Are You Wasting MIPS? We Can Help You Find Out!

Rodolfo Custodio - rodolfo.custodio@broadcom.com
Todd J. Gagle - todd.gagle@broadcom.com

Disclaimer

Certain information in this presentation may outline Broadcom's general product direction. This presentation shall not serve to (i) affect the rights and/or obligations of Broadcom or its licensees under any existing or future license agreement or services agreement relating to any Broadcom software product; or (ii) amend any product documentation or specifications for any Broadcom software product. This presentation is based on current information and resource allocations as of November 4, 2025, and is **subject to change or withdrawal by Broadcom at any time without notice**. **The development, release and timing of any features or functionality described in this presentation remain at Broadcom's sole discretion**.

Notwithstanding anything in this presentation to the contrary, upon the general availability of any future Broadcom product release referenced in this presentation, Broadcom may make such release available to new licensees in the form of a regularly scheduled major product release. Such release may be made available to licensees of the product who are active subscribers to Broadcom maintenance and support, on a when and if-available basis. The information in this presentation is not deemed to be incorporated into any contract.

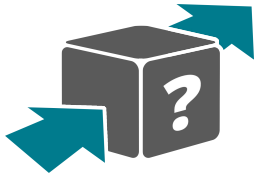
Broadcom may use any feedback provided by you related to a Broadcom product or this presentation for any Broadcom business purposes (including but not limited to, preparation, reproduction, and distribution of derivative works based upon such feedback), without any obligation to you including consent or payment.

THIS PRESENTATION IS FOR YOUR INFORMATIONAL PURPOSES ONLY. Broadcom assumes no responsibility for the accuracy or completeness of the information. TO THE EXTENT PERMITTED BY APPLICABLE LAW, BROADCOM PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. In no event will Broadcom be liable for any loss or damage, direct or indirect, in connection with this presentation, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if Broadcom is expressly advised in advance of the possibility of such damages.

Agenda

- Current Application Landscape
- Introduction to Application Profiler for z - AP4z
- Application Profiler for z - AP4z - Use Cases and Value Proposition
- Q&A

Customer Pain Points & Challenges due to Lack of Application Understanding



Application Visibility

" Mainframe is a **'black box'**. I need better visibility for my mainframe apps & services...."

Assessing Risk of Change

"Mainframe apps can be highly **complex**, making any changes to them is **high risk**."

Optimizing Cost & ROI

"Optimizing **performance and resource utilization** is challenging but has high value."

Impact & Root Cause Analysis

" **Root cause analysis** can be difficult and expensive, impacting resiliency and ops."

There are products that may help here, like source analyzers and execution monitors

Stating the Obvious Pain

SHARE
EDUCATE ▸ NETWORK ▸ INFLUENCE

Program Understanding is critical for:

- NEW PEOPLE joining MF teams for Succession Planning: Seeing ACTUAL control flow accelerates onboarding
- EXPERIENCED PEOPLE who need a refresher if not recently “in that code”
- ALL PEOPLE to understand how frequently used and how critical a module is before making changes

AP4z helps in this area, but it is NOT meant to provide complete Impact Analysis!

- YES, you can see how frequently code is used, thus having better insight into the potential impact of a change
- NO, you CANNOT use execution profiling to state that something is “unused” – need source analysis for that:
  - Programs used millions of times a day will be immediately “seen” by AP4z
  - A program used Feb 29th on leap years will only be “seen” every 1461 days of active profiling!

Stating the Obvious Pain Cont.

SHARE
EDUCATE ▸ NETWORK ▸ INFLUENCE

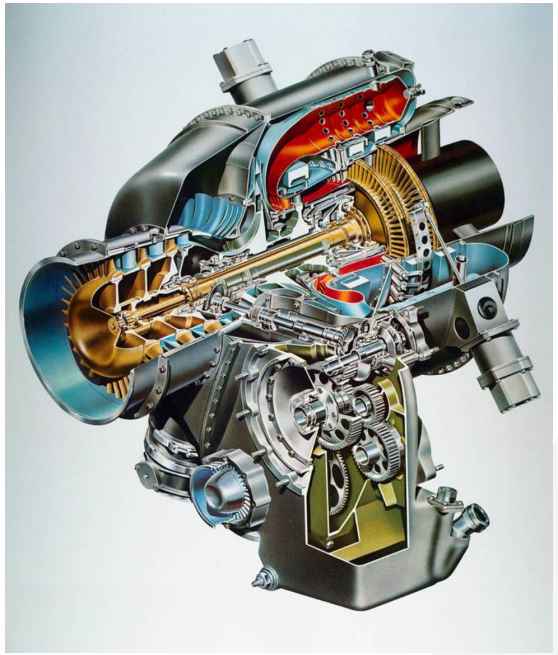
Performance is critical for:

- Client satisfaction (SLAs / Response Time)
- Resource / cost optimization
  - Reducing MIPS / MSU-based costs in MCL, TFP and 4HRA models
  - Headroom for additional workloads even if running Full Capacity

Everyone KNOWS this, BUT too often improvements are not made because of:

- Fear making changes will destabilize the system
- Lack of awareness of potential gains
- (Most common) INERTIA!

Performance and Program Understanding: an Aerospace Analogy



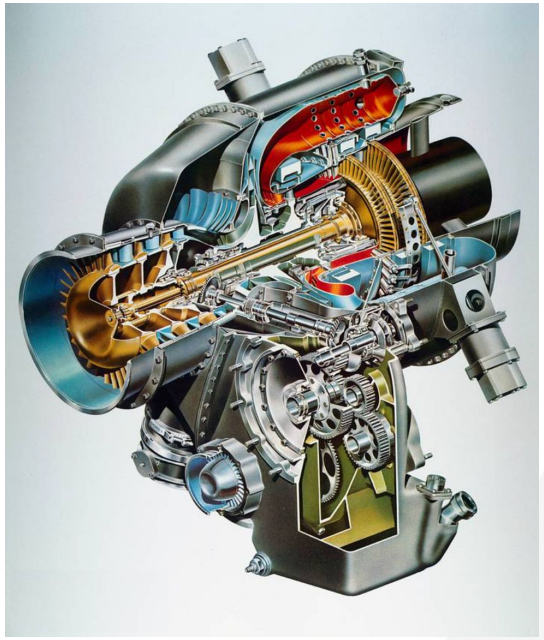
https://itoldya420.getarchive.net/media/jet-engines-965e4f

Program Source Analysis

Static code analysis through examination of source code, provide visibility into everything a program may potentially do and every resource it may potentially use, but it knows nothing about what happens when the program is executed.

```
 9          IDENTIFICATION DIVISION.
10          PROGRAM-ID. APZTC001.
11
12          DATA DIVISION.
13
14          PROCEDURE DIVISION.
15              CALL 'APZTC002'.
16              EXIT PROGRAM.
17          END PROGRAM APZTC001.
```

Performance and Program Understanding: an Aerospace Analogy



https://itoldya420.getarchive.net/media/jet-engines-965e4f

Program Source Analysis

Static code analysis through examination of source code, provide visibility into everything a program may potentially do and every resource it may potentially use, but it knows nothing about what happens when the program is executed.

