UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Science and Technology
Department of Computer Science

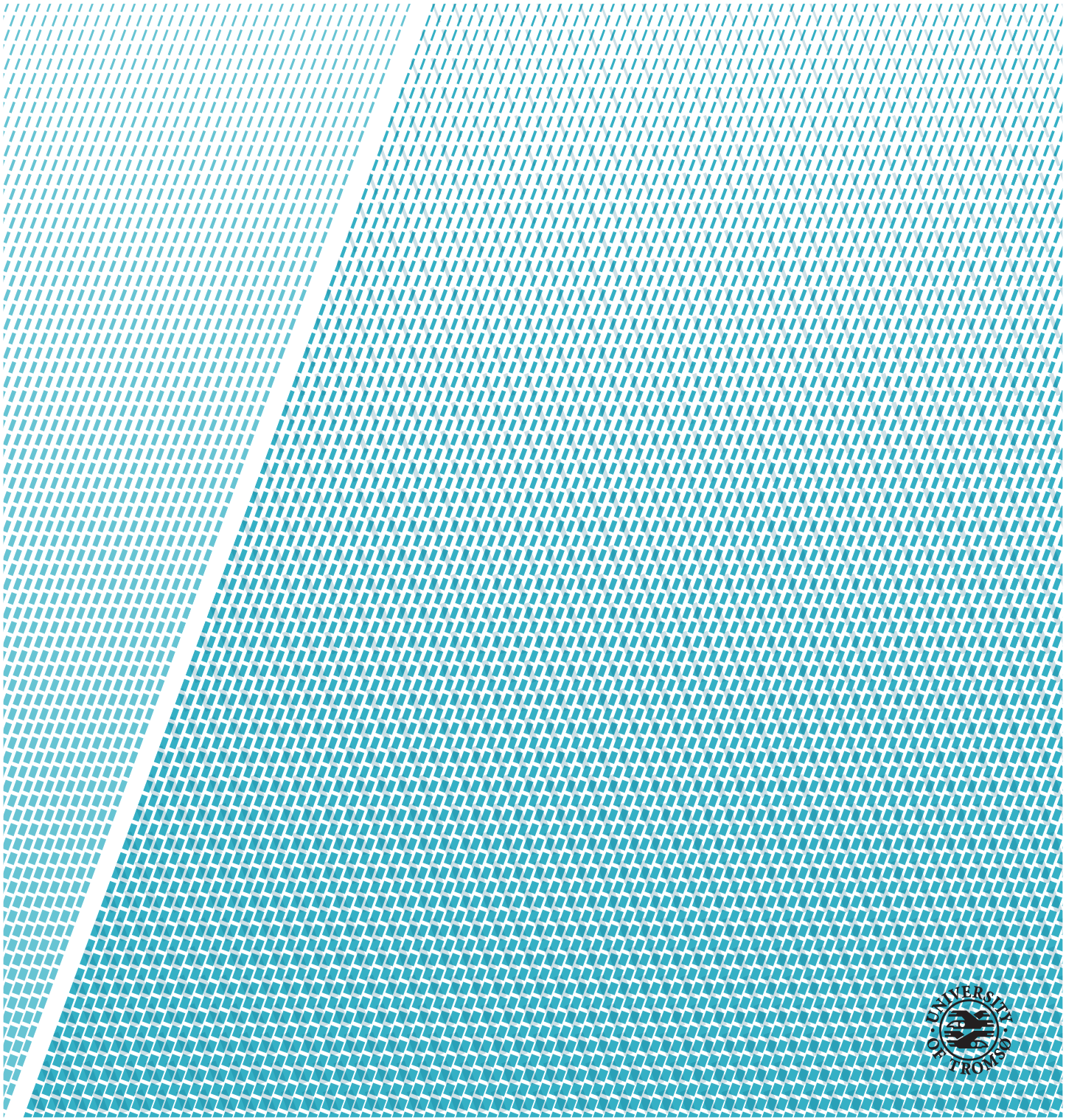
Capturing High Rate Satellite Data

An Analysis of Lossless, Persistent Reception, Local Storage and Transmission of High Rate Satellite Data From Time Window Based Datastream

—

Ruben Mæland

INF-3981 Master's Thesis in Computer Science, June 2016



Dear family,

this is for you.

Abstract

Satellites generate data through their instruments as they hover in orbit. Satellite data is widely used in weather forecasts, environmental science, for military purposes, earth observation and more. The world depend on satellite data, for many purposes.

As satellite technology moves forward, the amount of data generated per orbit increases. Satellites are not equipped with unlimited storage capacity, meaning that the generated data must be transmitted to a ground station at some point in orbit. Satellites transmit data through the use of radio waves, and have the later years used X-, S-, L-band, among others. An increase in generated data, require an increase in the transfer rates between the satellite and ground station. Therefore, the satellite industry will, in near future, build satellites using KA-band using a higher frequency area providing larger bandwidth. The increase in generated data forces the increase in data transmission rate, which require high performance ground stations that can capture incoming high rate data.

A ground station's purpose is to capture the received satellite data, without data loss. Until today several capturing systems provide rates between 1 and 3Gbps, depending on the band used, modulation, whether near real-time processing is offered or not and whether data is actually captured and stored within the same chassis[60, 4, 73, 36]. With KA-band, the rates can exceed 10Gbps, and it is the industry mission to provide systems that can capture such data rates.

This thesis evaluate the next generation technology used to capture data, determining the bottleneck in the setup used today through the analysis tool developed, and further resolving them. The changes suggested in this thesis are tested and evaluated, and show that the next generation servers can capture 10Gbps with the hardware and software available, which is higher than any other related system found.

Acknowledgements

First of all I would like to express my gratitude to my supervisor, Professor Randi Karlsen, for guiding me through this project. Your constructive feedback and support through this project is invaluable. Thank you very much.

I would like to thank my co-supervisors from Kongsberg Spacetec, Hans Berglund and Hårek Gamst, for spending your valuable time guiding me through this project. Your help, advice, visions and knowledge is really invaluable. I really appreciate getting to know you and to work as close as we have through this semester.

I want to express my gratitude to Kenneth K. Johnsen for helping me out with the test data generator and providing me with software to easily configure the test environment. Hans Sandsdalen, I really admire your knowledge of computer networking, security and sense of humor. Thank you for all your help and support, both of you.

Next I would like to express my gratitude to Kongsberg Spacetec for giving me the opportunity to work with one of your profiled systems. Providing me with access to one of your systems, and handing me the opportunity to learn the industry is much appreciated.

To all my fellow students, thank you for 5 fantastic years through the studies. I would never get through this without you all. Specially, I want to express my gratitude to Jarl Fagerli, Morten Grønnesby and Eirik “EM” Mortensen for all the cooperation through the years of study. I am honored to call you my friends.

Finally, I would like to thank my family for always being there for me.

Contents

List of Figures

List of Tables

List of Abbreviations

ALOS Advanced Land Observing Satellite

APSK amplitude and phase shift keying

BPSK binary phase shift keying

CCSDS The Consultative Comittee for Space Data Systems

CPU Central Processing Unit

CSV comma separated values

DMA Direct Memory Access

DMAc Direct Memory Access Controller

FPGA Field-Programmable Gate Array

FTL flash translation layer

GB GigaByte

Gbps Gigabit Per Second

HDD Hard Disk Drive

HDFS Hadoop Distributed File System

HRD High Rate Data

HRDFEP high rate demodulator and front end processor

IDE Integrated Drive Electronics

IOPS I/O Operations Per Second

JAXA Japan Aerospace Exploration Agency

KSPT Kongsberg Spacetec A/S

MB MegaByte

MBps Megabyte Per Second

Mbps Megabit Per Second

MEOS Capture Multimission Earth Observation Systems Capture

NAND Not-And

NASA National Aeronautics and Space Administration

NVMe Non-Volatile Memory Express

OS operating System

P-ATA parallell ATA

PCI Peripheral Component Interconnect

PCIe Peripheral Component Interconnect Express

PV Pipe Viewer

QAM Quadrature amplitude modulation

QPSK Quadrature phase shift keying

RAID Redundant Array of Inexpensive Disks

RAM Random Access Memory

RPM Revolutions Per Minute

SAR Synthetic Aperture Radar

SAS Serial Attached SCSI

SATA Serial Advanced Technology Attachments

SCCC serial concatenated convolutional coding

SCSI Small Computer Systems Interconnect

SRAM Static Random Access Memory

SSD Solid-State Drive

TCP Transmission Control Protocol

Tmux terminal multiplexer

UDP User Datagram Protocol

List of definitions

List of Listings

1 Introduction

In this chapter the motivation, goal, approach and a formulation of the thesis is presented. The layout for this document is also provided.

1.1 Motivation

The satellite industry has seen increasing transfer rates used by satellites when they transmit data to a ground station on earth. The first ever satellite was Sputnik-1 launched in 1957 according to National Aeronautics and Space Administration (NASA)[48]. Over the years, thousands, or even millions, of satellites has been launched since 1957 with different missions. The one thing all satellites have in common, is the limited storage capacity forcing them to transmit data to a ground station located somewhere on earth, in order to archive valuable data generated by sensors located on the satellites.

Satellites in orbit are visible for ground stations in a very limited time window, as shown in figure 1.1, which introduces some requirements with respect to receiving and storing the data transmitted at a particular ground station. The satellite is typically visible for an antenna when it is in a specific part of its orbit. This time window vary from satellite to satellite, usually from 1 to 15 minutes long. A ground station typically consists of an parabolic antenna which is connected to a demodulator that demodulate the received signal. This is typically a analogue to digital converter. The demodulated signal is sent

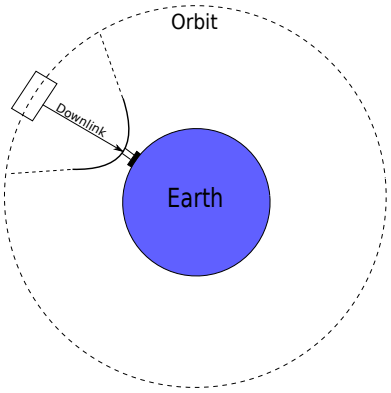


Figure 1.1: The figure illustrate satellite in orbit transmitting data to ground station. Note that when out of antenna bounds, the satellite is not reachable for the ground station.

to a server implementing functionality that enable the capturing of satellite data and possibly some level of processing. The processing of captured data converts the captured data into the original file format generated by the specific instrument. The processing may also be the generation of metadata[1]. If the system perform both generation of metadata and converts the data to the original file format, it is referred to as "level-o" processing. The parabolic antenna size vary depending on the frequencies used by the satellite for data transmission.

Satellites transmit data to a ground station through radio waves using defined protocols; i.e., The Consultative Comittee for Space Data Systems (CCSDS) serial concatenated convolutional coding (SCCC)[67], also known as turbo coding, and DVB-S2[46, 18]. It is also possible to transmit data without any protocols or coding, where raw data is transmitted. The protocols purpose is to enable high rate transmission of data, as well as ensuring the best tradeoff between high rates, robustness and reliability by generating checksums and mechanisms enabling bit error correction. The physical transmission is done through radio waves, which require modulation (at satellite) and demodulation (at ground station). Examples of such techniques is BPSK[63], Quadrature phase shift keying (QPSK)[22, 31] and Quadrature amplitude modulation (QAM)[50, 23], among others. It is the bandwidth, modulation and protocol that together decides the rate of data transmitted.

Some satellites today are equipped with instruments compatible with transfer rates of 600MBPS. Through 64-amplitude and phase shift keying (APSK) and turbo coding (or CCSDS SCCC), the antenna receives 6 bits per modulated

1. Metadata is data about data.

symbol[2]. The main increase in transfer speed has evolved through advanced algorithms such as CCSDS SCCC and modulation algorithms, since the X-band bandwidth is more or less constant. This increasing transmission rate forces the need for powerful machines with high performance hardware to receive and make sure the data is persistently stored and above all, not lost. Data is considered safe when it has been written to persistent storage.

The industry and agencies within the satellite business, want to distribute the newly received data as fast as possible to their customers. This complicates the process of receiving data without data loss. And to complicate the process even more, some customers demand data in specific formats which require some level of processing before it is distributed.

This thesis is done in cooperation with Kongsberg Spacetec A/S, using one of their systems, MEOS capture v4.5[36], for receiving satellite data. Current status is that through X-band[57, 5] and QPSK modulation, $1 - 3$Gbps transfer rate is considered as the maximum rate for the industry in general, to guarantee lossless receiving of data during a pass. The satellite industry wants to assess how to reach rates up to 10Gbps, by testing and evaluating candidate technologies for future systems as the satellite agencies want to use KA-band[56] providing larger bandwidth than X-band. This is the main motivation for the thesis.

When data is demodulated and decoded, it is sent to a server which mission is to make sure the data is persistently stored. As the transfer rate seems to increase by an order of magnitude in near future, the receiving system has to increase performance equally, or more. When the incoming data rate gets above 1 Gbit/s, the systems in general does not dump the incoming data to disk fast enough, which may cause buffer overflow in memory and data loss. Lost data cannot be restored during this process, and customers depend on the functionality of this system to capture satellite data without any loss.

1.2 Goal

The main goal for this thesis is to create a new baseline for the next generation HP ProLiant servers, determining the rate at which it can capture satellite data, lossless. In order to achieve this, the first part is to develop an analysis tool that monitors the load on the server hardware components in order to locate bottlenecks within the server. The second step, is to use the results produced to increase the overall performance for the server with respect to

2. A symbol is a radio wave carrying data, which is received by the satellite antenna.

the use case it can face when used in a system such as MEOS capture. The term performance means the capacity at which the system can receive and persistently store incoming data at high rates, without any data loss. The aim for this study is to determine whether the next generation HP ProLiant servers can be used in future versions of MEOS capture, where the transmission rates reach 10Gbps and above.

To increase the performance for the next generation HP ProLiant servers this thesis will use the information generated by an analysis, in order to make qualified decisions to solve the located bottlenecks. Based upon the analysis results the thesis will re-design and evaluate the changes made for the server with respect to performance. No analysis of this particular server, was found for this specific set of hardware. The hardware was used in one study [39] as the controller host connected to another HP machine, and it is the only found study that use the same server[21].

1.3 Approach

To achieve the goal mentioned above, thorough analysis of the server throughput will be done in order to locate existing bottlenecks. The analysis will include the creation of an analysis setup, based on existing analysis tools, that monitors the throughput throughout the system pipeline, and provide live information about the momentary performance by different parts of the system. This analysis tool will also be used to determine whether changes to the system gives improved performance.

When the analysis of the system is available, the thesis will present qualified decisions to where the system has potential for improvement. This includes presentation of a new design for the system, new implementation and new testing of the system.

The analysis tool developed provides runtime statistics with respect to the load that computer components are exposed to. To do so, the analysis tool uses already existing tools such as Htop and Dstat, and by monitoring the kernel log. The output from Dstat is stored in a file for further analysis purposes.

To locate possible bottlenecks, parts of the total pipeline will be tested in isolation in order to determine the maximum throughput between critical components. By testing the maximum achieved throughput between the data receiving board and host memory (or RAM), and between RAM and the ingest disks, we will be able to tell which part of the system pipeline offers the lowest rate. From there one can determine where the system has potential for

improvements.

Kongsberg Spacetec A/S (KSPT) has implemented a data generator that can generate and send out data at the required rate to simulate a satellite pass. This data generator is a Peripheral Component Interconnect Express (PCIe) board that is implemented to simulate a high rate stream of data. This test generator will be used to simulate satellite passes, in which the server will attempt to capture without data loss.

Further experimentation will determine maximum throughput when performing near real-time processing of data, to examine the system robustness. The near real-time processing conducted in this thesis will not generate metadata, or convert any data into any fileformats. The processing will only be simulated, meaning that the captured data is read from the ingest disk, and written to another set of disks storing processed data, in order not to have any further impact on the data capture. No further processing of the data will be performed.

1.4 Contributions

This thesis provide the following contributions:

- We have successfully built an setup for analysis tools that provides valuable information and statistics about the computer components and processes as a satellite pass occurs. Through the analysis tools we were able to determine where in the pipeline bottlenecks occurred. Based on the data provided by the analysis tool, we were able to resolve the bottlenecks.

- We evaluate a new baseline for the 9th generation HP ProLiant servers in terms of supported rates with the configurations used today, and the improvements made reached the goal of 10Gbps which is higher than any related system found.

1.5 Outline

The remainder of this thesis is structured as follows.
Chapter 2 presents an overview of technology used both in this thesis and currently used in the existing system.

Chapter 3 provides an introduction to the configurations used in the data

receiving front end server, and explains how the server receives and process data.

Chapter 4 gives an overview of the analysis tool setup developed. The architecture, design, and implementation are all presented in this chapter.

Chapter 5 presents the experiments performed on the test environment, using the configurations presented in chapter 3. The system evaluation is presented, in order to make qualified decisions to where the system can be improved.

Chapter 6 presents, tests and evaluates the changes performed.

Chapter 7 concludes the thesis and outlines future work.

2

Technical Background

This chapter introduces several topics related to satellites and satellite data. The chapter is divided into five main parts, each presented below.

Satellites Through section 2.1 some theory on satellites and how satellites generate their data is presented. The chapter also covers some theory on satellites in orbit.

Data transmission The theory behind data transmission between satellites and ground stations is presented in section 2.2.

Data storage What happens when data is received by the antenna? The theory on how data travel from the satellite antenna to a HDD is presented in section 2.3 and 2.4.

Challenges and tradeoffs Section 2.5 present some theory on fault tolerance and its impact on performance. Bottlenecks are also defined in this section.

Related work Some related work are presented in section 2.6.

2.1 Satellites

Since the launch of Sputnik-1, several satellites have been launched in orbit around the Earth. Examples of satellite missions can be routing of telephone traffic across continents, global positioning system (GPS), military and aerospace purposes and sampling of weather data. This divides satellites into two main categories; Communication satellites and earth observation satellites, where earth observation satellites is the one most relevant for this thesis. However, satellites consist of a chassis in which instruments and components are attached to. The satellite has an infrastructure in terms of connecting the instruments to a temporarily storage system. The instruments are typically photographic sensors taking pictures of the earth. The instruments may also be equipment for radio transmission and reception, in order for the satellite to dump data to a ground station, and receive updated firmware if needed.

One of the big issues with satellites is that once it is launched you cannot do any physical changes to it, at least in terms of changing the construction. Satellites have to use the equipment it is launched with. Due to this fact planning is done over several years to determine in detail how the satellite must be designed, built and function.

All vehicles launched to space are exposed to extreme conditions. Shocks, vibrations and temperature change during launch and radiation when in space. Especially electronic devices are prone to radiation, and it is proven that electronics are malfunctioning when exposed for high altitude radiation[55]. This means that all components on board have been thoroughly tested to prove that it will work in the environment in which it will operate. When a component pass all tests, it is radiation hardened and space qualified. European Space Components Coordination (ESCC) have published a list of qualified parts and manufacturers on their websites[19, 20]. This contributes to the high cost within satellite industries. In the work by P. Dieleman et al.[16] they testet an Heterodyne Instrument for the Far-Infrared (HIFI) for use on satellites, and explains how they tested their instrument for space qualification.

Once a satellite is launched it has to deploy all its modules containing instruments. The orbit is determined due to the mission for the particular satellite. Some satellite missions can be to gather data about polar areas, equator and or other interesting areas. This requires the satellites orbit to cross over the area of interest at some point. Orbital mechanics describe satellite orbits as elliptic curves[11]. The reason that satellites can mange to stay in orbit, is a combination of the earth's gravitation and the satellites velocity. One can think of this as a ball in a rope when you give the ball speed in some direction and hold the rope at the other end. When the ball is given speed, it rotates around your hand (representing the earth). If the ball lose speed it lose altitude and

finally fall to the ground. Reversible, if the ball increase velocity, it wants to get further away from your hand and hence gain altitude. The principle also applies for satellites; The higher the velocity gained at launch, the higher the altitude. It is important that the velocity does not exceed escape velocity which is the required velocity for a vehicle to break out of earths field of gravity[13].

The satellite mission determine the altitude the satellite will orbit. Polar satellites orbit at a much lower altitude than geostationary satellites. One issue with polar satellites is that they are affected by the earths gravity. To prevent altitude loss, polar orbit satellites are equipped with thrust engines to maintain the orbit. On-board fuel is a limited resource which means that the satellite will be forced out of orbit at some point by gravity and land on earth. Geostationary satellites have an orbital speed equal to the earth rotation speed, which means that it is kind of hovering above the earth at the same point at all times. Because of the great altitude the earth gravity do not affect geostationary satellites as much as polar satellites, enabling much less need for orbital correction.

The key features with satellites is that they can be used to produce data that one can not produce from on earth locations because one achieves a much better perspective from above. Data produced on satellites are produced by several sensors. Said sensors can measure temperatures, capture pictures through cameras of some pre defined area among others. The data generated is highly related to the sensor technology available and what sensors the satellite is equipped with.

As the satellite sample data in orbit, the satellite has to transmit data to a ground station to free memory for new samples. The amount of data a satellite possibly can produce during an orbit is determined by the sensors it is equipped with. Important aspects are the amount of time the satellite is within reach of the ground station antenna, the downlink speed, and how data is modulated. Some satellites may dump data to several ground stations during one orbit. The protocols implemented does not support any form of retransmission of lost data, such as Transmission Control Protocol (TCP)[1]. Satellites transmit data through radio waves which protocol is very similar to the User Datagram Protocol (UDP).

A lot of work has been published on how to design a satellite. Examples of published work on satellites are the successor of Landsat 7[30] for earth observations and METSAT[34] for weather and climate purposes. The Landsat project have also published a survey of current status of all Landsat satellites[41].

1. http://www.diffen.com/difference/TCP_vs_UDP

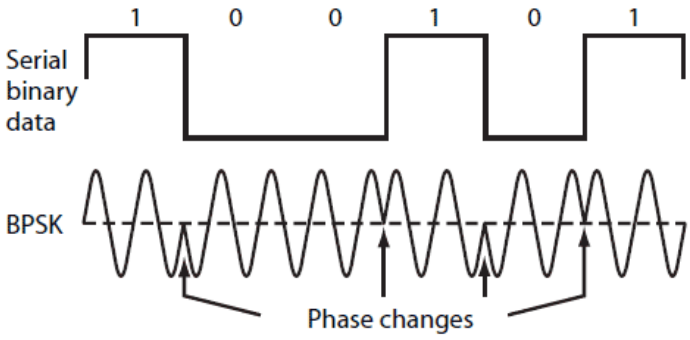
2.2 Satellite Data

To understand how satellite data are transmitted from satellite to a ground station, one must understand several topics. Each of them are presented throughout this section.

2.2.1 Data Modulation

Satellites generate data as they travel in orbit. A particular satellite is only visible for an antenna in a very small part of the orbit. The satellite use radio waves to communicate with the ground station, which requires the satellite to both have a modulator and demodulator. The downlink is typically much more powerful and faster than the uplink.

However, the satellite use a modulator to modulate the data, which means that the data is translated to radio waves. Exactly how this is done depends on the algorithm used. The satellite and ground station must agree to how this is done on beforehand, and is decided years before the satellite is launched. As mentioned in chapter 1.1, there are several modulation techniques available today. Some are more efficient and advanced than others. BPSK is simple and very easy to understand, which represent one bit per symbol. This means that a BPSK-symbol can look like described in figure 2.1. The figure show that BPSK use phase shift at 180 degrees, where the binary bit 1 can be identified by what looks like an `"M"`, and 0 as `"W"`[40].



2. In binary phase shift keying, note how a binary 0 is 0° while a binary 1 is 180°. The phase changes when the binary state switches so the signal is coherent.

Figure 2.1: An example of how a bit string can be modulated using BPSK.[40]

Where BPSK only recognize two phases, more advanced modulation techniques

use 4, 8, 16, 32 and even 64 phases. These modulation techniques allow $x - bits$ per modulated symbol depending on the technique used, which again increase the data rate.

This technology is also the very foundation of what we know as the internet today. Satellites have enabled world-wide communication through both internet and mobile cellphones[32, 10, 69].

2.2.2 High Rate Data

The data rate is now defined by the following formula:

$$Datarate = (SymbolRate * BitsPerSymbol) - p \tag{2.1}$$

where data rate is the effective rate when the data is demodulated, symbol rate is the frequency at which symbols arrive at the antenna, bits per symbol refers to the modulation technique used, and p refers overhead generated by the used protocol, if any. Protocols are said to steal between 1 and 10% of the bandwidth due to checksums and methods to enable bit error correction[67], depending on the protocol and modulation used. The newer satellites, yet to be launched, plan to use $2 \times 620 Mbit/s$ downlinks in combination with SCCC turbo coding and 64-APSK modulation. The transfer rate is determined by the formula 2.1.

```
Incoming data rate = ((620 Mbit/s * 2) * 6 bits per symbol) - 10%
Incoming data rate = 7 Gbit / s
```

This example is however not very representative, because very few satellites have implemented 64-APSK. But it shows that the limitations does not lie within future satellites. A more natural setup is as follows.

Two downlinks of 310 Mbit/s each, with QPSK modulation and no coding of the data, which generate no overhead in protocol. QPSK deliver 2 bits per symbol[40], which gives the following rate.

```
Incoming data rate = ((310 Mbit/s * 2 downlinks) * 2 bits per symbol) - 0%
Incoming data rate =  1240 Mbit/s
```

As we can see, this configuration achieves just over 1,2 Gbit/s, which is considered “state of the art".

2.3 Data Storage

Data storage have experienced vast improvements in storage capacity per square inch and performance. According to PCWorld[58], IBM launched a 5MegaByte (MB) drive in 1956, as big as two refrigerators, using 50 24-inch platters. In 2006 Cornice and Seagate launched a 1-inch HDD with 12GB.[58]. Throughout this section some background on HDDs and storing techniques are presented.

2.3.1 From Punched Cards to Flash Based Storage

Computer storage has evolved from punched cards, punched tapes, to floppy disks and to the most used storage today; HDDs. The evolution has compressed the bytes per square inch by orders of magnitude, from bytes per inch to MB per inch.

HDDs typically consist of several platters coated with magnetic material stacked on top of each other, with a mechanical arm that is able to swipe over the disk to access the different cylinders on each disk. When the arm moves from one sector to another is called seeking. The seek time vary from disk to disk depending on the diameter of the disk platters. The disks are often connected to the computer through Small Computer Systems Interconnect (SCSI) drives using the Serial Advanced Technology Attachments (SATA) bus, which is built on top of Integrated Drive Electronics (IDE). The downside with HDDs is that they are slow compared to RAM, even with the high end HDDs, typically with 10,000 Revolutions Per Minute (RPM) and 15,000 RPM. More information on HDDs general architecture can be found in [61].

As HDDs have proven themselves to be slow, a new generation of storage have entered the market. Flash based storage, or SSDs, have become more popular for high performance systems because of their high performance. The last few years, SSDs also have become more affordable. SSDs consist of logical Not-And (NAND) gates and can be connected to SATA via SCSI drivers. An even faster SSD is Non-Volatile Memory Express (NVMe), which can be connected to the PCIe bus. A previous study has proven that NVMe can achieve 120,000 I/O Operations Per Second (IOPS) with random writes and 40,000 IOPS with random reads of 4 KB data[17].

The corresponding mechanism for HDDs moving arm, is the flash translation layer (FTL)[44]. Because SSDs do not have any moving parts and are purely electronic devices, they are less power consuming, have lower access latency and are more resistant to shocks[42]. However, the expected lifetime is lower than HDDs. The reason SSDs are expected not to live as long as HDDs is

because they use something called “tunnel oxide layer” which get worn out after some number of accesses. This is why load balancing for SSDs are important. Algorithms have been developed to spread load over all logical circuits, to increase expected lifetime.

The load balancing algorithms developed for SSDs makes sure that one evenly distributes written data across all sectors. Reads are a bit different, since they must read data from the block in which the data exist. The distribution of written data takes into consideration the total wear of all available blocks and chose the least used one.

2.3.2 Storing Satellite Data

Satellite data is expensive, encouraging involved parties to engage long term storage in order for the data to be used for science and historical purposes. As satellites can generate several GBs of data per orbit, the stored dataset increases per pass. When dealing with datasets constantly (or at least per pass) increasing size, one will at some point deal with the term “Big Data”. According to Dictionary.com, big data is defined as follows;

The Definition 1. data sets, typically consisting of billions or trillions of records, that are so vast and complex that they require new and powerful computational resources to process.[14]

The definition address several issues related to big data, where one should consider highly effective compression algorithms to store the data effectively. The desire is to store as much data as possible, using the least storage capacity possible.

2.4 Data Transmission and Storage Components

The section provides some theory on how data are transferred from host memory to the storage device, typically HDDs or SSDs.

2.4.1 Computer Bus

A computer bus enables peripheral devices to be connected to a host, and establish data transfer between the host and the device. Several buses has been developed over the years, but this section will only consider the buses used in this thesis.

PCI Express All devices within a computer are connected to a bus, that allow transmission of data between devices. A bus consist of hardware (i.e., wire), software and some communictaion protocols. There exist several types of computer buses, where SATA is widely used in personal computers. PCIe is mostly used in systems that require high performance which makes it more expensive than for example SATA[68, 37].

PCIe is the result of work based upon Peripheral Component Interconnect (PCI), and have been released in several versions. The first generation PCIe was released with 2.5 Gbit/s raw bandwidth per lane, and its successor provide 5 Gbit/s[53] . The latest version, 3.0, released in 2010, which provide transfer speeds of 7.8 Gbit/s[54]. Another vast improvement is that version 3.0 support up to 32 lanes, each lane with transfer speed 7.8 Gbit/s. PCIe is based upon lanes that are bi-directional, meaning that one link is can send data one way, where the other can send the opposite way. One lane consist of two links, which is why the lane is bi-directional. The PCIe initially negotiates with the device on how many lanes to use. One link must support at least one lane, enabling a link to consist of N lanes in each direction. PCIe 3.0 can support up to 32 links[54].

Serial Attached SCSI (SAS) Serial Attached SCSI (SAS) is the successor of parallell ATA (P-ATA) which was limited to a maximum of 16 devices connected to the same bus, where one must be a host bus adapter[64]. P-ATA provide a maximum transfer rate of 320 MB/s. SAS support up to 127 devices connected with a fibre channel, and transmission speed of 600 MB/s. The main difference is that P-ATA use parallel transfers where SAS transfer data in a serial manner using frames[64].

An interesting observation is that while the rest of computer hardware moves more and more towards parallel solutions for better performance, data transfer to HDDs are moving away from parallel design to serial communication.

2.4.2 I/O Software

There exist at least three different schemes when dealing with I/O software. First there is programmed I/O[65] which have the CPU to do all the work. This scheme may be good for some use cases, for example when the CPU do not have anything else to do. When the CPU can spend time doing other work, it is considered a inefficient method. This is also where the second scheme is better off.

Interrupt-driven I/O[65] allow the CPU to schedule the I/O operation, do a

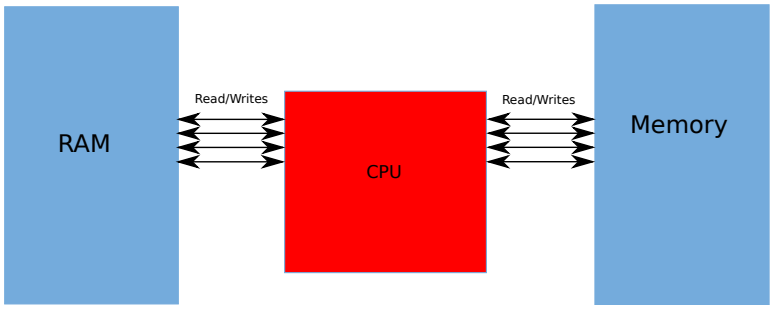


Figure 2.2: Programmed I/O, CPU is in control of a device's I/O within the device's memory space.

context switch and continue doing whatever it was working on. The downside by interrupt-driven I/O, is that the CPU is interrupted for every character written, in order for it to send the next character. When writing large amounts of data, this is a very inefficient design. To solve this, the third option is the one to prefer.

Direct Memory Access (DMA)[65] require a Direct Memory Access Controller (DMAC) which administrates the in- and outgoing I/O to a device, allowing the CPU to do other important work. The reduction of interrupts may be a huge improvement when the CPU actually do have work to do. If the CPU is idle during the I/O process, programmed I/O may perform better because the DMAC is usually much slower than the main CPU[65].

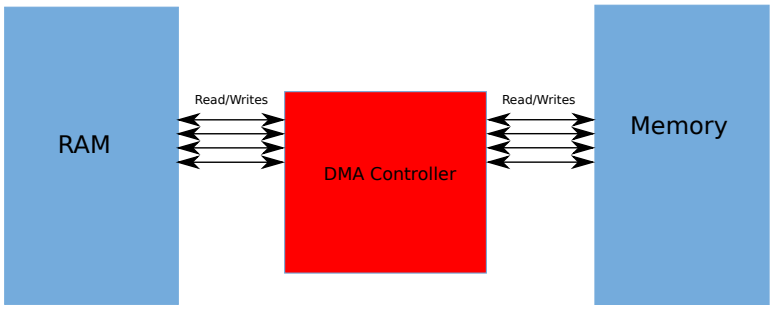


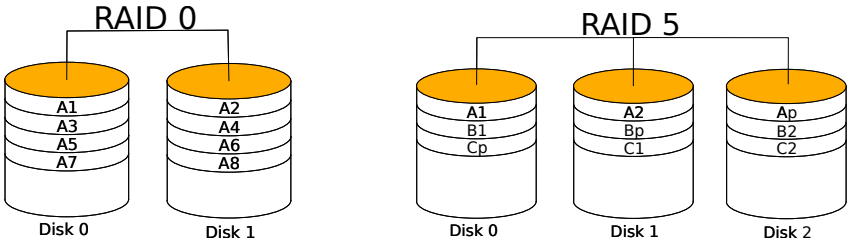
Figure 2.3: The DMA controller provide a device access to its I/O space within memory for data transfer.

