Efficient Resource Allocation to Optimize the Network Usage

By
Saba Maqbool
2012-NUST-MS IT-13

Supervisor
Dr.Asad Waqar Malik
NUST-SEECS

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(May 2016)

Approval

It is certified that the contents and form of the thesis entitled "**Efficient Resource Allocation to Optimize the Network Usage**" submitted by **Saba Maqbool** have been found satisfactory for the requirement of the degree.

Advisor: **Dr.Asad Waqar Malik**

Signature: _____
Date: _____

Committee Member 1: **Dr. Anis ur Rehman**

Signature: _____
Date: _____

Committee Member 2: **Dr. Muhammad Muneeb Ullah**

Signature: _____
Date: _____

Committee Member 3: **Dr. Arsalan Ahmad**

Signature: _____
Date: _____

Abstract

In this thesis we proposed an algorithm for efficient resource allocation in torus network. The torus is one of the topology used in High Performance Computing (HPC) to connect systems and high speed networks. The High Performance applications usually requires significant computing capacity and network bandwidth utilized during execution. Therefore, by applying this efficient resource allocator algorithm, we can reduce the excessive use of HPC resources for concerned user applications. The resources of the HPC systems are virtual machines, processes,high computing nodes that are using the torus network are handled intelligently by using the proposed algorithm(Efficient Resource Allocator - ERA).So, that the communication between these resources is minimized and in return reduce the network cost of torus network. ERA allocates the resources by using local information of the targeted application on torus. The efficient allocation of resources according to an assignment scheme and then migrate them (Virtual Machines). The objective is to reduced the communication among processes and thus optimize network usage. The evaluation and benchmarking of the proposed ERA shows significant reduction in terms of communication cost.

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 National University of Sciences & Technology (NUST) School of Electrical Engineering & Computer Science (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: **Saba Maqbool**

Signature: _____

Acknowledgment

In the name of Allah the beneficent and merciful, on whom we all are dependent for eventual support and guidance. I would like to express my immense gratitude to my supervisor Dr. Asad Waqar Malik, who has guided and supported me throughout my thesis work and allow me to work in my own way and polish my skills. I highly appreciate his help in technical writing. His mentorship was predominant in sustaining versatile experience in my long term career goals. I would also like to thank my committee members, Dr. Anis ur Rehman, Dr. Muneeb Ullah and Dr. Arsalan Ahmad who continuously guide me in my thesis, and provide their valuable suggestions and encouragement. Finally I would like to thanks my friends and colleagues who have lent a hand in order to complete my thesis.

Saba Maqbool

Contents

List of Abbreviations

Abbreviations	Descriptions
DRA	Distributed Re-assignment Algorithm
MEPL	Minimum Expected Path Length
MPSoC	Multiprocessor System on chip
BFD	Best Fit Decreasing
SLA	Service Level Agreement
LLC	Limited Look-ahead Control
MBFD	Modified Best Fit Decreasing

List of Figures

List of Tables

Chapter 1

INTRODUCTION

In the most recent couple of years, there has been an incredible exertion from cloud suppliers to offer easy to use situations that can be utilized by customers to execute applications. In [1] the services are conveyed to the cloud clients according to pay per use model, which implies that the proprietor of an application is required to pay that is corresponding to the measure of resources the individual application expends amid its execution on the cloud. Consequently, applying wise strategies to minimize the asset utilization is of vital significance.

The previously stated issue can be tackled by recognizing a task plan between (a) the communicating components of an application, for example, processes and virtual machines, (b) the computing VM/processes of a cloud framework, such that the aggregate sum of resources consumed by the individual application is minimized. In previous research [2] have concentrated on a varieties of the network topology, for example, homogeneous linear arrays or trees, which make the issue reasonable in the polynomial time [3]. Since our proposed model considers the relocation of components of an application (VM/processes) imparting with each other, processes and VMs are utilized conversely all through the content.

The issue turns out to be more intriguing while considering applications that face continuous changes in their assignment load and designs. It depends on the assignment scheme that is ideal for quite a while may become non ideal later. Hence, it is significant to progressively reassign the processes/VMs to nodes considering the new correspondence requests of the application segments. Be that as it may, additional care must be taken to evade repetitive calculations for VMs/processes that are as of now ideally situated inside the framework. In view of the extra requests in both time and memory, past approaches that expand on the minimum cut maximum flow strategy, for

example, the ones presented [4] are not versatile while considering extensive scale applications.

Consider a two dimensional rectangular grid of computing nodes in network and fold this grid from top to bottom and join the first row of nodes with a row of nodes at the bottom of the grid by communication links. Now the top row has direct link with nodes of last row. Then in last again fold and stretch the grid from left to right direction. In this way all the computing nodes on left column and in the right column are directly linked [5].Typical torus network shown in Figure 1.1.

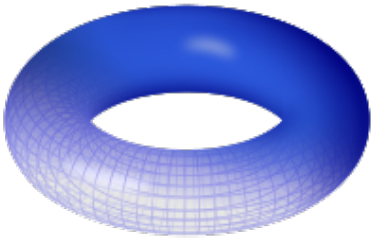

Figure 1.1: Torus Network

Recently a lot of work has been done [6] to facilitate the users of cloud applications. The clients of cloud are using the services provided by the cloud users according to SLA for cloud applications. Thus the clients are charged, how much they used their services. The more the numbers of resources are used by the client, more they have to pay for the execution of an application on cloud. So, a better scheme is required to allocate the resources dynamically for each application that the resource consumption is reduced. This problem can be solved if the processes and nodes are placed in such a way that the minimum number of resources [7] will be consumed and the overall cost is also decreased.

1.1 Problem Statement

*D*atacenter is often based on flat tree network, on which resource allocation is relatively more explored [8] as compare to torus network are more complex due to cycles involved. For resolving this problem consider that HPC applications and the resources (components of an application) to nodes are reassigned in such a way that the total communication and consumption of network should be minimized.

Major aspects of our proposed research work is summarized as follows:

- Handling the issue in a completely rapid and disseminated way.
- Making just local neighborhood decisions that cause insignificant overhead of framework control messages.
- Identifying the interdependencies between the processes of the specified application.
- Proving that calculation dependably brings about an ideal arrangement of nodes/processes/VM.
- Guaranteeing that the proposed solution dependably always converges.
- Developing a scheme that can similarly work well with trees and various leveled networks also.
- Proposing strategies to handle the situations where the super-nodes inside the framework are over-burden.

Chapter 2

LITERATURE REVIEW

This chapter covers the related work, divided them into the following classes: (a) task allocation, (b) VM placement and (c) migration mechanism (d) distributed dynamic environment (e) comparison and evaluation.

Torus network is the interconnection of processing nodes for parallel super computing systems [9]. In torus network the processes are connected with their neighbors in such a way that it seems like a mesh topology. In mesh topology each node have 2N connections. In 2D torus each node has 4 neighbors and in 3D torus network each node has 6 neighbors[10]. Torus network is made by a rectangular 2D grid. This rectangular grid is rolled as the opposite ended nodes directly link with each other. The rolled torus is visualized as tube. Now this tube is bend to form a torus in which nodes which far apart are closer due to these bending of rectangular lattice consist of a grid of rows and columns. However, for other networks a long length cable link is required to direct link those nodes which are on opposite edges. Each link in folded torus is as short as nearest neighbor link in simple grid of nodes. Thus, due to short links in torus network, applications has to face low hardware latency [11].

The cloud based applications demand is increasing day by day. The cloud providers are interested in to facilitate the cloud users and their as much as they can and provide them friendly environment. The data centers are more explored to optimize the resource usage of cloud based and high computing applications. Number of algorithms have proposed to increase the performance of high computing processes by providing the task allocation scheme of processes so that the resource utilization is reduced and the cloud users have to pay less for those applications. The reason is that the owners of the cloud applications have to pay for using the cloud services[10].

To solve the above mentioned problem, researcher Azeez et al. [12] has

been done and they had worked on assignment scheme of resources for the hosted applications so that the number of resources required for those applications have been minimized. Earlier work [13] to decrease the communication cost and execution cost is divided into two categories: (1) by task allocation scheme(2) by divided the whole task in virtual machines.

2.1 Task Allocation

Earlier related works [14] concentrated on task assignment issue over a double processor framework are such illustrations, for example, to minimize the aggregate communication cost. In [14],authors Jason et al. have more explored different structures of networks. The fundamental contrasts of the previously stated works with our proposed are (a) that the previously stated arrangements are centralized, while our proposed algorithms is completely distributed, and (b) that they don't consider the capacitated case. Related are likewise a few variants of the exemplary planning issue in multiprocessor frameworks where they include diverse enhancement targets. In particular,

the streamlining objective is to minimize both the aggregate make span and correspondence overhead at the same time. The researchers handle the issue of minimizing the aggregate fulfillment time of parallel occupations in heterogeneous frameworks [15]. The calculations minimizing in the meantime energy utilization furthermore, the aggregate make span. The creators of [16] explore the assignment scheme of enhancing (a) the productivity as far as social welfare, and (b) the decency as far as jealousy freeness. The researchers of [16] study the issue of expanding both the usage of the framework and the throughput execution on heterogeneous situations. All the previously stated research, related, varies from our proposed work.

2.2 VM Placement

There is additionally a numerous number of research works in the writing identified with energy utilization and VM union. However, the vast majority of them don't consider the system overhead when choosing about the task of VMs onto super nodes (servers). In [10], authors handle the issue of identifying over-burden has by optimizing the mean inter-migration time under the predefined nature of administrative services (QoS) objective in view of a Markov-chain model. In [16], the researchers propose energy consumption heuristics that combine VMs onto the basis of number of servers involved. At the same time, the researcher of [17] additionally investigate solution for

minimize the SLA (Service Level Agreement) disagreement experienced inside a cloud because of the workload combination systems strategies. The researcher propose a workload combination algorithm that changes the Best Fit Decreasing (BFD) approach [16]. The proposed calculation is named Modified Best Fit Decreasing (MBFD). The same issue is additionally handled utilizing Limited Look-ahead Control (LLC). All the previously stated works vary from our proposed work in that they don't consider the communication dependencies among VMs.

2.3 Migration of Processes/VM

In this section, we specify the most significant works we found in the writing with respect to the relocation components that they are supported at various levels: (i) thread- level, (ii) process-level, and (iii) VM-level. We should take note of that such works are corresponding to our proposed work, subsequent to our proposed algorithm could embrace such mechanism to actualize VM or migration of VM/processes in and of themselves. Thread level movement systems are considered [18]. In [11], the researchers can discover an order of process/VM/nodes relocation mechanism: (a) UNIX-like frameworks supporting straightforward migration, (b) frameworks with message passing interfaces, (c) microkernels, (d) client space movement, and (e) application-particular relocation. The researcher Buyya et al. [19] talk about a Java VM relocation system that considers the accompanying arrangement of properties: transparent, adaptability, consistency, fulfillment, versatility, efficiency, and power.

2.4 Affinity Aware Technique on Torus

The computing platform of virtual machines are used for large scale computing environment such as for data centers, grid and cloud environment. *Starling* reduced the communication overhead of these computing environments by proposing an affinity aware technique of migration. They [20] handled two main factors in network topology. First one heterogeneity and secondly dynamic network. The communication pattern of resources is managed by allocation of resources and placement of virtual machines so that the communication overhead is minimized in data centers. The difference in above research work and in our proposed work is that they work on grid and data centers environment [19] while we have optimized the torus network in which cycles are involved and due to these cycles the physical overhead is

minimized as well as the least communication cost bared by the owners of the cloud applications [10].

2.5 Locally Virtual Migration of Nodes/ Processes

Another most similar research work to our proposed work that migrate the virtual machines to reduce the communication overhead. They also used the technique of migration of VM on the nearest resource available so that the communication cost will somehow reduce. The drawback of this algorithm is that they had not considered overall effect on the whole network topology due to these migration. In other words they just consider the local cluster benefit between specified source and destination. However the whole network cost or global benefit is decreased due to such migration technique[21]. In contrast to this work our algorithm considered the overall network communication and execution cost and local cluster cost also. So, before migration we measured the local as well as global network communication cost [11]. At earlier times, on k-ary-n-tube interconnection topology or on torus a lot of research work had done on placement of nodes and used the codes for error correction using lee distance method[22].

