A Novel Load Balancing Algorithm for I/O-intensive Load in Heterogeneous Clusters

Pushpendra Kumar Chandra
Department of CSE
National Institute of Technology Rourkela
Orissa -769008
+91-9861286202
pushpendrachandra@gmail.com

Bibhudatta Sahoo
Department of CSE
National Institute of Technology Rourkela
Orissa -769008
+91-6612462358
bdsahu@nitrkl.ac.in

ABSTRACT

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 from with heavily loaded nodes to under loaded nodes. 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.

Categories and Subject Descriptors

D.3.3 [**Distributed System**]

General Terms

Algorithms

Keywords

Heterogeneous cluster, I/O-intensive load, Load balancing

1. INTRODUCTION

Load balancing scheme are widely recognized as important techniques for the efficient utilization of resources in network of workstations or cluster. Clusters have evolved to support applications ranging from supercomputing and mission-critical software, through web server and e-commerce, to high-performance database applications [12]. A cluster consist of a number of node has a combination of multiple types of resources, such as CPU, memory, disk and network connectivity. In a cluster system, load balancing schemes can improve system performance by attempting to assign a work to machines with idle or under-utilized resources. Many load balancing polices are achieved high system performance by increasing the utilization of CPU, memory, or a combination of both CPU and memory resources. While these load-balancing policies are very effective in increasing the utilization of resources in heterogeneous cluster system, they have ignored one type of resource, namely disk I/O. The impact of disk I/O on overall system performance is becoming increasingly significant as more and more data intensive or I/O-intensive applications are running on heterogeneous cluster. This makes storage device a likely performance bottleneck under I/O-intensive workload. Therefore, we believe that any load balancing scheme to be effective in this new application environment; it must be made I/O –aware. Typical example of I/O-intensive application includes long running simulations of time-dependent phenomena that periodically generate snapshots of their state, archiving of raw and processed remote sensing, biological sequence, multimedia and web based applications. These applications share a common feature in that their storage and computational requirements are extremely high. Therefore, the high performance of I/O-intensive applications heavily depends on the effective usage of storage, in addition to that of CPU and memory.

In this paper we proposed a novel load balancing algorithm for all job coming to cluster. And it will balance the load in such a way that I/O, CPU and memory resource at each node can be simultaneously well utilized.

The rest of the paper is organized as follows. In the section 2 that follows, related work in the literature is briefly reviewed. In section 3, we describe the system model. In section 4 we describe the novel load balancing algorithm for I/O intensive tasks. Finally section 5 concludes the paper by summarizing the main contribution of this paper

2. RELATED WORK

Load balancing strategies try to ensure that every nodes in the cluster does almost the same amount of work at any point of time. 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 an 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 computing platform 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 variability.

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. In the above scheme increasing attention has been drawn toward I/O-intensive application. Kandaswamy et al.[10] modeled the load balancing as an optimization problem and effectiveness of load balancing scheme against architecture scalability. They evaluated their proposed 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 a algorithm that take all the parallel task and it balance the I/O-intensive load with effective manner.

3. SYSTEM MODEL

In this study we have considered a cluster computing platform of heterogeneous system in which set of $N=\{N_1, N_2, N_3 \ldots N_n\}$ 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 dynamic load distribution algorithm must be general, adaptive, stable, fault tolerant and transparent to applications. Load balancing algorithms can be classified as (i) global vs. local, (ii) centralized vs. decentralized, (iii) Non-cooperative vs. cooperative, and (iv) adaptive vs. non-adaptive[13]. In this paper we have used centralized load balancing algorithm for Heterogeneous computing clusters. Load manager in the master node in the cluster is is responsible for load balancing and monitoring available resources of the node.

Load manager process all arrival task in a FCFS manner. Master node communicates the assimilated information to all individual computing nodes, so that the nodes know the system state, when they have to migrate their process or accept new process. The computing nodes in the cluster solely depend upon the information available with master node for allocation decision.

Task are to be executed in the cluster are arrives at master node. We shall assume that all arrival streams are Poisson process. Figure 1 shows the queuing model for load manager. After being handled by load manager task are dispatched to one of the best suited node for execution. T he nodes, each of which maintains a local queue, can execute task in parallel. Load manger is composed of three modules: *(1) predictor; (2) selector;* and *(3) scheduler*;

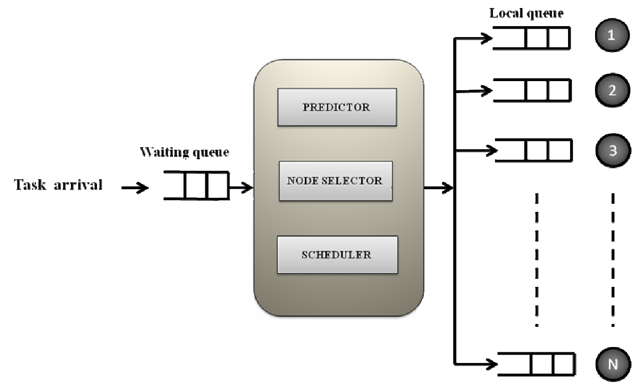

Figure 1. Queuing model for load manager

When a new task is arrives at load manger, the identification of program being executed is sent to the predictor, which predicts the resource requirements of the task. These predicted values are then fed to the selector, which selects the node with under utilized and well suited for its requirement resource.

