



Multi-core Faculty Training – Detailed Agenda [3 days]

Day 1

Welcome and Class Overview

Agenda review, introductions, logistics

Multi-core Architecture and Fundamental Tools

Objectives	
1	Be aware of and have access to several hours of multicore topics including <ul style="list-style-type: none"> • Architecture • Compiler Technology • Profiling Technology • Open MP • Cache Effects
2	Be able to take advantage of foil ware and labs to teach students how to get started in threading with OpenMP
3	Be able to create exercises on how to avoid coding common threading hazards associated with some multicore system – such as port cache utilization, false sharing and threading load imbalance
4	Be able to create exercises on how to use selected compiler directives and switches to improve behaviour on each core
5	Be able to create exercises on how to take advantage of underlying architecture to exploit parallelism within each core of a multicore system using SSE
Agenda	
1	Multi-core motivation
2	Fundamental Tools Overview
3	Taking advantage of multi-core
4	Taking advantage of parallelism within each core (SSEx)
5	Avoiding memory/cache effects

Parallel Programming Patterns

Objectives	
1	Describe the concepts behind the design patters and parallel design patterns
2	Given serial code or algorithms, choose the better algorithm structure design patter (either Task Parallelism or Geometric Decomposition) to be used in the threading the code and defend you choices
3	Given serial code or algorithms, choose the better supporting structure design patter (either SPMD, Loop Parallelism or Boss/Worker) to be used in threading the code and defend your choices.
Agenda	
1	Pattern Language Structure
2	Task Parallelism Pattern
3	Geometric Decomposition Pattern
4	Support Structures <ul style="list-style-type: none"> • SPMD – Same Program, Multiple Data • Loop Parallelism • Boss-Worker

Programming with Windows* Threads

Objectives	
1	In-depth introduction to multithreaded programming and techniques using Windows* Threading API
Agenda	
1	Thread creation, termination, and control
2	Data scoping; thread local storage methods
3	Mutual Exclusion and synchronization techniques

Day 2

Threading for Performance with Intel® Threading Building Blocks (3 hours)

Objectives	
1	List the different components of the Intel Threading Building Blocks (TBB) library
2	Code parallel applications using TBB
Agenda	
1	Intel® Threading Building Blocks background
2	Generic Parallel Algorithms: <ul style="list-style-type: none">• parallel_for (with programming activity)• parallel_reduce (with programming activity)• parallel_sort example
3	Task Scheduler (with programming activity)
4	Generic Highly Concurrent Containers (with programming activity)
5	Scalable Memory Allocation (with programming activity)
6	Low-Level Synchronization Primitives

Intel® Thread Checker (2 hours)

Objectives	
1	Use Thread Checker to detect and identify a variety of threading correctness issues in Windows* threaded applications
2	Determine if library functions are thread-safe
Agenda	
1	What is the Intel Thread Checker?
2	Race Conditions and how Thread Checker identifies them
3	Using Thread Checker as a threading assistant
4	Other error conditions Thread Checker can identify
5	Thread-Safety of library routines
6	Other features of Thread Checker

Intel® Thread Profiler for Windows Threads (2 hours)

Objectives	
1	Use Thread Profiler to recognize and fix common performance problems in Windows* threaded applications
Agenda	
1	What is the Intel Thread Profiler?
2	Critical Path Analysis
3	Data Views within Thread Profiler
4	Common Performance Issues of multithreaded applications
5	Focus on Load imbalance
6	Focus on Synchronization contention
7	General optimizations to gain better performance

Day 3

Multithreaded Programming Methodology (3 hours)

Objectives	
1	Understand the evolution of parallel processing architectures
2	Show how threading architectures relate to software development
3	Be able to rapidly prototype and estimate the effort required to thread time consuming regions
Agenda	
1	Processor and platform support for multithreaded applications
2	Four-step threading methodology <ul style="list-style-type: none">Analyze – find code locations for threadingDesign and implement code to threadTest for Correctness – ensure threading has not introduced errorsTune for performance – find performance bottlenecks from threads
3	Simple example code used for illustration of methodology

Scalability of Threaded Applications (3.5 hours)

Objectives	
1	Understand the need for designing multithreaded applications for scalability to take advantage of an increasing number of available cores
2	What tools are available to measure and predict scalability
3	How several different factors can inhibit scaling of applications on increased number of cores
Agenda	
1	Why focus of scalability? <ul style="list-style-type: none">Measuring and estimating scalabilityWhere would you start?
2	Tools for scalability analysis
3	Factors inhibiting scalability <ul style="list-style-type: none">Serially dominant workloadsGranularity and Parallel OverheadLoad ImbalanceSynchronization IssuesMemory Related IssuesI/O