UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Rapid, distributed processing of medium velocity datastreams into contextualised 3D visualisations: Evaluating the cloud computing paradigm

by

Lauren Hankel

Submitted in partial fulfilment of the requirements
for the degree

MSc Geoinformatics

in the

Faculty of Natural and Agricultural Sciences

University of Pretoria

Submitted on 4 December 2014

Resubmitted on 28 April 2015

Declaration

I, **Lauren Hankel,** declare that the dissertation, which I hereby submit for the degree **MSc Geoinformatics** at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.

SIGNATURE	DATE

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Rapid, distributed processing of medium velocity datastreams into contextualised 3D visualisations: evaluating the cloud computing paradigm

Lauren Hankel

Supervisor: Professor Serena Coetzee	Department: Geography, Geoinformatics and Meteorology
Co-Supervisor: Mr Graeme McFerren (CSIR)	Faculty: Natural and Agricultural Sciences
Degree: MSc Geoinformatics	University: University of Pretoria

Abstract

Vegetation wildfires occur in most parts of the world, sometimes with disastrous effects. Wildfire incidents could be mitigated if timeous and informative notifications are disseminated to appropriate parties. Hundreds of detections might be obtained from an intermittent data stream of active fire data, detected from earth observation satellites and might be disseminated to hundreds of parties at any satellite overpass. Ideally, when an interested or affected party receives a wildfire notification, the receiver should immediately be able to link to the visualisation resource via a web connected device, i.e. the visualisation should be available on demand or in 'demand-time'. To achieve demand-time results, datastreams of wildfire events need to be processed rapidly in relation to large datasets of contextual variables. Failure to do so would result in processing backlogs and cause a delay in the generation of three-dimensional (3D) visualisations. The primary purpose of this research was to evaluate the efficiency of different algorithmic and architectural styles for process chaining in the cloud to generate 3D spatial context visualisations around detected active fires in demand-time. This study investigated efficiencies across four dimensions: 1) software libraries, 2) tightly-coupled and serial versus loosely-coupled and distributed geoprocessing chain implementations, 3) standardised geoprocessing web service (OGC WPS) implementations versus bespoke software solutions and 4) system deployment in a cloud versus system deployment on a single thread on commodity hardware. Geoprocessing chains were implemented in Python using open-source libraries and frameworks. Results indicate that loading objects in memory by using a software library is more efficient than using a spatial database. Loosely-coupled distributed geoprocessing is faster than tightly-coupled serial geoprocessing. Web Processing Services cause a significant deterioration in the performance of the geoprocessing chain. Overall, geoprocessing with Web Processing Services on a single thread on commodity hardware does not deliver demand-time results. Demand-time could only be achieved with bespoke software solutions or with a larger number of cloud instances that was not cost effective. The cloud computing paradigm can be evaluated due to free or less costly options. As the number of cloud instances increased, the performance of the geoprocessing chain improved. Therefore, demand-time results can be achieved when using the optimal number of cloud instances to conduct geoprocessing. However, there is a trade-off between the number and/or size of instances and costs.

Acknowledgements

I would like to thank my supervisor Professor Serena Coetzee and my co-supervisor Mr Graeme McFerren.I sincerely appreciate the guidance that you have provided me throughout this experience. Your advice was invaluable. It was a honour to learn so much from you.

Professor Serena Coetzee, thank you for all the time that you have invested into guiding me with the write-up. Thank you for taking the time to thoroughly review draft versions of the dissertation. I value all the comments and I appreciate all the advice you have given me.

Mr Graeme McFerren, thank you for supplying resources required to set up the public cloud infrastructure. I would like to thank you for introducing me to all the technologies experimented with. I have gained a new body of knowledge and capabilities. Thank you for all of the advice you have given me and thank you for your patience.

Mr Uli Horn, although the OpenNebula private cloud was not used for one of the experiments, I do appreciate all the time that you have spent on helping me to set up machines in the private cloud. Experimenting with a private cloud was beneficial for future research. It was nice to learn from you.

Mr Sives Govender, thank you for helping me acquire resources required to set up the public cloud infrastructure. Thank you for the support you have given me.

Mr Riaan van den Dool, thank you for all of your support and interest in this research.

The management and staff at the CSIR who granted me the opportunity to complete my MSc, I feel extremely privileged to have learnt so much from such remarkable people. Thank you for funding the MSc and the trip to the 11th International Symposium on Location-Based Services in Vienna, Austria on 26-28 November 2014.

Finally, I would like to thank my mom, dad, sister and other members of the family for their love and support. I would not have come this far without your encouragement. Thank you for funding some of the cloud resources.

Herewith acknowledgement to my editor, Ms Liza Marx from Aqdemic Editing Services.

List of Abbreviations and Acronyms

Abbreviation or Acronym	Meaning
3D	Three Dimensional
AFIS	Advanced Fire Information System
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
AWS	Amazon Web Services
BPEL	Business Process Execution Language
CPU	Central Processing Unit
CSIR	Council for Scientific and Industrial Research
DEM	Digital Elevation Model
EBS	Elastic Block Store
EC2	Elastic Compute Cloud
ESA	European Space Agency
ESKOM	Electricity Supply Commission
GeoTIFF	Geo Tagged Image File Format
GML	Geography Markup Language
GPU	Graphics Processing Unit
HTML	Hypertext Transfer Markup Language
IaaA	Infrastructure-as-a-Service
IP	Internet Protocol
IT	Information Technology
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
KVP	Key Value Pair
MODIS	Moderate Resolution Imaging Spectroradiometer
MSG	Meteosat Second Generation
NASA	National Aeronautics and Space Administration
NLC	National Land Cover
OCG	Open Geospatial Consortium
PaaS	Platform-as-a-Service
PC	Personal Computer
RAM	Random Access Memory
RDS	Relational Database Service
ROM	Read-Only Memory
S3	Simple Storage Service
SaaS	Software-as-a-Service
SCP	Secure Copy
SEVIRI	Spinning Enhanced Visible & Infrared Imager
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRTM	NASA Shuttle Radar Topography Mission
SSH	Secure Shell

Abbreviation or Acronym	**Meaning**
URL	Uniform Resource Locator
WCS	Web Coverage Service
WFS	Web Feature Service
WKT	Well Known Text
WMS	Web Map Service
WPS	Web Processing Service
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language

Table of Contents

List of Figures

List of Tables

1 Chapter One: Introduction

1.1 Chapter Overview

This chapter provides an extensive, high level overview of the problem addressed in this research. This chapter puts the research problem into context by providing background information. The research problem statement, research question and research objectives are included to give a distinct description of the problem that the research attempted to solve and the objectives that had to be achieved to solve the problem. The research significance is included to highlight the research importance, and limitations are further included to provide detail on the constraints of this research and the challenges faced during this research. This chapter concludes with an overview of the remaining chapters in the dissertation.

1.2 Background

Vegetation wildfires occur frequently in many parts of the world. In South Africa, vegetation wildfires are well-known for being used as a land management tool. Wildfires have the potential to get out of control fast and destroy valuable resources (Davies *et al*., 2008). They can range from small scale fires that cause an insignificant quantity of damage (minimal damage to property) to large scale wildfires that can cause a considerable quantity of damage (complete destruction of property or loss of life). Wildfire incidents can be mitigated by the expeditious dissemination of information derived from earth observation data. This information allows a fire manager to gain early insight into wildfire incidents.

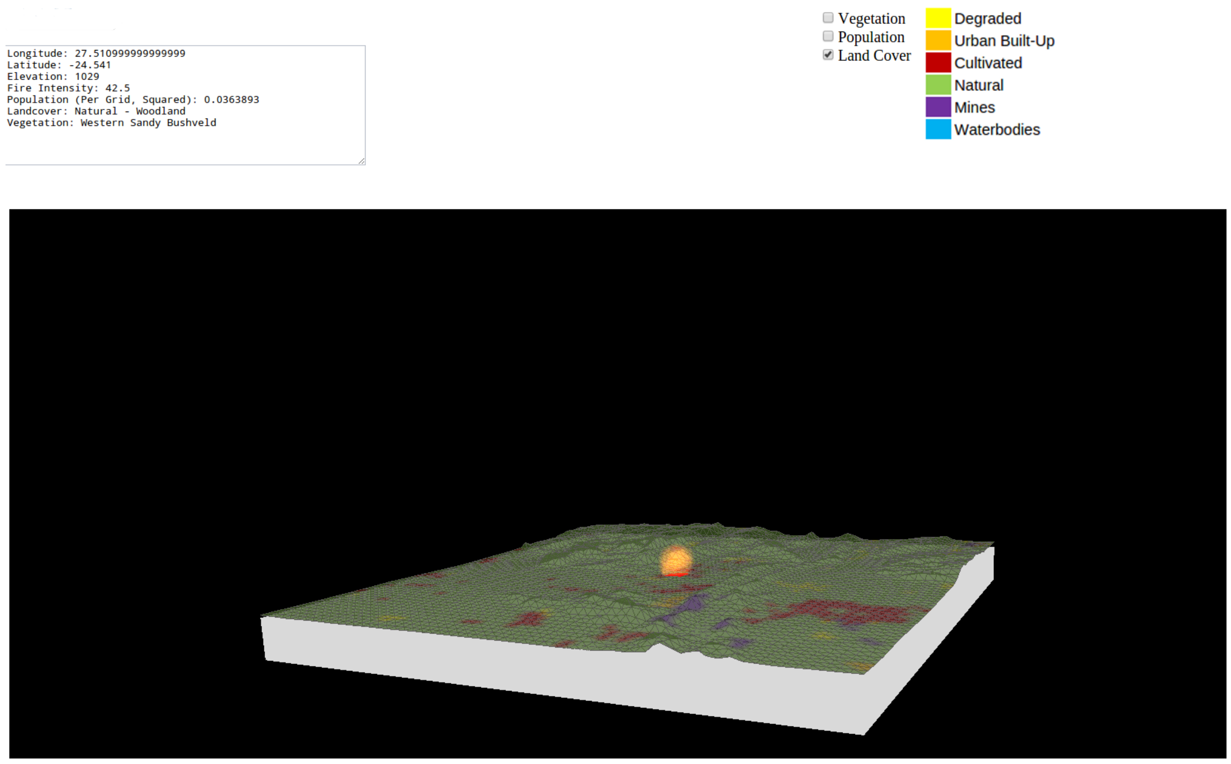
Kassab *et al*. (2010) stated that geographical data describes the spatial and descriptive semantics of fires and their surroundings. Situational awareness is described as information utilised to aid in the assessment and management of a situation (Vieweg *et al*., 2010). Situational awareness in a wildfire situation can be enhanced if appropriate parties are alerted of wildfire incidents in real-time and therefore wildfire notifications should be disseminated in real-time.

The CSIR's Meraka Institute developed the Advanced Fire Information System (AFIS) with the primary function to provide real-time information of active vegetation wildfires detected from earth observation satellite data (CSIR, 2012). This industry use case (AFIS) was used as inspiration for the research discussed in this dissertation.

AFIS provides users with prediction, detection, monitoring, alerting, and planning capabilities. All these capabilities were created through the use of earth observation satellites, weather models and information communication technologies (CSIR, 2012). The web-based mapping service further offers the above-mentioned capabilities. The AFIS mobile application was recently introduced for Apple iOS and Google Android mobile operating systems. It provides users with similar capabilities offered by the web-based mapping service (CSIR, 2012).

AFIS users receive wildfire notifications if wildfires occur nearby locations that users have a specific interest in (McFerren *et al*., 2009). This is achieved through using spatial filters. Wildfire notifications are disseminated as an SMS (short message service) or an e-mail (electronic mail) containing a limited quantity of static information, such as the location and intensity (FRP – Fire Radiative Power) of the detected fire as noted in McFerren *et al*. (2013). The next iteration of wildfire alerting payloads are interactive maps associated with wildfire notifications. Three-dimensional interactive maps are not included in the AFIS alerting payloads yet, but they would significantly enhance situational awareness.

3D spatial context visualisations provide more significant and useful information than 2D spatial context visualisations in the situation of a wildfire. This can be attributed to topography (layout of the land) being viewed in three dimensions. This information maybe vital information required to evaluate a wildfire situation. 3D wildfire context visualisations allows firefighters or emergency responders to visualise the terrain that they have to deal with when responding to the fire. Certain fires might not be reached if firefighters use their fire trucks (fires on top of a steep hill), therefore the firefighters might need a helicopter or another type of vehicle. 3D wildfire context visualisations may influence a user's decision on what is the safest route to follow when caught up in a wildfire situation. This might save lives.



```
Longitude: 27.510999999999999
Latitude: -24.541
Elevation: 1029
Fire Intensity: 42.5
Population (Per Grid, Squared): 0.0363893
Landcover: Natural - Woodland
Vegetation: Western Sandy Bushveld
```

Figure 1: Example 3D Visualisation

The motivation for this research was to develop an alerting component that adds a 3D spatial context visualisation of a detected wildfire to a notification message. Notifications are disseminated in near real-time. A Uniform Resource Locater (URL) is attached to a notification message that binds the 3D visualisation to the notification message. The spatial context referred to above, includes variables such as topography, vegetation types and condition, population density and land cover. These variables are referred to as geospatial data. A user is required to open the URL to view the 3D context visualisation message. *Figure 1* provides an example of a 3D wildfire contextual visualisation.

For this research, the term "demand-time" will be used with generating 3D context visualisations. It is required that when an interested or affected party receives a wildfire notification, the receiver should instantly link to the visualisation resource via a web connected device. This implies that the visualisation should be available on demand (generated rapidly, in “demand-time”). Demand-time is almost similar to real-time. Results or outgoing data should be distributed or disseminated approximately at the same rate of the incoming data. The fundamental difference is that the data has to be prepared (ready) when required. To achieve demand-time results (rapidly disseminate a wildfire notification that contains a link to a prepared 3D visualisation), datastreams of wildfire events need to be processed rapidly in relation to large datasets of contextual variables. Failure to do so would result in processing backlogs and unavailability of 3D visualisations in “demand-time”.

Datastreams of wildfire events are characterised as bursty streams of data. Bursty streams is the intermittent arrival of groups of data. Wildfire data does not stream at a constant rate. Thousands of wildfire event data messages stream for short periods of time and stop to stream at other periods. This can be illustrated as a stream heavily flooded with wildfire event data at specific periods and dried up at other periods in time. These datastreams may require processing by powerful computing resources. It is costly and inefficient to own these resources when they are seldom performing processing operations. Under these circumstances, it is viable to resort to the use of public cloud computing resources, for they require no upfront investment and can be accessed on a utility basis.

Cloud computing platforms offer significant advantages, but the most important advantage is elasticity. Cloud computing instances can be added or removed according to demand. For example, by taking into account the number of wildfire events in a datastream (equal to the number of wildfire event messages in the messaging queues). Adding cloud instances (nodes or virtual machines) will improve (shorten) the processing time of wildfire events as processing tasks will be divided amongst multiple cloud instances. Removing cloud instances by taking the wildfire event message loads (demand) into consideration will save costs as cloud instance use is billed by the hour. This means that cloud instances (nodes or virtual machines) will run when they are required. It is advantageous to know the peak fire event times of the day. Cloud instances (nodes or virtual machines) require time to boot, thus, by having prior knowledge of when fires occur is imperative for planning and auto-scaling. Auto-scaling is the automatic addition or removal of cloud instances according to demand. Messaging queues can be monitored in order to scale automatically. The scaling referred to is horizontal scaling.

Web Processing Services facilitates the discovery and publishing of geospatial processes (Open Geospatial Consortium, 2007). There are several benefits that can be gained when utilising Web Processing Services to conduct geoprocessing, for instance, Web Processing Services are interoperable, reusable, distributed and operating system independent. The benefits makes it an attractive option for a geoprocessing system.

The design of the system should be planned to the finest detail for the geoprocessing system to produce results in demand-time. The algorithmic style of each process should be carefully planned as it can have a great influence on the performance of a method. Algorithmic style refers to how the method should be constructed (loading an object from memory versus loading an object in a database).The time to conduct geoprocessing is influenced by the architectural style of the geoprocessing chain. Loose-coupling of geoprocessing chain components distributes the producers and consumers so that fire events can be processed in parallel (divide and conquer approach). Tight-coupling components of the geoprocessing chain means that nested method or function calls are used. Fire events are therefore processed in a serial style and thus one fire event must move through the entire geoprocessing chain before the next fire event can be processed.

Twitter receives 5700 tweets (messages) per second (Raffi, 2013). Twitter data can be viewed as high velocity data because of the fast rate of flow and the fact that it is real-time data. The fire data utilised for this research according to the RabbitMQ web management console, flows at a rate of 1009 messages per second. The fire data can therefore be referred to as medium velocity data.

This research was conducted towards determining if Web Processing Services deployed in a cloud computing environment are suitable to process streaming medium velocity geospatial data for the purpose of rapidly generating 3D wildfire context visualisations in demand-time. The primary purpose of this research was to evaluate the efficiency of different algorithmic and architectural styles for process chaining in the cloud to generate 3D spatial context visualisations around detected active fires in demand-time.

1.3 Research Problem Statement

Vegetation wildfires occur in most parts of the world. Wildfires can get out of control with devastating consequences. Wildfires can be detected by sensors on earth observation satellites. Incidents relating to wildfires can be mitigated by distributing timeous wildfire detection notifications. Streaming active fire detection geospatial data generated by earth observation satellites moves at a medium velocity. This streaming geospatial data can be characterised as bursty. Streams are flooded with active fire detection data during certain periods and run dry during other periods in time. Bursty medium velocity geospatial data is complex to manage. A special computing paradigm might be required to handle this data.

An alerting component that adds a 3D spatial context visualisation of a detected wildfire to a notification message should perform tasks at a rapid rate to produce results in “demand-time” when the wildfire event notification is ready to be disseminated. The alerting component is formed out of a geoprocessing chain as several steps are required to generate the 3D spatial context visualisation. The execution time of the entire geoprocessing chain is not fast enough for 3D spatial context visualisations to be rapidly generated with tightly-coupled processing chain components and components that make use of Web Processing Services on a single thread on commodity hardware. This implies that messaging queues that transport the wildfire notifications will backlog and thus, 3D spatial context visualisations will not be ready in “demand-time”.

Cloud computing might provide a potential solution to this problem. A benefit of a cloud computing environment is the effortless use of horizontal scaling (elasticity). Cloud instances can be added for the processing tasks to be divided amongst more instances to generate results more rapidly. This is similar to a divide and conquer approach as more instances will be responsible to do the work and the performance of the geoprocessing chain will improve. Cloud computing further provides auto-scaling capabilities to deal with the bursty nature of the data. By utilising auto-scaling, cloud instances will automatically start when there is a demand for processing and will automatically shut down when there is no demand. The use of public cloud computing is constrained by costs, thus, it will be required that resources be used sparingly and strictly on demand. The resources must be powerful enough to handle the fast incoming data or messages. This is required to prevent message queues from backlogging and prevent a delay in the dissemination of fire notifications. A trade-off exists between the cost of the cloud instances' usage per hour and the size and number of instances.

It is not yet known whether cloud computing can successfully address this problem but, from thoroughly reviewing literature is hypothesised that by using Web Processing Services deployed within a cloud computing environment, a 3D wildfire context visualisation will be generated in just enough time to be attached to a fire notification that can be rapidly disseminated.

1.4 Research Question

The initial research question that served as a guide for this research was:

Are Web Processing Services deployed in a cloud computing environment where horizontal scaling can be achieved a suitable mechanism for rapidly processing medium velocity streaming geospatial data in order to generate and deliver wildfire event notifications that include 3D context visualisations of a wildfire scene in demand-time?

Initial experiments illustrated that certain approaches to rapidly process medium velocity streaming geospatial data were inefficient, therefore the research question was changed to:

What is the optimal design for a geocomputational cloud environment that can rapidly process medium velocity streaming geospatial data given cost and elasticity constraints, data delivery requirements and interoperability tradeoffs in order to generate 3D wildfire context visualisations?

The hypothesis of this research is that medium velocity streaming geospatial data cannot be processed rapidly enough to generate and deliver notifications with 3D visualisation payload in "demand-time" by deploying a Web Processing Service-based solution on a single process on commodity hardware.

1.5 Research Aims and Objectives

- Conduct a literature review on various topics that relates to the design of a 3D wildfire context visualisation generating geoprocessing system.
- Design, setup and execute the desktop-based (single thread on commodity hardware) experiments to: determine the optimal algorithmic style for a system required to process medium velocity streaming geospatial data at a rapid rate and deliver 3D wildfire context visualisations in demand-time; determine the architectural style for a system required to process medium velocity streaming geospatial data at a rapid rate and deliver 3D wildfire context visualisations in demand-time; determine if Web Processing Services is suitable for a system required to process medium velocity streaming geospatial data at a rapid rate and deliver 3D wildfire context visualisations in demand-time.
- Design, setup (with the algorithmic style results of objective two) and execute the cloud-based experiment to: Evaluate if Web Processing Services are suitable for a system required to process medium velocity streaming geospatial data at a rapid rate and deliver 3D wildfire context visualisations in demand-time; determine if the horizontal scaling of cloud instances can speed-up the performance of the geoprocessing chain.
- Evaluate and compare the results of the desktop-based (single-thread on commodity hardware) and cloud-based experiments.

1.6 Significance of Research

The final result that the geoprocessing chain delivers(a 3D wildfire context visualisation) will help fire managers obtain enhanced insight into wildfire events. Wildfire context visualisations in three dimensions enable viewers to view the topography of a fire scene that has a significant influence on fire spread. The context visualisations will give emergency responders a better understanding of the terrain where fire occurs. It will thus aid in planning how to access and control fires. Cloud computing might improve the throughput of the geoprocessing chain responsible for generating the 3D wildfire context visualisations by providing horizontal scaling capabilities. Due to the bursty nature of the data, the cloud can help to save costs by auto-scaling (scaling according to demand-automatically).

1.7 Limitations of Research

The project was merely applied to one domain (wildfire). It is not generic for the purposes of this research, although it can be translated to other domains.

The utilisation of the public cloud (AWS - Amazon Web Services) was heavily constrained by costs. Only Amazon EC2 (Elastic Compute Cloud) micro-instances were used because of the free 750 hours a month on the free tier (eligible solely within the first 12 months of AWS registration). Three Amazon Web Services accounts were used which meant that simply20 cloud instances per account might run at a time. The results may have been better if more cloud instances were used.

A micro-instance (t2.micro) has one virtual CPU (Central Processing Unit), one GB memory and is the lowest-cost general-purpose instance type. The physical processor of a micro-instance belongs to the Intel Xeon family and has a clock speed of 2.5 GHz but it can boost up to 3.3GHz. The network performance of an EC2 instance is very low. The poor performance of a micro-instance will have a negative impact of the geoprocessing chain. The CPU of an Amazon EC2 micro-instance is burstable but the baseline performance is consistent (only 10%). The CPU can burst above the baseline CPU performance, but it is governed by CPU credits. This limitation (CPU credit governing) has the potential to slow down the CPU performance of a cloud instance after a while. This meant that every cloud instance had to be stopped and restarted to receive CPU credits all over again. Every time an instance was stopped and restarted, Amazon will charge an hour's usage. With 20 micro-instances running for one hour, Amazon subtracted 20 hours from the free 750 hours of the free tier.

Certain graphics cards might be problematic. This pertains to the rendering of 3D wildfire context visualisations (when a URL attached to the fire event notification, is opened). The graphics card drivers sometimes cause WebGL to stop working, often when drivers update.

1.8 Overview of Remaining Chapters

Chapter Two: A thorough literature review was conducted. The review includes aspects addressed in this research such as wildfires, the Advanced Fire Information System, big geospatial data, distributed and high performance computing, enterprise messaging, Web Processing Services and geovisualisation. A section on related work is further included and evaluated to illustrate research gap(s) in the field that can be addressed.
Chapter Three: The research methodology followed is included in this chapter. A section on the design of the experiment is further included. The design of the abstract geoprocessing chain and the data required for the abstract geoprocessing chain are also discussed in this chapter.
Chapter Four: The implementation details of the geoprocessing chain are discussed in this chapter. Other details such as the required data, software and hardware used are further included.
Chapter Five: This chapter provides the results of the pre-tests, commodity hardware experiment and cloud computing experiment conducted. An in-depth discussion of the results of the pre-tests, experiment one and experiment two are also included in this chapter. Experiment one refers to tests conducted in a desktop environment. Experiment two contains tests conducted in a public cloud computing environment.
Chapter Six: The final chapter includes conclusions drawn through the analysis of the results of this research. This chapter further includes recommendations for future research.

1.9 Chapter Summary

This chapter provided a high-level overview of the problem addressed in the research by providing background information, the research problem statement, the research question, research objectives, research significance and limitations and an overview of the remaining chapters. 3D wildfire context visualisations are important as they can change the potential outcome of disasters. The design of a geoprocessing system required to generate 3D wildfire context visualisations in demand-time should be carefully considered. Rapid geoprocessing is required and therefore it is pivotal to design such a system in the most optimal manner possible. Due to the elastic nature of clouds, cloud computing might be a suitable solution for this problem. Various implementations will be tested to inform the final design of the system. Constraints have to be taken into consideration which will have an outcome on the processing times. The following chapter provides the reader with more detail regarding the surrounding aspects of the research.

2 Chapter Two: Literature Review

2.1 Chapter Overview

Chapter One provided a high-level overview of the problem addressed in the research by providing background information, the research problem statement, the research question, research objectives, research significance and limitations and finally, an overview of the remaining chapters. It was stated that a 3D context visualisation that has to be prepared as soon as a viewer views a wildfire notification could change the outcome of a disaster. Cloud computing might suit this kind of scenario. This chapter will provide detail on the research context and related work. This chapter will therefore contain information on aspects related to a 3D wildfire context visualisation generating system.

The objective is to frame the research problem better. The reader will gain deeper insight of aspects of the problem faced in this research and aspects of the potential solution. This chapter starts with the basics by providing a high-level overview of wildfires. The industry use case AFIS is introduced in this chapter and a discussion is included on wildfire detection. Big geospatial data is discussed as the data that AFIS consumes is streaming medium velocity geospatial data. Distributed and high performance computing is often applied when dealing with geospatial data, relevant material is therefore included in this chapter. A section on enterprise messaging is included on page 24. Web Processing Services were an integral part of this research and is discussed in this chapter as well (on page 25). The final product of the geoprocessing chain was a 3D context visualisation, thus a section on 3D geovisualisation is included (on page 27). Related work is discussed on page 28 to highlight the similarities and gaps in this research. This chapter gives the reader a more coherent understanding of the research problem at hand and continues to give the reader an insight into a potential solution of the research problem.

2.2 Research Context

2.2.1 Wildfires

Wildfires are fires that occur in areas with combustible vegetation outside the boundaries of urban areas. Fires require 16% oxygen, fuel and heat to ignite (British Columbia Wildfire Management Branch, 2014). There is 21% oxygen in the earth's atmosphere, thus one third of the criteria for fire to occur is fulfilled automatically. Fuels refer to any living or dead material that will burn. Heat can refer to the heat caused by the rays of the sun, lightning or even lighted cigarettes (National Disaster Management, n.d.). Unfortunately, 90% of fires are caused by humans, whether it be due to negligence or whether they are caused deliberately. This implies that only 10% of fires are caused by natural occurrences, (lightning) (National Disaster Management, n.d.).

In South Africa, fires are utilised as a landscape management technique and 70% of ecosystems that cover the South African landscape are fire adapted. This implies that 70% of ecosystems in South Africa require fires to maintain themselves. (Working on Fire, 2012). Fires usually start small but they have the tendency to get out of control with devastating consequences (Working on Fire, 2012). The rate of fire spread depends on:

- Weather conditions
- Wind
- Season
- Fuel conditions
- Topography (Working on Fire, 2012; Southern Cape Fire Protection Association, 2012)

Slope, aspect and terrain are three important elements of topography that play a significant role in the spread of fire. Slope refers to the steepness of the land. Wildfires have a tendency to spread faster uphill than downhill due to several factors:

- Flames are closer to fuel on the uphill side
- Fuels will become drier on the uphill side of a slope and will therefore ignite quicker than on the downhill side
- Uphill wind currents push heat flames into new fuels
- A draft is caused by rising convected heat from a slope

Aspect refers to the direction that the land faces. Some aspects receive more direct heat from the sun. This will have an influence on the dryness of the vegetation, temperature and humidity (British Columbia Wildfire Management Branch, 2014). Terrain may cause obstructions. When wind flows through a narrow path, it will increase in strength and influence rapid fire spread (British Columbia Wildfire Management Branch, 2014).

All the above-mentioned situations are responsible for rapid fire spread (British Columbia

Wildfire Management Branch, 2014). Wildfires usually spread in the direction of the wind and they spread faster when there are dead plant materials. Wildfires spread faster in fine fuels (Southern Cape Fire Protection Association, n.d.; National Disaster Management, n.d.). Wildfires will continue to burn as long as there is enough vegetation (fuel) to burn and the weather conditions are favourable (Department of Water Affairs and Forestry, n.d.). Favourable conditions for vegetation wildfires are:

- Dry season
- Warm temperatures
- Moderate to high wind speeds
- Continuously spread out vegetation (fuels)

Intervention is pivotal for managing incidents that might arise from uncontrolled fires. Incidents relating to wildfires might be mitigated by the timeous dissemination of notifications that contain information derived from earth observing satellites. The elements of topography can be better viewed in three dimensions. Citizens and firefighters will gain a better understanding of a wildfire by viewing the scene in three dimensions.

It can thus be noted that there are many variables that have an influence on wildfires such as temperature, wind, fuel, topography and heat. All these variables can be visualised in two dimensions to give an user an indication of what is going on in the area of the fire. To enhance such a visualisation would be to create a three dimensional contextual visualisation around a fire to enhance or increase a user's situational awareness.

2.2.2 Industry Use-Case: Advanced Fire Information System (AFIS)

The research undertaken was inspired by an industry use case. Researchers from the CSIR's (Council for Scientific and Industrial Research) Meraka institute developed the Advanced Fire Information System, known as AFIS. AFIS is a tool that provides near real-time fire information derived from earth observation satellites to users in Southern Africa, South America, Eastern Europe and East Africa. It allows near real-time identification of fires and it has an automated notification capability. AFIS includes:

- Near real-time MODIS (Moderate Resolution Imaging Spectroradiometer) data
- SEVIRI (Spinning Enhanced Visible and Infrared Imager) data
- FIRMS (Fire Information for Resource Management System) data

- Weather station data
- Weather forecast data (CSIR, 2012)

The system utilises data from geostationary satellites such as:

- ESA's (European Space Agency)MSG (Meteosat Second Generation) satellite
- Polar orbiting satellites such as NASA's (National Aeronautics and Space Administration) Aqua and Terra satellites for fire detection(CSIR, 2012)

The European Space Agency's (ESA) Meteosat Second Generation (MSG) satellite is equipped with the Spinning Enhanced Visible and Infrared Imager (SEVIRI) instrument. The MSG is a geostationary satellite. SEVIRI data has a spatial resolution of approximately four kilometres and a temporal resolution of 15 minutes. The National Aeronautics and Space Administration's (NASA) Terra and Aqua satellites are equipped with the Moderate Resolution Imaging Spectroradiometer (MODIS) instrument. Terra and Aqua are polar-orbiting satellites and thus they have a higher spatial resolution than geostationary satellites. MODIS data has a spatial resolution of approximately one kilometre and a temporal resolution of six hours. The significant difference in spatial resolution is because the Meteosat Second Generation satellite is geostationary and Terra and Aqua are polar orbiting satellites. This implies moving much closer to the earth than the Meteosat Second Generation satellite. The higher spatial resolution data will represent more fires than the lower spatial resolution data because the higher spatial resolution sensor can identify smaller fires (McFerren *et al*., 2009).

Researchers from the CSIR developed algorithms to eliminate false alarms. Weather data is utilised along with scientific models to predict the fire danger index (by measuring the dryness of the vegetation) to predict whether an area is in danger of wildfires (Wild, 2013).

AFIS has a web-based mapping interface and a mobile application available for the Apple iOS and Android mobile operating systems (CSIR, 2012). The web-based mapping interface allows a user to view various map layers and query the layers. Fires can be viewed in near real-time. The AFIS web-based mapping service likewise known as the AFIS viewer is a free service but a user is required to register to use the service (CSIR, 2012). The mobile application allows a user to define locations. The user will then receive notifications if fires occur nearby defined locations. The mobile application provides the daily fire danger index (degree of danger) and fire danger forecasts. Near real-time active fires are displayed on a map and detailed information about the fires are included. The vegetation dryness and time series data ("a series of values of a quantity obtained at successive times (Oxford University Press, 2015)") is offered too. Geotagged photographs (photographs with a geographical location assigned to it) of fire observations can be uploaded (CSIR, 2012).

Prediction capabilities are provided by the fire danger indices and forecasts. This may help with planning, as a fire manager will be prepared for fires and will get the necessary tools and put the parties that will help to extinguish fires on stand-by. The AFIS web-based mapping interface enables the monitoring of fires. The mobile application additionally offers this capability. Alerting is achieved by disseminating wildfire notifications. A wildfire notification can be an SMS, an email or a push notification if a user is registered (CSIR, 2012).

AFIS provides users with prediction, detection, monitoring, alerting, and planning capabilities. The system grants users the ability to mitigate catastrophic situations caused by wildfires.

2.2.3 Wildfire Detection

In this research, wildfires are characterised as intermittent datastreams of events per detecting sensor. Two datastreams were utilised for the research. The first datastream consists of active fire events detected by the SEVIRI (Spinning Enhanced Visible & Infrared Imager) sensor aboard the Meteosat 8 geostationary satellite. SEVIRI data has a temporal resolution of 15 minutes. The maximum number of events detected by SEVIRI in the CSIR's database are 8362.

The second datastream consists of active fire events detected by the MODIS (Moderate Resolution Imaging Spectroradiometer), the instrument aboard Aqua and Terra polar orbiting satellites. MODIS data has a temporal resolution of six hours. The maximum number of detected MODIS events in the CSIR's database is 48606 (McFerren *et al*., 2009).

