Effects of the EPNs and FLPs on the Information Node for ALICE Cern

M.Q Puls

2018

Title	Effects of the EPNs and FLPs on the Information Node for ALICE
Name	Mitchell Quinn Puls
Student Number	500659986
Phone	+31615050310
Email	mitchpuls@upcmail.nl mitch.puls@hva.nl
Place	Amsterdam
Date	June 27 2018
University	University of Applied Sciences Amsterdam
Department	TI
Mentor	C. J. Rijsenbrij
Company	University of Amsterdam Software for Science Wibautstraat 2-4 Amsterdam 0205995555
Company Supervisor	Dr. Marten Teitsma
Period	Feb. 2018 - Jul 2018

Preface

This report is a representation of the hard times through my study. It is the product of my ups and downs. Without the support of my teachers at the Amsterdam University of Applied Science, I would not be here where I am now. I would like to thank Dr. Marten Teitsma for giving me the opportunity to perform this research, Heiko van der Heijden for the support during the project, and my friends and family for dealing with me during this semester.

Mitchell Quinn Puls, 27 June 2018

Contents

Chapter 1

Summary

One of CERNs detectors on the Large Hadron Collider is called ALICE, which stands for A Large Ion Collider Experiment. ALICE is a 10.000 tonne detector, which main purpose is detecting collisions, in particular lead particle collisions. Once these particles collide they become extremely hot and breakdown to a matter called quark-gluon plasma. ALICE detects this collision and gathers the data from it.

Starting in 2018, ALICE will go offline for a two year break to upgrade it's software. A piece of this software is a load balancing algorithm. When ALICE is running, it will pump out 1.1TB/S of data. This algorithm is supposed to be able to efficiently distribute this data over a total of 1750 computers. 250 of these computers are whats called First Level Processors, and the other 1500 are called Event Processing Nodes. All of these computers are meant to process, compress and store the data so that it can be recreated later on.

This data is distributed with heartbeats. These heartbeats occur every 20ms. FLPs will gather the data between two heartbeats, and compresses them into a Sub Time Frame. These STFs are sent to the EPNs. Here they will be compressed further down into a Time Frame. When an EPN is disabled, it should not receive any data anymore, at which it should be removed from the data line. When this event happens it is reffered to as a fail-over. This whole process of distributing data both efficiently and correctly is called Load Balancing.

A previous experiment done by Heiko van der Heijden showed that a Blacklist algorithm is a more efficient algorithm for this load balancing application, compared to a Re-initialization algorithm. The experiment was conducted using a cluster available on Nikhef. The amount of units used during the experiment was 2 FLPs and 12 EPNs, which is equivalent to the same ratio that is used at CERN. The availability of this cluster was uncertain, and hence a cluster was needed that was more available, and also expandable. This resulted in a cluster of Raspberry Pi's.

In order to validate this cluster compared to the one on Nikhef, a correlation needed to be found between the results from the cluster and the new one. Afterwards these experiments can be executed with an unit count of 3 FLPs and

18 EPNs, and 4 FLPs and 24 EPNs.

The software used consists of two main frameworks, FairMQ and Zookeeper. FairMQ is a data transport layer of the larger framework FairRoot. For this experiment a specially made trimmed down version was used in order to accommodate with the Raspberry Pi. Zookeeper regulates the Blacklist. Here the registration of online and offline EPNs are done so that the FLPs know what to send where. Zookeeper runs on a special computer called the Information Node.

At first this new prototype using Raspberry Pi's had to be developed and built. The complications lay in the Raspberry Pi having only one Ethernet connection natively, and the software needing changes to work with an ARM architecture. After that, three experiments were conducted that were the same as the previous experiment. The first one disables one EPN during the run, the second disables every EPN apart from one during a run, and the last one disables every EPN during a run, but also uses a random sample size. These three experiments were then analyzed with the previous experiment to calculate a correlation between them. After that the same three experiments are conducted again with a higher amount of FLPs and EPNs. Finally two new experiments are run, where multiple EPNs will disable at once. During the first one these EPNs will all be after one another in the list of available EPNs. During the second one these EPNs will be scattered in the list of available EPNs. This to check whether the Blacklist algorithm has a different results depending on the layout of the system architecture.

In conclusion the validity of the new prototype is proven by having a Pearson correlation value of 0.95, 0.99 and 0.99. With an error rate of 0.1%, these values all strongly deny the null hypothesis of there being no correlation between the experiments. Furthermore the expansion of the experiment shows that there is no negative effect on the Information Node. TF loss goes closer to 1, which is the minimal TF loss in this current setup. Lastly the layout of the system architecture is important for individual clusters that have a fail-over, but the total TF loss when every cluster has had a fail-over is still the same.

It is recommended to further increase the amount of units to check wether or not the TF loss would eventually get to 1, if it would reach an asymptote at some point, or if it turns around at some point and becomes a sine wave possibly. It is also recommended to continue doing these experiments using Ansible induced fail-overs, as opposed to TF induced fail-overs. This to eliminate the predictability of the fail-overs, and to eliminate the minimal 1 TF loss. Lastly it is recommended to use a lowest latency approach, as opposed to a round robin approach, for the data distribution. Using this in conjunction with the Blacklist might obtain a TF loss of 0, because of the EPN that has a fail-over would not be targetted.

Chapter 2

Introduction

This research is conducted as part of the final thesis of Mitchell Quinn Puls, Technical Computing student at the Amsterdam University of Applied Sciences. This research is about the effects of a high amount of computers, processing a vast amount of data. This research is, in assignment from CERN in Switzerland.

2.1 CERN

CERN is a European organization for nuclear research, situated in Geneve Switzerland. CERN was founded in 1954 and is one of Europe's first joint ventures. The main goal is to study the fundamental structure of the universe, by researching matter and particles using purpose built particle accelerators and detectors. Particle accelerators beam particles to high energies, before colliding them against each other against or a stationary object. Detectors record and observe this collision (About Cern, 2012). One of these detectors is ALICE

2.2 ALICE

ALICE stands for A Large Ion Collider Experiment, and is a detector mounted on the Large Hadron Collider at CERN. ALICE's main function is to study matter at extreme energy densities, where matter turns into a form called quark-gluon plasma. Everything in the universe is made from protons, neutrons (except hydrogen which does not have any neutrons) and electrons. Protons and neutrons are then build up with quarks and bound together with something called a gluon. Quark-gluon plasma is matter that appeared at the very start of the big bang, and is the matter that appears when the quarks and gluons are seperated from each other. CERN wants to observe this matter. The way they achieve this, is by shooting two lead ions against each other. This produces heat that is over 100,000 times hotter than the center of the Sun. This breaks the bounds between the quarks and the gluons and makes the quark-gluon plasma visible. (ALICE, 2012)

2.2.1 Upgrade

In July 2018 the accelerator will be stopped for around 18 months for a planned upgrade of the ALICE detector. (van der Lee, 2017, p. 1) During this period, CERN is upgrading it's hardware and software. This upgrade is in collaboration with various schools and universities throughout Europe, including the Amsterdam University of Applied Science. One of these upgrades is an algorithm for Load Balancing. In 2020, ALICE will restart with it's new upgraded detector. ("Technical Design Report for the Upgrade of the Online Offline Computing System", 2015, p. i)

2.3 Load Balancing

The data stream that comes from ALICE is equal to about 1.1 Terabyte per second. All of this data comes in what is known as a heartbeat. This heartbeat gets distributed over 250 First Level Processors and funneled through 1500 Event Processing Nodes. The efficient distribution of this process, and also the handling of data in case of a failure in the system, is what is known as Load Balancing. All of these computers are monitored by an Information Node.

Chapter 3

Research

3.1 Research

This research project is a continuation of a previous research projects done by Heiko van der Heijden and Tom van der Lee. Heiko's results show that of the two algorithms tested, Re-initialization and Blacklist, that the Blacklist algorithm has fewer Time Frames lost. (van der Heijden, 2018, p. 43)
Even though the same ratio of FLPs to EPNs that is situated at CERN was used (1/6), there were fewer computers used than at CERN. Because of this it is not sure whether or not the Information Node is able to handle 1700+ computers as compared to the 15 computers used in the experiment. This research is focused on the capability of the Information Node to monitor a higher number of FLPs and EPNs and what the effects are on the results compared to the previous experiment.

3.2 Research Model

Contrary to the previous research, which used a cluster of computers situated at Nikhef Amsterdam, this research will be conducted using a cluster of Raspberry Pi's. The first step is to recreate the previous experiment which was focused around the various ticktimes of Zookeeper.

The main purpose of this research is to both validate the results from the previous experiment, but also to validate the new Raspberry Pi cluster to confirm that this is a valid way to conduct this experiment. After recreating the first experiment, the same experiment will be conducted with a higher numbers of computers to see if this has an effect on the Information Node. Finally an experiment will be conducted to see whether the layout of the system architecture is of influence for the TF loss.

An overview can be seen in figure 3.1

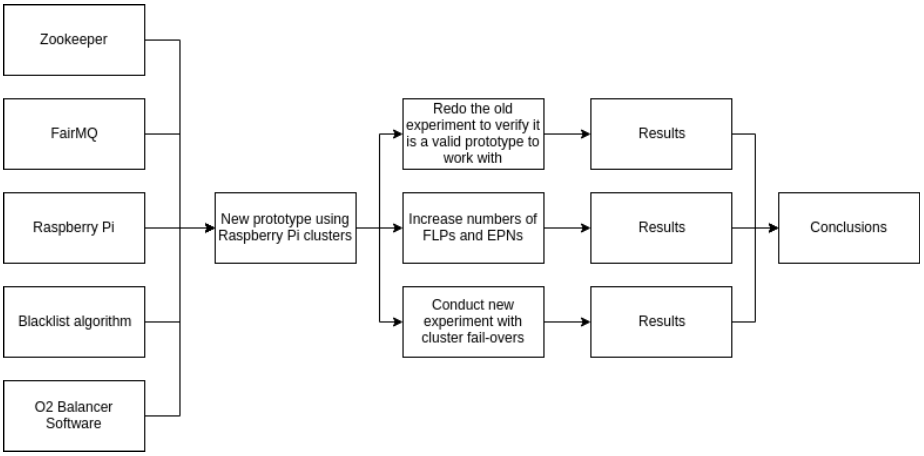

Zookeeper
FairMQ
Raspberry Pi
Blacklist algorithm
O2 Balancer Software
New prototype using Raspberry Pi clusters
Redo the old experiment to verify it is a valid prototype to work with
Increase numbers of FLPs and EPNs
Conduct new experiment with cluster fail-overs
Results
Results
Results
Conclusions

Figure 3.1: Research model

All technical documentation of what everything is will be explained in the next section including the definition of the experiments. The next chapter will explore more in depth of the prototype made for the experiment and it's difficulties that came with it. The following chapter looks at the results from the executed experiments. After this follows an analysis of the results of the experiment. Finally a conclusion and recommendations.

Chapter 4

Framework

This chapter will go more in depth about the several different kinds of hardware and software used. It will also elaborate on the different terms and names used for aspects of the research.

