International Journal of Advanced Research in IT and Engineering

ISSN: 2278-6244
Impact Factor: 4.054

LOAD DISTRIBUTION USING REFINED TASK ALLOCATION HEURISTICS

Prof. Minal Shahakar*

Prof. Rupesh Mahajan*

Abstract: *Distribution of tasks to different available resources is done using various heuristics like minmin+, maxmin+, sufferage+ and also using various hybrid techniques that is combination of different heuristics. Although using this heuristics tasks are distributed faster as well as execution also becomes faster, but still load balancing across resources is not achieved. This is overcome by our proposed load balancing algorithms Threshold based task allocation policy and Average Load balancing, both of these algorithms achieves load distribution in different manner and reduces time complexity and improve the system performance. In task allocation policy tasks are distributed on the basis of threshold value and in average load balancing algorithm average load of all available resources is calculated for all tasks, thus to assign best resource for each task.*

Keywords: *Load Balancing, Heterogeneity, Hybrid Heuristics, Task Assignment, Makespan, Minimum Completion Time.*

*Pad. Dr. D.Y.Patil Institute of Engg. & Techlogy, Pimpri, Pune

I. INTRODUCTION

Task allocation is main goal of distributed systems along with load balancing across all the available resources. Tasks are distributed using various heuristics techniques based on minimum completion time of each task. The term metatask is also very important that can defined as a set of task which is used for batch mode. There are two types of mode online mode and batch mode. In online mode, tasks are available to the resource for execution at run time wherein batch mode tasks are collected in a queue that is nothing but metatask. Since, in our proposed algorithm batch mode is used that is metatask.

Load balancing must be achieved during this task allocation to make utilization of all the available resources. When some heuristics is being applied to execute task, it is possible that some of the resources remains idle and resources with already having some task on it get more heavily loaded which results in slow execution of task and reduces system performance. Since load balancing heuristics must be applied to overcome such problem.

Heuristics like minmin+, maxmin+, and sufferage+ are applied to allocate tasks to available resources using minimum completion time, makespan and sufferage value respectively. Minmin+ allocates task to available resource that is having minimum completion time. Maxmin+ considers makespan, the term makespan is also very important that can be defined as maximum of overall completion time, the task having greater makespan is selected and allocated to available resource. Wherein, sufferage+ task is being allocated to available resource that is having greater sufferage value. Sufferage value here can be defined as the difference between two MCT values. All these heuristics are suitable for large scale application but doesn't support load balancing.

Our proposed algorithms which are described in following sections supports load balancing criteria in different manner. Since they distribute the tasks based on threshold value and average load.

II. LITERATURE REVIEW

Literature Review is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start

building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites.

Related work in literature was examined to select a set of heuristics appropriate for the heterogeneous computing environment is considered. In the literature survey of "Enhancing Performance of Task Assignment on The Basis of Heuristics Algorithm", mapping tasks onto machines is often referred to as scheduling. Several researchers have worked on the dynamic mapping problem from areas including job shop scheduling and distributed computer systems.

M. Maheswaran. [2], have proposed that the Minmin heuristic starts with a set of all unmapped tasks. The machine that has the minimum completion time for all jobs is selected. Then the job with the overall minimum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped tasks are assigned. Compared to MCT this algorithm considers all jobs at a time. So it produces a better makespan, in other words it begins with the set U of all unmapped tasks. Then, the set of minimum completion times, i.e. for each task is found. Next, the task with the overall minimum completion time from metatask is selected and assigned to the corresponding machine (hence the name Minmin).

T. Kokilavani. [4], have presented Opportunistic Load Balancing (OLB) system that assigns the jobs in a random order in the next available resource without considering the execution time of the jobs on those resources. Thus it provides a load balanced schedule but it produces a very poor makespan, in other words it assigns each task, in arbitrary order, to the next machine that is expected to be available, regardless of the task's expected execution time on that machine. The intuition behind OLB is to keep all machines as busy as possible. One advantage of OLB is its simplicity, because OLB does not consider expected task execution times, the mappings it finds can result in very poor makespan.

