UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDUARDO ROLOFF

Exploiting Cloud Heterogeneity for Cost-Efficient Execution of HPC Applications

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Philippe Olivier Alexandre Navaux
Coadvisor: Prof. Dr. Luciano Paschoal Gaspary

Porto Alegre
November 2019

CIP — CATALOGING-IN-PUBLICATION

Roloff, Eduardo

Exploiting Cloud Heterogeneity for Cost-Efficient Execution of HPC Applications / Eduardo Roloff. – Porto Alegre: PPGC da UFRGS, 2019.

109 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2019. Advisor: Philippe Olivier Alexandre Navaux; Coadvisor: Luciano Paschoal Gaspary.

1. Cloud Computing. 2. HPC. 3. Mechanism. 4. Heterogeneous Environment. I. Navaux, Philippe Olivier Alexandre. II. Gaspary, Luciano Paschoal. III. Título.

"If I have seen farther than others,
it is because I stood on the shoulders of giants."

— Sir Isaac Newton

ACKNOWLEDGMENTS

I would like to thank my long-term advisor, Professor Navaux, for his invaluable support during my research. He supported my research topic even though initially it was digressive to the GPPD overall direction. My co-advisor, Professor Luciano Gaspary, for the discussions and directions adjustments during my work. They have taught me how to be a good researcher.

Next, I would like to thank my GPPD mates. They gave me support in all aspects of academic life.

My employers, Getnet and Banco Topázio, for the support during my academic activities.

Finally, I want to thank my wife, Michele, and my daughters, Helena and Júlia, for their love and for the support they made when my energy was devoted to finishing this work.

ABSTRACT

Using Cloud Computing as an environment for running high-performance applications is already a reality today, where many scientists already use the cloud as their primary execution environment or at least as an environment that can extend a running environment, such as a cluster of computers. However, as cloud providers have a diverse range of instances with many heterogeneous configurations. In terms of processing power, cloud instances range from instances with shared processing cores to large instances focused on running high-performance applications. In terms of price, instances vary from configurations with values under US$ 0.01 per hour up to instances that cost tens of dollars per hour. On the other hand, high-performance applications split into several tasks that are executed in parallel to solve a problem; such tasks end up presenting different computational demands, causing the application to present a certain level of imbalance. The combination of the wide variety of instances offering and the applications that have different demands leads us to a problem, where it is difficult for the user to choose the right instance or instances to run their application. This thesis addresses this problem by conducting a study on cloud heterogeneity, seeking to identify opportunities that can be exploited both in terms of cost reduction and acceptable execution performance, without presenting significant losses in execution time. After this study, we proposed a mechanism called CloudHet that uses as input the cloud instance profile and the task load profile of a given application; Using this input data, our mechanism optimizes to provide the user with an execution scenario that fits their application profile, focusing on cost efficiency. Our results have shown that according to the application behavior, it is possible to achieve significant cost reduction by using heterogeneous configurations compared to homogeneous configurations. We achieved gains in cost reduction up to 63%. Besides, the performance loss keeps a tolerable rate, up to 7%.

Keywords: Cloud Computing. HPC. Mechanism. Heterogeneous Environment.

Explorando a heterogeneidade da nuvem para execução de aplicações HPC com eficiência de custo

RESUMO

O uso de Computação em Nuvem como um ambiente para a execução de aplicações de alto desempenho já é uma realidade atualmente, onde diversos cientistas já utilizam a nuvem como ambiente primário de execução ou, ao menos, como um ambiente que pode estender um ambiente de execução tradicional, tal como um aglomerado de computadores. No entanto, como provedores de nuvem possuem uma oferta diversa de instâncias, com várias configurações heterogêneas tanto em termos de poder de processamento como de custo. Há variação de instâncias com cores de processamento compartilhado até grandes instâncias com foco em execução de aplicações de alto desempenho. Em termos de preço, há variação de instâncias que apresentam valores menores do que um centavo de dólar por hora até instâncias que custam dezenas de dólares pela utilização de uma hora. Por outro lado, as aplicações de alto desempenho são divididas em diversas tarefas que são executadas paralelamente para resolver um problema, tais tarefas acabam apresentando diferentes demandas computacionais, fazendo com que a aplicação apresente um certo nível de desbalanceamento. Juntando-se a grande variedade da oferta de instâncias com aplicações que possuem diferentes demandas, é um problema para o usuário fazer a escolha da instância, ou instâncias, certas para a execução de sua aplicação. Essa tese trata desse problema, realizando um estudo sobre a heterogeneidade da nuvem, buscando identificar oportunidades que podem ser exploradas tanto em termos de redução de custo como buscando um desempenho de execução aceitável, sem apresentar grandes perdas no tempo de execução. Após esse estudo, é proposto um mecanismo chamado CloudHet que utiliza como entradas o perfil das instâncias da nuvem e o perfil de carga das tarefas de uma dada aplicação; utilizando esses dados de entrada o mecanismo faz uma otimização para propor ao usuário um cenário de execução que se adeque ao perfil da sua aplicação, com foco em eficiência de custo. Os resultados mostram que, de acordo com o comportamento da aplicação, é possível obter uma redução significativa de custos usando configurações heterogêneas em comparação com homogêneas. A redução de custos foi de até 63%, com uma perda de desempenho tolerável de até 7%.

Palavras-chave: Computação em nuvem, HPC, Mecanismo, Ambiente Heterogêneo.

LIST OF ABBREVIATIONS AND ACRONYMS

FLOPS	FLoating-point Operations Per Second
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
HPC	High-Performance Computing
IoT	Internet of Things
BSC	Barcelona Supercomputing Center
NASA	National Aeronautics and Space Administration
CERN	European Organization for Nuclear Research
JPL	Jet Propulsion Laboratory
INRIA	Institut National de Recherche en Informatique et en Automatique
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
MPI	Message Passing Interface
DoD	Department of Defense
ILP	Instruction Level Parallelism
ImbBench	Imbalance Benchmark

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

During the 1990s, *clusters* of computers became a popular alternative (BUYYA, 1999; BAKER; FOX; YAU, 1995) within the High-Performance Computing (HPC) community as an alternative environment for supercomputers. Mainly because the processing power of standard machine hardware had significantly evolved, and also because standard configurations have turned significantly cheaper when compared to supercomputers.

The Cloud Computing model (ARMBRUST et al., 2009; MELL; GRANCE, 2011) began to gain notoriety in the early 2010s as an alternative environment for running general-purpose applications. One of the notable benefits of Cloud Computing is the access to computing resources without the need to acquire them, where the user is charged only for the use of resources. Since the first public cloud computing providers began offering their services, they quickly attracted the attention of the HPC community (WALKER, 2008; VECCHIOLA; PANDEY; BUYYA, 2009; STERLING; STARK, 2009; ZHAI et al., 2011; MEHROTRA et al., 2012). Researchers began evaluating these services as an environment where computer clusters could be configured to run scientific applications. Nowadays, Cloud Computing is already a consolidated environment for running HPC applications (COUTINHO et al., 2016; LANGMEAD; NELLORE, 2018; MANGUL et al., 2019), with many performance and cost-efficiency studies demonstrating that Cloud Computing equals and in some cases, even surpasses traditional clusters (SAAD; EL-MAHDY, 2013) due to higher flexibility and lower up-front costs for the hardware.

However, research regarding HPC in the Cloud concentrates primarily on porting applications to the Cloud, evaluating their performance (MOHAMMADI; BAZHIROV, 2018; SCHEUNER; LEITNER, 2019), cost-efficiency (LI et al., 2018; ZHENG et al., 2018), and improving communication performance (CARRENO et al., 2019; ZAHID et al., 2018); Due to that, there are still open research opportunities to explore cloud resource offerings for HPC.

Most clusters built using cloud providers currently use the same instance types, same Virtual Machine (VM) configuration, or use different instance types only in the context of accelerators (such as Graphics Processing Unit (GPUs)) (CRAGO; WALTERS, 2015). There aspects of cloud provisioning that have received less attention. One of them is building a cluster by using different cloud instance types. We refer to such a system as a *heterogeneous cloud*.

A heterogeneous cloud is an attractive solution for the execution of large parallel applications, as these applications typically have different computational demands, with some tasks performing more work than others, thus generating a load imbalance. In such a scenario, application tasks that perform more work can be executed on faster but more expensive instances, while tasks that perform less work can be executed on slower and more cost-efficient instances, leveraging the heterogeneity of cloud instances. This thesis focused on exploring cloud heterogeneity to obtain a cheaper execution environment for the users.

1.1 Problem Definition and Scope

HPC applications are composed of several tasks that combine to solve a problem. All application tasks should have the same computational demand, so we get an application with an optimal load balance. Load balance is crucial for optimal resource utilization. However, applications often present a level of load imbalance that causes performance degradation and resource idleness (BONETI et al., 2008; PEARCE et al., 2012). On the other hand, the cloud presents a large set of instance types, with a significant variation in terms of price and performance.

Due to this, it is troublesome for the user to understand all the entire cloud offers and choose the most suitable instance, or instances, to execute his application (SAMREEN et al., 2016; ZHANG et al., 2012; GUO et al., 2014; NAIR et al., 2010). We addressed this problem by creating an automated recommendation mechanism that uses the profile of cloud instances and applications demands to recommend a cloud execution environment. Our mechanism creates a single execution environment that matches the processing demands of a given application.

Our proposal aims to create more adequate environments, using cloud instances that better match the applications' task demands. Also, our proposal is compatible with a wide range of applications and requires no changes to the applications or runtime environments. The audience of this work is research scientists investigating new environments for their applications. We also show the impact of changing the focus between shorter execution time and lower cost, which helps cloud tenants to choose their desired tradeoff for different execution scenarios.

1.2 Objectives

HPC applications are typically programmed using MPI, where a problem is divided into several tasks that are executed in parallel by using a cluster or supercomputer. As these applications typically have heterogeneous computational demands, with some tasks performing more work than others, a load imbalance situation could occur, leading to idle resources. Usually, traditional clusters have homogeneous nodes.

On the other hand, public cloud providers such as Amazon EC2 and Microsoft Azure provide a large number of cloud instance types with different numbers of cores, processing speeds, and network interconnections, as well as price. It is possible to create a heterogeneous cluster by using a combination of different instances. In such a scenario, tasks that perform more work can be executed on faster but more expensive instances, while tasks that perform less work can be executed on slower and more cost-efficient instances. These two aspects lead us to formulate the following hypothesis, which is the research line of this thesis.

Hypothesis: *The combination of (1) application load imbalance and (2) heterogeneous machines in the Cloud can be exploited to reduce the execution cost without significant performance loss.*

The specific objectives behind our hypothesis are

- Reduction of the execution cost.
- Automate the cloud instances selection to execute a given application.

By achieving these objectives, we are able to prove our hypothesis.

1.3 Contributions

In this thesis, we investigate heterogeneous clouds, focusing on their potential for cost-efficient execution of HPC applications. Our main contributions are the following.

- We perform an in-depth evaluation of heterogeneous clouds with a variety of instance combinations and parallel application behaviors.
- We developed the *ImbBench*, a benchmark that simulates several load imbalances patterns.
- We introduce *Cost-Delay Product (CDP)* as a metric for analyzing the tradeoffs

between cost and performance, inspired by the energy-delay product (EDP) (Laros III et al., 2013).

- We present *CloudHet*, a mechanism to recommend instance choosing based on application behavior and instance characteristics focusing on optimizing the cost-efficiency of the application's execution.

1.4 Text Organization

The remainder of this thesis is organized as follows. In Chapter 2, we concisely present the concepts of High-Performance Computing and Cloud Computing in order to provide the necessary background to the reader to properly comprehend the remainder of our work.

In Chapter 3, we present the most recent works that aim to provide cloud optimizations for the HPC users, becoming state of the art considered in our work. We compared the related works with our proposal, as well. The scope of our work is also presented, with the definition of what we are delivering and the restrictions that apply to our work.

Chapter 4 presents the study of heterogeneity in the cloud; we conducted a study about the cloud instances and how the user could benefit from leveraging heterogeneity. We introduce our proto benchmark *Imbbench* that implements several load imbalance patterns and is used to clarify the concepts of application heterogeneity. We evaluate the instance heterogeneity and how a user could benefit from it by using a real cloud provider, Microsoft Azure.

In Chapter 5, we present our mechanism, named *CloudHet*, that aims to help the user to chose the most suitable environment to execute their HPC applications. We describe the methodology to exploit the instance heterogeneity, the load imbalance of HPC applications is presented and discussed how we could befit from them as well. The CloudHet mechanism is presented in detail and then evaluated by using HPC applications.

Finally, the Chapter 6 concludes this thesis and summarizes the main contributions of our work. The research perspectives identified during the development of this work are presented and well. Closing the chapter, our relevant publications related to this thesis are listed.

2 BACKGROUND

The purpose of this chapter is to present the background concepts needed to follow the rest of this thesis. The concepts of High-Performance Computing (HPC) and Cloud Computing are presented; then, their combination is discussed. The essential works of HPC in the Cloud are presented as well, in order to characterize it as a viable platform for HPC.

The remainder of this Chapter is organized as follows. In Section 2.1, we present aspects of High-Performance Computing and its applications. In Section 2.2, we present the definition of Cloud Computing and the current significant players of public and private Cloud. In Section 2.3, we analyze the presence of Cloud Computing in the TOP500 rank. In Section 2.4, we provide an analysis of the notable works using Cloud Computing for HPC. Finally, in Section 2.5, we conclude and summarize this Chapter.

2.1 High-Performance Computing

Due to processing power requirements, several problems of science, engineering, and industry do not fit into conventional machines, such as notebooks and workstations. Usually, these types of problems have two characteristics that might demand more processing power: problem size and response time.

Many of the problems in this area require more than Gigabytes of memory or Terabytes of disk storage, not fitting to be processed by conventional machines due to persistence capabilities. Furthermore, due to the intensity of processing, sometimes it is also not possible to use regular machines due to time constraints since these problems would take long periods to finish their calculations.

2.1.1 History and Evolution of HPC Hardware

The first supercomputer of the world is considered the *CDC 6600*, released in 1964, which could perform 500 KFLOPS operations per second. In 1976, the *Cray 1* supercomputer was released and initiated a so-called "Cray era" on HPC. The Cray 1 achieved the performance of 133 MFLOPS. In the later years of the 1980 decade, Cray released two other systems that were remarkable: the *Cray X-MP* (1982) achieved 200

MFLOPS, and the *Cray Y-MP* (1988) achieved 333 MFLOPS. A common feature of all HPC systems so far is that they all had the characteristic of being shared memory systems. The performance upgrades were obtained by improving components, such as processor speed or memory size. In other words, the HPC systems relied on single-node computers.

The single-node systems require fast access to memory, although multiple processors working at the same time and accessing the memory create a bottleneck. The principal impediment to increasing the speed of the shared memory systems was the memory contention (BAILEY, 1987; NUMRICH, 1993; LANGGUTH; CAI; SOUROURI, 2019). The alternative model to shared memory is *distributed memory*, where each processor has its memory. Several alternatives were proposed, but they relied on proprietary and, many times, expensive hardware.

The motivation of the memory contention problem combined with the high costs of existing solutions lead to the creation of the *Beowulf Cluster* (RIDGE et al., 1997), which was introduced in 1994 by a group of NASA scientists. Beowulf was a cluster built by using commodity hardware; its configuration consists of 16 Intel 486DX PCs nodes connected with 10 Mb/s Ethernet. It achieved 1 GFLOP/s of performance, and the total costs were around US$ 50,000. Moreover, it can be said that the Beowulf cluster was the initial milestone in adopting computer clusters for HPC. Clusters are now the dominant standard for HPC systems.

All the components of a computing node continued to evolve; there were relevant improvements in the network interconnections, processors with several cores, high-speed memory, and disk storage. However, all the computation was based only on CPUs. In the 2000s, with the development of GPU (HILLIS; STEELE, 1986; STANDARDS; SOCIETY, 2019) and other accelerators, such as FPGA, the nodes of a cluster started to present hybrid configurations. The current supercomputers are mainly hybrid clusters with a large number of nodes interconnected with a high-speed network. Each node has one or more multicore processors with several levels of shared memory components and a certain quantity of GPUs or other accelerators used for specific types of computation.

2.1.2 HPC Software

The software used in HPC could be classified into two main groups: tools and applications. The tools are composed of languages, libraries, among others that are used to develop or execute applications. The applications are the problems that were programmed and will be executed in HPC environments using these tools.

2.1.2.1 HPC Software Tools

Here we present prominent software tools used to develop applications using shared memory, distributed memory, and accelerators.

- **OpenMP**: OpenMP is an Application Programming Interface (API) for C, C++, and Fortran programming languages that supports the Shared Memory model (CHANDRA et al., 2001; CHAPMAN; JOST; PAS, 2008). OpenMP provides a set of compiler directives that are used to spell out the parts of the program that should be parallelized. OpenMP also includes library routines and environment variables that influence the application behavior at execution time. OpenMP specification is maintained by The OpenMP Architecture Review Board (ARB), which is a consortium of major hardware and software vendors, such as AMD, Cray, IBM, Red Hat, and several research laboratories, such as ANL, BSC, NASA, as well. OpenMP is implemented in many Open Source and commercial compilers, such as GCC and ICC.
- **MPI**: Message Passing Interface (MPI) is a standard for data communication in parallel computing (SNIR et al., 1998; GROPP et al., 1999). In the MPI standard, an application is made up of multiple tasks that run in parallel on multiple nodes; such tasks exchange information through messages using calls to MPI functions. MPI provides a complete messaging infrastructure, supporting both synchronous and asynchronous communication, peer-to-peer communication as well as collective communication. The MPI standard is maintained by the MPI Forum [1], an independent organization. Today there are two main Open Source implementations of MPI, *MPICH*, and *OpenMPI*; there are several commercial and proprietary implementations as well.
- **CUDA and OpenACC**: Compute Unified Device Architecture (CUDA), and OpenACC (Open ACCelerators) are the main programming tools for accelerators today. CUDA is an Application Programming Interface (API) developed by NVIDIA to facilitate that programmers could use their graphic cards for parallel computing (SANDERS; KANDROT, 2010; COOK, 2012; GARLAND et al., 2008). CUDA was one of the first main adopted programming tools for accelerators and created

[1]https://www.mpi-forum.org/

the concept of *General Purpose GPU*, where a graphics card is used for general programming, and not only for graphics processing. This can be considered the initial milestone for hybrid computing. However, CUDA programming is not supported by other accelerators (HERDMAN et al., 2012). To solve this, the OpenACC standard was developed to simplify parallel programming of heterogeneous CPU/GPU systems. OpenACC code can be executed on NVidia, AMD, and Intel accelerators (FARBER, 2016).

2.1.2.2 HPC Applications

There were some classification efforts of HPC applications (ASANOVIC et al., 2006), but they rely on the main calculation characteristics of the problems. We used the characterization proposed on the DOE's *The Magellan Final Report on Cloud Computing* (YELICK et al., 2011a) that groups HPC applications in three domains, according to the application behavior. The three groups of applications are Large-scale tightly coupled, Mid-range tightly coupled, and High throughput.

- **Large-scale tightly coupled**: This group is composed of applications where the calculation is distributed among several cores. The scale of these applications frequently achieves the magnitude of thousands of cores. Typically, these applications are programmed by the use of MPI, and thus network performance is crucial. The weather forecast, seismic simulation, and computational fluid dynamics are examples of applications in this group. The most suitable platform to execute these applications is supercomputers.
- **Mid-range tightly coupled**: These applications perform their calculations in a distributed manner as well; however, their scale need is lower than the *Large-scale tightly coupled* group. Event-driven simulation and some kinds of geophysical simulations are examples of applications in this group. The network performance is still an issue for this group, but they are less sensitive to this. These applications execute well on both supercomputers and computer clusters.
- **High throughput**: These applications perform their computation using tasks characterized by a high level of decoupling or even independent tasks, with most of the computing requiring little or no communication. Due to this, the network dependency of this group is small. Bag-of-tasks applications, such as fractal calculations, computational biology, and Monte Carlo simulations, are examples of applications

in this group; the applications related to IoT data analysis could be included in this group as well. These applications usually do not execute on supercomputers, but execute well on computer clusters and Grids (SILVA; CIRNE; BRASILEIRO, 2003).

Using the classification presented above, we can classify applications in any of the three groups, given their characteristics.

2.2 Cloud Computing

The Cloud Computing paradigm was developed by the combination and evolution of *Distributed Computing* (TANENBAUM; STEEN, 2007) and *Virtualization* (BARHAM et al., 2003), with substantial contributions from GRID and Parallel Computing (BUYYA, 1999). There were many attempts to create a universally accepted definition of the Cloud Computing paradigm. Two outstanding efforts were the work of Grid Computing and Distributed Systems Laboratory (BUYYA, 2009) and the initiative from Berkeley University (ARMBRUST et al., 2009). In 2011, NIST published its definition of Cloud Computing (MELL; GRANCE, 2011) consolidating several studies and became widely adopted. It is the definition adopted in this work, as well. This Section describes the definition of Cloud Computing; presents the platforms that are the current state of the art for Private Cloud; and present the leading providers of Public Cloud.

2.2.1 Definition of Cloud Computing

The NIST definition of Cloud Computing is composed of five essential characteristics, three service models, and four implementation models.

2.2.1.1 Essential Characteristics

The five essential characteristics of Cloud Computing are described below.

- **On-demand self-service**: This characteristic defines that the user must be able to allocate a resource, using an automated tool from the provider at any time without the need for interaction with a human provider-side attendant.
- **Broad network access**: States that all provider functionality should be accessi-

ble over the network using standardized mechanisms and protocols that enable use across any platform, such as smartphones and personal computers.

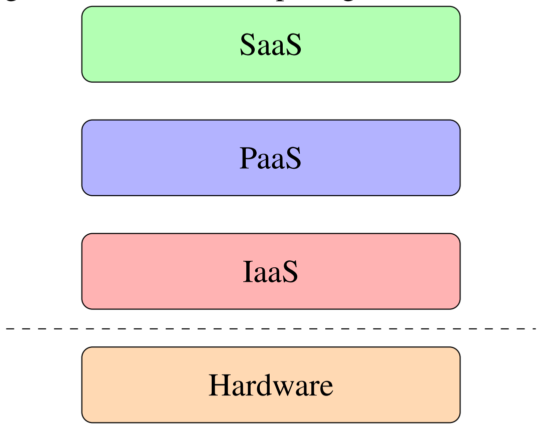
- **Resource pooling**: The resources of the provider are pooled to serve multiple users using a multi-tenant model, with different resources dynamically allocated and reassigned according to the user demand. The user has no control over the location of the resources; however, he or she is typically able to decide the location of the resources (e.g., country, region).
- **Rapid elasticity**: The resources could be provisioned and released flexibly, including automatically, to meet the user requests. From the user point of view, the availability of resources is unlimited, and he or she could allocate them in any quantity at any time.
- **Measured service**: Service Providers need to measure and control resource utilization by users, and then charge them according to their expenditure, similar to services such as electricity and water. This feature makes it possible to achieve the *pay-per-use* billing model.

2.2.1.2 Service Models

The NIST definition proposes three service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These service models define what the user will be offered and also the level of control they will have over features and services. The Figure 2.1 illustrates the abstraction levels of each service model. We settle a hardware layer at the bottom to illustrate that the further from this layer the service is, the more abstract the service will be, and the less control the user will have. Below is a definition of each one of these service models.

- **SaaS**: In this model, ready-to-use applications are available to the user. The user has limited control over the service provided, is limited to some aspects of user configuration and administration. All definitions of processing power supported libraries, and basic software that will be used are entirely from the service provider.
- **PaaS**: This model provides an application development and execution environment for users. In this model, users can run their applications within the definitions made by service providers; in other words, the user has full control over the application. Service providers are responsible for defining the supported programming languages and libraries, as well as the Operating System that is used as the basis for

Figure 2.1: Cloud Computing service models.



execution.

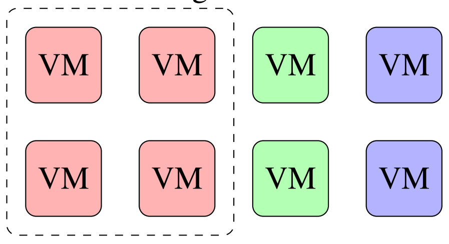
- **IaaS**: It is the model that offers greater control to the user since it provides virtual machines over which users have administrative control. In this model, the user has full control over his virtual machine instance and can choose from its processing capacity (e.g., number of processing cores) to any level of software to be used, including the Operating System.

2.2.1.3 Implementation Models

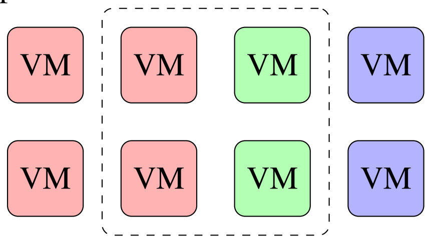
There are four implementation models of Cloud Computing, and we illustrate them in Figure 2.2. To illustrate the possession of the instances, we use color differentiation, where the instances in red represent instances that belong to a public provider, instances in green belongs to an organization, and instances in purple belong to another organization.

