DEVELOPMENT OF TOOLS FOR TRACING OF PARALLEL PROGRAMS

Svanadze Giorgi, Tsereteli Paata

St.Andrewthe First-Call Georgian University of the Patriarchate of Georgia, Tbilisi, Georgia

Abstract

Often the performance of programs on parallel computing systems is poor than programmer expects and performance analysis and debugging is extremely complicated process. We are developing software tools that will help programmer determine bottlenecks and improve performance. Our approach is discussed in this paper

Анотация

Часто производительность программы на параллельных вычислительных системах получается меньше ожидаемой, а анализ производительности является очень сложным.. Мы разрабатываем прграммные средства, которые помогут прграммисту найти узкие места в своей программе и повысить производительность. В данной статье рассмтривается функциональность нашего продукта

Introduction

Unfortunately, parallel programs are more difficult to write, optimize and understand than serial programs because of concurrency of processes or/and threads they used. Programmer must coordinate communication between processes, synchronize threads and processes, and control access to shared object by threads. Sometimes programmer may carry these actions not as effectively as it is possible, that causes poor performance of program. To determine the bottlenecks of his software product the programmer needs tools for monitoring of application. The objective of such tools is to help programmers to get highest possible performance from their programs on their target architecture. Performance debugging consists of several phases including the phase when performance data is collected and the phase of presentation of collected data to programmer.

In order to support performance debugging a large number of performance indices must be delivered by measurement tools so that programmer can detect and reduce the overheads of their programs. Among these indices completion time and processor utilization rate are very important. Completion time is time spent by execution of various parts of program and it must be reduced as much as possible. Processor utilization rate indicates what percentage of processor time is spent executing “useful” work, programmer need to know what time is spent executing synchronization, task creation and scheduling, communications between processes, etc. Moreover, performance debugging tool must be simple in use.[1,2]

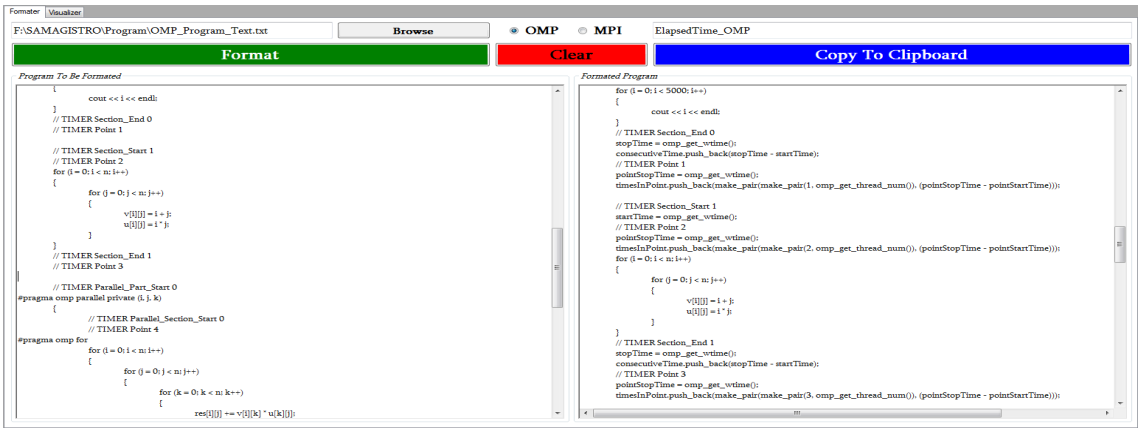
One of the ways to collect such information is tracing. It is most general event driven technique. It is very well suited to measure times spent by program into various parts of program, for example, to measure communication times, or times needed for execution of subprograms, loops, etc. It can be also used for observation of behavior of parallel threads/processes. In our approach user can set control points in the program in any place. There are two types of control points – the first is case when user wants to record only time of passing of the control points and the second is case when user wants to get execution time of program sections. Control points have form of C/C++ comments and are considered as usual comment by C/C++ compiler

The offered tool consists of two parts. The first part we call preprocessor and it processes control points and program sections set by user. In result we receive C/C++ program that includes all necessary statements for collecting times and recording them into the file. After the

program is compiled and run on target computer user obtain result file with time information. The second part of tool is visualizer. It intended for visualization of collected times.

