174].

The second mechanism considers the relationship between VM Interference due to contention for resources and energy-efficiency of a server. The proposed model reduces the energy waste by exploiting the workload heterogeneity that exists in Cloud environments; different types of workload are co-allocated within a server based on the level of interference created when deployed together in order to improve the energy-efficiency of the datacenter. By considering the resource consumption patterns of different types of tasks, we estimate the level of interference and decrement in energy-efficiency when tasks are co-located within the same physical server. Tasks are characterized by their resource usage patterns leveraged from the findings of the analysis presented in this chapter, and the current servers' performance interference level [175]. Using the findings of realistic workload characteristics within this chapter, the proposed mechanism reduces VM interferences by 27.5% and increases energy-efficiency up to 15% in contrast to current state-of-the-art workload allocation mechanisms.

Both of these above mechanisms leverage the proposed workload model as well as the statistical properties and derived models for tasks and user behaviour from the presented analysis in order to emulate the user and tasks patterns. For the overallocation mechanism, the model integrates the relationship between user demand and the actual resource usage - essential in both scenarios where the aim is to achieve a balance between resource request and utilization in order to reduce resource waste. For the performance-interference, workload utilization patterns are exploited when calculating the performance-interference between co-located tasks within a physical server, as well as task submission rates into the system.

Another important benefit of our approach is that values of user and task attributes are represented as proportions of resources requested or consumed, agnostic of underlying hardware characteristics. As a result, the proposed model

can be used to evaluate the performance of different datacenter configurations in terms of number of servers and their respective specification under the same workload conditions.

Furthermore, the comprehensive analysis at cluster and intra-cluster level, the workload model that integrates user and tasks patterns, and the applicability of the model independently of the hardware characteristics represent unique advances in comparison with the related work previously discussed in Chapter 2. In particular, the introduction of user behavioural patterns as well as extensively modelling attributes to be used by other researchers. Additionally, the proposed model supports the assessment of resource management mechanisms such as those recently presented in [176] and [179] with parameters from a large-scale production Cloud environment.

4.8 Summary

In this chapter, we have presented an in-depth analysis of workload characteristics and behavioural patterns within Cloud computing datacenters. The method of workload characterization defining key attributes that identify and affect user and task behaviour, as well as the clustering used for classification is presented. The analysis results are divided into two main sections: cluster and intra-cluster analysis. The cluster analysis enable the study and characterisation of users and tasks over different observation periods, while the intra-cluster analysis allows modelling of users and tasks which can be leveraged for practical usage in Cloud research. The results from the cluster and intra-cluster analysis quantify the inherit diversity and heterogeneity of workload in Cloud computing datacenters, primarily driven by users submitting tasks to fulfil different business objectives as hypothesized and stated in Chapter 2.2.2. Finally, the results are validated and several practical usages of these results are discussed.

The workload model defines the relationship between users and tasks formally, and includes the identified key attributes which have substantial impact on Cloud workload behaviour. Furthermore, the method of classification for users and tasks using *k*-means clustering is explained in detail, where the number of *k* clusters is formally determined through their respective attributes in order to avoid subjectivity in selecting an optimal number for *k*. Finally, the sampling

process of the trace log, including the selection of additional time periods of observation and the assumptions of workload behaviour are explained and justified in detail.

The cluster analysis of classified users and tasks is presented and discussed in detail, including the statistical properties and cluster shapes of users and tasks across different observational periods. From the analysis it is observable that the Cloud computing environment exhibits quantifiably diverse and heterogeneous behaviour in terms of user and task characteristics, represented by the variability of the number of clusters created and statistical properties that vary temporally and spatially over different observation periods.

The intra-cluster analysis studies and models the internal characteristics and distributions of attributes for each defined user and task cluster. These distributions are discussed in detail and their respective parameters are presented so they can be leveraged by other researchers for evaluation mechanisms under realistic Cloud computing environment conditions and simulation purposes. This analysis demonstrates further evidence on quantifiable heterogeneity of workload within the Cloud, as well as defines the characteristics of tasks, which is primarily divided into two types: A large proportion of tasks consuming small amount of resources, and a small number of tasks utilization large amount of resources.

An extension to CloudSim is developed to execute a simulation environment that uses the models and statistical properties derived from the analysis in order to validate their accuracy in comparison to the analyzed trace-log. This validation includes the use of a number of techniques include *p*-value test and Mann-Whitney which are described in further detail in [173]. Finally, a solution to accurately simulation attributes for users and tasks which contain multimodal distributions is presented.

Finally, practical applications of the derived analysis results are described, including developing realistic simulation environments of Cloud computing datacenters as well as describing two energy-aware resource management mechanisms that leverage the results of this work in the domains of overallocation and performance interference.

5 Server Characteristics

5.1 Overview

This chapter presents the method and empirical analysis of server characteristics, utilization patterns and resource inefficiencies within a large-scale Cloud computing datacenter. The method of extracting and parameterizing the resource utilization and inefficiency for servers and the system assumptions made when applying the method to the case study trace log are described in detail.

Next, the analysis of resource utilization for servers is presented, in terms of different server architecture types and different time periods, respectively. Furthermore, we quantify and analyze the resource utilization wasted due to Termination Events in order to identify operational inefficiencies within the Cloud datacenter. Finally, we discuss the meaning of the results extracted, and how they can be used practically by other researchers.

5.2 Methodology

5.2.1 Utilization method

Resource utilization of individual servers can be calculated by extracting data attributes recorded within the trace log related to resource utilization such as those presented in Table 3.3 within Chapter 3.2.1. The resource utilization of a server can be understood and measured in terms of CPU, memory, disk and network usage. For the purpose of the analysis presented, this chapter focuses on CPU and memory utilization as a result of data availability discussed in Chapter 3.3.

In the best case scenario, the resource utilization of a server can be extrapolated by studying the recorded resource utilization for individual servers over a specified time frame, and is represented as the sum of resource utilization divided by a defined time frame for individual servers. However, in some scenarios, such as the Google Cloud case study, such data may be omitted from the trace log. As a result, although some traces may not explicitly provide resource utilization of servers, it is possible to calculate the resource utilization of servers from the recorded utilization of individual tasks residing within a

server in a defined time frame. In other words, the CPU and memory utilization of a server can be expressed as the sum of the resource utilization of tasks executing within a specific server within a defined time frame.

Within the context of the Google Cloud trace log, such data are available through the use of the "task resource usage" data table presented in Table 3.2 within Chapter 3.1.4 and includes recorded timestamps and resource utilization values every 5 minutes. As a result, it is possible to calculate the sum and average utilization of specific servers within a defined time frame. Furthermore, due to the normalized values of server capacities described in Chapter 3.3, utilization values for servers are adjusted in accordance to the proportion to its maximum CPU and memory capacity (i.e. a server exhibiting 30% CPU with server capacity 0.5 is represented as 60% CPU utilization).

Another aspect of importance of understanding system operation and areas of operational inefficiency is quantifying the amount of resource utilization wasted per individual servers and server architecture types within Cloud datacenters.

5.2.2 Resource Utilization Wasted

Within a Cloud datacenter, it is possible for a task to experience Termination Events which result in unsuccessful completion of task execution within a server. Such behaviour is explicitly described in the context of the case study trace log described in Chapter 3.1.3. This consequently results in work performed by the task to be lost, and consequently waste in terms of server resource utilization. A Termination Event results in a task to be resubmitted onto a server as shown in Figure 3.3 in Chapter 3.1.3. As a result, it is possible to quantify the resource utilization of useful work performed by a task to successfully complete, contrasted against the total resource utilization to complete a task including wasted work due to termination events. In other words, it is possible to quantify

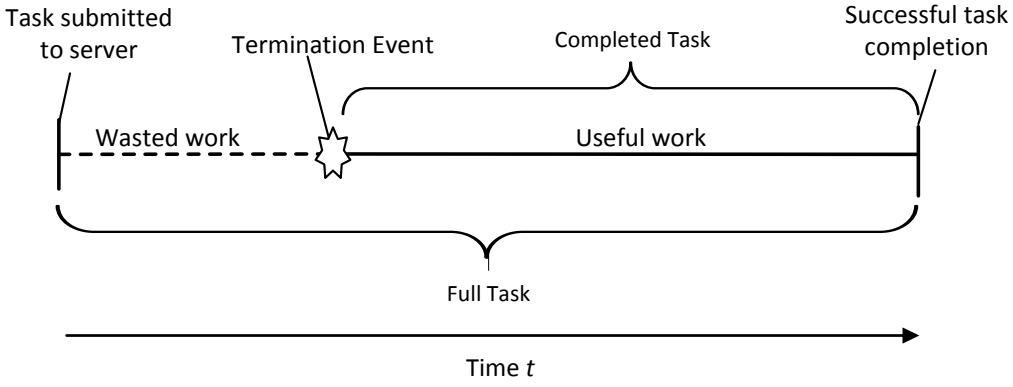


Figure 5.1 Depiction of Full and Completed task within the Cloud datacenter.

and compare the total resource utilization of completed tasks and full tasks as depicted in Figure 5.1. A *completed task* is defined as the task duration and resource utilization between the most recent task submission event and successful execution completion. A *full task* is defined as the total task duration and resource utilization from the first submission of the task into the system until successful task completion, inclusive of work performed prior to Termination Events.

Resource utilization waste can be extracted in a number of techniques as long as resource utilization of servers, Termination Events and appropriate time stamps of events are recorded within a Cloud trace log. Within the context of the Google Cloud case study, server utilization waste for CPU and memory is calculated as the sum of total computation of completed tasks subtracted from the sum of full task computation within a server in a defined time period.

5.2.3 Research Assumptions for Case Study

In order to conduct the analysis of the Google trace log, it is necessary to make the following assumptions and decisions:

- Server architectures 2,4,5, and 6 were the focus of the analysis as they represent over 98% of the total server population within the trace log as shown in Figure 3.8 in Chapter 3.3. Moreover, within these four architectures there exist 0.2% of servers which exhibit no task resource utilization, which are omitted from the analysis.
- Due to the uniform usage of Disk for tasks within the trace log discussed in Chapter 3.3, the analysis of server resource utilization and characteristics is focused on CPU and Memory.
- Tasks which do not start or finish execution within the trace log observation period are not considered, as it is not possible to characterize the behaviour of a task without both these values.
- Four observational periods were selected for the analysis of the trace log population, and were selected for a number of reasons. First, it is important to analyze system behaviour at different time periods in order to identify and investigate behavioural patterns within the system. Second, from the coarse-grain analysis discussed in Chapter 3.3, different days exhibit heterogeneous system behaviour in terms

of number of tasks and utilization. As a result, Days 2, 13, 14 and 28 were selected for analysis. Days 2 and 18 consist of high task submission rates and low and average task length, respectively, while Days 13 and 14 contain low task submission rates and average task length, respectively.

The remaining sections of this chapter present the analysis results of resource utilization and waste of CPU and memory for servers within the Google trace log. The method of extracting the statistical parameters and models of the analysis presented within this thesis follows the same technique and use of tools as discussed in Chapter 3.2.1.

5.3 Analysis Results

5.3.1 Resource Utilization

Figure 5.2 (a-d) and Figure 5.3 (a-d) present the distribution of server CPU and memory utilization for servers across four time periods, respectively. It is observable that the mean utilization of CPU and memory within the four time periods is between approximately 25-45% and around 50% for CPU and memory, respectively reflecting similar numbers reported in [157]. This level of server utilization could be seen as highly efficient in comparison to other Cloud datacenters, which report lower average utilization as discussed in Chapter 2.4.6.

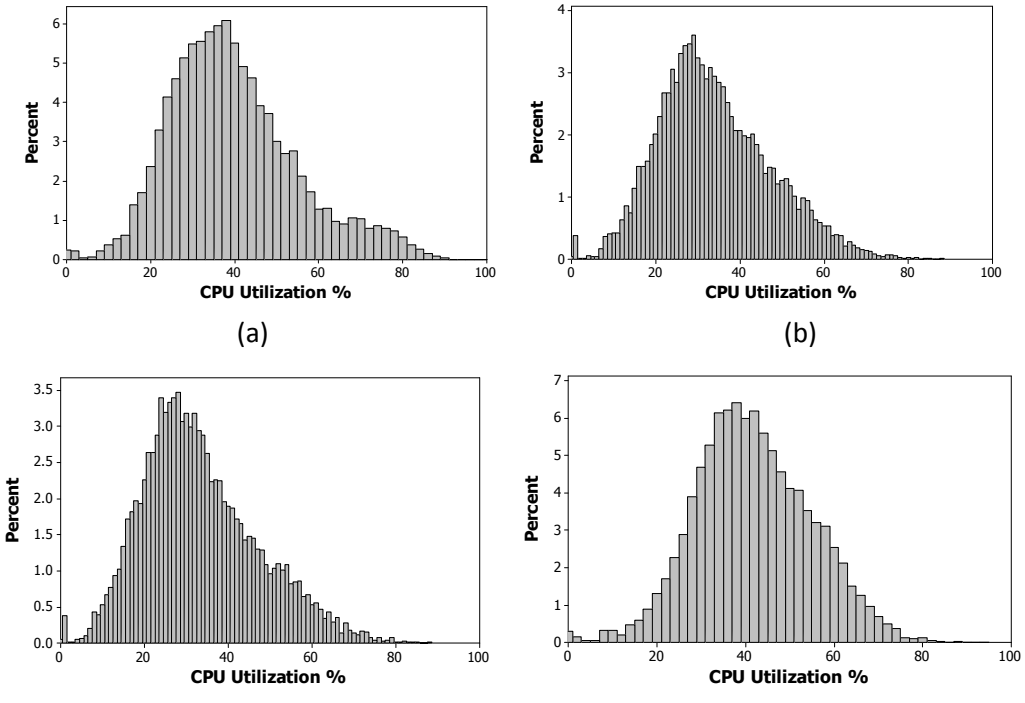


Figure 5.2. Distribution of server CPU utilization (a) Day 2, (b) Day 13, (c) Day 14, (d) Day 18.

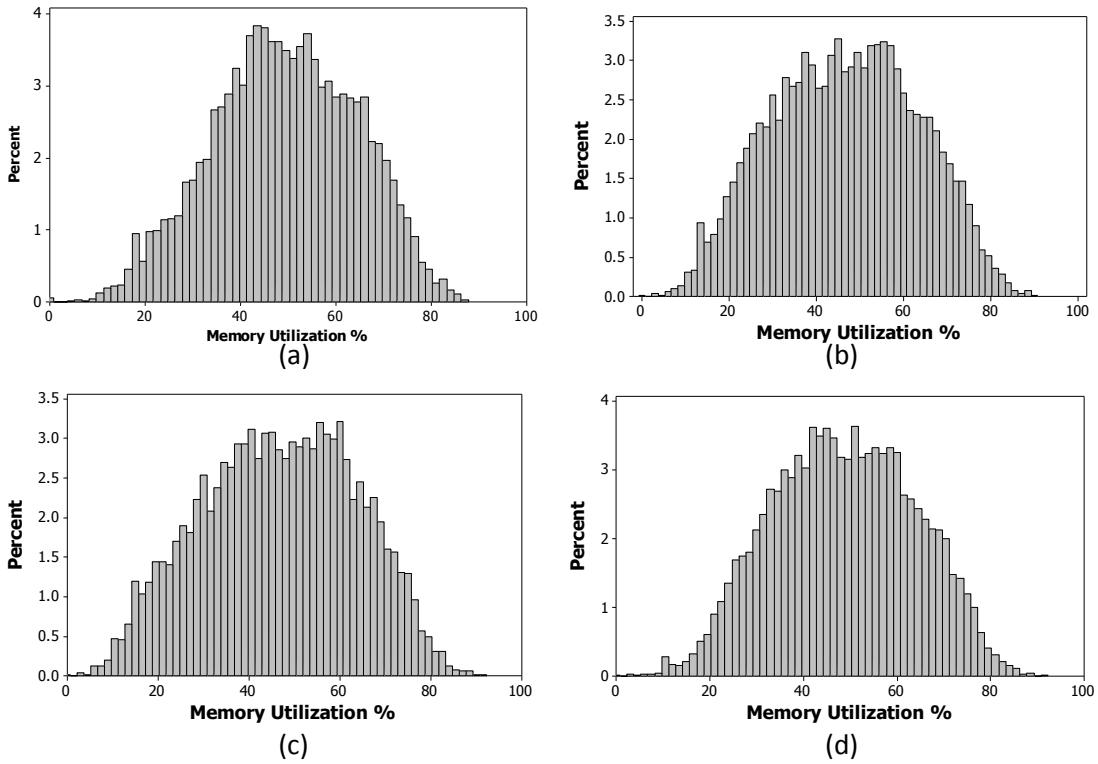


Figure 5.4 Distribution of server memory utilization (a) Day 2, (b) Day 13, (c) Day 14, (d) Day 18.

