ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

CPU and Memory Based Cluster Load Balancing for Jobs with Bursts of Loads

D. VIJAYA KRISHNA [1], G. SANTHOSH [2], Dr. P. SAMMULAL [3]

Department of Computer Science and Engineering, JNTUH College of Engineering, Jagtial, India [1]
Department of Computer Science and Engineering, JNTUH College of Engineering, Jagtial, India [2]
Department of Computer Science and Engineering, JNTUH College of Engineering, Jagtial, India [3]

Abstract--High performance computing is possible through cluster computing where load is shared by a pool of networked servers. The cluster computing can serve jobs which are memory and CPU-intensive in batch processing model. Existing systems focused on balancing CPU loads ignoring page faults and I/O operations. Recently Xiao et al. explored policies to share loads that also consider memory usage in addition to load balancing in clusters. The policies can withstand predictable and unpredictable bursts in demands. In this paper we implement those policies and perform trace-driven simulations. The empirical results revealed that the proposed system is capable of capturing dynamic memory access patterns in the presence of uncertain workloads. Thus the policies are able to improve the performance in terms of execution of jobs by optimally utilizing memory and CPU resources.

Index Terms – Cluster computing, load balancing, uncertain workloads, trace-driven simulations

I. INTRODUCTION

In this paper we considered a set of servers that work together in the form of a cluster. As part of cluster computing we consider the problem of load balancing in the presence of unpredictable workloads in terms of memory access patterns and CPU usage. The aim of the paper is to provide high-performance computing for jobs which are CPU and memory intensive. Using distributed resources such as memory, CPU, I/Os and so on, we implement load sharing policies that improve overall performance of the clusters in terms of load balancing and improving throughput of the network. Many existing load balancing schemes focused on CPU only [1], [2], [3], [4], [5], [6], and [15]. Moreover these schemes assume that the participant servers in the cluster have sufficient memory available. These schemes improved the performance of the clusters in terms of load balancing. With rapid development in CPU chips and other technologies, there are fewer problems with CPU but memory problem has to be focused. The memory related issues are ignored in existing systems.

In this paper our main focus in both on CPU and also memory access patterns. This is because the speed of processors dramatically increased with technologies like RISC and VLSI technologies. Therefore there is speed gap between the RAM and processor. When jobs are executed in distributed environment in clusters, it is essential now to consider memory as it has lagged behind in development when the new technologies are considered with respect to improving CPU speed. Memory access is very important for every application that runs in any operating environment. The demand for data access is increased dramatically so as to serve information needs of the people. The rapid growth of networking and Internet also contributed to this factor. Page faults are the common problem being witnessed. Memory miss latency is more than 1000 times when compared to memory hit [7], [8]. Therefore it is very important to minimize the page faults. This can be achieved by balancing memory usage along with CPU load balancing. When any node's memory is not in use, its memory is used by other nodes to balance load. This kind of requirement is actually the need of the hour.

In this paper we implement load sharing policies that focused on both CPU load balancing and also balancing of memory access patterns in order to improve the performance of execution of jobs in the clusters. The overall throughput of the network is dramatically improved with this consideration. The rest of the paper is structured as follows. Section II reviews literature which is relevant to load balancing in clusters. Section III provides the proposed load balancing policies. Section IV presents the experimental results while section V concludes the paper.

II. RELATED WORKS

Apart from load balancing in terms of CPU, there are related works that focused on memory resource for load balancing [9], [10], [11], and [12]. The early study made in [1] shows that memory of nodes that are not idle is used for load balancing. However, when compared with CPU based

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

polices, this study could not prove effective as the CPU cycles were costly when compared to memory that time. However, the rapid pace in which the CPU technologies grow later pushed the memory resource to lag behind. This is the reason why there is the need for the balancing the usage of memory as well. To attempt page faults overhead, global memory concept is used in [13], and [10]. They used remote paging techniques in order to achieve global memory concept. DoDo [12] was introduced in order to improve throughput of clusters by using memory efficiently. The global memory is actually is in the local systems. The memory which is present in all the participating nodes in the cluster becomes global memory that cluster. When the memory which if free is greater than certain threshold, it is considered for load balancing [11]. In this paper instead of considering only memory, we consider both CPU and memory both for load balancing which improves performance of the network.