Doreen. D et al. [7], have proposed an efficient Set Pair Analysis (SPA) based task scheduling algorithm named Double Minmin Algorithm which performs scheduling in order to enhance system performance in Hypercubic P2P Grid (HPGRID). The simulation result shows that the SPA based Double Minmin scheduling minimizes the makespan with load balancing and guarantees the high system availability in system performance.

Laszewski. G.V.[11], have proposed QoS tasks scheduling al-gorithm as an aggregation formula in a specific architecture called Grid-JQA. Such formula is a combination of parameters and weighting factors to evaluate QoS. Khanli's scheduling algorithm is not practical as it hasn't a practical mathematical solution. A new algorithm based on the conventional Minmin algorithm. The proposed algorithm which is called QoS guided Minmin, schedules tasks requiring high bandwidth before the others. Therefore, if the bandwidth required by dif-ferent tasks varies highly, the QoS guided Minmin algorithm provides better results than the Minmin algorithm. Whenever the bandwidth requirement of all of the tasks is almost the same, the QoS guided Minmin algorithm acts similar to the Minmin algorithm.

Kamalam et al.[12], presents a new scheduling algorithm named Min-mean heuristic scheduling algorithm for static mapping to achieve better performance. The proposed algorithm reschedules the Minmin produced schedule by considering the mean makespan of all the resources. The algorithm deviates in producing a better schedule than the Minmin algorithm when the task heterogeneity increases.

R. Joshi. C.[13], have proposed an algorithm depends on the original Minmin algorithm. It is called QoS guided Minmin, and it assigns tasks with high bandwidth before others. QoS acts similar to Minmin when available tasks have the same bandwidth so it preferred to use QoS guided Minmin when-ever submitted tasks have large bandwidth. At that moment, QoS guided Minmin produces better results.

Two heuristic algorithms: QoS Guided Weighted Mean Time-Min(QWMTM) and QoS Guided Weighted Mean Time Min-min Maxmin Selective(QWMTS). Both algorithms are for batch mode independent tasks scheduling. The network bandwidth is taken as QoS parameter.

Suri. P.K, [14]. provided a new algorithm, that uses Maxmin and Minmin algorithms to select one of these two algorithms depending on standard deviation of the expected completion times of the tasks on each of the resources.

A QoS based predictive Maxmin, Minmin Switcher algorithm for scheduling jobs in a grid. The algorithm makes an appro-priate selection among the QoS based Maxmin or QoS based Minmin algorithm on the basis of heuristic applied, before scheduling the next job. The effect on the execution time of grid jobs due to non-dedicated property of resources has

also been considered. The algorithm uses the history information about the execution of jobs to predict the performance of non-dedicated resources.

A load balancing and distributed systems aims to increase the utilization of resources with light load or idle resources thereby freeing the resources with heavy load. The heuristics algorithm tries to distribute the load among all the available resources. At the same time, it aims to minimize the makespan with the effective utilization of resources. The Non-traditional algorithms differ from the conventional traditional algorithms in that it produces optimal results in a short period of time. There is no best scheduling algorithm for all computing systems. A set of static heuristic for task scheduling in heterogeneous computing environments are available. A range of simple greedy constructions heuristic approaches are compared and some of them are briefly described below:-

In contrast to OLB, Minimum Execution Time (MET) assigns jobs to the resources based on their minimum expected execution time without considering the availability of the resource and its current load. This algorithm improves the makespan to some extent but it causes a severe load imbalance, in other words it assigns each task, in arbitrary order, to the machine with the best expected execution time for that task, regardless of that machine's availability. The motivation behind MET is to give each task to its best machine. This can cause a severe load imbalance across machines.

Minimum Completion Time (MCT) assigns jobs to the resources based on their minimum completion time. The completion time is calculated by adding the expected execution time of a job on that resource with the resources ready time. The machine with the minimum completion time for that particular job is selected. But this algorithm considers the job only one at a time, in other words assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task. This causes some of the tasks to be assigned to the machines that do not have the minimum execution time for them. The intuition behind MCT is to combine the benefits of OLB and MET, while avoiding the circumstances in which OLB and MET perform poorly.

