Dr. Dobb's JOURNAL

Developing Faster Applications
An Overview

A *Dr. Dobb's Journal*
Vendor Perspectives NetSeminar
Sponsored by Intel

Tuesday, April 11, 2006
9AM PT / 12PM ET

CMP
United Business Media

Developing Faster Applications
An Overview

Software & Solutions Group

Charles Congdon, Senior Software Engineer

April 11, 2006

Topics

Why optimize

Optimization methodology

Tools and strategies for optimization

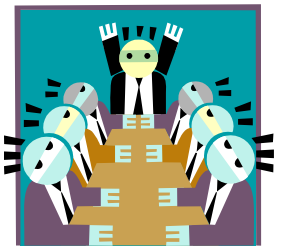
Case studies

Performance tuning 101

Why optimize your application?

Maintaining a competitive edge

Meeting customer demands for performance



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

Goal of Optimization

To improve application performance (runtime) where it matters most

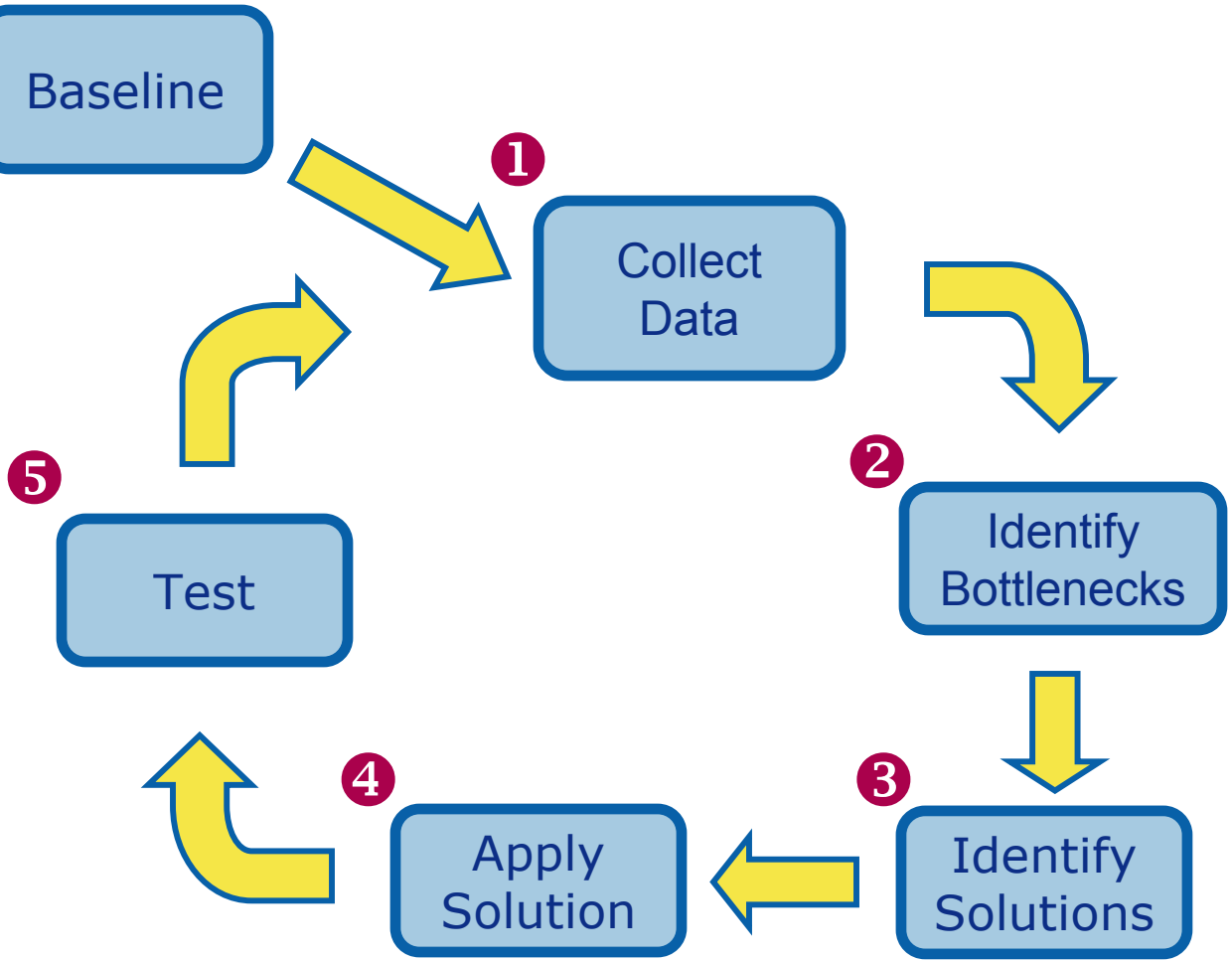
Process:

- Optimize code where the application ***spends the most time***
- Ignore the rest