2.4.3 Redundant Array of Inexpensive Disks (RAID)

RAID[9, 65], is a system that allow computers to use two HDD or more physical drives as one logical drive. This abstraction has proven itself to be very useful in, for example, computer clustering[29]. The idea with this abstraction is to achieve better performance when writing to or reading from disk, by distribut-

ing data across several physical disks. Another property that RAID can provide is redundancy and hence fault tolerance.

There exist several RAID[62] levels, each with different properties. RAID 0 provides no redundancy, but distributes data evenly across the disks participating within the RAID. This abstraction is purely for performance purposes, providing high throughput. RAID 1, or mirroring, provide fault tolerance by storing the same data on all participating disks. Other RAID levels provide some level of redundancy, using rotationally parity partitions spread across all disks participating (RAID 2-6).



(a) RAID level 0 **(b)** RAID level 5

Figure 2.4: Visualization of RAID-0 and RAID-5. As the picture illustrates RAID-5 use parity partitions and require at least 3 disks, while RAID-0 only distribute data across disks and require at least two disks.

Some RAID-levels allow one to remove one or more disks while the computer is still running, due to their redundancy. This is a very powerful property, because one can replace broken disks on the fly, without disconnecting the computer. The RAID controller makes sure that the new disk is equipped with its partitions of data, and parity partitions included so that it contain the exact same data as the broken disk had.

2.5 Challenges and Tradeoffs

Fault tolerance and performance are two terms that seems to be two poles apart. Very often one must agree upon some compromise when developing high performance systems, requiring some level of fault tolerance.

2.5.1 Fault Tolerance vs. Performance

In computer science the term fault tolerance refers to the amount of unexpected events that may occur before a system crashes. Fault tolerance is often achieved by redundancy, meaning that you have several replicas of the same resource

available. Fault tolerance theory state that a system is k-fault tolerant if and only if there exist $k + 1$ entries of the exact same property[47]. Practically this means that a system can only withstand k faults before it breaks and the service become unavailable, and not recoverable.

Building large complex systems often become very advanced and expensive when the goal is fault tolerance. Windows Azure[8] is a system that was built to be fault tolerant. In their work fault tolerance was achieved by storing 3 copies of the same data on several server racks within a data center, each rack connected to different power supply, and by replicating the data across geographical sites in case of extreme weather conditions.

On one hand one can build robust and fault tolerant systems almost unbreakable, on the other hand one can develop high performing systems. Dealing with redundancy requires resources to accomplish, and fault tolerance are often a compromise in combination with performance.

When dealing with satellite data, one must provide reliable storage with some level of redundancy to support unexpected events such as disk failures. But one must also provide enough performance so one can capture high rate data during a pass. Fault tolerance in such systems are often achieved through some kind of RAID level offering redundancy, where one should use striping when performance has the priority. Data redundancy does however only allow disk failures. There are several other precautions one should consider. Power supplies are often topic when building fault tolerant systems, and it is common practice to have your servers connected to two or more independent power supplies.

Fault tolerance tend to end up with bottlenecks, limiting performance.

2.5.2 Bottlenecks

A bottleneck is defined by Business Dictionary as;

The Definition 2. Department, facility, machine, or resource already working at its full capacity and which, therefore, cannot handle any additional demand placed on it. Also called critical resource, a bottleneck limits the throughput of associated resources.[7]

From the definition one can conclude that a bottleneck is the stage of a pipeline, limiting the total throughput. Almost every computer program can be seen as a pipeline, where some stages has to be the slowest one and hence the bottleneck. This also yield for capturing satellite data. Bottlenecks does however have two

properties associated with them; They are limiting throughput and they may be solved. The hardest part with bottlenecks are to locate them. It requires thorough testing and evaluation of every stage within the pipeline to conclude where the bottleneck appear.

2.5.3 Field Programmable Gate Array (FPGA)

FPGAs is a type of gate array that is programmed post manufacture[6]. PC Mag define FPGA as follows.

The Definition 3. A type of gate array that is programmed in the field rather than in a semiconductor fab.[51]

This enables FPGAs to be programmed on site where they shall be used. According to PC Mag, the majority of FPGAs are Static Random Access Memory (SRAM)-based, which require power to hold its content[52]. FPGAs are often used to enhance performance on some specific part of a process, because one may connect an FPGA to a circuit board, implementing functionality earlier done in software, now on hardware. The advantage is that there is no instruction, which requires the CPU to invoke it, to perform the operations. Dedicated hardware as used in FPGA may enhance performance by order of 100[6].

2.6 Related Work

Due to the very specific test environment for this thesis, not much related work has been found. One can relate some work regarding software and hardware configurations and performance. The most important ones is presented here.

The work done by McLemore et. al.[43], published in 2012, demonstrate successful capture of satellite data transmitted at 277 Megabit Per Second (Mbps) per second. The work used an older version of the exact same system as the system evaluated in this thesis, and it gives a perspective on the development of transfer rates from 2012 until today, where 1 Gbps is considered state of the art. The study enlighten a new perspective within satellites, where the satellite Advanced Land Observing Satellite (ALOS) was able to produce more data per orbit than it was able to transmit to earth. This enabled a cooperation between NASA and Japan Aerospace Exploration Agency (JAXA), which was its first of its kind. NASA offered several ground stations, enabling several passes per orbit and hence allowing the satellite to produce more data per orbit. JAXA provided

NASA with access to valuable Synthetic Aperture Radar (SAR) data.

When dealing with incoming High Rate Data (HRD) from satellites, one must ensure that the data is persistently stored lossless. The ingest disks are exposed for heavy load during a pass, which require some level of disk performance to enable lossless storage. Much recent work report impressing results when using NVMe[28] SSDs. In [33] they used NVMe to create a workload-aware budget compensating scheduler. The reason for using NVMe in that study is to embrace the performance offered by PCIe, instead of using what they call the bottleneck SAS. In [45] they used NVMe SSDs to enhance Hadoop MapReduce performance using Hadoop Distributed File System (HDFS). And in the work by Elsebø[17] they developed an NVMe driver for Vortex[38] that is exposed as a SCSI device. Vortex is an experimental implementation of the omni-kernel architecture. The study also measures disk performance, proving NVMe to be the best performing disks on the market today in terms of reads, writes and IOPS. Other studies have applied disk caching disks[26] to reduce latency and increasing performance, and placing data in the disks free blocks by observing the disk access patterns at runtime[27].

3

MEOS Capture

In this chapter the current state is documented for Kongsberg Spacetec's (KSPT) Multimission Earth Observation System (MEOS) Capture, introducing the architecture and execution flow. The test environment specifications are presented in section 3.2. MEOS capture is a capturing system that receives, demodulates, frame synchronizes and bit error corrects HRD from satellites. The system captures satellite data, lossless, at high incoming rates. The system can be delivered with a satellite antenna, spanning from 3.0 meters dish to 5.0 meter dish as specified in their data sheet[35].

3.1 MEOS Capture HRDFEP v4

The system architecture, hardware and configurations are documented in this section.

3.1.1 Architecture

The architecture is illustrated in figure 3.1, where the main components within MEOS capture and execution flow is presented.

The system consist of following modules:

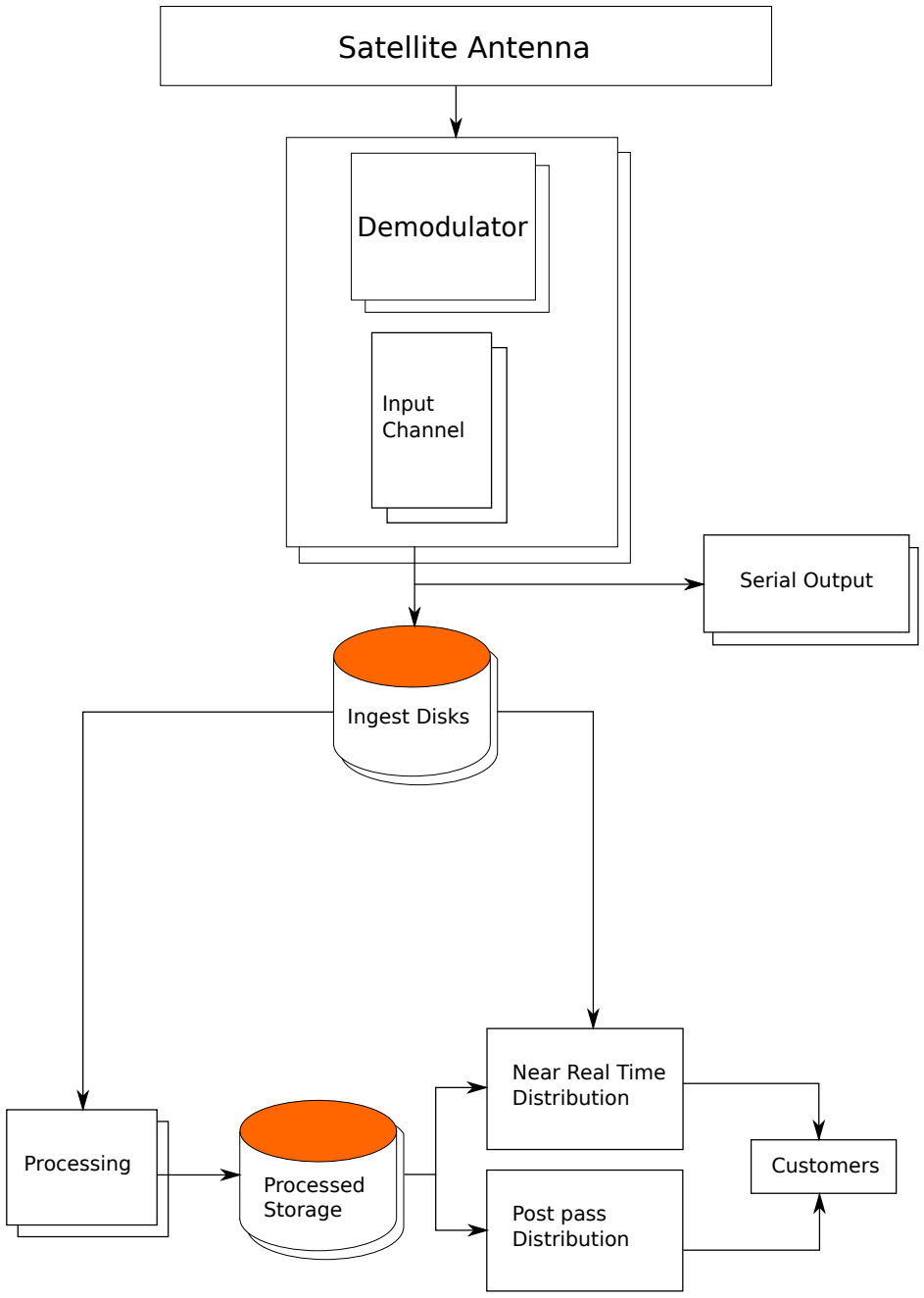


Figure 3.1: The architecture for MEOS capture. Note that data is either sent directly from ingest disks to distribution, or processed before distribution.

- Satellite antenna
- Demodulator
- Input Channel (Frame synchronization and decoding)
- Serial Output
- Ingest Disks
- Processing
- Processed storage
- Near Real Time Processing and Distribution
- Post Pass Processing and Distribution

Each and every one of these stages are further elaborated on in 3.3

3.2 Server Hardware and Configuration

This section presents the front end server, and how the server is configured. Note that this is what KSPT offer to customers as of today, and is under constant revision.

3.2.1 Server Hardware Specs and Software

The server delivered to customers today is an HP ProLiant ML350 Gen8, rack mounted chassis. The server use SUSE Linux Enterprise Server version 11 (SLES-11) as operating System (OS), and all disks installed use the XFS file system. The total storage capacity on the server is 8.7 terrabyte, where 2.4 terrabyte is used for data capture storage. The rest is allocated to store processed data, and to the OS.

3.2.2 Configuration

The different configuration details for the server ingest disks, system disks and processing storage are covered in this section.

Ingest Disks The server ingest disks consist of 4 HDDs running at 15,000 RPM. They are set up in a RAID with level 1+0. Raid 1+0 initializes two disks with raid level 1 (mirroring the disks), and stripes the two distributing data evenly across them. This is shown in figure 3.2 The reason for this configuration is because of its level of redundancy, theoretically allowing two disk failures within the RAID. The ingest disks does store data from previous passes as

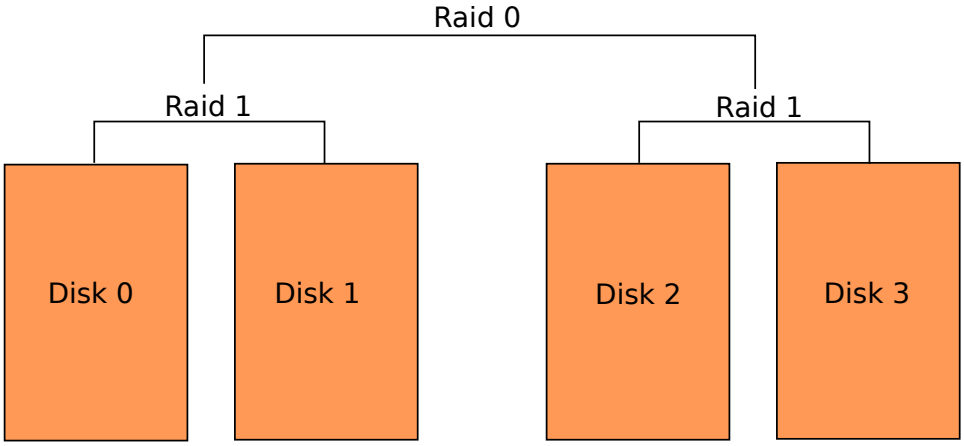


Figure 3.2: Two RAID level 1 are striped with RAID level 0, theoretically allowing two disks failures if and only if it happens within different mirroring RAIDs.

well. Eviction algorithms delete data from the oldest passes before a new pass, in order to make space for the new one. This is based upon what satellite is scheduled next and the amount of expected data. The eviction algorithm makes sure that the ingest disks never exceed 70% utilization.

Processing Storage When a product needs some sort of processing before distribution, it is stored on a RAID level 1+0, as the ingest disks. The only difference is that this raid consist of 6 disks, instead of 4.

System disks As for the system disks running the OS, two HDDs are mirrored through RAID level 1. This is purely done for fault tolerance where a crashed disk can be hot swapped[1] with a new disk without affecting the system uptime.

Process priority As data capture is a very I/O intensive process, meaning interrupts can cause loss of data. Therefore the capturing process is initialized with the highest priority (nice[2] value -20).

1. Hot swapping refers to the ability to remove a disk without powering down the system.

3.3 Execution Flow

MEOS Capture support two different initialization schemes where the first is data driven, as described by Cordier et. al. [12], and the second is schedule driven. The difference between the two is that schedule driven is pre programmed, based upon the time for when the satellite becomes visible to the ground station, and data driven are based upon automatic detection of data. Independent of the reception initialization scheme, the high rate demodulator and front end processor (HRDFEP) is configured and programmed to execute the steps listed in section 3.1.1.

The first module is the demodulator which demodulates radio signals received by the antenna, creating a digital stream of data. Demodulated data is sent to the server, or more specific, the data receiving board (marked as input channel in figure 3.1) which performs several operations on the received data. The data is frame synchronized and de-randomized through CCSDS coding[67]. At last the data is error corrected via algorithms such as Reed-Solomon decoding[36, 72] and Viterbi[71], timestamped and placed in an on-board memory pool[36]. From this memory pool, the data is fetched by an FPGA which implements an interface to the PCIe bus and puts the data onto said bus through DMA. The data then arrives in host RAM before it is written to disk. For some customers persistently stored data has to be processed before distribution. The data is then read from that particular ingest disk, processed and written to another disk before it is distributed.

Through serial output, it is possible to send all received data to tape recorders or other machines for further storage, processing and/or distribution. The different processing modules represent the different processing necessary for different customers. This processing can be the generation of metadata (also known as data about data) and source packet generation which convert the data to a specific file format which is the format delivered by the specific instrument. However, processed data are also stored to disk for persistent storage before it is distributed. Do note that there are two types of distribution. Near real-time distribution and post pass distribution. The difference is that near-real time distribution distributes processed data as soon as it is available, where post-pass distribution waits until the satellite moved out of antenna bounds (or finished the pass), and all data requiring processing have been processed before the distribution takes place.

Even though it is possible to send received data through the serial output to other computers, this is not common practice as the system would require several computers which would increase the total cost. Everything is normally handled by one single front end server with the hardware specifications and configurations it was installed with.

3.3.1 Known Bottlenecks

On the data generator board there is one known bottleneck. A memory buffer used for temporary storage before data is sent out on the PCIe bus, limiting the system because it must be written to and read simultaneously, which provides some “read my writes" constraints. However, this issue is not within the scope for this thesis because this occur on the data generator board on some of the FPGAs used.

3.4 Related Systems

KSPT is not the only provider of systems capturing satellite data. This chapter focuses on related systems, presenting their manufacturers and some key numbers and functionality to compare with MEOS capture.

ViaSat offer a high rate receiver with an optional front end processor which they claim to support up to 3200Mbps per channel, with support for two channels[70]. In effect, ViaSat guarantee lossless data capture of rates up to 6400MB per second. With processing and data archive enabled, they guarantee rates up to 4000Mbps per second.

Zodiac Aerospace deliver a system called Cortex HDR XXL[73], which they claim to support rates up to 2Gbps, over two input channels. Their data sheet say that they do deliver a front end processor as well, but they do not state whether they guarantee 2Gbps with processing enabled, or if it is only data capture.

RT Logic have developed a system called T1200HDR[60], which they claim to be capable of capturing rates up to 2Gbps according to their data sheet.

Antwerp Space’s Omnisat[4] claim that their system support rates up to 975Mbps. Omnisat offer a front end processor, capable of data distribution, but it is not informed whether this is near real-time, or post pass distribution.

All mentioned systems report to support turbo coding from CCSDS, and several different modulation schemes. The differences seems to be within the rates guaranteed, and to what level of processing is offered on the captured data.

The information presented in this section is based upon the information provided in the systems data sheet, and the rate the system claims to support. When assessing capturing systems, there are several parameters one must

consider. Such parameters are;

- Claimed rate
- Functionality
- Implementation loss
- Decoding
- Static and dynamic status logging
- Level of automation
- To what extent claimed performance is valid for all functionality simultaneously

The claimed rate is limited for a system providing near real-time processing, since the captured data may have to be read from disks which is exposed to heavy load as they capture data. The functionality provided refers to the ability to process the captured data, and to what level. The implementation loss typically occurs within the demodulator, and will degrade recorded data quality. The decoding may be iterative which again affect the quality of the frames received by the demodulator. The more iterations supported, the better the data quality.

4

Analysis Tool

This chapter presents the analysis tool, with its design and implementation. Some requirements to the analysis tool is presented, for evaluation purposes. The analysis tool purpose is to monitor hardware components during a simulated satellite pass and store the sampled data for further analysis. This creates the foundation for determining where possible bottlenecks may appear within the data capture system, and further, resolving them.

In order to determine where the bottlenecks appear, one must monitor several components within the computer. To determine what components to monitor one must take MEOS capture architecture into consideration, presented in figure 3.1. As the incoming bitstream from the satellite antenna arrives at the receiving server, the data receiving board place it in host memory through the PCIe bus. This requires cooperation between five components to accomplish, the PCIe bus, the DMAC, the CPU, kernel buffers and host memory. Therefore, one must monitor these five components to achieve an understanding of how the components cope with the incoming bitstream.

As data moves on from host memory to the ingest disks, there are other components in action. A RAID controller fetch data from host memory, distributing data across the RAID used as ingest disks. Therefore the RAID controller and the disks used as ingest disks must be monitored, in addition to those mentioned above. This provides a complete list of required components to monitor in order to achieve a precise picture of where possible bottlenecks may occur. The list of required devices to monitor is presented in listing 4.1.

Listing 4.1: List of analysis requirements. The list names all devices that require monitoring during experiment execution.

- PCIe bus
- DMAC
- CPU
- Memory usage
- Kernel buffer usage
- RAID controller
- Ingest disks

The analysis tool is configured to be a scientific tool within this thesis, and some changes must be done before it can be used in production. Production refers to existing systems performing data capture on a daily basis for customers. A suggestion for production setup is presented in section 4.3.

4.1 Design and Architecture

The analysis tool is designed as an overlay on top of existing monitoring tools within Linux. As all tools used, except for Dstat, are part of the Linux package everyone is free to use these tools. One may have to download and install Dstat manually, and the Dstat package can be found on GitHub[1]. The total package is a combination of several programs that allow you to monitor the total load on the computer at any time. The idea is to split a terminal window into several sub-windows, each window executing a carefully chosen tool to provide its user with information. Terminal multiplexer (tmux) is a tool that implements this functionality, where one can configure the size of each window within the screen bounds.

Terminal Multiplexer tmux[2] is a tool that allow its user to create, delete and switch between panes within a single terminal window. This is useful

1. https://github.com/dagwieers/dstat
2. http://www.openbsd.org/cgi-bin/man.cgi/OpenBSD-current/man1/tmux.1?query=tmux&sec=1

when one needs several windows within a terminal, such as when one want to monitor the system load on a computer using several tools.

Dstat Dstat[1] is a monitoring tool that allows its users to monitor the load components within a computer are exposed to. Dstat supports logging which enables further analysis when needed. Dstat is built on top of the Sysstat[3] group, which consists of several system monitoring tools[4]. Throughout this project, Dstat is used to monitor CPU usage, memory usage, disk input and output requests and the amount of data read or written to disk per second. One limitation in the monitoring of the ingest disks are that Dstat does not allow its user to monitor each and every physical disk. It is based upon the logical disks, and this thesis does only experiment with several physical disks, arranged in some RAID which is exposed to the OS as one, big, logical disk.

Dstat allows creation of your own plugins. Through this feature, one can tailor Dstat to your requirements without any larger effort. Since Dstat is built on top of the Sysstat group, it utilizes the files and catalogs within `/proc` where Linux keeps its device logs. The device logs' purpose is to log the current load per device, that be CPU utilization, memory usage, disk IO statistics and more. Dstat can be invoked with a delay, which says how often the statistics should be read. This must be an integer, and default is set to 1 second. According to the Dstat Linux manual page[5], Dstat does not display snapshots but rather the average system usage since the last update.

Dstat supports several flags[1], and the most relevant for this project are the ones that collect information about CPU usage, memory usage, disk utilization and I/O. The samples also include a timestamp for when the sample was taken. One may also set for how long Dstat should run before it terminates. Dstat also has support for logging its generated data samples, writing it to a file in comma separated values (CSV) format.

Other tools that monitor the same components does exist, achieving the same result. Since Dstat is an overlay over the Sysstat group, one could use the tools within Sysstat such as Iostat[6], which monitor disk transactions and input/output, and Vmstat[7] that monitors virtual memory usage. This would however require a combination of several tools to achieve the same result, which does not support logging files out of the box. The convenience of using Dstat, is that all this is gathered within one single tool, reducing the needed number of tools to monitor the same components in addition to its logging features.

3. `http://man7.org/linux/man-pages/man5/sysstat.5.html`
4. `http://sebastien.godard.pagesperso-orange.fr/documentation.html`
5. `http://linux.die.net/man/1/dstat`
6. http://linux.die.net/man/1/iostat
7. http://linux.die.net/man/8/vmstat

Kernel Log The analysis tool include a tmux pane displaying a log that is only used by the OS kernel. Since the data receiving board device driver run within the kernel, it has access to the kernel logs. Therefore, the device driver reports to the kernel log whenever an unexpected event occurs. Because the kernel logs tend to be very long, a program called Tail[8] is used to watch the last part of the log. Tail accepts user specified amount of lines to display, as shown in listing 4.2,

Listing 4.2: Example of Tail usage, to display and follow the last 15 lines of a file.

```
tail -f -n 15 <path/to/file>
```

will only show the 15 last lines of a specified file, and follow these lines as new messages are appended to it. For convenience, this file is the kernel log throughout this project.

All device drivers has access to the kernel logs, where the message printed from the data receiving boards driver is very specific for this particular system. Therefore, there does not exist any alternative monitoring tool. KSPT have not implemented any tool with this feature, this is purely based on human revision. Therefore it is included within this tool, in order to display it immediately to its user when the event occur.

Htop Htop is an interactive process viewer that, according to the manual page[9], allows its users to monitor the resource usage for each process running on the computer. Htop is included into this system because it provides overall CPU statistics per core, and it allows the user to manage processes runtime. Because of the non existent logging feature within this tool, the CPU statistics are logged from Dstat instead. But for monitoring momentary CPU usage and process management, this tool is invaluable.

Top is an alternative process manager that could be used. However, Htop offers more statistics, with a graphical visualization for per CPU core utilization, and memory usage, in addition to the process management. Htop is a more powerful tool than Top, which is why Htop was preferred over Top.

4.1.1 Architecture

Based on the design described, the architecture is shown in figure 4.1. The figure reflect the design proposed, where Tmux is used to split the terminal window into sub-windows, each presenting one tool used for the analysis.

8. http://man7.org/linux/man-pages/man1/tail.1.html
9. http://linux.die.net/man/1/htop

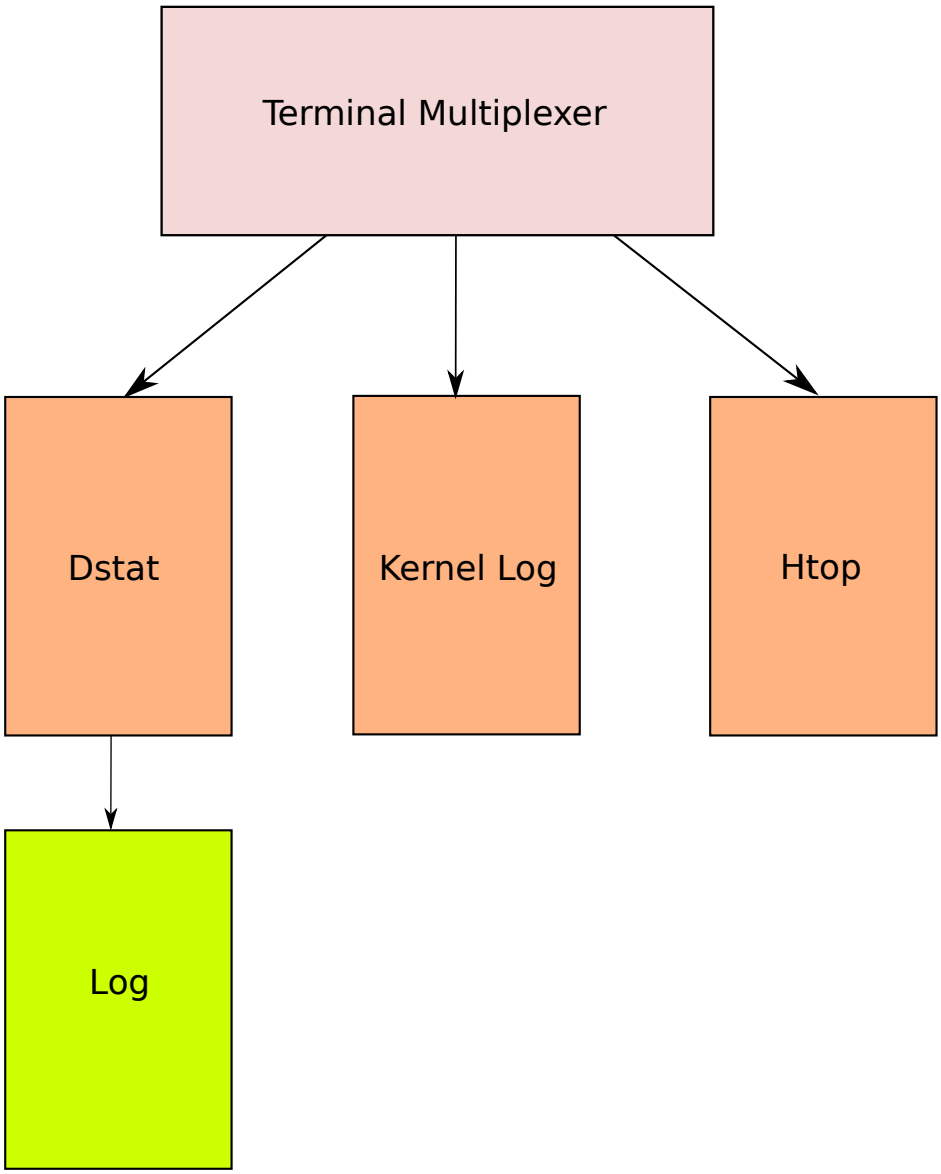


Figure 4.1: The analysis tool architecture

4.2 Implementation

Since the analysis tool exclusively consists of already existing tools native to Linux, the implementation became trivial as no new tools had to be developed. A script written in Bash launch TMUX before each and every tool presented in figure 4.1. Dstat is launched with several parameters in order to inform Dstat what to monitor. The decisions on what to monitor was based upon the analysis requirement stated in listing 4.1. Htop did not require any further parameters, while Linux kernel buffer usage was monitored by following the last 15 lines of the Linux kernel log, as presented in listing 4.2. Now, the implementation does only start different programs, with carefully chosen parameters.

The implementation was however a bit varying between the different experiments, due to the difference in experiments. When testing the ability to only capture data, Dstat only logged the read and write accesses to the ingest disks. When simulating both the capture of data and near real-time processing, Dstat was configured to log both the activity on the ingest disks as well as the activity on the disks used to store processed data.

The bash script first initializes an array of commands to run, which reflect what to monitor during execution. Next the script checks if a TMUX session named `monitors` already exists. If such session already exist, TMUX just attaches itself to that session, in order not to have several instances of the same process running during experiment execution. This is a neat feature when several users want to monitor the same server, from their personal computers. However, if the session does not already exist, each and every of the tools are launched and the terminal window split into several panes, with each pane executing one tool. The script is presented in its entirety in appendix B.

The implementation enables the gathering of several tools within one terminal window, which again offers its users an overview of what is happening on the computer in a meaningful manner.

4.3 Discussion

As the analysis requirements was stated in listing 4.1, one may have noticed that not all components are monitored with the system used within this thesis. The components not monitored are the PCIE bus, the DMAC and the RAID controller. In order to be able to monitor these components, one must be in possession of the driver source code implementing the components interface. The PCIE bus can be monitored through a PCIE bus analyzer, which have not been used in this thesis. KSPT is not in possession of the source code for any of

these components, as they are bought from third parties. This means that KSPT consider them as "black boxes". With the source code available, one could expand the analysis tool by implementing a Dstat plugin using the component interface to retrieve run time statistics.

The statistics that would be interesting to know from the different components not monitored today are discussed below.

PCIe bus To monitor the PCIe bus, which is provided as an extra device one must place between the PCIe connector and the PCIe compliant device. Teledyne LeCroy offers such a device[66] compliant with PCIe version 2.0 and 1.0. Through this device one will gain valuable information about per lane throughput in real time. Interesting metrics to pull out from this would be how much time the bus spend per transaction, and the total per lane throughput.

Since a PCIe analyzer is not provided in this project, one must determine the throughput by performing isolated experiments between the data receiving board and host memory. Through modifying the data receiving boards drivers, one could achieve some level of run time statistics such as how many clock cycles the FPGAs actually put data out on the bus, and how many they have to hold data back.

DMAC The DMAC requests a list of page descriptors from host memory. This memory reflect the kernel buffer size, which is configurable in MEOS capture. As data are written to host memory (or RAM), page by page, the file system makes sure that the RAID controller begin to write data to disk. The list of page descriptors works as a circular list. When the RAID controller cannot fetch data fast enough from RAM, meaning that the ingest disks are too slow, the DMAC can run through the list of page descriptors and overwrite data not written to disk yet. As of today, there are no mechanisms to stop this from happening. Therefore, it would be interesting to monitor where in the list of page descriptors the FPGAs on the data receiving card is, compared to how far the RAID controller has come. This could be implemented in the data receiving board device driver, as the DMAC is implemented on the FPGA placed on the data receiving board.

RAID controller The RAID controller used within this thesis is from Hewlett-Packard. It is specified to support 12Gbps which is equal to the specifications of the SAS bus. If one is in the possession of the firmwares source code, one would learn more about how the RAID controller distributes data across disks participating in the RAID, and one could monitor per physical disk statistics instead of per logical disk statistics as done today. This would provide a more precise picture of how the ingest disks cope with incoming rates as they increase.

This is very dependant on which RAID level one chose to use, especially when using data redundancy.

