Master of Science in Computer Science
February 2017

Performance evaluation of Cassandra in AWS environment

An experiment

Avinash Kumar Reddy Subba Reddy Gari

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

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

Contact Information:
Author(s):
Avinash Kumar Reddy Subba Reddy Gari
E-mail: avsu15@student.bth.se

University advisor:
Emiliano Casalicchio
Department of Computer Science and Engineering

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

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ABSTRACT

Context. In the field of computer science, the concept of cloud computing plays a prominent role which can be hosted on the internet to store, manage and also to process the data. Cloud platforms enables the users to perform large number of computing tasks across the remote servers. There exist several cloud platform providers like Amazon, Microsoft, Google, Oracle and IBM. Several conventional databases are available in cloud service providers in order to handle the data. Cassandra is a NoSQL database system which can handle the unstructured data and can scale large number of operations per second even across multiple datacentres.

Objectives. In this study, the performance evaluation of NoSQL database in AWS cloud service provider has been performed. The performance evaluation of a three node Cassandra cluster is performed for different configuration of EC2 instances. This performance has been evaluated using the metrics throughput and CPU utilization. The main aim of this thesis was to evaluate the performance of Cassandra under various configurations with the YCSB benchmarking tool.

Methods. A literature review has been conducted to gain more knowledge about the current research area. The metrics required to evaluate the performance of Cassandra were identified through literature study. The experiment was conducted to compute the results for throughput and CPU utilization under the different configurations t2.micro, t2.medium and t2.small for 3 node and 6 node cluster using YCSB benchmarking tool.

Results. The results of the experiment include the metrics, throughput and CPU utilization which were identified in the literature review. The results calculated were plotted as graphs to compare their performance for three different configurations. The results obtained were segregated as two different scenarios which were for 3 node and 6 node clusters.

Conclusions. Based on the obtained values of throughput the optimal or sub-optimal configuration of a data centre running multiple instances of Cassandra such that the specific throughput requirements are satisfied.

Keywords: Cassandra, AWS, YCSB, throughput, CPU utilization

ACKNOWLEDGEMENT

I would like to extend my immense gratitude towards my supervisor, Emiliano Casalicchio who has provided me with some very valuable inputs and suggestions. He was always available for questions that I had however trivial they may be.

I am thankful to my parents and my brother for their unconditional love. They have provided me with emotional support and guidance without which I wouldn't have been successful in completing my M.Sc.

I express my gratitude towards my friends namely Sowrabh Mummadi, Chaitanya Malladi Kashyap Boinapally, Prathisrihas Konduru, Swarajya Haritha Reddy Gadila, Pavan Varma Indukuri, Sai Srinivas Bodireddigari and Sai Naresh Kotikalapudi who were like a backbone to me when my spirits were down and made my time here in Sweden fun and memorable. They were always there during the hour of need and provided me with words of encouragement and guidance that helped me in completing my master thesis.

LIST OF TABLES

LIST OF FIGURES

Contents

1 INTRODUCTION

1.1 Overview

The concept of cloud computing is one of the biggest landmarks in the current scenario of the software industry in handling several problems such as data handling and security [1] . The virtualization technology acts as an essential component which can achieve the purpose of the cloud-based platforms and also has the capability to run the virtual machines on the virtual machine monitor (VMM) or hypervisor [2].

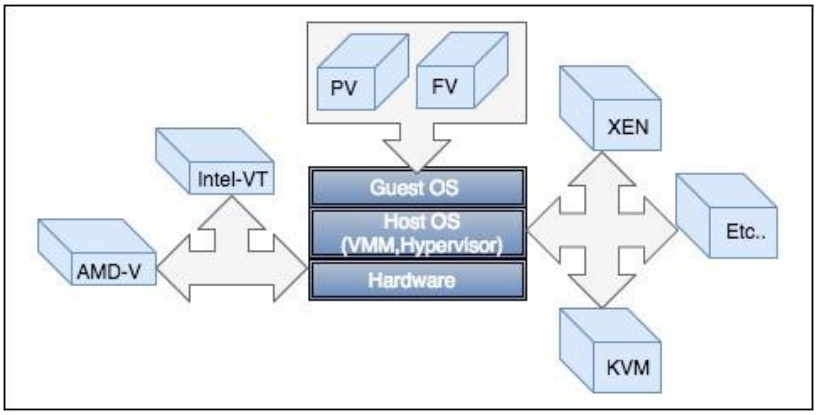

Figure 1: Virtualization architecture

The process of virtualizing the cloud platform increments the availability of resources and also provides flexibility in its management. The mechanism of hardware multiplexing enables to optimize the cost and energy saving [3]. The process of virtualization allows the user to share resources from different data centers and across multiple users [1]. The big data systems are very productive in handling the unstructured data. Apache Cassandra, mongo DB, and Hadoop are some of the effective big data systems used to handle several applications in real time.

Apache Cassandra is a NoSQL open source database management system which is designed to manage big data with high scalability and availability [4]. It supports the mechanism of replication across multiple data centers by supporting least latency for the users [5]. Cassandra supports replication of data automatically across a multiple number of nodes and also across multiple data centers because of which the replacement of failed nodes can be done immediately [5]. AWS provides special web service known as amazon elastic compute cloud (EC2) which provides a resizable compute capacity in the cloud environment in which a virtual machine can be deployed. Amazon EC2 enable the user to launch the instances and provides several advantages such as elastic web-scale computing, complete user control, flexible cloud hosting services, reliability, and security [6]. AWS is a virtual environment which supports web services API for launching and administrating virtual machines [2]. Amazon provides cloud US West, US East, Canada, south America, Europe, Asiapacific and china [6].

In this research, Cassandra is deployed in Frankfurt amazon servers. In this research, the performance of cloud-based Cassandra is analyzed by benchmarking using YCSB- yahoo cloud serving benchmark which is one of the most used benchmarks available to test the NoSQL databases [7]. The main role of YCSB is to

generate the workload in order to perform heavy load of operations the performance of different key-value and cloud serving stores [7].

1.2 Problem identification

Currently, Cloud Computing plays an important role to manage a large-scale data analysis due to its scalability and reliability [8]. It has turned up to be a new paradigm for delivering and receiving the services over the internet [9]. Although cloud computing offers huge opportunities to the IT industry, the development of cloud computing technologies is still at its early stages, with many issues still need to be addressed [10]. Due to the expeditious development of processing and storage technologies, computing resources have become cheaper and powerful than before [10]. In a cloud computing environment, AWS is one set of cloud service providers which is currently used by a large number of companies [9], [10]. AWS is a cloud service that provides a highly reliable and scalable framework to deploy web-scale solutions with minimal support and administration costs [10]. The offerings of AWS can be accessible over HTTP using REST and SOAP protocols [9].

With the growth in the development of Information and Communication technology, the storage type, functionalities and interaction with databases have improved [11]. Nowadays cloud computing plays an important role in dealing with huge amount of data flexibly by developing new cloud data managements which are NoSQL databases [8]. NoSQL databases were developed as scalable databases to allow data distribution over a number of servers easily [12]. They were extensively used in non-relational technology with the increased interest of researchers and companies [12]. With the growth of the database size exponentially, the access to the data also has to be made more efficiently [11]. When the amount of data increases in the servers, databases also become larger [11]. This has led to the well-known problem of efficiency in information abstraction [11]. Therefore, these situations led to the research in performance evaluation of NoSQL database in AWS. This study has been done to explore more about the performance of NoSQL database in an AWS cloud server.

1.3 Aim and objectives

The main aim of this research is to evaluate the performance of Cassandra in AWS environment under different configurations when benchmarked with YCSB

- Analyze the architecture and working of Cassandra
- Understanding the working of AWS cloud platform
- Identify the performance parameters to evaluate
- Deployment of Cassandra in servers of AWS
- Identify the workloads which can be generated by YCSB
- Deploy Cassandra in different configuration of AWS servers to measure scalability
- Evaluate the results and analyze the behavior of Cassandra based on its performance metrics.
- Parameterize the optimal or sub-optimal configuration of a data centre running multiple instances of Cassandra such that the specific throughput requirements are satisfied.

1.4 Research questions

RQ1: What is the throughput achievable for different VMs configuration when running Cassandra virtual nodes and when stressed by a different type of workload?
Motivation: to understand the behavior of Cassandra and identify the average number of operations per second which a user can perform under different types of workload for different virtual configuration.

RQ2: What is the throughput scalability of Apache Cassandra on a virtual environment for the different type of VM configuration and under different workload?
Motivation: to analyze the maximum number of operations per second which a user can perform with the increase in number of virtual nodes for different types of workloads.

RQ3: How can we find the optimal or sub-optimal configuration of a data center running multiple instances of Cassandra data center in a way that specific throughput requirements are satisfied?
Motivation: Since there exists no studies on multitenant Cassandra systems RQ3 will be answered using the results from RQ1 and RQ2 and running an optimization of new model which is available at Department of computer science and computer engineering (DIDD).

1.5 Thesis outline

Section 1 gives a brief introduction to the research work. Section 2 gives the background works about the technologies, cloud computing, apache Cassandra, AWS environment and YCSB. Section 3 gives an idea of earlier studies conducted in the selected research area including their contributions. Section 4 gives the literature review and the experiment methods which are used in this research. The results obtained from the conducted experiment are portrayed in section 5 followed by the analysis in section 6. The conclusion of this research, followed by possible areas for the future work are mentioned in section 8. This paper is concluded with references.

2 BACKGROUND

In the recent times, the usage of cloud storage has been increasing rapidly since it provides the flexibility and availability of the data to the user from any part of the computing world [13]. There has been a huge investment in developing cloud computing platforms by several countries and enterprises. A large number of storage devices, when integrated with cloud computing system, is converted into cloud storage systems which compute and processes a huge amount of data storage and management [14]**.** The data that is stored can be of any format such as text files, pictures, and videos. In order to handle such large number of data, NoSQL management systems such as Cassandra mongo DB, Hadoop helps to manage and distribute large sets of data [15] [16].

Cassandra is an open source NoSQL database management system which was developed initially by Facebook [5]. It was designed to handle a huge amount of data which is spread across multiple commodity servers. Cassandra offers several advantages over other NoSQL databases such as elastic scalability, rapid linear-scale performance, flexibility in data storage, ease of distribution of data, transactional support, and fast writes [17]**.** Large scale companies such as eBay, GitHub, Facebook, Netflix, Instagram, Reddit and more than 1500 companies are actively using Cassandra database management system [17]**.** In this paper in order to evaluate the performance of Cassandra in cloud-based platforms such as amazon web services (AWS), Microsoft azure, google cloud, the configuration of Cassandra is set up with its default configuration and is launched in different configurations of EC2 instances. The Cassandra EC2 instances are benchmarked with different workloads generated by YCSB.

2.1 Overview of apache Cassandra

Cassandra is an open source distributed database management system [4] it was designed in such a way that has the ability to handle huge workloads of data across different nodes with no single point of failure. [4]. Cassandra implements peer-to-peer distributed method across similar nodes where the data is distributed across all the nodes in the cluster. The communication of the nodes is occurred using simple network topology strategy. The architecture of Cassandra allows any authorized user to get the connection of a node in any datacenter and can access the data using Cassandra query language (CQL). To get a clear view of the architecture of Cassandra, it includes the following key structures [18] [19].

2.1.1 Node

The node is a component where the data is stored. For a given cluster of Cassandra, if the user writes some data in a specific node, it can be replicated across all other nodes in the cluster. Replication factor helps to replicate the data across the remaining nodes of a cluster, which helps the user to perform heavy operations. In the architecture of Cassandra, all the nodes in the cluster have same priority since they form a ring topology [18]. In this research the Cassandra nodes are implemented in AWS instances of type t2 micro, t2 small and t2 medium [19].

2.1.2 Datacentre

The datacentre is a collection of Cassandra nodes, which can be done either physical or virtual. A data center is created in order to separate the set of nodes of

different clusters. The default communication strategy does not let the data centers to communicate with each other. The network topology enables the data centers to communicate with each other. Using multiple data centers keeps the request nearby to each other for minimum latency and avoids the transactions of Cassandra to be influenced by different workloads [18] [19] .

