Vrije Universiteit Brussel

Operating Systems and Security

Mini-project essay

High availability and scalability on GNU/Linux clusters

Author:
Pieter Libin

Professor:
Prof. Dr. Timmerman
Assistant:
Mr. Fayyad-Kazan

January 12, 2014

Contents

0.1 Source code . . . 5
0.2 Videos . . . 5
0.3 Naming conventions . . . 5
0.4 Presentation . . . 5

1 Introduction **7**
1.1 Abstract . . . 7
1.2 Interest . . . 8
1.3 Relevance . . . 8

2 General scope of high-availability and scalability **9**
2.1 Airplane/car/spacecraft software . . . 9
2.1.1 On-board computers . . . 9
2.1.2 Computer Aided Design . . . 10
2.2 Healthcare . . . 10
2.2.1 Patient monitoring . . . 10
2.2.2 Drug development . . . 10
2.3 Desktop systems . . . 10
2.4 Home entertainment systems . . . 10

3 Different aspects in cluster availability **12**
3.1 Hardware . . . 12
3.1.1 Servers . . . 12
3.1.2 Rack server . . . 12
3.1.3 Blade server . . . 13
3.1.4 Network . . . 13
3.1.5 Storage . . . 13

3.2 Infrastructure 14
3.2.1 Sites 14
3.2.2 Location 14
3.2.3 Buildings 15
3.2.4 Energy providers 15
3.3 Internet Availability 15
3.4 Storage 15
3.5 Database systems 17
3.5.1 Relational database management systems 17
3.5.2 NoSQL databases 17
3.6 Application crashes 17
3.7 Human error 18
3.8 Operating systems 19
3.9 Monitoring 19
3.10 Virtualization 20
3.11 Different kinds of outages 20
3.11.1 Hardware 20
3.11.2 Environmental and physical failure 21
3.11.3 Network failures 21
3.11.4 Application failures 21
3.12 Definition of high availability, expectations and realistic goals 22
3.12.1 Definition of high availability 22
3.12.2 Expectations and realistic goals 22

4 Scalability and its effects on economy and environment 23
4.1 Power consumption 23
4.2 Software efficiency 23
4.3 Climate change 23

5 Identifying high availability and scalability solutions 25
5.1 Cluster nodes 25
5.1.1 Cluster node monitoring 25
5.2 Computational cluster nodes 26
5.2.1 Application monitoring 26

5.2.2 Cluster node management . . . 26
5.2.3 Load balancing proxies . . . 27
5.3 Storage and backup . . . 28
5.3.1 Data replication . . . 28
5.3.2 Backup . . . 29
5.3.3 Distributed file systems . . . 30
5.3.4 Databases . . . 30

6 Experimenting with HA and scalability solutions 32
6.1 Open source technologies . . . 32
6.2 Test environment setup . . . 32
6.3 Monitoring . . . 33
6.3.1 Experiment overview . . . 33
6.3.2 Nagios experiment . . . 33
6.3.3 Ganglia experiment . . . 42
6.3.4 Nagios and Ganglia: evaluation . . . 45
6.4 HAProxy experiment . . . 46
6.4.1 HAProxy installation . . . 46
6.4.2 HAProxy testing . . . 48
6.4.3 Conclusion and remarks . . . 48
6.5 Database replication experiment . . . 49
6.5.1 Installation . . . 49
6.5.2 Testing the experiment setup . . . 52
6.5.3 Conclusion and remarks . . . 52
6.6 Software disk replication experiment . . . 53
6.6.1 Adding the “disks” . . . 53
6.6.2 Setting up the network RAID-1 . . . 54
6.6.3 Conclusions and remarks . . . 57
6.7 Distributed file system experiment . . . 58
6.7.1 Architecture . . . 58
6.7.2 Installation . . . 58
6.7.3 Conclusion and remarks . . . 63
6.8 Database shards experiment . . . 64
6.8.1 Setting up the shards . . . 64

6.8.2 The database schema 65
6.8.3 The application 66
6.8.4 Conclusion and remarks 67
6.8.5 Additional thoughts 67

7 Creating a 'Wikpedia clone' cluster 69
7.1 Functional analysis 69
7.2 Some Wikipedia statistics 70
7.2.1 Presenting the statistics 70
7.2.2 Interpretation of the statistics 73
7.3 An attempt to setup an architecture 73
7.3.1 Storing the pages 73
7.3.2 HTTP server for editors 74
7.3.3 HTTP server for viewers 74
7.4 Testing this architecture 75
7.4.1 The content storage server 75
7.4.2 The HTTP nodes 77
7.4.3 Testing the setup 78
7.4.4 Conclusion and remarks 79

8 Conclusion 80
8.1 State of the art 80
8.2 What did I learn 80
8.3 What does this project contribute 81

Preface

0.1 Source code

When I reference to source code files, this will always be in the src directory. This directory can be found in this public github.com project: https://github.com/plibin-vub/OSSEC-mi

0.2 Videos

For some demonstrations I created a video, I've put these videos on Youtube and reference to them in this document via their YouTube URL.

0.3 Naming conventions

Server names are always expressed using single quotes (e.g.: 'node1').

0.4 Presentation

Before the presentation, I will hand in the following deliverables:

- this final report
- the final presentation
- a directory with all the edited videos
- a directory with all pictures that I used in the report
- source code (the src directory on github)

- the base ubuntu virtual machine, which I used to derive all other virtual machines from (root-password=”empty”, root-user=”plibin”)

During the presentation, I will present some videos that depict experiments that I executed (most of the videos in the deliverable).

Chapter 1

Introduction

1.1 Abstract

Nowadays, several web-applications exist (Facebook, Wikipedia, Twitter, ...) that serve millions of users concurrently. To achieve this, huge clusters are in place, where each cluster node serves a large amount of users. In order to assure the availability of such applications, it is required that whenever a cluster node crashes, the systems remains responsive and handles subsequent requests by another cluster node.
The number of users of such web applications can vary greatly over different time intervals (e.g.: Twitter tends to be under heavy load upon the arrival of significant events). Also, when an application gains in popularity, the number of users grows quickly in a short timespan. This requires an application to be able to scale the number of users it can serve.
This project situates these concepts in the general ICT landscape and makes a detailed analysis on the importance of availability and scalability provisions on GNU/Linux server clusters.
The project identifies a set of technologies that are available on the GNU/Linux platform to set up a system that can assure high availability and support proper scalability for both computational and storage nodes. The selected technologies will be tested in a simulated cluster setting.
In the last phase of the project, a high-availability and scalable cluster simulation is constructed that mimics the functionality of Wikipedia.

1.2 Interest

A large amount of today's software applications runs on remote servers, and this amount will most likely grow even more in the near future. In order to allow users to have a good experience, it is vital that the application remains available at all times. For applications that are only used by a handful of users, this might be an easy task by just providing a redundant server, however, for applications that are used by thousands (or millions) of concurrent users, this becomes quite complex.
Another aspect is that such applications need to be able to scale; a sudden rise in popularity may cause an application to be used by a x-fold of users from one day to another.
These considerations of high-availability and scalability interest me greatly because they have a huge impact on how applications are hosted; and this again determines how such applications are to be implemented.
In my opinion, having an overview and understanding of the technologies and architectures that high-availability and scalability setups use, is a big asset for a computer scientist.
I have always been fascinated by large data centers, while the hardware of such setups is no longer my main interest (this is more inclined to software these days) I still remain intrigued how the different components in such setups interact.

1.3 Relevance

Many huge companies (Google, Facebook, Microsoft, ...) and organisations (Wikipedia, ...) depend on available and scalable applications [1]. This makes high-available and scalability solutions economically relevant. Also in the High Performance Computing domain high-available and scalability solutions become more and more important to operate huge research clusters. Such research clusters are essential for scientists and companies to do research.

[1]And the number of companies that operate in the "Cloud" seems to grow every days

Chapter 2

General scope of high-availability and scalability

This project will focus on GNU/Linux clusters, however, the aspect of high-availability and scalability are not limited to this domain. In this chapter, I present a short overview of the importance of these aspects in other domains.

2.1 Airplane/car/spacecraft software

2.1.1 On-board computers

Cars and planes that are released today are operated by millions of lines of code [1]. It is of course vital in such devices that there is a high level of availability; important systems (breaks, steering, ...) should be available in a redundant way. Another important aspect is that failure should be detected as soon as possible; if certain parts of a car break down without the vehicle being aware of this, a very dangerous situation can arise.
For spacecrafts, the same rules apply when considering passenger safety. However, also unmanned flights, such as the mars exploration missions need to consider an extra dimension of availability. These vehicles can only be controlled by on-board computers or via a satellite up-link (where signals take 1 to 1.5 hours [2] to go from one end to the other). These vehicles have a huge development and production cost, which is an extra motivation to keep the system running as long as physically

possible.

2.1.2 Computer Aided Design

The design of vehicles can be assisted with several software solutions, these software packages require a lot of computational power and can benefit from the use of High Performance Clusters [3].

2.2 Healthcare

2.2.1 Patient monitoring

For patient monitoring equipment, it is vital that these devices report problems as soon as possible. Usually these devices can be quickly replaced with a working unit, but not knowing that a device is corrupted might cost patient lives.

2.2.2 Drug development

For drug development both academic researchers and pharmaceutical company staff use High Performance Clusters intensively to study the effects of drugs [4] [5].

2.3 Desktop systems

The stability of desktop computers has improved a lot the last decade, mainly by stabilising desktop operating systems. A huge source of system crashes in the past were buggy device drivers that made the entire operating system crash. Current desktop operating systems (e.g.: Windows NT) are built to recover from such crashes [6].

2.4 Home entertainment systems

These appliances (TV's, game consoles, media center) are used via a limited interface (e.g.: remote controls) and users expect them to work without any

problems (as their VCR a couple of decades ago).
While system crashes might still be acceptable on the desktop, this is definitely not the case with these kind of devices.

Chapter 3

Different aspects in cluster availability

3.1 Hardware

3.1.1 Servers

Servers are the processing nodes in a cluster, they contain a CPU (often more than one), memory (often a significant amount), one or more network interfaces, a power supply (often 2 for redundancy) and usually one or more hard disks.
Several server-cases exist:

Tower server

A tower server is comparable to a desktop PC tower. It's cheap, but it is an inconvenient format to use in cluster rooms.

3.1.2 Rack server

These servers are build as thin boxes that can be stacked on top of each other in a rack.

3.1.3 Blade server

A blade server is a large computer that is able to contain multiple blades. A blade is essentially a slice containing memory, a CPU and a disk. The power and networking is managed by the surrounding case. Blade servers are convenient for cluster setups since they allow to replace (broken) blades very easily.

3.1.4 Network

The network infrastructure connects the different elements in the cluster and allows the application that runs on the cluster to connect to the internet.
Several components are used to setup cluster networks:

- switches
- routers
- firewalls

3.1.5 Storage

Storage can be integrated in a server. Solutions exist for implementing storage that can be integrated directly in the network:

- network attached storage
- storage area network

There are also separate units that can be connected to a server directly (with no network in between) : direct-attached storage.

Disk types

Currently, two types of disks exist: hard disk drives (HDDs) and solid state disks (SSDs). A HDD contains rotating platters and stores data by magnetising parts of the disk. An SSD is fully implemented in an integrated electronic circuit; it contains no moving parts at all (comparable to a USB flash drive).
Overall, SSDs (which are only "recently" on the market) have a much better performance. However, traditional HDDs still offer a much better price/MB ratio

than SSDs.
Therefore the choice of using HDDs or SSDs usually depends on the problem you want to solve: quickly accessing a limited amount of data vs storing vast amounts of data that are only issued periodically [1].

3.2 Infrastructure

3.2.1 Sites

Big companies such as Google, Facebook, Microsoft, ... have their own sites to host their clusters. Smaller companies often start by hosting their applications at a third party's site. Some of these third party providers offer servers that can be completely administered by their users (e.g.: rackspace [7], Amazon EC2 [8]), others provide cloud platforms that allow the users to simply install an application (e.g.: Google App Engine [9]).

3.2.2 Location

An important aspect to consider is the distance of the cluster to the end users. When this distance is too big, users will suffer from the network connection's latency. For example: between the US and Europe, a latency of $\approx 90ms$ can be expected [10]. This is a significant number when we consider that a latency of 1/10th of a second (100ms) already starts to feel slow for an end user [11].
Therefore it is beneficial to have the data center as close to the end user as possible.

Another important aspect is the climate of the region where the data center is located. Data centers tend to get hot and cooling down the datacenter can become an expensive (and carbon-unfriendly) issue. A solution can be to move the datacenter to a colder region [12].

Proximity to an energy provider is another factor that needs to be considered (see subsection 3.2.4).

[1]The two systems are often used in combination, so the technologies can complement each other

3.2.3 Buildings

The buildings that host a data center need to take a set of precautions to avoid or handle natural disasters such as floods, fires, earthquakes and extreme weather.
The buildings should be well secured (physically), since quite often, private data of end-users will reside on the servers in the building.
Applying passive architectural techniques can allow the data center to consume dramatically less power [13].

3.2.4 Energy providers

Data centers might produce their own power on-site by using solar panels or even building a dedicated power plant.
It should be taken under consideration whether a country / region has trustworthy energy providers (which can be a problem in development countries).
A company will negotiate a contract that guarantees the availability of power within a reasonable degree of certainty.
The datacenter should be able to handle short power-outs by using UPSs. To deal with longer power-outs diesel-powered generators can be used.
Considering climate change, the cost of energy could rise significantly over the next years, rendering these aspects even more important.

3.3 Internet Availability

Most companies will work with a third party provider to setup a connection with the Internet.
A company will negotiate a contract that guarantees the availability of Internet within a reasonable a reasonable degree of certainty.

3.4 Storage

A proper backup strategy should be in place. This strategy will depend on the companies' resources and the importance of the different datasources. Some general guidelines for backups (based on [14]):

Mirroring does not replace backups

If a file is removed by a human error, a mirrored solution will not be able to restore the data. Additionally, it is still possible that these disks will fail at the same time (for example when the power is interrupted).

Backups are more often used to fix human errors than to fix a disaster

For this reason, it is important that restoring data can happen quickly.

Regularly test the restore procedure

It is always possible that there is a hardware problem or some backup rule is invalidated. To make sure this is not discovered in the case of a disaster, it is important to regularly test both the hardware and the restore procedures.

Handle hardware (and tapes) with care

A lot of backups are made on tapes, such cartridges are particularly sensitive to dust and moist, so it is important to store them appropriately.

Only use a tape as many times as the producer guarantees it will work

Tapes have an expiration date, make sure to stop using them after this date. If you dispose of them, make sure to destroy them thoroughly, since they might contain private data.

Multiple locations

If you can afford it, it is always better to distribute backups over multiple physical locations. In case there goes anything seriously wrong in one of the data centers, it is still possible to recover from this.

3.5 Database systems

The following two (mainly) database technologies are currently used in cluster settings: Relational database management systems and NoSQL systems.

3.5.1 Relational database management systems

This technology is based on the relational model [15]. It allows data to be modelled as tables that represent entities. These entities are then connected using relations (basically pointers from one table to another).
It is possible to query the information in the model using the declarative language SQL.
RDBMs implement ACID and transactions, two features which are very useful for application developers, but make scaling RDBMs difficult.

3.5.2 NoSQL databases

The concept of NoSQL databases was introduced as a reaction on the difficulties of scaling out RDBMs. While RDBMs store data using a relational model, most NoSQL solutions simply act as a key-value store ($\approx$ a distributed dictionary data structure).
NoSQL databases are usually simpler in design and allow for easier horizontal scaling [16].
However, since data is simply stored in a key-value store, joining over multiple datasets (which can be easily done with RDBMs) has to be implemented on top of the NoSQL database (doing this might still be slow [17]).

3.6 Application crashes

Software will never be perfect, therefore there will always be the possibility that a program crashes. Important is that developers try to create defensive software, that will show an appropriate error message, rather than crash.
That said, since we cannot built a high-availability cluster assuming our software will not crash, we need to try to contain the crashes as good as possible.

