Summary
6/1/12 12:06:30 PM +01'00'

Differences exist between documents.

New Document:	Old Document:
thesis_1.0	thesis_v1.1
73 pages (1.74 MB)	76 pages (1.73 MB)
6/1/12 1:06:03 PM +01'00'	6/1/12 1:05:58 PM +01'00'
Used to display results.	

Get started: first change is on page 6.

No pages were deleted

How to read this report

Highlight indicates a change.
~~**Deleted**~~ indicates deleted content.
 indicates pages were changed.
 indicates pages were moved.

Measuring the Performance and Availability of Cloud Infrastructures

by

©Arjan Borst

Master of Science

University of Amsterdam - Master Grid Computing

June 2012

Abstract

Cloud computing has already proven to be the path to utility computing. It is showing a fast use in many enterprise applications. The focus of this paper is on the non-functional requirements; performance and availability. By monitoring the availability and I/O related performance metrics we were able to provide detailed information about the resource consumption, connection latency and uptime of a cloud server. By integrating multiple IaaS providers into an existing enterprise environment a performance comparison could me made. Cloud providers can use this research to monitor their own I/O subsystem and hypervisor server, identify heavy users and prevent performance degradation. For current and future cloud users we provide guidance in selecting a cloud provider, what to expect from a cloud provider in terms of performance gain or loss, and how to monitoring on availability and performance.

Acknowledgements

First and foremost I would like to thank my daily supervisor and associate, drs. Jeffrey Barendse, for allowing to work on my Thesis during the last nine months. I would also like to thank Dr. Adam Belloum for his assistance and feedback and Masha Staal who helped me out with the English grammar. For helping me out with some nasty socket and threading problems during development I want to thank my colleague Jeroen Kamp. Finally, I would like to thank my girlfriend who supported me during the course of this dissertation. Without her assistance this work would not have been possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

During the past five years there has been an increase of implementing cloud services in new and existing projects. Armbrust from UC Berkeley describes the elasticity of cloud resources without paying a premium for large scale, as unprecedented in the history of IT [20]. Recent publications report many successful implementations of cloud computing services done by enterprises and governments [27, 28, 3].

In 2010, Gartner predictions showed that cloud computing is not just a hype [13]. Companies started and will continue to integrate their businesses using cloud services to become more dependent of the ever-increasing complex and costly infrastructure needed to support their businesses. However, when cloud services fail this new dependency can result in downtime for mission-critical applications, or even permanent loss of data. The examples reported in [7, 20] show clear evidence of a lack of a mature service level management and delivery.

In such infrastructures the resources between instances are real timed scheduled and optimized, therefore service availability is difficult to achieve when instances need a huge amount of additional resources. It is likely that performance degradation will occur and disrupt services because of resource sharing between competing instances.

The tradeoff between guaranteed resources or economical efficiency and elasticity remains a serious challenge for cloud users and cloud providers.

The primary focus of this thesis is on the importance of the non-functional requirements; performance and availability. These key requirements are important for cloud providers and cloud users to deliver their services as intended to their clients.

While the availability can be monitored by two simple metrics namely the uptime and connection time, the monitoring of the performance requires a more elaborated experiment, one approach is to determine how the providers perform under a continuous load. This technique is better known as 'Endurance testing', and provides a way to test a service without risk and gives an opportunity to investigate limitations and bottlenecks. Endurance testing aims at performing multiple experiments with different load characteristics with the goal to assess the quality of a given cloud provider over time.

Accurate measurements can be achieved by monitoring low level metrics supplied by the Linux kernel however in the case of cloud computing the hypervisor plays a major role and thus needs to be carefully monitored when studying the performance of cloud providers. This makes it possible to gain more insight information about the resource consumption and distribution on the hypervisor level. To achieve this goal we performed a number of experiments which aim at testing the I/O subsystem.

Cloud Computing uses virtualization technology to provide resource sharing. This is possible because it is known that physical servers do not utilize all resources. With virtualization, resources are shared and as a result hardware is utilized more efficiently. The difference between storage resources and other shared resources is that heavy utilization can be solved by using 'live migration' [14], which consists of moving an instance without noticeable downtime from one server to another with more resources available. Such a technology for moving disk images between different I/O subsystems

is currently not supported by all common hypervisors [14].

By performing a case study, performance opportunities are further explored by integrating cloud services into an IT environment. This study involves a website which has an average of 2500 active visitors and at peak times more than 5000 active visitors. The site has over 150,000 members, and is receiving an average of 14+ million pageviews per month. It is within the top 3.000 of biggest sites within the US. The adaption of this environment consist of two steps. The first step of this case study involves the adaption of this environment to work with multiple datacentre locations where the servers of the cloud provider are. This is done by integrating a cloud based traffic management solution into the existing environment to distribute traffic to the selected cloud providers. The second step is about selecting and adding cloud servers to the current environment to create a hybrid infrastructure. For both steps a comparison is made to research differences in performance between current and adapted environment. The end result of this case study is a hybrid architecture where multiple provides can be added or removed to extend current infrastructure.

The next section is named 'Related Work', were similar work will be discussed and explained why this research is different. The remainder of this thesis contains all conclusions. This thesis starts with an introductory chapter named 'Theoretical Background'. Here, different terms will be introduced which are related to the term cloud computing and explains more about the available types of cloud computing. Further it will explain what performance testing is and which metrics are available to test. In the next chapter named 'Experiment Setup' the different tools and developed applications are introduced and explained how the experiments are performed.

1.1 About the author

When someone would ask me to describe myself I would say that I am a multi personal character. As a software engineer I am able to think very abstract and highly structured. As a person I am creative and have excellent communication and collaboration skills. In both professional and personal life I always strife for more. The above claims are based on experience and achievements. I have walked 100 km in 24 hours, the Nijmeegse march (4x50 km) and run the '21 km of Egmond'. Furthermore, I have a diving and rock climbing license. I always set out new goals such as running a marathon (Marathon of Amsterdam) and to walk 150 km (Amsterdam Leeuwarden). I am always eager to learn more: I already hold one master title and am currently finishing a second master, which is a research master in Grid Computing.

I selected this thesis subject because of the involved multidisciplinary competences which match my interests and combines my broad knowledge about many topics and fields and hand-on experience in many technical ICT related disciplines. This mixture of technical knowledge, creativity and persistence results in pervasive solutions, because "they are ill discoverers that think there is no land, when they can see nothing but sea." (Francis Bacon, The Advancement of Learning).

1.2 About WireITup ICT Solutions

My daily work will be done at the office of my company WireITup ICT solutions which is located in Wormerveer. It's a company specialised in hosting solutions for SME. It has seven employees including the two owners.

The hosting facility is located at Evoswitch at Haarlem. At this location our virtualization platform is housed which has a total of 768GB of RAM and a total of 192 cores dived among 6 servers. Each server hosts between 10 and 50 virtual servers.

The servers are using KVM as their hypervisor and the operating system CentOS. During this research I monitor these hypervisor servers to collect valuable information about the behaviour of those servers under stress.

My normal work is developing new solutions for my customers. This thesis made a contribution about how to monitor at server and hypervisor level. This has already lead to the identfication of some heavy resource consumption by virtual servers which could otherwise not be monitored. The other research which involved the migration of a enterprise application to the cloud also made a great contribution to our service delivery. This had lead to the usage of DynDNS within multiple applications and prooved a great partner for extending our services.

Chapter 2

Theoretical Background

Literature, terminologies and background information used during this thesis are further explained in this chapter. It starts with the different cloud service types according to [1]. Secondly, more about performance testing and which type is selected for the experiments. Lastly the different types of metrics are explained and which of them are interesting to monitor during the different experiments.

2.1 Cloud Computing

Cloud computing builds further on existing technology such as virtualization, grid computing and distributed computing but with a different perspective and target audience. While grid computing and distributed computing are still mainly used for scientific applications [6], cloud computing technology is already adopted by a large number of enterprises and governments. When combined with load balancing cloud computing can deliver highly scalable infrastructure solutions. One approach to cloud computing is to offer scalable infrastructure as a service to companies and is known as 'IaaS' (Infrastructure as a Service). IaaS providers promise quick deployment, (infinite) scalability, almost zero upfront infrastructure investment, more efficient resource

Since the introduction of the term cloud computing, companies provide their existing services as a cloud service. This all, encouraged by the many definitions [19] and the immaturity of this new way of delivering services [7] companies introduce a variety of new cloud services.

2.1.1 Utility Computing

Cloud computing can be seen as the next step to utility computing where computing is described as the fifth utility (along with water, electricity, gas, and telephone). It would be a logical step, the same happened with electricity where every company had to produce their electricity on site. Take for instance a brewery operating a hundred years ago. As Amazon chief technology officer Werner Vogels famously puts it:
"They had to be experts in electricity to brew beer. Something is off there. These guys couldn't wait to dump their own generators and start to use electricity from other companies."
Today's organizations in the private and public sector faces the same problem with their IT and are forced to own and maintain their IT infrastructure themselves. Cloud computing could be the trend which hints to a future where companies can buy IT with the same ease as electricity [9].

2.1.2 Infrastructure as a Service

Cloud computing uses virtualization and hypervisor technology as their main building blocks for realizing a cloud infrastructure. This technology combined with load balancing can deliver highly scalable solutions. In cloud computing this infrastructure is offered as a service to companies and is known as IaaS. For instance you can rent one virtual server, configure it and run your software. When business demands are growing the sever can scale out in CPU, memory, disk I/O and bandwidth or increase

the number of instances using load balancing techniques to distribute the workload.

2.1.3 Software as a Service

Applications which runs on a cloud platform can be labelled as cloud services. They benefit of the scalable architecture provided by IaaS and when combined with an appropriate software architecture, the software can be provided as a service. Introducing new services into the cloud is not a straightforward job. Software engineers need to know new architectural and software patterns, programming environments and even execution environments. New designs needs to be loosely coupled and highly scalable, be able to run on different sites and fault tolerant.

2.1.4 HUman as a Service

This form of cloud computing focuses on using other people to perform some tasks. This type of cloud computing service is new and the most famous service which can deliver this is Amazon's Mechanical Turk. This kind of cloud computing is closely related to crowdsourcing which is introduced by Jeff Howe in 2006 [15] and is used to capture recent developments where companies and governments are using groups of people (crowds) for consultancy, development, innovation and research [15, 11]. These groups are not specified in advance and everyone (volunteers, professionals, experts) is welcome to participate in this process. Therefore open source development could be seen as one of the major types that we can distinguish in crowdsourcing. In open source, users of different disciplines can make contributions to the development of an application or website. Another interesting development in user participation is open innovation where users can help by contributing to an innovation project.

Using crowds is not something new, in history many examples exist where crowds participate in solving a dilemma. One of the most cited examples in literature is the

Cornish steam engine [21] open innovation project. This example starts in 1769 when James Watt patented an adjustment on the 'Newcomen Machine'. This resulted in a machine which was far more efficient than the competition and as a result mining companies in Cornwall bought many of these engines. Watt's refusal to license design improvements and his high royalty price resulted in more aversion to Watt's company [5]. The mining companies wanted to prevent this from happening again and they started a journal where all engineers could publish their ideas about making steam engines more efficient. In the 19th century the end result was one of the most efficient steam engines, which was not bound to any patents or other restrictions. This is clearly an example of using crowdsourcing to facilitate open innovation.

