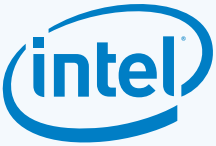


Quick-Reference Guide to Optimization with Intel® Compilers version 10.x

For IA-32 processors, Intel® 64[1] processors, and IA-64[2] processors.

Intel® Software Development Products

Application Performance

A Step-by-Step Approach to Application Tuning with Intel® Compilers

Before you begin performance tuning, you may want to check correctness of your application by building it without optimization using **/Od** (**-O0**).

1. Use the General Optimization Options (Windows* **/O1**, **/O2** or **/O3**; Linux* and Mac OS* **-O1**, **-O2**, or **-O3**) and determine which one works best for your application by measuring performance with each. Most users should start at **/O2** (**-O2**) (default) before trying more advanced optimizations. Next, try **/O3** (**-O3**) for loop-intensive applications, especially on IA-64-based systems.

2. Fine-tune performance to target systems based on IA-32 and Intel 64® with processor-specific options such as **/QxT** (**-xT**) for Intel® Core™2 processor family. For a complete list of recommended options for specific processors, see the table "Recommended Processor-Specific Optimization Options for IA-32 and Intel® 64 Architectures". For Dual-Core Intel® Itanium® 2 9000 Sequence processors, set **/G2-p9000** (**-mtune=itanium2-p9000**).

3. Use the Intel® VTune™ Performance Analyzer to help you identify performance "hotspots" so that you know which specific parts of your application could benefit from further tuning. The Intel® Compilers' optimization reports also help by showing where the compiler could benefit from your help.

4. Add in interprocedural optimization (IPO), **/Qipo** (**-ipo**) and/or profile-guided optimization (PGO), **/Qprof-gen** and **/Qprof-use** (**-prof-gen** and **-prof-use**), then measure performance again to determine whether your application benefits from one or both of them.

5. Optimize your application for multi-core, multi-processor, or Hyper-Threading Technology (HT Technology)-capable systems using the parallel performance options (**/Qparallel** (**-parallel**), **/Qopenmp** (**-openmp**)), or by using Intel® Performance Libraries, or the Intel® Threading Building Blocks.

6. Use Intel® Thread Profiler to help you understand the structure of your threaded applications and maximize their performance. Use Intel® Thread Checker to reduce the time to market for threaded applications by diagnosing threading errors and speeding up the development process. Both threading tools work with binary instrumentation. Using the Intel Compiler with source code instrumentation will give you more complete source code information.

Please consult the Compiler Documentation and the *Optimizing Applications with the Intel® C++ & Fortran Compilers* white paper for more details.

[1] Intel® 64 = Intel® Processors with Extended Memory 64 Technology [EM64T]
[2] IA-64 = Intel® Itanium® Processors

Included in this Guide:

General Optimization Options

Before you begin performance tuning, you may want to check correctness of your application by building it without optimization using **/Od** (**-O0**). Begin performance tuning with **/O1**, **/O2**, or **/O3** (**-O1**, **-O2**, or **-O3**). These are general optimization options that should be at the heart of any application tuning for all 32-bit and 64-bit Intel processors. Measure your performance before proceeding with more advanced options.

Parallel Performance

For systems with Hyper-Threading Technology , multi-core and/or multiple processors, Intel compilers support development of multi-threaded applications through two mechanisms, **/Qparallel** (**-parallel**) or **/Qopenmp** (**-openmp**).

If you are using Intel® Thread Profiler and Intel® Thread Checker to tune your threaded application, use **/Qtcheck** (**-tcheck**) to enable source instrumentation for Intel® Thread Checker and **Qtprofile** (**-tprofile**) to enable source instrumentation for Intel® Thread Profiler.

