

COMPILER & THREADED LIBRARIES

# Intel® Parallel Composer 2011

## Product Brief

Intel® Parallel Composer 2011



“Here at Trading Systems Lab, we got a 10% to 20% performance boost in the multimode trading simulator that’s used in our TSL Algo Auto-Design Platform by using the C++ Compiler in Intel® Parallel Studio. The compatibility with Microsoft Visual C++\* is great, and we’re looking forward to using more parallelism features in Parallel Studio.”

Mike Barna  
President  
Trading Systems Lab

## Build Serial and Threaded C and C++ Applications for Multicore Systems

Intel® Parallel Composer 2011 is a powerful set of Intel® C++ Compilers, performance libraries, parallel development models, and debugging capabilities for developers interested in delivering outstanding application performance in Windows\*-based client software. It integrates into Microsoft Visual Studio 2005\*, 2008\*, and 2010\*; is compatible with Microsoft Visual C++\*; supports the way you work; and protects your IDE investments.

Intel Parallel Composer can be purchased as part of Intel® Parallel Studio or as an individual product. In addition to Intel Parallel Composer, Intel Parallel Studio includes:

- An innovative threading assistant  
**Intel® Parallel Advisor** assists developers implementing parallelism in existing applications.
- A memory and threading error checker  
**Intel® Parallel Inspector** analyzes memory errors, including memory leaks, and is a thread debugging aid that checks thread correctness.
- A threading and performance profiler  
**Intel® Parallel Amplifier** speeds and simplifies finding multicore performance bottlenecks.

### Intel Parallel Composer Components

- Intel C++ Compilers for 32-bit processors and a cross-compiler to create 64-bit applications on 32-bit systems
- Intel® Parallel Debugger Extension, which integrates with the Microsoft Visual Studio\* debugger
- Intel® Parallel Building Blocks, which gives developers choices for implementing parallelism. It includes:
  - Intel® Threading Building Blocks (Intel® TBB) for task-based parallelism. Intel TBB enhances developer productivity for scalable high performance, cross-platform software on multicore platforms. Also available in open-source form.

- Intel® Cilk™ Plus, a C/C++ language extension. It applies loop-and-task parallelism and a compiler-assisted programming model, enabling a faster way to implement parallelism in C/C++ code.

- Intel® Array Building Blocks (available in beta) implements data parallelism for compute-intensive data parallel applications and enhances developer productivity for scalable higher performance applications on multicore and manycore.

▪ Intel® Integrated Performance Primitives (Intel® IPP). Intel IPP is an extensive library of multicore-ready, highly optimized software functions for multimedia, data processing, basic math functions, and communications applications. Intel IPP includes both hand-optimized, primitive-level functions and high-level threaded solutions such as codecs. It can be used for both Visual C++ and .NET development, in addition to working with the Intel® C++ Compiler. For more advanced math functions, Intel® Math Kernel Library is available separately.

▪ Sample code and a great Getting Started Guide to get you going quickly.

## It's Easy to Get Started and Stay Connected with a Growing Developer Community

Intel Parallel Composer includes sample source code that is featured in an easy-to-use Getting Started Guide. There are also short videos on the website and referenced in the Getting Started Guide.

The Getting Started Guide is accessible on the website, from the Microsoft Visual Studio\* Help menu (along with in-depth documentation), and from the Intel Parallel Studio or Intel Parallel Composer entries accessible from the Windows "Start" button. There is even a prompt for it upon completion of installation. Whether you are a beginner, a pro, or somewhere in between, it's worth a few minutes to go through the Getting Started Guide.

Once you get going, you will find it useful to join the growing community of developers taking advantage of systems based on Intel® multicore processors. Intel provides a dynamic forum for developers to exchange ideas, post comments and questions, and earn points to become an Intel® Black Belt Software Developer. Intel also provides blogs, downloads, and a large and growing knowledge base with a variety of topics for developers interested in parallelism. Visit the parallel programming and multicore community at <http://software.intel.com/en-us/articles/intel-parallel-studio/>.

## Intel C++ Compiler: Microsoft Visual Studio Integration and Compatibility

All Intel Parallel Composer features seamlessly integrate into Microsoft Visual Studio 2005, 2008, and 2010. They do not replace any Microsoft Visual Studio\* tools and do not change any option settings; the features are available as easy-to-use-options. Compiler features are available from additional toolbars that are installed with Intel Parallel Composer, from certain Visual Studio pull-down menus, such as the Project menu. Access them by right-clicking over a solution or project file or from the properties' popup windows. It's easy to tell Microsoft Visual Studio to use the Intel C++ Compiler for a solution or selected projects, and just as easy to switch back to Visual C++ Compiler. The compiler offers native 32-bit development and a cross-compilation environment (32-bit host to develop 64-bit applications). You have the option of installing only the 32-bit capability, only the 64-bit capabilities, or both.