- **Private cloud**: A private cloud is a model where the user, usually an organization, has control over the physical resources. This implementation model usually is delivered in two primary forms; an organization owns the computer resources and the facilities. It decides to control them by using a cloud platform (on-premises scenario); or if an organization wants to lease all resources from a provider (outsourced scenario). Figure 2.2c illustrates this model, where a single organization fully controls the green resources. An example of a user of this model is the US Department of Defense (DoD), which contracts two locations of Microsoft Azure for private use.
- **Community cloud**: A Community Cloud could be considered as a derivation of a Private Cloud. The difference is that in this model, cloud resources are shared

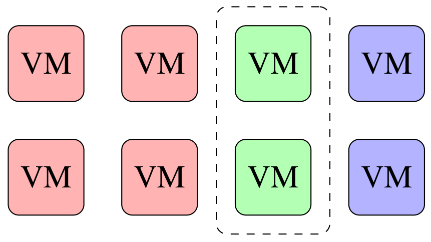
Figure 2.2: Cloud Computing implementation models.



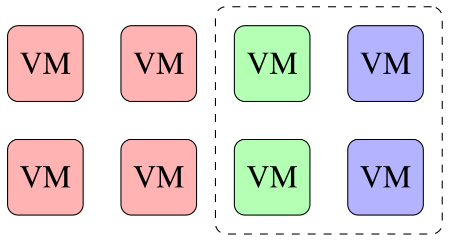
(a) Public



(b) Hybrid.



(c) Private.



(d) Community.

between organizations that participate in the community. Typically, members share common interests, such as different companies belonging to the same economic group or different departments within the same government. There are three main scenarios for deploying this model. The first is when only one community member maintains all resources and provides service to all other members. The second scenario is illustrated in Figure 2.2d, where one organization has resources represented in green, and another has control over resources represented in blue, in which case both organizations share such resources. The third scenario is where resources are rented from a provider and shared by the user community, analogous to the outsourced private scenario. An example of a user of this model is the US Government, which contracts four locations of Microsoft Azure and two locations of Amazon Web Services for private use.

- **Public cloud**: This implementation model is often confused with the definition of Cloud Computing, but this assumption is not correct, as all four are considered valid implementation models by the NIST definition. Resources belong to one provider that shares them among their various users. Typically, service providers are large companies with data centers distributed across the country or around the world. The provider sets their policies and pricing, and users must agree to them in order to access resources. The user of this model can be an entire organization or just one person since it is possible to allocate from just a single machine to a large number of servers. This model is adherent with all essential features of the cloud computing definition.

- **Hybrid cloud**: In theory, a Hybrid Cloud is a combination of two or more cloud installations, whether private, public, or community. A more realistic scenario is an organization that owns a private cloud and connects it to a public cloud provider. In this case, the public part is an extension of the private one. When the organization's dedicated resources are fully utilized, the organization can allocate new resources to the public provider. For this scenario to be possible, the cloud platform used internally by the organization must be compatible with the public provider. This model is often used by an organization that has seasonal demand, for example, retailers during Christmas sales, who do not want to have underutilized resources for most of the year. The Figure 2.2b shows a possible implementation, where the green resources belong to an organization, and they are extended to the public cloud provider that owns the red resources.

2.2.2 Private Cloud Platforms

The implementation of private clouds is not part of the scope of this thesis. However, as cloud management platforms of IaaS are relevant in the context of Cloud Computing, we present a brief introduction to the most relevant platforms today.

- **Open Nebula**: This platform is designed to manage the resources of a data center and create private, public, and hybrid cloud implementations. The service model supported is only the IaaS. The original idea behind OpenNebula is to create a simple platform to build Private Clouds and manage Data Center virtualization. The first release of OpenNebula occurred in 2008; the last stable release was in July 2019. OpenNebula is widely used by the scientific community, with some prominent users, such as BSC, NASA Langley Research Center, and Harvard University.
- **OpenStack**: It is one of the significant public available Cloud platforms. OpenStack started as a joint project of Rackspace and NASA; today, more than 500 organizations joined the project. Its first release was in 2010, and the last stable release was launched in July 2019. Technically, OpenStack is a group of software projects, currently 21, combined to create the platform; each project is responsible for a component of the infrastructure. Commercial companies and the scientific community broadly use OpenStack; some notable users are CERN, NASA JPL, and the University of Cambridge.

Table 2.1: Public Cloud Computing providers.

Company	Commercial Name	Worldwide Locations	Market-Share
Amazon	Amazon Web Services	25	47.8%
Microsoft	Microsoft Azure	54	15.5%
Alibaba	Aliyun	20	7.7%
Google	Google Cloud Platform	24	4.0%
IBM	IBM Cloud	23	1.8%

- **CloudStack**: CloudStack is a cloud platform project of the Apache Software Foundation. It started as a free software project and was then acquired and maintained by Citrix in 2011. Citrix donated CloudStack to the Apache Software Foundation in 2012. The first release of the platform was in 2010, and the last stable release launches in August 2019. CloudStack aims to be a platform that includes a comprehensive set of features to cover all the needs of organizations regarding the implementation of an IaaS cloud. Notable scientific users of CloudStack are INRIA and Melbourne University.

As a side note, due to its historical importance, the Eucalyptus project was one of the first platforms to create private IaaS. Hewlett-Packard acquired Eucalyptus in September 2014. However, due to changes in its strategy, the project lost internal relevance, and the Eucalyptus system eventually lost prominence in the scientific community.

2.2.3 Public Cloud Providers

Since 2006, when Amazon Web Services was launched, several cloud providers started to offer their services. Today we have a large number of cloud providers ranging from regional and niche players up to providers with a global scale. However, when we analyze the market-share numbers of IaaS providers, we could observe that this market is consolidating in a few large and dominant providers (GARTNER, 2019). This situation indicates that the scalability capability of the providers is essential. The five dominant providers are Amazon, Microsoft, Alibaba, Google, and IBM. These five providers represent around 75% of the IaaS market.

The Table 2.1 presents a brief profile of each of the five main providers.

The *Amazon Web Services* (AWS) provider was the first notable provider of public cloud, and it remains as the indisputable leader of IaaS up until today, with nearly 50% of the market share. It is present in 25 locations worldwide, with the majority of the zones configured as availability zones, which means the provider operates more than one data center in the location. AWS provides a full set of services for both IaaS and PaaS, including proprietary solutions developed exclusively for the service, such as the Dynamo DB database.

The *Microsoft Azure* provider is the Microsoft effort in Cloud Computing. It started as a PaaS provider to offer Microsoft's proprietary programming languages, such as the .NET platform, and today is a provider with full services on SaaS, PaaS, and IaaS. In terms of SaaS, Azure provides the whole Microsoft Office suite as a service, among other applications. The PaaS offering was improved, and today a large number of programming languages are supported, including Java and Python. The IaaS service provides several things that are relevant for both academic and industry communities, such as IoT support and artificial intelligence optimized instances. It is the provider with more locations worldwide, with 54 locations in total, possessing 15% of the IaaS market share.

The *Aliyun* is the cloud provider from the Alibaba Group, a Chinese multinational conglomerate holding company with a focus on retail and technology. It is the biggest cloud provider in China, but, since 2015, its services are available internationally. It operates 20 locations worldwide (with 8 in China). In China, its infrastructure comprehends the data center operations and even a proprietary backbone to interconnect all the locations.

The *Google Cloud Platform* is the Google initiative in Cloud Computing. Its primary focus is on application hosting with a comprehensive PaaS solution, but the provider offers IaaS as well. Google's IaaS has a unique capability compared with the other four providers; it is possible to configure customized VM instances where the user could choose the number of computing cores, memory size, and disk capacity.

IBM Cloud is a relatively new player to the public cloud scenario, with a focus on commercial services. The remarkable characteristic of IBM Cloud is the offer of instances with the Power processor; the other providers are based on x86 processors.

2.3 Cloud Computing and TOP500

Since 1993, the TOP500 [2] lists the most powerful computers in the world. The list is compiled twice a year, in June and November. The computers are classified based on the performance results of the LINPACK Benchmark (DONGARRA et al., 1979; DON-

[2]http://www.top500.org/

GARRA, 1987). This list reflects the actual stage of the development and the tendencies of HPC technologies.

As mentioned before, the first prominent provider of Cloud Computing, Amazon Web Services, was launched in 2006. From then on the HPC community started to discuss if, or when, Cloud Computing would be ready to be used for HPC and, eventually, reach the TOP500 rank (NAPPER; BIENTINESI, 2009). The first appearance of Cloud Computing computers in the TOP500 list was in November 2010, where an Amazon EC2 Cluster ranked 233 on the list (TOP500, 2010).

Since then, Cloud Computing has been increasing its participation in the TOP500 list. The Table 2.2 shows the quantity of Cloud Computing related configurations in the last eleven lists. The Cloud configuration that is best ranked and the sum of Teraflops of all the Clouds in the rank are shown as well. We could observe that the participation of the Cloud in the rank is established and constant. It is essential to mention that both configurations of end-users and Cloud providers installations were considered in the TOP500 rank.

2.4 High-Performance Computing in the Cloud

Since its first appearance, the Cloud Computing model has been attracting attention from the High-Performance Computing (HPC) community for the execution of large parallel applications, presenting the potential to extend or even substitute traditional clusters used in research labs (WALKER, 2008; EVANGELINOS; HILL, 2008; NAPPER; BIENTINESI, 2009; VECCHIOLA; PANDEY; BUYYA, 2009; ARMBRUST et

Table 2.2: Cloud Computing configurations in the TOP500 list.

List	Cloud	Best ranked	TFLOPS/s
Jun/2014	3	76	1,115
Nov/2014	16	101	8,502
Jun/2015	11	115	8,727
Nov/2015	12	179	7,697
Jun/2016	38	132	27,090
Nov/2016	51	112	44,556
Jun/2017	61	133	61,734
Nov/2017	45	174	51,204
Jun/2018	42	105	63,674
Nov/2018	41	136	77,236
Jun/2019	57	179	107,793

al., 2009; HE et al., 2010; JACKSON et al., 2010). This section discusses some of the most consistent efforts related to HPC in the Cloud. We could divide these studies into three areas, performance and suitability, optimization, and usability.

Several studies have been conducted in the past years to verify the viability of cloud computing as a suitable platform for HPC (GUPTA et al., 2013a; SADOOGHI et al., 2015; PANDA; LU, 2017). HPC applications are usually composed of processing batches, which need to be handled by the programs or users that run these sets of jobs, which are instances of the application with different inputs, and collect the data after the execution has finished. Based on that, users can decide if more job executions are needed. Therefore, executing HPC applications in the Cloud requires not only attention to resource allocation and optimizations in the infrastructure, but also other aspects, such as how users interact with this environment, what is the cost benefits in comparison with private clusters. The primary studies of HPC viability in the Cloud analyze mainly HPC benchmarks, focusing on CPU, memory, storage, and disk performance. Another effect of the evolution of cloud technologies over type is the availability of HPC-optimized cloud resources. In this category, the Magellan report (YELICK et al., 2011b) is one of the first and most comprehensive studies of HPC in the Cloud, with a wide range of architectures, workloads, and metrics.

On the performance optimization area, studies mainly propose ways to optimize the performance of HPC in the Cloud, with improvements targeted either at infrastructure level or resource management level. Studies on the infrastructure level are mostly focused on studying effects on networking, which is accounted for most of the inefficiencies of executing HPC workloads in the Cloud (MAUCH; KUNZE; HILLENBRAND, 2013). Resource management research focuses on schedule policies aware of both application and platform behaviors, with most studies being focused on exploring hybrid cloud or multiple cloud choices (GUPTA et al., 2013b). In this area, there are important works such as Marathe (MARATHE et al., 2013; AWAD; ARTOLI; AHMED, 2014; IOSUP et al., 2011), which compared a virtualized cloud cluster against a physical cluster. However, the first research works in the area lacked a broad evaluation of public clouds, evaluating a single cloud provider due to availability and costs.

Usability research is another important research area, composed by studies which abstract away the infrastructure from HPC users. The main goal is to create an "HPC as a service" platform where HPC applications are executed in the Cloud, abstracting away the underlining cloud infrastructure. Users submit the application, parameters, and

QoS expectations, such as deadline, via a web portal, and a middleware takes care of provisioning and execution. This category of studies is relevant to increase the adoption of HPC, and also highlights efforts on moving legacy applications to Software-as-a-Service deployments (BELGACEM; CHOPARD, 2015) (CHURCH et al., 2012). To mention a case about the cloud scalability, the work of Posey et al. (POSEY et al., 2018) discusses the challenges and solutions of managing a massive cluster in the Amazon Cloud. The authors achieved a peak of more than a million cores running simultaneously, with around 500,000 jobs executing in two hours.

Another relevant field of study is related to the cost, where two main groups were identified, those with a focus on efficiency and others with a focus on reduction. Cost optimization research focuses mainly on cost reduction and efficiency increase of HPC in the Cloud. The primary studies on this research area are based on comparisons between applications running in the Cloud (KOTAS; NAUGHTON; IMAM, 2018; PRUKKANTRAGORN; TIENTANOPAJAI, 2017). These studies explore mostly HPC benchmarks and scientific HPC applications running on the leading cloud providers (Microsoft, Amazon, and Google) (PRABHAKARAN; LAKSHMI, 2018). Most of them focus on the comparison of homogeneous machines, do not exploring the utilization of the multiple types of instances existent on the different cloud providers when running their evaluations.

At the beginning of this research, we conducted a study of Cloud Computing providers against traditional clusters (Roloff et al., 2017) regarding their performance and cost-efficiency. In this study, we performed an extensive evaluation of two major cloud providers, Amazon EC2 and Microsoft Azure, evaluating their network, CPU, and memory performance. We compared two academic clusters with two cluster configurations in Amazon EC2 and six cluster configurations in Microsoft Azure. We have built clusters with 256 cores each and applied the NAS suite (SAINI; BAILEY, 1996) to measure the computational performance; the STREAM (MCCALPIN, 1995) benchmark for memory evaluation; and the Intel MPI Benchmarks (IMB) for network evaluation.

Our results have shown that performance degradation due to virtualization and other cloud overheads is insignificant. The network performance remained a significant bottleneck for application performance; however, there were significant improvements.

Furthermore, we have found that paying for a more robust cloud does not always improve performance and can even lead to performance reductions. As a general conclusion, we were able to state that Cloud Computing is a viable environment for HPC.

However, it was necessary to know the application behavior to take advantage of the Cloud's characteristics.

2.5 Concluding Remarks

HPC applications, both shared and distributed memory, have the potential to be used in cloud environments, especially applications with less network dependency.

The most suitable Cloud Computing implementation model for HPC is IaaS because, through it, the user has access to VMs to which he has full access and can perform any necessary software level configuration. Using Cloud Computing for HPC is already a reality for many types of applications. In many cases, we realize that it has even some advantages over traditional HPC environments, especially in terms of cost.

The public cloud has also proved to be the implementation where studies should focus, mainly due to the absence of initial setup costs in the pay-per-use model. Major public cloud providers have service offerings that can be exploited for HPC use, and their computer infrastructure currently occupies some positions on the TOP500 list, demonstrating their potential.

3 LEVERAGING CLOUD HETEROGEINITY FOR HPC

Public cloud providers offer a wide range of instance types with different speeds, configurations, and prices, which allows users to choose the most appropriate configurations for their applications. When executing parallel applications that require multiple instances to execute, such as large scientific applications, most users pick an instance type that fits their overall needs best and then creates a cluster of interconnected instances of the same type. However, the tasks of a parallel application often have different demands in terms of CPU usage and memory usage. This difference in demands can be exploited by selecting multiple instance types that are adapted to the demands of the application. This way, the combination of public cloud heterogeneity and application heterogeneity can be exploited in order to reduce the execution cost without significant performance loss.

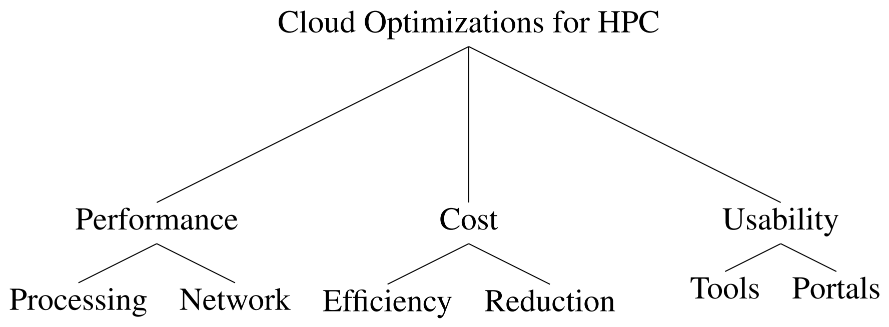
In this chapter, we present the state of the art of works related to optimizations of Cloud Computing for HPC. We present our research line to leverage the Cloud Heterogeinity for HPC as well.

3.1 State of the Art

The use of cloud computing is already a reality today. Several research groups already use the cloud in some way, whether to extend their local clusters, perform single experiments, or even wholly replace their computing infrastructure for cloud environments. However, there is still room for cloud optimizations for HPC in several forms. This section provides a review of significant works in the theme of cloud computing optimization for HPC; they are state of the art on this topic.

During the development of this thesis, we analyze several types of research and organize them according to the classification of Figure 3.1. The works related to performance optimizations are divided between those with a focus on processing and those whose focus is on the network. Another relevant field of study is related to the cost, where two main groups were identified, those with a focus on efficiency and others with a focus on reduction. The other leading group could be classified as usability, where the main focus is on creating tools to help users to use the cloud in a more friendly way. We present the more recent works of cloud optimization, classified into these three groups.

Figure 3.1: Taxonomy of Optimizations of Cloud Computing and HPC.



3.1.1 Performance Optimization

Regarding performance between instances in the cloud, Persico et al. (PERSICO et al., 2015) aim to create a methodology to identify the network throughput and applied it to Microsoft Azure Provider. They performed several tests over time and were able to conclude that the network is stable over time when the instances were deployed and not reallocated. Instances with bigger sizes performed better than the smallest ones as well. They have a second work (PERSICO et al., 2016) where they measured the network performance of instances allocated on other locations, for AWS and Azure providers. They deployed VM instances on five different geographic locations and performed throughput and latency tests. They conclude that the latency is comparable to both providers, and the throughput is around 50% better on Azure.

The work of Mariani et al. (MARIANI et al., 2017) implements a mechanism to predict the cloud performance for HPC applications. They proposed a machine learning algorithm that needs to be trained by using an instrumented application; after that, the model will be capable of predicting the performance of the cloud for a given application. Their model considers as application profile the mix of operations, the ILP (Instruction Level Parallelism), memory re-utilization rate, memory, and register traffic, library calls, and network operations. They validate their proposal by using the NAS suite of benchmarks in a private cloud. Their model achieved an error-level of less than 15% when predicting the performance of the applications running in their cloud.

The work of Li et al. (Li et al., 2019) performed an analysis of how to benefit from the cloud heterogeneity to perform video transcoding in order to maximize the efficiency of video streaming services. They analyzed the performance of several instances types of Amazon AWS, the trade-off between performance and price was studied as well. Their principal findings were that there is a correlation between the video content and the execution time of the transcoding. Moreover, the behavior of the instance is more related to the video content than the transcoding task itself.

3.1.2 Cost Optimization

The work of Kotas et al. (KOTAS; NAUGHTON; IMAM, 2018) has focused on the performance and cost-efficiency of the public cloud. They compared the Amazon AWS and Microsoft Azure as a platform to execute HPC. They chose one instance type of each provider and built several homogeneous cluster configurations varying its size, from 1 to 32 nodes. The workload used was the HPCC and HPCG benchmarks suite; by using them, the authors conducted several performance analysis. Using the results of the performance tests, they provide a cost-efficiency analysis. They found that the AWS instance offers a better performance rate per dollar, considering raw performance, and Azure offers better bandwidth and memory size per dollar. Regarding user usability, they conclude that the user needs to identify which provider is best suitable for a given application.

The work of Prabhakaran and Lakshmi (PRABHAKARAN; LAKSHMI, 2018) evaluate the cost-benefit of using Amazon, Google, and Microsoft cloud instances to execute HPC jobs instead of the SahasraT supercomputing from Indian Institute of Science. They calculated a cost of US$ 0.0126 per core per hour using 100% of the supercomputer uninterruptedly for five years and compared to the cost of the cloud instances. However, since the cloud instances are available in an on-demand base, the authors do not consider the current production utilization rate of the SahasraT, neither the resource utilization when an application is executing. Due to this, the cost comparison could change. Their general conclusion was that the cost-benefit of the supercomputer was better than the cloud.

The work of Prukkantragorn and Tientanopajai (PRUKKANTRAGORN; TIENTANOPAJAI, 2017) analyzes the performance and price tradeoff of using Amazon AWS. They compared the execution of High-Performance LINPACK in several of Amazon's instances. Their findings show that the price of bigger and powerful instances does not present a linear speedup. Comparing the biggest instance with the smallest one that the authors evaluated. The price relation of these instances shows a price increase of 16 times; however, the performance increase was only ten times for the same problem size. The overall conclusion is that depending on the problem size, the costly machines could not present better cost-efficiency than smaller, and cheaper ones.

The work of Costa et al. (COSTA. et al., 2018) performed performance and cost analysis comparing the on-demand versus preemptive instances, or spot instances, of the AWS and Google cloud providers. In terms of performance, they compared the CPU, I/O, and network, by using benchmarks used in HPC for profiling of systems. To analyze the cost, they used a MapReduce workload to compare them. Their findings were that the on-demand and preemptive instances do not present differences of performance for CPU, disk I/O, and network. In terms of cost, they achieved a reduction of 68% in AWS and 26% on Google.

3.1.3 Usability

Aljamal et al. (ALJAMAL; EL-MOUSA; JUBAIR, 2018) compared Azure, AWS, Google Cloud, and Oracle Cloud as a platform for HPC. The authors made a comparative analysis of the provider's offers. Their focus was on the features offered by the providers to be used for HPC applications; however, they did not perform a simulation. They concluded that there is not a cloud provider that fits all user requirements for HPC. The authors have an extension of their work (ALJAMAL; EL-MOUSA; JUBAIR, 2019), where they add a brief analysis of the configurations costs of the same four providers. Moreover, their focus still relies on the general characteristics of the providers from a user perspective.

The work of Cunha et al. (CUNHA et al., 2017) introduces a tool, named advisor, to make job placement decisions in HPC hybrid cloud environments. The advisor is a decision layer between the user and the resource manager; its procedure could be resumed as: the user submits a job to the advisor, advisor determines the best environment for job execution, and then forwards jobs to the resource manager. Its main decision variables are the wait time estimates (how long a job is expected to wait in a local resource manager queue before execution), and runtime estimates (how long a job is going to run, once it starts execution). Based on these, the advisor uses a machine learning model to decide where, local or cloud, the job will execute. They evaluate their mechanism by using real trace from supercomputer centers in the USA. Their principal findings are that the mechanism has the potential to enable faster decisions on hybrid cloud environments, and they were able to cut off the uncertainty by using a limit tolerated by the user to accept predicted results.

The Table 3.1 compares the most recent works presented in this section with our work. We point the studies related to the cost-efficiency of the cloud, CPU, and network performance analysis and improvements; if the works rely on a single machine or in a multi-instance cluster. The focus on HPC applications are evaluated as well, and if the application demands are considered and the heterogeneity of the cloud is exploited. If the work proposed an automated tool to help the user to deal with the decisions related to cloud instances, and if the work considers the user constraints.

3.2 Scope of this Thesis

In terms of actors interacting with Cloud Computing, we have two clearly defined roles, the provider and the user. The provider is the company that implements and provides cloud computing services and is paid for it. The provider is also responsible for all the technical definitions of what type of hardware will be used, which virtualizer, and also the billing model. The user is the consumer of the service provided by the provider, which may be either an end-user, a single person, or an organization.

The scope of our thesis is relative to the user's point of view, where we are studying the feasibility of using cloud focused on it, being agnostic to the service provider.

Table 3.1: Comparison between the State of the Art and our work.

	Cost-Efficiency	CPU Performance	Network Performance	Cluster	HPC Applications	Heterogeneous Configurations	Automated Tool	User Constraints
Costa et al. 2018	✓	✓	✓					
Li et al. 2018	✓	✓				✓		
Mariani et al. 2017		✓	✓	✓	✓			
Cunha et al. 2017		✓		✓	✓		✓	✓
Persico et al. 2015			✓					
Persico et al. 2016			✓					
Aljamal et al. 2018								✓
Prukkantragorn et al. 2017	✓	✓			✓			
Aljamal et al. 2019	✓							✓
Prabhakaran et al. 2018	✓							
Kotas et al. 2018	✓	✓		✓	✓			
Our Proposal	✓	✓		✓	✓	✓	✓	✓

What we seek to bring to the customer is the possibility of choosing an execution environment for HPC applications in an automated and cost-optimized manner. This way, the user will have a suggestion of the environment to be used that presents the best relation between cost and performance. To achieve this, we conducted a study of the cloud instances heterogeneity in terms of processing, as well as the processing demand between the various application processes. Using both studies, we developed a decision mechanism that matches the characteristics of the available instances and the demands of the application, causing the user to execute the application minimizing the idle time of the environment and reducing the cost. The delivery itself of this thesis is the mechanism of choice for the heterogeneous instances to be used for running HPC applications. The decision mechanism we developed was named *CloudHet*, referring to Cloud and Heterogeneity.

