UTILIZE CAPABILITIES WITHIN INTEL® XEON® PROCESSORS

A Concise Guide to Parallel Programming Tools for Intel® Xeon® Processors

WHITE PAPER

Q: How do I pick the right programming models and tools that boost application performance?

Introduction

As microprocessors transition from clock speed as the primary vehicle for performance gains to features such as multiple cores, wider vectors and increasing vector instruction sets; it is increasingly important for software developers to optimize their applications to fully utilize the inherent capabilities of the hardware. However, developers often don't realize specialized tools are now available to help them fully utilize these capabilities.

This paper explains the available hardware features, lists options for tools, describes each tool in more detail, and asks frank questions to help developers determine the right tool for their application. If you're a programmer working in a C, C++ or Fortran environment and are willing to make a few code changes, read on to learn how companies large and small are experiencing outstanding improvements in their applications performance - by using Intel's software tools.

Reasons to embrace parallelism

There are varying reasons to adopt parallelism, and it's important to understand your motivation and expectations for doing so. In our experience, organizations adopt parallelism for one or more of the following reasons:

- Customers complaining about application performance
- Applications where deriving the solution is time critical (e.g., predicting tomorrow's weather can't take a week)
- Competitors are implementing parallelism
- Power savings by doing the same amount of work via efficiently threaded code
- Allow offering new capabilities for your application (e.g. add security to an application with minimal impact to overall performance by offloading AES encryption to the CPU)

Performance oriented features in Intel® Xeon® Processors

Intel® Xeon® processors contain several inherent features that allow users to significantly enhance the performance of their application. However these features need programming expertise and the use of industry or Intel provided tools to fully utilize their benefits.

CPU Capability	Benefit to Applications
Intel® Smart Cache	Take advantage of fast CPU memory by reducing cache misses
Intel® Hyper-Threading Technology	Multi-threaded code executing across virtual cores increases performance
Intel® multi-core technology	Multi-threaded code executing across physical cores increases performance
Intel® SSE including Intel® AVX	Vectorized data runs in parallel within each core for increased performance
Intel® AES-NI	Protection using hardware accelerated security features
Performance Monitoring Unit (PMU)	PMU enables finding performance issues like cache misses

Available Tools

Intel offers developers with a wide choice of programming tools and models to embrace parallelism.

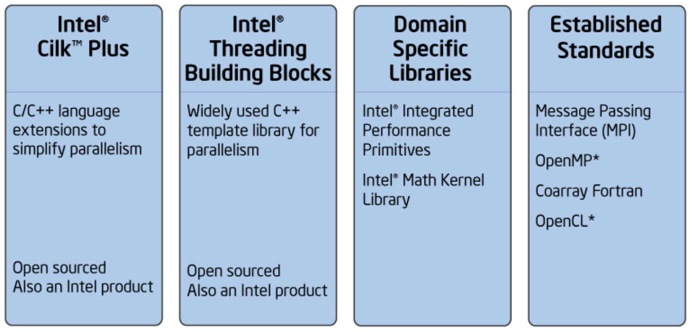
Programming Tools

Analysis tools quickly pinpoint problem areas in source code by looking for errors and security vulnerabilities, and provide insights to help you make decisions. Libraries provide pre-defined functions that can easily be added to your code to more efficiently use performance oriented features in Intel hardware. Compilers offer optimization features and multi-threading capabilities. Cluster Tools help analyze and optimize performance of parallel applications when developing for clusters using Message Passing Interface (MPI). Our bundled suites simplify the acquisition process by including most or all of these tools in a single installable package.