2.2 Performance testing

Performance testing is mostly associated with testing a service or system peak performance on a particular time. But it can also be used to investigate, measure, validate or verify other quality attributes of a system or service, such as scalability, reliability and resource usage. Endurance testing is a type of performance testing which test if a service or system can sustain a continuous expected load for a period of time. And by capturing resource usage it provides information about the quality of a service which can be used to show possible performance degradation.

Because most enterprise applications already know what their workload characteristics are, it makes more sense to select an IaaS provider which can deliver these characteristics, then selecting one based on a performance chart and neglect the possibility of performance degradation. This research explains how knowledge during the endurance test can be used in different ways. First it reveals different indicators which shows possible degradation of a service. Secondly it can be used as a baseline for fu-

ture reference. Lastly, when applying horizontal scaling, knowledge gained during the endurance test can be applied to down or upscale resources automatically.

To ensure that all metrics are captured during the test, a infrastructure needs to be constructed. Secondly, tools need to be built to simulate the workload characteristics and metrics need to be selected which are monitored. The next paragraph identifies the metrics which going to be monitored during the test. For system health purposes it is interesting to monitor different metrics from all subsystems available, which even can be extended with available metrics from daemons running on that system. This amount of metrics poses in combination with the frequency of checks results in a massive dataset. To solve this problem there is a need to carefully identify all metrics of importance and which are less important and the introduction of a system which can display results in a readable format.

2.3 Metrics

Because cloud services have shared resources it is possible that resources have better or worse performance characteristics over time. Garfinkel [12] already identified such differences. This is why it is important to know what one can expect from a provider during a longer period of time.

There are many metrics to monitor, in this research we only select a few key metrics that are able to give a global overview of the quality of the cloud service as a total. We start with a top down approach to narrow down the selection in metrics and to give more insight in what categories to select the metrics from. In [8] the authors distinguish three performance categories: 'Server performance', 'LAN latency and bandwidth' and 'WAN latency and bandwidth'. Server performance can be divided in four distinct subsystems: 'Memory', 'Network', 'Disk' and 'CPU' [4].

There are many performance related metrics to distinguish, measurement of these can be done using third party tools or by querying the raw kernel statistics. Third party tools such as iostat [1] provide ready to use metrics. For monitoring performance the counters provided by 'proc' have several advantages. The first is flexibility; by having the raw data it is possible to query this in anyway, even when the experiment is over. Secondly, the raw data provides many different counters which are not provided by any third party tools such as iostat. Laslty, if there are any bugs in iostat, the results are not accurate anymore. A good example is the 'svctm' metric [2] since version 9.1.3 [3] this metric is not reliable anymore.

The following metrics found on the virtual file system '/proc/' are monitored during this research: '/proc/stat' and '/proc/diskstats'. In Table 2.1 and 2.2 a full description can be found of those two virtual files. For completeness the subsystems network and memory are described in Table 2.3 and 2.4. Those two virtual files are monitored but not used during the experiments.

2.3.1 Uptime and connection performance

To measure availability the metric 'uptime' is used because it is a very accepted and understandable metric. It is used in SLA's (Service Level Agreements) to give a guarantee to the client of the availability of a service and is very straightforward to measure [18]. Connection performance is a general term used in this thesis to distinct the following metrics: connection time, download time and resolve time. The sum of these counters is the total time needed to load a page from the Internet and present it to the client.

[1] http://sebastien.godard.pagesperso-orange.fr/

[2] svctm, The average time (in milliseconds) for I/O requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them

[3] http://sebastien.godard.pagesperso-orange.fr/changelog.html

Table 2.1: '/proc/stat' - Explanation of cpu(x) fields found in '/proc/stat'

Field	Metric	Description
1	# user	Normal processes executing in user mode.
2	# nice	Niced processes executing in user mode.
3	# system	Processes executing in kernel mode.
4	# idle	Twiddling thumbs.
5	# iowait	Waiting for I/O to complete.
6	# irq	Servicing interrupts.
7	# softirq	Softirq: servicing softirqs.

2.3.2 CPU Metrics

The CPU metrics are available in '/proc/stat' which contains all CPU metrics of importance. In this file the field starting with 'CPU' is monitored. A description of i fields can be found in table 2.1. During the monitoring phase IOWait metric is selected to monitor from the '/proc/stat'. IOWait is a counter which indicates how long the CPU needs to wait on the I/O subsystem, because each processor core has his own IOWait counter it is possible that the IOWait is above 100%. The hypervisor server of WireITup has 48 cores which means that the IOWait could be maximum 4800%. In this research the IOWait of all cloud servers will be monitored to find out if this metric can be used as tool to find out if the I/O subsystem is not over utilized. The other CPU counters monitored during the execution of the experiments were of no interest. They did not reveal anything worth to show and because of this they where not presented in the results chapter.

2.3.3 I/O Metrics

All metrics about block devices are available in '/proc/diskstats' which is a list of all statistics from the block devices listed in '/sys/block/block device name/stat'. A

Table 2.2: '/proc/diskstats' - Explanation of the block device fields found in '/proc/diskstats'

Field	Metric	Description
1	# of reads completed	This is the total number of reads completed successfully.
2	# of reads merged	Reads which are adjacent to each other may be merged for efficiency.
3	# of sectors read	This is the total number of sectors read successfully.
4	# of milliseconds spent reading	This is the total number of milliseconds spent by all reads (as measured from ___make_request() to end_that_request_last()).
5	# of writes completed	This is the total number of writes completed successfully.
6	# of writes merged	writes which are adjacent to each other may be merged for efficiency.
7	# of sectors written	This is the total number of sectors written successfully.
8	# of milliseconds spent writing	This is the total number of milliseconds spent by all writes (as measured from ___make_request() to end_that_request_last()).
9	# of I/Os currently in progress	The only field that should go to zero. Incremented as requests are given to appropriate struct request_queue and decremented as they finish.
10	# of milliseconds spent doing I/Os	This field increases so long as field 9 is nonzero.
11	weighted # of milliseconds spent doing I/Os	This field is incremented at each I/O start, I/O completion, I/O merge, or read of these stats by the number of I/Os in progress (field 9) times the number of milliseconds spent doing I/O since the last update of this field. This can provide an easy measure of both I/O completion time and the backlog that may be accumulating.

reference to all fields in the '/proc/diskstats' file is described in table 2.2. The following metrics located in '/proc/diskstats' are used: TimeSpendIO, ReadsCompleted and WritesCompleted. The TimeSpendIO is a counter which keeps track on how many milliseconds the I/O subsystem has spend on I/O handling during a 1000 millisecond interval. ReadsCompleted and WritesCompleted are counters which count the number of read and write instructions. This is done for all cloud servers and for the hypervisor server which contains the cloud server of WireITup. This opportunity to monitor both cloud server and hypervisor could give interesting conclusions and insights which are under normal circumstances not available.

In this research the different metrics found in table 2.2 are monitored. This is done for all cloud servers but also on the hypervisor which contains the baseline cloud server of WireITup. This opportunity to monitor both cloud server and hypervisor could give interesting conclusions and insights which, under normal circumstances, would not be available.

Table 2.3: '/proc/meminfo' - Explanation of the memory fields found in '/proc/meminfo'

Metric	Description
# kB MemTotal	Total usable ram (i.e. physical ram minus a few reserved bits and the kernel binary code).
# kB MemFree	Is sum of LowFree+HighFree (overall stat).
# kB Buffers	Relatively temporary storage for raw disk blocks shouldn't get tremendously large (20MB or so).
# kB Cached	In-memory cache for files read from the disk (the pagecache). Doesn't include SwapCached.
# kB SwapCached	Memory that once was swapped out, is swapped back in but still also is in the swapfile.
# kB Active	Memory that has been used more recently and usually not reclaimed unless absolutely necessary.
# kB Inactive	Memory which has been less recently used. It is more eligible to be reclaimed for other purposes.
# kB HighTotal	Is the total amount of memory in the high region. Highmem is all memory above (approx) 860MB of physical RAM.
# kB HighFree	Highmem is all memory above 860MB of physical memory Highmem areas are for use by userspace programs, or for the pagecache.
# kB LowTotal	Lowmem is memory which can be used for everything that highmem can be used for, but it is also available for the kernel's use for its own data structures.
# kB LowFree	Bad things happen when you're out of lowmem.
# kB SwapTotal	Total amount of swap space available.
# kB SwapFree	Memory which has been evicted from RAM, and is temporarily on the disk.
# kB Dirty	Memory which is waiting to get written back to the disk.
# kB Writeback	Memory which is actively being written back to the disk.
# kB AnonPages	Non-file backed pages mapped into userspace page tables.
# kB Mapped	Files which have been maped, such as libraries.
# kB Slab	In-kernel data structures cache.
# kB PageTables	Amount of memory dedicated to the lowest level of page tables.
# kB NFS Unstable	NFS pages sent to the server, but not yet committed to stable storage.
# kB Bounce	Memory used for block device "bounce buffers".
# kB CommitLimit	Based on the overcommit ratio ('vm.overcommit_ratio'), this is the total amount of memory currently available to be allocated on the system.
# kB Committed_AS	The amount of memory presently allocated on the system.
# kB VmallocTotal	Total size of vmalloc memory area.
# kB VmallocUsed	Amount of vmalloc area which is used.
# kB VmallocChunk	Largest contiguous block of vmalloc area which is free.

Table 2.4: '/proc/net/dev' - Explanation of the network device fields found in '/proc/net/dev'

Field	Metric	Description
1	# Received bytes	The total number of bytes of data received by the interface.
2	# Received packets	The total number of packets of data received by the interface.
3	# Received errs	The total number of receive errors detected by the device driver.
4	# Received drop	The total number of packets dropped by the device driver.
5	# Received fifo	The number of FIFO buffer errors on this interface.
6	# Received frame	The number of packet framing errors.
7	# Received compressed	The number of compressed packets transmitted or received by the device driver.
8	# Received multicast	The number of multicast frames received by the device driver.
9	# Transmitted bytes	The total number of bytes of data transmitted by the interface.
10	# Transmitted packets	The total number of packets of data transmitted by the interface.
11	# Transmitted errs	The total number of transmit errors detected by the device driver.
12	# Transmitted drop	The total number of packets dropped by the device driver
13	# Transmitted fifo	The number of FIFO buffer errors on this interface.
14	# Transmitted colls	The number of collisions detected on the interface.