Facts of Life

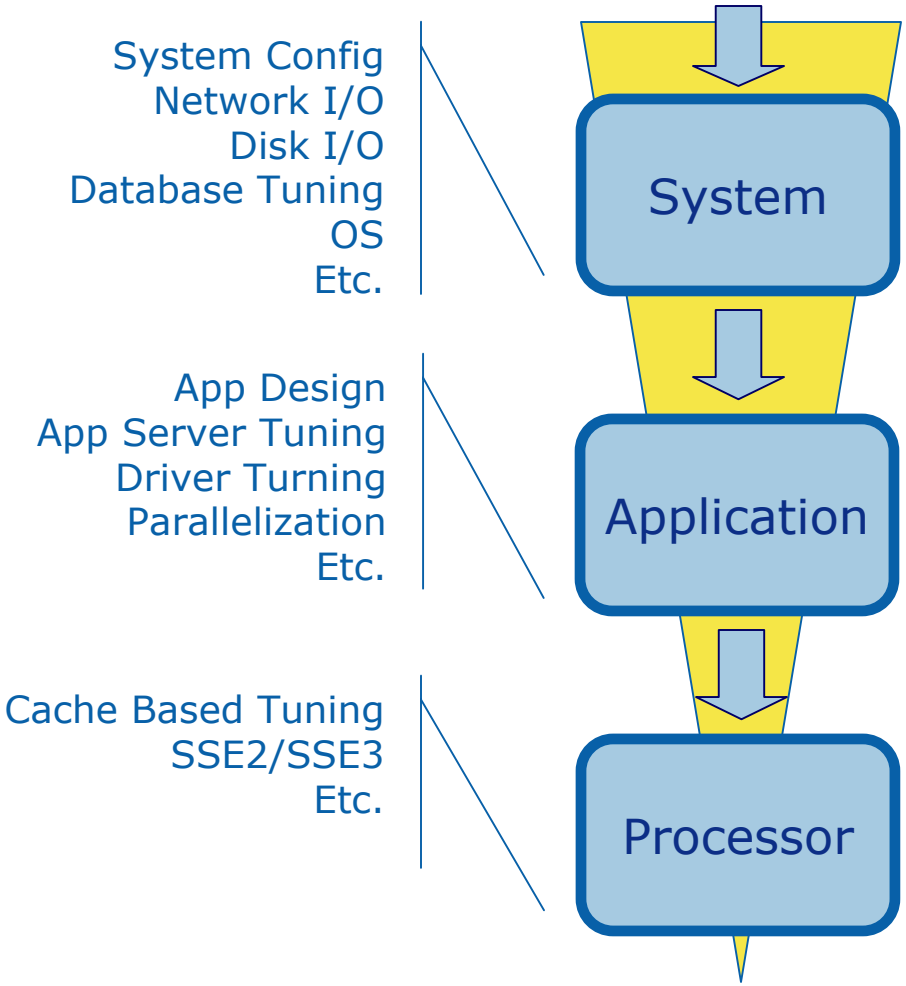
- There is always another bottleneck/hotspot
  - They get harder to find and fix (80/20 rule)
- The bottleneck may not be in code you can fix (e.g. 3rd party components)
  - Creative workarounds, algorithm changes may be needed
- There is no magic solution
  - Every application is different
    - Compiler optimizations will be different for each application
  - Every application ***workload*** can be different
    - **Pick those that are most important to the customer!!!**

Optimization: an iterative process



Question assumptions using repeatable and representative benchmarks

Optimization: a top-down approach



Speeding up applications

Need to know where you are slow

- Are you CPU-bound?
  - Maxing out the CPU?
  - Application scales with CPU speed?
  - Scale with number of processors?
- Are you I/O-bound?
  - If none of the above applies, remember:
    - I/O decreases CPU utilization
  - Possible causes
    - Paging
    - Large disk files (databases / out-of-memory data structures)
    - Network traffic

How do you know?

Are you CPU-bound?

Watch CPU utilization first

- CPU utilization close to 100%?
  - vmstat, sar, mpstat (Linux*)
  - Perfmon*, Task Manager* (Windows*)

If CPU is not maxed-out

- Watch interrupts, I/O, network
  - sar, iostat (Linux*)
  - Perfmon*, Task Manager* (Windows*)

Where is time being spent in *your* application?

- Drill down with Intel® VTune™ Performance Analyzer
  - Sampling (no instrumentation, minimal impact on performance)
  - Call graph (instrumentation, performance impact)
  - Counter monitor (minimal to no performance impact)

Are you CPU-bound?
Speed-up strategies

Algorithm changes

Intel® Performance Libraries

- Highly optimized ready-to-use standard building-block functions; cross-platform
- Easy to try: just link in
- Adapt to the hardware they are run on, support multi-threading
- Intel® Math Kernel Library
  - Standard Linear Algebra subroutines
  - Discrete Fourier Transforms (DFT)
  - Vector Math Library
  - Vector Statistical Library (Random Number Generators)
- Intel® Performance Primitives
  - DSP, Image processing

Are you CPU-bound?
Speed-up strategies cont.

Intel® Compiler

- Provides best performance on Intel® architectures
  - Several higher-level optimization switches beyond -O2
  - Profile Guided Optimization (PGO)
- Multi-threading support
  - OpenMP* pragmas
    - Parallelize pragma-identified loops
    - Provide function-level threading
  - Auto-parallelization compiler switch
    - Detects parallel loops and generates multi-threaded code
- Binary compatibility with Microsoft* and GCC* compilers
  - Object files can be mixed

Are you CPU-bound?
Speed-up strategies cont.

