Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 6, June 2016, pg.383 – 390

TASK MIGRATION STRATEGIES FOR FAST RETRIEVAL OF INFORMATION FROM A MULTIPROCESSOR SERVER

Mukul Varshney[1], Shivani Garg[2], Aparajita Naiwal[3]
CSE, Sharda University, India
mukul.varshney@sharda.ac.in, shivani.garg@sharda.ac.in, aparajita.naiwal@sharad.ac.in

Abstract*— *Achieving Parallelism is now a necessity to improve the performance of computer system. One of the main Issues is how to effectively utilized parallel computer that have become increasingly complex. The parallel computer is one of the remarkable developments of methodology and technology in computer science in recent years Nowadays, the use and access of fast processors and also multi-processors has widely increased.[1]

Due to multiprocessor structure of the computer architecture, this computer has a capability to execute multiple instructions or multiple data simultaneously. [6] Research is active in the direction of developing new multiprocessor architectures and schedule the partitioned program on to it to achieve higher performance.

Multiprocessor system must be very efficient at solving problems that can be partitioned into tasks with uniform computation and communication patterns. However, there exists a large class of non-uniform problems with uneven and unpredictable computation and communication requirements.[3]

Parallel computers perform their computation by executing different computational tasks on a number of processors concurrently. During the execution of the parallel code, The processors within a parallel computer generally exchange information. This transfer of information occurs either in the form of messages sent by one processor to another or different parallel processors sharing a specified common memory resource within the parallel computer. The main idea of load balancing is to use multiple nodes to replace only one node, thereby it can enhance the computing power and reliability of the node at a low cost and time.

Therefore Dynamic load scheduling (DLB) schemes are needed to efficiently solve non-uniform problems on multiprocessor systems.Dynamic load scheduling (DLB)algorithm are required to efficiently solve this problems on multiprocessor systems. [1,6].

In this paper our effort is concentrated on study and evaluation of various dynamic load balancing strategies such as SID, RID, DEM, GM HBM etc[8,13]

Keywords*— *Task migration strategies, Multiprocessor system, knowledge over head, threshold, RID, HBM ,GM

I. INTRODUCTION

In past some years, the internet has become a necessity of every human life in any field. And internet has been a popular source of data access day by day. On the other hand user's expectations have increased in terms of the data should be downloaded in the very shortest possible time. For this many multiprocessor system has been developed for accessing data from the internet or through the local database.

Exploiting parallelism is now a necessity to improve the performance of computer systems one of the most important issues is how to effectively utilized parallel computers that have become increasingly complex to improve the performance. Such systems are constructed by different processor connected with communication link to operate in parallel with relatively low cost known as multi processor system.

The parallel computer is one of the remarkable developments of methodology and technology in computer science in recent years. Due to multiprocessor structure of the computer architecture, this computer has a capability to execute multiple instructions or multiple data simultaneously. The parallel computer not only provides support for efficient computation of mathematical, economical, industrial, and ecological problems but also aims new computer architecture beyond the traditional von Neumann type.

Multiprocessor system must be very efficient at solving problems that can be partitioned into tasks with uniform computation and communication patterns. However, there exists a large class of non-uniform problems with uneven and unpredictable computation and communication requirements. Therefore Dynamic load balancing (DLB) schemes are needed to efficiently solve non-uniform problems on multiprocessor systems.

The evaluation of multiprocessor architecture has been influenced by several factors such as

(i) speed up (ii) scalability and (iii) flexibility.

In this work we evaluate the performance of different Dynamic task Migration strategies on the basis two factors
(i) load imbalance factor and (ii) load balancing time using different multiprocessor architecture.

Task migration involves assigning the work of each processor in such a manner so that we minimize the execution time of program and increase the performance. Static load balancing strategies perform by predetermined policy without consideration of status of the system. On the other hands dynamic load balancing strategies makes the load balancing in accordance of status of the system such as no. Of jobs, arrival rate of jobs etc.
Sender Initiated Diffusion (SID)' is a highly distributed local approach which makes use of near-neighbor load information to apportion extra load from heavily loaded processors to underloaded neighbors in the system. Receiver Initiated Difision (RID) is the converse of the SID strategy, where underloaded processors requisition load from heavily loaded neighbors. Hierarchical Balancing Method (HBM) is an asynchronous, global, approach which organizes the system into a hierarchy of subsystems. Load balancing is initiated at the lowest levels in the hierarchy with small subsets of processors and ascends to the highest level which encompasses the entire system. Gradient Model (GM) [6,11] employs a gradient map of the proximities of underloaded processors in the system to guide the migration of tasks between overloaded and underloaded processors. Dimension Exchange Method (DEM) [13,15], is a global, fully synchronous, approach. Load balancing is performed in an iterative fashion by “folding” an N processor system into log N dimensions and balancing one dimension at a time.