2.1.3 Cluster

A cluster is an essential component that contains one or more data centers. Each cluster is unique from each other i.e. two clusters cannot communicate with each other. Any number of nodes can be added into the cluster and also can be deleted.

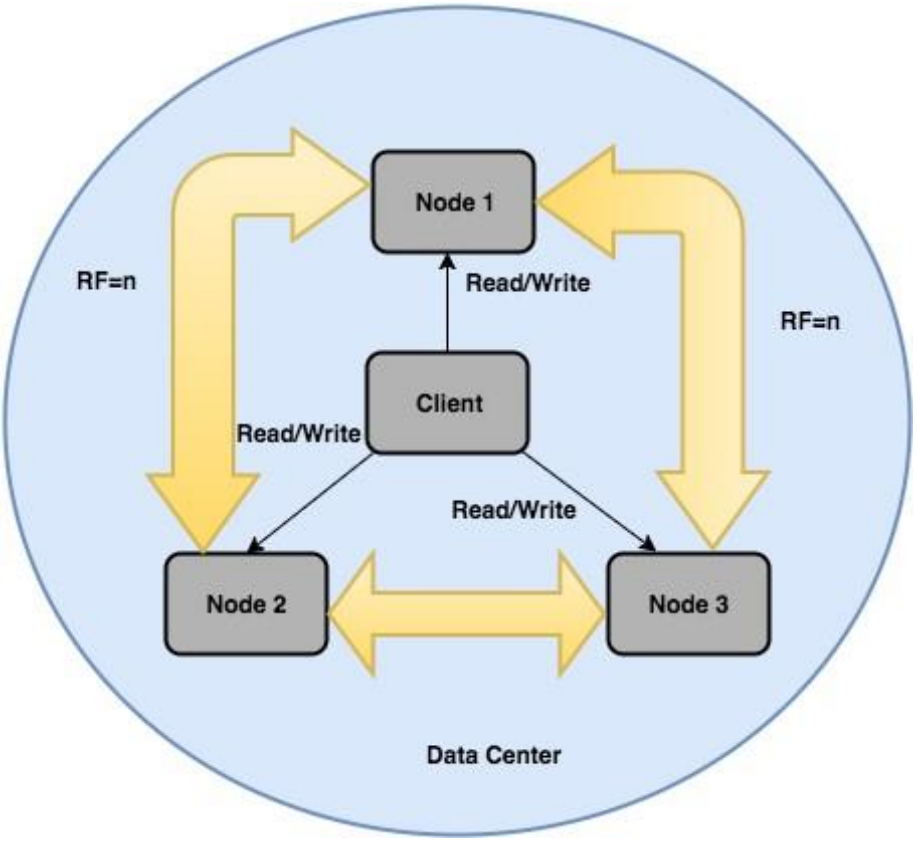

Figure 2: Cassandra Architecture

2.1.4 Keyspace

The keyspace in Cassandra is a peripheral container for the data in Cassandra. The keyspace contains the following attributes:

Replication factor: It is defined as a total number of replicas that is possible across the nodes in the cluster. i.e. if a replication factor is configured as 3, the data is copied into 3 nodes in the cluster. In general, the replication factor is set for each datacenter which is greater than one and limited to the number of nodes present in the cluster.

Replica placement strategy: In order to provide reliability and fault tolerance, Cassandra saves replicas of the data on multiple nodes which is determined by the replication strategy. In general, Network Topology Strategy is deployed because of its capacity to increase to multiple datacenters if required in the further works.

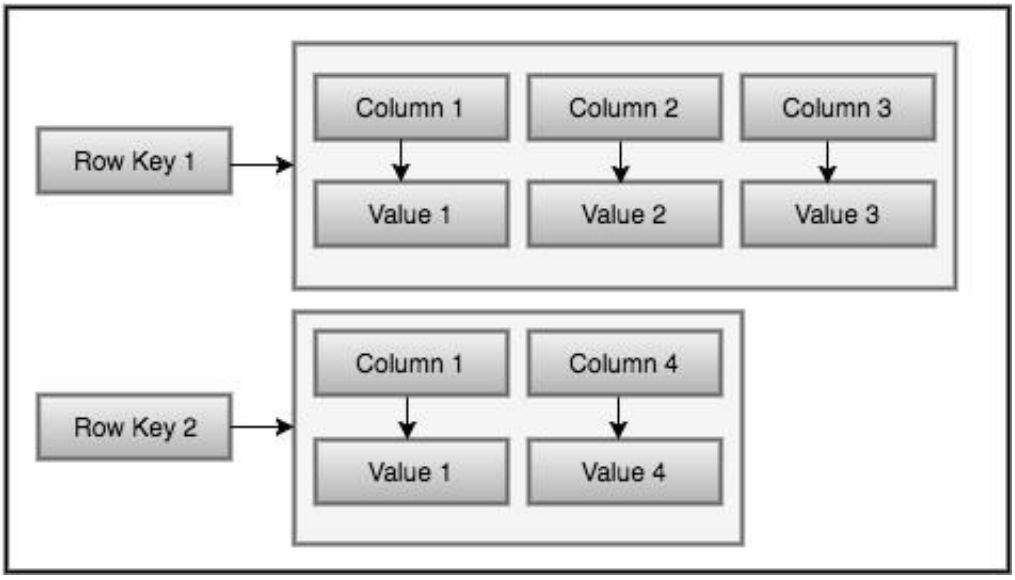

Figure 3: Keyspace structure

2.1.5 Snitch

A snitch is a collection of machines into the data centers and the racks for which the replications strategy collectively uses to locate the replicas. In order to configure the snitch while forming a cluster, the default "*SimpleSnitch*" must be changed to "*GossipingPropertyFileSnitch*" in *Cassandra YAML* configuration file.[18]

2.2 Overview of Amazon web services

Amazon web service (AWS) is one of the cloud computing platform provided by amazon which provides several features such as running a virtual server, store files, share digital media, deploy a website, host websites, running database, analyze the user data. These services enable any user to avoid financially and hardware expenses and its maintenance problems. Amazon web services provide data centers in 12 geographical locations, which helps the user to create virtual machines and deploy any database management systems in it. The following are some of the key parameters we need to configure in order to create a virtual server.

2.2.1 Virtual private cloud (VPC)

Amazon virtual private cloud provides several resources to create a virtual network which a user can define. The resources of amazon web services can be launched in the created virtual network. This virtual network provides the user providing a range of IP addresses, the creation of the subnets, configure of route tables and network gateways [15] [16]**.** The main benefits of launching instances using a virtual private cloud are: using public subnets user can connect directly to the internet, using private subnets user can connect to the internet using Network Address Translation (NAT) gateway, connect safely to the collective datacenters, connect privately to different VPC's.

2.2.2 Subnet

A subnet is a network that is created for the communication of the instances. In the AWS, multiple instances can be created in a single VPC. Each VPC has some default

subnets, each subnet is unique from each other. In this paper, the Cassandra instances are launched in default subnets available. The AWS also provides the user to create new subnets so that they can launch the instances in it [20] [18].

2.2.3 Security group

In order to control the inbound and outbound traffic of the instance a security group operates as a virtual firewall. For a given instance launched in a VPC, the user can assign a maximum of five security group. In order to establish network communication between the amazon EC2 instances, they must be under same security group [15] [18].

Inbound			
Source	**Protocol**	**Port Range**	**Description**
0.0.0.0/0	TCP	80	HTTP inbound access is allowed from all IPv4 addresses
: :/0	TCP	80	HTTP inbound access is allowed from all IPv6 addresses
0.0.0.0/0	TCP	443	HTTP inbound access is allowed from all IPv4 addresses
: :/0	TCP	443	HTTP inbound access is allowed from all IPv6 addresses
Network's public IPv4 address range	TCP	22	Inbound SSH access is allowed from IPv4 IP addresses from network to Linux instances
Network's public IPv4 address range	TCP	3389	Inbound RDP access is allowed from IPv4 IP addresses from network to Windows instances
Outbound			
Destination	**Protocol**	**Port Range**	**Description**
ID of security group – To access database servers	TCP	1433	Outbound Microsoft SQL server is allowed to access instances in specified security group
ID of security group – To access MySQL database servers	TCP	3306	Outbound MySQL server is allowed to access instances in specified security group

Table 1: List of Security groups

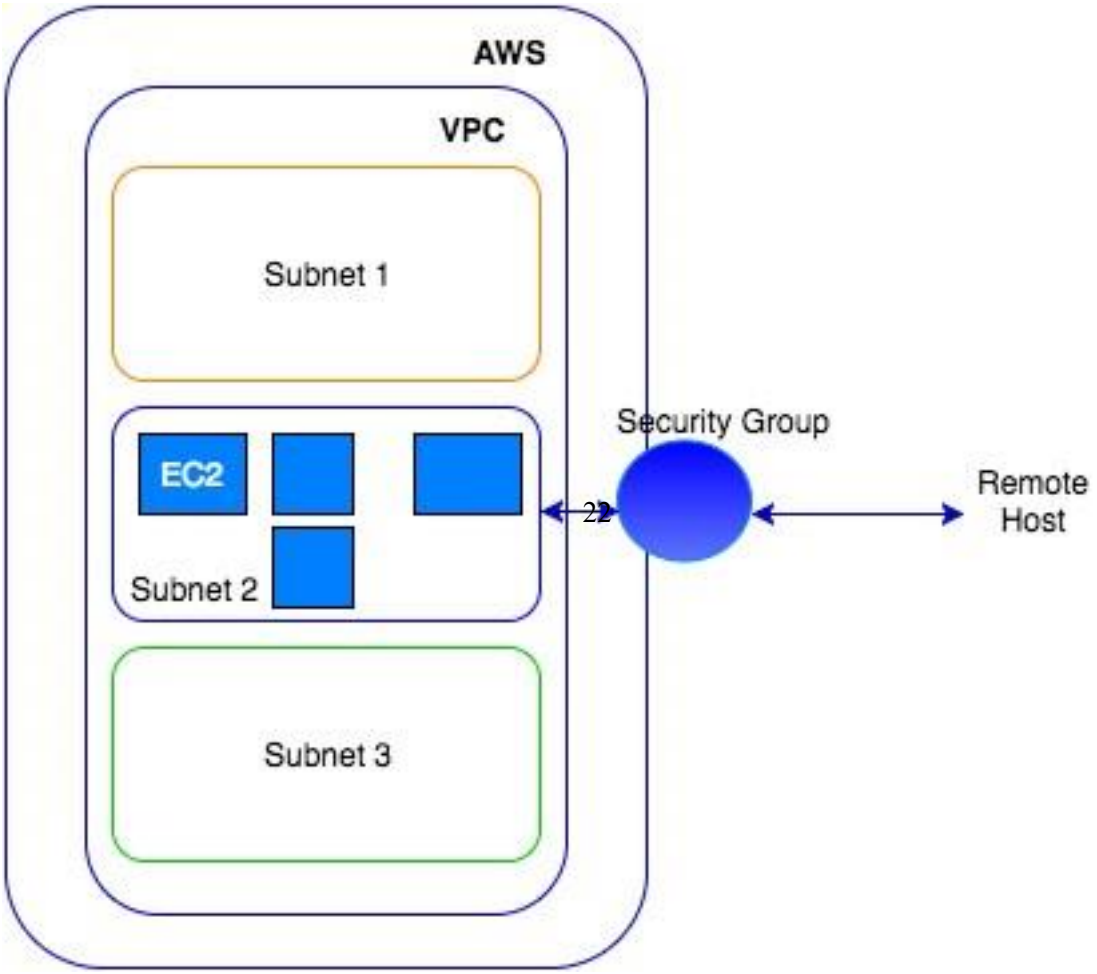

Figure 4:AWS architecture

2.3 Cassandra data structure

The architecture of Cassandra stores data in the form of tables. The keyspaces in Cassandra contains set of tables which are associated with different parameters such as replication factor, replica placement strategy, column families [21] [18].

2.3.1 Column family

The column family in the Cassandra contains an ordered collection of rows. Each row, in turn, is again an ordered collection of columns. In Cassandra, the column families are defined but not the columns. The user can add a column to any column family. The basic structure of column consists of the following: namely key, value and timestamp [18].

