Efficiency Measurement for Effective Stress Management in Heterogeneous 2-D Mesh Processor

Arashdeep Singh
Assistant. Prof.
LLRIET, Moga

Sunny Behal
Assistant. Prof.
SBSSTC, Ferozepur

Ankit Arora
Assistant Prof.
LLRIET, Moga

ABSTRACT
Multi-Processor interconnection with varying speed is a great attempt in massive parallel processors. Such types of distributed cluster along with heterogeneous behavior will requires vast amount of scheduling efforts. Complexity increases as scheduler has to detect dynamic characteristics of the processors. Parallel schedulers are implemented in cluster technology for job assignment and placement. Further, core processor technology will provide a greater endeavor for load balancing. This research covers heterogeneous multi-processors with 2-D mesh interconnection mapped to cube oriented memory mesh for job allocation and distribution. The job distribution will be based upon processor cycle speed. A two dimensional job slice is build, which in later stages along with many other slices overlapped to exhibit memory cube.

General Terms
Heterogeneous Processor Mesh, Memory Slice, Scheduling, Workload Distribution, Efficiency Degradation.

Keywords
Load Steadiness, Processor Cycle Speed, Efficiency Measurements, Dynamic Distribution Characteristics, Stress Management.

1. INTRODUCTION
Integrated hardware design for multiprocessor systems covers complex algorithmic structures and varied number of processing elements. Parallel hardware integration starts due to the uniprocessor limited speed and modern demands of real time computation intensive workload. Chip level computing is one way to achieve the parallel computation leading towards multi-core integration[5]. Application scenarios over multi-processor systems must incorporate parallel behavior. Such designs cover multiple processing application modules/threads which can be distributed and executed without any interference among them. Although, scheduling jobs or their distribution policy implementation is a very tricky task. An efficient scheduling mechanisms will provides an effective utilization of multiprocessor hardware with major aim is to exploit the machine efficiency. On board multiprocessor hardware usually covers homogeneous processor interconnection. Intel provides RISC based multiprocessor Intel paragon IPSC/860 with 2-D mesh architecture covers thousands of processor nodes [6]. Similarly CDC 6600 from control Data Corporation operates with 128 processor nodes on hypercube. On board heterogeneous multiprocessor are under consideration. Dynamic scheduling against heterogeneous hardware is a way to characterized application workload. This research deals with heterogeneous processor simulation covering job distribution with assumption that allocation well approximates after estimating processor frequency speed [9][12].

2. OBJECTIVE
Intention to develop multiprocessor simulation along with heterogeneity is to normalize the workload distribution, rather than to cure load balancing aspects after distribution has been takes place [1]. Such efforts will consume much of the processor time on the behalf of the task adjustment and placement. Further the study behind the current experiment is to create mesh based processor interconnection mapped to 2-D memory job slice. Each two dimensional job slice arranged according to processor efficiency estimation by measuring present processor load and frequency cycles. Parallel allocation takes place with overlapped multiple memory slices.

3. RELATED WORK
Existing literature about frequency based experiments illustrates uniform single core processors where each processor carries their individual cycle speed covering real time workload characterization [4]. Based upon metric measurements the workload will be distributed to intended processor best suited to current application. Further the experiments cover multi-core multiprocessors [10], where each core speed will be considered as basic scheduling factor in job distribution. Ultimate idea of core technology development is to balance the processor mass (in terms of workload) over their underlying sub-cores. Further the experiment in this research will generate two dimensional job slices overlapped with one another to produce cube like structure as described earlier, where each slice mapped downwardly over to processor mesh.

4. STRESS MANAGEMENT
Processor Stress management integrates processor utilization per unit time, i.e. how frequently the processor consumption reach upto 100% in one unit time or other means efficiency degradation. The idea behind this methodology is to maintain system's steady state. This technique estimates stress metric for load sharing aspects in multi-processor system for each and every processor in the pool [2]. The SM factor will offer processor indexes which are most frequently loaded in their execution life cycle. Further the load balancer performs load sharing to manage processor stress (in terms of workload) to no. of available processors having low SM value. This technique can be used in either homogeneous or heterogeneous system interconnection. Stress in terms of 100% utilization is a valuable metric for dynamic load balancing in multi-processor scheduling [3][11]. Although the technique follows load adjustment guidelines to maintain system's safe state.

