



# **Intel Compilers and Software Tools - Vision for Tomorrow**

Pad Iyer  
HPC Software  
June 2010



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.






- Main Compiler Features (current)
- Vectorization
  - SSE
  - AVX
- Parallelism
- Floating-Point
- Math Kernel Library
- Debugging
- What's coming

# Developer Tools for 32/64 bit on Windows\*, Linux\*, and Mac OS\* X



Intel® Software Development Products support multiple platforms

● = Currently Available

Intel® Software Development Products support multiple platforms		    						
		● = Currently Available		Operating Systems		Operating Systems		
				Windows*/Linux*	Windows	Linux	Mac OS*	
Compilers	C++	●	●	●	●			
	Fortran	●	●	●	●			
Performance Analyzers	VTune(tm) Performance Analyzer	●	●	●				
Performance Libraries	Integrated Performance Primitives	●	●	●	●			
	Math Kernel Library	●	●	●	●			
Threading Library	Threading Building Blocks	●	●	●	●			
Threading Analysis Tools	Thread Checker		●	●				
	Thread Profiler		●					
Cluster Tools	MPI Library	●	●	●				
	Trace Analyzer and Collector	●	●	●				
	Math Kernel Library	●	●	●				
	Cluster Toolkit	●	●	●				
All-in-one parallelism toolkit for C/C++	Intel Parallel Studio		●					

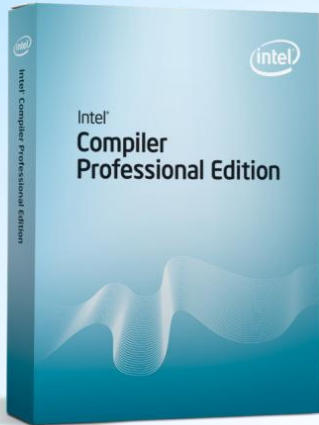


Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# Better performance, multi-core advancements and support for Intel® Core™ i3, i5, i7 and Intel® Xeon® 5500, 5600, 7500 processors

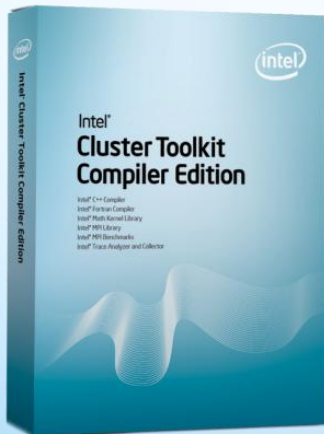


## Intel® Compiler 11.1 Professional Editions

include:

- Intel C++ Compiler and/or Fortran Compiler 11.1
- Intel Math Kernel Library 10.2
- Intel Integrated Performance Primitives 6.1
- Intel Threading Building Blocks 2.2

Windows\*  
Linux\*  
Mac OS X\*



## Intel® Cluster Toolkit Compiler Edition 3.2

includes:

- Intel C++ and Fortran Compilers 11.1
- Intel Math Kernel Library 10.2
- Intel MPI Library 3.2
- Intel Trace Analyzer and Collector 7.2  
with new MPI correctness checker
- Intel MPI Benchmarks 3.2
- Cluster installer

Windows  
Linux

<http://intel.com/software/products>



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# Intel Compilers: Some Generic Features



- Supports standards ( ANSI C, ISO C++, ANSI C99, Fortran95, most Fortran2003, some C++0x )
- C/C++ compatible with leading open-source tools ( ICC vs. GCC, IDB vs GDB, ICL vs CL, ... )
  - Windows\* compiler fully integrates into MS VS 05/08
  - Fortran is not binary compatible with g77 or gfortran
- Supports all instruction set extensions via vectorization
- OpenMP\* 3.0 support and Automatic Parallelization
- Sophisticated optimizations
  - Profile-guided optimization
  - Multi-file inter-procedural optimization
- Detailed optimization reports
- Parallel Debugger Extensions



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

## **icc (or icl or ifort) –O3**

- Loop optimizations:
  - Automatic vectorization<sup>‡</sup> (use of packed SIMD instructions)
  - Loop interchange (for more efficient memory access)
  - Loop unrolling<sup>‡</sup> (more instruction level parallelism)
  - Prefetching (for patterns not recognized by h/w prefetcher)
  - Cache blocking (for more reuse of data in cache)
  - Loop versioning (for loop count; data alignment; runtime dependency tests)
  - Memcpy recognition <sup>‡</sup> (call Intel's fast memcpy, memset)
  - Loop splitting <sup>‡</sup> (facilitate vectorization)
  - Loop fusion (more efficient vectorization)
  - Scalar replacement<sup>‡</sup> (reduce array accesses by scalar temps)
  - Loop rerolling (enable vectorization)
  - Loop peeling (allow for misalignment)
  - Loop reversal (handle dependencies)
  - etc.

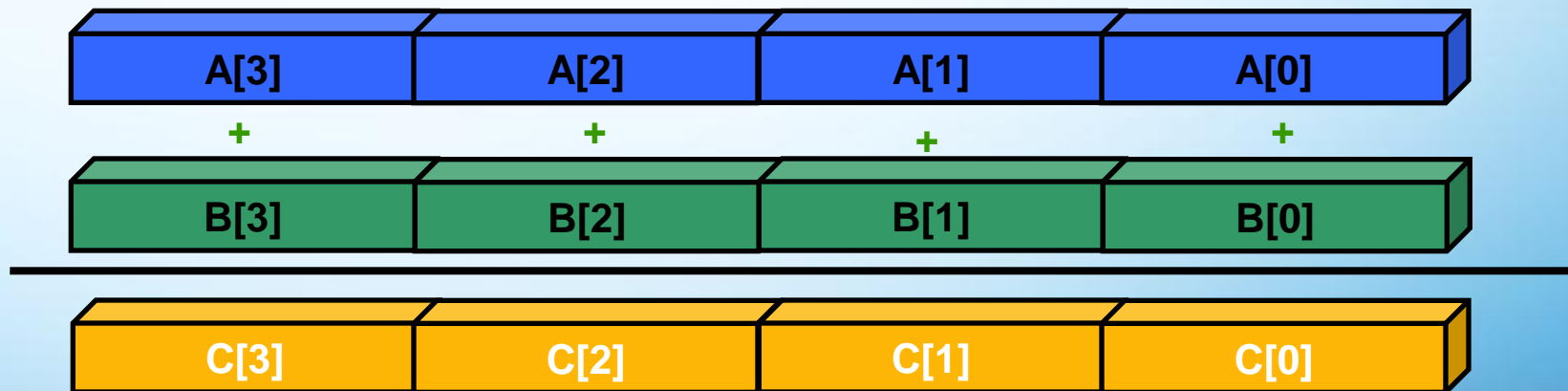
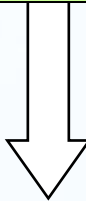
<sup>‡</sup> all or partly enabled at –O2

