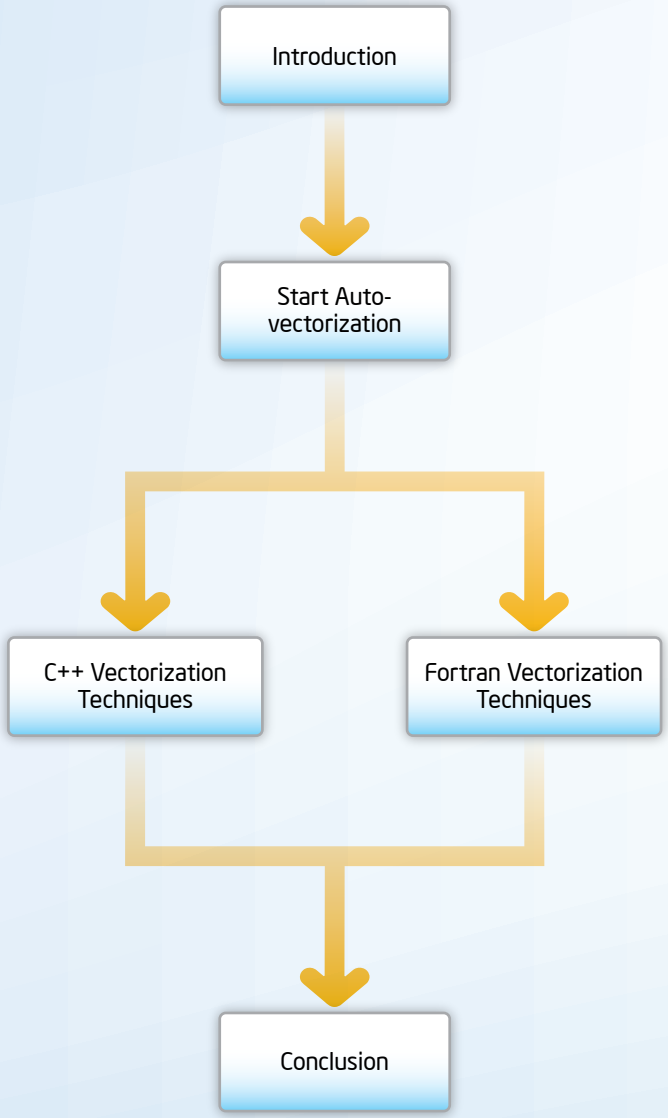
VECTORIZATION

CodeBook

intel®
Software

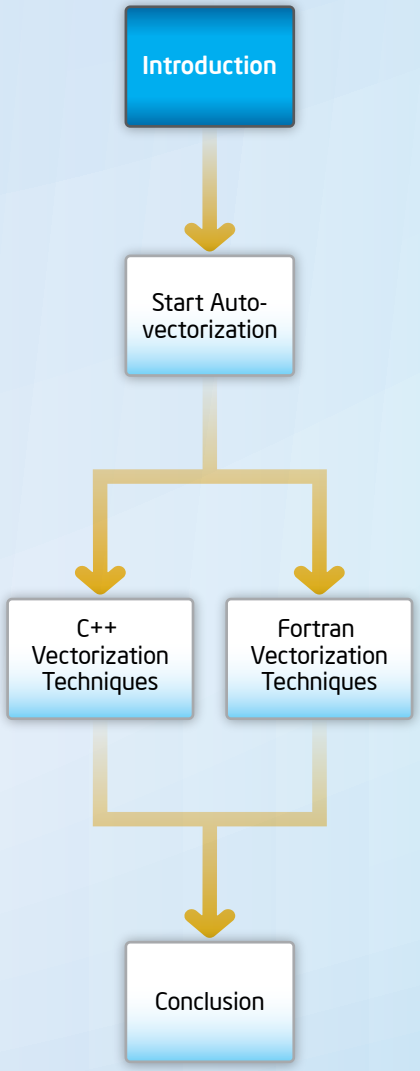


Welcome

This CodeBook focuses on vectorization techniques that can increase performance in both C++ and Fortran applications. In addition to explaining the concept of vectorization, we've gathered together the most relevant resources to help you address challenges specific to your software and development process.

