1-512-256-0197
www.mindshare.com
training@mindshare.com

Intel x86 Performance Analysis for Software Engineers

Let MindShare Bring "Intel Performance Analysis" to Life for You

This course is offered in two variants – a Linux variant and a Windows variant, reflecting the differences between the tools in the two different environments. The class will examine the performance of application code executing on the processor cores, with the assistance of the various tools available, and explore ways to optimize the performance. The class will include live demonstrations of the tools, allowing the students' questions to drive the demonstrations. The processors discussed in the class are Intel's core architecture.

You Will Learn:

- The x86 architectural features impacting the performance of application code, and possibilities for taking advantage of them:
    - The cache system
    - Multiple processor cores
    - Out-of-order execution of instructions
    - Virtual memory mappings and the TLB
- Some of the tools available for analyzing the performance of application code
- The methodology for getting to good performance

Note, this is not a operating system tuning class, this is a programmer's application code optimization class. The examples analyzed are all applications, but the ideas are also relevant to kernel code. There are no student exercises included as part of the class.

Course Length: 3 days

Course Outline:

- **Power and performance**
- **Importance of algorithm and data structures – design for performance**
- **Methodology**
- **Advantages and disadvantages of optimizing to a particular processor implementation**
    - ways that code can be made to be more flexible
- **The memory system**
    - Caches
    - Multi-processor
    - TLBs
    - Cache line bounce
    - Using the cache effectively
- **Microarchitectural optimizations**
    - Core pipeline overview
    - Out-of-order execution
    - Handling "slow" instructions
    - Execution serialization (e.g. microcode, MMIO), lightweight serialization
    - Issues with MMIO
    - Example optimizations, e.g loop unrolling
- **Hyperthreading**
- **Multiple processor cores**
    - Distribution of the workload across cores
    - Core affinity
    - Sharing issues
    - Synchronization issues

- **Compiler optimizations (Windows)**
    - Microsoft compiler
    - Intel compiler
- **Compiler optimizations (Linux)**
    - GCC
    - Intel compiler
- **Hotspot analysis, using perf (Linux)**
- **Hotspot analysis using VTune (Linux)**
- **Use of vector instructions (SSE, AVX)**
    - Benefits of using SSE/AVX even for applications where it may not seem appropriate
    - Loop ordering and data structures, to optimize cache behavior
    - Automatic vectorization
- **Analysis using performance counters**
    - Cache misses, with perf (Linux), with WPR (Windows)
- **Profile Guided Optimization**
- **System Topics**
    - Interrupts
    - Direct cache injection
- **Overview of virtualization optimization**
    - Analysis in a virtualized environment
    - Ways to make code "virtualization friendly"
    - Virtualization issues – caches, TLBs, multiple levels of page tables
    - Virtualization and I/O
        - Intel VT-d
        - PCIe SR
        - interrupt delivery

Recommended Prerequisites: Some experience of reading Intel assembly language is desirable, as the class will look at assembly code to see the details of execution. Familiarity with Intel processors and system architecture is also assumed.

Course Material: Students will be provided with electronic version of the slides.