

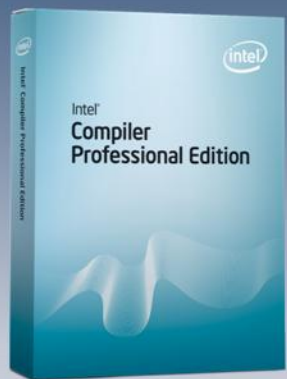


Next Generation Developer Tools and Parallel Programming Models from Intel

Developer Tools of Intel - Today

Fully Supported Developer Products:

Compilers



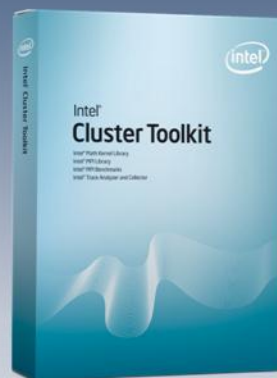
Libraries



Analyzers



Clusters



Parallel Productivity



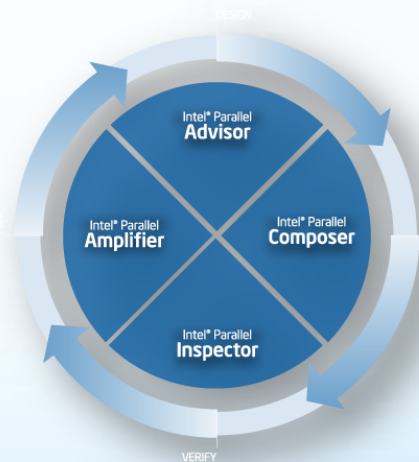
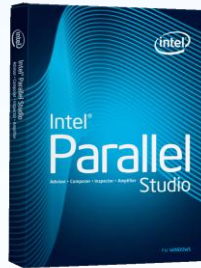
... and numerous unsupported tools like PIN, PTU, AVX Emulator, CnC freely available from whatif.intel.com and other public sites

Intel® Parallel Studio – Windows Only

New Version 2011 Released Sep 2nd !

Intel® Parallel Advisor

- Demystifies and speeds parallel application design
- Direct user where to parallelize
- Explorer & Modeler tools give parallelism design insight and analysis
- Proposes parallelism scheme best suited for application
- Summary view for decision-making



Intel® Parallel Composer

- C++ Compiler, libraries, debugger plug-in
- Intel® Parallel Debugger Extension: Simplify debugging parallel code
- A family of Parallel models **New!** Set of portable, reliable, future proof parallel models for both data and task parallelism, includes Intel TBB, Cilk Plus
- Support for Intel Array Building Blocks
- Intel® IPP, OpenMP* included

Intel® Parallel Inspector

- Dynamic Memory & thread Analysis for serial and parallel code
- Finds thread data races & deadlocks
- Finds memory leaks and corruption

Intel® Parallel Amplifier

- Parallel performance analyzer
- Find both serial and parallel performance bottlenecks
- Scale application performance with more processor cores
- No special compilers or builds necessary

Future: Intel® Parallel Studio XE 2011

Released in Q4/2010

Complete Package for Linux* and Windows*

- Single installer, single license, synchronized launch schedule
- Supports the latest Intel® x86 processors, both IA32 and Intel64

Intel® Composer XE 2011 – “Compiler release 12.0”

- Fortran and C/C++ Compiler Version 12.0
- New parallel programming models Cilk Plus and Co-Array Fortran
- Vectorization enhancements: GAP, SIMD pragma, Array notation
- Enhanced Intel® Debugger IDB

Intel® Inspector XE 2011 – “ThreadChecker next”

- Thread, **Memory** and Security checking
- Support of Parallel Programming Models: Intel® TBB, OpenMP*, Intel® Cilk Plus
- C/C++, Fortran, and .NET support

Intel® VTune™ Amplifier XE 2011 – “VTune+ThreadProfiler Next”

