Bandwidth-aware co-scheduling for clusters

A.Neela madheswari[1], M.Azath[2],Dr.R.S.D Wahida Banu[3]
[1,2]Asst.Professor,Department of CSE,Mets School Of Engineering,Thrissur.
[1]neela.madheswari@gmail.com
[2]mailmeazath@gmail.com
[3]Research Supervisor, Anna University, Coimbatore.
[3]drwahidabanu@gmail.com

Abstract : **Cluster computing are the best category of number of off-the-shelf commodity computers and resources that are integrated through hardware, networks and software to behave as a single computer simultaneously. In parallel applications, some processes are in need of executing simultaneously. We cannot be sure that all the processes are independent due to its communication behavior of some processes. Many of the processes are in need of co-scheduling each other. There are various types of co-scheduling available. This paper will focus mainly on the bandwidth and the memory concept mainly. This paper demands for the efficient resource utilization of cluster resources under the parallel execution of jobs using the newer bandwidth-aware co-scheduling concept which is put forth here.**

Keywords: Scheduling, Cluster, bandwidth, memory, co-scheduling.

I. INTRODUCTION

Parallel processing is being seen as the only cost-effective method for the fast solution of computationally large and data-intensive problems [1]. The largest shift towards parallel computing is occurring right now. A large majority of the desktop and notebook computers sold today for everyday use employs dual-core and quad-core chips. Several server, console and special purpose processors even contain between 8 and 96 cores and the trend to increase on-chip parallelism is expected to continue in the foreseeable future [2].

Clusters use intelligent mechanisms for dynamic and network-wide resource sharing, which respond to resource requirements and availability. These mechanism support scalability of cluster performance and allow a flexible use of workstations, since the cluster or network-wide available resources are expected to be larger than the available resources at any one node/workstation of cluster [7]. Many enterprises are now looking at clusters of high-performance, low cost computers to provide increased application performance, high availability and ease of scaling within the data enter. Interest in and development of computer clusters has largely been driven by the increase in the performance of off-the-shelf commodity computers, high speed, low latency network switches and the maturity of the software components [3].

Scheduling of processes onto processors of a parallel machine has always been an important and challenging area of research. Its importance stems from the impact of the scheduling discipline on the throughput and response times of a system. The research is challenging because of the numerous factors involved in the design and implementation of a scheduler [4].

Co-scheduling for clusters is a challenging problem because it must reconcile the demands of parallel and local computations, balancing parallel efficiency against local interactive response. Ideally a co-scheduling system would provide the efficiency of a batch-scheduled system for parallel jobs and a private timesharing system for interactive users. In reality, the situation is much more complex, as we expect some parallel jobs to be interactive [9].

II. RELATED WORK

In recent years researchers have developed parallel scheduling algorithms that can be loosely organized into three main classes according to the degree of coordination between processes namely: explicit scheduling, local scheduling and implicit scheduling.

Explicit co-scheduling [5] ensures that the scheduling of communication jobs is coordinated by creating a static global list of the order in which jobs should be scheduled and then requiring a simultaneous context-switch across all processors. Unfortunately this approach is neither scalable nor reliable. Further more it requires that the schedule of communicating processes be preempted, thus complicating the co-scheduling of applications and require pessimistic assumptions about which processes communicate

with one another. Explicit co-scheduling of parallel jobs also adversely affects the performance on interactive and IO based jobs.

Conversely local scheduling allows each processor to independently schedule its processes. The performance of fine-grain communicating jobs degrades significantly because scheduling is not coordinated across processors [10].

In implicit or dynamic co-scheduling, each local scheduler makes scheduling decisions that dynamically coordinate the scheduling actions of cooperating processes across processors. These actions are based on local events that occur naturally within communicating applications.

III. PROPOSED SCHEDULING STRATEGY

The scheduler must have information on the content of each machine's disk cache in addition to the availability of compute-slots on each machine to attain co-scheduling [4]. An acute complexity faced by the classes of co-scheduling is the computation of optimal co-schedules. This complexity stays unanswered. Detection of optimal co-schedules is significant for two reasons. First, the evaluation of a variety of scheduling systems has been facilitated by this. Second, a well-organized optimal co-scheduling algorithm can directly fit the necessity of practical co-scheduling. To find out their rate of communication, the communication between processes or threads has been monitored by the runtime activities. The need for co-scheduling has been typically associated with communication. Latency and bandwidth are two metrics associated with communication and memory. Neither of them is uniform, but is precise to a particular component of the memory hierarchy [8]. A new scheduling algorithm has been proposed based on the bandwidth and the memory.

