Locality aware distributed processes scheduling to improve efficiency using machine learning techniques

By

Saad Zaheer

00000118208

Supervisor

Dr. Asad Waqar Malik

Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science (MS CS)

In

School of Electrical Engineering and Computer Science,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(May, 2018)

Approval

It is certified that the contents and form of the thesis entitled "**Locality aware distributed processes scheduling to improve efficiency using machine learning techniques**" submitted by **Saad Zaheer** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Asad Waqar Malik**

Signature: ____________________

Date: ____________________

Committee Member 1: **Dr. Anis Ur Rahman**

Signature: ____________________

Date: ____________________

Committee Member 2: **Dr. Arslan Ahmed**

Signature: ____________________

Date: ____________________

Committee Member 3: **Dr. Muneeb Ullah**

Signature: ____________________

Date: ____________________

Abstract

Cloud provides a shared computing space on a pay-as-you-go model. Due to this sharing, it is difficult to execute the task efficiently in terms of time. Several factors play its parts such as process scheduling and computing requirement of other processes sharing the system. Moreover, network usage is highly depended on processes, typically processes are categorized computing and communication intensive. Such sharing of platform often degrade the performance of parallel and distributed simulations (PDS). The growth and advancements in the field of Cloud Computing has presented its users with new challenges. Cloud Computing offers resources to its consumers on pay-as-you-go basis. Cloud computing offers storage, virtual machines, memory, processor and network as resources to its consumers. The more you use these resources, the costly it becomes. Parallel and distributed simulation (PDS) involves a number of logical processes (LPs) executing simultaneously while communicating with each other by interchanging small messages called packets using network. Since network is also offered as a resource in cloud computing if a PDS involves exchanging a huge number of messages over a network, the procedure becomes expensive. In this dissertation a new way of simulating huge networks is presented that focuses on reducing network traffic.

Dedication

Dedicated to all the students who are struggling and are doing their best to complete their research.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Saad Zaheer**

Signature: ________________

Acknowledgment

I would like to thank my family for their support, my brother Naseer for the love and uphold and last but not least my supervisor Dr. Asad Waqar Malik who gave me courage and helped me in all aspects while I was doing my research. Thank you sir.

Table of Contents

List of Figures

Chapter 1

Introduction

The domain of parallel and distributed simulations (PDS) has evolved over the years. Its purpose is to simulate the behavior of executing processes on parallel or distributed processors. Communication is established between these processes via either message passing or shared memory. Improving the overall performance of the simulation on heterogeneous platforms is one of the major research concern in this field.

Cloud computing provides its users with hardware and software resources and these resources can be remotely utilized by the users. Goal of cloud service providers is to improve the performance by enhancing the application portability and offer more diverse resource deployment options. Cloud computing data centers require a comprehensive resource allocation system to manage both computational and network resources. There are many optimal and sub-optimal resource allocation techniques for cloud computing data centers [1]. With such appealing processing capabilities of cloud computing, many scientific research applications and paradigms use cloud computing architecture. Similarly, cloud computing presents a fascinating means to provide simulation applications to its users, specially the parallel and distributed simulations.

In traditional distributed simulation, logical processes (LPs) are mapped on different systems/cores. These LPs communicate with each other by sending and receiving time-stamped messages. However, process mapping on different Processing Entities (PEs) in a cloud can greatly affect the performance of entire simulation system.

Performance of Parallel and Distributed Simulations (PADS) degrades on account of the fact that logical processes exchange large number of *event messages* that are sent across the networks. A lot of network traffic is generated because of the recurring communication of processes. The goal of this work is to minimize this network traffic by using Machine Learning's clustering techniques.

We have suggested a migration technique that is based on Machine Learning that will reduce the overall communication expense of the network.

1.1 Background

The idea of parallel and distributed simulations (PDS) first came out four decades ago as an attempt make extremely large scale simulations less time consuming and efficient because the old techniques used for simulations at that time had some system limitations. For example in parallel discrete event simulations the processors that are involved in the simulation are located within the same room while in distributed simulations processors are distributed across the globe and they communicate using network. Another objective of coming up with distributed simulations was to utilize resources that are distributed over the globe. That being said it still had quite a few challenges to cope with because of the new software and hardware requirements. [2].

Simulating the pay-as-you-go nature of cloud computing framework using PDS is a good way of estimating the cost of the framework. A good amount of work has been done to make simulation as efficient and less costly as possible. However lesser attempts in terms of reducing network usage have been made.

1.2 Problem Statement

Just as a cloud resource provider provides resources like memory, storage and processors, network is also provided as a resource to users. Using network as a resource from a cloud provider can become expensive in terms of both time and cost if proper attention is not being paid to it. PDS is a nice way for performing parallel simulation but the performance is highly affected by communication overhead [3]

Processes that are distributed over large distances will rely on network for communication. In such cases frequent exchanging of messages between processes that are distant from each other will be a disadvantage as they will use network and the more you use network the expensive it becomes. This nature of cloud framework is not acceptable as it is not user friendly.

1.3 Thesis Outline

Chapter 2 contains a detailed literature review about advancements and developments in the different research areas of Parallel and distributed simulations. Chapter 3 discusses how the problem is being identified and laid out. Chapter 4 puts lights on the methodology proposed for making simulations network efficient.

Chapter 5 discusses with details the results of our proposed method and comparison with the old method and Chapter 6 contains the conclusion of the research work and the future efforts that could be made in the same domain.

Chapter 2

Literature Review

Parallel and distributed simulation involves a number of logical processes communicating with one another using tiny messages.

A sample of PDS is shown in Fig. 2.1. This is a simple example of PDS in which 5 logical processes are taking part and they are communicating with each other by exchanging small packets.

A significant amount of work has been done in the field of PDS to improve it in one way or another. There are approaches proposed to improve workload balancing. Other approaches like improving energy consumption, performance and network efficiency have also been proposed. Some of the relevant work in this domain will be discussed in this chapter with details. Fujimoto et al. [2] enlightened the state-of-the-art challenges related to research in PDS. In [4] Alfred et al. came up with a technique called Aurora that employs a master worker paradigm. The technique was designed in such a way that it supports the execution of huge simulations on distributed architecture. Aurora utilizes web services to lessen the space between PDS and computing. Aurora is based on the basic idea of parallel discrete event simulation (PDES) containing a number of processes communicating through messages that have a time stamp associated with them. Master is responsible for providing workers with jobs to execute and the workers have a duty of executing those jobs and providing the master with results. This techniques employs the conservative algorithm for synchronization.

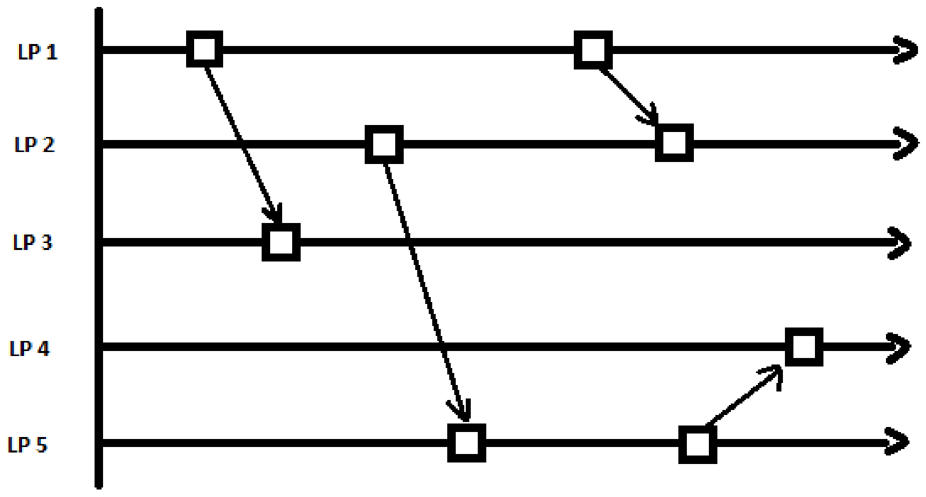


Figure 2.1: A simple simulation model

The cloud virtualization paradigm introduces new challenges for parallel and distributed simulation community. In [5], Srikanth B. Y. et al. explored the impact of runtime dynamics of cloud over PDES performance. The authors have presented the empirical study and proposed a new deadlock free scheduling algorithm designed for PDES application over the virtualized environment.

Shichao Guan et al. [6] proposed a cloud-based framework designed on simulation standards i.e. High Level Architecture (HLA) and Distributed Interactive Simulation (DIS). The basic objective is to manage underlying distributed resources efficiently and to provide unlimited computing for complex simulation scenarios. Moreover, the proposed framework support process migration and handle diverse network paradigms. Further, security features are also added to block malicious attach in the cloud. The experimental evaluation shows that the performance is similar to the grid but it has many advantages over traditional grid environment.

Unbalanced workload may cause frequent rollbacks and huge network traffic. Malik et al. in [7] in their article deals with such situations. Placing processes randomly over clouds can cause workload to be unbalanced due to which the performance of the simulation process will be hugely affected [8].

Perumalla et al [9] proposed a mechanism for processing optimistic parallel application using time warp. The mechanism supports a variety of synchronization techniques. It is designed in such a way that processes can dynamically choose between synchronization techniques. Jagtap et al. in [10] used a procedure for minimization of delays incurred during synchronization using a multi-threaded technique.

In an attempt to improve performance Wentong et al [16] came up with the idea to achieve the best speed for the execution of simulation. The basic concept behind the idea is to provide virtual machines with resources at runtime. They also made sure the nodes execute at a similar pace. Xiao et al. [11] presented a method that increases the performance of High Level

Architecture (HLA) systems. They did so by employing load balancing keep in account both computation and communication expenditure.

In [3] Jingjing Wang et al. approaches to PDES by proposing a multi-threaded simulator. The objective of the author's work is to increase the performance of simulations by using a threaded model. Doing so will increase the performance of communicating processes compared to using a non-threaded model. Munck et al [12] discussed the problems with conservative synchronization's null message protocol justifying that it sends enormous amount of null messages causing overhead and then proposes a technique that integrates the fruitfulness of existing techniques.

In [13], writers Gabriele D'Angelo et al. discussed the challenges faced in executing parallel and distributed simulations in clouds as well as multi-core systems. Under the constraints of performance and utility, authors evaluated the existing PADS techniques for deficiencies and then proposed an adaptive way to overcome those deficiencies. The proposed system lessens the cost of communication and improves load balancing.