Category	Intel product	Supported Language(s)
Analysis Tools	Intel® Parallel Advisor	C, C++
	Intel® VTune™ Amplifier XE	C, C++, Fortran, C#
	Intel® Inspector XE	C, C++, Fortran
Libraries	Intel® Math Kernel Library (Intel® MKL)	C, C++, Fortran
	Intel® Integrated Performance Primitives (Intel® IPP)	C, C++. Usage examples for C#, Java
	Intel® Threading Building Blocks (Intel® TBB)	C++ template library
Compilers	Intel® Composer XE	C, C++, Fortran
Cluster Tools	Intel® MPI Library	C, C++, Fortran
	Intel® Trace Analyzer and Collector	C, C++, Fortran
Bundled Suites	Intel® Parallel Studio XE	C, C++, Fortran
	Intel® Cluster Studio XE	C, C++, Fortran

Programming Models

Intel also offers a common set of parallel programming models that can be used both on multi- and many-core processors. Each model allows developers to introduce or improve parallelism in their applications. They range from those designed with high level abstraction and ease of use in mind (such as Intel® Cilk™ Plus and Intel® TBB), highly optimized and extensively threaded libraries designed for drop-in productivity (Intel® IPP, Intel® MKL) and tools implementing established standards that help scale across computing environments (MPI, OpenMP*, Coarray Fortran and OpenCL*). The table below shows the multiple choices available.


Intel® Cilk™ Plus
C/C++ language extensions to simplify parallelism
Open sourced
Also an Intel product
Intel® Threading Building Blocks
Widely used C++ template library for parallelism
Open sourced
Also an Intel product
Domain Specific Libraries
Intel® Integrated Performance Primitives
Intel® Math Kernel Library
Established Standards
Message Passing Interface (MPI)
OpenMP*
Coarray Fortran
OpenCL*

Family of Programming Models: offering developers a wide variety of tools.

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

Determine if optimizing applications for performance is right for you

Selecting the right feature(s) to utilize and then matching the right tool to unlock its potential depends on the specific application and environment. Our consulting engineers typically ask the following questions to determine whether parallelism is right for an organization, as well as suggesting what tools would be appropriate.

Question	Response
Are you a software developer with access to source code?	If yes, continue
Are you willing to spend effort to improve performance?	If yes, continue
Have you analyzed your code for performance?	• Start by profiling your code to determine where it's spending time, wasting time and waiting too long. Intel® VTune™ Amplifier XE uses the on-chip Performance Monitoring Unit on Intel® Xeon® processors for hardware event sampling. E.g., identifying non-optimal use of on-board cache (Intel Smart Cache Technology). • Intel® Parallel Advisor: Provides step by step help to identify and experiment with adding parallelism • Intel® Inspector XE: Can find both memory and threading errors that lead to parallelism problems, including difficult to find errors like deadlocks and race conditions. It ensures your code is correctly multi-threaded to allow execution across both virtual (Intel Hyper Threading Technology) and physical (Intel multi-core technology) cores.
Did you find hotspots in a function?	Replace slow library functions with highly tuned and parallelized versions that are available at low cost and without royalties. If a particular function (ex. BLAS, FFT, matrix multiply) is a hotspot, utilize our libraries. They contain <u>thousands</u> of pre-defined, highly optimized functions that can be dropped-in to your code, and <u>automatically scale</u> with future Intel CPU features without requiring a re-compile: • Intel® Math Kernel Library: functions for science, engineering and financial applications. e.g., By using Intel AVX optimizations, Intel MKL provides up to 90% performance improvement for SMP LINPACK on Intel Xeon processors) • Intel® Integrated Performance Primitives: functions for images & video, communications and signal processing, data processing. (e.g., By using the AES-NI hardware acceleration in Intel Xeon processors, Intel IPP provides up to 145% performance improvement over OpenSSL)
Did you find hotspots in a module?	If you find a hotspot within one module, you can recompile that portion with the Intel compiler, while continuing to use your existing compiler for the rest. Also try auto-vectorization and other automated features in the compiler.
Are you interested in additional performance?	Explore our entire toolkit including programming models listed in the table that follows.

Note: Applications that are floating point intensive are particularly well suited for acceleration via our tools

Determine the right programming model for your application and environment

Let's take a look at compatibility.

