System z Compilers and Business Technology Evolution

By Anne MacFarland

August 2012

In Lewis Carroll's Alice in Wonderland, the tea table used by the Mad Hatter, March Hare and Dormouse has enough room that, when things get messy, they can move on to new place settings. Most of us have found that this strategy only works in Wonderland. It works badly in business, where, even in chaotic markets, the experience of the past informs the opportunities of the present and the strategies for the future. It works even worse when the table settings are enterprise software. The exacting expectations of users (core business applications and end users) will require extensive customization and testing. Crafting an evolutionary path to a desired state is often the more prudent way forward. Four IBM customers share their experiences in this paper – on different platforms and in different situations. What they have in common is a focus on evolving their entire software stack, including the compilers, to optimize their IT environments.

Businesses focus on the next deal in much the way that sharks focus on their next meal. Today, many CIOs fret about how to generate the agility needed to support next-deal ambitions with existing (legacy) applications. In rapidly reconfiguring markets (as many are), the ability to evolve in ways that fully leverage existing infrastructure has huge business value. In data centers, this is usually achieved by a coordinated ballet of processors and the instruction sets they hold, middleware, and the compilers that translate programming into executable code.

Optimization of existing assets depends not just on the asset itself, but on how it has been maintained. With cars, for example, maintenance is a matter of mundane things like checking the tire pressure. With a business's technology assets, it is a matter of updating to get new features and higher performance, of extending monitoring to address new concerns, and of formulating and delivering on larger, longer term strategies to support business agility. When business technology assets are properly kept up, they become incredibly strategic because they fully reflect the specifics of business operations and also leverage the evolution built into the latest versions of technology assets.

The Tangible Benefits of Using the Latest Compilers

IBM's latest compilers and middleware (CICS, DB2, IMS) fully leverage new instructions in the latest System z processors. Today, core business information and applications are kept on System z because of the scale at which business critical applications and data are used, and for security reasons. A Foresights Software Survey published in Q4, 2010, showed that modernizing legacy applications is the highest priority item for all software leaders who participated in the study. Compilers increase the efficiency of business logic and provide features that enable modernization of applications. The latest versions of

middleware were designed to increase the efficiency of the application tiers that support business operations, collaboration, self-service functionality, and all the modern elements that help a business thrive.

IBM has found that keeping mainframe processors, compilers and middleware current has considerable value. IBM customers have experienced a 20-60% improvement in application performance with the latest compilers.[1] By leveraging modern application development tooling, companies have reported a 22-37% improvement in developer productivity[2]. Companies who have deployed IBM Rational's collaborative application lifecycle solution saw significant improvements in their team productivity and a 15-20% decrease in development cycle time[3]. These are impressive numbers.

IBM System z and the z/OS operating system provide strong backward compatibility support. This means that applications will continue to run even if applications are not upgraded to use the latest version of the middleware software or re-compiled with the latest version of the compilers. However, these applications will not perform as well as they could. And in an environment where 80%+ system utilization is the norm, that's a lot of performance waste, and a lot of hardware and middleware functionality that are not accessible. Remaining on out-of-date compilers impairs the ROI of your mainframe budget.

The C/C++ Customer Experience

Joe Devlin, Managing Director, R&D, Rocket Software, is particularly pleased with the way the Metal C feature of the IBM z/OS® XL C/C++ compiler significantly increased efficiency, reduced development time and enabled his company to leverage their C programming skills in developing low level applications on System z.

Rocket Software is a global software development organization that produces enterprise infrastructure products in the areas of business intelligence, storage, networks, terminal emulation, integration, security and databases. Their products mainly run on IBM System z. The company also licenses them to original equipment manufacturers (OEMs).

IBM z/OS XL C/C++ v1.13 Compiler

The benchmarks tell the story.

The IBM z/OS XL C/C++ v1.13 compiler achieved a 4% increase in the compute-intensive integer benchmark suite and a 7% increase in the compute-intensive floating point benchmark suite, when compared to the XL C/C++ v1.12 compiler.