2.5.1 1-Dimensional Torus

[22] The involvement of virtual networks have make work more easy and fascinating the researchers because it is easy to maintain them and re-mapped them over the original physical network. Virtualization provides flexibility in performing different complex operations. They solved the problem of scheduling the nodes and move them from its virtual placements to the original places so that decreases the cost and time both. Firstly, they [23] have done work on single node migration locally so that less cost and latency is faced by the network. Then they migrate more than one nodes at a time so that minimum disruption will be occurred in a network and also simultaneous migration is done in less time.

2.5.2 2-Dimensional Torus

Researcher Almohammad et al.[24] proposed three new algorithms for scheduling the virtual network. First is LMCF in which they migrate the single node in one transmission. According to two other algorithms MIS-SS and MIS-LMCF that pre than one node can be migrated simultaneously so that cost

and time is reduced. The advantage of this research is that they evolve the concept of more than one node migrate at the same time but the disadvantage of all these three algorithms is that they migrate them and calculate the benefit locally and the whole application cost is not optimized by them[25]. Jun Doi especially work on asymmetrical torus network and improve the performance of an application by improving the traffic of longest axis of the network [21]. They used a strategy to separate the thread of the longest axis, thus the traffic of the long route is separated from the traffic of others routes so that the long axis will not become the bottleneck for the whole application and scheduled the traffic of the long axis smoothly [26].

2.5.3 Resource Placement at Perfect Distance

The resource placement in network is explored since last few years and in this [10] research they have improved the network efficiency by changing the places of those nodes which are within the given distance that is d and those which are out of the range known as non-resource nodes. Due to this placement strategy they have improved the message latency of the network if the resources are at the perfect distance d from other resource node.

2.5.4 MEPL from the Center of the Region

The research work on networks, data centers and also on chip network where they [27] have reduced the minimum expected length of the path among the resources. The network efficiency increased locally in a specified small region and the distance from the central node of that small region is minimized. Thus the cost and time has been reduced by this scheduling scheme. The disadvantage of this research is that in case of some applications the whole network cost and time is increased because they have placed the resources without considering the whole network.

Chapter 3

SYSTEM MODEL AND PROBLEM FORMULATION

This chapter consists of following portions: (1) firstly we have discussed about system model then (2) problem formulation

3.1 System Model

We assumed a torus network consisting of n nodes arranged in two dimensional grid. In this grid of computing nodes, the nodes on the edges are also directly connected with a single link known as cycles. So, in this topology of nodes all the nodes on horizontal edges are directly connected with another horizontal extreme edge. In the same way, all the extreme top nodes are directly linked with a single links with the nodes of other extreme edge at the bottom. Let ni is the ith node in the network and hij is the path length between the i and j nodes. Np is the total number of processes which are in running state and nodes in the whole application. The communication among the processes is considered in the Np X Np matrix. The enhancement in this is that we managed the super nodes, each super node has supervised the group of virtual nodes or processes. The migration of the processes/VM from one super node to another super node. We considered that each server nodes have same category and capabilities as in EC2. To handle the communication dependencies, we add virtual machines and each group of VM is mainly supervised by the super node. The more there is communication between the nodes and processes the more there is communication dependency among the nodes and processes.

3.2 Problem Formulation

The VM arrangement problem is formally expressed as takes after. Given a system of N super nodes that host a utilization of P procedures (or VMs), reassign the VM/processes to super nodes, such that the application components usage is minimized, with consideration of capacity limitations. As talked about before, the application asset use relies on upon both the execution cost and the cost of communication caused by the VM/processes of an applications. Agreeing to the past documentation, the execution expense of an application can be specifically calculated by summing the computational requests of every VM/processes. Be that as it may, we can't straightforwardly include the information exchanged between the VM/processes to measure the communication cost. This is due to the communication of components of an

application utilization is straightforwardly associated. In this manner, two cases emerge for the correspondence between any pair of VM/processes: (a) the VM/processes are situated on the same super node, and,due to the interprocess correspondence is performed by getting to the local neighborhood memory, the resource utilization is unimportant, and (b) the procedures are situated on various super nodes in which the VM/processes utilization is relative to both the measure of information exchanged and the number of connections required for communication. The previously stated model is utilized by numerous cloud suppliers, for example, Amazon EC2, to charge the aggregate correspondence resource utilization of an application, which is relative to the measure of information sent over the torus network. As

an illustration, consider a communication of components of application utilization amongst pi and pj being equivalent to D*h*y, given that pi and pj transfer D information and they are situated on super nodes Snx and Sny, separately. If there is communication at local level, hxx = 0, which involves that the communication expense is zero. To express the issue through a thorough mathematical calculations, we characterize them as follows. Give F as P N framework catching the assigned super node for every VM/process, with fiSx = 1 if Snx is allocated to pi; generally, fiSx = 0. According to them, given a task F ,total execution of resources use is given by exec(F), as depicted in Eq. (1), while the cost of total communication (called the correspondence or torus network cost/overhead) is signified by comm(F), as portrayed in Eq. (2). From the above, we can conclude that the aggregate asset utilization can be expressed as total(F), which is communicated by Eq. (3). Thusly, the aggregate asset utilization can be minimized by finding a task F such that Eq. (3) is minimized under the imperative that the CPU necessities of the processes facilitated by a super node must not surpass the

aggregate capacity of that super node.

$$exec(F) = \sum_{i=1}^{P} \sum_{x=1}^{N} uifiSx \tag{1}$$

$$comm(F) = \sum_{1=1}^{P+N} \sum_{k=1}^{P+N} \sum_{x=1}^{N} \sum_{y=1}^{N} CikfiSxfkSyhxy \tag{2}$$

$$total(F) = exec(F) + comm(F) \tag{3}$$

Chapter 4

METHODOLOGY OF RESEARCH WORK

This chapter covers methodology of distributed system and the steps to implement the proposed methodology. The steps of proposed methodology are: (a) to apply the minimum cut algorithm [28] then in the last the some of the examples in which (b) migration of the virtual processes/nodes dynamically and (c) make different two types of calculations (1) before migration network cost (2) after migration network cost, (d) placement of components of an application and (e) also calculate benefit in terms of cost and time.

In figure 4.1 shows the methodology of our proposed Efficient Resource Allocation research work. According to the methodology we have to apply the minimum cut algorithm on application graph.

4.1 Procedure of Applying the Minimum Cut Algorithm

The network consist of nodes and processes/VM . The problem of minimum cut is to find the cut having minimum weight from source to the sink node. Minimum cut algorithm consists of following steps which are as follows:

In figure 4.2 the graph having three nodes and n1 has two processes P1 and P2, n2 has one process P3, and n3 has two processes P4 and P5. Apply the min cut algorithm on the application graph having 3 nodes and 5 processes. n1 is the source node and n3 is the sink node.

In figure 4.3 shows step 1 where min cut algorithm starts from source node n1 and get the min cut between (n1)and (P1,P3).In step 1 the weight of the cut is w=5. The sets are as follows: n1, P1, P2, n2, P3, n3, P4, P5.

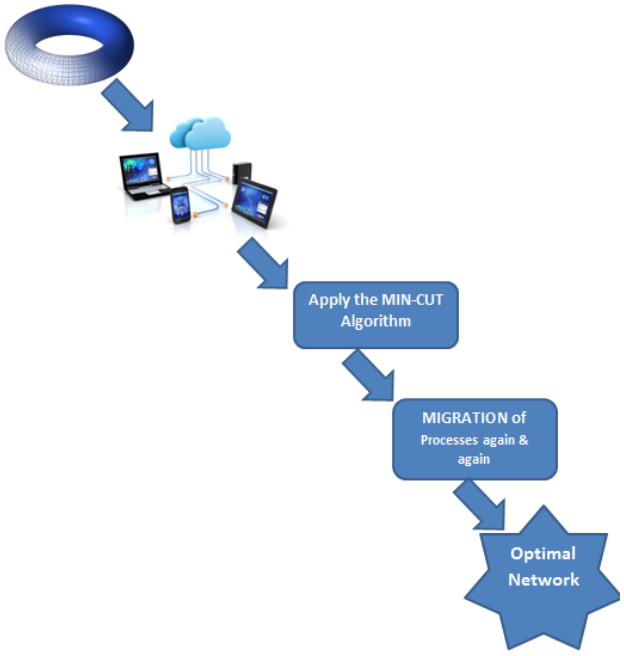

Figure 4.1: Model for research methodology of thesis

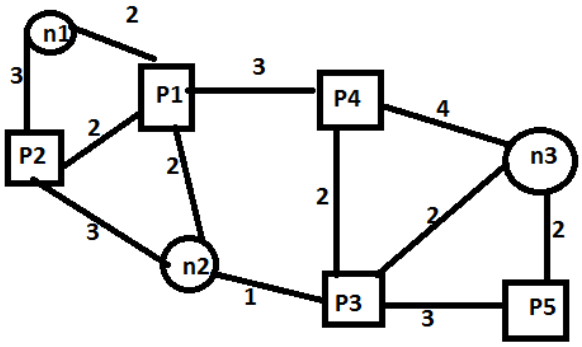

Figure 4.2: Application Graph having 3 nodes and 5 processes

A cut in the graph which divides the nodes into two sets. The weight of the cut is equal to weight of number of edges between two sets.

In figure 4.4 step 2 the weight of the cut is w=5. The sets are as follows: P5, n1, P1, P2, n2, P3, n3, P4, then edges contracted and the selected node (n1) merge with the nearest node to form a new node(n1,p2).

In figure 4.5 step 3, the weight of the cut is w= 7. The sets are P3, P5, n1, P1, P2, n2, n3, P4.The min cut is between (p3,p5)and (n2,n3,p4).

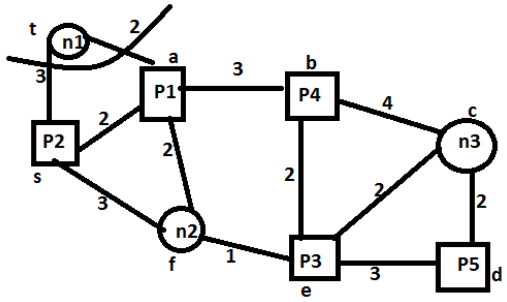

Figure 4.3: Step1: Min cut algorithm starting from source node n1

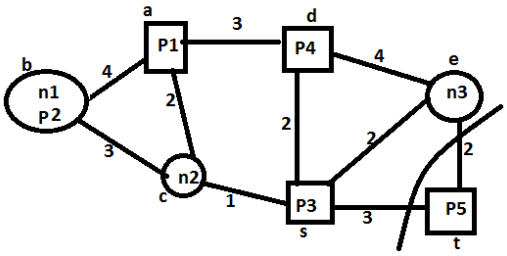

Figure 4.4: Step 2: Process P5 is selected and min cut weight w=5

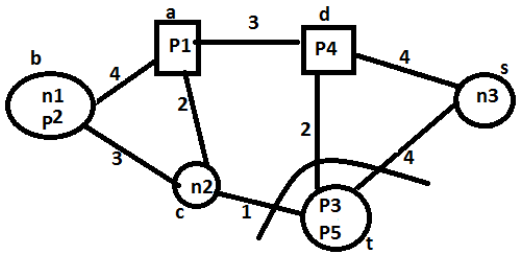

Figure 4.5: Step 3:New node (p5,p3) is selected and min cut weight w=7

In figure 4.6 step 4 the edges of min cut contracted and then merge(p3,p5)with (n5). The weight of the cut is w=7. The sets are as follows: P3, P5, n3, n1, P1, P2, n2, P3, n3, P4, P5.

In figure 4.7 step 5, the weight of the cut is w=4. The sets are: n3, P3, P4, P5, n1, P1, P2, n2.

In figure 4.8 step 6 the weight of the cut is w=7. The sets are: n1, P2, n2, P1, P3, P4, P5, n3.