Yagoubi. B et al.[15], have offered a model to demonstrate grid architecture and an algorithm to schedule tasks within grid resources. The algorithm tries to distribute the workload of the grid environment amongst the grid resources, fairly. Although, the mechanism used here and other similar strategies which try to create load balancing within

grid resources can improve the throughput of the whole grid environment, the total makespan of the system does not decrease, necessarily.

III. TASK DISTRIBUTION HEURISTICS

Minmin+ Heuristic

Minmin+ heuristic method combines advantages of minmin algorithm overcoming disadvantages like low run time performance of system it. Since in this algorithm MCT values are separately maintained in queue in sorted order wherein minmin each time MCT for each task needed to be calculated before assigning them to available resource. Steps for Minmin+ algorithm are given below.

Algorithm: Minmin+ Heuristic

1. Initialize current load L_k, unassigned tasks F[i], MCT Value Q
2. for all unassigned tasks T_i
3. for all processors P_k
4. Calculate completion time

   Cik = Eik + Rk
5. If current load Lk and execution time x of Pk is less than threshold value then
6. Assign task with minimum completion time to Pk
7. Task id of assigned task to processor is returned
8. Delete task Ti from queue
9. Update ready time of processor

Minmin+ contains various types of operation such as sorting of MCT values in queue, maintaining task id's and then deletion of assigned task to available resources as well as array is maintained that indicates which task is not yet assigned to available resource by maintaining task id. Since step 1 contains initialization of variables such as current load Lk on processor Pk, initially array will be initialized as false because not any task is assigned to processor yet, and finally the queue will be empty because MCT values are not yet calculated. In step 4 minimum completion time of each task is calculated and arranged in queue in sorted order. This MCT values are referred by this minmin+ heuristic to assign the task to available processor. Since in further steps current load and execution time of processor Pk is compared with threshold value, if it is less then threshold value, than the task with minimum completion time is assigned to the processor Pk and array that is F[i] will

return get initialize to true by returning task id and finally the task Ti assigned to processor Pk will be deleted from queue. This process is repeated until all the tasks are assigned to available processors.

Maxmin+ Heuristic

Maxmin+ heuristic works similar to minmin+ except it differs in task selection policy. Here the tasks are selected on the basis of makespan that is maximum overall completion time of task. The task with highest makespan is assigned to processor. Since this algorithm produce better results as compare to maxmin by making use of minmin+ strategy for initialization of variables and selection of task, and maxmin strategy for comparison of current load and execution time with threshold value.

Algorithm: Maxmin+ Heuristic

1. for all unassigned tasks Ti
2. for all processors Pk
3. Calculate completion time

   Cik = Eik + Rk
4. If current load Lk and execution time x of Pk < threshold value then
5. Initialize threshold value with current load and execution time
6. Select task Ti with MCT and processor Pk for task assignment
7. If current load and execution time > makespan then
8. for all processors Pk
9. If threshold > than max value
10. Initialize makespan with current load and execution time
11. select task Ti with overall maximum completion time and assign to processor Pk
12. Update ready time of processor

Once variables are initialized, completion time is calculated in step 3. Step 5 compares threshold value with current load and execution time, if it is less, then threshold value get replaced by this current load and execution time value and task with minimum completion time is selected, this threshold value is further compared with makespan, if it is greater, than makespan value is initialize to the current load and execution time value and hence the task with overall maximum completion time is selected and assigned to available processor. Since working of maxmin+ is assigning the task to processor with longest

completion time that is executed concurrently with the other remaining task with minimum completion time. Since working of maxmin+ is faster as compare to minmin+. Once these tasks are assigned to processor, then assigned task are deleted from queue and the whole process is repeated until all the tasks are assigned to available processors. Steps for Maxmin+ are shown in algorithm 2.

Sufferage+ Heuristic

To make sufferage heuristic useful for large scale application, it is combined with minmin+ under a new heuristic referred as Sufferage+. This heuristic makes use of minmin+ strategy along with Sufferage heuristic. Since again the task assignment is done on the basis of highest sufferage value of task that is defined as difference between first and second MCT values of tasks. Steps for Sufferage+ algorithm are given below.

