A Review of Load Balancing Strategies for Distributed Systems

Shubhinder Kaur, Gurpreet Kaur
CSE Department, Chandigarh
Gharuan, Punjab, India

ABSTRACT
The main concern of DCS is reliability. For improving the reliability one way is proper task allocation among the nodes and load balancing is one of the best ways to improve not only reliability but throughput and minimizes execution time. In this paper, we reviewed the concept of load balancing and various mechanisms and finally compared static and dynamic load balancing. The main purpose of this paper is to help in designing new algorithm in future by studying existing mechanisms for load balancing.

Keywords
Distributed System, Load balancing, Processor thrashing.

1. INTRODUCTION
The explosion of Internet has increased the amount of the online information and services available for the users and this growing information and services has placed distress on the Internet infrastructure. A distributed computing system is an aggregate of heterogeneous and geographically dispersed computing elements that cooperatively execute applications in a parallel fashion [1]. The performance of DCS depends on the allocation of tasks on the processor. If allocation of tasks is not done carefully the processors of the system spend more time to transfer the tasks rather than performing computation which is known as thrashing [2]. So to increase the performance and assigning & reassigning tasks efficiently a task allocation model is used. Load sharing- a process of redistribution of work load among multiple servers in DCS- have been extensively used to improve performance. Load sharing has also been referred to as load balancing [3]. The main goal behind distributing load is to distribute process so as to maximize throughput, maintain stability, minimize resource utilization, minimize computational time, minimize communication delays and fault tolerance.

Any load balancing algorithm balances the load by transferring it from heavy loaded node to the lightly loaded node so as to minimize execution time. Load balancing makes every processor equally busy and make them finish the work approximately at the same time.

2. RELATED WORK
For our study, we reviewed various papers on load sharing in distributed systems. Jagdish Chandra Patni, Dr. M.S. Aswal, Om Prakash Pal, Ashish Gupta [6] presented various load balancing strategies that control load of the entire system. A load balancing algorithm tries to improve response time by utilizing available resources. First, a dynamic tree-based model was proposed which represented Grid architecture and then a hierarchical load balancing strategy was developed. The basic idea was to transform any grid architecture into a tree and then to balance load at various levels of tree.

Mayuri A. Mehta, Devesh C. Jinwala [5] analyzed and explored different components for designing load balancing algorithm and proposed new information and location policies. A dynamic load balancing algorithm generally consists of four components namely: information policy, transfer policy, selection policy and location policy through which actions of load balancing algorithm can be governed. The paper proposed a Modified Demand Driven information policy (MDDIP) and Limited Broadcast location policy (LBLP).

Fouzi Semchedine, Louiza Bouallouche-Medjkoune, Djamil Aissani [4] conferred that in order to minimize the response time and improve the performance of the distributed systems the task assignment policy focuses on assigning tasks in an efficient manner. The paper presented and discussed two classes of task assignment policies: policies which assume task size is known a priori and policies which assume task size is not known a priori. Policies assuming a priori knowledge of task size can further be categorized as static or dynamic.

Yung-Terng Wang, Robert J.T. Morris [3] mentioned that the choice of load sharing strategy is one of the important part of distributed system design. The paper categorized load sharing algorithm into source initiative and server initiative and discussed various algorithms and evaluated their performance by using a metric called Q-factor.

Vinod Kumar Yadav, Mahendra Pratap, Dharamendra Kumar Yadav [2] solved the problem of maximizing reliability of DCS. For improving reliability they first determined candidate nodes and then utilized load sharing policies for handling node failure.

Jorge E. Pezoa, Sagar Dhakal, Majeed M. Hayat [1] suggested an analytical and probabilistic framework to devise decentralized load balancing policies that maximize the reliability of distributed computing systems in presence of any communication and node uncertainty.

3. LOAD BALANCING CONCEPT
To process any application, DCS, divide them into set of tasks then these tasks are given to different processors of DCS. The method of assigning tasks to the processors is known as task allocation [2]. Now, we will discuss load balancing/sharing- one of the task assignment methods for DCS, which is most researched area because of wide use of internet. Load balancing means that the work load is evenly distributed across the resources of the system that leads to performance

benefits which can be realized in terms of increased resource utilization, increased throughput, and reduced response time.