When software is implemented as a process that runs in the user space, it cannot crash the entire operating system, but only that specific process (if there are no bugs in the operating system). In my opinion this is essential to ensure the stability, if not, one error can ultimately take down the entire cluster.
But still, if we have a web server that is written in C/C++, when an error occurs in one of the applications threads, this can potentially bring down the entire application server. Java applications can cope better with these kind of errors: when one application thread throws an exception the application server (Tomcat, JBoss, ...) can continue to serve the other users.
In order to ensure that an error in a C/C++ program does not crash the entire application (and all sessions that are tied to it), a process can be assigned to each application session, however, running the application like this will require more computational resources.
When one session of the application throws an error and the application server can properly recover from it (by using one of the techniques described in the previous paragraphs), we can show the user an appropriate error message or try to continue this user's session in another thread/process.
Another very important remark : when a crash happens, it is important to properly log what is wrong and inform an administrator about this problem. This way the problem can be investigated and hopefully fixed. Because it is not always possible to look what is going on on the production server, it is important for the logging to be as complete as possible. This way developers and administrators can try to reproduce the problem without the need to access the production system.

3.7 Human error

A lot of problems are introduced by human errors: bugs in applications, bugs in operating systems, errors in architectures and their implementation, ...
While it makes sense to have redundancy in hardware, it makes equally sense to have redundancy in developers and system administrators.
For developers, it is vital to have peer reviews, or even pair development techniques for the more complicated parts of a system.
System administrators often take decisions that can impact huge parts of a cluster, so it would probably be a good idea to double-check such decisions before actually

applying them. This can be done by first applying the changes on a test setup before pushing them to the production cluster [2].
It is also important that the knowledge of systems and processes is well distributed over employees, in case any of the employees leaves, the system can still be operational. Therefore it is also important that architectures, procedures and system layouts are thoroughly documented, and that this documentation is kept up to date.

3.8 Operating systems

GNU/Linux is very popular in cluster settings [18]. The GNU/Linux operating system is known for its speed and stability. It is however important to note that the kernel has a monolithic design, while this makes it a fast kernel, it also allows system drivers to crash the entire kernel (in extremis).
Since clusters will usually use reputed well-tested hardware that has good driver support for the Linux kernel, this is usually not a big problem.
When an operating system does crash for some reason (a bug, or a hardware failure) this should be dealt with appropriately, for example by restarting the cluster node that has crashed or (if the crashed node cannot be started again) by starting a new node that will take the crashed node's place.
When a crash happens, it is essential to investigate what went wrong, so lessons can be learned and actions can be taken to avoid the problem from happening again.

3.9 Monitoring

It is important to monitor the status of a cluster, to take actions in case there is something wrong. On the other hand, we should also take care not to overload the network by constantly polling nodes.
Two aspects are particularly important when monitoring:

- gathering usage statistics (CPU usage, network load, ...)

[2]Its vital for system administrators to have a sandbox that allows them to perform realistic tests

- diagnose problems in time, and act on them

3.10 Virtualization

Virtualization is a technique that is used a lot in cluster environments. It allows us to deal with server resources in a more abstract fashion.
This allows certain actions to be automated in an easier way; for example: restarting a node only requires invoking a program.
Furthermore, it allows server resources to be split up easier; one virtualised node can use a part of the physical memory of an underlying server, or only 1 of the set of CPUs the underlying server has installed.
Virtualising hardware has some costs in terms of performance, however, these days, several hardware provisions are in place to further improve the performance of virtual machines (e.g.: Intel's Hardware-Assisted Virtualization Technology [19], AMD Virtualization [20]).

3.11 Different kinds of outages

I will present an overview of the different kind of outages that are possible (based on [14]):

3.11.1 Hardware

Hardware can fail whether it is new or old. Especially prone for defect are the moving parts in a computer: the HDDs, tape drives, tape libraries and fans.
To extend the computer's lifetime, the machine needs to be properly cooled [3].
There should be as little dust in the room as possible to avoid the fans to get dirty (and eventually fail).
The life-time of servers that run constantly under heavy load is considered to be 3 years [21].

[3]A sufficient amount of fans is required and the room needs to be air conditioned

3.11.2 Environmental and physical failure

Natural disasters

- earth quakes
- fires
- floods
- extreme weather

Infrastructure failure

- cables that break
- air conditioning that breaks down
- UPSs or diesel generators that fail

Third party delivery failure

- internet delivery failures
- power delivery failures

3.11.3 Network failures

It is possible that network components fail (this can be connected to error in hardware and/or software). Also problems with the DNS or DHCP servers might occur.
Distributed Denial-of-service (DDOS) or other security attacks can also shutdown an entire network.

3.11.4 Application failures

All layers of an application can experience problems: the web/application server, the database or the actual application. It is important to be aware of this and write applications in a defensive way.

3.12 Definition of high availability, expectations and realistic goals

3.12.1 Definition of high availability

I use the definition used in [14]:
"Availability is a measure of the time that a is server is functioning normally".
Availability can be calculated with this formula [14]: $A = \frac{MTBF}{MTBF+MTTR}$ [4].
An example: to achieve 99.9999 % availability one is permitted to 6 minutes downtime in 11.4 years! This is highly improbable to be achieved [14].
A more realistic goal could be 10 minutes downtime per year, this is expressed as an availability of 99.998% [14].

3.12.2 Expectations and realistic goals

Unavailability can be costly: people of your online web shop might leave and look for another shop, imagine the stock market not able to respond, ...
We should always try to be realistic in our expectations, if 50% of a company's data centers fail, there is a good chance not everyone will still be able to be server. However, this is such a far-fetched scenario that it is probably not cost-effective to try to preempt it.
No matter how well an infrastructure is set up, there are always aspects that are beyond our control [22], so it is important to be careful promising availability percentages to customers.

[4] MTBF = mean time between failures, MTTR = mean time to repair

Chapter 4

Scalability and its effects on economy and environment

4.1 Power consumption

The power consumed by large web applications is huge. As an example, in 2011 Facebook consumed 532 million kilowatt hours of energy and Google consumed 2 billion kilowatt hours of energy in 2010 [23] [1]

4.2 Software efficiency

For some companies (such as Facebook) one of the reasons why this power consumption is so high is that it takes a lot of computation to render PHP pages [24]. Facebook in particular is trying to fix this by creating a PHP virtual machine [25].
Note that for such projects, using more computation-efficient programming languages (such as C++ and Java) from the start could have prevented these efforts.

4.3 Climate change

Needless to say that numbers in the range of Google's or Facebook's power usage contribute significantly to climate change. Since it can be expected that the

[1]As a comparison: Belgium used 84 billion kilowatt hours in 2008

demand for cloud solutions will only go up, my opinion is that green data centers [2] will become more important in the near future.

[2]And applications!

Chapter 5

Identifying high availability and scalability solutions

5.1 Cluster nodes

5.1.1 Cluster node monitoring

The two major monitoring tools on GNU/Linux platforms are Nagios and Ganglia. Both tools implement several monitoring aspects, but the focus of each of the tools is different. Nagios is more oriented towards detecting system problems and sending out notifications for such events, while Ganglia provides a long term overview of the status of cluster nodes. To ensure high availability, detecting problems is the most important aspect. The sooner problems are revealed, the sooner actions can be undertaken to avoid any impact on end users. It is important that problems are properly communicated to system administrators, since they might be able to further investigate the problem. In the event of any hardware failure the intervention of an administrator will be required. However, it is also necessary to execute programs to repair the state of the cluster upon the detection of failure. Nagios specialises on both these issues [26]. In order to make new nodes available when necessary, we need to monitor the resource availability on all of the online nodes. Monitoring statistics also allows system engineers to gain insight in performance needs of an application over the course of time. Such statistics can also be used to setup models that can be used to predict system load [27]. Ganglia collects these kind of statistics and stores them in a round-robin database, namely RDDTool [28] [29].

5.2 Computational cluster nodes

5.2.1 Application monitoring

Monitoring a cluster node is not enough to guarantee that it is properly servicing clients. It is possible that the application (for example: the web application server) has gotten in an infinite loop, and as such is using practically all of the CPU's power, but in reality is processing no new client request. A possibility to monitor applications is to keep track of the number of client request per seconds (or another appropriate time unit) that get processed and compare it with the amount of computational resources that is used. This number of client requests is very application specific; a web server might process thousands client requests per second, while a server responsible for rendering a scene in an animated movie might only process one request per hour [30]. This information usually can be extracted from the log files generated by the server application (for example: the request log file for Apache httpd). This information than can be passed to Ganglia to be stored and monitored together with the other server's statistics [28].

5.2.2 Cluster node management

In large cluster environments, it is not possible to wait for a human intervention when a problem occurs. Therefore, when nodes or applications fail, the monitor server should restart them. There are several systems that allow for programs to restart physical servers that reside in a rack (for example Dell's DRAC). Since I will only use a simulated cluster in this mini-project, I will only deal with starting and stopping virtualised nodes [1]. I will use VirtualBox to set up my simulated cluster, this virtualization software has support to start and stop virtual machines using the command line. When an application stalls but the operating system is still fully functioning, the monitoring server can decide to restart the server

[1]Which I think should be a valuable contribution taken into account the large amount of virtualised server that are used in data centers these days

application, or (if the application has support for this) simply kill the thread that is responsible for the problems.

5.2.3 Load balancing proxies

Load balancing is a method for distributing workloads across multiple computing nodes. It can be used to improve the reliability of a system through redundancy and to provide scalability by sending requests to the most available node. Load balancing has several applications such as High Performance Clusters, database servers, and HTTP servers. In this section I will focus on HTTP load balancing. A prominent TCP/HTTP load balancing proxy is HAProxy. [31]. This software supports a set of balancing algorithms [32], most notably:

- round robin: each server is used in turns, taking into account the different server's weights (these weights can be configured per server and can be adjusted on the fly)
- balance leastconn: the server with the lowest number of connections receives the connection
- source/uri: this algorithm hashes respectively the source IP or URI of the request, this hash is than divided by the total weight of the running servers to determine which server should receive the request

HAProxy also supports Access Control Lists configurations that allow the load balancer to select a server based on a header in the HTTP request. When multiple server clusters exists in different countries and/or continents, this feature allows us to connect a user to a server that is located near his own location. Another important aspect to consider when setting up a load balancing infrastructure for HTTP servers is that many dynamic web applications rely on session context [32]. Since this session context is usually stored on the web server, it is necessary for the load balancer to pass the client's request to the same web server: this is called persistence.

5.3 Storage and backup

5.3.1 Data replication

Data replication implies that the same date is stored on multiple data storage systems. This should be transparent to the end user: the user should not be aware of this when using the data (note that when working with a multi-tier architecture, it is possible that the 'end-user' is represented by another tier in the system). Data replication is relevant for both:

- high availability systems: to ensure a smooth transition from a failing data storage node to a working node
- systems that require scalability: to make load balancing possible

We can distinguish 3 types of data replication [33] :

Database replication

Master-slave replication In this setup, there exists one master database, this database is the only instance that accepts updates. When an update statement is received by the master database, it is applied to the database and appended to the log. The statements in this log are than propagated to the slave databases. Most database systems support this replication strategy (Postgres [34], MySQL [35], ...).

Multi-master replication In this setup, multiple master instances exists, each of these master instances accept updates. These updates are than communicated with the other master instances. This has the advantages that :

- the master node can fail without interrupting the system
- the update load can be distributed over multiple nodes

There are also some significant disadvantages related to this technique:

- increased complexity
- most implementations violate the ACID constraints

- to fix conflicts that may arise between different database instances, resources of the database nodes are required and communication between the conflicted nodes will increase the network traffic

Note that there are other techniques to accommodate the advantage of load balancing mentioned above: database shards, NoSQL (reference to this section).

Disk storage replication

Disk storage replication collects updates to a block device and applies these updates collectively to multiple devices. This can be implemented in hardware, the functionality is than embedded in the array disk controller. DRBD [36] implements this functionality in software. This software allows users to mirror disks within a system or over a network (the DRBD website explains this as follows: "DRBD can be understood as network based RAID-1" [36]).

File-based replication

Disk storage replication replicates entire block devices, however for some applications, it might be necessary to only replicate parts of the logical file system.

Batch replication To replicate file systems in batch, synchronisation tools such as rsync [37] can be used. The rsync Unix tool can synchronise 2 directories from one location to the other (this can be done over a network). The advantage compared to a simple "scp -r" is that rsync will only transfer the changes by using delta encoding.

Real-time replication I was not able to find any tools on GNU/Linux that allow real-time file-based replication based with default filesystems such as ext4. It is however possible to achieve this kind of file-based replication by using one of the distributed file systems, which I will discuss in the next section.

5.3.2 Backup

In order to setup a reliable backup system, some technologies are required:

- the ability to take a snapshot of a file system: this can be achieved by using the Logical Volume Manager, which is part of the mainline Linux kernel [38]
- the ability to take snapshots and incremental backups of RDBMs: most RDBMS systems support this; some examples:
  - Postgres: via barman [39]
  - MySQL: via mysqldump (and by enabling the binary log)
- the ability to make incremental filesystem backups; this can be done using rsync [37]

5.3.3 Distributed file systems

Distributed file systems share storage using a network protocol, they deliver scalability and failure correction in such a way that it is transparent to the client. Such filesystems can restrict access to certain files based on access lists and/or file system quotas. Distributed file systems allow clients to access files in the same way as they would be able to do on their local file system [2]. The most interesting free software implementations I encountered were GlusterFS [40] and Ceph [41].

5.3.4 Databases

RDBMS sharding

For some application, the amount of data that needs to be stored can be so high that it is impossible to fit it on a single cluster node. In order to accommodate such a large database in a relational context, database sharding can be used. Database sharding allows databases to be partitioned horizontally; so that different rows of the database can reside on different cluster nodes. It is advisable that rows that are closely connected are located on the same cluster node, to improve query performance, however, it still remains possible to execute queries that involve rows distributed over multiple cluster nodes. For example: an application where the data that is stored is mostly private to a user (storage of emails, storage of Skype contacts, ...) could be sharded by the user name.

[2]A client might refer to an end user, but it might as well refer to one of the tiers in a multi-tier system

Document-oriented NoSQL databases

When the data that is stored is private to one user and performing queries over multiple users is a rare event (storage of emails, storage of Skype contacts, ...), it can be interesting to keep a database per user. This can be achieved by using so-called document-oriented NoSQL solutions. These solutions are able to store a document containing all information related to one user. This document can for example be structured using the JSON format. Contrary to RDBMS systems; document-oriented NoSQL solutions have no schemas that define how the data should be structured, instead fields can be freely added to the JSON documents. Although this provides a greater flexibility, this also makes it very difficult to perform queries over multiple documents that require data to be joined together. When data is grouped together in a document per entity (e.g.: a user) and this document is stored in one file; it becomes much easier to scale the data over multiple clusters. With the booming of the “cloud” several document-oriented NoSQL solutions have been developed, most notably: CouchBase and MongoDB.

Chapter 6

Experimenting with HA and scalability solutions

6.1 Open source technologies

In this project, I focus on open source (mainly free software, as in GPL licensed) technologies. In my opinion, open source technologies are ideal to setup large clusters. All the source code of the building blocks of the cluster is available and thus can be reviewed. Each cluster setup can be very specific towards the problem the company or research institution is trying to solve, therefore, being able to adapt the source code to meet specific solutions is definitely an advantage.

6.2 Test environment setup

I will simulate a mini-cluster using different virtual machines. I will create these virtual machines with VirtualBox [42]. The virtual machines will have "Ubuntu Linux 13.10 (AMD64) Server Edition" [43] installed as base operating system.

I want all the virtual machines to be able to contact each other, while it should also be possible for them to access the internet. To make this possible, I opted to use VirtualBox' virtualised NAT feature (this is a new feature, available since VirtualBox 4.3). This new feature emulates a NAT environment on your host machine. The advantage of this is that the test setup will also work when no network is available.

I created a base virtual machine with “Ubuntu Linux 13.10 (AMD64) Server Edition” and the VirtualBox guest additions installed. I will use this base virtual machine as a starting point for all the virtual machines that I will setup to execute experiments.
I will include this base virtual machine in my final deliverable, since all other virtual machines can be derived from it by following the instructions in this essay.
I also created a shared filesystem, this allows me to share files between the different virtual machines and my laptop.
First we need to create the NAT network:

```
VBoxManage natnetwork add -t nat-int-network
                          -n "192.168.15.0/24"
                          -e -h on
```

This commands creates the NAT network 'nat-int-network' with an IP address range between 192.168.15.0/24. After creating the NAT network, virtual machine guests can be configured as shown in fig. 6.1.