Algorithm: Sufferage+ Heuristic

1. for all unassigned tasks Ti
2. Find the overall highest sufferage value and processor to allocate task.
3. for all processors Pk
4. Calculate completion time

   Cik = Eik + Rk
5. If current load Lk and execution time x of Pk < threshold value then
6. Initialize threshold value with current load and execution time
7. Select task Ti with MCT and processor Pk for task assignment
8. If current load and execution time is greater than makespan then
9. for all tasks Ti
10. Calculate sufferage value:

    suffvalue = Second_MCT –First MCT
11. Find task with highest sufferage value
12. Initialize makespan with current load and execution time
13. Assign task Ti with highest sufferage value to processor Pk
14. Update ready time of processor

Sufferage+ heuristic initialize variables applying the minmin+ strategy, further in step 4 completion time is calculated. Step 6 compares threshold value with current load and execution time, if it is less, then threshold value get replaced by this current load and

execution time value and task with minimum completion time is selected, this threshold value is further compared with makespan, if it is greater, than for all tasks sufferage value is calculated as difference between second and first MCT in step 10. In further steps task with highest sufferage value is found and assigned to processor. Since these steps are repeated until all the tasks are assigned to available processors.

Randomize Heuristic

This heuristic is one type of hybrid heuristic that is combination of more than one heuristic. Since in this heuristic method minmin+ and maxmin+ heuristics are combined and called randomly for task allocation to processor based on MCT values. Since this heuristic make use of standard deviation concept to compare with threshold value to make a call for respective heuristic method for allocating tasks to available resources. Steps for Randomize are shown below:

Algorithm: Randomize Heuristic

1. for all unassigned tasks Ti
2. for all processors Pk
3. Calculate completion time

   Cik = Eik + Rk
4. Calculate the SD of completion time of all unassigned tasks.
5. Sort all unassigned tasks in increasing order of their completion times.
6. Find a position in this list where difference in completion time of two consecutive tasks is more than SD.
7. If SD < threshold value

   Apply existing heuristic
8. Else

   Apply refined heuristic

Hybrid heuristic initialize variables applying the minmin+ strategy, further in step 3 completion time is calculated. Step 4 calculates the standard deviation value to compare it with threshold value. Since, all the tasks are sorted based on their MCT values in increasing order. Finally the standard deviation is compared with threshold value, if it is less, then minmin+ heuristic is applied else maxmin+ heuristic is applied.

IV. LOAD BALANCING HEURISTICS

Average Load Balancing

To better utilize the resources, the load should be balanced on all the resources by allocating each of them with some task. In a heterogeneous system, execution time of a task varies on different resources. The task size can simply be measured as the average of the execution times over all resources. Tasks can be allocated with priorities based on their sizes, either favorable to the smaller tasks or to the larger tasks. Or the priority can be allocated independently of task sizes. It has been found that the load is severely imbalanced when smaller tasks are assigned first. If large tasks are given higher priorities, it generally leads to a better balanced load. Also, a priority independent of tasks sizes has a fairly good chance for load balancing. Heuristic mentioned above like minmin+ and maxmin+ maps different tasks to different available resources efficiently but it does not maintain proper load balancing among the resources due to which some resources are utilized and some remain idle. This load balancing concept can be applied to this scheduling to make execution faster [4]. Load balancing algorithm is shown below:

Algorithm: Average Load Balancing Heuristic

1. for all unassigned tasks Ti
2. for all processors Pk
3. if $e_k + \emptyset_i < min$
4. Calculate current load of processor Pk for task Ti

   $e_k \leftarrow e_k + \emptyset_i$
5. $min \leftarrow e_k$
6. k' ← k
7. for all processors Pj
8. while j < k do
9. Assign Pj ← Ti
10. Update load to processor

In this heuristic our approach is to find the average load of task Ti with respect to all the processors Pk that is calculated using equation (1). Step 4, compares the current load ek of processor and $\emptyset_i$ average load with threshold value and update the current load ek of processor Pk if condition satisfies. Step 8 initializes backup variable μ with current processor

k which is use to update the current load e'_μ of processor that has been assigned some task for execution. As this algorithm checks the load on each processor for given no. of tasks, the initial load backup is require which is stored in e_j that is used to update processor Pk that has not been assigned with any task Ti yet. The aim of our work is to minimize the makespan of the system by making utilization of all the available resources.