The overwhelming difference in the number of events detected by the two sensors relates to spatial resolution: MODIS detects fires at approximately a one kilometre resolution while SEVIRI detects at approximately a four kilometre resolution. MODIS can identify much smaller fires and will therefore detect a larger quantity of fires. It can be deducted that a large quantity of fires (atomic data) will be required to be processed rapidly with large datasets of contextual variables.

2.2.4 Big Geospatial Data

Geospatial data is divided into:

- Locational data (position and dimension similar to geometric characteristics -what)
- Attribute data (non-geometric characteristics -where)
- Temporal data (valid time -when) (Kraak *et al*., 2010)

Geospatial data is a discrete presentation of continuous phenomena (Karimi, 2014). Percivall (2013) stated that geospatial data has always been big data. The reasons that geospatial data can be viewed as big data are because geospatial data is large in size, comes in various forms and has an update rate that cannot be handled by standard spatial computing technologies (Shekhar *et al*., 2012).

There are various types of geospatial data such as raster and vector. Big geospatial data is an instance of these data types that exhibit at least one of the three Vs of big data, namely:

- Volume
- Velocity
- Variety (Karimi, 2014)

Consumers interact with several services that generate data daily. Several sensor arrays, that continuously generate data, exist across the globe. Streaming geospatial data have several characteristics:

- The data is voluminous
- It streams at high velocities
- It encompasses many things valuable (Goldberg *et al*., 2014)

According to Nick Skytland (2012) from NASA, big data is a “collection of datasets so large and complex that your legacy IT systems cannot handle them”. Big data requires more capabilities than average systems provide. Dumbill (2012) described big data as:

- Data that is large in size, that makes it too large to transfer
- Data that moves fast as it might stream in a matter of nanoseconds or milliseconds
- Data is diverse, as various data types or data structures might not fit into an existing database's structure or architecture (Dumbill, 2012)

Three Vs are used to describe the three characteristics of big data. The three characteristics (Vs) are volume, velocity and variety. Data that cannot be processed effectively and efficiently will remain meaningless. A range of problems introduced by the characteristics of big data can cause the processing of big data to be ineffective and inefficient. The characteristics will be discussed below.

The first V is used to describe volume. Volume is the primary characteristic of big data. Volume could be quantified in size of memory (Russom, 2011). The second V is used to describe velocity. Velocity refers to the rate at which data flows (frequency and speed). Velocity is not simply concerned with the rate of data input, but with the rate of data output or the rate of data storage as well (Russom, 2011). Russom (2011) stated that velocity could be thought of as “the frequency of data generation or the frequency of data delivery". Dumbill (2013) suggested that fast-moving data be streamed into bulk storage to do batch processing. The third V is used to describe variety. Source data can be obtained from many sources. The source data might have various data types and data structures. The represented entities might vary (Russom, 2011).

The paragraphs above confirm that geospatial data can be regarded as big data. The research conducted did not deal with the aspects of big data as a whole. The research addressed two out of three aspects of big data that placed obstacles in the design of the geoprocessing solution, they are:

- Medium velocity
- High volume

The first aspect was medium velocity streaming geospatial data. Wildfires are characterised as intermittent datastreams of events. For data to be characterised as high velocity data it must be streaming and it must be generated at a high rate (millions of messages per second). Twitter data (tweets)can be viewed as high velocity data because of the fast rate of flow and the fact that it is real-time data. Twitter receives 5700 tweets (messages) per second (Raffi, 2013). According to the RabbitMQ web management console, fire data flows at a rate of 1009 messages per second and the fire data is real-time data. Fire data can therefore be referred to as medium velocity data.

The second aspect was the volume geospatial data required for determining values at a fire scene. GeoTIFFS are high in volume. High volume data can introduce data transfer challenges (time consuming), depending on location and bandwidth.

In current times organisations that do not have suitable computing resources to process big data, have the opportunity to do so at affordable costs. This is due to a decrease in hardware costs, an increase in the number of free and open source technologies and an increase in cloud computing technologies (Dumbill, 2012).

The processing of streaming data can be problematic due to two reasons. The first reason, is when the data input rate is too fast and the second reason is when the application instantly requires a response (data output) (Dumbill, 2012). Dumbill (2012) noted that big data could be processed by using various parallel processing architectures (distributed computing) that distribute datasets across multiple servers, does some processing on the datasets and combines the processing results from the multiple servers.

Skytland (2012) noted that data is continuously streaming, faster than it can be stored, managed or interpreted. This is a perfect description of a problem that this research aimed to address. From this section, it could be observed that the characteristics of big data place specific requirements on how it should be processed. High performance computing or distributed computing is one requirement.

2.2.5 Distributed/High Performance Computing

2.2.5.1 Demand for Grid Computing and Cloud Computing

Earth observation systems gather large quantities of data about the planet. This quantity of data frequently increases at a rapid rate. Data sharing is becoming increasingly important amongst people that have an interest in earth observation. Due to the high volume of data, storage problems, computationally intensive processing, real-time access and resource sharing, earth observation systems can benefit by shifting to a distributed environment such as cloud computing or grid computing for big data processing (Petcu *et al*., 2010).

McCullough (2011) stated that: “Processing spatial data is notoriously time-consuming, and it is not uncommon for geoprocessing to present a bottleneck in spatial workflows”. Parallel processing can be utilised to reduce the processing time of spatial data. He stated that there is an urgent need for near real-time geoprocessing.

There are two categories of real-time geoprocessing, according to McCullough *et al* (2011).

The categories are snapshot geoprocessing and stream geoprocessing. Snapshot geoprocessing, refers to a fixed snapshot of spatial data processed. Snapshot geoprocessing forms a part of a real-time monitoring system, the results should be produced within a fixed time-frame. The input data used for the processing might come from an unreliable sensor (McCullough *et al*., 2011). Datastream processing refers to a series of observations (that represent a specific time interval) that will be processed. A datastream is a "potentially unbounded sequence of tuple time-stamp pairs" (McCullough *et al*., 2011).

Various strategies can be followed with parallel geoprocessing. The first strategy is synchronous geoprocessing. The same algorithm will run in parallel on several machines that share the same information in regular time intervals. If different algorithms are executed on different machines and they do not share information in regular time intervals, this is referred to as asynchronous geoprocessing (McCullough, 2011).

Cloud computing and grid computing are service oriented architectures (McCullough, 2011). Parallel processing can be performed in the cloud or on a grid. Cloud and grid computing (otherwise referred to as parallel computing) are effective because "monitoring events and entities, and making predictions about their future state carries a large computational burden (McCullough, 2011)". There are two categories of real-time systems: hard systems and soft systems. Hard systems are real-time critical systems. To avoid disaster, hard systems are required to meet a specific deadline. Soft systems are not real-time critical and thus, results are still useful after a long period of time (McCullough, 2011).

Real-time systems have the following problems:

- Have to process jobs of unknown size and time continuously
- Must keep track of the data as it arrives
- Must operate on datastreams by performing complex operations and pattern detection
- Should detect corrupt or missing data on the fly (due to the nature of satellite data) (McCullough, 2011)

Satellite data might be unreliable in terms of shot noise, line start or stop problems, line or column drops, or line or column stripping. Shot noise refers to multiple bad pixels in imagery. A bad pixel represents an individual pixel which a sensor does not record spectral data for (Jensen, 2004). Line start or stop problems occurs when spectral data was not recorded at the beginning or the end of a scan line. Pixel data might be placed at the incorrect location of the scan line in such a situation (Jensen, 2004). Line or column stripping occurs when a detector of a sensor goes out of radiometric adjustment. Certain pixels might have a systematically higher value than other for the same band (Jensen, 2004).

There are three reasons why distributed computing should be utilised when working with earth observation data:

- The first reason to use distributed computing is because sensor or satellite data is large and data storage might pose as a challenge
- The second reason is data analysis is time consuming because of the size of the data
- The third reason is that the software required to handle and process the data is expensive

Because earth observation data is large, distributed file systems will be required when working with earth observation data (Petcu *et al*., 2010). Cloud and grid computing can provide distributed file systems.

Cloud and grid computing are almost similar due to the fact that they have rapid processing capabilities. The first significant difference between the two computing paradigms is that grid computing is application specific and the main user of a grid is a scientist and cloud computing is not application specific and businesses usually utilises clouds (McCullough, 2011). The second difference, but perhaps the most important difference is that cloud computing systems can be used on-demand but grid computing systems use scheduling algorithms. A user will therefore be required to wait until other jobs are completed before his or her job can start to be processed with grid computing. Grid computing is not as flexible as cloud computing (McCullough, 2011).

2.2.5.2 Grid Computing

A grid is a collection of machines, referred to as nodes or resources (Kumar *et al*., 2012). Grid computing applications utilise high-end computers and huge storage systems because of the high dimensionality of the datasets processed on the grid. Geographically distributed organisations and scientists who wish to share resources may benefit from grid computing (Kumar *et al*., 2012).

Grid computing has several benefits. Grid computing can improve the performance of applications that involve significant computational modelling. Sensor based geoprocessing systems can either increase or decrease the scale of analysis in terms of datastreams, geographic extent and precision of the analysis. Grid computing saves money. Less money will be required to be invested in hardware and software processing resources, because access to resources can occur remotely. A central data repository can facilitate the integration of various data sources, thus it can be accessed as a service. Spatial data is large in volume and not portable. This is one reason why remote data analysis is suitable (McCullough, 2011).

Grid computing has several challenges as well. The first challenge is an architectural challenge. It is a challenge to orchestrate services across geospatial and grid computing domains. The second challenge is a computational challenge that deals with improving geoprocessing performance. The third challenge refers to semantic descriptions of geospatial services to facilitate discovery and reconfigurable chaining. Deploying web services on a grid presents a challenge. Web services are stateless and grid applications require the ability to store states (McCullough, 2011). It must be emphasised that real-time processing cannot occur on a grid, therefore cloud computing can be viewed as an alternative.

2.2.5.3 Cloud Computing

Goldberg *et al*. (2014) defined cloud computing as the hosting of services over the internet that can be located all over the world. Users of these remote services do not have to concern themselves over the underlying technologies. Baranski *et al*. (2011) stated that cloud computing provides the capability to distribute applications, processing and data on resources (nodes). There several other definitions of cloud computing. Vaquero *et al*. (2008) provided a definition for the cloud that encompassed parts of various definitions. They defined clouds as: “a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically re-configured to adjust to a variable load (scale) and allowing for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the infrastructure provider by means of customized Service Level Architectures” .

There are four cloud computing service models. They are Software-as-a-Service, Platform-as-a-Service and Infrastructure-as-a-Service (Armbrust *et al*., 2010) and Data storage-as-a-Service (Dillon *et al*., 2010). The first service model is Software-as-a-Service (SaaS). Cloud users release applications in a hosting environment accessed through networks by clients (Dillon *et al*., 2010). Jadeja *et al*. (2010) used when users run software from their computers, on demand and remotely as an example of Software-as-a-Service. The second model is Platform-as-a-Service (PaaS). Cloud users develop services and applications on the cloud (Dillon *et al*., 2010). Jadeja *et al*. (2010) used the following as an example of Platform-as-a-Service, when users deploy their own applications on a remote platform comprised of hardware, software and data. The third model is Infrastructure-as-a-Service (IaaS). Cloud users can directly use the information technology infrastructure. Jadeja *et al*. (2010) used when users deploy virtual machine instances as an example of Infrastructure-as-a-service. According to Dillon *et al*. (2010), Data storage-as-a-Service (DaaS) can be seen as the delivery of virtual storage on demand.

There are three cloud deployment models (Armbrust *et al*., 2010). They are public, private and hybrid. The public cloud is open for use by public consumers. The public cloud service provider has full ownership of the cloud infrastructure. A public cloud is generally preferred because no upfront investment is required for expensive infrastructure and public cloud service providers enforce a pay-as-you-use policy. A private cloud is a cloud infrastructure operated by a single organisation. A private cloud can be managed by the organisation or a third party. Generally a private cloud is usually preferred due to the optimisation of utilisation, security concerns, data transfer costs and control. The hybrid cloud is a combination of a public cloud and a private cloud. A hybrid cloud is preferred when a private cloud is used but more computing power is required at certain times. The private cloud will therefore be used for general processing, but during peak loads instances can scale to a public cloud (Dillon *et al*., 2010; Mell *et al*., 2011).

According to the documentation of Amazon Web Services (n.d.), a public cloud computing provider, there is a variety of benefits of cloud computing. The benefits can be divided into two comprehensive categories namely business benefits and technical benefits.

The most important business benefit is that no upfront investments are required for a public cloud infrastructure. A user is not required to pay any fixed costs for hardware, management, operations, personnel, real estate and physical security (Amazon Web Services, n.d.). By utilising public cloud computing, a user is simply required to pay for resources (instances, storage, etc.) that he/she has utilised. "Cloud computing providers have developed specialised software that allows them to link thousands of commodity computers to act as a cloud of cloud computing resources (Goldberg *et al*., 2014)". Computing resources can be shared in the cloud (Baranski *et al*., 2011). Cloud computing provides a just-in-time infrastructure. The just-in-time infrastructure refers to just-in-time provisioning. This implies that resources will strictly be used only when they are required if a user desires this feature. This concept can furthermore be referred to as on-demand computing. It provides more efficient resource utilisation by instances being started and stopped according to demand. When dealing with bursty data (data that arrives in irregular intervals), on-demand computing might aid in cost saving as merely the minimum quantity of required resources will be used. The overall concept described above is the elasticity of the cloud. Public cloud computing provides usage based costing by implementing a pay-as-you-go infrastructure. This may mean pay-by-the-hour or pay-by-the-number-of-requests. Unfortunately, every time an instance is stopped and restarted, the cloud computing service provider will charge for a new hour of usage. Cloud computing exploits the concept of parallelisation. Work can be divided amongst nodes to speed up the delivery time of results by conducting parallel processing. This implies that more than one node might be responsible for the same task, but several events (messages)can be processed simultaneously (at the same time) (Amazon Web Services, n.d.).

The first technical benefit is automation. Cloud computing provides APIs to enable a scriptable infrastructure so that portable systems can be designed to develop repeatable systems. Horizontal scaling can be performed automatically with no need for human intervention. When scaling occurs, the workload can be divided amongst cloud computing nodes to decrease the time to produce results. Traffic or usage can be analysed to determine usage patterns for proactive scaling and instances can be cloned seamlessly. Cloud computing encourages distribution and redundancy.   Data and applications in the cloud can be replicated in any other location in the world. Cloud computing is therefore reliable in terms of failover. Cloud computing enables load balancing. The overflow of the traffic to the cloud maybe handled automatically by horizontal scaling (starting new machines to conduct geoprocessing during peak times and reducing the number of active machines during less busy times) (Amazon Web Services, n.d.).

With public cloud computing data transfer costs and data security are significant concerns. Data transfer between different regions is costly. Security remains a concern, although cloud computing providers attempted to address it by implementing strong authentication mechanisms and rule-based firewalls.

Yang *et al*. (2013) demonstrated by doing a thorough literature review, that cloud computing is an imperative factor required to enable Digital Earth. From the literature review conducted in this research, it was explicitly illustrated that the benefits of cloud computing outweighs the benefits of grid computing. The challenges of cloud computing are far less than the challenges of grid computing. The horizontal scaling feature of cloud computing will greatly improve parallelisation.

2.2.6 Enterprise Messaging

The Advanced Message Queuing Protocol (AMQP) enables applications to communicate over messaging middleware servers, referred to as brokers (Aiyagari *et al*., 2008). The brokers must comply with the AMQP standard (Eugster *et al*., 2003). AMQP relates to enterprise messaging systems because it is a protocol that allows for the sending of messages between computer systems. Loosely-coupled architectures are encouraged by the AMQP.

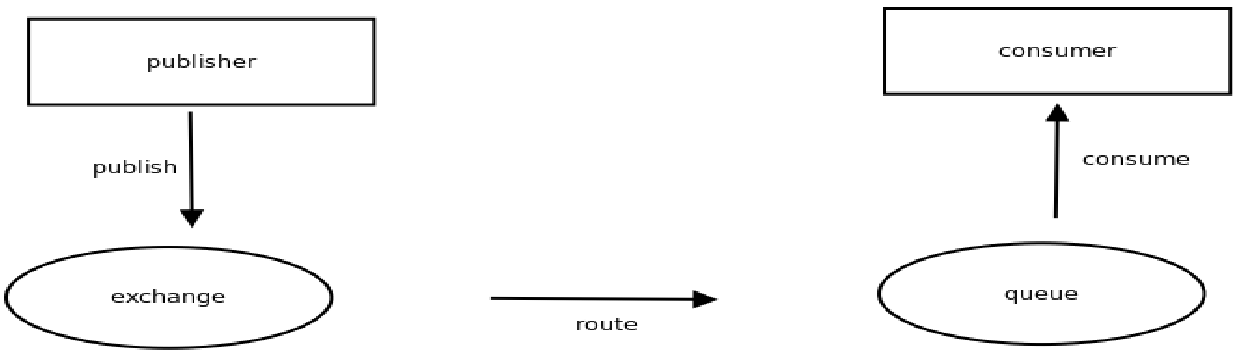


Figure 2: Advanced Message Queuing Protocol diagram

An AMQP model, illustrated in *Figure 2* consists of components that publish messages (known as producers), routers (known as exchanges) that move messages to queues according to a variety of messaging patterns (publish or subscribe), queues and consumers that act on messages.

An exchange receives messages from a publisher and will route the messages to message queues based on certain criteria. A message queue is a component that stores messages until consumers can process them. The routing criteria is defined as the relationship between a messaging queue and an exchange. The components of such a system are loosely-coupled, thus, they are not aware of each other (Eugster *et al*., 2003). This architecture allows flexible and scalable systems to be built. A scalable system such as a demand-time fire message geoprocessing and 3D visualisation generating system.

An AMQP approach can enable concurrency capabilities and allows for distributed processing. The protocol is binary and asynchronous. It can be divided into two layers namely, a functional layer and a transport layer. The functional layer has a set of methods that performs work on behalf of the application. The transport layer transports methods from the application to the server and back again. It conducts error handling, content encoding, data representation, heart-beating and multiplexing (Eugster *et al*., 2003).

Messaging brokers receive messages from producer applications that publish them and routes them to consumers (applications that process messages). The routing algorithm depends on the exchange type and rules referred to as bindings. Producers and consumers may reside on different machines. Producers publish messages. A producer is a user application that sends messages to exchange. Exchanges distribute messages to queues (buffers used to store messages). Brokers push messages to consumers (used to process messages) prescribed to queues or consumers. They can pull messages on demand. Whilst publishing a message, producers might specify various attributes.

Networks are unreliable and this is why acknowledgements are important. Once a message is delivered to a consumer, a consumer will notify the broker. A broker will then remove the message from the queue when the notification was received (Eugster *et al*., 2003).

Point-to-point and synchronous communication lead to rigid applications, the publish/subscribe style aims to improve this.

AMQP components are loosely-coupled. This means that the components are unaware of each other. A system can scale and grow effortlessly and information can be disseminated in demand-time. Multiple fire events can be processed in parallel if system components are loosely-coupled.

2.2.7 Web Processing Service

The OGC Web Processing Service (WPS) provides a mechanism to perform a variety of distributed web-based processing operations on geospatial data using the Remote Procedure Call (RPS) architectural style.

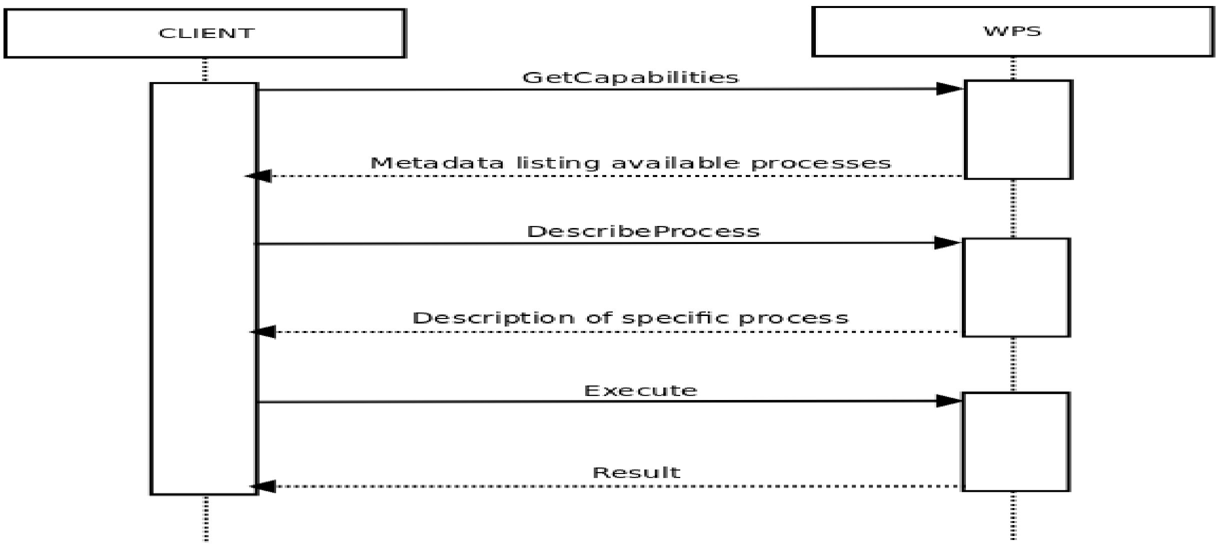

CLIENT
WPS
GetCapabilities
Metadata listing available processes
DescribeProcess
Description of specific process
Execute
Result

Figure 3: Web Processing Service sequence diagram

The OGC Web Processing Service is defined as a standardised interface that provides rules on how requests and responses of arbitrary geoprocessing services should be constructed. A Web Processing Service acts as a middleware service for data (Meng *et al*., 2009). Data can be obtained from external sources and middleware services.

The standardised interface contains three operations. *Figure 3* illustrates the sequence in which the operations occur. The first operation is GetCapabilities. The GetCapabilities operation provides metadata and information about the processes offered, and metadata about the service provider. The processes offered can be either spatial or non-spatial. The second operation is DescribeProcess. The DescribeProcess operation accepts a process listed by the GetCapabilities function as a parameter. The second operation includes metadata, input-parameters and output-parameters of the specific process displayed by the GetCapabilities operation. By executing the third operation, Execute, one can run a specific process inserted as the input parameter by the GetCapabilities operation.

The ExecuteResponse document responds to the Execute operation, which indicates a process status, used inputs and value outputs (if simple or if complex), and links. There are various process status messages such as ProcessAccepted, ProcessStarted, ProcessSucceeded or ProcessFailed. ProcessAccepted means the process is in the queue, waiting to start. ProcessStarted means the process has begun. ProcessSucceeded means the process has been executed successfully and ProcessFailed means that a problem has occurred (Michael *et al*., 2007). The status will show where the ExecuteResponse document is located and may show progress or error messages (Michael *et al*., 2007).

The Web Processing Service interface facilitates the discovery and publishing of geospatial processes (Open Geospatial Consortium, 2007). The main advantages of using the OGC Web Processing Service are interoperability (Geoprocessing.info, n.d.) and software implementation abstraction. Geoprocessing can take place regardless of the software or hardware on a user's computer. Multiple web service access approaches are supported by an OGC Web Processing Service. These include standardised approaches such as HTTP (HyperText Transfer Protocol) POST using Extensible Markup Language (XML), HTTP GET using KVP (Key Value Pair arguments) and SOAP (Simple Object Access Protocol) request (Kiehle *et al*., 2007).

Geoprocessing is highly distributed because it can occur anywhere on the internet and on demand (Geoprocessing.info, n.d.). Software implementations of a geoprocessing component on the server-side of an OGC Web Processing Service can change, but it will have no impact on the Web Processing Service client (as the calling interface remains unchanged).

Web Processing Services furthermore, enable clients to gain access to the most current data and processing implementations, without a need to change client implementations. Processes are re-usable in multiple applications (Geoprocessing.info, n.d.). Web Processing Services can be exploited for cloud computing, grid computing or other forms of high performance geoprocessing.

Web Processing Services are usually orchestrated (chaining) or combined to form a higher level process or workflows. Participating services are chosen and organised by a central orchestration service. Services are loosely-coupled with service orchestration. Web services can be orchestrated manually or by passing the output of one web service, as input to another web service. This can occur by using a configuration file to define the order of the web service interaction or it can occur in an automatic manner. The processes offered by a Web Processing Service execute locally inside an application-server (Baranski, 2008).

The advantages of Web Processing Services such as interoperability, software implementation abstraction and the fact that Web Processing Services can be chained to form higher level processes, makes it attractive to use for the geoprocessing of fire events.

2.2.8 Geovisualisation

The internet and a remarkable increase in geospatial data has created new opportunities to visualise and interact with spatial information (Hildebrand *et al*. 2010). Improving availability of more powerful computing resources enabled the access of distributed resources through the internet. Web based visual displays of spatial information can aid with everyday problem solving. Tiede *et al*., (2010) argues that 3D views provide additional information and enhances information delivery. Plaisant (2004) stated "Information visualization is sometimes described as a way to answer questions you didn't know you had".

There are several definitions of geovisualisation. Several authors have attempted to define the term geovisualisation.

Kwan *et al*. (2003) defined geovisualisation additionally known as the visualisation of geographic information as the use of visual representations and visualisation abilities to make the spatial context and problems visible. Spatial patterns, trends and relationships can be identified and interpreted with geovisualisation (Kraak, n.d.; MacEachren *et al*., 2001). Kraak (n.d.) and MacEachren *et al*. (2001) noted that geovisualisation integrates approaches from "scientific visualisation, cartography, image analysis, visualisation, explanatory data analysis and GIS. Geovisualisation provides theory, methods and tools for visual exploration, analysis, synthesis and presentation of geospatial data". Large quantities of data in various formats can be integrated to generate complex but realistic representations. These representations can be understood by utilising human visualisation capabilities as data can be viewed from various angles. Slocum *et al*. (2001) noted that the objective of geovisualisation is to develop techniques that will assist in understanding the earth's environment.

Castrillón *et al*. (2011) indicated that3Dvisualisations provide truthful and realistic depictions of the real world events that can be useful to fire managers. 3D fire visualisations can give a user or fire manager a better understanding of a fire situation and an indication fire spread. 3D visualisations are useful to emergency responders for planning in dangerous situations.

Geovisualisation may therefore play an important role monitoring wildfires and mitigating incidents related to wildfires.

2.3 Related Work

2.3.1 Distributed/High Performance Computing

Extensive research was conducted that relates to distributed and high performance computing. Researchers have experimented with public, private and hybrid clouds. Grid computing was studied extensively.

Baranski *et al*. (2010) discussed an implementation of a hybrid cloud infrastructure. Similar to previous experiments that they have conducted, it was based on 52°North's Web Processing Service that utilises OpenNebula. OpenNebula is software used for building and managing an infrastructure-as-a-service cloud. OpenNebula utilises infrastructures that already exist to ensure reliability. OpenNebula is open source and completely platform independent (OpenNebula, 2013).Huang *et al*. (2013) compared various open source cloud computing solutions against each other. The cloud computing solutions were OpenNebula, Eucalyptus and CloudStack. The authors found that OpenNebula was a good option for cloud computing as it has the fastest internal network of the three solutions. OpenNebula has the second fastest operations regarding handling virtual machines, images, snapshots, volumes and networking.

Java was used with 52°North's Web Processing Service. A hybrid cloud should be considered when an organisation might not have sufficient computational resources (in a private cloud). An organisation should then outsource some of their processes to public clouds. This concept can be referred to as a hybrid cloud (a combination of a public and a private cloud). The authors deployed OpenNebula within a cluster (it had one head node and various worker nodes). Virtual machines were interconnected in the virtual network and managed by the front end. A XEN hypervisor was used to configure the worker node. It enabled the Virtual Infrastructure Manager to start and stop a virtual machine at the worker nodes. A virtual machine image was supplied to the Virtual Infrastructure Manager to deploy on the worker nodes. An Amazon Machine Image was required to access the public cloud. Scheduling rules were then defined. The virtual machine image contained the Web Processing Service installation (Baranski *et al*., 2010). The article provided the reader with details regarding the setup of a cloud based geoprocessing system. The article highlighted the fact that private cloud computing should be combined with public cloud computing to save costs. Utilising a public cloud computing service on its own would be expensive.

Brown *et al*. experimented with cloud computing and geoprocessing. The authors acquired data and developed the geoprocessing functions (for the transformation and manipulation of imagery). Thereafter, they developed a Nebula cloud computing framework (Brown *et al*., 2012) and migrated their code. Imagery was obtained from NASA. Using the imagery, cloud enabled processing was tested; data products were generated and validated. Software packages such as ArcGIS and ERDAS imagine were used to test Nebula (Brown *et al*., 2012). The article provided details on how cloud computing works in terms of cloud instances, cloud storage and cloud security. The authors demonstrated the capabilities of the Nebula platform.

Gong *et al.* (2010) experimented with a cloud infrastructure and a Web Processing Service using Microsoft's Azure Platform. The Azure Platform merely supports Microsoft's Windows operating system. The authors had to adapt their existing spatial analysis platform for the cloud. They migrated their computational application to the compute service of the Azure platform. The authors used AppFabric to connect the cloud-based services to the geoprocessing applications. Storage services were used to store and manage the application data. Geoprocessing systems should be scalable due to the nature of geospatial data. The authors demonstrated that cloud computing may provide advantages if used for geoprocessing systems.

Juve *et al*. (2012) executed experiments using three workflow applications on Amazon's EC2 cloud computing platforms. From the results of the experiments, the authors concluded that cloud computing is convenient for deploying workflows. The authors noted that cloud computing costs drastically increase when multiple instances are used. The authors observed that the choice of storage system had an impact on workflow runtime. It can be noted from the graphs they provided, that using a local file system is more efficient than using the S3 file system (workflow runtime). The authors further noted that the processing times of the workflow will decrease by increasing the number of nodes (virtual machines) and number of cores (per machine).

Akioka and Muraoka (2010) investigated utilising Amazon's EC2 as a research tool. They investigated the computational performance of EC2 instances and estimated the operational costs of EC2 instances. This research raised a very important question, what is the virtual machine management policy of Amazon Web Services? Virtual instances might be allocated to a physical machine in excess. If true, this will slow down performance.

Several observations can be made from the work discussed in this subsection. Due to the nature of geospatial data (high volume, high velocity) and the fact that there might exist a high demand for geoprocessing, cloud computing can be considered as a good option as it provides scalability. Public cloud computing is costly, although, processing times will improve by increasing the number of processing nodes and "hiring" more powerful machines. A hybrid cloud is therefore suggested as during less busy times, a private cloud can be utilised that costs almost nothing. During peak times, processing can be outsourced to a public cloud. This combination will optimise costs. Using a public cloud is risky, as an insufficient amount of information is provided on the allocation of virtual instances and the condition of the public cloud provider's physical machines.

2.3.2 Enterprise Messaging

Numerous experiments were conducted to benchmark the performance of the Advanced Message Queuing Protocol. On internet forums, the community complains about the slow performance of technologies that use the Advanced Message Queuing Protocol.

After conducting experiments, Fernandes *et al*. (2013) observed that the Advanced Message Queuing Protocol provided acceptable results in performance when there were large quantities of messages to exchange. The Advanced Message Queuing Protocol did not provide acceptable results when few messages were exchanged. Johnsen *et al*. (2013) noted that the Advanced Message Queuing Protocol generated a lot of communication between the Advanced Message Queuing Protocol client and broker. AMQP should be used with high bandwidth networks. It can furthermore cause an overhead.

Two important observations were made that were of interest. The Advanced Message Queuing Protocol's performance is acceptable when large quantities of messages are involved. Fire datastreams contains large quantities of messages. The Advanced Message Queuing Protocol appears to be a good option for enterprise messaging. A high bandwidth network is a requirement for using the Advanced Message Queuing Protocol.

2.3.3 Web Processing Service

Researchers experimented with Web Processing Services for several years. Geoprocessing chains were created in most of the experiments that were conducted.

Baranski (2008) attempted to implement an approach to put a Web Processing Service on a grid. This was based on 52° North's Web Processing service. The processes and algorithms were managed inside an algorithm repository. Every process utilised third-party libraries (Baranski, 2008). When a distributed process was executed, the input data and application binaries were copied into the computation nodes and the application binaries were executed concurrently on each computation node. As soon as all of the calculation processes finished, the Web Processing Services fetched the resulting datasets and combined them (Baranski, 2008). The authors concluded with the remark that the performance of the calculations and availability of service maybe improved (Baranski, 2008).

Murillo evaluated the parallelisation of geoprocesses on Amazon Web Services through Web Processing Services using 52˚ North. Java was utilised. The number of micro instances were incremented by using GridGain. Murillo found that processing times decreased when the number of instances increased. Murillo concluded that using Web Processing Services in the cloud is acceptable for the purpose of calculating statistics.

Samadzadegan *et al*. (2013) proposed an architecture design based on OGC web services for automated workflow for the acquisition and processing of remotely sensed data for detecting fires and sending notifications out to authorities. The authors argued that Web Processing Services should be chained to detect fires from MODIS data. They evaluated the architecture by using a use case, the GeoPortal client. The authors determined that this architecture can be used for several disaster management and environmental monitoring geospatial applications.

Dasgupta and Ghosh (2011) orchestrated Web Processing Services to provide access to geospatial information on mobile devices. In 2010, Sun *et al*. (2010) created a workflow by integrating geoprocessing services with Web 2.0. Shao *et al*. (2012) implemented a geoprocessing service that integrated methods with Amazon Web Services to conduct geoprocessing in a distributed environment.

Westerholt & Resch (2014) noted that geoprocessing tasks are complex and time consuming due to messaging overheads. The authors proposed an event-driven architecture for web-based asynchronous geoprocessing with Web Processing Services. Push-based notifications are suggested to produce results in real-time. Evangelidis *et al*.(2014) chained geoprocessing services in a cloud computing environment to do on-the-fly geoprocessing.

