New W&W Application Profiler for Z

The looming (April 30, 2022) End-of-Support for IBM Enterprise COBOL for z/OS V4.2 has done a wonderful job of focusing minds on z/OS compilers in general, and the COBOL compiler in particular. Interestingly, this coincides with a growing acceptance of IBM's messages that, if you want to get the maximum performance from your CPC, you *have* to recompile your application programs with the latest compilers and the highest possible Architecture Level.

As installations started turning their attention to their COBOL program inventory, many application developers and DevOps managers had the same thought - "'We have *tens of thousands* of application programs; where do I start?"

Following those discussions with some of our customers, our colleague **Mario Bezzi** led a project to review the challenges that customers are facing, and how the existing tools and products could help them. We found that no single product addressed *all* their needs, and there were some questions that were not addressed by *any* product. Based on that work, we created the Watson & Walker Application Profiler for Z (W&W AP4Z).

What is the Watson & Walker Application Profiler for Z?

W&W AP4Z is designed to run *all* the time, gathering information about *all* executed application programs. Because it runs all the time, it is able to build a historical database of *all* your application programs, their inter-relationships, their compile attributes, their CPU usage, where and when they ran, and other data. It is a little like a software asset manager such as TADz, except that it is aimed at applications rather than software products. It is also a little like an execution sampler (Strobe, for example), except that it gathers information at the program level rather than the instruction level, resulting in an overhead that is nearly imperceptible. It is a little like a load library scanner, except that it reports that information for the programs that are *actually executed*. And finally, it is unlike any other product we are aware of in that it gathers information about *dynamically-called programs*. Because of its light weight and broad view, we believe AP4Z is *complementary to*, rather than being an alternative to, these products. With the widespread move to object-oriented programming and microservices, W&W AP4Z sheds light on programs that currently lurk in the background, invisible to performance analysts.

All of this information is integrated into a single database with an intuitive, open source, browser-based interface that supports batch reports and dynamic queries. The intended audience is performance analysts, application developers, database DBAs, DevOps analysts, change control staff, and system programmers.

We have reviewed W&W AP4Z with various development groups in IBM and with a number of customers, and have received feedback like “The Application Profiler for Z is a great tool! We are really impressed with the rich data it can collect and with its very low overhead!”[6] Our clients report an average CPU and elapsed time savings of about 20% when they move to COBOL V6 and recompile with a current ARCHLVL, and W&W AP4Z can help them with that migration. We think that you and your Application Development colleagues will be equally impressed with it.

Our clients report an average savings of 20% when moving from COBOL V4 to COBOL V6 with a current ARCHLVL.

W&W AP4Z Overview

Before we get into discussing the issues that W&W AP4Z addresses, I thought that it would help if we start with a 10,000 foot view of how it works. Hopefully this brief overview will give you some context to how W&W AP4Z would work in *your* enterprise.

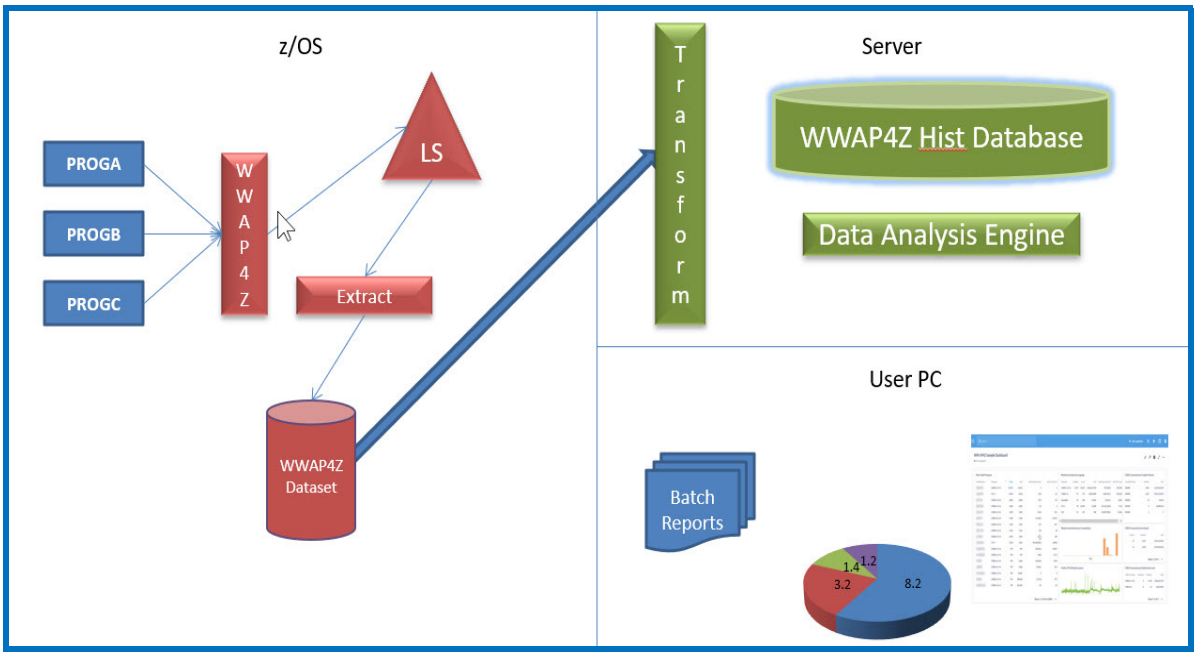
There are two parts to W&W AP4Z:

- A z/OS component that gathers information about executing application programs, writes that information to a logstream, and extracts the W&W AP4Z records from the logstream for subsequent post-processing. The logstream configuration can range from using a single logstream for all systems in the sysplex, down to different workloads in the same LPAR using different logstreams.
- A component running on a distributed platform that provides the repository for the W&W AP4Z data, a powerful open source data query tool to support ad-hoc queries, and policies and processes to create batch reports and perform housekeeping on the database.