6.3 Monitoring

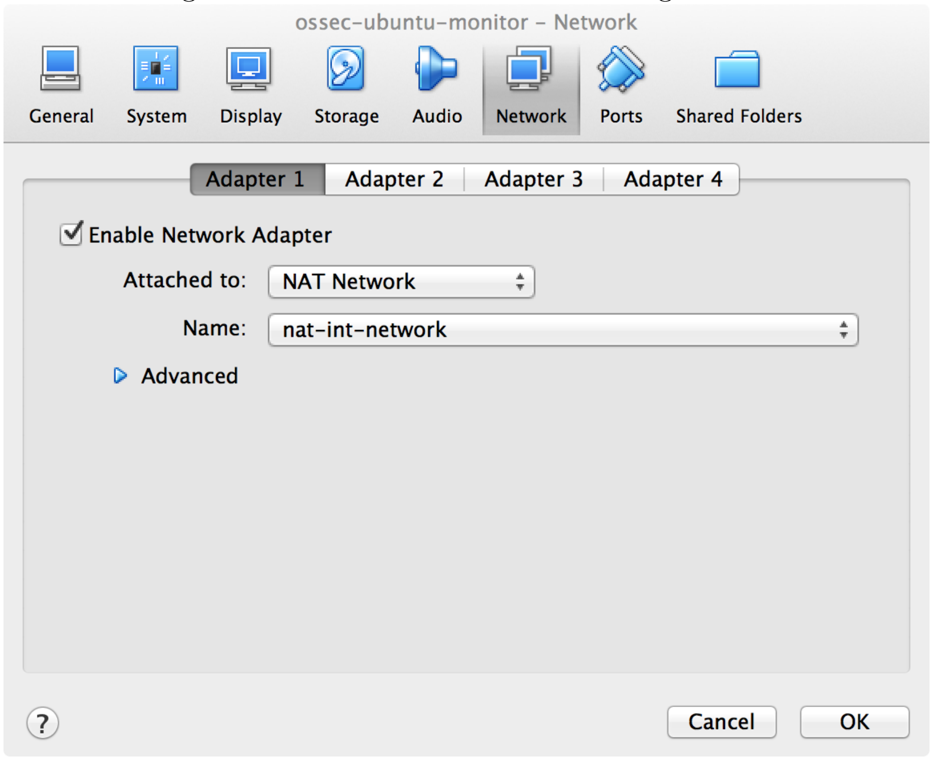
6.3.1 Experiment overview

To test nagios [26], I will test the detection of application and system failure. To test the application failure, I will write an example application that fails after 2 minutes. I will test the system failure by inducing a kernel panic. When an application failure is detected I will execute a script on the cluster node to restart the application. When a system failure is detected, I will restart the virtual machine.

6.3.2 Nagios experiment

I cloned the base virtual machine as described earlier and changed its hostname to 'monitor' (by editing */etc/hostname* and */etc/hosts* and rebooting).
On this machine, I installed nagios:

Figure 6.1: VirtualBox network configuration.



☑ Enable Network Adapter

Attached to: NAT Network

Name: nat-int-network

```
sudo apt-get install nagios3
```

During this installation a couple of questions were asked:

- email configuration: since I will not use this in my experiment I did not provide a configuration
- the nagios web administration password

In order to be able to access the web server, I opened port 80 of the machine's firewall. I first enabled the UFW firewall:

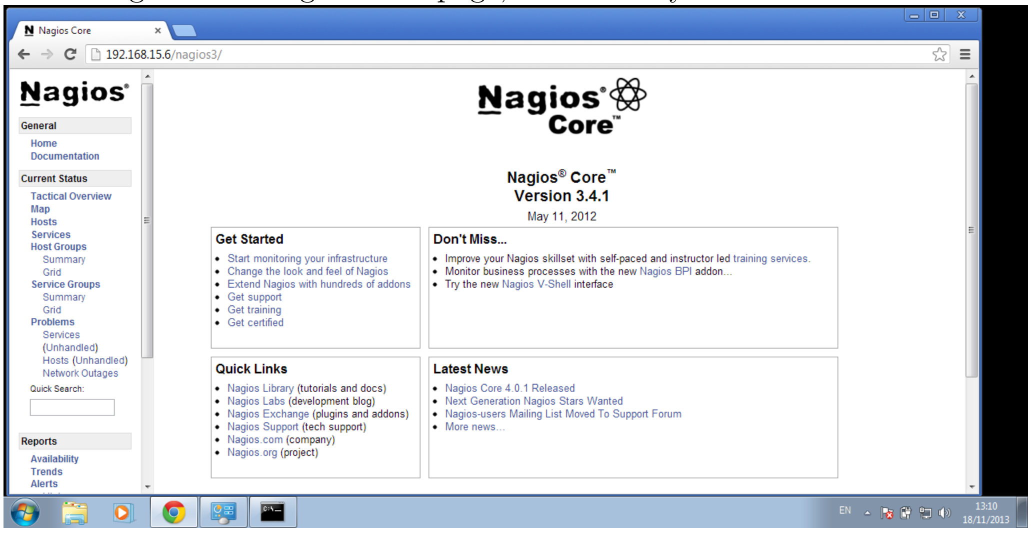
```
sudo ufw enable
```

Then I opened port 80:

```
sudo ufw allow 80/tcp
```

Since the NAT network setup does only allow virtual machines to access other virtual machines, I started my Windows virtual machine and used the web browser to access the nagios system. After providing the username and password, the browser showed the un-configured website as depicted in fig. 6.2.

Figure 6.2: Nagios start page, immediately after installation.



Nagios® Core™
Version 3.4.1
May 11, 2012

Get Started
- Start monitoring your infrastructure
- Change the look and feel of Nagios
- Extend Nagios with hundreds of addons
- Get support
- Get training
- Get certified

Don't Miss...
- Improve your Nagios skillset with self-paced and instructor led training services.
- Monitor business processes with the new Nagios BPI addon...
- Try the new Nagios V-Shell interface

Quick Links
- Nagios Library (tutorials and docs)
- Nagios Labs (development blog)
- Nagios Exchange (plugins and addons)
- Nagios Support (tech support)
- Nagios.com (company)
- Nagios.org (project)

Latest News
- Nagios Core 4.0.1 Released
- Next Generation Nagios Stars Wanted
- Nagios-users Mailing List Moved To Support Forum
- More news...

I cloned another pair of servers: 'node1' and 'node2', that will serve as cluster nodes (I performed the same steps as on 'monitor' to change the hostnames of these nodes).

To test the detection of application failure, I wrote a program that fails after 2 minutes.

The source code of this program can be found in src/nagios/nagios_app.cpp. I install it on 'node1' simply by building the source code:

```
g++ nagios_app.cpp -o nagios_app
```

After building the application, I copy it to */usr/sbin/*. [1]
In order to start/stop and restart the service, I create an init.d configuration script.
I based this script on the code I found on this website:
http://koo.fi/blog/2013/03/09/init-script-for-daemonizing-non-forking-processes/
I included the source code of the nagios_app init.d config script in

[1]In order to build this C++ code, we need gcc, this is included in the package build-essential, which I installed earlier

src/nagios/init.d/nagios_app. Now we still need to give the script the correct permissions:

```
sudo chmod 755 /etc/init.d/nagios_app
```

And we need to include it in the startup list:

```
sudo update-rc.d myscriptname defaults
```

We now can start/stop the program very easily:

```
sudo /etc/init.d/nagios_app start
sudo /etc/init.d/nagios_app stop
```

To monitor a remote server I use the NRPE plugin for Nagios. This can be easily installed on the server using this command:

```
sudo apt-get install nagios-nrpe-plugin
```

To allow nagios to detect the installation of the plugin, we need to restart apache:

```
sudo /etc/init.d/apache2 restart
```

On the server that is to be monitored ('node1'), we need to install the NRPE daemon, so that the Nagios server can communicate with the remote server

```
sudo apt-get install nagios-nrpe-server
```

We can check whether the NRPE plugin can connect to 'node1', with this command:

```
/usr/lib/nagios/plugins/check_nrpe -H 192.168.15.4
```

Upon the first try, I got this error message: "Could not complete SSL handshake".
I was able to solve this problem by adding the server's IP address to the list of allowed hosts in the NPRE config file (/etc/nagios/nrpe.cfg) on 'node1'. After restarting the NRPE daemon on 'node1', the test worked as expected.
Now let's configure 'monitor' to monitor the general health and CPU load of 'node1'.
I added a host and service configuration to the $/etc/nagios3/commands.cfg$ configuration file:

```
define host {
        use             generic-host
        host_name       node1
        address           192.168.15.4
}

define service {
        use                         generic-service
        host_name                   node1
        service_description        CPU Load
        check_command               check_nrpe!check_load
}
```

After making these changes I restarted apache.
I based all the previous actions on the official NRPE documentation [44], unfortunately, this did not result in Nagios adding the service to the list of monitored services. After some research, I found another tutorial [45], this tutorial mentioned I had to perform a reload of the Nagios service:

```
  /etc/init.d/nagios reload
```

This command (followed by a restart of apache), allowed Nagios to detect the new service. However, Nagios was not able to collect any information from this service. I tested the command via the command line:

```
  /usr/lib/nagios/plugins/check_nrpe -H 192.168.15.4 -c check_load
```

The execution of this command resulted in the expected output, so I understood the problem had to be related to the nagios configuration. I searched for the problem, and found a post on stackoverflow.com [46], that mentioned the same problem as I ran into. The solution for this problem was explicitly instructing Nagios to use the *check_nrpe* command that only accepts 1 argument. The problem was solved after changing the configuration file like this:

```
define host {
        use               generic-host
        host_name         node1
        address           192.168.15.4
}

define service {
```

```
          use                        generic-service
          host_name                  node1
          service_description        CPU Load
          check_command              check_nrpe_1arg!check_load
}
```

After reloading Nagios and restarting apache, Nagios correctly fetched data on the CPU load of 'node1' as depicted in fig. 6.3.

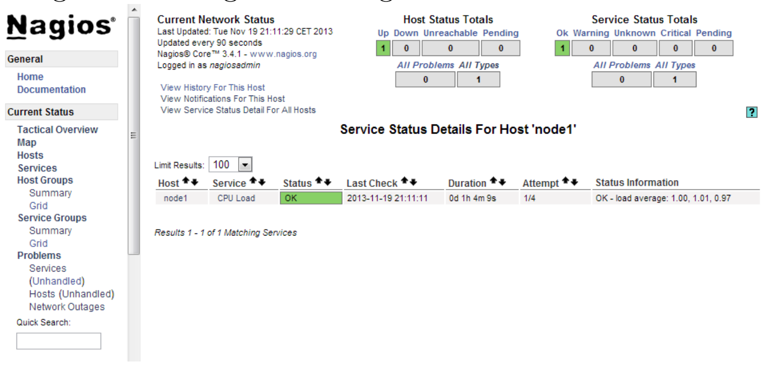


Figure 6.3: Nagios showing the “Check load” service

Current Network Status
Last Updated: Tue Nov 19 21:11:29 CET 2013
Updated every 90 seconds
Nagios® Core™ 3.4.1 - www.nagios.org
Logged in as *nagiosadmin*

Host Status Totals

Up	Down	Unreachable	Pending
1	0	0	0

All Problems	*All Types*
0	1

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
1	0	0	0	0

All Problems	*All Types*
0	1

Service Status Details For Host 'node1'

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
node1	CPU Load	OK	2013-11-19 21:11:11	0d 1h 4m 9s	1/4	OK - load average: 1.00, 1.01, 0.97

Results 1 - 1 of 1 Matching Services

To do some more involved tests, I added a new node (which I called 'node2') to my list of virtual machines. I configured this machine in the same way as I did on 'node1' and added it to the Nagios monitor server, as described earlier. This worked without any problems, and Nagios correctly shows the health and CPU load of both servers fig. 6.4.
Now, let's see how Nagios properly detects system failure. To test this, we will induce a kernel panic on 'node1'; this can be achieved by inserting a kernel module that calls the panic() kernel function in its init function. I found source code to achieve this (this code can be found in src/panic/) [47]. To make the kernel crash, first we need to build the module:

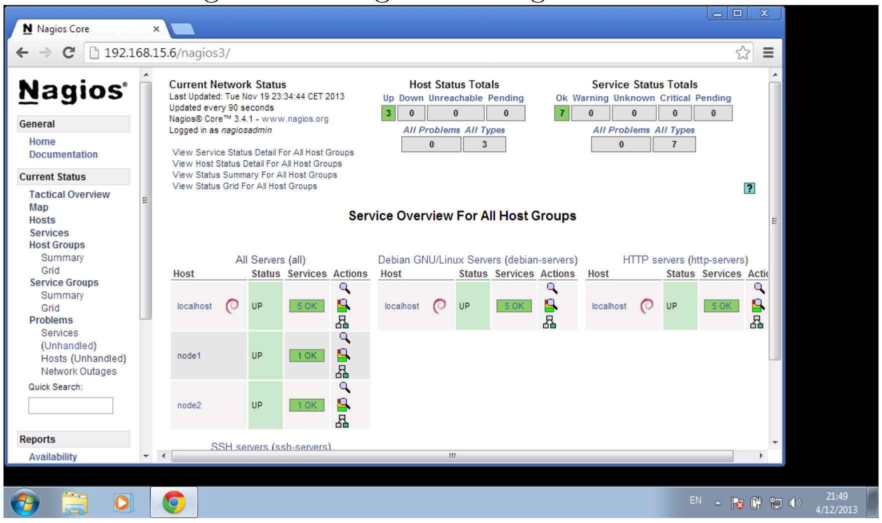
```
make
```

Now we can insert the module into our running kernel:

```
sudo insmod force_panic.ko
```

This immediately causes a kernel panic (see fig. 6.5). After about one minute, this

Figure 6.4: Nagios showing both nodes



Current Network Status
Last Updated: Tue Nov 19 23:34:44 CET 2013
Updated every 90 seconds
Nagios® Core™ 3.4.1 - www.nagios.org
Logged in as *nagiosadmin*

Host Status Totals

Up	Down	Unreachable	Pending
3	0	0	0

All Problems	*All Types*
0	3

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
7	0	0	0	0

All Problems	*All Types*
0	7

Service Overview For All Host Groups

All Servers (all)

Host	Status	Services	Actions
localhost	UP	5 OK	
node1	UP	1 OK	
node2	UP	1 OK	

Debian GNU/Linux Servers (debian-servers)

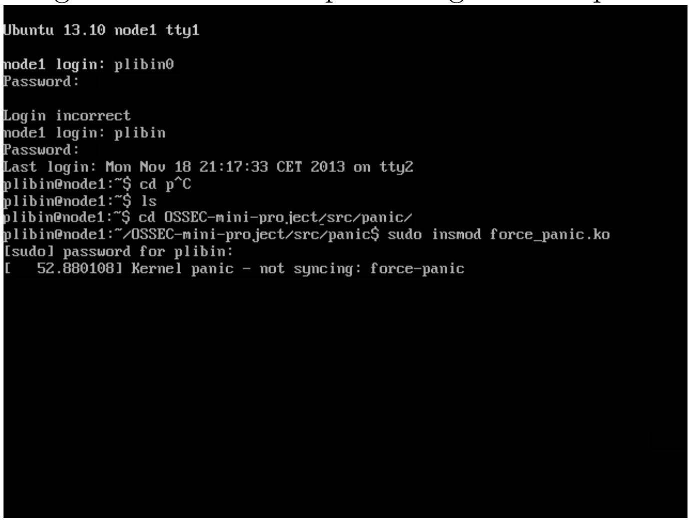
Host	Status	Services	Actions
localhost	UP	5 OK	

HTTP servers (http-servers)

Host	Status	Services	Acti
localhost	UP	5 OK	

SSH servers (ssh-servers)

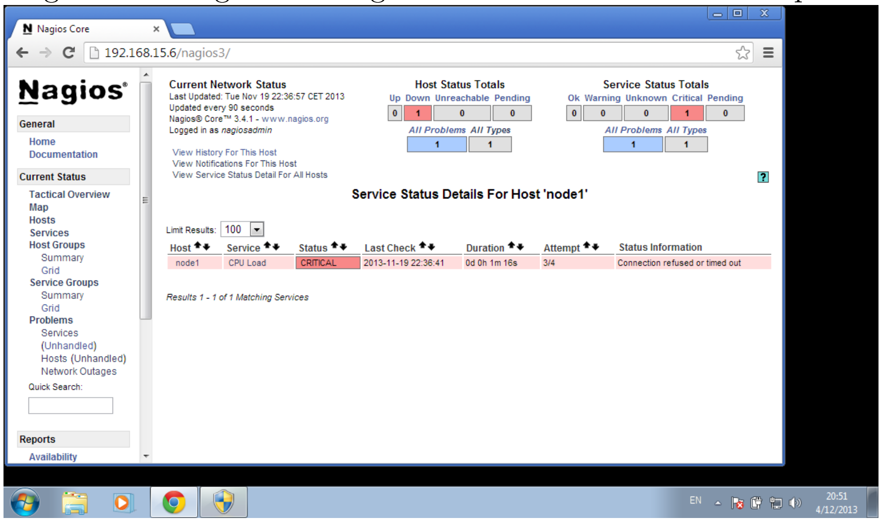
Figure 6.5: 'node1' experiencing a kernel panic



```
Ubuntu 13.10 node1 tty1

node1 login: plibin0
Password:

Login incorrect
node1 login: plibin
Password:
Last login: Mon Nov 18 21:17:33 CET 2013 on tty2
plibin@node1:~$ cd p^C
plibin@node1:~$ ls
plibin@node1:~$ cd OSSEC-mini-project/src/panic/
plibin@node1:~/OSSEC-mini-project/src/panic$ sudo insmod force_panic.ko
[sudo] password for plibin:
[   52.880108] Kernel panic - not syncing: force-panic
```

failure was detected by nagios, and the user interface depicted a "Critical error" (see fig. 6.6).

