Dynamic load distribution algorithm performance in heterogeneous distributed system for I/O- intensive task

Chandra, Pushpendra Kumar;
Sahoo, Bibhudatta;

TENCON 2008 - 2008, TENCON 2008.
IEEE Region 10 Conference
19-21 Nov. 2008 Page(s):1 - 5

http://dx.doi.org/10.1109/TENCON.2008.4766739

Dynamic Load Distribution Algorithm Performance in Heterogeneous Distributed System for I/O- intensive Task

Pushpendra Kumar Chandra
Department of CSE
National Institute of Technology Rourkela
pushpendrachandra@gmail.com

Bibhudatta Sahoo
Department of CSE
National Institute of Technology Rourkela
bdsahu@nitrkl.ac.in

Abstract.* The goal of load balancing is to assign to each node a number of tasks proportional to its performance. Many load balancers have been proposed that deal with applications with homogeneous tasks; but, applications with heterogeneous tasks have proven to be far more complex to handle. Load balancing techniques play a very important role in developing high-performance cluster computing platforms. Many load balancing polices achieve high system performance by increasing the utilization of CPU, memory, or a combination of CPU and memory. However, these load-balancing policies are less effective when the workload comprises of a large number of I/O-intensive tasks and I/O resources exhibit imbalanced load. The I/O intensive tasks running on a heterogeneous cluster needs effective usage of global I/O resources. We have proposed a load-balancing scheme based upon system heterogeneity and migrate I/O-intensive tasks to the fastest processor. The proposed load balancing scheme can minimizes the average slow down of all parallel jobs running on a cluster and reduces the average response time of the jobs.

***Index Terms:* Heterogeneous cluster, I/O-intensive task, Load balancing**

I. INTRODUCTION

Distributed heterogeneous computing is being widely applied to a variety of large size computational problems. These computational environments are consists of multiple heterogeneous computing modules, these modules interact with each other to solve the problem. In a Heterogeneous distributed computing system (HDCS), processing loads arrive from many users at random time instants. A proper scheduling policy attempts to assign these loads to available computing nodes so as to complete the processing of all loads in the shortest possible time.

The resource manager schedules the processes in a distributed system to make use of the system resources in such a manner that resource usage, response time, network congestion, and scheduling overhead are optimized. There are number of techniques and methodologies for scheduling processes of a distributed system. These are task assignment, load-balancing, load-sharing approaches. Due to heterogeneity of computing nodes, jobs encounter different execution times on different processors. Therefore, research should address scheduling in heterogeneous environment.

In task assignment approach, each process submitted by a user for processing is viewed as a collection of related tasks and these tasks are scheduled to suitable nodes so as to improve performance. In load sharing approach simply attempts to conserve the ability of the system to perform work by assuring that no node is idle while processes wait for being processed. In load balancing approach, processes submitted by the users are distributed among the nodes of the system so as to equalize the workload among the nodes at any point of time. Processes might have to be migrated from one machine to another even in the middle of execution to ensure equal workload. Load balancing strategies may be static or dynamic.

In static scheduling, the assignment of the tasks to the nodes is done before the execution of the program. Information regarding task execution time and processing resources is assumed to be known at compile time. A task is always executed on the node to which it is assigned. Dynamic scheduling is based on the re-distribution of processes among the processors during execution time. This redistribution is performed by transferring tasks from heavily-loaded processors to lightly-loaded processors with an aim to minimize the processing time of the application. The advantage of dynamic load balancing over static scheduling is that the system need not be aware of run-time behavior of the application before execution. The flexibility inherent in dynamic load balancing allows for adaptation to unforeseen application requirements at run-time.

