Upgrading XL C/C++ Compilers (Linux for Little Endian Distributions and AIX)

December 2017

Chapter 1. Overview

Upgrading your team's XL C/C++ compiler for AIX® or Linux to the latest release makes good business sense. An upgrade gives your programmers access to new capabilities, and leads to better business efficiency. New compilers allow your software to exploit the performance potential of new IBM hardware, and let you squeeze more performance out of your current hardware. Each version of the IBM® XL C/C++ compiler introduces performance enhancements, language support enhancements, and exploitation of the latest IBM hardware and operating systems.

Your applications can benefit from upgrading to newer versions of the XL C/C++ compilers even if your own development environment uses older hardware or operating systems. You can exploit new operating system features or hardware performance capabilities just by rebuilding your existing applications with newer compiler versions, without changing your source code. You can also take advantage of optimization, reliability, and usability enhancements added to each new compiler release.

IBM XL compilers provide a proven path to scalable development. They are the compilers of choice for IBM middleware products and for important industry applications and business solutions. This document outlines compiler features introduced in recent compiler releases, and highlights their benefits.

Chapter 2. System exploitation

When you build your software with IBM XL compilers, you can obtain exceptional performance, reliability, and energy efficiency on IBM servers. This combination of IBM servers and IBM XL compilers has produced performance leadership on several industry benchmarks. The compilers exploit the latest operating system and hardware features, and you can take advantage of those features with no source code changes, simply by rebuilding your applications with the most recent compilers.

System support and exploitation

While applications built with earlier versions of the XL C/C++ compiler will run on newer IBM servers, only by upgrading to the most recent compilers can you fully exploit the processor and system capabilities of the most recent IBM POWER® servers.

The following table shows the improvements for a representative sample of benchmarks compiled at the **-O2** optimization level with the default settings of **-qarch** and **-qtune**.

Note: The following measurements represent the geometric mean of a representative sample of floating-point and fixed point workloads. The measurements are based on SPEC2006 benchmarks on XL C/C++ for AIX compilers. Your application performance might vary.

Table 1. System support and exploitation

Compiler version	Benefits of upgrading	Sample gains over V10.1 on the current POWER8® hardware
V13.1	Adds exploitation of and tuning for IBM POWER8	• Fixed point / integer: – +12.5% with **-qarch=pwr8 -qtune=pwr8** – +8% with default **-qarch=pwr4 -qtune=balanced** • Floating point: – +19.5% with **-qarch=pwr8 -qtune=pwr8** – +13.1% with default **-qarch=pwr4 -qtune=balanced**
V12.1	Provides enhanced tuning for IBM POWER7® compared to V11.1	• Fixed point / integer: +4.3% • Floating point: +1%
V11.1	Adds exploitation of and tuning for IBM POWER7 including automatic exploitation of POWER7 VSX vector capability	• Fixed point / integer: +3.7% • Floating point: -1.2%
V10.1	Support for IBM POWER4, IBM POWER5, IBM POWER5+ and IBM POWER6®; tuning for IBM POWER6	N/A

Operating systems and toolchain exploitation

Newer operating system versions provide system and library support for more recent IBM processor levels, as well as general improvements to performance and reliability. Newer IBM XL compilers for Linux on POWER also support newer versions of the IBM Advance Toolchain. The Advance Toolchain is a set of open source development tools and runtime libraries for Linux on POWER, which allows users to take advantage of the functional and performance enhancements in the latest POWER hardware.

Table 2. Operating systems on AIX

Compiler version	Operating systems supported
V13.1.3	• AIX V6.1 TL 2 Service Pack 5 or later • AIX V7.1 • AIX V7.2 • IBM i V7.1 PASE V7.1 • IBM i V7.1 PASE V7.2
V13.1.2	• AIX V6.1 TL 2 Service Pack 5 or later • AIX V7.1 • IBM i V7.1 PASE V7.1 • IBM i V7.1 PASE V7.2
V13.1.0	• AIX V6.1 TL 2 Service Pack 5 or later • AIX V7.1 • IBM i V7.1 PASE V7.1
V12.1	• AIX 5.3 TL 5300-07 or later • AIX 6.1 • AIX 7.1 • IBM i V6.1 PASE V6.1 with PTF SI30636 or later • IBM i V7.1 PASE V7.1
V11.1	• AIX 5.3 TL 5300-07 or later • AIX 6.1 • IBM i V6.1 PASE V6.1 with PTF SI30636 or later
V10.1	• AIX V5.3 TL 5300-06 or later • AIX V6.1 • IBM i V6.1 PASE

