COBOL Applications: Techniques to Make Them Efficient

Priyal Shah
Bob St. John

Version Date: 9/27/2017

Contents

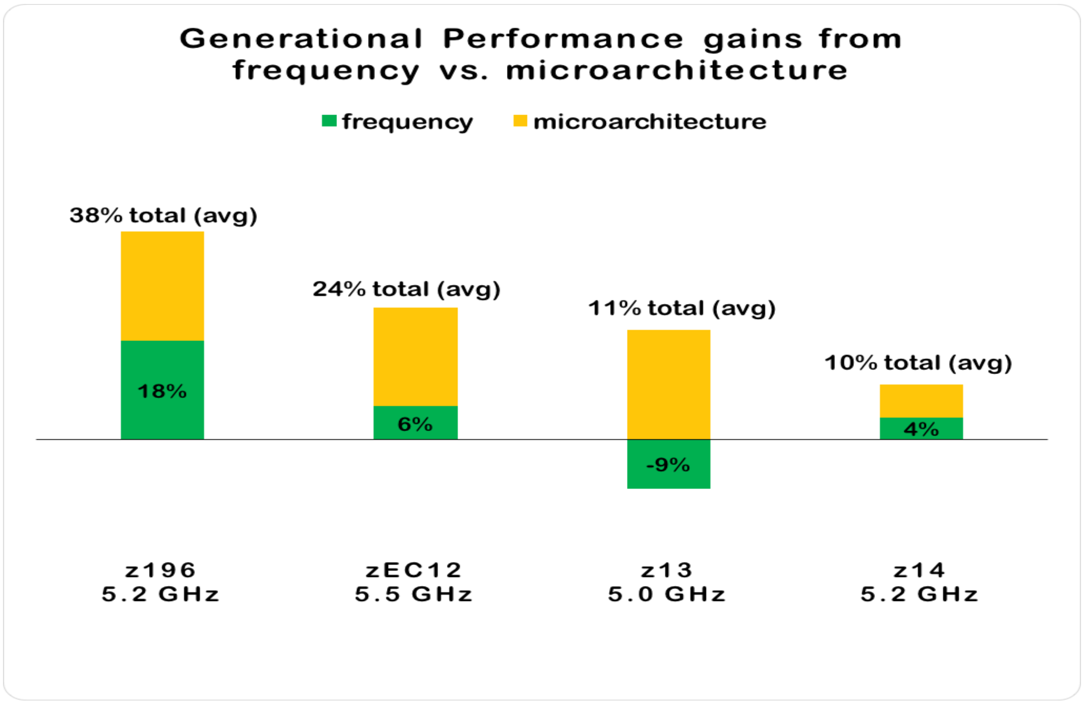
COBOL Applications: Techniques to Make Them Efficient

This document describes why it is important to stay current with the COBOL compiler versions and exploit the use of the highest arch level feasible for your applications in the new era of performance. But your installation may have thousands of COBOL programs, many of which consume very little CPU. Migrating all your applications to the new compiler version might be expensive and time consuming. By targeting the most performance critical sections of your application code for migration first, you can gain most of the performance advantage of the latest compiler technology with a much smaller effort.

Motivation:

With challenges in the micro-processor industry approaching physical limitations, dramatic frequency gains from generation to generation hardware are a thing of past. More and more, modern processor gains are coming from microarchitecture improvements rather than large frequency gains. These microarchitecture improvements include hardware design tradeoffs which can increase performance variability. Hardware design tradeoffs tend to favor optimized, well written code. Older code may not get as much help. Therefore, to get the best performance out of the latest hardware, we recommend performance sensitive code should be compiled with the latest compiler technology and with aggressive optimization.

To understand this fundamental change in processor performance, review the server frequency enhancements made since the z196:



When clock frequency increases, all applications see a performance boost. However, for microarchitecture improvements, a lot of analysis is done to figure out how to make specific constructs faster allowing mainstream workloads run more efficiently. Edge cases where application code hasn't kept up with recommended best practices may not see the full benefits from microarchitecture changes. For future generations, microarchitecture improvements will likely continue to play a major role in overall performance improvements. Improvement in per engine speed are now, and will likely continue to be in future, more reliant on well written code being able to match the server design. For compiled code this is done by staying current with compiler levels for critical applications.

Advanced compiler technology works in concert with the latest hardware to provide optimal application performance. For example, the new Enterprise COBOL v6.2 compiler fully exploits the Vector Packed Decimal Facility, which improves decimal and floating point intensive applications by up to 38% over those compiled with COBOL v6.1(*).