# Automatic Vectorization by Compiler

## Translate Loops into SIMD Parallelism (SSE)



```
for (i=0;i<=MAX;i++)  
    c[i]=a[i]+b[i];
```



# Requirements for Vectorization



- Must be an inner loop.
- Straight-line code (masked assignments OK)
- Avoid:
  - Function/subroutine calls (unless inlined)
  - Non-mathematical operators (sqrt, sin, exp,... OK)
  - Data-dependent loop exit conditions
    - Iteration count must be known at entry to loop
  - Loop carried data dependencies
    - Reduction loops are OK
  - Non-contiguous data (indirect addressing; non-unit stride)
    - inefficient
- Directives/pragmas can help:
  - `!DIR$ IVDEP .....` ignore potential dependencies
  - `!DIR$ VECTOR ALWAYS` ignore efficiency heuristics
  - `ALIGNED` assume data aligned
  - Compiler can generate runtime alignment and dependency tests for simple loops (but less efficient)
- See <http://software.intel.com/en-us/articles/requirements-for-vectorizable-loops/>



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.



# Problems with Pointers



- Hard for compiler to know whether arrays or pointers might be aliased (point to the same memory location)
  - Aliases may hide dependencies that make vectorization unsafe
- In simple cases, compiler may generate vectorized and unvectorized loop versions, and test for aliasing at runtime
- Otherwise, compiler may need help:
  - -fargument-noalias (/Qalias-args-)
  - “restrict” keyword with -restrict or -std=c99 (/Qrestrict or /Qstd=c99 )
  - #pragma ivdep
  - -fno-alias (/Oa) (high risk – you must be sure!)
  - or by inlining

```
void saxpy (float *x, float *y, float*restrict z, float *a, int n) {  
    int i;  
    #pragma ivdep  
    for (i=0; i<n; i++) z[i] = *a*x[i] + y[i];  
}
```



# Intel® Compilers:

## some useful loop optimization pragmas/directives



- IVDEP ignore vector dependency
- LOOP COUNT advise typical iteration count(s)
- UNROLL suggest loop unroll factor
- DISTRIBUTE POINT advise where to split loop
- PREFETCH hint to prefetch data
- VECTOR vectorization hints
  - Align assume data is aligned
  - Always override efficiency heuristics
  - Nontemporal advise use of streaming stores
- NOVECTOR do not vectorize
- PARALLEL override efficiency heuristics for auto-parallelization



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# Key Intel® Advanced Vector Extensions (Intel® AVX) Features



## KEY FEATURES

- Wider Vectors
  - Increased from 128 bit to 256 bit
- Enhanced Data Rearrangement
  - Use the new 256 bit primitives to broadcast, mask loads and permute data
- Three and four Operands, Non Destructive Syntax
  - Designed for efficiency and future extensibility
- Flexible unaligned memory access support

## BENEFITS

- Up to 2x peak FLOPs (floating point operations per second) output with good power efficiency
- Organize, access and pull only necessary data more quickly and efficiently
- Fewer register copies, better register use for both vector and scalar code
- More opportunities to fuse load and compute operations

**Intel® AVX is a general purpose architecture,  
expected to supplant SSE in all applications used today**



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

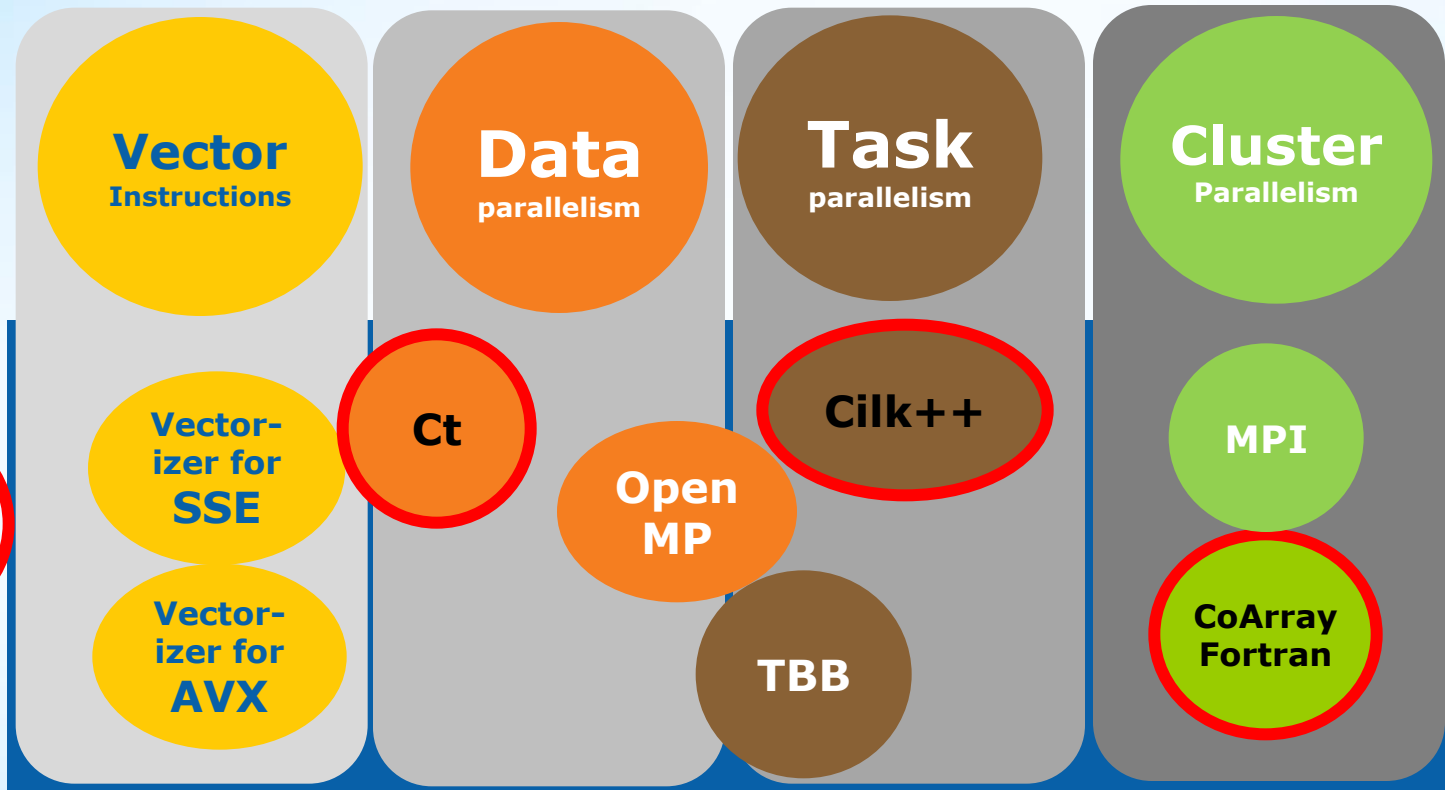
\*Other brands and names are the property of their respective owners.

