Oriental Journal of Computer Science & Technology Vol. **4**(1), 99-105 (2011)

Framework for Threshold based Centralized Load Balancing Policy for Heterogeneous Systems

ARCHANA B. SAXENA and DEEPTI SHARMA

Department of Computer Science, Jagan Institute of Management Studies,
Affiliated to GGSIPU, Delhi (India).

(Received: January 12, 2011; Accepted: February 18, 2011)

ABSTRACT

We propose threshold based centralized load balancing policy for heterogeneous systems, where all incoming jobs are acknowledged by central server and completed with the help of workstations. All the activities (Load distribution & Load sharing) within system are regulated with the help of a central server by maintaining Capability matrix and Load matrix. We present details how threshold based Load balancing is implemented in anticipated system. A node is assigned a relative load threshold on the basis of their operational capability. Overloaded (Load > Threshold) and under loaded (Load < threshold) nodes can route load balancing through central server. We intend to do a simulation study to compare the propose scheme with conventional Load balancing scheme to show that projected scheme will increase system throughput and condense execution time. The computer simulation is based on time, sender and receiver initiative load balancing scheme, and is tested for a number of decision thresholds. We hope our result will match our expectation.

Key words: Heterogeneous Systems, Central Server, Threshold, Load Balancing, Load Distribution, Centralized Load Balancing, Sender Initiative, Receiver Initiative.

Abbreviations: S: System, CS: Central Server, LB: Load Balancing, CM: Capability Matrix, LM: Load Matrix, Ji: Job, n: Node, Ji.Mi: Memory Requirements of Job, Ji.Si: Processing Speed Requirements of Job, Ni.Mi: Memory availability of with Node, Ni.Si: Processing Speed capabilities of Node.

INTRODUCTION

Heterogeneous system network interconnect a multitude of diverse machines to perform computationally intensive applications that have diverse computational requirements. Heterogeneous system comprises of hardware and operating system software from variety of vendors. Zina Ben Miled et al, 1998 established that heterogeneous machine can have higher cost-efficiency than the optimal homogeneous machine. The performance of this kind of system is very conditioned by the strong dependence that exists between their architecture and job allocation. System performance can be improved significantly if machine heterogeneity is taken into account by load distribution policy. In heterogeneous system, where different machines with different competency are available, scrutiny of all the nodes for best capable node that can share workload with currently overloaded node would involve substantial time delay and communication cost. An effective workload distribution and Load Balanced state is required in order to reduce the total execution time and increase system throughput.

In the direction to address above discussed issues, we have devised a centralized adaptive load balancing scheme, where Load distribution is a job scheduling policy regulated by Central server, which takes a job as a whole and assigns it to single node for execution on the basis of processing requirements of job by consulting capability matrix.

Apart from being the allocator, the next important role of ***CS*** is to act as load balancer in the system. In the above discussed system, Load Balancing is a threshold based policy where load threshold (an integer value) is associated with every machine on the basis of their operational capabilities. Whenever workload of a node exceeds stipulated threshold (Load > Threshold), then it comes in overloaded status and all the nodes where workload is lesser (Load < Threshold) then comes in under load category. Load balancing request can be initiated by Overloaded (Sender Initiative) or under loaded (Receiver Initiative) and routed through Central server.

This paper is organized as follows: Related work is reviewed in section II. The architecture of our policy is presented in section III. Traffic Model or Load distribution mechanism is described in section IV. Section V explains about Load Balancing strategies. . Future work and Conclusion are listed in section VI. Section VII details references that are related to this research work.

Related work

Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma [2008] have studied various Static(Round Robin and Randomized Algorithms, Central Manager Algorithm, Threshold Algorithm*)* and Dynamic (Central Queue Algorithm, Local Queue Algorithm) Load Balancing Algorithms in distributed system. The performance of these algorithms are measured by following parameters: Overload Rejection, Fault Tolerant, Forecasting Accuracy,, Stability,, Centralized or Decentralized,, Nature of Load Balancing Algorithms,, Cooperative, Process Migration,, Resource Utilization. Their results shows that static Load balancing algorithms are more stable with such parameters

Neeraj Nehra, R.B. Patel, V.K. Bhat [2007] have proposed a DDLB(Dynamic Distributed Load Balancing) scheme for minimizing the average completion time of application running in parallel and improve the utilization of nodes. In proposed scheme instead of migrating the process for load balancing between clusters, they split the entire process into job and then balance the load. In order to achieve their target they will make use of MA (Mobile agent) to distribute load among nodes in a cluster.

