Author:
Papaioannou, Antonios D

Title:
Network Aware Resource Management in Disaggregated Data Centers

Network Aware Resource Management in Disaggregated Data Centers

By

ANTONIOS D. PAPAIOANNOU

Department of Electrical and Electronic Engineering
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Engineering.

NOVEMBER 2018

Word count: 26.000

Abstract

COMPUTING power has become a fundamental prerequisite to modern life and is available in abundance to companies, researchers, and end-users. For a given price, practically unlimited computing power is available to anyone, and cloud computing removes any geographic barriers from access to computing infrastructure. However, irrespective of how easy and flexible access to computing power has become, data centres (DCs), the physical location that hosts the computing infrastructure itself, are based on the same stale architectural principals since their creation.

A DC is an extensive set of interconnected servers. Each server consists of a fixed set of computing resources, i.e. central processing unit (CPU), random access memory (RAM), networking interfaces and storage, all of which are connected to the motherboard available only to applications that are executed on that server. Even though DCs have repeatedly proven their ability to serve advanced applications and millions of users, the concept of disaggregated resources will revolutionize the paradigm. Disaggregation provides flexibility in resource management, improving overall hardware utilization and reducing energy footprints.The disaggregation concept was recently introduced to the research community, may be a "tabula rassa" for the researchers and based on the specific requirements of it to operate, its an opportunity to reinvent some of the core features of the data centres.

In this thesis, we propose a modular disaggregated data centre network architecture and a management software specifically designed for disaggregated computing. This management software is responsible to allocate both computing and network resources. We conduct an energy study based on the proposed management software and discuss the benefits that disaggregation will bring to operators. We also analyse the factors that may affect the overall energy consumption and the resource utilization.

Finally, we showcased that resource allocation algorithms specifically designed for disaggregated data centres require 85% less power than algorithms that are currently implemented in modern data centres.

ACKNOWLEDGEMENTS

Acknowledgements might be the only truly personal part in a PhD Thesis. Over the course of these four years, there were key people that helped go through this journey that had ups and downs. The following paragraphs are dedicated to the people that helped in the realization of this thesis.

First and foremost, I would like to express my gratitude to both my supervisors: Prof. Reza Nejabati and Prof. Dimitra Simeonidou, for giving me the opportunity to work along with them and their exceptional team. Prof Reza Nejabati provided generously continuous advice, support and guidance during every step of my PhD. Prof Dimitra Simeonidou, was an inspiration to me to always push harder, also at a personal level.

Special thanks go to my dear friend and ex-colleague Tasos Vlachogiannis for his help, support and encouragement throughout my PhD.

During my PhD I had an exceptional chance to work with British Telecoms and Intel Technologies on a project that defined the route of my PhD. Mr. Nektarios Georgalas (BT) and Dr. Victor Bayon-Molino (Intel), will always have my gratitude for the guidance,input and suggestions.

My thanks are extended to all of my colleagues that shared their views, ideas or a cup of coffee with me: Mr Kyriakos Sideris, Dr George Saridis, Dr Mayur Channegowda, Dr Yanni OU, Mr Konstantinos Antoniou, Dr. Anthony Paul Haigh to name a few.

Most importantly, the real strength to finish a PhD comes from within the personal life of a student. Friends and family are making life fun! So, I want to say a big thank you to my friends in Greece and the one I met in Bristol. I cannot mention them all, but if they ever read these lines (I doubt), they will identify themselves.

Last but not least, my family. It's hard to describe my feelings in the acknowledgements of a thesis. My family : my Daddy (Dimitris), my Mummy (Tonia), my sisters (Georgia and Eleni), and my new family: my Georgia, were always by my side. A source of love, strength, support and wisdom, that made this accomplishment possible.

Yours,
Antonis

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

List of Acronyms

AWG	Array Waveguide Grating
AWGR	Array Waveguide Guide Router
CAPEX	Capital Expenditure
CPU	Central Processing Unit
DC	Data Centre
DDC	Disaggregated Data Centre
DFR	DCell Fault-tolerant Routing
DOS	Data centre Optical Switch
ECMP	Equal Cost Multi-Path
FA	Full Anycast (Resource Allocation Algorithm)
FPGA	Field Programmable Gate Array
GPU	Graphic Processing Unit
IT	Information Technology
IoT	Internet of Things
MCRVMP	Min Cut Ratio-aware VM placement problem
MDP	Markov Decision Process
MEMS	Microelectromechanical Systems
MILP	Mixed Integer Linear Programming
NP-hard	Non-deterministic Polynomial-time hardn
OCS	Optical Circuit Switching
OPEX	Operational Expenditure
OPS	Optical Packet Switching
OPST	Optical Packet Switch and Transport

OSA	Optical Switching Architecture
PM	Physical Machine
POD	Performance Optimized Data centre
PSME	Pooled System Resource Manager
QoS	Quality of Service
RAM	Random Access Memory
RSD	Rack Scale Design
RTL	Remote-to-Local (referring to memory ratio in a DDC)
SDM	Space Division Multiplexing
SDN	Software Defined Networks, Software Defined Networking
SDRAM	Synchronous Dynamic Random-Access Memory
SIC	Switch and Interface Card
SLA	Service Level Agreement
SOA	Semiconductor Optical Amplifiers
TDM	Time Division Multiplexing
ToR	Top-of-Rack switches
TSH	Target selector heuristic
TWC	Tunable Wavelength Converter
VM	Virtual Machine
VSH	VM selector heuristic
WDM	Wavelength Division Multiplexing

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Throughout the IT history computing infrastructure was evolving and becoming bigger and more complicated. Servers have always been the cornerstone of computing and consequently data centres. Nowadays, data centres consists of thousands servers and are able to perform advanced computing tasks. They host numerous heterogeneous applications from consumer, scientific and business domain, with different requirements. This chapter presents some of the trends of data centres and their challenges, and introduces the concept of disaggregation as a energy efficient, flexible and scalable alternative to traditional DCs.

1.1 Trends and needs of modern data centres

Computing, internet services and applications, in general, are an integral part of our everyday lives. Data Centres (DC) are at the core and become a valuable asset in order to support all these services. IDC [14] predicts that by 2020 90% of the IT industry will be driven by mobile technologies, Cloud Services, Big Data Analytics and Social networking applications. Cloud computing and big data require an enormous amount of computing, networking and storage resources, resulting in constant growth of DC size and complexity.

Therefore, DCs have transformed to extreme-scale computer systems in controlled environments. Each piece of equipment is interconnected by specifically designed network. In designing such an environment, there is always a trade-off between cost and business requirements [15]. The objectives of each architecture design and implementation is highly impacted by one or more of the following criteria

1. *Large Scale*: DCs are growing at an exponential rate and are consisting of hundreds of thousands of servers[16]. Microsoft alone is hosting 1 million servers in 54 regions worldwide in more than 100 DCs [17, 18]. The challenge for interconnection, cost management and resiliency is equally huge.

2. *Wide Variety of Applications*: DCs host many different applications and services with diversified traffic characteristics and requirements. There are application sensitive to latency and those that require high throughput. Additionally, measurements have illustrated that DC traffic is bursty, even under heavy load.[19–24].

3. High Energy Consumption: In 2011 energy consumption of 100 billion kWh have been attributed to DCs [25], of which 10%-12% is consumed by the network. The interesting fact is that while servers are becoming more energy proportional to their application, the network may result to 50% of the total ICT energy consumption [7].

4. Strict Service Requirement: DCs are running 24/7/365, with uptime that may reach up to 99,999%. Robustness and quick recovery from hardware failures or human error is a prerequisite.

In most cases, the DC network has either multi-tier or fat-tree topology [26]. The infrastructure is based on commodity devices and equipments. Studies show that over 80% generated from the server is directed within the rack [20]. Special focus should be given to improving both computing and networking performance. While nobody can argue about the success of this

server-centric architecture in serving thousands of users and performing complicated tasks, a disaggregated approach has been proposed [2], in order to unleash the capabilities of modern data centres.

1.2 The disaggregated data centre model

In disaggregated data centres (DDCs), servers are replaced by specialised hardware devices called 'blades'. For each resource type (i.e. RAM, CPU, graphic processing units (GPUs), etc.), there is a stand-alone blade, that provides this features.The blades are distributed across the disaggregated data centre and they are connected to each other over the network. When a task is to be executed, there is a logical aggregation of the required resources, instead of the physical one that exist in current data centres. A simplified overview of the conceptual differences between DDCs and traditional DCs is shown in Figure 1.1.

Obviously, the network plays a major role in this endeavour. On top of the traffic generated by VM-to-Vm communication, the DDC network should be fully capable of handling the extra heavy load of communication between CPUs and RAM that currently exists inside the server and is served by the chipset and the motherboard buses. Hence, the network should be able to provide high bandwidth and low latency. Network requirements will be further discussed in section 2.2.1.

Through disaggregation, DC operators can retain flexible and fine-grained control over their resources. Customization of the infrastructure for specific workloads, without vendor limitation, can improve the performance for a given cost. Since individual technologies progress at different rates, it would be possible to upgrade any resource type with cutting-edge technology releases. By separating the resources, data centres can easily accommodate the state-of-the-art technology and add specialized hardware (e.g.GPU, media encoders/decoders) independently and in a far more cost-efficient manner.

Disaggregation of resources is also a promising solution to improve resource utilization efficiency and energy consumption within DCs. Large scale DCs suffer from significant hardware resource under-utilization due to their inflexibility in allocating computing resources to virtual machines (VMs). To elaborate, consider a commodity server with 8 cores and 64 GB of RAM in conjunction with a CPU intensive application request of 8 cores and 16 GB of memory. In a normal DC, all the CPU cores would be assigned and the remaining 48 GB of RAM would be completely unutilised and therefore wasted, not only in terms of computational resource, but also in terms of physical space. Currently, this problem is mitigated with hypervisors with oversubscription capabilities. On the virtualisation layer, hypervisors allow the provisioning of VMs with higher

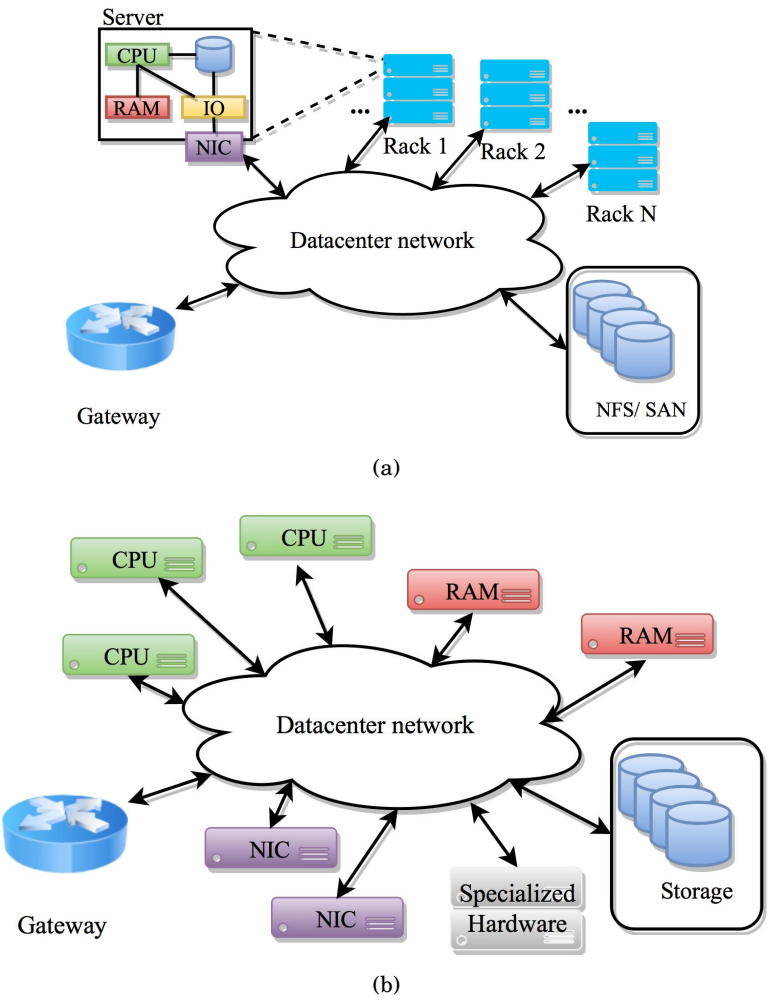


Figure 1.1: (a) The traditional data centre architectural approach, (b) The disaggregated data centre concept

aggregated nominal resource requirements than the host device, by exploiting the fact that VMs will not use all the provisioned resource altogether. So, In the case one application is idle, the remaining applications are still using the assigned resources. However, CPU utilization remains as low as 20% [27].

Presenting the previous paradigm in a DDC environment, where the resources are organised into independent blades, there will be a CPU blade with 8 cores and a RAM blade with 64GB of memory. Therefore, for the same application the 64 GB memory blade will have 48GB of RAM free to be allocated to another application. This flexibility of the architecture and the oversubscription of resource offered by hypervisors may increase the overall utilisation as it is presented in chapters 4 and 5.

Finally, the advantage of dynamically composing resources offered by disaggregated data centres will enable hosting applications with really large and extraordinary requirements, since the physical restrictions of single server does not exist and in-server memory and storage can infinitely increase.

In summary, disaggregation transforms the DC, making it more modular and flexible, providing a fine granular control over resources and enabling proper resource utilization.

1.2.1 Comparison with Distributed Computing Systems

The Disaggregated Data Centre model may be confused or compared with other distributed computing systems such as distributed memory or shared memory. However, there are some key architectural differences that the author will explain in brief, to avoid further misunderstanding in the following chapters.

A distributed computing system is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another [28]. The components interact with each other to achieve a common goal. Three significant characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components[28].

A distributed computing system may be on the hardware level or software level. For instance, Service Oriented Architecture(SOA)[1]-based systems[29] consist of independent services that are part of a larger application and communicate with each other over the network. A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently. In this case, the developer is responsible to design system with distributed functionalities and is independent of the underlying infrastructure.

On distributed memory systems, each processor has its private data. Computational tasks can be operated only on the local data of the processor and remote data should be requested via a messaging protocol from another processor. Therefore each processor is a self-contained system. On shared-memory systems, the memory may be accessed simultaneously by multiple processors allowing efficient sharing data between programs on the same system.

From the aforementioned examples, distributed systems refer to the distribution of the

[1] Not to be confused with SOA: Semiconductor Optical Amplifiers, using in optical networks.

computing tasks to different components. On the other end, the disaggregation of resources refers to the physical separation of resources and the logical connection of the resources to run a task. If that connection fails or one of the components fails, the entire application will fail, thus breaking one of the principles of distributed systems. However, distributed systems may run on disaggregated infrastructure.

1.3 Motivation and Objectives

The emergence of cloud computing and data- and compute-intensive applications (e.g. search engines, social networking, Artificial intelligence) require more powerful data centre that provide virtually unlimited resources [30].

The disaggregated data centre provides an alternative that will allow resources to scale independently within a DC, thus enabling the hosting of irregular size of applications. However, in disaggregation scenario, the required interaction between resources poses a significant challenge for the network. The network should become more efficient, provide high bandwidth , low latency and QoS guarantee, in order for this scheme to work. At the same time, it should be consistent on these features in the future, no matter the overall size of the DC.

While DCs will become more powerful eventually, the total power consumption is of high concern, mainly due its actual cost and to thermal constraints, [31, 32]. According to IDC [33, 34], over the years, the cost of IT equipment remained the same, while electricity bill has increased significantly. Thus, an ongoing challenge in the design and deployment of a new DC will be the power consumption.

Compute Infrastructure, cooling and network are the top 3 factors of energy consumption. To put that into perspective, the 100 billion kWh mentioned in section 1.1, represent 1,8% of the overall energy consumed that year [35]. In 2017, global data centre consumed 416 terawatts. That is roughly 3% of the energy consumed that year, and it is expected to rise up to 8% by 2030 [36]. Therefore, the efficient use of so big infrastructures is a challenge that many researchers tried to address.

Researchers have also focused on the network resources required to support DDCs [2], and have indicated the need for a unified resource management software. In current DCs, VMs are widely used and allow users to host applications without any knowledge about the underlying infrastructure. The management software is responsible to monitor servers and allocate the requested resources. Similarly, in a DDC, the hardware setup should be transparent to the VM [37]. A centralised network controller along with a job orchestrator as found in existing DCs [38],

will provide the necessary abstraction and consistency to operate a VM on a specific part of the DDC. The tight integration of network and compute allocation enables great flexibility to improve resource efficiency.

The author has two main objectives in the following chapters, The first one, is to identify a modular and scalable DDC architecture that would fit the needs of a DDC. The second, to use resource management, as a mean to reduce energy consumption, by efficiently utilising the infrastructure.

1.4 Thesis Overview

Disaggregation of resources as presented in this chapter is extremely promising for efficient resource management and hosting of diverse workloads, since a DDC is very versatile. Key milestone for the success of DDCs would be to actually achieve the performance indicated in literature on the hardware level on a production environment. The architecture of the DDC is an open challenge that academics are addressing since network requirements are extremely strict for CPU and RAM communication. Additionally, research studies have identified the need for a unified control of both network and compute resource in a disaggregated environment.

Author's main contribution is summarised in the following points:

- The author have studied state-of-the-art DC architectures and experimental DDC architectures and proposed a reference architecture that will be able to support a DDC scenario with commercially available equipment.
- The author investigates and analyses the resource allocation problem in a DDC and design a management software for DDC based in two tiers - scheduling and orchestration.
- The author contacts an energy based study on the benefits of disaggregation, evaluates the results of the management software and identifies factors that affect energy consumption of DDCs.

Thesis is structured in two Parts I and II, which are further split in several chapters to go into more detail to the respective subject. Part I consists of Chapter 2 and 3 and provides a background study for DC architectures and resource management. Part II and specifically Chapters 4 and 5 are the author's main contribution for the resource allocation problem and presents the first 2-tier management software for DDCs. The author is going into more detail in

the resource management of DDCs and presents the simulation and evaluation results of the management software.

Chapter 2: "Data Centres Architectures: A Background Study" is showcasing and analysing five of the most common electrical DC network architectures and three of the most promising Hybrid DC network architectures. The review is based on the complexity of the structures, their resiliency and cost efficiency. Additionally, the author analyses the first architecture for a full size DDC, which is based on FPGA SICs. Finally, the author showcases his view for DDC networking based on commercially available equipment.

Chapter 3: "Resource allocation techniques for Data centres:
A Critical Review" is the research core of this thesis. It analyses the importance of resource allocation in DCs and approaches the problem from an energy perspective. The industry-standard algorithms are presented and compared along with experimental heuristics on the energy consumption basis. Additionally, the need for network-aware allocation is emerged due to its critical effects on the application performance, and the existing algorithms are analysed. The final part of this chapter presents the efforts for resource allocation in DDCs and the views of the author.

Chapter 4: "Disaggregated Data Centre Resource Scheduling" is proposing a network-aware resource scheduling heuristic, it describes the simulation environment and evaluates the performance of the proposed scheduling heuristic. It also highlights the importance for specifically designed algorithms and investigates the effects of several DDC parameters on the utilisation.

Chapter 5: "Disaggregated Data Centre Resource Orchestration" is going a step further, and based on monitoring system it presents a software that will be able in retrospective monitor resource utilisation and re-distribute the workload to blades in order to reduce energy consumption. Finally, it investigates how a DDC is reacting to different workloads.

Chapter 6 presents a summary of the author's work and achievements.

1.5 Publications

1. A. D. Papaioannou, R. Nejabati and D. Simeonidou, "The Benefits of a Disaggregated Data Centre: A Resource Allocation Approach," 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, 2016, pp. 1-7.

CHAPTER 2

Data Centres Architectures: A Background Study

Data centres and big cloud providers have been the point of interest of research, due to the diverse nature of offered services. Lately, storage and analytics of big datasets are attracting a lot focus, which except for requirements for computing power they put a burden on the network to respond to such a heavy load. In this chapter, we will present widely-used DC architectures and proposed DC architectures, the research outcomes for DDC architectures and we will propose our architecture for DDCs. The author provides a comparison of the features of connectivity, scalability, energy consumption and technology, and identifies the key characteristics that are necessary in a DDC environment. Finally, it presents the challenges of a DDC network, related work on the topics and proposes a viable and modular DDC concept architecture.

2.1 Data Centre Network Architecture

Data centre network architecture has been a point of focus for the industry and the research community over the years due to the network's critical nature to DC performance. Therefore, many different architectures have been proposed each of them aiming to address needs such as resiliency, throughput or energy efficiency. This section will present and compare the most well-known electrical network architectures as well as hybrid network architectures.

2.1.1 Electrical Network Architecture

A modern DC usually consists of multiple racks, hosting up to 48 servers. The servers within a rack are connected with each other through Top-of-Rack switches (ToR). The switches are interconnected through the network fabric. An application usually needs the interaction of many servers in order to operate (e.g. web server and database), therefore the design and the capabilities of the network are significantly affecting the overall application performance, either in a positive or a negative way.

There are two main categories in DC network architectures: server-centric like torus[39], star[40], ring[41], DCell[16] and switch-centric such as Clos[42], VL2[15] and butterfly[43]. We are a considering and evaluating the following five architectures: two-tier, three-tier, DCell, fat-tree, and elastic-tree.

2.1.1.1 Two-Tier Architecture

The two-tier data centre architecture is shown in Figure 2.1. In the two-tier DC network, servers are arranged in racks and each rack has a Top of Rack (ToR) L2 switch. Racks are grouped into PODs (Performance Optimized Data centre), which can be up to 20ft or 40ft containers. This is the first tier. Servers within the rack are usually connected to ToR with 1Gbps links and the same stands for the connection of the ToR with the second tier. The second tier is formed by L3 switches which interconnect racks within the same POD and PODs with each other, usually with 10Gbps links. Equal Cost Multi-Path (ECMP) routing is used for load balancing [44].

The two-tier architecture provides low inter-POD latency due to few routers involved between PODs. However, it scales only up to 8000 servers [45], limited by the k number of ports in a switch/router. The links are usually oversubscribed by a ratio between 1:2.5 and 1:8 in order to reduce the number of necessary links and the cost of equipment [45].

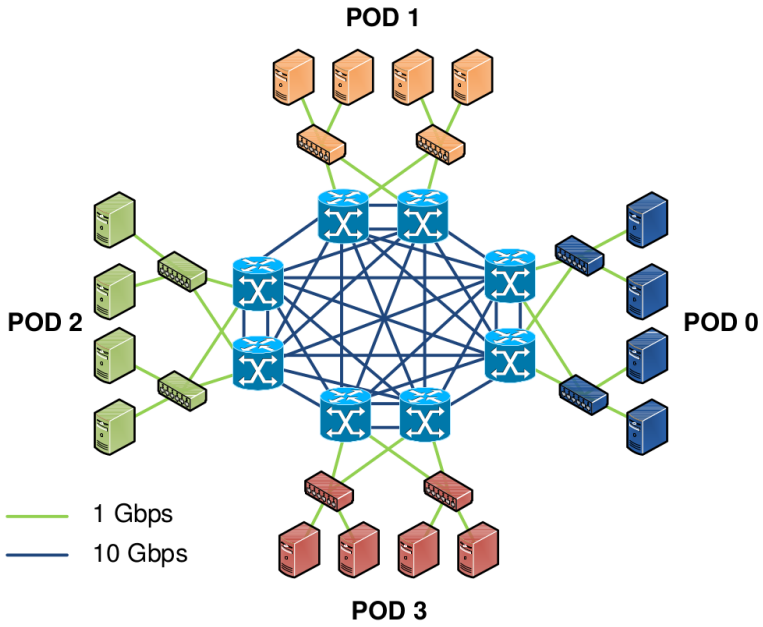


Figure 2.1: The two-tier DC network architecture [3]

2.1.1.2 Three-Tier Architecture

The three-tier is the common network architecture used in data centres[46]. Figure 2.2 illustrates the three-tier DC network architecture and shows the three different layers, namely access layer, aggregation layer, and the core layer. Similar to the two-tier architecture, servers are organised in PODs, which are connected to the L3 switches of the aggregation layer. The aggregation layer allows scaling the number of servers up to 10.000 [15] while maintaining cheaper L2 switches in the access layer.

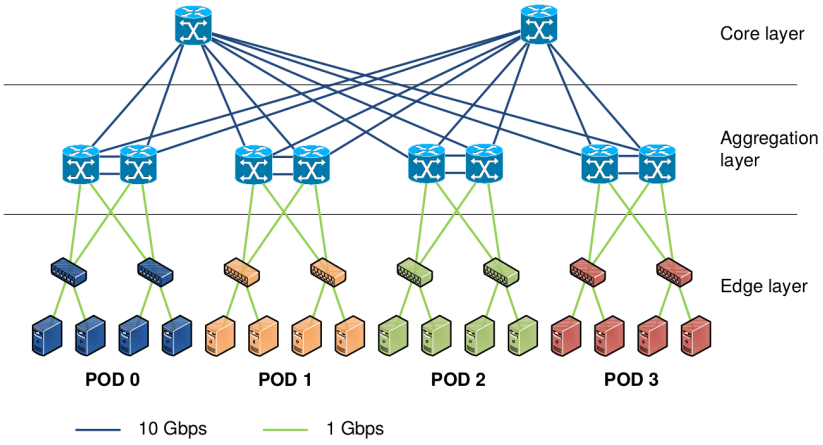


Figure 2.2: The three-tier DC architecture [3]

In the aggregation and core layers, expensive and energy-inefficient high-end equipment is used to accommodate highly oversubscribed traffic. Links may be oversubscribed in the range of 1:80 to 1:240 [15]. That leads to low end-to-end bandwidth and high network latency in the three-tier DCN architecture [47]. Three-Tier architecture is preferred in DCs where data analytics processes are running and data locality is important. Therefore, there is less traffic between racks. Although Figure 2.2 depicts only 2 core switches and links up to 10Gbps, three-tier architecture usually consists of eight core switches and links between aggregation and core layer may be up to 100Gbps [44].

Moreover, the conventional three-tier architecture also suffers from significant problems of low bisection bandwidth, poor scalability, and single failure point [3], [27]

Scalability and fault tolerance are the advantages of the three-tier architecture. However, it required routers with many ports, which are expensive, and a high number of links [9]. That increases the power consumption and the cost of ownership in general.

2.1.1.3 DCell Architecture

DCell architecture is designed focusing on scalability and resiliency against equipment failures, as described by Guo et al. [16]. DCell is an architecture of repeated server structures with servers equipped with multiple network ports. N servers are interconnected through small switches, as shown in Figure 2.3, forming the level-0 DCell. Guo [16], states that $N \leq 8$ is preferable in order to be able to use commercial 8-port switches with 1-10Gbps ports. To construct the level-1 DCell shown in Figure 2.3, $N+1$ level-0 DCells are utilised. Each DCell is connected to all the DCell with a single link. In the same fashion, level-k DCell requires $k * N + 1$ $DCell_{k-1}$.

DCell architecture is a server-centric structure based on commodity switches. It requires the least number of switches to operate, and enables the DC operator to scale up to millions of servers while being robust against a server, rack and link failures [16]. However, the complexity of cabling is the bottleneck. Additionally, it requires a special decentralised routing protocol, called DCell Fault-tolerant Routing (DFR).