Figure 9 on page 46 illustrates how W&W AP4Z would fit into your environment.

[6] Based on experiences in Beta customers, we expect the overhead to be as low as 0.5% of the profiled workload for Batch, and less than 1% for transactional workloads.

Figure 9 - W&W AP4Z High Level View



Data Collection

No source changes, JCL changes, user exits, or APF authorization required.

W&W AP4Z gathers a wealth of information about executed COBOL, C/C++, PL/I, and some Assembler programs. It doesn't require any JCL or source changes, doesn't use any system-level exits, and doesn't require APF authorization. Installation and implementation are specifically designed to be as easy and non-intrusive as possible. The installation of the z/OS part of W&W AP4Z is as simple as uploading a load library and a JCL data set to z/OS, defining a logstream, making one dynamic change to Parmlib, adding a load module to LNKLST, and starting a Started Task - the entire install took me about 15 minutes.

Terminology

W&W AP4Z gathers information about two 'types' of application programs. There is no common naming convention to differentiate between the two, so we will use:

- ***Main* program**. This is the program that is specified on the EXEC PGM= JCL statement.
- ***Called** program*. This is a program that is specified on a dynamic call statement.

AP4Z is specifically designed to profile *applications* and doesn't interact with the execution of non-application code.

For batch jobs, the overhead is in the order of a few hundred microseconds of CPU time per profiled jobstep, meaning that it can be enabled to collect information about *all* executed application programs while having an indiscernible overhead. It also provides the ability to filter the programs that will be monitored, either at the system level or the individual job or program level.

A very important capability of W&W AP4Z is that it also gathers information about *dynamically called* programs; this sets it apart from SMF type 30 records which do not have granularity below the program named on the EXEC PGM= statement. The only other ways that we are aware of to gain this visibility are:

- Use an execution sampler such as the BMC Compuware Strobe product or IBM's Application Performance Analyzer - however the CPU cost for gathering this information in W&W AP4Z is less than *one microsecond* per dynamic call (depending on how much information you want to collect), a tiny fraction of the cost of enabling an execution sampler. This makes it possible to profile all applications all the time, to collect a wealth of information about your entire application portfolio.

- Use a software asset management tool such as IBM's Tivoli Asset Discovery for Z (TADz) or UBS-Hainer's P-Tracker. They can optionally see and report on some application programs, but they don't provide the application-oriented information that W&W AP4Z provides, because that is not their primary focus.

The focus of W&W AP4Z is to provide information that is not available elsewhere, and to combine information that is spread across multiple products and integrate it into a single product to help you visualize and manage multiple aspects of your application program portfolio.

What Type of Information is Gathered?

W&W AP4Z is not a performance monitor, and it is not an execution sampling tool like Strobe. We, and our Beta customers, believe that W&W AP4Z addresses a niche that is not covered by any existing products. The following is a *sample* of the type of information that W&W AP4Z gathers about *each executed instance* of your application programs, including those dynamically-called programs (note that W&W AP4Z must be installed when the application program executes in order to gather its information):

- Sysplex name, System name, Job name and Job ID, Step name, Program name, and job and step start time.

- CPC type and model, and the operating system release where the program ran.

- Elapsed time and CPU time, broken out into general purpose and zIIP time. Information for each dynamically called program is summarized and written once, when the main program ends.

  - If the program abends, the information is still collected, along with information about the reason for the abend.

- Name of the programs that called this program *and* the name of programs that were *called by* this program. Also, the number of times the program was *loaded* and the number of times it was *called*.
- The compile time and date of each program, as well as the version and release of the compiler that was used.
- The compile time options and Architecture Level.
- CPU time consumed by the gathering task itself.
- W&W AP4Z can report on module load activity, program call activity, and error condition handling. These options can be enabled separately, and the level of profiling can be different for different Jobs / Programs, depending on your needs.

W&W AP4Z provides great granularity, so you can control which programs you want to profile, and the level of detail that will be collected, meaning that you can collect different levels of detail for different programs. Or you might decide to enable more detail on your Quality Assurance systems, so that you can analyze the detailed behavior of, and inter-relationships between, selected programs, and enable more summarized profiling of programs on the production systems.

Supported Environments

The initial W&W AP4Z release supports Assembler, C/C++, COBOL, and PL/I programs when used in batch jobs, Db2 stored procedures, under OMVS, or in TSO. We are already testing CICS support and expect to make that feature available before the end of the year. We expect W&W AP4Z to also work seamlessly with IMS, and will be happy to add support for IMS if enough customers show interest in this capability.

We considered making W&W AP4Z zIIP-eligible. However, the CPU time required to create and then destroy the environment to use a zIIP is greater than the total CPU time used by W&W AP4Z and the system services that it uses, so it didn't make sense to make it zIIP-eligible.

Reporting Environment

Collecting all that lovely interesting information isn't much help if you don't have a cost-effective way to query and manage that data. The distributed component of W&W AP4Z provides that functionality.