In figure 4.9 step 7 the weight of the cut is w=9, the sets are: n1, P2, n2, P3, P4, n3, P5, P1

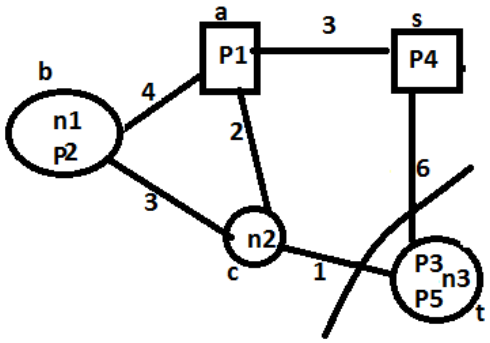

Figure 4.6: Step 4:Node (p3,n3,p5)is selected and min cut weight w=7

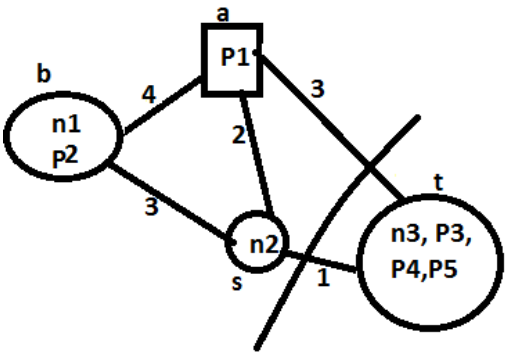

Figure 4.7: Step 5:Node (p3,n3,p4,p5)is selected and min cut weight w=7

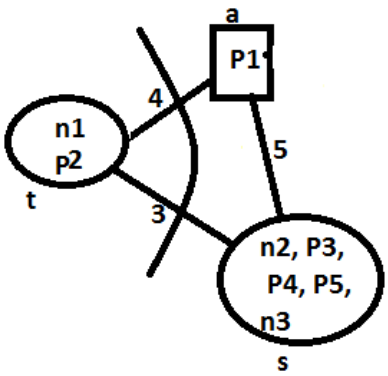

Figure 4.8: Step 6:Node(n1,p2)is selected and min cut weight w=7

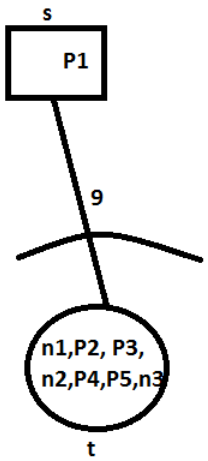

Figure 4.9: Step 7: p1 is selected and min cut weight w=9

Table 4.1: Results of minimum cut algorithm

Steps	cut	Min cut
0		-
1	5	5
2	5	5
3	7	5
4	7	5
5	4	4
6	7	4
7	9	4

Table 4.1 shows the results of all steps of the minimum cut algorithm which are as follows: 4.1

Result: In 4.1 the weight of the minimum cut is 4. This result shows that node 4 can migrate and will give benefit while others are not beneficial to move from their original locations.

4.2 Proposed Efficient Resource Allocator (ERA) Algorithm

In this chapter we proposed an algorithm that accentuates performing light weight figuring, we don't put the reliability of our issue at danger. This depends on our focal thought that we should abstain from running in to attain-

ability issues, for example, attempting to discover the min-cut of an application graph that does not by any means fit into the memory. Therefore, we focus on components that relocate a VM/processes or a group of VM/processes starting with one super node then onto the next, going for the aggregate communication cost decrease. We depict the single VM/processes/nodes migration. A noteworthy downside of the previously stated component is distinguished , driving that component to make imperfect choices. As a solution for the downside, we present the super process migration. In the last segment, we describe how to implement that algorithm.

4.2.1 Single VM/process/node Migration

Our goal is to make migrations that lessens the present torus network cost. In this segment, we consider relocating Processes/VM/nodes as single elements. In this manner, we require a metric to consider whether such a movement contributes adversely or emphatically towards the minimization of the aggregate torus network cost. To characterize such a metric, we initially need to present some additional yet important documentation. Let Mi , sd characterize a relocation of process pi (called target process) from the node ns (nearby node) to a 1-jump neighbour nd (destination node). For any of the previously stated movement, we have to recognize the following.

4.2.2 Steps for Identifying the un-balanced Super Process

- Construct a min cut graph represent real processes
- Add source node ns (source)
- Add destination node nd (destination)
- Remove edges having weight equal to zero
- Min cut algorithm is applied on graph
- Identifying the super process (group of co-located processes)
- Positive load

  This load speaks to the increase (as far as the all out correspondence overhead) while relocating pi from ns to nd. In particular, this movement will convey pi closer by 1 jump to the group of VM/processes that utilization the destination node nd to speak with pi at the point

when the last is situated on ns. Subsequently, the aggregate correspondence overhead will diminish by a sum that is equivalent to the volume of information traded among pi and the VM/processes having a place with source node.

- Negative load

  This load depicts the network cost (as far as the complete correspondence overhead) while relocating a VM/process pi from ns towards its 1-jump neighbour nd. In particular, when relocating process pi from source node (ns) towards destination node (nd), pi will move itself by 1 node from the group of VM/processes that don't utilize the destination node (nd) to speak with pi which finally found on ns. In this manner, the aggregate correspondence overhead will increment by a sum that is equivalent to the volume of information traded between pi and the VM/processes having a place with nd.

- Benefit

  This shows the metric that surveys whether a movement is valuable or not. In particular, the relocation of VM/process pi from ns to nd is viewed as advantageous if (pl) positive load is more than the nl negative load else, it is considered non-gainful. The benefit which states that the relocation Mi will bring about a decline or an expansion in the general framework correspondence overhead by a sum that is equivalent to the subtraction of nl from pl. If the result of taking difference is zero, then Mi won't influence the general correspondence overhead. At the point when the outcome is a positive, then the total correspondence overhead will diminish by a sum equivalent to that esteem; else, it will be expanded.

4.2.3 Examples of the proposed technique to optimize the network

There are some examples in which we apply our proposed methodology of implementing the proposed algorithm. Here we calculate the original network cost then again calculate network cost after executing our proposed algorithm and calculate benefit in terms of cost and time.

4.2.3.1 Example 1

In figure 4.11 Applying the min-cut algorithm on nodes N1 and N2 Applying the min cut max flow algorithm from [29] and create the min cut graph. The

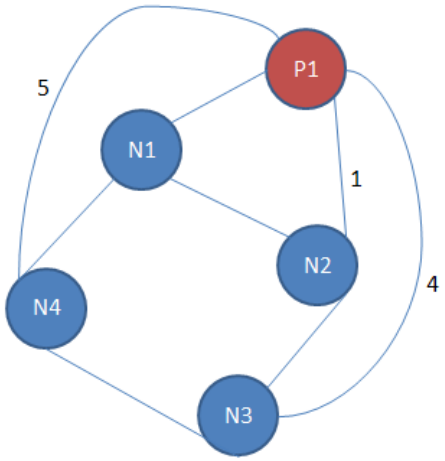

Figure 4.10: Application Graph having 4 nodes and 1 process

min-cut graph is as follows:

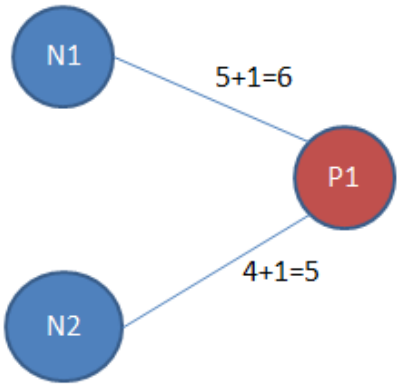

Figure 4.11: Min cut Graph N1-N2

Consider all the possible cases of migration for Process P1:

The original network cost for process P1 at Node N1:

The total original cost of process P1 at Node N1 is :(1*1) +(4*2) +(5*1)= 14.

Decision: (no migration benefit) So we will not move Process P1 to Node N2

In figure 4.12 Applying the min-cut algorithm on nodes N1 and N4

The min-cut graph is as follows:

In figure 4.12 shows applying the min-cut algorithm on nodes N1 and N4. The original network cost for process P1 at Node N1:

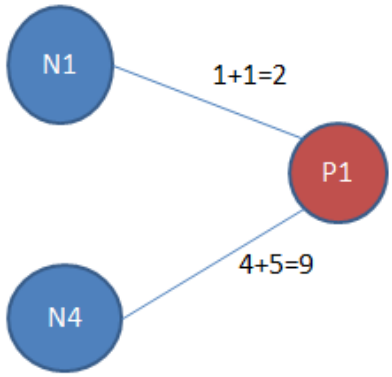

Figure 4.12: Min cut graph N1-N4

The total original cost of process P1 at Node N1 is : (1*1) +(4*2) +(5*1)= 14

The network cost for process P1 after migration to N4 = (5*0) +(1*1)+(1*2) + (4*1)=7

Migration Benefit = 14-7 =7

Result

The process P1 should migrate to the node N4, as it gives the benefit of cost 7

4.2.3.2 Example 2

An application graph having four nodes and three processes is selected. Applying the minimum cut algorithm on N1 and N2.

In figure 4.14 Applying the min-cut between the nodes N1 and N2

The min-cut graph is as follows:

In figure 4.14 Applying the min-cut between the nodes N1 and N2. The result of min cut algorithm is that P1 should migrate.

The original network cost for Process P1 at Node N1 is:

Network cost from N2 to N1 (INCLUDES THE COMBINED WEIGHT)= (6*1) +(1*1)+ (4*2) +(1*2)=17

Network cost from N4 to N1 = (3*1) =3

Total original network cost (SUM OF ALL)=6+1+8+2+3=20

Consider all the possible cases of migration for Process P1:

The network cost for process P1 after migration to N2

If process P1 migrate from N1 to N2 then :

Network cost after migration = (6*0) +(3*2) +(1*1)+(1*1)+ (1*4)=12

Benefit = 20-12=8

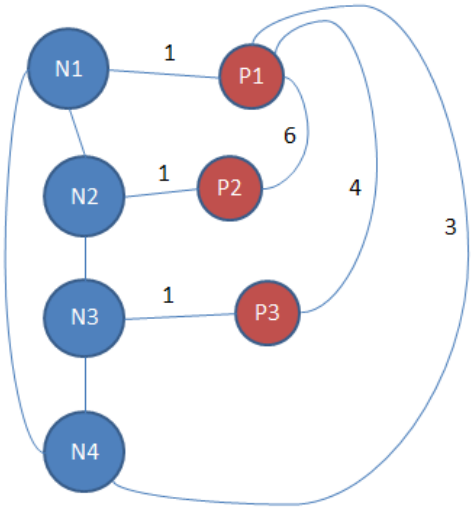

Figure 4.13: Application Graph having 4 nodes and 3 processes

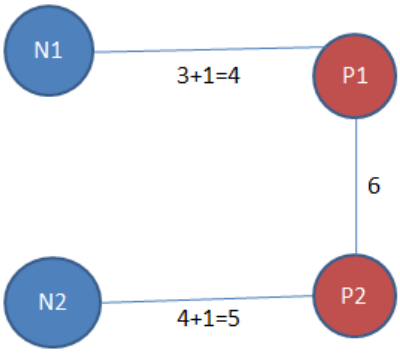

Figure 4.14: Min cut Graph N1-N2

The min-cut graph is as follows: In figure 4.15 shows applying the min-cut algorithm on N1 and N4.

However it is clear from the min cut graph that there is no migration benefit .

The original network cost for Process P1 at Node N1 is:

Total original network cost (SUM OF ALL)=6+1+8+2+3=20.

The network cost for process P1 after migration to N4

Network cost after migration = (6*2) +(3*0) +(1*1)+(1*2)+ (1*4)+(1*1)=20

Migration benefit= 20-20=0 (no migration benefit)

Decision: The Process P1 migrated to node N2

In figure 4.16 shows that process P1 is migrated to node N2.