The statistical properties of CPU and memory utilization for each server architecture type is shown Table 5.1 and depicted in Figures 5.4 and 5.5, respectively. From Table 5.1 we can observe a range of average utilization patterns across different architecture types, ranging from 28.34 - 55.66% for CPU and 31.51 - 50.83% for memory, respectively. Architecture 5, the third largest server population within the datacenter, exhibits the highest CPU utilization and lowest memory utilization on average across the sampled days, while

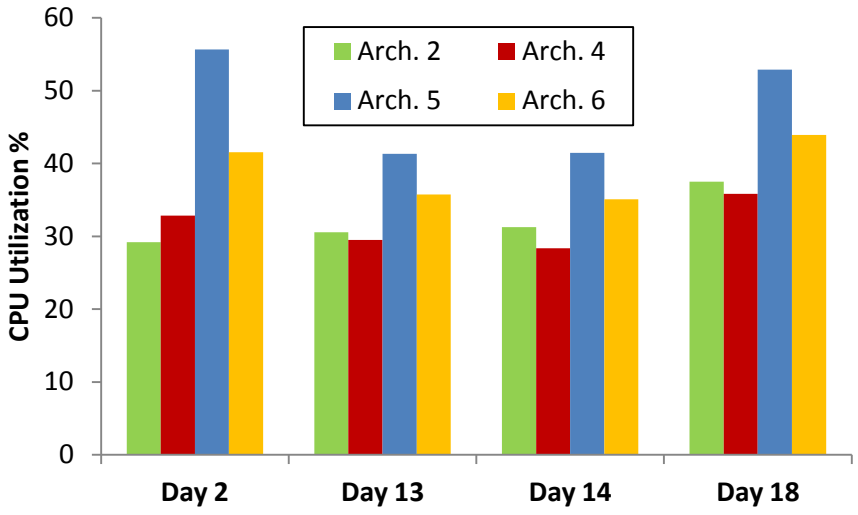


Figure 5.3 Memory Utilization of Server Architecture Types

Table 5.1 Server Architecture Resource Utilization

Server Arch.		Day 2	Day 13	Day 14	Day 18	Mean	St.Dev
Arch. 2	CPU	29.18	30.57	31.26	37.52	32.13	3.69
	Memory	32.86	29.49	28.34	35.84	31.63	3.4
Arch. 4	CPU	32.86	29.49	28.34	35.84	31.63	3.4
	Memory	50.83	49.31	48.85	50.85	49.96	1.03
Arch. 5	CPU	55.66	41.34	41.46	52.9	47.84	7.52
	Memory	39.11	31.55	31.51	34.96	34.28	3.6
Arch. 6	CPU	41.55	35.74	35.08	43.94	39.08	4.35
	Memory	49.86	47.05	47.21	49.24	48.34	1.42

Architecture 4, representing the second largest server population, contains the lowest CPU and memory utilization. Furthermore, we observe that the average utilization for architectures 6 and 5 in Days 13 and 14 is approximately 7% and 12% lower, respectively for CPU compared to Days 2 and 18, while the CPU utilization of architectures 4 and 2 remains relatively stable across all time periods. This is a result of interest when considering the rates of task submissions; Days 2 and 18 contain 2.8 times the amount of tasks submitted in comparison to Days 13 and 14, indicating a stronger correlation between task submission rate and server utilization for architectures 6 and 5 in comparison to architectures 4 and 2. This indicates that within the system, CPU utilization of server architectures are strongly influenced by the behaviour of users within the Cloud environment presented in Chapter 3.3 and Chapter 4.3.1. In comparison, memory utilization for server architectures remains relatively stable for architectures across all observation periods, suggesting a loose correlation between workload behaviour and memory utilization (as discussed in Chapter 3.3). This is worth noting, as it is intuitive to assume that all types of resource utilization within a server would be strongly correlated to the workload environment.

There are a number of reasons postulated for the above observed and analyzed behaviour. First, the utilization levels between 40-60% for servers across the system indicate the Cloud environment studied deploys a load balancing technique to keep resource utilization levels of servers balanced regardless of the behaviour of the Cloud workload submitting into the system environment. Second, while tasks may contain constraints which dictate the server

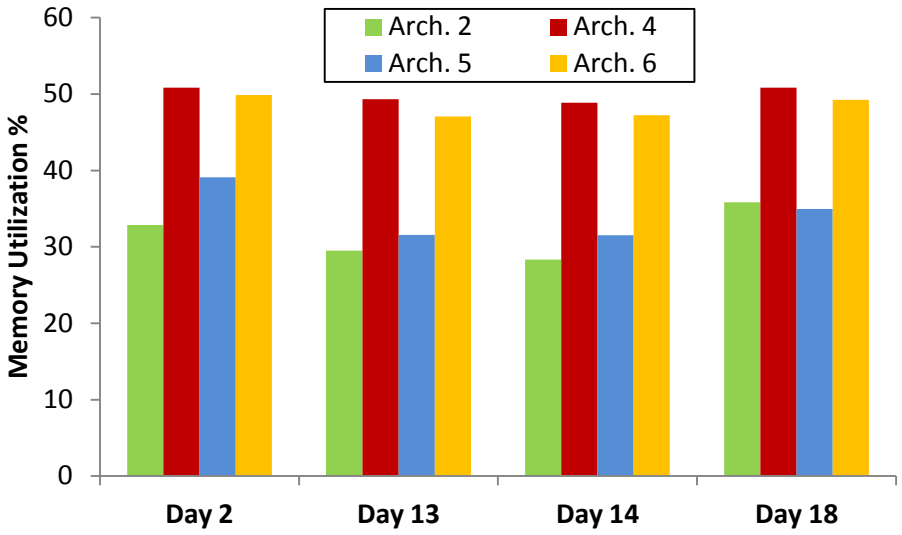


Figure 5.5 CPU Utilization of Server Architecture Types

architecture type required for execution, from our analysis only 5% of tasks possess one or more constraints for server architecture allocation.

5.3.2 Waste resource

The method described in Chapter 5.2 can be used to allow the identification and quantification of computation waste and inefficiency of servers in Cloud datacenters. Attention is now drawn to presenting examples of how this method can be applied to production systems by using the Google as the case study. Figures 5.6(a) and 5.7(a) presents the memory utilization disparity between full tasks and completed task utilization for all server architectures 4 and 2 in Day 2, respectively. Furthermore, Figure 5.6(b) and 5.7(b) present the utilization disparity for CPU within the same server architectures and time frame. It is observable that there exists a substantial level of resource utilization disparity between full tasks and completed tasks, and that different architecture types within the same time frame exhibit different levels of disparity as seen in Table 5.2. There is a 4.53 - 14.22% and 1.29 - 7.61% resource disparity between full and completed tasks for CPU and memory utilization, respectively for different server architectures. Furthermore, from these results we can observe that CPU waste appears to be more variant compared to memory utilization.

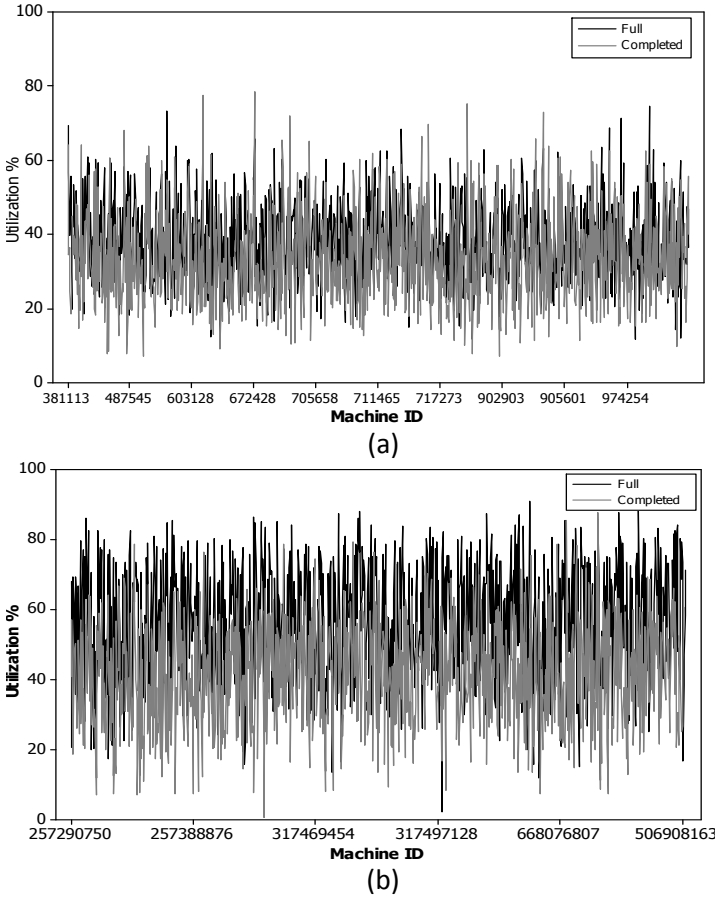
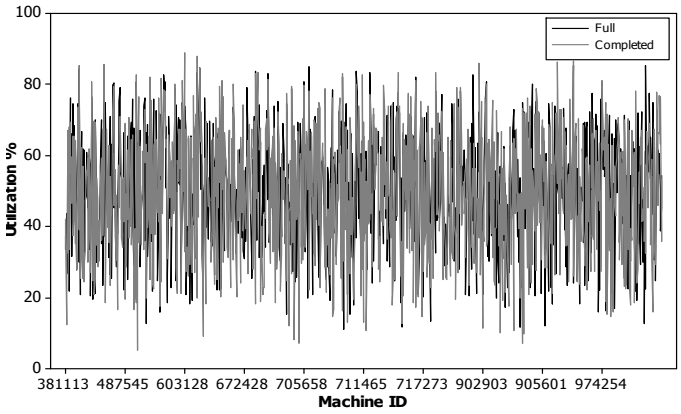


Figure 5.6. Comparison of full and completed tasks in Day 2 for (a) Architecture 4 memory, (b) Architecture 4 CPU.

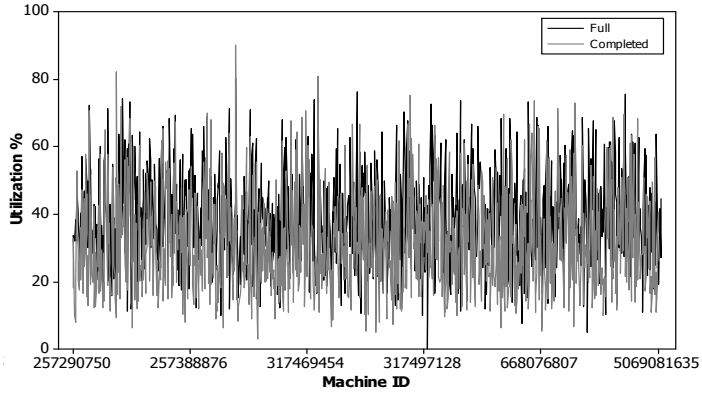
There are two reasons postulated for this resource disparity within the Google Cloud. The first is due to the nature of the Cloud scheduler; it is possible for tasks currently executing within a server to be evicted due to insufficient server capacity or higher priority tasks being scheduled to the server resulting in lower priority tasks being evicted. In each of these cases, the eviction of a task results

Table 5.2 Summary of Day 2 Wasted Resource Utilization

		Full Task %		Completed Task %		Resource Disparity %
	Server Architecture	Average	St. Dev	Average	St. Dev	
CPU	2	39.12	8.44	31.26	7.79	7.86
	4	32.86	12.38	28.33	12.49	4.53
	5	55.66	17.24	41.44	15.06	14.22
	6	41.55	14.16	35.08	13.77	6.46
Memory	2	51.32	13.39	50.03	13.54	1.29
	4	50.83	13.01	48.84	14.7	1.99
	5	39.11	13.71	31.51	15.13	7.61
	6	49.86	15.71	47.21	17.54	2.66



(a)



(b)

Figure 5.7 Comparison of full and completed tasks in Day 2 for
(a) Architecture 2 memory utilization, (b) Architecture 2 CPU utilization

in loss of computation and consequently resource utilization. The second reason postulated is the result of the Cloud workload environment being driven by user behaviour. Specifically, the characteristics of submission patterns of users can have a substantial impact on the type of tasks submitted into the system, which influences the resource utilization and subsequently inefficiency. An example of such behaviour can be observed as described in Chapters 3.3 and 4.3.1-4.3.2, respectively; which shows that specific users can have a strong influence on the workload characteristics within the Cloud datacenter, resulting in increased evictions and computation wasted.

5.4 Impact of Server Characteristics and Server Inefficiencies

The results presented by using the method described in this chapter exemplify and quantify two specific findings which are of importance to the Cloud computing research community. The first is the quantification of server architecture heterogeneity and resource utilization of a large-scale production

Cloud datacenter: It is observable when contrasting the related work in Chapter 2.4.6 that the average resource utilization of the case study trace log exhibits an efficient resource utilization of 40-60% indicating high average utilization for servers. Furthermore, it is observable that different server architectures exhibit heterogeneous resource utilization patterns due to server capacity and task constraints, and that there exists a correlation between resource utilization and the number of task submissions. The results presented in this chapter further our understanding of server characteristics within Cloud datacenter environments.

The second and more important finding is that the results presented in this chapter offer empirical evidence and quantification of resource inefficiency in servers within a Cloud datacenter. Resource utilization waste per server of 4.54% - 14.22% and 1.29 - 7.61% per server for CPU and memory, respectively should not be overlooked as an insubstantial amount when considering that there are potentially thousands of servers exhibiting similar levels of waste. This waste of resources translates into economic loss for providers in the form of energy consumption, as well as reduced availability of servers. Furthermore, these results provides Cloud providers a baseline in order to better understand how resources are being utilized by servers, as well as identify the cause of operational inefficiency hotspots within servers so they can be corrected.

5.5 Summary

This chapter has provided an analysis of server characteristics and resource utilization and inefficiency within a production Cloud datacenter.

The method of calculating resource utilization for individual servers has been described in detail and includes the calculation and quantification of server computation waste. Finally, the assumptions of the analysis when applying the method to the Google Cloud case study has been discussed.

The resource utilization analysis is presented and discussed in detail, including temporal and spatial analysis of CPU and memory utilization of individual servers and server architecture types within the case study trace log. From the analysis, it is observable that utilization patterns of server varies between 40-60% by server architecture type as well as time frame, and that there exists a correlation between the task submissions and server utilization.

The analysis of resource inefficiency within Cloud datacenters is presented and discussed with results demonstrating that 4.54% - 14.22% and 1.29 - 7.61% of CPU and memory, respectively is wasted per individual servers, and that this inefficiency varies by server architecture type. Reasons for this behaviour are postulated, including behaviour of the resource scheduler, and the workload submitted into the system.

From the analysis of server operational inefficiency, a baseline has been provided that demonstrates that it is possible to quantify computation waste within servers. The work has shown that it is necessary to investigate and quantify in more detail not only focusing on the sources of operational inefficiency within Cloud datacenters in terms of resource efficiency, but also failure characteristics and energy waste.

6 Failure and Failure-related Energy Waste Analysis

6.1 Overview

Following the identification of resource inefficiency within servers, this chapter presents a more comprehensive study and analysis of the failure characteristics and failure-related energy waste of tasks and servers within a large-scale Cloud environment. Specifically, this chapter contains two core contributions:

The first holistic empirical failure analysis of tasks and servers in a Cloud datacenter. This includes the temporal and spatial analysis of Termination Events, as well as failures and repair characteristics of tasks and servers within Cloud datacenters. Moreover, the statistical properties and probability distribution of failures are presented and discussed in detail and can be used by other researchers to build more realistic assumptions for Cloud system operation.

The first empirical quantification and analysis of energy waste produced by failures and Termination Events within Cloud datacenters. This work for the first time identifies and quantifies the cause and amount of energy waste produced by failures in tasks and servers, as well as analyzes their respective temporal and spatial properties of operational waste.

This chapter also identifies the key parameters of failure and repair times for tasks and servers which can be used by other researchers to simulate realistic workload and system behaviour. This chapter closes with discussion of the practical applications of the findings in this work.

6.2 Failure Analysis Method

The method used to extrapolate failures from a trace log can be divided into three main steps: event sampling, failure event identification, and failure-energy impact analysis. This was done in order to comprehensively conduct the failure-analysis and quantify failure-related energy waste.

6.2.1 Event Sampling

As discussed in Chapter 3.1.1, events recorded in the Google trace log used as our case study describe state transition for components within the system. Termination Events are used to identify unsuccessful task execution, and are composed of three unique events: EVICT (disk failure or scheduler eviction), KILL (user or task specified termination) and FAIL (task failure). In the context of servers, the REMOVE event indicates when a server has been removed from the cluster due to a failure or planned maintenance.

Events logs are divided into two different failure catalogs; tasks and servers. For the analysis it is necessary to calculate the elapsed time between termination and time to recovery for both tasks and servers. *Elapsed termination time* is calculated as the time between scheduling and Termination Event occurrence, and the elapsed time between ADD and REMOVE events for tasks and servers, respectively. *Repair time* is calculated by the elapsed time between a Termination Event and rescheduling, and the elapsed time between REMOVE and ADD events for tasks and servers, respectively. These elapsed times are calculated based on the timestamps recorded within the event logs, as discussed in Chapter 3.1.2 and 3.1.3.

This study does not consider task terminations and server failures that occur outside the trace log observational period, as it is not possible to characterize failure and repair times accurately without knowledge of both scheduling and termination time for tasks and servers. Inclusion of such data would be likely to skew results and modelling of failures. In the context of the Google Cloud trace log, this condition predominately excludes task monitoring servers running within the Cloud (task priorities 10 and 11) as discussed in Chapter 4.2.4. With this assumption, the task catalog consists of over 13,562,457 events, representing just over 98% of FAIL events recorded within the trace log.

6.2.2 Failure Event Identification

Not all task Termination Events that occur are due to task or server failures. For example, EVICT events can be a result of the scheduling policy of the system, representing typical system behaviour, or the result of a server hard disk failure. Due to this ambiguity, it is not possible to identify task failures solely from the previous derived failure catalog. Therefore, it is necessary to define a set of

assumptions - supported by observations of system behaviour within the trace log, as well as relevant literature - to identify task failure events separate from Termination Events.

Using this approach it was possible to identify two types of failures with occur within the system: Server failures and task failures. These failures and their respective assumptions derived from the literature and dataset observation are shown in Table 6.1.

Server failures are characterized as software or hardware crash failure; based on observations from the data. When a failure occurs, all tasks currently executing on the server are subsequently terminated with either the KILL or EVICT event. As stated in [157], the server REMOVE event is the result of a server failure, or planned maintenance. As a result, it is not possible to distinguish between the two. However, from analysis of the Google trace log data the vast majority of server REMOVE events occur while there are tasks executing on the server. As a result, we classify all REMOVE events as server failures (agnostic of maintenance), as the event causes tasks to deviate from correct service.

Task failures are defined as software crash failures, and are identified and filtered from the task event log by tasks which experience a FAIL event. In [157], FAIL events have been explicitly defined as the result of a software crash of the task.

Furthermore, it is also possible to identify terminated tasks which are the result of server failures. These failures were identified by tasks that were terminated by KILL, EVICT or FAIL events whose timestamp occurred within the time period between a server REMOVE and ADD event.

It has been well understood that the root causes of task and server failures might be physical, design (typically software), human-machine interaction faults, or even malicious attacks, or a combination of each [188]. In reality, transient hardware faults, hardware design faults and software bugs often cause similar system behaviour [189]. It was decided not to distinguish the root cause of a failure for servers due to ambiguities within the case study trace log as discussed in this section. However, we are able to filter tasks failures that are resultant of hardware or software crashes within servers and software crashes within a task.

Table 6.1 Failure Assumptions

Failure Observation	**Server Failure**	**Task Failure**
Actors	Server and Task	Task
Description	Server experiences a software or hardware crash failure.	Task experiences a software crash failure.
Precondition	1.Server is operational. 2.Tasks are eligible for submission or currently being executed on the server.	1. Task is currently executing on a server
Post Condition	1.Server event REMOVE occurs. 2.Tasks currently scheduled on the server result in either event EVICT or KILL.	1.Task event FAIL occurs. 2.Server continues operating.
Dataset Observation	1.After a server experiences a REMOVE event, all tasks that were scheduled onto the server subsequently experience KILL or EVICT events microseconds apart from each other. 2.No further tasks are scheduled onto the server until it recovers and rejoins the system where possible.	1.A portion of tasks within the trace log experience a FAIL event. 2.After a finite amount of time, the task recovers and is rescheduled back onto a server.