CloudHet has some conditions, as it was developed in line with public cloud instance offerings. The first condition is that CloudHet only considers instances of the same size in terms of the number of cores; all instances chosen must have the same amount of cores. At the same time, the quantity of MPI processes that the application uses to execute must be multiple of the number of cores of the instances chosen. For example, if the instances to be considered have eight cores, the application should execute using a multiple of 8 processes, such as 8, 16, 32, 40, 360. If we consider 32 cores instances, the application should execute a process quantity multiple of 32; and so on. The other condition is that CloudHet works only with applications with a fixed size of MPI processes, we do not consider the dynamic process creation at this time.

4 PROPOSED METRIC AND BENCHMARK FOR IMBALANCE WORKLOADS

Executing large parallel applications in cloud environments is now an important research focus in cloud computing. Among the three service models offered by the Cloud Computing model, the most suitable for High-Performance Computing (HPC) is the IaaS model, since it provides instances that can be customized to the users' needs and combined to build a cluster system. Currently, many cloud providers offer a large number of instance types that target applications from the HPC domain. In contrast to traditional cluster systems, which usually consist of homogeneous cluster nodes, cloud providers offer the possibility to allocate different types of instances flexibly and interconnect them, thus creating large *heterogeneous* platforms for distributed applications. Usually, the heterogeneity of the cloud is explored near the hardware level (DONG et al., 2017) and is mostly related to the service provider side. However, few studies aim to explore the heterogeneity in public clouds.

One interesting way to use such a heterogeneous system is by matching the demands of the application to the underlying hardware. In the context of this thesis, we focus on matching the *load* of the different processes of a parallel application to different instance types, thus providing a way to mitigate the *load imbalance* in such applications.

Previous studies (ROLOFF et al., 2017a; ROLOFF et al., 2017b) showed that such matching could be beneficial for the cost-efficiency of an application, mostly by executing processes with a larger load on fast, expensive instances, while executing processes with a lower load on slow, cheap instances. In this way, the overall cost of the execution in the cloud can be reduced while maintaining similar performance. Due to the *principle of persistence* (KALÉ, 2002), which states that load imbalance will stay the same during execution and between several executions for most applications, profiling and matching has to be done only once per application.

In this Chapter, we study the concepts and benefits of heterogeneous clouds, focusing on which pattern of load imbalance is most suitable for these cloud systems. We introduce a new metric, the *cost-delay product (CDP)*, with the purpose of measure the cost-efficiency of different cloud allocations. We introduce a benchmark, *ImbBench*, that can simulate several types of load imbalance patterns in parallel applications. With this benchmark, users can study load imbalance problems and use it to test different cloud instances, thereby defining which of these instances best meets their needs. To validate both the heterogeneous clouds and ImbBench, we performed several tests on the Microsoft Azure Cloud provider using the IaaS Service Model. Our main finding was that it is possible to save money while maintaining the performance of an unbalanced application across a wide range of imbalance patterns. We achieved that by using a heterogeneous cluster made with up to three different instance types. Furthermore, we also validate our approach using the MPI version of the NAS Parallel Benchmarks (NPB).

4.1 Cost-Delay Product: a metric to measure cost efficiency

An important aspect when evaluating cloud instance types is comparing their price/performance tradeoff. Several basic metrics have been proposed to evaluate this tradeoff. One metric that compares different configurations of the cloud is the cost-efficiency (ROLOFF et al., 2012a; MOSCHAKIS; KARATZA, 2012). However, these metrics do not allow more comprehensive evaluations, for example, by placing more emphasis on price or performance according to the user's preference.

We introduce a new metric, the *cost-delay product (CDP)*, which is modeled after the energy-delay product (EDP) (HOROWITZ; INDERMAUR; GONZALEZ, 1994; Laros III et al., 2013). The goal of EDP is not to minimize energy consumption at all, but it aims to introduce a metric to find a suitable trade-off of energy savings and performance degradation. The basic CDP metric is defined as follows:

$$CDP = cost\ of\ execution \times execution\ time \tag{4.1}$$

The $cost\ of\ execution$ represents the price of the environment used to execute the application. The majority of public cloud providers base their price model on hours of use, and the cost used in Equation 4.1 is the price per hour (in US$) of the instances allocated in a cloud provider. The $execution\ time$ is the application's execution time in the allocated environment.

Lower values for the CDP indicate a better cost-efficiency for a particular application in a particular environment. This metric can be used to directly compare two different cloud allocations, including in different providers. This basic version of the CDP resembles earlier proposals to quantify cost-efficiency (ROLOFF et al., 2012a).

In our previous research (ROLOFF et al., 2017b; Roloff et al., 2017), we observed that public cloud providers offer a wide range of instance configurations with different performance and price ranges. Therefore, the user can choose an environment that prioritizes performance or price. To express such a preference, the CDP metric can be extended by applying a weighted approach, depending on whether priority should be given to higher performance or lower cost. The resulting metrics, C^2DP, and CD^2P, are defined as follows:

$$C^2DP = (cost\ of\ execution)^2 \times execution\ time \tag{4.2}$$

$$CD^2P = cost\ of\ execution \times (execution\ time)^2 \tag{4.3}$$

Since they represent different units, values of CDP, C^2DP, and CD^2P can not be directly compared with each other. However, a user can easily calculate the three metrics for his target environments and compare them by the three different aspects, with CDP meaning the best cost-efficiency without focus on execution time or performance the C^2DP representing the less expensive environment and the CD^2P showing the configuration with better performance. With these metrics, a cloud tenant can select whether performance or cost is more important to him, and select the comparison function accordingly. We exemplify a weighted approach using a power of 2 to introduce the metrics, although different weights can be used to fine-tune the desired tradeoff.

4.2 The ImbBench benchmark

The HPC field uses many benchmarks that focus on specific performance aspects, such as the LINPACK benchmark (DONGARRA; LUSZCZEK; PETITET, 2003), which is used to measure performance for the TOP500 list; the NAS suite (SAINI; BAILEY, 1996), which consists of Computational Fluid Dynamics (CFD) benchmarks; and the Rodinia suite (CHE et al., 2009), which was designed to benchmark CPU+GPU systems, among others. In terms of benchmarks designed to the cloud, there are two prominent works. The CloudBench (Silva et al., 2013) is a tool to handle benchmarking in the cloud, where the configuration and allocation tasks could be performed by it. It was released and maintained by IBM. The SPEC Cloud IaaS (BASET; SILVA; WAKOU, 2017) is the SPEC initiative to benchmarking in the cloud. Both initiatives merged in some fashion, and since the 2018 release of SPEC Cloud IaaS, they are distributed together. However, to the best of our knowledge, there is a lack of benchmark suites designed to exploit the heterogeneity of cloud computing instances.

In most cases, HPC applications are executed on *homogeneous clusters* consisting of several nodes with the same configuration, number of cores, memory, and disk. Thus the available benchmarks were designed with a homogeneous behavior. However, cloud computing provides a large variety of heterogeneous resources, in which users can build clusters of nodes with different characteristics (CHOHAN et al., 2010), thus creating a *heterogeneous cluster*.

To better evaluate heterogeneous systems, we developed the *Imbalance Benchmark (ImbBench)*, which is a set of MPI-based applications that simulate several behaviors in terms of process loads. ImbBench was designed with the heterogeneity of the cloud in mind. Its goal is to help the user to choose the most suitable configuration to execute an application in the cloud. It distributes the load among all the available processes according to a preselected imbalance pattern. The code was developed in the C programming language and is publicly available at <https://github.com/Roloff/ImbBench>. ImbBench can create all the implemented patterns for any of the available evaluations, CPU, and memory. It currently can scale up to 1024 processes.

4.2.1 Load Imbalance Patterns

There are four different load imbalance patterns implemented by ImbBench: Balanced, Multi-Level (two up to eight), Amdahl, and Linear. These patterns represent common types of imbalance in parallel applications.

The *Balanced* pattern, represented in Figure 4.1, simulates completely balanced application. In the Figure, the y-axis indicates the relative load of each rank (normalized such that the maximum load equals 100), while the x-axis shows the MPI ranks (there

Figure 4.1: The ImbBench Balanced Pattern.

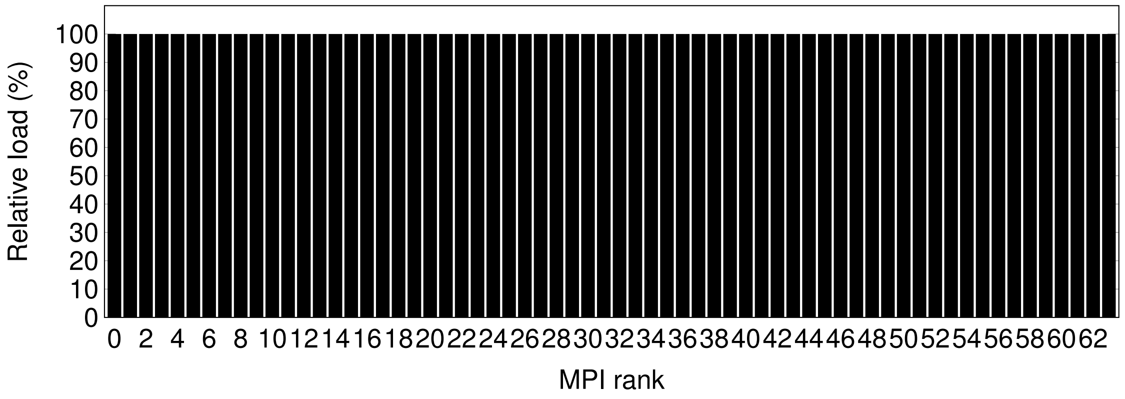
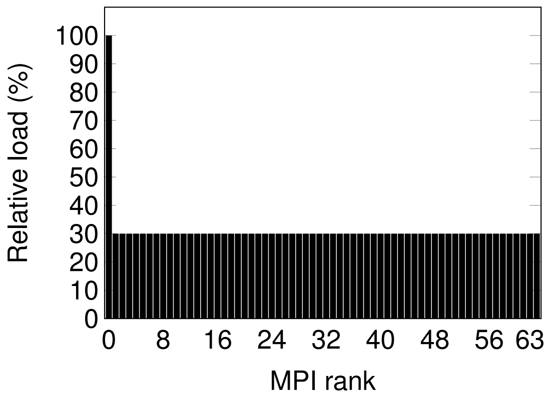
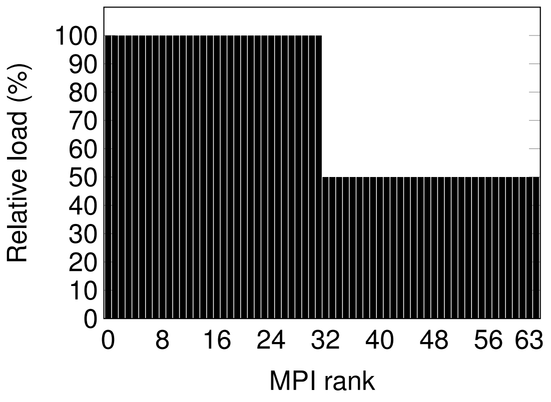


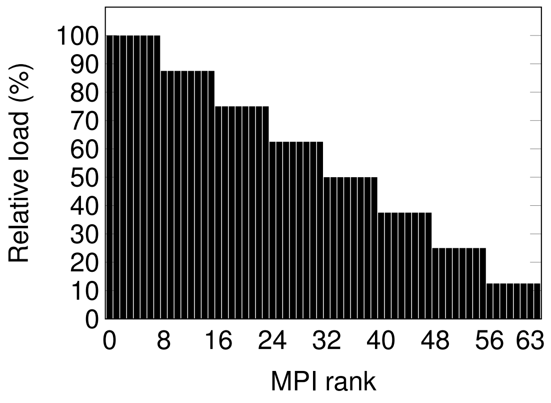
Figure 4.2: Load Imbalance Patterns simulated by the ImbBench.



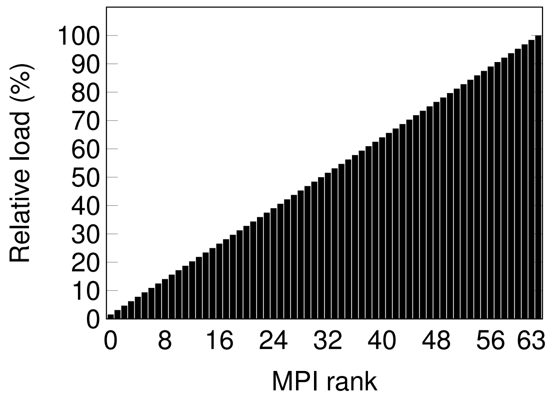
(a) Amdahl.



(b) Two-Level.



(c) Multi-Level.



(d) Linear.

are 64 ranks in total). This pattern simulates the most desirable behavior for an HPC application where all MPI tasks execute the same load and finishes its execution at same time; in this situation, there is no idle resources during the execution and, consequently, no money spent unnecessarily.

The other patterns, with load imbalance, are represented in Figure 4.2. Once again, the y-axis indicates the load of each rank, and the x-axis shows the MPI ranks. The *Amdahl* pattern is represented in Figure 4.2a. This pattern simulates an application that has one or more processes that execute much more work than all the other processes. Usually, this behavior presents itself when an application needs a central process to distribute all the tasks and consolidate the results. This pattern presents a high level of imbalance, and the majority of the processes, except the central one, present large amounts of idle execution time.

The *Multi-level* pattern, represented in Figure 4.2c, shows distinct load levels, between two and eight levels, simulating an application with several different loads among the processes. This pattern presents a mixed idle time, according to the load of the process.

The *Two-level* pattern is a particular case of the Multi-level pattern, which is represented in Figure 4.2b. We include it because it is a reasonably common imbalance pattern of parallel applications. In this case, half of the processes present no idle time, and the other half present idle time.

Finally, the *Linear* pattern simulates an application where all processes have a different load, starting from a low load on rank 0 and linearly increasing up to rank $n - 1$. Figure 4.2d represents the pattern, which idle time is 50% of total execution time.

4.2.2 Performance Workloads

ImbBench is designed to evaluate all aspects of cloud heterogeneity. Currently, processing power and main memory speed ratings are available for use. Network interconnection performance and disk I/O speed measurement will be developed in the future.

For the simulation of the maximum processing load supported ImbBench can currently use two workloads: random number generation and matrix multiplication. The random number generation workload uses the standard pseudo-random number generator from **libc**. When a process needs to execute a higher load than another, it merely generates more random numbers in a loop. This workload was profiled with the PERF tool (MELO, 2010) as almost 100% CPU-bound computation. The matrix multiplication workload decomposes a matrix into several pieces that are distributed among the various processes. When one process needs to carry more load than another, it gets a larger matrix chunk to calculate. The user can decide whether he or she wants the matrix to contain integers or floating-point numbers and can make a more refined evaluation of these types of computations. In the future, we intend to experiment with additional load creation strategies, such as additional algorithms or strategies that focus on specific computations.

For the memory speed test, ImbBench uses an insertion operation in a Binary Search Tree (BST). The processing load corresponds to the quantity of numbers it inserts into the BST; thus, higher loads result in larger trees. In our measurements, this workload represents a 50% memory bound and 50% CPU bound algorithm, according to the PERF profiling results. In the future, we intend to implement algorithms with different levels of memory load.

4.3 Profiling the Performance and Price of Azure Instances

In our experiments, we use Microsoft Azure for the evaluation, which has shown good performance for HPC applications (ROLOFF et al., 2012b). Among all the available instances in Azure, we selected the instances with 16 cores since there are seven different configurations for this number of cores, offering a multitude of heterogeneous choices. The instances used in our evaluation were: D16, D5, E16, F16, H16, G4, and L16. They are configurable with different memory, disk sizes, and processor types.

To profile the instances, we executed the High-Performance Linpack benchmark (DONGARRA; LUSZCZEK; PETITET, 2003) to measure the processing capacity of each instance in GigaFlops. The Linkpack benchmark execution is the methodology used to create the TOP500 rank. With the Linpack results, we can organize the instances into three different groups of processing power: High, Medium, and Low. The High group contains only the H16 instance type, which achieved more than 650 GigaFlops and was the most powerful machine in our experiments. The Medium group consists of the L16 and G4 instances; both achieved slightly more than 400 GigaFlops. The Low group is composed of the other four machine types: D16, D5, E16, and F16, with a Linpack performance of about 250 GigaFlops.

The price of the instances can also be classified into the same three groups. The G4 and H16 instances are in the High and Medium groups, respectively, with the G4 instance costing around three dollars per hour and the H16 instance costing around two dollars. All the other instances are in the Low-cost group, with the price close to one US dollar per hour.

Another aspect is the cost-efficiency of the instances, which helps users to understand how efficient an instance is, in terms of processing power per dollar. To compare this characteristic of the instances, we calculate how many GigaFlops an instance delivers per dollar. We can calculate this metric for the Linpack benchmark as follows:

$$GigaFlop/US\$ = \frac{Linpack\ result}{Price\ per\ hour} \tag{4.4}$$

Using Equation 4.4, we can determine that the instance H16 presents the best relation between price and performance. The L16 instance is slightly lower than H16 but still showed a good relation. The G4 instance shows the worst price-performance relation because it has the highest price among all the instances in our evaluation. The other instances presented a price-performance relation near the level of 250 GigaFlops

Table 4.1: Characteristics of the Azure Instance Types.

Name	Price/hour	Linpack (GFLOPS)	GFLOPS/US$
D16	US$ 0.936	247.54	264.46
D5	US$ 1.117	280.04	250.71
E16	US$ 1.186	254.97	214.98
F16	US$ 0.997	263.51	264.31
G4	US$ 3.072	417.32	135.85
H16	US$ 1.941	657.64	338.81
L16	US$ 1.376	406.63	295.52

per US$. Table 4.1 summarizes the results of the Linpack benchmark, the cost-efficiency, as well as the price of the profiled instances.

4.4 ImbBench CPU Performance and Cost-Efficiency

After completing the profile of the instances, we executed the ImbBench to evaluate if it is possible to benefit from the heterogeneity presented in these instances of Microsoft Azure. For the evaluation, we built clusters with 64 cores, using four Azure instances. The nodes are running with Ubuntu 16.04 (kernel 4.13), GCC version 5.4, and Open MPI version 1.10.2. We executed each ImbBench pattern 30 times; the illustrated results are the average execution time of these 30 executions. As the baseline, we use the most powerful instance type, H16, which also presented the best cost-efficiency. We built a homogeneous cluster with four H16 instances and compared the heterogeneous clusters against it. We created several heterogeneous configurations until we found the configuration most similar to the baseline cluster in terms of total execution time. Since we only identified three different performance levels of the Azure instances, we executed only the Two-Level and Four-Level patterns from the Multi-Level patterns of ImbBench. The Amdahl and Linear patterns were also used in the evaluation.

4.4.1 The Amdahl pattern

The first analyzed pattern is the Amdahl pattern. Figure 4.3 shows the results of the Amdahl pattern of ImbBench using the cluster made of four H16 instances. As can be seen in the figure, rank 0 presented a significantly higher execution time (approx. 5 seconds) than the other ranks (approx. 1 second), as expected.

With these results, we determined that a heterogeneous hardware environment can

Figure 4.3: Execution time of ImbBench using the Amdahl pattern executed in the homogeneous cluster of H16 instances.

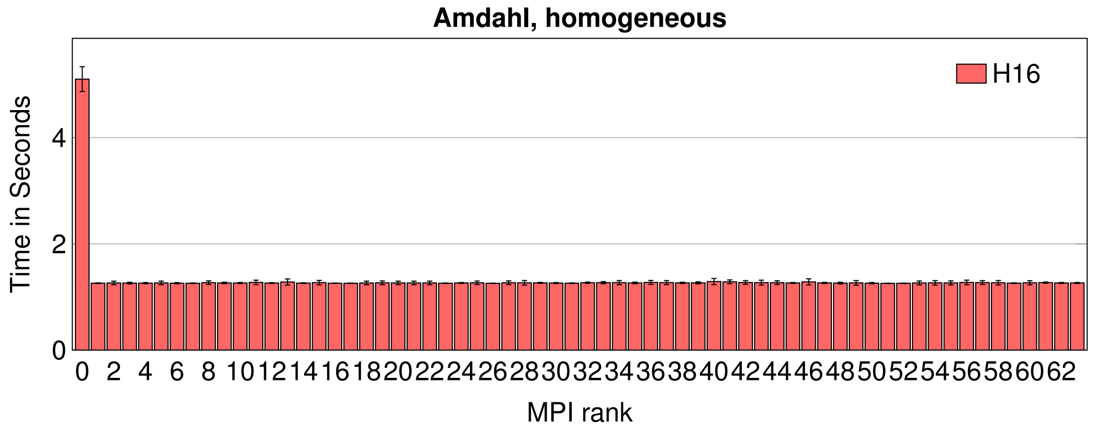
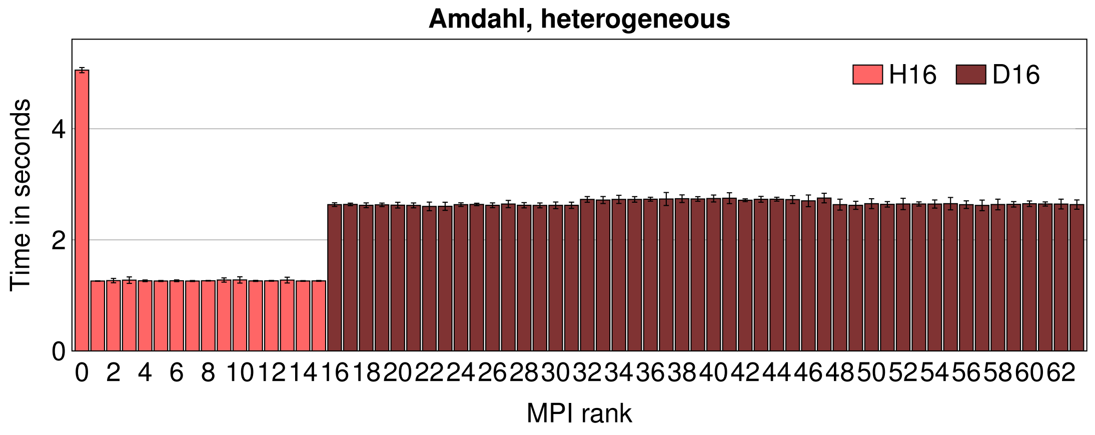


Figure 4.4: Execution time of ImbBench using the Amdahl pattern in the heterogeneous cluster of one H16 instance and three D16 instances. Ranks 0-15 are running on the H16 instance, while 16-63 are running on the D16 instances.



be beneficial for this load imbalance pattern, by using a powerful instance together with less powerful and less expensive instances. Thus, we build a cluster with one H16 instance and three D16 instances. The reason we chose the D16 instances was that it is the cheapest instance among all the instances used in our evaluation. The results of the heterogeneous execution of the Amdahl pattern are shown in Figure 4.4.

As we can see in the figure, the total execution time remains the same, because the process with the highest demand was executed in the same instance as before, and the other processes, even when executed in D16 instances, do not reduce the total execution time. It is possible to note that the processes from 0 to 15 executed faster than the processes 16 to 63. The reason is that the processes 0 to 15 were executed on the H16 instance, and took 1 second to finish, and the processes 16 to 63 were executed in the D16 instances and took approx. 3 seconds to finish. We can conclude that, even with 3/4 of the processes executed on instances with less processing power, the total execution time

Figure 4.5: Execution time of ImbBench using the two-level pattern for application load in the homogeneous H16 cluster.

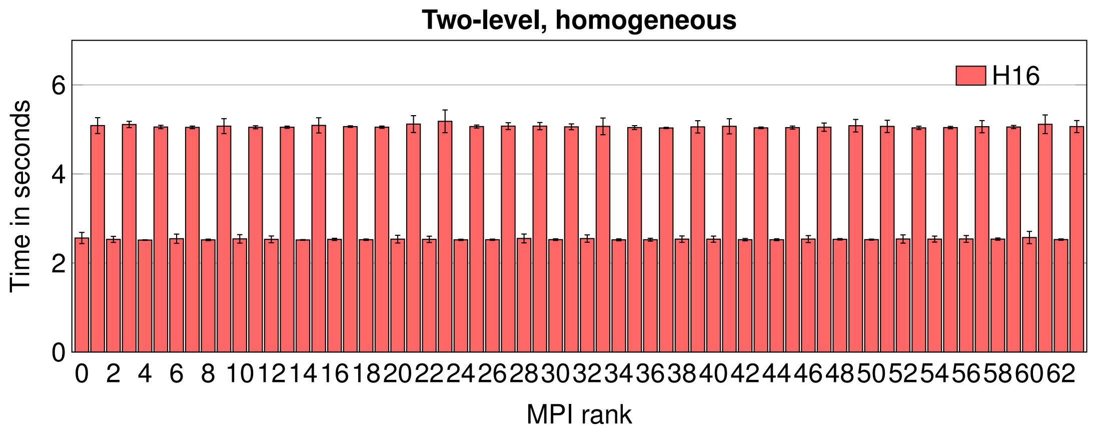
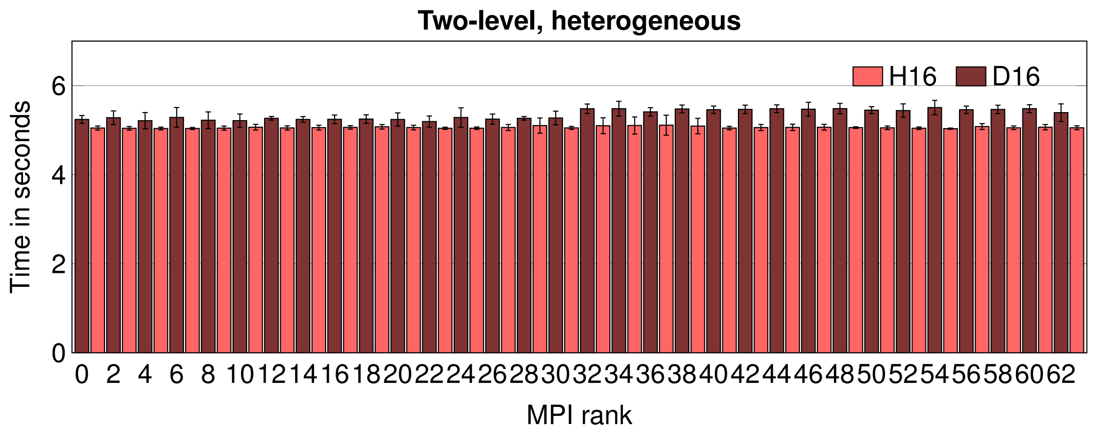


Figure 4.6: Execution time of ImbBench using the two-level pattern for application load in the heterogeneous cluster. Even ranks are running on D16 instances, odd ranks on H16 instances.



was not affected, because they executed faster than the process with the highest load.

