An MVS System Programmer's Trip to ApplicationLand

Mario Bezzi – mario@watsonwalker.com

SHARE Columbus, August 2022

Session 44114

Agenda

- Who am I.
- Why application tuning on z/OS.
- Tuning what – Data access vs data processing.
- Focusing on data processing.
- Pushing on zIIPs exploitation.
- Summary.

Who am I

- Long time MVS system programmer.
- I’ve spent most of my career working on Sysplex, WLM, infrastructure configuration and tuning.
- Started with Assembler and loved it, know some PL/I, Java, Python, nowadays I mostly use C and Metal-C.
- Lately got interested in application tuning, I’ve developed AP4Z, the Application Profiler for Z.
  - AP4Z is designed to profile applications’ execution at scale.
  - Simple, with no special system requirements, with a negligible impact on CPU consumption.
  - Able to build a long term history about active programs, their performance, their relationship.

Why Application tuning on z/OS

Historically the focus for tuning has been higher on infrastructures than on applications because of:

- Better documentation and tooling.
- More widely available infrastructure tuning skills.
- Small changes having wide impact.

But:

- The efficiency of commercial software products is usually better than that of applications.
- Infrastructure usually changes less frequently than applications, and in a more controlled way.
- After so many years of infrastructure tuning the opportunities are lessening.

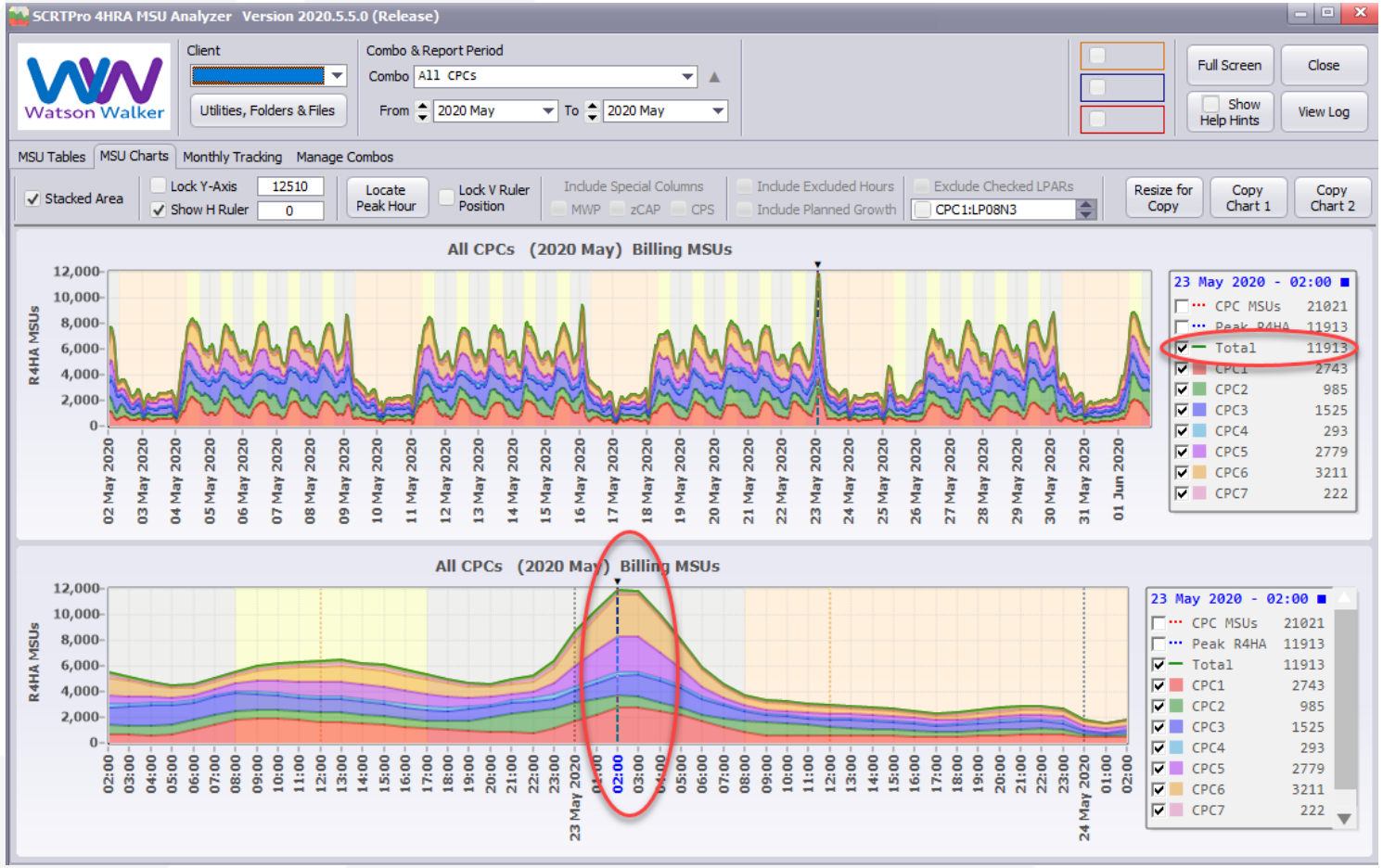
Tuning – 4HRA Peak or Aggregated Consumption ?

R4HA bill is driven solely by the peak 4HRA.

- Adding or removing work outside the peak has ZERO effect on your SW bill.
- Tune your 4HRA to reduce your costs.

With TFP, you pay for every MSU.

- TFP eliminates the 4HRA white space.
- If you add 1 MSU of work, you pay for 1 more MSU, regardless of when It is used.
- Tune against all your workload as any MSU you remove may reduce your bill.



Tuning what – Data Access vs Data Processing

- Depending on your environment data access activities may contribute significantly to total resource utilization / Response time.
- This is true for the cost of doing I/O, which may be avoided, and especially true for the cost of executing SQL statements when using Db2.
- There is less value in optimizing your data processing logic if it only accounts for a small part of the total.
- Do you have a process to detect and fix less than optimal [relational] data access patterns?

