

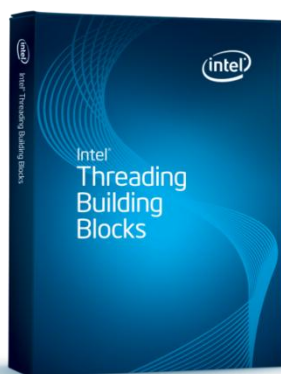
PROVEN C++ TEMPLATE LIBRARY FOR
WINDOWS*, LINUX*, AND MAC OS* X



Intel® Threading Building Blocks 4.0

Product Brief

Intel® Threading Building Blocks 4.0
For Windows*, Linux* and Mac OS* X



“Using Intel TBB’s new flow graph feature, we accomplished what was previously not possible, parallelize a very sizable task graph with thousands of interrelationships - all in about a week.”

Robert Link, GCAM Project Scientist, Pacific Northwest National Laboratory

“After evaluating Intel TBB, we realized that it was not only fulfilling our requirements but that it was also opening the door to a bunch of opportunities to take even more advantage of parallelism. TBB is definitely the way to go. Do not reinvent the wheel: use TBB.

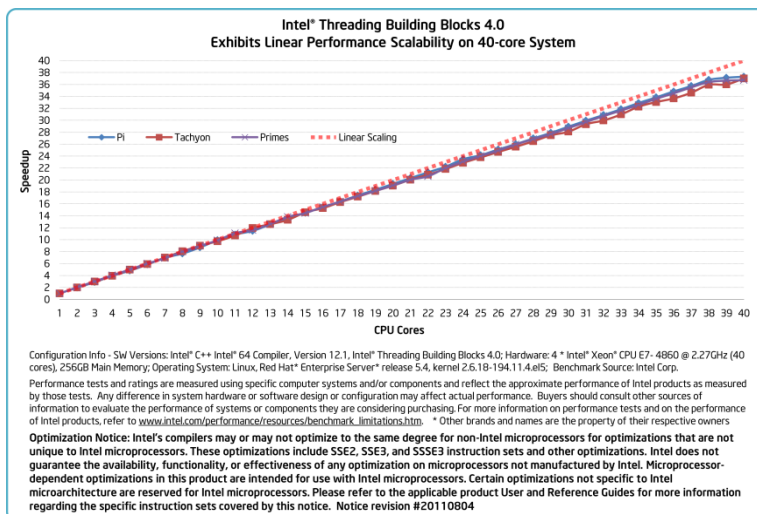
Bernard Laberge, Senior Principal Engineer, Avid Media Composer

Everything You Expect in Parallel Application Design: Productivity, Scalability, Portability, Composability and Performance.

- Widely used, award-winning C++ template library
- Future-proof parallel applications that tap multicore and manycore power
- Provides components to implement higher-level, task-based parallelism
- Compatible with multiple environments and easy to maintain

Features

- **Enhanced Productivity and Reliability**—Intel® TBB provides abstractions that make it easier to write scalable and reliable parallel applications with fewer lines of code. Task-based algorithms, concurrent containers, synchronization primitives, and a scalable memory allocator simplify parallel application development.
- **Scalability with Future-proofing**—Application performance automatically improves as processor core count increases by using abstract tasks. The sophisticated task scheduler dynamically maps tasks to threads to balance the load among available cores, preserve cache locality, and maximize parallel performance.
- **Portability**—Intel® TBB is validated and commercially supported on Windows*, Linux*, and Mac OS* X platforms, using multiple compilers. It is also available on FreeBSD*, IA Solaris*, Xbox* 360, and PowerPC-based systems via the open source community. Organizations can expand their customer base by using a production-ready, open solution for parallelism that is available on a broad range of platforms.
- **Composability** - Multiple Intel® TBB-based modules seamlessly interoperate in a user’s application ensuring efficient use of available hardware parallelism. Intel® TBB also ensures cooperative co-existence with other programming models within Intel’s family of parallel programming models.
- **Performance Advantage**—Intel® TBB is optimized for scalable multicore architectures including Non-Uniform Memory Access (NUMA). Intel® TBB delivers higher-performing and reliable code with less effort than hand-made threading.