Question	Response
What programming languages do you use?	**C** – Intel Cilk™ Plus, Intel IPP, Intel MKL, possibly Intel OpenCL* SDK **C++** - Either Cilk™ Plus, Intel TBB, Intel IPP, Intel MKL **Fortran** - Choose from Coarray Fortran, Intel MPI, Intel MKL, Intel IPP (via cross language calls) **C#** - some API support from Intel MKL and Intel IPP. **Scripting Languages (Python*, PERL*)** - limited options available via Intel MKL and Intel IPP
What OS support is needed?	**Windows*** – comprehensive support **Linux*** – comprehensive support **Mac OS* X** – support available via C++ Composer XE and Fortran Composer XE (Cilk™ Plus, Intel TBB, Intel IPP, Intel MKL, compilers, OpenMP*, Coarray Fortran).

Note: All tools provide comprehensive support for Intel Architecture and compatibles.

Selecting a programming model:

Question	Response
How willing are you to learn a new programming model? How tolerant are you of changes to your code?	Low willingness and tolerance– Use Intel IPP and Intel MKL libraries. Explore the compiler's auto-vectorization features. Somewhat willing and tolerant – Consider Intel Cilk™ Plus, Intel OpenMP*, Coarray Fortran in addition to libraries Very willing and tolerant – Consider Intel TBB, Intel MPI as they provide maximum flexibility and performance
Are you using shared memory, cluster or vector programming?	Programming for a cluster – use MPI for applications using distributed memory. Use OpenMP*, Intel Cilk™ Plus, Intel TBB for nodes under MPI. Programming for shared memory – Use OpenMP*, Intel Cilk™ Plus, Intel TBB. Programming for vectors (SSE/AVX etc) – Use Intel Cilk™ Plus, Fortran90 array notation.
Is CPU + GPU hybrid programming important?	If yes, consider using OpenCL*.

Programming Tools and Models: Brief Descriptions

Intel® Parallel Studio XE (bundled suite): A parallel software development suite that combines Intel's industry-leading C/C++ compiler and Fortran compiler; performance and parallel libraries; error checking, code robustness, and performance profiling tools into a single suite offering. This helps boost application performance and increase the code quality, security, and reliability needed by high-performance computing and enterprise applications. At the same time, the suite eases the procurement of all the necessary tools for high performance, and simplifies the transition from multicore to many-core processors in the future by using a common set of tools. Learn more at software.intel.com/en-us/articles/intel-parallel-studio-xe/

Intel® Cluster Studio XE (bundled suite): adds our cluster tools to Intel Parallel Studio XE. It is the first tool suite that enables maximum performance, reliability, and scalability for the development and analysis of shared, distributed and hybrid memory C++/Fortran applications on Intel® IA-32 and Intel® 64 architecture based Windows* and Linux* platforms. Learn more at software.intel.com/en-us/articles/intel-cluster-studio-xe/

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

Intel® Parallel Advisor (analysis): guides developers to add or improve parallelism to existing C/C++ programs. Developers can use it to identify the most time consuming serial code regions, insert annotations to experiment with adding parallelism, check the suitability of the proposed changes, and check for problems that would prevent the application from working correctly when parallelized. The tool is currently available on Microsoft Windows*. Learn more at: software.intel.com/en-us/articles/intel-parallel-advisor/

Intel® Composer XE (compiler and library): is the name for the product bundle that includes Intel's C++ Compiler, Intel Fortran Compiler, Intel Math Kernel Library, Intel Integrated Performance Primitives and Intel Threading Building Blocks. It allows C/C++ and Fortran developers to develop and maintain high-performance and enterprise applications on the latest Intel® Architecture processors. Using this product delivers improvements over and above the best optimizing C++ and Fortran compilers in the market. A simple recompile with Intel Composer XE can boost performance by 20 percent or more. Performance improvements are derived from optimizations in memory, auto-parallelization and vectorization. Learn more at software.intel.com/en-us/articles/intel-composer-xe/