Johan PARENT, Katja Verbeeck and Jan LEMEIRE [Brussels, Belgium][2002] have done experiment on heterogeneous cluster of PCs to execute parallel application with master-slave software architecture and show that using reinforcement learning it is possible to reduce the strain on the communication hardware. This can be achieved by individually adapting the amount of data (block size) requested by each slave.

Helen D. Karatza and Ralph C. Hilzer [2002] have worked together to find a load sharing policy in heterogeneous distributed environment where half of the total processor have double of the speed of others and excepts only two types of jobs: first class and Generic. They have studied only non-preemptive job scheduling policies where scheduler have exact information about queue length of all processor and queuing time of dedicated jobs in fast processor.

Kalim Qureshi* and Masahiko Hatanaka [2000] presented a short survey on HDC systems and identified load balancing problems in parallel raytracing in such systems. They have studied the performance of the RTS (Runtime Task scheduling) strategy for raytracing application on HDC systems and made suggestions to improve it.

Anna Hal aud Theodore J. Johnson [1986] has addressed a Load balancing problem in LOCUS distributed file system and proposed a CSS (centralized synchronization sites) policy for optimal process and read site placement. This algorithm considers 8 different job types (distinguish on the basis of CPU, disk and network requirements) and three classes of work load for the site.

Mohammed Javeed Zaki, Wei Li, and

Srinivasan Parthasarathy [1997] have worked on the concept of Dynamic Customized Load Balancing for heterogeneous network. Their experiment result shows that different load balancing schemes(Local, global, centralized and distributed) are best for different application under varying processor, program and system parameter and therefore application-driven customized dynamic load balancing becomes essential for good performance.

Andrew J. Page and Thomas J. Naughton have proposed a genetic algorithm (GA) to dynamically schedule heterogeneous task on heterogeneous system in a distributed environment to minimize total execution time. GA uses historical information to exploit the best solution and completes it process in three steps: Selection, Crossover and Random mutations. The GA algorithm is only performed if there are more unscheduled tasks than processors;

Kun-Ming V. Yu*, Chih-Hsun Chou* and Yao-Tien Wang have designed and implemented a Load balancing system based on fuzzy logic and proved that, this algorithm not only effectively reduces the amount of communication messages but also provides considerable improvement in overall performance such as short response times, high throughputs, and short turnaround times.

Zhiling Lan, Valerie E. Taylor and Greg Bryan have proposed a Load Balancing scheme for (SAMR) Structured Adaptive Mesh Refinement Application on distributed systems. In proposed scheme they consider the heterogeneity of processor and dynamic load on system and divide the complete Load balancing process into two phases Global Load Balancing and Local Load Balancing and.

Architectural description
System Layout

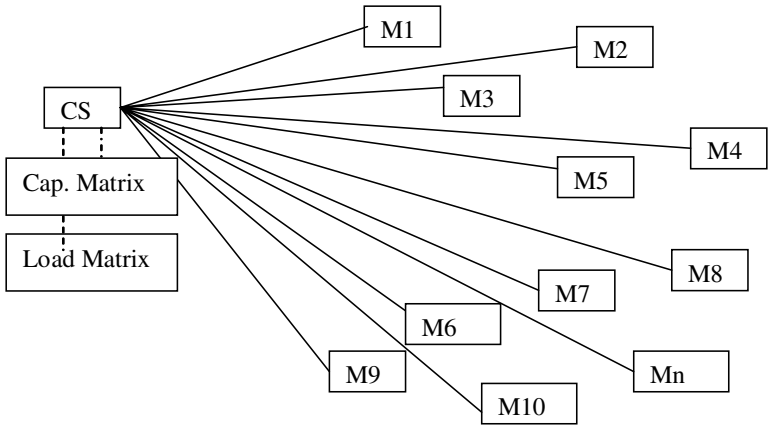

Fig. 1: Network model of system S, where heterogeneous machines are regulated by Central Server

