IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-5, May 2018 E-ISSN: 2347-2693

Dynamic Load Balancing for Computational Grids using Binary Heaps (DLBCGBH – H / D)

A. Kumar[1*], H.Pathak[2]

[1*]Department of Computer Science, Gurukul Kangri Vishwavidyalaya, Haridwar (UK), India
[2]Department of Computer Science, Gurukul Kangri Vishwavidyalaya, Haridwar (UK), India

**Corresponding Author: anurom_2001@rediffmail.com*

Available online at: www.ijcseonline.org

Accepted: 21/May/2018, Published: 21/May/20182018

Abstract- Grid Computing is a variant of distributed system wherein the small scale computational units are aggregated to develop a large computational machine to support complex computational problems. It poses a number of challenges for dynamic load balancing due to a large number of heterogeneous resources and the size of data to be moved among them thereby causing a number of issues to handle effectively. Further, load balancing problem in heterogeneous distributed computer systems is a NP-Hard problem. Therefore, researchers are constantly devising innovative approaches to optimize the load balancing in grid environment. In this paper, two algorithms for dynamic load balancing in computational grid viz. DLBCGBH - H & DLBCGBH – D are being described. These algorithms have already been implemented by the authors using GridSim 4.0 along with the comparison of the performance with the Built-in Space Shared utility of GridSim 4.0 for various performance metrics viz. Average Consumed Time, Average Waiting Time, Average Processing Cost and Number of Tasks Migrated.

Keywords-Grid Computing, Dynamic Load Balancing, Space Shared, Hierarchical Grid, Distributed Grid, Binary Heaps

1. INTRODUCTION

A Computational Grid is a combination of hardware and software infrastructure to provide a dependable, consistent, pervasive and inexpensive access to high end computational capabilities [1]. It is a type of parallel and distributed system that enables the selection, distribution and aggregation of resources dynamically at runtime depending on their availability, capability and performance. It focuses on large scale resource sharing and distributed system integration for effective utilization and high performance [2].

Topologically, a computational grid comprises of a set of clusters. Each cluster owns a set of worker nodes (storage and computing elements) that belong to a local domain i.e. a LAN (Local Area Network). The worker nodes generally form a heterogeneous environment due to varying processor characteristics. Every cluster is further connected to the global network or WAN (Wide Area Network) [3].

The grid computing concept is a special variant of distributed computing wherein different computers within a network share one or more resources. In this computational environment every resource is shared to turn the pool into a powerful supercomputer. As we know it is a classical concept of powerful computation but it is not yet perfect. Computer scientists and engineers are still working for improving this technology over different phases of gird establishment, networking and other concepts [4].

Grid computing systems works on principle of pooled resources wherein the resources indicate the CPU (central processing units), memory units, and storage. Basically a single computer runs with the limited resources and having the limitations for execution and storage. Additionally to scale their performance need by incorporating additional hardware for better performance poses challenges of increased cost. On the other hand, the grid computing enables a machine to join a computational network to scale its performance for both the computational ability and storage. [5].

The load balancing is a concept to normalize the load on every resource in an infrastructure so as to enable the heavily loaded resources to perform better during higher workload scenarios. In this context, a load balancing technique helps to distribute the load across different computational units, ensures the reliability and availability of resources, and provides the flexibility to add or remove resources into or from the infrastructure [6].

The further organization of this paper is as follows: Section 2 presents a brief literature review on the related work. Section 3 presents the basics about load balancing in hierarchical and distributed grid environments along with an introduction to space shared scheduling. Section 4 presents system models for hierarchical and distributed grid environments along with the major characteristics of the models. Section 5 describes the two proposed algorithms viz. DLBCGBH – H and DLBCGBH – D. Section 6 concludes the paper along with future directions for research.

2. RELATED WORK

Due to the distribution of a large number of resources in a Grid environment and the size of the data to be migrated among them, the traditional load balancing paradigms proposed for distributed systems do not provide promising results for Grid System. A number of factors that pose challenges for load balancing in Grid Systems include heterogeneity, autonomy, application diversity, dynamicity, adaptability, scalability, resource non-dedication, resource selection and computation – data separation [7].