Tip:

Your installation may have thousands of COBOL programs, many of which consume very little CPU. For most of these programs, the benefit of migrating to a new compiler may not be enough to justify the development and testing cost. By targeting the programs using the most CPU, you can get most of the performance benefits while minimizing the cost of migration.

Hotspot Analysis:

z/OS provides a wealth of tools to monitor the CPU consumption of subsystems, applications and transactions. Tools like RMF, OMEGAMON, other vendor products can be used to do this. z/OS also provides SMF records such as record type 110 for CICS, 120.9 for WAS, and 30 for batch. These SMF records can be analyzed directly to determine what applications on your system are consuming the most CPU.

When considering what applications to focus on, pay attention to the frequency with which the program is executed and the time-period it runs. A small CPU improvement in an application executed millions of times can reap significant performance improvements. Likewise, an application which is run once, in the batch window when there is plenty of capacity, may not be a good candidate.

Once you know what applications to focus on, a hotspot analysis tool can be used to find the CPU time of various COBOL modules in these applications. The modules consuming the most CPU will be the best candidates for migration.

Identifying the Hottest Modules:

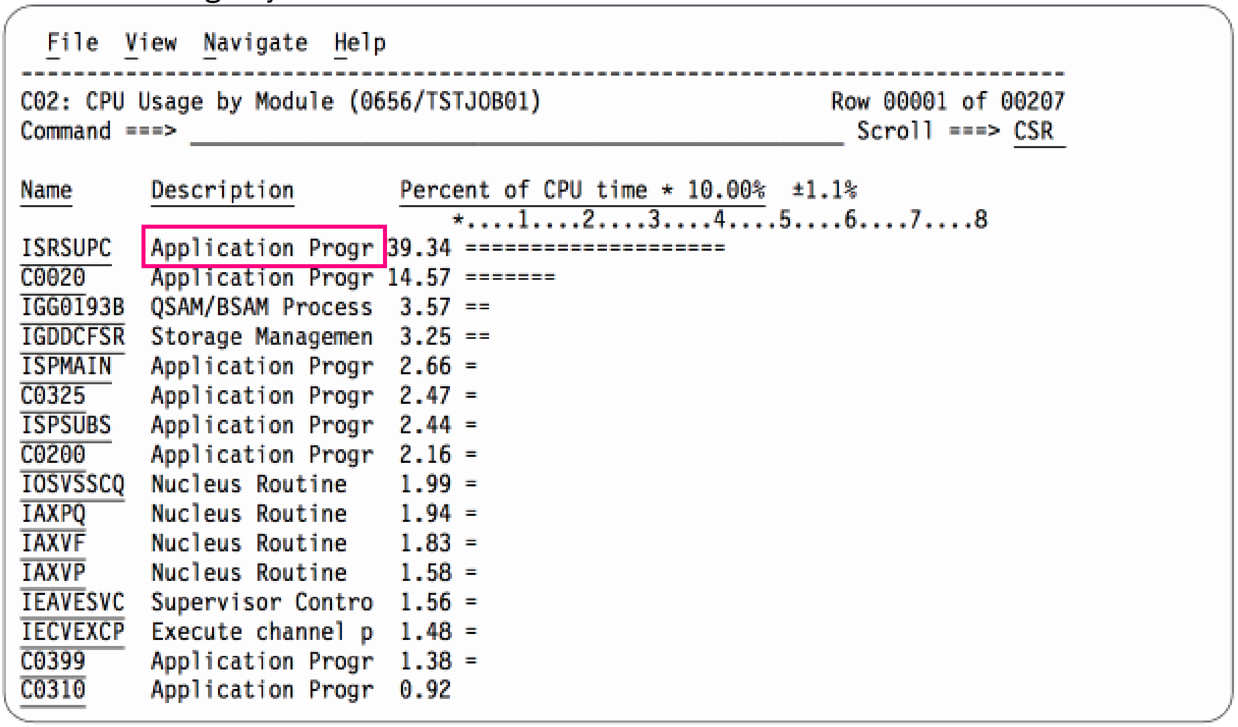
There are multiple hotspot analysis tools which can be used to determine which modules are consuming the most CPU resources. In the description below, we show how to do this analysis using the IBM Application Performance Analyzer for z/OS. A similar approach can be used with any other hotspot analysis tool.

IBM Application Performance Analyzer for z/OS, is an application performance-measurement tool designed for use on IBM z/OS systems. The product’s key function is to measure and report on how system resources are used by applications running in virtually any z/OS address space. APA will identify modules and specific code sections in COBOL programs which are CPU intensive. It does this by sampling the application address space and can be run for any address space, batch or OLTP.