A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) and reliability of applications [9]. Clients request for services that terminate at load balancer, which in turn forwards request to servers. The incoming traffic is distributed on network level using load balancing algorithm and load balancing algorithms uses network parameters of incoming traffic to make decisions where to forward traffic. The concept of load sharing/balancing is applied under following situations:

- When a processor or link fails
- When a processor becomes idle
- When a processor becomes overloaded

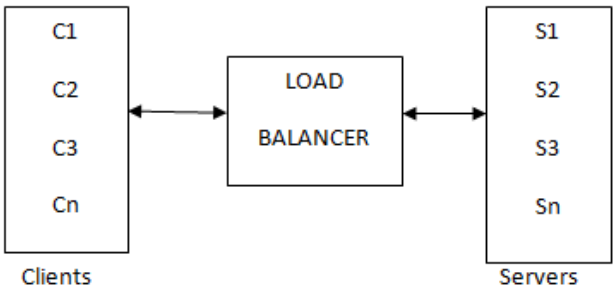

Figure 1: Load balancing concept

Following issues need to be taken into consideration while designing a load sharing algorithm: load estimation policy, process transfer policy, state information policy, location policy, priority assignment policy and migration limiting policy [2][5][6]. Various algorithms are used to distribute load among the servers. Load balancing/sharing techniques are broadly classified into two types: static load balancing and dynamic load balancing.

A. Static Load Balancing:

Static load balancing is based on the concept of master and slave. The performance of processor/server nodes is determined at the beginning of execution and on the basis of that performance workload is assigned by master node. The slave nodes evaluate their work and submit result to master node. The tasks are assigned to slave nodes at compile time, i.e., no change or reassignment is possible at runtime. There is little overhead involved when using static load balancing as there is no migration of jobs at runtime. The tasks are assigned randomly, in a cyclic fashion or based on size range [4].

Random Allocation: Tasks are assigned randomly to the servers. This method uses random numbers to select processors which are generated on the basis of some statistical distribution. Among N servers, server i receives the task with probability 1/N.

Round Robin Allocation: The tasks to the server are allocated in a cyclic fashion. Once the server is assigned a task it is moved to the end of the list so that load is equally shared. Random allocation and round robin policies are simple to implement but their performance deteriorate when the task size distribution is heavy tailed [4].

Threshold Based Allocation: The processes are allocated to the server nodes as soon as they are created. Each server node has a private copy of system's current load. The current load of server node could be under loaded, medium loaded and over loaded. Threshold parameters describe these levels:

Load < t_under UNDERLOADED

t_under <= Load <= t_upper MEDIUM

Load > t_upper OVERLOADED

Central Manager Allocation: The central processor/node selects the host for new process which has least node using the server's current system load information. All the remote server nodes update the load information by sending a message every time when their load changes.

*Size Based Policies [*4]: In these policies, size ranges are affected to the servers. The dispatcher assigns the tasks on the basis of size. *SITA-E* (Size Interval Task Assignment with equal load): The main idea behind this approach is that workload assigned to each cluster will be equal. SITA-E performs poorly when variability increases. *SITA-V* (Size Interval Task Assignment with variable load): When the tasks arrive, the dispatcher assigns all the small tasks to the server that is under loaded and the large tasks to the server that is over loaded. The main idea is to reduce the mean slowdown. *SITA-U* (Size Interval Task Assignment with unbalanced load): This policy uses the concept of cut-off to assign tasks to the server. Each server has a cut-off range and accepts only those tasks that fill in its cut-off.

B. Dynamic load balancing:

Unlike static load balancing policies, the dynamic policies attempt to balance load based on some load information at the servers [4]. Job/task assignment is done at run time. It results in better decision making as compared to static load balancing but load balancer has to monitor the current load on all nodes continuously, it becomes an extra overhead as monitoring consumes CPU cycles. Moreover, dynamic schemes spend more time in migration of jobs than executing any useful work [8].