Parallel computing

- Machines with Multiple processors
  - *Multi-process / multi-tasking*
    - Multiple processes work cooperatively
    - More OS overhead due to multiple full-blown processes
    - Requires minimal application rewrite
  - *Threading*
    - May require application rewrite
    - Common development errors
      - Race conditions, Deadlocks
      - Intel® Thread Checker helps speed the creation of error free-code
    - Common performance issues
      - Thread contention around critical resources, load imbalances
      - Intel® Thread Profiler can spot scalability issues
- Cluster and Grid Solutions
  - Distributed computing with message passing
  - Monitoring and analyzing distributed algorithms is a challenge
    - Intel® Cluster Toolkit provides the necessary diagnostic tools
  - Intel® Cluster Toolkit includes Intel® MPI Library and Intel® Cluster MKL Library
    - Simplify creation of distributed applications

Are you I/O-bound?
Speed-up strategies

Algorithm changes

- Reduce I/O, or do it in a different way (ex: separate non-blocking thread)
- Intel® Cluster Toolkit can be used to view communication between threads, processes, and machines

Minimize System Activities

- OS Paging
  - Increase memory
  - 64-bit addressing
- File I/O to disks
  - Faster disks, striped/Raid disk arrays
  - Asynchronous I/O
- Network traffic
  - Multiple network cards
  - Faster network
  - Isolated network

Performance Tuning 101

Optimization Methodology:

- Use top-down approach
- Understand application characteristics
  - Use appropriate tools
- Start with baseline build at default optimization
- Run Intel® VTune™ Analyzer to find bottlenecks
- Optimize *bottlenecks*
  - Improve algorithm / data structures
  - Recognize opportunities to use Intel®-optimized libraries
  - Try different compiler optimizations
  - Explore parallelization opportunities
- Evaluate changes, run Intel® VTune™, iterate …

Case study 1: Black-Scholes* benchmark

Derivative Valuation Benchmark

- Determines the value of an option based on its volatility
  - European option calculation using Black-Scholes formula
- Requires solving partial differential equations using explicit integration of 10 million steps
- High precision calculations required for Normal Distribution Function
  - Uses continuous fractions vs. polynomial
- Uses Monte Carlo techniques to explore problem space
  - Goal is to create an option portfolio with maximum profit and minimal risk

Where is time spent?

- Use Intel® VTune™ Performance Analyzer
  - Answer: Close to 100% time in derivative calculation (single function)

Case study 1: optimization strategies

Add Intel® Compiler optimizations

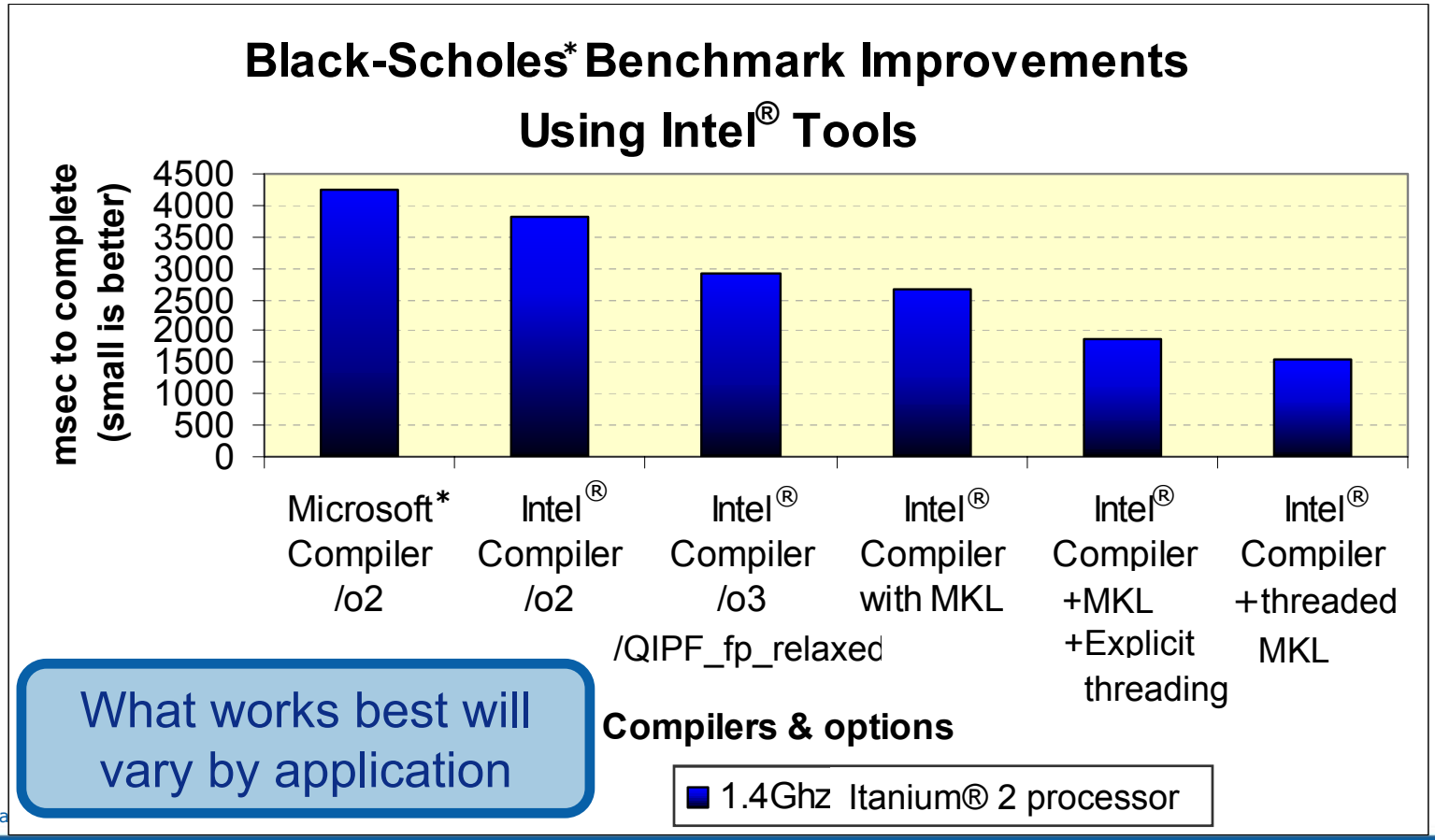
- High-level optimization
- Try high-precision versions of divide and sqrt vs. ultra-speed versions
- Review optimization report to ensure expected optimizations
  - Compilers don't always use a suggested optimization