Intel Inspector XE

Next-generation Confidence Tools

- Next generation high-end confidence tools
 - Replaces Intel® Thread Checker
 - Standalone GUI on Linux* and Windows*
 - Superset of Intel® Parallel Inspector features
 - Thread Checking – Race Conditions and Deadlocks
 - Memory Checking – new !!
 - Source Code Checking – display(GUI) part
 - No called “Static Security Analysis” (SSA)
 - Implemented in combination with compiler
- Uses new dynamic instrumentation engine based on Pin tool:
 - More reliable, lower startup overhead
 - Only functions that are called are instrumented
 - Long start up waits for instrumentation are gone

Observations

ID	Description	Problem	Source	Function	Module	Object ...	State
X53	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	14435	Not fixed
X54	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	65555	Not fixed
X55	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	87	Not fixed
X56	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	985	Not fixed
X57	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	1305	Not fixed
X58	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	15033	Not fixed
X59	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	174	Not fixed
X61	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	333927	Not fixed
X62	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	2706675	Not fixed
X63	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	2621459	Not fixed
X64	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	594	Not fixed
X65	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	8211	Not fixed
X66	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	42075	Not fixed
X67	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	42075	Not fixed
X68	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	131091	Not fixed
X60	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	35	Not fixed
X70	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	65555	Not fixed
X71	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	65555	Not fixed
X72	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	65555	Not fixed
X73	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	65555	Not fixed
X75	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	65555	Not fixed
X76	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	65555	Not fixed
X2	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	65555	Not fixed
X3	Allocation site	Memory leak	btaignedallocator.cpp:24	btAllocDefault	DebugMultiThreadedDemo...	65555	Not fixed

Filter

Sort

Severity

Error to All

Problem

Memory leak 78 items

Source

btaignedallocator.cpp 70 items
 demoapplication.cpp 3 items
 [Unknown] 2 items
 dbgheap.c 2 items
 spugatheringcollisionta ... 1 item

Function

btAllocDefault 70 items
 localCreateRigidBody 3 items
 _calloc_dbg 2 items
 UnregisterHook 2 items
 createCollisionLocalSto ... 1 item

Module

DebugMultiThreadedD ... 76 items
 LvHook.dll 2 items

State

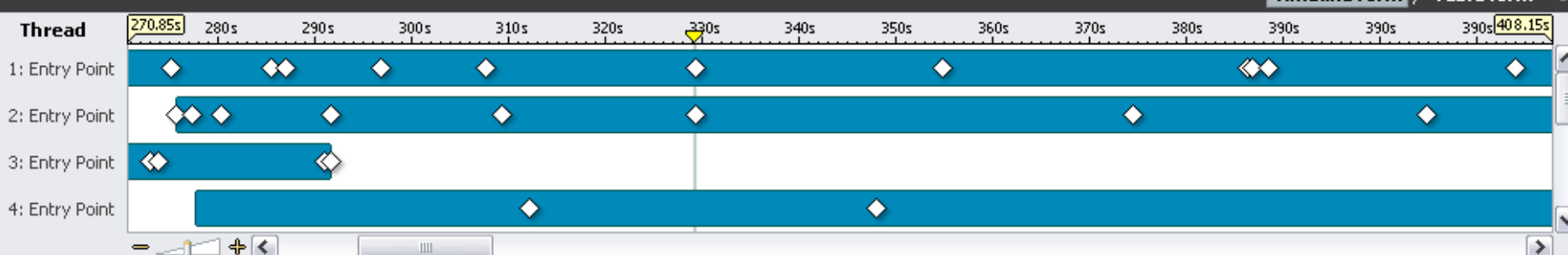
Not fixed 78 items

Details window

multiple observations selected and all instances
 (including stack variation) are shown on the timeline

