Mr. Amitkumar S Manekar, Mr. Mukesh D Poundekar, Prof. Hitesh Gupta, Prof. Malti Nagle
/International Journal of Engineering Research and Applications
(IJERA)ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue4, July-August 2012, pp.1914-1918

A Pragmatic Study and Analysis of Load Balancing Techniques In Parallel Computing

[1]Mr. Amitkumar S Manekar, [2]Mr. Mukesh D Poundekar, [3]Prof. Hitesh Gupta, [4]Prof. Malti Nagle

[1]Research scholar, PIES, Bhopal
[2]Research scholar, PCST, Bhopal
[3]Assistant Professor, HOD CSE PCST, Bhopal
[4]Assistant Professor, HOD PIES, Bhopal

Abstract

Allocation of the work load in to small processes is known as *Load Balancing* .Parallel programming is based on four phases finding Concurrency (by understanding the available concurrency and expose in algorithm design), Algorithm structure (programmer develop high level structure for organizing parallel algorithm), Supporting structure (in this code analyzing techniques used to manage data), Implementation mechanism (final steps to look specific software construct for parallel program implementation).The middle two phases based on patterns. With availability of parallle programming models OpenMP(Shared Memory Model) MPI(Distributed Memory Model) ande Hybrid(OpemMP and MPI) there is various aspects while doing load balancing in High Performance Computing also there are typical load balancing approach, Static and Dynamic are broadly categories. For this review paper keeping vision on efficiency and speed we have discussed the aspect and issues associated with typical categorised load balancing techniques.

Keywords:-*OpenMP (shared), MPI (Distributed), Parallel computing, Distributed Computing, Load balancing.*

1. Introduction

Load balancing involves assigning a task to each processor by minimizing execution of program [1].distributed system is tremendous performance on much application everywhere. Processor speed or performance emerges supercomputer used in many sector like Science, Engineering, Industries, Commerce for their day to day needs. This demand transform serial computer to supercomputer and supercomputers to parallel distributed computing with MPC (Massively Parallel Computers). MPC is a group of processor linked with memory modules through network such as mesh, hypercube or tours [7]. Distribution of load to processing elements is simply called as the load balancing problem [1].
For better performance potential, applications have to be parallelized by re-written in to explicit parallelism. There are head tick problem to develop parallel programming (1) *as it is complex and error prone with respect to sequential programming*. (2)*Programmer's skill is not useful to device parallel algorithm and data structure due to lack of training*. For parallel programming it is advisable that simple parallel programming models such as Bulk Synchronization Parallel (BSP) model as they are capable for (1) *they are flexible enough to be mapped for wide range of parallel architecture and* (2) *Simple enough to provide a good basis for the design and analysis of parallel algorithm and data structure that offers compatible extension of the execution theory* [2].

Why load balancing?

Performance on the basis of runtime in parallel programming on specific execution platform is evaluated. Parallel runtime ***Tp* (n)** of a program is the time between the start of program and end of execution on participating processes independent of local computation, Exchange of data, synchronization, waiting times. Table 1 will illustrate parallel programming models [2].

Progr. Model	***Control Structure***	***Data View***	***Main Restriction***	***Exampl-es***
Message Passing	Asynchronous, MIMD	Local	None	MPI
Shared Memory	Asynchronous, MIMD	Shared	None	Pthreads, OpenMP, UPC, Cilk
Data-Parallel	Synchronous, SIMD	Shared	SIMD-like Control	HPF
PRAM	Synchronous, MIMD	Shared	None	Fork

BSP	Bulk-synchr., MIMD	Local	Superstep Structure	BSPlib, PUB
NestStep-BSP	Nested BSP	Shared	Superstep Structure	NestStep

Table 1- Parallel Programming Models

II. Literature survey

In upcoming years degree of on chip parallelism wills significantly increases and processor are of 10 to 100 cores [6]. For demanding speech and image reorganization or parallel browsers used web based application will generate tremendous commercial importance on availability of cheap thread type parallelism [8]. Current operating system is capable of writing and optimizing the multi-programmed. In parallel application there is higher level of inter thread interaction. Parallel application based on high level of inters thread interaction, consequently; balancing with data synchronization dependences. Specifically design for parallel computing provided new ad hoc work stealing solution for load balancing e.g. Adaptive MPI [11] and Click [10].