2.1.1.4 Fat-Tree Architecture

Similarly to the DCell architecture, the fat-tree topology uses commodity low cost switches [45, 48]. Its main goal is to reduce oversubscription in links and remove the single point of failure present in hierarchical tiered architectures. Similar switches can be utilised in all layers, meaning that the cost of a fat-tree DC can be kept low. The architecture is not offering 1:1 oversubscription,

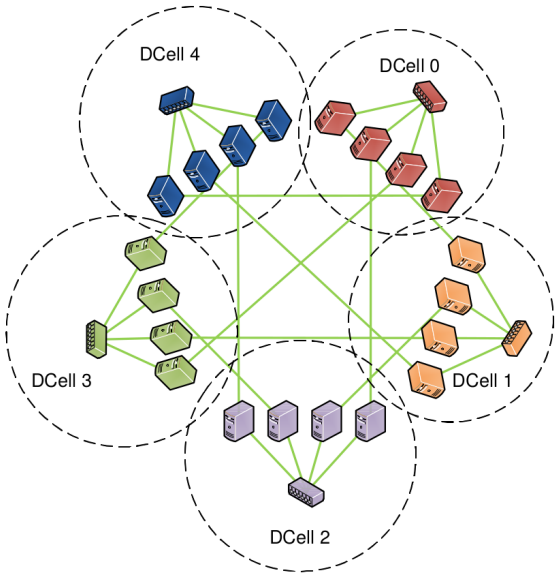


Figure 2.3: Level-1, $N = 4$ DCell network architecture [3]

but offers non-blocking path alternatives with full bandwidth, as show in the example in Figure 2.4. Additionally, Fat-tree offers full bisection bandwidth.

In a fat-tree topology with k PODs, the edge layer of each POD consists of $k/2$ switches connected to the servers, which in turn are connected to the $k/2$ aggregation layer switches [45]. The aggregation switches of the POD are connected to all the $(k/2)^2$ $k-$ port switches of the core layer. A fat-tree architecture built with k-port switch may support up to $k^3/4$ servers. As an example 24-port switches may support up to 3456 servers and 48-port switches up to 27648 servers.

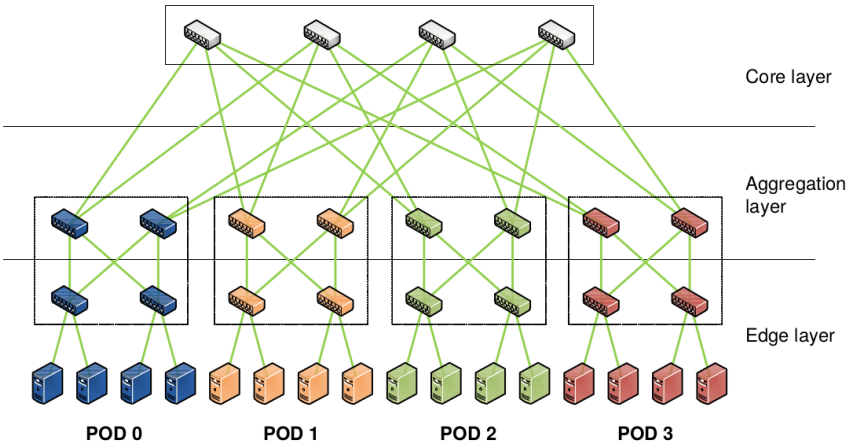


Figure 2.4: Fat-Tree architecture [3]

With networking increasing continuously, scalability of the network is of high concern. Fat-tree architecture offers a highly scalable network based on commodity switches, which is able to withstand hardware failure due to alternative paths.

2.1.1.5 Facebook Fabric

Based on the principles of fat-tree architecture in terms of redundancy, Facebook designed the fabric [4] network for their data centres and deployed in Facebook's Altoona, Iowa facility. Fabric's primary design focus is a building-wide high-performance network, easy to deploy and scale.

They started of by separating the network into pods, and that is Facebook's new unit of network. Each pod is layer3 cluster, which served by 4 fabric switches for the ToR switch uplinks. Each ToR is connected to the fabric switches through 4 x 40G uplinks, providing 160G total bandwidth capacity for a rack of 48 10G-connected servers, as shown in Fig. 2.5 [4]. To aggregate the traffic from the ToR switches, basic mid-sized switches are required. Such switches are more robust, with simple internal architecture and easy to find on the market.

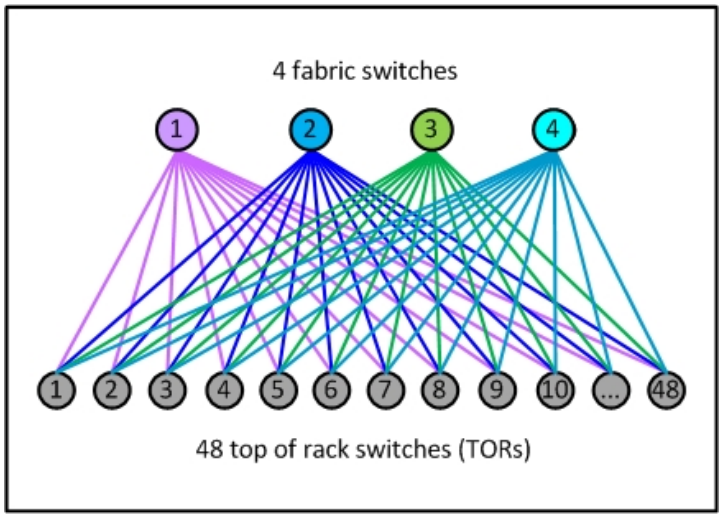


Figure 2.5: A sample pod - Facebook unit of network [4]

For inter-pod communication, engineers added a second layer of networking consisting of spine switches, which are organised in "planes". Each plane may hold up to 48 spine switches. Each fabric switch of each pod, is connected to the spine switches of the plane. So, the i_{th} fabric switch is connected to the spine switches of the i_{th} plane, forming a 2-D modular network topology able to interconnect hundreds of thousands of servers. That's how building-wide connectivity was achieved. Edge Pods are used for external connectivity and are connected the same way with the planes.

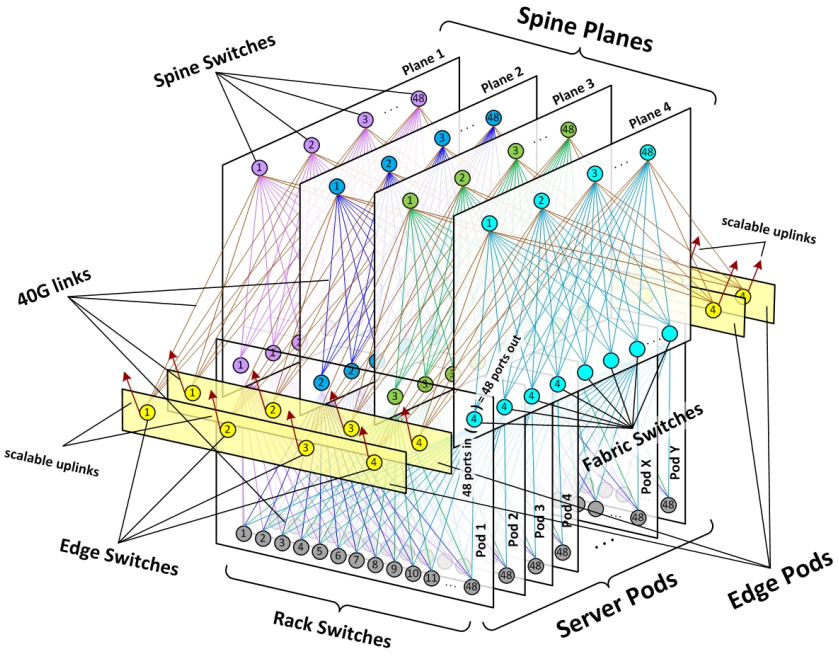


Figure 2.6: Schematic of Facebook data centre fabric network topology [4]

Fabric's modular design allows to scale capacity in any dimension. More pods provide more compute power, while more spine switches on each plane provide more inter-pod network capacity and more edge pods provide extra bandwidth outside the DC. For routing, Facebook uses standard BGP4 and developed its own BGP controller, able to override any routing path according to software automations.They call it "distributed control, centralized override" [4].

In summary, Facebook managed to provide high-capacity equal-path connectivity between any point in the network. The network is able to survive from multiple hardware failures. The whole architecture is based on the keep-it-simple principle, and although its large scale, is based on simple repetitive elements.

2.1.1.6 Elastic-Tree Architecture

The aforementioned network approaches, while being robust against hardware failures, they are not dependent on the network traffic and the switches are always powered on. That translates to high power consumption. In DCs, traffic volume fluctuates during the course of the day. That is the motive that led Heller *et al.* [49] to propose the elastic-tree architecture, which is based on the notion of dynamically shutting down equipment if it's redundant. Elastic-tree is lying on top of a fat-tree architecture. The example in Figure 2.7, showcases a scenario where 7 switches of the fat-tree infrastructure - as depicted in Figure 2.4 - have been turned off due to low traffic.

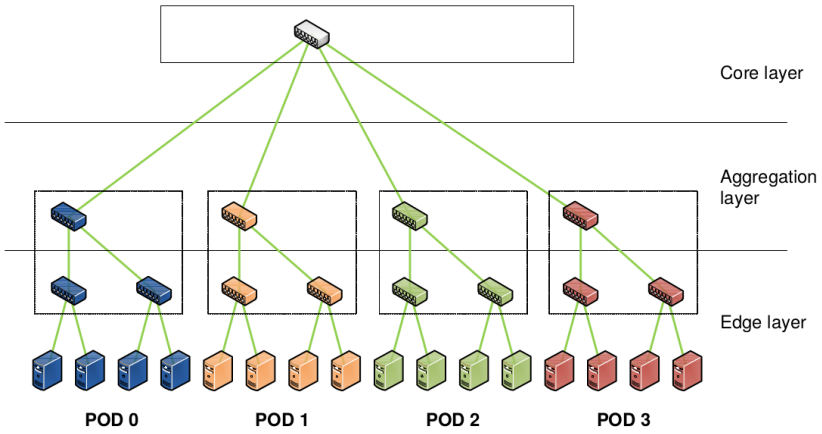


Figure 2.7: Elastic-Tree architecture [3]

Elastic-tree is more energy-efficient than the aforementioned architecture, however, there are few challenges to address. First of all, there should be a performance guarantee. During a peak load, the system should immediately react and activate the respective switches. The stands in case of hardware failure, where the switch that is idling, should instantly take over the traffic.

2.1.2 Hybrid and All-optical Network Architecture

The constantly increasing traffic in DCs generated from thousands of servers, require interconnection with high bandwidth switches. Networks based in electric packet switching consume excessive amounts of energy to serve this traffic [9]. On the other hand, optics offer high throughput with low energy consumption.

Over the last years, optics have been mainly used in the transport network. Many technologies have been developed, such as TDM (time division multiplexing) and WDM (wavelength division multiplexing), to exploit light's properties. In WDM, data are transmitted in separate multiplexed wavelengths through the fibre, resulting in the overall increase of bandwidth.

The price drop over the years for optical switching technology, have led to the adoption of optics in DC network architectures. Currently, it is mainly used for point-to-point connections, to provide high bandwidth on specific links. However, the combination of electrical and optical switching is the first step towards full optical networks. Electrical switching is used in the edge of the network, where flexibility is required, while optics are used in the core to aggregate traffic. The main drawback on this scenario, is the power hungry electrical-to-optical and optical-to-electrical transformations.

The future of DC networking, are the all optical networks, in which switching will be done in the optical domain. DC operators will use light technology to its full potential and have high bandwidth, low latency and significantly reduced energy consumption [50–52]. According to a report, savings can be up to 75% [53].

2.1.2.1 C-through

C-through is a tree topology with electrical ToR switches that are connected both to an electrical packet based network (i.e. Ethernet) and an optical circuit-based network, as shown in Figure 2.8. A centralised manager is continuously monitoring the traffic and decides upon the routing of the traffic. To route the traffic through the optical network, the optical circuit should be configured.

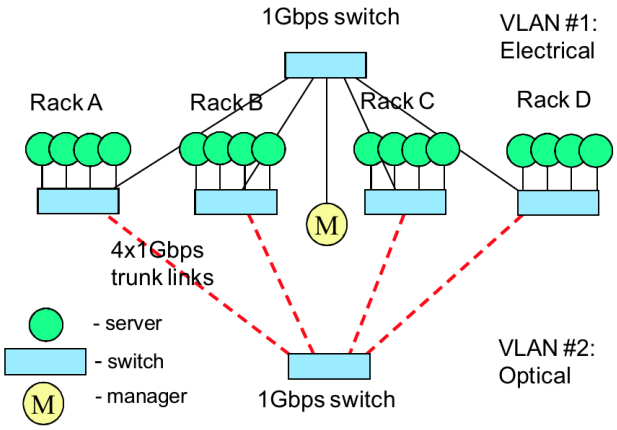


Figure 2.8: C-through architecture [5]

ToR switches are using VLAN-based routing. There is one VLAN for the electrical packet network and one for the optical circuit network. After the configuration of the OCS, the traffic manager notifies the ToR switch to route the traffic to the second VLAN.

2.1.2.2 Helios

Helios [6] is a 2-level tree consisted of pod switches and core switches, with multiple paths for inter-pod communication, as shown in Figure 2.9. Core switches are both electrical and optical. Helios is very similar with C-through but wavelength multiplexing is utilised on the optical links. Electrical switches are used for the unpredictable communication between all the pods, while optical switches are used for high bandwidth long lived communication between specific pods. The WDM links allow full bisection bandwidth.

Helios control loop is continuously monitoring the topology. By estimating the bandwidth demand among traffic flows [54], routes the traffic accordingly. In comparison with c-Through,

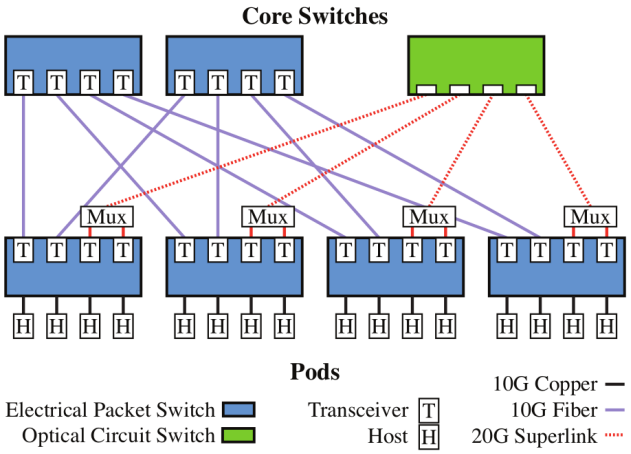


Figure 2.9: Helios architecture [6]

Helios is able to dynamically route the traffic in both optical and electrical switches, based on the computed configuration. Helios control loop requires up to 265ms to recalculate the route and reconfigure the switches if necessary [6]. Thus, the benefits of this topology are significant only in cases that traffic flows last more that few seconds, in order to compensate the reconfiguration overhead [9].

2.1.2.3 Optical Switching Architecture (OSA)

Xia et al. [55] propose Optical Switching Architecture (OSA), an all optical core network among ToR switches. As illustrated in Figure 2.10, each ToR switch is equipped with double transceivers and uses separate fibres to send and receive data. The multiplexers (Mux) combine the optical signals from many fibres to a single fibre and direct the signal to the 1x4 WSS (Wavelength Selective Switch). WSS distributes the traffic to the circulators according to the wavelength, which in turn direct the signal to the bipartite MEMS (Microelectromechanical systems) [56] switch to change the output port or the coupler. Finally, the DeMux separates the wavelengths to different fibres. WSS and MEMS can be easily configured to accommodate the traffic demand.

Authors claim that OSA offers high bisection bandwidth (60% - 100% non-blocking). Their switching time is 14 ms and 9 ms respectively, however, their impact on latency sensitive traffic requires further investigation. Similarly to Helios, OSA architecture works bet with stable traffic flows and may face performance degradation on highly dynamic traffic patterns due to the reconfiguration time required for the optical devices.

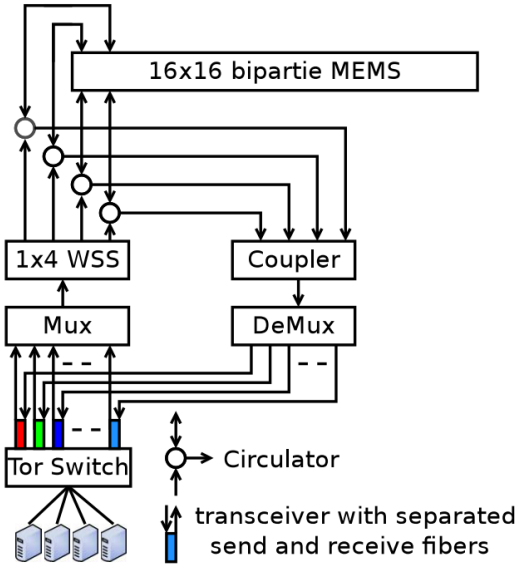


Figure 2.10: Optical switching architecture [7]

2.1.2.4 Data centre Optical Switch (DOS): A scalable optical switch

Ye *et al.* presented the DOS: scalable Data centre Optical Switch [8], which is based on AWGR (Arrayed Wavelength Guide Router) [57] and allows multiplexing in the output, thus enabling multiple inputs to connect to the same output simultaneously. The optical fabric consists of TWCs (tunable wavelength converters), the AWGR and the SDRAM (Synchronous Dynamic Random-Access Memory) buffer.

DOS is packet based. The signal goes in the label extractor(LE), which strips off the label and forwards the payload to the TWC. The control plane receives the label and based on the source and destination configures the transmitting wavelength of the TWC, hence the output port in the AWGR. In that way, any-to-any communication is achieved. If the number of clients that want to transmit to a specific port is greater than the receivers in the output, the surplus of packets is routed to the SDRAM buffer and stored there until the controller transmits a signal to release them towards the original output.

Scalability of DOS depends on the number of ports in the AWGR and the amount of wavelengths that TWCs support. Additionally, AWGRs and TWCs are quite expensive compared to commodity optical transceivers and the congestion management through the buffer is energy-intensive. DOS is not using buffers on switches, thus latency is independent to input and remain low even at high loads. Switching delay in TWS is just few nanoseconds compared to few microseconds in MEMS switches, making DOS suitable for bursty traffic.

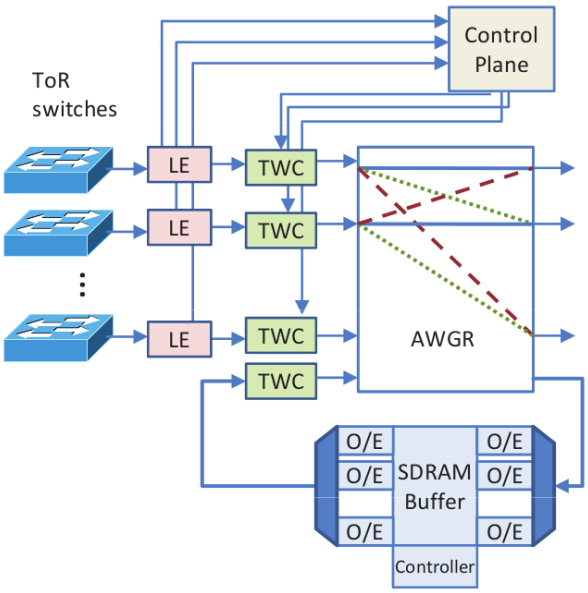


Figure 2.11: Data centre Optical Switch (DOS) [8]

2.1.2.5 Petabit Optical Switch

Based on the DOS, Chao *et al.* [58, 59], presented Petabit. Differences lies on the fact that Petabit is not using buffers inside the switch fabric. In order to avoid congestion in the output of the AWGR, Petabit is buffering traffic in line cards using a smart schedulers, thus avoiding power inefficient optoelectronic conversions.

2.1.2.6 Proteus

Proteus architecture provides an all optical architecture, which at its core has a MEMS based optical circuit switch [60, 61]. Each ToR has N optical transceivers for each of the N wavelengths that the network supports. A multiplexer combines the N wavelengths and routes them to the WSS. The WSS multiplexes the optical signals and forms k groups of wavelengths and forwards them to the respective port of the optical switch, Fig. 2.12. The opposite procedure is followed when receiving a signal. So there is a full bandwidth point-to-point connection between ToRs.

Proteus is able to provide flexible bandwidth for inter-ToR communication by activating more transceivers. However, due to limitations in the number of ports of the MEMS switches, there are ToRs that are connected directly, and ToRs that require extra hops to reach another ToR. Therefore, it is quite a challenge to identify the optimal configuration for the MEMS switch, to ensure full graph connectivity. For the same reason, Proteus is not flexible to dynamic traffic changes.

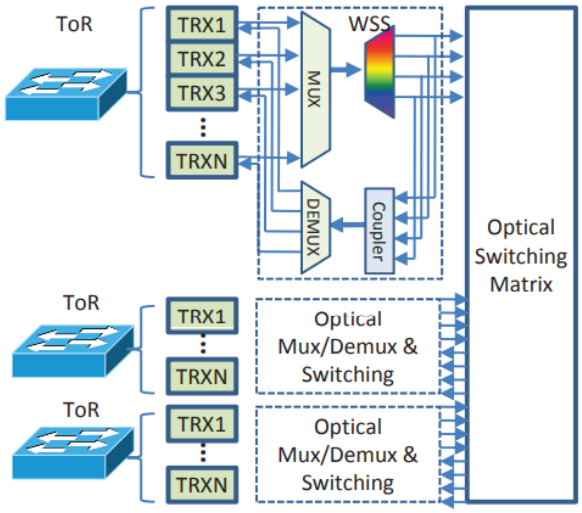


Figure 2.12: The Proteus Architecture [9]

2.1.2.7 Osmosis

Osmosis is an optical Broadcast-and-Select network, which features low-latency between ToRs and is based in wavelength and space-division multiplexing [62, 63]. In Osmosis, there are 2 distinct steps. In the first step, input signals are multiplexed in a WDM line and are broadcast to each one of the modules of the second step. Semiconductor optical amplifiers (SOA) are used as filters and select the wavelength that will guided to the output, Fig. 2.13.

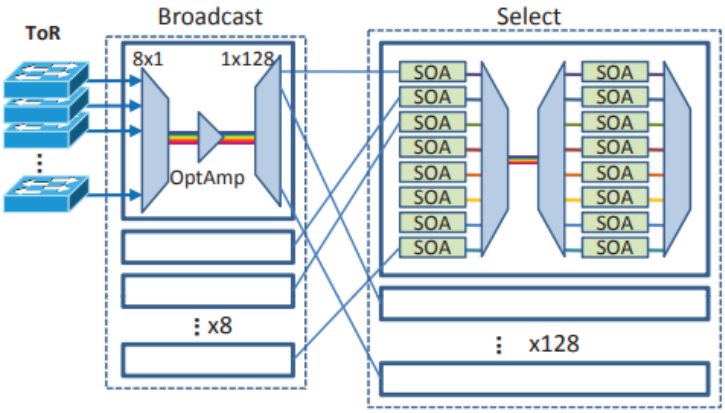


Figure 2.13: The Osmosis Architecture [9]

In order to avoid the use of TWCs, Osmosis is using a demux-SOA-select-mux scheme to pick the correct output port. This requires a centralised arbiter that works synchronously with fixed-length optical packets. In that way, Osmosis achieves high-bandwidth packet-level switching. SOAs can be reconfigured within nanoseconds.

Limited by the number of wavelengths that the Select step can handle, multiple switches can deployed in a tree topology, in order to scale Osmosis. Finally, the extensive use of SOAs will significantly the overall power consumption of the DC.

2.1.2.8 IRIS

Iris is an all optical core network that is based on WDM and AWG (Array Waveguide Grating) technologies, to provide a dynamically non-blocking communication [64]. IRIS has 3 distinct stages. In IRIS, N ToR switches connect to one of the ports using N wavelengths. In the first stage, an array of wavelength switches (WS), routes the wavelengths to the output, based on SOA converters that are used as filters. In the second stage, there are optical time buffers which are controlling the congestion in the third stage, without the need for SDRAM buffers, like in DOS. Finally, the signal is converter to the wavelength that corresponds to the AWG output port, Fig. 2.14.

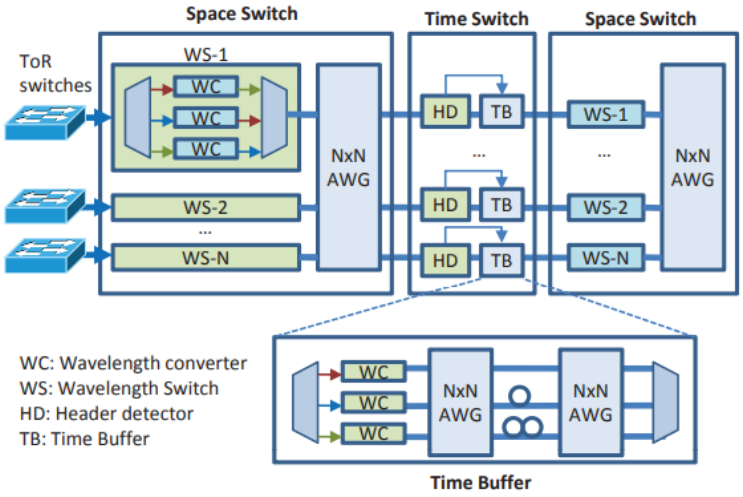


Figure 2.14: The IRIS project [9]

The controller is local to each time buffer and reads the optical packet headers to control the time switch. Time switches consists of an array of wavelength converters and two AWGs connected with different length of optical lines. The optical link length, depends on the delay that should be added to the signal. The WC converts the signal to the required wavelength. After the second AWG, the signals are multiplexed and guided in the last stage. IRIS enables high throughput in the entire network, since 40 Gb/s data packets and 80x80 AWGs provide $80x80x40Gbps = 256Tbps$ throughput.

2.1.2.9 Commercial optical interconnects

Most of the aforementioned hybrid network architectures, are proposed by the research community; either academia or research and development centres. However there are commercially available products.

I. *Polatis*

Polatis Inc. developed an optical switch that is the centre of a scheme that can be reconfigured according to traffic demand.The switch is based on piezo-electric optical circuit switching and beam steering technology. Polatis main advantage is that the setup can support any data rate and is consuming only 45W of power [65]. However, its MEMS technology requires up to 20ms switching time, making it unsuitable for bursty traffic.

II. *Intune*

Based on their fast tunable transmitters, Intune developed the Optical Packet Switch and Transport (OPST) Technology [10]. Every node is connected to the ring through a fast tunable laser and receiver. Each receiver is assigned a specific wavelength, and nodes communicate with each by configuring the transmitters frequency to the respective node's receiver wavelength, as shown Fig. 2.15. The Intune network may hold up to 80 channels per ring and 16 nodes can be attached to the ring and transmit at 80Gbps. Although Intune pushes this technology to transport networks, it can also replace core network in DCs, especially due to its low consumption compared to equivalent electrical commodity switches.

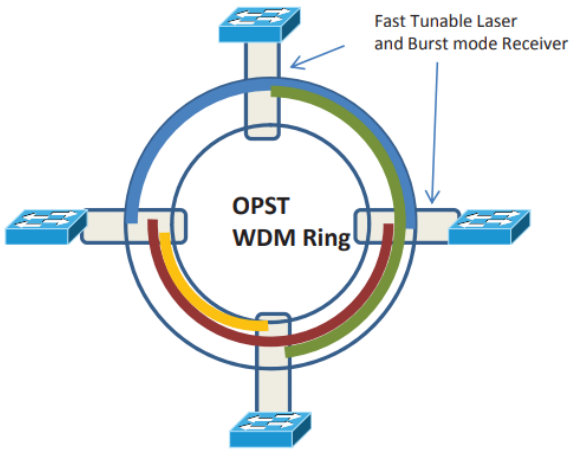


Figure 2.15: Intune's OPST Architecture [10]

2.1.3 Discussion

Over last 15 years the networking architectures in networking have greatly evolved, mainly due to the appearance of bandwidth-intensive workloads. Modern DCs based on Electrical packet switching managed to provide flexibility in routing and all-to-all communication, scalability and fault tolerance.

