Efficient Co-Scheduling of Parallel Jobs in Cluster Computing

A.Neela madheswari
Research Scholar, Anna University
Coimbatore, Tamilnadu, India

Dr.R.S.D.Wahida Banu
Research Supervisor, Anna University, Coimbatore
Tamilnadu, India

Summary
Co-scheduling of parallel jobs in the chips is well-known to produce benefits in both system and individual job efficiency**.** The existing works have shown that job co-scheduling can effectively improve the contention, yet the question on the determination of optimal co-schedules still remains unanswered. The need for co-scheduling has been typically associated with communication bandwidth and the memory. In our work we have proposed a novel scheduling algorithm for optimal co-scheduling of parallel jobs. This algorithm facilitates the scheduling of parallel jobs using bandwidth and memory concepts. The co-scheduling of the processes in the chips using the proposed algorithm shows satisfactory improvement in performance of running the parallel jobs.

Key words:
Co-scheduling, Bandwidth, Cache Memory, Scheduling Algorithms.

1. Introduction

Since the incorporation of a number of off-the-shelf commodity computers and resources incorporated through hardware, networks and software to act as a single computer [4], [8], [15], Cluster computing is best characterized. The communication of data across the clusters mostly deals with the data latency time and the bandwidth in cluster computing environment. An interesting feature of a network cluster is that it has private communication bandwidth with respect to other network clusters despite a shared bandwidth about the processor clusters [2]. So as to support high-bandwidth and low latency inter-processor communication between the processors in the chips [14], Clusters require to incorporate fast interconnection technologies. Several processors are present in a chip and many such chips may exist. The intra-chip processor communication is of little significance whereas inter-chip processor communications share the communication bandwidth.

The choice of processors to use for the execution of each of the parallel jobs and the time of execution are some of the constituents in the scheduling of parallel jobs on a parallel supercomputer [6]. The processes to be run of different chips pose a similar scenario and hence scheduling becomes inevitable for obvious reasons. It is inefficient to perform the scheduling of numerous scientific and high-performance computing applications be composed of multiple processes running on dissimilar processors in the chips [7]. The entire parallel applications are dependent on communication practically, but the communication pattern can differ considerably between applications. Between sequential applications with similar speedup to extremely parallel and distributed applications with linear speedup, the degree of parallelism of an application too differs a lot (depending on application type). Service time and resource requirements for instance memory size and network bandwidth are the additional parameters for showing high inconsistency in parallel applications [10].

Both the existing computational resources and efficient communication bandwidth have been degraded by the load of the processors in the chip. It is required to co-schedule the jobs running on the processors in the chips consequently. To create advantages in both system and individual job efficiency [3] [11] [1], Co-scheduling of parallel jobs across the chips is eminent. The processes constituting a parallel job endure high communication latencies due to processor thrashing [3] without synchronized scheduling. In timeshared environments, Co-scheduling has been exposed to be an unfavorable factor in achieving well-organized parallel execution [15]. Owing to the fact that co-scheduling is supposed to resolve the demands of parallel and local computations apart from stabilizing parallel efficiency against local interactive response the challenges faced while applying co-scheduling for chips are huge [3]. A co-scheduling system can efficiently play the role of a batch-scheduled system for parallel jobs besides being a timesharing system for interactive users. With coordinated time-slicing between them, the entire processes in a parallel application are scheduled at the same time. Generally, this yields good parallel program performance and this is widely used to schedule parallel processes involving frequent communication [9].

There exist many researches in scheduling of parallel jobs by using many parameters like memory, latency time, bandwidth, and so on [2] , [7] , [10] and [11]. In addition, many scheduling techniques strive to co-schedule jobs that communicate frequently. However the combined usage of the memory and bandwidth has not been given considerable significance. Therefore, the performance of parallel job scheduling of the processors in the chips by

Manuscript received November 5, 2008
Manuscript revised November 20, 2008

using the major parameters like bandwidth and the cache memory has been chiefly focused in this paper. Co-scheduling comprises of a number of interacting tasks that scheduled to run at same time on different processors.

The rest of the paper is organized as follows. Section 2 presents a brief overview of the concepts used in the proposed algorithm. The proposed efficient co-scheduling algorithm is presented in Section 3. The experimental results are given in Section 4 and conclusions are summed up in Section 5.

