Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)

Volume 7-Issue 1, January 2024
Paper: 69

A Clustered Task Scheduling Method for Heterogeneous Distributed System

Lipika Datta

Assistant Professor
Computer Science and Engineering department
College of Engineering and Management Kolaghat
KTPP Township, W.B., India
lipika.datta@cemk.ac.in

Abstract:- **A distributed system consists of several autonomous nodes. In a distributed system some of the nodes may be overloaded due to a large number of job arrivals while other nodes may remain idle without any processing. The performance of the distributed system depends on how the tasks are scheduled. If tasks are allocated wisely, then good performance of the system can be obtained. The primary goal of task scheduling in distributed system is to minimize the total execution time, so that maximum speed-up and efficiency can be achieved. The performance of a distributed system depends crucially on dividing up work effectively among the computing nodes. So a way is needed to share load across all the computing nodes. In centralized load balancing schemes, the load balancing decision is taken by a central server. So this scheme is not scalable. In contrast, fully distributed schemes are scalable, but they use local information. A hierarchical dynamic task scheduling model is proposed in this paper where an ordinary node does not need to have a global system wide knowledge about the states of other nodes in the system. The proposed model is semi distributed as each cluster is represented by a cluster master.**

Keywords: clustered distributed system, hierarchical load balancing, heterogeneous distributed system, average turnaround time, task scheduling.

I. Introduction

A distributed system can be viewed as a collection of computing and communication resources shared by active users. Users submit jobs at random times. In such a system, some computers are heavily loaded while others have available processing capacity. The objective of a load balancing protocol is to transfer the load from heavily loaded nodes to idle nodes, such that no processing element become neither overloaded nor idle, hence balance the load at the computers and increase the overall system performance. When the demand for computing power increases load balancing problem becomes important. Load balancing improves the performance of a distributed system through the appropriate distribution of load. This helps in improving response time by minimizing job's execution time, minimizing communication overheads and maximizing resource utilization. For a heterogeneous environment there is a wide variety of issues that need to be considered viz. different processors may have different capacities because of processor heterogeneity and according to loads imposed on them. Load balancing operation consists of four policies [1]: information policy, transfer policy, location policy and selection policy.

Selection policy works in preemptive or non-preemptive fashion. Newly generated process is selected by non-preemptive transfer policy and the running process is selected by preemptive transfer policy. Preemptive transfer policy is more costly than non-preemptive transfer policy while the preemptive one provides better load balancing.

The information policy is responsible for keeping up-to-date load information about each node in the system. A global information policy provides access to the load index of every node, at the cost of additional communication for maintaining accurate information.

The transfer policy deals with the dynamic aspects of a system. It uses the nodes' load information to decide when a node becomes eligible to act as a sender (transfer a job to another node) or as a receiver (retrieve a job from another node). Transfer policies are typically threshold based.

The location policy selects a partner node for a job transfer transaction. If the node is an eligible sender, the location policy seeks out a receiver node to receive the job selected by the selection policy.

Once a node becomes an eligible sender, a selection policy is used to pick which of the queued jobs is to be transferred to the receiver node.

The selection policy uses several criteria to evaluate the queued jobs. Its goal is to select a job that reduces the local load, incurs as little cost as possible in the transfer, and has good affinity to the node to which it is transferred. Selection policy works in preemptive or non-preemptive fashion. Newly generated process is selected by non-preemptive policy and running process is selected by preemptive policy. Preemptive policy is more costly than non-preemptive policy while the preemptive one provides better load balancing.

Load can be specified in many different ways.

1. Parameters for Static Load Balancing:

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7-Issue 1, January 2024
Paper: 69

a) Processor parameters including number of processor, speed of processors and ratio of speed of i^{th} processor with respect to a base processor.
b) The program parameters including data size, number of loop iterations, work per iteration, data communication per iteration and execution time of each of the iteration on a base processor.
c) Network parameters including network latency, network bandwidth and network topology.

