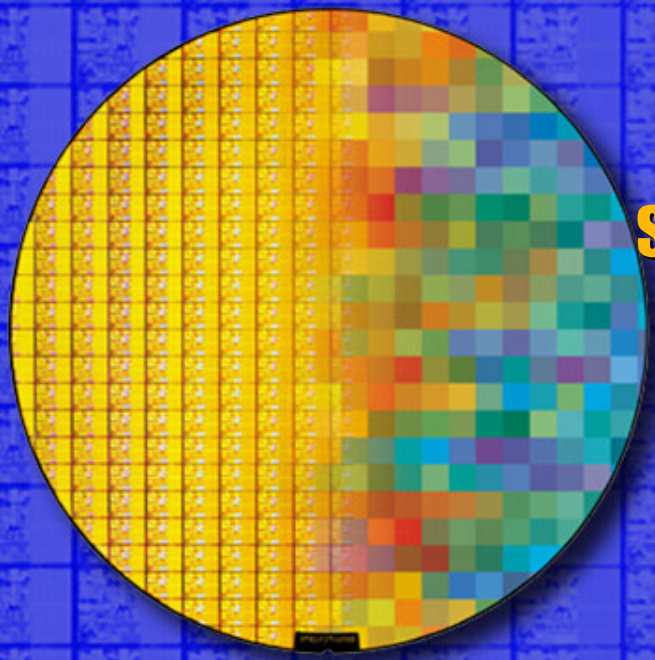


Thread Checker:

Saving Valuable Time Otherwise Spent Searching for Hard-to-find Threading Errors

Koby Gottlieb

Software Solution Group

Intel Corporation

intel

Agenda

- Threading: Why do we need it now?
  - The Dual core future
- Challenges Unique to Threading
- Intel® Thread Checker product
  - Features
  - Demo
  - How does it works?
- The Thread Profiler feature

Driving Parallelism

Moving from Chips/Computer to Computers/Chip

	2004	2005	2006*
Desktop (Perf)	55% HT	Shipping Dual-Core	> 40% Dual-Core
Servers	100% HT	Shipping Dual-Core	> 85% Dual/Multi-Core
Mobile (Perf)		Shipping Dual-Core	> 70% Dual-Core

"Exiting 2006, we believe that over 40 percent of the desktop product shipments will be dual core, over 80 percent of our server products will be multi or dual core, and over 70 percent of our mobile products will be dual core as well. We are dedicating all of our future product designs to multi-core environments. We have bet on this in terms of our software environment, our ecosystem development, and our Intel Capital infrastructure around it. We believe this is a key inflection point for the industry."

PAUL OTELLINI, IDF 7/2004

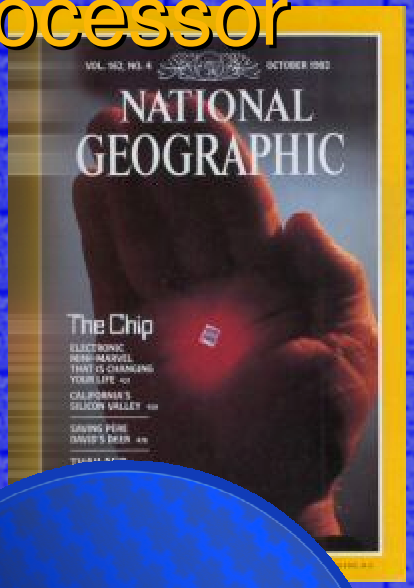
All CPU development on Dual/Multi-Core

Unique Dual/Multi-Core Products in all segments

intel

* data is run rate exiting the year

Next generation Dual-Core Itanium® processor



"Eventually **one billion** transistors, or electronic switches, may crowd a single chip, 1,000 times more than possible today."

National Geographic, 1982

Montecito

1.7B transistors

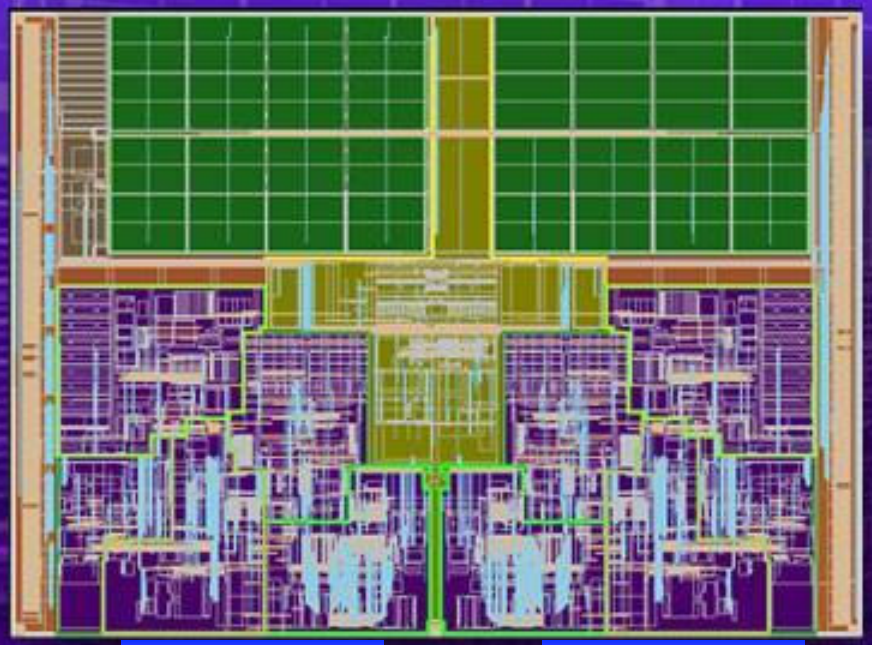
Next Generation – Montecito

- Dual Core *and* Multithreaded
- >3X increase platform bandwidth
- Higher performance, lower power
- 24MB Cache

intel

Sources: [1]IDC Server Tracker Q3'03, [2]HP RX1600 2p/2c configuration *Other names and brands may be claimed as the property of others

Future Capabilities
Napa Platform



Yonah: First mobile optimized Dual-core processor design from the ground up on 65nm:

LT, VT

Calistoga chipset: Integrated graphics for superb playback

Golan: Next generation wireless solution

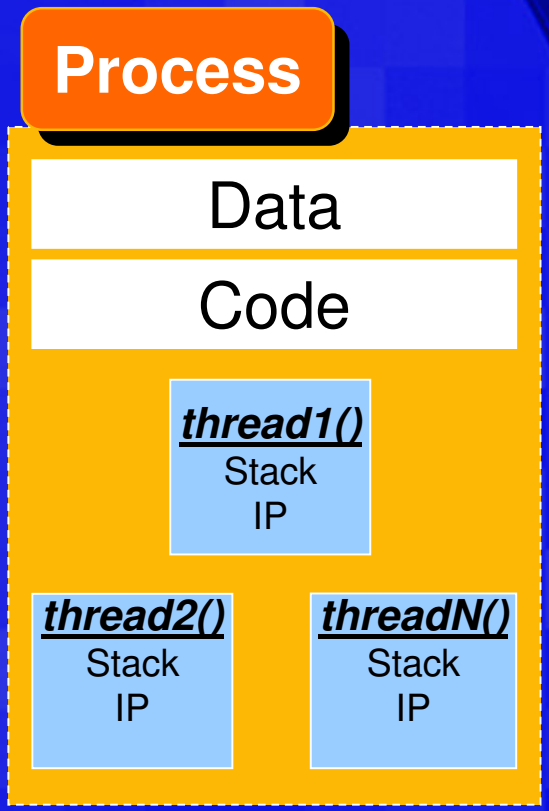
Dual-Core Systems

- One package with 2 cores
- Software impact
  - 2 Cores → 2 processors
  - 2 Cores → 2x resources