Minmin heuristic selects tasks with minimum completion time and allocates it to available resource, due to which task with maximum execution time remains unallocated although the resource is available that causes resources to remain idle. Similarly in maxmin heuristic tasks with maximum completion time is selected and allocated to resources, due to which tasks with minimum completion time are allocated after long time to available resources [7]. Solution for above is to apply such load balancing concept with this types of heuristic. This can be done when some heuristic method is applied to allocate tasks to available resources using minimum completion time, for balancing the load across resources the allocated tasks are rescheduled and allocated to respective available resource to make utilization of all the available resources [10].

δ Based Task Allocation

As our dissertation work mainly focuses on load balancing strategies, this heuristic is again another approach to balance load across system and distribute tasks. This heuristic is based on Threshold value task distribution concept that is server distributes task among clients. If task load is greater than threshold value that is if client machine is unable to execute complete task then it will report to server machine about partly completion of task and time it took to execute that task, then server will again redistribute remaining part of task to another client machine based on IP address of client machine that it have stored initially. Load balancing algorithm based on threshold value is shown below:

Algorithm: δ Based Task Allocation

1. for all unassigned task Ti
2. Check availability of clients C1, C2,….Cn
3. for all clients C1, C2,….Cn
4. Find current load ek
5. Calculate threshold δ
6. if current load ek < threshold δ

Ci ← Ti

7. Else

   Cj ← Ti

8. Calculate MCT for task Ti on Ci and Cj

   MCT ← MCT1 + MCT2

9. Delete task Ti from queue

In above heuristic load is balanced on the basis of threshold value δ. As shown in step 2 Server machine check client machine availability, then in step 3 it store the IP addresses of those machines that are available, it finds Current load ek of each client machine and threshold value is calculated using following equation:

$$\delta \leftarrow \sum_{k=0}^{n} e_k / n$$

In step 4, calculated δ is compared with current load ek, if ek of client Ci is less than δ then task Ti is allocated to Ci, else Ti is allocated to Cj. Even if task is partly executed, remaining part of it can be executed on another client machine that is Cj. finally in step 5 minimum completion time MCT is calculated required for task Ti to execute partly on client Ci and partly on client Cj. This calculated time is shown by server as a result and completed task Ti is deleted from queue.

V. RESULTS

In this section the empirical results are presented to show the effectiveness of the Enhanced Heuristics and Load Balancing algorithms. The performances of algorithms are tested on several datasets and tasks.

The performance of algorithm is compared with minmin, maxmin and sufferage heuristics which are among the standard heuristics algorithms at present. We have studied various datasets as well as some tasks related input files that were possible to apply for calculating MCT values. We have also shown results in graphical format that provides better visualization of which task is assigned to which processor. Since after experimentation results are improved as compared to previous method because we have also implemented two load balancing algorithms that are Average and threshold based heuristics, wherein we calculate average of all nodes for each task and update load accordingly of selected node.

ISSN: 2278-6244
Impact Factor: 4.054

In Average Load Balancing Heuristic, initial and final load, both are considered, that is initial load before allocating task to resource and final load means after allocation of task to resource as shown in figure 1:

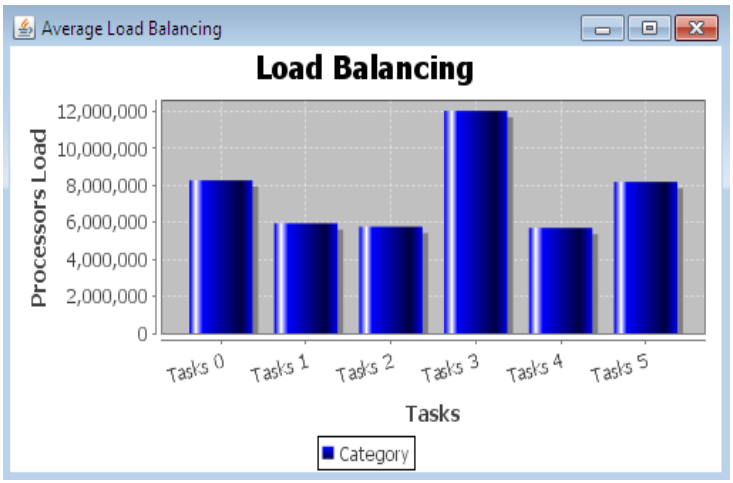

Figure 1: Graphical Representation of Load balancing Heuristic

In threshold based task allocation policy threshold value is calculated and task is distributed among nodes that provide us the better results by making utilization of all the available resources as shown in following results:

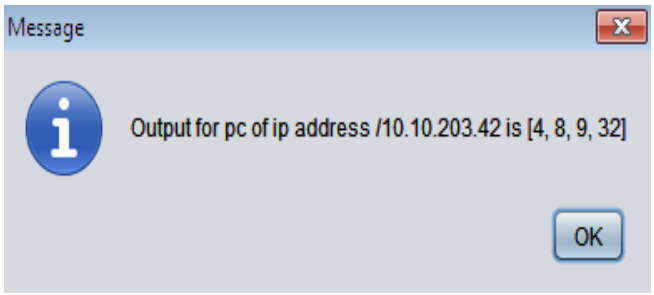

Figure 2: Task done by pc (10.10.203.42)

Figure 2 shows once the connection is successful, server node search for available client nodes and distribute task of sorting some given numbers to respective client node based on their initial load value.

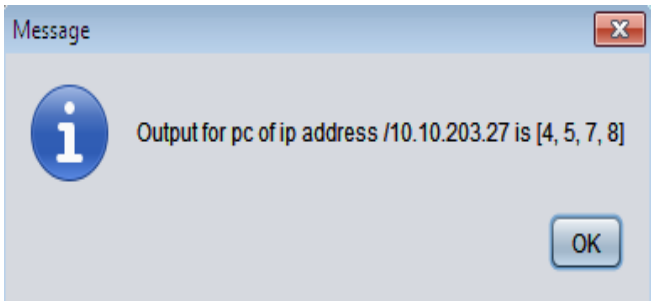

Figure 3: Task done by pc (10.10.203.27)

ISSN: 2278-6244
Impact Factor: 4.054

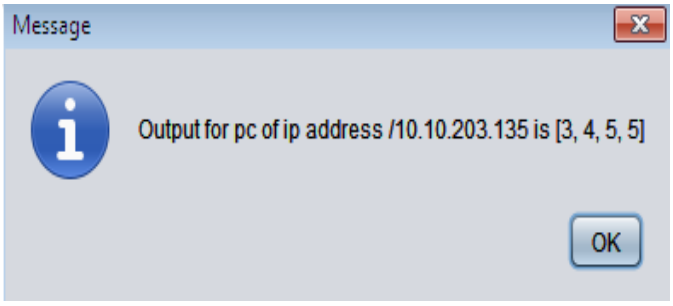

Figure 4: Task done by pc (10.10.203.135)

Figure 3 and figure 4 represent same function as figure 2 that is how many numbers are being sorted by respective client machine.

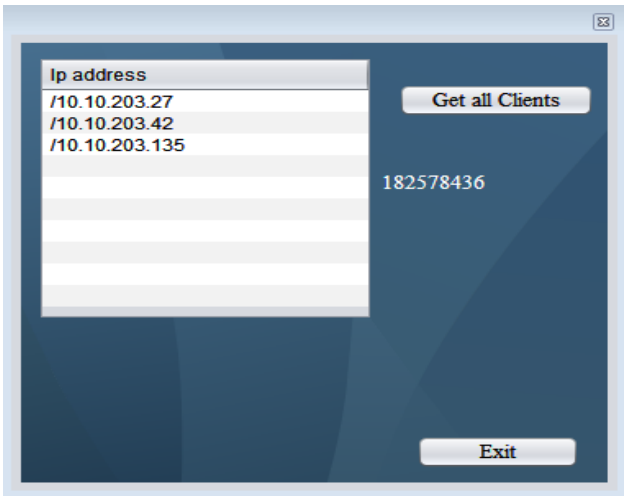

Figure 5: Threshold Load Balancing After Task Completion

Finally, figure 5 shows which client nodes the task was allocated by retrieving IP addresses of client nodes and overall threshold value after task is being distributed to client nodes. This results in better system performance and reduces overall complexity.