In this study, we assume a wired LAN of heterogeneous machines similar to the one depicted in Fig I. In the above portray system consist of **S**-11 heterogeneous processors each serving its own queue (A job as a whole). A high speed network connects nodes with regulating authority CS. For a regulating authority, it is essential to know who all are part of this network and what are their processing capabilities. So, when ever any node wants to associate with the network, it has to register itself with CS. Any machine is eligible to get any work load if it is registered with the CS. At the time of registration, node sends memory and processing capacity to the CS through highest quality signal then CS saves configuration of nodes in capability matrix for future use. In this paper, this procedure is assumed to be completed with the help of database attached with central server. First two

columns of Capability matrix represents identity and IP address of the machine and Last two columns shows Memory and processing capabilities of Node.

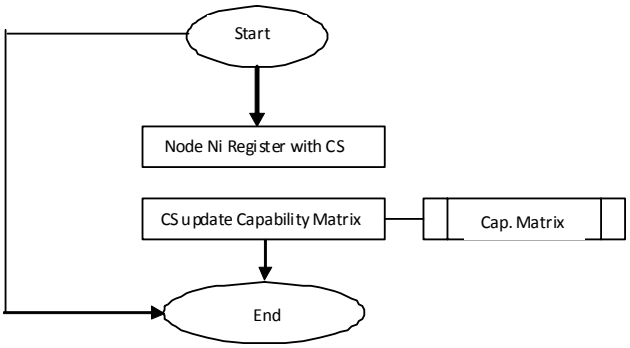

Fig. 2: Flow chart of new node registration with CS and updating of capability Matrix

Table 1: [Capability Matrix]: Machine Identity, IP address, Memory and Processing Capabilities of Machine that CS maintains for Load Distribution among Nodes

Mac Id	IP Address	RAM (N.Mi)Node.Memory	CPU (N.Si)Node.Speed
MAC-I	172.16.0.2	2048 MB	2.2GHZ
MAC-II	172.16.0.3	3072 MB	2.0GHZ
MAC-N	172.16.0.28	512 MB	2.6GHZ

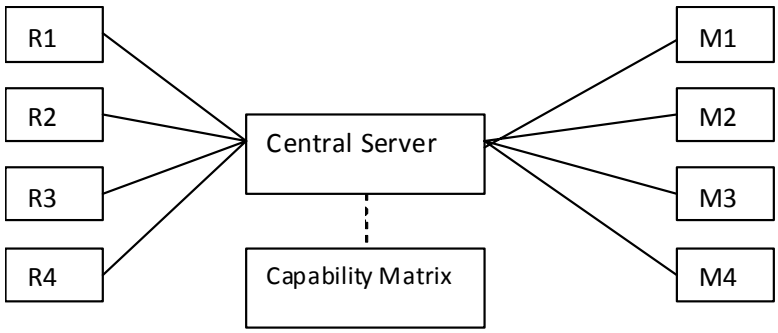

Fig 3: All incoming requests are distributed among registered node by consulting Capability matrix maintained by Central Server

Traffic model/ load distribution

Incoming traffic at the network is acknowledged by the CS (Central Server) and partitioned among all the registered nodes based on the processing capabilities of requested job. The job execution is highly independent. Each node has to process the job as a whole on its own. Each job *ji* requires memory requirements *Mi* and processing speed *Si*. Mi and Si is an integer value representing memory and processing speed that is required to process that job. A node *N* is eligible to process this job if N.Mi > *Ji.Mi* and N.Si > Ji.Si. CS selects host for new process/job by reviewing node's capabilities through CM.