Chapter 3

Experiment tools and setup

When using cloud providers it is of great importance to gain valuable knowledge about availability and performance characteristics of a particular cloud provider. Cloud providers are using virtualization and hypervisor technology to host multiple virtual servers on one physical server. One of the ideas behind virtualization is to share resources with other virtual servers. This is possible because it is known that physical servers do not utilize all resources. With virtualization, resources are shared and as a result hardware is utilized more efficiently. As discussed earlier both memory and CPU are less sensitive and easily distributed to other places in the cloud where there is more capacity. This does not apply for I/O because disk images are far more difficult to transfer due to their size.

Performance problems with the I/O subsystem can occur because of heavy utilization. The difference between storage resources and other shared resources is that heavy utilization can be solved by 'live migration' [14] technology. This is moving an instance without noticeable downtime from one server to another with more resources available. Such a technology for moving disk images between different I/O subsystems is currently not supported by all common hypervisors [14].

In this section the six selected cloud providers are discussed in ' Cloud Providers'. The six providers have different characteristics, datacenter locations and are using different technologies. The first part of this research is focused on the availability of the selected cloud providers. This is done by using a pervasive monitoring system to research cloud behavior and availability in all parts of the world. This system is provided by Watchmouse and is explained in more detail in sub section 'Watchmouse'. Secondly, a more challenging experiment is defined by using a modified load generating tool which makes it possible to simplify endurance testing. This enables us to test the quality of I/O and the influence on server performance, uptime and connection performance. The goal is to test the stability and quality of the I/O infrastructure of the selected cloud providers. Several tools need to be constructed for monitoring and visualization. In sub section 'Monitoring low level kernel metrics' a more detailed explanation is given of the tools constructed to gather the selected kernel metrics. For load generation a tool is modified which is explained in sub section 'Generating I/O Load'. The usage of these tools makes it possible to perform the second experiment. Lastly we select one of the providers and adapt current environment to work with a selected cloud provider. For this last experiment a traffic distribution system is selected, such systems are discussed in 'DNS infrastructure providers', and a cloud provider is selected and integrated into an existing IT environment which is described in sub section 'Enterprise environment'.

3.1 Measurement Tools

During the different experiments different CPU and I/O related metrics are monitored. The goal of this research is to investigate key performance metrics. This is done for all cloud servers and for the hypervisor server which contains the cloud server

of WireITup. This opportunity to monitor both cloud server and hypervisor could give interesting conclusions and insights which are under normal circumstances not available.

3.1.1 Watchmouse

For all experiments a SLA monitoring tool called Watchmouse [1] is used which makes it possible to check all cloud providers from 64 different locations around the world. During this experiment 57 of the 64 locations where available, the other monitoring stations were in maintenance. The locations are shown on a world map in Figure 3.1. Every 10 minutes, a monitoring site is selected randomly and performs a check which logs the date, time, station and connection time.

When a monitoring node reports an error, Watchmouse will use a second node to verify this by immediately performing the same action. Only a verified defect will count as an error, and will be counted as downtime. A error which could not be verified by another monitoring station is called a defect. The reason for doing this is to make a distinction between a minor connectivity outage and global/service outage. One of the main issues involving this kind of experiments is the lack of bandwidth or server capacity at the monitoring site. With this in mind a commercial application is selected which has a good reputation concerning SLA monitoring. It is very important that the results are flawless, which is also concluded in [22], to prevent incorrect results. This is the reason for selecting a third party application which is able to provide such a complex infrastructure and the experience with monitoring.

[1] www.watchmouse.com

Figure 3.1: Locations of all 64 monitoring nodes used to monitor connection time and uptime. They are spread over 6 continents, 40 countries and 61 cities.

3.1.2 Monitoring low level kernel metrics

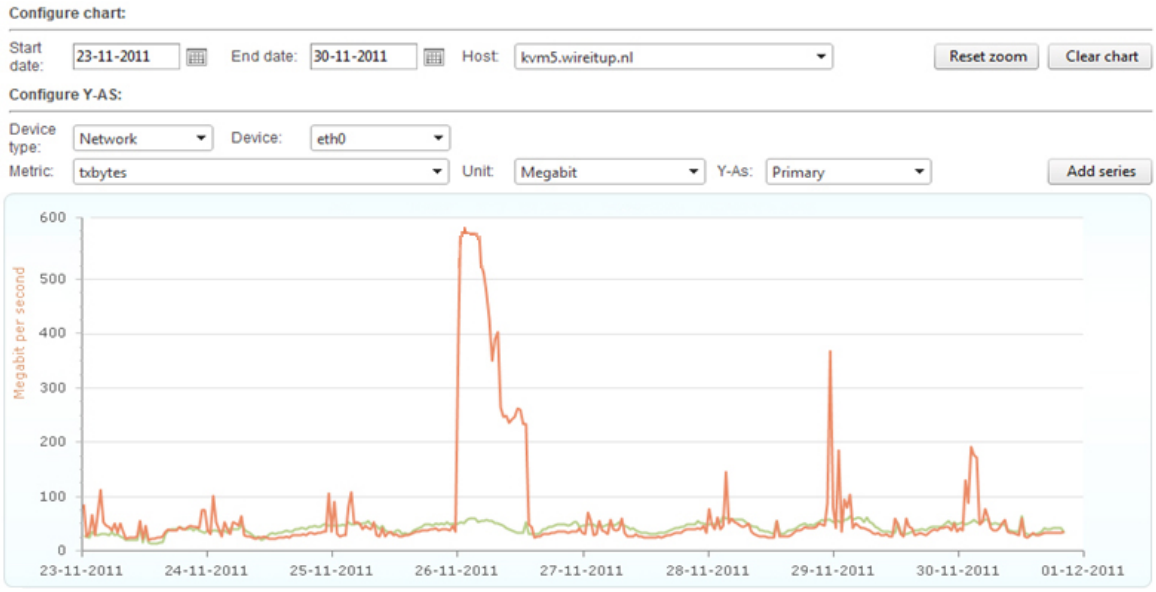
During the endurance experiments where the I/O performance is measured a custom made application is used to gather multiple performance related server metrics. This custom build monitoring system gathers a set of pre-defined metrics every five minutes. Every cloud server contains a daemon which sends the gathered information to a server which stores it in a database. Afterwards the gathered metrics of all cloud servers can be compared with each other by using a custom build front-end which is able to build customized graphs, in Figure 3.2 a screenshot of the user interface. The selected metrics found in this interface are discussed in paragraph "Metrics".

3.1.3 Generating I/O Load

To generate a large amount of I/O operations an open-source application called Sio_ntap developed by Networking Appliance[2] is used. Sio_ntap is a benchmark application which is able to generate a maximum load to determine the performance

[2]http://www.netapp.com/

Figure 3.2: User interface - Screenshot of the user interface build to create customized graphs. Select metrics from multiple servers and has the ability to add those metrics to the right or left axis.



of the I/O subsystem. To make Sio_ntap compliant with the requirements of this experiment there are two new features added. The first feature is called 'burst' and configures the number of I/O operations which need to be done by each thread sequentially. The second feature is a wait time between two independent I/O burst operation(s) and is called 'wait'.

By adding those two new parameters to sio_ntap, it is possible to adjust the delay time between one or multiple I/O requests (burst size). The application is configured to run every minute using the crontab and configured to a maximum amount of I/O per second. After this minute the application is forced to stop even when it did not completed his workload. A new instance is started which runs again one minute. Experiments showed that running this application for a longer period of time could result in a crash and by using this technique it is possible to prevent this.

When the application is stopped and it has not processed all workload the average

generated I/O operations will be lower. The variances of the number of completed I/O operations can be used to determine health and latency of the I/O subsystem. Besides Sio_ntap there are also other OS specific processes running in the background which could cause some overhead which results in a higher average I/O. Because the server uses a shared I/O subsystem the performance can degrade when other instances are also active.

3.2 IaaS Providers

During the experiments different cloud providers are used, in this section we describe how they are selected. During the last experiment we also need to select a IaaS provider which is able to provide traffic routing.

3.2.1 Cloud Providers

For the performed experiments six IaaS providers are selected namely; Amazon, GoGrid, Rackspace, Joyent, Leaseweb and WireITup. Five providers are selected to measure performance and availability related metrics. Important, to note here, is that the selected providers use different hypervisors. From the list of cloud providers for IaaS and web hosting mentioned in the Gartner paper [13], I have selected 4 providers. Because all the providers which are selected from the Gartner paper are located in North America, we have decided, for completeness, to add two other providers located in Europe.

Amazon EC2 is recognized by Gartner as the most visionary and one of the leaders in cloud Infrastructure services. Amazon has many customers and is discussed widely. Rackspace is an independent web hoster and a major player in managed hosting services. Joyent is a much smaller provider but has a remarkable track record of

clients such as Dell, Twitter and LinkedIn. Another reason to select Joyent is because of its technology which is a self-maintained version of Sun Solaris and using KVM as their hypervisor. GoGrid is also a smaller provider which has an intuitive user interface. Lastly, Leaseweb is a major Dutch hosting company. The main reasons for selecting Leaseweb are the price, data center location and the used hypervisor (VMWare).

As an addition to the five cloud providers, WireITup is added as a baseline. WireItUp is not a cloud provider but uses all common technologies such as a hypervisor. By adding WireITup as a provider to this experiment I also adding the ability to monitor the hypervisor server which is not possible with the other providers. During this experiments the virtual server provided by WireITup is also referenced as one of the cloud providers.

3.2.1.1 Features and specifications

[TODO komt dit er wel of niet in] One of the observations which can be made from table 3.1 and from Gartner's article is that the two leading companies on web- and cloud hosting use XEN as their major hypervisor. Another widely used hyperviser among different players is VMware. These two hypervisors are most commonly used under IaaS providers. The second observation is the location of the data centers, most providers have several locations in North America and only one or two in Europe or Asia.

Because of the rapid developments in technology, hard- and software it is almost impossible to give a complete overview of current features, pricing and specifications. Therefore more attention is given to what has changed in the last year. By researching the news archives of the selected providers the following big changes have been made. Joyent changed their hypervisor in September 2011 and are now using KVM. They

Table 3.1: Features and specifications of selected cloud providers

Provider	Cpu	Mem	Disk	Hyperv.
Amazon	2	613MB	20GB	XEN
Joyent	4	256MB	10GB	KVM
Leaseweb	1	512MB	40GB	VMWare
GoGrid	1	512MB	20GB	XEN
Rackspace	1	256MB	10GB	XEN
WireITup	1	1GB	100GB	KVM

also changed their billing model in August 2011 and are now able to provide a pay per hour pricing model. GoGrid opened a new datacentre location in Amsterdam, the Netherlands. Rackspace and Leaseweb did not had any recent changes to their features, pricing and specifications. Amazon really has an amazing list of improvements and added several new services to their product portfolio. As Gartner already identified they are really a pioneer on cloud computing, even competitors such as GoGrid recognize this:

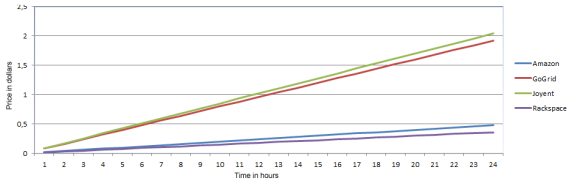
> *"Amazon will shine brighter than ever as the beacon of the future of computing. Thank goodness for Amazon. They are defining a new future and giving it credibility... "* - John Keagy (Chairman and Founder GoGrid)

3.2.1.2 Pricing

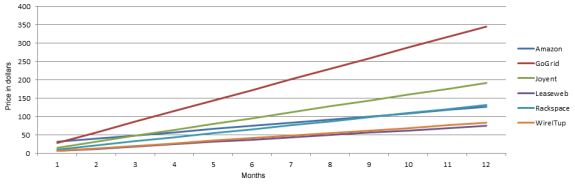
The used cloud definition in this article defines a "pay for what you use" pricing model. But it is unclear in which time frame a resource must be consumed and billed. The lack of clear definition and standardization results directly in a variety of billing models which is also the case for our selected IaaS providers. A good example is GoGrid which [TODO is it who's or not] model is based on the memory usage per hour. Therefore the different pricing models of the providers will be discussed briefly. All selected providers have the possibility to scale up and down within a certain time period. This makes them all compliant with the used standard definition. The

Figure 3.3: Show price of cloud provider per hour and month - Show price in dollars of the six selected cloud providers. The left figure shows th price per hour of the following providers Amazon, Joyent, GoGrid and Rackspace because WireITup and Leaseweb do not provide such a billing model. At the right side price per month which contains all six providers.