Use Intel® Math Kernel Library

Thread the application

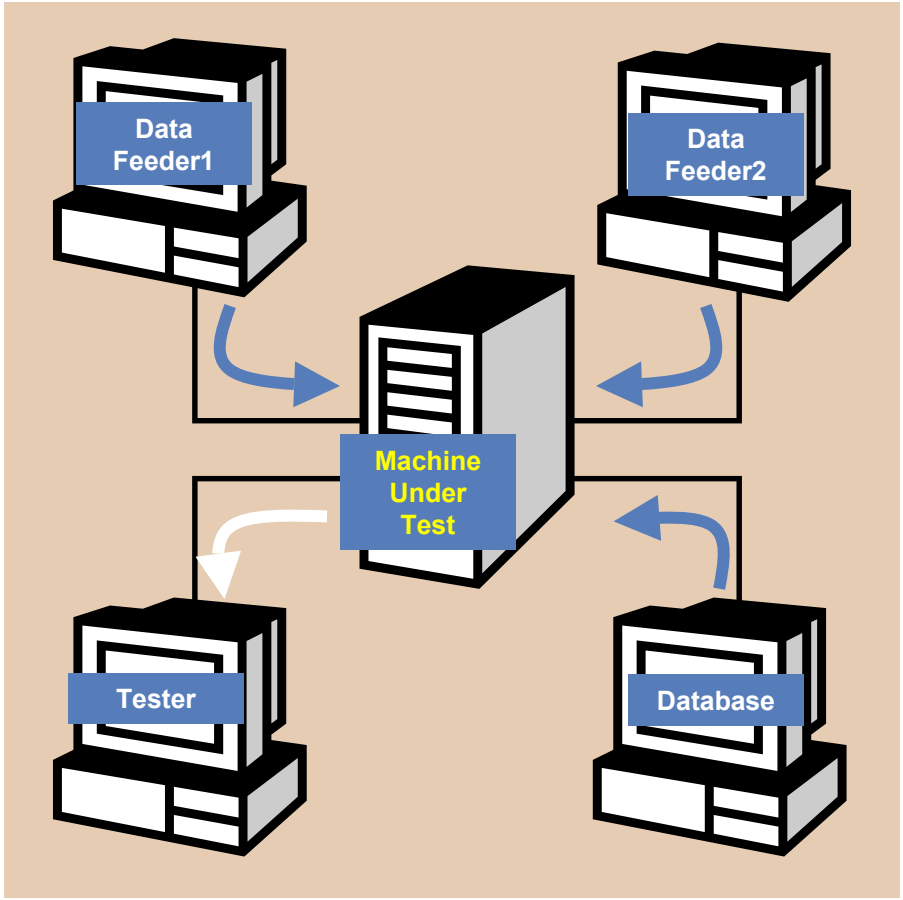
Case study 1: Black-Scholes* performance improvements