In general, load-balancing algorithms can be broadly categorized as centralized or decentralized, dynamic or static, periodic or non-periodic, and those with thresholds or without thresholds. We have used a centralized load-balancing algorithm framework as it imposes fewer overheads on the system than the decentralized algorithm The load-balancing problem, aim to compute the assignment with smallest possible makespan (i.e. the completion time at the maximum loaded computing node) The load distribution problem is known to be NP-hard in most cases and therefore intractable with number of tasks and/or the computing node exceeds few units. Here, the load balancing is a job scheduling policy which takes a job as a whole and assign it to a computing node. This paper considers the problem of finding an optimal solution for load balancing in heterogeneous distributed system. The rest of the paper is organized as follows. The next section discusses Heterogeneous distributed computing system (HDCS) structure and the load-balancing problem. Section 3 describes the different dynamic load distribution algorithms. We have simulated the behavior of different load balancing algorithm with our simulator developed using Matlab, where each task t_i is with the expected execution time e_{ij} and expected completion time C_{ij}, on machine M_j. The results of the simulation with scalability of computing nodes and tasks are presented in Section 4. Finally, conclusions and directions for future research are discussed in Section 7.

II. HETEROGENEOUS DISTRIBUTED COMPUTING SYSTEM

Heterogeneous distributed computing system (HDCS) utilizes a distributed suite of different high-performance machines, interconnected with high-speed links, to perform different computationally intensive applications that have diverse computational requirements. Distributed computing provides the capability for the utilization of remote computing resources and allows for increased levels of flexibility, reliability, and modularity. In heterogeneous distributed computing system the computational power of the computing entities are possibly different for each processor. A large heterogeneous distributed computing system (HDCS) consists of potentially millions of heterogeneous computing nodes connected by the global Internet. The applicability and strength of HDCS are derived from their ability to meet computing needs to appropriate resources.

Resource management sub systems of the HDCS are designated to schedule the execution of the tasks that arrive for the service. HDCS environments are well suited to meet the computational demands of large, diverse groups of tasks. The problem of optimally mapping also defined as matching and scheduling.

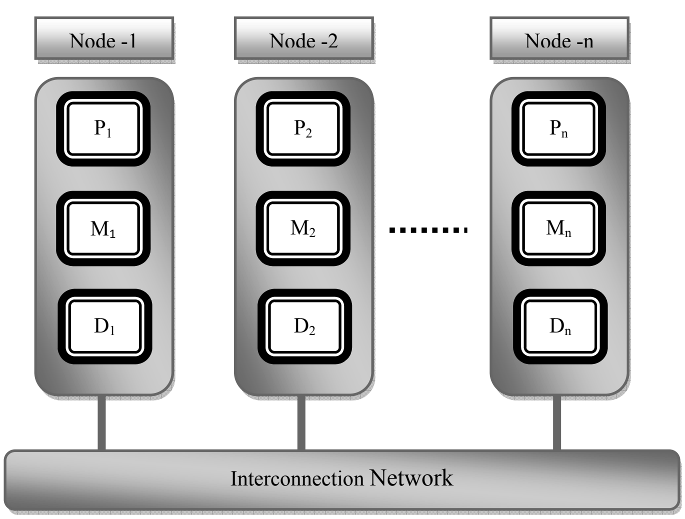

Figure: 1 Distributed Computing System

We consider a heterogeneous distributed computing system (HDCS) consists of a set of n { N1, N2, … Nn} independent heterogeneous, uniquely addressable computing entity (computing nodes). In figure 1 processor are represented as $\{P_1,P_2,P_3\ldots..P_n\}$ and memory $\{M_1,M_2,M_3\ldots..M_n\}$ and disk $\{D_1,D_2,D_3\ldots..D_n\}$ for each nodes. Each node having the different processing capability. Let there are m number of jobs with each job j has a processing time tj are to be processed in the HDCS with m nodes. Hence the generalized load-balancing problem is to assign each job to one of the node N_i so that the loads placed on all machine are as "balanced" as possible [5].

III. BACKGROUND

In the past decade, load balancing techniques in the context of CPU and memory resources has been extensively studied in recent year. There are many approaches to balancing load in disk I/O resource can be found in literature [1][2][3][4][6][10]. Xiao Qin[1] proposed a algorithm IOLB and compare this algorithm with conventional CPU- and memory-aware load balancing schemes and shows that the IOLB algorithm significantly improves the resource utilization of a cluster under I/O-intensive workload.