2. Concepts Utilized in the Proposed Algorithm

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 [12]. 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 [5]. For competent scheduling of processes in the chips, a new scheduling algorithm has been proposed based on the bandwidth and the memory.

2.1 Cache Memory

To decrease the average time to access memory, a cache is utilized by the central processing unit of a computer. Numerous caches are found in modern computers. Their internal organization is classically dissimilar across the cache memory with the variation in their size and functionality. The cache is a smaller, faster memory and the copies of the data taken from the most repeatedly used main memory locations are stored in this. For lessening inter-thread latency, the cache sharing is essential. This fetches cache contention between the processes in the chips too.

2.2 Bandwidth

The bandwidth is an important metric for several applications such as grid, clusters, video and voice streaming, overlay routing, and p2p file transfers. In addition, it gives information to network applications concerning the way to control their outgoing traffic and moderately share the network bandwidth [16]. Depending on the communication between the processors [13], bandwidth necessities differ from one network to another network. It is significant for upholding a fast, functional network, the determination of the number of bits per second travel across the network and the amount of bandwidth for each application users**.**

3. Efficient Co-scheduling Algorithm

A job can be split into a limited number of processes. These processes may have the communication to achieve the faster execution of a job. As the processes of a job are assigned to more than one chip, the bandwidth and the memory usage is mainly concerned. That is, the communication between the processes is noticed to run a particular job very efficiently. So we have proposed a scheduling algorithm which is based on the bandwidth usage and memory concepts. The processes have been grouped by the communication cost of each processes and assigned to the chip which is having significant amount of memory.

The processes of a particular job can be represented by the graph structure named as Bandwidth Usage Graph. In this graph, each vertex represents the processes (P) of a particular job. The cost of each edge denotes the bandwidth usage between the two processes which has been calculated by using the number of communication and the bandwidth needed between the two processes.

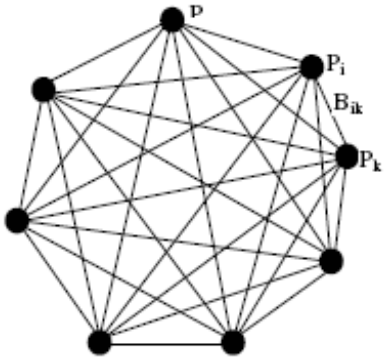

Fig. 1 Bandwidth Usage Graph

In the above graph, P represents the processes of a particular job and B_{ik} denotes the bandwidth used by the two processes (P_i, P_k) of a particular job.

The proposed scheduling algorithm aims to schedule the processes of a job in chips by grouping them as a set of processes based on their communication and memory

requirements. If all the processes of a job cannot be assigned in a chip because of less memory then the processes will be formed as a group. This grouping is done level by level. It 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 chip having sufficient amount of memory to accommodate all the processes in the group. If the available memory is not sufficient for any group, the processes of that job would be moved to the job queue.
The diagrammatic representation of the scheduling algorithm is given by

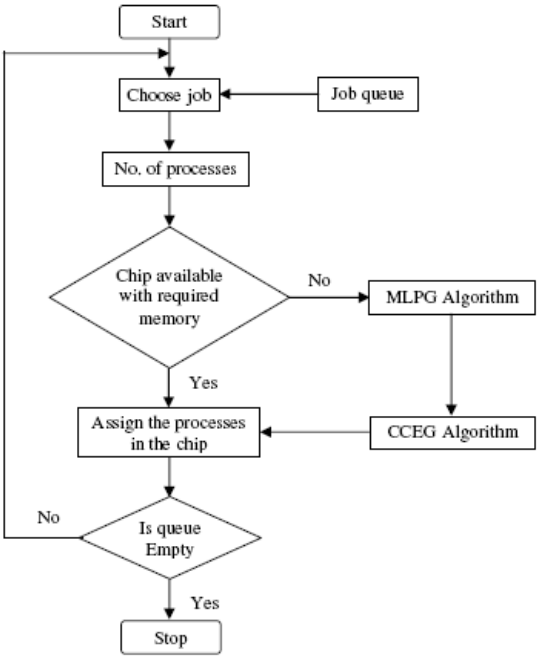

Fig. 2 Flow Diagram of Scheduling Algorithm

3.1 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. But this initial grouping is not enough for grouping of all the processes of a job. So we have to group the processes more effectively by using the Communication-Cost effective grouping (CCEG) algorithm.