5. MEASURING & MANAGING STRESS
Processor load basically helps in estimating processor stress. One distribution approach defined above based upon processor speed and existing workload measurements basically a stress management policy, estimating no. of cycles

required to complete exiting and present workload quickly. Other approach defined above as stress management-that how frequent a processor reaches its utilization upto 100% in one unit time. One solution to this problem is that find the SM value of each processor and find the candidate processor which requires load balancing. Now evaluate the processor ready queue and processes which are currently in running state. Find the process which consumes frequently large processor cycle pool. The Parallel workload analyzer interpret this job for parallel behavior, if the job having parallel modules can be executed simultaneously, then such parallel modules can be distributed to other processors, this is requires for high priority tasks for which immediate execution is also the constraint. In this way the processor stress will also be in steady state as well as job modules worked in parallel fashion. If job execution structure does not correspond to parallel behavior then the other processes running or ready that consumes equivalent no. of processor cycles/time in comparison to that process. Such processes can be distributed to other processors having low SM value. Although context switching is more in later case, but low effort in workload partitioning. In the former case workload partitioning consumes valuable processing time but context switching comprises low efforts. One another approach is to measure efficiency degradation i.e. time since last completed process by each processor. This will be measured as no. of CPU cycle consumed (NCC). The NCC Metric value conversion further takes place as no. of seconds elapsed (NSE) respective to processor frequency. NSE value greater than 1 will specifies the stress beginning and may reach to further degradation. NSE signifies even after consumption of too many cycles no rise in throughput

$$NCC = CURR\_CT - T\_LCP$$

$$NSE = NCC * \frac{1}{P\_Freq}$$

If NSE > 1 then

Requires load sharing/stress management

End if

Such situation may block the processors for further execution. The solution to these overburden situation could be defined as- Preempt the currently running process from that processor. Evaluate the parallel behavior if concurrency found between modules, distribute such modules to stress free processors that can handle. If the process behavior is standalone then find other processor after evaluating processor speed, its existing load along with current process load. Utilization of processor time in terms of actual processing and computation. This is because earlier studies behind computation theory describes much of the processor time wasted by currently running processes in terms of unuseful work like data transfers, memory management, managing process address space and context switching. So estimates ratio between computation cycle and control cycles. Any processor having more no. of control cycles elapsed than computations from very long duration. This will reduce the throughput, excessive cycles length is wasted and efficiency is degraded. Solution to this condition is to elaborate processor coupling in terms of communication hierarchy and performing transmission of module logic rather than obtaining data to intended processor. This is because bulky data transfer from one processor memory unit to another processor memory unit requires heavy time volume, so rather than obtaining data from neighboring processor, code/logical statements can be transferred to that processor. In this way computation can be performed at same place. Data transmission is reduced as well processor load sharing is performed. This type of situation occurs where distributed data takes place.

6. PROPOSED PARALLEL FLOW

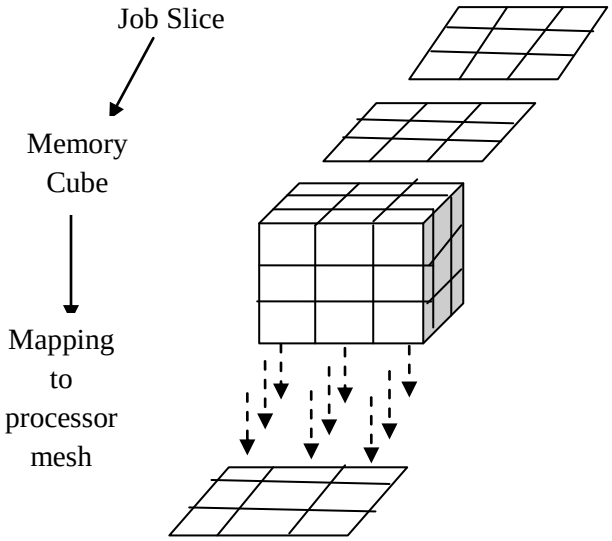

Fig 1: Processor Memory Map

7. SIMULATION PRACTICE

Simulation structure covers multi-threaded environment following heterogeneous processor mesh and incoming job queue. The job slice arranged with respective to the processor speed, existing load and current job workload (in terms of cycle required) described in section-8. The job slice then arranged like a cube overlapped with each other mapped to processor mesh. Multi-threaded synchronous interconnection is build with visual basic 6.0 language tools and libraries. Nine processors are organized as mesh. Further the simulation view will be described in figure-8.

