FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Interference Aware Scheduling for Cloud Computing

Diogo Trindade Basto

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Jorge Manuel Gomes Barbosa

February 1, 2015

Interference Aware Scheduling for Cloud Computing

Diogo Trindade Basto

Mestrado Integrado em Engenharia Informática e Computação

February 1, 2015

Abstract

Computing services is a growing industry in the last decade with a increasingly broader audience. Scientific organizations use these giant computational infrastructures to provide their collaborators computational power that their personal computers do not have, reducing task execution times like satellite photos compilation from several days to a few hours. In order to house more services and have shorter response times it is necessary to study system flaws and primary delayers and fix them. This report targets task scheduling in HPC networks and study virtualization technologies and the problem of interference between virtual machines. Through task allocation in hosts that minimize interference effects, execution time reduction allow the same resources to execute a bigger number of tasks in the same period of time. For demonstration purposes, the SimGrid platform is used to compare results of the proposed algorithm against standard industry algorithms.

Resumo

A indústria de serviços computacionais está em crescente desde a última década com uma atração de público cada vez mais geral. As organizações científicas utilizam estas infraestruturas computacionais gigantes para disponibilizarem aos seus colaboradores poder computacional que as suas máquinas não possuem, reduzindo o tempo de execução de tarefas como compilação de fotos de satélites de vários dias para poucas horas. Para poder albergar cada vez mais serviços e ter tempos de resposta mais rápidos é necessário estudar as falhas do sistema e principais causadores de atrasos e corrigi-los. Este relatório aborda a área de escalonamento de tarefas em redes computacionais de alto rendimento e estuda tecnologias de virtualização e o problema de interferências entre máquinas virtuais. Através de alocações de tarefas em nós que minimizem o efeito de interferência, a redução do tempo de execução de tarefas permite aos mesmos recursos executar um maior número de tarefas no mesmo período de tempo. Para efeitos de demonstração de resultados é utilizada a plataforma SimGrid e comparado os resultados do algoritmo proposto contra algoritmos já aplicados na indústria.

"Long you live and high you fly
smiles you'll give and tears you'll cry
all you touch and all you see
Is all your life will ever be. "

Roger Waters

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations

API	Application Programming Interface
CERN	Conseil Européen pour la Recherche Nucléaire
CPOP	Critical-Path-on-a-Processor
HEFT	Heterogeneous Earliest-Finish-Time
HPC	High Performance Computing
fMRI	functional Magnetic Resonance Imaging
IaaS	Infrastructure as a Service
KVM	Kernel-based Virtual Machine
Mcps	Million computations per second
MSG	MetaSimGrid
NSA	National Security Agency
OS	Operating System
PaaS	Platform as a Service
QoS	Quality of Service
RAM	Random Access Memory
SaaS	Software as a Service
SG	SimGrid
SLA	Service Level Agreement
VM	Virtual Machine
WWW	World Wide Web

Chapter 1

Introduction

The following dissertation will present a method for increasing the capacity of Cloud services through a scheduling heuristic that takes into account the interference caused by the virtualization technology. The heuristic presented is later tested against a industry standard heuristic.

The purpose of this first chapter is to give a description of the dissertation's purpose and describing its chapters.

1.1 Context

Software, Platform and Infrastructure as a Service (*SaaS,PaaS,IaaS*) are terms that are getting more and more common by the minute. Large computational grids, commonly referred as *Clouds*, are a sought after commodity by the technological industry, as they allow computational needs at lower prices [AFG+10]. Despite its recent trendiness, the technology isn't exactly new though. It's simply a combination of older technology, virtualization and internet [VRMCL08, Men05], being combined and applied in a new way, as a response to the World Wide Web (*WWW*) growth. This service is offered to common enterprises by the big internet companies like Google, Amazon and Microsoft, allowing them to expand their computational power to whatever their demand is, eliminating an upfront investment and increasing their scalability power. Bigger enterprises, like the Conseil Européen pour la Recherche Nucléaire (*CERN*) [Fou13] and the National Security Agency (*NSA*), built their own Clouds so that they can allocate computing power to whichever task is requiring more demand daily, allowing their collaborators to not rely solely on their own personal computers to run their own tasks, but pooling resources together to achieve faster performance of the collective.

1.2 Motivation

Usage optimization of available resources allows for improvement in the Quality of Service (*QoS*) supplied to the users, through faster and more responsive services. It also allows for cost reduction by the suppliers, due to less energy consumption by the more efficient resources and a better achievement rate of the Service Level Agreement (*SLA*), the contract that establishes the minimum performance requirements and the economical penalties for failing them, resulting in a gain for both parties of the Cloud market. Win-win situations are a great motivator for research, making the study of resource optimization highly desirable. One of the main parts of the Cloud is the scheduler, responsible for assigning tasks to the available hosts. This scheduling is made on-line [HKKS03], meaning the assignment is made when a task arrives to the Cloud and should remain unaltered, save for host failure. The Cloud scheduler needs to evaluate the available hosts and decide which one is more fit for the task and will get it done more efficiently. This evaluation takes the specification of the tasks into account, either by user provided information or by testing it on a sandbox and assess its needs, and hosts' workload and resource usage. Current standard scheduling techniques do not take into account the performance interference of Virtual Machines (*VM*) co-allocation [KKB+07, PLM+10]. This interference is a major bottleneck in current Cloud infrastructures, caused by the inability of virtualization technology and hardware architecture to fully separate some of the resources used, making multiple VMs race for the same resources on a given host. Rackspace, a major Cloud services supplier and founder of the OpenStack project, announced a new product designed to fight this same problem. Their solution is to remove co-allocation altogether and offer a contract that specifies that a given host is fully owned by a single user [Kon14]. The announcement voices their experience with virtualization and the negative effects that interference has had on their service.