IV. PROPOSED SCHEDULING ALGORITHM

The proposed scheduling algorithm aims to schedule the number of processes of a particular job in the processor. If all the processes of a job cannot be assigned in a processor without enough memory then the processes will be grouped level by level. This grouping can be done by using the Multi-Level Preliminary Grouping (MLPG) and Communication-Cost Effective Grouping (CCEG) Algorithms. The grouping of all the processes is mainly based on the communication between each of the processes of a job. This grouping can be done by calculating the communication cost of each of the processes. Then the grouped processes are scheduled to be assigned to the processor having sufficient amount of memory to accommodate all the processes in the group.

4.1 MULTI-LEVEL PRELIMINARY GROUPING (MLPG) ALGORITHM

In this algorithm, the processes of a job can be initially grouped on the basis of the communication cost between the processes. The grouping can be done level by level. The preliminary grouping is done on the basis of communication costs between the processes.

P_{CJ} → Processes of a current job

N_C → Number of communications between two processes.

B_R → Bandwidth required for communication between two processes

PG_C → Vector of process groups having communication with other processes

PG_{NC} → Vector of process groups not having communication with other processes

for each process p in P_{CJ}

if (*P* communicates with other processes)

$$PG_C << P;$$

end if

end for

$$PG_{NC} = P_{CJ} \setminus PG_C$$

The processes of the current job are divided into two sets based on their communication with other processes.

$N = \{No.\ of\ \ processes\ of\ \ a\ current\ \ job\}$

Let e be any single element of N

$$F(e,T) = \{X \cup \{e\} | X \in T\}$$

$$T = N \setminus \{e\}$$

$$P(N) = P(T) \cup F(e, P(T))$$

The number of sets with k elements in the power set of a set with n elements will be a combination of $C(n,k)$, where

$$C(n,k) = \binom{n}{k}, 1 > k < \alpha$$

Where n = number of processes of a current job. The processes of a particular job can be grouped level by level. Select $C(n,k)$ groups

from the set $P(N)$ and each group having k number of processes.

The communication cost between two processes and the total cost of each group is calculated by using the following equations.

$$Cost = (N_C * B_R)/2$$

$$Total \text{cost of each group} = \frac{1}{\text{nCr}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} (\text{cost}_{ij})$$

Where n = number of processes of each group and r = 2 i.e. communication between 2 processes. Then the total costs are sorted in descending order and stored in S_{pg}.

$$S_{pg} = sort(tot\cos t(g))_{dsc}$$

S_{pg} → Sorted group of processes

S_i → Each group of processes in S_{pg}

S_n → Selected groups for scheduling

for $i = 1$ *to size*(S_{pg})

if $(S_n \cap S_i) = \phi$ then

$S_n = S_n << S_i$

end if

end for

The sorted groups are use to schedule the groups that cover all the processes of a job but no two groups have common processes.

If any of the process have not been included in the processes group, then add those processes to the S_n.

4.2 Communication-Cost Effective Grouping (CCEG) Algorithm

The processes which have been already grouped by MLPG algorithm are again regrouped by using this CCEG algorithm. This grouping is done on the basis of communication costs between all the processes of a particular job and is found to be effective. This grouping is used to separate the processes which are not having any communication in the current group and reassign the processes to another group having maximum communication cost with any one of the processes in that particular group.

Each value in S_C represents the communication cost between the current process P_i and the other process in the process set S_n. Each value in S_C is represented by N_{ij} If there is no communication, then the value of N_{ij} is zero.

$$X = \{x : P(x)\}$$

$$P(x) = N_{ij} \neq 0$$

Where $i = 1 > 1 < n$ and j $= 1 >$ m $<$ k

$$G_{lm} = \Delta_I [X]_i^m$$

$$S_Q << P_i$$

Here $\Delta_I [X]_i^m$ represents the index of the maximum value of the communication cost between P_i and the corresponding process group and l is the corresponding qualified group namely S_Q.

