International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-1 Issue-3, August 2012

A Comparative Performance Analysis of Load Balancing Algorithms in Distributed System using Qualitative Parameters

Abhijit A. Rajguru, S.S. Apte

Abstract - A distributed system can be viewed as a collection of computing and communication resources shared by active users. In this environment, a number of workstations or computers are linked through a communication network to form a large loosely coupled distributed computing system. When the demand for computing power increases, the load balancing problem becomes important. The problem of task scheduling and load balancing in distributed system are most important and challenging area of research in computer engineering. Task Scheduling and load balancing in distributed system has an important role in overall system performance. Task scheduling in distributed system can be defined as allocating processes to processor so that total execution time will be minimized, utilization of processors will be optimized. Load balancing is the process of improving the performance of system through a redistribution of load among processor.

In this paper we present the performance analysis of various load balancing algorithms based on different parameters, considering two load balancing approaches static and dynamic. The analysis indicates that static and dynamic both types of algorithm have some advantages as well as disadvantages. The main purpose of this paper is to help in design of new algorithms in future by studying existing static and dynamic algorithms.

Keywords: Load balancing, static load balancing, dynamic load balancing, comparative parameters

I. INTRODUCTION:

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 [1]. Load balancing involves the distribution of jobs throughout a networked computer system, thus increasing throughput without having to obtain additional or faster computer hardware. [2] Load balancing is to ensure that every processor in the system does approximately the same amount of work at any point of time. [1]

An important problem here is to decide how to achieve a balance in the load distribution between processors so that the computation is completed in the shortest possible time.

II. LOAD BALANCING:

Load balancing is the process of improving the performance of system through a redistribution of load among processor.

2.1 Issues of load balancing and scheduling

The load balancing mechanism in distributed systems has more issues as there is no centralized authority to allocate the work load among multiple processors. Some of the issues are described below,

- A good load balancing scheme needs to be general, stable, scalable, and to add a small overhead to the system. These requirements are interdependent. [3]
- Load balancing is critical because processes may migrate from one node to another even in the middle of execution to ensure equal workload. [1]
- An important problem is to decide how to achieve a balance in the load distribution between processors so that the computation is completed in the shortest possible time.
- 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. [1]
- Load sharing struggle to avoid the unshared state in processors which remain idle while tasks compete for service at some other processor [1]
- One of the crucial aspects of the scheduling problem is load balancing. [4] The challenge for a scheduling algorithm is that the requirements of fairness and data locality often conflict.
- Load balancing and task scheduling in distributed operating systems is a critical factor in overall system efficiency because the distributed system is non-uniform and non-preemptive, that is, the processors may be different. [4]

2.2 Types of Load Balancing Algorithms:

Load balancing algorithms can have three categories based on initiation of process as follows:

- **Sender Initiated:** In this type the load balancing algorithm is initialized by the sender. In this type of algorithm the sender sends request messages till it finds a receiver that can accept the load.
- **Receiver Initiated:** In this type the load balancing algorithm is initiated by the receiver. In this type of description algorithms the receiver sends request messages till it finds a sender that can get the load.
- **Symmetric:** It is the combination of both sender initiated and receiver initiated Depending on the current state of the system, load balancing algorithms can be divided into 2 categories as static and dynamic load balancing algorithms.

Revised Manuscript Received on 30 August 2012.
* Correspondence Author
Abhijit Rajguru*, Pursuing the Ph.D.(CSE) Degree from Solapur University, India
Dr.Mrs.S.S.Apte, Professor, CSE department in WIT Solapur, India

III. STATIC LOAD- BALANCING

Static load balancing policies are generally based on the information about the average behavior of system; transfer decisions are independent of the actual current system state. Static load balancing schemes use a priori knowledge of the applications and statistical information about the system.

In static load balancing, the performance of the processors is determined at the beginning of execution. Then depending upon their performance the work load is assigned 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. The goal of static load balancing method is to reduce the execution time, minimizing the communication delays.