Db2 Aggregated Accounting Statistics

SHARE
EDUCATE • NETWORK • INFLUENCE

```
---- HIGHLIGHTS ------------------------------------------------------------------------------------------------------------
BEGIN RECORD: 22-08-08 09:00:00.74   TOTAL THREADS    :   290799   AUTH SUCC.W/OUT CATALOG:   276380   DBAT QUEUED:        0
END RECORD  : 22-08-08 18:51:00.73   TOTAL COMMITS    :  5958985   BUFF.UPDT/PAGES WRITTEN:     8.59   DB2 COMMAND:     5340
ELAPSED TIME:       9:50:59.993358   INCREMENTAL BINDS:    26028   PAGES WRITTEN/WRITE I/O:     8.20   TOTAL API  :  1298239
```

CPU TIMES	TCB TIME	PREEMPT SRB	NONPREEMPT SRB	CP CPU TIME	PREEMPT IIP SRB
SYSTEM SERVICES ADDRESS SPACE	22:10.437200	3:06.722709	10.954048	25:28.113957	5:11.579065
DATABASE SERVICES ADDRESS SPACE	1:41.728340	36:31.472784	3.984849	38:17.185973	3:03:20.737732
IRLM	0.000993	0.000000	4.719280	4.720273	0.000000
DDF ADDRESS SPACE	1:06.923895	17:24.453935	11.546062	18:42.923893	18:43.330459
TOTAL	24:59.090428	57:02.649428	31.204239	1:22:32.944096	3:27:15.647255

CONNTYPE	CL1 ELAPSED	CL1 CPU	CL1 SE CPU	CL2 ELAPSED	CL2 CPU	CL2 SE CPU	CL3 SUSP	CL2 NOT ACC	QUANTITY
BATCH	9 12:22:42.4	7:00:51.1857	1:28:37.0785	1 22:49:52.3	5:04:17.7084	1:26:15.9099	13:41:03.944	1 04:04:30.7	21685.00
CICS	11:52:02.124	1:30:38.3520	4:47.816126	1:04:45.1623	16:16.664259	3:05.522975	26:38.141112	21:50.356930	247.4K
DDF	3 05:46:59.6	46:44.151363	1:13:10.9758	16:39:23.957	27:45.601943	34:58.132718	1:10:21.4723	15:01:16.883	2780.1K
IMS	2 07:45:32.4	32:37.589873	1:32:45.1084	3:07:34.1866	13:48.805087	9:54.963829	37:02.657566	2:16:42.7240	9831.00
RRSAF	6 11:31:14.3	9:33.892336	0.000000	11:20.208253	5:50.076371	0.000000	2:14.246203	3:15.885679	48535.00
UTILITY	6:03:17.4966	12:38.331053	25:36.678291	2:42:50.0181	6:30.802753	25:36.678286	45:29.758584	1:50:49.4568	13418.00

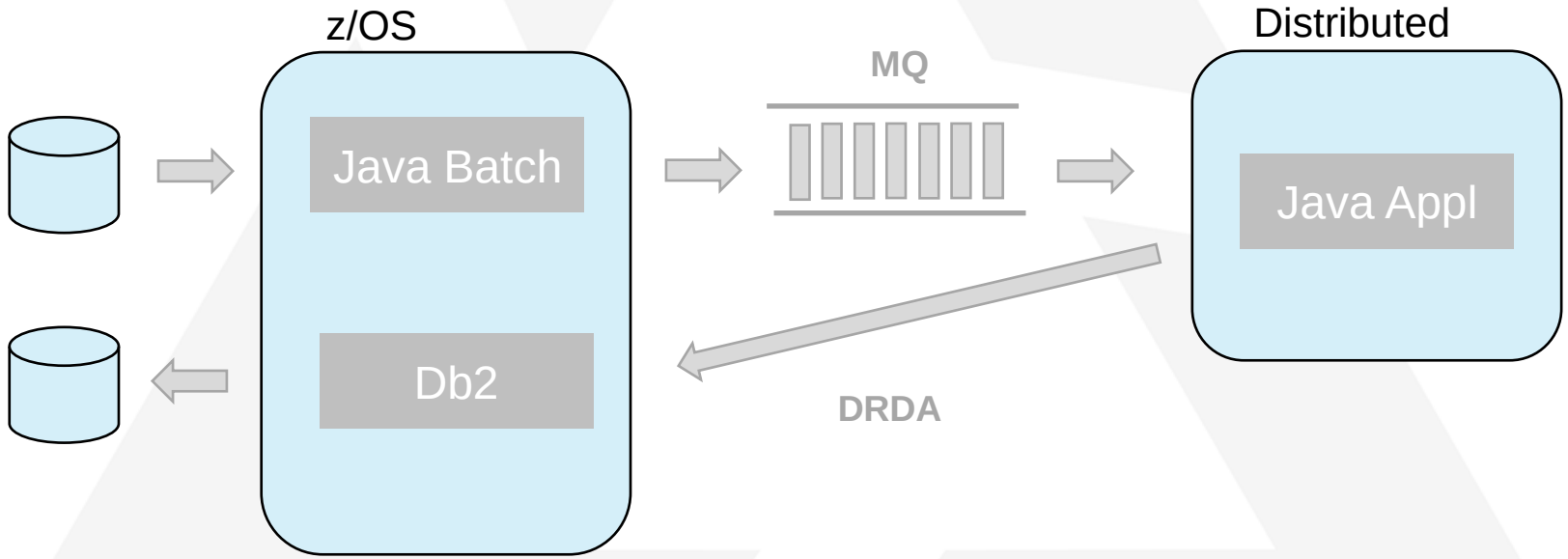
Aggregated Accounting Statistics enabled by activating Statistic Trace Class 9 or IFCID 369
Data only available if Accounting Trace Class 3 is also active

Focusing Data Processing

- Application Design change
- Compiler currency
- Program attributes / runtime options

Application design change

Look at the below real life scenario: A z/OS Java job that sends data processing requests to a distributed server using MQ. To process the request the distributed server gets data from the same z/OS image via DDF.



Can something like this really be changed?

