



# **Performance Evaluation of Apache Cassandra using AWS (Amazon Web Services) and GCP (Google Cloud Platform)**

**Alluri Gayathri  
Thanuja**

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

**Contact Information:**

Author(s): Alluri Gayathri Thanuja

E-mail: [gaal17@student.bth.se](mailto:gaal17@student.bth.se)

P.No:960214-2508

**University advisor:** Mikael Svahnberg

Faculty of Computing Blekinge Institute of  
Technology SE-371 79 Karlskrona, Sweden

## ABSTRACT

**Context:** In the field of computer science and communication systems, cloud computing plays an important role in Information and Technology industry, it allows users to start from small and increase resources when there is a demand. AWS (Amazon Web Services) and GCP (Google cloud Platform) are two different cloud platform providers. Many organizations are still relying on structured databases like MySQL etc. Structured databases cannot handle huge requests and data efficiently when number of requests and data increase. To overcome this problem, the organizations shift to NoSQL unstructured databases like Apache cassandra, Mongo DB etc.

**Objectives:** In this research, we started with studying of cassandra, AWS and GCP. Experiments were conducted to evaluate cassandra's cluster on different configurations of EC2 (Elastic Compute Cloud) and GCP instances. Throughput and latency were used to evaluate cassandra's performance.

**Methods:** Initially, a literature review has been carried out to acquire knowledge on cloud computing, AWS, GCP, apache cassandra and to design experiment with the mentioned problems. Next, the experiment has been carried out to evaluate the performance of apache cassandra using AWS and GCP. GCP instances will be created and, on each node (3) apache cassandra will be installed on all nodes and the similar fashion will be followed for AWS. Cassandra stress will be used for benchmarking cassandra using different combinations of read, mixed and write. These throughput and latency evaluate how the AWS and GCP are affected in the experimental environment.

**Results:** Performance of apache cassandra has been performed on both the clouds (AWS and GCP) in terms of latency and throughput using cassandra stress tool. In both clouds the performance of cassandra has increased significantly from thread count 50 to thread count 600 in terms of latency and throughput.

**Conclusions:** From the literature review, I have gained knowledge regarding the cloud computing, problems existed in cloud, which leads to setup this research in evaluating the performance of cassandra on AWS and GCP. The conclusion from the experiment is that as the thread count increases throughput and latency has increased gradually till thread count 600 in both the clouds. By comparing both the clouds throughput values, AWS scales up compare to GCP. GCP scales up, when compared to AWS in terms of latency.

**Keywords:** Apache Cassandra, AWS, Google Cloud Platform, Cassandra Stress, Throughput, Latency.

# Acknowledgement

I would like to thank my professor Mikael Svahnberg for guiding and supporting me throughout this research. He was always available to answer my questions that I had while performing this research.

I would also thank my seniors Manjunath Reddy, Neeraj Reddy and Shravani Nelapudi for their unconditional support and helped me in providing guidance with the technical knowledge during this research.

My special thanks to Manjunath Reddy, Harsha and Akki Reddy who is always been like my backbone when I'm low and made my days in Sweden memorable.

I am thankful to my parents for their emotional support and guidance in completing my master's degree.

# CONTENTS

- ABSTRACT-----III
- ACKNOWLEDGEMENT-----IV
- CONTENTS-----V-VI
- 1. INTRODUCTION-----1
  - 1.1 MOTIVATION-----1
  - 1.2 PROBLEM IDENTIFICATION -----1-2
  - 1.3 CONTRIBUTION-----2
  - 1.4 AIM AND OBJECTIVES-----2
  - 1.5 RESEARCH QUESTIONS-----2
  - 1.6 THESIS OUTLINE-----2
- 2. BACKGROUND-----3
  - 2.1 CLOUD COMPUTING-----3
  - 2.2 APACHE CASSANDRA-----3
    - 2.2.1 Working of Apache Cassandra-----3
    - 2.2.2 Architecture of Cassandra-----3-4
  - 2.3 AMAZON WEB SERVICES-----4
  - 2.4 GOOGLE CLOUD PLATFORM-----5
  - 2.5 CASSANDRA STRESS TOOL -----5
- 3. RELATED WORK-----6
  - 3.1 RESEARCH GAP-----6
- 4. METHODOLOGY -----7
  - 4.1 LITERATURE REVIEW-----7-8
  - 4.2 EXPERIMENT-----8
  - 4.3 EXPERIMENTAL SETUP FOR AWS AND GCP-----8
    - 4.3.1 Environment for AWS and GCP -----8
    - 4.3.2 Setup for AWS and GCP instances-----9
  - 4.4 CLUSTER FORMATION FOR AWS-----9-10
  - 4.5 CLUSTER FORMATION FOR GCP-----10
  - 4.6 CASSANDRA STRESS TOOL-----10
  - 4.7 EXPERIMENT DESIGN-----11
  - 4.8 METRICS-----11
- 5. RESULTS-----12
  - 5.1 EXPERIMENTAL RESULTS FOR THREE-NODE CLUSTER-----12
    - 5.1.1 Avg throughput values of both AWS and GCP for read operation-----12

5.1.2	Avg throughput values of both AWS and GCP for write operation-----	13
5.1.3	Avg throughput values of both AWS and GCP for mixed operation-----	13-14
5.2	LATENCY VALUES OF AWS AND GCP FOR THREE-NODE CLUSTER-----	14
5.2.1	Avg latency values of both AWS and GCP for read operation-----	14
5.2.2	Avg latency values of both AWS and GCP for write operation-----	15
5.2.3	Avg latency values of both AWS and GCP for mixed operation-----	15
5.3	EXPERIMENTAL RESULTS FOR FIVE-NODE CLUSTER-----	16
5.3.1	Avg throughput values of both AWS and GCP for read operation-----	16
5.3.2	Avg throughput values of both AWS and GCP for write operation-----	16-17
5.3.3	Avg throughput values of both AWS and GCP for mixed operation-----	17
5.4	LATENCY VALUES OF AWS AND GCP FOR FIVE-NODE CLUSTER-----	17-18
5.4.1	Avg latency values of both AWS and GCP for read operation-----	18
5.4.2	Avg latency values of both AWS and GCP for write operation-----	18-19
5.4.3	Avg latency values of both AWS and GCP for mixed operation-----	19
6.	ANALYSIS-----	20
6.1	THROUGHPUT -----	20-23
6.2	LATENCY-----	23-25
7.	DISCUSSION-----	26
7.1	ANSWERS TO RESEARCH QUESTIONS-----	26-28
7.2	VALIDITY THREATS-----	28
8.	CONCLUSION AND FUTURE WORK-----	29
9.	REFERENCES-----	30-31

## List of Tables:

table1: describes the instances, memory, OS, nodes used in the research for AWS.

table2: describes the instances, memory, OS, nodes used in the research for GCP.

table3: three-node read operation throughput values of mean, S.D. of AWS and GCP.

table4: three-node write operation throughput values of mean, S.D. of AWS and GCP.

table5: three-node mixed operation throughput values of mean, S.D. of AWS and GCP.

table6: five-node read operation Throughput values of mean, S.D. of AWS and GCP.

table7: five-node write operation Throughput values of mean, S.D. of AWS and GCP.

table8: five-node mixed operation Throughput values of mean, S.D. of AWS and GCP.

table9: three-node read operation Latency mean values of AWS and GCP.

table10: three-node write operation Latency mean values of AWS and GCP.

table11: three-node mixed operation Latency mean values of AWS and GCP.

table12: five-node read operation Latency mean values of AWS and GCP.

table13: five-node write operation Latency mean values of AWS and GCP.

table14: five-node mixed operation Latency mean values of AWS and GCP.

**ABBREVIATIONS:**

AWS: Amazon web services

GCP: Google Cloud Platform.

EC2: Elastic Compute Cloud.

SLA: Service Level Agreement.

NoSQL: Not Only Structured Query Language.

SQL: Structured Query Language.

IP: Internet Protocol.

VPC: Virtual Private Cloud.

SSTable: Sorted Strings Table.

CPU: Central Processing Unit.

IaaS: Infrastructure as a Service.

PaaS: Platform as a Service.

SaaS: Software as a Service.

YCSB: Yahoo Cloud Service Benchmarking.

YAML: Yet Another Markup Language.

SSH: Secure Shell.

I/O: Input or Output.

vCPU: Virtual CPU.

VM: Virtual Machine.

## **LIST OF FIGURES:**

Figure 1: architecture of cassandra

Figure 2: Average throughput values of both AWS and GCP for thread count 50 to 600.

Figure 3: Average throughput values of both AWS and GCP for thread count 50 to 600.

Figure 4: Average throughput values of both AWS and GCP for thread count 50 to 600.

Figure 5: Average latency values of both AWS and GCP for thread count 50 to 600.

Figure 6: Average latency values of both AWS and GCP for thread count 50 to 600.

Figure 7: Average latency values of both AWS and GCP for thread count 50 to 600.

Figure 8: Average throughput values of both AWS and GCP for thread count 50 to 600.

Figure 9: Average throughput values of both AWS and GCP for thread count 50 to 600.

Figure 10: Average throughput values of both AWS and GCP for thread count 50 to 600.

Figure 11: Average latency values of both AWS and GCP for thread count 50 to 600.

Figure 12: Average latency values of both AWS and GCP for thread count 50 to 600.

Figure 13: Average latency values of both AWS and GCP for thread count 50 to 600.

# 1 Introduction

Cloud computing is a service over the internet, which eliminates the requirement of maintaining the hardware and software at the individual level [1]. Now-a-days, many of the organizations started collecting data from stakeholders, for future scrutiny. The generally used terms for defining their product are Software as a Service, Platform as a Service and Infrastructure as a Service.

The main goal behind the migration of desktop computing to cloud computing is to use virtually formed clusters at available data centers [2]. In the past, organizations depend on the traditional databases to analyze the data collected from the stakeholder's feedback for their service [3]. Recently, most of the organizations are shifted to non-relational databases i.e. NoSQL databases for handling unstructured data such as multimedia, social media and so on.

There are several cloud providers available on the market, where they differ in many ways like openness, complexity and performance. Complexity and openness are easy to identify while performance is hard to differentiate between the cloud providers [4]. Performance involves variety of domains like storage, network and computing. The major issues companies faces are to find suitable cloud platform in terms of performance [5]. The process of making right decision differ on application the companies uses [4]. However, few questions need to be answered like how to identify suitable virtual machines over other cloud providers.

The prime focus of this research is to identify suitable cloud provider over other cloud providers by evaluating the performance of cassandra, a NoSQL database running on AWS and GCP cloud platforms in a virtualized environment. Primarily, this research is aimed in measuring throughput and latency by adding the cassandra nodes on both AWS and GCP cloud services. Literature review and experimentation are used as a research method to collect the results. Cassandra cluster is stressed via the tools mentioned in this document, through which we have measured the throughput and latency of the cassandra cluster. Based on the results gathered, latency and throughput are measured and thus the performance of cassandra has been evaluated.

## 1.1 Motivation:

Now-a-days, most of the multi-national companies have tendency towards cloud storage than the traditional physical storage. Due to this tendency from traditional to cloud storage, it led to the development of various cloud service providers. Various SLA's are used for ensuring a secure service to the users for the service they requested. In this current research, two public clouds were chosen for evaluating the performance of two different clouds by conducting various experiments and determine a better cloud service provider when Apache cassandra is used as a database. It is important to assess and determine a better service by analyzing the performance of the cloud services.

## 1.2 Problem Identification:

In today's world, cloud computing plays an important role in large scale data analysis because of its scalability and reliability [10]. In a cloud environment, AWS and GCP are set of cloud service providers which are currently used by a huge number of companies [11] [12]. AWS and GCP provides highly reliable and scalable framework to deploy web-scale solutions with minimal support and administrative costs [12]. With the tremendous growth in information and communication technology, the storage type, functionalities and interaction with databases have improved [13].

Currently there is a lot of research done on NoSQL databases, by comparing other NoSQL databases like Hadoop based on their performance [30]. There is a little or no research is available by comparing the performance of cassandra on AWS with GCP. This research was performed to test if the cassandra can manage high load and how the cluster size effects the execution time in both AWS and GCP virtual environment. This led to the research gap for quantifying latency and throughput by deploying apache cassandra on GCP and AWS. Though, the previous studies have provided partial results to the study being addressed currently but this research answers basic issues of how to evaluate the performance of cassandra on GCP? How to calculate metrics like throughput, latency on GCP over AWS? On which cloud platforms (AWS and GCP) the

performance of cassandra is better?

### **1.3 Contribution:**

The main contribution of this research is to evaluate the performance of cassandra running on AWS and GCP cloud platforms and the effects of throughput and latency. It is observed from the results that AWS has higher throughput compare to GCP and vice versa in terms of latency.

