Benefits of upgrading to the latest z/OS XL C/C++ compiler

Overview

Upgrading to the latest IBM® z/OS® XL C/C++ compiler makes good business sense. Upgrading puts new capabilities into the hands of your programmers making them and your business more efficient. New compilers are essential for fully exploiting new hardware but they also help you squeeze more out of your current systems. Every release of the IBM z/OS XL C/C++ compiler introduces a number of new features including increased performance optimizations, additional support for new language specifications, and exploitation of any new hardware and software environments.

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

Each compiler release 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 releases of the compiler. Every new hardware or z/OS release supported includes new features that the compiler exploits to get more functionality and additional performance benefits. Recompiling with a newer release of the compiler often allows 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 recent releases of the z/OS XL C/C++ compiler and their benefits are outlined in the following sections.

IBM Z exploitation

Used with IBM compilers developed to exploit their capabilities, IBM Z® 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 z/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 IBM z14 processors, 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 advancements.

	Benefits of upgrading			
Feature	**XL C/C++ V2.1**	**XL C/C++ V2.1.1**	**XL C/C++ V2.2**	**XL C/C++ V2.3**
New processor exploitation	zEC12, zBC12	z13™, z13s™		z14

Feature	Benefits of upgrading			
	XL C/C++ V2.1	**XL C/C++ V2.1.1**	**XL C/C++ V2.2**	**XL C/C++ V2.3**
New operating system exploitation	z/OS V2.1		z/OS V2.2	z/OS V2.3
z/OS updated architecture level set (ALS)	z9 EC, z9 BC		z10 EC, z10 BC	zEC12, zBC12
Default ARCH	**ARCH(7)**		**ARCH(8)**	**ARCH(10)**

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 Z. The performance gain from years of compiler optimization experience can be seen in the release-to-release compiler improvements.

The leading-edge compiler optimizations improve the performance of applications running on IBM servers helping maximize the return on hardware investment. This compiler exploits new instructions in the IBM z14. For example, z/OS V2R3 XL C/C++ provides **ARCH(12)** and **TUNE(12)** options to help you exploit new instructions that are available on z14 servers. You can use the new **TUNE(12)** option to generate code that is optimized for z14 processors. These options are designed to provide better performing applications tuned for the new server. These changes can improve the performance of generated code without the need for changes to the source code. Taking advantage of the new architecture, new vector built-in functions are added for enhanced vector support.

z/OS V2R3 XL C/C++ demonstrates on average 13% reduction in CPU time for floating point intensive applications on z14 over the same applications that are built with z/OS V2R2 XL C/C++ on z14. z/OS V2R3 XL C/C++ also demonstrates on average 8% reduction in CPU time for compute intensive applications on z14 over the same applications that are built with z/OS V2R2 XL C/C++ on z14.[1]

Feature	Benefits of upgrading		
	XL C/C++ V2.1	**XL C/C++ V2.1.1 and V2.2**	**XL C/C++ V2.3**
Processor exploitation	Optimization and tuning for zEC12 and zBC12	Optimization and tuning for z13 and z13s	Optimization and tuning for z14
Optimizations	• Exploits the Miscellaneous-Instruction-Extensions Facility and Transactional-Execution Facility • Supports hardware built-in functions for decimal floating point zoned conversion • Provides better performance tuning for LP64 applications	• Exploits the Vector Facility for z/Architecture® • Supports IBM Mathematical Acceleration Subsystem (MASS) and Automatically Tuned Linear Algebra Software (ATLAS) libraries for high-performance mathematical computing	Exploits the z/Architecture® vector facilities in the following aspects to gain potential performance improvements: • Binary floating-point data types • Built-in library functions • Fixed-point decimal operations

1. The performance improvements are based on internal IBM lab measurements. All CPU intensive integer and floating-point benchmarks were compiled in 31-bit addressing mode and built using **XPLINK**, **HGPR**, **O3**, **HOT** and **IPA LEVEL(2)** with **PDF** compiler options. The benchmarks compiled with the z/OS V2R2 XL C/C++ compiler were built using the **ARCH(11) TUNE(11)** options; the benchmarks compiled with the z/OS V2R3 XL C/C++ compiler were built using the **ARCH(12) TUNE(12)** options; All benchmarks were executed on a z/OS V2R2 dedicated LPAR with 1 CP and 8GB Central Storage on z14. Performance results for specific applications will vary, depending on the source code, the compiler options specified, and other factors.

Standards compliance

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

The z/OS XL C/C++ compiler complies to the C89, C99, C++ 98, and C++03 international programming language standards, including language interoperability standards, providing support for code portability between multiple operating systems and hardware platforms. The z/OS XL C/C++ compiler is being upgraded to support the new C11 and C++11 international programming language standards and additional language extensions and features offered by the GNU C/C++ compilers.

	Benefits of upgrading			
	XL C/C++ V2.1	**XL C/C++ V2.1.1**	**XL C/C++ V2.2**	**XL C/C++ V2.3**
GNU C and C++ language extensions	• `__attribute__((malloc))` • `__builtin_expect` • Propagation of attributes to function template instantiations • Zero initialization of objects with an initializer of `()` and an implicitly defined default constructor • Improved diagnostics for invalid template template argument	• `may_alias` type attribute		
C11 subset	• Complex type creation • Static assertions • Anonymous structures • Generic type generics • `_Noreturn` function attribute	• Atomic type		
C++11 subset	• Explicit conversion operators • Generalized `const` expression • `default` and `deleted` functions • Strongly scoped enums • `rvalue` references • Right angle brackets		• Placement `new` operators	

Compiler option control

The z/OS XL C/C++ compiler provides a rich set of options that enable users to get their solutions up and running quickly.

