Augmenting Hierarchical Load Balancing with Intelligence in Grid Environment

Joshua Samuel Raj[1], Hridya K. S[2] and V. Vasudevan[3]

[1]*Assistant Professor, Department of Computer Science and Engineering, Karunya University*
[2]*PG Student, Department of Computer Science and Engineering, Karunya University*
[3]*Director, Software Technologies Lab, TIFAC, Core in Network Engineering, Srivilliputhur, India*
joshuasamuelraj@gmail.com, k.hridyas@gmail.com, drvvmca@yahoo.com

Abstract

Scheduling independent tasks to homogeneous resources is an ineluctable issue to be dealt with. Load balancing of resources is a crucial matter of concern. This paper comes out with an enhancement of hierarchical load balancing algorithm. In this paper, to evaluate cluster imbalance, probability of deviation of average system load from average load of cluster is calculated and checked for confinement within a defined range of 0 to 1. The algorithm also compares the expected computing power of jobs with average computing power of clusters to allocate fittest resources to jobs. In addition to the load balancing and fittest resource allocation, the contribution of our algorithm is twofold. Our algorithm ensures that no cluster remain idle by employing random stealing and random pushing, whereby jobs are taken from other clusters and executed in cluster with empty queue such that the queue length remains within a fixed threshold. The contributions of augmented hierarchical load balancing with intelligence is that it reduces the makespan of algorithm execution together with balancing the overall system load and reduces the idle time of clusters.

Keywords: *clustering, job scheduling, load balancing, probability concepts*

1. Introduction

The sophisticated technological manipulation demands the utilization of grid systems. Grid computing necessitates dynamic resource creation, job allocation which can be computationally intense, clustering, and coordination of the processing entities. The heterogeneous and homogeneous characteristics [1] of resources add to the complexity of the scheduling problem.

For scheduling jobs [2] to dynamic resources, improved versions of traditional algorithms were formulated. Most of the algorithms that dealt with scheduling were inconsiderate of load balancing [3] issues, which is a critical issue in grid environment. Scheduling algorithms computes the fitness factor for resources and allocates jobs to most eminent resources. The drawback in this method of allocation is that fittest resources get overloaded and other resources remain unutilized. Such scheduling practices impact the overall system throughput.

Most of the scheduling and load balancing algorithm ignores the idle condition of a cluster

after the completion of jobs assigned to it. The cluster may remain idle until the scheduler identifies the idle cluster and assigns tasks to it. This can lead to wastage of processing time, which can impact the overall performance of the system.

This paper puts forth an enhanced algorithm for allocating fittest resources to jobs coupled with load balancing together with reducing the idle time of clusters. The experimental results shows improved system throughput and reduced makespan for computation.

This paper is divided into sections as follows: Section 2 explains the key concepts, section 3 describes the related works, Section 4 gives the description of enhanced algorithm and finally, Section 5 provides implementation details, Section 6 gives the conclusion and future enhancement.

2. Key Concepts

2.1 Clustering

Clustering [4] is the grouping of multiple resources to form a single large highly capable system. The resources include multiple computers, workstations, storage devices, interconnections etc. Clustering can be employed for accomplishing load balancing and system availability.

Clustering also comes up with the advantages like reliability, performance and low- cost. Clusters divide computationally intense tasks among multiple systems so as to reduce workload on a particular server, thus accomplishing load balancing. Clustering finds its applications in scientific computing, commercial servers etc.

2.2 Job Scheduling

Ungurean as in [5] defines scheduling issue as best mapping of jobs to processors so that the desired cost function is optimized. Objective of scheduling must be to reduce the total cost, i.e., the sum of computation cost and communication cost. Scheduling is a set of rules and policies to control the order of job execution in a system. Scheduling can be long term, medium or short term according to the type of task to be executed. Scheduling can be static or dynamic, online or batch mode.

2.3 Load Balancing

Belabbas et al as in [6] explains load balancing as an efficient methodology in grid system whereby tasks are distributed among multiple resources so as to improve efficiency and system performance.

Load balancing strategies include local or global, static or dynamic, distributed or centralized, cooperative or non cooperative etc. Load balancing can be intra-cluster, inter-cluster, static or dynamic.