### **1.4 Aim and Objectives:**

The aim of this research is to evaluate the performance of cassandra on AWS and GCP by measuring latency and throughput on both platforms.

The following are the objectives to achieve the aim.

- Studying and researching about GCP, apache cassandra, cassandra stress tool and AWS.
- Evaluating the performance of cassandra using cassandra stress benchmarking tool on AWS and GCP for measuring throughput and latency with same workload.
- Analyzed the results of cassandra in both the cloud services to check whether they are invariant with similar workload.
- Contributed to the state of the art in cloud computing.

### **1.5 Research Questions:**

The following are the research questions to achieve the aim.

**RQ 1)** How to measure throughput on Apache Cassandra when AWS and GCP virtual environments are considered? Also, are the results being invariant with each other?

Motivation: The main reason for choosing this research question is to evaluate the performance of apache cassandra using the factors throughput, in AWS and GCP environment with similar workloads.

**RQ 2)** How to measure latency on Apache Cassandra when AWS and GCP virtual environments are considered? Also, are the results being invariant with each other?

Motivation: The main reason for choosing this research question is to evaluate the performance of cassandra using the metrics latency in both AWS and GCP cloud environments.

### **1.6 Thesis Outline:**

Section 1 gives the brief introduction to the research study, research gap, aim and objectives and research questions. Section 2 describes the background of all the technologies used like cloud computing, apache cassandra, AWS environment, GCP environment and cassandra stress tool. Section 3 gives the idea of earlier research studies in the selected research area. Section 4 describes the methodology used for this research i.e. literature review and experiment methods. Section 5 describes the results. Section 6 describes the analysis. Section 7 describes the discussion and answers to the research questions and validity threats. Section 8 gives the conclusion of the research and possible areas for the future work. Section 9 gives the references.

## 2 Background

This section describes the key concepts of cloud computing, cassandra, AWS, GCP and cassandra stress tool. These concepts were used to conduct this research.

### 2.1 Computing:

In today's scenario, the cloud usage has been increasing due to its availability and flexibility of the data to the user across the world [14]. Several countries and enterprises have made huge investments in developing the cloud platforms. The data stored in the cloud can be of any format like videos pictures and text files. According to NIST, cloud computing is defined as "a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"[15].

### 2.2 Apache Cassandra:

Apache cassandra is an open-source NoSQL database, for which it is optimal for managing huge amounts of semi-structured, structured, and unstructured data across different locations [16]. A NoSQL database is non-relational database that handles huge amounts of data unlike a traditional database management system [17]. Cassandra is developed to handle large amount of data that is spread across multiple servers. Cassandra is a data structure that is highly efficient with high volume write operations which is based on log-structured merge-tree [18]. Cassandra has been designed in such a way that it can run with austere resources or on cheap commodity hardware and handle large volumes of data without the loss of efficiency [19].

Motivation behind choosing Cassandra over other relational databases is its ability to handle large amounts of unstructured data and capability for replication. "managed cassandra service providers are emerging to hide the complexity of its installation, fine tuning and operation of cassandra virtual data centers" [20]. Cassandra is an open source, uses peer-to-peer architecture rather than master-slave architecture. Cassandra features High availability, fault tolerance, elastic scalability, column-orientation [21]. Due to above mentioned reasons, we have chosen apache cassandra rather than other NoSQL databases. Large companies like netflix, instagram, face book, reddit, gitHub, eBay and many more companies are using cassandra database [22].

#### 2.2.1 Working of apache Cassandra:

Apache Cassandra is built in a way to handle thousands of users/operations simultaneously per second and can handle petabytes of data [23]. From the documentation of cassandra various aspects of working are explained [24]. Replication: Cassandra provides replication by creating the reductant copies of data across all the nodes, which is a built-in feature of cassandra. In any case, if one node fails then it automatically backups that data into another node in the same cluster. Data distribution: Cassandra distributes its data across all the nodes automatically without any intervention from the user and the data is distributed across all the nodes in the cluster.

#### 2.2.2 Architecture of Cassandra:

**Node:** it is the place where data is stored in cassandra.

**Cluster:** A cluster is a main component that contains one or more data centers. Two clusters cannot communicate each other i.e. each cluster is unique from each other [25]. Any number of nodes can be added or deleted from the cluster.

**Commit log:** all the write operation in cassandra are written here.

**Mem-table:** it is a memory-resident data structure. The data will be written to the mem-table after the commit log.

**SSTable:** the data from the mem-table is flushed to SSTable when its contents reached the threshold value.

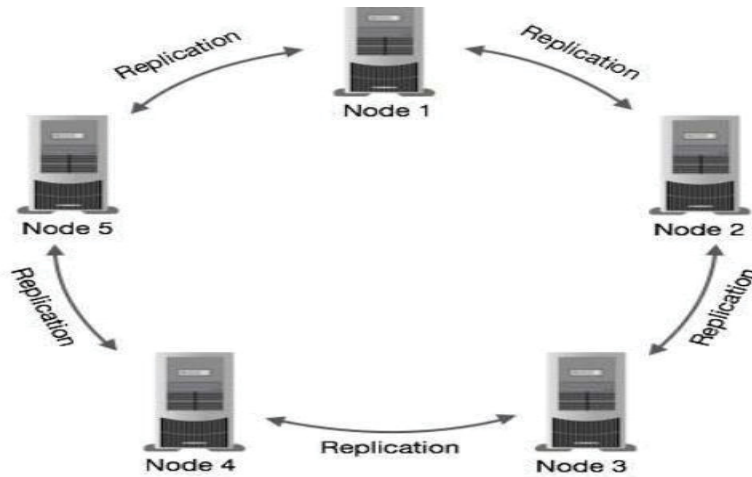


Figure 1: architecture of Cassandra [31]

### 2.3 Amazon Web Services:

Amazon web services are started in 2006, to offer IT services in the form of web services to the market i.e. now-a-days known as cloud computing. With the cloud users do not need to plan servers and other IT infrastructure. Instead, these services directly generate thousands of servers in minutes. AWS is cost efficient because we pay only for what we use with no additional charges. It provides several features such as host websites, store files, deploy a website, running database, running a virtual server, share digital media and analyzing user data. AWS helps user to create virtual machines and deploy any database management systems as it provides data centers in 12 different geographical locations [25].

Motivation behind choosing AWS is to access the data from different geographical locations, cloud computing allows the users to use the applications from any place in this world through internet connectivity [6]. In today's world, cloud computing technologies are provided by several providers which offers several cloud resources like containers, data storage resources, virtual machine instances, and computations [25]. At the same time, due to its efficient machine imaging where in the migration of data from the smaller instance to larger one effortlessly [26].

The following are the key parameters to create a virtual server.

- Virtual private cloud (VPC): VPC allows the users to use aws recourses in a virtual network. Creating subnets, selecting own IP address range, configuring network gateways and route tables can be customized by the users as they like in the virtual networking environment. If the account supports ec2-vpc platform only then it comes under default VPC. Users can create their own vpc and configure it as they need is known as non-default VPC. Internet gateway is included for default vpc and all the default subnets are public subnets. Each default subnet has a private and public IPv4 address when an instance is launched in it. Through internet gateway these instances can communicate with the internet.
- Security groups: To control inbound and outbound traffic of an instance a security group acts as a virtual firewall. The user can launch up to 5 security groups to the instance, when a user launches an instance in a vpc. The instance is automatically assigned to default security group for the vpc, when the user doesn't specify a group. User can add rules to each security group to control inbound traffic to instances and separate set of rules to handle outbound traffic. VPC has a default security group. The user can change the default security rules but cannot delete the default security group.

## **2.4 Google Cloud Platform (GCP):**

The GCP is a cloud computing platform which contains virtual resources, like virtual machines and set of physical assets, like hard disk drivers and computers that are stored in google datacenters within the globe [28]. The distribution of resources around the globe contains several advantages like reduced latency by locating resources closer to clients and redundancy in case of any failure [28]. The companies that use GCP services are Coca-Cola, pay pal, htc, Philips, best buy, domino's pizza, Sony music etc. GCP provides many products for developers to build programs from simple websites to complex distributed applications

Motivation behind choosing GCP over other virtual machines is its pricing, hybrid nature, databases, live migration of virtual machines and reductant back-up's. GCP provides wide range of customization to any instance. GCP is continued to provide open source, which means that its benefits users by providing well- supported and frictionless frameworks. Some of the open source technologies provided by GCP are Kubernetes, GO, tensorflow, Datastax, MongoDB etc.

## **2.5 Cassandra stress tool:**

Cassandra stress tool is a java-based benchmarking tool used for basic benchmarking and to test Cassandra cluster. Motivation behind choosing Cassandra Stress tool over YSCB is because it is provided in the default Cassandra package. Tables are created within a new key space using Cassandra stress tool. In this research we load the test and measure the metrics using theses tables and key spaces [29].It determines the capacity of production.

- How your database scales are easily understood by cassandra stress tool.
- It optimizes your data settings and models.
- It quickly determines how a schema works

For defining specific schemas with cache settings, various compaction strategies and types, cassandra stress supports yaml-based profile. YAML is used to test reads, writes and mixed workloads. User defined schemas, tables and key spaces are also supported by cassandra stress.

## 3 Related Work

This section of the paper provides existing available literature. Various literatures have been reviewed to figure out state-of-art benchmarking of NoSQL databases. Section 4.1 explains in detail how literature review was carried out. To obtain suitable literature both forward (finding citations of the paper) and backward (from the reference lists) snowballing and inclusion/exclusion criteria was carried. After analyzing and evaluating the literature, the following are the most related research papers obtained.

Authors in [4] evaluated the performance analysis of cloud computing architecture using different cloud services such as AWS, Azure, GCP and Open stack. This paper identified major research gap i.e. how to pick up the right cloud service provider and price for the end user. To achieve this author first collected the data; next analytics is performed and then represented the data in the form of graphs. The author concluded from his results that AWS scaled in terms of throughput compare to other cloud providers. While GCP scales up in terms of latency compare to other cloud providers.

Authors in [25] evaluated the performance of cassandra's virtual nodes with the configuration of three ec2 instances using ycsb benchmark to find the optimal and sub-optimal configuration of a data center such that the throughput requirements are satisfied. In this paper, author has used different workloads of YCSB. The author used cpu utilization and throughput (ops/sec) to measure the performance. The author concluded that the optimal and sub-optimal configuration of the datacenter running multiple instances of cassandra, that the throughput requirements are satisfied.

Authors in [32] measured the scalability and latency of apache cassandra in ec2 instances and how consistency and replication factor have affected the autoscaling of cassandra instances. In this research, author had used 10 million read, writes and mixed to measure the performance of cassandra. Then cassandra stress tool is used for bench marking. The author concluded that mean latency and cpu utilization are averaged separately and compared.

### 3.1 Research Gap

It's been a while since clouds appeared in the IT industry, and eventually clouds have spread around the world. There are many cloud platforms that differ in many ways, like in terms of openness, complexity, or performance etc. While the openness and complexity of the cloud are often easy to see but it is more difficult to differentiate based on their performance [4]. The first difficulty comes from the variety of areas that need to be taken care of such as latency (storage) and throughput (networking). It becomes more complex to fit exact application due to its needs because few need more storage while others require more network performance, etc. [5]. Most often, companies face problems in search of a suitable cloud platform which is not as systematic as it should be. They end up making the decision that they discovered first. But making a correct decision depends on the application [4]. There is a research done in evaluating the performance of cassandra using AWS in terms of cpu utilization [25]. But there is no little research done in evaluating the performance of cassandra using AWS and GCP in terms of latency and throughput.

In this research we have taken cloud platforms like GCP and AWS to evaluate their performance using Cassandra as a database. This research provides answers like how to identify suitable cloud platforms over the other cloud providers like GCP and AWS? On which platform (GCP, AWS) is the performance of Cassandra better in terms of throughput and latency? For answering RQ1, RQ2 experiment is conducted. For experimental setup of AWS and GCP, cluster formation of AWS and GCP, literature review is used.

## 4 Research Methodology

The well-suited methodology to answer the above research questions and to achieve the aim of this research is to perform the literature review and by conducting an experiment. The aim of this research is to evaluate the performance of cassandra on AWS and GCP instances. Cassandra stress tool is used for benchmarking. Initially, literature review identified different performance factors and helps in experimenting.

### 4.1 Literature Review:

For any research to be conducted, suitable literature study needs to be done to diagnose, identify and formulate an idea or a concept [26]. In an academic project, literature review plays an important role because it identifies the related work and research gap. In this research, literature review is carried out during initial stages to formulate the idea in this context. Literature review has been carried out to gain knowledge in three major topics used.