In similar attempts Weiwei Chen et al. [14] enhances the performance of PDES by using multi-threading. The objective of the work is to decide at runtime whether it is a good idea or not to launch a group of thread. In [15] Nguyen et al. presented an idea for the execution of large scale simulations. Parallelism at upper levels has been employed to improve network efficiency. The criteria produces efficiency in the networks by giving high level parallelism. The gist of this work is to replace *node model* by *link model*, that is an LP portrays a link to the network and not a node and it will perform better.

Some of the contribution made by researches to the domain of Parallel and Discrete Event Simulation are shown in the figure. Research areas in PDES includes energy efficiency, performance, power efficiency, workload balancing and network performance. Many researches have worked on one or more areas in their research for example Wentong Cai et al. [16] worked on performance and load balancing, Jinjing Wang et al. [3] worked on network performance and overall performance, Angelo et al. [13] did their research on network efficiency, simulation performance and load balancing, Munck et al. [12] worked on just simulation performance and so on.
Randomly placed nodes may cause network traffic to rise as they communicate more and more with each other. In this dissertation a contemporary method has been employed to minimize the cost of network making the communication cost less and thus improving the performance of the simulation.

Table 2.1: Recent Research Contributions in Parallel and Distributed Simulation - Summary

Authors	**Energy Effici.**	**Network Effici.**	**Performance**	**Power Effici.**	**Workload Balancing**
Wentong Cai et al. [16]			✓		✓
Jingjing Wang et al. [3]		✓	✓		
Gabriele DAngelo et al. [13]		✓	✓		✓
S. De Munck et al. [12]			✓		
Weiwei Chen et al. [14]			✓		
Weiwei Lin et al. [17]			✓		✓
Ke Wang et al. [18]			✓		✓
Philipp A. et al. [19]			✓		
Wei Wang et al. [20]	✓		✓		
Xiao Song et al. [21]			✓		✓
Srikanth B. Y. et al. [22]			✓		
Shichao Guan et al. [6]	✓		✓		
Shichao Guan et al. [23]	✓		✓		
Zengxiang Li et al. [24]			✓	✓	
Vy Thuy Nguyen et al. [25]			✓		
Hongjian Li et al. [26]	✓				✓
Juan Fang et al. [27]	✓				
Shuting Xu et al. [28]	✓		✓		
Zhou, Z et al. [29]	✓				
M. A. Khoshkholghi et al. [30]	✓				
Xinhu Liu et al. [31]			✓		

Chapter 3

Problem Formulation

In this chapter the proposed method for minimizing network usage to improve PDES efficiency has been proposed. In a typical PDES we have a number of logical processes (LPs) that are taking part in it. By default these LPs are randomly placed on a number of nodes. The nodes may belong to the same rack or different racks of the data center.

A simulation involves the participation of a number of Logical Processes (LP's) i.e.

$$LP_1, LP_2, ...LP_m \in LP$$

These logical processes are scheduled on a number of Compute Codes (CN's). A data center is a huge station containing a lot of physical cores also called as Processing Elements (PE's). Each Processing Element is capable of scheduling multiple Virtual Machines. Logical processes are scheduled on these virtual machines. Compute Nodes are same as virtual machines. It is to be noted that these nodes may or may not belong to the same rack in a data center.

$$CN_1, CN_2, ...CN_m \in CN$$

Furthermore to layout the network we have supposed a mesh topology. Doing so will enable an LP to be able to communicate with every other LP on the network. There is a cost associated with every path through which a processes will communicate given by:

$$cost_{x,y} | \{weight(x, y) = weight(y, x)\}$$

Additionally, LP's can either be scheduled on the same compute node or on different compute nodes. The cost of communication between such LP's is represented by $Cost_{localNode}$. LP's could also be scheduled on different compute nodes but within one rack. In that case, cost of communication is maintained by $Cost_{localRack}$. In another case LP's could be scheduled on different nodes across different racks and then the cost for this situation is maintained by $Cost_{remoteRacks}$.

The cost of communication between LP's on the same compute node must be the lowest whereas the cost of communication between LP's on different racks must be the highest i.e.

$$Cost_{localNode} < Cost_{localRack} < Cost_{remoteRacks}$$

As cloud supplies its users with resources like processing power, memory and storage on pay-as-you-go basis. Users are charged on account of memory usage, amount of computation, amount of communication and usage of storage. The goal here is to lessen the cost of communication by migrating processes near to its communicating counterpart.

Assuming, if we have six compute nodes on a data center i.e. $CN_1, CN_2, CN_3...CN_6$ and an equivalent amount of LP's are launched on those compute nodes i.e.

$$\begin{aligned}
\{lp_{1.w}, lp_{1.w+1}, ..., lp_{1.x}\} &\rightarrow cn_1 \\
\{lp_{2.x}, lp_{2.x+1}, ..., lp_{2.y}\} &\rightarrow cn_2 \\
&... \\
\{lp_{6.y}, lp_{6.y+1}, ..., lp_{6.z}\} &\rightarrow cn_6 \\
R_1 + R_2 + ... + R_k &\rightarrow DataCenter
\end{aligned}$$

R stands for Racks that belong to a data center.
Additionally nodes pairs are placed on racks.

$$\begin{aligned}
cn_1, cn_2 &\rightarrow R_1 \\
cn_3, cn_4 &\rightarrow R_2 \\
&and \\
cn_5, cn_6 &\rightarrow R_3
\end{aligned}$$

Assuming that communication occurred between process located on different racks is represented by m, communication occurred between processes on different nodes but same rack is represented by r and communication occurred between processes on the same node is represented by q. Assuming

$m > r > q$

$$(cn_1, cn_3) \Rightarrow (p_{1.r+r}, p_{3.j}) \rightarrow m$$
$$(cn_5, cn_6) \Rightarrow (p_{5.l}, p_{6.m}) \rightarrow r$$
$$(cn_4, cn_4) \Rightarrow (p_{4.x}, p_{4.x+1}) \rightarrow q$$

Equations above represent how communication occurred between processes on different racks, on different nodes but same rack and on the same node.

The expense of the total communication is devised as:

$$f(x) = \{\sum_{i=1}^{m} Cost_{remoteRacks} + \sum_{i=1}^{r} Cost_{sameRack} + \sum_{i=1}^{q} Cost_{localNode}\}$$

Considering a three-tier architecture, communication happened between processes on different racks will require at least 3 hops as the message traverser through TOR-AGGREGATE-TOR switches. Similarly, on same rack, the communication will require just one hop and on the same node, there are no hops as the communication is done locally.

$$min\{\textstyle\sum_{i=1}^{m} Cost_{remoteRacks} + \sum_{i=1}^{r} Cost_{localRack} + \sum_{i=1}^{q} Cost_{localNode}\}$$

There is a way [32] to lessen the traffic on the network by introducing locality between communicating processes. More number of hops means more delay incurred. So:

$$delay_{localNode} < delay_{localRack} < delay_{remoteRacks}$$

Bringing processes who communicate more on the same node can lessen the traffic on the network, that is another goal of our work. The idea is to lessen delay incurred in communication between processes that transmit message most frequently by introducing locality between such processes. In traditional synchronization [33] methodology, rollbacks can affect the performance of the simulation by congesting the network which will resultantly make the system costly. So the goal is:

$$min\{\sum_{i=1}^{m} del_{remRacks} + \sum_{i=1}^{r} del_{locRack} + \sum_{i=1}^{q} del_{locNode}\}$$

Reducing the amount of rollbacks means reducing the amount of anti-messages sent across the network, that in result will reduce the network usage and thus enhance the performance.

3.1 Data Center Design

In this research work we proposed a three-tiered data center. Fig. 3.1 shows a typical three-tiered data center. In such data centers there are core switches at the very top level. At the lower level we have aggregate (distribution) switches and at the last level we have top-of-the-rack (access) switches. Physical nodes as many as possible can be connected to the top-of-the-rack switches, aggregate switches connect top-of-the-rack switches and the core switch at the top connects the aggregate switches.

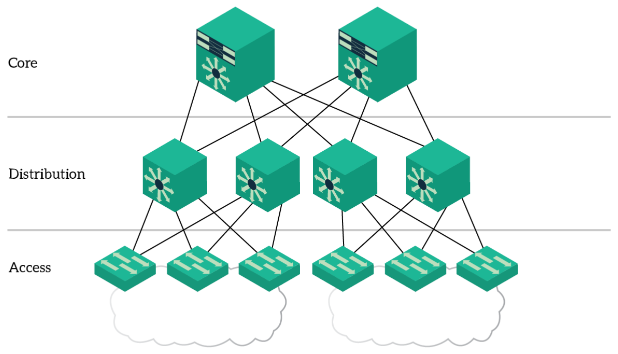


Figure 3.1: Proposed Methodology Diagram

For our problem we used the same three-tiered data center with five nodes connecting to each of the four top-of-the-rack switches. Fig. 3.2 shows the architecture of data center we used for our research problem. There are a total of twenty nodes. Each top-of-the-rack switch can support only five nodes at the same time and no more in our approach.

When a node has to send a message to another node it can either send the message to a node that is located in the same rack or to a node that is located in a different rack. The far the destination node is from the sender node, the higher number of hops the message will take to reach there.

Looking at Fig. 3.2 if node N1 has to send a message to node N2, N3, N4 or N5 the message will take only one hop as it will pass through just the TOR1 switch. If node N1 has to send a message to node N6, N7, N8, N9 or N10 the message will take three hops as it will pass through the TOR1 switch then Aggregate1 switch and then throgh TOR2 switch.

Similary for nodes N11, N12, N13, N14, N15 the messages takes five hops given that the message originates from node N1. For N16, N17, N18, N19 or N20 the number of hops are again five (TOR1-Aggregate1-Core-Aggregate2-TOR4).

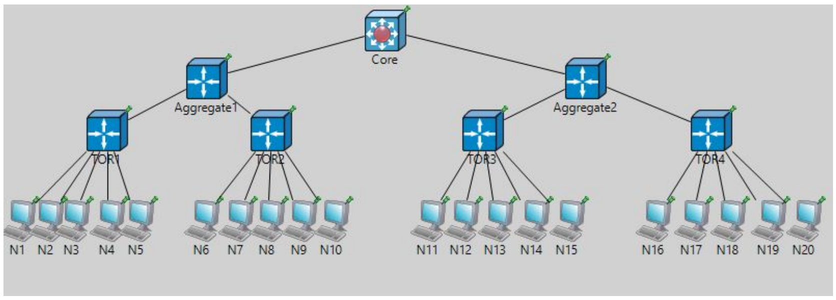


Figure 3.2: Proposed Methodology Diagram

To summarize the message could reach from a sender to destination taking either one, three or five hops. If there is frequent communication between nodes that would require more hops to sent their message to the receiver then in such case there will be a lot of network traffic generated from the nodes. More network traffic will cause the system to consume more bandwidth hence making the process costly and also degrade the performance of the over all simulation process.

