International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 2– No.7, February 2012 – www.ijais.org

Performance Evaluation in Distributed System using Dynamic Load Balancing

Rutuja Jadhav
PG Student
KKWIEER, Nasik

Snehal Kamlapur
Associate professor
KKWIEER, NASIK

I Priyadarshini
PG Student
KKWIEER, Nasik

ABSTRACT
Distributed computing system (DCS) is the collection of heterogeneous and geographically dispersed computing nodes. Nodes co-operatively work to complete the task in the DCS. But because of the dynamic nature of DCS, nodes may fail randomly thus performance is an important factor to be considered. In order to achieve improved performance resource management plays an important role. In this paper dynamic load balancing is focused to achieve better performance results even in case of node failure using regenerative theory.

General Terms
Distributed Computing Systems

Keywords
Distributed Computing System, Reliability, Load balancing

1. INTRODUCTION

1.1 Motivation
Distributed-computing system (DCS) is a collection of heterogeneous and geographically dispersed computing elements (CEs) that allows nodes to cooperatively execute applications in a parallel fashion. Dynamic nature of DCS affects the performance of DCS, as any node in the DCS can fail at any random time. As there are heterogenous nodes in the DCS, their service times vary. Because of which some nodes in the DCS may be overloaded, some may be underloaded or some of them may be even idle. Thus to avoid this type of situation resource management plays important role and helps in achieving better performance.

1.2 Existing Systems
In order to achieve better performance and to avoid above problems, various solutions exist. Some solutions are based on optimal task assignment, some use load balancing and some use load sharing approach.

1.3 Concept
The aim of the paper is to use dynamic decentralized algorithm in order to improve performance in the distributed computing systems. In DCS nodes can fail at any random time, thus assuming backup nodes to all computing nodes and using decentralized dynamic load balancing we will try to improve performance.

2. LITERATURE SURVEY
To date different methods were proposed based on Task assignment, load balancing and load sharing.

2.1 Based on Resource Management
Whenever tasks enter into a DCS, following methods [1] are available in order to assign those tasks to the processor

2.1.1 Task Assignment Approach
In this approach each process submitted is viewed as a collection of related tasks and these tasks are scheduled to suitable nodes so as to improve performance. In this it is assumed that process is already split into tasks, speed of each processor is known, computation by tasks is known, speed of processor is known, processing cost of each task on every node is known, interprocess communication between tasks is known also resource requirements of the tasks and available resource at each node is known. Based on above all information an optimal assignment of tasks is found. But reassignment of the tasks is generally not possible in this approach.

2.1.2 Load Balancing Approach
It is also known as load leveling approach. In this approach all processes submitted by the users are distributed among the nodes of the system so as to equalize the workload among the nodes by transparently transferring workload from heavily loaded nodes to lightly loaded nodes. It can be classified as static versus dynamic algorithms. Static algorithms use only information about the average behavior of the system ignoring the current state of the system. Dynamic algorithms react to the system state that changes dynamically. Static algorithms are further classified as Deterministic versus Probabilistic. Deterministic algorithms use the information about the properties of the nodes and characteristics of the processes to be scheduled to deterministically allocate processes to nodes. Probabilistic algorithms uses information regarding static attributes of the system such as the number of nodes, processing capability of each node, network topology and so on to formulate simple placement rules. Dynamic algorithms can be centralized or distributed. In Centralized algorithms the responsibility of scheduling physically resides on a single node called centralized server node. In distributed algorithms the work involved in making process assignment decision is physically distributed among the various nodes of the system. Distributed algorithms may be further classified as Cooperative and Non-Cooperative. In Non-Cooperative algorithms individual entities act as autonomous entities and make scheduling decisions independently of the actions of other entities. In Cooperative algorithms distributed entities cooperate with each other to make scheduling decisions.

Following issues are to be considered while designing load balancing algorithms

Load Estimation Policy: To estimate workload of a particular node of the system. CPU utilization of the node is the measure used.