4.1 O^2 Balancer

O^2 Balancer is a framework used by CERN for simulation experiments for ALICE. The code is open source and licensed under the GNU General Public License V3.0. (GNU General Public License v3.0, 2007)

4.1.1 Devices

The O^2 Balancer consist of a cluster of 1750 computers, divided in 250 First Level Processors (FLPs) and 1500 Event Nodes (EPNs) These computers are meant to process the data stream coming from ALICE. All of these computers are monitored using an Information Node. (”Technical Design Report for the Upgrade of the Online Offline Computing System”, 2015, p 33-35)

First Level Processors

The FLPs are the first computers in the line. They receive the data stream (approximately 1.1TB/s) from ALICE and need to distribute that to the next line of computer. In order to do that it takes the data received between two heartbeats, and compresses that into something that's called a Sub Timeframe (STF). A heartbeat lasts for about 20ms. It will then send this STF to the next line of computers which are the EPNs. Every EPN needs to get the same amount of STFs at the same time for recreation purposes. These STFs can then be further examined from there.(”Technical Design Report for the Upgrade of the Online Offline Computing System”, 2015, p. 33)

Event Processing Nodes

The next line of computers are the EPNs. These receive the STFs from the FLPs and then compress them back into a time frame (TF). This compression reduces it's size by a factor of eight. These TFs are then stored for further use and examination.(”Technical Design Report for the Upgrade of the Online Offline Computing System”, 2015, p. 33)

Information Node

There is one final computer which is the Information Node (IN). This computer keeps track of all the FLPs and EPNs that are online and makes sure that FLPs don't send data to offline EPNs. (”Technical Design Report for the Upgrade of the Online Offline Computing System”, 2015, p. 34)

4.2 FairMQ

The transport layer used for the O^2 Balancer is FairMQ. This is a transport layer from the larger framework FairRoot (FairRootGroup/FairRoot, 2018) created by GSI Darmstadt. In order to accommodate the smaller processing size of the Raspberry Pi, a trimmed down version of FairRoot is used which is just FairMQ. This is a data transport layer used to send data in between the IN, FLPs and EPNs.

4.2.1 Splitting off FairMQ

During this report, the FairMQ repository was split off from FairRoot. In the first stages of the prototype an emergency version was used made by Heiko van der Heijden. Meanwhile a pull request (Make a special repository for FairMQ, 2018) was done on the FairRoot github which resulted in the official FairMQ repository.

4.3 Zookeeper

Zookeeper is a program made by Apache (Apache Zookeeper, 2017) to regulate the whole load balancing process. It is run on the Information Node and from there pings to all EPNs to check whether they are online or not. It then creates a list of online EPNs which it gives to the FLPs so that they know to what EPN to send data to. The frequency of these pings are called the Ticktime.

4.4 Fail-over

When an EPN goes offline it is called a fail-over. When this happens, Zookeeper will know that it is offline and will notify the FLPs to not send any data to this specific EPN anymore.

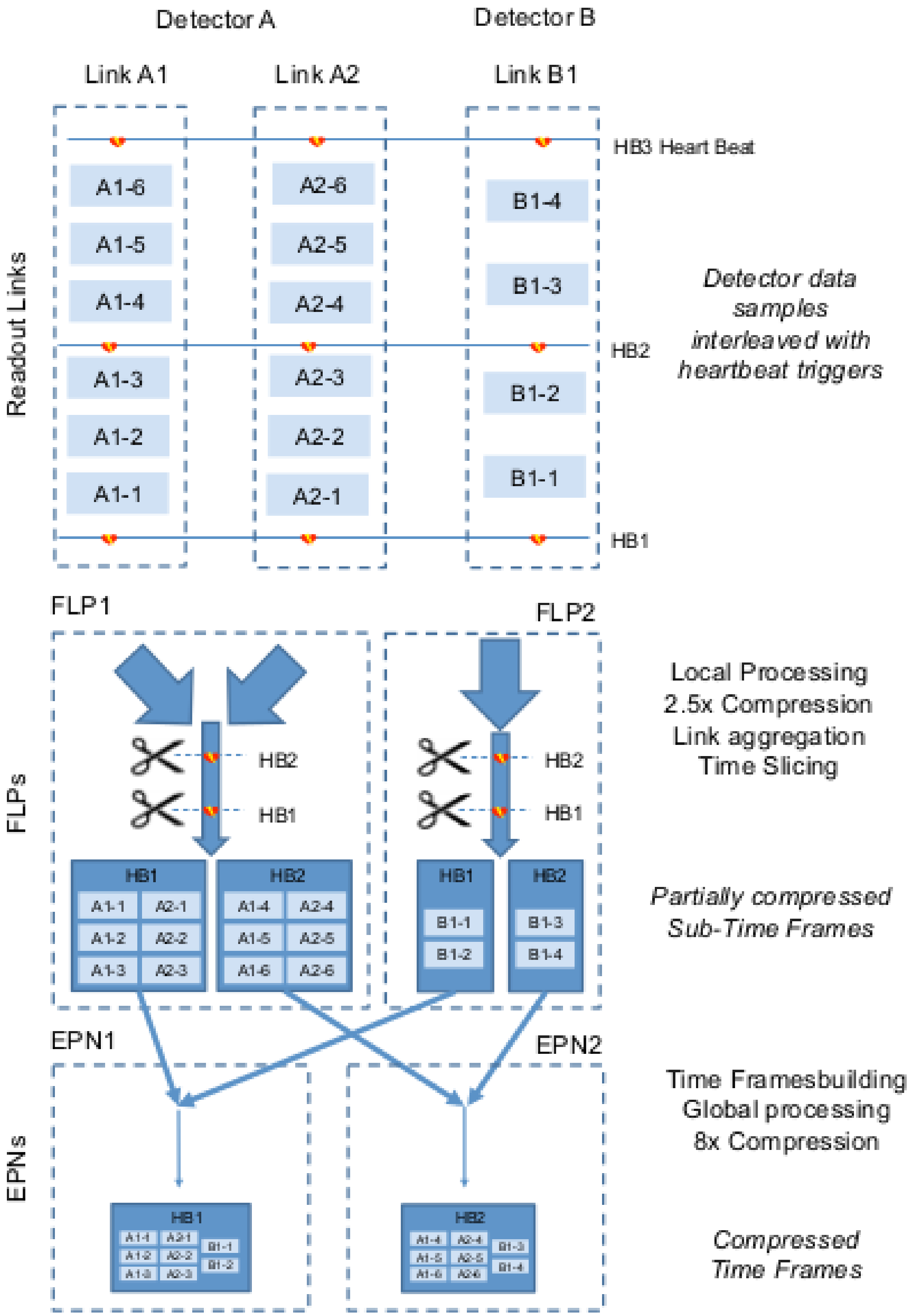


Figure 4.1: Aggregation of data ("Technical Design Report for the Upgrade of the Online Offline Computing System", 2015, p. 34)

4.5 Ansible

Ansible is a deployment software used to create simple automation for large infrastructures. This is used to automate repetitive task for the experiment, and for deploying software stacks to every unit.

4.6 Blacklist Algorithm

The algorithm used in the previous experiment is called a Blacklist Algorithm. This algorithm constantly keeps a list of ofline channels which is updated by the Information Node using Zookeeper. Once Zookeeper realizes that an EPN is offline, it will update the list so that the algorithm will skip that offline EPN. With this list of online EPNs, the algorithm uses a Round Robin approach to distribute the STF over the EPNs. A way to implement the Blacklist algorithm is shown in figure 4.2

```
class FLPDevice(FairMQDevice):
  def __init__(self):
        self.__zookeeper = Zookeeper()
  def ConditionalRun(self):
        # Create some parts to send
        parts = FairMQParts()
        parts.AddPart(NewMessage())
        parts.AddPart(NewMessage())
        fChannels.at(”stf1”).Receive(parts.at(0))
        heartbeatID = int(parts.at(0).GetData())

        # Get channels
        channels = fChannels.at(”stf2”)
        # Filter the channels
        filteredList = [
                online.GetAddress() for
                online in channels if
                online.GetAddress() not in
                self.__zookeeper.getBlacklist()]
        # Execute Round Robin
        nr = heartbeatID % len(filteredList)

        # Determine the correct index it was in
        for i in range(len(channels)):
          if channels[i].GetAddress() == filteredList[nr]:
                send(parts, ”stf2”, i)
                return True
        return False
```

Figure 4.2: Blacklist algorithm as it could be implemented in Python (van der Heijden, 2018, p. 20)

4.7 Raspberry Pi

Raspberry Pi is a low cost small computer used for prototyping projects (Raspberry Pi About Us, n.d.). These projects go from small sensor applications, to bigger host-server applications. The Raspberry Pi used for this research is the model 3 B+ variant. As of the time this report is written this is the latest version released. This model is used because of the high Ethernet speeds on the board itself. The reason a Raspberry Pi was used for this experiment is because it's a smaller and cheaper alternative compared to what the cluster on Nikhef used. (van der Heijden, 2018, p. 21) and because of this, it is more easily expanded.

Chapter 5

Experiments

This chapter goes more in depth of the specifics on the experiments conducted. It describes the exact hardware specifications and software libraries used. It also defines all the different experiments and includes several figures for added information. At first the hardware used will be discussed. After that the software and it's specifications. Finally every experiment that is conducted and their definitions.

5.1 Hardware

The hardware used for this experiment are specially designed clusters made with Raspberry Pi's 3 model B+. These Pi's are assigned using the same ratio of FLPs and EPNs at CERN (1:6) and 1 Information Node. The Pi's are modified with an extra Ethernet port. The exact specifications are in table 5.1.

Processor	Cortex-A53 1.4GHZ
RAM	1GB LPDDR2 SDRAM
Network	1 300MbE 1 10/100MbE
Operating System	Raspbian Stretch (Debian 9)

Table 5.1: Specifications modified Raspberry Pi 3 model B+

Everything is setup and built from a single server unit. The IP addresses are also managed from this unit. Specifications are in table 5.2.

Processor	Cortex-A53 1.4GHZ
RAM	1GB LPDDR2 SDRAM
Network	300MbE
Operating System	Raspbian Stretch (Debian 9)

Table 5.2: Specifications management server

The network configuration consists of a bidirectional connection. This is used to reduce the bottleneck over one interface. Load Balancing is done over the first interface, and monitoring is done over the second interface. Figure 5.1 shows a network overview:

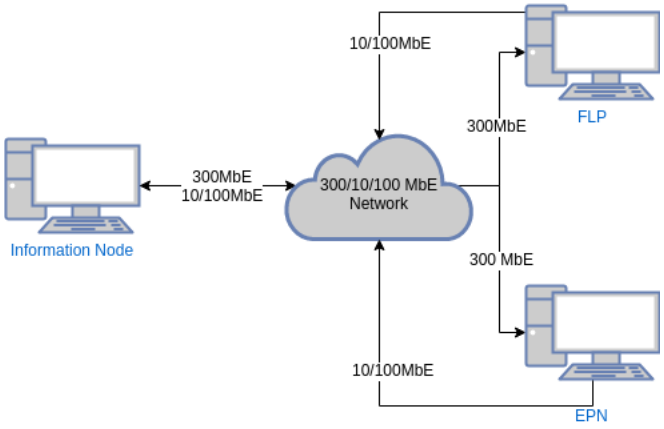


Figure 5.1: Network setup

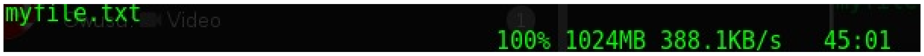
The clusters of Pi's are build using sets of four Raspberry Pi 3 Model B+ with extra ethernet interfaces. These are connected to each other using two tp-link TL-SF100D switches. From these switches the sets of Pi's are connected through one Juniper EX3400 Ethernet switch. Specifications are in table 5.3. This setup is used because of the high amount of ethernet ports required, and a switch that could accomidate all of it (+/- 60) was not available.