1.3 Objectives

This dissertation aims to develop an algorithm that takes into account interference values and works towards reducing its impact on the Cloud service. The interference values come from previous observation studies in existing articles. By using a Cloud simulator and inserting the interference values, the performance degradation can be observed and different schedulers compared to perceive how much of an impact the interference causes and how well the new scheduler performs. As the baseline of the scheduling algorithm, an open-source scheduler is used, *OpenStack's Filter Scheduler* [Gon12]. The simulation is made with the SimGrid framework.

1.4 Report Structure

This report is divided into five additional chapters. Chapter 2 reviews previous works on the field of cloud networks, scheduling tasks for clouds and virtualization interference. Chapter 3 explains some concepts about cloud and virtualization technology and presents the data on interference

effects. Chapter 4 presents the scheduling algorithms used and how the interference was simulated. Chapter 5 presents the results of the simulation. Chapter 6 closes the report and presents some additional ideas on how to resolve the interference problem.

Chapter 2

State of the Art

The following chapter identifies previous work on the subject of Cloud computing, its uses and proliferation. Later it presents studies of task scheduling on computational networks, both in the Cloud and High Performance Computing (*HPC*) networks, followed by studies showing the degradation of performance on co-allocated VMs and previous work using the selected simulation framework, SimGrid.

2.1 Cloud Computing

Cloud computing is a general term to refer to several similar services. Multiple definitions were collected in [VRMCL08], and the consensus presented is:

> "Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and / or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs."

In the article by Armbrust *et al.*[AFG+10], the authors briefly present the utility of the computational services offered by the Cloud vendors and the major challenges and opportunities in the technology. In [BYV+09], the authors celebrate computing as the "fifth utility (after water, electricity, gas, and telephony)", with Clouds being one of the major distributors. They proceed to explain the business model used by main Cloud vendors and the service options they offer. The contracted services are stipulated in a SLA, making the vendors responsible for achieving an agreed desired performance of the service. For the fulfilment of this performance, over-provisioning of resources is made to ensure that delays and critical host failures are compensated by having a redundant number of hosts, as shown in [AFG+10, NKG10, CFF12]. This severely reduces the efficiency ($

/ machine) of the resources a vendor has available. In [WGL+12] a comparison is made between two major Cloud management programs, OpenStack and OpenNebula. Vecchiola *et al.*[VPB09] demonstrate the utility of Cloud computing (more explicitly, HPC) in the scientific / medical environment, presenting a study about the application of the computing power in the classification of gene expression data and Functional Magnetic Resonance Imaging (*fMRI*) compilation. The work of the dissertation is based on the OpenStack scheduling algorithm, described in [Gon12].

2.2 Virtualization Technologies

Virtualization is the technology that makes it possible for a single computer to run multiple independent working environments, executing simple tasks or full Operating Systems (*OS*) [SN05]. Virtual Machines appeared in the 1970's when hardware costed tens of thousands of dollars [Men05]. This allowed enterprises to have multiple employees working on the same mainframe on their own environment. As hardware became more affordable in the 1980's their use declined, however, this trend has reversed in the past years due to an increase in reliability and decrease in costs.

Multiple VMs on a host are created and managed by an hypervisor. The hypervisor can run as a program inside an existing OS, such as the *Kernel-based Virtual Machine* (*KVM*) [Hab08], or contain their own OS, such as *Xen* and *VMware* [GZ12]. Comparison between different hypervisors is available [Hab08, GZ12, YHB+11]. The tests made by Younge *et al.* favour the KVM hypervisor for managing HPC Clouds[YHB+11]. Xenoprof, a tool developed for the Xen hypervisor, identifies virtualization costs on network transfers and pinpoints a slowdown in data transfer rate when there are more than two network interfaces because the host becomes CPU capped instead of the network speed cap, having fifteen percent less performance than a non-virtualized environment due to overhead[MST+05].

2.2.1 Interference on Virtualized Environments

Interference between VMs is a phenomenon that causes performance decay due to multiple VMs racing for the same resources that cannot be fully separated, such as high level cache [PLM+10]. This is demonstrated experimentally by observing execution times of paired tasks, with various task types using different computational resources, reaching the conclusion that the pairing of tasks that depend on different resources results in a decreased interference. Koh *et al.* observed the same performance hits and devised a interference prediction model, based on data collected across different tasks and machines [KKB+07].

2.3 Task Scheduling on Clouds

There are several studies of scheduling tasks on Clouds, yet there is little consideration for VM interference on the scheduling rules and heuristics.

2.3.1 Offline Scheduling

On [ABN00] it is studied the use of probabilistic algorithms, such as genetic algorithms, simulated annealing and tabu search, for task scheduling. On [DK08] it is devised a list scheduling algorithm. However, the two previous studies rely on having full knowledge of all the tasks coming in to the cloud, a rare occurrence on the Clouds.