The compiler is optimized for the target architecture without requiring source code modification. It has ISA exploitation and performance tuning. It understands and applies the instruction and scheduling trade-offs on the specific hardware being targeted. Continued attention to source and binary compatibility enables more straightforward porting of C/C++ applications to the z/OS platform.

METAL C functionality allows programmers to avoid dealing with the HLASM Assembler language. METAL C code looks and feels like C/C++, and is usable by those with C/C++ skills. This addresses, and may solve, a skills issue.

[1] Results are based on a compute-intensive integer benchmark suite compiled with z/C/C++ v1.9 executing on a System z10 and compared to the same benchmark compiled with z/OS C/C++ v1.12 executing on System zEnterprise 196.

[2] "2010 IBM Rational Benchmarking study on IDE efficiency and developer productivity

[3] IBM customer case study

Joe is responsible for programming and developing operational and monitoring software for performance and capacity management at Rocket Software. His challenge is to find a tool that can increase efficiency and time to market for their software products for System z. Development of mainframe software often involves integrating with existing applications, many of which are large and complex. Therefore, it's essential to use the right language for the job. High-level programming languages aren't always useful for low-level application development. Rocket Software takes on new projects and products constantly and needs to have the right tools to support their development efforts and keep up with their growth. They started to use the Metal C feature of the IBM z/OS® XL C/C++ compiler on a test project. The results were so successful that the company decided to use it to develop their software products. They were able to build a compliance product in four months. In the past, such a project might have taken two to three times as long to complete. With Metal C, developers can also take advantage of advanced optimization technology in the z/OS XL C/C++ compiler to produce high-performance code that works seamlessly with code written in IBM High-Level Assembler language (HLASM). C programmers, who do have extensive mainframe backgrounds, can also work on these projects, significantly broadening the company's resource pool. Since this project, Rocket Software has continued to train and add developers to the group using the Metal C feature. As Joe puts it, "Metal C in z/OS XL C/C++ is yet another powerful tool helping to turn the economics of System z software development into a completely new equation."

> Metal C in XL C/C++ optimizes system programs for new zArchitecture without changes to the source code!

> The IBM z/OS XL C/C++ v1.13 Compiler has an advanced high-level optimizer which enables powerful capabilities such as loop, whole program and profile-directed optimizations.

The Enterprise COBOL Customer Experience

Company B is a very large insurance company that prefers to remain anonymous. An Infrastructure Planning Engineer from this organization has compiled and shared a long list of tangible benefits from leveraging System z enhancements and the current version of the Enterprise COBOL compiler. His focus is at the application and middleware level – and on implementing new mainframe functionality to support his organization's business goals. He manages a large enterprise Java and COBOL shop with multiple simultaneous short- and long-term projects. This company's primary focus is on cost savings (often driven by reductions in MIPS use) and process improvements.

The Planning Engineer says that IBM's Enterprise COBOL v4.2 has enabled them to eliminate post processing on XML data that is to be exported via a Web Service. This saved them both time and energy costs. Both CICS and IMS were endowed with Web Service functionality, and Web Services are heavily used in his applications. The engineer remarks that CICS supports top-down Web Services while IMS remains more bottom up in orientation. He prefers the top-down, modeled approach to Web Services in both CICS and IMS. His frequent participation in beta tests and early releases enables him to test new features in both compilers and middleware and provide timely feedback to IBM.

Company B's Planning Engineer is also pleased with Enterprise COBOL v4.2's support for XML System Service (z/OS XML) parser. This feature has already yielded significant savings for his organization. z/OS XML is a system level XML parser that is intended to be used by system components, middleware and applications. z/OS XML also allows XML documents to be offloaded to a zAAP specialty engine for processing, so companies can reduce their MIPS cost.

IBM Enterprise COBOL for z/OS v4.2 Compiler

Version 4.2 supports significantly improved UNICODE performance; provides better integration of existing applications with Web Services; provides an integrated CICS and SQL pre-processor; and supports the latest CICS and DB2 features.