Table 3. Operating systems and toolchain exploitation on Linux for little endian distributions

Compiler version	Operating systems supported	Toolchain exploitation
V13.1.6	• Red Hat Enterprise Linux 7.3 (RHEL 7.3) • Red Hat Enterprise Linux 7.4 (RHEL 7.4) • Red Hat Enterprise Linux 7.4 for Power® Little Endian (POWER9) • SUSE Linux Enterprise Server 12 (SLES 12) • SUSE Linux Enterprise Server 12 Service Pack 3 (SLES 12 SP3) • Ubuntu Server 14.04 • Ubuntu Server 14.10 • Ubuntu Server 16.04 • Community Enterprise Operating System 7 (CentOS 7) **Note:** To compile programs that contain code to be offloaded to the NVIDIA GPUs, you must use either of the following operating systems: • Ubuntu Server 16.04.3 • Red Hat Enterprise Linux 7.3 (RHEL 7.3) or above	• IBM Advance Toolchain 11.0 for Linux on Power • IBM Advance Toolchain 9.0 for Linux on Power • IBM Advance Toolchain 8.0 for Linux on Power
V13.1.5	• Red Hat Enterprise Linux 7.1 (RHEL 7.1) • Red Hat Enterprise Linux 7.2 (RHEL 7.2) • Red Hat Enterprise Linux 7.3 (RHEL 7.3) • SUSE Linux Enterprise Server 12 (SLES 12) • SUSE Linux Enterprise Server 12 Service Pack 1 (SLES 12 SP1) • Ubuntu Server 14.04 • Ubuntu Server 14.10 • Ubuntu Server 16.04 • Community Enterprise Operating System 7 (CentOS 7)	• IBM Advance Toolchain 9.0 for Linux on Power • IBM Advance Toolchain 8.0 for Linux on Power

Table 3. Operating systems and toolchain exploitation on Linux for little endian distributions (continued)

Compiler version	Operating systems supported	Toolchain exploitation
V13.1.4	• Red Hat Enterprise Linux 7.1 (RHEL 7.1) • Red Hat Enterprise Linux 7.2 (RHEL 7.2) • SUSE Linux Enterprise Server 12 (SLES 12) • SUSE Linux Enterprise Server 12 Service Pack 1 (SLES 12 SP1) • Ubuntu Server 14.04 • Ubuntu Server 14.10 • Ubuntu Server 16.04 • Community Enterprise Operating System 7 (CentOS 7)	• IBM Advance Toolchain 9.0 for Linux on Power • IBM Advance Toolchain 8.0 for Linux on Power
V13.1.3	• Red Hat Enterprise Linux 7.1 (RHEL 7.1) • Red Hat Enterprise Linux 7.2 (RHEL 7.2) • SUSE Linux Enterprise Server 12 (SLES 12) • Ubuntu Server 14.04 • Ubuntu Server 14.10	• IBM Advance Toolchain 9.0 for Linux on Power • IBM Advance Toolchain 8.0 for Linux on Power
V13.1.2	• Red Hat Enterprise Linux 7.1 (RHEL 7.1) • SUSE Linux Enterprise Server 12 (SLES 12) • Ubuntu Server 14.04 • Ubuntu Server 14.10	• IBM Advance Toolchain 8.0 for Linux on Power
V13.1.1	• SUSE Linux Enterprise Server 12 (SLES 12) • Ubuntu Server 14.04 • Ubuntu Server 14.10	• IBM Advance Toolchain 8.0 for Linux on Power

Table 4. Operating systems on Linux for big endian distributions

Compiler version	Operating systems supported
V13.1.0	• SUSE Linux Enterprise Server 11 Pack 2 (SLES 11 SP2) or later • Red Hat Enterprise Linux 6.4 (RHEL 6.4) or later • Red Hat Enterprise Linux 7.0 (RHEL 7.0) or later
V12.1.0	• SUSE Linux Enterprise Server 10 Pack 4 (SLES 10 SP4) • SUSE Linux Enterprise Server 11 Pack 2 (SLES 11 SP2) • Red Hat Enterprise Linux 5.7 (RHEL 5.7) • Red Hat Enterprise Linux 6.2 (RHEL 6.2)