On the contrary if we have frequent communication between nodes that will require less number of hops to interchange their messages then less traffic will be generated, less bandwidth will be consumed, performance will be less affected hence resulting in efficient simulation process.

Chapter 4

Proposed Methodology

4.1 Proposed Approach

As discussed previously that randomly placing processes might not be a good idea as there is a higher chance that high network traffic will be generated because of the fact that there might be frequent communication between processes that are placed at a distance from each other.

We also discussed that in our data center design if nodes communicate with other nodes that are part of the same rack will consume less network bandwidth than with those which are part of a different node. We need some approach that can group those nodes together who are frequently communicating with each other and decouple those in between which there is no or less communication.

4.1.1 Clustering Analysis

In search for a better way to minimize the gap between high communicating processes we came up with a technique that involves the use of clustering algorithms. The main purpose of such algorithms is to group together most frequent communicating processes and decouple the non or less communicating ones.

Clustering algorithms are techniques that categorizes different points of data. A number of data points are given that are fed to clustering algorithms and the job of those algorithms is to categorize each of those points into a single relevant and convenient category. Those points in the data that share the same characteristics are put in the same category. In Machine Learning those characteristics are called features. There are mainly two types of Machine Learning techniques i.e. Supervised Learning and Unsupervised Learning. Clustering algorithm are associated to Unsupervised Learning.

Clustering algorithms provides valuable feedback for that data it works on by putting them in different clusters. There are various types of Clustering algorithms like K-Means Clustering, Density based Spatial Clustering and Agglomerative Hierarchal Clustering etc. A brief detail about each of those algorithms is give below.

K-Means Clustering

One of best knowns algorithm for clustering is called K-Means. It's is simple to understand and to work with. K-Means works in the following manner:

- The first step is to set the number of clusters. A random central point will be selected for each cluster by the algorithm
- In the next step each data elements is put in a cluster that it belongs to using a distance function
- Each cluster has a central point that is derived by taking the mean of all the points in the cluster
- This process is repeated for a number of times until when there is no or insignificant change in the central points

K-Means is known for being fast and simple. We just have to derive the distance between data points and central points of clusters.

Hierarchal Clustering

- There are two approaches to Hierarchal Clustering i.e. Top-down and Bottom-up. Each point of data is considered as one cluster in bottom-up approach.
- There are multiple iterations involved in hierarchal clustering. Each iteration merges two clusters into one using some distance measures (complete link, average link etc). For example average link works by finding the mean distance between the data points in one cluster and data points in the other.
- Any two clusters are merged in one if they have the smallest distance between them
- The above step is repeated till we merge the all data points in one cluster.

- We can select the number of clusters by specifying when to stop merging clusters.
- A tree like structure is maintained in this algorithm

This technique works best in the cases where the structure of the data it is working on is hierarchal and you need to derive that hierarchy.

Density Based Spatial Clustering of Application with Noise (DBSCAN)

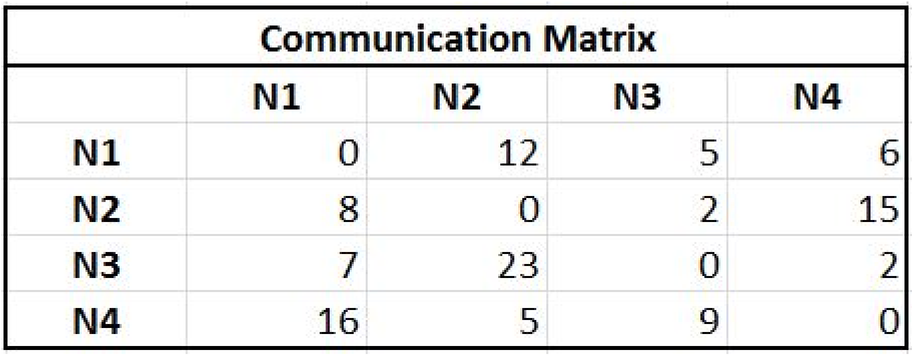
- DBSCAN starts with random unvisited initial data point. This data point has a defined neighborhood that is derived using a distance function ϵ. Data points falling inside this ϵ are neighboring data points
- A criteria called minPoints is defined that is actually a minimum number of points that should fall inside the ϵ. So if there are enough data points the meets the defined criteria then the process starts and the current points becomes the initial point this new cluster. On the other hand if there are no enough data points that meet the minimum criteria then that point is considered as noise
- The points that fall inside the ϵ are added to the cluster and this processes is repeated for those new points again
- This processes is kept continuous until all the data points within ϵ are visited and tagged.
- After being done with this cluster a new point is selected that is unvisited by that time and the same procedure takes place for it.

Due to ϵ and the minPoints criteria, DBSCAN does not perform as good as other clustering algorithms if the cluster have different density or the data has more features.

Proposed approach

This is a technique that we designed ourself and its purpose is also to cluster together frequent communicating processes. The technique works in the following way:

- An n*n matrix is generated by running simulation for sometime. The processes placed in the initial simulation follow the random placement technique.
- The n*n matrix shows the number of times each process that is involved in the simulation, communicates with each other processes in the simulation. So for example if there are 20 processes involved in the PDS, the generated matrix will be 20*20
- The next step is to combine the number of messages that two processes exchange with one another throughout the simulation, for example if a node N1 sends 30 messages to another node N8 and N8 sends 12 messages to N1, we will combine these two values and make them a pair so we will have N1N8/N8N1 = 42. Similarly we will pair up all the remaining nodes. For example:



Communication Matrix				
	N1	N2	N3	N4
N1	0	12	5	6
N2	8	0	2	15
N3	7	23	0	2
N4	16	5	9	0

Figure 4.1: Communication matrix of 5 nodes

- Now we will pair up the nodes and add up their messages like:

$$\begin{aligned}
N1N2 = N1 \rightarrow N2 + N2 \rightarrow N1 = 12 + 8 = 20\\
N1N3 = 5 + 7 = 12\\
N1N4 = 6 + 16 = 22\\
N2N3 = 2 + 23 = 25\\
N2N4 = 15 + 5 = 20\\
N3N4 = 2 + 9 = 11
\end{aligned}$$

- So the total number of pairs for a 4*4 matrix will be 6.
- Now if we have to make a cluster of 4 processes , we will proceed as:
  - Select the pair with highest value for example in our case N2N3 is the highest and add them to cluster

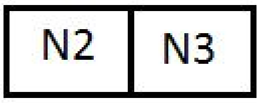


Figure 4.2: Communication matrix of 5 nodes

  - Below N2, add the node with which N2 has the most communication and below N3 add the node with which N3 has the most communication i.e.

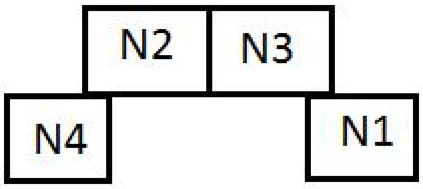


Figure 4.3: Communication matrix of 5 nodes

  - Next compare the two pairs N2N4 and N3N1 and add the highest to the cluster and then continue until you reach the cluster limit
  - After completing one cluster, delete all the pairs in which any of the node is involved that is already added in a cluster and do so again for the next cluster.

- Algorithm 4.1 contains the pseudo code for the above explained procedure.

Algorithm 4.1 Proposed Method

```
1: Generate an $n * m$ communication matrix
2: convert the matrix into key/value pairs
3: Integer cluster_size
4: Integer total_clusters
5: List current_cluster
6: for $i = 1$ to total_clusters do
7:     pick the pair$(x, y)$ with max value
8:     add pair$(x, y)$ to the current_cluster
9:     while current_cluster.size() $<$ cluster_size do
10:        if either $x$ or $y$ belongs to current_cluster then
11:            list the pairs that x&y mostly communicate with
12:            compare the two pairs
13:            add max to the list
14:    Delete all the pairs containing nodes from the current_cluster
```

4.2 Implementation of Proposed Way

In this research work we employed Unsupervised Machine Learning techniques i.e. K-Means clustering, Hierarchal clustering and Density Based clustering to find structure in our data produced by processes involved in our simulation.

Initially all processes were placed across the data center randomly. Random communication was established between those processes and the communication patterns were recorded from the simulation.

The features that makes up our are the number of messages sent by a process, the number of messages received by a process, the total number of messages sent or received by a process, the total number of hops took by messages sent by a single process and the means hops of each process.

We ran the simulation for a million events and wrote the data to a file with the aforementioned features. We used Weka [34] that is one of the best tool for application of Data Mining diverse algorithms on user's data, for cluster analysis. We fed our data to different clustering algorithms using Weka and made changes in our Data Center's structure according to the results of those algorithms. The cluster algorithm will decide based on the features we provided which processes have identical communication patterns and group them together. The results will be discussed in the Chapter. 5

Chapter 5

Results

5.1 Assumptions

We have kept the number of clusters to be made by the clustering algorithm to be exactly four because we are considering a data center with four racks. Meaning the a physical node can be in either of those four racks. Now the clustering algorithm can have results that will put more nodes in one cluster and less in another. For example if the algorithm gives us four clusters containing 3, 7, 4, 6 nodes. We will have to do load balancing in such case and keep all cluster of the same size that is 5, 5, 5, 5 because there can only be five nodes in a rack in our proposed data center.

5.2 Experiments and their results

In this section we will be discussing about the results of our research work. How random placement generated huge network traffic and how our proposed methods reduced it. As already discussed, a message to be sent from one node to another can either take one, three or five hops. The best case that will consume less bandwidth is that a processes be sending a message that requires only one hop to reach its destination. The next best case will be for the message to require three hops and the worst case that would consume more bandwidth would be that a message requires 5 hops to reach its intended destination.

We will be showing the following in our results:

Within a rack/cluster:

- Number of messages sent within a rack taking only 1 hop
- Number of messages sent across a rack taking 3 hops
- Number of messages sent between racks taking 5 hops

Within the Data Center

- Number of messages sent within the Data Center taking only 1 hop
- Number of messages sent within the Data Center taking 3 hops
- Number of messages sent within the Data Center taking 5 hops

In the end will be comparing how our proposed method resulted in a scenario that reduced network traffic i.e. reduced number of messages that took five hops and increased those which took one or three hops.

5.2.1 Results of Random Process Placement

The initial random method where processes are randomly placed across the data center resulted in greater network traffic as huge amount of messages were sent across the network which consumed a lot of bandwidth and affected the simulation performance. Figure. 5.1 shows the representation of how the initial random method performed within each of the four clusters.