Assumptions

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. The processes having communication with other processes are grouped into a set as PG_C and those not having communication into other set named as PG_{NC} .

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

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 the 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.

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

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

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}.

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

Assumptions

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 $(S_n \bigcap S_i) = \phi$ then

$S_n = S_n << S_i$

end if

end for

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

if (n mod k)!= 0 then

$S_n = S_n << PG_C \setminus S_n$

end if

If any process is left alone without being added to the process groups, then it is added to the S_n.

3.2 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 such processes are reassigned to another group having maximum communication cost with any one of the processes in that particular group.

$$S_C = \{communicat\,ion\cos ts\ of\ all\ \text{the}\ processes\}$$

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 $i = 1 > l < n$ 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. After regrouping all the processes based on the communication costs, the processes set can be represented by

$S_n = \{$

$\{P_1, P_2, P_3, ..., P_i\}$

$\{P_1, P_2, P_3, ..., P_j\}$

$\{P_1, P_2, ..., P_k\}$

$\{P_1, P_2, P_3, ..., P_1\}$

.

.

.

$\{P_1, P_2, P_3, ..., P_n\}$

$\}$

Where $k < l < i < j < n$.

3.3 Scheduling Algorithm

In this algorithm, the number of processes (n) of a particular job has been scheduled to be assigned to any of the chips. 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. For each level, the processes can be split into n / l, where l is the level for each stage. The following conditions should be checked for each level.

Assumptions

P_m → Total memory of each group of processes

C_m → Available free memory in each chip

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 job cannot be assigned to chips at 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 chip C_j, then the memory of the chip gets reduced.

i.e., $Cm_j = Cm_j - Pm_i$

4. Experimental Results

This section contains an extensive experimental evaluation. We have implemented the proposed scheduling algorithm in Java. The proposed algorithm has effectively grouped the processes of a job based on their communication and memory requirements and scheduled in right chips for better performance. 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 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 chip 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 chip 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 chip 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

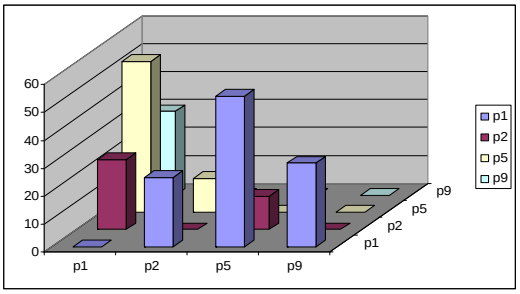

Chart 1: Communication costs between the processes of same group assigned on one chip of a particular job.

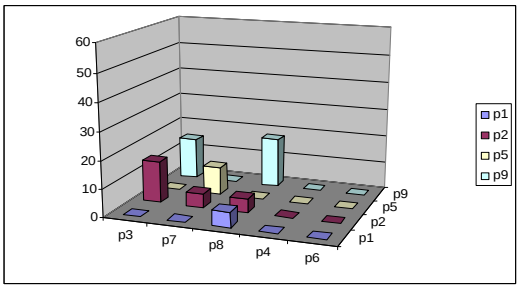

Chart 2: Communication costs between the group of one processes and group of other processes assigned on another chip of a particular job

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 chip 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 chip 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

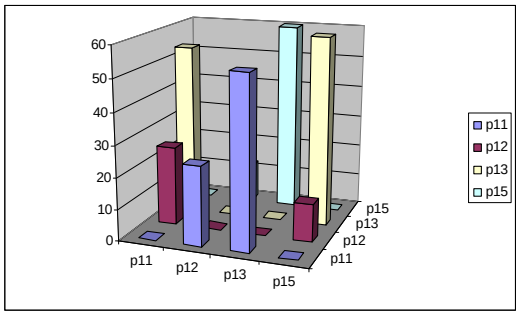

Chart 3: Communication costs between the processes of same group assigned on one chip of a particular job

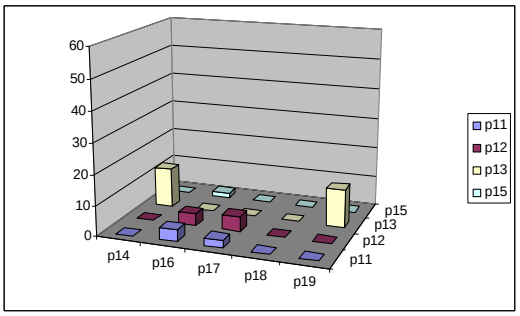

Chart 4: Communication costs between the group of one processes and group of other processes assigned on another chip of a particular job

5. Conclusion

This paper investigated the problem of optimal job co-scheduling on the chip multiprocessors. We have proposed the scheduling algorithm 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 chips having sufficient amount of memory. The tables and the charts have shown better results when the communication costs between the processes of the same group assigned in a single chip is found to be maximum than the other group of processes. All the processes in the chips have co-scheduled simultaneously while running the parallel jobs.