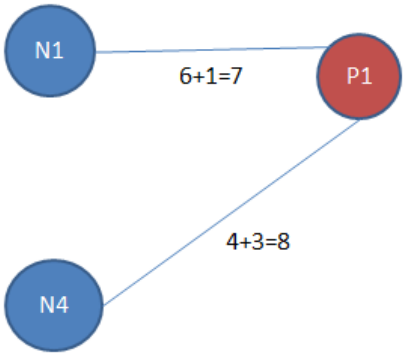

Figure 4.15: Min cut Graph N1-N4

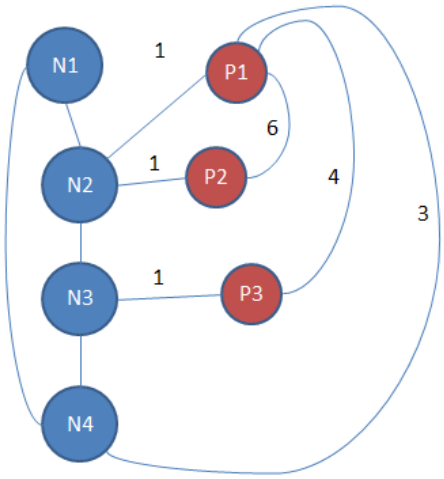

Figure 4.16: Application Graph in which P1 migrated to N2

The min-cut graph is as follows:
In figure 4.17 Applying the min-cut algorithm on N2 and N3
The original network cost for Process P1 at Node N2 is:
The original network cost = (6*0) +(1*1)+(4*1) +(1*1)+(3*2) = 12
The network cost for process P1 after migration to N3
Network cost after migration = (6*1) +(1*1)+(1*1)+(4*0) + (3*1)=11
Benefit : 12-11=1
Decision: The process P1 should migrate to the node N2
The original network cost for Process P2 at Node N2 is:
The original network cost = (6*0)+(1*1)=1
The network cost for process P2 after migration to N3
Network cost after migration =(6*1)+(1*1)+(1*1)=8

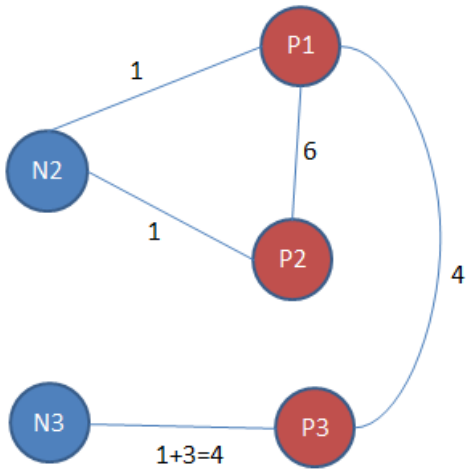

Figure 4.17: Min cut Graph N2-N3

(no migration benefit)

Result: The process P1 should migrate to the node N2 then to node N3 and no migration for process P2.

4.2.3.3 Example 3

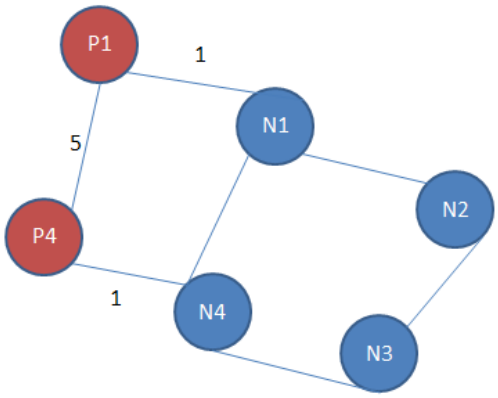

Figure 4.18: Application Graph having 4 nodes and 2 processes

Figure 4.18 shows the application graph having four nodes and two processes.

The min-cut graph is as follows:

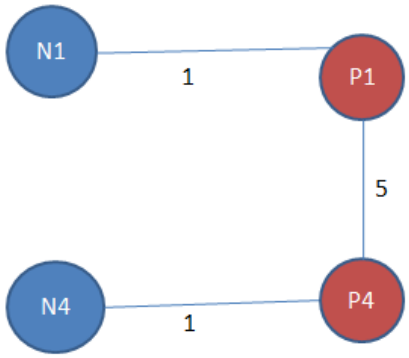

Figure 4.19: Min cut Graph N1-N4

In figure 4.19 Applying the min-cut between the nodes N1 and N4
The process P1 should migrate.
The original network cost for Process P1 at Node N1: = (5*1)=5
After migration of P1 to node N4 = 5*0+(1*1) = 1
Benefit = 5-1=4
Result:
So, P1 should migrate to N4.

4.2.3.4 Example 4 : (3 x 3) Torus Network

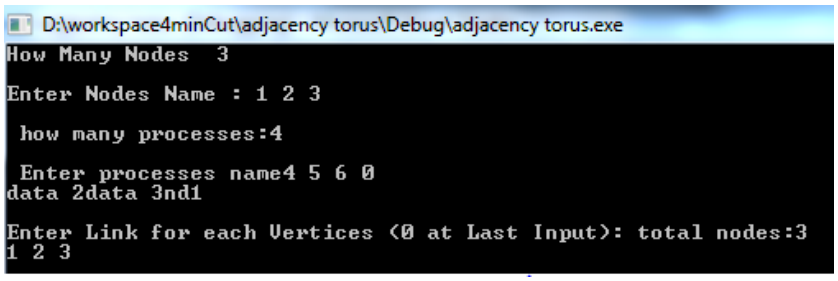

Figure 4.20: Input for 3x3 Torus Application Graph

In figure no. 4.20 , here I take the input from the user that how many nodes = 3 and their names 1 , 2 , 3 Then how many processes then their names and end with terminator zero.

Then, adjacency matrix of the whole graph is created which shows the links and weights of all the nodes and their processes. According to figure no.4.21 the 12 number of edges are created , these edges have the information about their source , destination and their weights as (edge src :1, edge dest

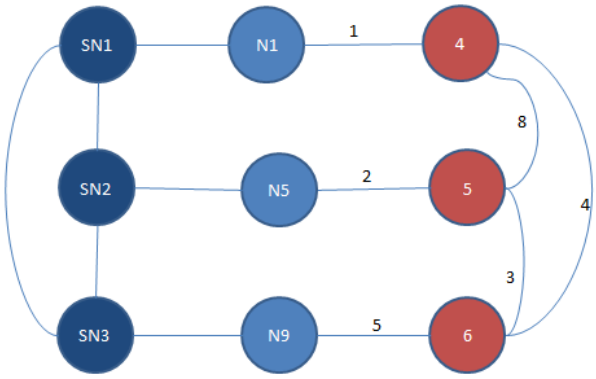

Figure 4.21: Application Graph having 3 super nodes

: 4 ,w: 1). Then min cut graph is created of first two nodes 1,2 and their respective processes. After applying the mincut algorithm on this min cut graph the , the result of the min cut algorithm is (1-4) means first node and its process named 4.

Figure 4.21 shows the output console of our proposed algorithm. Now applying the min cut algorithm on node SN2 and node SN3. The result of min cut graph is (SN2-5) means node SN2 and its process 5. Now we calculate the network cost for process 5. In above figure no 4.22 the network cost of process 5 at node SN2 is 3. Next we calculate the migration cost of process 5.

As the process 4 is migrated to node SN2, now the network cost of process 4 at node SN2 is 5 and network cost of process 5 at node SN2 is 3. As shown below in figure no 4.23, so, the total cost at node SN2 is 5+3=8. Then the after migration cost of process 4 and process 5 at node 3 are calculated. The network cost of process 5 at node SN3 is 2 and the network cost of the process 4 at node SN3 is 1. So, the total cost at node SN3 is 3.

Calculating the networking cost for both 4 and 5 at SN2:
The network cost for 4 at SN2 =(1*1)+(8*0)+(4*1) =5
The network cost for 5 at SN2 =(2*1)+(8*0)+(3*0) =3
Total cost for 4 and P5 at SN2 = 5+3 = 8
Calculating the networking cost for both 5 and after migration to SN3:
The network cost for 4 at SN3 =(1*1)+(8*0)+(4*0) =1
The network cost for 5 at SN3 =(2*1)+(8*0)+(3*0) =2
Total cost for 4 and 5 at SN3 = 3
Thus the migration benefit is 8-3 =5 for process 4 and 5.
Overall the result is that process 4 and process 5 is migrated to node 3

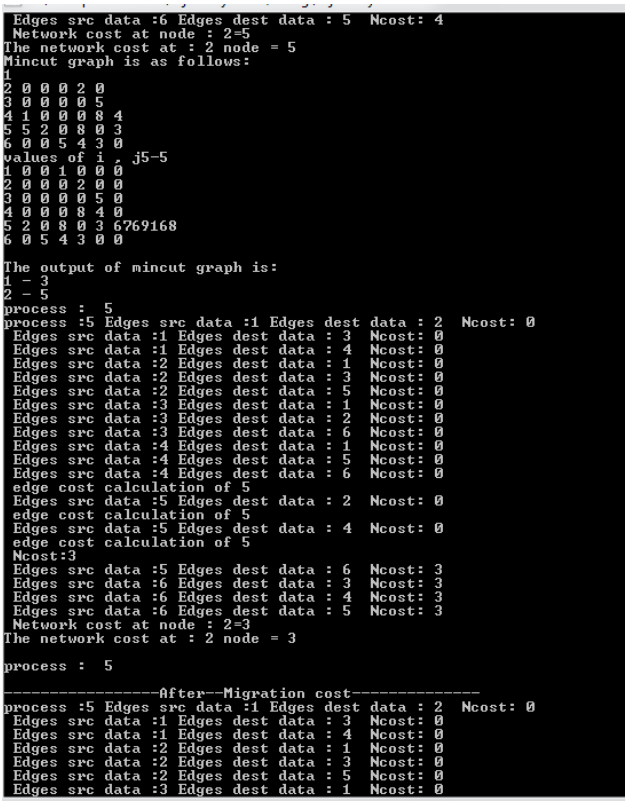

Figure 4.22: Output console of our proposed algorithm :process 5 is selected

and benefit is 15 in terms of network cost

4.2.3.5 Example: 5 (5x5 Torus Network)

P1 is located on N1
P9 is located on N9
P20 is located on N20
P24 is located on N24
(P1, N1):1 (i.e., communication dependencies between P1 and N1)
(P9, N9):10
(P20, N20):5
(P24, N24):10
(P1, P9):8
(P1, P20):1
(P9, P20):1
(P20, P24):10
GRID

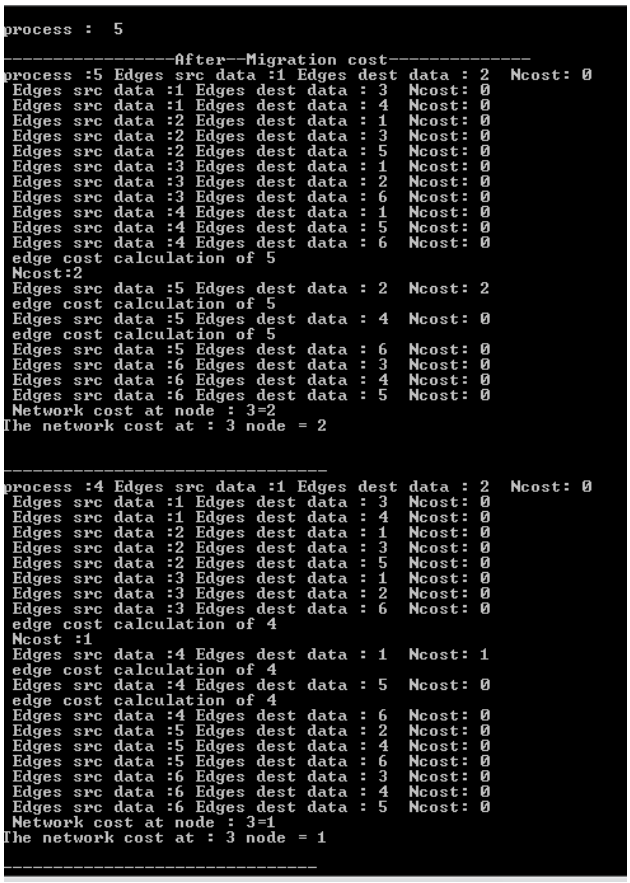

Figure 4.23: Min cut between (SN2-5)process 5 is selected

Figure 4.24 shows grid consists of rows and columns having nodes and processes. The application graph of grid have 5 super nodes an 4 processes. Figure 4.25 shows the application graph consist of 5 super nodes, 4 nodes and 4 processes.