If a node experiences a system failure, it is useful that an action is executed that tries to solve the problem. An Ubuntu server can be configured to automatically restart upon a kernel panic [48], however, when a hardware problem is the cause of the system crash, restarting the node will be of no avail. It is better to start a new node, and investigate the problem on the crashed node in detail.



Current Network Status
Last Updated: Tue Nov 19 22:36:57 CET 2013
Updated every 90 seconds
Nagios® Core™ 3.4.1 - www.nagios.org
Logged in as *nagiosadmin*

View History For This Host
View Notifications For This Host
View Service Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
0	1	0	0

All Problems	*All Types*
1	1

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
0	0	0	1	0

All Problems	*All Types*
1	1

Service Status Details For Host 'node1'

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
node1	CPU Load	CRITICAL	2013-11-19 22:36:41	0d 0h 1m 16s	3/4	Connection refused or timed out

Results 1 - 1 of 1 Matching Services

Figure 6.6: Nagios showing a critical error after a kernel panic

It is possible to configure an event handler for a Nagios service. This handler will execute a script on the monitor server, every time an exceptional event occurs. For our simulation, we will execute this scenario:

- crash 'node1'
- Nagios will notice this crash and an event handler will be triggered
- this event handler will send a message to the virtual machine controller (in our setup, this is my MacOS X operating system)
- when the controller receives this message, it will start 'node2'

Sending a message will be implemented as writing a file to a shared directory [2]. This directory has the following layout:

```
./nodes/node1
./nodes/node2
```

First let's define an event handler that places a file named 'crash' in the shared directory *./nodes/node1/* whenever Nagios detects a critical error. For this to work we need to configure an event handler connected to the host, we can do this by adding this line to the host definition (*/etc/nagios3/commands.cfg*) of 'node1':

[2]Shared between my MacOS X, 'node1', 'node2' and 'monitor

```
event_handler     inform-controller
```

We now have to define the $inform - controller$ command:

```
define command {
    command_name      inform-controller
    command_line      /soft/nagios/scripts/inform-controller.sh
$HOSTNAME$ $HOSTSTATE$
}
```

The script inform-controller simply creates a file
/media/sf_ossec_vm_share/nodes/node1/crash whenever Nagios detects 'node1' goes down. Find the code for the inform-controller.sh script in src/vm_controller/inform-controller.sh.

Now we need to detect the creation of the file
/media/sf_ossec_vm_share/nodes/node1/crash on the controller, which is a MacOS X system in my testing environment. On MacOS X, this can be implemented using "Folder actions", with some customised scripts [49] it is possible to execute bash scripts when an file event occurs. I configured my MacOS X operating system to run this script whenever a file was added to one of the node directories. This script will check whether a crash file exists in the *./nodes/node1* directory, and if this is the case, start 'node2' using VirtualBox's command line tools [3]
This simple setup shows how Nagios can be used to react on the failure of cluster nodes. Note that an actual setup might be more involved; we would for example keep a pool of available systems that can be started when one of the running cluster nodes fails. In a cluster environment, our VBoxManage call might be replaced with a command to direct a hardware controller to power on an available rack server.
I recorded a video to demonstrate this experiment:
https://www.youtube.com/watch?v=Hwzlyl_FoZM

[3]Source of the script: src/FolderActions.sh

6.3.3 Ganglia experiment

I will use the same set of virtual machines to setup the Ganglia experiment: 'node1', 'node2' and 'monitor'. The 'monitor' node will periodically ask the client nodes for their metrics and store these metrics in a round-robin database. To install the software required for this monitoring (gmetad) we execute the following command:

```
sudo apt-get install ganglia-monitor gmetad
```

These programs only include the monitoring system, in order to visualise the results in a web interface we should also install the web-frontend:

```
sudo apt-get install ganglia-webfronted
```

Ganglia supports 2 modes: multicast and unicast. The multicast is the default mechanism, and is the easiest way to configure a Ganglia cluster. For our test setup, where we have an explicit monitor server, we will configure Ganglia using unicast mode.
Let's start by configuring the 'monitor' cluster node. I adapted the default configuration file that was installed with Ganglia based on an installation manual [50]. Find a listing of the most important changes I've made in
src/ganglia/ganglia_monitor_config

The most important segments of this configuration listing are:

- $deaf = no$: the monitor server should listen to the cluster nodes
- cluster name
- channel (UDP/TCP) configurations

To configure the cluster nodes, we only need to install ganglia-monitor:

```
sudo apt-get install ganglia-monitor
```

I adapted the default configuration file that was installed with Ganglia based on an installation manual [50]. Find a listing of the most important changes I've made in src/ganglia/ganglia_node_config
The most important segments of this configuration listing are:

- *deaf* = *yes*: the cluster nodes should not listen to other cluster nodes, but only send their data to the monitor server

- cluster name

- channel (UDP) configuration

To finalise the configuration, we need to configure gmetad on 'monitor', we can do this by providing gmetad with this config file (located at /etc/ganglia/gmetad.conf):

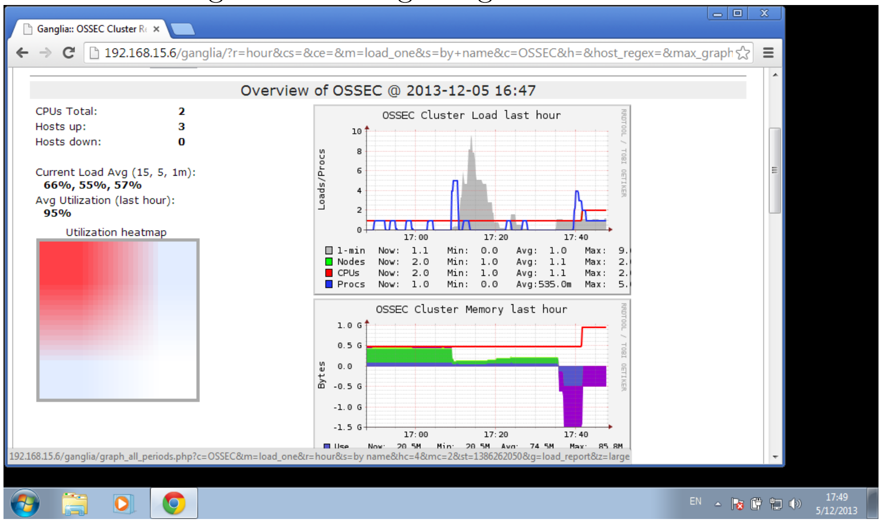
```
data_source "OSSEC" localhost
```

To allow the cluster nodes to send their data to the monitor server, we still need to open the UDP port *gmetad* is listening on:

```
sudo ufw allow 8649/udp
```

OK, let's take it for a spin! When trying to access the web application, there was a problem, Apache was not aware of the ganglia web application. After copying Ganglia's Apache configuration to */etc/apache2/sites – enabled*, the web application was functional and shows a general overview (fig. 6.7). All configured

Figure 6.7: Ganglia's general overview



Ganglia:: OSSEC Cluster R

192.168.15.6/ganglia/?r=hour&cs=&ce=&m=load_one&s=by+name&c=OSSEC&h=&host_regex=&max_graph

Overview of OSSEC @ 2013-12-05 16:47

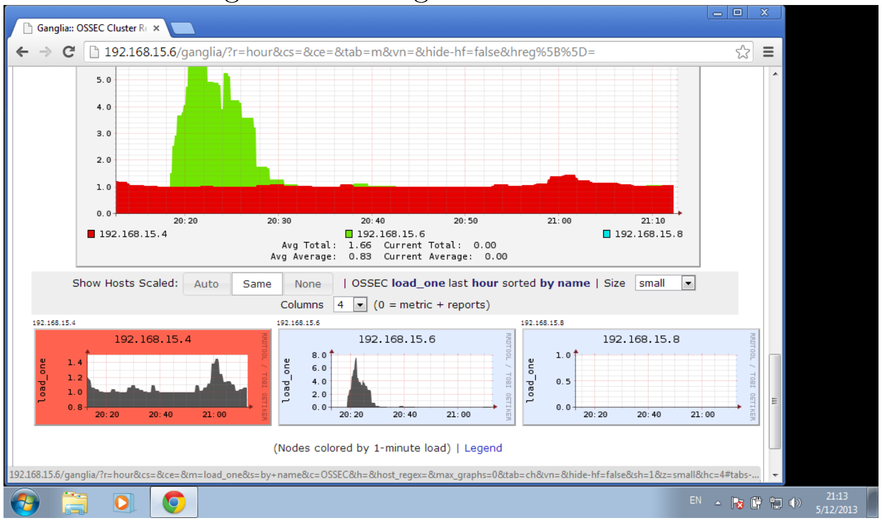
```
CPUs Total:      2
Hosts up:        3
Hosts down:      0

Current Load Avg (15, 5, 1m):
  66%, 55%, 57%
Avg Utilization (last hour):
  95%
```

```
OSSEC Cluster Load last hour
1-min   Now: 1.1   Min: 0.0   Avg: 1.0       Max: 9.
Nodes   Now: 2.0   Min: 1.0   Avg: 1.1       Max: 2.
CPUs    Now: 2.0   Min: 1.0   Avg: 1.1       Max: 2.
Procs   Now: 1.0   Min: 0.0   Avg:535.0m     Max: 5.
```

192.168.15.6/ganglia/graph_all_periods.php?c=OSSEC&m=load_one&r=hour&s=by name&hc=4&mc=2&st=1386262050&g=load_report&z=large

nodes were detected, and an overview of the CPU load of all 3 nodes was visible on the bottom of the Ganglia's web application start page fig. 6.8. I've started nagios_app (reference to the previous section) on 'node1' (IP=192.168.15.4) to put

Figure 6.8: Ganglia's node overview



192.168.15.6/ganglia/?r=hour&cs=&ce=&tab=m&vn=&hide-hf=false&hreg%5B%5D=
192.168.15.4
192.168.15.6
192.168.15.8
Avg Total: 1.66 Current Total: 0.00
Avg Average: 0.83 Current Average: 0.00
Show Hosts Scaled: Auto Same None
OSSEC load_one last hour sorted by name | Size small
Columns 4 (0 = metric + reports)
load_one
(Nodes colored by 1-minute load) | Legend

the node under heavy load. This is visible in the node overview (fig. 6.8).

Since the collected metrics are stored in a round-robin database, it is possible to extract this data to do research on it. Such research can help companies to predict/anticipate sudden changes in system load.

All data is stored in */var/lib/ganglia/rrds/*, for each cluster, there is a directory. In a cluster directory, each node that belongs to the cluster has a directory with its name/IP address. For example: if we want to look at the CPU user-usage of our 'node1', we need to query this file:
*/var/lib/ganglia/rrds/OSSEC/*192.168.15.4*/cpu_user.rrd*.
This file is encoded in a binary format; we can export it to a CSV-file using the rrd2csv program [51]:

```
perl rrd2csv.pl
   /var/lib/ganglia/rrds/OSSEC/192.168.15.4/cpu_user.rrd
   > node1_cpu_user.csv
```

However, this command will only export the data of the last minute. To export all data that was collected starting from a point further in time, we need to explicitly pass the start date in epoch format. An example to extract all data that is one hour old:

```
now = `date +%s`
one_hour_ago = `expr $now - 3600`
perl rrd2csv.pl -start $one_hour_ago
  /var/lib/ganglia/rrds/OSSEC/192.168.15.4/cpu\_user.rrd
  > node1_cpu_user.csv
```

This kind of data (in CSV format) can be imported directly by statistical analysis environments such as R [52].

6.3.4 Nagios and Ganglia: evaluation

Both systems are very interesting and mostly complementary.

Nagios

Nagios has a very powerful component: the event handler .This system makes it possible to react on failures. The software is very extensible and configurable. Some negative remarks:

- the documentation is not always up to date and sometimes incomplete
- the installation was difficult [4]
- the installation required a lot of manual configurations, maybe it would be better that some working default configuration was in place (to give users a working starting point)

Ganglia

The installation was easy; apt-get did almost everything. The configuration that was required to enable unicast was not that difficult to perform (the documentation is in good shape and this helped a lot). Collected data is stored in a round-robin database by default, this is positive, since there exists already a lot of software packages to query/extract/graph data from such databases.

[4]largely because of the bad quality of the documentation

6.4 HAProxy experiment

HAProxy is a prominent software load balancer. It acts like a reverse proxy; pretending it is the main server and then forwarding the requests to one of the HTTP server cluster nodes. Several strategies are available to distribute requests over a HTTP server cluster; these strategies are explained in subsection 5.2.3. In this experiment, I will setup a HAProxy that sends requests to 2 HTTP server cluster nodes using the 'round robin' distribution strategy.

6.4.1 HAProxy installation

I will install HAProxy on a new Ubuntu virtual machine: 'haproxy'. I will use the virtual machines 'node1' and 'node2' as HTTP server cluster nodes. Let's start by properly configuring 'node1' and 'node2' that it can serve this simple PHP program (from [53]):

```
<?php
header('Content-Type: text/plain');
echo "Server IP: ".$_SERVER['SERVER_ADDR'];
echo "\nClient IP: ".$_SERVER['REMOTE_ADDR'];
echo "\nX-Forwarded-for: ".$_SERVER['HTTP_X_FORWARDED_FOR'];
?>
```

This program will print the HTTP server's IP address, the client IP address (which in our case will be the IP address of our HAProxy server) and the X-Forwarded-for header (which will allow us to identify the originating IP address: i.e. the IP-address of our browser). First we'll need to install Apache and PHP:

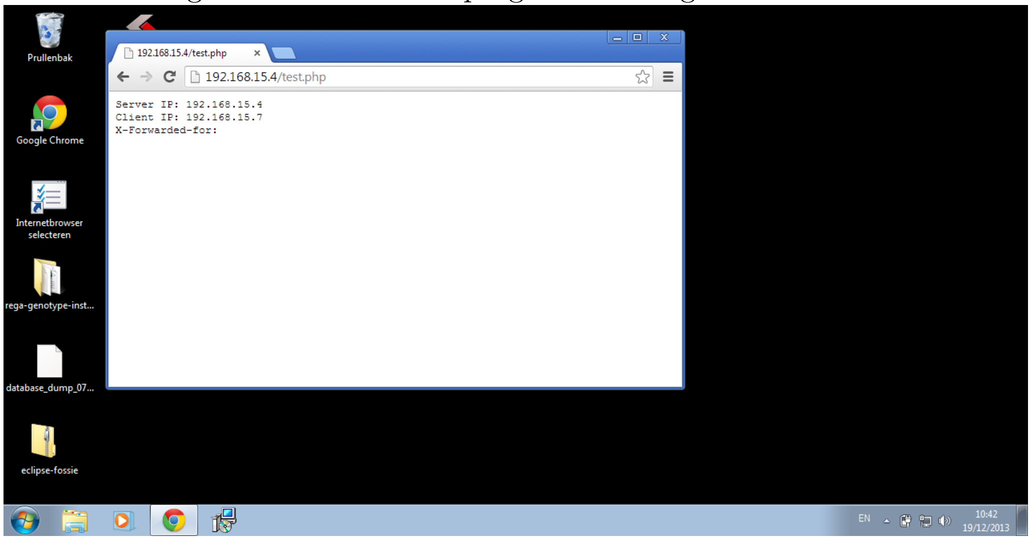
```
sudo apt-get install apache2
sudo apt-get install php5
sudo apt-get install libapache2-mod-php5
sudo /etc/init.d/apache2 restart
```

Now we can copy our test program subsection 6.4.1 to */var/www/test.php*.
Then I opened port 80:

```
  sudo ufw allow 80/tcp
```

After these changes, the nodes were able to serve this PHP test program: fig. 6.9.