It must be emphasized that each failure event for servers and tasks may not necessarily correspond to a unique failure, and that failure events that are temporally close together may be caused by the same failure. This vagueness is a result of the lack of precise data concerning failures, even after filtering failure events apart from Termination Events. However as stated in previous works [113][114][133], it is extremely difficult to identify the root cause as well as the duration of a failure.

6.2.3 Failure Analysis

The failure analysis presented follows a similar method described in Chapter 4.2.5, and includes the statistical properties for failure and repair times, including the Mean (μ), Standard Deviation (σ) and Squared Coefficient of Variance (Cv). Furthermore, we match the closest theoretical distributions applying AD GoF tests to obtain the statistical parameters of Mean Time Between Failure (MTBF) and Mean Time to Repair (MTTR). To deal with the large amount of records

present after extrapolating the data using the analysis infrastructure, Minitab is used to efficiently perform a large portion of the analysis.

In addition, we have evaluated the data against a number of distributions including Weibull, Gamma, Loglogistic, Exponential and Lognormal. Lastly, we have presented a fit comparison in the form of Empirical Cumulative Distribution Functions (CDFs) between the overall system where applicable and priority 9 tasks, as the latter represent production tasks within the Cloud environment. We believe that these tasks are of high relevance and importance to the Cloud research community in providing good representations of the distribution of data for further research.

6.2.4 Energy Analysis

Energy waste is calculated by the energy consumed by a task prior to Termination Event occurrence, which is based on the power profile of the server where the task executes, and the average CPU load imposed by the task on the server. As shown in Table 6.2, and discussed in detail in Chapter 3.3, servers within the trace log are heterogeneous in nature and have been grouped into 3 platforms, each containing different a combination of chipset versions, CPU capacity and micro-architecture. Each platform contains one or more configurations that vary in memory capacity.

Table 6.2 Server Mapping from Trace log to Real Systems

Trace log				**SpecPower2008**			
Server Platform	**Server Type**	**CPU Capacity**	**Memory Capacity**	**Server Platform**	**Server Type**	**CPU Capacity (ssjops)**	**Memory Capacity (GB)**
A	1	0.25	0.25	ProLiant DL365 G5	1	337,543	8
B	2	1.00	1.00	PRIMERGY RX200 S7	2	1,338,554	32
	3	1.00	0.50		3	1,338,554	16
C	4	0.50	0.25	1022G-NTF	4	793,535	8
	5	0.50	0.75		5	793,535	24
	6	0.50	0.50		6	793,535	16
	7	0.50	0.97		7	793,535	32
	8	0.50	0.12		8	793,535	4
	9	0.50	0.03		9	793,535	1
	10	0.50	0.06		10	793,535	2

As discussed in Chapter 4.2.4, the actual characteristics and power profiles of the servers under consideration are obfuscated, therefore the best assumption that can be made is to match the profiles and characteristics of real datacenter servers based on the CPU and memory capacity within the trace log. These profiles are considered from the results of the SpecPower2008 Benchmark [182]. The profiles presented by SpecPower2008 are preferred over other available server benchmarks as the results are obtained following a strict methodology of experimentation and monitoring, presented in [182]. Specifically, the selection of specific profiles is based on the proportional similarity between obfuscated server capacities of CPU and memory, and the specific server configuration provided in SpecPower2008 results. This allows us to perform energy calculations proportionally similar to those that could be measured directly from the actual Cloud datacenters shown in Table 6.2. For example, the ProLiant platform has approximately 25% of the CPU and memory capacity of the PRIMERGY platform which is proportionally equivalent to Platform A, which has 25% of CPU and memory capacity compared to Platform B within the trace log. This proportionality of matching server profiles is the same when comparing the 1002G-NTF and PRIMERGY platforms to platforms B and C.

According to [183][185], CPU consumes the largest amount of total power demand in physical servers in comparison to other resources. Therefore, it is assumed that servers which share the same platform exhibit the same power profiles, due to sharing identical micro-architecture and CPU capacity. Each platform contains a unique power profile which is parameterized using 10 measurements between 0% and 100% for system utilization, with increments of 10% and their respective power consumption. Moreover, as stated in [182], system utilization is measured by Server Side Java Operations (ssjops), and power is measured in watts. The power models of the three platforms described are presented in Figure 6.1.

The described model that connect servers identified in the trace log to specific server platforms in accordance with the SpecPower2008 benchmark allows the approximation of energy E consumed by specific tasks considering their power usage during time execution period t (measured as elapsed time). Power usage P and CPU consumption u of a task is estimated by applying linear interpolation

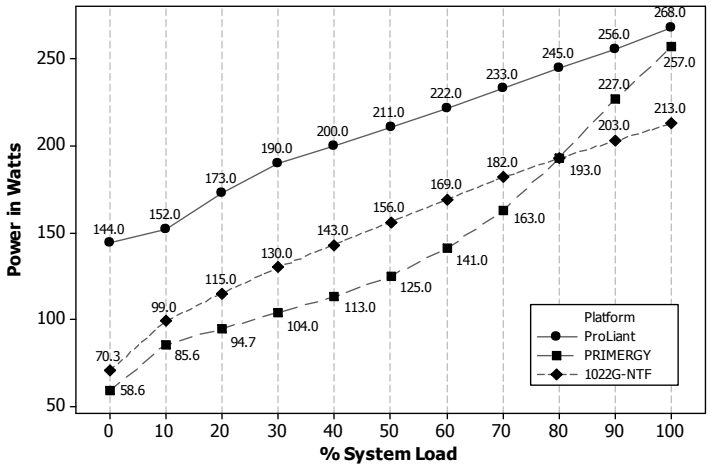


Figure 6.1 Power models of the selected platforms

between measurements α and β where α is the lowest measurement of utilization u_α and power consumption $P(u_\alpha)$, while u_β and $P(u_\beta)$ represents the highest measurement of utilization power consumption, respectively. An example of this would be to determine the power consumption of a system load at 32% between 30-35% for a server platform. The formalization of these concepts is presented in Equations 6.1 to 6.3.

$$P(u) = \Delta Pow \cdot u + (P(u_\alpha) - \Delta Pow \cdot u_\alpha) \tag{9}$$

$$\Delta Pow = \frac{P(u_\beta) - P(u_\alpha)}{u_\beta - u_\alpha} \tag{10}$$

$$E(t) = P(u) \cdot t \tag{11}$$

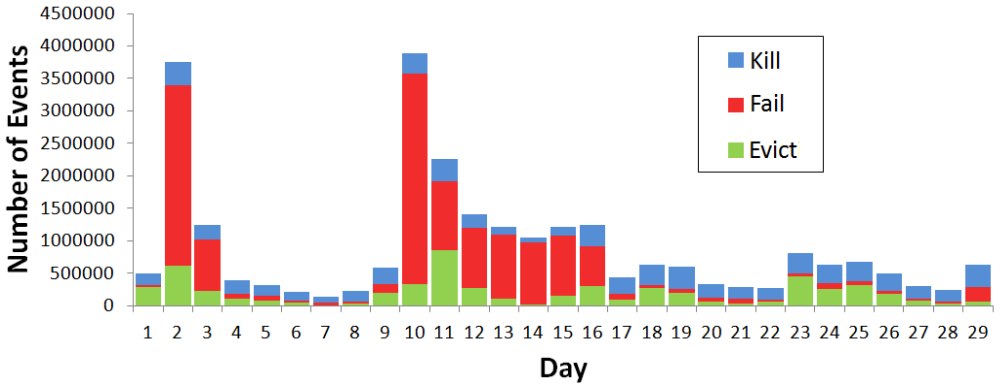
Using our scheme, it is possible to calculate the total energy waste calculated for tasks and servers. However such energy waste can potentially be reduced when considering checkpointing of tasks. However, when consulting the supporting literature of the trace log, as well as analyzing and characterizing task and resource utilization, we found no evidence of tasks exhibiting the behaviour of checkpointing. In addition, Termination Events result in the work performed prior to termination to be lost; this behaviour in a subset of tasks is supported in [3][160] which states that "a task failure is an interruption on a running task, requiring the system to re-execute the interrupted task", indicating that a failure results in a task being restarted from the beginning of execution. It is possible to assign theoretical checkpointing frequency and overhead time arbitrarily to tasks

within the system, however such an approach introduces subjectivity and distorts the behaviour of tasks actually observed within the system. Furthermore, checkpoint mechanisms themselves consume considerable amounts of energy such as memory and HDD logging [186][187].

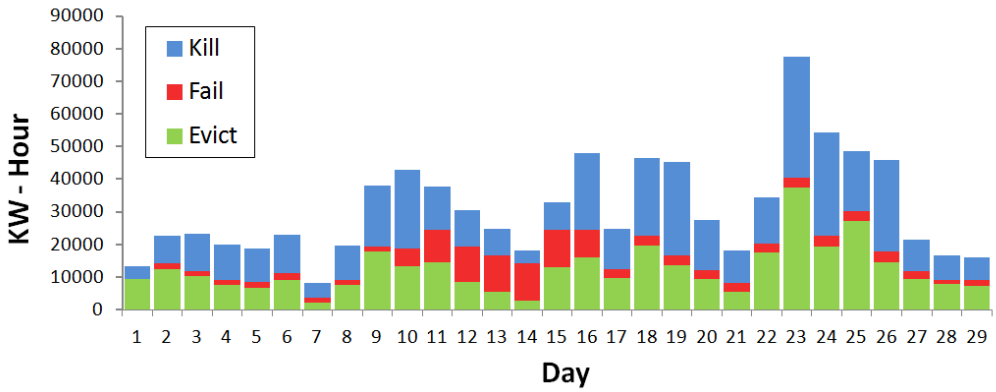
6.3 Failure Analysis

6.3.1 Termination Events Analysis

Applying the method described in Chapter 6.2.2 to the Google Cloud trace log, a total of 25,927,826 Termination Events are identified, with 52%, 22% and 26% corresponding to FAIL, EVICT and KILL events, respectively as depicted in Figure 6.3(a). It can be observed that the volume and proportion of Termination Events occurring varies temporally as shown in Figure 6.2(a), with Days 2 and 10 experiencing a substantially large volume of Termination Events. The reason for this behaviour is postulated in [157] due to a phenomena known as "crash-loops", caused by tasks deterministically failing shortly after beginning execution, yet are subsequently configured to restart shortly after that failure. In addition, a

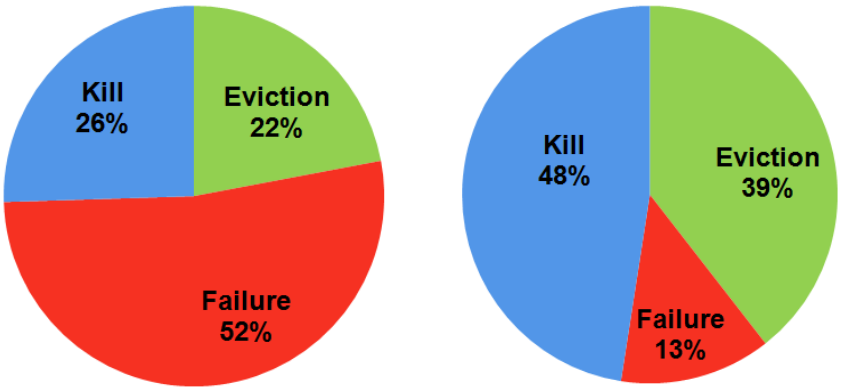


(a)



(b)

Figure 6.2 Daily Termination Event (a) Occurrence, (b) Energy Waste



(a) (b)

Figure 6.3 Proportion of Termination Event (a) Occurrence, (b) Energy waste

substantial portion of task failures which occur within these two time periods are due to tasks failing and being rescheduled multiple times, resulting in significant increase in the amount of work the scheduler performs.

When contrasting the proportion of energy waste generated by Termination Events, FAIL events only contribute 13% in comparison to 48% and 39% for KILL and EVICT, respectively as illustrated in Figure 6.3(b). This disparity between Termination Events and energy waste indicate two system properties. First, task FAIL events mostly occur at a very early stage during execution, reducing overall elapsed time and subsequently energy waste. The second is that KILL and EVICT events affect more long running tasks, as indicated by a larger proportion of energy waste in proportion to the number of Termination Events. As demonstrated in Figure 6.2(b), it is observable that daily energy waste generated by task failures is more variable compared to KILL and EVICT events, indicating that the majority of these events are due to typical scheduler operational behaviour, while FAIL events are the result of abnormal system behaviour.

Table 6.3 General Failure Statistics.

Total Server Failures	8954
Number of Servers Failed	5056
Task Failures	13,572,457
Unique tasks failed	829,738

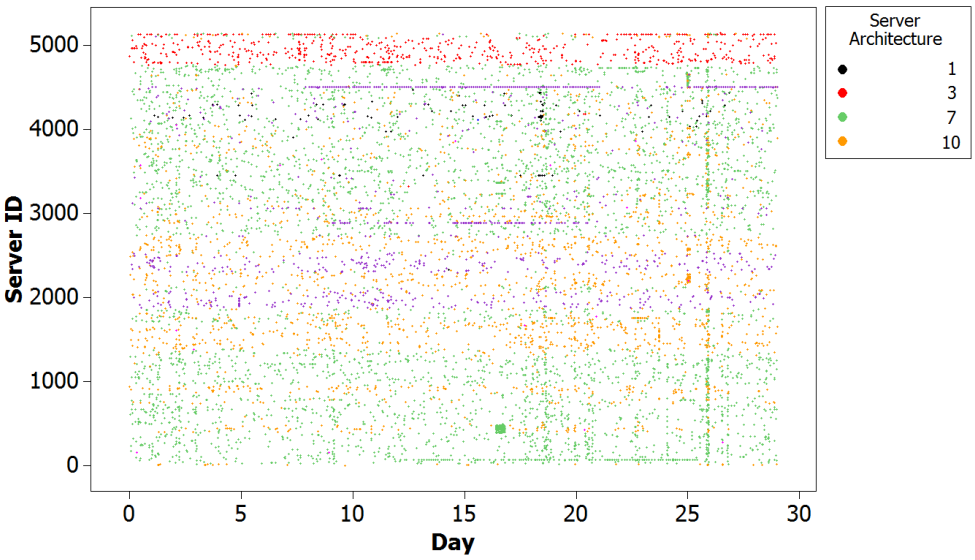


Figure 6.4 Server failures within the observational period

Table 6.3 depicts general statistics of failure events within the trace log after applying the filtering method described in Chapter 6.2.2. It is observable that 3.26% of tasks experience one or more failures; this indicates that a relatively small amount of unique tasks (3.26% of total tasks) are responsible for a significant amount (52%) of failures which occur. This percentage of failure is comparable to other distributed systems studied in [115] and [107], where 5-8% and 2.4% of jobs failed after an undefined period of execution, respectively.

6.3.2 Servers

There are a total of 8,954 server failures which occurred within 5,056 servers as depicted in Figure 6.4, and an average of 308 servers failing daily with a standard deviation of 101 as depicted in Figure 6.5. It is observable from Figure 6.6 that a small proportion of servers experience a high occurrence of failures, and the rest experiencing substantially less events in similar proportions. Furthermore, from Figure 6.5 we can observe that the proportion of failures be server architecture type stays relatively stable agnostic by the number of server failure events daily.

The statistical properties and probability distributions of the MTBF for server architecture populations greater than 1% (representing 99.56% of the total server population) are shown in Table 6.4. It is observable that server architecture types exhibit similar MTBF between 12.2 - 13.04 days, exhibiting low variance reflected by C2 values between 0.24 - 0.44. Furthermore, it is noticeable that the best fit distribution for all server architectures is Weibull,

Table 6.4 Statistical Properties and Model Parameters of Server MTBF

	Failure				
Server Architecture	**Best Fit Distribution**	**Parameters**	**μ (Days)**	**σ (Days)**	***Cv***
1	Weibull	k = 2.191 λ = 12.80	12.239	5.952	0.237
3	Weibull	k = 1.463 λ = 13.79	12.55	8.28	0.435
5	Weibull	k = 1.516 λ = 14.01	12.489	7.79	0.389
7	Weibull	k = 1.540 λ = 13.77	12.71	8.057	0.402
10	Weibull	k = 1.641 λ = 14.50	13.046	7.784	0.356

depicted in Figure 6.7, conforming to previous analysis findings of server failure characteristics [1]. 59,583 tasks failed due to server failures, corresponding to 21% and 79% for KILL and EVICT events, respectively. These failures represent 0.44% of the total task failures within the trace log.

Table 6.5 presents the statistical properties and probability distribution characteristics for server repair time. It is observable that the median repair time is substantially lower than that of the MTTR, ranging from 0.15-0.28 hours and 1.48-9.17 hours, respectively. Server repair time was best fit by Lognormal and Loglogistic distributions for server architectures across the entire system as

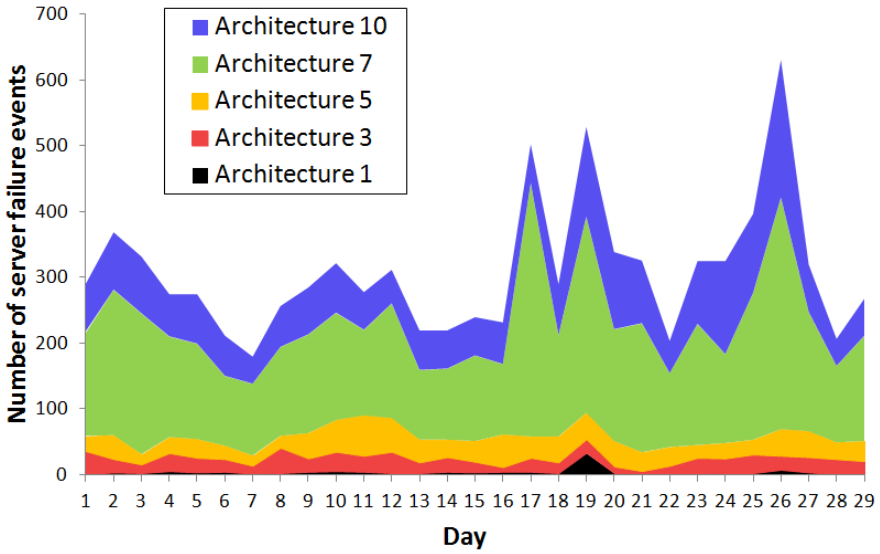


Figure 6.5 Daily server failures within the observational period

Table 6.5 Statistical Properties and Model Parameters of Server MTTR.