Mais Nijim Tao Xie, 2005 developed a performance model for self-manage computer systems under dynamic workload condition, where both CPU- and I/O-intensive applications are running in computer systems. They show that the controller is capable of achieving high performance for computer systems under workloads exhibiting high variabilities.

Xiao Qin et al.[4] proposed a feedback control mechanism to improve the performance of a cluster by adaptively manipulating the I/O buffer sizes. The primary objective of this mechanism is to minimize the number of page faults for memory-intensive jobs while improving the buffer utilization of I/O-intensive jobs. The feedback controller judiciously configures the weights to achieve an optimal performance. Meanwhile under a workload where the memory demand is high, the buffer sizes are decreased to allocate more memory for memory-intensive jobs, thereby leading to a low page-fault rate. Increasing attention has been drawn toward I/O-intensive application.

Kandaswamy et al. [10] examined optimization techniques and architecture scalability. They evaluated the effect of the techniques using five I/O-intensive applications from both small and large applications domain. Xiao Qin et al.[6] developed two effective I/O-aware load-balancing schemes, which make it possible to balance I/O load by assigning I/O-intensive sequential and parallel jobs to nodes with light I/O loads. However, the above techniques are insufficient for automatic computing platforms due to the lack of adaptability. We proposed an algorithm that take all the parallel task and it balance the I/O-intensive load with effective manner.

IV. SYSTEM MODEL AND METHODOLOGY

In our study we have considered a cluster computing platform of heterogeneous system in which set of N= {N1 ,N2 , N3 …..Nn} n nodes are connected via a high speed network. Each node in this model composed of a combination of various resources including processor, memory, disk ,network connectivity and every node is differ with their processor, memory and disk. A load manger or master node is responsible for load balancing and monitoring available resources of the node. Figure 1 shows the queuing model for load manager.

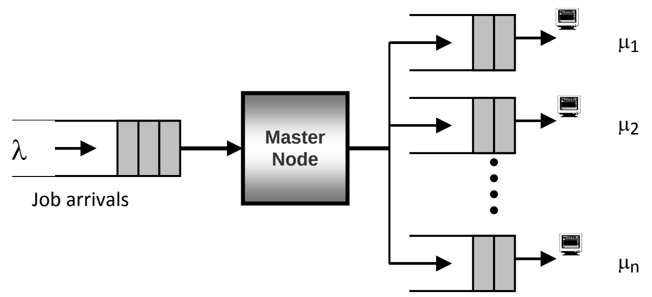

Figure 1: M/M/n heterogeneous system

Here we are considering a variant M/M/n queue where the service rates of the two processors are not identical this is the case of heterogeneous multiprocessor system. The queuing structure is shown in below figure. Assume without loss of generality that $\mu_1 > \mu_2 > \mu_3 > \mu_4 \ldots\ldots\ldots > \mu_n$.

The state of the system is defined to be the tuple $(k_1,k_2,k_3\ldots\ldots k_n)$ where n1≥0 denotes the number of jobs in the queue including any at the faster processor and n2 denotes the number of jobs at slower processor. Jobs wait in line in the order of their arrival. When processors is ideal, the faster processor is scheduled for service before the slower processor. The traffic intensity for this system is

$$\rho = \frac{\lambda}{\sum_{i=1}^{n} \mu_i} \tag{1}$$

The average number of jobs in the system may now computed by observing that the number of jobs in the system.Therefore the average number of jobs is given by:

$$E[N] = \frac{1}{A(1-\rho)^2} \tag{2}$$

Where

$$A = \frac{(1+2\rho)\prod_{i=1}^{n} \mu_i}{\lambda \sum_{i=1}^{n} (\lambda + \mu_i)} + \frac{1}{1-\rho} \tag{3}$$

The prediction scheme consists of two parts. In the first part, which is an off-line procedure, resource usage states are determined for program executions of a given UNIX system. Resource usage data is collected for all processes that ran on the system for a few days, this data is analyzed as follows: Each process is represented by a point in a three-dimensional space, where each dimension corresponds to the resources of the system, i.e., the CPU, the memory, and the file I/O. A statistical clustering algorithm is then used to identify the high density regions of this three-dimensional space (i.e., determine the number of such regions and the means of their centroids). By definition, most program executions occur in or near these regions, and therefore they are referred to as the resource usage states.