Struggles with some programming behaviors

```
 9          IDENTIFICATION DIVISION.
10          PROGRAM-ID. APZTC001.
11
12          DATA DIVISION.
13
14          PROCEDURE DIVISION.
15              CALL 'APZTC002'.
16              EXIT PROGRAM.
17          END PROGRAM APZTC001.
```

```
 9          IDENTIFICATION DIVISION.
10          PROGRAM-ID. APZTC001.
11
12          DATA DIVISION.
13          WORKING-STORAGE SECTION.
14             77 PGMNAME PIC X(8) USAGE DISPLAY VALUE 'APZTC001'.
15
16          PROCEDURE DIVISION.
17              CALL PGMNAME.
18              EXIT PROGRAM.
19          END PROGRAM APZTC001.
```

Performance and Program Understanding: an Aerospace Analogy

SHARE
EDUCATE • NETWORK • INFLUENCE

In-depth Program Monitoring: MAT

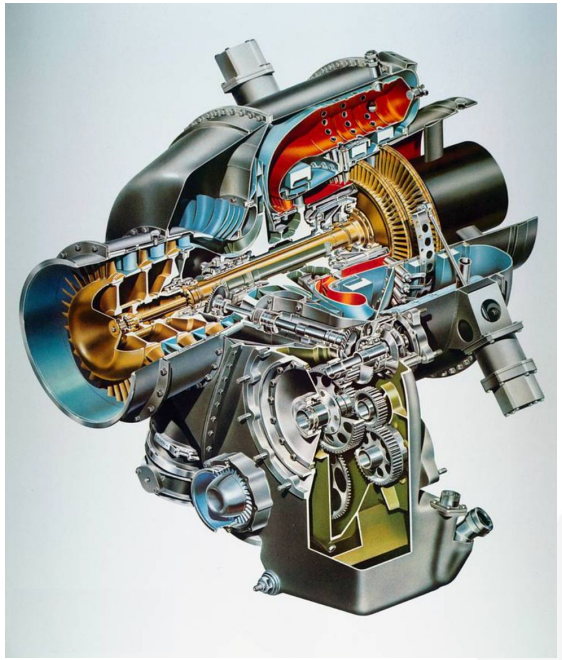
Very powerful tools that allow to perform deep dive analysis of specific programs' behavior when executed ,down to the single instruction. They also see I/O, subsystems usage etc.

Due to their overhead though, they can only be used "on demand" against a limited set of programs at a time.

https://nara.getarchive.net/media/senior-airman-alex-sharpe-911th-aircraft-maintenance-e4b4dd

In-depth Program Monitoring

Performance and Program Understanding: an Aerospace Analogy



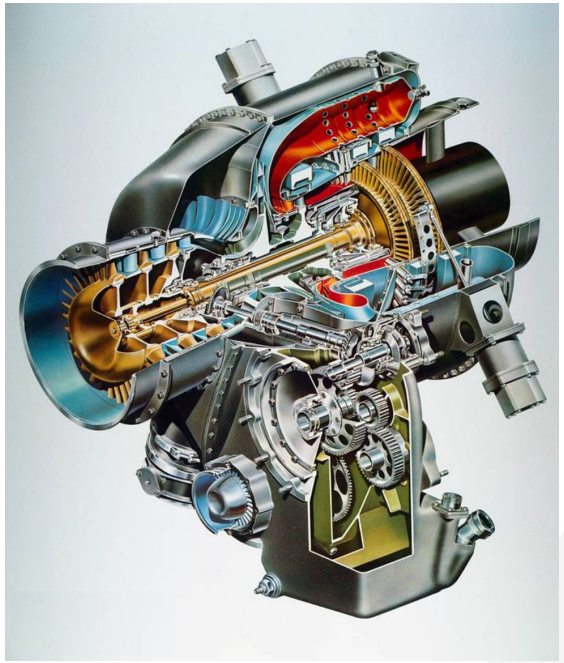
https://itoldya420.getarchive.net/media/jet-engines-965e4f

Program Source Analysis

https://nara.getarchive.net/media/senior-airman-alex-sharpe-911th-aircraft-maintenance-e4b4dd

In-depth Program Monitoring

Performance and Program Understanding: an Aerospace Analogy



https://itoldya420.getarchive.net/media/jet-engines-965e4f

https://nara.getarchive.net/media/a-crewman-uses-an-air-traffic-control-radar-atcr-screen-aboard-the-nuclear-24fa2e

https://nara.getarchive.net/media/senior-airman-alex-sharpe-911th-aircraft-maintenance-e4b4dd

Program Source Analysis

Continuous Application Profiling

AP4z (Broadcom Application Profiler for z)

In-depth Program Monitoring

MAT (Broadcom Mainframe Application Tuner)

Performance and Program Understanding: an Aerospace Analogy

https://nara.getarchive.net/media/a-crewman-uses-an-air-traffic-control-radar-atcr-screen-aboard-the-nuclear-24fa2e

Continuous Application Profiling: AP4z and no others

Lightweight monitoring to track the actual execution of entire application workloads, and measure a limited number of noticeable activities like for example dynamic program calls, identifies active modules, their attributes, inter-relationships, call frequency and duration.

Doesn't compete with in-depth program monitoring that provides much more information for a much narrower span, rather it complements and potentially feeds it.

AP4z – The Application Profiler for z

SHARE
EDUCATE • NETWORK • INFLUENCE

AP4z is the only available z/OS tool with the following capabilities:

- Specifically designed to profile applications' execution at scale.
- Aimed at providing a full picture about programs / subprograms and their actual relationships.
- Able to report Elapsed Time and CPU consumption at the individual (sub)program level.
- Automatically collecting each active module's compile date, used compiler release, and compiling options.
- Able to build a call graph showing who actually calls who.
- Simple, with no special system requirements, and with a negligible impact in terms of CPU consumption.
- Able to build a long term history of active programs their performance, their relationship to each other.