By load balancing it is possible to every processor equally busy and to finish the work approximately at the same time [1]. Load balancing operation is based on three rules. ***Location Rule,***

Distribution Rule and Selection Rule [12] [13] [14] [16] [17]. In practice load balancing decision are taken jointly by location and distribution rule [13] [17]. Balancing load is categories as ***LOCAL and***

GLOBAL.

Local Balancing- balancing decision is taken from group of nearest neighbors by exchanging the local workload information [1].

Global balancing- balancing decision is taken by triggering transfer patterns across the whole system and it exchange workload information globally [1].
Further load balancing types for optimizing problem is shown in fig 1.

Advantages of load balancing

(1) In load balancing overall system performance is enhance by improving the performance of each node.
(2) Less job idle time.
(3) Long starvation is avoided for small jobs.
(4) Utilization of resources is high with shorter response time.
(5) High throughput, reliability with low cost but high gain.
(6) Extendable and incremental approach.

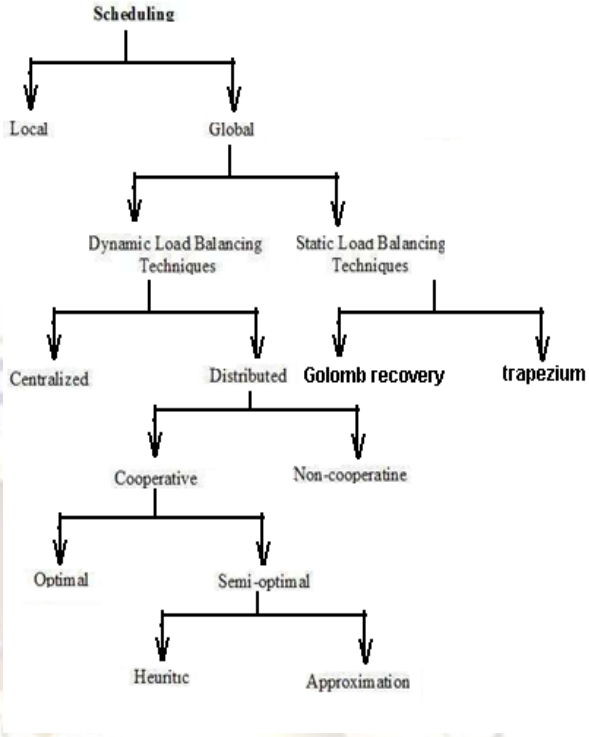

Fig1- Different Types of Load Balancing Techniques

III. Issues in load balancing

Processing element and processor (processor) execute stream of instructions is depend on both hardware and software. In some programming environments, each workstation is treated as executing a single instruction stream; in this case, a processing element is a workstation. However some may treat as individual processor.

Data Processing according to Flynn's classification of parallel architecture ***SISD***(Single Instruction Single Data) process only one stream of instructions and one stream of data, ***SIMD*** (Single Instruction Multiple Data) the control unit transmits the same instruction, simultaneously to all processing elements, ***MISD*** (Multiple Instruction Single Data), multiple instructions operate on a single data stream. Using ***MIMD*** (Multiple Instruction Multiple Data), multiple autonomous processing elements simultaneously execute different instructions on different data [15].

Some goals of load balancing algorithms is pointed as [3]

(1)Performance Improvement- By reducing task response time while keeping acceptable delay.
(2) Job equality- equally treatment for all jobs in system beside of considering their origin.
(3) Fault tolerance- partially failure of system will not have endurance on performance.

(4)Modifiability-ability of modify.

There are levels of parallelism stated as Instruction level parallelism, Statement level parallelism, Loop level parallelism and Function level parallelism, which gives different granularity result. Table 2 illustrates the different granularity between these levels.

Level Of Parallelism	***Task Performed***	***Granularity***
Instruction or statement level	Small number of instructions or statement are group to form a task	Fine grained
Functional level	Function used to form a task comprise significant amount of computation	Coarse grained
Loop level	One loop or iteration consist of several statement	Medium grained

Table 2 Different Granularity

Aspects of load balancing in mainstream architecture of HPC (OpenMp, MPI and Hybrid)

In High Performance Computing performance gain depends on settlement of point in Multi-socket Multicore shared memory computer nodes coupled via high speed interconnection [4].Fig 2 Shows typical Multi-socket, Multi-core SMP cluster.