Intel® VTune Amplifier XE (analysis tool): is a threading and performance profiling software for C/C++, Fortran and C# developers who need to understand serial and parallel behavior to improve performance and scalability. As a software performance analyzer for applications on Windows and Linux, Intel VTune Amplifier XE removes the guesswork by providing quick access to scaling information for faster and improved decision making. Fine-tune for optimal performance, ensure cores are fully exploited and new processor capabilities are supported to the fullest. The software features a number of new pre-defined performance profiling experiments for quickly getting detailed profiling information without having to know microarchitectural details. After profiling, analysis features such as timeline, filtering and frame analysis turn data into actionable information. Learn more at www.intel.com/software/products/vtune

Intel® Inspector XE (analysis tool): helps improve application reliability by detecting memory and threading errors. The tool is designed for C, C++, C# and Fortran developers building software on Windows* and Linux* systems. The Memory analysis tool can detect memory leaks and memory corruption early in the development cycle. The Thread analysis tool and debugger finds threads that have problematic interactions, and identifies data races and deadlocks. It also finds intermittent and non-deterministic errors, even when the error causing timing scenario does not happen. Using this tool early in the development cycle can detect and resolve security issues that would be far more expensive to resolve once the software has been deployed. Learn more at software.intel.com/en-us/articles/intel-inspector-xe/

Intel® MPI Library (cluster library): increases application performance on Intel® architecture-based clusters by implementing the high performance Message Passing Interface -2 (MPI-2) specification on multiple fabrics. It allows for changes or upgrades to new interconnects without requiring major changes to the software or operating environment. Use this high-performance message-passing interface library to develop applications that can run on multiple cluster fabric interconnects chosen by the user at runtime. Intel also provides a free runtime environment kit for products developed with the Intel MPI library. Get best-in-class performance for enterprise, divisional, departmental, and workgroup high performance computing. Learn more at software.intel.com/en-us/articles/intel-mpi-library/

Intel® Trace Analyzer and Collector (cluster analysis): is a powerful tool for understanding MPI application correctness and behavior. It includes a low-overhead tracing library that performs event-based tracing in applications. You can analyze the collected trace data for performance hotspots and bottlenecks. The product is completely thread safe and integrates with C/C++, FORTRAN and multithreaded processes with and without MPI. It supports binary instrumentation and fail-safe mode. Additionally it can check for MPI programming and system errors. The Intel® Trace Analyzer provides a convenient way to monitor application activities gathered by the Intel Trace Collector through graphical displays. You can view the desired level of detail, quickly identify performance hotspots and bottlenecks, and analyze their causes. Bundled together, ITAC provides optimized analysis and visualization capabilities. Together they offer fast graphical rendering of complex profiling data and they easily scale up to hundreds of processes. The tool is available on Linux* and Microsoft Windows*. Learn more at software.intel.com/en-us/articles/intel-trace-analyzer/

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

Programming Models

Libraries (Intel® Math Kernel Library and Intel® Integrated Performance Primitives): Libraries provide an important abstract parallel programming method that needs to be considered before jumping into programming. Library implementations for algorithms including BLAS, video or audio encoders and decoders, FFT, solvers and sorters, are important to consider. Intel's libraries offer advanced implementations of many algorithms that are highly tuned to utilize SSE and AVX instruction sets, multicore and many-core processors. A single source code can get these benefits by a single call into a routine in one of the libraries. MKL offers the standard interfaces in Fortran, and supports the new industry standard for C interfaces to LAPACK that Intel helped create. Standards combined with Intel's relentless pursuit of high performance in their libraries, make libraries an easy choice to utilize as the first choice in parallel programming. Learn more at www.intel.com/software/products/mkl/ and www.intel.com/software/products/ipp/

Matrix Multiply in Fortran using Intel® Math Kernel Library

```
call
DGEMM(transa,transb,m,n,k,alpha,a,lda,b,ldb,b
eta,c,ldc)
```