II. TASK MIGRATION STRATEGY

Dynamic load balancing algorithms can be classified into three categories: Sender Initiated (SI), Receiver Initiated (RI), and Periodically Exchanged (PE). [2,4] In the SI algorithms, a heavily loaded node initiates the load balancing by requesting the load information of other nodes and sending out its jobs to the lightly loaded nodes. In the RI algorithms, a lightly loaded node initiates The load balancing by sending job request messages to other nodes and waiting for remote jobs. In periodical exchange sender and receiver both initiating load balancing in a fix time interval so that load of the over all system will be balanced.

III.MODEL FOR TASK MIGRATION STRATEGIES

We have developed a general model for dynamic task migration strategies. This model is organized as a four phase process[6,13]

(1) Processor load evaluation
(2) Load balancing profitability Determination
(3) Task migration strategy
(4) Task selection strategy

A. *Processor Load Evaluation*

- A load value is estimated for each processor in the system.
- These values are used as input to the load balancer to detect load imbalances and make load migration decisions.

B. *Load Balancing Profitability Determination:*

- The imbalance factor quantifies the degree of load imbalance within a processor domain.
- It is used as an estimate of potential speedup obtainable through load balancing
- It is weighed against the load balancing overhead to determine whether or not load balancing is profitable at that time.

C. Task Migration Strategy:

Sources and destinations for task migration are determined. Sources are notified of the quantity and destination of tasks for load balancing**.**

D. Task Selection Strategy:

Source processors select the most suitable tasks for efficient and effective load balancing and send them to the appropriate destinations**.**

- The first and fourth phases of the model are application dependent and purely distributed. Both of these phases can be executed independently on each individual processor.
- Our focus is on the Profitability Determination and Task Migration phases, the second and third phases, of the load balancing process
- As the program execution evolves, the inaccuracy of the task requirement estimates leads to unbalanced load distributions.
- The imbalance must be detected and measured (Phase 2) and an appropriate migration strategy devised to correct the imbalance (Phase *3).*
- During the Profitability Determination Phase a decision is made as to whether or not to invoke the load balancer.
- The load *imbalance factor* $\Phi(t)$ is an estimate of the potential speedup obtainable through load balancing at time t .
- It is defined as the difference between the maximum processor loads before and after load balancing, $L_{\max}$ *and* L_{bal} , respectively.

$$\Phi(t)= L_{\max} - L_{\text{bal}}$$

IV. TASK MIGRATION STRATEGIES

The schemes ,mention below, vary in the amount of processing and communication overhead and in the degree of knowledge used in making balancing decisions.

The following five **TMS** strategies [4,14]are designed to support highly parallel systems.

1. *Sender Initiated Diffusion (SID)*
2. *Receiver Initiated Diffusion (RID)*
3. *Hierarchical Balancing Method (HBM)*
4. *Gradient Model (GM)*
5. *Dimension Exchange Method (DEM)*

A. Sender Initiated Diffusion (SID)

The SID strategy is a, local, near-neighbor *diffusion* approach which employs overlapping balancing domains to achieve global balancing. for an N processor system with a total system load L, a diffusion approach, such as the SID strategy, will cause each processor's load to converge to L/N. [1,7,8]

Balancing is performed by each processor whenever it receives a load update message from a neighbor indicating that the neighbors load, 1_i<Ideal Load , where *Ideal Load* is a preset threshold. Each processor is limited to load information from within its own domain, which consists of itself and its immediate neighbors.

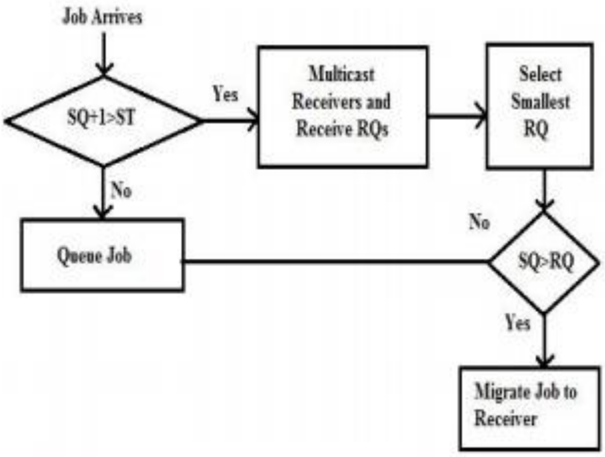

Fig. 1 Flowchart of Sender –Initiated Algorithm

SQ- Queue length of sender
RQ- Queue length of receiver

B. Receiver Initiated Diffusion (RID)

(1) First, the balancing process is initiated by any processor whose load drops below a pre specified threshold ***(L_{Low}) . [7,8]***