Java interoperability with COBOL is achieved through development using object-oriented COBOL syntax. This enables integration of COBOL applications with Web applications. COBOL applications can access enterprise beans running on *WebSphere Application Server* or J2EE-compliant Enterprise Java Bean Servers. Version 4.2 enables COBOL applications to work with Java 5 and Java 6 SDKs.

The support for high speed XML parsing built into the v4.2 COBOL compiler is particularly useful in an era where many kinds of data are being used in new ways. Encoding in UTF-8, UTF-16 and EBCDIC codepages are supported, as is the offloading of XML parsing to the zAAP processor.

Chargeback is a way of life at Company B, and the Planning Engineer is delighted that CICS v4.2 supports more granular chargeback. His environment is still 60-70% custom code but there is a careful transition to new packages, and he notes that these packages often generate more (not fewer) IMS and CICS transactions. One strategy to combat the growth of MIPs charges is to build Web Services that generate XML to prune the data transfers between the packages. He likes the COBOL support in the IBM Fault Analyzer that now has new functions to deal with storage violations. He strongly recommends to other companies that they keep current on IBM's Debug Tool that enables "jump-to" project debugging.

Overall, he is very pleased with the breadth of new compiler functionality that can support major projects such as standardization of the Enterprise COBOL Code checking process, and propagating a DB2 and CICS integrated co-processor in the Enterprise COBOL compiler. More recently, he has begun a modernization project that requires COBOL programs to interoperate with Java programs. He used Rational Development and Test Environment for System z, RD&T, (formerly known as RDz Unit Test), which provides a local test environment for unit testing System z applications on a PC running Linux, to design and test the application. RD&T gave flexibility to development teams while providing MIPS savings. The COBOL modernization project will be well-instrumented and documented. They plan to publish time and cost improvements when the project is completed. Dramatic savings are expected, and there is a strong foundation for continuing to move in this direction in the future.

Two PL/I Customer Experiences

Company C also wishes to remain anonymous, but they shared their PL/I experiences with us. They are a very large financial institution that uses a bevy of zEnterprise 196 servers to support their global operations. Their core applications are written in PL/I and COBOL. They also use C/C++ and Java for some applications.

This company is committed to PL/I because they have millions of lines of well-performing PL/I code that their core business relies on. New modules are mostly developed in Enterprise COBOL. They are currently upgrading from Enterprise v3.9 to v4.2 to improve application performance on their new zEnterprise 196 hardware. Although the project is still in progress, their 500 mainframe developers are able to optimize PL/I code in remarkable ways. One of their teams expects to cut their MIPS usage by 50% with the latest compiler optimizations.

IBM Enterprise PL/I for z/OS v4.2

PL/I is a strategic language originally designed for scientific and engineering applications, but it has been well adapted for business applications. IBM's first Enterprise PL/I compiler was released in 2001, and it has been updated every year since. Version 4.2, the current version, shares optimizing back-end technology with the IBM z/OS XL C/C++ compiler. Enterprise PL/I for z/OS also provides a migration path from PL/I for MVS and VM Compilers.

The latest PL/I compiler delivers some significant improvements in functionality beyond the traditional performance boost (up to 10%), and enhanced support of XML, Web Services and new processor instructions. A smaller, faster and more powerful SQL pre-processor is 8x smaller and 40% faster in processing SQL source data, which will be significant for certain workloads. In addition, the integration of CICS and SQL pre-processors supporting the latest versions of CICS and DB2 will simplify application modernization operations.

This company finds IBM's support for compiler-related issues to be excellent. They respect the warnings generated by the PL/I compiler, such as, for example, one about CICS and static variables, and the remedial recommendations that accompany them. They urge all programmers to follow IBM's advice. Their development teams are also adopting Rational Team Concert (RTC), a collaborative application lifecycle management solution. Rational Asset Analyzer (RAA) provides in-depth information on the structure of the applications and maps dependencies, setting the stage for efficient modernization. They are also working on modernizing legacy code that would benefit from the newer compiler's optimization features.