2.4 Probability Concepts

Probability [7] distribution is the likelihood of a variable to take a given value. Probability distribution can be discrete or continuous. Discrete probability distribution calculates the probability of occurrence of countable number of events or samples.

3. Related Works

3.1 Dynamic Load Balancing Algorithm

In DLBA[8], each cluster communicates their load with each other. When tasks are allocated to the clusters, current load is calculated. If the cluster becomes overloaded the task is transferred to the neighborhood cluster whose load value is below threshold. DLBA employs decentralized load balancing methodology and task transfer will be carried out on the fly. The algorithm ensures performance of the system and avoid system imbalance.

3.2 Most Fit Task First

In MFTF[9], jobs are allocated to resources based on the fitness value. Higher the fitness value of a resource, more suitable the resource. Fitness value is calculated based on expected execution time and estimated execution time. If the estimated execution time is less than the expected execution time, the source is considered to be of high fitness value.
The fitness value is calculated using the equation:

$$\text{Fitness}(i, j) = 100000 / (1 + | W_i / S_j - E_i |) \quad (1)$$

Where W_i is the workload of job i, S_j is the CPU speed of node j, and E_i is the expected execution time of job i.

3.3 Balanced Ant Colony Optimization Algorithm

BACO[10] is derived from Ant Colony Optimization algorithm. The purpose of BACO is to reduce the job completion time together with load balancing. The algorithm calculates the weight value, called pheromone for each resource. Jobs are assigned to resources based on the pheromone value. Initially the pheromone value and pheromone indicator value will be same. The pheromone indicator shows the status of resource, size of jobs, program execution time so as to select the best suitable resource.
The pheromone indicator PI is calculated as given below:

$$PI_{ij} = [\mathbf{M_i / bandwidth_i} + \mathbf{T_i / (CPU\_Speed_i \times (1 - load_i))}]^{-1} \quad (2)$$

M_i is the size of the job j, $bandwidth_i$ is the bandwidth between a scheduler and the resource to which it is connected. The estimated execution time is calculated using the equation $M_i / bandwidth_i$. T_j is the time taken to execute job j, CPU_speed_i is the CPU speed of resource i, and $load_i$ is the current workload on the resource i.

3.4 Hierarchy Load Balancing Algorithm

In HLBA[11], Yun et al puts forth the idea of load balancing so as to reduce completion time of jobs and to increase system performance. The algorithm chooses suitable resource for jobs by considering the average computing power of each resource. The average computing power is calculated using available CPU utilization and current CPU utilization. HLBA then calculates the average load of each resource and compares it with a balance threshold. If ALC is higher than balance threshold, the cluster is considered to be overloaded. The under loaded clusters with high ACP are chosen and jobs are assigned to it.

3.5 Random Stealing

RS [12] is an efficient load balancing algorithm which steals jobs from a random cluster when a cluster is idle. It thus saves the idle time of the processing elements in the cluster. The RS algorithm makes multiple attempts to steal job from other clusters. If there are N clusters, the algorithm will make N-1/N attempts on an average to steal a job. Job stealing is done only when a cluster is idle.

3.6 Random Pushing

In RP [12], load balancing is accomplished by pushing jobs from queue to cluster processor. This is done when queue length is found to be increasing above a threshold value. The job allocation from a queue will be to a random cluster. Job will be pushed into the processor even before the processor demands for it, thus saving processor idle time.

4. Proposed Algorithm

In this section, the proposed algorithm called Augmented Hierarchical Load Balancing with Intelligence is being explained. It puts forth a method for job allocation to fittest resource coupled with load balancing. It also accomplishes reduced idle time.

4.1 Enhanced Hierarchical Load Balancing Algorithm

In AHLBI, a method for resource allocation with load balancing is being put forth. It inherits the ideas of Hierarchical Load Balancing algorithm in "[11]". When a job request comes, the scheduler initializes job parameters and find the Expected computing power, ECP for each job using the equation (3)

$$ECP_i = \sum_{k=1}^{t} CPUSpeed_k / t \quad (3)$$

Where $CPUSpeed_k$ is the MIPS requirement of each task k in a job i. t is the number of tasks in job i.