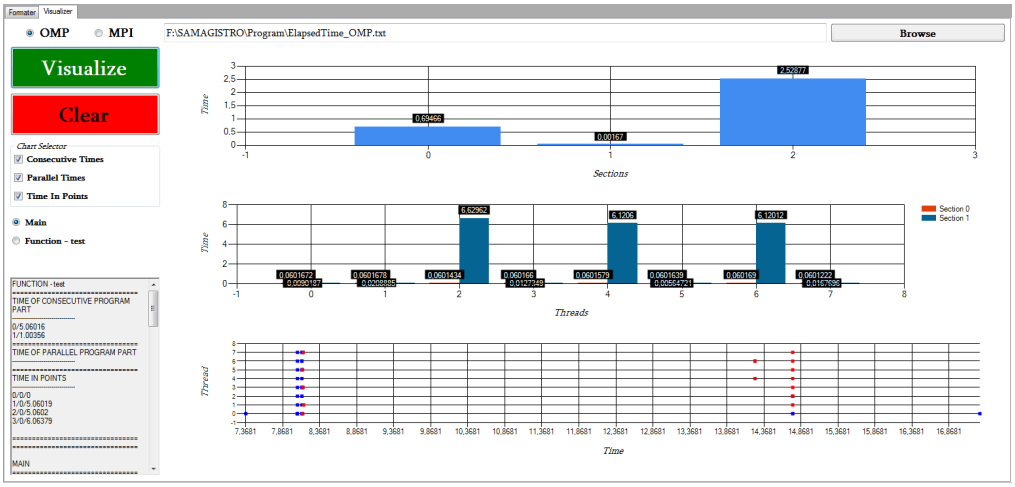
Preprocessing

Before the formatting, the user selects the type of program to be formatted, or which of OpenMP and MPI technology has been used in the program. Preprocessor does not allow file selection, until the user selects the relevant technology. After marking one of the technologies, the user uploads the text file, where the programming code is recorded with the special comment. Upon the start of the formatting the program will begin to search special comments in the code and to record the time counting and file recording commands in their place. After formatting we will get the same program, in which in the place of special comments there will be recorded time counting, file recording and related commands. The result will be recorded in the second non-editable text area, where the text may be copied by pressing a button. In addition, the user will be able to select the time recording file name in the special field.



Visualization

Before the visualization the user selects the technology used in investigated program. As in case of the preprocessing, the program does not allow to upload the file until user selects one of the two technologies. After selecting the technology, the program visual aspect changes depending on the technology, OpenMP or MPI, selected by the user. The file content upon the uploading is recorded in the non-editable text area. In case of OpenMP technology the availability of function times in the file is examined. In case of availability their number is counted and the radio buttons of this number are reflected in the program. Before the visualization the program reads the times recorded in the predefined format from the time files and other related information. Thereafter three charts in case of OpenMP technology or two charts in case of MPI technology appear. The charts may be selected through the check boxes available in the program.



Error handling

After formatting and visualization the program runs several checks. In case of failure to pass the checks the program stops to work and deliver the relevant message to the user. Also the program stops to work, if the special comments or directives required for time counting are wrongly written or are not written at all by the user.

In order to catch the errors in addition to the general errors and relevant texts the program code provides the special error texts. This makes it easier for the user to work with the program and in case of uploading of a wrong program or time files allows to correct the error.

Conclusion

In this work we presented the use of developed tool for improving of performance. It may be used by any programmer who would like to watch program execution process. The investigated program is not necessary to be parallel (written using MPI or Open MP technologies). In case of serial program the user will be able to determine execution time of various sections of program. In case of parallel program programmer can collect trace information and visualize it.

Future work includes transferring tool on web-platform, development of tools for understanding program structure, supporting on-line visualization.

References:

1.J.C. de Kergomeaux, E,Maillet and J.-M.Vincent Monitoring Parallel Programs for Performance Tuning in Cluster Environments. In “;Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments”, Nova Science Publishers, New York, 2001.

2.J.M. Wozniak, A.Chan, T.G. Armstrong, M.Wilde, E.Lusk, I.T.Foster A model for tracing and debugging large-scale task-parallel programs with MPE // *Proc. Workshop on Leveraging Abstractions and Semantics in High-performance Computing at PPoPP*, 2013.