"Vector instructions—also known as SIMD—offer parallelism that is incremental to multicore, also known as task, parallelism. Vector capabilities are a key form of hardware support for data parallelism. Vectorization can help software developers achieve immediate performance gains by using vector instructions. The right tools can offer vectorization in a way that also helps scale-forward for future architectures including processor support for SSE and AVX instructions, as well as Knights Corner vector instructions and future hardware vector innovations."

James Reinders
CHIEF SOFTWARE EVANGELIST
INTEL CORPORATION



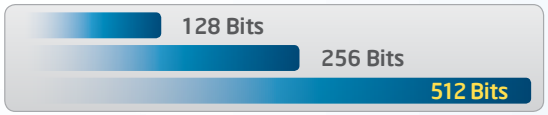
Introduction

Moore's Law correctly predicted that processor transistor density would increase year after year. This has enabled powerful hardware capabilities that can help increase performance. For example, instruction set extensions to support SIMD (single instruction, multiple data) parallelism. These instructions operate on a vector of data in parallel. The vector width, and therefore the number of elements that can be accessed in parallel, continues to expand as new processor technologies are introduced.

Applications need to be vectorized to take advantage of SIMD instructions and to utilize the expanded vector width. Here you will find simple, yet powerful vectorization techniques that can be used by just about any application developer using Intel® compilers and libraries.

More Cores

Wider Vectors



Why Vectorize

In computer science, vectorization is the process of converting an algorithm from operating on single pieces of data at one time to operating on multiple pieces of data at one time.

Scalar Implementation

Conducts an operation one pair of operands at a time

Vector Process

A single instruction can refer to a vector *(series of adjacent values)*

In effect, a vector process adds a form of parallelism to software in which one instruction or operation is applied to multiple pieces of data. When done on computing systems that support such actions, the benefit is **more efficient processing and improved application performance.**

So where does the vectorization speedup come from? Let's look at a sample code fragment, where a, b, and c are integer arrays:

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

If vectorization is not enabled, (e.g., because we used /Od, /O1 or /Qvec- compiler options), the compiler may do something like this:

Scalar Implementation

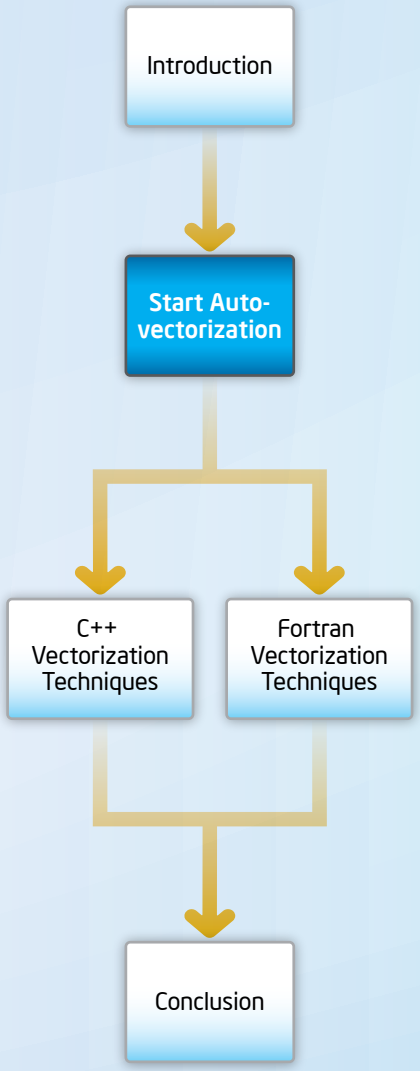
unused	unused	unused	A[0]
+	+	+	+
unused	unused	unused	B[0]
unused	unused	unused	C[0]

So we have a lot of unused space in our SIMD registers, which could hold three additional integers.