Since KSPT solely depend on the Linux kernel and file system to take care of captured data after it is placed in host memory, this is also considered a "black box", offering reliable data transfer from host memory to the ingest disks. Monitoring the RAID controller could offer more detailed information about this part of the pipeline, and possibly reveal existing bottlenecks.

Kernel Log Monitoring the kernel log is purely based upon human revision, through this tool. An attempt was made to implement a Dstat plugin retrieving the current buffer usage, but this was only partly successful. The reason for this is that the kernel print messages to its log per microsecond when the buffer usage is exceeds one buffer. As system fetched the last line from the kernel log and extracted the number of used buffers, if any, the buffer usage could have been much larger just few microseconds ago. The issue is that Dstat only update the numbers per second. One could fetch the 10 latest lines and display an average of the 10, but as the number of buffers used becomes equal to the number of buffers available, or worse, larger, the user must be informed in some way. Therefore an average is not a suitable solution. One could log the maximum recorded value and print that to screen, but it would not be as informative as the momentary usage of buffers. It would only inform whether there was an overflow or not during the experiment, in which there are other mechanisms embracing this feature. Logging the maximum value would be redundant. Therefore, the monitoring of buffer usage in kernel, is purely based upon human revision and not integrated into Dstat through a plugin.

4.3.1 Setup for production purposes

The analysis tool can very easily be configured for production use. The first change to do is to create a mechanism that enables logging of events, i.e., when resource utilization gets very high. This can be implemented, because MEOS capture writes the currently used numbers of buffers in kernel to `/var/log/messages`, which is the kernel log. When MEOS capture uses more than one buffer in kernel, the incoming rate is greater than what the ingest disks can write per second for a short period of time. The analysis tool may provide information about what caused this drop in disk performance.

For example, if MEOS capture is invoked with kernel buffer size of 64MB, the kernel has 16 buffers of 4MB each available. To set this in context, when the satellite transmits data at a higher rate than the system can write to disk, the buffers in kernel quickly get filled up. Normally, MEOS capture uses one, and

only one buffer in kernel. When MEOS capture has to use more than one buffer in kernel, it writes the number of used buffers to its log. This can be monitored, and invoke this analysis tool for logging purposes on the whole system only on this event. To do this sort of mechanism, what one must do is to to use `tail` displaying the last 10 lines of the kernel log. The last 10 lines should at first contain old values, where the last line should inform when the data receiving board device driver was initiated. The message in listing 4.3 occur within the kernel log during MEOS capture execution and inform the number of buffers used.

Listing 4.3: Example of kernel log message reporting number of used kernel buffers.

```
2016-04-05T13:28:29.864062+00:00 fes39-vlan5 kernel:
[362474.731050] sfep: (0x20)intcount=2
```

The message show a timestamp for when the message was reported and the host name. Next there is a timestamp in microseconds determining when the message was reported since the device driver was initiated. After the timestamp, the message display that it is the data receiving board device driver that reports the message (sfep), and at last the `intcount` is the number of currently used buffers.

4.3.2 Evaluation

As the analysis tool does not implement all requirements mentioned in listing 4.1, one can still achieve a good idea of where the bottlenecks appear within the system. Nevertheless, one can gain information about the parts not monitored by isolating parts of the pipeline, and hence determine where bottlenecks occur. With this approach, one must rely more upon qualified assumptions to fill in the missing parts within the analysis.

The analysis is very reliant upon Dstat, in order to produce runtime statistics. The fact that Dstat does support creation of logs is invaluable, which is not provided natively in many other monitoring tools within the Sysstat group. Htop is used as process manager due to its user interface and runtime process management. As Htop visualizes per core CPU utilization and memory usage, it provides instant feedback to its user whether the server is under heavy loads or not. In addition, its process management is very powerful, as it allows one to sort processes by resource usage.

In total, the analysis tool provides its users with much valuable information, gathered in one window. The support of logging features enables users to create web pages with live graphs showing computer load, in order to monitor it

remotely. The analysis tool is highly configurable through the tools used.

4.4 Future Work

Since some components in the system pipeline are not monitored as of today, it is regarded as future work. As discussed in section 4.3 monitoring the DMAC, RAID controller and PCIe bus would provide useful statistics when determining where bottlenecks occur. In addition, defining a suitable solution for monitoring the instantaneously buffer usage during a satellite pass would instantly inform the user whether the data capture is successful or not. It is possible that one must develop a system only focusing on the kernel log in order to display the momentary usage of buffers. The issue is that messages are posted to the kernel log very rapidly once the buffer usage exceeds one buffer. One may have to look at other solutions, i.e., store the highest value reported per pass. If one just display the values naïvely to screen, the buffer usage may not be very human readable since the update frequency may become per millisecond.

In addition the changes needed for production purposes presented in section 4.3.1, is left out as future work.

5

Experiments and Evaluation

As MEOS capture is a large and complex system, this chapter embrace only a small part of it as the device under test is the next generation HP ProLiant server with candidate hardware and technology. This chapter presents the experiments performed on the test environment, determining the performance provided withconfigurations KSPT use today.

The analysis tool presented in chapter 4 will be used to monitor the computer load for all experiments. As the first few seconds of the experiments will contain values from other processes and daemons, we will ignore the first few seconds of all experiments.

5.1 Test Environment

The test environment differ from the system KSPT deliver to their customers for production. The main difference is that the test environment use a 9th generation HP ProLiant server, where the production system use 8th generation HP ProLiant server.

5.1.1 Hardware Specs and Software

The test environment includes the following components.

Chassis The server chassis is a rack mounted HP ProLiant ML350 Gen9[1].

CPU The processors are two Intel Xeon E5-2623v3[2] running on 3Ghz, quad-core, with 10 MB cache and 64-bit instruction set.

Memory The memory is 128GB of RAM, divided over four 16GB Dual Rank x4 DDR4-2133 CAS-15-15-15 Registered Memory Kit[3] running at 2133 MHz.

Storage It is installed with 10 HP 600GB 12G SAS 15k rpm HDD[4], and two HP 200GB 12G SAS Mainstream Endurance SFF 2.5-in ENT Mainstream SC 3yr Wty H2 Solid State Drive[5] capable of 70,000 read IOPS and 51,000 write IOPS according to HP's sites[25]. It is also equipped with two HP 146GB 6G SAS 15K rpm SFF (2.5-inch) SC Enterprise 3yr Warranty Hard Drive[6].

RAID-controller The RAID-controller is a HP Smart Array P440/4GB FBWC 12Gb 1-port Int FIO SAS Controller[7].

Data Receiving Board The data receiver runs in testmode during all experiments. When testmode is enabled the FPGA generates a pre defined sequence of data at a user defined rate, and sends it out on the bus.

Software As for the software, the server run SUSE Linux Enterprise Server version 12 (SLES-12) and XFS file system.

Configurations Finally, the system configurations will initially be as defined in section 3.2.2, where the only differences are the hardware available within the test environment.

1. http://h20195.www2.hp.com/v2/GetPDF.aspx/c04346270.pdf
2. http://ark.intel.com/products/83354/Intel-Xeon-Processor-E5-2623-v3-10M-Cache-3_00-GHz
3. http://www8.hp.com/us/en/products/smartmemory/product-detail.html?oid=6987437
4. http://www8.hp.com/us/en/products/server-hard-drives/product-detail.html?oid=6826123#!tab=specs
5. http://www8.hp.com/ph/en/products/proliant-servers/product-detail.html?oid=7153470#!tab=specs
6. http://www8.hp.com/lamerica_nsc_carib/en/products/oas/product-detail.html?oid=5194515
7. http://www8.hp.com/h20195/v2/getpdf.aspx/c04346277.pdf?ver=1.0

5.1.2 Perfio

KSPT has developed a program named Perfio for performance testing. The architecture for Perfio is presented in figure 5.1. Perfio initializes the data receiving board, to generate a pre defined sequence of data. Perfio takes several user inputs, specifying the execution of the program. The parameters used are as follows.

- Select the test generator (Important when using two test generators)
- Number of seconds to run
- Data destination (Typically a file)
- Enable DMA
- Kernel buffer size
- Enable test mode
- Requested data rate

Since the test server can be configured with several logical disks, using different RAIDs, each RAID is mounted to a directory, which name reflect what raid it implements. The data destination will be the folder a particular RAID is mounted to, in which one want to do experiments on. An example of how to run Perfio is presented in listing 5.2. From the example one can see that Perfio is instructed to run for 600 seconds, writing data to `/raid10/testfile`, using 36MB buffer space in kernel and with a requested rate of 350MBps.

Before the program execution begins Perfio initializes the data receiving board installed in the test server, test mode enabled, specifying the data rate. When the execution starts, the DMAC allocates memory in kernel equal to the kernel buffer size as specified by user input. The DMAC receives a list of page descriptors. Each page descriptor contains information about where in physical memory the 4MB page exist, and hence where the data should be written to. The FPGA that sends data to the PCIe bus, must request a page descriptor from host memory before the data is sent out on the bus. To limit the rate at which the DMAC writes data to memory at, clock cycles are used to limit the amount of data entering the PCIe bus per second. The system can be even more delayed if the DMAC has to wait for the page descriptor from host memory. When data arrives in memory, the RAID controller immediately fetch data via the device driver from memory in order to write it to persistent storage, the ingest disks.

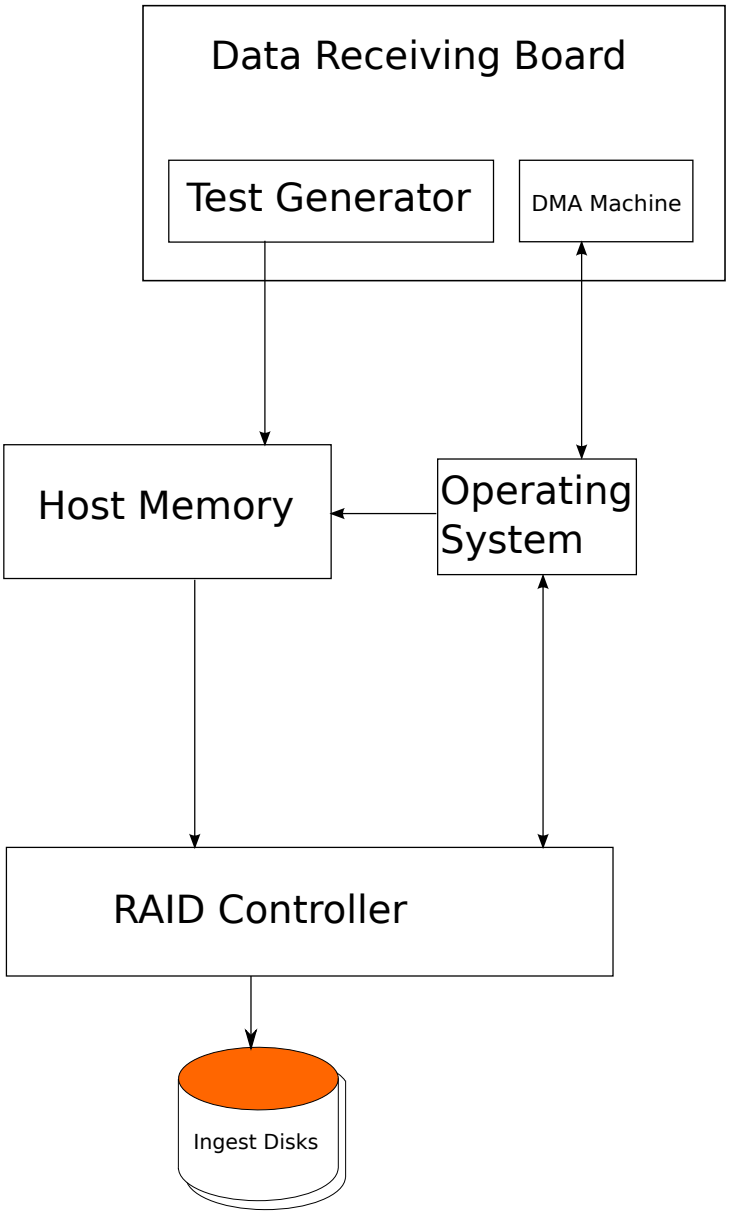


Figure 5.1: Figure illustrates the architecture for Perfio, and it is considered equivalent with MEOS capture.

The data generated by Perfio, is a sequence of 1024 byte frames. Each frame consists of a synchronizing word[8], and consist of 254 four byte words containing a counter that counts upwards by one for every word. An example of the data sequence from Perfio is presented in listing 5.1. One should avoid using all zeroes as data pattern because of smart caching mechanisms and predictions.

Listing 5.1: Example output from Perfio, with the syncword intact.

```
0000400: 1acf fc1d 0000 0001 0000 00fe 0000 00ff
    ................
```

Perfio display the requested rate, specified by the user. Perfio does only support rates that are a multiple of 62, 5, which is caused by a limitation in the FPGA used to generate test data. Therefore Perfio adjusts the requested rate to the closest rate which is a multiple of 62, 5 and reports this as the `actual rate`. Perfio also reports the exact amount of bytes written to disk, and how many seconds it was executing. Based on the amount of bytes written, and the exact execution time, the average rate is calculated. This is reported as `achieved rate`. The achieved rate is used to calculate how much the FPGA has been limited during execution, by calculating how many percent the achieved rate is of the actual rate. The functionality implemented in Perfio is considered equal to MEOS capture, in terms of the data path after the satellite signal has been received by the antenna, demodulated and arrived at the data receiving board.

Listing 5.2: Example command illustrating the input parameters used to determine execution flow for Perfio.

```
perfio  --device /dev/sfep0a
        --time 600
        --file /raid10/testfile
        --dma
        --kernelmem 36
        --pcie_tg 350
```

As Perfio generates data, it provides its user with some runtime information if the system seems to overflow. To be sure that the generated data is intact, KSPT developed a naïve program that runs through the file looking for errors in the byte sequence. This program was used to verify the dataset generated in all experiments.

8. The synchronizing word is `0x1acffc1d`

5.2 Determining Maximum Throughput

The experiments have been conducted in a test environment as described in section 5.1, and the first experiment determines the maximum rate supported by the next generation HP ProLiant servers with the ingest disk configurations used today[9]. This is done to create a new baseline, as the set of hardware is new and no numbers in terms of throughput are known so far. The analysis tool will monitor the system load during all experiments, to gain information about possible bottlenecks.

In order to make the experiments as close as possible to a real world satellite pass the length of all tests was set to 600 seconds, or 10 minutes. Some satellites may have even longer passes, but the majority of all polar orbit satellites finish a pass during a 10 minute period.

All experiments presented in this section, will only use one data receiving board in test mode, unless otherwise specified. The environment does support several data receiving boards.

5.2.1 Simulating Capture Only

To determine the maximum rate supported by todays system as described in section 5.1, the first experiment started out with an ambiguous rate in comparison with what was known as the maximum rate. The experiments purpose, is to determine the maximum throughput of the system pipeline, without any processing enabled. This will provide a baseline for the capturing rate supported by next generation HP ProLiant servers capture without data loss. The results will be analyzed in search for bottlenecks.

The initial requested rate was 350 MBps (or 2.8 Gbps) over 600 seconds. The average achieved rate was 331.8 MBps, transferring 194GB over 599.2 seconds. Since the experiment goal was to determine the absolute maximum rate this configuration and test environment is able to capture, the rate was increased in steps of 10 MB per experiment iteration. The initial experiment did not show any sign of lost data, or heavy load before 500 MBps was passed, and the limit seems to be at a requested rate of 540 MBps. The parameters used for the last experiment are presented in table 5.1, and the resulting output from Perfio are presented in table 5.2.

Note that the data destination is a directory that the ingest disks have been mounted to.

9. RAID level 1+0

Seconds to run	600 seconds
Data destination	/raid10/testfile
Iterations	1
Kernel buffer size	36MB
Data rate	540MBps

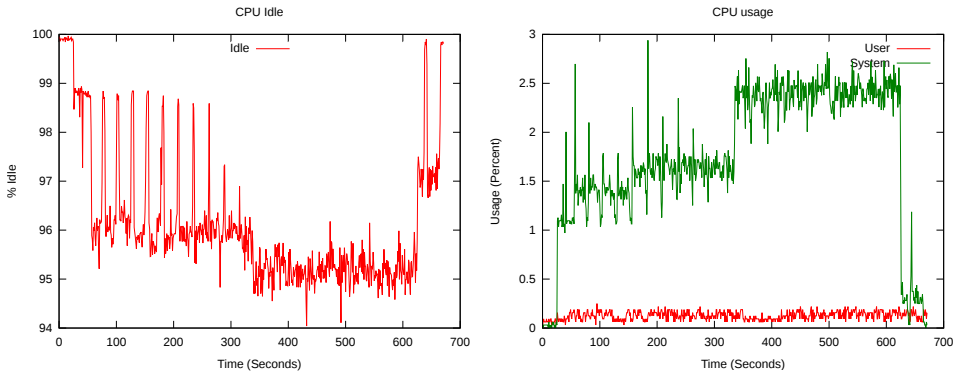
Table 5.1: Perfio input parameters for highest successful data rate, with next generation servers.

Requested rate	540 MBps
Actual max rate	539.06 MBps
Total data written	298, 173, 071, 360 bytes
Exact runtime	599.38 seconds
Pages written	72, 796, 679 pages
Page size	4096 bytes
Achieved rate	474.43MBps
Limited to	88.0 % of max rate

Table 5.2: Perfio output for testrun with the highest successful data rate

As the output from Perfio states, the test generator FPGA has been held back to 88.0% of its configured max rate. This is a good indication that there is a bottleneck somewhere. In a perfect environment, the FPGA would yield data to the PCIe bus 100% of the time, but this can not be achieved due to the PCIe protocol overhead and the fetching of page descriptors from host memory. It is observed that as the data rate increases, the FPGA is increasingly held back. The output from Perfio also tells that the total amount of data generated and written to disk through this experiment is 283.6 GB. There is a rate difference in requested data rate and actual data rate because the FPGA generating data can, at the lowest, write data at 62.5 Mbps, which is also the step size.

As figure 5.2 suggests, the CPU is not the bottleneck within this system, due to the time spent idle is 95-96% on average. The lowest recorded value is just above 94%, and one can see that the CPU becomes more occupied just after 300 seconds. That is because the memory usage reaches its peak at the same time, which means the CPU has to spend more time administrating the file system and permitting the RAID controller access to the memory space in which the data lies in. The memory usage is further elaborated on later. The figure also show the percentage of time spent on user and system level processes. One can see that system level processes spend the most time running on the CPU which is caused by the process priority set for Perfio. The reason for the low CPU usage, is because the system use DMA where the CPU only provide the DMAC access to the memory locations needed for the operation. Because of these results, further analysis is needed.



(a) Illustrates the amount of time the CPU spends idle

(b) Visualization of scheduled runtime for user and system level processes.

Figure 5.2: Figure shows the amount of time the CPU spends idle, and the total amount of time user and system level processes actually run on the CPU. The figure suggests that the CPU is not a bottleneck within this system.

To determine whether there exist bottlenecks in this system, one must look at memory and disk load. The memory usage is presented in figure 5.3. As the figure illustrates, the disk cache increases almost linearly in size until it peaks just above 120GB after approximately 330 seconds. If one recall from figure 5.2, the system level processes required more resources from the CPU after 330 seconds of execution. It looks like the memory usage is closely related to this, as the server is forced to free pages in memory in order not to overflow the memory buffers. The memory used are as one can see only a few GB. This suggest that the memory is not the bottleneck within the system.

The amount of data written to disk per second is visualized in figure 5.4, and support the presented average write speed from table 5.2. The disk activity also shows stable behaviour from 290 seconds of execution until the experiment ends. The spikes shown in the first 290 seconds is caused by the file system caching mechanisms and temporary storage in diverse caches (such as disk caches, caches on the RAID controller). After 330 seconds we know that the memory has been filled up with data, and it seems like the caching effects disappear after 330 seconds due to the stable disk behaviour.

The average load is presented in figure 5.5. The average load is a visualization of processes placed in queue to run on the CPU. As the graph shows, the last minute load peaks on just about 1.6 meaning one CPU core are overloaded by 60% on a single core system. This is however not the case for the test system because the total number CPU cores in the test environment is 32, meaning that 32 processes or threads can execute in parallel. The average load can very well pass 25 before it becomes threatening in any way. As the figure shows,

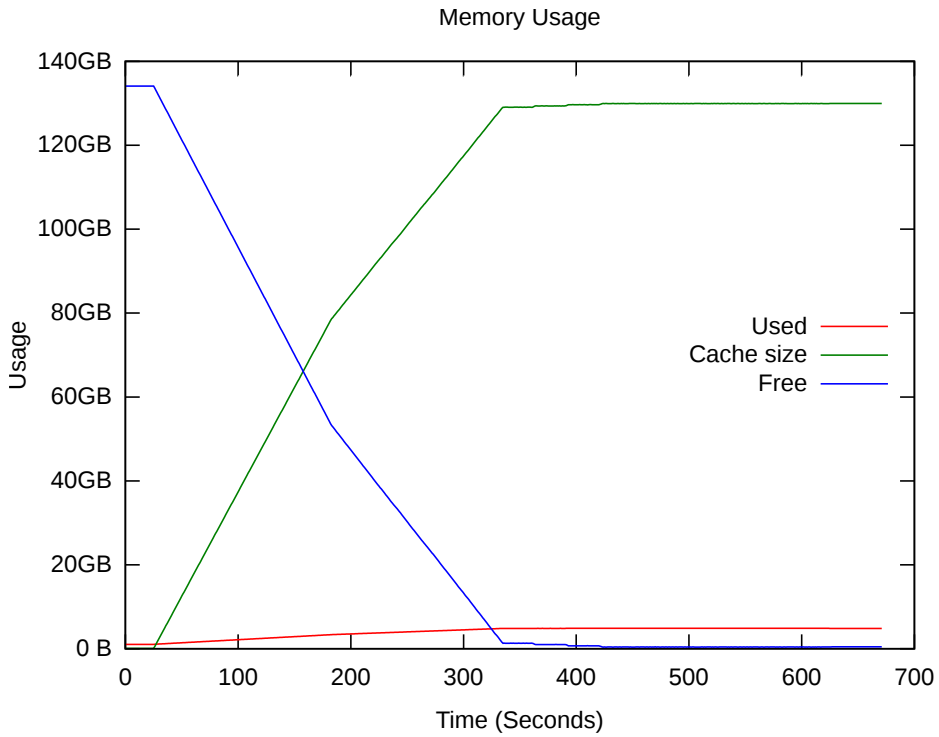


Figure 5.3: Figure illustrates the memory usage for the whole experiment execution. Note that the disk caches use almost all memory available.

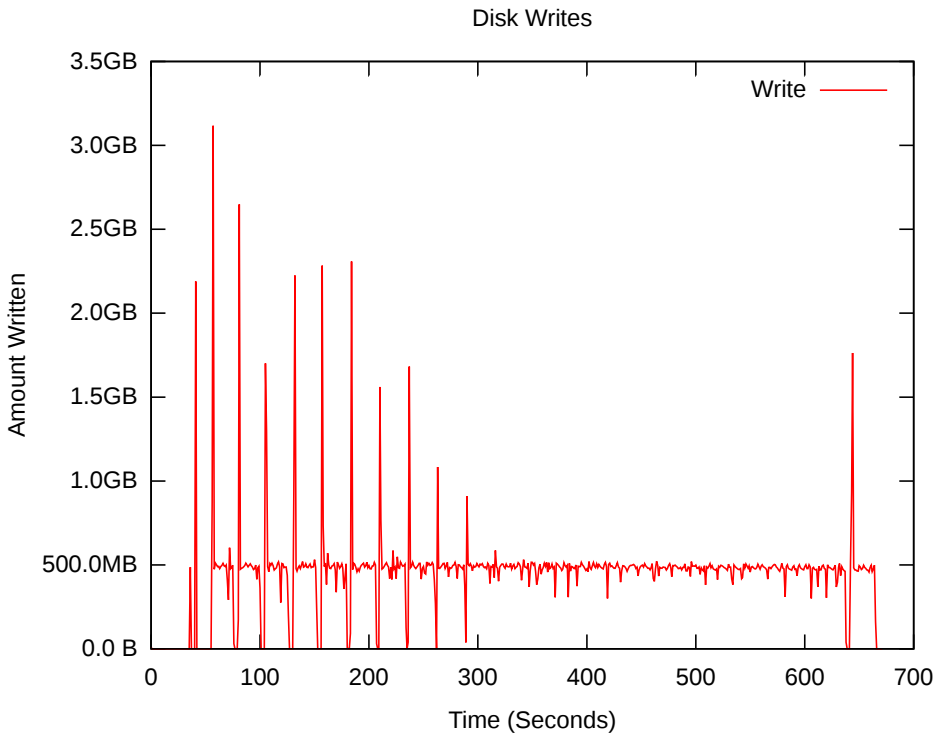


Figure 5.4: The amount of data written to disk per second. The spikes indicates that a lot of data have been written to the disk cache, and one can see that the general write speed is just below 500MBPS, as the results in table 5.2 suggests.

the last minute bar is the one most unstable. The reason for that is because the integration time is much shorter for the last minute compared to the last 5 and 15 minutes. The last minute does however show that on average two processes waited in queue to execute on the CPU after 550 seconds, and it is speculated that this is a consequence of the process priority set for Perfio. As the experiment duration is 600 seconds, or 10 minutes, the last 5 minutes and the last 15 minutes is the one most representative for the load generated on the CPU. As the line representing the last 15 minutes shows, the average load just passes 1 process, which is a good sign. Figure 5.2 show that it spend most of its time executing system level processes, which in this case will be the administration of the file system. So from the graph, one can read that the last 15 minutes 1.2 processes in average waited for execution time on the CPU. An observation through the experiments is that if a daemon of some sort perform a sanity check to the file system, it causes the system to overflow as Perfio executes due to two-way traffic through the RAID controller. This is further elaborated on in chapter 6.

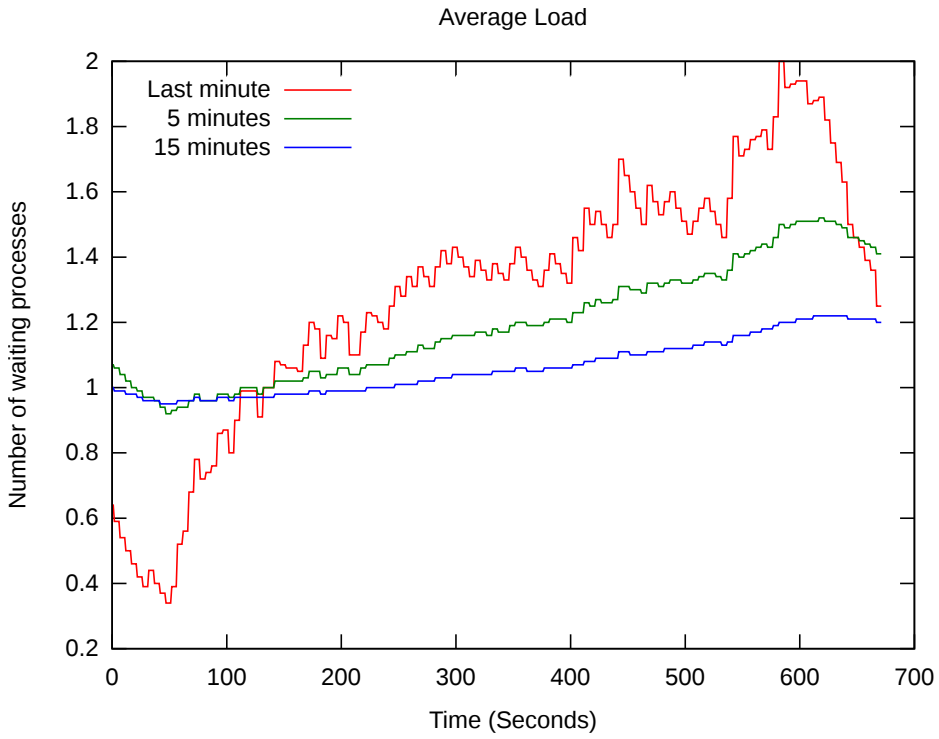


Figure 5.5: The graph illustrates the number of processes waiting in queue to run on the two CPUs. Pay the most attention to 5 and 15 minutes, as they are the ones most stable ones due to the longer integration time.

To sum up the results, the CPU is not exposed for significant load, the memory

Seconds to run	600 seconds
Data destination	/raid10/testfile
Iterations	1
Kernel buffer size	36MB
Data rate	113MBps

Table 5.3: Perfio input parameters for maximum successful rate when simulating capture and processing.

usage is significant but it does not seem like it is the bottleneck, the disk caches become significantly big with more than 120GB, and the HDDs are exposed for heavy loads and write 474MBps (or 3.8Gbps), which is quite far from the goal of 10Gbps. To determine where the bottleneck actually exists, one must do further investigation. Therefore, one must determine how fast one can write data directly to RAM.

Some tests were conducted at 550MBps requested rate on the PCIe FPGA, but the results were inconsistent in terms of overflowing. Some of them were successful, where others cause overflow in kernel memory. This indicate that this is the very limit for what this test environment can capture.

5.2.2 Simulating Capture and Processing

The goal for this experiment is to determine the system throughput when processing is enabled. The reason for this is that many customers want their data as soon as possible, which require MEOS capture to perform capture and processing, concurrently. This will contribute to generating the new baseline with next generation servers.

To simulate capture and near real-time processing, Perfio was used to simulate the capturing process and Pipe Viewer (PV)[3] was used to simulate the processing. PV is a program that allow the user to move data from source to destination, while monitoring runtime statistics such as the momentary amount moved, and momentary rate. Data was written to the same RAID as the previous experiments, to test its performance when writing and reading simultaneously. At first, the experiment was launched with 128GB RAM available, but Dstat was not able to record any reading activity from any disks. The reason is that the reading process read data directly from the disk cache, instead of the physical disk. Therefore a big RAM-disk was set up and filled with data, so only 16GB RAM was available. After this change the reading process was recorded by Dstat, and runtime statistics were provided. By limiting the RAM available, the sustained rates to disk had to be decreased a bit.

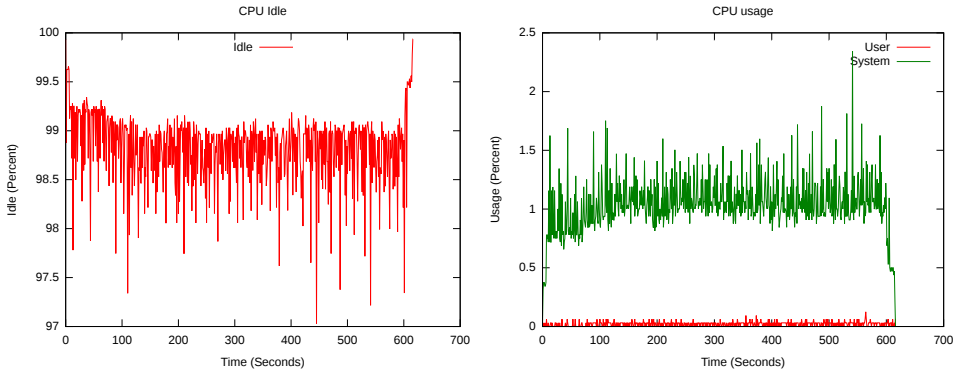
Requested rate	113 MBps
Actual max rate	109.38 MBps
Total data written	65, 544, 388, 608 bytes
Exact runtime	599.97 seconds
Pages written	16, 002, 173 pages
Page size	4096 bytes
Achieved rate	104.19MBps
Limited to	95.3 % of max rate

Table 5.4: Perfio output for testrun with the highest successful data rate, when simulating capture and processing.

However, with 16 GB RAM available, Perfio was launched with parameters as presented in table 5.3 and the resulting data are presented in table 5.4. As for the reading process, PV is implemented to allow the user to limit the rate at which data is read from disk. This was used within this experiment, and the reading rate was set to 105 MBps, to ensure that the reading process is always slower than Perfio. The reading process must be slower because PV terminates when it believes end of file is reached. The read data was written to another RAID, to simulate storage of processed data where the RAID performance is irrelevant. By this setup, the reading process will become increasingly behind Perfio, forcing the system to read from the HDDs at some point. PV reported an average reading rate of 100MBps.

As for the analysis, the CPU load is negligible as the graphs in figure 5.6 demonstrate. Because of the very low incoming data rate, and the fact that the system uses DMA, reduce the amount of work the CPU have to do, resulting in very low load. One may speculate that when performing actual level-0 processing as specified within KSPT websites[36] the CPU usage will be somewhat higher, as the CPU actually does some operations on the data. However, the memory usage is presented in figure 5.7. The memory usage are, as expected, very high. The disk caches quickly uses a lot of memory, indicating that the HDDs are under high load. As figure 5.8 illustrate, the HDDs does not handle concurrent reads and writes very well. This is caused by the seeking time needed to place the mechanical disk head to the right position before accessing a particular disk space. The graphs show that the traffic is very bursty, which is because of caching mechanisms in the file system. The results does however show that when performing read and write accesses to HDDs, the throughput becomes very low with 100 MBps, or 800 Mbps, reading rate as the bottleneck within the pipeline.

This is the absolute worst case scenario, as data does often exist within caching mechanisms in either the file system, RAID controller or disk caches. This



(a) Visualization of the time the CPU spends idle

(b) Visualization of scheduled runtime for user and system level processes.

