Upgrading 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.

z 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 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 z13™ 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.

Feature	Benefits of upgrading				
	XL C/C++ V1.12	**XL C/C++ V1.13**	**XL C/C++ V2.1**	**XL C/C++ V2.1.1**	**XL C/C++ V2.2**
Processor exploitation	z196, z114, z10™ EC, z10 BC, z9® EC, z9 BC, z990, z890, z900, z800 • optimization and tuning for z196	zEC12, zBC12, z196, z114, z10 EC, z10 BC, z9 EC, z9 BC, z990, z890, z900, z800 • improved z196 and z114 code generation and tuning • initial support for zEC12 and zBC12 (September 2012 PTF)	zEC12, zBC12, z196, z114, z10 EC, z10 BC, z9 EC, z9 BC • optimization of code on zEC12 and zBC12 systems	z13, zEC12, zBC12, z196, z114, z10 EC, z10 BC, z9 EC, z9 BC • optimization of code on z13 systems	z13, zEC12, zBC12, z196, z114, z10 EC, z10 BC • optimization of code on z13 systems
Operating system exploitation	z/OS V1.12	z/OS V1.13	z/OS V2.1	z/OS V2.1	z/OS V2.2

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 Systems™. 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 z13. For example, XL C/C++ for z/OS V2R2 provides ARCHITECTURE(11) and TUNE(11) options to help you exploit new instructions that are available on z13 servers. 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. Additional language support and tuning was done to exploit single instruction, multiple data (SIMD). New built-in functions for vector support were added as well.

The z/OS V2R2 XL C/C++ compiler delivers up to 24% throughput improvement on z13 compared to the z/OS V2R1 XL C/C++ compiler on zEC12. This significant improvement in throughput is achieved through advancements in the hardware and improved compiler optimizations. [1]

1. The performance improvements are based on internal IBM lab measurements. All CPU-intensive integer and floating-point benchmarks were compiled in 64-bit addressing mode and built using XPLINK, HGPR, O3, and HOT compiler options. The majority of the benchmarks were also built with IPA LEVEL(2) with the PDF compiler option. The benchmarks compiled with the z/OS V2R1 compiler were executed on zEC12 and built using the ARCH(10) and TUNE(10) options; the benchmarks compiled with the z/OS V2R2 compiler were executed on z13 and built using ARCH(11) and TUNE(11) options. Performance results for specific applications will vary, depending on the source code, the compiler options specified, and other factors.

Feature	Benefits of upgrading			
	XL C/C++ V1.12	**XL C/C++ V1.13**	**XL C/C++ V2.1**	**XL C/C++ V2.1.1 and V2.2**
Processor exploitation	optimization and tuning for z196	improved z196 code generation and tuning by accessing instructions in the hardware	optimization and tuning for zEC12 and zBC12	optimization and tuning for z13
Optimizations	• provides better performing applications tuned for the new server including improvements to floating-point performance	• simplifies programming access to interlocked storage access instructions • delivers single instruction hexadecimal floating point calculations	• 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 MASS (Mathematical Acceleration Subsystem) and ATLAS (Automatically Tuned Linear Algebra Software) libraries for high-performance mathematical computing

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++ V1.12	**XL C/C++ V1.13**	**XL C/C++ V2.1**	**XL C/C++ V2.2**
• C++11 subset – auto type deduction – C99 long long – C99 preprocessor features adopted in C++0x - decltype - delegating constructors - explicit instantiation declarations - extended friend declarations - inline namespace definitions - static assertion - variadic templates	• GNU C and C++ language extensions – __attribute__((used)), __attribute__((gnu_inline)) – extending the lifetime of C++ temporaries – labels as values – computed goto – extra text after #endif • C++11 subset – trailing return type	• 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 • C11 subset – complex type creation – static assertions – anonymous structures – generic type generics – _Noreturn function attribute • C++11 subset – explicit conversion operators – generalized const expression – default and deleted functions – strongly scoped enums – rvalue references – right angle brackets	• C++11 subset – 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++ V1.12	**XL C/C++ V1.13**	**XL C/C++ V2.1**	**XL C/C++ V2.1.1**	**XL C/C++ V2.2**
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 z196 servers • matches behavior of IPA on other platforms for consistency when using XL compilers on the other platforms	• options to improve hexadecimal floating point calculations • generate interlocked access storage instructions in a single instruction • advanced optimization options to improve optimization of Metal C programs	• 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
Language element control: specify the characteristics of the source code, to enforce or relax language restrictions, and enable or disable language extensions	• option to enable deeper pointer analysis and results in improved performance of the application being compiled • options to control language elements	• option to provide increased C++ template control (maximum number of recursively instantiated template specializations to be processed by the compiler)	• 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++ V1.12	XL C/C++ V1.13	XL C/C++ V2.1	XL C/C++ V2.1.1	XL C/C++ V2.2
Error checking and debugging: allow users to detect and correct problems in the source code		• debug parameters for inlined procedures	• new debug levels which 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[]
Listings, messages, and compiler information: allow users control over the listing file, as well as how and when to display compiler messages.	• aliasing diagnostics • modify diagnostic severity to match development environment	• informational messages is the default in z/OS USS for consistency with batch compilations			
Building: although building occurs automatically, these options allow users to effect the build input and output process	• set the severity for certain messages to match build process rules for cases which are known not to be problems • an environment variable enables control over the size of the temporary data set used during IPA link			• -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 Tool 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 Debug Tool for z/OS, which enables you to examine, monitor, and control the execution of C, C++, COBOL, and PL/I programs.