Figure 6.9: Test PHP program working in a browser



```
Server IP: 192.168.15.4
Client IP: 192.168.15.7
X-Forwarded-for:
```

To install HAProxy:

```
apt-get install haproxy
```

After this installation, a reboot of the virtual machine was required. In order to make HAProxy start by the init script we need to put the ENABLED variable in */etc/default/haproxy* to 1. The haproxy Ubuntu package comes with a proper config file by default (in src/haproxy/haproxy-basic.cfg). In order to configure HAProxy for this specific application, I conceived this configuration snippet (the full config file can be found in src/haproxy/):

```
listen ossec_app 0.0.0.0:80
        mode http
        balance roundrobin
        option forwardfor
        server node1 192.168.15.4:80 check
        server node2 192.168.15.8:80 check
```

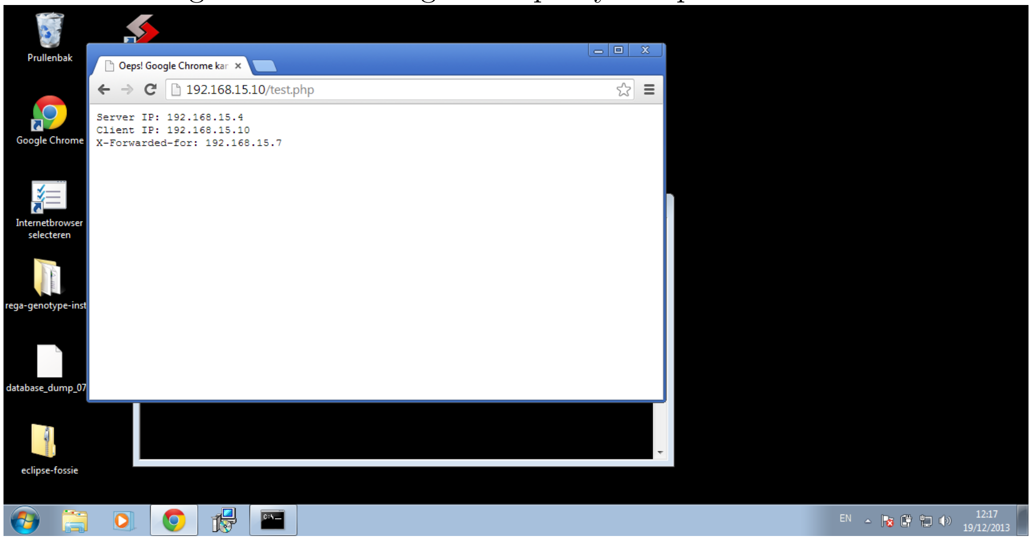
This configuration defines the application ossec_app on port 80. I selected the 'roundrobin' algorithm, enabled the forwardfor (so we can read the X-Forwarded-for HTTP header in our PHP test program) and added our 2 HTTP server cluster nodes to the configuration. Then I opened port 80 on the 'haproxy' server:

```
  sudo ufw allow 80/tcp
```

6.4.2 HAProxy testing

After making the configurations as described in the previous section, I could visit the test PHP program via the haproxy server (IP address: 192.168.15.10); as demonstrated in fig. 6.10.

Figure 6.10: Testing the haproxy setup in a browser



```
Server IP: 192.168.15.4
Client IP: 192.168.15.10
X-Forwarded-for: 192.168.15.7
```

When testing this setup, it becomes clear that HAProxy forwards the request to both cluster nodes (subsequently to 'node1' and than to 'node2').
I recorded a video to demonstrate this experiment:
https://www.youtube.com/watch?v=hrjdLpFLEDM

This setup also handles failure of nodes, if we shutdown apache on one of the nodes, HAProxy will automatically forward the incoming requests to the working node.
I also recorded a video to demonstrate this:
https://www.youtube.com/watch?v=ME9HS_r6nZg

6.4.3 Conclusion and remarks

I was able to extract a lot of valuable information from this online article: [53]. The online documentation of HAProxy [31] is also very good, and explains all configuration items in detail, often with some pointers on when an how to use them. The documentation for example advices not to use the 'leastconn' algorithm for HTTP services, since this algorithm is not well-suited for services that use short-lived sessions. The installation went smoothly on Ubuntu, some manual configurations were necessary, but since the documentation of the project is clear and up-to-date, this did not cause any problems.

6.5 Database replication experiment

I will setup a master-slave replicated MySQL database. I will use the 'haproxy' virtual machine to act as my master database and I will install 2 slave databases on my 'node1' and 'node2' virtual machines. I used the following documentation and tutorials: [54], [55], [56].

6.5.1 Installation

I will start by installing mysql (server and client) on all the involved virtual machines ('haproxy','node1','node2').

```
sudo apt-get install mysql-server mysql-client
```

Let's create a test database on our master-node:

```
mysql -u root -p
*type password*
mysql> CREATE DATABASE ossec;
```

No we can use this database, and create a simple table in it:

```
mysql> USE ossec;
mysql> CREATE TABLE friend (first_name text, last_name text);
```

And we add a friend to it:

```
mysql> INSERT INTO friend (first_name, last_name)
       VALUES ('Eric', 'Northman');
```

The next step is the configuration of the master. Most of the configuration is done in */etc/mysql/my.cnf*. First we need to comment out the bind-address line, this will inform MySQL to listen on all IP addresses rather than only on localhost:

```
#bind-address           = 127.0.0.1
```

MySQL slaves use the logs that a master-node generates to replicate the database. In order to make this work, we need to tell the MySQL master-node where to generate these log files and for which databases the log files need to be generated.

```
log-bin = /var/log/mysql/bin.log
#multiple binlog-do-db lines are allowed
binlog-do-db = ossec
```

We need to specify the server's id as well, since this is our master server, we use $id = 1$:

```
server-id = 1
```

Now we're ready for restarting the database:

```
sudo /etc/init.d/mysql restart
```

We need to give a MySQL user (let call him 'reppy') the permission to replicate our database:

```
mysql -u root -p
*type password*
mysql> GRANT REPLICATIOn SLAVE ON *.* TO 'reppy'@''%'
       IDENTIFIED BY 'reppy_password';
mysql> FLUSH PRIVILEGES;
```

The % in this command allows the user to login from any IP address.

To get the information to configure the slave, we need to execute these commands to select our database 'ossec' and lock all the tables in it:

```
mysql> USE ossec;
mysql> FLUSH TABLES WITH READ LOCK;
```

No we can extract the information we need:

```
mysql> SHOW MASTER STATUS;
```

The output of this command is:

```
+------------------+-----------+---------------+--------------------+
| File              | Position  | Binlog_do_db  | Binlog_ignore_db  |
+-------------------+-----------+---------------+-------------------+
| bin.000003        | 333       | ossec         |                   |
+-------------------+-----------+---------------+-------------------+
1 row in set (0.00 sec)
```

Now we need to dump the current database:

```
mysqldump -u root -p --opt ossec > ossec.sql
```

It is important to unlock the tables again before we continue:

```
mysql> UNLOCK TABLES;
```

To allow the slave nodes to connect to the master database, we need to open up the MySQL port (3306) of the master node:

```
sudo ufw allow 3306/tcp
```

Now let's configure the slaves, we start by creating the 'ossec' database on the slaves:

```
mysql -u root -p
*type password*
mysql> CREATE DATABASE ossec;
```

Now we need to apply the dump we obtained on the master server (I copied the ossec.sql using *scp*):

```
mysql -u root -p ossec < ossec.sql
*type password*
```

Next we configure the server-id and the database that is replicated in */etc/mysql/my.cnf*:

```
#on node 1
server-id = 2
#on node 2
server-id = 3

#same for node 1 and 2
replicate-do-db = ossec
```

After these configurations, we need to restart the MySQL server:

```
sudo /etc/init.d/mysql restart
```

And execute this SQL to let the slave know where to find its master:

```
mysql> CHANGE MASTER TO MASTER_LOG_FILE='bin.000003',
   MASTER_LOG_POS=333, MASTER_HOST='192.168.15.10',
   MASTER_USER='reppy', MASTER_PASSWORD='reppy_password';
```

Now we can start the slave:

```
mysql> START SLAVE;
```

6.5.2 Testing the experiment setup

After this, when a change is made to the master database, these changes are automatically propagated to the slave databases.
I recorded a video to demonstrate this experiment:
https://www.youtube.com/watch?v=hQfTVrYzT3o

Another video show how 2 DRBD nodes find each other when they are started up:
https://www.youtube.com/watch?v=DP923CWyZhM

6.5.3 Conclusion and remarks

Setting up a replicated MySQL environment was not easy, mainly since I had to consult several separate documents ([54], [55], [56]). The main problem seems to be that the way things are configured in MySQL seems to change significantly over time. Several settings I found in different tutorials, were no longer supported on the most recent MySQL package distributed with Ubuntu. On the other hand, if something went wrong or behaved unexpectedly, the output in the log files was quite informative. One important remark is that on a slave node, it is no longer possible to configure the master settings (host, log_pos, log_file, user, password) in the configuration file; you should use the $CHANGEMASTER$... SQL command.

6.6 Software disk replication experiment

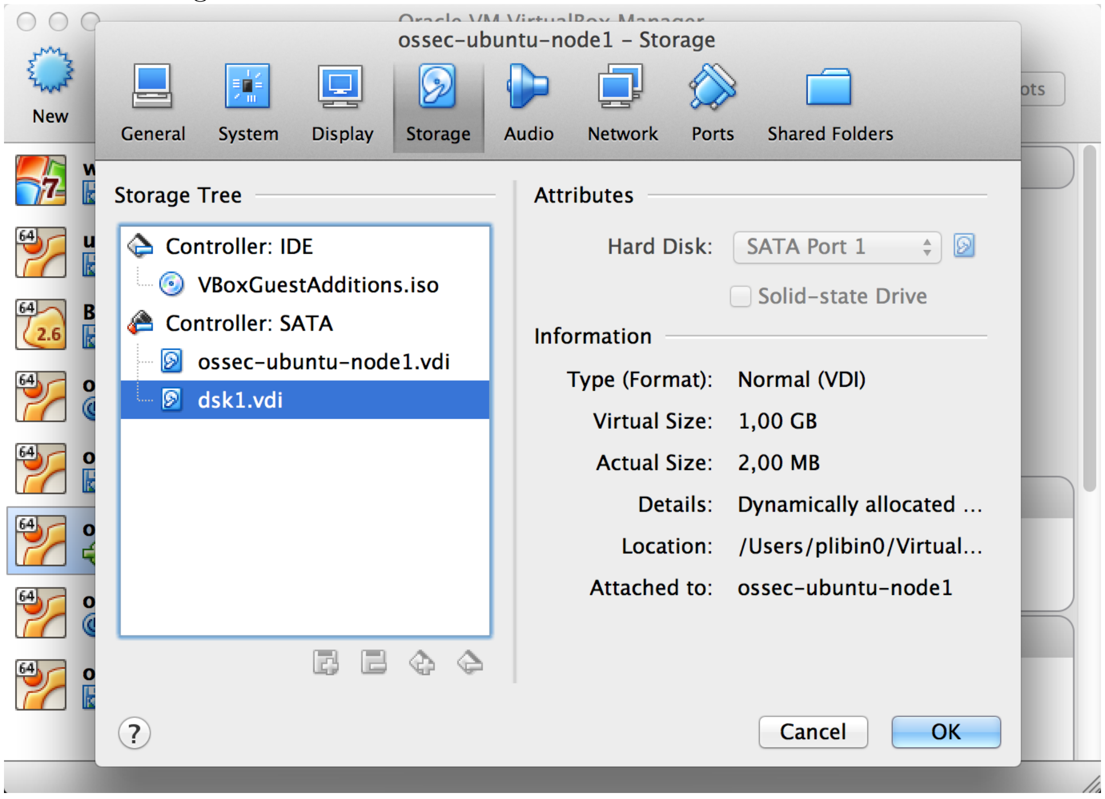
I will setup a network RAID-1 (mirror) system on 2 nodes, for this I will use the software DRBD (Distributed Replicated Block Device) [36]. I will create a disk (virtual machine disk) of 1GB on my virtual machines 'node1' and 'node2' (since we're using RAID, it is vital that these disks have the exact same size).

I used the following documentation and tutorials: [57], [58], [59].

6.6.1 Adding the "disks"

Firstly, I will add a disk to my virtual machines 'node1' and 'node2'. This can be done via the VirtualBox user interface, when the virtual machines are powered off fig. 6.11.

Figure 6.11: Added a disk to one of the virtual machines



ossec-ubuntu-node1 – Storage

Controller: IDE
VBoxGuestAdditions.iso
Controller: SATA
ossec-ubuntu-node1.vdi
dsk1.vdi

Hard Disk: SATA Port 1
Solid-state Drive

Type (Format): Normal (VDI)
Virtual Size: 1,00 GB
Actual Size: 2,00 MB
Details: Dynamically allocated ...
Location: /Users/plibin0/Virtual...
Attached to: ossec-ubuntu-node1

6.6.2 Setting up the network RAID-1

To ensure the synchronisation is timed well, I will install the NTP packages.

```
sudo apt-get install ntp ntpdate
```

In order for DRBD to work, each node should be able to access the other node by name. Since we have no DNS in place, we need to configure this in the */etc/hosts* config file.
We added a disk to the virtual machine, however, this disk is currently an unpartitioned and unformatted device. Let's first have a look at our device:

```
sudo fdisk -l /dev/sdb
```

This command show the following report:

```
Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

To create a partition:

```
sudo fdisk /dev/sdb
Command (m for help): n #create a new partition
Select (default p): p #primary partition
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048): 2048
Last sector (2048-2097151, default 2097151): 2097151
```

The partition is created, but we still need to write the partition table to the disk:

```
Command (m for help): w #write the partition table to disk
```

When running $fdisk - l$ again, we get this output:

```
Disk /dev/sdb: 1073 MB, 1073741824 bytes
139 heads, 8 sectors/track, 1885 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
```

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x1053cb0b

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     2097151     1047552   83  Linux
```

Now we need to install the DRBD utilities:

```
sudo apt-get install drbd8-utils
```

And load the DRBD kernel module:

```
sudo modprobe drbd
```

We can configure these nodes with the following configuration file (based on: [57] [58]):

```
global {
        #do NOT participate in DRBD's online usage counter:
        #explicitely disable this, and also disable that we are
        #asked for this on every restart of the drbd daemon
        usage-count no;
}
resource r0 {
        net {
                #one of the 3 possible protocols: A, B, C
                #see documentation for details
                protocol C;
                #specifies the HMAC algorithm (see Appendix)
                #to enable peer authentication
                cram-hmac-alg sha1;
                #secret used in peer authentication
                shared-secret "testing";
        }
        on node1 {
                volume 0 {
                        #the new device we configure
                        device    /dev/drbd0;
                disk        /dev/sdb1;
                meta-disk  internal;
            }
            address     192.168.15.4:7789;
        }
        on node2 {
            volume 0 {
                #the new device we configure
                device      /dev/drbd0;
                disk        /dev/sdb1;
                meta-disk  internal;
            }
            address     192.168.15.8:7789;
        }
}
```

Most of the configuration items are straightforward, I commented the other elements to clarify their meaning. The configuration file should be placed in */etc/drbd.conf*. We need to open the ports 7789 on both nodes:

```
sudo ufw allow 7789/tcp
```

We also need to restart the drbd daemon:

```
sudo /etc/init.d/drbd restart
```

First I started the DRBD daemon on 'node1', this daemon waited until the BRBD daemon on 'node2' came online.
Now we configured the mirrored device */dev/drbd*0, but we can't use this device yet. We first need to initialise the meta data storage of each physical device, one both nodes, we need to run this command:

```
sudo drbdadm create-md r0
```

We also need to decide which node will be the primary node; on this node we will be able to read and write data d. For now, I will make 'node1' the primary node for this device. First we need to invalidate the data on 'node2', so BRBD knows this data can safely be overiden; we use this command:

```
sudo drbdadm invalidate r0
```

To make the device on 'node1' the primary device, we need to run this command on 'node1':

```
sudo drbdadm --force primary r0
```

Now the device on 'node1' gets automatically synched with the one on 'node2'. But there doesn't exist a filesystem on our device yet; let's create an ext3-filesystem (this command is executed no 'node1'):

```
sudo mkfs.ext3 /dev/drbd0
```

Now we can mount the filesystem (on 'node1'):

```
sudo mkdir /media/drdb0
sudo mount /dev/drbd0 /media/drbd0
```

To test that the replication actually works, we need to:

- create some files on */media/drbd*0 on 'node1'
- unmount the */media/drbd*0 on 'node1'
- make 'node1' the secondary host
- make 'node2' the primary host
- mount the device on 'node2'
- check that the files we placed on the device from 'node1' also appear on 'node2'

I recorded a video to demonstrate this experiment:
https://www.youtube.com/watch?v=oMCyT1wCPmA

6.6.3 Conclusions and remarks

The drdb service can send statistics (anonymized) to a server of the DRBD organisation (to get a better overview about who is using the software). This option is however enabled by default, in my opinion, this should be disabled by default [5].
The online documentation on how to configure the software is up-to-date and helpful. It is nice and interesting software, that easily allows you to setup replication between data centers on block level.