Intel® Cilk™ Plus: A quick, easy and reliable extension to C and C++. It provides tasking extensions and array notations for effective use of task, data and vector parallelism. These extensions trace their history back to M.I.T. research by Prof. Leiserson and the company he founded known as Cilk Arts. Intel® Cilk™ Plus should be promoted to C and C++ programmer looking to harness task, data and vector parallelism on processors and co-processors. Cilk Plus should be considered before Intel TBB in any project that would benefit from vector parallelism. Intel Cilk Plus and Intel TBB features can be mixed and used together. Therefore, Intel TBB users may use data and vector parallelism capabilities of Cilk Plus in a program using Intel TBB. Learn more at http://cilkplus.org

Parallel function invocation in C using Intel® Cilk™ Plus

```
cilk_for (int i=0; i<n; ++i) {
  Foo(a[i]);
}
```

DAXPY in array notation using Intel® Cilk™ Plus

```
a[0:n] += x * b[0:n];
```

Intel® Threading Building Blocks (Intel® TBB): The most popular abstraction for parallel programming in C++. Introduced by Intel in 2006, it became an open source project in 2007, and has enjoyed considerable adoption in the industry and has surpassed OpenMP in popularity. Companies that have publicly talked about using Intel TBB include Adobe, Autodesk and Dreamworks Animation to name a few. Intel TBB has been ported to numerous operating systems and processors. Intel TBB should be used by C++ developers looking to harness multicore or many-core parallelism. More information is available here http://threadingbuildingblocks.org

Parallel function invocation in C++ using Intel® Threading Building Blocks

```
parallel_for (0, n,
   [=](int i) { Foo(a[i]); }
);
```

OpenMP*: In 1996, the OpenMP* standard was proposed as a way for compilers to assist in the utilization of parallel hardware. After more than a decade, every major compiler for C, C++ and Fortran supports OpenMP*. OpenMP is especially well suited for the needs of Fortran programs and scientific programs written in C. Intel is a member of the OpenMP* work group and a leading vendor of implementations of OpenMP* and supporting tools. OpenMP* is applicable to multicore and many-core programming. Learn more at http://openmp.org

Summing vector elements in C using OpenMP

```
#pragma omp parallel for reduction(+: s)
for (int i = 0; i < n; i++) {
  s += x[i];
}
```

Message Passing Interface (MPI): For programmers utilizing a cluster, in which processors are connected by the ability to pass messages but not always the ability to share memory, the Message Passing Interface (MPI) is the most common programming method. In a cluster, communication continues to use MPI, as they do today, regardless of whether a node has many-core processors or not. The widely used Intel® MPI library offers both high performance and support for virtually all interconnects. The Intel® MPI library supports multicore and many-core processor based systems creating ranks on multicore and many-core processors in a fashion that is familiar and consistent with MPI programming today. Learn more at http://intel.com/go/mpi

MPI code in C for clusters

```
for (d=1; d<ntasks; d++) {
  rows = (d <= extra) ? avrow+1 : avrow;
  printf(" sending %d rows to task %d\n",
rows, dest);
  MPI_Send(&offset, 1, MPI_INT, d, mtype,
MPI_COMM_WORLD);
  MPI_Send(&rows, 1, MPI_INT, d, mtype,
MPI_COMM_WORLD);
  MPI_Send(&a[offset][0], rows*NCA,
MPI_DOUBLE, d, mtype, MPI_COMM_WORLD);
  MPI_Send(&b, NCA*NCB, MPI_DOUBLE, d,
mtype, MPI_COMM_WORLD);
  offset = offset + rows;
}
```

Coarray Fortran: The Fortran 2008 standard introduced the "co-array" extensions as an additional method to use Fortran as a robust and efficient parallel programming language. The Intel® Fortran Compiler supports parallel programming using coarrays as defined in the Fortran 2008 standard for both shared memory and distributed memory systems. Coarray Fortran uses a single-program, multi-data programming model (SPMD). Learn more at http://intel.com/software/products