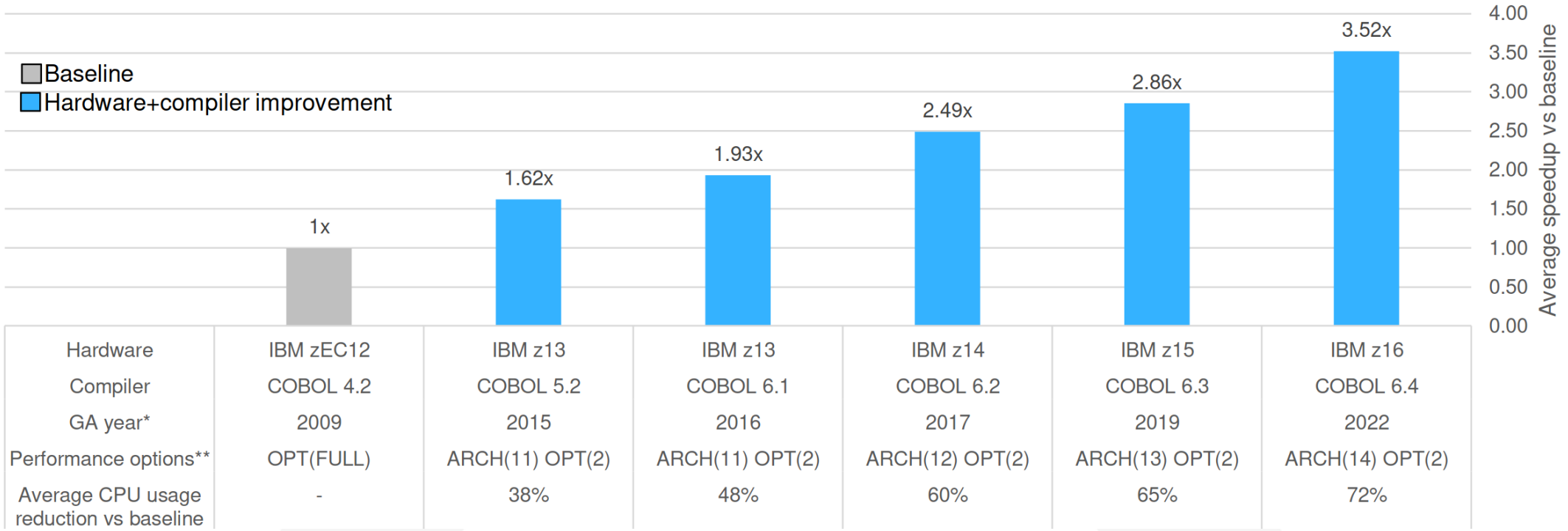
Why (COBOL) Compiler currency

- Enterprise COBOL 4 and 5 **compilers** went out of support.
- The COBOL V5 and V6 are modern compilers aimed at generating very efficient code.
- COBOL V6 offers 3 levels of optimization, allowing to achieve savings of between 15% and 43% compared to COBOL V4, depending on which level of optimization is requested at compile time.
- COBOL V4 offered an OPTIMIZE option. But according to IBM benchmarks, using this option only decreased CPU consumption by an average of 1%.
- COBOL V4 generates code based on 1990s technology. COBOL V6 **optionally** produces code that exploits the latest processor technologies.
- In IBM benchmarks, selecting z15 as the target environment rather than the COBOL V6 default, (z10), delivers CPU savings of up to 23%.

Supported ARCHLVLs by compiler version

ARCH LVL	7	8	9	10	11	12	13	14
Machine Model	2094 (z9 EC) 2096 (z9 BC)	2097 z10 EC) 2098 (z10 BC)	2817 (z196) 2818 (z114)	2827 (zEC12) 2828 (zBC12)	2964 (z13) 2965 (z13s)	3906 (z14) 3907 (z14 ZR1)	8561 (z15) 8562 (z15 T02)	3931 (z16)
Enterprise COBOL 6.4				Y	Y	Y	Y	Y
Enterprise COBOL 6.3		Y	Y	Y	Y	Y	Y	
Enterprise COBOL 6.2	Y	Y	Y	Y	Y	Y		
Enterprise COBOL 6.1	Y	Y	Y	Y	Y			
Enterprise PL/I 6.1				Y	Y	Y	Y	Y
Enterprise PL/I 5.3			Y	Y	Y	Y	Y	
Enterprise PL/I 5.2			Y	Y	Y	Y		
Enterprise PL/I 5.1		Y	Y	Y	Y			
ABO 2.2						Y	Y	Y
ABO 2.1				Y	Y	Y	Y	
ABO 1.3				Y	Y	Y		

History of Enterprise COBOL Performance



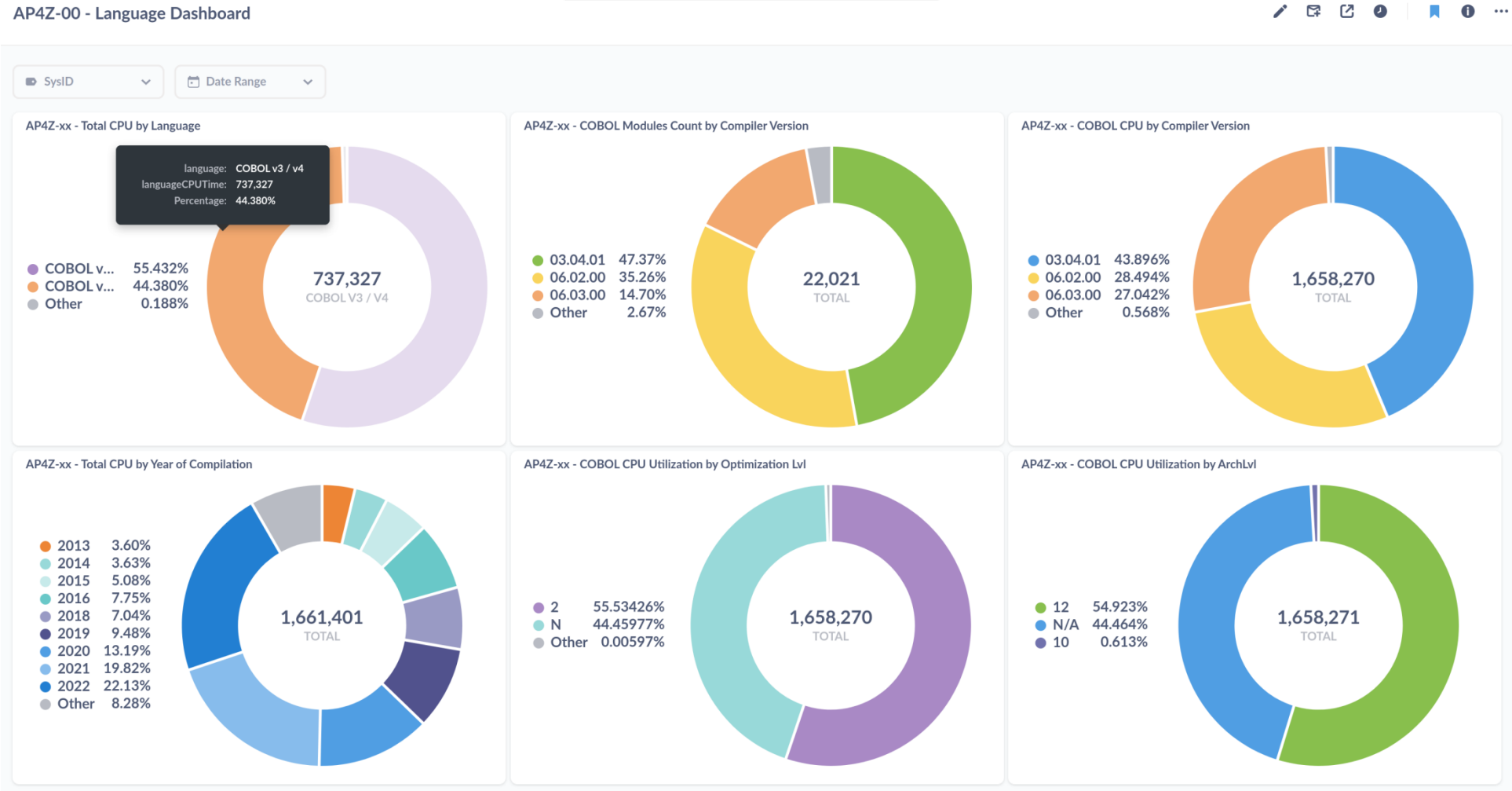
Hardware	IBM zEC12	IBM z13	IBM z13	IBM z14	IBM z15	IBM z16
Compiler	COBOL 4.2	COBOL 5.2	COBOL 6.1	COBOL 6.2	COBOL 6.3	COBOL 6.4
GA year*	2009	2015	2016	2017	2019	2022
Performance options**	OPT(FULL)	ARCH(11) OPT(2)	ARCH(11) OPT(2)	ARCH(12) OPT(2)	ARCH(13) OPT(2)	ARCH(14) OPT(2)
Average CPU usage reduction vs baseline	-	38%	48%	60%	65%	72%