Figure 5.6: Figure shows the amount of time the CPU spends idle, and the total amount of time user and system level processes actually run on the CPU. The CPU usage when simulating capture and processing are lower than when performing capture only.

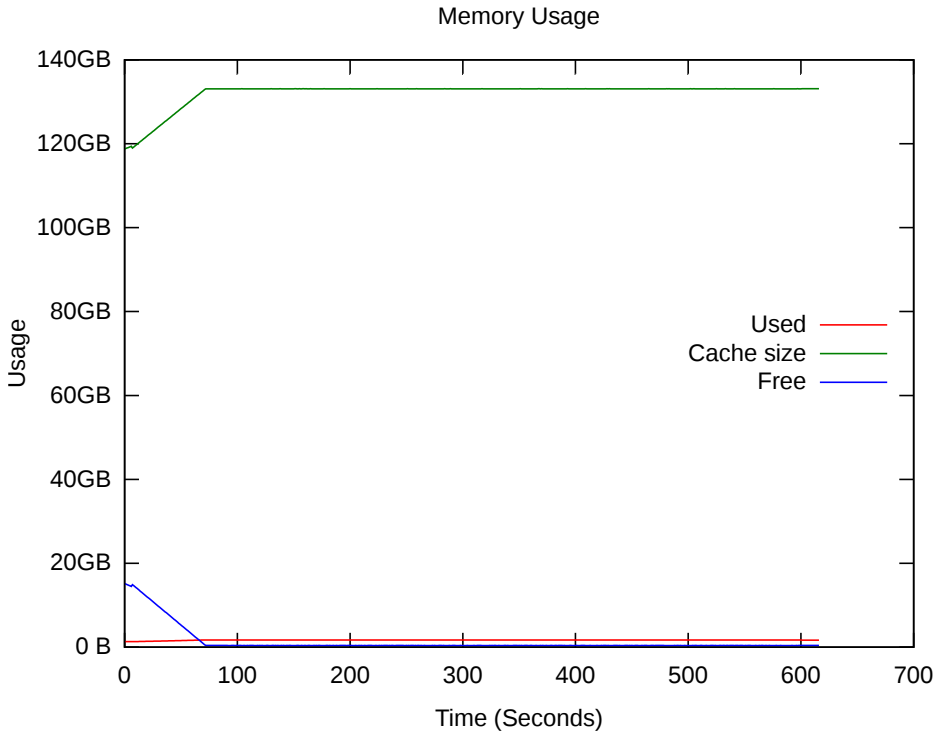
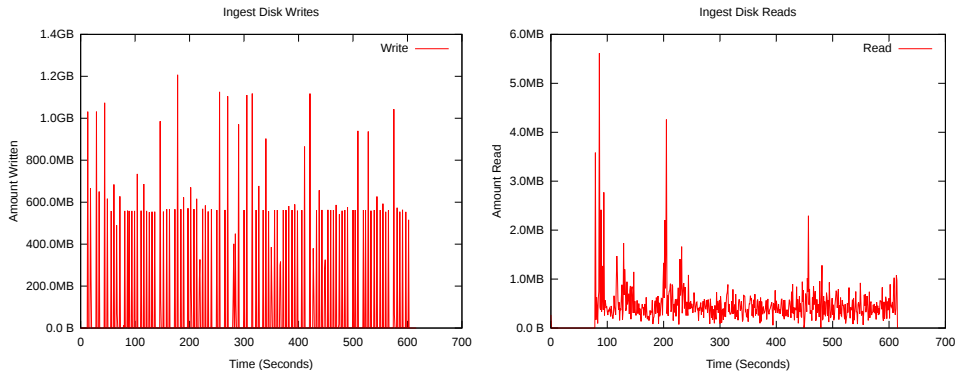


Figure 5.7: The graph illustrates the memory usage during simulation of capture and processing of data. Note that the disk caches start at about 112GB, which is a result of the RAM disk used to limit the memory available.



(a) Visualization of writes to the ingest disks.

(b) Visualization of data read from the ingest disks.

Figure 5.8: Figure illustrates the disk load generated on the ingest disks, configured in RAID level 1+0 during capture and processing. The samples are from the experiment conducted with the highest successful capturing rate of 104.19 MBPS and reading data at a rate of 100MBPS.

means that the reading process rarely has to read data from the physical disk anyway, and this was therefore tested as well.

However, the experiment presented in this section was conducted with 16 GB RAM available. The same experiment was conducted with 128GB RAM available as well. The achieved capturing rate was 148.82MBPS, while PV reported 130MBPS average rate. The only difference between the experiments is the amount of RAM available, which indicate that the host memory does affect the capturing ability of the system. This is just above what KSPT guarantee today, according to their data sheet[36], which is 125MBPS with near real-time processing enabled.

We can see that with processing, the test server is able to ingest 148.82MBPS while processing is enabled. This is equal to 1.2Gbps per input channel. When using two input channels, the rate can in theory be doubled. The system should be capable of this, as the previous experiment simulating capture only demonstrate almost 4Gbps per channel. The result demonstrated in this experiment indicate that state of the art can be achieved with next level servers, without changing software.

5.3 Determining the Bottlenecks

To determine where the bottleneck lie within this system, further examination is required. The experiments above suggest that the HDDs can be one bottleneck, so more experiments must be conducted to reveal any other possibilities. In order to do so, one must have a look at the pipeline of components the incoming data pass through and test the performance between each component. The data path is from the data receiving boards FPGA to RAM via the DMAC. From RAM the data is written to HDDs via the RAID-controller. Therefore, two additional experiments should be enough to determine whether the HDDs are the bottleneck or not; Write data directly to RAM, and next determine the supported rate between RAM and HDDs.

5.3.1 From Data Receiver to Memory Performance

By isolating parts of the production pipeline, one can determine the throughput between components. With such numbers available, one achieves a much more detailed picture of the practical throughput. This experiments purpose is to determine the maximum throughput one can achieve between the data receiving board and host memory. The cooperation between the receiving board, PCIe bus and host memory as described in figure 5.9 is tested through this experiment.

To accomplish this experiment, a chunk of memory in RAM was allocated and exposed as a RAM disk. By using such a RAM disk, one can create files and use that disk as one use disks traditionally. One can also use it to capture simulated satellite data, which is this experiments intention. One exposes a chunk of memory as a logical disk by adding a line in the file `/etc/fstab` which initializes all mounting points added in the file on startup. Since the test environment are equipped with 128GB of RAM, 120 GB was allocated for storage within this experiment. Adding the line in listing 5.3 to the file `/etc/fstab` enables this feature.

Listing 5.3: Command used in fstab to set up temporary file system in computer memory, exposed as a logical harddisk.

```
tmpfs <mountpoint> tmpfs size=120G
```

where `<mountpoint>` is the directory the memory area is mounted to. In this case, the RAM disk is mounted to a directory named `disk_ram`. It is a problem monitoring reads and writes to such mounting points because Dstat require users to specify which physical device to monitor through the directories within the `/dev/` directory. This means that the RAM disk stats as presented above can not be provided in the same manner through this experiment as the previous

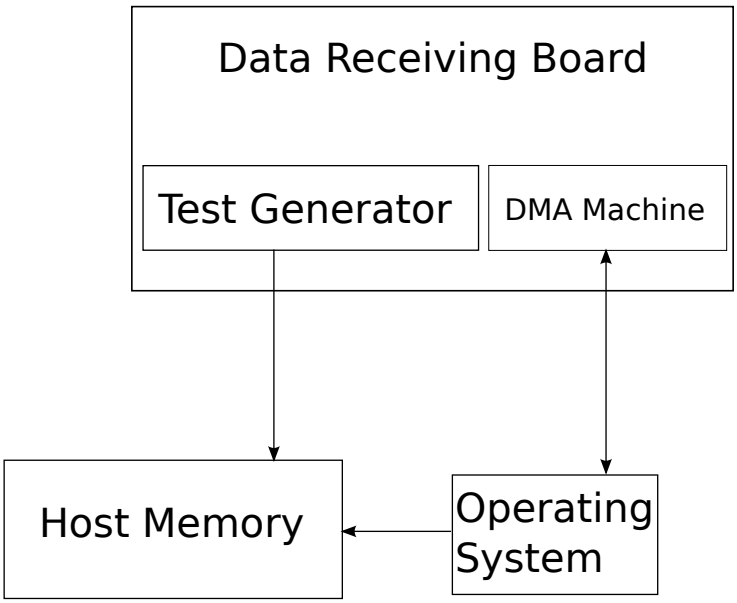


Figure 5.9: Figure illustrates the data path from receiving board to host memory, which is the components tested through this experiment.

Requested rate	2000 MBps
Actual max rate	2000 MBps
Total data written	125, 992, 697, 856 bytes
Exact runtime	109.98 seconds
Pages written	30, 760, 429 pages
Page size	4096 bytes
Achieved rate	1092.47MBps
Limited to	54.6 % of max rate

Table 5.5: Perfio output for test run when writing directly to memory when requesting the FPGAs max rate.

ones. The RAM disk activity can however be monitored through the memory statistics provided by Dstat, meaning one will achieve some level of information anyway. The important stats from this experiment is the practical transfer rate between the data receiver and host memory, in addition to the CPU load. Because the computer is not equipped with enough memory to temporarily store the whole dataset of 283.6 GB, this experiment duration can not be as long as 600 seconds, as in previous experiments. The experiment duration will vary through this experiment, because of the limitations in disk size, starting at 60 seconds. The experiment duration was incremented until the generated dataset was nearly equal to the disk of 120GB.

To test the PCIe bus and memory capabilities, the rate was set to the FPGA absolute max rate of 2000 MBps. The result are presented in table 5.5. One would expect the actual average rate to come much closer to the requested rate of 2000MBps than it does, which the PCIe bus theoretically should support. This occurs because the DMAC is not implemented with prefetching of page descriptors from the host memory, which limits the system when reaching such rates. With prefetching, one can store the page descriptors locally on the data receiving board before the execution begin. Such requests are expensive and limit throughput. If pre fetching of page descriptors is implemented, the performance is expected to increase. The interesting aspect of this results, are that the transfer rate between the data receiving board and host memory are much better than between host memory and the ingest disks. This gives reason to believe that the ingest disks are one bottleneck within the test environment.

Even though the demonstrated max rate for transferring data between the data receiving board and host memory is 8.73Gbps, some improvements can be done to achieve better rates if one implements prefetching of page descriptors on the DMAC. The rate demonstrated in this experiment does not fulfill the requirement of 10Gbps either, and it is speculated that one has two possibilities

in order to achieve the result. The first alternative is to move the data receiving board from PCIe version 2.0 and over to a newer version of the bus with better theoretical throughput, as the server support version 3.0. This is also the only alternative to achieve 10Gbps per input channel. The other alternative, is to rely upon two input channels, and hopefully achieve 10Gbps in total, which seems to be possible as each input channel are placed on separate PCIe buses, and hence can achieve 8.73Gbps per channel to host memory. This must however be tested in order to confirm.

5.3.2 From Memory to Disk

The purpose of this experiment is to determine the actual throughput between host memory and the ingest disks. Figure 5.10 shows the different components that this test exploits, as well as how they are connected. This experiment is a big part of gaining a detailed picture of the systems throughput with next generation servers. To test the transmission speed between memory and the ingest disks PV is used. The program monitors the progress of data through a pipe, and can be intentionally used to transfer data from one file to another. By using the file generated in the previous experiment, where data was transferred from the data receiving board to host memory one can transfer the exact same data from host memory to the ingest disks while the process is monitored. When launching PV without specifying any data rates, it moves the data as fast as possible without any limitations in software. When PV has terminated, the final statistics from its execution is presented. The final statistics are total amount of data moved, the total time spent and average transfer rate. For this experiment, PV was invoked as shown in listing 5.4.

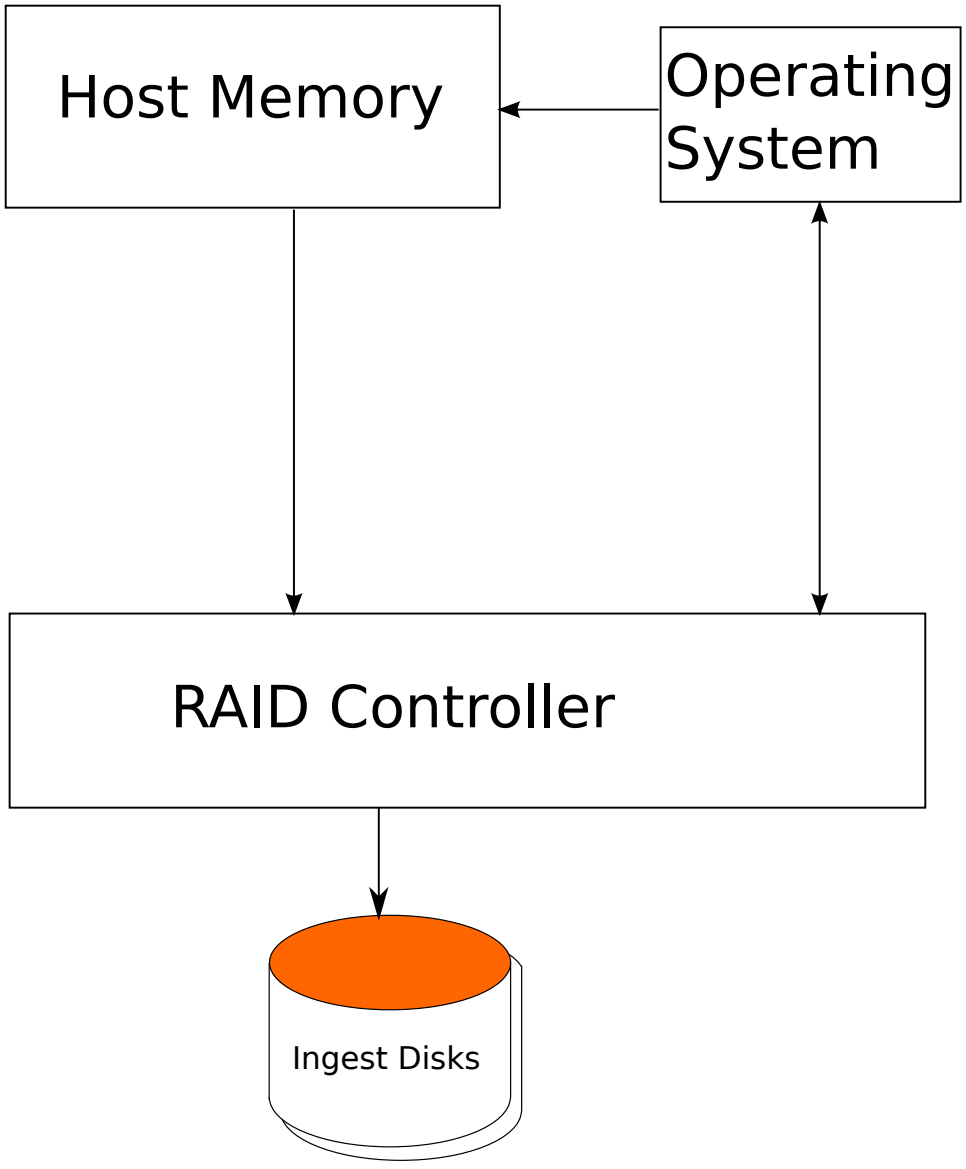


Figure 5.10: Figure illustrates the data path from host memory to ingest disks, and the components in action through this experiment.

Listing 5.4: Script used to move data from the disk in memory to the configured ingest disks. Deleting the file makes sure that one writes to an empty disk.

```
#!/bin/bash

iterations="$1"
i=0

while [ $i -lt $iterations ]
do
        #Copy data from disk_ram to ingest disks
        pv /disk_ram/test > /raid10/testfile

        #Delete the file, need empty disks.
        rm /raid10/testfile

        #Pv require existing files.
        touch /raid10/testfile

        #Increment sequence counter.
        ((i++))
done
```

This experiment is executed 20 times to eliminate the impact of high and low values in disk activity, often caused by the file system caching mechanisms. The results are presented in figure 5.11. The output from PV report the maximum average transfer rate of 479MB/s and the minimum to be 464MB/s, which is just below the 474MB/s reported by Perfio presented in table 5.2. This seems to be legitimate results as the HDDs most likely cannot write data faster, which also means that this is the maximum rates that the test system used can capture data.

5.4 Summary

Through the experiments conducted on the test server within the test environment presented in section 5.1, it is demonstrated that the maximum supported data rate is 474.43MBps (or 3.8Gbps), when capturing a satellite pass without any processing involved. It is demonstrated that the computer is exposed to very little load, except for memory usage and the HDDs. This is further confirmed by the experiment where the pass was stored in memory, achieving rates above 1000MBps. Further experiments show that the ingest disks used do not support rates above 480MBps, which is not sufficient when the future

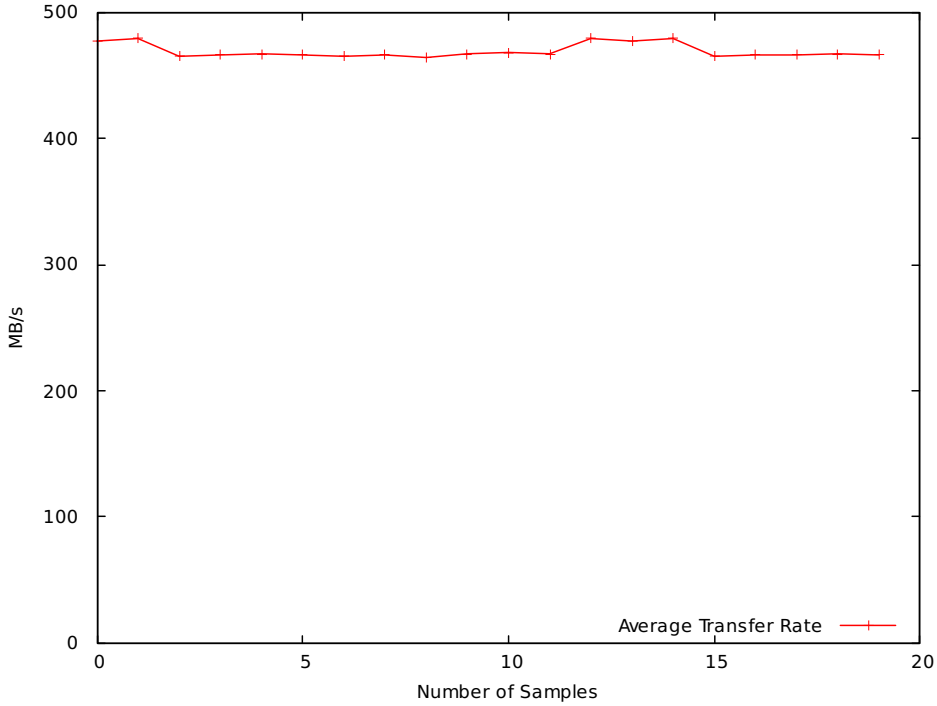


Figure 5.11: Visualization of average transfer rate between host memory and ingest disks with RAID 1+0 over 20 experiments.

required rates will exceed 10 Gbps. This leads to a conclusion that the ingest disks are the bottleneck within the test system.

6

Resolving the Bottlenecks

This chapter will, based upon the results presented in chapter 5, describe how MEOS capture can improve the data throughput. The redesign mainly concern different RAID configurations to achieve better throughput, as the bottleneck was the ingest disks. All changes will be tested and evaluated, in isolation in order to determine whether it was an improvement or not.

6.1 Disk Configurations

In chapter 5, the ingest disks were revealed as the bottleneck within the test environment. Therefore the first changes will be done based on this experience. MEOS capture has until now been using RAID level 1+0, meaning 50% redundancy of data, which is useful for long term storage of valuable data. During a satellite pass, the desire for capturing lossless data is considered more important than keeping the data consistent in a long term perspective. One must make sure that the data is lossless as it is captured, but in a long term perspective, a capturing system does not offer long term storage. Redundancy in data storage generates overhead as auxiliary data is created. As the overhead increases, the performance must necessarily decrease. Therefore this chapter will mainly focus on how KSPT can configure their ingest disks in order to achieve better throughput, both focusing on capture only and with near real-time processing enabled.

The remaining of this chapter will focus on determining the maximum throughput with new configurations for the ingest disks when performing data capture, and data capture with near real-time processing enabled. The RAIDs tested are striping[1] and RAID level 5. Both setups will be tested with both 4 and 8 physical HDDs participating in the RAID.

6.1.1 Ingest Disks With No Redundancy

The goal of the experiment is to determine how much data the test system can theoretically capture with a new setup for the ingest disks. In order to determine how much the RAID level limits the system throughput, this experiment only changes the ingest disk RAID level. Therefore, the four HDDs were striped, because this provides the best writing performance according to Peter M. Chen et. al.[9].

One must be aware that when using striping, one does not have any form for redundancy of data. When a disk participating in a striped RAID fail, the whole array of data is broken with no possibility for recovery. For a system such as MEOS capture, the level of redundancy required is as much a political decision as it is a question of the desired performance. Disks do fail, and according to Murphy's law, everything that can go wrong will go wrong[15]. By using two servers that is mirrored[2], one can tolerate disk failures during a satellite pass, even though the ingest disks are striped. This will result in a much more expensive system, because it requires two servers instead of one. Some customers do however require capturing redundancy through two servers.

The striped RAID will be tested through two experiments. The first experiment tests data capture. The second experiment tests both data capture and near real-time processing.

Capture The baseline created in section 5.2.1 reported a maximum capture rate of 474,43MBps when performing capture only. The experiment started with a requested rate of 540MBps. The requested rate was increased in steps of 10MB until the system lost data due to overflowing buffers. The experiment results are presented in table 6.1.

The highest rate achieved is 931.12MBps (or 7.45Gbps) which is almost twice as much as the baseline. The reason for this is that instead of striping the data across two mirrored RAIDs, one stripes the data across four. One can, through

1. RAID 0
2. Mirroring — the servers contain the exact same data

Requested rate	1285 MBps
Actual max rate	1281.25 MBps
Total data written	585, 252, 208, 640 bytes
Exact runtime	599.43 seconds
Pages written	142, 886, 301 pages
Page size	4096 bytes
Achieved rate	931.12MBps
Limited to	72.7 % of max rate

Table 6.1: Perfio output for experiment with highest successful rate, with striped ingest disks.

this configuration, write data to twice as many disks which gives a significant gain in writing performance.

As the result from Perfio demonstrate, the total throughput has almost doubled itself with the changes made. The ingest disk activity is presented in figure 6.1. The graph shows that the ingest disks were exposed to high load, as the drops in writing are limited to just above 800MBps as the lowest value. The large drop at 620 seconds indicates that Perfio terminated, and the following writes indicate that the file system synchronizes its caches with the disks.

The memory usage is presented in figure 6.2b. The usage is as expected very similar to previous experiments, where disk caches use the largest amount of memory. One interesting observation is that the disk cache has been filled to max after 150 seconds of execution in this experiment, where the previous experiments spent just over 300 seconds to fill up the caches. This is visualized in figure 6.2. However, this may be a natural behaviour as the rate in the second experiment is doubled.

The CPU has not been exposed to any significant loads during the experiment, as figure 6.4 show. This demonstrates that the implementation of DMA is working as intended. This is further supported by the average load, as figure 6.3 illustrates. As the average load never show more than 2.2 processes queued, even for the last minute, the CPU is not exposed to heavy load before it passes about 25 due to the 32 cores available.

To conclude, the system seems to cope well with rates just below 7.5Gbps for one input channel, where the memory usage is, as expected, significant and the ingest disks are exposed to heavy load. The increase in performance is as expected nearly doubled. It is expected because one can now perform write accesses to all four disks, instead of just two as with RAID level 1+0. In the experiment presented in section 5.3.1 when testing the throughput between

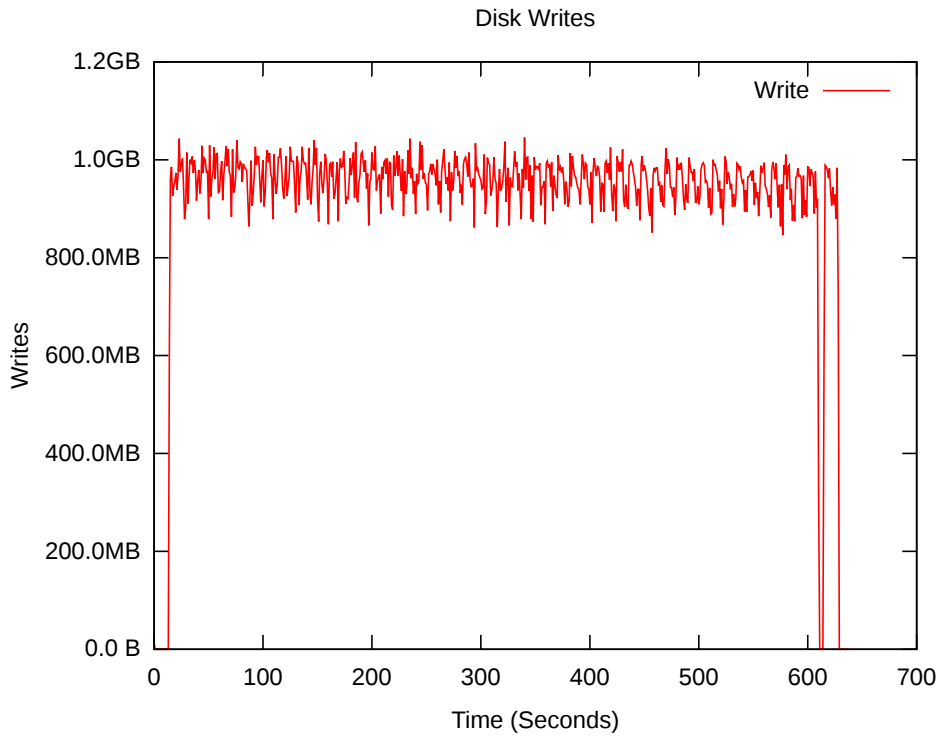
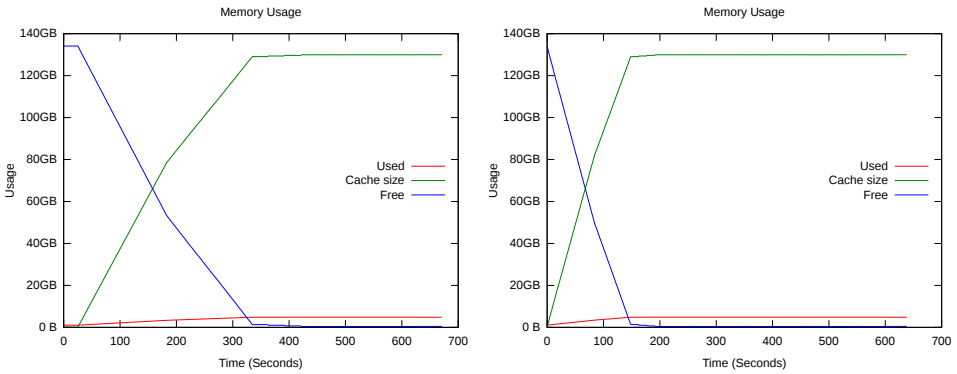


Figure 6.1: Figure visualizes the disk behaviour when four HDDs are striped and used as ingest disks.



(a) Memory usage with ingest disks configured to RAID 1+0.

(b) Memory usage with four disks striped, used as ingest disks.

Figure 6.2: Figure shows the difference in memory usage with two different configurations for the ingest disks.

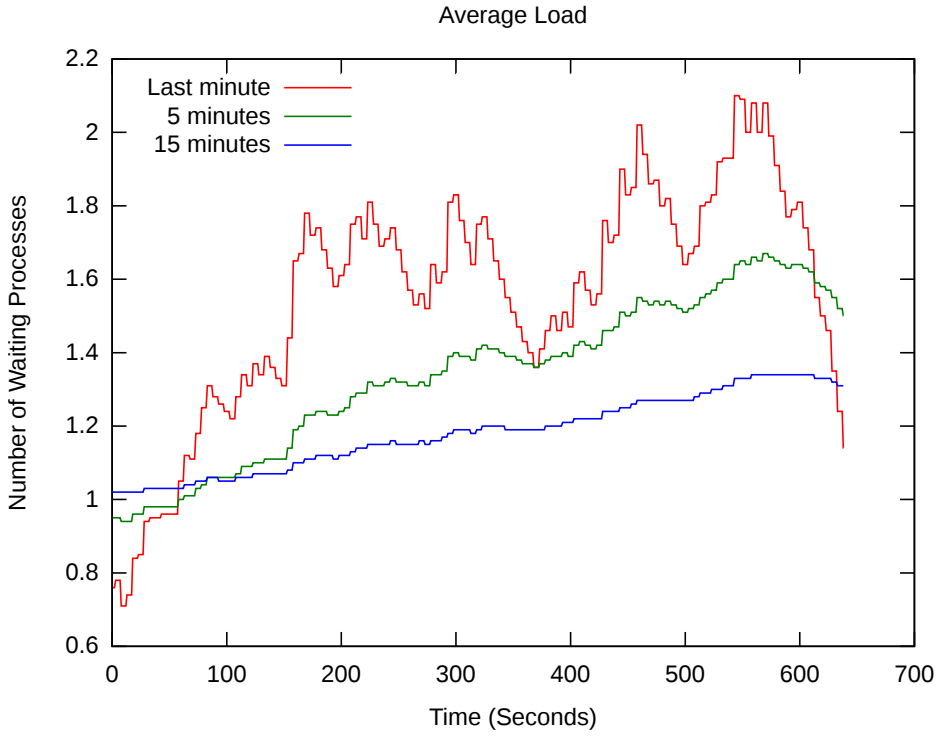
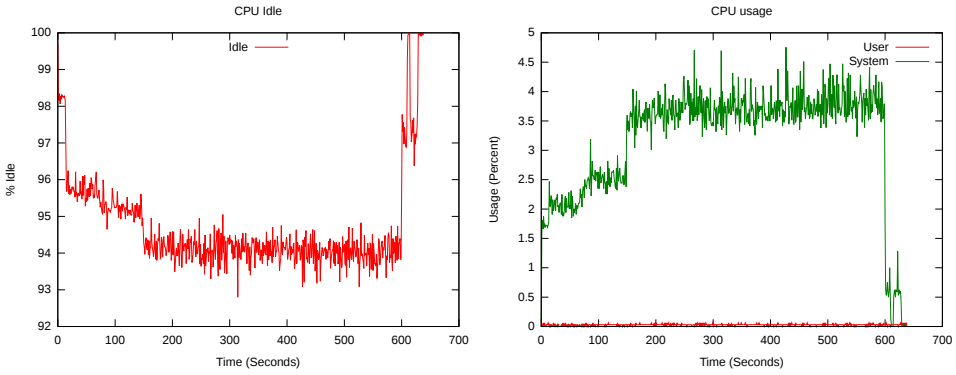


Figure 6.3: Figure visualizes the average load when executing Perfio to four striped HDDs as ingest disks.



(a) The figure illustrates the amount of time the CPU spends idle, when Perfio executed to striped ingest disks.

(b) Visualisation of scheduled runtime for user and system level processes, with striped ingest disks.

Figure 6.4: Figure shows the amount of time the CPU spends idle, and the total amount of time user and system level processes actually run on the CPU. The figure suggests that the CPU is not a bottleneck with rates above 7Gbps.

Requested rate	300MBps
Actual max rate	296.88MBps
Total data written	176, 886, 382, 592 bytes
Exact runtime	599.64 seconds
Pages written	43, 185, 462 pages
Page size	4096 bytes
Achieved rate	281.32MBps
Limited to	94.8 % of max rate

Table 6.2: Perfio output for experiment with highest successful rate, with four striped ingest disks.

the data receiving board and host memory, prove supported rates above 8Gbps. This indicates that the ingest disks are still the bottleneck when the near real-time processing is disabled.

Capture and Processing As KSPT offer their customers with the possibility of near real-time processing, the goal for this experiment is to determine whether striping is a better alternative than RAID 1+0 when simulating both capture and processing.

Let's recall from the previous experiment in section 5.2.2, where the ingest disks were configured to RAID 1+0, and the demonstrated capturing rate was reported to 104.19MBps as the worst case scenario, when reading with 100MBps for processing purposes. When 128GB RAM is available the reading process is allowed to fetch data from disk caches, delaying the read accesses to physical disks as much as possible, the demonstrated ingest rate was 148.82MBps. One would expect this disk configuration to perform better, due to the number of disks that the data is spread across without redundancy. This is the baseline for the next generation servers, and therefore this experiment started out with rate of 150MBps, and increased by 10MBps as long as data is not lost.

This experiment was conducted with all RAM available, in order to test the system as close to the original setup as possible. This resulted in a maximum ingest rate of 281.32MBps while reading rate was reported to be 261MBps. The full output from Perfio is presented in table 6.2.

As one can see, the FPGA has generated 164.74GB over 599.64 seconds. The FPGA has not been limited like it is when the rates become very high. There is no doubt that the HDDs are still the bottleneck, as the system has proven itself to cope with rates above 8Gbps per channel. This is further confirmed by the average load presented in figure 6.5, stating that not many processes are running on the server during the experiment. The figure shows that the

average load is almost negligible the last 15 minutes as 0.7 processes waited in queue to run on the CPU. The 1 minute mark show a peak just above 1.8 queued processes, which is still very low since this number can exceed 25, due to the 32 cores available.

