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	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

Contents

Summary

Chapter 1

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.

1.1 CERN

CERN is a European organization for nuclear research, situated in Geneve Switserland. CERN was founded in 1954 and is one of Europe's first join 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. One of these detectors is ALICE

1.2 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 matter that appeared at the very start of the big bang, and is the thing that binds the gluons and quarks together. CERN wants to observe this matter. The way they achieve this, is by shooting two lead ions against each other. This produces a 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.

1.2.1 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)

1.3 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 268 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 would be monitored by an Information Node.

1.4 Research

This research is a continuation of a previous research done by Heiko van der Heijden. His results show that of the two algorithms tested, Re-initialization and Blacklist, that the Blacklist algorithm has fewer Time Frames lost.
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.

1.5 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. This decision has been made because of an unfortunate failure of communication from Nikhef about the availability of the cluster for this research period. 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.

An overview can be seen in figure 1.1

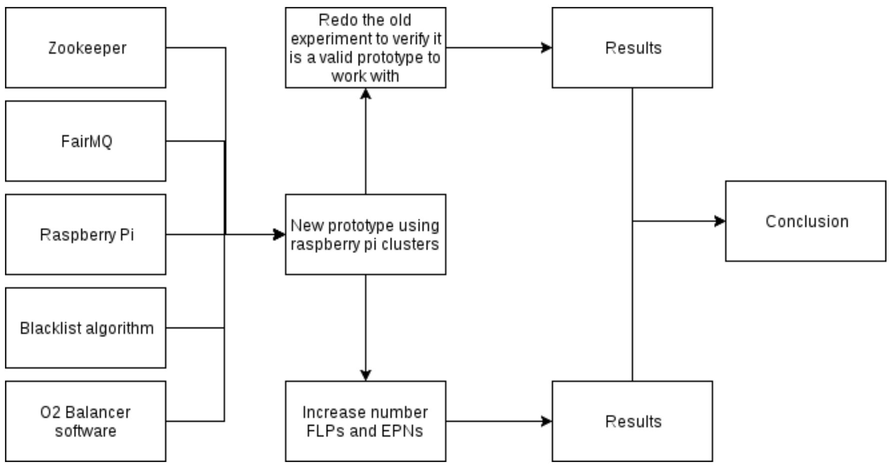

Figure 1.1: Research model

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 2

Framework

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

2.1 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.

2.1.1 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.

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.

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.

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.

2.2 FairMQ

The transport layer used for the O^2 Balancer is FairMQ. This is a transport layer from the larger framework FairRoot 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.

2.2.1 Splitting off FairMQ

During this report, the FairMQ repository had to be split off from FairRoot still. In the first stages of the prototype an emergency version was used made by Heiko van der Heijden. Meanwhile a pull request [1] was done on the FairRoot github which resulted in the official FairMQ repository.

2.3 Zookeeper

Zookeeper is a program made by Apache 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.

2.4 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 these EPNs anymore.