Due to heterogeneity of computing nodes, jobs encounter different execution times on different processors. Therefore, research should address scheduling in heterogeneous environment. The load balancing problem is to compute the assigned task with the smallest possible makespan. The load balancing problem is a minimization problem, to minimize the makespan of n tasks on m computing nodes and has been proved to be a NP-hard problem. Approximation algorithms are used to handle NP-hard optimization problems as they provide a near optimal solution in polynomial time. [8 - 11].

A simple load balancing approximation algorithm for Heterogeneous Distributed Computer System (HDCS) based on greedy paradigm is proposed in [12] wherein each task is assigned one by one to the computing nodes by selecting the node with minimum load. Selecting the minimum load from the m nodes can be possible in O (1) time with the use of a binary min-heap. A min-heap with m nodes can be used maintain the current load of m computing nodes in HDCS. The heap can be updated in O (log m) time for each task. As n number of task to be assigned the running time of the proposed algorithm is O (n log m).

A hierarchical model for computational grids is proposed in [13] wherein the grid manager maintaining the global load information is vulnerable to become bottleneck. A distributed model was proposed in [14] by mapping a grid into a forest based model wherein the local load balancing is preferred over the global load balancing.

Load Balancing on Arrival (LBA) [15] efficiently minimizes the response time for small-scale grids wherein system parameters viz. job arrival rate, CPU processing rate, load on each processor etc. and expected finish time on buddy processors are estimated on arrival of a job to immediately migrate the same. This algorithm also considers job transfer cost, resource heterogeneity and network heterogeneity while making migration decision.

A fully decentralized two-level load balancing policy for computationally intensive tasks on a heterogeneous multi-cluster grid environment to resolve the single point of failure problem is proposed in [16] wherein any site manager receives two kinds of tasks namely, remote tasks arriving from its associated local grid manager, and local tasks submitted directly to the site manager by local users in its domain. This approach distributes the grid workload based on the resources occupation ratio and the communication cost to minimize the mean task response time.

A load balancing policy to distribute the system workload based on the processing elements capacity to minimize the overall job mean response time and maximize the system utilization and throughput at the steady state is proposed in [17] for heterogeneous grid environments.

Recent trends in load balancing and job migration research in the domain of grid computing as discussed in an extensive survey reveals that recently the researchers are shifting towards hierarchical approaches for load balancing in grid environment [18].

An extensive survey of the existing load balancing techniques applicable for various systems depending upon the needs of the computational Grid, the type of environment, resources, virtual organizations and job profile in grids is presented in [19].

In view of the aforesaid facts, load balancing in grid environment seems to be a promising research area. In this paper, two algorithms for effective load balancing using binary heaps are proposed. In view of the heterogeneity involved in the grid environment, Processing Time has been chosen as the metric for workload estimation which is calculated as the ratio of workload on a node to its processing speed. For efficiency purposes the Binary Heaps are used to maintain the workload information at the manager level as it enables to locate the source and destinations during a migration decision in O(1) time and can be reorganized in $O(\log_2 n)$ time thereby ensuring efficient solution.

3. BACKGROUND DISCUSSION

In this section, the load balancing under Hierarchical and Distributed grid environments are described followed by an overview to the load balancing in space shared environment, as the performance of the two proposed algorithms,

described in this paper, has already been compared against the built-in space shared utility of GridSim 4.0 [20].

3.1 Load Balancing for Hierarchical Grid Environment

In comparison to the distributed grid, hierarchical grid has a different structure and requires a different approach for balancing the load effectively. The hierarchical load balancing concept uses the tree data structure to make decision regarding the placement of tasks on Virtual Machine. In order to utilize the resources efficiently and to satisfy the QoS requirements of the users, several hierarchical load balancing algorithms have been proposed by researchers for various applications. In Hierarchical grid environment, the nodes at different level, denoted by a tree data structure, coordinate with the nodes at a level below the hierarchy to make decisions. In hierarchical approach, the scheduling and load balancing is performed at various levels. Each level uses a scheduling algorithm to assign work to the next lower level. Every node in the tree is balanced under the supervision of its parent node. The hierarchical grid infrastructure is shown in Figure – 3.1.