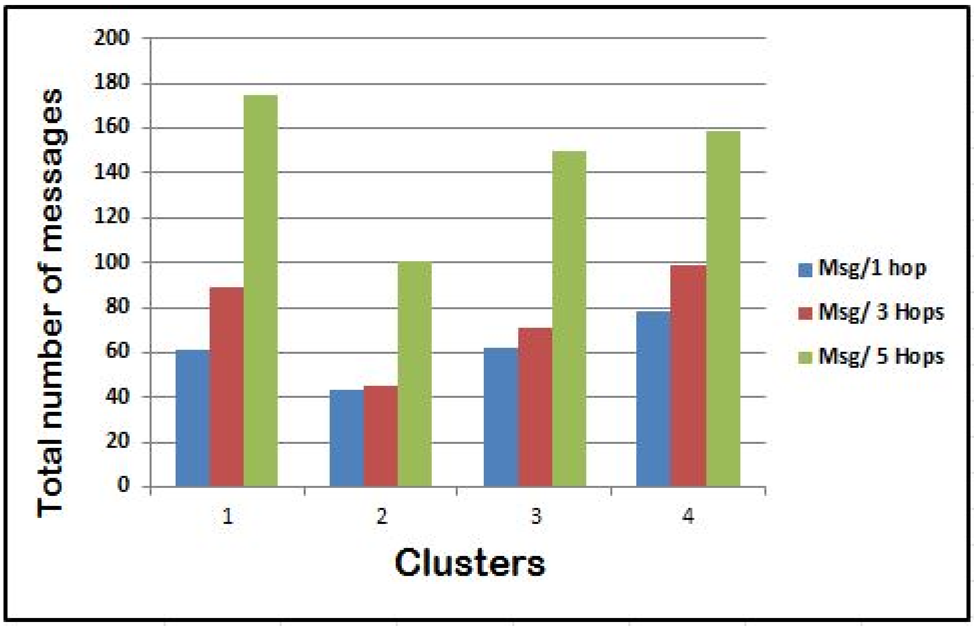


Figure 5.1: Cluster-wise Performance of Random method

On x-axis we have four clusters and on y-axes we have the number of messages sent by processes. It can be clearly seen that the number of messages that took five hops were greater in each one of the rack while the one that took only one hop were minimum. Figure. 5.2 shows the data produced by random method.

In cluster 1 the messages taking only one hop to reach their destinations

Random Method						
Cluster	1	2	3	4	Total	%
Messages/1 hop	61	43	62	78	244	21.5357
Messages/3 hops	89	45	71	99	304	26.8314
Messages/5 hops	175	101	150	159	585	51.6328
Total	325	189	283	336	1133	100

Figure 5.2: Data produced by random method

were only 61, those taking three hops were 89 and the ones that took five hops were of greater number 175. In cluster 2 we have 43 messages taking one hop, 45 taking three hops and 101 taking five hops. Cluster 3 has 62, 71, 150 message taking one , three and five hops respectively and cluster 4 has 78, 99, 159. Fig. 5.3 shows how the random method performed within the whole data center.

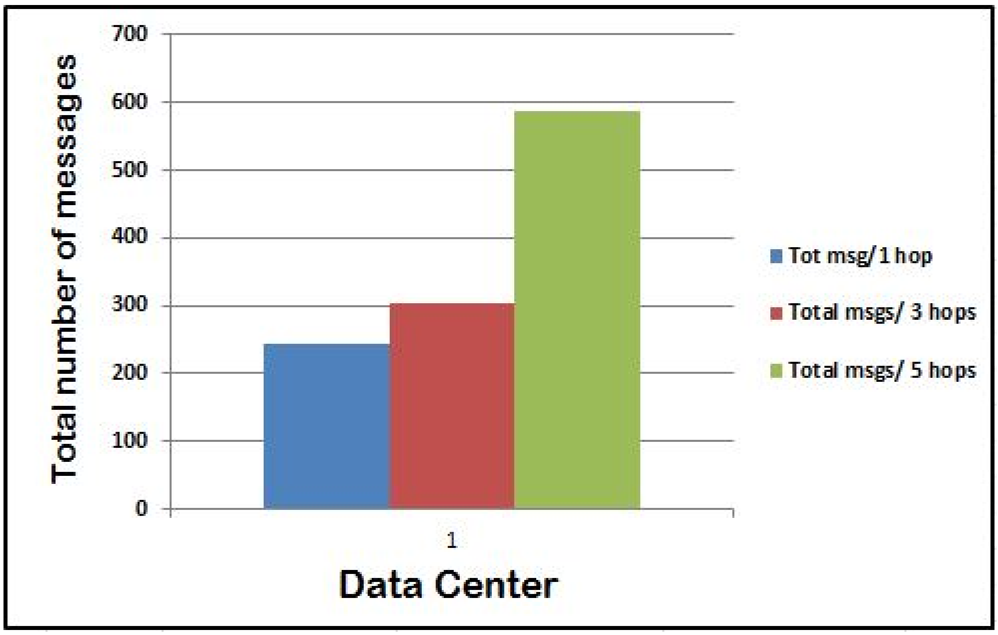


Figure 5.3: Data Center Performance of Random method

So the total number of messages across the whole data center that took 1 hop were only 244, messages taking three hops were 304 and those which took the maximum number of hops i.e. 5 were 585. This is like the complete opposite of our ideal scenario where we want to reduce the network traffic as much as possible.

5.2.2 Results using DBSCAN

The first clustering algorithm we used for grouping related processes (more communicating) together was DBSCAN. Fig. 5.4 shows the results of DBSCAN and Fig. 5.5 shows the statistics.

In Cluster 1 messages taking:

- Only 1 hops were 33
- 3 hops were 51
- 5 hops were 117

In Cluster 2 messages taking:

- Only 1 hops were 117
- 3 hops were 137
- 5 hops were 234

In Cluster 3 messages taking:

- Only 1 hops were 56
- 3 hops were 54
- 5 hops were 99

In Cluster 4 messages taking:

- Only 1 hops were 59
- 3 hops were 72
- 5 hops were 104

Similarly the overall results of the data center after using DBSCAN algorithm were:

DBSCAN						
Cluster	1	2	3	4	Total	%
Messages/1 hop	33	117	56	59	264	23.3892
Messages/3 hops	51	137	54	72	314	27.714
Messages/5 hops	117	234	99	104	554	48.8967
Total	201	488	209	235	1133	100

Figure 5.5: Data produced by DBSCAN method

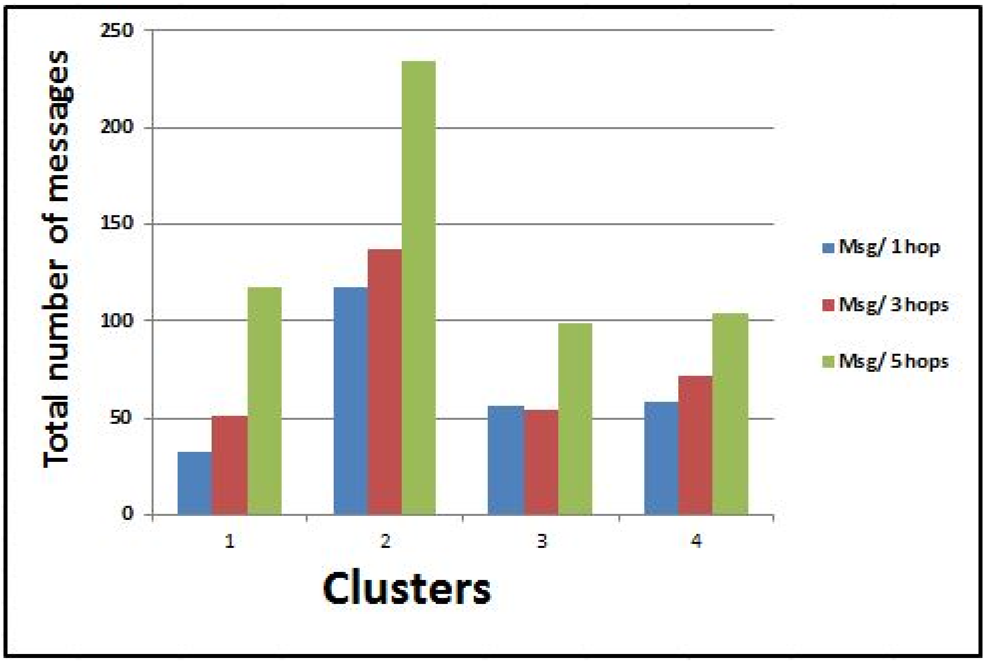


Figure 5.4: Cluster-wise Performance of DBSCAN method

Similarly the overall results of the data center after using DBSCAN algorithm were:

In Data Center the messages taking:

- Only 1 hop were 264
- 3 hops were 314
- 5 hops were 554

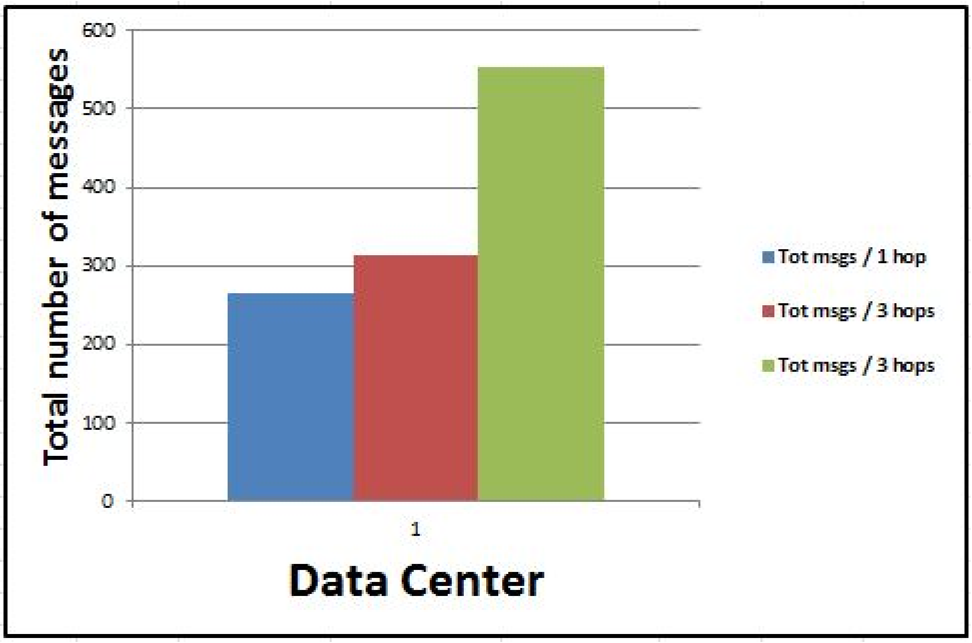


Figure 5.6: Data Center Performance of DBSCAN method

DBSCAN did only just better than the random method and resulted in just slightly less network traffic. It increased hop 1 messages by only a small percentage that is 1.8535%, increased hop 3 messages by just 0.88% and decreased hop 5 messages by just 2.7401%.