Data Source: Intel Corporation



What works best will vary by application

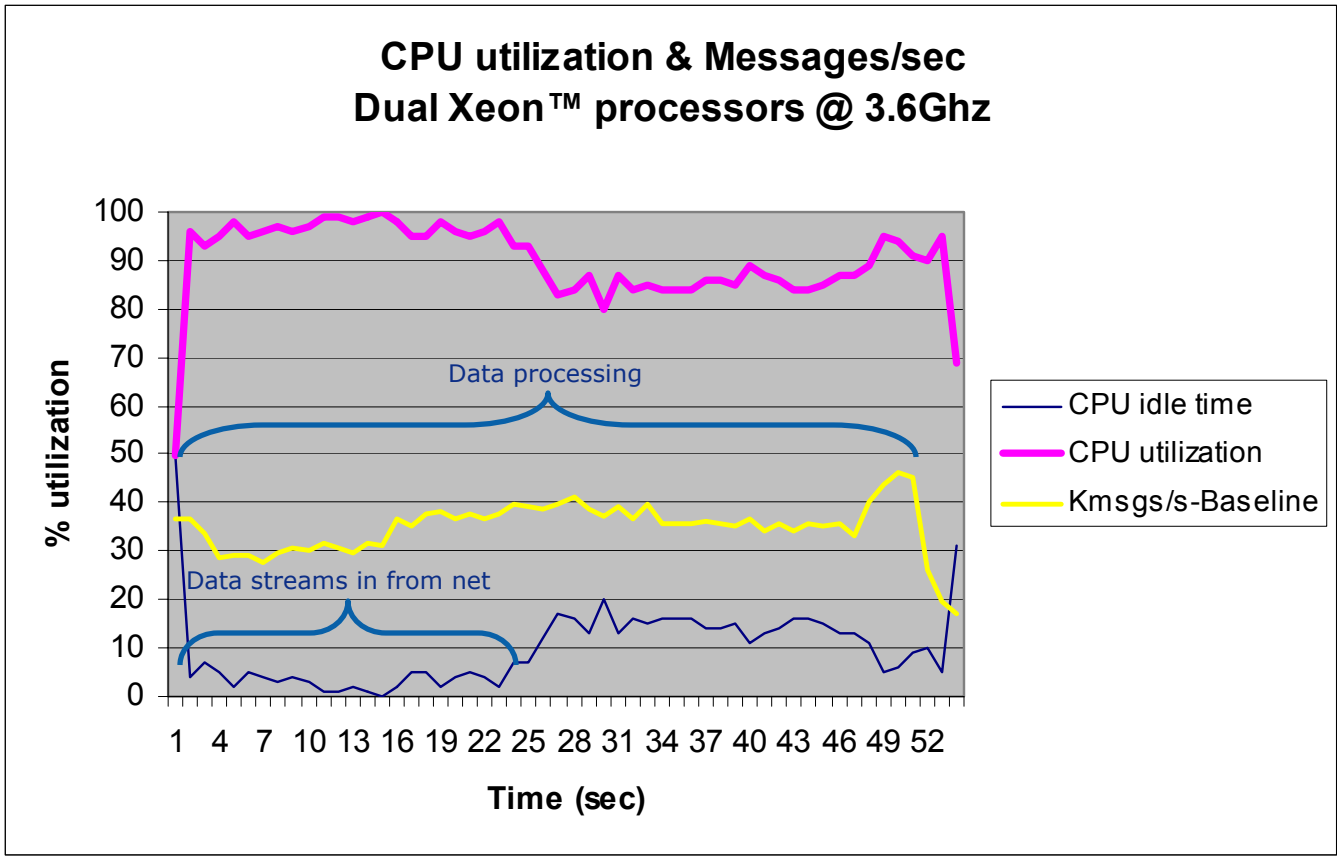
Case study 2: Application server in multi-tier installation



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

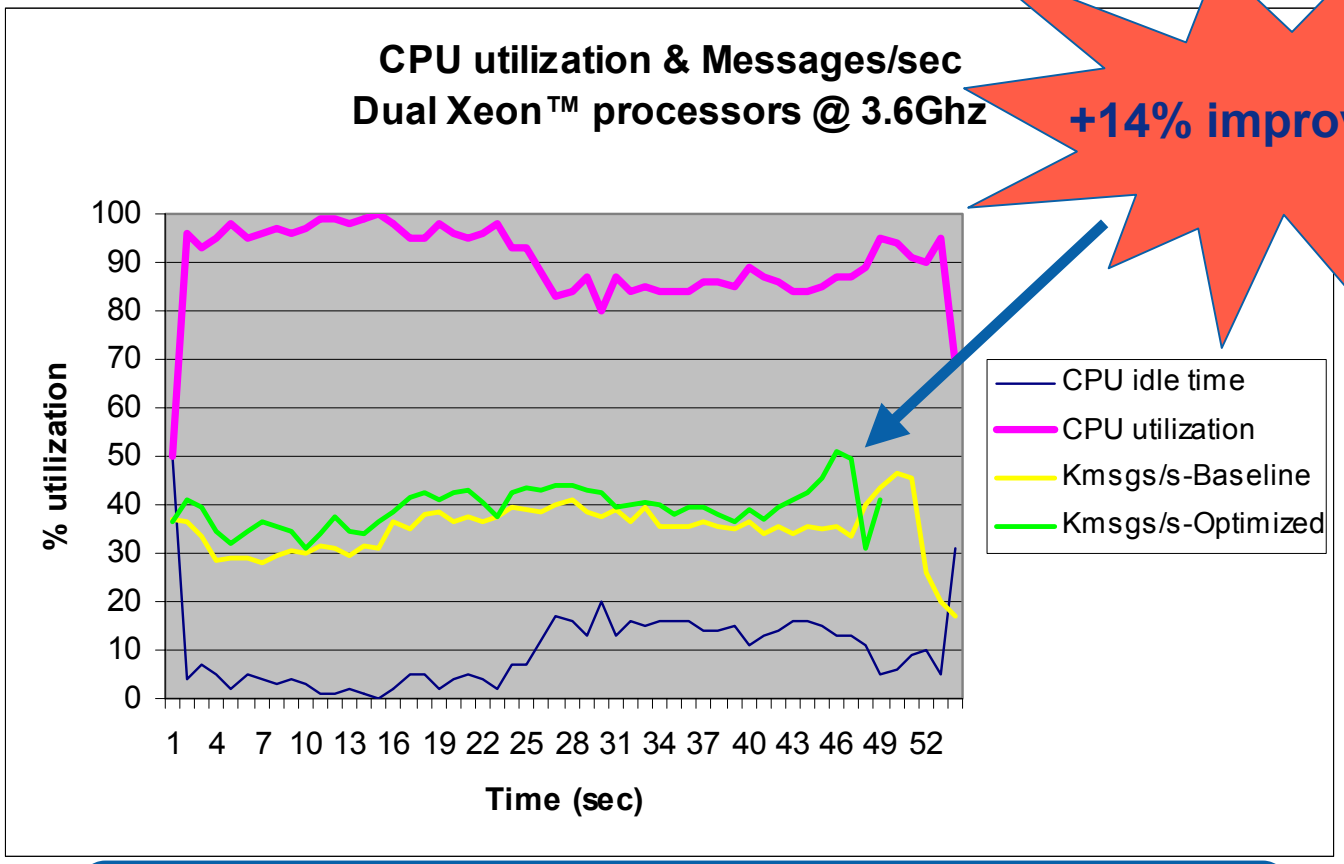
Case study 2: baseline performance and CPU utilization