Intel® Threading Building Blocks yields linear scaling on a 40 core system
in these three example applications

Features and Benefits

Feature	Benefit
Performance and Productivity	
Parallel Algorithms Generic implementation of common parallel performance patterns	Generic implementations of parallel patterns such as parallel loops, flow graphs, and pipelines can be an easy way to achieve a scalable parallel implementation without developing a custom solution from scratch.
Scheduler Engine that manages parallel tasks and task groups	Intel® TBB task scheduler supports task-based programming and utilizes task-stealing for dynamic workload balancing – a scalable and higher level alternative to managing OS threads manually. The implementation supports C++ exceptions, task/task group priorities, and cancellation which are essential for large and interactive parallel C++ applications.
Concurrent Containers Generic implementation of common idioms for concurrent access	Intel® TBB concurrent containers are a scalable alternative to serial data containers. Serial data structures (such as C++ STL containers) often require a global lock to protect them from concurrent access and modification; Intel® TBB concurrent containers allow multiple threads to concurrently access and update items in the container maximizing the amount of parallel work and improving application's scalability.
Synchronization Primitives Exception-safe locks; mutexes, condition variables, and atomic operations	Intel® TBB provides a comprehensive set of synchronization primitives with different qualities that are applicable to common synchronization strategies. Exception-safe implementation of locks help to avoid dead-locks in C++ programs which use C++ exceptions. Usage of Intel® TBB atomic variables instead of C-style atomic API minimizes potential data races.
Scalable Memory Allocators Scalable memory manager and false-sharing free memory allocator	The scalable memory allocator avoids scalability bottlenecks by minimizing access to a shared memory heap via per-thread memory pool management. Special management of large (>=8KB) blocks allow more efficient resource usage, while still offering scalability and competitive performance. The cache-aligned memory allocator avoids false-sharing by not allowing allocated memory blocks to split a cache line.
Documentation and Samples	A complete documentation package and code samples are readily available both as a part of Intel® TBB installation and online. The Getting Started Guide and the Tutorial provides an introduction into Intel® TBB. The Reference Manual contains a formal descriptions of all classes and functions implemented in Intel® TBB, while the Design Patterns discuss common parallel programming patterns and how to implement them using Intel® TBB.
Flexibility	
Applicability to Various Application Domains	The Intel® TBB flow graph as well as generic template functions are customizable to a wide variety of problems.
User-Defined Tasks	When an algorithm cannot be expressed with high-level Intel® TBB constructs, the user can choose to create arbitrary task trees. Tasks can be spawned for better locality and performance or enqueued to maintain FIFO-like order and ensure starvation-resistant execution.
Forward-scaling	
Dynamic Task Scheduling	Intel® TBB allows a developer to think of parallelism at the higher level avoiding dealing with low level details of threading. A developer expresses a parallel model in terms of parallel tasks and relies on Intel® TBB to execute them in an efficient way by dynamically detecting the appropriate number of threads. This makes Intel® TBB based solutions independent of the number of CPU's and allows for improved performance and scalability with the growing number of CPUs in the future.
Composability	
Support for Various Types of Parallelism	Intel® TBB task scheduler and parallel algorithms support nested and recursive parallelism as well as running parallel constructs side-by-side. This is useful for introducing parallelism gradually and helps independent implementation of parallelism in different components of an application.
Compatibility	
Co-existence with Other Threading Packages	Intel® TBB is designed to co-exist with other threading packages and technologies (Intel® Cilk™ Plus, Intel® OpenMP, OS threads, etc.). Different components of Intel® TBB can be used independently and mixed with other threading technologies.
Compiler-independent Solution	Intel® TBB is a library solution and can be used in software projects built by multiple compilers, across numerous platforms.
Simple Licensing	
Royalty-free Distribution	Redistribute unlimited copies of the Intel® TBB libraries and header files with your application.
Open Source version	Available for download from threadingbuildingblocks.org . The broad support from an involved community provides developers access to additional platforms and OS's.