2.3.2 Super column

The super column is a special column which stores the outline of sub-columns. In general, the column families are located on the disk in individual files. In order to achieve optimal performance, the columns are categorized in the column family and a super column is useful [18].

2.4 Cassandra parameters

2.4.1 Cache performance (read performance)

Cassandra provides integrated caching and cache data distribution of data across the cluster. In a given cluster when the nodes are down, the user can read the data from another cached replica of the data [22] [18]. the process of caching is started only when a table is created. The data caching in Cassandra is of two types namely:

- Row caching
- Partition key caching

In order to improve the read performance and causes read path short, we can enable caching. While creating a table caching can be enabled or disabled. The read performance can be improved up to 30% by the row cache, which can access the recent data [23]**.** In order to reduce excessive usage of resources, the key cache is configured for high performance. The cache performance can be optimized through six ways out of which three are through key cache parameters. They are:

- Number of keys
- Size of the cache (KEYCACHE SIZE)
- Time period to save (KEYCACHE SAVE TIME PERIOD)

Since cache memory uses the heap memory, there exists high usage of system resources when we configure the cache parameters. The following are a list of system metrics that can be effected by varying the cache parameters:

- Memory utilization
- Read throughput (MBps)
- Overall throughput (ops/sec)
- CPU utilization

2.4.2 Mem table (write performance)

In order to improve the write performance, Cassandra does not provide much configuring parameter options to the user. The compaction and compression are the factors that influence the disk write throughput and the disk space. By varying the size in SStables or by describing the compaction strategy we can achieve optimal compaction. The data that is written into Cassandra enters into two regions namely: memTable and comitlog. By configuring the comitlog parameters there exists improvement in the disk throughput and write performance [24] [18]**.**

2.5 Yahoo cloud service benchmark

Yahoo! Cloud Service Benchmark (YCSB), is aimed to develop a framework and an arrangement of workloads for assessing the performance of various “key-value” and “cloud” serving platforms [12]. YCSB is a java embedded client which is designed to generate the operations to form a workload [25]. The YCSB has the following components:

- YCSB Client; which is an extensible workload generator.

- Core workloads; which is a set of workload scenarios to be executed by the generator.

The core workloads give a balanced picture of the performance of a system, but the client is extensible so that you can define new and diverse workloads to look at system aspects, or application scenarios, not satisfactorily secured by the core workload. Correspondingly, the client provides extensive support to benchmark different databases. YCSB provides an example code for benchmarking the databases like HBase, Cassandra, Infinispan and MongoDB, though it is clear to write a new interface layer to benchmark any database [25].

A typical utilization of the instrument is to benchmark numerous frameworks and compare them. For instance, you can introduce various frameworks on a simple hardware configuration, and run the same workloads against each framework. Then performance can be plotted for each system to see which system gives better performance.

2.5.1 Workload

YCSB incorporates a set of core workloads that characterize an essential benchmark for cloud systems. However, the workloads can be defined for each database. YCSB core workloads are useful for initial step. To understand the performance trade-offs of different systems, obtaining these benchmark numbers for different systems are considered. The following are some of the core workloads that can be used for testing a use case:

Workload A: Update heavy workload

This workload is a mix of 50/50 reads and writes. The session store recording of recent actions is an application example [25]

Workload B: Read mostly workload

This workload is a 95/5 reads/write mix. Photo tagging is an application example which has most operations as read tags [26]

Workload C: Read only

This workload is 100% read. Construction of user profile cache-profile cache is an application example [25]

Workload D: Read latest workload

In this workload, new records are inserted, and the most recently inserted records are popular. User status updates is an application example [25].

Workload E: Short ranges

In this workload, the short ranges are questioned, rather than the individual records. In the case of thread based conversations, for given thread, each scan is performed which is an application example [22].

Workload F: Read-modify-write

In this workload, read-modify-writes are performed [25].

The YCSB client enables to modify some of the core properties that can before running a workload. The following are some of the core properties that can be set:

- **Workload:** the workload class to be used.
- **Db:** the database class to be used.
- **Exporter:** the measurement exporter class to be used.
- **Export file:** for the path to a file where the output must be written, instead of stdout.
- **Threadcount:** the number of YCSB client threads.

Measurement type: Histogram and time series are the measurement types supported by ycsb. [26]

Core workload package properties:

The property file which can be used for the core workload generator can be specified with the following parameters [26]

- **Field count:** the number of fields in a record.
- **Field length:** the size of each field.
- **Read all fields** reads the read fields either true or false.
- **read proportion:** the proportion of operations which should be read.
- **Update proportion:** the proportion of operations.

3 RELATED WORK

In this section, the related research articles that were earlier conducted on cloud computing and NoSQL databases are discussed.

3.1.1 State of art

Cassandra database management system is used to store large data seta and also supports cloud infrastructure making it suitable for smart grid implementation. There exists a wide range of applications that extensively use Cassandra database management system such as Facebook, eBay, Netflix due to its scalability and reliability [27]. Cassandra was designed to handle large amount of distributed data and requests while providing no single point of failure and automatic data distribution over a cluster [12]. It performs consistently than other popular NoSQL database management since it provides multiple choices because of its fundamental architecture. [17]. In this research, the process of configuring the Cassandra in amazon cloud environment across different configurations of instances has been addressed. The performance of Cassandra is evaluated by different metrics based on the workload when benchmarked by YCSB. The architecture of Cassandra has also been discussed.

In the paper [28] the performance of MySQL, apache Cassandra, and HBase is evaluated for heavy write operations by a web-based REST application. The experiment involves a comparison of throughputs with respect to a number of transactions per second (TPS) for standard relational database MySQL and two popular NoSQL databases namely Cassandra and HBase. This paper concludes that Cassandra has more scalability than other two databases and provides rapid fast write speeds and HBase performs write speed nearly double the write speed resulted by the traditional relational database, MySQL.

In the paper [29] presents a detailed evaluation of apache Cassandra NoSQL database when it is used in affiliation with Hadoop map reduce engine. Also this paper describes the performance for a broad range of representative use cases which includes its comparison and evaluation. This evaluation of performance of Cassandra enables to educate the application developers and can be able to make decisions depending on the size of data, size of cluster, replication factor and also the strategy of partitioning in order to meet the requirements of performance. This paper concludes that the turnaround time of Hadoop is not affected by an increase in the replication factor of apache Cassandra and the performance of CPU loads is decent on using Hadoop-naïve, but the difference is minimum on using Cassandra.

In the paper [30] presents a brief description of the design of performance monitoring tool and the results generated are represented in the form of statistics and graphs. This helps to make the decisions for the optimization of the performance of Cassandra on different platforms. This paper also discusses the node utility which enables to gather performance statistics of Cassandra.

In the paper [31] presents a description of the architecture of quality of services (QoS) infrastructure in order to attain controlled performance of an application over apache Cassandra distributed database storage system. The implementation of the architecture and the results are achieved from the evaluation using YCSB on the amazon EC2 cloud servers. The evaluation of the results depicts that the methodology

is much efficient in predicting the capacity of the server requirements for the given sample descriptions of application workloads.

AWS computing management:

The amazon cloud platform provides a wide range of cloud resources such as running a virtual server, store files, share digital media, deploy a website, hosting a website, run a database, analyze the data. In order to understand different techniques and methods [19] provides information about the cost management and implementation ideas in AWS platform. In this research, the architecture of Cassandra is deployed in different configurations of EC2 instances such as t2 micro, t2 small, t2 medium. These instances are launched in Frankfurt data centers since the pricing of the instances vary depending on the geographical location of the data centers.

4 METHODOLOGY

The main aim of the present study is to evaluate the performance of Cassandra virtual nodes implemented in amazon EC2 instances when benchmarked with YCSB tool. In order to achieve this, initially, a literature review is conducted which is followed by the experiment. On conducting a literature review various performance parameters are identified in order to conduct the experiment.

Motivation behind the selection of research method

In order to identify and formulate the idea of any concept, a literature review has to be conducted for obtaining the appropriate literature [32]**.** Hence a literature review is conducted in the initial stages of the present research work. Any research cannot be conducted without framing the previous related research works [33], so literature review has been chosen to investigate and gain knowledge about the earlier researches. In this study literature review has been conducted in order to learn about the behavior of Cassandra database management system and the metrics to evaluate its performance.

Experiment is the method where it investigates few variables and the ways in which they are affected in experimental conditions [33]. Experiments are often conducted for evaluating a model of any system and running simulations to see how the model is affected by a different variable [34]. As this study deals with the evaluation of performance of Cassandra in AWS, the metrics throughput and CPU utilization are used to evaluate the performance. These metrics evaluate how AWS is affected by them in experimental condition and therefore experiment is chosen as the research method for this study.

Exclusion of alternative methods:

Surveys are conducted when a method or a tool already exists and further their implementation is based on the decision of the practitioners [35]. The survey results can be benefited in opinion polls and market research, but cannot help in selecting a particular model. In this paper, the survey is not chosen as a research method, since the results of the performance and behavior of Cassandra database management system cannot be defined by any sample.

Case-study is conducted in order to investigate a particular entity within a given time constraint and an organization [35]. The process of case-study is appropriate for evaluation of methods and tools in any industry to prevent scale-up issues. In this study a real time scenario is considered to generalize the results to all the cloud service providers and not to a particular organization, therefore case study was not considered as the research method.

4.1 Literature review

A literature review is a summary and critical analysis of the relevant existing research literature on a particular topic which is being studied [36]. Literature review on a prior is very important in any academic project [37]. It gathers information from many sources on a particular subject [36]. The related work and the research gap can be identified by conducting a literature review. In the present study, literature review has been conducted by following these steps [36].

- **STEP 1**: Selection of search strings

The first step was to identify the search strings in the current research area. Few keywords as “NoSQL”, “cloud computing”, “AWS”, “Cassandra”, “YCSB”, were identified based on the topic. Furthermore, keywords were explored by improving the search strings which have been obtained previously.

- **STEP 2**: Searching and gathering the literature

The search strings obtained were used to search relevant articles, journals, and books in various databases such as Google Scholar, INSPEC, IEEE Xplore, etc. Google Scholar has a large number of articles, journals, and information and INSPEC was used as it contains controlled vocabulary. The necessary information was gathered from the sources and new keywords were formed based on the results obtained. These new keywords were again used as search strings to explore more on the current research study.

- **STEP 3**: Reading and analyzing the literature

The information obtained in the previous step was carefully read. The information was considered relevant by reading the titles and abstracts of that particular articles and journals. After reading the information it was analyzed more carefully and if the information was relevant it was further taken to write the review.

- **STEP 4**: Writing the review

The information which was considered relevant and important was taken from the sources and was written as related work and background of the current research study.

- **STEP 5**: References

Finally, the literature review has been concluded with the complete bibliographical list of articles, journals, books and its authors.

Forward and backward snowballing method was also used to find in depth information from the relevant research works [38]

4.2 Experiment

4.2.1 Environment

AWS console is a simple web-based user interface which enables to quickly view the resources and can modify the rationally created isolated networks.[19] [6]. AWS provides EC2 instances of different configuration such as T2, M4, M3, C4, C3, X1, R4, R3, P2, G2, F1, I2 and D2 each of which is subcategorized based on vCPU’s, Memory (GiB), storage (GB) [6]. In this research, the environment is setup in T2 micro, small and medium instances which have sufficient requirements as shown in table 2.

Motivation: In order to access the data from different locations, cloud computing platform enables the user to access the applications from anywhere in the world through internet connectivity [13]**.** Cloud computing enables to compute application through various configurations of hardware infrastructure [39]. In order to handle the

huge data through cloud computing, the user needs a database management system. There exist several providers of cloud computing technologies which offer numerous cloud resources such as virtual machine instances, containers, for computations and data storage resources. Amazon web services (AWS), Microsoft azure, google cloud are some of the cloud platform technologies available in the present time. Amazon web services are widely used in today's market providing numerous cloud resources with an approximate market share of 30%, exceptionally good for infrastructure as a service (IaaS) [39].