Use threads to exploit full resources of dual core processors

Threads Defined

- OS creates process for each program loaded
  - Each process executes as a separate thread
- Additional threads can be created within the process
  - Each thread has its own Stack and Instruction Pointer
  - All threads share code and data



Threading Software

- OpenMP* threads
  - http://www.openmp.org/
- Windows* threads
  - http://msdn.microsoft.com/
- POSIX* threads (pthreads)
  - http://www.ieee.org/

If both cores fully busy, then 2x speedup possible

Correctness Bug: Data Races

- Suppose: a=1, b=2

Thread1	Thread2
x = a + b	b = 42

- What is value of x if:
  - Thread1 runs before Thread2? x = 3
  - Thread2 runs before Thread1? x = 43
- Data race: concurrent read, modify, write of same address

Outcome depends on thread execution order

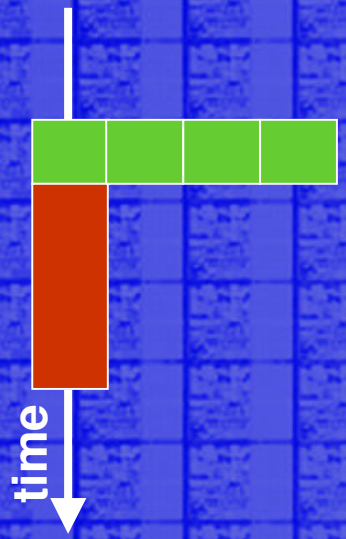
Solving Data Races: Synchronization

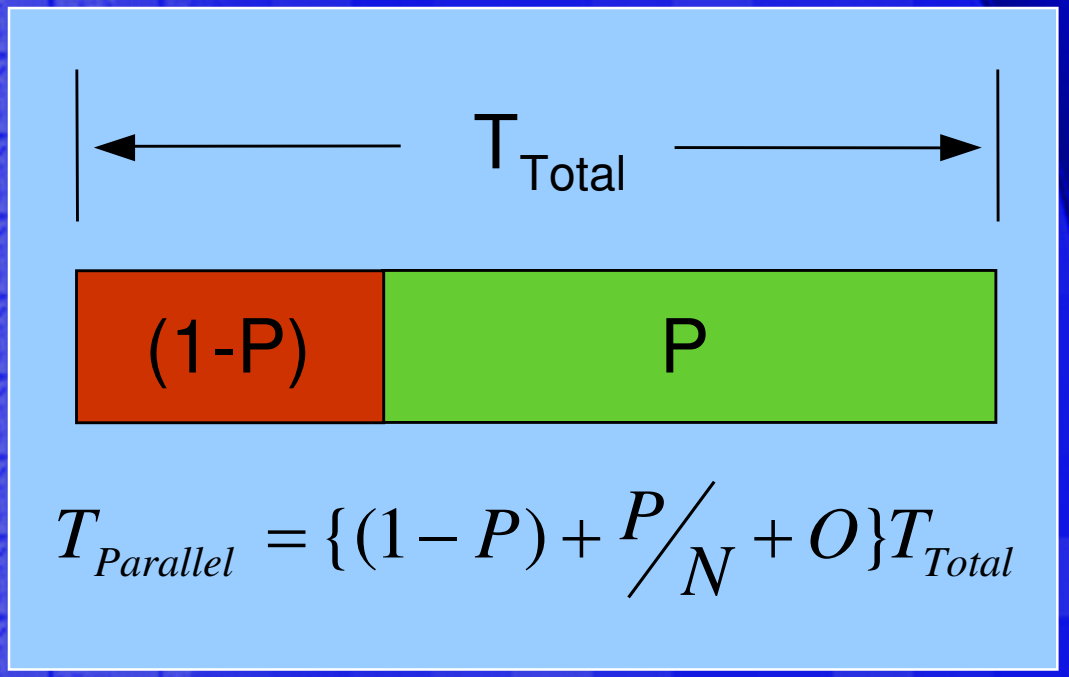
Thread1
Acquire(L)
a = 1
b = 2
x = a + b
Release(L)

Thread2
Acquire(L)
b = 42
Release(L)

- Acquisition of mutex L ensures atomic access
  - Only one thread can hold lock at a time
- Example APIs:
  - EnterCriticalSection(), LeaveCriticalSection()
  - pthread_mutex_lock(), pthread_mutex_unlock()

Amdahl's Law





$$T_{Parallel} = \{(1-P) + P/N + O\}T_{Total}$$

If only 1/2 of the code is parallel, 2X speedup is unlikely

P = parallel portion of process

N = number of processors (cores)

O = parallel overhead

Threads Intro New Class of Problems

- Correctness bugs
  - Data races
  - Deadlock
  - and more…
- Performance bottlenecks
  - Overhead
  - Load balance
  - and more…

Intel® Thread Checker finds correctness bugs

Thread Profiler feature pinpoints bottlenecks

Intel® Threading Tools can help!

Intel® Thread Checker Intro

- Identifies threading bugs in applications threaded with:
  - Microsoft* Windows* threads on Microsoft* Windows* systems
  - POSIX* pthreads on Linux* systems
  - OpenMP* on Microsoft* Windows* and Linux* systems
- Plugs into VTune™ environment
  - Microsoft* Windows* for IA-32 systems
  - Linux* for IA-32 and Itanium®-based systems

Intel® Thread Checker Analysis

- Dynamic monitoring as software runs
  - Data (workload) -driven execution
- Includes monitoring of:
  - Thread and Sync APIs used
  - Thread execution order
    - Scheduler impacts results
  - Memory accesses between threads