In terms of price, the execution of an Amdahl-like application in a heterogeneous environment is very advantageous. The price per hour of the H16 cluster is US$ 7.764, and the price per hour of the heterogeneous cluster is US$ 4.749, a saving of US$ 3.015 per hour, corresponding to a 38% total cost reduction.

4.4.2 The Two-level pattern

The next pattern is the Two-Level, where the application is divided into two levels of demand. The higher level computes a certain amount of work, and the low level computes exactly half of the high level. Figure 4.5 shows the results of the Two-Level pattern of ImbBench using the cluster made of four H16 instances. As seen in the Figure, the odd processes are the high demand processes, and they took around 5 seconds to perform their work. The even numbers are the low demand processes, and they fulfilled their work in around 2.5 seconds.

Analyzing the results, we can conclude, as in the Amdahl pattern, that a user of an application with this behavior can take advantage of the heterogeneous cloud computing instances by mixing two instance types. Using an Instance with high processing power to execute the processes with high demand and an instance with less processing power to execute the processes with less demand. We build a cluster with two instances types and with two instances of each type. We chose the H16 and D16 instances, the H16, because it is the baseline instance and the cheap instance in our chosen group, the D16. The results of the Two-Level pattern execution on the heterogeneous cluster are shown in Figure 4.6.

As we can see in the figure, the total execution time was slightly higher than the execution on the H16 cluster. We can observe that the even processes, which are processes with less demand, now took more time to perform their work, because they were executed in D16 instances. We can conclude that, even with half of the processes being executed in instances with less processing power, the total execution time presented only a small increase.

In terms of price, the execution of such an application in a heterogeneous environment is very advantageous. The price per hour of the H16 cluster is US$ 7.764, and the price per hour of the heterogeneous cluster is US$ 5.754, a saving of US$ 2.01 per hour or a 25% reduction of the total cost.

4.4.3 The Four-level pattern

In the Four-Level pattern, the application load is divided into four levels of demand. The difference between the levels is constant, and the lowest level computes *X* instructions, the next level computes *2X*, and so on. Figure 4.7 shows the results of the Four-Level pattern of ImbBench using the homogeneous cluster of four H16 instances. The Figure shows that the processes' demands are divided into groups of four, where the fourth process in each group is the one that performs more computation than the others. In practical terms, the time spent by every fourth process is the time spent by the application, approximately 5 seconds.

By analyzing the results, we can see the first, second, and third processes can be executed on machines with less power without increasing the total execution time. After a few simulations, we determined that the most suitable configuration for this pattern was to

Figure 4.7: Execution time of ImbBench using the four-level pattern for application load in the homogeneous H16 cluster.

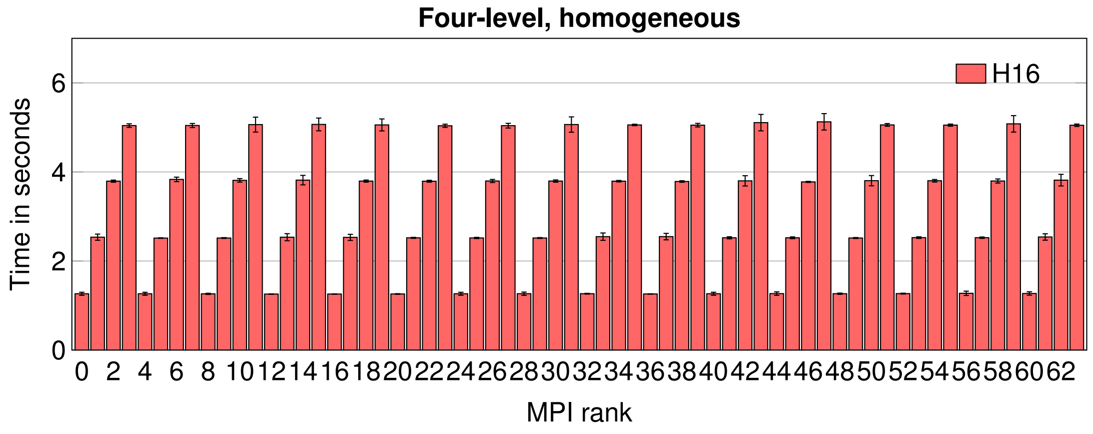
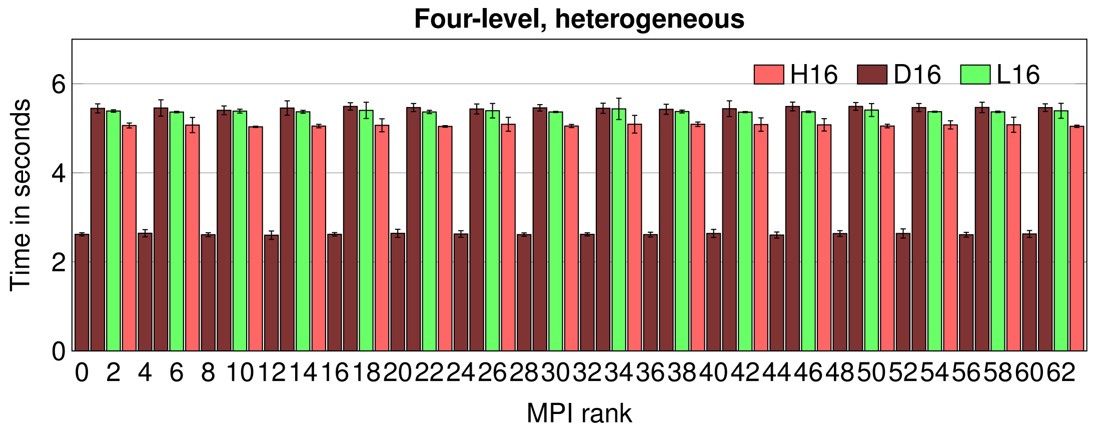


Figure 4.8: Execution time of ImbBench using the four-level pattern for application load in the heterogeneous cluster. Ranks 0-1,4-5,8-9,... run on D16 instances, ranks 2,6,10,14,... run on an L16 instance , ranks 3,7,11,15,... run on an H16 instance.



create a cluster with three different instance types. The processes with less demand were executed on two D16 instances, the processes with intermediate demand were executed on an L16 instance, and the processes with high demand were executed on an H16 instance. The results of the execution of the Four-Level pattern on the heterogeneous cluster are shown in Figure 4.8.

We can observe in the Figure that the execution time was slightly higher than the execution on the H16 cluster. The group of processes with less demand executed in D16 instances increased their time but did not affect the total time. The group of processes that were executed in the L16 instance increased their time as well, and the processes executed in an H16 instance kept their execution time. We can conclude that, even with 3/4 of the processes being executed on instances with less processing power, the total execution time only slightly increased.

In terms of cost-efficiency, the execution of an application similar to the Four-level

Figure 4.9: Execution time of ImbBench using the Linear pattern for application load in the homogeneous H16 cluster.

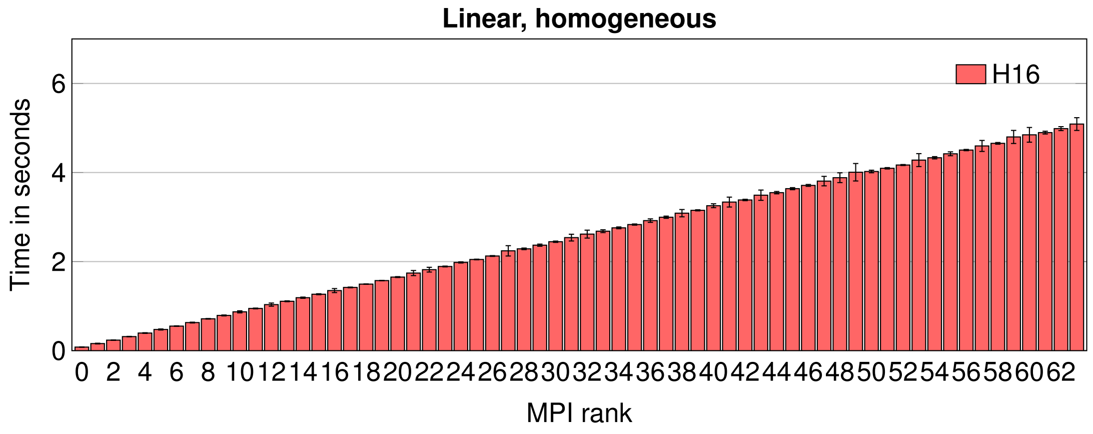
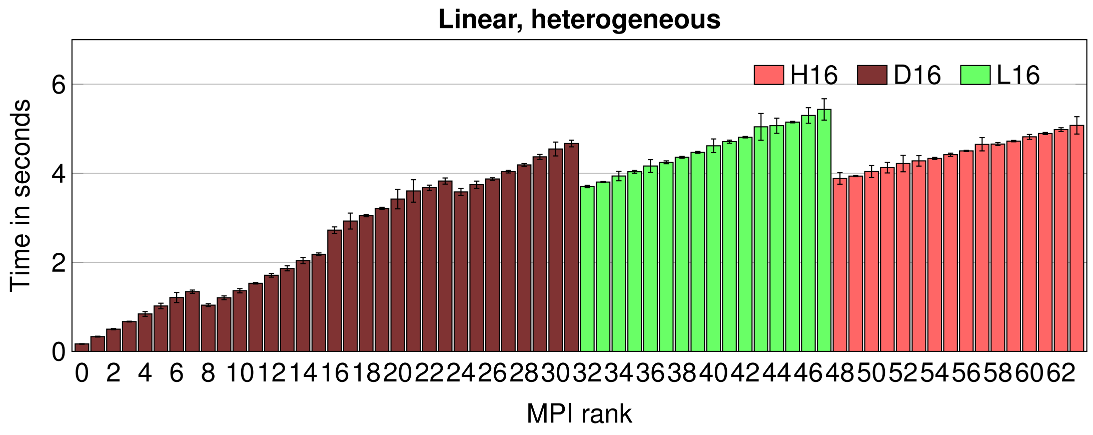


Figure 4.10: Execution time of ImbBench using the Linear pattern for application load in the heterogeneous cluster. Ranks 0-31 run on D16, ranks 32-47 on L16, ranks 48-63 on H16.



pattern is advantageous. The price per hour of the H16 cluster is US$ 7.764, and the price per hour of the heterogeneous cluster is US$ 5.189, a saving of US$ 2.575 per hour or a 33% total execution cost.

4.4.4 The Linear pattern

The last pattern is the Linear pattern, where each process of the application has a different load. The process with rank 0 has the lowest computational demand, and the demand increases linearly when the rank increases. Figure 4.9 shows the results of the Linear pattern of ImbBench using the cluster of four H16 instances. The figure shows that the processes' execution time increases as expected when the process rank increases. The total execution time of the application is the execution time of the process with the highest rank, which is around 5 seconds.

To create a configuration with mixed instances that do not affect the execution time, a few simulations were made, and the most suitable configuration was the same as in the Four-Level pattern, a cluster with two D16 instances, one L16 instance and one H16 instance. The first 32 processes were executed on the D16 instances, the processes from 32 up to 47 were executed on the L16 instance, and the processes from 48 up 63 were executed on the H16 instance. The results of the execution of the Linear pattern in the heterogeneous cluster are presented in Figure 4.10.

As seen in the Figure, the first 32 processes increased their execution time, but this does not affect the total execution time since their execution time was shorter than the processes with high demand. The processes executed on the L16 instance increased their time as well and had slightly higher execution time than when executing on the H16 cluster. The processes that were executed on the H16 instance kept their execution time. The conclusion is the same as for the Four-Level pattern; even with 3/4 of the processes being executed in instances with less processing power, the total execution time presented an increase up less than 7%.

In terms of price, the execution of an application with a linear behavior presents an edge in a heterogeneous environment. The price per hour of the H16 cluster is US$ 7.764, and the price per hour of the heterogeneous cluster is US$ 5.189, a saving of US$ 2.575 per hour or a 33% reduction of the total cost.

4.4.5 Summary of CPU results

Table 4.2 summarizes the results of ImbBench using both homogeneous and heterogeneous clusters. As we can see in the Table, the performance loss when using the heterogeneous configuration was smaller than 7% for all the cases. The only exception is the Amdahl Pattern, where we observed a small performance gain. When analyzing the cost-efficiency, the CDP results are better with the heterogeneous configurations, with a minimum gain of 21.23%, which means that this configuration is 21.23% more cost-efficient than the homogeneous one. If we use the C^2DP to focus on the cost of execution, then the results are much more favorable for the heterogeneous configurations, with cost-efficiency gains of up to 62.92% compared to the homogeneous configuration. When the CD^2P is used to focus on execution performance, the results of the heterogeneous configurations presented results up to 39.92% better than the homogeneous configuration. In terms of costs, the heterogeneous configurations were between 25% and 38% cheaper

Table 4.2: Summary of the ImbBench results.

		Two-Level	Four-Level	Amdahl	Linear
Execution Time	Hom.	5.18s	5.13s	5.10s	5.09s
	Het.	5.51s	5.49s	5.05s	5.43s
	gains	-5.92%	-6.65%	0.90%	-6.35%
CDP	Hom.	1.12	1.11	1.10	1.10
	Het.	0.88	0.79	0.67	0.73
	gains	21.23%	28.40%	39.38%	33.60%
C^2DP	Hom.	2.41	2.38	2.37	2.37
	Het.	1.41	1.14	0.88	1.13
	gains	41.62%	52.15%	62.92%	52.30%
CD^2P	Hom.	5.79	5.67	5.61	5.58
	Het.	4.85	4.35	3.37	4.26
	gains	16.27%	23.29%	39.92%	23.78%

than the homogeneous one, and on a cloud-scale, these results can have a huge impact on the user's budget.

4.5 ImbBench Memory Evaluation

To evaluate the memory heterogeneity, we chose the instances with eight cores from Microsoft Azure. We initially made a profile of the memory speed of instances by using the STREAM benchmark (MCCALPIN, 1995). The list of all instances that were considered is shown in Table 4.3. The Table shows the instances' prices and the amount of memory as well.

Table 4.3: Profile and results of the STREAM benchmark for the Azure Instances.

Instance type	Price/hour	Memory (GB)	STREAM Results (in GB/sec)			
			Copy	Scale	Add	Triad
A8	US$ 0.975	56	11.85	13.11	13.79	13.62
A8v2	US$ 0.40	16	5.86	6.39	6.68	6.69
D4	US$ 0.559	28	11.03	11.00	12.42	12.26
D13	US$ 0.741	56	10.09	9.85	11.23	11.15
E8	US$ 0.593	64	8.99	9.46	10.07	10.15
F8	US$ 0.498	16	10.50	10.49	12.00	11.70
G3	US$ 2.44	112	11.53	11.85	13.39	13.16
H8	US$ 0.971	56	13.06	12.78	14.70	14.39
H8m	US$ 1.301	112	13.08	12.88	14.71	14.48
L8	US$ 0.688	64	9.96	10.10	11.33	11.20

Figure 4.11: Results for the Linear pattern and Memory performance for the homogeneous H8 cluster.

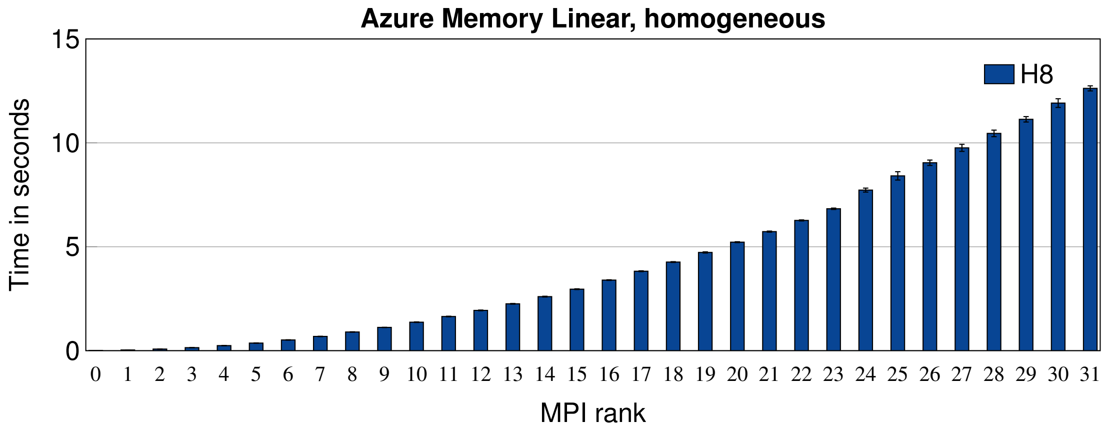
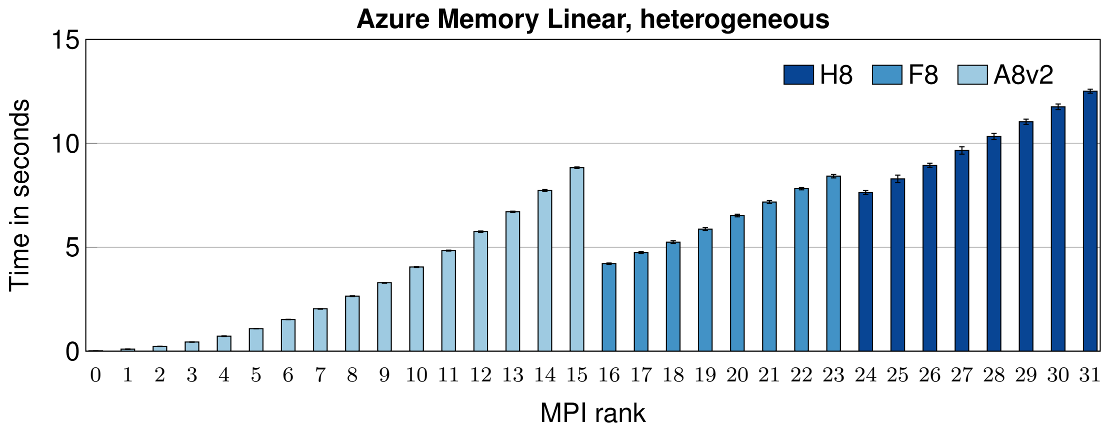


Figure 4.12: Results for the Linear pattern and Memory performance for the Azure heterogeneous cluster.



In terms of memory, there is less heterogeneity among the Microsoft Azure instances. Only the A8v2 instance presented a result that is far different from the other instances. The rest of the instances are closed, but we still can group them into two groups. We could separate the instances that performed 13 and 14 GBytes/sec from the instances that performed 10,11 and 12 GBytes/sec. The first group is composed of the A8, G3, H8, and H8m instances, and the second contains the D4, D13, E8, F8, and L8 instances. These groups have close results, but after some experimentation, we found some heterogeneity that could be exploited.

For the experiments, we configured a cluster of 32 cores consisting of four instances with eight cores each. All instances had Ubuntu 18.04 LTS installed. For each experiment, we run 30 executions and show average values as well as the standard deviation. We use the same methodology as the CPU evaluation, with a baseline configuration and a heterogeneous one.

As the concept of heterogeneity and the different ImbBench patterns were already extensively demonstrated in the previous Section, we will only show the results of linear pattern memory tests.

Figure 4.11 shows the results of the ImbBench Linear pattern for testing the memory in a homogeneous H8 cluster. As observed, the growth of execution time is linear, according to the increase of the process number. Likewise, the CPU testing, the process with the biggest rank, in our experiments the process 31, is the critical one for the execution time.

After some experiments with different instance combinations, we found a configuration that does not increase the execution time while reducing the cost. Figure 4.12 shows the results of a cluster composed of two A8v2 instances, one F8 instance, and one H8 instance. As shown, the total execution time was not affected, because the processes with higher load (24-31) were still executed in the most powerful instance. Furthermore, the processes with less load (0-23) were executed by using less powerful instances. This configuration presented the best cost-efficiency without increasing the execution time.

4.6 Evaluation of the NAS Parallel Benchmarks

After evaluating heterogeneous clouds with ImbBench, we tested whether the heterogeneous configurations are suitable for real HPC applications. We chose the MPI version of the NAS suite of parallel benchmarks (NPB) (BAILEY et al., 1991), version 3.3.1, which is a widely used workload in the HPC community. NAS is the opposite end to ImbBench, while ImbBench is an unbalanced benchmark, NAS is a suite of applications that has nearly equally split computing among its MPI processes. That is, ImbBench is

Figure 4.13: Cost-efficiency results of the NAS benchmarks. Lower values indicate higher cost-efficiency.

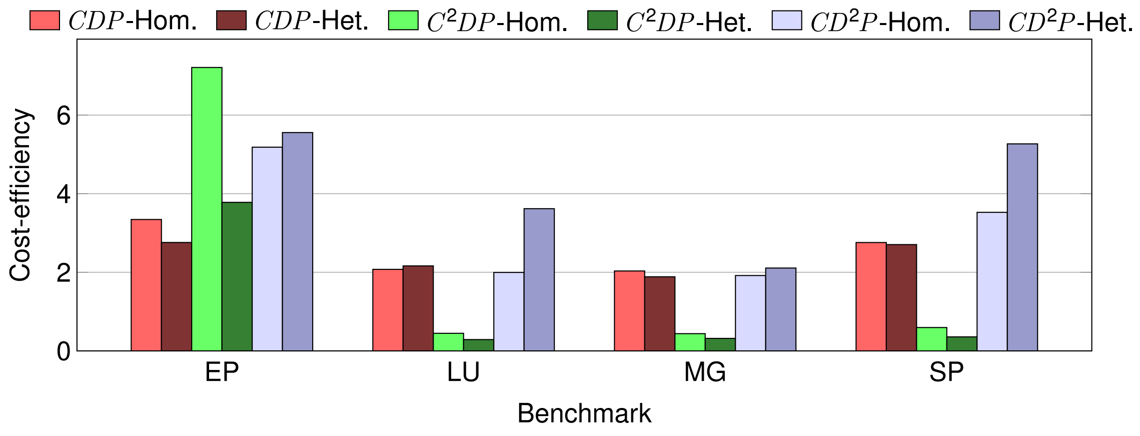


Table 4.4: Summary of the NAS results.

Name	Metric	Gains	Best het. configuration
EP	CDP	17%	1-H16 3-F16
EP	C^2DP	47%	1-H16 3-F16
EP	CD^2P	-7%	1-H16 3-F16
LU	CDP	-4%	1-H16 3-L16
LU	C^2DP	35%	1-H16 3-D16
LU	CD^2P	-38%	1-H16 3-L16
MG	CDP	7%	1-H16 3-L16
MG	C^2DP	27%	1-H16 3-D16
MG	CD^2P	-9%	1-H16 3-L16
SP	CDP	2%	1-H16 3-D16
SP	C^2DP	39%	1-H16 3-D16
SP	CD^2P	-49%	2-H16 2-L16

the best scenario to use in heterogeneous environments, and NAS is close to the worst possible scenario, where we have a theoretically balanced application. However, it was possible to identify some imbalance in NAS, with patterns similar to the Amdahl and Two-level patterns discussed in Section 4.2.

From the NAS suite, we selected the applications with the highest imbalance for our evaluation: EP, LU, MG, and SP. We measure the cost-efficiency of the execution on the H16 homogeneous cluster and several mixed instances, using 64 cores and MPI ranks in all cases, as before. All the NAS applications were executed 30 times; we show the average values for the results.

The results of the four homogeneous H16 instances and the best heterogeneous configurations are shown in Figure 4.13. As we can see in the Figure, the applications EP, MG, and SP presented better CDP in the heterogeneous execution than in the homogeneous execution. This means that these applications executed at an acceptable time at a lower cost. As expected, the C^2DP was lower in all the four heterogeneous executions, because its dominant factor is the cost. Regarding the CD^2P, the homogeneous configurations were slightly better in EP and MG and much better in LU and SP.

