

The Sun Application Tuning Seminar An Overview

Summary - *The Sun Application Tuning Seminar shows end-users of Sun UltraSPARC®, AMD and Intel based single as well as multicore systems how to get good performance out of their application. The seminar covers how to use the Sun development environment in an optimal way, as well as general optimization and parallelization techniques to improve performance on multicore systems.*

The general philosophy of the seminar is to build up understanding of key concepts that are relevant to obtain good application performance. Once this is achieved, it is much easier to use a development environment in the best possible way.

The seminar covers serial optimization and shared memory parallelization using automatic parallelization by the compiler, as well as the de-facto standard OpenMP shared memory programming model. The latter is very well suited to develop parallel applications for multicore and other shared memory architectures.

Other than some programming experiences (preferably in C or Fortran), no specific background in application tuning is assumed.

Examples given are in C and/or Fortran, but they are simple enough to be understood by anybody with some programming experience in another programming language.

In addition to the presentations, lab sessions are organized. These are meant to challenge the attendees to apply the theory in practice.

1. Objectives

After this three day seminar, attendees should be much more knowledgeable and know how to:

- Get good performance out of a Sun system
 - i. Use the Sun Compilers and Performance Analyzer to optimize an application
 - ii. Understand the underlying mechanisms that impact performance
- Use the Sun compilers to automatically parallelize a program
- Use the shared memory OpenMP programming model to explicitly parallelize an application for multicore and other shared memory architectures
- Use the Sun tools to speed up development of OpenMP applications

Attendees should also be able to give practical advise to others what to do and what not to do when it comes to making their program(s) run more efficiently.

2. Target audience

This seminar primarily targets people that are interested to learn more about the technical details how to tune the performance of a technical-scientific application.

It is assumed attendees have some practical programming experiences and basic Unix skills (e.g. editing a file, moving and copying files, etc), but there are no assumptions regarding any background in application tuning. All concepts are built up from the ground up.

The examples and the lab exercises are in C and/or Fortran. Therefore some experience in either one of these languages is of help, but it is not a requirement.

Even though C++ is not covered explicitly, we believe that C++ programmers may benefit from the seminar as well.

The focus is on serial performance and shared memory parallelization using the OpenMP programming model. The Message Passing Interface (MPI) programming model is not covered.

3. Lab Sessions

The seminar is practically oriented. In order to benefit most from the material presented, several lab sessions are included as part of the seminar. Fortran and C lab exercises are made available. These are very simple to understand, but have been chosen to highlight a specific feature discussed in the seminar. Attendees are encouraged to work on these examples.

4. Dress code

The seminar is rather intense and comfortable cloths may be most appropriate. However, we prefer not to impose any preferences regarding this. Please dress in the way you feel most comfortable with.

5. Seminar language

The seminar is given in English.

6. The Seminar Topics

In this section we list the topics covered in the seminar. Per topic, a short description is included.

Introduction

After a brief overview of the seminar contents and scope, several pointers to more information are given.

Prologue

Some of the basic terminology used throughout the seminar is introduced and explained. Next, all factors that affect application performance are discussed in a global manner. Throughout the remainder of the seminar, much more details on these topics are given.

The Memory Hierarchy

Today's microprocessors make heavy use of caches to bridge the gap between the speed of the processor and main memory. The resulting memory hierarchy can be rather complex. Often, tuning an application is about using the entire memory hierarchy in the best possible way.

In this module we explore the memory hierarchy in a reasonable level of detail. The concepts and characteristics presented here are needed in the remainder of the seminar.

Processor Architectures

This module mainly provides background information on various Sun UltraSPARC, AMD and Intel single and multicore architectures. When tuning the performance of an application, one often does not need to know the details of underlying microprocessor.

Only those characteristics relevant when tuning an application are covered. The focus is on the multicore aspects of the various architectures, plus some details on the memory hierarchy.

The Sun Compilers

The Sun compilers are supported on Solaris and Linux. They can be used to compile applications for (Ultra)SPARC, AMD and Intel processors. The compilers provide a comprehensive set of options. In this extensive module we focus on those options that are relevant for performance. As explained and discussed, several categories of optimization switches are available.

In addition to compiler options, one can also use the highly tuned mathematical libraries that are bundled with the compilers.

Sun Application Tuning Seminar Overview

Frequently used intrinsic operations have been optimized extensively. In addition to this, vectorized versions of important intrinsic functions are available.

The Sun Performance Library not only provides tuned (and shared memory parallel) versions of the BLAS1, BLAS2, BLAS3, LAPACK, FFTPACK and VFFTPACK public domain packages, but also contains additional functionality for sparse matrices, convolutions, correlations, etc.