New features in Intel® TBB 4.0

Feature	Benefit
Flow Graph Dependency graphs and data flow graphs	Flexible and convenient API for expressing static and dynamic dependencies between computations. Also extends applicability of Intel® TBB to event-driven/reactive programming models.
Task and task group priorities	Provides ability to specify task execution order based on three priority levels (low, normal, and high). Static priorities are supported for enqueued tasks; dynamic priorities are supported for task groups.
Memory Pools	Enables Intel® TBB memory allocator to work on user-provided memory regions. This mechanism allows greater flexibility and performance by getting thread-safe and scalable object allocation in the application-specific memory blocks with custom life-span and growth policies.
Concurrent Priority Queue Priority queue with concurrent operations	A queue that allows pulling data out in a user-defined priority order. The concurrent priority queue is useful when a certain execution order is enforced by priority relationship between parallel tasks and/or data.
Unordered set with concurrent operations	New thread-safe container to store and access user objects using a hash key. Implements a concurrent variation of a standard C++ class <code>std::unordered_set</code> .
Generic GCC* Atomics Support	Greater library portability which enables the user to develop Intel® TBB-based solutions on a broader range of platforms. The user now has a choice to use built-in GCC* atomics routines instead of supplied platform-specific ones.
New Examples	New examples demonstrate usage of major new features: <ul style="list-style-type: none"> - Shortest path (<code>concurrent_priority_queue</code>) - Dining philosophers, binpack (flow graph) - Mandelbrot fractal (task priority)

Purchase Options: Language Specific Suites

Several suites are available combining the tools to build, verify and tune your application. The products covered in this product brief are highlighted in green. Single or multi-user licenses and volume, academic, and student discounts are available.

Suites >>		Intel® Parallel Studio XE	Intel® C++ Studio XE	Intel® Fortran Studio XE	Intel® Composer XE	Intel® C++ Composer XE	Intel® Fortran Composer XE	Intel® Cluster Studio XE	Intel® Cluster Studio
Components	Intel® C / C++ Compiler	●	●		●	●		●	●
	Intel® Fortran Compiler	●		●	●		●	●	●
	Intel® Integrated Performance Primitives ³	●	●		●	●		●	●
	Intel® Math Kernel Library ³	●	●	●	●	●	●	●	●
	Intel® Cilk™ Plus	●	●		●	●		●	●
	Intel® Threading Building Blocks	●	●		●	●		●	●
	Intel® Inspector XE	●	●	●				●	
	Intel® VTune™ Amplifier XE	●	●	●				●	
	Static Security Analysis	●	●	●				●	
	Intel® MPI Library							●	●
	Intel® Trace Analyzer & Collector							●	●
	Rogue Wave IMSL* Library ²						●		
Operating System ¹	W, L	W, L	W, L	W, L	W, L, M	W, L, M	W, L	W, L	

Note: (1)¹ Operating System: W=Windows, L= Linux, M= Mac OS* X. (2)² Available in Intel® Visual Fortran Composer XE for Windows with IMSL* (3)³ Not available individually on Mac OS X, it is included in Intel® C++ & Fortran Composer XE suites for Mac OS X

Selecting the right Intel® TBB license

Intel® TBB is available via multiple licensing options:

1. Commercial Binary Distribution: for customers who may require commercial support services. Distributions are validated and officially supported for a variety of select hardware, software, operating systems, and compilers.
2. Open Source: use if you are familiar with the restrictions for using open source software. Allows support for additional OSs and hardware platforms. Both source and binary forms are available for download from <http://threadingbuildingblocks.org>.
3. Academic: attractive pricing for academic, student and classroom usage.
4. Custom: if you require the ability to modify or distribute the commercial source code of Intel® TBB, contact your Intel representative for more information.

Technical Specifications	
Processor Support	Validated for use with multiple generations of Intel and compatible processors including but not limited to: Intel® Xeon™ Processor, Intel® Core™ processor family and Intel® Atom™ processor family.
Operating Systems	Use the same API for application development on multiple operating systems: Windows*, Linux* and MAC OS*.
Development Tools and Environments	Compatible with compilers from vendors that follow platform standards (e.g., Microsoft*, GCC, Intel). Can be integrated with Microsoft Visual Studio* (2005, 2008, and 2010).
Programming Languages	Natively supports C++ development; cross language usage examples provided for C#/NET.
System Requirements	Refer to www.intel.com/software/products/systemrequirements/ for details on hardware and software requirements.
Support	All product updates, Intel® Premier Support services and Intel® Support Forums are included for one year. Intel Premier Support gives you confidential support, technical notes, application notes, and the latest documentation. Join the Intel® Support Forums community to learn, contribute, or just browse! http://www.intel.com/software/products/support/
Community	Share experiences with other users of Intel® TBB and other parallel programming tools at the Intel moderated forum: http://software.intel.com/en-us/forums/ .

Download a trial version today
www.intel.com/software/products/eval



Order the [Intel® Threading Building Blocks](http://amazon.com) book online at amazon.com.

Optimization Notice

Notice revision #20110804

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.