In this paper we focus on the problem of predictable memory access and also unpredictable memory access patterns. In other words our study focused on the load sharing with unknown memory demands. In order to achieve load balancing in such scenarios, we implemented load sharing schemes that made use of both CPU and memory which is globally available for load balancing.

III. PROPOSED LOAD BALANCING POLICIES

The proposed load balancing policies are aimed at improving the overall performance of the clusters in terms of processing or executing jobs effectively. We consider the problem of unknown workloads that cause problems in processing. It does lower the throughput of the network. To overcome this problem we consider both the CPU and memory sharing globally so as to improve the execution process in the cluster. We are making the following assumptions.

- Global load index is maintained by each node in the cluster. This information gets updated periodically.
- The load balancing scheme determines the node that has to process the jobs.
- During job executions, the page faults are uniformly distributed.
- The memory load of given job is 40% of the memory that has been requested for the job.

We use the load index available at each node for load balancing in terms of CPU usage and also the memory usage. The load index is computed as follows.

$$Index_{hp}(j)(L_j, ML_j) = \begin{cases} L_j, & ML_j < RAM_j, \\ CT_j, & ML_j \geq RAM_j. \end{cases}$$

Where j represents a node. RAMj represents the amount of available RAM on given node. MLj represents the memory load out of the total memory.

CPU Based Load Sharing

In each node of the cluster, the load index indicates the length of CPU waiting queue Lj. Each CPU has its threshold represented as CTj which is the maximum number of jobs. The CPU threshold is verified every time when request is made and decision is made to make use of this node or not. If this node does not the satisfy the condition Lj < CTj the load is balanced by considering a remote node which satisfies the condition as it has less load in terms of CPU.

Memory Based Load Balancing

We do not use Lj. Instead we use memory load in order to represent load index (MLj). For every request that arrives a condition is verified that is MLj<RAMj. If the condition is satisfied, the job gets executed in the local node otherwise it is balanced by giving it to a remote node that can satisfy the condition. Here is an important point considered. The memory based load balancing is used when there is no sufficient memory. When the memory is very sufficient then it is ignored and CPU based load balancing is applied. Load sharing is done with unknown memory demands also. This is achieved by analyzing workload traces. We also use a page faults model in order to characterize it in the solution. The paging rate model is computed as follows.

$$R_p(Q_o) = R_{max} - \frac{\alpha}{\beta Q_o^2 + \gamma Q_o + \delta}$$

Memory Centric Load Sharing Scheme

While the load sharing system is on, the following algorithm is executed in order to achieve memory centric load sharing.

```
while (Qo(j) < MT(j) and Njob(j) < CT(j))
        enjoy page-fault-free job executions and
        allow new job submissions;
block job submissions;
if (Qo(j) ≥ MT(j))
   if (Qo(j) ≤ 0)
      continue enjoying page-fault-free executions of
      existing jobs;
      if there are arrival jobs to the machine
         while NOT find_a_suitable_machine_for_
         remote_submission
               continue to block job submissions;
            submit the first arrival job to the
            suitable_machine;
          else
            while NOT find_a_suitable_machine_for_
            migration
                  continue local executions;
            migrate the identified_job to the
            suitable_machine;
            Njob(j) = Njob(j) − 1;
```

Fig. 1 – Memory centric load sharing

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

As can be seen in figure 1, the memory centric load sharing mechanism performs load balancing based on the threshold and condition explored earlier. When the workloads exceed thresholds with respect to page fault rate based, working set size based, age based, and CPU memory or I/O based decisions are made accordingly to balance load to maximize throughput of the network. More technical details can be found in [14].

