Chapter 5
Effective Open-Source Performance Analysis Tools

Prashobh Balasundaram
IBM Dublin Software Laboratories, Republic of Ireland

ABSTRACT

This chapter presents a study of leading open source performance analysis tools for high performance computing (HPC). The first section motivates the necessity of open source tools for performance analysis. Background information on performance analysis of computational software is presented discussing the various performance critical components of computers. Metrics useful for performance analysis of common performance bottleneck patterns observed in computational codes are enumerated and followed by an evaluation of open source tools useful for extracting these metrics. The tool's features are analyzed from the perspective of an end user. Important factors are discussed, such as the portability of tuning applied after identification of performance bottlenecks, the hardware/software requirements of the tools, the need for additional metrics for novel hardware features, and identification of these new metrics and techniques for measuring them. This chapter focuses on open source tools since they are freely available to anyone at no cost.

INTRODUCTION

Performance optimization of computational software is often an iterative and tedious process. The algorithms developed by the scientist/engineer may work well on one computer architecture and may need further performance tuning on another architecture. The need to compare performance characteristics across many hardware architectures is a routine task performed when benchmarking high performance computing systems for procurement. Open-source tools for performance analysis are suited to compare application performance characteristics across a wide range of computing

DOI: 10.4018/978-1-61350-116-0.ch005

architectures. The main goal of this chapter is to outline a few open-source development and performance analysis tools that the author found to be effective in a wide range of computing hardware.

Today, CPUs using the x86 or x86_64, power instruction set architectures are widely used for scientific high performance computing. Single core processors gave way to dual and quad core processors mainly due to power and thermal constraints. By 2011, six core to twelve core processors from Intel and AMD are expected to be widely used in HPC systems. Another clear trend in scientific computing hardware is the use of general purpose graphics processing units (GPGPUs) as accelerators for HPC applications. Programmed using high level programming languages like CUDA (Nvidia Corporation) and OpenCL implementations (Khronos group), these commodity products are redefining the architecture of HPC systems especially at the low and medium scale deployments. The capability supercomputing space is dominated by massively parallel distributed memory supercomputers. These machines often use multi-core processors running at relatively lower frequencies enabling higher packaging densities.

The most widely used programming model for distributed memory parallel computers is based on the Message Passing Interface (MPI). Most recent distributed memory machines use multi-core chips and therefore shared memory programming models like OpenMP (OpenMP ARB) is used to exploit parallelism at the node level. This helps applications scale better by reducing the inter-processor communication through the interconnect (Smith L & Bull M, 2000). As the number of processor cores in a single chip increases, contention for shared resources within a single multi-core processor becomes a major issue preventing linear scaling. Recent processor designs focus on techniques to reduce contention for shared processor resources like memory bandwidth. The use of accelerators in HPC systems introduces an additional layer of software and hardware components. Effective analysis of the performance characteristics of software on these hybrid systems is an active area of research.

Analysis and understanding of performance bottlenecks on scientific computing applications across systems of varying hardware architectures is an important task for HPC practitioners. Standards based open source performance analysis applications facilitate the comparison of application characteristics across many hardware architectures. They are available on a wide range of systems and often offer an independent toolchain in addition to those offered by the vendor of the HPC system.

Background

A well designed computing hardware may not always offer the best theoretical performance expected from an algorithm due many development/execution environment, software implementation issues. The most common development environment issues are related to inefficiencies in the operating system (OS), compiler used during development of the application.

OS noise or jitter (Tsafrir D et.al, 2005) is an important factor affecting the load imbalance and hence the scalability of HPC applications. The scalability of a parallel application follows Amdahl's law. According to the Amdahl's law, the speedup of a program using several processors for parallel computing is constrained by the sequential component. OS jitter affects massively parallel capability computing systems adversely, since it can occur at random on any processor of the system. The cumulative effect of this across the system impacts distributed memory clusters with thousands of cores and nodes. OS jitter is attributed to many causes including overhead of periodic OS clock ticks, timer interrupts used for maintaining control as well as system daemons. This issue is solved on modern supercomputers by using a specialized light weight operating system kernel tuned to reduce OS jitter. Some HPC