Overview

Upgrading to the latest IBM® XL Fortran compilers makes good business sense. Upgrading puts new capabilities into the hands of your programmers making them and your business more efficient. New compilers are a must for fully exploiting new hardware but they also help you squeeze more out of your current systems. The last few versions of the IBM XL Fortran compiler introduced a number of new features that provide increased performance optimization, support new language specifications, and exploitation of new hardware and software environments.

IBM has developed extensive and mature, industry-leading compilation technology that covers multiple platforms. This technology uses a modular development structure to deliver performance optimization and functionality on all platforms to all languages.

Each compiler version derives from a common code base. Features and performance optimizations are tested in multiple languages on multiple platforms, making source-level portability of applications between platforms easier and providing a reliable development environment. A common compiler architecture provides a cost efficient and scalable development environment that addresses the requirements of a dynamic enterprise.

Users benefit from upgrading to newer versions of the compiler. The best performance can be obtained in a mixed hardware and software deployment environment, even if the development environment is at an earlier version. Every new hardware or OS version supported includes new features that the compiler uses to get more functionality and additional performance benefits. Recompiling with a newer version of the compiler often allows customer applications to benefit from these new features and get performance gains without source code changes.

IBM compilers are proven technology in scalable development environments and are the compilers of choice for IBM middleware as well as important industry applications and business solutions. New features introduced in the last few releases of the compiler, and their benefits are outlined in the following sections.

Systems exploitation

Used with IBM compilers developed to exploit their capabilities, IBM servers can deliver unprecedented performance, reliability, and energy efficiency. IBM servers, running applications built with IBM compilers have achieved leading performance on industry benchmarks. The OS and hardware exploitation features of the compilers improve programmer productivity by transforming and optimizing code generation and enabling programmers to exploit leading-edge performance of the new hardware without source code changes. The capability of supporting the latest processors, like the POWER7™ processor, offers greater computational performance and precision for business and financial applications, giving users control over performance versus accuracy trade-offs for floating point calculations. Matching the hardware with compilers that are crafted to fully exploit it ensures that your application benefits from the latest industry leading hardware advances.
### Performance

The key strength of the IBM XL compiler family is performance. The IBM XL compilers are unmatched in their ability to optimize and tune code for execution on IBM Power platforms. The performance gain from years of compiler optimization experience can be seen in the release-to-release compiler improvements.

The leading-edge compiler optimizations maximize the performance of applications running on IBM servers helping customers optimize return on their hardware investment. The IBM family of compilers implement internal automatic parallelization and autosimdization as well as the AltiVec/VMX programming interface, the VSX programming interface, and the OpenMP specification for shared memory programming model. All these features raise the parallelism abstraction above APIs making code more portable and programmers more productive.
### Standards compliance

XL compilers comply to the latest international programming language standards, including language interoperability standards, providing support for code portability between multiple operating systems and hardware platforms.

XL compilers assist with programmer productivity and lowering maintenance costs by diagnosing language semantics adherence.

XL compilers help protect the customer’s investment in existing Fortran source and object code built with previous versions of the compilers (supporting FORTRAN 77, Fortran 90, Fortran 95, and Fortran 2003) allowing users to rebuild and link all the components into the same applications using the latest compiler versions. Similarly, object code or libraries compiled with previous versions of XL Fortran are still compatible with the newest XL Fortran compiler and runtime environment.
### Compiler option control

XL compilers provide a rich set of options which enable customers to get their solutions up and running quickly while dealing simply and effectively with product maintenance.

The options provide flexibility in adapting compiler functionality to the required tasks without requiring source code changes. The options assist with programmer productivity and lowering maintenance costs by diagnosing potential language semantics adherence while controlling reliable code generation.
<table>
<thead>
<tr>
<th>Option Category</th>
<th>Benefits of upgrading</th>
</tr>
</thead>
</table>
| **Optimization and tuning:** allow users to control the optimization and tuning process, which can improve the performance of applications at run time | - provide exploitation/tuning for POWER5+ architecture  
- VMX exploitation/tuning  
- support new build-in functions  
- provide information about vectorized program loops including information about when loops cannot be SIMD vectorized  
- support optimization of operations involving complex division and complex absolute values  
- generate optimized code across a range of processor architectures including POWER6™  
- provide user controlled amount of work required in a loop before the compiler will consider it for automatic parallelization  
- support new build-in functions  
- provide fine-grained control over the trade-off between compiler optimization and strict program language specifications  
- options to tune your applications to use the fast scalar versions of the math routines instead of the default versions  
- options for loop transformation analysis of nested loops to improve runtime performance. |
| **Output Control:** control the type of file the compiler produces its location. | control the inline assembler interpretation and code emission  
control the emissions of predefined and user-defined macros  
option to place instructions for each function in a separate object file control section or CSECT which might reduce the size of your program. |
| **Input Control:** specify the type and location of your source files. | allows Fortran-specific preprocessing features in the C preprocessor which ships with XL Fortran |
| **Object Control:** affects the characteristics of the object code, preprocessed code, or other output generated by the compiler | - option to create shared objects  
- option to place instructions for each function in a separate object file control section or CSECT which may reduce the size of a program. |
## Benefits of upgrading