The work discussed above proved that Web Processing Services have benefits that cannot be ignored. Web Processing Services can be used for alerting applications. A large amount of effort should be made to include Web Processing Services in the design of a geoprocessing system.

2.3.4 Geovisualisation

A large quantity of work was conducted on3D geovisualisation. Work was further conducted on the creation of 3D visualisations of wildfires. Various libraries such as WebGL, OpenSceneGraph and OpenGL were experimented with. Some visualisations were developed for desktop-based viewing and some for cloud-based viewing.

Over *et al*. (2010) generated 3D city models from OpenStreetMap data, this was a web-based solution. Hildebrandt el al. (2010) discussed the design of a 3D geovisualisation solution. They suggested that the system should be distributed, standards-based and service oriented. Resch *et al*. (2011) used WebGL to create a system that provides web-based 4D visualisation of marine geographic data. Resch *et al*. (2011) stated that information should be understandable. “Too much simplification might lead to a high level of abstraction in conveying complex spatio-temporal geographic processes. The more abstract the visualisation, the more disconnected from the physical world the visualisation is, and the more effort is required to interpret the information represented in the digital environment (*Rensch el al.,* 2011).”

Kim *et al*. (2014) integrated a spatial database management system, RESTful API and WebGL to create 3D visualisations of satellite images on smart devices. Prandi *et al*. (2014) created 3D city visualisations. Knoch *et al*. (2014) developed a solution to visualise 3D hydrological data on the web to gain a better understanding of the groundwater resources in New Zealand. Distributed data and processing resources were combined to generate an on-demand 3D visualisation of geological and hydrological data (Knoch *el al.*, 2014).

Heirring *et al*. (2011) developed a 3D campus information system by using WebGL. The users were able to navigate the terrain of the campus and interact with the buildings to obtain information. Zipf (2011) created a 3D viewer that displays terrain, buildings and points-of-interest. Feng *et al*. (2012) developed a system that experimented with a web-server, terrain-server, image-server and model-server. The data from the servers were combined and rendered with the WebGL engine.

Woo *et al*. (2014) combined several existing technologies to create a 3D geovisualisation developed for COMS (Korean Communication, Ocean and Meteorological Satellite) satellite images on smart phones and tablets. They integrated a spatial database management system (PostgreSQL and PostGIS), a RESTful application programming interface and WebGL to create this system. To represent satellite images, spatio-temporal objects were constructed within a database management system and a RESTful application programming interface was created for querying the database. Satellite images were represented in WebGL. Elevation data was used to provide a more realistic depiction of reality (Woo *et al*., 2014). Wu *et al*. (2010) proposed a Web Service Oriented Architecture based on a virtual globe for the creation of an 3D environment where urban planners can share information. This architecture is based on CityGML.

Castrillón *et al*. (2011) designed a wildfire forecasting application based on a fire simulation engine and a 3D virtual environment. The system took variables such as wind, vegetation and topography into consideration. This system enabled users to monitor fires and also provided the user with an indication of fire spread. The visualisations appear realistic. Yun *et al*. (2012) implemented a wildfire spread simulation system to visualise wildfire spread. The system is based on FARSITE (Fire Area Simulator) and Open Scene Graphs. A 3D wildfire scenario can be visualised with this system and several factors can make the simulation more realistic (terrain slope and wind).

From this subsection, it can be noted that a large amount of work was conducted towards the development of various kinds of 3D visualisations. From the few fire visualisations, none of the 3D fire visualisations were intended to be attached to wildfire notifications.

2.4 Chapter Summary

A high level overview of wildfires was provided in this chapter. The industry use case AFIS was introduced and a discussion on wildfire detection and on big geospatial data was included. Relevant material on distributed and high performance computing was also included. Sections on enterprise messaging (page 24), Web Processing Services (page 25) and 3D geovisualisation (page 27) formed part of this chapter. Related work was also discussed (page 28). This chapter gave the reader background information on portions of this research and highlighted the gaps.

A large volume of research has been conducted on cloud computing, enterprise messaging, Web Processing Services and geovisualisation separately. No research has been conducted in the optimisation of geoprocessing chains in software, only high performance distributed computing. 3D wildfire visualisations have been created before, but not with the intention of rapidly generating 3D wildfire visualisations and attaching them to wildfire notifications. This gap should be filled as it can contribute significantly to managing wildfires.

The following chapter provides the reader with information on the methodology followed during the research, the design of the geoprocessing chain and the experiment design of the research.

3 Chapter Three: Research Methods and Experiment Design

3.1 Chapter Overview

The previous chapter provided related information on specific topics from literature, including sections on wildfires (page 11), the Advanced Fire Information System (page 13), wildfire detection (page 15), big geospatial data (page 16), distributed and/or High Performance Computing (page 18), enterprise messaging (page 24), Web Processing Services (page 25) and geovisualisation (page 27). The objective was to provide the reader with an in-depth background of this research. A section on related work (page 28) was included and a gap was identified. No research was conducted that designed a rapid geoprocessing system that generates 3D wildfire context visualisations to attach them to wildfire notification messages.

This chapter discusses the research methodology followed during the research (page 36). This chapter also contains information that describes the abstract geoprocessing chain (page 39) and the data required for an abstract geoprocessing chain (page 39). An abstract geoprocessing chain separates the description and purpose of geoprocessing chain components from the implementation specifics. The components of the abstract geoprocessing chain were implemented. This chapter also provides the design of the research experiment (page 37). The reader will therefore gain an understanding of the high level design of the geoprocessing chain.

3.2 Research Methods

The research conducted falls within the positivistic research paradigm. The research was conducted impartially and with the objective to measure the performance of various geoprocessing techniques for an on-demand rapid geoprocessing system. Precise quantitative measurements or benchmarks were noted for various fire event processing implementations. Relationships amongst variables such as time and algorithmic style, time and architectural style and time and implementation types were observed. Graphs that represent the relationships amongst the variables were created and analysed to provide a more coherent understanding. The research can therefore be viewed as quantitative research.

Benchmarking was conducted in the pre-tests to determine which library would produce results in the fastest time for specific functions of the geoprocessing chain such as the "area-of-interest check", the "buffer and bounding box calculation operation" and the "get elevation operation". Implementations were benchmarked to determine what implementation such as loosely-coupled Web Processing Services, loosely-coupled function calls, tightly-coupled Web Processing Services and tightly coupled function calls, provided the fastest performance (produced results in the fastest time). Specifically, the time dimension was measured. Benchmarking was therefore used to obtain quantitative measurements to ensure the objectiveness of the research.

A prototype is a simplified system that can serve as an example for a complex system. The complex system is the Advanced Fire Information System. The prototype or experimental system focussed on spatial filtering, obtaining contextual variables and 3D visualisation generation. These specific aspects of the system were focussed on. The complex system can disseminate text messages and emails. A prototype (proof-of-concept prototype) was implemented to demonstrate such a system (rapid 3D fire visualisation rendering system) could work efficiently and effectively. The prototype was used to benchmark focussed aspects of such a system. The benchmarks of the pre-tests were required for the final implementation of the prototype. The benchmarks measured the speed of the system and only the implementations that produced output in the fastest time were chosen for the rapid geoprocessing system.

Two experiments were conducted, a commodity hardware-based experiment (experiment one) and a cloud-based experiment (experiment two). The experiments were conducted to determine what geoprocessing chain implementation would produce results (output-3D fire contextual visualisations) in the fastest time(which experimental inputs generated 3D fire contextual visualisations in the fastest time). The commodity hardware-based experiment was conducted in a controlled environment. The cloud-based experiment was conducted in a semi-controlled environment. Control was enforced as rigorously as possible, but during the experimentation process, there were external factors (variables) that had an impact beyond the control of an Amazon Web Services user. All of the external variables are not known. Current literature as discussed in Chapter Two illustrates some of the known variables. The uncertainty is due to the nature of public cloud computing, which is also one disadvantage of public cloud computing.

3.3 Experiment Design

The aim of this research was to find an optimal design to handle the geoprocessing of bursty datastreams of geolocated events into notifications with visualisations, sent to appropriate users in “demand-time” (as rapidly as possible).

The first step of the experiment design was to determine the structure or design of the geoprocessing chain. After the structure or design was determined, certain components in the geoprocessing chain were implemented in up to four alternative ways by using different software libraries or packages. The components were scripted in the Python programming language. Several timing tests were conducted for each of the four alternative implementations (the alternative implementations were benchmarked) to determine which style of geoprocessing component offered the fastest performance. These timing tests are referred to as pre-tests in this dissertation.

The first experiment was conducted on a single thread on commodity hardware. Web Processing Services were then set up for each stage of the geoprocessing chain using the fastest component implementations determined by the pre-tests. Timing evaluations were performed on the Web Processing Services in two configurations (architectural styles): a) tightly-coupled chaining- using method call chaining (Combination One)a)single detected wildfire event must be processed before the next detected wildfire event can be processed); and b) loosely-coupled chaining- using AMQP consumers/producers (Combination Three) (detected wildfire events are processed in parallel). Thereafter geoprocessing components were set up without Web Processing Services, also using the tightly-coupled (Combination Two) and loosely-coupled (Combination Four) configurations. The components utilised function calls instead of Web Processing Services. They were subjected to the same tests. The experiment was aimed at showing which configuration offered the fastest throughput of events to notifications with a visual payload.

The second experiment was conducted in a public cloud computing environment. Web Processing Services were set up for each stage of the geoprocessing chain using the fastest component implementations determined by the pre-tests. Timing evaluations were performed on the Web Processing Services in one configuration: loosely-coupled chaining - using AMQP consumers and producers. Geoprocessing components were then set up without Web Processing Services, further using the loosely-coupled configuration, and were then subjected to the same tests. The components utilised function calls instead of Web Processing Services. Only the loosely-coupled chaining configuration was tested due to the cloud computing environment. Cloud computing was considered because of horizontal scaling (adding computing resources on demand). Therefore, conducting tightly-coupled component tests was irrelevant.

Table 1: Combinations of Experimental Architectural Styles

	With WPSs	**Without WPSs (function call implementation)**
Tightly-Coupled Geoprocessing Chain Components	Combination One	Combination Two
Loosely-Coupled Geoprocessing Chain Components	Combination Three	Combination Four

Table 1 illustrates the combinations of experimental architectural styles.

3.4 Required Data(Abstract)

Table 2 lists the data required for generating the 3D visualisation and context. It describes all required datasets.

Table 2: Abstract Description of Required Data

Name	**Description**
Vegetation data	Data utilised for indicating vegetation types over a terrain and for indicating the vegetation at the location of a fire. The data gives an indication of fuel types.
Elevation data	Data utilised for building a realistic depiction of real world terrain and for extracting the elevation at the location of a fire. The elevation data gives an indication of the topography for fire spread. Emergency responders will benefit from elevation data.
Population data	Data utilised for indicating the population density over a certain area and for indicating the population density at the location of a fire. The data gives an indication of the number of people affected by a wildfire.
Land cover data	Data utilised for indicating land cover types over a certain area and for indicating the land cover type at the location of a fire. The data gives an indication of fuel types.

3.5 Geoprocessing Chain (Abstract)

Table 3 lists the abstract geoprocessing chain components. It describes the purpose for each of the components. Refer to *Figure 5 (page 47)* in Chapter Four to view the component corresponding to the component number in *Table 3 (page 40)*. Refer to *Table 6 (page 46)* for implementation details. *Figure 4 (page 40)* shows the functions of the components of the geoprocessing chain.

Table 3: Description of the Abstract Geoprocessing Chain Components

Component (refer to *Figure 5 on page 47*)	**Name**	**Description**
1	agpc_parser	Parses streaming data (lists of tuples) representing fire detection events into a JSON (JavaScript Object Notation) format in order to standardise the data format.
2	agpc_areaofinterest_checker	Determines if a fire event occurred within a set of area of interest geometries. If so, the event data is passed along to the next step of the geoprocessing chain. If a fire event did not occur within an area of interest, the fire event data is discarded.
3	agpc_bufferboundingbox_calculator	Derives a bounding box from a buffer of the event point location. The bounding box is used in extracting contextual data around an event to generate a 3D (three-dimensional) scene visualisation.
4	agpc_elevation_calculator	Calculates the elevation at the location of the fire from a DEM (Digital Elevation Model) (raster).
5	agpc_population_calculator	Calculates the population density at the location of the fire from a population GeoTIFF (raster).
6	agpc_landcover_calculator	Calculates the land cover at the location of the fire from a land cover GeoTIFF (raster).
7	agpc_cookiecutter	Cookie-cuts rasters and a shapefile according to the extent calculated by agpc_bufferboundingbox_calculator.
8	agpc_converter	Converts the rasters and shapefile that were cookie-cutted by agpc_cookiecutter to other formats.
9	agpc_html_generator	Generates a HTML (HyperText Markup Language) file that contains JavaScript code that will setup the 3D contextual visualisation of the scene of the fire.

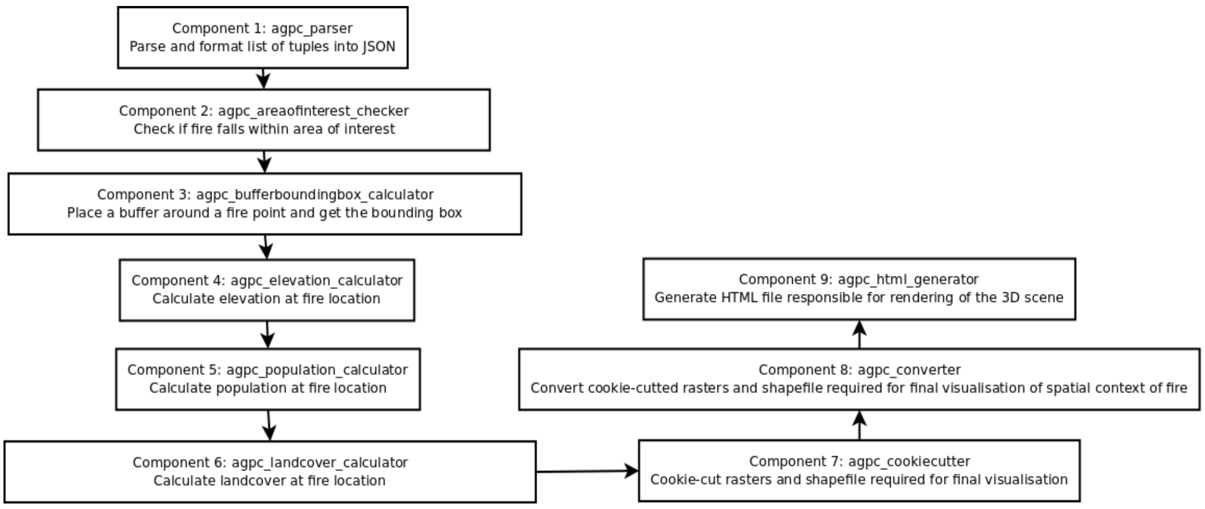

Component 1: agpc_parser
Parse and format list of tuples into JSON
Component 2: agpc_areaofinterest_checker
Check if fire falls within area of interest
Component 3: agpc_bufferboundingbox_calculator
Place a buffer around a fire point and get the bounding box
Component 4: agpc_elevation_calculator
Calculate elevation at fire location
Component 5: agpc_population_calculator
Calculate population at fire location
Component 6: agpc_landcover_calculator
Calculate landcover at fire location
Component 7: agpc_cookiecutter
Cookie-cut rasters and shapefile required for final visualisation
Component 8: agpc_converter
Convert cookie-cutted rasters and shapefile required for final visualisation of spatial context of fire
Component 9: agpc_html_generator
Generate HTML file responsible for rendering of the 3D scene

Figure 4: High-level description of geoprocessing chain components

3.6 Chapter Summary

This chapter describes the related methods used during the research. The research can be categorised into the positivistic research paradigm. Various implementations were benchmarked and quantitative measurements were noted by conducting various experiments. The benchmarks informed the design of the prototype created. Various "flavours" of the prototype were tested to reach a conclusion on the most optimal design for a rapid 3D wildfire visualisation geoprocessing system. The chapter further provides details on the design of the abstract geoprocessing chain, the required data and the design of the research experiment. The following chapter will provide an in-depth discussion of the implementation details.

4 Chapter Four: Research Implementation

4.1 Chapter Overview

The preceding chapter provided the reader with details on the methodology followed during this research. Various implementations and geoprocessing chain "flavours" were benchmarked to determine the most optimal design of a 3D wildfire context visualisation generating geoprocessing system. A prototype was created that does not contain all of the functionality of AFIS (the industry use case). It further provided the reader with a detailed description of an abstract geoprocessing chain and the data required for an abstract geoprocessing chain. The design of the research experiment was further included in Chapter Three.

This chapter focuses on the implementation details of the geoprocessing chain such as the data required for the final implementation and the software libraries used for the final implementation. Details are provided on the experimental implementations. Details such as the design of the implemented processing chain, how benchmarking was conducted and details on the pre-tests are provided. The four architectural styles are additionally discussed in this chapter. This chapter includes discussions on experiment one and experiment two. The discussions relate to hardware and software required for the two experiments and the methods used to conduct the experiments.

4.2 Required Data (Implementation)

Two types of fire data were utilised for this research to provide locations of active fires. The two types of data were utilised because the industry use case AFIS utilises them. The first type of data was derived from detections of the SEVIRI (Spinning Enhanced Visible & Infrared Imager) sensor aboard ESA's Meteosat 8 geostationary satellite. SEVIRI data has a spatial resolution of approximately four kilometres and a temporal resolution of 15 minutes.

The second type of data was derived from MODIS (Moderate Resolution Imaging Spectroradiometer) sensor detections. NASA's polar orbiting Terra and Aqua satellites are equipped with these sensors. MODIS data has a spatial resolution of approximately one kilometre and a temporal resolution of six hours if both satellite orbits are used.

The data used for the contextual variables was acquired from the World Wide Web. This was the vegetation data, elevation data, population data and land cover data. *Table 4 (page 43)* lists the data required for the implemented geoprocessing chain. A description of the data is further provided.

Table 4: Description of Data Required for Implementation

Name	Description
Vegetation data-National Vegetation Map 2006	The vegetation data, vector data, was saved in a shapefile format. 440 Zonal and azonal vegetation types are contained within this dataset. The vegetation map was compiled in 2006. Several organisations contributed to this map over an extended period. It provides vegetation data for South Africa, Lesotho and Swaziland. Fires require oxygen, fuel and heat to occur. Vegetation data was important as it gave an indication of the types of fuel. Certain types of vegetation are more prone to burn that other types of vegetation (South African National Biodiversity Institute, 2006).
Elevation data-SRTM90	For elevation values, a SRTM (NASA Shuttle Radar Topography Mission) 90m DEM was used. The DEM provides high quality elevation data for the entire world. It has got a 90m resolution at the equator and the tiles are mosaiced as 5º x 5º tiles (they are available for download). SRTM is available as 3 arc second DEMs. The vertical error of these DEMs is less than 16m. No data values are yielded for places with water or places with large voids. Elevation data was used to build the terrain of the 3D visualisation. Terrain data was required to show the topography of the land. Users can get an indication of potential fire behaviour (because fires spread fast uphill). The terrain data can give emergency responders a better idea of the terrain, it can thus help them with planning (NASA Land Processes Distributed Active Archive Centre, 2013).
Population data-Worldpop South Africa Population	Population data was obtained from the WorldPop project website. The population data(raster data) was saved in a geoTIFF format. GlobeCover was used to capture these datasets as well as census data from various countries. The spatial resolution of GlobeCover was resampled to 100m and the urban class that caused an over-estimation was removed. The unit of the dataset is population per square kilometre. The 2015 estimates dataset was used. Population data was important as a user was able to see how many people was affected by a certain fire (Worldpop, 2013).
Land cover data-National Land Cover 2000 and 2009	Land cover data and vegetation data was supplied by the South African National Biodiversity Institute. The land cover data(raster data) was saved in geoTIFF format. Land cover indicates the impact on biodiversity. The National land cover dataset of South Africa contains eight extensive classes, namely: natural, cultivated, degraded, urban built-up, water bodies, plantations, mines and missing data. The NLC2009 (National Land Cover 2009) dataset was derived from various data sources such as municipal land cover data, provincial land cover data, ARC cultivation filled boundaries data and the ESKOM Spot 5 building count dataset. Land cover data was important as it gave an indication of the types of fuel. Certain types of vegetation are more prone to burn that other types of vegetation (South African National Biodiversity Institute, 2009).

4.3 Software

Table 5: High Level Description of Used Software Libraries or Packages

Name	**Description**
Fiona	Python package providing Python interfaces to OGR functions (Python Community, 2013). It can be used to read and write spatial data files. It relies on Python types. Fiona was used for the pre-tests.
GDAL	C++ translator library for geospatial raster data formats (OSGeo, 2013). Library used to read and write raster data. GDAL was used for the pre-tests as well as the final implementation of the geoprocessing chain.
OGR	C++ translator library for geospatial vector data formats (OSGeo, 2013). Library used to read and write vector data. OGR was used for the pre-tests and the final implementation of the geoprocessing chain.
Pika	Python implementation of the AMQP protocol (Python Community, 2014). It was developed with the intention to work with RabbitMQ. Pika was used to work with RabbitMQ (refer to loosely-coupled implementations)
PostGIS	Spatial database extender for PostgreSQL. Allows queries on geographic objects in SQL (OSGeo, 2013). PostGIS was used for the pre-tests.
Pyproj	Python package that can be used to perform cartographic transformations and geodetic computations (Python Community, 2014). It is based on the open-source geometry engine, GEOS. Pyproj was used to reproject data.
PyWPS	Python implementation of a Web Processing Service (Cepicky, 2013). PyWPS was the preferred library for the WPSs.
RabbitMQ	Open source enterprise messaging system based on the AMQP standard (Pivotal Software, inc n.d.). RabbitMQ was used for the enterprise messaging system.
Rtree	Python package providing advanced spatial indexing features (Python Community, 2012). Rtree was used for the pre-tests.
Shapely	Python package for manipulation and analysis of planar geometric objects (Python Community, 2012). Shapely was used for the pre-tests.
Three.js	Three.js is a JavaScript 3D library which makes WebGL applications easier to develop (MrDoob, 2014). WebGL is a cross-platform Application Programming Interface (API) that can be used to create 3D graphics in a web browser. It is based on OpenGL (Khronos, 2012). Three.js was used to create the 3D WebGL visualisation.

Table 5 lists software packages or libraries used. A high level description is given for each of the libraries. It is also mentioned where the software packages or libraries was used in the scripts.

4.4 Experimental Implementations

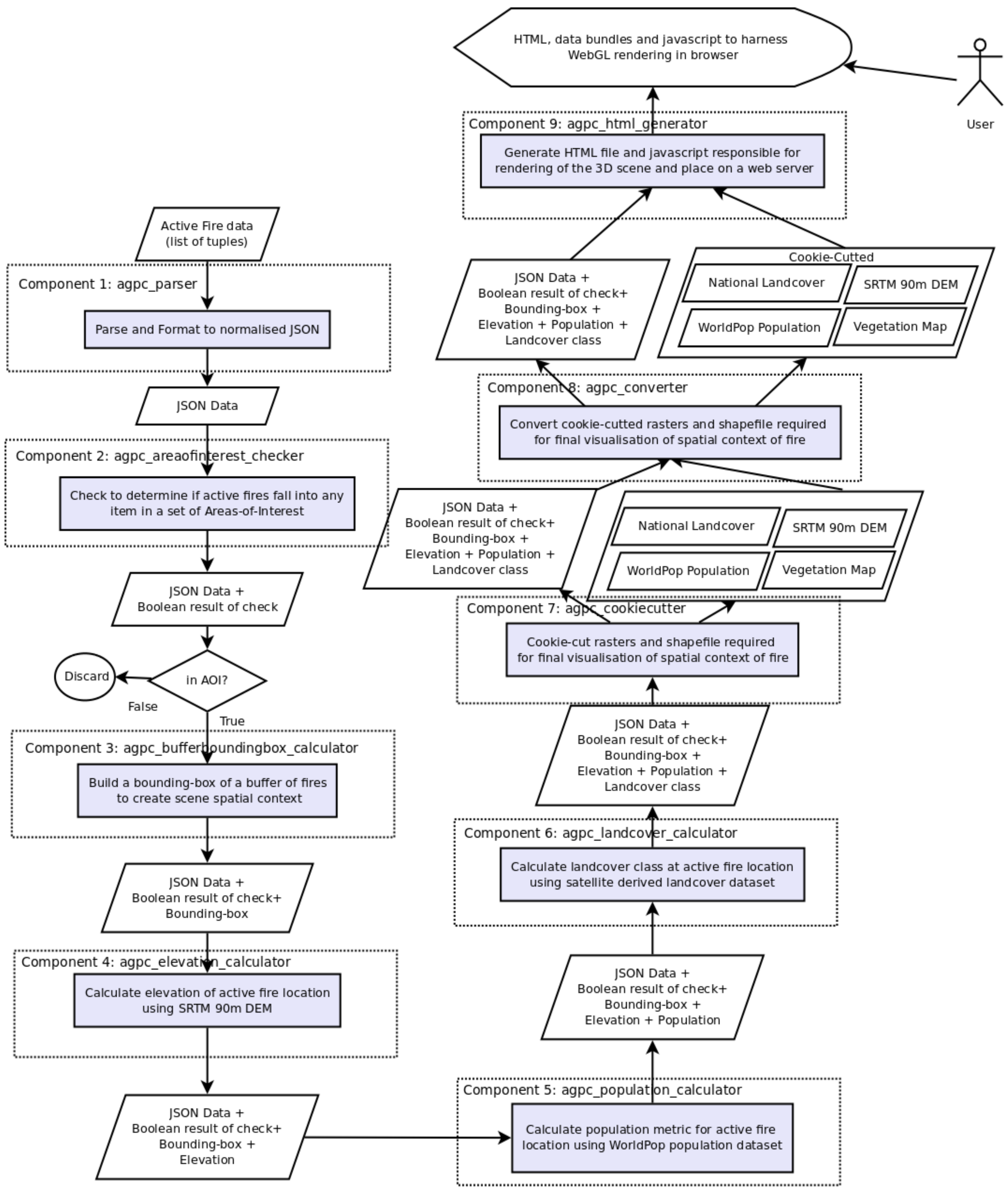
4.4.1 Geoprocessing Chain(Implementation)

Table 6 lists the components of the implemented geoprocessing chain. It provides the function of every component and lists the software used by each. *Figure 5 (page 47)* is a diagram of the geoprocessing chain. Refer to the section on the abstract geoprocessing chain design (Chapter Three, Section 3.5, page 39).

Table 6: Description of the Implemented Geoprocessing Chain Components (Experimental)

Components described in this implements abstract components described in Table 3 (page 40) of Chapter Three.

Name	**Implements**	**Description**	**Software**
egpc_parser	agpc_parser	1) Receives incoming fire data. 2) Parses streaming fire event data that is a list of tuples are parsed into a JSON format.	Python
egpc_areaofinterest_checker	agpc_areaofinterest_checker	1) Loads a shapefile that contains the defined areas-of-interest into memory with OGR. 2) Constructs a point that represents a fire at a specific location. 3) Conducts a check to determine if the point location of the fire intersects with the polygons in the shapefile that contains the areas-of-interest.	Python OGR
egpc_bufferboundingbox_calculator	agpc_bufferboundingbox_calculator	1) Constructs a point that represents a fire at a specific location. 2) Places a buffer around the point. 3) Calculates the bounding box of the buffer with OGR.	Python OGR
egpc_elevation_calculator	agpc_elevation_calculator	1) Loads a elevation GeoTIFF into memory with GDAL. 2)Extracts the pixel value at the location of the fire.	Python GDAL
egpc_population_calculator	agpc_population_calculator	1) Loads a population GeoTIFF into memory with GDAL. 2) Extracts the pixel value at the location of the fire.	Python GDAL
egpc_landcover_calculator	agpc_landcover_calculator	1) Loads a land cover GeoTIFF into memory with GDAL. 2) Extracts the pixel value at the location of the fire.	Python GDAL
egpc_cookiecutter	agpc_cookiecutter	1) Takes rasters and a shapefile (contextual data) as input and cookie-cuts them according to the extent calculated by egpc_bufferboundingbox_calculator. Uses OGR and GDAL.	Python OGR GDAL Pyproj
egpc_converter	agpc_converter	1) Takes the cookie-cutted data as input and converts the cookie-cutted rasters and shapefile to PNG files.	Python OGR GDAL
egpc_html_generator	agpc_html_generator	1) Takes all the data passed along the geoprocessing chain and generates a HTML file that creates a 3D context visualisation and a information box with static data.	Python Javascript Three.js



HTML, data bundles and javascript to harness WebGL rendering in browser
User
Component 9: agpc_html_generator
Generate HTML file and javascript responsible for rendering of the 3D scene and place on a web server
Active Fire data (list of tuples)
Component 1: agpc_parser
Parse and Format to normalised JSON
JSON Data
Component 2: agpc_areaofinterest_checker
Check to determine if active fires fall into any item in a set of Areas-of-Interest
JSON Data + Boolean result of check
in AOI?
Discard
False
True
Component 3: agpc_bufferboundingbox_calculator
Build a bounding-box of a buffer of fires to create scene spatial context
JSON Data + Boolean result of check+ Bounding-box
Component 4: agpc_elevation_calculator
Calculate elevation of active fire location using SRTM 90m DEM
JSON Data + Boolean result of check+ Bounding-box + Elevation
Component 5: agpc_population_calculator
Calculate population metric for active fire location using WorldPop population dataset
JSON Data + Boolean result of check+ Bounding-box + Elevation + Population
Component 6: agpc_landcover_calculator
Calculate landcover class at active fire location using satellite derived landcover dataset
JSON Data + Boolean result of check+ Bounding-box + Elevation + Population + Landcover class
Component 7: agpc_cookiecutter
Cookie-cut rasters and shapefile required for final visualisation of spatial context of fire
National Landcover
SRTM 90m DEM
WorldPop Population
Vegetation Map
Component 8: agpc_converter
Convert cookie-cutted rasters and shapefile required for final visualisation of spatial context of fire
Cookie-Cutted

Figure 5: Tightly-Coupled Geoprocessing Chain Components

4.4.2 Benchmarking

Several events were required to be processed. Various processing implementations were benchmarked. The pre-tests included the processing of 1000 fire events, ten times. The tests of experiment one and experiment two included the processing of fire events in 1000 intervals. The tests started with the processing of 1000 fire events and ended with the processing of 10 000 fire events. The time that elapsed to execute a script (process several events) was determined by using the python time module (benchmarking was achieved by utilising the python time module). A queue listener was created to monitor the number of messages in the messaging queues for the loosely coupled tests. Benchmarking was initiated as soon as the fire events were produced. The listener indicated how many messages there were in the messaging queues. As soon as there were no messages left in the queue, the result of a test was recorded.

4.4.3 Pre-Tests

Pre-tests were conducted to determine which implementation of a geoprocessing chain component provided the fastest performance. *Table 7* describes the pre-tests conducted. The pre-tests were conducted to inform the final design of the geoprocessing chain.

Only three pre-tests were conducted because the function of the third test was similar to components that followed in the geoprocessing chain. The primary purpose of the tests is:

Test One: Accepts fire data in a JSON format, extracts the point location of the fire and checks if the point falls within an AOI (Area of Interest).
Test Two: Accepts fire data in a JSON format, extracts the point location of the fire, places a 2.5 km buffer around the fire point, calculates the bounding box of the buffer.
Test Three: Accepts fire data in a JSON format, extracts the point location of the fire, queries the elevation from a DEM at the fire location.

Table 7: Description of the Pre-Tests

Name	**Implements**	**Description**	**Software**
pgcp_areaofinterest_check_rtree	agpc_areaofinterest_check	1) Created indices that represents the area of interest. 2) Constructed fire point location 3) Checked if the point intersected with an index.	Python Rtree
pgcp_areaofinterest_check_ogr	agcp_areaofinterest_check	1) A shapefile (area of interest) was loaded into memory. 2) Constructed fire point location 3) Checked if the geometry of the shapefile contained the point.	Python OGR
pgcp_areaofinterest_check_fiona+shapely	agcp_areaofinterest_check	1) Loaded a shapefile (area of interest). 2) Determined the bounds of the area of interest. 3) Constructed fire point location 4) Checked if the point was contained within the bounds (optimised with prepared geometries).	Python Fiona + Shapely
pgcp_areaofinterest_check_postgis	agcp_areaofinterest_check	1) Imported the shapefile (area of interest) to database. 2) Constructed fire point location. 3) Connected to database, checked if geometry of point intersected with the geometry of the shapefile (area of interest).	Python PostGIS
pgpc_bufferboundingbox_calculator_ogr	agpc_bufferboundingbox_calculator	1) Constructed fire point location 2) Placed a buffer around that point. 3) Calculate extent around the buffer.	Python OGR
pgpc_bufferboundingbox_calculator_fiona+shapely	agpc_bufferboundingbox_calculator	1) Constructed fire point location. 2) Created a buffer around the point. 3) Calculate extent around the buffer.	Python Fiona+Shapely
pgpc_bufferboundingbox_calculator_postgis	agpc_bufferboundingbox_calculator	1) Constructed fire point location. 2) Created a buffer around the point. 3) Calculate extent around the buffer.	Python PostGIS
pgpc_elevation_calculator_gdal	agpc_elevation_calculator	1) Loaded the GeoTIFF into memory. 2) Extracted a pixel value at the specific location of the fire.	Python GDAL
pgpc_elevation_calculator_postgis	agpc_elevation_calculator	1) Imported the DEM into a database using PostGIS raster. 2) Constructed fire point location. 3) Selected value from database where the point intersects with the DEM.	Python PostGIS

Process execution times were collated and compared for each library for all of the tests to select the optimal algorithmic style for every component in the geoprocessing chain. The process execution was timed for 1000 fire events repeated 10 times to account for vagaries in system load.