https://www.ibm.com/links?url=https%3A%2F%2Fibm.ent.box.com%2Fv%2FCOBOLMigrationWebinars%2Ffile%2F944448003276

COBOL migration – some considerations

- Programs compiled with COBOL 6 cannot be mixed in an application with OS/VS COBOL programs or with VS COBOL II NORES programs.
- Programs processing invalid data may behave differently when compiled by COBOL 6 .
- IBM recommend migrating using a two phases approach where phase I is focused on spotting and fixing invalid data issues, while phase II compiles and tests programs for deployment in production.
- No need to recompile everything, focus on programs being actively developed, and among them on those using most CPU or being used more frequently first.
- For programs not actively developed IBM offers ABO, the Automatic Binary Optimizer for z/OS.

Assessing the impact of COBOL migration



Assessing the impact of COBOL migration

AP4Z-21 - Very Old COBOL Modules

SysID | Date Range | ModuleName | Resident Option

AP4Z-xx - Very Old COBOL Modules

moduleName	language	compilerVersion	compileDate	compileTime	mainProgram	Reentrant	transactionCount	loadCount	callCount	sampleCount	averageElapsedTime	averageCPUTime	totalCPUTime (Projected)	executionVelocity
SYSTTDIF	VS COBOL II	01.03.02	1997-04-17	13:44:52	N	N/A	26	26	0	0	-	-	-	-
SIM0KADA	VS COBOL II	01.03.02	1998-01-29	14:29:02	N	N/A	1	1	0	0	-	-	-	-
PO00TIME	VS COBOL II	01.03.02	1998-02-04	09:12:45	Y	N/A	154	154	154	154	241,838	825	126,978	0
ELBAUT00	VS COBOL II	01.03.02	1998-04-27	12:51:07	Y	N/A	1	1	1	1	2,728	856	856	0.31
ELBAUT01	VS COBOL II	01.03.02	1999-07-15	13:49:06	Y	N/A	3,561	3,561	3,561	3,561	132,742	1,561	5,559,880	0.01
FUT0R027	VS COBOL II	01.03.02	1999-12-28	15:58:35	N	N/A	24	24	0	0	-	-	-	-
SM0SPACE	VS COBOL II	01.03.02	1999-12-30	10:35:32	N	N/A	4	4	0	0	-	-	-	-
ELBAUTC2	VS COBOL II	01.03.02	2000-02-18	11:44:17	Y	N/A	1	1	1	1	25,093	1,622	1,622	0.06

Powered by Metabase

List of OS/VS COBOL and VS COBOL II modules actually seen being used.

For each of them you can get which Job / Transaction used it, who were the callers

Some good (COBOL) compiler options

- **OPTIMIZE** - Use OPTIMIZE to reduce the run time of your object program. Optimization might also reduce the amount of storage your object program uses.

- **ARCH** - The ARCH option specifies the machine architecture for which the executable program instructions are to be generated. Your application might abend if it runs on a processor with an architecture level lower than what you specified with the ARCH option.

- **RENT** - A program compiled as RENT is generated as a reentrant object program. If all programs in a program object are compiled with RENT, it is recommended that the program object be link-edited with the RENT binder (linkage-editor) option

- **INITCHECK** - Use the INITCHECK option to have the compiler check for uninitialized data items and issue warning messages when they are used without being initialized.

https://www.ibm.com/docs/en/cobol-zos/6.3?topic=ptg-how-tune-compiler-options-get-most-out-v6

Some not so good COBOL compiler options

- **SSRANGE** - SSRANGE generates code that checks whether subscripts try to reference areas outside the region of their associated tables. Variable-length items are also checked to ensure that references are within their maximum defined length. If the SSRANGE option is in effect, range checks will be generated by the compiler and the checks will always be conducted at run time.

- **NUMCHECK** - NUMCHECK tells the compiler whether to generate extra code to validate data items when they are used as sending data items. NUMCHECK is much slower than NONUMCHECK, depending on how many zoned decimal (numeric USAGE DISPLAY) data items, packed decimal (COMP-3) data items, and binary data items are used in a COBOL program.

https://www.ibm.com/docs/en/cobol-zos/6.3?topic=ptg-how-tune-compiler-options-get-most-out-v6

Important program attributes – Reentrancy

- A computer program is considered reentrant if multiple invocations can safely run concurrently using the in memory copy of the program. A reentrant program is always serially reusable, the opposite is not true.

