2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

A performance goal oriented processor allocation technique for centralized heterogeneous multi-cluster environments

Po-Chi Shih
Dept. of Computer Science
NTHU
Hsinchu, Taiwan
shedoh@sslab.cs.nthu.edu.tw

Kuo-Chan Huang
Dept. of Computer and Information Science
NTCU
Taichung, Taiwan
kchuang@ntcu.edu.tw

Che-Rung Lee
Dept. of Computer Science
NTHU
Hsinchu, Taiwan
cherung@cs.nthu.edu.tw

I-Hsin Chung
IBM T.J. Watson Research Center
Yorktown Heights
NY 10598
ihchung@us.ibm.com

Yeh-Ching Chung
Dept. of Computer Science
NTHU
Hsinchu, Taiwan
ychung@cs.nthu.edu.tw

***Abstract*—This paper proposes a processor allocation technique named temporal look-ahead processor allocation (TLPA) that makes allocation decision by evaluating the allocation effects on subsequent jobs in the waiting queue. TLPA has two strengths. First, it takes multiple performance factors into account when making allocation decision. Second, it can be used to optimize different performance metrics. To evaluate the performance of TLPA, we compare TLPA with best-fit and fastest-first algorithms. Simulation results show that TLPA has up to 32.75% performance improvement over conventional processor allocation algorithms in terms of average turnaround time in various system configurations.**

Keywords-multi-cluster; look-ahead; processor allocation

I. INTRODUCTION

This paper focuses on the processor allocation issues in centralized heterogeneous multi-cluster (CHMC) system. A CHMC system consists of a collection of interconnected clusters and a central job manager. Each cluster has homogeneous processors while the number and the speed of processors in different clusters may be different. The central job manager entails two tasks: job scheduling and processor allocation. Job scheduling determines the execution order of the submitted jobs, while processor allocation assigns the job to a set of available processors for execution. Job submission is in an on-line manner, which means the job manager has no information of future job submissions. Each job can be sequential (runs on single processors) or parallel (executed on multiple processors simultaneously) and there is no dependency among the jobs. Each submitted job needs to specify the number of required processors and estimated job runtime.

Processor allocation methods in CHMC can be classified into three categories, which are single site allocation [1], multi-site co-allocation [2], and adaptive allocation [3]. This paper focuses on proposing a new single site allocation algorithm in CHMC. In such environments, *spatial fragmentation* of available processors and *speed heterogeneity* among clusters are two major performance issues. Conventional Best-Fit (BF) [4] and Fastest-First (FF) [5] algorithms are designed to cope with spatial fragmentation and speed heterogeneity respectively. Their performance is unstable and largely depends on the workload and system configurations. In this paper, we propose a processor allocation technique, called *temporal look-ahead processor allocation* (TLPA), to take both spatial fragmentation and speed heterogeneity into consideration. Given a target waiting job to be allocated, the design philosophy of TLPA is to find an allocation for the target job such that this allocation will result in the best overall performance for all waiting jobs(include the target job).

In the CHMC system, processor allocation algorithms need to work together with job scheduling algorithms. With different job scheduling approaches, it requires some adaptions to utilize TLPA into processor allocation decision. To demonstrate the capability of TLPA, we propose TLPA_BJS, which is a TLPA-based processor allocation algorithm designed to work with basic job scheduling algorithms such as First-Come-First-Served (FCFS) or Shortest-Job-First (SJF). The proposed TLPA technique and TLPA_BJS processor allocation algorithm will be covered in next section.

II. TLPA TECHNIQUE AND TLPA_BJS ALGORITHM

Every algorithm that utilizes TLPA into processor allocation decision needs to specify a *scoring function*. The scoring function takes four inputs:

- ***j***: the job to be executed.
- ***c***: the cluster to be simulated for allocation.
- ***d***: the simulation depth, which is a positive integer indicating the maximum number of subsequent waiting jobs to be simulated when calculating score.
- ***p***: the performance metric to optimize.

and outputs a numerical value, called *score*. This score represents the expected performance in terms of p for those d+1 jobs (job j and d subsequent waiting jobs) if job j is allocated to cluster c. This paper focuses on the performance metric p=average turnaround time (ATT) which is defined as

$$ATT = \frac{\sum_{\forall \text{ job } i} endTime_i - submitTime_i}{\text{total number of jobs}} \quad (1)$$

The score is calculated by averaging the expected turnaround time of those d+1 jobs in the simulation procedure.

978-0-7695-4395-6/11 $26.00 © 2011 IEEE
DOI 10.1109/CCGrid.2011.81

The TLPA technique works as follows. For the job *j* to be executed, all allocable clusters need be evaluated by the scoring function, and the cluster with the best score is chosen to allocate the job for execution.