[5]To disable this, we need to add the line "usage-count no;" to the config file's global section.

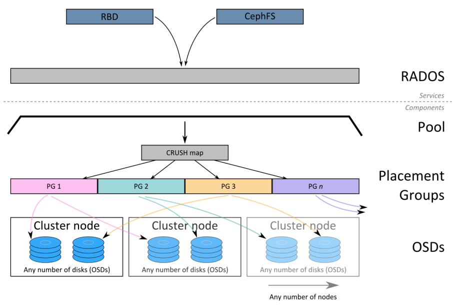
6.7 Distributed file system experiment

I will experiment with Ceph, since this seems to be the de-facto standard distributed filesystem in the Ubutu landscape. Except for better compatibility with Ubuntu, there is no real reason to choose Ceph over GlusterFS; both seem to be decent projects [60], [61]. Ceph is a large project, with many features: object storage, file system, block device layer, ... In this experiment, I will focus on the distributed object storage component.

6.7.1 Architecture

Ceph has a layered architecture as demonstrated in fig. 6.12 [62].

Figure 6.12: Ceph archtitecture



The RADOS layer allows for client software to use the cluster infrastructure (file systems, key value stores,...). This layer communicates with the cluster pool via the CRUSH layer; which distributes objects over the different nodes in the cluster, and makes sure that they can be fetched optimally.
The architecture allows for easy scaling since nodes can be added and removed while the system is running. When such an event occurs the CRUSH layer will automatically rebalance the objects over the cluster.
Reference: [63].

6.7.2 Installation

To perform this installation I used the official Ceph documentation [64]. Ceph can be installed in 2 ways: via ceph-deploy (as piece of software that is installed on an administrator node that automatically installs the necessary components on the nodes in your Ceph cluster) and a manual installation. The advantages of the first approach:

- it's quicker to setup
- commands entered on the administrative node are propagated to all relevant cluster nodes (this makes upgrading nodes much easier)

The main advantage of a manual installation is that it allows you to fully control where and when all necessary components are installed on the different nodes. This might be interesting on a production cluster, however I think the installation via ceph-deploy is more appropriate for this experiment, since it will allow me to test more of Ceph's features in a shorter timespan.

Nodes

We will first setup a cluster with 1 administrator node and a Ceph Storage Cluster consisting of 3 cluster nodes (this setting is advised by the official documentation to experiment with the software [64]). For this, I create 4 new clones of my base virtual machine: 'ceph-admin', 'ceph-node1', 'ceph-node2', 'ceph-node3'.

Preparation of the cluster nodes

Ceph requires some basic stuff to be in place, in order to be able to use ceph-deploy to configure the cluster and cluster nodes. The instructions are well documented in the official documentation's "Ceph node setup" section [64]. A quick overview of the different steps:

- create a 'ceph' user on each node
- add root privileges to the 'ceph' user
- install an ssh server
- configure the admin node so that it can access the cluster nodes without using a password (using SSH keys)

- all nodes should be able to access each other using hostname resolution, in my case, we had to update the */etc/hosts* file to make this possible, since we have no DNS server in our lab setting

Configuring the ceph-admin node

The documentation requests to add the ceph-deploy package to your local APT repository, however on Ubuntu 13.10, this package was already available by default. So we can install it like this:

```
sudo apt-get install ceph-deploy
```

We also need to create a directory to maintain the configuration of our cluster (the name is not important, but we will use /ossec-cluster consistently through the example for clarity):

```
mkdir ~/ossec-cluster
```

It is important that we run the ceph-deploy command in this directory, since it will export its output to the current directory.

Setting up the cluster

Now we can start configuring our cluster [6]. We will configure 'ceph-node1' as a monitor node, 'ceph-node2' and 'ceph-node-3' will be object storage daemons (see subsection 6.7.1).
To start a new cluster with monitor node 'ceph-node1':

```
cd ~/ossec-cluster
ceph-deploy new ceph-node1
```

After executing these commands, a set of files will be created in /ossec-cluster.
To install ceph on our nodes:

```
ceph-deploy install ceph-node1 ceph-node2 ceph-node3
```

Every Ceph cluster is required to have at least one monitor [7] :

[6]The following steps are executed on the 'ceph-admin'

[7]Multiple monitor nodes are also supported; this is vital for a high availability production cluster, but outside the scope of this experiment

```
ceph-deploy mon create ceph-node1
```

To be able to add Object Storage Daemons, we need to gather the monitor node's keys:

```
ceph-deploy gatherkeys ceph-node1
```

To add the Object Storage Daemons (OSDs), we first need to make a uniquely named directory on the nodes. I created /opt/OSD-disk0 on ceph-node2 and /opt/OSD-disk1 on ceph-node3. Now we can prepare and activate the OSDs:

```
ceph-deploy osd prepare ceph-node2:/opt/OSD-disk0
  ceph-node3:/opt/OSD-disk1
ceph-deploy osd activate ceph-node2:/opt/OSD-disk0
  ceph-node3:/opt/OSD-disk1
```

In order to use the ceph command line tool without the need to specify the monitor address, we need to copy the configuration and admin key to all nodes. This goes as advertised in the documentation for the cluster nodes:

```
ceph-deploy admin ceph-node1 ceph-node2 ceph-node3
```

However, this does not work for the administrator node; apparently this is caused by a bug [65] that results in the fact that no config file is written to /etc/ceph. We can circumvent this problem by first copying the config files that reside in /ossec-cluster to /etc/ceph and than running "ceph-deploy admin". In order to make sure the correct config file is written to /etc/ceph, we need to specify the option "–overwrite-conf". This full set of commands:

```
sudo cp -R ~/ossec-cluster /etc/ceph
ceph-deploy admin --overwrite-conf ceph-admin
```

Testing our cluster

Now, we have setup the cluster properly, we can control the cluster with the "ceph" command, to install this command run:

```
sudo apt-get install ceph-common
```

First let's check the status of our cluster:

```
ceph health
```

Executing this command should return “HEALTH_OK”.
I recorded a short video that shows the different cluster nodes and lists the monitor and OSD nodes using the ’ceph’ command:
https://www.youtube.com/watch?v=HjfILFBWRwA

To store and retrieve an object to our cluster, we can do this with the rados tool (see subsection 6.7.1). This tool allows us to read/write objects respectively from/to a pool. Pools can be created and removed with the rados tool, when setting up the cluster, three pools are created automatically: data, metadata, rbd. To store a file to the default data pool, we can use the following commands:

```
echo "Dear Diary, today I played with Ceph..." > diary.txt
rados put diary-text-obj diary.txt --pool=data
```

To view what is in the data pool, we can use this command:

```
rados --pool=data ls
```

We can also request the “physical” location of our file, the so-called placement group (see subsection 6.7.1):

```
ceph osd map data diary-text-obj
```

Note that this location may change as our cluster’s design changes (OSD’s that are added, or removed). To get our precious diary back:

```
rados get diary-text-obj -o diary.txt --pool=data
```

I recorded a video that demonstrates these commands:
https://www.youtube.com/watch?v=1WV9o9CoIIM

Adding OSDs on the fly

It is possible to add OSDs on the fly, to increase capacity for example. When an OSD is added the cluster will automatically be rebalanced. Adding OSDs is done in the same way as I approached earlier, to add a OSD on ’ceph-node1’ we need to execute this code:

```
ceph-deploy osd prepare ceph-node1:/opt/OSD-disk2
ceph-deploy osd activate ceph-node1:/opt/OSD-disk2
```

6.7.3 Conclusion and remarks

The setup that is advised upon in the official documentation asks to create a user with root permissions to control the cluster nodes; security wise this might not be the best default to present in a tutorial.
I was quite impressed with how easy it was to use “ceph-deploy to install and configure the cluster.
Because I was interested in why this installation and configuration was so easy to run over multiple cluster nodes, I read some more on this. I found out there is another software that allows for software to be easily deployed and configured on clusters with a large amount of nodes: CHEF [66]. Unfortunately I did not have the time to look into these software any more, but I think they are very interesting to experiment with.
Because of the bug I ran into [65], I had to redo the installation quite a few times, to figure out a solution for the problem. It is comforting to know that you can undo all installation residues with the following commands:

```
sudo cp -R ~/ossec-cluster /etc/ceph
ceph-deploy purge ceph-admin ceph-node1 ceph-node2 ceph-node3
ceph-deploy purgedata ceph-admin ceph-node1 ceph-node2 ceph-node3
ceph-deploy forgetkeys ceph-admin ceph-node1 ceph-node2 ceph-node3
```

Note that the purge command is not mentioned in the documentation, but this is important as well! Of course it would be nice that the installation procedure did not have any bugs at all; however knowing that you can safely screw things up and return to a clean slate, is quite comforting.

Summary

Ceph is a very powerful and useful piece of software, that is remarkably easy to setup.
It has a clean architecture with different, well-separated layers.
There is the ability to configure what is important: redundancy, read speed, write speed or a combination of these (albeit with an implication on the amount of hardware that will be necessary).

The system also allows you to make snapshots of the entire system or parts of the system.

6.8 Database shards experiment

Horizontal scaling in RDBMs is quite involved and their are different aspects to it: managing shards, propagating queries to shards, inserting data into the appropriate shards. I did some research into existing solutions for MySQL and Postgres, but I was not able to find a solution that properly covers all the aforementioned aspects and that would fit in the scope of this experiment. Some of the solutions that I picked up that looked quite interesting:

- vitesse cite
- cubrid
- MySQL NDB cluster
- pgPool

For this experiment, I will setup a cluster with 2 MySQL servers (shards), each of these servers will have a database with the exact same schema. I will then write a program that inserts new data in the appropriate shard and write a query to fetch data appropriately. Handling sharding on application level, or in a layer where the application is build upon, is a common practice [67].

6.8.1 Setting up the shards

I created 2 clones of my base Ubuntu virtual machine: 'msql-shard1' and 'mysql-shard2'. On both of these virtual machines I installed MySQL:

```
sudo apt-get install mysql-server mysql-client
```

We also need to open up the firewall on port 3306:

```
sudo ufw allow 3306/tcp
```

6.8.2 The database schema

I will use this simple database schema to represent a user in my application:

```
CREATE TABLE user (
  name VARCHAR(100) UNIQUE NOT NULL,
 PRIMARY KEY (name)
);
CREATE TABLE post (
  id INT NOT NULL AUTO_INCREMENT,
  message TEXT NOT NULL,
  name VARCHAR(100) NOT NULL,
 PRIMARY KEY (id),
 FOREIGN KEY (name) REFERENCES user(name)
);
```

We have a “user” entity and each “user” has 0 or more “post”s. Our database schema allows us to cleanly partition our dataset (based on the user entity). A user row is stored on a specific shard, and all this user’s posts are stored on the same shard.
We determine which shard a user should be put in based on the hash of the user’s name.
Let’s create a new database on both nodes:

```
mysql -u root -p
*type password*
mysql> CREATE DATABASE ossec;
```

Now we can use this database, and create our user table:

```
mysql -u plibin -p ossec < mysql_shards_schema.sql
```

To create a dedicated user ’plibin’ and grant this user access to our newly created database (from all the IP address in my local (VirtualBox) network), we need these commands:

```
mysql> CREATE user 'plibin'@'192.168.15.%'
       IDENTIFIED BY 'plibin';
mysql> GRANT ALL privileges on ossec.*
```

```
        TO 'plibin'@'192.168.15.%'
        IDENTIFIED BY 'plibin';
```

It is import to also pass the password to the *GRANTALL* statement. If not, the specified user no longer has to provide a password!

6.8.3 The application

I wrote functions to allow for the following functionality:

- insert a new user in the correct shard
- post a message (as a user)
- query for users that posted a message with a certain string pattern
- get the posts of a specific user

I grouped the aforementioned function in a command line program. All source code for this program can be found in src/shard/java.
I ran the program on 'node1', in order to do this, I had to install Java [68] and the ant build software [69]:

```
apt-get install openjdk-7-jdk
apt-get install ant
```

I also added the IP addresses of 'shard1' and 'shard2' to 'node1's /etc/hosts (since my program relies on hostnames, rather than IP addresses).
The Java project that I created to write this program includes a ant build file which can be used to create a JAR-file. This JAR-file can be executed using the Java runtime.
To build the source code and generate a JAR-file:

```
cd shards/java
ant
```

I'll give an example how to execute the different functions of the program.
To add a user:

```
java -jar dist/shard-test.jar add-user plibin
```

To post a message as a user:

```
java -jar dist/shard-test.jar add-user plibin
```

To list all posts made by a user:

```
java -jar dist/shard-test.jar get-posts plibin
```

To list all posts that contain a certain pattern:

```
java -jar dist/shard-test.jar users-that-posted-pattern plibin
```

I recorded two videos to demonstrate this experiment:

- https://www.youtube.com/watch?v=XDcDdGzdWaA
- https://www.youtube.com/watch?v=-DLZnS6ykg

6.8.4 Conclusion and remarks

Setting up a sharded database environment is quite involved; for this experiment I only looked at the application side of the problem. There are much more aspects to it: rebalancing your shards, adding new shards, ... The application that I conceived is a small experiment, but it clearly shows some important aspects of handling shards from an application point of view as well as from the database instance point of view. The nice thing about this example is that several instances of the application could run simultaneous without any modifications of the program.
Implementing parallel queries (parallel in a sense that the query is executed simultaneous on the different queries) could be done without much effort.

6.8.5 Additional thoughts

Libraries for application sharding

There also exist libraries that make this kind of application sharding easier, some examples are: HiveDB [70] and Hibernate [71].

Relational database on top of distributed object storage system

By doing research into this field, I came to understand that horizontal scaling RDBMs is involved and complicated. I can only imagine that things get much more complicated when you have to deal with huge amounts of data.
Because of this, I started wondering whether it would not be possible to scale the datastore (using a distributed object storage system) rather than the number of database servers. I was not able to find such solutions yet, but it is also an idea that is suggested on a page on the Ceph website [72].

NoSQL database

A natural solution for the kind of problems sharding tries to solved can be found in NoSQL databases. Several solutions exist and I already mentioned some of them in one of the previous sections (see subsection 5.3.4). I had hoped to investigate one of these systems in this project, but unfortunately, I did not have the time to further look into this. However, this is certainly a technology that I want to investigate further.

Chapter 7

Creating a 'Wikpedia clone' cluster

Wikipedia is a highly popular free online encyclopaedia, free as in "free beer", but also and maybe more importantly as in "free speech" (text is contributed to wikipedia under the GNU Free Documentation License [73] and Creative Commons Attribution-ShareAlike License [74]).
It is a collaborative platform: everyone is able to edit any content.
In this chapter I will try to figure out an architecture that fits the Wikipedia platform in terms of high-availability and scalability. I will focus my work on the handling of text articles, rather than multimedia content.

7.1 Functional analysis

There are 2 main features to Wikipedia:

- editing content or adding new content
- viewing content

Content is submitted to Wikipedia in wikitext format [75], the wikitext language is a lightweight markup language. To show this content to the user in a web browser, the wikitext needs to be converted to HTML. Content can contain internal links (Wikipedia's MediaWiki software calls these free-links). An internal link refers to a page within the wiki's domain, and is expressed in specific syntax, for MediaWiki this syntax is [[*page_name*]]. For example: the Wikipedia page of "Lisbon" contains an internal link to "Portugal" as can be seen in fig. 7.1.

