Why should I upgrade to the latest compiler?
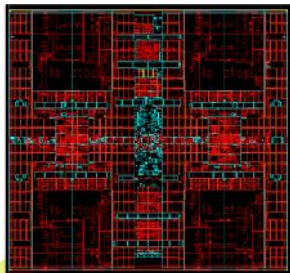
Why IBM Compilers ?

- Reduce CPU utilization by improving performance of applications

  – Exploitation of z/Architecture

  – Advanced Optimization

- Support next-generation Applications

  – Work with latest IBM Middleware (CICS, IMS, DB2…)

  – Modernize business critical applications to support new applications (e.g. web and mobile applications)

- Improve Productivity

  – New programming features

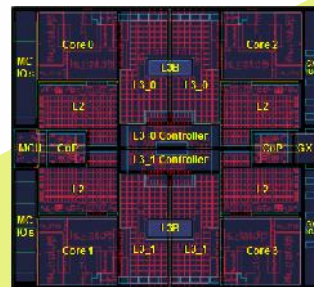 – New problem determination features

Exploitation of z Processors

Need Update

zNext
2015

zEC12
2012

z196
2010

z10
2008

Leadership Single Thread, Enhanced Throughput

Improved out-of-order

Transactional Memory

Dynamic Optimization

2 GB page support

Step Function in System Capacity

Top Tier Single Thread Performance, System Capacity

Accelerator Integration

Out of Order Execution

Water Cooling

PCIe I/O Fabric

RAIM

Enhanced Energy Management

Workload Consolidation and Integration Engine for CPU Intensive Workloads

Decimal FP

Infiniband

64-CP Image

Large Pages

Shared Memory

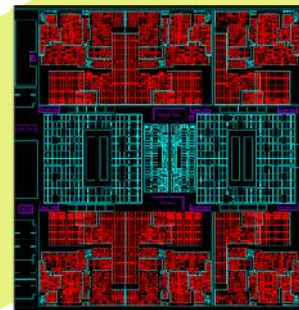# Developing next-generation Applications

Extend Business Critical (SoR) Applications to support new System of Engagement (SoE) Applications; Increase performance and increase efficiency and quality of software delivery with new compiler technologies