TLPA_BJS is designed to cooperate with the basic job scheduling approaches. The scoring function of TLPA_BJS is shown in Fig. 1.

Scoring Function of TLPA_BJS *(j, c, d, p)*

I. Simulate allocating job *j* to cluster *c*, estimate the runtime of *j*, and calculate the score of *j* by using *p*.

II. For i = 1 to *d* or until no jobs in the waiting queue.

(a). Pick up a job α_i from the waiting queue using the job scheduling algorithm.

(b). Find the earliest time that some cluster(s) *C'* is able to accommodate job α_i

(c). For each cluster *k* in *C'*, calculate the temporary score if allocating job α_i to cluster *k* using *p*.

(d). Simulate allocating job α_i to the cluster with the best temporary score, and set the score of job α_i to the best

Figure 1. Scoring function of TLPA_BJS

III. Experiments and Discussions

To show the effectiveness of TLPA, we compare TLPA_BJS with BF and FF under cooperation with FCFS and SJF. A series of simulations has been conducted using publicly downloadable workload trace named SDSC SP2 log [6]. Two variables, *system loading* (SL) and *system heterogeneity* (SH), were added to simulation parameters to increase the dimensions of comparison basis. SL changes the heaviness of the input workload while SH controls the variance of the computing speed among the clusters. All parameter settings used in the simulations are summarized in Table I.

Table II shows the average performance improvement of TLPA_BJS with respect to BF and FF respectively. Each result is the average of all the combinations of three SL and three SH settings. The simulation depth with the best performance in each set of experiments is shown in red color and boldface. There are two observations. First, the results show that TLPA_BJS outperforms BF and FF for all simulation combinations in terms of ATT. Second, the results reveal a clear correlation between simulation depth and performance improvement, that is, the deeper simulate depth, the better performance improvement.

IV. Conclusions

This paper investigates the issues of processor allocation in CHMC and proposes the TLPA technique to improve system performance. Experimental results show that system performance can be improved up to 32.75% by using TLPA into processor allocation decision.

TLPA provides a brand-new viewpoint to processor allocation. First, the allocation decision can be based on a performance metric other than simple policies. Second, the allocation decision is made based on simulation, not just some static rules. We anticipate further improvement can be made by utilizing those concepts in the design of new processor allocation algorithms.

TABLE I. All parameter settings used in the simulations

Number of clusters in CHMC	5
Processors in each cluster	8, 128, 128, 128, 50
Job scheduling algorithm	FCFS, SJF
Workload source	SDSC's SP2 log
System loading (SL)	Low, Medium, High
Speed heterogeneity (SH)	0, 0.1, 0.2
Simulation depth *d*	2, 4, 8, 16, 32, 64

TABLE II. Average performance improvement of TLPA_BJS with respect to BF and FF respectively

Workload source	Job scheduling algorithm	Compared processor allocation algorithm	Simulation depth	Performance improvement of TLPA_BJS
SDSC's SP2 log	FCFS	BF	2	1.84%
			4	3.59%
			8	11.79%
			16	20.41%
			32	25.21%
			64	**30.97%**
		FF	2	10.28%
			4	12.73%
			8	18.20%
			16	24.20%
			32	27.89%
			64	**32.75%**
	SJF	BF	2	10.32%
			4	10.31%
			8	10.88%
			16	10.77%
			32	10.21%
			64	**11.06%**
		FF	2	2.97%
			4	2.98%
			8	3.57%
			16	3.46%
			32	2.96%
			64	**3.79%**

References

[1] D. England and J. B. Weissman, "Costs and Benefits of Load Sharing in the Computational Grid," *Job Scheduling Strategies for Parallel Processing*, ed, 2005, pp. 160-175.

[2] O. Sonmez, H. Mohamed, and D. Epema, "On the Benefit of Processor Coallocation in Multicluster Grid Systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. PP, pp. 1-1, 2010.

[3] K.-C. Huang, P.-C. Shih, and Y.-C. Chung, "Using Moldability to Improve Scheduling Performance of Parallel Jobs on Computational Grid," *Advances in Grid and Pervasive Computing*, ed, 2008, pp. 116-127.

[4] K.-C. Huang and H.-Y. Chang, "An Integrated Processor Allocation and Job Scheduling Approach to Workload Management on Computing Grid," *Proceedings of the 2006 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'06)*, Las Vegas, USA, 2006, pp. 703-709.

[5] K.-C. Huang, P.-C. Shih, and Y.-C. Chung, "Towards Feasible and Effective Load Sharing in a Heterogeneous Computational Grid," *Advances in Grid and Pervasive Computing*, ed, 2007, pp. 229-240.

[6] *Parallel Workloads Archive,* http://www.cs.huji.ac.il/labs/parallel/workload/.