[1] https://github.com/FairRootGroup/FairRoot/issues/736

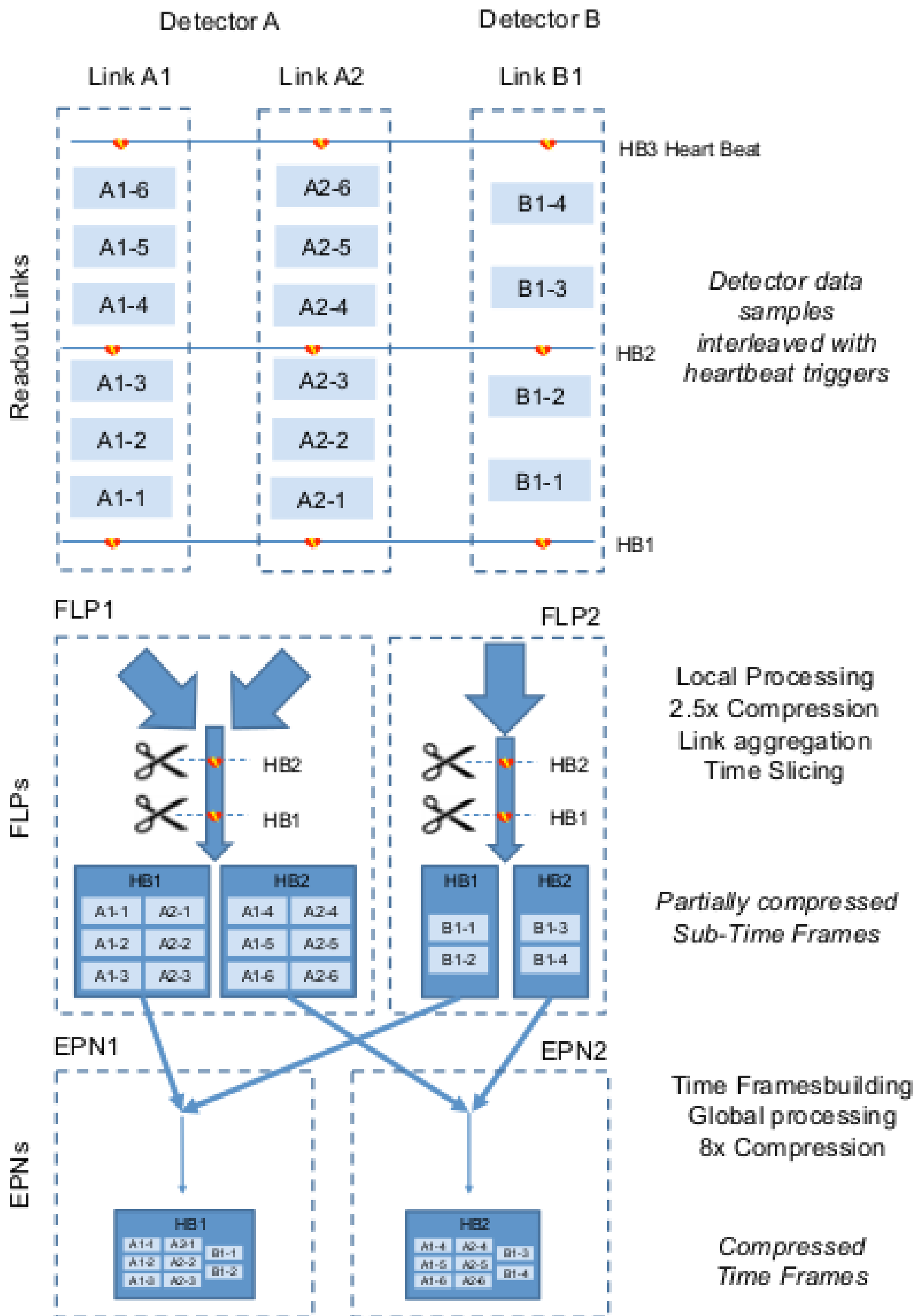

Figure 2.1: Aggregation of data ("Technical Design Report for the Upgrade of the Online Offline Computing System", 2015, p. 34)

2.5 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.

2.6 Blacklist Algorithm

The algorithm used in the previous experiment is a Blacklist Algorithm. This algorithm constantly keeps a list of online 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 2.2

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

2.7 Raspberry Pi

Raspberry Pi is a low cost small computer used for prototyping projects. These projects can reach 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 higher Ethernet speeds on the board itself.

Chapter 3

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.

3.1 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 3.1.

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

Table 3.1: 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 3.2.

Processor	Cortex-A53 1.4GHZ
RAM	1GB PLDDR2 SDRAM
Network	300MbE
Operating System	Raspbian Stretch (Debian 9)

Table 3.2: 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 3.1 shows a network overview:

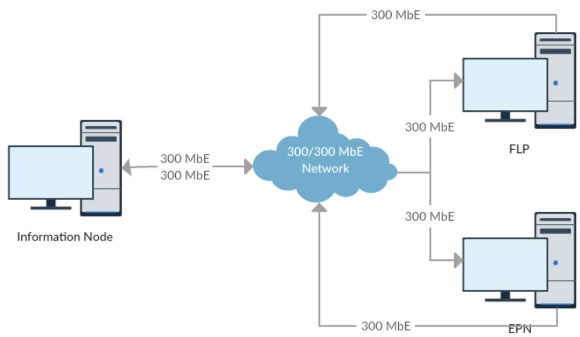

Figure 3.1: Network setup

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 3.3.