Switch	Speed
tp-link TL-SF100D	10/100Mbps
Juniper EX3400	1/10GbE

Table 5.3: Ethernet switch speeds

5.1.1 Complications

The second interface is obtained using a LogiLink UA0025C USB2.0 to Ethernet adapter. This adapter has a speed of 10MbE/100MbE and is primarily used by Zookeeper to register which Pi is still running. During the build of the prototype there was also an option of using an ENC28J60 Ethernet board instead. This would be plugged onto the GPIO pins on the Raspberry Pi. This board is a lot slower than the LogiLink though, clocking in at around 380 KB/s.



```
myfile.txt
                                   100% 1024MB 388.1KB/s   45:01
```

Figure 5.2: ENC28J60 Ethernet board connection speed

At first the experiments were done using one set of four Pi's using the ENC28J60 boards, and three sets of four Pi's using the LogiLink adapter. During these experiments it was found out that the ENC28J60 boards weren't that reliable. The FLPs and Information Node were using these interfaces, but during tests they would completely disable and ruin the tests results. After that an experiment was done by swapping the FLPs and Information Node with EPNs. This result showed that the specific EPNs would disable midway through as well. Because of this the one set of Pi's was also converted to use the LogiLink instead.

5.2 Software

The software used is found at https://github.com/SoftwareForScience/O2-Balancer. It consists of 3 executable programs to represent a FLP, EPN and Information Node. These programs are sent to the devices to represent their respective units. Apart from that it uses a stripped down version of FairRoot (FairRootGroup/-FairRoot, 2018). Since only the FairMQ (FairRootGroup/FairMQ, 2018) part is needed for the programs to run it has been split off from this code.
The Infomation Node serves two purposes. At first it generates the TFs that are send to the FLPs using FairMQ. It also receives notifications from the EPNs when they have received the full TF from the FLPs again. The configuration is done using YAML files. A diagram of the connections and dependencies are shown in figure 5.3 and table 5.4.

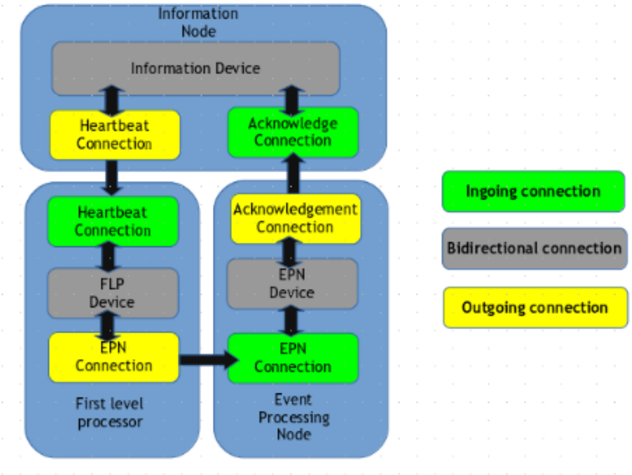


Figure 5.3: Block diagram of the cluster connections (van der Heijden, 2018, p. 22)

Library/Tool	Version
FairMQ	1.1.5
ZeroMQ	4.2.1
Zookeeper	3.4.9
Cmake	3.11.0
Boost	1.66.0
Yaml-cpp	0.5.2
Compiler	gcc 6.3.0 20170516 (Raspbian 6.3.0-18+rpi1+deb9u1)

Table 5.4: Dependencies software

5.3 Data analysis tools

In order to stay consistent with the previous experiment, the same tools will be used to create and visualize the data. This can be found at https://github.com/valvy/BalancerScripts. These scripts are written in Python and ROOT to generate graphs and histograms from the log files received from the software. The dependencies are specified in table 5.5.

Tool	Version
Operating system	Centos 7
ROOT	6.13.02
Python	2.7.13
Ansible	2.2.1
Scipy	1.1.0

Table 5.5: Dependencies for the analysis scripts

5.4 Experiments

In order to check whether or not the Information Node has any issues with increased numbers of FLPs and EPNs, the same experiments need to be done as described in the previous report (van der Heijden, 2018, p23-p27) but will be briefly summarized again. The experiments are done in two steps. At first the experiments need to be run using the same amount of FLPs and EPNs to verify whether or not the new setup of Raspberry Pi clusters are able to give the same result. After that the experiments need to be run again using more FLPs and EPNs to check if it indeed does have an effect on the Information Node. Lastly two additional experiments will be conducted to check whether or not the layout of the system architecture is of influence to the TF loss.

5.4.1 Experiment one

Ticktime influence on the Blacklist algorithm with one fail-over.

The first experiment uses a fixed sample size to be sent from the Information Node, and will have 1 fail-over during its run. This sample size is set to 100 kilobyte and the heartbeat rate is set to twenty milliseconds. This heartbeat is set at the same rate used at CERN. This sample size is set to accommodate the slower Ethernet and processing speed of Raspberry Pi as compared to units used in the previous experiment (van der Heijden, 2018, p.23).
This experiment will disable one EPN when it receives the first STF from hearbeat 3.000. Then special scripts will parse the logfiles between heartbeat 2.000 and 10.000 to have enough of a buffer to read from. A flowchart is shown in figure 5.4.

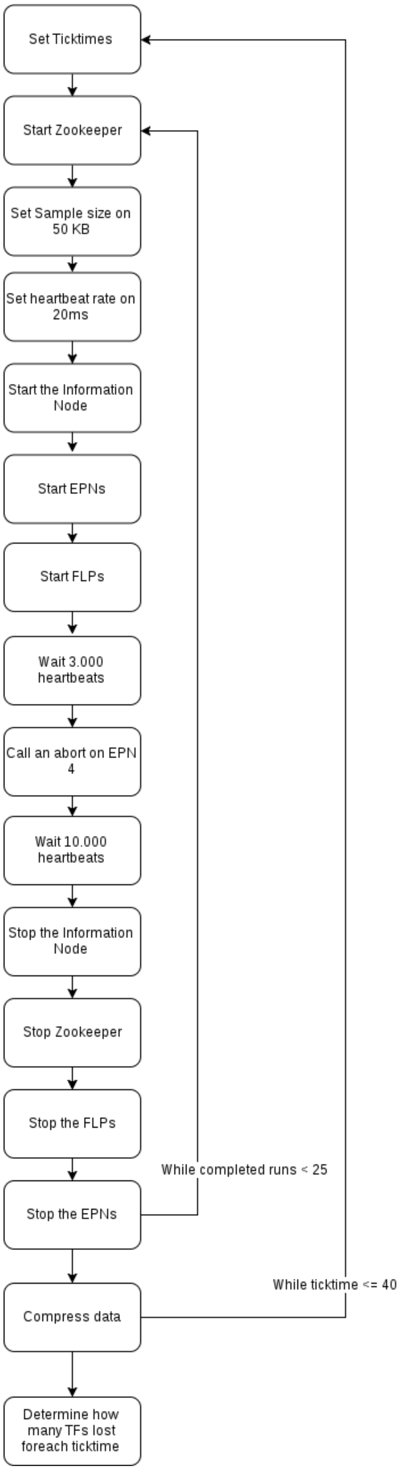


Figure 5.4: Flow chart experiment one

5.4.2 Experiment two

Ticktime influence on the Blacklist algorithm with all but one fail-over.

The same heartbeat rate and sample size are used from experiment one. For this experiment the first EPN will be disabled at heartbeat 2.000. After that every 1.000 heartbeats an additional EPN will be disabled until there is only 1 left. Scripts will parse all the logs to check how many TFs were lost during this progress. A flowchart can be found in figure 5.5.

5.4.3 Experiment three

Ticktime influence on the Blacklist algorithm with all but one fail-over with random sample size.

The same heartbeat rate is used from experiment one. A random sample size will be generated from the FLPs with an average size of 100 kilobytes. Apart from that the same configuration will be used as in experiment two. The first EPN will be disabled at heartbeat 2.000 and after that every 1.000 heartbeats an additional EPN will be disabled. A flowchart can be found in figure 5.6.

5.4.4 Experiment four

Ticktime influence on the Blacklist algorithm with a fixed cluster fail-over pattern using a random sample size

The same heartbeat rate is used from experiment one. The same random sample size from experiment three is used. For this experiment the first cluster of four raspberry pi's will be disabled at heartbeat 3.000. After that every 2.000 heartbeats an additional cluster will be disabled until there is only one cluster left. A flowchart can be found in figure 5.7.

5.4.5 Experiment five

Ticktime influence on the Blacklist algorithm with a random cluster fail-over pattern using a random sample size

The same heartbeat rate is used from experiment one. The same random sample size from experiment three will be used. For this experiment a random set of four raspberry pi's will be disabled at heartbeat 3.000. After that, every 2.000 heartbeats an additional set of four pi's are disabled until there is only one set left. A flowchart can be found in figure 5.8.

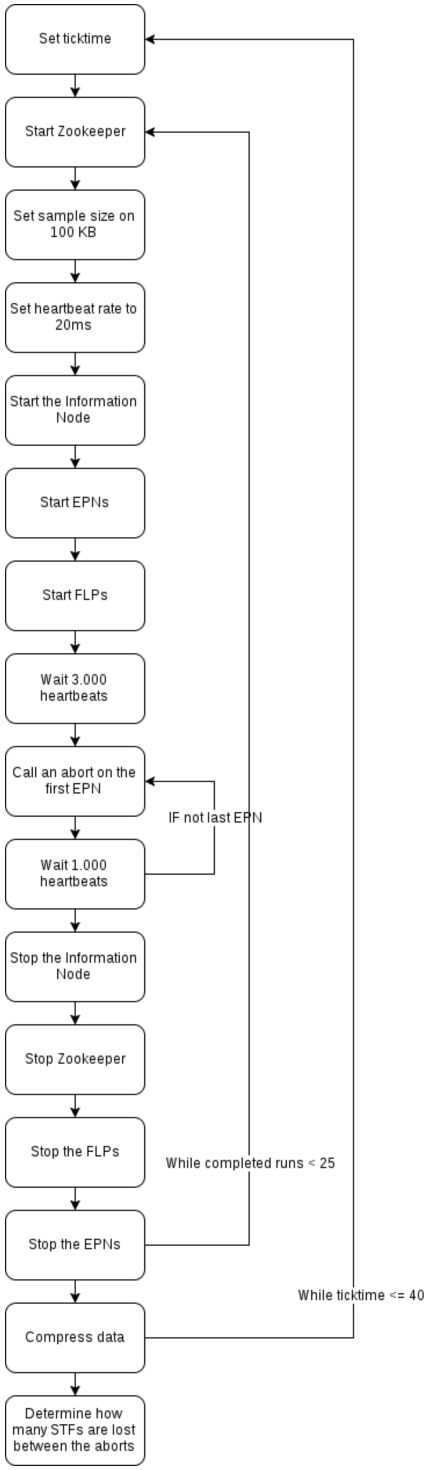

Set ticktime
Start Zookeeper
Set sample size on 100 KB
Set heartbeat rate to 20ms
Start the Information Node
Start EPNs
Start FLPs
Wait 3.000 heartbeats
Call an abort on the first EPN
IF not last EPN
Wait 1.000 heartbeats
Stop the Information Node
Stop Zookeeper
Stop the FLPs
While completed runs < 25
Stop the EPNs
While ticktime <= 40
Compress data
Determine how many STFs are lost between the aborts