Central queue algorithm: All new activities and unfulfilled requests are stored as cyclic FIFO queue on the main host. Each new activity which arrives is inserted into the queue. When the request for the activity is received it is removed from the queue.

Local queue algorithm: The basic idea behind this allocation is static allocation of new processes with process migration initiated by host when its load falls under threshold limit which is user defined parameter that defines the minimal number of ready processes the load manager attempts to provide on each processor.

Least Loaded First (LLF): Tasks are assigned to the server on the basis of its load. Examples of least loaded first mechanism are Shortest queue and Least work remaining. In *shortest queue*, based on the queue's contents the dispatcher assigns the incoming tasks so that the server that has least number of tasks in the queue will be assigned the next task for processing. *Least work remaining* uses the remaining work at the server to dispatch the tasks and then selects the server that has least of the remaining work in terms of task size.

Cycle stealing with central queue: This is a variant of the central queue policy. The servers are divided into two different groups: the first one is the group of short servers which processes the small tasks and the second is the group of long servers to process the large tasks. When one of the short servers becomes free, it picks the small task in the queue for processing. Similarly if one of the long servers becomes free it processes large task. If there is no large task long server processes small tasks and when there is no small task small server processes long tasks. Now we will compare static load balancing and dynamic load balancing considering few parameters. The comparison is listed in Table 1.

4. CONCLUSION

The task assignment policies whether static or dynamic are totally dependent upon situations in which workload is assigned, during compile time or runtime. The more efficient a load balancing algorithm better is the performance of computing system. We studied the concept of load balancing and discussed some of the static and dynamic load balancing mechanisms. There exists no perfect load balancing strategy every mechanism has some flaws but it can be used depending on one's need and requirements.

Table 1 Comparison of Load Balancing Techniques

PARAMETERS	**STATIC**	**DYNAMIC**
Nature	Work load assigned at compile time	Work load assigned at run time
Reliability	Less	More
Communication overhead	Less	More
Resource Utilization	Less	Greater
Predictability	Easy to predict	Difficult to predict
Adaptability	Less adaptive	More adaptive
Stability	More	Less
Complexity	Less	More
Processor thrashing	No	Considerable thrashing

5. REFERENCES

[1] Jorge E. Pezoa, Sagar Dhakal, Majeed M.Hayat, Decentralized Load balancing for improving Reliability in Heterogeneous Distributed systems, International conference on parallel processing Workshops, pp. 214-221, Vienna, Austria, 2009.

[2] Vinod Kumar Yadav, Mahendra Pratap Yadav, Dharmendra Kumar Yadav, Relaible Task Allocation in heterogeneous Distributed System with random node Failure: Load Sharing approach, International conference on Computing Sciences, pp. 187-192, India, 2012.

[3] Yung-Terng Wang, Robert J.T. Morris, Load sharing in Distributed Sytems, IEEE trans. on computers, vol. c-34, No. 3, pp. 204-217, 1985.

[4] Fouzi Semchedine, Louiza Bouallouche-Medjkoune, Djamil Aissani, Task Assignment policies in distributed server systems: A survey, Journal of network and Computer Applications, pp. 1123-1129, 2011.

[5] Mayuri A. Mehta, Devesh C. Jinwala, Analysis of significant components for designing effective dynamic load balancing algorithm in distributed systems, International Conference on Intelligent systems modelling and simulation, pp. 531-536, Kota Kinabalu, Malaysia, 2012.

[6] Jagdish Chandra Patni, Dr. M.S. Aswal, Om Prakash Pal, Ashish Gupta, Load balancing strategies for Grid computing, IEEE, pp.239-243, 2011.

[7] Shilpa Gambhir, Er. Sonia Goyal, Reliable task allocation in distributed mobile computing system with random node movement: Replication and Load sharing Approach, IJARECE, vol. 3, pp. 659-663, 2014.

[8] Md. Firoj Ali, Rafiqul Zaman Khan, The study on Load Balancing strategies in distributed computing system, IJCSES, vol. 3, pp. 19-30, 2012.

[9] Load Balancer, https://f5.com/glossary/load-balancer.