International Journal of Advance Research in Science and Engineering
Vol. No.4, Issue 11, November 2015
www.ijarse.com
IJARSE
ISSN 2319 - 8354

PERFORMANCE ANALYSIS OF SLB AND DLB ALGORITHMS IN DISTRIBUTED COMPUTING ENVIRONMENT

D.Sandhya Rani[1], D.Shalini[2], M.Suman Kumar[3]

[1]*Computer Science & Informatics, University College of Engineering& Technology, MGU, (India)*

[2]*Computer Science & Engineering, SRTIST, (India)*

[3]*Electronics & Communication Engineering, GNIT, (India)*

ABSTRACT

Distributed Systems are gradually being accepted as the dominant computing paradigm of the future. A Distributed system is a software system in which components located on networked computers communicate and co-ordinate their actions by passing messages. Load Balancing involves assigning tasks to each processor and minimizing the execution time of the program. In this paper we present the performance analysis of various load balancing algorithms based on different parameters considering two typically load balancing approaches static and dynamic and also their merits and demerits and by comparision on certain parameters. The main purpose of this paper is to help in design of new algorithms in future by studying the behaviour of various existing algorithms.

Keywords: Distributed Computing, DLB, Load Balancing, SLB

I. INTRODUCTION

Processing speed of a system is always highly intended. In parallel and distributed systems more than one processor process parallel programs. The amount of processing time needed to execute all processes assigned to a processor is called workload of a processor. The amount of processing time needed to execute all processes assigned to processor is called load of a processor. Recently distributed systems with several hundred powerful processors have been developed. Distributed computing system provides high performance environment that are able to provide huge processing power. The main goal is to distribute processes among processors to maximize throughput, minimize communication delays, maximize resource utilization, maintain stability and fault tolerant[1]. The distribution of loads to the processing elements is called load balancing problem. In a system with multiple nodes there is a high chance that some nodes will be idle while the other will be overloaded. In previous load balancing studies, a common approach is to use a simple model of the distributed system with assumptions such as large communication bandwidth, negligible load balancing and to search for complex load balancing algorithms whose viability is questionable and which might provide only little or no gain when evaluated on realistic systems. In this paper, the goal of the load balancing algorithms is to maintain the load to each processing element such that all processing elements become neither overloaded nor idle[2][3]. Therfore the proper design of a load balancing algorithm may significantly improve the performance of a system.

II. LOAD BALANCING

Load Balancing is used to distribute work between two or more processors, computers, networks or memory devices in order to channelize the resources in an efficient manner and to get optimized response times and throughputs. Load balancing is the way of distributing load units across a set of processors which are connected to a network which may be distributed across the globe. By load balancing it is possible to make every processor equally busy and to finish the works approximately at same time.

2.1. Benefits of Load Balancing

a) Load Balancing improves the overall performance of each node and hence the overall system performance.
b) Load balancing reduces the job idle time.
c) Maximum utilization of resources.
d) Higher throughput.
e) Response time becomes shorter
f) Low cost but high gain

2.2 Static Load Balancing

In static load balancing algorithm the processes are assigned to the processors at the compile time according to the performance of the node. Number of jobs in each node is fixed in static load balancing algorithm .The assignment of jobs is done to the processing nodes on the basis of the following factors: incoming time, extent of resources needed, mean execution time and inter process communication.

Fig 1. Shows the schematic diagram of static load balancing where local tasks arrive at the assignment queue. A job either be transferred to a remote node or can be assigned to threshold queue from the assignment queue. Once a job is assigned to threshold queue, it cannot be migrated to any node. A job arriving at any node either processed by that node or transferred to another node for remote processing through the communication network.