EHLBA uses HLBA in calculating the ACP, ALC and Average System Load as shown below. The algorithm calculates ACP, average computing power using equation (4)

$$ACP_i = \sum_{k=1}^{n} CPU\_speed_k * (1 - CPU_k) / n \quad (4)$$

Where CPU_speed_k is the available MIPS of resource k in cluster i and CPU_k is the current CPU utilization of resource k. Here n is the number of resources in cluster i. Three weighted value parameters are used for load calculation, namely a1, a2 and a3. These are known as the weight factors. The weight factors a_1, a_2 and a_3 provide a support for CPU utilization, network utilization and memory utilization respectively. For the cluster to execute instructions at a faster rate, the CPU utilization should be high. In the case of network utilization, if the data transfer speed is less, then the transfer time of the cluster will increase. Similarly, if the memory utilization is high, time taken for computation will increase. All these factors may affect the load of the system and in turn the system throughput. So, we choose weight values that are high for CPU utilization and network utilization. And small weight value for memory utilization. The algorithm calculates the average load of cluster using equation (5).

$$ALC_i=\sum_{k=1}^{m}\sqrt{a_1 CPU_{k,i}^2 + a_2 NU_{k,i}^2 + a_3 MU_{k,i}^2 + q_{k.i}^2} \quad (5)$$

Where $CPU_{k,i}$ is the current CPU utilization of resource k, $NU_{k,i}$ is the current network utilization of resource k, $MU_{k,i}$ is the current memory utilization of the resource k, and $q_{k,i}$ is the queue length of resource k in cluster i. a_1, a_2, a_3 are the three weight values with values 0.6, 0.3 and 0.1 respectively.To calculate the average system load using equation (6)

$$AL=1/m\sum_{i=1}^{m} ALC_i \quad (6)$$

Here ALC_i is the Average Load of Cluster *i*, *m* is the number of clusters in the system.

The deviation of Average System Load from ALC of each cluster is found out and the probability value is checked for confinement within the range of 0 to 1 as shown below.

$diff\_c_i=ALC_i$ – Average system load

if($0< P(diff\_c_i)<1$)

$underloaded\_list[]=c_i$

All clusters with probability of deviation within the given range is considered as under loaded. The clusters with values outside the given range is marked as overloaded and discarded from current scheduling cycle.

The ACP of these under loaded clusters is compared with the ECP value of jobs. Clusters with ACP less than or equal to ECP is marked as fittest and jobs are allocated to it. The algorithm is represented using Figure 1.

In HLBA, when a job request comes, the scheduler initializes the cluster parameters and retrieves cluster information from Grid Information system like ALC, memory utilization etc and computes the ACP for each cluster. The cluster with the highest ACP is selected and checked if its ALC is less than balance threshold. If the selected cluster is under loaded job is allocated to it. Otherwise, cluster with next highest ACP is which is underutilized is found out. The algorithm also employs local and global updates to inform the scheduler the current load status of the system and the clusters.

Whereas, in AHLBI, on receiving a job request the scheduler initializes job as well as cluster parameters. The job parameters include OS requirement, MIPS rating etc for a cluster. Whereas, the job parameters initialized by scheduler include, current memory utilization, current load on cluster, current number of jobs waiting to be executed etc. In our algorithm, we consider only the MIPS rating requirement in the task clause. The scheduler calculates ECP of each job together with ALC, Average System Load and ACP of clusters before job allocation. The algorithm find the deviation of ALC with the Average system load and find out the probability