One of the main factors that limit the ability to scale in tree topologies is the number of ports in switches, which also leads to link oversubscription. Oversubscription is the ratio between the server's bandwidth and the available bandwidth on the link. So, in the aggregation and core layers of tree topologies, the number of servers that share those links is larger and the link capacity may not be enough. Congestion created on oversubscribed links, will overload switch buffers, and in extreme cases result in dropped packets. Another problem with oversubscribed links, is that packets needs to stay in the buffer. Even if the packet is not dropped, there would be additional latency along the path.

An excellent example of a tree topology that handles scalability challenges is Facebook Fabric. Although cabling may be complex, Facebook invented a way to scale up to hundreds of thousand of servers based on low-cost commodity switches, while based on BGP protocol for routing. On the other hand, DCell is also able to scale to great extent, but the proprietary routing protocol and the complex cabling make it less appealing.

Hybrid networks offers the opportunity of incorporating optics in existing DCs and incrementally increasing their capacity. ToR switches equipped with optical modules will increase the bandwidth and reduce latency on large traffic flows, while the existing network will serve burst all-to-all communication. However, limitations in the number of ports in circuit switches limit their ability to scale, thus making them not necessarily a viable solution for future DCs. Polatis and Intune are such cases, as well.

Optical interconnects, are split into two major group depending on circuit or packet switching. Circuit switching is usually implement with MEMS, that have high reconfiguration time. That makes suitable to DCs that have stable high bandwidth traffic flows. A main advantage of circuit switches is their ability to support any data rate, since they are redirecting the beam .

On the other hand, optical packet switching provides all-to-all communication. They are usually based on AWG and TWC technologies, and by selecting the correct wavelength at source, the beam will transmit to the correct output port. DOS, Petabit and the IRIS architectures are such examples. Therefore, their switching capacity is limited by the available technology on the data rate or TWC. Similarly, Osmosis is limited by the data rate of the SOA technology.

An interesting intermediate is this two approaches, is the Proteus network, that provides all-to-all communication, although it is based on circuit switching. The is achieved by doing multiple hops through directly connected switches.

In terms of scalability, optical switches with high port density, are extremely expensive. As an alternative, many of the schemes could be used to form a cluster of racks, and then another layer of the same scheme could be added on top, to form the core layer [9].

2.2 Disaggregated data centre architecture and benefits

2.2.1 DDC network challenges

Disaggregated Data Centres are a leap in the computing technology as we know it. The network is the spine of a DDC, and ultimately the overall performance of a DDC is dependent on it.The amount of communication that is being handled, requires appropriate network design, That is the greatest challenge for the system designer, since the network architecture and technologies should evolve and improve both in terms of capacity and latency. TABLE 2.1, summarizes the minimum network requirements for a functional DDC [1, 2].

Due to the combined requirements for high bandwidth and low latencies, optical networking technologies are the primary candidates for implementation. In terms of capacity, traffic can be accommodated by existing optical technologies, however latency is the critical parameter. Normal fibres have an approximate delay of 5ns per meter. Recently, hollow core fibres have been created and they have 30% less latency than conventional fibres [66]. Additionally, faster interconnects and switches with low port-to-port latency may be implemented. Lower delay allows DDC operators to have a bigger shared pool of resources and scale their DDC implementation, and possibly enjoy more of the benefits that a DDC has to offer.

To further reduce latency, new protocols should also be developed. In [67], Saridis et al. suggested an FPGA based switch and interface card that can be plugged directly to the server, to reduce blade to blade latency. Additionally, in [68], authors propose the implementation of a 2D mesh or a 3D torus network architecture for rack-scale DDCs, where each blade will serve as a switch module. Thus, each blade will directly access resources in other blades, without delays imposed by switches. Authors argue that they can achieve higher throughput due to path diversity and lower queueing delays due to better load balancing.

The current and forthcoming networking technology is expected to be unable to carry out CPU-to-CPU and pure CPU-to-RAM communication because of their extremely high bandwidth

Communication	Latency	Bandwidth	Link Length
CPU - CPU	10 ns	200 - 320 Gb/s/CPU	0.1 - 1 m
CPU - Memory	10 - 50 ns	300 - 800 Gb/s/CPU	1 - 5 m
CPU - Disk	1 - 10 us	5 - 128 Gb/s/device	5 m - 1 km

Table 2.1: Resource communication requirements for disaggregated data centres [1, 2]

and low latency requirements. To address this issue, it has been proposed to maintain a small amount of RAM in each CPU blade, that will serve as a cache for frequent memory requests [2]. This RAM, will be available only to the co-located CPU. In that way, reusable data will be in close proximity to the CPU and will not be fetched from the remote memory blade.

2.2.2 Towards DDCs

The creation of a DDC has attracted a lot of interest from both industry and academia alike, leading also to commercial products, that offer several levels of disaggregation. Academic researchers have also contributed towards this direction with examples such as soNUMA [69] and Firebox [70] that allows the operating system to access the remote memory. In [71], page swapping techniques were proposed.

SeaMicro has developed a single server architecture to serve computationally heavy tasks. All the non CPU and RAM resources were removed from the motherboard, resulting in a small compute unit. FPGAs are utilised to interconnect all the compute units [72].Based on the same concept, HP has implemented its own solution with HP Moonshot [73]. HP created a software-defined server that carries 45 resource cartridges and user can configure it to fit their needs.

2.2.2.1 Rack Scale Design (Intel)

The most prominent solution towards a completely disaggregated DC, is Intel's Rack Scale Design (RSD) [74]. Compared to the approaches of SeaMicro and HP, Intel has extended this idea to work at rack scale. RSD offers all the benefits described in the Introduction, including:

- Faster and easier scale out
- Higher Efficiency due to better resource utilisation: a user may plug in the RSD only what is required.
- Lower Capex, due to independent resource upgrade cycle
- Flexibility in optimizing application performance, via custom configuration.

RSD, is organised in Pods. Each Pod is a physical and logical collection of racks with a shared management application, called Pod Manager. Each rack consists of a set of compute, memory, storage or networking modules, that are shared for all application and can be modified. Resource that are part of the rack, have at least 2 network connections. The first one is a high-bandwidth data connection and the second one, an Ethernet link to the management network. Pods are interconnected via Top-of-Rack (ToR) switches.

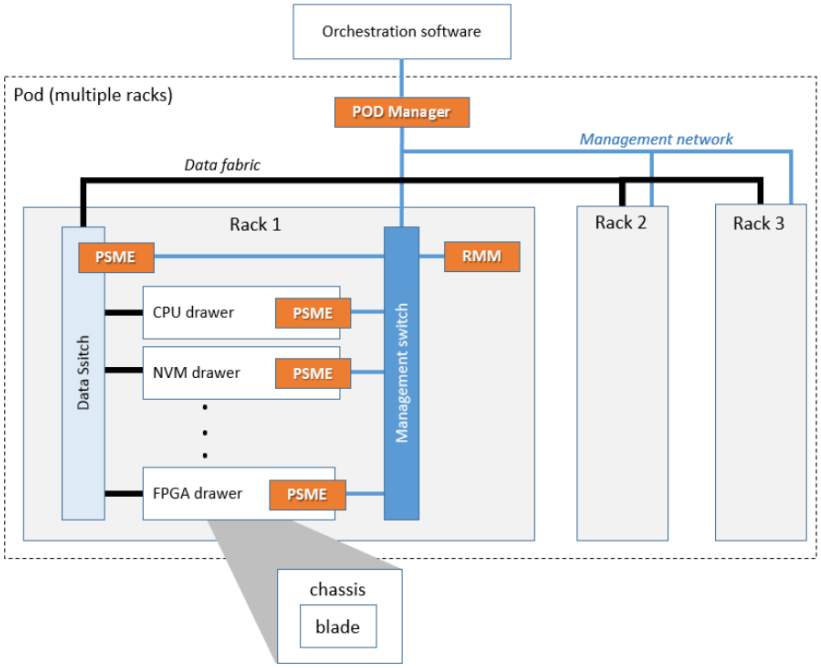


Figure 2.16: Intel RSD Architecture

The Pod manager is controlling the resources through and open API, compatible with DMTF's Redfish RESTful framework [75]. On the hardware level, each blade should have a software component called the Pooled System Resource Manager (PSME). The Pod Manager communicates with the orchestrator in order to send information about the available resources on the infrastructure and receive requests for applications.

The composition of the system is achieved through the data fabric. Pods are linked together using whatever network topology is determined the data centre administrators [76].

2.2.2.2 All-Optical Programmable DDC Network

Yan et al. [11], proposed an FPGA-based switch and interface card (SIC) that attaches to the server, that supports optical packet switching (OPS)/optical circuit switching (OCS) or time division multiplexing (TDM)/wavelength division multiplexing (WDM)traffic on demand. This design

pushes the switching functionality to the server, resulting to direct blade-to-blade communication and extremely low buffer-to-buffer latency. Based on that functionality, the authors were able to design an all-optical programmable disaggregated data centre network architecture.

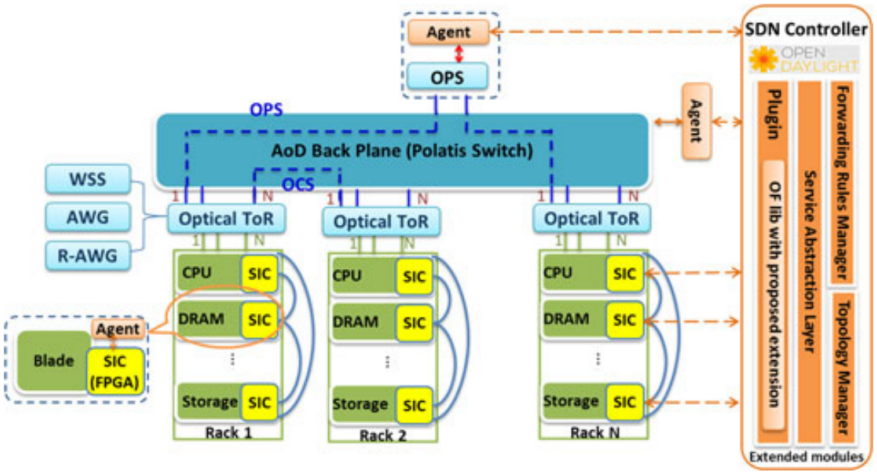


Figure 2.17: All Optical Disaggregated DC architecture [11]

By utilising the SIC, every resource blade of the rack is connected to the optical ToR. The ToR may be a WSS [77] or a AWGR, through the direct optical link of the SIC. An AoD node [78] interconnects all the input and output ports of the optical ToRs. OCS and OPS are supported for both inbound and outbound traffic, providing flexibility and all-to-all communication. The infrastructure is SDN-enabled and managed by OpenDaylight Controller, that is able in real-time to configure the SICs and consequently the routing of the traffic.

2.2.3 Hybrid architecture for DDC

Based on the concept and architectural principals of c-Through and Intel's RSD, we are proposing a 2-layered network architecture which relies upon both optical and electrical network switching equipment, and satisfies the latency requirements of a DDC. Figure 2.18 depicts a high-level view of this architecture, which is separated into two independent networks. There is an all optical network (lower part of the image), which from now on we be called the "*fast backplane*". The second network, which can be either electrical, hybrid or a second all optical, the upper part of Figure 2.18, will be called the "*generic backplane*".

The reason for the existence of two separate independent backplanes is that there are two traffic patterns each with their own distinct requirement. The first one represents all the communication that occurs inside the servers in a traditional DC, i.e. the communication between CPU and RAM, GPU, network cards, etc. As shown in Table 2.1, this communication

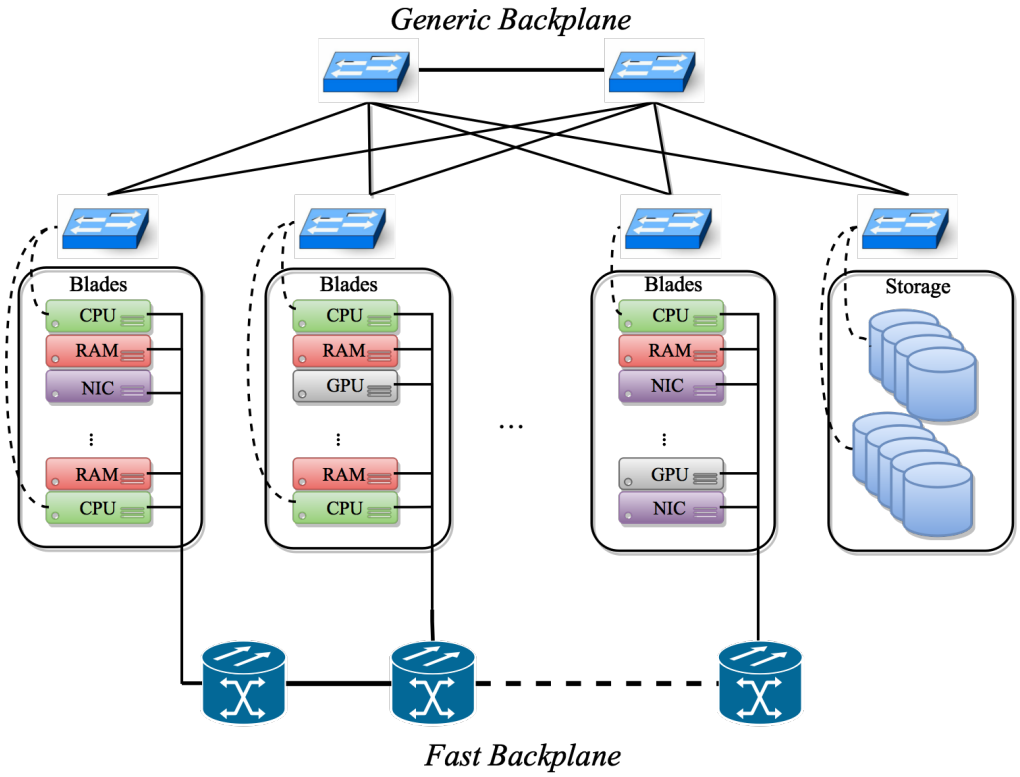


Figure 2.18: The 2-layered network architecture for DDC

is characterized by high bandwidth and very low latency. Studies have shown that a small variance of latency may cause serious performance degradation [2]. Optics can provide low and deterministic latency, and therefore a steady performance. To achieve 10-50 ns latency for CPU-to-RAM traffic, the network architecture must be simple, with as few links as possible.

The optical links should be as physically short as possible and the port-to-port latency in switches must be low. Normal fibres have an approximate latency of 5 ns/meter. Recently, hollow-core fibres have achieved 30% less latency than conventional fibres. This means that the fabric can increase from 5 m up to 9 m, depending on the network design whilst maintaining equivalent latencies. As measured in the lab, optical circuit switches (OCS) and wavelength selective switches (WSS) can provide low port-to-port switching delays: 5 ns and 20 ns respectively. The same measurements for Ethernet and INFINIBAND top-of-the-rack (ToR) switches are 250 ns and 100 ns [67]. Along with buffering in electronic switches and routers in DCs, classic network architectures are inappropriate for such latency-sensitive communications. The communication between the CPU and the disk doesn't fall into this category since the latency requirements are quite loose in comparison, and the required bandwidth capacity can be achieved by the generic

backplane. The discovery of available resources that satisfy the Table 2.1 criteria relies upon the management software as it was briefly discussed in Chapter 1 and will be further analysed in Chapter 3, 4 and 5.

The second traffic pattern represents the communication that currently exists within a DC. The generic backplane would provide the necessary flexibility for VM-to-VM and storage communication. This traffic might have high bandwidth requirements, but is not latency sensitive. Therefore, it is not affected from the buffering in switches due to the varying buffer length. It is up to the administrator of the DDC to decide the network architecture of the generic backplane, based on the particular traffic requirements of the application hosted.

Such a 2-layered architecture allows simpler integration of the disaggregated technology with existing DCs. It is possible to create clusters of resource blades that have their own separate fast backplane. The cluster can seamlessly connect to the rest of the DC through the generic backplane, which, in this case, is part of the existing operator network fabric and serves outward communication.

The size of the cluster relies upon the physical installation of the servers, the length of the links and the port-to-port switching delay of the switches, in order to achieve the necessary 10-50ns latency. Link failures on the fast-backplane may cause severe performance degradation, so it is advisable to deploy redundant links.

2.3 Summary

This chapter provided a brief overview of existing DC network architectures, as well as some experimental. Networking is an essential part of the DC, as it can heavily affect application performance or even break an application down in case of failure. Therefore resiliency, scalability, high-bandwidth and low-latency are required characteristics of modern network architecture.

Traditional DCs have been around for many years, Electrical or Hybrid architectures have been determined by the market and we presented the most popular along with their key characteristics. Each of the architectures has pros and cons and the decision is based on the use case. The DC operator should find the trade-off between cost and performance. The most interesting approach is Elastic-Tree architecture. If shutting down and activating a networking device may be done effectively, it will revolutionise the network architecture and even trigger a new way to design networks.

On the other hand in DDC network architecture is a new field that requires further investiga-

tion. As an obvious choice is the use of optical component for blade interconnection due to the strict requirements. Optical switching equipment is progressing and achieves faster switching time and hollow-core fibres allow less transmission time. The advancement of these technologies, may result to more flexible network architectures or larger scale of DDC cluster. The research community should capture this opportunity.

CHAPTER 3

Resource allocation techniques for Data centres: A Critical Review

Resource allocation in DCs has been a point of interest for the research community. Efficient resource allocation can be one of the main differentiator between DCs, with a huge impact on the initial and operating cost. Usually the point of focus is the utilisation of servers and the energy consumption. However, most of them only focus on the compute resources and completely ignore the network. To address the problem of resource allocation in a DDC effectively, the author has studied resource management in traditional DCs and mostly focused on network-aware management algorithms, due to DDC requirements. This chapter reviews techniques that aim in improving the overall energy consumption in a traditional DC, and then focuses on network-aware resource allocation algorithms either on scheduling or on orchestration level.

3.1 Importance of proper resource allocation

Data centres are facilities that may host thousands of servers with kilometres of network cables running through it, to interconnect all the devices. DC may be used to centralise an organization's IT operations or provide a centralised place to co-host an application like in the Cloud. According to a report by Lawrence Berkeley National Laboratory, 70 billion kWh were consumed by DCs only in United States in 2016 [79]. This corresponds to 1.8% of all the energy consumed in the United States that year.

In a modern DC, the management software is the key for the proper operation of the infrastructure. It holds information about the DC assets, provides monitoring and management capabilities for data and media storage, server images, authorisation of the users, billing and many more. The functionality of interest in this chapter is the allocation of resources to an incoming VM request, and the effects on server utilisation and energy consumption, as well as the life-cycle management of that VM.

Utilisation of resources and energy consumption are of high concern for DC operators. Cost of equipment accounts for 65% and power accounts for the 18% of the monthly costs of a DC [32]-based on 3year server amortization. Therefore, all servers should ideally be fully utilised. Since, that it is not possible due to fluctuations in demand, the energy consumption should as low as possible.

Energy consumption is related to the utilisation of a resource. To put that into perspective, Eq. 3.1 describes the energy model for compute and network devices, in its simplest form [12, 80]. Both type of resources have a similar behaviour regarding energy consumption.

$$\begin{aligned} & E = a * max\_power + (1-a) * utilisation * max\_power \\ & or \\ & E = (a + (1-a) * utilisation) * max\_power \end{aligned} \tag{3.1}$$

where max_power is the maximum power consumed when a device is fully utilised; α is the fraction of max_power the device consumed in idle state and utilisation of the device is expressed as a number between 0 and 1. Thus, the total energy consumption of the devices (compute, network and storage) in a DC is calculated as follows:

$$E_{DC} = \sum_{i=1}^{N} E_i \tag{3.2}$$

where,

$$E_i = \begin{cases} (a_i + (1 - a_i) * utilisation_i) * max\_power_i & \text{when the } i^{th} \text{ device is powered on} \\ 0 & \text{when the } i^{th} \text{ device is powered off} \end{cases} \quad (3.3)$$

As seen in 3.1,hardware, namely servers and networking equipment, consumes a lot of energy in idle state. The idle state energy consumption is usually around 75% for servers and 85-90% for electrical switches of the maximum energy that each device may consume. So, it is advisable, if possible, to turn the electronic circuits and systems off when not being utilised. Many of the following algorithms and approaches are based on this assumption.

It can be inferred from Eq. 3.2, that energy consumption can be minimised by fully utilizing a piece of equipment and using as less devices as possible. However continuously increasing the utilisation may cause performance problems. In a case were all of the VMs suddenly request access to the whole of their allocated resources, the host would not have enough resources to serve the application's requests. That will result to a Service Level Agreement (SLA) violation, harming the operator financially.

In DC that hosts thousands of servers, proper resource management may result in economic benefits for the DC owner, in many ways. For instance, improved utilisation of compute resources, will lead to better forecast for server purchases, or extra capacity to host applications. In a scenario of variable load, the DC will be able to adapt and reduce its energy requirements, and on high load its more like to provide the agreed SLA. The resource management algorithm should consider all the economic and performance factors and strike a balance for the benefit of the DC operator and the end-user.

In order to identify the key elements of a management software that are responsible for resource allocation, the authors have studied several cloud management framework in the market like Apache CloudStack [81], OpenStack [82], CloudHealth [83] and Scalr [84]. Each of them is using their own software architecture to manage the available resources. In Openstack software architecture [82], that we use as a reference later in the chapter and in Chapters 4 and 5, resource allocation is managed by two individual components. The scheduler [85] and the orchestrator [86].

- Scheduler is the service that determines how to dispatch VM requests. When a new VM request is arriving, it is placed in a queue. The scheduler, picks up one-by-one the requests

from the queue scans the entire DC for available resources, and then based on a predefined policy, it allocates the VM to a host with sufficient resources, namely CPU, RAM, storage.

- Orchestrator is the service that interfaces with the user in order to automate the creation and control of cloud components like networks, instances, storage devices and many more. From the operators perspective it is the tool to enforce policies on resource allocation holistically, based on monitoring results about performance and the load of the servers. Thus, it may trigger a VM's migration to follow the operator's policies.

As presented in the following sections there is research focusing on either or both of those components, aiming to reduce the overall energy consumption or improve the performance of the computing and networking resources in a DC.

3.1.1 The challenge of heterogeneity

For the resource allocation problem, it is extremely difficult to achieve the optimal solution in a real data centre, because of the heterogeneous nature of DC. Many authors describe this problem even in simplified-homogeneous versions as NP-Hard [80, 87–89], and in the end they study a subset of the problem that fits their needs and approach the solution with heuristics.

Heterogeneity features are present in many aspects of a DC. Three of them are presented:

A. **Resource heterogeneity:** Hardware in a DC is heterogeneous by design [90–92], since there are several server configuration. For example, the are servers with different types of CPU, and diverse amount of memory and disk. Additionally, there are also different types of networking equipment as it is discussed in Chapter 2. The heterogeneity is also amplified by required maintenance or upgrade in the DC. Broken or outdated equipment is replaced with newer, which and outperform the previous ones. In that way the DC operator is able to host the ever-increasing request for computational resources [93–96]. Moreover, the upgraded equipment is usually more energy efficient, presenting a opportunity to reduce the operational expenditure (OPEX) of a DC

B. **Application type heterogeneity:** A DC is able to run many types of applications, each of those request a diverse set of resources. There are CPU- (e.g., GZIP data compression [97], scientific computing [96]), memory- (Multigrid application [98], multimedia applications [99]) and network- (e.g., Web services [100]) intensive applications, and their performance is determined by the availability of the respective resource. As a result, the resource

manager, should be able to treat these types in a different way and allocate the necessary resource in order not to impede their performance.

C. **SLA heterogeneity:** Service Level Agreement (SLA), represents a quality of service (QoS) guarantee between a client and a DC operator. The SLA specifies the QoS metrics, and the DC operator should manage the resources accordingly in order to meet those requirements. There are several QoS metrics across different providers [101, 102], however all of them are related to the performance or availability of the service. QoS can be defined as the uptime of a VM [80], or the average response of a VM in the DC to a incoming request [93, 103–106], while it can also be defined as the average acceptable packet loss percentage in the network [107]. Therefore many different performance models should be evaluated to predict the application performance and adjust resource provisioning accordingly.

That is the point were an efficient resource allocation algorithm can make the difference. While all these heterogeneous factors are in place, the target is strike a balance between increased VM density in a server and reduced risk for SLA violations.

3.2 Network-agnostic algorithms in DCs

In a traditional DC, compute resources draw most of the power. As discussed above, the optimisation of resources in a data centre is a key objective for data centre operators. In order, not only to reduce the equipment required, but also the energy consumption holistically.

Two broadly known concepts for resource allocation are the *fill-first* or *greedy* and *spread-first* or *round-robin*. Fill-first focuses on optimising power consumption and resource utilisation, while the later is improving device reliability and possibly application performance.

The concept of a greedy resource allocation is that VM are assigned to a server until the server's resources are exhausted. When the first server is full, it then moves to the next and the process is repeated. The result is a server tightly packed with VMs, high utilisation and less powered physical machine, therefore it is very energy efficient.

On the other hand, the spread-first concept, distributes the VMs across the DC in a round-robin way and balances the VM load across the entire DC, thus, increasing the number of active physical devices. That is the default strategy for resource allocation in OpenStack [108]. Specifically, the Openstack scheduler, picks the servers that have the most available RAM to host a VM request. However, in most cases, the servers and network are not heavy loaded, the applications run smoothly and the performance may be enhanced.

Although these strategies may seem naive and not able to provide the optimum results according to specific data load characteristics, they are implemented in real DCs and they are usually either the point of reference or the base of many more sophisticated algorithms. However there are several aspects to consider when implementing these two approaches.They are addressed in many ways in the literature,as presented in the following paragraphs. In short: in fill-first algorithms, continuous high load on servers will eventually affect their reliability. Additionally if the DC operator want to provide an SLA, there is no available capacity to accommodate load burst that can be predicted and managed in virtualised environments. In spread-first algorithms, the main concern is about wasting resources, since more hosts and networking devices are activated than required.

A energy aware resource management algorithm based on the fill-first concept is presented in [80]. Beloglazov *et al.* have studied both the scheduling and the orchestration processes. Their energy-aware model has been based upon the Eq 3.1. For the scheduling, the VM is placed in the host that have will result to the minimum energy increment in the DC, meaning to an already utilised server or if not any available to a new one. To avert problems with over-utilisation there is high limit of utilisation that makes a server candidate.

In order to optimize the existing placed VMs and adapt to the dynamic nature of DCs, their orchestration algorithm is continuously monitoring the infrastructure and selects VM that if migrated could improve energy consumption, according to a defined policy. The *minimization of migrations* policy, is based on two thresholds for the maximum and minimum acceptable utilisation of a server. If the overall utilisation of the server is below the minimum threshold, all the VMs are migrated and the host is put to sleep. If it is over the maximum threshold, the host's VMs are sorted based on the power they consume. The VM(s) with the minimum power share that will eventually drop overall utilisation below the threshold after the migration are selected. Then the VMs are provisioned according to the scheduling algorithm. The second policy is called the *highest potential growth* policy, and is in principal the same with the previous one, but on a high load the VM is selected as follows. The VM are sorted according to the percentage of the actual CPU usage relatively to the requested VM usage. So, the VM that utilises the least CPU of what it provisioned, has the potential to request it and should be migrated in order to avoid problems in the future.

In future work[109], the authors have conducted a cost-based analysis, about when a VM from an over-utilised server should be migrated. As cost, they define the price C_p that a DC owner pays for the consumed energy and the price C_v that the owner pays to the end-customer in case of an SLA violation. Both of these metrics are per time unit. In a DC environment, the

workload of a VM is variable and that is the reason that DC operators are provisioning VMs requests to hosts beyond their nominal capacity. In case though that all the VMs require their assigned resources, the SLA is inevitable, and one or more VMs should be migrated. If everything run smoothly the operator pays only the C_p, while during an SLA violation of duration t_v, the cost (for the t_v timeframe) is $(C_p + C_v) * t_v$. In case a migration of duration t_m is triggered, then the total cost is the power of the current and the new host for t_m and the cost of violation for as long the violation occurs.