Workload Traces

Workload traces are collected from the datasets provided by Los Alamos National Lab. The traces contain details like number of requests, nodes, load of the nodes in terms of memory and CPU, the available resources in the nodes of cluster and so on. With respect to jobs, the trace contains arrival time, arrival node, requested memory size, requested CPU time.

IV. EXPERIMENTAL RESULTS

The trace based experiments are made in order to test the efficiency of the proposed sharing policies. The experiments are made in terms of average page fault rate, paging time reduction, memory threshold, paging time, memory threshold vs. migration time, and memory threshold vs. queuing time.

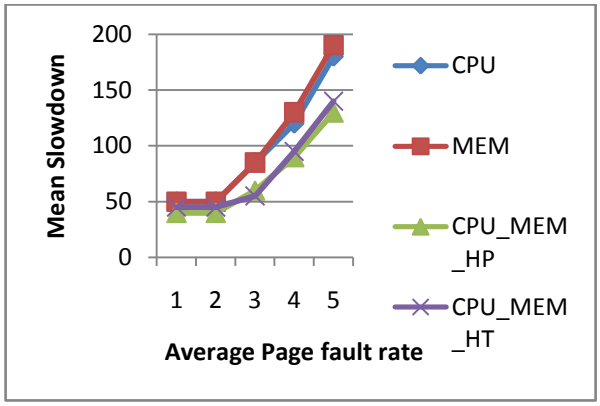

Fig. 2 – Average page fault rate vs. mean showdown (May trace)

As can be seen in figure 2, it shows the average fault rate and mean show down for load balancing policies like CPU, MEM, CPU_MEM_HP and CPU_MEM_HT for May trace.

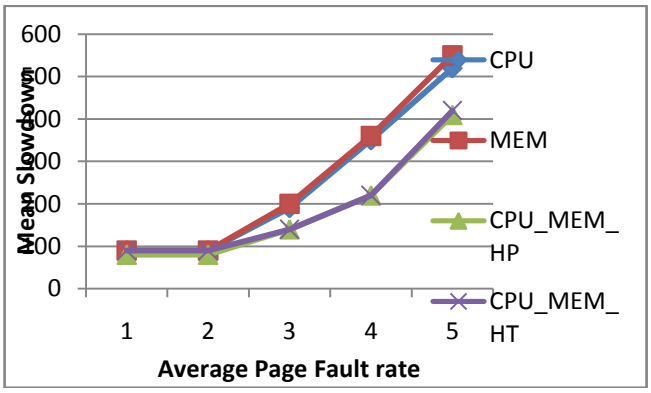

Fig. 3 – Average page fault rate vs. mean show down (June trace)

As can be seen in figure 3, it shows the average fault rate and mean show down for load balancing policies like CPU, MEM, CPU_MEM_HP and CPU_MEM_HT for June trace.

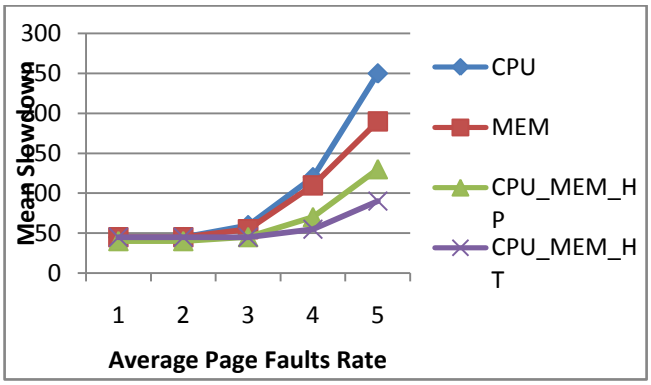

Fig. 4 – Average page fault rate vs. mean showdown (July trace)

As can be seen in figure 4, it shows the average fault rate and mean show down for load balancing policies like CPU, MEM, CPU_MEM_HP and CPU_MEM_HT for July trace.