5.2.3 Results using Hierarchical Clustering

In comparison to random method using Hierarchical clustering to structure to our network also performed but with just slight improvement. Fig. 5.7

shows the network analysis and Fig. 5.8 shows its statistics.

In Cluster 1 messages taking:

- Only 1 hops were 46
- 3 hops were 64
- 5 hops were 101

In Cluster 2 messages taking:

- Only 1 hops were 37
- 3 hops were 50
- 5 hops were 107

In Cluster 3 messages taking:

- Only 1 hops were 94
- 3 hops were 116
- 5 hops were 147

In Cluster 4 messages taking:

- Only 1 hops were 81
- 3 hops were 113
- 5 hops were 177

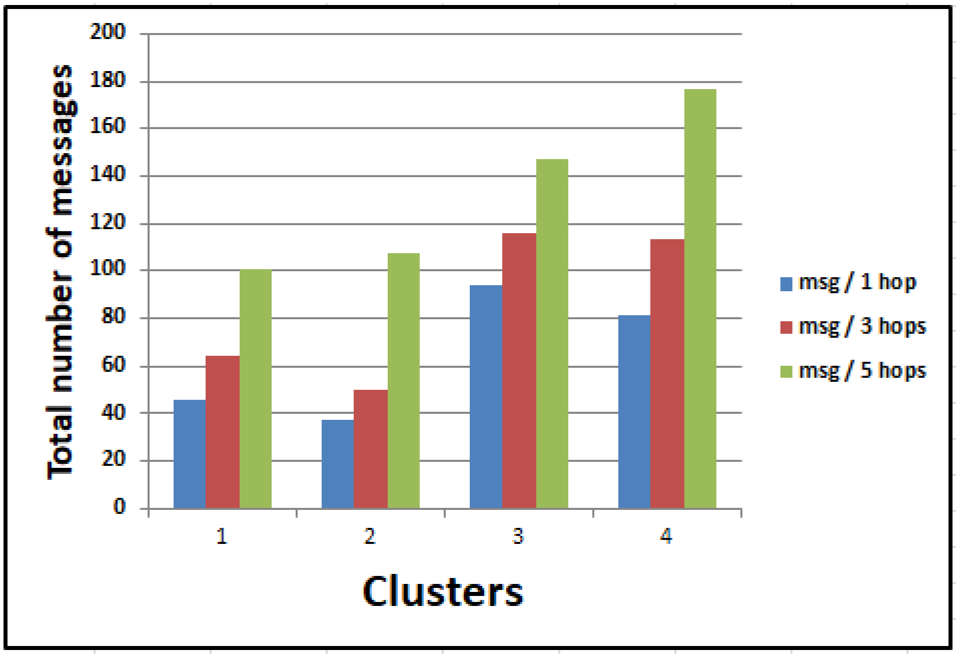


Figure 5.7: Cluster-wise Performance of Hierarchical method

Hierarchal						
Cluster	1	2	3	4	Total	%
Messages/1 hop	46	37	94	81	258	22.7714
Messages/3 hops	64	50	116	113	343	30.2736
Messages/5 hops	101	107	147	177	532	46.955
Total	211	194	357	371	1133	100

Figure 5.8: Data produced by Hierarchical method

Throughout the data center the performance of hierarchical clustering was as follows:

In Data Center the messages taking:

- Only 1 hop were 258
- 3 hops were 343
- 5 hops were 532

Fig. 5.9 depicts the data center's data representation.

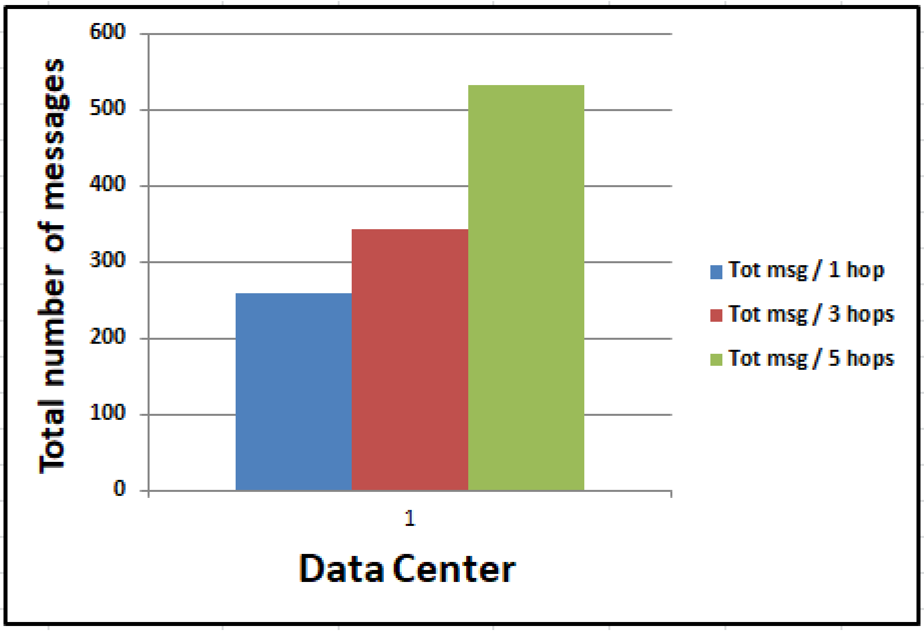


Figure 5.9: Data Center Performance of Hierarchical method

There was an increase of 1.2% in hop 1 messages, an increase of 3.44% in hop 3 messages and a slight decrease of 4.67% in hop 5 messages. Again the change was not significant but still the results were better than in case of random method.

5.2.4 Results using K-Means

K-Means gave us better results by outclassing all both DBSCAN and Hierarchical Clustering algorithms. Fig. 5.10 shows the results of K-Means in all four clusters and the statistic of this representation is given in Fig. 5.11.

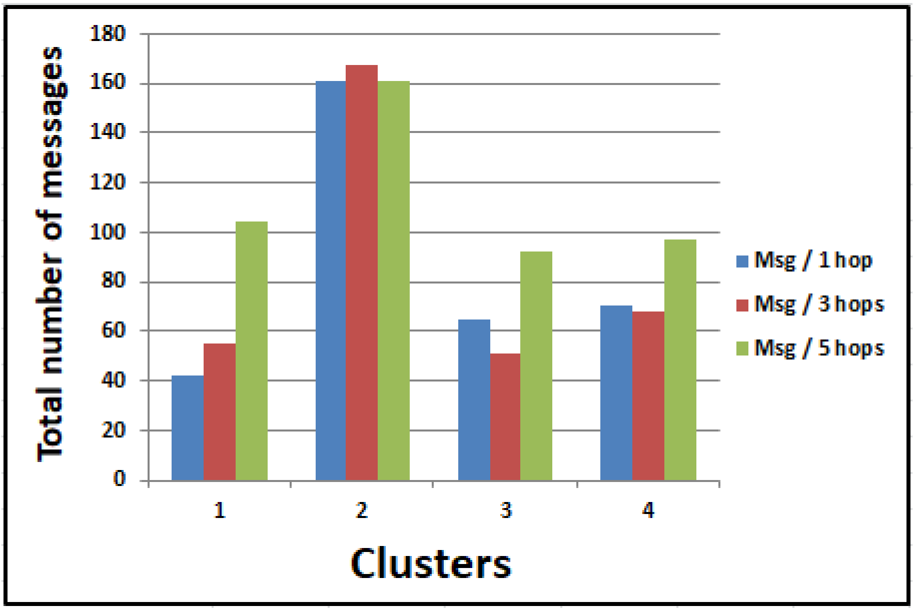


Figure 5.10: Cluster-wise Performance of K-Means

In Cluster 1 messages taking:

- Only 1 hops were 42
- 3 hops were 55
- 5 hops were 104

In Cluster 2 messages taking:

- Only 1 hops were 161
- 3 hops were 167
- 5 hops were 161

In Cluster 3 messages taking:

- Only 1 hops were 65
- 3 hops were 51
- 5 hops were 92

In Cluster 4 messages taking:

- Only 1 hops were 70
- 3 hops were 68
- 5 hops were 97

K-Means						
Cluster	1	2	3	4	Total	%
Messages/1 hop	42	161	65	70	338	29.8323
Messages/3 hops	55	167	51	68	341	30.0971
Messages/5 hops	104	161	92	97	454	40.07061
Total	201	489	208	235	1133	100

Figure 5.11: Data produced by K-Means

Fig. 5.12 shows data center performance after using K-Means. The facts of the data center are:

In Data Center the messages taking:

- Only 1 hop were 338
- 3 hops were 341

- 5 hops were 454

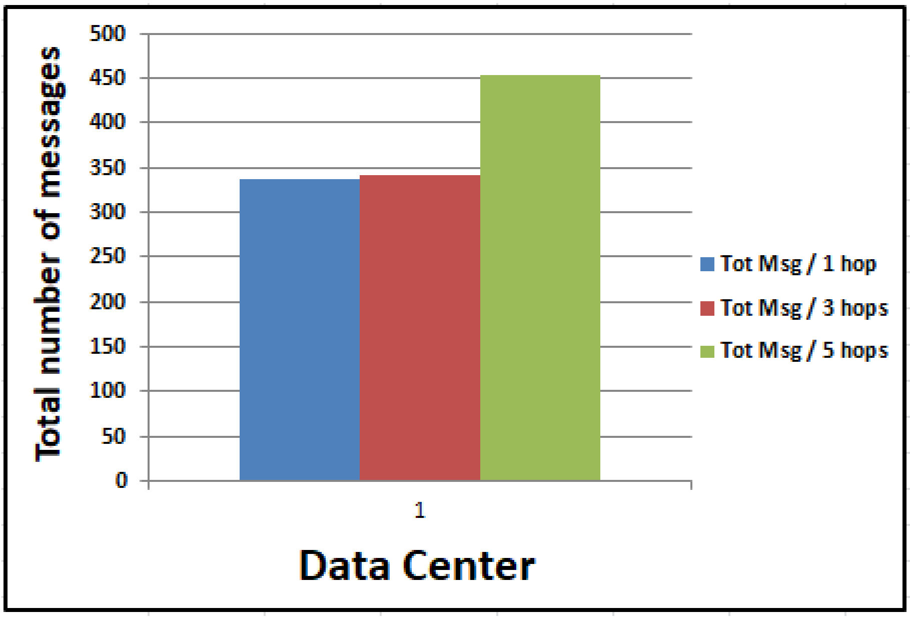


Figure 5.12: Data Center Performance of K-Means

In comparison to random method K-Means performed way better and it also did well than its partner clustering algorithms i.e. DBSCAN and Hierarchical Clustering. After using K-Means the number of messages taking only 1 hop were increased by 8.29%. Those messages that took 3 hops increased by 3.26% and the number of messages taking the maximum 5 hops to reach their destination were reduced by 11.56% which beats all the other algorithms.

