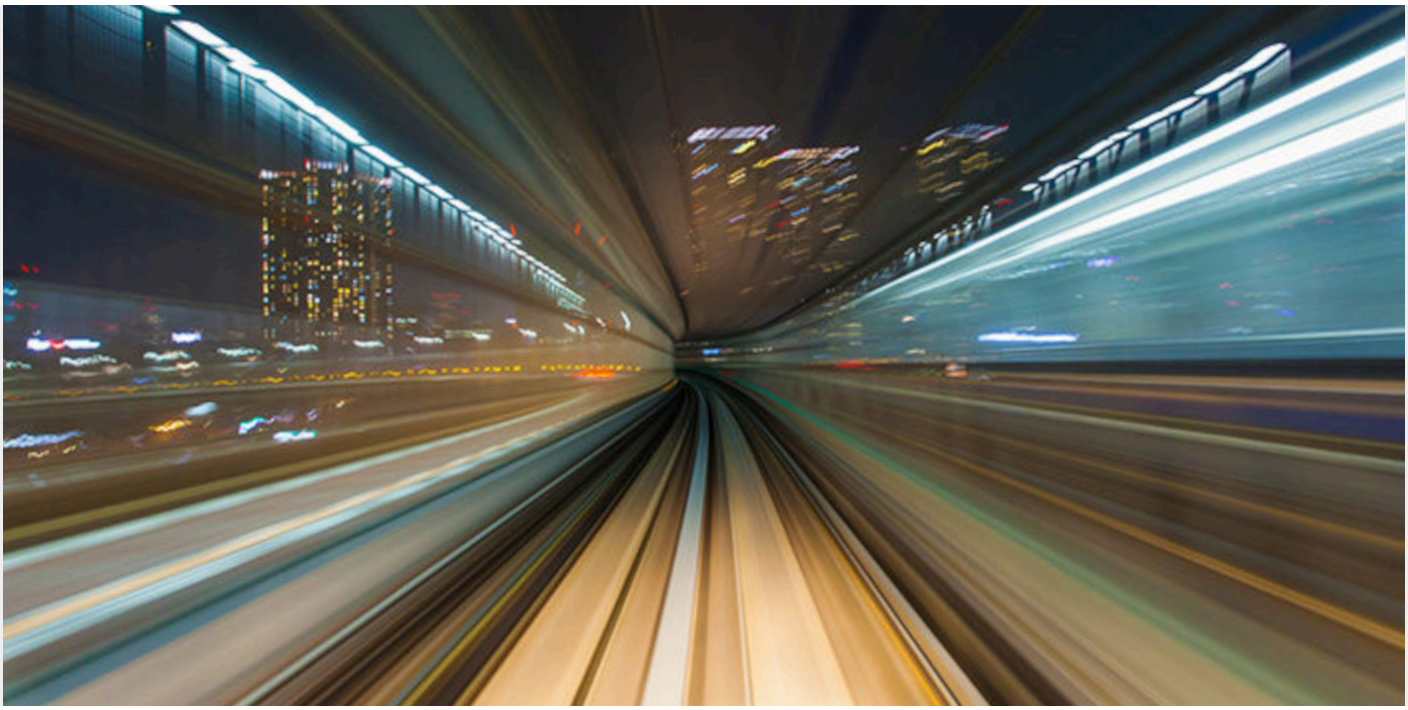
HOW TO IDENTIFY AND USE NEW COBOL OPTIMIZATION FEATURES



Here is an overview of the types of performance optimization available in COBOL Versions 5 and 6 and how to identify and use these new COBOL optimization features.

You've tested and debugged your COBOL programs, now it's time to optimize them. In this post, we'll provide an overview of the types of performance optimization available in COBOL Versions 5 and 6, as well as direct you how to identify and use these new COBOL optimization features.

IBM COBOL Versions 5 and 6 have provided more than a few types of optimization to date, and more are likely to arrive in future releases and through on-stream COBOL maintenance. It's important, however, to provide a measurement of your gains using a performance profiler like BMC AMI Strobe.

It's imperative to provide application performance baselines before and after optimization so capacity and performance specialists can get the data they need. This data can provide future guidance for your applications and be significant to reducing your 4HRA costs.

Implementation Hints and COBOL Optimization Opportunities

Before you start looking for big optimization opportunities, start small. Run with OPT(0) to flush any data format problems, then move to OPT(1) or OPT(2). Make sure you test for accurate results at each optimization level, as some bugs have appeared only with higher OPT levels.