If vectorization is enabled, the compiler may use the additional space in the SIMD registers to perform four additions in a single instruction:

Vector Implementation

A[3]	A[2]	A[1]	A[0]
+	+	+	+
B[3]	B[2]	B[1]	B[0]
C[3]	C[2]	C[1]	C[0]



Start Auto-vectorization

Intel® C++ and Intel® Fortran Compilers implement SIMD by generating instructions from the Intel® Streaming SIMD Extensions (Intel® SSE) and Intel® Advanced Vector Extensions (Intel® AVX) on both IA-32 and Intel® 64 processors. Both compilers do auto-vectorization—generating Intel SIMD code to automatically vectorize parts of application software when certain conditions are met. Because no source code changes are required to use auto-vectorization, there is no impact on the portability of your application.

To take advantage of auto-vectorization, applications must be built at default optimization settings (-O2) or higher. No additional switch settings are needed. The compiler will automatically look for opportunities to execute multiple adjacent loop iterations in parallel using packed SIMD instructions. If one or more loops have been vectorized, diagnostics can also be enabled that will emit a remark to the build log that identifies the loop and says the "LOOP WAS VECTORIZED".

When you use Intel compilers on systems that use Intel® processors, you get 'free' performance improvements that will automatically take advantage of processing power as the Intel® architecture gets more parallel.

Get Started

6-step Process for Vectorizing with Intel® Software Products

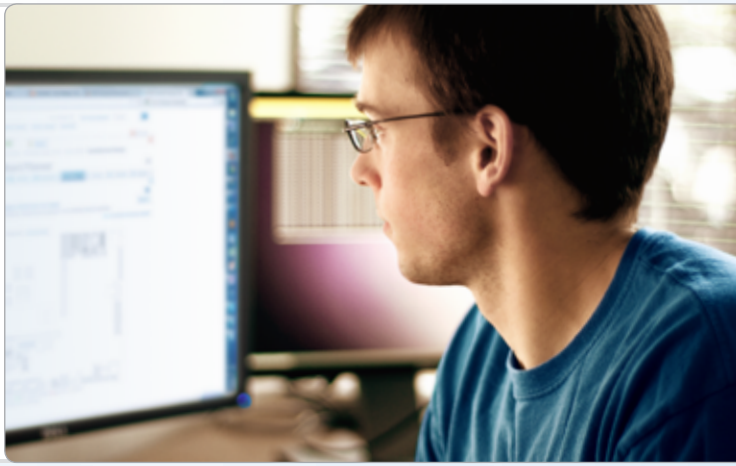
A Guide to Auto-vectorization with Intel® C++ Compilers

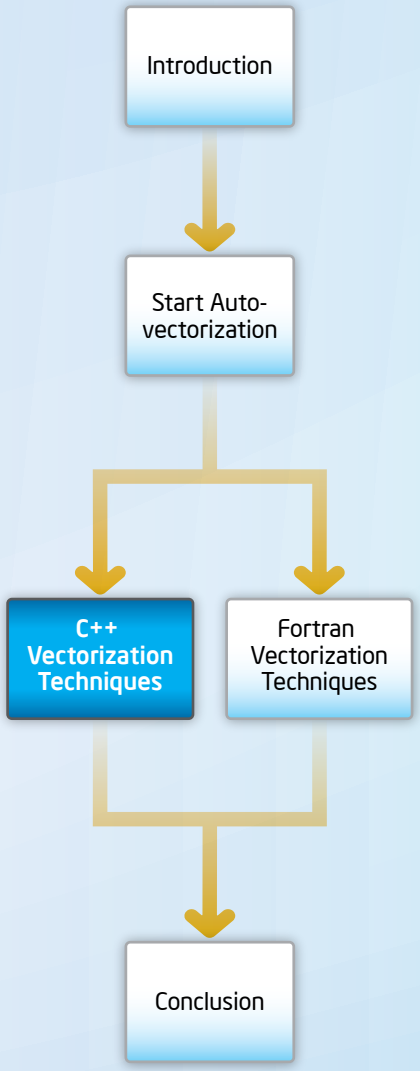
Intel® C++ Composer XE 2011 Getting Started Tutorials