8. DISTRIBUTION APPROACH

Distribution approach comprises relationship among processor frequency cycle speed and computation. Distribution takes place to the processor consuming minimum amount of cycles for its current and new incoming workload ready to distribute. Due to the variation in speed and distributed workload this may be possible that a low speed processor may quickly complete the new incoming workload. Following is the metric used as cycle based scheduling.

$$T\_OnePCycle = \frac{1}{P\_Freq}$$

For each *Pth* Processor where p=1 to n

w=0

For I=1 to RQueue[p].length

w = w +Job_Wload [I]

end for

P_ Exist_wload[p] = w

End for

For each *Pth* Processor where p=1 to n

T_Comp_Cycle= P_Exist_Wload[p] + New_JobWorkload
*Comp_Time[p]=T_comp_Cycle * (1/Freq[p])*
End for

PIndex = Min (comp_Time)

Allocate(New_job , PIndex)

9. RESULTS AND DISCUSSIONS

Underlying results from simulation practices illustrates controlled form of load distribution with consistency. The ultimate objectivity concepts will be elaborated with in the Illustrations. Simulation results are carried out periodically at some time barrier point. Although results provides approximately 90% stability control. But gives benefits over load balancing which is performed after load assignment and corresponds to the load adjustment policies. Further the results exhibits their intended performance.

Table-1 Load Distribution at Time Barrier 18

Processor_Freq	Workload Cycle
P-1 GHZ	1221
P-1.7 GHZ	3279
P-2.4 GHZ	4265
P-3.2 GHZ	9105
P-4.0GHZ	9571
P-4.8 GHZ	11323
P-5.6 GHZ	15188
P-6.2 GHZ	18081
P-7.0 GHZ	18058

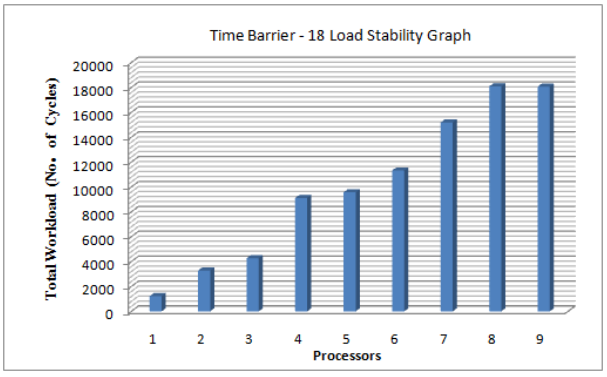

Fig 2: Load Stability Graph-1

Table-2 Load Distribution at Time Barrier 25

Processor_Freq	Workload Cycle
P-1 GHZ	2097
P-1.7 GHZ	4761
P-2.4 GHZ	5204
P-3.2 GHZ	9430
P-4.0GHZ	12625
P-4.8 GHZ	14056
P-5.6 GHZ	16492
P-6.2 GHZ	19040
P-7.0 GHZ	22750

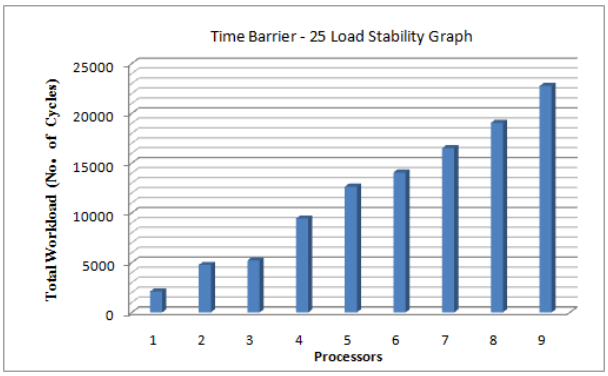

Fig 3: Load Stability Graph-2

Table-3 Load Distribution at Time Barrier 35

Processor_Freq	Workload Cycle
P-1 GHZ	3200
P-1.7 GHZ	4583
P-2.4 GHZ	7428
P-3.2 GHZ	9991
P-4.0GHZ	11469
P-4.8 GHZ	16336
P-5.6 GHZ	16968
P-6.2 GHZ	20823
P-7.0 GHZ	24973

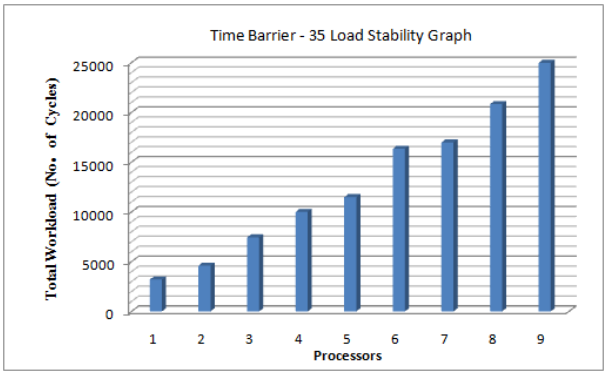

Fig 4: Load Stability Graph-3

Table-4 Load Distribution at Time Barrier 45

Processor_Freq	Workload Cycle
P-1 GHZ	3488
P-1.7 GHZ	6452
P-2.4 GHZ	8837
P-3.2 GHZ	11956