	Repair					
Server Architecture	**Best Fit Distribution**	**Parameters**	**Median (Hours)**	***μ* (Hours)**	***σ* (Hours)**	**Cv**
1	Lognormal	μ = -1.620 σ = 2.964	0.28	9.17	28.8	9.86
3	Loglogistic	α = -1.661 β = 0.7326	0.24	1.48	8.99	36.89
5	Loglogistic	α = -1.249 β = 1.075	0.27	4.81	13.32	7.67
7	Lognormal	μ = -1.529 σ = 2.156	0.19	4.17	17.22	17.05
10	Lognormal	μ = -1.152 σ = 2.125	0.15	8.17	12.33	2.28

depicted in Figure 6.8. From the statistical properties and empirical CDF, it is demonstrable that a large quantity of repair time spans a relatively short amount of time of under 30 minutes. There are two reasons for this characteristic; Firstly, as stated in [157], a portion of REMOVE events are due to maintenance. Secondly, we postulate that a large portion of server failures are corrected by simply restarting the server. Moreover, it is observable that a small proportion of servers require an extended period of time to be repaired, spanning several days. This is an indication that the failure within the server is more complicated and is not corrected by simply restarting the server, requiring several days for repair. Such behaviour is indicated by the high variability of server repair times, demonstrated by the long-tail of the CDF as well as the *Cv* value between 2.28 - 36.89.

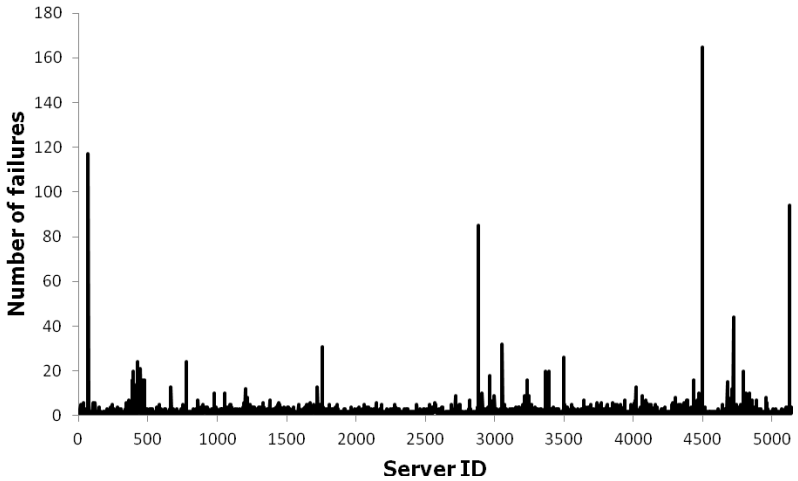


Figure 6.6 Number of failures per server.

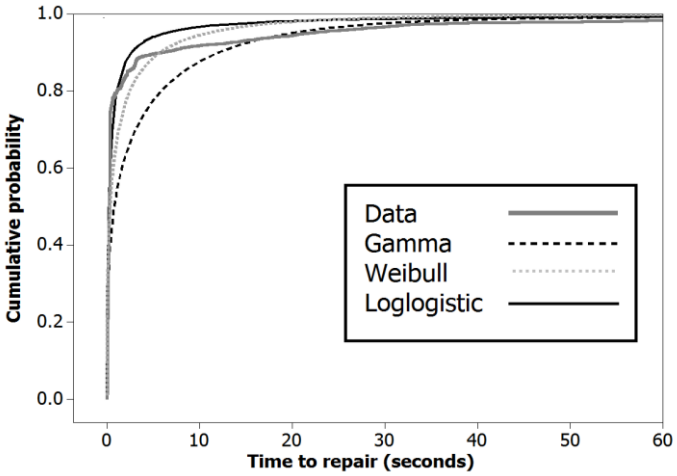


Figure 6.8 Empirical CDF of server repair times.

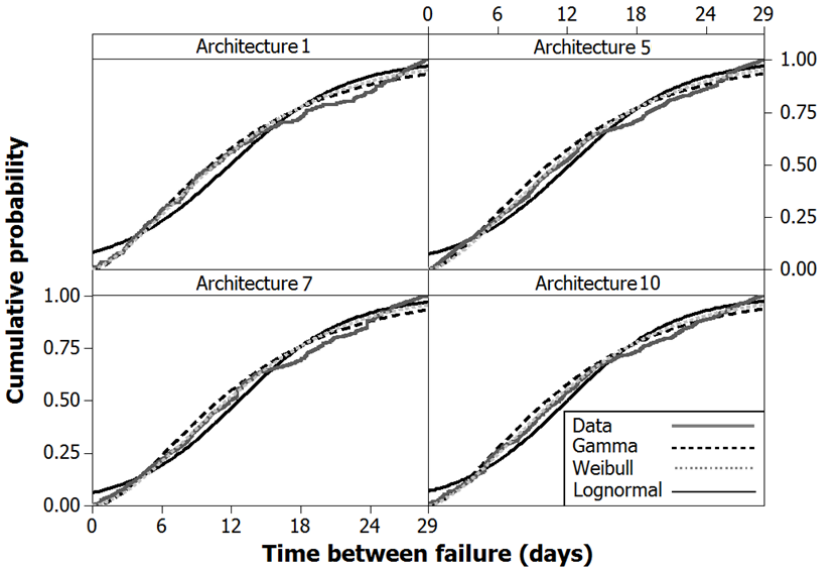


Figure 6.7 Empirical CDF of time between failures for server architectures.

6.3.3 Tasks

As per our discussion in Chapter 2.4.7, failure analysis of tasks is crucial in characterizing realistic task behaviour. From our analysis, we have discovered the following properties: Table 6.6 presents the statistical properties of task failures due to software crashes, as well as the best fit probability distribution classified by task priority. An attempt was made to fit a theoretical probability distribution to task MTBF agnostic of priority, however it was discovered that due to the different failure patterns of tasks with different priorities (represented by a *Cv* values of 46, indicating significant variability), it was not possible to feasibly fit the data to a distribution visually or by using a GoF test. The reason for this high

Table 6.6 Statistical Properties and Model Parameters of Task MTBF

	Failure				
Priority	**Best Fit Distribution**	**Parameters**	**μ (Hours)**	**σ (Hours)**	***Cv***
0	Weibull	k = 0.5107 λ = 0.3342	1.063	4.925	4.63
1	Lognormal	μ = -1.638 σ = 1.665	1.694	8.083	4.77
2	Lognormal	μ = -0.3489 σ = 2.152	3.825	11.836	3.09
4	Lognormal	μ = -1.921 σ = 1.763	1.019	4.967	4.87
6	Loglogistic	α = -3.073 β = 0.3129	0.062	0.093	1.50
8	Loglogistic	α = -0.2421 β = 2.154	48.53	64.190	1.32
9	Gamma	k = 0.2215 λ = 265.1	58.72	95.030	1.62

variability is the result of the substantially different failure characteristics of tasks with different priorities, reflected by a substantially different mean and standard deviation for MTBF. When studying the MTBF of each task priority, it can be observed that the *Cv* value is considerably lower than the task MTBF obtained for the complete set of tasks, and consequently makes it impossible to successfully fit the data to a single probability distribution. It was discovered that different task priorities best fit a number of distribution types, ranging from Lognormal, Loglogistic and Weibull distribution, that are all right-skewed as shown in Figure 6.9. Such behaviour further reinforces the observed characteristics of crash-loops discussed in Chapter 6.3.1.

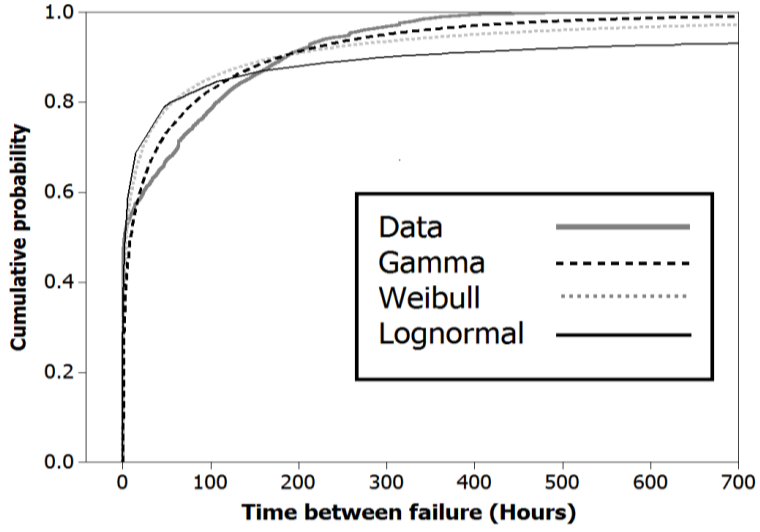


Figure 6.9 Empirical CDF of time between failure for production tasks.

Furthermore, priority 9 tasks, which represent production tasks within the Cloud datacenter, exhibit a high MTBF of 59.72 hours and 95.03 hours for mean and standard deviation, respectively and consequently a *Cv* value of 1.61; considerably lower in comparison to other task priorities. We observe that production tasks best fit a Gamma distribution (Shape parameter 0.22) as depicted in Figure 6.10. From the empirical CDF it is observable that there exist two types of failure characteristics for tasks; tasks that fail near the beginning of execution, and tasks that fail far into their lifespan. For example, Figure 6.10 demonstrates that out of the 3.26% of tasks which fail within the Cloud datacenter, approximately 70% of these tasks fail within the first hour of execution, while the remaining 30% of task MTBF ranges between over an hour to up to 300 hours, reflected by a larger mean and right-skew within the empirical CDF. One of the reasons for this phenomena is user behaviour; there exists a single user within the system which submits 65% of total production tasks that fail, all of which occur within Day 3 as depicted in Figure 6.2(b), and fail just under a minute of execution. Such behaviour is worth highlighting, as Cloud environments are driven by user behaviour with varying QoS demands as discussed in Chapter 2.2.2 which empirically analyzes user characteristics. As a result, there is potential concern in large-scale systems that such user behaviour, with correlation between workload, type, system size and complexity, can cause failure characteristics within the system and affect other users.

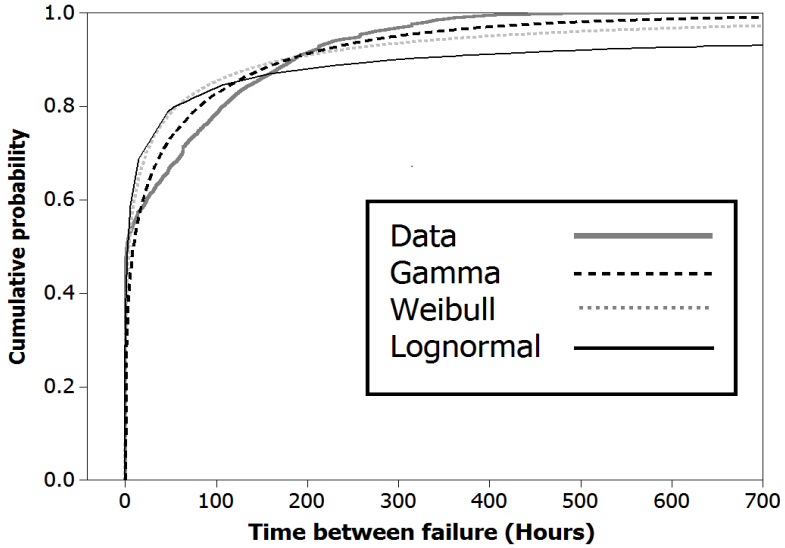


Figure 6.10 Empirical CDF of time between failure for production tasks.

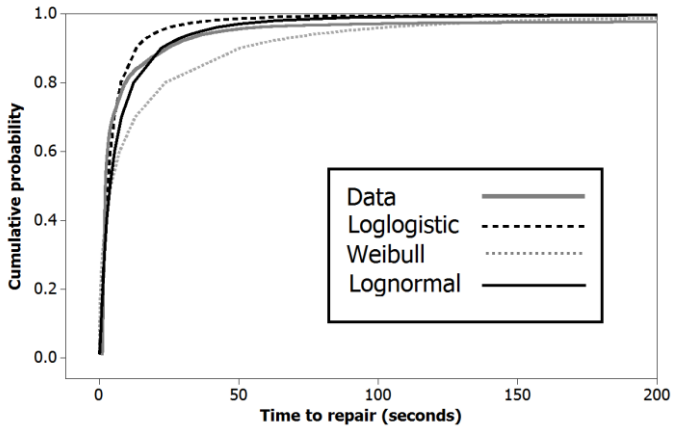
Table 6.7 Statistical Properties and Model Parameters of Task MTTR.

	Repair					
Priority	**Best Fit Distribution**	**Parameters**	**Median (Hours)**	**μ (Seconds)**	**σ (Seconds)**	***Cv***
0	3-Param Loglogistic	α = 0.9497 β = 1.115 T = 0.9822	2.9	122.9	1472.2	143.5
1	3-Param Lognormal	μ = 1.206 σ = 1.758 T = 1.049	4	161	7157	1976
2	Lognormal	μ = 1.216 σ = 1.227	2	28.7	182.1	40.26
4	3-Param Loglogistic	α = 0.0067 β = 0.6553 T = 0.9737	1.91	16.32	173.36	112.8
6	3-Param Loglogistic	α =-0.529 β = 0.5227 T = 1.089	1.67	2.67	4.06	2.31
8	3-Param Lognormal	μ = 0.2871 σ = 2.083 T = 1.317	2.44	8.37	14.52	3
9	3-Param Lognormal	μ = 0.4904 σ = 1.274 T = 1.031	2.43	4.75	5.77	1.48

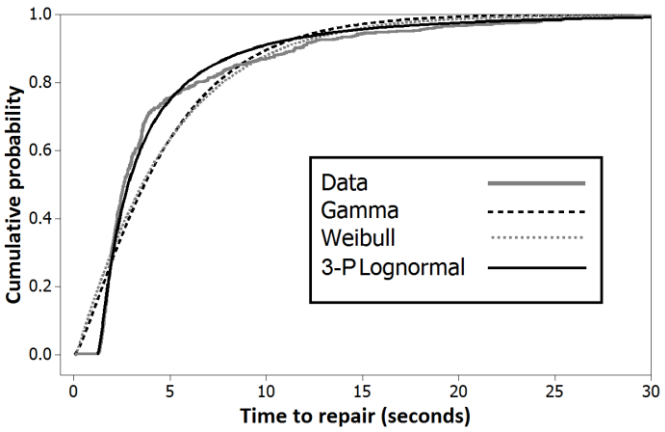
Table 6.7 presents the statistical properties of repair times for tasks; there were similar challenges in fitting the empirical data to a theoretical distribution both visually and using GoF tests holistically over the system. This is due to similar challenges as with modelling task MTBF; significant variability of recovery times within the system. This is particularly noticeable within lower priority tasks, which exhibit *Cv* values between 112 - 1976. Repair time for higher priority tasks (6-9) is more stable, indicated by the lower *Cv* values, as well as closer median and mean values. From Table 6.7 it is observable that similar to task failure times, repair times are also heterogeneous, ranging from under 3 seconds to 123 seconds indicating that restarting tasks appears to correct a large proportion of faults. Furthermore, there exists a correlation between the task priority level and repair time, indicted by lower priority tasks exhibiting a longer repair time in comparison to higher priority tasks. The reason for this is due to the nature of the scheduler and the task's function: Lower priority tasks are more likely to be delayed for high priority tasks to be allocated to a server within the system. It is

also assumed that there exists a correlation between the task priority level and task criticality from discussion in [156][161], which states that lower priority tasks are less developmentally mature, consequently resulting in more frequent failure and longer repair times. Lastly, the existence of crash-loops also plays a considerable factor in data skewness for repair times within the trace log, represented by a low median and a higher mean and standard deviation.

Figure 6.11(a) and Figure 6.11(b) present the empirical CDF for task repair time for all tasks and production tasks within the trace log, respectively. It is observable that the Lognormal distribution is the best fit distribution for repair times; such findings align to failure-analysis of past distributed systems [1]. In the case of Figure 6.11(a) however, this distribution can be misleading due to the AD value calculated for the GoF test being unacceptable high, signifying that the empirical data significantly deviates from the theoretical distribution. In contrast, the AD value calculated for Figure 6.11(b) is hundreds of time lower than that of the entire trace, and fits to a 3-Parameter Lognormal distribution that both



(a)



(b)

Figure 6.11 Empirical CDF of repair times for a) All tasks, b) Production tasks

visually fits and satisfies GoF tests. In comparison, the distributions Gamma and Weibull are poorer fits for system wide and task priority agonist repair times.

As observed from the analysis, it is demonstrable that tasks and servers exhibit diverse failure and repair characteristics which are capable of being modelled. Attention is now drawn to their impact on the energy waste of the system due to these failure characteristics.

6.4 Failure-related Energy

6.4.1 Task Failure Energy Waste

Figure 6.12 presents a comparison of the number of failure events per task priority and their respective energy waste. It is noticeable that although priority 0 task failures occur in 80% of cases and produce just under 50% of the total energy waste within the system, there is a weak correlation between the number of failure occurrences and waste produced represented as 0.412 on the Pearson scale. This is the result of the significant variability in the MTBF of tasks: as discussed previously, the vast majority of tasks fail near the start of their execution while a few larger tasks fail much later into their lifecycle; generating small and large amounts of energy waste, respectively.