References

[1] Dror G. Feitelson, Larry Rudolph., "Gang Scheduling Performance Benefits for Fine-Grained Synchronization", Journal of Parallel and Distributed Computing, pp. 306-318, December 1992.

[2] Jon B. Weissman, "Scheduling Parallel Computations in a Heterogeneous Environment", University of Virginia, August 1995.

[3] Patrick G. Sobalvarro, Scott Pakin, William E. Weihl, "Dynamic coscheduling on workstation clusters", Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Vol.1459, pp. 231-256, 1998.

[4] R. Buyya, Rajkumar, "High Performance Cluster Computing : Architectures and Systems", Published by Prentice Hall, Vol. 1, 1999.

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

[6] Yair wiseman, Dror G.Feitelson, "Paired Gang scheduling", IEEE transactions on parallel and distributed systems, Vol. 14, No. 6, pp. 581-592, 2003.

[7] D. G. Feitelson, L. Rudolph, U. Schwigelshohn., "Parallel job scheduling– A status report", Proceedings of the Tenth Workshop on Job Scheduling Strategies for Parallel Processing, Vol. 3277, pp. 1–16. , 2004.

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

[9] Gyu Sang Choi, Jin-Ha Kim, Deniz Ersoz , Andy B. Yoo, Chita R. Das, "Coscheduling in Clusters: Is It a Viable Alternative?", Proceedings of the ACM/IEEE Conference on Supercomputing, page. 16, November 2004.

[10] Eitan Frachtenberg, Dror G. Feitelson, "Pitfalls in Parallel Job Scheduling Evaluation", 11th Workshop on Job Scheduling Strategies for Parallel Processing, Vol. 3834, pp. 257-282, 2005.

[11] Frachtenberg, E. Feitelson, G. Petrini, F. Fernandez, J., "Adaptive parallel job scheduling with flexible coscheduling" , IEEE Transactions on Parallel and Distributed Systems, Vol. 16, pp.1066- 1077, 2005.

[12] A. Romosan, D Rotem, A Shoshani, D Wright , "Co-Scheduling of Computation and Data on Computer Clusters", Proceedings of the 17th International Conference on Scientific and Statistical Database Management, pp. 103-112, 2005.

[13] Darius Buntinas, Guillaume Mercier, William Gropp, "Data Transfers between Processes in an SMP System: Performance Study and Application to MPI", Proceedings of the IEEE International Conference on Parallel Processing, pp. 487 - 496, 2006.

[14] Chee Shin Yeo et.al, "Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers", White Papers on University of Melbourne, pp.521-551, 2006.

[15] 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.

[16] 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.

A.Neela Madheswari received her Master of Computer Science and Engineering degree from Vinayaka Missions University, on June 2006. Currently, she is doing his research in the area of Parallel and Distributed systems under Anna University, Coimbatore. Earlier she completed her B.E, in Mahendra Engineering College from Madras University of Computer Science and Engineering, Chennai on April 2000. Later, she

joined as Lecturer at Mahendra Engineering College in CSE department from 2002 and now she serves as Lecturer at Vinayaka Missions University. Her research interest includes Distributed Computing and Web Technologies. She is a member of the Computer Society of India, Salem.

Dr.R.S.D.Wahida Banu obtained B.E. degree in 1981 and her M.E. degree in Jan '85 from GCT, Coimbatore, Madras University. She got the Ph.D. degree in 1998 from Anna University, Chennai. First lady to acquire Ph.D. in Chennai zone and second qualified Ph.D. supervisor in the area of Computer Science and Engineering related areas. As expertise is less it continues in the Directorate of Technical Education, Tamilnadu. She is the member of ISOC, IAENG, VDAT and life member of ISTE, IE, CSI and SSI. She is currently working as Professor and Head of Electronics and Communication Engineering, Government College of Engineering, Salem.