P-4.0GHZ	15479
P-4.8 GHZ	19889
P-5.6 GHZ	23211
P-6.2 GHZ	24837
P-7.0 GHZ	31021

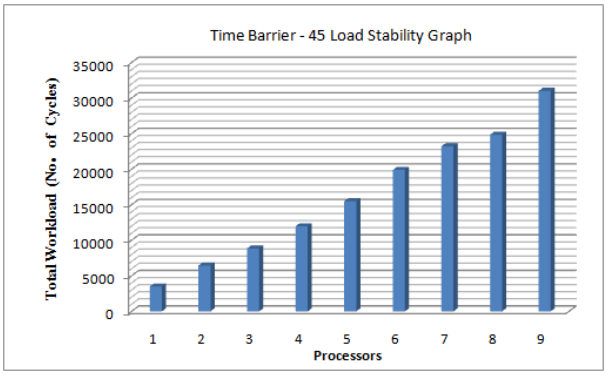

Fig 5: Load Stability Graph-4

Table-5 Load Distribution at Time Barrier 85

Processor_Freq	Workload Cycle
P-1 GHZ	4369
P-1.7 GHZ	8996
P-2.4 GHZ	12428
P-3.2 GHZ	16238
P-4.0GHZ	22872
P-4.8 GHZ	26420
P-5.6 GHZ	32343
P-6.2 GHZ	35061
P-7.0 GHZ	37342

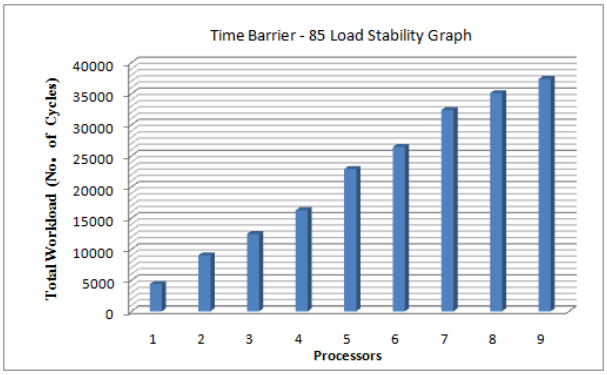

Fig 6: Load Stability Graph-5

Table-6 Load Distribution at Time Barrier 110

Processor_Freq	Workload Cycle
P-1 GHZ	6163
P-1.7 GHZ	10918
P-2.4 GHZ	15828
P-3.2 GHZ	19985
P-4.0GHZ	25325
P-4.8 GHZ	32724
P-5.6 GHZ	39643
P-6.2 GHZ	41675
P-7.0 GHZ	46119

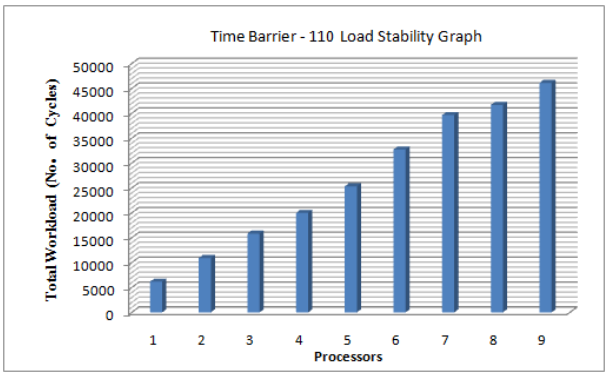

Fig 7: Load Stability Graph-6

10. CONCLUSION & FUTURE WORK

Results concluded from simulation exhibits a great effort for load stability, consistent distribution is performed throughout the whole process. Examination above captured is based upon different simulation execution scenarios. Illustration produced with respect to the beginning, intermediate and after very long processed time barrier. All of the graphs characterize workload distribution in a steady state. Although not completely balanced but stabled results up to large extent. Simulation long duration execution will show more and more consistent state, more balanced distribution will takes place. Future work of this research further explores stress management approaches with dynamic distribution and workload characterization. More advance simulation practices and theory will lead to the effective implementation structure.

