ISSN No. 0976-5697

Volume 3, No. 1, Jan-Feb 2012

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

A Survey- Review on Load balancing Algorithms

Nitin R. Chopde*
Department of Computer Science & Engg and IT
Sipana College of Engg & Technology
Amravati, India
nitin.rchopde@gmail.com

Prof. Pritish A. Tijare
Department of Computer Science & Engg and IT
Sipna College of Engg & Technology
Amravati, India
pritishtijare@rediffmail.com

Abstract: Load balancing in distributed computer systems is the process of redistributing the work load among processors in the system to improve system performance. Trying to accomplish this, however, is not an easy task. In recent research and literature, various approaches have been proposed to achieve this goal. Due to the outstanding progress in computer technology and an ever-rising demand for high-speed processing able to support the distributed mode there is an increasing trend towards the use of parallel and distributed systems. In addition, one of the important stages of any system utilizing parallel computing is the load balancing stage in which the balance of workload among all of the system's processors is aimed. In this paper various techniques of load balancing algorithm are discussed, analyzed and compared. Also new load balancing algorithm which has new capabilities to provide optimum load balancing in system is discussed.

Keywords: Parallel computing; load balancing; processors;

I. INTRODUCTION

Load balancing is an important and pervasive problem in all distributed systems. Network Load Balancing provides scalability and high availability. Load balancing, the process of distributing the work fairly among participating processors, is a sub-problem of a bigger dilemma: distributed scheduling. Distributed scheduling is composed of two parts: local scheduling, which takes care of assigning processing resources to jobs within one node, and global scheduling, which determines which jobs are processed by which processor. Load balancing is a vital ingredient in any acceptable global scheduling policy.

The aim of load balancing is to improve system performance by preventing some processors from being overwhelmed with work while others find no work to do. Network Load Balancing servers (also called *hosts*) provide key benefits, including:

a. ***Scalability-*** Network Load Balancing scales the performance of a server-based program, such as a Web server, by distributing its client requests across multiple servers within the cluster.

b. ***High availability-***Network Load Balancing provides high availability by among the remaining servers within ten seconds, while providing users with continuous service.

The presence of a number of processors in these kinds of systems, shows, from one perspective, the necessity of the uniform distribution of workload among these processors, since, studies have shown that in these systems the probability of a processor being idle in the system and other processors having a queue of tasks at hand is very high.

The issue can, in fact, be thus presented that the use of parallel and distributed systems, due to the speed which they add to the processing tasks, is an important factor, but the capital needed to elevate systems to the parallel system type seems logical only on condition that the workload of the system be distributed suitably among the processors. Such an aim becomes practical in parallel and distributed systems by the implementation of a certain type of algorithm called "load balancing algorithm".

The purpose of this algorithm is that a processor may neither be idle nor overloaded but have both idle and overloaded processors neighboring it and can therefore serve to relate them. In other words the relater processor can send the message that it is not idle itself but does have an idle neighbor processor to its neighboring processor.

Load balancing is a technique used to distribute load on server to increase performance and speed of work designated to particular server. There are various techniques that can be used to accomplish load balancing task with the help of different types of computer hardware and software components. Application of load balancing is used by different companies that conduct huge business transactions using internet.

Even companies who are maintaining large number of computer network for single user use load balancing to ensure that each computer is performing well and has the correct amount of power to be able to perform functions that are intended for particular work. Benefit of load balancing is, if one of your server goes off then you have other the option of alternate server to work upon.

Load balancing improves the performance of server due to distributed load and is used for busy and large networks .Without balancing load in busy networks it is very difficult to satisfy the entire request issued to server. Organization into the web services or online business normally makes use of load balancing technique and engages two web servers (or more servers accordingly) .

If one of the web server gets overloaded or goes off, in that case alternate server activates and access the requested load. Load balancing is done by assigning particular service time for each process in order to ensure that several requests are handled without causing traffic. In other words, specific time is assigned to each process in the server for its execution and the process no more stay in the server once service time extends. Once load balancer works actively, service time reduces for each process reduces.

Dynamic load balancing strategies for minimizing the execution time of single applications running in parallel on multicomputer systems are discussed. Dynamic load balancing (DLB) is essential for the efficient use of highly parallel systems when solving non-uniform problems with unpredictable load estimates.