Table 4. Operating systems on Linux for big endian distributions (continued)

Compiler version	Operating systems supported
V11.1.0	• SUSE Linux Enterprise Server 10 Pack 2 (SLES 10 SP2) • SUSE Linux Enterprise Server 11 Pack 1 (SLES 11 SP1) • Red Hat Enterprise Linux 5.5 (RHEL 5.5)
V10.1.0	• SUSE Linux Enterprise Server 10 Pack 2 (SLES 10 SP2) • Red Hat Enterprise Linux 5.2 (RHEL 5.2)

Chapter 3. Performance and optimization

The IBM XL compilers excel at delivering optimal performance to customer applications, IBM middleware, and industry benchmarks running on IBM POWER platforms. With each successive compiler release, the XL compilers deliver both runtime performance improvements and reduced compile time, allowing your software to run faster on your existing IT infrastructure as well as to exploit the latest IBM processors.

The IBM compilers have introduced advancements across multiple releases that deliver increased parallelization to your applications, typically with little or no effort on the part of your programmers. For example, at higher optimization levels, IBM compilers provide automatic parallelization and automatic vectorization. The IBM XL compilers, from version 13.1, provide significant performance improvements to OpenMP applications, and to applications built with automatic parallelization. Each compiler level released to support a newer IBM POWER server also adds support for programmer exploitation of the latest VMX or VSX vector registers, through the AltiVec programming interface. All these features help abstract the level of parallelism, making your source code more portable, and your programmers more productive.

Vectorization support

Automatic SIMDization is an optimization where the compiler generates vector (Single Instruction, Multiple Data, or SIMD) instructions in loops operating on arrays of floating point or integer values. These SIMD operations can operate on two, four, eight, or sixteen values at a time, depending on the size of the data type.

Auto-vectorization is an optimization in which the XL compilers detect calls to math library functions, such as pow(), exp(), and cos(), in loops and replace those calls with calls to the Vector MASS library functions. Vector MASS library functions provide significant performance gains in exchange for, in some cases, slightly reduced accuracy compared to the default library, or differences in the handling of corner cases.

Table 5. Vectorization support

Compiler version	**Vectorization support added in the release**
V13.1.6	Re-engineers automatic SIMDization component for improved exploitation of POWER9 vector instructions
V13.1	Re-engineers automatic SIMDization component for improved exploitation of POWER6, POWER7, and POWER8 vector instructions
V12.1	Adds directives for more fine-grained control of automatic SIMDization
V11.1	Adds support for the IBM POWER7 VSX vector instruction set and enhanced autovectorization support
V10.1	Adds support for IBM POWER6 VMX instructions. Provides AltiVec support using the single instruction, multiple data instruction set. Adds support for automatic vectorization of elementary math functions

Optimization support

Table 6. Optimization support

Compiler version	Optimizations added or enhanced
V13.1.6	• Provides MASS libraries tuned for POWER9 • Enhances the support for offloading computations to the NVIDIA GPUs, including improved GPU code generation and specification of GPU architectures for the generated code
V13.1.5	• Offloads compute-intensive parts of an application and associated data to the NVIDIA GPUs by using the device constructs that are supported by XL C/C++ for Linux, V13.1.5
V13.1.4	• Improves Profile Directed Feedback (PDF) by allowing dumping snapshot PDF profiling information to files during execution, which is especially useful if you want to save the generated PDF profiling information when the application is to be terminated abnormally
V13.1	• Adds visibility attribute support, which helps decrease shared library size, improves the efficiency of dynamic linking, and allows more optimizations at compile and link step • Improves Profile Directed Feedback (PDF) by allowing segregation of PDF data into workloads and avoiding lock contention on the PDF data file • Provides inlining enhancements including support for inlining small functions, support for marking functions as always inline in the source code, and support for controlling inlining of particular functions from the command line
V12.1	• Extends the **-qpic=large** option to produce a more efficient TOC when building large applications • Provides additional loop pragmas to allow programmers to guide optimization by indicating loop iterator ranges or loop frequency
V11.1	• Adds three new **-qpdf** suboptions, which allow more fine-grained control over performance improvement and extend **-qpdf** to support multiple pass profiling, cache miss profiling, block counter profiling, call counter profiling, and value profiling • Adds a new **-qhot** suboption to add more aggressive loop analysis • Adds the new **-qrestrict** option, which enables the compiler to perform more aggressive transformations, providing further performance improvements
V10.1	• Adds suboptions to the **-qstrict** option to allow more fine-grained control over optimizations and transformations. These suboptions allow relaxation of specific strict program semantics, which can give you the benefit of faster code without turning off all semantic verification