The authors conclude that mathematically the optimal moment to trigger the migration is t_m before the violation starts. Since algorithms are not able to have accurate predictions about workloads in the future, the best possible moment to migrate the redundant VM is the moment that the SLA violation will be detected. Extending their work for VM selection, they introduce a *minimum migration time policy* and choose the VM or VMs that require the minimum time to move to a new host and will reduce the utilisation sufficiently to avert the violation. The migration time is estimated as the amount of RAM of the VM divided by the available network bandwidth between the two hosts, while the host is selected with the power-aware scheduling algorithm presented in [80]

Similarly, Jansen *et al*. [110], have proposed an energy-aware enhancement to a greedy algorithm, the *Watts per Core* and *Cost per Core* policies. In the first case, the algorithm filters out the servers that do not have available resource and chooses the server that will have the least additional power consumption per core, while in the later policy, the server that will increase the least the electricity cost per core is chosen. The results suggest that the proposed algorithms have almost the same performance, achieving up to 14% energy savings and up to 26% cost saving comparing to the a greedy algorithm. The difference between cost and wattage based policies, is expected to be greater in more heterogeneous environments or inter-DC VM allocation, where electricity costs differ.

A performance oriented scheduling algorithm is presented by Lloyd *et al.* in [111]. They introduce a busy-metric, utilised to always select the server which is the least busy. Busy-metric is based on CPU time, disk read/writes and network aggregated incoming and outgoing traffic. Its factor is normalised to 1, and extra weight is put to CPU time due to its importance as a resource. Additionally there is a VM parameter to avoid putting too many VMs on single PM:

$$BM_n = (2 * cputime_n) + dsr_n + dsw_n + nbr_n + nbs_n + \frac{2 * Hosted\_VMs}{PM_{cores}} \tag{3.4}$$

Least-busy placement algorithm manages to improve performance by 19-29% on the tested application compared simple round-robin while reducing the number of required server by 2-3%

In [112], Lawson *et al.*, are proposing a power aware decision algorithm for VM scheduling. They consider a scenario, where the DC is also generating renewable energy for its own needs. The model assumes that it is possible to switch on and off the servers according to the incoming compute load. To estimate the energy consumption of the DC the authors are based on Eq 3.2. To optimize the system they are applying the Markov Decision Process(MDP) in an M/M/c/k queue. A new server is added in the pool of servers if all the previous are fully occupied. Although that is an interesting way to present the scheduling problem, the authors do not provide any evidence whether arrival rates or servicing rates can be modelled like this. Furthermore, the algorithm optimizes the energy consumption in "greedy" way and does not cater for performance of the applications.

Sun *et al.* [89], have studied the workload-based VM resource allocation, namely, the continuous monitoring of the actual load in a server in order to take a decision. Many resource management platform vendors like VMware DRS [113], Microsoft PRO [114], HP PRM [115] and IBM PLM [116] are dynamically monitoring the workload of VM to improve performance isolation between VMs (one heavy loaded VM should not impact the performance of another co-hosted VM) and provide a platform to implement different resource allocation strategies. The problem with workload-based resource management is that it is unaware of implications on the application layer, so it hard to monitor and prevent SLAs. To address this issues, workload-based algorithms are based on the assumption that if utilisation is beyond a threshold, the risk of an SLA violation is high, so the utilisation should be kept below this threshold [117], [118], [119] and [80]. Usually there is also a lower threshold to detect underutilised servers. In both cases, VM migration is triggered to move the workload to a more appropriate server.

Upon the determination of over- or under-utilised server the resource manager should select the VMs that need to be migrated. There are several studies proposing different VM selection criteria, in order to migrate a VM to a new server that will neither over- or under-utilised. Wood *et al.*[118] have proposed to choose the smaller VM, meaning with the lower resource requirements, to be migrated, while Farahnakian *et al.* [119] selected the VM with the minimum migration time, in order to reduce the load on the network (migration time was calculated based on the VM's memory size and network availability [120]).

Chen [121] and Bobroff [122], have worked on minimizing the number of under provisioned server for each time slot. Finding the optimal solution for each time slot for all the VM of the DC will consume a lot of resources and the performance of the algorithm in a big DC will be

challenging. Moreover, the most important problem, is that this approach might results into continuous VM migrations of the same VM.

Xiao *et al.* [117], have a more fine-grained approach for resource utilisation. Each server may be considered Hot or Cold in case any of the critical resources(CPU, RAM, disk, network), reach the minimum or maximum threshold. The target is to remove all servers from the Hot group, while minimizing the servers in the Cold group. For each server in the sorted Hot group, the algorithm picks the VM(s) that will drop the server's temperature the most and migrates it the server that will have the most even distribution of utilisation across all resource. For servers in the Cold group, the algorithm will search for candidate hosts for the server's VMs and if it succeeds, the VMs will be migrated, otherwise they will remain in their original server.

Hierarchical Resource Management (HRM), has been proposed by Goudarzi *et al.* [123], in order to reduce problem complexity for optimal resource provisioning. They propose a resource manager for each level of architecture, so from bottom upwards there are chassis, rack, cluster, DC and even inter-DC resource managers. A VM is assigned each time to level below, based on resource availability, power capacity and temperature. The VM assignment method follows the greedy concept from the higher level of resource manager to the lower one.

For monitoring and performance purposes the information flow is bottom-up, and each level is receiving an abstract chunk of information. In cases that a VM migration is dictated, the lower level resource manager is reporting to the level above and the VM assignment procedure is started. The abstraction of resource in each level, allows the resource manager to solve only a subset of the problem. Additionally, a opportunity of HRM is the ability to implement different resource management algorithms in each level of the hierarchy. For instance, greedy concept on the cluster level, but spread-first on the rack level. In that way, the traffic and load is aggregated to a specific part of the DC, but the load inside the cluster is balanced.

3.3 Network-aware algorithms in DCs

In a DC the network consumes approximately 12%-15% [49] [124] of the overall energy consumed by the data centre. However, networks have received relatively low attention comparing to IT resources, although apart from an important energy factor, may affect largely the performance application, especially those that are latency sensitive. Efficient network resource allocation may affect the infrastructure in several ways. For instance, VMs placed closely together will communicate with less latency and put less strain in the core part of the network which is usually loaded [20].

From an energy point of view, there are two approaches to reduce energy consumption in the network:

- Improve the hardware energy consumption
- Implement efficient algorithms to improve link utilisation and put idle devices to sleep-mode

In this thesis we are focusing in the later one. Data centre networks are usually designed and provisioned by taking into consideration their peak load, but normally they actually traffic is much lower. Studies have shown that an idle switch consumes up to 90% of the energy that it would consume when performing with the highest traffic as shown in Fig. 3.1. Designing adaptive networks, which will be able to switch off the elements when the traffic is low, will lead to 100-2000W/switch power reduction depending on the switch [12].

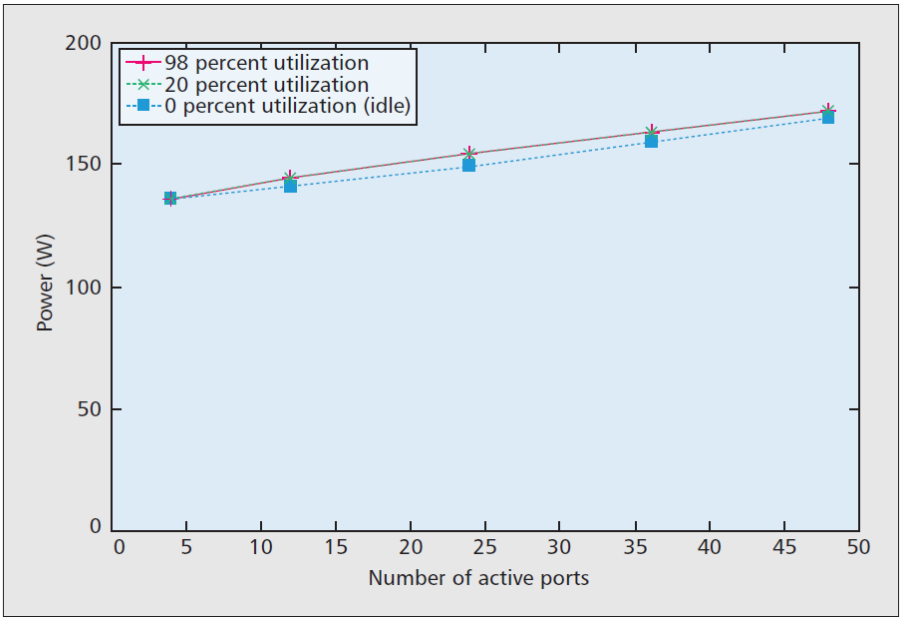


Figure 3.1: Power consumed by a 48-port switch as a function of the load (traffic) through the switch[12].

Gupta and Singh [125] were among the first researchers proposing energy savings in communication networks by putting devices in sleep-mode. Their study showcased that it is possible to put interfaces to sleep on the implementation of switch protocols and OSFP and IBGP routing protocols. Off course, the protocols' specification should change to adapt to this potential change. Under this perspective we are reviewing the following papers and designing our scheduling and orchestration algorithms in Chapters 4 and 5.

Although energy is an important issue, the overall performance of the network can affect the performance of the applications as well. Bottlenecks in the network, may affect the desired throughput, and paths with high latency affect data intensive application sensitive in time and synchronisation. Therefore, the resource allocation algorithm has many aspects and cannot stick on satisfying only one criterion. The need to add network awareness in scheduling and orchestration algorithms has been identified by several researchers.

There are two approaches for network-aware resource allocation algorithms. The first one studies only the computing and network load in order to decide where to place a new VM or where to migrate an existing one, and the second approach identifies under- or over-utilised physical machines and calculated the cost of migration before placing a VM.

3.3.1 Network traffic oriented algorithms

Adami et al. have suggested an algorithm based on performance parameters and economic cost of computational resources [126]. They have defined a set of performance indexes in order to find the optimal server:

- **Computational index (CI)** indicates the capability to a server to perform a certain computation task. In this paper it depends on the number of CPUs and the amount of RAM. But it can also include other aspects, such as processing power or GPU processing power.
- **Data index (DI)** measures the minimum time required to move an amount of data from a server A to a server B. This is related to application that require to transfer data all over the data centre, either for storing (e.g. data base replication) or for distributed data-intensive applications.
- **Latency Index (LI)** is critical for data intensive application, since it can affect their performance Thus, lower latency paths are preferred.
- **Network Index (NI)** estimates the ratio of latency to bandwidth of a path from point X to point Y in a data centrer.

The algorithm aims to strike a balance between network and compute resource by minimising the distance in the 3-dimensional of CI, DI, and LI. Weights may also apply to the function, in order to prioritise specific indexes. The algorithm has been compared with Random VM placement, Round-Robin server selection and the 2-dimensional aspect of the this algorithm, that took into account CI and DI.

Random and Round-Robin algorithms produce unpredicted results. Either the execution time is not stable or the compute/network resources are not efficiently utilised. The 2-dimensional algorithm has a good performance but in some scenarios tends to overload servers and consequently specific links that might lead to congestion. The proposed algorithm, on the other hand, has the best performance since it has more granular control and deeper knowledge of the infrastructure.

To address the resource allocation problem, Dong et al. [127], have proposed a methodology based in two stages the static and the dynamic stage. The authors aim to optimize at the same time network traffic and energy consumption in the DC.

In the static phase, they are achieving that by provisioning VMs that exchange data intensively into the same PM or nearby PMs. They start off by placing the first pair in the same host and then continue until the PM full, then move to a second PM. If not both of the VMs can fit in a single PMs, the second one is chosen based on the distance, measured in number of links, from the first one.

In the dynamic stage, in pre-defined intervals the static phase is being run to identify configurations that could potentially improve the status of the infrastructure, due to changes in the VM workloads. That will result into a list of VMs that will have to be migrated. If this number is acceptable by the DC operator, then the migration is triggered. If not, then the algorithm randomly selects a single VM from the candidates for migration and migrates it to the new PM. and starts over. This procedure runs up to a certain number of iteration or until the number of VMs to migrate meets the desired threshold.

The first problem with the algorithm, is that exhausts both compute and network resources, making it prone to possible degradation of service. Additionally, it assumes that the workload is known in advance, which is hardly the case in production environment. and the thresholds for VM migration are loosely defined. However, the results show that the overall energy consumption in the simulated DC has been reduced.

In [88], Biran et al. introduce the Min Cut Ratio-aware VM placement problem (MCRVMP). MCRVMP considers constraints on both IT resources, such as CPU and memory and network resources emerging from the complexity of network topologies in and routing schemas in modern cloud date centres. MCRVMP starts with the assumption that traffic demands are time-variant and works on minimizing the maximum ratio of the demand and the capacity across the network, in order to find network cuts that can absorb unpredictable traffic bursts. Since MCRVMP is a NP-hard problem two heuristics have been applied:

1. **2-Phase Connected Component-based Recursive Split (2PCCRS)**
   The idea is to place VMs interacting intensively close to each other. In more detail, the first phase assigns connected VMs to subtrees rooting to associated switches recursively. In the second phase, it splits the connected VMs and tries to place according to the problem defined above, which is possible to solve in really small non-realistic scale.

2. **Greedy Heuristic**
   This algorithm greedily provisions VMs on available hosts.The only objective is to identify the traffic demands from two connected VMs, sort them from higher to lower and then try to place the VM in the host that minimizes the maximum cut.

The effectiveness of the proposed algorithms is measured with 2 metrics. First is the cut load in parts of the network and second is the execution time of the algorithms itself in order to take a decision. Execution time is similar between the two heuristics for small and medium DCs and there is a big difference in bigger DCs. The execution time of the greedy heuristic is increasing linearly with size of the DC and it was measured up to 50s, while 2PCCRS, which is recursive and tries to find a solution close to optimal, may require up to 1800s. However, 2PCCRS may be limited to the first optimization step, since the results are usually good enough.

The results suggest that MCRVMP-based placement algorithm improve the data centre scalability, achieve lower network cut-load and support time-varying traffic demands. The heuristic can be efficiently implemented in medium size data centre in order to have a low overhead in scheduling procedure.

Graph partitioning techniques have also been proposed by Alicherry [128] for VM placement. Alicherry assumes that the workload of the infrastructure is known in advance and can be represented as a graph $G = (V, E)$ where nodes are the tasks and edges are the communication cost. The graph then can be partitioned to have either none or minimum communication between partitions, so that the communication cost will be minimised and the performance of the application will be improved due to lower latency.

Each partitioned can be scheduled into a rack or cluster of racks (depending on the design and the need of the DC owner), so it has a physical upper bound of capacity, but there is lower bound. The VM placement follows the greedy approach. The most heavy communicating VM is placed first, and its neighbours are provisioned in the same partition if possible. As an enhancement to existing solution, the authors suggest to migrate pairs of VMs that are communicating heavily and have provisioned to different partitions to the same partition of the DC. The results show great improvement in the diameter of the placements(i.e. the max distance between any pair of

VM) comparing to greedy and round-robin approaches and smaller effect for the inter-partition traffic. That presents an opportunity for energy consumption reduction. Due to more closely located VMs and fewer utilised servers.

Similar objectives for the VM placement has been introduced by Vu *et al.* [129] in the orchestration level. A power- and network-aware algorithm has been proposed Vu *et al.*. It is an orchestration software that optimizes and reconfigures the existing infrastructure to reduce communication footprint and energy in a heterogeneous environment in three steps:

1. Identify VMs that are over- or under-utilised according to a desired threshold
2. If the physical machine (PM) is underutilised, all the VMs are migrated. If it is over-utilised, one or more VMs are selected for migration.
3. Find the destination server for the selected VMs

Authors' contribution lies on the final step. Although they have created a communication cost based decision process, that minimised overloaded links by migrating VMs interacting heavily close to each other, they realised that this resulted to underutilised resources. The reason is that network intensive VMs are not necessary CPU-intensive, resulting in PMs that had loaded links but available resource that could potential host CPU-intensive applications.

To overcome this problem, they implement 2 strategies, depending whether the migration is triggered by over- or under- utilised PM. If the migration is caused due to over-utilised server, they follow a Greedy (Fill-First) approach, which proven to reduce power consumption. On the other hand, if the migration is triggered by under-utilised PM, the follow the network-aware approach. In that way, the authors argue that a mix is created in heavily used PMs, and the balance between power/utilisation and network communication cost is achieved, as per the simulation results.

Researchers have also studied the resource allocation focusing entirely on the network. In [87], authors assume that CPU/memory capacity tools have already decided the number of VMs a server can accommodate and a slot can be assigned to any VM. They also consider static and single-path routing between a pair of hosts and calculate the communication cost. That narrows down the problem, since performance and utilisation issues in servers are not taken into account. Trying to solve the problem even in the simplified version is NP-hard and the authors resolve it to two heuristic algorithms.

The first one implement the Cluster-and-Cut approach aiming VM pairs with heavy mutual traffic to be assigned to slot pairs with low-cost connections. The second principle is divide-and-

conquer. The algorithms partitions slots in slot-clusters and VMs in VM-clusters and initially tries to associate slot-clusters to VM-clusters and then maps VMs to slots.The VM clustering is done through a min-cut algorithm that groups VM with large traffic in the same group. Slot-clusters is defined according to slot's cost connection. Slots with low-cost connection are placed in the same cluster.

The authors argue that following such approach and implementing their algorithm is useful in scenarios where the VM communication follows a diverse traffic pattern, or the application has many components or the network itself is not load-balancing the traffic by design. The algorithm itself though makes some assumptions that are not realistic for traditional DCs. Most of these refer to the compute side of the VM placement problem. VM requests are not of the same size and in most cases very dynamic, meaning that the number of VMs running is changing and dynamically adapting to workload requirements. Additionally, is not realistic to claim that VM requests are known beforehand and placement will be a one-off decision. However, if the application designers follow the trend of micro-services, and DC operators design infrastructure to host such an adverse workload, aspects of this algorithm and pure network-aware resource allocation algorithms, in general, may be very applicable and useful.

In [130], Martini *et al.*, have conducted a study on four resource allocation strategies that built on top of an SDN-based orchestrator and they based their decision on real time flow statistics. They conceived four resource allocation algorithms split into 2 categories:

- Server-Driven: first finds the host device and then finds the network path towards the selected host

- Network-Driven: initially identifies optimal network routes and then a server attached to that route to deploy the VM.

For the selection of the server or the routing path at each step the authors utilised the fill-first and the spread-first heuristics, resulting in the four resource selection algorithms. That means that each time, they either used the most utilised or the least utilised server/path respectively. Link utilisation is based on available bandwidth and historical data to have a short term estimation of the traffic, and server utilisation is based on available CPU cores. The authors argue that in the near future, CPU cores will be so abundant that will not constrain the server selection in a DC and they present results only for the Network-Driven scheduling algorithms. The results are compared with a pure Spread-First algorithm that picks the least loaded server

with shortest path selection and an algorithm that randomly picks an available server and the network route is predefined.

The evaluation metrics were the blocking probability of an incoming VM request, the percentage of links with utilisation lower than 50% and the percentage of links that were overloaded due to lack of network awareness during the provisioning. The authors concluded that the Network driven - Fill-first selection provided the best results since the VM request rejection rate was low and the overall utilisation improved.

On the other end, there are researchers that focus on a single step decision for network and compute resources. In [131], Buysse *et al.*, based on their previous work have studied whether integrating network and compute resources in the scheduling decision will lower the total energy consumption in optical clouds instead of the classic 2-step approach where the network routing is decided upon VM placement has finished. The authors have developed Full-Anycast (FA) algorithm, which is entirely based on power consumption to decide the resources that will allocate to a request, on a single step. As shown in 3.5, the factors that determine the provisioning of the VM, are the additional power required for the links that will be activated, the networks nodes and the IT infrastructure. FA algorithm outcome is determined by the weighted sum of these 3 factors, through a set of parameters $\{\alpha, \beta, \gamma\}$, so that the DC operator can have granular control over the process. For example, the DC operator can optimize towards network resources with a set like $\{1, 1, 0.001\}$ or IT resources with a set like $\{0.001, 0.001, 1\}$

$$P^{FA}(l, \phi) = \alpha * P_{link}(l) + \beta * P_{node}(l) + \gamma * P_{DC}(l, \phi) \tag{3.5}$$

Their results suggest that for optimizing energy consumption, jointly compute and network resources should be taken into account, especially for low to medium load cases. The optimal network energy consumption cannot be achieved with the 2-step approach. Moreover, the researchers concluded that optimising a single resource with FA, or following a greedy approach, will not minimise the overall energy required. Specifically, the figures show a energy reduction ranging from 38% to 48%, in case of a network "friendly" but balanced set of weights similar to $\{0.1, 0.1, 0.001\}$ or $\{0.1, 0.01, 0.001\}$. Finally, they argue that energy reduction does not lead to excessive service blocking conditions, except for cases that the network is sparse and under heave load.

Finally, an interesting approach for the VM placement problem is of Piao *et al.* [132]. The authors are based on the assumption that a VM is usually mounted on some remote data storage

or accessing a remotely placed database, and although VMs are easy to move around, migrating data is much more difficult to ensure resiliency and availability during migration.

Piao *et al.*,present a network aware resource allocation algorithm, taking into account remote data storage. The storage access time is the key factor. In public data centres, data is stored arbitrarily and can be stored across the data centre or even over different date centres, depending on the application and the price the customer is paying. In the known resource allocation algorithms, as discussed above, the scheduler provisions regardless of the storage location and the access time. In some cases, this might significantly reduce the performance of the VM since data would be transferred over unnecessary long distance. In the authors' approach, the data is within federated storage resources and the bandwidth between storage and the server that will host the upcoming VM request already known.

The VM placement algorithm, sorts all the available hosts according to the access time for storage blocks that VM want to access. However the dynamic nature of network, might lead to unexpected network. In that case the performance of data intensive applications will be degraded and can become intolerable according to the SLA. Therefore they have proposed a mechanism that is triggered when the applications execution time crosses the threshold specified in the SLA and the storage access time have changed comparing to the initial. If such a situation occurs, the VM placement algorithm is activated in order to migrate the VM to a server that will satisfy the SLA. If this is not possible, the server with the minimum access time is chosen.Although this approach cannot enforce the SLA, the simulation proves that improve the task completion time.

3.3.2 Migration cost oriented algorithms

Among the first authors to take into account cost of migration for a VM placement or replacement are *Mann et al.* [13]. The authors have introduced Remedy as network state management system, that utilised VM migration, a mean of decongesting heavy loaded links on the long term, if simple flow re-routing is not enough. The generated traffic load due to the migration is also calculated on top and that represents the migration cost. Remedy is broken into two basic parts :

1. **VSH** - VM selector heuristic
   VSH selects the candidate VMs for migration based on: a)the number of VM communicating with, b) the total inbound and inbound traffic they generate on congested links for their communication with other VMs and c)the migration cost. The migration cost is calculated based on the VM's memory size, their page dirty rate and the available bandwidth on the link. The VM that scores the lowest sum for each of these three factors is migrated.

2. **TSH** - Target selector heuristic
   Candidate target hosts are defined upon availability of compute resources in physical machine and network bandwidth along the route from source host. Additionally, TSH calculates the available bandwidth in the part of the network that will be affected, as a result of the migration. That means the rerouted flows for VM-to-VM communication. TSH, takes the weighted sum of all available bandwidths and utilises it to rank all candidate target hosts.

```
1: function VSH
2:     rank VMs in increasing order of their approx migration cost = 120% of VM size
3:     rank VMs in the increasing order of their number of communicating neighbor
4:     rank VMs in the increasing order of their total input/output traffic
5:     return VM with lowest sum of the three ranks
6: function TSH (VM v)
7:     H ← set of all hosts with enough spare CPU capacity to place v; H_f ← ∅
8:     calculate available b/w from current host of v to each host in H
9:     for each host h in H do
10:        calculate the migration traffic and migration feasibility of VM v to host h following Sec. 2
11:        if migrating v to h is feasible then
12:            add h to H_f
13:            calculate available b/w for each flow of v when re-route to host h
14:            calculate normalized weighted sum of available b/ws calculated in previous step
15:    return a ranked list of hosts in H_f in the decreasing order of their normalized weighted sum
16:
```

Figure 3.2: Pseudocode for VHS and THS. [13]

The ultimate goal of the aforementioned procedure is to balance the network bandwidth in the long term. The authors created a simulation to evaluate the performance of Remedy and claim that they managed to reduce unsatisfied bandwidth by 80% to 100%.

The main contribution of the Remedy methodology is a fast model to quantify the migration cost in terms of the overhead in the network, which may be a real problem in already congested links. It enables any orchestration module to have a quick and dirty metric to evaluate according to its respective criteria, whether the VM migration should be triggered or not. That is the reason why Remedy is used in literature as a point of reference.

In [133], the authors applied the Remedy methodology in a real test-bed that emulated a DC to allocate compute and network resources to workloads. In parallel they enforced QoS policies by reserving bandwidth for migration and prevented network overload during the migration. The reserved bandwidth was the minimal required to complete the VM migration in a user-

specified time slot . Their results suggest that Remedy approach performs better in DCs with high bandwidth links, i.e. 1Gbps and above.

In [134] and [135], the authors present S-CORE and implement it in a SDN environment as well. S-CORE is a distributed VM management algorithm that optimizes network's communication load, in contrast with Remedy that is centralised. VM's are gathering information about communication generated or received with another VM and they are triggering the migration, one VM at a time.

The VM candidate for migration is decided upon a token policy. The VM that holds the token may start the process. The way the token is distributed to the VMs is defined by the system administrator. The communication cost is calculated if the VM is not collocated with another, meaning that the flow is routed thought at least on switch and a pair of links. The VM is also able to predict the traffic load after a probable migration. So if the future communication cost is lower than the current, the migration is triggered. However migration overhead for the network is not taken into consideration.

S-CORE authors argue that network bottlenecks are caused mainly towards the core of the network, where the traffic is aggregated, rather that the edge. Therefore, they are trying to exploit locality and migrate VMs that are heavily interacting as close as possible, even in the host. The results show that S-CORE managed to reduce communication cost by 87% compared to other approximately optimal approaches applied on traditional DC architectures.

3.4 Resource Allocation in DDCs

As mentioned already, resource allocation is even more important in DDCs, due to the nature of the architecture and the requirement for logical connection of resources, in order the application to be executed. Resource allocation involves selecting the necessary resources, provisioning the resources at the different blades and configuring the network that interconnects the specified blades.

To the best of the author's knowledge, there is limited contribution for DDC's resource allocation. Most of the work, has been carried out by Ali et al. in [136], [137] and [138].

In [137], the authors have developed a mixed integer linear programming (MILP) model to enhance resource utilisation that included also the power consumption of the network fabric. The MILP model was simulated to a set of 5 servers and three types of workload Processor-, Memory and IO-intensive applications (PI, MI and IOI respectively). The results suggested an

optimum allocation in a DDC may result to 42% in average savings for MI applications, 24% for IOI applications and 11% for PI application, compared to the normal DC.

To examine the problem in larger scale, the authors have developed the EERPVMM-DS heuristic that follows the greedy approach and attempts to minimize power consumption [138]. The flowchart of the heuristics is presented in Fig. 3.3. The authors are based on the assumption that VMs are arriving on predefined time slots and they have a finite duration, which is an integer number of timeslots. In each time slots all the "active" VMs will be allocated from scratch. That means, VMs with large life-cycle could possibly be migrated several times.