The Intel C++ Compiler is binary compatible with Visual C++ and, in many cases, can offer a significant performance advantage to modules or full applications built with the compiler. Another advantage of using the Intel compiler is the associated Intel Parallel Debugger Extension, which can speed debugging of parallelized code. Using Intel C++ offers developers a number of advantages, but it is not required for the use of other components in Intel Parallel Composer or the full Intel Parallel Studio. You can use Intel TBB and Intel IPP with the Visual C++ Compiler or with the Intel C++ Compiler. You can also use Intel Parallel Inspector's memory leak, concurrency checking, and other capabilities on applications built with Visual C++ or the Intel C++ Compiler. In short, there are good reasons for you to use the full Intel Parallel Studio suite, including a powerful, easy-to-use, compatible Intel C++ Compiler, though you can also continue to use Visual C++.

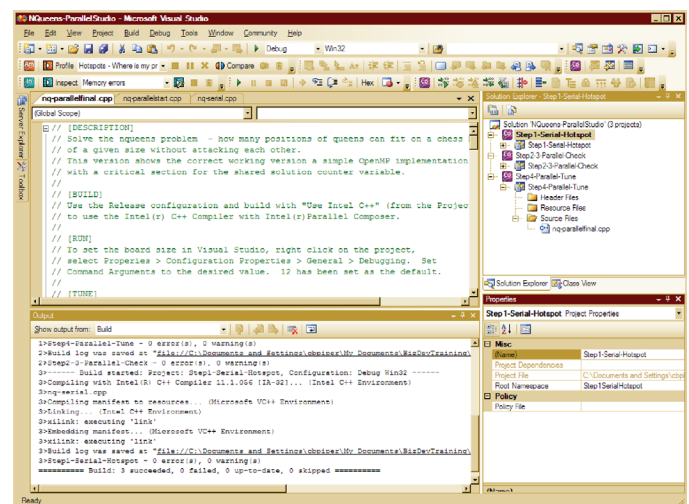


Figure 1: Intel® Parallel Composer integrates into Microsoft Visual Studio\*. The solution on display is from the sample code used in the Intel® Parallel Studio Getting Started Guide.

# Intel Parallel Building Blocks

## Intel TBB

Intel Threading Building Blocks (Intel® TBB) is a C++ template library solution that can be used to enable general parallelism. It includes scalable memory allocation, load-balancing, work-stealing task scheduling, a thread-safe pipeline and concurrent containers, high-level parallel algorithms, and numerous synchronization primitives. Intel TBB is for C++ developers who write general-purpose loop and task parallelism applications.

Intel TBB solves three key challenges for threaded programming:

- **Productivity:** Simplifies the implementation of threading
- **Correctness:** Helps eliminate parallel synchronization issues
- **Maintenance:** Aids in the creation of applications ready for tomorrow, not just today

Advantages of using Intel TBB:

- **Future-proof applications:** As the number of cores (and threads) increases, application speedup increases using Intel TBB's sophisticated task scheduler.
- **Portability:** Implement parallelism once to execute threaded code on multiple platforms
- **Interoperability:** Benefit from an ability to work with a variety of threading methods, hardware, and operating systems

**Active open-source community:** Intel TBB is also available in an open-source version. [opentbb.org](http://opentbb.org) is an active site with forums, blogs, code samples, and much more.

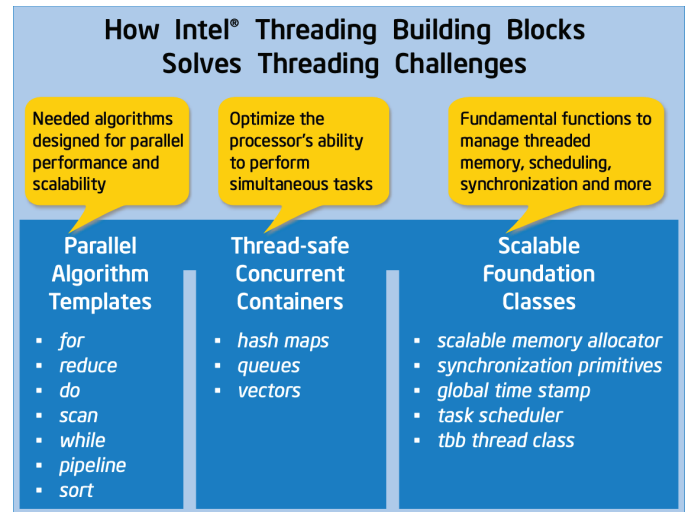


Figure 2: Major function groups within Intel® TBB

Intel TBB can be used to solve a wide variety of threading tasks. Figure 3 presents three common threading problems addressed by Intel TBB.