VM type	T2.Micro	T2.small	T2.medium
vCPU	1	1	2
Mem(GiB)	1	2	4
Operating System	Ubuntu 14.04	Ubuntu 14.04	Ubuntu 14.04

Table 2:EC2 instances configurations

4.2.2 Setup

This section depicts the detailed explanation of how the cloud architecture is deployed on AWS servers in order to conduct the experiment. In the present research, Cassandra is deployed in AWS cloud servers. The experimental setup is carried out in a cloud based environment of AWS with the following requirements:

- **Creating a Virtual Private Cloud (VPC):**

In AWS, a virtual private network using VPC is configured in order to send and receive packets from the localhost. In the experiment conducted, the EC2 instances are setup in the default "eu-central-1a" with a network tenancy assigned as both public and private addresses. The network inbounds and outbound rules are configured in order to allow the flow of traffic.

- **Creating a subnet:**

Each virtual private network allows the user to create two types of subnets: private and public subnet. The experimental setup is deployed under public subnet as we can connect through public IP address. In the present research, the EC2 instances are launched in the default subnet, which has 4093 IP addresses.

- **Launching an EC2 instance:**

The specification of EC2 instance must be selected which is appropriate for the requirement of the experiment. In the present research, T2 instances micro (1vCPU,1GiB memory), small (1vCPU, 2GiB memory) and medium (2vCPU, 4GiB memory) are selected. The instances of the above configuration are launched under same subnet and security group.

4.2.2.1 Apache Cassandra

The T2 instance that is created must be installed with Java, Python, and Java Native Access. Since Apache Cassandra is a Java based NoSQL database management

system, OpenJDK 8 is installed in each instance. The instances must also be installed with the latest version of Python, which helps for the monitoring tools.

After the deployment of Cassandra in AWS cloud service providers, Cassandra 2.2 is installed from the datastax community after updating the ubuntu14-04 packages must be updated [40]. These packages must be installed using command line interface since EC2 instance cannot access the user to use any web browser. Once the installation is completed, the data must be cleared after stopping Cassandra as the Debian packages start the Cassandra service automatically.

Motivation: Unlike SQL which is based on Relational database management system (RDBMS) and other NoSQL databases such as MongoDB, HBase, Cassandra database management system has the ability to perform tremendously over a large amount of data [16]**.** The following are some of the essential aspects because of which Cassandra has been selected for this research:

- **Efficient architecture:** The masterless architecture of Cassandra is built in a way that it delivers continuous availability of data in order to establish reliability and robustness [41]**.**
- **Rapid linear-scalable performance:** The performance of Cassandra is not affected even when the data is added. The scalability component of Cassandra enables to the distribution of the data across all the nodes and performed consistently [42].
- **No single point of failure:** The data can be replicated across all the nodes spread in different datacenters. It features high data availability even if failure of the node occurs [5].
- **Adequate data modeling techniques to store data:** The data modeling techniques which are used in Cassandra are effective in handling the old data that can be achieved by triggering the data using compaction factor [40].

4.2.2.2 Cluster formation

In order to form a cluster, Cassandra must be installed in all EC2 instances. All the nodes must be verified whether the Cassandra status is working or not. The IP addresses of all the Cassandra nodes are listed and out of which only one node is used as a seed node in order to make the experiment easy. Before creating a cluster, all the Cassandra process must be stopped in all nodes. The YAML file of Cassandra in all nodes must be configured for creating the cluster [42]. The following are the parameters that are to be modified in YAML file of each Cassandra node.

- Seed: IP address of the seed node
- Listen address: IP address of the respective node
- RPC address: 0.0.0.0
- broadcast address: IP address of the respective node
- endpoint snitch: GossipingPropertyFileSnitch

In order to verify whether the cluster is formed or not we verify the list of all IP addresses of the nodes in the cluster through node tool utility [43].

4.2.2.3 Creating a keyspace

A keyspace can be created with parameters such as keyspace name, replica placement strategy, replication factor and size of the column. The replication factor enables to create multiple copies of keyspace data which is stored in the cluster. In this

research, a keyspace is created with a replication factor of three and with simple strategy class. In the created keyspace table with ten fields is created.

4.2.2.4 YCSB

In this experiment, YCSB is used to generate read write requests for the Cassandra cluster. YCSB tool comes with several configuring parameters such as thread count, operation count, record count which can be specified for different workloads. In order to benchmark with YCSB, it is installed in a new EC2 instance. The table 3 shows different types of workloads which are predefined in YCSB.

Workloads	Operations ratio
A- update heavy	Read 50% writes 50%
B- read heavy	Read 95% write 5%
C- read only	Read 100%
D- read latest	Read 95% insert 5%
E- short ranges	Scan 95% insert 5%

Table 3:YCSB workload description

Motivation: YCSB framework supports extensibility; i.e. the availability of workloads in open source such that the developers can evaluate different database management systems [25]**.** The YCSB client grants the user to vary the throughput (ops/sec) through command line utility by changing several parameters in the workload.

4.2.3 Experiment design

In order to run different workloads of YCSB, the following steps are followed in this research: [26]

- **Setup the database system to test:**

In this step, the database system which we are supposed to be tested must be setup. This research deals with the testing of Cassandra multi-node cluster. A keyspace must be created which stores the records. In this experiment, a keyspace is created with ten records. The tables are created depending on the workload because the YCSB client which has to be run will not request to create the table automatically [26]. For the core workloads, the YCSB client is designed in a way that there exists a table. The database interface layer will receive the requests for writing or reading the records in the table and translate them into requests for the actual storage which we have allocated.

- **Selecting the appropriate workload:**

During the loading phase, workload describes the data which has to be loaded into the database and during the transaction phase the operations are executed. A workload is combined with a workload java class and a parameter file. The workload class is loaded dynamically by the client from the parameters file and then followed by the execution of the workload [22].

The core workload is a set of standard workloads which are distributed with YCSB and are used directly. The core workloads are categorized as workload A, B, C D, E, F as mentioned in background section. In this research, we have benchmarked Cassandra using workload A, B, and C.

- **Selecting the suitable run time parameters:**

The default workload class and the parameter file describe a particular workload; the user can add modify the parameters of the workload as per the requirement for a particular run of the benchmark [26]. The following are some of the parameters that are modified for a long run of the workload.

- **Threads**: The number of client threads: YCSB client provides to increase the number of threads which increases the amount of the load given to the selected database.
- **Target**: The target number of operations per second: the YCSB client can perform as many as operations we specify. E.g. If each operation takes 10000 milliseconds, then the YCSB client performs 1000 operations per second per working thread [25].
- **Status**: While running a workload, the client can report the status to convince the user that there does not occur any crash. In the command line, if we specify -s while running any workload, the YCSB client reports the status for every 10 seconds.

- **Loading the data:**

In this phase, we define the data which has to be inserted. In order to load the data, we run the YCSB client and execute the loading phase. In the loading phase, we configure using additional parameters to the workload file apart from its default values. In this research, to run a benchmark, the properties such as record count, operation count, and field length is increased based on the requirements.

Alternatively, to achieve the maximum throughput for a particular workload in a particular VM type, the thread count and operation count are increased gradually with the initial ratio as 1:10000. After the thread value reaches the threshold for that particular VM type, the throughput will decrease. So, for each VM type and workload maximum throughput is calculated by using this method.

- **Execute the workload:**

Once the data is loaded, workload is executed. This is done by calling the transaction phase of the workload [7]. The run parameter tells the YCSB client to use the transaction phase. The workloads are attached with additional parameters such as threads and target. At the end of the run process, the YCSB client will report the statistics containing several output values such as runtime(ms), throughput (ops/sec), operations, average, minimum, maximum,95th and 99th percentile latencies for each operation [26].

Independent variables: types of workloads (A, B and C), types of EC2 instances (t2 micro, t2 small and t2 medium), number of nodes in a cluster (3 and 6).

Dependent variables: throughput (ops/sec) and CPU utilization.

This experiment was conducted for ten times to obtain the accurate average value in two scenarios:

SCENARIO 1:

In this scenario, the experiment was performed for three nodes. The three virtual configurations selected are t2micro, t2 small and t2 medium. The results are sampled

for workload A, workload B, and workload C. before running a specific workload, each workload file is configured with the following parameters:

- record count = 10000
- operation count = 100000
- field length = 1000

Initially the experiment was conducted with the default parameters (record count= 100 operation count = 1000 field length = 100). The load lasted for a very short duration which resulted in negligible difference in CPU utilization and the number of operations runs per second for different workloads. In order to generate heavy workload for analyzing the real time scenario, the workload parameters are increased to record count = 10000, operation count = 100000, field length = 1000. On increasing the values of the workload parameters for more than the threshold resulted in the abnormal outcome of CPU utilization and throughput.

SCENARIO 2:

Similarly, the experiment was conducted for six node cluster which was considered as scenario 2.

4.2.4 Metrics

In order to identify the behavior of the Cassandra virtual nodes, we calculate CPU utilization and throughput for the different benchmarking workloads of YCSB.

CPU utilization [30] provides the percentage of CPU utilized by the Cassandra processes when stressed with heavy read-write operations. In this research, “sar” utilities are used to measure the CPU utilization. The sar command is a part of sysstat package which can observe and monitor the CPU utilization for longer period of time. The CPU utilization is monitored during the running process of a workload generated from YCSB client. The output of the sar command is saved in a CSV file for further data interpretation.

YCSB client gives the details about the workload which has been run. This information includes throughput achieved for the assigned workload which is considered for this experiment. The workload is made run for 10times to get the accurate value of the throughput.

4.2.5 Adaptation model

In order to identify the optimal configuration of the data centre while running multiple instances of Cassandra data centre can be obtained using an adaptation model which is defined such that the specific throughput requirements are met. The adaptation model was proposed by Emiliano Casalicchio, lars lundberg and sogand shirinab in [44]. The adaptation model was defined for the optimization problem which provides optimal (or suboptimal) adaptation approach [44][44]. The solution for each user specifies the following parameters:

- The size of the Cassandra virtual datacentres in respect to the number of virtual nodes [44].
- The configuration of the virtual nodes in respect to the number of CPU capacity and memory size [44].
- The placement of the virtual nodes on the physical foundation [44].

The adaptation model sub defined with the following models to obtain the optimization problem:

- Workload and SLA model
- Architecture model
- Throughput model

Workload and service level agreement model:

In this model the workload of a Cassandra virtual datacentre can be described by the following aspects: the different types of request such as read only, write only, reads and writes, scan or multiple combinations of them; the rate of operation requests, the dataset size; and the data replication factor [45]. this workload model is designed based on the following assumptions:

Assumption 1: the system workload exists with a set L of read (R), write (W) and reads & writes operations. It can be represented as:[44]

$$L= \{R, W, RW\}$$

The operation requests are generated by N independent applications and we can consider independent applications. Also we consider that the i application generates the requests of type li ∈ L. and suppose $li = R$ or $li = W$ have 100% read or write requests. [44]

Assumption 2: operation requests of type li are created at a given rate which is measured in operations per second. [44]

Assumption 3: the size of the data set for the application i is r_i GB and can be replicated by the replication factor D_i. [44]

Assumption 4: the workload is only CPU constrained and hence the memory requirements are satisfied. [44]

In accordance to the above assumptions the service level agreement for the user i can be structured in a tuple: [44]

$$\{l_i, T_i^{min}, D_i, r_i\}$$

which contains the information on the agreed workload (l_i *and* r_i) and the service level objectives (T_i^{min} *and* D_i) is the minimum throughput which the service provider must assure to process the request from the application i. Considering the assumption 1 the study is limited to the set L={R, W, RW} [44]

The assumption 4 indicates that the service provider has to set up while the application on running phase and control at the run time, specific number of virtual nodes for tenant I [44].

Architectural model:

In the architecture model, we consider a datacentre which comprises of H homogeneous physical machines which are installed in the same geographical location and a set of V virtual machine configurations [44]. Considering the table which is described for the three different VM configurations. Each Cassandra virtual node runs on the VM of type j and a Cassandra virtual datacentre is comprised on $n_i \geq D_i$ and minimum of D_i out of n_i virtual nodes should run on different physical machines [44]**.**

The configuration of the data centre which is running on N independent variable applications can be represented as the vector x = [$x_{i,j,h}$] in which $x_{i,j,h}$ refers to the number of Cassandra virtual nodes serving applications *i* and with virtual machine configurations *j* assigned on physical machine *h* [44].