2. Parameters used for load measurement in dynamic load balancing are:
   a) Processor Queue Length
   b) Execution Time
   c) Utilization of RAM
   d) CPU utilization

In addition to load indices, the performance of load balancing algorithms is also affected by several other factors related to collection of load information [15]. These factors are:

(a) Collection of processing load: Workload can be collected using job traces in trace driven simulation study. Workload can also be generated synthetically. Generally, Poisson's process arrival and exponentially distributed service times are used. Inter-arrival time distribution and service time distribution can also be modeled using Lognormal, Weibull or Pareto distribution if variance in distribution of inter-arrival time and service time is high. In peak times, arrival of processes is less bursty and service time distribution has a low variation.

(b) Load update interval: The collection of load information can be periodic or event based. A typical period is one second or longer while typical events are process creation, termination or migration. The intervals between load index updates should be chosen carefully for the stability of the system. If the interval in collecting load information is too long, the basic objective of load balancing is defeated. This results in poor performance as the load balancer is maintaining outdated information about the system load. On the contrary, collecting load information in short intervals may result in increased networking overheads and over reaction by the load balancing algorithm [16].

In this paper workload is specified in terms of process size, processor speed, processor queue length and utilization of RAM. Depending on workload each node acts as a sender or a receiver. For proper load balancing each node must have knowledge about the all other nodes of the system. The sender/receiver has to discover a partner node by sending a broadcast message to all other nodes and expecting a response. This creates huge traffic in the network and a considerable communication overhead when all the nodes are involved.

The focus of this work is to design a load balancing algorithm which considers the dynamic, scalable and heterogeneous nature of the distributed environment. To reduce the complexity of the load balancing process, a load balancing method that uses a two-level strategy to balance workload by proper task scheduling among the nodes, is proposed. The total system is grouped into clusters consisting of a subset of nodes in the system. In the first step any lightly loaded node is searched for within the home cluster, otherwise a lightly loaded node is searched for in other clusters. The proposed algorithm privileges local load balancing over global load balancing so that the communication cost is reduced. Simulation results show that the proposed method performs significantly better in minimizing the average turnaround time and balancing load among computing nodes in comparison to some existing algorithms.

The work is organized as follows: The status of the considered domain is presented in section II. Section III describes the system model and the responsibilities of each type of node. The 2-Level Task Scheduling Policy (2LTS) is discussed in Section IV. Section V analyses the communication cost. Simulation results of our experiments are presented in Section VI. Section VII includes the conclusion part.

II. Related work

Studies on Load balancing for distributed computing system have been going on for a long time. Load balancing algorithms are broadly classified in two categories: static and dynamic.

Static load balancing: Processes are assigned to processors at compiling time. Assignment of jobs to processors depends on static information such as CPU capacity, memory space, etc. to load balancing purpose [2,3,4]. These algorithms do not collect any information about the node. Once the processes are assigned no reassignment can be done in run-time.

Dynamic load balancing: Assignment of jobs is done at run-time. Depending on the current state of the system, load can be transferred from heavily loaded node to lightly loaded node therefore is able to further improve the system performance [3,5,6]. With the increase of number of processors the communication overhead also increases. ELISA [7] uses periodic status exchange concept and load balancing decision is taken based only on queue length. Kielmann et al. [8] considered a collection of clusters as a hierarchical system and used a tree topology to do load balancing. Willebeek and Reeves presented a hierarchical balancing method (HBM) [9] which organizes the multi-computer system into a hierarchy of balancing domains.

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7-Issue 1, January 2024
Paper: 69

In particular, the HBM strategy is most efficiently mapped to systems which are based on a tree. The root of the HBM strategy centralizes the global information about the system and can be a bottleneck. PENG Limin et. el. [10] proposed load balance model that organizes the system into a hierarchy of balancing domain. Load balancing decision is based solely on information pertaining to those nodes within each domain. Payli et el.[11,12,] have described a system consisting of clusters and each cluster is represented by a coordinator. Each coordinator first attempts to balance the load in its cluster and if this fails, communicates with the other coordinators to perform transfer or reception of load. This process is repeated periodically. Mohammad I. Daoud et el. proposed a customized genetic algorithm to create high-quality task schedules [17].