Parallelization support

Table 7. Parallelization support

Compiler version	**Parallelization added or enhanced**
V13.1.6 (LE Linux)	• The LE compiler adds support for these OpenMP 4.5 features: – Directives - omp simd - omp for simd - omp distribute simd - omp distribute parallel for simd - omp parallel for simd - omp target simd - omp target parallel for simd - omp target teams distribute simd - omp target teams distribute parallel for simd - omp teams distribute simd - omp teams distribute parallel for simd – Clauses - omp_declare_target - omp_ordered - omp_target - omp_target_data - omp_target_enter_data - omp_target_exit_data - omp_target_update - omp_task

Table 7. Parallelization support (continued)

Compiler version	Parallelization added or enhanced
V13.1.5 (LE Linux)	• The LE compiler adds support for these OpenMP 4.5 features: – Directives - omp declare target - omp distribute parallel for - omp target - omp target data - omp target enter data - omp target exit data - omp target parallel - omp target parallel for - omp target teams - omp target teams distribute - omp target teams distribute parallel for - omp target update - omp teams - omp teams distribute - omp teams distribute parallel for – Functions - omp_get_default_device - omp_get_initial_device - omp_get_num_devices - omp_get_num_teams - omp_get_team_num - omp_is_initial_device - omp_set_default_device - omp_target_alloc - omp_target_associate_ptr - omp_target_disassociate_ptr - omp_target_free - omp_target_is_present - omp_target_memcpy – Environment variables - OMP_DEFAULT_DEVICE = *n* - XLSMPOPTS = TARGET = {MANDATORY | OPTIONAL | DISABLE}

Table 7. Parallelization support (continued)

Compiler version	Parallelization added or enhanced
V13.1.4 (LE Linux) V13.1.3 (LE Linux)	• The LE compiler adds support for these OpenMP 4.0 features: – omp_get_proc_bind function – The OMP_PLACES environment variable The following environment variables are extended to control the thread affinity policy: – OMP_DYNAMIC – OMP_DISPLAY_ENV – OMP_PROC_BIND – OMP_THREAD_LIMIT The ATOMIC directive is extended to support sequentially consistent atomic operations by specifying a new optional clause seq_cst. • The LE compiler adds support for these OpenMP 4.5 features: – omp_get_num_places function – omp_get_partition_num_places function – omp_get_partition_place_nums function – omp_get_place_num_procs function – omp_get_place_proc_ids function – omp_get_place_num function – The proc_bind clause
V13.1.2 (LE Linux)	• The LE compiler fully supports OpenMP 3.1 features. • The LE compiler adds support for these OpenMP 4.0 features: – Atomic update, capture, and swap – OMP_DISPLAY_ENV environment variable
V13.1	• Re-engineers the OpenMP runtime to significantly improve performance in OpenMP and autoparallelized applications (AIX only). • Updates the tasks algorithm to make it more efficient by using multiple queues (task stealing). • Adds support for the OpenMP 4.0 specification, including the following features: – Atomic update and capture clauses enhancements – OMP_DISPLAY_ENV environment variable – Nested parallelism
V12.1	Adds support for the OpenMP V3.1 specification, including the following features: • User-defined task switching • Atomic constructs extensions to atomic read, write and capture • Task data environments • OMP_NUM_THREADS environment variable • OMP_PROC_BIND environment variable • Max/min C/C++ reduction list
V11.1	Adds support for the OpenMP specification, including the following features: • Enhanced automatic parallelization • Reordered OMP Outlining (DO instruction) • Allow thread binding to multiple logical processors

Table 7. Parallelization support (continued)

Compiler version	Parallelization added or enhanced
V10.1	Adds support for the OpenMP V3.0 specification, including the following features: • Task-level parallelization • New variable types in for loops • Stack size control • Task directive clauses • OMP loop collapsing • OMP_THREAD_LIMIT environment variable • OS TLS for omp threadprivate

Compile time improvements

Successive releases of the IBM XL compilers have provided reductions in compilation time for a given optimization level, while delivering equivalent or better performance. This makes it easier for you to exploit more advanced optimization levels.