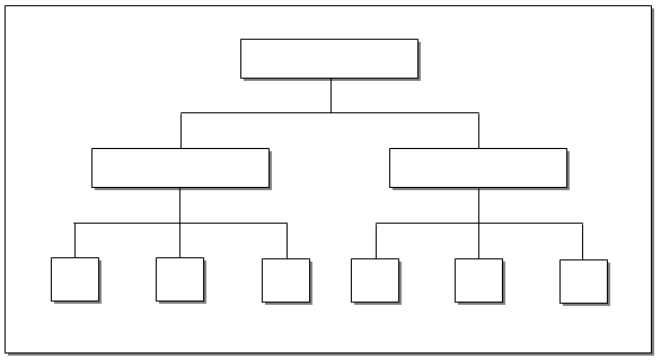

Figure – 3.1: Hierarchical Grid infrastructure

3.2 Load Balancing for Distributed Grid Environment

The load balancing in distributed grid applies to load balancing on each service to connect to the attached resources. The traditional approach for load balancing proposes to put a load balancer in front of available resources and the load balancer takes the responsibility of distributing the load across resources. On the other hand, in distributed load balancing, there are no central load balancers present. Each client that requires some service uses that service via local coordinator. Local coordinator is always up-to-date with existing services i.e. whenever a new service is being provisioned it is accordingly updated. This local coordinator takes care of the load-balancing, so it is a client-side load-balancing. Whenever a client makes a request, based on the load-balancing strategy, the local coordinator distributes the request to the attached resources. In this approach the cluster managers perform the load balancing with a direct interaction in their peer group and thereby avoiding the use of wide area network. The distributed grid infrastructure is shown in Figure – 3.2.

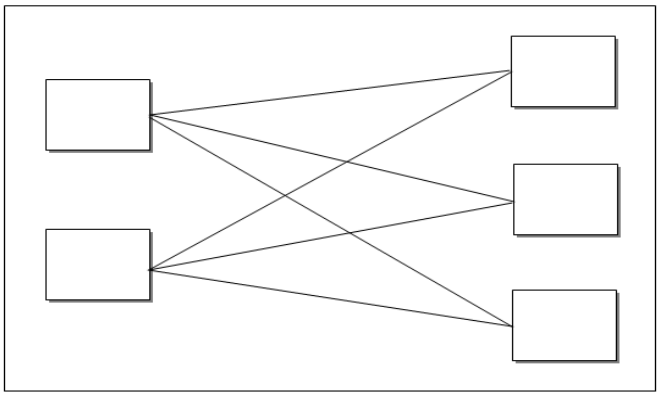

Figure – 3.2: Distributed Grid infrastructure

3.3 Load Balancing for Space Shared Environment

Due to the presence of multiple processors in a parallel and distributed computing space sharing and time sharing can be used for scheduling. In Space Sharing, a job is allocated a distinct subset of processors; that is, no processor is concurrently assigned to more than one job. Space Sharing may be static or dynamic. In Static Space Sharing, the subset is fixed for the lifetime of the job. However, it can change in size and in the processors it contains in Dynamic Space Sharing.

Scheduling multiple jobs at the same time across multiple CPUs is called Space Sharing. The simplest space sharing algorithm works like this: Assume that an entire group of related threads is created at once. At the time it is created, the scheduler checks to see if there are as many free CPUs as there are threads. If there are, each thread is given its own dedicated (i.e., non multi programmed) CPU and they all start. If there are not enough CPUs, none of the threads are started until enough CPUs are available. Each thread holds onto its CPU until it terminates, at which time the CPU is put back into the pool of available CPUs. If a thread blocks on I/O, it continues to hold the CPU, which is simply idle until the thread wakes up. When the next batch of threads appears, the same algorithm is applied.