- How to achieve reentrancy: In Assembler language the programmer is responsible of using appropriate techniques to achieve it. For compiled languages the compilers usually takes care of generating reentrant code. If all programs in a program object are compiled with RENT, this can, and should be link-edited with the RENT binder option.

- Why it is important: In a multi-processing environment, like CICS, IMS, or with WLM managed Db2 Stored Procedures, reentrancy allows to avoid reloading the same load module again and again.

    - You must define a Db2 Stored Procedures as STAY_RESIDENT(YES) to achieve the above.

LE storage management model

- Application programs run under **LE threads**. Within a thread execution is serial.
- One or more LE threads are associated to a **LE enclave** or **run-unit**.
- Resources (storage, programs ..) are owned by the enclave and shared with daughter threads.
- Multi-threading under an enclave is supported, but unusual among traditional applications.

- LE manages two types of storage for use by applications:
- HEAPs – used for COBOL WORKING-STORAGE, C malloc(), and PL/I ALLOCATE requests.
- STACKs – used for save areas plus COBOL LOCAL-STORAGE , C and PL/I automatic variables.

A regular batch program runs under a single enclave, The first program run by a CICS transaction, and all the subsequent ones invoked via EXEC CICS LINK each run in a different enclave.

Every new enclave allocates its own HEAP and STACK

How COBOL program storage gets allocated

WORKING-STORAGE is shared among all the programs running under the same enclave.

- It is taken from the LE HEAP when the COBOL run unit (LE enclave) is started.
- Any data items that have VALUE clauses are initialized to the appropriate value at that time. If a VALUE clause is not specified, the initial value of the item is **undefined**.

A separate copy of **LOCAL-STORAGE** data is taken from the LE STACK for each call of a program or invocation of a method, and is returned from the program or method when it ends.

- If you specify a VALUE clause for a LOCAL-STORAGE item, the item is initialized to that value on each call or invocation. If a VALUE clause is not specified, the initial value of the item is **undefined**.

You may ask LE to initialize either or both, but this is **EXPENSIVE** (more later).

Important Runtime Options – RPTSTG

RPTSTG

Derivation: ReP orT ST or aGe

RPTSTG generates, after an application has run, a report of the storage the application used. Language Environment writes storage reports only in mixed-case US English.

Use the storage report information to help you set the ANYHEAP, BELOWHEAP, HEAP, HEAP64, HEAPPOOLS, HEAPPOOLS64, IOHEAP64, LIBHEAP64, LIBSTACK, STACK, STACK64, THREADHEAP, THREADSTACK, and THREADSTACK64 runtime reports for the best storage tuning.

Performance considerations

This option increases the time it takes for an application to run. Therefore, use it only as an aid to application development.

The storage report generated by RPTSTG(ON) shows the number of system-level calls to obtain storage that were required while the application was running. To improve performance, use the storage report numbers generated by the RPTSTG option as an aid in setting the initial and increment size for stack and heap. This reduces the number of times that the Language Environment storage manager makes requests to acquire storage. For example, you can use the storage report numbers to set appropriate values in the HEAP *init_size* and *incr_size* fields for allocating storage.

Important Runtime Options – HEAP (HEAP64)

HEAP

HEAP controls the allocation of the initial heap, controls allocation of additional heaps created with the CEECRHP callable service, and specifies how that storage is managed.

Heaps are storage areas where you allocate memory for user-controlled dynamically allocated variables such as:

- C variables allocated as a result of the `malloc()`, `calloc()`, and `realloc()` functions
- COBOL WORKING-STORAGE data items
- PL/I variables with the storage class CONTROLLED, or the storage class BASED

Non-CICS default

HEAP=((32K,32K,ANYWHERE,KEEP,8K,4K),OVR)

CICS default

HEAP=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR)

Performance considerations

To improve performance, use the storage report numbers that are generated by the RPTSTG runtime option as an aid in setting the initial and increment size for HEAP.

Important Runtime Options – STACK

STACK

STACK controls the allocation of the stack storage of the thread for both the upward and downward-growing stacks. Typical items residing in the upward-growing stack are C or PL/I automatic variables, COBOL LOCAL-STORAGE data items, and work areas for COBOL library routines.

Non-CICS default

STACK=((128K,128K,ANYWHERE,KEEP,512K,128K),OVR)

CICS default

STACK=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR)

Performance considerations

To improve performance, use the storage report numbers generated by the RPTSTG runtime option as an aid in setting the initial and increment sizes for STACK.

Important Runtime Options – STORAGE

STORAGE

STORAGE controls the initial content of storage when allocated and freed. It also controls the amount of storage that is reserved for the out-of-storage condition. If you specify one of the parameters in the STORAGE runtime option, all allocated storage processed by that parameter is initialized to the specified value. Otherwise, it is left uninitialized.

You can use the STORAGE option to identify uninitialized application variables, or prevent the accidental use of previously freed storage. STORAGE is also useful in data security. For example, storage containing sensitive data can be cleared when it is freed.

Non-CICS default

STORAGE=((NONE,NONE,NONE,0K),OVR)

Performance considerations

The use of the STORAGE runtime option to set values within heap and stack storage can have a negative impact on the performance of a Language Environment application.

IBM strongly recommends that you use STORAGE(NONE,NONE,NONE,0K) when you are not debugging, especially with any performance-critical applications. Do not set the STORAGE runtime option to values other than NONE at the system or region level if at all possible, because settings at those levels affect many more programs than necessary. Instead, use language mechanisms to ensure that automatic variables and data structures, including those contained within COBOL LOCAL-STORAGE and WORKING-STORAGE, are properly initialized.

Program storage allocation under CICS

- An automatic LE storage tuning feature is available under CICS, it is called **AUTODST**. When enabled LE monitors the amount of storage used by each program and increases the initial storage allocation for the next execution accordingly. This process helps to minimize the number of GETMAINs and FREEMAINs that CICS has to perform.

  https://www.ibm.com/docs/en/cics-ts/6.1?topic=summary-autodst

- During the execution of a CICS task, every program invoked via EXEC CICS LINK / XCTL runs under a separate enclave. Each enclave's initialization drives multiple getmain / freemain requests for **run-unit** work areas (**RUWA**). CICS can optionally create a run-unit work area pool at task initialization. This reduces the number of GETMAINS and FREEMAINS for tasks that perform many EXEC CICS LINKS. See the **RUWAPOOL** CICS initialization option.

  https://www.ibm.com/docs/en/cics-ts/6.1?topic=summary-ruwapool

Heads Up – HEAPPOOLS (HEAPPOOLS64)

HEAPPOOLS (C/C++ and Enterprise PL/I only)

Derivation: HEAP storage POOLS

The HEAPPOOLS runtime option is used to control an optional heap storage management algorithm known as *heap pools*. This algorithm is designed to improve performance of multithreaded C/C++ applications with high usage of `malloc()`, `__malloc31()`, `calloc()`, `realloc()`, `free()`, `new()`, and `delete()`. When active, heap pools can eliminate contention for heap storage.