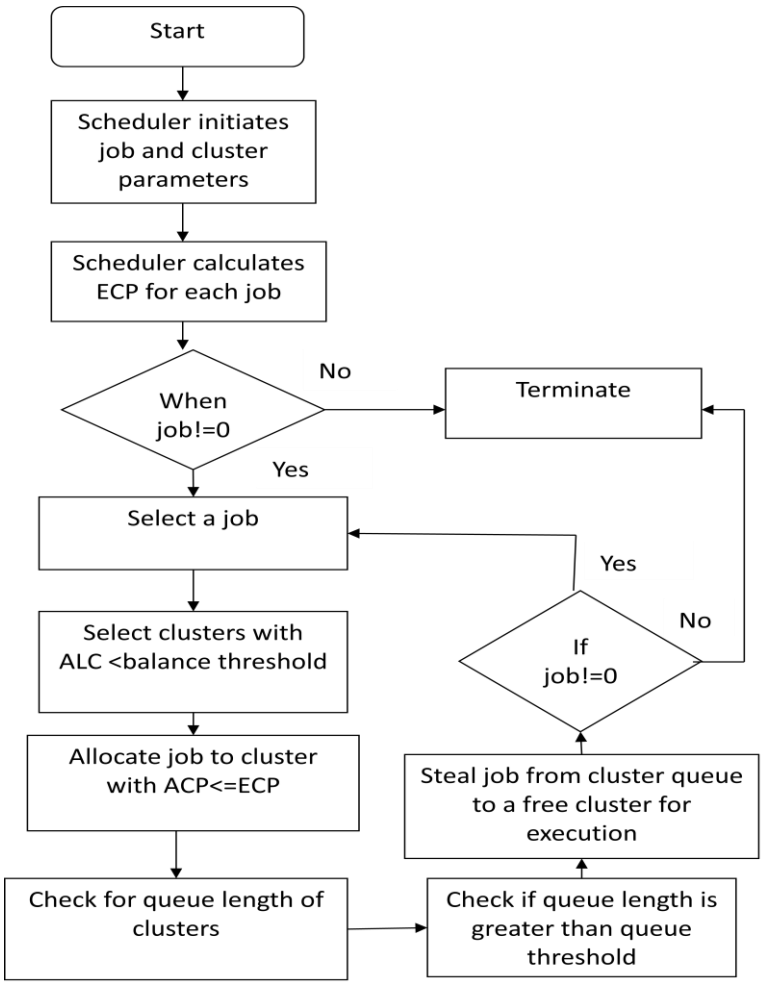

Figure 1. Flowchart for AHLBI

value of deviation for every cluster. If the probability of deviation is within the range of 0and 1, the cluster is marked as under loaded. The ACP of under loaded clusters is compared with the ECP of jobs. If the ACP value of a cluster is less than or equal to ECP of jobs, the cluster is considered as fittest and job is allocated to it.

After job allocation to clusters, some clusters may remain underutilized. This lead to wastage of processor time. To avoid this, AHLBI compares the queue length of all the clusters. Jobs from clusters with large queue size are stolen and allocated to free clusters. Similarly, when the number of jobs waiting to be executed in a cluster's queue increases, jobs from queue tail is allocated to free clusters for execution.

5. Implementation

In EHLBA task clauses in the job request are recorded and the number of tasks are initialized. A task clause includes the stipulations for a job. The number of tasks in a job is determined by the number of task clauses. The scheduler calculates the number of tasks t by perusing the number of task clauses. The computing power requirement, ECP of a job i is calculated using the equation (3).

ECP is the expected computing power requirement of a job i, $CPUSpeed_k$ is the MIPS rating required by the clusters to execute task k in job i. t is the number of tasks in job i.

The ACP, ALC, and Average system load is calculated using the HLBA algorithm. To calculate the load of each cluster the weighted values a_1, a_2, a_3 is chosen from a set of three, ie, a_1=0.6, a_2=0.3, a_3=0.1; a_1=0.3, a_2=0.6, a_3=0.1; a_1=0.1, a_2=0.3, a_3=0.6.