- Gaining knowledge about the Cassandra configurations, its dynamics, and the metrics to evaluate the performance of Cassandra.
- Gaining knowledge about the AWS console and its configurations.
- Gaining knowledge about Google Cloud Platform instances and its configurations.

Following are the inclusion (articles considered)/exclusion (articles not considered) criteria performed in this literature review:

- Whether the article is written in English?
- The full text available?
- Is the article related to Cassandra, AWS, and GCP?
- Is the article discussing the metrics like latency, throughput?
- Is the article published in 2005-2022?
- Articles written in other languages except English are not considered.
- Articles published before 2005 are not considered.
- Articles, whose full text is not available, are not considered.

To find more relevant literature, sampling both forward (finding citations to the paper) and backward (from the reference lists) is done. Further the same research articles have been extracted. This literature is conducted based on [26] in the following fashion:

#### 1. Search strings selection:

Several databases like Inspec, IEEE Explorer, Engineering Village, Scopus, Google Scholar with the suitable key words like 'Cassandra', 'AWS', 'GCP', 'Cassandra Stress', 'Cloud Computing', 'Cassandra evaluation', 'NoSQL database' are selected and searched. Several search strings like (Latency) OR (Throughput) AND Cassandra, Cassandra AND GCP OR (AWS), (AWS) OR (GCP) AND Cassandra, Cassandra AND Performance were selected based on the title to find more relevant literature for related work and for further documentation.

#### 1. Gathering and analyzing the literature:

The literature is gathered in the form of articles, journals, books using the databases like IEEE Explorer, Inspec, google scholar, and Scopus. Analysis on the gathered literature is performed by comparing, defining, and separating from the other less important gathered literature.

#### 2. Evaluation of Literature:

In this step, assessing or judging the gathered papers whether they are relevant by reading to analyze the data carefully not through key words.

#### 3. Writing a review:

In this step, the information is extracted from gathered papers in a well-structured piece of writing. This writing is well organized to make the reader understand the subject and context through thorough motivations.

#### 4. *Writing references:*

The conclusion of literature review is done with complete bibliographical list of articles, journals with their authors.

By performing literature review, I have reviewed more research papers and articles related to my research and identified a research gap for my thesis. Obtained knowledge From YouTube tutorials [9], whitepapers on AWS [16] and GCP [27], and Cassandra stress documentation of Datastax [38], about configurations of cassandra clusters and how to deploy them on AWS and GCP cloud platforms.

## 4.2 Experiment

Experiment is the method where it investigates few variables and the ways in which they are affected in experimental conditions [34]. This research is all about evaluation of cassandra in AWS and GCP instances using metrics like throughput and latency with different workloads. Experiments on AWS and GCP are carried out on 3 nodes, and 5 nodes clusters and repetition of experiment by changing configurations. Cassandra stress tool is used to find out throughput and latency on AWS and GCP, and then the results are compared to check which cloud provider is better in terms of latency and throughput.

- Independent Variables: Apache Cassandra, Ubuntu desktop, AWS, GCP, Cassandra stress tool
- Dependent Variables: throughput, Latency.

## 4.3 Experimental setup for AWS and GCP:

### 4.3.1 Environment for AWS and GCP:

AWS console is a simple web-based user interface which allows the users to have a quick view of the resources and created isolated networks can be modified by the users [36,37]. Different configurations of ec2 instances are provided by aws such as T2, M4, M3, C4, C3, X1, R4, R3, P2, G2, F1, I2 and D2 and each of these are subcategorized based on storage (GB), Memory (GB), and on vCPU's [37]. To conduct this experiment, we have used T2 micro, small instances to setup the environment.

The following table describes the instances, memory and operating system used in this research:

VM type	T2.Micro
Vcpu	2
Memory (GiB)	2
Operating system	Ubuntu 18.04 lts
No. of nodes in the cluster	3 node,5 node

table1: describes the instances, memory, os, nodes used in the research for AWS.

The following table describes the instances, memory and operating system in this research for GCP.

VM type	n1-standard-1
Vcpu	2
Memory (GiB)	2 GB
Operating system	Ubuntu 18.04 lts
No of nodes in the cluster	3 node, 5 node

table2: describes the instances, memory, os, nodes used in the research for GCP.

#### 4.3.2 Setup for AWS and GCP instances:

This section explains how to conduct experiment in the cloud architecture by aws servers. Setup knowledge for both AWS and GCP was obtained by conducting literature review. In this research, apache cassandra is deployed on aws cloud servers. The following are the requirements to setup experimental cloud-based environment in aws:

##### 1 Creation of Virtual Private Cloud (VPC):

To send and receive packets from the localhost in aws, vpc is configured using virtual private network. In this experiment, ec2 instances are launched in the europe-north region by assigning network tenancy as both public and private addresses [8]. To allow the flow of traffic, network inbound and outbound rules of aws are configured [8].

##### 2 Creating a Subnet:

User can create two types of subnets i.e. private and public subnets by each virtual private network. In this experiment, ec2 instances are launched in the aws default subnet having 4093 as IP address [8].

##### 3 Launching of EC2 instance:

In this research, ec2 instances are selected based on the experimental requirement. So, t2.small (2vCPU, 2GiB memory) instances are selected in this experiment. The above selected instances are deployed under same default subnet and security group.

In the similar fashion gcp instances are created for experimentation.

## 4.4 Cluster Formation for AWS

- Next, ssh each ec2 instance
- After ssh, install latest version of java on each instance
- Then, install apache cassandra on each node

After deploying the apache cassandra on aws, cassandra 3.11.4 is installed from the datastax community after updating ubuntu 18-04 packages [38].

- For formation of the cassandra cluster, few changes need to be made in the YAML configuration. For 3,5-node cluster, one node must be seed and other nodes act as work nodes.

Following are the changes made in seed node on both the clouds i.e. aws.

- Seeds: IP address of all the three nodes
- Listen\_address: IP address of the respective node.
- Rpc\_address: IP address of the respective node.
- Endpoint snitch: GossipingPropertyFileSnitch
- Auto\_bootstrap: false (as i start my cassandra with no data this change is required)

Following are the changes are made in the work nodes in aws.

- Seeds: IP address of all the three nodes
- Listen\_address: IP address of the respective node.
- Rpc\_address: IP address of the respective node.
- Endpoint snitch: GossipingPropertyFileSnitch
- Auto\_bootstrap: false (as i start my cassandra with no data this change is required)

After making these changes in YAML configuration, cassandra cluster is formed.

#### 4.5 Cluster Formation for GCP:

- After creating gcp instances then ssh each gcp instance.
- Then, install latest version of java on each gcp instance.
- Next, install apache cassandra on each node.
- For formation of cluster few changes are required in YAML configuration. For 3-5, node cluster one node must be seed node and the remaining nodes act as work nodes.

Following are the changes made in seed node:

- Seeds: IP address of all the three nodes
- Listen\_address: IP address of the respective node.
- Rpc\_address: IP address of the respective node.
- Endpoint snitch: GossipingPropertyFileSnitch
- Auto\_bootstrap: false (as i start my cassandra with no data this change is required)

Following are the changes are made in the work nodes.

- Seeds: IP address of all the three nodes
- Listen\_address: IP address of the respective node.
- Rpc\_address: IP address of the respective node.
- Endpoint snitch: GossipingPropertyFileSnitch
- Auto\_bootstrap: false (as i start my cassandra with no data this change is required)

After making these changes in YAML configuration, cassandra cluster is formed.

#### 4.6 Cassandra Stress Tool:

For testing a cassandra cluster and benchmarking, cassandra stress tool is used. cassandra stress tool gives better understanding of creating better data models and database performance. By using the cassandra stress tool, the number of threads required to get maximum performance is obtained.

Following are the formats of cassandra stress tool that were used in this experiment:

- \$ cassandra-stress read : this must be first populated by write requests and this command generates load by performing read operation.
- \$ cassandra-stress write : it generates multiple write requests against the cluster
- \$ cassandra-stress mixed : this should be first populated by write test and it used to perform both read and write operations in a specified ratio.
- \$ cassandra-stress command [options]

## 4.7 Experiment Design:

This experiment contains two parts. In the first part we evaluate the performance of cassandra in aws and in the second part we evaluate the performance of cassandra in gcp. In both the parts the metrics used for evaluation are latency, throughput. Following are the steps same for both the parts:

- Initially all the software required for this research are installed. Since the experiment is conducted in virtual environment so, the virtual machines are installed and running on ubuntu 18.04 lts. Cassandra is installed on aws servers and the changes required to create a three, five-node cluster are made in the main configuration file i.e. YAML file. Once the cluster is created, the experiment begins here.
- To start the experiment, the cassandra stress tool initially performs write operation from the node 4 to populate the database and later other operations like read and mixed in the following steps.
- Next, node 4 performs the other operations with the other in the cluster and the time-taken to perform an operation is around 15 minutes in an initially populated database. So, the time durations are considered as 15 minutes for the further experiment.
  1. Mixed loads for every read and write operations in the ratio of 1:1
  2. Next read load by applied by the stress tool
  3. Finally write load is applied.
- Cassandra-stress tool generates the statistics values which contains throughput, latency at the end of the load generation.
- Similarly, the same procedure is repeated for five node cluster in both AWS and GCP. And compared the both the results.

## 4.8 Metrics:

The following are the metrics measured during the experiment in both GCP and AWS:

Throughput: Total operation rate is known as throughput. Average number of operations made in a second is measured as throughput.

Latency: it is displayed after each stress tool run and it is measured in millisecond.

## 5 Results:

In this section, we discuss and analyze the results gathered from the experiment. The results from both AWS and GCP are compared and analyzed to draw the conclusions to answer the research questions. Cassandra Stress tool is used to find the maximum performance or operations per second by using write, read and mixed command. This write command is executed and specifies only to write many requests i.e. (10 million write). In the similar fashion read and mixed are executed with many requests i.e. 10 million reads and 10 million mixed. The thread count starts with smaller threads and increases until the performance is observed in the cluster and stops when there is no change in performance. The thread count usually ranges in 50,100,150,200,250,300,350,400,450,500,550,600.

### 5.1 Experimental results for three-node cluster:

The experiment of a three-node cluster has been carried out for the read, write and mixed operations in Cassandra stress tool. The values obtained are the following for throughput and latency. The experiment has been carried five times in ec2 and gcp instances. After conducting the experiments for 5 times, almost similar results are observed while performed the experiment for the 6th time and 7<sup>th</sup> time. So, I have chosen to repeat the experiments for five times. The thread count starts with 50 and increases till 600 and stopped i.e. no performance is observed after thread count 600.

The below values depict cumulative distribution curves that show the effect of throughput and latency on the system. Different plots for different Cassandra operations have been plotted and analysis has been carried out calculating the results of all iterations in the experiment. All the experiments have the same duration of 10 million (read, write and mixed) and have been subjected to the same workload of equal read, write and mixed.

#### 5.1.1 Avg Throughput values of both AWS and GCP for Read operation:

The following are the values obtained in read operation. The following graph depicts the average values of throughput in GCP and AWS of thread count 50 to 600 with respect to time. As the throughput values increases from thread count 50 to 600 and stopped in both the clouds. X-axis represents thread count and y-axis represents avg throughput in AWS and GCP. The unit of throughput is bps

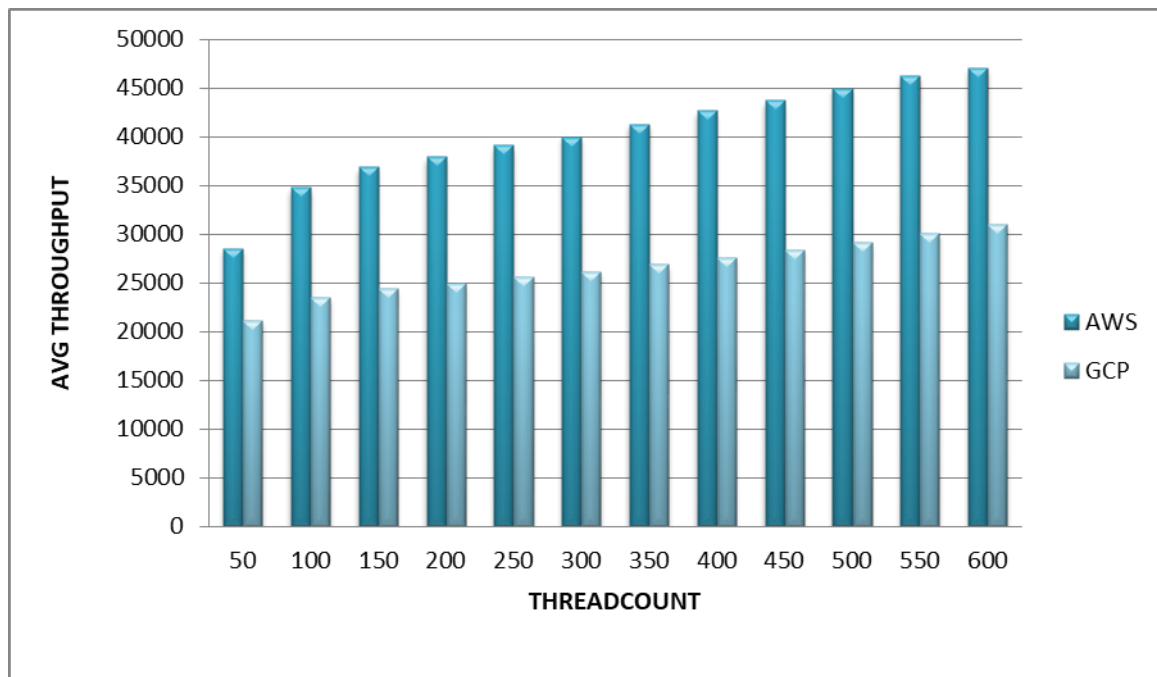


Figure 2: Average throughput values of both AWS and GCP for thread count 50 to 600.