Observations in Problem Set

Timeline form / Table form



Intel VTune™ Amplifier XE

Next-generation Performance Tool

- Integrates popular and mature features of Intel® VTune™ Performance Analyzer, Intel® Parallel Amplifier, Intel® Thread Profiler and Intel® Performance Tuning Utility
 - But not a super-set in all cases
 - Some additional features being worked on and will be added later; some are still being evaluated/might be added to future updates
- Standalone GUI on Linux* and Windows*
 - GUI in all environments based on wxWidgets: Very fast and stable
 - Same look-and-feel for Linux & Windows
 - Fast and native implementation on Linux
 - No sluggish and fragile emulations anymore !!
- Comprehensive Command Line interface
- New instrumentation technologies for data collection

Intel VTune™ Amplifier XE

Feature Highlights

- **Ease of use is key focus**

- Simple configuration of analysis session
 - Copy and modify existing analysis types to adapt to special needs
- Intuitive filtering and display of data collected
- Stand-alone GUI but also seamless integration into Microsoft Visual Studio* on Windows*

- **Extended Platform Coverage**

- Windows* and Linux
- Microsoft* .NET* C# applications

- **Advanced Source / Assembler View**

- Analysis / event data mapped to the source / assembler code
- View and analyze assembler code as basic blocks

C:\Users\hakyil\Documents\My Amplifier XE Projects\My Amplifier XE Project - Intel VTune Amplifier XE 2011

File Project Help

New Intel VTune Amplifier XE 2011 Result x r037cc

Choose Analysis Type

Analysis Type

- Algorithm Analysis
 - Lightweight Hotspots
 - Hotspots
 - Concurrency
 - Locks and Waits
- Hardware-level Analysis
 - Memory Accesses
 - LLC Misses
 - Data Sharing
- Advanced Hardware-level Analysis
 - Core2 - Bandwidth
 - Core2 - Bandwidth Breakdown
 - Core2 - Cycle Usage
 - Core2 - uOp Flow
 - Corei7 - Cycles and uOps
 - Corei7 - Front End Investment
 - Corei7 - General Exploration
 - Corei7 - Memory Access
- Custom Analysis
 - Memory Latency 0

Details

Events configured for CPU: Intel(R) Xeon(R) Processor

Event Name	Sample After	LBR Filter
CPU_CLK_UNHALTED.REF	2000000	Reference cycles when
CPU_CLK_UNHALTED.THREAD	2000000	Cycles when thread
INST_RETIRED.ANY	2000000	Instructions retired

Buttons: Analyze, Analyze Paused, Project Properties, Resume, Pause, Stop, Cancel, Mark Timeline