Figure 5.5: Flow chart experiment two

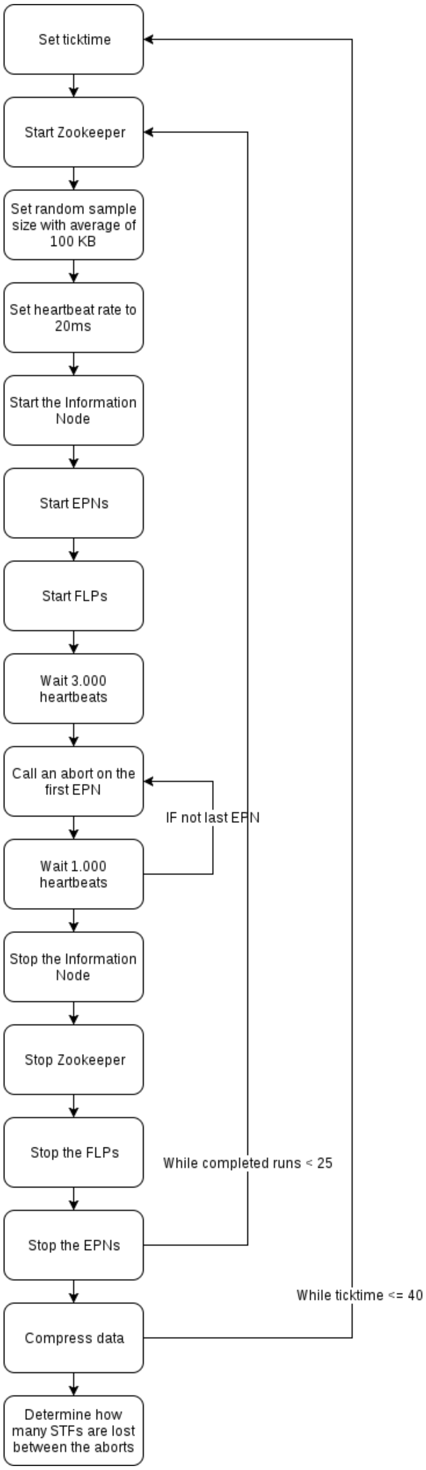

Set ticktime
Start Zookeeper
Set random sample size with average of 100 KB
Set heartbeat rate to 20ms
Start the Information Node
Start EPNs
Start FLPs
Wait 3.000 heartbeats
Call an abort on the first EPN
IF not last EPN
Wait 1.000 heartbeats
Stop the Information Node
Stop Zookeeper
Stop the FLPs
While completed runs < 25
Stop the EPNs
While ticktime <= 40
Compress data
Determine how many STFs are lost between the aborts

Figure 5.6: Flow chart experiment three

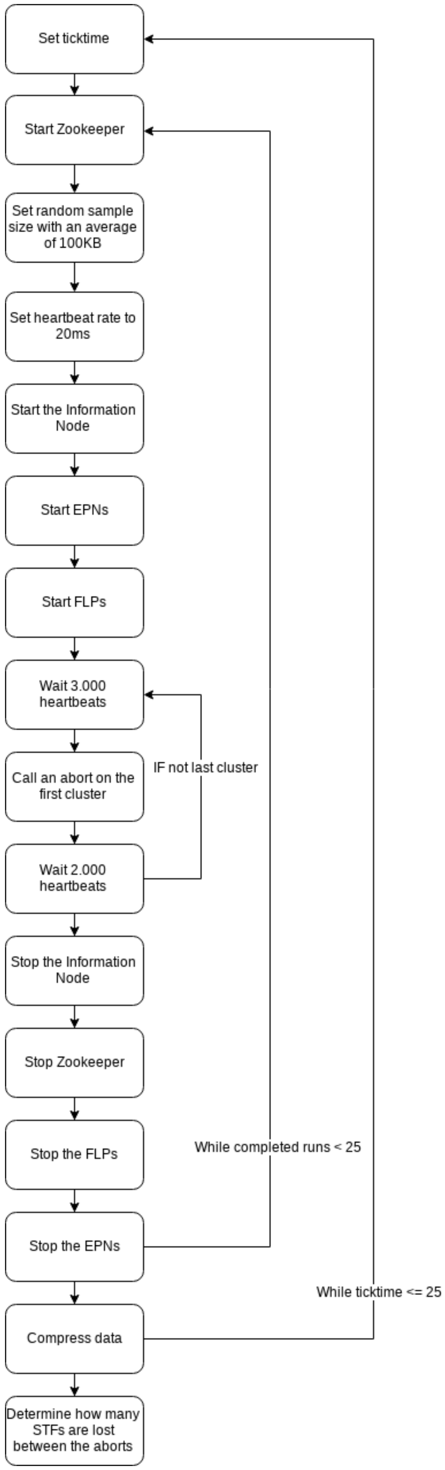

Set ticktime
Start Zookeeper
Set random sample size with an average of 100KB
Set heartbeat rate to 20ms
Start the Information Node
Start EPNs
Start FLPs
Wait 3.000 heartbeats
Call an abort on the first cluster
IF not last cluster
Wait 2.000 heartbeats
Stop the Information Node
Stop Zookeeper
Stop the FLPs
While completed runs < 25
Stop the EPNs
While ticktime <= 25
Compress data
Determine how many STFs are lost between the aborts

Figure 5.7: Flow chart experiment four

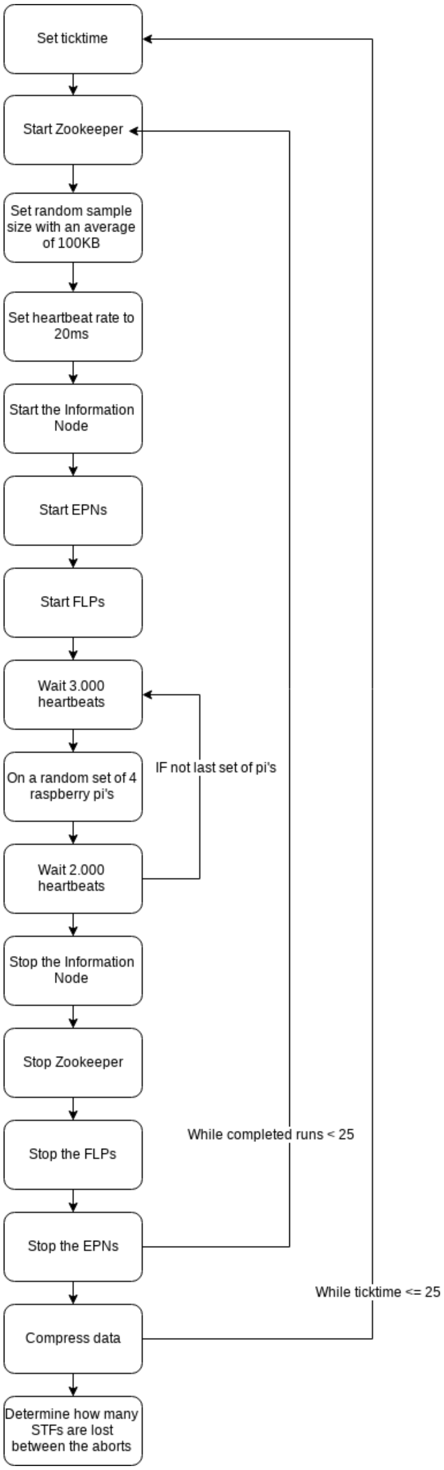

Set ticktime
Start Zookeeper
Set random sample size with an average of 100KB
Set heartbeat rate to 20ms
Start the Information Node
Start EPNs
Start FLPs
Wait 3.000 heartbeats
On a random set of 4 raspberry pi's
IF not last set of pi's
Wait 2.000 heartbeats
Stop the Information Node
Stop Zookeeper
Stop the FLPs
While completed runs < 25
Stop the EPNs
While ticktime <= 25
Compress data
Determine how many STFs are lost between the aborts

Figure 5.8: Flow chart experiment five

Chapter 6

Results

6.1 Results

Ex.1 2 FLPs 12 EPNs

Ticktime influence on the Blacklist algorithm with one fail-over.

The results of the experiment are shown in table 6.1. It shows that there is a linear line going up with the amount of TFs lost compared to ticktime. It also shows that the mean and standard deviation at ticktime 20 are round. This would be because of the ticktime being in sync with the heartbeat rate.

Experiment one (2/12) using a cluster of Raspberry Pi's

Ticktime	Mean TF loss	Standard Deviation
5	2.16	0.49
10	2.13	0.47
15	3.08	0.2
20	3.0	0
25	3.52	0.51
30	4.32	0.57
35	4.13	0.79
40	4.56	0.16

Table 6.1: Results of the TFs lost with 1 fail over using a cluster of Raspberry Pi's

If we compare this to the previous experiment shown in table 6.2, we can see that there is a small differences in the mean of the TFs lost (<1), and some small variations in the standard deviation.

Experiment one (2/12) using a cluster on Nikhef

Ticktime	Mean TF loss	Standard Deviation
5	1.24	0.4271
10	1.84	0.731
15	2.16	0.3666
20	2.12	0.325
25	2.708	0.4545
30	3	0
35	3.52	0.4996
40	3.76	0.4271

Table 6.2: Results of the TFs lost with 1 fail over using a cluster of Intel Xeons (van der Heijden, 2018, p. 36)

Ex.2 2 FLPs 12 EPNs

Ticktime influence on the Blacklist algorithm with all but one failover.

Table 6.3 shows the lost TFs per ticktime/EPN ratio. For every extra EPN that is lost, the lost TFs increase in a linear motion. This is the same linear motion visible in the previous experiment. These results are shown in table 6.4. A histogram of ticktime 5 shown at 6.1 shows that the standard deviation is 0.9568.

Experiment two (2/12) using a cluster of Raspberry Pi's

Lost EPNs	1 EPN	2 EPNs	3 EPNs	4 EPNs	5 EPNs	6 EPNs	7 EPNs	8 EPNs	9 EPNs	10 EPNs	11 EPNs	Total
Ticktime												
5	2	1.92	1.96	1.92	1.92	2.08	2.63	2.92	3.21	3.71	4.67	**18.92**
10	2	2.36	2.6	3.2	3.08	3.04	3.08	3.72	4.04	5.08	6.68	**28.88**
15	2.96	2.96	2.96	3	3.58	3.58	3.96	4.38	5.25	6.38	8.54	**37.54**
20	2.96	3.13	3.25	3.75	4	4.29	4.67	5.46	6.29	8.04	10.38	**46.67**
25	3.36	3.64	3.92	4	4.56	4.96	5.44	6.32	7.4	9	12.68	**55.28**

Table 6.3: Cumulative lost TFs by ticktime/EPN ratio with a flat sample size for the Blacklist algorithm

Experiment two (2/12) using a cluster on Nikhef

Lost EPNs	1 EPN	2 EPNs	3 EPNs	4 EPNs	5 EPNs	6 EPNs	7 EPNs	8 EPNs	9 EPNs	10 EPNs	11 EPNs
Ticktime											
5	1	2	2	2	2	2	2	3	3	3	4
10	1	2	3	3	3	3	3	3	4	4	5
15	1	3	3	3	3	3	4	4	4	5	7
20	9	3	3	3	4	4	4	5	5	6	8
25	11	3	4	4	4	4	5	5	6	7	9

Table 6.4: Cumulative lost TFs by ticktime by lost EPNs (van der Heijden, 2018, p. 38)

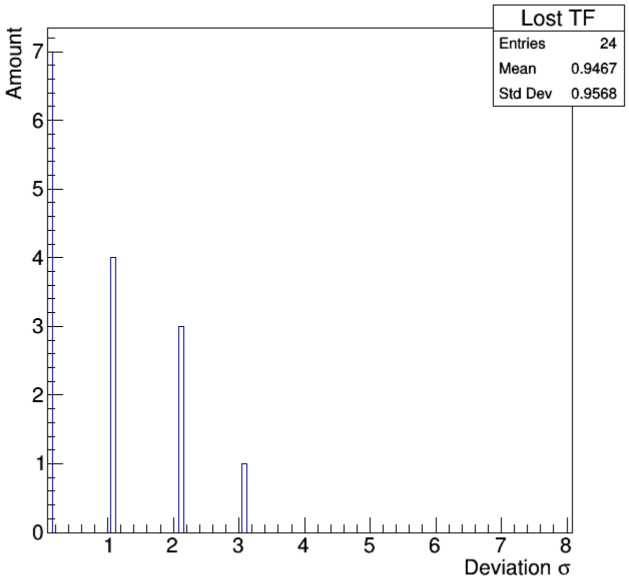


Figure 6.1: Histogram of the TF lost for ticktime 5

Ex.3 2 FLPs 12 EPNs

Ticktime influence on the Blacklist algorithm with all but one-fail over with random sample size.