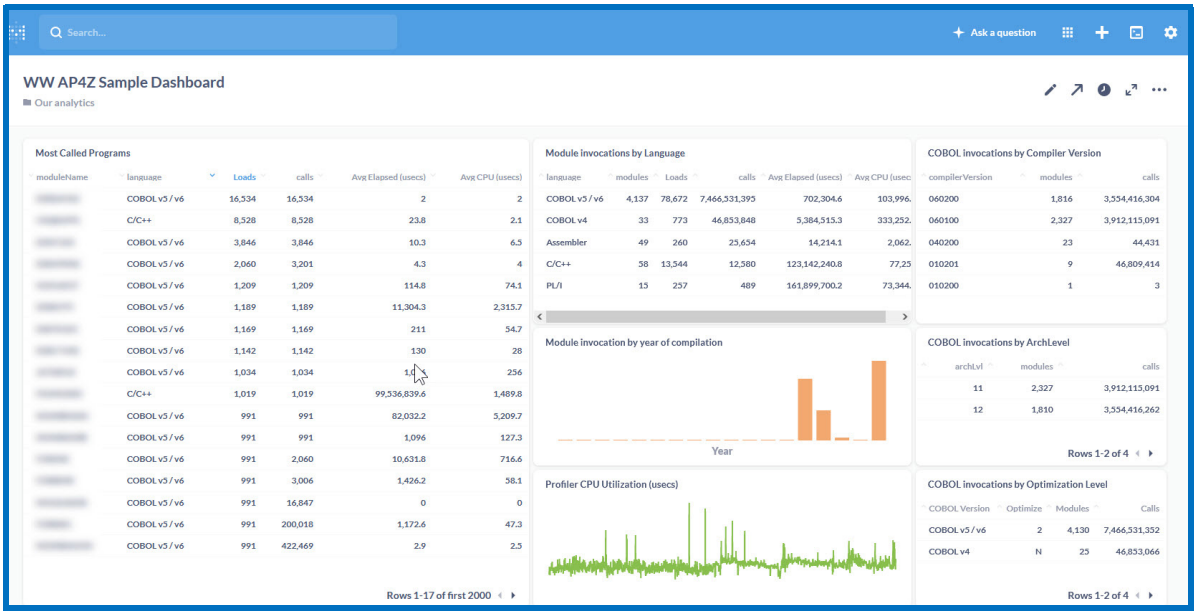
The distributed component is based on popular open source technologies and is packaged as a Docker container for ease of installation[7]. It includes:

- An embedded database manager, plus the utilities to load and manage it and run batch reports.
- An easy-to-use dynamic query tool (Metabase).

The user interface uses a standard browser, meaning that no additional software needs to be installed on the user's PC. The container can be hosted on a Linux or Windows server.

The product comes with a set of standard reports (number of executions of each program, number of programs that were compiled with each compiler level, number of programs by ARCHLVL, and so on). A sample of the W&W AP4Z user interface is shown in Figure 10 - don't worry if you can't read the individual reports, we will get into more detail about those soon.

Figure 10 - W&W AP4Z User Interface



Additional reports can easily be created using the included Metabase product. If you are an SQL wizard, you can create the queries using SQL. But if (like me) your SQL expertise doesn't go much beyond 'SELECT * from TABLE', Metabase can easily be used to create your own tabular or graphical reports. Additionally, W&W AP4Z includes the source for the sample queries, allowing you to take a working example and adjust it to fit your needs.

Hopefully this gives you a little insight into how the various parts of W&W AP4Z work and how it would fit into your environment. Now let's have a look at the types of questions that W&W AP4Z can answer for you, and who in your IT department might be interested in that information.

[7] The required components are also available individually for customers that are not interested in, or are not able to use, a packaged solution.

What Issues are Addressed by W&W AP4Z?

W&W AP4Z *started* as a project to help customers identify programs that were compiled using a compiler that will soon run out of support. As nearly everyone knows, moving from COBOL V4 to V6 is not as simple as just re-compiling your programs. It requires extensive testing, and no one wants to be in the position of having to choose between recompiling a critical program with an unsupported compiler, or having to move the program to a new compiler version when you are under time pressure to provide a working program.

However, as we discussed our plans for W&W AP4Z with customers, the list of requirements quickly grew. As you can imagine, we got lots of “Hey, if you are collecting *that* information, could you also get *this* while you are at it?” and “Our Performance/Application Development Support/Change Control/Audit team would *love* to be able to see xxx if you could capture that as well.” Based on all that enthusiastic feedback, the following are some examples of how W&W AP4Z can add value.

Which Programs Are We Actually Using?

One of the most common requests we had was from customers that were trying to figure out which of their tens of thousands of load modules are *actually being used*. There are numerous tools that scan load libraries and extract information about compile dates and compiler levels and so forth, however that is *static* information. You can have a load library with thousands of load modules, but how do you know which of those are actually still being used? And of the ones that *are* used, which are used the most often, or use the most CPU time (and therefore are more likely to benefit from being recompiled)?

W&W AP4Z lets you see which programs are actually being executed, as well as the elapsed time, CPU time, and compiler options and levels for each program instance.

What About Programs That Are Called Dynamically?

As mentioned above, the problem with SMF type 30 records is that they only show the program that is named on the EXEC PGM= statement. If that program then performs dynamic calls to 50 other programs, there is *nothing* in the type 30 records about those other programs. The tools that report on load library contents can tell you that the load modules or program objects exist, but they have no knowledge about which ones are executed or how often.

Because W&W AP4Z sees *all* application program executions (if you allow it to), it sees programs that are called on the EXEC PGM= statement (and are reported in the SMF type 30 records) AND all the ones that are called dynamically. As someone that can barely spell Db2, it drives me nuts when I look in SMF type 30 records and see job steps that consumed 6 hours of CPU time, and the program that used all that time was .... IKJEFT01. But thanks

to W&W AP4Z, I can now immediately see that the offending program hiding behind that IKJEFT01 was ... FRANKDB2. Oops.

Which Of My Programs Were Compiled With Which Compiler Level?

The looming end of support for the COBOL V4 compiler has attracted a lot of attention to compilers and migrations. However, the reality is that many sites are still running programs that were compiled with compilers even older than COBOL V4. One of the great things about z/OS is that a program that was compiled and link edited 30 years ago can still be running happily on your z/OS 2.4 system on a z15.