Applying Min Cut on Super Column SN1, SN2, SN3, SN4, SN5

First applying the algorithm on super node SN1 and SN5:

The first min cut is between N1 and P1. So, P1 should migrate.

Min cut graph is as follows:

In above figure no. 4.26, weight 8 comes from (P1, P9), weight 10 comes from (P20, P24), and 1 comes from (P20, P9).

Calculating the network cost for process P1 at node SN1:

The original network cost for P1= (1*0) + (2*8) + (1*1) =16+1=17.

Network cost after migration of P1 to SN5:

The network cost for P1 at SN5 = (1*1) + (1*8) + (1*0) =9

SN1	N1 P1	N6	N11	N16	N21
SN2	N2	N7	N12	N17	N22
SN3	N3	N8	N13	N18	N23
SN4	N4	N9 P9	N14	N19	N24 P24
SN5	N5	N10	N15	N20 P20	N25

Figure 4.24: Application grid having 5 super nodes and 4 processes

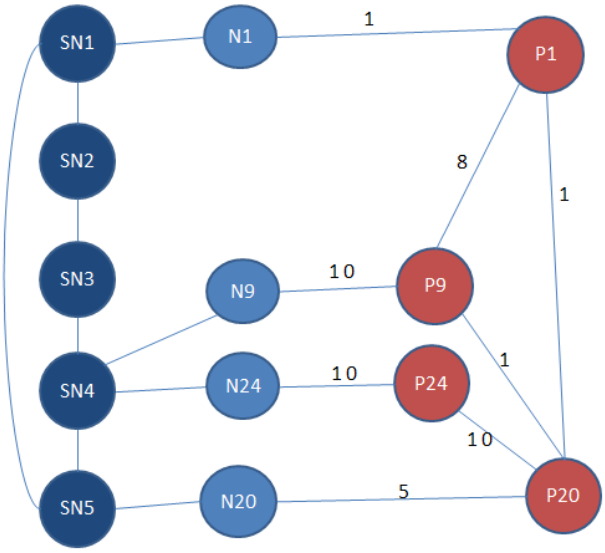

Figure 4.25: Application Graph having 5 super nodes and 4 processes

Benefit =17-9=8
Result
So, P1 should migrate to super node SN5 with migration benefit of 8.
Again apply the algorithm on super nodes SN5 and SN4:
The min cut graph is as follows
Figure 4.27 shows the min cut super nodes SN5 and SN4.

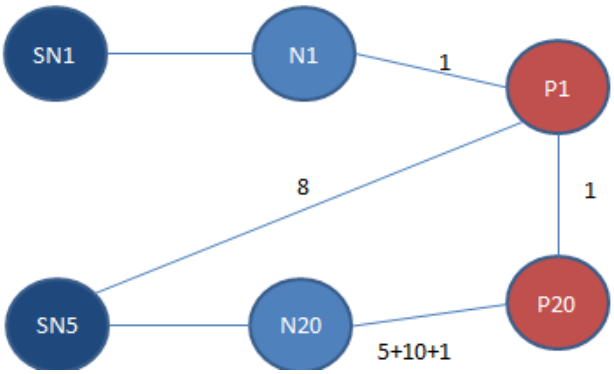

Figure 4.26: Min cut Graph of SN1-SN5

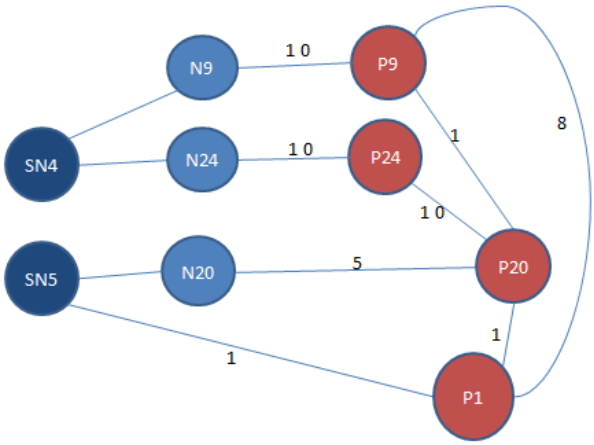

Figure 4.27: Min cut Graph of SN4-SN5

The min cut is between (SN5, P1) AND (P20, P1)
Calculating the network cost for process P1, P20 at node SN5:
Network cost for P1 at SN5 = (1*1) + (8*1) + (1*0) =9
Network cost for P20 at SN5 = (5*0) + (10*1) + (1*1) + (1*0) = 11
Total network cost of P1, P20 at SN5 =9+11 = 20
Calculating the network cost for process P1, P20 at node SN4:
Network cost of P1 at SN4 = (1*2) + (8*0) + (1*1) = 3
Network cost of P20 at SN4 = (5*1) + (10*0) + (1*0) + (1*0) = 5
Total network cost of P1, P9, P20 at SN4 =3+5 = 8
Benefit =20-8=12
Result: P1, P20 is migrated to SN4.
Figure 4.28 shows grid of 5x5 torus application graph.

SN1	SN6	SN11	SN16	SN21
N1 **P1**	N6	N11	N16	N21
N2	N7	N12	N17	N22
N3	N8	N13	N18	N23
N4	N9 P9	N14	N19	N24 **P24**
N5	N10	N15	N20 **P20**	N25

Figure 4.28: Grid of 5x5 Torus Application Graph

Applying Min Cut on Super ROW SN1, SN6, SN11, SN16, SN21

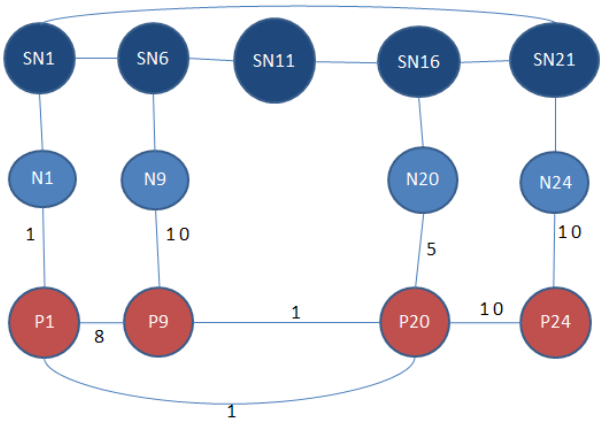

Figure 4.29: 5x5 Torus Application Graph

After applying the algorithm on super node SN1 and SN6: Min cut graph is as follows for SN1 and SN6:

Figure 4.30 shows min cut graph of SN1 and SN6. *The min cut is between (SN1, P1) AND (SN1, P9)*

Calculating the network cost for process P1, P9 both at node SN1

Network cost for P1 at SN1 = (1*0) + (8*0) + (1*2) =2

Network cost for P9 at SN1 = (10*1) + (1*2) + (8*0) = 12

Total cost for P1 and P9 at SN1 = 12 + 2 = 14

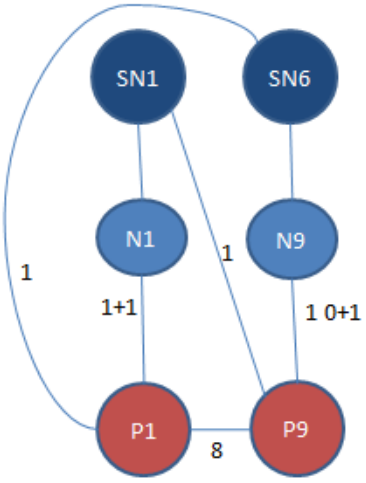

Figure 4.30: Min cut Graph from SN1-SN6

Calculating the network cost for process P1, P9 both at node SN6
Network cost for P1 at SN6 = (1*1) + (8*0) + (1*2) =3
Network cost for P9 at SN6 = (10*0) + (1*2) + (8*0) = 2
Total cost for P1 and P9 at SN6 = 3 + 2 = 5
Benefit = 14-5=9
Result:P1 is migrated to SN6
After applying the algorithm on super node SN6 and SN11:

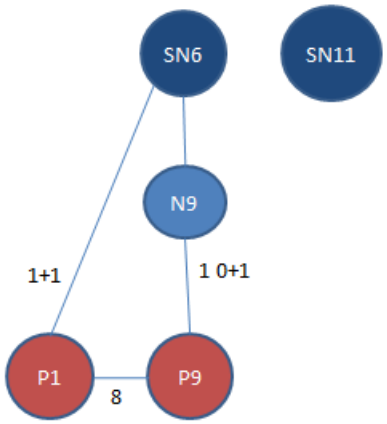

Figure 4.31: Min cut Graph SN6-SN11

Figure 4.31 shows min cut graph of SN6 and SN11. No min cut, no

migration

Again apply the algorithm on super nodes SN16 and SN21:

Min cut graph is as follows:

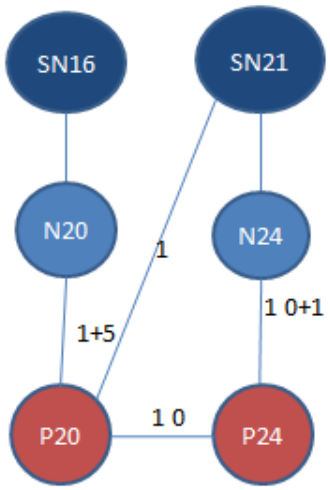

Figure 4.32: Min cut Graph SN16-SN21

Figure 4.32 shows min cut graph of SN6 and SN21. The min cut is between (SN16, P20)

Calculating the network cost for process P20 at node SN16:

Network cost for P20 at SN16 = (1*2) + (5*0) + (1*2) + (10*1) =14

Total cost for P20 at SN16 = 14

Calculating the network cost for process P20 at node SN21:

Network cost for P20 at SN21 = (1*2) + (5*1) + (1*1) + (10*0) =

Total cost for P20 at SN21 = 8

Benefit= 14-8=6

Result

P20 is migrated to SN21. Again apply the algorithm on super nodes SN1 and SN21:

Min cut graph is as follows:

Figure 4.33 shows min cut graph of SN1 and SN21. No min cut No migration benefit

Conclusion:

P1 should migrate to (SN4)4rth row, 2nd column (SN6) So, P1 is migrated to node N9 P20 migrated to 4rth row (SN4), 5th column (SN21) So, P20 should migrate to node N24

Total network cost of all the processes = 1+1+8+10=20

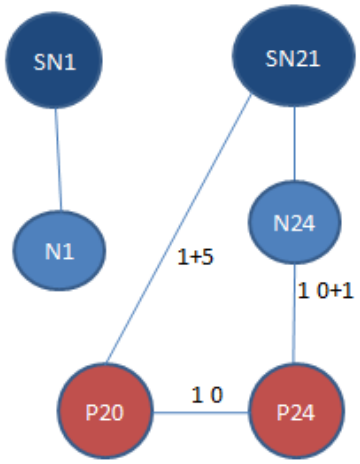

Figure 4.33: Min cut Graph SN1-SN21

Total network cost of all the processes after migration =1+1= 2
Benefit = 20-2=18

Chapter 5

EVALUATION OF RESULTS

In this chapter we have compared self-adjusting algorithm [27] with our proposed approach. The comparison and evaluation of these algorithms are as follows:

5.1 Initial cost of network before execution of algorithms

Calculate the initial network cost of self adjusting and our proposed algorithm. The initial network cost is actually the original cost of torus network which it bears to run an application.

Initial network cost of both the self adjusting algorithm and proposed ERA algorithm is 227564.

5.2 Final network cost after execution of algorithms

By applying the algorithm on network the communication overhead and network cost due migration is reduced. The network cost of proposed ERA algorithms after execution is less than self adjusting algorithm.

In figure 5.1 shows Network cost after execution of both algorithms.