Benefits of upgrading		
XL C/C++ V1.13	**XL C/C++ V2.1**	**XL C/C++ V2.1.1 and V2.2**
• improved debugging of development-time programs to more closely align with production programs • debugging APIs for easier access to debug information in debug files • debug information for inline procedures, ability to set entry breakpoints at all inline instances	• 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

Profiling support

The z/OS XL compiler performs 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 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 programmers with tuning application performance and therefore result in improved return on hardware and software investments.

In XL C/C++ for z/OS, V1R13, a feature to reuse pdf data until the data is out of date and to provide percent of relevancy is added.

Middleware support

IBM z/OS XL C/C++ is not only designed to unleash the full power of the IBM z Systems architecture but also the middleware. z/OS XL C/C++ targets application development and deployment on z/OS and supports batch processing, legacy file systems, and provides a major programming interface for CICS®, IMS™ and DB2®.

With the integrated SQL and CICS co-processors, z/OS XL C/C++ allows efficient application design for working in a CICS environment or interacting with DB2.

Middleware	Benefits of upgrading			
	XL C/C++ V1.12	**XL C/C++ V1.13**	**XL C/C++ V2.1**	**XL C/C++ V2.2**
CICS	• an option supports constructed reentrancy for Metal C programs with writable static and external variables so Metal C programs can now be concurrently used by multiple users, and Metal C can be used to write programs to run in CICS Transaction Server for z/OS • CICS transactions can be extended with technologies such as Java™, C/C++ and the Metal C facility to deliver value in new and innovative ways, without incurring the substantial cost required to rip and replace current core assets. Provides ability to operate both new and existing applications within the same system and in close proximity to the corporate data residing on z/OS	• support for CICS Transaction Server for z/OS (V3, V4)	• support for CICS Transaction Server for z/OS (V3, V4, V5)	• support for CICS Transaction Server for z/OS (V3, V4, V5)
DB2	• support for DB2 V9 for z/OS	• support for DB2 for z/OS (V9, V10)	• support for DB2 for z/OS (V9, V10, V11)	• support for DB2 for z/OS (V9, V10, V11)

Middleware	Benefits of upgrading			
	XL C/C++ V1.12	**XL C/C++ V1.13**	**XL C/C++ V2.1**	**XL C/C++ V2.2**
IMS	• IMS transactions can be extended with technologies such as Java, C/C++ and the Metal C facility to deliver value in new and innovative ways, without incurring the substantial cost required to rip and replace current core assets. Provides ability to operate both new and existing applications within the same system and in close proximity to the corporate data residing on z/OS • support for IMS (V10)	• XL C/C++ provides a library function and callable interface to use IMS from C/C++ • support for IMS (V10, V11)	• support for IMS (V11, V12, V13)	• support for IMS (V11, V12, V13)

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 z Systems. 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++ V1.12	**XL C/C++ V1.13**	**XL C/C++ V2.1**	**XL C/C++ V2.2**
Metal C	• constructed re-entrancy to Metal C programs allows them to be concurrently used by multiple users	• supports HOT optimizations to improve Metal C programs that use loops • supports IPA optimizations enabling performance gains for Metal C programs • identifies C functions and their associated properties when code scanning or dump reading • enables automatic argument parsing to ease programming burden, simplify new program development, and porting programs to Metal C • provides a means of reserving a space of pointer size on the stack	• 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	• allows requesting user fields of a specific size to be reserved on the stack

September 2015