4. SYSTEM MODEL

This section describes the system models for load balancing in Hierarchical and Distributed computational grid environment as follows:

4.1 Hierarchical Computational Grid

The system model for hierarchical computational grid is based on mapping a grid into a forest based model [20] as shown in Figure-3.1.

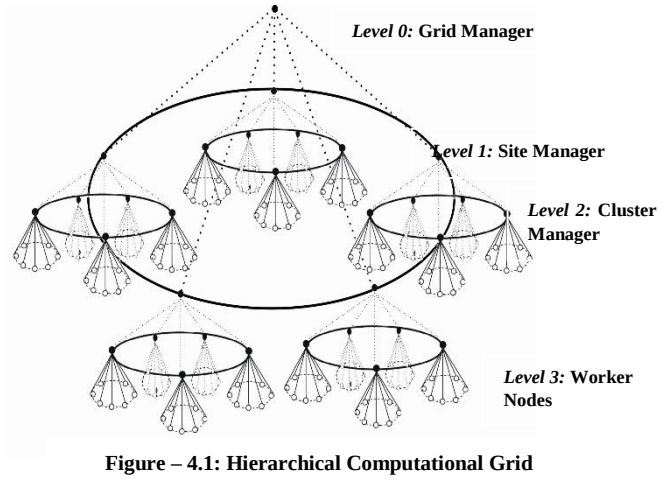

Figure – 4.1: Hierarchical Computational Grid

The Grid Manager at Level 0 centralizes the Global load information on the grid. The Site Manager at Level 1 manages a specific physical site of the grid. The Cluster Managers at Level 2 manages a physical cluster of the grid. The Worker Nodes linked to their respective clusters are at Level 3. Responsibilities of the Grid Managers, Site Managers, and cluster Managers & Worker Nodes are as follows:

Grid Manager at Level 0:

- It maintains the workload information related to each one of its site managers.
- It estimates the workload of the associated grid.
- It takes decision of performing inter-site load balancing and acts accordingly.
- It communicates the load balancing decisions to the site managers for further execution at cluster level.

Site Manager at Level 1:

- It maintains the workload information related to each one of its cluster managers.
- It estimates the workload of the associated site and periodically sends the related information to the grid manager.
- It takes decision of performing inter-cluster load balancing & acts accordingly.
- It executes the load balancing decisions communicated by the Grid Manager.
- It communicates the load balancing decisions to the cluster managers for further execution.

Cluster Manager at Level 2:

- It maintains the workload information related to each one of its worker nodes.
- It estimates the workload of the associated cluster and periodically sends the related information to the site manager.
- It takes decision of performing intra-cluster load balancing & performs the same.
- It communicates the load balancing decisions to the worker nodes for further execution.
- It executes the load balancing decisions communicated by the concerned Site Manager.

Worker Node at Level 3:

- It is responsible for execution of the incoming jobs.
- It maintains the workload related information and periodically sends the same to Cluster Managers.
- It executes the load balancing decisions taken by the concerned Cluster Manager.

4.2 Distributed Computational Grid

The system model for distributed computational grid as shown in Figure –3.2 resembles the lowest two levels of Hierarchical Model explained in the previous section with the difference that the Cluster Managers at Level 0 are connected to each other through Mesh Topology and interact among themselves. The Worker Nodes linked to their respective clusters are at Level 1. Responsibilities of the Cluster Managers and the Worker Nodes are as follows:

Cluster Manager at Level 0:

- It maintains the workload information related to each one of its worker nodes.
- It estimates the workload of the associated cluster and periodically sends the related information to the other cluster managers.
- It takes decision of performing intra-cluster load balancing & performs the same.
- It communicates the load balancing decisions to the worker nodes for further execution.
- It decides to perform inter-cluster Load Balancing whenever required and performs the same with proper communication with other Cluster Managers.

Worker Node at Level 1:

- It is responsible for execution of the incoming jobs.
- It maintains the workload related information and periodically sends the same to Cluster Managers.
- It executes the load balancing decisions taken by the concerned Cluster Manager.

The Cluster Managers maintain & supervise the workload of worker nodes and communicates directly to other Cluster Managers for implementing the Inter Cluster Load Balancing, if required.

