World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:4, 2008

Qualitative Parametric Comparison of Load Balancing Algorithms in Parallel and Distributed Computing Environment

Amit Chhabra, Gurvinder Singh, Sandeep Singh Waraich, Bhavneet Sidhu, and Gaurav Kumar

***Abstract*—**Decrease in hardware costs and advances in computer networking technologies have led to increased interest in the use of large-scale parallel and distributed computing systems. One of the biggest issues in such systems is the development of effective techniques/algorithms for the distribution of the processes/load of a parallel program on multiple hosts to achieve goal(s) such as minimizing execution time, minimizing communication delays, maximizing resource utilization and maximizing throughput. Substantive research using queuing analysis and assuming job arrivals following a Poisson pattern, have shown that in a multi-host system the probability of one of the hosts being idle while other host has multiple jobs queued up can be very high. Such imbalances in system load suggest that performance can be improved by either transferring jobs from the currently heavily loaded hosts to the lightly loaded ones or distributing load evenly/fairly among the hosts .The algorithms known as load balancing algorithms, helps to achieve the above said goal(s). These algorithms come into two basic categories - static and dynamic. Whereas static load balancing algorithms (SLB) take decisions regarding assignment of tasks to processors based on the average estimated values of process execution times and communication delays at compile time, Dynamic load balancing algorithms (DLB) are adaptive to changing situations and take decisions at run time.

The objective of this paper work is to identify qualitative parameters for the comparison of above said algorithms. In future this work can be extended to develop an experimental environment to study these Load balancing algorithms based on comparative parameters quantitatively.

***Keywords*—**SLB, DLB, Host, Algorithm and Load.

I. Introduction to Parallel and Distributed Computing

THE advents in the today's micro-electronic technology have resulted in the availability of fast, inexpensive processors and advancement in the communication technology has resulted in the availability of cost-effective and highly efficient computer networks. The net result of the advancement in these two technologies is that the price/performance ratio has now changed to favor the use of interconnected, multiple hosts instead of single high-speed processor. These interconnected multiple hosts either loosely or tightly coupled constitutes distributed and parallel computing environment respectively.

Some of the major benefits of parallel computing systems are information sharing among distributed users, resource sharing, better price/performance ratio, shorter response time, higher throughput, higher reliability, extensibility and incremental growth.

. Manuscript received on 20 May 2006.

All authors are with Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar, India (e-mails: chhabra_amit78@yahoo.com, gsbawa71@yahoo.com, sandeepwraich@yahoo.com, bhavneet_sidhu@yahoo.co.in, gauravk6in@yahoo.com).

II. Load Balancing Algorithms

Load balancing on multi computers is a challenge due to the autonomy of the processors and the interprocessor communication overhead incurred in the collection of state information, communication delays, redistribution of load etc. Parallel and distributed computing environment is inherently best choice for solving/running distributed and parallel program applications. In such type of applications, a large process/task is divided and then distributed among multiple hosts for parallel computation. [5] Has pointed out that in a system of multiple hosts the probability of one of the hosts being idle while other host has multiple jobs queued up can be very high. Here load balancing is likely to improve performance Such imbalances in system load suggest that performance can be improved by either transferring jobs from the currently heavily loaded hosts to the lightly loaded ones or distributing load evenly/fairly among the hosts .The algorithms known as load balancing algorithms, helps to achieve the above said goal(s).

The processors are categorized according to workload in their CPU queues as heavily loaded (more tasks are waiting to be executed), lightly loaded (less tasks are waiting to be executed in CPU queue) and idle processors/hosts (having no pending work for execution). Here CPU queue length is used as an indicator of workload at a particular processor. Some others workload indicators (also known as load index) have been suggested by researchers. But none of them is found to be an idle one. .

Substantial work has been done on load balancing in the past years, taking on a variety of forms. The general problem may be studied in different types of computing environments, using different strategies and at different levels. The system may be loosely coupled, with a number of functionally complete computers connected by one or more networks through which messages may be transmitted and remote resources accessed, or may be a tightly coupled, with several CPU-memory combinations connected by a bus to shared

memory and secondary storage. The resources to be shared may be of the same type and capacity (homogeneous system), or of different type and/or capacities (heterogeneous system). The algorithms used for load balancing may require no information, or only information about individual jobs (static algorithm) or may make decisions based on the current load situation (dynamic algorithm). The transfer of a job may be initiated by the originating host (source-initiative algorithm), or by the target host (server-initiative algorithm). The unit of execution that is to transferred/redistributed may range from complete jobs submitted by the users, or individual processes, or even smaller program modules. The units may also be the components of parallel computations with specific communication requirements. Finally, the transfer of a job may be restricted to be done prior to the start of its execution (initial job placement), or may also be allowed during its execution (process migration).

However scope of this paper is limited to identify qualitative parameters for comparison of both forms of load balancing algorithms (static and dynamic) in both parallel and distributed computing system with tightly and loosely coupled environment respectively and assuming resources to be shared are homogeneous ones. It is also assumed that preemptive process migration is possible.

The algorithm adopted for load balancing is closely related to the type and amount of load and job information assumed to be known to the decision-making modules. Accordingly these algorithms come into two basic categories - static and dynamic.