Systems of Engagement (SoE) Apps

Web Apps

Databases

Systems of Record (SoR) Apps

Databases

New System z Compilers

- Enterprise COBOL for z/OS v5.2

- XL C/C++ Compiler for Linux on z Systems **NEW**

- Enterprise PL/I for z/OS v4.5

- z/OS XL C/C++ V2R1M1

New System z Compilers
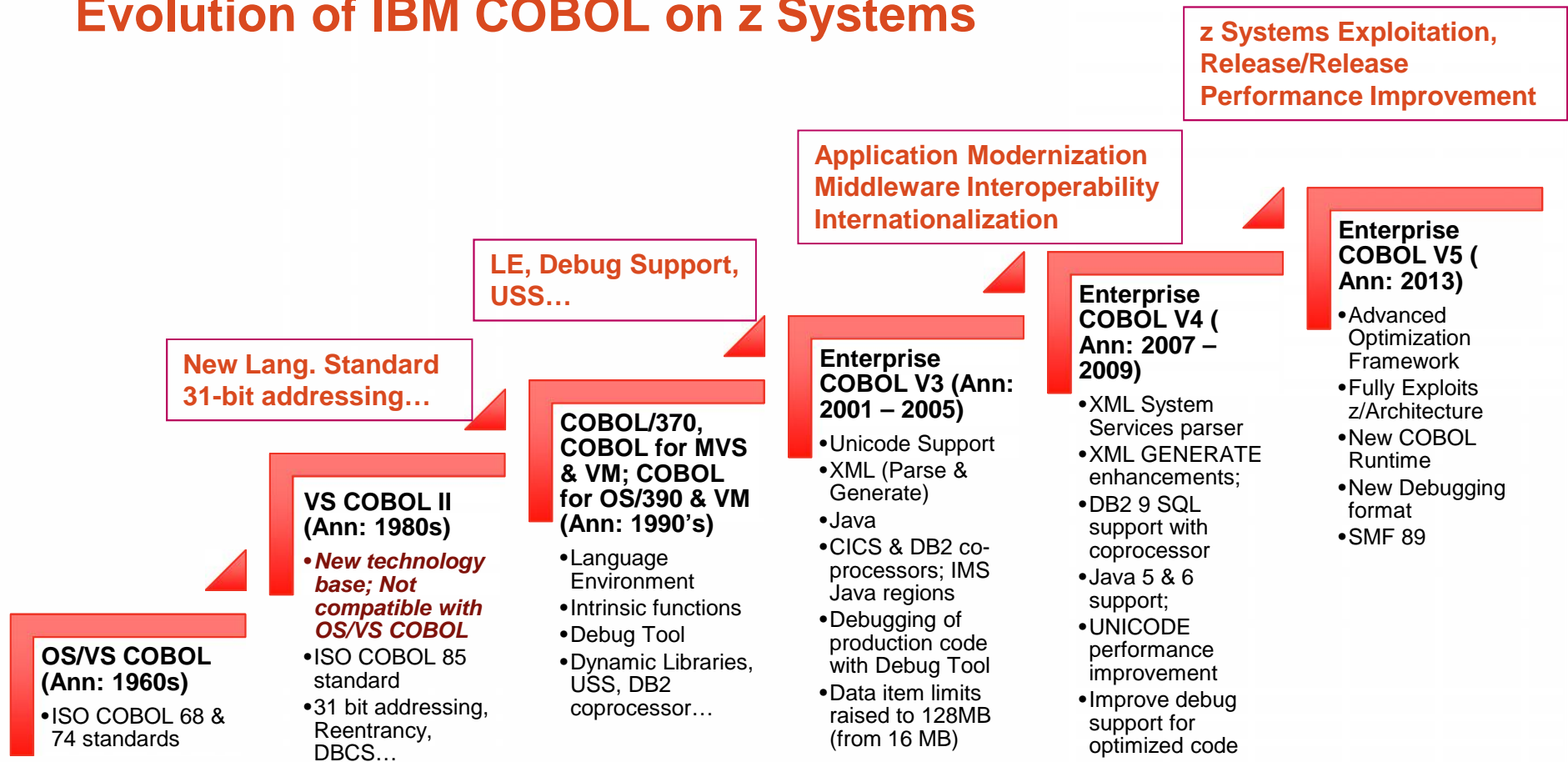
- Enterprise COBOL for z/OS v5.2

- XL C/C++ Compiler for Linux on z Systems

- Enterprise PL/I for z/OS v4.5

- z/OS XL C/C++ V2R1M1

Evolution of IBM COBOL on z Systems

z Systems Exploitation, Release/Release Performance Improvement

Application Modernization Middleware Interoperability Internationalization

LE, Debug Support, USS…

New Lang. Standard 31-bit addressing…

Enterprise COBOL V5 (Ann: 2013)
- Advanced Optimization Framework
- Fully Exploits z/Architecture
- New COBOL Runtime
- New Debugging format
- SMF 89

Enterprise COBOL V4 (Ann: 2007 – 2009)
- XML System Services parser
- XML GENERATE enhancements;
- DB2 9 SQL support with coprocessor
- Java 5 & 6 support;
- UNICODE performance improvement
- Improve debug support for optimized code

Enterprise COBOL V3 (Ann: 2001 – 2005)
- Unicode Support
- XML (Parse & Generate)
- Java
- CICS & DB2 co-processors; IMS Java regions
- Debugging of production code with Debug Tool
- Data item limits raised to 128MB (from 16 MB)

COBOL/370, COBOL for MVS & VM; COBOL for OS/390 & VM (Ann: 1990's)
- Language Environment
- Intrinsic functions
- Debug Tool
- Dynamic Libraries, USS, DB2 coprocessor…

VS COBOL II (Ann: 1980s)
- *New technology base; Not compatible with OS/VS COBOL*
- ISO COBOL 85 standard
- 31 bit addressing, Reentrancy, DBCS…

OS/VS COBOL (Ann: 1960s)
- ISO COBOL 68 & 74 standards

Enterprise COBOL for z/OS V5.1
GA: Jun 21, 2013

- New optimization framework
  - Delivered greater than 10% performance improvement over Enterprise COBOL v4 for well structured, CPU-intensive batch applications[1]
  - Lays solid groundwork for delivering release-to-release performance improvement roadmap for COBOL on z Systems
- New COBOL Runtime
- New Object and Debugging formats
- New programming features

Migrating to COBOL V5 requires advanced planning and more work than migrating for COBOL V3 to V4

[1] Results are based on an internal compute-intensive test suite. Performance results from other applications may vary.