5.2.5 Results using proposed method - Experiment 1

Experiment 1 consisted of 10% defined communication and the rest was random. Tree method performed in the best way and the results were better than random method and all the other approaches that we used. Fig. 5.13 shows the results of Tree method in all four clusters and the statistics of this representation is given in Fig. 5.14

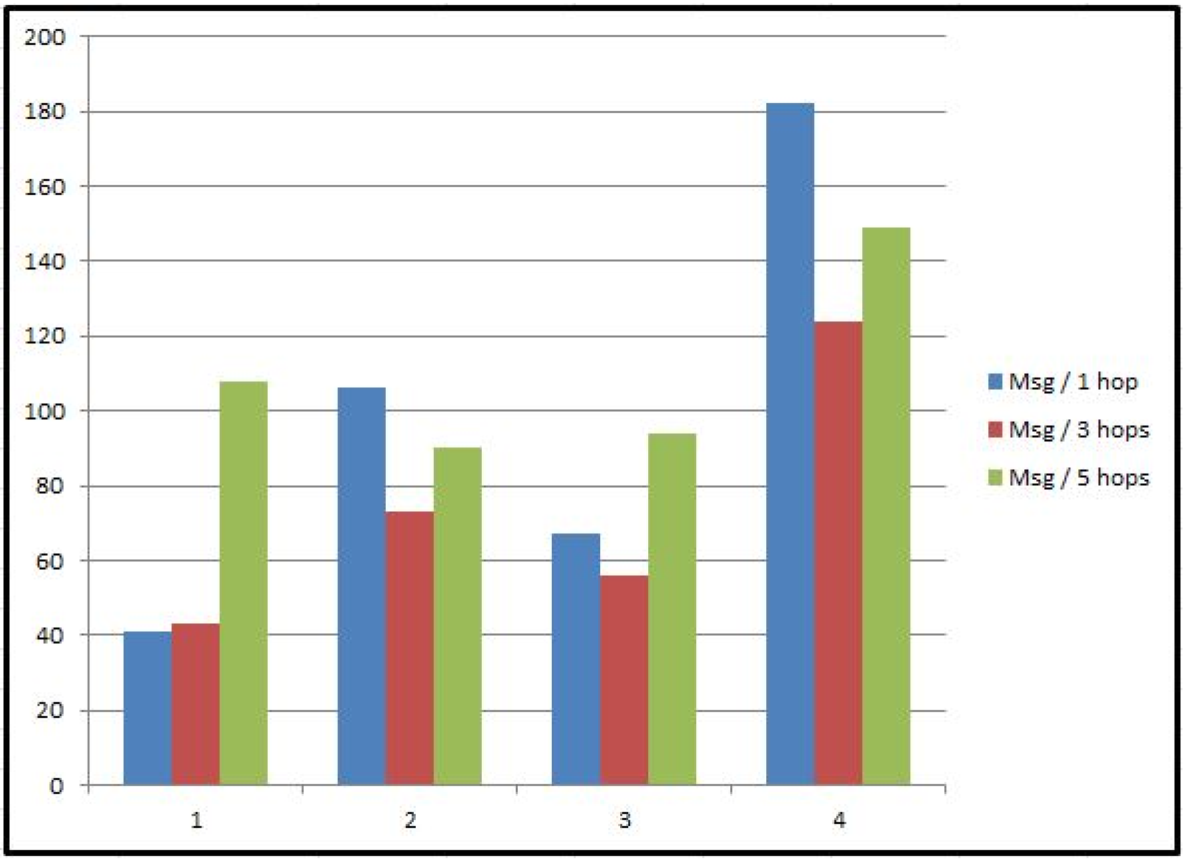


Figure 5.13: Cluster-wise Performance of Tree Method

In Cluster 1 messages taking:

- Only 1 hops were 41
- 3 hops were 43
- 5 hops were 108

In Cluster 2 messages taking:

- Only 1 hops were 106
- 3 hops were 73
- 5 hops were 90

In Cluster 3 messages taking:

- Only 1 hops were 67
- 3 hops were 56
- 5 hops were 94

In Cluster 4 messages taking:

- Only 1 hops were 182
- 3 hops were 124
- 5 hops were 149

Tree Method						
Cluster	1	2	3	4	Total	%
Messages/1 hop	41	106	67	182	396	34.95146
Messages/3 hops	43	73	56	124	296	26.12533
Messages/5 hops	108	90	94	149	441	38.92321
Total	192	269	217	455	1133	100

Figure 5.14: Data produced by proposed method

Fig. 5.15 shows data center performance after using proposed method. The facts of data center are:

In Data Center the messages taking:

- Only 1 hop were 396
- 3 hops were 296
- 5 hops were 441

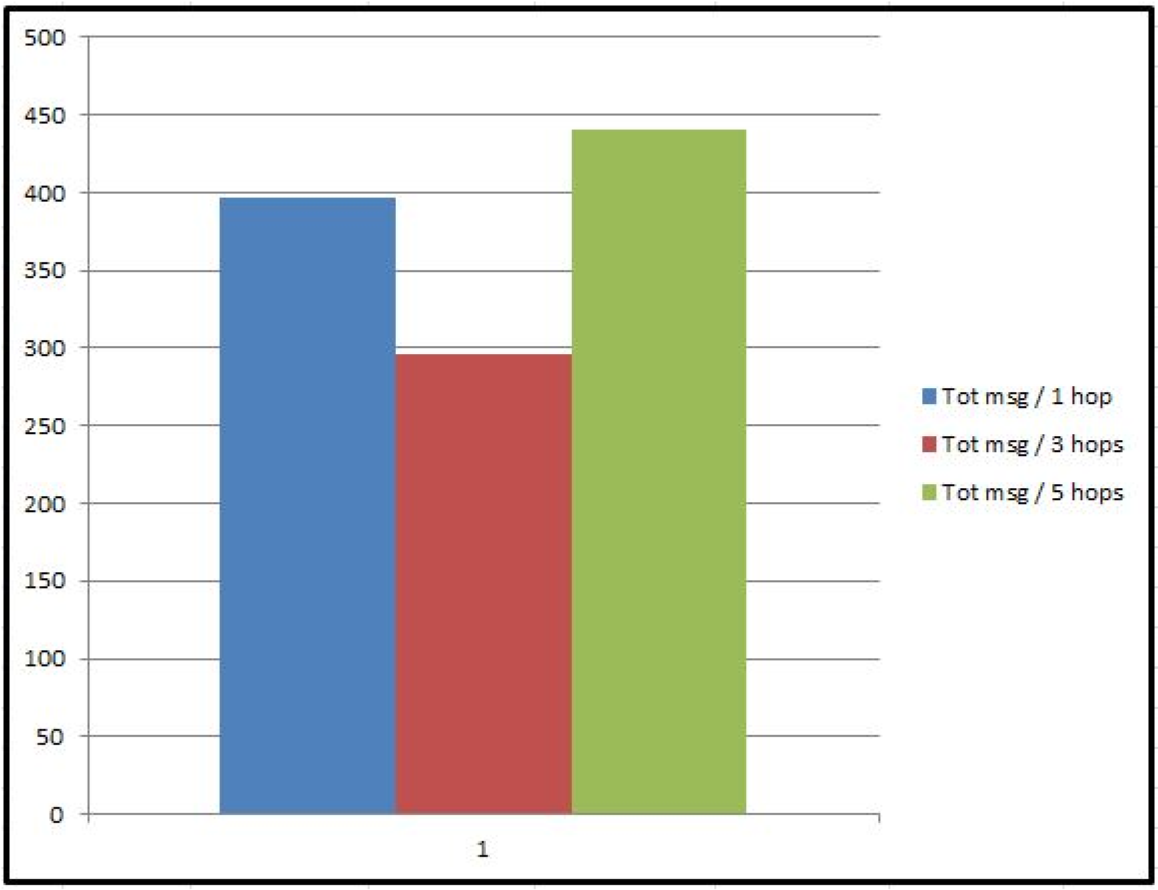


Figure 5.15: Data Center Performance of proposed Method

21.5357,26.8314,51.6328 In comparison to random method, proposed method performed way better and it also did well than all other algorithms i.e. DB-SCAN, Hierarchical Clustering and K-Means. After using proposed Method the number of messages taking only 1 hop were increased by 13.4%. Those messages that took 3 hops increased by 0.7% and the number of messages taking the maximum 5 hops to reach their destination were reduced by 12.709% which beats all the other algorithms.

5.2.6 Results using proposed method - Experiment 2

So to make sure the results are better we applied our proposed approach on another scenario where the communication patterns are different than the last simulation's communication patterns. We increased the planned communication upto 15% in this experiment and the rest of the communication followed a random pattern. Fig. 5.16 shows how the results were in case the processes were placed randomly, fig. 5.17 shows the the performance of the overall data center for the same case and fig. 5.18 shows the statistics.