A. Static Load Balancing Algorithms (SLB)

In these types of algorithms, no dynamic load information is used, the assignments of the jobs to the processing hosts are made a priori using job information (e.g. arrival time, average execution time, amount of resources needed and their inter-process communication requirements), or probabilistically. However in reality this whole information may not be known a priori (at compile time) and things will get worse if we work in heterogeneous systems where job execution times on processing nodes will vary according to capacity of hosts. System administrators have attempted static balancing of workloads with user accounts for a long time. User accounts are assigned to the available machines in such a way that the workload generated by the users is balanced in the long run. Such a method is simple and potentially effective, but is severely limited by administrative considerations (e.g., students in one class need to be assigned to the same machine) and often in situations in which some of the machines is heavily congested, while others stay almost idle. Periodic reassignment of the user accounts may also be necessary as the user demand change.

Static load balancing can be classified into two categories – optimal and sub-optimal.

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. 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 [6][7], Mathematical programming and queuing theoretic. Some other are round-robin algorithm, recursive-bisection algorithm, heuristic algorithms and randomized algorithms.

B. Dynamic Load Balancing Algorithms (DLB)

The main problem with SLB algorithms was that they assume too much job information which may not be known in advance even if it is available, intensive computation may be involved in obtaining the optimal schedule. Because of this drawback much of the interest in load balancing research has shifted to DLB algorithms that consider the current load conditions (i.e. at execution time) in making job transfer decisions. So here the workload is not assigned statically to the processing hosts as was being done in SLB but instead of this workload can be redistributed among hosts at the runtime as the circumstances changes i.e. transferring the tasks from heavily loaded processors to the lightly loaded ones.

Dynamic load balancers continually monitor the load on all the processors, and when the load imbalance reaches some predefined level, this redistribution of work takes place. But as this monitoring steals CPU cycles so care must taken as when it should be invoked. This redistribution does incur extra overhead at execution time.

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.

A large number of algorithms have been proposed, mostly heuristic in nature, as the optimal solution often requires future knowledge and is computationally intensive. The most widely approach for studying DLB algorithms is analytic modeling and simulation. For analytic modeling, the computer system is modeled as a queuing network with job arrivals and

their resource consumptions following certain probabilistic patterns. Queuing network solution techniques are used to compute performance measures [1] [2] [3] [8]. Due to limitations of the solution techniques, simulation is often resorted to for approximate solutions [4] [5]. Some of the source-initiated DLB algorithms are by Eager [2] [3][4].

After studying various static and dynamic load-balancing algorithms, now it's time to identify several qualitative parameters for comparative study.

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 non-deterministic 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 without accomplishing any useful work in an attempt to properly schedule the processes for better performance.

SLB algorithms are free from Processor thrashing as no relocation of tasks place.

DLB algorithms incurs substantial processor thrashing.

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 non-deterministic 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.

SLB algorithms in this context can be considered as stable as no information regarding present workload state is passed among processors.

However in case of DLB such kind of information is exchanged among processors and if this information is out of date i.e. information which is not updated regularly or periodically among processors then it can lead the whole system to an unstable state.

11. Issues

Considering other major issues related with load balancing algorithms.

SLB algorithms-The major issue in static load balancing is to accurately determining the process execution times, communication delays and other resource needs of a processor a priori. A prior accurate estimation is not possible in reality, so emphasis can be done on to estimation of such quantities close to accurate value.

DLB algorithms- The major issue concerning DLB algorithms is to develop fast methods for distributed termination detection and to develop techniques of reducing overhead which includes inter processor communication overhead and task migration overhead, which is main problem in dynamic load balancing.

IV. Comparison

This comparison work in tabular form is shown below.

TABLE I
SHOWING COMPARISON WORK

Load balancing / Parameters	SLB Algorithms	DLB Algorithms
1.Nature	Static i.e. workload is assigned at compile time	Dynamic i.e. workload is assigned at run time
2.Associated overhead	Lesser overhead	More overhead
3.Resource Utilization	Lesser Utilization	More Utilization
4.Processor Thrashing	No Thrashing	Substantial Thrashing
5.Preemptiveness	Non-preemptive	Preemptive and Non-preemptive
6.Predictability	More Predictable	Lesser predictable
7.Adaptability	Less adaptive	More Adaptive
8.Reliability	Less	More
9.Response Time	Less	More
10.Stability	More	Less
11.Other Issues	Determining process execution time at run time	Developing techniques to reduce communication overhead

V. Conclusion and Future Directions

Use of load balancing algorithms is totally dependent upon underlying situations if process execution times and inter processor communication time can be estimated a priori i.e. at compile time which is practically difficult then Static Load balancing algorithms should be employed otherwise dynamic load balancing algorithms should be used. The purpose of this paper was to compare these load balancing algorithms based on identified qualitative parameters.

References

[1] Y.C. Chow and W. Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Mu1tiple Processor System," *IEEE Transactions on Computers*, Vol. C-28, pp. 334-361, May 1979.

[2] D L Eager, E D Lazowska , J Zahorjan, "A comparison of receiver-initiated and sender-initiated adaptive load sharing", *Performance Evaluation*, v.6 n.1, p.53-68, March 1986.

[3] Derek L. Eager, Edward D. Lazowska , John Zahorjan, "Adaptive load sharing in homogeneous distributed systems", *IEEE Transactions on Software Engineering*, v.12 n.5, p.662-675, May 1986.

[4] C.H.Hsu and J.W.Liu "Dynamic Load Balancing Algorithms in Homogeneous Distributed System," *Proceedings of The 6th International Conference on Distributed Computing Systems*, May, 1986, pp. 216-223.

[5] Miron Livny, Myron Melman, "Load balancing in homogeneous broadcast distributed systems", *Proceedings of the Computer Network Performance Symposium*, p.47-55, April 13-14, 1982, College Park, Maryland, United States.

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

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

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