- Compile with `-xavx`
  - Vectorization works just as for SSE
  - Main speedups are for floating point
    - No integer 256 bit instructions in first generation
    - Up to  $\sim 1.8x$  performance for Linpack
  - More loops can be vectorized than with SSE
    - Individually masked data elements
    - More powerful data rearrangement instructions
- `-xavx` will give both SSE and AVX code paths
  - use `-x` switches to modify the default SSE code path
    - Eg `-xavx -xsse4.2` to target Nehalem and AVX
- Math libraries will target AVX automatically at runtime

# Providing Choice of Parallel Methods



Classification is indicative only, categories overlap...



Resource Management

Multi-core IA Platforms

Many-core IA Platforms



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# Floating Point Semantics



- Floating-point arithmetic involves tradeoffs between accuracy/precision, reproducibility and performance.
- The `-fp-model (/fp:)` switch lets you choose the floating point semantics at a coarse granularity. It lets you specify the compiler rules for:
  - Value safety
  - FP expression evaluation
  - FPU environment access
  - Precise FP exceptions
  - FP contractions
- For a fuller discussion, see <http://software.intel.com/en-us/articles/consistency-of-floating-point-results-using-the-intel-compiler/>



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# The `-fp-model` switch



- `-fp-model`
  - `fast [=1]` allows value-unsafe optimizations (default)
  - `fast=2` allows additional approximations
  - `precise` value-safe optimizations only  
(also source, double, extended)
  - `except` enable floating point exception semantics
  - `strict` precise + except + no assumptions about  
FP environment
- Replaces `-mp`, `-IPF-fltacc`, `-flt-consistency`, etc
- `-fp-model precise` (`-fp-model source`) is recommended for  
ANSI/ IEEE standards compliance (C++ & Fortran)

Further information:

<http://www.intel.com/cd/software/products/asmo-na/eng/279090.htm>

“Floating Point Calculations and the ANSI C, C++ & Fortran Standard”



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# Value Safety - Examples



- In SAFE (precise) mode, the compiler may not make any transformations that could affect the result
- Disabled by `-fp-model precise`:
  - reassociation, eg  $(a+b) + c \rightarrow a + (b+c)$
  - zero folding eg  $X+0 \rightarrow X$ ,  $X*0 \rightarrow 0$
  - multiply by reciprocal eg  $A/B \rightarrow A*(1/B)$
  - approximate sqrt
  - flush-to-zero
  - drop RHS to LHS precision, etc.
- UNSAFE (fast) mode is the default
  - The variations implied by “unsafe” are usually very tiny





- Addition & multiplication are “associative” (& distributive)
  - $a+b+c = (a+b) + c = a + (b+c)$
  - $a*b + a*c = a * (b+c)$
- These transformations are equivalent mathematically
  - but not in finite precision arithmetic
- Reassociation can be disabled in its entirety
  - $\Rightarrow$  standards conformance
  - Use **-fp-model precise**
  - May carry a significant performance penalty (other optimizations also disabled)
  - or “black belt” switch
    - Less performance impact, but unsupported
  - -assume protect\_parens (Fortran only)
    - Respects the order of evaluation specified by parentheses

# Example from CAM (Community Atmosphere Model):



- Issue: Different results with/without optimization
  - Residuals increase by an order of magnitude

- Cause: reassociation of

$$A(I) + B + TOL \rightarrow A(I) + (B + TOL)$$

when  $A(I) = -B \gg TOL$  the result may become zero or very small – impact on final result can be large

- Solution: -fp-model precise      or  
-assume protect-parens with source change  
 $(A(I) + B) + TOL$



# Intel® Math Kernel Library (MKL)

contents:



- **BLAS** (vector & matrix computation routines.)
  - BLAS for sparse vectors/matrices
- **LAPACK** (Linear algebra)
  - Solvers and eigensolvers. Many hundreds of routines total!
  - Cluster implementation (SCALAPACK)
- **DFTs** (General FFTs)
  - Mixed radix, multi-dimensional transforms
  - Cluster implementation
- **Sparse Solvers (PARDISO, DSS and ISS)**
  - OOC version for huge problem sizes
- **Vector Math Library** (vectorized transcendental functions)
- **Vector Statistical Library** (random number generators)
- **Optimization Solvers** (non-linear least squares, ...)
- **PDE solvers**