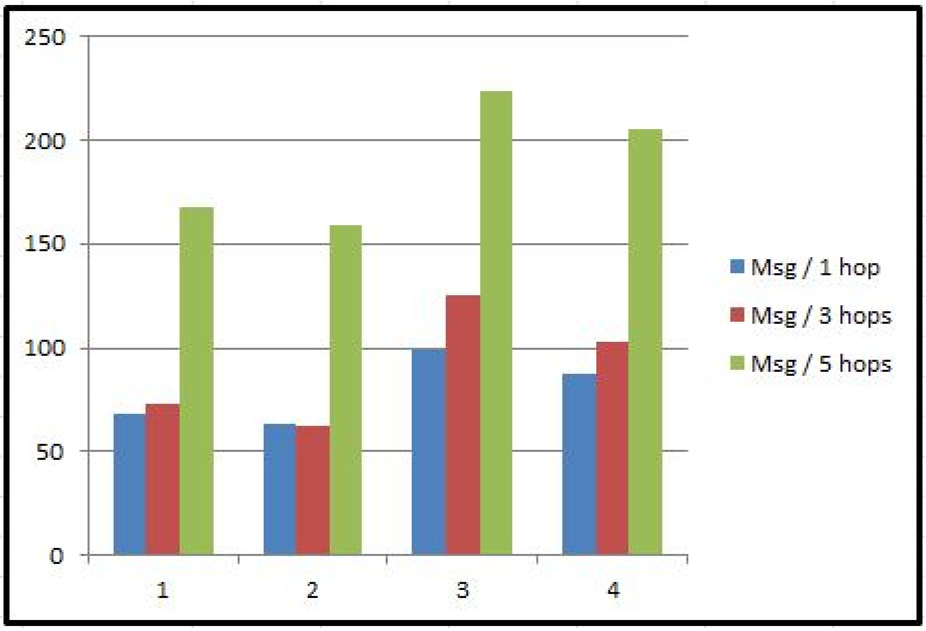


Figure 5.16: Rack-wise Performance of Random Method

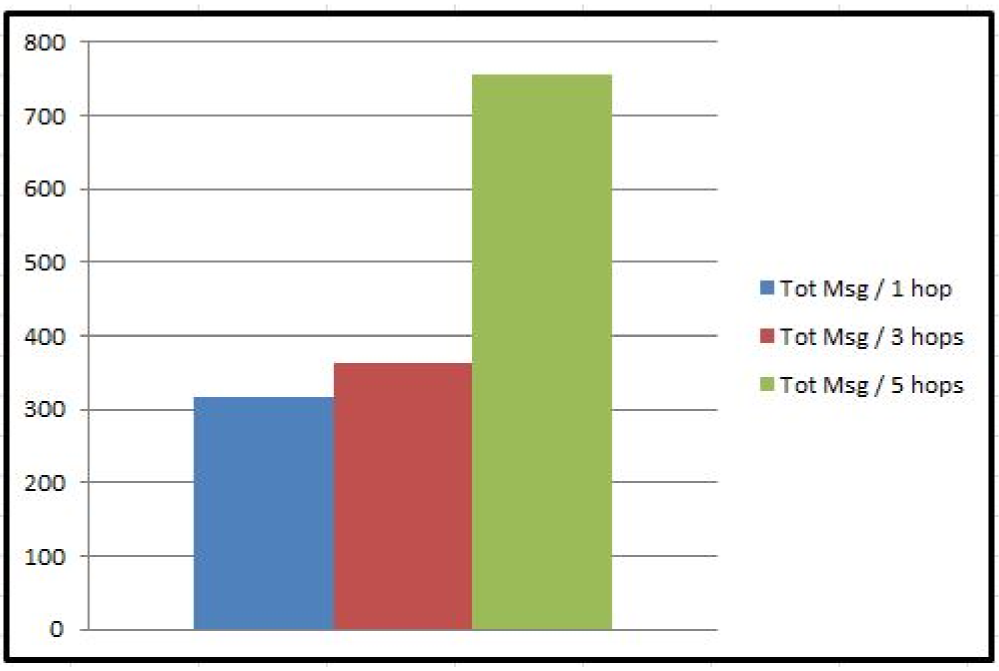


Figure 5.17: Data Center performance of Random Method

Random Method						
Cluster	1	2	3	4	Total	%
Messages/1 hop	68	63	99	88	318	22.12944
Messages/3 hops	73	62	125	103	363	25.26096
Messages/5 hops	168	159	224	205	756	52.6096
Total	309	284	448	396	1437	100

Figure 5.18: Data produced by Random method for experiment 2

Now we will apply our proposed approach for the same simulation and compare the results with random method. Fig. 5.19, fig. 5.20 and fig. 5.21 rack-wise performance, statistics and data center wise performance of our proposed method respectively.

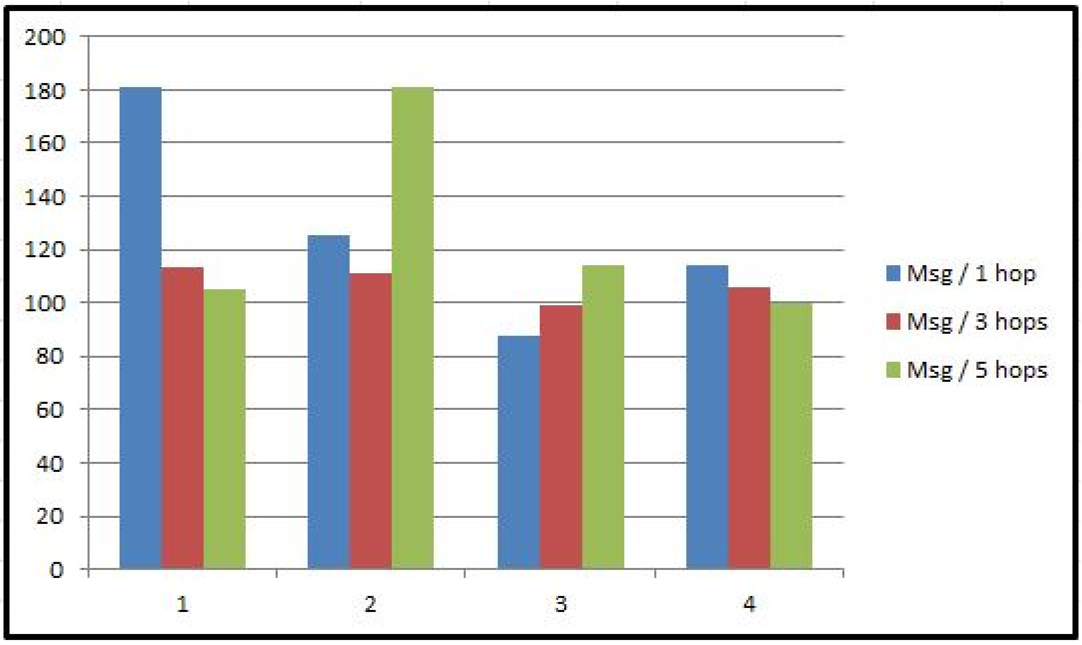


Figure 5.19: Cluster-wise Performance of Proposed Method

In Cluster 1 messages taking:

- Only 1 hops were 181
- 3 hops were 113
- 5 hops were 105

In Cluster 2 messages taking:

- Only 1 hops were 125
- 3 hops were 111
- 5 hops were 181

In Cluster 3 messages taking:

- Only 1 hops were 88
- 3 hops were 99
- 5 hops were 114

In Cluster 4 messages taking:

- Only 1 hops were 114
- 3 hops were 106
- 5 hops were 100

Proposed Method						
Cluster	1	2	3	4	Total	%
Messages/1 hop	181	125	88	114	508	35.35143
Messages/3 hops	113	111	99	106	429	29.85386
Messages/5 hops	105	181	114	100	500	34.79471
Total	399	417	301	320	1437	100

Figure 5.20: Data produced by proposed method

In Data Center the messages taking:

- Only 1 hop were 508
- 3 hops were 429
- 5 hops were 500

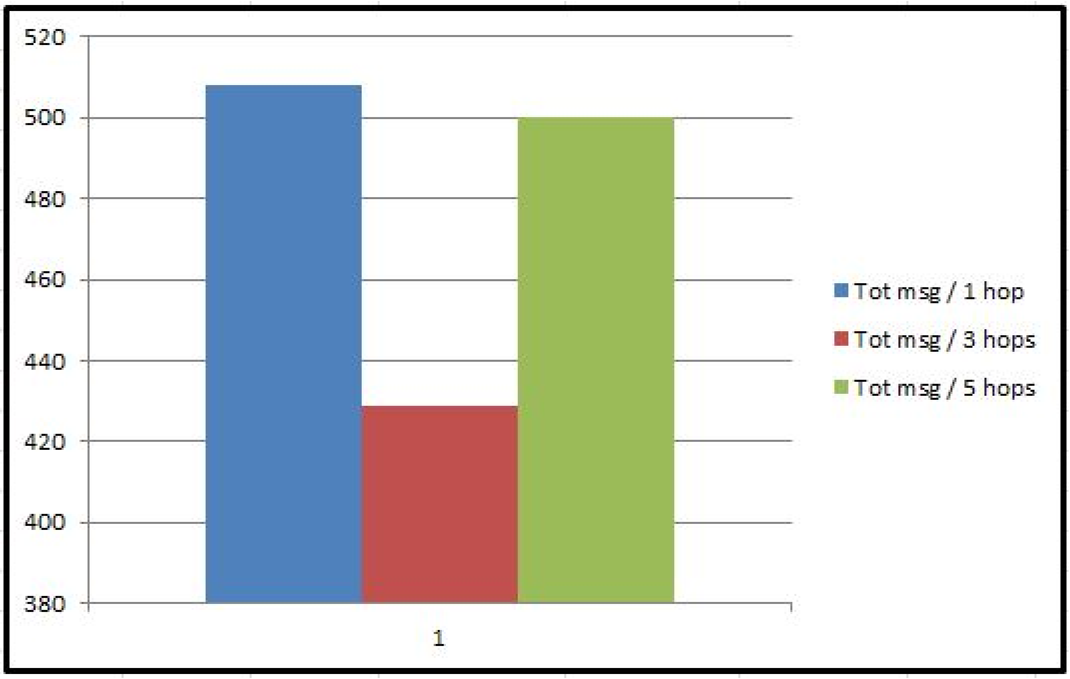


Figure 5.21: Data Center Performance of proposed Method

The number of messages taking only 1 hop were increased by 13.22%. Those messages that took 3 hops increased by 4.59% and the number of messages taking the maximum 5 hops to reach their destination were reduced by 17.81% which again is a significant increase in performance.

5.3 Summary of Results

To summarize the results we will discuss in this section how each technique helped in reducing the total number of hops throughout the entire data center. Fig. 5.22 shows the total number of hops took by messages in all of the four techniques we experimented on in this thesis. Fig. 5.23 shows the statistics.