Enterprise COBOL for z/OS V5.1
GA: Jun 21, 2013

- Maintains source and binary compatibility
    - Correct COBOL programs will compile and execute without changes and produce the same results
    - "Old" and "new" code can be mixed within an application
    - Removed some old language extensions and options

- Supports the ecosystem of programming tools supplied by IBM and ISVs.

[1] Results are based on an internal compute-intensive test suite. Performance results from other applications may vary.

Things you should know when upgrading to V5.1

- H/W and S/W Pre-requisites

- PDSE load libraries

- Dataset and Memory Requirements & SMF

- Old "OS/VS COBOL" and "VS COBOL II NORES" Code

References:

developerWorks Article: Prepare to upgrade to Enterprise COBOL for z/OS V5
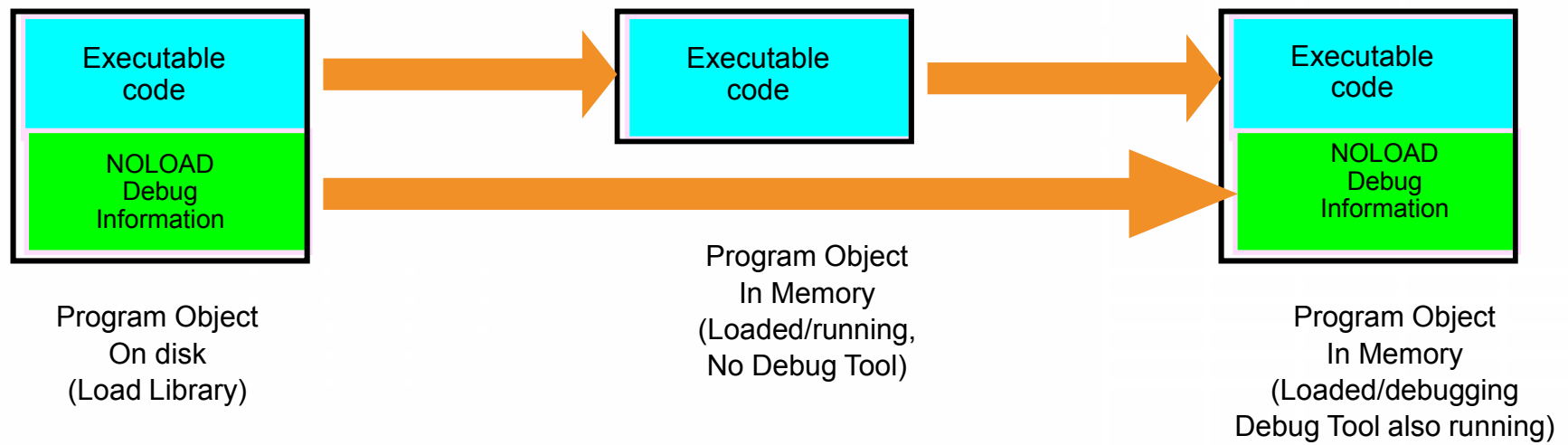
Enterprise COBOL for z/OS V5.1.1 Migration Guide

Pre-requisites

- Hardware
  - Minimum hardware requirement: z990/z890.
  - Programs compiled with the new COBOL v5.1 compiler will not run on machines that pre-date z990/z890

- Software
  - z/OS V1R13 or later
  - CICS Transaction Server, V3 or later
  - IBM DB2 V9 or later
  - IBM IMS V11 or later
  - PD tools V12 or later (Debug Tool, Fault Analyzer Application Performance Analyzer (V13))
  - Rational Developer for System z V9

- Required PTFs
  - Use *SMP/E FIXCAT* to identify all required service PTFs
  - Required PTFs are not listed in the COBOL for z/OS V5.1 Migration Guide or PSP buckets

PDSE

- Enterprise COBOL V5 executable are Program Objects
  - Must reside in PDSE datasets
  - Must move PDS datasets for load modules to PDSE dataset load before running COBOL V5 programs