In this paper a new scheduling approach in order to load balancing is described in which the network is considered a two level hierarchy. It considers only non-preemptive load transfer. This model improves average turnaround time of the system and provides resource utilization by distributing the workload evenly among the nodes.

III. System model

In this paper a network is considered consisting of N homogeneous nodes P_1, P_2…, P_N connected by a communication network. Each node has some computational facilities and local memory. For any node in the distributed system jobs are assumed to arrive at each node i according to Poisson rate λ. The service time of the jobs are exponentially distributed with mean of $1/\mu$. The jobs are assumed to be independent and can be executed at any node. Each node has a buffer of infinite capacity to store jobs which are waiting for execution. The jobs are assumed to be non-preemptive. Each node maintains its own ready queue and follows FCFS scheduling policy. In the proposed model the network is organized into L clusters. Each cluster has a specific node designated as the Cluster Master (CM), which is responsible for managing the workload of each node in that cluster. CM maintains a table indicating the load status of each node in that cluster. New job is submitted to CM. Intra-cluster load balancing is done by CM upon receiving a job if any under-loaded node is found in its own cluster. On failure inter-cluster balancing is achieved by balancing loads between any one of the other L_{N-1}clusters. Load can be specified in many different ways. One common approach is the total size of the processes waiting in the ready queue and the queue length of the processor. Load factor (LF) of each node in a cluster is calculated by CM. LF depends on the processing capability of the processor, processor queue length and utilization of RAM. If the LF of a node is within the upper threshold and lower threshold of LF then it is considered moderately loaded. A node is lightly-loaded if its LF is lower than the lower threshold value and a node is overloaded if its LF is greater than the upper threshold of LF. This proposed model is semi-distributed and decentralizes the load balancing process. It is scalable as it minimizes communication overhead.

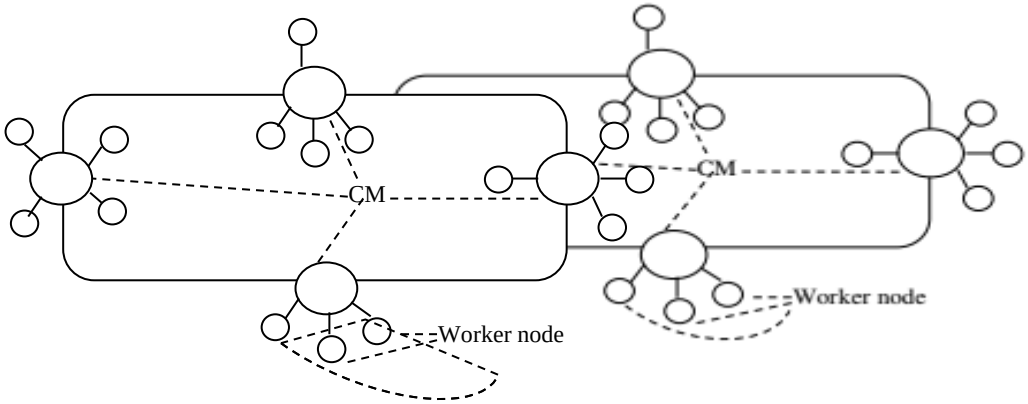


Fig 1. Clustered representation of a distributed network

Each worker node is responsible for:

1. maintaining its own ready queue
2. scheduling jobs of ready queue according to FCFS algorithm
3. responding to CM's query in order to present load information

Each CM is responsible for:

1. estimating workload information of each of the worker node
2. maintaining load table for its cluster
3. deciding when to start intra cluster load balancing
4. deciding to initiate inter cluster load balancing

Each worker node maintains a table containing following parameters:

- pid of job
- execution time of each job

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7-Issue 1, January 2024
Paper: 69

- start time and end time of each job
- arrival time of each job
- status of each job

Each CM maintains a table containing following parameters for each node in its cluster:

- node id
- load
- status

IV 2 Level Task Scheduling policy (2LTS)

