Chapter 5
Communication Analysis and Performance Prediction of Parallel Applications on Large-Scale Machines

Yan Li
Intel Labs China, China

Jidong Zhai
Tsinghua University, China

Keqin Li
State University of New York, USA

ABSTRACT

With the development of high performance computers, communication performance is a key factor affecting the performance of HPC applications. Communication patterns can be obtained by analyzing communication traces. However, existing approaches to generating communication traces need to execute the entire parallel applications on full-scale systems that are time-consuming and expensive. Furthermore, for designers of large-scale parallel computers, it is greatly desired that performance of a parallel application can be predicted at the design phase. Despite previous efforts, it remains an open problem to estimate sequential computation time in each process accurately and efficiently for large-scale parallel applications on non-existing target machines. In this chapter, we will introduce a novel technique for performing fast communication trace collection for large-scale parallel applications and an automatic performance prediction framework with a trace-driven network simulator.

INTRODUCTION

Different applications in the high performance computing (HPC) field exhibit different communication patterns, which can be characterized by three key attributes: volume, spatial and temporal (Chodnekar, et al., 1997; Kim & Lilja, 1998). Proper understanding of communication patterns of parallel applica-

DOI: 10.4018/978-1-5225-0287-6.ch005

tions is important to optimize the communication performance of these applications (Chen et al., 2006; Preissl, et al., 2008a). For example, with the knowledge of spatial and volume communication attributes, MPIPP (Chen, et al., 2006) optimizes the performance of Message Passing Interface (MPI) programs on non-uniform communication platforms by tuning the scheme of process placement. Besides, such knowledge can also help design better communication subsystems. For instance, for circuit-switched networks used in parallel computing, communication patterns are used to pre-establish connections and eliminate the runtime overhead of path establishment. Furthermore, a recent work shows spatial and volume communication attributes can be employed by replay-based MPI debuggers to reduce replay overhead significantly (Xue, et al., 2009).

Previous work on communication patterns of parallel applications mainly relies on traditional trace collection methods (Kim & Lilja, 1998; Preissl et al., 2008b; Vetter & Mueller, 2002). A series of trace collection and analysis tools have been developed, such as ITC/ITA (Intel, 2008; Kerbyson et al., 2001), KOJAK (Mohr & Wolf, 2003), TAU (Shende & Malony,2006), DiP (Labarta et al., 1996) and VAMPIR (Nagel et al., 1996). These tools need to instrument original programs at the invocation points of communication routines. The instrumented programs are executed on full-scale parallel systems and communication traces are collected during the execution. The collected communication trace files record type, size, source and destination etc. for each message. The communication patterns of parallel applications can be easily generated from the communication traces. However, traditional communication trace collection methods have two main limitations: huge resource requirement and long trace collection time. For example, ASCI SAGE routinely runs on 2000-4000 processors (Kerbyson, et al., 2001) and FT program in the NPB consumes more than 600 GB memory for Class E input (Bailey, et al., 1995). Therefore, it is impossible to use traditional trace collection methods to collect communication patterns of large-scale parallel applications without full-scale systems. Moreover, it takes several months to complete even on a system with thousands of CPUs. It is prohibitive long for trace collection and prevents many interesting explorations of using communication traces, such as input sensitivity analysis of communication patterns. Additionally, MPIP (Vetter, et al., 2001) is a lightweight profiling library for MPI applications and only collects statistical information of MPI functions. However, all these traditional trace collection methods require the execution of the entire instrumented programs, which restricts their wide usage for analyzing large-scale applications. Our method adopts the similar technique to capture the communication patterns at runtime as the traditional trace collection methods.

We have two observations on existing communication trace collection and analysis approaches:

1. Many important applications of communication pattern analysis, such as the process placement optimization (Chen, et al., 2006) and subgroup replay (Zhang, et al., 2009), do not require temporal attributes.
2. Most computation and message contents in message-passing parallel applications are not relevant to their spatial and volume communication attributes.

Motivated by the above observations, we describe a novel technique in this chapter, called FACT, which can perform fast communication trace collection for large-scale parallel applications on small-scale systems. Our idea is to reduce the original program to obtain a program slice through static analy-