- Requires some advanced planning if using PDS load libraries
  - Moving load modules from PDS to PSDE is fairly straight forward

Executable code	Executable code	Executable code
NOLOAD Debug Information		NOLOAD Debug Information

Program Object
On disk
(Load Library)

Program Object
In Memory
(Loaded/running,
No Debug Tool)

Program Object
In Memory
(Loaded/debugging
Debug Tool also running)

Dataset and Memory Requirements & SMF

- Dataset and Memory
  - COBOL V5 requires more datasets and memory at compile time
    - Performs more aggressive optimizations to increase the performance of COBOL programs.
    - In addition to the work datasets SYSUT1-SYSUT7, Enterprise COBOL v5.1 requires SYSUT8-SYSUT15 and SYSMDECK
    - Recommended region size: 200MB

- SMF
  - Automatically generates SMF89 records
  - Used by the Sub-capacity Reporting Tool (SCRT) V21.2.0.
  - No need to manually track usage of the COBOL compiler.

Old COBOL Code

- LE
  - Complete migration to LE

- OS/VS COBOL
  - Cannot mix OS/VS COBOL programs with programs compiled with Enterprise COBOL V5.1
  - Must migrate to Enterprise COBOL

- VS COBOL II NORES
  - Recompile VS COBOL II NORES programs with Enterprise COBOL to work with V5.1

Enterprise COBOL for z/OS V5.1 – Since GA

Continuously Delivered Improvements based on customers' feedback via PTF stream

- COBOL V3/V4 features to improve migration
    - AMODE 24, XMLPARSE(COMPAT), VLR(COMPAT), MAP(HEX)
- New features
    - Support for IMS V13 EXEC SQLIMS
- Performance Improvements
    - Execution Performance: Working Storage, Procedure Pointer calls Unstring…
    - Compile time Performance

Complete Fix list for Enterprise COBOL for z/OShttp://www-01.ibm.com/support/docview.wss?uid=swg27041164

Enterprise COBOL for z/OS V5.2
Ann: Jan 14, 2015; GA Feb 27, 2015

2nd release of the new COBOL roadmap

- Provides easy migration from COBOL V5.1

- Includes all enhancements delivered in COBOL V5.1 PTFs

  – Restored migration features, performance features, new programing features

- Provide Day 1 support for z13 processor

  – Instruction scheduler tuned to new micro architecture at ARCH 11

  – Expanded use of Decimal Floating Point for PACKED-DECIMAL data

  – Take advantage of Vector Facility for INSPECT TALLYING or REPLACING statements

- Improve Application Performance

  – Well-structured, compute intensive batch applications running on z13 (compiled with Enterprise COBOL V5.2) have shown CPU time reduction of up to 14% over the same applications running on zEC12 (compiled with the GA release of Enterprise COBOL V5.1) [1]

[1] Results are based on an internal compute-intensive test suite. Performance results from other applications may vary.

Exploiting z/Architecture with ARCH option

z10

ARCH(8)

- Decimal Floating Point (DFP) unit

- Wider immediate data

z196/z114

ARCH(9)

- Distinct Operand

- Conditional Load

- High Word Instructions

zEC12/zBC12

ARCH(10)

- Decimal-Floating-Point Zoned-Conversion Facility

Z13

ARCH(11)

- Decimal-Floating-Point Packed-Conversion Facility

- Vector Facility

NEW

ARCH 6 is no longer supported in COBOL V5.2

Default ARCH level is now ARCH 7 - IBM System z9EC (2094-xxx models) IBM System z9 BC (2096-xxx models)

IBM

Exploiting z/Architecture – Example
Decimal Divide Where Operands Exceed Packed Decimal Hardware Limits

```
1   z14v2 pic s9(14)v9(2)
1   z13v2 pic s9(13)v9(2)
. . .
Compute z14v2 = z14v2 / z13v2
```