Note:

HEAPPOOLS/HEAPPOOLS64 are disabled by default. Using them may benefit multi-threaded applications

A major example:

- AT-TLS (TCP/IP Application Transparent Transport Layer Security) exploits z/OS System SSL.
- AT-TLS is a 64 bit LE application which makes use of multi-threading to support encryption parallelism.
- Using HEAPPOOLS64 for AT-TLS can significantly improve networking latency

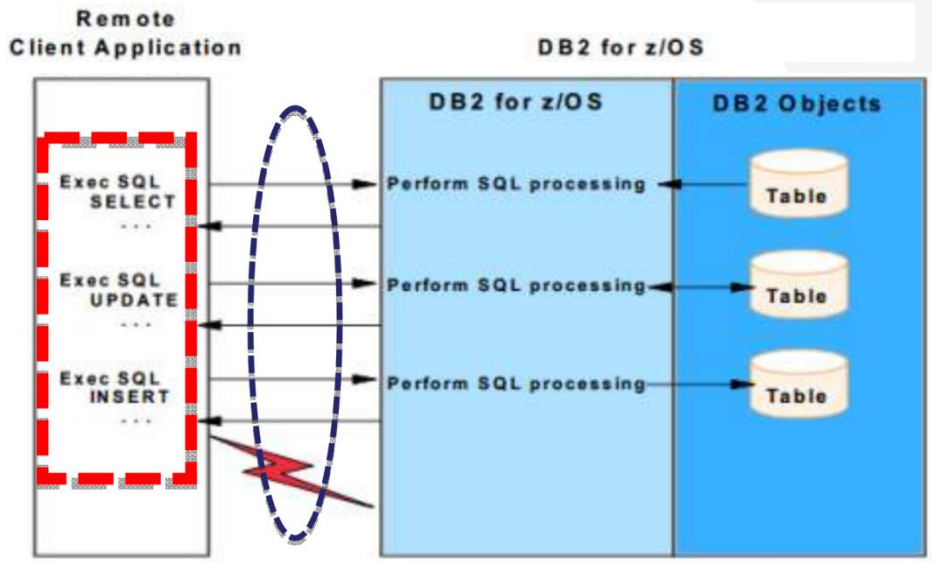
See – "z/OS Communications Server Performance Update" by Mike Fitzpatrick and Dave Herr

Pushing on zIIPs exploitation – Popular Options

- Move business logic off z/OS and access Db2 remotely
  - Eventually reusing existing mainframe logic via Db2 stored procedures
- Run Java based business logic on z/OS
  - Which Java options are available under z/OS?

Move business logic off z/OS using Db2 DRDA

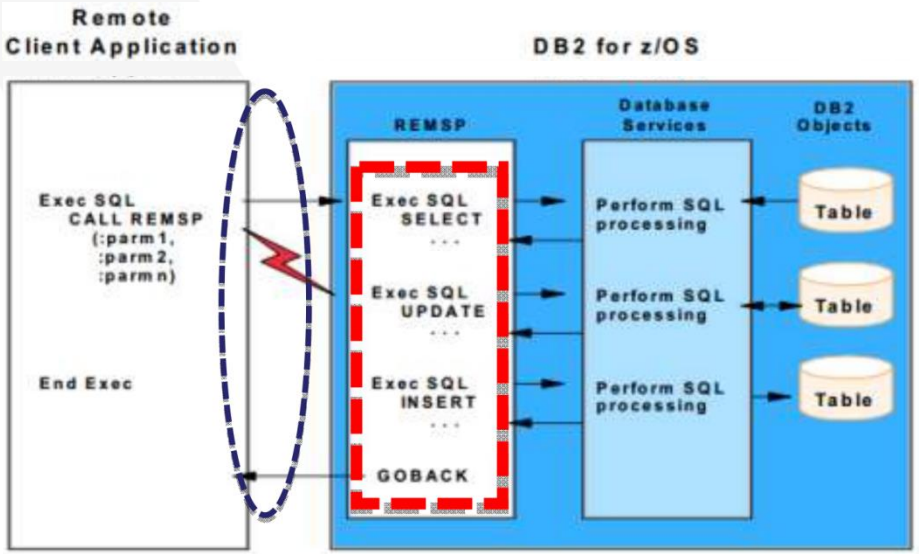
DDF



All business logic moved off to distributed

55-60% of ALL SQL processing is zIIP eligible

DDF + External Stored Procedures



Reusing some existing mainframe business logic

55-60% of Remote SQL processing is zIIP eligible

Java – which options are available under z/OS

IBM provides Java SE and Java EE platforms for z/OS:

- IBM SDK for z/OS, Java Technology Edition, Version 8 provides a Java 8 SE SDK for z/OS and is available in 31 and 64 bit versions.
- “Just” announced IBM Semeru Runtime Certified Edition for z/OS, Version 11.0 provides a Java 11 SE SDK for z/OS and is only available in the 64-bit version.
- Java EE as part of WebSphere full profile and WebSphere Liberty profile.
  - Liberty also shipped as a stand-alone product and as a component of other products (ex. z/OSMF).

Available Java deployment environments

- Java SE applications can be deployed under Batch, CICS, IMS, Db2, and WebSphere Application Server (both Liberty and WAS Full profiles).
- Java EE applications can be deployed under CICS and WebSphere Application Server (both Liberty and Full profiles).
- CICS, IMS and Db2 all make use of persistent Java Virtual Machines. In IMS, if you mix Java and legacy code JVM persistency is subject to the characteristics of the legacy code.
- Under Batch the JVM is created and destroyed for every single program execution (usually a step). Make sure that the associated cost doesn't offset the savings.

Java Interoperability with other languages

Interoperability between Java and “legacy” languages can be achieved in two ways:

- Doing it *the Java way*, with direct language calls through the Java Native interface (JNI).
- Using functionalities of the runtime environment, such as CICS or DB2 (not for Batch).

In the latter case Inter Language Communication (ILC) is transparent to the involved application programs, with middleware taking care of all the associated complexity.