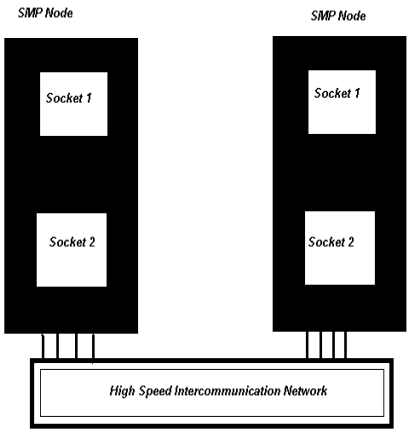

Fig 2- Typical Multi-socket, Multi-Core SMP Cluster [4]

MPI- Message Passing Interface with explicit control of parallelism in distributed memory model has static scheduling. Data placement problem are rarely observed and synchronization occurs implicitly with subroutine call and hence minimized naturally. Decomposition and debugging of application can may demand time and code change which overhead code granularity and latency problem. With MPI, global operation is very expensive [5]. Fig 3 shows Distributed Memory model MPI (Message Passing Interface).

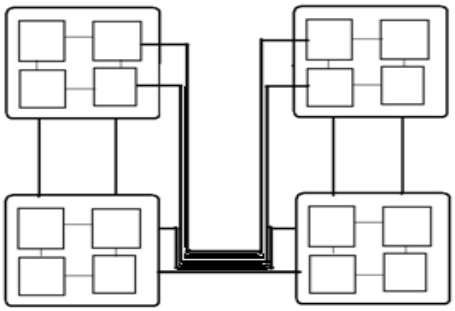

Fig3- MPI (Distributed memory model)

OpenMP- OpenMP for shared memory model is industry standard depend upon combination of compilers directives, library routines and environment variables [18]. it provide implicit communication with runtime scheduling for both fine and coarse grain parallelism with shared memory parallelism data placement may be bottleneck problem. Explicit synchronization is required just like MPI.

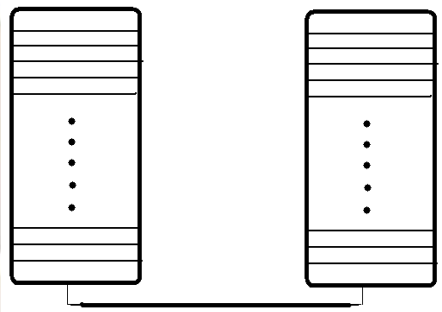

Fig4-Fully Hybrid(MPI+OpenMP)

Hybrid- Combining OpenMP and MPI for MPI's data placement and finer grain parallelism of OpenMP based on hierarchical model. At top level MPI parallelization for hybrid the noteworthy thing is that consideration of how each paradigm's carries out parallelization and whether combining two parameters provide an optimal parallelization strategy. Fig 4 shows Fully Hybrid Memory Model and fig 5 shows Mixed Model more than one MPI process per node [4].

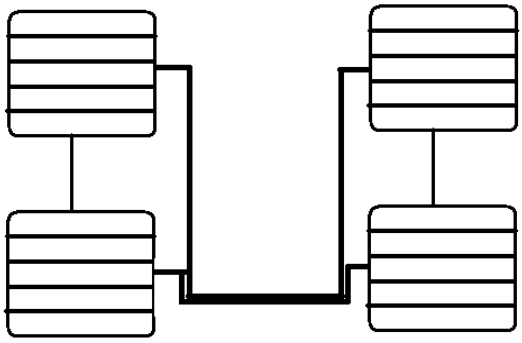

Fig5- Mixed Model more than one MPI process per node

Comparison of static vs. Dynamic Load balancing

Load balancing algorithms can be defined by their realization of the following policies [19]. *Information policy*- specifies what workload information to be collected, when it is to be collected and from where. *Triggering policy*- determines the appropriate period to start a load balancing operation. *Resource type policy*- classifies a resource as server or receiver of tasks according to its availability status. *Location policy*- uses the results of the resource type policy to find a suitable partner for a server or receiver. *Selection policy*- defines the tasks that should be migrated from overloaded resources (source) to most idle resources. Load balancing overcome problem of deciding which jobs should be allocated to which processor. Here we compare static vs dynamic load balancing.