Furthermore, it is observable that priority 8 and 9 tasks proportionally waste a large amount of energy in comparison to the number of task failures. This is a result of the failure characteristics of these tasks, which exhibit a large MTBF as

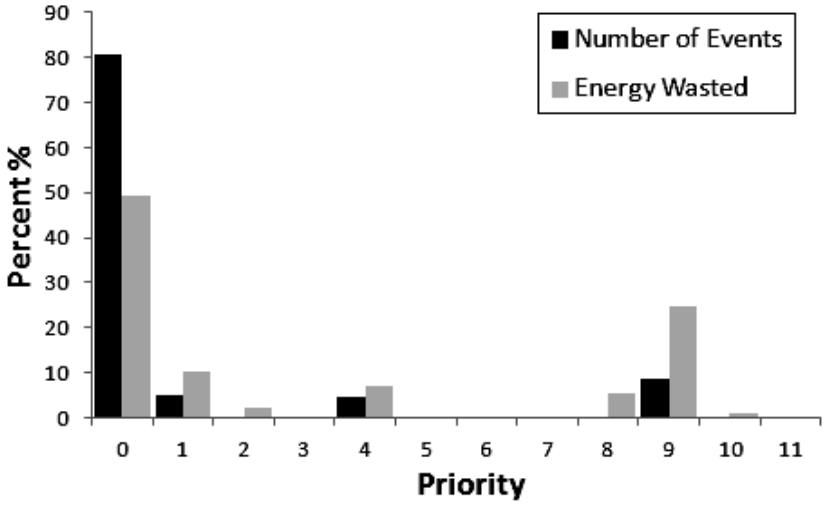


Figure 6.12 Task energy waste per priority.

shown in Figure 6.10, resulting in a large amount of energy waste due to computation lost upon failure.

In terms of task-user ownership, it was discovered that 10 users within the trace log constitute to 91% of the total task failures within the system and are the prime contributors to the crash-loops generated on Days 2 and 10 as depicted in Figure 6.13. It is interesting to note that while User 1 is responsible for the largest number of task failures at 42%, this only translates into approximately 2% total energy waste. This is in contrast to User 2, where 37% of task failures occur resulting in 47% of the energy waste. The reason for this large disparity between the characteristics of these two users is due to the nature of the tasks each user submits; User 2 submits tasks of higher priority which execute for longer periods of time across the system, while User 1 predominately submits priority 0 tasks within the crash-loop time periods. Furthermore we observe that there is a strong correlation between the number of failure events and energy wasted, measured at 0.645 on the Pearson scale.

6.4.2 Server Failure Energy Waste

As discussed in Chapter 6.3.1, server failures resulted in 59,583 task failures, with 21% and 79% of the events corresponding to KILL and EVICT, respectively. This has a considerable impact on the energy waste proportionally when filtered from the total proportion of system energy waste presented in Figure 6.3 as displayed in Table 6.8.

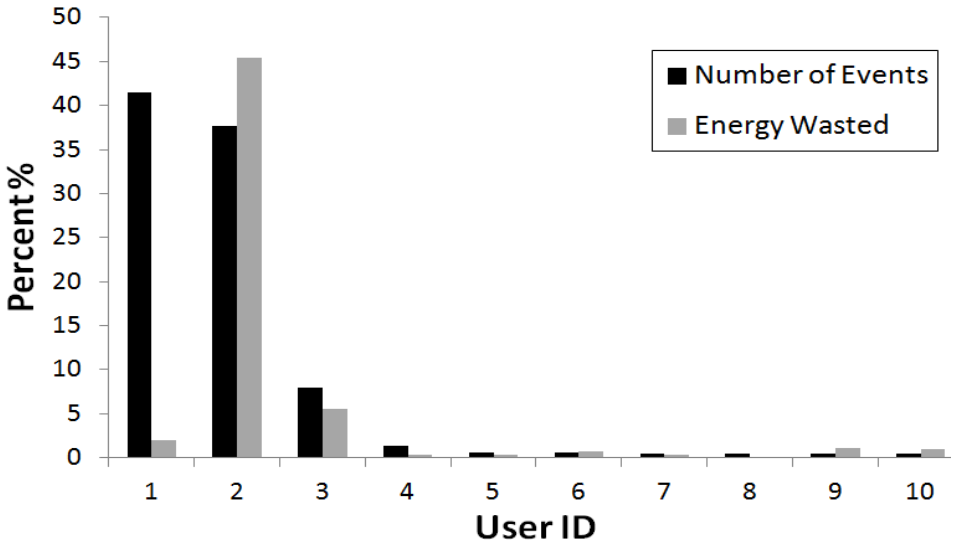
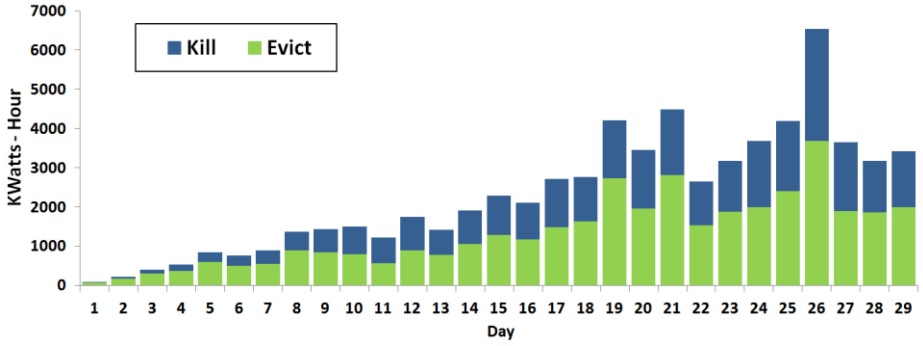


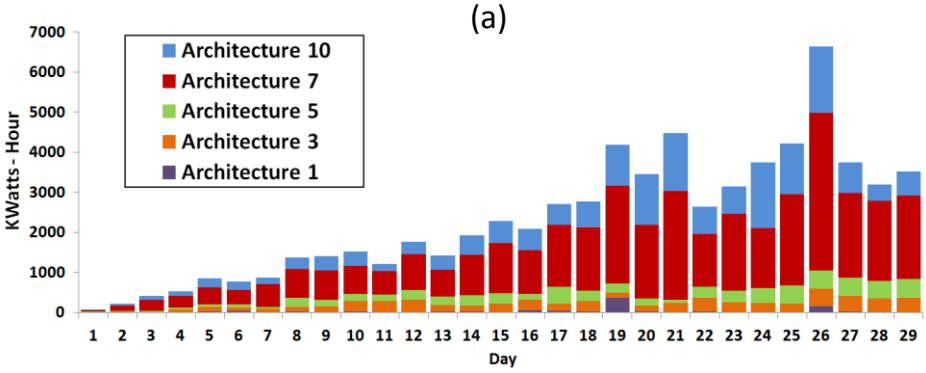
Figure 6.13 Task energy waste per user.

As observed in Table 6.8, although the number of task failures due to server failure represent less than 1% of the total number of EVICT and KILL events within the system, they proportionally produce significant amount of energy wasted in matters of 10.90% and 6.58% for EVICT and KILL, respectively. Additionally, Figure 6.14(a) presents a temporal analysis of daily energy waste produced by server failures within the Cloud datacenter, where it can be observed that although the energy waste produced by server failures varies considerably, the proportion of EVICT and KILL events remains stable at approximately 60% and 40%. Furthermore, it is noticeable that there is an incremental tendency of energy-waste across the days, suggesting that server failures affect a large number of long duration tasks. Therefore, the longer a task has been running, the greater the impact that an eviction or killing has in terms of energy waste.

Furthermore, there is a weak positive correlation between the number of server failures and total energy waste (measure as 0.579 on the Pearson scale). There also appears to be a visual correlation between these two variables when examining the proportion of server architecture types as shown in Figures 6.5 and 6.14(b). Statistically, server types 1, 7 and 10 exhibit a stronger correlation between 0.702 - 0.922 while server types 3 and 5 range between 0.268 - 0.431.



(a)



(b)

Figure 6.14 Energy waste due to server failure a) Events, b) Server architecture.

Table 6.8 Server Failure Energy Waste

	Number of Events	Event Proportion %	Energy Wasted KWatts - Hours	Energy Proportion %
Total EVICT	5,711,417	100.00	355,047	100.00
EVICT by Server Failures	47,234	0.82	38,700	10.90
EVICT by scheduling	5,664,183	99.18	415,991	89.10
Total KILL	6,608,916	100.00	427,476	100.00
KILL by Server Failures	12,349	0.19	28,190	6.58
KILL by user	6,596,567	99.81	399,072	93.42

Figure 6.15 depicts the number of task termination events which occur upon server failure compared with the total energy wasted within the trace log. Here it is observable that up to 27% of failed tasks correspond to priority 9, characterized as long running production tasks, which are responsible for a significant amount of energy waste produced by servers (up to 65%).

From these observations, a strong correlation between server failures and the termination of priority 9 tasks, causing increased energy waste across the analyzed days in the trace log, can be depicted. This is also indicated by a Pearson scale value of 0.92.

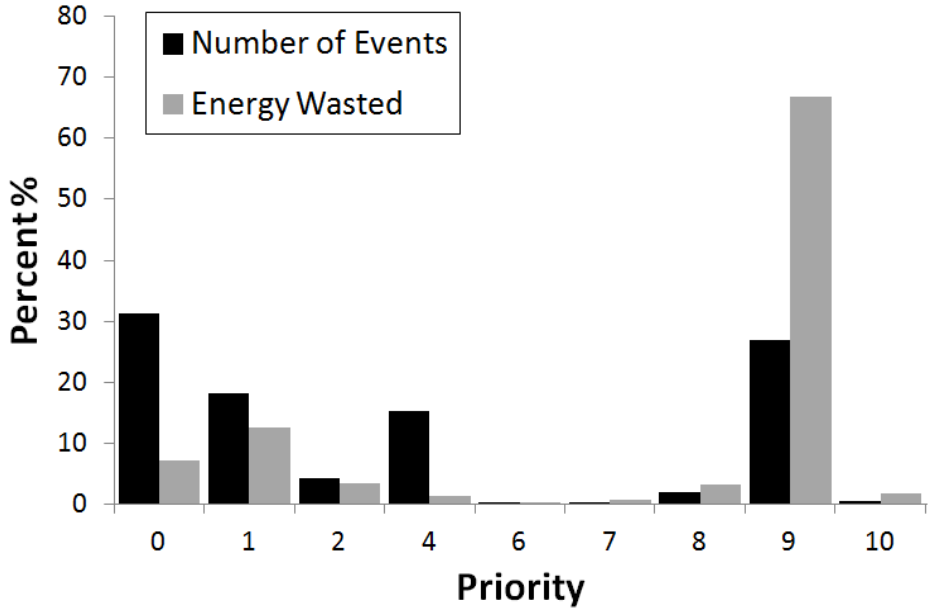


Figure 6.15 Energy waste due to server failure per priority.

Within their lifecycle, individual servers experience an average 1.63 failures with a standard deviation of 2.68, which is an indication that the majority of servers share similar probability of failure occurring over their lifetimes. Moreover, there are a minority of servers which experience 10 or more failures, with a maximum value of 165. In broader terms, the top 10 servers (representing 0.2% of the failed server population) contribute 7.17% of the total server failures, and consequently are responsible for 5% of total task failures due to server failures. This behaviour contrasts with work reported in [14], which observed that a small minority of servers incurred a larger proportion of failures at 70%. Figure 6.16(a) and 6.16(b) show the 3006 tasks that failed due to server failures in the nine servers which experienced the most failures, and their corresponding energy waste, respectively. It is observable that server 7, which experiences 13 failures, contained the lowest proportion of energy waste. We postulate the reason for this result is that although the server is logged by the system as recovered, it continues to exhibit incorrect service. While this behaviour reduces the

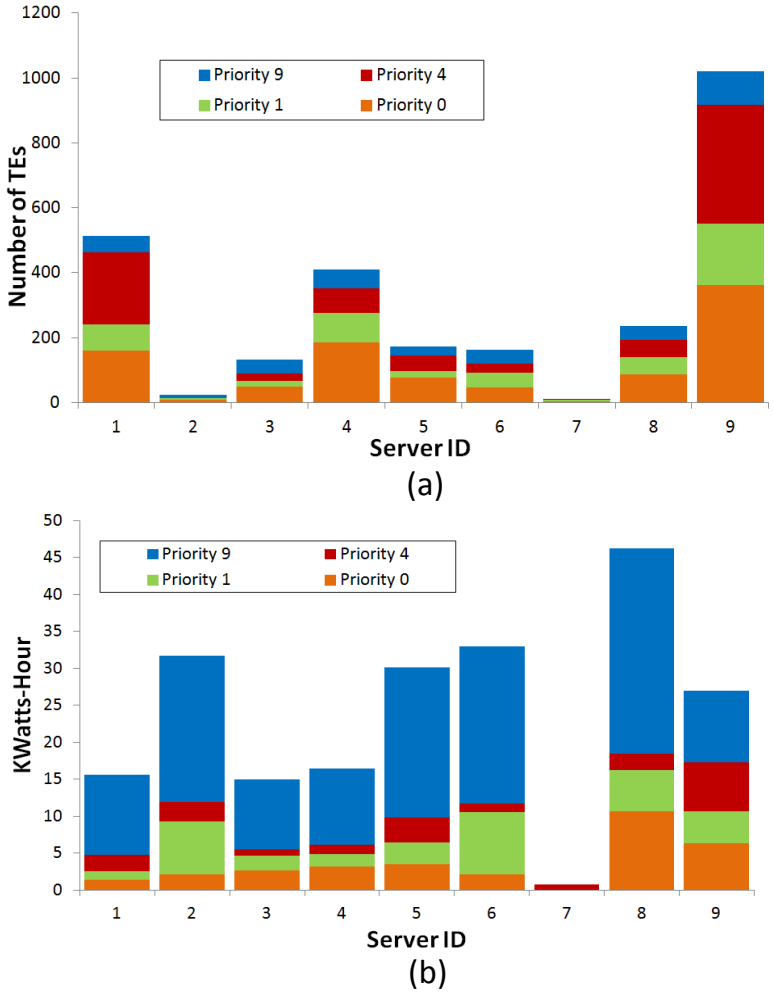


Figure 6.16 Top nine failed nodes (a) Failed tasks, (b) Energy waste.

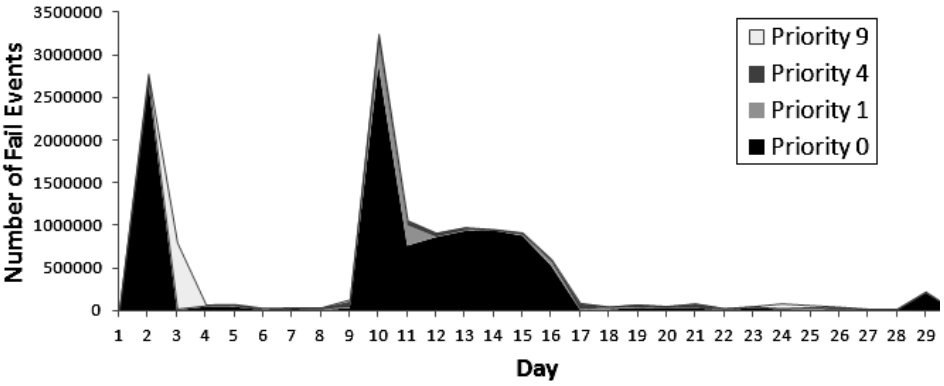
availability of the system, it does not result in a significant negative impact of energy waste within the system.

In contrast, server 2 experienced 23 failures and a significantly larger proportion of energy wasted of up to 32 KWatts-Hours. This is a result of the larger temporal distance between server failures, allowing tasks to execute more work prior to failures. This is particularly noticeable for tasks with longer execution times - priority 9 tasks which are production tasks that only compose a small proportion of the total scheduled tasks as shown in Figure 3.6 in Chapter 3.3, but represent a significant proportion of the total energy wasted due to server failures. We can observe from this behaviour that a small proportion of tasks with high priority result in a proportionally higher waste of energy consumption due to server failures.

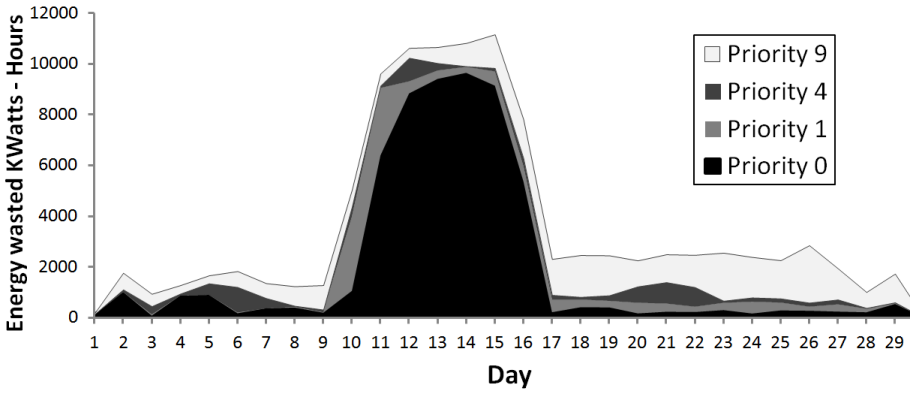
6.5 Energy Waste Impact on the System

Figures 6.17(a) and 6.17(b) depict the number of failure events and the energy waste due to task failures over the trace log time span, respectively. It is observable that up to 80% of failures are priority 0 task failures. This is inflated by a number of days that experience a substantial number of FAIL events. Such behaviour is clearly produced by the activity of specific users presented in Figure 6.13 that not only submit a large number of tasks, but also introduce a large number of task resubmissions which fail repeatedly resulting in crash-loops. When omitting the influence of these users, the number of events and their respective energy waste produced by failures on Priority 0 tasks is very similar to other days, even lower than Priority 9 which produces proportionally more energy waste.

By filtering the energy waste produced by EVICT and KILL events due to server failures from energy waste produced by all task FAIL events, tasks and servers contribute 13% and 8% of the total Termination Event energy waste, respectively as shown in Figure 6.18. This translates to 9.91% and 6.10% in the context of the total energy consumption of the datacenter, which includes energy consumption of server operation and successful task execution. This represents a significant amount of energy waste within a Cloud datacenter and identifies an important point of improvement that can lead to reducing operational costs while



(a)



(b)

Figure 6.17 Daily task failure (a) Number of events (b) Energy Waste.

maintaining provisioned QoS. For example, by minimizing the impact of server failures on long running tasks it is possible to reduce energy waste by up to 8%. Within this context, the application of mechanisms such as those presented in [2] for checkpointing can be applied in order to maintain the progress of long running tasks and reduce the amount of repeated computation.

However, in the case of task FAIL events, the use of checkpointing may by an ineffective measure and in some cases increase the amount of energy waste due to the high frequency of failure occurrence in lower priority tasks. In this case, improved policies to avoid highly recurrent resubmissions such as those described in [123] could reduce the proportion of Termination Event energy waste by up to 13%.