Company D is a major bank in northern Europe with 30,000 MIPS in production, which also wishes to remain anonymous. Their environment is 70% mainframe and 30% PC networks. Their mainframe portfolio includes PL/I, COBOL, EGL/VAG (VisualAge Generator), HLASM (High Level Assembler), and C/C++. A few years ago, they migrated their core PL/I applications (40+ million lines of code) from a legacy PL/I compiler to Enterprise PL/I v3. Moving to Enterprise PL/I solved storage contention issues between their many CICS load modules, and enabled them to reduce the size of PL/I load modules and CICS MIPS consumption. They approached this modernization project very carefully because many of their programs have not been touched in years.

They divided the project into 3 phases. The first phase focused on fixing compatibility issues between the old and new PL/I compilers. The second phase targeted interoperability between the old and new compilers, since not all programs can be fully migrated to the new compiler. Rational Asset Analyzer (RAA) was instrumental in this phase. It helped the team scope the impact of their code changes. The team used this information to mitigate their risk. The last phase focused on implementing line items with new features supported in Enterprise PL/I v3. They were able to successfully complete this project on time and within budget. Recently, they have upgraded to Enterprise PL/I v4. They recompiled their most frequently used programs in their applications and achieved significant MIPS savings. This upgrade

went smoothly without any compatibility issues, and many of the programs that were recompiled have significantly reduced storage access and CPU consumption by up to 50%.

Interview Take-Aways

These companies all had very different kinds of challenges – but in every case, use of the latest compilers delivered significant and enduring value. Data centers must blend both tactical and strategic application development as part of their operational strategy. The application rejuvenation process has been greatly enhanced by IBM innovations such as the COBOL-Java integration, XML support, and the METAL C. The evolutionary paths of the PL/I users were quite different but both addressed large challenges in a sensible fashion and took advantage of the latest enhancements in PL/I compilers to improve application performance and improve their return-on-investment of their System z hardware.

Improved Programming Support for Middleware

New compilers provide improved programming support for new features in middleware (i.e., CICS, DB2 and IMS). Enterprise COBOL v4.2 provides new SQL data types that were first introduced in DB2 v9. Enterprise PL/I v4.2 provides improved support for DB2 applications with multi-row insert and multi-row fetch. PL/I v4.2 improved performance of processing SQL statements by up to 40% with up to 8X less memory footprint. In addition, COBOL and PL/I have improved problem determination support and can process XML documents received from IBM middleware.

Rational Broadens the Horizon

IBM Rational offers a broad set of development and application lifecycle management solutions for System z. IBM Rational Developer for System z software (RDz) enables development of COBOL, PL/I, C++, HLASM (High Level Assembler), and Java applications for batch, CICS, IMS, and DB2. It also helps developers to rapidly create web applications that interoperate with CICS, and IMS transaction applications and WebSphere environments.

IBM Rational Team Concert (RTC) is an application lifecycle management solution that improves collaboration between members of development teams to maximize business agility. It provides functions for task tracking, source control, agile planning, managing governance, and continuous builds.

Companies B and C are currently running pilot projects in their respective organizations for RDz and RTC. Their early results are very positive. Both companies are currently planning to increase their use of these tools within their respective development communities.

Rational Development and Test Environment for System z, (RD&T), which was formerly named Rational Developer for System z Unit Test (RDz-UT), provides a low cost test environment for System z applications. It creates a personal z/OS environment on an x86 desktop or a shared server, reducing MIPS usage on System z for developing and testing applications. It runs with full System z fidelity. Final build as well as production testing must still be done on a production System z server. Company B used RD&T in their modernization project. It provided them with increased flexibility and MIPS savings.

Rational Asset Analyzer (RAA) was an essential tool used by Companies C and D. It provided insight into

the structure of their large PL/I applications. Understanding the relationships of existing applications helped the companies better plan their migration efforts, and reduced the overall risk of their projects. RAA also enabled them to focus on comprehensive testing, which helped both companies complete their projects on time and within budget.