However, what happens if something requires one of those *really* old programs to be recompiled? Never mind the fact that the old compiler is no longer supported. The first question would be do you still *have* that ancient compiler? If it was last used 30 years ago, do you still *have* the compiler data sets, compile JCL, and the associated product manuals? And if you do, will that old compiler *work* on current operating system and hardware levels?

Before you start losing sleep over yet another concern, a good place to start would be to find out if you actually *have* any of those really old programs that are still in use. You probably have some very old programs in your load libraries, but you might want to focus on the ones that are still being used first. Some of the pieces of information that W&W AP4Z gathers are the programming language, the compile date and time, and the compiler level for each executed program. One of the canned reports that is provided with W&W AP4Z reports the number of executed programs for each level of each compiler. If the number is greater than zero, you can then drill down and get a list of the programs for each compiler level and the number of times that each was executed. Hopefully the answer will be a relatively small number, and you can get a good night’s sleep again.

Which Programs Are My CPU Hogs?

I think that every site I have ever visited had a listing showing their top 50 CPU hog programs. However, I have yet to find one that built that list from anything other than SMF type 30 or Db2 accounting SMF records. Not that there is anything wrong with type 30 data. But many of our customers rely heavily on dynamic calls so they can have *one* program to perform some common task. The problem with relying on type 30 records is that you don’t have *any* visibility into those dynamically-called programs. In all likelihood, none of them ever appear on your top 50 list. In fact, it is likely that you probably don’t even know how many times those programs are called, or how much CPU time they use. For Db2 programs, the Db2 Class 7 package-level accounting trace gathers information about the CPU time and elapsed time to process the SQL statements in programs that issue SQL calls. But if a main program calls two sub-programs, both of which execute 1M COBOL instructions, and only one of which issues an SQL call, the Class 7 record will have *no* information about the

program that doesn't call SQL, and information for only the processing associated with that SQL statement in the other program.

Figure 11 - Sample Top 50 Program report

Frank's Dashboard

Frank Kyne's Personal Collection

Top Programs by CPU Time (mics)

moduleName	Language	Loads	Calls	Avg Elapsed (usecs)	Avg CPU (usecs)	Total Elapsed (usecs)	Total CPU (usecs)
[redacted]	COBOL v5 / v6	18	4,290,642,435	13.5	2.8	61,200,526,547	8,074,390,394
[redacted]	COBOL v5 / v6	3	3	512,269,282	428,143,868.7	1,536,807,846	1,284,431,606
[redacted]	COBOL v5 / v6	1	1	619,696,671	422,923,248	619,696,671	422,923,248
[redacted]	COBOL v5 / v6	4	67,540	1,632,339.2	819,482	2,033,687,611	583,615,341
[redacted]	COBOL v5 / v6	3	3	112,621,855.7	98,277,432	337,865,567	294,832,296
[redacted]	COBOL v5 / v6	1	1	10,491,652,429	128,235,199	10,491,652,429	128,235,199
[redacted]	COBOL v5 / v6	1	1	202,506,238	127,717,974	202,506,238	127,717,974
[redacted]	COBOL v5 / v6	6	1,444,955,475	53.5	17.8	2,494,565,302	518,911,776
[redacted]	COBOL v5 / v6	2	2	90,947,828.5	81,785,474.5	181,895,657	163,570,949
[redacted]	COBOL v5 / v6	1	7,216,346	1,601	11	11,559,368,316	85,928,642
[redacted]	COBOL v5 / v6	4	4	50,949,689.3	45,567,962.3	203,798,757	182,271,849
[redacted]	COBOL v5 / v6	2	2	88,344,298	79,128,212	176,688,596	158,256,424
[redacted]	COBOL v5 / v6	2	2	114,667,432	70,395,701	229,334,864	140,791,402
[redacted]	COBOL v5 / v6	1	1	285,441,764	49,194,397	285,441,764	49,194,397
[redacted]	COBOL v4	3	3	499,167,423.3	72,708,269.7	1,497,502,270	218,124,809
[redacted]	COBOL v5 / v6	1	1	50,780,564	44,899,398	50,780,564	44,899,398
[redacted]	COBOL v5 / v6	1	152,548	3,959	289	603,965,122	44,162,606
[redacted]	COBOL v5 / v6	1	1	145,899,402	40,816,966	145,899,402	40,816,966
[redacted]	COBOL v5 / v6	1	152,548	278	246	42,510,181	37,671,889