In the second part, which is an on-line procedure, actual prediction is made. The prediction scheme builds and maintains a state-transition model for each program on an on-going basis. The states of the model are the resource usage states defined above. Suppose a program has been executed several times, providing a sequence of execution instances. First, the sequence of execution instances is converted into a sequence of resource usage states by assigning the nearest resource usage state to each execution instance. The state transition probabilities are then calculated from this new sequence to build a state-transition model for the program. The prediction is a weighted mean calculation of resource requirements using the program's current state-transition model and the actual resource usage in its most recent execution. See [7] for further details. Then predicted value is fed to the selector that is used to select the best node among all nodes where the task will execute. That node is under-loaded and gives response effectively. Scheduler is responsible to dispatch the task to the node selected by the selector. Then task will send to that node and task will execute there. Load manager update the load status table.

V. DYNAMIC LOAD DISTRIBUTION ALGORITHM

We proposed a algorithm for a wide variety of workload conditions including I/O-intensive, CPU-intensive and memory-intensive load. The objective of the proposed algorithm is to balance the load of three types of resources across all nodes in a cluster. In this study analytically evaluate the performance of algorithm; we are focused on a remote execution mechanism in which task can be running on a remote node where it started execution. Thus preemptive migrations of tasks are not supported in our algorithm.

To describe this algorithm first we introduce the following three load indices with respect to I/O, CPU, memory resources. (1) CPU load of a node is characterized by the length of CPU waiting queue, denoted as LCPU(i). to identify whether node i's CPU is overloaded. (2) Memory load of a node is the sum of the memory space allocated to all the tasks running on that node. The memory load of node i is denoted as LMEM(i) (3)I/O load measures two types of I/O accesses, i.e. (a) implicit I/O request includes by page fault; (b) explicit I/O request issued from tasks. IO load index of node i is denoted as $L_{IO}(i)$. Table 1 shows the definition of notation we used in this paper.

Table 1: Definition of Notation

Notation	Definition
N	Number of node in heterogeneous system
j	Task submitted to the system
λ	Arrival rate of task
μn	Service rate of heterogeneous system
$IOREQ_j$	I/O requirement of task j
$CPUREQ_j$	CPU requirement of task j
$MEMREQ_j$	MEMORY requirement of task j
L_a^{IO}	I/O load on node(1≤a≤n)
L_a^{CPU}	CPU load on node(1≤a≤n)
L_a^{MEM}	MEMORY load on node(1≤a≤n)
L_{IO}^k	I/O load index on set of k node that satisfy all requirements
L_{CPU}^k	CPU load index on set of k node
L_{MEM}^k	MEMORY load index on set of k node
R_j^k	Response time of task on set of k nodes

Now we describe the load balancing algorithm of which the pseudo code is given above. Given a set of independent tasks submitted to the load manager. Our algorithm make an effort to balance the load of the cluster resource's by allocating each task to a node such that the expected response time is minimized. For each task j, our algorithm repeatedly performs steps 2-19 described follows:

First it will predict all three IOREQj, CPUREQj, MEMREQj requirements of task j from set of task by step 2. This three predicted value are important because according to this value task execute with best suited node. Step 3 is used to find the highest requirements of task and it is responsible for initiating the process of balancing I/O resources. Steps 4-7 are used to balance the I/O load. In step 4, if the I/O requirements of task j are high then it will find the set of nodes where I/O load is minimum and satisfies all the three requirements of the task. Step 5 calculates the response time of task with all selected nodes. In Step 6, if the response time is minimum with particular node then task will be sent to that specific node.