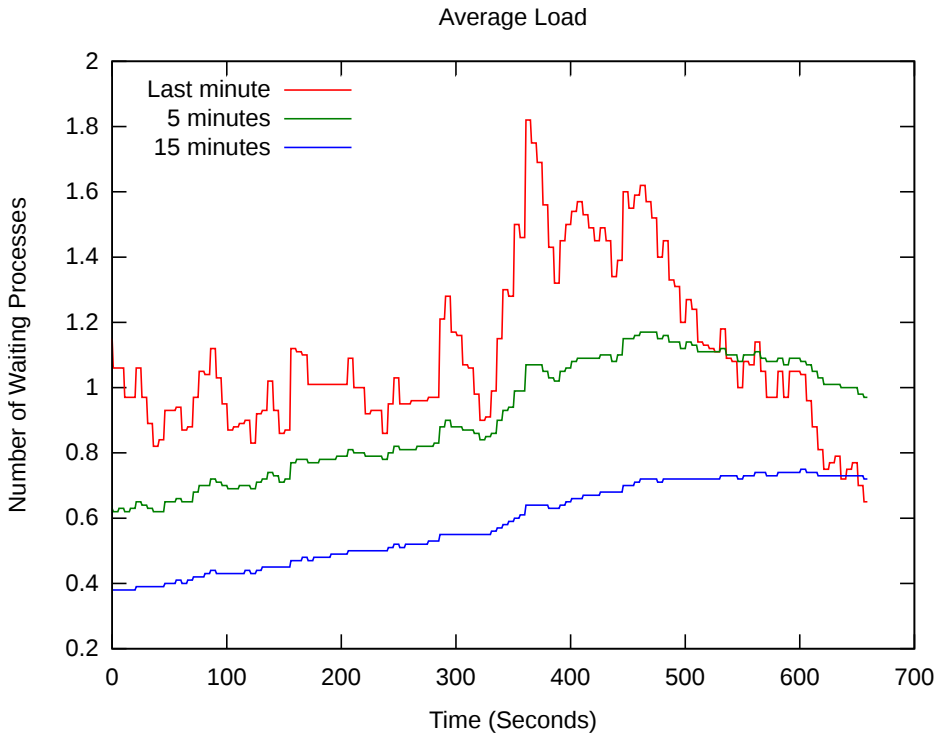
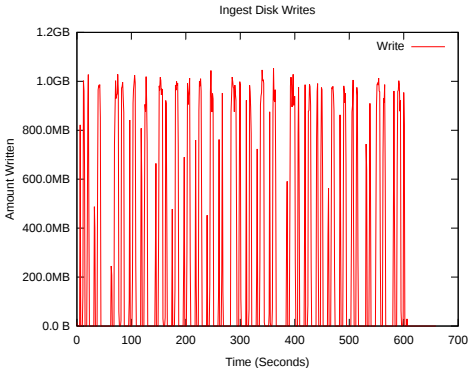


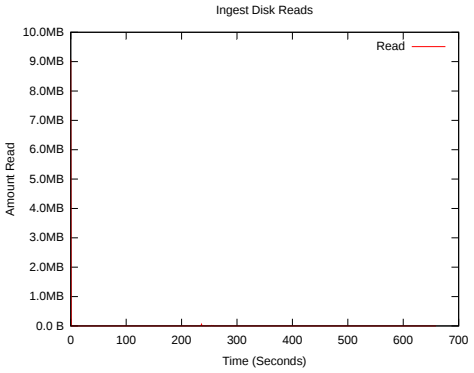
Figure 6.5: Figure visualizes the average load with four HDDs as ingest disks, and performing near real-time processing concurrently.

The disk activity presented in figure 6.6, one can see that the disks are exposed to very bursty load. The reason for the spikes is the caching mechanisms. It is also observed that as the rate increases, the disks have to spend more time writing data, resulting in very stable behaviour without the large drops or high spikes as shown in figure 6.1. This indicates that the disks are capable of writing much more data per second, as demonstrated in section 6.1.1, paragraph `Capture`. The rates are expected to drop as processing is enabled anyway because of the two-way traffic through the RAID controller, and internal buses.

The disks used to store the processed data are configured as a striped RAID of 8 disks, capable of much more throughput than the ingest disks are capable of. This was done to in order to eliminate the processed storage as a bottleneck. The load generated to processed storage is therefore negligible, as shown in



(a) Figure visualizes the average load with four HDDs as ingest disks, and performing near real-time processing concurrently.



(b) Visualization of reading activity when performing data ingest and processing with four striped HDDs.

Figure 6.6: Figure shows the disk activity, when using four HDDs striped as ingest disks. Note that the system have only accessed the physical disk once, after 240 seconds, without causing any overflow.

figure 6.7.

As for the memory usage, presented in figure 6.8, we observe some thrash values at the very beginning of the experiment. As stated in the beginning of chapter 5, we will ignore these values. Now, the memory usage peaks just around 200 seconds into the experiment, which is fairly late. From this point and out, it is a fair risk that one has to read data from the physical disks instead of disk caches. The high memory usage seems to be consistent from 200 seconds and to the end of the experiment.

Summary The results summary are presented in table 6.3, along with the results generated with ingest disks configured to RAID level 1+0, in order to easily spot the improvement. However, based on the results generated in section 5.3.1, we know that the system is able to cope with rates above 8Gbps, or 1000MBps, per input channel. These experiments show that while only performing data ingest, the rates are getting much closer to what one input channel can handle. When performing data capture and near real-time processing the rate was almost doubled.

Because it is proven that the system can achieve up to 8.7Gbps from the data receiving board to host memory, the next step is to increase the number of physical HDDs in the ingest RAID, and hopefully demonstrate better rate than done so far. The system seems to have the largest potential for improvement when performing data ingest and near real-time processing, according to the

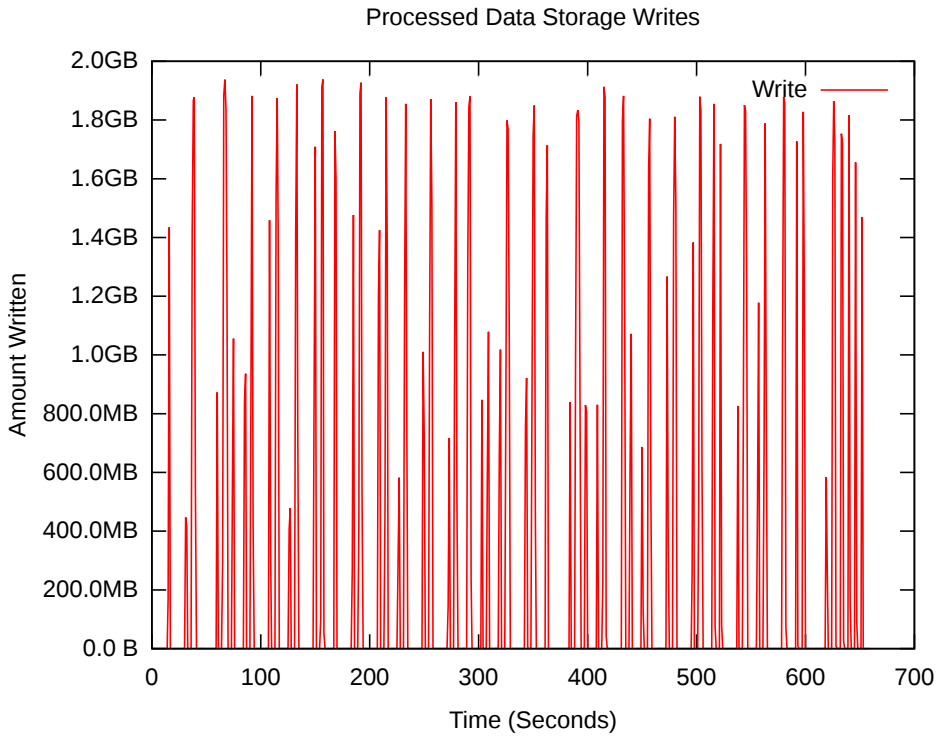


Figure 6.7: Illustration of the disks used to store processed data. The RAID consist of 8 HDDs, striped.

RAID configuration	Input Channels	Ingest only (MBPS)	Ingest and processing	
			Ingest rate (MBPS)	Read rate (MBPS)
4X HDDS RAID 1+0	1	474.43	148.82	130
4X HDDS striped	1	931.12	281.32	261

Table 6.3: Summary of test results and improvement when changing disk configuration from RAID level 1+0 to striping.

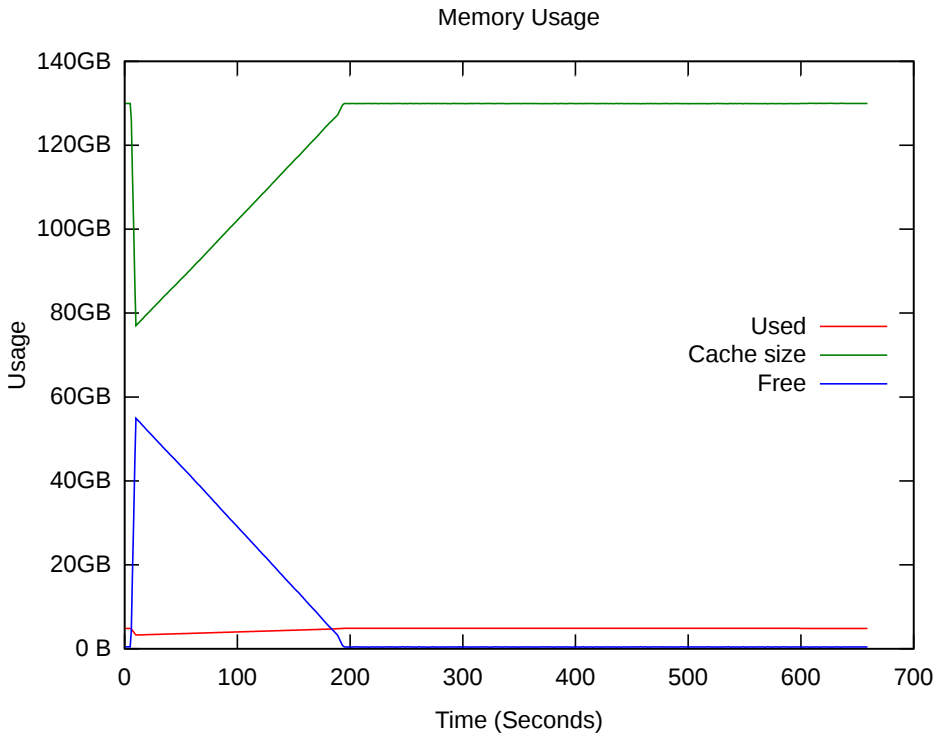


Figure 6.8: Illustration of the memory usage when using four disks as ingest disks. Note that the usage peaks after 200 seconds of experiment runtime.

Requested rate	1547MBps
Actual max rate	1546.88MBps
Total data written	662, 762, 946, 560 bytes
Exact runtime	599.60 seconds
Pages written	161, 808, 402 pages
Page size	4096 bytes
Achieved rate	1054.13MBps
Limited to	68.1 % of max rate

Table 6.4: Perfio output for experiment with highest successful rate, with 8 striped ingest disks.

results so far.

6.1.2 Increasing the Number of Disks

This experiment will increase the number of disks used in the ingest RAID from 4 to 8 disks. The reason for this is to find out whether the RAID scales linearly, and to see if one can achieve more than 1000MBps (or 8Gbps) from host memory to disk. The theory is that one should get higher rates as the data is spread across more disks.

As with the previous experiment, the RAID configuration will be tested for two schemes. The first will test the ability to only capture data, while the second will test the ability to both capture data and perform near real-time processing.

Capture This experiment used 8 HDDs striped, with the purpose of capturing data without processing. The experiments were executed over 600 seconds per experiment, with rates starting at 1000MBps since 4 HDDs handled 931.12MBps, shown in table 6.3. The system did not show any sign of significant load until the rates exceeded 1040MBps, where the highest successful rate was reported to be 1054.13MBps. The full Perfio output is presented in table 6.4.

From the table one can see that the achieved rate is becoming very close to what was demonstrated in table 5.5, which reported 1092.47MBps from the data receiving board to host memory. The total amount of data created in this experiment is 617.25GB, and the test generator has been limited to 68.1%. The ingest disk behaviour suggests that the disks are capable of higher rates, so for the first time the experiment results suggest that the ingest disks are not the bottleneck anymore. The disk activity is presented in figure 6.9. This means that the optimal number of disks is somewhere between 4 and 8 disks when

performing ingest only, for this specific test environment. The bottleneck now may be the same as discussed in section 5.3.1, which discussed prefetching of page descriptors implemented on the FPGA with interface to the PCIe bus. The reason that this is speculated, is because the transfer rate between the data receiving board and host memory was 8.7Gbps. The rate achieved in this experiment is 8.4Gbps, which means we are getting very close to what the system can manage without replacing the PCIe bus with a newer version. One could eliminate this by increasing the buffer size in kernel memory from 36MB to 64MB or 128MB, but this is speculation which require testing in order to have scientific proof.

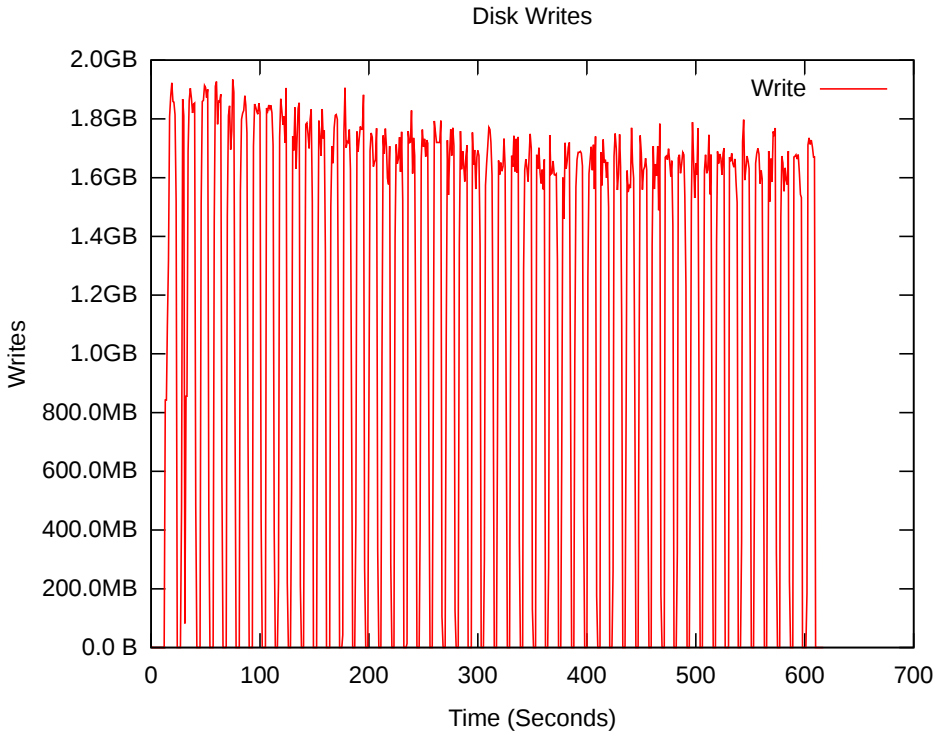


Figure 6.9: Visualization of disk activity when using 8 striped HDDs as ingest disks.

Regarding the memory usage, the usage peaks just after 120 seconds into execution, which is visualized in figure 6.10. This is expected behaviour, based upon the previous experiments. We have seen in earlier experiments that the memory usage peaks just after 100 seconds where the execution was successful, but at lower rates.

As for the average load during the experiment, it is as expected, the highest recorded so far. That is, without being even close to heavy load for the CPUs in general. The average load is visualized in figure 6.11. As the figure suggests,

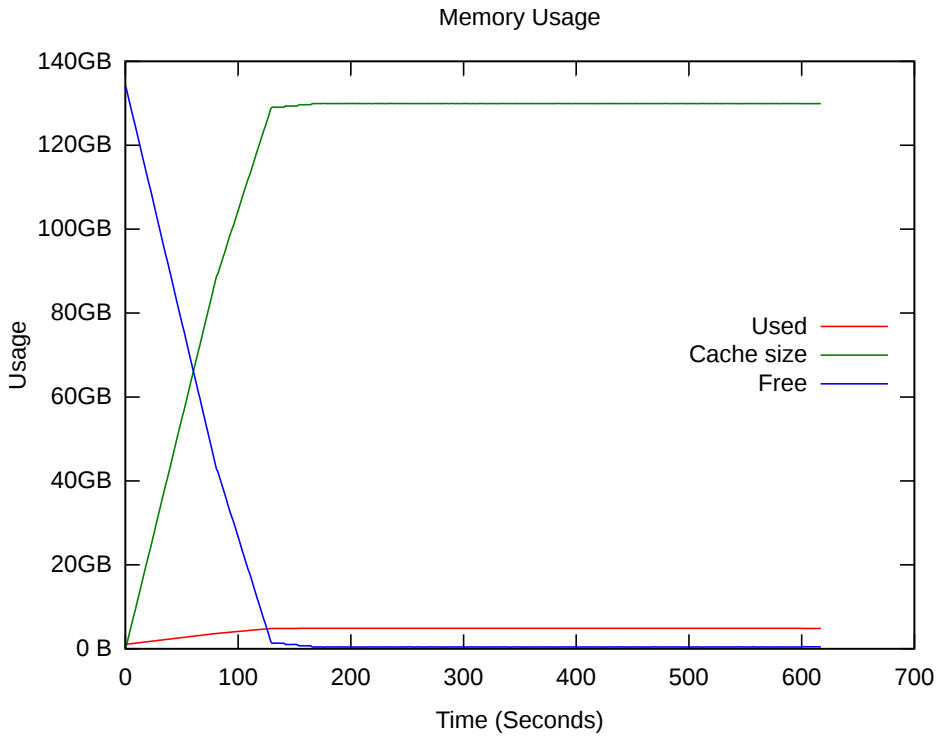


Figure 6.10: Visualization of memory usage when using 8 HDDs, striped, as ingest disks.

the average load the last 15 minutes, where 10 of them are the experiment execution time, touches just above 1 waiting process at average. The last minute show that on the most about 2 processes waited to execute, where the last 5 minutes peaks at 1.4 waiting processes. This average load is negligible, and the server can allocate its resources to perform data capture.

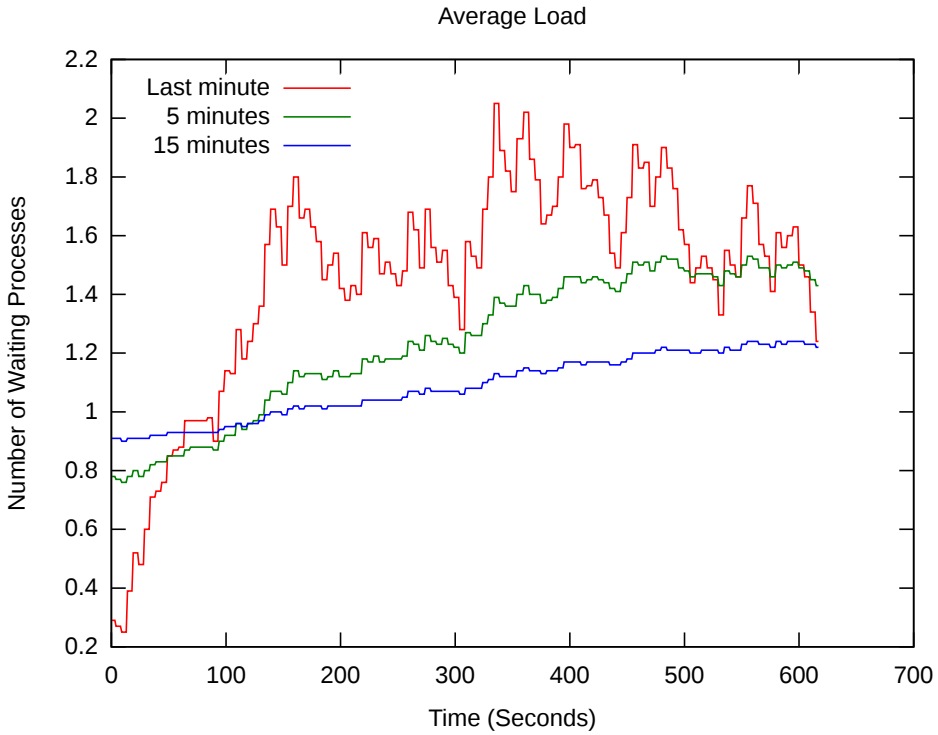
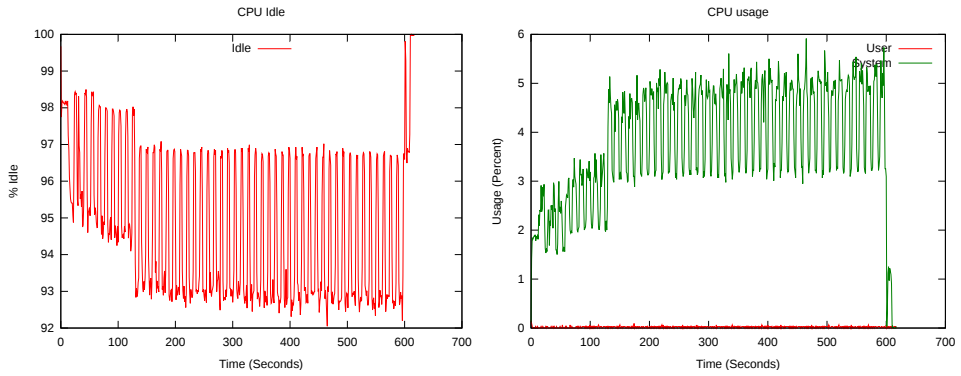


Figure 6.11: Visualization of average load when using 8 HDDs, striped, as ingest disks.

The CPUs' activity are presented in figure 6.12. As the figure shows, the CPUs still idle a lot, which is because of the DMA implemented. We do, however, see an increased CPU utilization compared to other experiments, where the system level processes peaks just under 6% utilization. Still, even though this program is multithreaded, the total CPU utilization is very low.

To conclude this experiment, we have seen the highest throughput so far in this thesis. The ingest disks are proven not to be the bottleneck anymore when using only one input channel. The total system seems to be very comfortable with such high rates, as the CPUs are exposed to low load caused by the use of DMA. The memory usage is as expected high which is expected. This suggests that one can achieve twice the speed when using two input channels. In real world, it is not so easy. When running over 8Gbps per channel to host memory,



(a) Figure visualizes the amount of time the CPU spends idle during the experiment execution.

(b) Visualisation of the amount of resources used by user and system level processes during the experiment..

Figure 6.12: Figure shows the general load on the CPU during experiment execution.

we know the total incoming rate is above 16Gbps. The theoretical max rate supported by the SAS interface, which connects the RAID controller to the ingest RAID only support 12Gbps theoretical maximum. In order to achieve such rates, the SAS bus must be replaced with a PCIe bus version 3, or newer using more than 4 lanes. This also requires the use of new disk technology, and the only disks directly connected to PCIe bus so far is NVMe disks. HP claims on their websites that their write intensive workload accelerators[24] is capable of writing 1700MBps (13,6Gbps) sequentially, and it provides 800GB of storage capacity. This would result in a much more expensive system, but one can achieve rates up to 10Gbps more than any other system found.

Capture and process Using 8 HDDs striped has proven itself to perform better than using 4 HDDs. This experiment will explore how well such RAID performs with concurrently read accesses and write accesses, by performing data ingest as well as simulating processing. This experiment will explore the scalability for 8 disks in a striped RAID when performing ingest and processing simultaneously.

The results are presented in table 6.5. The ingest rate is reported to be 259.60MBps, while the reading process reported 235MBps. This resulted in a total of 151.91GB generated over 599.23 seconds. From this, it seems like the scalability stalled somewhere between 4 and 8 disks participating in the striped RAID for this specific environment. The server load is negligible, due to the very low rates demonstrated. The results shows that striping 8 HDDs is a very bad idea, when performing both data ingest and near real-time processing. This may be caused by the capabilities within the RAID controller, because it must manage in- and outgoing data for 8 disks instead of 4. Because Perfio

Requested rate	275MBps
Actual max rate	273.44MBps
Total data written	163, 116, 482, 560 bytes
Exact runtime	599.23 seconds
Pages written	39, 823, 661 pages
Page size	4096 bytes
Achieved rate	259.60MBps
Limited to	94.9 % of max rate

Table 6.5: Perfio output for experiment with highest successful rate, with 8 striped ingest disks.

rely upon the OS, file system and disk drivers and file system to handle all data traffic to disks, these factors may also be the reason why this setup does not provide good results. It is however a bit odd if the limitation is within the OS or file system because the RAID is abstracted away from the two. The file system only knows about one logical disk, without being in possession of any information whether this is a RAID or one single, large, physical disk. This is all administrated by the RAID controller. One can however never be sure without further investigation to where the bottleneck lies within this setup.

With the analysis tool used, one cannot be sure where the bottleneck occurs as the RAID controller is not monitored. However, based on previous results and experiences, the RAID controller seems like a plausible explanation to why this setup performs so poorly when doing both data capture and near real-time processing.

The results have been analyzed, and the memory usage seems to peak after just over 300 seconds, which is the latest peak recorded. The increase in memory usage seems to be close to linear for all experiments so far. This is shown in figure 6.13. It is also worth mentioning that there was not recorded one single read access to the ingest disks, which means all data has been read from some temporary storage, or disk caches, through the whole experiment. This is shown in figure 6.14

Because the CPU statistics, average load and disk activity follow the same patterns as presented in section 5.2.1 the graphs for this experiment are presented in appendix A.1.

Summary To sum the results up so far, using 8 HDDs, striped, performs very well when only performing data capture demonstrating the highest rate so far at 1054.13MBps. It seems like the RAID controller or SAS controller become a bottleneck when performing data capture and near real-time processing, as

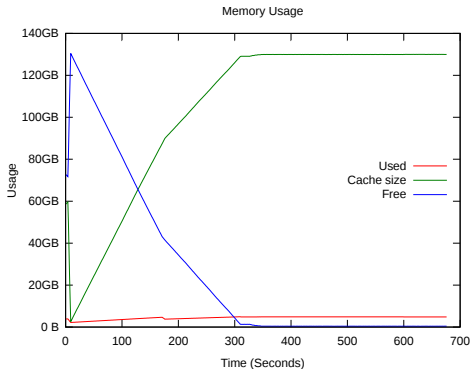


Figure 6.13: The figure shows the memory usage when performing data capture and near real-time processing with 8 HDDs striped as ingest disks.

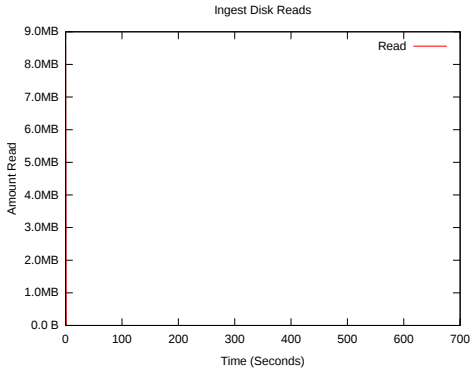


Figure 6.14: The figure shows that not a single read access was performed to the physical RAID used as ingest disks.

RAID configuration	Input Channels	Ingest only (MBps)	Ingest and processing	
			Ingest rate (MBps)	Read rate (MBps)
4x HDDs RAID 1+0	1	474.43	148.82	130
4x HDDs striped	1	931.12	281.32	261
8x HDDs striped	1	1054.13	259.60	235

Table 6.6: Summary of test results and improvement when changing disk configuration from RAID level 1+0 to striping. Note that 8 striped HDDs was outperformed by 4 striped HDDs.

the configuration achieved lower rates than when using 4 HDDs striped. All results so far are presented in table 6.6.

6.1.3 Introducing Redundancy With RAID 5

The previous experiments demonstrate great increase in throughput when using striping compared to RAID level 1+0. Striping is, however, more of a risk than RAID 5, as the whole array is broken and unrecoverable when a disk fails. Therefore, RAID level 5 is introduced as it is very related to striping, but with some level of redundancy. RAID 5 stripes all participating disks, where parity is rotated over all disks as shown in figure 6.15. Through the use of parity partitions, RAID 5 can tolerate one disk failure without breaking the whole array which makes the whole construction more fault tolerant and recoverable. If any customer require redundancy on their ingest disks for any reason, this experiments purpose is to determine what rate KSPT can guarantee their customers with such requirements.

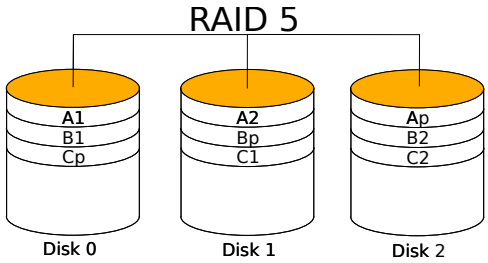


Figure 6.15: Example of how RAID-5 rotates parity partitions along all disks. Note that disk 0 hold the parity partition for partition C, disk 1 holds the parity partition for partition B and disk 2 holds the parity partition for partition A.

The ingest RAID will through this experiment consist of 4 HDDs organized in RAID level 5, with stripe size 256 kilobytes. The setup will be exposed for data ingest, only as well as with processing enabled. It is expected that this

Requested rate	808MBps
Actual max rate	804.69MBps
Total data written	421, 976, 342, 528 bytes
Exact runtime	599.39 seconds
Pages written	103, 021, 980 pages
Page size	4096 bytes
Achieved rate	671.39MBps
Limited to	83.4 % of max rate

Table 6.7: Perfio output for experiment with highest successful rate, when using four HDDs configured in RAID 5, as ingest disks.

setup can achieve nearly the same rates as striping did, but as it requires some overhead in terms of generating the parity partitions and checksums this setup is expected to achieve lower rates.

Capture The tests performed focusing on data capture only, acquired the results as presented in table 6.7. The total ingest rate achieved was reported to be 671.39MBps (or 5.37Gbps), when using one input channel. The resulting dataset was 393GB big and it was generated over 599.39 seconds. This is, as expected, a bit lower than the 931.12MBps gained through striping, but it is a more fault tolerant setup.

As for the analysis, it all seems to follow the same patterns as before. The figures not presented here can be found in appendix A.2.1. The CPU spends 94-95% of the time idling, and system level processes seem to run on the CPU for 1 to 3% of the time. The exceptions is the disk activity and memory usage through this experiment, which is why this is the only statistics presented for this setup. The disk activity is presented in figure 6.16, and from the figure one can see that the disk activity is very spiking the first 190 seconds of the execution. As discussed earlier, this is caused by the disk caching mechanisms. The graph shows very stable behaviour from 190 seconds and until the experiment ends. At the end, one can see one big spike. This is the disk cache being flushed to disk. The disk behaviour confirms that this is the maximum sustained performance for this setup because of the stable behaviour even with file system caching enabled.

The memory usage are presented in figure 6.17. The memory usage peaks after about 190 seconds, which is the same time as when the disk activity seems to stabilize itself. This means that the benefit of disk caching is no longer a significant factor, and the system is then limited to the capacity of which the disks can provide. This is also when the system are the most vulnerable, due to the high incoming rate, as one read access caused by any process may cause

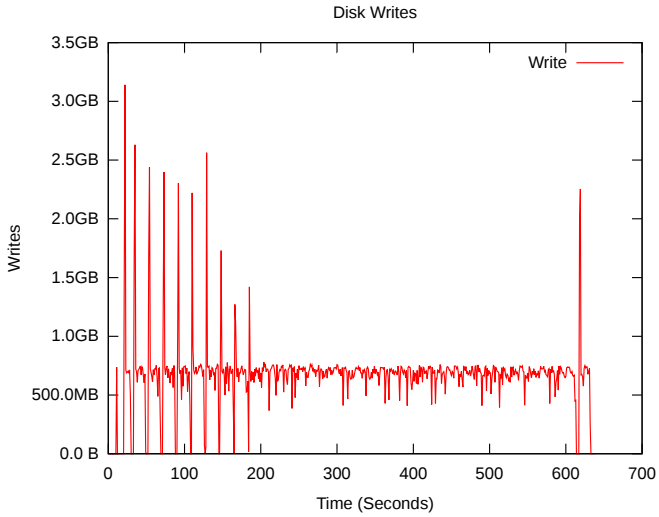


Figure 6.16: Visualization of disk activity when using 4 HDDs organized in RAID 5 as ingest disks.

the system to overflow and data is lost.

As this test have only exploited the maximum performance when the RAID is intact, one must assume that the RAID will tolerate a disk failure during a satellite pass at this rates. The reason for that is because one does not need to generate data checksums for parity purposes. Therefore, there is less load generated on the RAID controller, and the array is still intact. If two disks or more fails, one must assume that data will be lost as the array will lose the writing performance of one disks in addition to the possibility for parity partitions.

Capture and Processing This experiment will explore the capabilities of four HDDs organized in RAID 5 when performing both data capture and near real-time processing.

The results are presented in table 6.8. The results show that the generated dataset is equal to 169.61GB, and it was created over 599.93 seconds. The capturing rate was therefore reported to 289.49MBps. The data was then read from disk with an average rate of 261MBps. This is, as expected, marginally better than four striped disks performed which also means that this is so far the best setup when performing near real-time processing.