The best heterogeneous configuration for LU regarding the three metrics was one H16 instance and three F16 instances. LU performed better for CDP with a cluster composed of one H16 instance and three F16 instances. For the C^2DP, the best result was achieved using a configuration with one H16 instance and three D16 instances. For the CD^2P, the best result was achieved with one H16 and three F16 instances. The MG application performed better for both CDP and CD^2P using a configuration with one H16 instance and three L16 instances. For the C^2DP metric, the best result was achieved by using a configuration with one H16 and three D16 instances.

SP performed better for the CDP and C^2DP by using a configuration with one H16 and three D16 instances. For the CD^2P, the best result was achieved with a configuration of one H16 and three L16 instances. Table 4.4 summarizes the results of CDP, C^2DP, and CD^2P of NAS. We can conclude that it is possible to benefit from the cloud heterogeneity and mix of different instance types to create a more suitable execution environment, even for the NAS suite. As a side remark, we performed several simulations mixing three and four instance types, but the results were not better than the ones with two instances for these benchmarks.

4.7 Concluding Remarks

When developing HPC applications, it is important to distribute the work equally among the tasks. However, this goal can not always be achieved, and a certain amount of imbalance can be observed in most parallel applications. Heterogeneous clouds that are composed of different instance types are an interesting way to execute parallel, load imbalanced applications. In such a heterogeneous system, tasks with lower computational demands execute on slower but cheaper machines, while tasks with higher demands execute on faster but more expensive machines, thus increasing the overall cost-efficiency of the application.

In this Chapter, we introduced the *Imbalance Benchmark (ImbBench)*, whose main purpose is to help users to profile their environment in terms of the heterogeneity of the instances, and to discover opportunities for heterogeneous clouds. During the evaluation of ImbBench we discovered that, depending on the application imbalance pattern, it is possible to improve the cost-efficiency of the cloud environment with negligible increases in execution time. Our results have shown that heterogeneous instance allocations can increase the cost-efficiency by up to 63$\%$ with less than 7$\%$ of execution time overhead. Experiments with the NAS Parallel Benchmarks showed that these gains could also be observed with traditional distributed applications. These results show us that users can take advantage of the heterogeneity offered by cloud providers.

The overall conclusion of this Chapter is that the construction of heterogeneous cloud environments has been validated and is a research subject that should be considered for future initiatives.

5 CLOUDHET: A MECHANISM FOR HETEROGENEOUS CLOUD ALLOCATION

Public cloud providers offer a wide range of instance types, with different processing and interconnection speeds, as well as different prices. Furthermore, the tasks of many parallel applications show different computational demands. These differences can be exploited for improving the cost-efficiency of parallel applications in many cloud environments, by matching application requirements to different instance types. A heterogeneous cloud is an interesting solution for the execution of large parallel applications, as these applications typically have heterogeneous computational demands, with some tasks performing more work than others. In such a scenario, tasks that perform more work can be executed on faster but more expensive instances, while tasks that perform less work can be executed on slower and more cost-efficient instances.

In this Chapter, we introduce the *CloudHet* mechanism. CloudHet aims to solve the problem of which cloud instance combination is better to execute the application. Its focus is on building clusters to execute MPI parallel applications optimizing the application's cost-efficiency. It takes advantage of the cloud heterogeneity to create heterogeneous environments to match the application processes demands better. First, we will present the methodologies we used to define the profile of cloud instances and the methodologies used to define the application profile. Following that, CloudHet is presented with its features and internal logic. Finally, CloudHet is validated with real HPC applications as well as our benchmark ImbBench.

5.1 Cloud Instances Analysis and Profiling

Most public cloud providers offer a wide variety of instance types with different characteristics and prices. We started with an analysis of homogeneous cloud clusters, that is, clusters that are composed of cloud instances of the same type, in terms of their computational performance and cost. The purpose is to determine the price of each execution in such environments to motivate the use of heterogeneous configurations.

5.1.1 Platforms and Experimental Design

Experiments in this Section were performed with the following methodology. First, we selected a group of homogeneous clouds to run performance and cost tests, focusing on instance types with similar configurations to provide a better comparison. Based on these results, we selected instance types that are used for the rest of this work. Second, we verified the computational load profile of several parallel applications by measuring the number of instructions per task. These results were used as an input to design CloudHet as well.

All experiments were performed on the Microsoft Azure public cloud, which was selected since it has the most significant number of instance types among the leading cloud providers. We selected the A10, D4 v2, F8, G3, and H8 instances of Azure to verify their efficiency in terms of performance and cost. All instance types consist of eight cores, which is the most common instance size in Azure. The multi-instance experiments use eight nodes, for a total number of 64 cores in all cases.

The software environment consists of Ubuntu server 16.04, with Linux kernel 4.4. We use Open MPI (GABRIEL et al., 2004) 1.10.2 as the parallel runtime environment. All applications were compiled with gcc/gfortran 5.4.0, using the `-O2` optimization level.

We measured the raw computational performance of a single instance of each type with the High-Performance Linpack (HPL) benchmark (PETITET et al., 2012). To compare the cost of the machines, we calculate the price of a TFlop based on performance results with the Linpack and the price for each instance. Table 5.1 presents an overview of the characteristics of these instance types as well as the performance and cost results.

The location used to allocate the machines on Microsoft Azure was "West US". All experiments were executed 10 times. Applications were configured to run with 64 ranks (1 rank per core). In our experiments, we did not notice significant differences between executions at different times of the day and between different allocations of instances.

Table 5.1: Characteristics of the Azure instance types. Instance types that are evaluated in depth in this paper are marked in **bold**.

Instance name	Price/hour	Linpack perf. (GFLOPS)	Price/TFLOP
A10	US$ 0.780	155.35	US$ 5.02
D4 v2	US$ 0.559	265.00	US$ 2.11
F8	US$ 0.513	246.10	US$ 2.08
G3	US$ 2.440	280.17	US$ 8.71
H8	US$ 0.971	324.34	US$ 2.99

Experiments were performed with a variety of MPI-based parallel applications. We used the MPI implementation of the NAS Parallel Benchmarks (NPB) (BAILEY et al., 1991; SAINI; BAILEY, 1996), version 3.3.1. Experiments were performed with input class *C*, which represents a medium-large input size. The *DT* application was not used because it needs at least 85 MPI processes to execute using input size *C*. Since we do not want to overload the nodes, we decided not to use it in this work.

5.1.2 Experimental Results

The execution performance and standard deviation of NAS benchmarks are shown in Figure 5.1. The performance results are the mean of all the executions. As expected, the instances show behavior that correlates with our earlier evaluation of the Linpack performance.

The cost per execution of the NAS benchmarks is shown in Figure 5.2. The cost was calculated by multiplying the price per second of each instance with the execution time of the benchmark.

The G3 instance presented a high-performance score by using LINPACK; however, with the NAS benchmarks, its performance was adherent with instances with low performance. Moreover, it presents the highest execution cost of all benchmarks. Despite its high performance, G3 has a cost that is several times bigger than the other instances. The A10 instance presented the second highest cost per execution. Furthermore, it does not present an excellent execution performance. Due to the relation between the price and performance, these two instances were discarded as candidates for our evaluation.

The F8 and H8 instances presented similar execution performance, with slightly better results for H8 in LU and SP tests, and F8 performing slightly better in FT. Both are the machines with better overall performance in the NAS benchmarks execution. In terms of LINPACK measurement, the H8 was the instance with the highest performance, by far. However, in terms of price, the F8 instance presented better results than the H8 instance in all the NAS applications, showing that F8 has better cost-efficiency. The results of the price per TeraFlop presented better results for F8 as well. Therefore, the F8 was one of the instances selected for the rest of our evaluation. The remaining instance is the D4 v2, which presented a balanced price/performance relation; due to this, it was the second instance chosen to continue our evaluation.

Based on these preliminary results, we selected two instances, D4 v2 and F8, as

Figure 5.1: Performance results of the NAS benchmarks (64 ranks) on homogeneous 8-node cloud systems.

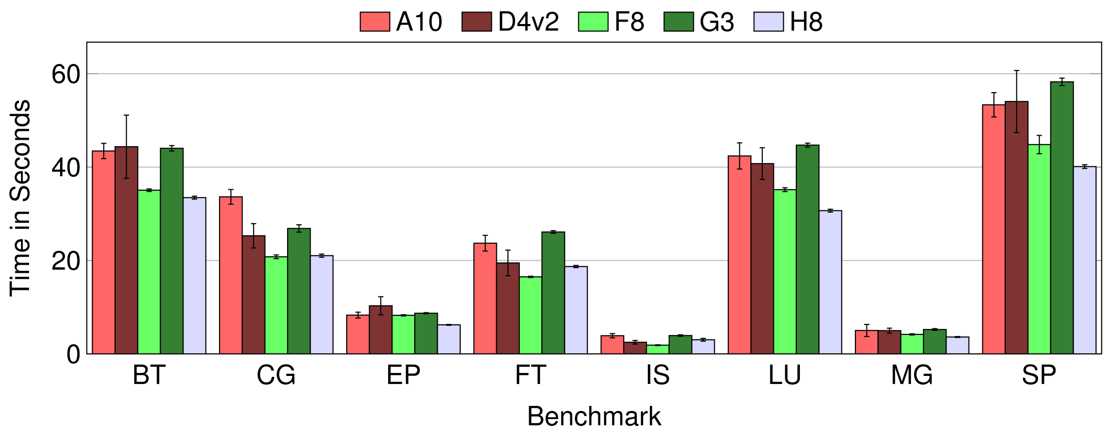
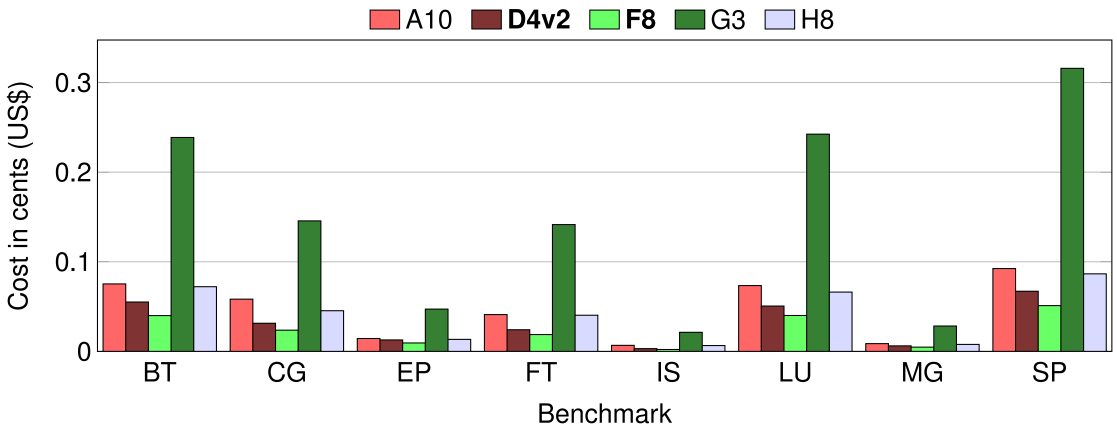


Figure 5.2: Cost per execution (in US$ cents) of each NAS benchmark on homogeneous eight-instance cloud systems.



the target instances for our analysis in this thesis. They were chosen because they have the best relation between cost and performance among all the types we evaluated. Furthermore, despite having differences in the price per hour and performance, their relation between cost and performance is very similar and therefore provides an attractive tradeoff between price and speed. For simplicity, the D4 v2 instance will be referred to as D4 in the rest of this chapter.

5.2 Application Load Imbalance Analisys

Another important aspect of our research is that parallel applications are executed by splitting the work into several tasks. Moreover, these tasks have different computational demands. This aspect will be evaluated in this Section. The applications we used to conduct load imbalance investigation were NAS, *BRAMS*, and *Alya* suite. We first present and characterize these applications and follow up with an in-depth analysis of each application.

We used the MPI version of the Numerical Aerodynamic Simulation Parallel Benchmarks, or usually NAS, version 3.3.1. NAS was designed to compare the performance of parallel systems and is composed of kernels which calculate Computational Fluid Dynamics (CFD), derived from important problems in aerophysical applications. These CFD simulations reproduce a large portion of the data movement, and computation found incomplete CFD codes. Table 5.2 presents an overview of the NAS applications.

BT, *SP*, and *LU* have their parallelism expressed as the division of the spatial domain while sharing data in the borders of each subdomain, showing linear access to data. The *CG* application presents random data access characteristics. The *MG* application works on contiguous data, varying the access pattern in each computing step, going from unaligned accesses to aligned accesses as the grid is refined. Thus, this application can be considered as an intermediate between the regular access applications (*BT*, *LU*, and *SP*) and the irregular access application (*CG*). *EP* is completely parallel and has few memory accesses. The performance of this application can be used as a reference for the peak computational performance of a system. *IS* only uses integer calculations and not floating-point calculations as the other benchmarks. It also has a medium amount of communication between the tasks. *FT* performs several all-to-all communication.

Alya is a simulation code for multi-physics problems, based on a variational multiscale finite element method for unstructured meshes. It is used in areas such as wind energy, aerospace, oil and gas, biomechanics and biomedical research, environment, and the automotive industry. It is developed at Barcelona Supercomputing Center, written in Fortran 90/95, combining MPI and OpenMP. Parallelization of the work is mainly performed using MPI; the original mesh is partitioned into sub-meshes that are executed for

Table 5.2: Overview of the NAS Benchmarks used in the Evaluation.

Name	Description	Focus	Language
BT	Block Tridiagonal	Floating point performance	Fortran
CG	Conjugate Gradient	Irregular communication	Fortran
EP	Embarrassingly Parallel	Floating point performance	Fortran
FT	Fast Fourier Transform	All to All communication	Fortran
IS	Integer Sort	Integer performance	C
LU	Lower and Upper Triangular	Regular communication	Fortran
MG	Multigrid	Regular communication	Fortran
SP	Scalar Pentadiagonal	Floating point performance	Fortran

MPI processes (VÁZQUEZ et al., 2014; VáZQUEZ et al., 2016).

BRAMS (Brazilian developments on the Regional Atmospheric Modeling System) (FREITAS et al., 2009) is the extended version of the RAMS (Regional Atmospheric Modeling System) (PIELKE et al., 1992) weather prediction model. It is one of the most widely employed numerical models in regional weather centers in South America.

To evaluate the load imbalance of the parallel applications, we used the `perf` tool (MELO, 2010), measuring the number of instructions executed by each MPI rank. All applications were executed with 64 ranks. We determine the imbalance with the *percent imbalance metric,* λ (PEARCE et al., 2012), which is calculated with the following equation:

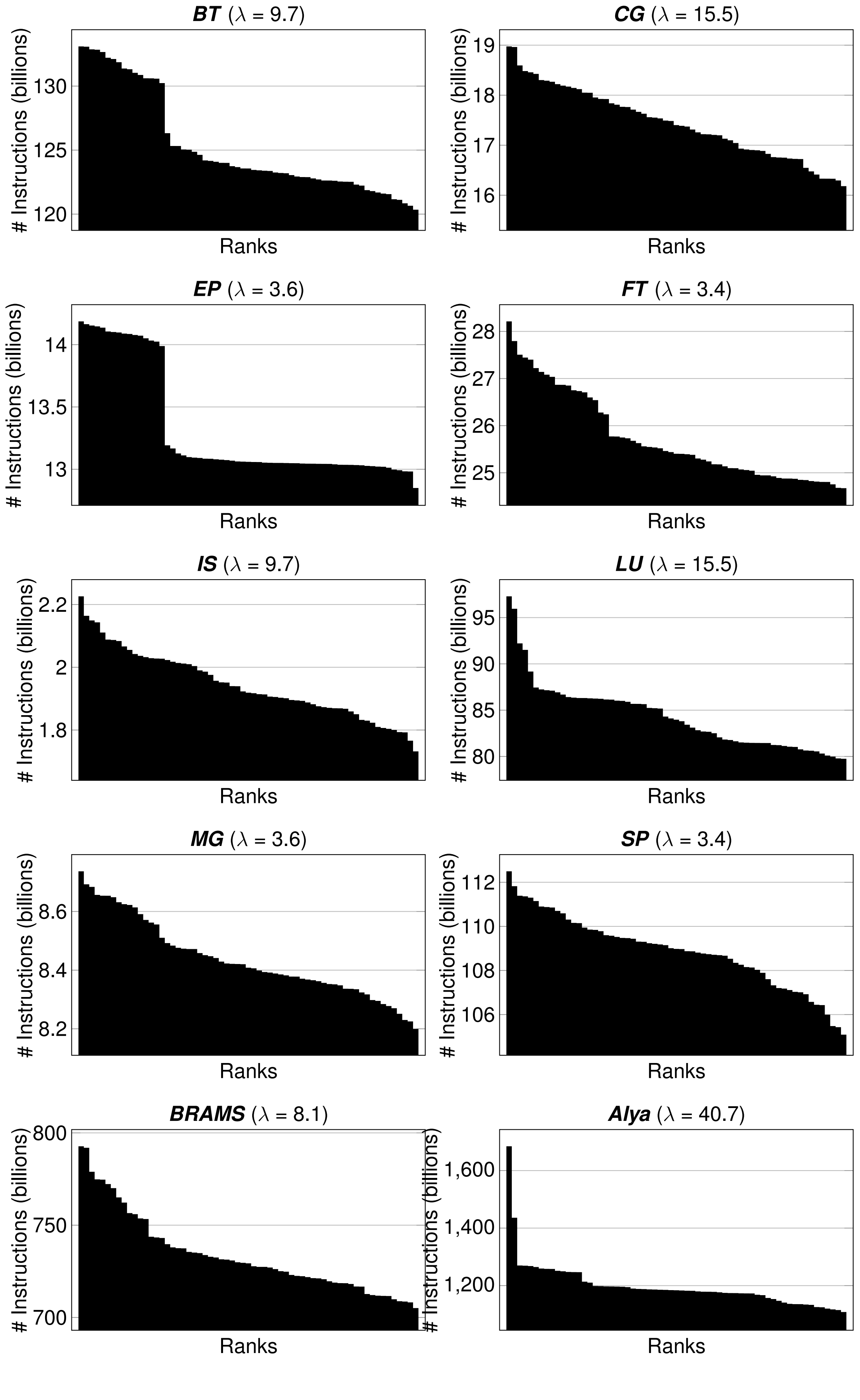
$$\lambda = \left(\frac{\max(L)}{\text{avg}(L)} - 1 \right) \times 100\% \tag{5.1}$$

In the equation, L represents the vector that contains the number of instructions of each rank. A value of 0 indicates that each rank executed the same number of instructions, while higher values of λ indicate an imbalance.

The results of this experiment are shown in Figure 5.3. In the Figure, we show the number of instructions executed by each rank for each benchmark, sorted in descending order. The results show varying degrees of imbalance between the applications. In general, the imbalance is considerable, with differences between the minimum and maximum numbers of instructions for each benchmark reaching up to 35% in the case of *Alya*. Some applications, such as *FT*, *SP*, and *Alya*, have considerable sequential parts that increase their imbalance. Others, such as *BT*, *EP*, and *FT*, show two distinct levels of numbers of instructions executed by the ranks. Executing the applications multiple times results in a very similar load profile, with similar loads for each rank. This means that for these applications, the profile can be reused for future executions.

The results in this Section have shown that performance and cost vary widely between instance types. Although more expensive instance types are generally faster than cheaper ones, however, sometimes, their performance does not scale linearly with the higher prices, which motivates using these more expensive instances only for those tasks that can benefit from the performance increase. We have also seen that most of the parallel applications we evaluated have heterogeneous computational demands, which makes them interesting candidates to execute on heterogeneous environments.

Figure 5.3: Load distribution of the benchmarks, running with 64 ranks. Each bar corresponds to the number of instructions executed by a rank. Ranks are sorted according to the number of instructions executed.



5.3 The CloudHet Mechanism

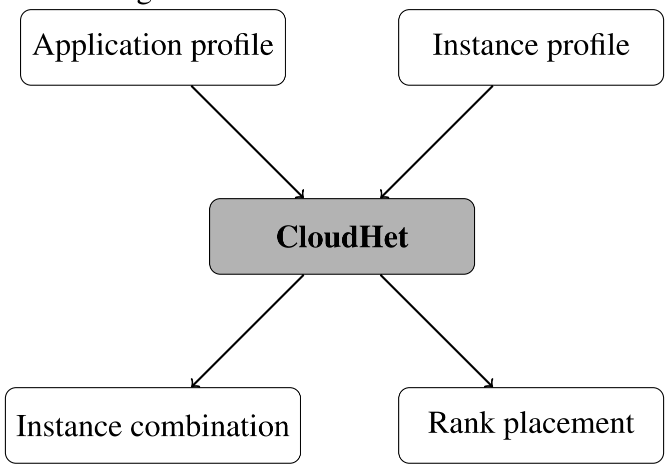
This Section describes our proposed mechanism, *CloudHet*, which automatically leverages the heterogeneity for improved cost-efficiency. Given an application profile of an application and the available cloud instance types, CloudHet calculates how many instances of each type should be used, and which MPI ranks should be executed on each instance. Thus, CloudHet solves the problem of choosing which instances to use for a given application executing in the cloud. A high-level view of the CloudHet operation is shown in Figure 5.4.

CloudHet receives as input the load profile of the application, and the instance profile of the target instance types.Currently, two different instance types are taken into account, which we refer to as HI and LO, corresponding to the D4 and F8 instances in our experiments. We focus on two instance types since many of the applications show a two-level-like load distribution. The mechanism outputs the instance combination, that is, how many instances of each type should be used, and which ranks should be placed on each instance. In the next subsection, we present details about the input, output, and how CloudHet makes the matching.

5.3.1 CloudHet Inputs

To generate the *application profile* of the target application, the user needs to use the standard Linux `perf` tool (MELO, 2010). Perf can generate a complete profile of the application, with information on how many floating-point or integer operations the appli-

Figure 5.4: Overview of CloudHet.



cation performs, how many accesses to the main memory are made, among others. Our experiments using a fine-grained differentiation did not improve the results of CloudHet. Due to that, we focus solely on the total number of instructions per rank. This application profile usually needs to be generated only once for a given set of input data and the number of ranks, due to the persistence principle (KALÉ, 2002). This input itself is a vector with all the ranks of the applications and their number of instructions.

The *Instance profile* is composed of two main values, the performance of the instance and their price per hour. The price per hour is obtained by consulting the cloud provider. The performance of the instance is obtained by using the High-Performance LINPACK (PETITET et al., 2012), expressed in FLOPS.

5.3.2 CloudHet Outputs

The first output of CloudHet is to determine the *Instance combination*. We group the sorted list of ranks into groups of the same size as the number of cores per instance and calculate the cumulative load of each group. CloudHet then iterates over the list of groups and calculates the load ratio between subsequent groups. Until the ratio reaches a threshold, groups will be executed on HI instances. When the ratio is above the threshold, all subsequent groups will be executed on LO instances. The threshold is determined by the performance ratio of the instance types. Therefore, this output is the number of instances of each type that needs to be used.

After determining the number of HI and LO instance types, CLoudHet assigns ranks to the instances in the following way. It iterates over the sorted list of ranks and assigns ranks to instances sequentially, starting with the ranks with the highest load, which are assigned to the HI instances. As soon as one instance has the maximum number of ranks assigned to it, the mechanism continues the rank placement with the next instance, until all ranks are assigned. As the final output, CloudHet creates a rank file that specifies the task to instance assignment.

5.3.3 CloudHet Mechanism

This Section details the CloudHet internal operation; Algorithm 1 expresses its procedures. We will detail the algorithm by providing an example of our benchmark eval-

Algorithm 1: Algorithm of CloudHet

```
    Input: List AP, List IP, Integer NC
    Output: RankMap, Map of MPI Integer Rank to String Instance ID
 1  begin
 2      $IP \longleftarrow sort(IP, keyIntegerPerformance)$
 3      $AP \longleftarrow sort(AP, keyIntegerLoad)$
 4      $RankLeaders \longleftarrow \emptyset$
 5      $nallocs \longleftarrow length(AP) \div NC$
 6      $indexrank \longleftarrow 0$
 7      for $index \longleftarrow 0$ to $nallocs$ do
 8          define the leader ranks of each instance allocation
             $RankLeaders[index] \longleftarrow AP[indexrank]$
 9          $indexrank \longleftarrow indexrank + NC$
10      end
11      $Selected \longleftarrow \emptyset$
12      $CostPivot \longleftarrow RankLeaders[0].Load \div IP[0].Performance$
13      $Selected[0] \longleftarrow IP[0].ID$
14      for $Leader \longleftarrow 1$ to $nallocs$ do
15          $MinDiff \longleftarrow \infty$
16          for $Profile \in IP$ do
17              $NewDiff \longleftarrow |CostPivot - (RankLeaders[Leader] \div$
                 $IP[Profile].Performance)|$
18              if $NewDiff < MinDiff$ then
19                  $MinDiff \longleftarrow NewDiff$
20                  $Selected[Leader] \longleftarrow IP[Profile].ID$
21              end
22          end
23      end
24      $RankMap \longleftarrow \emptyset$
25      for $Leader \longleftarrow 0$ to $nallocs$ do
26          for $indRank \longleftarrow 0$ to $NC$ do
27              $RankMap[AP[Leader + indRank].Rank] \longleftarrow$
                 $Selected[Leader]$
28          end
29      end
30  end
```

uation shown in Section 4.4, i.e, the ImbBench Four-Level Pattern executed in Microsoft Azure.