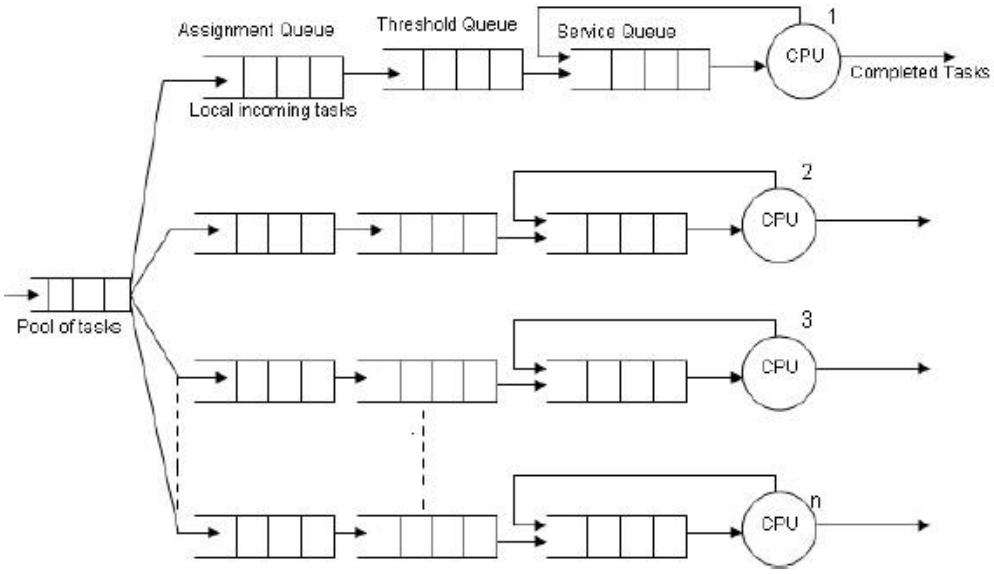

Fig 1. Model of Processing Node

Static Load Balancing can be classified into two categories – optimal and sub-optimal.

2.2.1. Optimal SLB

When all the information regarding the state of the system as well as the resource needs is known an optimal assignment can be made based on some criterion function. Examples of optimization measures are minimizing total process completion time, maximizing utilization of resources in the system, or maximizing system throughput. For example simulated Annealing (SA) and genetic algorithms (GA's) are optimization techniques

2.2.2. Sub-Optimal SLB

When for some of computations, optimal solution does not exist then sub-optimal methods can be applied. These methods rely on the rules-of-thumb and heuristics to guide a scheduling process. List scheduling is the most popular technique despite of poor performance in high communication delay situations. Lot of static algorithms, taking into account their optimal and sub-optimal nature, has been suggested by researchers so far. This includes approximate algorithms like Solution space enumeration and search, Graph theoretic approach[4][5] , Mathematical programming and queuing theoretic. Some other are round-robin algorithm, recursive-bisection algorithm, heuristic algorithms and randomized algorithms.

2.3 Dynamic Load Balancing

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.

2.3.1. Central Queue

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.

2.3.2. Local Queue

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.

A DLB algorithm considers following issues:

(**1) Load estimation policy**: which determines how to estimate the workload of a particular node of the system.

(2) Process transfer policy: which determines whether to execute a process locally or remotely.

(3) State information exchange policy: which determines how to exchange the system load information among the nodes.

(4) Priority assignment policy: which determines the priority of execution of local and remote processes at a particular node.

(5) Migration limiting policy: which determines the total number of times a process, can migrate from one node to another.

III. IDENTIFICATION OF COMPARATIVE PARAMETERS

1. Nature: This factor is related with determining the nature or behavior of load balancing algorithms, that is whether the algorithm is of static or dynamic nature, pre-planned or no planning. SLB algorithms are of static and planned nature as tasks are assigned statically i.e. at compile time in a planned manner at compile time to processors and there will be no redistribution of tasks takes place afterwards and outcome of the algorithm is deterministic as much of the job information is known a priori. DLB algorithms are of dynamic and no-planning nature as tasks are assigned at run-time to processors and tasks redistribution can take place if task assignment that was earlier done is not giving good performance (that is if proper balancing of load is not there). So their behavior is totally nondeterministic and no initial planning is done for assigning load to hosts as this work is done at run-time.

2. Overhead Associated : This factor is related with determining the amount of overhead involved while implementing a load-balancing algorithm. It is composed of overhead due to movement (relocation) of tasks, inter-processor communication, and inter-process communication. SLB algorithms incurs lesser overhead as once tasks are assigned to processors, no redistribution of tasks takes place, so no relocation overhead. DLB algorithms incur more overhead relatively as relocation of tasks takes place here.

