International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 07 – Issue 01, January 2018

A Review of Dynamic Load Balancing Algorithms for Shared Memory Parallel Computing

Frankline Makokha
School of Computing and Informatics
University of Nairobi
Nairobi, Kenya
Email: goldmedalist321 [at] gmail.com

Okelo-Odongo
School of Computing and Informatics
University of Nairobi
Nairobi, Kenya

Abstract—**Load balancing algorithms are used in parallel systems as a means to increase performance of the parallel systems by ensuring all the computing cores are kept busy at all times and at no single time is one core overloaded while the other cores remain idle. This paper analyses existing load balancing algorithms as used in parallel systems with the main focus being dynamic load balancing algorithm for shared memory parallel systems. Whereas studies have been done on use of load balancing algorithms, most of those studies have focused on distributed memory parallel systems leaving out shared memory parallel systems. Further those studies focused on qualitative metrics such as overload rejection, reliability, predictability, adaptability, scalability, and stability with no focus on quantitative metrics like CPU idle time and processing time. This paper recommends a comparative study of dynamic load balancing algorithms on shared memory systems, using quantitative metrics with a view of recommending the best algorithms for use in shared memory parallel computing platforms, and for which class of computational problems**

Keywords-- load balancing; distributed memory parallel systems; shared memory parallel systems; parallel computing ; distributed computing

I. INTRODUCTION

Parallel and distributed computing have offered the opportunity of solving a wide range of computationally intensive problems by increasing the computing power of sequential computers [1]. Although important improvements have been achieved in this field in the last 30 years, there are still many unresolved issues [1]. These issues arise from several broad areas such as the design of parallel systems and scalable interconnects, *the efficient distribution of processing tasks*, and the development of parallel algorithms.

It is the issue of efficient distribution of processing tasks that calls for the need for load balancing in parallel systems and distributed computing systems.

II. LOAD BALANCING

Load balancing is the process of improving the performance of a parallel and distributed system through a redistribution of tasks among the processors [2].

Load balancing is one of the central problems, which has to be solved in parallel computations [3]. Since load imbalance leads directly to processor idle times, high efficiency can only be achieved if the computational load is evenly balanced among the processors.

III. CATEGORIES OF LOAD BALANCING ALGORITHMS

Load balancing algorithms can be broadly categorized as either static load balancing algorithms or dynamic load balancing algorithms. Static load balancing algorithms distribute the tasks to processing elements at compile time, while dynamic algorithms bind tasks to processing elements at run time [4].
Dynamic load balancing is often used in dynamic and irregular problems that have been parallelized using Genetic Algorithms, in different application areas such as the electronic structure, molecular dynamics, or computational physics based on adaptive mesh refinement [5].

In dynamic load balancing, applications need information on both when and how to rebalance. The three load balancing steps are: evaluate the imbalance; decide how to balance if needed; redistribute work to correct the imbalance [6].

A further classification of load balancing algorithms by [7] is: static algorithms, dynamic algorithms, and adaptive algorithms. Static algorithms decide how to distribute the workload according a prior knowledge of the problem and the system characteristics. Dynamic algorithms use state information to make decisions during program execution. Finally, Adaptive algorithms are a special case of dynamic algorithms. They dynamically change its parameters in order to adapt its behavior to the load balancing requirements.

Dynamic load balancing strategies can be divided basically into two main classes: centralized dynamic load balancing and distributed dynamic load balancing [7].

These strategies define where the load balancing decisions are made. In a centralized scheme, the load balancer is implemented on one master processor and all decisions are made there. In a distributed scheme, the load balancer is replicated on all processors.

Dynamic load balancing algorithms are essential for efficient use of highly parallel systems when solving problems with unpredictable load estimates [8]. Recently CPU workload hardware technology and multiprocessor service are developing rapidly [9]. If all the parallel computers are not of the same type i.e. not same configuration then proper load balance may not occur and so some computers may finish their work earlier than others and sit idle. This degrades the performance of the multi computer system [9].