4.4.4 Architectural Styles

4.4.4.1 Combination One: Web Processing Service Implementation with Tightly-Coupled Components

PyWPS was utilised as the implementation of the Web Processing Service (Cepicky 2013). A tightly-coupled Web Processing Service chain class was created. Several methods were implemented (Section 4.4, page 44). Each method invoked a different Web Processing Service that performed an operation on the input data and returned a result. Urllib was used to invoke a Web Processing Service. Urllib provides a high level interface to retrieve data across the internet (Python Community 2013). The chaining was designed so that each method called the following method in the geoprocessing chain. Every result was passed along the geoprocessing chain by nesting method calls. A class instance was created and the first method to parse the data from the fire detection source was called. This initiated the geoprocessing chain. The result of a process (returned result of a method) was used as input parameter (input data) for the next process executed within the geoprocessing chain. Nested method calls were used.

4.4.4.2 Combination Two: Function Call Implementation with Tightly-Coupled Components

A tightly-coupled geoprocessing chain class was created. Several methods were implemented (each method represents a component in the geoprocessing chain). Each method called the next method in the geoprocessing chain. Every result was passed along the geoprocessing chain by nesting method calls. A class instance was created and the first method to parse the data from the fire detection source was called. This initiated the geoprocessing chain. The result of a process (returned result of a method) was used as input parameter (input data) for the next process executed within the geoprocessing chain. Nested method calls were used. The Web Processing Service processes included XML parsing that used the xml.dom.minidom library. To test the process chaining without using Web Processing Services, XML parsing was accounted for by writing our result into XML and parsing the XML with the xml.dom.minidom library.

4.4.4.3 Combination Three: Web Processing Service Implementation with Loosely-Coupled Components

Classes were created to represent various components of the geoprocessing chain. Every component utilised a Web Processing Service. Active fire event data is produced and parsed into a JSON format by a consumer that listens on the queue that the fire data is produced on. From here on the JSON data is consumed and used by a geoprocessing chain component. Data is added to a JSON string and passed along by the production of the JSON string onto another queue. The process is repeated for every component of the geoprocessing chain until the HTML file that sets up the context visualisation is generated. Refer to *Figure 6* (page 52) to see the diagram of the loosely-coupled geoprocessing chain. More than one fire detection event moved through the entire geoprocessing chain (from process 1 up to process 9 in the diagram) at a time.

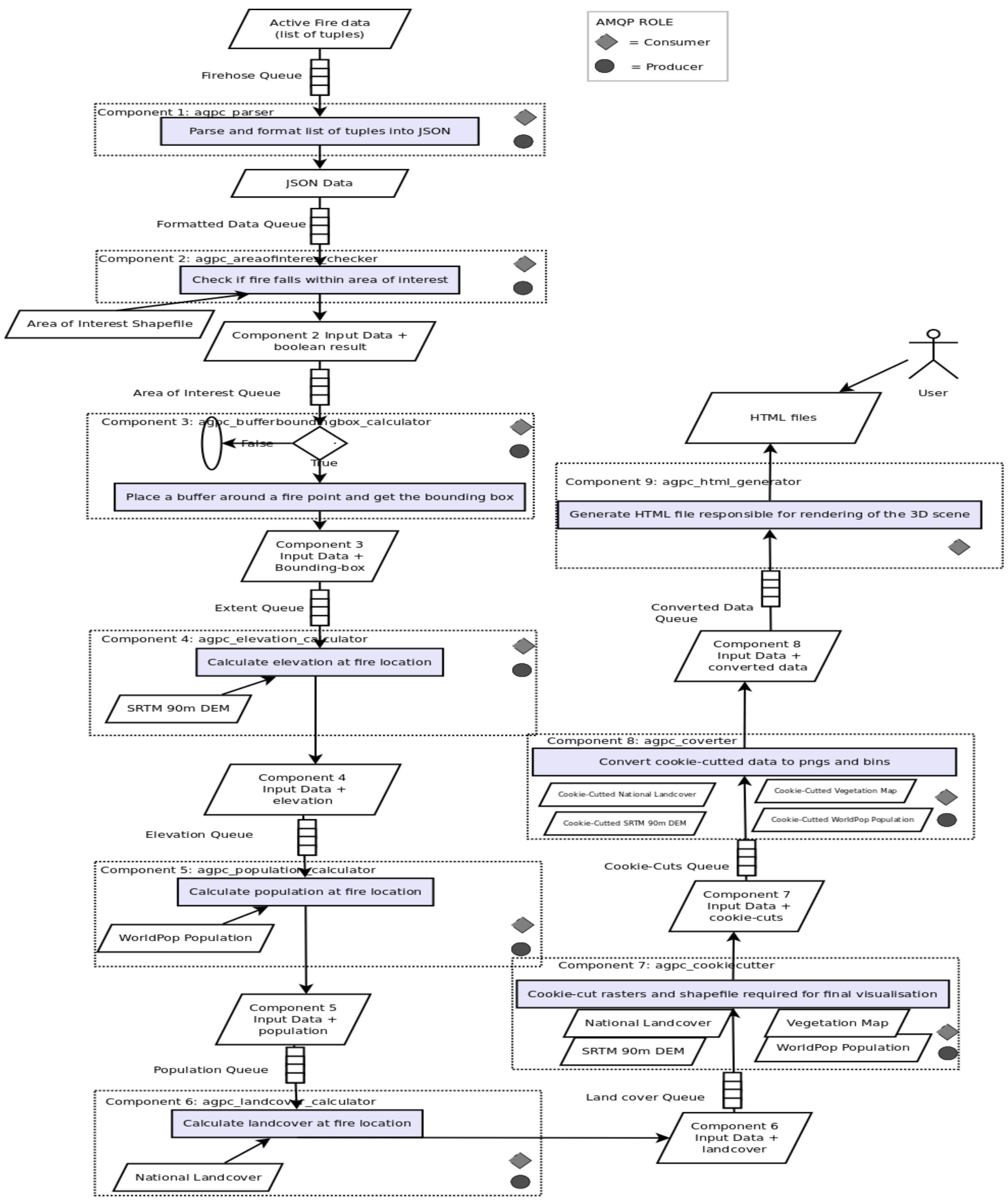

Active Fire data (list of tuples)
Firehose Queue
AMQP ROLE
= Consumer
= Producer
Component 1: agpc_parser
Parse and format list of tuples into JSON
JSON Data
Formatted Data Queue
Component 2: agpc_areaofinterest_checker
Check if fire falls within area of interest
Area of Interest Shapefile
Component 2 Input Data + boolean result
Area of Interest Queue
Component 3: agpc_bufferboundingbox_calculator
False
True
Place a buffer around a fire point and get the bounding box
Component 3 Input Data + Bounding-box
Extent Queue
Component 4: agpc_elevation_calculator
Calculate elevation at fire location
SRTM 90m DEM
Component 4 Input Data + elevation
Elevation Queue
Component 5: agpc_population_calculator
Calculate population at fire location
WorldPop Population
Component 5 Input Data + population
Population Queue
Component 6: agpc_landcover_calculator
Calculate landcover at fire location
National Landcover
Component 6 Input Data + landcover
Land cover Queue
Component 7: agpc_cookiecutter
Cookie-cut rasters and shapefile required for final visualisation
National Landcover
Vegetation Map
SRTM 90m DEM
WorldPop Population
Component 7 Input Data + cookie-cuts
Cookie-Cuts Queue
Component 8: agpc_coverter
Convert cookie-cutted data to pngs and bins
Cookie-Cutted National Landcover
Cookie-Cutted Vegetation Map
Cookie-Cutted SRTM 90m DEM
Cookie-Cutted WorldPop Population
Component 8 Input Data + converted data
Converted Data Queue
Component 9: agpc_html_generator
Generate HTML file responsible for rendering of the 3D scene
HTML files
User

Figure 6: Loosely-Coupled Geoprocessing Chain Components

4.4.4.4 Combination Four: Function Call Implementation with Loosely-Coupled Components

Classes were created to represent various components of the geoprocessing chain. Active fire event data is produced and parsed into a JSON format by a consumer that listens on the queue that the fire data is produced on. From here on the JSON data is consumed and used by a geoprocessing chain component. Data is added to a JSON string and passed along by the production of the JSON string onto another queue. The process is repeated for every component of the geoprocessing chain until the HTML file that sets up the context visualisation is generated. Refer to *Figure 6* (page 52) to see the diagram of the loosely-coupled geoprocessing chain. More than one fire detection event moved through the entire geoprocessing chain (from process 1 up to process 9 in the diagram) at a time. The Web Processing Service processes included XML parsing that used the xml.dom.minidom library. To test the process chaining without using Web Processing Services, XML parsing was accounted for by writing our result into XML and parsing the XML with the xml.dom.minidom library.

4.4.5 Experiment One - Single Thread on Commodity Hardware

4.4.5.1 Hardware

The tests were conducted on a laptop with an Intel Core i7, quad core CPU (Central Processing Unit) with four multi-threaded cores, therefore, eight cores running at 2.10GHz with 8GB of RAM (Random Access Memory).

4.4.5.2 Software

Ubuntu 12.04 was used as the operating system, simply because it was the latest Ubuntu version with long-term support at the time when this research started. Python was used as the programming language of choice as parts of the industry use case, AFIS, are also implemented in Python. The software described in Section 4.3 (page 44) was utilised for the final implementation of the geoprocessing chain. Zoo WPS was also experimented with when it was observed that the performance of PyWPS was slow.

The performance of the PyWPS and Zoo WPS implementations were compared. As the performance improvement of Zoo WPS was almost negligible compared to the performance of PyWPS, it was decided to not continue experimenting with Zoo WPS as it would not have been beneficial.

Initially, Cesium was used as the library to create 3D visualisations. Cesium is a free virtual globe and map engine, implemented in WebGL (AGI, 2013). Initially, Cesium was used as the technology for creating the 3D wildfire visualisations. The wildfire data had to be converted to CZML (JSON schema) to be visualised on the globe. Several layers including WMS layers were included as well. Cesium has streaming capabilities and it has its own terrain provider that utilises the SRTM90 and GTOPO30 datasets (AGI, 2013). It was observed that tiles took too long to load. Thereafter it was decided that Three.js would be used for the creation of the 3D visualisations.

Three.js is a JavaScript 3D library, which makes WebGL applications easier to develop (MrDoob, 2014). Rasters cannot directly be used by WebGL, thus raster data had to be read as arrays to display in the web browser. Several layers were added. Three.js is ideal to use if one is required to present fires, because particle systems can be created.

4.4.5.3 Methods

The geoprocessing chain was set up in a loosely-coupled style and in a tightly-coupled style by using Web Processing Services and by using function calls (without Web Processing Services). Therefore, four combinations or configurations were set up; they are listed in *Table 8*. The execution of the entire geoprocessing chain was timed or benchmarked several times. The execution of the geoprocessing chain was timed for each of the configurations. The results for each of the four configurations were compared against each other. The test was conducted for each one of the four configurations 10 times, from 1000 up to 10 000 fire events were used as input in intervals of 1000.

Table 8: Combinations of Experimental Architectural Styles

	With WPSs	**Without WPSs (function call implementation)**
Tightly-Coupled Geoprocessing Chain Components	Combination One	Combination Two
Loosely-Coupled Geoprocessing Chain Components	Combination Three	Combination Four

4.4.6 Experiment Two - Cloud Computing Environment

4.4.6.1 Hardware

The distributed tests of experiment one (loosely-coupled components) were repeated in a cloud computing environment. Amazon Web Services was utilised as the public cloud service provider. EC2 (Elastic Compute Cloud) micro-instances with one virtual CPU and 0.6GB RAM were used as the virtual machines. The physical processor of a micro-instance belongs to the Intel Xeon family and has a clock speed of 2.5 GHz but it can boost (over-clock) up to 3.3GHz. The Amazon S3 (Simple Storage Service) was used to store the resources in the cloud (Amazon Web Services, n.d.).

4.4.6.2 Software

The experimental system was implemented using the Python programming language, because parts of the industry use case, AFIS, are implemented in Python and because it was used in experiment one. Ubuntu 14.04 was used as the operating system. Ubuntu 14.04 was used because Amazon Web Services offered it and it is the latest Ubuntu version with long-term support at this time. The software described in Section 4.3 (page 44) was utilised for the final implementation of the geoprocessing chain.

4.4.6.3 Methods

Amazon Web Services was chosen as the public cloud service provider. Amazon Web Services presents a user with computing resources that can be used for application development and it utilises a pay-as-you-go structure. A virtual server can be rented and used in a similar fashion to a physical server. The virtual server runs on a network managed by Amazon Web Services. During peak times or high demand-time, the virtual server scales into multiple servers automatically. Amazon Web Services provides advantages such as APIs, horizontal scaling of resources, per hour billing and instance variations.

Amazon Web Services provides several services (Amazon Web Services, n.d.). Only a small fraction of these services were utilised for this research. The first service of interest is the Amazon Elastic Compute Cloud (EC2). It provides an opportunity to launch a virtual server. The virtual server is similar to a physical server, but can automatically scale horizontally (Amazon Web Services, n.d.). The second service of interest is the Amazon Relational Database Service (RDS). This allows a person to run a PostgreSQL database engine on Amazon Web Services (Amazon Web Services, n.d.). The third service of interest is the Amazon Simple Storage Service (S3). The S3 exists for the storage and retrieval of digital files (Amazon Web Services, n.d.). The fourth service of interest is the Amazon Elastic Block Store (EBS). This provides a persistent file system used to store created Amazon EC2 instances, even if the instances were terminated (Amazon Web Services, n.d.). There are two services that can aid with cost saving - the auto scaling service and load balancing service. The auto scaling services add and remove virtual servers according to the demand in traffic. The load balancing service is responsible for distributing traffic amongst multiple virtual servers (Amazon Web Services, n.d.).

A t2 Amazon EC2 instance is a low cost general purpose cloud instance type designed to provide a baseline level of CPU performance but can burst above the baseline level when required. Every t2 Amazon EC2 instance continuously receives CPU credits. At startup the amount of credits per instance is 30 (restart). A credit is equal to one virtual CPU being utilised at 100% of the capacity for one minute. Credits are accumulated when an instance is idle and credits are consumed whenever an instance is active. A set rate of six credits per hour can be earned at millisecond-level resolution. When an instance is in an idle state (CPU requires less resources than its baseline performance), the unused credits are stored in the credit balance for 24 hours (the maximum balance for a t2.micro-instance is 144 CPU credits). When an instance is in a running state (CPU requires more resources than its baseline performance), credits are consumed from the credit balance to burst the CPU utilisation up to 100% (Amazon Web Services, n.d.).

More credits are equal to longer bursts beyond the CPU's baseline performance. Unfortunately, credits do expire after 24 hours of acquisition. The credit balance is non-persistent between instance stops and starts. Stopping an instance causes it to lose its credit balance and restarting an instance causes it to receive its initial credit balance (Amazon Web Services, n.d.).

In a situation where all the credits are used, the performance remains at the baseline performance level. The CPU performance is gradually lowered to the baseline performance level over a 15 minute interval as credits are consumed. The performance is gradually lowered to prevent a steep drop in performance (Amazon Web Services, n.d.).

T2.micro-instances are on-demand instances. Billing occurs at an hourly rate and a partial instance hour is billed as a full hour (Amazon Web Services, n.d.). When an instance is stopped, it will cause the shutdown of the instance. An AWS user will not get charged for the hourly usage of a stopped instance although charging will occur for Amazon EBS volumes (Amazon Web Services, n.d.). Every time a stopped instance is started, Amazon will charge a full instance hour (even if it is restarted multiple times within a single hour). When an instance is explicitly restarted, Amazon will not charge a full instance hour (Amazon Web Services, n.d.) and the instance will not receive the initial amount of CPU credits.

The utilisation of public clouds was heavily constrained by costs. If cloud instances run for too long, the cost may become too high. If too little cloud instances get provisioned, generating 3D wildfire context visualisations might take too long. This has a very high impact on the research as it might have a drastic effect (negative) on the results. This can be mitigated by doing careful planning, such as finding a good balance between cost and performance.

An Amazon Web Services account was created to receive free tier privileges. Every Amazon Web Services customer receives free privileges within the first year (12 months) of registration. Any of 18 Amazon Web Services products or services can be used for free but within certain usage limits. Services or products of interest for this experiment were the EC2 (Elastic Compute Cloud), S3 (Simple Storage Service), RDS (Relational Database Service) and EBS (Elastic Block Store). A customer will get 750 hours of Linux t2.micro instance usage free per month. With the S3, a customer will receive 5GB of standard storage free per month plus 20000 get requests and 2000 put requests. For the RDS a customer will receive 750 hours of micro instance usage per month for free. A customer will also receive 20GB of storage and backups, and 10000000 I/Os (input and output operations) free. A customer will receive 30GB of EBS storage for free, along with 2000000 I/Os.

Thereafter, an Amazon S3 bucket (georesources) was created to store all the resource files (Oregon region). Amazon Web Services are hosted in multiple locations worldwide. A region is a specific geographic area. Every region comprises multiple zones. Amazon Web Services prefer instances to be hosted in multiple zones for failover. Keeping costs as low as possible was a challenge. Data transfer in different zones costs money, whilst data transfer in the same zone is free.

Resource files were transferred to the Amazon S3 bucket named Georesources from South Africa. This was a time-consuming process (it took approximately 60 minutes) due to the difference in region. The file permissions were changed so that the cloud instances may gain access to the files.

EC2 instances (in the Oregon region) were created for each component of the geoprocessing chain. All of the required packages and software were installed on each respective instance. Amazon Web Services use public-private key authentication for security purposes. AMIs (Amazon Machine Images) were created for every component. This makes the setup of an instance less tedious as software does not have to be reinstalled on every machine. An AMI is an image of the operating system (customised Ubuntu 14.04 in this case).

Enforced security rules had to be altered (custom TCP for ports 5432, 5672, 15672 and ssh for port 22)for the instances to communicate with each other. Amazon has a security mechanism for protecting instances, and is similar to a rule-based firewall. A user can specify which machines (with IP addresses in specific regions) can connect to specific ports. Required resource files were downloaded to their specific instances from the S3 bucket. This is faster and cheaper than secure copying the resource files from a local machine due to the difference in region. Cloud resource usage in certain regions are more expensive than resource usage in other regions. At the time that the second experiment started, Oregon was the cheapest region (us-west-2b).

If free tier privileges had not been available, the costs would have been as follows:
EC2 micro-instance usage: 58 instances * R0.14 = R8.12 per hour * 40 stop-starts = R324.8
EBS storage usage: 58 instances * 8 GB = 464 GB * R1.07 – R496.48 per GB per month
S3 storage: 97 GB * R0.32 (for 1st terrabyte) = R0.32
Put/Copy/Post/List Requests = 110000/1000 * R0,05 = R5.5
Get/Other Requests = 110000/10000 * R0.4 = R0.44

Total: R 827.54 (strict control of computing hours, weak EC2 instances)

Instance usage is not cheap and if larger instances were required the cost would have been much higher. Computing hours should not be wasted and should be controlled for cost saving purposes. EBS storage is also expensive but was required for the number of micro instances used. S3 storage is very cheap considering the cost is for one terabyte. Data transfer within the same region was free, but if instances were created in different regions, it would have been costly.

No pre-tests were conducted. OGR was the library of choice for the implementation of the geoprocessing chain because it was the most performant library in experiment one. Tests were conducted by chaining components of the geoprocessing chain that utilises Web Processing Services in a loosely-coupled style and chaining the components of the geoprocessing chain that utilises function calls (without using Web Processing Services), in a loosely-coupled style. The data was stored on the computing nodes and in a S3 bucket. Checks were conducted to determine if accessing the resources from the computing nodes (cloud instances) was more efficient than accessing resources from an Amazon S3 bucket. Up to six nodes were utilised for every component of the geoprocessing chain. The results of Chapter Five are included for four consumers per geoprocessing chain component (*Figure 7*) and six consumers per geoprocessing chain component (*Figure* 8).

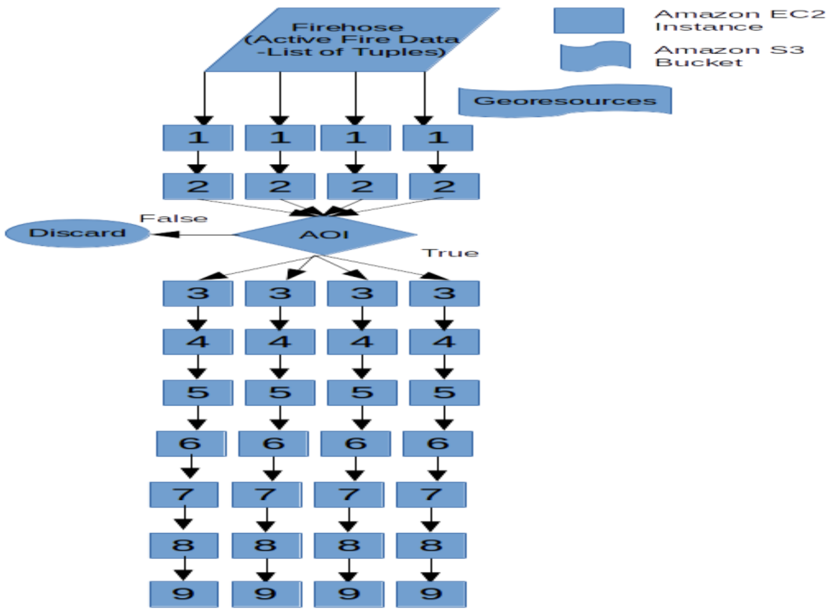


Figure 7: Cloud configuration-four consumers per geoprocessing chain component.

Refer to Table 3 (page 40) to check the corresponding component number to see the function of a component.

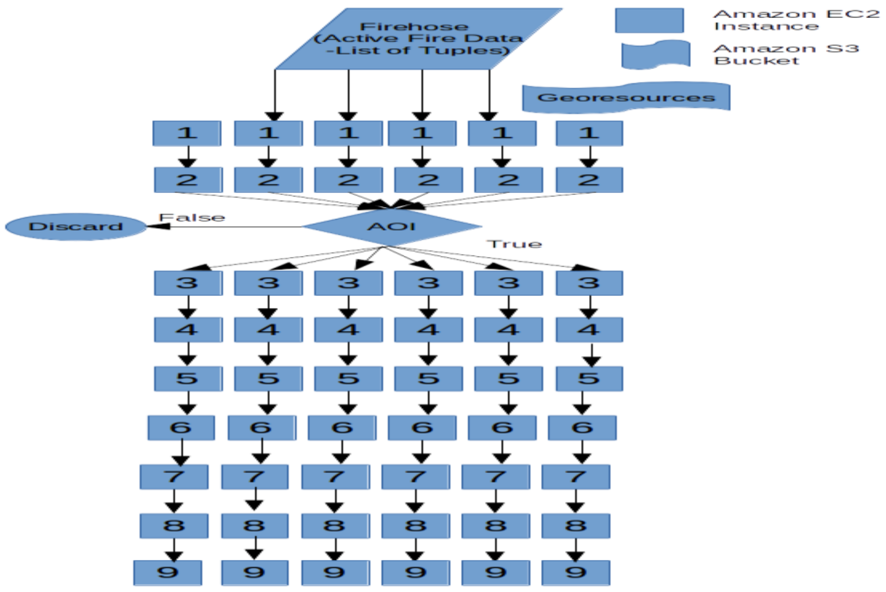


Figure 8: Cloud configuration-six consumers per geoprocessing chain component.

Refer to Table 3 (page 40) to check the corresponding component number to see the function of a component.

4.5 Chapter Summary

This chapter included implementation details such as the data, hardware and software required for the experiments. The methods used in this research were also discussed in this chapter. Various implementations were benchmarked to inform the most optimal design of the prototype. Pre-tests were conducted to determine the best algorithmic style for components of the geoprocessing chain. Four "flavours" of the geoprocessing chain were benchmarked. The first "flavour" is a tightly-coupled Web Processing Service implementation. The second "flavour" is a tightly-coupled function call implementation. The third "flavour" is a loosely-coupled Web Processing Service implementation and the fourth "flavour" is a loosely-coupled function call implementation. The proceeding chapter will provide the results of the experiments. It will also contain an in-depth discussion of the results.

5 Chapter Five: Results and Discussion

5.1 Chapter Overview

The previous chapter provided an overview of the implemented experimental system by discussing implementation detail. It provided details on the data, software and hardware used for the prototype. It also provided information on the various "flavours" of the geoprocessing chain that were experimented with.

This chapter provides illustrations of the results delivered in the pre-tests, experiment one and experiment two. Illustrations are included comparing the results of experiment one and the results of experiment two. A detailed discussion of the results is provided in this chapter.

All of the results in this chapter represent the tests that were created to time process execution from start to finish. Shorter process execution times are preferred. More detailed graphs are included in Appendix B.

5.2 Pre-Tests Results

5.2.1 Graphs

Referring to *Figure 9*, the OGR implementation outperformed the other implementations. The Rtree implementation delivered the second best processing times and the Fiona+Shapely implementation produced the third best. The PostGIS implementation underperformed.

Referring to Figure 9, using the OGR library for the area of interest test proved to be the best and fastest option. The Rtree and Fiona plus Shapely libraries used for this test, proved to be the second and third best respectively. The PostGIS implementation exhibited the poorest performance by a significant margin.

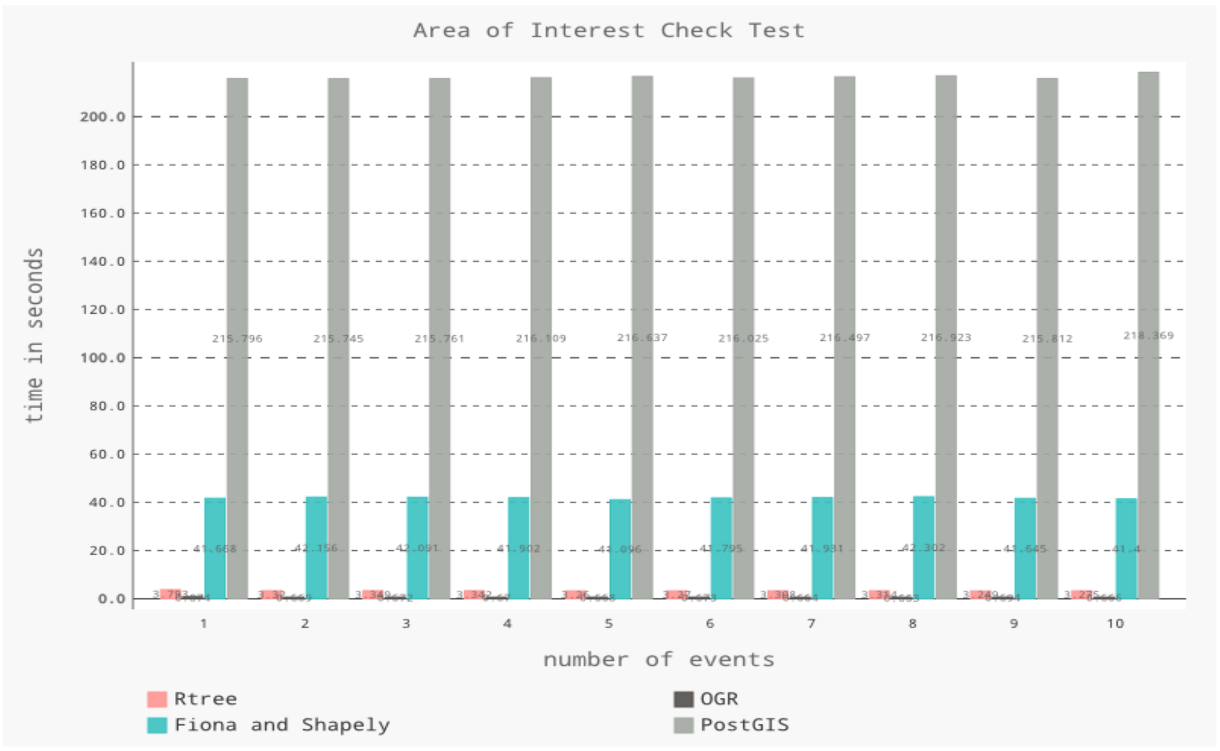


Figure 9: Area of Interest Check-test results (Section 4.4.3, page 48)

Referring to *Figure 10*, the OGR implementation performed the best in the buffer and bounding box test. The Fiona plus Shapely implementation performed slower than the OGR implementation. The PostGIS implementation underperformed.

Referring to *Figure 11*, the GDAL implementation strongly outperformed the PostGIS implementation in the elevation calculation test. Other components of the geoprocessing chain (getPopulation and getLandCover) utilised a similar approach to extract values from a geoTIFF. The results for those components are likely represented by the elevation extraction test.

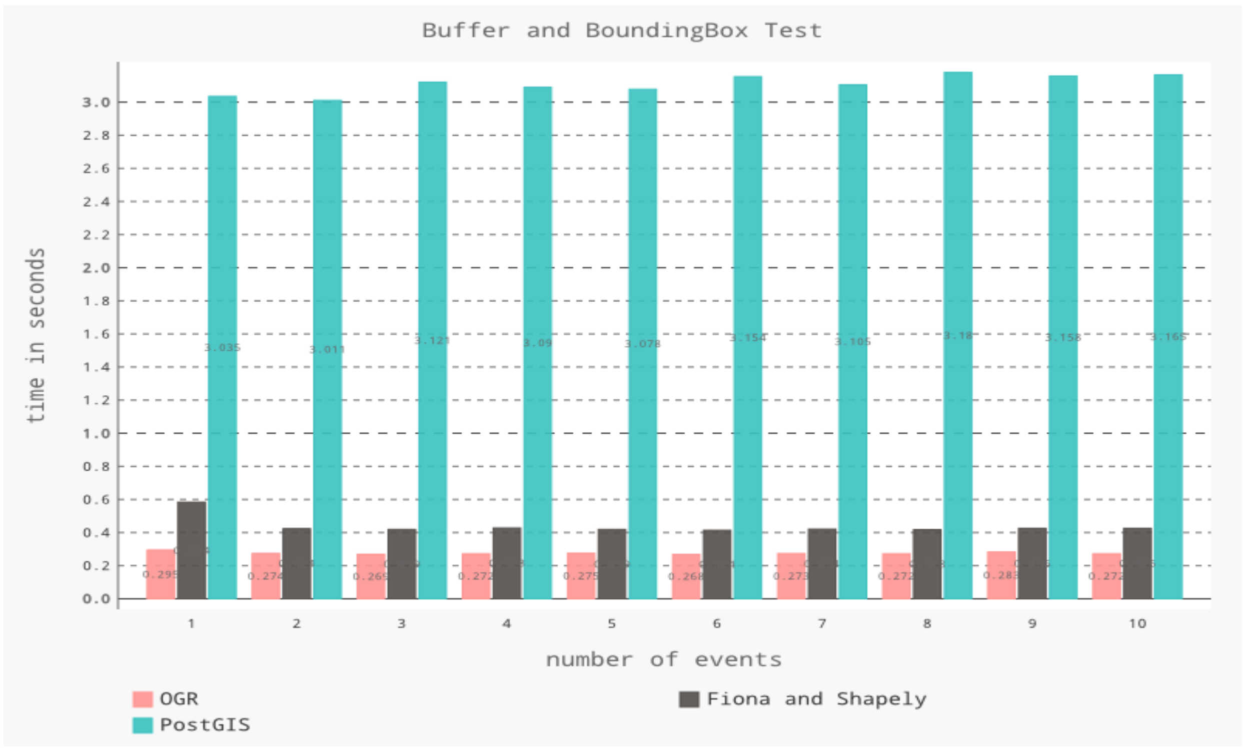


Figure 10: *Buffer and Bounding Box-test results (Section 4.4.3, page 48)*

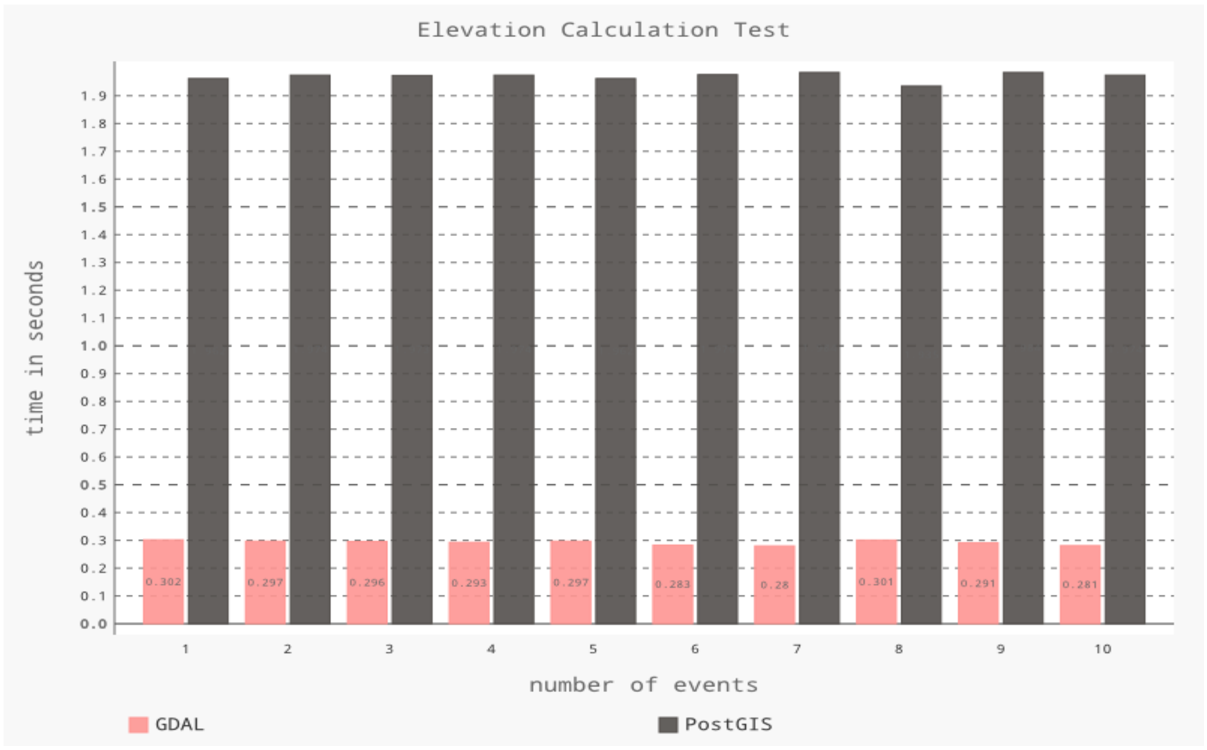


Figure 11: *Elevation Calculation-test results (Section 4.4.3, page 48)*

5.2.2 Discussion

Results from the first three sets of tests were surprising. It was expected that Rtree would perform best in test one(the area of interest check test), due to its specialised spatial indexing functionality. Fiona plus Shapely and PostGIS were expected to rival or surpass OGR in the tests, due to the underpinning specialised geometry engine (GEOS-Geometry Engine Open Source) software (OSGeo, 2014), but Rtree performed worse than OGR in the area of interest check process. Several optimisations to improve the performance of Rtree indexing are possible, such as streamed data loading, which improves performance via pre-sorting before indexing (Python Community, 2012). If the size of the index is known a-priori, Rtree will gain performance by pre-sizing the index to save on storage.