(a) Price per hour



(b) Price per month



providers can be segmented in two sets. The first set is Amazon, Joyent, GoGrid and Rackspace provide a pay for what you use model by the hour or month3. The second set is Leaseweb and WireITup, they only provide a billing model by the month 3.3because of the rapid developments the price is changes constantly and can be related to the location of the datacenter.

An important part of the definition of cloud related services defined by [23, 25] is to pay for what you use. Leaseweb does not support this way of billing and Joyent only started to supports this billing model on the 15th of September 2011. Another cloud criteria is the quick deployment of new resources. Leaseweb do not support instant creation of new cloud servers. The last criteria is (infinite) scalability where computer servers can scale to fulfill growing demands. Only Amazon EC2 can both scale cores and memory with minimal downtime. Others such as Rackspace, Gogrid and Joyent can only scale in memory and Leaseweb cannot scale instantaneously.

3.2.2 DNS infrastructure providers

For geographical traffic distribution and onsite loadbalancing there is a need to select a provider which is able to provide a solution which is fast and reliable. Technology

such as anycast and loadbalancing can provide such a solution. There are multiple vendors which combine this and offer this as a service. In this section the providers are selected, discussed and one is selected. Lastly, the results are disucussed from before and after the implementation to research the benefits and impact of such an solution.

In figure 3.2 all specifications of importance are summarized for the select providers. All providers except the current situation have a point of presence in multiple continents and have nameservers distributed around the world. The technology is based on anycast technology which makes it possible to route clients to nearest location. For example a client in Australia can be redirected to a server located in Australia. In the old situation the client is always directed to the servers located in Europe.

UltraDNS is targeted at the Enterprise market and is asking a premium for their services. It offers a loadbalancer solution which is able to redistribute load depending on server performance using connection time as a metric and redistribute traffic when connection time is getting higher. DNS Made Easy is a provider which is targeted at the lower segment and does not include a more advanced loadbalancer but only offers round robin loadbalancing. lastly, DynDNS offers a slightly less advanced loadbalancer then UltraDNS because it can only determine if the server is up or down but has more competitive prices for queries and services.

During the inventarisation phase all providers are contacted and the possibilities are discussed. The support and sales staff of UltraDNS and DynDNS were fast and their support were of good quality. They both offered different presentations and conference calls to explain there product thoroughly. If there was any question they replied the same day with accurate and detailed answers. The support of DNS made easy is less extensive but their prices are very competitive compared to DynDNS and UltraDNS.

Figure 3.4: Anycast routing - Routing client traffic to the nearest data center location all over the world.

Table 3.2: Features and pricing comparison between current and DNS infrastructure providers - This table contains all features and pricing information in dollars per month from current and DNS infrastructure providers.

Description	Current	UltraDNS	DynDNS	DNS Made Easy
Technology	Unicast	Anycast	Anycast	Anycast
Nameservers	3	15	17	12
Asia	no	yes	yes	yes
North America	no	yes	yes	yes
South America	no	no	no	no
Europe	yes	yes	yes	yes
Australia	no	yes	yes	no
Africa	no	yes	no	no
Fail over	no	yes	yes	yes
Load balancing	no	yes	yes	no
GLSB	no	yes	yes	yes
Queries (100Million)	$ 0	$ 1100	$ 495	$ 150
Failover service	no	$ 125	$ 100	free
Load balancing	no	$ 150	$ 200	no
Geo-based DNS	no	$ 250	$ 200	$ 45
Additional domains	$ 0	unknown	$ 0,50	$ 0,13

3.2.3 DNS based Loadbalacing

To understand the working of DNS based load balancing it is important to know how name resolving in functions. When a client visits a website, the web browser needs to resolve that domain name to an IP address before it can load the page. When his web browser makes a request for that DNS record, it first checks the operating system resolver. If the IP for the domain name isn't cached there, the OS queries a nameserver for the DNS record, which then gets passed to the browser. When the TTL has passed, the operating system purges that resolved domain from the cache and subsequent requests to that domain would cause a new query to your nameserver. DNS based loadbalancing finds place at the nameserver level and its working relies heavily on when the client performs a lookup again. Every time when a lookup is performed an IP number is returned to the client just as a normal resolve would have done. The main difference is that t  with every lookup an pool of IP numbers are used which are evenly distributed among the requests. When a server is failing, the IP number of that server will be removed and at regular intervals the server is checked and added when it functions again. Because browsers such as Internet Explorer are using a very long interval time to perform a lookup again, see Table 3.3, it could take some time before existing clients are redistributed to a working server while new clients are immediately redistributed to a working servers because they do not receive the IP number of malfunctioning server.

When the browser cached the DNS query then it depends on the used browsers what happens next. For example Internet Explorer will resolve a name after 30 minutes and Opera is caching (seemingly) indefinitely until you restart the browser. Most browsers completely discards TTL, you may be wondering why web browsers would even cache DNS entries if the operating system already does this. Supposedly, this happens to reduce DNS server load and speed up response time, although it is difficult to believe

Table 3.3: TTL of IE and Chrome - In this table the default TTL in seconds for Internet Explorer and Chrome are determined using Wireshark.

Description	Internet explorer	Chrome
First startup	Always	Always
Second startup	between 30 and 60 seconds	90 seconds
New tab page	between 30 and 60 seconds	90 seconds
New browser	between 30 and 60 seconds	90 seconds
Reload page	between 30 and 60 seconds	90 seconds
Forced Reload	between 30 and 60 seconds	90 seconds
TTL	30 minutes	90 seconds

that a hit on your OS DNS cache is all that expensive of an operation.

In table 3.3 the basic actions a client can perform on their browser is summarized. Results are gathered using a packet sniffer called WireShark [3]. When testing with Internet Explorer in several occasions it showed different timeouts before doing a lookup. Chrome at the other hand was very consistent and always performs a lookup after 90 seconds.

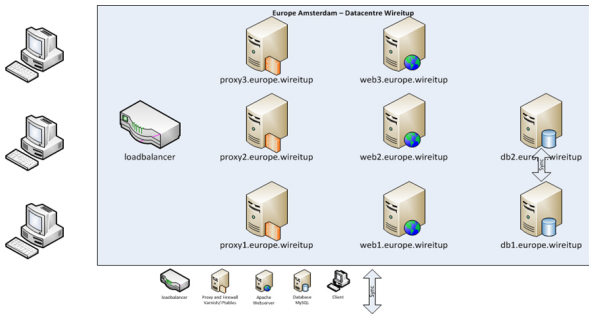
3.3 Enterprise environment

In the last two experiments two IaaS providers are integrated to extend an existing IT environment. Before starting it is important to know how the current IT environment is configured before migrating several infrastructure components to the cloud. In Figure 3.5a the current infrastructure architecture is given. All traffic is routed to the Barracuda loadbalancer located at the front of the caching servers. The loadbalancer is running at high availability mode which means that if one of the loadbalancers is failing the other is resuming his task. Traffic is directed at one of the Varnish HTTP caching servers based on round robin. Every minute the loadbalancer checks if the caching server is functioning properly if not the server is removed from the pool until

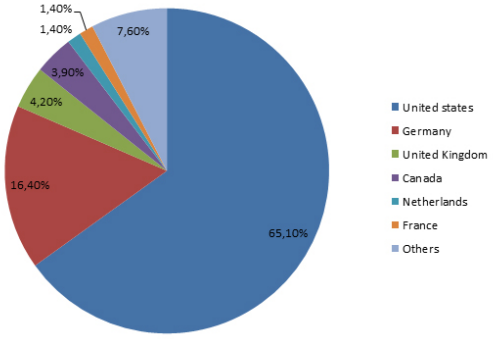
[3] www.wireshark.com

Figure 3.5: Overview of current IT environement - Showing geographical distribution of clients at the right side and at the left side a simplified presentation of the current IT environment.

(a) IT architecture



(b) Geographical distribution of clients



the server is working again. If the request is not in the cache it is redirected to the web server. The two database servers are running at a master slave configuration. Only when the master fails the slave server becomes master until this is manually corrected.

The application running on this configuration is used 24 hours per day with an average of over 2500 active visitors and at peak times more than 5000. The site has over 150,000 members, and receiving an average of 14+ Million pageviews per month. It is within the top 3.000 of biggest sites within the US according to Alexa site index [4]. This figures gives a good impression about the size of the site and the importance of providing a fast and reliably environment because downtime has a big impact on revenue.

In Figure 3.5b the geographical distribution of clients is presented. Most clients (65,1%) are located in the United States and 23,4% of the clients are from European country's. 7,6% of the client's are from country's all over the world.

[4] www.alexa.com

Chapter 4

Measuring the performance and availability of cloud infrastructures

During the past five years there has been an increase of implementing cloud services in new and existing projects. Armbrust from UC Berkeley describes the elasticity of cloud resources without paying a premium for large scale, as unprecedented in the history of IT [20]. Recent publications report many successful implementations of cloud computing services done by enterprises and governments [27, 28, 3].

In 2010, Gartner predictions showed that cloud computing is not just a hype [13]. Companies started and will continue to integrate their businesses using cloud services to become more dependent of the ever-increasing complex and costly infrastructure needed to support their businesses. However, when cloud services fail this new dependency can result in downtime for mission-critical applications, or even permanent loss of data. The examples reported in [7, 20] show clear evidence of a lack of a mature service level management and delivery.

In such infrastructures the resources between instances are real timed scheduled and optimized, therefore service availability is difficult to achieve when instances need a

huge amount of additional resources. It is likely that performance degradation will occur and disrupt services because of resource sharing between competing instances. The tradeoff between guaranteed resources or economical efficiency and elasticity remains a serious challenge for cloud users and cloud providers.