Recommended Processor-Specific Optimization Options for IA-32 and Intel® 64[1] Architectures

Use **/QxT** (**-xT** on Linux* and Mac OS*) for best performance on the Intel® Core™2 processor family, and **/QxP** (**-xP** on Linux*) on older Intel-based systems that support SSE3 instructions. We recommend **/QaxT /QxW** (**-axT -xW** on Linux*) for best performance on the Intel® Core™2 processor family, and good performance on other systems that support SSE2 including those from AMD. For best performance on non-Intel processors that support SSE3 instructions, we recommend using **/QxO** (**-xO**) in place of **/QxW** (**-xW**). For recommended options for older processors, see the table entitled "Recommended Optimization Options for Specific Intel® Processors".

These options allow you to tune performance for specific Intel processors. As with each previous step, measure the performance benefit of each option to guide your decisions. Use the Intel compilers' optimization reports to assist in determining whether you can provide more help to the compiler to resolve possible dependencies or aliases.

IA-64 (Intel® Itanium®) Processor-Specific Optimization Options

In general, using **/O3** (**-O3**), IPO and/or PGO, in conjunction with the optimization reports (described in the Fine-Tuning section of this document), to help resolve possible aliases and improve memory utilization provides the best performance for IA-64-based systems.

Interprocedural Optimization (IPO) and Profile-Guided Optimization (PGO) Options

IPO includes function-inlining to reduce function call overhead and expose more optimization opportunities. PGO provides runtime feedback to guide optimization decisions about data and code layout to improve instruction-cache efficiency, paging and branch prediction. However, IPO can increase code size. Be sure to measure your execution performance, compile time, and code size tradeoffs with these options. IPO is best used in conjunction with PGO to guide which functions to inline.

Floating-Point Arithmetic Options

The Intel® compilers provide options for enhancing the consistency or precision of floating-point results on all Intel® architectures, at some cost in performance. Refer to the Compiler Options section of the *Intel® C++ and Fortran Compiler Documentation* for detailed information on floating-point options.

Fine-Tuning (All Processors)

Once you have identified performance hot-spots, you may need to provide the compiler with more information to fine-tune specific functions. The optimization and vectorization reports may show places where loops could not be optimized fully due to pointer aliasing or memory-access overlaps, for example. The *Intel® C++ and Fortran Compiler Documentation* includes details on other #pragmas, directives, and intrinsics that can be used to control software-pipelining, loop unrolling, vectorization, and prefetching for further fine-tuning within your application code.

General Optimization Options

Windows*	Linux* Mac OS*	Comment
/Od	-O0	**No optimization.** Used during the early stages of application development and debugging. Use a higher setting when the application is working correctly.
/O1	-O1	**Optimize for size.** Omits optimizations that tend to increase object size. Creates the smallest optimized code in most cases. This option is useful in many large server/database applications where memory paging due to larger code size is an issue.
/O2	-O2	**Maximize speed.** Default setting. Creates faster code than **/O1** (**-O1**) in most cases.
/O3	-O3	Enables **/O2** (**-O2**) optimizations plus more aggressive loop and memory-access optimizations, such as scalar replacement, loop unrolling, code replication to eliminate branches, loop blocking to allow more efficient use of cache and, on IA-64-based systems only, additional data prefetching. The **/O3** (**-O3**) option is particularly recommended for applications that have loops that heavily use floating-point calculations or process large data sets. These aggressive optimizations may occasionally slow down other types of applications compared to **/O2** (**-O2**).
/Zi	-g	Generates debug information for use with any of the common platform debuggers. This option turns off **/O2** (**-O2**) and makes **/Od** (**-O0**) the default unless **/O2** (**-O2**) (or another **O** option) is specified.
/debug:full	-debug full	Allows easier debugging of optimized code by adding full symbol information, including the local symbol table information, regardless of the optimization level. This may result in minor performance degradation. If this option is specified for an application that makes calls to C library routines that will be debugged, the option /dbglibs must also be specified to link the appropriate C debug library.

