ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

Adjacent Selection Method for Load Balancing in Distributed Network by Artificial Intelligence

Riyazuddin Khan[1], Mohd Haroon[2], Shahid Husain[3], Afsaruddine[4]

PG Student [M.Tech], Dept. of CSE, Integral University, Lucknow, Utter Pradesh, India[1]

Associate Professor, of CSE, Integral University, Lucknow, Utter Pradesh, India[2]

Assistant Professor, Dept. of CSE, Integral University, Lucknow, Utter Pradesh, India[3,4]

ABSTRACT: Distributed computing and grid computing used to execute parallel and distributed applications, which require important quantity of computing resources, moreover in the type of computational processing resources or data storage. In a cluster, a number of heterogeneous nodes contain, by load balancing approach the response time for parallel or distributed systems are minimized. The workload for all systems in the cluster cannot be uniformly distributed, but has to be in use as a significant parameter for the load balancing strategy. A centralized load balancer would necessitate a global load balancer that will be overwhelmed with communication messages in a large clustered. This overhead can be addressed by a decentralized come up, to do load balancing where decisions are formulated by the nodes performing scheduling algorithm communicating with all member nodes of the cluster. Even if this will lead to an overall increase of communications, it will remove the bottleneck of including a centralized load balancing node. In an overloaded node (sender) initiate decentralized load balancing approach, each sender will need to have a prearranged list of adjacent nodes to relieve of their jobs. In this paper, we propose an artificial intelligence based adjacent collection algorithm that selects an adjacent for job distribution every time overload occurs. Simulations were conducted by means of the OMNeT & C simulator and the generally proposed performance was evaluated to other obtainable adjacent selection method. From our simulation results, we found that our proposed method to demonstrate the overall development for load balancing by reducing the average response time.

KEYWORDS: Decentralized distributed system, adjacent selection, dynamic load balancing, and task re-distribution.

I.INTRODUCTION

A distributed system is a computing model which comprises of a set of autonomous computing nodes, contributing their processing resources to attain a common goal. It can be used for sharing program or jobs crossways the network or processing information. On the other hand, grid computing is a category for distributed systems which combine computing resources from multiple organizational domains to form a big supercomputer. In the matter of this, it generally demands a lot of computing time and memory storage, for the benefits of the computationally intensive applications. For example, ERIS, a scientific application, the realistic computer simulation of the Milky Way took 9 months of computing time by means of a collection of supercomputers in regulate to see the spiral of our galaxy .In distributed, there is a computing node could be in the state of idle for higher probability, or lightly loaded or even heavily loaded with jobs. Jobs that come out to be in the heavily loaded computing nodes turn out to take a longer time to be executed since a job have to remain for the previous job to complete. In other words, the waiting time for a job to be executed amplify as the job queue grows.

To distribute jobs equally, the load balancing method is engaged by distributing jobs to lightly loaded nodes from heavily loaded nodes. Where load balancing is employed by using a number of dissimilar approaches and it has been extensively studied. Thomas and Jon. studied on the classification of load balancing and additional grouped it into several subclasses, namely static and dynamic load balancing. Static load balancing is anywhere every essential information to formulate the job distribution decision was unspoken to be recognized at the compilation time of a job. Dynamic load balancing, in dissimilarity, endeavour to make decisions during the run time of a job. Hence, every node

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

has to frequently swap overload information with each other in order to obtain the mainly up to date load information to put together the decision. Load information exchange can be completed via periodically or on demand of a node and it has to be taken cautiously in order to prevent any further overwhelming of communication messages in the system. Besides that, the load balancing algorithm can be implemented in a centralized or decentralized draw near. A centralized approach is where the responsibility for the job scheduling resides in a single node. Although this method is easier to be implemented, this approach is not scalable due to the communication bottlenecks at the centralized node as the size of the system increases. Decentralized approach, on the other hand, has the limitation of overall high communication expenditure. However decentralized come near eliminates the bottleneck of having a centralized node[1].

In this paper, we optimized the assortment of adjacent on each node individually instead of having the knowledge of all nodes in the system. In this case, the load balancer (load balancer) does not have to swap overload information with all other nodes. In addition, communication messages can be additional condensed by triggering the load information exchange when there is only an imbalance of workload within the adjacent. The main contributions of this paper are as follows:
· Enhanced adjacent selection for job distribution.
. Minimizing the standard deviation and optimizes load balancing by of average response time.
In the next section, we sum up the work has been completed by previous researchers. Then in section 3, we explain the model, simulation setup and implementation, followed by the results and discussions in section 4. In conclusion, we conclude our conclusion in section 5.