Process Transfer policy: Threshold policy is used to decide whether a node is lightly loaded or heavily loaded. It can be static where each node has a predefined threshold value depending on its processing capability. Whereas in dynamic the threshold value of a node is calculated

Location policies: In order to select destination node following policies are used: 1] Threshold: In this a destination node is selected randomly.2] Shortest: In this distinct nodes are chosen at random and each is polled to determine its load and then the process is transferred to the node having minimum load value. 3] Bidding: in this each node plays 2 roles manager and contractor. Manager represents a node having a process in need of a location to execute. Contractor represents a node that is able to accept remote processes. To select a node, manager broadcasts a request for bids message to all nodes then controller returns a bid to manager which contains prices based on processing capability, memory size, resource availability and so on .The manager chooses the best bid. And 4] Pairing : In this 2 nodes that differ greatly in load are temporarily paired with each other and load balancing is carried out by transferring process from overloaded node to under loaded node.

State Information Exchange policies: 1] Periodic broadcast: each node broadcasts its state information only when the state of the node changes because of arrival or departure of a process. 2] On demand Exchange: A node broadcasts a state information request message when its state switches from normal to under loaded or overloaded region. Then the other node sends current state.3] Exchange by Polling: A node searches a suitable partner by randomly polling other nodes one by one.

Priority Assignment Policies: 1] Selfish: Local processes are given higher priority than remote processes. 2] Altruistic: Remote processes are given higher priority than local processes.3] Intermediate priority of processes depends on the number of local processes and number of remote processes at the concerned node.

Migration Limiting Policies: Classified as Controlled and Uncontrolled. In Uncontrolled a process may be migrated any number of times while in Controlled a migration count is used to fix a limit on the number of times a process may migrate.

2.1.3 Load Sharing Approach

It assumes that instead of balancing load on all nodes it is necessary and sufficient to prevent the nodes from being idle or having more than 2 processes. So called load sharing.

In this paper we will focus on Load balancing as the process scheduling technique in order to improve the performance. Load balancing as discussed in above section can be Static or Dynamic. Let us see the comparison of static v/s dynamic load balancing algorithms

Following table gives the comparison between two:

TABLE I Comparative study of stataic and dynamic lb

Sr. No.	Parameters	Static LB	Dynamic LB
1	Workload Assignment	Compile Time	Run Time
2 PERFORMANCE	Response Time	Less	More
	Reliability	Less	More
	Resource Utilization	Less	More
	Overhead	Less	More
	Processor Thrashing	No	Substantial
	Stability	More	Less
	Predictability	More	Less
	Adaptability	Less	More

Thus observing above comparison we can say that even though dynamic load balancing is complicated but can produce better performance results. Thus in the following paper we will focus on dynamic load balancing to achieve better performance.But as discussed in above sections Dynamic load balancing can be Centarlized or Decentralized. Our focus will be on Decentralized, but again it is classified as Cooperative or Non- Cooperative. Our focus will be on Cooperative.

2.2 Deterministic communication delays

H.lee et al. [2] assumes one execution predictor, by which the execution time of the nodes will be known and thus allocate tasks to the nodes with less execution time. J. Palmer et al. [13] Model based on hyperdistribution is constructed and anayzed, which is used to evaluate and optimize multiple server systems subject to breakdowns and repairs.

2.3 Exploiting a priori information on the network configuration

S. srinivasan et al. [3] considers initial network configuration, processor speed, type of interconnection between nodes and other number of parameters that define relationship between them. V. Ravi et al. [12] Task allocation algorithm for heterogenous distributed computer systems is discussed considering hardware configuration is fixed inoder to compensate for delays

2.4 Heuristics algorithms such as genetic algorithms and simulated annealing

Simulated annealing (SA) is a generic probabilistic metaheuristic for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. D. vidyarthi et al. [4] used to find the assignment of modules to processors such that a

measure of performance is optimized, the requirements of each module are met and the capacities of the resources are not violated. D. Vidyarthi et al. [11] Genetic algorithms has emerged as an suceessful tool for optimization.Allocation algorithm based on GA is discussed in order to consider the identification of an appropriate allocation by applying genetic operators crossover and mutation on candidate solutions.