The primary focus of this section is on the importance of the non-functional requirements; performance and availability. These key requirements are important for cloud providers and cloud users to deliver their services as intended to their clients.

While the availability can be monitored by two simple metrics, namely the uptime and connection time, the monitoring of the performance requires a more elaborated experiment. One approach is to determine how the providers perform under a continuous load. This technique is better known as 'endurance testing', and provides a way to test a service without risk and gives an opportunity to investigate limitations and bottlenecks. Endurance testing aims at performing multiple experiments with different load characteristics with the goal to assess the quality of a given cloud provider over time.

Accurate measurements can be achieved by monitoring low level metrics supplied by the Linux kernel. However, in the case of cloud computing the hypervisor plays a major role and thus needs to be carefully monitored when studying the performance of cloud providers. This makes it possible to gain more insight information about the resource consumption and distribution on the hypervisor level. To achieve this goal a number of experiments are performed which aim at testing the I/O subsystem. Cloud Computing uses virtualization technology to provide resource sharing. This is possible because it is known that physical servers do not utilize all resources. With virtualization, resources are shared and as a result hardware is utilized more efficiently. The difference between storage resources and other shared resources is that heavy utilization can be solved by using 'live migration' [14], which consists of moving an

instance without noticeable downtime from one server to another with more resources available. Such a technology for moving disk images between different I/O subsystems is currently not supported by all common hypervisors [14].

4.0.1 Availability (Uptime)

This experiment is used to determine the availability of the selected cloud providers. The six cloud providers are monitored from one of the 57 locations world-wide. Every 10 minutes, a monitoring site is selected randomly. When a monitoring station could not make a connection with one of the cloud providers, another monitoring station immediately retries the same action. When this retry is successful, this is counted as one defect. In case this retry also fails it is counted as one error.

After a month of testing a total of 28263 tests were collected, 17 defects and only two errors where counted which means a total error rate (errors and defects are included) of 0.00074%, Table 4.1. Twelve defects where reported by monitoring stations in Europe when testing the American providers. Both errors were reported for Leaseweb, other providers did not have any outages during this period. The providers with only one defect were Amazon and WireITup. For Gogrid, Leaseweb and Rackspace a total of three defects was counted and for Joyent a total of six defects was counted.

The worst day was October 1st where a total of seven defects and one error were recorded and presented in Table 4.2. Six of the defects were measured at the monitoring station located in Madrid on October 1st. The defect was reported four times for Joyent and two times for GoGrid. Rackspace and Amazon were not affected, which rules out the possibility that the monitoring station was defect. Further investigation revealed that both Cloud Providers have Level 3 as their transit provider. It is reasonable to assume that this local connectivity problem did occur at least between 12PM and 9PM. Other defects were more evenly distributed among provider and time of

Table 5.1: Defects and Errors IaaS providers - In this table all errors and defects detected by monitoring stations from Watchmous are summerized which are detected during the availability experiment.

Datetime	Provider	Location	Type
09-26 13:08	Joyent	Athens	defect
09-26 13:40	Joyent	Amsterdam1	defect
09-30 11:58	Amazon	Vancouver	defect
10-01 12:16	Joyent	Madrid	defect
10-01 15:18	Joyent	Madrid	defect
10-01 16:17	GoGrid	Madrid	defect
10-01 16:55	Leaseweb	Shanghai	error
10-01 16:55	Leaseweb	Austin	error
10-01 17:15	Leaseweb	Austin	defect
10-01 18:29	GoGrid	Madrid	defect
10-01 19:08	Joyent	Madrid	defect
10-01 20:49	Joyent	Madrid	defect
10-04 10:22	Rackspace	Moscow	defect
10-05 05:15	Leaseweb	London	defect
10-06 17:15	Leaseweb	Kharkov	defect
10-11 08:03	GoGrid	Cologne	defect
10-11 20:04	Rackspace	Stockholm	defect
10-11 20:59	WireITup	Stockholm	defect
10-20 10:47	Rackspace	Amsterdam	defect
10-27 06:13	Leaseweb	Sofia	error
10-27 06:13	Leaseweb	Munchen	error

occurrence.

The number of defects compared to the number of errors shows the importance of the connection in service availability. The incident on October 1st, where GoGrid and Joyent had connection problems with a monitoring station in Madrid, proves this furthermore. When operating at global scale it is important to know that the service does not suffer from local connection problems.

Figure 4.1: Europe and America - Average connection time in milliseconds gathered from the monitoring stations located in Europe and America for the six selected cloud providers.

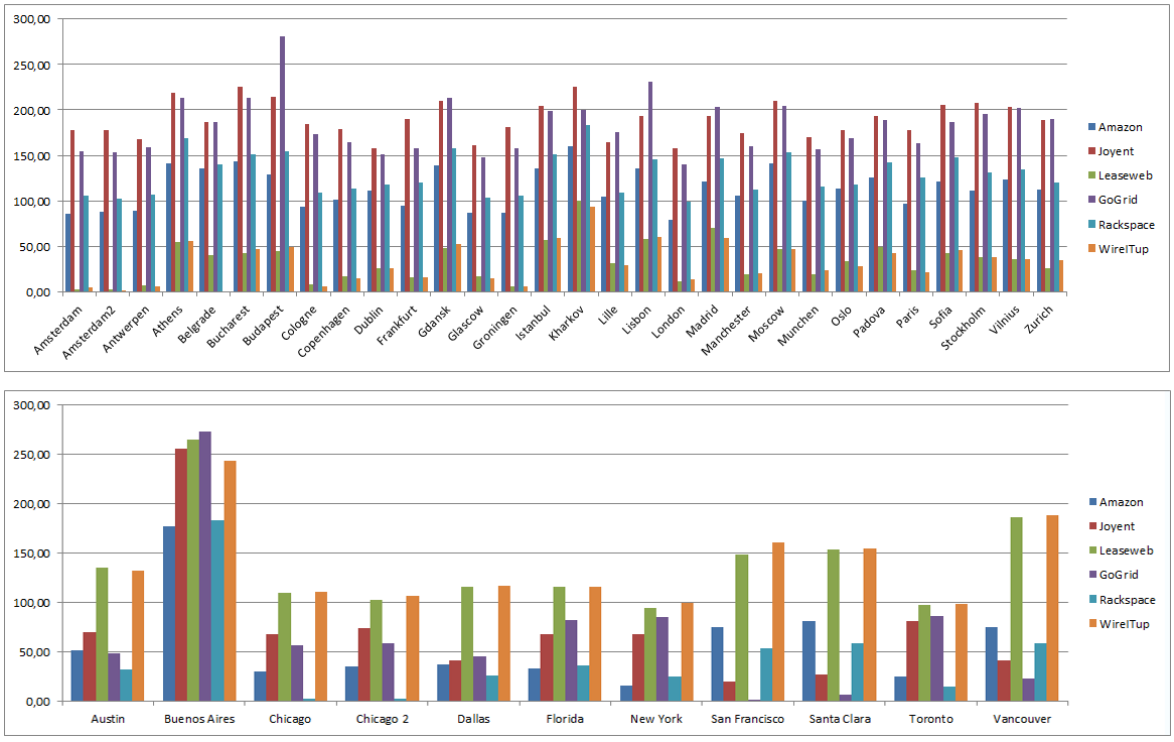


Table 4.2: Madrid monitoring station 1 october - All checks performed by the monitoring station located in Madrid, Spain from Watchmous are summerized in this table.

Time	Provider	Type
00:39	GoGrid	OK
07:09	Amazon	OK
07:46	Leaseweb	OK
08:06	Leaseweb	OK
11:54	Leaseweb	OK
12:16	Joyent	Packets lost
14:29	Amazon	OK
14:39	Amazon	OK
15:18	Joyent	Packets lost
16:17	GoGrid	Packets lost
18:29	GoGrid	Packets lost
19:08	Joyent	Packets lost
19:45	Rackspace	OK
20:49	Joyent	Packets lost

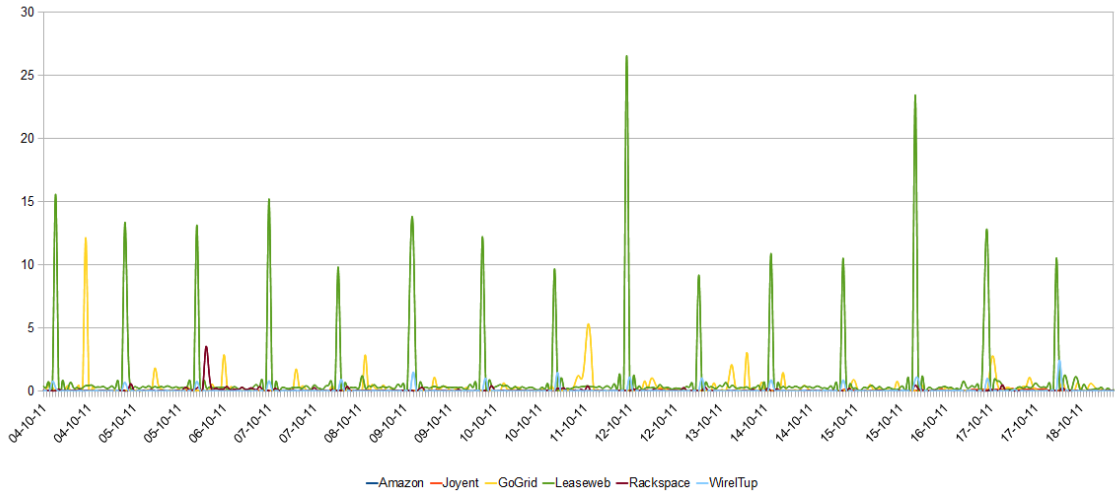
4.0.2 Connection time

The connection time measured for each of the six providers is shown in Figure 4.1. In this figure the average connection time in milliseconds are presented for the European and American monitoring stations.

As expected the providers located in Europe have a better connection time in their own continent than the cloud providers in North America and vice versa. The European providers had an average latency of 82ms to North American monitoring stations and the North American providers had an average latency of 121ms to the European monitoring stations. The connection performance tests also shows that the providers connection performance is near optimal, when a provider is located in the same city, country or state as the monitoring station. An example is GoGrid which has only a latency of a few milliseconds to Chicago.

To deliver a fast and stable service it is important to have a cloud provider close to

Figure 4.2: Idle load - In this experiment there is no load generated. The IOWait metric shows that the processor for some providers needs to wait relatively long for I/O although the server is running idle.



their clients. In some scenario's this is not possible because clients are spread around the world, in those cases cloud providers dispose of multiple datacenter locations and solutions to route the traffic to the nearest location.

4.0.3 I/O performance

This Section describes the endurance tests performed with the selected cloud providers over a period of two to four weeks. One set of tests consisted in monitoring the servers in idle mode, and a second set of tests when the servers are subjected to a workload generating 20, 100 and 500 I/O operations.