A Look into the Future

IBM Fellow Kevin Stoodley sees System z compilers as an exciting space to work in because of the breadth of the benefits that modern compilers now offer. An ongoing and significant story is the modernization of COBOL itself. It is a preparatory step to the larger vision of a common back end for all IBM compilers on System z. The new COBOL infrastructure will enable full exploitation of current and future z/Architecture and provides a solid foundation to support 64-bit application development. A managed beta of the new COBOL compiler currently in progress allows companies that use System z to more fully understand the benefits of upcoming compiler releases and contribute to the development process.

In the very near future, optimizations will address the whole program scope and optimization decisions will be guided by performance profiles. Some of these features are currently available in C/C++ compilers. Stoodley believes that COBOL and PL/I could follow a similar evolutionary path. The common optimization technology that Stoodley envisions will expose new potentials for efficiency and agility for all supported programming languages.

Stoodley believes that Metal C will continue to leverage advanced optimization features of the C/C++ compiler to help developers deploy optimized, low-level applications, as well as system programs to all z/Architecture without source code modification. He also mentioned that C/C++, COBOL and PL/I are strategic to the System z platform, and IBM engineers are working hard to bring new innovation and value to these compilers.

IBM System z Evolutionary Opportunities

Businesses with IBM System z have a wide variety of evolutionary paths open to them.

- The rapid evolution of recent IBM System z hardware is significant. z10 started the new accelerated cadence, and z/114 and z/196, with optional z/BX “sidecar” for UNIX and Windows workloads, take it a lot further.
    - New instructions on System z processors coordinate with enhancements in middleware by means of compilers. At IBM, these three efforts have been coordinated over decades.
    - IBM’s active collaboration with its large global customer base allow them to better plan what new functionalities should be evolved and which are the most urgent.
    - A very large variety of partners, ISVs and distributors give opportunities for customization and building in additional business differentiation.

Conclusion

The performance of the technical infrastructure that underpins business operations determines the cadence of business operations and the satisfaction of their customers. Compilers are a key part of this support, not just for their own innovation, but because of how they complement middleware. They are the least expensive element in the software stack and can offer significant MIPS savings in running your applications. Therefore, the best practice is to upgrade compilers when you upgrade System z hardware, z/OS, or middleware (CICS, DB2, IMS). You can leverage programming support provided by compilers for new middleware features, maximize application performance, and improve programmer productivity. Best of all, you can do this with one quality assurance cycle. Upgrading compilers does not require you to recompile your entire application. You need only recompile the parts you modified.

System z was built with the assumption of multi-tenancy, and z/OS has the controls to prioritize the dynamic assignment of resources – it's the secret behind its high utilization rates and low latency. You probably wouldn't trade it for a sprawl of under-utilized assets, like the Mad Hatter's tea table. The more you talk with other companies about their use of mainframe assets, the more opportunities you will find for application optimization.

(excerpts from IBM podcasts on the joint value of compilers and middleware)

Question: Do I need to recompile my application when I upgrade my compilers?

"I understand the concern about recompiling entire applications. Recompiling an entire application is a very time-consuming effort; it can also be quite expensive, but there's really good news here: COBOL and PL/I have maintained very good source and binary compatibility. This means that users only need to recompile the specific files that have been changed and link in the old objects. There is no need to recompile their entire application."

"There are other ways to minimize the risk for upgrading compilers. Scoping the impact of the change you would like to make is very important. The compilers work with IBM's set of advanced programming tools that cover the entire application lifecycle."

"Rational Asset Analyzer collects information about your software assets and shows you the impact of your planned changes. So, if you want to touch a file, it shows you what the impact of that change will be."

"And, Rational Team Concert is a collaborative application lifecycle management tool that helps improve collaboration between developers and the entire change management process."

"These tools can help mitigate risk and prioritize the parts of your application for recompiling or modernizing."

"Upgrading compilers at the same time as you upgrade middleware also has an additional cost benefit: you can leverage the same testing cycle to achieve both upgrades. If you upgrade separately, you'll need to employ a whole separate testing cycle, and we all know that could be very costly."