Level 0: **Cluster Manager**

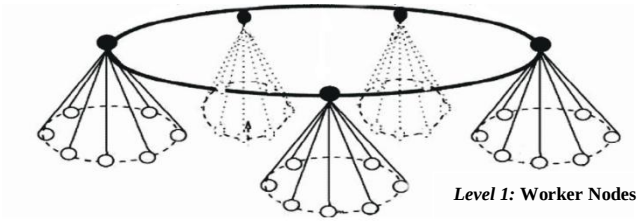

Figure – 4.2: Distributed Computational Grid

Major characteristics of our system model which are of immense importance for the proposed load balancing approaches are as follows:

1. The computing elements have different characteristics viz. speed, period for sending load information etc. due to heterogeneity of worker nodes.

2. Tasks are independent and non-preempt able.

3. Inter-Cluster communication costs are different due to the heterogeneity of WANs

4. Intra-Cluster communication costs are the same due to the similar bandwidth provided by LAN to all its worker nodes.

5. Cluster Managers can either be dedicated or have a dual role i.e. performing task and managing cluster.

5. PROPOSED LOAD BALANCING ALGORITHMS

On the basis of the System Models for hierarchical and distributed grids described in the previous section, the proposed algorithms for load balancing in hierarchical and distributed grids are described in this section wherein the state of imbalance for an element is determined by the standard deviation of its processing time (i.e. the metric for workload estimation as described in Section 2) over the processing time of the respective group (Cluster / Site / Grid). The description of two threshold values viz. Balance Threshold & Saturation Threshold, used in these algorithms, is as follows:

(i) Balance Threshold: In view of the aforesaid description, small values of standard deviation represent the balance state of elements and to avoid unnecessary task migration in such state, a fraction viz. Balance Threshold is used as a simulation parameter whose value ranges from 0 to 1.

(ii) Saturation Threshold: As the balanced state of an element does not avoid the possibility of it being saturated, so another threshold viz. Saturation Threshold is needed to check whether the element has reached to the state of saturation. In case the element has reached to saturation state, an effort of intra-group (Cluster / Site / Grid) balancing may result into crashing of the element and hence should be avoided. It is a simulation parameter whose value ranges from 0 to 1. It is calculated as the ratio of the present workload and the capacity of an element / group (Cluster / Site / Grid).

[A] Dynamic Load Balancing for Hierarchical Computational Grids using Binary Heaps (DLBCGBH–H)

As shown in the proposed model for Hierarchical Grids in the previous section, the load balancing is performed at three levels as per the algorithms described below:

1. INTRA-CLUSTER-LOAD-BALANCING

Input: Workload Information about each worker node belonging to the Cluster, Various Threshold Values

Output: Load Balancing Fail or Success

Process:

1. All the nodes send their workload information to the corresponding Cluster Managers.
2. Cluster Managers compute the Processing Time of Nodes & the Cluster, Cluster Capacity, and the Standard Deviation of the nodes' processing time over processing time of the respective cluster.
3. Cluster Managers send the workload information to the respective Site Managers.
4. In view of the values computed in Step 2, Balance Threshold & Saturation Threshold, the Cluster Manager decides the State of Cluster as Balanced, Saturated, or Imbalanced.
5. For Imbalanced & Saturated Clusters, the nodes are classified as Overloaded, Under loaded, or Balanced
6. Cluster Managers create the Max-Heap of Overloaded nodes and Min-Heap of under loaded Nodes on Load Index (i.e. Processing Time) for the nodes.
7. If the Cluster is Not-Saturated OR total load causing overloading of the Overloaded nodes of the Cluster IS LESS THAN the total load causing underloading of the Underloaded nodes of the Cluster

   Then

   INTRA-CLUSTR-TASK-TRANSFER ( )

   Else

   INTRA-SITE-LOAD-BALANCING ( )

1.1 INTRA-CLUSTER-TASK-TRANSFER

Input: Max-Heap of Overloaded nodes, Min-Heap of underloaded nodes

Output: Intra-Cluster Task Migration

Process:

1. While the existence of Max-Heap of Overloaded Nodes & Min-Heap of Under-loaded Nodes at the Cluster

Manager's level, the most suitable task from the root of Max-Heap is transferred to the root of Min-Heap
2. The loads of the two nodes affected in Step 1 are updated and the heaps at the respective Cluster Manager are reorganized.