Similar to the previous results, the TF loss to ticktime ratio rises in a linear line. Results can be seen in table 6.5 and table 6.6. A histogram for ticktime 5 can be seen at 6.2. It shows a standard deviation of 1.569 for all the TFs lost in the entire run.

Experiment three (2/12) using a cluster of Raspberry Pi's

Lost EPNs	1 EPN	2 EPNs	3 EPNs	4 EPNs	5 EPNs	6 EPNs	7 EPNs	8 EPNs	9 EPNs	10 EPNs	11 EPNs	Total
Ticktime												
5	1.92	1.96	1.96	2.2	2.52	2.88	2.96	3.08	3.56	4.2	4.92	**23.88**
10	1.96	2.3	2.52	2.74	2.81	2.93	2.85	3.48	3.85	4.74	6.33	**26.52**
15	3	2.92	2.96	2.96	3.21	3.38	4.04	4.42	5.25	6.29	8.67	**37.08**
20	3	3	3.38	3.5	3.92	4.17	4.79	5.17	6.08	7.54	10.88	**45.12**
25	3.71	3.54	3.92	3.83	4.13	4.79	5.33	6.13	7.21	9.17	13.38	**55.13**

Table 6.5: Cumulative lost TFs by ticktime/EPN ratio with a random sample size for the Blacklist algorithm

Experiment three (2/12) using a cluster on Nikhef

Lost EPNs	1 EPN	2 EPNs	3 EPNs	4 EPNs	5 EPNs	6 EPNs	7 EPNs	8 EPNs	9 EPNs	10 EPNs	11 EPNs
Ticktime											
5	1	2	2	2	2	3	3	3	3	4	4
10	1	3	3	3	3	3	3	4	4	5	6
15	11	3	3	3	3	3	4	4	5	6	7
20	10	3	3	3	4	4	4	5	6	7	9
25	13	3	4	4	4	4	5	5	7	8	10

Table 6.6: Cumulative TF data loss across events with the Blacklist algorithm and a random sample size (van der Heijden, 2018, p. 40)

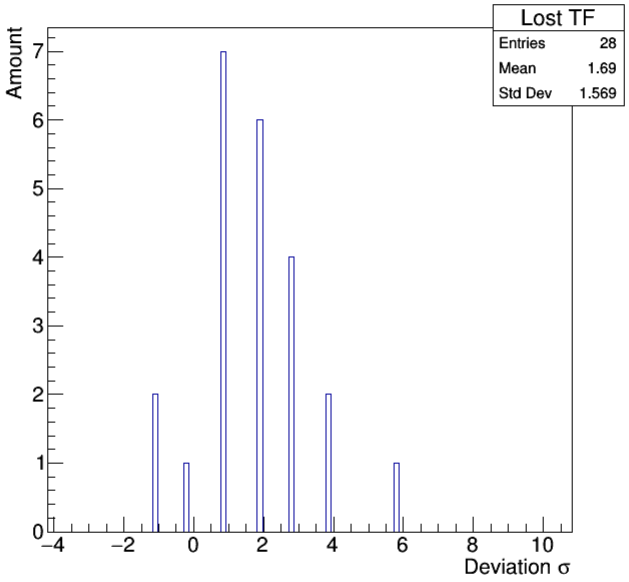


Figure 6.2: Histogram of the TF loss for ticktime 5

Ex.1 3 FLPs 18 EPNs

Ticktime influence on the Blacklist algorithm with one fail-over.

The results of the experiment are shown in table 6.7. It shows a linear line going up. At ticktime 10 there is only an average of 1 TF loss. This is the 1 TF that is lost when a shutdown is initialized. These results are lower compared to the results in table 6.1.

Experiment one (3/18) using a cluster of Raspberry Pi's

Ticktime	Mean TF loss	Standard Deviation
5	1.13	0.34
10	1	0
15	1.92	0.50
20	2.08	0.28
25	2.08	0.28
30	2.08	0.28
35	2.52	0.51
40	2.84	0.37

Table 6.7: Results of the TFs lost with 1 fail over using a cluster of Raspberry Pi's

Ex.2 3 FLPs 18 EPNs

Ticktime influence on the Blacklist algorithm with all but one failover.

The results of the experiment are shown in 6.8. It shows a lower TF loss compared to 6.3. It shows that at a ticktime of 5, every fail over will have a TF loss of < 2. This 1 TF comes from the shutdown signal. This means that the time between shutdown, and coming back around again to the EPN, is longer than the time needed to refresh the blacklist.

Experiment two (3/18) using a cluster of Raspberry Pi's

Lost EPNs	1 EPN	2 EPNs	3 EPNs	4 EPNs	5 EPNs	6 EPNs	7 EPNs	8 EPNs	9 EPNs	10 EPNs	11 EPNs	Total
Ticktime												
5	1.14	1.1	1.05	1.05	1.05	1.1	1.5	1	1.67	1.52	1.62	**3.33**
10	1.04	1.25	1.29	1.17	1.88	2.04	2	2.08	1.96	2.08	2.04	**8.83**
15	2	2.08	2.08	2.08	2.04	2	2.08	2.08	2.33	2.71	2.79	**14.29**
20	2.16	2.12	2.08	2.08	2.16	2.12	2.4	2.52	2.96	3.04	3.08	**16.72**
25	2.04	2.16	2.04	2.16	2.6	2.48	2.92	3.08	3.2	3.36	3.72	**19.76**

Table 6.8: Cumulative lost TFs by ticktime/EPN ratio with a flat sample size for the Blacklist algorithm

Ex.3 3 FLPs 18 EPNs

Ticktime influence on the Blacklist algorithm with all but one-fail over with random sample size.

The results of the experiment are shown in table 6.9. Oddly enough these results show a more sine wave in the data loss. All these results are again lower compared to 6.5.

Experiment three (3/18) using a cluster of Raspberry Pi's

Lost EPNs	1 EPN	2 EPNs	3 EPNs	4 EPNs	5 EPNs	6 EPNs	7 EPNs	8 EPNs	9 EPNs	10 EPNs	11 EPNs	Total
Ticktime												
5	2.56	2.39	2.22	1.17	1.22	2	1.11	2.17	1.44	1.33	1.72	**9.33**
10	2.88	2.88	2.38	1.08	1.71	2	1.54	2.25	2.08	2	2.29	**13.08**
15	3.08	4	3	2.04	2.04	2.58	2.08	3.04	2.33	2.25	2.54	**19**
20	3.28	4.04	3.24	2.08	2.12	3.12	2.16	3.16	3	2.88	3.08	**22.16**
25	3.7	4.09	3.87	2.52	2.78	3.13	2.83	3.65	3.39	3.3	3.96	**27.22**

Table 6.9: Cumulative lost TFs by ticktime/EPN ratio with a random sample size for the Blacklist algorithm

Ex.1 4 FLPs 24 EPNs

Ticktime influence on the Blacklist algorithm with one fail-over.

Experiment one (4/24) using a cluster of Raspberry Pi's

Ticktime	Mean TF loss	Standard Deviation
5	1	0
10	1.08	0.28
15	1.76	0.53
20	2.12	0.44
25	2	0
30	2	0
35	2.16	0.37
40	2.33	0.48

Table 6.10: Results of the TFs lost with 1 fail over using a cluster of Raspberry Pi's

Ex.2 4 FLPs 24 EPNs

Ticktime influence on the Blacklist algorithm with all but one fail-over.

Table 6.10 shows the results of the experiment. It shows a the same linear line going up per EPN as in the previous experiments. These results are yet again smaller than the results from table 6.7.

Experiment two (4/24) using a cluster of Raspberry Pi's

Lost EPNs	1 EPN	2 EPNs	3 EPNs	4 EPNs	5 EPNs	6 EPNs	7 EPNs	8 EPNs	9 EPNs	10 EPNs	11 EPNs	Total
Ticktime												
5	1.15	1.15	1.25	1.15	1.05	1.15	1.15	1.15	1.1	1.1	1.25	**2.65**
10	1.08	1.04	1.08	1.08	1.48	1.48	1.52	1.8	2.04	1.96	2.24	**6.8**
15	1.4	1.68	1.72	2	2.04	2.04	2.08	2	2.08	2	2.08	**11.12**
20	2.12	2.04	2.08	2	2.16	2.12	2.08	2.08	2.12	2.08	2.24	**13.12**
25	2.12	2.12	2.16	2.12	2.12	2.04	2.2	2.08	2.6	2.68	2.64	**14.88**

Table 6.11: Cumulative lost TFs by ticktime/EPN ratio with a flat sample size for the Blacklist algorithm

Ex.3 4 FLPs 24 EPNs

Ticktime influence on the Blacklist algorithm with all but one fail-over with random sample size.

Experiment three (4/24) using a cluster of Raspberry Pi's

Lost EPNs	1 EPN	2 EPNs	3 EPNs	4 EPNs	5 EPNs	6 EPNs	7 EPNs	8 EPNs	9 EPNs	10 EPNs	11 EPNs	Total
Ticktime												
5	1.1	2.14	1.14	1.57	1.24	2.14	1.14	2	2.1	1.48	2.05	**8.1**
10	1.52	1.96	1.04	1.68	1.4	2.04	1.36	1.88	2.04	2.04	2.16	**9.12**
15	2.08	2.04	1.04	2.16	2.16	2.4	2.12	1.92	2.12	2.08	2.28	**12.4**
20	2.04	2.2	1.48	2.32	2.16	2.88	2.04	2.8	2.96	2.2	2.96	**16.04**
25	2.13	2.54	2.08	2.33	2.33	3.13	2.13	2.96	3	2.92	3.04	**18.58**

Table 6.12: Cumulative lost TFs by ticktime/EPN ratio with a random sample size for the Blacklist algorithm

Ex.4 4 FLPs 24 EPNs

Ticktime influence on the Blacklist algorithm with a fixed cluster fail-over pattern using a random sample size.

Table 6.13 shows the results of the experiment. It shows that for the first four clusters the TF loss is low, but for the last cluster it's substantially higher. Because of this, the total average is also a lot higher.