Helps creating new analysis types

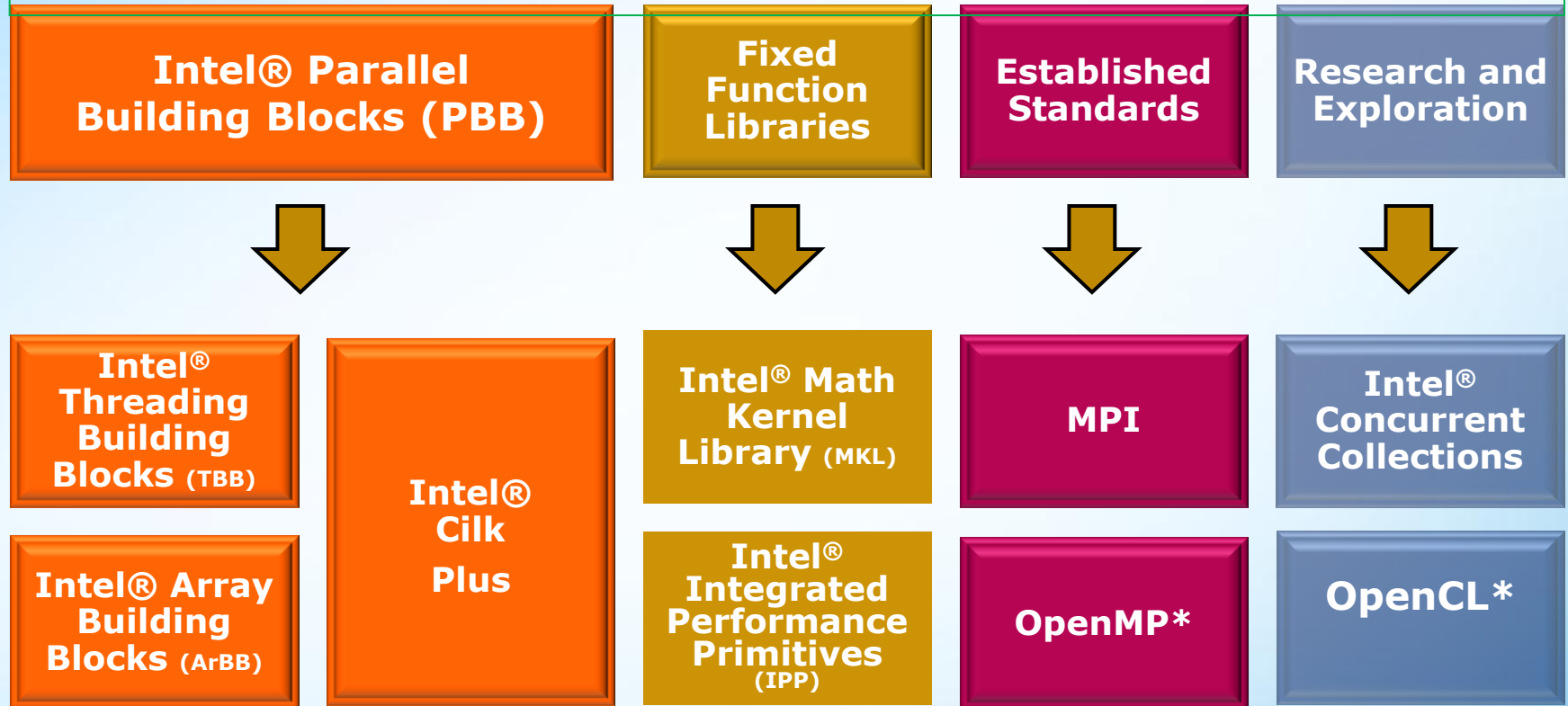
Copy from current
New CPU Event Collector analysis
New Stack-sampling Collector analysis

Copy the commandline to clipboard

Get Command Line

Intel VTune Amplifier XE 2011

Intel's Family of Parallel Programming Models




Intel® Cilk Plus, Intel® TBB: Part of Intel® Parallel Studio (XE)

Intel® Array Building Blocks: Known by code names 'Intel Ct' or 'Intel Firetown'; public beta started around mid September 2010

Intel® Parallel Building Blocks

Comprehensive tools to deliver outstanding app performance



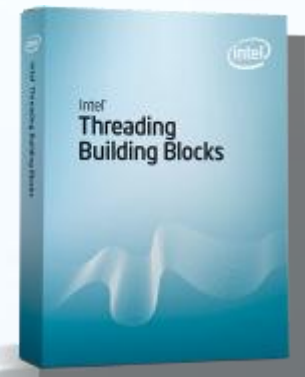
Intel® Parallel Building Blocks

	Intel® Cilk™ Plus	Intel® Threading Building Blocks	Intel® Array Building Blocks
What is it?	Language extensions to simplify task and vector parallelism	Widely used C++ template library for task parallelism	Sophisticated C++ template library for vector parallelism
Features	<ul style="list-style-type: none">▪ 3 Simple keywords and array notations for parallelism▪ Support for Task and Vector parallelism▪ Similar semantics as serial code	<ul style="list-style-type: none">▪ Parallel Algorithms and Data Structures▪ Scalable Memory Allocation and Task Scheduling▪ Synchronization Primitives	<ul style="list-style-type: none">▪ Automatically scales to future Intel platforms▪ Use of cores, threads, SIMD determined by run time compiler
Reasons to Use	<ul style="list-style-type: none">▪ Simple way to parallelize your code▪ Sequentially consistent + low overhead = powerful solution▪ Supports C & C++; Windows* and Linux*	<ul style="list-style-type: none">▪ Rich feature set for general purpose parallelism▪ Available as open source or commercial▪ Supports C++; Windows, Linux, Mac OS*, other OS's	<ul style="list-style-type: none">▪ Used for flexible vector parallelism▪ JIT & VM technology = flexible and powerful▪ Supports C++; Windows & Linux