The OGR library allows performant spatial indexing of shapefiles through support for the .sbn and .sbx indexes (GDAL, n.d.). The floating point comparisons in Rtree are a likely explanation for the slower performance of Rtree compared to OGR (Python Community, 2012) which uses integer comparisons (Rouault, 2012). The difference in behaviour of Rtree and OGR was distinctly demonstrated in the spatial filtering step(area of interest check process) and the above-mentioned might be the reason for the difference in performance.

PostGIS supports indexing, although recent versions use Rtree-over-GIST (Generalised Search Tree) schemes for indexing (OSGeo, 2013). This advanced indexing feature was expected to deliver good performance, but the connection setup and query planner overhead are suspected to be the reasons for slowing the process substantially (the two processes or operations takes time to execute). Raster tile sizes in the PostgreSQL database can similarly influence the performance of the processes. The atomic nature of the data was an obstacle. A connection to the spatial database had to be established every time a new fire event was processed. This caused an increase in the geoprocessing time (caused an overhead) as it is time consuming to make a connection to the database. It is preferred with relation databases, to work with bulks of data and not atomic pieces of data or messages that require repeated connections.

The result of the Fiona plus Shapely implementation can be explained effortlessly. Fiona utilises Python objects which impact negatively on the performance, compared to OGR which uses C pointers (Python Community, 2013)which are less memory intensive. Fiona's documentation states that it “trades memory for simplicity and reliability”. Fiona copies features from the data source to Python objects. The performance of the Fiona plus Shapely implementations improved after using prepared geometries.

5.3 Experiment One: Loosely-Coupled and Tightly-Coupled Architectural Style and Web Processing Service Tests Conducted on a Single Thread on Commodity Hardware

5.3.1 Graphs

For all of the tests conducted in experiment one, illustrated times represent the interval between fire event ingest and 3D scene rendering output. It therefore represents the time it takes for the specified number of events to move through the entire geoprocessing chain (geoprocessing time). An increase or improvement in performance is equivalent to a decrease in the geoprocessing time.

The difference in the resulting execution or processing times of the tightly-coupled function call implementation and the loosely-coupled function call implementation (*Figure 12),* is almost negligible. The loosely-coupled process chaining implementations performed significantly faster than the tightly-coupled process chaining implementations. The function call implementations performed far better than the Web Processing Service implementations. The performance of the four implementations was consistent for various numbers of events as the pattern of the graph remained consistent.

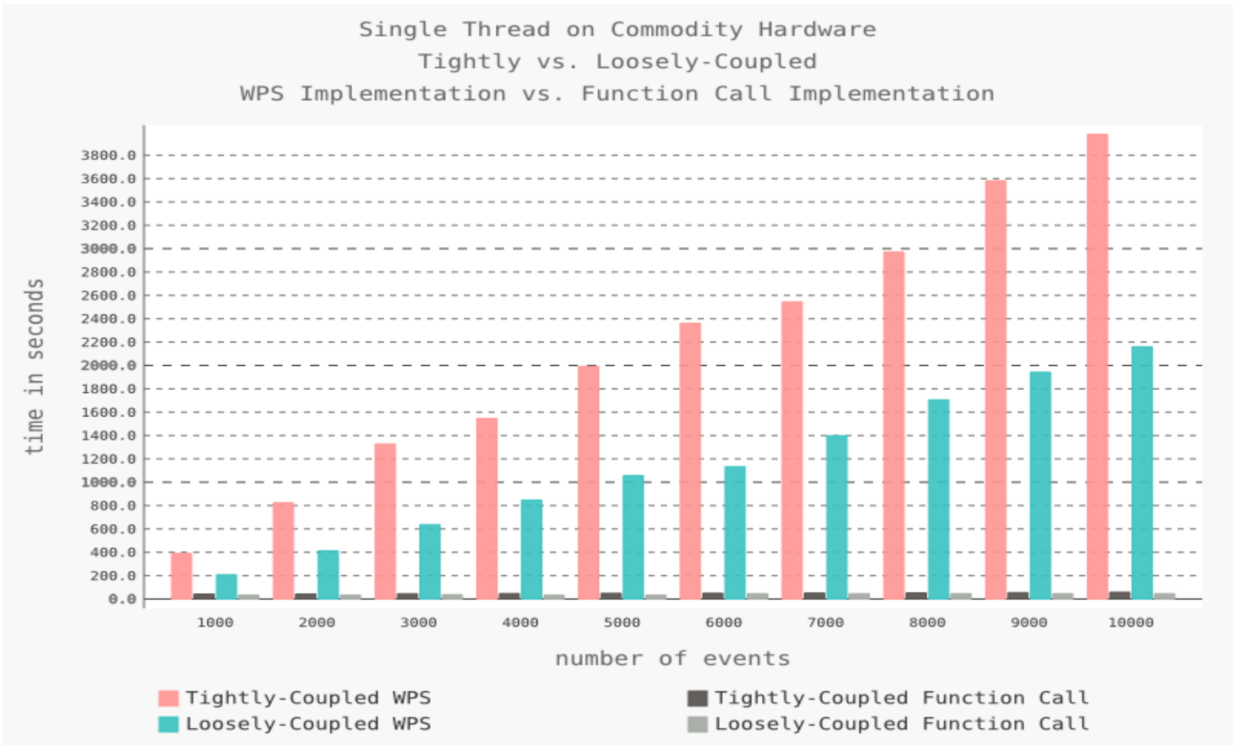


Figure 12: Single thread on commodity hardware tightly-coupled versus loosely-coupled and WPS implementation versus function call implementation-permutation test results. One consumer (Sections 4.4.4 (page 50), 4.4.5 (page 53))

No general pattern can be viewed from the graph in *Figure 13*. It was expected that the processing times would decrease as the volume of consumers increased. The expectation was met in the case of the Web Processing Service implementation where 2000, 3000, 4000 and 10000 events had to be processed. For the function call implementation, the expectation was met where 6000, 7000, 8000 and 9000 events had to be processed.

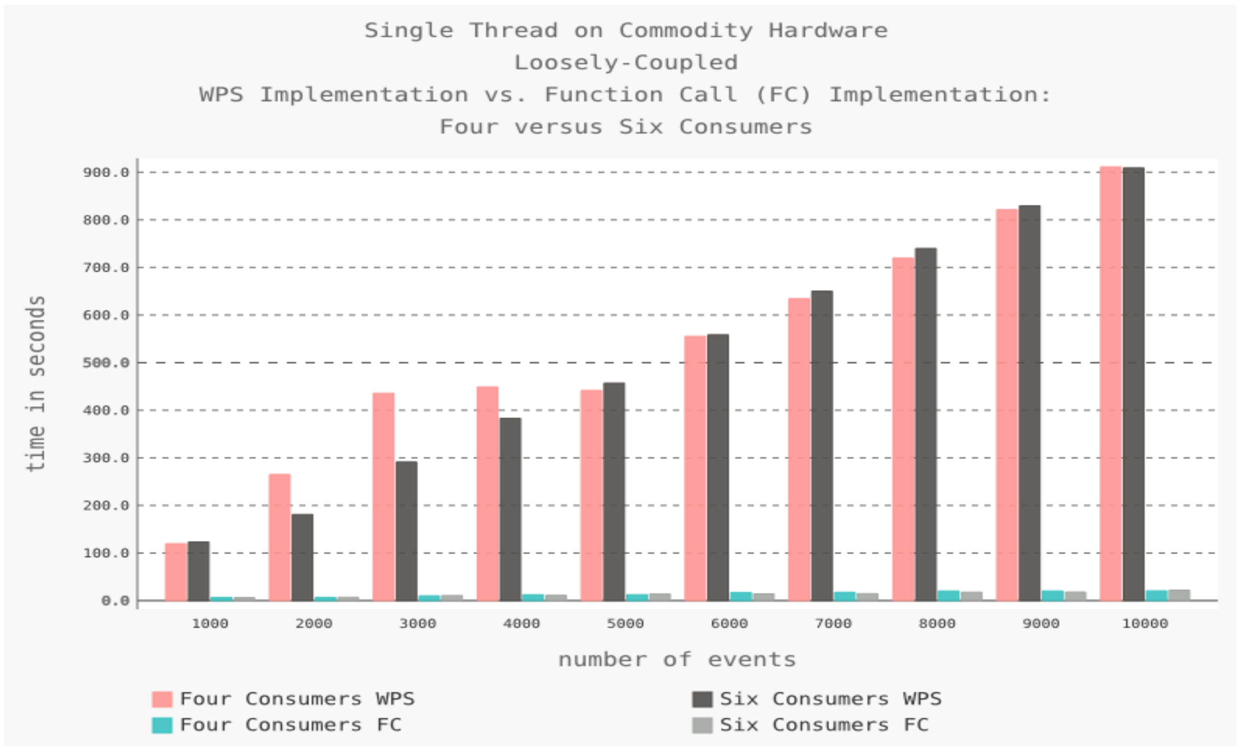


Figure 13: Single thread on commodity hardware tightly-coupled versus loosely-coupled and WPS implementation versus function call implementation-permutation test results. Four versus six consumers(Sections 4.4.4 (page 50), 4.4.5 (page 53))

5.3.2 Discussion

By comparing the two chaining approaches, the loosely-coupled component chaining approach provided a better performance than the tightly-coupled component chaining approach. The process execution time halved with the loosely-coupled component chaining approach that utilises the enterprise messaging solution, compared to the tightly-coupled component chaining approach where the process execution time doubled. This is because with the tightly-coupled component chaining approach, one event has to be processed from start to finish before the next one can commence. This can be referred to as serial geoprocessing. With the loosely-coupled component chaining approach, one event does not have to be fully processed for the next one to start because of multiple consumers and producers in the system. This can be referred to as parallel geoprocessing. The components of the system are unaware of each other.

A big difference in performance is noted from the process chaining with a Web Processing Service and without a Web Processing Service. The processing time from 1000 to 10000 events without the Web Processing Service for tight and loosely-coupled component chaining was almost constant. The processing time with the Web Processing Service from 1000 to 10000 events increased linearly. The PyWPS implementation itself might be slow, but performance likely suffers due to the time it takes to construct and destroy Web Processing Service instances in a web server environment such as Apache. Due to the atomic nature of the data, repeated "calls" to Web Processing Service had to be made. Differences between passing geoprocessing payloads by POST payloads or GET URL key/value pairs do not appear to be a significant contributor to performance results. These results require further investigation, beyond the scope of this master's research that requires demonstrating that one can master a skill or skills.

5.4 Experiment Two: Loosely-Coupled Architectural Style and Web Processing Service Tests in Cloud Environment

5.4.1 Graphs

For the tests conducted in experiment two, illustrated times represent the interval between fire event ingest and 3D scene rendering output. It represents the time it takes for the specified number of events to move through the entire geoprocessing chain (geoprocessing time). An increase or improvement in performance is equivalent to a decrease in the geoprocessing time.

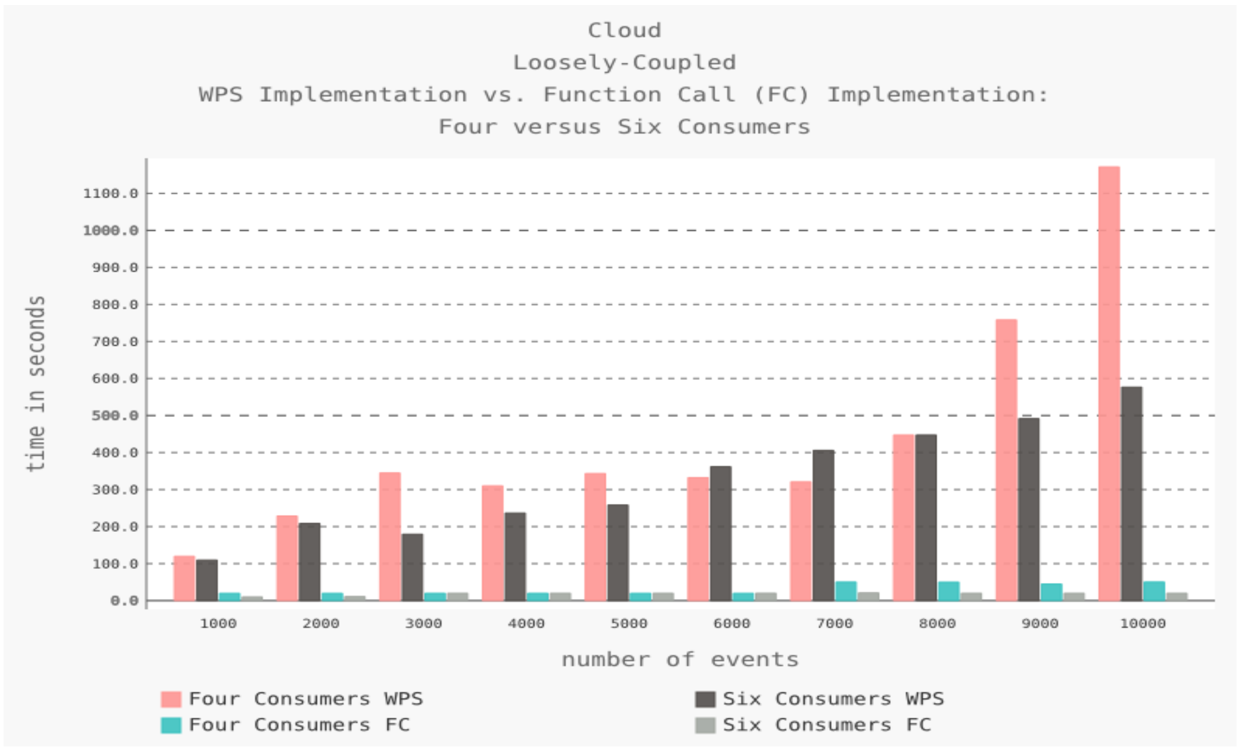


Figure 14: Cloud loosely-coupled WPS implementation versus function call implementation test results. Four consumers versus six consumers (Sections 4.4.4 (page 50), 4.4.6 (page 55))

As seen in *Figure 14*, an increase in consumers (four to six) leads to a decrease in the duration of processing fire events. More nodes were added to process events; therefore processing nodes had less responsibilities. There were two cases with outliers, when 6000 and 7000 events were processed. Function call process chaining implementations produced output in a shorter time than Web Processing Service implementations.

5.4.2 Discussion

Due to the limitation of solely using micro-instances within the limits of the free tier, tests had to be conducted within the 750 hour monthly limit. As stated, two instances running concurrently for one hour is billed for two hours.

Initially one consumer was utilised for every component in the geoprocessing chain for the tests conducted in the cloud. By conducting the test that did not utilise Web Processing Services to facilitate geoprocessing in the cloud (for 1000 events), it was established that geoprocessing with one consumer took over double the time it took on a desktop.

In order to not waste any computing hours, there was no need to continue experimenting with one consumer per component of the geoprocessing chain.

The experiment resumed with two and three consumers per component and because it did not provide sufficient results, there was no need to repeat the experiment with more fire events. When the time arrived to repeat the experiment with four consumers per component, there was a remarkable improvement in the performance of the geoprocessing chain.

The approach that utilises Web Processing Services in the cloud became faster than the one on the desktop. It was expected that the performance of the Web Processing Services would be slow due to a statement made by Giuliani *et al*. (2012). The authors noted that high-performance computing is essential in a situation where Web Processing Services are successfully utilised.

The experiment concluded with six consumers per component due to only having access to the free tier of three accounts and being limited to only running 20 cloud micro-instances at a time.

It was previously stated that the geoprocessing chain consists of a fire producer and nine components. Every account (there were three accounts) enabled that every component was assigned two consumers, thus 18 cloud instances were running on every account.

It was required for every account to have its own queue listener for timing the geoprocessing chain execution. A new public IP address is assigned to an Amazon EC2 instance every time it is stopped and started. Not only was it tedious to modify the IP addresses of the consumers to create a connection, but the process had to be modified 55 times for 10 connections. It is more expensive to access an EC2 instance from a public IP address than a private IP address due the difference in geographic region.

The solution was to create an Amazon EC2 instance for one listener per account. A listener listened on the messaging queues for each account. The connections were created by using private IP addresses.

The results of the tests conducted in the cloud by processing from 6000 and 7000 fire events(using four consumers per component), were troubling. It appeared as if Amazon enforced some kind of message throttling. Amazon does not have any documentation on message throttling with regards to AMQP but they do throttle requests to their Simple Queue Service (similar to AMQP) when necessary (Amazon Web Services, n.d.). Another explanation for this occurrence, maybe that more fires occurred within the area of interest and were required to move down along the geoprocessing chain. Two consumers were added per component, making the total six consumers per component and the curves appearing to be more smoothed out. This result verified the assumption that more fires occurred within the area of interest.

Amazon's CPU credit concept was unknown at that stage. After observing the appearance of the graphs and by conducting research into possible explanations for the strange appearance, the CPU credit concept became clear.

Refer to the second paragraph on page 56 for an explanation.

When a significant quantity of geoprocessing is required, cloud instances might utilise all their CPU credits. This leads to a degradation in the performance of the geoprocessing chain that will cause messaging queues to backlog and therefore not prepare the 3D wildfire context visualisations in time. This is the reason why auto-scaling is imperative. Cloud instances can be stopped and started according to demand and will lead to the receiving of a large amount of CPU credits every time an instance is started.

In this situation, an instance receives 30 credits at start-up equal to 30 minutes of 100% CPU utilisation. The tests did not take 30 minutes to execute, thus running out of CPU credits was not the cause of the degradation in performance.

Lê-Quôc *et al*. (2014) states that there can be several reasons why Amazon Web Services Elastic Compute Cloud has performance issues. Amazon only releases a small quantity of information regarding the hardware that the virtual machines (cloud instances) are running on. Amazon Web Services is an opaque system, thus a user has no idea as to the state of the hardware and how it is utilised (what is happening "under-the-hood"). Hardware is shared amongst several users. Hardware can be allocated virtual machines that exceed its limits even to a point where multiple users might compete for resources. The Amazon Web Services infrastructure is large, and because of this, instances might run on damaged components which can take a while to detect that the components are damaged. Because of Amazon Web Services (where several users share resources), performance cannot be guaranteed. The authors of this eBook presented a few situations that may have an impact on the EC2 performance. The main conclusion that can be drawn after reading this book, was that larger instances should avoid sharing resources.

Mukherjee *et al*. (2013) tested the performance of web applications running on different sizes of EC2 instances. They found that results of response times fluctuate for the same workload. They accounted for the EC2 instance type, time of the day and day of the week.

Wayner (2013) observed that the behaviour in performance (of an EC2 instance) was similar to the behaviour of the instances observed in experiment two. EC2 instances will have a relatively good performance at certain times and then for no specific reason, the performance will degrade at other times. These findings are concerning and Amazon Web Services should be prompted for an explanation, or further research should be conducted. Mukherjee *et al*. (2013) stated that more research is required on this topic.

Every time an instance is started, an hour's usage is added to the bill. This may possibly make the stop-start (auto-scaling) solution more expensive, but there must be some kind of trade-off (as the CPU credits are designed to make an instance burstable). It should be noted that resetting an instance will enable an instance to receive the initial amount of CPU credits and therefore an hour's usage will not be added to the bill.

Because of the slow performance of the initial cloud tests, two approaches to storing data had to be investigated. The first approach was to upload the resource files to an Amazon S3 bucket and the second approach was to store the resource files on the computing nodes (EC2 instances).

The first approach enabled the mounting of the Amazon S3 file system (bucket) to the cloud instance. The second approach entailed downloading the resource files from the S3 bucket once the instance was started for the first time. Downloading a file from the S3 bucket to an EC2 instance was faster than downloading a file from the S3 bucket to a machine in Pretoria, South Africa. The reason for this being that the S3 bucket and the EC2 instance were in the same region (Oregon, US). Data transfer within the same region is free but data transfer between different regions is not free. Accessing data when using the first approach was slower than accessing data when using the second approach. This was due to the connection that had to be kept alive between the EC2 instance and the S3 bucket in the first approach. There was no connection that had to be kept alive for the second approach. The only thing that was time-consuming, but only once, was the downloading of the data when the instance was started for the first time.

Similar to the results of the tests conducted in experiment one, in a desktop (single thread on commodity hardware) environment, process chaining without a web processing service performed better than process chaining with a web processing service. The performance of the PyWPS implementation itself might be slow, but it does take extra time to invoke and destroy Web Processing Server instances in a web server environment.

5.5 Comparison between Tests of Experiment One and Experiment Two

5.5.1 Graphs

In *Figure 15*, it can be viewed that an increase in the number of consumers for the cloud-based implementation will lead to a decrease in the processing times of fire events. There are outliers such as with the cloud-based Web Processing Services implementation where 6000 and 7000 fire events were required to be processed. The situation is not similar for the commodity hardware-based implementations. The performance of the implementations were inconsistent as there is no general pattern that can be viewed. This might be due to the single thread used.

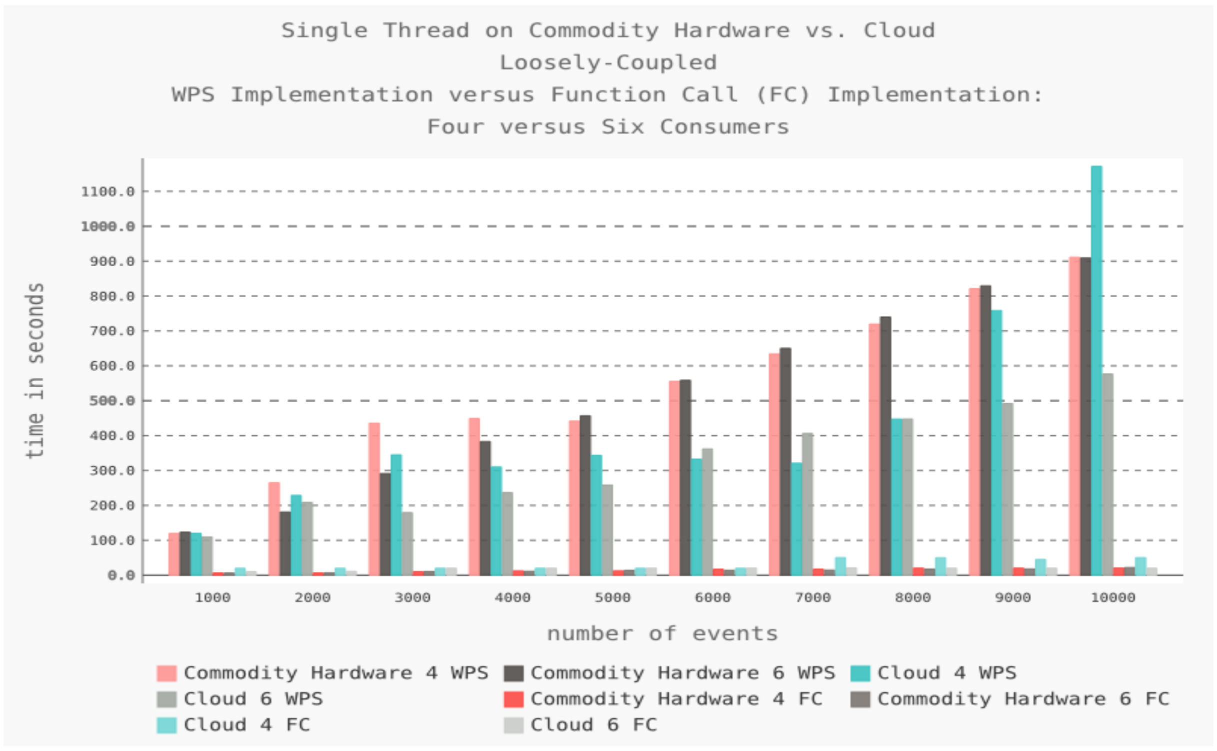


Figure 15: Single thread on commodity hardware versus cloud, loosely-coupled, Web Processing Service versus function call permutation tests results. Four consumers versus six consumers (Sections 4.4.4 (page 50), 4.4.5 (page 53), 4.4.6 (page 55))

5.5.2 Discussion

Several observations have been noted by comparing the desktop-based (single thread on commodity hardware) and cloud-based tests. By not utilising Web Processing Services and utilising four consumers (per component of the geoprocessing chain) for the desktop-based (single thread on commodity hardware) and cloud-based tests, the desktop-based (single thread on commodity hardware) approach to do geoprocessing performed better than the cloud-based approach to do geoprocessing. The result was similar when utilising six consumers (per component of the geoprocessing chain).

By utilising Web Processing Services to conduct geoprocessing with the number of consumers (per geoprocessing chain component) being any number from one to three, the cloud-based approach performed worse than the desktop-based (single thread on commodity hardware) approach.

As discussed, Web Processing Service invocation and destruction overhead maybe responsible for the slow performance of Web Processing Services-based geoprocessing. It is unclear if the implementation of the Web Processing Service (PyWPS) is responsible for slowing down the performance as well. The hypothesis was that adding more consumers may smooth out the graph as the geoprocessing tasks are divided amongst more nodes. This is equal to a divide and conquer approach.

By utilising Web Processing Services to do geoprocessing and utilising four consumers (per component of the geoprocessing chain) for the desktop-based (single thread on commodity hardware) and cloud-based test, the performance of the cloud-based approach improved. It performed faster than the desktop-based (single thread on commodity hardware) approach. The situation was similar when utilising six consumers (per component of the geoprocessing chain), but the performance was faster with the utilisation of six consumers (per component of the geoprocessing chain) compared to the utilisation of four consumers (per component of the geoprocessing chain).

There is a simple explanation for this occurrence. It was previously stated that the desktop machine is a laptop with an Intel Core i7, quad core CPU (Central Processing Unit) with four multi-threaded cores, therefore, eight cores running at 2.10GHz, but can turbo boost (dynamic over-clocking) up to 3.10GHz and it has 8GB of RAM (Random Access Memory). A consumer is an EC2 cloud micro-instance with one CPU and 0.6GB RAM. The physical processor of a micro-instance belongs to the Intel Xeon family and has a clock speed of 2.5 GHz but can boost (over-clock) up to 3.3GHz. For the approaches that utilised Web Processing Services, combining several small machines might have resulted in an improvement in performance because of a single core being solely dedicated to one consumer and the fact that a consumer has its own dedicated RAM.

5.6 Chapter Summary

This chapter included illustrations of the results delivered in the pre-tests, experiment one and experiment two. Illustrations were correspondingly included that compared the results of experiment one and the results of experiment two. The illustrations were discussed in-depth. Several observations were made. It is faster to load an object into memory than to access it from a database if it is desired to obtain values for a specific location. A loosely-coupled process chaining implementation produces results faster than a tightly-coupled process chaining implementation. Web Processing Service implementations underperformed compared to implementations that use function calls. By adding more consumers in a loosely-coupled configuration the processing times will speed up (decrease).The following chapter will conclude the dissertation and make recommendations for future research.

6 Chapter Six: Conclusions and Future Research

6.1 Chapter Overview

The preceding chapter presented the results and an in-depth discussion of the results of the pre-tests, experiment one and experiment two. A comparison between the results of these experiments was included. The observation made is that it is faster to load an object into memory than to access it from a database if it is desired to obtain values for a specific location. A loosely-coupled process chaining implementation produces results faster than a tightly-coupled process chaining implementation. Web Processing Service implementations underperformed compared to implementations that use function calls. Adding more consumers in a loosely-coupled configuration will speed up (decrease) processing times.

Included in this chapter are the conclusions drawn from this research and the recommendations for further study.

6.2 Conclusions

The research question related to the optimal design for a geo-computational cloud environment that can rapidly process medium velocity streaming geospatial data given cost and elasticity constraints, data delivery requirements and interoperability tradeoffs to generate 3D wildfire context visualisations.

The optimal design for a geo-computational cloud environment that can rapidly process medium velocity streaming geospatial data to generate 3D wildfire context visualisations was determined in several steps. Pre-tests were conducted to determine which "algorithmic style" (implementations that used different libraries) offered output in the shortest amount of time for specific components of the geoprocessing chain.

The first aim of the research was to conduct a literature review on:

- Cloud computing
- Algorithmic styles

- Architectural styles
- Enterprise messaging solutions
- Web Processing Services
- 3D visualisations

The literature review is included in Chapter Two. There was no literature that compared algorithmic styles and architectural styles to each other. Pre-tests were conducted thereafter to benchmark the algorithmic and architectural styles. The pre-tests are described in Section 4.4.3 on page 48. The results of the pre-tests indicate that an implementation that utilises a library will produce output faster than an implementation that utilises databases. The OGR library proved to be the best option as it provided output in the shortest quantity of time. From the results of the pre-tests, it can additionally be noted that it is faster to retrieve values from rasters by loading objects into memory than by accessing objects from a database. The libraries generating output in the least quantity of time were chosen for the geoprocessing chain component implementations used in experiment one and in experiment two. The results of the pre-tests are described in Section 5.2 on page 61.

The second aim of the research was to design and execute desktop-based (single thread on commodity hardware) experiments to:

- Determine the optimal algorithmic style for a system required to process medium velocity streaming geospatial data at a rapid rate, and deliver 3D wildfire context visualisations in demand-time
- Determine the architectural style for a system required to process medium velocity streaming geospatial data at a rapid rate and deliver 3D wildfire context visualisations in demand-time
- Determine if Web Processing Services is suitable for a system required to process medium velocity streaming geospatial data at a rapid rate and deliver 3D wildfire context visualisations in demand-time. The desktop-based experiments are described in Section 4.4.5 on page 53.

Results of tests conducted in experiment one indicates that geoprocessing techniques that invoke Web Processing Services had a poor performance, compared to techniques that do not invoke Web Processing Services. Results of tests conducted in experiment two also confirmed the fact that Web Processing Services are responsible for a decrease in geoprocessing performance. The processing time will correspondingly decrease (improve) only if the quantity of machines responsible for geoprocessing increases. This will divide the workload amongst the quantity of machines. This situation is similar for geoprocessing techniques that do not utilise Web Processing Services. The results of the desktop-based experiments are included in Section 5.3 on page 65.

The third aim of the research was to design and execute the cloud-based experiment to evaluate if Web Processing Services are suitable for a system required to process medium velocity streaming geospatial data at a rapid rate and deliver 3D wildfire context visualisations in demand-time. This was done to determine if the horizontal scaling of cloud instances would speed-up the geoprocessing performance. The cloud-based experiments are described in Section 4.4.6 on page 55.

By deploying the experimental processing system in a cloud computing environment, storing resource files directly on computing nodes and accessing them from a node itself is more efficient than mounting a S3 file system and retrieving files from a mounted file system.

Web Processing Services deployed within a cloud computing environment are not able to generate 3D wildfire context visualisations in just enough time for them to be attached to wildfire notifications. The performance of public cloud computing instances (in this case Amazon Web Service EC2 instances) is unreliable due to unknown reasons. Amazon Web Services are unreliable because it can be viewed as a black box. This is due to a lack of supporting documentation and a lack of information on virtual machine allocation and throttling. The results of the cloud-based experiments are included in Section 5.4 on page 68.

The fourth aim of the research was to evaluate and compare the results of the desktop-based (single-thread on commodity hardware) and cloud-based experiments. The graphs comparing the results of the desktop-based and cloud-based experiments are included in Section 5.5 on page 73.

When more consumers were added, the cloud-based experiments seemed to generate 3D wildfire context visualisation in less time than the desktop-based experiments.

Based on the results of the research and by using all the available resources, Web Processing Services is not recommended for geoprocessing within a cloud environment when dealing with atomic or bursty data where demand-time results are required. Demand-time results can be achieved without the use of Web Processing Services. The demand-time production of 3D wildfire context visualisations cannot be achieved by using Web Processing Services. The overhead might be caused by a function call to a Web Processing Service. Many repeated "calls" to Web Processing Services will slow down the performance of a geoprocessing chain. If less pieces of data or messages were required to be processed, but the pieces required more processing, the geoprocessing performance in the cloud might have improved. A solution to improve the geoprocessing performance might have been to utilise less Web Processing Service components.

The optimal design for a geo-computational cloud environment that can rapidly process medium velocity geospatial data to generate 3D wildfire context visualisations is therefore a design utilising OGR/GDAL as the library to handle geospatial data with. The design should moreover not make use of databases to access data. Files should rather be stored on a machine/node's local file system. The design should not make use of Web Processing Services. The design should allow that the highest possible number of consumers be assigned to every component of the geoprocessing chain. Specifically referring to this research and given the cost constraint, six consumers per component of the geoprocessing chain is required.

6.3 Recommendations for Future Research

Several questions were raised during the research that were not necessarily answered. It should be investigated why querying rasters by using a database query language approach performed slower than using an approach that utilised a library to load rasters into memory. Indexing may very well be the reason for this result, but it should be investigated.

The reason why Web Processing Services drastically slowed down the execution of the geoprocessing chain should be investigated. It might be because of the Web Processing Service invocation and destruction. Another factor to rule out is that the cause of the poor performance might be the Web Processing Service implementation itself. Several Web Processing Service implementations should be compared to each other.

The reason behind the unreliable performance of Amazon EC2 instances should be thoroughly investigated as it might be too unreliable for demand-time or real-time applications.