The compile time percentage improvements in the following table are based on SPEC2006 benchmarks on XL C/C++ for AIX compilers.

Compiler version	Compile time improvements compared with the previous release
V13.1	37.8% faster at -O2; 67.3% faster at -O3; 87% faster at -O2 IPA (interprocedural analysis)
V12.1	13.2% faster at -O2
V11.1	19.7% faster at -O2
V10.1	N/A

Chapter 4. Standards compliance

The XL compilers provide support for the programming language standards, including standards for interoperability between different languages and support for C11, C++11, and C++14. This conformance makes it easier for your programming team to port applications between operating systems and hardware platforms. The compilers analyze your source code for language semantic adherence, to help your programmers write standards-conforming applications. The compilers also support commonly used language extensions to provide code portability across the full range of IBM and OEM platforms.

Table 8. Standards compliance

Compiler version	**Language standards supported**
V13.1.6 (LE Linux) V13.1.5 (LE Linux) V13.1.4 (LE Linux) V13.1.3 (LE Linux)	• Compliance with C11 and C++11 • Partial support for C++14 • Support for the following optional C11 and C++11 language features: – Atomic types and operations (partial support) to enable synchronization and communication between threads. – Thread-Local Storage to maintain data that is local to a thread. Note that the compiler fully supports the thread-local feature only when GCC runtime library V4.8 or later is used. • Full support for OpenMP 3.1 • Partial support for OpenMP 4.0 and OpenMP 4.5. For more information about OpenMP 4.5 support, see Parallelization support.

Table 8. Standards compliance (continued)

Compiler version	Language standards supported
V13.1.2 (LE Linux)	The XL C/C++ for Linux compiler on little endian distributions adds support for the following language features: • C++14 – Polymorphic lambda expressions – Variable templates • ISO/IEC 14882:2011 (C++11) – Alignment support – constexpr – Explicit overrides and final – Generalized attributes – Inheriting constructors – Local and unnamed types as template arguments – Monomorphic lambdas expressions – New character types – New definitions of POD types – noexcept – Nonstatic data member initializers – Range-based for – Raw string literals – ref_qualifiers – Template aliases – Unicode names (UCN) and unicode literals – Uniform initialization – Unrestricted unions – User-defined literals • C11 – Complex type initializations – Composite types for variable length arrays – Conversions between pointers and floating types – Generic selection – Temporary lifetime extensions – typedef redeclarations – Unicode and UTF-8 literals • Full support for OpenMP 3.1 • Partial support for OpenMP 4.0
V13.1	• ISO/IEC 14882:2011 (C++11) – Defaulted and deleted functions – Generalized constant expressions – Rvalue references (part 2) – Uniform initialization (part 1 to support Linux header usage) • C11 – Generic selection – The typedef redeclaration • (Big endian only) Partial support for OpenMP 4.0

Table 8. Standards compliance (continued)

Compiler version	Language standards supported
V12.1	• ISO/IEC 14882:2011 (C++11) – Explicit conversion operators – Forward declaration of enumeration – Generalized constant expressions – Reference collapsing – Right angle brackets – Rvalue references (part 1) – Scoped enumeration – Trailing return type • C11 – Anonymous structures – Complex type initialization – New language level -qlanglvl=extc1x – The _Noreturn function specifier – Static assertions • OpenMP V3.1 conformance • GCC atomic memory access intrinsic function support • Improved support for C99 restrict pointers
V11.1	• ISO/IEC 14882:2011 (C++11) – Auto type deduction – C99 long long – C99 preprocesoor features adopted in C++11 – Decltype – Delegating constructors – Explicit instantiation declarations – Extended friend declarations – Extern templates – Inline namespace definitions – Static assertion – Variadic templates • Language extensions to support VSX vector programming

Table 8. Standards compliance (continued)

Compiler version	Language standards supported
V10.1	• ISO/IEC 9899:1999 (C99) • ISO/IEC 9899:1990 (C89) • ISO/IEC 14882:2003 (C++03) • ISO/IEC 14882:1998 (C++98) • Language extensions to support VMX vector programming • Decimal floating point support (C and C++) • TR1 library extensions for C++ Subset of GNU C and C++ extensions • ISO/IEC 14882:2011 (C++11) – A new language level for C++11 – New integer promotion rules for arithmetic conversions involving long long data types – Support of C99 preprocessor features in C++ for easier porting of code between C and C++ • OpenMP V3.0 extensions to support parallelized programming

Chapter 5. Debug capabilities

XL compilers help increase programmer productivity and lower maintenance costs by providing information consumable by standard symbolic debugging tools. This allows your programmers to take advantage of a familiar development environment by using debugging tools of their choice. Support for debugging optimized code in later releases also enables your programmers to debug the exact version of compiled code that will be shipped to customers.