A general disadvantage of static approaches is that the final selection of a host for process allocation is made when the process is created and cannot be changed during process execution to make changes in the system load.

There are four types of static load balancing: - Round Robin algorithm, Randomized algorithm, Central Manager Algorithm, and Threshold algorithm.

3.1 Round Robin Algorithm

Round Robin algorithm [5] distributes jobs evenly to all slave processors. All jobs are assigned to slave processors based on Round Robin order, meaning that processor choosing is performed in series and will be back to the first processor if the last processor has been reached. Processors choosing are performed locally on each processor, independent of allocations of other processors. The main advantage of Round Robin algorithm is that it does not require inter process communication. In general Round Robin is not expected to achieve good performance in general case.

However when the jobs are of unequal processing time this algorithm suffers as the some nodes can become severely loaded while others remain idle. Round Robin is generally used in web servers where generally HTTP requests are of similar nature and thereby be distributed equally.

3.2 Randomized Algorithm

Randomized algorithm [5] uses random numbers to choose slave processors. The slave processors are chosen randomly following random numbers generated based on a statistic distribution. Randomized algorithm can attain the best performance among all load balancing algorithms for particular special purpose applications.

3.3. Central Manager Algorithm

Central Manager Algorithm [6], in each step, central processor will choose a slave processor to be assigned a job. The chosen slave processor is the processor having the least load. The central processor is able to gather all slave processors load information, thereof the choosing based on this algorithm are possible to be performed. 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.

3.4. Threshold Algorithm

In Threshold algorithm [6], 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: under loaded, medium and overloaded. Two threshold parameters t_under and t_upper can be used to describe these levels.

Under loaded: load < t_under ,
Medium : t_under ≤ load ≤ t_upper ,
Overloaded: load > t_upper.

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 another overloaded processor, causing significant disturbance in load balancing, and increasing the execution time of an application [5].

IV. DYNAMIC LOAD BALANCING ALGORITHMS

In dynamic load balancing algorithms work load is distributed among the processors at runtime. The master assigns new processes to the slaves based on the new information collected [7]. In a distributed system, dynamic load balancing can be done in two different ways: distributed and non-distributed. In the distributed one, the dynamic load balancing algorithm is executed by all nodes present in the system and the task of load balancing is shared among them. The interaction among nodes to achieve load balancing can take two forms: cooperative and non-cooperative [8]. In cooperative, the nodes work side-by-side to achieve a common objective, for example, to improve the overall response time, etc. In non-cooperative, each node works independently toward a goal local to it, for example, to improve the response time of a local task. Dynamic load balancing algorithms usually generate more messages than the non-distributed ones because, each of the nodes in the system needs to communicate with every other node. A advantage of this is that even if one or more nodes in the system fail, it will not cause the total load balancing process to halt, it instead would affect the system performance to some extent. In non-distributed type, either one node or a group of nodes do the task of load balancing. Non-distributed dynamic load balancing algorithms can take two forms: centralized and semi-distributed.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

In centralized form, the load balancing algorithm is executed only by a single node in the whole system: the central node. This node is solely responsible for load balancing of the whole system. The other nodes interact only with the central node.

In semi-distributed form, nodes of the system are partitioned into clusters, where the load balancing in each cluster is of centralized form. A central node is elected in each cluster by appropriate election technique which takes care of load balancing within that cluster. Hence, the load balancing of the whole system is done via the central nodes of each cluster [8]. Centralized dynamic load balancing takes fewer messages to reach a decision, as the number of overall interactions in the system decreases drastically as compared to the semi distributed case. However, centralized algorithms can cause a bottleneck in the system at the central node and also the load balancing process is rendered useless once the central node crashes. Therefore, this algorithm is most suited for networks with small size.

