PMT: A Generic Object-Oriented Parallel MultiThread Programming Interface

Xiaoqi Yang  
National High Performance Computing National High Performance Computing Center at Hefei  
Department of Computer Science and Technology  
University of Science and Technology of China  
xqyang11@mail.ustc.edu.cn

Qilong Zheng  
National High Performance Computing National High Performance Computing Center at Hefei  
Department of Computer Science and Technology  
University of Science and Technology of China  
qlzheng@ustc.edu.cn

Guoliang Chen  
National High Performance Computing National High Performance Computing Center at Hefei  
Department of Computer Science and Technology  
University of Science and Technology of China  
glchen@ustc.edu.cn

ABSTRACT
With the increasing popularity of shared-memory programming model, especially at the advent of Multi-Core processors, more and more programmers hope to write parallel multithread programs conveniently and effectively with the help of parallel multithread programming interface. Unfortunately, these interfaces are not well accepted by sequential programmers, because of incomplete elimination of lower-level details, inflexibility selection of library functions, depending on specific compilers. This paper presents a unique generic object-oriented parallel multithread programming interface, which is designed and implemented to solve the drawbacks of current parallel multithread programming interfaces. Evaluation results show that programmers can benefit a lot from the interface.

Categories and Subject Descriptors  
D.1.3 [Programming Techniques]: Concurrent Programming; D.2.13 [Software Engineering]: Generic Object-Oriented Libraries

General Terms  
Algorithms, Performance, Design

Keywords  
Multi-Core, Parallel Multithread Programming

1. INTRODUCTION
Compared with sequence programming, parallel programming models are more complex and diverse. Parallel programming models act as an abstraction above hardware and memory architectures, which mainly includes the message-passing model and the shared-memory model [2]. The message-passing model is intuitive and easy to implement on parallel computing environments based on distributed memory, such as MPI and PVM. However, shared-variable model is concise and intuitive in programming on parallel computers based on shared memory.

As Intel puts two cores on a single piece of silicon and as multi-socket systems continue to grow, shared memory systems are going to become the norm. To get the most out of shared memory processing power, programmers must embrace parallel programming. But parallel programming comes with its share of challenges. Data structures are tedious to write, even for experienced developers. Therefore, a good parallel multithread interface is inevitable to generate significant interest among developers, and makes them program conveniently and effectively.

2. RELATED WORK
UNIX systems, a standardized C language threads programming interface has been specified by the IEEE POSIX 1003.1c standard. Implementations that adhere to this standard are referred to as POSIX threads, or Pthread. Explicit threading libraries, however, are very complicated to use. QThreads, ZThreads and Boost Threads are advanced object-oriented, cross-platform C++ threading and synchronization libraries. These are system level interfaces without providing support for parallel programming design. OpenMP [3] is a collection of directives and supporting runtime library routines to support multi-threaded application programming. However, it lets the programmer adopt a programming discipline in which a program smoothly evolves from a working sequential program into a working parallel program. In addition, it depends on the specific compiler. Intel Thread Building Block is a C++ runtime-library that supports scalable data parallel programming., while it is still under construction.

3. The Design and Implementation of PMT
PMT is the library level solution, and it abstracts and encapsulates the basic functions of Pthread with the help of generic technology [1]. Without depending on any specific compilers, PMT can realize the functions similar to those of OpenMP. In addition, PMT provide more comprehensive parallel computing support, such as support for shared variable, point and thread safe access to complex data structure. Furthermore, PMT realizes the support of abstract data structure due to the drawbacks of OpenMP. For example, the reduction variable can be arbitrary user-defined type, and reduction operators can also extend to any user-defined operators. Based on PMT, the basic structure of parallel
multithread algorithms can be generated. The parallel multithread
realizes the function of one program section, and there are some
mechanics to support the synchronization of these threads.
Therefore, the correct program result is promised and the program
efficiency is improved.

The ultimate goal of PMT is to provide programmers with ability
of generating parallel generic algorithm.

4. EVALUATION
Parallel FFT iteration algorithm [2] is realized by PMT under the
evaluation environment (SMP: Four-processor SMP; Processor:
Pentium III; Processor Frequency: 866MHz; Memory: 512 MB).
The parallel algorithms realized by PMT flexibility. As can be
seen from the figure 2, the program has a good speed ratio. The
speed ratio is improved with the number of processors applied.

5. Conclusions and Future Work
PMT provides the support of parallel task group and common
basic communication atomic applied in the parallel algorithm
design. It defines rich methods to generate parallel region, and the
dynamic and static schedule for parallelizing loop. Furthermore,
PMT provides high-level abstraction of shared data mechanism,
such as reduction variable, shared variable, shared pointer, variety
kinds of shared data containers and so on. Compared with
OpenMP, PMT does not depend on any special language
extension and compilers. Therefore, it can be applied more
widely, and deployed on the shared-memory system more easily,
flexibility and expansibility. Currently, we focus our interest on
Speculative Parallel Threading [4] technology, which is a
promising parallel programming technique for the future multi-
core architectures and will be incorporated into our PMT system.

6. ACKNOWLEDGMENTS
This work was supported by the National Natural Science Fund
Key Project (60533020), Intel Higher Research Fund
(4507146713), and Anhui Province Natural Science Fund
(050420205). Many thanks for Dr. Zhen Yao’s previous work
related to the project.

7. REFERENCES
Programming and Design Patterns Applied. AW C++ in
Simulation of Speculative Parallel Threading using
Transactional Execution, MINI - MICRO SYSTEMS,2007