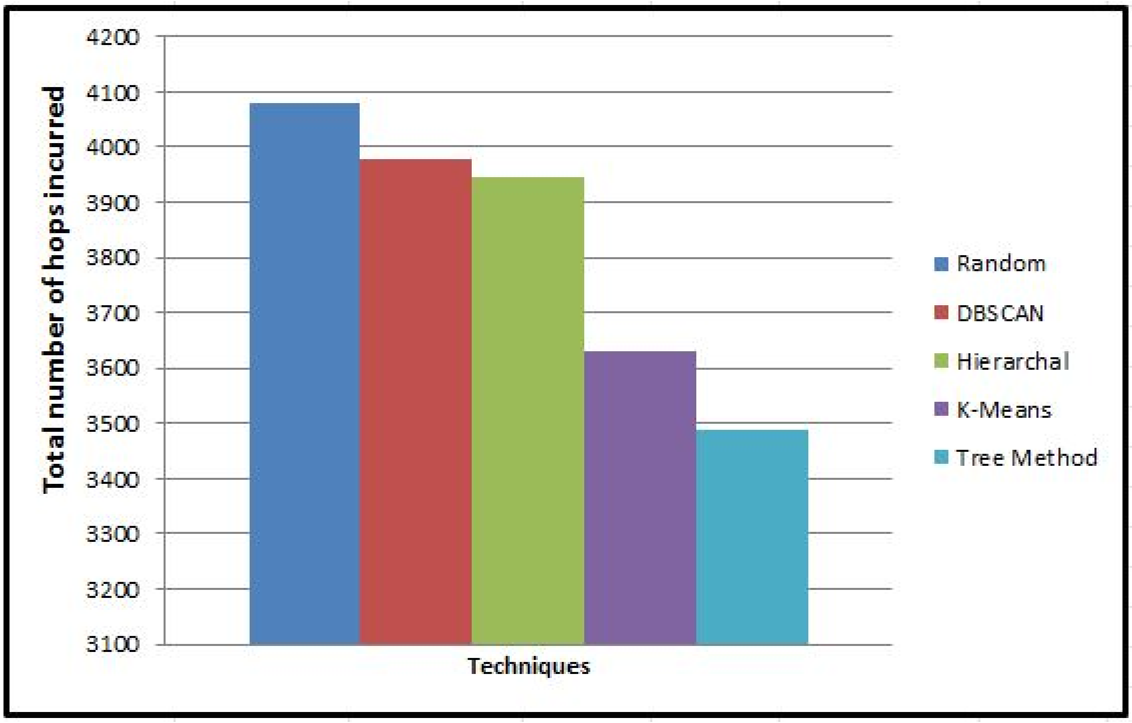


Figure 5.22: Comparison of techniques

Number of Hops Comparison					
Technique	Random	DBSCAN	Hierarchical	K-Means	proposed Method
Hops	4081	3977	3947	3631	3489

Figure 5.23: Hop-wise comparison of discussed techniques

There were a total of 1133 message packets sent across the data center by the 20 nodes that were taking part in the simulation. Each one of these 1133 messages took either one, three or five hops to reach its intended destination. Our simulation performance will be affected highly if the majority of those messages took five hops to reach their destination. On the other hand the performance will be better if somehow the messages taking five hops were reduced and those taking just one hops were increased. More hop 3 messages is also a better scenario than more hop 5 messages. The results of the techniques we employed are discussed below.

The total number of hops throughout the simulation in experiment 1 in:

- Random method were: 4081
- DBSCAN method were: 3977
- Hierarchical Clustering were: 3947
- K-Means Clustering were: 3631
- Proposed method were: 3489

Reduction percentage in total number of hops in experiment 1 by using:

- DBSCAN was 2.54%
- Hierarchical Clustering was 3.28%
- K-Means was 11.02%
- Proposed Method was 14.50%

The total number of hops throughout the simulation in experiment 2 in:

- Random method were: 5187

Reduction percentage in total number of hops in experiment 2 by using:

- Proposed Method was 17.19%

Chapter 6

Conclusion

In Cloud computing framework resources such as memory, storage, processing power and network are shared by a huge number of users. Different users have different workload demands. Executing Parallel and Discrete Event Simulation for such platforms may effect the performance of the simulation if the processes are placed randomly across the network as huge amount of message will travel through the network which will generate huge network traffic and consume a lot of bandwidth and thus make the system costly. We need a criteria that can dynamically keep the system state balanced and efficient.

We proposed an automated system that dynamically identifies those processes which communicate with each other a lot and place them near one another as compare to the random method that place a process randomly at any location generating huge network traffic. Our method worked better and it significantly decreased the amount of messages that traveled across the network. This reduction in the number of messages lead to reduction in network delays and hence improved the overall performance of our PDS.

In future efforts could be made to apply clustering techniques to reduce energy and power consumption and load balancing.

Bibliography

[1] Mohamed Abu Sharkh, Abdallah Shami, and Abdelkader Ouda. Optimal and suboptimal resource allocation techniques in cloud computing data centers. *Journal of Cloud Computing*, 6(1):6, 2017.

[2] Richard M Fujimoto. Research challenges in parallel and distributed simulation. *ACM Trans*, 26(4):1–29, 2016.

[3] Jingjing Wang, Deepak Jagtap, Nael Abu-Ghazaleh, and Dmitry Ponomarev. Parallel discrete event simulation for multi-core systems: Analysis and optimization. *IEEE Transactions on Parallel and Distributed Systems*, 25(6), 2014.

[4] Alfred J. Park and Richard M. Fujimoto. Efficient master/worker parallel discrete event simulation on metacomputing systems. *IEEE Transactions on Parallel and Distributed Systems*, 23(5):873–880, 2012.

[5] Srikanth B. Yoginath and Kalyan S. Perumalla. Efficient parallel discrete event simulation on cloud/virtual machine platforms. *ACM Trans. Model. Comput. Simul.*, 26(1):5, 2015.

[6] S. Guan, R. De Grande, and A. Boukerche. A multi-layered scheme for distributed simulations on the cloud environment. *IEEE Transactions on Cloud Computing*, pp(99), 2015.

[7] A. Malik, A. Park, and R. Fujimoto. Optimistic synchronization of parallel simulations in cloud computing environments. *IEEE International Conference on Cloud Computing*, pages 49–56, 2009.

[8] R. M. Fujimoto, A. W. Malik, and A. Park. Parallel and distributed simulation in the cloud. *SCS M&S Magazine*, 3(4), 2010.

[9] K. S. Perumalla. Scaling time warp-based discrete event execution to 104 processors on a blue gene supercomputer. *Proceedings of the 4th international conference on Computing frontiers, ACM*, pages 69–76, 2007.

[10] D. Jagtap, N. Abu-Ghazaleh, and D. Ponomarev. Optimization of parallel discrete event simulator for multi-core systems. *In Parallel & Distributed Processing Symposium (IPDPS), IEEE*, pages 520–531, 2012.

[11] Xiao Song, Yaofei Ma, and Da Teng. A load balancing scheme using federate migration based on virtual machines for cloud simulations. *Mathematical Problems in Engineering*, 2015(506432), 2015.

[12] S. De Munck, K. Vanmechelen, and J. Broeckhove. Revisiting conservative time synchronization protocols in parallel and distributed simulation. *Concurrency and Computation: Practice and Experience*, 26(2):468–490, 2014.

[13] Gabriele D'Angelo and Moreno Marzolla. New trends in parallel and distributed simulation: From many-cores to cloud computing. *Simulation Modelling Practice and Theory*, 49:320–335, 2014.

[14] Weiwei Chen, Xu Han, Che-Wei Chang, Guantao Liu, and Rainer Dömer. Out-of-order parallel discrete event simulation for transaction

level models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(12), 2014.

[15] Vy Thuy Nguyen and Richard Fujimoto. Link partitioning in parallel simulation of scale-free networks. *IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications*, 2016.

[16] ZengXiang Li, Xiaorong Li, Long Wang, and Wentong Cai. Hierarchical resource management for enhancing performance of large-scale simulations on data centers. *Proceedings of the 2nd ACM SIGSIM Conference on Priciples of Advanced Discrete Simulation*, pages 187–196, 2014.

[17] Weiwei Lin, Chen Liang, James Z. Wang, and Rajkumar Buyya. Bandwidth-aware divisible task scheduling for cloud computing. *Journal of: Software: Practice and Experience*, 2014.

[18] Ke Wang, Xiaobing Zhou, Tonglin Li, Dongfang Zhao, Michael Lang, and Ioan Raicu. Optimizing load balancing and data-locality with dataaware scheduling. *IEEE International Conference on Big Data*, pages 119–128, 2014.

[19] Philipp Andelfinger and Hannes Hartenstein. Exploiting the parallelism of large-scale application-layer networks by adaptive gpubased simulation. *Simulation Conference (WSC) ,10.1109/WSC.2014.7020179*, pp:3471–3482, 2014.

[20] W. Shu, W. Wang, and Y. J Wang. A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. *https://doi.org/10.1186/1687-1499-2014-64*, (64), 2014.

[21] Xiao Song, Yaofei Ma, and Da Teng. A load balancing scheme using federate migration based on virtual machines for cloud simulations. *Mathematical Problems in Engineering*, 2015(506432), 2015.

[22] Srikanth B. Yoginath and Kalyan S. Perumalla. Efficient parallel discrete event simulation on cloud/virtual machine platforms. *Transactions on Modeling and Computer Simulation (TOMACS) - Special Issue PADS TOMACS Homepage archive*, 26(1):5, 2015.

[23] Shichao Guan, Robson Eduardo De Grande, and Azzedine Boukerche. Enabling hla-based simulations on the cloud. *IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications*, pp:112–119, 2015.

[24] Zengxiang Li, Wentong Cai, Stephen John Turner, Xiaorong Li, Ta Nguyen Binh Duong, and Rick Siow Mong Goh. Adaptive resource provisioning mechanism in vees for improving performance of hla-based simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 26(1):1, 2015.

[25] Vy Thuy Nguyen and Richard Fujimoto. Link partitioning in parallel simulation of scale-free networks. *IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications*, pages 77–84, 2016.

[26] H. Li, G. Zhu, and C Cui. Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing. *Computing*, 98(3):303–317, 2016.

[27] Juan Fang, Lifu Zhou, Xiaoting Hao, Min Cai, and Xingtian Ren. Energy and performance efficient underloading detection algorithm of vir-

tual machines in cloud data centers. *IEEE International Conference on Cluster Computing*, pp:134–135, 2016.

[28] S. Xu, Wu C.Q., A. Hou, Y. Wang, and M. Wang. Energy-efficient dynamic consolidation of virtual machines in big data centers. *International Conference on Green, Pervasive, and Cloud Computing*, 10232:191–206, 2017.

[29] Zhou Zhou, Zhi gang Hu, Jun yang Yu, Jemal Abawajy, and Morshed Chowdhury. Energy-efficient virtual machine consolidation algorithm in cloud data centers. *Journal of Central South University*, 24(10):2331–2341, 2017.

[30] Mohammad Ali khoshkholghi, Mohd Noor Derahman, Azizol Abdullah, Shamala Subramaniam, and Mohamed Othman. Energy-efficient algorithms for dynamic virtual machine consolidation in cloud data centers. *IEEE Access*, 5:10709–10722, 2017.

[31] Xinhu Liu and Philipp Andelfinger. Time warp on the gpu: Design and assessment. *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 109–120, 2017.

[32] W. Jian, W. KwameLante, G. Kartik, and XenLoop. A transparent high performance inter-vm network loopback. *Proceedings of the 17th International Symposium on High Performance Distributed Computing*, pages 109–118, 2008.

[33] K. M. Chandy and J. Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452, 1979.

[34] https://www.cs.waikato.ac.nz/ml/weka/.