Static Load Balancing -In static load balancing distribution finding optimal solution is the main goal. Static load balancing involved heuristics development, one common heuristics is priority to larger task. In static load balancing the challenge is that partitioning task to processor in a way that will use maximum processors utilization and minimum communication time. Fig 6 Shows model of processing node [1].

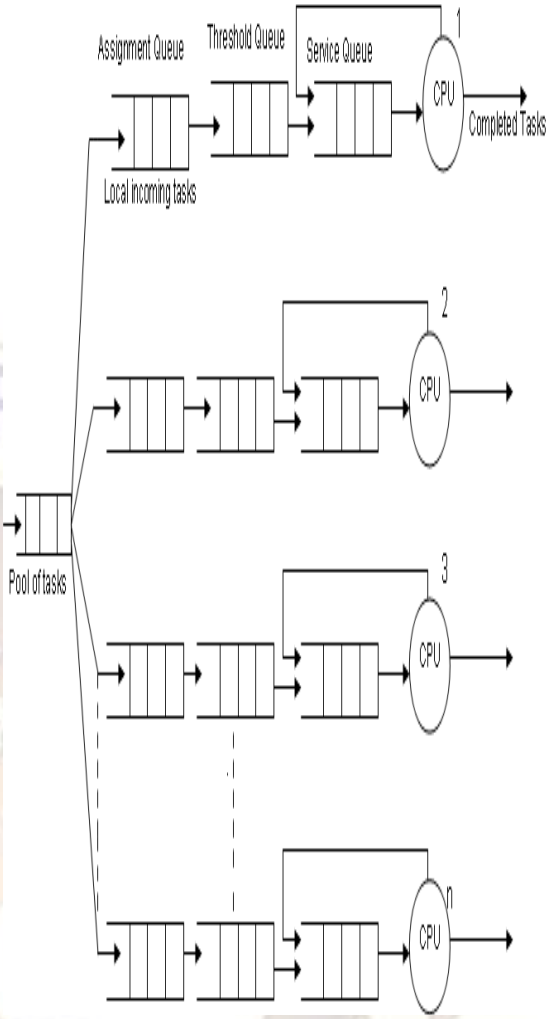

Fig 6 -Model of Processing Node

Dynamic Load Balancing-in dynamic load balancing if runtime overhead is greater than the extra work that is accomplished by moving task from heavily loaded processors to lightly loaded processors, then efficiency is bottleneck problem. Dynamic load balancing are very good at maximum utilization of processors. Dynamic load balancing algorithm can be centralized or distributed. In centralized all tasks is assign to one point for decision making. In distributed task is not shared globally on other hand all processors take part in deciding when load balancing occurs and which task will be migrated. Table 3 shows the difference between dynamic and static load balancing for parallel models adiscuss above based on some parameters. Fig 7 shows "Job Migration in Dynamic Load Balancing Strategy" [1].

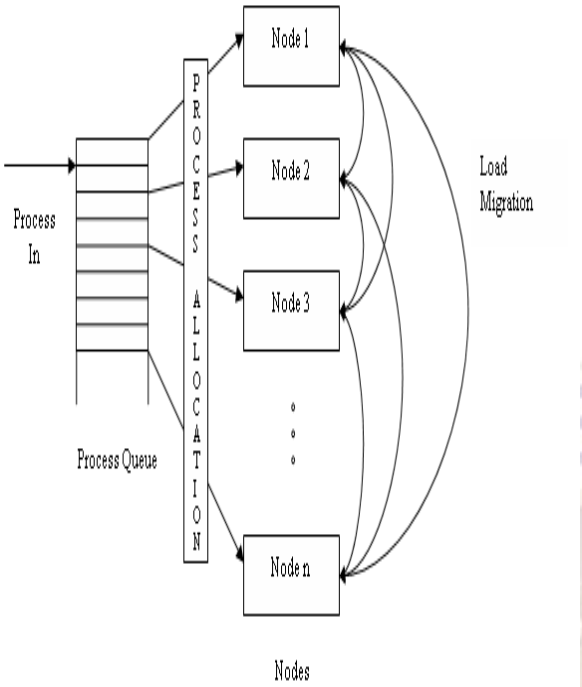

Fig 7- Job Migration in Dynamic Load Balancing Strategy.