4.0.4 Experiment without load

For a period of two weeks the cloud servers where monitored and results are presented in Figure 4.2. It is very interesting to observe that the IOWait metric is far from idle at regular intervals. The most noticeable peaks are recorded for GoGrid and

Leaseweb, a closer examination shows that both Rackspace and WireITup also have a small raise at a regular interval.

The resource consumption of the hypervisor is shown in Figure 4.4. In Figure 4.4a the ReadsCompleted, WritesCompleted, TimeSpentIO and IOWait are shown for the hypervisor server when the WireITup cloud server is running idle. In the graph is shown that when the time spent on I/O operations (TimeSpentIO) approaches 1000ms the IOWait at the WireITup cloud server located on the hypervisor, also increases because processes need to wait longer before an I/O request is done. This not only causes stalling of processes, but also results in unnecessary usage of CPU resources.

Luckily the high IOWait occurs at regular interval (IOWait increases around 4AM each day) which makes it easier to investigate this phenomenon. The hypervisor of WireITup hosts roughly 60 instances and 20 of them are sharing the same I/O subsystem. Most of those instances are based on the Linux operating system. The increase in IOWait is the result of background Linux tasks, where the Linux operating system starts rotating the log, restarts processes and do other maintenance activities. This experiment pr s that, as expected, resource sharing has always some performance impact on the cloud server because other cloud instances on the same I/O subsystem are competing on the available resources. In case of I/O this can happen when the hypervisor server shows a utilization of 100% on the I/O subsystem and this directly reflects in degradation even if the server is doing nothing except running OS.

4.0.5 Experimenting with load

In the second endurance test the servers where subject to a continuous load of 20, 100 and 500 random write instructions. 20 I/O per second should not be an issue since a

SATA disk delivers at least 75 random write instructions. Thus the experiment with 20 instructions should not be considered as a real problem while 100 instructions could be causing utilization problems. Lastly, the 500 instructions experiment will definitely push a cloud server to his limits.

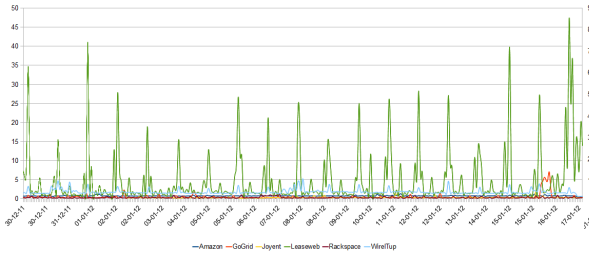
After a few hours the Leasweb server crashed during the 20 random write instructions test and could not be restarted until the support staff restored the server. The server was stable after this restart but had the highest average IOWait. Another interesting observation is that Amazon's WritesCompleted counter was very unstable and did not generate the expected 20 I/O instructions per second. A possible explanation for this behaviour is that the different I/O operations where merged together and thus not counted separately. To investigate this problem more throughly a debug version of Sio_ntap was developed which is able to keep a counter of success write operations separately from the kernel metrics. By comparing the kernel counter and the Sio_ntap counter I found out that Sio_ntap was generating the configured operations per second but that the kernel metric showed a lower average count.

Because the counter of the kernel increased when starting Sio_ntap the only conclusion could be that the proprietary kernel of Amazon is merging the writes together and count a merged write as one operation.

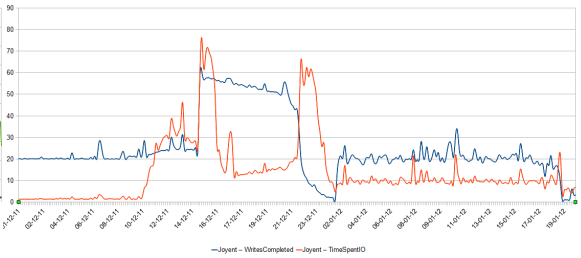
Besides Leaseweb all providers were stable and completed the test without any issues, but there was a noticeable increase in the IOWait for all providers compared to the idle experiment. For Joyent this increase was not severe, Amazon, Rackspace and WireITup performed very well during this test and only had a slight increase, but all within acceptable boundaries. In multiple occasions GoGrid had a high IOWait time for a relatively long period. The worst results were recorded for Leaseweb, not only the server was unstable during this test, Leaseweb also had the highest average IOWait.

Figure 4.3: Experiment 100 IOPS - The six selected cloud providers where stressed by generating 100 random write instructions per second. In this figure on the left the IOWait and right the average generated write instructions according to the /proc/diskstats and /proc/stat. At the right a visual presentation of the WriteCompleted and TimeSpentIO during the performance degradation of the Joyent server during this test.

(a) Experiment 100 IOPS



(b) Performance degradation at Joyent



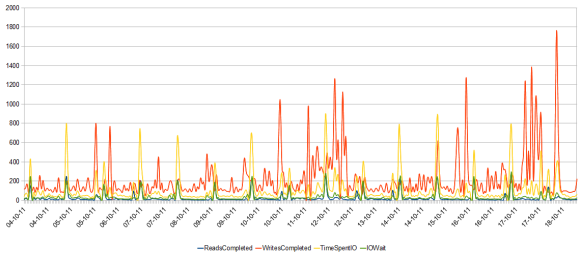
Where Joyent showed excellent performance during the last experiments, the experiment with a load of 100 I/O operations reveals that the cloud server can hardly reach the 100 I/O operations per second, see Figure 4.3b. This performance degradation occurred during Christmas when testing with a load of 50 I/O instructions. This figure visualizes how the performance degrades during Christmas until the server stopped responding.

After contacting the Joyent support team, they informed me that the cause of the faulty behaviour was a server migration from a poorly populated server pool to another one which has more instances. While the variance in performance is most likely due to multi-tenant effect and their I/O scheduler which discriminates between smaller and larger instance sizes.

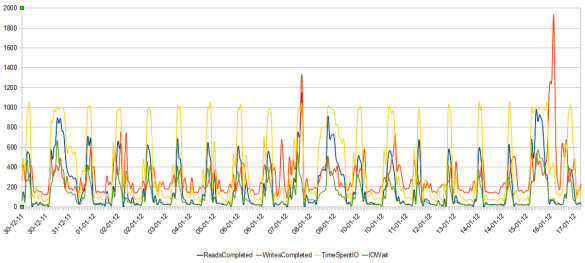
The other providers showed good performance in Figure 4.3a. Especially GoGrid which was able to generate an average of almost 103 I/O instructions. The servers from Rackspace and WireITup showed a very stable generated load of 96 instructions. Only Leaseweb showed some performance dips which occured at the same time as the increase of the IOWait metric. Comparing the TimeSpendIO from Joyent before and

Figure 4.4: Hypervisor statistics - The statistics of the hypervisor containing the WireITup cloud server. First graph shows the statistics when the cloud server is running IDLE and second when running 100 random write instructions per second. In the figures we present the average generated write-, read instructions, timecompleted, and the IOWait according to the /proc/diskstats and /proc/stat

(a) Without load on WireITup server



(b) With 100 I/O instructions on WireITup server



after the performance degradation we can see that the TimeSpendIO is raised from 1 to 10 milliseconds. A clear indicator that this server has problems with the I/O subsystems.

In the 500 I/O instructions experiment only three of the six providers where used. Gogrid showed excellent performance compared to the others and was able to generated an average of 461 I/O instructions. Rackspace generated an average of 443 I/O instructions. At the start of the experiment, Leaseweb had severe stability problems and could not generate an average of 320 I/O instructions, the IOWait and TimeSpentIO were also very high. When the experiment progressed the average generated I/O instructions became higher and the IOWait and TimeSpentIO lower. After consulting the helpdesk they informed me that instances running on the same I/O subsystem where consuming less I/O which resulted in more available resources. In this last experiment the IOWait and TimeSpentIO demonstrated again to be good indicators for detecting performance degradation.

Finally WireITup's hypervisor showed an increase in both IOWait and TimeSpendIO metrics comparing the idle experiment with the 100 I/O experiment Figure 4.4a and 4.4b. The IOWait raised to an average of 113,83%. When the I/O experiment was

stopped, the IOWait of the hypervisor server wend back to an average of 42%.

After starting the first experiment the server with the highest IOWait crashed within hours and before the 100 I/O experiment started, another server crashed during Christmas. Those two incidents showed that the identification of possible performance degradation by cloud providers remains a serious challenge. The results from the hypervisor reflects this conclusion: With only a relatively small increase in read and write instructions, the overall I/O utilization can be increased considerable.

San Francisco, California and Ashburn, Virginia and two different datacenters located in Amsterdam, the Netherlands.

By using a hybrid architecture the traditional IT service provisioning are combined with cloud infrastructure services. This approach makes it possible to take advantage of all what cloud services promises. This research starts with the selection and testing of an anycast based traffic director. This infrastructure provider makes it possible to manage and control your traffic. By setting up an experiment it is possible to gather performance statistics before and after the integration of this IaaS provider and compare them.

The addition of the cloud providers should add scalability and performance to the environment. Scalability should improve because cloud services are known for their scaling opportunities but the experiments done are focused on measuring the achieved performance increase. Because most of the clients are located in the United States, this research focuses more on the performance increase for those clients. To measure the resulted performance increase, three experiments were setup. The first experiment focuses on the increase in performance by adding a traffic management IaaS provider to the current setup. During that experiment the resolve performance is monitored for 12 weeks. The first six weeks consisted in monitoring the current situation and the last six weeks the new situati This is done by 57 monitoring stations distributed around the world. The second experiment is about measuring the performance increase by adding two Varnish caching servers located at San Francisco, California and Ashburn, Virginia. In this experiment the download performance is measured when a page is cached and download by one of the monitor servers. The second experiment, is about measuring the bandwidth performance by downloading a file of 1MB. The monitoring for those two experiments is done by 57 monitoring stations from different location each location is routed to the nearest location.

Results gathered during the three experiments are extremely valuable for key decision makers. It outlines statistics about performance improvement and deterioration when integrating two IaaS providers into an existing and complex enterprise environment. Although this research was focused on improving the experience on this environment for clients located in the United States it is easy to project the method used in the research for other continents or even countries. For IT architects who are looking for a technique or method to distribute traffic over multiple datacenter this research can be of great value. In the first experiment such a method is implemented and used in practice. Additionally, it provides great scalability and an increase in performance for name resolving.

5.1 Adding traffic management

In this experiment a traffic management IaaS provider is integrated in an existing IT environment. For this an IaaS provider needs to be selected which can provide such a service. Earlier in this thesis three providers were discussed. As concluded in the last chapter it is important to know that a provider is easy to communicated with and to test this I asked (non)technical questions to see how competent the people are. All providers proved to be very competent and helpful. The decision for selecting a provider had to be made based on money and offered services. Because DNS made easy did not provide an suitable loadbalancer I needed to select between UltraDNS and DynDNS. Because DynDNS is far more cheaper with there queries and services I selected them to use.

DynDNS is a specialist in outsourced DNS and anycast DNS [1] they are really what you can call IaaS experts. They provide their services to Enterprise companies such

[1] www.dyn.com

as Twitter but also to small and medium businesses. DynDNS provide an intuitive interface which integrates all their DNS services and monitoring. Monitoring can be done on record and domain level. This is important because billing is done on the average QPS and you need to know how the overall distribution is per resource record.