MIX AND MATCH TO OPTIMIZE YOUR APPLICATION'S PERFORMANCE

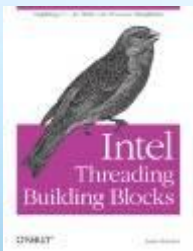
Intel® Threading Building Blocks

- Scalable performance
- Make multi-threaded application development practical
 - C++ template library that uses familiar task patterns, not threads
 - high level abstraction requiring less code for threading without sacrificing performance
 - Reduce maintenance as number of cores increase
- Maximize application performance
 - Appropriately scales to the number of cores available
- Utilize one threading library for 32 & 64 bit Windows*, Linux* and Mac OS* X
 - The thread library API is portable across Linux, Windows, or Mac OS platforms
 - Works with all C++ compilers (e.g., Microsoft, GNU and Intel)



"TBB helped KnowledgeMiner achieve speeds 8x faster on an 8 core system. In addition, a strict redesign of KnowledgeMiner for parallel computing is giving a total speedup over the previous version 400x. This astonishing change in speed allows KnowledgeMiner to operate in almost real time, something we didn't previously think was possible."

Frank Lemke, Founder and President KnowledgeMiner Software



Book available from O'Reilly publishing

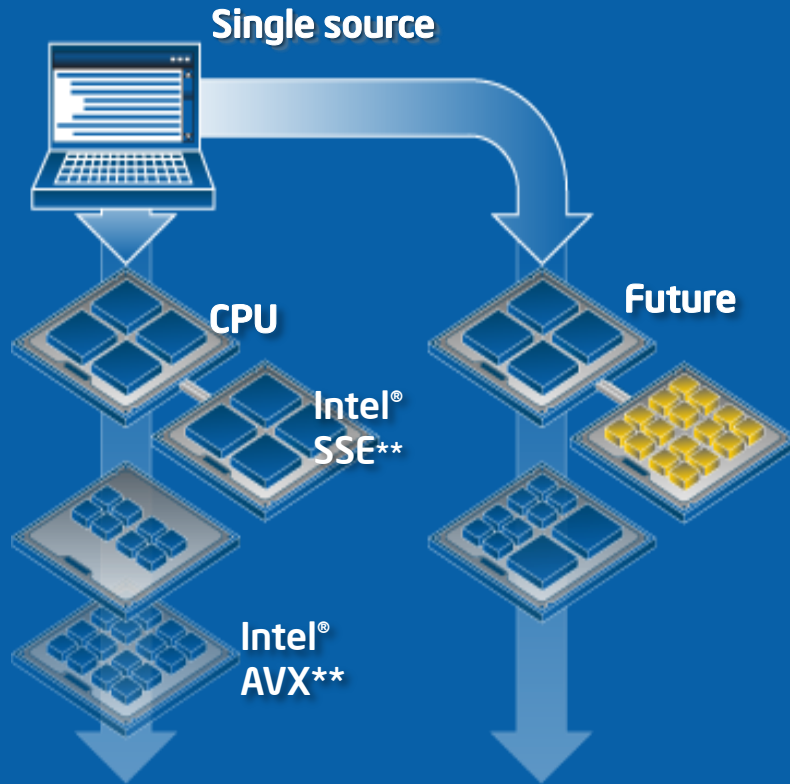


OS	IA-32	Intel® 64	IA-64
Windows*	•	•	•
Linux*	•	•	•
Mac OS* X	•	•	

Intel® Threading Building Blocks Components

Intel® Threading Building Blocks v3.0		
Generic Parallel Algorithms	Concurrent Containers	Task Scheduler
<ul style="list-style-type: none"> ▪ <code>parallel_for(range)</code> ▪ <code>parallel_reduce</code> ▪ <code>parallel_for_each(begin, end)</code> ▪ <code>parallel_do</code> ▪ <code>parallel_invoke</code> ▪ <code>pipeline</code> ▪ <code>parallel_pipeline</code> ▪ <code>parallel_sort</code> ▪ <code>parallel_scan</code> 	<ul style="list-style-type: none"> ▪ <code>concurrent_hash_map</code> ▪ <code>concurrent_queue</code> ▪ <code>concurrent_bounded_queue</code> ▪ <code>concurrent_vector</code> ▪ <code>concurrent_unordered_map</code> 	<ul style="list-style-type: none"> ▪ <code>task_group</code> ▪ <code>structured_task_group</code> ▪ <code>task_scheduler_init</code> ▪ <code>task_scheduler_observer</code>
Synchronization Primitives		Memory Allocation
<ul style="list-style-type: none"> ▪ <code>atomic</code> ▪ <code>mutex</code> ▪ <code>recursive_mutex</code> ▪ <code>spin_mutex</code> ▪ <code>spin_rw_mutex</code> ▪ <code>queuing_mutex</code> 	<ul style="list-style-type: none"> ▪ <code>queuing_rw_mutex</code> ▪ <code>reader_writer_lock</code> ▪ <code>critical_section</code> ▪ <code>condition_variable</code> ▪ <code>null_mutex</code> ▪ <code>null_rw_mutex</code> 	<ul style="list-style-type: none"> ▪ <code>tbb_allocator</code> ▪ <code>cache_aligned_allocator</code> ▪ <code>scalable_allocator</code> ▪ <code>zero_allocator</code>
Thread Local Storage	Threads	Miscellaneous
<ul style="list-style-type: none"> ▪ <code>enumerable_thread_specific</code> ▪ <code>combinable</code> 	<ul style="list-style-type: none"> ▪ <code>thread</code> 	<ul style="list-style-type: none"> ▪ <code>tick_count</code> ▪ <code>captured_exception</code> ▪ <code>moveable_exception</code>

Intel® Array Building Blocks



Productivity

- Integrates with existing tools
- Applicable to many problem domains
- Safe by default: maintainable