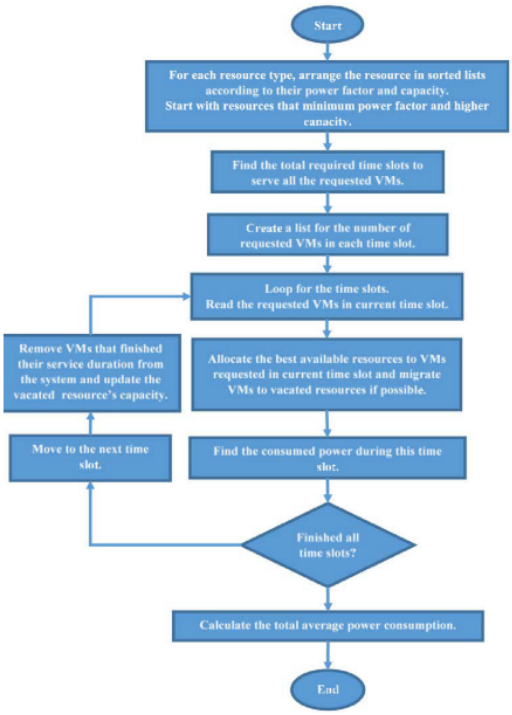


Figure 3.3: EERPVMM-DS heuristic flowchart [138]

The target is to minimize the resources utilised, while picking the less power efficient devices. The heuristic first identifies the CPU, Memory and IO blades that have available resources. Then, it sorts them based on the Power Factor (PF). PF is a metric related to the power consumption of the device in idle state. The higher the PF, the lower the consumption in idle state. Devices with high PF are preferred. If two devices have the same PF, the one with highest capacity is picked. Resource are allocated the following order: CPU, RAM, IO.

The results showcased an average of 55% savings for MI applications, 36% for IOI applications and 21% for PI application, compared to the normal DC. The significant improvement between the two papers, is due to the infinite duration of the VM in the MILP model.

Extending their work, the authors presented the EERP-DSCF (Energy Efficient Resource Provisioning in Disaggregated Servers With Communication Fabric) heuristic in [136]. They have dropped the time-slotted assumption and the VM requests are served upon arrival. The heuristics instead of using the PF to sort the blades, picks the first available CPU blade from the first available rack and then continuous by identifying the closest available Memory and IO blades to allocate the VM. They have implemented a shortest path calculation to minimise delay for blade communication. However, the algorithm provide no guarantee for available bandwidth and the delay for blade communication, since the allocation is based on network's best effort.

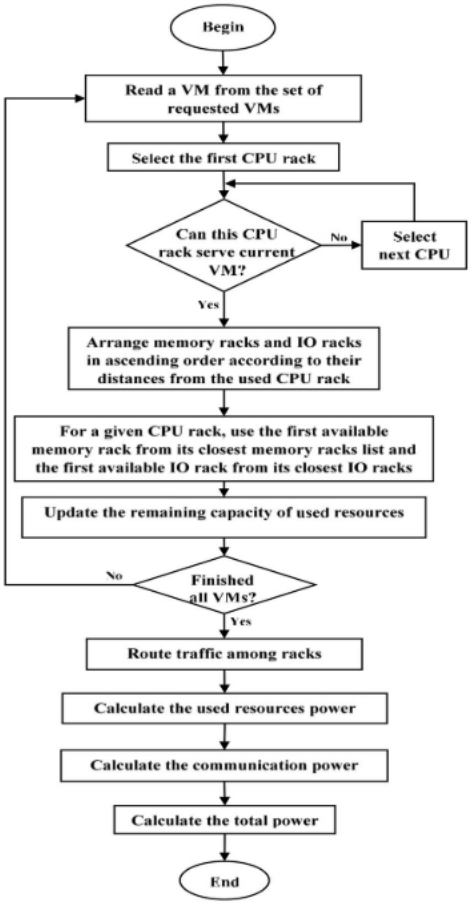


Figure 3.4: EERP-DSCF heuristic flowchart [136]

For the same set of workloads, the results followed a different trend than the previous simulations. MI and IOI application had similar behaviour and achieved 42% average savings, while PI applications only 3% savings. Table 3.1, provides a cumulative representation of the aforementioned results.

	PI	MI	IOI
MILP	11%	49%	24%
EERPVMM-DS	21%	55%	36%
EERP-DSCF	3%	42%	42%

Table 3.1: Cumulative Table of DDC resource management power saving results

3.5 Discussions

Resource management is usually referred as the automated procedures regarding to arranging and coordinating multiple resource groups deployed across a DC in order to expose their capabilities as a single instance that satisfies the user requirements. In this chapter, the author gave an overview of network agnostic resource allocation and analysed the status quo of network-aware resource allocation algorithms.

In Section 3.2 most of the algorithms presented are aiming to improve server utilisation or the overall energy consumption. Creating groups of hot and cold servers of servers that Xiao [117] proposed improves identification process of servers that need to migrate their workload. One of the drawbacks in some cases like the algorithm in [128], require knowledge of all the VM requests in advance. Finally many algorithms like [80, 109, 110] measure utilisation only for CPU cores or energy consumption of CPUs, although memory is usually the resource that is not enough.

In Section 3.3 the algorithms presented aim to improve the overall network utilisation in a cloud environment. However, there are two limitations, that make them inapplicable to a DDC. First and foremost, they don't consider that a logical combination of multiple resources is required to complete a certain task, because they are applied on traditional servers. Secondly, in the majority of the algorithms no bandwidth reservation policies are applied, which is a prerequisite for the proper functionality of a DDC, and more specifically CPU-to-RAM communication. Although they are trying to improve network performance, in the end they cannot guarantee a QoS. That is not sufficient for a DDC where strict bandwidth and latency requirements, as analysed in Table 2.1, should be satisfied.

Although the algorithms cannot be utilised as is in a DDC, the ideas and concepts may provide good indications towards a solution for the resource allocation problem. The access time as a measurement to describe network performance [132], is a metric that would be very

useful to identify resilient paths for CPU-to-RAM communication. Divide and Conquer [87], is a nice approach to break down the problem at an orchestration level and quickly take decision. Furthermore, controlling the scheduling outcomes by weighted functions [126] provides fine-grained control over the resources to the DC operator, making it adaptable to diverse workloads.

Finally in Section 3.4, Ali et al. [137, 138], have presented a scheduling algorithm for DDC that follows the greedy approach, meaning that first a resource blade is exhausted before moving to the next one. In terms of energy efficiency, they are reaching close to optimal. However, there is no delay guarantee for CPU-to-RAM communication to ensure performance, and as with all greedy approach the system is prone to SLA violations. Additionally, their approach on migrating VMs, may results in severe performance degradation, since in their attempt to minimize the energy consumption they take the decision to migrate the VM lightly.

CHAPTER 4

DISAGGREGATED DATA CENTRE RESOURCE SCHEDULING

Disaggregated Data Centres require logical connection, along with physical connection of the hardware, in order to accommodate a job, task, application or a VM. The management software has complete control over this procedure for the proper functionality of the application running on top. Scheduler is the functional component that allocates resources per client or application. In Chapter 3, we have discussed many aspects of scheduling in a traditional DC. This chapter defines the objectives for an effective resource allocation algorithm for a DDC. The efforts to develop the first scheduling heuristic for a DDC that is network-aware and its outcomes are presented and analysed in detail. Finally the evaluation model is described and the simulation results are discussed.

4.1 Network- aware resource scheduling for DDCs

In a DDC, the coordination and logical connection of all the hardware and software falls under the control of the management software. As in a traditional DC, the scheduler is the element of the management software stack that is responsible for finding available resources, applying the respective allocation policies and take a one-off decision in which server the VM will be provisioned [108]. However, developing the scheduler that will take advantage of the a DDC infrastructure is a a challenge, since the network resources should also be taken into consideration, on top of the compute resources. Therefore, the scheduler should have a global overview of the environment and unified control over compute and network resources.

Knowledge of the specific characteristics and requirements of disaggregation is required during design of a DDC scheduling heuristic, in order to ensure smooth operation and steady performance. The main objective is to satisfy the network requirements as presented in Table 2.1. According to [2], CPU-CPU communication is not possible with current technologies and it will be challenging to implement in the future due to its extreme bandwidth and latency requirements. If processor power sustains pace with Moore's Law, it is safe to assume that there will be no application requiring more cores than a single server can provide. On the other hand, CPU-Memory communication is possible to implement with specific network design. As proposed in section 2.2. Bandwidth and delay are the critical parameters, that should be constantly monitored and carefully allocated to an application request.

Similarly to a traditional DC, energy consumption is of high importance in a DDC, as well. The author has argued in Chapter 1 that the disaggregation of resources could improve DC computing and networking resource utilization. This hypothesis is based on the fact that disaggregation provides the flexibility to logically connect any blade with another and avoid wasted resources as in paradigm in section 1.2. Disaggregation thus can allow statistical multiplexing to occur at a much larger scale and hence naturally enables higher efficiency, which in turn will result into reduced energy consumption, due to less equipment required. To evaluate our case, we designed a scheduler based on the following assumptions.

A VM request is considered as the basic request for resources. It is the standardized way to allocate resources in current DCs and will help the comparison in the next section. However, that does not exclude the case of allocating resources in smaller batches, for example, on a container, micro-service, application, or task basis. It is finally assumed that some local cache memory exists on each CPU blade. This assumption is based on two reasons. Firstly, applications are known to access specific memory blocks continuously. Therefore, fetching these memory blocks from the RAM blade for each memory access multiplies the traffic. Secondly, local memory will

deliver these specific memory blocks much faster and result in a reasonable performance, since the access time is affected by network latency. So, the RAM blade can be considered as a memory expansion module that, as opposed to cache memory, can be allocated to any CPU blade. The amount of local memory is configurable, and in the next section we will examine how it affects the utilization of the servers.

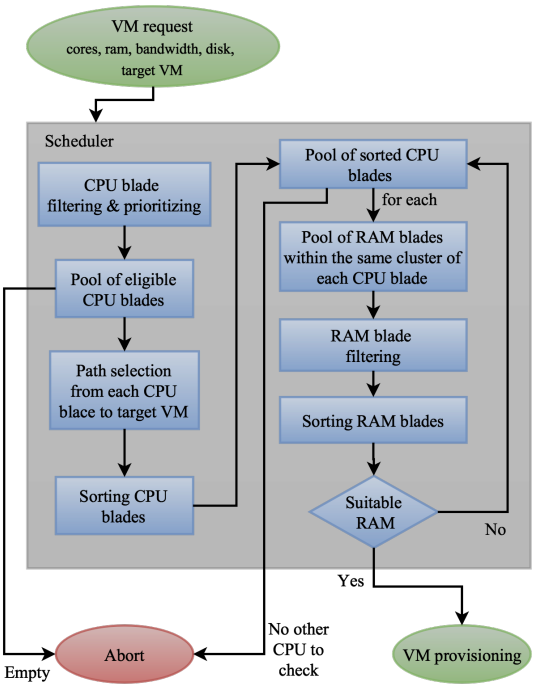


Figure 4.1: Flowchart of the proposed scheduler.

The proposed scheduler provides system administrators with the flexibility to define their own way of managing resources. This is achieved by applying policies which are enforced through three sets of weights, as explained in Eq. 4.1, 4.2 and 4.3, and they are the input to the scheduling software. The scheduler takes the weights into account while ensuring smooth operation and the strict network requirements of a DDC are satisfied. The scheduler is an iteration of filtering, prioritizing and sorting the network and compute resources which makes the best decision according to the policy provided.

It is assumed that each VM request comes with a set of requirements, namely CPU cores, RAM size and network bandwidth for each of the VMs that requires connection, if any. The scheduler has no knowledge of upcoming VM requests, so it performs a real-time, one-off decision based on the current state of the DDC. The flowchart of the proposed scheduler is depicted in Fig 4.1, and the steps undertaken by the scheduler to define the necessary resources are the following:

I. *CPU Blade Filtering and Prioritizing*
The scheduler filters all the CPU blades and determines those that have available cores and network bandwidth for VM-to-VM communication. Then, it prioritizes them according to whether they can accommodate the VM with the currently assigned memory.

II. *Path Selection*
For each eligible CPU blade, the scheduler finds the path that satisfies the following equation for each path connecting the given CPU blade with the target VMs:

$$min(wp_d * p_d + wp_b * p_b + wp_h * p_h) \tag{4.1}$$

where p_d, p_b, p_h are the score[1] for path latency, bandwidth and number of hops; wp_d, wp_b, wp_h the weights for the respective scores; and $wp_d + wp_b + wp_h = 1$.

III. *Sorting CPU Blades*
The scheduler sorts the CPU blades according to performance and utilization factors, as they have been defined by the system operator in the CPU weights:

$$wc_p * c_p + wc_u * c_u + wsp_b * sp_b + wsp_d * sp_d \tag{4.2}$$

where c_p, c_u, sp_b, sp_d are the score[2] for the CPU blade priority and utilization and selected path bandwidth and latency; wc_p, wc_u, wsp_b, wsp_d the weights for the respective scores; and $wc_p + wc_u + wsp_b + wsp_d = 1$. This equation maximizes for the best possible blade.

Finally, the scheduler takes the sorted list of available blades and tries to allocate an available RAM blade for first CPU blade in the list. If it finds a RAM blade, it stops and returns the resources that should be allocated. If not it attempts to find an available RAM blade for the next CPU blade in the list. To find a suitable RAM blade, the scheduler follows the last next two steps:

V. *RAM Blade Filtering*
The scheduler filters the RAM blades and finds those that have enough space and a path that can satisfy the latency requirement.

[2] In order to be able to apply weights to our equations, we have normalised the related values of the variables. If we have N variables to compare, the score s_j, of the j_{th} value of the variable with actual value x_j is :

$$s_j = \frac{x_j}{max(x_1, ..., x_N)}$$

VI. *Sorting RAM Blades*

From the suitable RAM blades, the scheduler sorts the blades based on the latency to the target CPU blade and its utilization, using the following equation:

$$wr_u * r_u + wrp_b * rp_b + wrp_d * rp_d \tag{4.3}$$

where r_u, rp_b, rp_d are the score[1] for the RAM blade utilization, path bandwidth and latency for CPU-to-RAM communication; wr_u, wrp_b, wrp_d the weights for the respective scores; and $wr_u + wrp_b + wrp_d = 1$. Again, the advantage of latency and bandwidth over utilization in decision process is defined by the system administrator through the set of weights.

In modern DCs, as mentioned above, hypervisors with oversubscription of CPU and RAM typically increase the utilization of the virtualised computing resources. In DDCs, the same rule can be applied and can increase even further the resource utilization. Therefore, oversubscription ratios are considered in the scheduling heuristic when calculating available resources, if they are set by the administrator.

4.2 Simulation Model and Scheduler Evaluation

In order to quantify and measure the benefits of a DDC in terms of resource utilization and evaluate the author's effort on a scheduling level, we have created a simulation and examined the effectiveness of the proposed heuristic.

The author created its own simulation tool, due to the special requirements of DDCs, where VMs are assigned to multiple devices. More details about the technical implementation of the tool can be found in Appendix A.

The scheduler should be able to calculate the latency between any given RAM and CPU blades before deciding whether the latency requirements are satisfied. However, in the simulation, blade-to-blade latency has been calculated offline, for simplicity, without affecting the generality of the results.

4.2.1 Infrastructure

Similarly to the proposed architecture in section 2.2, the blades are organized into clusters, which are form by groups of blades that ensure that CPU-to-RAM communication latency requirements will be preserved. The size of the cluster is one of the factors that will be examined.

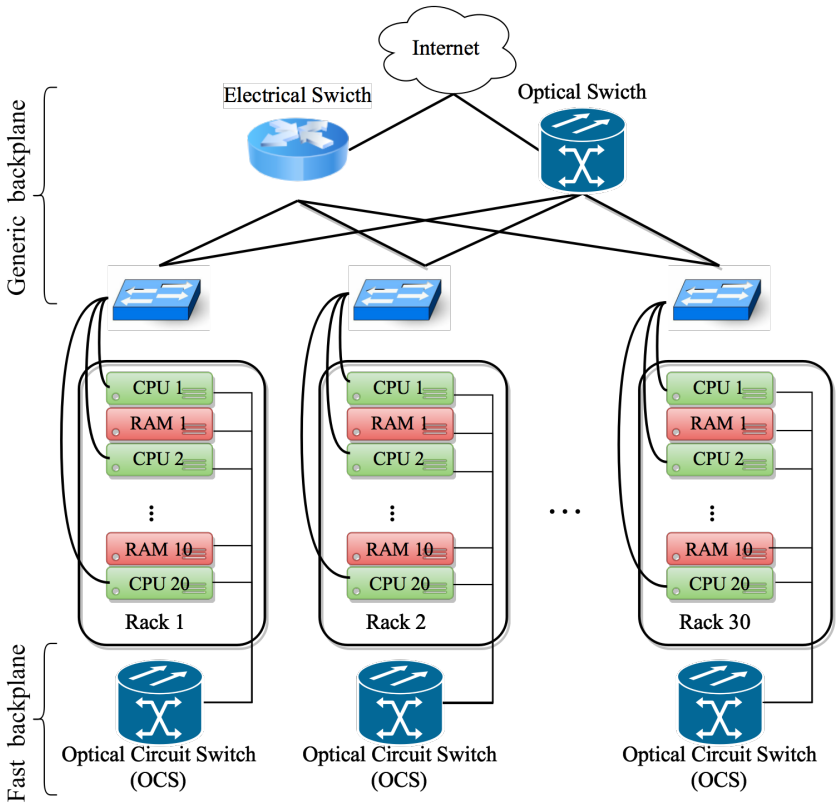


Figure 4.2: Simulation Set-Up

As shown in Figure 4.2, in the disaggregated environment, the "fast backplane" is simulated as a single OCS switch-per-cluster. This does not affect the results, since CPU blades can only connect with RAM blades within their cluster through the fast backplane. In all cases, for VM communication, a hybrid leaf-spine network architecture has been simulated. The leaf switches are 1 G ToR switches with 10 G electrical uplink and 40 G optical uplink. The spine consists of both an electrical and an optical switch. Traffic from VMs that require large amount of bandwidth will be redirected through the optical link to improve the overall throughput.

4.2.2 Virtual Machine request arrival and Energy Model

All cases are tested under the same set of VM requests. The VM request arrival time and life duration is estimated according to the study of Peng *et al.* in [139] and the simulation runs for 40,000 time units. The requirements of each VM request are chosen randomly between the options in Table 4.1. The choices are the default cloudlets in DigitalOcean [140]. Finally, the target VMs are again chosen randomly between the existing VMs. The number of target VMs is limited to 3.

Name	Cores	RAM (MB)	Bandwidth (Mbps)
Tiny	1	512	25
Small	1	1024	50
Medium	2	2048	75
Large	2	4096	100
Xlarge	4	8192	125
XXlarge	8	16384	150

Table 4.1: VM Instances

The policy applied is balanced between performance and utilization criteria. The reason is that in a real DC the operator would like to increase the utilization of each machine, but not stack all the VMs into a single server. This is particularly undesirable because it increases the probability of a service level agreement (SLA) violation, costing the operator more than distributing the load to multiple servers. Storage load and utilization in a DDC, assuming we are following the proposed architecture, is identical to that of modern DCs. There is extensive work that demonstrates techniques increasing the network's and storage servers' utilization and are it is applicable to a DDC [132], therefore it is not included as part of this simulation.

In order to estimate the energy consumption of the DC and the DDC, and have measurable results, we used an energy model, based on the following findings. Hardware consumes a lot of energy in idle state. So, it is advisable to turn the electronic circuits and systems off when not being utilized. The idle state energy consumption is usually around 75% for servers and 85-90% for electrical switches of the maximum energy that each device can consume. Both types of resources have a similar behaviour regarding energy consumption, which in its most simple form can be described by Eq. 3.1 [80] [12], as it was analysed in Chapter 3 and copied here for convenience.

$$E = (a + (1 - a) * utilization) * max\_power \tag{3.1}$$

In Eq. 3.1, max_power is the maximum power consumed when a device is fully utilised; α is the fraction of max_power the device consumed in idle state and utilisation of the device is expressed as a number between 0 and 1. We assume that the inactive devices, can be dynamically switched off.

Additionally, we are not measuring the exact amount of power that a DC may consume during our simulation. but the percentage of the maximum power that it could possible consume. The

author came to this decision, because of the diversity of the devices (servers, CPU blades, RAM blades) and the lack of energy figures for disaggregated blades.

4.2.3 Heuristic Evaluation

Three scenarios have been simulated and compared, to investigate the effectiveness of the proposed heuristic:

1. The Smart or Network-aware scenario. A DDC with the proposed scheduler.

2. The Round-Robin scenario. A DDC with a scheduler following the Round-Robin approach and choosing the blades with the most available resources. Since this is a scenario running in disaggregation mode, the scheduler considers and satisfies the networking requirements

3. The Traditional scenario. A modern DC with the default scheduling OpenStack algorithm, which implements Round-Robin logic and chooses the server with the most available RAM. The scheduler in this scenario, is not considered as network-aware, since there are no networking requirements to be satisfied.

The amount of computing resources available and the network configuration are the same in all simulations to ensure fairness and to properly evaluate the results. We have considered a DC with 900 devices split into 30 racks. The DC size is based on two factors: 1)we have data for VM arrival model for small sized DCs, 2) to run the simulation in a sensible time. A time unit is equal to a minute.

In Traditional scenario, each of the 900 servers has 8 cores, 64 GB of RAM, one 1 G Ethernet card to connect to the generic backplane and one 40 G network card to connect to the fast backplane. The oversubscription ratio for CPUs is 4, while it is 1.5 for RAM. That means that the scheduler my assign up to 32 cores of CPU and 96 GB of RAM.

In the Smart And Round-Robin scenarios, this amount of computing power translates into 600 CPU blades and 300 RAM blades. Each CPU blade has 12 cores, 19 GB of RAM, and is similarly connected to both backplanes. The RAM blades are equipped with 153 GB RAM and one 40 G network card and they are only connected in the fast backplane. Based on the same oversubscription ratio, the scheduler is able to assign up to 48 cores and 28.5 GB RAM in CPU blades, and 230 GB of memory in RAM blades. The remote-to-local ratio is set to 4, so a CPU blade can take up to 115 GB RAM.

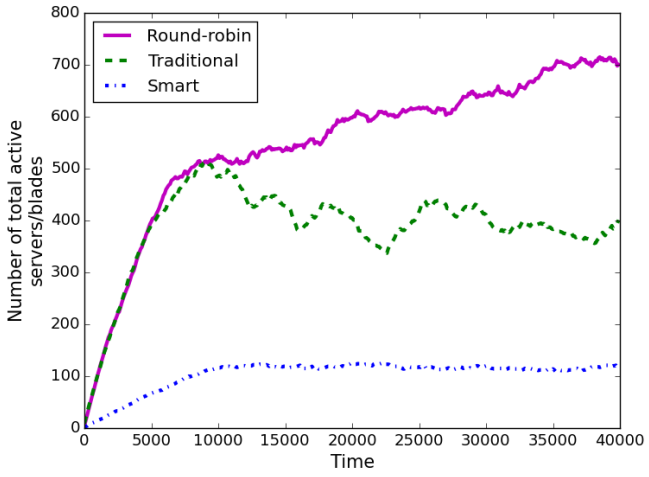


Figure 4.3: Active CPU blades / servers

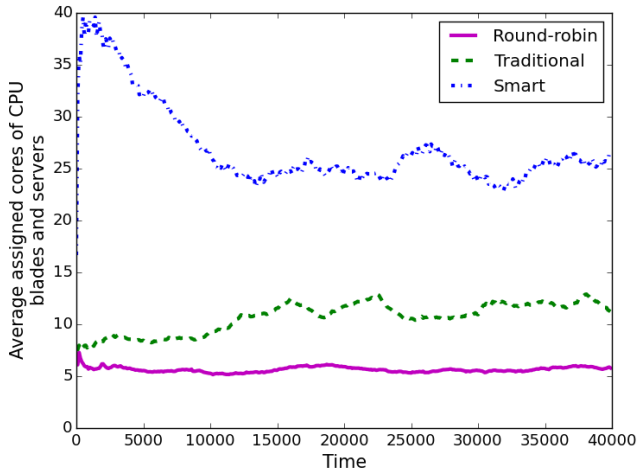


Figure 4.4: Average assigned cores per active CPU blade/server

The simulation results clearly demonstrate the benefits of the proposed scheduler in a DDC in terms of active blades and average host utilization. With the word *active*, we describe the blades, servers or switches that have resources assigned to one or more VMs. As shown in Figure 4.3, at the end of the simulation in its stable state, the proposed scheduler uses approximately 100 - 120 blades to serve the given load. On the other hand, in the Traditional scenario the number of the utilized servers is within the range of 320-420, representing an average increase of 260%, which is significant. Additionally, the total number of active servers and blades increases even further in the Round-Robin scenario, eventually approaching 700 blades. This reinforces our argument in section3.5 that existing scheduling algorithms are inefficient in a DDC.

The lower number of active blades in the Smart scenario, is the result of the increase in the utilization of each blade. Therefore, in Figure 4.4, it is possible to infer that the proposed scheduler is capable of allocating approximately 3-4 times the number of cores per blade/server comparing to the schedulers in both the Traditional and Round-Robin scenarios, respectively. The actual CPU utilization, in terms of CPU cycles as is measured by monitoring tools, may vary according to the application that each VM is running. However, the higher the quantity of assigned cores, the higher the CPU utilization.

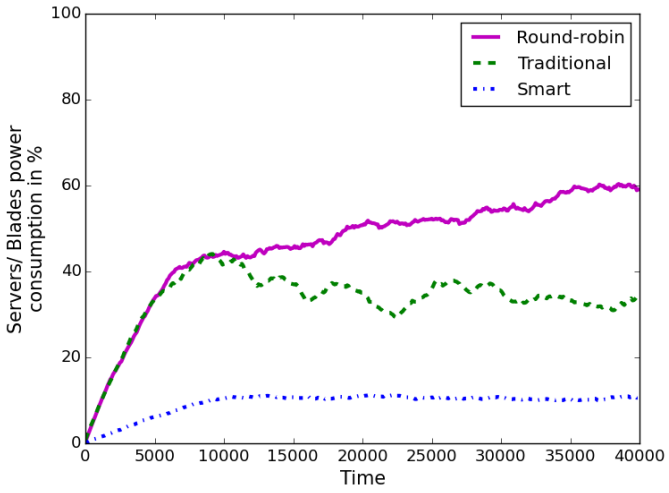


Figure 4.5: Percentage of maximum power consumed for compute resources

It should be noted that the percentage of the assigned cores per server/blade includes the oversubscription ratio. Hence, it is measured on the number of virtual cores that each CPU blade/server can provide, which is 48 for the Smart and Round-Robin scenarios and 36 for the Traditional scenario. Taking this into consideration, we see in Figure 4.4 that in a stable state, the number of assigned cores is slightly above 25% of the total number of virtual cores. In that case, if all the VMs request the entirety of their assigned computing power, there would be a performance degradation and an SLA violation. However, the possibility of this event occurring is relatively low and the system operator can modify this behaviour by adjusting the weights of the scheduler or the oversubscription ratio.