2.5 Using Static load balancing

Dai et al. [5] Load balancing takes place at the instant of arrival of any job rather than waiting for transfer instant. Also it computes the job finish time at the arrival instant rather than waiting for estimating instant.

2.6 Using dynamic load balancing

pezoa et al. [6] considers a probabilistic framework to characterize service reliability by solving dynamic load balancing problem. It uses the concept of stochastic regeneration for achieving reliability.

As discussed earlier we will focus on Load balancing approach. We will be focusing on dynamic load balancing approach as it has many advantages over Static load balancing as seen above.

3. SYSTEM ARCHITECTURE AND WORKFLOW

Consider a set of n nodes in the DCS. Each node will broadcast its queue information. Thus all nodes will have load on all nodes and also the overall system load. Each node then will check whether it is overloaded or underloaded by comparing its load with the systems load. Overloaded nodes will then perform load balancing by identifying the recipient or under loaded nodes to which extra task transfer will take place. In case of any regeneration event or node failure, load balancing will be performed at that instant. Thus achieving better performance.

Assumptions followed by pezoa are also followed in the proposed method and are as follows:-

1] All random variables such as service time ST of task at any node, transfer time TT of QI packet, failure time FT of any random variable, transfer time TFN of failure notice and transfer time of group of tasks TTG are exponentially distributed. We will assume any other random distribution and compare the performance.

2] All random variables are mutually independent

3] Mean transfer time of group of tasks follows first order approximation.

4] There are no external tasks arriving in the DCS after t = 0.

5] Each node is equipped with back up nodes.

Load Balancing Issues discussed in section I are resolved in the proposed system as follows :-

1] Load Estimation Policy: CPU queue length will be considered as the workload of the node.

2] Process Transfer Policy: It is dynamic. The nodes will be computing its extra workload with reference to average load in the system.

3] Location Policy: Overloaded nodes will identify randomly set of under loaded nodes.

4] State information Exchange Policy: It will use periodic broadcast and will broadcast its state information whenever there is any change.Node state describes 1) number of tasks queued at each node, 2) whether a particular node is functioning or down and 3) number of tasks in transit to particular node.

5] Priority Assignment policy: It may use Intermediate priority of processes which depends on the number of local processes and number of remote processes at the concerned node.

6] Migration Limiting Policy: In this case Controlled policy will be used and up to some threshold value migration may take place.

Following figure shows the overall workflow of the system

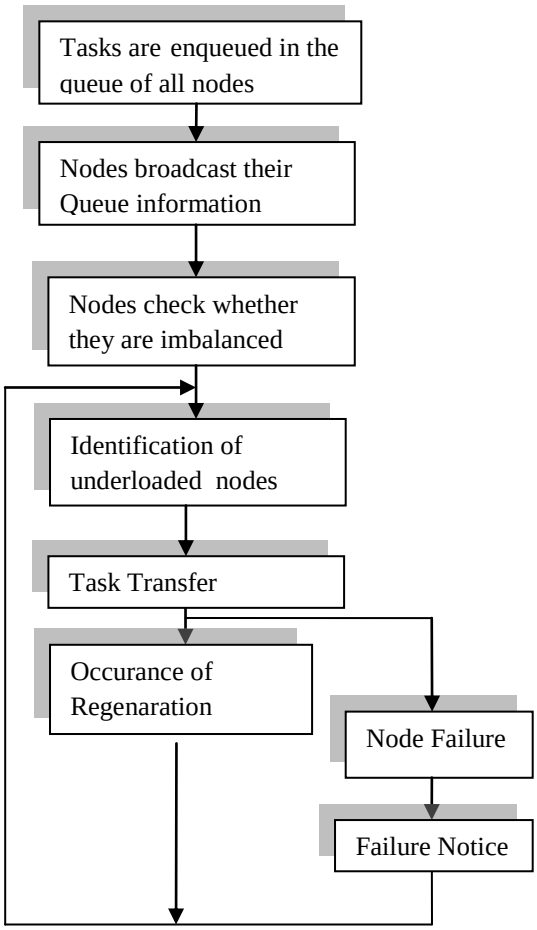


figure I: workfolw of the system

4. DETAILED DESIGN

4.1 System Model

Reallocation Policy:

1] LB will be performed by all the nodes independently and will identify whether it is overloaded or under loaded. Excess load will be computed using following equation:

$$L_j^{ex}(t_b) = Q_j(t_b)\frac{\Lambda_j}{\sum_{l\in w_j}\Lambda_l}M_j(t_b),$$

where $M_j(t_b) = Q_j(t_b) + \sum_{l=1,l\neq j}^{n} Q_{lj}(t_b)$ is the estimate of the workload in the system as perceived by the j th node at time $t = t_{b,} W_j$ is the collection of nodes that are functioning as perceived by the j th node at time $t = t_b$.

2] Each overloaded node will then calculate the tasks to reallocate. For each overloaded node j, let U_j be the collection of candidate task-receiver nodes as all those nodes that, at time t_b, are perceived by the sender node as functioning and under loaded with respect to their own perceived fair shares of the total workload; namely, $U_j = \{K : L_{kj}^{ex}(t_b) < 0.K\varepsilon W_j\}$ where $j\varepsilon v$ and Let $L_{kj}^{ex}(t_b)$ is the excess load at the kth functioning node as perceived by the jth node and is defined as $L_{kj}^{ex}(t_b) = Q_{kj}(t_b) - \Lambda_k M_j(t_b)/\sum_{l\varepsilon wf}\Lambda_l$.

3] Each overloaded node will then partition its excess task among the available under loaded nodes. jth node partitions its excess load among all the candidate task receiver nodes. For the kth candidate task-receiver node, the partition p_{jk} is defined as $p_{jk} = L_{kj}^{ex}(t_b)\sum l\varepsilon uj L_{lj}^{ex}(t_b)$ whenever. $K\varepsilon U_j$ For convenience, the partition $p_{ji} = 0$ for all $i \notin U_j$.

4] In the event of any node failure, back up node will broadcast the Failure Notice and redistribute unserved taks to under loaded nodes.

5] At any regeneration event, LB algorithm will be executed. Regeneration time, is the minimum of the following five random variables: the time to the first task service by any node, the time to the first occurrence of failure at any node, the time to the first arrival of a QI packet at any node, the time to the first arrival of an FN packet at any node, or the time to the first arrival of a group of tasks at any node.

4.2 Mathematical Model

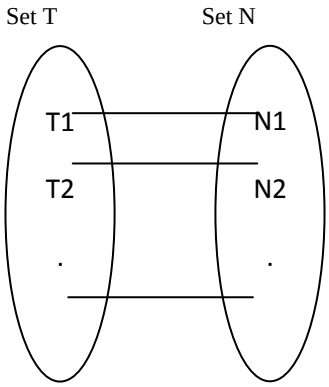
a) Let N = { N1, N2,…Nn } be the set of nodes in the DCS, wherein nodes can be Functioning nodes, Non-Functioning nodes, Back up nodes, Unbalanced nodes, or Balanced nodes.

b) Let T = {T1, T2 ...Tm} be the set of tasks that can be assigned to any nodes in the DCS.

c) And let F = { F1, F2,.....Fk } be the set of various functions performed in the DCS like Broadcasting no. of tasks at each node, Checking for imbalance, Identifying set of recipient nodes, Load balancing to transfer load, Redirecting Load of Failed or non-functioning node etc.

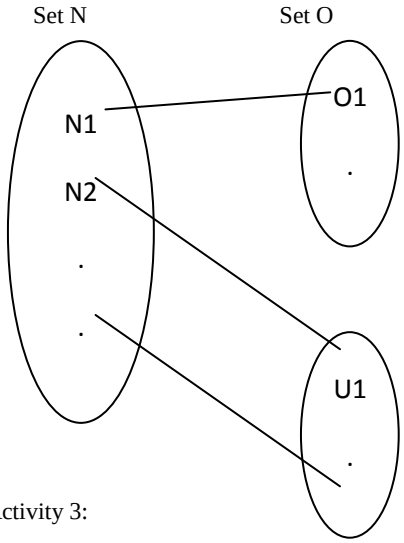
Activity 1:

Task Assignment



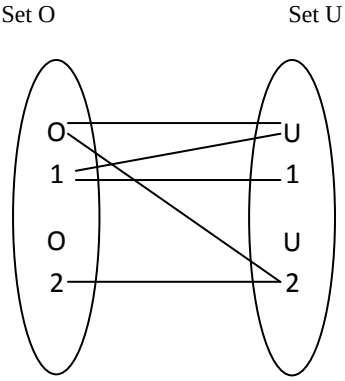
Activity 2:

- After broadcasting queue information all nodes will check whether they are overloaded or under loaded.
- Let set O indicate overloaded nodes and set U indicate under loaded nodes.



Activity 3:

Set of overloaded nodes perform load balancing and find out recipient nodes in order to transfer load.



5. EXPERIMENTAL SETUP

Proposed system will be demonstrated in 2 ways:-

1] NS-2 Simulation: In this demonstration wireless nodes will be considered.

2] On Distributed Environment:

Configuration of real time systems will be:

Nodes may have configuration:-

Intel P IV 3.0 GHz HT Processor

RAM 512 MB, 80 GB SATA HDD

And /or

Intel Core 2 Duo

E7400, 2.8 GHz HT Processor, RAM 2GB

250 GB SATA HDD

6. TEST CASES

Table III Test cases

Test id	Test description	Expected input	Actual output
1.	Tasks n > 2	Number of tasks are assigned to set of n nodes where n > 2.	Proper load balancing among all n nodes and improved performance.
2.	Arrival of External tasks	At some random amount of time some tasks enters DCS	Considering it as one of the regeneration event Load balancing should be performed again.
3	Node failure	At some random time one of the functioning node should go down.	Back up node should redistribute all incomplete tasks to other nodes.
4	Threshold for migration	Migration of jobs should not cross threshold value	Migration of job will be stopped if it has crossed threshold value

5. EXPECTED RESULT DISCUSSION

As per the proposed system it is expected that the overall performance of the system should improve. No nodes should be overloaded for maximum time and no nodes should be idle in the system. Overall Response time of the system should be minimized. All the tasks allocated in the system should be completed before all the nodes in the system fails.

7. CONCLUSION

The proposed method assumes n nodes and each node is associated with back up node. Back up nodes will not serve the tasks but will transfer the unserved tasks to other underloaded nodes. At the balancing instant overloaded nodes are transfering extra load to the underloaded nodes, performing load balancing of the system. Even in case of node failure back up nodes take care of the unserved tasks.Thus we conclude that even in case of node failure or regeneration event above proposed method will help in improving the performance.

8. REFERENCES

[1] P. K. Sinha, "Distributed Operating Systems – Concepts and Design, IEEE Computer Society Press.

[2] H. Lee, S. Chin, J. Lee, D. Lee, K. Chung, S. Jung, and H. Yu, "A Resource Manager for Optimal Resource Selection and Fault Tolerance Service in Grids," Proc. IEEE Int'l Symp. Cluster Computing and the Grid (ISCCG), 2004.

[3] S. Srinivasan and N. Jha, "Safety and Reliability Driven Task Allocation in Distributed Systems," IEEE Trans. Parallel and Distributed Systems, vol. 10, no. 3, pp. 238-251, Mar. 1999.

[4] D. Vidyarthi and A. Tripathi, "Maximizing Reliability of a Distributed Computing System with Task Allocation Using Simple Genetic Algorithm," J. Systems Architecture, vol. 47, pp. 549-554, 2001.

[5] Y.-S. Dai and G. Levitin, "Optimal Resource Allocation for Maximizing Performance and Reliability in Tree-Structured Grid Services," IEEE Trans. Reliability, vol. 56, no. 3, pp. 444-453, Sept.2007.