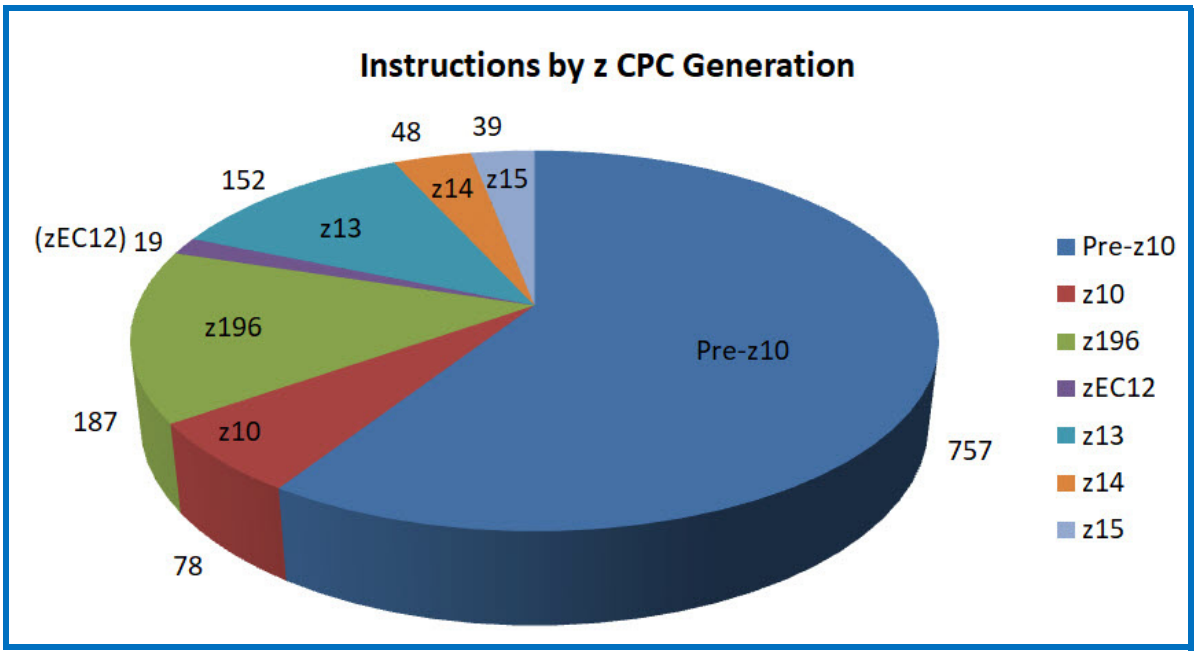
Rows 1-19 of first 2000

Figure 11 shows a sample top program report. Because W&W AP4Z sees *all* application program executions, those dynamically-called programs are now just as visible as all the programs in your top 50 list. Due to their nature, it is quite likely that each one doesn't use much CPU time. However, that 'insignificant little service program' might be called 1,000,000 times a day. If that is the case, that program might now make its debut on your top 50 list, as you can see in the figure. In fact, in this example, the program that used the most total CPU time of *all* application programs was a service routine that was called over 4 *billion* times during the reporting interval. And if *none* of your dynamically-called programs reach your list, at least you *know* that they are not heavy CPU consumers, rather than *guessing* that they are not.

Is It Really Worth Recompiling With a Higher ARCHLVL?

IBM's Z Performance team in general, and **David Hutton** in particular, has been working hard to help customers understand the importance of recompiling programs with current compilers. On the COBOL side, **Tom Ross** ('Captain COBOL') has also been encouraging customers to recompile with the highest ARCHLVL possible. The reason for these recommendations is clearly seen in Figure 12 on page 53 - nearly 40% of all Z instructions have been introduced after the z9 generation. The modules created by the COBOL Version 4 compiler do not exploit *any* of the instructions that were added since the z9.

Figure 12 - Z Instructions by ARCHLVL (© IBM Corporation)

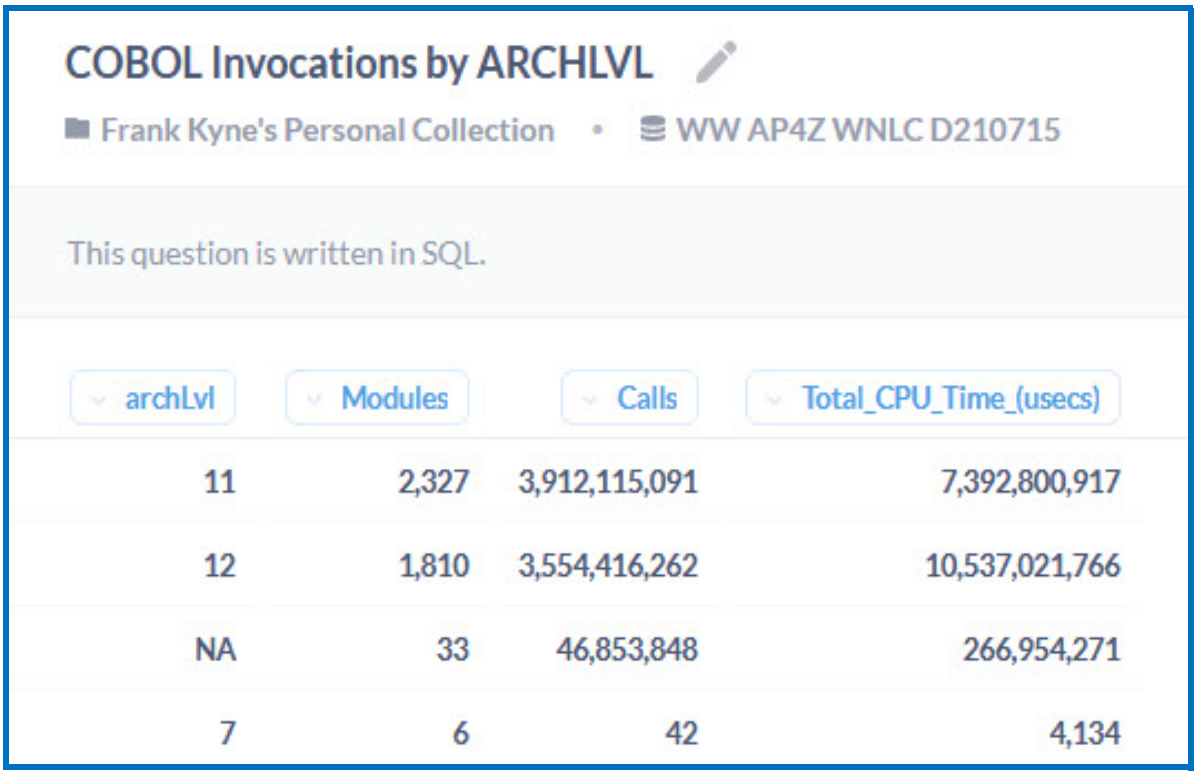


The challenge, both for IBM and for customers, is in trying to predict how much benefit *your* workload would get from using a higher ARCHLVL. If your programs happen to use a function that has been enhanced in one of those later ARCHLVLs, you might see significant CPU time reductions. On the other hand, if your programs don't use *any* of the enhanced functions, then they probably won't see any benefit from a higher ARCHLVL.