This graph shows CPU-utilization and message-processing speed as a block of data is received from the network and processed



Data Source: Intel Corporation

Case study 2: optimized



Same amount of data processed in less time

Data Source: Intel Corporation

Case study 2: Intel® VTune™ data (GCC –O3)

System profile

Data Source: Intel Corporation

Process	Clockticks %	Clockticks events
App X	98.13	366,087,600,000
pid_0x0	1.76	6,555,600,000

The application (App_X) is making full use of CPU resources – the CPU is not stalled waiting for I/O

Case study 2: Intel® VTune™ data (GCC –O3)

Data Source: Intel Corporation

System profile

Process	Clockticks %	Clockticks events
App X	98.13	366,087,600,000
pid_0x0	1.76	6,555,600,000

Drill down into App X process

Module	Clockticks %	Clockticks events
libc-2.3.2.so	40.58	148,561,200,000
App X	37.06	135,655,200,000
vmlinux-2.4.21-15.ELsmp	7.21	26,388,000,000
libpthread-0.60.so	5.02	18,392,400,000
libstdc++.so.5.0.3	4.07	14,900,400,000
libgcc_s-3.2.3-20040414.so.1	3.26	11,930,400,000

Significant time spent in runtime libraries

Case study 2: Intel® VTune™ data (GCC –O3)

Drill down into libc-2.3.2.so module

Data Source: Intel Corporation

libc-2.3.2.so functions	Clockticks %	Clockticks events
_int_malloc	20.97	31,150,800,000
__cfree	14.08	20,923,200,000
__libc_malloc	12.33	18,309,600,000
__GI_memcpy	7.47	11,095,200,000
__GI___strtol_internal	4.9	7,286,400,000

Most of the time in the C-runtime library has to do with memory allocation and release

Case study 2: Intel® VTune™ data (GCC –O3)

Drill down into libc-2.3.2.so module

Data Source: Intel Corporation

libc-2.3.2.so functions	Clockticks %	Clockticks events
_int_malloc	20.97	31,150,800,000
__cfree	14.08	20,923,200,000
__libc_malloc	12.33	18,309,600,000
__GI_memcpy	7.47	11,095,200,000
__GI___strtol_internal	4.9	7,286,400,000

Relatively flat profile – no really hot functions

Drill down into App X module

App X functions	Clockticks %	Clockticks events
Function A	12.26	16,527,600,000
Function B	5.43	7,318,800,000
Function C	4.86	6,552,000,000
Function D	3.3	4,449,600,000
Function E	3.3	4,442,400,000
Function F	3.12	4,208,400,000
Function G	2.98	4,014,000,000
Function H	2.84	3,826,800,000

Case study 2: optimization strategies

Optimize “Function A”

Use Intel® C++ Compiler features

- O3 + Pentium® 4 processor switches + Profile Guided Optimization (PGO)

Re-architect memory management?

- Longer term project

Case study 2: Intel® VTune™ data comparisons

Data Source: Intel Corporation

System profile comparison

	Process	Clockticks %	Clockticks events
Before	App X	98.13	366,087,600,000
After	App X	97.96	322,442,800,000

After *Before*

+14%

Fewer events means the application ran for less time

Case study 2: Intel® VTune™ data comparisons

System profile comparison

Data Source: Inte

	Process	Clockticks %	Clockticks events
Before	App X	98.13	366,087,600,000
After	App X	97.96	322,442,800,000

+14%

Spend less time in all functions, particularly Function A

Drill down into App X module

App X functions	*After* Clockticks events	*Before* Clockticks events
Function A	3,294,000,000	16,527,600,000
Function B	6,174,000,000	7,318,800,000
Function C	3,369,600,000	6,552,000,000
Function D		4,449,600,000
Function E	4,003,200,000	4,442,400,000
Function F		4,208,400,000
Function G	2,786,400,000	4,014,000,000
Function H	3,124,800,000	3,826,800,000