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

Table 3.3: Ethernet switch speeds

3.1.1 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.

Figure 3.2: 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.

3.2 Software

The software used is found at https://github.com/SoftwareForScience/O2-Balancer. It consists of 3 executable programs to represent an FLP, EPN and Information Node. These programs are send to the devices to represent their respective units. Apart from that it uses a slightly modified version of FairRoot [1]. Since only the FairMQ [2] 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 as followed.

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 3.4: Dependencies software

[1] https://github.com/FairRootGroup/FairRoot
[2] https://github.com/FairRootGroup/FairMQ

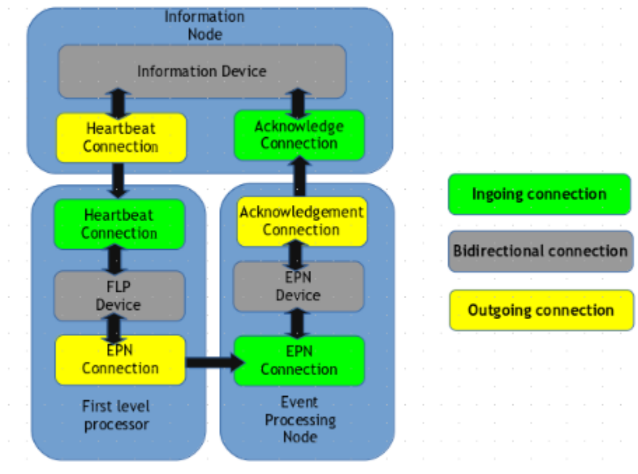

Figure 3.3: Block diagram of the cluster connections (van der Heijden, 2018, p. 22)

3.3 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 3.5.

Tool	Version
Raspbian	Centos 7
ROOT	6.13.02
Python	2.7.13
Ansible	2.2.1

Table 3.5: Dependencies for the analysis scripts

3.4 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 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.

3.4.1 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 recieves 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 can be found in figure 3.4.

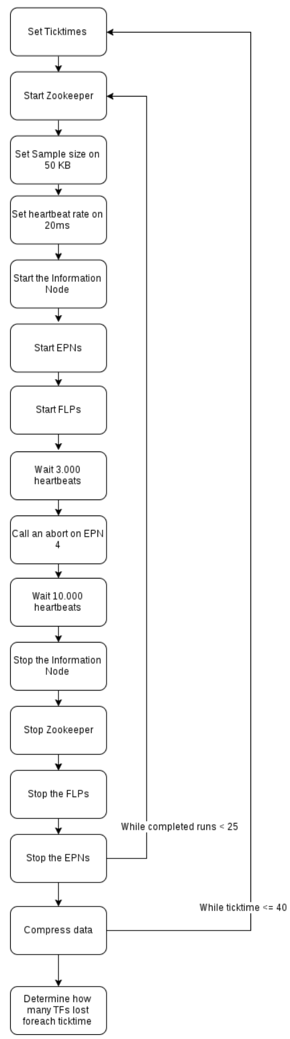

Figure 3.4: Flow char experiment one

3.4.2 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 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 3.5.

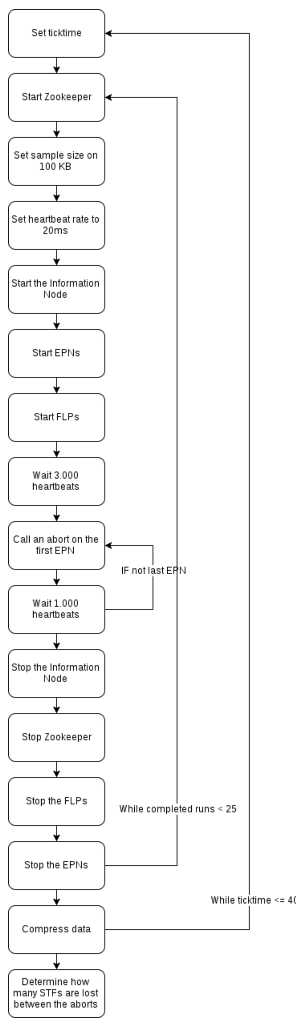

Figure 3.5: Flow chart experiment two

3.4.3 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 a average size of 100 kilobyte . 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 3.6.