W&W AP4Z lets you easily see the difference that recompiling with a higher ARCHLVL has made to *your* programs. Because it captures the ARCHLVL, elapsed time, and CPU time, you can very quickly see when the ARCHLVL for a program changed, and what, if any, difference that made to the program's resource consumption. Depending on your results across a representative sample of your application programs, you can use that information to prioritize, or de-prioritize, the effort to recompile your programs with a higher ARCHLVL.

As shown in Figure 13 on page 54, you can also monitor your progress in terms of how many of the programs that you are actually running are on each ARCHLVL, and the total amount of CPU time used by programs at each ARCHLVL. In this example, you can see that the bulk of programs are already using either ARCHLVL 11 or 12, so they are in good shape.

Figure 13 - COBOL program invocations and CPU time by ARCHLVL



COBOL Invocations by ARCHLVL

Frank Kyne's Personal Collection • WW AP4Z WNLC D210715

This question is written in SQL.

archLvl	Modules	Calls	Total_CPU_Time_(usecs)
11	2,327	3,912,115,091	7,392,800,917
12	1,810	3,554,416,262	10,537,021,766
NA	33	46,853,848	266,954,271
7	6	42	4,134

If it is any help, our clients have found that moving from COBOL V4 to COBOL V6 with ARCHLVL 12 or 13 results in an *average* drop in CPU consumption of about 20%. You can find more information about the performance considerations of recompiling programs in the section titled 'Compile / Recompile Programs With Newer ARCH Values' in the 'Job-level CPU Saving Opportunities in an Enterprise Consumption Environment' article in *Tuning Letter 2021 No. 1*. You can also find information about the results of IBM's internal measurements with different ARCHLVLs in *IBM Enterprise COBOL for z/OS, V6.3 Performance Tuning Guide*, SC27-9202. While there have been many successful migrations to higher ARCHLVLs, there are still issues when the application allows invalid data into the programs. For more information, see 'COBOL News' on page 12.

Which Programs Could Be Affected If I Change Program 'A'?

We all know how much IT people *love* creating documentation. It is nearly as much fun as *maintaining* that documentation. And after all, we all have so much spare time on our hands, why would documentation ever be out of date? I remember speaking to a sysprog one time who described his site's application documentation as 'detached from reality', but I'm sure that his company was unique ☺.

The reality is that we all do our best to keep our documentation up to date. But if you are about to update some program that provides a service to *many* other programs, are you 100% sure that you have a *complete* list of those other programs so that you can test the impact of your change on *all* those programs?

One of the pieces of information that W&W AP4Z optionally collects about programs is the name of the program that called that program. With W&W AP4Z, it takes just a few seconds to get a list of all the programs that called your program.

You can also use W&W AP4Z to go in the other direction, and get a list of all the other programs that your program calls using dynamic calls. While it *might* be possible for you to get this information yourself by scanning your program source, W&W AP4Z will provide the same information in a few seconds. In fact, the W&W AP4Z information might be more valuable, because it shows the programs that were *actually* called, as opposed to a list of the ones that *might* be called, depending on the flow of control in the program.

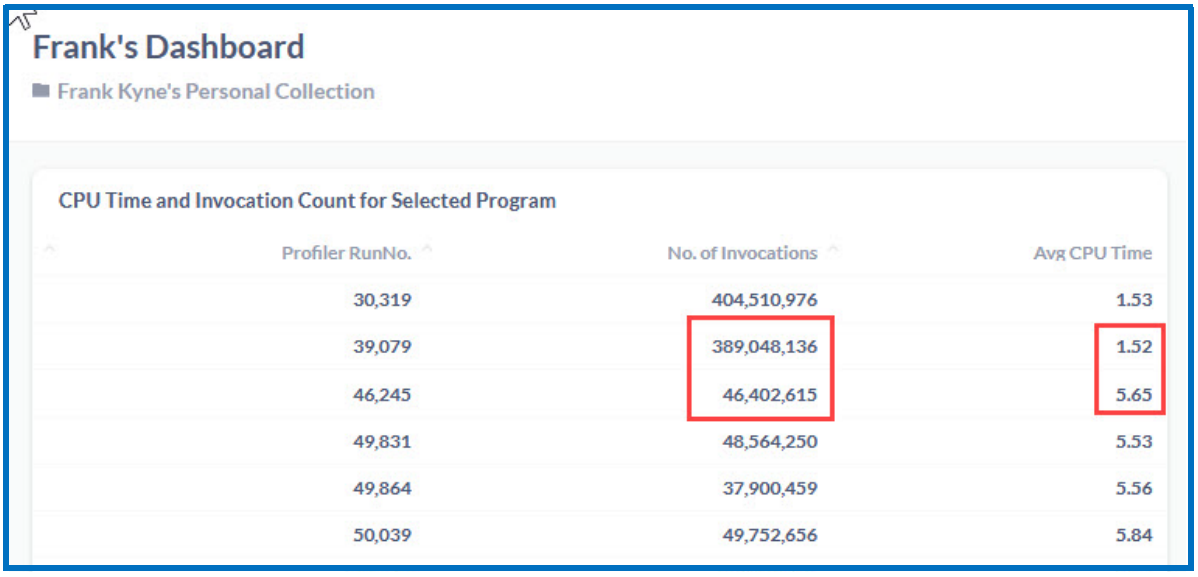
What is the Performance Impact of an Application Change?

You will have seen frequent references in the Tuning Letter to situations where an application change brought unanticipated performance improvements or degradations. This seems to be particularly prevalent among Db2 programs, where a seemingly trivial change to the program logic causes Db2 to select a different access path for the data used by that program.