### 5.1.2 Avg Throughput values of both AWS and GCP for write operation:

The following are the values obtained in write operation. The following graphs depict the average values of throughput in GCP and AWS of thread count 50 to 600 with respect to time. X-axis represents thread count and y-axis represents avg throughput in AWS and GCP. The unit of y-axis is bps

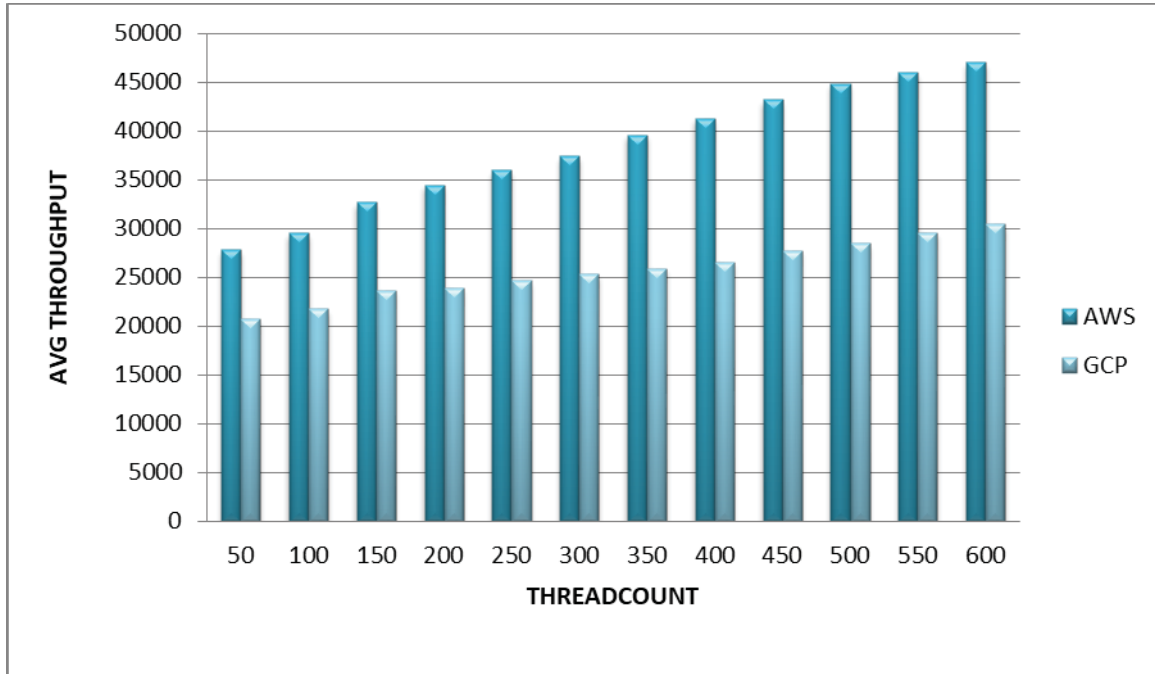


Figure 3: Average throughput values of both AWS and GCP for thread count 50 to 600.

### 5.1.3 Avg Throughput values of both AWS and GCP for mixed operation:

The following are the values obtained in mixed operation. The following graphs depict the average values of throughput in GCP and AWS of thread count 50 to 600 with respect to time. The throughput values increases from thread count 50 to 600 in both the clouds. X-axis represents thread count and y-axis represents avg throughput in AWS and GCP. The unit of y-axis is bps.

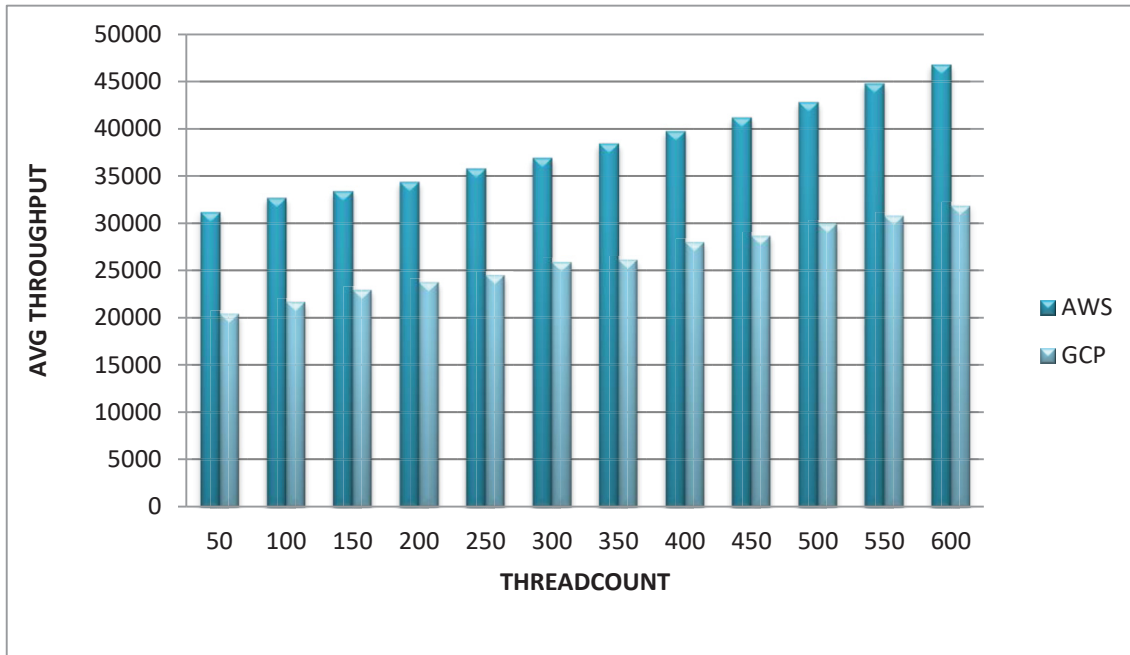


Figure 4: Average throughput values of both AWS and GCP for thread count 50 to 600.

## 5.2 Latency values of AWS and GCP in three-node cluster:

For the latency values it is observed that GCP has slight increase when compared to AWS. It is measured in milliseconds and unit of latency is ' $\mu s$ '.

### 5.2.1 Avg Latency values of both AWS and GCP for Read operation:

The following are the values obtained in read operation. The following graphs depict the average values of latency in a GCP and AWS of thread count 50 to 600 with respect to time. In the below graph purple color represents AWS and lavender color represents GCP. The unit y-axis is ' $\mu s$ '.

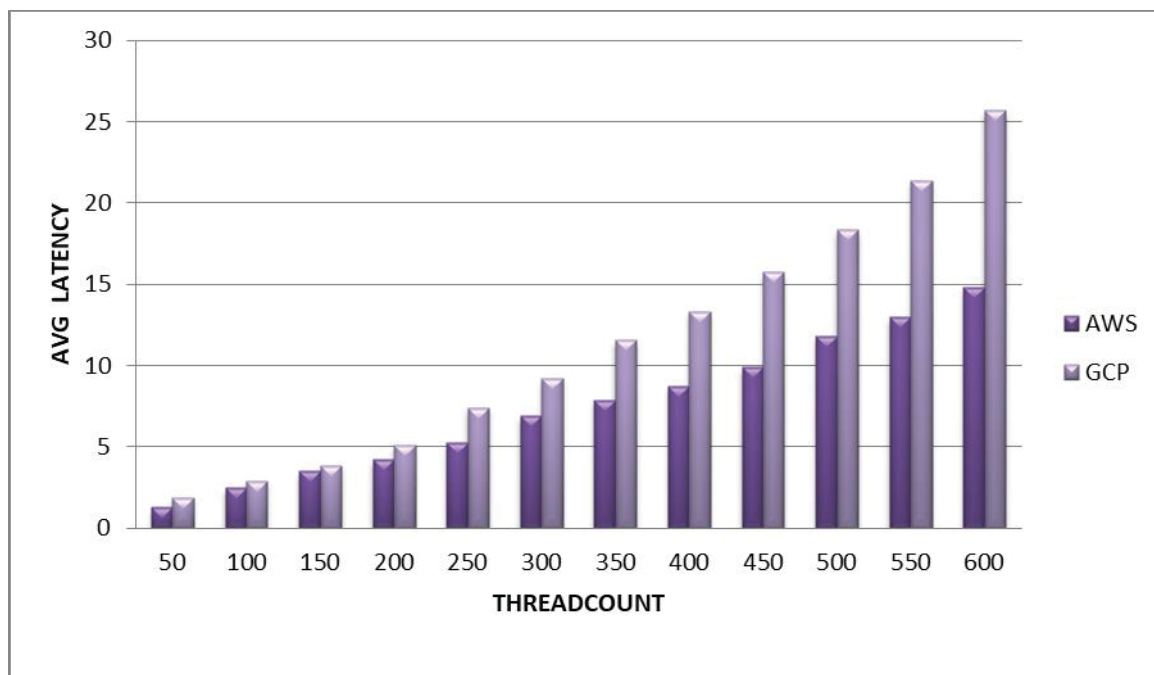


Figure 5 Average latency values of both GCP and AWS with thread count 50 to 600.

### 5.2.2 Avg Latency values of both AWS and GCP for write operation:

The following are the values obtained in write operation. The following graph depicts the average values of latency in a GCP and AWS of thread count 50 to 600 with respect to time. In the below graph purple color represents AWS and lavender color represents GCP. The unit of y-axis is ' $\mu$ s'.

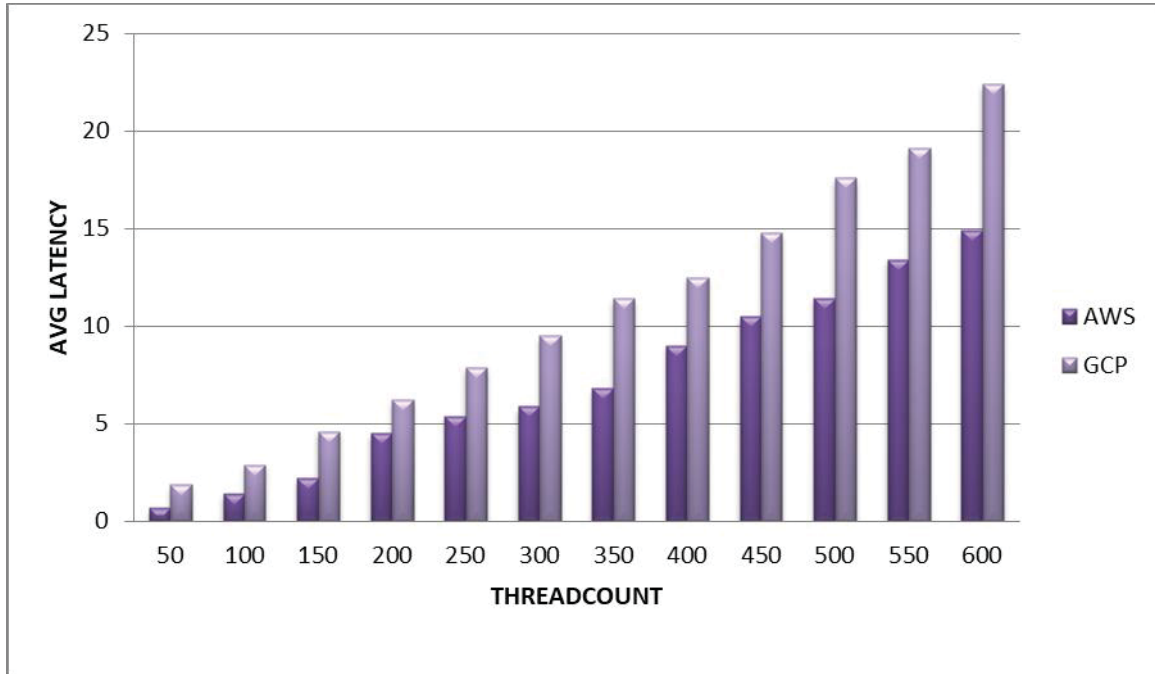


Figure 6 Average latency values of both GCP and AWS with thread count 50 to 600.