We were successful to display the information of assigned task and processor like processor ip address, initial load, final load that is load after task allocation, threshold value, etc. So after comparing the results it becomes possible to find out which algorithm is most suitable for distributing tasks to different available resources.

VI. CONCLUSION

We have proposed a completely new approach for load balancing using average load calculation of task on each processor. Distribution of task in heterogeneous environment is done using various heuristic methods like minmin, maxmin and sufferage heuristic. Enhanced heuristic presents certain performance improvements over existing heuristics by

providing the better results in minimum time. Also this heuristic improves the worst case runtime complexity in assigning N in-dependent tasks to K processors. Moreover, the heuristic MaxMin+ and Sufferage+ heuristics, which are hybrid versions of MaxMin and Sufferage, obtained by combining the latter heuristics with MinMin and combines the advantages of minmin and maxmin and overcomes the disadvantages of these heuristic algorithms. Still this heuristics doesn't provide proper load balancing which is overcome by our designed algorithm that is Average Load Balancing which calculates load on each machine and allocates tasks accordingly.

REFERENCES

[1] Kamali Gupta, Manpreet Singh, —Heuristic Based Task Scheduling In Grid, International Journal of Engineering and Technology (IJET), vol. 4, pp. 254-260, Aug-Sep 2012.

[2] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, —Dynamic mapping of a class of independent tasks onto heterogeneous computing sys-tems, J. Parallel Distrib. Comput., vol. 59, pp. 107131, 1999.

[3] H. J. Siegel and S. Ali, —Techniques for mapping tasks to machines in hetero-geneous computing systems, J. Syst. Archit., vol. 46, no. 8, pp. 627639, 2000.

[4] T. Kokilavani, Dr. D.I. George Amalarethinam, —Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing, International Journal of Computer Applications, vol. 20, April 2011.

[5] George Amalarethinam. D.I, Vaaheedha Kfatheen .S, —Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems, International Journal of Computer Science and Information Technologies, Vol. 3, pp. 3659-3663, 2012.

[6] K. Kaya, B. Ucar, and C. Aykanat, —Heuristics for scheduling file-sharing tasks on heterogeneous systems with distributed repositories, J. Parallel Distrib. Comput., vol. 67, no. 3, pp. 271285, 2007.

[7] Doreen Hephzibah Miriam. D and Easwarakumar. K.S, —A Double Min Min Algorithm for Task Metascheduler on Hypercubic P2P Grid Systems, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 5, July 2010.

[8] He. X, X-He Sun, and Laszewski. G.V, —QoS Guided Minmin Heuristic fr Grid Task Scheduling, Journal of Computer Science and Technology, Vol. 18, pp. 442-451, 2003.

[9] T. D. Braun,H. J. Siegel,N. Beck, L. L. Boloni, —A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous dis-tributed computing systems, J. Parallel Distrib. Comput., vol. 61, no. 6, pp. 810837, 2001.

[10] Doreen Hephzibah Miriam. D and Easwarakumar. K.S, \A Double Min Min Algorithm for Task Metascheduler on Hypercubic P2P Grid Systems", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 5, July 2010.

[11] He. X, X-He Sun, and Laszewski. G.V, \QoS Guided Minmin Heuristic for Grid Task Scheduling", Journal of Computer Science and Technology, Vol. 18, pp. 442-451, 2003.

[12] Kamalam.G.K and Muralibhaskaran.V, , \A New Heuristic Approach:Min- Mean Algorithm For Scheduling Meta-Tasks On Heterogenous Computing Systems", International Journal of Computer Science and Network Security, VOL.10 No.1, January 2010.

[13] Sameer Singh Chauhan,R. Joshi. C, \QoS Guided Heuristic Algorithms for Grid Task Scheduling", International Journal of Computer Applications (09758887), pp 24-31, Volume 2, No.9, June 2010.

[14] Singh. M and Suri. P.K, \QPS A QoS Based Predictive Max-Min, Min-Min Switcher Algorithm for Job Scheduling in a Grid", Information Technology Journal, Vol. 7, pp. 1176-1181, 2008.

[15] Yagoubi. B, and Slimani. Y, \Task Load Balancing Strategy for Grid Compu-ting", Journal of Computer Science, Vol. 3, No. 3, pp. 186-194, 2007.