The various information used in making a load balancing decision are called policies [4]. The policies could be local or global. In global policies, the performance profile of all the processing units in the system is considered while in local policies, only the performance profile of a few processing units is considered when making the load balancing decisions.

The various policies used by the load balancing algorithms are: Information policy which specifies what workload information to be collected, when it is to be collected and from where; Triggering policy which determines the appropriate period to start a load balancing operation; Resource type policy which classifies a resource as server or receiver of tasks according to its availability status; Location policy which uses the results of the resource type policy to find a suitable partner for a server or receiver and Selection policy which defines the tasks that should be migrated from over loaded resources (source) to most idle resources [4].

In a set up where the load balancing decision is made by a designated processing unit, then that particular load balancing technique is categorized as centralized, while where each processing unit makes a load balancing decision, then the algorithm could be categorized as distributed.

A dynamic load balancing algorithm consists of four components: Load measurement rule, an Information Exchange rule, an Initiation rule and a Load Balancing Operation [4].

In a comparative analysis done by [4], centralized algorithms have limited scalability as their major setback while the decentralized ones are scalable. This comparison only gives a qualitative aspect of the algorithm and not a quantitative evaluation like actual performance in terms of time to completion of executing a given problem.
A good load balancing scheme needs to be general, stable, scalable, and to add a small overhead to the system [10], given that the working condition of a multicomputer system based on message passing communication is changeable and unpredictable [11].

Any good algorithm for a system must be adaptive to the dynamic change of working condition. One simple parallel method for dynamic load balancing is for each processor to transfer an amount of work to each of its neighbors which is proportional to the load difference between them (diffusion method). This process is iterated until the load difference between any two processors is smaller than a specified value [3]. However, [3] points out a key disadvantage to this method: the number of iterations required by the load balancer may be high, making the algorithm too expensive to use.

IV. EXAMPLES OF DYNAMIC LOAD BALANCING ALGORITHMS

The various dynamic load balancing algorithms include [12]:

A. *Nearest Neighbour Algorithm*

Each processor considers only its immediate neighbor processors to perform load balancing operations. A processor takes the balancing decision depending on the load it has and the load information to its immediate neighbors. By exchanging the load successively to the neighboring nodes the system attains a global balanced load state. The nearest neighbor algorithm is mainly divided into two categories, which are diffusion method and dimension exchange method.

B. *Centralised Dynamic Load Balancing Algorithm*

The Master thread holds the collection of tasks to be performed. Tasks are sent to the slave processes. When a slave process completes one task, it requests another task from the master processor.

C. *Random(RAND) Algorithm*

As soon as a workload (greater than threshold load) is generated in a processor, it is migrated to a randomly selected neighbor. It does not check state information of a node.

This algorithm neither maintains any local load information nor sends any load information to other processors. Furthermore, it is simple to design and easy to implement. Its main drawback is that it causes considerable communication overheads due to the random selection of lightly loaded processor to the nearest neighbors.

D. *Adaptive Contracting with Neighbour(ACWN)*

In this algorithm, as soon as a workload is newly generated, it is migrated to the least loaded nearest neighbor processor. The load accepting processor keeps the load in its local heap. If the

load in its heap is less than its threshold load then no problem, otherwise it sends the load to the neighbor processor, which has a load below its threshold load.
This algorithm requires maintaining the local load information and also the load information of the neighbors for exchanging the load periodically.

E. Prioritized Random (PRAND) Algorithm

In this algorithm, the work loads are assigned index numbers on the basis of the weight of their heaps. PRAND is similar to RAND except that it selects the second largest weighted load from the heap and transfers it to a randomly selected neighbor. On the other hand, Prioritized ACWN selects the second largest weighted workload and transfers it to the least loaded neighbor.

F. Cyclic Algorithm

The workload is assigned to processors in a recurrent fashion. This algorithm always remembers the last core to which a workload was sent.