AP4z exploits the Language Environment Profiling Interface

Believe it or not...

SHARE
EDUCATE • NETWORK • INFLUENCE

AP4z can be run:

- Continuously (to get the most value by looking at historical trends, behavior changes)

Because its overhead is negligible:

- We assert the overhead is <1%...
- But real client examples show <0.1%. The overhead is MINUSCULE!

AP4z and Application Optimization

SHARE
EDUCATE • NETWORK • INFLUENCE

Problem Statement:

Performance analysts want to optimize applications to consume fewer resources. Accurate measurements based on real production metrics allow to prioritize efforts to achieve a better ROI.

With AP4z you can:

- Identify programs consuming too many resources
- Focus on programs being used
- Identify old compiler versions for quick gains
- Measure baseline for ROI comparison

Supported Use Cases:

- **Optimization**
- Modernization
- Troubleshooting

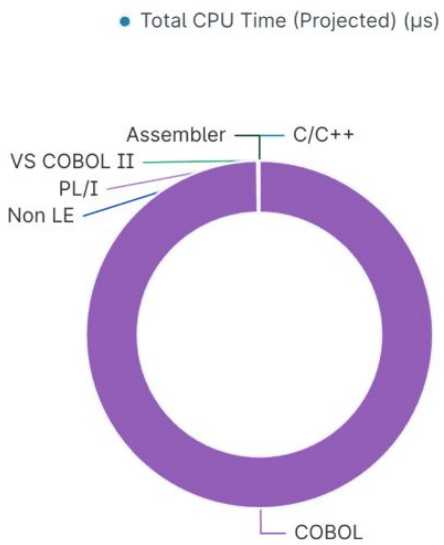
AP4z and Application Optimization

Language Dashboard Uncover actual usage (CPU and No. of modules) by compiler versions

Filter

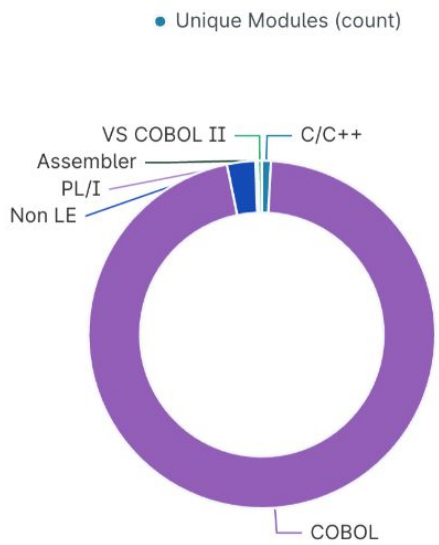
Language and CPU | Year of Compilation and CPU | COBOL Modules by Architecture Level

Projected CPU Usage By Language



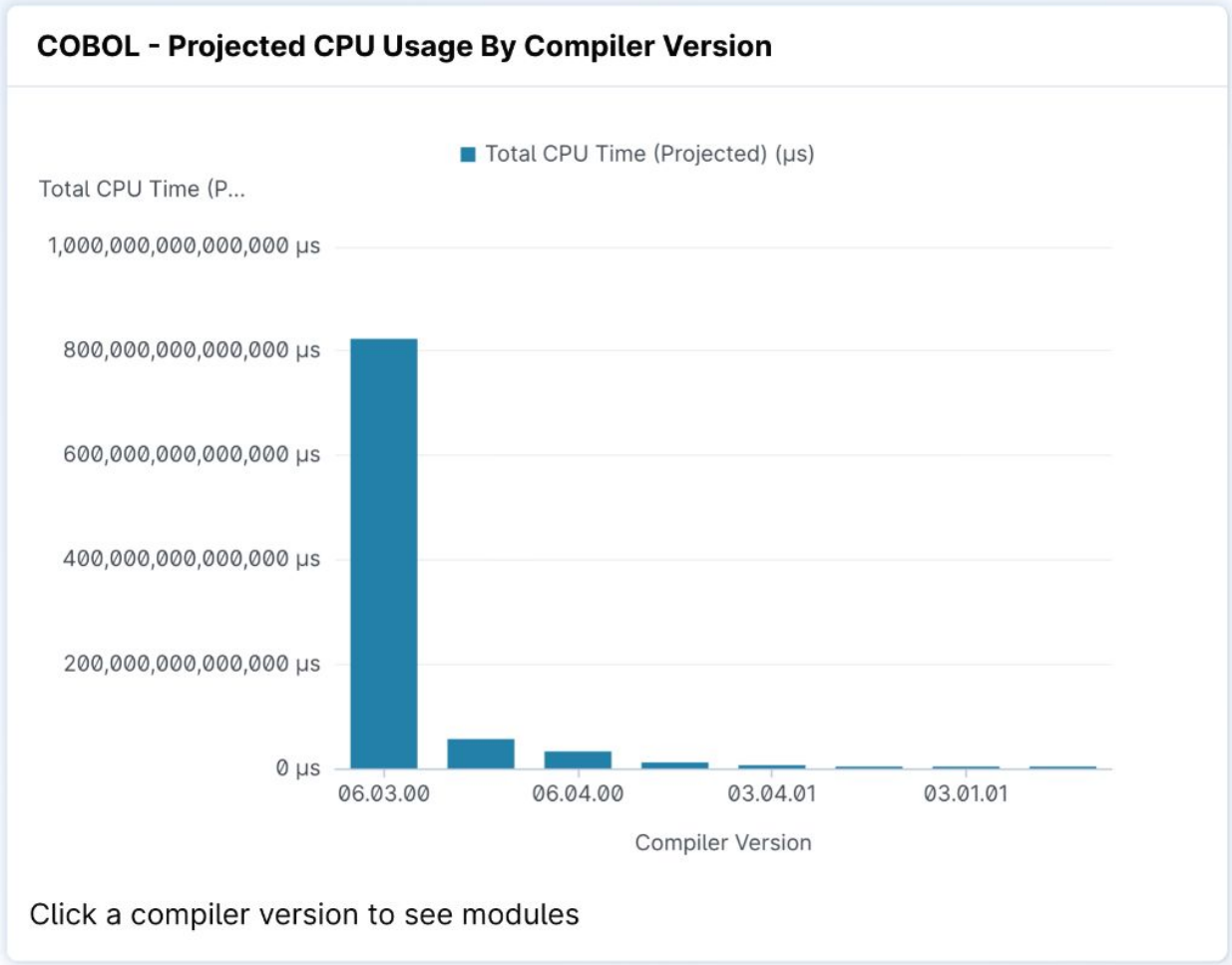
Click a language to see compiler versions