As for the analysis, it follows the same patterns as presented in section 6.1.1, paragraph `Capture and Processing`. The graphs can be found in appendix A.2.2. The experiment has not exposed the system to any significant load in terms

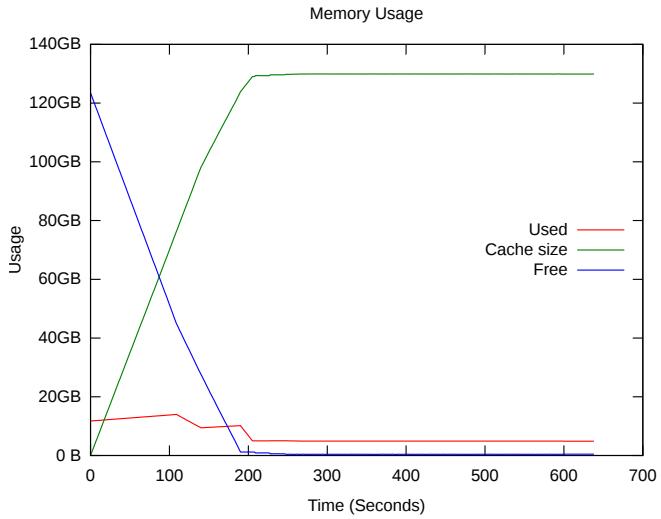


Figure 6.17: Visualization of memory usage when using 4 HDDs organized in RAID 5 as ingest disks.

Requested rate	305MBps
Actual max rate	304.69MBps
Total data written	182, 112, 485, 376 bytes
Exact runtime	599.93 seconds
Pages written	44, 461, 407 pages
Page size	4096 bytes
Achieved rate	289.49MBps
Limited to	95.0 % of max rate

Table 6.8: Perfio output for experiment performing data capture with processing enabled, using 4 HDDs organized in RAID 5 as ingest disks.

RAID configuration	Input Channels	Ingest only (MBPS)	Ingest and processing	
			Ingest rate (MBPS)	Read rate (MBPS)
4X HDDS RAID 1+0	1	474.43	148.82	130
4X HDDS striped	1	931.12	281.32	261
8X HDDS striped	1	1054.13	259.60	235
4X HDDS RAID 5	1	671.39	289.49	261

Table 6.9: Summary of test results, introducing RAID level 5 results with 4 HDDs.

of CPU usage. The memory usage is high as experienced in previous experiments, and peaks after about 200 seconds. The analysis suggest that the ingest disks are the bottleneck when performing both data ingest and near real-time processing.

Summary In table 6.9 the results are all presented. Note that 4 HDDs in RAID 5 provides better performance when performing both data capture and near real-time processing than the two striped RAID configurations with 4 and 8 HDDs. It is also worth noticing that 4 striped HDDs outperform RAID 5 by 260MBPS, when only performing data capture. One can discuss how necessary the data redundancy are for a system as MEOS capture because one can split the demodulated signal and send it to two receiving servers. Through such a setup one can achieve better throughput by striping 8 HDDs as ingest disks and achieve redundancy across several servers.

6.1.4 Increasing the RAID 5 participants

As RAID 5 has shown itself as an interesting setup when performing both data capture and near real-time processing, this experiments goal is to explore how RAID 5 scales by using 8HDDs as ingest disks instead of 4. If one recalls from section 6.1.2 which tested 8 striped disks, the throughput became lower than when using 4 disks.

Capture When performing capture only, 8HDDs organized in RAID 5 provides results as presented in table 6.10. From the table one can see that the total amount of data written is equal to 553.53GB over 599.20 seconds, achieving an average rate of 945.96MBPS (or 7.57Gbps). The target of 10Gbps has not been achieved yet. This means that when one only has to capture data, striping is a better configuration as the achieved rate for 8 HDDs striped was 1054.13MBPS.

The analysis does not show anything new. This experiment follows the same

Requested rate	1215MBps
Actual max rate	1218.75MBps
Total data written	594, 349, 654, 016 bytes
Exact runtime	599.20 seconds
Pages written	145, 105, 359 pages
Page size	4096 bytes
Achieved rate	945.96MBps
Limited to	77.6 % of max rate

Table 6.10: Perfio output for experiment performing data capture only, using 8 HDDs organized in RAID 5 as ingest disks.

patterns as presented in section 6.1.2 which presents the analysis for 8 HDDs, striped. The disk activity is very spiking for both configurations, and the general CPU load is almost negligible where system level processes tend to occupy the CPU for 2-6%. The memory usage seems to hit the top after 150 seconds, which is a bit later than with the striped RAID. Because the results follow very similar patterns as experienced before, the graphs for this experiment can be found in appendix A.2.3.

Capture and Processing Based on the results presented in section 6.1.3 it seems like RAID 5 is performing well when performing both data capture and near real-time processing. The results so far show that 4 HDDs organized in RAID 5 provides the best throughput of all so far, which is why this experiment goal is to determine RAID 5's scalability in terms of increasing the number of disks used in the RAID from 4 to 8. It was demonstrated in section 6.1.2 that a striped RAID does not scale very well up to 8 HDDs when performing both data capture and near real-time processing.

The experiment results are presented in figure 6.11, and show that the ingest disks captured 173.60GB of data over 599.09 seconds. This results in an average ingest rate of 296.72MBps. The process reading data achieved an average rate of 268MBps. The results demonstrate a slight increase in ingestion rate compared to 4 HDDs in RAID 5. It is only a few MB better, which means that RAID 5 does not scale very well up to 8 HDDs.

As for the analysis, the analysis shows nothing new. The memory usage seems to peak after 220 seconds, and has a stable behaviour throughout the experiment which is seen in all experiments so far. With disk caches enabled, the disk behaviour become very spiking as expected, and the CPU seems to spend more than 90% of its time idling through the experiment. As no new trends or interesting results are provided from the graphs, they are displayed in appendix A.2.4.

Requested rate	315MBPS
Actual max rate	312.50MBPS
Total data written	186, 399, 064, 064 bytes
Exact runtime	599.09 seconds
Pages written	45, 507, 923 pages
Page size	4096 bytes
Achieved rate	296.72MBPS
Limited to	95.0 % of max rate

Table 6.11: Perfio output for experiment performing data capture with processing enabled, using 8 HDDs organized in RAID 5 as ingest disks.

RAID configuration	Input Channels	Ingest only (MBPS)	Ingest and processing	
			Ingest rate (MBPS)	Read rate (MBPS)
4X HDDS RAID 1+0	1	474.43	148.82	130
4X HDDS striped	1	931.12	281.32	261
8X HDDS striped	1	**1054.13**	**259.60**	**235**
4X HDDS RAID 5	1	671.39	289.49	261
8X HDDS RAID 5	1	**945.96**	**296.72**	**268**

Table 6.12: Summary of test results, introducing RAID level 5 results with 8 HDDs. The two best configurations is presented with bold numbers.

Summary To conclude the results so far, all results are presented in table 6.12. It seems like RAID 5 and striping both scale very poorly up to 8 HDDs, where RAID 5 is the preferred setup when performing both data capture and near real-time processing. Note that 8 HDDs in RAID 5 only provide 7MBPS more than 4 HDDs. When performing data capture only, it seems like striping 8 HDDs is the best option as it achieved the best result.

6.1.5 Changing Disk Technology

This experiment goal is to determine whether MEOS capture can achieve better results by using SSDs instead of HDDs. It is expected that the SSDs will perform much better when performing both data ingest and near real-time processing, because there are no physical, moving parts in such disks like in mechanical HDDs. There are however some other challenges forcing some changes in the experiment setup.

Because the test environment is only equipped with two relatively small SSDs, with 200GB each, one cannot perform tests over 600 seconds as done in earlier experiments. The disks in total provide 373GB of storage capacity, and one

Requested rate	2000MBps
Actual max rate	2000MBps
Total data written	341, 189, 853, 184 bytes
Exact runtime	299.41 seconds
Pages written	83, 300, 063 pages
Page size	4096 bytes
Achieved rate	1086.79MBps
Limited to	54.3 % of max rate

Table 6.13: Perfio output for experiment with highest successful rate, when using two SSDs, striped, as ingest disks.

cannot verify the generated dataset if the disk array is overwritten. Therefore the experiment length will be cut to 300 seconds, or 5 minutes, instead of 10 minutes as in previous experiments. Recall from section 6.1.2, paragraph `Capture`, the largest dataset created so far was 617,25GB after 10 minutes. By running the same rate in 5 minutes, the dataset size will be half the size. As there are only two SSDs available, one cannot configure them to RAID level 5 for example, because it requires at least 3 disks. Therefore this experiment will test the two disks performance in stripe[3].

Capture This experiment will explore the striped SSDs abilities in terms of data ingest only. The goal is to find out how two SSDs perform in comparison with 8HDDs to ingest data.

The experiment results are presented in table 6.13. The highest recorded rate was reported to be 1086.79MBps, which is marginally better than what 8 HDDs managed. This is very convenient in several ways. First of all, by using two physical disks instead of eight, the RAID controller are faced with less work in terms of spreading data evenly across the raid, enabling higher throughput. Two disks use less space in the server chassis than eight, which allow space for other components if needed. On the contrary, SSDs are expensive compared to HDDs. This will, as all other technology, become more affordable as the years go by. Based on the test results, SSDs are the recommended disk technology to use for future generations of MEOS capture.

In terms of the analysis, the experience is very much like demonstrated with 8 HDDs. The average load is presented in figure 6.18, and the figure suggest that the computer's CPUs can manage much more load. The last 15 minutes show peak just below 1.2 waiting processes, where the last minute peaks just under 2.2 waiting processes.

3. RAID level 0

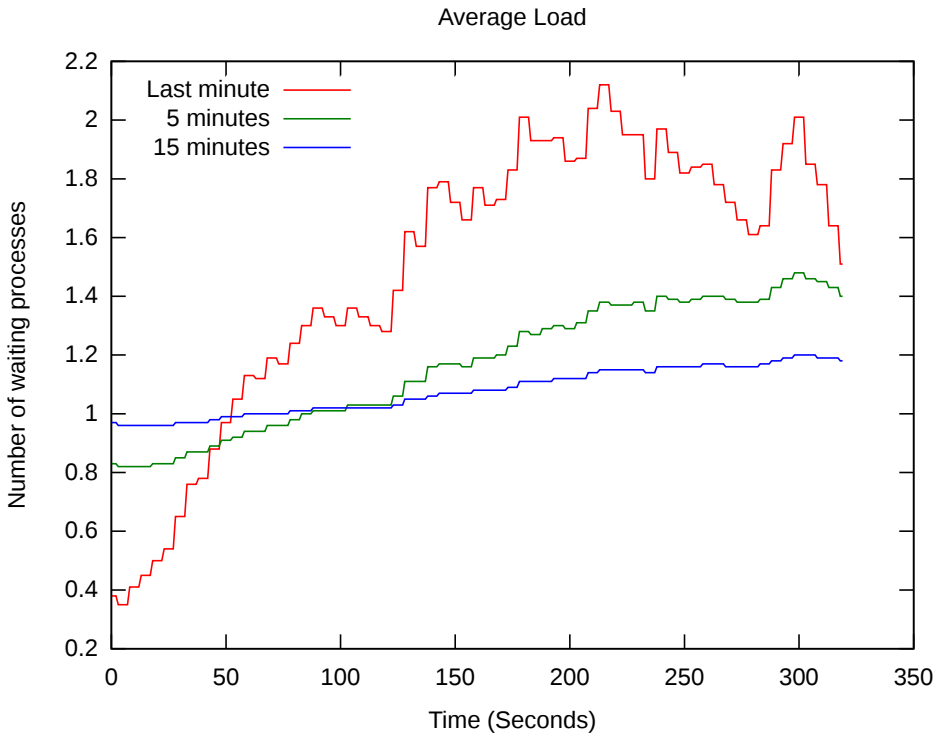


Figure 6.18: Visualization of average load when using two SSDs striped, as ingest disks.

The memory usage is presented in figure 6.19, which shows that the memory usage peaks after about 120 seconds of execution time. This is a slight advantage compared to figure 6.10, which demonstrates memory usage with eight HDDs striped as ingest disks. This means that the system becomes less prone to drop in disk write performance, as the memory usage peaks at a later time, meaning the disks ingest data faster. This is only an advantage, not an improvement, because the problem now occurs on a later time in the satellite pass.

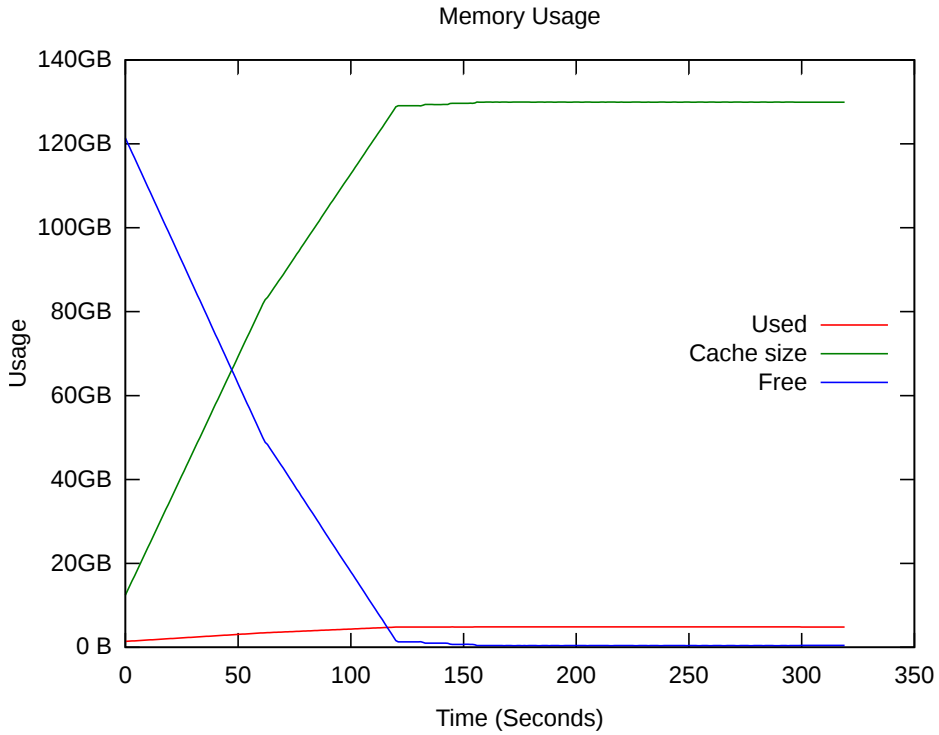


Figure 6.19: Visualization of memory usage when using two SSDs striped, as ingest disks.

The disk activity through this experiment is presented in figure 6.20. As the figure suggests, the SSDs may cope with higher rates due to the drops that occur but. To determine this, one must implement prefetching of page descriptors and store them locally on the data receiving board, and perform tests with this feature. It is assumed that such a feature would increase the throughput.

To sum this up, the system is as we know generally comfortable with rates up to just over 8Gbps. The challenge is to get all data to disk, and SSDs prove themselves to be very effective in terms of data ingest. The question now is how good they perform with both data capture and near real-time processing.

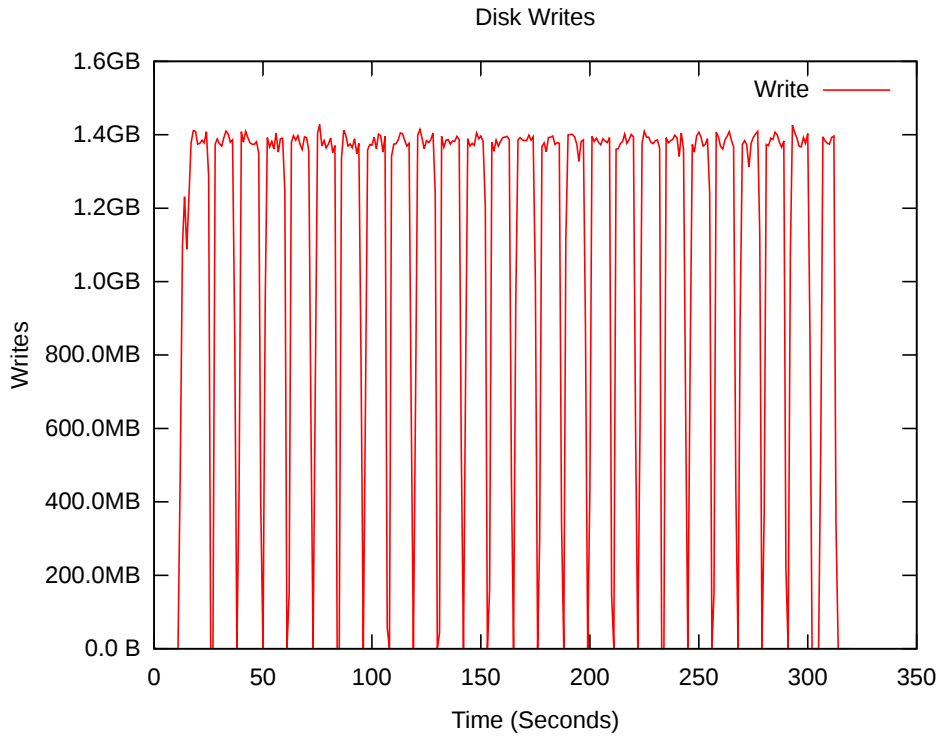


Figure 6.20: Visualization of disk activity when using two SSDs striped, as ingest disks.

Requested rate	482MBps
Actual max rate	484.38MBps
Total data written	138, 684, 661, 760 bytes
Exact runtime	299.10 seconds
Pages written	33, 858, 887 pages
Page size	4096 bytes
Achieved rate	442.20MBps
Limited to	91.3 % of max rate

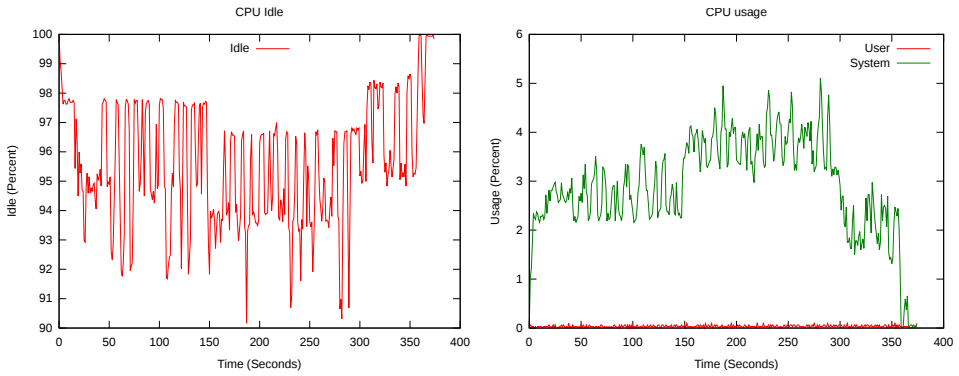
Table 6.14: Perfio output for experiment with highest successful rate when using two striped SSDs as ingest disks. This experiment was executed with processing enabled.

Capture and Process This experiment is probably the experiment where it is expected that the SSDs really show what they can do. This experiment will determine how much data the SSDs can ingest when performing near real-time processing on the captured data. Because the SSDs have no physical moving parts, it is expected that this is their field of excellence, providing top numbers in terms of throughput.

The experiment does provide fairly good results, and the output from Perfio is shown in table 6.14. As one can see, the ingest rate is reported to be 442.20MBps (3.537Gbps). The reported reading rate from PV was 372MBps (2.976Gbps). It looks very promising as this is done with only two SSDs available. If this scales perfectly, one can capture and process data with rates above 7Gbps with four SSDs available. This would however require further testing to confirm, but as the system has proven itself to cope with rates above 8Gbps for data ingest only it seems plausible. The question is whether the RAID controller can manage such rates with two-way traffic or not.

However, in terms of the analysis, the system does not show very interesting results. The CPU seems to be very comfortable through this experiment, as demonstrated in figure 6.21. From the figure we can see that the CPU drops to 90% idle time just before 200 seconds of execution time. In general it seems like the CPU spends 95% of its time idle, as average. The figure also show that user level processes almost do not get to run at all, and the CPU allows system level processes to run. Because Perfio use the data receiving board device driver which perform operations within the Linux kernel, the device driver is regarded as a system level process, and it seems to use between 1 and 5% of the CPU.

The memory usage is presented in figure 6.22. The memory usage peaks after 150 seconds, and is generally stable after 150 seconds of execution time. One



(a) Figure visualizes the amount of time the CPU spends idle, when using two SSDs as ingest disks.

(b) Visualization of the amount of resources used by user and system level processes during the experiment.

Figure 6.21: Figure shows the general load on the CPU during experiment execution when using two SSDs as ingest disks.

may expect that the memory usage would peak at an even later point if there was more SSDs participating in the RAID, as the ingest rate would increase.

As for the disk activity, the analysis tool is not able to record any reading activity from the physical disk. This means that the reading process has fetched all its data from caches in the hierarchy. That is either the cache on the RAID controller or disk cache. The disk activity will therefore only show the ingest activity, and is presented in figure 6.23. As the figure shows, the disk activity is very spiking and periodical. This indicates that the disks are capable of ingesting more data per second. Nevertheless, the incoming rate is limited by the outgoing data where the RAID controller and file system has to read the data and write it to a processed storage as well.

The average load calculated is presented in figure 6.24 and confirms the low CPU usage shown in figure 6.21. The server is not affected by any significant load during this experiment. The average load show that the last 15 minutes is just around 1,4, meaning that 1,4 processes has been queued for run-time on the CPU the last 15 minutes. The last minute spikes after 150 seconds where 2.2 processes are waiting to run on the CPU. The other processes must be the OS daemons administrating some activity between user and system level. As the system is equipped with 32 cores, and can run 32 processes in parallel, the load is negligible.

To sum this up, the experiment demonstrate that two SSDs are better at performing read and writes than 4 and 8 HDDs, where the system in general does not show any sign of high CPU load at this rates.

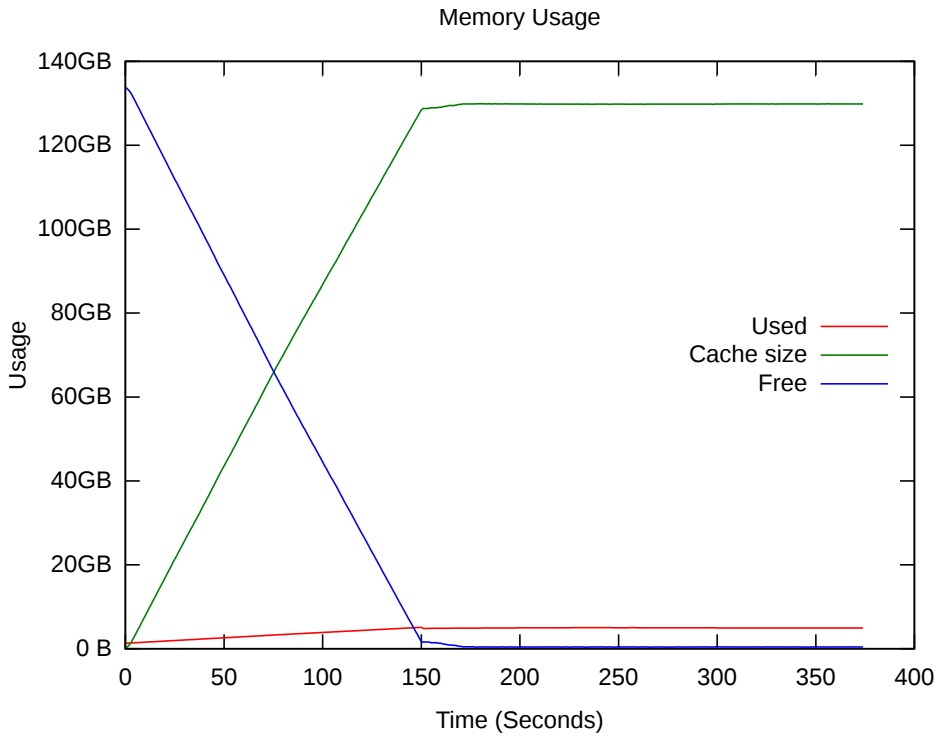


Figure 6.22: Visualization of memory usage when using two SSDs striped, as ingest disks.

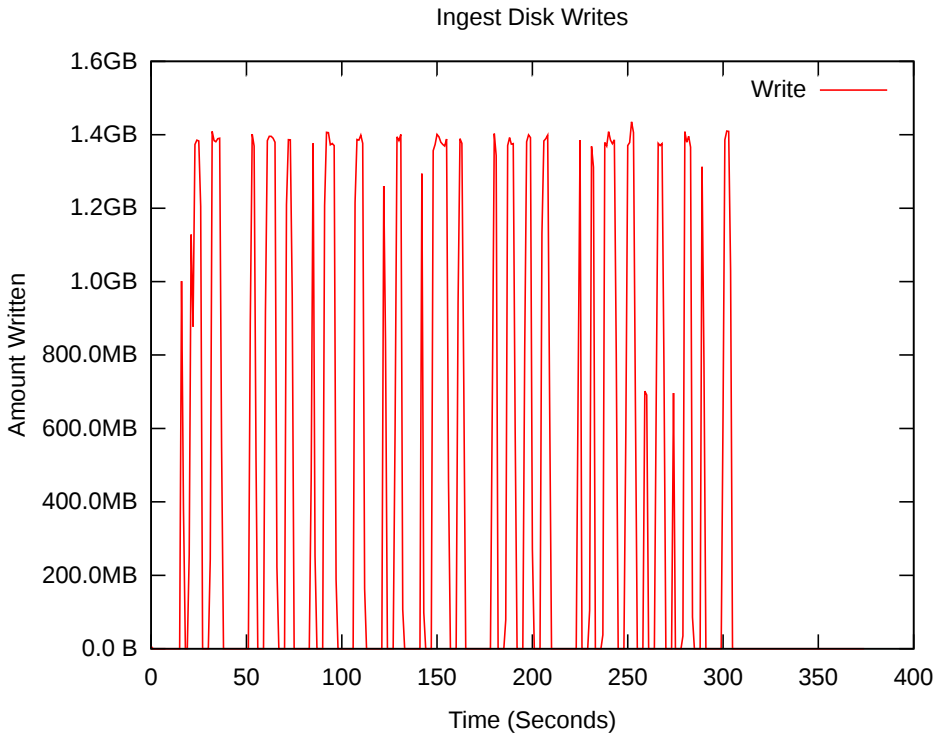


Figure 6.23: Visualization of disk activity when using two SSDs striped, as ingest disks.

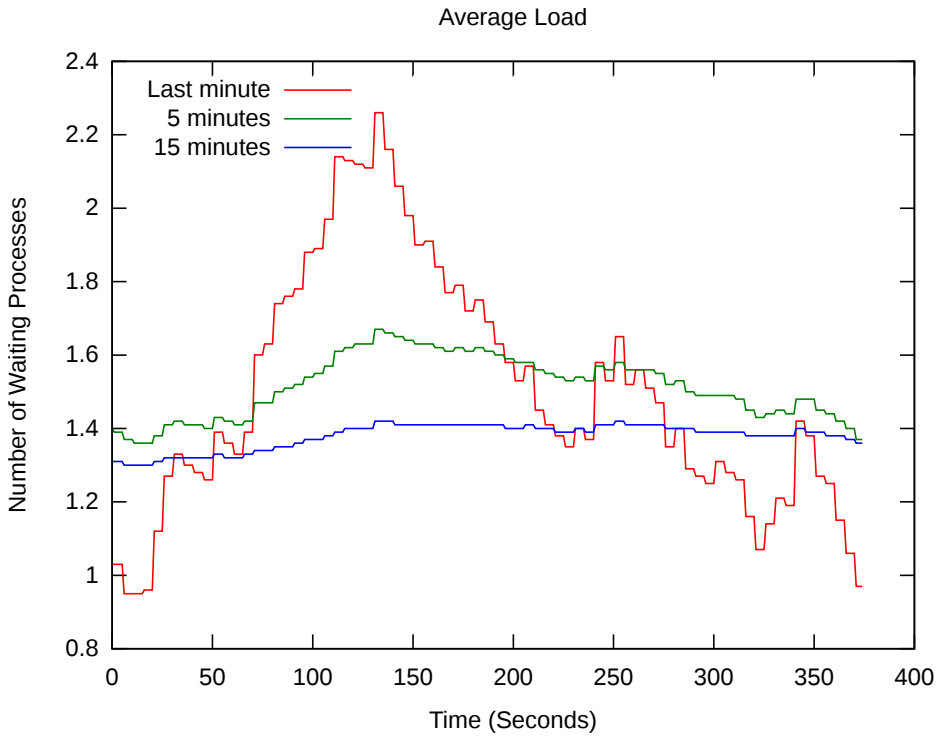


Figure 6.24: Visualization of the average load when using two SSDs striped, as ingest disks.

RAID configuration	Input Channels	Ingest only (MBPS)	Ingest and processing	
			Ingest rate (MBPS)	Read rate (MBPS)
4x HDDS RAID 1+0	1	474.43	148.82	130
4x HDDS striped	1	931.12	281.32	261
8x HDDS striped	1	1054.13	259.60	235
4xHDDS RAID 5	1	671.39	289.49	261
8x HDDS RAID 5	1	945.96	296.72	268
2x SSDS striped	1	1086.79	442.20	372

Table 6.15: Summary of test results, introducing two SSDs striped.

Summary The experiment results so far is presented in table 6.15. The two experiments in this section have demonstrated that SSDs cope with high rate data very well, both when performing data ingest only, and data ingest with near real-time processing. The advantage of not having any physical moving parts within the disks seems to provide great improvements, especially with two-way traffic. Even though these tests have been conducted with only two SSDs, they still demonstrate their superiority over traditional HDDs in terms of performance. One must however perform further testing with larger disks, or more disks in order to put them through full time tests running for 10 minutes or more, which is the length of a satellite pass. There is no indication that this would cause any problems, but this has to be tested in order to prove it.

Introducing SSDs into production systems provide other issues. One would have to replace the disks more often, as SSDs have shorter life cycles than traditional HDDs, where the disks used in this thesis are guaranteed, by HP, to withstand 10 full rewrites per day for 5 years[25].

6.2 Two Test Generators

All experiments so far, have been conducted with only one test generator representing one input channel. If one recall from chapter 3, MEOS Capture supports several input channels. Since the thesis goal of capturing 10Gbps has not been achieved yet, another input channel was introduced to the system. The idea is that since one input channel can manage more than 8Gbps, it is possible to achieve 10 Gbps when using two input channels. The reason for this is that the two data receiving boards are placed on their own, separate PCIe bus. It is also known that the theoretical transfer rate between host memory and ingest disks are 12Gbps through the specifications of the SAS bus[64]. When one decrease the incoming data rate per data receiving board and writing the

data generated by the two test generators to separate files, one can achieve higher combined throughput as the data run through different buses into host memory. From there the RAID controller, OS and file system takes care of ingesting the data into two separate files within the file system.

These experiments will only perform data capture in order to determine the highest rate supported, for this specific test environment and configurations. Based on previous results, performing near real-time processing significantly limits the ingest rate. Therefore, near real-time processing will not be tested with this setup.

6.2.1 SSDs as Ingest Disks

Since the server are equipped with two data receiving boards and the fact that Perfio does specify what data receiving board to use, one can run two instances of Perfio, each instance are assigned to its own data receiving board, writing data to its own file. In terms of the analysis, no changes must be done as it runs within the OS, and the ingest disks are exposed to the OS as one logical disk. The ingest disks will consist of the two SSDs used in previous experiments, as they demonstrated the best performance when performing capture only. The experiment runtime will be 300 seconds, because of the limitation in disk size as discussed in section 6.1.5.

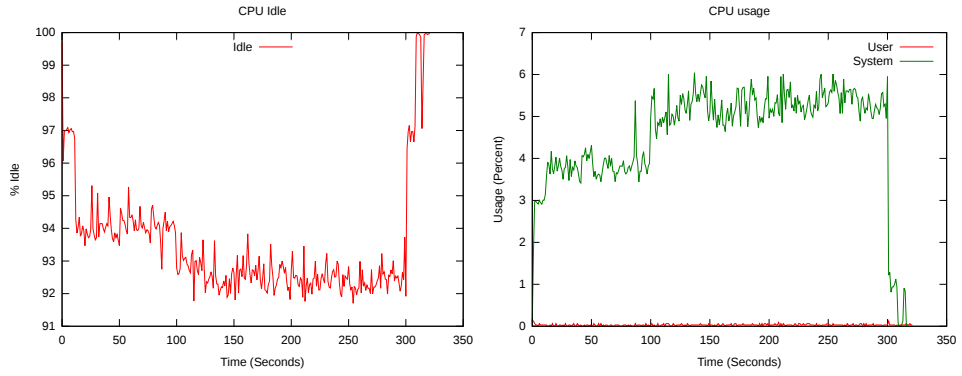
The experiment result are presented in table 6.16, and as one can see the two input channels reached rates of 636.76MBps and 632.88MBps. This represents a combined rate of 1269.64MBps, which is equal to 10.16Gbps. This means that the goal of capturing 10Gbps has been achieved in this experiment. Now, as the amount of data captured at this rate became 371GB, the system have not been tested with higher rates as the ingest disks only provide 373GB. The disks are not big enough to handle higher rates over 300 seconds. One could decrease the experiment runtime to 150 seconds, but that would be even less representative for a satellite pass making the experiment more a proof of concept. But the proof of concept would not be very good, because the system tends to become vulnerable after the memory usage peaks, which is typically around 100 seconds at rates above 8Gbps. The system vulnerability is caused by the full disk caches which means the system are down to the raw rate at which the ingest disks can write in order not to overflow in memory. Therefore, it is the time after 100 seconds that is interesting where the ingest disks really have to ingest data in order to not suffer from buffer overflow in RAM.