Given a set of independent tasks submitted to the system. The algorithm makes an effort to balance the load of the cluster resources by allocating each task to a node such that the expected response time is minimized. The load balancing process starts on the receipt of new load in CM. The CM sends a query to all working node in that cluster to know the current workload of the node. Each node responds to query by sending its processor power, queue length, free memory. From these information the CM calculates the load factor for each node in its cluster and updates load table. Firstly, intra cluster load balancing is tried. The initiating CM searches for a lightly loaded node in its cluster. If such a node is found, the job is sent to that node. On failure, inter cluster load balancing is required. The CM broadcasts a message to all other CMs if they are willing to share load. Any CM upon receiving the message checks its load table for lightly loaded node. If it finds a lightly loaded node sends response containing least LF of that cluster to the initiating CM. Initiating CM selects the node with least LF and the job is transferred to the respective CM. If the total system is overloaded initiating CM does not receive any reply. On time out load is accommodated in the node having least LF in home cluster.

A. Cluster Master algorithm

The cluster master is responsible to monitor local loads, upon receiving new load initiate transfer of load to lightly-loaded node if there are local matches and initiate inter-cluster load balancing on failure of intra-cluster load balancing. Its state diagram is depicted in figure 2.

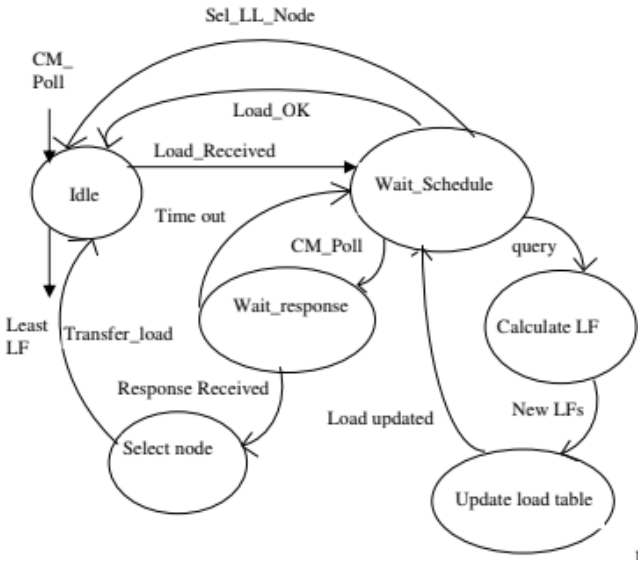


Fig 2. Cluster Master state machine

CM is awakened by Load_Receipt/CM_Poll. Upon receiving Load_Receipt the state is changed to Wait_Schedule. Then, it sends a query message to the worker nodes and the state is Calculate LF until it receives response from all worker nodes. After calculation of LFs of all nodes, the load table is updated. If any lightly loaded node is found state changes to Idle. If no lightly loaded node is found the CM sends CM_Poll message to all other CMs and its state is changed to Wait_response. If more than

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7-Issue 1, January 2024
Paper: 69

one response is received from CMs, the node with least LF is selected and the load is transferred to that node. If no response is received before the time out, the new process is sent to the node with least load in the home cluster. Upon receipt of CM_Poll, it updates load table and sends the least LF to the initiating CM.

Process CM

1. Begin
2. While TRUE
3. Wait for Load_Receipt;
4. Send Load_query to all worker nodes in its cluster;
5. Calculate LF of each worker node
   LF = (1/power)*queue length
6. Update load table
7. If lightly loaded node is found in home cluster
8. Check for free RAM space to accommodate new load
9. If such a node found
10. Send New_Load to selected node;
11. Else
12. Broadcast CM_Poll to all other CMs;
13. If response is received before the time out
14. Select the response with least LF;
15. Send New_Load to selected CM;
16. Else
    Send New_Load to least loaded node in home cluster
17. End

B. Worker Node Alogorithm

Each worker node receives load_query from its CM. In response it sends three tuple {processor power, queue length, free ram space} to the CM.