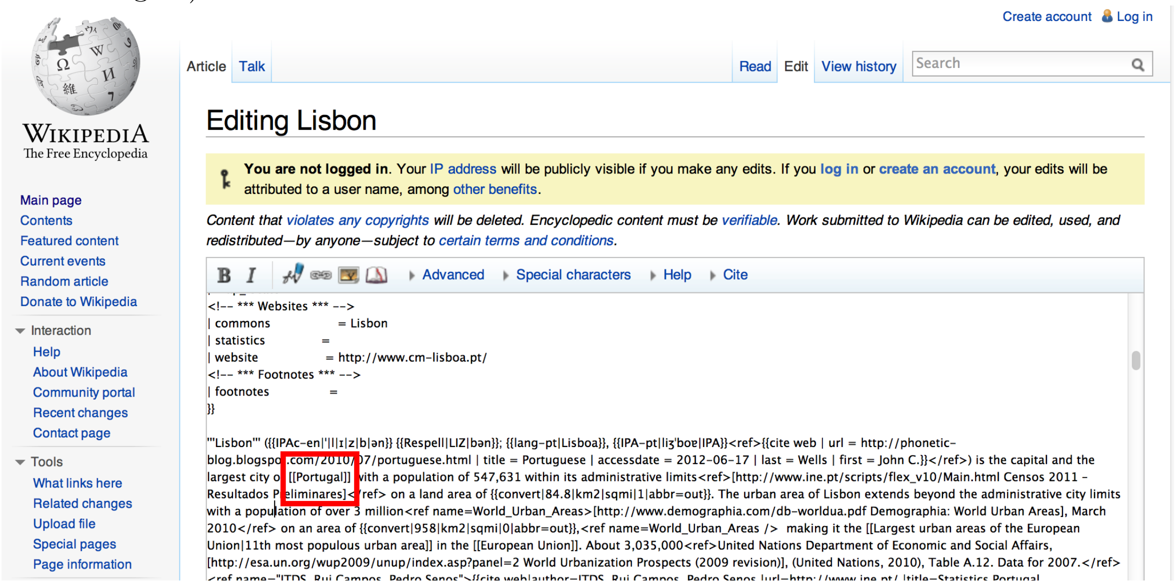


Figure 7.1: Wikipedia page of "Lisbon" in edit mode (annotated in red: a free-link to "Portugal")

Editing Lisbon

You are not logged in. Your IP address will be publicly visible if you make any edits. If you log in or create an account, your edits will be attributed to a user name, among other benefits.

Content that violates any copyrights will be deleted. Encyclopedic content must be verifiable. Work submitted to Wikipedia can be edited, used, and redistributed—by anyone—subject to certain terms and conditions.

```
<!-- *** Websites *** -->
| commons              = Lisbon
| statistics           =
| website              = http://www.cm-lisboa.pt/
<!-- *** Footnotes *** -->
| footnotes            =
}}

'''Lisbon''' ({{IPAc-en|'|l|ɪ|z|b|ən}} {{Respell|LIZ|bən}}; {{lang-pt|Lisboa}}, {{IPA-pt|liʒ'boɐ|IPA}}<ref>{{cite web | url = http://phonetic-
blog.blogspo .com/2010/ 7/portuguese.html | title = Portuguese | accessdate = 2012-06-17 | last = Wells | first = John C.}}</ref>) is the capital and the
largest city o [[Portugal]] vith a population of 547,631 within its administrative limits<ref>[http://www.ine.pt/scripts/flex_v10/Main.html Censos 2011 -
Resultados P eliminares]< ref> on a land area of {{convert|84.8|km2|sqmi|1|abbr=out}}. The urban area of Lisbon extends beyond the administrative city limits
with a population of over 3 million<ref name=World_Urban_Areas>[http://www.demographia.com/db-worldua.pdf Demographia: World Urban Areas], March
2010</ref> on an area of {{convert|958|km2|sqmi|0|abbr=out}},<ref name=World_Urban_Areas />  making it the [[Largest urban areas of the European
Union|11th most populous urban area]] in the [[European Union]]. About 3,035,000<ref>United Nations Department of Economic and Social Affairs,
[http://esa.un.org/wup2009/unup/index.asp?panel=2 World Urbanization Prospects (2009 revision)], (United Nations, 2010), Table A.12. Data for 2007.</ref>
<ref name="ITDS, Rui Campos, Pedro Senos">{{cite web|author=ITDS, Rui Campos, Pedro Senos |url=http://www.ine.pt/ |title=Statistics Portugal
```

Since everyone can edit any content, the submission of incorrect content and vandalism can be expected. In order to keep this under control, it is important that content is versioned. This way, when incorrect content or vandalism is discovered, the version containing the problematic content can simply be removed.

7.2 Some Wikipedia statistics

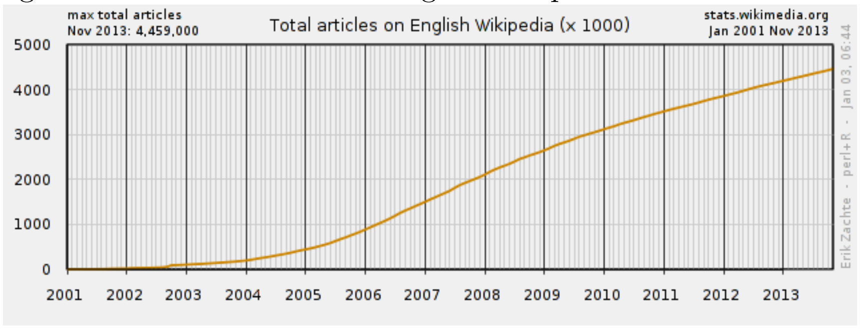
To get a better idea about the needs of the Wikipedia platform in terms of high-availability and scalability, let's take a look at some statistics about Wikpedia.

7.2.1 Presenting the statistics

The Wikimedia foundation publishes statistics about all of their products [76]. The general overview of Wikipedia statistics [77] shows that the English Wikpedia is the largest instance of all Wikipedia. Looking at the statistics for this instance

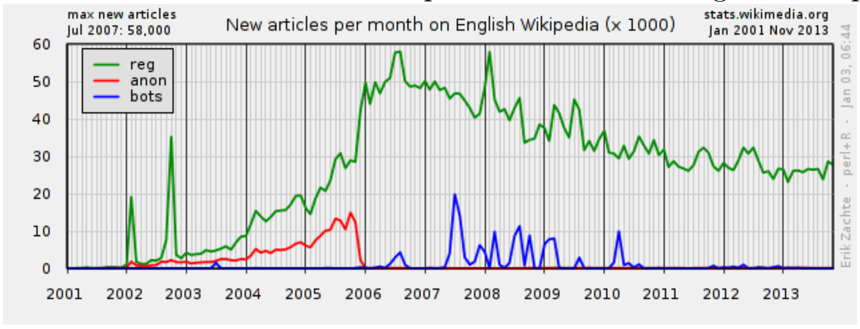
will be relevant to make assumptions in terms of high-availability and scalability. A summary of statistics is available for every Wikipedia instance. We'll walk through the statistics summary of the English Wikipedia instance, note that the presented charts are copies of the Wikipedia statistics website [77], downloaded January 7th 2014.
The total number of articles seems to increase linearly (see fig. 7.2). The amount

Figure 7.2: Total number of English wikipedia articles over time



of new articles per month seems to stay pretty much the same over the last 5 years, however, there were some peaks earlier in the history of Wikipedia (see fig. 7.2). The number of editors stays quite stable since the beginning of 2007 for frequent

Figure 7.3: Number of new articles per month on the English Wikipedia

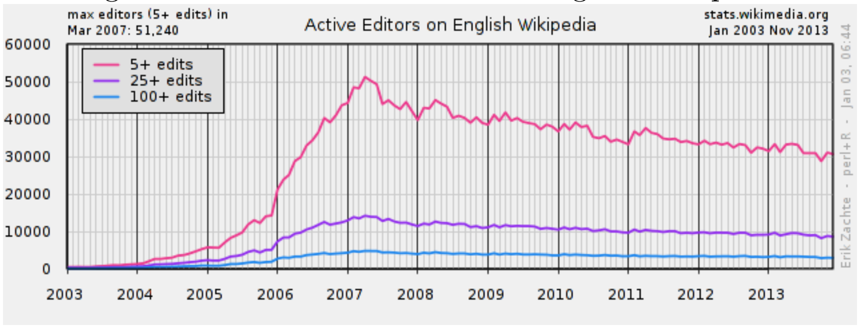


and very frequent editors. Since 2008, the number of infrequent also has stabilised (see fig. 7.4). Between 2005 and 2007 there was a steep growth (see fig. 7.4), but this growth was not that spectacular in absolute numbers:

- 5+ edits: from ≈ 5000 editors to ≈ 45000 editors
- 25+ edits: from ≈ 2000 editors to ≈ 13000 editors

- 100+ edits: from ≈ 1000 editors to ≈ 5000 editors

Figure 7.4: Active editors on the English Wikipedia

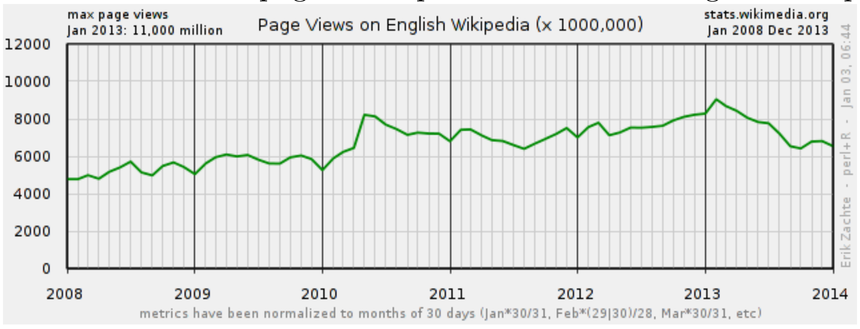


The average number of edits per month is reported as 2, 927, 878 edits. The chart that summarises the number of page views per month shows a steady growth (see fig. 7.5). The growth in absolute numbers is significant:

- first quarter 2008: ≈ 4500 million
- first quarter 2013: ≈ 8400 million

I was not able to find any statistics about page view count before 2008, but since Wikipedia only started in 2001, with only a significant amount of articles since 2005 (see fig. 7.2), we can assume that there was a steep growth between 2005 and 2008. There were statistics about the growth of the Wikipedia database [78], but

Figure 7.5: Number of page views per month on the English Wikipedia



these statistics seem to be discontinued. As to be expected, this database keeps growing linearly, and there is no reason to assume this would change. The actual articles do not take that much space: the XML dump of the current database is only $\approx 44GB$ [79]. On the other hand, the database with all revisions included is much bigger: [79] reports it to be several TBs after decompression (in 2010 this was $\approx 5TB$ [80]).

7.2.2 Interpretation of the statistics

There is a significant difference between the number of edits and the number of views (per month). The difference appears to be a factor > 1000 over the last 6 years. We can also see that the number of users (viewers) keeps growing steadily (linearly) once the instance gained some popularity. From the moment an instance is started up until the moment an instance gains popularity, a steep growth takes place; adding millions of users (viewers) to the system over a period of a few months/years.
The following high-availability and scalability requirements we need to meet:

- the possibility to handle a large number of simultaneous viewers: from the statistics we can expect ≈ 2500 requests per second
- the possibility to easily scale in the number of viewers
- the possibility to handle a moderate amount of editors: from the statistics we can expect ≈ 2.5 requests per second
- the content should be properly backup-ed and stored in a redundant way, since the content will grow incrementally the storage solution should allow to grow with it

7.3 An attempt to setup an architecture

Since there is clearly a difference in requirements between viewing content and editing content, I think it would make sense to separate this functionality.

7.3.1 Storing the pages

Wikimedia currently stores the Wikipedia data in databases. However, it occurred to me that the data consists out of wikitext sections (representing the articles) that have a history. So we could also store this as a set of text files kept under revision control. This would mean that we can store all articles and their history on a plain file system. Since we are interested at the history of one file at a time, using revision control such as git is overkill. A more efficient approach would be to keep a directory with the name of the page and in this directory store a set of patch files that represents the history of the page. For this to work, we would need to write a considerable amount of code, so for the sake of this demonstration I will use git. We expect our content to be >6TB (a reasonable assumption based on the fact that it was $\approx 5TB$ in 2010). Currently there is raw storage hardware available that supports up to 10TB ([81] for example), which should be sufficient for at least 5 years of scalability. Note that we could also opt for a ceph cluster subsection 5.3.3, with a practically unlimited scalability, however, for the current state of affairs this might be overkill. Several hardware solutions exist to setup this data storage in a redundant way and make sure it is properly backup-ed. However it is also necessary to make explicit incremental backups (since we're storing the different revisions of our articles, this could be done by only collecting the most recent revisions), these backups should also be stored in a datacenter at another physical location. The content storage server could than be implemented by a cluster of redundant nodes with a unified interface, so that all other servers can talk to this cluster as if it is only one machine.

7.3.2 HTTP server for editors

Since we only expect ≈ 2.5 edits per second, this could be handled by one machine that hosts a web application that allows editing pages and communicates with the content storage server [1]. If one server would no longer be sufficient to handle the load, an additional web server could be used to setup an editor cluster, behind a HAProxy proxy.

7.3.3 HTTP server for viewers

The current implementation of Wikipedia serves the content to viewers using the MediaWiki software, which is PHP software that has to render the pages on the

[1]To get revisions and submit changes to content

server side before it can serve them to the user. We no longer store the content in a a database, but as a separate file, so we might skip this rendering phase, and serve the files directly [2]. Since we expect >2500 requests per second, we still need a cluster of HTTP servers to handle this load. We can do that by using HAProxy as a front-end for our HTTP server cluster, this way, we can also easily add extra HTTP servers to accommodate an increase in requests. However, as mentioned before, our content is stored in wikitext format, we cannot serve this directly to our viewers, since their browser expects HTML. We can however pre-render the wikitext files (as soon an edit was done on one of the files) and store the results of this pre-rendering (an HTML file) directly on the HTTP servers that are about to serve the content to our viewers [3].