(2) Second, upon receipt of a load request, a processor will fulfill the request only up to an amount equal to half of its current load

(3) The RID strategy differs from its counterpart **SID** in the task migration phase. Here, an underloaded processor first sends out requests for load and then receives acknowledgment for each request

C. Hierarchical Balancing Method (HBM)

- It is an asynchronous global, approach which organizes the system into a hierarchy of subsystems. [1,7]
- Load balancing is initiated at the lowest levels in the hierarchy with small subsets of processors and ascends to the highest level which encompasses the entire system.
- Specific processors are designated to control the balancing operations at different levels of the hierarchy.

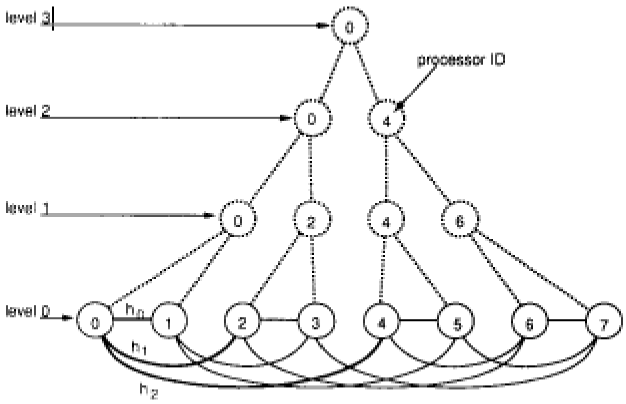

Fig. 2 Hierarchical organization of an eight-processor system with hypercube interconnections, where h_k is the connection to the neighbor at the kth level.The processor IDs at intermediate nodes in the tree represent those processors delegated to manage the balancing of corresponding lower-level domains.

- The hierarchical balancing scheme functions asynchronously.
- The balancing process is triggered at different levels in the hierarchy by the receipt of load update messages indicating an imbalance between lower level domains.
- All load levels are initialized with each processor sending its load information up the tree.

D. The Gradient Model (GM)

- The gradient model [5,13] is a demand driven approach .
- The basic concept is that underloaded processors inform other processors in the system of their state, and overloaded processors respond by sending a portion of their load to the nearest lightly loaded processor in the system.
- This model employs a gradient map of the proximities of underloaded processors in the system to guide the migration of tasks between overloaded and underloaded processors.

The resulting effect is a form of relaxation where tasks migrating through the system are guided by the proximity gradient and gravitate towards underloaded points. The scheme is based on two threshold parameters: the *Low-Water-Mark (LWM)* and the *High- Water-Mark (HWM).* A processor's state is considered light if its load is below the LWM, heavy if above the HWM**,** and moderate otherwise.

E. The Dimension Exchange Method (DEM)

The DEM strategy [15] is similar to the HBM scheme in that small domains are balanced first and these then combine to form larger domains until ultimately the entire system is balanced. This differs from the HBM scheme in that it is a synchronized approach. The DEM strategy was conceptually designed for a hypercube system but may be applied to other topologies with some modifications. In the case of an N processor hypercube configuration, balancing is performed iteratively in each of the logN dimensions. All processor pairs in the first dimension, those processors whose addresses differ in only the least significant bit, balance the load between themselves. Next, all processor pairs in the second dimension balance the load between themselves, and so forth, until each processor has balanced its load with each of its neighbors.

V. COMPARISON ANALYSIS OF ALGORITHMS

We observed from [6] that the Rendez-vous algorithm is convergent. On a „n" processor system, at most log2(n) are needed to reach a steady state where each processor has got the average load of the system as shown in table 1. Communication cost is taken into consideration for accurately evaluating the behavior of redistribution.

TABLE I
ALGORITHM CONVERGENCE & COMMUNICATION COST

Algorithm	Speed of Convergence	Communication Cost	Communication Policy
Rendez-vous	~log2(n)	O(nlogn)cg	Global
Tiling	>n/2	O(n)cn	Local
X-Tiling	~log2(n)	O(logn)cu	Uniform
Sliding	<n	O(n)cn	Local

A. Implementation Results

All five schemes were implemented on a hypercube multiprocessor architecture. The strategies were first tested and analyzed using artificially generated data consisting of a set of tasks with random computational requirements executed as busy loop.