4.3 Scheduling Algorithm

In this algorithm, the number of processes (n) of a particular job has been scheduled to be assigned to any of the processors. The processes have been grouped based on their communication costs. For this algorithm to take effect, we have to check the following conditions level by level.

P_m → Total memory of each group of processes

C_m → Available free memory in each processor

S_i → Each group of processes

$C_j = \{S_i : P(S_i)\}$

where $i = 1, \ldots, n$ and $j = 1, \ldots, c$

$$P(S_i) = pm_i <= Cm_j$$

If all the processes of a particular job cannot be assigned to the processors in first level due to the shortage of memory, then it would go to next level and check the above conditions. After assigning each process group to processor C_j, then the memory of the processor gets reduced.

V. Experimental Results

This section contains an extensive experimental evaluation. According to this algorithm, the processes in a certain job are initially scheduled and then assigned to the processors. The results thus obtained were

analyzed and were proved to be better in terms of communication between the processes of a particular job. The communication between the processes and the memory required for storage were the two criterions upon which the processes were grouped. In the results, we have compared the communication costs within the grouped processes to that of the individual processes. The proposed algorithm yielded better experimental results in terms of the communication costs between the processes.

The processes of a particular job have been grouped on the basis of their cost of communication with the other processes. In the tables given below, the grouped processes sets have been compared with the same processes and the other processes of a particular job with communication cost being the condition for comparison. Communication cost between the processes of the same group assigned in one processor is found to be maximum than the other processes group. This can be verified upon analysis of the tables and charts given subsequently. Considering the communication costs between the processes of a job, the proposed algorithm was found to perform better. The processes table of the job and the respective charts are given below.

Table 1: Communication costs between the processes of same group assigned on one processor of a particular job

process	p1	p2	p5	p9
P1	0	25	54	30
P2	25	0	12	0
P5	54	12	0	0
P9	30	0	0	0

Table 2: Communication costs between the group of one processes and group of other processes assigned on another processor of a particular job

process	p3	p7	p8	p4	p6
p1	0	0	5.5	0	0
p2	15	5	5	0	0
p5	0	10	0	0	0
p9	15	0	18	0	0

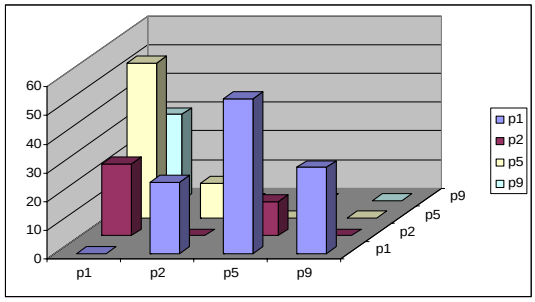

Chart 1: Communication costs between the processes of same group assigned on one processor of a particular job.

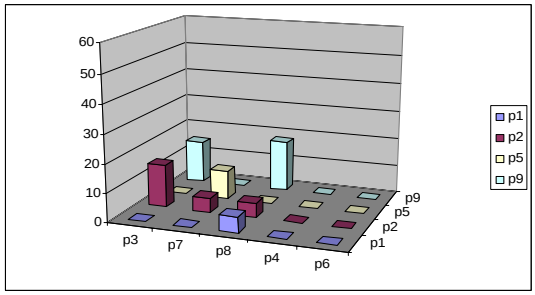

Chart 2: Communication costs between the group of one processes and group of other processes assigned on another processor of a particular job

The processes tables and charts of another one job is given as follows.