From the analysis, there are two well defined scenarios where system failures result in a substantial negative impact on the system environment. The first is task failures that affect low priority tasks in 80% of cases, with a MTBF of approximately 0.97 hours; this is predominately driven by 10 users whose tasks

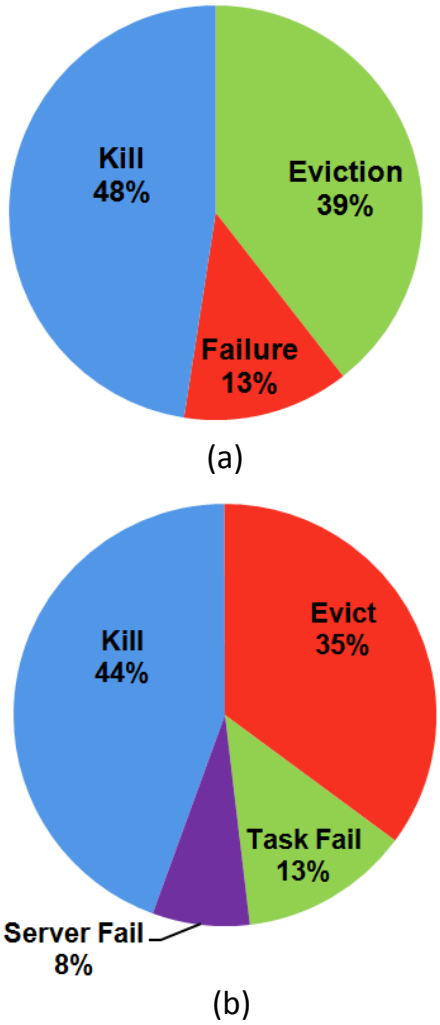


Figure 6.18 Energy waste of trace log (a) TEs, (b) TEs and failure events breakdown.

represent 90% of the total FAIL events in the environment due to crash-loops, which occur during days 2 and 10. The second involves task termination due to server failures, affecting 30% of tasks with high priority and a MTBF of 58.72 hours. This scenario is driven by hardware and software failures of servers uniformly distributed according to the size of their population. This indicates that not one of the server types within the trace log fails more frequently than the others in proportion to their respective population.

6.6. Summary and Application of Analysis

Although the results obtained in this analysis using the method described in Chapter 6.2 are specific to the studied environment, the findings of this work can be used as a baseline for analysis of similar practical systems. Researchers and practitioners can use the derived observations and conclusions to develop,

enhance and evaluate energy-aware dependable mechanisms as well as identify specific scenarios when and where failures have a significant impact on the energy waste within the system. The results of this analysis and how they can be applied are stated in the following paragraphs:

Provide failure and repair model parameters derived from empirical data to understand realistic system operation. The results presented can be used by other researchers to support realistic assumptions for failures within large-scale Cloud datacenters. Furthermore, the statistical properties and model parameters for failure and repair times can be leveraged to develop simulation environments that reflect realistic operational conditions, which can greatly assist in simulating realistic behaviour of Cloud workload by either introducing failure parameters or enhancing Cloud workload behaviour that currently depend on theoretical parameters and assumptions for failure characteristics.

Improve the effectiveness of developing and evaluating energy-aware dependable mechanisms. To give a practical example, failure-aware scheduling [96][107] focuses on developing more effective resource management in order to increase task reliability and system availability. Current mechanisms focus on improving the availability, reliability and performance of the system and do not consider energy-efficiency; an increasingly important factor in large-scale systems. As demonstrated in the analysis, unique server platforms exhibit different energy profiles at the same system load, which may result in a failure-aware scheduler to place tasks onto a server with lower capacity but is more energy-inefficient. As a result, it is possible to develop a failure-aware scheduling algorithm that selects servers for task allocation which provides a balance between optimal server reliability and energy-efficiency task execution. The work presented within this study can be leveraged in order to evaluate the effectiveness of such a mechanism and quantify the improvements in energy-efficiency based on empirical data as opposed to relying on theoretical values for energy-waste of software and hardware in large-scale systems.

In addition, the findings presented in this work can be applied in the following ways:

To support and adjust the claims of failure energy-aware mechanisms according to the characteristics of a real environment. Although existing theoretical analyses remark that significant energy waste is produced by failures, and propose elaborated mechanisms to address this problem, they do not present or discuss any insight into the actual amount of energy waste based on empirical findings. In fact, the claimed improvements against quantified waste in real operational scenarios when deploying energy-aware mechanisms are never contrasted, limiting their effectiveness.

Assist practitioners from similar system environments to decide the appropriate dependability mechanisms and when to apply them in order to maximize effectiveness. As discussed in Chapter 2.3.3, failures within large-scale systems are the norm rather than the exception, resulting in Cloud providers needing to decide what type of faults they should invest time and resources in correcting when considering their impact on system dependability, provisioned QoS, energy waste and development cost. For example, the work in [190] states that there are a number of limitations in applying fault-tolerant run-time techniques such as checkpointing, due to their potential to not only introduce high overhead, but more importantly difficulty in deciding when and where to apply such mechanisms effectively. The results presented in this work provide quantified dimensions for Cloud failures and energy waste which can be leveraged when making decisions to deploy a mechanism such as checkpointing, as well as deciding what type of workload to apply checkpointing.

To delimitate the energy waste produced by the "normal" operational inefficiencies and those introduced by task and server failures. Understanding the causes and characteristics of these inefficiencies can assist in identifying the most effective course of action to reduce their negative impact on the system.

Understanding the sources and dimensions of these inefficiencies helps to identify the most effective courses of action to reduce their negative impact. From the studied environment it is noticeable that although failures introduce close to 21% of the total energy waste, 79% of energy waste is introduced by scheduling operations such as KILL and EVICT. Such findings are critical in future research areas which aim to realistically model large-scale system environments [191].

6.7 Summary

This chapter has presented an in-depth failure analysis and failure related energy of a large-scale production Cloud computing datacenter. As outlined in the overview, this has provided:

- *The first holistic empirical failure analysis of tasks and servers in Cloud datacenters.*
- *The first empirical quantification and analysis of energy waste produced by failures and termination events within Cloud datacenters.*

The method of filtering and identifying failures for both tasks and servers based on supporting relevant literature and data observations from the trace log has been presented and discussed in detail. Furthermore, the energy profiles for unique server platforms at different levels of resource utilized are discussed. The analysis has been primarily divided into two sections: failure analysis and failure-related analysis.

The failure analysis focuses on coarse-grain statistics of Termination Events within the system, as well as analyzing the statistical properties and probability distribution modelling of task and server MTBF and MTTR both temporally and spatially. From the analysis it is observable that tasks and servers exhibit varying degrees of failure characteristics by task priority and server architecture type, respectively. Furthermore, there are predominantly two specific types of failure characteristics identified: tasks of low priority which fail early into their lifespan and are repeatedly rescheduled forming crash-loops, and higher priority tasks which fail later into their lifespan.

The failure-related energy analysis focuses presents the temporal and spatial properties of energy waste generated by server and task failures. This specifically includes the analysis of the temporal and spatial characteristics of energy waste due to server and tasks failures. This analysis demonstrates that there are two specific scenarios for energy waste within the Cloud datacenter; frequent task failures in low priority tasks and server failures affecting longer running production tasks resulting in 13% and 8% energy waste, respectively. Furthermore, it demonstrates that energy waste can be produced under a

number of different conditions and scenarios including user behaviour, server architecture and task priority type.

Finally, this chapter has discussed the application of the work. Specifically, how the findings in this chapter can be leveraged by researchers in order to enhance understanding inefficiencies in system operation, developing realistic Cloud environment simulation and improving energy-aware dependable resource management.

The final chapter of this thesis presents the conclusions of this research and discusses future work directions.

7 Conclusion and Future Work

7.1 Summary

The work in this thesis presents the empirical analysis and characterization of a large-scale Cloud computing datacenter. The need for analytics of large-scale Cloud systems, and their benefits for enhancing research and technical operation within the Cloud computing domain are discussed in detail. Specifically, this work for the first time presents a holistic analysis and method to characterise components within the Cloud computing environment including users, tasks, servers, failures and its respective energy waste. These analyses are leveraged for practical usage, including providing distribution parameters for accurate simulation of Cloud environments, quantifiable impact of operational inefficiency within production systems, and have been used to enhance energy-efficient resource management mechanisms.

Chapter 2 introduces the concept of the system environment and its respective components, as well as the evolution of the modern distributed system to Cloud computing. The concept and taxonomy of Cloud computing is presented in detail, including identified characteristics, actors, deployment models and service models. The concept of dependability, and how it can be used to enhance Cloud computing research is explored and discussed in detail. It is shown that while Cloud fault-tolerance is an active research area, there is a challenge in evaluating its effectiveness as current Cloud datacenter assumptions rely on theoretical, small-scale or non-Cloud systems for experiment parameters.

Next, the concept of system analytics and how it can be used to enhance Cloud computing research is demonstrated. The current state-of-the-art in analytics for Cloud components are presented and discussed in detail, and current gaps in the literature are identified. Finally, the importance and applicability of Cloud analytics is presented, including a discussion on how it can be used to enhance Cloud technical and commercial operation. From this chapter it is demonstrated that there exist a number of gaps within holistic Cloud datacenter analytics, including lack of user characterization in workload models, large-scale empirical failure analysis and quantifying the impact of failures on energy waste.

Chapter 3 presents the case study trace log of a production Cloud datacenter. This chapter includes the description of the system, the life cycle of tasks and servers, and attributes including resource utilization, and events of Cloud components within the system. Additionally the analysis infrastructure - composed of a cluster of machines running Apache MapReduce and HIVE - used to extract the large volume of data in a timely manner is presented and detailed. This chapter concludes by presenting and discussing the statistical properties and coarse-grain statistics of high level operation of the Cloud datacenter; presenting an overview of provisional evidence of heterogeneity in terms of users, tasks and server behaviour and characteristics.

Chapter 4 presents a method and in-depth analysis of workload behavioural patterns and characteristics within Cloud computing datacenters. The workload model which defines the relationship between users and tasks, and their respective key attributes are defined and explained in detail. The subsequent empirical analysis when applying the method to the Cloud datacenter case study quantifies the large degree of heterogeneity in user and task behaviour spatially and temporally. This heterogeneity is quantified in terms of resource estimation and submission patterns, as well as resource utilization and execution length for users and tasks, respectively. Furthermore, this chapter presents detailed probability distribution analysis of users and tasks, and extracts their respective parameters for other researchers to leverage in their own experiments. These distribution models are implemented in CloudSim, a popular Cloud computing simulation framework, in order to statistically validate their accuracy compared to the empirical data. Finally, this chapter discusses practical uses of the analysis presented in this chapter, presenting two concrete mechanisms within the domain of energy-efficient resource management - overallocation and performance interference.

Chapter 5 presents an empirical analysis of server characteristics within the Cloud datacenter case study, including resource utilization and inefficiency patterns. The method of extracting resource utilization per server, as well as quantifying the amount of resource utilization wasted due to Termination Events is detailed. Results indicate server resource utilization is between approximately 25% - 45% and 50% for CPU and memory, respectively: with this utilization

varying by server architecture type, as well as temporal patterns due to user submission patterns discussed in Chapters 2 and 4. Furthermore, our analysis results demonstrate that 4.54% - 14.22% and 1.29% - 7.61% of CPU and memory, respectively is wasted per individual server, with this inefficiency varying by server architecture type. The reasons for this behaviour are postulated to be due to the behaviour of the resource scheduler, as well as the type of the workload submitted into the system.

Chapter 6 presents an empirical analysis of failure and failure-related energy waste within a large-scale Cloud production environment. The method of identifying and filtering failures of tasks and servers is detailed, supported by the relevant literature and data observations from the case study trace log and past failure analysis of distributed systems. Furthermore, the profiles of server architectures and their respective energy profiles are defined and discussed in detail in order to quantify the total energy waste generated due to failures and Termination Events within the Cloud datacenter. Analysis results demonstrate that the failure and repair characteristics of tasks and servers can manifest in a number of conditions, and that the failure characteristics of tasks can be identified: tasks of low priority fail early within their lifespan due to repeating failure loops, and higher running tasks fail later in their lifespan. Furthermore, our analysis results show that energy waste is generated from two concrete conditions: frequent task failure of low priority tasks, and server failures affecting higher priority tasks which run for longer, resulting in 13% and 8% energy waste, respectively. Finally, this chapter discusses the application of this work, including the significance of these results and its potential for enhancing energy-aware dependable resource mechanisms.

7.2 Research Contributions

The identification of in-depth analysis of operational traces from Cloud computing datacenters as an effective means to comprehensively understand system behaviour and enhance system assumptions of Cloud research. There is a critical need for comprehensive and holistic analysis of Cloud computing datacenters in order to enhance our understanding of system characteristics and behaviour, as well as to validate and provide realistic system assumptions for Cloud datacenter environments. A large body of Cloud computing research

currently depend on theoretical values, non-Cloud systems or test beds for constructing research assumptions which may not accurately reflect realistic operation behaviour and characteristics. This thesis provides a detailed discussion of why analytics are a critical requirement in Cloud computing research, and presents the practical advantages of such results for research and commercial problems.

A solution to the challenges of analyzing large-scale Cloud environments to obtain meaningful results and identify relationships between components within the system. Analysing large-scale Cloud environment is challenging due to the volume of data generated by the system, complex relationships and interactions between these components, and the requirement of expert domain knowledge of Cloud computing and datacenter operation. The work within this thesis provides a framework for comprehensively models the relationship and interactions of Cloud components as well as the life cycle of servers and tasks within the datacenter.

An empirical analysis of Cloud datasets to study and model realistic user behaviour, task classification, Cloud utilization models and failure characteristics and models. There is a critical need to empirically study real Cloud computing datacenters in order to quantify system behaviour, as well as provide parameters which can be used practically by researchers and Cloud providers. This thesis comprehensively analyzes and model key Cloud components of workload, servers, failures and their respective energy waste. From the results it is possible to observe that there exists substantially heterogeneous behaviour and characteristics in terms of user submission and resource estimation patterns, task execution length and resource utilization, and task MTBF and MTTR. Additionally, this thesis provides the distribution parameters of workload and failures to be used for the construction of realistic simulation environments, and can be applied to enhance the practicality of energy-efficiency resource management techniques.

The study and quantification of operational inefficiencies within Cloud environment. This work for the first time quantifies the amount of resource and energy wasted due to Termination Events and failures within a large-scale production Cloud computing datacenter. Our results demonstrate that the

resource inefficiency of individual server varies per server architecture type, ranging between 4.54 - 14.22% and 1.29 - 7.61% for CPU and memory, respectively due to the behaviour of the resource scheduler and task failures. Furthermore, the analysis demonstrates two scenarios for energy waste; frequent task failures in low priority tasks and server failures affecting longer running tasks resulting in energy waste of 13% and 8%, respectively. These results demonstrate that inefficiencies can be produced under a number of conditions due to user, server and task behaviour, and highlight well defined areas of improvement for Cloud datacenters.

7.3 Overall Research Evaluation

The four research objectives of this thesis are discussed in Chapter 1. The success criteria of this research in relation to achieving the proposed research objectives are listed as follows:

i) *To enable a more thorough understanding of the issues in accurately modelling Cloud computing environments and comprehensively study how Cloud behavioural characteristics impact the system.* This thesis has explored and discussed the challenges in accurately characterizing and modelling the Cloud environment, and has quantified the key characteristics which impact the system environment. Chapter 4 has demonstrating that tasks exhibit a large degree of heterogeneity in terms of resource utilization and execution length due to the behavioural patterns of different user types, and that it is possible to model and simulate their behaviour. Chapters 5 and 6 has identified and quantified the scenarios where server resource inefficiency and failure-related energy waste primarily occur within a Cloud datacenter, respectively.

ii) *To provide an in-depth method of holistic analysis for real Cloud datasets to study and model Cloud behaviour*. This thesis presents a detailed method of analysis for each of the key components within Cloud datacenters for workload characterization, quantification of server utilization and inefficiencies, failure identification and the quantification of failure-related energy waste. The derived methods have been abstracted to a level where they can be applied to a numerous Cloud datacenter trace logs, agnostic of its schema and format.

iii) *Empirical analysis and modelling of large-scale Cloud computing behavioural patterns and characteristics.* This thesis has presented a comprehensive empirical analysis of a Cloud datacenter, including the quantification and modelling of its respective components including workload, servers, failures and failure-related energy waste. The practical uses of the analysis results and modelling have been provided and discussed, and realistic simulation models and enhanced energy-efficiency resource management mechanisms are demonstrated.

iv) *To identify and quantify operational inefficiency in terms of wasted resource utilization and energy waste due to failures within large-scale Cloud environments.* Through the use of the developed analysis method detailed in this thesis, it is possible to quantify operational waste within Cloud datacenters in terms of resource utilization and energy waste due to failures. Furthermore, the analysis has identified the causes and scenarios where operation waste is generated, providing datacenter providers and researchers empirical and quantifiable points of improvement when applying resource management mechanisms.

In summary, it can be observed that all four main research objectives have been successfully completed.

7.4 Future Work

There are a several ways that the work presented in this thesis could be enhanced as well as number of future research areas of opportunity which build upon the foundation of this research. They are summarized as follows:

7.4.1 Dataset

Analysis of additional Cloud datacenter trace logs. Although the selected case study trace log for applying the analysis methods provides an extensive system-scale and time frame in comparison to current state-of-the-art Cloud datacenters, there are limitations in abstracting the findings of the system to the general behaviour of Cloud datacenters. This is particularly true when quantifying specific elements such as total energy consumption, number of users, types of applications, etc. Future work should include applying the

methods described in this thesis with a number of different Cloud datacenter trace logs, in order to validate a more generic model of Cloud datacenters.

Analysis of non-obfuscated server attributes for Cloud datacenter trace logs. A number of strong assumptions were made within the thesis due to the obfuscation of attributes within the case study trace log, predominantly focused on the physical characteristics of servers which are represented as normalized resource capacities and obfuscated platform architecture as discussed in Chapters 3.3. While it was possible to take values from the SpecPower Benchmark and match similar servers based on the similarities of characteristics, it would be much more effective to work with the direct type of servers within the case study trace log. This will result in deriving more accurate energy models and resource utilization of servers and tasks within the Cloud datacenter.

More detailed failure logs of Cloud datacenter trace logs. Due to the nature of the case study trace log, it was necessary to identify task and server failures by using the failure identification method presented in Chapter 6.2. Unfortunately, there is a lack of detailed information pertaining to diagnosing the cause of failures within tasks or servers, which are only represented at a coarse-grain level as discussed in Chapter 3.1. Future analysis of trace logs which include failure logs that provide detail of the specific component which has failed would greatly enhance diagnosing the cause of failures within Cloud datacenters.