2.3.2 Online Scheduling

On [THW02] the more common used algorithms in the industry are presented and evaluated. Those presented are Heterogeneous Earliest-Finish-Time (*HEFT*) and the Critical-Path-on-a-Processor (*CPOP*), algorithms focused on delivering a good quality schedule on a fast run time, for heterogeneous systems. The OpenStack algorithm is based on the HEFT algorithm [Gon12].

Some research focus on scheduling scientific applications, complex workflows which can be divided in atomic tasks [Meh13]. To resolve this NP-complete scheduling problem, Gupta *et al.* use a location based heuristic, trying to minimize the network distance between tasks of the same workflow, in an attempt to accelerate the data-sharing process time [GMK12]. Chen and Zhang follow the same pattern, using a ant-colony optimization [CZ09]. Su *et al.* use a Pareto dominance algorithm to schedule the workflow in a way that minimizes the monetary costs, iterating over their solution repeatedly to improve it [SLH+13]. Zheng and Sakellariou extend the HEFT algorithm to make it more fit for workflows [ZS13].

Schedulers generally control the task allocation for the whole Cloud, however, Cucinotta *et al.* present a distributed scheduling system in which each node is responsible for assuring the quality of the scheduling, dynamically changing the resources allocated to each VM [CGFC11].

2.3.3 Interference Aware Scheduling

The following research incorporate virtualization interference on their approaches, showing an improvement on all cases. By reserving resources without a specific task assigned in each host, Nathuji *et al.* were able to produce better results by using the unused resources to compensate interference and allocate them to one of the VMs present [NKG10]. Zhu and Tung developed a prediction model which they use to define rules for task assignment, deciding if one task can be executed concurrently with another on the same host or should wait its completion [ZT12]. Govindan *et al.* developed a scheduler that assigns tasks based on cache memory usage by each task [GLKS11]. Chiang and Huang use machine learning algorithms to improve their scheduling heuristic with data collected during the Cloud operation [CH11].

2.4 Simulators

There are a plethora of Cloud simulators or simulation frameworks. A comprehensive list and comparison by Malhotra and Jain is available [MJ13]. SimGrid is the framework to be used for development and evaluation, because it is open-source, highly scalable and has previous work as a

Cloud simulator [DRC+13]. Articles by Casanova and Legrand demonstrate several use cases of SimGrid, its evolution over the years and its academic contributions [Cas01, LMC03, CLQ08].

2.5 Conclusion

The study of previous work reveals the importance of Clouds in the current Internet industry. Being a technology still on development, it presents challenges and opportunities. The problems identified in the technology deserve both corporate and academic attention, in an effort to solve them and improve the service provided. The studies about interference reveal a problem with current virtualization technology, showing a reduction of performance that can go as far as one third of the original performance, making that two serialized tasks can outperform the same tasks parallelized. The interference problem is due to sharing of non-dividable resources, like cache memory or network connections, making VMs race for them and get in the way of each other instead of cooperating. This can be detected and prepared for at schedule time, making sure the tasks get minimal interference or wait for a better time to start. Other solutions include developing different and specialized hardware, that is completely dividable for each VM.

Chapter 3

Cloud Computing

Cloud providers run a multi-millionaire service for billionaire enterprises, virtual servers are a staple of today's internet services, but the question remains, what is a cloud. A lot of different definitions can be found on the works cited in 2.1 but the simplest and broadest is [AFG+10]:

> "Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services."

In order to provide these services, providers rely heavily on virtualization for reliability and security. The work of this dissertation aims to develop a better scheduler for tasks in a cloud focusing in the impact of interference between tasks, improving the performance of the cloud and increasing availability.

3.1 Virtualization

Virtualization is the key component of every cloud. Virtualization software creates a virtual computer on a real[1] computer, enabling a server to appear like multiple different servers, each with its own operating system and isolated from the other virtual machines. VMs are easily replicated and destroyed, allowing for a fast deployment as more or less processing power is required. Their independence of the host installed OS and libraries ensures that any program running on a VM will run on any computer that has a copy of that VM and meets the hardware requirements, so the program can be developed and tested with a single VM instance running on a personal computer and then deployed to the cloud.

Virtual machines are created and controlled by a hypervisor[2]. The hypervisor can be of two types:

[1] VMs can be executed inside another VM, but it would be extra complexity and overhead for zero gain, except for some particular edge cases.

[2] A hypervisor can also be called a virtual machine monitor.

- **Native** when the hypervisor is running directly on the host's hardware.
- **Hosted** when the hypervisor is running as a program inside the host's OS.

Hosted hypervisors have more execution overhead so cloud providers stick to native hypervisors. The hypervisor is responsible for the host's resource distribution for its VMs, called guests. The resources distributed are:

- **CPU** a whole CPU or individual cores from a multi-core CPUs.
- **RAM** can be any discrete value.
- **Disk** portions of a disk, whole partitions or entire disks.
- **Input and Output devices** like network cards, USB ports and CD drives.

Although most of these resources can be shared between guests, they are kept separated for security and performance reasons.