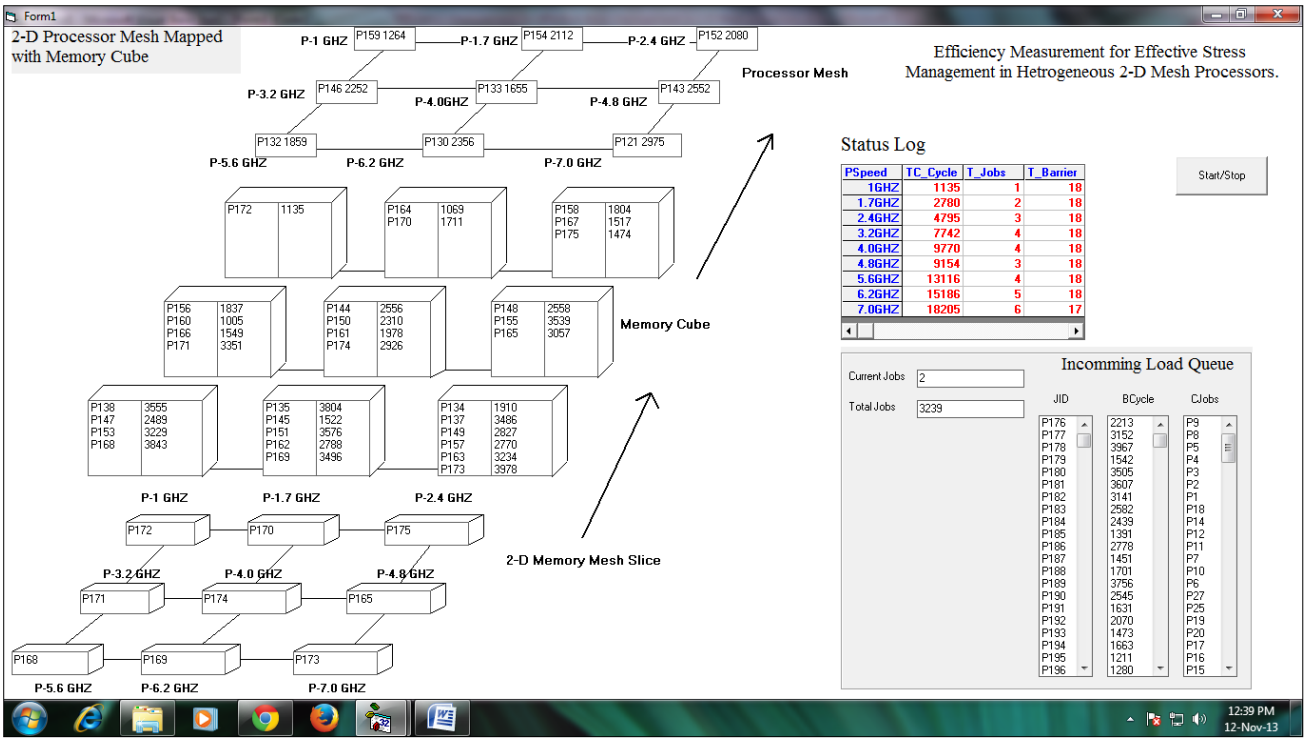

Fig 8: Simulated View (Memory mapped over 2-D Mesh)

11. REFERENCES

[1] Mendel R, 1995 Complete Computer System Simulation:The SimOS approach IEEE Parallel and Distributed Computing.

[2] Orleans, L.F IEEE 2007 Fair Load-Balancing on Parallel System, International Conference on parallel processing

[3] Chhabra, A. and Singh, G. 2009 Simulated Performance Analysis of Multiprocessor Dynamic Space-Sharing Scheduling policy

[4] Bobrek A., Paul M. 2010 IEEE Stochastic Contention for Single-Chip Heterogeneous Multiprocessor.

[5] Varbanescu, A. 2010. On the effective parallel programming on multi-core processors. Universities POLITEHNICA Bucuresti Romania Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[6] Kot, A. IEEE 2011 The Evaluation of an Effective Out-of-Core Run-Time System in the Context of Parallel Mesh Generation. Parallel and distributed symposium IPDS.

[7] Marowka, A. J. 2011 Back to thin-core massively parallel processors. Bar-llan University, Israel. IEEE computer society.

[8] Chhabra, A. and Arora, A. 2011. Cluster based performance evaluation of Run-length image compression.

[9] Lokhande, M. , Atique, M. , 2012. Real-Time Scheduling for Parallel Task Models on Multi-Core Processors-A Critical review".

[10] Hager, G. and Wellein, G. 2012 Ingredients for good parallel performance multi-core based systems spring sim, Alexander university Orlando USA.

[11] Srinivasa Rao, P. and Govardhan, 2013 A. Dynamic Load Balancing With Central Monitoring of Distributed Job Processing System. Foundation of Computer Science New York.

[12] Arora, A., Arora, A. 2013. Scheduling Simulations: An Experimental Approach to Time-Sharing Multiprocessor Scheduling Schemes.