No of Task	RID	SID	GM	DEM	HBM
16	4	0	4	4	4
32	4	2	4	4	4
64	4	5	5	5	5
128	5	5	5	5	10
256	5	6	6	10	11
512	6	6	6	11	11
1024	6	7	6	15	15
2048	10	10	10	16	16
4096	15	15	21	21	21
8192	20	28	37	43	43
16384	37	54	60	78	70
32768	78	94	114	147	146
65536	170	190	220	270	270
131072	433	341	411	451	448
262144	702	702	800	921	921
524288	1351	1471	1644	2021	2007
1048576	2612	7282	3385	3620	3434

TABLE I- Artificially Generated Task (task vs time)

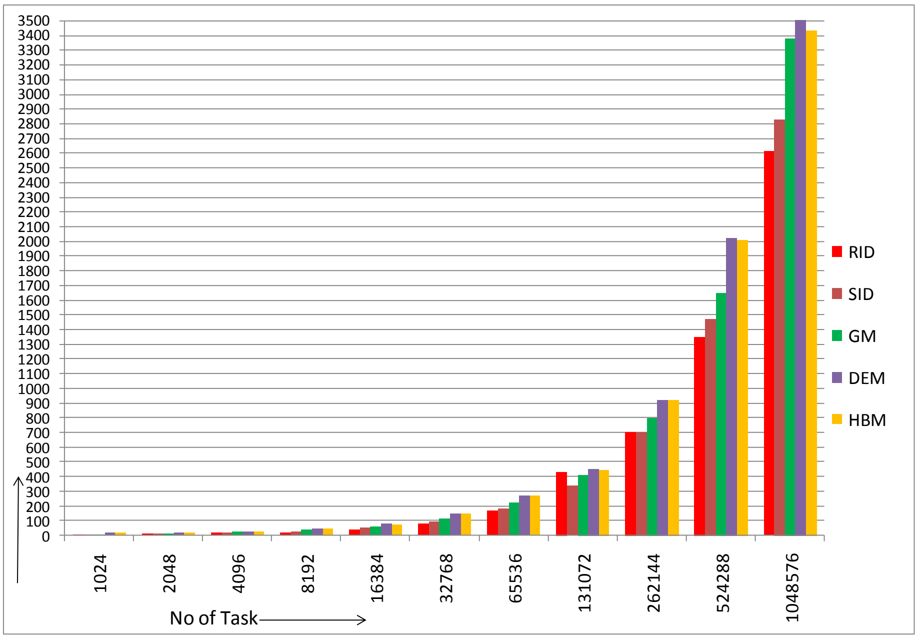

Fig. 3 Task Vs Time (Comparative Study)

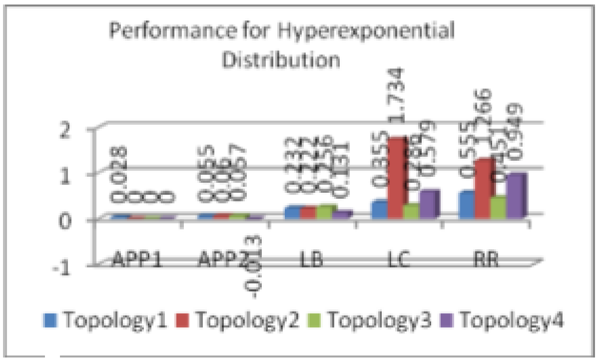

Fig 4: Performance for Hyper exponential Distribution

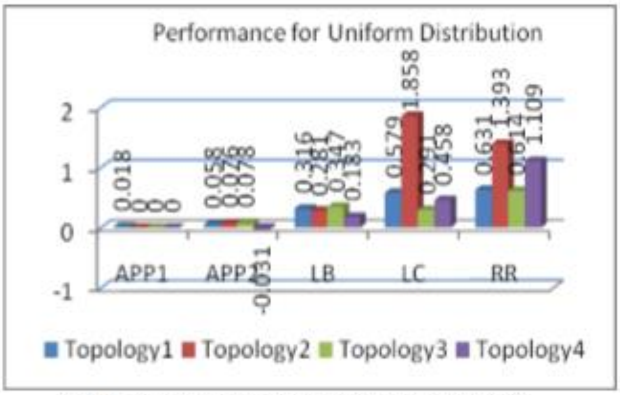

Fig 5: Performance for Uniform Distribution

VI. CONCLUSIONS

Here task migration 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.

Five dynamic task migration strategies designed to support highly parallel systems have been presented and compared. The different strategies exemplify some of the main issues and tradeoffs that exist in dynamic load balancing, specifically in reference to highly parallel systems. Two major issues, that of load balancing overhead and the degree of knowledge used in balancing decisions were discussed. Also considered were, the concept of balancing domains, the aging of information, and the form of balancing initiation.