The options provide flexibility in adapting compiler functionality to the required tasks without requiring source code changes. The options assist programmer productivity and lower maintenance costs by diagnosing potential language semantics adherence while controlling reliable code generation.

Option category	Benefits of upgrading			
	XL C/C++ V2.1	**XL C/C++ V2.1.1**	**XL C/C++ V2.2**	**XL C/C++ V2.3**
Optimization and tuning: allow users to control the optimization and tuning process, which can improve the performance of applications at run time	• Options to optimize and tune your applications to exploit the instructions on zEC12 and zBC12 servers • A new SMP option indicating that program code with OpenMP directives, compliant to the OpenMP API 3.1 standard, should be explicitly parallelized • A new **NOTHREADED** option that asserts to the compiler that the code being compiled is single-threaded, allowing for potential compile time and run time performance benefits	• Options to optimize and tune your applications to exploit the instructions on z13 servers	• Option to enable the compiler to generate code, when possible, using the SIMD instructions enabled under the Vector facility for z/Architecture	• Options to optimize and tune your applications to exploit the instructions on z14 servers
Language element control: specify the characteristics of the source code, to enforce or relax language restrictions, and enable or disable language extensions	• Options to enable select features of the new C11 and C++11 language standards	• Option to enable inline assembly code inside C/C++ programs • Option to enable compiler support for vector data types and operations	• Suboption to control whether the `nullptr` feature is enabled	

Option category	Benefits of upgrading			
	XL C/C++ V2.1	XL C/C++ V2.1.1	XL C/C++ V2.2	XL C/C++ V2.3
Error checking and debugging: allow users to detect and correct problems in the source code	• New debug levels that allow you to specify the balance between ease of debugging and the performance of the code generated	• Option to generate function entry events	• Option to control whether a null pointer check is performed on the pointer that is returned by an invocation of the throwing versions of operator new and operator `new[]`	• New **`STACKPROTECT`** option provides protection against malicious code or programming errors that overwrite or corrupt the stack • New suboptions of **`INFO`** that comtrol whether to emit warnings for procedures that are not protected against stack corruption • A new **`DEBUG(NOFILE)`** suboption allows you to place the debugging information in the object file
Listings, messages, and compiler information: allow users control over the listing file, as well as how and when to display compiler messages.				• New suboptions of **`AGGREGATE`** that allow you to specify whether to list the structure member offsets in decimal format or hexadecimal format
Building: although building occurs automatically, these options allow users to effect the build input and output process		• **`-M`** suboptions and enhancements to allow missing headers and setting targets for dependency file generation		

Debug capability

z/OS XL C/C++ helps increase programmer productivity and lower maintenance costs by providing information consumable by standard symbolic debugging tools, including IBM Debug for z Systems and `dbx`. The user benefits from a familiar development environment using debugging tools of choice with increased proficiency and productivity, debugging source and some optimized code.

z/OS XL C/C++ supports the DWARF industry standard format for debugging information. The z/OS XL C/C++ compiler generates debugging information in both DWARF format and the legacy ISD format for compatibility. z/OS XL C/C++ also supports IBM Developer for z Systems, which enables you to examine, monitor, and control the execution of C, C++, COBOL, and PL/I programs.

Benefits of upgrading		
XL C/C++ V2.1	**XL C/C++ V2.1.1 and V2.2**	**XL C/C++ V2.3**
• Improved debugging of optimized code • Debug information for parameters and local variables of each inline instance of a procedure	• Non-XPLINK CDA runtime library for use with non-XPLINK calling code • More complete debugging information by including empty header file names in the `mdbg` file	• Debugging information can be put in the GOFF NOLOAD classes in the object file instead of a separate debug side file

System programming support

The z/OS XL C/C++ compiler provides a feature to support system programming capabilities. This Metal C facility allows you to use C in place of assembler language for system program development. It is capable of generating optimized assembler code, which can take experienced mainframe assembler programmers a relatively long time to develop. Metal C code is highly portable amongst IBM Z. To move from one platform to another, all you need is to recompile Metal C source to target the new platform. This requires no coding effort and therefore significantly reduces cost, risk and time to market.

System programming	Benefits of upgrading			
	XL C/C++ V2.1	**XL C/C++ V2.1.1**	**XL C/C++ V2.2**	**XL C/C++ V2.3**
Metal C	• Allows AMODE-switching at the IPA link step so Metal C applications with AMODE-switching requirements can take advantage of IPA optimization • Allows users to associate an alternate name to `main`. The routine identified as `main` gets all main characteristics such as main style default prolog/epilog, argc/argv parsing, and hook into the RENT run-time • Supports the new **`SYSSTATE`** option that allows users to set the OSREL, ASCENV for the generated HLASM, or both	• The option DSAUSER allows users to leave space size of a pointer on the stack. DSAUSER has been enhanced to accept a sub option **`DSAUSER(value)`**	• Allows requesting user fields of a specific size to be reserved on the stack	• Sets two new flags in the prefix data block to indicate the presence of two new optional fields: one contains the offset of the end of current CSECT and the other one contains the offset of the debug data block • A new debug data block is added for each CSECT, which can be used to check whether the debug side file matches the object file • Allows you to declare a function pointer with the new `__fdptr` keyword so that this function pointer points to a Metal C function descriptor

September 2017