7.4.2 Analysis Extension

The analysis presented within this thesis could be extended to explore a number of additional system characteristics.

Extension of workload models inclusive of job behaviour, and inclusion of task constraints based on server characteristics. As stated in Chapter 4.2.4, analyzing tasks instead of job behaviour was selected due to tasks representing the most fine grained characteristic of workload. It is possible for tasks to be grouped together within a single job belonging to a user; as a result, future work involves applying the described workload analysis method to study job characteristics and behaviour when contrasted against tasks. Furthermore, the workload model does not include the scheduling constraints of tasks onto servers, which can also be included to study their behaviour in comparison to non-constrained tasks.

Analysis of failure correlation within Cloud datacenters. As described in Chapter 2.3.1, it is possible for failures to be correlated, and to cause a fault-error-failure chain which cascades through the system environment. In this work, an assumption was made that all failures are unique and there is no explicit correlation between components as described in Chapter 6.2.2. This is due to the challenges identified in the current failure analysis literature when identifying failures which are temporally close together, as well as lacking sufficient time and resources to comprehensively explore failure correlation events. As a result, future work within this area should include dedicating more time to attempting to identify unique failure events and correlations and their consequent impact on failure-related energy waste.

7.4.3 Development of Cloud Mechanisms

This results presented within this thesis have already been implemented in a number of mechanisms including simulation environments and energy-efficient resource management mechanisms discussed in Chapter 4.7. However, there are a number of opportunities to further validate the effectiveness of the derived results by the following:

Actual physical prototyping and experimentation of the described fault-tolerant mechanisms. Chapter 6.6 describes how the results of the failure and failure-related energy analysis can be used in order to create checkpointing and failure-aware scheduling mechanisms. The next step of this research would be to develop an implementation to be deployed within real systems for experimentation to study their effectiveness when compared against the current state-of-the-art. A practical example is the work in [47] which injects faults into two physical Cloud systems in order to evaluate their tolerance to Byzantine faults; such work would be greatly enhanced by using the results extrapolated in this thesis when calculating the MTBF and MTTR of the Cloud components within each system.

Integrate all derived simulation parameters into a holistic Cloud simulation framework. Chapter 4.6 described the results of the integration of workload and server parameters into the CloudSim framework. However, in order to build more realistic and holistic Cloud simulations, it is necessary to include additional behavioural characteristics of workloads and servers, including simulation MTBF

and MTTR, as well as quantifying the energy consumption of system operation. This would allow further realism of simulating Cloud datacenters, as well as studying the impact of applying developed mechanisms on the system environment in terms of performance, energy and dependability.

References

[1] B. Schroeder, G. A. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," IEEE Transactions Dependable and Secure Computing, vol. 7, pp. 337-351, 2010.

[2] A. Gainaru, F. Cappello, M. Snir, and W. Kramer, "Fault prediction under the microscope: a closer look into HPC systems," in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis Salt Lake City, Utah: IEEE Computer Society Press, pp. 1-11, 2012.

[3] J. A. Quiane-Ruiz, C. Pinkel, J. Schad, and J. Dittrich, "RAFTing MapReduce: Fast recovery on the RAFT," in Proceedings of IEEE International Conference on Data Engineering (ICDE), pp. 589-600, 2011.

[4] T. Anderson and P.A. Lee, “Fault Tolerance: Principles and Practice”, Prentice Hall, 1981.

[5] A. Avizienis , J.C. Laprie , B. Randell , C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE Transactions on Dependable and Secure Computing, vol. 1, p.11-33, 2004.

[6] G. Alonso, F.Casati, H.i Kuno, V. Machiraju, "Web Services: Concepts, Architectures and Applications", Springer Publishing Company, 2010.

[7] P.A.Bernstein, "Middleware: A Model for Distributed System Services", Communications of ACM, Vol. 39, Issue 2, pp. 86-98, 1996.

[8] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Journal of Future Generation Computer Systems, Vol. 25 Issue 6, Pg 699-616, 2009.

[9] D. Parkhill, " The Challenge of the Computer Utility", Addison-Wesley, 1966.

[10] R. Buyya, R. Ranjan, R.N. Calheiros, "InterCloud: Utility-oriented Federation of Cloud computing Environments for Scaling of Application Services," in Proc. 10th Int. Conf. on Algorithms and Architectures for Parallel Processing, pp. 13-31, 2010.

[11] P.Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I.Pratt, A.Warfield, "Xen and the Art of Virtualziation", in

proceedings of SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles, pp. 164-177, 2003.

[12] W. Gropp, E. Lusk, T.Sterling "Beowulf Cluster Computing with Linux", MIT Press: Cambridge, 2003.

[13] G. F. Pfister "In Search of Clusters", Prentice-Hall PTR: Upper Saddle River, 1998.

[14] R.Buyya "High Performance Cluster Computing: Architectures and Systems", Prentice-Hall PTR: Upper Saddle River, 1999.

[15] R.Buyya, T.Cortes, H.Jin "Single System Image (SSI)", the International Journal of High Performance Computing Applications, pp. 124–135, 2001.

[16] C.S. Yeo. R. Buyya, "A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing", Journal of Software - Practise & Experience, Vol. 36, Issue 13, pp 1381-1419, 2006.

[17] R. Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications", Proceedings of the IEEE 1st International Conference on Peer-to-Peer Computing, 2002.

[18] D.S. Milojibic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S.Rollins, Z. Xu, "Peer-to-Peer Computing", Technical Report HPL-2002-57R1, HP Labs, 2003.

[19] S. Saroiu, P.K.Gummadi, S.D. Gribble "Measurement Study of peer-to-peer File Sharing Systems", in the Proceedings of Multimedia Computing and Neworking, 2002.

[20] M. Ripeanu "Peer-to-Peer Architecture Case Study: Gnutella Network", Technical Report, 2001.

[21] I. Foster, “What is the Grid? A Three Point Checklist”, in GRIDToday, 2002.

[22] I. Foster, C. Kesselman, S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, in Proceedings of the International Journal on Supercomputer Applications, pp.200-222, 2001.

[23] “UDDI Executive White Paper,” White Paper, uddi.org, 2001

[24] P. Patel, A.H. Ranabahu, A.P.Sheth "Service Level Agreement in Cloud Computing", in Proceedings of Cloud Workshops at OOPSLA, 2009.

[25] "2012 Census Report: Energy", DCD Intelligence White Paper, 2012.

[26] "Cisco Visual Networking Index: Forecast and Methodology, 2012-2017", Cisco, 2013.

[27] U. Hölzle, L. A. Barroso, "The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines", Synthesis Lectures on Computer Architecture, 2009.

[28] " Gartner Global IT Spending Forecast 2013" Gartner Report, 2013.

[29] "Going Digital: Where is our data?", Data Centre Alliance, New Statesman, 2013.

[30] P.Mell, T. Grance, "The NIST Definition of Cloud Computing," National Institute of Science and Technology, 2011.

[31] R. Buyya, C. S. Yeo, and S.R Venugopal, "Market-oriented Cloud Computing: Vision, Hype, and Reality for Delivering it Services as Computing Utilities", in Proceedings of 10th IEEE International Conference on High Performance Computing and Communications, 2008.

[32] N. Leavitt "Is Cloud Really Ready for Prime Time?" Computer Journal, Vol 42, pp. 15-20, 2009.

[33] "Forecast Overview: Public Cloud Services 2011-2016, 4Q12", Gartner report, 2013.

[34] G.Pallis, "Cloud computing: The New Frontier of Internet Computing", IEEE Internet Computing, pp. 70-73, 2010.

[35] P.M. Corcoran, "Cloud Computing and Consumer Electronics: A Perfect Match or a Hidden Storm?", IEEE Consumer Electronics Magazine, pp. 14-19, 2012.

[36] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, F.Galan, "RESERVOIR - When One Cloud is not Enough" IEEE Computer, pp. 44-51, 2011.

[37] Z. Zheng, Y. Zhang, M.R. Lyu, "CloudRank: A QoS-Driven Component Ranking Framework for Cloud Computing", in the Proceedings of International Symposium on Reliable Distribution Systems (SRDS) pp. 184-193, 2010.

[38] M. Andrealini, S. Casolari, M. Colajanni, M. Messori, "Dynamic Load Management of Virtual Machines in Cloud Architectures", In Proceedings of IEEE conference on Cloud Computing, 2009.

[39] J. Zhan, L. Wang, W. Shi, S. Gong, X. Zang "Phoenix Cloud: Provisioning Resources for Heterogeneous Cloud workloads", First Workshop on Cloud Computing and its Applications (CCA08), 2008.

[40] A, Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, N. Sharma, "Towards Autonomic Workload Provisioning for Enterprise Grids and Clouds", in the Proceedings of the 10th IEEE/ACM International Conference on Grid Computing, pp. 50-57, 2009.

[41] F. Liu, J.Tong, J.Mao, R. Bohn, J.Messina, L.Badget, D.Leaf, "NIST Cloud Computing Reference Architectue: Recommendations of the National Institute of Standards and Technology", NIST Special Publication 500-292, 2011.

[42] D.Amrhein, et al. "Cloud Computing Use Cases", White paper, Cloud Computing Use Case Discussion Group, 2009.

[43] L. Vaquero, L. Rodero Merino, J. Caceres, M. Lindner, "A break in the clouds: towards a cloud definition", ACM SIGCOMM Computer Communications Review, Vol. 39(1), pp. 50 – 55, 2009.

[44] H. S. Gunawi, T. Do, J. M. Hellerstein, I. Stoica, D. Borthakur, J. Robbins, "Failure as a Service (FaaS): A Cloud Service for Large-Scale, Online Failure Drills", Technology Report, 2011.

[45] L. Wang, G. Von Laszewski, M. Kunze, J.Tao " Cloud Computing: A Perspective Study" New Generations of Computers, Vol. 28(2), pp 137, 2010.

[46] H. Al-Aqrabi, L.Liu, J. Xu; R.Hill, N.Antonopoulos; Y. Zhan, "Investigation of IT Security and Compliance Challenges in Security-as-a-Service for Cloud Computing," in the Proceedings of the 15th IEEE International Symposium on Object/Components/Service-Oriented Real-Time Distributed Computing Workshops, pp. 124-129, 2012.

[47] P. Garraghan, P.Townend, J.Xu " Using Byzantine Fault-Tolerance to Improve Dependability in Federated Cloud Computing", International Journal of Software and Informatics, Vol. 7, Issue 2, Pg 221-237, 2013.

[48] A. Celesti, F. Tusa, M.Villari, A. Puliafito "Improving Virtual Machine Migration in Federated Cloud Environments", International Conference on Evolving Internet, 2010.

[49] I.Goiri, J.Guitart, J. Torres " Characterizing Cloud Federation for Enhancing Providers' Profit", Proceedings of 3rd IEEE International Conference on Cloud Computing, 2010.

[50] M. A. El-Refaey and M. A. Rizkaa, "Virtual Systems Workload Characterization: An Overview," in Proceedings of 2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, pp. 72-77, 2009.

[51] B. Sharma, et al., "Modelling and synthesizing task placement constraints in Google compute clusters," in Proceedings of ACM Symposium on Cloud Computing, pp. 1-14, 2011.

[52] G. Bruce Berriman, E.Deelman, G. Juve, M. Rynge, Jens-S. Vöckler "The application of cloud computing to scientific workflows: a study of cost and performance" Philosophical Transactions of the Royal Society A, 2012.

[53] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, D. Epema " A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing" Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, Vol. 34, pp. 115-131, 2010.

[54] K.H. Kim, A. Beloglazov, R. Buyya " Power-aware Provisioning of Cloud Resources for Real-time Services" in the Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science, 2009.

[55] IBM "Success in the Cloud: Why workload matters" IBM Corporation,White paper, 2012.

[56] S.Crago, K. Dunn, P.Eads, L. Hochstein, D. Kang "Heterogeneous Cloud Computing", IEEE International Conference on Cloud Computing, 2011

[57] T. Sridhar, "Cloud Computing -A Primer Part 1: Models and Technologies," the Internet Protocol Journal, vol. 12, pp. 2-19, September 2009

[58] VMware "Understanding Full Virtualization, Paravirtualization and Hardware Assist", VMWare, White paper, 2007.

[59] IBM "Power Systems: Introduction to Virtualization" IBM, Technical Report, 2009.

[60] Dell, "An Overview of Xen Virtualization" Dell, White Paper, 2005.

[61] M. Helsley, " LXC: Linux container tools", IBM report, Technical Report 2009.

[62] K. Kolyshkin " Virutalization in Linux", Technical Report, 2006.

[63] F. W. Clegg, G. S. Ho, S. R. Kusmer, J. R. Sontag "The HP-UX operating system on HP Precision Architecture computers", Hewlett-Packard Journal, 1986.

[64] A. Andreiux, K.Czajkowski, A.Dan, K. Keahey, H.Ludwig, J. Pruyne, J.Rofrano, S.Tuecke, M.Xu, "Web services agreement specification (WS-Agreement)", 2004.

[65] M.Alhamad, T. Dillon, E.Chang " Conceptual SLA Framework for Cloud Computing" in Proceedings of 4th IEEE International Conference on Digital Ecosystems and Technologies, pp. 606-610, 2010.

[66] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared", in the Proceedings of Grid Computing Environments Workshop, pp.1-10, 2008.

[67] L.F. Bittencourt, C.R. Senna, E.R.M. Madeira, "Enabling Execution of Service Workflows in Grid/Cloud Hybrid Systems," in Proceedings of IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS Wksps), pp.343-349, 2010.

[68] X.Wang, B.Wang, J. Huang, "Cloud Computing and its Key Techniques", in Proceedings of IEEE International Conference on Computer Science and Automatic Engineering, pp. 404-410, 2011.

[69] J.C. Laprie, “Dependability – Its Attributes – Impairments and Means”, in Predictably Dependable Computing Systems, Springer-Verlag, pp. 3-24 1995.

[70] J.C. Laprie, "Dependability of computer systems: concepts, limits, improvements," in proceedings of the Sixth International Symposium on Software Reliability Engineering, pp.2-11, 1995.

[71] F.Cristian, H.Aghili, R.Strong, "Clock Synzhronization in the Presence of Omissions and Performance Faults, and Processor Joins," in Proceedings of 16th International Symposium on Fault Tolerant Computing Systems, 1986.

[72] D.Powell, "Failure Mode Assumptions and Assumption Coverage", in Proceedings of 22nd IEEE International Symposium of Fault-Tolerant Computing, pp. 386-395, 1995.

[73] K.Kim, "Fair Distribution of Concerns in Design and Evaluation of Fault-Tolerant Distributed Computer Systems" Computer Communications, Vol 17. pp. 699-707, 1994.

[74] F.Cristian, H.Aghili, R.Strong, D.Dolev, "Atomic Broadcast: from Simple Message Diffusion to Byzantine Agreement", in Proceedings of IEEE 15th International Symposium on Fault-Tolerant Computing, pp 993-940, 1987.

[75] L. Lamport, R.Shostak, M.Pease, "The Byzantine Generals Problem", ACM Transactions on Programming Languages and Systems, Vol. 4, pp. 382-401, 1982.

[76] X. Cruz "Assessing the Costs of Cloud Outages", Article, CloudTimes.org, 2012, Internet: http://cloudtimes.org/2012/06/30/costs-cloud-outages/

[77] J. Li, M.Humphrey, Y. Cheah, Y. Ryu, D.Agarwal, K.Jackson, C. van Ingen, "Fault Tolerance and Scaling in e-Science Cloud Applications: Observations from the Continuing Development of MODISAzure", in the proceedings of sixth IEEE International Conference on e-Science, pp. 246-253, 2010.

[78] S.Shankland "Amazon suffers U.S. outage on Friday' Internet" Article: Internet: http://news.cnet.com/8301-10784_3-9962010-7.html

[79] "Amazon S3- More than 449 Billion Objects" Amazon, Article, Internet: http://aws.typepad.com/aws/2011/07/amazon-s3-more-than-449-billion-objects.html

[80] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, R. Sahoo, "BlueGene/L Failure Analysis and Prediction Models," in Proceedings of the IEEE International Conference on Dependable Systems and Networks, pp. 425-434, 2006.

[81] K. Vaidyanathan, R. E. Harper, S. W. Hunter, and K. S.Trivedi, "Analysis and Implementation of Software Rejuvenation in Cluster Systems" in Proceedings of ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems, pp. 62–71, 2001.

[82] M. Lyu, V. Mendiratta, "Software Fault Tolerance in a Clustered Architecture: Techniques and Reliability Modelling", in Proceedings of IEEE Conference on Aerospace, pp. 141–150, 1999.

[83] H. Fadishei, H. Saadatfar, H. Deldari , "Job Failure Prediction in Grid Environment Based on Workload Characteristics," in Proceedings of 14th International Computer Conference, pp.329-334, 2009.

[84] R.K. Iyer, D.J. Rossetti , "Effect of System Workload on Operating System Reliability: A Study on IBM 3081," IEEE Transactions on Software Engineering, vol.11, no.12, pp. 1438-1448, 1985.

[85] Z.Zheng, T. Zhou, M.Lyu, I.King "FTCloud: A Component Ranking Framework for Fault-Tolerant Cloud Applications" in Proceedings of IEEE 21st International Symposium on Software Reliability Engineering, pp 398-407, 2010.

[86] Y.Dai, B.Yang, J.Dongorra, G.Zhang, "Cloud Service Reliability: Modelling and Analysis", in Proceedings of 15th IEEE Pacific Rim International Symposium on Dependable Computing.

[87] J. Xu, A. Bondavalli, F. Di Giandomencio "Dynamic Adjustment of Dependability and Efficienct in Fault-Tolerant Service" Predictably Dependable Computing Systems, pp. 155-172, 1995.

[88] Z. Zheng, M. Lyu, "A Distributed Replication Strategy Evaluation and Selection Framework for Fault Tolerant Web Services,", in Proceedings of IEEE International Conference on Web Services, pp. 145-152, 2008.

[89] Q. Zheng, "Improving MapReduce Fault-tolerance in the Cloud," in Proceedings of 2010 IEEE International Symposium on Parallel and Distributed Processing: Workshops and Phd Forum (IPDPSW), pp.1-6, 2010.

[90] Q. Wei, B.Veeravalli, B. Gong, L. Zeng, D. Feng, "CDRM: A Cost-Effective Dynamic Replication Management Scheme for Cloud Storage Cluster," in Proceedings of IEEE International Conference on Cluster Computing (CLUSTER), pp. 188- 196, 2010.