II. LITRATURE REVIEW AND RELATED WORK

Load balancing algorithms are divided in to two major groups: static and dynamic load balancing algorithms. In the former type, based on an estimation of the time needed to complete any given task, tasks are assigned to processors during the compile time and their relation is determined and there is no decision at this type regarding a shift of task from one processor to another during the execution time [1]. Dynamic balancing schemes are able to identify load changes, but they can be misled by differences in resource capacity and dynamic load changes. But in dynamic load balancing algorithms (DLB), the load status at any given moment is used to decide on task shifts between processors [1].Also, load balancing algorithms can be compared with respect to parameters of quality of which nature and overhead-associated can be mentioned [2]. A mathematical model of load balancing was improved and an adaptive load balancing optimization scheduling based on genetic algorithm was proposed, analyzed and simulated. Empirical results show that the algorithm can reduce effectively the average execution time of all requests and speed up the average response time. Meanwhile, with the increment of the cluster size, the algorithm running time is not increased significantly while maintain good performance [2].

Principally, processors in parallel and distributed systems in relation to load balancing algorithms are divided into three groups based the workload level:

a. Processors which have a large number of tasks in waiting to be done called heavily loaded processors or sometimes overloaded processors.
b. Processors which have a small number of tasks in waiting to be done referred to as lightly loaded processors and also under loaded processors.
c. Processors named idle processors which have no tasks to be done [1].

An algorithm has been proposed for load balancing in dynamic, heterogeneous peer-to-peer systems. This algorithm may be applied to balance one of several different types of resources, including storage, bandwidth, and processor cycles. It make possible to achieve the load balancing without having the need to maintain tables for the process migration at each node as a result lot of processing efforts are saved[3]. The algorithm is designed to handle heterogeneity in the form of (1) varying object loads and (2) varying node capacity, and it can handle dynamism in the form of (1) continuous insertion and deletion of objects, (2) skewed object arrival patterns, and (3) continuous arrival and departure of nodes[4]. A mathematical model of load balancing was improved and an adaptive load balancing optimization scheduling based on genetic algorithm was proposed, analyzed and simulated. Empirical results show that the algorithm can reduce effectively the average execution time of all requests and speed up the average response time. Meanwhile, with the increment of the cluster size, the algorithm running time is not increased significantly while maintain good performance [5].

Basically, the proposed solutions aim to speed up simulation processing pace by minimizing communication delays or improving utilization of resources. However, these schemes present drawbacks that motivated the design of a self-adaptive balancing scheme. Some balancing schemes observe look-ahead or the communication rate in simulations to decrease their communication delays. The dynamic approach looks at the load balancing problem more realistically by assuming little information is available before any assignment is made. It does not presume any knowledge of where a certain task will finally execute or in what environment [6]. The analysis of look-ahead evidences the simulation dependencies that might be slowing down the simulation. The study of communication rate indicates the delays they introduce in the simulation, which can be performed statically. The overheads normally incurred from implementing any load balancing policy are always subject to strategies aiming at reducing such overhead, also to reduce the overheads stem from the communication problem and location problem [7].

The growing autonomy of servers may significantly deteriorate the performance of traditional load-balancing strategies. Indeed, the authoritative decision belongs to the load-balancer, but the autonomous servers may reject the requests on their own convenience. The load-balancing strategy for transferring authority from the load-balancer to the autonomous servers [8].The use of a central scheduler was also effective as it can handle all load-balancing decisions with minimal inter-processor communication. The threshold policy provided better performance in comparison to the first fit algorithm that does not have such a mechanism [9].

The idea of previous load balancing algorithm stemmed from the perspective that a processor may neither be idle nor overloaded but have both idle and overloaded processors neighboring it and can therefore serve to relate them; in other words the relater processor can send the message that it is not idle itself but does have an idle neighbor processor to its neighboring processors [1]. This algorithm working with defined parameter to implement load balancing technique.

Load balancing is to ensure that every processor in the system does approximately the same amount of work at any point of time. Processes may migrate from one node to another even in the middle of execution to ensure equal workload. Algorithms for load balancing have to rely on the assumption that the on hand information at each node is accurate to prevent processes from being continuously circulated about the system without any progress. Load balancing is one of prerequisites to utilize the full resources of parallel and distributed systems. Load balancing may be centralized in a single processor or distributed among all the processing elements that participate in the load balancing process. Several tasks are scheduled for separate processors, based on the current load on each CPU. Many researchers have been carried out on load balancing for many years with the aim is to find the load balancing schemes with overhead as low as possible [10].

In earlier researchers had examined methods for load balancing in pipelined term-distributed architectures, and propose a suite of techniques for reducing net querying costs. In particular, they explored the load distribution behavior that pipelining displays and show that the imbalances can be addressed by techniques that include predictive index list assignments to nodes, and selective index list replication [11].The algorithm has been proposed for load balancing strategy always converges, and tends to be in a steady state in a negligible processing time. This include, the load status and the locations of the nodes regarding the system's topology are irrelevant to load balancing process [12].