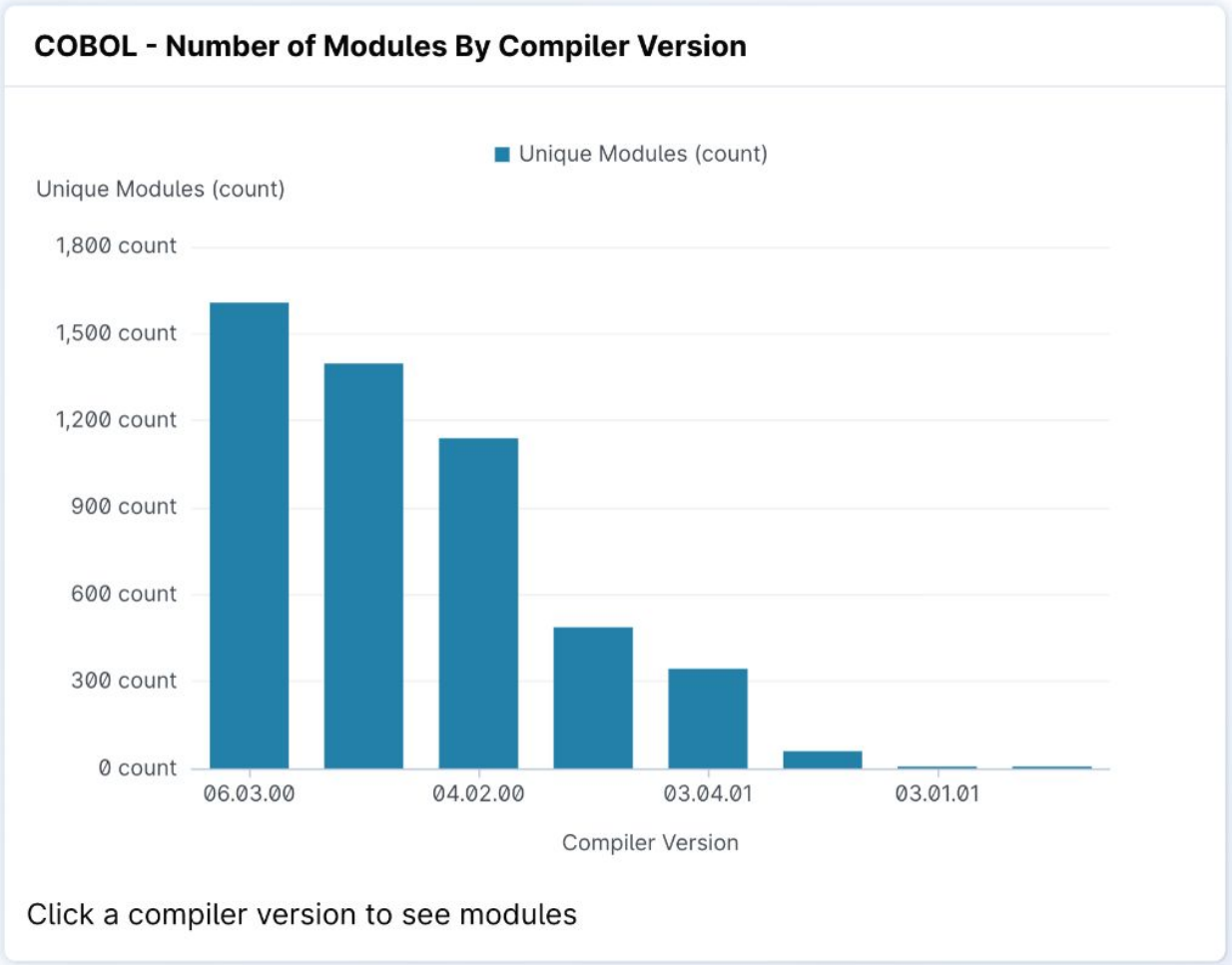
Number of Modules By Language



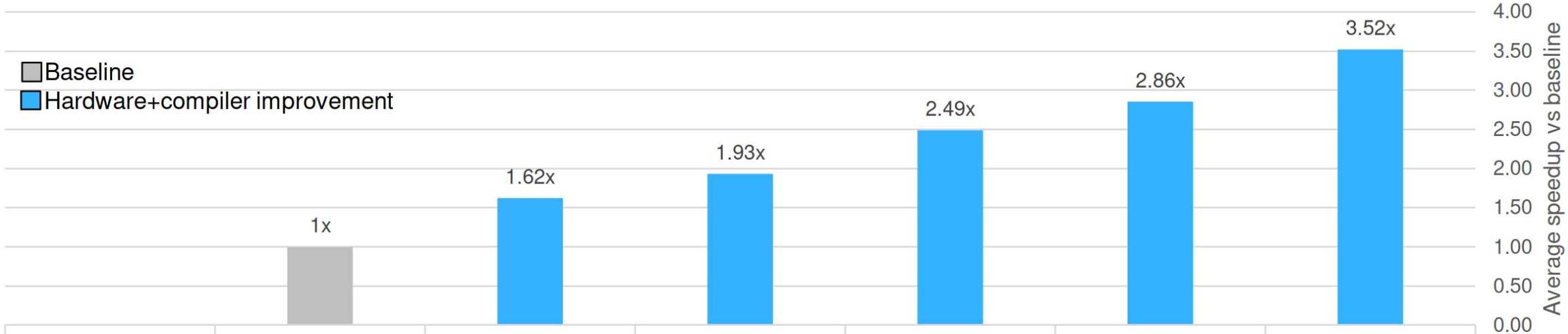
Click a language to see compiler versions