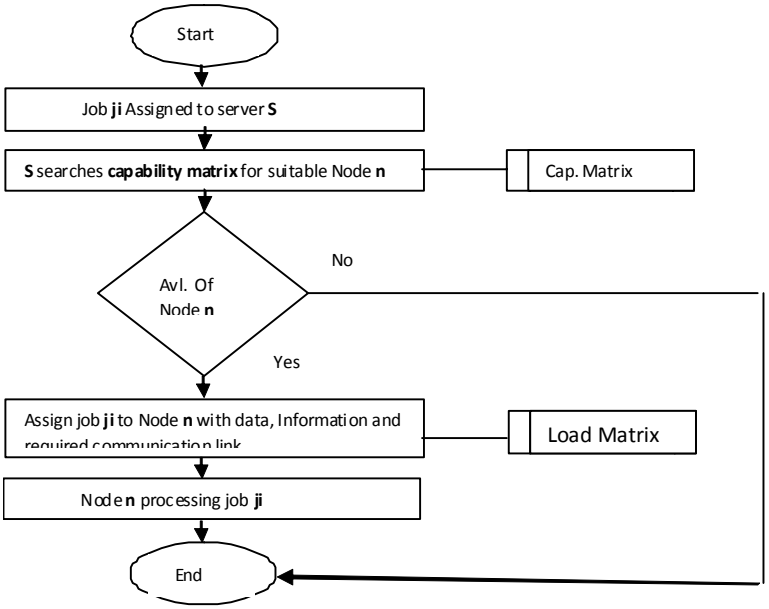

Fig. 4: Flow Chart for Job Assignment

After receiving a Job for execution, 'CS' carries out the following Load distribution procedure. It consists of four steps

A. Acknowledgement of Job receipt by CS.

B. 'CS' searches capability matrix for suitable node. For this study, we assume capability matrix consist of 'N' nodes. CS starts from first node in the capability matrix. If Ni.Mi > Ji.Mi then go to C other wisr move to step B and compare next Node in the matrix.

C. If Ni.Si > Ji.Si then go to D otherwise move to B and compare Next Node in the matrix.

D. Job is assigned to Node Ni and all the details are recorded in the Load Matrix.

Table 2: [Load Matrix]: Process Id of Job, Machine Identity, Machine capabilities and Job Status update in Load Matrix by Central server at the time of Load Distribution and Load Balancing

Process ID	Mac ID	Mac IP	RAM	CPU	Job Status
P001	Mac I	172.16.0.28	1800 MB	1GHz	Executing
Poo2	Mac-II	172.16.0.29	2200 MB	2Ghz	Executing

Load balancing strategy

In projected system, we apply threshold based dynamic load balancing scheme where CS balance load among under loaded or overloaded node on the basis of request generated and by monitoring their current position. CS keeps a private copy of node's load status. Load Matrix is one of the key elements in load balancing activity. (Load matrix maintained in database attached with CS). The Load of a node can be characterized by one of three levels:

- Load balanced: Load = Threshold
- Over Loaded : Load > Threshold
- Under Loaded : Load < Threshold

The policy in present system can be initiated by Overloaded (Sender Initiative) or Under loaded (Receiver Initiative) node. In both the cases, request is routed through CS.

Sender Initiative Load Balancing (Overloaded Node: Load > Threshold): when a Node *ni* would like to share its workload, it sends a request through communication link to *CS* for load sharing. In order, to respond the requested node *n*, *CS* proceeds with following actions:

1. To confirm job status of ni, CS checks LM and procedure stops/ request rejected if ni is not overloaded, otherwise move to step 2.
2. CS checks LM for non executing processes/ jobs pending with ni. (In present system only non-executing jobs are considered for migration). If there is no such job then procedure ends otherwise move to step 3.
3. After selecting the job for migration, CS looks for (Through CM) node that can process selected job. Move to step 4 if one of them is found to have enough processing capabilities to execute selected job.
4. The last step involved job migration from Overloaded (ni) to newly selected node and updating LM.

Receiver Initiative Load Balancing (Under loaded Node: Load < Threshold): when an under loaded Node *ni* would like to share its workload, it sends a request through communication link to *CS* for load sharing. In order to respond the requested node *n, CS* proceeds with following actions:

1. To confirm job status of ni, CS checks LM and procedure stops/ request rejected if ni is not overloaded, otherwise move to step 2.
2. CS checks the processing capabilities of required node ***ni,*** in order to decide what sort/ type of job can be assign to his node ***ni.***
3. CS checks LM for non executing processes/ jobs pending with any node where job requirements matches with processing capabilities of node ni. (In present system only non-executing jobs are considered for migration). If there is no such job then procedure ends otherwise move to step 4.
4. The last step involved job migration from Overloaded(ni) to newly selected node and updating LM.