G. Fuzzy Algorithm

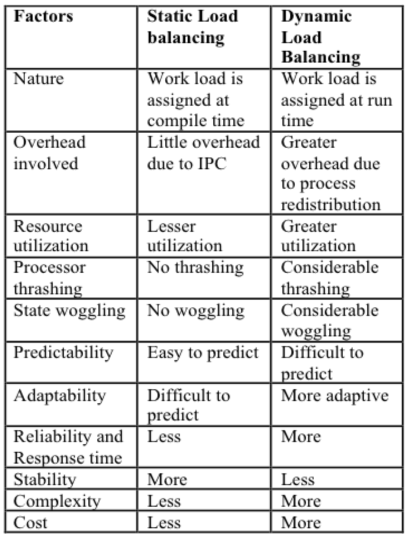
This algorithm makes use of Routing table, Load index, Cost table and a fuzzy controller, which manages Load balancing of system [13].

The routing table presents the communication links among the nodes in the system while the Load index indicates the load of its related node. In order to determine the node status as a sender, receiver or neutral by using fuzzy controller and based on fuzzy rules, there is a cost table that provides the nodes communication costs and the number of heavy loaded nodes. The cost table is obtained by using load index and routing table while the number of heavy loaded nodes can be extracted from the cost table.

V. PERFOMANCE COMPARISON FOR VARIOUS ALGORITHMS

Static and dynamic load balancing algorithms compare as shown in table 1 .

Table 1: Comparison between Static and Dynamic Load Balancing Algorithms.

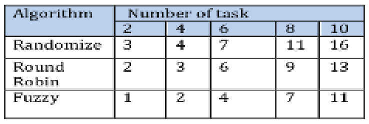


Factors	**Static Load balancing**	**Dynamic Load Balancing**
Nature	Work load is assigned at compile time	Work load is assigned at run time
Overhead involved	Little overhead due to IPC	Greater overhead due to process redistribution
Resource utilization	Lesser utilization	Greater utilization
Processor thrashing	No thrashing	Considerable thrashing
State woggling	No woggling	Considerable woggling
Predictability	Easy to predict	Difficult to predict
Adaptability	Difficult to predict	More adaptive
Reliability and Response time	Less	More
Stability	More	Less
Complexity	Less	More
Cost	Less	More

Adapted from [13]

According to a study done by [14] the fuzzy algorithm performs better that the round robin and randomized algorithm as shown in table 2 below.

Table 2: Performance Comparison between three dynamic load balancing algorithms

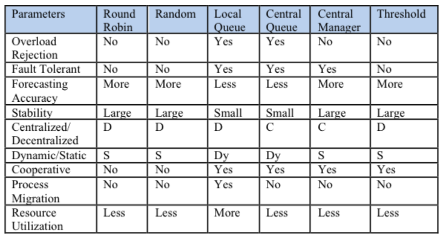


Algorithm	Number of task				
	2	4	6	8	10
Randomize	3	4	7	11	16
Round Robin	2	3	6	9	13
Fuzzy	1	2	4	7	11

Adapted from [14]

The various static and load balancing algorithms compare in performance as shown in table 3 below.

Table 3: Performance comparison between various load balancing algorithms



Parameters	Round Robin	Random	Local Queue	Central Queue	Central Manager	Threshold
Overload Rejection	No	No	Yes	Yes	No	No
Fault Tolerant	No	No	Yes	Yes	Yes	No
Forecasting Accuracy	More	More	Less	Less	More	More
Stability	Large	Large	Small	Small	Large	Large
Centralized/ Decentralized	D	D	D	C	C	D
Dynamic/Static	S	S	Dy	Dy	S	S
Cooperative	No	No	Yes	Yes	Yes	Yes
Process Migration	No	No	Yes	No	No	No
Resource Utilization	Less	Less	More	Less	Less	Less

Adapted from [2].