AP4z and Application Optimization





AP4z and Application Optimization



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

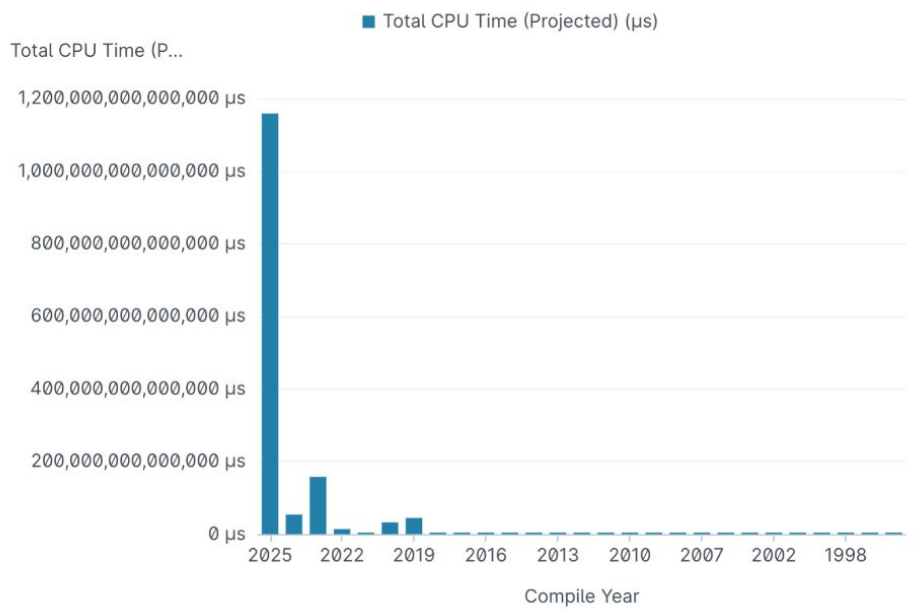
AP4z and Application Optimization

Language Dashboard Uncover actual usage (CPU and No. of modules) by compiler versions

Filter

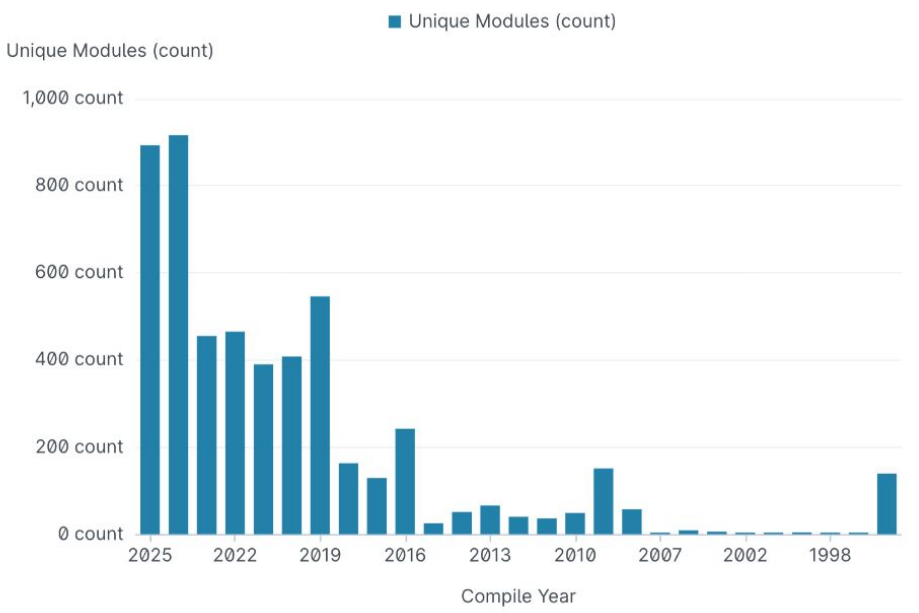
Language and CPU | **Year of Compilation and CPU** | COBOL Modules by Architecture Level

Total Projected CPU by Year of Compilation



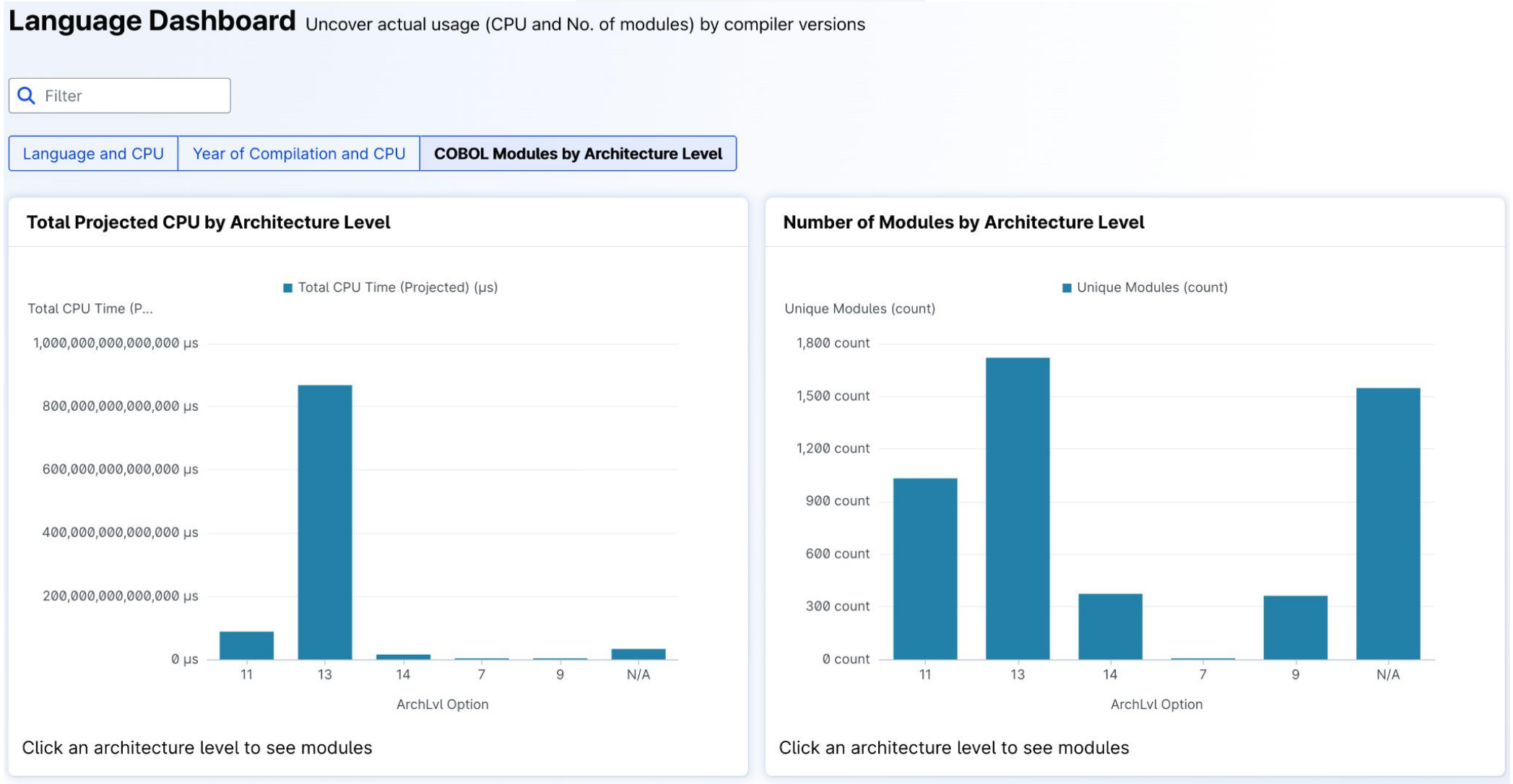
Click a year of compilation to see modules