If both the Java part and the legacy part make use of Db2, they need a syncpoint coordinator. CICS, IMS and of course Db2 provide one, Batch doesn’t.

Amode 31 and Amode 64 interoperability

SHARE
EDUCATE • NETWORK • INFLUENCE

PH28966: NEW FUNCTION

A fix is available

Obtain the fix for this APAR.

APAR status

Closed as new function.

Error description

New Function

Local fix

Problem summary

```
**************************************************************
* USERS AFFECTED:                                            *
*  z/OS Language Environment users who need AMODE 31 and AMODE *
* 64 programs interoperability support.                      *
**************************************************************
* PROBLEM DESCRIPTION:                                       *
* APAR PH28966 provides AMODE 31 and AMODE 64 programs       *
* interoperability support. The support provides the         *
* capability to run AMODE 31 programs and AMODE 64 programs  *
* together in an application in the same address space.      *
**************************************************************
* RECOMMENDATION:                                            *
**************************************************************
See problem description.
```

Document Information

Software version:
7B0

Operating system(s):
z/OS

Document number:
6462449

Modified date:
31 July 2021

Subscribe

You can track all active APARs for this component.

Notify me when an APAR for this component changes.

IBM z/OS V2.5: Enabling innovative development to support hybrid cloud and AI business applications

Table of contents

z/OS V2.5 supports the scale and simultaneous deployment of agile business use cases for hybrid cloud and AI capabilities and delivers the following values, features, and capabilities to help organizations succeed in their modernization efforts:

- Enterprise modernization with more seamless COBOL-Java interoperability. This gives application developers full application transparency by extending application programming models.

Needed if you want to intermix legacy 31 bit mode code with Java code running in a 64 bit JVM

Summary

- Application tuning offers very promising saving opportunities.
- But siloed organizations and lack of communication make it hard.
- Inhabitants of ApplicationLand really look similar to us. ☺
- Learning to speak a common language is needed to start knowing each other.

Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

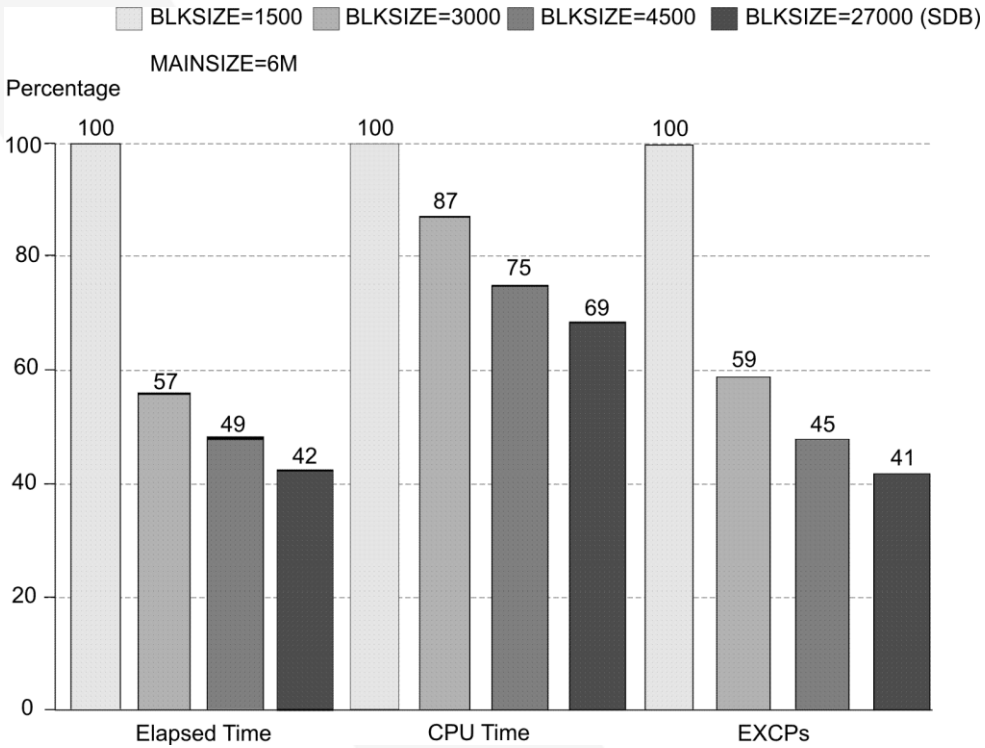
Submit a session evaluation for each session you attend:

www.share.org/evaluation

Backup Slides

Tuning Data Access – Few examples

- PS System Determined Blocksize
  - Use SMF 42.6 to identify candidates and to measure improvements.
- VSAM System Managed Buffers
  - Use SMF 64 to identify candidates and to measure improvements.
  - Requires the dataset to be in extended format, hence SMS managed.
- Large Db2 Buffer Pools
  - Use Db2 Buffer Pool simulation, SMF 100 to measure the benefit.



https://www.ibm.com/docs/en/zos/2.1.0?topic=sizes-io-performance

Db2 Tuning – Most effective actions

- Design databases for performance / design indexes for performance.
- Maintain data organization physically well-organized.
- Maintain Db2 database statistics.
  - The ability of Db2 to choose efficient access paths depends on reliable database statistics
- Manage query access paths.
  - Access paths are among most important aspects of SQL query performance.
- Efficiently use Db2 buffer pools.
- Use dynamic statement caching for dynamic SQL statements
  - Dynamic statement caching saves statements that are already prepared and reuses them.

Routinely measure Db2 performance, Routinely measure Db2 performance, Routinely measure Db2 performance . .

Advanced optimization technology

Automatic Binary Optimizer for z/OS uses advanced technology to optimize COBOL code and targets the latest IBM z/Architecture®. Automatic Binary Optimizer for z/OS helps future-proof the performance of COBOL applications, without the need for recompilation or source code changes when upgrading to new IBM Z hardware.