### 5.2.3 Avg Latency values of both AWS and GCP for mixed operation:

The following are the values obtained in mixed operation. The following graph depicts the average values of latency in a GCP and AWS of thread count 50 to 600 with respect to time. In the below graph purple color represents AWS and lavender color represents GCP. The unit of y-axis is ' $\mu$ s'.

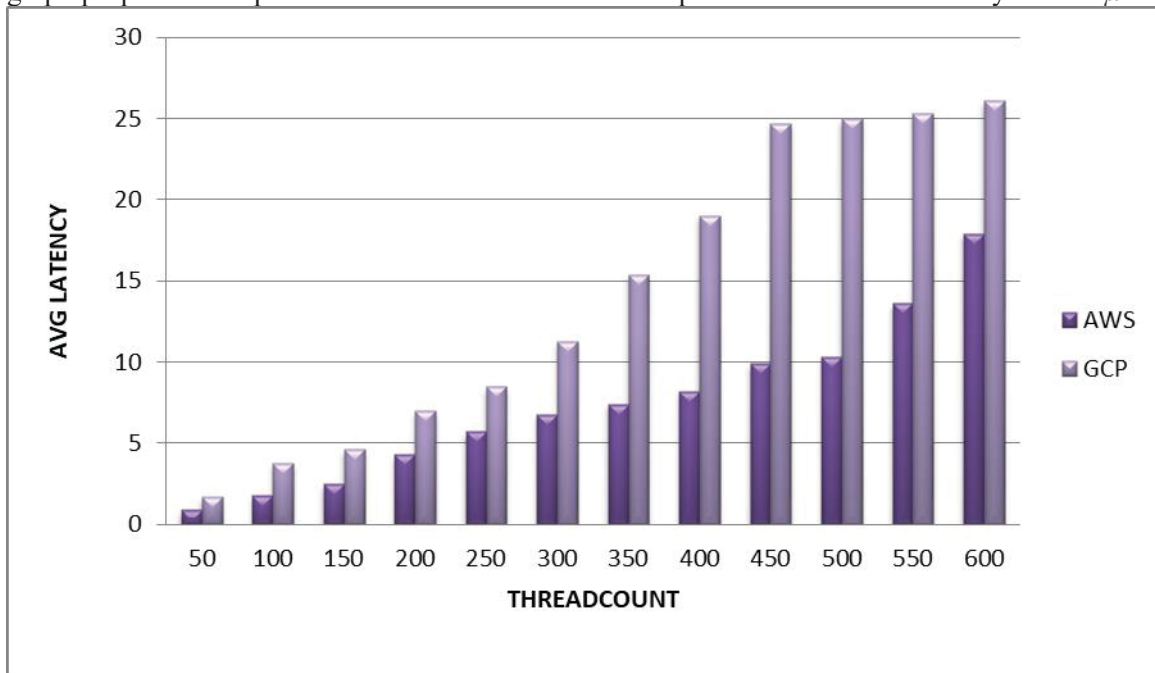


Figure 7 Average latency values of both GCP and AWS with thread count 50 to 600.

### 5.3 Experimental results for five-node cluster:

The experiment of a five-node cluster has been carried out for the read, write and mixed operations in cassandra stress tool. The values obtained are the following for throughput and latency. The experiment has been carried five times in ec2 and gcp instances. After conducting the experiments for 5 times, almost similar results are observed while performed the experiment for the 6th time and 7<sup>th</sup> time. So, I have chosen to repeat the experiments for five times. Graphs 8-10 depict the avg throughput values in both AWS and GCP. 11-13 depicts the avg latency.

#### 5.3.1 Avg Throughput values of both AWS and GCP for Read operation:

The following are the values obtained in read operation. The following graphs depict the average values of throughput in gcp and aws of thread count 50 to 600 with respect to time. The throughput values increases from thread count 50 to 600. X-axis represents thread count and Y-axis represents avg throughput in AWS and GCP. The unit of Y-axis is bps.

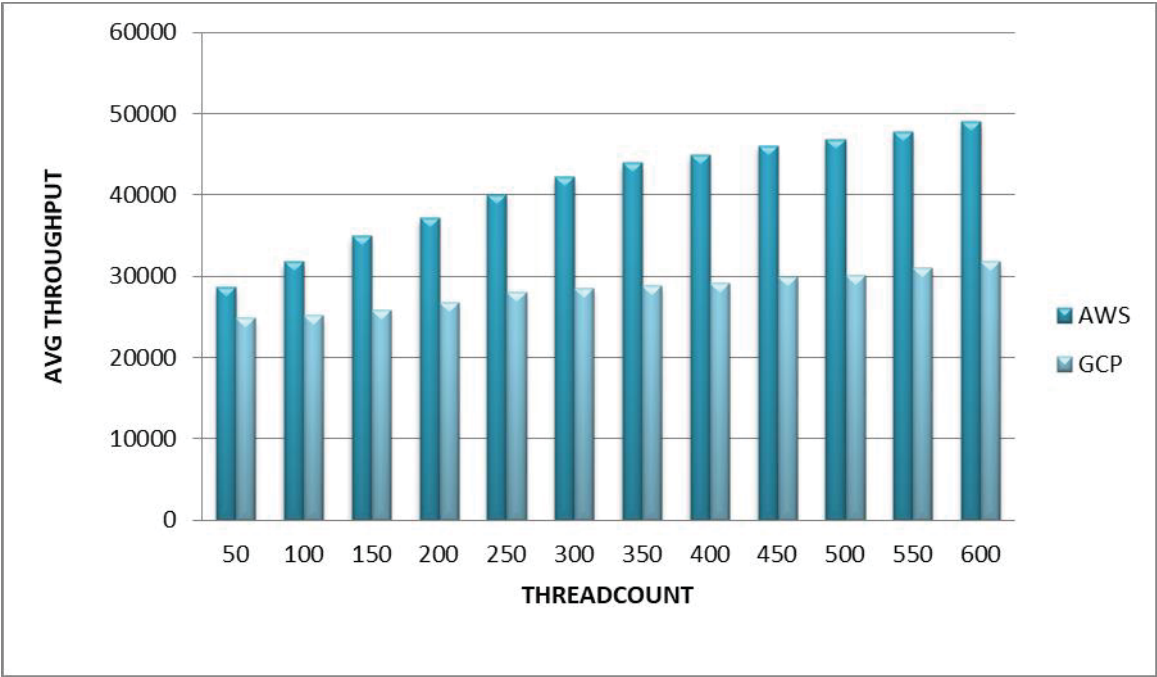


Figure8: Average throughput values of both GCP and AWS with thread count 50 to 600.

#### 5.3.2 Avg Throughput values of both AWS and GCP for write operation:

The following are the values obtained in write operation. The following graphs depict the average values of throughput in GCP and AWS of thread count 50 to 600 with respect to time. X-axis represents thread count and Y-axis represents avg throughput in AWS and GCP. The unit of Y-axis is bps.

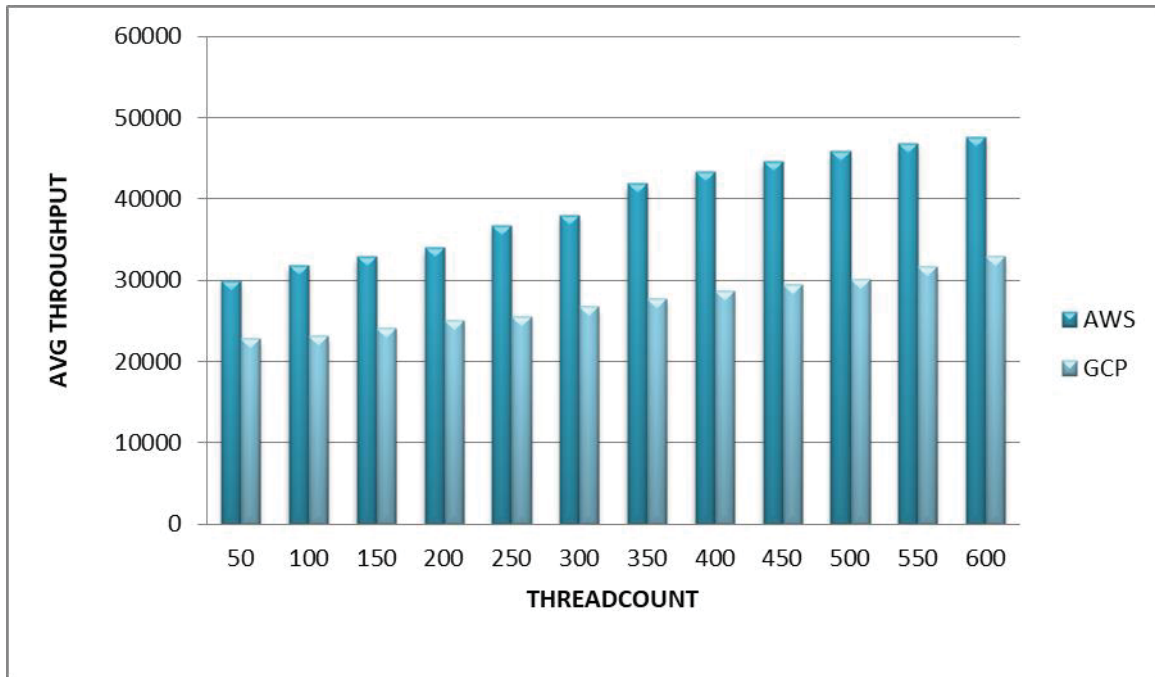


Figure 9 Average throughput values of both GCP and AWS with thread count 50 to 600.

### 5.3.3 Avg Throughput values of both AWS and GCP for mixed operation:

The following are the values obtained in mixed operation. The following graphs depict the average values of throughput in GCP and AWS of thread count 50 to 600 with respect to time. The throughput values increases from thread count 50 to 600 in both the clouds. X-axis represents thread count and Y-axis represents avg throughput in AWS and GCP. The unit of y-axis is bps.

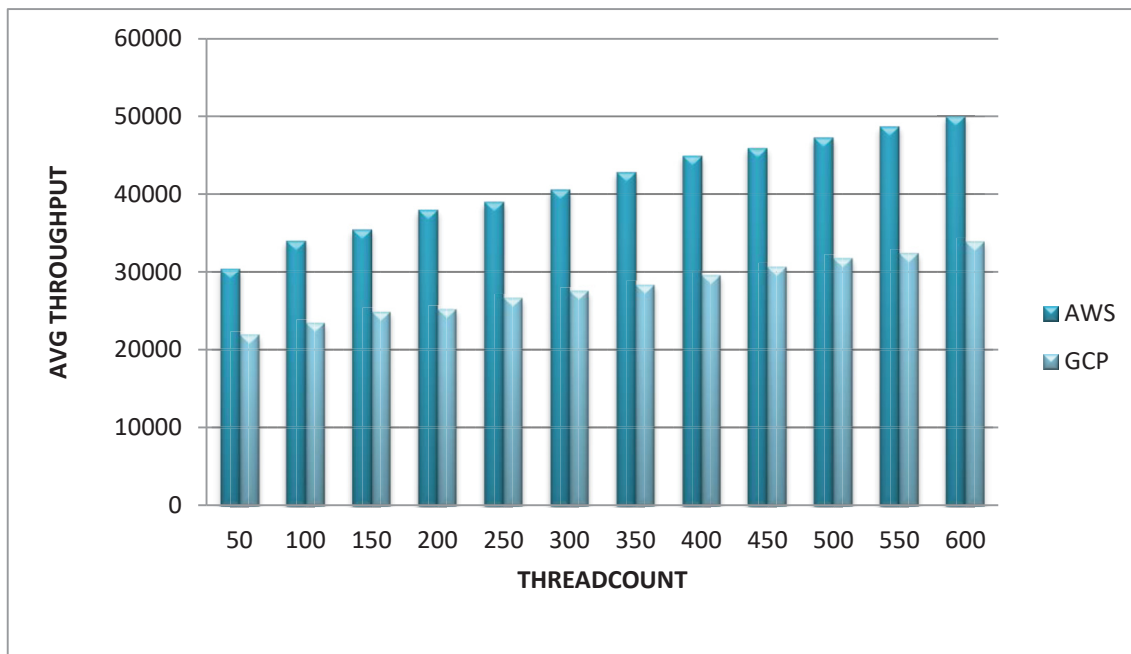


Figure 10 Average throughput values of both GCP and AWS with thread count 50 to 600.