3. Resource Utilization : This factor is used to check the resource utilization. SLB algorithms have lesser resource utilization as static load balancing methods just tries to assign tasks to processors in order to achieve minimize response time ignoring the fact that may be using this task assignment can result into a situation in which some processors finish their work early and sit idle due to lack of work. DLB algorithms have relatively better resource utilization as dynamic load balancing take care of the fact that load should be equally distributed to processors so that no processors should sit idle.

4. Processor Thrashing: Processor thrashing occurs when most of the processors of the system are spending most of their time migrating processes

5. Preemptiveness : This factor is related with checking the fact that whether tasks in execution can be transferred to other nodes (processors) or not. SLB algorithms are inherently non-preemptive as no tasks are relocated. DLB algorithms are both preemptive and non preemptive.

6. Predictability: This factor is related with the deterministic or nondeterministic factor that is to predict the outcome of the algorithm. SLB algorithm's behavior is predictable as most of the things like average execution time of processes and workload assignment to processors are fixed at compile-time. DLB algorithm's behavior is unpredictable, as everything has been done at run time.

7. Adaptability :This factor is used to check whether the algorithm is adaptive to varying or changing situations i.e. situations which are of dynamic nature. SLB algorithms are not adaptive towards all circumstances as this method fails in dynamic or varying nature problems i.e. situation in which number of processes are not fixed, also in situations which may require indeterminate steps towards solution. DLB algorithms are adaptive towards every situation whether numbers of processes are fixed or varying one.

8. Reliability: Which algorithm is more reliable in case of some host failure occurs. SLB algorithms are less reliable because no task/process will be relocated / transferred to another host in case a node fails at run-time. DLB algorithms are relatively more reliable as here processes can be transferred to other nodes in case of failure of node occurs.

9. Response Time: How much time a distributed system using a particular load balancing algorithm is taking to respond? SLB algorithms have shorter response time as one should not forget that in SLB there is lesser overhead as discussed earlier so emphasis is totally on executing jobs in shorter time rather than optimally utilizing the available resources. DLB algorithms may have relatively higher response time as sometimes redistribution of processes takes place. Some time is being consumed during task migration

10. Stability : Stability can be related to the exchange of present workload state information among processors.

IV. COMPARISION

This comparison work in tabular form is shown below.

S.NO	parameters	Load Balancing	
		SLB Algorithm	**DLB algorithm**
	Nature	Static i.e. workload is assigned at compile time	Dynamic i.e. workload is assigned at run time
1	Associated overhead	Lesser overhead	More overhead
2	Resource Utilization	Lesser Utilization	More Utilization
3	.Processor Thrashing	No Thrashing	Substantial Thrashing
4	Preemptiveness	Non-preemptive	Preemptive and Nonpreemptive
5	Predictability	More Predictable	Lesser predictable
6	Adaptability	Less adaptive	More Adaptive
7	.Reliability	Less	More
8	Response Time	Less	More
9	Stability	More	Less

V. CONCLUSION

Load balancing algorithms is totally dependent upon in which situations workload is assigned, during compile time or execution time. The above comparison shows that static load balancing algorithms are more stable than dynamic. But dynamic load balancing algorithms are always better than static as per as overload rejection, reliability, adaptability, cooperativeness, fault tolerant, resource utilization, response & waiting time and throughput is concert. The purpose of this paper was to compare these load balancing algorithms based on identified qualitative parameters.

REFERENCES

[1] Abhijit A. Rajguru, S.S. Apt" A Comparative Performance Analysis of Load Balancing Algorithms in Distributed System using Qualitative Parameters" International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-1, Issue-3, August 2012.

[2] Berenbrink P., Friedetzky T. and Steger A. "Randomized and Adversial Load Balancing". CIteSeerx,1997

[3] Hamdi M. and Lin C.K. "Dynamic Load Balancing of Data Parallel Applications on a Distributed Network". In 9th International Conference on Supercomputing,ACM,170-179,1995.

[4] H.S. Stone. "Multiprocessor scheduling with the aid of network flow algorithms". IEEE Trans of Software Engineering, SE-3(1):95--93, January 1977.

[5] H.S. Stone, "Critical Load Factors in Two-Processor Distributed Systems," IEEE Trans. Software Eng., vol. 4, no. 3, May 1978.

[6] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma," Performance Analysis of Load Balancing Algorithms", World Academy of Science, Engineering and Technology 38 2008

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