Automatic Binary Optimizer for z/OS can directly optimize the compiled binary code in COBOL modules, ensuring the COBOL program logic remains the same. Automatic Binary Optimizer for z/OS requires no option tuning, and there are no interoperability concerns after optimization. This allows improvements in the performance of COBOL applications more efficiently and with less testing.

https://www.ibm.com/support/pages/system/files/inline-files/aboz_v22_datasheet.pdf

Testing information

The optimized modules that ABO produces will run faster but will have the same behavior, except from some isolated error message and abend code differences, as the original COBOL modules. ABO is able do this because it processes the binary code within the COBOL module so it is able to ensure the low level logic of the program remains the same. This means that users of ABO do not have to perform full functional verification testing of the ABO optimized modules. Some limited testing is recommended to ensure basic functioning of the applications using the ABO optimized modules prior to deploying ABO optimized modules into a production environment.

https://www.ibm.com/docs/en/abo/2.2?topic=process-testing-information

Automatic Binary Optimizer

IBM Support > Product lifecycle >

Product lifecycle

Product name (** indicates comment, policy exception or more information)	Version	Policy type	Product ID	General availability	End of support
Enterprise COBOL for z/OS	6.4.x	Enhanced	5655-EC6	2022-05-27	
Enterprise COBOL for z/OS	6.3.x	Enhanced	5655-EC6	2019-09-06	
Enterprise COBOL for z/OS	6.2.x	Enhanced	5655-EC6	2017-09-08	
Enterprise COBOL for z/OS	6.1.x	Enhanced	5655-EC6	2016-03-18	2022-09-30
Enterprise COBOL for z/OS (withdrawn)	5.2.x	Enhanced	5655-W32	2015-02-27	2020-04-30
Enterprise COBOL for z/OS (withdrawn)	5.1.x	Enhanced	5655-W32	2013-06-21	2020-04-30
Enterprise COBOL for z/OS (withdrawn)	4.2.x	Standard	5655-S71	2009-08-28	2022-04-30
Enterprise COBOL for z/OS (withdrawn)	4.1.x	Standard	5655-S71	2007-12-14	2014-04-30
Enterprise COBOL for z/OS (withdrawn)	3.4.x	Standard	5655-G53	2005-07-01	2015-04-30

Enterprise COBOL compiler lifecycle

IBM Support > Product lifecycle >

Product lifecycle

Product name (** indicates comment, policy exception or more information)	Version	Policy type	Product ID	General availability	End of support
IBM Enterprise PL/I Value Unit Edition for z/OS **	6.1.0	Enhanced	5655-EP6	2022-05-27	
IBM Enterprise PL/I Value Unit Edition for z/OS	5.3.x	Enhanced	5655-EPL	2019-09-06	
IBM Enterprise PL/I Value Unit Edition for z/OS	5.2.x	Enhanced	5655-EPL	2017-09-08	2022-09-30
Enterprise PL/I for z/OS (withdrawn)	4.5.x	Enhanced	5655-W67	2015-02-27	2020-04-30
Enterprise PL/I for z/OS (withdrawn)	4.4.x	Enhanced	5655-W67	2013-09-06	2018-09-30
Enterprise PL/I for z/OS (withdrawn)	4.3.0	Enhanced	5655-W67	2012-09-28	2017-09-30
Enterprise PL/I for z/OS (withdrawn)	4.2.0	Standard	5655-W67	2011-09-30	2016-09-30
Enterprise PL/I for z/OS (withdrawn)	4.1.0	Standard	5655-W67	2010-09-24	2014-04-30
Enterprise PL/I for z/OS (withdrawn)	3.9.x	Standard	5655-H31	2009-10-30	2015-04-30

Enterprise PL/I compiler lifecycle

IBM Support > Product lifecycle >

Product lifecycle

Product name (** indicates comment, policy exception or more information)	Version	Policy type	Product ID	General availability	End of support
z/OS	2.5.0	Enhanced	5650-ZOS	2021-09-30	
z/OS	2.4.0	Enhanced	5650-zOS	2019-09-30	
z/OS	2.3.x	Enhanced	5650-zOS	2017-09-29	2022-09-30
z/OS (withdrawn)	2.2.x	Enhanced	5650-zOS	2015-09-30	2020-09-30
z/OS (withdrawn)	2.1.x	Enhanced	5650-zOS	2013-09-30	2018-09-30
z/OS (withdrawn) **	1.13.x	Standard	5694-A01	2011-09-30	2016-09-30
z/OS (withdrawn) **	1.12.x	Standard	5694-A01	2010-09-24	2014-09-30
z/OS (withdrawn)	1.11.x	Standard	5694-A01	2009-09-25	2012-09-30

XL C/C++ compiler lifecycle

Product name (** indicates comment, policy exception or more information)	Version	Policy type	Product ID	General availability	End of support
IBM Automatic Binary Optimizer for z/OS **	2.2.0	Continuous Delivery	5697-AB2	2022-05-31	
IBM Automatic Binary Optimizer for z/OS **	2.1.x	Continuous Delivery	5697-AB2	2019-09-06	2023-09-30
IBM Automatic Binary Optimizer for z/OS **	1.3.x	Continuous Delivery	5697-AB1	2017-09-08	
IBM Automatic Binary Optimizer for z/OS (withdrawn) **	1.2.0	Continuous Delivery	5697-AB1	2016-11-11	2020-04-30
IBM Automatic Binary Optimizer for z/OS (withdrawn) **	1.1.0	Enhanced	5697-AB1	2015-11-06	2019-09-30

Automatic Binary Optimizer lifecycle

Portions of certain specific Db2 processes are authorized and eligible for dispatching to IBM® Z Integrated Information Processor (zIIP).

Some or all of the following processes are authorized [1] and eligible for dispatching to IBM Z Integrated Information Processor (zIIP):

SQL request workloads that use DRDA to access Db2 for z/OS® over TCP/IP connections and native REST calls over HTTP

Up to 60% of the Db2 for z/OS instructions executing such SQL requests, when running in Enclave SRB Mode and accessing Db2 for z/OS.

Parallel query child processes

After reaching a CPU usage threshold, up to 100% of the processing of long-running parallel queries for Db2 for z/OS. Long running parallel queries are those which the Db2 for z/OS Query Optimizer determines should be run in parallel and whose execution exceeds an identified period of time as established by Db2 for z/OS (the "CPU usage threshold.") The CPU usage threshold is defined by IBM uniquely for each IBM Z Machine type.

Utility processes

Including the following processes:

- Up to 100% of the portion of LOAD, REORG, and REBUILD INDEX utility function that is used to maintain index structures.
- Portions of RUNSTATS processing, including column group distributed statistics processing

XML processing

Including the following processes:

- Up to 100% of XML schema validation and non-validation parsing.
- Up to 100% of the deletion of unneeded versions of XML documents.

Authorized zIIP uses for Db2 processing

https://www.ibm.com/docs/en/db2-for-zos/12?topic=db2-authorized-ziip-uses-processing