Sum in Fortran, using co-array feature

```
REAL SUM[*]
CALL SYNC_ALL( WAIT=1 )
DO IMG= 2,NUM_IMAGES()
  IF (IMG==THIS_IMAGE()) THEN
    SUM = SUM + SUM[IMG-1]
  ENDIF
  CALL SYNC_ALL( WAIT=IMG )
ENDDO
```

OpenCL*: OpenCL* offers a close-to-the-hardware interface offering some important abstraction and substantial control coupled with wide industry interest and commitment. OpenCL* may require the most refactoring of any of the solutions covered in this whitepaper, specifically refactoring based on advanced knowledge of the underlying hardware. Results from refactoring work may be significant for multicore and many-core performance, and the resulting performance may or may not be possible without such refactoring. A goal of OpenCL* is to make an investment in refactoring productive when it is undertaken. Solutions other than OpenCL* offer methods to avoid the need for refactoring based on advanced knowledge of the underlying hardware. Learn more at http://intel.com/go/opencl

Per element multiply in C using OpenCL

```
kernel void
  dotprod(   global const float *a,
       global const float *b,
       global float *c) {
    int myid = get_global_id(0);
    c[myid] = a[myid] * b[myid];
}
```

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

Summary

Don't leave performance on the table. As Intel CPU's continually add new performance oriented features, source code must be vectorized and multi-threaded. This paper has shown how analysis tools can identify focus areas for performance, how slow functions can be swapped out with highly optimized functions in Intel's libraries, how a hotspot within a module can be re-compiled using our compilers, and how multiple programming models exist that can take you all the way.

Intel has been providing developers with programming tools to utilize our processors capabilities for over 25 years. Intel understands application needs vary widely, and we provide developers with a wide choice of tools and options to solve a variety of programming problems. Amongst those who have used our tools, it's common to report performance gains of 10%, 2x, even 20x. The tools are designed to complement your current environment, whether its Microsoft Visual Studio*, Eclipse*, Xcode, makefile or command-line builds, with either the Microsoft, Intel, or GCC compilers.

As products based on Intel® Many Integrated Core Architecture (Intel® MIC architecture) become available, its increasingly important to ensure your source code is ready to take full advantage of the available horsepower. The good news is the same standardized, high level x86 programming models that apply to multicore also apply to many-core. Once optimized for Intel Xeon processors, developers can reuse their existing code and programming expertise on Intel MIC architecture. We encourage you to learn more at: software.intel.com/parallel/, or post your question on a forum at software.intel.com/en-us/forums/threading-on-intel-parallel-architectures/.

Purchase Options: Language Specific Suites

Several suites are available combining the tools to build, verify and tune your application. Single or multi-user licenses and volume, academic, and student discounts are available.

	Suites >>	Intel® Parallel Studio XE	Intel® Cluster Studio XE	Intel® C++ Studio XE	Intel® Fortran Studio XE	Intel® Composer XE	Intel® C++ Composer XE	Intel® Fortran Composer XE
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	W, L, M	W, L, M

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

Evaluate a tool

Download a free evaluation copy of our tools. If you're still uncertain where to begin, we suggest:

- For bundled suites, try Intel Parallel Studio XE or Intel Cluster Studio XE
- Intel® VTune™ Amplifier XE performance analyzer
- Intel® Parallel Advisor for Windows*

Learning Tools

- Intel Learning Lab, collection of tutorials, white papers and more.
- Technical Presentations and Videos on Parallel Programming

Suggested Reading

- Intel Guide for Developing Multithreaded Applications
- Webinar – "The Key to Scaling Applications for Multicore"
- Intel® VTune™ Amplifier XE – a short overview movie and detailed step-by-step getting started tutorials.
- Article: Getting code ready for parallel execution (includes more detailed code examples)
- 774 page Manual: Intel 64 and IA-32 Architectures Reference Manual
- Intel® TBB book on Amazon.com