Of the five strategies proposed, the DEM strategy tended to outperform the rest for all granularities. The efficiency of the DEM and the HBM strategies, depends heavily on the system interconnection topology. The hypercube topology is ideally suited to match these two strategies communication dependencies. Further- more, the system sizes tested were very small in the context of highly parallel systems. The RID strategy, on the other hand, is easily ported to simpler topologies, and can scale gracefully for larger systems. Finally, for a wider variety of applications, exhibiting local communication dependencies between tasks, the RID scheme is able to maintain task locality. Therefore, since its performance was shown to be comparable to those of the DEM and HBM approaches, the RID strategy may be best suited for a broader range of systems supporting a large variety of applications.

REFERENCES

[1] A.Samad, M.Q. Rafiq and Omar Farooq "A novel algorithim for fasT retrival of information from a multiprocessor server" Proc. Intl. on parallel and distributed system, U.K. Feb 20 to feb 22, 2008
[2] Z. Zeng and B.Veeravalli "Design and Performance of Queue and Rate Adjustment Dynamic load Balancing Polices for Distributed Network". IEEE trans. On computer, vol 55 ,no. 11, pp. 1410-1422,nov 2006.
[3] Rafiq, M. Q.; Padam kumar and gupta J. P. "A Novel Tree –structured multiprocessor Network." *Proc. Int'l conf on robotics vision and parallel processing for automation*, Malaysia, VOL. 2 ,pp 576,585,1999
[4] Fahd Alharbi, "Simple Scheduling Algorithm with Load Balancing for Grid Computing", Asian Transactions on Computers, Volume 2, Issue 2, May 2012.
[5] Junwei Cao, Daniel P. Spooner, Stephen A Jarvis, Graham R. Nudd, "Grid Load Balancing UsingIntelligent Agents", ACM, Volume 21, Issue 1, January 2005.
[6] Mohammad Haroon, Mohammad Husain, "Analysis of a Dynamic Load Balancing in Multiprocessor System", International Journal of Computer Science engineering and Information Technology Research, Volume 3, Issue 1, March 2013.
[7] Abhijit A. Rajguru, S.S. Apte, "A Comparative Performance Analysis of Load Balancing Algorithms in Distributed System using Qualitative Parameters", International Journal of recent Technology and Engineering, Volume 1, Issue 3, Au1gust 2012.
[8] Urjashree Patil, Rajashree Shedge, "Improved Hybrid Dynamic Load Balancing Algorithm for Distributed Environment", International Journal of Scientific and Research Publications, Volume 3, Issue 3, March 2013.
[9] Robson Eduardo De Grande, Azzedine Boukerche, and Hussam Mohamed Soliman Ramadan, "Measuring Communication Delay for Dynamic Balancing Strategies of Distributed Virtual Simulations", IEEE Transactions on Instrumentation and Measurement, Vol.60, No.11, November 2011.
[10] Dennis Roubos and Sandjai Bhulai, "Session-Level Load Balancing for High-Dimensional Systems", IEEE Transactions on Automatic Control, Vol.54, No.8 August 2009.
[11] Xiao Qin, Hong Jiang, Adam Manzanares, Xiaojun Ruan, Shu Yin, "Communication-Aware Load Balancing for Parallel Applications on Clusters", IEEE Transaction on Computers, Vol.59, No.1, January 2010.
[12] Tarek Helmy, Fahd S. Al-Otaibi, "Dynamic Load-Balancing Based on a Coordinator and Backup Automatic

Election in Distributed Systems", International Journal of Computing & Information Sciences, Vol. 9, No. 1, pp.19 27, ISSN: 1708-0460 (print) - 1708-0479 (online), April 2011.

[13] Jie Chang, Wen‟an Zhou, Junde Song, Zhiqi Lin, "Scheduling Algorithm of Load Balancing Based on Dynamic Policies", Sixth International Conference on Networking and Services (ICNS), IEEE Xplore Digital Library, Cancun, ISBN: 978-1-4244-5927-8, May 6, 2010

[14]Sun Nian, Liang Guangmin, "Dynamic Load Balancing Algorithm for MPI Parallel Computing", International Conference on New Trends in Information and Service Science (NISS), IEEE, Gyeongju, Korea, pp.95–99, June 30 - July 2, 2009.

[15] Yongzhi Zhu, Jing Guo, Yanling Wang, "Study on Dynamic Load Balancing Algorithm Based on MPICH", World Congress on Software Engineering, IEEE, Xiamen, China, May 19-21, 2009.