COBOL V4 – Zoned & Packed decimal arithmetic that cannot fit within hardware instruction limits are implemented by generating a call to a divide or multiply routine in the runtime.

```
V4
•   Calls out to library routine
•   Runtime path length is > 100 instructions

PACK   344(9,13),0(16,2)
PACK   360(16,13),16(15,2)
MVC    376(32,13),59(10)
MVC    398(9,13),344(13)
NI     406(13),X'F0'
MVN    407(1,13),352(13)
L      3,92(0,9)
L      15,180(0,3)
LA     1,146(0,10)
BASR   14,15
NI     431(13),X'0F'
ZAP    431(9,13),431(9,13)
UNPK   0(16,2),431(9,13)
```

COBOL V5 - Packed and Zoned decimal arithmetic that does not fit in the hardware instruction limits are converted to DFP for z10 and above. Arithmetic operations are done in DFP unit, then converted back to the original packed/zoned type.

```
V5
•   Inlined with 6 instructions
•   CDZT/CZDT are new EC12 instructions to
    convert between zoned and DFP types
•   ARCH (10)

CDZT    FP0,152(16,R8),0x8
CDZT    FP1,168(15,R8),0x8
SLDT    FP0,FP2,2
DDTR    FP0,FP0,FP1
FIDTR   FP0,9,FP0
CZDT    FP0,152(16,R8),0x9
```

Performance Comparison

```
Timing (100 million in a loop)
V5 : 1.08 cpu seconds
V4 : 4.81 cpu seconds
```

Advanced Optimizations

- Provide multiple levels of optimization
- Debugging of optimized code is supported with OPT + TEST options

OPT(0)
- Minimum Optimization

OPT(1)
- Increased Optimization
  - e.g. Inline PERFORM statement
  - Commoning sub-expressions in a block
  - Sequential constant store simplification…

OPT(2)
- Maximum Optimization
  - e.g. Eliminating a stored value that is never re-used anywhere in the program
  - Global view of register assignment…
  - Instruction scheduling to exploit micro-architecture…

Longer compile time
Reduced debugging
Faster Executing Code

Advanced Optimization – Example
Instruction Scheduling for Performance

```
1 z7v2a pic s9(7)v9(2).
1 z7v2b pic s9(7)v9(2).
1 z7v2c pic s9(7)v9(2).
...
ADD 1 TO z7v2a z7v2b z7v2c
```

COBOL V4 – each PACK/AP/ZAP instruction group for each receiver is generated in order.

V4 – OPTIMIZE
- *Instructions appear in original order and subject to hardware read after write penalties*

```
PACK   344(5,13),0(9,2)
AP     344(5,13),51(2,10)
ZAP    344(5,13),344(5,13)
UNPK   0(9,2),344(5,13)
PACK   344(5,13),16(9,2)
AP     344(5,13),51(2,10)
ZAP    344(5,13),344(5,13)
UNPK   16(9,2),344(5,13)
PACK   344(5,13),32(9,2)
AP     344(5,13),51(2,10)
ZAP    344(5,13),344(5,13)
UNPK   32(9,2),344(5,13)
```

COBOL V5 - at OPT(2) low level instruction scheduling is performed to reduce data dependencies, avoid hardware penalties and to best take advantage of the micro-architecture.

V5 – OPT(2)
- *Independent operations are grouped to reduce read after write hardware penalties*
- *ARCH(8)*

```
PACK   352(5,R13),152(9,R8)
PACK   344(5,R13),168(9,R8)
PACK   336(5,R13),184(9,R8)
AP     352(5,R13),416(2,R3)
AP     344(5,R13),416(2,R3)
AP     336(5,R13),416(2,R3)
ZAP    352(5,R13),352(5,R13)
ZAP    344(5,R13),344(5,R13)
ZAP    336(5,R13),336(5,R13)
UNPK   152(9,R8),352(5,R13)
UNPK   168(9,R8),344(5,R13)
UNPK   184(9,R8),336(5,R13)
```

Performance Comparison
Timing – (100 million in a loop)
```
V5 : 2.35 cpu seconds
V4 : 2.50 cpu seconds
```

Enterprise COBOL for z/OS V5.2 - New features

- Access to z/OS JSON services from COBOL.
  - Provide mobile (i.e. system of engagement) applications easy access to data and the processing they need from business critical enterprise (i.e. system of record) applications written in COBOL

- Improved XML GENERATE
  - More powerful SUPPRESS capabilities

- RULES option to help programmers write better code

- COPYRIGHT and SERVICE options to better manage applications

- New VOLATILE attribute and SERVICE LABEL functionality
  - Enable full optimization of User-Written condition handlers

- New format of the SORT statement, the table SORT statement (COBOL 2002 feature)
  - Arranges table elements in a user-specified sequence.