The Cxx set of reports will provide detailed information for every CSECT that is sampled. Most reports can be expanded to multiple levels for greater detail.

Identify top application modules by looking at the C02 report and picking the module names with “Application Progr” in the description field. For these module, filter out hottest COBOL modules. These will be the candidates for recompiling with the latest COBOL compiler version and highest feasible Arch level.

C02 – CPU usage by module:



```
  File  View  Navigate  Help
--------------------------------------------------------------------------------
C02: CPU Usage by Module (0656/TSTJOB01)                       Row 00001 of 00207
Command ===> __________________________________________________  Scroll ===> CSR

Name      Description       Percent of CPU time * 10.00%  ±1.1%
                                *....1....2....3....4....5....6....7....8
ISRSUPC   Application Progr 39.34 ====================
C0020     Application Progr 14.57 =======
IGG0193B  QSAM/BSAM Process  3.57 ==
IGDDCFSR  Storage Managemen  3.25 ==
ISPMAIN   Application Progr  2.66 =
C0325     Application Progr  2.47 =
ISPSUBS   Application Progr  2.44 =
C0200     Application Progr  2.16 =
IOSVSSCQ  Nucleus Routine    1.99 =
IAXPQ     Nucleus Routine    1.94 =
IAXVF     Nucleus Routine    1.83 =
IAXVP     Nucleus Routine    1.58 =
IEAVESVC  Supervisor Contro  1.56 =
IECVEXCP  Execute channel p  1.48 =
C0399     Application Progr  1.38 =
C0310     Application Progr  0.92
```

If interested in further analysis details, the following reports can be useful:

C01 – CPU usage by category. Category is the top level in the hierarchy. CPU consumption is categorized as Application code, SYSTEM, DB2SQL, DATAMG, IMSDLI, ADABAS or NOSYMB

C03-Code Slice report will break things down to a range of instructions (the range, or code slice size can be modified in the Setup options).

C08-Referred Attribution report will attribute CPU usage by system modules back to the calling application program.

C07 report can be used to summarize usage by procedure. You can also use the Source Program mapping feature to display usage at source code level. After a program has been mapped, the C07 report is used to summarize usage by procedure

Recommendation for Hottest COBOL Modules:

For your hottest COBOL modules, we recommend you follow best practices for COBOL v6.2 Migration as suggested by the Migration Assistant and Migration Guide for upgrading compiler levels. Especially pay close attention to Chapter 18, “Adding Enterprise COBOL Version 5 or Version 6 programs to existing COBOL applications” in the migration guide. Also note, Enterprise COBOL v5, v6 or later executables are Program Objects and can reside only in PDSE data sets. If your hottest COBOL modules load libraries are in PDS data sets, when you recompile these modules with latest COBOL compiler, bind them into a PDSE data set, and add the new data set to your load library concatenation. We recommend you have the same naming convention for your PDS and PDSE data sets and remember to add the new PDSE data set to your current load library concatenations.

Benefits of Automatic Binary Optimizer:

If you don’t have the source code for one of the identified hot COBOL module or for some reason do not want to recompile, you can make use of the Automatic Binary Optimizer for z/OS (ABO). IBM Automatic Binary Optimizer (ABO) is a unique, cutting-edge technology designed to optimize the performance of COBOL program modules built with VS COBOL II v1.3 to Enterprise COBOL for z/OS v4.2 without the need for recompiling from program source. ABO uses advanced optimization technology shipped in IBM Enterprise COBOL for z/OS to perform high-fidelity optimization and generates code to fully exploit IBM z Systems mainframes without affecting program logic. As a result, the optimized modules run faster but program behavior remains unchanged, significantly reducing testing effort. For more information please refer to: https://www.ibm.com/us-en/marketplace/improved-cobol-performance.

*Disclaimer: all performance results reported in this article are based on internal IBM compute-intensive test suites. Performance results from other applications may vary.

References:

The COBOL Migration Assistant: https://cobol-migration-assistant.mybluemix.net/

COBOLv6.2 Migration Guide:
http://publibfp.boulder.ibm.com/epubs/pdf/igy6mg20.pdf

ABO for z/OS v1.3 Users Guide:
http://publibfp.boulder.ibm.com/epubs/pdf/c2785454.pdf

APA library link: https://www-01.ibm.com/software/awdtools/apa/library/ - version14

APA user guide: http://publibfp.boulder.ibm.com/epubs/pdf/cazgug00.pdf

APA customization guide: http://publibfp.boulder.ibm.com/epubs/pdf/cazgcg00.pdf