[6] Jorge Pezoa, Sagar Dhakaal, Majeed Hayat, "Maximising service reliability in distributed computing systems with random node failures : Theory and implementation",IEEE transaction on parallel and distributed systems, vol. 21,no. 10 october 2010

[7] S. Dhakal, B. Paskaleva, M. Hayat, E. Schamiloglu, and C. Abdallah, "Dynamical Discrete-Time Load Balancing in Distributed Systems in the Presence of Time Delays," Proc. IEEE Conf.Decision and Control (CDC), 2003.

[8] Z. Lan, V. Taylor, and G. Bryan, "Dynamic Load Balancing forAdaptive Mesh Refinement Application," Proc. Int'l Conf. ParallelProcessing (ICPP), 2001.

[9] V. Ravi, B. Murty, and J. Reddy, "Nonequilibrium Simulated- Annealing Algorithm Applied to Reliability Optimization ofComplex Systems," IEEE Trans. Reliability,vol. 46, no. 2, pp. 233-239, June 1997.

[10] G. Koole, P. Sparaggis, and D. Towsley, "Minimizing Response Times and Queue Lengths in Systems of Parallel Queues," J. Applied Probability, vol. 36, pp. 1185-1193, 1999.

[11] D. Vidyarthi and A. Tripathi, "Maximizing Reliability of a Distributed Computing System with Task Allocation Using Simple Genetic Algorithm," J. Systems Architecture, vol. 47,pp. 549-554, 2001.

[12] V. Ravi, B. Murty, and J. Reddy, "Nonequilibrium Simulated-Annealing Algorithm Applied to Reliability Optimization of Complex Systems," IEEE Trans. Reliability, vol. 46, no. 2, pp. 233-239, June 1997.

[13] J. Palmer and I. Mitrani, "Empirical and Analytical Evaluation of Systems with Multiple Unreliable

Servers," Proc. Int'l Conf. Dependable Systems and Networks, pp. 517-525, 2006.

[14] R. Shah, B. Veeravalli, and M. Misra, "On the Design of Adaptiveand Decentralized Load Balancing Algorithms with Load Estimation for Computational Grid Environments," IEEE Trans. Parallel and Distributed Systems, vol. 18, no. 12, pp. 1675-1686, Dec. 2007.

[15] L. Tassiulas and A. Ephremides, "Stability Properties of

Constrained Queuing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," IEEE Trans. Automatic Control, vol. 37, no. 12, pp. 1936-1948, Dec. 1992.

[16] M. Neely, E. Modiano, and C. Rohrs, "Dynamic Power Allocation and Routing for Time Varying Wireless Networks," Proc. IEEE INFOCOM, 2003.

[17] G. Koole, P. Sparaggis, and D. Towsley, "Minimizing ResponseTimes and Queue Lengths in Systems of Parallel Queues,"J. Applied Probability, vol. 36, pp. 1185-1193, 1999.

[18] L. Golubchik, J. Lui, and R. Muntz, "Chained Declustering: Load Balancing and Robustness to Skew and Failures," Proc. WorkshopResearch Issues on Data Eng., pp. 88-95, 1992.

[19] A. Brandt and M. Brandt, "On a Two-Queue Priority System withImpatience and Its Application to a Call Center," Methodology and Computing in Applied Probability, vol. 1, pp. 191-210, 1999.

[20] M. Hayat, S. Dhakal, C. Abdallah, J. Birdwell, and J. Chiasson, "Advances in Time Delay Systems" Dynamic Time Delay Models for Load Balancing. Part II: Stochastic Analysis of the Effect of Delay Uncertainty, pp. 355-368, Springer-Verlag, 2004.

[21] S. Dhakal, M. Hayat, J. Pezoa, C. Yang, and D. Bader, "Dynamic Load Balancing in Distributed Systems in the Presence of Delays: A Regeneration-Theory Approach," IEEE Trans. Parallel and Distributed Systems, vol. 18, no. 4, pp. 485-497, Apr. 2007.

[22] S. Dhakal, M. Hayat, J. Pezoa, C. Abdallah, J. Birdwell, and J. Chiasson, "Load Balancing in the Presence of Random Node Failure and Recovery," Proc. IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS), 2006.