5.1.1 Measuring resolve time

For this experiment the IaaS provider DynDNS is selected to integrate within a current environment. The integration of this provider should add geographical distribution of traffic to the current environment. This is needed for extending the environment with one or more datacenter sites. Additionally it also should add an increase in resolving performance because of the geographically spread 17 DNS servers and the usage of anycast technology. To test if it did increase the resolve performance an experiment is setup which monitors the site for a duration of 12 weeks. The first six weeks shows how the environment performs without the integration of DynDNS and the last six weeks shows how it performs with addition of DynDNS. This is done by 57 different monitoring stations which are located around the world. Each monitoring stations monitors monitor location; date/time and resolve time;.

On the 16th of September this experiment started with monitoring. After six weeks, on the 5th of December, the services of DynDNS is enabled. In Table 5.1 the resolve time is presented before and after the use of DynDNS. In Figure **??** the results gathered by monitoring stations located in America are shown in a graph.

The resolve time is decreased significantly especially for the continents America and Australia. The monitoring stations located in America shown an increase of 312ms which is an improvement of 559%. For Australia an decrease of 369ms was monitored. For Asia and Africa the increase in resolve time is 268% and 220%. Even for the European cities an improvement of 43ms is measured because the servers are

Figure 5.1: DynDNS resolve performance graph for America - In this graph the resolve time presented measured from multiple location in America. In this Figure the impact is shown which DynDNS have made on the resolve time from client's located in America.

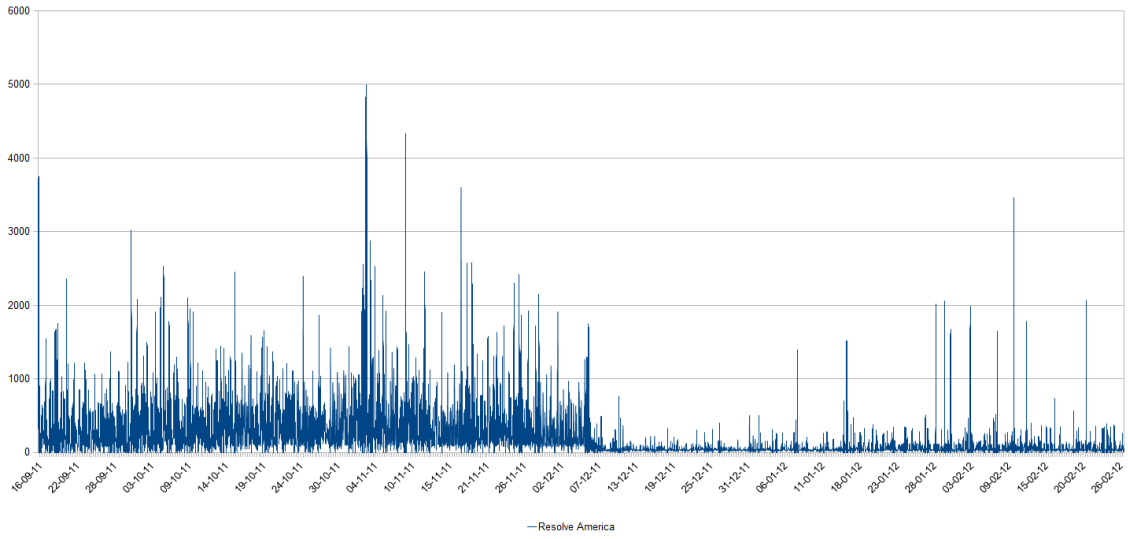


Table 5.1: Adding DynDNS to the current environment - Performance gain in milliseconds and percentage are presented. This is done for both before and after the integration of DynDNS.

Description	Asia	Australia	America	Africa	Europe
Before DynDNS	405ms	493ms	380ms	502ms	93ms
After DynDNS	151ms	123ms	68ms	228ms	50ms
Difference in %	268%	400%	559%	220%	186%

distributed over multiple countries withing Europe. In Figure 5.1 the graph also show a more stable resolve time, before the integration of DynDNS the resolve time was very unstable and it had large spikes. When DynDNS was introduced the resolve time shows a more stable resolve time with occasionally a spike.

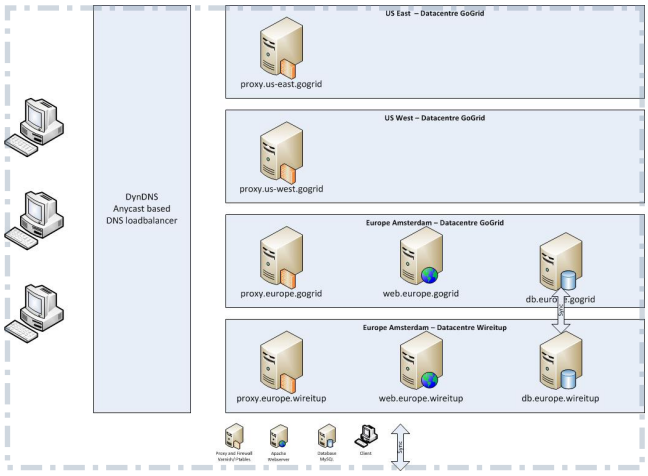
With the integration of DynDNS in this environment it is now possible to direct traffic from all over the world. DynDNS distinct seven traffic regions where each region can be configured with traffic redirection rules. With this it is now possible to direct traffic derived from West Europe to a server located in West Europe. When a server fails traffic can be redirected to another location or the faulty server can be remove from the pool.

5.2 Create a Hybrid environment with Cloud Infrastructure Services

The integration of DynDNS within the environment of the enterprise application makes it possible to add new server locations on different locations around the world. The goal of this experiment is to make the application faster for the clients connection from the United States because 65% of the clients are located there. This is done by adding two datacenter locations located in San Francisco, California and Ashburn, Virginia. The GoGrid servers are selected to provide the servers on those locations and DynDNS will route traffic from clients to the nearest datacenter location using anycast technology. Additionally, a datacenter in europe is added for redundancy and resilience.

The addition of the three datacenters provided by GoGrid results in the following architecture, Figure fig-gogridenvironmen. At the US datacenter locations San Francisco and Ashburn, proxy servers are installed which are configured with Varnish.

Figure 5.2: Adding three GoGrid datacenter locations to current environment - Showing the new IT environment which includes the three new datacenter locations offered by IaaS provider GoGrid.



The datacenter site from GoGrid located in Europe is a fully mirror of the WireITup datacenter site. The webserver and database server are mirrored but not used by the Varnish proxy. All proxy's are making connection to the WireITup location at Amsterdam. When that location has an outage the clients will be redirected to the GoGrid datacenter located in Amsterdam.

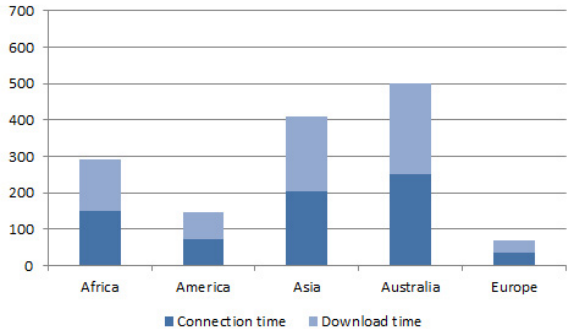
Two experiments are defined to test if the new environment results in a faster page loading for the clients connection from the US. The first experiment is measuring the performance change when a HTML page is cached. The second experiment involves a file of approximately 1MB in size to measure change in bandwidth. Those two experiments should give enough information about the performance impact of the new created IT environment.

5.2.1 Cache experiment

In Figure 5.3 and Table 5.2 the results are outlined from the first experiment. In this experiment a cache able page is downloaded from different locations around the

Figure 5.3: Experiment 1, adding GoGrid cloud servers to the environment - The enterprise environment is tested by downloading a cached page from 57 location all around the world to test connection and download performance for a duration of 7 days. At the right side are the results where the GoGrid servers are used to extend current environment and at the left side the results without the GoGrid servers.

(a) Local



(b) Global

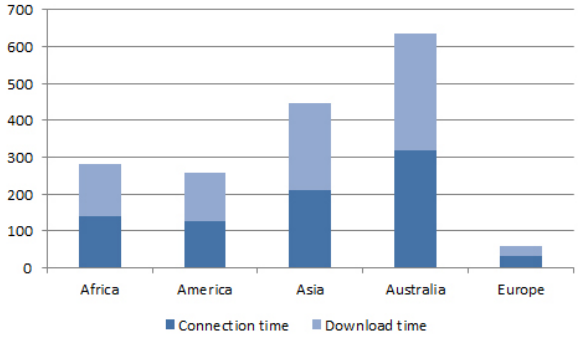


Table 5.2: The enterprise environment is tested with a cached page to test connection performance for a duration of 7 days and results are presented in milliseconds and difference between them in percentage.

Description	Africa	America	Asia	Australia	Europe
Before GoGrid	283ms	259ms	446ms	636ms	60ms
After Gogrid	293ms	144ms	408ms	501ms	70ms
Difference in %	-4%	44%	9%	21%	-16%

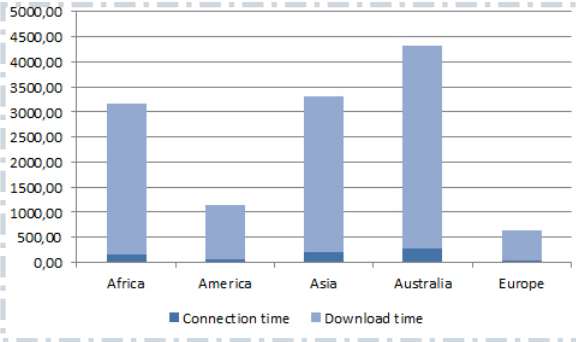
world. The primary goal of this experiment is to verify if the performance for the clients located in the United States is better because of the additional data center location in that region. According to our experiment the pages are served 115ms faster which is an improvement of 44%. The results are even lowered to 56ms if the monitoring stations from South America are removed.

5.2.2 Download experiment

In the next experiment a cached file of 1MB is downloaded during a period of two weeks to test if the addition of the GoGrid servers also adds more bandwidth perfor-

Figure 5.4: Experiment 2, adding GoGrid cloud servers to the environment- The enterprise environment is tested by downloading a 1MB file from 57 location all around the world to test connection and download performance for a duration of 7 days. At the right side are the results where the GoGrid servers are used to extend current environment and at the left side the results without the GoGrid servers.