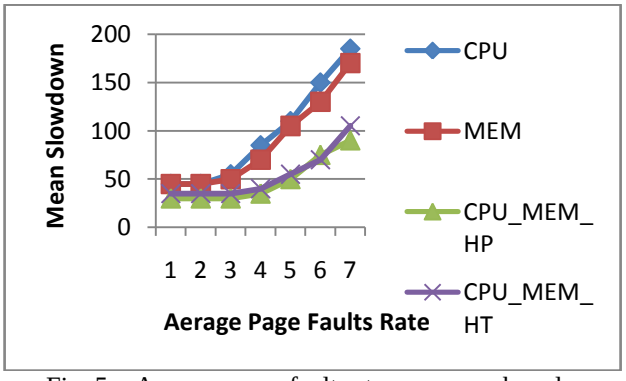

Fig. 5 – Average page fault rate vs. mean showdown (August trace)

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

As can be seen in figure 2, it shows the average fault rate and mean show down for load balancing policies like CPU, MEM, CPU_MEM_HP and CPU_MEM_HT for August trace.

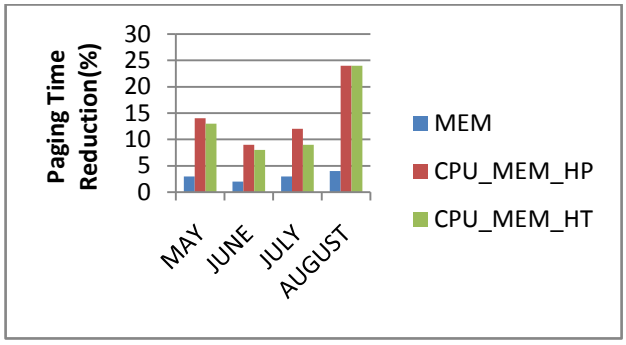

Fig. 6 – Paging time reduction

As can be seen in figure 6, it shows the paging time reduction percentage for load balancing policies like MEM, CPU_MEM_HP and CPU_MEM_HT.

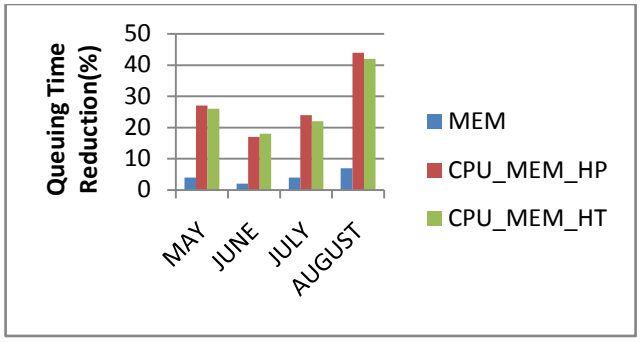

Fig. 7 – Queuing time reduction

As can be seen in figure 7, it shows the queuing time reduction percentage for load balancing policies like MEM, CPU_MEM_HP and CPU_MEM_HT.

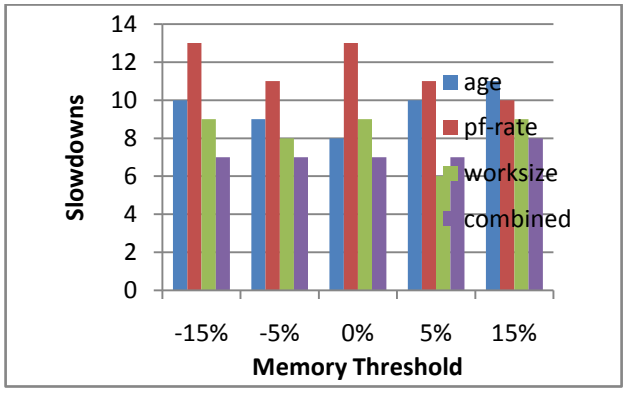

Fig. 8 – Memory threshold vs. slowdown

As can be seen in figure 8, it shows the memory threshold and slowdowns with attributes such as age, pf-rate, work size, and combined.