II. RELATED WORK

Yongsheng Hao et al. proposed a dynamic, distributed load balancing method for a grid, which provides deadline manager of tasks. They planned a new calculation method and confidential resource into three types: overloaded, in general load and under loaded. Unassigned grid lets list used to lay up the new arriving grid, let and the incomplete grid lets coming from the resource when the carrying out fails. Finally proposed a novel based load balancing method such that resources and grid broker participate in load balancing.

U. Karthick Kumar proposed a dynamic load balancing algorithm for light scheduling. Issue using signify waiting time he addressed the fairness issue. Tasks are rescheduling by means of waiting time and scheduled by means of fair completion time of each task to get load balance[2].

Stylianos Zikos and Helen D. Karatza suggest a load balancing and site allocation scheduling of volatile jobs is two level heterogeneous grid architecture (GS, LS). Three scheduling strategy (Basic hybrid, PAD, FZF) at grid level, which makes examine the site load information. For allocating jobs to all PEs Shortest queue plan has used to at the resource level. These policies construct for dynamic site load information to share the load while communication overhead owing to information exchange is taken into account.
Malarvizhi Nandagopal et al. [5] proposed a sender initiated decentralized dynamic load balancing method for multi-cluster computational grid (SI-DDLB).

Yajun Li et al. Addressed the problem of load balancing by hybrid come near (both average based and instantaneous measures based) for sequential tasks in grid computing. A cautiously designed genetic algorithm was chosen as a legislature of both classes to work together, A first come first served to attain load balancing. The sliding window technique was used for activating GA into action.
Malarvizhi Nandhagopal and Rhymend V. Uthariaraj addressed the problem of load balancing and expansion in a grid resource where computational resources are disconnected in a different organizational domain. It addresses the problem of load balancing with min cost policy's main load and while scheduling jobs to multi cluster. It considers both network load and communication cost for scheduling jobs to resources in different clusters. Three step strategies are used to determine a resource for an arriving job[3].

D. Grosu et al. planned a non cooperative load balancing game for distributed systems, but did not consider the communication delay in a grid.

Keqin Li planned an optimal load balancing in a non dedicated cluster with heterogeneous servers. The optimization problem is solved for three queuing disciplines, namely dedicated applications with no priorities, prioritized dedicated applications without preemption, and prioritized dedicated applications with preemption.
The game-theoretic approach proposed by Zomaya et al. [11] considers only individual response time as the objective and does not consider average response time.

When compared with the obtainable work, the main characteristics of the planned strategy can be summarized as follows, It privileges a decentralized load balancing. To decrease the overhead implicated in site state information exchange between resources is done through mutual information feedback[4]. Jobs are computed exhaustive. Jobs are non preemaptable which means that their execution on a resource cannot be balanced under completion. Jobs are autonomous which means that there is no communication between them.

III. SYSTEM MODEL

In this paper, the distributed system model proposed is based on distributed clusters and the communications delays are unspecified to be minimal or negligible. Fig 1 illustrates the reveal model of each cluster where the resource refers to computing nodes and user task is the job submitted by the user at any cluster. Each cluster comprises of interconnected Computing nodes and a load balancer. The load balancer is responsible for managing its own resources, scheduling user's jobs and balancing workload. As for the job scheduling, load balancer examines all arrival jobs from users and decides whether to offload the job or not, based on both the load information on its adjacent and itself. Hence, this load balancing algorithm is initiated by the heavily loaded cluster. As regards to load information exchange, it is based on

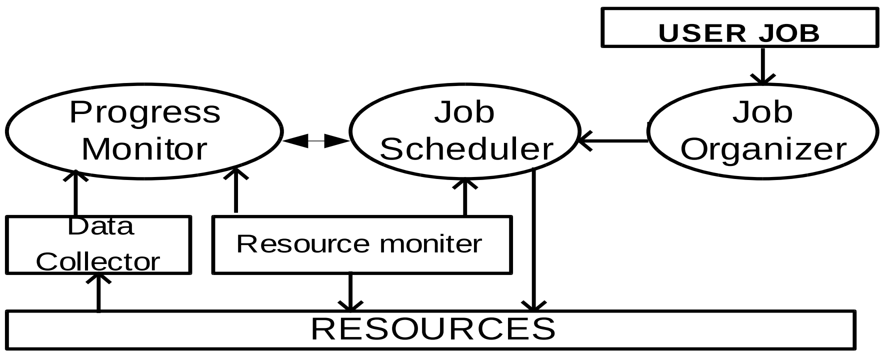

Fig. Distributed Cluster model

[5], [6], [7].