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, DDT, and IBM Debugger for AIX. On Linux, the compilers support DDT, gdb, or TotalView (which fully supports debugging OpenMP applications).

Table 9. Debug capability

Compiler version	**Benefits of upgrading**
V13.1.6 (LE Linux)	• Enables the DWARF debugging information to be generated in one or more separate .dwo files through `-gsplit-dwarf option`. • Provides general debug at opt support with optimization level -O2 through `-g8 and -g9 options`, which provide the same functionality as provided in previous IBM XL C/C++ releases for big endian platforms. • Enable debugging inlined functions when no optimization (-qnoopt) or optimization level 2 (-O2) is enabled.
V13.1	**C++11 rvalue references debug support** Provides a way of describing rvalue references in DWARF so that users can access debug information related to an rvalue reference. **C++11 Strongly Typed Enums** Provides a way for the user to identify scoped vs. unscoped enums when looking at variables and types. **C++ qualifiers** Provides the C++ qualifier "::" instead of qualifier "__" on DWARF so that users can access debug information in a way that is aligned with C++ syntax. **C++ template arguments debugging** Helps obtain the actual value and its original argument name of a template argument, whether it is a non-type template argument, a type template argument, or a template template argument. **C++ namespace support** Generates namespace debug information in DWARF. The scope information was not accessible in the debugger before.
V12.1	Provides better support for debugging optimized code through various debug levels of -g (0-9).
V11.1	Provides function tracing support, which allows user routines to be called at function entry and exit.

Table 9. Debug capability (continued)

Compiler version	Benefits of upgrading
V10.1	• (XL C/C++ Enterprise Edition for AIX only) Includes the IBM Debugger for AIX. • Produces optimized pseudocode that can be read by a symbolic debugger.

Chapter 6. MASS library support

XL compilers are shipped with high performance MASS (Mathematical Acceleration SubSystem) libraries providing scalar, vector, and SIMD versions of common elementary functions (such as `exp`, `log`, `sin`, `cos`) in both single and double precision. These libraries target specific POWER processors to maximize application performance. They increase user productivity, speed initial development, and cut long-term maintenance costs.

All versions of MASS provide tuned mathematical intrinsic functions for 32-bit and 64-bit modes that are threadsafe and offer improved performance over `libm`. The functions are invoked automatically by the compiler for specific levels of optimization or they can be explicitly called in the source program at any optimization level.

In addition to the following features listed, V10.1 and later also contain the MASS scalar library, the MASS vector library tuned for POWER5 and POWER5+ processors, and the MASS vector library tuned for POWER6 processors.

Table 10. MASS library support

Compiler version	**Benefits of upgrading**
V13.1.6 (LE Linux)	• Adds a new MASS vector library tuned for POWER9 architecture • Adds a new MASS SIMD library tuned for POWER9 architecture
V13.1	• Adds a new MASS vector library tuned for POWER8 processors, that exploits the POWER8 vector instruction set • Adds a new MASS SIMD library tuned for POWER8 processors, that exploits the POWER8 vector instruction set • Adds 6 new MASS vector functions for automatic invocation by the compiler • Provides additional tuning of the compiler's autovectorization capability
V12.1	• Provides additional performance tuning of the MASS vector and SIMD libraries for the POWER7 and POWER7+™ processors • Improves MASS vector function performance for misaligned operand vectors • Adds a new MASS vector reciprocal multiply-add function for automatic invocation by the compiler • Provides additional tuning of the compiler's autovectorization capability
V11.1	• Adds a new MASS vector library tuned for POWER7 processors that exploits the POWER7 vector instruction set. This library has 18 new functions, for a total of 78 functions. • Adds a new MASS SIMD library tuned for POWER7 processors that exploits the POWER7 vector instruction set. This library has 74 new functions and supports vector datatype operands and results.

Printed in USA