Considering each physical machine h has a nominal CPU capacity C_h, calculated in number of available cores and the memory of M_h GB. The virtual machine of type j is configured with c_j virtual cores, m_j GB of memory and a maximum $heapsize_j$ GB [44]. The heap size determines the size of the data any Cassandra virtual nodes which can store the main memory for rapid retrieval and processing. Finally, in order to make CPU bound the virtual data centre initiated for an application i is required a n_i number of nodes can be represented as follows:

$$n_i \geq D_i \cdot \frac{r_i}{heapSize_j}, \ \forall j \in \mathcal{J}$$

For $r_i > heapsize_j$,the above equation is constrained otherwise considering the number n_i of the virtual nodes can be defined as

$$n_i = \mathrm{X}\, x_{i,j,h} \ \forall i \in \mathrm{I}.\ j \in \mathrm{J}, h \in \mathrm{H}$$

And by considering the experimental case is always $r_i \geq heapsize_j$ for the configurations j, the above constraints are represented using the following equations:

$$\sum_{j \in \mathcal{J}, h \in \mathcal{H}} x_{i,j,h} \geq D_i \cdot \frac{r_i}{heapSize_j}$$

$$\sum_{j \in \mathcal{J}} y_{i,j} = 1$$

$$\sum_{h \in \mathcal{H}} s_{i,h} \geq D_i$$

From the above equations represented $y_{i,j} = 1$ if the application *i* implements a virtual machine configuration *j* to run the Cassandra virtual nodes and alternatively $y_{i,j} = 0$; $s_{i,h} = 0$.

Throughput model:

The throughput model is designed as per the actual throughput T_i which is offered to the application *i* by the provider in the function of $x_{i,j,h}$ [44]

For the CPU bound workload, the throughput for a particular Cassandra virtual datacentre serving requests of the type l_i and running on a specific virtual machine type *j* can be represented with the linear segment with the slope $\delta^k_{l_i,j}$. finally, we represent the expression for $nk_{-1} \leq n_i \leq n_k$ as follows:

$$t(n_i) = t(n_{k-1}) + t^0_{l_i,j} \cdot \delta^k_{l_i,j} \cdot (n_i - n_{k-1}) \qquad \text{-(6)}$$

Where $k \geq 1$, $n_0 = 1$ and $t_{i,j,h}(1) = t^0_{l_i,j}$ and finally the overall throughput T_i is defined as follows:

$$T_i(\mathbf{x}) = t(n_i), \ \forall i \in \mathrm{I} \qquad \text{- (7)}$$

Power consumption model:

This model is to implement the balanced usage of the resources which results in optimal power consumption. The power consumption models usually represent a linear relationship between the total amount of the power consumed by the system in the function of the CPU utilization [44]. This model uses a linear model where the power represented as P_h used by a specific physical machine h is a function of CPU utilization and of the system configuration x:

$$P_h(\mathbf{x}) = k_h \cdot P_h^{max} + (1 - k_h) \cdot P_h^{max} \cdot U_h(\mathbf{x}) \quad \text{-(8)}$$

From the above equation P_h^{max} represents the maximum power consumed for the physical machine h when it is completely consumed, k_h represents the fraction of the power which is consumed by the idle physical machine h [44]. the CPU utilization for the physical machine h can be represented as

$$U_h(\mathbf{x}) = \frac{1}{C_h} \cdot \sum_{\mathcal{I},\mathcal{J}} x_{i,j,h} \cdot c_j \qquad (9)$$

Optimization problem:

The goal of the service provider is to optimize the overall energy consumption P(x) which is represented as

$$P(\mathbf{x}) = \sum_{h \in \mathcal{H}} P_h(\mathbf{x}) = \sum_{h \in \mathcal{H}} P_h^{max} \left(k_h \cdot r_h + \frac{(1 - k_h)}{C_h} \sum_{\mathcal{I},\mathcal{J}} x_{i,j,h} \cdot c_j \right) \qquad (10)$$

Where $r_h = 1$ if $x_{i,j,h} > 0$ for some $i \in I$ and $j \in J$.

Hence the optimal adaptation plan is described as an instance of x solution of the optimization problem min f(x) = P(x) represented as:

$$\sum_{\mathcal{J},\mathcal{H}} t(x_{i,j,h}) \geq T_i^{min},\ \forall i \in \mathcal{I} \quad (11)$$

$$\sum_{\mathcal{H}} x_{i,j,h} - \Gamma \cdot y_{i,j} \geq \frac{D_i \cdot r_i}{heapSize_j} - \Gamma,\ \forall i \in \mathcal{I}, j \in \mathcal{J} \quad (12)$$

$$x_{i,j,h} \leq \Gamma \cdot y_{i,j},\ \forall i \in \mathcal{I}, j \in \mathcal{J}, h \in \mathcal{H} \quad (13)$$

$$\sum_{\mathcal{J}} y_{i,j} = 1,\ \forall i \in \mathcal{I} \quad (14)$$

$$\sum_{\mathcal{I},\mathcal{J}} x_{i,j,h} \cdot c_j \leq C_h,\ \forall\ h \in \mathcal{H} \quad (15)$$

$$\sum_{\mathcal{I},\mathcal{J}} x_{i,j,h} \cdot m_j \leq M_h,\ \forall\ h \in \mathcal{H} \quad (16)$$

$$\sum_{\mathcal{H}} s_{i,h} \geq D_i,\ \forall i \in \mathcal{I} \quad (17)$$

$$\sum_{\mathcal{J}} x_{i,j,h} - s_{i,h} \cdot \Gamma \leq 0,\ \forall h \in \mathcal{H} \quad (18)$$

$$-\sum_{\mathcal{J}} x_{i,j,h} + s_{i,h} \leq 0,\ \forall h \in \mathcal{H} \quad (19)$$

$$\sum_{\mathcal{I}} s_{i,h} - r_h \cdot \Gamma \leq 0,\ \forall h \in \mathcal{H} \quad (20)$$

$$-\sum_{\mathcal{I}} s_{i,h} + r_h \leq 0,\ \forall h \in \mathcal{H} \quad (21)$$

From the above equations

EQ 11. Assures that the Service Level Agreement (SLA) is satisfied in the point of minimum throughput for all users [44].

EQ 12: provides set of constraints in order to assure the number of virtual nodes are placed are sufficient to handle the dataset by each virtual node with the replication factor D_i.[44].

EQ 13 & 14: represents an assumption that each user is allocated with similar virtual machines [44].

EQ 15: commands the highest capacity of the physical machine is not surpassed [44].

EQ 16: manages the memory which is allocated for the virtual nodes is not exceeding the memory size of the physical nodes [44].

EQ 17: assures the cassandra virtual nodes are instantiated on different physical machines D_i.[44].

EQ 18 & 19: makes the value of $s_{i,j}$ to be 1 when the physical machine is used by any application i and set to 0 for its opposite contrast [44].

EQ 20 & 21: sets the value of $s_{i,j}$ to 1 if any physical machine is used and set to 0 if not used [44].

EQ 22 and 23: represents the structural constraints of the optimization problem [44].

Using the above adaptation model the optimal or the suboptimal solution when running multiple instances of Cassandra is calculated from the obtained values of throughput in different EC2 instances. Mathematically the procedure was carried out using MATLAB.

5 RESULTS

5.1 Scenario 1

After conducting the experiment, the results were obtained for three node cluster. The values were recorded for each workload (A, B, C) of the configurations micro, small, and medium instances of T2. Workloads are initially configured with 10 threads and 1,00,000 operation count. The average of the obtained values of CPU utilization and throughput has been calculated.

5.1.1 Workload A

The workload A which is a combination of 50% reads and 50% writes, was calculated for t2micro, t2small, and t2 medium. The obtained values of CPU utilization and throughput are tabulated in Table 4. The CPU utilization throughout the process of loading and running of workload A is shown in the figure 5.

Workload A	T2 micro	T2 small	T2 medium
CPU utilization (%)	71.5	50	48.75
Throughput (ops/sec)	2795	3099	3868

Table 4: CPU utilization and Throughput of Workload-A for 3 node cluster

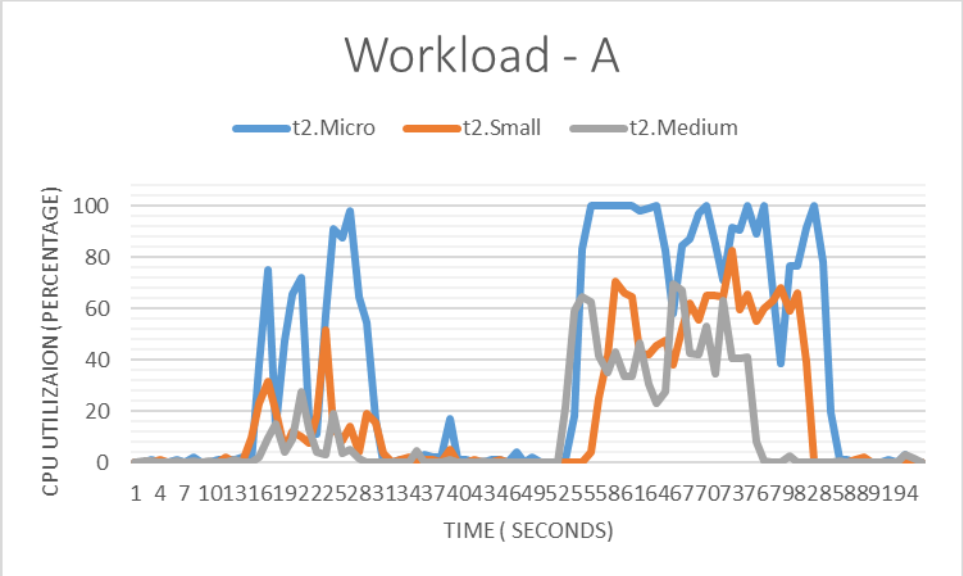

Figure 5:CPU utilization of Workload-A for 3 node cluster

From the Figure 4, we can analyse that CPU utilization has reached 100% for most of the time and value decreased by increasing the VM size i.e, micro, small and medium. Later the intensity of the workload is increased step by step to get the maximum possible throughput. By increasing the thread count and operations count proportionally from the initial values, maximum throughput is achieved at 20threads for t2.micro, 100threads for t2.small and t2.medium as shown in table 5.

Workload - A	T2 Micro	T2 Small	T2 Medium
Throughput (operations/second)	2795	6364	10336
Thread count	20	100	100

Table 5:Maximum Throughput of Workload-A for 3 node cluster

5.1.2 Workload B:

Similarly, the values for workload B which is a combination of 95%reads and 5% writes were calculated and presented in Table 6. However, maximum CPU utilization has reached 90% for workload B and the overall values are shown in the figure 6 for the different virtual configurations.

Workload B	T2 Micro	T2 Small	T2 Medium
CPU utilization (%)	39.1	28.8	14.19
Throughput (ops/sec)	3279	3437	3692

Table 6:CPU utilization and Throughput of Workload-B for 3 node cluster

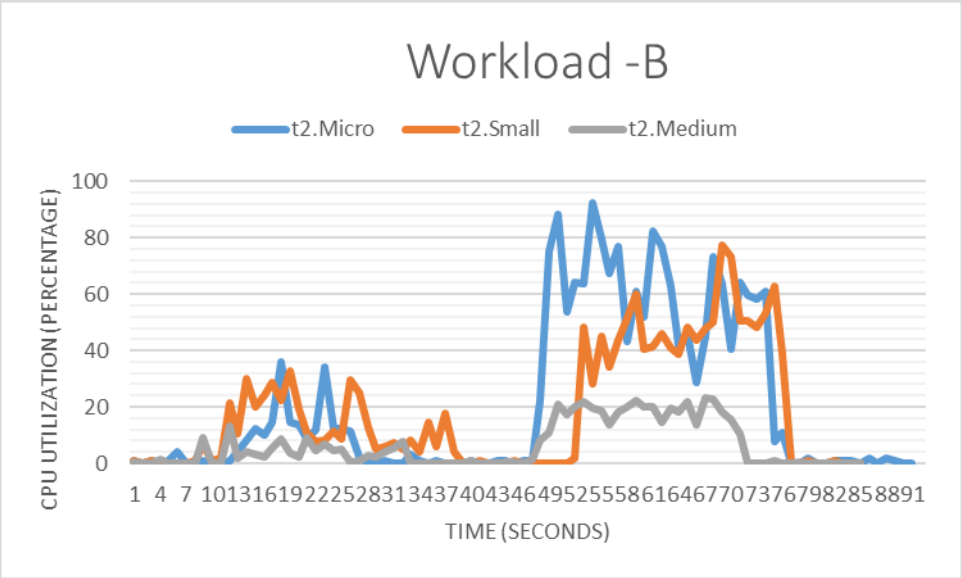

Figure 6:CPU utilization of Workload-B for 3 node cluster

From the graph in Figure 6, we can analyze that CPU utilization has decreased gradually by increasing the size of the VM.

