A Dynamic Partitioning Self-scheduling Scheme for Parallel Loops on Heterogeneous Clusters

Chao-Tung Yang[1,*], Wen-Chung Shih[2], and Shian-Shyong Tseng[2,3]

[1] High-Performance Computing Laboratory
Department of Computer Science and Information Engineering
Tunghai University Taichung 40704, Taiwan, R.O.C.
ctyang@thu.edu.tw
[2] Department of Computer and Information Science
National Chiao Tung University
Hsinchu 30010, Taiwan, R.O.C.
{gis90805, sstseng}@cis.nctu.edu.tw
[3] Department of Information Science and Applications
Asia University Taichung 41354, Taiwan, R.O.C.
sstseng@asia.edu.tw

Abstract. Loop partitioning on parallel and distributed systems has been an important problem. Furthermore, it becomes more difficult to deal with on the emerging heterogeneous PC cluster environments. In this paper, we propose a performance-based scheme, which dynamically partitions loop iterations according to the performance ratio of cluster nodes. To verify our approach, a heterogeneous cluster is built, and two kinds of application programs are implemented to be executed in this testbed. Experimental results show that our approach performs better than traditional schemes.

Keywords: Parallel loops, Self-scheduling, Cluster computing, MPI, Heterogeneous, PC cluster.

1 Introduction

As more and more inexpensive personal computers (PC) are available, clusters of PCs have become alternatives of supercomputers which many research projects cannot afford. However, it is difficult to deal with the heterogeneity in a cluster [2, 4], especially for the parallel loop scheduling problem [5].

Previous researchers [3, 6, 7] propose a two-phased self-scheduling approach, which is applicable to PC-based cluster environments. These two-phased schemes collect system configuration information, and then distribute some portion of the workload among slave nodes according to their CPU clock speed [6] or HINT measurements [7]. After that, the remaining work load is scheduled by some well-known self-scheduling scheme. Nevertheless, the performance of this approach depends on the appropriate choice of scheduling parameters. Besides, it estimates node performance only by CPU speed or HINT benchmark, which is one of the factors affecting node performance.

* Corresponding author.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part I, LNCS 3991, pp. 810 – 813, 2006.

In [3], an enhanced scheme, which dynamically adjusts scheduling parameters according to system heterogeneity, is proposed.

Previous work in [3, 7] and this paper are all inspired by [6], the α self-scheduling scheme. However, this work has different viewpoints and unique contribution. First, while [3, 6] partition α % of workload according to performance weighted by CPU clock speed in phase one, our scheme conducts the partition according to a general performance function (PF). The PF obtained by the HPL benchmark [8] can estimate performance of cluster nodes rather accurately.

Second, the scheme in [6] utilizes a fixed α value, and [3, 7] adaptively adjust the α value according to the heterogeneity of the cluster. In a word, both schemes depend on a properly chosen α value to get good performance. Nevertheless, our scheme focuses on accurate estimation of node performance, so the choice of α value is not very critical.

2 Our Approach

We propose to partition α% of workload according to the performance ratio of all nodes, and the remaining workload is dispatched by some well-known self-scheduling scheme, for example, GSS [5]. Using this approach, we do not need to know the real computer performance. However, a good performance ratio is desired to estimate performance of nodes accurately.

2.1 Performance Function and Performance Ratio

We first define the Performance Function (PF) to represent the performance index of each node. In this paper, our PF for node j is defined as

$$PF_j = \frac{B_j}{\sum_{\forall node_i \in S} B_i} \tag{1}$$

where

S is the set of all cluster nodes.

B_i is the performance value of node i measured by the HPL benchmark [8] before each workload partitioning.

The performance ratio (PR) is defined to be the ratio of all performance functions. For instance, assume the PF of three nodes are 1/2, 1/3 and 1/4. Then, the PR is 1/2 : 1/3 : 1/4; i.e., the PR of the three nodes is 6 : 4 : 3. In other words, if there are 13 loop iterations, 6 iterations will be assigned to the first node, 4 iterations will be assigned to the second node, and 3 iterations will be assigned to the last one.

2.2 Our Algorithm

The algorithm for the master node of our approach is described as follows.

Algorithm MASTER:

```
1. Evaluate Performance Ratio by the HPL benchmark.
2. Dispatch α% of loop iterations according to the
performance ratio of nodes.
3. Master does its own computation work
4. Dispatch (100-α)% of loop iterations into the task queue
using GSS.
END MASTER
```

3 Experimental Results

We have built a heterogeneous cluster which consists of 8 PCs. The configuration of this cluster testbed is shown in Table 1.