### 5.4 Latency values of AWS and GCP for five-node cluster:

For the latency values it is observed that GCP has slight increase when compared to AWS. It is measured in milliseconds and unit of latency is  $\mu s$

From the above table displays the statistical values of latency in AWS and GCP clouds for mixed operation with thread count 50 to 600. For mixed operation, the thread count is started at 50 and stopped at 600 with 10 million requests. It increase from thread count 50 to 600 and stopped in both clouds. From the table it is observed that GCP as slight increase compared to AWS in terms of latency.

**5.4.1 Avg Latency values of both AWS and GCP for Read operation:**

The following are the values obtained in read operation. The following graph depicts the average values of latency in a GCP and AWS of thread count 50 to 600 with respect to time. For read operation, the thread count is started at 50 and stopped at 600 with 10 million requests. In the below graph purple color represents AWS and lavender color represents GCP. The unit of y-axis is ‘ $\mu s$ ’.

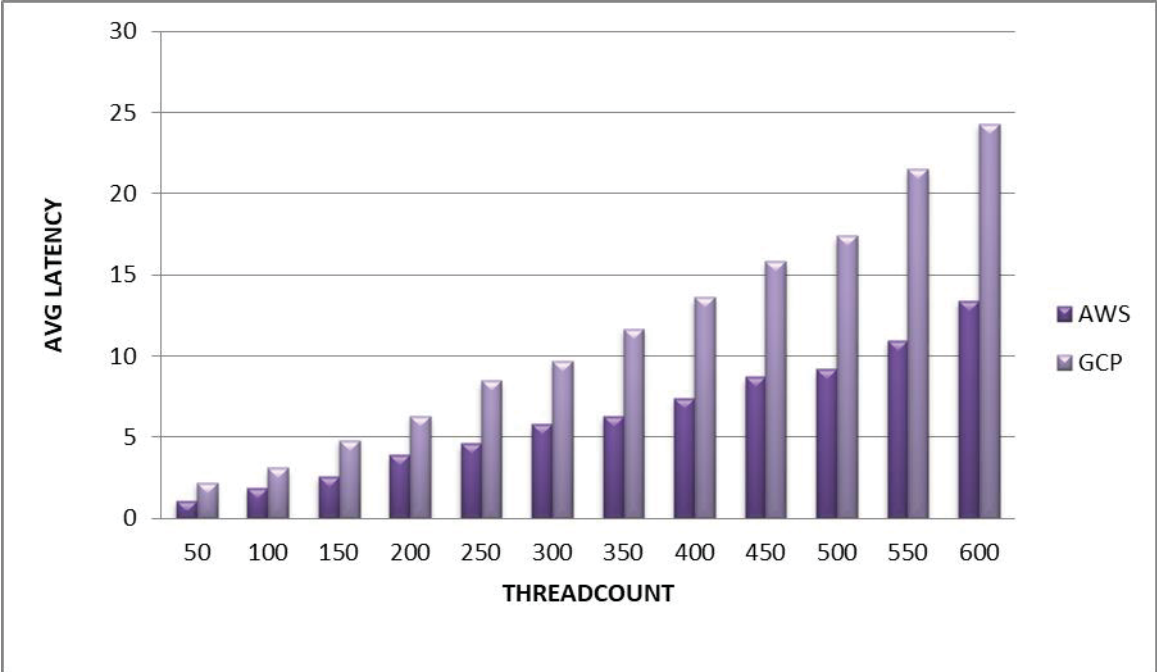


Figure 11 Average latency values of both GCP and AWS with thread count 50 to 600.

**5.4.2 Avg latency values of both AWS and GCP for write operation:**

The following are the values obtained in write operation. The following graphs depict the average values of latency in a GCP and AWS of thread count 50 to 600 with respect to time. In the below graph purple color represents AWS and lavender color represents GCP. The unit of y-axis is ‘ $\mu s$ ’.

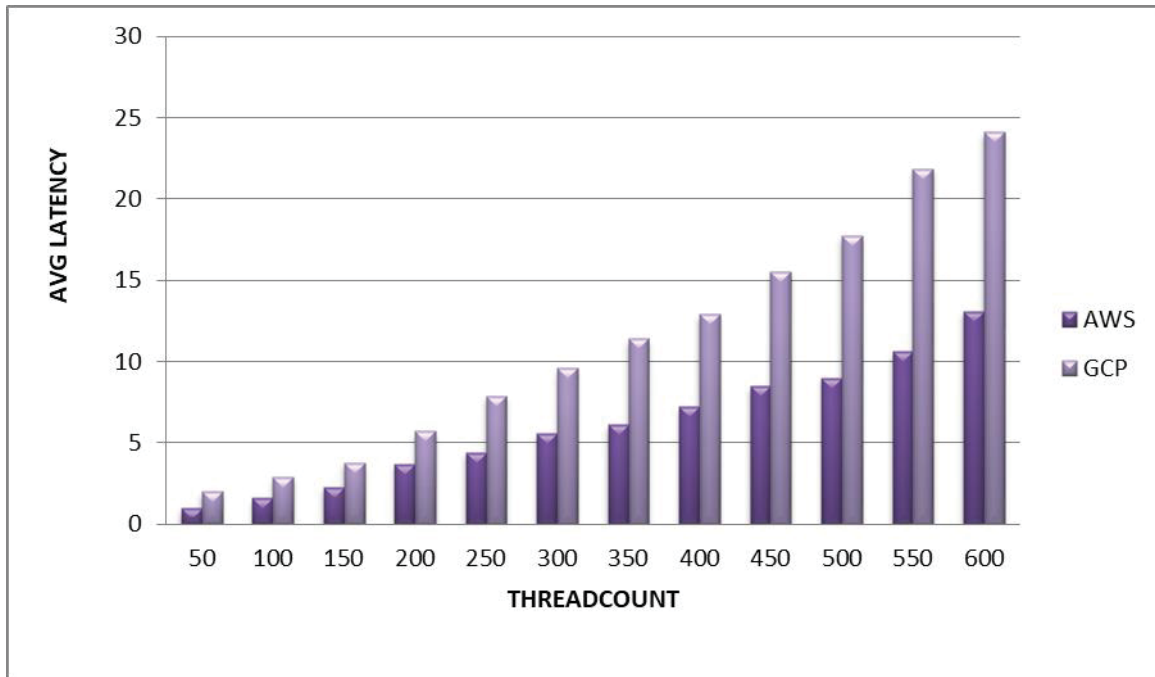


Figure 12 Average latency values of both GCP and AWS with thread count 5 to 600.

### 5.4.3 Avg latency values of both AWS and GCP for mixed operation:

The following are the values obtained in mixed operation. The following graphs depict the average values of latency in a GCP and AWS of thread count 50 to 600 with respect to time. In the below graph purple color represents AWS and lavender color represents GCP. The unit of y-axis is 'µs'.

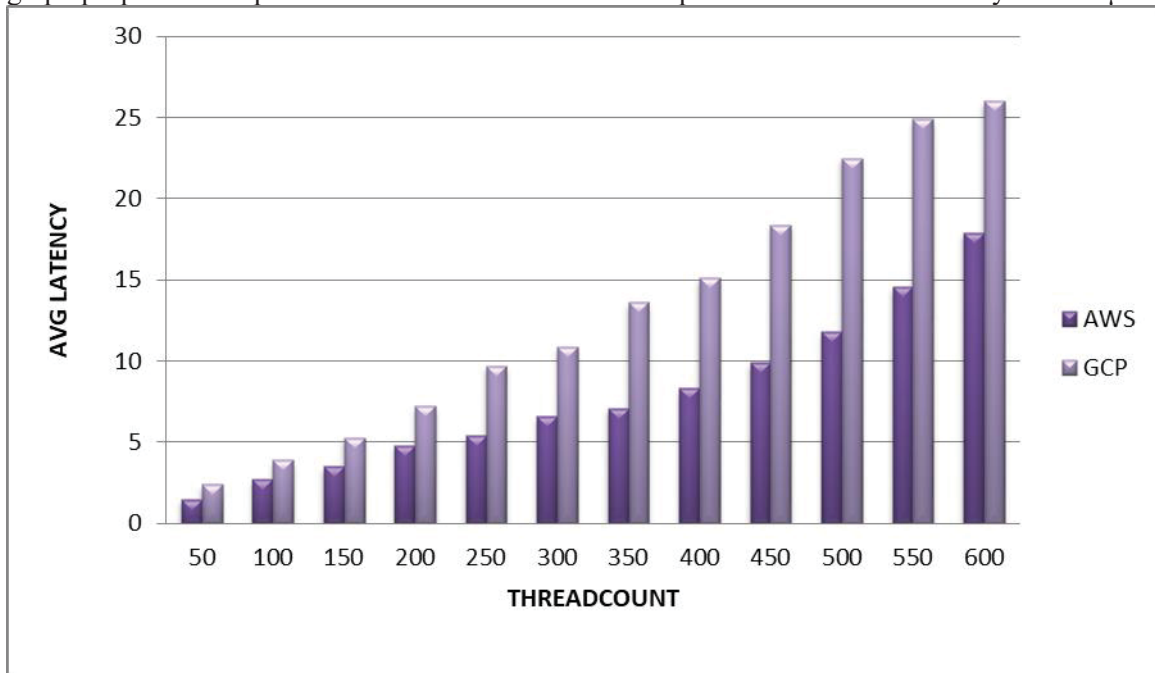


Figure 13 Average latency values of both GCP and AWS with thread count 50 to 600.

## 6 Analysis:

This section depicts the analysis for the results gathered and reasons with them. The first subsection explains analysis of the results for throughput and second subsection explains analysis of latency.

### 6.1 Throughput:

It has been observed from the results that the throughput in all the operations with different thread count has increased throughput in AWS compared to GCP. Below tables depicts average throughput, standard deviation along with the t-test value. The changes in all the thread counts with different operations are evident.

Throughput is affected by the response time of Apache Cassandra, i.e. latency. It can be observed that throughput is higher in all the cases for the AWS EC2 instances when compared to GCP instances. For the comparison of the values obtained, both t-test and standard deviation are listed below. It can be observed that AWS and GCP have more difference in terms of Throughput in the case of three node cluster.

A t-test is a statistical type which is used to determine if there exists a significant difference between two mean groups. It is used as a hypothesis testing tool. The t-test contains t-statistic, t-distribution values and degree of freedom to figure out the probability of difference between two groups of data.

Furthermore, the motivation for rejecting other statistical investigations is due to the following. Commonly Cohen's d value shows the difference in the effect size. But our goal is to analyze the difference between the two data sets and hence t-tests with unequal mean and variance are considered.

$$t = \frac{(\bar{x} - \bar{y}) - (\mu_x - \mu_y)}{\sqrt{\frac{s_x^2}{n_x} + \frac{s_y^2}{n_y}}} \text{-----(1)}$$

Where x = mean of AWS

y = mean of GCP

sx = variance of AWS

sy = variance of GCP

nx = no.of values in AWS

ny = no.of values in GCP

Where f is the degree of freedom,

$$f = \frac{\left(\frac{S_x^2}{n} + \frac{S_y^2}{m}\right)^2}{\frac{S_x^4}{n^2(n-1)} + \frac{S_y^4}{m^2(m-1)}} \text{-----(2)}$$

For the three-node cluster, the average values obtained for operation per second i.e. throughput to both AWS and GCP are compared and tabulated below.

The table 3 depicts the calculated mean, variance and standard deviation by performing statistical analysis of the data obtained from the results.

Thread count	AWS mean	AWS S.D.	GCP mean	GCP S.D.	t-tests value	f-value
50	28546	2314.5	21177	1944.10	0.0025	1.71
100	34810	2669.50	23542	1965.43	0.0031	1.71
150	36924	2917.6	24411	2393.74	0.0027	1.71
200	37923	3110.6	24956	2441.33	0.0024	1.71
250	39112	3279.15	25673	2720.92	0.0021	1.71
300	39990	3487.12	26144	2969.54	0.0020	1.71
350	41276	3573.79	26900	3064.6	0.0019	1.72
400	42651	3974.2	27633	3182.8	0.0017	1.72
450	43732	4449.93	28418	3216.2	0.0015	1.72
500	44981	4528.29	29210	3362.57	0.0014	1.72

550	46311	4538.77	30114	3430.13	0.0014	1.72
600	46994	4772.50	30997	3523.07	0.0013	1.72

table3: three-node read operation throughput values of mean, S.D. of AWS and GCP

In the above table mean, standard deviation and t-test of three-node read operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated. As in all the cases  $t < t_{\alpha-1}$ , hence hypothesis is accepted for both AWS and GCP.