Parallel Performance

Windows*	Linux* Mac OS*	Comment
/Qopenmp	-openmp	Enables the parallelizer to generate multi-threaded code based on the OpenMP* directives.
/Qopenmp-report {0|1|2}	-openmp-report {0|1|2}	Controls the OpenMP parallelizer's diagnostic levels. The default is **/Qopenmp-report1.**
/Qparallel	-parallel	Detects simply structured loops capable of being executed safely in parallel and automatically generates multi-threaded code for these loops.
/Qpar-report {0|1|2|3}	-par-report {0|1|2|3}	Controls the auto-parallelizer's diagnostic levels as follows: **0** – Displays no diagnostic information. **1** – Indicates loops successfully parallelized (default). **2** – Adds information on loops that were not parallelized.. **3** – Adds information about any proven or assumed dependencies inhibiting auto-parallelization (reasons for not parallelizing).
/Qpar-threshold[n]	-par-threshold[n]	Sets a threshold for the auto-parallelization of loops based on the probability of profitable execution of the loop in parallel, **n=0** to **100**. Default: **n=100**. **0** – Parallelize loops regardless of computation work volume. **100** – Parallelize loops only if profitable parallel execution is almost certain. Must be used in conjunction with **/Qparallel** (**-parallel**).
/Qtprofile	-tprofile	Enables source instrumentation to capture information about the structure of threaded applications for use in tuning them to maximize performance. This option creates a binary which will generate results that can be viewed with Intel® Thread Profiler.
/Qtcheck	-tcheck	Enables source instrumentation to capture information for diagnosing threading errors in threaded applications. This option creates a binary which will generate diagnostics that can be viewed with Intel® Thread Checker.
/Qopt-mem-bandwidth<n> (IA-64 only)	-opt-mem-bandwidth<n> (IA-64 only)	Restricts certain optimizations that may increase memory bandwidth requirements. **/Qopt-mem-bandwidth0** (**-opt-mem-bandwidth0**) - no restriction (default for serial compilation) **/Qopt-mem-bandwidth1** (**-opt-mem-bandwidth1**) – restricts optimizations for loops in OpenMP parallel regions (default with **/Qparallel** (**-parallel**) or **/Qopenmp** (**-openmp**)) **/Qopt-mem-bandwidth2** (**-opt-mem-bandwidth2**) - restricts optimizations for all loops. May be useful for MPI or other parallel applications. **Note:** For Mac OS*, this option is not supported.

Recommended Processor-Specific Optimization Options for IA-32 and Intel® 64[1] Architectures