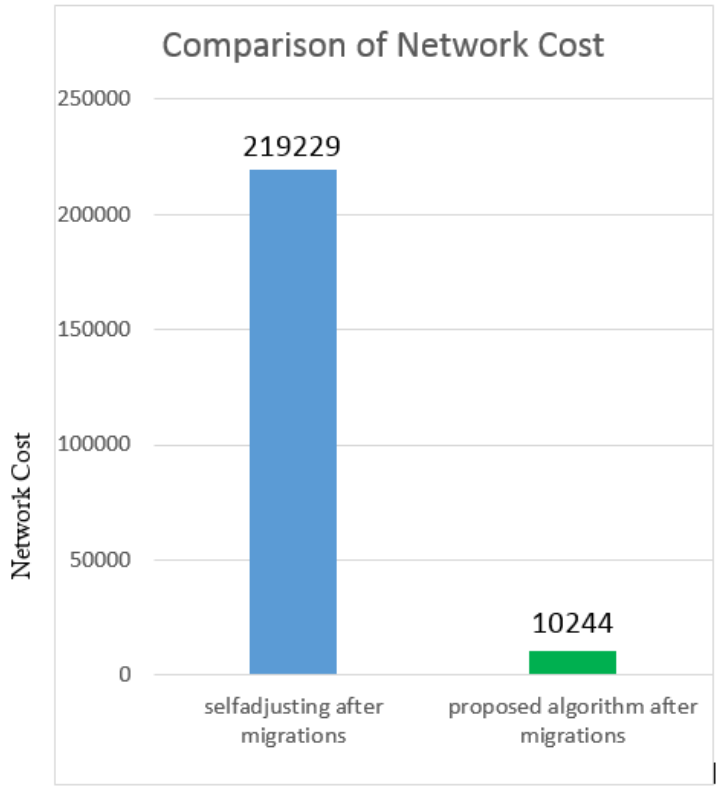

Figure 5.1: Network Cost after execution of both algorithms

5.3 Benefit in terms of communication cost of self- adjusting algorithm

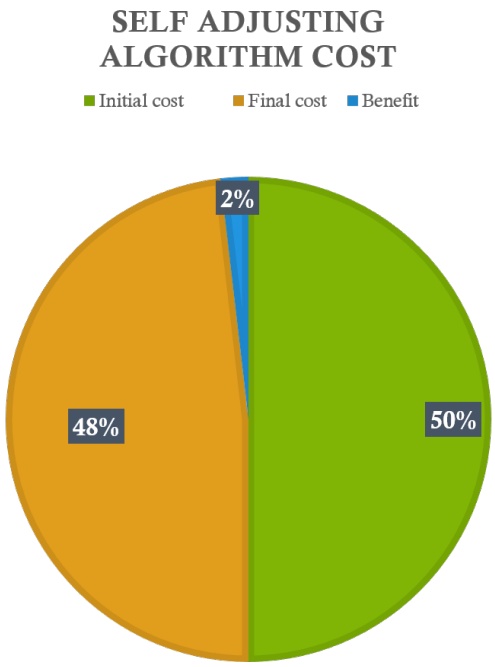

Figure 5.2: Self Adjusting algorithm

In figure 5.2 Initial cost before executing the self adjusting algorithm = 227564

Final cost after executing the self adjusting algorithm = 219229

Difference between the initial and final network cost = 8335

5.4 Benefit in terms of communication cost of proposed ERA algorithm

In figure 5.3 shows benefit by our proposed ERA algorithms.

In figure 5.3 Initial cost before executing the proposed ERA algorithm = 227564

Final cost after executing the proposed ERA algorithm = 10244

Difference between the initial and final network cost = 217320

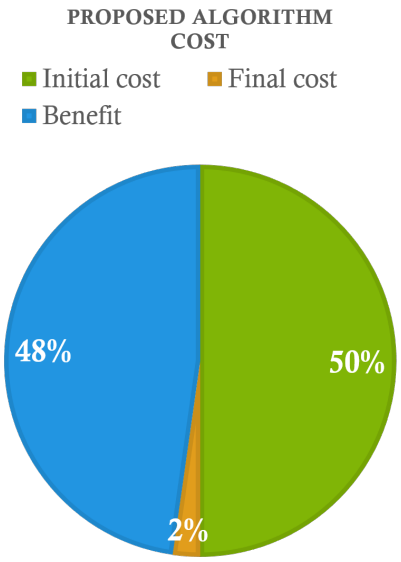

Figure 5.3: Graph of proposed algorithm

5.5 Comparison of ERA and self adjusting algorithm in terms of benefit

Benefit in terms of network cost of self adjusting algorithms is = 8335

Benefit in terms of network cost of proposed ERA algorithms is = 217320

ERA based proposed algorithm give 208985 more benefit than self adjusting algorithm.

In figure 5.4 taking some points of migration of both of the algorithms having same scenario and compares the difference in their costs at different points. The space between upper and lower graph line.

In figure 5.7 Line graph after applying self-adjusting algorithm

At run time during migration of proposed algorithm: with before migration cost and after migration costs.

Figure 5.6 and 5.8 shows lines graph of proposed algorithm with no migration and after migrations respectively.

5.6 Results of different applications on torus network

At run time the results of proposed algorithm: with no migration and after migrations.

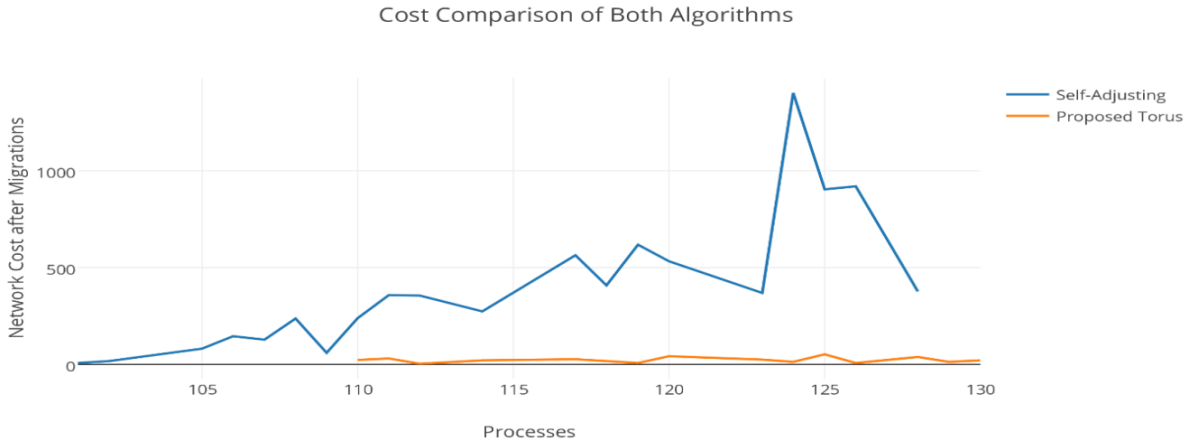

Figure 5.4: Cost Comparison of both algorithms

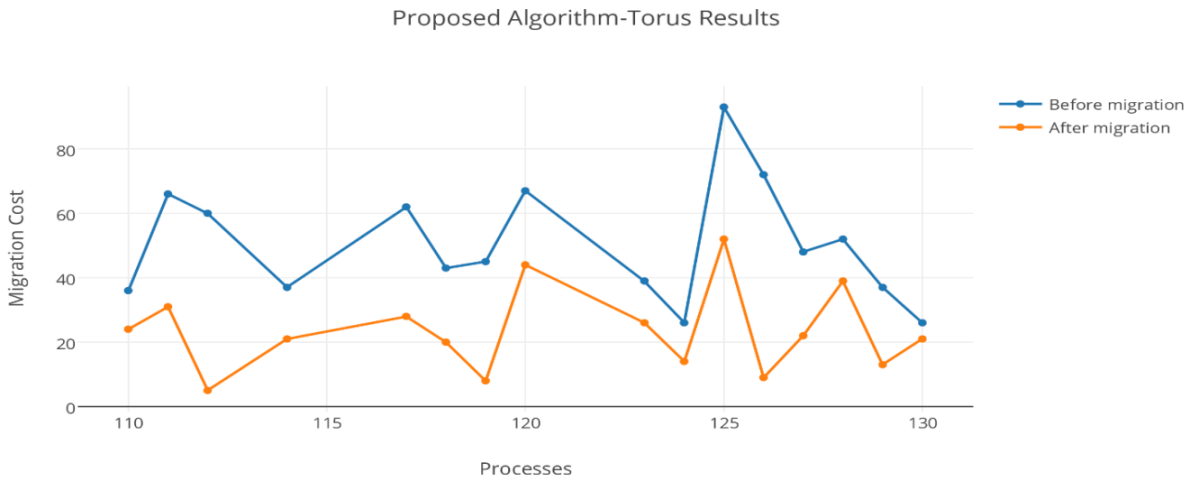

Figure 5.5: Proposed ERA algorithm

In figure 5.9 shows the results of 17 x 17 torus network.

At run time the results of proposed algorithm 17x17: with no migration and after migrations

Figure 5.10 shows the results of proposed algorithm: with no migration and after migrations

Figure 5.11 shows the graph of 20x 20 torus network having 400 nodes. Result of 13 x 13 Torus having 169 nodes. Shows the Result in Graphs of

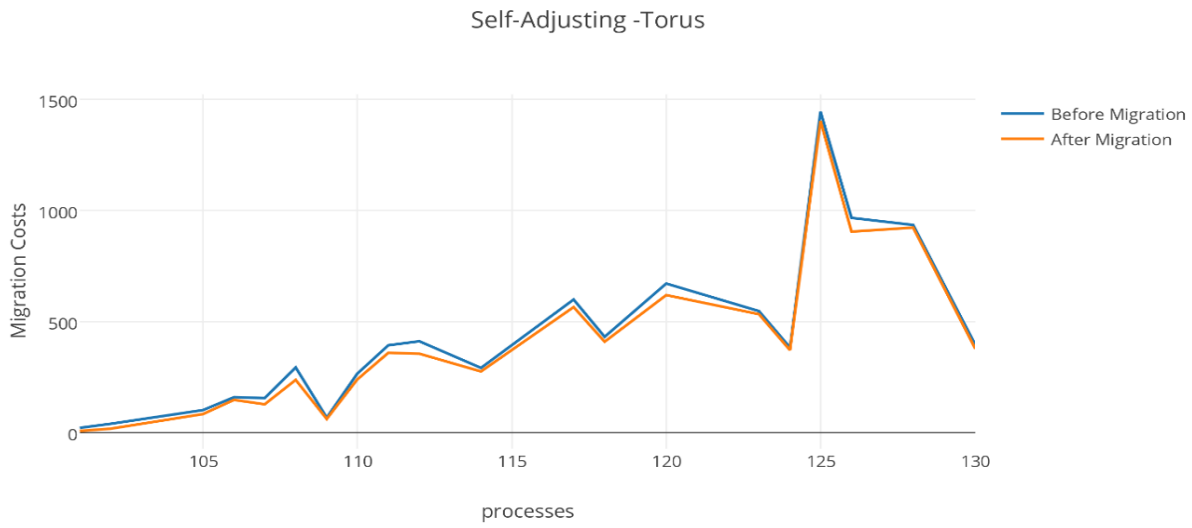

Figure 5.6: Self Adjusting Algorithm

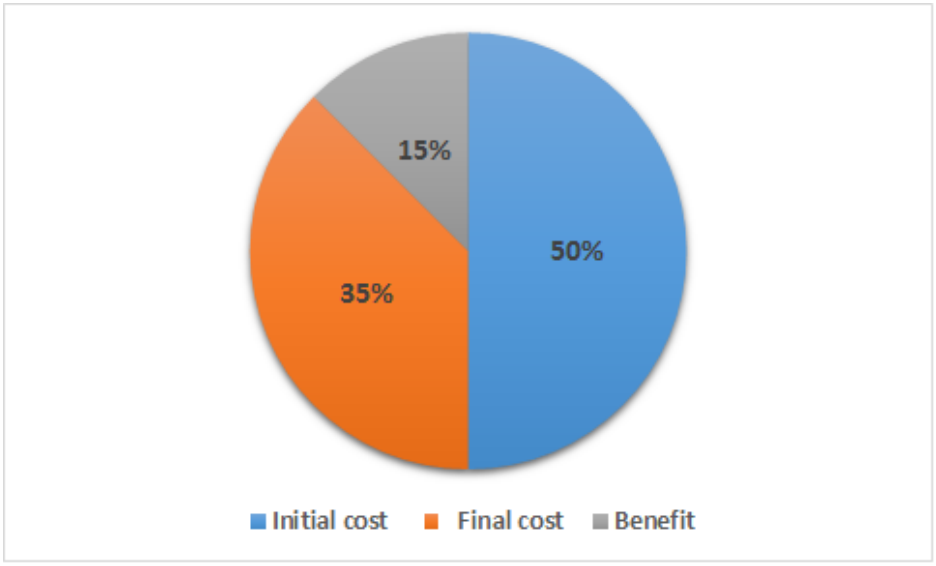

Figure 5.7: Proposed ERA Algorithm

Higher and Complex Examples on Torus

13 x 13 Application of Torus Network

Figure 5.14 shows the communication cost for topology of 13x13 torus having 169 nodes. This topology has 13 super nodes, having 9 processes, only 2 processes can migrated and give cost benefit of cost 9.