The maximum throughput for workload B in a 3 node cluster is achieved by increasing the intensity of the workload. The obtained values are analyzed for the corresponding thread count, averaged and are tabulated in table 7.

Workload - B	T2 Micro	T2 Small	T2 Medium
Throughput (operations/second)	3333	8357	10336
Thread count	20	100	100

Table 7:Maximum Throughput of Workload-B for 3 node cluster

5.1.3 Workload C

In this scenario, the workload C which is 100% read was calculated. The values obtained are shown in table 8:

Workload C	T2 Micro	T2 Small	T2 Medium
CPU utilization (%)	33.6	25.61	12.4
Throughput (ops/sec)	3450	3544	3730

Table 8:CPU utilization and Throughput of Workload-C for 3 node cluster

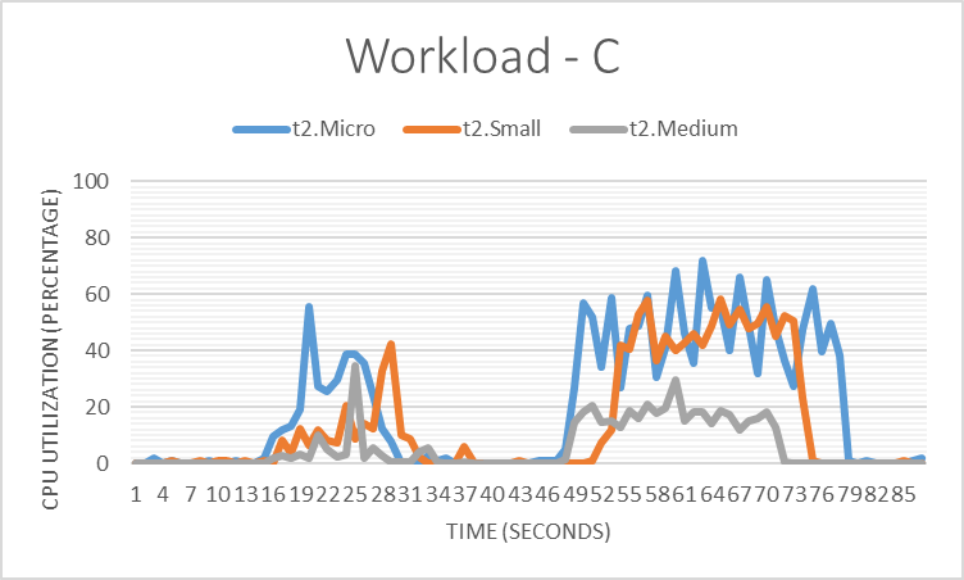

Figure 7: CPU utilization of Workload-C for 3 node cluster

From the graph in Figure 7, the maximum CPU utilization achieved for workload C is around 70% for t2 micro instance. By changing the type of VM with higher memory storage decreased the CPU utilization gradually. To obtain the maximum throughput for a particular VM type and workload, the thread count is increased gradually by maintaining the ratio between operation count. The achieved throughput for different VM types is shown in table 9.

Workload C	T2 Micro	T2 Small	T2 Medium
Throughput (operations/second)	3450	8709	9694
Thread count	20	100	100

Table 9:Maximum Throughput of Workload-C for 3 node cluster

5.2 Scenario 2

The experiment is now extended to a six-node cluster by adding three new similar nodes to the previously existing setup. The values were recorded for workloads (A, B, C) of t2 micro, t2small, and t2 medium. The workload parameters are initially configured as 10 threads, 100000 operation count and 10000 record count.

5.2.1 Workload A:

The workload A is a combination of 50% reads and 50% writes. The CPU utilization and throughput are tabulated with the mean value of the obtained results shown in table 10.

Workload A	T2 Micro	T2 Small	T2 Medium
CPU utilization (%)	53	45.5	28.2
Throughput (ops/sec)	3028	2469	3288

Table 10: CPU utilization and Throughput of Workload-A for 6 node cluster

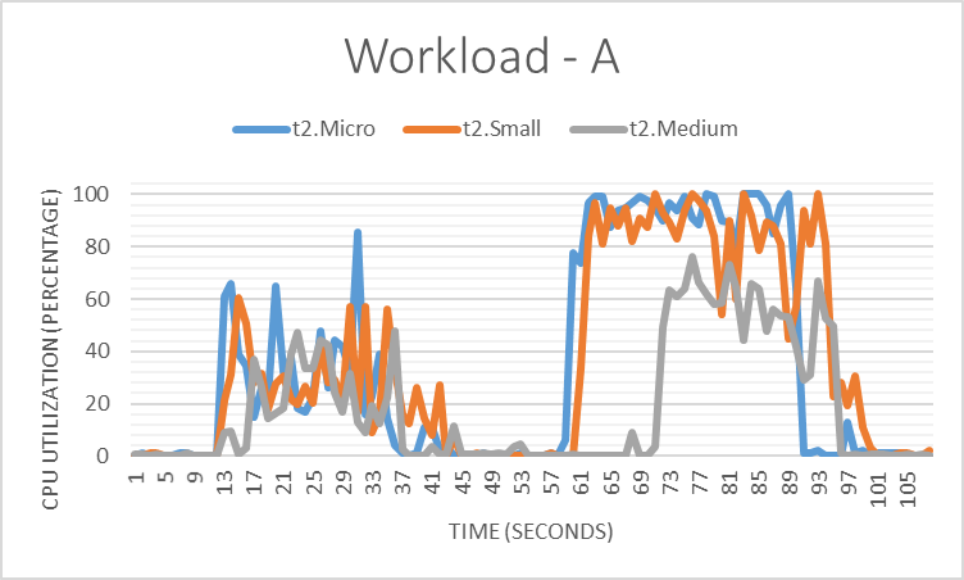

Figure 8: Maximum Throughput of Workload-A for 6 node cluster

.

From the figure 8, the CPU utilization has reached 100% for both the VM types, micro and small instances of T2, which shows higher utilization when compared to 3 node cluster. The best possible throughput is achieved at 100 threads for t2. micro and 100threads for small and medium instances of T2 which are present in the table 11.

Workload - A	T2 Micro	T2 Small	T2 Medium
Throughput (operations/second)	7899	11239	19186
Thread count	100	200	200

Table 11:Maximum Throughput of Workload-A for 6node cluster

5.2.2 Workload B

The workload B, which is a combination of 95% reads and 5% writes, CPU utilization and throughput for initial configuration are calculated and the results are tabulated below in table12:

Workload B	T2 micro	T2 small	T2 medium
CPU utilization (%)	19.8	16.7	9.2
Throughput (ops/sec)	3412	3396	3626

Table 12:CPU utilization and Throughput of Workload-B for 6 node cluster

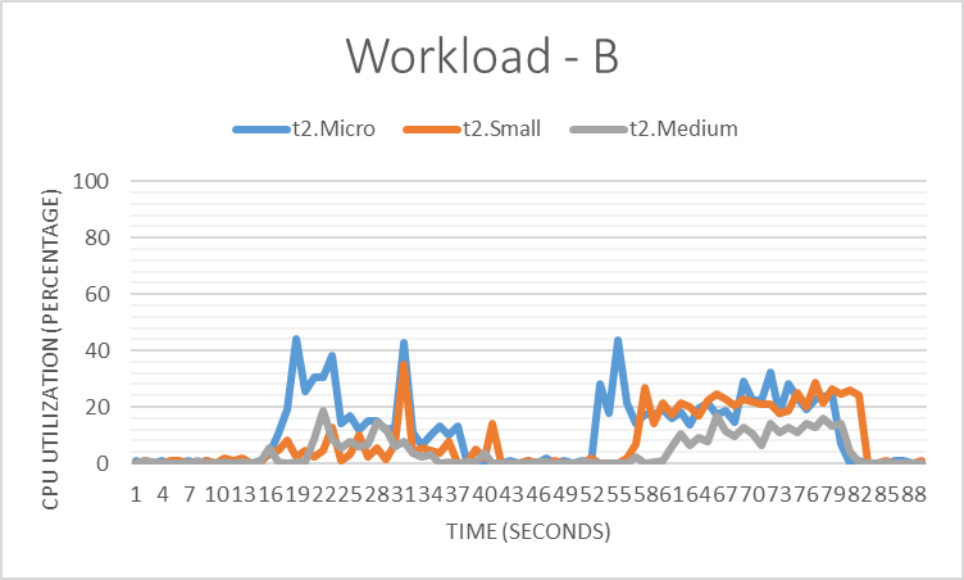

Figure 9:CPU utilization of Workload-B for 6 node cluster

In the figure 9, shows the CPU utilization for different types of T2 instances throughput the loading and running process of Workload-B. Also, the parameters configured for achieving maximum throughput are shown in the below table 13.

Workload - B	T2 Micro	T2 Small	T2 Medium
Throughput (operations/second)	5251	8714	10894
Thread count	50	50	100

Table 13:Maximum Throughput of Workload-B for 6 node cluster

5.2.3 Workload C:

Similarly, for workload C, which is 100% read, CPU utilization and throughput is analyzed and the results are tabulated in table 14:

Workload C	T2 micro	T2 small	T2 medium
CPU utilization (%)	17.1	11.05	8.3
Throughput (ops/sec)	3886	2912	3649

Table 14:CPU utilization and Throughput of Workload-C for 6 node cluster

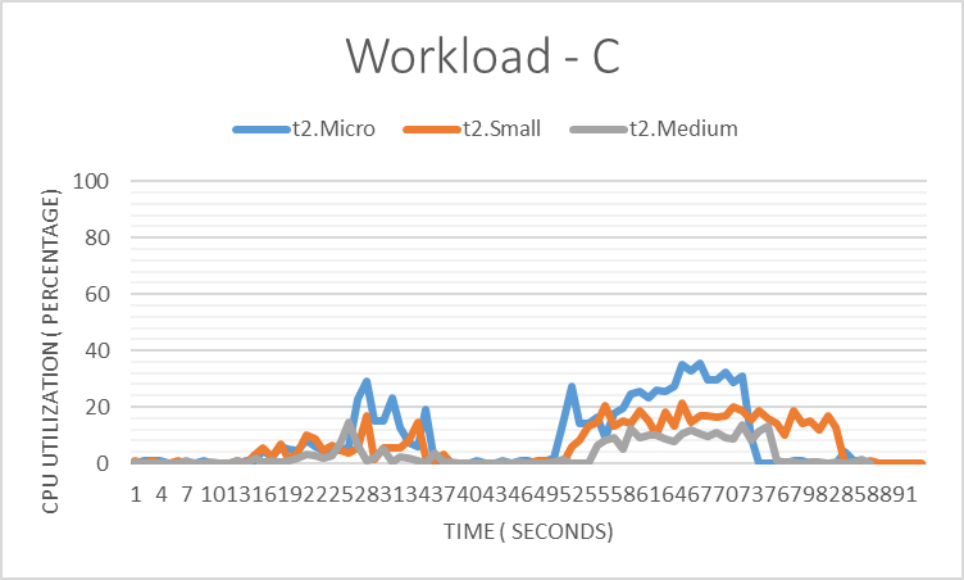

Figure 10:CPU utilization of Workload-C for 6 node cluster

In the Figure 10 CPU utilization is low throughout the process and reached a maximum value of 35%. The maximum throughput achieved for different instances of T2 are presented in below table 15.