III. STATIC LOAD BALANCING

In this technique the performance of the processors is determined at the beginning of execution. Then depending upon their performance the work load is distributed in the start by the master processor. The slave processors calculate their allocated work and submit their result to the master. A task is always executed on the processor to which it is assigned that is static load balancing methods are non-preemptive.

A. Round Robin and Randomized Algorithms:

In the round robin [13] processes are divided evenly between all processors. Each new process is assigned to new processor in round robin order. The process allocation order is maintained on each processor locally independent of allocations from remote processors. With equal workload round robin algorithm is expected to work well. Round Robin and Randomized schemes [14] work well with number of processes larger than number of processors. Advantage of Round Robin algorithm is that it does not require inter-process communication. Round Robin and Randomized algorithm both can attain the best performance among all load balancing algorithms for particular special purpose applications.

B. Central Manager Algorithm:

In this algorithm load manager selects hosts for new processes so that the processor load confirms to same level as much as possible. On other hand information on the system load state central load manager makes the load balancing judgment. This information is updated by remote processors, which send a message each time the load on them changes. This information can depend on waiting of parent's process of completion of its children's process, end of parallel execution. The load manager makes load balancing decisions based on the system load information, allowing the best decision when of the process created. High degree of inter-process communication could make the bottleneck state. This algorithm is expected to perform better than the parallel applications, especially when dynamic activities are created by different hosts.

C. Threshold Algorithm:

According to this algorithm, the processes are assigned immediately upon creation to hosts. Hosts for new processes are selected locally without sending remote messages. Each processor keeps a private copy of the system's load. The load of a processor can characterize by one of the three levels: underloaded, medium and overloaded. Two threshold parameters *tunder* and *tupper* can be used to describe these levels.

Under loaded - load < tunder

Medium - tunder ≤ load ≤ tupper

Overloaded - load > tupper

Initially, all the processors are considered to be under loaded. When the load state of a processor exceeds a load level limit, then it sends messages regarding the new load state to all remote processors, regularly updating them as to the actual load state of the entire system.

If the local state is not overloaded then the process is allocated locally. Otherwise, a remote under loaded processor is selected, and if no such host exists, the process is also allocated locally. Thresholds algorithm have low inter process communication and a large number of local process allocations. The later decreases the overhead of remote process allocations and the overhead of remote memory accesses, which leads to improvement in performance. A disadvantage of the algorithm is that all processes are allocated locally when all remote processors are overloaded. A load on one overloaded processor can be much higher than on other overloaded processors, causing significant disturbance in load balancing, and increasing the execution time of an application.

IV. DYNAMIC LOAD BALANCING

It differs from static algorithms in that the work load is distributed among the processors at runtime. The master assigns new processes to the slaves based on the new information collected [15] [16]. Unlike static algorithms, dynamic algorithms allocate processes dynamically when one of the processors becomes under loaded. Instead, they are buffered in the queue on the main host and allocated dynamically upon requests from remote hosts.

A. Central Queue Algorithm:

Central Queue Algorithm [17] works on the principle of dynamic distribution. It stores new activities and unfulfilled requests as a cyclic FIFO queue on the main host. Each new activity arriving at the queue manager is inserted into the queue. Then, whenever a request for an activity is received by the queue manager, it removes the first activity from the queue and sends it to the requester. If there are no ready activities in the queue, the request is buffered, until a new activity is available. If a new activity arrives at the queue manager while there are unanswered requests in the queue, the first such request is removed from the queue and the new activity is assigned to it.

When a processor load falls under the threshold, the local load manager sends a request for a new activity to the central load manager. The central load manager answers the request immediately if a ready activity is found in the *process-request queue*, or queues the request until a new activity arrives.

B. Local Queue Algorithm:

Main feature of this algorithm [17] is dynamic process migration support. The basic idea of the local queue algorithm is static allocation of all new processes with process migration initiated by a host when its load falls under threshold limit, is a user-defined parameter of the algorithm. The parameter defines the minimal number of ready processes the load manager attempts to provide on each processor. Initially, new processes created on the *main* host are allocated on all under loaded hosts. The number of parallel activities created by the first parallel construct on the main host is usually sufficient for allocation on all remote hosts. From then on, all the processes created on the main host and all other hosts are allocated locally. When the host gets under loaded, the local load manager attempts to get several processes from remote hosts. It randomly sends requests with the number of local ready processes to remote load managers. When a load manager receives such a request, it compares the local number of ready processes with the received number. If the former is greater than the latter, then some of the running processes are transferred to the requester and an affirmative confirmation with the number of processes transferred is returned.