2. INTRA-SITE-LOAD-BALANCING

Input: Workload Information about each Cluster belonging to the site, Various Threshold Values

Output: Load Balancing Fail or Success

Process:

1. All the Clusters send their workload information to the corresponding Site Managers.
2. Site Managers compute the Processing Time of Clusters & the Site, Site Capacity, and the Standard Deviation of the Clusters' processing time over processing time of the respective Site.
3. Site Managers also maintain a matrix of Communication Cost between each pair of Clusters.
4. Site Managers send the workload information to the Grid Manager.
5. In view of the values computed in Step 2, Balance Threshold & Saturation Threshold, the Site Manager decides the State of Site as Balanced, Saturated, or Imbalanced.
6. For Imbalanced & Saturated Sites, the Clusters are classified as Overloaded, Under-loaded, or Balanced
7. Site Managers create the Max-Heap of Overloaded Clusters and Min-Heap of Under-loaded Clusters on Load Index (i.e. Processing Time) for the nodes.
8. If the Site is Not-Saturated OR total load causing overloading of the Overloaded Clusters of the Site IS LESS THAN the total load causing under loading of the Under loaded Clusters of the Site

   Then

   INTER-CLUSTER-TASK-TRANSFER ( )

   Else

   INTRA-GRID-LOAD-BALANCING ( )

2.1 INTER-CLUSTER-TASK-TRANSFER

Input: Max-Heap of Overloaded nodes, Min-Heap of underloaded nodes, and Matrix of Communication Cost between Clusters

Output: Inter-Cluster Task Migration

Process:

1. While the existence of Max-Heap of Overloaded Nodes at Cluster Manager, it locates a Cluster Manager with minimum inter-cluster Communication Cost having Min-Heap of Under-loaded Nodes, the most suitable task (i.e. the one with the largest remaining processing time on the current node) from the root of Max-Heap is transferred to the root of Min-Heap subject to the following condition for avoiding unnecessary task migration: (Latency of Task 'x' in Source Node + Cost of Transfer) < Latency of Task 'x' in Destination Node
2. The loads of the two nodes affected in Step 1 are updated and the two heaps at the respective Cluster Managers are reorganized.

3. INTRA-GRID-LOAD-BALANCING

This algorithm performs global load balancing among all sites of the grid and is used in the extreme case when all the site managers fail to locally balance their load. It is same as Intra-Site Load Balancing with the following two differences:

1. In the absence of any higher level, grid manager need not send workload information.
2. There is no need to test that whether load balancing has resulted after the execution of this algorithm as there is no other alternative.

[B] Dynamic Load Balancing for Distributed Computational Grids using Binary Heaps (DLBCGBH – D)

In the proposed load balancing algorithm for distributed grids, the cluster managers perform the load balancing with a direct interaction in their peer group and thereby avoiding the use of wide area network. The algorithm is described as follows:

1. INTRA-CLUSTER-LOAD-BALANCING ( )

Input: Workload information from worker nodes

Output: Task Migration Decision

Process:

1. All the nodes send their workload information to the corresponding Cluster Managers.
2. Cluster Managers compute the Processing Time of Nodes & the Cluster, Cluster Capacity, and the Standard Deviation of the nodes' processing time over processing time of the respective cluster.
3. Cluster Managers also maintain a matrix of Communication Cost between each pair of Clusters.
4. Cluster Managers send the workload information to the other Cluster Managers.
5. In view of the values computed in Step 2, Balance Threshold & Saturation Threshold, the Cluster Manager decides the State of Cluster as Balanced, Saturated, or Imbalanced.
6. For Imbalanced & Saturated Clusters, the nodes are classified as Overloaded, Under-loaded, or Balanced