Experimental evaluation/ performance evaluation

Traditional load balancing scheme is compared with proposed load balancing scheme. Through simulation we will try to prove that our load balancing scheme improves work through put.

REFERENCES

1 Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, (April, 2008), "Performance Analysis of Load Balancing Algorithms"

2. Branco Kalinka R. L. J. Castelo, Santana Marcos José , Santana Regina H. C. , Bruschi Sarita Mazzini, Kawabata Célia Leiko Ogawa, Ordonez Edward David Moreno, PIV and WPIV: Two New Performance Indices for Heterogeneous Systems Evaluation (2007).

3. Neeraj Nehra, R.B. Patel, and V.K. Bhat "A Framework for Distributed Dynamic Load Balancing in Heterogeneous Cluster" (2007).

4. Lan Zhiling, Taylor Valerie E., "Dynamic Load Balancing of SAMR Applications on Distributed Systems", (2007).

5. Othman Ossama, and Schmidt Douglas C., "Optimizing Distributed System Performance via Adaptive Middleware Load Balancing", (2007).

6. Godfrey Brighten, Lakshminarayanan Karthik, Surana Sonesh, Richard Karp, Stoica Ion (2004), "Load Balancing in Dynamic Structured P2P Systems."

7. Helen D. Karatza and Ralph C. Hilzer, "Load sharing in heterogeneous distributed systems" (2002).

8. Karatza Helen D. , Hilzer Ralph C., Load sharing in heterogeneous distributed systems (2002).

9. Johan PARENT, Katja Verbeeck and Jan LEMEIRE, "Adaptive Load Balancing of Parallel Applications with Reinforcement Learning on Heterogeneous Networks"

(2002).

10. Othman Ossama, O'Ryan Carlos, and Schmidt Douglas C, "An Efficient Adaptive Load Balancing Service for CORBA" (2001).
11. Shahzad Malik, "Dynamic Load Balancing in a Network of Workstations" (2000).
12. Kalim Qureshi and Masahiko Hatanaka," An introduction to load balancing for parallel raytracing on HDC systems" (2000).
13. Maheswaran, M., S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund., "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing Systems." (1999).
14. Mohammed Javeed Zaki, Wei Li, and Srinivasan Parthasarathy2, "Customized Dynamic Load Balancing for a Network of Workstations1" (1997).
15. Li Jie and Kameda Hisao "Load Balancing Problems for Multiclass Jobs in Distributed/ Parallel Computer Systems (1998).
16. Mirchandaney, R., D. Towsley, and J. Stankovic. Adaptive load sharing in heterogeneous systems (1990).
17. Shenker, S., and A. Weinrib. "The optimal control heterogeneous queuing systems: a paradigm for load sharing and routing." (1989).
18. Tanenbaum Andrew S. And Renesse Robbert Van, "Distributed Operating System", Computing Surveys, **17**(4): (1985).
19. Cow, Y.-C., and H. W. Kohler., Models for dynamic load balancing in a heterogeneous multiple processor system. *IEEE Transactions on Computers* **28**(5): 354-361 1979.
20. Kun-Ming V. Yu, and Chih-Hsun Chou "A Fuzzy-Based Dynamic Load-Balancing Algorithm"
21. Andrew J. Page and Thomas J. Naughton, "Framework for task scheduling in heterogeneous distributed computing using genetic algorithms"
22. Zhiling Lan, Valerie E. Taylor and Greg Bryan, "Dynamic Load Balancing of SAMR Applications on Distributed Systems"
23. Anna HaL aud Theodore J. Johnson, "A Study of Dynamic Load Balancing in a Distributed System"
24. Zina Ben Miled , Zina Ben , José A.B. Fortes, Rudolf Eigenmann , Valerie Taylor "On the Cost-efficiency of Hierarchical Heterogeneous Machines for Compiler- and Hand-Parallelized Applications", (1998).