Initially, a load balancer does not have any load information respecting its adjacent; therefore, this information was assumed to be the same at the beginning.
Over a prolonged period of time, load balancer learns about its adjacent by exchanging load information. The trigger of exchanging load information is when a heavily loaded cluster, the sender, needs to transfer jobs to a remote cluster which the load is light or upon job completion[5]. The following are the notations
· J is denoted as set of jobs.
· C signifies cluster.
· C Adj, the set of adjacent.
· W is a cluster in which concepts are implemented.
·β is the load level.
· ∀C (c comprises of the following properties:
–Pwr the processing power.
–Delay the communication delay.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

–Load, current load index.
–Time the last updated time for load index.

Load balancer ID	6	8	10	12	15
Processing Power (Job size/Sec)	10	20	30	40	50

Table 1 load balancer verses processing power

IV. LOAD INFORMATION SHARING

Load information sharing is an essential process in dynamic load balancing where a load balancer determines the job distribution based on such information. The load information keeps on exchanging as long as there is a job distribution process is going on or upon a job completion reply. In exchange load information, the sender sends the job and associated the load information of itself and α percentage of its adjacent to the receiver, which is also employed by [5], [6], [7]. The load information sent is mainly to update the receiver regarding load of other clusters in the system. However, with reference to the piggyback load information, it is possible that there were no adjacent to be included if the value is small enough or the sender is the highest processing node which contains no adjacent at all. In order to address this, a sender, as long as is not the highest processing node, must include at least one adjacent to send to. Upon the receiver receives the job and the adjacent load information, the receiver updates its adjacent list based on the time As regards to this, the receiver compares the time marked in the load information from the sender and its own, hence, the update can be only successful
If the time value is greater than the one it has.
Adjacent selection of job distribution
Algorithm 1 Artificial intelligence Adjacent Selection

```
Input: arrival job, jε J
Output: adjacent cluster, c ε C
1: if the arrival of job is from w then
2∃ min ε C neighbor, where min possesses the least load.min
3: if w.load () –min.load ()<β then
4: return w
5: else
6: return min
7: end if
8: else if the arrival of j is from cεC then
9: N bar ← set of adjacencies associated from c
//Find the adjacent node on the basis of A* algorithms
10: if N bor≠ 0 then
11: (min εbor where min possesses the least load.
12: if w.load () min.load() > βthen
13: return min
14: else if w.load () -c.load () >β then
15: return c
16: end if
17: end if
18: return w
19: end if
```

Every individual load balancer possesses its own adjacent list for job distribution. Such adjacency list is created based on Malarvizhi [6, 7] model, where each load balancer selects those clusters that possess greater processing power and with the communication delay not additional than 1.5 times of the lowest one. Based on simulation results in [6, 7], it was completed that the ratio of 1.5 produces a good result[8]. Upon imbalances of workload occur, this will trigger the load balancer to choose its adjacent which possesses the minimum load to reallocate the job[12,13]. However, the highest processing node will face problem on redistributing jobs due to an empty adjacent list. In this abstract, the highest processing node always en queue arrival jobs despite if it is overloaded. Eventually the job queue built up and

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

forces each of the jobs in the queue to wait for the preceding job to be completed. In order to address this matter, load balancer with an empty adjacent list be supposed to make use of the arrival load information sent by the sender to reselect an adjacent for job processing. Algorithm 1 outlined the method we used to select an adjacent upon job arrival, namely Artificial intelligence Adjacent Selection Algorithm (AIAS).

1. Implementation and simulation setup

We conducted the simulations under various workloads using a disconnected event network simulator, In order to demonstrate the issue of workload imbalance in distributed clustered system OMNET++ [8]. Hence, only the inter cluster load balancer was simulated instead of simulating the computing nodes in a cluster. This simulation was intended to simulate the inter-cluster load balancing. The reproduction model mainly comprises of 10 overlays distributed load balancer with a network diameter of 6. Table I depicts the stipulation of load balancer where the slighter the load balancer's ID, the smaller the processing power (the processing power is calculated in conditions of job size per second) and. In our simulation, the network topology consists of 3 spinal column routers of the network, which were connected using 10Gbps fiber line to 1ms wait and the rest of the connections were using 1Mbps digital subscriber line (DSL) with 20ms wait[8,9,10].