7. Cluster Managers create the Max-Heap of Overloaded nodes and Min-Heap of Under-loaded Nodes on Load Index (i.e. Processing Time) for the nodes.
8. If the Cluster is Not-Saturated OR total load causing overloading of the Overloaded nodes of the Cluster IS LESS THAN the total load causing under loading of the Under loaded nodes of the Cluster

    Then

        INTRA-CLUSTER TASK TRANSFER ( )

    Else

        INTER-CLUSTER TASK TRANSFER ( )

1.1 INTRA-CLUSTER-TASK-TRANSFER

Input: Max-Heap of Overloaded nodes, Min-Heap of Under-loaded nodes

Output: Intra-Cluster Task Migration

Process:

1. While the existence of Max-Heap of Overloaded Nodes & Min-Heap of under-loaded Nodes at the Cluster Manager's level, the most suitable task from the root of Max-Heap is transferred to the root of Min-Heap
2. The loads of the two nodes affected in Step 1 are updated and the heaps at the respective Cluster Manager are reorganized.

1.2 INTER-CLUSTER TASK TRANSFER

Input: Max-Heap of Overloaded nodes, Min-Heap of Underloaded nodes, and Matrix of Communication Cost between Clusters

Output: Inter-Cluster Task Migration

Process:

1. While the existence of Max-Heap of Overloaded Nodes at Cluster Manager, it locates a Cluster Manager with minimum inter-cluster Communication Cost having Min-Heap of Underloaded Nodes, the most suitable task (i.e. the one with the largest remaining processing time on the current node) from the root of Max-Heap is transferred to the root of Min-Heap subject to the following condition for avoiding unnecessary task migration: (Latency of Task 'x' in Source Node + Cost of Transfer) < Latency of Task 'x' in Destination Node
2. The loads of the two nodes affected in Step 1 are updated and the two heaps at the respective Cluster Managers are reorganized.

6. CONCLUSION AND FUTURE SCOPE

In this paper two algorithms are described for solving the problem of load balancing in grid environment. Firstly, in the Hierarchical Algorithm, multiple authorities are required for managing and balancing the load in effective manner. Secondly, in the Distributed Algorithm, the local authority is responsible for balancing the load effectively.

Both of these algorithms have already been implemented using GridSim 4.0 for a comparison of simulation results with built-in space shared utility of GridSim 4.0 in view of the various performance metrics viz. Average Consumed Time, Average Processing Cost, Average Waiting Time, and Number of Tasks Migrated. It has been observed from the comparison of performance metrics that both of the proposed algorithms outperform the built-in space shared utility of GridSim 4.0. Further, the distributed algorithm outperforms the hierarchical one on all the four performance metrics [20].

In future, the proposed hierarchical and distributed algorithms can be enhanced for performance and fault tolerance using soft computing techniques. Improvements can also be tried by taking into account the characteristics of task viz. Resource Requirements, CPU Bound, Deadline etc. during its transfer to the best suited node in the grid environment. Further, the complexity of the proposed algorithms can mathematically be analyzed.

REFERENCES

[1] I.Foster, C.Kesselman, J.M.Nick, and S.Tuecke, "Grid services for distributed system integration", IEEE Computer, vol. 35, num. 6, pages 37-46, 2002.

[2] Tarek Helmy, Hamdi Al-Jamimi, Bahar Ahmed, Hamzah Loqman, "Fuzzy Logic – Based Scheme for Load Balancing in Grid Services", A Journal of Software Engineering and Applications, 5, pages 149-156, 2012.

[3] K.Lu, R.Subrata, and A.Y.Zomaya, "An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites", Proc. 25th IEEE Int. Performance Computing and Comm. Conf. (IPCCC '06), 2006.

[4] Gilles Fedak, "Contributions to Desktop Grid Computing", University of Lyon, 28 Mai 2015.

[5] P. K. Suri &Sunita Rani, "Resource Management in Grid Computing: A Review", Global Journal of Computer Science and Technology, Network, Web & Security, Volume 13 Issue 17 Version 1.0 Year 2013.