The overheads stem from message passing and the scalability issues are among the main objectives of any load balancing system. To this aim, two methods of grouping the nodes were introduced, devised and tested. The first is to group the nodes in couples while the second one is to group the nodes into triples. The numerical results show that the overhead stem from computations is reduced dramatically in both methods. On the other hand, the number of messages is not any more an important issue, since it turns to be fixed with small number of messages as well as the utilization of the system is maximized. The proposed policies guaranteed the distributed system to be scalable.

The earlier study has been try to solve problem of load balancing in peer to peer networking systems. Based on the LBVR algorithm we have extended the features of the RRAVR AND QCHAVR algorithms and they have been proposed one algorithm for Dynamic Load Balancing among the peers, hence the computation over heads are small [20]. Dynamic Load Balancing scalability determines how its performance improves as hosts are added to the cluster. Scalable performance requires that CPU overhead and latency not grow faster than the number of hosts. If the number of servers and operations of a scheduler increases, eventually overhead on the proxy to manage the request also increases. So, in future also need to maintain more number of proxies in order to perform more number of tasks. In our paper we are not primarily concerned with the security issues.

In current research proposed a fault-tolerant and reliable load balancing CBAE algorithm that works by assigning one node as a coordinator and another node as a backup. They compare the CBAE with a well known random election algorithm. Results show that all nodes are balanced by using CBAE, whereas by using a random approach they are obviously imbalanced. So we conclude that CBAE produces a balanced load among nodes as well as low freezing time. The random approach is better in relation to the number of communication messages needed [18]. Some attributes cannot be measured by simulation, e.g. the network traffic and the actual freezing time that is affected by the network delay. As a future work in this context, therefore, we plan to verify our simulation by experiments in a real environment.

V. PARAMETERS AFFECT ON LOAD BALANCING

The performance of various load balancing algorithms is measured by the following parameters.

A. Performance of LBA:

If Load Balancing is not possible additional overload rejection measures are needed. When the overload situation ends then first the overload rejection measures are stopped. After a short guard period Load Balancing is also closed down.

B. Fault Tolerant capacity of LBA:

This parameter gives that algorithm is able to tolerate tortuous faults or not. It enables an algorithm to continue operating properly in the event of some failure. If the performance of algorithm decreases, the decrease is proportional to the seriousness of the failure, even a small failure can cause total failure in load balancing.

C. Forecasting Accuracy in LBA:

Forecasting is the degree of conformity of calculated results to its actual value that will be generated after execution. The static algorithms provide more accuracy than of dynamic algorithms as in former most assumptions are made during compile time and in later this is done during execution.

D. Stability:

Stability can be characterized in terms of the delays in the transfer of information between processors and the gains in the load balancing algorithm by obtaining faster performance by a specified amount of time.

E. Centralized or Decentralized

Centralized schemes store global information at a designated node. All sender or receiver nodes access the designated node to calculate the amount of load-transfers and also to check that tasks are to be sent to or received from. In a distributed load balancing, every node executes balancing separately. The idle nodes can obtain load during runtime from a shared global queue of processes.

F. Nature of Load Balancing Algorithms:

Static load balancing assigns load to nodes probabilistically or deterministically without consideration of runtime events. It is generally impossible to make predictions of arrival times of loads and processing times required for future loads. On the other hand, in a dynamic load balancing the load distribution is made during run-time based on current processing rates and network condition. A DLB policy can use either local or global information.

G. Cooperative behavior:

This parameter gives that whether processors share information between them in making the process allocation decision other are not during execution. What this parameter defines is the extent of independence that each processor has in concluding that how should it can use its own resources. In the cooperative situation all processors have the accountability to carry out its own portion of the scheduling task, but all processors work together to achieve a goal of better efficiency. In the non-cooperative individual processors act as independent entities and arrive at decisions about the use of their resources without any effect of their decision on the rest of the system.

H. Process Migration statergy in LBA:

Process migration parameter provides when does a system decide to export a process? It decides whether to create it locally or create it on a remote processing element. The algorithm is capable to decide that it should make changes of load distribution during execution of process or not.

I. Resource Utilization:

Resource utilization include automatic load balancing A distributed system may have unexpected number of processes that demand more processing power. If the algorithm is capable to utilize resources, they can be moved to under loaded processors more efficiently.