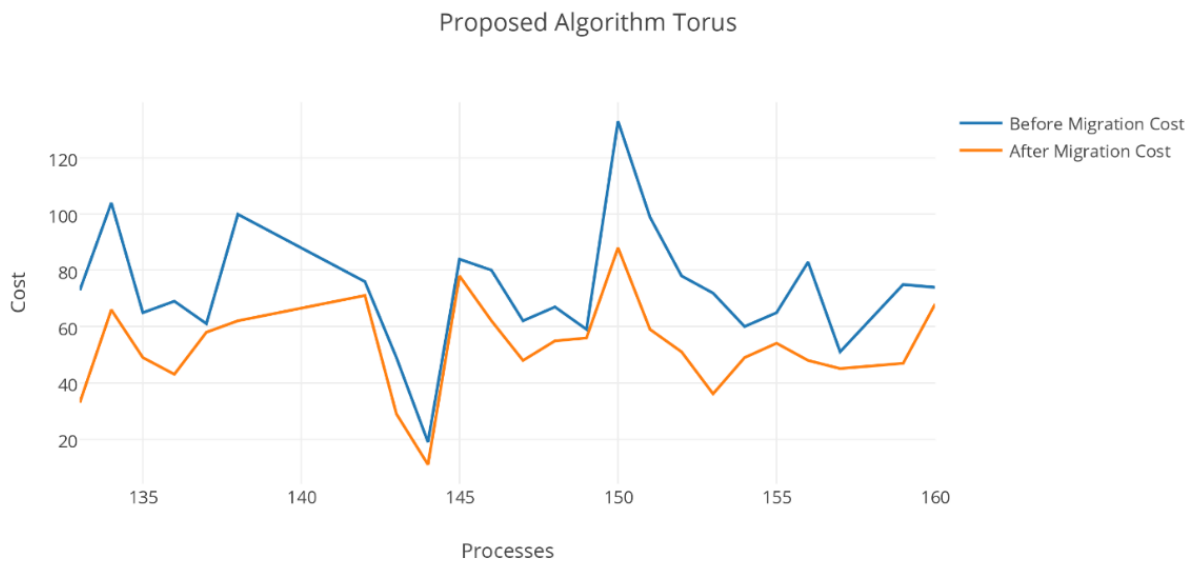

Figure 5.8: Results of Proposed ERA Algorithm

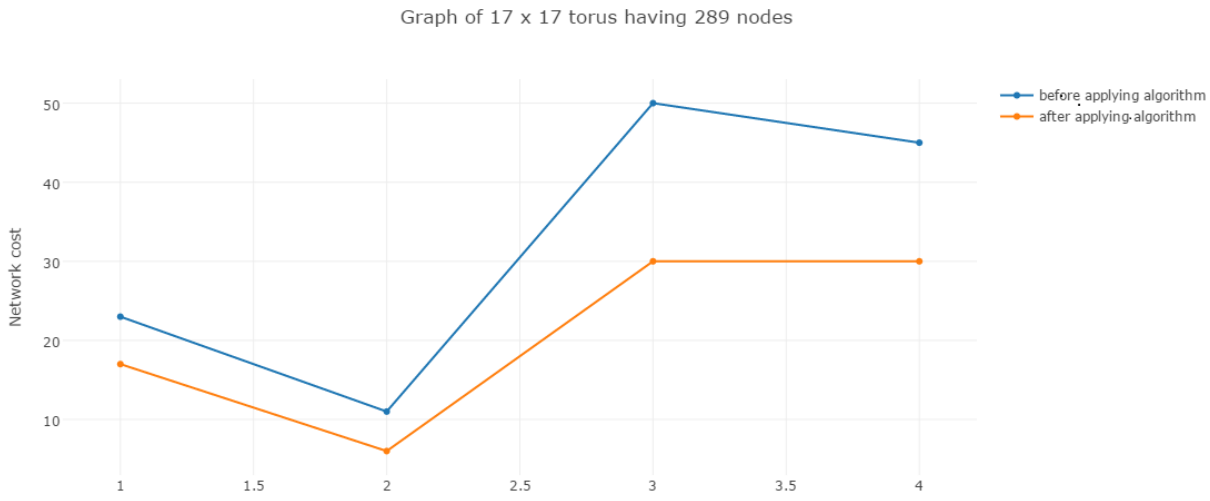

Figure 5.9: Graph of (17 x 17) Torus Network

Figure 5.15 shows the communication cost for topology of 13x13 torus network.

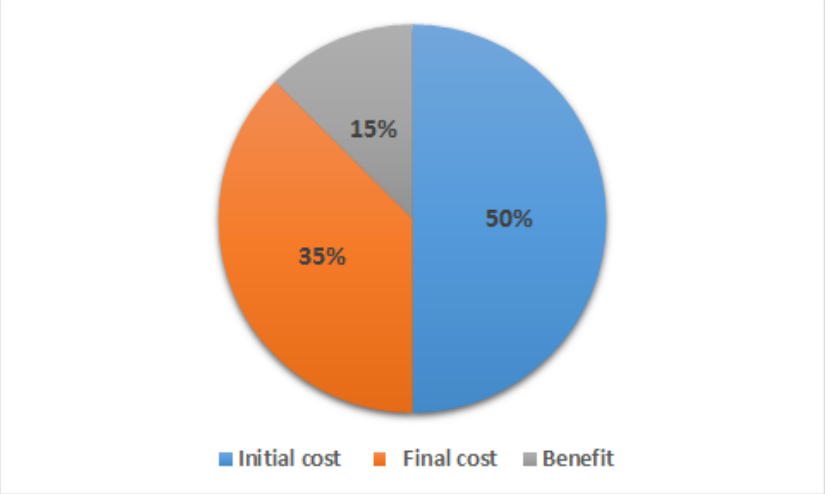

Figure 5.10: Result of Proposed Algorithm

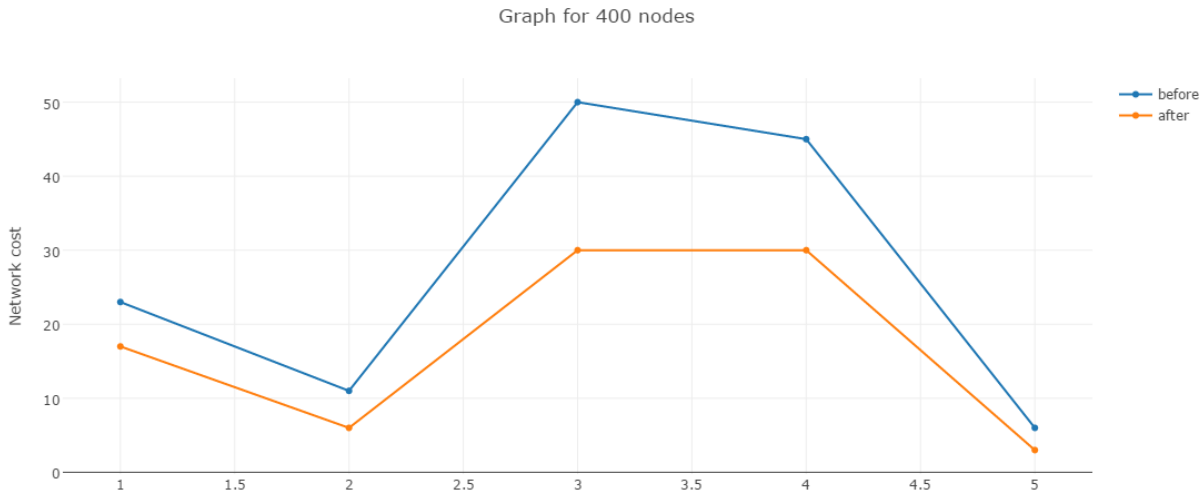

Figure 5.11: Proposed ERA algorithm for 400 nodes of torus network

17 x 17 Application of Torus Network Figure 5.16 shows the communication cost for topology of 17x17 torus network having 289 nodes. This topology has 17 super nodes, having 14 processes over different nodes, 4 processes can migrated and give benefit in terms of cost that is 46. Results of 20 x 20 Torus Network

Figure 5.17 shows the communication cost for topology of 20x20 torus

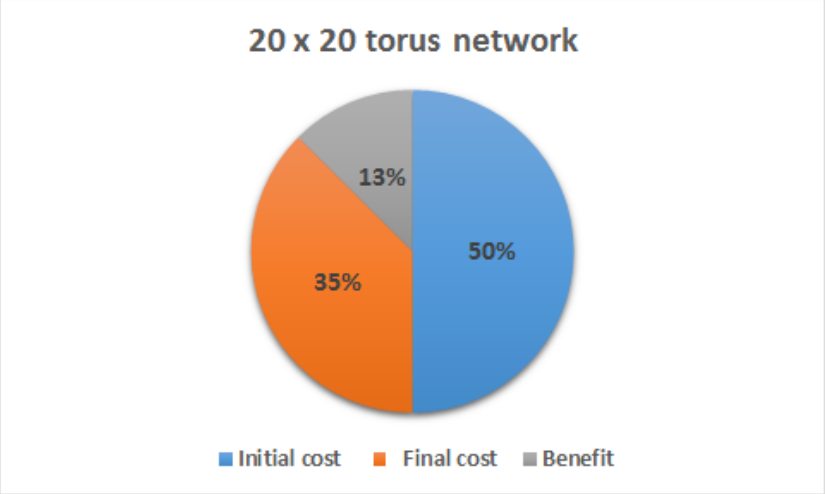

Figure 5.12: Result of 20 x 20 network with ERA algorithm

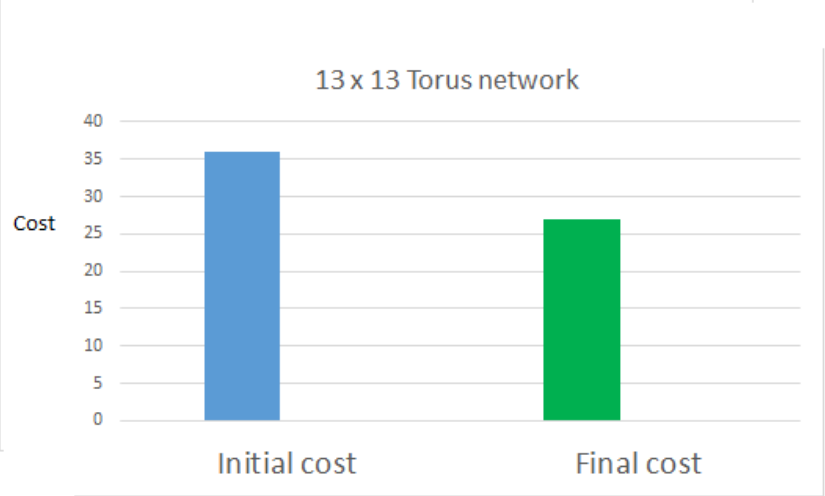

Figure 5.13: Result (13 x 13) Torus Network

having 400 nodes. This topology has 20 super nodes, having 16 processes over different nodes, 5 processes can migrated and give benefit in terms of cost value is 49.