The jobs, generate by a job computing machine enthusiastic on the cluster are at every 2 seconds. A job generator reads the job from a file which contains 10,000 jobs where each job is illustrated by its size ranging from 20 to 99. We also introduced pour jobs, Apart from generating jobs one by one which is an amount of jobs pouring into a cluster at a single rate of time in order to simulate the heavy workload. The simulation ends after all jobs are completed.
There are 4 different scenarios to simulate the heavy workload in different kinds of load balancer shown in Table 2.
Scenario 1

S.NO	LOAD BLANCER ids	Know of job
1	5	300
2	13,14	300
3	5,13,14	300

Table 2: load balancer verses no of jobs

Load balancer's Id on Which Pour Jobs Occur and the Amount of Jobs Poured per Load balancer. Is where all load balancer receives jobs at a constant rate of time, non pour jobs happen in this scenario? In scenario 2, is where the poor jobs occur in a single low processing capability load balanced at an amount of 300 jobs [5,6,8]. For performance evaluation, we deliberate and record the response time for all single job. The timer for the response time start instantly after a job generated at a cluster and stops after the job completion at the same cluster. The time for waiting time start together with the response time, but stops after a job being executed. In this paper, the performance of this proposed method, AIAS, is evaluated using the simulation setup described and compared with the method proposed by Malarvizhi . The processing power of each cluster is shown in Table I. In measuring load balancing, the difference of each job response time has to be small. The more assortments the response time of a job is the bigger difference. With regard to this, standard deviation is used. The overall average response time of 10,000 jobs for the AIAS method in scenario 1 are 36.70s and 36.94s respectively[2,3].

Generally less than 1s and with our AIAS method fashioned roughly 12% higher than ABLA method, however, this difference is insignificant. From our simulation result, we can conclude that in the normal situation the AIAS method works just the standard deviation for in cooperation methods are relatively small, generally less than 1s and with our NSA method produced approximately 12% higher than ABLA method, however, this difference is trivial. From our simulation result, we can conclude that in the normal situation the AIAS method works just as fine as the ABLA method [7].

Parameters	Values
Total number of load balancer	10000
Job size	20-99
Job arrival rate (Sec),	2
Load tolerance level	.9
Percentage of adjacent	20%

Table 3 workload in different kinds of load balancer

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

In scenario 2, pour jobs at an amount of 300 was introduced at a single rate of time in the weakest cluster which is load balancer 5. Upon such event occur in using ABLA method; load balancer 5 selects the adjacent which has the negligible load to transfer the job, load balancer 10 for an instant. Upon load balancer 10 receives the job; it will moreover recognize the job if it is not heavily loaded, or else it will repeat the same thing as what load balancer 5 did [8]. Eventually, the job reaches either load balancer 13 or 14 which possess the highest processing capability in the distributed system. This load balancer cannot additional transfer the job to other adjacent due to an empty adjacent list, hence; they would have to en queue the job[1,4,6], which eventually increase the size of the queue. With reference to this, The AIAS method, on the other hand, works slightly differently on management arrival job[11,12]. While job arrives on load balancer 14, at the same time it also receives the load information of the sender and its adjacent. We search

all the information to search for suitable adjacent which can process the job. Therefore, in circumstances 2, then by and large average response time for the AIAS method and ABLA method are 147.04s and 153.38s as a result [9,12]. And, the AIAS method has a standard deviation of 4.92s which is just about 67% less than ABLA method.

2. Effect of system heterogeneity: We carry out a series of simulations with the algorithms described above for three different heterogeneous systems, under a different system utilization parameter. We first considered only situations where the fastest Virtual organizations have up to 10 times higher relative processing power than the slowest Virtual organization, because this is true of most of the current heterogeneous distributed systems [8,9]. We present a highly heterogeneous system configuration with four different processing powers. We varied the system loading by varying the mean inter arrival time (initiation time) of the jobs, 1/λ. We can conclude that ABLA behaves poorly in a highly heterogeneous system[5,8,9]. IA gives the minimum average response time across all values of load. At light or medium system loading (10–60%), AIAS performs significantly better than ABLA [13]. For example, at system, loading of 50%, the average response time using AIAS is 36.92% less than ABLA and the difference reaches the highest point.