[6] Nada M. Al Sallami, Ali Al daoud, Sarmad A. Al Alousi, "Load Balancing with Neural Network", International Journal of Advanced Computer Science and Applications,Vol. 4, No. 10, 2013.

[7] D.K. Patel et al., "Survey of load balancing techniques for grid", Journal of Network and Computer Applications, 65, 103–119, 2016.

[8] J. Kleinberg and E. Tardos, "Algorithm Design", Pearson Education Inc., 2006.

[9] A. Zomaya and Y. Teh, "Observations on using genetic algorithms for dynamic load-balancing", Parallel and Distributed Systems, IEEETransactions on, vol. 12, no. 9, pp. 899–911, 2001.

[10] D. S. Hochbaum, "Approximation Algorithms for NP-Hard Problems", Thomson Asia Pte Ltd., 2003.

[11] M. Garey and D. Johnson, "Computing and Intractability, A Guide to the Theory of NP-Completeness", New York: W.H. Freeman and Company, 1979.

[12] Bibhudatta Sahoo, Sanjay Kumar Jena, Sudipta Mahapatra, "Load Balancing in Heterogeneous Distributed Computing Systems using Approximation Algorithm", **Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA); Athens** : 38-43.

[13] Yagoubi B., "Modele d'equilibrage de charge pour les grilles de calcul", Revue Africaine de la Recherche en Informatique et Mathematiques Appliquees: ARIMA, vol. 7, pages 1-19, 2007.

[14] Yagoubi B., Meddeber M., "Distributed Load Balancing Model for Grid Computing", ARIMA, vol. 12, pages 43-60, 2010.

[15] E. Saravanakumar and P. Gomathy,"A novel load balancing algorithm for computational grid", Int. J. of Computational Intelligence Techniques, vol. 1, no. 1, 2010.

[16] El-Zoghdy, S. F., "A hierarchical load balancing policy for grid computing environment", International Journal of Computer Network and Information Security, Volume 5, pages 1–12, 2012.

[17] El-Zoghdy S.F., "A capacity-based load balancing and job migration algorithm for heterogeneous Computational grids", International Journal of Computer Networks & Communications (IJCNC) Vol.4, No.1, pp. 113-125, 2012.

[18] Neeraj Rathore, Inderveer Chana, "Load Balancing and Job Migration Techniques in Grid: A Survey of Recent Trends", Wireless Pers Commun, Springer Science+Business Media New York 2014.

[19] Deepak Kumar Patel, Devashree Tripathy, C. R. Tripathy, "Survey of load balancing techniques for grid",Journal of Network and Computer Applications, Volume 65 Issue C, pages 103-119, April 2016.

[20] Anuj Kumar, Heman Pathak, "A Comparative Study of Grid Load Balancing", International Journal of Computer Applications (IJCA), 179 (18): 25-33, February 2018.

Author Profiles

Mr. Anuj Kumar pursued M.Sc. (Computer Science) from G.B.Pant University of Agriculture & Technology, Pantnagar and PGDM from Shri Ram Murti Smarak College of Engineering & Technology, Bareilly. He is currently working as Associate Professor, Department of Computer Applications, Shri Ram Murti Smarak College of Engineering & Technology, Bareilly. He is pursuing Ph.D. in Computer Science from Department of Computer Science, Faculty of Technology, Gurukul KangriVishwavidyalaya, Haridwar, India. His area of interest in teaching and research include Algorithms, Soft Computing and Distributed Systems.

Dr. Heman Pathak pursued M.Sc. (Computer Science) from BHU, Varanasi and Ph.D. in Computer Science from Gurukul Kangri Vishwavidyalaya, Haridwar. She is currently working as Associate Professor, Department of Computer Science, Faculty of Technology, Kanya Gurukul Campus, Dehradun (Second Campus of Gurukul KangriVishwavidyalaya, Haridwar, India). During her teaching experience of 19 years and research experience of 10 years, she has published more than 30 research papers in reputed International & National Journals and has attended 12 International and 08 National conferences. Her areas of interest in research include Distributed Systems.