3.2 Task Scheduling in Clouds

Cloud systems have a scheduler responsible for assigning tasks to a host. The scheduler communicates with the hypervisors to keep a list of status of the hosts and tasks. When a new task is requested to the cloud, the scheduler assigns the task to a host that meets the tasks requirements and has free resources to execute it. This type of scheduling is called *on-line scheduling*, executing a selection algorithm each time a task arrives, opposed to *off-line scheduling*, a type of scheduling that requires the knowledge of the entire task list to execute an algorithm a single time and determine from the start where each task is executed.

3.3 Interference between Virtual Machines

Virtual machines are assigned their own set of exclusive resources on their host as explained on section 3.1. However not all resources can be assigned to a specific VM because they are hardware or firmware controlled and are shared between multiple guests, as is the case with cache memory and memory buses. The sharing of these resources cause a race condition for them and one guest will perform almost optimally while the others are under-performing due to a bottleneck in these resources. This event is called interference and is triggered when a task is executing slower instructions than its neighbours. For example, when a task is retrieving data from disk it will quickly loose cache space to a task executing arithmetical operations, resulting in its later operations being slower due to an increase in cache misses and requiring these values from RAM. On an Intel i7 2.4GHz, RAM access is five times slower than L3 cache access [Lev09].

The interference effect values used for this simulation is based on prior research by [KKB+07] and [ZT12]. The graph 3.1 shows the performance of a pair of collocated VMs each executing a

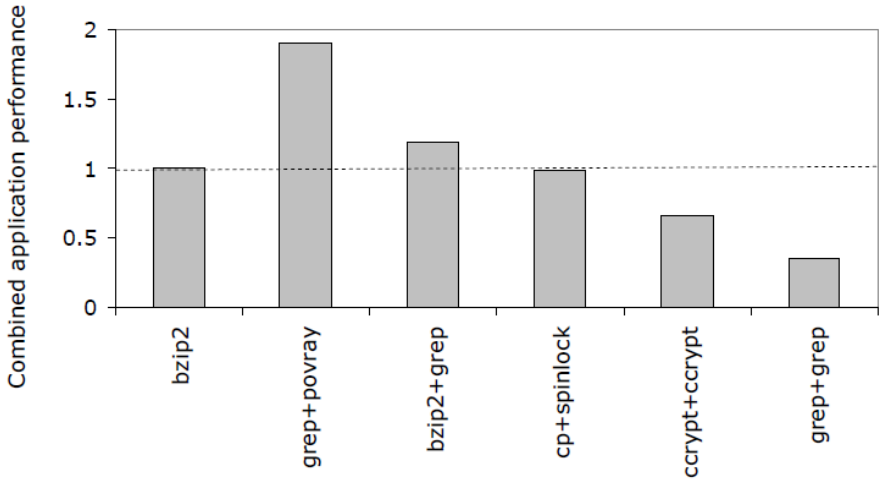

Figure 3.1: Normalized performance of two virtual machines executing *shell* commands. Source: [KKB+07]

shell command. The VMs have the same amount of resources and the host has enough resources to offer them exclusively to each VM. The result is normalized to their standalone execution time. As the graph shows, the performance of the **grep** command with the **povray** command has a performance score close to 2, meaning it's almost as fast as if they were running alone and do not interfere with each other. However, other combinations show a performance below 1, meaning they would be faster if executed sequentially rather than in parallel. The worst case has a performance of 35% for two parallel **grep**s, meaning that the two parallel tasks take almost as long as running six of them sequentially.

The work in [ZT12] compares the execution time of disk I/O intensive tasks running in parallel on the same host with another tasks that have different resource bottlenecks, namely CPU intensive tasks, memory intensive tasks, network communication intensive tasks and disk I/O intensive tasks. The graph 3.2 shows that bigger tasks cause bigger interference and the interference is highest between two disk I/O intensive tasks, reaching a slowdown bigger than two, meaning the two tasks would be faster if executed sequentially instead of concurrently. On the other hand, a disk I/O intensive task suffers almost no slowdown when paired with a CPU heavy task.

3.4 SimGrid for Cloud Simulation

Simgrid is a complex framework, divided in several modules. For the purpose of this work, the most suitable module is MetaSimGrid (MSG), the module commonly used for scheduling tests and simulations. However, Simgrid has no simulation of interference between tasks so this must be introduced into the simulation.