7 References

AGI 2013, Cesium - WebGL Virtual Globe and Map Engine, viewed 7 July 2013, http://cesiumjs.org

Aiyagari, S, Arrott, M, Atwell, M, Brome, J, Conway, A, Godfrey, R, Greig, R, Hintjens, P, O'hara, J, Radestock, M, Richardson, A, Ritchie, M, Sadjadi, S, Schloming, R, Shaw, S, Sustrik, M, Trieloff, C, van der Riet, K & Vinoski, S 2008. *Advanced Message Queuing Protocol Specification,* Version 0.9.1, OASIS

Akioka, S,& Muraoka, Y 2010, HPC benchmarks on Amazon EC2, *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference,* 20-23 April 2010, Perth, Australia, pp. 1029-1034, doi: 10.1109/WAINA.2010.166

Amazon Web Services n.d., Amazon Simple Queue Service, viewed 17 February 2015, http://docs.aws.amazon.com/AWSSimpleQueueService/2008-01-01/SQSDeveloperGuide/index.html?Query_QueryErrors.html

Amazon Web Services n.d., *Amazon Web Services-Cloud Computing Services,* viewed 1 May 2014,http://aws.amazon.com

Armbrust, M, Fox, A, Griffith, R, Joseph, AD, Katz, R, Konwinski, A & Zaharia, M 2010. A view of cloud computing, *Communications of the ACM*, vol. *53*(4), pp. 50-58

Baranski, B, Schaffer, B, Lange, K & Foerster, T 2010, *Geoprocessing in hybrid clouds*, Geoinformatik, 2010, Kiel, Germany, pp. 13-19

British Columbia Wildfire Management Branch 2014, *Wildfire Management Branch*, viewed 21 July 2014, http://bcwildfire.ca

Brown, RB, Smoot, JC, Underwood, L & Armstrong, CD 2012, Investigation into Cloud Computing for more robust automated bulk image geoprocessing, *MAPPS/ASPRS Speciality Concurrence,* 29 October – 1 November 2012, Tampa, Florida, United States

Castrillón, M, Jorge, PA, López, IJ, Macías, A, Martín, D, Nebot, RJ & Trujillo, A 2011. Forecasting and visualization of wildfires in a 3D geographical information system, *Computers & Geosciences*, vol. *37* (3), pp. 390-396

Cepicky, J 2013, *Welcome to PyWPS \&amp;mdash; PyWPS,* viewed 21 November 2013, http://pywps.wald.intevation.org

CSIR 2012, *Advanced Fire Information System,* viewed 21 July 2014, www.afis.co.za

Dasgupta, A & Ghosh, SK 2011, Service chaining for accessing geospatial information in mobile devices.*Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications,* ACM, doi: 10.1145/1999320.1999343

Davis, DK, Vosloo, HF, Frost, PE & Vannan, SS 2008, Near real-time fire alert system in South Africa: From desktop to mobile service, *Proceedings of the 7th ACM Conference on Designing Interactive Systems,* 25 – 27 February 2008, Cape Town, South Africa, pp. 315-322

Department of Water Affairs and Forestry n.d., Veldfires in South Africa, viewed 3 November 2014, www.daff.gov.za/doaDev/sideMenu/ForestryWeb/webapp/Documents/ForestFire/192.168.10.11/nvffa.nsf/cba79e2e60cb841f2256d6eoo3942fa/64a7c7f6bea728c942256dff003121a502.ec.html?OpenDocument

Dillon, T, Wu, C, Chang, E Chang 2010, Cloud Computing: Issues and Challenges, *24th IEEE International Conference on Advanced Information Networking and Application Cloud Computing*, 20-23 April 2010, Perth, Australia, pp.27-33

Dumbill, E 2012, *What is big data: An introduction to the big data landscape,* viewed 1 March 2013*,* http://radar.oreilly.com/2012/01/what-is-big-data.html#variety

Eugster, PTH, Felber, PA, Guerraoui, A & Kermarrec, AM 2003, The many faces of publish/subscribe, *ACM computing surveys,* vol.35, pp. 114-131

Evangelidis, K, Ntouros, K, Makridis, S & Papatheodorou, C 2014, Geospatial services in the Cloud, *Computers & Geosciences*, vol. *63*, pp. 116-122

Fernandes, JL, Lopes, IC, Rodrigues, JJ, & Ullah, S 2013, Performance evaluation of Restful Web services and AMQP protocol. The Fifth International Conference on *Ubiquitous and Future Networks,* 2-5 July 2013, IEEE*,* pp. 810-815, doi:10.1109/ICUFN.2013.6614932.

GDAL n.d., *OGR: Simple Feature Library,* viewed 21 November 2013, http://www.gdal.org/ogr/

Geoprocessing.info n.d., *Geoprocessing*.viewed 21 November 2013, http://geoprocessing.info/wpsdoc/

Giuliani, G, Nativi, S, Lehmann, A, & Ray, N 2012, WPS mediation: An approach to process geospatial data on different computing backends, *Computers & Geosciences*, vol. *47*, pp. 20-33

Goldberg, D, Olivares, M, Li, Z & Klein, AG 2014, Maps and GIS Data Libraries in the Era of Big Data and Cloud Computing, *Journal of Map and Geography Libraries,* vol.10, pp. 100-120

Gong, J, Zhou, H & Yue, P 2010, Geoprocessing in the Microsoft Cloud Computing Platform-Azure, *International Journal of Digital Earth, vol.*6, pp. 404-425

Hankel, L 2013, Exploiting Cloud Computing and Web Processing Services for the Processing and Visualisation of Big Geospatial Data, *First Postgraduate Research Seminar by the Department of Geography*, Geoinformatics and Meteorology, University of Pretoria, 10-11 October 2013, Pretoria, South Africa

Hankel, L, McFerren, G, Coetzee, S 2014, Distributed Geoprocessing of Streaming Data for a 3D Context Aware Visualisation Solution of a Wildfire Scenario, *The 11th International Symposium on Location Based Services*, 26-28 November 2014, Vienna, Austria

Hildebrandt, D & Döllner, J 2010, Service-oriented, standards-based 3D geovisualization: Potential and challenges, *Computers, Environment and Urban Systems,* vol. 34(6), pp. 484-495

Huang, Q, Yang, C, Liu, K, Xia, J, Xu, C, Li, J, & Li, Z 2013, Evaluating open-source cloud computing solutions for geosciences, *Computers & Geosciences*, vol. *59*, pp. 41-52

Hyong-Woo, K, Dae-Sun, K, Yang-Won, L & Jae-Seong, A 2014, 3-D Geovisualization of satellite images on smart devices by the integration of spatial DBMS, RESTful API and WebGL, *Geocarto International*, doi: 10.1080/10106049.2014.888485

Jadeja, Y & Modi, K 2012, Cloud computing-concepts, architecture and challenges. *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET), Kumaracoil,* pp. 877-880, doi:10.1109/ICCEET.2012.6203873

Jensen, JR 2004, *Introductory Digital Image Processing*, 3rd edition. Prentice Hall

Johnsen, FT, Bloebaum, TH, Avlesen, M, Spjelkavik, S,& Vik, B 2013, Evaluation of transport protocols for web services. *Military Communications and Information Systems Conference (MCC),* St. Malo, pp. 1-6

Juve, G, Deelman, E, Berriman, GB, Berman, BP, & Maechling, P 2012, An evaluation of the cost and performance of scientific workflows on Amazon EC2, *Journal of Grid Computing*, vol. *10* (1), pp. 5-21

Karimi, HA 2014*, Big Data: Techniques and Technologies in Geoinformatics*. CRC Press.

Kassab, A, Liang, S & Gao, Y 2010, Real-time notification and improved situational awareness in fire emergencies using geospatial-based publish/subscribe, *International Journal of Applied Earth Observation and Geoinformation, vol. 12(6), pp. 431-438*

Kiehle, C, Greve, K & Heier, C 2007, Requirements for next generation spatial data infrastructures -standardized web based geoprocessing and web service orchestration, *Transactions in GIS,* vol.11, pp. 819-834

Kim, HW, Kim ,DS, Lee ,YW & Ahn,JS 2014, 3-D Geovisualization of satellite images on smart devices by the integration of spatial DBMS, RESTful API and WebGL, *Geocarto International,*(ahead-of-print), pp.1-19

Kmoch, A & Klug, H 2014, Visualization of 3D Hydrogeological Data in the Web, GI Forum 2014 Proceedings, Symposium and Exhibit Geospatial Innovation for Society, 1-4 July 2014, Salzburg, Austria, viewed 24 August 2013, http://hw.oeaw.ac.at/0xc1aa500d%200x0030d3d3.pdf

Kraak, MJ & Ormeling, F 2011, *Cartography: Visualization of Spatial Data,* 3rd edn. Pearson Education Limited, London

Kraak, MJ n.d., The Space-time cube revisited from a geovisualization perspective, *Proceedings of the 21st International Cartographic Conference*

Kumar, A & Bawa S 2012, Distributed Big Data Storage Management in Grid Computing, *International Journal of Grid Computing and Applications (IJGCA),* vol. 2, pp. 28-29

Kwan, MP, Lee, J 2003, Geovisualization of Human Activity Patterns Using 3D GIS: A Time-Geographic Approach, *Spatially Integrated Social Science: Examples and Best Practice*, Oxford University Press

Lê-Quôc, A, Fiedler, M, Cabanilla, C 2013, The Top 5 AWS EC2 Performance Problems, viewed 27 October 2014, http://www.infoworld.com/article/2613784/cloud-computing/benchmarking-amazon-ec2--the-wacky-world-of-cloud-performance.html

MacEachren, AM, Kraak, MJ 2001, Research Challenges in Geovisualisation, *Cartography and Geographic Information Sciences*, vol.28 (1), pp. 3-2, DOI:10.1559/152304001782173970.

McCullough, A, James, P, Barr, S 2011, A Typology of Real-Time Parallel Geoprocessing for the Sensor Web Era, *Proceedings of the Workshop on Integrating Sensor Web and Web-Based Geoprocesssing*, AGILE, 18 April 2011, Utrecht, CEUR, pp. 1-5

McCullough, AR 2011, Sensor Web Processing on the Grid, Phd Thesis, Newcastle University, England, Handle: http://hdl.handle.net/10443/1321

McFerren, G & Frost, P 2009, The South African Advanced Fire Information System, *Proceedings of the 6th International ISCRAM Conference,* 10 – 13 May 2009, Gothenburg, Sweden, viewed 25 July 2014, http://hdl.handle.net/10204/3631

McFerren, G, Swanepoel, D & Lai, C 2013, Wide Area Alerting System for Wildfires &Other Nasties, *E-learning for the Open Geospatial Community Conference Series,* FOSS4G 2013, 17-21 September 2013, Nottingham, United Kingdom

Mell, P & Grance, T 2011, The NIST Definition of Cloud Computing, Special Publication pp.800-146

Meng, X, Bian, F & Xie, Y 2009, Geospatial Services Chaining with Web Processing Service, *Proceedings of the International Symposium on Intelligent Information Systems and Applications,* 28-30 October 2009, Qingdao, China, pp. 7-10

Michaelis, CD & Ames, DP 2007, Evaluation of the OGC Web Processing Service for use in a client-side GIS, *OSGEO Journal,* vol. 1, pp. 1-8

MrDoob 2014, JavaScript 3D Library, viewed 1 June 2014, https://github.com/mrdoob/three.js

Mukherjee, J, Wang, M & Krishnamurthy D 2014, Performance Testing Web Applications on the Cloud, *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (ICTSTW)*, 31 March - 4 April 2014, Cleveland, Ohio, pp. 363-369, doi: 10.1109/ICSTW.2014.57

Murillo, CAO 2011, Parallelization of Web Processing Services on Cloud Computing: A case study of Geostatistical Methods, Masters Dissertation, Universidade Nova de Lisboa, Handle: http://hdl.handle.net/10362/8294

NASA Land Processes Distributed Active Archive Centre (LP DAAC), 2013, SRTM Version 3. USGS/Earth Resources Observation and Science (EROS) Centre, Sioux Falls, South Dakota

National Disaster Management n.d., Veld Fire Awareness, viewed 3 November 2014, www.iimp.co.za/Brochures/Veld_Fire_Awareness.pdf

Open Geospatial Consortium (OGC) 2007, OpenGIS® Web processing Service Version 1.0.0. OGC 05-007r7, viewed 20 August 2014, http://www.opengeospatial.org/standards/wps

OpenNebula 2013, *OpenNebula,* viewed 26 February 2013, http://opennebula.org/

OSGeo 2013, *GDAL/OGR Info Sheet,* viewed 21 November 2013, http://www.osgeo.org/gdal_ogr

OSGeo 2013, *PostGIS – Spatial and Geographic Objects for PostgreSQL,* viewed 21 November 2013, http://postgis.net/

OSGeo 2014, *GEOS - Geometry Engine, Open Source,* viewed 20 June 2013, http://trac.osgeo.org/geos/wiki.

Over, M, Schilling, A, Neubauer, S & Zipf, A 2010, Generating web-based 3D City Models from OpenStreetMap: The current situation in Germany, *Computers, Environment and Urban Systems,* vol. *34* (6), pp. 496-507

Oxford University Press 2015, Definition of time-series, viewed 4 March 2014, http://www.oxforddictionaries.com/definition/english/time-series

Percivall, G 2013, *Big Processing of Geospatial data,* viewed 9 August 2014, http://www.opengeospatial.org/blog/1866

Petcu, D, Neagul, M, Frincu, M, Zaharie, D & Panica, S 2010, Earth Observation Data Processing in Distributed Systems,*Informatica,* vol. 34, pp. 463-476

Pivotal Software, inc n.d., *RabbitMQ – Messaging that just works,* viewed 21 November 2013, http://www.rabbitmq.com/

Plaisant, C 2004, The Challenge of information visualization evaluation, *AVI'04 Proceedings of the working conference on advanced visual interfaces,* 25-28 May 2004, SIGCHI, Italy, pp. 109-116

Prandi, F, Soave, F, Devigli, M, Andreolli, R & De Amicis, R 2014, Services oriented smart city platform based on 3D city model visualization, *ISPRS Annals of the*

Photogrammetry, Remote Sensing and Spatial Information SciencesVol.2, ISPRS Technical Commission IV Symposium, 14 – 16 May 2014, Suzhou, China

Raffi 2013, New tweets per second record and how, viewed 3 March 2015, https://blog.twitter.com/2013/new-tweets-per-second-record-and-how

Resch, B, Wohlfahrt, R & Wosniok, C 2014, Web-based 4D visualization of marine geo-data using WebGL, *Cartography and Geographic Information Science,* vol. 41(3), pp. 235-247

Roualt, E 2012, *Geo tips \&tricks: GDAL/OGR using Shapefile native .sbn spatial index,* viewed 18 November 2013, http://erouault.blogspot.com/2012/06/gdalogr-using-shapefile-native-sbn.html

Russom, P 2011, Best Practices Report: 4th Quarter. *Big Data Analytics*

Samadzadegan, F, Saber, M, Zahmatkesha, H, & Khanlou, HJG 2013, An Architecture for Automated Fire Detection Early Warning System Based on Geoprocessing Service Composition, *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. *1*(3), pp. 351-355

Shao, Y, Di, L, Bai, Y, Guo, B & Gong, J 2012, Geoprocessing on the Amazon cloud computing platform—AWS. *Agro-Geoinformatics,The First International Conference on Agro-Geoinformatics*, 2-4 August 2012, Shanghai, pp. 1-6, doi: 10.1109/Agro-Geoinformatics.2012.6311655

Shekhar, S, Gunturi, G, Evans, MR, & Yang, KS 2012, Spatial Big-Data Challenges Intersecting Mobility and Cloud Computing. *Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*, 20 May 2012, Arizona, USA, pp. 1-6, doi: 10.1145/2258056.2258058

Skytland, N 2012, *Big data: What is NASA doing with big data today?,* viewed 1 March 2013, http://open.nasa.gov/blog/2012/10/04/what-is-nasa-doing-with-big-data-today

Slocum, TA, Blok, C, Jiang, N, Koussoulakon, A, Montello, DR, Fuhrmaann, Hedley, NR 2001, Cognitive and Usability Issues in Geovisualization, *Cartography and Geographic Information Science*, vol. 28 (1), pp. 61-75, DOI: 10.1559/152304001782173998

South African National Biodiversity Institute 2006, Vegetation Map 2006, viewed 20 August 2014, http://www.bgis.sanbi.org/vegmap/project.asp

South African National Biodiversity Institute 2009, National Land Cover 2009, viewed 20 August 2014, http://www.bgis.sanbi.org/land cover/project.asp

Southern Cape Fire Protection Association n.d., Fire Safety-Some facts about fire and how to deal with it, viewed 3 November 2014, http://www.scfpa.co.za/index.php?comp=content&id=7

Sun, Z,& Yue, P 2010, The use of Web 2.0 and geoprocessing services to support geoscientific workflows, *18th International Conference on Geoinformatics,* Beijing, pp. 1-5, doi: 10.1109/GEOINFORMATICS.2010.5567702

The Python Community 2012, *Rtree 0.7.0: Python Package Index*, viewed 21 November 2013, https://pypi.python.org/pypi/Rtree

The Python Community 2013, *Fiona 1.0.2: Python Package Index,* viewed 21 November 2013, https://pypi.python.org/pypi/Fiona

The Python Community 2013, *Open arbitrary resources by URL Python Package Index,* viewed 21 November 2013, http://docs.python.org/2/library/urllib.htm

The Python Community 2013, *Shapely 1.2.18: Python Package Index,* viewed 21 November 2013, https://pypi.python.org/pypi/Shapely

The Python Community 2014, Pika 0.9.14*: Python Package Index,* viewed 19 November 2014, https://pypi.python.org/pypi/pika

The Python Community 2014, pyproj 1.9.3: *Python Package Index*, viewed, 18 November 2014, https://pypi.python.org/pypi/pyproj

Tiede, D, & Lang, S 2010, Analytical 3D views and virtual globes—scientific results in a familiar spatial context, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol.*65*(3), pp. 300-307

Vaquero, LM, Rodero-Merino, L, Caceres, J, Linder, M 2008. A break in the clouds: towards a cloud definition, *ACM SIGCOM Computer Communication Review*, vol. 39(1), pp. 50-55

Vieweg, S, Hughes, AL, Starbird, K & Palen, L 2010, Microblogging during two natural hazard events: What Twitter may contribute to situational awareness, *CHI 2010: Crisis Informatics*. 10-15 April 2014, Atlanta, Georgia, USA

Wayner, P 2013, Benchmarking Amazon EC2: The wacky world of cloud performance, viewed 27 October 2014, http://www.datadoghq.com/wp-content/uploads/2013/07/top_5_aws_ec2_performance_problems_ebook.pdf

Westerholt, R, & Resch, B 2014, Asynchronous Geospatial Processing: An Even Driven Push Based Architecture for the OGC Web Processing Service, *Transactions in GIS*, doi: 10.1111/tgis.12104

Wild, S, 2013, Hot warning system for tracking fires, *Mail and Guardian*, viewed 26 September 2014, http://mg.co.za/article/2013-09-06-00-hot-warning-system-for-tracking fires

Working on Fire 2012, Fire in South Africa, viewed 3 November 2014, http://www.workingonfire.org/index.php/about-wof/fire-in-sa

WorldPop, 2013, WorldPop Population Map, viewed 20 August 2013, http://www.worldpop.org.uk/data/summary/?contselect=Africa&countselect=South+Africa&typeselect=Population

Wu, H, He X, Gong, J 2010, A virtual globe-based 3D visualization and interactive framework for public participation in urban planning processes, *Computers, Environment and Urban Systems*, vol.34, pp. 291-298

Yang, C, Xu, Y, & Nebert, D 2013, Redefining the possibility of digital Earth and geosciences with spatial cloud computing. *International Journal of Digital Earth*, vol. *6*(4), pp. 297-312

Yun, S, Chen, C, Li, J & Tang, L 2011, Wildfire spread simulation and visualization in virtual environments. International Conference on *Spatial Data Mining and Geographical Knowledge Services (ICSDM),* Fuzhou,pp. 315-31

Appendix A - Source Code

The code developed for the experimental geoprocessing system can be found at:
https://www.dropbox.com/sh/m9v695zq4p7e56e/AAAe3iNRhJnLXYaFqPQ8pUBga?dl=1

Appendix B - Extended Results

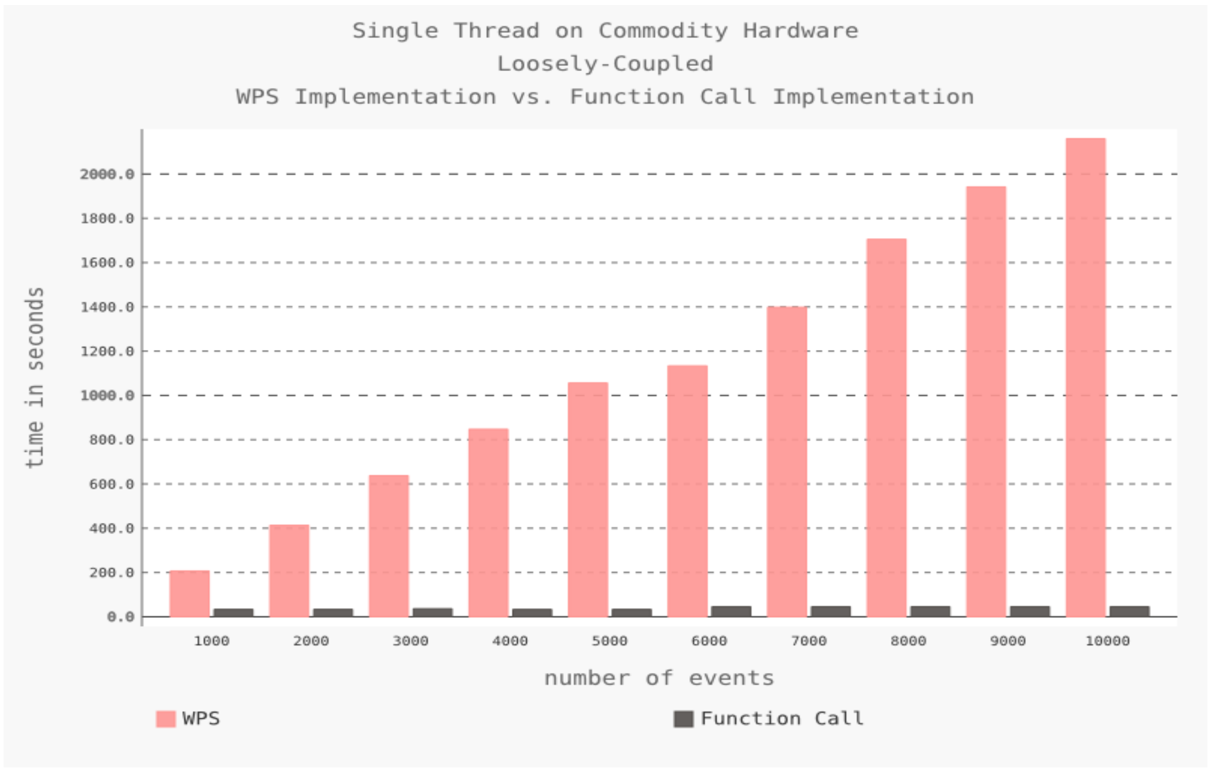


Figure 16: Single Thread on Commodity Hardware Loosely-Coupled WPS Implementation versus Function Call Implementation. One Consumer per Geoprocessing Chain Component

It can be observed from the graph in *Figure 16*, that a loosely-coupled function call implementation produces output faster than a loosely-coupled Web Processing Service implementation. The processes were executed on commodity hardware. The performance of the function call implementation remained consistent whist the performance of the Web Processing Service slowed down linearly as the number of messages (events) increased.

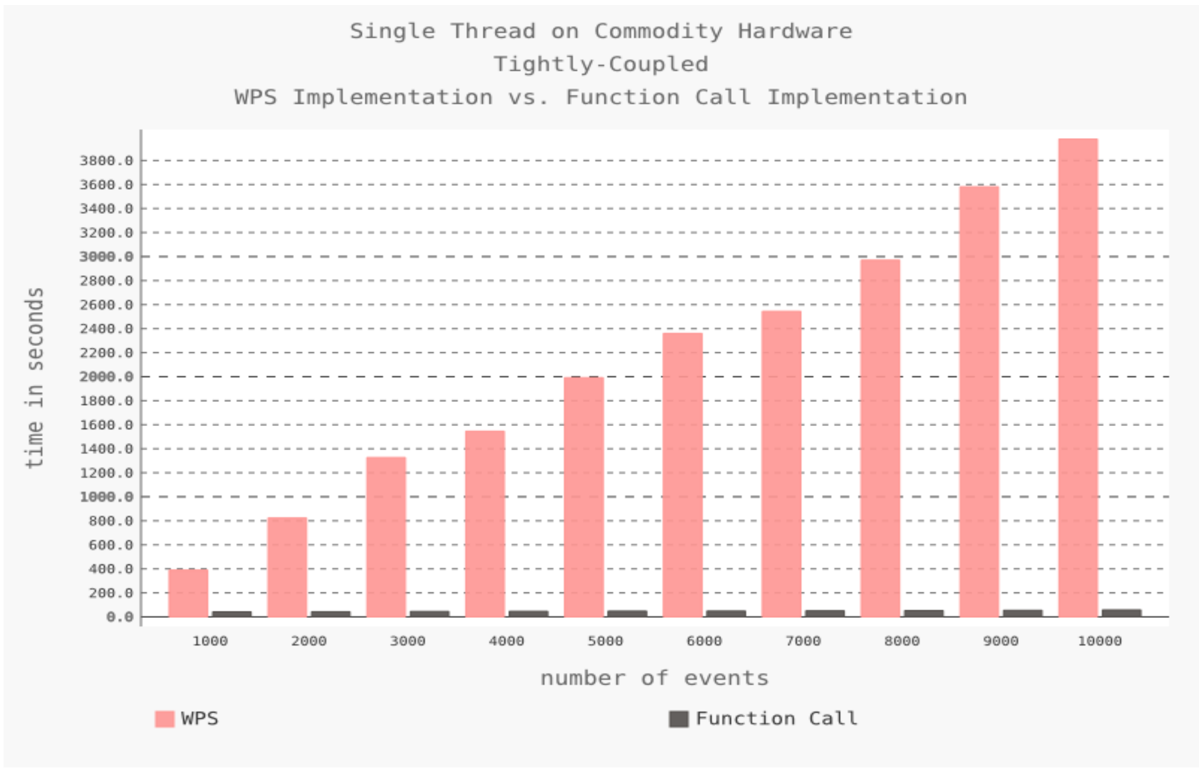


Figure 17: Single Thread on Commodity Hardware Tightly-Coupled WPS Implementation versus Function Call Implementation

It can be noted from the graph in *Figure 17*, that a tightly-coupled function call implementation produces output faster than a tightly-coupled Web Processing Service implementation. The processes were executed on commodity hardware. The performance of the function call implementation remained consistent whilst the performance of the Web Processing Service slowed down linearly as the number of messages (events) increased.

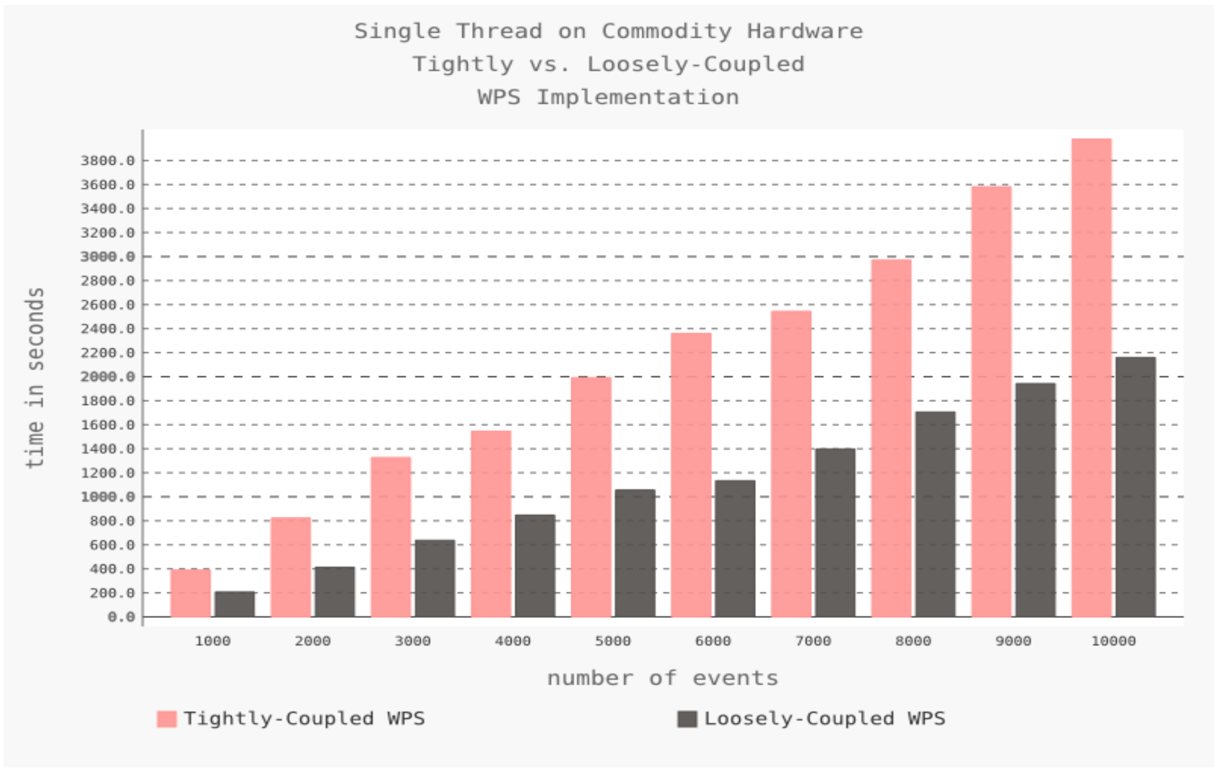


Figure 18: Single Thread on Commodity Hardware Tightly-Coupled versus Loosely-Coupled WPS Implementation

Referring to *Figure 18,* it can be noted that a loosely-coupled Web Processing Service implementation produces output quicker than a tightly-coupled Web Processing Service implementation. The processing was conducted on commodity hardware. As the number of messages (events) increased, the processing time slowed down linearly.

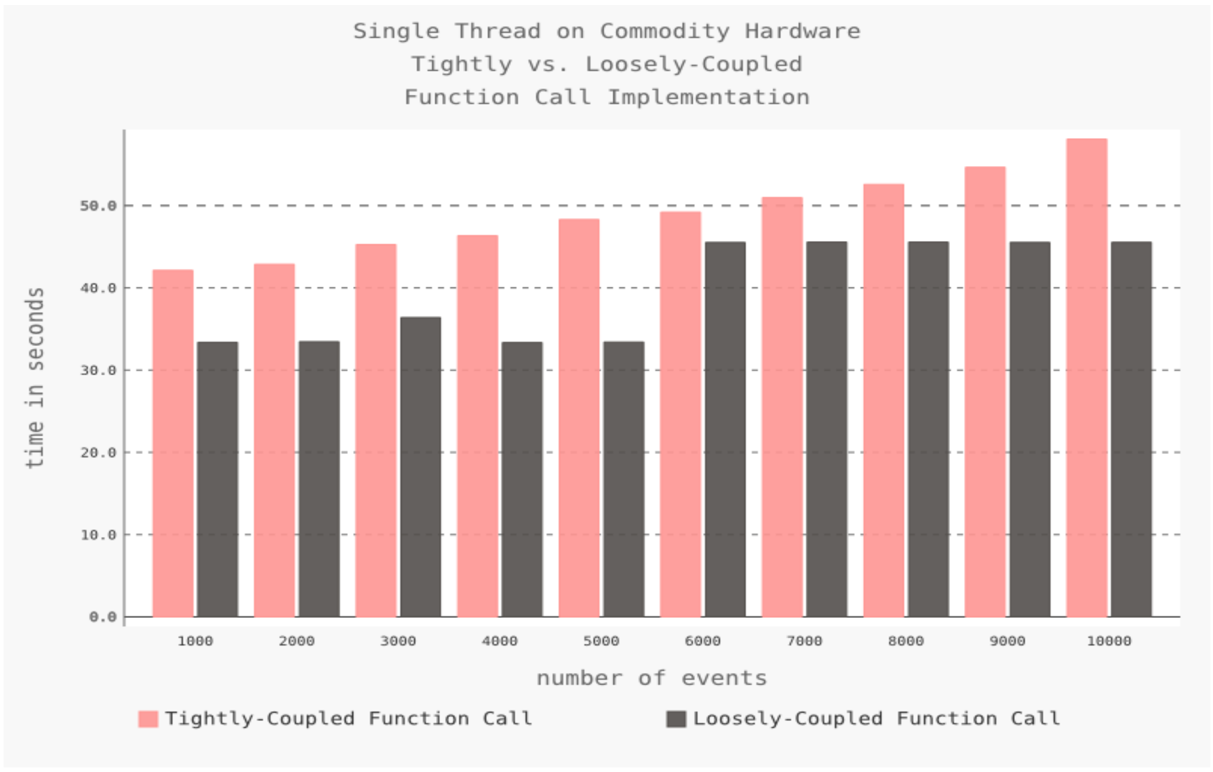


Figure 19: Single Thread on Commodity Hardware Tightly-Coupled versus Loosely-Coupled Function Call Implementation

It can be observed from *Figure 19,* that a loosely-coupled function call implementation produces output faster than a tightly-coupled function call implementation. The processing was conducted on commodity hardware. The tightly-coupled implementation had a linear decrease in performance (slow down) whilst no pattern can be viewed with the loosely-coupled implementation.