Average processing Power	1	1	1	2	2	2	5	5	10	10
Total job completed by AIAS	.15	.15	.15	.14	.14	.14	.38	.38	.29	.29
Total job completed by ALBA	0.12	0.12	0.12	0.17	0.17	0.17	0.39	0.39	0.33	0.33

Table 4 Processing Power among AIAS and ALBA

When the system loading becomes high, the difference between the average response time of ABLA and AIAS decreases. At high system load of 90%, AIAS yields the average response time, which is 17.14% less than ABLA. The AIAS has an average improvement factor of 29.68% over ABLA. Analysis of the results revealed the following reasons for the relative performance of each algorithm in terms of the average response time when the system loading is light or moderate for AIAS, AIAS plays a crucial role and LAP makes little influence on the average response time of the jobs [7,11]. ABLA transfers a job to an idle adjacent Virtual organization, which can be much slower in a highly heterogeneous system than a closer non adjacent Virtual organization that has simply a small amount of jobs in the queue (or that is now processing a work and has an empty queue).

5. Conclusion: In this paper, we draw round an algorithm for job distribution by choosing a neighbour upon a job arrival into a cluster. We build simulation for 10 distributed clusters with dissimilar processing capability. We simulated 10,000 jobs below different situation to simulate the overloading in clusters. The results show that our proposed method able for job processing upon load imbalance occurred and reduce the standard deviation of response time as a quantity of load balancing improved the neighbour selection. However, this algorithm still experiencing some jobs cannot be transferred due to no prior knowledge of neighbours were in a few situations. For our future works, we plan to work on these issues as well as varying the jobs arrival time

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

REFERENCES

[1]. Said Fathy El-Zoghdy. A Load balancing Policy for Heterogeneous Computational Grids Vol. 2, No. 5, 2011
[2]. S. Xian-He, W. Ming, GHS: A performance system of Grid computing, in: Proceedings of the 19th IEEE International Symposium on Parallel and Distributed Processing, 4–8 April 2003.
[3]. X. Tang and S. T. Chanson. Optimizing static job scheduling in a network of heterogeneous computers. In Proc. of the Intl. Conf. on Parallel Processing, pages 373–382, August 2000.
[4]. Mohd Kalamuddin Ahmad, Mohd Husain," Required Delay of Packet Transfer Model For Embedded Interconnection Network", International Journal of Engineering Research, Vol 2, issue 1, Jan 2013.
[5] Mohammad Haroon, Mohammad Husain," Analysis of a Dynamic Load Balancing in Multiprocessor System", International Journal of Computer Science engineering and Information Technology Research, Volume 3,March 2013.
[6] Mohammad Haroon, Mohammad Husain," Different Scheduling Policy For Dynamic Load Balancing in Distributed System", 3rd international conference TMU Moradabad.
[7] Mohammad Haroon, Mohammad Husain," Different Types of Systems Model For Dynamic Load Balancing", IJERT, Volume 2, Issue 3, 2013.
[8] Mohammad Haroon, Mohammad Husain," Different Policies For Dynamic Load Balancing", International Journal of Engineering Research And Technology, Volume 1, issue 10, 2012.
[9] Mohd Haroon Ashwani Singh, Mohd Arif, "Routing Misbehabiour In Mobile Ad Hoc Network", IJEMR, Volume 4, Issue 5, October 2014
[10] Abdul Muttalib Khan, Mohd. Haroon Khan, Dr.Shish Ahmad," Security In Cloud By Diffie Hellman Protocol",International Journal Of Engineering And Innovative Technology(IJEIT), Volume 4 , Issue 5 , November 2014.
[11] mohd haroon, mohd Husain," Interest Attentive Dynamic Load Balancing in Distributed Systems", ieeexplore.ieee.org.
[12] mohd haroon, mohd Husain, "Server Controlled Mobile Agent", International Journal of Computer Applications (0975-8887).
[13] Sanjeev Srivastava, Mohd Haroon, Anu Bajaj, "Web document information extraction using class attribute approach" , Computer and Communication Technology (ICCCT), 2013 4th International Conference , 978-1-4799-1569-9