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

a simple way to thread your application

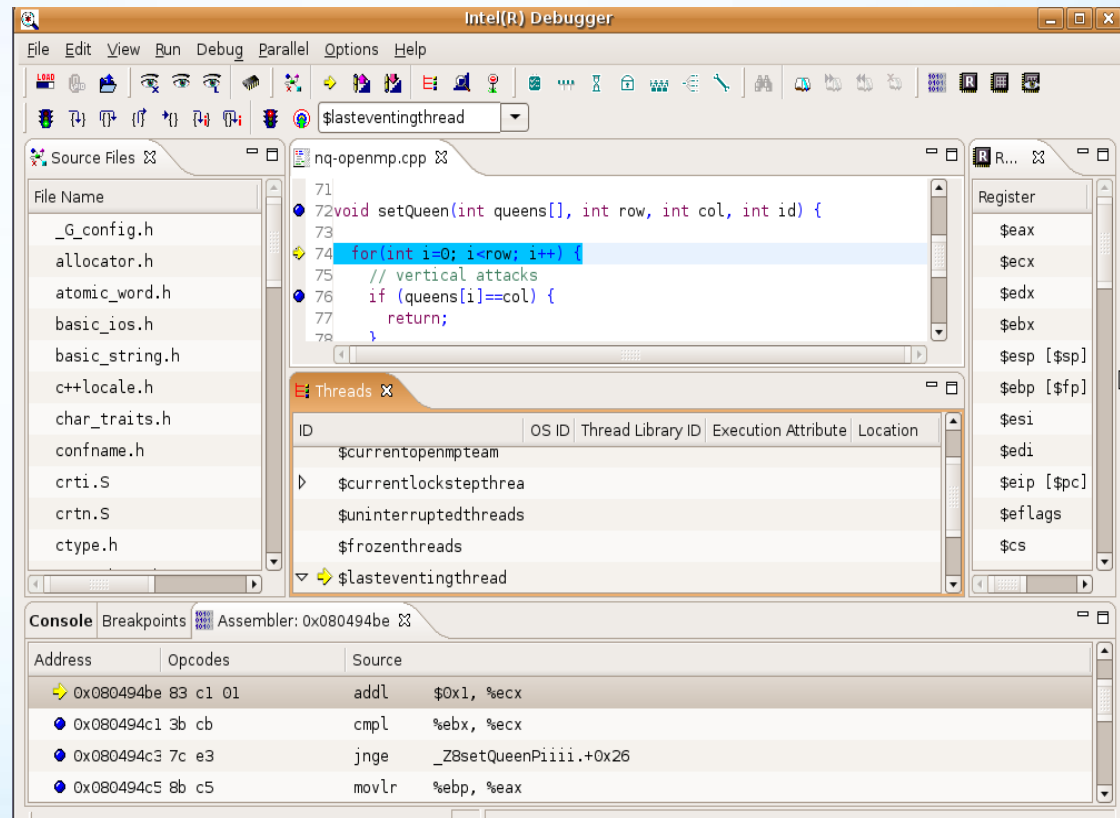
- Many components of MKL have threaded versions
  - Based on the compiler's OpenMP runtime library
- Link threaded or non-threaded interface
  - libmkl\_intel\_thread.a or libmkl\_sequential.a
  - Use the link line advisor at  
<http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/>
- Set the number of threads
  - export MKL\_NUM\_THREADS or OMP\_NUM\_THREADS
  - Call mkl\_set\_num\_threads or omp\_set\_num\_threads
- Optimized for different processor families
  - Loads appropriate version at runtime

# Intel® Debugger (IDB) for Linux\*

GUI and Support for Debugging Parallel Code



- Thread Shared Data Event Detection
  - Break on Thread Shared Data Access (read/write)
- Syncpoint for thread specific debugging
- SIMD SSE Registers Window
- Enhanced OpenMP\* Support
  - Serialize OpenMP threaded application execution on the fly
  - Insight into thread groups, barriers, locks, wait lists etc.



# IDB: Debugging Multi-threaded Applications



## • Thread Control

### • Flexible Thread Execution Models

- Concept of “focus” thread set for next/step/finish
- Freezing and thawing of individual threads
- Can detach threads from debug control

### • Thread Grouping

- Smart default groupings by debugger (All, Team, Lockstep)
- Explicit grouping by user if required

### • Thread Group aware breakpoints

- Break when any member of the trigger group hit
- Can stop a specific thread set only

### • Thread Synchronizing Breakpoint “syncpoint”

- Stop when **all** threads in the trigger group reach it
- Similar to “barriers” in parallel programming

## • OpenMP\* Support

### • Dedicated OpenMP Info Windows

- Threads
- Teams
- Tasks
- Task spawn trees
- Barriers\*
- Taskwaits\*
- Locks
- Serial execution of a parallel region at debug time\*

### • Technology

- Uses special OpenMP RTL



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

Intel Confidential

# What is coming ... (in beta testing)



- More support for parallelism and vectorization
  - **Cilk++**
  - **Ct**
  - Array notation for C
  - CoArray Fortran
  - Guided auto-parallelization (advice)
  - “vectorize or fail” #pragma simd
  - Much extended support for **AVX**
- Language features from new standards
- New Correctness tools
  - Memory checking
  - Static security analysis
  - Thread checking, detection of race conditions
- Easy to use profiling tools
- Opportunities available to beta test new features



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# Compiler 12.0 - Cilk++ integration



Feature	Example	Semantics
Spawning a function call	<code>x = _Cilk_spawn func(y);</code>	func executes asynchronously
Synchronization statement	<code>_Cilk_sync;</code>	Wait for all children spawned inside the current function
Parallel for loop	<pre>parallel _Cilk_for (int i=0; i&lt;N; i++) { statements; }</pre>	Loop iterations execute in parallel.
Hyperobjects		Allow parallelization of reduction operation with minimal source changes