Three-node write operation throughput values of both AWS and GCP:

Thread count	AWS mean	AWS S.D.	GCP mean	GCP S.D.	t-tests value	f-value
50	27836	2374.16	20682	1860.08	0.0024	1.71
100	29517	2488.05	21799	1887.3	0.0024	1.71
150	32684	2948.41	23668	2083.87	0.0022	1.71
200	34457	3018.4	23949	2204.87	0.0023	1.71
250	35963	3212.86	24640	2640.71	0.0020	1.71
300	37417	3326.34	25318	2841.88	0.0019	1.72
350	39582	3558.20	25906	3064.6	0.0019	1.72
400	41315	3924.83	26584	3195.62	0.0017	1.72
450	43216	4117.63	27730	3229.36	0.0017	1.72
500	44858	4227.94	28485	3316.70	0.0017	1.72
550	45965	4318.01	29561	3339.9	0.0016	1.72
600	46995	4758.33	30444	3563.39	0.0014	1.72

table4: three-node write operation throughput values of mean, S.D. AWS and GCP

In the above table mean, standard deviation and t-test value of three-node write operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated.

As in all the cases  $t < t_{\alpha-1}$ , hence hypothesis is accepted in both clouds.

Three-node mixed operation throughput values of both AWS and GCP:

Thread count	AWS mean	AWS S.D.	GCP mean	GCP S.D.	t-tests value	f-value
50	31176	2565.43	20334	1801.29	0.0033	1.71
100	32699	2746.7	21632	1864.83	0.0029	1.71
150	33337	3082.01	22860	2075.20	0.0022	1.71
200	34327	3257.12	23719	2124.02	0.0020	1.71
250	35771	3326.52	24456	2551.52	0.0019	1.72
300	36810	3564.02	25885	2909.63	0.0016	1.72
350	38339	3645.61	26111	3067.98	0.0016	1.72
400	39714	3970.00	27925	3132.6	0.0014	1.72
450	41183	4198.29	28634	3241.65	0.0013	1.72
500	42824	4390.74	29905	3391.53	0.0013	1.72
550	43209	4456.26	30722	3456.11	0.0012	1.72
600	44462	4566.79	31816	3521.08	0.0011	1.71

table5: three-node mixed operation throughput values of mean, S.D. of AWS and GCP

In the above table mean, standard deviation and t-test value of three-node mixed operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated:

As in all the cases  $t < t_{\alpha-1}$ , hence hypothesis is accepted.

In the below table mean, standard deviation and t-tests of five-node read operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated:

As in all the cases  $t < t_{\alpha-1}$ , hence hypothesis is accepted.

Operation	Thread count	AWS Mean	AWS S.D.	GCP Mean	GCP S.D.	t-tests value	f-value
Read	50	28656	2588.85	24871	2012.8	0.0010	1.71
	100	31798	2642.96	25190	2105.43	0.0017	1.71
	150	34982	2955.04	25900	2369.89	0.0019	1.71
	200	37193	3207.29	26781	2536.62	0.0019	1.71
	250	40119	3401.43	27987	2602.37	0.0020	1.71
	300	42187	3587.78	28567	2869.84	0.0019	1.72
	350	43978	3832.25	28900	3047.75	0.0035	1.72
	400	44891	3908.78	29108	3121.16	0.0019	1.72
	450	46092	4230.6	29956	3225.88	0.0017	1.72
	500	46900	4434.78	30145	3300.14	0.0016	1.72
	550	47722	4509.59	30978	3438.38	0.0015	1.72
	600	48981	4707.07	31782	3505.92	0.0015	1.71

table6: five-node read operation throughput values of mean, S.D. of AWS and GCP

In the below table mean, standard deviation and t-tests of five-node write operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated:

As in all the cases  $t < \alpha - 1$ , hence hypothesis is accepted.

Operation	Thread count	AWS Mean	AWS S.D.	GCP Mean	GCP S.D.	t-tests value	f-value
Write	50	29871	2622.71	22778	1927.40	0.0020	1.71
	100	31872	2726.50	23151	2122.25	0.0022	1.71
	150	32875	3017.09	24144	2364.89	0.0018	1.72
	200	33995	3111.27	24990	2528.12	0.0017	1.72
	250	36728	3404.11	25583	2736.43	0.0018	1.72
	300	38056	3724.62	26720	2756.8	0.0016	1.72
	350	41885	3981.13	27802	2989.72	0.0017	1.72
	400	43396	4024.47	28700	3074.86	0.0017	1.72
	450	44651	4283.4	29501	3178.95	0.0016	1.72
	500	45882	4454.26	30099	3254.17	0.0015	1.72
	550	46811	4659.93	31682	3497.68	0.0013	1.72
	600	47698	4868.36	32967	3505.56	0.0012	1.71

table7: five-node write operation throughput values of mean, S.D. of AWS and GCP

In the below table mean, standard deviation and t-tests of five-node mixed operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated:

As in all the cases  $t < \alpha - 1$ , hence hypothesis is accepted.

Operation	Thread count	AWS Mean	AWS S.D.	GCP Mean	GCP S.D.	t-tests value	f-value
Mixed	50	30277	2656.12	21845	2018.02	0.0023	1.71
	100	33920	2849.49	23356	2105.47	0.0025	1.71
	150	35335	3020.11	24847	2358.31	0.0021	1.71
	200	37935	3231.34	25145	2643.66	0.0022	1.71
	250	38990	3500.56	26645	2863.28	0.0018	1.72
	300	40527	3613.34	27471	2904.81	0.0018	1.72
	350	42765	3856.59	28322	3095.33	0.0018	1.72
	400	44861	4038.03	29584	3157.29	0.0017	1.72
	450	45899	4116.5	30616	3163.01	0.0016	1.72
	500	47186	4255.19	31730	3352.7	0.0015	1.72
	550	48674	4417.0	32401	3447.11	0.0014	1.72
	600	49974	4803.39	33856	3549.60	0.0013	1.71

table8: five-node mixed operation throughput values of mean, S.D. of AWS and GCP.

## 6.2 Latency:

From the results gathered in this research, that all the operations have increase in latency as the thread count increases. But in all the cases AWS has less latency compared to GCP. From the results, it can be observed that all the operations have similar average latencies on both cloud infrastructures. It can also be seen that latency decreases when the additionally two nodes were added to the cluster. Furthermore, it can also be seen that latency almost doubled in GCP 5-node cluster compared to AWS. But, eventually latency in all the cases is less in AWS when compared with GCP.

Three-node latency values of read operation for both AWS and GCP:

Thread count	AWS mean	GCP mean
50	1.3ms	1.9ms
100	2.5ms	2.9ms
150	3.5ms	3.88ms
200	4.2ms	5.1ms
250	5.3ms	7.4ms
300	6.9ms	9.2ms
350	7.89ms	11.56ms
400	8.7ms	13.3ms
450	9.9ms	15.76ms
500	11.8ms	18.33ms
550	13ms	21.4ms
600	14.8ms	25.7ms

table9: three-node read operation Latency mean values of AWS and GCP

In the below table, mean of three-node write operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated:

Thread count	AWS mean	GCP mean
50	0.7ms	1.88ms
100	1.4ms	2.9ms
150	2.2ms	4.6ms
200	4.5ms	6.2ms
250	5.4ms	7.9ms
300	5.9ms	9.5ms
350	6.8ms	11.4ms
400	9.0ms	12.5ms
450	10.5ms	14.8ms
500	11.44ms	17.6ms
550	13.4ms	19.1ms
600	14.9ms	22.4ms

table10: three-node write operation Latency mean values of AWS and GCP

In the below table, mean of three-node mixed operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated:

Three-node latency values of mixed operation for both AWS and GCP:

Thread count	AWS mean	GCP Mean
50	0.9ms	1.7ms
100	1.8ms	3.8ms
150	2.5ms	4.6ms
200	4.3ms	6.9ms
250	5.7ms	8.5ms

300	6.8ms	11.3ms
350	7.4ms	15.4ms
400	8.2ms	19ms
450	9.9ms	24.7ms
500	10.3ms	24.9ms
550	13.6ms	25.3ms
600	17.9ms	26.1ms

table11: three-node mixed operation latency mean values of AWS and GCP

In the below table, mean of five-node read operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated:

Thread count	AWS mean	GCP Mean
50	1.1ms	2.2ms
100	1.9ms	3.1ms
150	2.6ms	4.8ms
200	3.9ms	6.3ms
250	4.6ms	8.5ms
300	5.8ms	9.7ms
350	6.3ms	11.67ms
400	7.4ms	13.6ms
450	8.7ms	15.88ms
500	9.2ms	17.4ms
550	10.98ms	21.5ms
600	13.4ms	24.3ms

table12: five-node read operation latency mean values of AWS and GCP

In the below table, mean of five-node write operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated:

Thread count	AWS mean	GCP Mean
50	1.0ms	2.0ms
100	1.6ms	2.9ms
150	2.3ms	3.8ms
200	3.7ms	5.7ms
250	4.4ms	7.9ms
300	5.6ms	9.6ms
350	6.1ms	11.4ms
400	7.2ms	12.9ms
450	8.5ms	15.5ms
500	9.0ms	17.7ms
550	10.6ms	21.88ms
600	13.1ms	24.1ms

table13: five-node write operation latency mean values of AWS and GCP

In the below table, mean of five-node mixed operation with thread count 50,100,150,200,250,300,350,400,450,500,550,600 are calculated:

Five-node mixed operation latency values of AWS and GCP:

Thread count	AWS mean	GCP Mean
50	1.5ms	2.4ms
100	2.7ms	3.9ms
150	3.5ms	5.3ms
200	4.8ms	7.2ms

250	5.4ms	9.7ms
300	6.6ms	10.9ms
350	7.1ms	13.6ms
400	8.33ms	15.1ms
450	9.9ms	18.4ms
500	11.8ms	22.5ms
550	14.56ms	24.9ms
600	17.9ms	26.02ms

table14: five-node mixed operation latency mean values of AWS and GCP

## 7 Discussion:

This section compares and explains the results that have been gathered in this research with the previous related work. In this research, the performance of apache cassandra was evaluated on ec2 and gcp instances in a virtualized environment.

In [4], Ales Komarek, Jakub Pavlik, and Vladimir Sobeslav concluded that AWS has better throughput and latency compared to other cloud providers like Open Stack, GCP and Azure. In this research, AWS backs again in terms of throughput and latency compared with GCP.

In [33], V. Abramova, J. Bernardino, and P. Furtado evaluated cassandra scalability by creating 1, 3 and 6 nodes clusters with different thread count variations like 1, 6, 60, 600, 1200, 6000. The author concluded, throughput increases till thread count 600 and further no improvement was shown. In this study, throughput had increased significantly from thread count 50 to 600.

### 7.1 Answers to the Research Questions:

In this section, discusses answers for research questions and later section discusses threads to validity

#### **RQ 1) How to measure throughput on Apache Cassandra when AWS (Amazon Web services) and GCP virtual environments are considered? Also, are the results being invariant with each other?**

Answer) By performing experiment on apache cassandra nodes from three nodes, five nodes of AWS and GCP, the metrics throughput is measured using cassandra stress tool under the different thread count 50,100,150,200,250,300,350,400,450,500,550,600. This experiment has been repeated for different configurations of cassandra. From the results it is evident that as the thread count increases, throughput also increases in both AWS and GCP clouds. The mean throughput has been quantified based on different operations such as write, read and mixed.

- 3-node read operation:  
For read operation, throughput started from 28546 at thread count 50 and increased to 46994 at thread count 600 in AWS and for GCP it started from 21177 at thread count 50 and increased to 30997 at thread count 600.
- 3-node write operation:  
For write operation, throughput started from 27836 at thread count 50 and increased to 46995 at thread count 600 in AWS and for GCP it started from 20682 at thread count 50 and increased to 30444 at thread count 600.
- 3-node mixed operation:  
For mixed operation, throughput started from 31176 at thread count 50 and increased to 46804 at thread count 600 in AWS and for GCP it started from 20334 at thread count 50 and increased to 31816 at thread count 600.

From the results, it is observed that thread counts start from 50 and increases up to 600. The throughput significantly increases from thread count 50 to 600 in both cloud environments. The highest throughput is observed at thread count 600 in both clouds. When values are compared from both the clouds, AWS scales up compared to GCP in terms of throughput (higher value is better). None of the throughput values are similar in any thread count ranging from 50 to 600. Therefore, results are invariant with each other.