[91] S. Malik, F.Huet, "Adaptive Fault Tolerance in Real Time Cloud Computing," in Proceedings IEEE World Congress on Services, pp. 280-287, 2011.

[92] W.Tsai, P. Zhong; J.Elston, X. Bai, Y. Chen, "Service Replication Strategies with MapReduce in Clouds," in Proceedings of 10th International Symposium on Autonomous Decentralized Systems (ISADS), pp.381,388, 2011.

[93] W.Zeng, Y. Zhao, K. Ou, W. Song, "Research on Cloud Storage Architectures and Key Teechnologies", in Proceedings of International ACM Conference on Interaction Sciences: Information Technology, Culture and Humans, pp. 1044-1048, 2009.

[94] H. Abu-Libdeh, L.Princehouse, H.Weatherspoon, "RACS: A Case for Cloud Storage Diversity", in Proceedings of 1st ACM Symposium on Cloud Computing, PP. 229-240, 2010.

[95] G. Tian, D. Meng, "Failure Rules Based Node Resource Provision Policy for Cloud Computing," in Proceedings of International Symposium on Parallel and Distributed Processing with Applications (ISPA), pp.397-404, 2010.

[96] S. Fu, "Failure-aware Resource Management for High-availability Computing Clusters with Distributed Virtual Machines", Elsevier Journal of Parallel and Distributed Computing, vol. 70, pp. 384-393, 2010.

[97] B. Javadi, J. Abawajy, R. Buyya, "Failure-aware Resource Provisioning for Hybrid Cloud Infrastructure", Elsevier Journal of Parallel and Distrbiuted Computing, vol. 72, pp. 1318-1331, 2012.

[98] C. Cachin, R. Haas, M. Vukolic, " Dependable Storage in the Intercloud: Research Report", IBM Research, 2010.

[99] I.Goiri, J. Guitart, J. Torres, "Characterizing Cloud Federation for Enhancing Providers' Profit", in Proceedings of IEEE 3rd International Conference on Cloud Computing, 2010.

[100] A. M. Law, W.D.Kelton, "Simulation, Modelling and Analysis: Third Edition", McGraw-Hill Series in Industrial Engineering and Management Science, 2000.

[101] G.S. Fishman, "Principles of Discrete Event Simulation," John Wiley & Sons, 1978.

[102] R. G. Sargent, "Verification and validation of simulation models," in Proceedings on Winter Simulation Conference, pp. 166-183, 2010.

[103] G. Wang, A.R. Butt, H. Monti, K.Gupta, "Towards Synthesizing Realistic Workload Traces for Studying the Hadoop Ecosystem," in Proceedings of IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, pp.400-408, 2011.

[104] Google. "Google Cluster Data V1", Available: http://code.google.com/p/googleclusterdata/wiki/TraceVersion1

[105] Qi Zhang, J.L. Hellerstein, R. Boutaba, "Characterizing Task Usage Shapes in Google Compute Clusters," in Proceedings of the 5th International Workshop on Large Scale Distributed Systems and Middleware, pp.2-8, 2011.

[106] A. K. Mishra, J.L. Hellerstein, W. Cirne, C. R. Das, "Towards characterizing cloud backend workloads: insights from Google compute clusters," SIGMETRICS Performance Evaluation Review, vol. 37, pp. 34-41, 2010.

[107] S. Kavulya, J. Tan, R. Gandhi, P. Narasimhan, "An Analysis of Traces from a Production MapReduce Cluster," in Proceedings of IEEE/ACT International Conference on Cluster, Cloud and Grid Computing, pp. 94-103, 2010.

[108] Yahoo. Yahoo! M45 Supercomputing Project . Available: http://research.yahoo.com/node/1884

[109] S. Aggarwal, S. Phadke, M. Bhandarkar, "Characterization of Hadoop Jobs Using Unsupervised Learning," in Proceedings of 2nd International Conference on Cloud Computing Technology and Science, pp. 748-753, 2010.

[110] A.L Freitas, N. Parlavantzas and J. Pazat, "Cost Reduction through SLA-driven Self-Management," in Proceedings on IEEE European Conference on Web Services, pp.117-124, 2011.

[111] J. Gray, "A Census of Tandem System Availability between 1985 and 1990", IEEE Transactions on Reliability , 39(4), 1990.

[112] R. K. Iyer, D. J. Rossetti, and M. C. Hsueh, "Measurement and modelling of Computer Reliability as Affected by System Activity", ACM Transactions on Computer Systems, 1986.

[113] D. Tang, R. K. Iyer, and S. S. Subramani, "Failure Analysis and Modelling of a VAX cluster system", in Proceedings of International Symposium on Fault-Tolerant Computing, 1990.

[114] Y. Liang, Y. Zhang, M. Jette,, A. Sivasubramaniam, R. Sahoo, "BlueGene/L Failure Analysis and Prediction Models," in Proceedings of the IEEE International Conference on Dependable Systems and Networks, pp. 425-434, 2006.

[115] H.Li, D.L. Groep, L.Wolters, J. Templon, "Job Failure Analysis and Its Implications in a Large-Scale Production Grid," in Proceedings of the International Conference on e-Science and Grid Computing, pp. 27, 2006.

[116] R. K. Sahoo, A. Sivasubramaniam, M. S Squillante, Y. Zhang. "Failure Data Analysis of a Large-Scale Heterogeneous Server Environment," in Proceedings of International Conference on Dependable Systems and Networks, IEEE Computer Society, pp. 772-781, 2004.

[117] L. Fiondella, S.S. Gokhale, V.B. Mendiratta "Cloud Incident Data: An Empirical Analysis," in Proceedings of IEEE International Conference on Cloud Engineering (IC2E), pp.241,249, 2013.

[118] K. V. Vishwanath and N. Nagappan. "Characterizing Cloud Computing Hardware Reliability", in Proceedings of ACM Symposium on Cloud Computing (SOCC), 2010.

[119] S. Harizopoulos, M. A. Shah, J.Meza, P.Ranganathan, "Energy Efficienecy: The New Holy Grail of Data Management Systems Research" in Proceedings of Fourth Biennial Conference on Innovative Data Systems, 2009.

[120] M.G. Patterson, "What is Energy Efficiency? Concepts, Indicators and Methodological Issues", Elsevier, Energy Policy, Vol. 24, No.5, pp. 377-390, 1996.

[121] R. Brown, et al."Report to Congress on Server and Datacenter Energy Efficiency Public Law 109-431", U.S. Environmental Protection Agency ENERGY STAR Program, 2007.

[122] Emerson Network Power "Energy Efficiency Cooling Solutions for Data Centers", White Paper, 2007.

[123] M. Pedram, "Energy-Efficient Datacenters," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, pp. 1465-1484, 2012.

[124] X. Fan, W. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” in Proceedings of International Symposium on Computer Architecture, 2007.

[125] "Power Efficiency Comparison of Enterprise-Class Blade Servers and Enclosures", DellInc., 2010. Available at: www.dell.com/.../BladePowerStudyWhitePaper_08112010_final.pdf

[126] EPA Conference on “Enterprise Servers and Datacenters: Opportunities for Energy Efficiency,” Lawrence Berkeley National Laboratory, 2006. Available at: http://hightech.lbl.gov/DCTraining/presentations.html.

[127] T. Nguyen, W. Shi, "Improving Resource Efficiency in Data Centers using Reputation-based Resource Selection," in Proceedings of the International Conference on Green Computing, IEEE Computer Society, pp. 389-396, 2010.

[128] T. Ropars, A. Guermouche, B. Ucar, E. Menese, L. V. Kale, F. Ceppello, "On the Use of Cluster-based Partial Message Logging to Improve Fault Tolerance for MPI HPC Applications," in Proceedings of International Conference on Parallel processing - Volume Part I, Springer-Verlag, pp. 567-578, 2011.

[129] H. Liu, "A Measurement Study of Server Utilization in Public Clouds," in Proceedings of IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), pp.435-442, 2011.

[130] Amazon EC2 Cloud, Amazon, Internet: http://aws.amazon.com/

[131] GoGrid Cloud, GoGrid, Internet: http://www.gogrid.com/

[132] Amazon EC2 VM sizes, Amazon, Internet: http://aws.amazon.com/ec2/instance-types/

[133] M. Kalyanakrishnam, Z. Kalbarczyk, R. Iyer, "Failure data analysis of a LAN of Windows NT based computers," in Proceedings of the IEEE Symposium on Reliable Distributed Systems, pp. 178-187, 1999.

[134] D. Oppenheimer, A. Ganapathi, D. A. Patterson, "Why do Internet Services Fail, and what can be done about it?," in Proceedings of USENIX

Symposium on Internet Technologies and Systems, USENIX Association, vol. 4, 2003.

[135] R. L. Keeney, R. S. Gregory, "Selecting Attributes to Measure the Achievement of Objectives", Operations Research, 2005.

[136] A.K Jain, M.N. Murty, P.J. Flynn "Data Clustering: a Review" ACM Computing Surveys, vol. 31, Issue 3, pp 264-323, 1999.

[137] M.J. Bayarri, J.O. Berger, "The Interplay of Bayesian and Frequentist Analysis", Statistical Science, Vol. 19, pp 58-80, 2004.

[138] B. Walsh, "Introduction to Bayesian Analysis", Lecture Notes for EEB 596z, 2002.

[139] J.Berger, "The case for objective Bayesian Analysis", Bayesian Analysis, pp 385-402, 2006.

[140] F. Hartwig, B.E. Dearing "Exploratory Data Analysis", SAGE, 1979.

[141] SAS Institute, "SAS OnlineDoc", Chapter 38: Distribution Modelling, SAS Institute Inc.: Manual, 1999.

[142] C. Tang, C.S. Raghavendra, "Correlation analysis and applications in wireless microsensor networks", The First Annual Conference on Mobile and Ubiquitous Systems: Networking and Services, pp.184-193, 2004.

[143] H. Mi, H. Wang, G. Yin, H. Cai, Q. Zhou, T. Sun, "Performance Problems Diagnosis in Cloud Computing Systems by Mining Request Trace Logs," in Proceedings of the IEEE Network Operations and Management Symposium (NOMS), pp.893-899, 2012.

[144] J.A. Hartigan, M.A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm", Journal of the Royal Statistical Society, pp. 100–108, 1979.

[145] D. T. Pham, S. S. Dimov, C. D. Nguyen, "Selection of K in K-means clustering," in Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, vol. 219, pp. 103-119, 2005.

[146] R. Birke, L.Y. Chen, E. Smirni , "Data Centers in the Cloud: A Large Scale Performance Study," in Proceedings of IEEE 5th International Conference on Cloud Computing, pp.336-343, 2012.

[147] L. Yudan, R. Nassar, C. B. Leangsuksun, N. Naksinehaboon, M. Paun, S. L. Scott, "An optimal checkpoint/restart model for a large scale

high performance computing system", in Proceedings of IEEE IPDPS, pp. 19, 2006.

[148] N. Mitra, Y. Lafon, "SOAP Version 1.2 Part 0: Primer (Second Edition),"

[149] W3C, April 2007. {http://www.w3.org/TR/soap12-part0

[150] B. Sutor, "Something old, something new: Integrating Legacy Systems", Article, 2004.

[151] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay, M. Munro, "Service-Based Software: The Future for Flexible Software", in Proceedings of the Seventh Asia-Pacific Software Engineering Conference, pp. 214-221, 2000.

[152] M. Stonebraker, "Too Much Middleware", ACM Sigmod Record, pp. 97-106, 2002.

[153] H.Ludwig, A. Keller, A. Dan, R. P. King, R. Franck "Web Service Level Agreement (WSLA) Language Specification", IBM Corporation, 2003.

[154] Leonard Richardson , Sam Ruby, "Restful web services", O'Reilly, 2007.

[155] Google. Google Cluster Data V2. Available: http://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1

[156] C. Reiss, J. Wilkes , "Google Cluster-Usage Traces: Format + Schema," Google Inc., White Paper, 2011.

[157] C.Reiss, A.Tumanov, G. R. Ganer, R. H. Katz, M.A.Kozuch "Heterogeneity and Dynamicty of Clouds at Scale: Google Trace Analysis", Proceedings of the Third ACM Symposium on Cloud Computing, 2012.

[158] C.Reiss, A.Tumanov, G. R. Ganer, R. H. Katz, M.A.Kozuch "Towards understanding heterogenous clouds at scale: Google Trace Analysis", Technical Report ISTC-CC-TR-12-101, Intel Science and Technology Center for Cloud Computing, 2012.

[159] P. P. Chen, "The Entity-Relationship Model: Toward a Unified View of Data", ACM Transactions on Database Systems, pp. 9-36, 1975.

[160] J.Dean, S.Ghemawat, "MapReduce: simplified data processing on large clusters", Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation, p.10-10, 2004.

[161] A.Thusoo, et al "Hive: a warehousing solution over a map-reduce framework", Processing of VLDB Endowment , Vol. 2, Issue 2, pp 1626-1629, 2009.

[162] Minitab, "Distribution Analysis," in Minitab Users' Guide, ed, 2011.

[163] Institute of Statistics and Mathematics. (2012, July 22). The R Project for Statistical Computing . Available: http://www.r-project.org/

[164] EasyFit "Distribution fitting", Mathwave data analysis and simulation, 2014.

[165] B. Newton and H. VanHook, "Cloud Cover Delivering on the Value of the Cloud in Public Sector IT Organizations," BMC Software, White Paper 129978, 2010.

[166] X. Rui, D. Wunsch, II, "Survey of clustering algorithms," IEEE Trans. on Neural Networks, vol. 16, pp. 645-678, 2005.

[167] P. Garraghan , et al. , "Real-Time Fault-Tolerance in Federated Cloud Environments," in 2012 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), pp. 118-123, 2012.

[168] R. Buyya, et al., "Modelling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in Proc. Intl Conf. on High Performance Computing and Simulation, pp. 1-11, 2009.

[169] R. N. Calheiros, et al., "CloudSim: A Toolkit for Modelling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," Software: Practice and Experience, pp. 23-50, 2010.

[170] S. K. Garg and R. Buyya, "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations," in Proceedings of IEEE International Conference on Utility and Cloud Computing, pp. 105-113, 2011.

[171] B. Wickremasinghe, R. N. Calheiros R, Buyya, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications," in Proc. IEEE Intl. Conf. Advanced Information Networking and Applications, pp. 446-452. 2010.

[172] O. Balci, R. G. Sargent, "Some examples of simulation model validation using hypothesis testing," in Proc. Conf. Winter Simulation - Volume 2, 1982.

[173] I. Solis Moreno, P. Garraghan, P. Townend, J.Xu, "Analysis, Modelling and Simulation of Workload Patterns in a Large-scale Utility Cloud", IEEE Transactions on Cloud Computing, 2014.

[174] I. Solis Moreno, J. Xu, "Neural Network-Based Overallocation for Improved Energy-Efficiency in Real-Time Cloud Environments," in Proceding of the IEEE International Symposium of Object/Component/Service-Oriented Real-Time Distributed Computing, pp. 119-126, 2012.

[175] I. Solis Moreno, R. Yang, J. Xu, T. Wo, "Improved Energy-Efficiency in Cloud Datacenters with Interference-Aware Virtual Machine Placement," in Proc. IEEE Int. Symp. on Autonomous Decentralized Systems ISADS, pp. 1-8, 2013.

[176] M. Kesavan, I. Ahmad, O. Kreiger, R. Soundararajan, A. Gavrilovska, K. Schwan, "Practical Compute Capacity Management for Virtualized Datacenters," IEEE Transactions on Cloud Computing, vol.1, no.1, pp. 88-100, 2013.

[177] D. Fraser, et al., "Combining p-Values: A Definitive Process," Journal of Statistical Research, vol. 44, p. 15, 2010.

[178] D. T. Mauger, G. L. Kauffman Jr, "82 - Statistical Analysis—Specific Statistical Tests: Indications for Use," Surgical Research, W. S. Wiley and W. W. Douglas, Eds., ed San Diego: Academic Press, pp. 1201-1215, 2001.

[179] J. Doyle, et al., "Stratus: Load Balancing the Cloud for Carbon Emissions Control," IEEE Transactions on Cloud Computing, vol.1, no.1, pp.116-128, 2013.

[180] D. Borcard and P. Legendre. "Exploratory data analysis," Numerical Ecology with R. Springer New York, pp. 9-30, 2011.

[181] S.Pal and P. Bhattacharyya, “Multipeak histogram analysis in region splitting: a regularization problem,” in Proc. IEEE Computers and Digital Techniques, vol 138, pp 285-288, 1991.

[182] Standard Performance Evaluation Corporation, "SPECpower_ssj2008 Results" vol. 2012, 2012. Available: http://www.spec.org/power_ssj2008/results/

[183] R. Buyya, A. Beloglazov, J. Abawajy, "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges," in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, pp.1-12, 2010.

[184] D. Tsirogiannis, S. Harizopoulos, M. A. Shah, "Analyzing the energy efficiency of a database server," in the Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 231-242, 2010.

[185] S. Rivoire, P. Ranganathan, C. Kozyrakis, "A comparison of high-level full-system power models," in Proceedings of Conference on Power aware computing and systems, pp. 3, 2008.

[186] M. M. Diouri, et al., "Energy considerations in checkpointing and fault tolerance protocols," in Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks, pp.1-6, 2012.

[187] E. Meneses, O. Sarood, and L. V. Kale, "Assessing Energy Efficiency of Fault Tolerance Protocols for HPC Systems," Proc. IEEE Intl. Symp. Computer Architecture and High Performance Computing, pp.35-42. 2012.

[188] J.N. Gray, “A census of Tandem system availability between 1985 and 1990,” IEEE Transactions on Reliability, vol.39, no. 4, pp. 409-418, 1990.

[189] D. Powell (Ed.). "Delta-4: a generic architecture for dependable distributed computing", Research Reports ESPRIT. Springer-Verlag, 1991.

[190] L. Yudan, R. Nassar, C. B. Leangsuksun, N. Naksinehaboon, M. Paun, and S. L. Scott, ``An optimal checkpoint/restart model for a large scale high performance computing system,'' in Proceedings of IEEE International Symposium on Parallel and Distributed Processing IPDPS, pp. 1-9, 2008.

[191] X. Lu, H. Wang, J. Wang, J. Xu, and D. Li, ``Internet-based virtual computing environment: Beyond the data center as a computer,''J. Future Generat. Comput. Systems, vol. 29, no. 1, pp. 309-322, 2013.