That is most of the language.  
Semantics are quite simple and powerful



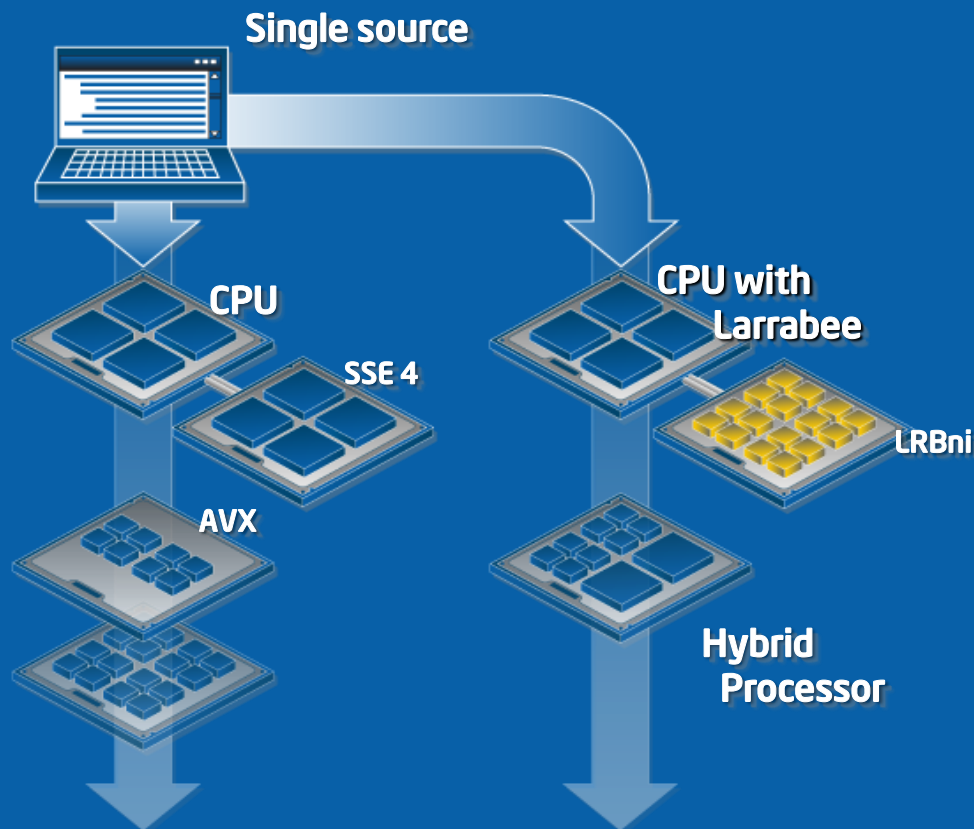
Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.



# Ct Technology: High Productivity on Manycore



## Productivity

- Eases the path to data parallelism
- Safety by default → fewer bugs, more maintainable → lower ownership cost
- Use existing tools and compilers
- Interoperates with other approaches to parallelism with Intel and third-party tools
- Widely applicable

## Performance

- Scalable and efficient (SIMD + threads)
- One specification → many mechanisms
- Eliminates modularity overhead of C++

## Portability

- High-level abstraction vs. uarch-specific
- Future proofing for more threads and new ISAs via a dynamic compiler



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# Intel's Ct Technology

*Data Parallel Programming for Multi-core and Many-core Processors*



- **Productive Data Parallel Programming**
  - High-level abstraction, natural notation
  - Ease of programming, portability
    - Abstracts away architectural details
  - Good (but not ultimate) application performance
    - Low-level coding may still run faster
  - Built-in thread safety (no data races)
- **Investment protection via Forward-scaling**
  - Performance across present and future processor generations
- **Extends standard C++ using templates**
  - Adds parallel containers & methods to C++
  - Works with standard C++ compilers
    - Intel® C++ Compiler, GCC\*, Microsoft Visual C++\*
  - Interoperates with other Intel tools



**Software & Services Group, Developer Products Division**

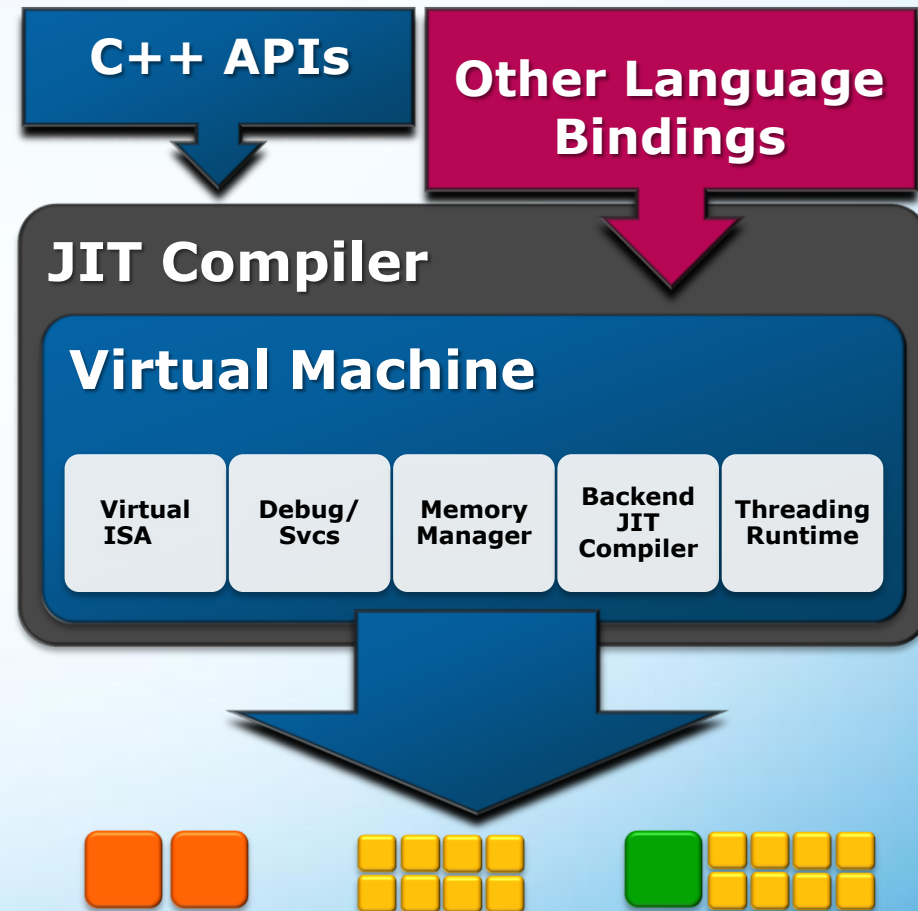
Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# The Runtime



- Intel's Ct Technology offers a standards compliant C++ library...  
...backed by a runtime
- The runtime generates and manages threads and vector code, via
  - Machine independent optimization
  - Offload management
  - Machine specific code generation and optimizations
  - Scalable threading runtime based on Intel® Threading Building Blocks (Intel® TBB)



Read more at  
[http://www.intel.com/software/data\\_parallel/](http://www.intel.com/software/data_parallel/)



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.