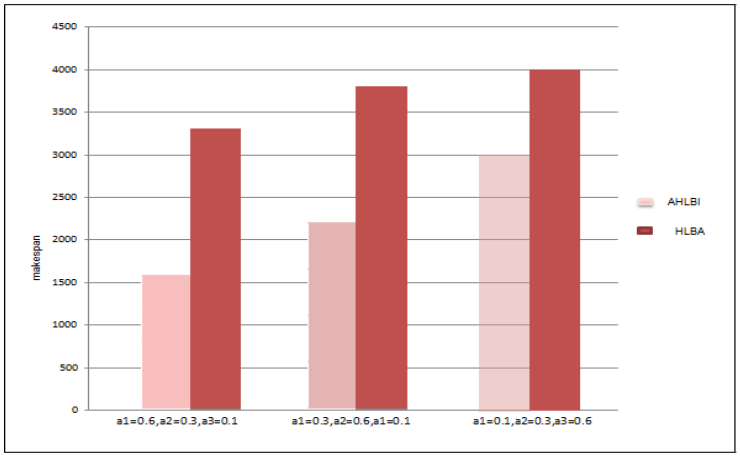

Figure 2. Graph Showing Makespan for Weighted Values in AHLBI and HLBA

The weighted value set, a_1=0.6, a_2=0.3, a_3=0.1, with reduced makespan is employed for load calculation of each resource in a cluster. Each weight factor corresponds to three different cluster parameters namely, CPU utilization, network utilization and memory utilization. AHLBI then calculates the average load of cluster ALC is calculated using Eq. (5). After calculating ALC for clusters, the algorithm calculates the Average System Load, AL using Eq. (6). The deviation between ALC of each cluster and the Average system load, which is calculated using Eq. (7) is ciphered. The probability that the deviation value x falls between the range ($0<x<1$) is checked for each cluster. If the condition is satisfied the cluster is marked as under loaded.

Before allocating jobs to under loaded clusters ECP of job is checked with the ACP of cluster. Clusters with ACP less than ECP of jobs are the most suitable resource and jobs are allocated to it. Fittest cluster is the one whose ACP is same as the ECP of jobs. Figure 2 shows the flowchart for algorithm execution.

The flowchart shows the algorithm execution steps. HLBA and AHLBI are initially compared. Both the algorithms are executed with 500 and 1000 jobs each. The experimental results show that EHLBA has lower makespan than HLBA. The parameters used for simulation are shown in Table 1.

Table 1. Parameters For AHLBI

Parameters	Value
Number of clusters	15
Number of jobs	50
Number of tasks	500
Computing power of resource nodes	3000-5000 MIPS
Baudrate	100-1000 Mbps

HLBA selects clusters based on the highest computing power. But the algorithm does not consider the load factor for these resources, which results in multiple search cycles to come up with a resource that is under loaded and with high ACP. Another demerit of HLBA is that a job will be allocated to a cluster with high ACP, even if the computing power requirement of the job is minimum. This can impact on the system throughput. Whereas, AHLBI, underutilized clusters are selected with ACP less than or equal to ECP. Figure 3 show the result of both the algorithms.

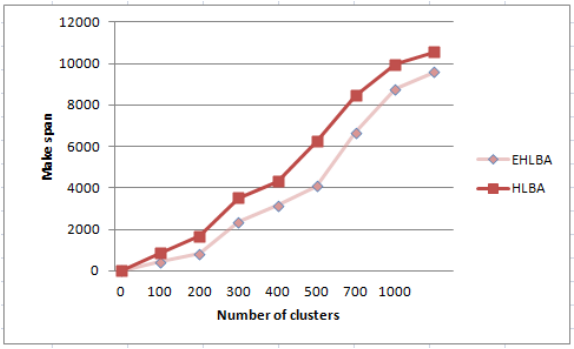

Figure 3. Graph Showing Makespan for HLBA and AHLBI

In AHLBI, the job allocation is done ensuring that the load of the system is balanced. AHLBI establishes more number of balanced clusters than HLBA. This improves the system performance together with load balancing which in turn improves the performance of individual resources in the system. This is reflected in the Figure 3, which shows the reduced makespan of individual clusters in the computational grid system.

6. Conclusion and Future Work

In this paper, an enhanced algorithm is proposed to reduce the makespan of algorithm execution and to allocate fittest resources to jobs coupled with load balancing. AHLBI collects job and cluster parameters to calculate the ECP and ACP for the jobs and clusters. In order to evaluate cluster imbalances the ALC and Average System Load are calculated. The clusters with probability of deviation between ALC and Average system load within the range of 0 and 1 are marked as under loaded clusters. The ACP of these clusters is compared with the ECP of jobs. If the value of a given cluster is less than or equal to ECP of a job, the cluster is considered fittest and job is allocated to it. The algorithm also employs methodology for reducing idle time of clusters. When a cluster finishes job execution, it steals jobs from clusters whose queue length is found to be high on comparison and executes the job. The experimental results show that the makespan of AHLBI is found to decrease than HLBA. The load balancing methodology employed in AHLBI is found to be less complex compared to that in HLBA.