Only executed code path is analyzed

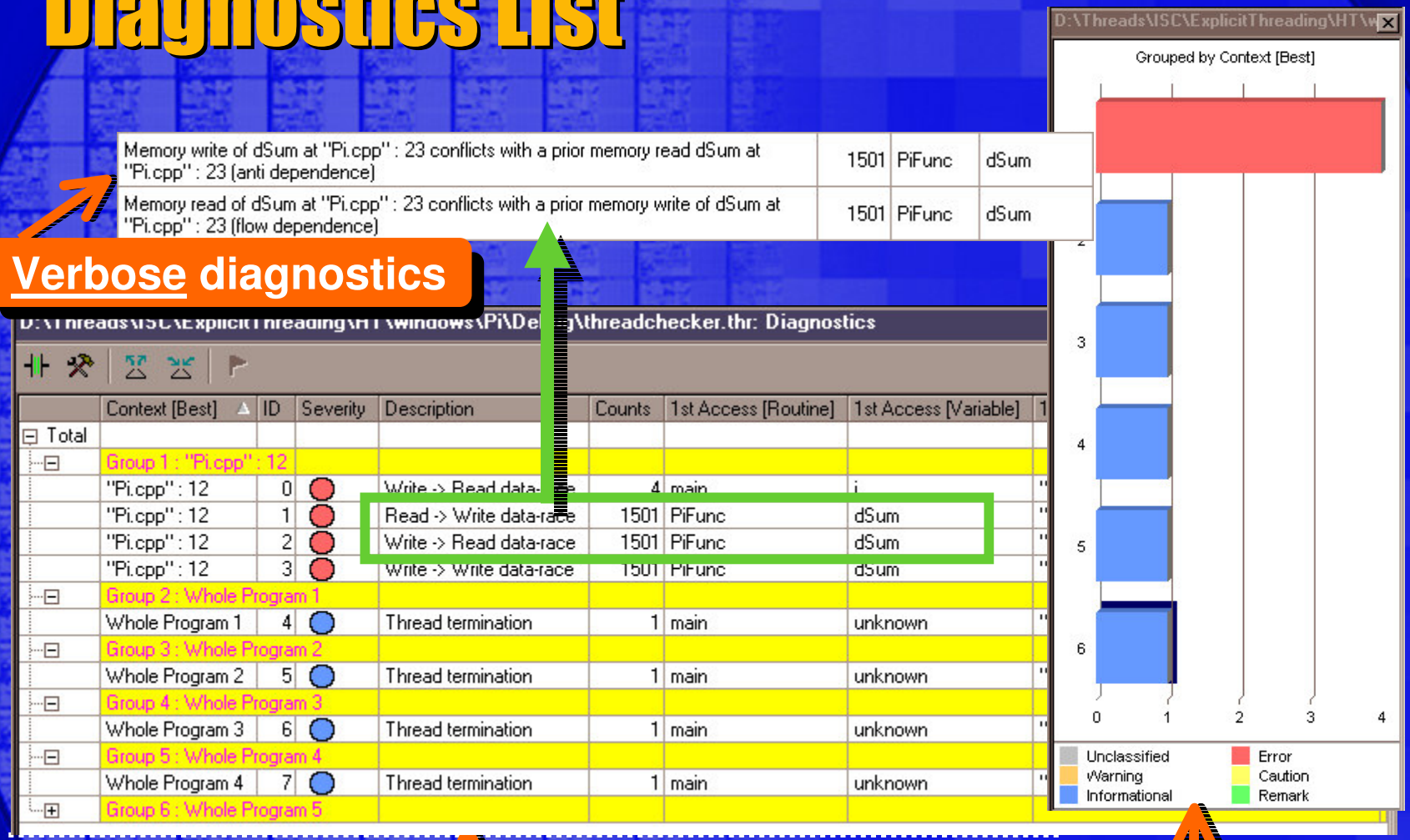
Intel® Thread Checker 2.0

Features

- Locates threading bugs:
  - Data races (storage conflicts)
  - Deadlocks (potential and actual)
  - Win32 threading API usage problems
  - Memory leaks and overwrites
- Isolates bugs to source code line
- Describes possible causes of errors and suggests resolutions
- Categorizes errors by severity level