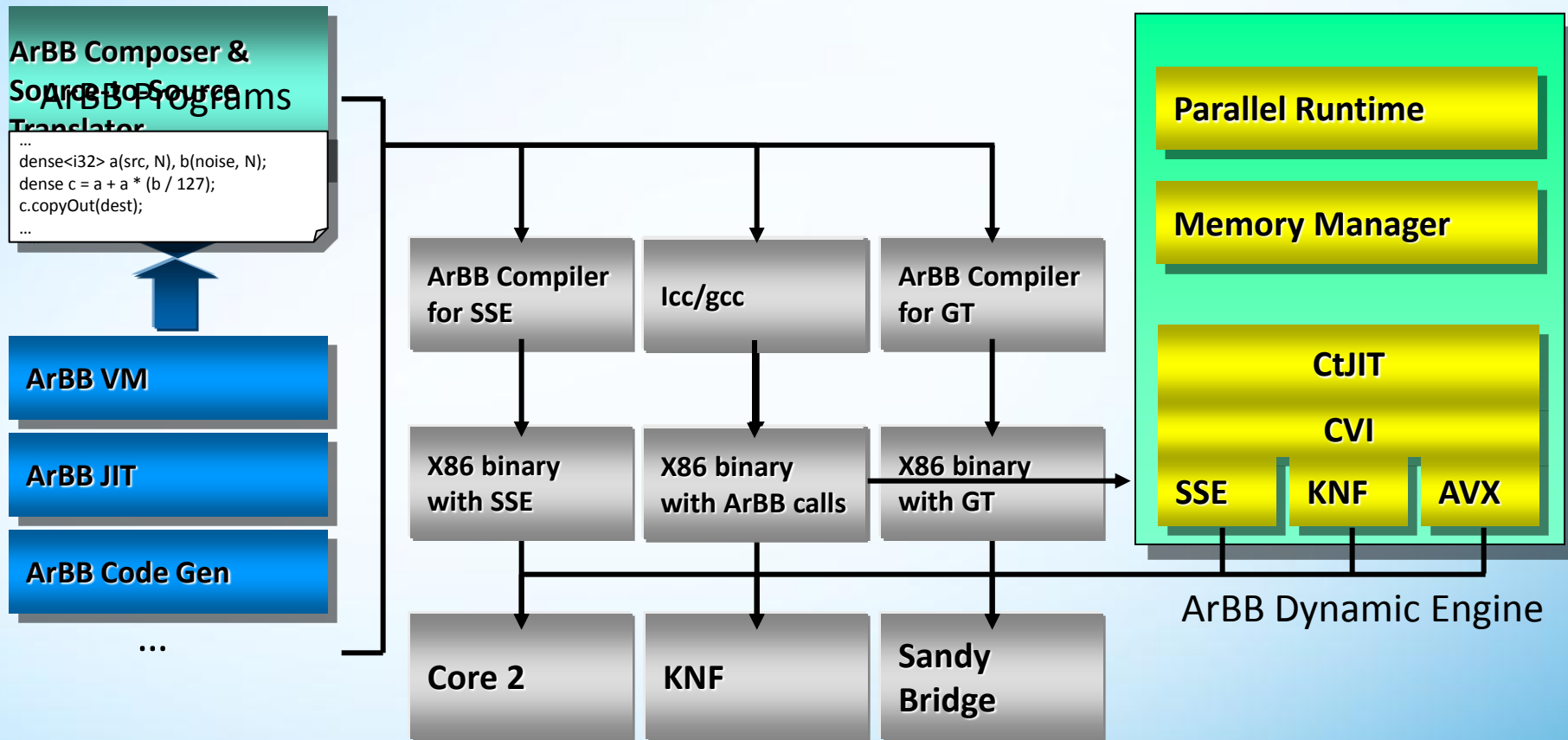
Performance

- Efficient and scalable
- Harnesses both vectors and threads
- Eliminates modularity overhead of C++