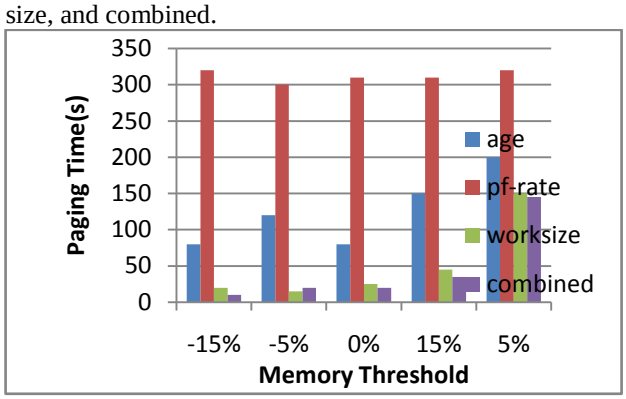

Fig. 9 – Memory threshold vs. paging time

As can be seen in figure 9, it shows the memory threshold and paging time with attributes such as age, pf-rate, work size, and combined.

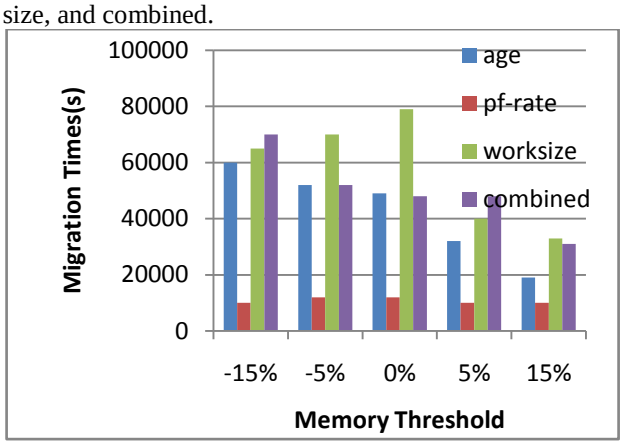

Fig. 10 – Memory threshold vs. migration time

As can be seen in figure 10, it shows the memory threshold and migration time with attributes such as age, pf-rate, work size, and combined.

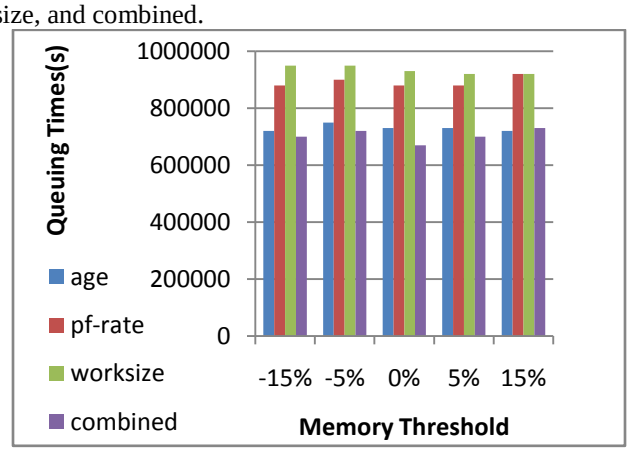

Fig. 11 – Memory threshold vs. queuing time

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

As can be seen in figure 8, it shows the memory threshold and queuing time with attributes such as age, pf-rate, work size, and combined.

V. CONCLUSIONS

In this paper we focused on the improving the performance of clusters by optimally balancing memory and CPU resources. To achieve this we implemented various load sharing policies. We assume that jobs are uncertain. It does mean there are known and unknown demands with respect to jobs execution. Unless the workload is balanced, the cluster computing cannot improve performance. Our sharing policies consider both CPU and memory resources to balance the load. As there are unknown memory demands and unpredictable memory access patterns, our policies optimize the usage of CPU and memory resources and balance load for improving jobs execution performance of clusters. We tested our sharing policies with trace-based simulations. The empirical results reveal the effectiveness of the proposed solution.