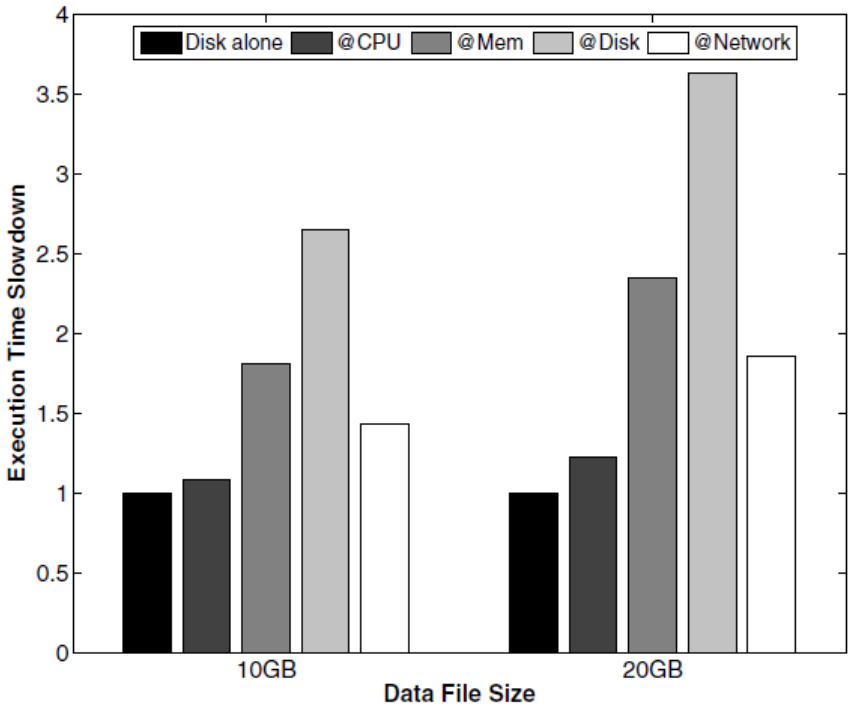

Figure 3.2: Execution time of two collocated VMs. Source: [ZT12]

3.5 Conclusion

The data studied clearly shows that task interference heavily impacts the performance of the hosts. The development of effective fixes to this problem is of high academic and industrial interest. In the next chapter it is presented the scheduling algorithm developed to minimize the effects of interference.

Chapter 4

Approach

The following chapter describes the work developed during this dissertation, starting with the definition of the data structures used. The first implementation detail was introducing interference between tasks as is described on 3.3, moving onto developing a basic cloud scheduler and then evolving that scheduler to take interference into account. Please note that, in this chapter and the next, a task is the same as a VM, since each VM executes a single task and both terms can be used interchangeably.

The schedulers have knowledge of the type of each task. The values for interference used on the simulation are on table 4.1. These are an approximation of the values shown on 3.3, but the interference aware scheduler uses different, but close, values to these to calculate the expected interference, as seen on table 4.2. The schedulers have no knowledge of how much time each task should take under optimal conditions.

Task type	CPU	Memory	Disk
CPU	0.94	0.96	0.91
Memory	0.92	0.84	0.85
Disk	0.79	0.77	0.65

Table 4.1: Interference values used in the simulation

Task type	CPU	Memory	Disk
CPU	0.94	0.96	0.91
Memory	0.92	0.84	0.85
Disk	0.79	0.77	0.65

Table 4.2: Interference values expected by the scheduler

4.1 Data Structures

There are two important data structures utilized in the simulation, task and host. Understanding them is simple but key for understanding how the simulation works.

4.1.1 Task

Each task instance is described by their type, amount of computation cycles required, creation, start time and finish time, number of required CPU cores and how much memory and disk space they need. The task type can be CPU, Memory or Disk, dictating how they interact with other tasks, how much interference they cause and how much they suffer. The amount of computation cycles required is invisible to the scheduler and is only used to determine when a task should end.

4.1.2 Host

Each host instance contains a list of running tasks, and is described by its speed[1] available number of CPU cores, available memory and available disk space.

4.2 Introducing Interference on Simgrid

Simgrid, specifically the MSG module, only has two ways of tuning down the performance of a host, thus simulating the effects of interference. This tune down is made by either reducing the speed of the host or changing it to another power state, however neither approach has effect on tasks already running on the host and requires them to be paused and continued. In order to execute this pause and resume there needs to be a process overseeing the host and its tasks, making it extremely less practical. The power states also have to be defined before the simulation and can only be discrete values. These approaches also affect the tasks in the same manner and, as seen earlier, some tasks cause heavy interference on another tasks but suffer little interference themselves.

Another approach involved creating dummy tasks that would consume resources on the host for some time, thus simulating the interference, but suffering the same problem above that tasks are affected equally.

To resolve this issue, the MSG functions for automatically executing tasks in the host were dropped, and the tasks were ticked down by a custom process running for each task. The process knows which host it is executing on and which tasks it is executing and calculates how much interference is being done in that time instance and subtracts the amount of work done on that tick to the task until the amount required to complete the task reaches zero and the task is marked as completed. This allows to simulate accurately the interference on each task. The interference is calculated by the following formula:

[1] CPU speed is defined by computation cycles per unit of time.

$$Interference = \frac{\prod_{i=1}^{numberoftasks} interferencematrix[type_{current}][type_i]}{interferencematrix[type_{current}][type_{current}]}$$

Since the current task is included in the task list and therefore is calculated on the product operation it is later removed by the division of the interference with itself, since a task doesn't interfere with itself. The number of computation cycles a task has executed in an interval of time is:

$$Cycles = TimeElapsed * CoresAssigned * CoreSpeed * Interference$$

Time elapsed is a constant that can be arbitrarily defined, but should not be too long. The value used for the simulations presented was five. For each task, a VM instance is created for that task with its required capacities but the VM feature in MSG is still new and underdeveloped and doesn't really affect the outcome, simply creating a task and assigning it to the required number of CPU cores would achieve the same result.

Interference is an unpredictable event and two collocated tasks can suffer different interference effects. This would call for introducing random variations on the interference value, however the random values can have a higher effect on one of the simulations and affect the results. Since the purpose is to compare the algorithms rather than calculating a real world value for the time the tasks would take, removing the randomness component and using average values allows for a better comparison.