Case study 2: Intel® VTune™ data comparisons

Data Source: Intel Corporation

System profile comparison

	Process	Clockticks %	Clockticks events
Before	App X	98.13	366,087,600,000
After	App X	97.96	322,442,800,000

+14%

Some reductions dramatic

Drill down into App X module

App X functions	*After* Clockticks events		*Before* Clockticks events
Function A	3,2	400%	16,527,600,000
Function B	6,1	20%	7,318,800,000
Function C	3,3	90%	6,552,000,000
Function D			4,449,600,000
Function E	4,0	10%	4,442,400,000
Function F			4,208,400,000
Function G	2,7	40%	4,014,000,000
Function H	3,1	20%	3,826,800,000

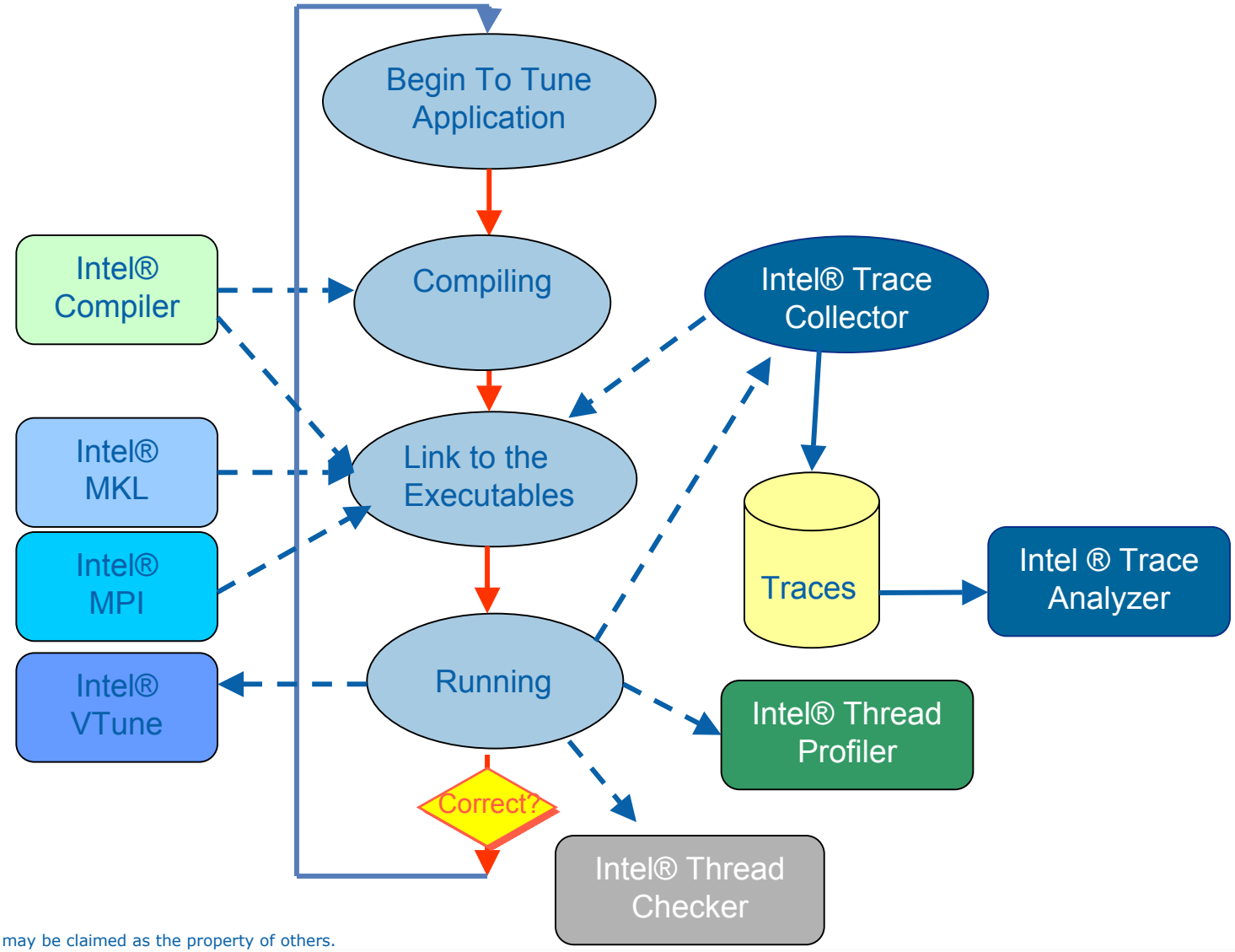
In Summary
Performance Tuning 101

- Use top down-approach
- Understand application characteristics
  - Use appropriate tools
- Use optimization methodology
  - Start with baseline build at default optimization
  - Run Intel® VTune™ Analyzer to find bottlenecks
  - Optimize *bottlenecks*
    - Improve algorithm / data structures
    - Recognize opportunities to use Intel®-optimized libraries
    - Try different compiler optimizations
    - Explore parallelization opportunities
  - Evaluate changes, rerun Intel® VTune™, iterate …
- What works best is different for each application
- Intel is here to help with tools, training, engineering, and hardware resources

Backup

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

Development Tools Flowchart



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

Where Intel Tools can help

Intel® C++ Compiler or Intel® Fortran Compiler

- Typically improve the performance of C++ and Fortran applications on Intel Architectures
  - Support Microsoft Windows*, Linux, Apple OS*, and some embedded operating systems

Intel® VTune™ Performance Analyzer

- Can analyze the performance of almost any 32-bit or 64-bit code on Linux*, Windows*, or Apple OS*
  - Including system code, Java*, or C#*
  - Degree of information returned depends on availability of code symbols or source
  - Wizards mean you don't need to be an expert to get something useful out of the Intel® VTune™ Performance Analyzer
  - Native GUI on Windows* and Linux*, remote collectors for all operating systems

Where Intel Tools Can Help cont.