<table>
<thead>
<tr>
<th>Option Category</th>
<th>V10.1</th>
<th>V11.1</th>
<th>V12.1</th>
<th>V13.1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language element control:</strong> specify the characteristics of the source code, to enforce or relax language restrictions, and enable or disable language extensions</td>
<td>adds compatibility options with previous versions of the compilers</td>
<td>control conformance to IEEE 754 semantics, control operation reordering that may violate program language semantics</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Error checking and debugging:</strong> allow users to detect and correct problems in the source code</td>
<td>verify Fortran 2003 compliance, produce information for use by a symbolic debugger</td>
<td>• check for stack overflow &lt;br&gt; • avoid toc overflow conditions in 64-bit compilations &lt;br&gt; • produce optimized pseudocode that can be read by a symbolic debugger</td>
<td>report possible violations of the ANSI aliasing</td>
<td>option to protect your application from malicious code or errors that might corrupt the stack,</td>
</tr>
<tr>
<td><strong>Listings, messages, and compiler information:</strong> allow users control over the listing file, as well as how and when to display compiler messages.</td>
<td>• produce compiler product information &lt;br&gt; • generate listing offsets from the start of a procedure, generate of #line directives in preprocessed output</td>
<td>display information for each compiler component invoked during compilation</td>
<td>• the ability to emit compiler optimization reports in XML format, improving the accessibility of this information for programmers tuning their applications &lt;br&gt; • introduces the compiler utilization tracking and reporting feature for understanding compiler usage within an organization</td>
<td></td>
</tr>
</tbody>
</table>

### Debug capability

XL compilers help increase programmer productivity and lowering maintenance costs by providing information consumable by standard symbolic debugging tools. The user gets increased proficiency and productivity by working in a familiar development environment with chosen debugging tools while debugging source and optimized level code.

XL compilers support application debugging with standard symbolic debugging tools. On AIX, the compilers support any symbolic debugger that supports the AIX XCOFF executable format including dbx, TotalView, and DDT. On Linux®, the compilers support DDT, gdb, or TotalView (which fully supports debugging OpenMP applications).
### Profiling support

XL compilers perform Profile Directed Feedback (PDF) optimization, by collecting information about a program run with typical input data and then applying transformations to the program based on that information. PDF can perform program restructuring in order to ensure that infrequently executed blocks of code are less likely to affect program path length.

Application profile monitoring and Profile Directed Feedback capabilities assist users with tuning application performance. Optimized performance improves return on hardware and software investment.

XL compilers produce monitoring code consumed by the `prof` or `gprof` commands to generate a runtime profile to help tune application performance.

<table>
<thead>
<tr>
<th>V10.1</th>
<th>V11.1</th>
<th>V12.1</th>
<th>V13.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>XL compilers produce information for use by a symbolic debugger</td>
<td>XL compilers produce optimized pseudocode that can be read by a symbolic debugger</td>
<td>function tracing support allowing user routines to be called at function entry and exit</td>
<td></td>
</tr>
</tbody>
</table>

### Library support

XL compilers are shipped with a number of high-quality libraries intended to increase user productivity and speed initial development. Using these libraries can result in fewer bugs and lower long-term maintenance costs.

Optimized libraries, such as MASS, are designed to improve application performance and provide a cost efficient and scalable development environment.
### Library

**Mathematical Acceleration Subsystem (MASS)**
- tuned mathematical intrinsic functions for 32-bit and 64-bit modes, that are thread-safe and offer improved performance over libm
- the functions are called automatically for specific levels of optimization or explicitly through calls at any optimization level
- new scalar and vector functions, support for the POWER5 processor architecture
- scalar and vector functions tuned for the POWER5+™ processor architecture
- scalar and vector functions tuned for the POWER6™ processor architecture
- scalar and vector functions tuned for the POWER7 processor architecture

**Basic Linear Algebra Subprograms (BLAS)**
- high-performance algebraic functions used to perform combined matrix multiplication and addition on general matrices or their transposes

**SMP runtime library**
- supports both explicit and automated parallel processing

---

**April 2010**

References in this document to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM’s program product may be used. Any functionally equivalent program may be used instead.

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

© Copyright IBM Corporation 2010.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.