Ex.5 4 FLPs 24EPNs

Ticktime influence on the Blacklist algorithm with a random cluster fail-over pattern using a random sample size

Experiment four (4/24) using a cluster of Raspberry Pi's

Lost Clusters	1 Cluster	2 Clusters	3 Clusters	4 Clusters	5 Clusters	Total
Ticktime						
5	2.83	5.25	1.88	4.67	250.67	**261.29**
10	3.64	3.76	6.04	4.52	285.92	**299.88**
15	5.6	2.04	6.28	7.76	277.68	**295.36**
20	7.26	4.74	6.26	13.3	296.09	**323.65**
25	7.44	8.4	8.12	9.44	254.72	**284.12**

Table 6.13: Cumulative lost TFs by ticktime/EPN ratio with a random sample size for the Blacklist algorithm and a fixed cluster fail-over pattern

Table 6.14 shows the result of the experiment. It shows that with a random cluster fail-over pattern, the total average TF loss is close to the results in table 6.13, but for each seperate cluster they are higher. Unlike the previous experiment where it's all lower numbers first and then finally a very large one.

Experiment five (4/24) using a cluster of Raspberry Pi's

Lost Clusters	1 Cluster	2 Clusters	3 Clusters	4 Clusters	5 Clusters	Total
Ticktime						
5	3.21	23.53	38.32	79.53	174.89	**315.47**
10	4.48	21.92	35.36	40.44	185.04	**283.24**
15	5.52	23	36.84	28.16	188.4	**277.92**
20	5.68	24.8	39.6	42.52	190.64	**299.24**
25	6.26	26.74	42.83	84.61	194.39	**350.83**

Table 6.14: Cumulative lost TFs by ticktime/EPN ratio with a random sample size for the Blacklist algorithm and a random cluster fail-over pattern

6.2 Analysis

6.2.1 Comparison to the previous experiment

To calculate the validity of the results, two methods will be used. An analysis of the trend line will be done, and the Pearson correlation will be used. To calculate the trend line, the following equations are used:

Slope =

$$\alpha = \frac{n\sum(xy) - \sum(x)\sum(y)}{n\sum(x^2) - (\sum(x))^2}$$

Intercept =

$$\beta = \frac{\sum(y) - \alpha\sum(x)}{n}$$

Trend line =

$$y = \alpha x + \beta$$

Figure 6.3, 6.4 and 6.5 show graphs of both data sets combined. Table 6.15 shows every experiments slope and intercept, and also the ticktime where the first TF loss would be > 1, and how much the ticktime would have to increase after that to be + 1 TF again.

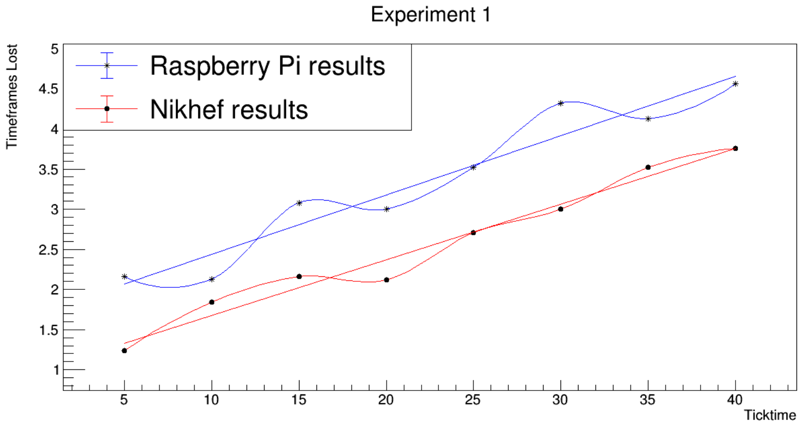


Figure 6.3: Combined results Experiment 1

	Slope Pi	Intercept Pi	Slope Nikhef	Intercept Nikhef	First > 1	Next + 1
Experiment 1	0.074	1.7	0.069	0.982	64	222
Experiment 2	1.810	10.305	1.794	4.077	-	64
Experiment 3	1.622	13.216	1.783	9.072	-	6

Table 6.15: Slope and Intercepts combined experiments

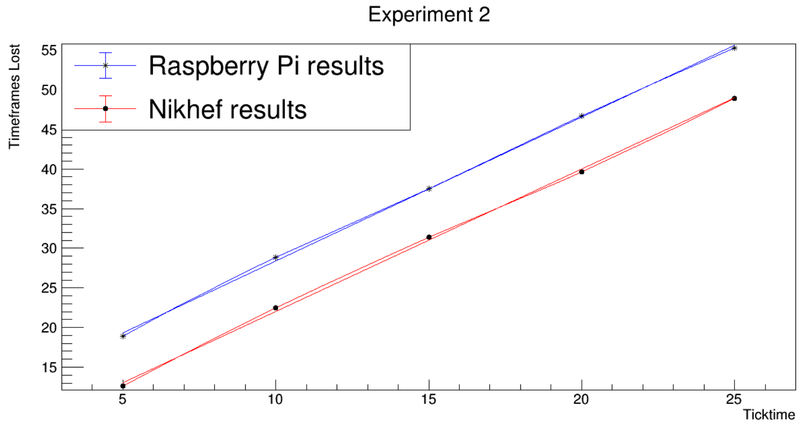


Figure 6.4: Combined results Experiment 2

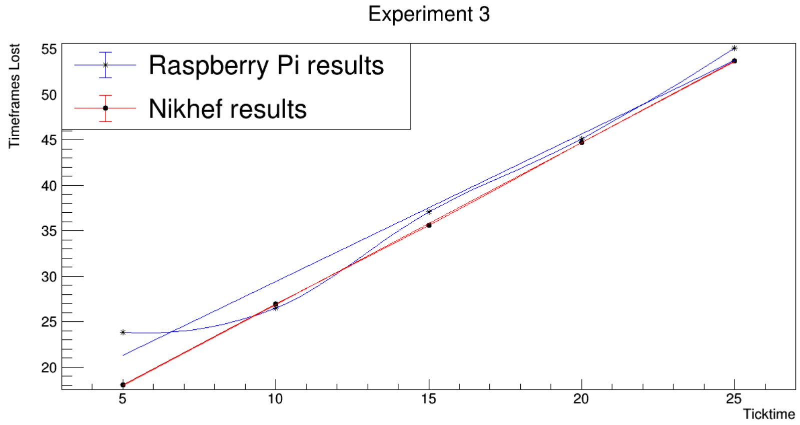


Figure 6.5: Combined results Experiment 3

These figures show that the for the first experiment, the ticktime has to increase by 222 in order for the difference in TF loss be + 1 again. For experiment two this is 64, and for experiment 3 it's the other way around, where the results from Nikhef will end up higher than from the Raspberry Pi. The first time this happens is at ticktime 32. This means that for experiment one and two, the results are closely related to one another. For experiment three the results are a bit different. For every ticktime below 30 the results are closely related. After ticktime 30 the results would have to be further analyzed. The previous experiment already deemed a ticktime above 30 to be less viable as from that point the TF loss will only go up, which is what the algorithm should try to prevent.

The Pearson correlation is calculated as followed:

$$\rho x, y = \frac{\sum(x - m_x)(y - m_y)}{\sqrt{\sum(x - m_x)^2(y - m_y)^2}}$$

Where m_x is the mean of vector x, and m_y is the mean of vector y.
A way to calculate this in python is shown in listing 6.1 The results of this can be found in table 6.16. The totals of all separate tests are combined to calculate this correlation. If the Pearson correlation is close to 1, it means that there is a strong positve linear correlation. If it's 0 then there is no linear correlation. If it's -1 then there is a strong negative linear correlation. In order to deny the null hypothesis, as in there is no correlation between the two experiments, with a 0.1% error margin, the correlation value for experiment 1 would have to be >= 0.742, for experiment 2 >=0.872 and for experiment >=0.872(Table of Critical Values for r, n.d.)

Listing 6.1: Pearson correlation calculation in Python

```
from scipy.stats.stats import pearsonr
def list of values X
def list of values Y
pearsonr(X,Y)
```

Experiment	Pearson Correlation
1	0.95
2	0.99
3	0.99

Table 6.16: Pearson correlation between the raspberry pi cluster and the cluster on Nikhef

These correlations all strongly deny the null hypothesis, and thus there is some sort of statistical relationship. They also are close to 1, which means they are close to having a total positive linear correlation. Thus concluded that there is indeed a correlation between the setup on Nikhef, and the setup using Raspberry Pi's.

6.2.2 Increasing the numbers of FLPs and EPNs

Differences between 2/12 and 3/18

Table 6.17 shows that over all the ticktimes that are measured, there is an average difference of 1.41. Table 6.18 shows that this difference is 24.87, and table 6.19 shows that it is 19.39. This means that for every experiment conducted there are fewer TFs lost when the amount of units are increased.

Experiment 1	2/12	3/18	Nominal difference	% difference
Ticktime				
5	2.16	1.13	1.03	-47.68
10	2.13	1	1.13	-53.05
15	3.08	1.92	1.16	-37.66
20	3	2.08	0.92	-30.67
25	3.52	2.08	1.44	-40.91
30	4.32	2.08	2.24	-51.85
35	4.13	2.52	1.61	-38.98
40	4.56	2.84	1.72	-37.72
Average			**1.41**	**-41.82**

Table 6.17: Differences for experiment one using 2/12 and 3/18 ratio's

Experiment 2	2/12	3/18	Nominal difference	% difference
Ticktime				
5	18.92	3.33	15.59	-82.4
10	28.88	8.83	20.05	-68.43
15	37.54	14.29	23.25	-61.93
20	46.67	16.72	29.95	-64.17
25	55.28	19.76	35.52	-64.25
Average			**24.87**	**-66.4**

Table 6.18: Differences for experiment two using 2/12 and 3/18 ratio's

Experiment 3	2/12	3/18	Nominal difference	% difference
Ticktime				
5	23.88	9.33	14.55	-60.93
10	26.52	13.08	13.44	-50.68
15	37.08	19	18.08	-48.76
20	45.12	22.16	22.96	-50.89
25	55.13	27.22	27.91	-50.63
Average			**19.39**	**-51.64**

Table 6.19: Differences for experiment three using 2/12 and 3/18 ratio's

Differences between 3/18 and 4/24

Table 6.20 shows that over all the ticktimes that are measured, there is an average difference of 0.15. Table 6.21 shows that this difference is 2.87, and table 6.22 shows that it is 5.31. These results yet again shows fewer TFs lost when the units are increased, though not as many as with the previous increase. Further increasing the units should inevitably level the amount of TFs lost to 1.

Experiment 1	3/18	4/24	Nominal difference	% difference
Ticktime				
5	1.13	1	0.13	-11.5
10	1	1.08	-0.08	8
15	1.92	1.76	0.16	-8.33
20	2.08	2.12	-0.04	1.92
25	2.08	2	0.08	-3.85
30	2.08	2	0.08	-3.85
35	2.52	2.16	0.36	-14.29
40	2.84	2.33	0.51	-17.96
Average			**0.15**	**-7.67**

Table 6.20: Differences for experiment one using 3/18 and 4/24 ratio's

Experiment 2	3/18	4/24	Nominal difference	% difference
Ticktime				
5	3.33	2.65	0.68	-20.42
10	8.83	6.8	2.03	-22.99
15	14.29	11.12	3.17	-22.18
20	16.72	13.12	3.6	-21.53
25	19.76	14.88	4.88	-24.7
Average			**2.87**	**-22.82**

Table 6.21: Differences for experiment two using 3/18 and 4/24 ratio's

Experiment 3	3/18	4/24	Nominal difference	% difference
Ticktime				
5	9.33	8.1	1.23	-13.18
10	13.08	9.12	3.96	-30.28
15	19.0	12.4	6.6	-34.74
20	22.16	16.04	6.12	-27.62
25	27.22	18.58	8.64	-31.74
Average			**5.31**	**-29.24**

Table 6.22: Differences for experiment three using 3/18 and 4/24 ratio's

6.2.3 Comparing cluster fail-over patterns

When comparing the data between the experiments, the t-test method is used. A way to implement a t-test analysis in Python is shown in listing 6.2. For every increase in units the t-test will be performed.