The number of active devices and their utilization directly reflect their energy consumption. We have performed an energy study for compute and network devices, based on Eq. 3.1 and 3.3. One of the major assumptions for our energy figures, is that blades without allocated resource are deactivated, thus, they are not consuming energy. Off course, the activation time required to start up a resource may be significant, but since this is done during the provisioning of the VM, it

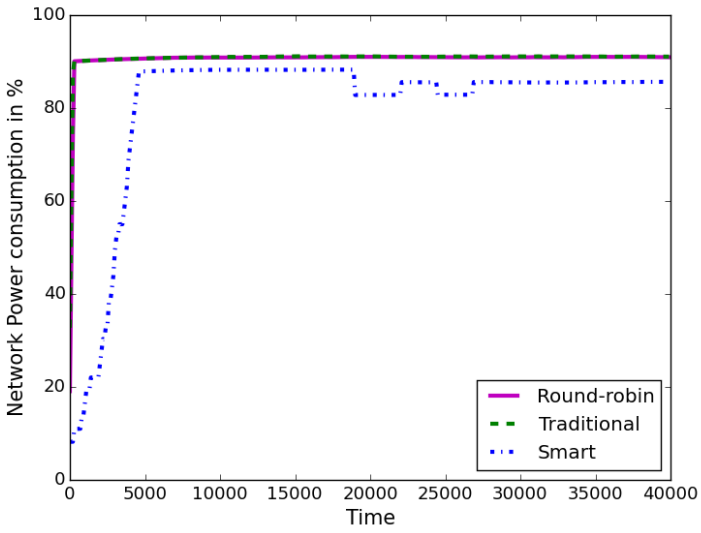


Figure 4.6: Percentage of maximum power consumed for network resources

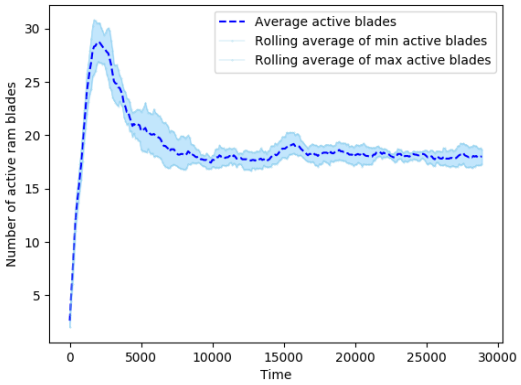
will affect only user experience and not application performance.

The energy consumption comparison between each of the three scenarios is depicted in Figure 4.5 and includes the computing resources (i.e. the total number of blades or servers). The proposed scheduler managed to lower energy consumption to just 30% of the energy consumption on the Traditional scenario, which is a significant saving. Additionally, in Figure 4.6, the results show that in the Smart scenario, we managed to have minor savings in the energy consumed in the network. The important finding though, is that the scheduler managed to adapt to the load variation and reduce the network energy consumption, when needed.

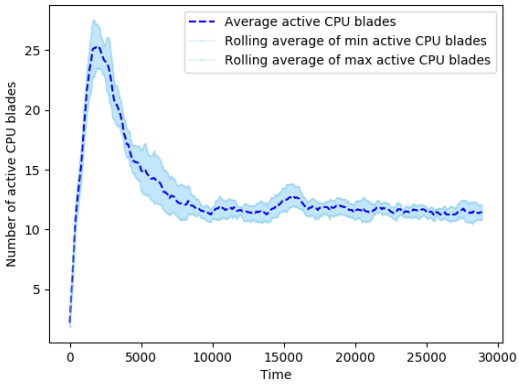
4.2.4 Heuristic and Simulation Stability

In order to evaluate the stability of our simulation and its ability to deliver consistent results, we run the simulation another 20 times only for the SMART scheduler with a downsized version of the problem. Specifically, the simulation run for a shorter period of time and there were only 9 racks hosting 20 blades each. Clusters consists of single rack and the RTL ratio is set to 4.

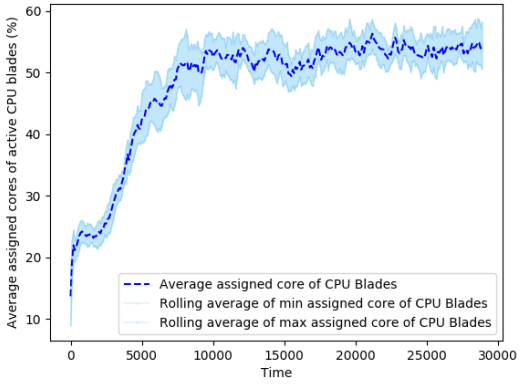
The results in Figure 4.7 showcase that the heuristic is consistently following a trend and the deviation from the mean value is small. This deviation is derived mainly from the VM request, that follow a distribution but are not the same in each simulation run.



(a) Active CPU and RAM blades



(b) Active CPU Blades



(c) Average assigned CPU cores of active blades

Figure 4.7: Simulation results that represents the mean, min and max of each metric after 20 iterations.

4.2.5 Comparative analysis

Comparing to related work in DDC resource allocation, we managed to achieve up to 70% energy consumption savings during the stable state. On the other hand, related work, as presented in section 3.4 showcased a maximum of 42% savings for memory intensive workloads using the heuristic. Comparing to the MILP models, the proposed scheduler achieved 20% lower savings. However, the MILP model is based on the assumption that VMs have an infinite duration and that would result in higher power consumption in our case as well. Additionally, Ali *et al.*, did not provide enough data on VM arrival model, so I could not recreate it in order to have a more accurate comparison.

The main difference though between the two approaches is that the achieved those results

following a greedy approach, while the scheduler is following a more balanced approach in order to avoid exhaustive use of resources, as shown in the results.

4.3 Remote-To-Local ratio and Cluster Size effects on resource utilisation

Even when a DDC operator has concluded on the architecture of their data centre, there are still some parameters, that need fine tuning. In the proposed DDC architecture, two of the most important parameters are the amount of local memory and the number servers in a cluster.

4.3.1 Remote and Local Memory

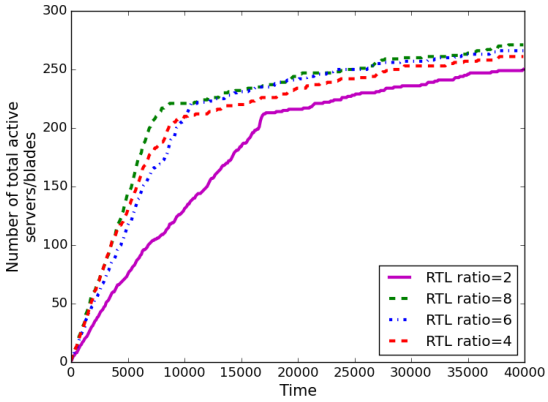
Han *et al.* [2], concluded that as local memory present in the server decreases, there is an increasing burden in the performance of the application running in the DDC. However, after a certain point, there is no further performance degradation. In this paper, we investigate whether there is a relationship between the amount of local memory in CPU blades and the resulted utilization and energy consumption of the resources.

Remote to Local Ratio	Local RAM	Remote MB
2	32 GB	128 GB
4	19 GB	153 GB
6	13 GB	164 GB
8	11 GB	170 GB

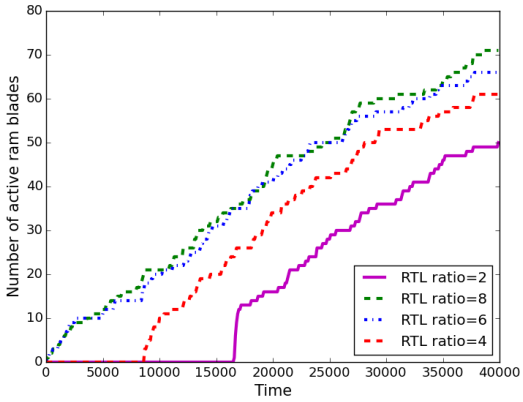
Table 4.2: Local and Remote memory values

To study this problem, we scaled down our simulation setup to 300 devices split equally into 15 racks. However, we maintain the same configuration for the servers and network. As with the previous experiment, the amount of aggregated computing resources is the same throughout the experiment. Hence, we alter the remote-to-local (RTL) memory ratio, while maintaining the overall amount of RAM. Table 4.2 , shows the respective values for the ratio and the resulted memory amount in CPU and RAM blades. We have proven that our proposed scheduler outperforms existing approaches, and hence, is being used for the rest of the simulations.

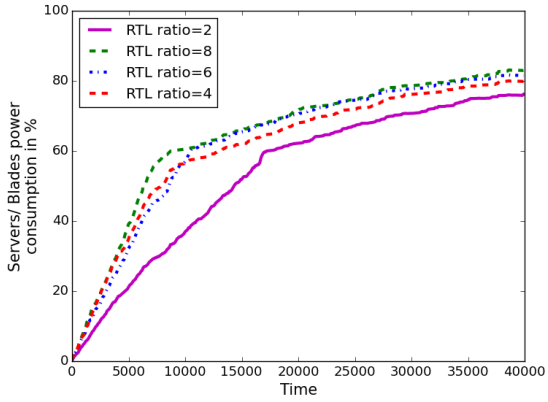
In Figure 4.8(a), it can be observed that increasing the RTL ratio from 2 to 8 slightly increases the number of active blades by 10 to 50, depending on the load of the DC. This is justified, because the scheduler is trying to optimise the overall energy consumption due to the applied policy. CPU blades have more RAM locally and they can individually host more applications. Therefore, the



(a) Active CPU and RAM blades



(b) Active RAM Blades



(c) Percentage of maximum power consumed by the blades

Figure 4.8: Simulation results for different RTL ratio.

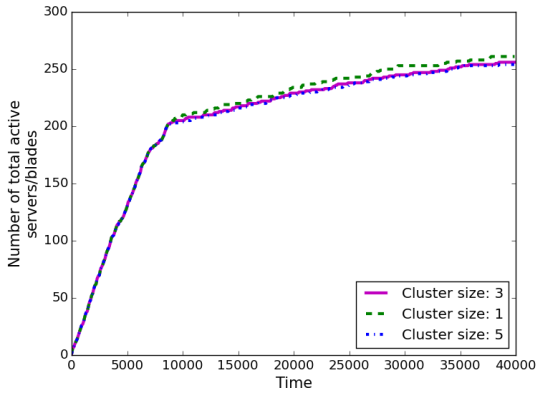
number of activated RAM blades are reduced in comparison. By inspecting Figure 4.8(a) and 4.8(b) in more detail, it is obvious that the difference in the number of active blades is a direct result of the number of active RAM blades. Furthermore, the scheduler manages to achieve its goal of reducing the overall energy consumption, as shown in Figure 4.8(c). Configuring a DC with RTL = 2, which may result to ~10% energy savings under heavy load compared to the configuration with RTL = 8. It should be noted, that reducing the RTL ratio (having more local RAM) is a factor to decrease the overall energy consumption. However, following such an approach results in providing a more *aggregated* DC with plenty of local RAM, and losing all the other benefits that come with disaggregation.

We would like to point out that in this paper, we are not questioning the effects that this might have in the performance of the applications. However, in the case that the research community or

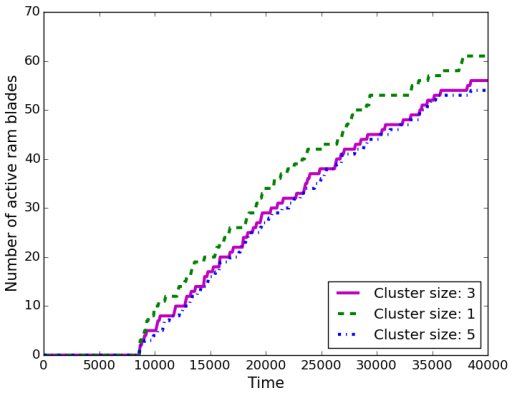
the industry define the RTL ratio that will not affect application performance, the effects and benefits on resource utilization are presented above.

4.3.2 Cluster Size

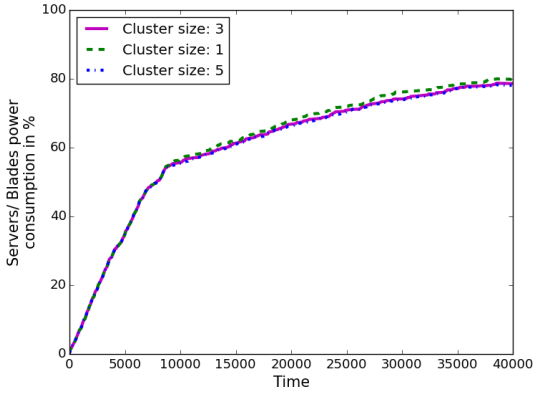
In the proposed architecture, we have presented the option to create clusters of computing resources with dedicated fast backplane. The size of the cluster entirely depends on the capabilities of the optical networking technologies to deliver the required bandwidth and latency between CPU and RAM blades. By utilizing classical optical technologies, the cluster size can range from 1-5 racks. The size was calculated by measuring the link length from each blade to the switch (1-4 m optimally) and the port-to-port delay of an OCS switch, namely 5 ns, resulting in 50 ns round-trip latency. With the use of hollow core fibres, the cluster can be increased to 6 racks, or the latency reduced whilst maintaining equivalent lengths.



(a) Active CPU and RAM blades



(b) Active RAM Blades



(c) Percentage of maximum power consumed by the blades

Figure 4.9: Simulation results with varying cluster size.

We have conducted the same utilization and energy study to investigate the effects of the cluster size on the resource utilization and the energy consumption of the DC. Apart from the cluster size, all other configuration options have remained the same as in the original simulation in section 4.2.3. The results illustrated in Figure 4.9 suggest there will be no further improvement in the resource utilization and the energy consumption of the DDC. Therefore, there is no obvious reason to increase the number of CPU or RAM blades in a single cluster. However, it should be pointed out that keeping the optical links short improves the RAM access time and the overall application performance.

4.4 Discussion

In this chapter , the author addresses the resource allocation problem in a DDC by specifically designing a scheduling heuristic. The simulation results show that the proposed scheduling heuristic can significantly improve resource utilization compared to state-of-the-art heuristic and thus reduce the energy consumption. The author demonstrated that applying existing approaches for scheduling to a disaggregated data centre will have negative effects on the hardware utilization and its cost efficiency.

Two of the parameters that a DDC owner can change during the design of their DDC, is the number of blades in a single cluster and the ratio between remote and local memory. The results suggest that the size of the cluster does not have any effect on energy consumption. As for the amount of local memory, it is possible to achieve minor savings in energy, if the DDC owner increase local memory, thus, returning to a more monolithic approach.

CHAPTER 5

Disaggregated Data Centre Resource Orchestration

Orchestration in a data centre, refers to a set of tools that enables end-to-end management of the offered solutions and processes. The mission of an orchestration software is to manage the entire life-cycle of infrastructure and applications within a DC. This chapter examines the orchestration software in a DDC and focuses on monitoring and migration techniques in order to optimize resource utilisation. The implementation of the orchestration software is discussed within the scope of monitoring a DDC and migrating VMs to improve utilisation and the results are discussed. Finally, it investigates whether variations in the workload require different cluster set-ups.

5.1 Orchestration in DDCs

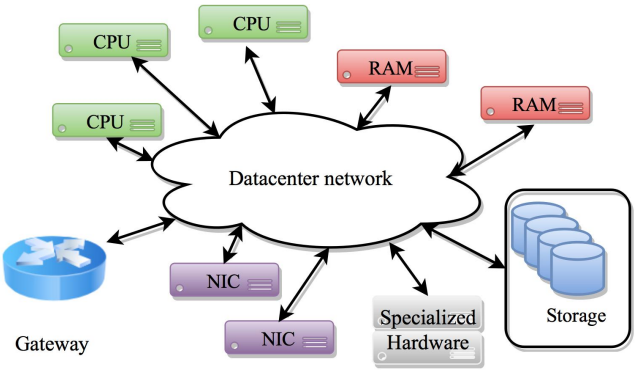
Current DCs are growing larger every year, hence, it is a definite need to have a unified and holistic resource management to avoid conflicts and optimize the efficiency of the DC, that is programmatically controlled. The operation, know as orchestration, is a solution that DCs have implemented in order to provide to end-users a automated way to manage their resource and simultaneously enable DC operators to centrally control the infrastructure. This chapter will study the orchestration software in a DDC and its utility from the operators perspective.

The orchestration service or orchestrator, coordinates the overall provisioning process and is enabled by several modules. Similar to the OpenStack [108] architecture, that we are using as a reference, in the DDC case, every VM request from the user is submitted to the orchestrator. The request describes a single VM or a set of VMs that need to be interconnected. The orchestrator, after acknowledging the request passes it on to the scheduling software, which as analysed in Chapter 4, takes an one-off decision to provision the VM in a set of resource blades.

Following-up the VM provisioning, the orchestrator is responsible for the life-cycle management of the application. Therefore, the orchestration service in cooperation with the monitoring service should track application performance and globally optimize the resource utilisation of the DDC. To achieve this goal, the orchestrator might need to migrate the VM to a different set of resources, and the scheduler provisions the VM once again.

An orchestrator is a powerful tool for the DDC operator in order to improve energy consumption and reduce overall required physical machines in a DDC. In Chapter 4, it was demonstrated that with the proper scheduling algorithm, energy consumption for compute resource in a DDC can be a third of the energy that a traditional DC would consume in a similar case. The reason why the orchestration layer is required, is its ability to monitor the infrastructure and at any time trigger processes that will globally benefit the resource utilization. On the other hand, although the scheduler may be able to determine energy policies and improve efficiency on its own, the nature of its functionality does not allow to intervene in the resource optimizations processes, and thus is limited to one-off decisions.

Taking a step further into investigating the energy consumption problem in DDC, the author designed an orchestrator for a DDC that will continuously monitor the infrastructure and could trigger the VM migration in order to improve efficiency, with performance always as a top priority.



(a)

5.2 Proposed Orchestrator

In this section an orchestration heuristic is presented, that is designed to enable DDC operators to control underlying infrastructure and improve energy consumption. Before going into details about the heuristic, let me define the problem under consideration.

We consider a DDC architecture with a set of resource blades B, arranged in a number of clusters Cl, hosting a number of VM requests N. Each resource blade has a specific amount of CPU, RAM and network bandwidth that can allocate to a VM request, B_{CPU_i},B_{RAM_i} and B_{BW_i} respectively. Additionally, the energy consumption of an active blade is calculated based on Eq. 3.3 and is represented by E_{B_i}. The optimization problem at hand consists of minimizing the overall energy consumption of the DDC. Since most of the energy consumed in a resource is in idle state, the optimisation problem could be transformed to minimizing the number of active blades required for the certain workload of VMs N.

That problem is further reduced to minimizing the number of active clusters. We are taking this decision for several reasons. As mentioned in Chapter 4 section 4.2.1, clusters are a key in the proposed architecture since they guarantee that bandwidth and delay requirements for CPU-to-RAM communication will be satisfied, according to Table 2.1. That is the most critical parameter in a DDC. Therefore in order to be certain that this parameter is preserved, we will be moving the entire application in a single cluster. Additionally, the dynamics of resource provisioning in a DDC, are completely different than in a normal DC. A single VM may be provisioned to multiple blades, for instance, one (1) CPU blade, two (2) RAM blades, one (1) NIC blade and so on. Assimilating this fact, we are determined that it is better to migrate the VM in new cluster, where the utilisation of all the resource will have a better balance, instead of staying in a cluster where few of the resources are underutilised. Finally, the complexity of the orchestration problem increases drastically, in a case that a VM could partially be migrated.

The orchestrator should be able to quickly react to infrastructure changes and adapt to achieve its goal. Therefore, the orchestrator continuously monitors, from a high level, the cluster utilisation and dives into to resource blade level when required, to identify cases that would improve the efficiency of the entire infrastructure.

5.2.1 Orchestration Workflow

Having introduced the orchestrator and the principal on which is based, the detailed workflow of resource management is presented. The specific steps, as show in Fig. 2, are as follows:

I. *Monitor cluster utilization*
   The monitoring service holds information about the status, the utilisation and the energy consumption of all compute and network resources of the infrastructure. The orchestrator is creating knowledge on the status of the DDC and is able to react or pro-act to serve its purpose.

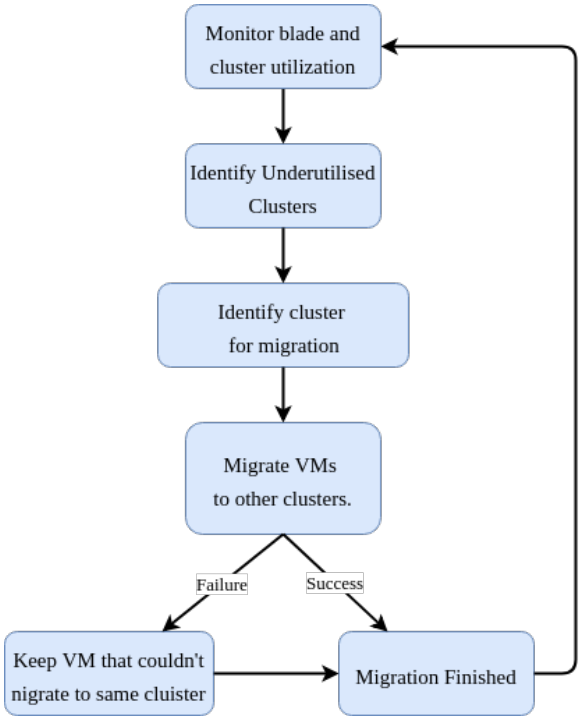
II. *Identify under-utilised clusters*
   Based on the knowledge about the physical infrastructure, the orchestrator is capable of identifying clusters that either their resource are not efficiently utilized or the hosted VMs could potentially be migrated to improve overall efficiency.

   To characterise a cluster as under-utilised 2 conditions have to be met.

   a) The average utilisation is below a predefined threshold Cl_{thres}.
   b) The average utilisation of the active resource blades are below another predefined threshold $Cl_{thres-active}$.

   Thresholds Cl_{thres} and $Cl_{thres-active}$ are defined by DDC operators. In that way, they are able to control how easily the clusters are considered under-utilised. Two different thresholds were incorporated in order to avoid a migration of VMs running on efficiently utilised resource that work smoothly.

   Based on the results and the design of the proposed scheduler in Chapter 4, we argue that there will not be over-utilised clusters and/or blades in the DDC. The scheduler targets to evenly distribute workload on active servers without stressing any of them and activates a new one only when its required. If the load of the DDC increases so much that one server is over-utilised, then that means that there was no other option and all the blades close to be considered over-utilised. In that case, the DC operator should expand the available resources in order to host more applications and manually trigger migration for the most sensitive -in terms of SLA- applications.



(a)

Figure 5.1: Orchestration solution overview

III. *Identify cluster for migration*

With such information, the orchestrator is to decide which cluster to migrate. One cluster is migrated each time because the status of the physical infrastructure will change upon migrating the VMs, so the loop will start over again. In order to decide the cluster that its VMs will be migrated the orchestrator is estimating the power savings and the migration cost. We are stating that the orchestrator is estimating these metrics, because it is not the module that will decide where the VMs will be provisioned - that is fulfilled by the scheduler - so it cannot be precise, and is based in the following assumptions in order to have a point of reference and quickly take the decision.

a) Migration Cost Estimation

According to [141] and [133], the migration cost relates to the time that is required for a VM migration, which in turn is linearly related to the RAM size of the VM instance and the available bandwidth to transfer the data over the network.

As a benchmark, we are based on the work of *Voorsluys et al* [142], who state that for a VM of 2vCPUs and 2 GB RAM, 44 seconds are required hence 45 MBps should be allocated, for our calculations. We are assuming that all the VMs will be migrated. Given that a maximum bandwidth of BW_{mig} will be allocated to migrate all the VMs,

the number of simultaneous migrations is:

$$N_{mig} = \frac{BW_{mig}}{45MBps} \tag{5.1}$$

and the migration time is :

$$mig_{time} = \frac{\frac{\sum_i^n RAM_{VM_i}}{45MBps}}{N_{mig}}, \tag{5.2}$$

where VMs with indexes [i, ..,n] are hosted inside the cluster. By referring to Eq. 3.1, the power overhead of the network during the migration if a number of links (N_{links}) and BW_{mig} are utilised is :

$$P_{mig} = (1-a) * N_{links} * \frac{\sum_i^n RAM_{VM_i}}{BW_{mig}}, \tag{5.3}$$

b) Power Savings Estimation

As already stated, the orchestrator has no control over the provisioning of the VMs we are assuming that all the VMs will be migrated to already active resource blades and not shut down. Although, the metric might not be accurate, we argue that by this assumption we have a quick point of reference for the orchestrator to take a decision. In that scenario, the power savings comes only from the idle consumption of the resource blades that will shutdown, since the load will linearly consume power in the new blades. Thus, it is calculated, according to Eq. 3.1, as follows:

$$P_{savings} = \sum_i^n AB_i * a_i * max\_power_{AB_i}, \tag{5.4}$$

where blades with indexes [i, ..,n] are utilised from the existing workload and are active inside the cluster, and a_i is the percentage of maximum power consumed in idle state from the active blade AB_i

As it can be easily inferred, the power savings from resource blades, will always be greater than the power overhead of the migration over the network. However, another factor that should be taken into consideration, is that a live VM migration is always causing performance degradation of the application and even downtime is some cases [142]. Therefore, the orchestrator should not be lightly trigger the migration of the VMs, and allow the savings to be greater by some factor of the migration power cost before deciding. This factor

Mig_{factor} is defined from the DDC operator according to the workloads running each time on the physical infrastructure. In case that,

$$P_{mig} * Mig_{factor} < P_{savings},$$

the orchestrator is triggering the migration of the cluster.

4. *Migrate VMs*
   The final step is to dive into clusters and select the VMs that can be migrated. As mentioned above, for performance purposes not all VMs is possible to be migrated, either because the migration will inevitably cause a downtime [142], that may result in a SLA violation, or a VM may not be able to fit in a new cluster, because of its requirements. In order to select the VMs, the orchestrator is coordinating closely with the scheduler. The orchestrator should be aware of application sensitive to SLA violation and avoid their migration.

   Additionally, as already mentioned in chapter 4, the scheduler is designed to provision resources, based also on the traffic between VMs. The migration, will be an opportunity to re-examine this relationship. For instance, VMs that are heavily interacting with VMs in other clusters, may be migrated in a single cluster so keeping traffic within the cluster and improving latency may benefit the application's performance.

5.3 Evaluation of the proposed orchestrator

In this section, we quantify the benefits of an holistic resource management in a DDC. In order to evaluate the results of the joint efforts of the scheduling and orchestration software, we run an experiment similar to the one described in Section 4.2, extended with the functionalities offered by the orchestrator. Additionally, we investigate how the results are affected under different types of workload.

5.3.1 Simulation environment

A. *Infrastructure*
   For benchmark purposes, we have set-up the simulation environment identical to the one presented in the previous chapter. We have considered a DDC with 600 CPU resource blades and 300 RAM resource blades split into 30 racks and organised in single-rack clusters. The "fast backplane" is simulated as a single OCS switch per cluster and the "generic backplane" is a hybrid leaf-spine network.

Each CPU blade has 12 cores, 19 GB of RAM, and is connected to both backplanes. The RAM blades are equipped with 153 GB RAM and one 40 G network card and they are only connected in the fast backplane. The oversubscription ratio for CPUs is 4, while it is 1.5 for RAM and the RTL ratio is set to 4. Thus the resource manager is virtually handling 48 cores and 28.5 GB RAM in CPU Blades and 230 GB memory in RAM blades. The additional RAM connected to a CPU blade is a maximum of 155 GB.

B. *VM request arrival model*
We are performing the simulation based on the exact same arrival model that we utilised in chapter 4 for the sake of integrity of the simulation results.

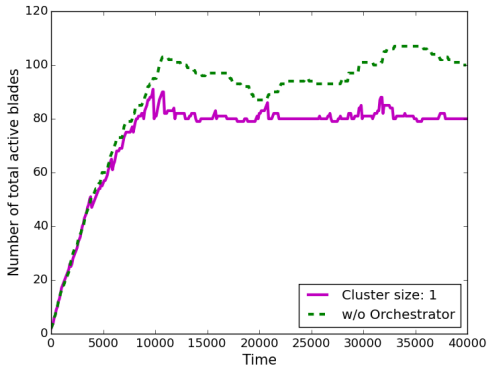
C. *Orchestration Parameters*
During the analysis of the orchestration workflow, we have identified several parameters that a DDC operator is able to define, in order to fine-tune the behaviour of the orchestration software. Cl_{thres} and $Cl_{thres-active}$ are set to 0.3 and 0.4, meaning that if less than 30% of the available resources in a cluster are utilised and the utilisation of the active blades is less than 40%, the cluster will be considered underutilised and will trigger further investigation whether migration is required.