Problem	Solution
How to simplify adding threading	<b>Intel® TBB <code>parallel_for</code> command</b> <ul style="list-style-type: none"> <li>▪ Straightforward replacement of <i>for/next</i> loops to exploit advantages of threading</li> <li>▪ Load-balanced parallel execution of fixed number of independent loop iterations</li> </ul>
Management of threads to get best scalability	<b>Intel® TBB Task Scheduler</b> <ul style="list-style-type: none"> <li>▪ Manages thread pool and hides complexity of native threads</li> <li>▪ Designed to address common performance issues of parallel programming                             <ul style="list-style-type: none"> <li>- Oversubscription: One scheduler thread per hardware thread</li> <li>- High overhead: Programmer specifies tasks, not threads</li> <li>- Load imbalance: Work-stealing balances load</li> </ul> </li> </ul>
Memory allocation is a bottleneck in concurrent environment	Intel TBB provides tested, tuned, and scalable memory allocator based on per-thread memory management algorithm <ul style="list-style-type: none"> <li>▪ As an allocator argument to STL template classes</li> <li>▪ As a replacement for <i>malloc/realloc/free</i> calls (C programs)</li> <li>▪ As a replacement for global <i>new</i> and <i>delete</i> operators (C++ programs)</li> </ul>

Figure 3: Intel® TBB addresses three major threading issues

## Intel® Cilk™ Plus

Intel® Cilk™ Plus is an Intel C/C++ Compiler-specific implementation of parallelism: It offers superior functionality by combining vectorization features with high-level, loop-type data parallelism and tasking. Intel Cilk Plus is for C++ software developers who write simple loop, data, and task parallel applications.

## Intel® Array Building Blocks

(In beta) Intel® Array Building Blocks is an API backed by a sophisticated runtime library. It provides a generalized data parallel programming solution that frees application developers from dependencies on particular low-level parallelism mechanisms or hardware architectures. It produces scalable, portable, and deterministic parallel implementations from a single, high-level, maintainable, and application-oriented specification of the desired computation. Intel Array Building Blocks is for software developers who write compute-intensive, data parallel algorithms. Intel Array Building Blocks is currently available in beta at: <http://software.intel.com/en-us/data-parallel/>.

## Intel® Integrated Performance Primitives (Intel® IPP)

Parallel Composer includes Intel® IPP and offers an extensive library of multicore-ready, highly optimized software functions for multimedia, data processing, and communications applications. It contains thousands of optimized functions covering frequently used fundamental algorithms in image processing, image compression, data compression, video coding, string processing, cryptography, signal processing, audio coding, speech coding, speech recognition, computer vision, image color conversion, and vector/matrix mathematics. For more complex math, Intel® Math Kernel Library (Intel® MKL) is also available as an option.

Intel IPP includes both hand-optimized primitive level functions and high-level threaded solutions, such as codecs, and can be used for both Visual C++ and .NET development. All of these functions and solutions are fully thread-safe, and many are internally threaded, helping you get the most out of today's multicore processors and scale to future manycore processors.