Windows* | Linux* | Mac OS* X

Intel® Fortran Composer XE 2011 Getting Started Tutorials

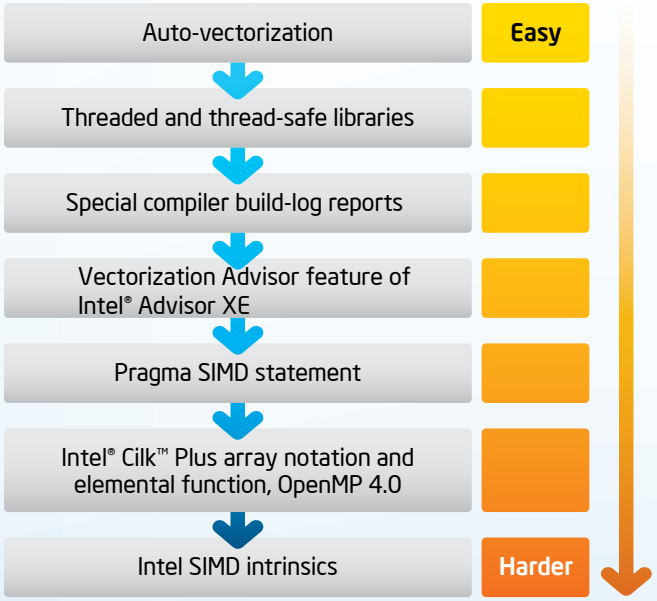
Windows* | Linux* | Mac OS* X





C++ Vectorization Techniques

Just about every C++ application developer can use the vectorization techniques listed here. The first are those that are the easiest to use. They require no changes to code. Next are libraries. For some applications, more advanced techniques may be needed to give information to the compiler so that code can be vectorized. The rest of the techniques focus on providing information to the compiler, including using compiler optimization reports and features such as Intel® Cilk™ Plus or OpenMP 4.0*.



Explore C++ Vectorization Techniques

Webinar

Future-Proof Your Application's Performance with Vectorization Technical Presentation

White Paper

An Introduction to Vectorization with the Intel® C++ Compiler

Articles

Vectorization Essentials

Vectorization: Writing C/C++ Code in VECTOR Format

Intel® Cilk™ Plus Overview. Overviews, Videos, Getting Started Guide, Documentation, White Papers, and a Link to the Community

