

Title Of Tutorial

ATTACKING PERFORMANCE BOTTLENECKS

Motivation

With CPU chip speeds peaking, simply upgrading the processor no longer offers the performance benefit that it did, leaving many of us facing a brick wall. This performance tutorial will teach you how to overcome this barrier and get more out of your applications using your *existing* system(s). Basically, there are two approaches to address a performance bottleneck: 1) software – modify the code itself to parallelize and optimize it; 2) hardware – increase the efficiency of utilization through scheduled resource allocation. Both approaches will be covered in this tutorial through detailed case studies presented jointly by Sun and Delaware Biotechnology Institute (DBI). Researchers from DBI will present several problems they (and their users) typically encounter and each problem will be solved based on software parallelization achieved through the OpenMP programming model or distributed resource management approaches. Solution instructions will be provided so those with similar problems can then use these approaches to address their own IT issues.

In recent years, clusters of relatively inexpensive servers have become the most popular solution to cope with a growing requirement for computational resources, although internal competition for storage space in the data center as well as access to those resources is becoming especially acute. Without an appropriate resource management policy, coupled with an approach to optimize application code, these growing clusters are either frequently idle and underutilized or oversubscribed. The advantages of using resource management software effectively are quite clear: increased use of compute resources, reduced wait time to access these resources, fewer administrative and maintenance problems, increased productivity and reduced overhead. Thus enabling scientists to conduct more ambitious studies (such as protein folding), allows them to spend more time on research and less time managing their network, and permits them to submit and secure larger grants focused on more complex computational problems.

Topics Included

Application performance, performance analysis tools, parallel architectures, parallel programming models, OpenMP, computational biology, Beowulf clusters, and grid technology will be covered. All topics, including every example and proposed solution, are applicable to all contemporary computer systems (hardware, network interconnects, storage, operating systems, *etc.*). Thus, all instructions presented will have broad appeal and application.

Tutorial Level

Medium

Intended Audience

System administrators supporting life sciences research teams as well as application developers should especially benefit from this tutorial, but anyone with an interest in performance tuning or grid computing for application in computational biology is targeted. Basic experience with a distributed resource management software system (SGE, LSF, PBS, *etc.*) , experience in a higher-level programming language, and some knowledge of UNIX system administration will be helpful but are not required. The topics and concepts covered will be generally applicable across all sectors and aspects of computational biology.

Goals And Objectives

Attendees will be introduced to two methods of addressing performance bottlenecks: a) software optimization; b) hardware optimization through increased utilization. In many cases, all it takes is some basic understanding of the underlying mechanisms and some instruction on which tools are the right ones to use and when to use them.

- Demonstrate different ways to overcome performance problems
- Show how parallelization can be applied to improve application performance
- Explain how to apply Grid technology to better utilize existing compute resources
- Demonstrate some performance problems common to computational biologists (including those that relate to resource allocation in large Beowulf clusters)
- Provide some samples of grid-based solutions to address common problems

Following this tutorial, attendees should be capable of parallelizing basic software applications using the OpenMP programming model and applying the distributed resource management techniques learned in this tutorial to their own clusters with little or no modification (depending on their context).

Detailed Outline Of The Presentation

Today's systems are potentially very powerful with microprocessor speeds of at least 1 GHz and are capable of executing multiple instructions every clock cycle, giving rise to a theoretical peak performance in the range of billions of operations per second. Unfortunately, the realized performance of a microprocessor is typically only 5-10% of the theoretical performance. The main reason for this discrepancy is the substantial difference between the processor speed and the time needed for memory to supply data to the processor. This gap can easily be a factor of 50, or more. But, with the right tools and some basic knowledge of the CPU architecture of a contemporary microprocessor, application performance can be boosted substantially.

After a brief introduction into parallel architectures and parallel programming models, we will zoom in on OpenMP (<http://www.openmp.org>). This is an emerging *de-facto* standard for portable shared memory parallel programming. All major hardware vendors, as well as several independent software companies, now offer OpenMP-compliant compilers. OpenMP offers a compact, elegant, but powerful way of parallelizing an application for a shared memory system. Unlike other popular parallel programming paradigms such as Posix Threads, the explicit threading model of Java and the Message Passing Interface (MPI), OpenMP preserves the sequential version of the program and remains fully portable to any platform. These key features and others will be presented in the context of examples of how and when to use OpenMP in practical situations.

In the second half of our tutorial, key distributed resource management (DRM) technologies (LSF, SGE, PBS) and how they differ, as well as key grid concepts including definition with respect to Life Sciences, policy allocation, resource reservation, backfilling, accounting, reporting, and billing will be discussed. Examples of public/private grids, current needs and issues of the life science community, and future features requested by users of these technologies will be highlighted.

Next, DBI will present a series of real-life IT-based problems in computational biology coupled with solution instructions that attendees will be able to replicate and apply to their own situations. Key DRM concepts, including policy allocation, resource reservation, backfilling, accounting and reporting, and billing will be studied through these use-cases. Although the scale will be limited to an 8 CPU cluster, live demonstrations will be used throughout the tutorial to better illustrate how to execute the instructions that will be provided.