4.1. Policies or Strategies in dynamic load balancing

There are different policies in dynamic load balancing [8]:

1. **Transfer Policy:** The part of the dynamic load balancing algorithm which selects a job for transferring from a local node to a remote node is referred to as Transfer policy or Transfer strategy.
2. **Selection Policy:** It specifies the processors involved in the load exchange (processor matching)
3. **Location Policy:** The part of the load balancing algorithm which selects a destination node for a transferred task is referred to as location policy or Location strategy.
4. **Information Policy:** The part of the dynamic load balancing algorithm responsible for collecting information about the nodes in the system is referred to as Information policy or Information strategy.
5. **Load estimation policy**: which determines how to estimate the workload of a particular node of the system.
6. **Process transfer policy**, which determines whether to execute a process locally or remotely.
7. **Priority assignment policy**: which determines the priority of execution of local and remote processes at a particular node.
8. **Migration limiting policy**: which determines the total number of times a process, can migrate from one node to another.

There are two types of dynamic load balancing: - Central Queue Algorithm, Local Queue Algorithm

4.2. Central Queue Algorithm

Central Queue Algorithm [9] 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.

4.3. Local Queue Algorithm

Main feature of Local Queue algorithm [9] 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.

V. QUALITATIVE PARAMETERS

5.1. Nature

This factor is related with determining the nature or behavior of load balancing algorithms that is whether the load balancing algorithm is of static or dynamic nature, pre-planned or no planning.

5.2. Overload Rejection

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.

Static load balancing algorithms incurs lesser overhead as once tasks are assigned to processors, no redistribution of tasks takes place, so no relocation overhead. Dynamic Load Balancing algorithms incur more overhead relatively as relocation of tasks takes place.

5.3. Reliability

This factor is related with the reliability of algorithms in case of some machine failure occurs.

Static load balancing algorithms are less reliable because no task/process will be transferred to another host in case a machine fails at run-time. Dynamic load balancing algorithms are more reliable as processes can be transferred to other machine in case of failure occurs.

5.4. Adaptability

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

This factor is used to check whether the algorithm is adaptive to varying or changing situations i.e. situations which are of dynamic nature.

Static load balancing algorithms are not adaptive as this method fails in varying nature problems i.e. situation in which number of processes are not fixed. Dynamic load balancing algorithms are adaptive towards every situation whether numbers of processes are fixed or varying one.

5.5. 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.

Static load balancing algorithm considered as stable as no information regarding present workload state is passed among processors. However in case of dynamic load balancing such kind of information is exchanged among processors.

5.6. Predictability

This factor is related with the deterministic or nondeterministic factor that is to predict the outcome of the algorithm.

Static load balancing 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. Dynamic load balancing algorithm's behavior is unpredictable, as everything has been done at run time.

5.7. Forecasting Accuracy

Forecasting is the degree of conformity of calculated results to its actual value that will be generated after execution.

5.8. Cooperative

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.

5.9. Fault Tolerant

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.

5.10. 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.

Static load balancing 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.

Dynamic load balancing 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.

5.11. Process Migration

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.

5.12. Preemptiveness

This factor is related with checking the fact that whether load balancing algorithms are inherently non-preemptive as no tasks are relocated. Dynamic load balancing algorithms are both preemptive and non preemptive.

5.13. Response Time

How much time a distributed system using a particular load balancing algorithm is taking to respond?

Static load balancing algorithms have shorter response time as one should not forget that in Static load balancing there is lesser overhead as discussed earlier so emphasis is totally on executing jobs in shorter time rather than optimally utilizing the available resources.

Dynamic load balancing algorithms may have relatively higher response time as sometimes redistribution of processes takes place. Some time is being consumed during task migration

5.14. Waiting Time

Waiting Time is the sum of the periods spent waiting in the ready queue.

5.15. Turnaround Time

Te interval from the time of submission of a process to the time of completion is the turnaround time.

5.16. Execution System

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.

5.17. Through put

Throughput is the amount of data moved successfully from one place to another in a given time period.

5.18. 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.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Static load balancing algorithms are free from Processor thrashing as no relocation of tasks place. Dynamic load balancing algorithms incurs substantial processor thrashing.