- Accessing VSAM data sets with extended addressability attribute is now supported

Developer Trial

- Zero cost evaluation license for 90 days
  - Does not initiate Single Version Charging (SVC)

- Assess the value that could be gained from upgrading to Enterprise COBOL V5.2

- Offer same functionalities as Enterprise COBOL for z/OS V5.2
  - Code compiled with Enterprise COBOL Developer Trial cannot be used for production

- Available as standard offering through ShopzSeries on March 27, 2015
  - Contact your IBM representative for ordering assistance

http://www-03.ibm.com/software/products/ph/en/enterprise-cobol-developer-trial-for-zos

New System z Compilers

- Enterprise COBOL for z/OS v5.2

- XL C/C++ Compiler for Linux on z Systems

- Enterprise PL/I for z/OS v4.5

- z/OS XL C/C++ V2R1M1

Linux for z Systems

- Growing rapidly
  - Installed IFL MIPS increased 12% from 4Q'13 to 4Q'14.
  - As of 4Q14…
    - 27.3% of Total Installed MIPS run Linux
    - 39% of z Systems Customers have IFL's

- Only choice for C/C++ developers is GNU C/C++

- IBM middleware, Business Intelligence & Analytic, and ISV workloads are driving demand to increase performance

XL C/C++ for Linux on z Systems
Ann: Jan 14, 2015; GA: Feb 16,2015

- Based on new technologies
  - Clang front end
    - Strong C/C++ language support (e.g. C11, C++11, C++14)
    - GNU C/C++ language and option compatibility
    - Allows Easy migration from distributed Linux systems to Linux for z Systems
  - IBM optimization technology shipped in Enterprise COBOL and IBM Java
- Ships with high performance Math Libraries tuned for z Systems
  - MASS and ATLAS
  - Provide elemental and basic linear algebra functions to simplify coding and improve application performance
- Runs on RHEL (6 & 7); and SLES (11 & 12)

XL C/C++ for Linux on z Systems

- Provides significant performance advantage over GNU C/C++
  - Advanced optimization and Full z/Architecture exploitation
  - CPU intensive applications compiled with XL C/C++ for Linux on z Systems V1.1 have shown up to 10% performance improvement over the same applications compiled with GNU C/C++ V4.4 [1]
    - 7% performance improvement over the same applications compiled with GNU C/C++ V4.7[1].
  - World class service and support by IBM

For more information, visit http://www.ibm.com/software/products/en/czlinux

[1] Results are based on an internal compute-intensive test suite. Performance results from other applications may vary.

New System z Compilers

- Enterprise COBOL for z/OS v5.2

- XL C/C++ Compiler for Linux on z Systems

- Enterprise PL/I for z/OS v4.5

- z/OS XL C/C++ V2R1M1

Enterprise PL/I

- Strategic Programming Language
  - Significant use in business applications but also in some scientific and engineering application

- Advanced optimization technology
  - Shares optimizing back-end technology with z/OS XL C/C++
    - Timely delivery of leading edge optimization and hardware exploitation to PL/I customers

- Time proven
  - First Enterprise PL/I product released in 2001 (Enterprise PL/I for z/OS and OS/390 v3.1)
  - Latest release of Enterprise PL/I for z/OS (v4.5) is based on same architecture
    - Provides easy migration

- Shipped new release every year since 1999
  - Addressed customer requirements
  - Improved optimization technology, z/Architecture exploitation, usability, middleware support, and application modernization features.