Algorithm: IOCM Load balancing
Input: a job with task *j* submitted to master node
1. for each task do
2. Predict the value of IO,CPU and memory requirements
3. if $IOREQ_j = \max(IOREQ_j, CPUREQ_j, MEMREQ_j)$
4. choose set of k node such that node $L_{IO}^k = \min_{a=1}^{n}(L_a^{IO})$ satisfy the all three requirements
5. calculate response time R_j^k of task j in set of k node
6. if $R_j^i = \min_{b=1}^{k}(R_j^b)$ then
7. dispatch the task to node N_i and execute there
8. else if $MEMREQ_j = \max(IOREQ_j, CPUREQ_j, MEMREQ_j)$
9. choose set of k node such that node $L_{MEM}^k = \min_{a=1}^{n}(L_a^{MEM})$ satisfy the requirements
10. calculate response time R_j^k of task j in set of k node
11. if $R_j^i = \min_{b=1}^{k}(R_j^b)$ then
12. dispatch the task to node N_i and execute there
13. else if $CPUREQ_j = \max(IOREQ_j, CPUREQ_j, MEMREQ_j)$
14. choose set of k node such that node $L_{CPU}^k = \min_{a=1}^{n}(L_a^{CPU})$ satisfy the requirements
15. calculate response time R_j^k of task j in set of k node
16. if $R_j^i = \min_{b=1}^{k}(R_j^b)$ then
17. dispatch the task to node N_i and execute there
18. update the load status;
19. end for;

Second, in step 8, if the memory requirements of task are high then it will perform steps 9-12 to balance memory load among all the nodes. Page fault behaviors occur when the memory space allocated by running tasks exceeds the amount of available memory. That's why, it is necessary to balance memory to minimize the page fault. Step 9 searches the set of nodes with minimum memory load and satisfies all the three resource requirements of the task. Step 10 calculates the response time of the task with all selected node. Step 11 finds the minimum response time of the task from selected node. Step 12 dispatches the task to selected node.

Third, step 13 is responsible if the CPU requirement of the task is high and step 14 searches the set of nodes with minimum CPU load among all the nodes that satisfy all requirements of the task. And then calculate the response time of the task in each selected node. Step16 finds node that gives minimum response time to execute the task. Step 17 dispatches the task to the selected node. Lastly, step 21 maintains updated load information that is send to the load manger.

VI. SIMULATION RESULT

The following results summarize the overall model performance. Here we are simulating the model by using the metric like throughput, number of tasks waiting in the queue with in interval. All jobs are dynamically created and allotted to the processor and processor is selected according to the conditioned as specified in our algorithm. For each processor from 2 to 50 we have taken 100 instances where job are created dynamically and allotted to the processor and then we have plotted a graph of number of processor and number of job completed. And we have seen that throughput increase if the task is I/O intensive related task.

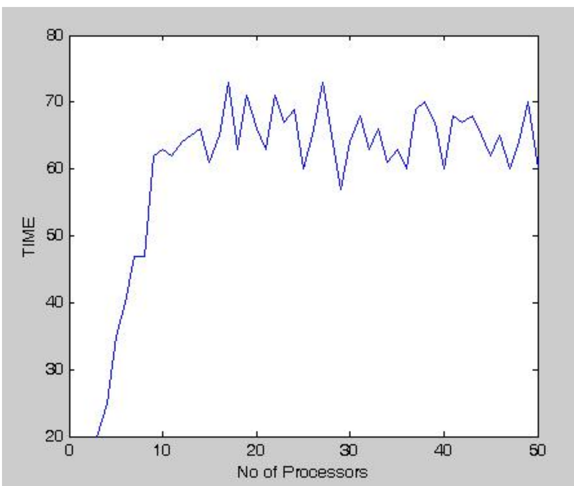

Figure 2: Throughput graph number of processor vs. time

We analyze the system performance and scalability of computing nodes with load balancing. As distributed systems continue to grow in scale, in heterogeneity, and in diverse networking technology, they are presenting challenges that need to be addressed to meet the increasing demands of better performance and services for various distributed application**.**