4.3 Basic Scheduler

The basic scheduler starts by filtering out hosts that do not meet the task requirements. Since there are no security requirements for the simulated tasks, the filter weeds out hosts that do not have the hardware requirement of the task, either because the host doesn't have them or they are already being used by earlier tasks. From the remaining hosts the scheduler seeks:

- The host with the minimum value of concurrent tasks divided by its speed.
- In case of a tie on the first condition, the host with more free resources from the hosts tied for the first condition.
- In case there is still a tie, the first host from the hosts tied for the second condition.

The first condition guarantees that faster hosts will run more tasks, even when they have more tasks assigned than slower hosts, provided that they are faster by a factor of at least the number of additional tasks they have, e.g. a host that is at least twice as fast can be selected over a host that has half the tasks running on it. Additionally, having less tasks means having less interference, so this scheduler already reduces the interference condition by spreading tasks evenly over hosts with the same speed. Openstack's FilterScheduler only checks for the second condition after filtering, selecting a random host from the list of hosts with more resources available [Gon12].

4.4 Interference Aware Scheduler

The interference aware scheduler builds upon the basic scheduler and differs very little from it. Yet the small differences are key to achieve a better assignment and minimizing interference effects. The scheduler seeks:

- The host with the maximum value of expected interference multiplied by the host's core speed.
- In case of a tie on the first condition, the host with more free resources from the hosts tied for the first condition.
- In case there is still a tie, the first host from the hosts tied for the second condition.

The first condition is the different condition. Instead of selecting a minimum, this equation gives higher score to better hosts, instead of a lower score. The equation is:

$$Score = ExpectedInterference_{host} * CoreSpeed_{host}$$

Expected interference is the interference the scheduler expects the task to find on the host, using the approximated interference values instead of the real values and is calculated similar to the host interference, except since the task is not yet present on the host's task list it doesn't need to be removed from the final value:

$$ExpctdInt = \prod_{i=1}^{numberoftasks} ExpctdIntMatrix[type_{new}][type_i]$$

4.5 Conclusion

A small change in the scheduling algorithm can have a big impact in the performance of the hosts. In the next chapter it's shown the improvements this approach generated.

Chapter 5

Result Analysis

The following chapter describes the test generation process and the grid configurations used for testing. Simgrid requires a task file, containing the tasks to be performed by the cloud and a platform file, describing the configuration of the grid and the capacity of each computing node. Finally, it presents the simulation results between the two algorithms. Note that the simulation is independent of a unit of time, so for simplicity and readability the unit used is **second**.

5.1 Test generation

The tests were randomly generated with a program developed for the purpose of this dissertation. It generates a list of tasks, described by their type and requirements. The requirements generated fall into discrete categories, with each category having a probability of occurring with an addition of hard restrictions that can affect these probabilities. These probabilities were modified to generate different tests, to observe the interference effect on different workloads and cloud usage. The hard restrictions are consistent between all the generated tests:

- If a task requires more than 500 gigabytes of disk it is a disk type task.
- If a task requires less than 2 gigabytes of disk it is either a CPU or Memory type task.

Three different task files were generated to evaluate the performance of the algorithms.

- **Test 1** consists of 100 tasks, with 36% CPU tasks, 35% memory tasks and 29% disk tasks. The last task arrives at second 335. This test was conceived with the same probability for each task type.
- **Test 2** consists of 160 tasks, with 28% CPU tasks, 22% memory tasks and 50% disk tasks. The last task arrives at second 551. This test was conceived to have much more disk type tasks since the interference effects are heavier on this kind of tasks. The tasks are spread

over a period of time very close to the generated on test 1, with a task arriving each 3.44 seconds on average compared to the 3.35 of Test 1.

- **Test 3** consists of 250 tasks, with 32% CPU tasks, 34% memory tasks and 33% disk tasks. The last task arrives at second 426. This test was designed to overload the grid, with a task coming every 1.7 seconds on average.

5.2 Grid configuration

Three different configurations were used, a homogeneous configuration, and two heterogeneous configurations. All of them have full routing between the hosts with 15 megabit bandwidth and 10 millisecond latency, however since network tasks aren't used these values don't affect the simulation. The first configuration, *A*, consists of 10 hosts with 4 cores each and 100 million computations per second (*Mcps*). Configuration *B* has a total of 10 hosts with 4 cores each:

- two nodes capable of 50Mcps
- four nodes capable of 80Mcps
- four nodes capable of 100Mcps

Configuration B is a slightly weaker configuration than *A*. Configuration *C* is the strongest and most diverse of the three with a total of 12 hosts, comprised by:

- two 4 core 50Mcps nodes
- two 4 core 80Mcps nodes
- two 4 core 100Mcps nodes
- two 8 core 80Mcps nodes
- two 8 core 100Mcps nodes
- two 12 core 100Mcps nodes.

Configuration C is used on less tests than A or B, since it has much more power, usually resulting in over provisioning the resources needed and resulting in very small differences between the algorithms, since there are more hosts and more power the interference effects are much lower.

5.3 Results

The following tables show the average time it took to complete each task on the simulated grid and compares the average time by both algorithms, calculating the performance increase of the interference aware version compared to the normal algorithm.