- More C++0x and C99 features
  - E.g. rvalue references
    - reduce temporary copies
  - Maintain Microsoft Visual Studio\* compatibility
- Optimized string intrinsics
  - Using SSE4 instructions
- New array syntax for C/C++
  - More readable; Helps the compiler vectorize and parallelize
  - Somewhat similar to Fortran90 concept
    - `<array base> [ <lower bound> : <length> [ : <stride> ] ]`
    - E.g. `a[0:s] += b[2:s:2]`
- Incorporation of Cilk technology
  - Simple keywords, especially for task parallelism
  - “Hyperobjects” to facilitate thread-safe access
    - Provide thread safe reduction operations

# CEAN: C++ Extended Array Notations for data parallelism



- This new array extension to C/C++ language allows users to specify data parallelism on array objects, similar to F90 array expressions. By directly specifying parallel operations on arrays (instead of sequential loops), users experience “what they write is what they get” in terms of vectorization and parallelization.
  - Language must let the developer get to hardware performance easily
  - Data parallel array notations provide a natural mapping between data parallel algorithms and underlying parallel hardware (multi-core CPU/GPU) to achieve high performance and utilization
  - C/C++ is the dominant language for sequential performance applications because its constructs map easily to single core ISA
  - Data parallel extensions to C/C++ try to accomplish the same performance goal for parallel application and hardware:
    - Expressing data parallelism that is syntactically and semantically consistent with the familiar language
    - Predictable performance based on mapping parallel constructs to underlying multi-threading and SIMD hardware in a native environment
    - Work seamlessly with existing C/C++ frameworks and runtimes: TBB, OpenMP, MPI, Clik, Pthread





- Support for CoArray Fortran
  - Shared memory
  - Distributed memory (available with cluster tool suite)
  - Uses Intel MPI technology
- Other Fortran 2008 features
  - Submodules
  - DO CONCURRENT
  - CONTIGUOUS
- More Fortran 2003 features
  - Complete type-bound procedures (GENERIC, OPERATOR,...)
  - FINALization

# Intel® Math Kernel Library 10.3



Benefit from hand-optimized and threaded math and signal processing functions

Threading implemented with OpenMP

- e.g. control number of threads by OMP\_NUM\_THREADS

## Features

- Linear algebra , FFT, Vector math, statistics
- Automatic runtime processor detection
- New threading layer
  - ensures compatibility with Microsoft\*, GCC and Intel Compiler threading models
- Extended AVX/Sandybridge support
  - many optimized kernels
- More C/C++ support
  - C interface to LAPACK with row major matrices
- Dynamic accuracy control for VML
- Summary Statistics Library
  - threaded analysis of multi-dimensional data sets
    - quantiles, moments, correlations)
- C#/.NET support
  - Dynamic library interface to improve linkage from C#



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

Intel Confidential



# Intel's Cantua HE Checker

## Ensures secure, reliable, high quality code



- **Built-in Static and Dynamic Analysis tools for Parallel and Serial Code**
  - Memory checking that detects memory leaks and corruption issues
  - Thread checking finds threading errors
  - Source checking that detects memory and security vulnerabilities
  - Data race and deadlocks detection
- **Thread safety validation** – verifies threads will operate correctly
- **Scalability** – Analysis tool performance scales with available number of cores
- **Enhanced Developer Productivity**
  - Intuitive GUI and CLI interface to access a wide range of functions
  - Displays detailed reports and comprehensive error data analysis
- **Supported parallel models:** Intel® TBB, OpenMP, Intel® Cilk++, Ct, and Microsoft PPL/ConcRT
- **Supported environments:**
  - C/C++, Fortran and .Net
  - Microsoft Visual Studio 2005, 2008, 2010
  - Windows and Linux operating environments



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.





- Simple Setup, Integrated Results
  - Windows and Linux native versions
  - Stand-alone GUI, Command line, Visual Studio Integration
  - No special re-compiles required
    - Works with Microsoft, GCC, Intel and other compilers that follow standards
    - Include symbols for best results
  - Simple time-based sampling
  - Event-based sampling
    - Cache misses, branch mispredicts, etc
  - Timeline display
  - Memory analysis
  - Fast statistical call graph
  - Thread profiler



# Intel Cluster Tools 4.0

What's new?



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

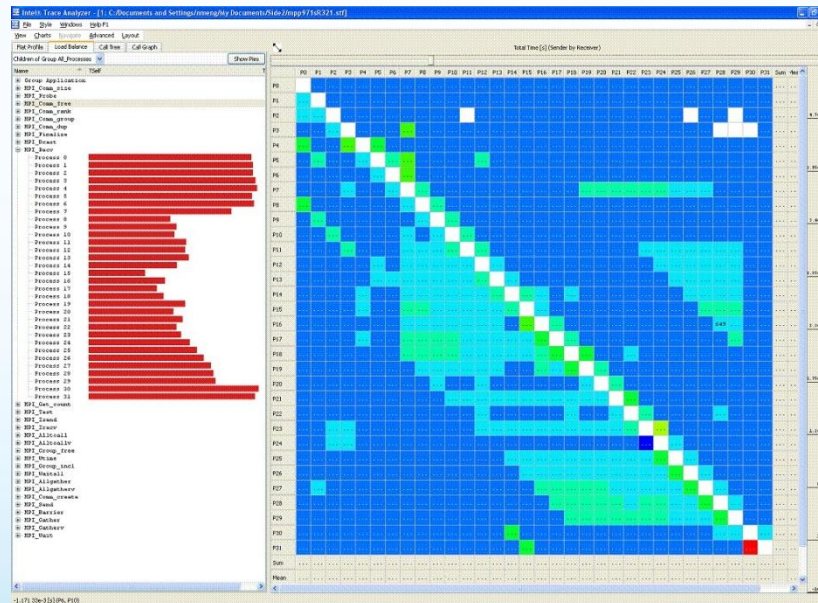
\*Other brands and names are the property of their respective owners.