Workload - C	**T2.micro**	**T2.small**	**T2.medium**
Throughput (operations/second)	3886	8908	10900
Thread count	20	100	50

Table 15:Maximum Throughput of Workload-C for 6 node cluster

5.3 Adaptation model parameters

5.3.1 Workload and SLA model

From the available results, the maximum throughput and memory available for 3 nodes and 6 node cluster for the corresponding VM types are presented in table 16 and 17.

	VM type and config.			**Throughput for different workloads (ops/sec)**		
j	c_j	m_j	$heapSize_j$	**Workload A**	**Workload B**	**Workload C**
1	1	1	0.5	2795	3279	3450
2	1	2	1	6364	3437	3544
3	2	4	2	10336	3692	3730

Table 16:Memory available for the dataset in a Cassandra for 3 node cluster

	VM type and config.			Throughput for different workloads (ops/sec)		
j	c_j	m_j	$heapSize_j$	**Workload A**	**Workload B**	**Workload C**
1	1	1	0.5	7899	5251	3886
2	1	2	1	11239	8714	8908
3	2	4	2	19186	10894	10900

Table 17:Memory available for the dataset in a Cassandra for 6 node cluster

The following table 18 depicts the model parameters which are used in adaption model.

Parameters	Value	description
N	3	Number of users
V	3	Number of VM types
H	8	Number of Physical machines
D_i	3	Replication factor
L	{R, W, RW}	Types of operations
T_i^{min}	6000-10000 *ops/sec*	Minimum throughput based on the SLA
c_j	1,1,2	Number of virtual cores used by VM type *j*
m_j	1,2,4	Memory used by the VM type *j*
$heapsize_j$	0.5,1,2	Heapsize of VM type *j*

Table 18:Model parameters used in the experiments

6 ANALYSIS AND DISCUSSION

Throughput:

For three node cluster, the average values obtained for different T2 instances are compared in the below figure 11. The variation in the number of operations per second is can be plotted for the workloads A, B and C.

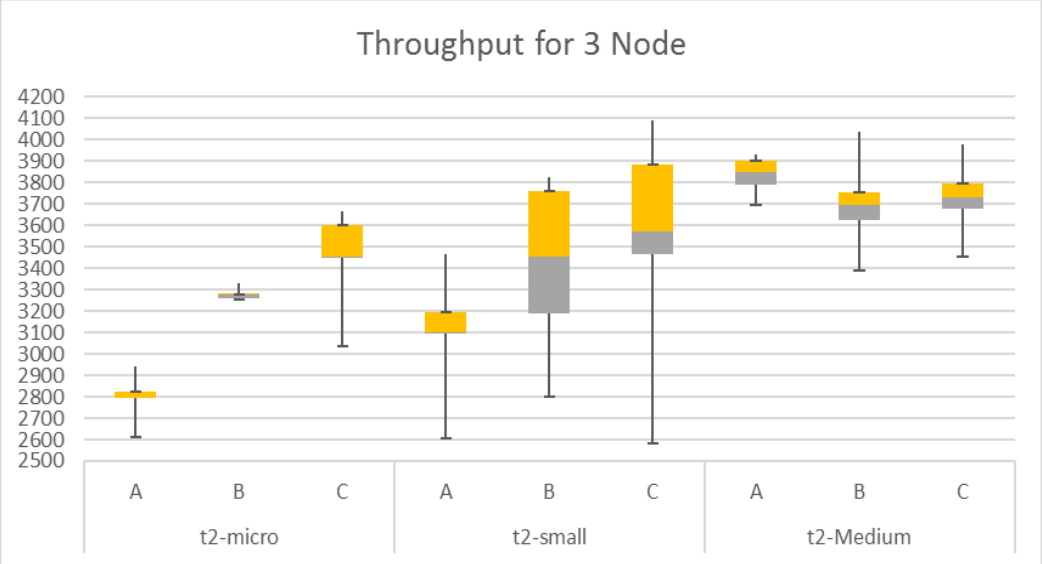

Figure 11: Throughput boxplot for 3 node cluster

The appropriate analysis of the data is performed statistically for the obtained results are performed by calculating the mean, variance and standard deviation depicted in the table 19.

3-node		Workload A	Workload B	Workload C
T2 micro	Mean	2795	3279	3450
	Variance	11429.91	711.84	51084.96
	Standard deviation	106.91	26.68	226.02
T2 small	Mean	3099	3437	3544
	Variance	78729.55	170927.6	284074.9
	Standard deviation	280.59	413.43	532.99
T2 medium	Mean	3868	3692	3730
	Variance	13652.25	46233.57	23759.71
	Standard deviation	116.84	215.02	154.14

Table 19:Statistical data for 3 node cluster

Similarly, for the 6 node cluster, the average values for different t2 instances are compared in the figure 12. The variation in the number of operations per second can be plotted for the workloads A, B and C.

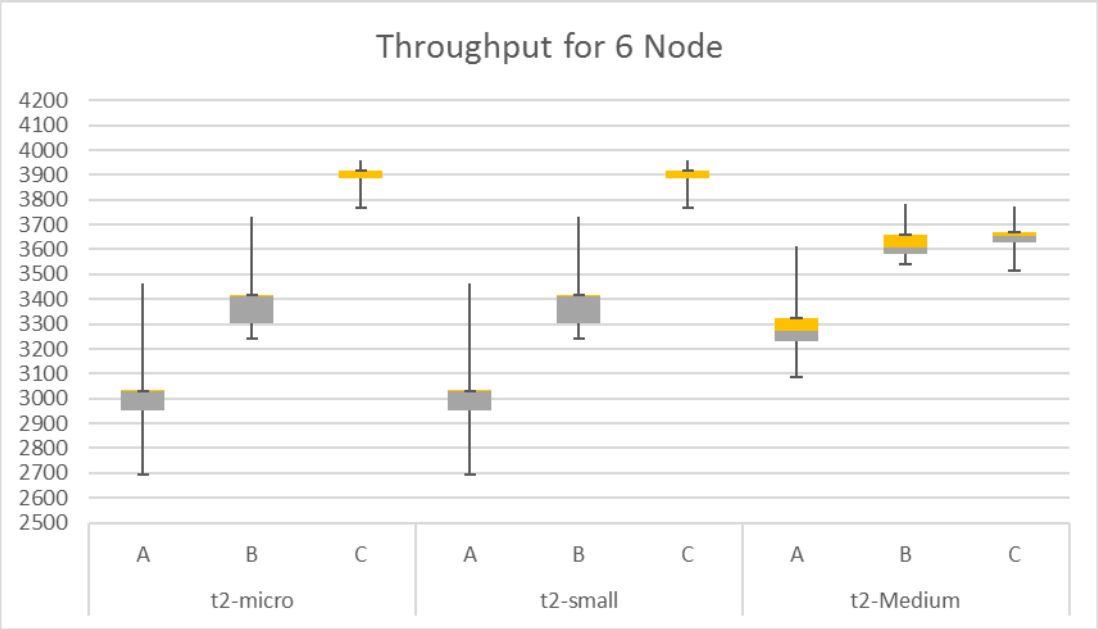

Figure 12:Throughput boxplot for 6 node cluster

The mean, variance and standard deviation obtained for 6-node cluster are tabulated in table 20 for the workloads A, B and C

6-node		Workload A	Workload B	Workload C
T2 micro	Mean	3028.9	3412.8	3885.47
	Variance	61975.91	30747.74	4220.95
	Standard deviation	248.95	175.35	64.97
T2 small	Mean	2469	3396.5	2912.5
	Variance	49990	57722.64	28047.36
	Standard deviation	223.58	240.26	167.47
T2 medium	Mean	3287.84	3626.11	3649.16
	Variance	24044.88	6050.83	4901.43
	Standard deviation	155.06	77.79	70.01

Table 20:Statistical data for 6 node cluster

Similarly, like 3-node cluster, the mean, variance and standard deviation can be calculated for the maximum throughput values attained for 6 node Cassandra cluster in t2 micro, t2 small and t2 medium under workload A, B and C.

6.1 Discussions:

This section explains and compares the conclusions that have been obtained in this study with the findings of the previous research studies. The main goal of this study was to evaluate performance of the NoSQL database Cassandra in AWS cloud service provider environment. The experiment was conducted in AWS cloud environment to evaluate performance of NoSQL database Cassandra using the metrics, throughput and CPU utilization.

Veronika et al. have conducted an experiment to perform evaluation of NoSQL database using YCSB benchmarking tool [11]. In order to understand how scalability affects the execution time, the authors have created three scenarios by forming a cluster with 1, 3 and 6 nodes respectively [12]. The authors have evaluated the scalability by varying the number of threads and data size simultaneously. After conducting the experiment, the authors have stated that scaling the number of nodes doesn't guarantee the performance improvement. But by analyzing the results obtained from the current study shows that CPU utilization decreases and throughput increases with the increase in the number of nodes, which shows that the performance has increased from 3 nodes to 6 node cluster. However, by increasing the number of nodes the performance might not be improved in all scenarios as mentioned in [12]. This is because in the experiment conducted, in t2 medium instance, the workload B and C hasn't shown much increase in throughput from 3 node cluster to 6 node cluster. From [12] and the current study it can be stated that the the performance of Cassandra is unpredictable for each different scenario.

In [7], the authors have studied the scalability characteristics and the Cassandra architecture. For the evaluation of Cassandra scalability, the authors have created various database size and cluster size scenarios (1, 3 and 6 nodes). In this study the authors have evaluated Cassandra's scalability while considering 10,000 operations and different thread variations (1, 6, 60, 600, 1200, 6000). The conclusion which has been stated in [7]is that as the number of requests increases, the performance of Cassandra has improved to a certain limit. In this case it has been explained as when the number of threads are bigger than 600, the performance of the system was not improved. Further the results with 1200 and 6000 threads were considered acceptable but the performance has been worsened when that compared with the threads less than 600. Similarly, in the current study the performance of Cassandra has been improved as the throughput increases with the increase in the number of threads till the threshold point. Further the performance was shown better only to a threshold limit and further it has dropped its performance.

The authors Vishal and Akshay in [28] have conducted an experiment to compare throughput in terms of Transactions per Second (TPS) for heavy write operations in a relational database MySQL and two NoSQL databases, Cassandra and HBase [28]. The conclusion given in [28] is that Cassandra scaled up the most amongst the three databases which is its unique feature. In the current study also Cassandra has shown a better performance in terms of throughput for a 3 node and 6 node cluster irrespective of the write operations up to a certain threshold limit.

6.2 Answers to research questions:

RQ1: What is the throughput achievable by different VMs configuration when running Cassandra virtual nodes and when stressed by a different type of workload (A, B and C)?

Answer: the following table 21 and 22 shows the throughput values of Cassandra virtual nodes for different configurations of workload A, B and C for the record count=10000, operation count=100000 field length=1000

Workload	T2micro	T2small	T2medium
A	2795	3099	3868
B	3279	3437	3692
C	3450	3544	3730

Table 21:Average throughput for different workloads in 3 nodes

Workload	T2micro	T2small	T2medium
A	3028	2469	3288
B	3412	3396	3626
C	3886	2912	3649

Table 22:Average throughput for different workloads in 6 nodes cluster

RQ2: What is the throughput scalability of Apache Cassandra on a virtual environment for the different type of VM configuration and under different workload (A, B and C)?

Answer: For 3node, the maximum throughput achieved when increased the intensity of the workloads are tabulated in table 23

Workload	T2micro	T2small	T2medium
A	2795	6364	10336
B	3333	8357	9791
C	3450	8404	8709

Table 23: Maximum throughput for different Workloads in 3 node cluster

For 6node, the maximum throughput achieved by varying the parameters of thread count and operation count to increase the intensity of the workload are depicted in table 24.

Workload	T2micro	T2small	T2medium
A	7899	11239	19186
B	5251	8764	10894
C	3886	8908	10900

Table 24:Maximum throughput for different Workloads in 6 node cluster

The maximum throughput is calculated in case of throughput scalability because in accordance the service level agreement model (SLA) [44], the optimal solution can be provided to the users of Cassandra with the operations which are less than or equal to the number of maximum operations per second that can be performed under each virtual machine configuration with minimum energy consumption.