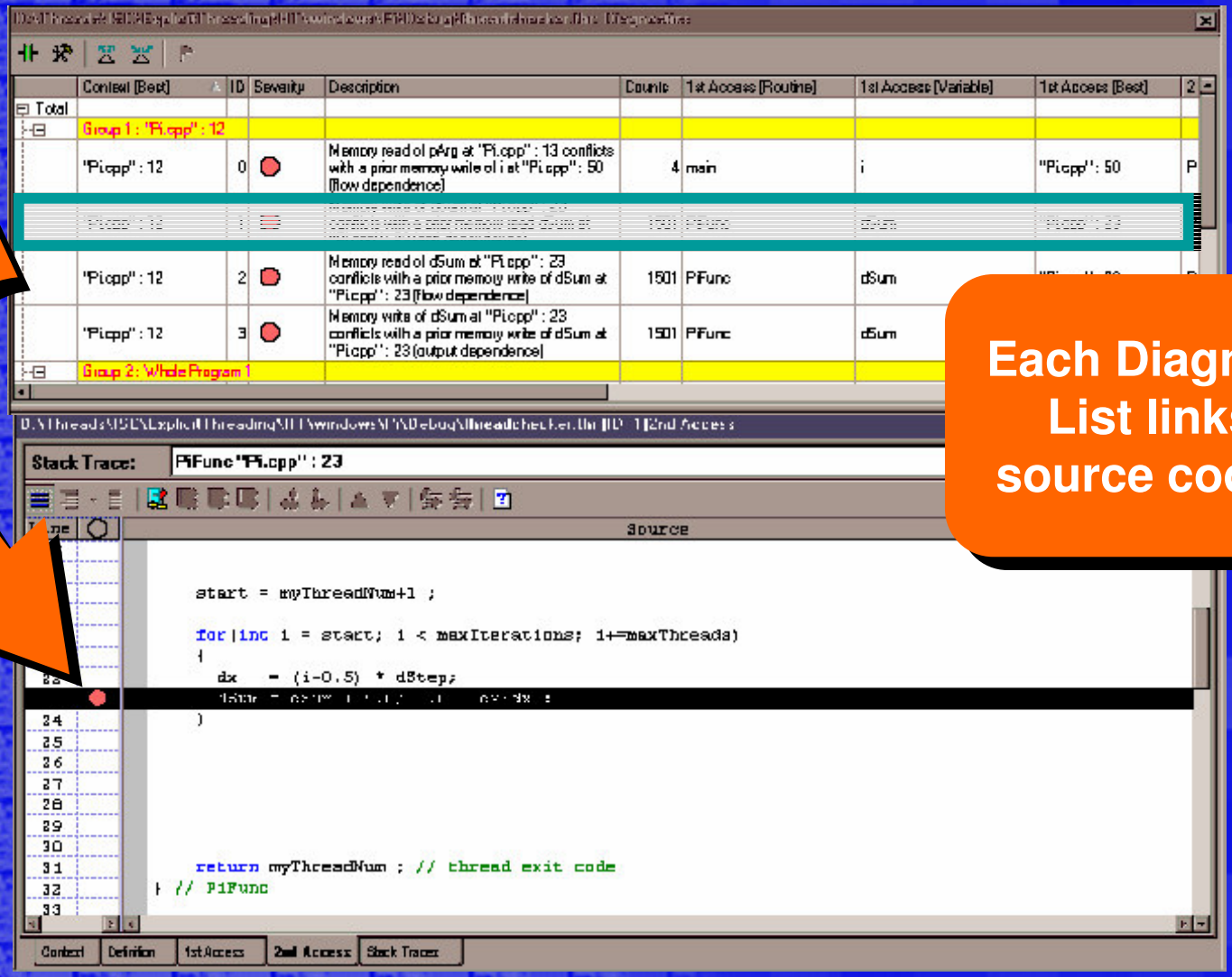
Screen shot: Intel® Thread Checker

Diagnostics List



Screen shot: Intel® Thread Checker

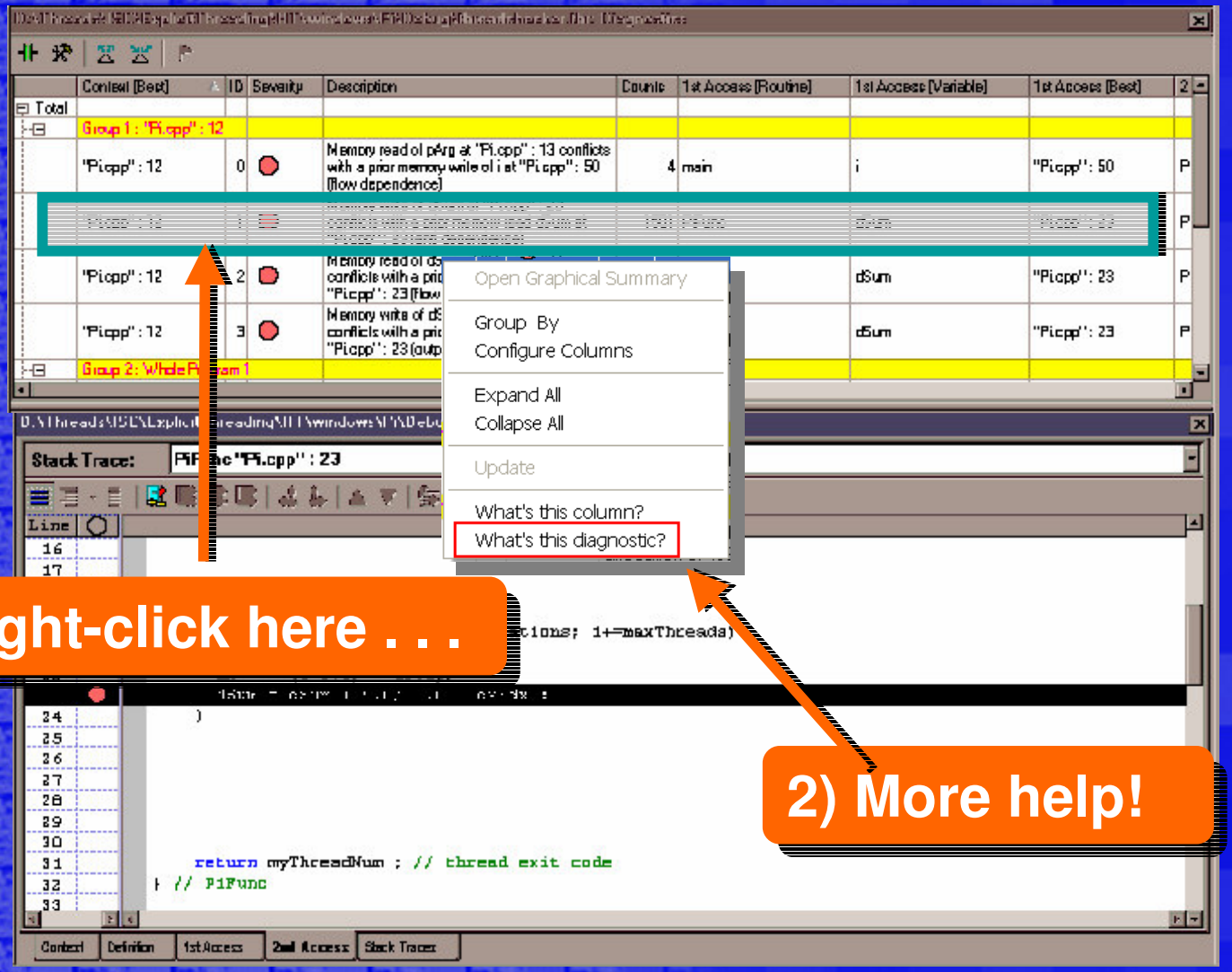
Source Code View



Each Diagnostics in List links to its source code line(s)

Screen shot: Intel® Thread Checker

Help with Diagnostics



Intel® Thread Checker Demo

Find threading bugs faster

The secrets behind the tool

- Automatic application instrumentation
  - Binary instrumentation – VTune™ analyzer technology
  - Source instrumentation – Compiler technology
- Semantic replacement of Threading functions
  - Observe thread management and synchronization
  - Observe system functions which modify arguments
- Partial order of threads drives analysis
- Execution generates annotated address trace
- Produce error diagnostics in a single execution

intel

Skip secretes

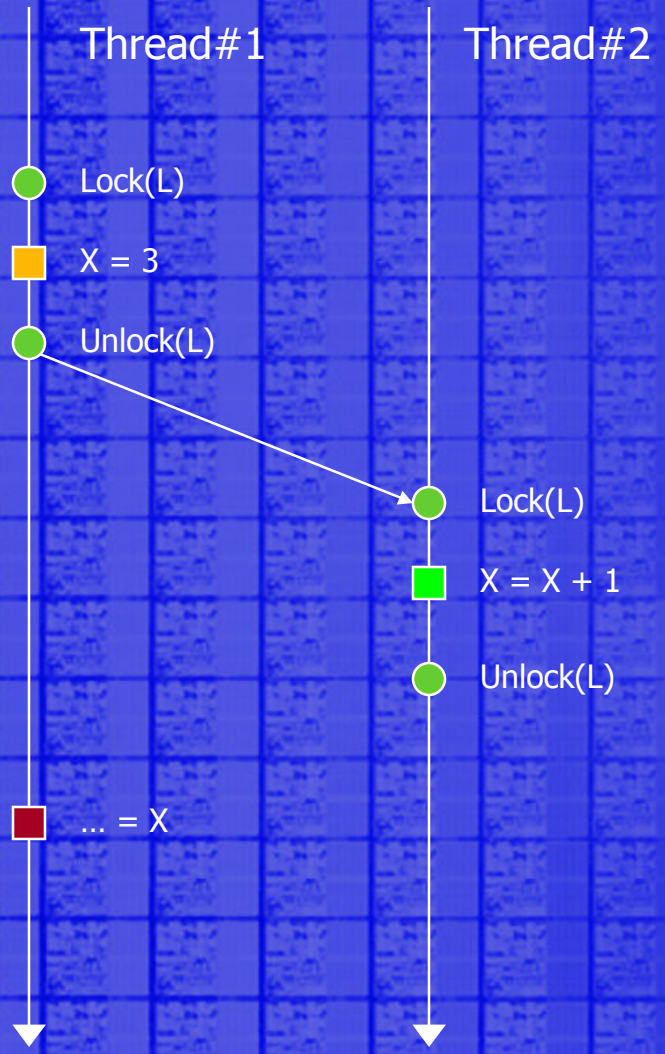
Data Structures

- Threads maintain an integer clock
- Objects (Thread & Sync) have a time vector
  - T#1 = [0, 0]
  - Vector length is the number of threads
  - Hold partial order relationship between objects
- Memory has a shadow cell
  - X = (T#, 0)
  - Record last accesses (thread id and time)
- The Max function is used to merge the vectors
  - Merge([A,B], [C,D])=[Max(A,C), Max(B,D)]