If the program is self-contained (that is, it doesn't call any other programs), you might be able to use Db2 accounting SMF records to visualize the program's CPU and elapsed times. However, if the program in question is one that is called by many other programs, accessing that data might not be so easy. And the SMF type 30 records are no help because they don't contain the application program name.

W&W AP4Z can help in this situation. You simply produce a report like the one shown in Figure 14 to show the CPU and/or elapsed times for each instance of the program for as far back as you keep the data in your W&W AP4Z database.

Figure 14 - Plotting CPU and elapsed times for an application program



In this interesting example, each line show the number of times the called program was invoked, and the CPU time (in mics) per invocation. You can see that in the third line, the CPU time increased by a little under 4x, however the number of times the program was called reduced by a factor of nearly 8.5:1. I don't know how you would obtain that information without something like an execution sampler, which has a far higher overhead.

How do I Reduce My Peak Rolling 4-Hour Average?

Sites that are still paying for software based on a peak Rolling 4-Hour Average frequently approach us saying that they have done all the 'normal' R4HA tuning, but their execs want even more cuts, and they don't know where to turn next. The Application Development teams are naturally focused on delivering new functions, because that is what their funders pay them for. And the performance team don't always have the application insight, or supporting performance data, to create a business case that is strong enough to justify an Application Developer's time to investigate and tune application performance.

For such scenarios, W&W AP4Z is ideal because you can quickly generate a report showing the top X application programs for the time period surrounding your peak rolling 4-hours. With that information, you can quickly go back and plot the CPU consumption of each of the top programs and see if there are any obvious behavior changes. Perhaps one of the top consumers during your peak period is a called-program, so it never actually shows up in a report, and no one might be aware of the impact it is having. Maybe the top users are still using an old ARCHLVL. There are many insights that W&W AP4Z can provide, at least some of which might help you justify the time of one of your Application Development colleagues to investigate the opportunities for performance improvements in the application programs that are driving your peak R4HA.

Problem Determination

One of the first questions that is asked when someone is investigating a problem is 'what changed?' An obvious candidate is the program that abended - was *it* changed recently? You can use W&W AP4Z to quickly check the compile date and return code of the last x runs of that program. If it was changed, the next step might be to check the change management database to see what change was made.

However, what about the scenario where the abending program was not changed, but one of the programs that it calls *was* changed? With the tools available to you today, how would you identify that situation? Using W&W AP4Z, it takes seconds to generate a report showing the compile date of all programs that were called by the program you are investigating. If none of them have been changed, then at least you know that for a fact. And if one of them *was* changed since the last time the abending program ran successfully, that might be a fertile area to focus your investigation on.

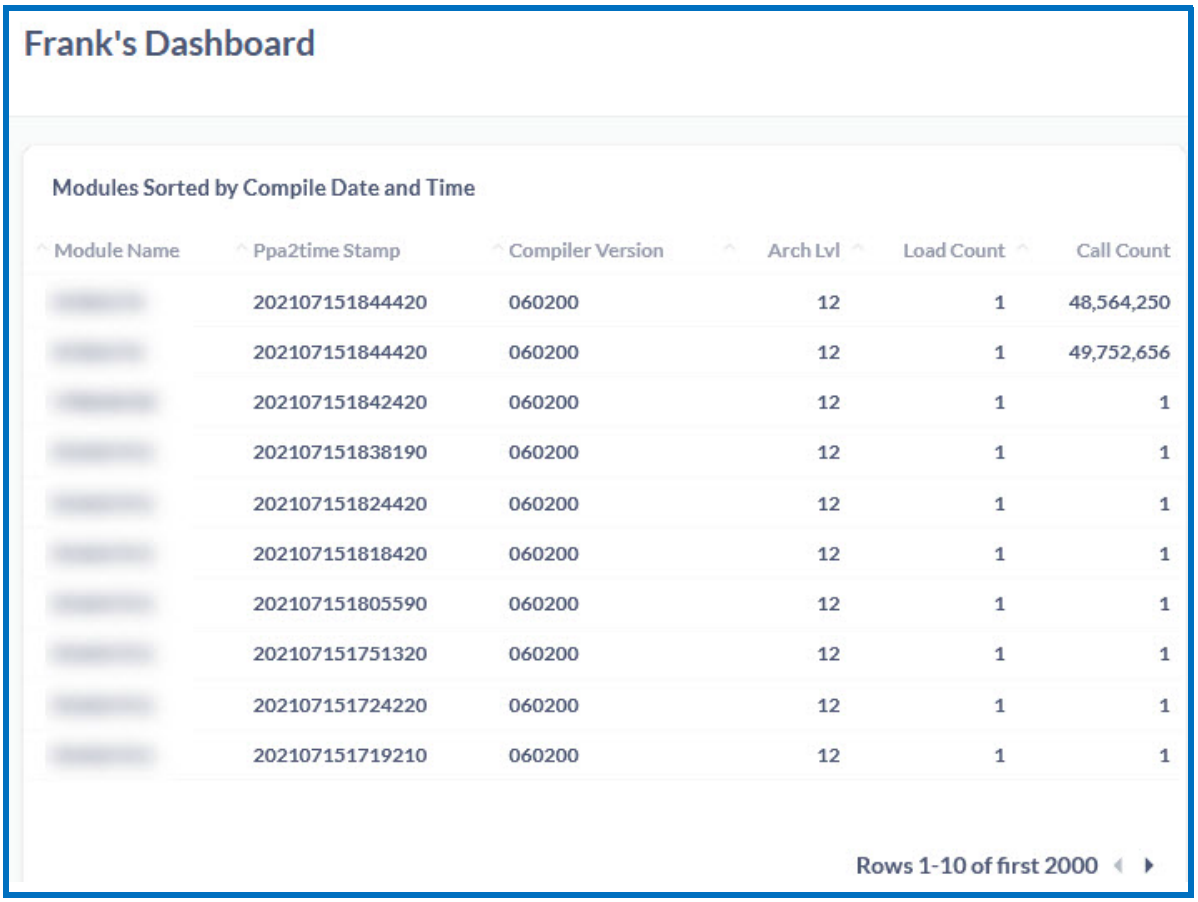
Hi, I'm From the Change Management Team

Right up there with creating and maintaining documentation, another favorite activity of every application developer is attending change management meetings. Not that there is anything wrong with the people in the change management team - the problem is waiting for all your fellow developers to describe their boring changes while you wait to exhilarate everyone with the details of your latest super enhancement.