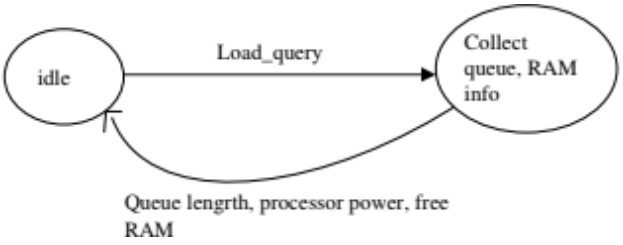
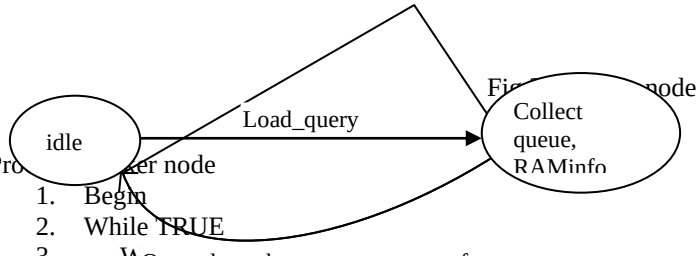


Fig. node state machine



Pro er node

1. Begin
2. While TRUE
3. W
4. C
5. Send queue length, free RAM space, processor power to CM
6. End

V Analysis

Let us assume k, m and d be the upper bounds on the number of clusters, nodes in a cluster in the distributed system, and the diameter of a cluster respectively. A node can communicate to its CM in maximum d steps.

Theorem 1. The total time to assign a task is between 2dT + L and (4d + k)T + L where T is the average message transfer time between adjacent nodes and L is the actual average load transfer time.

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7-Issue 1, January 2024
Paper: 69

Proof: CM sends load query to all nodes of the cluster, and each node sends reply to CM which requires 2d steps. So, total time to load transfer is 2dL. If Intra cluster load balancing is not possible, CM sends poll to k-1 CMs requiring maximum hops k/2, each of them collects load information from each of the nodes of its cluster requires max 2d steps. Then, response to the initiating CM requires maximum k/2 hops to reach. So the messages to load transfer have to pass through (2d + k + 2d) hops resulting in (4d + k)T + L time. If the diameter of cluster decreases this approach produces better result than [12] in inter cluster load balancing.

Theorem 2. The total number of messages to assign a task in order to load balancing is between (2m) to (2km+2(k-1)).

Proof: After the receipt of New_load, a message is to each worker node in the cluster. Each worker node sends response to CM. So total number of messages is 2m. In intra cluster load balancing 2m message passing is required. In inter cluster load balancing k-1 request messages are sent and at most k-1replies can be received. Each CM sends and receives replies from each worker node resulting (k -1)*2m messages. So maximum number of messages is (2m + 2(k-1) + (k -1)*2m). This is much less than required in [13].

VI Simulation and Result

Simulation experiments were conducted to evaluate the performance of the proposed 2LTS policy. The experiments were performed by varying several performance parameters namely the number of worker nodes, power and RAM of each worker node and the number of tasks.

Table 1. Parameter values

Parameters	Values
Number of processors	10 ~ 40
Power of processor	1 ~ 4
Number of tasks	50 ~ 800
Task size	1000 ~ 2500
Job inter arrival time	Exponentially distributed with mean 2

Average turnaround time and average load on each processor are calculated for the system before and after balancing and also for Least Load (LL) and Round Robin (RR) algorithm. The number of jobs is varied between 50 and 800 and the number of processor is varied from 10 and 40. It is assumed that execution time of a task having 1000 unit size takes 20 unit of time on basic processor. It is observed that both parameters have improved after balancing the load of the system using 2LTS.

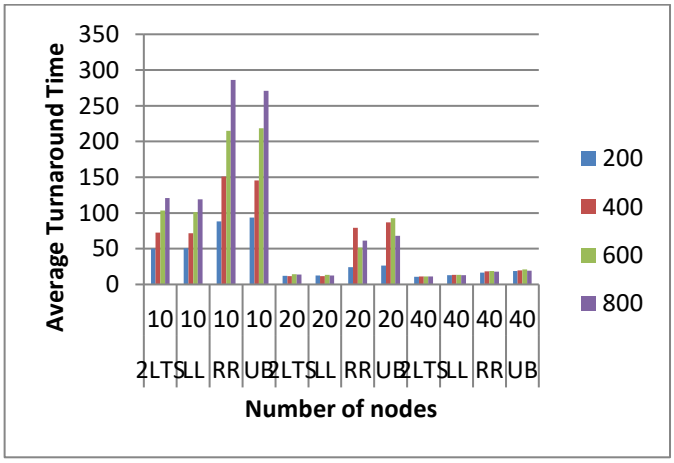


Fig 4. Average TAT for various no. of jobs with various no. of nodes.