The library can be called both from Fortran and C programs. Fortran 95 applications can take advantage of parameter type checking and use generic calls. C programmers benefit from the native C interfaces.

For completeness, we also cover some options that can be helpful when porting and/or debugging an application.

The Sun Performance Analyzer

The Sun Performance Analyzer is a powerful and useful tool to analyze the performance of an application. It can be used on existing, non-instrumented, executables and provides important information on the behavior of the application. If one is willing and able to recompile the application with the -g option added, performance and compiler optimization information at the source line level are given.

The Performance Analyzer is available for Solaris and Linux. It supports Fortran, C, C++ and Java. Automatic parallelization, POSIX threads, MPI and OpenMP are supported as well. In this seminar we focus on the Fortran and C languages. With respect to parallelization, support for Automatic Parallelization and OpenMP is covered

Extensive post-processing features on the statistics gathered allow the user to obtain insight into the performance characteristics of the application.

Several metrics (CPU time, wall clock time, system time, etc.) can be obtained. The performance analyzer also gives easy access to the on-chip hardware event counters supported on the various processors supported.

Serial Optimization Techniques

One may be tempted to think that serial optimization is not worth considering, because that is an area well under control. Even though a lot of progress has been made, there is often room for significant improvement in many applications.

Serial performance is important for several reasons:

- Existing resources are used more optimally, typically without additional cost
- The performance increase can be noticeable
- If parallelized, a program tuned for sequential performance usually also gives better performance using many cores/processors

A variety of techniques to optimize specific programming constructs have been developed over the years. Nowadays, the most important ones have been integrated into compilers and

Sun Application Tuning Seminar Overview

are part of standard optimizations. However, even though compilers get more sophisticated over time, they sometimes lack information to make the optimal decision.

Therefore we cover all these key optimization techniques such that users can decide for themselves what should be done to assure good performance, with or without the help of the compiler.

Instruction scheduling is another important aspect when it comes to getting good application performance. Sophisticated techniques to exploit the superscalar architecture of a microprocessor and take advantage of pipelined operations are needed to obtain good performance.

The Sun compilers implement such techniques. Using the Modulo Scheduler as an example, instruction level optimization is explained.

The Compiler Commentary is one of the features of the Sun compilers. With the help of these messages, the user can obtain details on the optimizations performed by the compiler. Several examples how to use this information in order to tune an application are presented.

Case Studies Serial Optimization

By now, one should have a good understanding of what the compilers can do, how to use them in the best way and what possible source code optimization techniques could be applied to enhance performance even further.

This section concludes the serial performance tuning part. We present some case studies through which we hope to show the tuning methodology. The optimizations implemented go beyond what can be expected from a compiler.

We demonstrate that, with a modest effort, rather dramatic performance improvements can be realized. Several examples are based on real-life applications.

Introduction Into Parallelization

The basics of parallel processing are introduced and explained. This module serves as a basis for the remainder of the seminar. Several important concepts (speed-up, Amdahl's law, load balancing, data dependences, cache coherency, parallel architectures, programming models, data races, etc.) are introduced and discussed.

Automatic Shared Memory Parallelization

In order to safely parallelize a specific program construct, one has to know whether a data dependence exists or not.

With automatic parallelization, the compiler has to answer the question whether the candidate loop (nest) has such a dependence.

The Sun compilers support automatic parallelization. We present and discuss the compiler options for automatic parallelization, and show several examples in C and Fortran.

An Overview of OpenMP

It might happen that the compiler lacks information to resolve a data dependence, or that the user would like to implement a rather high level of parallelization. In such a situation, one can pass on additional information to the compiler through so-called directives.

Nowadays, the de-facto OpenMP standard provides a compact, but powerful, environment to implement such explicit parallelization.

We cover the OpenMP model in quite some detail, both for C and Fortran. Examples to illustrate the most important concepts are presented and discussed.

This module concludes with an overview of the Sun specific enhancements to the OpenMP environment, plus an overview of the Sun Thread Analyzer, a tool to detect data races and deadlock in a shared memory application.

OpenMP and Performance

Similar to serial optimization, parallel performance becomes relevant once the program is ready from a functional point of view and produces correct answers.

Therefore, after having used OpenMP to parallelize the application, it makes sense to look into performance.

After an overview of the main inhibitors to scalable performance, we conclude with some examples to show how to efficiently parallelize a program using the OpenMP model¹.

In particular we focus how to apply OpenMP constructs to implement a higher level of parallelism.

¹ *We do not discuss how to design and write parallel algorithms. This is beyond the scope of the seminar.*