The algorithm does not deal with dynamic heterogeneous clusters. In future, we would like to extend our work to accomplish load balancing using heterogeneous dynamic clusters.

References

[1] E. Holemen, M. Forbord, E. Gressetvold, A. C. Pedersen and T. torvatn, "A Paradox? Homogeneity in the imp perspective", 19th Annual International IMP Conference, (2003) September, pp. 1-20.
[2] H. Shan, L. Oliker, W. Smith and R. Biswas, "Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration", **(2004)**.
[3] S. Sharma, S. Singh and M. Sharma, "Performance Analysis of Load Balancing Algorithms", World

Academy of Science, Engineering and Technology, vol. 38, **(2008)**, pp. 269-272.
[4] M. Wu and X. H. Sun, "A General Self Adaptive Task Scheduling System for Non Dedicated Heterogeneous Computing", Research work supported by National Science Foundation, **(2005)**.
[5] Ungurean, "Job Scheduling Algorithm based on Dynamic Management of Resources Provided by Grid Computing Systems ", Electronics and Electrical engineering, vol. 103, no. 7, **(2010)**.
[6] B. Yagoubi and Y. Slimani, "Load Balancing Strategy in Grid Environment", Journal of Information Technology and Applications, vol. 1, no. 4, **(2007)** March, pp. 285-296.
[7] C. M. Grinstead and J. L. Snell, "Introduction to Probability".
[8] E. E. Ajaltouni, A. Boukerche and M. Zing, "An Efficeint Dynamic Load Balancing Scheme for Distributed Simulations on a Grid Infrastructure", 12th 2008 IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, pp. 61- 69.
[9] R. S. Chang, C. F. lin and J. J. Chen, "Selecting the most fitting resource for task execution", Future Generation Computer Systems, vol. 27, **(2011)**, pp. 227-231.
[10] R. S. Chang, J. S. Chang and P. S. Lin, "An Ant Algorithm for Balanced Job Scheduling in Grids", Future generation Computer Systems, vol. 25, **(2009)**, pp. 20-27.
[11] Y. H. Lee, S. Leu and R. S. Chang, "Improving job scheduling algorithms in a grid environment", Future generation computer systems, **(2011)** May.
[12] R. V. van Nieuwpoort, T. Kielmann and H. E. Bal, "Efficient Load Balancing for Wide- Area Divide-and-Conquer Applications", ACM, PPOPP'01, **(2001)** June 18-20.

Authors

R. Joshua Samuel Raj

Affiliation: Assistant Professor / CSE, Karunya University

Brief Biographical History:

2005 -Graduated in 2005 from the Computer Science and Engineering Department from PETEC under Anna University

2007 -Received M.E Degree in Computer Science and Engineering from Jaya College of Engineering under Anna University

2009 Working towards the Ph.D degree in the area of Grid scheduling under Kalasalingam University

Main Works:

Grid computing, Mobile Adhoc Networking, Multicasting and so forth

Hridya K. S

Affiliation: PG student, SE, Karunya University

Brief Biographical History:

Graduated in 2009 from Information Technology Department from Mangalam College of Engineering under Mahatma Gandhi Unversity. Currently persuing M. Tech in Software Engineering in Karunya University.

V. Vasudevan

Affiliation: Director, Software Technologies Lab, TIFAC Core in Network Engineering, Srivilliputhur, India

Brief Biographical History:

1984- M. Sc in Mathematics and worked for several areas towards Representation Theory

1992 Received his Ph.D. degree in Madurai Kamaraj University

2008- the Project Director for the Software Technologies Group of TIFAC Core in Network Engineering and Head of the Department for Information Technology in Kalasalingam University, Sirivilliputhur, India

Main Works:

Grid computing, Agent Technology, Intrusion Detection system, Multicasting and so forth