Oracle Solaris Studio
Performance Analyzer

The Oracle Solaris Studio Performance Analyzer provides unparalleled insight into the behavior of your application, allowing you to identify bottlenecks and improve performance by orders of magnitude. Bring high-performance, high-quality, enterprise applications to market faster and obtain maximum utilization from today’s highly complex systems.

KEY FEATURES
- Low overhead for fast and accurate results
- Advanced profiling of serial and parallel applications
- Rich set of performance data and metrics
- Easy to use GUI
- Remote and cross-platform analysis
- Supports C, C++, Fortran and Java

KEY BENEFITS
- Maximize application performance
- Improve system utilization and software quality
- Increase developer productivity

Introduction
Is your application not performing optimally? Are you taking full advantage of your high-throughput multi-core systems? Understanding the performance of enterprise applications is not a simple task. Enterprise systems are created for extreme scalability, and high performance for demanding and consolidated enterprise workloads. Creating optimally tuned applications in these complex environments is a daunting task. Oracle Solaris Studio provides compilers and libraries that can tune an application for the best code sequences, but understanding performance characteristics of complex, long running, inter-dependent applications is challenging. The Performance Analyzer is a powerful market leading performance analysis tool for developers who need to optimize application performance and scalability. It provides in-depth analysis that enables you to quickly understand your application’s behavior, allowing you to easily eliminate hotspots, bottlenecks and areas of high resource consumption.

Analyzer Serial and Parallel Applications
The Performance Analyzer is optimized for a variety of programming scenarios and helps pinpoint the offending code with the fewest number of clicks. In addition to supporting C, C++, and Fortran applications, the Performance Analyzer also includes support for Java code. All data is converted into function source, lines of code or disassembly code with caller/callee values. Additionally, compiler transformations are interpreted to show how the optimizer transforms source code. For Java programs, the Performance Analyzer highlights which methods are most expensive and how they are affected by Hotspot compilation and how memory allocation and garbage collection affect behavior. The Performance Analyzer also handles C++ and Java mangled names.

The Performance Analyzer can be used to profile single-threaded as well as multi-threaded applications written using PThreads, Solaris threads and OpenMP. These multi-threading models are as very common as underlying chips offer and increasingly
larger numbers of cores and hardware threads. Thus, application performance is emerging as a critical factor, with bad performance increasingly considered a program failure. The Performance Analyzer highlights how threads are being used and how expensive it is to synchronize them, enabling users to understand which loops can be parallelized and what the obstacles are to parallelizing them.

The Performance Analyzer also handles multiple concurrent processes to collect system-wide performance data, providing application insights down to the OS kernel, presenting a graphical identification of bottlenecks and helping fully optimize application performance. In addition, it profiles fully optimized and parallelized production code.

**Advanced Performance Metrics**

The Performance Analyzer provides a rich set of data and metrics for diagnosing a wide variety of performance problems and probes applications with very low overhead. It records a variety of important metrics as a series of events. Each metric includes event-specific data and a call stack, thread-id, cpu-id and high-resolution timestamp. These metrics are then attributed to functions, source lines or (disassembled) instructions:

- **Clock-based Profiling**: Statistical in nature, this metric collects information on kernel accounting microstates and supports the user CPU time metric.
- **Hardware Counter (HWC) Overflow Profiling**: Also statistical in nature, this metric is supported with special registers in hardware that count specific hardware events. The most useful counters indicate the behavior of the memory (cache) subsystems. The latest multi-core systems contain 200+ hardware counters and are fairly extensive.
- **Dataspace and Memoryspace Extensions to HWC Profiling**: Dataspace profiling enables users to determine which data structures are responsible for cache misses. Memoryspace profiling allows users to find hot cache lines, and understand which data addresses, threads, and instructions are making them hot. On machines such as SPARC T4, with precise HW counter interrupts for memory counters, no special compilation is needed, and memoryspace profiling allows the detection of false sharing of cache lines.
- **Thread Synchronization Delay Tracing**: Provides an understanding of the efficiency of multithreading in a program.
- **Samples and Execution Statistics**: Allows users to sample their application at specific points in time. The execution statistics provide a global view of the program including microstate accounting data, time stamps and kernel monitors.
- **Kernel Profiling**: Provides information on how the operating system kernel interacts with the application, using Oracle Solaris DTrace technology. User processes can also be simultaneously profiled along with the kernel. Kernel profiles can be based on either clock-ticks or HWC-overflow events.
- **Java Profiling**: Java programs often contain Java code mixed with C/C++ native code as well as Hotspot compiled methods. This metric seamlessly combines and reconciles two callstacks, the Java stack and the machine stack, into a single view.
- **Heap Tracing**: Tracks Heap (malloc, calloc, free) usage and identifies memory leaks or inefficient allocations in user code.
• **I/O Tracing:** Allows you to identify I/O patterns in your application and quickly pinpoint I/O bottlenecks that may be impacting application performance.

• **Cycles per Instruction (CPI) / Instructions per Cycle (IPC) Tracing:** Provides metrics to track CPI and IPC to help you identify where your application is running efficiently or inefficiently. These metrics are available when you perform hardware counter profiling on your application and specify the counters for cycles and instructions.

• **MPI Tracing:** Identifies causes of communication delays in MPI applications.

**Intuitive and Easy to Use Graphical Interface**

The Performance Analyzer identifies application performance bottlenecks, by specifying not only which functions, code segments, and source lines are having an impact on performance but by also providing an easy-to-use GUI to tune for optimal performance. The GUI provides advanced sorting, filtering and timeline visualization capabilities for rapid identification of performance bottlenecks. The Performance Analyzer GUI is organized around data views that each show a different perspective of the performance metrics for your profiled application.

![Figure 1. Overview Screen shows a summary of performance metrics for the loaded experiment](image-url)
Figure 2. Timeline View allows you to visualize performance hotspots, shown here with a multi-threaded application.

Figure 3. Functions View helps you understand which functions are taking the most time.
Figure 4. Source View allows you to profile data on your source code, also includes syntax highlighting based on the source language and hyperlinks for the caller and callee functions.

Remote and Cross-Platform Analysis

The Performance Analyzer provides support for remote analysis, allowing you to profile applications, collect performance data, and view experiments on a remote server from a Linux, Windows, MAC or Solaris client environment. In addition, you can read experiments recorded on any platform with cross-architecture analysis support. The Performance Analyzer is a powerful and robust application profiling tool with industry-leading functionality to help you efficiently optimize enterprise applications for maximum performance and scalability.

CONTACT US
For more information about Oracle Solaris Studio, visit oracle.com/goto/solarisstudio or call +1.800.ORACLE1 to speak to an Oracle representative.

Hardware and Software, Engineered to Work Together

Copyright © 2014, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.