When you're ready to search out optimization opportunities, here's what IBM has been focused on:

- **PERFORM statements** that can be eliminated, made more efficient by "block reordering" or replaced with more efficient inline code. These optimization processes are especially helpful if

the PERFORMs are frequently called because it removes subroutine calls, which add overhead, especially if they're called repetitively.
- **COMPUTE statements**, especially anywhere significant arithmetic is involved. This moves arithmetic to higher-performing, newer instructions, such as SIMD. Decimal arithmetic can be exceptionally fast using the newer DFP features on newer z Systems hardware, z9 and higher. Conversions to and from DECIMAL were especially slow on COBOL Version 4 and below. The new DFP conversion instructions (CONVERT FROM/TO PACKED) on the z13 accelerate a COBOL "DIVIDE A BY B GIVING C."

There are also compile options available during testing that help programmers identify issues that would affect application optimization:

- **RULES(NOLAXPERF)** will issue warnings for inefficient coding practices, as well as compiler options that can impact performance.
- **NUMPROC(PFD)** can significantly improve decimal and zone-decimal processing; however, the data must adhere to specific IBM system standards (refer to the IBM programming guide for more information).

Areas of Greatest Return

There are significant areas of improvement in COBOL Versions 5 and 6 and in hardware systems that that could accelerate your COBOL programs.

Binary Floating Point (BFP)

COBOL Versions 5 and 6 have significant performance improvements when optimizing PERFORM statements and anywhere there's significant arithmetic. Version 5 will use all 16 BFP registers. Older COBOL versions used only the original four BFP registers. Using register arithmetic is faster than using memory and swapping to/from registers. Having 12 more BFP registers increases register use, keeps memory use at a minimum and helps optimize hardware cache use.

Vector Registers

The z13 Systems introduced vector register instructions, enabling COBOL to use the z13 single instruction, multiple data (SIMD) features to help arithmetic performance. There are 32 vector registers on the z13 and there is a movement to use more SIMD instructions to further performance gains, according to IBM Systems Magazine. In his SHARE presentation, Tom Ross described how the z13 instructions accelerate COBOL arithmetic calculations, including integer and floating point computations, and string manipulations like string search and string comparisons.

Decimal Floating Point (DFP)

Another significant area of improvement involves decimal arithmetic acceleration using the DFP feature on the newer machines. DFP was implemented starting with the z9 Systems and enhanced for each successive machine (five total) through the z13. PACKED-DECIMAL data in your COBOL programs could easily benefit and be especially quick for complex calculations.

Conversions to and from decimal and other data formats could be accelerated with DFP conversion instructions, too. IBM has shown that replacing decimal conversion instructions generated by the

COBOL Version 4.2 compiler could be many times faster by just using the newer DFP conversion instructions. Elimination of conversions that usually involve the use of common subroutine calls with in-line DFP instructions can reduce CPU usage and increase performance.

Single Instruction, Multiple Data (SIMD)

A final area of great return, the SIMD feature on the z13 Systems can be exploited by COBOL Version 5.2 and 6.1. Functions like INSPECT TALLYING or the REPLACING statement can benefit. We'll probably see future COBOL versions exploit more SIMD features.

Keeping Hardware and Software Up-to-Date

COBOL has just started with optimization and use of new features on the newer machines. You'll definitely see those advantages appear in newer releases of COBOL. This fact makes it imperative to keep your software and hardware up to date. This list is not in any way complete, but types of optimization that we've encountered include:

- **Statement deletion**: Some COBOL statements are just gone, and if your debugging tool isn't enabled, you'll have problems.
- **In-lining**: Instead of calling a subroutine/function as the COBOL code implies, optimization could instead insert the code in the COBOL stream. You'll see no subroutine/function call.
- **Arithmetic type substitution**: This is an equivalent way to perform the calculation. Use of DFP on all z Systems machines and use of SIMD on z13 are common COBOL Version 5 and 6 techniques to accelerate arithmetic computations.

Hardware provides the features but you can't take advantage unless you keep your software up to date. Plan a regimen to update both hardware and software. Updating the hardware but not the software will likely lead to discussions around ROI as your optimization benefits decrease. Keep statistics and measure with the proper tools. You'll need to show progress and that will keep the skeptical ROI discussions to a minimum.