Actions

- ACQUIRE sync action (e.g. LOCK API)
  - The lock is merged with the thread
  - The thread is advanced
- RELEASE sync action (e.g. UNLOCK API)
  - The thread is advanced
  - The thread is merged with the lock
- ACCESS memory action (e.g. READ/ WRITE)
  - Check shadow cell for conflict
  - Updated shadow cell with threads info

Example



Start: T#1 = [1,0] ; T#2 = [0,1] ; L = [0,0]; X = ( )

T#1 = [2,0]

compare X=( ) with T#1=[2,0] : **FIRST ACCESS**
X = (T#1, 2)

T#1 = [3,0]; L = [3,0]

T#2 = [3,2]

compare X=(T#1, 2) with T#2=[3,2] : **OK**
X = (T#2, 2)

T#2 = [3,3]; L = [3,3]

compare X=(T#2 , 2) with T#1 [3,0] : **ERROR**
X = (T#1, 3)

Intel® Threading Tools

"Intel® Thread Checker helped Siemens by identifying issues in software we develop and in software we purchase from third parties. We use Intel Thread Checker to improve the quality of our software and look forward to expanding the use of the tool in more of our software development groups."

-Andreas Dietrich, Research and Development Image Processing, Siemens Medical Solutions

"Using Intel Thread Checker we discovered two elusive bugs on the very first day, as well as numerous inconsistencies and opportunities for performance improvement. We were pleasantly surprised because our product, AcuSolve*, has been running successfully on multiple platforms for many years. We have now incorporated it in our basic development and release process." *-Farzin Shakib, President ACUSIM Software, Inc.*

*Other names and brands may be claimed as the property of others.

The Thread Profiler Feature

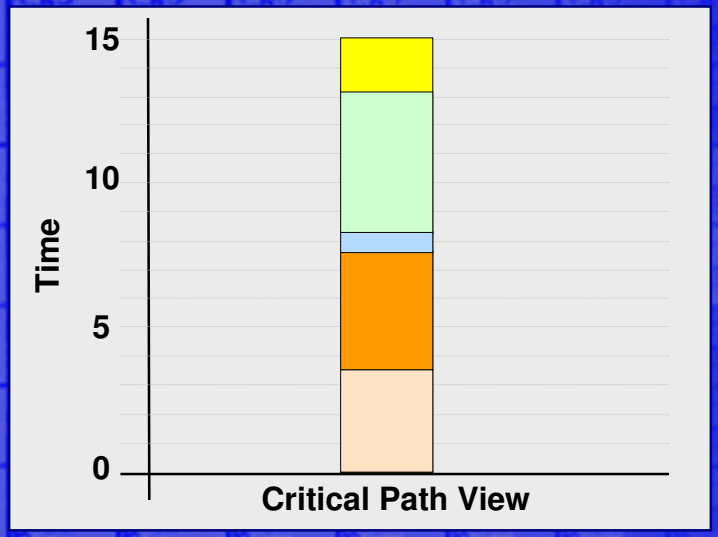
- Pinpoints threading performance bottlenecks in apps threaded with:
  - Microsoft* Windows* threads on Microsoft* Windows* systems
  - POSIX* pthreads on Linux* systems
  - OpenMP* on Microsoft* Windows* and Linux* systems
- Plugs into VTune™ environment
  - Microsoft* Windows* for IA-32 systems
  - Linux* for IA-32 systems

Thread Profiler Feature Analysis

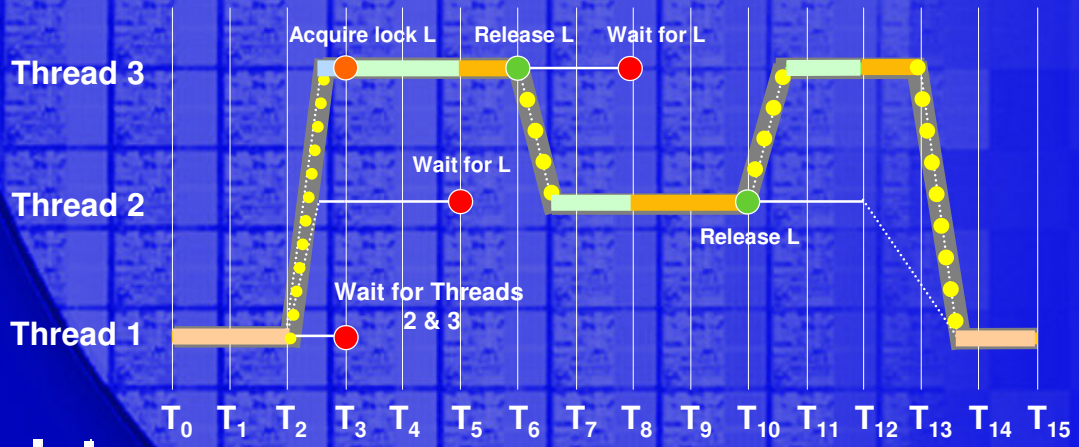
- Monitors execution flows to find Critical Path
  - Longest execution flow is the Critical Path
- Analyzes Critical Path
  - System utilization
    - Over-subscribed vs. under-subscribed
  - Thread state transitions
    - Blocked -> Running
- Captures threads timeline
  - Visualize threading structure

Intel® Threading Tools

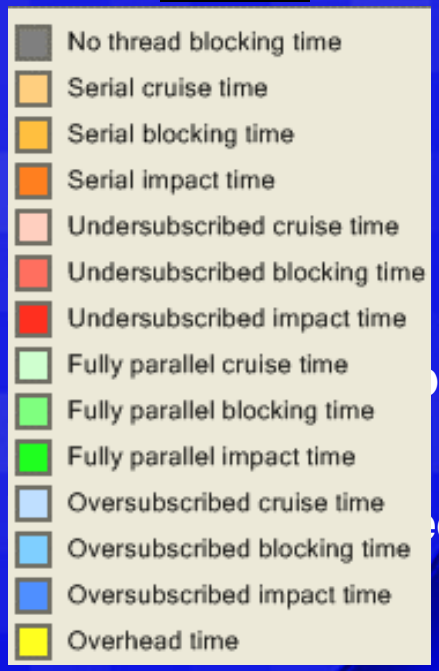
Thread Profiler Critical Path



Analysis shown for 2-way system



- Start with the critical path
- Separate according to system utilization
- Add overhead
- Further analyze by thread state



intel

Exploiting Dual Core Systems: The Intel® Threading Tools

- Add threads to realize full performance benefits of multi cores
- Intel® Threading Tools can help:
  - Intel® Thread Checker finds threading bugs
  - Thread Profiler pinpoints threading bottlenecks

Intel® Threading Tools shortens development cycle for threaded apps

Backup

Collateral

- Intel® Threading Tools

  http://www.intel.com/software/products/
- OpenMP* threads
  - http://www.openmp.org/
- Windows* threads
  - http://msdn.microsoft.com/
- POSIX* threads (pthreads)
  - http://www.ieee.org/

intel

*Other names and brands may be claimed as the property of others.