# Intel® Trace Analyzer and Collector 8.0

## What's new



- Application Imbalance diagram for simplified application analysis
- Ideal Interconnect Simulator (IIS) to understand application balance
- Intel® Custom Plug-in Framework (ICPF) to simulate application behavior over different interconnects



Novel tools for application analysis

# Intel® Cluster Toolkit Compiler Edition & Cluster Ready Program

**Most comprehensive hardware and software solution for high performance computing**



Design

**Intel® Cluster Ready Program**  
(verify with Intel® Cluster Checker)

Implementation

**Intel® C++ and Fortran Compilers,  
Performance Libraries**

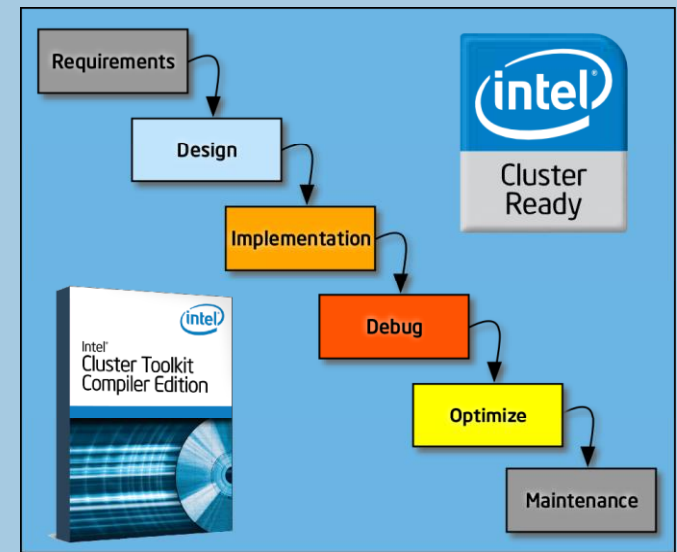
Debug

**Intel® Debugger, MPI Benchmarks**

Optimize

**Intel® Trace Collector and Trace Analyzer**

**Click Here to  
Learn More**



**Improved Cluster Performance with  
Intel® Cluster Toolkit 3.1 Compiler Edition**

Source Code

**Intel® Cluster Toolkit 3.1 Compiler Edition**

Intel® C++ & Fortran Compiler with Cluster Open MP for Intel® Compilers	Intel® MPI Library	Intel® Math Kernel Library	Intel® Trace Analyzer and Trace Collector
<b>Optimization for the Node</b> OpenMP*, Auto-Parallelization, Vectorization, PGO, IPO & HPO Optimization <b>Cluster OpenMP* Intel® Compiler add-on</b> Distributed memory version of OpenMP*, known as Cluster OpenMP*	<b>Deployment on Multiple Fabrics</b> Based on ANL MPICH2, Multiple fabric support, Automated fabric selection, Enhanced process pinning	<b>Optimized Functions For Math Processing</b> BLAS, LAPACK, Sparse Solvers, Fast Fourier Transforms, Vector Math, Statistics ScalaPack, Cluster FFT	<b>The world's best analysis tool for MPI applications</b> Increase productivity and cluster application performance. Very low impact. Excellent scalability on time and processors
<b>Intel® Debuggers</b> Multi-node debug support			

**Highly Optimized  
MPI and Threaded Application Performance**



**Software & Services Group, Developer Products Division**

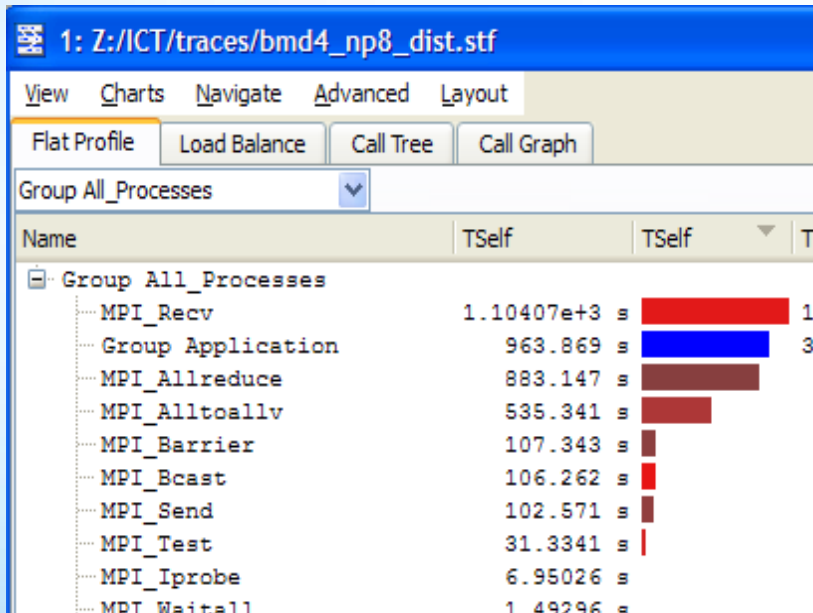
Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

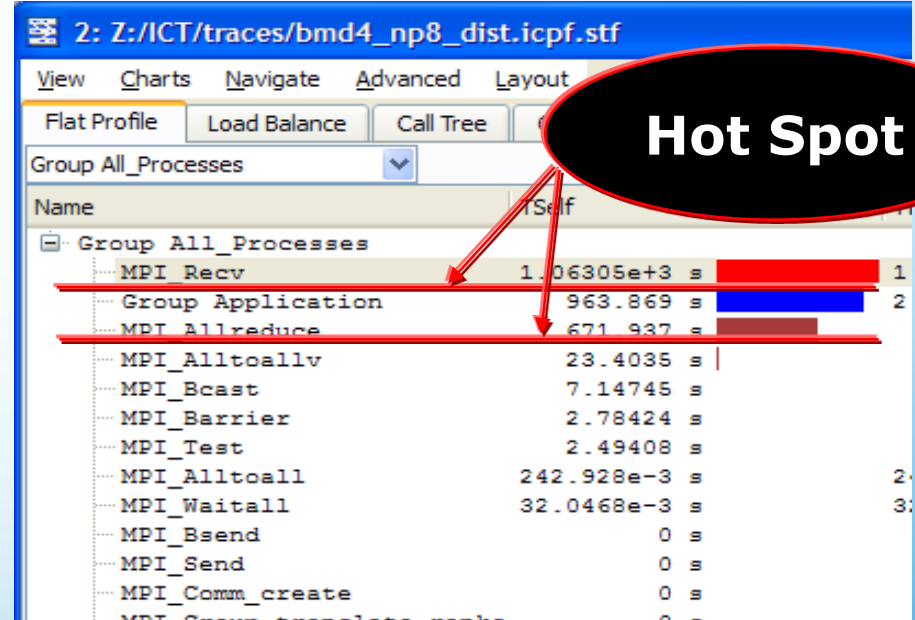
# Ideal Interconnect Simulator (IIS)