There are three inputs for CloudHet: a list of Application Profile (AP), a list of Instance Profile (IP), and the Number of Cores (NC) of the considered instances. The *Application Profile* is a list of tuples, where each tuple contains a MPI rank and its number of instructions. In our example, we executed ImbBench by using 64 MPI processes so that we will have an Input with 64 tuples associating each rank with its number of instructions. The Table 5.4 illustrates this input value; we could observe that the MPI rank 0 presents around 8 billion instructions, the MPI rank 1 presents around 6 billion instructions, the MPI rank 2 presents around 4 billion instructions, and the MPI rank 3 presents around 2 billion instructions; this pattern repeat until the 63 MPI rank. Our purpose here is to demonstrate the concept, so we do not present all 64 lines of the entry here. The *Instance Profile* is a list of profiles for all the instances that the user wants to consider. Each position of this list contains a tuple containing three fields: the instance name, the instance performance, and the instance cost. The instance name is the ID that identifies the instance type, the instance performance is its Linpack performance in FLOPS, and the instance cost is the price charged by the provider to use the Instance. The Table 5.3 shows the input used in the ImbBench evaluation. Finally, the *Number of Cores* is self-explained, it is the number of cores of the instances considered by the user. It is essential to remark that all the instances must have the same number of cores.

The algorithm starts with the data preparation, where the Instance Profile (line 2) and Application Profile (line 3) are sorted. The IP is sorted by calculating the cost-efficiency, which is obtained through the division of performance by price, the current version of CloudHet is considering just the two instances with better cost-efficiency, the HI and LO instances. By sorting the AP using the number of instructions as key, we obtain a decrescent list of processes' instructions, which represents the computing demand of

Table 5.3: Instance Profile (IP) Input example.

Instance	Performance (GFLOPS)	Cost
D16	247	0.936
D5	280	1.117
E16	254	1.186
F16	263	0.997
G4	417	3.072
H16	657	1.941
L16	406	1.376

each process. We also sort the instances according to their performance. The next relevant step is the determine the number of chunks (variable *nallocs*), which is the number of instances that will be allocated. We divide the number of processes for the number of cores per instance to obtain this number. In our example, we divide 64 by 16 and obtain four chunks.

We consider the process with the largest demand as the chunk demand, dubbing it as the chunk's leader. The most demanding process is the relevant processing demand of the chunk, as each process of the chunk is executed in a dedicated core of the instance, and all the other processes have equal or less demand than the leader. We populate a list that contains all the chunk's leaders using the loop from line 7 to line 10. The loop traverses the sorted list AP in "number of cores" increments, as all following ranks of a rank selected to be a leader will be allocated in the same instance as the leader.

The next step is to calculate the ratio between the chunk's demand and the power of the instance; this is calculated in the from line 12 to line 23. As the purpose of CloudHet is to recommend a heterogeneous configuration with more cost-efficiency than an entirely homogeneous one, the first allocation is always allocated in the instance with the most processing power. Lines 12 and 13 correspond to this selection, and the ratio between the leader demand and the processing power of the most powerful instance is calculated. We consider this value the *CostPivot*, and all other ratios are defined in relation to it. The following nested loops, lines 14 to 23, present a greedy algorithm that calculates the ratio between each of other leaders and all the instance types from the IP input. The loops try to find, for each leader, the instance allocation whose ratio is closest to the CostPivot's ratio. For each instance, it defines that instance's distance to the CostPivot; thus, for each leader, the algorithm finds the instance with the minimum distance from the CostPivot's ratio. Finally, in the loop from lines 24 to 29, the algorithm formats the return result as a

Table 5.4: Application Profile (AP) Input example.

Rank	Number of Instructions
0	8,064,733,231
1	6,133,903,029
2	3,969,654,123
3	2,070,761,324
4	8,386,539,374
...	...
61	6,149,751,975
62	3,966,275,098
63	2,078,499,734

map whose keys are the ranks of the application and the values are the instances in which each rank should be allocated.

5.4 Results

This Section presents the evaluation methodology used to validate the CloudHet proposal. We used a set of real HPC applications to perform our evaluation. We validated both CloudHet and the heterogeneous configurations by using a whole set of instance conbinations.

5.4.1 Methodology

For our experiments, we use the chosen Azure instance types, D4, and F8. All experiments use eight instances, and we build clusters with a total of 64 cores. We vary the mix of instances between:

- Fully homogeneous: all eight instances are of the same type, D4 or F8.
- Heterogeneous to varying degrees: 1–7 instances of each type, totaling eight instances.

We use the cost-efficiency results to compare the different configurations. In the figures illustrating the results, the line represents the cost-efficiency when varying the mix of instance types. To calculate the cost-efficiency metric, we use the following equation.

$$\textit{cost-efficiency} = \textit{execution time} \times \textit{price of execution} \tag{5.2}$$

Lower values of the metric indicate a higher cost-efficiency, this metric is based on our previous work (ROLOFF et al., 2012a)

Machines of all instance types were only allocated once and not reallocated between executions. Further experiments after deallocating and allocating new instances of the same types (not shown in the thesis) resulted in quantitatively and qualitatively similar behaviors. Results show the average values of 10 executions. We begin with a discussion of the NAS benchmarks, followed by the *BRAMS* and *Alya* applications.

5.4.2 The NAS benchmarks

The cost-efficiency results of the NAS benchmarks are shown in Figure 5.5.In the Figure, each line represents the cost-efficiency for one metric when varying the mix of instance types. The y axes show the values of the metric, while the x axes indicate the mix of instance types in the form a/b, where a represents the number of D4 instances and b represents the number of F8 instances. 0/8 and 8/0 are the homogeneous clouds, while 1/7 – 7/1 are heterogeneous instances.

The most cost-efficient instance combination is marked with a dashed circle (◌). The results of the instance combination calculated by CloudHet mechanism are marked with an unbroken circle (○).

Several interesting results can be pointed out. First of all, heterogeneous clouds are the most cost-efficient environments for the majority of the benchmarks. Heterogeneous environments are beneficial for five out of the eight benchmarks (*BT*, *EP*, *FT*, *MG*, and *SP*), while homogeneous environments are more appropriate for *CG*, *IS*, and *LU*. The five benchmarks have an increased cost-efficiency between 3.0% (*SP*) and 18% (*FT*) compared to the best cost-efficiency of a homogeneous environment. These results show that cost-efficiency can be improved substantially via heterogeneous clouds.

Although not achieving the optimal results for all applications, CloudHet can result in substantial cost-efficiency improvements close to the optimum in most scenarios. This shows that profiling all possible instance combinations is not required in order to reduce costs. Another important result is that almost all heterogeneous and homogeneous instance combinations are selected as the most cost-efficient environment in at least one experiment. This shows that simple policies that do not take the specific characteristics of the environment and application behavior into account can not result in optimal cost-efficiency.

Finally, it is necessary to point out that the most cost-efficient mix of instance types not only changes between different benchmarks but also between different metrics. A user that is interested in a higher cost-efficiency might choose a different mix of instances to increase cost-efficiency while slightly reducing performance.

An overview of the cost-efficiency results for the NAS benchmarks is shown in Table 5.5. The table shows for each benchmark the most cost-efficient configuration and the cost-efficiency gains of this configuration compared to the best homogeneous environment for each benchmark.

Figure 5.5: Cost-efficiency results for the NAS benchmarks on the D4 and F8 instances. Lower values indicate a higher cost-efficiency. The highest cost-efficiency is marked with a dashed circle ◌. The results of our mechanism are marked with an unbroken circle ○.

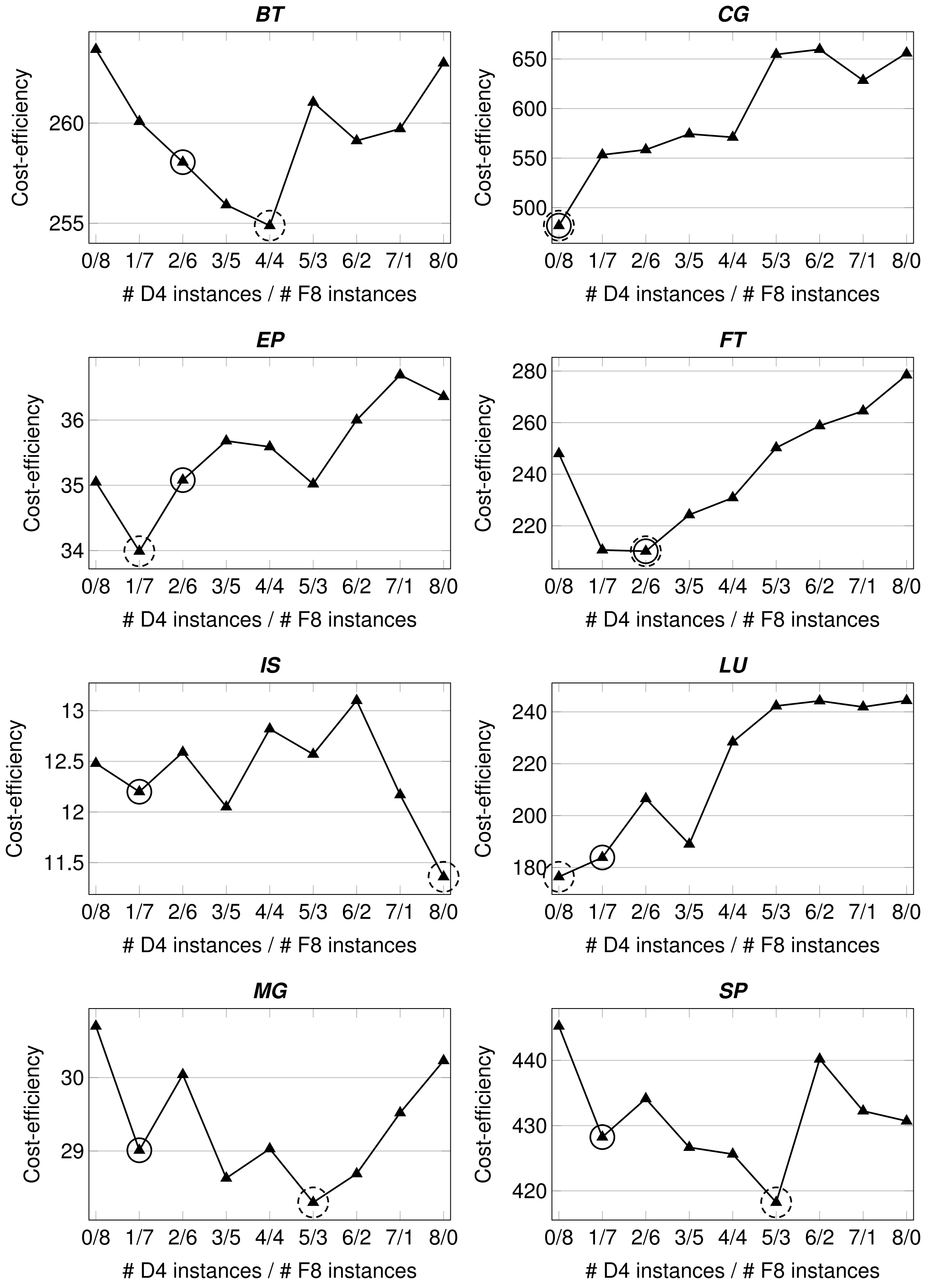


Table 5.5: Cost-efficiency gains compared to the best homogeneous configuration, for the best case and the CloudHet.

	Best case		Mechanism	
App.	Config.	Gains	Config.	Gains
BT	4-D4 4-F8	3.2%	2-D4 6-F8	1.9%
CG	0-D4 8-F8	0.0%	0-D4 8-F8	0.0%
EP	1-D4 7-F8	3.1%	2-D4 6-F8	-0.0%
FT	2-D4 6-F8	18.0%	2-D4 6-F8	18.0%
IS	8-D4 0-F8	0.0%	1-D4 7-F8	-6.9%
LU	0-D4 8-F8	0.0%	1-D4 7-F8	-4.0%
MG	5-D4 3-F8	6.8%	1-D4 7-F8	4.2%
SP	5-D4 3-F8	3.0%	1-D4 7-F8	0.5%
BRAMS	1-D4 7-F8	0.8%	1-D4 7-F8	0.8%
Alya	7-D4 1-F8	0.6%	7-D4 1-F8	0.6%
avg.	—	4.7%	—	1.5%

Table 5.6: Performance gains compared to the fastest homogeneous configuration.

App.	Best case	Mechanism
BT	-1.1%	-4.4%
CG	0.0%	0.0%
EP	-0.7%	-2.8%
FT	20.6%	20.6%
IS	0.0%	-13.6%
LU	0.0%	-3.0%
MG	-5.6%	-3.3%
SP	-0.2%	-6.7%
BRAMS	1.9%	1.9%
Alya	-0.4%	-0.4%
avg.	1.5%	-1.2%

The performance analysis of the heterogeneous allocations is also an important aspect for the user. An overview of the performance gains compared to the fastest homogeneous execution is shown in Table 5.6.

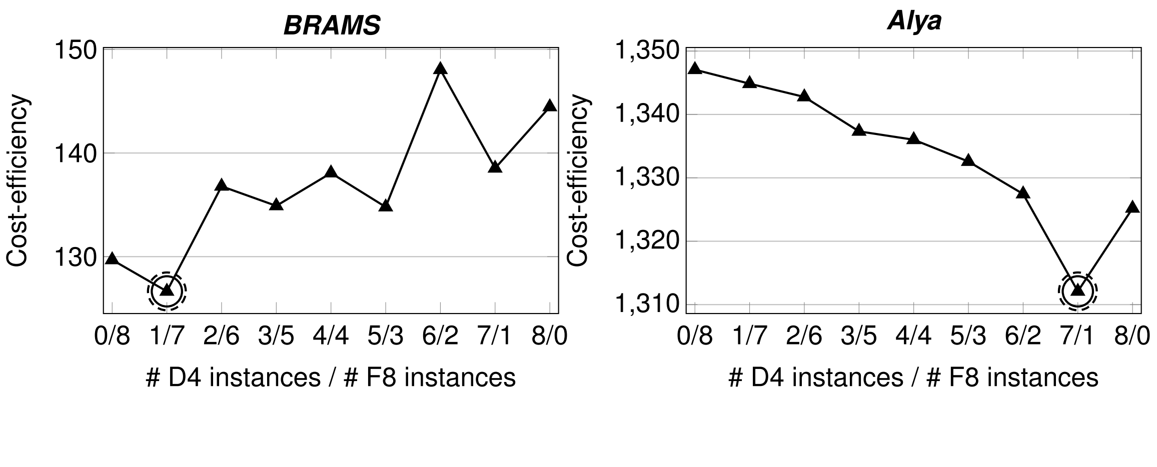
There are four benchmarks that presented performance losses (*BT*, *EP*, *MG*, and *SP*), between 0.2% (*SP*) to 5.5% (*MG*). The *FT* benchmark presented gains of 20% when compared to the homogeneous allocations. It is important to remark that when comparing the performance loss with the cost-efficiency gain, the heterogeneous allocations present better ratios for all the cases. This means that the performance decrease is smaller than the cost-efficiency gain. The best ratios were obtained with *SP* (0.2% loss - 3.0% gain) and *EP* (0.7% loss - 3.1% gain).

5.4.3 *BRAMS* and *Alya*

The cost-efficiency results for the two scientific applications, *BRAMS*, and *Alya*, are shown in Figure 5.6. The results echo our analysis of the NAS benchmarks. Both applications can benefit significantly from a heterogeneous environment and show significant cost-efficiency improvements in almost all cases. When comparing the results of *BRAMS*, we observed that *BRAMS* presented a cost-efficiency gain of 4.6%. We can conclude that *BRAMS* needs are better served with a heterogeneous execution environment. Remarkably, the best configuration for all the metrics was the one with one D4 instance and seven F8 instances. This means that the slightly better cost-efficiency of D4 over F8 was enough to present better performance and a huge decrease of the cost for the cloud tenant. When we relate the cost results with the performance results we observed that the user will decrease the price.

When analyzing the results of *Alya*, we observed a performance loss of 0.4%, while presenting a cost-efficiency gain of 1.8% for the cloud tenant. Observing the load distribution of *Alya* in Figure 5.3, we note that the imbalance of Alya is high, with a few processes executing much more operations than the average. The instances used in our experiments, D4 and F8, present a close performance. Due to the *Alya* load distribution, we can conclude that it could benefit from instances with higher differences between them, and from mixing more instances types.

Figure 5.6: Cost-efficiency results for *BRAMS* and *Alya* on the D4 and F8 instances. Lower values indicate a higher cost-efficiency. The highest cost-efficiency is marked with a dashed circle ◌. The results of our mechanism are marked with an unbroken circle ○.



5.5 Concluding Remarks

The cloud has become an interesting environment for the execution of parallel applications due to the easy and flexible availability of different instance types that vary in performance and price. Most current cloud deployments for parallel applications are homogeneous; that is, they are composed of several instances of the same type. Since the computational demands of parallel applications are not uniform in most cases, such a heterogeneous cloud can better match the requirements of the application, improving the price and cost-efficiency of the execution. In this Chapter, we validated our mechanism of deployment that is based on heterogeneous instances of different types.

We introduced the *CloudHet* mechanism, which uses the load imbalance of the application combined with cloud heterogeneous price/performance trade-offs to select a cost-efficient cloud instance allocation. Our evaluation with MPI-based applications on an Azure cloud shows that this efficiency can be improved significantly, by up to 42.3%, depending on the load imbalance of the application and the employed metric, while maintaining similar performance. Gains achieved by our mechanism's allocation were close to the optimal allocation in most cases. Therefore, we have shown that a user can obtain improvements from heterogeneous execution without requiring time-consuming evaluations of all possible instance combinations.

6 CONCLUSION

The use of Cloud Computing for High-Performance Computing has produced outstanding results, both in terms of use as an environment used to extend traditional environments and as a replacement for traditional clusters. The public Cloud has several characteristics that can be beneficial to the user, among which two stand out: the pay-per-use billing model and elasticity. Another feature that is inherent in the cloud computing model is heterogeneity, where several instances are available with different configurations and prices. To fully benefit from these characteristics, it is essential to analyze the application processing demand and use this information to create cloud environments that match the application behavior, improving the allocation's cost-efficiency. This thesis advances the field of HPC in the Cloud in two ways. First, we introduce and validate the use of heterogeneous configurations to execute HPC applications improving the cost-efficiency. Second, we have introduced a mechanism that automates the choice of instances to use according to application demand.

6.1 Summary of Main Results

In this thesis, we presented an analysis of heterogeneous clouds and a methodology to decide which cloud instances are more suitable for a given parallel application, based on its computing demand. We developed the CloudHet mechanism, a mechanism to select and map cloud instances to efficiently execute an application based on the application's processing demands. CloudHet does not require any change to the application. We introduced the CDP metric, which is a metric to compare the cost-efficiency of two different cloud environments for a given application directly. Through it, a user can determine whether they prefer a better performing environment or a lower-cost one. We introduced the ImbBench as well, which is a proto benchmark that simulates a parallel application with several load imbalance patterns. It was used to validate the heterogeneous cloud proposal as a viable execution environment.

Returning to the hypothesis that was tested in this thesis, which stated: *The combination of (1) application load imbalance and (2) heterogeneous machines in the cloud can be exploited to reduce the execution cost without significant performance loss.* Our results have shown that according to the application behavior, it is possible to achieve significant cost reduction by using heterogeneous configurations compared to homogeneous configurations. We achieved gains in cost reduction up to 63%. Besides, the performance loss keeps a tolerable rate, up to 7%; and the tradeoff between cost-efficiency and performance lost is beneficial to the user. Due to this, we conclude that the hypothesis is correct.

The approach we use in this thesis is also a way to solve the problem of choosing which cloud instance is the most suitable to be used to execute a given application. Our approach has some restrictions as the tested applications are HPC applications, and our approach has not been tested for use by generic applications. However, it can be extended to cover a broader range of applications.

6.2 Released Software

This section briefly describes the software that was released as part of this thesis. All software is released as open-source and licensed under the GPL-v2 license.

1. **ImbBench**: It is a set of MPI-based applications that simulate several load imbalance patterns in terms of process loads. It was used in Chapter 4 to validate the proposal to use heterogeneous configurations to execute parallel applications. It is available at <https://github.com/Roloff/ImbBench>.
2. **CloudHet**: It is the implementation of the mechanism defined on Chapter 5, which is responsible for defining the instances that need to be used to execute the application. It was used after the validation of the heterogeneous configurations and to solve the cloud instance selection. It is available at <https://github.com/Roloff/CloudHet>.

6.3 Research Perspectives

The following opportunities were identified to extend the research presented in this thesis.

1. **Instance Profiling**. This thesis focuses mainly on CPU performance and the main memory speed. Because of this, the instance profiling was performed by considering these two aspects. However, both network interconnection performance and I/O performance are important aspects to HPC applications and need to be considered as

well. We could extend this aspect of our thesis in two directions. By using consolidated benchmarks for network and I/O, such as as (CAPPS; NORCOTT, 2008; INTRODUCING..., 2019). Alternatively, by creating a set of microbenchmarks that further evaluate instance-specific aspects such as integer operations, cache speed, and more, creating a more fine-grained profile of cloud instances.

2. **Network and I/O Performance**. In this thesis, we introduce the ImbBench benchmark, whose scope was to validate the heterogeneous environments. Since the scope of this thesis was on CPU and memory performance, it was designed to evaluate these two aspects only. However, it is necessary to extend it to cover other aspects relevant to HPC, such as the network interconnection and I/O performance. The patterns are already defined; however, the extension needs to focus on defining relevant tests to evaluate these components.
3. **CloudHet Improvements**. The CloudHet was designed to automate the instance selection and prove that it is possible to choose cloud instances to be used automatically. In the current version, CloudHet outputs a list of instances that need to be allocated by the user and a list of task allocation on the instances. One of the future improvements of CloudHet needs to be the user experience. CloudHet could generate a script that automates the allocation for the user, where the user needs to execute the script on his cloud provider. Moreover, it is possible to integrate CloudHet with the main cloud providers and allocate the instances automatically.
4. **Hybrid Scheduler**. In this thesis, we focus on building execution environments in the cloud. A possible scenario of cloud usage for HPC applications is by Cloud Bursting, where the cloud is used to extend the processing capacity of traditional clusters. If the local resources are full and the user applications have time constraints, it can be executed using a public cloud. However, the actual models of cloud bursting do not provide efficient management of the resources, and the user is responsible for determining that the application goes to the cloud. An interesting research opportunity is to propose a whole new scheduler, based on CloudHet, that could execute batch jobs both on traditional clusters and using the cloud, transparently to the user, creating a scheduler agnostic to the environment and with a focus on the applications behavior and user constraints.

6.4 Publications

The following papers (listed in reverse chronological order) were published during the Ph.D. program and contain material that is relevant to this thesis.

1. Eduardo Roloff, Matthias Diener, Luciano Paschoal Gaspary, Philippe Olivier Alexandre Navaux: **"Exploring Instance Heterogeneity in Public Cloud Providers for HPC Applications."** CLOSER 2019: 210-222
2. Anderson Mattheus Maliszewski, Adriano Vogel, Dalvan Griebler, Eduardo Roloff, Luiz Gustavo Fernandes and Philippe Olivier Alexandre Navaux: **"Minimizing Communication Overheads in Container-based Clouds for HPC Applications."** ISCC 2019
3. Otávio Carvalho, Eduardo Roloff, Philippe O. A. Navaux: **"GaruaGeo: Global Scale Data Aggregation in Hybrid Edge and Cloud Computing Environments."** CLOSER 2019: 437-447
4. Emmanuell D. Carreño, Marco A. Z. Alves, Matthias Diener, Eduardo Roloff, Philippe Olivier Alexandre Navaux: **"Multi-phased Task Placement of HPC Applications in the Cloud."** ISPDC 2019: 103-111
5. Eduardo Roloff, Matthias Diener, Luciano Paschoal Gaspary, Philippe Olivier Alexandre Navaux: **"Exploiting Load Imbalance Patterns for Heterogeneous Cloud Computing Platforms."** CLOSER 2018: 248-259
6. Eduardo Roloff, Matthias Diener, Emmanuell Diaz Carreño, Luciano Paschoal Gaspary, Philippe Olivier Alexandre Navaux: **"Leveraging Cloud Heterogeneity for Cost-Efficient Execution of Parallel Applications."** Euro-Par 2017: 399-411
7. Eduardo Roloff, Matthias Diener, Luciano Paschoal Gaspary, Philippe Olivier Alexandre Navaux: **"HPC Application Performance and Cost Efficiency in the Cloud."** PDP 2017: 473-477
8. Otávio Carvalho, Manuel Garcia, Eduardo Roloff, Emmanuell Diaz Carreño, Philippe O. A. Navaux: **"IoT Workload Distribution Impact Between Edge and Cloud Computing in a Smart Grid Application."** CARLA 2017: 203-217
9. Eduardo Roloff, Matthias Diener, Emmanuell Diaz Carreño, Francis Birck Moreira, Luciano Paschoal Gaspary, Philippe Olivier Alexandre Navaux: **"Exploiting Price and Performance Tradeoffs in Heterogeneous Clouds."** UCC (Companion") 2017: 71-76

10. Otávio Carvalho, Eduardo Roloff, Philippe O. A. Navaux: **"A Distributed Stream Processing based Architecture for IoT Smart Grids Monitoring."** UCC (Companion") 2017: 9-14
11. Emmanuell Diaz Carreño, Eduardo Roloff, Philippe Olivier Alexandre Navaux: **"Towards Weather Forecasting in the Cloud."** PDP 2016: 659-663
12. Eduardo Roloff, Emmanuell Diaz Carreño, Jimmy Kraimer Martín Valverde Sánchez, Matthias Diener, Matheus da Silva Serpa, Guillaume Houzeaux, Lucas Mello Schnorr, Nicolas Maillard, Luciano Paschoal Gaspary, Philippe Olivier Alexandre Navaux: **"Performance Evaluation of Multiple Cloud Data Centers Allocations for HPC."** CARLA 2016: 18-32

REFERENCES

ALJAMAL, R.; EL-MOUSA, A.; JUBAIR, F. A comparative review of high-performance computing major cloud service providers. **2018 9th International Conference on Information and Communication Systems, ICICS 2018**, v. 2018-Janua, p. 181–186, 2018.