VII. CONCLUSION

This paper studies the performance of system under different type of load like I/O as well as CPU, MEMORY based on IOCM dynamic load balancing algorithm in heterogeneous computing system. There are number of different dynamic load balancing techniques for cluster systems; their efficiency depends on topology of the communication networks that connects nodes. This research has developed an efficient load balancing for I/O-, CPU- and MEMORY-intensive tasks. For this we developed a new way to predict and calculate the load of cluster nodes. The proposed load balancing scheme aim to achieve the effective usage of global disk resources in cluster. This can minimizes the average slow down of all parallel jobs running on a cluster and reduce the average response time of the jobs.

Future studies can be performed in following direction. First, we will evaluate the performance of scheme on a large scale of cluster. Second, we have assumed the task is independent, so we will also simulate this scheme for inter-dependent task. Third, in this study we have assumed network communication cost is negligible; therefore we will extend this to balance the load in network resource.

Acknowledgments. This research was supported by R&D project grant 2005-2008 of MHRD Government of India with the title as "Fault Tolerant Real Time Dynamic Scheduling Algorithm For Heterogeneous Distributed System" and being carried out at department of Computer Science and Engineering, NIT Rourkela.

REFERENCES

1. Xiao Qin, Performance comparisons of load balancing algorithms for IO-intensive workloads on clusters, Journal of Network and computer applications(2006), doi:10.1016/j.jnca.2006.07.001.
2. Xiao Qin ,Dynamic Load Balancing for IO-Intensive Tasks on Heterogeneous Clusters, Proceeding of the 2003 International Conference on High Performance Computing(HiPCO3).
3. Xiao Qin ,Hong Jiang ,Yifeng Zhu ,David R. Swanson ,A Dynamic Load Balancing Scheme for IO-Intensive Applications in Distributed Systems, Proceeding of 2003 international conference on Parallel processing Workshop(ICPP 2003 Workshop).
4. Xiao Qin, A feedback control mechanism for balancing I/O-intensive and memory-intensive applications on cluster, parallel and distributed computing practices journal.
5. Paul Werstein ,Hailing Situ and Zhiyi Huang , Load balancing in cluster computer, Proceeding of the seventh international conference on Parallel and Distributed Computing, Applications and Technology (PDCAT'06.
6. Xiao Qin, H.Jiang, Y.Zhu and D.swanson, Toward load balancing support for I/O intensive parallel jobs in a cluster of workstation, Poc. Of the 5th IEEE international conference cluster computing(cluster 2003) ,Hong Kong, Dec. 1-4-2003
7. Kumar K. Goswami, Murthy Devarakonda and Ravishankar K. Iyer, Prediction–baesd dynamic load-sharing heuristics, IEEE transaction on parallel and distributed systems, VOL.4, No.6, june 1993.
8. Xiao Qin, An availability-aware task scheduling strategy for heterogeneous systems, IEEE transaction on computers.
9. Mohammed Javeed Zaki, Wei Li, Srinivasan Parthasarathy, A Review of Customized Dynamic Load Balancing for a Network of Workstations.
10. M. Kandaswamy, M.Kandemir, A.Choudhary, D.Benholdt, Performance implication of architectural and software techniques on I/O intensive application, Proc International conference parallel processing 1998
11. Neeraj Nehra, R.B.Patel, V.K. Bhat ,A Framework for Distributed Dynamic Load Balancing in Heterogeneous Cluster,Journal of computer science 3(1):14-24-2007.
12. Marc H. Willebeek-LeMair , Strategies for Dynamic Load Balancing on highly parallel computer IEEE Transactions on parallel and distributed systems Vol. 4,No. 9, September 1993.
13. Bibhudatta Sahoo, S. Soma Sekhar, and Sanjay Kumar Jena, "Dynamic Load Balancing In Heterogeneous Distributed Systems Using Genetic Algorithm", Advances in Information and Communication Technology, Macmillan India Ltd., 2007, pp. 223-230.
14. K.Trivedi, Probability abd Statistics with Reliability,Queuing and Computer Science Applications.