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7-Issue 1, January 2024
Paper: 69

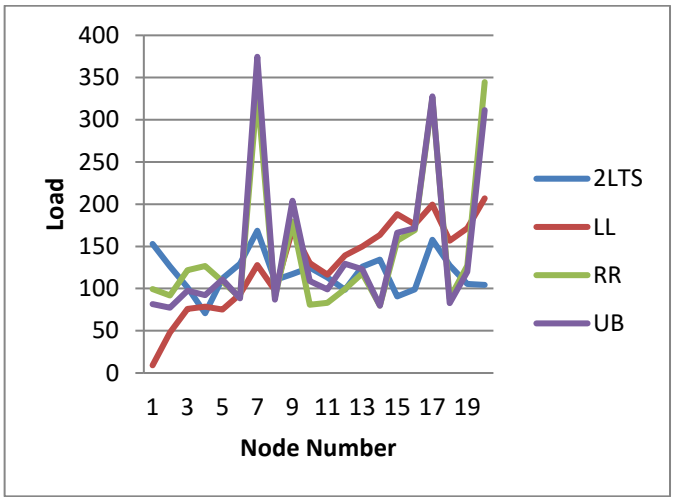


Fig 5. Load of different nodes for average 10 tasks per node

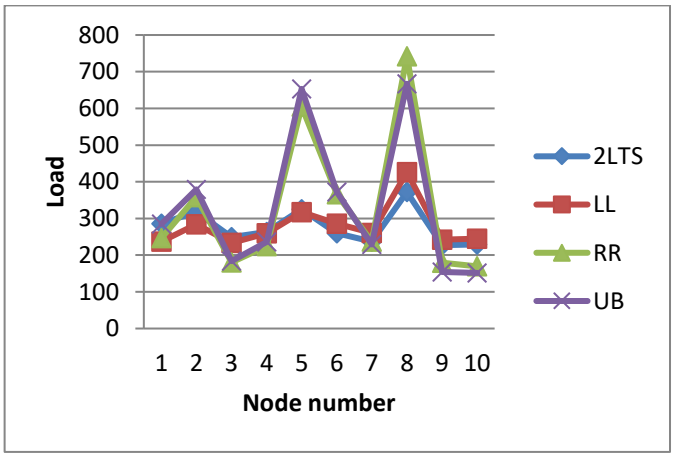


Fig 6. Load of different nodes for average 20 tasks per node

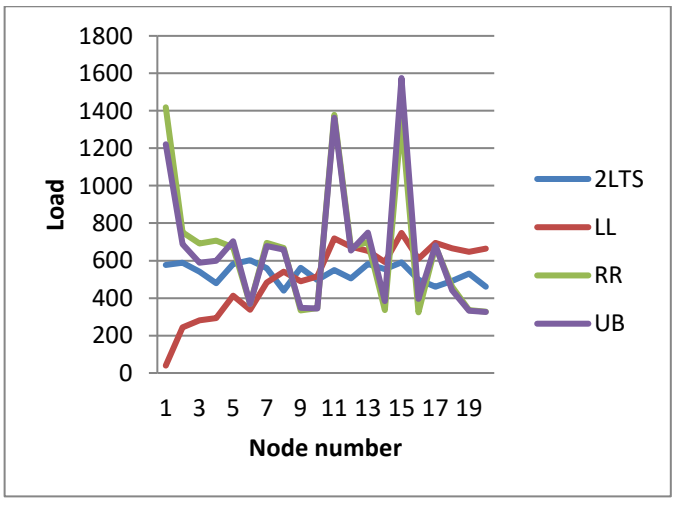


Fig 7. Load of different nodes for average 40 tasks per node

Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)
Volume 7-Issue 1, January 2024
Paper: 69