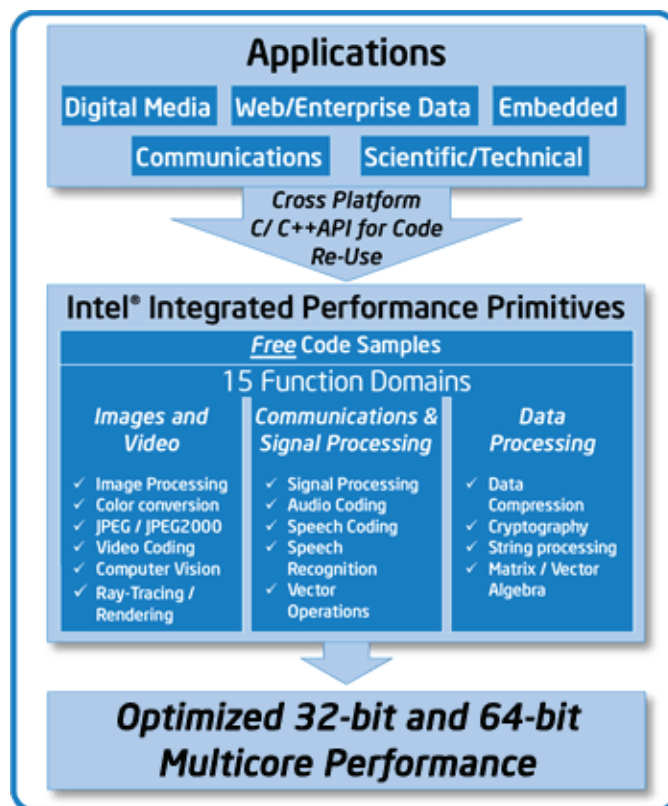


Figure 4: Intel® IPP is included in Intel® Parallel Composer, and features threaded and thread-safe library functions over the wide variety of domains listed above

## Intel IPP Performance

Depending on the application and workload, Intel IPP functions can perform many times faster than the equivalent compiled C code. In Figure 5, the same operation that required 338 microseconds to execute in compiled C++ code required only 111 microseconds when Intel IPP image processing functions were used. That is a 300 percent performance improvement.

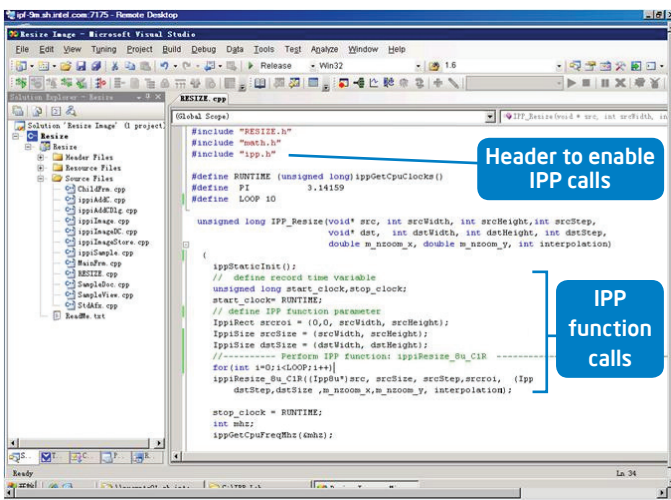


Figure 5: It's easy to incorporate Intel® IPP library calls into your Microsoft Visual Studio\* code

In addition to C++ projects, Intel IPP can also be used in C# projects using the included wrapper classes to support calls from C# to Intel IPP functions in the string processing, image processing, signal processing, color conversion, cryptography, data compression, JPEG, matrix, and vector math domains.

In this image-resizing example, Intel® IPP code ran 3x faster than compiled C++ code.



Figure 6. In this image-resizing example (from 256 x 256 bits to 460 x 332 bits), the Intel® IPP-powered application ran in 111 msec vs. 338 msec for compiled C++ code (system configuration: Intel® Xeon®, 2.9 GHz, 2 processors, 4 cores/processor, 2 threads/processor)

## Optimize Embarrassingly Parallel Loops

Algorithms that display data parallelism with iteration independence lend themselves to loops that exhibit “embarrassingly parallel” code. Intel Parallel Composer supports three techniques to maximize the performance of such loops with minimal effort: auto-vectorization, use of Intel-optimized valarray containers, and auto-parallelization. Intel Parallel Composer can automatically detect loops that lend themselves to auto-vectorization. This includes explicit *for* loops with static or dynamic arrays, vector and valarray containers, or user-defined C++ classes with explicit loops. As a special case, implicit valarray loops can either be auto-vectorized or directed to invoke optimized Intel IPP library primitives. Auto-vectorization and use of optimized valarray headers optimizes the performance of your application to take full advantage of processors that support the Streaming SIMD Extensions (SSE).

Auto-parallelization improves application performance by finding parallel loops capable of being executed safely in parallel and automatically generating multithreaded code, allowing you to take advantage of multicore processors. It relieves you from having to deal with the low-level details of iteration partitioning, data sharing, thread scheduling, and synchronizations.

Auto-parallelization complements auto-vectorization and use of optimized valarray headers, giving you optimal performance on multicore systems that support SSE. For more information on multithreaded application support, see the user guide (<http://software.intel.com/en-us/intel-parallel-composer/>), then click the documentation link).