Enterprise PL/I for z/OS v4.5
Ann: Jan 14, 2015; GA: Feb 27,2015

- Advanced optimization and full z13 exploitation
  - New optimizations to Improve performance for packed decimal
  - Use of SIMD instructions to improve code for SEARCH and VERIFY
  - Inlining of MOD and REM of large packed decimal and multiply

- Provides significant performance improvements over Enterprise PL/I V4.4
  - Compute intensive applications running on z13 (compiled with Enterprise PL/I V4.5) have shown CPU time reduction of up to 17% over the same applications running on zEC12 (compiled with Enterprise PL/I V4.4). Performance Improvement

- Addressed 28 RFEs

Enterprise PL/I for z/OS v4.5 New Features

- Ships a JSON parser
  - Support Parse, Generate, and Validate
  - Works with z/OS JSON services
  - Allows enterprise (i.e. system of record) applications written in PL/I to be extended to handle data access and processing requests from mobile (i.e. system of engagement) applications

- Improved middleware support
  - Faster code for CICS calls
  - Support for named constants as SQL host variables
  - Improved error reporting for invalid SQL statements

- New productivity features
  - Extend size of strings from 32K to 128M
  - New INLIST and BETWEEN built-in functions
  - New built-in functions to generate the store-clock hardware instructions
  - New REINIT statement
  - Support for apostrophes in PICTURE variables

For more information, visit http://www-03.ibm.com/software/products/en/plizos

New System z Compilers

- Enterprise COBOL for z/OS v5.2

- XL C/C++ Compiler for Linux on z Systems

- Enterprise PL/I for z/OS v4.5

- z/OS XL C/C++ V2R1M1

IBM z/OS XL C/C++

- Optionally priced feature of z/OS
  - Enables development of high performing business applications, system programs and low level C applications

- IBM has been delivering leading edge C/C++ compilers on z/OS for over 20 years
  - Every release sets new standard for performance
  - Includes advanced optimization technology originally designed for HPC applications, and innovations to improve programmer productivity
  - Improves support for C and C++ language standards

- Provides system programming capabilities with Metal C option
  - Allows developers to use C syntax to develop system programs and low level free standing applications on z/OS without coding in HLASM
    - Significantly shortens the learning curve
    - Leverage advanced optimization technology to generate high performance optimized code

z/OS XL C/C++ V2R1M1
Ann: Jan 14, 2015; GA: Feb 27, 2015

- Web deliverable for z/OS 2.1
    - Provide Day 1 support for z13
    - Target clients running z/OS V2.1 on new z13 Systems
    - Replaces z/OS XL C/C++ V2.1 after installation

- Support for Vector/SIMD
    - Compile option, datatype, and built-in functions

- Ships with High performance Math Libraries tuned for z13
    - MASS Mathematical Acceleration Subsystem) library providing scalar, vector, and SIMD mathematical elemental functions
    - ATLAS (Automatically Tuned Linear Algebra Software) library providing basic linear algebra functions

z/OS XL C/C++ V2R1M1

- New Programming Features
    - New support for inline assembler (GNU compatibility)
    - New Debug support for Vector/SIMD data type
    - Header file cache optimization

- Provides significant performance improvements over z/OS XL C/C++ V2R1.
    - CPU intensive applications running on z13 (compiled with z/OS XL C/C++ V2R1M1) have shown up to 17% throughput improvement over the same applications running on zEC12 (compiled with z/OS XL C/C++ V2R1)

For more information, visit http://www-03.ibm.com/software/products/en/czos

Metal C Option

Compile with XL C compiler

Invoke EDCC proc in JCL:
 METAL option
 GENASM option

Invoke xlc in USS:
 -qMETAL
 -S

C source → HLASM source

Assemble with HLASM

Invoke ASMA90 in batch
Invoke as in USS

Object File

Bind object(s)

Invoke binder in JCL:
 ENTRY control statements to specify name of entry point

Invoke ld in USS:
 -e option to specify name of entry point

Executable

- Generates optimized ASM source code that is Language Environment independent

- Interoperates with existing HLASM programs

    – Uses MVS™ linkage conventions

- Supports AMODE 64 (allow mixing of AMODE 31 & AMODE 64 code)

- Provides a subset of C library functions.

Benefits of Metal C

- Provides high level language alternative to writing the program in assembly language.

- Allows developers to use C syntax to develop system programs
  - Significantly shortens the learning curve
  - No need to manage use of registers and developing the correct assembly instruction sequences

- Allows developers to develop low level free standing applications on z/OS
  - Supports embedding of HLASM source within C statements
  - Provide direct access to z/OS System services

- Leverage advanced optimization technology to generate high performance optimized code

- Develop the application once
  - Recompile to optimize for new z/Architecture

www.ibm.com/software/rational