REFERENCES

[1] T.Kunz, "The influence of Different workload Descriptions on a heuristic Load balancing scheme," IEEE Trans. Software Eng., vol. 17, no. 7, pp.725-730, 1991.
[2] C.-C. Hui and S.T.Chanson, "Improved Strategies for Dynamic Load Sharing, " IEEE Concurrency., vol. 7, no. 3, pp.58-67, 1999.
[3] M.Harchol-Balter and A.B.Downey," Exploting Process Lifetime Distributions for Dynamic Load Balancing," ACM Trans. Computer Systems, vol. 15, no. 3, pp.253-285, 1997.
[4] D.L.Eager, E.D.Lazowska and J.Zahorjan, "The Limited Performance Benfits of Migrating Active Processes for Load Sharing, "Proc.ACM SIGMETRICS Conf.Measuring and Modeling of Computer Systems, pp.63-72, May 1988.
[5] X.Du and X.Zhang, "Coordinating Parallel Process on Networks of Workstations,"J.Parallel and Distributed Computing, vol. 46, no. 2, pp.125-135, 1997.
[6] F.Douglis and J.Ousterhout,"Transparent Process Migration: Design alternatives and the Sprite Implementation," Software Practice and Experience, vol. 21, no. 8, pp.757-785, 1991.
[7] M.D.Flouris and E.P.Markatos, "Network RAM," High Performance Cluster Computing, Chapter 16, R.Buyya, ed., vol.1, pp.383-508, New Jersey: Prentice Hall 1999.
[8] J.L.Hennessy and D.A. Patterson, Computer Architecture: A Quantative Approach, second ed., Morgan Kuaffman, 1996.
[9] E.P.Markatos and G.Dramitinos, " Implementation of a Reliable Remote Memory Pager," Proc.1996 Usenix Technical Conf., pp177-190, Jan.1996.
[10] M.J.Feeley et al., " Implementing Global Memory Management Systems," Proc. 15th ACM Symp. Operating System Principles, pp.201-212, Dec.1995.
[11] A.Barak and A.Braverman, " Memory Ushering in a Scalable Computing Cluster," J.Microprocessors and Microsystems, vol.22,No.3-4, pp.175-182, Aug.1998.
[12] A.Achraya and S.Setia, "Availability and Utility of Idle Memory in Workstations Cluster," Proc. ACM SIGMETRICS Conf. Measuring and Modeling of Computer Systems, pp.35-46, May 1999.
[13] G.M.Voelker, H.A.Jamrozik, M.K.Vernon, H.M.Levy and E.D.Lazowska, "Managing Server Load in Global Memory Systems ," Proc. ACM SIGMETRICS Conf. Measuring and Modeling of Computer Systems, pp.127-138, May 1997.
[[14] Li Xiao, Songqing Chen and Xiaodong Zhang, " Dynamic Cluster Resource Allocations for Jobs with Known and Unknown Memory Demands," IEEE Transactions on Parallel and Distributed Systems, Vol.13, No.3, March 2002.
[15] D. Vijaya Krishna and Dr. P. Sammulal, "Advances in Parallel computing from the past to the future", International journal of advanced research in computer science and management studies, Vol.1, Issue.4, September 2013.

BIOGRAPHIES

Dandanayakula Vijaya Krishna, He completed M.Tech (CSE) in JNTUH College of Engineering, Jagtial, INDIA. He has received B.Tech degree in Computer Science and engineering. His main research interest includes Cluster Computing, Remote Sensing.

Gajarla Santhosh, He completed M.Tech (CSE) in JNTUH College of Engineering, Jagtial, INDIA. He has received B.Tech degree in Information Technology. His main research interest includes Image Processing and Cluster Computing.

Dr. P Sammulal, He did his PhD (Computer Science and Engineering) from Osmania University, Hyderabad, India. He has received the B.E degree in 2002. He is in teaching and research since 2002. He has published 30 papers in International conferences/journals. His main research interest includes Parallel, Distributed, Cluster, Grid Computing concepts. He is working as an Assistant Professor in JNTUH College of Engineering (JNTUH University), Jagtial, INDIA.