New in this release is an extension to auto-vectorization called Guided Auto-Parallelization, or GAP, which is an analyzer. The results of running a GAP analysis appear in the build-log and provide you with suggested source code changes that, if made, will cause the compiler to safely apply the auto-vectorization and auto-parallelization capabilities to optimize your application.

## Intel Parallel Debugger Extension

Intel Parallel Composer includes the Intel Parallel Debugger Extension, which can be accessed through the Microsoft Visual Studio\* Debug pull-down menu. It features support for the extensions included in the Intel C++ Compiler and can be used to help debug code compiled with the Visual C++ Compiler. Note that it is an extension to the Microsoft Visual Studio Debugger, not a replacement.

The Intel Parallel Debugger Extension provides you with additional insight and access to shared data and data dependencies in your parallel application. This facilitates faster development cycles and early detection of potential data access conflicts that can lead to serious runtime issues.

To take advantage of the advanced features of the Intel Parallel Debugger Extension, such as shared data event detection and function re-entrancy detection, compile your code with the Intel Compiler using the `/debug:parallel` option for debug info instrumentation.

For more information, see the Intel Parallel Debugger Extension white paper at <http://software.intel.com/en-us/articles/parallel-debugger-extension/>.

## System Requirements

Microsoft Visual Studio 2005\*, 2008\*, 2010\*  
(except the Express Edition)

For the latest system requirements, go to:

[www.intel.com/software/products/systemrequirements/](http://www.intel.com/software/products/systemrequirements/).

## Support

Purchase of Intel® Parallel Studio products include Premium Support service which allows you to submit questions, access to product updates, and technical documentation.

For more information, go to

<http://software.intel.com/en-us/articles/intel-parallel-studio/>.

## Download a Trial Version Today

Evaluation copy available at:

[www.intel.com/software/products/ParallelStudio/](http://www.intel.com/software/products/ParallelStudio/)

## Features Summary

- Intel Parallel Composer integrates into Visual Studio and preserves your IDE investment, while providing a number of features to help you deliver outstanding application performance. It extends Visual Studio and does not replace any Microsoft Visual Studio functionality.
- Intel C++ Compiler is part of Intel Parallel Composer and is compatible with Microsoft Visual C++.
- Intel Parallel Debugger Extension integrates with the Microsoft Debugger\*, enhancing Microsoft Visual Studio\* to help find and address threading issues. This saves time in getting applications ready to be used.
- Intel Parallel Composer includes simple concurrency functions, data parallel arrays, and thousands of threaded library functions, which simplify threading tasks and speed application development.
- Auto-parallelization and auto-vectorization options, including the new Guided Auto-Parallelization feature, are part of Intel Parallel Studio and simplify development and save time.
- Intel Parallel Building Blocks provides easy-to-use parallelism functions for task and data parallelism, supports faster development of parallel applications, and smoothes development of great-performing serial applications. It includes Intel TBB, Intel Cilk Plus, and Intel ArBB (beta). In addition, integrated array notation, data-parallel Intel IPP functions are included, speeding audio, video, signal analysis, and other application classes.
- There is extensive documentation, including code samples, and a Getting Started Guide.
- Community support includes the growing community of developers adding threading to their code. Draw on the experience of others, contribute your own knowledge and experience, and win prizes while doing it.
- Intel Parallel Composer is also available as part of Intel Parallel Studio, which includes Intel Parallel Advisor, which gives you advice on how to parallelize existing applications; Intel Parallel Inspector, which helps as a debugging aid for things like memory leaks and thread correctness; and Intel Parallel Amplifier, a profiler that helps you get every measure of application performance.

## The Ultimate All-in-One Performance Toolkit—Intel® Parallel Studio 2011

### Designed for today's serial applications and tomorrow's software innovators

Intel brings simplified threading to Microsoft Visual Studio\* C++ developers with a complete productivity solution designed to optimize serial and new threaded applications for multicore and scale for manycore.

#### INNOVATIVE THREADING ASSISTANT

**Intel® Parallel Advisor 2011:** Demystify and speed threaded application design.

#### COMPILER AND THREADED LIBRARIES

**Intel® Parallel Composer 2011:** Develop effective applications with a C/C++ compiler and advanced threaded libraries.

#### MEMORY AND THREADING ERROR CHECKER

**Intel® Parallel Inspector 2011:** Ensure application reliability with proactive parallel memory and threading error checking.

#### THREADING AND PERFORMANCE PROFILER

**Intel® Parallel Amplifier 2011:** Quickly find bottlenecks and tune threaded applications for scalable multicore performance.