5.7 Conclusion:

From experimental evaluation of both algorithms on torus we concluded that our proposed ERA algorithm is 48 percent more beneficial than self adjusting

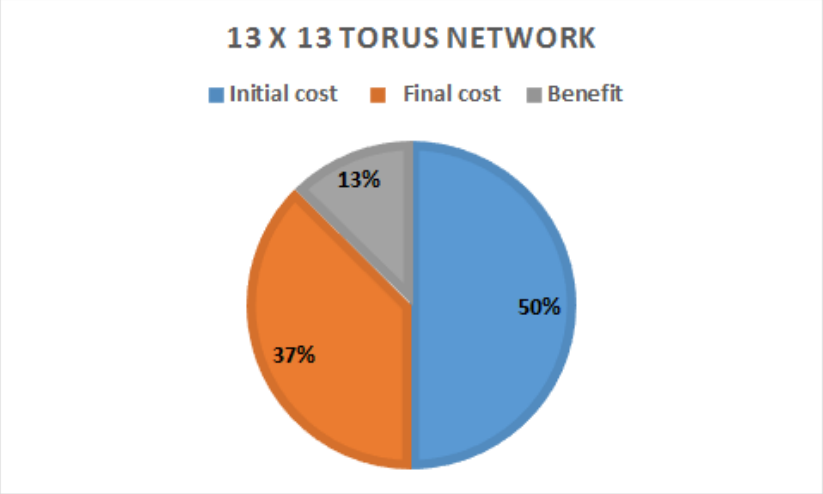

Figure 5.14: Results (13 x 13) Torus Network

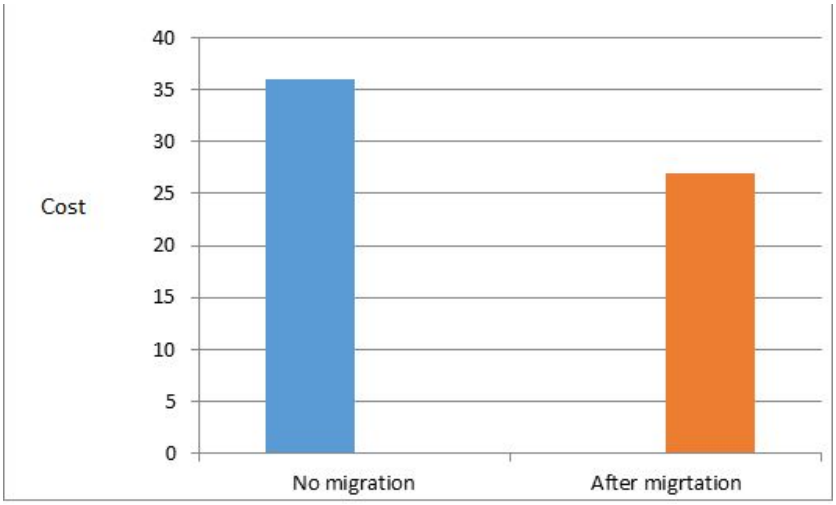

Figure 5.15: Results of Application Graph (13 x13)Torus Network

algorithm. It is proved through experiments that ERA algorithm is an efficient resource allocation algorithm as it reduces the communication overhead. hence the total consumption of cloud resources on torus is also minimized.

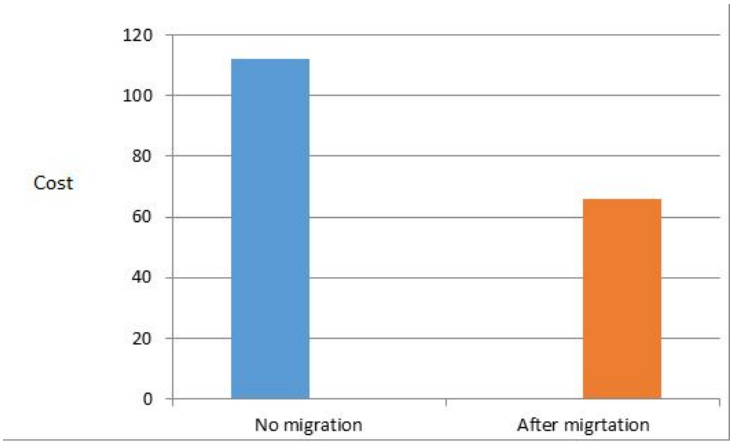

Figure 5.16: Results of (17 x 17) Torus Network

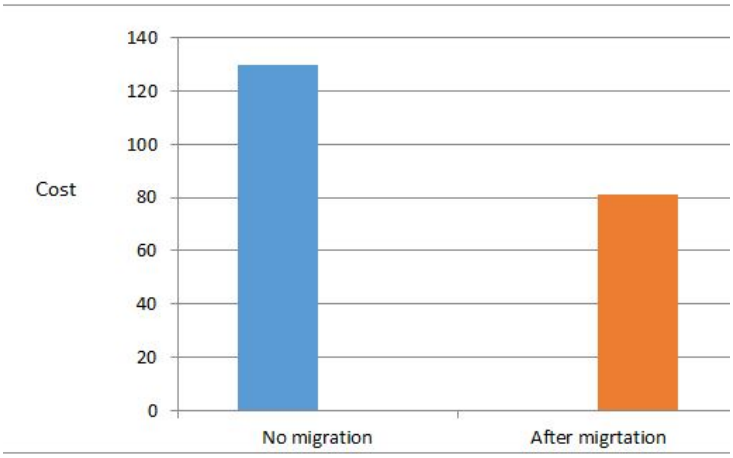

Figure 5.17: Results of (20 x 20) Torus Network

Chapter 6

CONCLUSION AND FUTURE WORK

In high performance applications there is excessive use of HPC resources. The owners of the application have to pay for the consumption of these resources. So, there should be a dynamic re-assignment scheme for these HPC resources. The objective is to reduced the communication among processes and thus optimize network usage. The main contribution of our proposed research is the development of an algorithm for efficient resource allocator for HPC applications. The problem is to minimize the consumption of resources of an application (which is already deployed over cloud distributed system)during its execution on torus network. The solution of the problem is provided by proposing the ERA algorithm. In this algorithm the HPC resources are dynamically re-assigned and make migrations of computing resources. So, that we get the benefit in terms of network cost and reduce the communication overhead. We make evaluation and comparison of our proposed (ERA)algorithm with self adjusting algorithm on torus network. We described the theocratical analysis of our proposed algorithm and also make experimental evaluation of ERA algorithm and prove that this algorithm is optimal for HPC applications on torus in distributed environment. we achieve 48 percent reduction in network cost comparable to self adjusting algorithm.

Major aspects of our proposed research work is summarized as follows:

- Handling the issue in a completely rapid and disseminated way.
- Making just local neighborhood decisions that cause insignificant overhead of framework control messages.

- Identifying the interdependencies between the processes of the specified application.
- Proving that calculation dependably brings about an ideal arrangement of nodes/processes/VM.
- Guaranteeing that the proposed solution dependably always converges.
- Proposing strategies to handle the situations where the super-nodes inside the framework are over-burden.
- Developing a scheme that can similarly work well with trees and various leveled networks also.

6.1 Future Work

We observed the migration of super process is difficult to handle dynamically with respect to CPU utilization. so as future work there is a plan to extend our system model for reducing the CPU utilization and energy consumption on torus network.

Bibliography

[1] Y. W. a. M. B. Blake, "Service-oriented computing and cloud computing," *IEEE Computer Society, University of Notre Dame*, december 2010.

[2] M. O.-K. I. A. a. L. M. M. S. Bani-Mohammad1., "A fast and efficient processor allocation strategy which combines a contiguous and non-contiguous processor allocation algorithms," *gla.ac.uk*, 2007.

[3] Y. Z. X. S. R. H. W. W. J. Z. X. Zhang, "Load balancing algorithm based virtual machine dynamic migration scheme for datacenter application with optical networks," *7th International ICST Conference on Communications and Networking in China (CHINACOM), Beijing, 100876, P.R.China*, 2012.

[4] h. s. dawid zydek, "Fast and efficient processor allocation algorithm for torus based chip microprocessors," *science Direct*, 2011 january.

[5] . K. Q. H.-O. L. J.-S. K. J. S. W. K., "Some properties and algorithms for the hyper-torus network," *Supercomput, J.-S. Kim et al*, 2014.

[6] U. o. F. . G. F. . S. O. a. A. George HCS Research Lab, ECE Dept., "Multicast performance analysis for high-speed torus networks," *HCS Research Lab, ECE Dept., University of Florida, 32611, Gainesville, FL*, 2002.

[7] C. . W. L. Dong Xiang ; Hunan Electric Power Corporation, "An efficient adaptive deadlock-free routing algorithm for torus networks," in *IEEE Transactions on Parallel and Distributed Systems*, 2012.

[8] . U. K. . S. J. S. S. A. M., . K. B. . H., "Energy-efficient data centers," *Springer*, no. 973994, 2012'Abbottabad 22060.

[9] J. D. a. Y. Negishi1, "Overlapping methods of all-to-all communication and fft algorithms for torus-connected massively parallel supercomputers," *IEEE Xplore Tokyo IBM Research*, 2010.

[10] T. L. S. L. . P. L. N. Tziritas, "Gral: A grouping algorithm to optimize application placement in wireless embedded systems," *IEEE International Parallel & Distributed Processing Symposium*, 2011.

[11] R. P. a. Y. D. M. Peng Zhang, "Interlacing bypass rings to torus networks for more efficient networks," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS,*, Feb 2011.

[12] H. K. Y. J. E. J. K. B. Azeez, "I/o node placement for performance and reliability in torus networks," *citeSeerX, Texas A&M University College Station, TX, USA*, 2013.

[13] M. M. B. a. B. Bose, "Resource placement in torus-based networks," *Department of Computer Science Oregon State University*, 2000.

[14] J. G. R. R. a. A. C. Jason Sonnek, "Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration," *9th International Conference on Parallel Processing, Minneapolis*, 2010.

[15] A. B. a. R. Buyya, "Energy efficient resource management in virtualized cloud data centers," *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Australia*, 2010.

[16] C.-Z. X. T. L. N. T. S. U. Khana, "On minimizing the resource consumption of cloud applications using process migrations," *J. Parallel Distrib. Comput.*, vol. vol. 73,, pp. pp. 1690–1704, 2013.

[17] J. S. B. Linnert, B. G. Tech. Univ. Berlin and L.-O. Burchard, "Mapping algorithms optimizing the overall manhattan distance for pre-occupied cluster computers in sla-based grid environments cluster, cloud and grid computing (ccgrid)," *14th IEEE/ACM International Symposium*, 2014.

[18] M. A. E. Z. S. Lo, "Design and analysis of schedules for virtual network migration," *IEEE networking conference, Atlanta*, 2013.

[19] . A. B. a. J. A. Rajkumar Buyya1, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *arxiv.org,1006.0308,Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Manjrasoft Pty Ltd, Australia*, 2010.

[20] T. L. M. Alicherry, "Net work aware resource allocation in distributed clouds," *Proceedings IEEE INFOCOM*, 2012.

[21] R. P. a. Y. D. D. M. V. D. Peng Zhang, "Efficient load balancing algorithm in cloud environment," *International Journal Of Computer Science And Applications*, 2013.

[22] S. U. K. C.-Z. X. H. N. Tziritas, "An optimal fully distributed algorithm to minimize the resource consumption of cloud applications," *IEEE xplore Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference*, 2012.

[23] H. T. Y. A. T. S. a. H. I. Syunji Yazaki, "An efficient all-to-all communication algorithm for mesh/torus networks," *10th IEEE International Symposium on Parallel and Distributed Processing with Applications, Tokyo, Japan 1828585 Information Technology Center*, 2012.

[24] B. B. B. Almohammad, "Resource placements in 2d tori," *Oregon State University USA*, 2013.

[25] R. P. a. Y. D. M. Peng Zhang, "Interlacing bypass rings to torus networks for more efficient networks,," *IEEE*, 2011.

[26] K. I. a. H. A. Cesur Baransel, "A new parallel matrix multiplication algorithm forwormhole-routed all-port 2d/3d torus networks," *springer*, 2013.

[27] . B. B. H. Z. L. C. Avin, "Self-adjusting grid networks to minimize expected path length," *Springer international publishing*, 2013.

[28] C. W.-N. Glencora Borradaile, Piotr Sankowski, "Min st-cut oracle for planar graphs with near-linear preprocessing time," *Journal ACM Transactions on Algorithms (TALG)*, january 2015.

[29] A. V. G. R. E. Tarjan, "Efficient maximum flow algorithms," *Communications of the ACM*, August 2014.