Table 1. Hardware configuration

Host Name	CPU Type	CPU Speed	Number of CPU	RAM
hpc	Intel Xeon™	2.4GHz	2	1GB
amd1	AMD Athlon™ MP	1.8GHz	2	2GB
amd1-dual1	AMD Athlon™ MP	2.2Ghz	2	512MB
amd1-dual01	AMD Athlon™ MP	2.0Ghz	2	2G
dna2	AMD Athlon™ MP	2.0Ghz	2	2G
piii-dual1	Intel Pentium III	866MHz	2	1GB
xeon2	Intel Xeon™	3.0GHz	2	512MB
hpc2	Intel Xeon™	3.0GHz	2	1GB

We have implemented the Mandelbrot set application programs in C language, with message passing interface (MPI) directives for parallelizing code segments to be processed by multiple CPUs. In this experiment, the scheduling parameter α is set to be 50 for all two-phased schemes, except for the schemes by [7], of which α is dynamically adjustable according to cluster heterogeneity.

The Mandelbrot set is a problem involving the same computation on different data points which have different convergence rates [1]. In this experiment, execution time on the heterogeneous cluster is investigated. Figure 1(a) illustrates execution time of static scheduling, dynamic scheduling (GSS [5]) and our scheme, with input image size 64×64, 128×128 and 192×192 respectively. Experimental results show that our scheduling scheme got better performance than static and dynamic ones. In this case, our scheme for input size 192×192 got 95% and 86% performance improvement over the static one and the dynamic one respectively.

Figure 1(b) illustrates execution time of previous two-phased schemes ([6] and [7]) and our scheme, with input image size 64×64, 128×128 and 192×192 respectively. Experimental results show that our hybrid scheduling scheme got better performance than [6] and [7]. In this case, our scheme for input size 192×192 got 83% and 69% performance improvement over the static one and the dynamic one respectively.

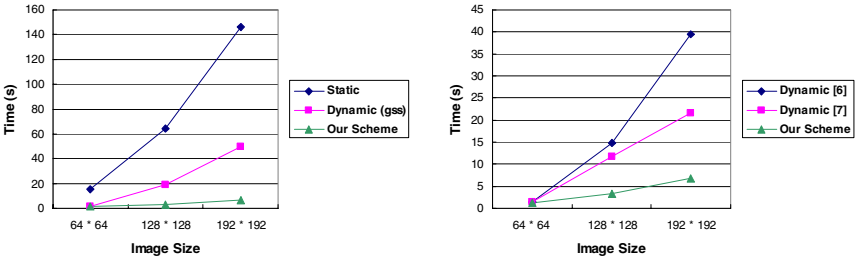

Fig. 1. Mandelbrot execution time on the heterogeneous cluster (a) Static, dynamic and our scheme; (b) Dynamic schemes [6], [7] and our scheme

4 Conclusions and Future Work

In this paper, we propose a dynamic loop partitioning scheme, and compare it with previous algorithms by experiments on the Mandelbrot application programs in our heterogeneous cluster environment. In each case, our approach can obtain performance improvement on previous schemes. In our future work, we will implement more types of application programs to verify our approach. Furthermore, we hope to find better ways of modeling the performance function, incorporating network information.

References

1. Introduction To The Mandelbrot Set, http://www.ddewey.net/mandelbrot/
2. M. Baker and R. Buyya, "Cluster Computing: The Commodity Supercomputer," International Journal of Software Practice and Experience, Vol. 29, No. 6, 2002, pp.551-575, 1999.
3. Kuan-Wei Cheng, Chao-Tung Yang, Chuan-Lin Lai, and Shun-Chyi Chang, "A Parallel Loop Self-Scheduling on Grid Computing Environments," Proceedings of the 2004 IEEE International Symposium on Parallel Architectures, Algorithms and Networks, pp. 409-414, KH, China, May 2004.
4. A. T. Chronopoulos, R. Andonie, M. Benche and D.Grosu, "A Class of Loop Self-Scheduling for Heterogeneous Clusters," Proceedings of the 2001 IEEE International Conference on Cluster Computing, pp. 282-291, 2001.
5. C. D. Polychronopoulos and D. Kuck, "Guided Self-Scheduling: a Practical Scheduling Scheme for Parallel Supercomputers," IEEE Trans. on Computers, vol. 36, no. 12, pp 1425-1439, 1987.
6. Chao-Tung Yang and Shun-Chyi Chang, "A Parallel Loop Self-Scheduling on Extremely Heterogeneous PC Clusters," Journal of Information Science and Engineering, vol. 20, no. 2, pp. 263-273, March 2004.
7. Chao-Tung Yang, Kuan-Wei Cheng, and Kuan-Ching Li, "An Efficient Parallel Loop Self-Scheduling on Grid Environments," NPC'2004 IFIP International Conference on Network and Parallel Computing, Lecture Notes in Computer Science, Springer-Verlag Heidelberg, Hai Jin, Guangrong Gao, Zhiwei Xu (Eds.), Oct. 2004.
8. http://www.netlib.org/benchmark/hpl/