Helps to figure out application's imbalance simulating it's behavior in the "ideal communication environment"

Real trace

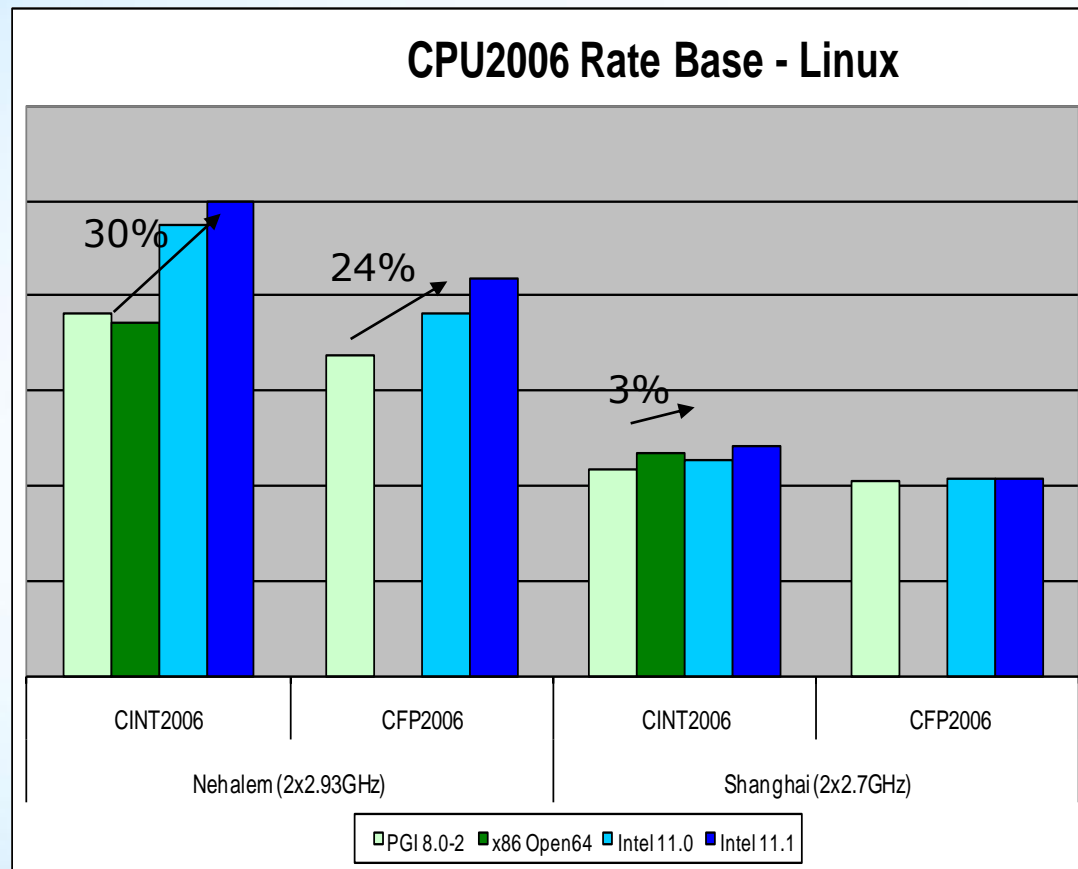


Ideal trace



Easy way to identify application bottlenecks

# Compiler Performance Comparison on Intel and AMD\* processor-based systems



Source: Intel Corporation. Test results aggregated with overall performance scores based on geometric mean. Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to [www.intel.com/performance/resources/benchmark\\_limitations.htm](http://www.intel.com/performance/resources/benchmark_limitations.htm).



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# Further Information



- <http://software.intel.com/en-us/intel-hpc-home>

<http://software.intel.com/en-us/articles/consistency-of-floating-point-results-using-the-intel-compiler/>

<http://software.intel.com/en-us/articles/how-to-check-auto-vectorization/>

<http://software.intel.com/en-us/articles/requirements-for-vectorizable-loops/>

<http://software.intel.com/en-us/articles/intel-compiler-professional-editions-white-papers/>

- **The Intel® C++ and Fortran Compiler User and Reference Guides**,  
[http://software.intel.com/sites/products/documentation/hpc/compilerpro/en-us/cpp/lin/compiler\\_c/index.htm](http://software.intel.com/sites/products/documentation/hpc/compilerpro/en-us/cpp/lin/compiler_c/index.htm) or  
[http://software.intel.com/sites/products/documentation/hpc/compilerpro/en-us/fortran/lin/compiler\\_f/index.htm](http://software.intel.com/sites/products/documentation/hpc/compilerpro/en-us/fortran/lin/compiler_f/index.htm)
- And the User Forums and Knowledge Base,  
<http://software.intel.com/en-us/forums>  
<http://software.intel.com/en-us/articles/tools>



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.



- Comprehensive set of tools for multi-core and cluster parallelism from Intel for x86 architecture
  - Best performance on Intel architecture, and competitive performance on AMD systems
  - Intel tools can be used to standardize x86 development C++/Fortran development
- Our focus is on
  - Best Performance
  - Comprehensive coverage of parallelism
  - Ease of use
  - Compatibility and software investment protection

**Visit <http://intel.com/software/products>**