Number of Modules by Year of Compilation



Click a year of compilation to see modules

AP4z and Application Optimization



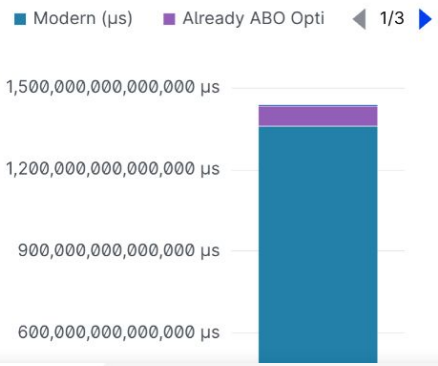
AP4z and Application Optimization

COBOL Modules & ABO Optimization

Legacy means COBOL V4 & earlier. Modern means COBOL V5 & later.

Legacy COBOL Projected CPU Distribution

Contribution to Total Projected CPU Time



COBOL Module Count by ABO Status

Modern (count)	Already ABO Optimized (count)	ABO Candidates (count)
3,492	735	812

ABO Candidates

Showing the highest CPU-consuming, unoptimized, ABO-eligible modules

Filter

Language equals: COBOL

Full Screen Table

System Name	Module Name	Detected Execution Environment	ABO Optimized	COBOL Generation	Total CPU Time (Projected) (µs)
ZCS8	Y73036B	Batch	N	Modern (V5 & later)	9,326,800
ZCS9	ZQIPARB	Batch	N	Legacy (V4 & earlier)	1,383
ZCS8	L7QDMR0A	Batch	N	Modern (V5 & later)	5,312
ZCS9	A9QDLAE1	Batch	N	Legacy (V4 & earlier)	11,730
ZCS8	AZPBOPI3	Batch	N	Modern (V5 & later)	19,259
ZCS8	SBGE0010	Batch	N	Modern (V5 & later)	54,266,710

1 - 15 of 4820

1 2 3 4 5

AP4z and Application Understanding

Problem Statement:

Mainframe applications are complex and layered, less and less people understand how they work. This makes modernization projects risky, slow and difficult to start.

With AP4z you can:

- Identify and visualize actual relationships between programs and applications
- Detect unexpected relationships, enforce application boundaries
- Assess resource usage and complexity of applications candidate to modernization

Supported Use Cases:

- Optimization
- **Modernization**
- Troubleshooting

AP4z and Application Understanding

Dynamically Loaded Modules

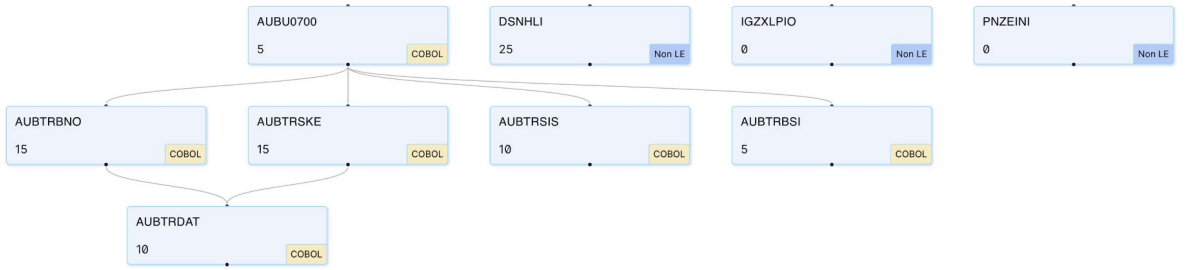
Display as Graph | Filter | System Name equ... ZCS8 | Job Name equals AUOEB4... | Job Id equals J00080... | Job Start Timestamp e... 2025-06-... | Step Start Timestamp e... 2025-06-... | Step Name equals KI9LE... | +2

System Name	Module Name	Language	Compiler Version	Compile Timestamp	Main Program	# Loads (count)	# Calls (count)	# Executions (count)	Total CPU Time (Projected) (µs)	Total Elapsed Time (µs)
ZCS8	AUBU0700	COBOL	06.04.00	2025-06-26 18:05:06	Y	5	0	5	6,853,918,720	12,499,726,090
ZCS8	AUBTRSKE	COBOL	06.03.00	2024-07-10 17:25:25	N	5	15	0	6,745	30,580
ZCS8	AUBTRBSI	COBOL	06.01.00	2021-12-09 12:33:13	N	5	5			
ZCS8	DSNHLI	Non LE	N/A		N	5	25			
ZCS8	AUBTRSIS	COBOL	04.02.00	2014-11-24 16:33:14	N	5	10			
ZCS8	AUBTRBNO	COBOL	03.04.01	2010-11-23 19:24:43	N	5	15			
ZCS8	AUBTRDAT	COBOL	04.02.00	2014-11-20 22:57:53	N	5	10			
ZCS8	IGZXLPIO	Non LE	N/A		N	5	0			
ZCS8	PNZEINI	Non LE	N/A		N	5	0			

1 - 9 of 9

Batch Call Graph

Display as Table | Layout | Filter | System Name equ... ZCS8 | Job Name equals AUOEB4... | Job Id equals J00080... | Job Start Timestamp e... 2025-06-... | Step Start Timestamp e... 2025-06-... | +3



AP4z and Application Understanding

Modules Called by SBGE9501

Filter | System Name equ... ZCS9 | Origin Module Name ... SBGE95...

Origin Module Name	Target Module Name	Origin Is Main	# Loads (count)	# Executions (count)	# Samples (count)	Total Elapsed Time (µs)	Total CPU Time (µs)
SBGE9501	SBGE0004	N	4	0	67	57	56
SBGE9501	SBSB016A	N	0	0	5,994	1,210	1,063

Modules Calling SBGE9501

Filter | System Name equ... ZCS9 | Target Module Name ... SBGE95...

