Multi-threaded Software Development with the Intel® Threading Tools

A Half-Day Tutorial at PACT
September 17, 2005

Tutorial Organizers

Douglas Armstrong, Zhiqiang Ma, Paul Petersen, Sanjiv Shah
Intel Americas, Inc.

Abstract

Multi-threaded programming has been used in the software industry to improve the responsiveness of GUI applications, to improve the throughput of network based applications, and to reduce the turnaround time of compute bound applications through the use of parallelism.

To encourage the software industry to expand the usage of multi-threading, Intel has being pushing the development of tools and languages which ease the burden for the software developer. Multi-threaded programming enables new capabilities, but also comes with additional challenges. These challenges include how to specify the parallelism that you want to exploit with multi-threading, how to find defects in multi-threaded software and how to tune the performance of the multi-threaded software to make sure the developer is achieving the goals of redesigning the software to enable parallelism.

This tutorial will walk through a Methodology for multi-threading software development that leverages the Intel® Threading Tools to dramatically reduce the time required to utilize multi-threading.

Tutorial Outline

1. Multi-threading Methodology
   a. Specification (languages and libraries)
   b. Correctness (debugging and analysis)
   c. Performance (tuning and bottle-neck detection)
2. Intel® Thread Checker Overview
   a. System Architecture
   b. Theory of Operation
   c. Capabilities
3. Intel® Thread Profiler Overview
   a. System Architecture
   b. Theory of Operation
   c. Capabilities
4. Case Studies and Examples

Audience

This tutorial is appropriate for those interested in the area of multi-threaded programming and software development. Some basic familiarity with multi-threaded software development is assumed, with the focus of the tutorial on how to improve the development process by leveraging a systematic progress that is enabled by the Intel® Threading Tools.

Biographies

Intel's Parallel and Distributed Solutions Division (PDSD) within the Software and Solutions Group was created following Intel's acquisition of Kuck & Associates, Inc (KAI) and Pallas, GmbH. The Intel® Threading Tools are the evolution of the KAI KAP/Pro Toolset work which included the Assure analysis tool for the detection of data-races in OpenMP applications, and the GuideView analysis tool for performance analysis of OpenMP applications. The Intel Threading Tools (www.intel.com/software/products/threading/) continue this approach to parallel and multi-threaded software analysis. The Intel® Thread Checker is designed to analyze applications written in OpenMP, POSIX Threads (for Linux*) or WIN32 Threads (for Microsoft Windows*) to detect threading related issues such as incorrect lock usage, data-races, deadlock, and incorrect API usage. The Intel® Thread Profiler is designed to detect inefficiencies in OpenMP, POSIX or WIN32 applications where either an incorrect amount of parallelism is used (either too few, or too many threads), or the interactions between the thread via synchronization instructions are not as efficient as needed.

All of the tutorial organizers are currently employed by Intel in the Parallel and Distributed Solutions Division.

Douglas Armstrong is a software engineer and team manager. At Intel he has worked on the parallel run-time components of the Intel OpenMP implementation. He currently works on the Intel® Thread Profiler focusing on design and development for the performance analysis of multithreaded applications.

Zhiqiang Ma is a software engineer. At Intel he has worked on the KAI C++ compiler. He currently works on the Intel® Thread Checker focusing on dynamic software validation and error detection of multithreaded applications.

Paul Petersen is a senior principal engineer. During his career at KAI and Intel he has worked on the design and implementation of performance and correctness analysis tools, languages, compilers and runtimes for parallel applications and systems.

He earned a PhD. degree in Computer Science from the University of Illinois at Urbana-Champaign, and is a member of ACM.

Sanjiv Shah is a senior principal engineer and a group manager. He manages the Intel® Threading Tools product line. During his career at KAI and Intel, Sanjiv has worked on many aspects of performance and correctness analysis tools, compilers and runtimes for parallel applications and systems.

Sanjiv has been extensively involved in the creation of the OpenMP specifications and of the OpenMP Architecture Review Board. He is the CEO of the OpenMP ARB and serves on its Board of Directors.

He earned both Bachelor's and Master's degrees in Computer Science & Engineering from the University of Michigan.