Elemental Functions: Writing Data Parallel Code in C/C++ Using Intel® Cilk™ Plus

Requirements for Vectorizable Loops

Getting the most out of the Intel compiler with new optimization reports

Vectorization Advisor-Linux* OS

Vectorization Advisor-Windows* OS

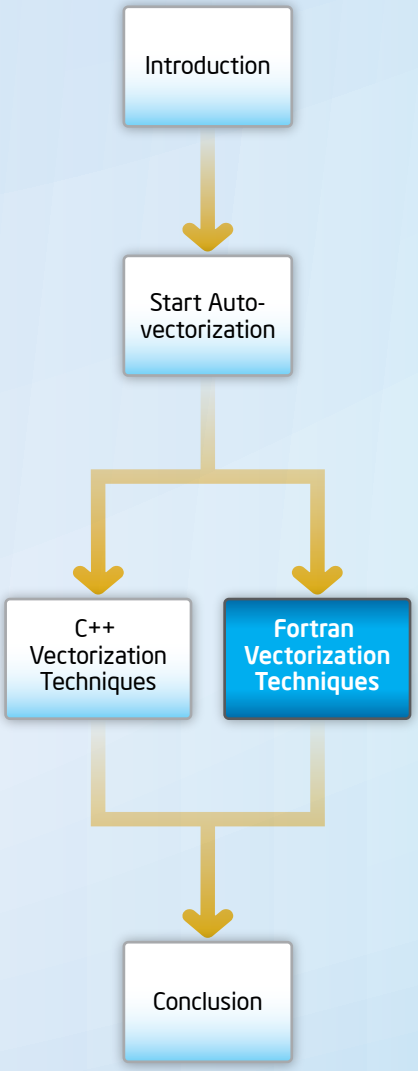
Documentation

Intel® C++ Composer XE Documentation

Guides

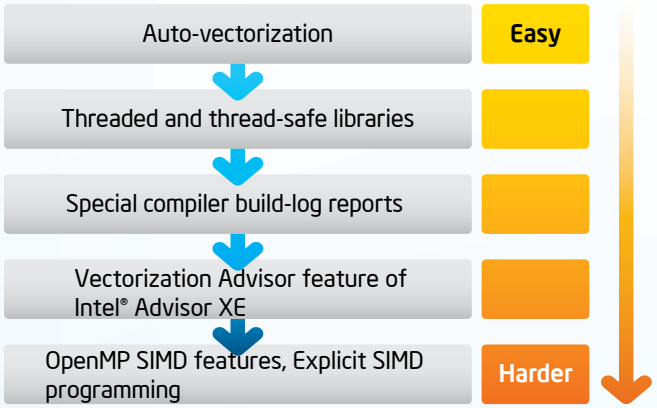
Vectorization Toolkit

Quick-Reference Guide to Optimization with Intel® Compiler Version 15 (for IA-32 Processors and Intel® 64 Processors)



Fortran Vectorization Techniques

Just about every Fortran application developer can use the vectorization techniques listed here. The first are those that are the easiest to use. They require no changes to code. Next are libraries, followed by capabilities that are built into the Intel® Fortran compiler, including vectorization advice to the programmer and guided auto-parallelism.



Explore Fortran Vectorization Techniques

Webinar

Future-Proof Your Application's Performance with Vectorization Technical Presentation

White Paper

An Introduction to Vectorization with the Intel® Fortran Compiler

Articles

Vectorization Toolkit. Six Steps to Increase Performance through Vectorization in Your Application

Vectorization Essentials

Requirements for Vectorizable Loops

Explicit Vector Programming in Fortran

Vectorization Advisor-Linux* OS

Vectorization Advisor-Windows* OS

Documentation

Intel® Fortran Composer XE Documentation

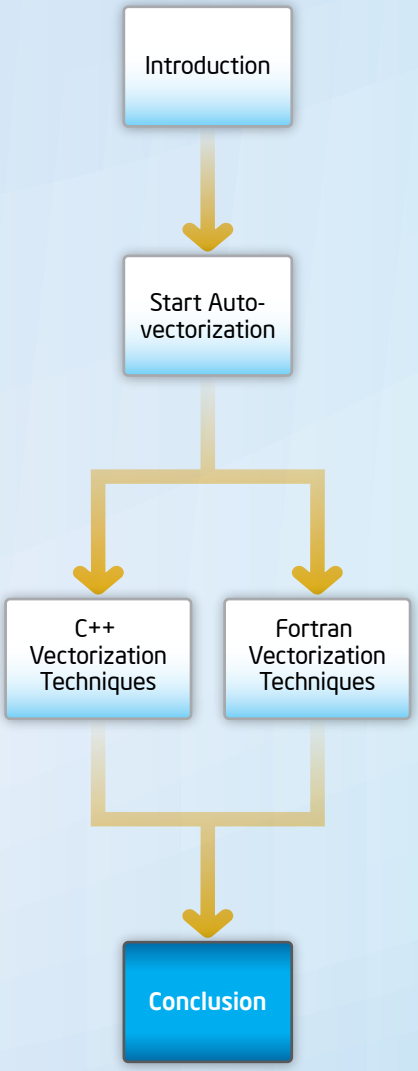
Guides

Quick-Reference Guide to Optimization with Intel® Compiler Version 15 (for IA-32 Processors and Intel® 64 Processors)

Blog

Software Blog by "Dr. Fortran"
with Steve Lionel

Intel Software



Conclusion

We hope you will see improved optimization and increased application performance from the vectorization techniques presented through the discussion and in-depth resources in this CodeBook.

Visit the Vectorization website for a comprehensive view of available resources.

Explore Other Paths to Performance

Intel® Software Development Tools

Go Parallel

Parallel Universe Magazine ›

Optimization Notice

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.

Notice revision #20110804