Windows*	Linux* Mac OS*	Comment
/Qx {S| T| P| O| N| W| K}	-x {S| T| P| O| N| W| K}	Processor-specific targeting. Generates specialized code for the indicated processor and enables vectorization. The executable should only be run on the targeted compatible processors. **S** – May generate SSE4, SSSE3, SSE3, SSE2, and SSE instructions for Intel processors. Optimizes for a future Intel® processor that supports SSE4 Vectorizing Compiler and Media Accelerators. **T** – May Generate SSSE3, SSE3, SSE2, and SSE instructions for Intel processors. Optimizes for the Intel® Core™2 Duo Processor family, Quad-Core Intel® Xeon® processors, and Dual-Core Intel® Xeon® 5300, 5100 and 3000 series processors. **P** – May Generate SSE3, SSE2, and SSE instructions for Intel processors. Optimizes for Intel® Core™ microarchitecture, Intel® Pentium® 4 processors with SSE3, Intel® Xeon® processors with SSE3, Intel® Pentium® dual-core processor T2060, Intel® Pentium® Extreme Edition processor, and Intel® Pentium® D processor. Performs optimizations not enabled with **/QxO** (**-xO**). **N** – May Generate SSE2 and SSE instructions for Intel processors. Optimizes for the Intel® Pentium® 4 processor, Intel® Xeon® processor with SSE2, and Intel® Pentium® M processor. Performs optimizations not enabled with **/QxW** (**-xW**). **W** – May Generate SSE2 and SSE instructions. Optimizes for the Intel® Pentium® 4 processor and Intel Xeon® processor with SSE2. Code path may execute on Intel® and Non-Intel Processors which support SSE2 and SSE*. **K** – May Generate SSE instructions. Optimizes for the Intel® Pentium® III processor and Intel® Pentium® III Xeon® processor. Code path may execute on Intel® and Non-Intel Processors which support SSE*. **Note:** On Mac OS*, options **O**, **N**, **W** and **K** are not supported. For Mac OS* systems using IA-32 architecture, **-xP** is default. For Mac OS* systems using Intel® 64 architecture, **-xT** is default.
/Qax {S| T| P| N| W| K}	-ax {S| T| P| N| W| K}	Automatic Processor Dispatch. Generates specialized code and enables vectorization for the indicated processors while also generating non-processor-specific code. You can use more than one letter to tune for multiple processors in the same executable. For example, for best performance on the Intel® Core™2 Duo Processor family, Quad-Core Intel® Xeon® processors, and Dual-Core Intel® Xeon® 5300, 5100 and 3000 series processors while also running well on an AMD processor that supports only SSE2, use **/QaxT /QxW** (**-axT -xW** on Linux*) to generate a binary that will utilize SSSE3 and be tuned for non-SSSE3 x86-64 processors via CPU dispatch. In this example, the **/QaxT /QxW** (**-axT -xW** on Linux*) combination will produce binaries with two code paths, using the process-dispatch technology. One code path will take full advantage of the Intel® Core™2 Duo Processor family, Quad-Core Intel® Xeon® processors, and Dual-Core Intel® Xeon® 5300, 5100 and 3000 series processors. The other code path also takes advantage of the capabilities provided by the Intel processor and will also run on processors that do not support SSE3. At runtime, the application automatically identifies the Intel processor on which it is running and selects the appropriate implementation, either specialized or generic. **Notes**: Option **O** is not supported for **/Qax** (**-ax**). On Mac OS*, options **P**, **N**, **W** and **K** are not supported.
/Qvec-report [n]	-vec-report [n]	n = **0**: no information n = **1**: indicates vectorized loops (default) n = **2**: indicates vectorized and non-vectorized loops n = **3**: indicates vectorized loops and explains why non-vectorized loops were not vectorized

* The *option* values **O**, **W**, and **K** produce binaries that should run on processors not made by Intel such as AMD processors that implement the same capabilities as the corresponding Intel processors. **P** and **N** option values perform additional optimizations that are not enabled with option values **O** and **W**.

IA-64² Processor-Specific Optimization Options

Windows*	Linux*	Comment
/G2	-mtune=itanium2	Targets optimization for the Intel Itanium 2 processor. Generated code is also compatible with the older IA-64 processor (default).
/G2-p9000	-mtune=itanium2-p9000	Targets optimizations for Dual-Core Intel® Itanium® 2 9000 Sequence processors. Generated code is also compatible with all IA-64 processors, unless the user program calls intrinsic functions specific to the Dual-Core Intel Itanium 2 9000 Sequence processors.
/QIPF-fma[-]	-IPF-fma[-]	Enables [disables] the combining of floating-point multiply operations and add/subtract operations. (Enabled by default)
/Qivdep-parallel	-ivdep-parallel	Indicates that there is no forward or backward loop-carried memory dependency in the loop where the IVDEP directive is specified. Typically used in conjunction with **/Qparallel** (**-parallel**).
/Qprefetch[-]	-prefetch[-]	Enables or disables prefetch insertion.

