Parallelization of Sparse Cholesky Factorization on an SMP Cluster

Shigehisa Satoh, Kazuhiro Kusano, Yoshio Tanaka, Motohiko Matsuda, and Mitsuhisa Sato

Tsukuba Research Center
Real World Computing Partnership
1-6-1 Takezono, Tsukuba, Ibaraki 305-0032, JAPAN

Abstract. In this paper, we present parallel implementations of the sparse Cholesky factorization kernel in the SPLASH-2 programs to evaluate performance of a Pentium Pro based SMP cluster. Solaris threads and remote memory operations are utilized for intranode parallelism and internode communications, respectively. Sparse Cholesky factorization is a typical irregular application with a high communication to computation ratio and no global synchronization between steps. We efficiently parallelized using asynchronous message handling instead of lock-based mutual exclusion between nodes, because synchronization between nodes reduces the performance significantly. We also found that the mapping of processes to processors on an SMP cluster affects the performance especially when the communication latency can not be hidden.

1 Introduction

Recent progress in microprocessors and interconnection networks motivated a trend towards high performance computing using clusters made out of commodity hardware. Symmetric multiprocessors (SMPs) are also becoming widely available. As a result, clusters of SMPs are expected to be one of the most cost-effective parallel computing platforms. In this context, we have built a PC-based SMP cluster, named COMPaS, which consists of eight quad-processor Pentium Pro SMPs connected by a Myrinet high speed network.[1][2].

COMPaS (Cluster Of Multi-Processor Systems) consists of eight quad-processor Pentium Pro PC servers connected to a Myrinet switch. Each node is a Toshiba GS700 with four 200MHz Pentium Pro CPUs and has 512KB L2 caches and 512MB of main memory. The operating system on each node is Solaris 2.5.1 which supports both the Solaris thread library and the POSIX thread library. Our user-level communication layer for Myrinet, called NICAM, provides low-overhead and high-bandwidth communication. NICAM supports remote memory data transfer primitives and global synchronization primitives. Parallel programs for COMPaS can be made using Solaris threads for intranode parallelism, and remote memory operations for communications between nodes.

Our previous work based on some regular applications produced the following guidelines for effective use of COMPaS: (1) Cache-locality is crucial to attain
high performance in an SMP node, because the shared-bus bandwidth is not sufficient for memory intensive programs, (2) Internode communication latency can be hidden by overlapping communication and computation. Though it is clear that synchronization between threads in the same node are faster than ones between processes running on different nodes, the performance bottlenecks for regular applications we have parallelized so far have to do with shared-bus bandwidth rather than synchronization overheads. The bus traffic can be mitigated by effective use of cache memories.

The synchronization performance is more important for irregular applications than regular applications. Sparse Cholesky factorization is one such irregular application. Irregular structure of matrices introduces irregular patterns of memory accesses and synchronization. Fortunately, a good deal of the factorization process can be performed by solving regular subproblems, i.e. computations for small dense matrices[3][4]. Blocked fan-out algorithm[3] is such an algorithm and is the basis of the Cholesky kernel in the SPLASH-2 programs[5].

The SPLASH-2 programs are a collection of parallel programs for shared address space multiprocessors. Although there are several studies which use the SPLASH-2 programs for performance evaluation, a few works are done for Cholesky on clusters. Some studies on clusters of uniprocessors use Cholesky for evaluation of software DSM systems[6][7][8]. These works show that the efficient parallel execution of Cholesky is difficult due to high synchronization overheads.

In this paper, we present parallel implementations and performance evaluation of Cholesky on an SMP cluster COMPaS. The original program is already parallelized using PARMACS macros for shared address space multiprocessors. We modify this parallel program for SMP clusters using Solaris threads and remote memory operations. The problems for such parallelization are as follow: (1) How to share data between nodes. (2) How to implement synchronization primitives. (3) How to assign processes to processors. The data sharing mechanism and the process to processor assignment policy are important for Cholesky.

There are two kinds of shared data in Cholesky: nonzero elements of the factor matrix and task queues for each of the processes. We use remote memory operations to keep these data coherent. The sharing patterns of nonzero elements and task queues are one-producer with multiple-consumers and migratory, respectively. Migratory sharing between nodes is quite expensive, because synchronization overhead between nodes is larger than one within the same node.

We have implemented two versions to share task queues, which have quite different synchronization characteristics. One is using messages which can be sent and received asynchronously with sender and receiver processes, respectively. The other is using global locks for mutual exclusion, which may induce the lock-serialization.

The mapping of processes to processors on each SMP node is also investigated, because the intranode communication is significantly faster than the internode communication.

This paper is organized as follows: Section 2 briefly describes characteristics of Cholesky. Parallel implementations of Cholesky for COMPaS is described in