Intel® Threading Tools

- Intel® Thread Checker
  - For debugging threading code in the early stages of development (threading correctness)
  - Can also be used with existing threaded code with care
- Intel® Thread Profiler
  - For locating performance issues related to threading
- Expert Tools
  - Not intuitively obvious for the novice – some training really helpful
- Support Windows* and Linux*

Where Intel Tools Can Help cont.

Intel® Integrated Performance Primitives

- Provide highly optimized replacement functions for code that does:
  - Audio, image, signal, speech, matrix, string, or video processing
  - JPEG, vector math
  - Speech coding, computer vision, cryptography
- Support Linux* and Windows*

Intel® Math Kernel Library

- Provide highly optimized replacement versions of the following standard packages:
  - Linear algebra: BLAS, LAPACK, PARDISO sparse solver
  - Discrete Fourier Transforms
  - Vector Math Library
  - Vector Statistical Library (random number generators)
- Thread-safe and automatically take advantage of multiple processors
- Support Linux* and Windows*

At runtime, both libraries automatically detect processor and run code optimized for that processor

Where Intel Tools Can Help cont.

Intel® Cluster Toolkit

- Contains:
  - Intel® MPI Library
  - Intel® Cluster Math Kernel Library
  - Intel® Trace Collector
  - Intel® Trace Analyzer
  - Intel® MPI Benchmarks
- MPI enables you to build distributed memory applications capable of execution across multiple network architectures
- Not for applications that require fault tolerance
  - Entire calculation must be redone if participating server drops out mid-way.
- Linux* support only

Some important Compiler Switches

Intel® C++ Compiler

- Use –O2 or better where possible
- Start aggressive optimization with –fast on hottest functions & modules
  - –fast equivalent to "-xP -O3 -ipo –static"
  - Enables SSE3 vectorization (xP), advanced loop transformations (O3), interprocedural optimizations (ipo), and link statically
- Try -xP, -O3, -ipo, and Procedurally Guided Optimization on hotspots
  - Individually and in combination
  - Remove –fast before these experiments
- No one set of options is right for everyone – experiment!

Tip: For Compiler Option Recommendations

- **See www.spec.org/cpu2000**
- **Useful for new platforms – the best numbers come from the best option combinations**

Some important Compiler Switches cont.

GCC 3.x*

- Use -O2 where possible (default is –O0)
- Use –march=noncona (default favors Opteron*)
- gcc 3.4 gives much better performance than gcc 3.2
- Additional help can come from combinations of: -O3 -ffast-math -funroll-all- loops -finline-limit=2000

Microsoft C++ Compiler*

- Use –O2 or better where possible
- /favor:em64t
  - Microsoft Visual Studio 2005* only
  - Default in beta releases favored Opteron*
- /GL (whole program optimization), /fp:fast, and Profile-Guided Optimization can also help
  - Again, target to hot spots when possible

No one set of options is right for everyone – experiment!

Don’t accept compiler defaults!

Further Resources

"The Software Optimization Cookbook, High Performance Recipes for the Intel® Architecture" by Rich Gerber, Intel Press, ISBN 0-9712887-1-2 (2nd edition coming November 2005)

"VTune™ Performance Analyzer Essentials, Measurement and Tuning Techniques for Software Developers" by James Reinders, Intel Press, ISBN 0-9743649-5-9

Intel Software Network

Intel Processors homepage, which links to optimization documentation for each processor

You Don't Believe These Myths of Software Optimization – Do you?

Further Resources cont.

Boosting Performance of Oracle* Database,

- www.intel.com/software/products/global/techtopics/oracle.pdf

Intel® Compilers for Linux*: Compatibility with GNU Compilers

- www.intel.com/software/products/compilers/techtopics/LinuxCompilersCompatibility.htm

Quick-Reference Guide to Optimization with Intel® Compilers

- www.intel.com/software/products/compilers/docs/qr_guide.htm

Optimizing Applications with the Intel® C++ and Fortran Compilers

- www.intel.com/software/products/compilers/techtopics/compiler_optimization.pdf

Dr. Dobb's JOURNAL

Q&A

Please Submit Your Questions Now

CMP
United Business Media

Resources

Intel Performance
www.ddj.com/intel/16.htm

Intel Software Development Products
www.ddj.com/intel/17.htm

Intel Solution Services
www.ddj.com/intel/18.htm

Intel Software Network
www.ddj.com/intel/19.htm

Intel Business Enterprise Resource Center
www.ddj.com/intel/20.htm