Furthermore, the cassandra nodes from three-nodes have scaled over to five-nodes. The mean throughputs of five- node cluster for different workloads are identified as follows:

- 5-node read operation:  
For read operation, throughput started from 28656 at thread count 50 and increased to 48981 at thread count 600 in AWS and for GCP it started from 24871 at thread count 50 and increased to 31782 at thread count 600.
- 5-node write operation:

For write operation, throughput started from 29871 at thread count 50 and increased to 47698 at thread count 600 in AWS and for GCP it started from 22778 at thread count 50 and increased to 32967 at thread count 600.

- 5-node mixed operation:

For mixed operation, throughput started from 30277 at thread count 50 and increased to 49974 at thread count 600 in AWS and for GCP it started from 21845 at thread count 50 and increased to 33856 at thread count 600.

From the above values, results are invariant with each other. It is also observed that as the thread count increases throughput also increases till the threshold limit i.e. in this research 600 in both the virtual cloud environments. But AWS has higher throughput (higher value is better) compared to GCP. Hence, AWS is better compared to GCP in terms of throughput for networking performance (throughput).

## **RQ 2) How to measure latency on Apache Cassandra when AWS (Amazon Web services) and GCP virtual environments are considered? Also, are the results being invariant with each other?**

Answer) By performing experiment on apache cassandra nodes from three, five nodes of AWS and GCP, the metrics latency is measured using cassandra stress tool under the different thread count 50,100,150,200,250,300,350,400,450,500,550,600. From the results in all the three operations as the thread count increases the average latency in both the clouds increases. From both AWS and GCP it is evident GCP has higher latency compare to AWS (lower latency is better).

The mean latency of 3-node cluster has been quantified based on the different operations such as write, read and mixed of cassandra for both clouds are as follows:

- 3-node read operation:

For read operation, latency started from 1.3ms at thread count 50 and scaled up to 14.8ms at thread count 600 in AWS and for GCP it started from 1.9ms at thread count 50 and increased to 25.7ms at thread count 600.

- 3-node write operation:

For write operation, latency started from 0.7ms at thread count 50 and scaled up to 14.9ms at thread count 600 in AWS and for GCP it started from 1.88ms at thread count 50 and increased to 22.4ms at thread count 600.

- 3-node mixed operation:

For mixed operation, latency started from 0.9ms at thread count 50 and increased to 17.9ms at thread count 600 in AWS and for GCP it started from 1.7ms at thread count 50 and increased to 26.1ms at thread count 600.

From the above obtained results, it is observed that thread counts start from 50 and increases up to 600. Similarly, latency also increases from thread count 50 to thread count 600. The highest latency is observed at thread count 600 in both clouds. But when compared the latency values from both clouds, AWS (lower is better) has lower values compared to GCP.

Furthermore, the cassandra nodes from three-nodes have scaled over to five-nodes. The mean latency of five- node cluster for different workloads with read, write and mixed operation are as follows:

- 5-node read operation:

For read operation, latency started from 1.1ms at thread count 50 and scaled up to 13.4ms at thread count 600 in AWS and for GCP it started from 2.2ms at thread count 50 and increased to 24.3ms at thread count 600.

- 3-node write operation:

For write operation, latency started from 1ms at thread count 50 and scaled up to 13.1ms at thread count 600 in AWS and for GCP it started from 2ms at thread count 50 and increased to 24.1ms at thread count 600.

- 3-node mixed operation:

For mixed operation, latency started from 1.5ms at thread count 50 and increased to 17.9ms at thread count 600 in AWS and for GCP it started from 2.4ms at thread count 50 and increased to 26.02ms at thread count 600.

From the above values, results are invariant with each other. It is also observed that as the thread count increases latency also increases in both the virtual cloud environments. But GCP has higher latency compared to AWS. The author [4] evaluated the scalability characteristics of Cassandra architecture in terms of throughput and latency. The author concludes from his research that GCP scales up in terms of latency compared to AWS, open Stack. Similarly, the current research concludes by comparing the statistical analysis of both cloud's latency values, it is evident that GCP scales up compared to AWS (lower is better). In terms of latency, lower values are better; hence AWS is better compare to GCP.

## **7.2 Validity threats:**

Following are the threats identified during this thesis study and are classified into two types:

- Internal: An internal threat occurs without the knowledge of the researcher [39]. Selection of tools, instruments required for the experiment or due to selection of components in the experiment [23] leads to errors or mistakes. The internal threat error is mitigated by repeating the experiment several times at every stage on different types of workloads. Therefore, in this case, experiment is repeated five times in order to obtain results by simulating multiple times and statistical data has been extracted using standard deviation.
- External: The results obtained can be generalized to other environments is known as external threat. The major threats in this study are databases, cloud providers because they might be outdated. To avoid this threat, AWS, GCP, cassandra stress, and cassandra were the trending technologies used to carry out this experiment.

## 8 Conclusion And Future Work:

The main motive of this research is to test if the Cassandra can manage high load and how the cluster size affects the execution time in both AWS and GCP virtual environments and comparing their results whether they are variant or invariant. At the beginning, we created 3, 5-node cluster for both AWS and GCP instances with same workloads and benchmarked it using cassandra stress tool. The metrics throughput, latency is calculated for same workloads under t2 small instances to test whether cassandra can manage high load when the thread counts increases from 50 to 600 in both AWS and GCP. The results concluded that performance of Cassandra increases till the threshold value i.e. 600 with 10 million reads, writes and mixed. Later we compared the results from both AWS and GCP clouds to validate whether the results obtained are variant or invariant. From, the results it is concluded that AWS is better compared to GCP, in terms of both latency and throughput. The key-takeaways from this research are as follows. This research has a value to companies to decide which cloud provider to choose in terms of networking and storage performance i.e. latency and throughput. To give a wider view to the reader, different workloads have been simulated to view how AWS and GCP clouds behave when simulated.

The same experiment can be conducted in different cloud environments like Azure cloud with GCP or Azure cloud with AWS. The benchmarking tool can be replaced with YCSB tool with different workloads like workload A, B, C, D, and E or with stress -n tool. The database can be replaced with mongo DB.

## 9 REFERENCES:

- [1] “Cloud Computing Virtualization | Virtual Machine | Virtualization.” [Online]. Available: <https://www.scribd.com/document/355509902/Cloud-Computing-Virtualization>. [Accessed: 15-May- 2018].
- [2] H. Takabi, J. B. D. Joshi, and G. J. Ahn, “Security and Privacy Challenges in Cloud Computing Environments,” *IEEE Security Privacy*, vol. 8, no. 6, pp. 24–31, Nov. 2010.
- [3] “IEEE Xplore Full-Text HTML : Security model for cloud database as a service (DBaaS).” [Online]. Available: <https://ieeexplore-ieee-org.miman.bib.bth.se/xpls/icp.jsp?arnumber=7336974>. [Accessed: 15-May-2018].
- [4] Komarek A., Pavlik J., Sobeslav V. (2017) Performance Analysis of Cloud Computing Infrastructure. In: Younas M., Awan I., Holubova I. (eds) Mobile Web and Intelligent Information Systems. MobiWIS 2017. Lecture Notes in Computer Science, vol 10486. Springer, Cham.
- [5] Gillam, L., Li, B., O’, J. and Loughlin, N. (2014). Benchmarking cloud performance for service level agreement parameters. *International Journal of Cloud Computing*, 3(1), p.3.
- [6] Planet Cassandra, <http://planetcassandra.org/>  
HYPERLINK "http://planetcassandra.org/".
- [7] Apache Cassandra Project, <http://cassandra.apache.org/>
- [8] “Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS,” Amazon Web Services, Inc. [Online].
- [9] <https://www.youtube.com/watch?v=kGv0Xt1u9V0&t=2s>
- [10] Y. T. Hsu, Y. C. Pan, L. Y. Wei, W. C. Peng, and W. C. Lee, “Key Formulation Schemes for Spatial Index in Cloud Data Managements,” in 2012 IEEE 13th International Conference on Mobile Data Management, 2012, pp. 21–26.
- [11] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, May 2010.
- [12] J. Varia, “Best practices in architecting cloud applications in the AWS cloud,” *Cloud Comput. Princ. Paradig.*, pp. 457–490, 2011.
- [13] V. Abramova and J. Bernardino, “NoSQL databases: MongoDB vs Cassandra,” in *Proceedings of the International C\* Conference on Computer Science and Software Engineering*, 2013, pp. 14–22.
- [14] R. Buyya, “Cloud computing: The next revolution in information technology,” in *Parallel Distributed and Grid Computing (PDGC)*, 2010 1st International Conference on, 2010, pp. 2–3.
- [15] P. Mell, T. Grance, and T. Grance, “The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology.”
- [16] [https://d1.awsstatic.com/whitepapers/Cassandra\\_on\\_AWS.pdf](https://d1.awsstatic.com/whitepapers/Cassandra_on_AWS.pdf)
- [17] A. Lakshman, “Cassandra - A Decentralized Structured Storage System.” *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.
- [18] [https://en.wikipedia.org/wiki/Log-structured\\_merge-tree](https://en.wikipedia.org/wiki/Log-structured_merge-tree)
- [19] “Performance Evaluation of Cassandra Using Cloud Services - Google Search.” n.d. Accessed April 22, 2018.
- [20] E. Casalicchio, L. Lundberg, and S. Shirinbab, “Energy-aware Auto-scaling Algorithms for Cassandra Virtual Data Centers,” *Clust. Comput.*, vol. 20, no. 3, pp. 2065–2082, 2017.
- [21] Cassandra, Apache. 2015. “Apache Cassandra.” *Google Scholar*.
- [22] Y.-C. Chen, S.-T. Wang, H.-Y. Chang, T.-M. Chen, and C.-H. Li, “The performance analysis for virtualisation cluster and cloud platforms,” *Int. J. Comput. Sci. Eng.*, vol. 6, no. 4, pp. 255–263, Jan. 2011.

- [23] “Performance Evaluation of Cassandra in a Virtualized Environment.” n.d. Accessed June 5, 2018. <http://bth.divaportal.org/smash/record.jsf?pid=diva2%3A1083541&dswid=9605>.
- [24] “Documentation - Apache Cassandra TM,” 2016.
- [25] Gari, Subba Reddy, and Avinash Kumar Reddy. “Performance Evaluation of Cassandra in AWS Environment: An Experiment,” n.d., 52.
- [26] Yair Levy and Timothy J. Ellis, “A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research.” [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.2369&rep=rep1&type=pdf>. [Accessed: 15- May-2018]
- [27] <https://www.devinline.com/2018/09/cassandra-cluster-on-google-compute.html>.
- [28] “Machine Type Comparative of Leading Cloud Players Based on Performance & Pricing - IEEE Conference Publication.” Accessed September 3, 2019. <https://ieeexplore-ieee-org.miman.bib.bth.se/document/8554366>.
- [29] E. Hewitt, Cassandra - The Definitive Guide. 2010.
- [30] Gandini, Andrea, Marco Gribaudo, William J. Knottenbelt, Rasha Osman, and Pietro Piazzolla. 2014. “Performance Evaluation of NoSQL Databases.” In *Computer Performance Engineering*, edited by András Horváth and Katinka Wolter, 8721:16–29. Cham: Springer International Publishing. [https://doi.org/10.1007/978-3-319-10885-8\\_2](https://doi.org/10.1007/978-3-319-10885-8_2).
- [31] [https://www.tutorialspoint.com/cassandra/cassandra\\_architecture.htm](https://www.tutorialspoint.com/cassandra/cassandra_architecture.htm)
- [32] Srinadhuni, Siddhartha. 2018. *Performance Evaluation of Cassandra Scalability on Amazon EC2*. <http://urn.kb.se/resolve?urn=urn:nbn:se:bth-16141>.
- [33] V. Abramova, J. Bernardino, and P. Furtado, “Evaluating cassandra scalability with YCSB,” in International Conference on Database and Expert Systems Applications, 2014, pp. 199–207.
- [34] “Installing DataStax Community on Debian-based systems.” [Online]. Available: <http://docs.datastax.com/en/archived/cassandra/2.2/cassandra/install/installDeb.html>. [Accessed: 11- Jan- 2017].
- [35] “Top 7 Advantages of Using Google Cloud Hosting - Whizlabs Blog.” Accessed September 3, 2019. <https://www.whizlabs.com/blog/google-cloud-hosting/>.
- [36] “Whitepapers – Amazon Web Services (AWS),” Amazon Web Services, Inc. [Online].
- [37] “Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS,” Amazon Web Services, Inc. [Online].
- [38] “Installing DataStax Community on Debian-based systems.” [Online]. Available: <http://docs.datastax.com/en/archived/cassandra/2.2/cassandra/install/installDeb.html>. [Accessed: 11- Jan- 2017].
- [39] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, 28 Experimentation in software engineering. Springer, 2012.