Task Type	Average time		Performance
	Normal	Aware	
Cpu	61,94	62,64	98,89%
Memory	67,14	65,71	102,17%
Disk	69,48	66,72	104,13%
Total	65,95	64,9	101,62%

Table 5.1: Results of simulation with task file 1 on configuration A

Table 5.1 shows a very marginal increase in performance with the task set 1 and configuration A. However if we observe the same task set on configuration B 5.2, the algorithm performs 10% better, meaning configuration A has enough resources to run the set without being much stressed. Since configuration A is already sufficient to run the tasks smoothly there is no need to evaluate with the more powerful grid C.

The tests with task set 2 show the same previous trend, where the interference aware version performs much better when there are less resources to go around for all tasks, doubling the performance gain from configuration A to configuration B. Tables 5.3 and 5.4 also show that the interference aware algorithm increases the performance on disk heavy tasks severely, more so than on other tasks types, even increasing the average time on CPU tasks but contributing to the collective of the tasks.

The tests with task set 3 do not strictly observe the trend that the performance increase is bigger with less resources as 5.5 and 5.6 show, but the increased performance is very good since there are many more tasks assigned to the grid. Since the grids are saturated on this task set, grid C was used and the results on 5.7 are positive even with such capacity increase.

5.4 Conclusion

As the tables show, the algorithm makes slight improvements on the grid performance in all of the test cases. Even the worst case improvement observed is non-negligible. A 10% improvement means that each day a cloud can save 2.4 hours, or 144 minutes, of free time that can be used

Task Type	Average time		Performance
	Normal	Aware	
Cpu	77,64	65,97	117,68%
Memory	85,57	80,43	106,39%
Disk	93,97	87,93	106,86%
Total	85,15	77,4	110,01%

Table 5.2: Results of simulation with task file 1 on configuration B

Task Type	Average time		Performance
	Normal	Aware	
Cpu	60,78	61,89	98,20%
Memory	66,29	65,57	101,09%
Disk	77,56	70,63	109,82%
Total	70,38	67,06	104,94%

Table 5.3: Results of simulation with task file 2 on configuration A

Task Type	Average time		Performance
	Normal	Aware	
Cpu	76,78	67,44	113,84%
Memory	86,00	83,43	103,08%
Disk	116,31	106,56	109,15%
Total	98,56	90,50	108,91%

Table 5.4: Results of simulation with task file 2 on configuration B

Task Type	Average time		Performance
	Normal	Aware	
Cpu	213,92	217,22	98,48%
Memory	905,86	937,59	96,62%
Disk	2 841,37	1 797,20	158,10%
Total	1 337,54	998,78	133,92%

Table 5.5: Results of simulation with task file 3 on configuration A

Task Type	Average time		Performance
	Normal	Aware	
Cpu	259,81	203,80	127,48%
Memory	1 094,94	741,84	147,60%
Disk	2 320,18	2 043,69	113,53%
Total	1 242,72	1 009,24	123,13%

Table 5.6: Results of simulation with task file 3 on configuration B

Task Type	Average time		Performance
	Normal	Aware	
Cpu	222,28	189,43	117,34%
Memory	914,89	741,95	123,31%
Disk	1 889,11	1 919,52	98,42%
Total	1 023,36	963,02	106,27%

Table 5.7: Results of simulation with task file 3 on configuration C

to save electricity, perform hardware maintenance or run another tasks. Investing on software improvements can lead to better improvements than investing on better or more hardware.

Chapter 6

Conclusion

6.1 Objective Fulfilment

The proposed scheduling algorithm developed showed promising results in simulation, achieving better performance on all tests than the base algorithm. One flaw of the proposed solution is that it's not self-learning and requires the interference effects to be known previously.

6.2 Future Work

The simulation results are interesting for real world application. Therefore, it's implementation in cloud schedulers can be explored. The work on interference is still very infant and there should be a push to explore interference effects on current hardware, since the ones observed earlier are a couple of years old and on the technological world, that is a very long time. There's also the possibility to explore interference effects on another virtualization technologies, or container technology, like *Docker*.

Conclusion

References

[ABN00] Ajith Abraham, Rajkumar Buyya, and Baikunth Nath. Nature's heuristics for scheduling jobs on computational grids. In *The 8th IEEE international conference on advanced computing and communications (ADCOM 2000)*, pages 45–52, 2000.

[AFG+10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[BYV+09] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[Cas01] Henri Casanova. Simgrid: A toolkit for the simulation of application scheduling. In *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, pages 430–437. IEEE, 2001.

[CFF12] Antonio Corradi, Mario Fanelli, and Luca Foschini. Vm consolidation: a real case based on openstack cloud. *Future Generation Computer Systems*, 2012.

[CGFC11] Tommaso Cucinotta, Dhaval Giani, Dario Faggioli, and Fabio Checconi. Providing performance guarantees to virtual machines using real-time scheduling. In *Euro-Par 2010 Parallel Processing Workshops*, pages 657–664. Springer, 2011.

[CH11] Ron C Chiang and H Howie Huang. Tracon: Interference-aware scheduling for data-intensive applications in virtualized environments. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 47. ACM, 2011.

[CLQ08] Henri Casanova, Arnaud Legrand, and Martin Quinson. Simgrid: a generic framework for large-scale distributed experiments. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, UKSIM '08, pages 126–131, Washington, DC, USA, 2008. IEEE Computer Society.