As for the analysis, the graphs follow very similar patterns as demonstrated before. The CPU utilization visualized in figure 6.25, shows that the CPU idle time drops to between 92 and 93%, which is reflected by the usage represented

Parameters	Perfio Instance 1	Perfio Instance 2
Requested rate	770MBps	770MBps
Actual max rate	773.44MBps	773.44MBps
Total data written	200, 030, 552, 064 bytes	198, 814, 203, 904 bytes
Exact runtime	299.59 seconds	299.59 seconds
Pages written	48, 836, 495 pages	48, 539, 311 pages
Page size	4096 bytes	4096 bytes
Achieved rate	636.76MBps	632.88MBps
Limited to	82.3 % of max rate	81.8 % of max rate

Table 6.16: Perfio output for experiment with highest successful rate when using two striped SSDs as ingest disks, and two test generators. Note that both input channels have achieved over 630MBps.

by system level processes peaking at 6%. This is however expected as Perfio now is launched in two instances, generating more load on the two CPUs.



(a) CPU Idle time **(b)** User and System level utilization

Figure 6.25: Figure show the CPU utilization when performing data capture from two input channels.

The average load is presented in figure 6.26. As one can see, the average load peaks just under 2.5 queued processes for the last minute, which is expected as two instances of Perfio are executed with the highest process priority (nice value). The last 15 minutes show more stable behaviour just above 1,5 queued processes. As argued before, this is negligible load to the computer CPUs meaning very few other processes generate significant load while the experiments are executed.

As for the memory usage, it peaks after 100 seconds of execution time, which is expected due to the previous experiments performing higher than 8Gbps seems to use just over 100 seconds to fill the memory available with data. The

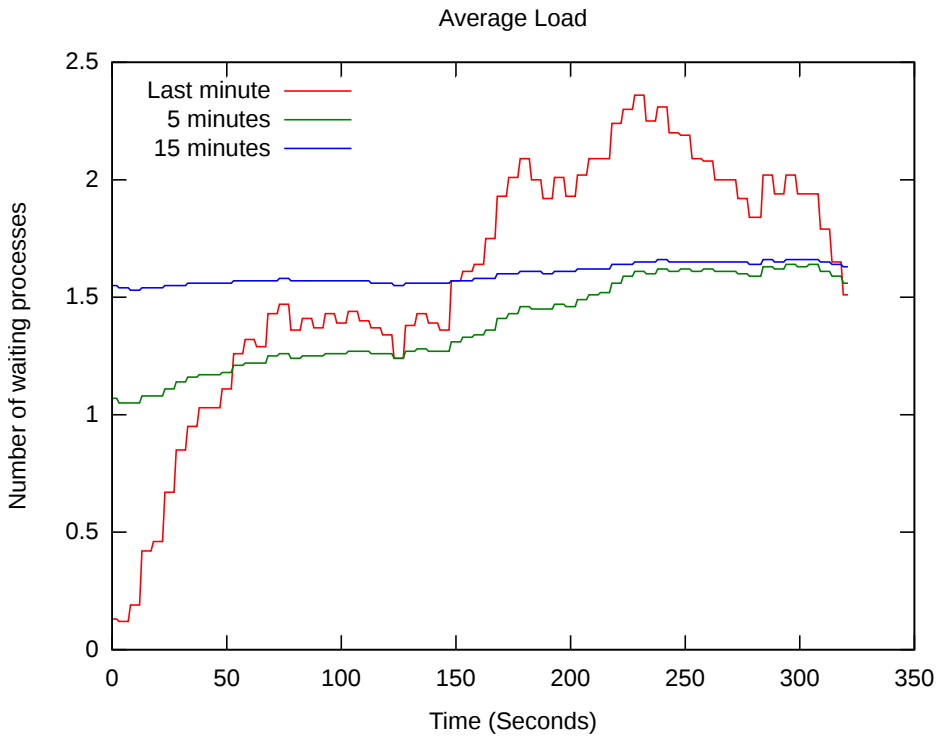


Figure 6.26: Average load when using SSDs as ingest disks and two input channels.

memory usage is presented in figure 6.27.

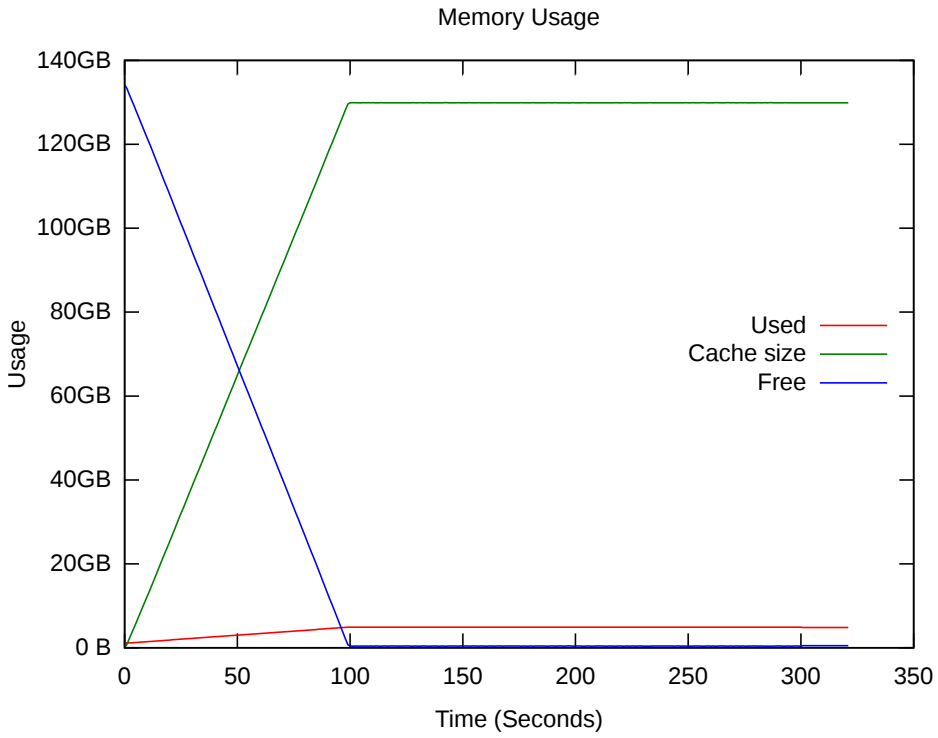


Figure 6.27: Memory usage when using SSDs as ingest disks and two input channels.

The disk activity is presented in figure 6.28. The disk activity suggest that the system throughput is close to what the system can manage. The drops in write performance indicates that it is still possible to write a few more MBPS, and to find out how much one must perform further testing. Further testing would as discussed earlier require larger disks or shorter experiment duration. By the two options, it is recommended to increase the per disk storage capacity or number of disks in the RAID in order to perform tests over 600 seconds which is more representative for a satellite pass compared to the 300 seconds used in this experiment.

Summary The table of all results so far is presented in table 6.17. Note that the two SSDs have not been exposed for two input channels, with near real-time processing enabled as this experiment's purpose was to determine whether one could capture 10Gbps or not. Since 10Gbps is equal to 1250MBPS, one can see that this is achieved through this experiment, with only two SSDs.

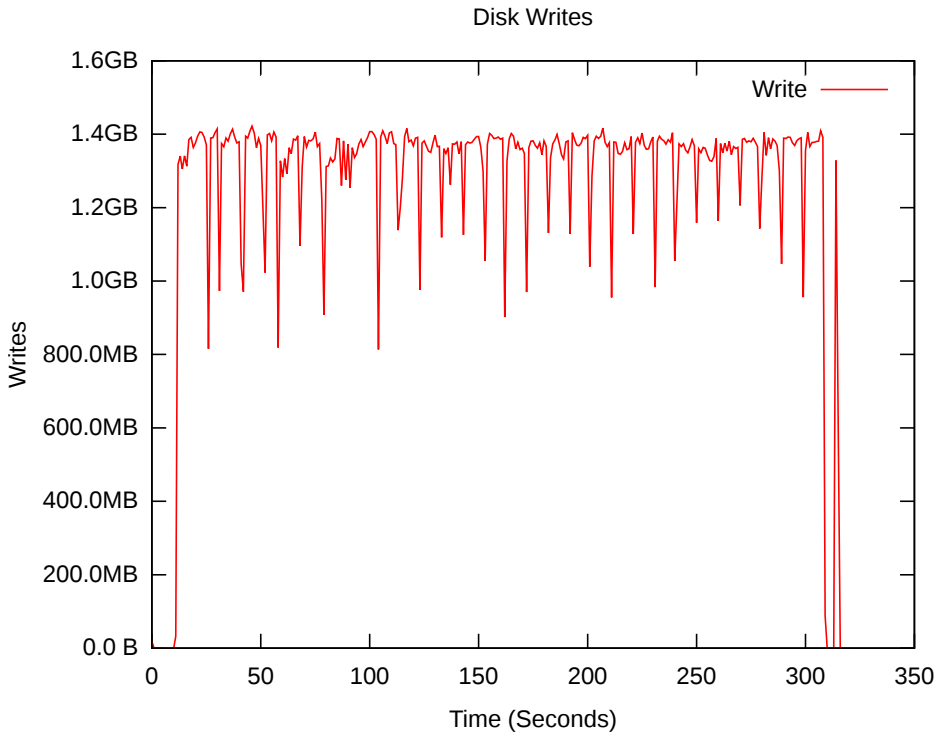


Figure 6.28: Disk activity when using SSDs as ingest disks and two input channels.

RAID configuration	Input Channels	Ingest only (MBPS)	Ingest and processing	
			Ingest rate (MBPS)	Read rate (MBPS)
4x HDDs RAID 1+0	1	474.43	148.82	130
4x HDDs striped	1	931.12	281.32	261
8x HDDs striped	1	1054.13	259.60	235
4x HDDs RAID 5	1	671.39	289.49	261
8x HDDs RAID 5	1	945.96	296.72	268
2x SSDs striped	1	1086.79	442.20	372
2x SSDs striped	2	1269.64	—	—

Table 6.17: Summary of test results, introducing SSDs when using two input channels. Note that the goal of 1250MBPS has been achieved through the last experiment. The setup has not been tested with near real-time processing, which is why the two are missing.

Parameters	Perfio Instance 1	Perfio Instance 2
Requested rate	750MBps	750MBps
Actual max rate	750MBps	750MBps
Total data written	394, 755, 309, 568 bytes	390, 779, 109, 376 bytes
Exact runtime	599.07 seconds	599.06 seconds
Pages written	96, 377, 099 pages	95, 405, 661 pages
Page size	4096 bytes	4096 bytes
Achieved rate	628.43MBps	622.10MBps
Limited to	83.8 % of max rate	82.9 % of max rate

Table 6.18: Perfio output for experiment with highest successful rate when using 8 striped HDDs as ingest disks. This rates combined just achieves above 10Gbps.

6.2.2 HDDs as Ingest Disks

This experiment will test whether 10Gbps is achievable when using HDDs as ingest disks and with two input channels. As one recall from table 6.17, the best setup using HDDs for data capture only was 8 HDDs striped. For one input channel, that particular setup achieved 1054.13MBps, and was beaten by the SSDs by 32MBps. Recall from section 6.1.2, where the disk activity presented suggested that the HDDs was capable of ingesting more data, as the behaviour was very spiking. Therefore, it is expected that this setup is capable of achieving 10Gbps when only performing data capture.

This experiment's duration will differ from what was used for the SSDs, because the total storage capacity for this RAID is 4.4 terrabyte. Therefore this experiment will perform a full 10 minutes test.

The experiment started out with a requested rate of 500MBps per channel, and ended at a requested rate of 750MBps. The experiment results are presented in table 6.18. From the table one can see that the two test generators have created two datasets, where channel one generated 367.43GB and channel two generated 363.94GB in total. This combined is 731.37GB, generated over 599 seconds. This gives an average combined rate of 1250.53MBps. In other words, 10.02Gbps.

As for the analysis, it seems like the server is not exposed to any significant load. The average load shows that the last minute, two processes on average was queued. The last 15 minutes passes just above 1, which indicates that the server has not had many processes queued for the last 15 minutes. This is displayed in figure 6.29.

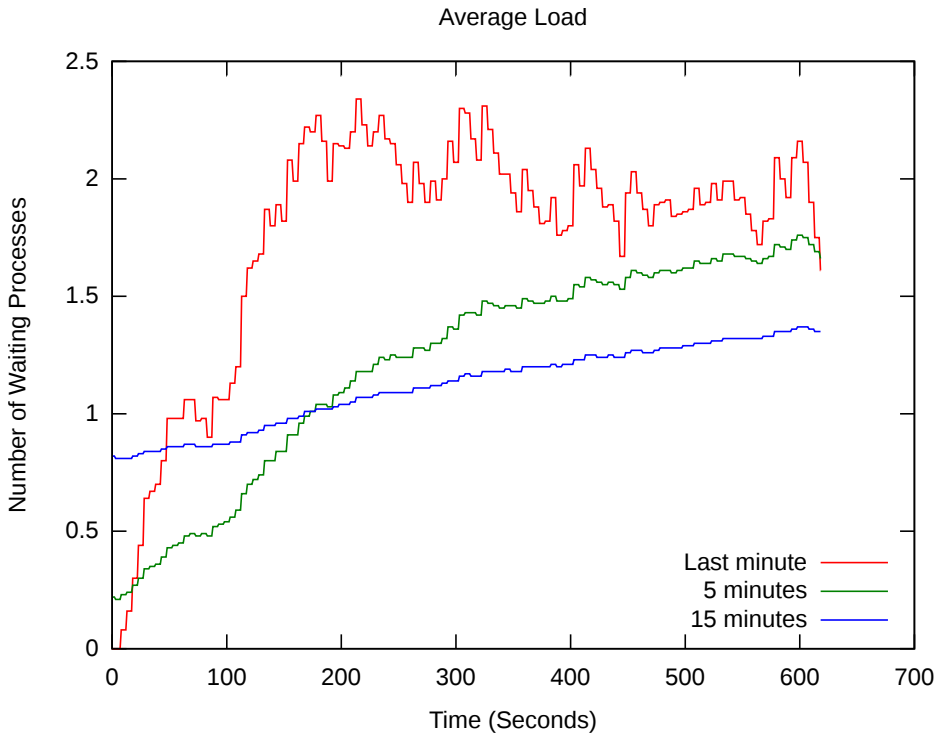


Figure 6.29: The figure shows the average load on the server during the experiment, with 8 HDDs as ingest disks and two input channels.

The CPU seems to spend more time executing system level processes than seen before, as the highest recorded value is over 8%. The graph, presented in figure 6.31, shows that the CPU spends about 4-7% of its time on system level processes which is the highest recorded average through all experiments. This is interesting as the previous experiment, presented in section 6.2.1, achieved higher average rate but lower CPU usage on system level processes (see figure 6.25). It may be explained by the disk caching mechanisms impact on the sustained rate due to the experiment runtime, where the experiment presented in section 6.2.1 runtime was 300 seconds compared to 600 in this experiment.

The CPU also seems to spend over 90% of the execution time idling, as shown in figure 6.30. As all previous experiments, the CPU is not exposed to high load.

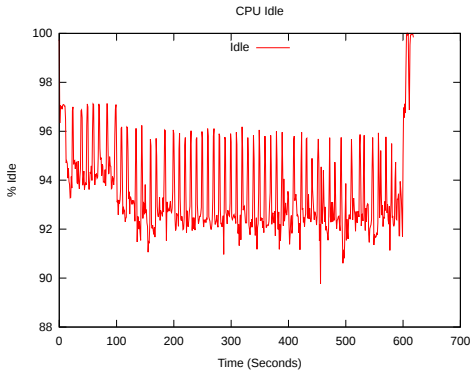


Figure 6.30: The figure shows the amount of time the CPU spent idle through the experiment. Note that we have never recorded values below 90% in any experiment.

Figure 6.32 shows the disk activity through the experiment. The behaviour seems to be very spiking, even at such high rates. In previous experiments, the trend seems to be that when the disks are exposed to input rates reaching the limit of how much data the disks can ingest, the behaviour becomes stable without the large drops in writing performance. For example, figure 6.28, show a much more stable behaviour than figure 6.32. This leads to the assumption where the ingest disks are capable of capturing higher rates, and that the bottleneck seems to lie within other parts of the system. There is reason to believe that the SAS bus is exposed to very high load as its theoretical maximum rate is 12Gbps. This experiment's rate of 10Gbps are coming very close to this upper limit. The RAID controller is another component that may be causing the system to overflow when the rates are above 10Gbps.

The memory usage, visualized in figure 6.33, does not show any new trends

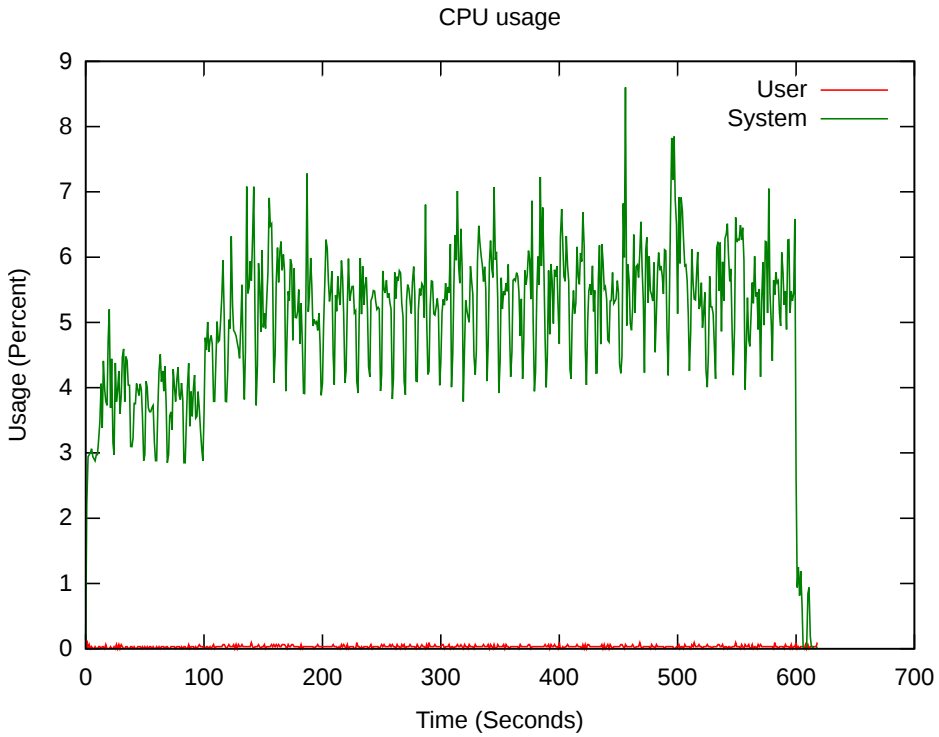


Figure 6.31: The figure shows the amount of time, in percent, the CPU has spent on executing system and user level processes.

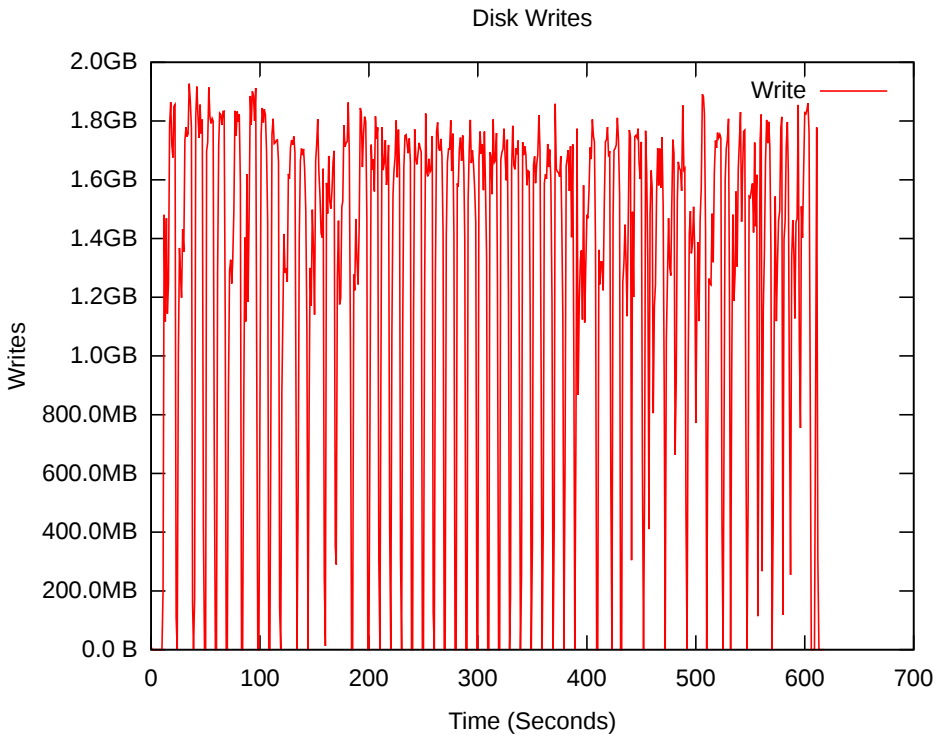


Figure 6.32: The figure shows the ingest disks activity through the experiment.

or patterns, as it peaks after 100 seconds exactly. The behaviour is exactly the same as we experienced in section 6.2.1.

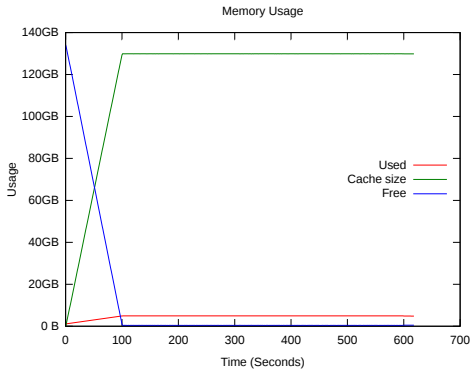


Figure 6.33: The figure shows the memory usage through this experiment. Note that the caches peaks after 100 seconds, which we have seen before.

6.3 Summary and Discussion

All results presented throughout this chapter are summarized in table 6.19. From the table one can see that the SSDs provides the best results, even though all tests were executed with only two SSDs available. Of all configurations using HDDs, striping 8 HDDs seems to be the better setup when only performing data capture, achieving 1053.13MBPS. When processing is enabled, RAID 5 using 8 HDDs seems to be the setup by 7MBPS. Therefore one may argue that the cost is larger than the gain when using 8 HDDs instead of 4 when performing both capture and near real-time processing.

In the bigger picture, it seems like performing both data capture and near real-time processing simultaneously is a significant limiting factor in terms of system throughput. Even when using SSDs as ingest disks one only gain 3.5Gbps ingest rate as 2.9Gbps are outgoing for processing purposes. The experiment results show a significant drop in performance when one has to write and read data simultaneously, for all configurations. It is recommended that one should wait with the processing until the satellite pass is complete before one starts processing the data. By postponing the data processing, the server can use all its available resources to capture data, assuring that data will not be lost as the rates increase. The question is, how much data can one capture when performing capture and near real-time processing during a pass, and how much time is required to complete the processing of data when the data dump is done. If one instead postpone the processing to when the data dump is done, one first of all have captured more data because the rates are

RAID configuration	Input Channels	Ingest only (MBps)	Ingest and processing	
			Ingest rate (MBps)	Read rate (MBps)
4x HDDs RAID 1+0	1	474.43	148.82	130
4x HDDs striped	1	931.12	281.32	261
8x HDDs striped	1	1054.13	259.60	235
4x HDDs RAID 5	1	671.39	289.49	261
8x HDDs RAID 5	1	945.96	296.72	268
2x SSDs striped	1	1086.79	442.20	372
2x SSDs striped	2	1269.64	—	—
8x HDDs striped	2	1250.53	—	—

Table 6.19: Summary of test results, introducing two input channels with 8 HDDs as ingest disks. Note that the goal of 1250MBps has been achieved when using two input channels for both SSDs and HDDs. The setup has not been tested with near real-time processing, which is why the two are missing.

higher, and one can then allocate all resources available on the server to process the captured data. This two-step solution may decrease the total amount of time spent on both capturing and performing the processing. If near real-time processing is required by any customer, one have at least enabled the capture of larger amounts of data through the use of SSDs. This must be tested in order to achieve exact numbers, but it is a question KSPT should address at some point when moving over to the next generation servers.

Another interesting aspect is that the disk configuration proven to be the worst performing configuration, has been the one preferred so far. This demonstrates the necessity for performing the experiments done in this thesis. Changing the RAID setup is a cheap fix for systems struggling with rates today.

RAID 5 and striping seems to provide similar results, which is expected as both setups are based on striping data across disks. The difference lies within the parity partition provided by RAID 5. As previously discussed, RAID 5 provides some level of data redundancy, making it able to tolerate 1 disk failure. One would achieve the same level of redundancy, with better performance if one uses two servers. When using two servers, one split the demodulated signal and send it to the two servers. If one then stripe the ingest disks in both servers, one would achieve the same level of fault tolerance as with RAID 5. This is a more expensive solution, but one can then achieve data redundancy in combination with 10Gbps when using 8 HDDs, or two SSDs, striped as ingest disks through two input channels (or two data receiving boards).

RAID 5 provides the best performance for capture and near real-time processing

for both 4 and 8 HDDs. In this thesis, RAID 5 was not tested with SSDs because it is required at least 3 disks in such a RAID. We know, from the test results, that SSDs performs much better than HDDs when performing data capture and near real-time processing. It would be interesting to see how SSDs perform when configured in RAID 5. The results suggests that RAID 5 may achieve better performance than striping when using SSDs as ingest disks as well.

When performing data capture, we have seen through the experiments in section 5.3.1 that the system today supports 8.7Gbps. It is expected that one can achieve near 16Gbps when using two input channels with some changes to the system. By using two RAID controllers, with their own set of ingest disks, one has reason to believe that 16Gbps is achievable. That is because the data would follow completely separate paths from the data receiving boards to the ingest disks, except for when it is placed in host memory. The RAID configuration is essential here. One must use either 8 HDDs or two, or more, SSDs striped in order to achieve such rates to disk.

If KSPT implement prefetching of page descriptors, so that a list of all available page descriptors are stored locally on the data receiving board, it is expected that the system may achieve better performance per input channel. Implementing such a feature is recommended as one can improve the total throughput by a few Gbps. If the future rates exceed 20 Gbps and beyond, KSPT should consider moving the data receiving board to a newer version of the PCIe bus. Through such a change one may exceed 10Gbps per input channel, and 20Gbps with two or more input channels. Since PCIe version 3.0 supports 7.7Gbps per lane[54], with support for up to 32 lanes, one achieves much bigger throughput between the data receiving board and host memory. Such rates would also require PCIe version 3.0 from host memory to the RAID controller, and further to the ingest disks. When moving over to the newest version of the PCIe bus, the rates will not be supported by the SAS bus, which is used between the RAID controller and the ingest disks. Therefore, it is suggested that when the PCIe bus is replaced with a newer version, one also changes the ingest disks to NVMe disks which may be connected to the PCIe bus directly. Such a setup will be expensive, but the rates will possibly exceed 20Gbps and beyond.

7

Conclusion

The goal for this thesis was to create a new baseline for the use of next generation HP ProLiant servers, in terms of the ability to capture very high rate data. More specific, the goal was to test whether the system can capture a rate of 10Gbps without data loss. To achieve this goal, I developed an analysis setup, as an overlay over already existing tools, that monitors critical components within the server, which purpose was to identify bottlenecks. The analysis tool developed presents much information to the user about what the computer does on a per second basis. Several tools were gathered within the same terminal window through Tmux. The monitoring programs used are already existing systems available in Linux OS's. The analysis tool evaluation shows some room for improvement, as some important components are not monitored. The experiments performed in chapter 5 show that one can gain information about the components not monitored through isolating parts of the pipeline, in order to fill in information about the non-monitored components.

The analysis tool was used to evaluate the next generation HP ProLiant servers with the configurations used in the current generation HP servers, as used in MEOS capture as of today. The analysis shows that the ingest disks are the bottleneck, and the experiments conducted demonstrate that, with suitable configuration, the next generation HP ProLiant servers support rates above 10Gbps when performing data capture.

The experiments demonstrate that two SSDs in stripe achieved a total rate of 10,1Gbps when performing data capture using two input channels. One

does effectively achieve 5Gbps per input channel, where the maximum rate demonstrated per input channel was 8.7Gbps. When performing both data capture and near-real time processing, I demonstrate significant increase in performance when using SSDs instead of HDDs as ingest disks. The experiments show that with two SSDs one can achieve 3.5Gbps capturing rate and 2.9Gbps reading rate for processing purposes. With HDDs the best setup is by using 8HDDs in RAID 5, providing 2.3Gbps capturing rate and 2.1Gbps reading rate. The changes made to the system are considered low cost changes as many of them only require a new RAID configuration. However, as the RAIDs tested seems to provide poor scalability from 4 to 8 disks, it seems like using 8 HDDs provides more cost than what one gain as the increase in data throughput was not significant at all.

We have also seen through the discussion in section 6.3 that there exists hardware providing far better transfer rate than used in this thesis exist, which is why it was speculated that one could achieve more than 10Gbps per input channel. This would however be an expensive improvement to the system as KSPT must develop a new data receiving board with support for PCIe version 3.0 or newer. It would also require a new set of RAID controllers, and disks used in the ingest RAID.

The work performed through this thesis has been limited to the equipment available in the server provided by KSPT. No further hardware was made available through the thesis, and it is known that there exist hardware with better specifications. Another limitation is obviously that the thesis must be completed within the time constraints of the semester.

Some lessons learned through this project is that it is wise to start early and fail early. One can detect flaws and errors before the “point of no return”, where one do not have time to change them before delivery. Make decisions early, and be faithful to the decisions made. These lessons was put to the test through this thesis, and they made the whole process of completing this thesis much more enjoyable. With this, the thesis is concluded by outlining future work.

7.1 Future Work

The topics I did not have time to explore as much as desired, or at all, are presented. The server still possess a large potential for improvement, and the areas for improvement are addressed in this section.

Exploring the features for supporting large transfer rates in software, such as the file system used would be a natural step towards higher rates, as some file

systems may be more effective in terms of throughput to disk. Many file systems, such as XFS which is used in this thesis, keep track of changes done (also known as journaling) to the file system which generate overhead. Other file systems worth testing are `Btrfs`[59], as it has support for both SSDs[1] and HDDs. It also has RAID support built into the file system. The negative side of `Btrfs`, is that it does a lot of housekeeping which increase the overhead per write. It is however possible that it would increase the overall performance.

When using two input channels, exploring the use of two RAID controllers can be an improvement to the system. Through two RAID controllers one allows two sets of ingest disks each with its own RAID controller, connected with separate PCIe buses. This could increase the performance, as the data from both channels only share the address space in the OS kernel and host memory along the pipeline. It is also expected that one could increase the performance when performing data capture and near-real time processing through the use of two RAID controllers and two input channels. This would decrease the total load generated per controller, which may provide better overall throughput.

The satellite industry faces transmission rates of 10Gbps in near future. At some point KSPT must consider moving their data receiving board over to a newer version of the PCIe bus, because the server supports PCIe version 3.0. The backward compatibility within the PCIe bus enables the use of the data receiving board with PCIe version 2.0. This would require NVMe disks in order to ingest the incoming rate. This is regarded as an expensive solution as the NVMe disks can cost 6.000$ and more[24], and it would require large amount of development to accomplish as one must develop a new set of drivers for the data receiving board.

According to Network Computing[49], HP will introduce a new series of non-volatile RAM. This means data can be stored persistently in RAM even under unexpected events such as the loss of power. This can be used to store a satellite pass in RAM instead of disks as the RAM provide persistent storage. With such memory, one can remove several components from MEOS capture used today. Examples are the ingest disks, RAID controller and SAS controller, which all have been suspected as the bottleneck when the rate increases. Not much are known about such technology yet, as they are not available on the market. One problem is however the amount of provided storage per RAM brick, in addition to the price tag.

The kernel buffer size has not been exploited as much as it should have in this

1. Evenly distribute write accesses across the whole disk in order to increase the expected life time.

thesis. Therefore it would be a natural step to play around with different buffer sizes. It is especially important when performing data capture and near-real time processing, as the kernel buffer space is temporary storage before it is written to disk. It is expected that using a larger buffer size can reduce the risk of buffer overflows, creating a system much more tolerant against drops in disk writing performance as we have seen through the experiments.

A

Test Result Figures

A.1 8 HDDs Striped

A.1.1 Data Capture and Near Real-time Processing

The figures presented in this section were generated by using 8 HDDs as ingest disks, when performing both data capture and near real-time processing. The experiment is presented in its entirety in section 6.1.2, paragraph `Capture and Process`.