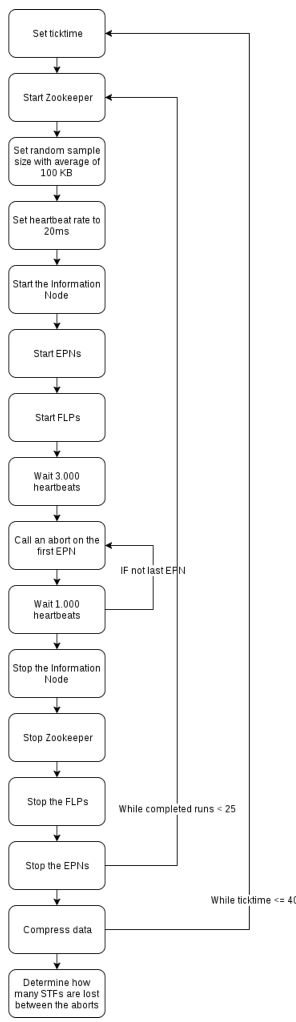

Figure 3.6: Flow chart experiment 3

3.4.4 Experiment four

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

The same heartbeat rate is used from experiment one. For this experiment all but one EPN will be disabled at heartbeat 3.000. After that the system will run for another 10.000 heartbeats to get enough of a buffer to read from. A flowchart can be found in figure 3.7.

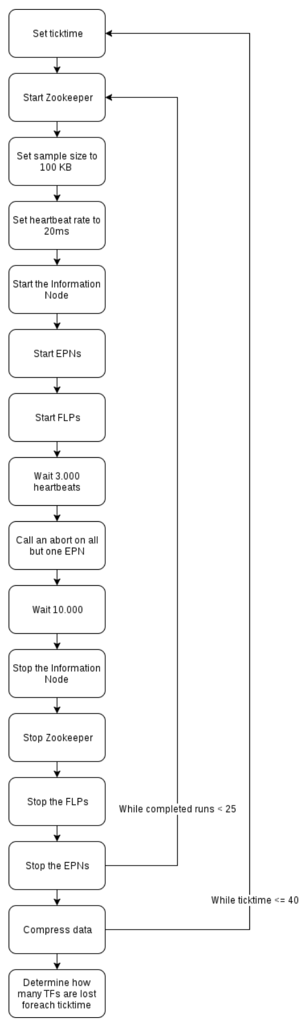

Figure 3.7: Flow chart experiment 4

Chapter 4

Results

4.1 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 4.1. It shows that the new setup does not work all to well with lower ticktimes. At 20 the lost TFs become fewer and from there is also keeps it's standard deviation leveled. It also shows a difference between uneven ticktimes and even ticktimes, preferring even ticktimes.

Ticktime	Mean TF loss	Standard Deviation
5	3.915	11.01
10	3.04	13.21
15	5.267	12.35
20	1.123	12.78
25	2.112	12.48
30	1.304	12.17
35	3.114	8.638
40	1.434	11.4

Table 4.1: 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 4.2, we can see that that the lost TFs are lower with the new setup, but the standard deviation is quite higher. (Factor 40)

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	4.996
40	3.76	0.4271

Table 4.2: Results of the TFs lost with 1 fail over using a cluster of Intel Xeons (van der Heijden, 2018, p. 36)

4.2 Ex.2 2 FLPs 12 EPNs

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

Table 4.3 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 compliant with the previous experiment. These results are shown in table 4.4. A histogram of ticktime 5 shown at 4.1 shows that the standard deviation is 0.9568.