ALJAMAL, R.; EL-MOUSA, A.; JUBAIR, F. A user perspective overview of the top infrastructure as a service and high performance computing cloud service providers. In: **2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT 2019 - Proceedings**. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2019. p. 244–249. ISBN 9781538679425.

AMAZON Web Services - HPC. 2019. <https://aws.amazon.com/hpc/>. Accessed: 2019-09-19.

ARMBRUST, M. et al. Above the clouds: A Berkeley view of cloud computing. **University of California, Berkeley, Tech. Rep. UCB**, p. 07–013, 2009. ISSN 00010782. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>.

ASANOVIC, K. et al. The landscape of parallel computing research: A view from berkeley. 2006.

AWAD, O. M. O.; ARTOLI, A. M. A.; AHMED, A. H. A. Cloud computing versus in-house clusters: a comparative study. In: **Computer Applications and Information Systems (WCCAIS), 2014 World Congress on**. [S.l.: s.n.], 2014. p. 1–6.

BAILEY, D. H. Vector Computer Memory Bank Contention. **IEEE Transactions on Computers**, C-36, n. 3, p. 293–298, feb 1987. ISSN 00189340. Disponível em: <https://ntrs.nasa.gov/search.jsp?R=19850010340>.

BAILEY, D. H. et al. The nas parallel benchmarks. **The International Journal of Supercomputing Applications**, Sage Publications Sage CA: Thousand Oaks, CA, v. 5, n. 3, p. 63–73, 1991.

BAKER, M.; FOX, G. C.; YAU, H. W. Cluster Computing Review. **Northeast Parallel Architecture Center**, jan 1995. Disponível em: <https://surface.syr.edu/npac/33>.

BARHAM, P. et al. Xen and the art of virtuaBarham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., . . . Warfield, A. (2003). Xen and the art of virtualization. In Proceedings of the nineteenth ACM symposium on Operating systems principles - SOSP ’03 (Vol. 37, p. 1. In: **Proceedings of the nineteenth ACM symposium on Operating systems principles - SOSP ’03**. New York, New York, USA: ACM Press, 2003. v. 37, n. 5, p. 164. ISBN 1581137575. ISSN 0163-5980. Disponível em: <http://portal.acm.org/citation.cfm?doid=945445.945462http://dl.acm.org/citation.cfm?id=945445.945462>.

BASET, S.; SILVA, M.; WAKOU, N. Spec cloud™iaas 2016 benchmark. In: **Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering**. New York, NY, USA: ACM, 2017. (ICPE ’17), p. 423–423. ISBN 978-1-4503-4404-3. Disponível em: <http://doi.acm.org/10.1145/3030207.3053675>.

BELGACEM, M. B.; CHOPARD, B. A hybrid hpc/cloud distributed infrastructure: Coupling ec2 cloud resources with hpc clusters to run large tightly coupled multiscale applications. **Future Generation Computer Systems**, Elsevier, v. 42, p. 11–21, 2015.

BONETI, C. et al. A dynamic scheduler for balancing HPC applications. In: **International Conference for High Performance Computing, Networking, Storage and Analysis**. Ieee, 2008. p. 1–12. ISBN 978-1-4244-2834-2. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5217785>.

BUYYA, R. **High performance cluster computing**. Prentice Hall PTR, 1999. ISBN 0130137847. Disponível em: <https://dl.acm.org/citation.cfm?id=520257http://dpnm.postech.ac.kr/cluster/ppt/Cluster-Tutorial>.

BUYYA, R. Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. In: **2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009**. North-Holland, 2009. v. 25, n. 6, p. 1. ISBN 9780769536224. ISSN 0167739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X08001957>.

CAPPS, D.; NORCOTT, W. **IOzone filesystem benchmark**. 2008.

CARRENO, E. D. et al. Multi-phased Task Placement of HPC Applications in the Cloud. In: **2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)**. IEEE, 2019. p. 103–111. ISBN 978-1-7281-3801-5. Disponível em: <https://ieeexplore.ieee.org/document/8790858/>.

CHANDRA, R. et al. **Parallel programming in OpenMP**. [S.l.]: Morgan kaufmann, 2001.

CHAPMAN, B.; JOST, G.; PAS, R. V. D. **Using OpenMP: portable shared memory parallel programming**. [S.l.]: MIT press, 2008. v. 10.

CHE, S. et al. Rodinia: A benchmark suite for heterogeneous computing. In: **2009 IEEE International Symposium on Workload Characterization (IISWC)**. [S.l.: s.n.], 2009. p. 44–54.

CHOHAN, N. et al. See spot run: Using spot instances for mapreduce workflows. **HotCloud**, v. 10, p. 7–7, 2010.

CHURCH, P. et al. Toward exposing and accessing hpc applications in a saas cloud. In: IEEE. **2012 IEEE 19th International Conference on Web Services**. [S.l.], 2012. p. 692–699.

COOK, S. **CUDA programming: a developer's guide to parallel computing with GPUs**. [S.l.]: Newnes, 2012.

COSTA., B. G. S. et al. Performance and cost analysis between on-demand and preemptive virtual machines. In: INSTICC. **Proceedings of the 8th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,**. [S.l.]: SciTePress, 2018. p. 169–178. ISBN 978-989-758-295-0.

COUTINHO, R. et al. A dynamic cloud dimensioning approach for parallel scientific workflows: a case study in the comparative genomics domain. **Journal of Grid Computing**, v. 14, n. 3, p. 443–461, Sep 2016. ISSN 1572-9184. Disponível em: <https://doi.org/10.1007/s10723-016-9367-x>.

CRAGO, S. P.; WALTERS, J. P. Heterogeneous cloud computing: The way forward. **Computer**, v. 48, n. 1, p. 59–61, Jan 2015. ISSN 0018-9162.

CUNHA, R. L. et al. Job placement advisor based on turnaround predictions for HPC hybrid clouds. **Future Generation Computer Systems**, Elsevier B.V., v. 67, p. 35–46, feb 2017. ISSN 0167739X.

DONG, D. et al. Managing and unifying heterogeneous resources in cloud environments. In: INSTICC. **Proceedings of the 7th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,**. [S.l.]: SciTePress, 2017. p. 143–150. ISBN 978-989-758-243-1.

DONGARRA, J. J. The linpack benchmark: An explanation. In: SPRINGER. **International Conference on Supercomputing**. [S.l.], 1987. p. 456–474.

DONGARRA, J. J.; LUSZCZEK, P.; PETITET, A. The linpack benchmark: past, present and future. **Concurrency and Computation: practice and experience**, Wiley Online Library, v. 15, n. 9, p. 803–820, 2003.

DONGARRA, J. J. et al. **LINPACK users' guide**. [S.l.]: SIAM, 1979.

EVANGELINOS, C.; HILL, C. . . . Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models . . . . **Ratio**, 2008. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.296.3779http://scholar.google.com/scholar?q=Cloud+Computing+for+parallel+Scientific+HPC+Applications:+Feasibility+of+running+Coupled+Atmosphere-Ocean+Climate+Models+on+Amazon's+EC2.{&}hl=en{&}btnG>.

FARBER, R. **Parallel programming with OpenACC**. [S.l.]: Newnes, 2016.

FREITAS, S. R. et al. The Coupled Aerosol and Tracer Transport model to the Brazilian developments on the Regional Atmospheric Modeling System (CATT-BRAMS) – Part 1: Model description and evaluation. **Atmospheric Chemistry and Physics**, v. 9, n. 8, p. 2843–2861, 2009. ISSN 1680-7324.

GABRIEL, E. et al. Open MPI: Goals, concept, and design of a next generation MPI implementation. In: **Recent Advances in Parallel Virtual Machine and Message Passing Interface (PVMMPI)**. [S.l.: s.n.], 2004. p. 97–104.

GARLAND, M. et al. Parallel computing experiences with cuda. **IEEE micro**, IEEE, v. 28, n. 4, p. 13–27, 2008.

GARTNER. **Gartner IaaS Public Cloud Market**. 2019.

GROPP, W. et al. **Using MPI: portable parallel programming with the message-passing interface**. [S.l.]: MIT press, 1999. v. 1.

GUO, T. et al. Cost-aware cloud bursting for enterprise applications. In: **ACM Transactions on Internet Technology**. ACM, 2014. v. 13, n. 3, p. 1–24. ISSN 15576051. Disponível em: <http://dl.acm.org/citation.cfm?doid=2630790.2602571>.

GUPTA, A. et al. The who, what, why, and how of high performance computing in the cloud. In: **Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom**. IEEE, 2013. v. 1, p. 306–314. ISBN 978-0-7695-5095-4. ISSN 23302186. Disponível em: <http://ieeexplore.ieee.org/document/6753812/>.

GUPTA, A. et al. Hpc-aware vm placement in infrastructure clouds. In: IEEE. **2013 IEEE International Conference on Cloud Engineering (IC2E)**. [S.l.], 2013. p. 11–20.

HE, Q. et al. Case study for running HPC applications in public clouds. In: **HPDC 2010 - Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing**. New York, New York, USA: ACM Press, 2010. p. 395–401. ISBN 9781605589428. Disponível em: <http://portal.acm.org/citation.cfm?doid=1851476.1851535>.

HERDMAN, J. et al. Accelerating hydrocodes with openacc, opencl and cuda. In: IEEE. **2012 SC Companion: High Performance Computing, Networking Storage and Analysis**. [S.l.], 2012. p. 465–471.

HILLIS, W. D.; STEELE, G. L. Data parallel algorithms. **Communications of the ACM**, ACM, v. 29, n. 12, p. 1170–1183, dec 1986. ISSN 15577317. Disponível em: <http://portal.acm.org/citation.cfm?doid=7902.7903>.

HOROWITZ, M.; INDERMAUR, T.; GONZALEZ, R. Low-power digital design. In: **Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium**. [S.l.: s.n.], 1994. p. 8–11. ISBN 0-7803-1953-2.

INTRODUCING Intel MPI Benchmarks. 2019. Disponível em: <https://software.intel.com/en-us/articles/intel-mpi-benchmarks/>.

IOSUP, A. et al. Performance analysis of cloud computing services for many-tasks scientific computing. **IEEE Transactions on Parallel and Distributed Systems**, v. 22, n. 6, p. 931–945, June 2011. ISSN 1045-9219.

JACKSON, K. R. et al. Performance analysis of high performance computing applications on the Amazon Web Services cloud. In: **Proceedings - 2nd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2010**. IEEE, 2010. p. 159–168. ISBN 9780769543024. Disponível em: <http://ieeexplore.ieee.org/document/5708447/>.

KALÉ, L. V. The virtualization model of parallel programming : Runtime optimizations and the state of art. In: **LACSI 2002**. Albuquerque: [s.n.], 2002.

KOTAS, C.; NAUGHTON, T.; IMAM, N. A comparison of Amazon Web Services and Microsoft Azure cloud platforms for high performance computing. **2018 IEEE International Conference on Consumer Electronics, ICCE 2018**, v. 2018-Janua, p. 1–4, 2018.

LANGGUTH, J.; CAI, X.; SOUROURI, M. Memory Bandwidth Contention: Communication vs Computation Tradeoffs in Supercomputers with Multicore Architectures. In: **Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS**. IEEE, 2019. v. 2018-Decem, p. 497–506. ISBN 9781538673089. ISSN 15219097. Disponível em: <https://ieeexplore.ieee.org/document/8644601/>.

LANGMEAD, B.; NELLORE, A. Cloud computing for genomic data analysis and collaboration. **Nature Reviews Genetics**, Nature Publishing Group, v. 19, n. 4, p. 208, 2018.

Laros III, J. H. et al. Energy Delay Product. In: **Energy-Efficient High Performance Computing: Measurement and Tuning**. [S.l.: s.n.], 2013. v. 3, p. 51–55. ISBN 978-1-4471-4491-5.

LI, W. et al. Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds. **IEEE Access**, IEEE, v. 6, p. 61488–61502, 2018.

Li, X. et al. Performance analysis and modeling of video transcoding using heterogeneous cloud services. **IEEE Transactions on Parallel and Distributed Systems**, v. 30, n. 4, p. 910–922, April 2019.

MANGUL, S. et al. How bioinformatics and open data can boost basic science in countries and universities with limited resources. **Nature biotechnology**, Nature Publishing Group, v. 37, n. 3, p. 324, 2019.

MARATHE, A. et al. A comparative study of high-performance computing on the cloud. In: **International Symposium on High-performance Parallel and Distributed Computing**. [S.l.: s.n.], 2013. p. 239–250. ISBN 978-1-4503-1910-2.

MARIANI, G. et al. Predicting Cloud Performance for HPC Applications: A User-Oriented Approach. In: **Proceedings - 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017**. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2017. p. 524–533. ISBN 9781509066100.

MAUCH, V.; KUNZE, M.; HILLENBRAND, M. High performance cloud computing. **Future Generation Computer Systems**, Elsevier, v. 29, n. 6, p. 1408–1416, 2013.

MCCALPIN, J. D. **Sustainable Memory Bandwidth in Current High Performance Computers**. [S.l.], 1995. Disponível em: <http://www.cs.virginia.edu/{~}mccalpin/papers/bandwidth/bandwidth.ht>.

MEHROTRA, P. et al. Performance evaluation of Amazon EC2 for NASA HPC applications. In: **Proceedings of the 3rd workshop on Scientific Cloud Computing Date - ScienceCloud '12**. New York, New York, USA: ACM Press, 2012. p. 41. ISBN 9781450313407. Disponível em: <http://dl.acm.org/citation.cfm?doid=2287036.2287045>.

MELL, P.; GRANCE, T. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. **Nist Special Publication**, v. 145, p. 7, 2011. ISSN 00845612. Disponível em: <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>.

MELO, A. C. de. The New Linux ’perf’ Tools. In: **Linux Kongress**. [S.l.: s.n.], 2010.

MOHAMMADI, M.; BAZHIROV, T. Comparative benchmarking of cloud computing vendors with High Performance Linpack. In: **ACM International Conference Proceeding Series**. New York, New York, USA: ACM Press, 2018. p. 1–5. ISBN 9781450363372. Disponível em: <http://dl.acm.org/citation.cfm?doid=3195612.3195613>.

MOSCHAKIS, I. A.; KARATZA, H. D. Evaluation of gang scheduling performance and cost in a cloud computing system. **The Journal of Supercomputing**, v. 59, n. 2, p. 975–992, Feb 2012. ISSN 1573-0484. Disponível em: <https://doi.org/10.1007/s11227-010-0481-4>.

NAIR, S. K. et al. Towards secure cloud bursting, brokerage and aggregation. In: **Proceedings - 8th IEEE European Conference on Web Services, ECOWS 2010**. IEEE, 2010. p. 189–196. ISBN 9780769543109. Disponível em: <http://ieeexplore.ieee.org/document/5693261/>.

NAPPER, J.; BIENTINESI, P. Can cloud computing reach the top500? In: **Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop - UCHPC-MAW ’09**. New York, New York, USA: ACM Press, 2009. p. 17. ISBN 9781605585574. Disponível em: <http://portal.acm.org/citation.cfm?doid=1531666.1531671>.

NUMRICH, R. W. Cray-2 memory organization and interprocessor memory contention. In: MEYER, C. D.; PLEMMONS, R. J. (Ed.). **Linear Algebra, Markov Chains, and Queueing Models**. New York, NY: Springer New York, 1993. p. 267–294. ISBN 978-1-4613-8351-2.

PANDA, D. K.; LU, X. HPC Meets Cloud. In: **Proceedings of the10th International Conference on Utility and Cloud Computing - UCC ’17**. New York, New York, USA: ACM Press, 2017. p. 189–190. ISBN 9781450351492. Disponível em: <http://dl.acm.org/citation.cfm?doid=3147213.3149455>.

PEARCE, O. et al. Quantifying the effectiveness of load balance algorithms. In: **ACM International Conference on Supercomputing (ICS)**. [S.l.: s.n.], 2012. p. 185–194. ISBN 9781450313162.

PERSICO, V. et al. A first look at public-cloud inter-datacenter network performance. **2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings**, 2016.

PERSICO, V. et al. On network throughput variability in microsoft azure cloud. **2015 IEEE Global Communications Conference, GLOBECOM 2015**, n. ii, 2015.

PETITET, A. et al. **HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers**. 2012. Disponível em: <http://www.netlib.org/benchmark/hpl/>.

PIELKE, R. A. et al. A Comprehensive Meteorological Modeling System- RAMS. **Meteorology and Atmospheric Physics**, v. 91, n. 1-4, p. 69–91, 1992.

POSEY, B. et al. Addressing the challenges of executing a massive computational cluster in the cloud. In: **2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)**. Los Alamitos, CA, USA: IEEE Computer Society, 2018. p. 253–262. Disponível em: <https://doi.ieeecomputersociety.org/10.1109/CCGRID.2018.00040>.

PRABHAKARAN, A.; LAKSHMI, J. Cost-benefit Analysis of Public Clouds for offloading in-house HPC Jobs. **2018 IEEE 11th International Conference on Cloud Computing (CLOUD)**, IEEE, p. 57–64, 2018. ISSN 21596190.

PRUKKANTRAGORN, P.; TIENTANOPAJAI, K. Price efficiency in high performance computing on amazon elastic compute cloud provider in compute optimize packages. In: **20th International Computer Science and Engineering Conference: Smart Ubiquitos Computing and Knowledge, ICSEC 2016**. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2017. ISBN 9781509044207.

RIDGE, D. et al. Beowulf: Harnessing the power of parallelism in a Pile-of-PCs. In: **IEEE Aerospace Applications Conference Proceedings**. IEEE, 1997. v. 2, p. 79–91. ISBN 0-7803-3741-7. Disponível em: <http://ieeexplore.ieee.org/document/577619/>.

ROLOFF, E. et al. High Performance Computing in the Cloud: Deployment, Performance and Cost Efficiency. In: **IEEE International Conference on Cloud Computing Technology and Science (CloudCom)**. [S.l.: s.n.], 2012. p. 371–378. ISBN 9781467345101.

ROLOFF, E. et al. High performance computing in the cloud: Deployment, performance and cost efficiency. In: IEEE. **Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on**. [S.l.], 2012. p. 371–378.

ROLOFF, E. et al. Exploiting price and performance tradeoffs in heterogeneous clouds. In: **Companion Proceedings of the 10th International Conference on Utility and Cloud Computing, UCC 2017, Austin, TX, USA, December 5-8, 2017**. [s.n.], 2017. p. 71–76. Disponível em: <http://doi.acm.org/10.1145/3147234.3148103>.

ROLOFF, E. et al. Leveraging cloud heterogeneity for cost-efficient execution of parallel applications. In: RIVERA, F. F.; PENA, T. F.; CABALEIRO, J. C. (Ed.). **Euro-Par 2017: Parallel Processing: 23rd International Conference on Parallel and Distributed Computing, Santiago de Compostela, Spain, August 28 – September 1, 2017, Proceedings**. Springer International Publishing, 2017. p. 399–411. ISBN 978-3-319-64203-1. Disponível em: <https://doi.org/10.1007/978-3-319-64203-1_29>.

Roloff, E. et al. Hpc application performance and cost efficiency in the cloud. In: **2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)**. [S.l.: s.n.], 2017. p. 473–477.

SAAD, A.; EL-MAHDY, A. Network topology identification for cloud instances. **Proceedings - 2013 IEEE 3rd International Conference on Cloud and Green Computing, CGC 2013 and 2013 IEEE 3rd International Conference on Social Computing and Its Applications, SCA 2013**, p. 92–98, 2013.

SADOOGHI, I. et al. Understanding the Performance and Potential of Cloud Computing for Scientific Applications. **IEEE Transactions on Cloud Computing**, PP, n. 99, p. 358–371, 2015.

SAINI, S.; BAILEY, D. H. NAS Parallel Benchmark (Version 1 . 0) Results 11-96. n. November, p. 1–53, 1996.

SAMREEN, F. et al. Daleel: Simplifying cloud instance selection using machine learning. In: **Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium**. IEEE, 2016. p. 557–563. ISBN 9781509002238. Disponível em: <http://ieeexplore.ieee.org/document/7502858/>.

SANDERS, J.; KANDROT, E. **CUDA by example: an introduction to general-purpose GPU programming**. [S.l.]: Addison-Wesley Professional, 2010.

SCHEUNER, J.; LEITNER, P. Performance Benchmarking of Infrastructure-as-a-Service (IaaS) Clouds with Cloud WorkBench. In: **ICPE 2019 - Companion of the 2019 ACM/SPEC International Conference on Performance Engineering**. New York, New York, USA: ACM Press, 2019. p. 53–56. ISBN 9781450362863. Disponível em: <http://dl.acm.org/citation.cfm?doid=3302541.3310294>.

SILVA, D. P. D.; CIRNE, W.; BRASILEIRO, F. V. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In: SPRINGER. **European Conference on Parallel Processing**. [S.l.], 2003. p. 169–180.

Silva, M. et al. Cloudbench: Experiment automation for cloud environments. In: **2013 IEEE International Conference on Cloud Engineering (IC2E)**. [S.l.: s.n.], 2013. p. 302–311.

SNIR, M. et al. **MPI–the Complete Reference: The MPI core**. [S.l.]: MIT press, 1998. v. 1.

STANDARDS, M.; SOCIETY, C. **IEEE Standard for Floating-Point Arithmetic - IEEE Xplore Document**. [s.n.], 2019. ISBN 9781504459242. Disponível em: <http://ieeexplore.ieee.org/document/4610935/>.

STERLING, T.; STARK, D. A High-Performance Computing Forecast: Partly Cloudy. **Computing in Science & Engineering**, IEEE Computer Society, v. 11, n. 4, p. 42–49, jul 2009. ISSN 1521-9615. Disponível em: <http://ieeexplore.ieee.org/document/5076318/>.

TANENBAUM, A. S.; STEEN, M. V. **Distributed systems: principles and paradigms**. [S.l.]: Prentice-Hall, 2007.

TOP500. **Amazon EC2 Cluster Compute Instances**. 2010. Disponível em: <https://www.top500.org/system/177002>.

VÁZQUEZ, M. et al. Alya multiphysics simulations on intel's xeon phi accelerators. In: SPRINGER. **Latin American High Performance Computing Conference**. [S.l.], 2014. p. 248–254.

VECCHIOLA, C.; PANDEY, S.; BUYYA, R. High-Performance Cloud Computing: A View of Scientific Applications. In: **2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks**. IEEE, 2009. p. 4–16. ISBN 978-1-4244-5403-7. Disponível em: <http://ieeexplore.ieee.org/document/5381983/>.

VáZQUEZ, M. et al. Alya: Multiphysics engineering simulation toward exascale. **Journal of Computational Science**, v. 14, p. 15 – 27, 2016. ISSN 1877-7503. The Route to Exascale: Novel Mathematical Methods, Scalable Algorithms and Computational Science Skills. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877750315300521>.

WALKER, E. benchmarking Amazon EC2 for high-performance scientific computing. **Program**, p. 18–23, 2008. Disponível em: <https://www.usenix.org/publications/login/october-2008-volume-33-number-5/benchmarking-amazon-ec2-high-performance>.

YELICK, K. et al. **The Magellan Report on Cloud Computing for Science**. Washington, DC, 2011.

YELICK, K. et al. The magellan report on cloud computing for science. **US Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR)**, v. 3, 2011.

ZAHID, F. et al. A self-adaptive network for HPC clouds: Architecture, framework, and implementation. **IEEE Transactions on Parallel and Distributed Systems**, v. 29, n. 12, p. 2658–2671, dec 2018. ISSN 15582183. Disponível em: <https://ieeexplore.ieee.org/document/8385210/>.

ZHAI, Y. et al. Cloud versus in-house cluster: Evaluating Amazon cluster compute instances for running MPI applications. **2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)**, p. 1–10, 2011.

ZHANG, M. et al. A declarative recommender system for cloud infrastructure services selection. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.]: Springer, Berlin, Heidelberg, 2012. v. 7714 LNCS, p. 102–113. ISBN 9783642351938. ISSN 03029743.