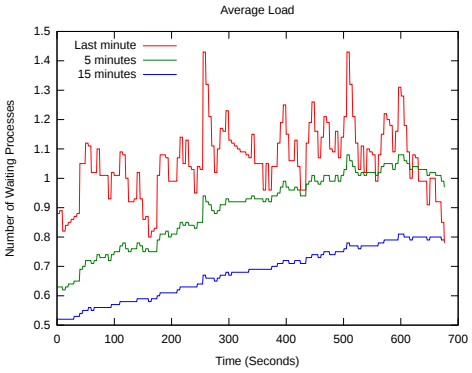


Figure A.1: The figure shows the average load during the experiment. The average load is the average number of queued processes, where 1, 5 and 15 minutes representing the integration time. 1 minute seems much more unstable than 5 and 15 minutes, because of the integration time length.

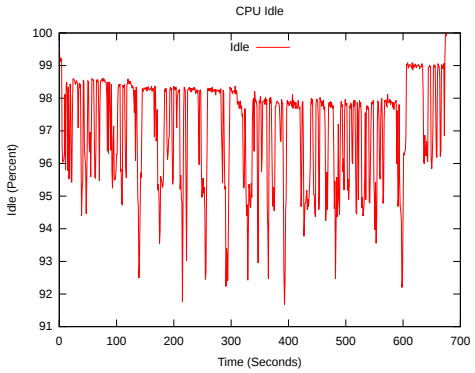


Figure A.2: The figure shows the amount of time the CPU has spent idle through the experiment.

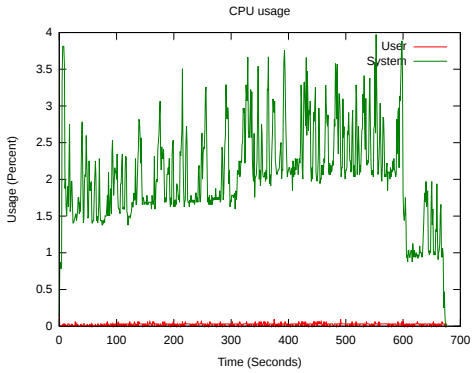


Figure A.3: The figure shows the amount of time the CPU has spent executing user level and system level processes through the experiment.

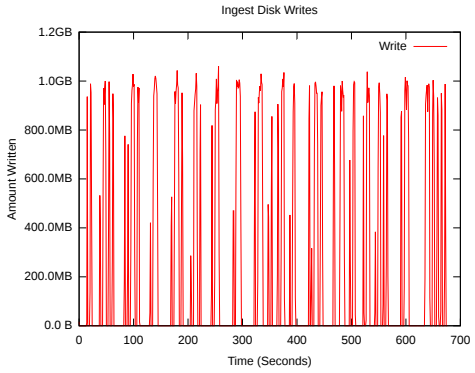


Figure A.4: The figure shows the ingest disks write activity through the experiment.

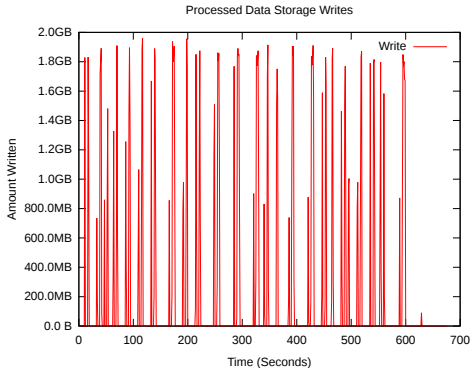


Figure A.5: The figure shows the write accesses to the disks used to store processed data.

A.2 RAID 5

A.2.1 Four HDDs Performing Data Capture

The figures presented in this section show the CPU usage and average load as four HDDs were used as ingest disks when performing data capture. The experiment is presented in its entirety in section 6.1.3, paragraph `Capture`.

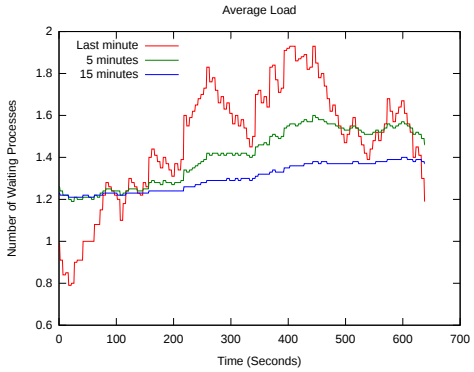


Figure A.6: The figure shows the average load (queued processes) through this experiment.

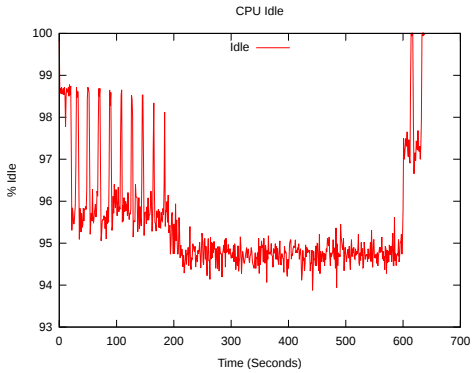


Figure A.7: The figure shows the amount of time, in percent, the CPU have spent idle through the experiment.

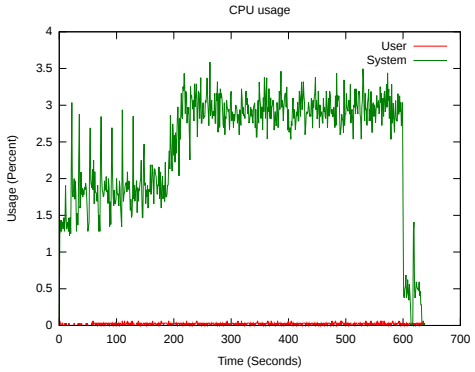


Figure A.8: The figure shows the amount of time the CPU have spent on user- and system-level processes during the experiment.

A.2.2 Four HDDs As Ingest Disks With Near Real Time Processing Enabled

The figures presented below were generated through performing data ingestion and near real-time processing with 4HDDs organized in RAID 5 as ingest disks. The experiment is presented in its entirety in section 6.1.3, paragraphs `Capture and Processing`.

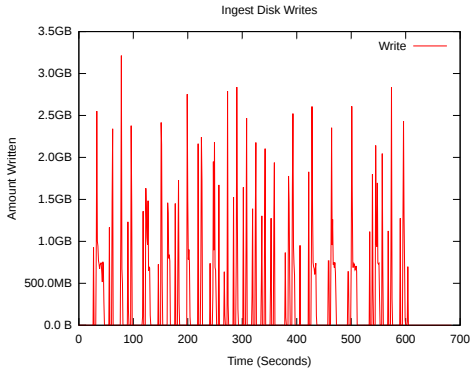


Figure A.9: The figure shows the disk activity when using 4 HDDs in RAID 5 during both capture and processing.

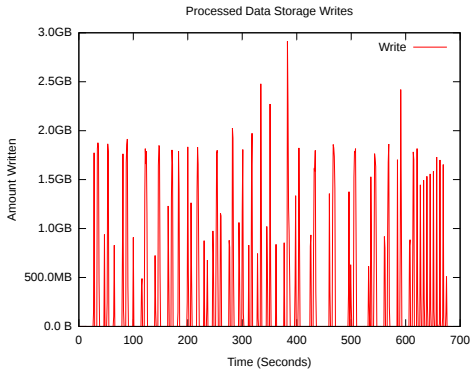


Figure A.10: The figure shows the disk activity representing the RAID that stores the processed data.

A.2.3 8 HDDs As Ingest Disks Performing Capture Only

The figures presented were generated by performing data ingest only with 8 HDDs configured to RAID 5 as ingest disks. The experiment is presented in its entirety in section 6.1.4, paragraph `Capture`.

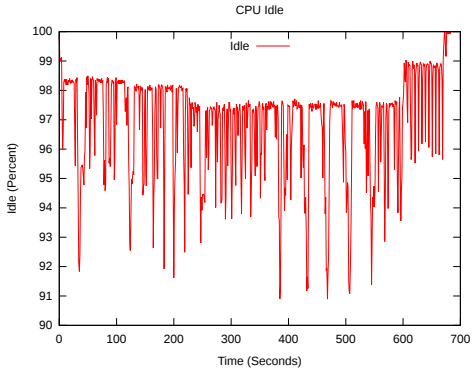


Figure A.11: The figure shows the amount of time the CPU has spent idle during the experiment.

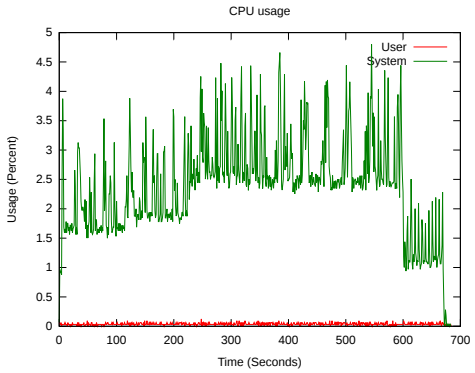


Figure A.12: The figure shows the amount of time the CPU have spent on user and system level processes during the experiment.

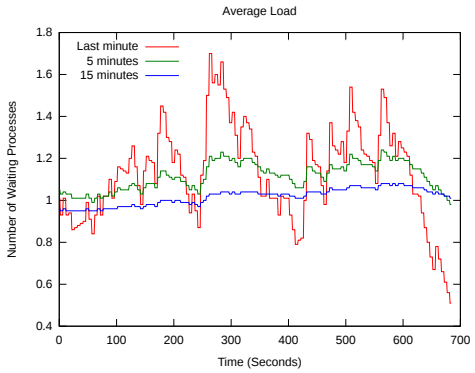


Figure A.13: The figure shows how many processes have been queued on average, to run on the CPU during the experiments.

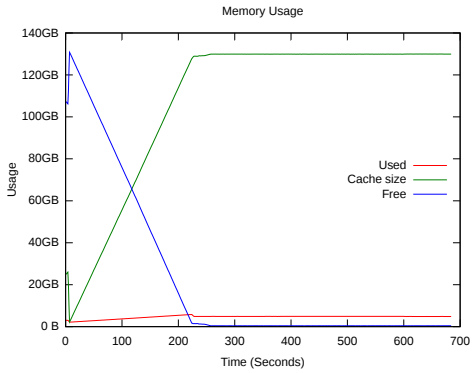


Figure A.14: The figure shows the memory usage through this experiment.

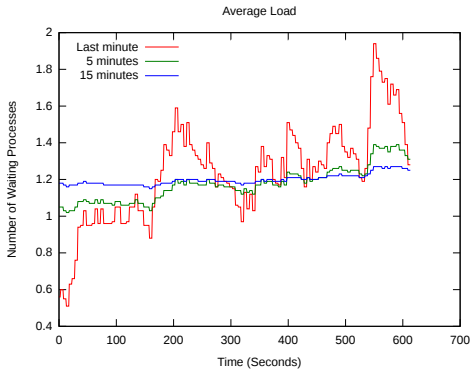


Figure A.15: The figure shows the average load for this experiment.

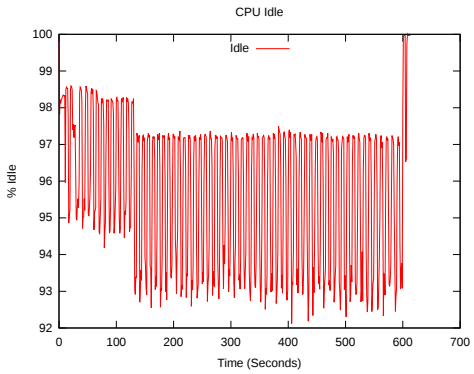


Figure A.16: The figure shows the amount of time the CPU has spent idle through the experiment. The figure suggest that it has been idling for more than 90% of the execution time.

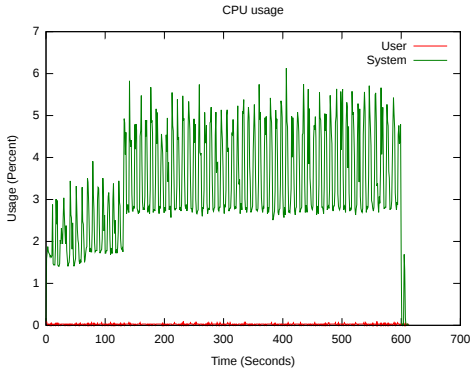


Figure A.17: The figure shows the amount of time the CPU has spent on user and system level processes during the experiment.

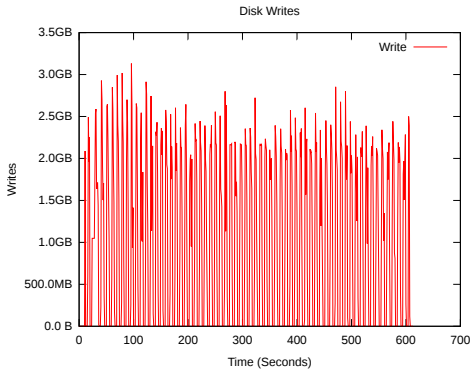


Figure A.18: The figure shows the activity in which the total RAID has performed through this experiment.

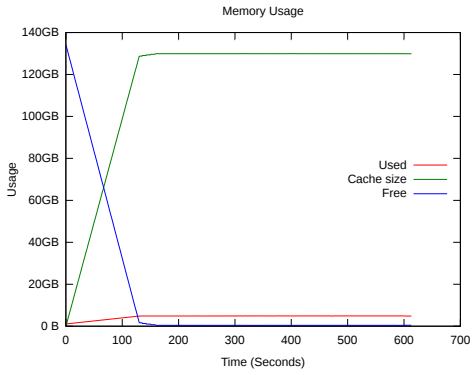


Figure A.19: The figure shows the memory usage through this experiment. Note that the usage is at its maximum from about 150 seconds.

A.2.4 8 HDDs As Ingest Disks Performing Capture and Near Real-time Processing

The runtime statistics presented were generated by performing data capture in addition to near real-time processing. The ingest disks through this experiment were 8 HDDs configured in RAID 5. The experiment is presented in its entirety in section 6.1.4, paragraph `Capture and Processing`.

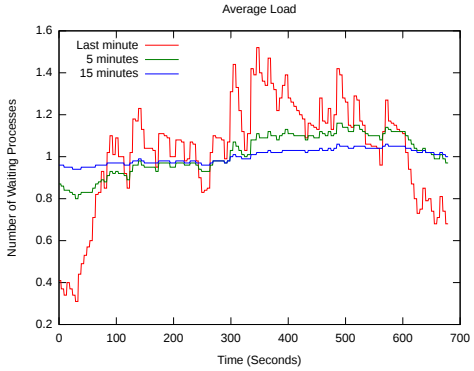


Figure A.20: The figure shows the average load through the experiment.

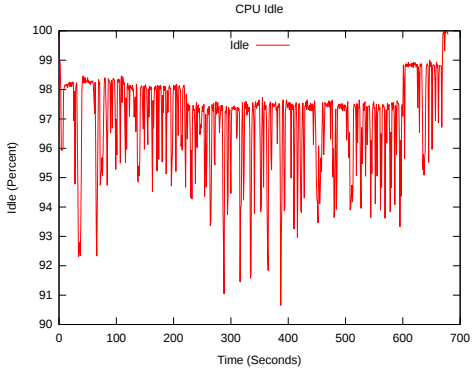


Figure A.21: The figure shows the amount of time the CPU spent idling through the experiment. Not surprisingly it stay above 90%.

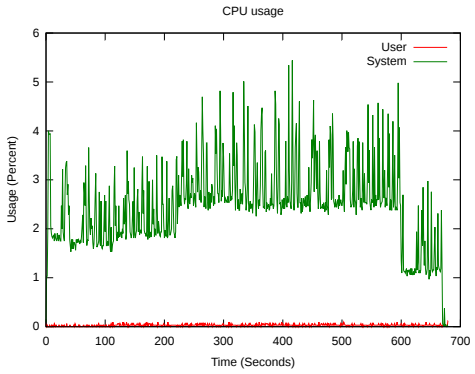


Figure A.22: The figure shows the amount of time the CPU spent on user and system level processes.

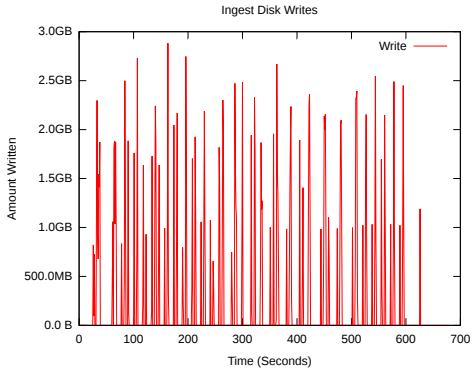


Figure A.23: The figure shows the ingest disk write activity. The spiking behaviour is caused by the combination of low rates and that file system caching is enabled.

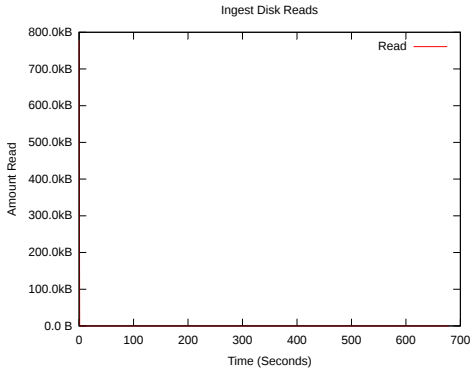


Figure A.24: The figure show the ingests disk reading activity. As with previous experiments, all read accesses fetch data from some cache, resulting in absolutely no read accesses from the physical RAID.

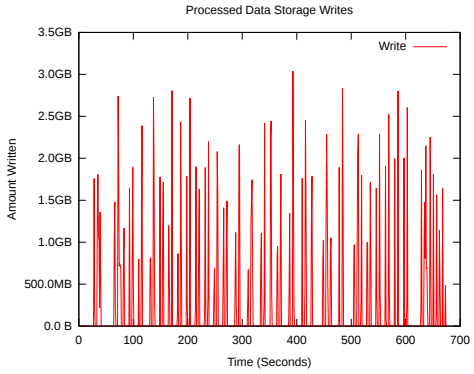


Figure A.25: The figure shows the processing storage activity. These disks are only written to, as their mission is to store processed data.

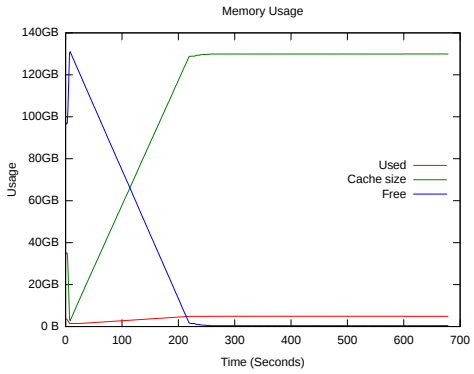


Figure A.26: The figure shows the memory usage through this experiment. The usage peaks after about 220 seconds, and show a stable behaviour throughout the experiment.

B

Analysis Tool Script

The script used as analysis tool are presented below. The script will not create a new session for each user who launches the tool, for performance purposes. If a Tmux session already exists, new users will just attach themselves to the existing one, instead of creating a new session.

Listing B.1: The code used to launch the analysis tool

```
#!/bin/bash

declare -A monitors
monitors[dstat]="dstat -tcmlrd -D sdc --noheaders --
    output dstat_750.dat"
monitors[klog]="tail -f /var/log/messages"
monitors[htop]="htop"

# If session already exist, just attach it
if ! tmux list-sessions 2>/dev/null | grep '^monitor:'
    &>/dev/null; then
        tmux new -d -n main -s monitor "${monitors[
            dstat]}"
        tmux select-pane -t monitor
        tmux split -h "${monitors[htop]}"
        tmux split -v "${monitors[klog]}"
fi
```

```
tmux attach -t monitor
```

Bibliography

[1] dstat(1) — linux man page. http://linux.die.net/man/1/dstat. [Online; Accessed 8-April-2016].

[2] Nice — Linux Man Page. http://linux.die.net/man/2/nice. [Online; Accessed 22-February-2016].

[3] pv(1) — Linux man page. http://linux.die.net/man/1/pv. [Online; Accessed 16-March-2016].

[4] Antwerp Space. Omnisat. http://www.antwerpspace.be/sites/default/files/products/as-spe-110234-05-00-omnisat-g3_leaflet.pdf. [Online; Accessed 29-April-2016].

[5] Piermario Besso, Maurizio Bozzi, Luca Perregrini, Luca Salghetti Drioli, and Will Nickerson. Deep-space antenna for rosetta mission: Design and testing of the s/x band dichroic mirror. *Antennas and Propagation, IEEE Transactions on*, 51(3):388–394, 2003.

[6] Stephen D Brown, Robert J Francis, Jonathan Rose, and Zvonko G Vranesic. *Field-programmable gate arrays*, volume 180. Springer Science & Business Media, 2012.

[7] Business Dictionary. Bottleneck. http://www.businessdictionary.com/definition/bottleneck.html. [Online; Accessed 7-March-2016].

[8] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, et al. Windows azure storage: a highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 143–157. ACM, 2011.

[9] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. Raid: High-performance, reliable secondary storage. *ACM Comput. Surv.*, 26(2):145–185, June 1994.

[10] Prakash Chitre and Ferit Yegenoglu. Next-generation satellite networks: architectures and implementations. *Communications Magazine, IEEE*, 37(3):30–36, 1999.

[11] Vladimir A Chobotov. *Orbital mechanics*. Aiaa, 2002.

[12] Guy Cordier, Carlos Gomez-Rosa, Gene Knoble, and Fred McCaleb. Autonomous multi-missions, multi-sites x-band data capture systems. In *SpaceOps 2006 Conference*, page 5688, 2006.

[13] Mark Dery. *Escape velocity: cyberculture at the end of the century*. Grove Press, 1996.

[14] Dictionary.com, LLC. big data. `http://dictionary.reference.com/browse/big-data`. [Online; Accessed 4-March-2016].

[15] Dictionary.com, LLC. Murphy's Law. `http://www.dictionary.com/browse/murphy-s-law`. [Online; Accessed 28-April-2016].

[16] P Dieleman, W Luinge, ND Whyborn, D Teyssier, JW Kooi, WM Laauwen, and MWM de Graauw. Hifi flight model testing at instrument and satellite level. In *Proceedings 19th International Symposium on Space Terahertz Technology*, pages 106–110, 2008.

[17] Kristian Elsebø. Improving disk performance in vortex with nvme. 2015.

[18] Mustafa Eroz, Feng-Wen Sun, and Lin-Nan Lee. Dvb-s2 low density parity check codes with near shannon limit performance. *International Journal of Satellite Communications and Networking*, 22(3):269–279, 2004.

[19] European Space Components Coordination (ESCC). `https://escies.org/webdocument/showArticle?id=119`. [Online; Accessed 18-February-2016].

[20] European Space Components Coordination (ESCC). `https://escies.org/webdocument/showArticle?id=727&groupid=6`. [Online; Accessed 18-February-2016].

[21] Tomoya Fukami. Experiment of 348 mbps downlink from 50-kg class satellite. In *Proc. Small Satellites for Earth Observation, 10th International Symposium of the International Academy of Astronautics (IAA), IAA-B10-1302, Berlin, Germany*, 2015.

[22] M.J. Gans, M.V. Schneider, and R.F. Trambarulo. Qpsk modulator or

demodulator using subharmonic pump carrier signals, September 16 1986. US Patent 4,612,518.

[23] Lajos Hanzo, Soon Xin Ng, WT Webb, and T Keller. Quadrature amplitude modulation: From basics to adaptive trellis-coded, turbo-equalised and space-time coded ofdm, cdma and mc-cdma systems. 2004.

[24] Hewlett Packard. HP 800GB NVMe Write Intensive HH/HL PCIe Workload Accelerator. `http://www8.hp.com/us/en/products/accelerators/product-detail.html?oid=7876058#!tab=features`. [Online; Accessed 25-May-2016].

[25] HP. QuickSpecks, HPE Solid State Drives (SSD). `http://www8.hp.com/h20195/v2/GetPDF.aspx%2Fc04154378.pdf`. [Online; Accessed 19-February-2016].

[26] Yiming Hu and Qing Yang. Dcd—disk caching disk: A new approach for boosting i/o performance. *SIGARCH Comput. Archit. News*, 24(2):169–178, May 1996.

[27] Hai Huang, Wanda Hung, and Kang G Shin. Fs2: dynamic data replication in free disk space for improving disk performance and energy consumption. In *ACM SIGOPS Operating Systems Review*, volume 39, pages 263–276. ACM, 2005.

[28] Ji Jun Hung, Kai Bu, Zhao Lin Sun, Jie Tao Diao, and Jian Bin Liu. Pci express-based nvme solid state disk. In *Applied Mechanics and Materials*, volume 464, pages 365–368. Trans Tech Publ, 2014.

[29] Kai Hwang, Hai Jin, and Roy Ho. Raid-x: A new distributed disk array for i/o-centric cluster computing. In *High-Performance Distributed Computing, 2000. Proceedings. The Ninth International Symposium on*, pages 279–286. IEEE, 2000.

[30] James R Irons, John L Dwyer, and Julia A Barsi. The next landsat satellite: The landsat data continuity mission. *Remote Sensing of Environment*, 122:11–21, 2012.

[31] Mohammed Safiqul Islam, Gouri Rani Barai, and Atiq Mahmood. Performance analysis of different modulation schemes using ofdm techniques in rayleigh fading channel. *International journal of fundamental Physical sciences*, 1(1):22–27, 2011.

[32] A. Jamalipour and T. Tung. The role of satellites in global it: trends and

implications. *IEEE Personal Communications*, 8(3):5–11, Jun 2001.

[33] Byunghei Jun and Dongkun Shin. Workload-aware budget compensation scheduling for nvme solid state drives. In *Non-Volatile Memory System and Applications Symposium (NVMSA), 2015 IEEE*, pages 1–6, Aug 2015.

[34] VK Kaila, AS Kirankumar, TK Sundaramurthy, S Ramakrishnan, MYS Prasad, Pranav S Desai, V Jayaraman, and B Manikiam. Metsat-a unique mission for weather and climate. *CURRENT SCIENCE-BANGALORE-*, 83(9):1081–1087, 2002.

[35] Kongsberg Spacetec AS. MEOS Antenna. `http://www.spacetec.no/pdf/meos-antenna-l-s-x-band-antenna`. [Online; Accessed 11-March-2016].

[36] Kongsberg Spacetec AS. MEOS Capture HRDFEP v4 High Rate Demodulator & Front End Processor. `http://www.spacetec.no/pdf/meos-capture-hrdfep-v4`. [Online; Accessed 22-February-2016].

[37] M.J. Koop, Wei Huang, K. Gopalakrishnan, and D.K. Panda. Performance analysis and evaluation of pcie 2.0 and quad-data rate infiniband. In *High Performance Interconnects, 2008. HOTI '08. 16th IEEE Symposium on*, pages 85–92, Aug 2008.

[38] Åge Andre Kvalnes. The omni-kernel architecture: Scheduler control over all resource consumption in multi-core computing systems. 2014.

[39] Jiyang Liu, Liang Zhu, Weiqiang Sun, and Weisheng Hu. Scalable application-aware resource management in software defined networking. In *Transparent Optical Networks (ICTON), 2015 17th International Conference on*, pages 1–5, July 2015.

[40] Lou Frenzel — Electronic Design. `http://electronicdesign.com/communications/understanding-modern-digital-modulation-techniques`, January 2012. [Online; Accessed 28-January-2016].

[41] Thomas R Loveland and John L Dwyer. Landsat: Building a strong future. *Remote Sensing of Environment*, 122:22–29, 2012.

[42] Dongzhe Ma, Jianhua Feng, and Guoliang Li. A survey of address translation technologies for flash memories. *ACM Comput. Surv.*, 46(3):36:1–36:39, January 2014.

[43] Bruce McLemore, Guy R Cordier, Terri Wood, and Hårek Gamst. Edos data capture for alos. 2012.

[44] Micron. http://d1.amobbs.com/bbs_upload782111/files_46/ourdev_684405XFKZAB.pdf. [Online; Accessed 27-January-2016].

[45] Sangwhan Moon, Jaehwan Lee, Xiling Sun, and Yang-suk Kee. Optimizing the hadoop mapreduce framework with high-performance storage devices. *The Journal of Supercomputing*, 71(9):3525–3548, 2015.

[46] Alberto Morello and Vittoria Mignone. Dvb-s2: The second generation standard for satellite broad-band services. *Proceedings of the IEEE*, 94(1):210–227, 2006.

[47] W. Najjar and J.-L. Gaudiot. Network resilience: a measure of network fault tolerance. *Computers, IEEE Transactions on*, 39(2):174–181, Feb 1990.

[48] National Aeronautics and Space Administration. Sputnik 1. http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=1957-001B. [Online; Accessed 19-January-2016].

[49] Network Computing — Howard Marks. Welcome To The Persistent Memory Era. http://www.networkcomputing.com/storage/welcome-persistent-memory-era/232589253?_mc=NL_NWC_EDT_NWC_converations_20160419&cid=NL_NWC_EDT_NWC_converations_20160419&elqTrackId=de463c549cbc4838be8827edf4e889ae&elq=ae997492d7cf4b39a3affac8b6abde66&elqaid=69205&elqat=1&elqCampaignId=20669. [Online; Accessed 18-May-2016].

[50] Woo H Paik, Scott A Lery, and John M Fox. Mode selective quadrature amplitude modulation communication system, November 8 1994. US Patent 5,363,408.

[51] PC Mag. Definition of: FPGA. http://www.pcmag.com/encyclopedia/term/43445/fpga. [Online; Accessed 25-February-2016].

[52] PC Mag. Definition of: static RAM. http://www.pcmag.com/encyclopedia/term/52041/static-ram. [Online; Accessed 25-February-2016].

[53] PCI-SIG. PCI Express Base Specification, Revision 2.1. http://iu4ever.org/files/Skhemotekhnika_JEVS9/1697_Skhemotekhnika_JEVS9_PCI_Express_Base_r2_1.pdf, 03 2009. [Online; Accessed 22-February-2016].

[54] PCI-SIG. PCI Express Base Specification Revision 3.0. http://composter.com.ua/documents/PCI_Express_Base_Specification_Revision_3.0.pdf, November 2010. [Online; Accessed 22-February-

2016].

[55] R.L. Pease, A.H. Johnston, and Joseph L. Azarewicz. Radiation testing of semiconductor devices for space electronics. *Proceedings of the IEEE*, 76(11):1510–1526, Nov 1988.

[56] B Pillans, S Eshelman, A Malczewski, J Ehmke, and C Goldsmith. Ka-band rf mems phase shifters. *Microwave and Guided Wave Letters, IEEE*, 9(12):520–522, 1999.

[57] PD Potter. S-and x-band rf feed system. *The Deep Space Network Progress Report*, pages 53–60, 1972.

[58] Rex Farrance, PCWorld. Timeline: 50 Years of Hard Drives. `http://www.pcworld.com/article/127105/article.html`. [Online; Accessed 4-March-2016].

[59] Ohad Rodeh, Josef Bacik, and Chris Mason. Btrfs: The linux b-tree filesystem. *ACM Transactions on Storage (TOS)*, 9(3):9, 2013.

[60] RT Logic. T1200HDR High Data Rate Receiver. `http://www.rtlogic.com/~/media/datasheets/rtl-dst_t1200hdr.pdf`. [Online; Accessed 29-April-2016].

[61] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *Computer*, 27(3):17–28, 1994.

[62] Samara Lynn. RAID Levels Explained. *PCmag Digital Group*, March 27th 2014. [Online; Accessed 22-January-2016].

[63] H.B. Sanderford. Binary phase shift keying modulation system and/or frequency multiplier, June 23 1998. US Patent RE35,829.

[64] Seagate. Serial Attached SCSI (SAS) Product Manual, Revision C. `http://www.seagate.com/staticfiles/support/disc/manuals/Interface%20manuals/100293071c.pdf`, December 2009. [Online; Accessed 22-February-2016].

[65] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2007.

[66] Teledyne LeCroy. Summit T24 Analyzer. `http://teledynelecroy.com/protocolanalyzer/protocoloverview.aspx?seriesid=445&capid=103&mid=511`. [Online; Accessed 20-April-2016].

[67] The Consultative Committee for Space Data Systems, CCSDS. Flexible Advanced Coding and Modulation Scheme for High Rate Telemetry Applications, 2012.

[68] B. Tierney, E. Kissel, M. Swany, and E. Pouyoul. Efficient data transfer protocols for big data. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pages 1–9, Oct 2012.

[69] Chai-Keong Toh and Victor OK Li. Satellite atm network architectures: an overview. *IEEE network*, 12(5):61–71, 1998.

[70] ViaSat. ViaSat High Rate Receiver 3200. `https://www.viasat.com/sites/default/files/legacy/High_rate_modem_3200_Datasheet_005_web.pdf`. [Online; Accessed 29-April-2016].

[71] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[72] Stephen B Wicker and Vijay K Bhargava. *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[73] Zodiac Aerospace. Cortex HDR XXL – High Data Rate Receiver. `http://www.zodiacaerospace.com/en/cortex-hdr-xxl-high-data-rate-receiver`. [Online; Accessed 29-April-2016].