(a) Local



(b) Global

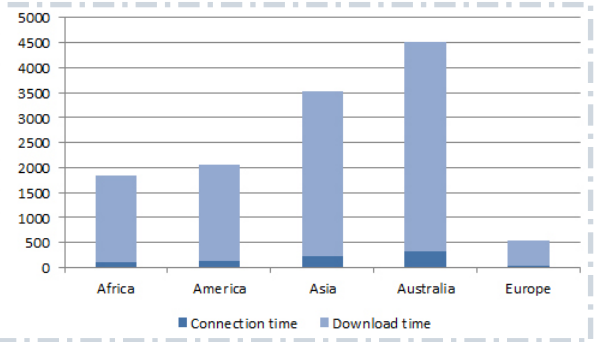


Table 5.3: The enterprise environment is tested with a file of 1MB test download time difference for a duration of two weeks and results are presented in milliseconds and difference between them in percentage.

Description	Africa	America	Asia	Australia	Europe
Before GoGrid	1842ms	2057ms	3526ms	4509ms	544ms
After Gogrid	3156ms	1147ms	3303ms	4309ms	631ms
Difference in %	-71%	44%	6%	4%	-15%

mance to the environment, in Figure 5.4 and Table 5.3 the results are presented. The bandwidth performance decreased for the monitoring stations located in Africa and Europe. The main reason for the decrease in Europe is that the monitoring station located in France had a w[illegible]e connection the servers of GoGrid located in Amsterdam. The average download time for that station was 1969ms. The goal to give a better performance for clients located is su[illegible]ed, the download speed is lowered with 910ms to 1147ms which is an improvement of 44%. Removing the results gathered from the monitoring station outside the United States results in an further improvement of an average of 655ms.

The two experiments shows that the addition of the two datacenter locations in the US results in a better connection performance. Cached pages and downloads results in a decrease of 44% for both experiments. For applications and websites where serving static content or where caching is possible these experiments shows that clients can benefit significantly by providing caching servers closer to your clients. The experiments shows that adding servers does not always results in an improvement. A good example are the results from Europe where GoGrid have connection problems with the monitoring station located in France. This resulted in a significant higher connection time.

Chapter 6

Related work

The integration of cloud computing services in enterprise applications is increasing furthermore. The quality of a single service can literally affect millions of customers. Therefor monitoring and assessing cloud services is done by many researches such as [2]. In those studies they generally focus on one or more subsystems (network, CPU, memory and disk I/O). They do multiple benchmarks which run relatively short and compare the outcome.

Other studies focus more on the migration of existing applications to cloud providers and compare differences in speed and cost [10, 26]. Jayasinghe found out that exchangeability of good performing configurations is not something trivial among providers. They also identified that there are differences between cloud providers and or hardware located in a datacentre. Wang found out that resource sharing can result in significant throughput instability and abnormal delay variation caused by the use of context switching.

This research is different because it is not focusing on raw speed or complex configurations but on endurance and the possible identification of performance degradation by monitoring low level kernel/OS metrics. In contrast with the benchmark studies

it is done by long running experiments on connection and the I/O subsystem. Other studies are focused on the QoS (Quality of a Service) delivered by cloud services [17]. GoIT is investigating the possible usage of a CPU metric in SLA for determine the QoS of a service. Many providers do not offer such a fine grained guarantee on the resource level. For example Amazon only offers an availability metric, annual uptime percentage and this need to be determined by the customer itself.

The method used in this research can be used to select a provider but we found out that the metrics could be used as a way to communicate with the technicians of cloud providers. It proofed to be a very powerful tool for communication when discussing the I/O problems with one of the IaaS providers. Something which I would not be able to do when the SLA would have been applied. It also give explanation for the disk subsystem why and how it can be over utilized and which signals it gives when this happens. This explanation is given by monitoring a hypervisors which contains one of the cloud services.

The case study done in the research of Khajeh-Hosseini ?? identified the potential benefits and risks associated with the migration to the cloud. In his case-study only cloud servers are used provided by Amazon and moved the entire environment to Amazon's servers. In our case study we setup an hybrid solution where servers provided by GoGrid where added to the existing enviroment to support current infrastructure. Additionally the performance gain was analysed by monitoring the old and the new environment.

Chapter 7

Conclusions

In this thesis endurance tests were used as a means to evaluate the availability and the performance of six IaaS providers. The objective was to point out behaviours of the servers which cannot be identified in a standard peak performance study. In a number of tests presented in this paper, we have identified behaviours. Firstly, the increase of I/O operations, which results in an increase of IOWait and TimeSpendIO. Secondly, the increase in I/O operations has a direct impact on other instances running on the same I/O subsystem, and finally a relatively small increase in I/O operations could overload an I/O subsystem.

It is thus in the interest of IaaS providers to be aware of such phenomena in order to react quickly or even prevent a drop in performance of its servers. This in fact happened during one of the endurance tests. After contacting the responsible provider and reporting the drop in performance, they immediately decided to schedule a replacement for the faulty serve. IaaS providers should consider to monitor all write-, read- I/O operations, IOWait and TimeSpendIO metrics of all cloud servers they host. This knowledge can prevent resource starvation on the I/O subsystem and therefore provide better resource sharing and the identification of heavy utilized cloud instances

on their I/O subsystem.

When application characteristics are known, system administrators can identify which IOWait and TimeSpendIO should be considered as normal. The usage of thresholds to check if kernel metrics are within acceptance boundaries, system administrators can respond to performance degradation. During the second experiment one of the cloud servers suffered from performance degradation and eventually crashed, there was enough time to respond and inform the cloud provider which would prevent possible downtime.

When selecting a cloud provider it is very important to know the performance characteristics and the location of the consumers. For most applications there are good commercial load generation tools available. Our experiments show that when the I/O subsystem is heavily utilized the IOWait and TimeSpentIO of a cloud instance is also increasing, this knowledge can be used as a reference to select a provider. Last but not the least, the support desk is of proved to be of great help when problems occur, it is thus important to take into consideration when selecting a cloud provider.

The second section is about the integration of two IaaS providers into an existing IT environment. This is done by means of a case study. Such a case study adds besides theoretical also practical value to this research. The primary goal of the experiments done was to identify the performance gain for the clients located in the United States. Secondly, to identify the global performance increase or decrease. Lastly, write down the experiences and difficulties by doing such an case study.

By implementing a traffic management IaaS provider, a means to distribute traffic over multiple datacenters is added. Secondly it adds an performance to name resolving of 559% for clients located in the United States. Lasty, a more resilience IT environment is created because traffic can me managed and redirected to other data centers. The experiment and method can be of use for any IT architect who want to implement

such an technique into their own environment. The experiment shows performance gains for all continents.

The second experiments adds three datacenter locations to the existing environment. By adding two datacenter locations in the United States our aim was to improve the experience of the clients located in the United States. The experiments done to verify this shows an overall increase of 463% (259 naar 56ms) in performance gain by the monitoring stations located in the United States. These two experiments also demonstrates that adding additional datacenter locations became very easy because of the use of traffic management provider.

Bibliography

[1] Jens Nimis Stefan Tai Alexander Lenk, Markus Klems and Thomas Sandholm. What's inside the cloud? an architectural map of the cloud landscape. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 23–31, Vancouver, BC, May 2009.

[2] M. Nezih Yigitbasi Radu Prodan Thomas Fahringer Alexandru Iosup, Simon Ostermann and Dick H.J. Epema. Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on parallel and distributed systems*, 22(6):931–945, June 2011.

[3] David Greenwood Ali Khajeh-Hosseini and Ian Sommerville. Cloud migration: A case study of migrating an enterprise it system to iaas. In *IEEE 3rd International Conference on Cloud Computing*, pages 450–457, Miami, Florida, July 2010.

[4] Paul Barham. Xen and the art of virtualization. 2003.

[5] Mildo van Staden Bas Kotterink. Crowdsourcing strategieen voor de publieke sector. 2009.

[6] A. Belloum, M.A. Inda, D. Vasunin, V. Korkhov, Zhiming Zhao, H. Rauwerda, T.M. Breit, M. Bubak, and L.O. Hertzberger. Collaborative e-science experiments and scientific workflows. *Internet Computing, IEEE*, 15(4):39 –47, july-aug. 2011.

[7] Anthony Bisong and Syed (Shawon) M. Rahman. An overview of the security concerns in enterprise cloud computing. *International Journal of Network Security and Its Applications*, 3(1):30–45, January 2011.

[8] Brebner. Performance and cost assessment of cloud services. 2011.

[9] R. Buyya, C.S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. *High Performance Computing and Communications*, pages 5–13, September 2008.

[10] Qingyang Wang Jack Li Pengcheng Xiong Deepal Jayasinghe, Simon Malkowski and Calton Pu. Variations in performance and scalability when migrating n-tier applications to different clouds. In *IEEE 4th International Conference on Cloud Computing*, pages 73–80, Washington, DC, July 2011.

[11] Nathan Eagle. txteagle: Mobile crowdsourcing. In *Internationalization, Design and Global Development Third International Conference, IDGD 2009, Held as Part of HCI International 2009*, pages 447–456, San Diego, CA, USA, July 2009.

[12] Simson L. Garfinkel. An evaluation of amazon's grid computing services: Ec2, s3 and sqs. August 2007.

[13] Gartner. Gartner says worldwide cloud services market to surpass 68 billion in 2010. `http://www.gartner.com/it/page.jsp?id=1389313`. STAMFORD, Conn., June 22, 2010.

[14] Takahiro Hirofuchi. A live storage migration mechanism over wan for relocatable virtual machine services on clouds. In *Proceeding CCGRID '09 Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 460–465, Washington, DC, USA, 2009.

[15] Jeff Howe. The rise of crowdsourcing. 2006.

[16] Ioan Raicu Ian Foster, Yong Zhao and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop (GCE '08)*, Austin, TX, November 2008.

[17] J. Oriol Fito Mario Macías Inigo Goiri, Ferran Julia and Jordi Guitart. Resource-level qos metric for cpu-based guarantees in cloud providers. In *Economics of Grids, Clouds, Systems, and Services*, volume 6296, pages 34–47, Ischia, Italy, August 2010.

[18] Alexander Keller and Heiko Ludwig. The wsla framework: Specifying and monitoring service level agreements for web service. *Network and Systems Management*, 11(1):57–81, March 2003.

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

[20] Rean Griffith Anthony D. Joseph Randy Katz Andy Konwinski Gunho Lee David Patterson Ariel Rabkin Ion Stoica Michael Armbrust, Armando Fox and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. *Technical Report No. UCB/EECS-2009-28*, February 2009.

[21] Alessandro Nuvolari. Collective invention during the british industrial revolution: the case of the cornish pumping engine. *Cambridge Journal of Economics*, 28(3):347–363, February 2004.

[22] Ayodele Onibokun and Mayur Palankar. Amazon s3: Black-box performance evaluation. *Project Report*, 2007.

[23] Dayanand Rudraiahl. Cloud computing is key to business virtualization. 2010.

[24] SUN. Introduction to cloud computing architecture. 2009.

[25] Jinesh Varia. Cloud architectures. 2008.

[26] Guohui Wang and Eugene Ng. The impact of virtualization on network performance of amazon ec2 data booktitle =.

[27] Darrell M. West. Saving money through cloud computing. April 2010.

[28] Jinzy Zhu, 2010. IBM Cloud Computing Case Studies.