ZHENG, W. et al. Cost optimization for deadline-aware scheduling of big-data processing jobs on clouds. **Future Generation Computer Systems**, Elsevier, v. 82, p. 244–255, 2018.

APPENDIX A — RESUMO EM PORTUGUÊS

In this chapter, we present a summary of this Ph.D. Thesis in the Portuguese language, as required by the PPGC Graduate Program in Computing.

Neste capítulo, é apresentado um resumo desta tese de doutorado na língua Portuguesa, como requerido pelo Programa de Pós-Graduação em Computação.

A.1 Introdução

Durante a década de 1990, *clusters* de computadores tornaram-se uma alternativa popular (BUYYA, 1999; BAKER; FOX; YAU, 1995) dentro da comunidade de Computação de Alto Desempenho (HPC) como um ambiente alternativo para supercomputadores. Principalmente porque o poder de processamento do *hardware* da máquina padrão tinha evoluído significativamente, e porque as configurações padrão se tornaram significativamente mais baratas quando comparadas aos supercomputadores.

O modelo de *Cloud Computing* (ARMBRUST et al., 2009; MELL; GRANCE, 2011) começou a ganhar notoriedade no início de 2010 como um ambiente alternativo para executar aplicações de uso geral. Um dos benefícios notáveis da *Cloud Computing* é o acesso aos recursos computacionais sem a necessidade de adquiri-los, onde o usuário é cobrado apenas pelo uso de recursos. Desde que os primeiros provedores de *Cloud Computing* pública começaram a oferecer seus serviços, eles rapidamente atraíram a atenção da comunidade de HPC (WALKER, 2008; VECCHIOLA; PANDEY; BUYYA, 2009; STERLING; STARK, 2009; ZHAI et al., 2011; MEHROTRA et al., 2012). Os pesquisadores começaram a avaliar esses serviços como um ambiente onde os *clusters* de computadores poderiam ser configurados para executar aplicações científicas. Hoje em dia, *Cloud Computing* já é um ambiente consolidado para a execução de aplicações de HPC (COUTINHO et al., 2016; LANGMEAD; NELLORE, 2018; MANGUL et al., 2019), com muitos estudos de desempenho e custo-eficiência demonstrando que *Cloud Computing* é igual e, em alguns casos, até supera os *clusters* tradicionais (SAAD; EL-MAHDY, 2013), devido à maior flexibilidade e menores custos iniciais para o *hardware*.

No entanto, a pesquisa sobre HPC em *Cloud Computing* se concentra principalmente em portar aplicações para a *Cloud*, avaliar seu desempenho (MOHAMMADI; BAZHIROV, 2018; SCHEUNER; LEITNER, 2019), custo-eficiência (LI et al., 2018; ZHENG et al., 2018), e melhorar o desempenho da comunicação (CARRENO et al., 2019;

ZAHID et al., 2018); Devido a isso, ainda existem oportunidades de pesquisa aberta para explorar as ofertas de recursos em *Cloud Computing* para HPC.

A maioria dos *clusters* construídos usando provedores de *Cloud* atualmente usa os mesmos tipos de instância, a mesma configuração de Máquina Virtual (VM), ou usa diferentes tipos de instância apenas no contexto de aceleradores (como unidades de processamento gráfico (GPUs)) (CRAGO; WALTERS, 2015). Há aspectos do provisionamento da *Cloud* que têm recebido menos atenção. Um deles é a construção de um *cluster* usando diferentes tipos de instância de *Cloud*. Nós nos referimos a esse sistema como uma *Cloud* heterogênea.

Uma *Cloud* heterogênea é uma solução atraente para a execução de grandes aplicações paralelas, uma vez que essas aplicações normalmente têm demandas computacionais diferentes, com algumas tarefas executando mais trabalho do que outras, gerando assim um desbalanceamento de carga. Nesse cenário, as tarefas de aplicações que executam mais trabalho podem ser executadas em instâncias mais rápidas, mas mais caras, enquanto as tarefas que executam menos trabalho podem ser executadas em instâncias mais lentas e econômicas, aproveitando a heterogeneidade das instâncias de *Cloud*. Esta tese se concentrou em explorar a heterogeneidade da *Cloud* para obter um ambiente de execução mais barato para os usuários.

A.1.1 Definição e Escopo do Problema

As aplicações de HPC são compostas por várias tarefas que se combinam para resolver um problema. Todas as tarefas de aplicação devem ter a mesma demanda computacional, de modo que se obtém uma aplicação com um balanceamento de carga ótimo. O balanceamento de carga é crucial para a utilização ótima dos recursos. No entanto, aplicações frequentemente apresentam um nível de desbalanceamento de carga que causa degradação de desempenho e ociosidade de recursos (BONETI et al., 2008; PEARCE et al., 2012). Por outro lado, a *Cloud* apresenta um grande conjunto de tipos de instância, com uma variação significativa em termos de preço e desempenho.

Devido a isso, é problemático para o usuário entender todas as ofertas de *Cloud* inteira e escolher a instância ou instâncias mais adequadas para executar sua aplicação (SAMREEN et al., 2016; ZHANG et al., 2012; GUO et al., 2014; NAIR et al., 2010). Nós abordamos esse problema criando um mecanismo de recomendação automatizado que usa o perfil de instâncias de *Cloud* e de aplicações para recomendar um ambiente de execução de *Cloud*.

Nossa proposta visa criar ambientes mais adequados, utilizando instâncias de *Cloud* que melhor atendam às demandas das tarefas das aplicações. Além disso, nossa proposta é compatível com uma ampla gama de aplicações e não requer alterações nas aplicações ou ambientes de execução. O público-alvo deste trabalho são cientistas pesquisadores investigando novos ambientes para suas aplicações. Também mostramos o impacto de mudar o foco entre menor tempo de execução e menor custo, o que ajuda os locatários da *Cloud* a escolher o *trade off* desejado para diferentes cenários de execução.

A.1.2 Contribuições

Nesta tese, investigamos nuvens heterogêneas, focando em seu potencial para a execução econômica de aplicações de HPC. Nossas principais contribuições são as seguintes.

- Nós realizamos uma avaliação profunda de nuvens heterogêneas com uma variedade de combinações de instâncias e comportamentos de aplicação paralela.
- Desenvolvemos o *ImbBench*, um *benchmark* que simula vários padrões de desbalanceamento de carga.
- Nós introduzimos o *Cost-Delay Product (CDP)* como uma métrica para analisar os *tradeoffs* entre custo e desempenho, inspirada no *Energy-Delay Product* (EDP) (Laros III et al., 2013).
- Nós apresentamos o *CloudHet*, um mecanismo para recomendar a escolha da instância com base no comportamento da aplicação e nas características da instância focando na otimização do custo-benefício da aplicação.

A.2 Estado da Arte

Desde sua primeira aparição, o modelo *Cloud Computing* vem atraindo a atenção da comunidade de Computação de Alto Desempenho (HPC) para a execução de grandes aplicações paralelas, apresentando o potencial de estender ou até mesmo substituir *clusters* tradicionais utilizados em laboratórios de pesquisa (WALKER, 2008; EVANGELINOS; HILL, 2008; NAPPER; BIENTINESI, 2009; VECCHIOLA; PANDEY; BUYYA, 2009; ARMBRUST et al., 2009; HE et al., 2010; JACKSON et al., 2010). Esta seção discute alguns dos esforços mais consistentes relacionados à HPC em *Cloud Computing*.

Vários estudos têm sido realizados nos últimos anos para verificar a viabilidade da *Cloud Computing* como uma plataforma adequada para HPC (GUPTA et al., 2013a; SADOOGHI et al., 2015; PANDA; LU, 2017). Aplicações de HPC são geralmente compostas por alto processamento, o qual precisa ser tratado pelos programas ou usuários que executam esses conjuntos de trabalhos, que são instâncias de aplicações com diferentes entradas. Portanto, a execução de aplicações de HPC na Cloud requer não apenas atenção à alocação de recursos e otimizações na infraestrutura, mas também outros aspectos, como a forma como os usuários interagem com esse ambiente, qual é o custo-benefício em comparação com os *clusters* privados. Os estudos primários de viabilidade de HPC na Cloud analisam principalmente *benchmarks* de HPC, com foco em CPU, memória, armazenamento e desempenho de disco. Outro efeito da evolução das tecnologias de *Cloud* sobre o tipo é a disponibilidade de recursos de *Cloud* otimizados para HPCs (AMAZON..., 2019). Nesta categoria, o relatório de Magellan (YELICK et al., 2011b) é um dos primeiros e mais abrangentes estudos de HPC em *Cloud Computing*, com uma ampla gama de arquiteturas, cargas de trabalho e métricas.

Na área de otimização de desempenho, os estudos propõem principalmente formas de otimizar o desempenho de HPC na Nuvem, com melhorias direcionadas tanto no nível de infraestrutura quanto no nível de gestão de recursos. Estudos sobre o nível de infraestrutura são principalmente focados em estudar os efeitos sobre a rede, que é responsável pela maioria das ineficiências da execução de cargas de trabalho HPC na Cloud (MAUCH; KUNZE; HILLENBRAND, 2013). Pesquisa de gestão de recursos centra-se em políticas de programação consciente de ambos os comportamentos de aplicação e plataforma, com a maioria dos estudos sendo focada em explorar *Cloud* híbrida ou múltiplas instâncias de *Cloud* (GUPTA et al., 2013b). Nesta área, há trabalhos importantes como Marathe (MARATHE et al., 2013; AWAD; ARTOLI; AHMED, 2014; IOSUP et al., 2011), que comparou um *cluster* de *Cloud* virtualizado com um *cluster* físico. No entanto, os primeiros trabalhos de pesquisa na área careceram de uma avaliação ampla das nuvens públicas, avaliando um único provedor de *Cloud* devido à disponibilidade e aos custos.

A pesquisa de usabilidade é outra importante área de pesquisa, composta por estudos que abstraem a infraestrutura dos usuários de HPC. O principal objetivo é criar uma plataforma de "HPC como serviço" onde as aplicações de HPC são executadas na *Cloud*, abstraindo a infraestrutura subjacente da cloud. Os usuários submetem a aplicação, os parâmetros e as expectativas de QoS, como prazo, via portal web, e um *middleware* cuida do provisionamento e execução. Esta categoria de estudos é relevante para aumentar a adoção da HPC, e destaca os esforços em mover as aplicações legadas para implantações de Software como Serviço (BELGACEM; CHOPARD, 2015) (CHURCH et al., 2012). Para mencionar um caso sobre a escalabilidade da *Cloud*, o trabalho de Posey et al. (POSEY et al., 2018) discute os desafios e as soluções de gerenciamento de um cluster maciço na Amazon *Cloud*. Os autores atingiram um pico de mais de um milhão de núcleos executados simultaneamente, com cerca de 500.000 trabalhos executados em duas horas.

Outro campo de estudo relevante está relacionado ao custo, onde foram identificados dois grupos principais, aqueles com foco em eficiência e outros com foco em redução. A pesquisa de otimização de custos foca principalmente na redução de custos e no aumento da eficiência do HPC na *Cloud*. Os estudos primários nesta área de pesquisa são baseados em comparações entre aplicações em execução na *Cloud* (KOTAS; NAUGHTON; IMAM, 2018; PRUKKANTRAGORN; TIENTANOPAJAI, 2017). Esses estudos exploram principalmente *benchmarks* de HPC e aplicações científicas de HPC em execução nos principais provedores de *Cloud* (Microsoft, Amazon e Google) (PRABHAKARAN; LAKSHMI, 2018). A maioria deles se concentra na comparação de máquinas homogêneas, não explorando a utilização de vários tipos de instâncias existentes nos diferentes provedores de *Cloud* ao executar suas avaliações.

No início desta investigação, realizámos um estudo sobre os fornecedores de *Cloud Computing* face aos *clusters* tradicionais (Roloff et al., 2017) no que diz respeito à sua performance e relação custo-eficácia. Neste estudo, foi realizada uma extensa avaliação de dois grandes provedores de cloud, Amazon EC2 e Microsoft Azure, avaliando seu desempenho de rede, CPU e memória. Foram comparados dois *clusters* acadêmicos com duas configurações de *cluster* no Amazon EC2 e seis configurações de *cluster* no Microsoft Azure. Construímos *clusters* com 256 núcleos cada e aplicamos a suíte NAS (SAINI; BAILEY, 1996) para medir o desempenho computacional; o benchmark STREAM (MCCALPIN, 1995) para avaliação de memória; e o Intel MPI *Benchmarks* (IMB) para avaliação de rede.

Nossos resultados mostraram que a degradação do desempenho devido à virtualização e outras despesas gerais da *Cloud* é insignificante. O desempenho da rede permaneceu um gargalo significativo para o desempenho da aplicação; no entanto, houve melhorias significativas.

Além disso, descobrimos que pagar por uma *Cloud* mais robusta nem sempre melhora o desempenho e pode até mesmo levar a reduções de desempenho. Como conclusão geral, pudemos afirmar que a *Cloud Computing* é um ambiente viável para HPC. No entanto, foi necessário conhecer o comportamento da aplicação para aproveitar as características da Cloud.

A.3 Explorando a Heterogeneidade da Nuvem para HPC

Os provedores de *Cloud* pública oferecem uma ampla gama de tipos de instância com diferentes velocidades, configurações e preços, o que permite que os usuários escolham as configurações mais adequadas para suas aplicações. Ao executar aplicações paralelas que exigem a execução de várias instâncias, como grandes aplicações científicas, a maioria dos usuários escolhe um tipo de instância que melhor atende às suas necessidades gerais e, em seguida, cria um *cluster* de instâncias interconectadas do mesmo tipo. No entanto, as tarefas de uma aplicação paralela muitas vezes têm demandas diferentes em termos de uso de CPU e uso de memória. Essa diferença de demandas pode ser explorada selecionando vários tipos de instância que são adaptadas às demandas da aplicação. Dessa forma, a combinação da heterogeneidade da *Cloud* pública com a heterogeneidade da aplicação pode ser explorada para reduzir o custo de execução sem perda significativa de desempenho.

A.3.1 Escopo da Presente Tese

Em termos de atores que interagem com a *Cloud Computing*, temos dois papéis claramente definidos, o de provedor e o de usuário. O provedor é a empresa que implementa e fornece serviços de *Cloud Computing* e é pago por isso. O provedor também é responsável por todas as definições técnicas de que tipo de *hardware* será usado, que virtualizador, e o modelo de faturamento. O usuário é o consumidor do serviço fornecido pelo provedor, que pode ser um usuário final, uma única pessoa ou uma organização.

O escopo de nossa tese é relativo ao ponto de vista do usuário, onde estamos estudando a viabilidade da utilização da *Cloud* focada nela, sendo agnóstica para o provedor de serviços. O que buscamos trazer ao cliente é a possibilidade de escolher um ambiente de execução para aplicações de HPC de forma automatizada e otimizada em termos de custo. Dessa forma, o usuário terá uma sugestão do ambiente a ser utilizado que apresente a melhor relação entre custo e desempenho. Para isso, realizamos um estudo da heterogeneidade das instâncias de *Cloud* em termos de processamento, bem como da demanda de processamento dos diversos processos da aplicação. Utilizando ambos os estudos, desenvolvemos um mecanismo de decisão que combina as características das instâncias disponíveis e as demandas da aplicação, fazendo com que o usuário execute a aplicação minimizando o tempo ocioso do ambiente e reduzindo o custo. A própria entrega desta tese é o mecanismo de escolha para as instâncias heterogêneas a serem utilizadas na execução de aplicações HPC. O mecanismo de decisão que desenvolvemos foi chamado de *CloudHet*, referindo-se a Cloud e Heterogeneidade.

O CloudHet tem algumas condições, pois foi desenvolvido em linha com as ofertas de instâncias de *Cloud* pública. A primeira condição é que o CloudHet considera apenas instâncias do mesmo tamanho em termos de número de *cores*; todas as instâncias escolhidas devem ter a mesma quantidade de *cores*. Ao mesmo tempo, a quantidade de processos MPI que a aplicação usa para executar deve ser múltipla da quantidade de *cores* das instâncias escolhidas. Por exemplo, se as instâncias a serem consideradas tiverem oito *cores*, a aplicação deve executar utilizando um múltiplo de 8 processos, como 8, 16, 32, 40, 360. Se considerarmos 32 instâncias de *cores*, a aplicação deve executar um múltiplo de quantidade de processos de 32; e assim por diante. A outra condição é que o CloudHet trabalha somente com aplicações com um tamanho fixo de processos MPI, não consideramos a criação de processos dinâmicos neste momento.

A.4 Métrica e Benchmark Propostos para Cargas de Trabalho Desbalanceadas

A execução de grandes aplicações paralelas em ambientes de *Cloud* é agora um importante foco de pesquisa em *Cloud Computing*. Dentre os três modelos de serviços oferecidos pelo modelo *Cloud Computing*, o mais adequado para High-Performance Computing (HPC) é o modelo IaaS, pois fornece instâncias que podem ser customizadas para as necessidades dos usuários e combinadas para construir um sistema de *cluster*. Atualmente, muitos provedores de *Cloud* oferecem um grande número de tipos de instância que visam aplicações do domínio de HPC. Em contraste com os sistemas de *cluster* tradicionais, que geralmente consistem em nós de *cluster* homogêneos, os provedores de *Cloud* oferecem a possibilidade de alocar diferentes tipos de instâncias de forma flexível e interligá-los, criando assim grandes plataformas heterogêneas para aplicações distribuídas. Normalmente, a heterogeneidade da *Cloud* é explorada perto do nível de *hardware* (DONG et al., 2017) e está relacionada principalmente ao lado do provedor de serviços. No entanto, poucos estudos visam explorar a heterogeneidade em nuvens públicas.

Uma maneira interessante de usar um sistema tão heterogêneo é combinando as demandas da aplicação com o *hardware* subjacente. No contexto desta tese, nós focamos em combinar a carga dos diferentes processos de uma aplicação paralela para diferentes tipos de instância, fornecendo assim uma maneira de mitigar o desbalanceamento de carga em tais aplicações.

A.4.1 Cost-Delay Product: uma métrica para medir a eficiência de custos

Um aspecto importante ao avaliar os tipos de instância de *Cloud* é a comparação de seu *trade off* de preço/desempenho. Várias métricas básicas foram propostas para avaliar esse *trade off*. Uma métrica que compara diferentes configurações da *Cloud* é a métrica custo-eficiência (ROLOFF et al., 2012a; MOSCHAKIS; KARATZA, 2012). No entanto, essas métricas não permitem avaliações mais abrangentes, por exemplo, colocando mais ênfase no preço ou no desempenho de acordo com a preferência do usuário.

Introduzimos uma nova métrica, a *Cost-Delay Product (CDP)*, que é modelada após o produto de atraso de energia (EDP) (HOROWITZ; INDERMAUR; GONZALEZ, 1994; Laros III et al., 2013). O objetivo da EDP não é minimizar o consumo de energia, mas sim introduzir uma métrica para encontrar um trade-off adequado de poupança de energia e degradação do desempenho. A métrica básica do CDP é definida da seguinte forma:

$$CDP = cost\ of\ execution \times execution\ time \tag{A.1}$$

O $cost\ of\ execution$ representa o preço do ambiente usado para executar a aplicação. A maioria dos provedores de *Cloud* pública baseia seu modelo de preço em horas de uso, e o custo usado na Equação A.1 é o preço por hora (em US$) das instâncias alocadas em um provedor de *Cloud*. O $execution\ time$ é o tempo de execução da aplicação no ambiente alocado.

Valores mais baixos para o CDP indicam uma melhor relação custo-benefício para uma determinada aplicação em um determinado ambiente. Essa métrica pode ser usada para comparar diretamente duas alocações de *Cloud* diferentes, inclusive em provedores diferentes. Esta versão básica do CDP se assemelha às propostas anteriores para quantificar o custo-eficiência(ROLOFF et al., 2012a).

A.4.2 O *benchmark* ImbBench

Para melhor avaliar sistemas heterogêneos, desenvolvemos o *Imbalance Benchmark (ImbBench)*, que é um conjunto de aplicações baseadas em MPI que simulam vários comportamentos em termos de cargas de processamento. O ImbBench foi projetado com a heterogeneidade da *Cloud* em mente. Seu objetivo é ajudar o usuário a escolher a configuração mais adequada para executar uma aplicação em *Cloud Computing*. Ele distribui a carga entre todos os processos disponíveis de acordo com um padrão de desbalanceamento pré-selecionado. O código foi desenvolvido na linguagem de programação C e está disponível publicamente em <https://github.com/Roloff/ImbBench>. ImbBench pode criar todos os padrões implementados para qualquer uma das avaliações disponíveis, CPU e memória. Ele atualmente pode escalar até 1024 processos.

Existem quatro diferentes padrões de desbalanceamento de carga implementados pelo ImbBench: Balanced, Multi-Level (dois até oito), Amdahl, e Linear. Estes padrões representam tipos comuns de desbalanceamento em aplicações paralelas. O padrão *Balanced*, simula uma aplicação completamente equilibrada. Este padrão simula o comportamento mais desejável para uma aplicação HPC onde todas as tarefas MPI executam a mesma carga e terminam sua execução ao mesmo tempo; nesta situação, não há recursos ociosos durante a execução e, consequentemente, nenhum dinheiro gasto desnecessariamente.

O padrão *Amdahl* simula uma aplicação que possui um ou mais processos que executam muito mais trabalho do que todos os outros processos. Normalmente, esse comportamento se apresenta quando uma aplicação precisa de um processo central para distribuir todas as tarefas e consolidar os resultados. Este padrão apresenta um alto nível de desbalanceamento, e a maioria dos processos, exceto o central, apresenta grandes quantidades de tempo de execução ocioso.

O padrão *Multi-level* mostra níveis de carga distintos, entre dois e oito níveis, simulando uma aplicação com várias cargas diferentes entre os processos. Esse padrão apresenta um tempo ocioso misto, de acordo com a carga do processo. O padrão *Two-level* é um caso particular do padrão *Multi-level*. Nós o incluímos porque é um padrão de desbalanceamento razoavelmente comum em aplicações paralelas. Neste caso, metade dos processos não apresentam tempo ocioso, e a outra metade apresenta tempo ocioso.

Finalmente, o padrão *Linear* simula uma aplicação onde todos os processos têm uma carga diferente, começando com uma carga baixa no processo 0 e aumentando linearmente até chegar no processo $n - 1$.

A.5 CloudHet: Um Mecanismo para Alocação de Nuvens Heterogêneas

Os fornecedores públicos de serviços de *Cloud Computing* oferecem uma vasta gama de tipos de instâncias, com diferentes velocidades de processamento e interconexão, bem como diferentes preços. Além disso, as tarefas de muitas aplicações paralelas apresentam diferentes demandas computacionais. Estas diferenças podem ser exploradas para melhorar a relação custo-benefício das aplicações paralelas em muitos ambientes de *Cloud Computing*, fazendo corresponder os requisitos das aplicações a diferentes tipos de instâncias. Uma *Cloud* heterogênea é uma solução interessante para a execução de grandes aplicações paralelas, uma vez que essas aplicações normalmente têm demandas computacionais heterogêneas, com algumas tarefas executando mais trabalho do que outras. Nesse cenário, as tarefas que executam mais trabalho podem ser executadas em instâncias mais rápidas, mas mais caras, enquanto as tarefas que executam menos trabalho podem ser executadas em instâncias mais lentas e mais econômicas.

Nesta seção, introduzimos nosso mecanismo chamado *CloudHet*, que usa o comportamento da aplicação combinado com *trade-offs* heterogêneos de preço/desempenho em *Cloud Computing* para selecionar uma alocação de instância de *Cloud* econômica. Nossa avaliação com aplicações baseados em MPIs em uma *Cloud* Azure mostra que essa eficiência pode ser significativamente melhorada, em até 42,3%, dependendo do desbalanceamento de carga da aplicação e da métrica empregada, mantendo um desempenho semelhante. Os ganhos obtidos pela alocação do nosso mecanismo foram próximos da alocação ótima na maioria dos casos. Portanto, mostramos que um usuário pode obter melhorias a partir de uma execução heterogênea sem exigir avaliações demoradas de todas as combinações de instâncias possíveis.

A.6 Conclusão

A utilização da *Cloud Computing* para computação de alto desempenho produziu resultados notáveis, tanto em termos de utilização como um ambiente utilizado para alargar os ambientes tradicionais como para substituir os *clusters* tradicionais. A Nuvem Pública tem várias características que podem ser benéficas para o utilizador, entre as quais se destacam duas: o modelo de faturação *pay-per-use* e a elasticidade. Outra característica inerente ao modelo de *Cloud Computing* é a heterogeneidade, onde estão disponíveis várias instâncias com diferentes configurações e preços. Para se beneficiar plenamente dessas características, é essencial analisar a demanda de processamento da aplicação e usar essas informações para criar ambientes de *Cloud* que correspondam ao comportamento da aplicação, melhorando a relação custo-benefício da alocação. Esta tese avança o campo de HPC em *Cloud Computing* de duas maneiras. Em primeiro lugar, introduzimos e validamos o uso de configurações heterogêneas para executar aplicações HPC, melhorando a relação custo-benefício. Em segundo lugar, introduzimos um mecanismo que automatiza a escolha das instâncias a serem utilizadas de acordo com a demanda da aplicação.