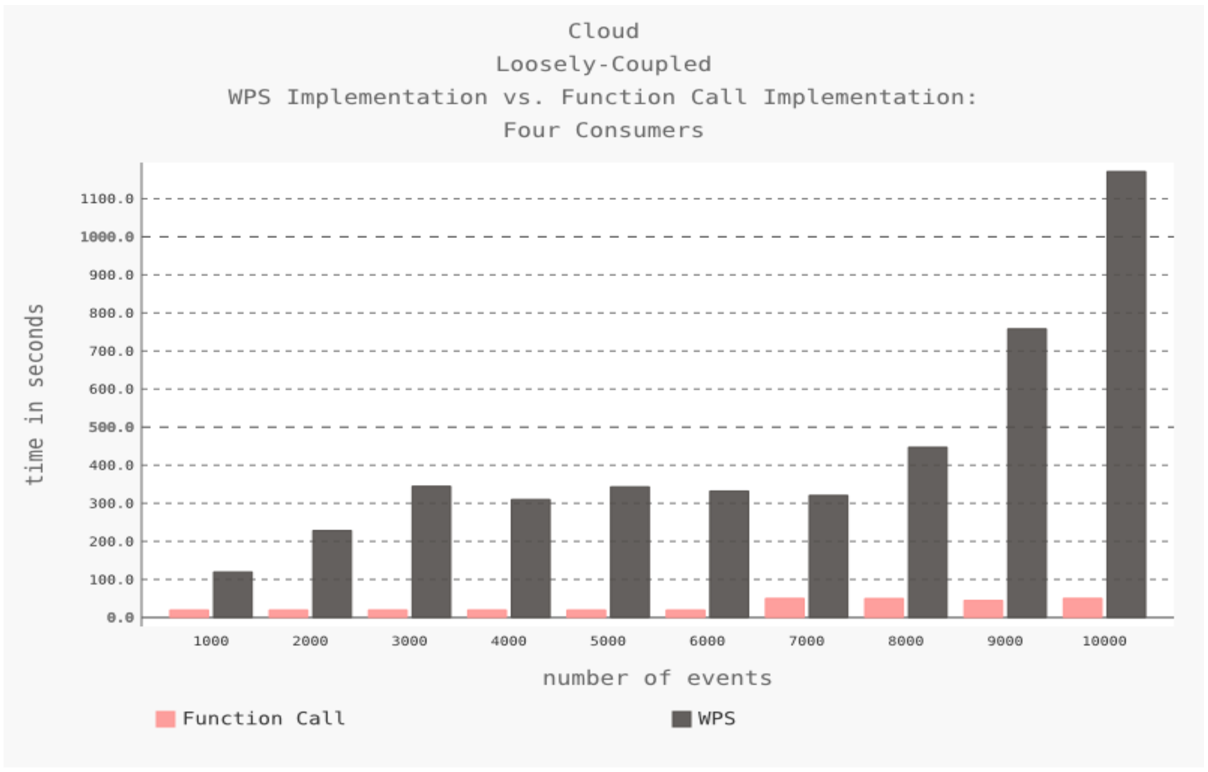


Figure 20: Cloud-Based Loosely-Coupled WPS Implementation versus Function Call Implementation. Four Consumers per Geoprocessing Chain Component

It can be noted from *Figure 20*, that a loosely-coupled cloud-based function call implementation that utilises four consumers per component of the geoprocessing chain, produces results in much less time than a cloud-based Web Processing Service implementation that utilises four consumers per component of the geoprocessing chain. The performance of the function call implementation was almost constant except for when 7000, 8000, 9000 and 1000 messages (events) were processed. There was a decrease (slow down) in geoprocessing performance.

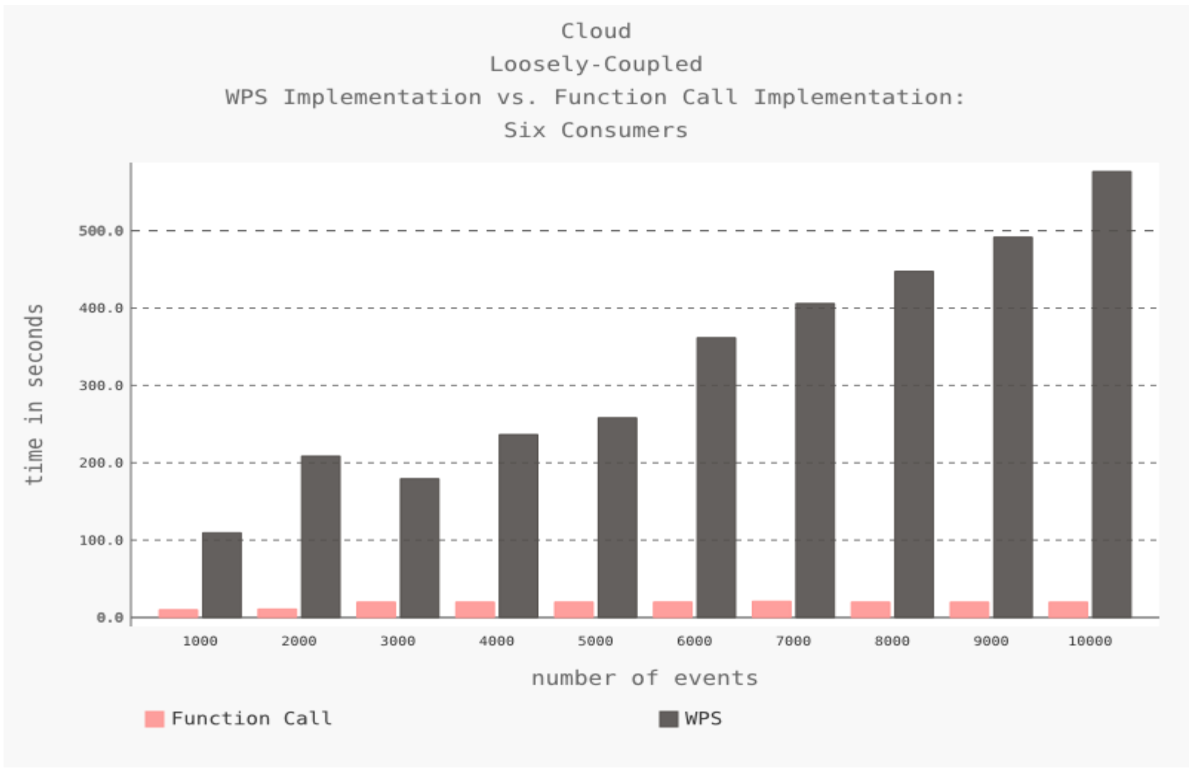


Figure 21: Cloud-Based Loosely-Coupled WPS Implementation versus Function Call Implementation. Six Consumers per Geoprocessing Chain Component

It can be observed from *Figure 21*, that a loosely-coupled cloud-based function call implementation that utilises six consumers per component of the geoprocessing chain, produces results in less time than a cloud-based Web Processing Service implementation that utilises six consumers per component of the geoprocessing chain. A linear decrease in performance of the Web Processing Service implementation can be viewed as the number of messages (events) increased. There was one outlier (spike), when 2000 messages were processed. The performance of the function call implementation was almost consistent, except for the decrease in performance between 2000 and 3000 events.

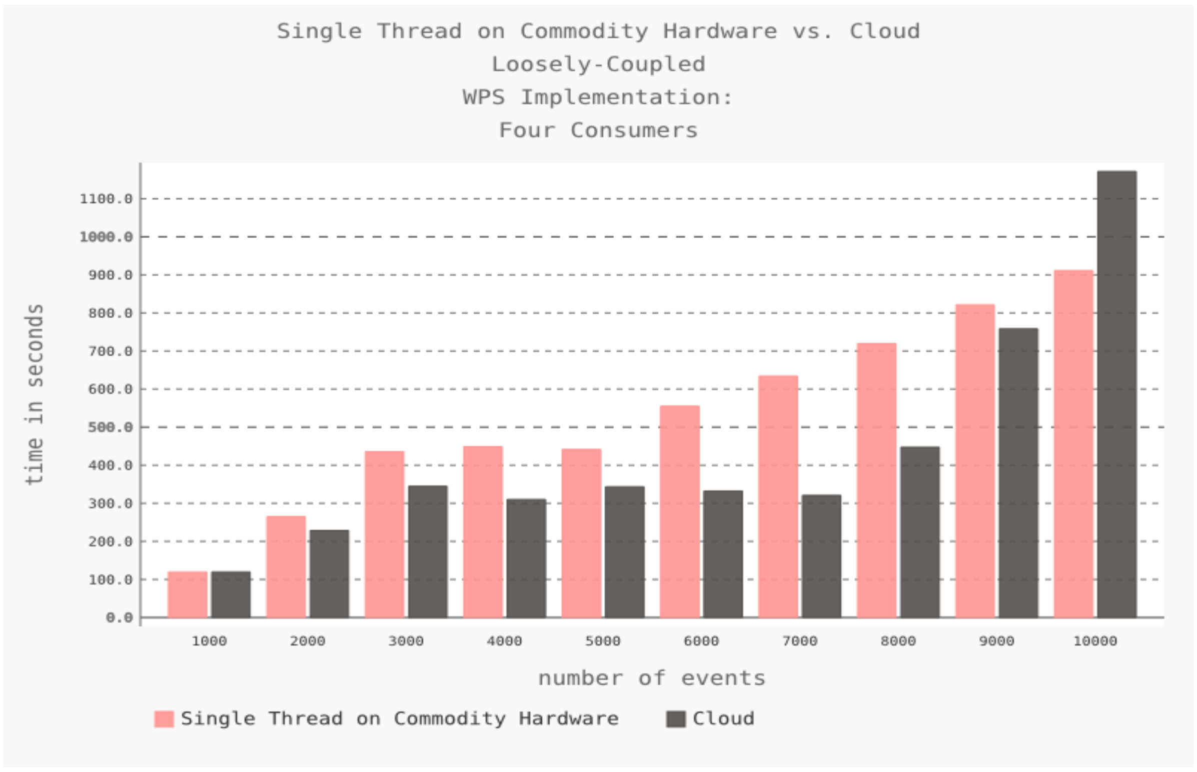


Figure 22: Single Thread on Commodity Hardware versus Cloud-Based Loosely-Coupled WPS Implementation. Four Consumers per Geoprocessing Chain Component

From *Figure 22*, it can be observed that a cloud-based loosely-coupled Web Processing Service implementation that utilises four consumers per geoprocessing chain component delivers output quicker than a commodity hardware-based loosely-coupled Web Processing Service implementation that utilises four components per geoprocessing chain component in most cases. The single outlier was when 10000 messages (events) were processed.

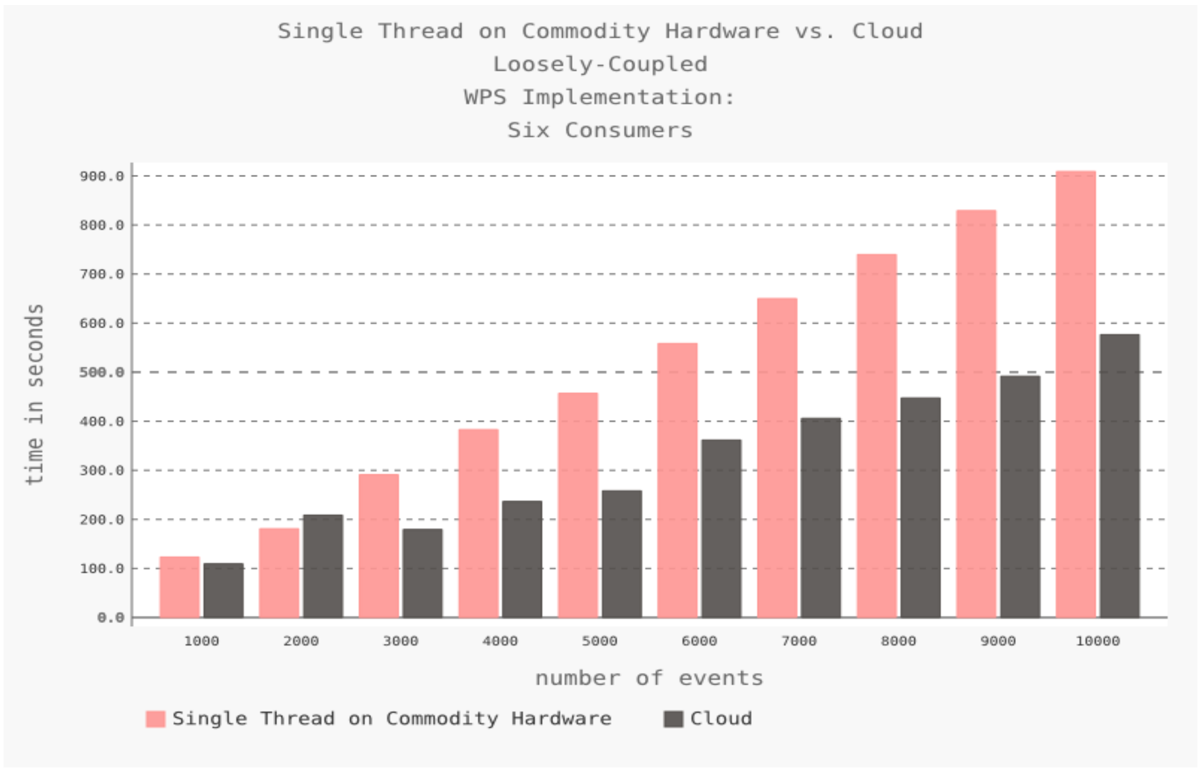


Figure 23: Single Thread on Commodity Hardware versus Cloud-Based Loosely-Coupled WPS Implementation. Six Consumers per Geoprocessing Chain Component

It can be noted from *Figure 23*, that a cloud-based loosely-coupled Web Processing Service implementation that utilises six consumers per geoprocessing chain component produces output quicker than a commodity hardware-based loosely-coupled Web Processing Service implementation that utilises six components per geoprocessing chain component, in most cases. The one outlier that exists is when 2000 messages (events) were processed.

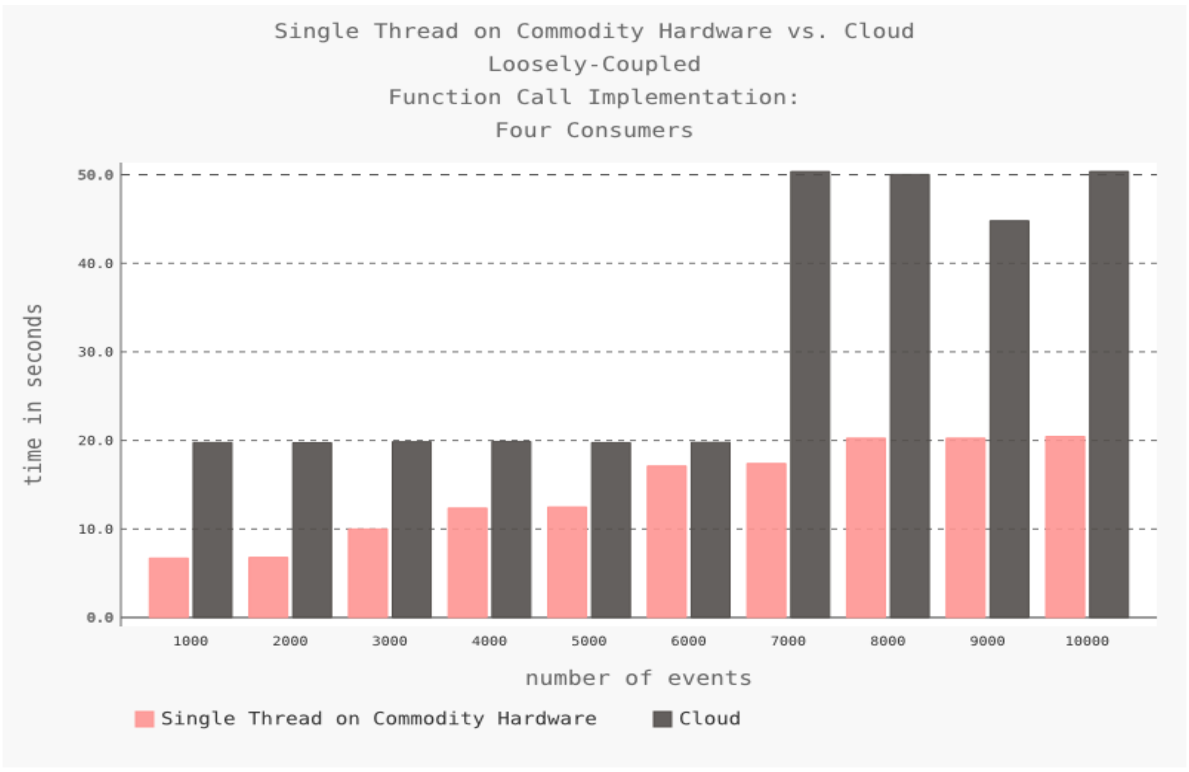


Figure 24: Single Thread on Commodity Hardware versus Cloud-Based Loosely-Coupled Function Call Implementation. Four Consumers per Geoprocessing Chain Component

It can be observed from *Figure 24*, that a loosely-coupled commodity hardware-based function call implementation that utilises four consumers per geoprocessing chain component produces output faster than a cloud-based loosely-coupled function call implementation that utilises four components per geoprocessing chain component.

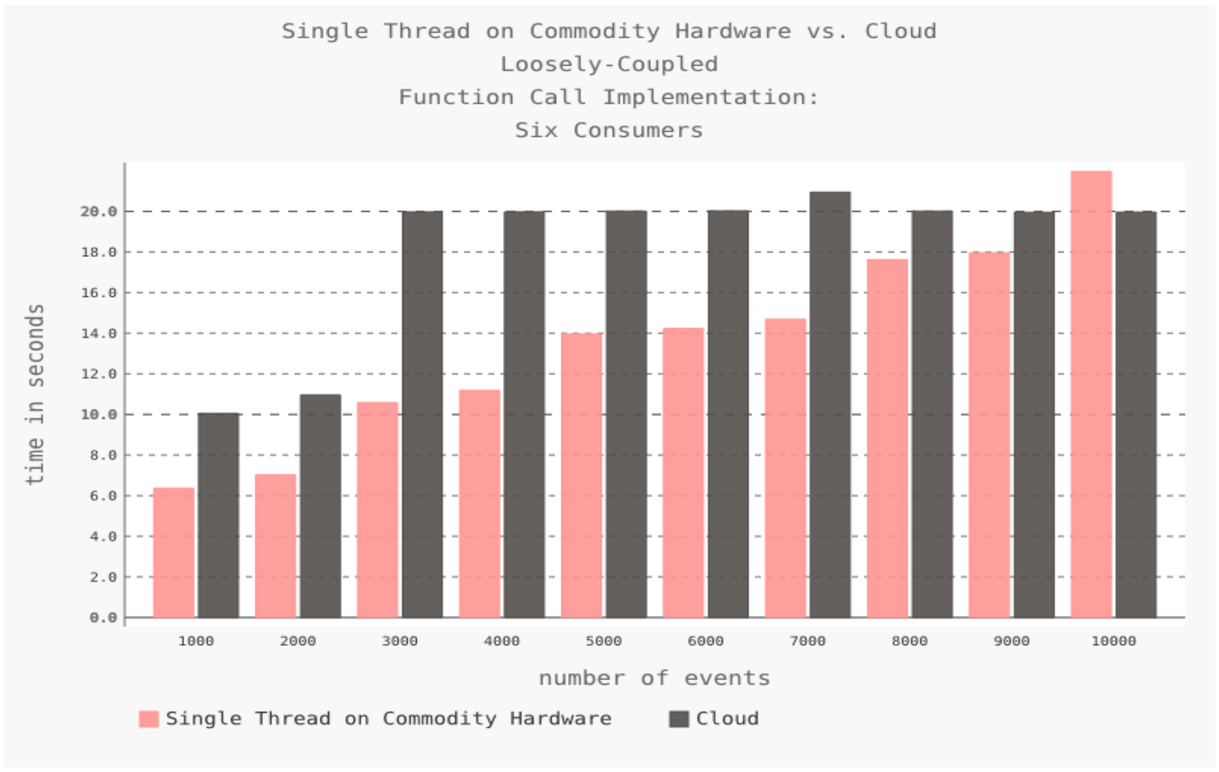


Figure 25: Single Thread on Commodity Hardware versus Cloud-Based Loosely-Coupled Function Call Implementation. Six Consumers per Geoprocessing Chain Component

From *Figure 25* it can be noted that a loosely-coupled commodity hardware-based function call implementation that utilises six consumers per geoprocessing chain component produces output faster than a cloud-based loosely-coupled function call implementation that utilises six components per geoprocessing chain component. The processing time of the instance where 10000 events were processed, was an anomaly.

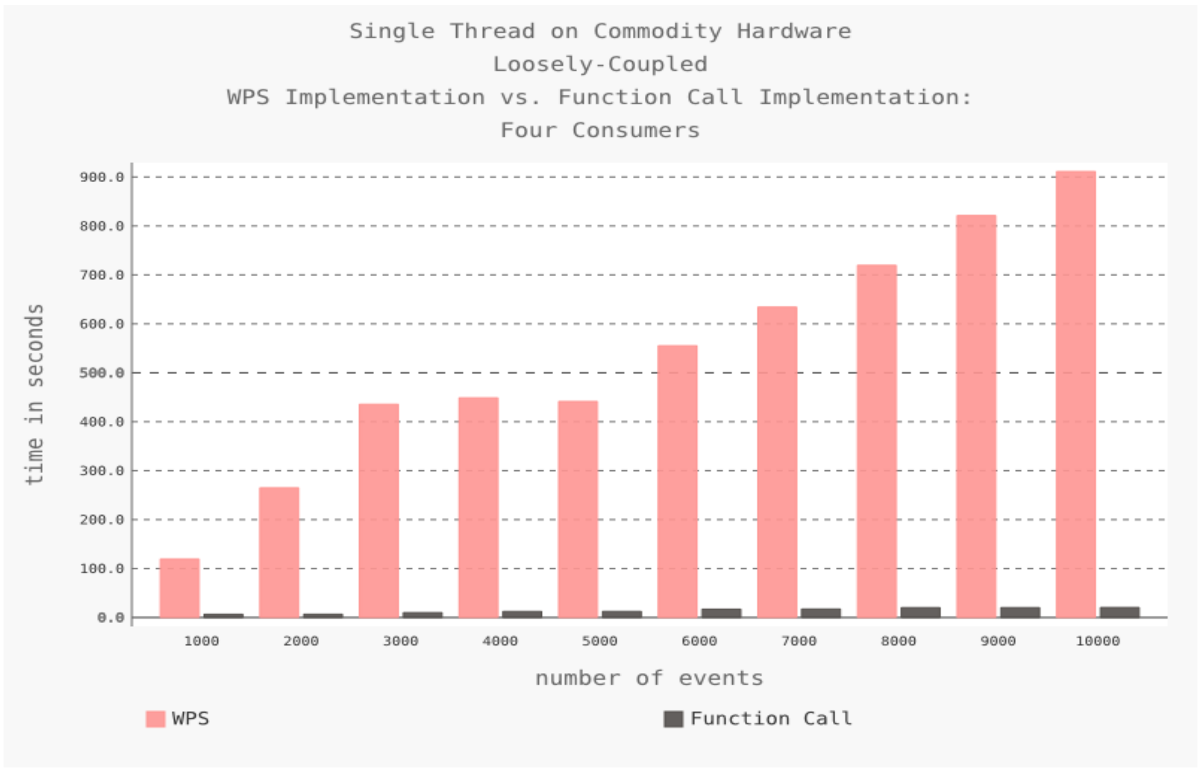


Figure 26: Single Thread on Commodity Hardware Loosely-Coupled WPS Implementation versus Function Call Implementation. Four Consumers per Geoprocessing Chain Component

From *Figure 26* it can be observed that a loosely-coupled function call implementation can produce output faster than a loosely-coupled Web Processing Service implementation that utilises four consumers per geoprocessing chain component. The performance of the function call implementation was constant and there was an almost linear decrease in the performance of the Web Processing Service implementation.

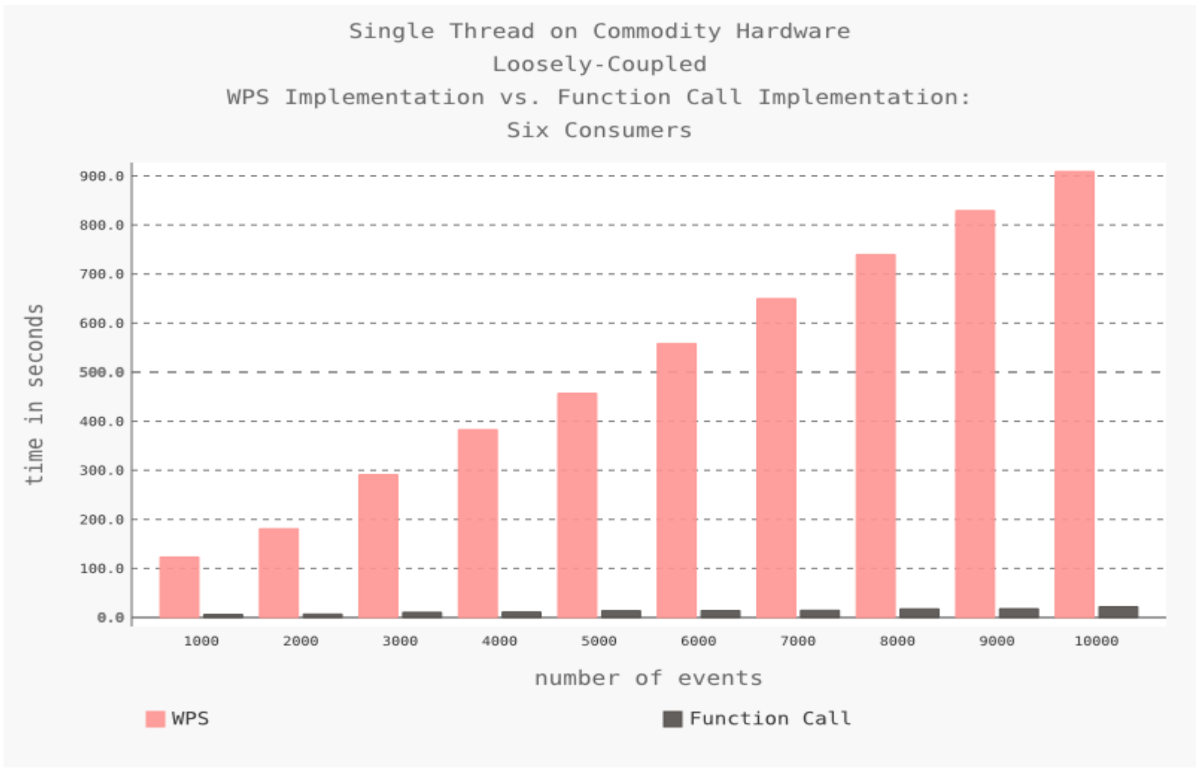


Figure 27: Single Thread on Commodity Hardware Loosely-Coupled WPS Implementation versus Function Call Implementation. Six Consumers per Geoprocessing Chain Component

It can be observed from *Figure 27* that a loosely-coupled function call implementation can produce output faster than a loosely-coupled Web Processing Service implementation that utilises six consumers per geoprocessing chain component. The performance of the function call implementation was constant and there was a linear decrease in the performance of the Web Processing Service implementation.

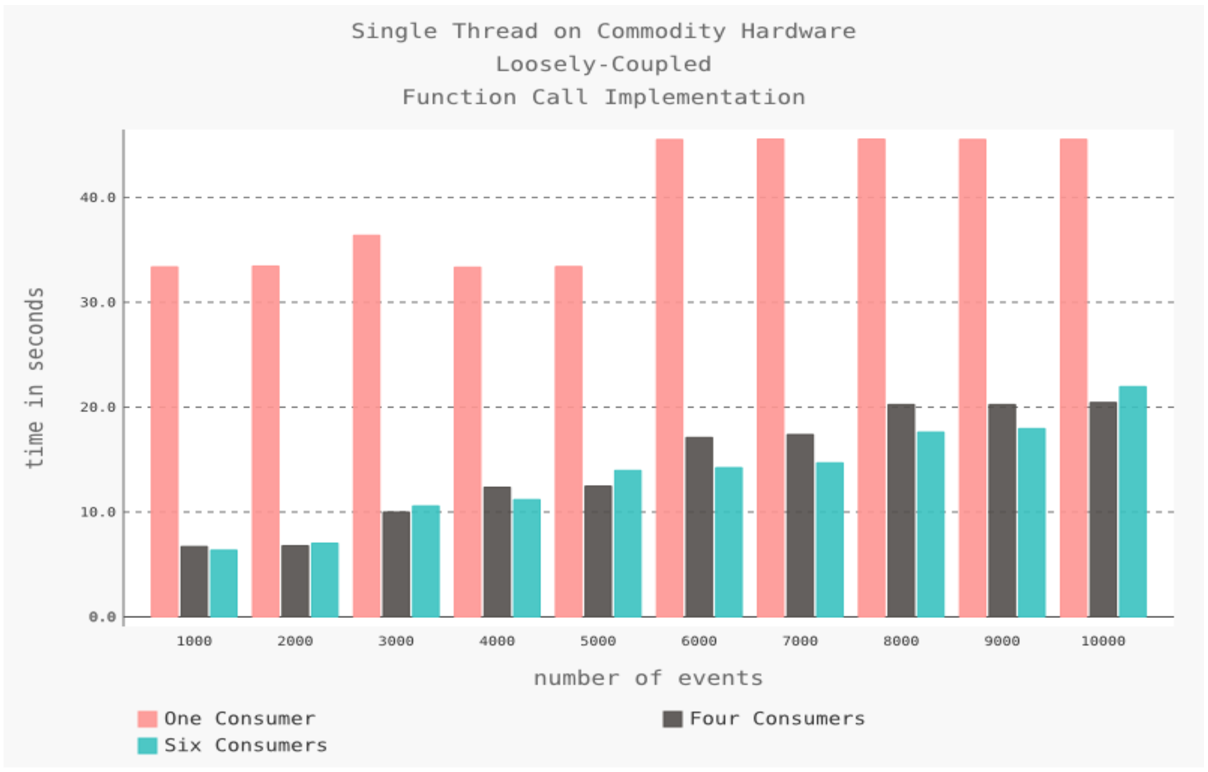


Figure 28: Single Thread on Commodity Hardware Loosely-Coupled Function Call Implementation. One versus Four versus Six Consumers per Geoprocessing Chain Component

It can be observed from *Figure 28* that message (event) processing time will decrease as more consumers are added. This is the case in six out of ten instances (when 1000, 4000, 6000, 7000, 8000 and 9000 events were processed). The implementation utilised is the loosely-coupled function call implementation.

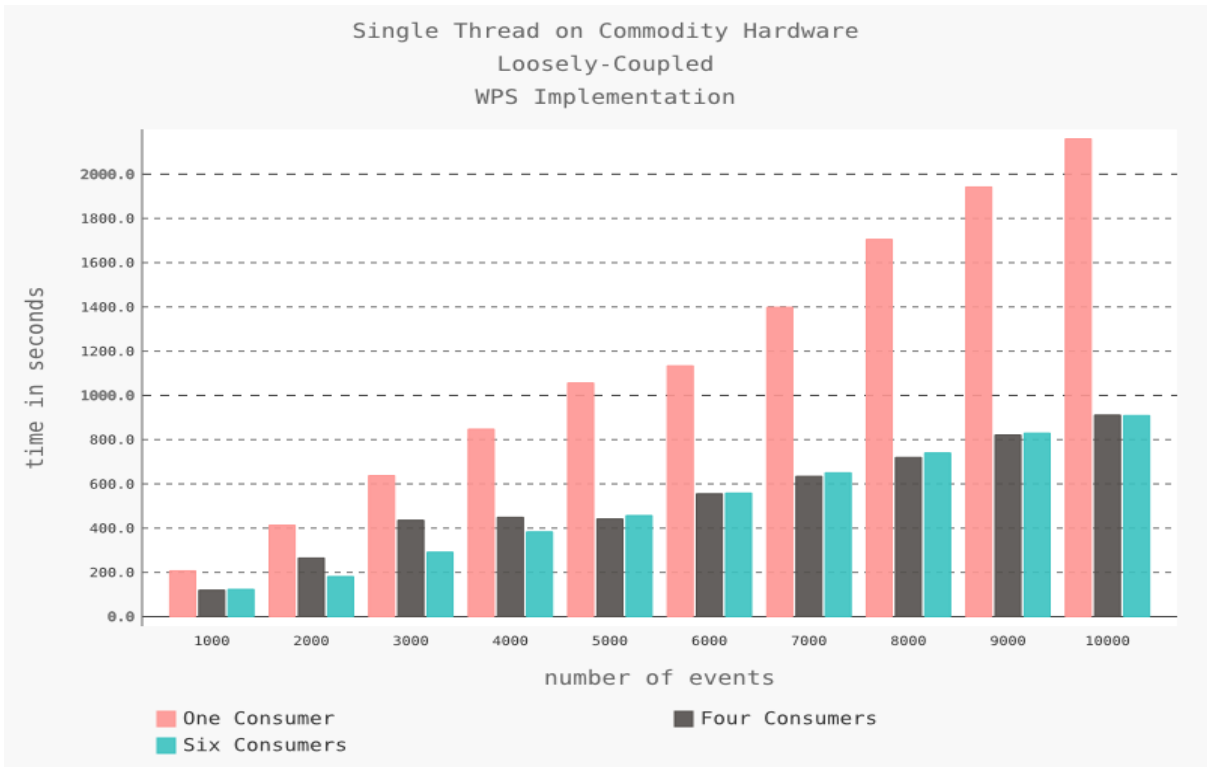


Figure 29: Single Thread on Commodity Hardware Loosely-Coupled WPS Implementation. One versus Four versus Six Consumers per Geoprocessing Chain Component

From *Figure 29* it can be noted that the performance of the Web Processing Service implementation becomes unpredictable. It was expected for the processing time to decrease as the number of consumers increased. This was not the case, it merely happened for three out of ten instances (when 2000, 3000 and 4000 events were processed).