Interprocedural Optimization (IPO) and Profile-Guided Optimization (PGO) Options

Windows*	Linux* Mac OS*	Comment
/Qip	-ip	Single file optimization. Interprocedural optimizations, including selective inlining, within the current source file. **Caution**: For large files, this option may sometimes significantly increase compile time and code size.
/Qipo[value]	-ipo[value]	Permits inlining and other interprocedural optimizations among multiple source files. The optional **value** argument controls the maximum number of link-time compilations (or number of object files) spawned. Default for ***value*** is 0 (the compiler chooses). **Caution**: This option can in some cases significantly increase compile time and code size.
/Qipo-jobs[n]	-ipo-jobs[n]	Specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO). The default is 1 job.
/Ob2	-finline-functions -finline-level=2	This option enables function inlining within the current source file at the compiler's discretion. This option is enabled by default at **/O2** and **/O3** (**-O2** and **-O3**). **Caution**: For large files, this option may sometimes significantly increase compile time and code size. It can be disabled by **/Ob0** (**-fno-inline-functions** on Linux* and Mac OS*).
/Qinline-factor=n	-finline-factor=n	This option scales the total and maximum sizes of functions that can be inlined. The default value of **n** is 100, i.e., 100% or a scale factor of one.
/Qprof-gen	-prof-gen	Instruments a program for profiling.
/Qprof-use	-prof-use	Enables the use of profiling information during optimization.
/Qprof-dir *dir*	-prof-dir *dir*	Specifies a directory for the profiling output files, *.dyn and *.dpi.

Floating-Point Arithmetic Optimizations

Windows*	Linux* Mac OS*	Comment
/fp:name	-fp-model name	This method of controlling the consistency of floating point results by restricting certain optimizations is recommended in preference to the **/Op** (**-mp**) and **/Qprec** (**-mp1**) switches. The possible values of **name** are: **precise** – Enables only value-safe optimizations on floating point code. **double/extended/source** – Implies **precise** and causes intermediates to be computed in double, extended or source precision. The **double** and **extended** options are not available for Intel® Fortran. **fast=[1|2]** – Allows more aggressive optimizations at a slight cost in accuracy or consistency. (**fast=1** is the default) **except** – Enables floating point exception semantics. **strict** – Strictest mode of operation, enables both the **precise** and **except** options and disables fma contractions. **Recommendation: /fp:source** (**-fp-model source**) is the recommended form for the majority of situations on IA-64 processors, on processors supporting Intel® 64, and on IA-32 when SSE are enabled with **/QxW** (**-xW**) or higher when enhanced floating point consistency and reproducibility are needed.
/Qfp-speculation *mode*	-fp-speculation *mode*	Enables floating-point speculations with one of the following modes: **fast** – Speculate floating-point operations. (default) **off** – Disables speculation of floating-point operations. **safe** – Do not speculate if this could expose a floating-point exception. **strict** – This is the same as specifying off.
/Qftz[-]	-ftz[-]	When the main program or dll main is compiled with this option, denormal results are flushed to zero for the whole program (dll). Setting this option does not guarantee that all denormals in a program are flushed to zero. It only causes denormals generated at run time to be flushed to zero. On IA-64-based systems, the default is off except at **/O3** (**-O3**). On IA-32- based systems and Intel® 64-based systems, the default is on except at **/Od** (**-O0**), but only denormals resulting from SSE instructions are flushed to zero.

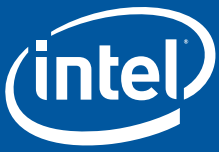
Fine-Tuning (All Processors)