Additional Reference Materials

- A comprehensive source of information on tools and techniques for developers of software for dual core systems
  - "Programming with Hyper-Threading Technology"
    How to Write Multithreaded Software for Intel® IA-32 Processors
    Richard Gerber and Andrew Binstock, ISBN 0-9717861-4-3
- More info at www.intel.com/intelpress

Purchase Intel Press books:

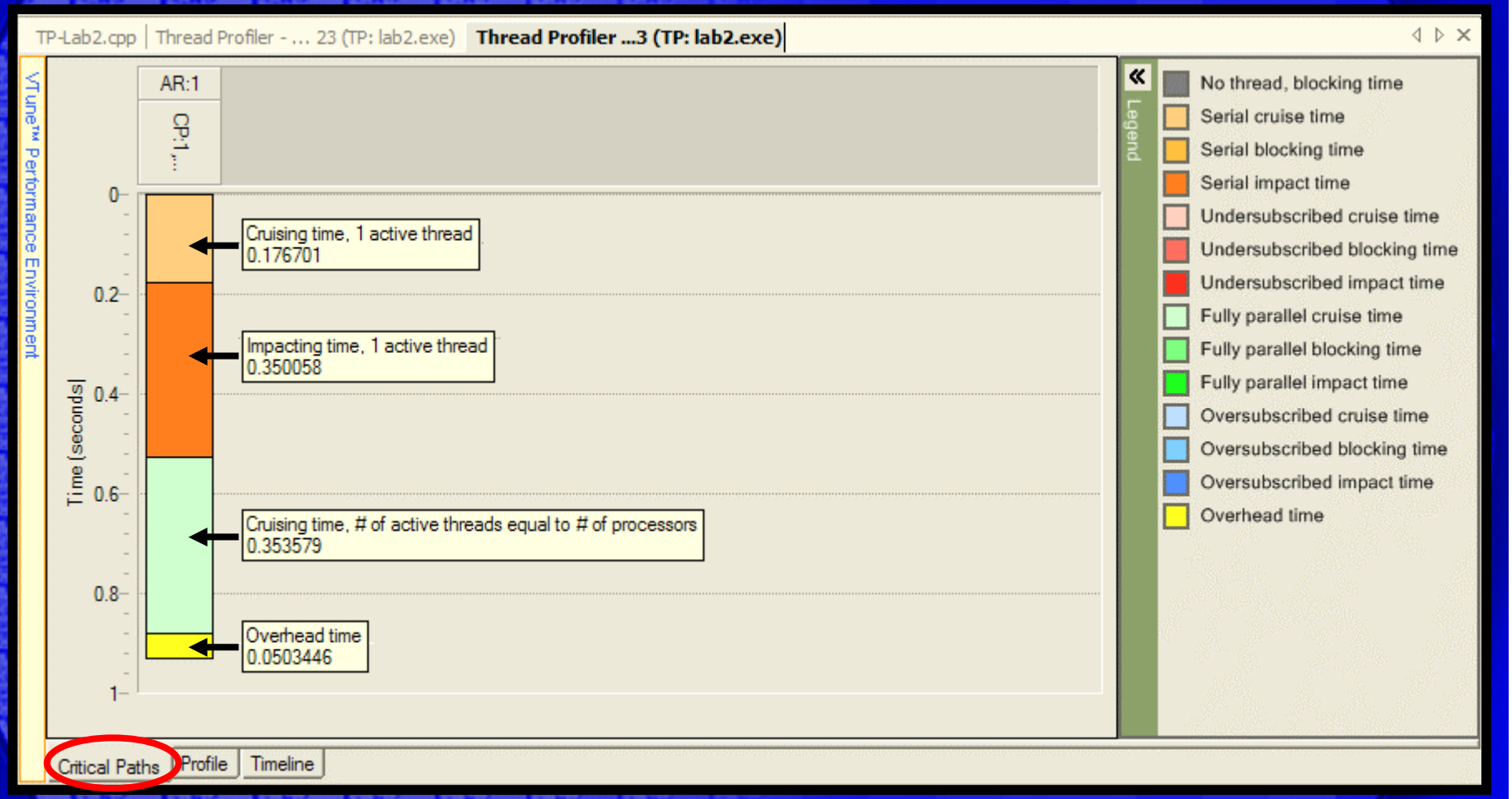
- IDF bookstore
  (20% discount during the conference)
- ShopIntel.com
- Amazon.com

Acronyms

- Pthreads: POSIX* threads

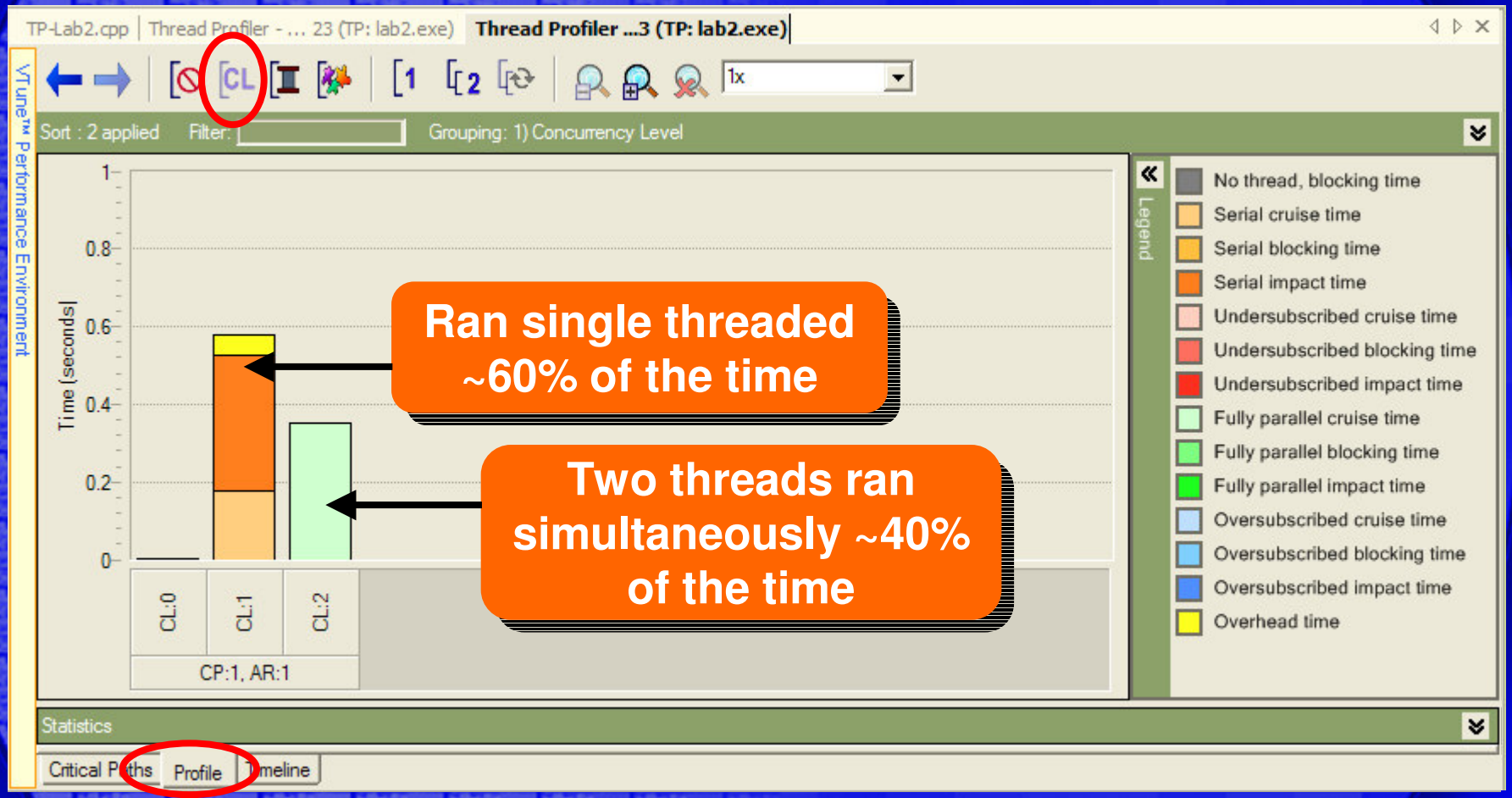
Backup screen shot: The Thread Profiler Feature