VI. ACKNOWLDGEMENT

The above reviews, discussion and analysis shows that static load balancing algorithms are more stable in compare to dynamic and it is also ease to predict the behavior of static, but at a same time dynamic distributed algorithms are always considered better than static algorithms.

VII. REFERENCES

[1] Kamran Zamanifar, Naser Nematbakhsh, Razieh Sadat Sadjady, ”A New Load Balancing Algorithm in Parallel Computing,” proceeding in Conference on Communication Software and Networks, 26-28 Feb 2010, pp. 449-453.

[2] J. A. Chhabra, G. Singh, E. Waraich, B. Sidhu, G. Kumar, “Qualitative parametric comparison of load balancing algorithms in parallel and distributed computing environment,” Proc. World Academy of Science, Engineering and Technology (PWASET), vol. 16, November 2006.

[3] Kashif Bilal, Tassawar Iqbal, Asad Ali Safi and Nadeem Daudpota, “Dynamic Load Balancing in PVM Using Intelligent Application,” Proc. World Academy of Science, Engineering and Technology May 2005.

[4] Brighten Godfrey,Karthik Lakshminarayanan, Sonesh Surana , Richard Karp Ion Stoica, “Load Balancing in Dynamic Structured P2P Systems,” IEEE INFOCOM, 2004.

[5] Juanjuan Min,Huazhong Liu, Anyuan Deng, Jihong Ding, “Adaptive load balancing optimization scheduling based on genetic algorithm”, Proc. ICCSIT, vol. 8, July 2010, pp. 81-85.K. Elissa, “Title of paper if known,” unpublished.

[6] Ali M. Alakeel, “Load Balancing in Distributed Computer Systems,” IJCSIS, International Journal of Computer Science and Information Security, vol.8, April 2010.

[7] Hakoan Bryhni, ”A Dynamic Sliding Load Balancing Strategy in Distributed Systems,” The International Arab Journal of Information Technology, Vol. 3, No. 2, April 2006.

[8] Badonnel, R. Burgess, M. Oslo Univ. Coll., Oslo, ”Dynamic pull-based load balancing for autonomic servers”, Network Operations and Management Symposium, April 2008, pp. 751 – 754.Electronic Publication: Digital Object Identifiers (DOIs):

[9] Albert Y. Zomaya, “Observations on Using Genetic Algorithms for Dynamic Load Balancing,” Transactions on Parallel and Distributed Systems, Vol. 12, No. 9, Sept 2001.

[10] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, “Performance Analysis of Load Balancing Algorithms,” World Academy of Science, Engineering and Technology, 2008, pp. 269.

[11] Alistair Moffat William Webber Justin Zobel, “Load Balancing for Term Distributed Parallel Retrieval,” ACM SIGIR, 2006.

[12] Ahmad Dalal’ah, “A Dynamic Sliding Load Balancing Strategy in Distributed Systems,” The International Arab Journal of Information Technology, Vol. 3, No. 2, April 2006.

[13] Zhong Xu, Rong Huang, "Performance Study of Load Balancing Algorithms in Distributed Web Server Systems", CS213 Parallel and Distributed Processing Project Report.

[14] R. Motwani and P. Raghavan, “Randomized algorithms”, ACM Computing Surveys (CSUR), 28(1):33-37, 1996.

[15] S. Malik, “Dynamic Load Balancing in a Network of Workstation”,95.515 Research Report, 19 November, 2000.

[16] Y.Wang and R. Morris, "Load balancing in distributed systems," IEEE Trans. Computing. C-34, no. 3, pp. 204-217,Mar.1985.

[17] William Leinberger, George Karypis, Vipin Kumar, "Load Balancing Across Near-Homogeneous Multi-Resource Servers", 0-7695-0556-2/00, 2000 IEEE.

[18] Tarek Helmy* Fahd S. Al-Otaibi, “Dynamic Load-Balancing Based on a Coordinator and Backup Automatic Election in Distributed Systems, “International Journal of Computing & Information Sciences Vol. 9, No. 1, April 2011.

[19] Pavankumar Kolla, Kola Haripriyanka , “Efficient Dynamic Load Balancing Algorithms for Multiclass Jobs in Peer to Peer Networks, “ International Journal of Computer Science and Network Security, Vol.11 No.3, March 2011.

[20] Zeng Zeng, and Bharadwaj Veeravalli, “Design andPerformance Evaluation of Queue-and-Rate-AdjustmentDynamic Load Balancing Policies for Distributed Networks”IEEE TRANS. ON COMPUTERS, Vol. 55, No. 11, November 2006.