Table. 3 Communication costs between the processes of same group assigned on one processor of a particular job

process	p11	P12	p13	p15
P11	0	25	54	0
P12	25	0	0	12
P13	54	0	0	63
P15	0	12	63	0

Table. 4 Communication costs between the group of one processes and group of other processes assigned on another processor of a particular job

process	P14	p16	p17	P18	p19
P11	0	4	2.5	0	0
P12	0	4	5	0	0
P13	13	0	0	0	12.5
P15	0	1.5	0	0	0

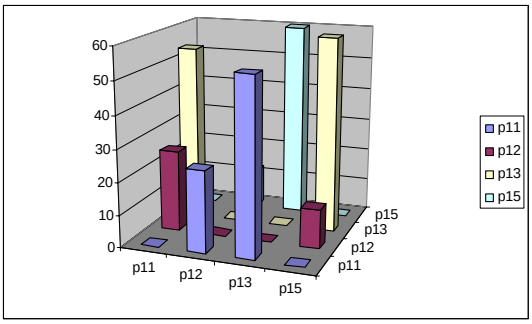

Chart 3: Communication costs between the processes of same group assigned on one processor of a particular job

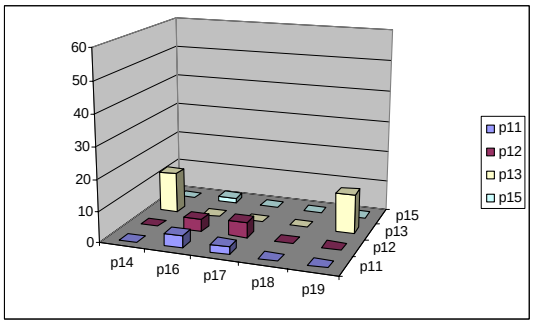

Chart 4: Communication costs between the group of one processes and group of other processes assigned on another processor of a particular job

VI. CONCLUSION

This paper investigated the problem of optimal job co-scheduling on the processors. A scheduling algorithm is proposed to improve the optimal co-scheduling of the processes of a job by utilizing the main parameters such as bandwidth and memory. Based on the usage of both the bandwidth and memory, the processes of a particular job are assigned to the processors having sufficient amount of memory. The tables and the charts of the proposed framework gives better results when the communication costs between the processes of the same group assigned in a single processor is found to be maximum than the other group of processes. All the processes in the processors are co-scheduled simultaneously while running the parallel jobs.

References

[1] "Introduction to Parallel computing", II Edition, Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar, Jan 2003.

[2] "New Challenges of parallel job scheduling", Eitan Frachtenberg, Uwe Schwiegelshohn, Workshop on Job Scheduling Strategies for Parallel Processing, 2007.

[3] "Cluster Computing" White paper from Cisco Systems, USA, 2004.

[4] "A close look at coscheduling approaches for a network of workstations", Shailabh nagar, Ajit banerjee, Anand sivasubramaniam, Chita.R.Das, Symposium on parallel algorithms and architectures, 1999.

[5] "Implications of I/O for gang scheduled workloads",L.Rudolph, W.Lee, M.Frank, K.Mackenzie, LCNS, Springer-Verlag, JSSPP 1997.

[6] "Parallel Job Scheduling and Workloads", Dror Feitolson.

[7] "Parallel programming models and Paradigms", Luis Moura E Silva, Rajkumar Buyya.

[8] "Memory Hierarchy in Cache-Based Systems", Technical report, High Performance Computing, Sun Microsystems, 2003.

[9] "Dynamic Coscheduling on Workstation Clusters", Patrick G.Sobalvaro, Scott Pakin, William E.Weihl and Andrew A.Chien, Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Vol.1459, pp. 231-256, 1998.

[10] "Buffered Coscheduling: A New Methodology for Multitasking Parallel Jobs on Distributed Systems", Fabrizio Petrini, Wu-chun Feng, 2000.

[11] "A comparative evaluation of implicit coscheduling strategies for networks of workstations", Cosimo Anglano, 2000.

[12] "Improving response time in cluster-based web servers through coscheduling, Jin-Ha-Kim, Gyu Sang Choi, Deniz Ersoz, Chita.R.Das, In the proceedings of the International Parallel and distributed processing symposium,, 2004.

[13] Platonov, A. P., Sidelnikov, D. I., Strizhov, M. V., Sukhov, A. M., "Estimation of available bandwidth and measurement infrastructure for Russian segment of Internet", arXiv:0803.1723, RIPE 56 Meeting, March 2008.

[14] P. Sammulal, A. Vinaya Babu, "Efficient and Collective Global, Local Memory Management For High Performance Cluster Computing", International Journal of Computer Science and Network Security, Vol. 8 , No. 4, pp. 81-84, 2008.