Scheduling Efficiently for Irregular Load Distributions in a Large-scale Cluster*

Bao-Yin Zhang\textsuperscript{1}, Ze-Yao Mo\textsuperscript{1}, Guang-Wen Yang\textsuperscript{2}, and Wei-Min Zheng\textsuperscript{2}

\textsuperscript{1} Institute of Applied Physics and Computational Mathematics, Beijing, 100088, P.R. China
\textsuperscript{2} Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, P.R. China
zby@tsinghua.edu.cn

Abstract. Random stealing is a well-known dynamic scheduling algorithm. However, in a large-scale cluster, an idle node must randomly steal many times to obtain a task from another node, especially, this problem severely affects performance in systems where only a few nodes generate most of the system workload. In this paper, we present an efficient dynamic scheduling algorithm, Transitive Random Stealing (TRS) based on random stealing, which makes any idle node rapidly obtain a task from another node for irregular load distributions in a large-scale cluster. Then by the random baseline technique, we experimentally compare TRS with Shis, one of load balance policies in the EARTH system, and random stealing for different load distributions in the Tsinghua EastSun cluster and show that TRS is a highly efficient scheduling algorithm for irregular load distributions in a large-scale cluster. Finally, TRS is implemented in the Jcluster environment, a high performance Java parallel environment, and an experiment result is given in the HKU Gideon 300 cluster.

Keywords: Scheduling, irregular load distribution, large-scale cluster, transitive random stealing.

1 Introduction

The availability of high speed networks and increasingly powerful commodity microprocessors is making the usage of clusters of computers an appealing vehicle for cost-effective parallel computing. The scale of the clusters is becoming more and more large, which is up to hundreds of and thousands of nodes. In order to achieve scalable performance, it is important to evenly schedule the workload among the processing nodes. Two basic approaches \cite{4} to dynamically scheduling task loads can be found in current literature - random stealing and work sharing.

Random stealing attempts to steal a task from a randomly selected node when a node finds its own task queue empty, repeating steal attempts until it succeeds. Random stealing is provably efficient in terms of time, space, and communication

\* This work is supported by Chinese NSF for DYS granted by No. 60425205 and National Postdoctor Science Foundation of China.
for the class of fully strict computations [4][13]; and the natural work stealing algorithm is stable [2]. Communication is only initiated when nodes are idle. When the system load is high, no communication is needed, causing the system to behave well under high loads. Some systems that implement random stealing include Cilk [3], Jaws [8], and Satin [9]. Cilk [3] provides an efficient C-based runtime system for multithreaded parallel programming with a random stealing scheduler. JAWS [8] efficiently schedule load over a dynamically varying computing infrastructure with random stealing algorithm, Satin [9] presents a system for running divide-and-conquer programs on distributed memory systems with random stealing. The EARTH runtime system [7] supported several dynamic load balancer policies, which goal is to design simple balancers that deliver good load distribution with minimum overheads. But a virtual ring network topology is adopted in all the balancers with nodes numbered clock-wise. The authors of the paper [5] evaluate these load-balancing schedulers for a fine-grain multithreading environment.

In this paper, we study the dynamic scheduling algorithms for a large-scale cluster. For random stealing in a large-scale cluster, an idle node must randomly steal many times to obtain a task from another node. Especially, this problem severely affects performance in systems where only a few nodes generate most of the system workload [12]. For overcoming this problem, Shis, one of load balance policies in the EARTH system [5], which slightly modifies random stealing was to remember the originating node (history information) from which a task was last received, and to send requests directly to that node. The authors of the paper [11], present two relatively complicated adaptive location policies which record more history information for global scheduling algorithms.

Here we present a scheduling algorithm, Transitive Random Stealing (TRS), which further improves Shis not only remember the originating node from which a task is stolen but also forward the information of the node to other remote nodes which want to steal a task from it. With the transitive policy, TRS can make any node obtain a task faster with less times to steal in a large-scale cluster, reduce the idle time for all nodes and improve the overall performance of the system. Then by the random baseline technique, we experimentally compare the performance of TRS with Shis and random stealing for different load distributions in the Tsinghua EastSun cluster, and show that TRS outperforms Shis and random stealing in all test cases. Finally, TRS is implemented in the Jcluster environment [1], a high performance Java parallel environment, and an experiment result is given on HKU Gideon 300 cluster.

In the rest of this paper, we first give the transitive random stealing algorithm in next section. Section 3 evaluates the performance of TRS, Shis and random stealing by the random baseline technique. We show an experiment result on HKU Gideon 300 cluster in Jcluster environment in Section 4. Finally, Section 5 concludes our works.

2 Design the Transitive Random Stealing Algorithm

Our design philosophy for scheduling algorithms is to reduce the idle time for all nodes, rather than balancing work loads equally on all nodes. A node is said to