Moreover, in order to avoid interference with existing traffic in the network, $180Mbps$ per link were allocated for VM migration. Referring to 5.1, that translates to 4 VMs concurrently migrating over a single link. The Mig_{factor} is set to 10. Since this is a simulated environment, we are assuming that all VMs could be migrated and there are not restraints by the SLA.

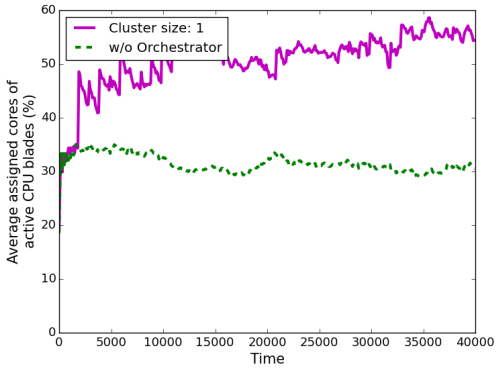
5.3.2 Effects of the orchestration software

The orchestrator is a resource management tool, that extends the abilities of the DDC operator to have control over the infrastructure. To demonstrate our case, and evaluate the performance of the orchestration software, we compare the results of this simulation with the results of the simulation in Section 4.2.3. Figure 5.2 showcases this comparison for computing resources.

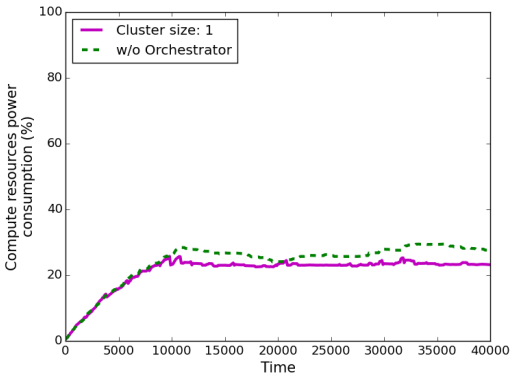
The results clearly demonstrate that we have a significant improvement by utilising the proposed orchestrator in coordination with the scheduler, especially in the number of active resource blades, as show in Fig. 5.2(a). In more detail, the orchestrator managed to lower the number of active blades by up to 30, by utilising more efficiently, already active blades. Hence, the average utilisation of active CPU blades was almost doubled during the time of the simulation (Fig 5.2(b)).



(a) Active CPU and RAM blades



(b) Percentage of assigned cores per CPU blade



(c) Percentage of maximum power consumed for compute resources

Figure 5.2: Comparison of simulation results w/ and w/o the orchestration software.

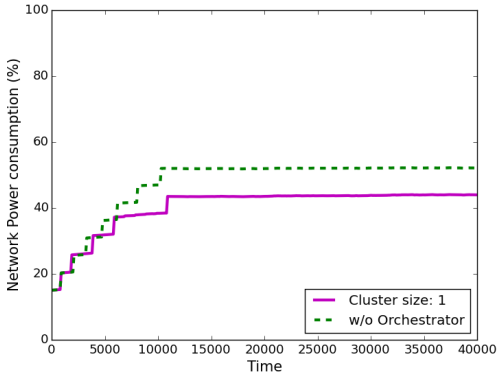


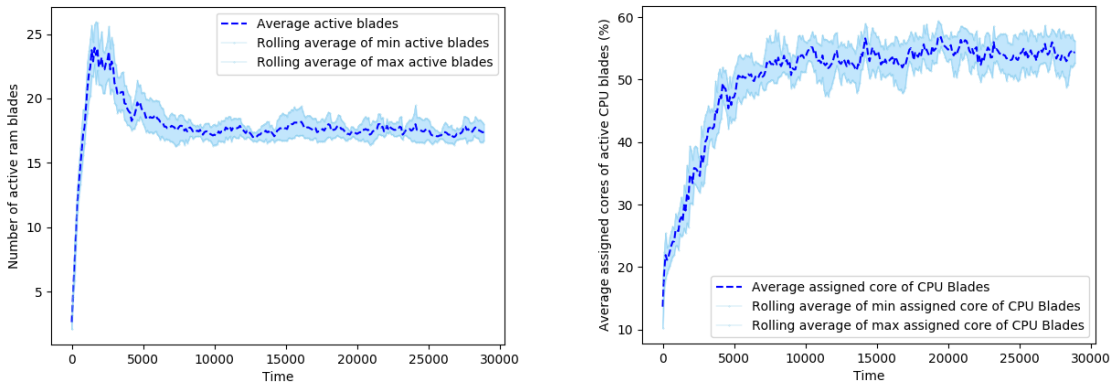
Figure 5.3: Percentage of maximum power consumed for network resources

In terms of the overall power consumption, results in Fig. 5.3 suggest that we achieved 5% on top of the savings that we had due to effective scheduling in DDC, compared to normal DC. In total, the holistic approach in resource management managed to operate the DDC with just 25% of the energy required for the same load on a normal DC. Moreover, in Fig. 5.3, it is apparent that the orchestrator further reduced the network consumption by approximately 10%. This is the result of the reduced number of active blades, that require less networking machines to communicate.

However, the most essential effect of the orchestrator, is that it managed to stabilise the number of the resource blades as the simulation was progressing. As shown in Fig. 5.2(a), after a certain point the number of the required resource blades was fairly steady. That's a huge advantage for DDC operators, since they can predict their needs for physical machines in a longer run, given a certain load.

5.3.2.1 Heuristic and Simulation Stability

Similarly to simulation that run in Chapter 4.2.4, we are performing the experiment with the orchestrator to confirm the consistency of the simulation program. The results state that consistency is achieved and that minimum and maximum values are not fluctuating a lot around the average values. Therefore, we argue that the heuristic is able to provide steady results, as they presented in the previous section.



(a) Active CPU and RAM blades

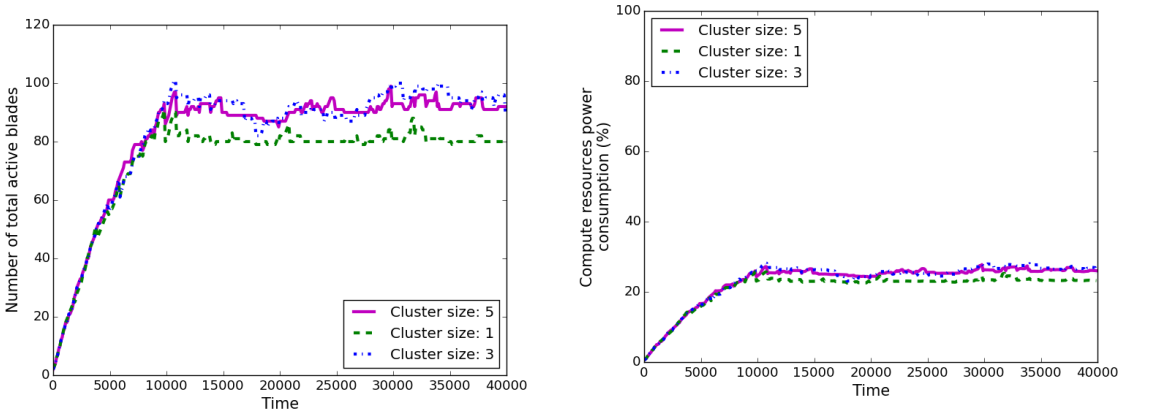
(b) Average assigned CPU cores of active blades

Figure 5.4: Simulation results that represents the mean, min and max of each metric with the orchestrator after 20 iterations.

5.3.3 Cluster size effects

In Section 4.3.2, we have concluded that an increment of the cluster size will not have any obvious benefits for the DC operator in terms of energy consumption and resource utilisation, and will increase intra-cluster link, affecting the application's performance. We conducted the same experiment to examine whether the use of the orchestrator is able to provide value to the DDC operator if the cluster size is varying. In theory, bigger clusters provide the opportunity for a better multiplexing of the RAM and CPU resources, aiming to improve efficiency.

The results in Fig. 5.5 suggest that in terms of energy consumption, there is not any major improvement, so the theory for this workload is not confirmed. However it is self-explanatory that keeping the cluster smaller, allows the orchestrator to have finer control over the resource and achieve better efficiency of the utilisation on the resources, in comparison with bigger clusters.



(a) Active CPU and RAM blades

(b) Percentage of maximum power consumed for compute resources

Figure 5.5: Simulation results with varying cluster size.

5.3.4 Infrastructure behaviour to different workloads

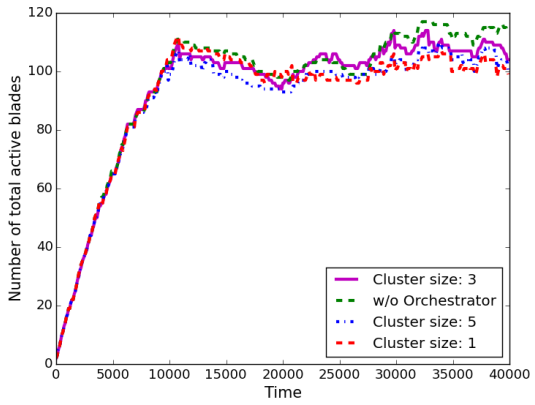
As mentioned before, to run our simulation, we were based on data from DigitalOcean [140] and the work of Peng *et al.* in [139]. However, are creating two extra and different workloads, to examine how the orchestrator would react to CPU-intensive and RAM-intensive applications.

I. CPU Intensive Workload

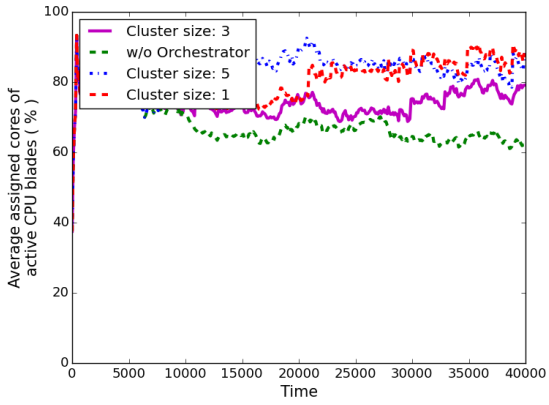
For CPU intensive workloads, the number of cores required by the VMs is doubled. In Fig. 5.6, we present in unified images, the results without the orchestrator and with the orchestrator for varying cluster size. In that way, in a single view we observe that the

orchestrator performs almost the same with the previous simulation in section 5.3.2, and achieved minor improvements in energy consumption, comparing with a scenario that the orchestrator is not utilised.

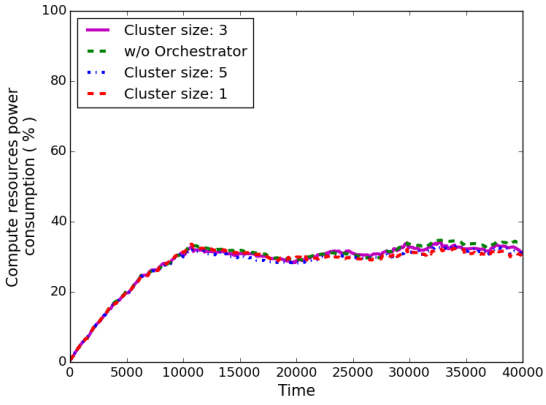
Furthermore, it is observable that a cluster consisting of more racks is actually returns worse figures for a CPU-intensive workload as the number of racks in a cluster increases. The number of active blades is increased by 15 for cluster size 3 and 5 and the energy figures are slightly worsened as well.



(a) Active CPU and RAM blades



(b) Percentage of assigned cores per CPU blade



(c) Percentage of maximum power consumed for compute resources

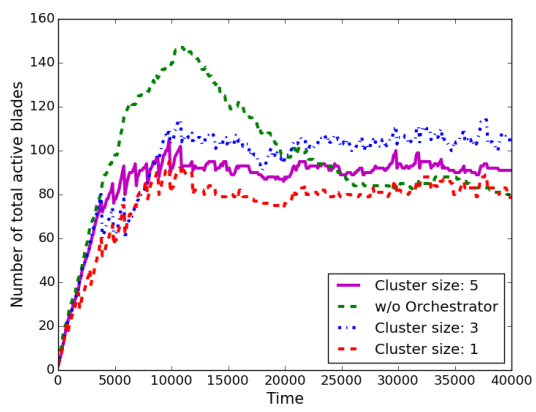
Figure 5.6: Comparison of simulation results w/ and w/o the orchestration software for CPU intensive workloads.

II. RAM Intensive Workload

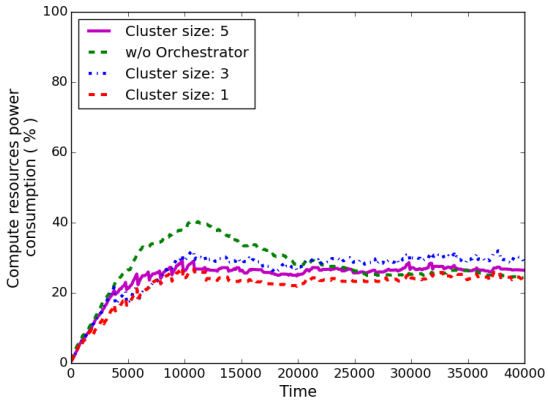
Likewise with the previous experiment, to introduce a RAM intensive workload, the ram requested by a VM is doubled. The utilisation and energy consumption figures in this case

indicate that there are not significant savings in energy consumption and in the number of physical machines required, in the long run. However, we would like to point out in Fig. 5.7(a) and Fig. 5.7(b) that the orchestrator managed to balance the system and achieve a stable situation faster than the scheduler alone. That is important, because in peaks of the workload, the DDC operator will not need extra blades to host the additional traffic.

It is interesting to point out the effects of cluster size in the case of RAM intensive workloads. The simulation results, in Fig. 5.7(a), demonstrate that although with smaller clusters we achieve more efficient control overall, if for some reason the DDC operator decides upon increasing the cluster size, it's preferable to choose bigger cluster size, since that would result in 10% lower needs for physical equipment. This phenomenon occurs, because although the precision of provisioning and control capability over the resources is reduced, the combined efforts of the orchestrator and scheduler, manage to mitigate this disadvantage, take advantage of the multiplexing opportunity and reduce the number of active blades required.



(a) Active CPU and RAM blades



(b) Percentage of maximum power consumed for compute resources

Figure 5.7: Comparison of simulation results w/ and w/o the orchestration software for RAM intensive workloads.

5.4 Discussion

In this chapter , the author is moving a step forward in addressing the resource allocation problem, expand the control capabilities of the operator and explore the possibility of improving the energy consumption in a DDC, by designing an orchestration heuristic. The orchestrator combined with the already presented scheduler are providing and holistic solution to the resource allocation problem.

The simulation clearly demonstrate that is possible to attain additional energy savings on top of what is achievable by the transition from a traditional DC to a DDC, and also predict the required physical resources for a certain type of workload in the long run. That makes it easier for DDC operators to forecast their requirements and CAPEX for compute resources.

Finally, we see that the cluster size factor is of low importance even with the orchestrator for normal VM load and is actually a negative factor in the type of workload change to either RAM intensive or CPU intensive. Aiming to create smaller cluster, will provide shorter links meaning better performance and fine-tuned control of the resource to achieve the energy savings while maintaining the versatility that disaggregation provides.

CHAPTER 6

CONCLUSION

This chapter is summarising and evaluating the contributions of this thesis. The thesis points that were analysed in previous chapters are reviewed and highlighted in order. Thoughts and ideas about future work are being presented.

6.1 DDC architecture contribution

Disaggregated data centres is a new field of research with great interest. There are many factors that will enable this technology and make it successful or not. Advancements in hardware design and new protocols are definitely few of those. However, physics and light speed cannot be overcome. Thus, network design is important not only for resiliency but also for performance. The CPU-to-RAM communication should be kept to kept within the acceptable thresholds, not matter the size of the DDC, that will eventually comprise thousands of blades.

The realisation of a full blown DDC is an open challenge. For future design of disaggregated architecture, one of the most important issues is to preserve and guarantee the communication between core computing components to avoid performance degradation, and optics may be the only solution. On the other hand, the architecture should be able to adapt to dynamic traffic patterns that exist in the Cloud.

My approach was to define an architecture based on currently commercially available equipment. The aim was to keep the network simple, avoid switching overhead as much as possible and keep the link length as short as possible, to reduce latency for the CPU-to-RAM communication. That very special type of communication was decoupled by the rest, so that we can have a quality guarantee. The rest of the traffic has different characteristics that are identical to current DC needs. The network should adaptable, configurable, flexible and scalable.

By organising the blades in clusters, I created a modular and extendable approach, allowing flexibility to DC operator to design a network for intra-cluster communication that will serve the respective needs. Finally, the way that the system is built allows the integration of disaggregated clusters in current DCs, enabling a smooth transition to a new era without huge investments.

6.2 DDC Resource Management contributions

In Chapter 3, I strongly argued about the need to design algorithms specific for DDCs. Existing successful resource allocation algorithms can only provide ideas and indications towards resolving the problem under investigation, due to the DDC's structure, and they cannot be directly applied.

In the second part of this thesis, I proposed the first full software stack for allocating resources to tasks in DDCs. Specifically, I argued that it should be split into two separate services. The first one - the scheduler - should be able to take a quick but fairly accurate decision for VM placement, while the second one - the orchestrator - should monitor the lifetime of the application and continuously optimize resource efficiency and energy consumption.

The proposed scheduling heuristic is the first attempt to balance performance and energy efficiency through a weighed decision function that incorporates network, CPU and RAM resources.

The weights are defined by the DC operator, who has control over the result. Additionally, it can guarantee bandwidth and latency for sensitive DDC communication. The results illustrated 70% of energy savings between a modern DC and a DDC with the proposed scheduler. Both scenarios are based on the assumption that inactive hardware will be deactivated.

In general, the scheduling algorithm should be able to quickly serve and take the best possible decision. In that way future migrations will be avoided. Specifically for DDCs, the scheduler has to be network aware to ensure the steady performance of the system. Most importantly, I showcased in the simulation, what would be the effects of applying common DC scheduling algorithms to DDC infrastructure. The results demonstrated triple power consumption. Therefore, I proved what was speculated in the literature, that DDCs require special resource allocation design.

At the orchestration level, I am assuming the existence of a monitoring scheme that will provide real-time metrics from the infrastructure. Based on that, the orchestration software identifies cases of underutilised resources. I propose that utilisation should be measured both on blade level as well as cluster level in order to reduce problem complexity (divide and conquer). Upon defining the underutilised cluster, the orchestrator may dive into to identify underutilised blades and take the decision whether to migrate the workloads. The placement of the VM is left upon the scheduler. The proposed orchestration algorithm is able to further reduce energy consumption by 5%, due to its ability to identify inefficient cluster utilisation.

The significance of the proposed management stack lies to the fact that the DDC operator has control over the end result. Software configuration may dynamically change for test purposes or workload requirements. VM migrations will always have some downtime, which depending on the application may be a problem or not. Of critical importance, for the resource orchestration, are the cases were circular VM migrations are triggered. This event is occurred, when the system is sensitive to server/blade load, or when the equipment is overloaded and the scheduler keeps pushing VMs to the same piece of hardware. Furthermore, VM migration cost over benefit should be estimated. In a DDC environment, extra caution in required to migrate, wherever the architecture impose, all types of resources in order to keep the performance stable.

The open challenge is to experimentally evaluate a resource allocation stack. That plan relies the actual implementation of disaggregated data centre. To create a DDC, there are several components that needs to be in place. First, the optics should be integrated into servers to avoid electro-optical conversion in transceivers, and there are steps towards that, as we have discussed. Secondly, special network protocols are to be designed in order to provide priority to traffic flows, or reduce complexity and even on the network interface instead on the software stack to reduce blade to blade communication.

An interesting topic for future reaseach, is the integration with Internet of Things (IoT). IoT is a trend that generates enormous amount of sensor data throughout the world. In the proposed

algorithms, we are considering only CPU, RAM and network metrics. However, sensors are able to provide other types that may or may not affect the decision process for the VM placement. Metrics such as room/rack/blade temperature, cooling performance or even humidity, may affect the resource allocation outcome. In such cases Big data technologies would be the enabler to digest all the data and produce valuable result.

Finally, Machine learning and AI applications are gaining traction in the global market. The allocation problem both in DCs and DDCs, solved by AI, is a promising field for future researchers that may make current algorithms obsolete. These algorithms are able to crunch huge amount of data gathered from monitoring software, sensors, external sources and create patterns that the human mind wouldn't be able to recognisee. Based on this patterns they take decisions and are able to evaluate the results in order to get better next time. The problem lying in similar applications is always the "ground truth". Ground truth is the definition of good or bad results, so that the algorithm can learn based on these resutls. The DC operators will be responsible to decide and define the ground truth based on their business and financial priorities.

APPENDIX A

Appendix A : The simulation tool

The simulation software that was used throughout Chapters 4 and 5 is presented in more detail in this appendix.The author have built a simulation tool in order to study and evaluate the effect of resource management in DDCs and compare it with traditional solutions in data centres. The author aims to provide as many details as possible in order future researchers to be able to reproduce and enhance the results. Details are targeted towards the technical side of the implementation rather than the logic behind the schedulers, since that has been extensively analysed in Chapters 4 and 5. The simulation software was written in Python, with MySQL is used to store data of the entities.

A.1 Simulation Entities and Parameters

This section focuses on the structural blocks of the simulation tool, that are necessary to apply the resource management logic.

A.1.1 Infrastructure Entities

I. Rack

Description: A rack is set of hosts. Racks combined together form a cluster at runtime.

II. Host

Description: A host is the basic building block of our simulation, since its resources can be allocated to incoming VM requests. Based on its type, a Host may a CPU blade, a RAM blade or a normal Server.

Properties:

- id: The id of the instance.
- rack: The id of the rack that this device is placed.
- type: The device is either CPU, RAM or None. If type is defined, the host is a CPU or RAM blade respectively, and if it none, the host is a normal Server.
- mem_total: The total amount of memory available in the device. That is number expressed in MBs.
- cores: The number of cores available in the device
- assigned_ram: A positive number representing the amount of assigned memory in MBs.
- assigned_bandwidth: A positive number representing the amount of assigned bandwidth in Mbps.
- remote_memory: A positive number representing the amount of remote memory connected to the device, in MBs. *(relevant only for CPU blade type)*
- assigned_cores: The number of cores that are assigned. *(relevant only for CPU blade type)*
- priority: A positive number set at runtime by the scheduler to define the most appropriate device to place a VM request. Priority can be defined as high or low with a numeric value. Priority values will be explained later in section A.2. *(relevant only for CPU blade type)*

III. Switch

Description: A switch is a device that is represented as a single network interface, that may connect through links to many other interfaces and has the following properties.
Properties:

- id: The id of the instance.
- switching_delay: The switching delay of this switch in seconds.
- type: Optical or Electrical Switch

IV. Network Interface

Description: A network interface of any device.
Properties:

- id: The id of the instance.
- bandwidth: The available bandwidth of the interface
- assigned: A boolean, to define if the network interface is in use.
- device: The Host or the Switch it is related to.
- connected_to: The Host or Switch that it is connected on the other end. *if applicable*

V. Memory Block

Description: Applies only to disaggregated scenarios. Memory from RAM blades is assigned to CPU blades in blocks according to the incoming requests. CPU blade is then responsible to make this extra memory available to the VM.
Properties:

- id: The id of the instance.
- ram_blade: The id of a Host of RAM type, that this partition is related.
- size: The amount of memory of the partition in MBs
- assigned_to: The id of a Host of CPU type that the partition is assigned.

VI. Link

Description: Defines the properties of a Link between two devices. The devices may be Hosts or Switches.
Properties:

- id: The id of the instance.
- endpoints: A list of ids from network interfaces that is connected .

- type: The amount of memory of the partition in MBs
- length: The physical length of the link in meters.
- latency: A number defining the latency in the link in seconds. It is calculated at runtime during the environment set-up. (please refer to A.2.1)
- assigned_to: The id of a Host of CPU type that the partition is assigned.

VII. Optical Link

Description: A special type of Link that connects CPU and RAM blades only. This type of link is used only at the fast backplane (refer to Fig. 2.18)
Properties:

- id: The id of the instance.
- cpu_blade: The id of a Host of CPU type, that the link is connected.
- ram_blade: The id of a Host of RAM type, that the link is connected.
- cpu_network_interface: The id of the interface in the CPU blade that the link is connected.
- ram_network_interface: The id of the interface in the RAM blade that the link is connected.

VIII. Path

Description: A Path is set of links that connects one host to another.
Properties:

- id: The id of the instance.
- switches: A list of switches that are part of the path.
- links: A list of links that consist the path.
- source: The id of the interface in the source Host.
- destination: The id of the interface in the destination Host.
- latency: The aggregated latency along from path from links and switches.
- bandwidth: The available bandwidth along the path in Mbps.

IX. Vm Instance

Description: VM instances are all the different types of VM requests that will be simulated.
Properties:

- id: The id of the instance.

- name: A verbal identifier as in Table 4.1.
- cores: CPU cores required by the instance.
- ram_mb: RAM required by the instance.
- bandwidth_mb: Bandwidth required by the instance.

X. VM Request

Description: The incoming VM request with all the necessary information for its life-cycle in a DC.
Properties:

- id: The id of the instance.
- cpu_blade: The allocated CPU blade id.
- ram_blade: The allocated RAM blade id.
- time: The time of arrival of the request.
- expiration: The point in time that the VM will be terminated.
- target_VM: With which VM does this VM need to communicate.
- vm_communication_path: The allocated path for VM to VM communication.
- status: Active or Inactive.

A.1.2 Monitoring entities

Monitoring entities are used for the reporting of the results at the end of the simulation, as well as by the orchestrator at runtime.

I. CPU Blade Utilization

Description: Time series with utilisation values for CPU blades. The value and the ram_util are calculated as the fraction of the assigned cores and memory and the overall cores memory of the blade multiplied by the respective oversubscription ratios.
Properties:

- blade: The id of the CPU blade.
- timestamp: Timestamp of recording.
- value: Value of utilisation.
- ram_util: The utilisation of embedded RAM in CPU Blades.

II. RAM Blade Utilization

Description: Time series with utilisation values for RAM blades. The value is calculated as the fraction of the assigned memory and the overall memory of the blade multiplied by the oversubscription ratio for RAM.

Properties:

- blade: The id of the RAM blade.
- timestamp: Timestamp of recording.
- value: Value of utilisation.

III. Link Utilization

Description: Time series with utilisation values for Links. The value is calculated as the fraction of the assigned and the maximum bandwidth of the link

Properties:

- link: The id of the Link.
- timestamp: Timestamp of recording.
- value: Value of utilisation.

IV. Link Utilization

Description: Time series with utilisation values for Switches. For simplicity, we assume that a switch is fully utilised if all its likns are fully utilised. Thus, the value is calculated as the average utilisation of all links of the switch.

Properties:

- switch: The id of the Switch.
- timestamp: Timestamp of recording.
- value: Value of utilisation.

A.1.3 Simulation Parameters

In order to start the simulation, the following parameters should be determined by the user, as they affect the end results.

Parameter	**Description/ Comments**
Number of Hosts and their type	The number of hosts in the simulated data centre. Along with number, their type should be defined as well.
Number of Racks	Hosts are equally spread, if possible, to this number of racks.