Of course, not *every* change is that exciting. And no one would really care about that little 5-line change that you were working on, right? So, you tell yourself, I would actually be doing everyone a *favor* if I just slip this little change in without wasting the valuable time of those nice change management people.

But spare a thought for those poor change management people, trying to keep on top of all the changes you and your super-enthusiastic colleagues are making. It is probably as much fun as trying to herd cats.

Figure 15 - List of executed programs sorted by compile date



Well, you can buy *mucho* brownie points with the change management team when you show them a report like the one shown in Figure 15. This report took me about 5 minutes to create (remember that I don't speak SQL) - it shows a list of executed programs (either 'main' or 'called') that have been recompiled in the last X days. They can then compare this list to their

list of change requests. While your load library scanner can identify the compile date for all modules in the library, it can't tell if any of the recently-compiled ones have been executed, or how many times, or when and on which system - but W&W AP4Z can tell them that. And if your installation's DevOps process is supposed to ensure that programs can't be compiled directly into a program library, you can use W&W AP4Z to show your friendly auditors how you verify that your promote process really *is* bulletproof.

Unchange Management

If the change management team are responsible for keeping an eye on programs that are changing, who is responsible for the programs that are *not* changing? Or more importantly, is anyone responsible for cleaning up old programs that are no longer in use? If someone does have that unenviable role, how do they differentiate between an *active* dynamically-called program (one that will never show up in JCL libraries or SMF type 30 records), and a program that is simply never used any more?

Unfortunately, W&W AP4Z can't magically make this problem go away. However, it *can* help you reduce the list of programs that can potentially be cleaned up, by providing a list of dynamically-called programs that *are* still in use.

The Dreaded Re-platforming

Some of you might be experienced enough to remember the era of 'downsizing'. If you were to believe the 'airline magazines' of the time, every mainframe in the world was one step away from being replaced with a second hand PC running under someone's desk. Well, we all know how successful that was.

Roll the clock forward 30 years, and the answer to every IT problem is now the magical mystical Cloud. If you want to pluck an application off your z/OS system and move it to some other platform, you will presumably want to know the relationship, if any, between that application and other applications residing in that z/OS system.

W&W AP4Z can assist in that effort by showing you the programs that are *called* by the application that is being moved, and also the list of other applications that are *calling* the application that is being moved.

These are just a *sample* of the use cases that customers have brought to us. We have many others, and we are sure that many of our readers have still more that we haven't even thought of yet.

Product Roadmap

We will provide a product roadmap after we receive feedback from a larger number of interested customers - obviously our objective will be to prioritize enhancements based on the number of clients that would benefit from each one.

At a high level, enhancements that we hope to deliver include support for CICS transactions, support for IMS, Java program profiling (especially important in light of the COBOL-Java interoperability enhancements in z/OS 2.5), additional canned reports, the ability to create call graphs, drill-down support in the canned reports, and others. Depending on customer interest, it might also be possible to run the W&W AP4Z under zCX.

Pricing

As with all Watson & Walker offerings, our objective is to offer pricing that delivers a fair balance of value to our clients, and revenue to allow us to support and enhance our offerings. We are currently working to identify pricing options that reflect the size of the client's environment while also not generating inordinate amounts of work for either our clients or ourselves. I never thought that I would find myself saying that I have sympathy for IBM's software pricers, but I do! ☺

Summary

We believe that W&W AP4Z addresses an important gap in the marketplace. It combines Cheryl's famous no-nonsense data-driven approach and close customer relationships with Mario's reputation for designing robust, super-efficient code. We have over a hundred customers that have downloaded each of WWUNTERSE and the Important Messages Health Checker - free tools that were also created by Mario. Users of those tools have nothing but positive things to say about them, and we expect an equally enthusiastic response to W&W AP4Z.

If you are under time pressure to identify and recompile your critical COBOL programs before support runs out for the COBOL V4 compiler, W&W AP4Z lets you quickly focus on the subset of programs that are actually in use and prioritize the ones that are most frequently run or that use the most CPU time.

However, the COBOL V4 to V6 migration is really only the tip of the iceberg - it just happens to be the event that sparked the project that resulted in the development of W&W AP4Z. As you've seen above, W&W AP4Z provides information and insight that can be used by many groups in the IT department. It is unique in being able to provide so much insight, especially into dynamically-called programs, with such minuscule overhead. And because it is complementary to your existing products like TADz, Strobe, and IntelliMagic Vision, you can continue to benefit from your experience and skills with those tools, and get even more value

from them by jointly using W&W AP4Z to help you quickly identify where you should be focusing your attentions.

While the techies will be focused on the types of data that W&W AP4Z collects, getting *real value* from that data requires an intuitive, easy-to-use, and low cost reporting interface. The distributed part of W&W AP4Z provides that capability via a combination of canned reports that are delivered as part of the product, and a popular open source data query function that can be used by any user to quickly generate ad-hoc queries.

If you are interested in seeing how W&W AP4Z can help you and your colleagues, contact us about a demo and let us show you how W&W AP4Z can help you cut your costs.