Listing 6.2: t-test analysis in Python

```
from scipy.stats import ttest_ind
def list of values X
def list of values Y
print ttest_ind(X, Y, equal_var=False)
```

The results are shown in 6.23. It shows that the t-statistic is -0.75 and the p-value is 0.474. This means that there is no significant statistical difference. Looking at the results it shows that that experiment four has lower values for the first four cluster fail-overs, but then has a higher value for the last cluster fail-over, whereas experiment five has higher values overall for each cluster fail-over. This shows that there is a difference in the lost TFs per individual cluster that has a fail-over depending on the layout, but that in the end the total amount is still similar.

Ticktime	Experiment 4	Experiment 5	t-statistic	p-value
5	261.29	315.47		
10	299.88	283.24		
15	295.36	277.92		
20	323.65	299.24		
25	284.23	350.83		
Total			**-0.75**	**0.474**

Table 6.23: t-test results of experiment four and five

Chapter 7

Conclusion and Recommendations

7.1 Conclusion

The new setup is a viable way of conducting experiments for testing load balancing algorithms. With a Pearson correlation value of 0.95 and higher for the three experiments, we can conclude that there is a very strong positive linear correlation. Further more looking at the data for the expansion of the experiment, we can see that the Information Node does not have a negative effect with handling the extra units. In fact, because it has more time to handle the units in between round robins, less TFs are lost in the process.

Lastly, there is a difference in the amount of TFs lost depending on the layout of the system. Using the Blacklist algorithm, each seperate cluster fail-over will have a higher TF loss if the IPs of the machines that have a fail-over are more spread out, as compared to the IPs being one after another.

7.2 Recommendations

7.2.1 Increase the unit count even further

Due to time constraints, this experiment stopped at a 4/24 ratio. Since this prototype is more easily expanded it is recommended to increase it even further to check wether the TF loss would eventually level out, have an asymptote at some point, or turns around back up again and becomes a sine.

7.2.2 Do the experiment with Ansible induced fail-overs

All these experiments are done with fail-overs that were induced using the TF that was send at that point. This means that the fail-overs are in a sense predictable, and means that there will always be atleast one TF lost. It is recommended to create scripts that induce a fail-over using Ansible at random time intervals. This to prevent having the one certain TF loss, and to randomize the moment when an EPN will have a fail-over, as this would be a more realistic simulation.

7.2.3 Use a lowest latency approach, as opposed to a round robin for the distribution

Currently the distribution is done using a Round Robin approach. This means that the distribution will go from the top of the list, to the bottom each time. This means that if there is a fail-over induced by a TF, it will have the entire list still as a buffer to be targeted again. If there is a lowest latency approach used, then the Information Node will target the EPN that is best suited at the time to receive the TF. If there is a fail-over happening at some point, then this EPN won't be targeted anymore by default.

Chapter 8

Bibliography

Ansible, Red Hat Ansible, n.d., IT Automation with Ansible, online accessed at April 25, retrieved from
https://www.ansible.com/overview/it-automation

Apache Zookeeper, 2017, online accessed at July 2, retrieved from
http://zookeeper.apache.org/

CERN, 2012, About CERN, online accessed at April 25, retrieved from
https://home.cern/about

CERN, 2012, ALICE, online accessed at April 25, retrieved from
https://home.cern/about/experiments/alice

FairRoot, n.d., About FairRoot, online accessed at April 24, retrieved from
https://fairroot.gsi.de/?q=about

FairRootGroup, 2018, FairRootGroup/FairMQ, online accessed at April 24, retrieved from
https://github.com/FairRootGroup/FairMQ

FairRootGroup, 2018, FairRootGroup/FairRoot, online accessed at April 24, retrieved from
https://github.com/FairRootGroup/FairRoot

FairRootGroup, 2018, Make a special repository for FairMQ, online accessed at April 24, retrieved from
https://github.com/FairRootGroup/FairRoot/issues/736

How to calculate Trendline, June 25 2018, Classroom, online accessed at July 2, retrieved from
https://classroom.synonym.com/calculate-trendline-2709.html

Juniper, 2017, EX3400 Ethernet Switch, online accessed at May 9, retrieved from
https://www.juniper.net/assets/us/en/local/pdf/datasheets/1000581-en.pdf

Raspberry, 2018, About us, online accessed at April 26, retrieved from
https://www.raspberrypi.org/about/

Raspberry, 2018, Raspberry Pi 3 Model B+, online accessed at April 26, retrieved from
https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

SoftwareForScience, 2018, SoftwareForScience/O2-Balancer, online accessed at April 23, retrieved from
https://github.com/SoftwareForScience/O2-Balancer

StatSoft Inc, 2013, Electronic Statistics Textbook. online accessed at May 23, retrieved from
http://www.statsoft.com/textbook/statistics-glossary/p# Pearson%20Correlation

Table of Critical Values for Pearson's r, online accessed at June 27, retrieved from
https://www.radford.edu/jaspelme/statsbook/Chapter%20files/Table_of_Critical_Values_for_r.pdf

Technical design report for the upgrade of the online offline computing system, 2015, The ALICE Collaboration

Tp-Link, 2017, 5-Port 10/100Mbps Desktop Switch TL-SF1005D, online accesed at May 9, retrieved from
https://www.tp-link.com/us/products/details/cat-5581_TL-SF1005D.html

Appendices

A Glossary

Term	Abbreviation	Meaning
CERN		European association for nuclear research
LHC	Large Hardon Collider	Largest particle collider at CERN used for researching matter
ALICE	A Large Ion Collider Experiment	Detector on the LHC for detecting quark-gluon plasma
IN	Information Node	Computer that runs manages all the underlying computers
FLP	First Level Processor	The first computer in line for receiving data
EPN	Event Processing Node	Last computer in line that stores all the data received
STF	Sub Time Frame	Data from the detector
TF	Time Frame	Compressed data from 2 STFs
Fail-Over		An event where an EPN is disabled
FairRoot		A simulation framework based on ROOT
FairMQ		The data transport layer of FairRoot
Zookeeper		A service for maintaining information and configurations of units
Ticktime		The time in ms that Zookeeper uses to update
Ansible		Deployment software for IT infrastructures
Blacklist		An algorithm that maintains a list of targets to work with

B Summary in Dutch

Eén van de detectoren op de Large Hadron Collider bij CERN is genaamd ALICE, wat staat voor A Large Ion Collider Experiment. Her primaire doel van ALICE is voor de detectie van botsingen, en dan vooral de botsingen tussen lood deeltjes. Wanneer deze deeltjes botsen worden ze extreem heet, en breken ze af in een materie die quark-gluon plasma heet. ALICE detecteerd deze botsingen en registreerd de data ervan.

Aan het eind van 2018 gaa ALICE offline voor een twee jaar durende upgrade aan de software. Een deel van deze upgrade is een Load Balancing algoritme. Tijdens een run van ALICE komt er een data stream van 1.1TB/S. Het algoritme is bedoeld voor de efficiënte distributie van deze datas naar 1750 computers. Deze computers zijn onderverdeeld in 250 First Level Processors, en 1500 Event Processing Nodes. Deze computers zorgen ervoor dat de data gecomprimeerd wordt en wordt opgeslagen zodat het later gereconstrueerd kan worden.

Deze data komt in heartbeats. Een heartbeat is elke 20ms. De FLPs pakken de data tussen twee heartbeats en comprimeerd deze in een Sub Time Frame. Deze STFs worden dan doorgestuurd naar de EPNs. Hier worden ze weer verder gecomprimeerd in een Time Frame, en worden ze opgeslagen. Wanneer een EPN uitvalt moet die geen data meer ontvangen en zal hij dus van de datalijn af moeten. Wanneer dit gebeurt heet dat een fail-over. De distributie van deze data heet Load Balancing.

In een vorig experiment dat uitgevoerd was door Heiko van der Heijden kwam eruit dat een Blacklist algoritme een efficiënter algoritme is voor deze load balancing applicatie. Dit experiment was uitgevoerd met een cluster aan computers die in bruikleen waren van Nikhef. Tijdens dit experiment werd er gebruik gemaakt van 16 computers, waarvan 12 de rol van EPN hadden, en 2 de rol van FLP. Deze ratio van FLP:EPN is gelijk aan de ratio die gebruikt wordt bij CERN. De bruikbaarheid van deze cluster was niet gegarandeerd, en er was dus de vraag naar een cluster die bruikbaarder was, maar die ook makkelijker uitgebreid kon worden. Dit resulteerde in een cluster van Raspberry Pi's.

Om deze cluster te valideren tegen de cluster die gebruikt was op Nikhef, moest er een correlatie gevonden worden tussen de resultaten van de twee clusters. Als deze correlatie er is, dan betekent dat, dat het het aantal van de FLPs en EPNs uitgebreid kan worden om te kijken of het algoritme bij een hoger aantal nog steeds hetzelfde werkt.

De software die gebruikt wordt bestaat grotendeels uit twee frameworks, FairMQ en Zookeeper. FairMQ is de data transport laag van een groter framework genaamd FairRoot. Voor dit experiment is een speciaal gemaakte aftakking gebruikt van FairMQ omdat de rekenkracht van de Raspberry Pi lager ligt dan die van de cluster van Nikhef. Zookeeper regelt de Blacklist in het algoritme. Zookeeper regelt de registratie van EPNs die online of offline zijn en stuurt dit door naar de FLPs zodat die weten naar welke EPNs ze data kunnen versturen. Zookeeper loopt op een aparte computer genaamd de Information Node.