Table 3 shows comparison of static and dynamic on some aspect of load balancing issues [1]

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

Table 3- comparison of static and dynamic load balancing techniques

IV. Conclusion

In this paper we have taken review on OpenMp for shared memory computer model, MPI a message passing library for distributed memory computer model on cluster to solve multiple task simultaneously and bigger problem is based on design pattern's ,load balancing strategies their merits and demerits, comparison on the basis on certain parameters. Based on this review we concluded that while distributing load in *Static* load balancingruntime system does not need to known in advanced, whereas in *Dynamicload balancing* runtime overload in gathering information and distribution of task to different processes. Also we concluded that if load change mechanism is faster than Static gives best performance and other side foe longer objective dynamic gives best performance.

References

[1] Md. Firoj Ali, Rafiqul Zaman Khan," The study on load balancing strategies in distributed computing system"IJCSES, Vol 3 No 2,pp 19,April 2012.

[2] C W Kessler"Teaching Parallel Programming Early" Proceedings of Workshop on Developing Computer Science Education – How Can It Be Done?, March 10, 2006, Link¨oping university, Link¨oping, Sweden.

[4] Rolf Rabenseifner, Georg Hager, Gabriele Jost," Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes" IEEE, Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on, pp427 - 436 , 2009

[5] Lorna Smith and Mark Bull "Development of mixed mode MPI / OpenMP Applications, Scientific Programming 9 (2001) 83–98, IOS Press, Scotland, UK.

[6] Steven Hofmeyr, Costin Iancu, et. al "Load Balancing on Speed"*upc.lbl.gov/publications/ppopp141-hofmeyr2010.*

[7] Antonis K., Garofalakis J., Mourtos I. and spirakis P. "A Hierarchical Adaptive Distributed Algorithm for Load Balancing". Journal of Parallel and Distributed Computing, Elsevier Inc.2003.

[8] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands,K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams,and K. A. Yelick. The Landscape of Parallel Computing Research: AView from Berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.

[9] Thomas Rauber, Gudula Rünger," Parallel Programming: for Multicore and Cluster Systems"Springer-Varlag Berling Heidelberg 2010.
[10] Leiserson. Adaptive Task Scheduling with Parallelism Feedback. In Proceedings of the Annual ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), 2006.
[11] C. Huang, O. Lawlor, and L. V. Kal. Adaptive MPI. In In Proceedings of the 16th International Workshop on Languages and Compilers for Parallel Computing (LCPC 03), pages 306–322, 2003.
[12] Ahmad I., Ghafoor A. and Mehrotra K. "Performance Prediction of Distributed Load Balancing on Multicomputer Systems". ACM, 830-839, 1991.
[13] Bernard G., Steve D. and Simatic M. "A Survey of Load Sharing in Networks of Workstations". The British Computerm Society, The Institute of Electrical Engineers and IOP Publishing Ltd, 75-86,1993.
[14] Dandamudi S.P. "Sensitive Evaluation of Dynamic Load Sharing in Distributed System". IEEE, 1998.
[15] Biagio Cosenza "Efficient Distributed Load Balancing for Parallel Algorithms"pp20-27 ,Nov 2010 Universita degli studi disalerno,Italy
[16] Lin H. C. and Raghavendra C.S. "A Dynamic Load Balancing Policy with a Central Job Dispatcher (LBC)". IEEE,1992.
[17] Xu C. and Lau F.C.M. "Load Balancing in Parallel Computers: Theory and Practice". Kluwer Academic Press, 1997.
[18] OpenMP, The OpenMP ARB, http://www.openmp.org.
[19] D. Klepacki, Mixed-mode programming, T.J. Watson Research Center presentations, IBM, 1999, http://www.research.ibm.com/actc/Talks/DavidKlepacki/MixedMode/index.htm.