Origin Module Name	Target Module Name	Origin Is Main	# Loads (count)	# Executions (count)	# Samples (count)	Total Elapsed Time (µs)	Total CPU Time (µs)
SBGB9500	SBGE9501	N	0	0	2,213	43,909,915	37,813,003

AP4z and Troubleshooting

SHARE
EDUCATE • NETWORK • INFLUENCE

Problem Statement:

Root cause analysis of poorly understood applications is difficult, negatively impacting resiliency, increasing cost of operations and time to resolution for operators and SMEs.

With AP4z you can:

- See what the abending transaction did up to the point it failed - **FFDC**
- Assess module level CPU usage trends
- Identify programs changing their behavior and when
- Detect potential security exposure: Unexpected program runs, version changes
- Focus your quality assurance tests, deploy better quality applications

Supported Use Cases:

- Optimization
- Modernization
- **Troubleshooting**

AP4z and Troubleshooting

Modules with Multiple Versions

Filter

Unique Modules > 1

System Name	Module Name	Detected Execution Environment	Unique Modules (count)
ZCS8	UIBBUT79	Batch	23
ZCS9	UIBBUT77	Batch	14
ZCS8	UIBBUT77	Batch	13
ZCS9	LCTEST18	Batch	12
ZCS8	LCR006	Batch	12
ZCS9	LCR006	Batch	11
ZCS8	HZ9B0609	Batch	9
ZCS8	LCTEST18	Batch	8
ZCS8	HJBA29	Batch	7
ZCS9	UIBBUT26	Batch	7
ZCS8	J0B50XCT	Batch	6

1 - 15 of 124

1 2 3 4 5

AP4z and Troubleshooting

Batch Abend Causes

Filter

Error Code	Abend Completion Code	Abend Reason	Abend Dump Requested	# Abends (count)
CEE3380W	U4038		No	1,618
IBM0181S	U4038		No	193
IBM0368W	U4038		No	128
CEE3207S	S199	07	Yes	36
IGZ0006S	U4038		No	36
IGZ0201W	U4038		No	35
CEE3250C	U1		No	25
EDC6010S	U4038		No	24
IGZ0035S	U4038		No	12
IGZ0017S	U4038		No	8
CEE3250C	U91		No	4

1 - 15 of 16

1 2

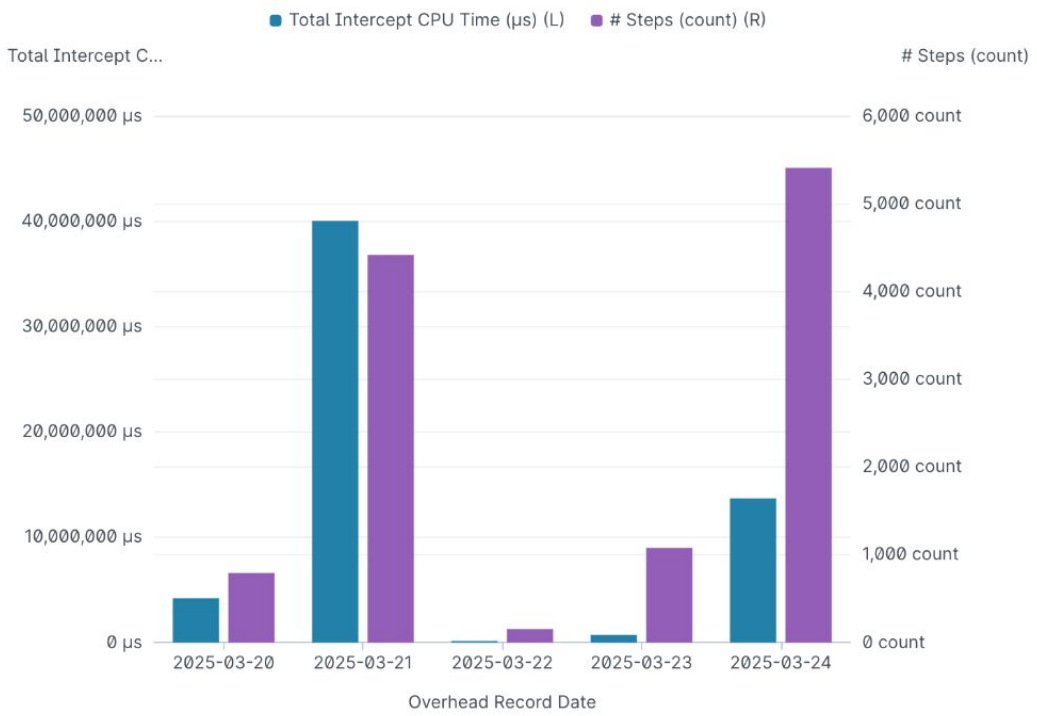
AP4z Overhead

Profiler Summary

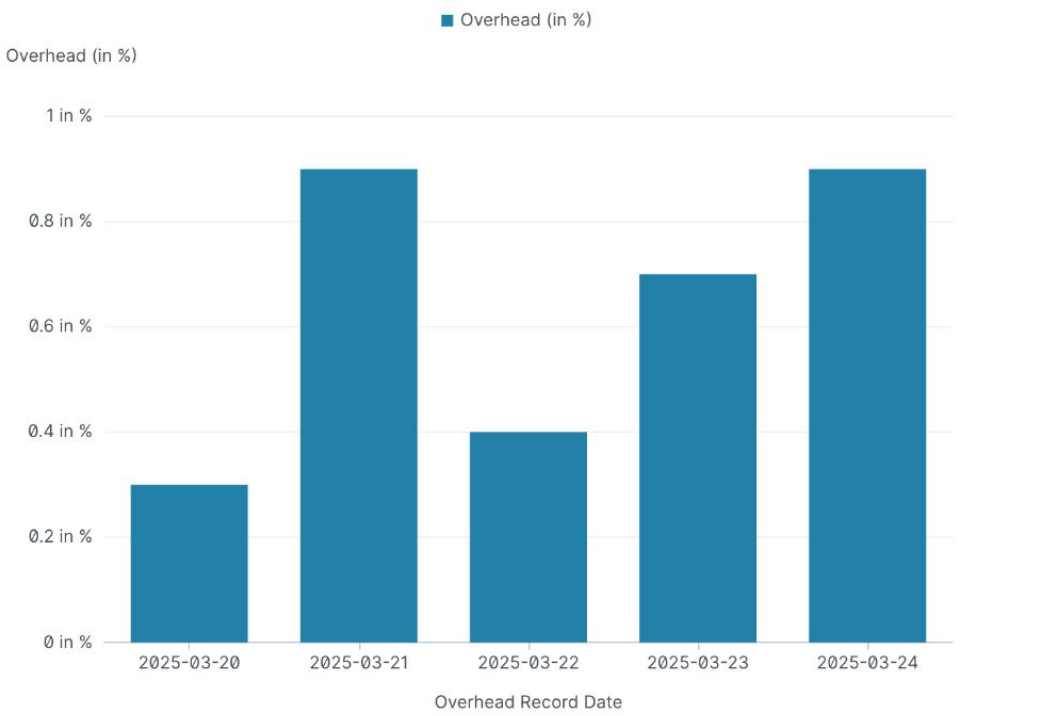
AP4z Profiling Costs | AP4z Internal Metrics

The average cost of running WatchTower Application Profiler in your Batch environment should be less then 1% of the profiled workload.