In eerste instantie moest de nieuwe cluster gemaakt worden. De complicaties waren vooral het feit dat de Raspberry Pi maar één Ethernet poort heeft stan-

daard, en dat alle software omgebouwd moest worden om te werken met een ARM architectuur. Hierna moesten dezelfde drie experimenten uitgevoerd worden die tijdens het vorige experiment ook uitgevoerd waren. Tijdens het eerste experiment wordt er één EPN uitgeschakeld. Het tweede experiment schakelt elke EPN uit op één na, en het derde experiment schakelt elke EPN uit op één na maar gebruikt ook een random waarde voor de hoeveelheid data. Deze resulten zijn toen vergeleken met de resulltaten van het vorige onderzoek om te kijken naar een correlatie tussen de data. Hierna werden de drie experimenten opnieuw uitgevoerd met hoger aantal FLPs en EPNs (3:18 en 4:24) om te kijken of het uitbreiden van het experiment een effect heeft op het algoritme en de Information Node. Op het laatst is er ook een extra experiment uitgevoerd waarbij meerdere EPNs tegelijkertijd uitviel. Het doel van deze experimenten was om te kijken of de lijst van EPNs een invloed had op het data verlies. Het eerste experiment liet vier EPNs uitvallen die achter elkaar in de lijst stonden. Het tweede experiment liet vier EPNs uitvallen op willekeurige plekken in de lijst.
De conclusie is, is dat het nieuwe prototype een valide alternatief is. De drie experimenten hadden onderling een Pearson correlatie van 0,95; 0,99 en 0,99. Deze correlatie waardes spreken allemaal de nul hypothese tegen dat er geen correlatie is tussen de clusters. Daarnaast liet de uitbreiding van het experiment zien dat het aantal data verlies aleen maar dichterbij 1 kwam, wat gelijk staat aan het minimale data verlies met hoe de setup is op het moment. Het laatste experiment liet zien dat het data verlies voor individuele clusters die uitgeschakeld werden verschillend was, maar dat het totale data verlies weer gelijk aan elkaar was.
Het is aanbevolen om het experiment opnieuw uit te voeren met nog meer FLPs en EPNs om te kijken of het data verlies uiteindelijk 1 wordt, om te kijken of er misschien een asymptoot is ergens, of om te kijken of het misschien weer omhoog gaat en er een sinus vorming ontstaat. Daarnaast is het aanbevolen om deze experimenten voortaan uit te voeren met fail-overs die geïnitieerd worden door Ansible in plaats van Time Frames. Hiermee elimineer je de voorspelbaarheid van de fail-overs en elimineer je het verlies van de Time Frame die je altijd kwijt bent met de huidige setup. Tenslotte is het aanbevolen om te kijken naar een lowest latency aanpak voor de data distributie tegenover een round robin die nu gebruikt wordt.

C Planning

Planning as of 25/04/2018

Name	Description	Deliverable	Est.	Complete
Create scripts for all experiments	Create all the scripts for the deployment of experiments	Experiment scripts	6 hours	19/04/2018
Experiment 1 (2/12)	Ticktime influence on the blacklist algorithm with one fail-over	Compressed Ex.1 2/12 results	105 hours	30/04/2018
Experiment 2 (2/12)	Ticktime influence on the blacklist algorithm with all but one fail-over	Compressed Ex.2 2/12 results	19 hours	01/05/2018
Work out Ex.1 (2/12)	Compile results of Ex.1 (2/12)	Compiled results of Ex.1 (2/12)	2 hours	30/04/2018
Write chapter 6.1	Write down the results for chapter 6.1 about the results of Ex.1 (2/12)	Chapter 6.1	3 hours	30/04/2018
Experiment 3 (2/12)	Ticktime influence on the blacklist algorithm with all but one-fail over with random sample size	Compressed Ex.3 2/12 results	19 hours	02/05/2018
Work out Ex.2 (2/12)	Compile results of Ex.2 (2/12)	Compiled results of Ex.2 (2/12)	2 hours	01/05/2018
Write chapter 6.2	Write down the results for chapter 6.2 about the results of Ex.2 (2/12)	Chapter 6.2	3 hours	01/05/2018
Experiment 2 (3/18)	Ticktime influence on the blacklist algorithm with all but one fail-over	Compressed Ex.2 3/18 results	19 hours	03/05/2018
Work out Ex.3 (2/12)	Compile results of Ex.3 (2/12)	Compiled results of Ex.3 (2/12)	2 hours	02/05/2018

Write chapter 6.3	Write down the results for chapter 6.3 about the results of Ex.3 (2/12)	Chapter 6.3	3 hours	02/05/2018
Experiment 2 (4/24)	Ticktime influence on the blacklist algorithm with all but one fail-over	Compressed Ex.2 4/24 results	19 hours	04/05/2018
Work out Ex.2 (3/18)	Compile results of Ex.2 (3/18)	Compiled results of Ex.2 (3/18)	2 hours	03/05/2018
Write chapter 6.5	Write down the results for chapter 6.5 about the results of Ex.2 (3/18)	Chapter 6.5	3 hours	03/05/2018
Experiment 3 (3/18)	Ticktime influence on the blacklist algorithm with all but one-fail over with random sample size	Compressed Ex.3 3/18 results	19 hours	09/05/2018
Work out Ex.2 (4/24)	Compile results of Ex.2 (4/24)	Compiled results of Ex.2 (4/24)	2 hours	04/05/2018
Write chapter 6.8	Write down the results for chapter 6.8 about the results of Ex.2 (4/24)	Chapter 6.8	3 hours	04/05/2018
Experiment 3 (4/24)	Ticktime influence on the blacklist algorithm with all but one-fail over with random sample size	Compressed Ex.3 4/24 results	19 hours	10/05/2018
Work out Ex.3 (3/18)	Compile results of Ex.3 (3/18)	Compiled results of Ex.3 (3/18)	2 hours	09/05/2018
Write chapter 6.6	Write down the results for chapter 6.6 about the results of Ex.3 (3/18)	Chapter 6.6	3 hours	09/05/2018
Experiment 1 (3/18)	Ticktime influence on the blacklist algorithm with one fail-over	Compressed Ex.1 3/18 results	105 hours	14/05/2018

Work out Ex.3 (4/24)	Compile results of Ex.3 (4/24)	Compiled results of Ex.3 (4/24)	2 hours	11/05/2018
Write chapter 6.9	Write down the results for chapter 6.9 about the results of Ex.3 (4/24)	Chapter 6.9	3 hours	11/05/2018
Experiment 1 (4/24)	Ticktime influence on the blacklist algorithm with one fail-over	Compressed Ex.1 4/24 results	105 hours	19/05/2018
Work out Ex.1 (3/18)	Compile results of Ex.1 (3/18)	Compiled results of Ex.1 (3/18)	2 hours	15/05/2018
Write chapter 6.4	Write down the results for chapter 6.4 about the results of Ex.1 (3/18)	Chapter 6.4	3 hours	15/05/2018
Compile + write appendixes	Compile and write down everything that needs to go in the appendix, which include the bibliography, email documentation, and glossary	Appendixes	4 hours	16/05/2018
Work out Ex.1 (4/24)	Compile results of Ex.1 (4/24)	Compiled results of Ex.1 (4/24)	2 hours	21/05/2018
Write chapter 6.7	Write down the results for chapter 6.7 about the results of Ex.1 (4/24)	Chapter 6.7	3 hours	21/05/2018
Write a summary in 2/3 languages	Write a summary of the thesis in English, Dutch and German	Summaries	16 hours	18/05/2018
Experiment 4 (4/24)	Ticktime influence on the blacklist algorithm with all but one fail-over at once	Compressed Ex.4 4/24 results	19 hours	22/05/2018
Work out Ex.4 (4/24)	Compile results of Ex.4 (4/24)	Compiled results of Ex.4 (4/24)	2 hours	23/05/2018

Write chapter 6.10	Write down the results for chapter 6.10 about the results of Ex.4 (4/24)	Chapter 6.10	3 hours	23/05/2018
Write a preface	Write a small preface	Preface	1 hours	24/05/2018
Write an introduction	Write an introduction for the thesis	Introduction	2 hours	24/05/2018
Write chapter 7.1	Write down the conclusions based on the results of the previous experiments	Chapter 7.1	3 hours	25/05/2018
Write chapter 7.2	Write down the recommendations based on the results of the previous experiments	Chapter 7.2	3 hours	25/05/2018

Planning as of 22/05/2018

Day	Activity	Description	Deliverable	Complete
22/05/2018	Remake a planning for the final period	Remake the planning for the thesis to be done before 11/05/2018	Planning as of 22/05/2018	22/05/2018
23/05/2018	Take a more scientific approach to the results of the experiments	Make some scripts to calculate the pearson correlation between the 2 data sets	Paragraph 4.4.1	23/05/2018
24/05/2018				
25/05/2018				
28/05/2018	Configure new clusters to work with the old network	Install the SD cards and configure hostnames and IP addresses.	New clusters as part of the network	28/05/2018
29/05/2018	Test the new clusters	Test the clusters with several small deployment tests	Usable clusters	29/05/2018
30/05/2018	Start experiment 1 (3/18)	-	Raw data from experiment 1 (3/18)	31/05/2018
31/05/2018	Compile all the data of experiment 1 (3/18)	Look at the data from the experiment and write them down in tables	Compiled data experiment 1 (3/18)	31/05/2018
01/06/2018	Start experiment 2 (3/18)	-	Raw data from experiment 2 (3/18)	02/06/2018
04/06/2018	Compile all the data of experiment 2 (3/18)	Look at the data from the experiment and write them down in tables	Compiled data experiment 2 (3/18)	04/06/2018
05/06/2018	Start experiment 3 (3/18)	-	Raw data from experiment 3 (3/18)	06/06/2018

06/06/2018	Compile all the data of experiment 3 (3/18)	Look at the data from the experiment and write them down in tables	Compiled data experiment 3 (3/18)	06/06/2018
07/06/2018	Configure new clusters to work with the old network	Install the SD cards and configure hostnames and IP addresses.	New clusters as part of the network	07/06/2018
08/06/2018	Test the new clusters + Start experiment 1 (4/24)	Test the clusters with several small deployment tests	Usable clusters + Raw data from experiment 1 (3/18)	09/06/2016
11/06/2018	Compile all the data of experiment 1 (4/24)	Look at the data from the experiment and write them down in tables	Compiled data experiment 1 (4/24)	11/06/2018
12/06/2018	Start experiment 2 (4/24)	-	Raw data from experiment 2 (4/24)	13/06/2018
13/06/2018	Compile all the data of experiment 2 (4/24)	Look at the data from the experiment and write them down in tables	Compiled data experiment 2 (4/24)	13/06/2018
14/06/2018	Start experiment 3 (4/24)	-	Raw data from experiment 3 (4/24)	15/06/2018
15/06/2018	Compile all the data of experiment 3 (4/24)	Look at the data from the experiment and write them down in tables	Compiled data experiment 3 (4/24)	15/06/2018
18/06/2018	Fly to Switzerland for CERN	-	-	-
19/06/2018	Go to CERN for the day	-	-	-
20/06/2018	Spend the day in Geneve	-	-	-

21/06/2018	Write scripts to analyze all the data from previous experiments	-	Scripts	21/06/2018
22/06/2018	Start writing the summary in English	Start writing a summary till atleast the results	70 percentile summary	22/06/2018
25/06/2018	Analyze data from previous experiments	Analyze all the data from the experiment and the relations with them	Conclusions of the experiments	25/06/2018
26/06/2018	Write down conclusions	Write down all the conclusions and recommendations	Conclusions chapter	26/06/2018
27/06/2018	Finish the summary in English	Finish the summary with the conclusions and recommendations	Summary	27/06/2018
28/06/2018	Proof read everything for grammar and spelling errors	Correct all mistakes for a final version of the thesis	-	28/06/2018
29/06/2018	Finalize everything and send a version to Marten Teitsma, Heiko van der Heijden and Kees Rijsenbrij	-	-	29/06/2018
02/07/2018	Translate the summary in Dutch	Translate the summary in Dutch as an appendix	Dutch summary	02/07/2018
03/07/2018	Attempt a translation in German	Translate the summary in German as an appendix	German summary	03/07/2018
04/07/2018				

05/07/2018	RETRIEVE FEEDBACK	Day I would like to have some feedback points to process	Feedback	05/07/2018
06/07/2018				
09/07/2018	Finalize everything with feedback	-	Version 1.1	09/07/2018
10/07/2018	Send version 1.1 to Marten Teitsma, Heiko van der Heijden and Kees Rijsenbrij	-	-	10/07/2018