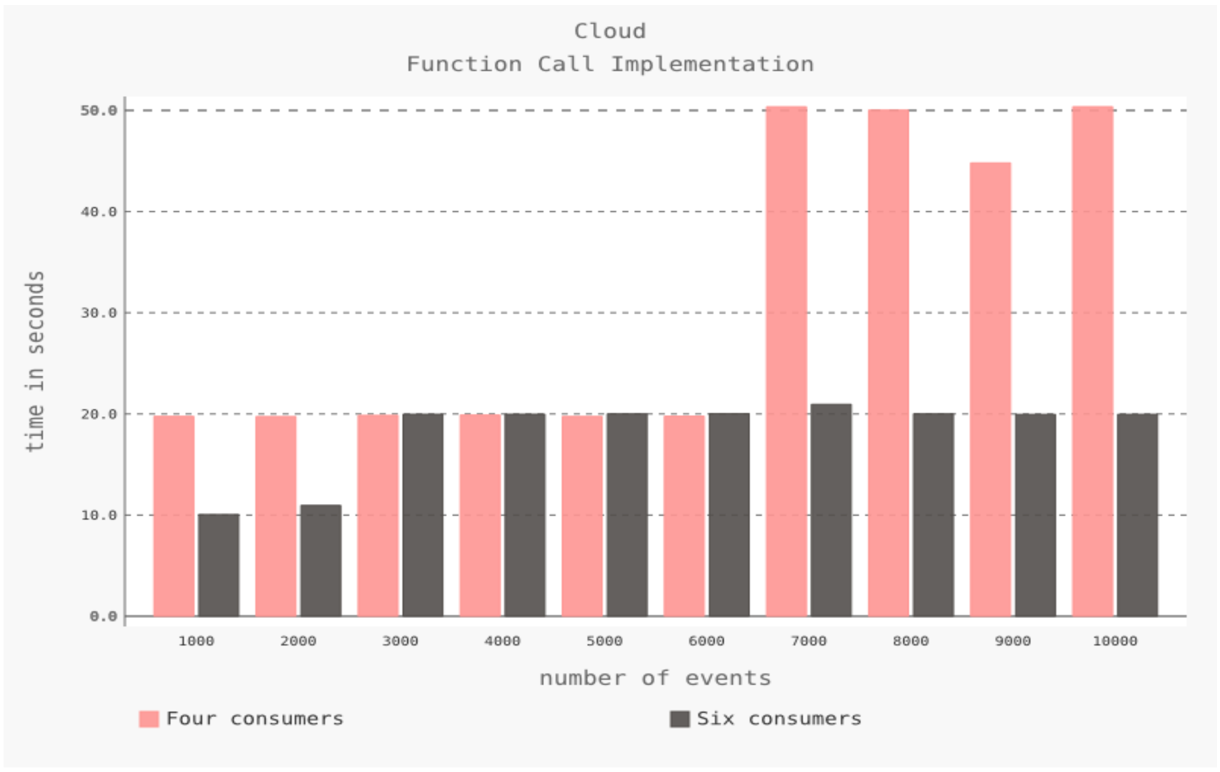


Figure 30: Cloud-Based Function Call Implementation. Four versus Six Consumers per Geoprocessing Chain Component

Referring to *Figure 30*,it can be observed that the cloud-based function call implementation provided inconsistent processing times. It was expected for the event processing times to decrease as consumers were added. The expectation was only met when 1000, 3000, 7000, 8000, 9000 and 1000 events were processed.

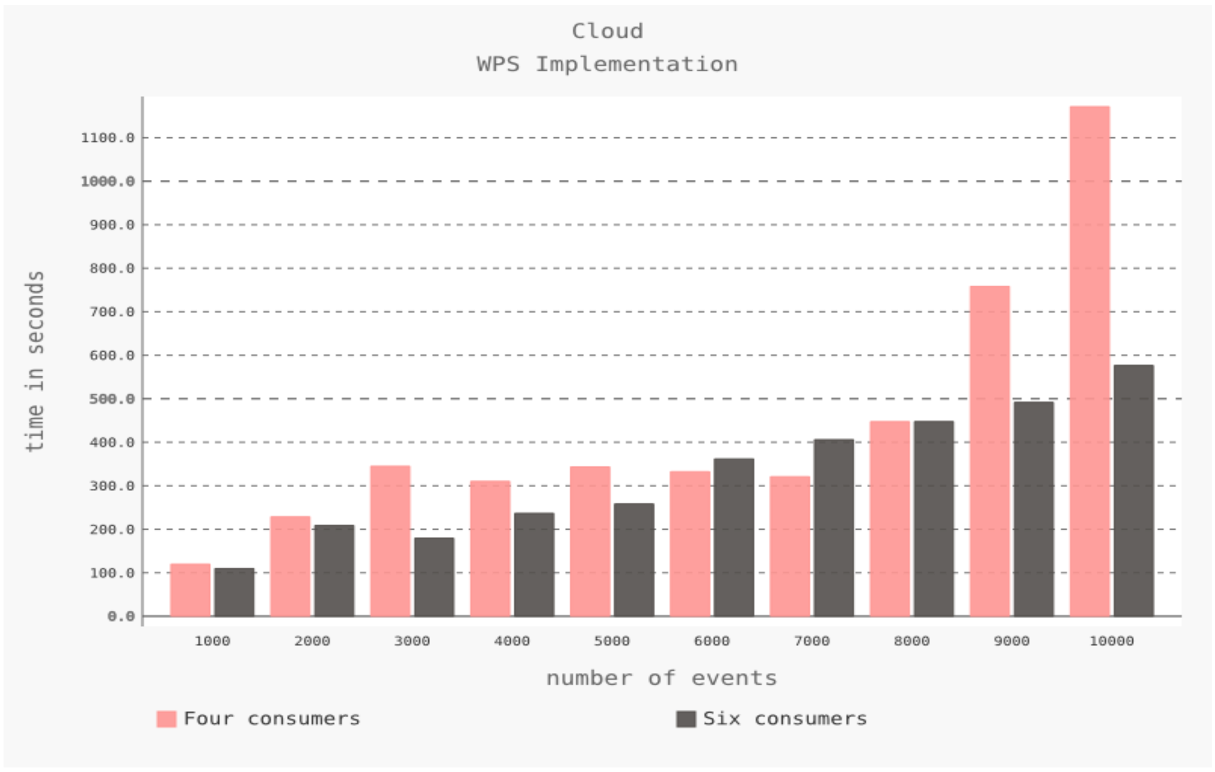


Figure 31: Cloud-Based WPS Implementation. Four versus Six Consumers per Geoprocessing Chain Component

From *Figure 31*, it can be noted that by processing messages (events) using a cloud-based Web Processing Service implementation and by increasing the number of consumers per geoprocessing chain component, the processing time will decrease in most cases (when 1000, 2000, 3000, 4000, 5000, 9000 and 1000 events were processed).

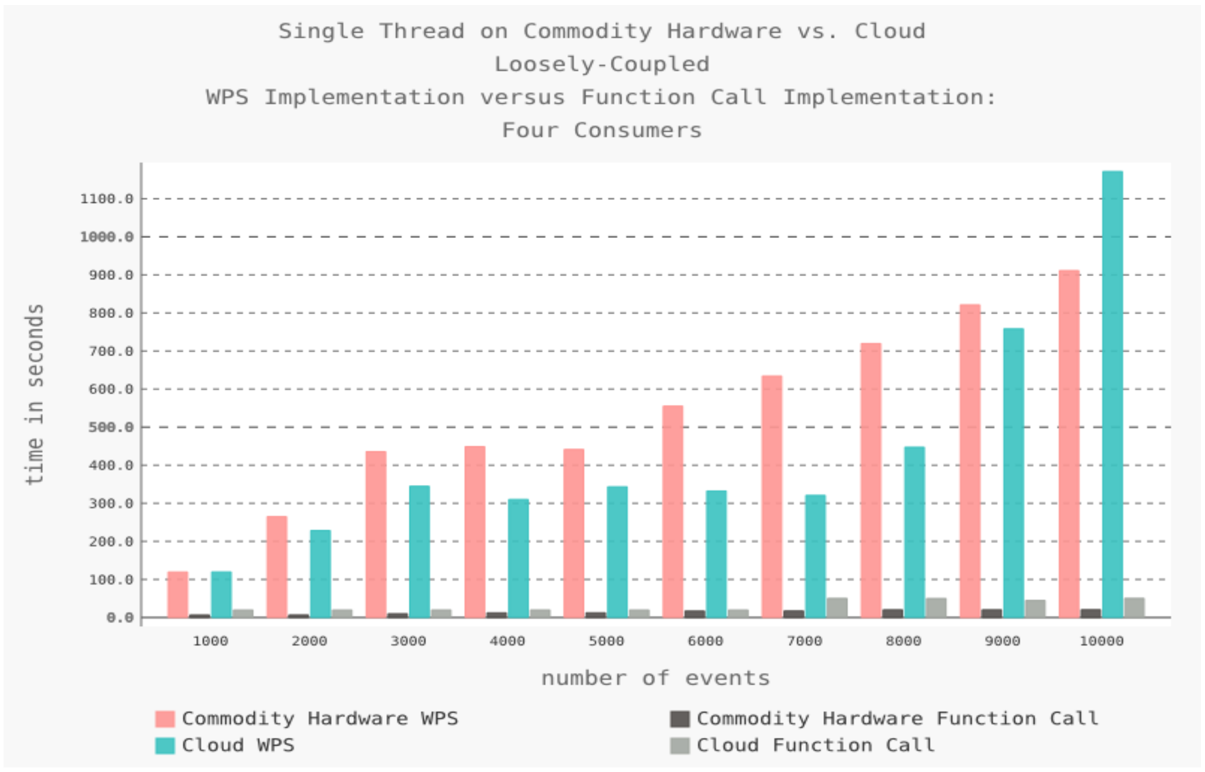


Figure 32: Single Thread on Commodity Hardware versus Cloud-Based Loosely-Coupled WPS versus Function Call Implementation. Four Consumers per Geoprocessing Chain Component

It can be noted from *Figure 32*, that the commodity hardware-based function call implementation that utilises four consumers per geoprocessing chain component produced output in the fastest time. The cloud-based function call implementation performed the second best. The cloud-based Web Processing Service implementation produced output in the third fastest time and the commodity hardware-based Web Processing Service implementation produced output in the slowest time.

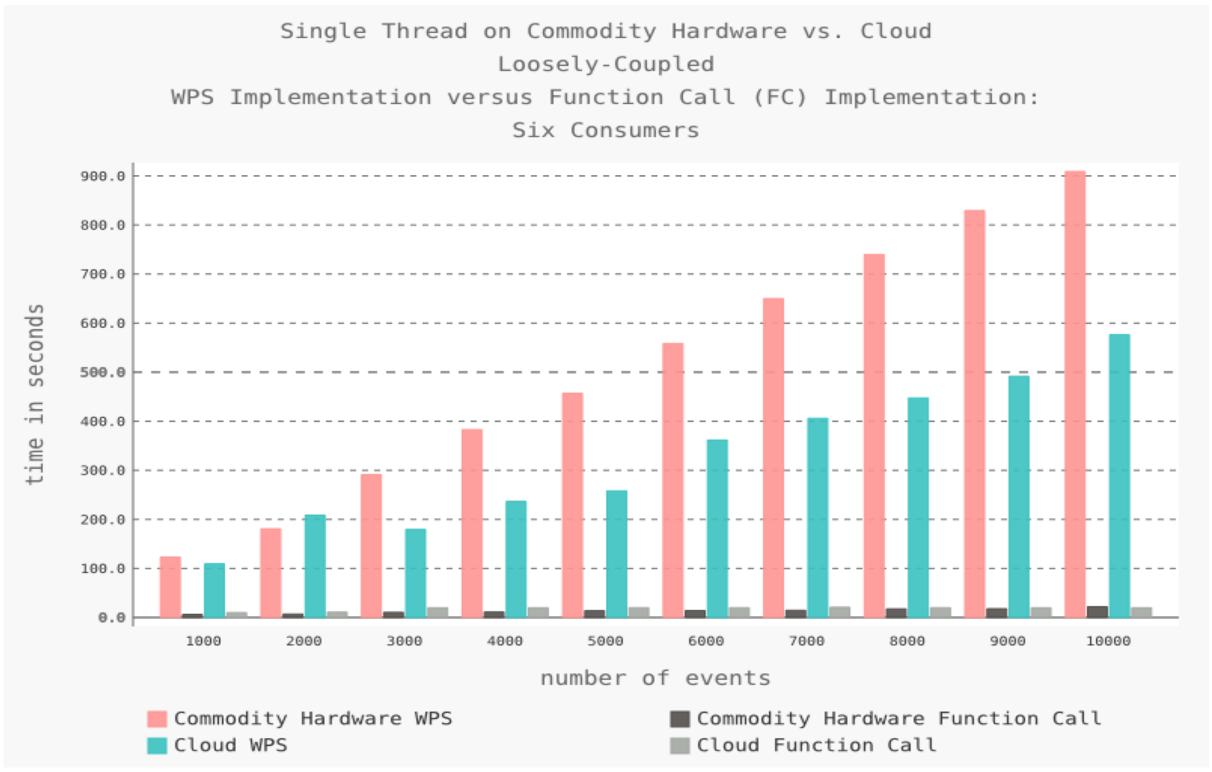


Figure 33: Single Thread on Commodity Hardware versus Cloud-Based Loosely-Coupled WPS versus Function Call Implementation. Six Consumers per Geoprocessing Chain Component

It can be noted from *Figure 33*, that the cloud-based function call implementation that utilises six consumers per geoprocessing chain component produced output in the fastest time. The commodity hardware-based function call implementation produced output in the second fastest time. The cloud-based Web Processing Service implementation produced output in the third fastest time and the commodity hardware-based Web Processing Service implementation produced output in the slowest time.

Appendix C - Categorised References

Peer-Reviewed Scientific Literature

Akioka, S,& Muraoka, Y 2010, HPC benchmarks on Amazon EC2, *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference,* 20-23 April 2010, Perth, Australia, pp. 1029-1034, doi: 10.1109/WAINA.2010.166

Armbrust, M, Fox, A, Griffith, R, Joseph, AD, Katz, R, Konwinski, A & Zaharia, M 2010. A view of cloud computing, *Communications of the ACM*, vol. *53*(4), pp. 50-58

Baranski, B, Schaffer, B, Lange, K & Foerster, T 2010, *Geoprocessing in hybrid clouds*, Geoinformatik, 2010, Kiel, Germany, pp. 13-19

Brown, RB, Smoot, JC, Underwood, L & Armstrong, CD 2012, Investigation into Cloud Computing for more robust automated bulk image geoprocessing, *MAPPS/ASPRS Speciality Concurrence,* 29 October – 1 November 2012, Tampa, Florida, United States

Castrillón, M, Jorge, PA, López, IJ, Macías, A, Martín, D, Nebot, RJ & Trujillo, A 2011. Forecasting and visualization of wildfires in a 3D geographical information system, *Computers & Geosciences*, vol. *37* (3), pp. 390-396

Dasgupta, A & Ghosh, SK 2011, Service chaining for accessing geospatial information in mobile devices.*Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications,* ACM, doi: 10.1145/1999320.1999343

Davis, DK, Vosloo, HF, Frost, PE & Vannan, SS 2008, Near real-time fire alert system in South Africa: From desktop to mobile service, *Proceedings of the 7th ACM Conference on Designing Interactive Systems,* 25 – 27 February 2008, Cape Town, South Africa, pp. 315-322

Dillon, T, Wu, C, Chang, E Chang 2010, Cloud Computing: Issues and Challenges, *24th IEEE International Conference on Advanced Information Networking and Application Cloud Computing*, 20-23 April 2010, Perth, Australia, pp.27-33

Eugster, PTH, Felber, PA, Guerraoui, A & Kermarrec, AM 2003, The many faces of publish/subscribe, *ACM computing surveys,* vol.35, pp. 114-131

Evangelidis, K, Ntouros, K, Makridis, S & Papatheodorou, C 2014, Geospatial services in the Cloud, *Computers & Geosciences*, vol. *63*, pp. 116-122

Fernandes, JL, Lopes, IC, Rodrigues, JJ, & Ullah, S 2013, Performance evaluation of Restful Web services and AMQP protocol. The Fifth International Conference on *Ubiquitous and Future Networks,* 2-5 July 2013, IEEE, pp. 810-815, doi:10.1109/ICUFN.2013.6614932.

Giuliani, G, Nativi, S, Lehmann, A, & Ray, N 2012, WPS mediation: An approach to process geospatial data on different computing backends, *Computers & Geosciences*, vol. *47*, pp. 20-33

Goldberg, D, Olivares, M, Li, Z & Klein, AG 2014, Maps and GIS Data Libraries in the Era of Big Data and Cloud Computing, *Journal of Map and Geography Libraries,* vol.10, pp. 100-120.

Gong, J, Zhou, H & Yue, P 2010, Geoprocessing in the Microsoft Cloud Computing Platform-Azure, *International Journal of Digital Earth, vol.*6, pp. 404-425

Hildebrandt, D & Döllner, J 2010, Service-oriented, standards-based 3D geovisualization: Potential and challenges, *Computers, Environment and Urban Systems,* vol. 34(6), pp. 484-495

Huang, Q, Yang, C, Liu, K, Xia, J, Xu, C, Li, J, & Li, Z 2013, Evaluating open-source cloud computing solutions for geosciences, *Computers & Geosciences*, vol. *59*, pp. 41-52

Hyong-Woo, K, Dae-Sun, K, Yang-Won, L & Jae-Seong, A 2014, 3-D Geovisualization of satellite images on smart devices by the integration of spatial DBMS, RESTful API and WebGL, *Geocarto International*, doi: 10.1080/10106049.2014.888485

Jadeja, Y & Modi, K 2012, Cloud computing-concepts, architecture and challenges. *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET), Kumaracoil,* pp. 877-880, doi:10.1109/ICCEET.2012.6203873

Johnsen, FT, Bloebaum, TH, Avlesen, M, Spjelkavik, S,& Vik, B 2013, Evaluation of transport protocols for web services. *Military Communications and Information Systems Conference (MCC),* St. Malo, pp. 1-6

Juve, G, Deelman, E, Berriman, GB, Berman, BP, & Maechling, P 2012, An evaluation of the cost and performance of scientific workflows on Amazon EC2, *Journal of Grid Computing*, vol. *10* (1), pp. 5-21

Kassab, A, Liang, S & Gao, Y 2010, Real-time notification and improved situational awareness in fire emergencies using geospatial-based publish/subscribe, *International Journal of Applied Earth Observation and Geoinformation, vol. 12(6), pp. 431-438*

Kiehle, C, Greve, K & Heier, C 2007, Requirements for next generation spatial data infrastructures -standardized web based geoprocessing and web service orchestration, *Transactions in GIS,* vol.11, pp. 819-834

Kim, HW, Kim ,DS, Lee ,YW & Ahn ,JS 2014, 3-D Geovisualization of satellite images on smart devices by the integration of spatial DBMS, RESTful API and WebGL, *Geocarto International,*(ahead-of-print), pp.1-19

Kmoch, A & Klug, H 2014, Visualization of 3D Hydrogeological Data in the Web, GI Forum

2014 Proceedings, Symposium and Exhibit Geospatial Innovation for Society, 1-4 July 2014, Salzburg, Austria, viewed 24 August 2013, http://hw.oeaw.ac.at/0xc1aa500d%200x0030d3d3.pdf

Kraak, MJ n.d., The Space-time cube revisited from a geovisualization perspective, *Proceedings of the 21st International Cartographic Conference*.

Kumar, A & Bawa S 2012, Distributed Big Data Storage Management in Grid Computing, *International Journal of Grid Computing and Applications (IJGCA),* vol. 2, pp. 28-29

MacEachren, AM, Kraak, MJ 2001, Research Challenges in Geovisualisation, *Cartography and Geographic Information Sciences*, vol.28 (1), pp. 3-2, DOI:10.1559/152304001782173970.

McCullough, A, James, P, Barr, S 2011, A Typology of Real-Time Parallel Geoprocessing for the Sensor Web Era, *Proceedings of the Workshop on Integrating Sensor Web and Web-Based Geoprocesssing*, AGILE, 18 April 2011, Utrecht, CEUR, pp. 1-5

McFerren, G & Frost, P 2009, The South African Advanced Fire Information System, *Proceedings of the 6th International ISCRAM Conference,* 10 – 13 May 2009, Gothenburg, Sweden, viewed 25 July 2014, http://hdl.handle.net/10204/3631

McFerren, G, Swanepoel, D & Lai, C 2013, Wide Area Alerting System for Wildfires & Other Nasties, *E-learning for the Open Geospatial Community Conference Series,* FOSS4G 2013, 17-21 September 2013, Nottingham, United Kingdom

Meng, X, Bian, F & Xie, Y 2009, Geospatial Services Chaining with Web Processing Service, *Proceedings of the International Symposium on Intelligent Information Systems and Applications,* 28-30 October 2009, Qingdao, China, pp. 7-10

Mukherjee, J, Wang, M & Krishnamurthy D 2014, Performance Testing Web Applications on the Cloud, *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (ICTSTW)*, 31 March - 4 April 2014, Cleveland, Ohio, pp. 363-369, doi: 10.1109/ICSTW.2014.57

Over, M, Schilling, A, Neubauer, S & Zipf, A 2010, Generating web-based 3D City Models from OpenStreetMap: The current situation in Germany, *Computers, Environment and Urban Systems,* vol. *34* (6), pp. 496-507

Petcu, D, Neagul, M, Frincu, M, Zaharie, D & Panica, S 2010, Earth Observation Data Processing in Distributed Systems,*Informatica,* vol. 34, pp. 463-476

Plaisant, C 2004, The Challenge of information visualization evaluation, *AVI'04 Proceedings of the working conference on advanced visual interfaces, 2*5-28 May 2004, SIGCHI, Italy, pp. 109-116

Prandi, F, Soave, F, Devigli, M, Andreolli, R & De Amicis, R 2014, Services oriented smart city platform based on 3D city model visualization, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information SciencesVol.2,* ISPRS

Technical Commission IV Symposium, 14 – 16 May 2014, Suzhou, China

Resch, B, Wohlfahrt, R & Wosniok, C 2014, Web-based 4D visualization of marine geo-data using WebGL, *Cartography and Geographic Information Science,* vol. 41(3), pp. 235-247

Samadzadegan, F, Saber, M, Zahmatkesha, H, & Khanlou, HJG 2013, An Architecture for Automated Fire Detection Early Warning System Based on Geoprocessing Service Composition, *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. *1*(3), pp. 351-355

Shao, Y, Di, L, Bai, Y, Guo, B & Gong, J 2012, Geoprocessing on the Amazon cloud computing platform—AWS. *Agro-Geoinformatics,The First International Conference on Agro-Geoinformatics*, 2-4 August 2012, Shanghai, pp. 1-6, doi: 10.1109/Agro-Geoinformatics.2012.6311655

Shekhar, S, Gunturi, G, Evans, MR, & Yang, KS 2012, Spatial Big-Data Challenges Intersecting Mobility and Cloud Computing. *Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*, 20 May 2012, Arizona, USA, pp. 1-6, doi: 10.1145/2258056.2258058

Slocum, TA, Blok, C, Jiang, N, Koussoulakon, A, Montello, DR, Fuhrmaann, Hedley, NR 2001, Cognitive and Usability Issues in Geovisualization, *Cartography and Geographic Information Science*, vol. 28 (1), pp. 61-75, DOI: 10.1559/152304001782173998

Sun, Z,& Yue, P 2010, The use of Web 2.0 and geoprocessing services to support geoscientific workflows, *18th International Conference on Geoinformatics,* Beijing*,* pp. 1-5, doi: 10.1109/GEOINFORMATICS.2010.5567702

Tiede, D, & Lang, S 2010, Analytical 3D views and virtual globes—scientific results in a familiar spatial context, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol.*65*(3), pp. 300-307

Vaquero, LM, Rodero-Merino, L, Caceres, J, Linder, M 2008. A break in the clouds: towards a cloud definition, *ACM SIGCOM Computer Communication Review*, vol. 39(1), pp. 50-55

Vieweg, S, Hughes, AL, Starbird, K & Palen, L 2010, Microblogging during two natural hazard events: What Twitter may contribute to situational awareness, *CHI 2010: Crisis Informatics*. 10-15 April 2014, Atlanta, Georgia, USA

Westerholt, R, & Resch, B 2014, Asynchronous Geospatial Processing: An Even Driven Push Based Architecture for the OGC Web Processing Service, *Transactions in GIS*, doi: 10.1111/tgis.12104

Wu, H, He X, Gong, J 2010, A virtual globe-based 3D visualization and interactive framework for public participation in urban planning processes, *Computers,* *Environment and Urban Systems*, vol.34, pp. 291-298

Yang, C, Xu, Y& Nebert, D 2013, Redefining the possibility of digital Earth and geosciences with spatial cloud computing. *International Journal of Digital Earth*, vol. *6*(4), pp. 297-312

Yun, S, Chen, C, Li, J & Tang, L 2011, Wildfire spread simulation and visualization in virtual environments. International Conference on *Spatial Data Mining and Geographical Knowledge Services (ICSDM),* Fuzhou,pp. 315-331

Online References

AGI 2013, Cesium - WebGL Virtual Globe and Map Engine, viewed 7 July 2013, http://cesiumjs.org

Amazon Web Services n.d., Amazon Simple Queue Service, viewed 17 February 2015, http://docs.aws.amazon.com/AWSSimpleQueueService/2008-01-01/SQSDeveloperGuide/index.html?Query_QueryErrors.html

Amazon Web Services n.d., *Amazon Web Services-Cloud Computing Services,* viewed 1 May 2014,http://aws.amazon.com

British Columbia Wildfire Management Branch 2014, *Wildfire Management Branch*, viewed 21 July 2014, http://bcwildfire.ca

Cepicky, J 2013, *Welcome to PyWPS \&amp;mdash; PyWPS,* viewed 21 November 2013, http://pywps.wald.intevation.org

CSIR 2012, *Advanced Fire Information System,* viewed 21 July 2014, www.afis.co.za

Department of Water Affairs and Forestry n.d., Veldfires in South Africa, viewed 3 November 2014, www.daff.gov.za/doaDev/sideMenu/ForestryWeb/webapp/Documents/ForestFire/192.168.10.11/nvffa.nsf/cba79e2e60cb841f2256d6eoo3942fa/64a7c7f6bea728c942256dff003121a502.ec.html?OpenDocument

Dumbill, E 2012, *What is big data: An introduction to the big data landscape,* viewed 1 March 2013*,* http://radar.oreilly.com/2012/01/what-is-big-data.html#variety

GDAL n.d., *OGR: Simple Feature Library,* viewed 21 November 2013, http://www.gdal.org/ogr/

Geoprocessing.info n.d., *Geoprocessing*.viewed 21 November 2013, http://geoprocessing.info/wpsdoc/

Lê-Quôc, A, Fiedler, M, Cabanilla, C 2013, The Top 5 AWS EC2 Performance Problems, viewed 27 October 2014, http://www.infoworld.com/article/2613784/cloud-computing/benchmarking-amazon-ec2--the-wacky-world-of-cloud-performance.html

MrDoob 2014, JavaScript 3D Library, viewed 1 June 2014, https://github.com/mrdoob/three.js

National Disaster Management n.d., Veld Fire Awareness, viewed 3 November 2014, www.iimp.co.za/Brochures/Veld_Fire_Awareness.pdf

OpenNebula 2013, *OpenNebula,* viewed 26 February 2013, http://opennebula.org/

OSGeo 2013, *GDAL/OGR Info Sheet,* viewed 21 November 2013, http://www.osgeo.org/gdal_ogr

OSGeo 2013, *PostGIS – Spatial and Geographic Objects for PostgreSQL,* viewed 21 November 2013, http://postgis.net/

OSGeo 2014, *GEOS - Geometry Engine, Open Source,* viewed 20 June 2013, http://trac.osgeo.org/geos/wiki.

Oxford University Press 2015, Definition of time-series, viewed 4 March 2014, http://www.oxforddictionaries.com/definition/english/time-series

Percivall, G 2013, *Big Processing of Geospatial data,* viewed 9 August 2014, http://www.opengeospatial.org/blog/1866

Pivotal Software, inc n.d., *RabbitMQ – Messaging that just works,* viewed 21 November 2013, http://www.rabbitmq.com/

Raffi 2013, New tweets per second record and how, viewed 3 March 2015, https://blog.twitter.com/2013/new-tweets-per-second-record-and-how

Roualt, E 2012, *Geo tips \&tricks: GDAL/OGR using Shapefile native .sbn spatial index,* viewed 18 November 2013, http://erouault.blogspot.com/2012/06/gdalogr-using-shapefile-native-sbn.html

Skytland, N 2012, *Big data: What is NASA doing with big data today?,* viewed 1 March 2013, http://open.nasa.gov/blog/2012/10/04/what-is-nasa-doing-with-big-data-today

Southern Cape Fire Protection Association n.d., Fire Safety-Some facts about fire and how to deal with it, viewed 3 November 2014, http://www.scfpa.co.za/index.php?comp=content&id=7

The Python Community 2012, *Rtree 0.7.0: Python Package Index*, viewed 21 November 2013, https://pypi.python.org/pypi/Rtree

The Python Community 2013, *Fiona 1.0.2: Python Package Index,* viewed 21 November 2013, https://pypi.python.org/pypi/Fiona

The Python Community 2013, *Open arbitrary resources by URL Python Package Index,* viewed 21 November 2013, http://docs.python.org/2/library/urllib.htm

The Python Community 2013, *Shapely 1.2.18: Python Package Index,* viewed 21 November 2013, https://pypi.python.org/pypi/Shapely

The Python Community 2014, Pika 0.9.14*: Python Package Index,* viewed 19 November 2014, https://pypi.python.org/pypi/pika

The Python Community 2014, pyproj 1.9.3: *Python Package Index*, viewed, 18 November

2014, https://pypi.python.org/pypi/pyproj

Wayner, P 2013, Benchmarking Amazon EC2: The wacky world of cloud performance, viewed 27 October 2014, http://www.datadoghq.com/wp-content/uploads/2013/07/top_5_aws_ec2_performance_problems_ebook.pdf

Wild, S 2013, Hot warning system for tracking fires, *Mail and Guardian*, viewed 26 September 2014, http://mg.co.za/article/2013-09-06-00-hot-warning-system-for-tracking fires

Working on Fire 2012, Fire in South Africa, viewed 3 November 2014, http://www.workingonfire.org/index.php/about-wof/fire-in-sa

Wild, S, 2013, Hot warning system for tracking fires, *Mail and Guardian*, viewed 26 September 2014, http://mg.co.za/article/2013-09-06-00-hot-warning-system-for-tracking fires

Working on Fire 2012, Fire in South Africa, viewed 3 November 2014, http://www.workingonfire.org/index.php/about-wof/fire-in-sa

Other References

Aiyagari, S, Arrott, M, Atwell, M, Brome, J, Conway, A, Godfrey, R, Greig, R, Hintjens, P, O'hara, J, Radestock, M, Richardson, A, Ritchie, M, Sadjadi, S, Schloming, R, Shaw, S, Sustrik, M, Trieloff, C, van der Riet, K & Vinoski, S 2008. *Advanced Message Queuing Protocol Specification,* Version 0.9.1, OASIS

Jensen, JR 2004, *Introductory Digital Image Processing*, 3rd edition. Prentice Hall

Karimi, HA 2014*, Big Data: Techniques and Technologies in Geoinformatics*. CRC Press

Kraak, MJ & Ormeling, F 2011, *Cartography: Visualization of Spatial Data,* 3rd edn. Pearson Education Limited, London

Kwan, MP, Lee, J, 2003, Geovisualization of Human Activity Patterns Using 3D GIS: A Time-Geographic Approach, *Spatially Integrated Social Science: Examples and Best Practice*, Oxford University Press

McCullough, AR 2011, Sensor Web Processing on the Grid, Phd Thesis, Newcastle University, England, Handle: http://hdl.handle.net/10443/1321

Mell, P & Grance, T 2011, The NIST Definition of Cloud Computing, Special Publication pp.800-146

Michaelis, CD & Ames, DP 2007, Evaluation of the OGC Web Processing Service for use in a client-side GIS, *OSGEO Journal,* vol. 1, pp. 1-8

Murillo, CAO 2011, Parallelization of Web Processing Services on Cloud Computing: A case study of Geostatistical Methods, Masters Dissertation, Universidade Nova de

Lisboa, Handle: http://hdl.handle.net/10362/8294

NASA Land Processes Distributed Active Archive Centre (LP DAAC) 2013, SRTM Version 3. USGS/Earth Resources Observation and Science (EROS) Centre, Sioux Falls, South Dakota

Open Geospatial Consortium (OGC) 2007, OpenGIS® Web processing Service Version 1.0.0. OGC 05-007r7, viewed 20 August 2014, http://www.opengeospatial.org/standards/wps

Russom, P 2011, Best Practices Report : 4th Quarter. *Big Data Analytics*

South African National Biodiversity Institute 2006, Vegetation Map 2006, viewed 20 August 2014, http://www.bgis.sanbi.org/vegmap/project.asp

South African National Biodiversity Institute 2009, National Land Cover 2009, viewed 20 August 2014, http://www.bgis.sanbi.org/land cover/project.asp

WorldPop, 2013, WorldPop Population Map, viewed 20 August 2013, http://www.worldpop.org.uk/data/summary/?contselect=Africa&countselect=South+Africa&typeselect=Population

Conferences Presented

Hankel, L 2013, Exploiting Cloud Computing and Web Processing Services for the Processing and Visualisation of Big Geospatial Data, *First Postgraduate Research Seminar by the Department of Geography*, Geoinformatics and Meteorology, University of Pretoria, 10-11 October 2013, Pretoria, South Africa

Hankel, L, McFerren, G, Coetzee, S 2014, Distributed Geoprocessing of Streaming Data for a 3D Context Aware Visualisation Solution of a Wildfire Scenario, *The 11th International Symposium on Location Based Services*, 26-28 November 2014, Vienna, Austria