Critical Path View



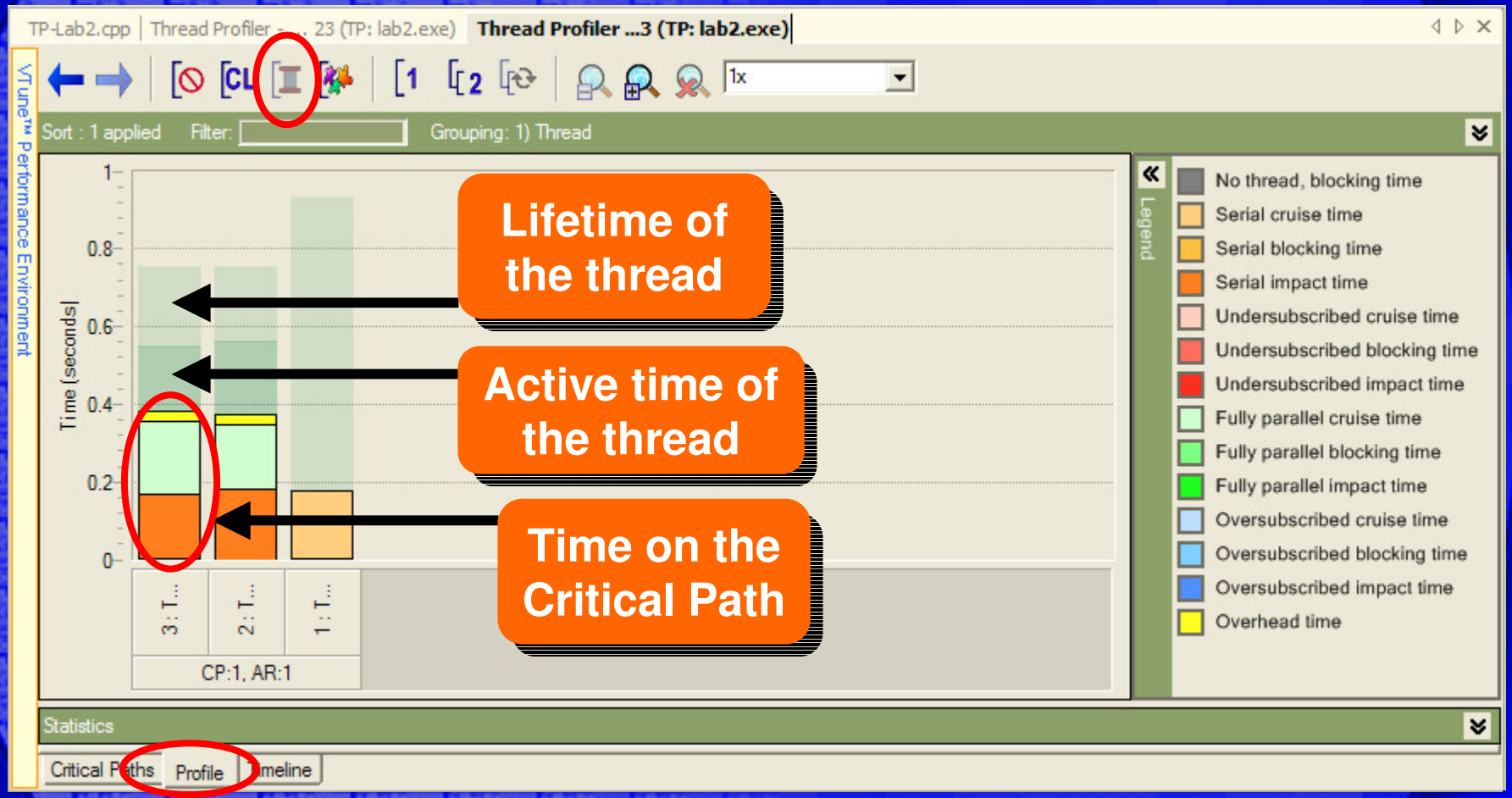
Backup screen shot: The Thread Profiler Feature