Windows*	Linux* Mac OS*	Comment
/Qunroll[n]	-unroll[n]	Sets the maximum number of times to unroll loops. **/Qunroll0 (-unroll0)** disables loop unrolling. The default is **/Qunroll (-unroll)**, which lets the compiler choose.
/Qrestrict[-]	-[no]restrict	Enables [disables] pointer disambiguation with the restrict keyword.
/Oa	-fno-alias	Assumes no aliasing in the program.
/Ow	-fno-fnalias	Assumes no aliasing within functions.
/Qalias-args[-]	-alias-args[-]	Implies function arguments may be aliased [are not aliased].
/Qopt-class-analysis[-]	-[no-]opt-class-analysis	This option uses C++ class hierarchy information to analyze and resolve C++ virtual function calls at compile time. If a C++ application contains non-standard C++ constructs, such as pointer down-casting, it may result in different behaviors. Default is off, but it is turned on by default with the **/Qipo** (Windows) (**-ipo** on Linux and Mac OS) compiler option, enabling improved C++ optimization. **Note**: Supported for C++ only.
	-fexceptions	This option enables exception handling table generation. In mixed-language applications, this option prevents Fortran routines from interfering with exception handling between C++ routines. Default for C++.
	-fno-exceptions	This option disables exception handling table generation, resulting in smaller code. When this option is used, any use of C++ exception handling constructs (such as try blocks and throw statements) when a Fortran routine is in the call chain will produce an error.
/Qopt-report	-opt-report	Generates an optimization report directed to stderr.
/Qopt-report-level*level*	-opt-report-level*level*	Specifies the verbosity level of the output. Valid ***level*** settings are min (default), med, and max.
/Qopt-report-phase*name*	-opt-report-phase*name*	Reports are generated. The option can be used multiple times in the same compilation to get output from multiple phases. Valid ***name*** arguments are as follows: **all** – All possible optimization reports for all phases **ipo** – Interprocedural Optimizer **ipo_inl** – Gives only the report on function inlining **hlo** – High Level Optimizer **hpo** – High Performance Optimizer **ecg** – Code Generator (Windows* and Linux* on IA-64 only) **ecg_swp** – Gives only the report on software pipelining component of the Code Generator (Windows* and Linux* on IA-64 only) **pgo** – Profile Guided Optimizer
/Qopt-report-routine[rtn]	-opt-report-routine[rtn]	Specifies a routine ***rtn***. Generates reports from all routines with names that include ***rtn*** as part of the name. By default, generates reports for all routines.
/Qopt-report-help	-opt-report-help	Displays all possible settings for **/Qopt-report-phase** (**-opt-report-phase**). No compilation is performed.

Recommended Optimization Options for Specific Intel® Processors

For Best Performance on Processor	Windows*	Mac OS*	Linux*
A future Intel processor that supports SSE4 Vectorizing Compiler and Media Accelerators	/QxS /QaxS	-xS -axS	-xS -axS
Intel® Core™2 Extreme processor Intel® Core™2 Duo processor Dual-Core Intel® Xeon® 5300, 5100,and 3000 series processors Quad-Core Intel® Xeon® processors	/QxT /QaxT	-xT -axT	-xT -axT
Intel® Core™ Duo, Intel® Core Solo processor Intel® Pentium® 4 processor with Streaming SIMD Extension 3 (SSE3) instruction support Intel® Pentium® D processor Intel® Xeon® processor (only on processors that support SSE3) Intel® Pentium® dual-core processor T2060 Intel® Pentium® Extreme Edition processor Dual-Core Intel®Xeon®7000, 5000, and 3200 Sequence processors Dual-Core Intel® Xeon® ULV and LV processor Dual-Core Intel® Xeon® 2.8 processor	/QxP /QaxP	-xP	-xP -axP
Intel processor-based systems supporting SSE2 and SSE* Non-Intel processor-based systems supporting SSE3, SSE2, and SSE* such as AMD processors	/QxO		-xO
Intel Pentium 4 processor Intel® Pentium® M processor Intel Xeon processors without SSE3 support (IA-32 only)	/QxN /QaxN		-xN -axN
Intel processor-based systems supporting SSE* Non-Intel processor-based systems supporting SSE2 and SSE* such as AMD processors	/QxW /QaxW		-xW -axW
Intel® Pentium® III processors Intel® Pentium® III Xeon® processors Non-Intel x86 processor-based systems supporting SSE* such as AMD processors	/QxK /QaxK		-xK -axK
Intel® Itanium® 2 processor	/G2		-mtune=itanium2
Dual-core Intel® Itanium® 2 9000 Sequence processors	/G2-p9000		-mtune= itanium2-p9000

* The *option* values **O**, **W**, and **K** produce binaries that should run on processors not made by Intel such as AMD processors that implement the same capabilities as the corresponding Intel processors. **P** and **N** option values perform additional optimizations that are not enabled with option values **O** and **W**.



For product and purchase information, visit the Intel® Software Development Products site at:
www.intel.com/software/products/compilers

0306/DAM/OMD/PP/3000 254349-006