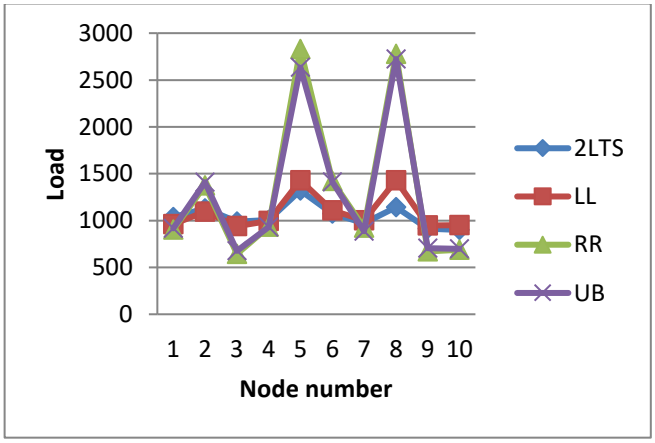


Fig 8. Load of different nodes for average 80 tasks per node

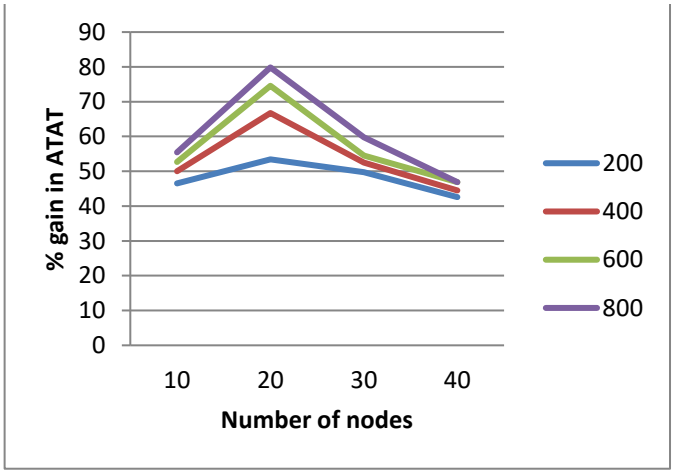


Fig 9. Percentage gain in TAT in respect to unbalanced system

From figure 4, it can be seen that the proposed algorithm reduces Average Turnaround Time in compare to unbalanced system (UB) and the existing algorithms. In LL algorithm for assigning each new task huge number of messages passing is needed, which takes some time and incur cost. That time is not considered while calculating the TAT of each job here. Even though, if is observed that 2LTS's performance is the same, sometimes better than LL. Figure 5, 6, 7 and 8 show the load on each node of the system for varying average workload. It is observed that, while tasks are assigned using 2LTS algorithm, load on each node is varying a little. So, the aim of balancing load for proper resource utilization is assured. From the graphs it can be inferred that if the nodes are heavily loaded, then the performance of the balanced system is much better than unbalanced system. With the reduction in average system load per node, as resources becoming underutilized, average turnaround time of tasks are not improved much. From figure 9 it is observed that gain (%) in average turnaround time varies from 42.59to 79.85. The gain increases with increase in workload. It is also observed that, the gain is less if the nodes are lightly or heavily loaded than moderately loaded. So taking all variations in consideration it is seen that 2LTS improves the overall system performance.

VII Conclusion

In this paper a semi-distributed load balancing method is proposed for heterogeneous distributed system. New task is assigned a processing node considering the present load status of each node. The system is partitioned into a number of clusters and each cluster has a master to perform local load balancing and also to communicate with other cluster masters of the system

to provide inter-cluster load transfers, if needed. The load balancing method is scalable and has low message and time complexities. The method of partitioning the system into clusters and the method of load transfer are not addressed. It is assumed that the tasks have no precedence. Cluster master may fail and due to their important functionality in the proposed model, new masters should be elected. Also, a mechanism to exclude faulty nodes from a cluster and add a recovering or a new node to a cluster is needed. These procedures can be implemented using algorithms as in [14]. As a future work I will focus on jobs with precedence relations and deadline.