RQ3: How can we find the optimal or sub-optimal configuration of a datacentre running multiple instances of Cassandra datacentre in a way that specific throughput requirements are satisfied?

Answer: Based on the results obtained for RQ1 and RQ2 the suboptimal solution mentioned in RQ3 can be addressed. The sub optimal solution can be obtained using the resource management for the data storage systems is a challenging task and complexity increase when multitenancy is considered [44].

Hypothesis: there are three different users of Cassandra instances used and are distributed across 8 physical servers working under three different application types A, B and C.

By using the above adaptation model, to attain the optimal placement of Cassandra virtual nodes across the servers are categorized in the following cases:

Case 1: with the minimum throughput for application A= 8000, B= 6000 and C= 8000 the optimal placement of the Cassandra virtual nodes obtained is represented in the table 25

	Workload A	Workload B	Workload C
Minimum throughput	8000	6000	8000

Table 25: Minimum Throughput for case 1

	T2 micro	T2 small	T2 medium
Workload A	14	0	0
Workload B	4	0	0
Workload C	3	0	0

Table 26:Total virtual nodes for each workload for case 1

	Workload A	Workload B	Workload C
Server 1	0	0	0
Server 2	0	0	0
Server 3	0	0	0
Server 4	0	0	0
Server 5	6	1	1
Server 6	6	1	1
Server 7	0	0	0
Server 8	2	2	2

Table 27:Optimal virtual node placement for case 1

The optimal power consumption obtained =**1247**

Case2: with the minimum throughput for application A= 8000, B= 8000 and C= 10000, the optimal placement of the Cassandra virtual nodes obtained is represented as:

	Workload A	Workload B	Workload C
Minimum throughput	8000	8000	10000

Table 28:Minimum Throughput for case 2

	T2 micro	T2 small	T2 medium
Workload A	14	0	0
Workload B	5	0	0
Workload C	4	0	0

Table 29:Total virtual nodes for each workload for case 3

	Workload A	Workload B	Workload C
Server 1	0	0	0
Server 2	0	0	0
Server 3	6	1	1
Server 4	2	3	2
Server 5	0	0	0
Server 6	6	1	1
Server 7	0	0	0
Server 8	0	0	0

Table 30:Optimal virtual node placement for case 3

The optimal power consumption obtained = **1266**

Case3*:* with the minimum throughput for application A= 10000, B= 10000 and C= 10000, the optimal placement of the Cassandra virtual nodes obtained is represented as:

	Workload A	Workload B	Workload C
Minimum throughput	10000	10000	10000

Table 31:Minimum Throughput for case 3

	T2 micro	T2 small	T2 medium
Workload A	17	0	0
Workload B	6	0	0
Workload C	4	0	0

Table 32:Total virtual nodes for each workload for case 3

	Workload A	Workload B	Workload C
Server 1	0	0	0
Server 2	0	0	0
Server 3	0	0	0
Server 4	7	1	0
Server 5	5	0	1
Server 6	2	0	0
Server 7	0	4	1
Server 8	3	1	2

Table 33:Optimal virtual node placement for case 3

The optimal power consumption obtained = **1653**

Case4: the dataset size is increased to 1 Giga byte and with the minimum throughput for application A= 8000, B= 6000 and C= 8000, the optimal placement of the Cassandra virtual nodes obtained is represented as:

	Workload A	Workload B	Workload C
Minimum throughput	8000	8000	8000

Table 34:Minimum Throughput for case 4

	T2-micro	T2-small	T2-Medium
Workload A	14	0	0
Workload B	6	0	0
Workload C	6	0	0

Table 35: Total virtual nodes for each workload for case 4

	Workload A	Workload B	Workload C
Server 1	7	0	1
Server 2	0	0	0
Server 3	0	0	0
Server 4	6	1	1
Server 5	0	0	0
Server 6	0	1	1
Server 7	0	0	0
Server 8	1	4	3

Table 36:Optimal virtual node placement for case 4

The optimal power consumption obtained = **1920**

The optimal solution is obtained in the above cases for the three types of instances gives the optimal configuration that minimize the energy consumption which is obtained using t2 micro instance. For each cases considered, the optimal solution comprises of the placement of Cassandra virtual nodes with the minimum energy consumption.

6.3 Validity threats

- **Internal validity:**

Internal validity refers to how well the experiment has been conducted and how the treatment has actually caused the outcome [43]. In this study workload and number of nodes are the threats which may change the outcome of the experiment. To mitigate this problem, the experiment has been conducted for different workloads and different number of nodes.

- **External validity:**

External validity is concerned if the results obtained in the experiment can be generalized to the outside the scope of the study [43]. Cloud service providers, database and the stress stool used in this study can be the major threats as they might not be used in the current data centers and which may be outdated. But this threat has been mitigated by using AWS cloud service provider, Cassandra (NoSQL database) and YCSB benchmarking tool. AWS, Cassandra and YCSB are the trending cloud service provider, NoSQL database and workload stress tool respectively which are widely used in the real world scenario, therefore the threat has been mitigated.

- **Construct validity:**

Construct validity is defined as the relation between the theory behind the experiment and its observations [43]. The major threat that could be caused to this study is the throughput which was the metric chosen to evaluate the performance of NoSQL database in AWS. But this threat was mitigated by choosing another metric CPU utilization which has made the study efficient in evaluating performance.

7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

The main aim of this research is to evaluate the performance of Cassandra virtual nodes under three configurations of EC2 instances: t2micro, t2small and t2medium AWS and find the optimal or sub-optimal configuration of a data centre running multiple instances of Cassandra data centre in a way that specific throughput requirements are satisfied. In order to measure the performance, the Cassandra EC2 instances are benchmarked with YCSB. The throughput is calculated under the workloads A (50/50 reads and writes), B (95/5 read writes mix). The values of throughput are calculated for same workloads under t2micro, t2small and t2 medium. In order to understand the scalability of the Cassandra virtual nodes, the number of nodes in the cluster are increased along with the intensity of the workload. To measure the performance of the Cassandra virtual nodes, the setup was benchmarked with YCSB below the workloads A, B and C. the performance of Cassandra EC2 instances are measured by the performance metrics: CPU utilization and throughput (ops/sec). In the later stages, using the adaptation model we identify the optimal or sub-optimal configuration of a datacentre running multiple instances of Cassandra datacentre in a way that specific throughput requirements are satisfied.

This experiment can be extended to the remaining workloads categorized in YCSB for workload D and E [26]**.** The benchmarking tool can be replaced with other available tools such as stress-ng and Cassandra stress tool. Similar experiment can be conducted in different cloud platforms such as Microsoft azure and Google cloud.

REFERENCES

[1] L. Malhotra, D. Agarwal, and A. Jaiswal, "Virtualization in cloud computing," *J Inf. Tech Softw Eng*, vol. 4, no. 136, p. 2, 2014.

[2] K. R. Jackson *et al.*, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 2010, pp. 159–168.

[3] M. Bourguiba, K. Haddadou, I. El Korbi, and G. Pujolle, "Improving network I/O virtualization for cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 673–681, 2014.

[4] A. Cassandra, *Apache cassandra*. 2013.

[5] "Apache Cassandra." [Online].

[6] "Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS," *Amazon Web Services, Inc.* [Online].

[7] 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.

[8] 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.

[9] 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.

[10] J. Varia, "Best practices in architecting cloud applications in the AWS cloud," *Cloud Comput. Princ. Paradig.*, pp. 457–490, 2011.

[11] 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.

[12] V. Abramova, J. Bernardino, and P. Furtado, "Testing cloud benchmark scalability with cassandra," in *2014 IEEE World Congress on Services*, 2014, pp. 434–441.

[13] 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.

[14] W. Chen, Y. L. Zhao, S. Q. Long, and X. L. Xie, "Research on cloud storage security," in *Applied Mechanics and Materials*, 2013, vol. 263, pp. 3068–3072.

[15] "Security Groups for Your VPC - Amazon Virtual Private Cloud." [Online]..

[16] "Amazon VPC Connectivity Options," *Amazon Web Services, Inc.* [Online]..

[17] 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.

[18] K. Sathvik, *Performance Tuning of Big Data Platform : Cassandra Case Study*. 2016.

[19] "Whitepapers – Amazon Web Services (AWS)," *Amazon Web Services, Inc.* [Online].

[20] "VPCs and Subnets - Amazon Virtual Private Cloud." [Online].

[21] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 35–40, 2010.

[22] teddymaef, *Data Caching*. .

[23] M. Brown, *Learning Apache Cassandra*. Packt Publishing Ltd, 2015.

[24] P. Menon, T. Rabl, M. Sadoghi, and H. A. Jacobsen, "CaSSanDra: An SSD boosted key-value store," in *2014 IEEE 30th International Conference on Data Engineering*, 2014, pp. 1162–1167.

[25] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proceedings of the 1st ACM symposium on Cloud computing*, 2010, pp. 143–154.

[26] “brianfrankcooper/YCSB,” *GitHub*. [Online].
[27] M. Mayilvaganan and M. Sabitha, “A cloud-based architecture for Big-Data analytics in smart grid: A proposal,” in *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on*, 2013, pp. 1–4.
[28] V. D. Jogi and A. Sinha, “Performance evaluation of MySQL, Cassandra and HBase for heavy write operation,” in *Recent Advances in Information Technology (RAIT), 2016 3rd International Conference on*, 2016, pp. 586–590.
[29] E. Dede, B. Sendir, P. Kuzlu, J. Hartog, and M. Govindaraju, “An Evaluation of Cassandra for Hadoop.,” *IEEE CLOUD*, vol. 2013, pp. 494–501, 2013.
[30] P. Bagade, A. Chandra, and A. B. Dhende, “Designing performance monitoring tool for NoSQL Cassandra distributed database,” in *Education and e-Learning Innovations (ICEELI), 2012 International Conference on*, 2012, pp. 1–5.
[31] M. Chalkiadaki and K. Magoutis, “Managing service performance in the cassandra distributed storage system,” in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, 2013, vol. 1, pp. 64–71.
[32] “‘J. Rowley and F. Slack,’ Conducting a literature review, ‘Management Research News, Vol. 27, no. 6, pp. 31-39, 2004.’ - Google Scholar.” [Online].
[33] J. J. Randolph, “A guide to writing the dissertation literature review,” *Pract. Assess. Res. Eval.*, vol. 14, no. 13, pp. 1–13, 2009.
[34] M. Berndtsson, J. Hansson, B. Olsson, and B. Lundell, *Thesis Projects: A Guide for Students in Computer Science and Information Systems*. Springer Science & Business Media, 2007.
[35] C. Wohlin, M. Höst, and K. Henningsson, “Empirical research methods in software engineering,” in *Empirical methods and studies in software engineering*, Springer, 2003, pp. 7–23.
[36] P. Cronin, F. Ryan, and M. Coughlan, “Undertaking a literature review: a step-by-step approach,” *Br. J. Nurs.*, vol. 17, no. 1, p. 38, 2008.
[37] J. Webster and R. T. Watson, “Analyzing the Past to Prepare for the Future: Writing a Literature Review,” *MIS Q.*, vol. 26, no. 2, pp. xiii–xxiii, 2002.
[38] S. Keele, “Guidelines for performing systematic literature reviews in software engineering,” in *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*, 2007.
[39] E. B. Johnsen, J.-C. Lin, and I. C. Yu, “Comparing AWS deployments using model-based predictions,” in *International Symposium on Leveraging Applications of Formal Methods*, 2016, pp. 482–496.
[40] “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].
[41] “What is Apache Cassandra | DataStax Academy: Free Cassandra Tutorials and Training.” [Online].
[42] E. Hewitt, *Cassandra: the definitive guide*. O’Reilly Media, Inc., 2010.
[43] “nodetool status.” [Online].
[44] E. Casalicchio, L. Lundberg, and S. Shirinbab, “Energy-aware adaptation in managed Cassandra datacenters,” in *Cloud and Autonomic Computing (ICCAC), 2016 International Conference on*, 2016, pp. 60–71.
[45] T. Rabl, S. Gómez-Villamor, M. Sadoghi, V. Muntés-Mulero, H.-A. Jacobsen, and S. Mankovskii, “Solving big data challenges for enterprise application performance management,” *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 1724–1735, 2012.