Parameters	Round Robin	Random	Local queue	Central Queue	Central Manager	Threshold
Nature	Static	Static	Dynamic	Dynamic	Static	Static
Overload Rejection	No	No	Yes	Yes	No	No
Reliability	Less	Less	More	More	Less	Less
Adaptability	Less	Less	More	More	Less	Less
Stability	Large	Large	Small	Small	Large	Large
Predictability	More	More	Less	Less	More	More
Forecasting Agency	More	More	Less	Less	More	More
Cooperative	No	No	Yes	Yes	Yes	Yes
Fault Tolerant	No	No	Yes	Yes	Yes	No
Resource Utilization	Less	Less	More	Less	Less	Less
Process Migration	No	No	Yes	No	No	No
Preemptiveness	Non-preemptive	Non-preemptive	Preemptive and Non-preemptive	Preemptive and Non-preemptive	Non-preemptive	Non-preemptive
Response Time	Less	Less	More	More	Less	Less
Waiting Time	More	More	Less	Less	More	More
Turnaround Time	Less	Less	More	More	Less	Less
Execution System	Decentralized	Decentralized	Decentralized	Centralized	Centralized	Decentralized
Throughput	Low	Low	High	High	Low	Low
Processor Thrashing	No	No	Yes	Yes	No	No

Table1: Comparative Analysis of Load Balancing Algorithms

VI. CONCLUSION

The purpose of this paper was to compare different load balancing algorithms based on identified qualitative parameters. In this paper we have carried out the analysis of different load balancing algorithms, various parameters are used to check the results.

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. In future work, we need more and more real experimentation to choose good load balancing algorithm.

REFERENCE

1. Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, “Performance Analysis of Load Balancing Algorithms”, World Academy of Science, Engineering and Technology, 2008.
2. Hisao Kameda, El-Zoghdy Said Fathyy and Inhwan Ryuz Jie Lix, “A Performance Comparison of Dynamic vs. Static Load Balancing Policies in a Mainframe { Personal Computer Network Model”, Proceedings of the 39th IEEE Conference on Decision and Control, 2000.
3. Daniel Grosua, Anthony T. and Chronopoulosb,”Non-cooperative load balancing in distributed systems”, Elsevier, Journal of Parallel and Distributed Computing, 2005.
4. M. Nikravan and M. H. Kashani, “A Genetic Algorithm for Process Scheduling in Distributed Operating Systems Considering Load balancing”, Proceedings 21st European Conference on Modelling and Simulation (ECMS), 2007.
5. Hendra Rahmawan, Yudi Satria Gondokaryono, “The Simulation of Static Load Balancing Algorithms”,
6. 2009 International Conference on Electrical Engineering and Informatics, Malaysia.
7. Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, “Performance Analysis of Load Balancing Algorithms”, academy of science, engineering and technology, issue 38, February 2008, pp. 269-272.
8. S. Malik, “Dynamic Load Balancing in a Network of Workstation”, 95.515 Research Report, 19 November,2000.[8] Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNSInternational Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.
9. William Leinberger, George Karypis, Vipin Kumar, "Load Balancing Across Near Homogeneous Multi-Resource Servers", 0-7695-0556-2/00, 2000 IEEE.

Abhijit Rajguru received the B.E (CSE).and M.Tech(CSE) degree, from Shivaji Univ. in 2007 and 2009, respectively. He is pursuing the Ph.D.(CSE) degree from Solapur Univ. He is working as a assistant professor (from 2009) in the Dept. of Computer Science & Engineering, in SKN Sinhgad College of Engg., His research interest includes load balancing Distributed System, cloud computing, grid computing.

Dr.Mrs.S.S.Apte received Ph.D. degree in Computer Engineering. She having 33 years teaching experience and 02 years industry experience. She is working as professor in CSE department in WIT Solapur. Her research interest includes Computer architecture, distributed system, image processing.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)