Profile Views: Concurrency Levels



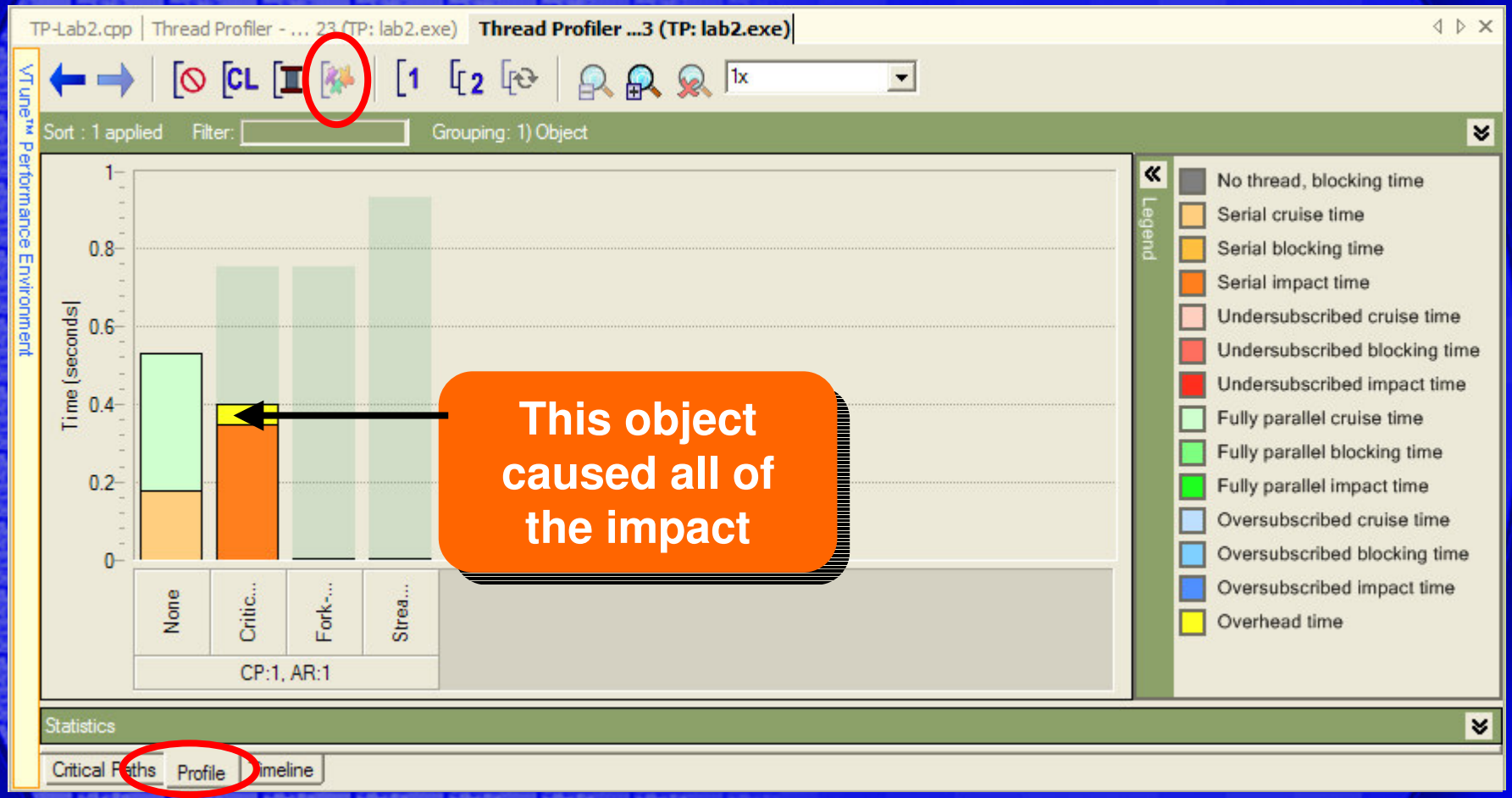
Backup screen shot: The Thread Profiler Feature

Profile Views: Threads View



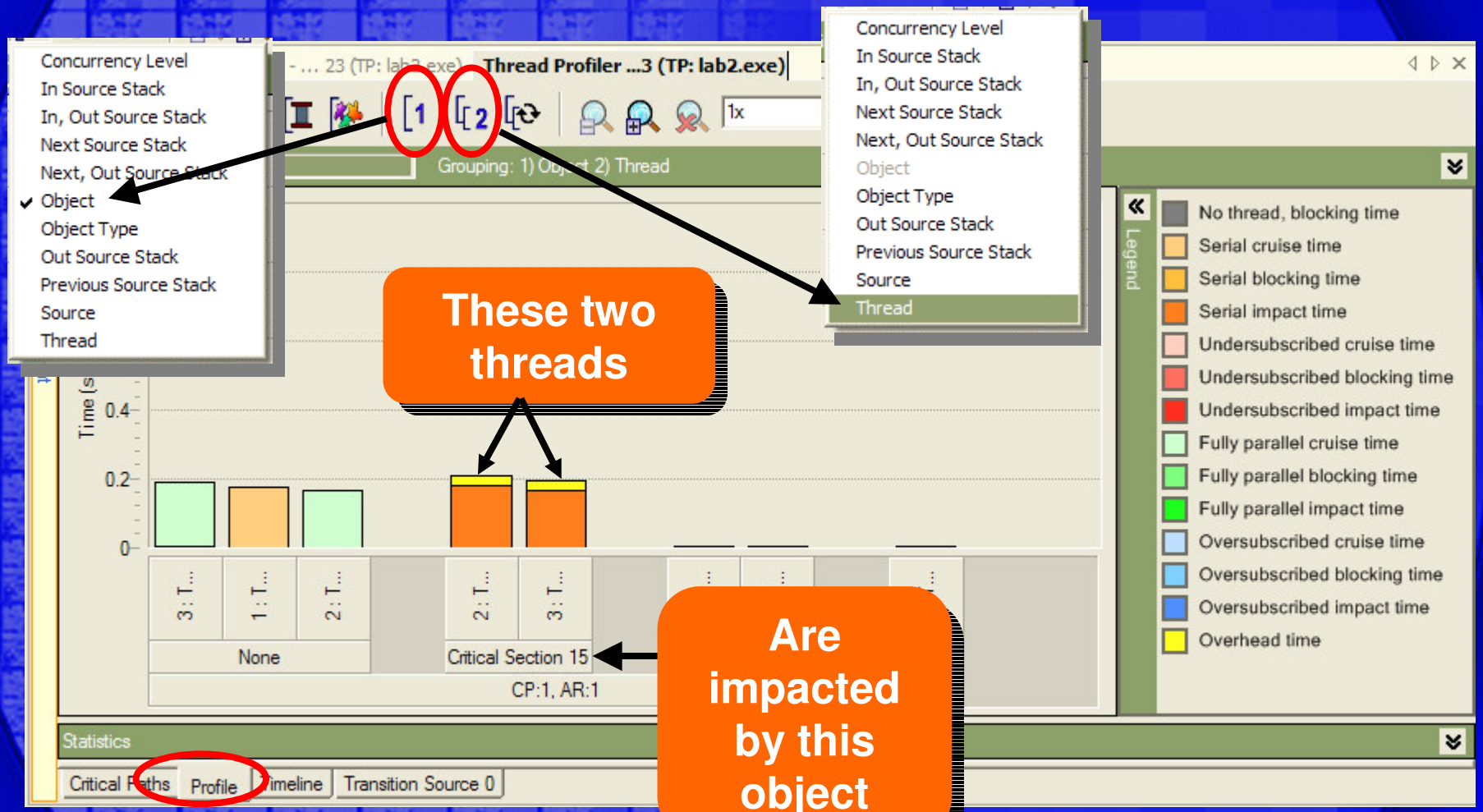
Backup screen shot: The Thread Profiler Feature

Profile Views: Objects View



Backup screen shot: The Thread Profiler Feature

Profile Views: Grouping



Backup screen shot: The Thread Profiler Feature

Timeline View

Performance Penalty: Synchronization

- Thread blocked waiting for Mutex
  - Thread not running, so no parallelism
- Mutex Release, Acquire takes time
  - Release marks mutex free
  - Acquire must check for free
    - If free, mark as in use
    - If not free, thread put to sleep
      - Costs context switch out and in of processor

Create private copies of frequently accessed data to reduce required synchronization