A Productivity Centered Tools Framework for Application Performance Tuning

H. Wen  S. Sbaraglia  S. Seelam  I. Chung  G. Cong  D. Klepacki
IBM T.J. Watson Research Center, Yorktown Heights, NY
{hfwen, ssbarag, sseelam, ihchung, gcong, klepacki}@us.ibm.com

Abstract

Our productivity centered performance tuning framework for HPC applications comprises of three main components: (1) a versatile source code, performance metrics, and performance data visualization and analysis graphical user interface, (2) a unique source code and binary instrumentation engine, and (3) an array of data collection facilities to gather performance data across various dimensions including CPU, message passing, threads, memory and I/O. We believe that the ability to decipher performance impacts at the source level and the ability to probe the application with different tools at the same time at varying granularities, while hiding the complications of binary instrumentation, leads to higher productivity of scientists in understanding and tuning the performance of associated computing systems and applications.

1 Introduction

Performance tuning of High Performance Computing (HPC) applications is an iterative process during which the user goes through the following sequence of steps: (1) selection of performance data to be collected. (2) selection of an appropriate data collection tool. (3) instrumentation of the application. (4) execution of the application. (5) visualization and analysis of the collected data. After the analysis, different performance data may be needed therefore the instrumentation is then fine-tuned and all the steps are repeated. Each of these repetitions may potentially require a different tool. This process is continued until a performance bottleneck is successfully isolated, at which point the necessary modifications to the application’s source code, runtime environment, or system configuration may be made. In the above process, we believe that the end users productivity can be increased dramatically with mechanisms that decipher performance impacts at the source level and probe the application with different tools simultaneously.

In response to the productivity challenge of the U.S. DARPA HPCS [1] initiative, we have developed a methodology that addresses the various challenges in the iterative tuning process and goes beyond them to improve user productivity. The main characteristics of our framework are that: (1) It is an integrated environment for simultaneous investigation of a number of aspects of performance (e.g. CPU, memory, threads, message passing). (2) It requires no source code modifications. The user interacts with our framework by using symbolic names in the source domain, but all the modifications to the application are transparently performed on the binary. As a result, no recompilation of the application is needed. (3) It collects data for CPU, message passing, thread activity and memory, simultaneously in a single run, thereby it significantly reduces the time needed to profile and analyze the application. (4) It presents performance data information in a user-friendly manner that highlights the relation between the performance metrics and the source code statements. (5) It supports iterative tuning by allowing the analyst to refine the data collection rules and rerun the experiment results. The results of the new execution can be displayed along with the previous data for direct comparison. (6) It permits data collection at different granularities and is designed for extensibility.

2 Tools Framework

The overall structure of our framework is depicted in Figure 1. It consists of three core components: (1) a versatile source code, performance metrics, and performance data visualization and analysis Graphic User Interface (GUI), (2) a unique source code and binary instrumentation system, and (3) an array of data collection facilities to gather performance data across five dimensions: CPU, message passing, threads, memory and I/O. From previous studies with a variety of HPC applications, we have found that these five dimensions provide an excellent starting point for a programmer to understand the performance behavior of their applications. We have five different standalone tools for these five dimensions, they have been published previously (e.g., [2, 3]), hence we will not describe them here.
A Versatile Graphical User Interface (GUI): peekperf

GUI shown in Figure 2 is the control center of our framework. The entire performance tuning process, from instrumentation to execution and analysis of data can be conducted from here. This interface provides three different types of windows. The source window shows the source code files of the application, tool window controls the data collection and instrumentation, and the data window displays the performance data. The tool window displays the program structure that is specifically designed for different aspects of performance data (HPM: hardware counter data, MPI: message passing, OPENMP: threads, SIGMA: memory). By selecting the appropriate tab, the GUI displays the program structure for the selected performance dimension as a hierarchical tree and also displays the predefined instrumentation points. Users may operate on the source code or on the hierarchical tree to specify an instrumentation region, which will be highlighted in the tree structure and in the source window.

The result of executing the instrumented application is stored in a set of XML and plain text files, one file per process per dimension of performance data. These XML files follow a standard specification so that the GUI can render the performance data for different dimensions and can correlate performance metrics with the application source code. Figure 2 shows an example of performance data visualization. When a performance metric is selected, the corresponding statements are highlighted in the source code window so that users can immediately relate the collected performance information (e.g., timing, number of cache misses) to the corresponding source code sections (e.g., loops) of the application. The GUI also allows customized performance data displayed in the metric browser window.

**Binary Instrumentation:** Binary instrumentation framework, shown in Figure 1, consists of two modules, a Binary Analysis System (BAS) and a Binary Instrumentation System (BIS). BAS provides the program structure of the executable to the visualization system. It provides information about statements, variables, instructions and other information in the program. With the knowledge of the program structure, our framework is able to provide an instructive environment for the instrumentation, control of data collection, and for mapping performance data to the corresponding source code of the application. BIS provides a low-level binary modification facility, whose task is to modify the execution of an application in order to execute certain actions specified by the user. These two modules together allows the GUI to modify the application binary as required by the user.

### 3 Concluding Remarks

In this paper, we described a new tool framework for performance tuning that emphasizes productivity. Our GUI is an integrated interface for instrumentation, data visualization, and analysis, and is the means by which the programmer selectively chooses the degree of probing. The GUI is assisted by a sophisticated facility that takes specifications at the source-level and implements the binary instrumentation. It also correlates the performance data to the source lines. By abstracting the details of binary instrumentation away from the analyst and providing an automated and quickly repeatable “probe-result” cycle at the source level, our framework contributes to increase the productivity of the end user.

### References

[1] Darpa high productivity computing systems (hpcs).