Portability

- High-level abstraction
- Hardware independent
- Forward scaling

Supporting All Intel Platforms



Use Animation

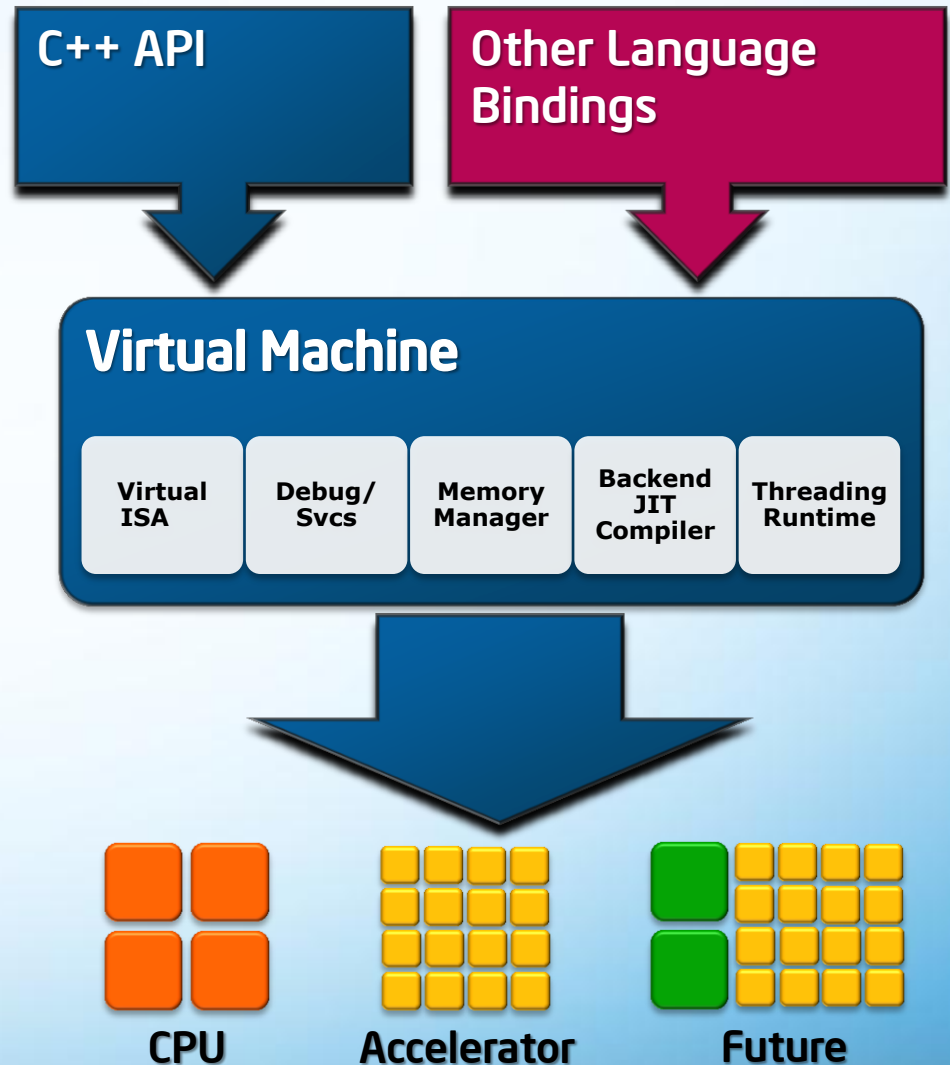
Developer Products Division

Software

Copyright © 2010, Intel Corporation. All rights reserved. *Other brands and names are the property of their respective owners.

The ArBB Runtime

- Intel ArBB Technology offers a standards compliant C++ library...
... backed by a runtime
- Runtime generates and manages threads and densetor code, via
 - Machine independent optimization
 - Offload management
 - Machine specific code generation and optimizations
 - Scalable threading runtime (based on TBB!)



Intel® Cilk™ Plus

Combines and integrates a task-parallel and data (vector) parallel programming language extension for C/C++ :

- Task parallelism
 - Realized by **Cilk** as defined by MIT Cilk project
 - Only 3 simple keywords
 - Hyperobjects for reductions etc
 - Exploits core/thread parallelism
 - In general not deterministic program execution
- Data parallelism
 - Realized by
 - **C Array Notation**: Array sections in C/C++
 - **SIMD Pragma**: new pragma class for vectorization
 - **Elemental function**: Functions operating on array elements in parallel
 - Exploits SSE/AVX parallelism
 - Not restricted to a single core however
 - Deterministic execution model

Intel® Cilk™ Plus

Intel Cilk Plus What is it?

- Compiler assisted solution offering a tasking system via 3 simple keywords
- Includes array notation to specify vector code
- Has a hyper objects library which offers powerful parallel data structures
- Based on 15 years of research at MIT
- Pragmas to force vectorization of loops and specify functions that can be applied to all elements of arrays

Intel Cilk Plus Key Benefits

- Simple syntax which is very easy to learn and use
- Array notation guarantees fast vector code
- Pragmas guarantee vectorization of loops over arbitrary user code
- Fork/join tasking system is simple to understand and offers safety from errors
- Low overhead tasks offer scalability to high core counts
- Hyper objects enable reductions which give the same answers as serial code
- Mixes with Intel TBB and Intel ArBB for a complete task and vector parallel solution

Intel® Cilk™ Plus – when to use

- Seeking task or vector parallelism
- Serial semantics task based parallelism is required
- Reduction operations need consistent answers as number of cores vary
- Need a compiled language with no JIT/VM capability
- A fork/join tasking model is sufficient
- Need to guarantee array notation or loops run as high performance vector code
- Vectorize loops over arbitrary user functions applied to entire arrays

Cilk Plus

A powerful yet simple & easy to learn compiler assisted capability offering low-overhead, high-performance task & vector parallelism