[CZ09] Wei-Neng Chen and Jun Zhang. An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 39(1):29–43, 2009.

[DK08] Mohammad I Daoud and Nawwaf Kharma. A high performance algorithm for static task scheduling in heterogeneous distributed computing systems. *Journal of Parallel and distributed computing*, 68(4):399–409, 2008.

REFERENCES

[DRC+13] Frédéric Desprez, Jonathan Rouzaud-Cornabas, et al. Simgrid cloud broker: Simulating the amazon aws cloud. 2013.

[Fou13] OpenStack Foundation. Cern uses openstack. http://www.openstack.org/user-stories/cern/, 2013.

[GLKS11] Sriram Govindan, Jie Liu, Aman Kansal, and Anand Sivasubramaniam. Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 22. ACM, 2011.

[GMK12] Abhishek Gupta, Dejan Milojicic, and Laxmikant V Kalé. Optimizing vm placement for hpc in the cloud. In *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*, pages 1–6. ACM, 2012.

[Gon12] Yong Sheng Gong. Openstack nova-scheduler and its algorithm. https://www.ibm.com/developerworks/community/blogs/e93514d3-c4f0-4aa0-8844-497f370090f5/entry/openstack_nova_scheduler_and_its_algorithm27?lang=en, April 2012.

[GZ12] Zonghua Gu and Qingling Zhao. A state-of-the-art survey on real-time issues in embedded systems virtualization. *Journal of Software Engineering and Applications*, 5(4):277–290, 2012.

[Hab08] Irfan Habib. Virtualization with kvm. *Linux Journal*, 2008(166):8, 2008.

[HKKS03] Matthias Hovestadt, Odej Kao, Axel Keller, and Achim Streit. Scheduling in hpc resource management systems: Queuing vs. planning. In *Job Scheduling Strategies for Parallel Processing*, pages 1–20. Springer, 2003.

[KKB+07] Younggyun Koh, Rob Knauerhase, Paul Brett, Mic Bowman, Zhihua Wen, and Calton Pu. An analysis of performance interference effects in virtual environments. In *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on*, pages 200–209. IEEE, 2007.

[Kon14] Ev Kontsevoy. Onmetal: The right way to scale. http://www.rackspace.com/blog/onmetal-the-right-way-to-scale/, June 2014.

[Lev09] David Levinthal. Performance analysis guide for intel core i7 processor and intel xeon 5500 processors. *Intel Performance Analysis Guide*, 2009.

[LMC03] Arnaud Legrand, Loris Marchal, and Henri Casanova. Scheduling distributed applications: the simgrid simulation framework. In *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*, pages 138–145. IEEE, 2003.

[Meh13] Gaurang Mehta. Workflowgenerator pegasus. https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator, October 2013.

[Men05] Daniel A Menascé. Virtualization: Concepts, applications, and performance modeling. In *Int. CMG Conference*, pages 407–414, 2005.

REFERENCES

[MJ13] Rahul Malhotra and Prince Jain. Study and comparison of various cloud simulators available in the cloud computing. *International Journal*, 3(9), 2013.

[MST+05] Aravind Menon, Jose Renato Santos, Yoshio Turner, G John Janakiraman, and Willy Zwaenepoel. Diagnosing performance overheads in the xen virtual machine environment. In *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, pages 13–23. ACM, 2005.

[NKG10] Ripal Nathuji, Aman Kansal, and Alireza Ghaffarkhah. Q-clouds: managing performance interference effects for qos-aware clouds. In *Proceedings of the 5th European conference on Computer systems*, pages 237–250. ACM, 2010.

[PLM+10] Xing Pu, Ling Liu, Yiduo Mei, Sankaran Sivathanu, Younggyun Koh, and Calton Pu. Understanding performance interference of i/o workload in virtualized cloud environments. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 51–58. IEEE, 2010.

[SLH+13] Sen Su, Jian Li, Qingjia Huang, Xiao Huang, Kai Shuang, and Jie Wang. Cost-efficient task scheduling for executing large programs in the cloud. *Parallel Computing*, 2013.

[SN05] James E Smith and Ravi Nair. The architecture of virtual machines. *Computer*, 38(5):32–38, 2005.

[THW02] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *Parallel and Distributed Systems, IEEE Transactions on*, 13(3):260–274, 2002.

[VPB09] Christian Vecchiola, Suraj Pandey, and Rajkumar Buyya. High-performance cloud computing: A view of scientific applications. In *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, pages 4–16. IEEE, 2009.

[VRMCL08] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[WGL+12] Xiaolong Wen, Genqiang Gu, Qingchun Li, Yun Gao, and Xuejie Zhang. Comparison of open-source cloud management platforms: Openstack and opennebula. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2457–2461. IEEE, 2012.

[YHB+11] Andrew J Younge, Robert Henschel, James T Brown, Gregor von Laszewski, Judy Qiu, and Geoffrey C Fox. Analysis of virtualization technologies for high performance computing environments. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 9–16. IEEE, 2011.

[ZS13] Wei Zheng and Rizos Sakellariou. Budget-deadline constrained workflow planning for admission control. *Journal of Grid Computing*, pages 1–19, 2013.

[ZT12] Qian Zhu and Teresa Tung. A performance interference model for managing consolidated workloads in qos-aware clouds. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 170–179. IEEE, 2012.