7.4 Testing this architecture

To test this architecture, we'll setup the following experiment:

- a content storage server
- a cluster of HTTP servers that can serve HTML files rendered from the wiktext files on the content storage server, I will use my existing HAProxy cluster for this (see section 6.4)

7.4.1 The content storage server

I created a new virtual machine named 'content' by clone my base Ubuntu virtual machine. To install git (what I will use to handle the revision control):

```
sudo apt-get install git
```

Let's git know who we are:

```
git config --global user.name "plibin"
git config --global user.email "pieter.libin@vub.ac.be"
```

[2]Seperating dynamic and static content is an optimization technique that is also advised by W. Tarreau of HAProxy [31]

[3]It is a realistic to want to keep a copy of the HTML file on each HTTP server, since we know the dataset only containing the current version of the files only takes $\approx 44GB$ (in XML format, we can assume the HTML exports of the pages will be of similar size)

Now we can setup the server git repository [82] [4]:

```
cd /var/
sudo addgroup gituser
sudo mkdir wiki-pages.git
sudo chown -R plibin:gituser wiki-pages.git
cd wiki-pages.git
git --bar init --shared=group
```

To properly initialise the server repository, we need to push something to it (still on 'content') [82]:

```
cd /home/plibin
mkdir wiki-pages
cd wiki-pages
git init
echo "Wikipedia␣experiment" > Readme
git add Readme
git commit -m 'initial commit'
git remote add origin /var/wiki-pages.git
git push origin master
```

We need to make it possible for 'content' to contact 'node1' and 'node2' whenever a new (git) push is received by 'content'. The easiest way to do this is by letting 'content' executing a remote command on the nodes via ssh.
In order to do this we need to configure the nodes so that we do not need to provide a password when logging in with ssh. We can do this by executing the following commands (on 'content') [5]:

```
ssh-keygen
ssh-copy-id node1
ssh-copy-id node2
```

Now we need to add a hook to our git repository (on 'content) that executes an update-script on our nodes ('node1' and 'node2') [83]. We need to add a

[4]Make sure to add this group and the $--shared = group$ option, if not you WILL have problems, unfortunately this is not mentioned in the the git documentation

[5]I added 'node1' and 'node2' to 'content's /etc/hosts file

'post-receive' hook, such a hook is triggered whenever the git repository receives a push. To add such a hook:

```
cd /var/wiki-pages.git
echo "ssh plibin@node1 /home/plibin/update-wiki-pages.sh"
  >> ./hooks/post-receive
echo "ssh plibin@node2 /home/plibin/update-wiki-pages.sh"
  >> ./hooks/post-receive
chmod +x ./hooks/post-receive
```

7.4.2 The HTTP nodes

Now we need to provide an update-wiki-pages.sh script on the 2 nodes, this script will:

- do a git pull to get the latest content from the storage server 'content'
- convert all files (that are in wikitext form: more specifically in the creole dialect [84]) to HTML; for this I use the txt2tags program [6]

The code for this script:

```
src/wikipedia/update-wiki-pages.sh
```

This script calls another simple script that converts a creole file to an HTML file:

```
src/wikipedia/creole2html.sh
```

We need to install txt2tags to convert the Creole files to HTML and git (this command should be executed on 'node1' and 'node2'):

```
sudo apt-get install txt2tags
sudo apt-get install git
```

To make sure that this script can do its job, we need to clone the git repository from the 'content' server on both nodes ('node1' and 'node2'):

```
cd ~
git clone content:/var/wiki-pages
```

[6]Updating all files is not efficient; in a production environment we would only update the files that actually changed

We now also need to be able to contact 'content' over ssh without using a password (these commands should be executed on 'node1' and 'node2'):

```
ssh-keygen
ssh-copy-id content
```

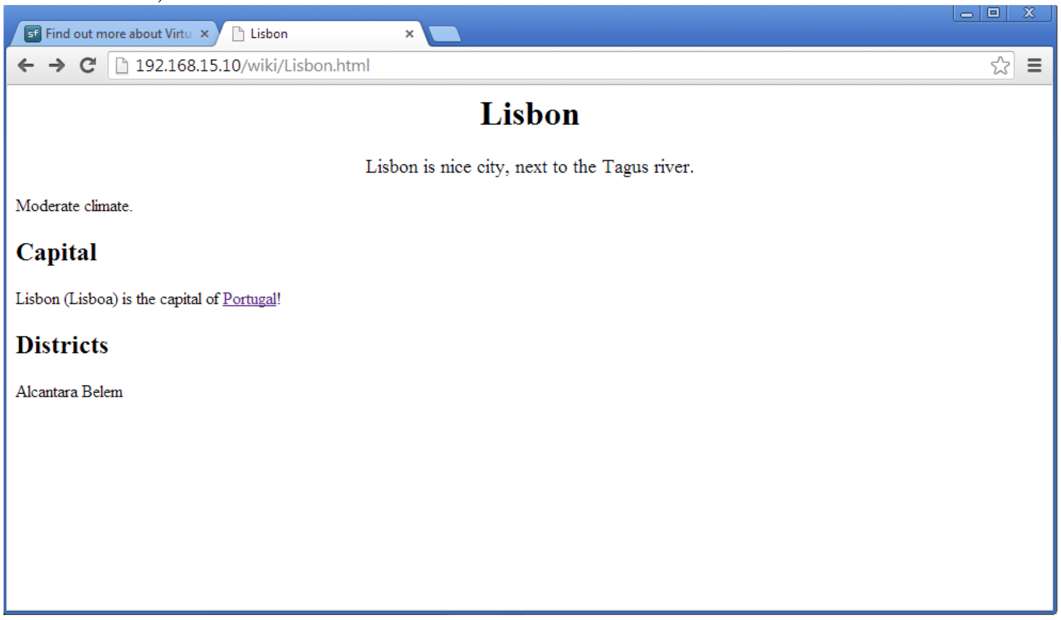
To configure the web server, we simply need to make a link to our wiki-pages git repository in /var/www [7]:

```
cd /var/www/
sudo ln -s ~/wiki-pages/ wiki
```

7.4.3 Testing the setup

After adding a "Portugal"-page and "Lisbon"-page that references the Portugal page in Creole format to the content storage server, the wiki seems to work fine (as can be seen in fig. 7.6).

Figure 7.6: Accessing our wiki via HAProxy (haproxy server has IP-address 192.168.15.10)



Lisbon

Lisbon is nice city, next to the Tagus river.

Moderate climate.

Capital

Lisbon (Lisboa) is the capital of Portugal!

Districts

Alcantara Belem

[7] Simply making a symbolic links allows the users to also see the .crl files, which is not our intention, this can be fixed by configuring Apache to only show HTML files.

I recorded a video where I add another page about "Faial", which is an island in the Portuguese Azorean archipelago:
https://www.youtube.com/watch?v=zk-Ny046kuw
I recorded another short video where I add a link to Portugal on the Faial page:
https://www.youtube.com/watch?v=T-_baLRBBfE

7.4.4 Conclusion and remarks

In my opinion, my architecture should be able to handle a website such as Wikipedia (although some additional developments would be necessary).

Performance

When looking at some benchmarks ([85]); if we expect 2500 requests per second, we would only need a couple of servers to handle this. Other web servers such as lighthttpd and nginx seem to do event better with static content.
However, to make any serious claims about the amount of servers that would be needed, we need to perform a large-scale performance test, which is out of the scope of this experiment.

Simplifications

I had to make some simplifications in my setup; the most important one being that there is no support for any dynamic content in the pages. This is however necessary for certain parts of a page (e.g.: to calculate the age of a person), however, most of these computations can also be done in JavaScript. If there would be computations that cannot be done in JavaScript, it would still be possible to let JavaScript invoke a REST service to do this computation.

Chapter 8

Conclusion

8.1 State of the art

There exist tons of open-source softwares developed for the GNU/Linux operating system that allow to build scalable and available systems. The quality of this software varies, but in general, most of this software is in good shape.
I executed several experiments, and I did not run into a lot of bugs. However, for quite some projects, the documentation was not really up to date. Because of this, I often had to lookup various documents and tutorials.

8.2 What did I learn

I really enjoyed experimenting with these different technologies. I learned a lot about how these technologies work (architecture-wise) and how they can be installed and used.
I covered several aspects of scalable and available system design and development:

- I studied existing technologies and their architecture
- I installed and configured existing technologies in a simulated environment (virtualised cluster)
- I wrote specific softwares to explore the development aspect of this domain

- I thought through an architecture and system design for a real-world scenario

Working through these different aspects thought me a lot about the domain.

8.3 What does this project contribute

First of all, I learned a huge amount by executing this project, so it was really valuable for my personal and professional development.
I think this project gives an overview of a lot of different aspects that are relevant in the domain of scalable and available clusters. The project provides a theoretical foundation, explores some of the solutions and executes real-world experiments.
I tried to thoroughly document the different steps of the experiment, to ensure the reader can reproduce all experiments [1]. When documenting these steps I often based my work on tutorials and documentation that I found online or in books, I always tried to accurately reference to these documents to avoid any plagiarism.

[1]I thought it was important to be thorough in this documentation, since very often, I had to consult many different tutorials. I hope this document can act as a starting point for executing similar experiments, alleviating the need to browse through tons of different documentation files.

Bibliography

[1] "This car runs on code."
URL: http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code.

[2] "Mars data communication."
URL: http://marsrover.nasa.gov/mission/comm_data.html.

[3] "Swift engineeringmaking cars go faster with platform hpc."
URL: http://www-01.ibm.com/software/success/cssdb.nsf/CS/GHAU-8ZU9D9
.

[4] K. Deforche, R. Camacho, Z. Grossman, T. Silander, M. A. Soares, Y. Moreau, R. W. Shafer, K. Van Laethem, A. P. Carvalho, B. Wynhoven, and Others, "Bayesian network analysis of resistance pathways against HIV-1 protease inhibitors," *Infection, Genetics and Evolution*, vol. 7, no. 3, pp. 382–390, 2007.

[5] T. Lengauer and T. Sing, "Bioinformatics-assisted anti-HIV therapy," *Nature Reviews Microbiology*, vol. 4, no. 10, pp. 790–797, 2006.

[6] "Windows nt kernel architecture."
URL: http://en.wikipedia.org/wiki/Architecture_of_Windows_NT .

[7] "Rackspace."
URL: http://www.rackspace.com .

[8] "Amazon elastic computing cloud."
URL: http://aws.amazon.com/ec2/ .

[9] "Google app engine."
URL: https://developers.google.com/appengine .

[10] "Verizon: Ip latency statistics."
URL: http://www.verizonenterprise.com/about/network/latency/ .

[11] "Stackoverflow: What is the shortest perceivable application response delay?."
URL: http://stackoverflow.com/questions/536300/what-is-the-shortest-perceivable-application
.

[12] "Data center operators flock to cold climates."
URL: http://www.networkcomputing.com/next-generation-data-center/news/servers/data-cen
.

[13] "Passive cooling in data centers."
URL: http://continuingeducation.construction.com/article.php?L=299&C=900.

[14] E. Marcus and H. Stern, *Blueprints for high availability.* Wiley. com, 2003.

[15] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, pp. 377–387, June 1970.

[16] "Wikipedia: Nosql."
URL: http://en.wikipedia.org/wiki/NoSQL .

[17] "Why nosql will not kill the rdbms."
URL: http://www.mohawksoft.org/?q=node/63 .

[18] "Wikipedia: Usage share of operating systems."
URL: http://en.wikipedia.org/wiki/Usage_share_of_operating_systems .

[19] "Intel: Hardware-assisted virtualization technology."
URL: http://www.intel.com/content/www/us/en/virtualization/virtualization-technology/har
.

[20] "Amd virtualization."
URL: http://www.amd.com/us/solutions/servers/virtualization/Pages/virtualization.aspx
.

[21] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[22] N. N. Taleb, *The Black Swan:: The Impact of the Highly Improbable Fragility.* Random House Digital, Inc., 2010.

[23] "Facebook unveils energy use, carbon emissions data."
URL: http://gigaom.com/2012/08/01/facebook-unveils-energy-use-carbon-emissions-data/
.

[24] "Facebook (php) is not very kopenhagen."
URL: http://www.webtoolkit.eu/wt/blog/2009/12/17/facebook__php__is_not_very_kopenhagen
.

[25] "Hiphop php vm."
URL: http://www.hhvm.com/blog/ .

[26] "Nagios software."
URL: http://www.nagios.org.

[27] M. Andreolini and S. Casolari, "Load prediction models in web-based systems," 2006.

[28] "Ganglia."
URL: http://ganglia.sourceforge.net.

[29] "Rddtool software."
URL: http://oss.oetiker.ch/rrdtool/.

[30] "Wikipedia: Application performance management."
URL: http://en.wikipedia.org/wiki/Application_performance_management.

[31] "Haproxy software."
URL: http://haproxy.1wt.eu.

[32] W. Tarreau, "Making applications scalable with load balancing," *Willy Tarreau's articles*, 2006.

[33] "Wikipedia: Data replication."
URL: http://en.wikipedia.org/wiki/Replication_(computing).

[34] "Postgresql db software."
URL: http://www.postgresql.org.

[35] "Mysql db software."
URL: http://www.mysql.com.

[36] "Drbd software."
URL: http://www.drbd.org.

[37] "rsync software."
URL: http://rsync.samba.org.

[38] "Linux kernel."
URL: http://kernel.org.

[39] "barman software."
URL: http://www.pgbarman.org.

[40] "Glusterfs software."
URL: http://www.gluster.org.

[41] "Ceph software."
URL: http://ceph.org.

[42] "Oracle virtualbox software."
URL: http://virtualbox.org.

[43] "Ubuntu linux 13.10 (amd64) server edition."
URL: http://www.ubuntu.com/download/server/.

[44] "Official nrpe configuration document."
URL: http://nagios.sourceforge.net/docs/nrpe/NRPE.pdf.

[45] "Life runs on code blog: Nrpe configuration post."
URL: http://blog.roozbehk.com/post/25059446631/nrpe-monitoring-linux-remote-hosts-nagios

[46] "stackoverflow.com: post on an nrpe connection problem."
URL: http://stackoverflow.com/questions/4044728/nagios-nrpe-giving-a-no-output-returned-fr

[47] "Documentation on how to simulate a linux kernel crash."
URL: http://evan.borgstrom.ca/post/33758728773/forcing-a-linux-kernel-panic.

[48] "Documentation on how to restart a linux server when a kernel panic is experienced."
URL: http://www.cyberciti.biz/tips/reboot-linux-box-after-a-kernel-panic.html.

[49] "Documentation on how to execute a bash script when a file is added on macos x.."
URL: http://j4zzcat.wordpress.com/2010/01/06/folder-actions-unix-style/.

[50] "Ganglia installation manual."
URL: http://sourceforge.net/apps/trac/ganglia/wiki/ganglia_quick_start.

[51] "Rrd 2 csv convertor."
URL: https://code.google.com/p/rrd2csv/.

[52] "R statistical software."
URL: http://www.r-project.org.

[53] "Haproxy installation tutorial."
URL: https://www.digitalocean.com/community/articles/how-to-use-haproxy-to-set-up-http-l

[54] "Mysql official replication documentation."
URL: http://dev.mysql.com/doc/refman/5.0/en/replication.html .

[55] "Howtoforge: Mysql replication documentation."
URL: http://www.howtoforge.com/mysql_database_replication .

[56] "Stackexchange: Mysql replication documentation."
URL: http://dba.stackexchange.com/questions/8680/what-is-the-best-way-to-create-mysql-ma
.

[57] "Drbd ubuntu documentation."
URL: https://help.ubuntu.com/10.04/serverguide/drbd.html.

[58] "Drbd official documentation."
URL: http://www.drbd.org/users-guide/re-drbdconf.html.

[59] "Howtoforge: Setting up raid-1 with drbd.."
URL: http://www.howtoforge.com/setting-up-network-raid1-with-drbd-on-ubuntu-11.10-p2
.

[60] “Glusterfs vs. ceph.”
URL: http://hekafs.org/index.php/2013/01/ceph-notes/ .

[61] “Glusterfs vs. ceph: debate.”
URL: http://www.youtube.com/watch?v=JfRqpdgoiRQ .

[62] “Ceph archtecture.”
URL: http://www.anchor.com.au/wp-content/uploads/2012/09/ceph_stack.png
.

[63] “Ceph: A linux petabyte-scale distributed file system.”
URL: http://www.ibm.com/developerworks/library/l-ceph/ .

[64] “Ceph official documentation.”
URL: http://ceph.com/docs/master/ .

[65] “Ceph bug report: ceph-deploy install does not create ceph.conf in /etc/.”
URL: http://tracker.ceph.com/issues/5849 .

[66] “Chef: It automation.”
URL: http://www.getchef.com/chef/ .

[67] “Tumblr on database sharding.”
URL: http://velocityconf.com/velocityeu/public/schedule/detail/21678 .

[68] “Java software.”
URL: http://java.com.

[69] “Ant build software.”
URL: http://ant.apache.org.

[70] “Hivedb sharding library.”
URL: http://www.hivedb.org .

[71] “Multi-tenancy in hibernate.”
URL: http://in.relation.to/Bloggers/MultitenancyInHibernate .

[72] “Ceph: more than an object store.”
URL: http://ceph.com/community/more-than-an-object-store/ .

[73] "Gnu free documentation license."
URL: http://www.gnu.org/copyleft/fdl.html .

[74] "Creative commons attribution-sharealike license."
URL: http://creativecommons.org/licenses/by-sa/2.5/ .

[75] "Wikitext markup language."
URL: http://en.wikipedia.org/wiki/Wiki_markup .

[76] "Wikimedia statistics."
URL: http://stats.wikimedia.org .

[77] "Wikipedia statistics."
URL: http://stats.wikimedia.org/EN/Sitemap.htm .

[78] "Wikipedia database growth statistics."
URL: http://stats.wikimedia.org/EN/ChartsWikipediaSIMPLE.htm .

[79] "Wikipedia's size."
URL: http://en.wikipedia.org/wiki/Wikipedia:Database_download .

[80] "Wikipedia's full database size."
URL: http://www.quora.com/Wikipedia/What-is-the-data-size-of-Wikipedia-in-Jan-2011
.

[81] "Hp 10tb raw storage."
URL: http://www8.hp.com/us/en/products/file-object-storage/product-detail.html?oid=5335
.

[82] "Setup a git server."
URL: http://git-scm.com/book/en/Git-on-the-Server-Setting-Up-the-Server
.

[83] "Git hooks."
URL: http://git-scm.com/book/en/Customizing-Git-Git-Hooks .

[84] "Creole wikitext dialect."
URL: http://www.wikicreole.org .

[85] “Web server performance comparison.”
URL: http://wiki.dreamhost.com/Web_Server_Performance_Comparison .