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	2	2	2	2	2	3	3	3	4	4	5
10	2	3	3	3	3	4	4	4	4	5	7
15	3	3	3	3	4	4	4	5	6	7	9
20	3	4	4	4	4	5	5	6	7	9	11
25	4	4	4	4	5	5	6	7	9	10	13

Table 4.3: Cumulative lost TFs by ticktime/EPN ratio with a flat sample size for the Blacklist algorithm

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

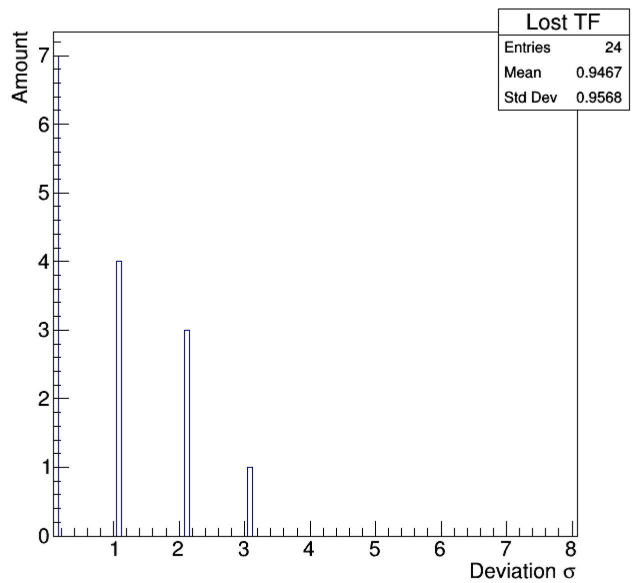

Figure 4.1: Histogram of the TF lost for ticktime 5

4.3 Ex.3 2 FLPs 12 EPNs

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

Compliant with the previous results, the TF loss to ticktime ratio rises in a linear line, with no absurdities along the way. The amounts are also negligibly different to the previous results. Results can be seen in table 4.5 and table 4.6. A histogram for ticktime 5 can be seen at 4.2. It shows a standard deviation of 1.569 for all the TFs lost in the entire run.

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											
10	2	2	3	3	3	3	3	3	4	5	7
15	3	3	3	3	4	4	4	5	6	7	9
20	3	3	4	4	4	4	5	6	6	8	12
25	4	4	4	4	4	5	6	7	8	10	13

Table 4.5: Cumulative lost TFs by ticktime/EPN ratio with a random sample size for the Blacklist algorithm

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

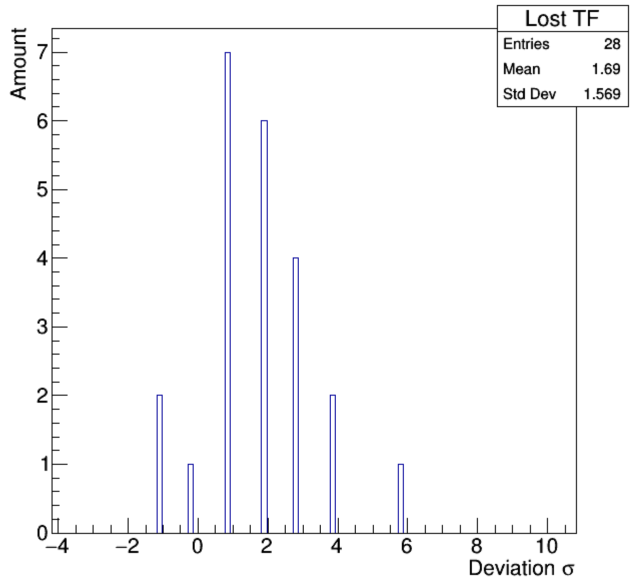

Figure 4.2: Histogram of the TF loss for ticktime 5

4.4 Analysis

4.4.1 Comparison to the previous experiment

Looking at the data from experiment two and three, the difference is almost negligible. For the first EPN that crashes there is a 1-2 difference of TFs lost, and for the last EPN there is 2-3 difference. The difference is mainly from the first experiment. It seems that the new cluster is a bit less functional on uneven ticktimes.

Chapter 5

Conclusion

Chapter 6

Bibliography

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

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

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

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

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/

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

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

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

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

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

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 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 failover	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 failover	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 failover 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