MPI Profiling with the Sun™ Studio Performance Tools

Marty Itzkowitz, Yukon Maruyama, Vladimir Mezentsev

Sun Microsystems, 16 Network Circle, Menlo Park, CA 94025, USA

1 Introduction

This paper describes the various techniques implemented in the Sun Studio Performance Tools to profile MPI applications. We describe the characteristics of the MPI programming model, and review the specific performance issues with this model, and show how the tools can help.

1.1 The Sun Studio Performance Tools

The Sun Studio Performance Tools are designed to collect performance data on fully optimized and parallelized applications written in C, C++, Fortran, or Java, and any combination of these languages. Data is presented in the context of the user's programming model.

The tools support code compiled with the Sun Studio or GNU compilers. They also work on code generated by other compilers, as long as those compilers produce compatible standard ELF and DWARF symbolic information.

The tools run on the Solaris™ or Linux operating systems, on either SPARC® or x86/x64 processors. The current version, Sun Studio 12 update 1, is available for free download [1].

1.1.1 The Sun Studio Performance Tools Usage Model

The usage model for the performance tools consists of three steps. First, the user compiles the target code. No special compilation is needed, and full optimization and parallelization can be used. It is recommended that the **`-g`** flag be used to get symbolic and line-number information into the executable. (With the Sun Studio compilers, the **`-g`** flag does not appreciably change the generated code.)

The second step is to collect the data. The simplest way to do so is to prepend the **`collect`** command with its options to the command to run the application. The result of running **`collect`** is an experiment which contains the measured performance data. With appropriate options, the data collection process has minimum dilation and distortion, typically about 5%, but somewhat larger for MPI runs.

The third step in the user model is to examine the data. Both a command-line program, **`er_print`**, and a GUI interface, **`analyzer`**, can be used to examine the data. Much of the complexity introduced into the execution model of the code comes from optimizations and transformations performed by the compiler. The Sun compilers insert significant compiler commentary into the compiled code. The performance tools show the commentary, allowing users to understand exactly what transformations were done.

2 MPI Performance Issues

MPI programs run as a number of distinct processes, on the same or different nodes of a cluster. Each process does part of the computation, and the processes communicate with each other by sending messages.

The challenge in parallelizing a job with MPI is to decide how the work will be partitioned among the processes, and how much communication between the processes is needed to coordinate the solution. To address these aspects of MPI performance, data is needed on the overall application performance, as well as on specific MPI calls.

Communication issues in MPI programs are explicitly addressed by tracing the application's calls to the MPI runtime API. The data is collected using the VampirTrace [2] hooks, augmented with callstacks associated with each call. Callstacks are directly captured, obviating the need for tracing all function entries and exits, and resulting in lower data volume.

MPI tracing collects information about the messages that are being transmitted and also generates metrics reflecting the MPI API usage: MPI Time, MPI Sends, MPI Receives, MPI

Bytes Sent and MPI Bytes Received. Those metrics are attributed to the functions in the call-stack of each event.

Unlike many other MPI performance tools, the Sun Studio Performance Tools can collect statistical profiling data and MPI trace data simultaneously on all the processes that comprise the MPI job. In addition, during clock-profiling on MPI programs, state information about the MPI runtime is collected indicating whether the MPI runtime is working or waiting. State data is translated into metrics for MPI Work Time and MPI Wait Time. State data is available only with the Sun HPC ClusterTools™ 8.1 (or later) version of MPI, but trace and profile data can be captured from other versions of MPI.

2.1 Computation Issues in MPI Programs

The computation portion of an MPI application may be single-threaded or multi-threaded, either explicitly or using OpenMP. The Sun Studio Performance Tools can analyze data from the MPI processes using any of the techniques described in the previous sections for single- and multi-threaded profiles. The data is shown aggregated over all processes, although filtering can be used to show any subset of the processes. Computation costs are shown as User CPU Time (with clock-profiling); computation costs directly attributable to the MPI communication are shown as MPI Work time, a subset of User CPU Time. Time spent in MPI is shown as MPI Time, which represents the wall-clock time, as opposed to the CPU Time, spent within each MPI call.

2.2 Parallelization Issues in MPI Programs

Problems in partitioning and MPI communication can be recognized by excessive time spent in MPI Functions. The causes of too much time in MPI functions may include: load imbalance; excessive synchronization; computation granularity that is too fine; late posting of MPI requests; and limitations of the MPI implementation and communication hardware.

Many MPI programs are iterative in nature, either iterating on a solution until numerical stability is reached, or iterating over time steps in a simulation. Typically, each iteration in the computation consists of a data receive phase, a computation phase, and a data send phase reporting the results of the computation.

3 Using The Sun Studio Performance Tools to Analyze MPI programs

3.1 Using the MPI Timeline to Visualize Job Behavior

The MPI Timeline gives a broad view of the application behavior, and can be used to identify patterns of behavior and to isolate a region of interest.

3.2 Using MPI Charts to Understand Where Time Was Spent

The Analyzer's initial MPI Chart shows in which MPI function the time is spent.

The MPI Charts can be used to understand the patterns of communication between processes.

If some processes are running slower than others, or if the behavior is consistent over time, the MPI Charts provide a powerful way to explore these types of issues.

3.3 Using Filters to Isolate Behaviors of Interest

The MPI filters can be used to pick out behaviors of interest and determine which events are responsible.

References

1. Sun Studio Downloads, http://developers.sun.com/sunstudio/downloads/index.jsp
2. VampirTrace, http://www.tu-dresden.de/zih/vampirtrace