Cluster Size	How many Racks consist a cluster.
CPU cores /Host	Sets the cores in Host, either it a CPU blade or a Server.
Total RAM	The amount of memory in RAM Blades, CPU blades or Servers (in GB)
Oversubscription Ratio for CPU	Defaults to 4
Oversubscription Ratio for RAM	Defaults to 1.5
Interface bandwidth	There are 4 major groups of interfaces: on the core and TOR switches of the generic backplane (refer to Fig 4.2), on hosts for VM-to-VM communication and on Hosts facing the fast backplane. For each of these interfaces, their characteristics should be defined.
Switch Switching Delay	The switching delay of each switch. Defaults to 250ns for ToR electrical switches and 5ns for OCS switches.
Latency per meter on links	On every link there a propagation delay that relates to its length. The user should define the latency in ns/m. The default value is 5ns/m.
Remote to Local ratio	The ration between the local and remote memory that a CPU blade can accommodate Defaults to 4.
Path selection weights (refers to: Eq 4.1): All weights add up to 1.	
wp_d	The weight for the normalised delay value of a path among candidate paths. Defaults to 0.2.
wp_b	The weight for the normalised available bandwidth of a path among candidate paths. Defaults to 0.6.
wp_h	The weight for the normalised number of hops of a path among candidate paths. Defaults to 0.2.
CPU selection weights (refers to: Eq 4.2): All weights add up to 1.	
wc_p	The weight for the normalised priority value of a CPU blade among candidate CPU blades. Defaults to 0.3.
wc_u	The weight for the normalised utilisation value of a CPU blade among candidate CPU blades. Defaults to 0.4.
wsp_b	The weight for the normalised available bandwidth along a selected path of this candidate CPU blade. Defaults to 0.25.
wsp_d	The weight for the normalised delay value along a selected path of this candidate CPU blade. Defaults to 0.05.
RAM Blade selection weights (refers to: Eq 4.2): All weights add up to 1.	

wr_u	The weight for the normalised utilisation value of a RAM blade among candidate RAM blades. Defaults to 0.7.
wrp_b	The weight for the normalised available bandwidth along the path of this candidate RAM blade to the selected CPU blade. Defaults to 0.2.
wrp_d	The weight for the normalised delay value along the path of this candidate RAM blade to the selected CPU blade. Defaults to 0.1.
Cluster Utilisation Threshold	A number between 0 and 1 that reflects the lower acceptable cluster utilisation and will trigger the procedure for a possible VM migration. Defaults to 0.3.
Blade Utilisation Threshold	A number between 0 and 1 that reflects the lower acceptable blade utilisation and will trigger VM migration if another host is available. Defaults to 0.4.
Mig_factor	This factor defines the sensitivity of the orchestrator when triggering a migration. If the cost savings are not greater by this much from the cost of migration, then the migration is not triggered. Defaults to 10

A.2 Simulation Tool Execution Flow

A.2.1 Environment Set-Up

The most important part of the tool, is the set-up of the environment. It creates and brings together all the necessary pieces. The set-up will create the infrastructure on which the scheduler and the orchestrator will run and take decisions.

I. The first step is to create the VM instances that will be hosted in our environment. For information on instances and the default values, please refer to Table 4.1.

II. Then, the tools creates the VM requests. Based on the work of Peng et al [139], the VM inter-arrival rate follows the Poison distribution with median rate equal to 2hr, and the VMs lifetime follows an almost uniform distribution with media value equal to 80min. For simplicity, we followed the uniform distribution for VMs lifetime calculation. The VM instance that defined the requested resource is picked randomly.

III. The next step is to set up the environment. Based on the parameters already defined, the software created the DB entries for Hosts and network devices. Hosts are distributed to racks and racks are formed into clusters.

Then, it sets up links between the devices according to the design in Fig 4.2, discovers the NxN unidirectional paths between the N hosts of the simulations, and calculates the available bandwidth and latency across the path.

A.2.2 Simulation Execution

With the infrastructure in place, the next step is to run the simulation. The simulation tools picks up all the generated VM requests sorts them based on the time of arrival and takes the following steps:

1) It releases resources that were used by VMs, which have "expired". That includes CPU cores, memory and bandwidth from hosts and bandwidth from links and paths. Please note, that memory blocks that are connected to CPU blades remain as such. That is memory, is considered as a pool of memory dedicated to this specific blade. CPU blades that have enough memory on to host a VM without extra memory, are not taking precedence over other CPU blades that will need extra memory blocks.

2) Executes the respective scheduling scenario as it was presented in Chapter 4.

   If the orchestration software is running as well, it will optimise the efficiency of the infrastructure by migrating the necessary VMs to other hosts, as it was presented in Chapter 5. Empty hosts will be deactivated, allocated memory blocks will be deallocated and all resources will be de-provisioned.

3) Calculates the utilisation of all the devices and their energy figures. Since we have no mean of actually measuring the utilisation of each resource. The tool assumes that all the assigned resources are in use.

BIBLIOGRAPHY

[1] OSA Industry Development Associates (OIDA), "Photonics for Disaggregated Data Centers Workshop," *Optical Future Communications (OFC)*, no. March, 2015.

[2] S. Han, N. Egi, A. Panda, S. Ratnasamy, G. Shi, and S. Shenker, "Network support for resource disaggregation in next-generation datacenters," *HotNets-XII Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, pp. 1–7, 2013.

[3] R. Pries, M. Jarschel, D. Schlosser, M. Klopf, and P. Tran-Gia, "Power Consumption Analysis of Data Center Architectures," in *Green Communications and Networking* (J. J. P. C. Rodrigues, L. Zhou, M. Chen, and A. Kailas, eds.), (Berlin, Heidelberg), pp. 114–124, Springer Berlin Heidelberg, 2012.

[4] "Introducing data center fabric, the next-generation Facebook data center network." Available at: `https://engineering.fb.com/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/`.

[5] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 327–338, 2011.

[6] "Helios: a hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 40, p. 339, 2010.

[7] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A Survey on Data Center Networking (DCN): Infrastructure and Operations," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 1, pp. 640–656, 2017.

[8] X. Ye, Y. Yin, S. B. Yoo, P. Mejia, R. Proietti, and V. Akella, "Dos: A scalable optical switch for datacenters," in *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, p. 24, ACM, 2010.

[9] C. Kachris and I. Tomkos, "A survey on optical interconnects for data centers," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1021–1036, 2012.

[10] "ivx8000 product datasheet. intune networks," 2010.

[11] Y. Yan, G. M. Saridis, Y. Shu, B. R. Rofoee, S. Yan, M. Arslan, T. Bradley, N. V. Wheeler, N. H. Wong, F. Poletti, M. N. Petrovich, D. J. Richardson, S. Poole, G. Zervas, and D. Simeonidou, "All-optical programmable disaggregated data centre network realized by fpga-based switch and interface card," *Journal of Lightwave Technology*, vol. 34, pp. 1925–1932, April 2016.

[12] P. Mahadevan, S. Banerjee, P. Sharma, a. Shah, and P. Ranganathan, "On energy efficiency for enterprise and data center networks," *Communications Magazine, IEEE*, vol. 49, no. August, pp. 94–100, 2011.

[13] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer, "Remedy: Network-aware steady state vm management for data centers," in *Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part I*, IFIP'12, (Berlin, Heidelberg), pp. 190–204, Springer-Verlag, 2012.

[14] R. Villars, J. Koppy, and K. Quinn, "Idc predictions 2013: The new data centre dynamic," *IDC, Framingham, MA, USA*, 2012.

[15] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *ACM SIGCOMM computer communication review*, vol. 39, pp. 51–62, ACM, 2009.

[16] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 75–86, Aug. 2008.

[17] "Microsoft's Cloud Infrastructure: Datacenters and Network Fact Sheet." Available at: `http://download.microsoft.com/download/8/2/9/8297F7C7-AE81-4E99-B1DB-D65A01F7A8EF/Microsoft_Cloud_Infrastructure_Datacenter_and_Network_Fact_Sheet.pdf`.

[18] "Azure global infrastructure." Available at: `https://azure.microsoft.com/en-us/global-infrastructure/`.

[19] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pp. 202–208, ACM, 2009.

[20] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," *Proceedings of the 10th annual conference on Internet measurement - IMC '10*, p. 267, 2010.

[21] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 123–137, ACM, 2015.

[22] D. Ersoz, M. S. Yousif, and C. R. Das, "Characterizing network traffic in a cluster-based, multi-tier data center," in *null*, p. 59, IEEE, 2007.

[23] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 350–361, ACM, 2011.

[24] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*, pp. 65–72, ACM, 2009.

[25] R. Brown, "Report to congress on server and data center energy efficiency: Public law 109-431," *Available at: https://escholarship.org/uc/item/74g2r0vg*, 2008.

[26] R. Branch, H. Tjeerdsma, C. Wilson, R. Hurley, and S. McConnell, "Cloud computing and big data: a review of current service models and hardware perspectives," *Journal of Software Engineering and Applications*, vol. 7, no. 08, p. 686, 2014.

[27] R. Shi, J. Zhang, W. Chu, Q. Bao, X. Jin, C. Gong, Q. Zhu, C. Yu, and S. Rosenberg, "2015 IEEE International Conference on Services Computing MDP and Machine Learning-Based Cost-Optimization of Dynamic Resource Allocation for Network Function Virtualization," 2015.

[28] M. Van Steen and A. S. Tanenbaum, "Distributed systems principles and paradigms," *Network*, vol. 2, p. 28, 2002.

[29] "IBM Knowledge Center: Service-oriented architecture (SOA)." Available at: https://www.ibm.com/support/knowledgecenter/en/SSMQ79_9.5.1/com.ibm.egl.pg.doc/topics/pegl_serv_overview.html.

[30] S. Sakr, A. Liu, D. M. Batista, and M. Alomari, "A survey of large scale data management approaches in cloud environments," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 311–336, 2011.

[31] G. Schulz, *The green and virtual data center*.
Auerbach Publications, 2016.

[32] "Overall Data Center Costs." Available at:https://perspectives.mvdirona.com/2010/09/overall-data-center-costs/.

[33] J. Scaramella, "Worldwide server power and cooling expense 2006-2010 forecast," *Market analysis, IDC Inc*, 2006.

[34] J. Humphreys and J. Scaramella, "The impact of power and cooling on data center infrastructure," *comment id*=, 2006.

[35] A. Shehabi, S. Smith, D. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo, and W. Lintner, "United States Data Center Energy Usage Report ." Available at: `https://eta-publications.lbl.gov/sites/default/files/lbnl-1005775_v2.pdf`, 2016.

[36] Fortune, "The Internet Cloud Has a Dirty Secret."

[37] Y. Cheng, R. Lin, M. De Andrade, L. Wosinska, and J. Chen, "Disaggregated data centers: Challenges and tradeoffs," *IEEE Communications Magazine*, 2019.

[38] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center.," in *NSDI*, vol. 11, pp. 22–22, 2011.

[39] T. Wang, Z. Su, Y. Xia, B. Qin, and M. Hamdi, "Novacube: A low latency torus-based network architecture for data centers," *2014 IEEE Global Communications Conference, GLOBECOM 2014*, pp. 2252–2257, 02 2015.

[40] "Star network." Available at: `https://en.wikipedia.org/wiki/Star_network`.

[41] "Ring network." Available at: `https://en.wikipedia.org/wiki/Ring_network`.

[42] F. K. Liotopoulos *et al.*, "A modular, 160 gbps atm switch architecture for multimedia networking support, based on a 3-stage clos network," in *Proceedings of the International Teletraffic Congress. ITC-16. Teletraffic Engineering in a Competitive World. Edinburgh, UK*, pp. 529–538, 1999.

[43] S. Jadhav, *Advanced computer architecture and computing*.
Technical Publications, 2009.

[44] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[45] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 63–74, Aug. 2008.

[46] "Cisco Data Center Infrastructure 2.5 Design Guide." Available at: `https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCI_SRND_2_5a_book.html`.

[47] K. Bilal, S. U. Khan, and A. Y. Zomaya, "Green data center networks: Challenges and opportunities," in *2013 11th International Conference on Frontiers of Information Technology*, pp. 229–234, Dec 2013.

[48] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: A scalable fault-tolerant layer 2 data center network fabric," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 39–50, Aug. 2009.

[49] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree : Saving Energy in Data Center Networks," *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, pp. 17–17, 2010.

[50] A. Davis, "Photonics and future datacenter networks," in *2010 IEEE Hot Chips 22 Symposium (HCS)*, pp. 1–38, IEEE, 2010.

[51] M. Glick, "Optical interconnects in next generation data centers: An end to end view," in *Optical Interconnects for Future Data Center Networks*, pp. 31–46, Springer, 2013.

[52] C. Kachris and I. Tomkos, "The rise of optical interconnects in data centre networks," in *Proc. 14th Int. Conf. Transparent Opt. Netw.(ICTON)*, pp. 1–4, 2012.

[53] "Vision and Roadmap: Routing Telecom and Data Centers Toward Efficient Energy Use ." Vision and Roadmap Workshop on Routing Telecom and Data Centers, 2009.

[54] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks.," in *Nsdi*, vol. 10, pp. 89–92, 2010.

[55] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "Osa: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 498–511, 2014.

[56] J. J. Berenz, G. W. McIver, and A. E. Lee, "Micro-electro system (mems) switch," May 30 2000.
US Patent 6,069,540.

[57] K. A. McGreer, "Arrayed waveguide gratings for wavelength routing," *IEEE Communications Magazine*, vol. 36, no. 12, pp. 62–68, 1998.

[58] K. Xi, Y. Kao, M. Yang, and H. Chao, "Petabit optical switch for data center networks. polytechnic inst. new york univ., brooklyn," tech. rep., NY, Tech. Rep, 2010.

[59] H. J. Chao, K.-L. Deng, and Z. Jing, "Petastar: a petabit photonic packet switch," *IEEE journal on Selected Areas in Communications*, vol. 21, no. 7, pp. 1096–1112, 2003.

[60] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, "Proteus: A topology malleable data center network," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, (New York, NY, USA), Association for Computing Machinery, 2010.

[61] A. Singla, A. Singh, K. Ramachandran, Lei Xu, and Yueping Zhang, "Feasibility study on topology malleable data center networks (dcn) using optical switching technologies," in *2011 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference*, pp. 1–3, March 2011.

[62] R. Luijten, W. E. Denzel, R. R. Grzybowski, and R. Hemenway, "Optical interconnection networks: The osmosis project," in *The 17th Annual Meeting of the IEEELasers and Electro-Optics Society, 2004. LEOS 2004.*, vol. 2, pp. 563–564, IEEE, 2004.

[63] R. Hemenway, R. Grzybowski, C. Minkenberg, and R. Luijten, "Optical-packet-switched interconnect for supercomputer applications," *Journal of Optical Networking*, vol. 3, no. 12, pp. 900–913, 2004.

[64] J. Gripp, J. Simsarian, J. LeGrange, P. Bernasconi, and D. Neilson, "Photonic terabit routers: The iris project," in *Optical Fiber Communication Conference*, p. OThP3, Optical Society of America, 2010.

[65] "The New Optical Data Center Polatis Data Sheet Polatis Inc.." More info: https://www.polatis.com/, 2009.

[66] F. Poletti, N. Wheeler, M. Petrovich, N. Baddela, E. N. Fokoua, J. Hayes, D. Gray, Z. Li, R. Slavík, and D. Richardson, "Towards high-capacity fibre-optic communications at the speed of light in vacuum," *Nature Photonics*, vol. 7, no. 4, p. 279, 2013.

[67] G. M. Saridis, E. H. Salas, Y. Yan, S. Yan, S. Poole, G. Zervas, and D. Simeonidou, "DORIOS: Demonstration of an All Optical Distributed CPU, Memory, Storage Intra DCN Interconnect," 2015.

[68] P. Costa, H. Ballani, and D. Narayanan, "Rethinking the Network Stack for Rack-scale Computers," *6th USENIX Workshop on Hot Topics in Cloud Computing*, 2014.

[69] S. Novaković, A. Daglis, E. Bugnion, B. Falsafi, and B. Grot, "Scale-Out NUMA c," *Proceedings of ASPLOS-XIX*, 2014.

[70] K. Asanović and D. Paerson, "FireBox : Upcoming Applications," *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST '14)*, 2014.

[71] K. Lim, Y. Turner, J. R. Santos, A. AuYoung, J. Chang, P. Ranganathan, and T. F. Wenisch, "System-level implications of disaggregated memory," *HPCA '12 Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture*, pp. 1–12, 2012.

[72] SeaMicro, "Technology Overview," *http://www.seamicro.com/*.

[73] HP, "Moonshot," *https://www.hpe.com/uk/en/servers/moonshot.html*.

[74] Intel, "Rack Scale Design Overview," *https://www-ssl.intel.com/content/www/us/en/architecture-and-technology/rack-scale-design-overview.html*.

[75] "DMTF's Redfish ." More info: `https://www.dmtf.org/standards/redfish`.

[76] "Rack Scale Design Architecture ." More info: `https://www.intel.com/content/www/us/en/architecture-and-technology/rack-scale-design/rack-scale-design-architecture-white-paper.html`, 2019.

[77] E. R. Ranalli and B. A. Scott, "Wavelength selective switch," Sept. 4 2001. US Patent 6,285,500.

[78] B. Guo, S. Peng, C. Jackson, Y. Yan, Y. Shu, W. Miao, H. Dorren, N. Calabretta, F. Agraz, J. Perelló, S. Spadaro, G. Bernini, R. Monno, N. Ciulli, R. Nejabati, G. Zervas, and D. Simeonidou, "SDN-enabled Programmable Optical Packet/Circuit Switched Intra Data Centre Network," in *Optical Fiber Communication Conference*, p. Th4G.5, Optical Society of America, 2015.

[79] A. Shehabi, S. Smith, D. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo, and W. Lintner, "United states data center energy usage report," 2016.

[80] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[81] "Overall Data Center Costs." Available at:https://perspectives.mvdirona.com/2010/09/overall-data-center-costs/.

[82] OpenStack Conceptual Architecture, "https://docs.openstack.org/install-guide/get-started-conceptual-architecture.html,"

[83] "Overall Data Center Costs." Available at:https://perspectives.mvdirona.com/2010/09/overall-data-center-costs/.

[84] "Overall Data Center Costs." Available at:https://perspectives.mvdirona.com/2010/09/overall-data-center-costs/.

[85] OpenStack Nova Scheduler, "https://docs.openstack.org/nova/latest/user/architecture.html,"

[86] OpenStack Heat, "https://wiki.openstack.org/wiki/Heat,"

[87] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," *Proceedings - IEEE INFOCOM*, 2010.

[88] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware VM placement for cloud systems," *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, pp. 498–506, 2012.

[89] X. Sun, N. Ansari, and R. Wang, "Optimizing Resource Utilization of a Data Center," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 4, pp. 2822–2846, 2016.

[90] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, (New York, NY, USA), pp. 7:1–7:13, ACM, 2012.

[91] K. Su, L. Xu, C. Chen, W. Chen, and Z. Wang, "Affinity and conflict-aware placement of virtual machines in heterogeneous data centers," in *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*, pp. 289–294, March 2015.

[92] J. Hwang, S. Zeng, F. y. Wu, and T. Wood, "Benefits and challenges of managing heterogeneous data centers," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 1060–1065, May 2013.

[93] K. Patel, M. Annavaram, and M. Pedram, "Nfra: Generalized network flow-based resource allocation for hosting centers," *IEEE Transactions on Computers*, vol. 62, pp. 1772–1785, Sept 2013.

[94] E. Rotem, U. C. Weisser, A. Mendelson, A. Yassin, and R. Ginosar, "Energy management of highly dynamic server workloads in an heterogeneous data center," in *2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pp. 1–5, Sept 2014.

[95] S. Zhang, Y. Liu, B. Wang, and R. Zhang, "Analysis and modeling of dynamic capacity provisioning problem for a heterogeneous data center," in *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 785–790, July 2013.

[96] S. Zhang, Y. Liu, B. Wang, and R. Zhang, "A novel resource allocation algorithm for a heterogeneous data center," in *2013 International Conference on Information Science and Applications (ICISA)*, pp. 1–4, June 2013.

[97] "VMWare. ESX Server Performance and Resource Managerment for CPU-Intensive Workloads.." Available at: `http://www.vmware.com/pdf/ESX2_CPU_Performance.pdf`, 2016.

[98] Y. Park, R. Scott, and S. Sechrest, "Virtual memory versus file interfaces for large, memory-intensive scientific applications," in *Proceedings of the 1996 ACM/IEEE Conference on Supercomputing*, Supercomputing '96, (Washington, DC, USA), IEEE Computer Society, 1996.

[99] C.-H. Lee, M. C. Chen, and R.-C. Chang, "Hipec: High performance external virtual memory caching," in *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation*, OSDI '94, (Berkeley, CA, USA), USENIX Association, 1994.

[100] U. Vallamsetty, P. Mohapatra, R. Iyer, and K. Kant, "Improving cache performance of network intensive workloads," in *International Conference on Parallel Processing, 2001.*, pp. 87–94, Sept 2001.

[101] Y. Song, Y. Sun, and W. Shi, "A two-tiered on-demand resource allocation mechanism for vm-based data centers," *IEEE Transactions on Services Computing*, vol. 6, pp. 116–129, First 2013.

[102] H. Qian and D. Medhi, "Data centre resource management with temporal dynamic workload," *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pp. 948–954, 2013.

[103] M. Menarini, F. Seracini, X. Zhang, T. Rosing, and I. Kruger, "Green web services: Improving energy efficiency in data centers via workload predictions," pp. 8–15, May 2013.

[104] L. Liu, J. Xu, H. Yu, L. Li, and C. Qiao, "A novel performance preserving vm splitting and assignment scheme," pp. 4215–4220, June 2014.

[105] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '05, (New York, NY, USA), pp. 303–314, ACM, 2005.

[106] Z. I. M. Yusoh and M. Tang, "A penalty-based grouping genetic algorithm for multiple composite saas components clustering in cloud," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1396–1401, Oct 2012.

[107] I. Cunha, J. Almeida, V. Almeida, and M. Santos, "Self-adaptive capacity management for multi-tier virtualized environments," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 129–138, May 2007.

[108] OpenStack, "https://www.openstack.org/,"

[109] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurrency Computation Practice and Experience*, vol. 24, pp. 1397–1420, 2012.

[110] R. Jansen and P. R. Brenner, "Energy efficient virtual machine allocation in the cloud: An analysis of cloud allocation policies," *2011 International Green Computing Conference and Workshops, IGCC 2011*, pp. 1–8, 2011.

[111] W. Lloyd, S. Pallickara, O. David, M. Arabi, and K. Rojas, "Dynamic Scaling for Service Oriented Applications: Implications of Virtual Machine Placement on IaaS Clouds," *Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E '14)*, pp. 271–276, 2014.

[112] T. Y. Lawson and Z. Dziong, "Economic Framework for Resource Management in Data Centers," 2016.

[113] "Resource Management With VMware DRS." Available at: `https://www.vmware.com/pdf/vmware_drs_wp.pdf`, 2006.

[114] "Microsoft Performance and Resource Optimization." Available at: http://technet.microsoft.com/enus/library/cc917965.aspx.

[115] "HP Process Resource Manager." Available at: http://h71019.www7.hp.com/enterprise/downloads/5982-4359EN-Process-Resource-Manager.pdf, 2016.

[116] "Advanced POWER Virtualization on IBM System P5: Introduction and Configuration." Available at: http://www.redbooks.ibm.com/abstracts/sg247940.html?Open, 2007.

[117] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 1107–1117, June 2013.

[118] T. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif, *et al.*, "Black-box and gray-box strategies for virtual machine migration.," in *NSDI*, vol. 7, pp. 17–17, 2007.

[119] F. Farahnakian, T. Pahikkala, P. Liljeberg, and J. Plosila, "Hierarchical agent-based architecture for resource management in cloud data centers," in *2014 IEEE 7th International Conference on Cloud Computing*, pp. 928–929, June 2014.

[120] J. Heo, X. Zhu, P. Padala, and Z. Wang, "Memory overbooking and dynamic control of xen virtual machines in consolidated environments," in *2009 IFIP/IEEE International Symposium on Integrated Network Management*, pp. 630–637, June 2009.

[121] L. Chen and H. Shen, "Consolidating complementary vms with spatial/temporal-awareness in cloud datacenters," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pp. 1033–1041, April 2014.

[122] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 119–128, May 2007.

[123] H. Goudarzi and M. Pedram, "Hierarchical SLA-Driven Resource Management for Peak Power-Aware and Energy-Efficient Operation of a Cloud Datacenter," *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 222–236, 2016.

[124] O. Popoola and ·. Bernardi Pranggono, "On energy consumption of switch-centric data center networks," *The Journal of Supercomputing*, vol. 74, pp. 334–369, 2018.

[125] M. Gupta and S. Singh, "Greening of the internet," *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '03*, pp. 19–26, 2003.

[126] D. Adami, C. Callegari, S. Giordano, and M. Pagano, "A hybrid multidimensional algorithm for network-aware resource scheduling in clouds and grids," in *2012 IEEE International Conference on Communications (ICC)*, pp. 1297–1301, 2012.

[127] J. Dong, H. Wang, Y. Li, and S. Cheng, "Virtual machine scheduling for improving energy efciency in IaaS cloud," *China Communications*, vol. 11, no. March, pp. 1–12, 2014.

[128] M. Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," *Proceedings - IEEE INFOCOM*, pp. 963–971, 2012.

[129] H. T. Vu and S. Hwang, "A Traffic and Power-aware Algorithm for Virtual Machine Placement in Cloud Data Center," *International Journal of Grid and Distributed Computing*, vol. 7, pp. 1–10, 2014.

[130] B. Martini, D. Adami, M. Gharbaoui, P. Castoldi, L. Donatini, and S. Giordano, "Design and evaluation of SDN-based orchestration system for cloud data centers," *2016 IEEE International Conference on Communications, ICC 2016*, 2016.

[131] J. Buysse, K. Georgakilas, A. Tzanakaki, M. De Leenheer, B. Dhoedt, and C. Develder, "Energy-Efficient Resource-Provisioning Algorithms for Optical Clouds," *Journal of Optical Communications and Networking*, vol. 5, no. 3, p. 226, 2013.

[132] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," *Proceedings - 9th International Conference on Grid and Cloud Computing, GCC 2010*, pp. 87–92, 2010.

[133] H. Maziku and S. Shetty, "Towards a Network Aware VM Migration: Evaluating the Cost of VM Migration In SDN-Based Cloud Computing Network," pp. 114–119, 2016.

[134] R. Cziva, S. Jouet, D. Stapleton, F. P. Tso, and D. P. Pezaros, "Sdn-based virtual machine management for cloud data centers," *IEEE Transactions on Network and Service Management*, vol. 13, pp. 212–225, June 2016.

[135] F. P. Tso, K. Oikonomou, E. Kavvadia, and D. P. Pezaros, "Scalable traffic-aware virtual machine management for cloud data centers," in *Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems*, ICDCS '14, (Washington, DC, USA), pp. 238–247, IEEE Computer Society, 2014.

[136] H. M. M. Ali, T. E. H. El-Gorashi, A. Q. Lawey, and J. M. H. Elmirghani, "Future energy efficient data centers with disaggregated servers," *Journal of Lightwave Technology*, vol. 35, pp. 5361–5380, Dec 2017.

[137] H. M. M. Ali, A. Q. Lawey, T. E. H. El-Gorashi, and J. M. H. Elmirghani, "Energy Efficient Disaggregated Servers for Future Data Centers,"

[138] H. M. Ali, A. M. Al-Salim, A. Q. Lawey, T. El-Gorashi, and J. M. Elmirghani, "Energy efficient resource provisioning with VM migration heuristic for Disaggregated Server design," *International Conference on Transparent Optical Networks*, vol. 2016-Augus, pp. 3–7, 2016.

[139] C. Peng, M. Kim, Z. Zhang, and H. Lei, "VDN: Virtual machine image distribution network for cloud data centers," *Proceedings - IEEE INFOCOM*, pp. 181–189, 2012.

[140] DigitalOcean, "https://www.digitalocean.com/,"

[141] W. Dargie, "Estimation of the cost of VM migration," *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, 2014.

[142] W. Voorsluys, B. J, V. S, and B. R, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," *Cloud Computing. CloudCom 2009*, 2009.