References

[1] William Leinberger, George Karypis, Vipin Kumar, and Rupak Biswas, "Load Balancing Across Near-Homogeneous Multi-Resource Servers", 9th Heterogeneous Computing Workshop (HCW 2000*)* 0-7695-0556- 2/00, 2000 IEEE
[2] El- Zoghdy S. F., Kameda H., and Li J. Comparison of dynamic vs. static load balancing policies in a mainframe-personal computer model, INFORMATION, 5(4):431–446, 2002
[3] Kabalan K.Y., Smari W.W. and Hakimian J.Y. "Adaptive load Sharing in heterogeneous system: Policies,Modification and Simulation, CiteSeerx, 2008
[4] Lin H. C. and Raghavendra C.S. "A Dynamic Load Balancing Policy with a Central Job Dispatcher (LBC)".IEEE 1992
[5] Konstantinou, Ioannis; Tsoumakos, Dimitrios; Koziris, Necta/8532.rios, "Fast and Cost-Effective Online Load-Balancing in Distributed Range-Queriable Systems", IEEE Transactions on Parallel and Distributed Systems, Volume: 22, Issue: 8 (2011), Pages 1350-1364
[6] Ahmad I., Ghafoor A. and Mehrota K. "Performance Prediction of Distributed Load balancing on Multicomputer systems", ACM 830-839, 1991
[7] L. Anand D.Ghose, and V.Mani, "ELISA: An Estimated Load Information Scheduling Algorithm for distributed computing systems", International Journal on Computers and Mathematics with Applications, Vol.37, Issue 8, pp. 57-85, April 1999.
[8] Niewpoort, Kielmann,Bal "Efficieent Load Balancing for wide area divide and conquer applications ", proceedings of the eighth ACM SIGPLAN symposium on Principles and Practises of Paralle programming Pg:34-43
[9] Marc H. Willebeek-LeMair, Anthony p. Reeves, "Strategies for dynamic load balancing on highly parallel computers", IEEE Transactions on Parallel and Distributed systems, vol. 4, No. 9, September 1993.
[10] PENG Limin , XIAO Wenjun, "A Binary-Tree based Hierarchical Load Balancing Algorithm in Structured Peer-to-Peer Systems", Journal of Convergence Information Technology, Volume 6, Number 4. April 2011
[11] Resat Umit Payli, Kayhan Erciyes, Orhan Dagdeviren, "Cluster-based load balancing algorithms for grids", International Journal of Computer Networks & Communications (IJCNC) Vol.3, No.5, Sep 2011
[12] Kayhan Erciyes, Resat Umit Payli , "A Cluster-Based Dynamic Load Balancing Middleware Protocol for Grids", Advances in Grid Computing - EGC 2005, LNCS 3470, 805-812, Springer-Verlag, Berlin
[13] Chatterjee, M.; Setua, S.K., "A new clustered load balancing approach for distributed systems," in Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference on , vol., no., pp.1-7, 7-8 Feb. 2015doi: 10.1109/C3IT.2015.7060188
[14] Tunali, T., Erciyes, K., Soysert, Z. (2000) A Hierarchical Fault-Tolerant Ring Protocol For A Distributed Real-Time System. Special issue of Parallel and Distributed Computing Practices on Parallel and Distributed Real-Time Systems, 2(1), 33-44.
[15] Priyesh Kanungo, " Load Measurement Issues in Dynamic Load Balancing in Distributed Computing Environment", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 10, October 2013
[16] M. Andreolini, M. Colajanni and R. Morselli, "Performance Study of Dispatching Algorithms in Multi-tier Web Architectures," ACM Performance Evaluation Review, Vol. 30, No. 22, pp.10-20, Sept. 2002.
[17] Mohammad I. Daoud and Nawwaf Kharma, "An Efficient Genetic Algorithm for Task Scheduling in Heterogeneous Distributed Computing Systems", 2006 IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006