VI. CONCLUSION

Whereas work has been done in analysis of the various dynamic load balancing algorithms as highlighted in the literature above, most emphasis has been on distributed systems and using qualitative parameters e.g. overload rejection, reliability, predictability, adaptability, scalability, stability, waiting time, through put etc., there has been little practical emphasis on shared memory parallel systems and use of quantitative parameters like execution time and processor idle times.

However with the proliferation of multicore systems like multicores in laptop computers and smart mobile phones, there is need to perform a study on the performance of dynamic load balancing algorithms in shared memory systems with the intent of optimizing the performances of this newly emerging multicore systems.

The outcome from the study will be of importance to parallel software developers since they will use the outcome to make informed decisions on which dynamic load balancing algorithms to use and for which class of computational tasks, when developing parallel applications for shared memory multicore systems. For example for mathematical problems, sorting problems and searching problems, how do the various dynamic load balancing algorithms compare in terms of processing time and CPU idle times on shared memory parallel systems?

REFERENCES

[1] Ros, A. (Ed. (2010)) *Parallel and Distributed Computing*. Croatia: In-Tech.

[2] Sharma, S., Singh, S., and Sharma, M. (2008) Performance Analysis of Load Balancing Algorithms. *World Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Automation, Control and Information Engineering.* Vol: 2, No: 2.

[3] Horton, G. (1993) *A multi-level diffusion method for dynamic load balancing.* Erlangen: Elsevier Science Publishers B.V.

[4] Mandal, A. and Chandra, S. (2010) An Empirical Study and Analysis of the Dynamic Load Balancing Algorithms Used in Parallel Computing Systems: Proceedings of ICCS-2010,19-20 Nov. West Bengal: University of North Bengal.

[5] Krishnamoorthy, S., Sadayappan P., Nieplocha P., and Krishnan M. (2005) Locality-aware Load Balancing for Dynamic and Irregular Computations.*1st Workshop on Patterns in High Performance Computing*. Urbana-Champaign

[6] Pearce, O., Gamblin, T., Supinskiy, B., Schulzy, M., and Amato, M. (2012) *Quantifying the Effectiveness of Load Balance Algorithms.*US*:* Department of Energy

[7] Aldasht, M., Ortega, J. and Puntonet, C. (2007) *Dynamic Load Balancing in Heterogeneous Clusters: Exploitation of the Processing Power.* 2nd Palestinian International Conference on Computer and Information Technology (PICCIT), Hebron, Palestine.

[8] LeMair W.H. and Reeves P. A. (1993) Strategies for Dynamic Load Balancing on Highly Parallel computers: *IEEE Transactions on Parallel and Distributed Systems.* Vol. 4, No. 9.

[9] Pandey, S. K., and Tiwari, R. (2013). The Efficient load balancing in the parallel computer: ***I**nternational Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 4, April 2013*.

[10] Rajguru, A. and Apte, S. (2012) 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, Vol. 1, Issue-3.

[11] Kee Y. and Ha, S. *A (1998)* Robust Dynamic Load-balancing Scheme for Data Parallel Application on Message Passing Architecture. *Proceedings of the International Conference on Parallel and Distributed Processing Algorithms and Applications,* Vol. 2, Las Vegas, NM, 1998. CSREA Press: Athens, GA, 1998; 974–980.

[12] Firoj, A. & Khan, Z. (2012) The Study on Load Balancing Strategies in Distributed computing System: *International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.2.*

[13] Manekar, S. A., Mukesh, D. P., Gupta, H. and Nagle, M. (2012). A Pragmatic Study and Analysis of Load Balancing Techniques In Parallel Computing. *International Journal of Engineering Research and Applications*. Vol. 2, Issue 4.

[14] Karimi, A., Zarafshan, F., Jantan, A. B., Ramli, A. R. and Saripan, M. I.B.(2009). *International Journal of Computer Science and Information Security,* Vol. 6, No. 1.