Predictor is used to predict the file I/O, CPU and memory requirements of a task. For this we use prediction scheme described in [7], which uses a statistical pattern-recognition to predict the task requirements. The prediction is made at the beginning of a process's life, given the identity of the program being executed. 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 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 [7].

Then predicted value is fed to the selector, which responsible for selecting the best node among all nodes. *Scheduler* is responsible to dispatch the task to the node selected by the selector. On task assigned to the selected node for execution, *Load manager* update the load status table.

4. I/O-INTENSIVE LOAD BALANCING ALGORITHM

We proposed an 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.

Algorithm: IO load balancing

Input: a job with task j submitted to load manger

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 $L^k_{IO} = \min_{a=1}^{n}(L^{IO}_a)$ satisfy the all three requirements
5. calculate response time R^k_j of task j in set of k node
6. if $R^i_j = \min_{b=1}^{k}(R^b_j)$ 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 $L^k_{MEM} = \min_{a=1}^{n}(L^{MEM}_a)$ satisfy the requirements
10. calculate response time R^k_j of task j in set of k node
11. if $R^i_j = \min_{b=1}^{k}(R^b_j)$ 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 $L^k_{CPU} = \min_{a=1}^{n}(L^{CPU}_a)$ satisfy the requirements
15. calculate response time R^k_j of task j in set of k node
16. if $R^i_j = \min_{b=1}^{k}(R^b_j)$ then
17. dispatch the task to node N_i and execute there
18. update the load status;
19. end for

Figure 2. Pseudo code of the IO load balancing 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 $L_{CPU}(i)$. 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 task running on that node. The memory load of node i is denoted as $L_{MEM}(i)$ (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 $L_{IO}(i)$.

Now we describe the load balancing algorithm of which the pseudo code is shown in Figure 2. 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 t our algorithm repeatedly performs steps 2-19 described follows:

First it will predict all three $IOREQ_j$, $CPUREQ_j$, $MEMREQ_j$ 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. Step 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 node where I/O load is minimum and satisfy all three requirements. Step 5 calculates the response time of task with all selected node. Step 6 if the response time is minimum with particular node then task will send to that specific node.

Second, step 8 if the memory requirements of task are high then it will perform to step 9-12 to balance memory load among all 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 node with minimum memory load and satisfies all three resource requirement of task. Step 10 calculate the response time of task with all selected node then step 11 find the minimum response time of task from selected node then step 12 dispatch the to selected node.

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

5. CONCLUSION

Cluster computing has emerged as a result of the convergence of several trends, including the availability of inexpensive high performance microprocessors and high speed networks, the development of standard software tools for high performance distributed computing, and the increasing need of computing power for computational science and commercial applications. Even though there are number of different dynamic load balancing techniques for cluster systems, their efficiency depends topology of the communication network that connects nodes. This research has developed an efficient load-balancing algorithm for I/O intensive tasks that uses a new procedure for calculating the load at individual node. The proposed load balancing scheme aim to achieve the effective usage of global disk resources in the cluster. This can minimizes the average slow down of all parallel jobs running on a cluster and reduces the average response time of the jobs. Future studies can be performed to evaluate the effectiveness of proposed approach in following directions: (i) on scalability of cluster size, (ii) dependent tasks and periodic tasks with and without dead line, (iii) communication latency and type switching technology used in cluster.

6. 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.

7. REFERENCES

[1] Xiao Qin.2006. Performance comparisons of load balancing algorithms for IO-intensive workloads on clusters, Journal of Network and computer applications doi:10.1016/j.jnca.2006.07.001.

[2] Xiao Qin.2003, 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.2003, 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.2006 , 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.2003, 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.1993, 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.1998, 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.2007 ,A Framework for Distributed Dynamic Load Balancing in Heterogeneous Cluster,Journal of computer science 3(1):14-24-2007

[12] Marc H. Willebeek-LeMair.1993, Strategies for Dynamic Load Balancing on highly parallel computer IEEE Transactions on parallel and distributed systems Vol. 4,No. 9, September 1993.

[13] Jie Wu.1999 Distributed system design,(CRC press, 1999)

[14] Buyya, R.1999 High Performance Cluster Computing: Architectures and Systems .