Total CPU Time and Step Count



Average Profiling Cost %



AP4z Coverage

Supported Languages:

- COBOL, PL/I, C/C++, Fortran and other AMODE 24/31 compiled languages
- Assembler programs using LE
- Assembler programs not using LE but directly called by higher level language programs

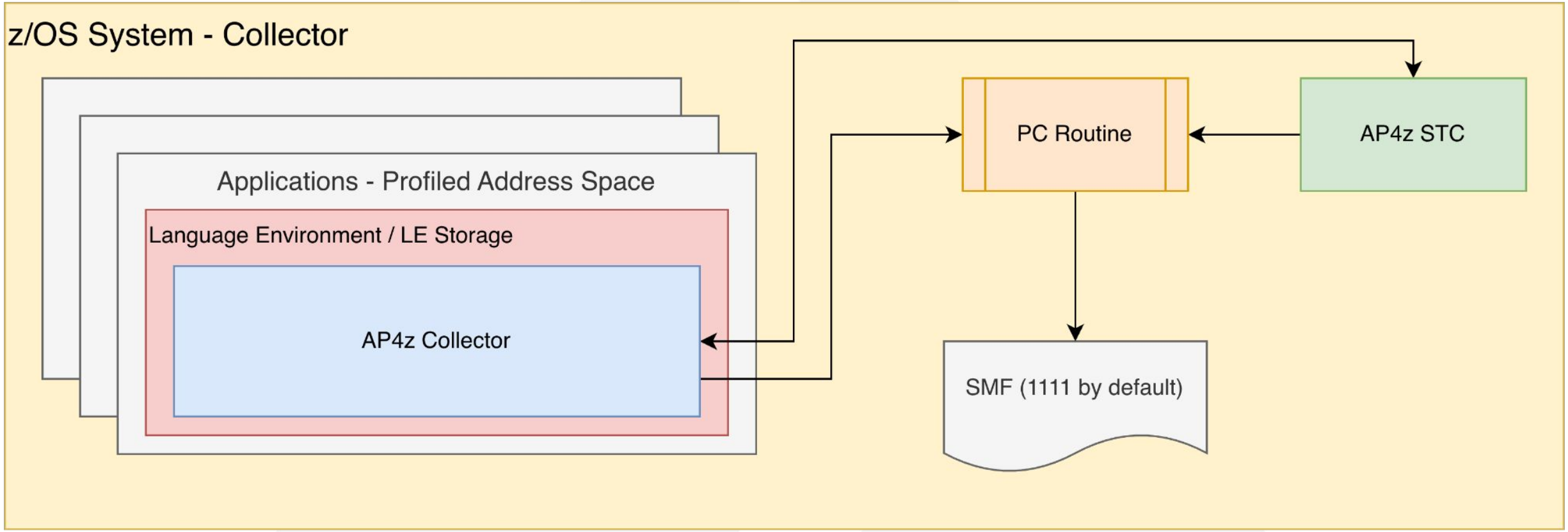
Supported Runtime Environments:

- Batch, TSO, USS & CICS runtime environments supported
- IMS™ and Db2 External StorProcs support **in the development roadmap**

Not profiled by AP4z:

- Assembler programs not using LE services nor called by higher level language programs
- Interpreted languages, including Rexx, Java, Python; AMODE 64 programs including Go, Node.JS,
- Static program calls but it is **in the development roadmap**

AP4z Collector Architecture



The Language Environment PROFILE Runtime Option

Profiling can be activated at the individual job step level using the CEEOPTS DDNAME in the execution JCL:

```
//CEEOPTS   DD    *
PROFILE(ON,"options string")
//*
```

Alternatively it can be activated system wide, dynamically, using the SETCEE command:

```
SETCEE CEEDOPTS, PROFILE(ON,"options string")
```

Or permanently, system wide, using a CEEPRMxx PARMLIB member:

```
CEEDOPT(
         PROFILE(ON,"options string")
        )
```

Be Part of AP4z vNext Pre-GA

Direct Influence on Product Design

- Provide direct feedback to the development team.
- Final product solves your specific mainframe performance and utilization challenges.

Early Access to Innovations

- Gain a first-mover advantage by exploring advanced profiling capabilities.

Validation with your Own Data

- Validate the AP4z solution within your own unique environment, reducing technical uncertainty before committing to a full-scale (DEV, QA, PROD) deployment.

Q & A

Join a Mainframe Technical Exchange

SHARE
EDUCATE • NETWORK • INFLUENCE

Apr. 21-23, 2026	June 23-25, 2026	Oct. 20-22, 2026
European MTE Prague, Czech Republic	Virtual MTE Held Virtually	North American MTE Plano, TX

- Network with peers and Mainframe technical experts
- Participate in technical how-to sessions and hands-on workshops
- No registration fee!

Recommended Observability Sessions

Monday

09:45-10:05	GenAI on Mainframe: The Augmentation, Not the Replacement	Salon 20
13:15-14:15	How to Integrate and Manage Alerts with WatchTower and Mainframe Data Sources	Salon 21

Tuesday

09:15-10:15	Beyond Silos: Leveraging WatchTower and ServiceNow for Unified and Effective Mainframe ITSM	Salon 21

Wednesday

09:15-10:15	Are You Wasting MIPS? We Can Help You Find Out!	GS Room 2
13:15-14:15	Demystifying Kubernetes: From PARMLIBs to Pods	GS Room 2
14:30-15:30	Mainframe Observability with OpenTelemetry and OPS/MVS	GS Room 2

Thursday

10:30-11:30	Leveraging Your Domain Data for Observability	Salon 14
13:15-14:15	Come Together With the Fantastic Four: Optimization, Performance, Cost and AI	Salon 14

Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation