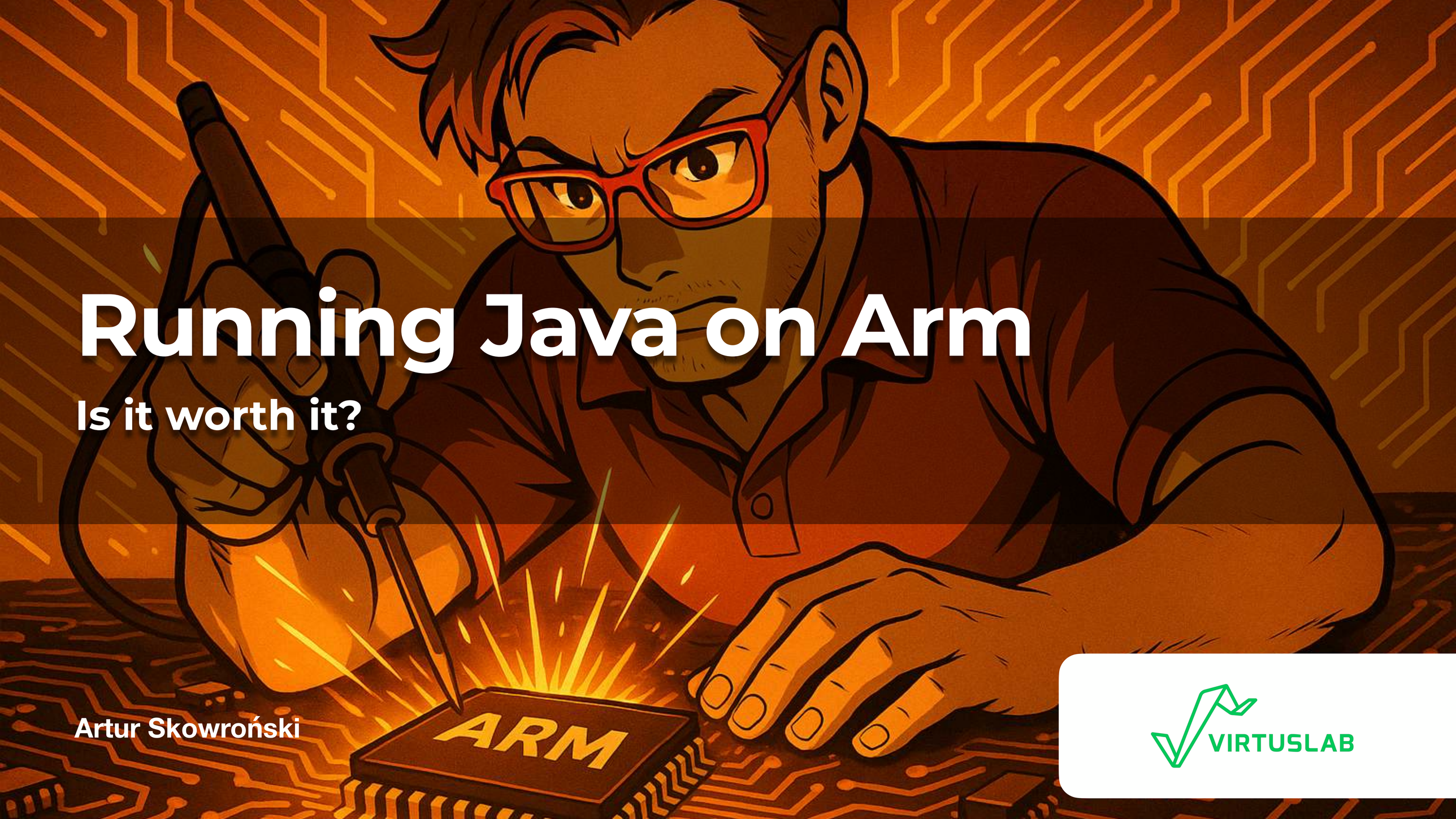


Running Java on Arm

Is it worth it?

Artur Skowroński

VIRTUSLAB

Artur Skowronski

Head of Java / Kotlin Development

VIRTUSLAB

SCALA 3.5.0 RELEASED!

Thursday 22 August 2024

Paweł Marks, VirtusLab

Contents

New default runner - Scala CLI
- Short example

What's new in 3.5.0?
- Best effort compilation
- Support for pipelined builds
- var in refinements
- Support for binary integer literals
- Experimental: named tuples
- Experimental: new givens and context bounds syntax

Work on a better scheme for given prioritization

What's next?

Contributors

Problem with this page? Please help us fix it!

JVM Weekly

Home | Notes | Foojay.io Community Newsletter | Archive | About

"Optimizing the GC when Migrating Cloud Workloads to Arm" with Kieran Hejma...

Today, we have a guest post from Kieran Hejmadi, an Software and Academic Ecosystem Manager in Arm.

MAY 1 · ARTUR SKOWRONSKI AND KIERAN HEJMADI

Latest | Top | Discussions

April - The Rest of the Story - JVM Weekly vol. 127

I know there is a lot of April left, not my fault, calendar is working against my publishing schedule this month...

APR 24 · ARTUR SKOWRONSKI

Java in AWS with Wojtek Gawroński - JVM Weekly vol. 126

I'm experimental mood recently, so today, we have the interview!

APR 17 · ARTUR SKOWRONSKI AND WOJTEK GAWROŃSKI

Make Final Mean Final... and the other JEPs - JVM Weekly vol. 125

It's been a while since we've seen JEPs! But you know, not the fully baked ones, not the ones that have advanced far i...

JVM Weekly

From the latest updates in JVM languages like Java, Kotlin, and Scala to emerging technologies like GraalVM and Quarkus, this newsletter covers a wide range of topics that are of interest to developers and tech enthusiasts.

✓ Subscribed

Recommendations MANAGE

- **Tales from the jar side** — Ken Kousen
- **Architecture Weekly** — Oskar Dudycz
- **Engineering Primer** — Bartek Antoniak
- **Tech Talks Weekly**

When someone tells me consulting sounds like googling with extra steps

I’m mostly involved in Software Consulting

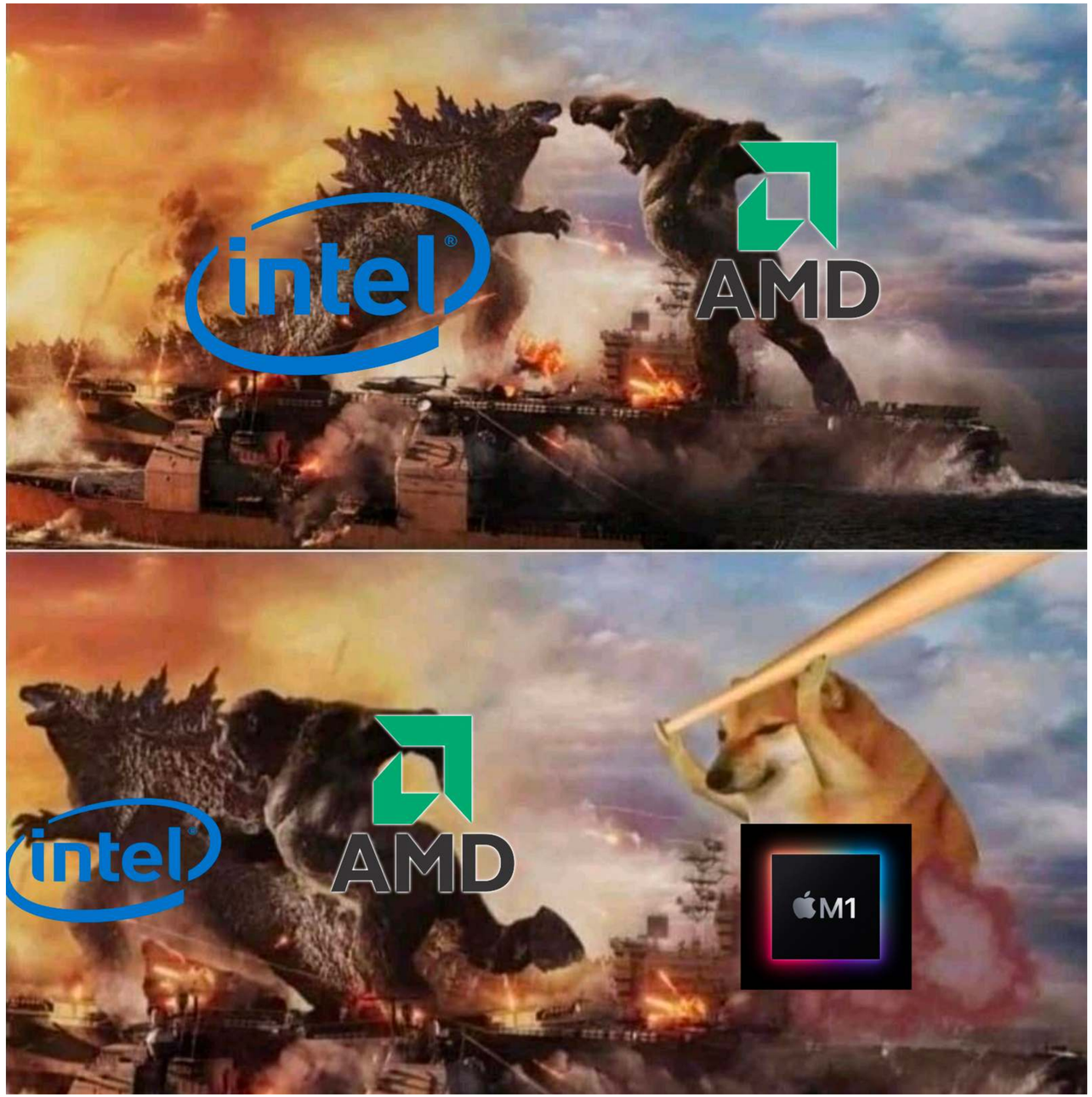
A lot of different optimizations we tried

Low Hanging Fruits

Stopy zwrotu: 10 lat: **+20 652.95%**, YTD: -17.34%

Everybody recently loves Silicon

YOU'D LIKE SOMEONE TO LOOK AT YOU ONE DAY
THE WAY CRAIG FEDERIGHI
LOOKS AT A NEW MACBOOK
imgflip.com


intel
AMD
intel
AMD
M1

Has x86 lost the data center battle? Arm claims victory as it declares close to 50 percent of compute shipped to top hyperscalers in 2025 will be Arm-based

News By Wayne Williams published April 5, 2025

AI demand is behind the Neoverse architecture's meteoric rise

Comments (1)

When you purchase through links on our site, we may earn an affiliate commission. Here's how it works.

(Image credit: Shutterstock / gguy)

- Arm says Neoverse powers nearly half of 2025 hyperscaler compute shipments
- AI growth makes power-efficiency essential for hyperscale data centers
- AWS, Google, Microsoft are leading shift from x86 to Arm infrastructure

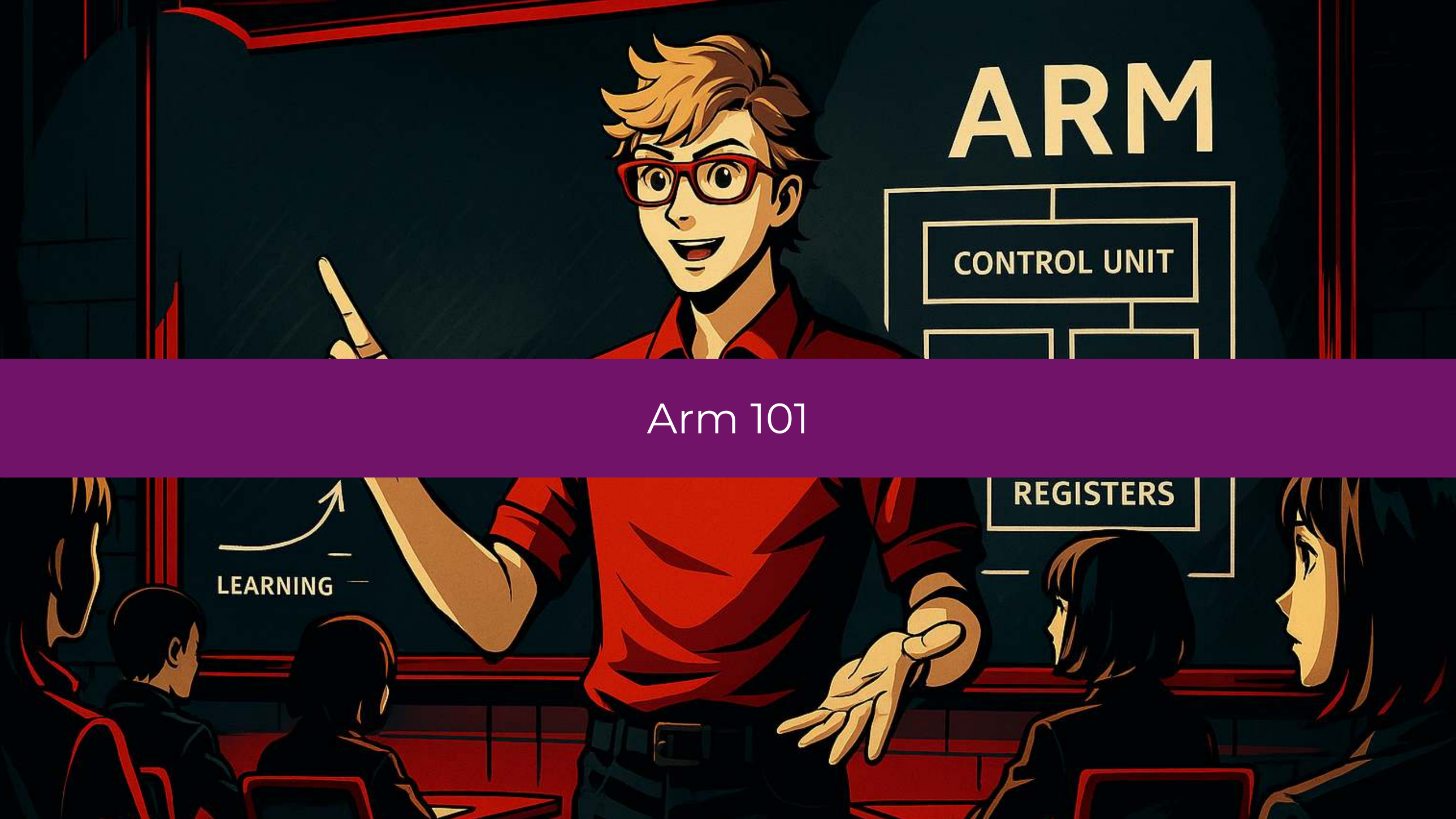
ARM 101

WHY NOW?

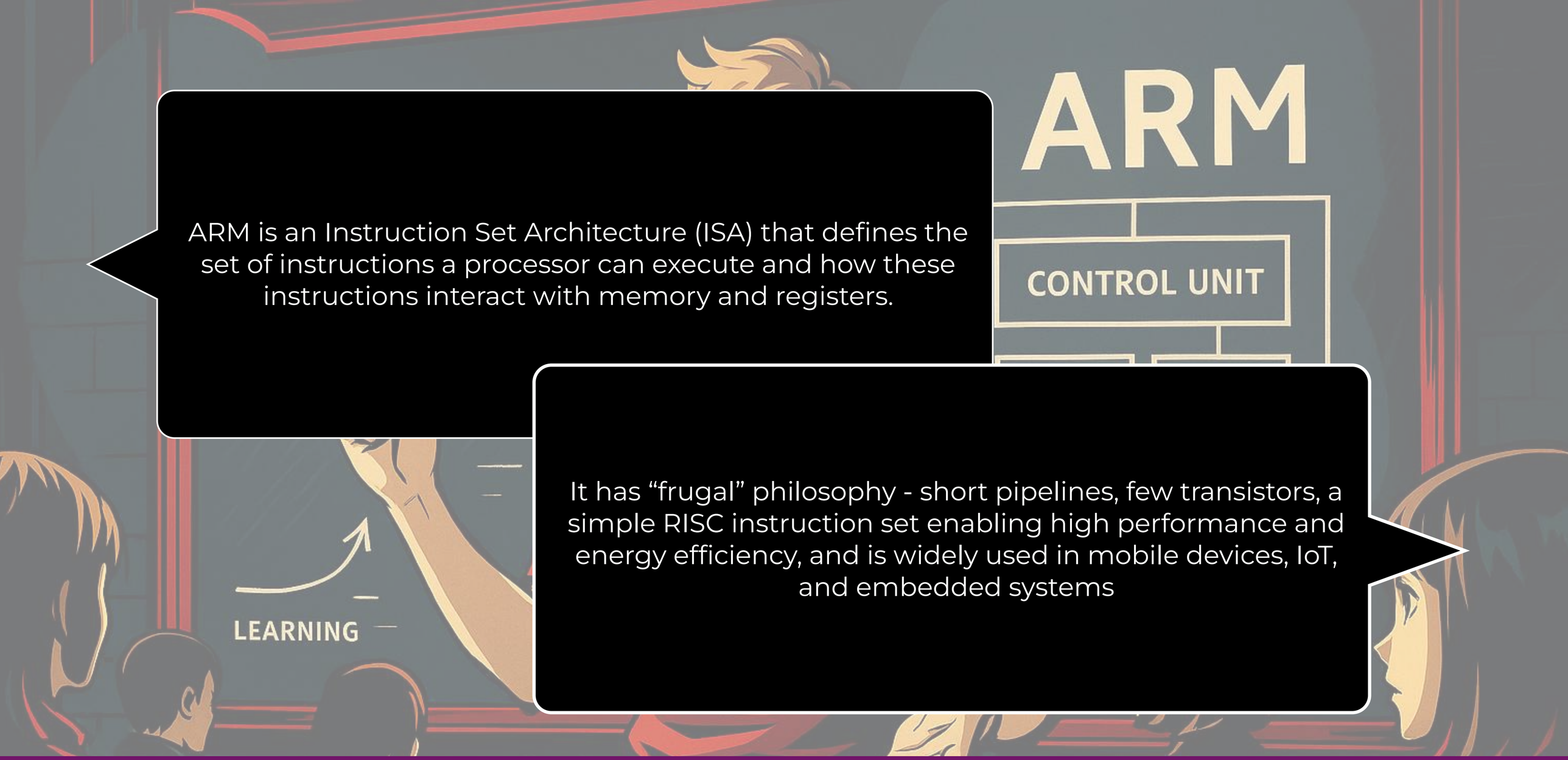
JAVA & ARM

BENCHMARKS, FINOPS, SUSTAINABILITY

140 Slides / 45 Minutes



Arm 101



ARM is an Instruction Set Architecture (ISA) that defines the set of instructions a processor can execute and how these instructions interact with memory and registers.

It has “frugal” philosophy - short pipelines, few transistors, a simple RISC instruction set enabling high performance and energy efficiency, and is widely used in mobile devices, IoT, and embedded systems

Arm

A - ADVANCED

R - RISC

M - MACHINE

A - ACORN

R - RISC

M - MACHINE

We will return to it

A - ACORN

R - RISC

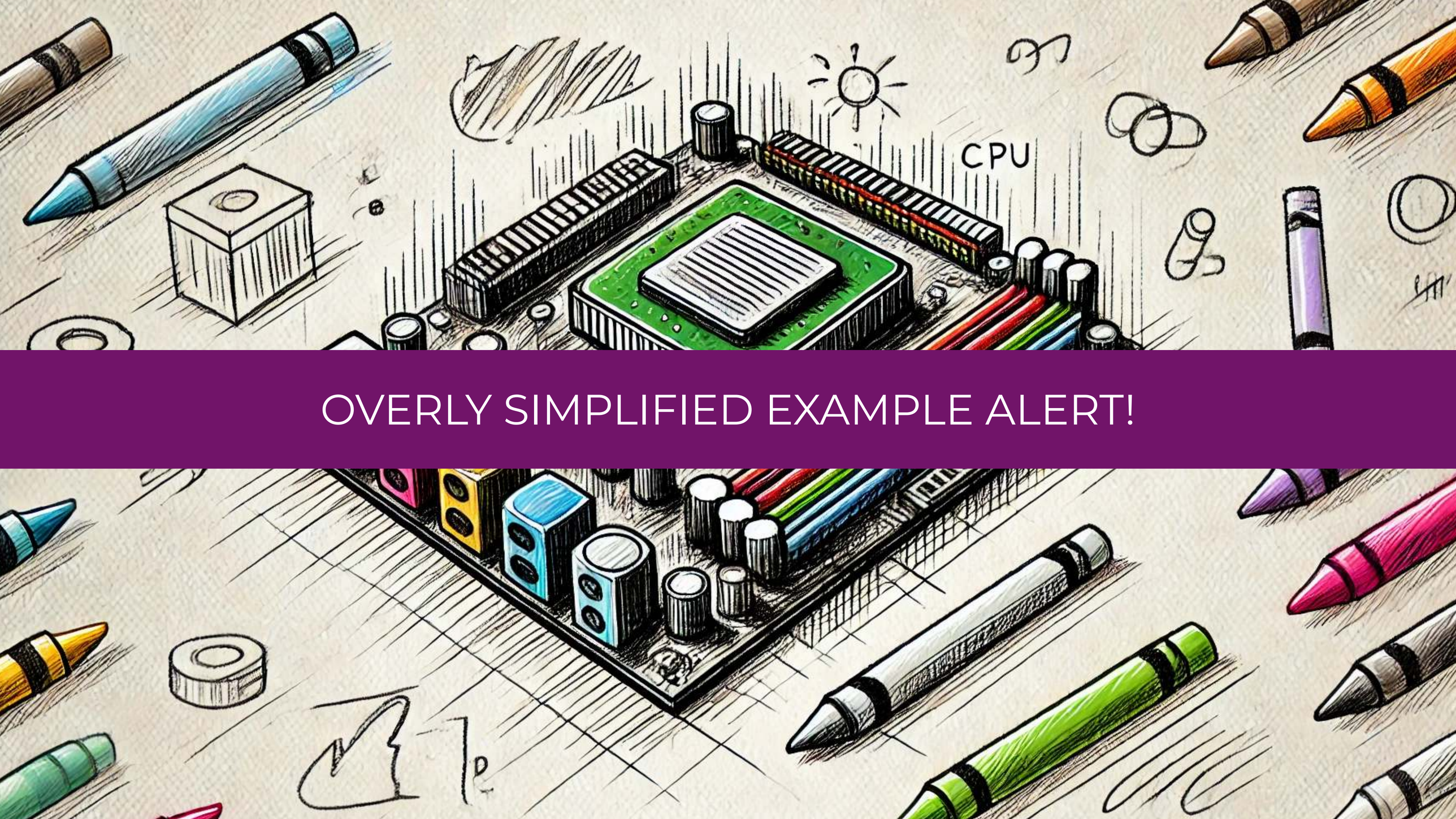
M - MACHINE

A - ADVANCED

R - RISC

M - MACHINE

Let’s talk about ISA



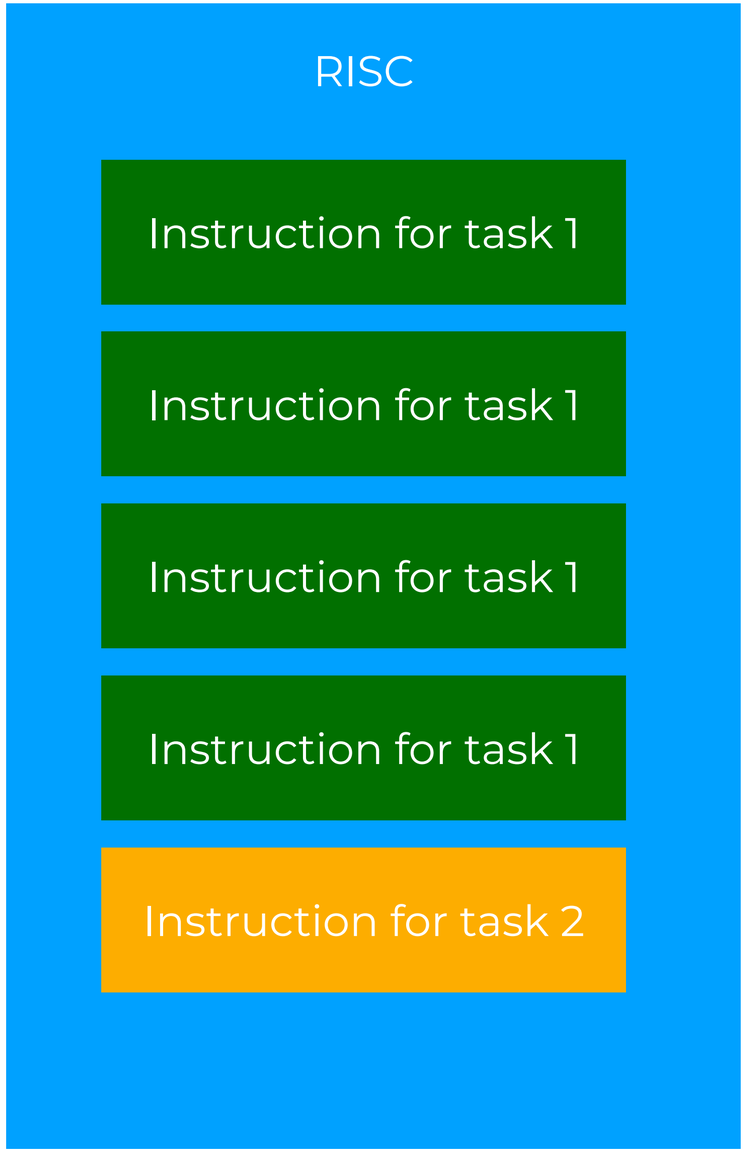
OVERLY SIMPLIFIED EXAMPLE ALERT!

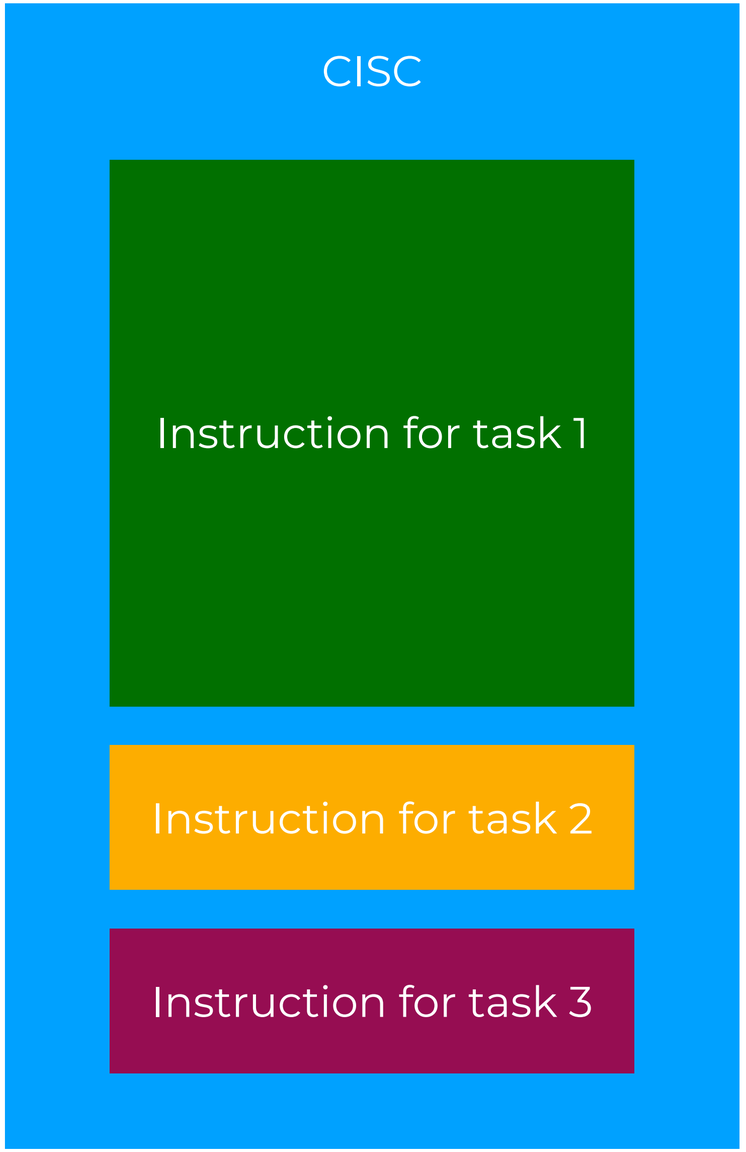
ISA (Instruction Set Architecture) is the interface between a computer's hardware and its software. It defines the set of instructions that a processor can execute, and how they are encoded, decoded, and executed by the processor.

In simple terms, the ISA specifies the machine language the processor understands, as well as how the processor interacts with memory, handles data, and performs operations.

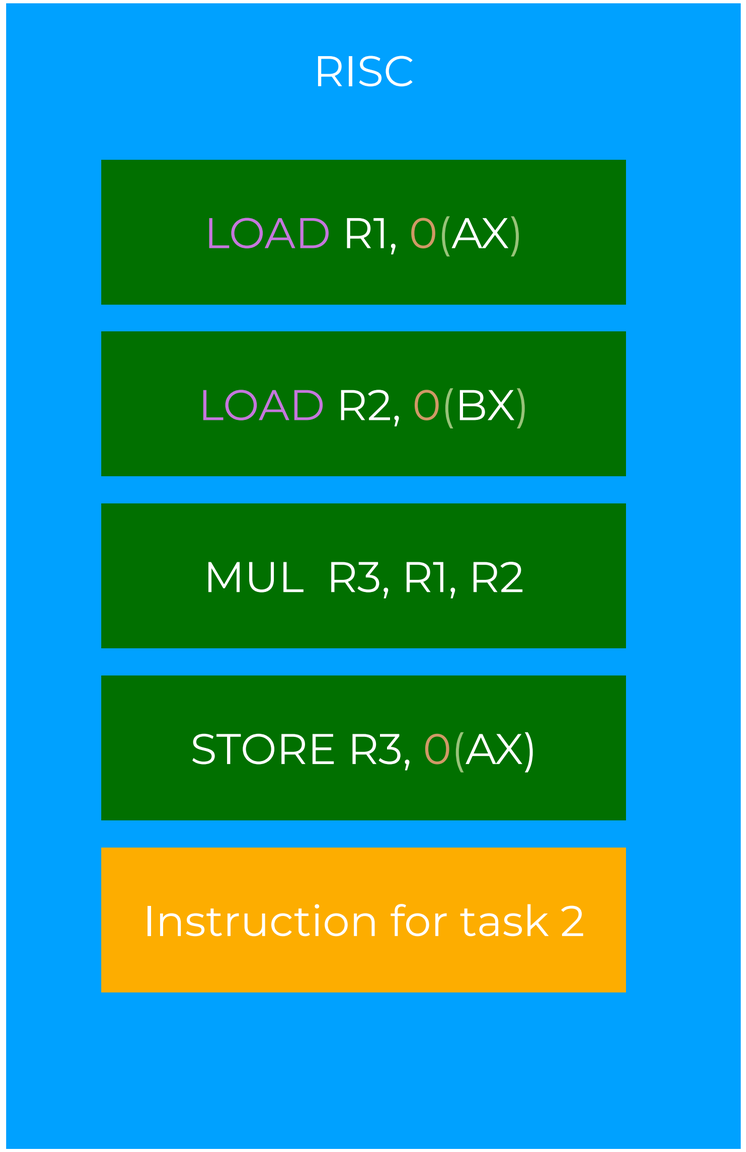
Arm

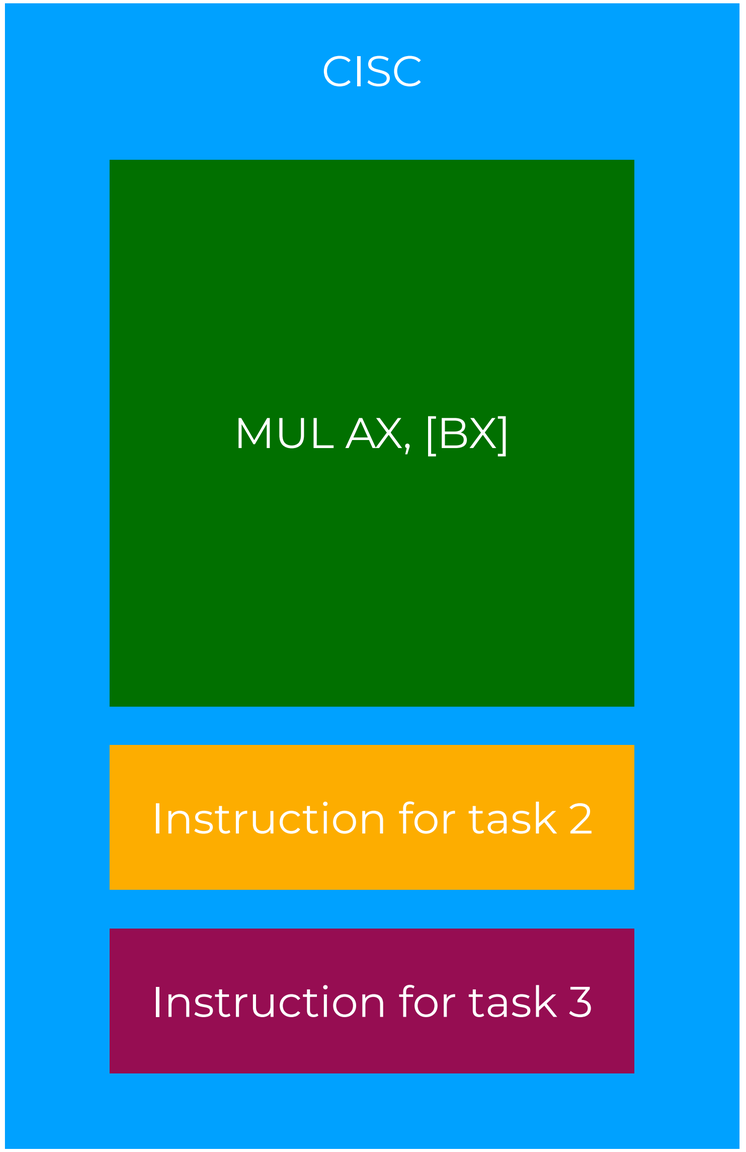
x86 ISA (8086 Instruction Set) is CISC





Reduced Instruction Set Computing vs Complex Instruction Set Computing


RISC
LOAD R1, 0(AX)
LOAD R2, 0(BX)
MUL R3, R1, R2
STORE R3, 0(AX)
Instruction for task 2


CISC
MUL AX, [BX]
Instruction for task 2
Instruction for task 3

RISC vs CISC

Bit like Unix Philosophy for Hardware

RISC

LOAD R1, 0(AX)

LOAD R2, 0(BX)

MUL R3, R1, R2

STORE R3, 0(AX)

Instruction for task 2

MUL AX, [BX]

Instruction for task 2

Instruction for task 3

RISC vs CISC

	x86	ARM
Instruction Length	Variable (1 to 15 bytes) require more complex logic to find where each instruction starts and ends.	Fixed (4 bytes)
Basic Instructions	~200-300 instructions	~100-150 instructions
Extended Instructions	>1,000 instructions (with MMX, SSE, AVX, etc.)	100-200 instructions (with NEON, SIMD, etc.)

RISC vs CISC

1976

There is a lot of legacy instructions

Intel terminates x86S initiative — unilateral quest to de-bloat x86 instruction set comes to an end

News By Paul Alcorn published December 19, 2024

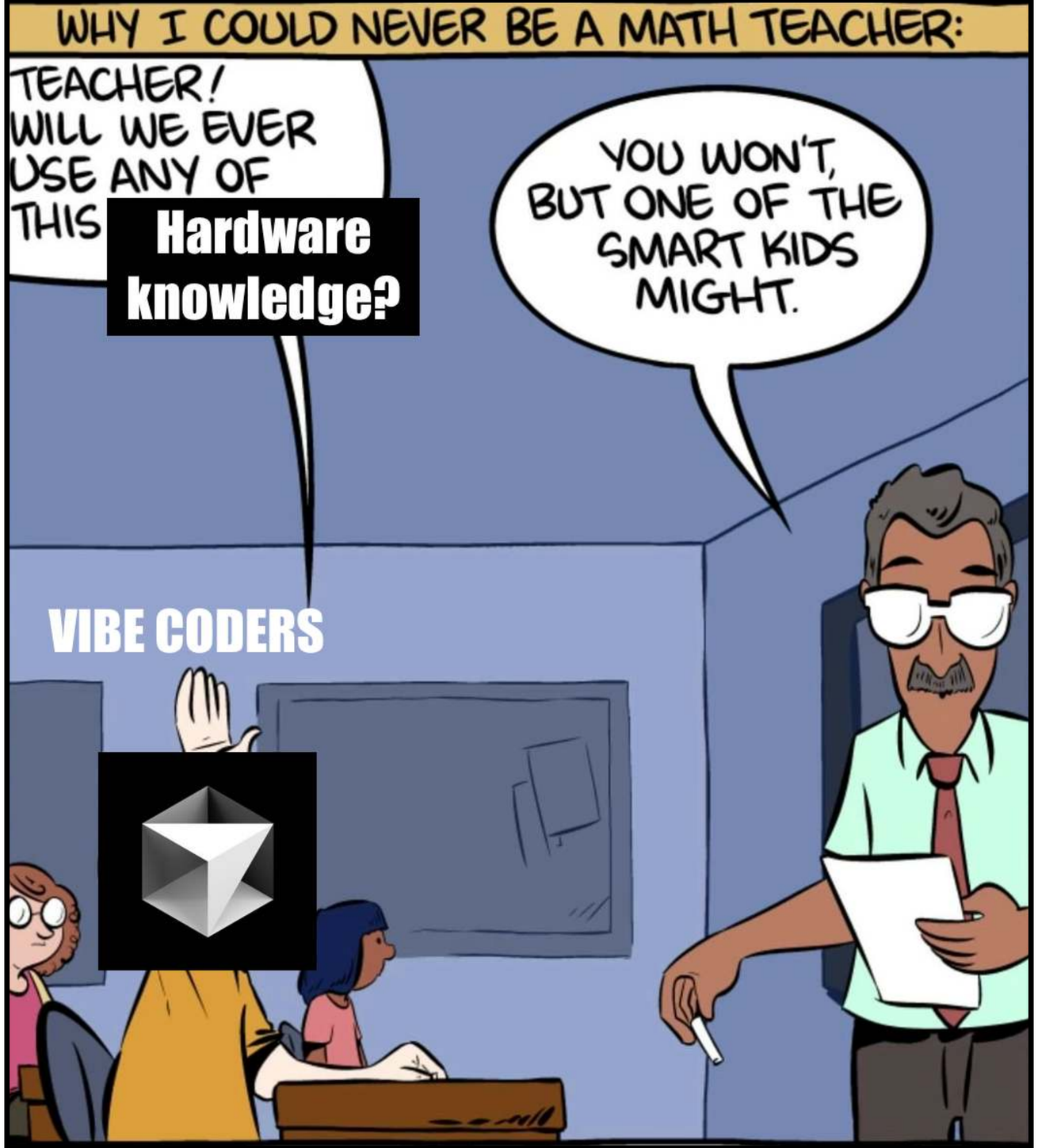
A pivot here, a pivot there.

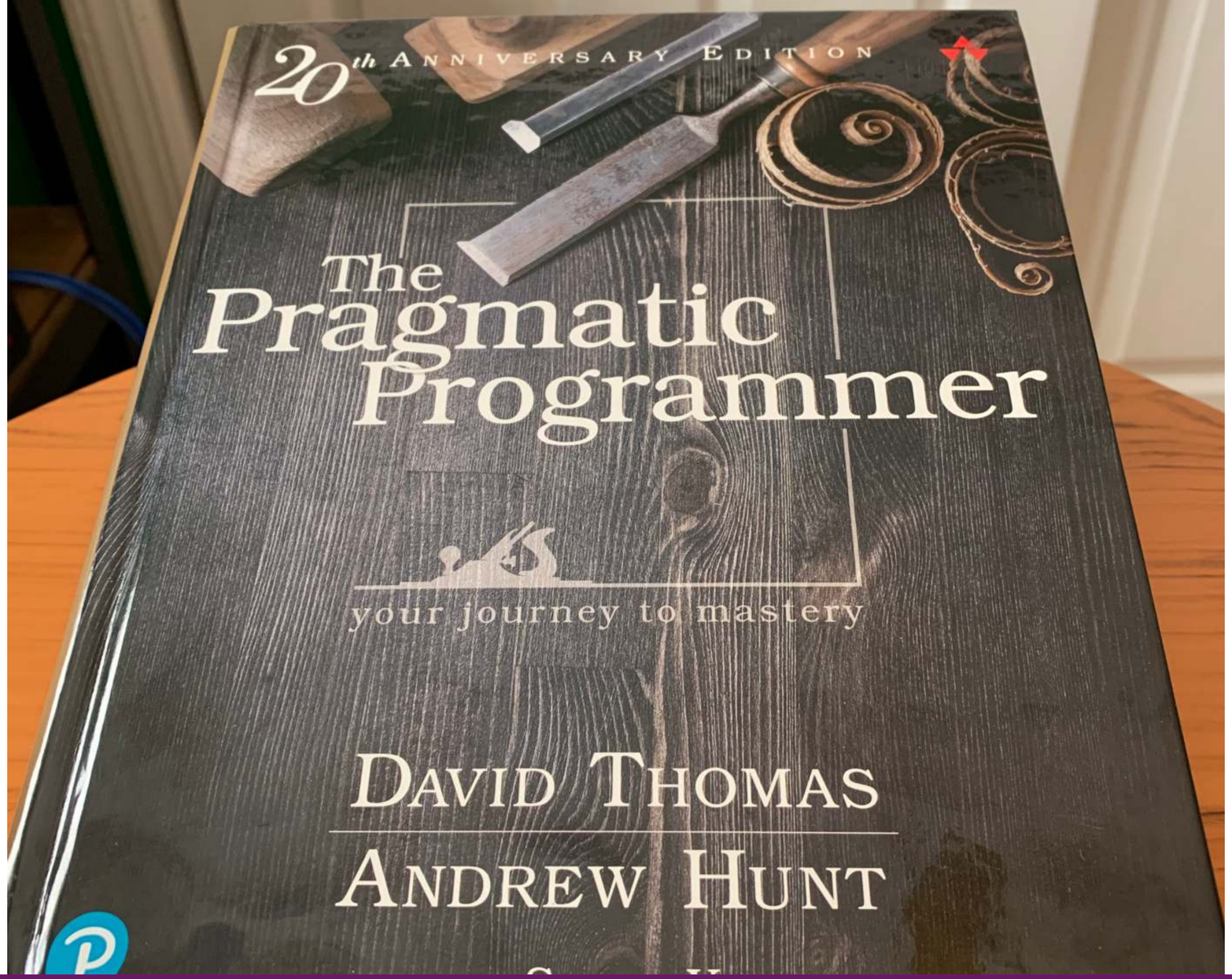
Comments (45)

When you purchase through links on our site, we may earn an affiliate commission. Here's how it works.

(Image credit: Tom's Hardware)

RISC vs CISC


WHY I COULD NEVER BE A MATH TEACHER:
TEACHER! WILL WE EVER USE ANY OF THIS
Hardware knowledge?
YOU WON'T, BUT ONE OF THE SMART KIDS MIGHT.
VIBE CODERS



Let’s bring a bit more of pragmatism

Why it is the perfect time to get interested in 2025?

Apple Newton (1992)

The Apple Newton, the failure that led to the iPhone

Mistakes Were Made

Apple Newton (1992)

Apple found a small British computer company, Acorn, which had improbably created a new CPU design that offered decent speeds at impossibly low power requirements. Apple invested $3 million into the company and helped design a new revision of its chip, the Acorn RISC Machine.

Apple Newton (1992)

I promised 😀

A - ACORN

R - RISC

M - MACHINE

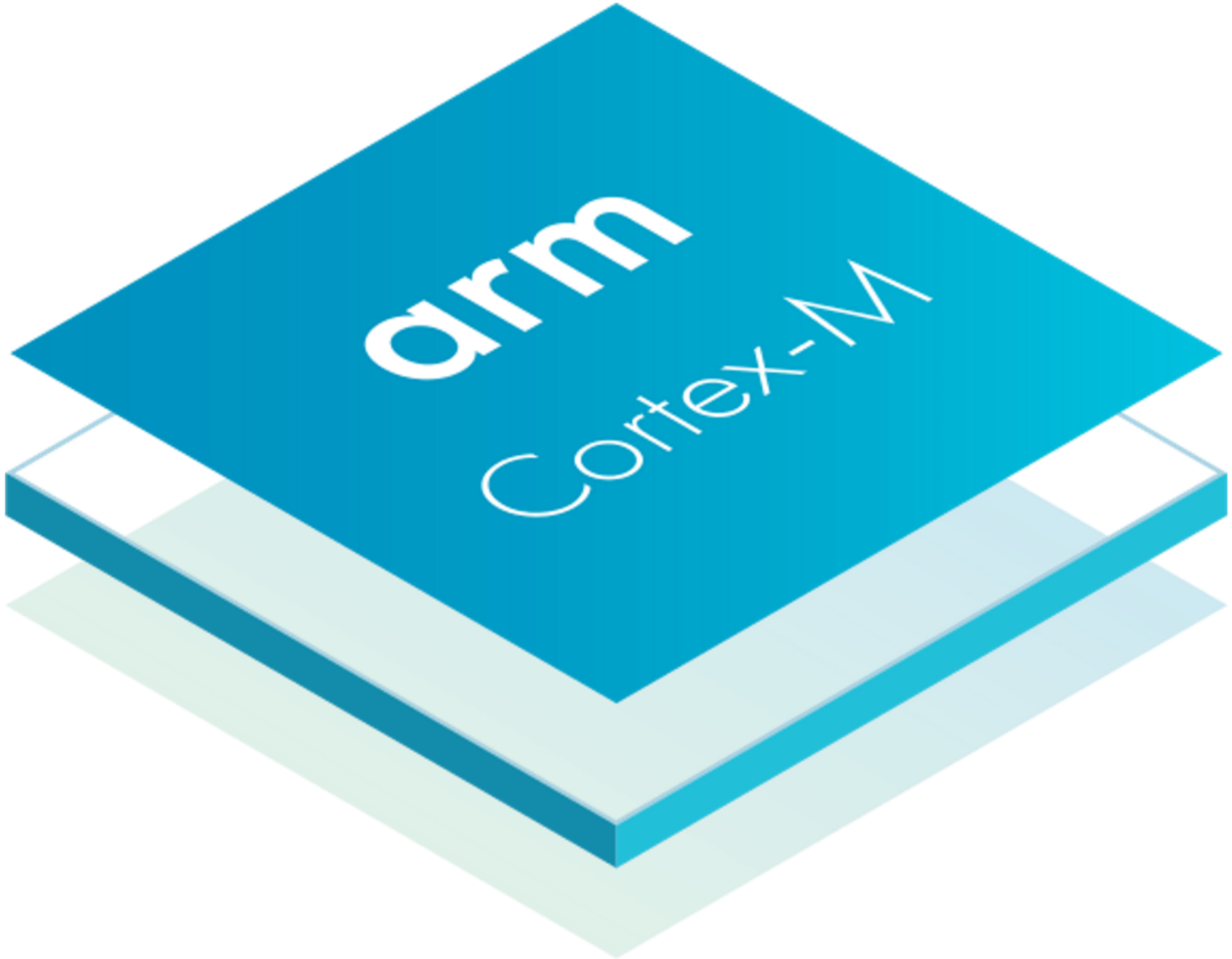
Apple found a small British computer company, Acorn, which had improbably created a new CPU design that offered decent speeds at impossibly low power requirements. Apple invested $3 million into the company and helped design a new revision of its chip, the Acorn RISC Machine.

Apple Newton (1992)

Intel suspends planned new plants in Poland and Germany for two years

SEP 17, 2024 | BUSINESS

ARM do not have factories, they license IP



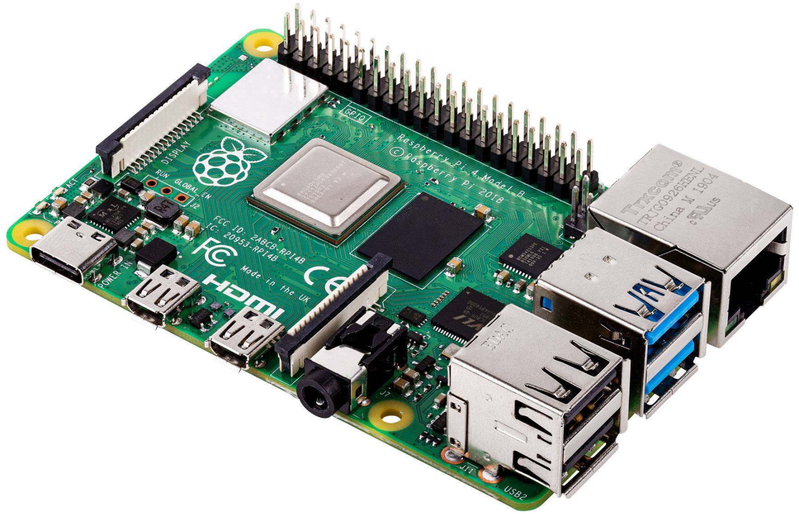
ISA is like a blueprint for CPUs, Cortex is a processor designs.

We know from Nintendo's spec page that Nvidia will be providing the custom chip. The best rumors indicate it will use an Ampere (RTX 30-series) GPU, potentially with 1536 CUDA cores enabled — six times the number in the Switch, and an architecture that's three generations newer. (Nvidia has since stated that the GPU is about 10X more powerful than in the original Switch.) It is possible it could also include some form of DLSS upscaling to help with performance. The CPU will supposedly be an 8-core Arm Cortex-A78, though it might not have all eight cores enabled.

ISA is like a blueprint for CPUs, Cortex is a processor designs.

A - Application
R - Real-time
M - Microcontroller
X - Special

Use also X cores



Cortex A

Firestorm/Icestorm Architecture

MEH.

Neoverse - family of high-performance CPU core designs

arm Products Markets Partners Developers Support & Training Company

CPU

Neoverse N1

Accelerating the Cloud-to-Edge Infrastructure Transformation

The Neoverse N1 CPU is optimized for handling a wide range of cloud-native workloads at world-class levels of performance, efficiency and compute density. It adds up to a more flexible, more scalable and more competitive infrastructure for cloud providers, carriers, developers and customers.

Read Solution Brief

Features and Benefits

Revolutionary Compute Performance

Delivering market-leading server performance at half the power, Neoverse N1 marks a revolution in compute efficiency and performance/watt for the cloud and the edge. Customers have experienced

Infrastructure-Specific Features

Supporting the Armv8.2-A specification, the Neoverse N1 platform delivers a host of architectural and implementation features targeted at the infrastructure market, including include server-class

Extreme Scalability and Diversity

Offering at least 30% compute efficiency gain over previous generation Arm infrastructure CPUs, the N1 platform scales from sub-35W systems for networking and specific tasks to hyperscale workloads

Neoverse - family of high-performance CPU core designs

N (scale) - Computing
V (perf) - HPC, ML, AI
E (eff) - Edge

Kinds of Neoverse Chip Types

We will return to it

Performance-per-Watt

To that too ;)

NEON - SIMD Support

Enhanced Memory Bandwidth

```
add z0.d, z1.d, z2.d
```

Vector addition of 64-bit values.

SVE / SVE2 – Scalable Vector Extension

CAS, LDADD, SWP

Efficient atomic instructions (for lock-free algorithms)

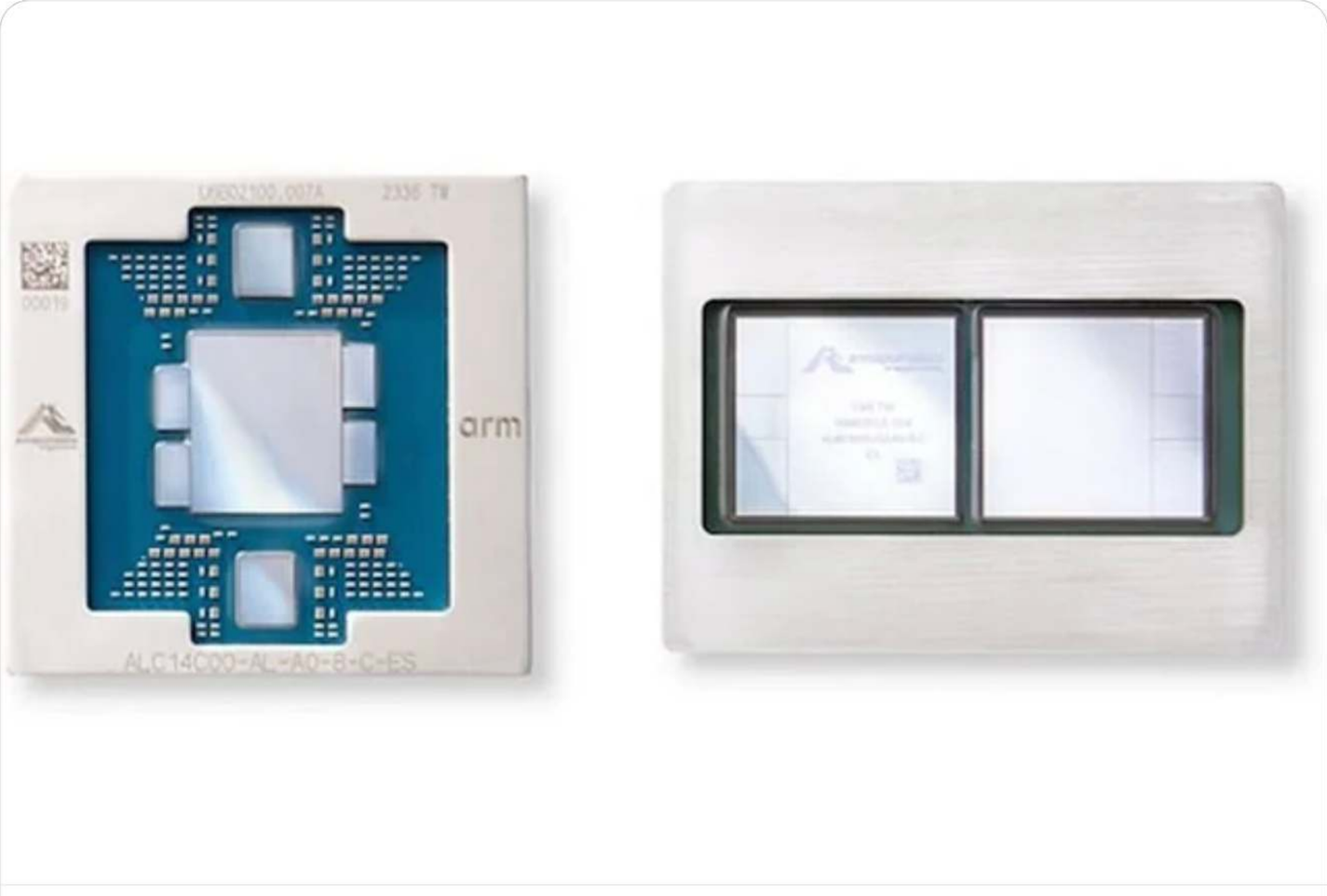
LSE – Large System Extensions

CAS, LDADD, SWP

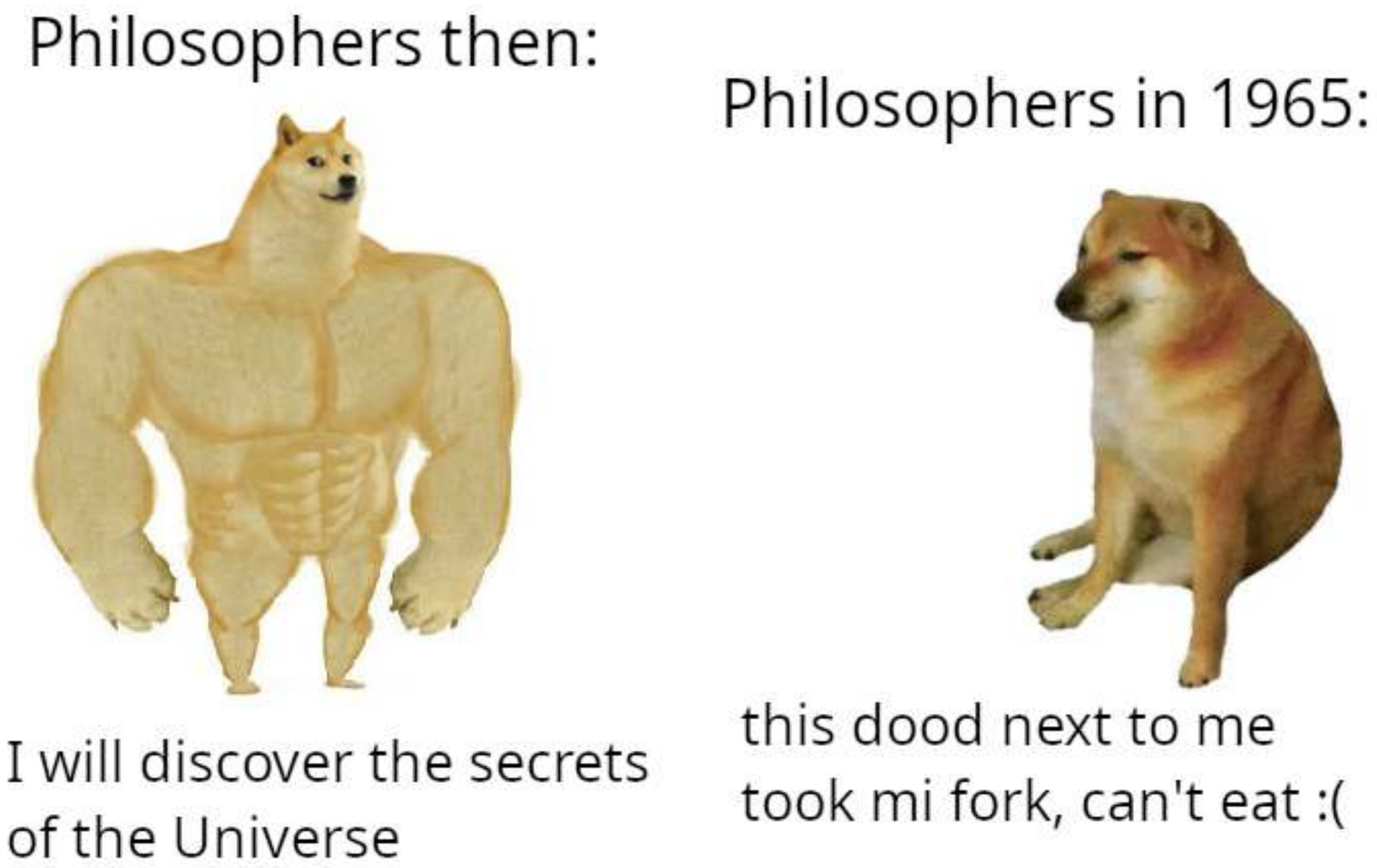
Efficient atomic instructions (for lock-free algorithms)

LSE – Large System Extensions

Amazon Unveils Graviton4: A 96-Core ARM CPU with 536.7 GBps Memory Bandwidth



LSE – Large System Extensions



LSE – Large System Extensions

Cloud
ARM
ARM
ARM
ARM
ARM

Has x86 lost the data center battle? Arm claims victory as it declares close to 50 percent of compute shipped to top hyperscalers in 2025 will be Arm-based

News By Wayne Williams published April 5, 2025

AI demand is behind the Neoverse architecture's meteoric rise

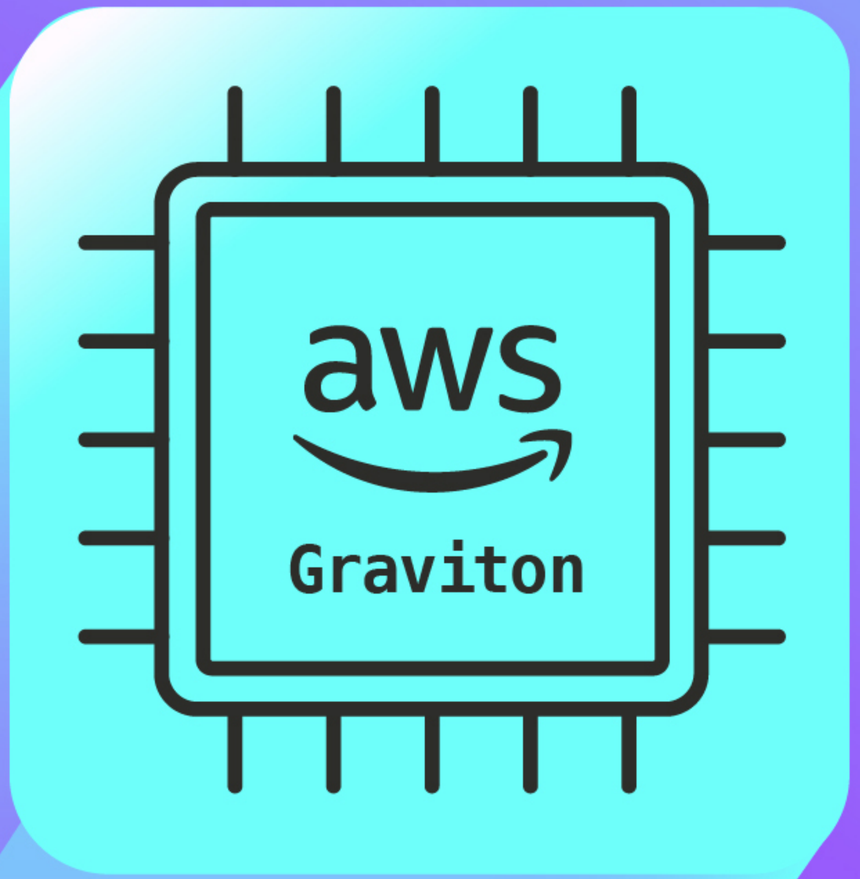
Comments (1)

When you purchase through links on our site, we may earn an affiliate commission. Here's how it works.

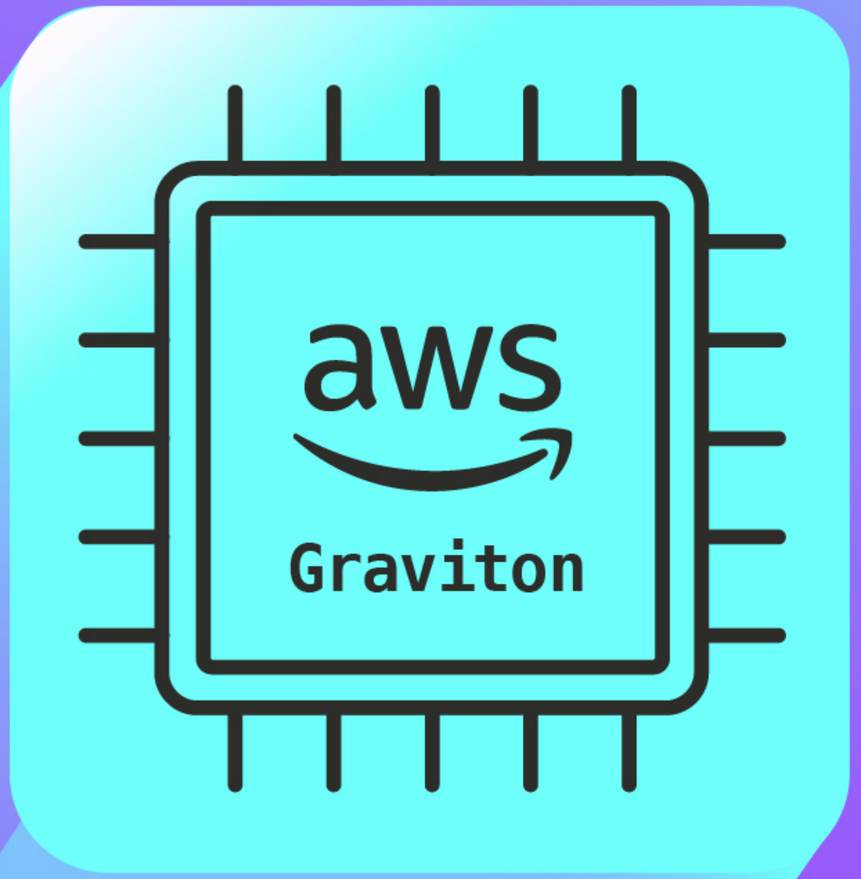
(Image credit: Shutterstock / gguy)

- Arm says Neoverse powers nearly half of 2025 hyperscaler compute shipments
- AI growth makes power-efficiency essential for hyperscale data centers
- AWS, Google, Microsoft are leading shift from x86 to Arm infrastructure

ARM AFTER INVENTING
SUSTAINABILITY
imgflip.com



AWS Graviton (2018)



2018 - AWS Graviton (Cortex-based)

2019 - AWS Graviton 2 (Neoverse-based)

2021 - AWS Graviton 3

2023 - AWS Graviton 4

AWS Graviton (2018)

Ampere Altra (2020)

Provider	Instance family	Ampere Altra processor type
Google Cloud Platform	Tau T2A	Ampere® Altra® (64-core, 3.0 GHz)
Microsoft Azure	Dpsv5 (general-purpose), Epsv5 (memory-optimized)	Ampere® Altra® (up to 64 vCPU, 3.0 GHz)
Oracle Cloud Infrastructure	Ampere A1 Flexible VM, Ampere A1 Bare Metal	Ampere® Altra® (1–80 cores, 3.0 GHz)
Alibaba Cloud	g8y (general-purpose), c8y/c6r (compute-optimized), r8y (memory-optimized), g6r (general-purpose), c6r (compute-optimized)	Ampere® Altra® (2.8 GHz)
Tencent Cloud	SR1	Ampere® Altra® (up to 70 cores, 2.8 GHz)
Equinix Metal	c3.large.arm (bare-metal)	Ampere® Altra® (80-core, 3.0 GHz)
Scaleway	COP-ARM (cost-optimized), AMP2 (Altra Max)	Ampere® Altra® / Altra Max™ (128 cores, 3.0 GHz)

Everybody else uses Ampere Altra

UPDATED 20:30 EDT / MARCH 19 2025

INFRA

SoftBank agrees to buy Arm chipmaker Ampere Computing for $6.5B

BY MIKE WHEATLEY

BLOG | APRIL 9, 2025

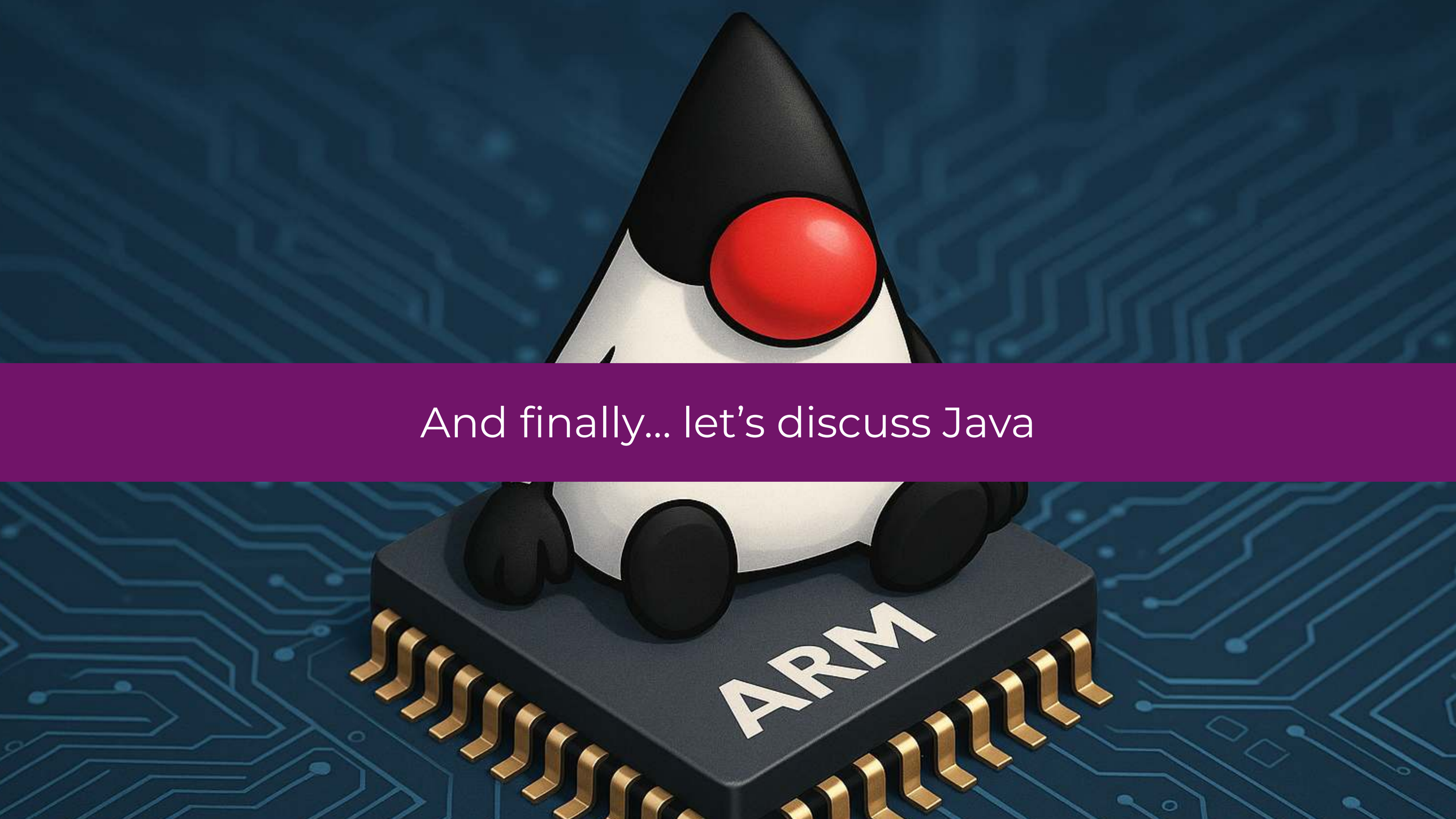
Unpacking Axion: Google Cloud's Custom Arm-based Processor Built for the AI age

How innovations in silicon empower organizations and developers to deploy scalable, high-performance cloud-native and AI applications

By Bhumik Patel, Director, Server Ecosystem Development, Arm

Cloud Computing

Google Axion



And finally… let’s discuss Java

We will return to it

OpenJDK zero on sparc linux

Christian Thalinger twisti at complang.tuwien.ac.at
Mon Apr 7 06:35:39 PDT 2008

- Previous message: OpenJDK zero on sparc linux
- Next message: OpenJDK zero on sparc linux
- **Messages sorted by:** [date] [thread] [subject] [author]

```
On Mon, 2008-04-07 at 15:21 +0200, seb at frankengul.org wrote:
> Unfortunately, I'm unable to trick the build into using gcc-4.1
> instead of gcc.
> Even passing gcc-4.1 as CC works with the configure script but is
> ignored when building
> the interpreter.

When cross-compiling I use:

$ make ARCH=arm ARCH_DATA_MODEL=32 ALT_COMPILER_PATH=/path/to/compiler

This should let you pick another GCC version.

- twisti
```

- Previous message: OpenJDK zero on sparc linux
- Next message: OpenJDK zero on sparc linux
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the distro-pkg-dev mailing list

OpenJDK Zero (2008)

JEP 237: Linux/AArch64 Port

Owner	Andrew Haley
Type	Feature
Scope	Implementation
Status	Closed / Delivered
Release	9
Component	hotspot / compiler
Discussion	aarch64 dash port dash dev at openjdk dot java dot net
Effort	L
Duration	S
Reviewed by	Mikael Vidstedt, Vladimir Kozlov
Endorsed by	Mikael Vidstedt
Created	2014/06/02 17:15
Updated	2017/03/08 23:38
Issue	8044552

Summary

Port JDK 9 to Linux/AArch64.

Description

We (the AArch64 Porting Project) have ported the JDK to a new platform: Linux/AArch64. We have implemented the template interpreter, the C1 (client) and the C2 (server) JIT compilers.

The focus of this JEP is not the porting effort itself, which has been mostly completed, but rather the integration of the ports into the OpenJDK master repositories.

Currently we have a large number of trivial changesets in the shared part of the HotSpot repository. These are mostly #ifdef's that include the relevant platform-specific files. There are a few changes of other kinds, but again these are guarded by #ifdef AARCH64. Thus, the risk to other platforms is low.

There are also changes to the build machinery of HotSpot and rest of the JDK to add appropriate definitions for endianness, word size, and so on. Again, these should not affect other platforms.

Most of the changes to be integrated won't affect the current OpenJDK platforms in any way because they are only active on the new platform.

There are also changes in the build system but they shouldn't cause much trouble.

Here is a patch of the changes needed to the shared code in HotSpot.

All changesets are collected in a staging repository:
http://hg.openjdk.java.net/aarch64-port/stage/

JEP 237: Linux/AArch64 Port (2014)

made with mematic

How Red Hat ported OpenJDK to 64-bit Arm: A community history

February 1, 2021 | Andrew Haley

Related topics: Java, Open Source Communities, Linux

Share:

Table of contents:

It has been quite a year for Arm Ltd., the firm that designs reduced instruction set computing (RISC) architectures for computer processors. The news that Arm-based computers will be important for the foreseeable future has even reached the mainstream media. At the end of 2019, Amazon Web Services announced Arm-based Graviton2 servers. In June 2020, Apple announced its plans to move Macintosh computers over to Apple silicon—which means Arm.

For those of us who have worked for years on Arm servers, this shift has been a long time coming.

In the beginning

Wind back to a day in 2011. I was having lunch at The Wrestlers (the Cambridge U.K. tech community's favorite Thai eatery) with Jon Masters, Red Hat's lead Arm architect. He had exciting news to share: Arm would produce a 64-bit architecture, and Red Hat was going to port Red Hat Enterprise Linux (RHEL) to it. We'd need to solve quite a few problems to get this done, but one was particularly difficult. The software then available for the new 64-bit architecture did not include Java. Java is a key component of much enterprise software, so this news was a pretty big deal.

Someone would have to write a port. I had heard that two experts could write a bare-bones port of OpenJDK in about a year, but that was all I knew. If my team were going to do it, we'd have to learn on the job.

Recent Posts

New C++ features in GCC 15 →

Evaluating memory overcommitment in OpenShift Virtualization →

Cracking the code: How neural networks might actually "think" →

How to use content templates in Red Hat Insights →

Fine-tune LLMs with Kubeflow Trainer on OpenShift AI →

JEP 237: Linux/AArch64 Port (2014)

JEP 297: Unified arm32/arm64 Port

Author	Edward Nevill
Owner	Ed Nevill
Type	Feature
Scope	Implementation
Status	Closed / Delivered
Release	9
Component	hotspot / compiler
Discussion	aarch32 dash port dash dev at openjdk dot java dot net
Effort	M
Duration	S
Reviewed by	Bob Vandette, Mikael Vidstedt, Vladimir Kozlov
Endorsed by	Mikael Vidstedt, Vladimir Kozlov
Created	2016/10/23 21:57
Updated	2025/01/21 06:09
Issue	8168503

Description

The contribution from Oracle provides C1 and C2 support for both arm32 and arm64. The code has been merged into a JDK 9 tree in a separate repository in the aarch32 project area.

Oracle's intention to open source the ARM port was announced on the aarch32 mailing list on Aug 23, 2016 and there have been several discussion threads on the aarch32 mailing list. Please follow the links below to see these discussions.

- Open sourcing JDK 9 ARM ports
- Open sourcing ARM ports status
- Webrev of Oracle ARM & AARCH64 Sources
- What next for aarch32

JEP 297: Unified arm32/arm64 Port (2016)

JEP 388: Windows/AArch64 Port

Authors	Monica Beckwith, Ludovic Henry, Bernhard Urban-Forster
Owner	Vladimir Kozlov
Type	Feature
Scope	Implementation
Status	Closed / Delivered
Release	16
Component	hotspot / runtime
Discussion	aarch64 dash port dash dev at openjdk dot java dot net
Effort	M
Duration	S
Blocks	JEP 391: macOS/AArch64 Port
Reviewed by	Vladimir Kozlov
Endorsed by	Mark Reinhold, Vladimir Kozlov
Created	2020/06/29 19:13
Updated	2024/07/26 14:26
Issue	8248496

JEP 391: macOS/AArch64 Port

Authors	Anton Kozlov, Vladimir Kempik
Owner	Vladimir Kempik
Type	Feature
Scope	JDK
Status	Closed / Delivered
Release	17
Component	hotspot / runtime
Discussion	aarch64 dash port dash dev at openjdk dot java dot net
Effort	M
Duration	M
Depends	JEP 388: Windows/AArch64 Port
Reviewed by	Andrew Haley, Vladimir Kozlov
Endorsed by	Vladimir Kozlov
Created	2020/08/07 07:08
Updated	2024/07/04 16:43
Issue	8251280

Ports for MacOS/Windows AArch64

JEP 388: Windows/AArch64 Port

Authors	Monica Beckwith, Ludovic Henry, Bernhard Urban-Forster
Owner	Vladimir Kozlov
Type	Feature
Scope	Implementation
Status	Closed / Delivered
Release	16
Component	hotspot / runtime
Discussion	aarch64 dash port dash dev at openjdk dot java dot net
Effort	M
Duration	S
Blocks	JEP 391: macOS/AArch64 Port
Reviewed by	Vladimir Kozlov
Endorsed by	Mark Reinhold, Vladimir Kozlov
Created	2020/06/29 19:13
Updated	2024/07/26 14:26
Issue	8248496

JEP 391: macOS/AArch64 Port

Authors	Anton Kozlov, Vladimir Kempik
Owner	Vladimir Kempik
Type	Feature
Scope	JDK
Status	Closed / Delivered
Release	17
Component	hotspot / runtime
Discussion	aarch64 dash port dash dev at openjdk dot java dot net
Effort	M
Duration	M
Depends	JEP 388: Windows/AArch64 Port
Reviewed by	Andrew Haley, Vladimir Kozlov
Endorsed by	Vladimir Kozlov
Created	2020/08/07 07:08
Updated	2024/07/04 16:43
Issue	8251280

JEP 297: Unified arm32/arm64 Port (2016)

16-ea+10-macOS-aarch64 Pre-release

Compare

lewurm released this 12 Nov 2020 · 31 commits to master since this release 16-ea+10-ma... 2f8fc04

This release is an early access build of our macOS+AArch64 port (also known as Apple Silicon), based on openjdk/jdk@ `68da63d`

Together with Azul Systems we are working on JEP 391 to upstream relevant patches to the OpenJDK project. The WIP branch can be found here: https://github.com/openjdk/aarch64-port/tree/jdk-macos

16-ea+10-macOS-aarch64 Pre-release

Compare

lewurm released this 12 Nov 2020 · 31 commits to master since this release 16-ea+10-ma... 2f8fc04

This release is an early access build of our macOS+AArch64 port (also known as Apple Silicon), based on openjdk/jdk@ `68da63d`

Together with Azul Systems we are working on JEP 391 to upstream relevant patches to the OpenJDK project. The WIP branch can be found here: https://github.com/openjdk/aarch64-port/tree/jdk-macos

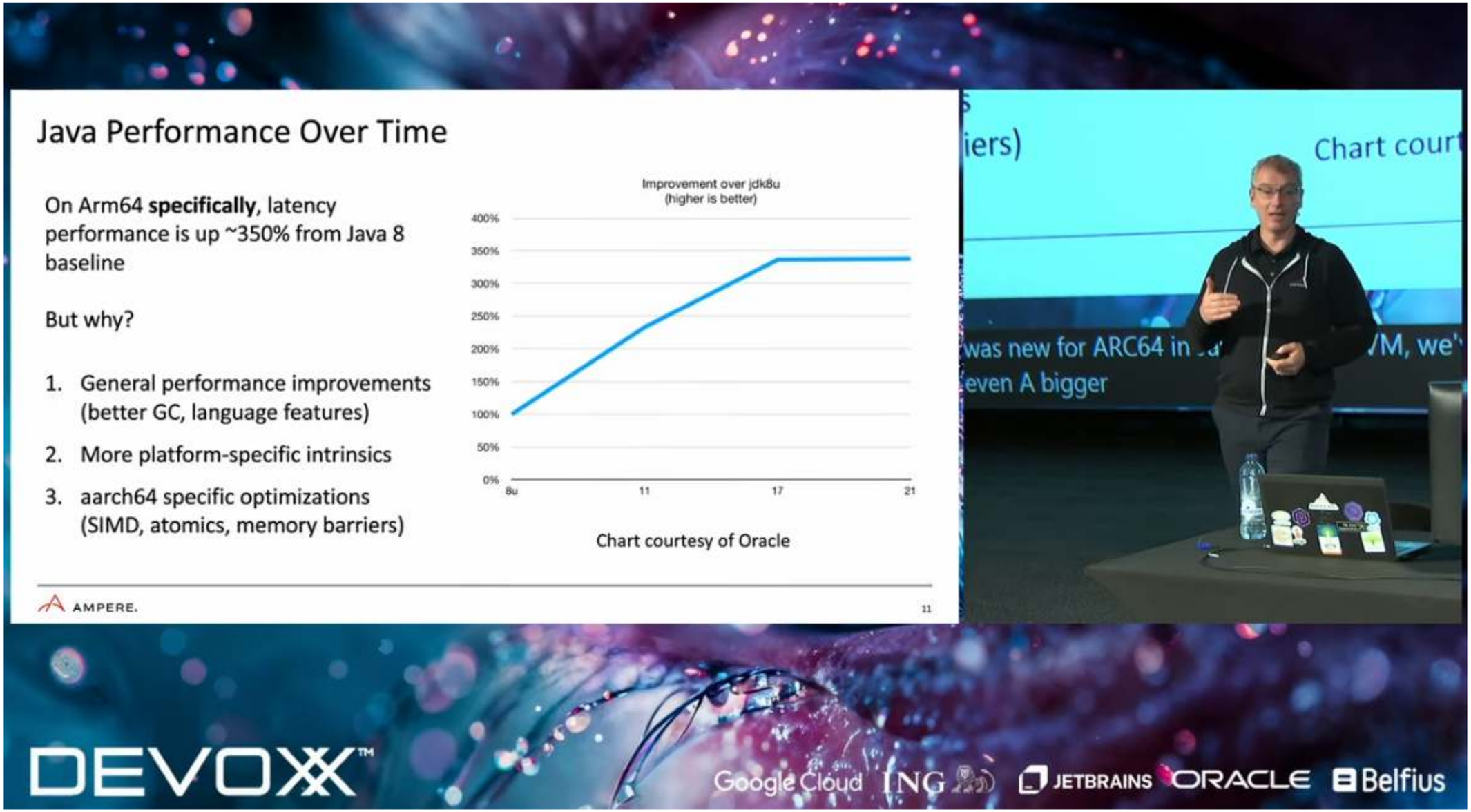
16-ea+10-macOS-aarch64 Pre-release

Compare

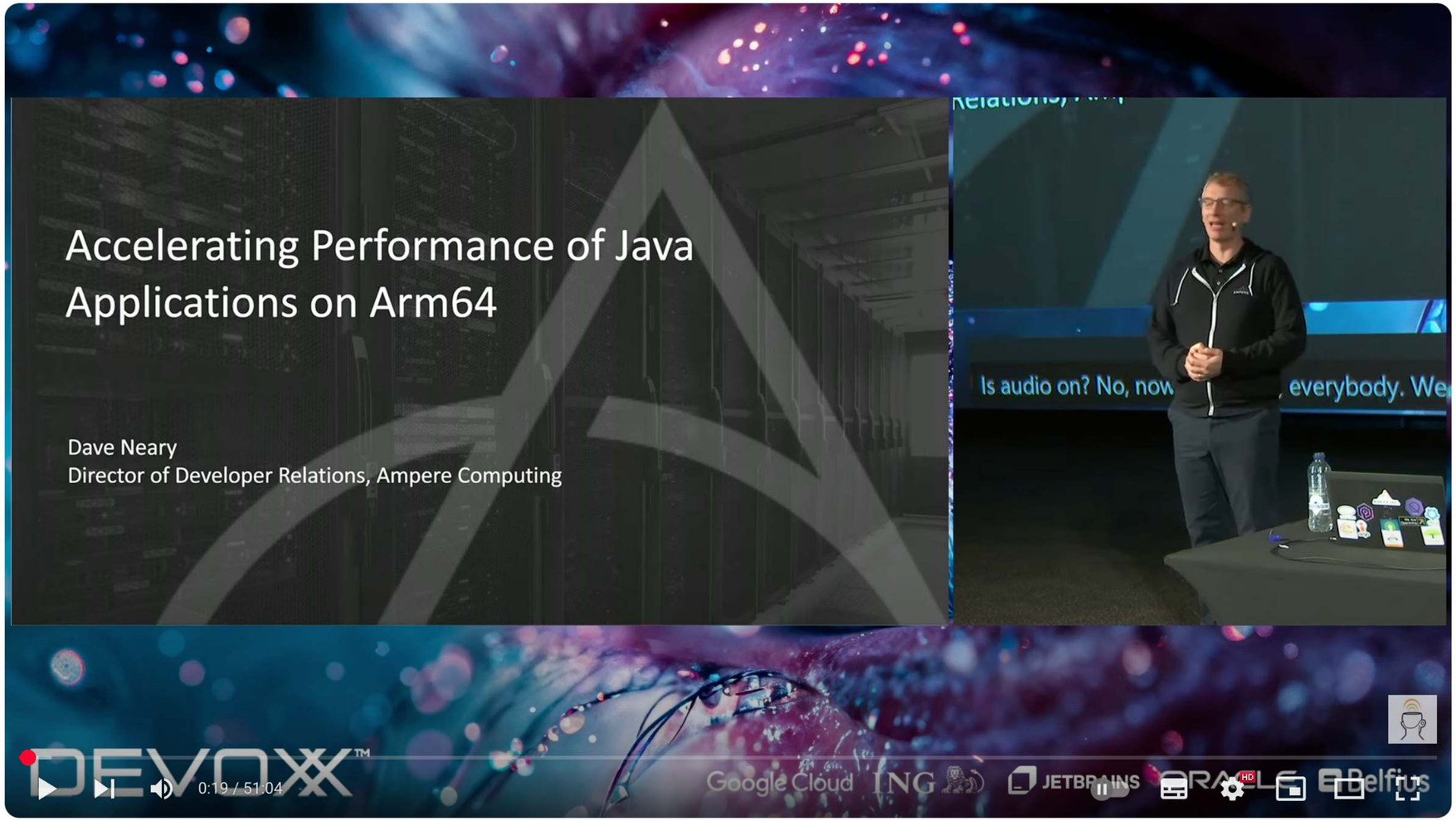
lewurm released this 12 Nov 2020 · 31 commits to master since this release 16-ea+10-ma... 2f8fc04

This release is an early access build of our macOS+AArch64 port (also known as Apple Silicon), based on openjdk/jdk@ `68da63d`

Together with Azul Systems we are working on JEP 391 to upstream relevant patches to the OpenJDK project. The WIP branch can be found here: https://github.com/openjdk/aarch64-port/tree/jdk-macos



Dave Neary from Ampere - JDK 8-25 350%



Accelerating performance of Java applications on Arm64 by Dave Neary

Devoxx
164 tys. subskrybentów
Subskrybuj

33 | Udostępnij | Pobierz | Klip | ···

Accelerating performance of Java applications on Arm64 by Dave Neary

JEP 315: Improve Aarch64 Intrinsics

Owner	Dmitrij Pochepko
Type	Feature
Scope	Implementation
Status	Closed / Delivered
Release	11
Component	hotspot / compiler
Discussion	hotspot dash compiler dash dev at openjdk dot java dot net
Effort	L
Duration	L
Reviewed by	Mikael Vidstedt, Vladimir Kozlov
Endorsed by	Vladimir Kozlov
Created	2017/10/10 12:40
Updated	2023/05/03 08:21
Issue	8189104

Always use min JDK 11

Summary

Improve the existing string and array intrinsics, and implement new intrinsics for the `java.lang.Math` sin, cos and log functions, on AArch64 processors.

Summary

Improve the existing string and array intrinsics, and implement new intrinsics for the `java.lang.Math` sin, cos and log functions, on AArch64 processors.

Non-Goals

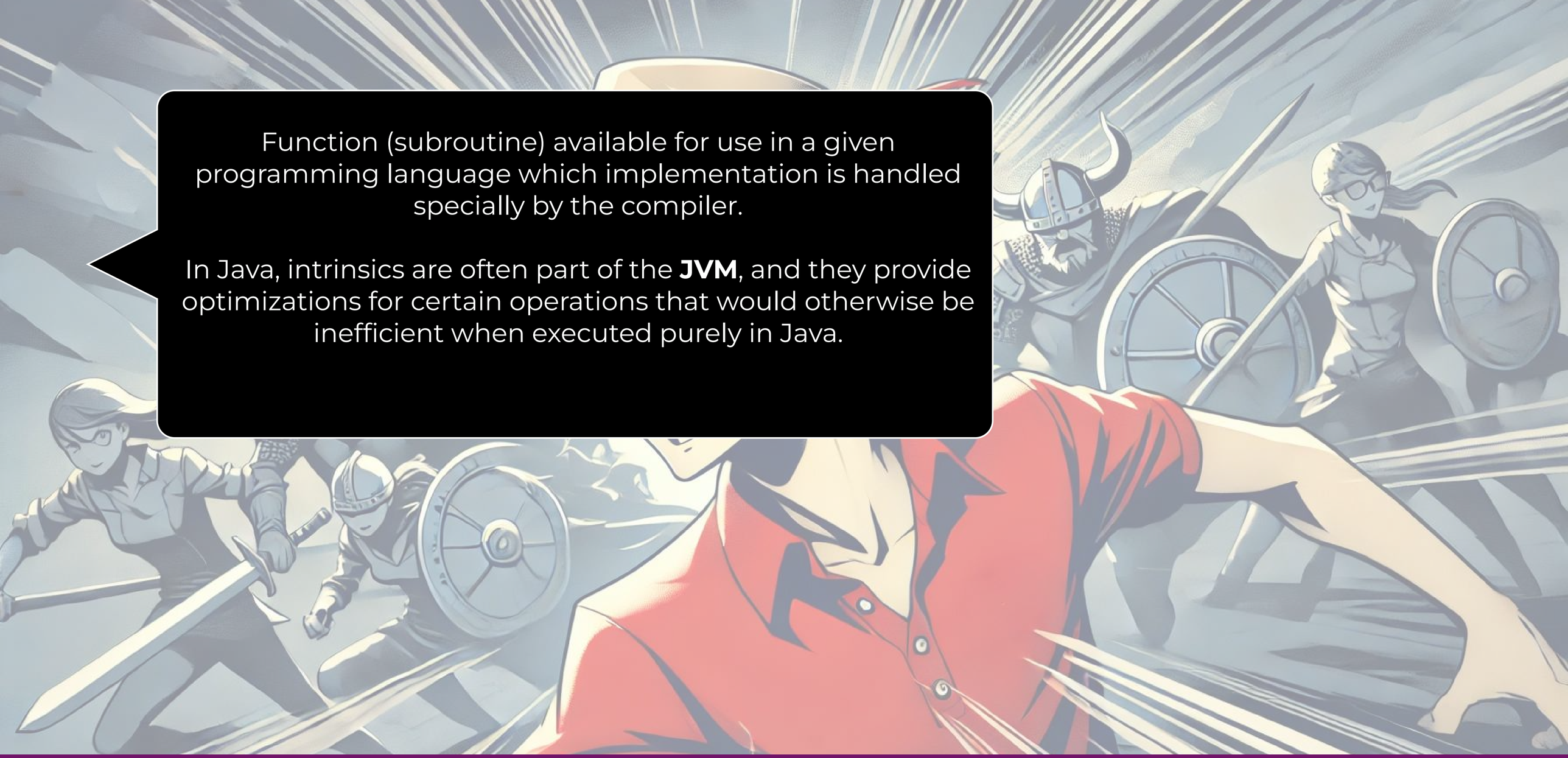
- Compare to and match the performance of other architectures
- Tune generic AArch64 port intrinsics for optimal performance on a single ARM64 architecture implementation only
- Port intrinsics to the ARM CPU port

Motivation

Specialized CPU architecture-specific code patterns improve the performance of user applications and benchmarks.

JEP 315: Improve Aarch64 Intrinsics (2017)

Feature Support



Function (subroutine) available for use in a given programming language which implementation is handled specially by the compiler.

In Java, intrinsics are often part of the **JVM**, and they provide optimizations for certain operations that would otherwise be inefficient when executed purely in Java.

JVM Intrinsics

Zero-Assembler Project

Zero is a port of OpenJDK that uses no assembler and therefore can trivially be built on any system. The goal of this project is be to be able to build a TCK-compliant OpenJDK of reasonable performance on any platform with no additional porting work.

The interpreter part of Zero is known to work on PowerPC (32- and 64-bit), x86-64, IA-64, ARM and zSeries. Zero is currently Linux- and GCC-specific, but supporting other operating systems and compilers is one area in which contributions are particularly welcome.

Work is currently under way on an LLVM-based JIT known as Shark.

This Project is sponsored by the Porters Group.

Community

- Mailing list
  - zero-dev
- Bloggers
  - Gary Benson
- Links
  - Zero and Shark FAQ.

When JDK Zero is useful

- For platforms that lack native support
- During the development of new platforms
- For portability

Have you heard about JDK Zero?

Intrinsic	What it does	Key ARM tricks
hasNegatives (ASCII detector)	Checks if any byte ≥ 0x80; short-circuits UTF-16 paths	ORR v0.16b, v1.16b then UMAXV
StringLatin1.inflate	Expand Latin-1 byte[] → UTF-16 char[]	UZIP1/UZP1 interleave + optional PRFM prefetch
StringUTF16.compress	Collapse char[] → Latin-1 byte[] when all < 256	Pair-wise pack with XTN, bail-out on high bit
String.indexOf	Vector search for needle byte/char	CMEQ mask, CNT+CLS to find first set bit
String.equals / compareTo	16-byte parallel compare, early exit on first diff	LDP loads + CMHI/CMHS + UADDLV reduction

JDK / JDK-8184943

AARCH64: Intrinsify hasNegatives

Resolved

Details

Type:	Enhancement	Resolution:	Fixed
Priority:	P3	Fix Version/s:	10
Affects Version/s:	10		
Component/s:	hotspot		
Labels:	aarch64 intrinsic openjdk		
Subcomponent:	compiler		
Resolved In Build:	b21		
CPU:	aarch64		

Description

There is no hasNegatives intrinsic implemeneted for AArch64. It should be implemented.

String performance improving Intrinsics

JDK / JDK-8252204

AArch64: Implement SHA3 accelerator/intrinsic

Resolved

Export

Details

Type:	Enhancement	Resolution:	Fixed
Priority:	P4	Fix Version/s:	16
Affects Version/s:	16		
Component/s:	hotspot		
Labels:	intrinsic		
Subcomponent:	compiler		
Resolved In Build:	b22		
CPU:	aarch64		
OS:	generic		

People

Assignee:	Fei Yang
Reporter:	Fei Yang
Votes:	0 Vote for this issue
Watchers:	5 Start watching this issue

Dates

Created:	2020-08-23 20:00
Updated:	2024-12-09 14:17
Resolved:	2020-10-21 21:39

Description

The existing implementation of ` sun.security.provider.SHA3::implCompress` is done in Java.
When compiled to native code, the C2 JIT code does not vectorize well and it contains some Java-isms like array-bound checks.
That hurts performance for an algorithm that can be used to validate large amounts of data.

Armv8.2 optionally provides "ARMv8.2-SHA, SHA2-512 and SHA3 functionality".
To speed it up, we could simply create an intrinsic that implements the algorithm using the new SIMD instructions:
- EOR3 Three-way Exclusive OR (page C7-1479)
- RAX1 Rotate and Exclusive OR (page C7-1892)
- XAR Exclusive OR and Rotate (page C7-2303)
- BCAX Bit Clear and Exclusive OR (page C7-1418)

This would also help eliminate the Java-isms.
That is a similar approach to intrinsics for SHA1/SHA256/SHA512.

Reference implementation for core SHA-3 transform using ARMv8.2 Crypto Extensions:
https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/arch/arm64/crypto/sha3-ce-core.S?h=v5.4.52

Hardware-accelerated instructions for ARMv8 Crypto

JDK / JDK-8268232

Aarch64: Use Neon in intrinsics for String.indexOf

Open

Details

Type:	Enhancement	Resolution:	Unresolved
Priority:	P4	Fix Version/s:	tbd
Affects Version/s:	17		
Component/s:	hotspot		
Labels:	c2 huawei-interest intrinsic		
Subcomponent:	compiler		
CPU:	aarch64		
OS:	generic		

Description

We can use neon instructions in C2_MacroAssembler::string_indexof to improve the performance of String.indexOf's intrinsics.

JDK / JDK-8268229

Aarch64: Use Neon in intrinsics for String.equals

Closed

Details

Type:	Enhancement	Resolution:	Won't Fix
Priority:	P4	Fix Version/s:	tbd
Affects Version/s:	17		
Component/s:	hotspot		
Labels:	c2 huawei-interest intrinsic		
Subcomponent:	compiler		
CPU:	aarch64		
OS:	generic		

Description

Now we use 64-bits comparsion in MacroAssembler::string_equals, we can use neon instructions to improve the intrinsic's performance.

```
// Compare Strings

// For Strings we're passed the address of the first characters in a1
// and a2 and the length in cnt1.
// elem_size is the element size in bytes: either 1 or 2.
// There are two implementations. For arrays >= 8 bytes, all
// comparisons (including the final one, which may overlap) are
// performed 8 bytes at a time. For strings < 8 bytes, we compare a
// halfword, then a short, and then a byte.

void MacroAssembler::string_equals(Register a1, Register a2,
                                   Register result, Register cnt1, int elem_size)
```

Hardware-accelerated instructions for ARMv8 Crypto

JDK / JDK-8268232

Aarch64: Use Neon in intrinsics for String.indexOf

Open

Details

Type:	Enhancement	Resolution:	Unresolved
Priority:	P4	Fix Version/s:	tbd
Affects Version/s:	17		
Component/s:	hotspot		
Labels:	c2 huawei-interest intrinsic		
Subcomponent:	compiler		
CPU:	aarch64		
OS:	generic		

Description

We can use neon instructions in C2_MacroAssembler::string_indexof to improve the performance of String.indexOf's intrinsics.

JDK / JDK-8268229

Aarch64: Use Neon in intrinsics for String.equals

Closed

Details

Type:	Enhancement	Resolution:	Won't Fix
Priority:	P4	Fix Version/s:	tbd
Affects Version/s:	17		
Component/s:	hotspot		
Labels:	c2 huawei-interest intrinsic		
Subcomponent:	compiler		

Fei Yang added a comment - 2022-03-31 19:07

Close this issue since proposed solution won't benifit common cases.

Description

Now we use 64-bits comparsion in MacroAssembler::string_equals, we can use neon instructions to improve the intrinsic's performance.

```
// Compare Strings

// For Strings we're passed the address of the first characters in a1
// and a2 and the length in cnt1.
// elem_size is the element size in bytes: either 1 or 2.
// There are two implementations. For arrays >= 8 bytes, all
// comparisons (including the final one, which may overlap) are
// performed 8 bytes at a time. For strings < 8 bytes, we compare a
// halfword, then a short, and then a byte.

void MacroAssembler::string_equals(Register a1, Register a2,
                                   Register result, Register cnt1, int elem_size)
```

Hardware-accelerated instructions for ARMv8 Crypto

JDK / JDK-8210858

AArch64: remove Math.log intrinsic

Resolved

Details

Type:	Enhancement	Resolution:	Fixed
Priority:	P4	Fix Version/s:	23
Affects Version/s:	11, 17, 21, 22, 23		
Component/s:	hotspot		
Labels:	amazon-interest openjdk oracle-triage-11 performance		
Subcomponent:	compiler		
Resolved In Build:	b08		
CPU:	aarch64		

Description

Remove Math.log instrinsic (it is already disabled).

Reason: the available implementation gave incorrect results:

```
public class LogTest {
   public static void main(String[] args) {
      double x = 4.9E-324;
      System.out.println(Math.log(x));
      System.out.println(StrictMath.log(x));
   }
}
```

Should be -744.4400719213812, result is -710.989276736877.
Correct on Intel x86.

Second reason: We do not have proof of the monotonicity for the patch of this intrinsic.

Sometimes intrinsics needs to be disabled

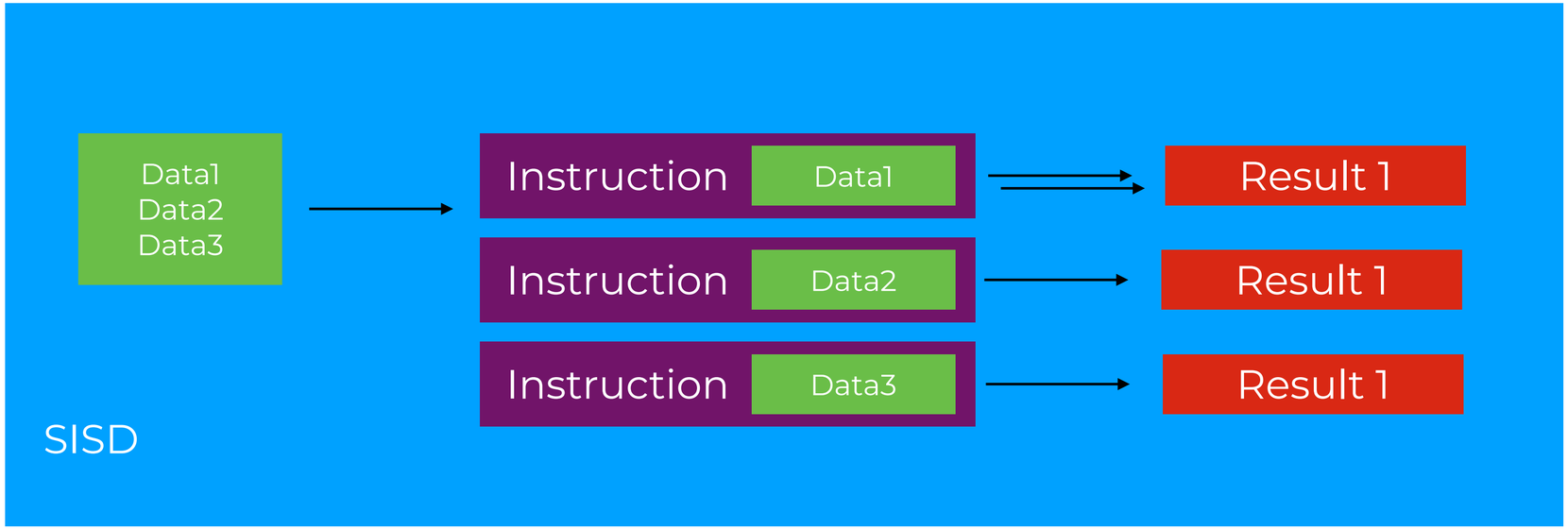
arm | Developer

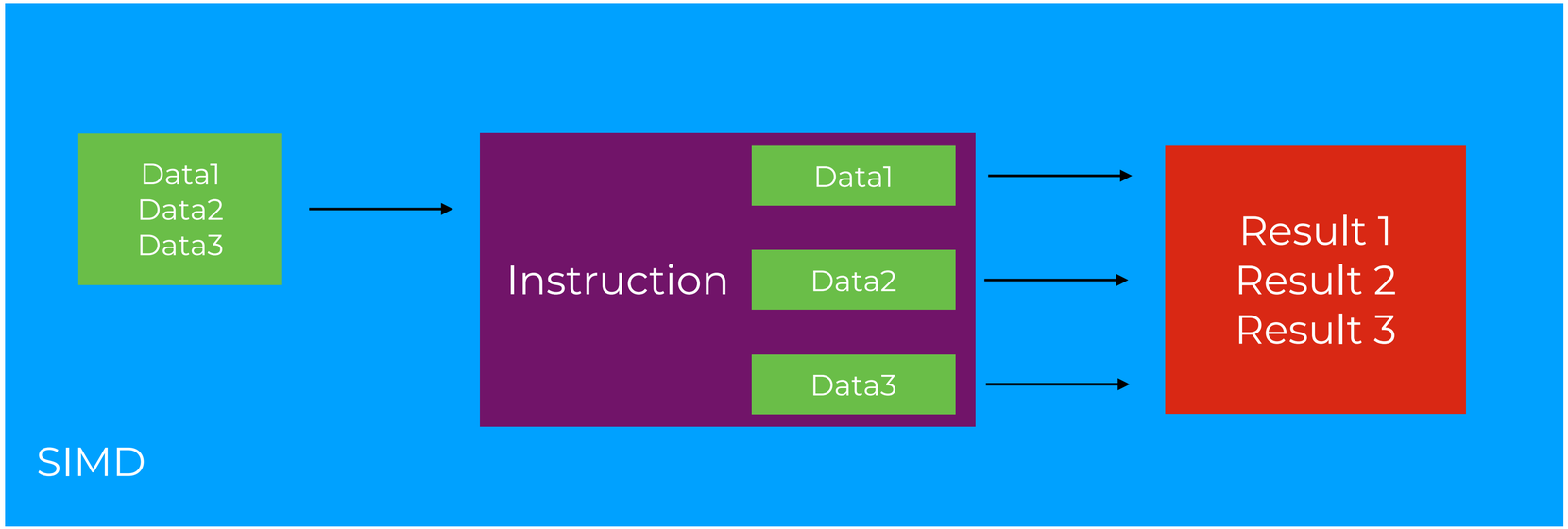
Technical Information | Resources | Tools and Software | Support and Training | Works Well With

Overview

Arm Neon technology is an advanced Single Instruction Multiple Data (SIMD) architecture extension for the A-profile and R-profile processors.

Neon technology is a packed SIMD architecture. Neon registers are considered as vectors of elements of the same data type, with Neon instructions operating on multiple elements simultaneously. Multiple data types are supported by the technology, including floating-point and integer operations.

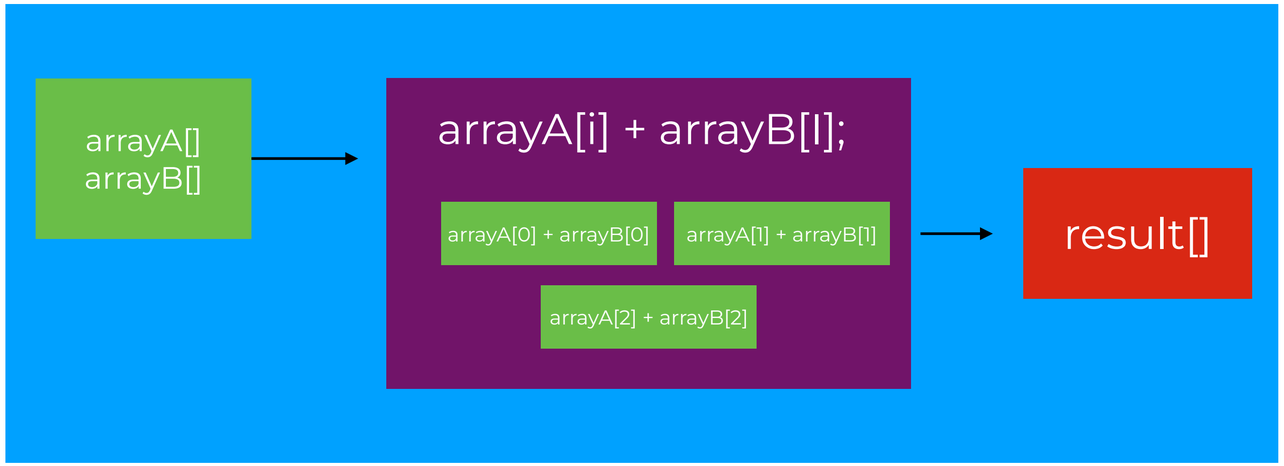




Panama Vector API and Vector Databases

```java
public class VectorizationExample {
    public static void main(String[] args) {
        int[] arrayA = new int[1000];
        int[] arrayB = new int[1000];
        int[] result = new int[1000];

        // Initialize arrays
        for (int i = 0; i < arrayA.length; i++) {
            arrayA[i] = i;
            arrayB[i] = i * 2;
        }

        // Perform element-wise addition (this can be autovectorized)
        for (int i = 0; i < arrayA.length; i++) {
            result[i] = arrayA[i] + arrayB[i];
        }

        // Output some of the results
        for (int i = 0; i < 10; i++) {
            System.out.println("Result[" + i + "] = " + result[i]);
        }
    }
}
```

JIT



Autovectorization

OpenJDK

Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code
Mercurial
GitHub

Tools
Git
jtreg harness

Groups
(overview)
Adoption
Build
Client Libraries
Compatibility & Specification Review
Compiler
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization
JMX
Members
Networking
Porters
Quality
Security
Serviceability
Vulnerability
Web

Projects
(overview, archive)
Amber
Babylon
CRaC
Caciocavallo
Closures

JEP 469: Vector API (Eighth Incubator)

Owner	Paul Sandoz
Type	Feature
Scope	JDK
Status	Closed / Delivered
Release	23
Component	core-libs
Discussion	panama dash dev at openjdk dot org
Effort	XS
Duration	XS
Relates to	JEP 460: Vector API (Seventh Incubator)
Reviewed by	Vladimir Ivanov
Endorsed by	John Rose
Created	2024/02/27 20:50
Updated	2024/07/15 15:59
Issue	8326878

Summary

Introduce an API to express vector computations that reliably compile at runtime to optimal vector instructions on supported CPU architectures, thus achieving performance superior to equivalent scalar computations.

History

We first proposed the Vector API in JEP 338, integrated into JDK 16 as an incubating API. We proposed further rounds of incubation in JEP 414 (integrated into JDK 17), JEP 417 (JDK 18), JEP 426 (JDK 19), JEP 438 (JDK 20), JEP 448 (JDK 21), and JEP 460 (JDK 22).

We propose here to re-incubate the API in JDK 23 with no API changes and no substantial implementation changes relative to JDK 22.

The Vector API will incubate until necessary features of Project Valhalla become available as preview features. At that time, we will adapt the Vector API and its implementation to use them, and will promote the Vector API from incubation to preview.

Supports NEON SVE/2

Panama Vector API and Vector Databases

arm Developer Develop Blogs Community CPU & Hardware Support Documentation

Developer Hub / Learning Paths / Learning-Paths / Servers and Cloud Computing / Tune the Performance of the Java Garbage Collector

Tune the Performance of the Java Garbage Collector

Tune the Performance of the Java Garbage Collector

- Introduction
- Overview
- Setup
- Types of Garbage Collector
- Example Application
- Basic GC Tuning Options
- Intermediate GC Tuning Options
- Next Steps

Log an issue
Fork and edit
Discuss on Discord

About this Learning Path

Skill level: Introductory
Reading time: 45 min
Last updated: 23 Apr 2025

Author: Kieran Hejmadi
Arm IP: Neoverse
Tags: Performance and Architecture | Linux | Java | Runbook

Who is this for?

This Learning Path is for Java developers aiming to optimize application performance on Arm-based servers, especially those migrating applications from x86-based to Arm-based instances.

What will you learn?

Upon completion of this learning path, you will be able to:

- Describe the key differences between individual Java Garbage Collectors (GCs).
- Monitor and interpret Garbage Collector performance metrics.
- Adjust core parameters to optimize performance for your specific workload.

Prerequisites

Before starting, you will need the following:

Garbage Collectors unaffected

Some memory warranties ale lower

Garbage Collectors

Java on Graviton

Java is a general-purpose programming language. Compiled Java code can run on all platforms that support Java, without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. Note: there are multiple programming languages such as Kotlin, Scala, and Groovy that compile to byte code and run on top of the JVM, so those also tend to be highly portable across architectures. *Wikipedia*

Java is well supported and generally performant out-of-the-box on arm64. Amazon Corretto, a no-cost, multiplatform, production-ready distribution of the Open Java Development Kit (OpenJDK) supports Graviton-powered instances. While Java 8 is fully supported on Arm processors, some customers haven't been able to obtain Graviton's full performance benefit until they switched to Java 11.

This page includes specific details about building and tuning Java application on Graviton.

Java Graviton Migration Checklist

This checklist summarizes portions of the Java on Graviton section and can be helpful for getting started. It is not a comprehensive summary, so we also recommend reading through all the content below.

- Check AMI & Kernel Version support
  - Pre-2020 Linux distributions are unlikely to contain the right optimizations.
  - Amazon Linux: AL2023 is ideal. AL2 is fine with a recent kernel (i.e. not 4.14). AL2 is EOL in June 2025.
  - Ubuntu: Use at least Ubuntu 20.04. More recent versions are even better.
  - Red Hat Linux: RHEL9 is ideal. Use at least RHEL8.2 (be aware kernel uses unusual 64KB memory pages).
  - For the full list, see Operating Systems available for Graviton based instances.

https://github.com/aws/aws-graviton-getting-started/blob/main/java.md

Java JVM Options

There are numerous options that control the JVM and may lead to better performance.

- Flags `-XX:-TieredCompilation -XX:ReservedCodeCacheSize=64M -XX:InitialCodeCacheSize=64M` have shown large (1.5x) improvements in some Java workloads. Corretto 17 needs two additional flags: `-XX:CICompilerCount=2 -XX:CompilationMode=high-only`. `ReservedCodeCacheSize` / `InitialCodeCacheSize` should be equal and can be in range: 64M...127M. The JIT compiler stores generated code in the code cache. The flags change the size of the code cache from the default 240M to the smaller one. The smaller code cache may help CPU to improve the caching and prediction of jitted code. The flags disable the tiered compilation to make the JIT compiler able to use the smaller code cache. These are helpful on some workloads but can hurt on others so testing with and without them is essential.

- Crypto algorithm AES/GCM used by TLS has been optimized for Graviton. On Graviton2 GCM encrypt/decrypt performance improves by 3.5x to 5x. The optimization is enabled by default in Corretto and OpenJDK 18 and later. The optimization has been backported to Corretto and OpenJDK 11 and 17 and can be enabled with the flags `-XX:+UnlockDiagnosticVMOptions -XX:+UseAESCTRIntrinsics`. As an alternative, you can use Amazon Corretto Crypto Provider JNI libraries.

https://github.com/aws/aws-graviton-getting-started/blob/main/java.md

1 Answer

Sorted by: Highest score (default)

3

Native code needs to be compiled for the specific architecture and operating system where it will be executed. You can not just use an Intel library in an ARM CPU. You need to get the appropriate build from the provider, or if you have the source build it yourself.

JNativeHook 2.1.0 is too old and didn't include macOS (Darwin) with ARM as a target. Apple M1 chip based computers were not even released at that time.

You need to update to the current release, 2.2.2 at this time, which supports the Darwin/aarch64 target. The distribution includes the appropriate version of the native library.

Share Improve this answer Follow

edited Apr 3, 2024 at 16:30
Basil Bourque
342k • 124 • 936 • 1.3k

answered Apr 3, 2024 at 12:41
aled
26.1k • 4 • 34 • 48

@basilBorque I am using this updated library but after sometime key pressing it's not working. – SYED MUSTAFA HUSSAIN Apr 5, 2024 at 5:34

Then the issue from your question is solved, right? – aled Apr 5, 2024 at 9:58

yes, resolved. Thank you! – SYED MUSTAFA HUSSAIN Apr 5, 2024 at 11:48

Add a comment

JNI...

GraalVM

Native Image ARM64 Support

As of version 21.0, GraalVM distributions for Linux on ARM 64-bit system are available. Unless explicitly noted within the limitations, all Native Image features should work on this architecture.

Limitations

Mostly all Native Image features are supported on ARM64, except for the limitations described below.

- `-R:[+|-]WriteableCodeCache`: must be disabled.
- `--libc=<value>`: `musl` is not supported.
- `--gc=<value>`: The G1 garbage collector (`G1`) is not supported.

For more information about these options, please refer to Native Image Options and Native Image Hosted and Runtime Options.

Support for Apple M1 (`darwin-aarch64`) #2666

Closed Support for Apple M1 (`darwin-aarch64`) #2666

fniephaus opened on Jul 10, 2020 · edited by fniephaus

Edits

Update from 2023/02/01:

Support for Apple Silicon has landed in the 22.1 release of GraalVM. We gradually ship more GraalVM components for `darwin-aarch64` machines in new releases.

The latest state can be tested with the latest GraalVM dev build. Please help testing and report bugs!

Component	Status
Compiler	experimental
Native Image	experimental
LLVM runtime	experimental
LLVM toolchain	experimental
JavaScript	experimental
Node.js	experimental
Java on Truffle	experimental
Python	experimental
Ruby	experimental
R	planned
WebAssembly	planned
VisualVM	experimental

Update from 2022/03/22:

Apple M1 support for @GraalVM Native Image, libgraal, JavaScript, and Espresso has landed! 🎉🚀
M1 support for Ruby has also landed now.

You can now grab a GraalVM dev build for `darwin-aarch64` at: https://github.com/graalvm/graalvm-ce-dev-builds/releases/latest

Please help testing and report bugs!

Windows ARM64 Support #9215

Closed

Glavo opened on Jun 30, 2024

Since the release of Qualcomm Snapdragon X Elite, Windows ARM64 has received more attention. I wonder if there are plans for GraalVM to support Windows ARM64 platform now?

1

Glavo added **feature** on Jun 30, 2024

selhagani self-assigned this on Jul 1, 2024

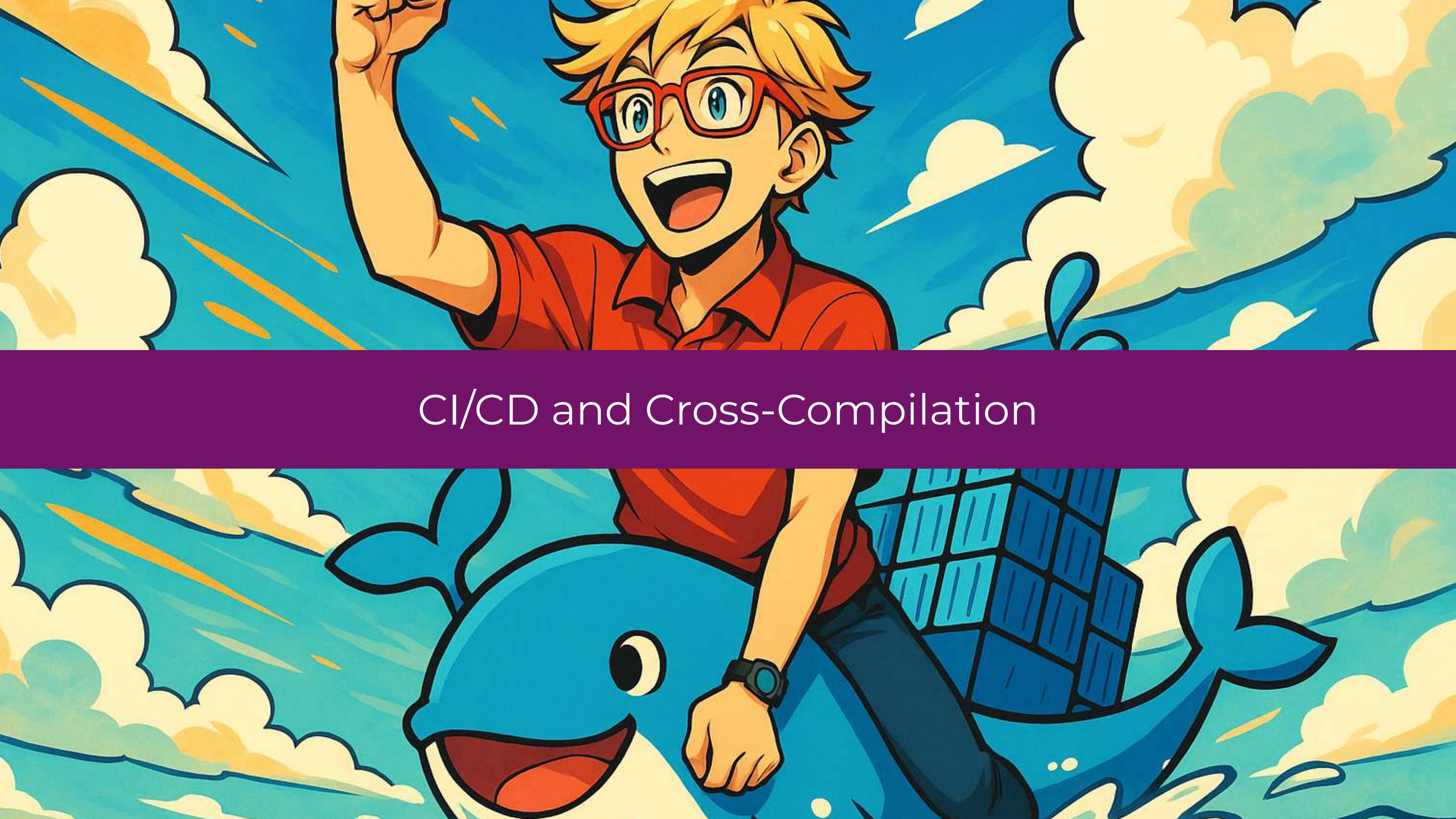
selhagani on Jul 1, 2024 Member

Hi @Glavo, thank you for reaching out to us!

Support for Windows ARM64 is not planned for GraalVM currently.

1 7

As a day of check (27th of April) no support for Windows ARM64



CI/CD and Cross-Compilation

AND THEN WE TOLD THEM
WRITE ONCE RUN ANYWHERE
memegenerator.net

Just JAR

DAD WHAT IS COMPATIBILITY ISSUE
WE DONT KNOW SON
WE ARE DOCKER USERS

Just JAR

🐳 **Docker Buildx is a Docker CLI plugin that:**

- uses the modern **BuildKit** backend (a faster, more flexible way to build images),
- enables building for **multiple architectures** at once (linux/amd64, linux/arm64, linux/s390x, etc.),
- supports **layer caching** across builds (locally, remote registry, S3),
- **automatically creates multi-architecture manifests** (docker manifest),
- allows **different output types** (pushing to registries, tar files, cache exports),
- can **build images remotely** on another machine.

In short:
Docker Buildx = Docker build on steroids, optimized for multi-architecture and modern CI/CD pipelines.

Docker Buildx

QEMU is a hardware emulator and CPU instruction translator.

In the Docker context:

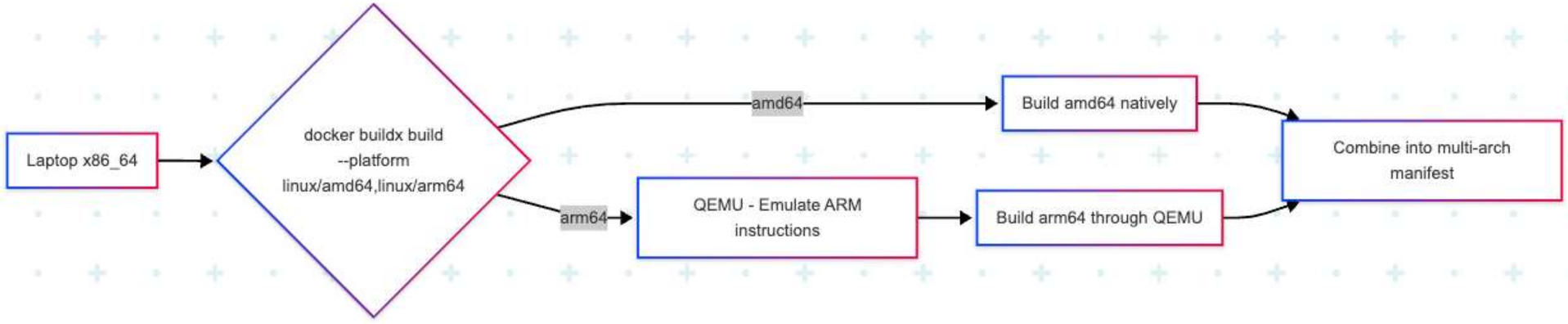
- it allows you to **run linux/arm64 images** on an x86_64 (Intel/AMD) machine,
- it **emulates** a different CPU architecture, so your container can run without modification.

Example: you have an x86 laptop (Intel), but you want to build an AWS Graviton (aarch64) image.
QEMU intercepts and translates ARM instructions on-the-fly, so the container *works* even though your machine isn't ARM.

Docker uses QEMU via the **binfmt_misc** kernel feature:

- at startup (docker/setup-qemu-action), it registers binary format handlers (qemu-aarch64, qemu-arm, etc.),
- after that, Docker Buildx "sees" ARM as a valid build platform, even if your host is x86.

QEMU


Laptop x86_64
docker buildx build
--platform
linux/amd64,linux/arm64
amd64
Build amd64 natively
arm64
QEMU - Emulate ARM
instructions
Build arm64 through QEMU
Combine into multi-arch
manifest

QEMU

```yaml
strategy:
  matrix:
    arch: [amd64, arm64]
runs-on: ${{ matrix.arch == 'arm64' && 'graviton-runner' || 'ubuntu-latest' }}
steps:
  - uses: actions/checkout@v4
  - name: Set up JDK ${{ env.JAVA_VERSION }}
    uses: actions/setup-java@v4
    with:
      distribution: corretto
      architecture: ${{ matrix.arch }}
  - name: Build & test
    run: ./gradlew build
  - name: Build multi-arch image
    run: docker buildx build --platform linux/amd64,linux/arm64 …
```

Cross-build Docker Images

```
RUN./mvnw -Pnative native:compile
```

You need to build native deps inside the image

Building image is a slog 🐢

QEMU is ~5–15 × slower for CPU-heavy steps

Platform	Native ARM64 Runners?	Details
GitHub Actions	✅ Yes	Hosted runner: ubuntu-22.04-arm64 (public preview as of April 2024) You can also run self-hosted ARM64 runners (on EC2 C7g, C8g).
CircleCI	✅ Yes	Supports native **ARM64 resource classes** (arm.medium, arm.large, etc.).
AWS CodeBuild	✅ Yes	Supports ARM_CONTAINER environment type. Official ARM64 build images (e.g., aws/codebuild/amazonlinux2-aarch64-standard:3.0).
GitLab CI	✅ Yes	Supports native **ARM64 hosted runners**
Google Cloud Build	🚧 Limited	Still **x86_64-only** runners officially. You can hack it by using self-hosted runners on **C4A Axion VMs** (native ARM64) but not "official" yet.
Azure DevOps	🚧 Limited	No native hosted ARM64 runners yet. But you can spin up a **self-hosted ARM64 agent** easily on Azure VM ARM64 instances (e.g., Dpsv5-series).
Bitbucket Pipelines	🚫 No	Only x86_64 runners officially. Self-hosted runners must be x86_64.

CI Support for ARM64 Runners

Performance

Trying to save on AWS? Click here to get a demo of Vantage from a FinOps expert.

Region: US East (N. Virginia) | Pricing Unit: Instance | Cost: Hourly | Reserved: 1-year - No Upfront | Visible: Columns | Compare | Clear Filters | Export | Search...

Name	API Name	Instance Memory	vCPUs	Instance Storage	Network Performance	On Demand	Linux Reserved cos
Filter...	Filter...	Min Mem: 0	Min vCPUs: 0	Min Storage: 0	Filter...	Filter...	Filter...
T4G Nano	t4g.nano	0.5 GiB	2 vCPUs for a 1h 12m burst	EBS only	Up to 5 Gigabit	$0.0042 hourly	$0.0026 ho
T3A Nano	t3a.nano	0.5 GiB	2 vCPUs for a 1h 12m burst	EBS only	Up to 5 Gigabit	$0.0047 hourly	$0.0029 ho
T3 Nano	t3.nano	0.5 GiB	2 vCPUs for a 1h 12m burst	EBS only	Up to 5 Gigabit	$0.0052 hourly	$0.0033 ho
T2 Nano	t2.nano	0.5 GiB	1 vCPUs for a 1h 12m burst	EBS only	Low to Moderate	$0.0058 hourly	$0.0036 ho
T4G Micro	t4g.micro	1.0 GiB	2 vCPUs for a 2h 24m burst	EBS only	Up to 5 Gigabit	$0.0084 hourly	$0.0053 ho
T3A Micro	t3a.micro	1.0 GiB	2 vCPUs for a 2h 24m burst	EBS only	Up to 5 Gigabit	$0.0094 hourly	$0.0059 ho
T3 Micro	t3.micro	1.0 GiB	2 vCPUs for a 2h 24m burst	EBS only	Up to 5 Gigabit	$0.0104 hourly	$0.0065 ho
T2 Micro	t2.micro	1.0 GiB	1 vCPUs for a 2h 24m burst	EBS only	Low to Moderate	$0.0116 hourly	$0.0072 ho
T4G Small	t4g.small	2.0 GiB	2 vCPUs for a 4h 48m burst	EBS only	Up to 5 Gigabit	$0.0168 hourly	$0.0105 ho
T3A Small	t3a.small	2.0 GiB	2 vCPUs for a 4h 48m burst	EBS only	Up to 5 Gigabit	$0.0188 hourly	$0.0118 ho
T1 Micro	t1.micro	0.613 GiB	1 vCPUs	EBS only	Very Low	$0.0200 hourly	$0.0140 ho
T3 Small	t3.small	2.0 GiB	2 vCPUs for a 4h 48m burst	EBS only	Up to 5 Gigabit	$0.0208 hourly	$0.0130 ho
T2 Small	t2.small	2.0 GiB	1 vCPUs for a 4h 48m burst	EBS only	Low to Moderate	$0.0230 hourly	$0.0144 ho
A1 Medium	a1.medium	2.0 GiB	1 vCPUs	EBS only	Up to 10 Gigabit	$0.0255 hourly	$0.0161 ho
T4G Medium	t4g.medium	4.0 GiB	2 vCPUs for a 4h 48m burst	EBS only	Up to 5 Gigabit	$0.0336 hourly	$0.0211 ho
C6G Medium	c6g.medium	2.0 GiB	1 vCPUs	EBS only	Up to 10 Gigabit	$0.0340 hourly	$0.0214 ho
C7G Medium	c7g.medium	2.0 GiB	1 vCPUs	EBS only	Up to 12.5 Gigabit	$0.0363 hourly	$0.0239 ho
T3A Medium	t3a.medium	4.0 GiB	2 vCPUs for a 4h 48m burst	EBS only	Up to 5 Gigabit	$0.0376 hourly	$0.0236 ho

Trying to save on AWS? Click here to get a demo of Vantage from a FinOps expert.

Region: US East (N. Virginia) | Pricing Unit: Instance | Cost: Hourly | Reserved: 1-year - No Upfront | Visible: Columns | Compare | Clear Filters | Export | Search...

- 1-year - No Upfront
- 1-year - Partial Upfront
- 1-year - Full Upfront
- 3-year - No Upfront
- 3-year - Partial Upfront
- 3-year - Full Upfront
- 1-year convertible - No Upfront
- 1-year convertible - Partial Upfront
- 1-year convertible - Full Upfront
- 3-year convertible - No Upfront
- 3-year convertible - Partial Upfront
- 3-year convertible - Full Upfront

Name			vCPUs	Instance Storage	Network Performance	On Demand	Linux Reserved cos
Filter...			Min vCPUs: 0	Min Storage: 0	Filter...	Filter...	Filter...
U-12TB1 Metal			448 vCPUs	EBS only	100 Gigabit	unavailable	unavail
HPC6ID 32xlarge			64 vCPUs	EBS only	200 Gigabit	unavailable	unavail
HPC7A 12xlarge			24 vCPUs	EBS only	300 Gigabit	unavailable	unavail
HPC6A 48xlarge			96 vCPUs	EBS only	100 Gigabit	unavailable	unavail
HPC7A 48xlarge			96 vCPUs	EBS only	300 Gigabit	unavailable	unavail
TRN2 48xlarge			192 vCPUs	EBS only	3200 Gigabit	unavailable	unavail
U-9TB1 Metal			448 vCPUs	EBS only	100 Gigabit	unavailable	unavail
U-6TB1 Metal			448 vCPUs	EBS only	100 Gigabit	unavailable	unavail
I2 Large	i2.large	15.0 GiB	2 vCPUs	EBS only	Moderate	unavailable	unavail
MAC1 Metal	mac1.metal	32.0 GiB	12 vCPUs	EBS only	25 Gigabit	unavailable	unavail
MAC2-M2 Metal	mac2-m2.metal	24.0 GiB	8 vCPUs	EBS only	10 Gigabit	unavailable	unavail
MAC2 Metal	mac2.metal	16.0 GiB	8 vCPUs	EBS only	10 Gigabit	unavailable	unavail
MAC2-M2PRO Metal	mac2-m2pro.metal	32.0 GiB	12 vCPUs	EBS only	10 Gigabit	unavailable	unavail
U-18TB1 Metal	u-18tb1.metal	18432.0 GiB	448 vCPUs	EBS only	100 Gigabit	unavailable	unavail
MAC2-M1ULTRA Metal	mac2-m1ultra.metal	128.0 GiB	20 vCPUs	EBS only	10 Gigabit	unavailable	unavail
U-24TB1 Metal	u-24tb1.metal	24576.0 GiB	448 vCPUs	EBS only	100 Gigabit	unavailable	unavail
High Storage Eight Extra Large	hs1.8xlarge	117.0 GiB	16 vCPUs	EBS only	10 Gigabit	unavailable	unavail
Cluster Compute Eight Extra Large	cc2.8xlarge	60.5 GiB	32 vCPUs	EBS only	10 Gigabit	unavailable	unavail

README MIT license

ARM Java Benchmarks

This benchmark suite was created for the Devoxx UK presentation **"Running Java on ARM - is it worth it?"** comparing performance and cost efficiency of Java applications across ARM-based cloud instances.

Project Overview

Automated infrastructure and benchmarking tools for comparing:

- Google Cloud Axion/Tau instances
- AWS Graviton instances (1-4)
- Azure Ampere Altra instances

Includes full-stack automation for:

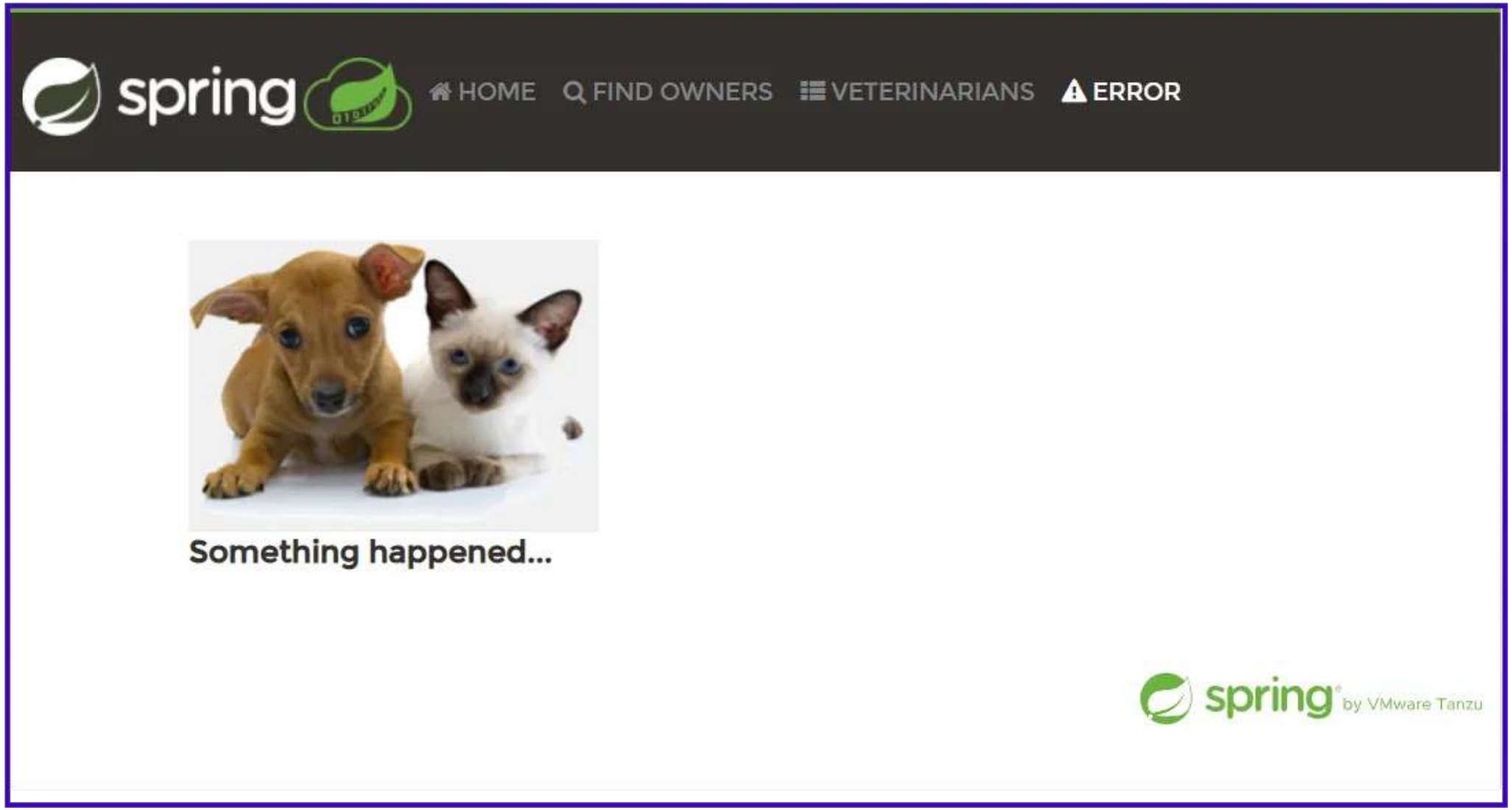
- Cloud instance provisioning
- Java application deployment
- JMeter performance testing
- JVM optimization analysis

Quick Start

Prerequisites

- Cloud accounts (GCP, AWS, Azure)
- Cloud CLIs installed (`gcloud` , `aws` , `az`)
- SSH keys configured for each cloud

https://github.com/ArturSkowronski/jdk-arm-benchmarks



Spring PetClinic

README MIT license

ARM Java Benchmarks

This benchmark suite was created for the Devoxx UK presentation **"Running Java on ARM - is it worth it?"** comparing performance and cost efficiency of Java applications across ARM-based cloud instances.

Project Overview

Automated infrastructure and benchmarking tools for comparing:

- Google Cloud Axion/Tau instances
- AWS Graviton instances (1-4)
- Azure Ampere Altra instances

Includes full-stack automation for:

- Cloud instance provisioning
- Java application deployment
- JMeter performance testing
- JVM optimization analysis

Quick Start

Prerequisites

- Cloud accounts (GCP, AWS, Azure)
- Cloud CLIs installed (`gcloud` , `aws` , `az`)
- SSH keys configured for each cloud

https://github.com/ArturSkowronski/jdk-arm-benchmarks

```
gcloud compute machine-types list --filter="name:c4a-standard-2" --format="table[box](zone,name)"
```

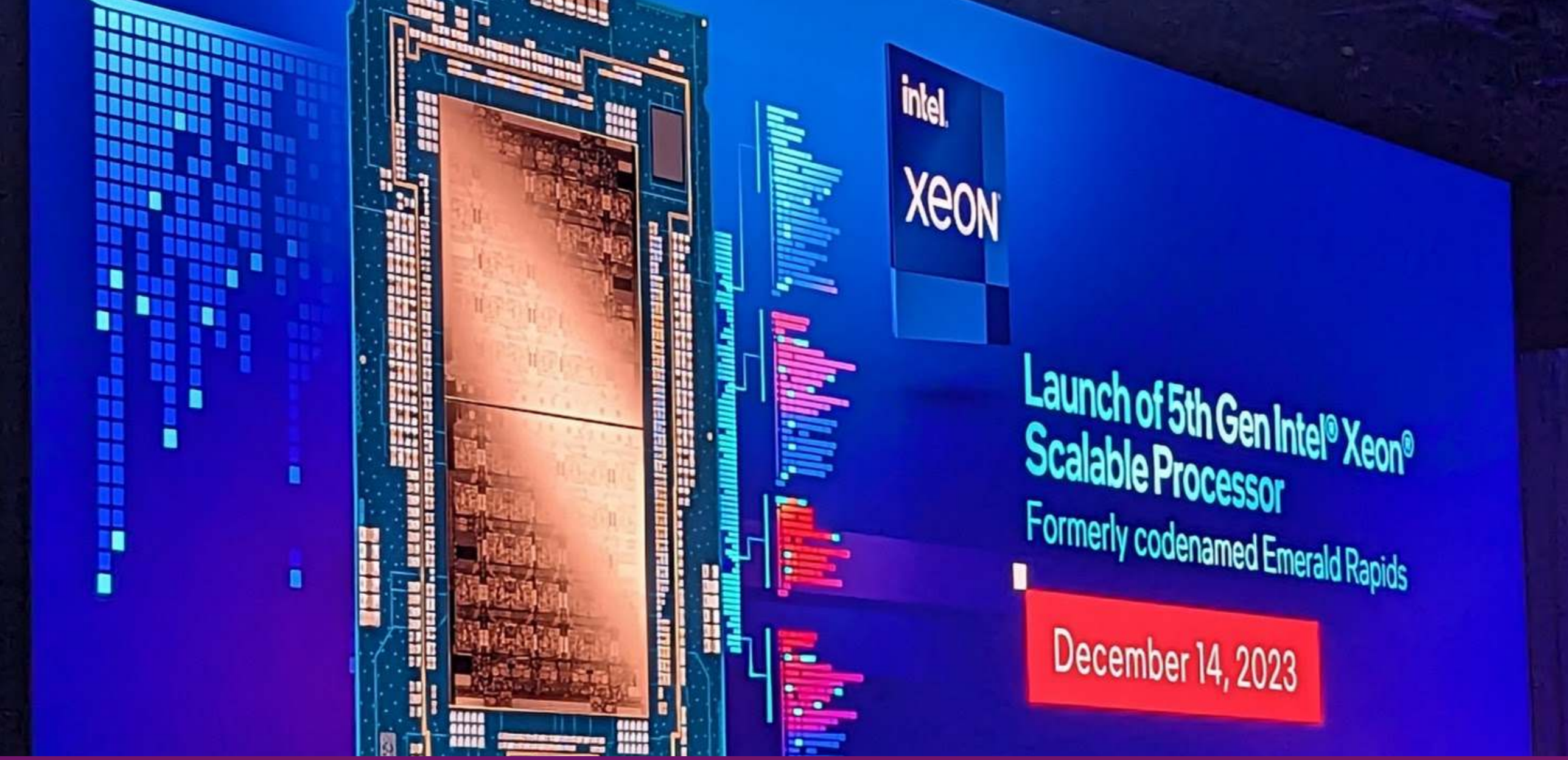
ZONE	NAME
us-central1-a	c4a-standard-2
us-central1-b	c4a-standard-2
us-central1-c	c4a-standard-2
europe-west1-b	c4a-standard-2
europe-west1-c	c4a-standard-2
europe-west1-d	c4a-standard-2
us-west1-a	c4a-standard-2
us-west1-c	c4a-standard-2
asia-east1-a	c4a-standard-2
asia-east1-b	c4a-standard-2
asia-east1-c	c4a-standard-2
us-east1-b	c4a-standard-2
us-east1-c	c4a-standard-2
us-east1-d	c4a-standard-2
asia-northeast1-b	c4a-standard-2
asia-northeast1-c	c4a-standard-2
asia-southeast1-a	c4a-standard-2
asia-southeast1-b	c4a-standard-2
asia-southeast1-c	c4a-standard-2
us-east4-a	c4a-standard-2
us-east4-b	c4a-standard-2
us-east4-c	c4a-standard-2
europe-west2-a	c4a-standard-2
europe-west2-b	c4a-standard-2
europe-west2-c	c4a-standard-2
europe-west3-c	c4a-standard-2
europe-west3-a	c4a-standard-2
europe-west3-b	c4a-standard-2
asia-south1-b	c4a-standard-2
asia-south1-c	c4a-standard-2
europe-west4-c	c4a-standard-2
europe-west4-b	c4a-standard-2
europe-west4-a	c4a-standard-2
europe-north1-b	c4a-standard-2
europe-north1-a	c4a-standard-2
australia-southeast2-a	c4a-standard-2
us-east7-a	c4a-standard-2
us-west8-c	c4a-standard-2

```
❯ ./create_axion_instance.sh jdk-arm-benchmarks europe-central2-a
Setting gcloud project to jdk-arm-benchmarks
WARNING: Your active project does not match the quota project in your local Application Default Credentials file. This might result in unexpected quota issues.

To update your Application Default Credentials quota project, use the `gcloud auth application-default set-quota-project` command.
Updated property [core/project].
Creating Axion VM instance: test-app-instance
API [compute.googleapis.com] not enabled on project [jdk-arm-benchmarks]. Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [compute.googleapis.com] on project [jdk-arm-benchmarks]...
Operation "operations/acf.p2-842397957748-85c3c84f-7e98-41cd-8e1e-27d2bd6b1fee" finished successfully.
ERROR: (gcloud.compute.instances.create) Could not fetch resource:
 - Invalid value for field 'resource.machineType': 'https://compute.googleapis.com/compute/v1/projects/jdk-arm-benchmarks/zones/europe-central2-a/machineTypes/c4
a-standard-2'. Machine type with name 'c4a-standard-2' does not exist in zone 'europe-central2-a'.

Use `gcloud compute machine-types list --zones` to see the available machine  types.
```

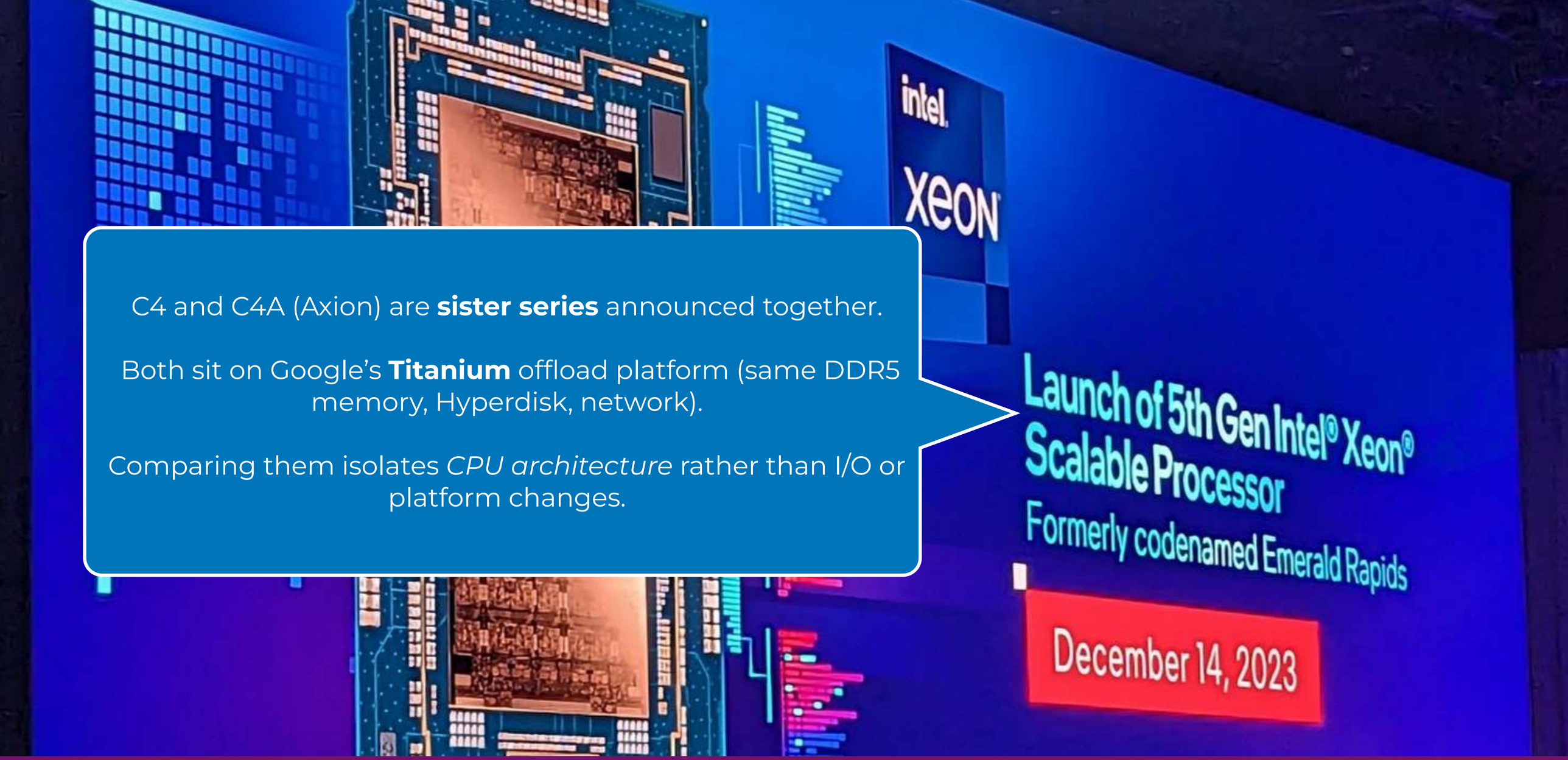
```
To update your Application Default Credentials quota project, use the `gcloud auth application-default set-quota-project` command.
Updated property [core/project].
Creating Axion VM instance: test-app-instance
Created [https://www.googleapis.com/compute/v1/projects/jdk-arm-benchmarks/zones/us-central1-a/instances/test-app-instance].
NAME               ZONE          MACHINE_TYPE    PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP    STATUS
test-app-instance  us-central1-a  c4a-standard-2               10.128.0.2   34.134.183.19  RUNNING
Configuring firewall rule for port 8080
Creating firewall...⠶Created [https://www.googleapis.com/compute/v1/projects/jdk-arm-benchmarks/global/firewalls/default-allow-http-8080].
Creating firewall...done.
NAME                    NETWORK  DIRECTION  PRIORITY  ALLOW     DENY  DISABLED
default-allow-http-8080 default  INGRESS    1000      tcp:8080        False
Listing instances to obtain external IP:
```

NAME	ZONE	MACHINE_TYPE	INTERNAL_IP	EXTERNAL_IP	STATUS
test-app-instance	us-central1-a	c4a-standard-2			RUNNING

Google Cloud (T2A, Axion)



Axion vs Xeon vs T2A



C4 and C4A (Axion) are **sister series** announced together.

Both sit on Google's **Titanium** offload platform (same DDR5 memory, Hyperdisk, network).

Comparing them isolates *CPU architecture* rather than I/O or platform changes.

Axion vs Xeon vs T2A

CoreMark®

An EEMBC Benchmark

About · FAQ · Download · Scores · Submit Score · Google Group

What is CoreMark?

CoreMark is a simple, yet sophisticated benchmark that is designed specifically to test the functionality of a processor core. Running CoreMark produces a single-number score allowing users to make quick comparisons between processors.

Latest Certified Scores

While anyone can upload a CoreMark score, certified scores have passed the rigorous analysis of the EEMBC Certification Lab.

Renesas Electronics RX66T

CoreMarks: 721

CoreMarks/MHz: 6.01

EMBC CERTIFIED

Renesas Electronics RX66T

CoreMarks: 961

CoreMarks/MHz: 6.00

EMBC CERTIFIED

Renesas Electronics RX72T

CoreMarks: 1.20e+3

EMBC

Details

EEMBC's CoreMark® is a benchmark that measures the performance of microcontrollers (MCUs) and central processing units (CPUs) used in embedded systems. Replacing the antiquated Dhrystone benchmark, Coremark contains implementations of the following algorithms: list processing (find and sort), matrix manipulation (common matrix operations), state machine (determine if an input stream contains valid numbers), and CRC (cyclic redundancy check). It is designed to run on devices from 8-bit microcontrollers to 64-bit microprocessors.

CoreMark News

News: A new whitepaper and video explain how to port CoreMark-Pro to bare-metal

News: EEMBC Standardizes CoreMark® Energy Measurement with New ULPMark™ Benchmark

News: STMicroelectronics New STM32L4 Microcontroller Series Devices Show Off EEMBC ULPBench and CoreMark Results

News: STM32F469/479 deliver the industry's highest ARM® Cortex® M4 processor

Axion vs Xeon vs T2A

phoronix

ARTICLES & REVIEWS | NEWS ARCHIVE | FORUMS | PREMIUM | CONTACT | CATEGORIES

ENHANCED BY Google Search

Latest Linux Hardware Reviews, Open-Source News & Benchmarks

FEX 2505 Released With Many Fixes For Running x86_64 Binaries On Linux AArch64

7 Hours Ago - Free Software - FEX 2505 - 2 Comments

FEX 2505 is out today as the newest version of this user-space software that enables running x86_64 binaries on ARM64/AArch64 Linux hosts with great speed that a number of games and applications are enjoying much success with this emulator.

Redox OS Seeing More Software Porting, Many Internal Improvements

10 Hours Ago - Operating Systems - Redox OS - 9 Comments

The open-source, Rust-based Redox OS operating system has published their April 2025 status report to outline interesting changes made over the past several weeks to this innovative project.

Featured 21-Way Intel Core / AMD Ryzen Linux Laptop Comparison On Ubuntu 25.04

5 May 10:32 AM EDT - Computers - 10 Comments

As part of fresh re-testing of existing laptops on-hand given the recent release of Ubuntu 25.04 and then also recent Linux reviews of some interesting models like the Framework Laptop 13 with AMD Strix Point and the Lenovo ThinkPad X1 Carbon Gen 13 Aura Edition, I have been running a lot of Linux laptop benchmarks the past few weeks. I ended up taking things a bit further after those reviews and have now extended it to a 21-way laptop comparison of AMD Ryzen and Intel Core SoCs from the past few generations in looking at their performance on Ubuntu 25.04 across more than 200 benchmarks.

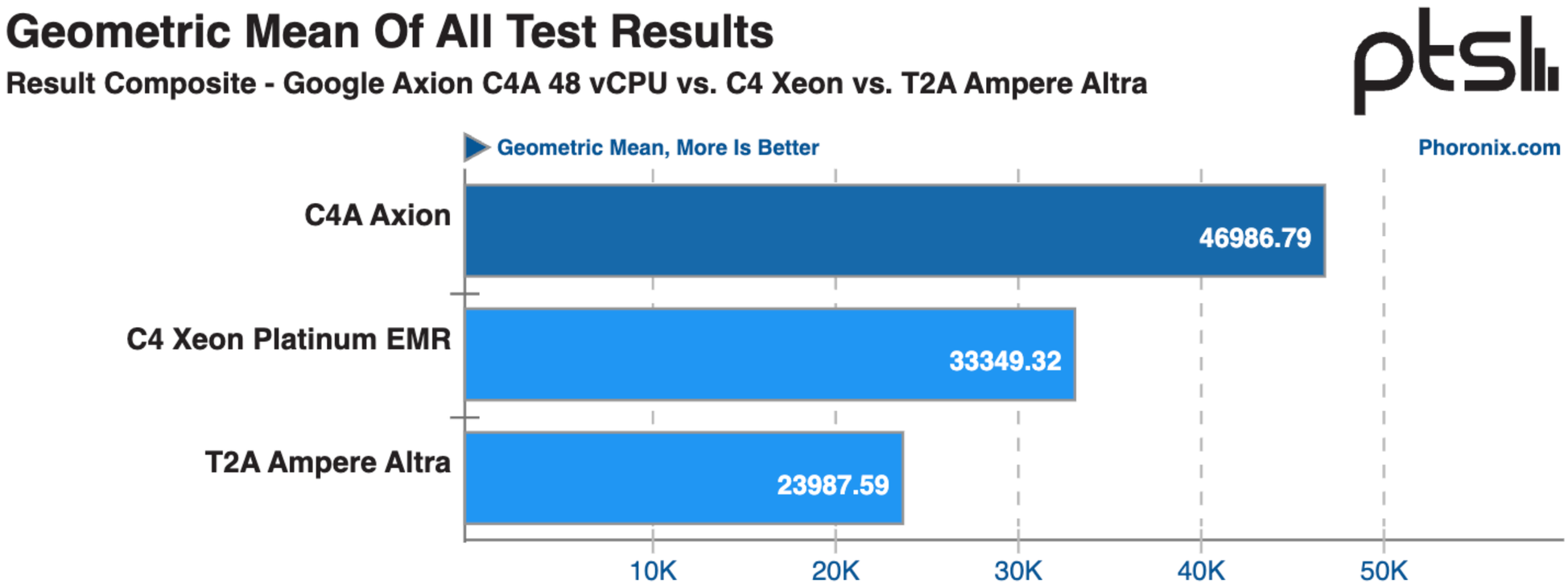
Most Popular News This Week

- Bytedance Proposes Faster Linux Inter-Process Communication With "Run Process As Library"
- Valve's Proton 10.0 Beta Released With More Windows Games Now Playable On Linux
- Valve Releases Steam Survey Results For April 2025
- KDE Plasma 6.4 Tackles An 18 Year Old Feature Request, More Wayland Protocols Added
- Wine 10.7 Brings An Exciting Performance Optimization
- Plasma LTS Releases Being Discontinued, Better KDE Telemetry Like Valve's Steam Survey
- Linux 6.16 Will Begin Reporting The Cause Of Your AMD Zen System Being Reset/Rebooted
- Redis 8.0 Released: Now Tri-Licensed With AGPLv3
- Kcompressd Proposed For Accelerated Memory Compression On Linux
- Intel's Vulkan Linux Driver Adds Memory Pool Support For Some Massive Performance Gains

Popular In The Forums

- NVIDIA Encouraging CUDA Users To Upgrade From Maxwell / Pascal / Volta
  49 Comments
- Full Disk Encryption Performance With Ubuntu 25.04 + Framework Laptop 13 Strix Point
  44 Comments

Phoronix



Axion vs Xeon vs T2A

VM type	uArch / ISA	Hourly price (us-central1)	CoreMark (4 vCPU)	CoreMark per $ (higher = better)	$/perf index*
N2-standard-4	Intel Ice Lake (x86)	**$0.1942**	**66 884**	344 k	**1.0 ×**
T2A-standard-4	Ampere Altra (Arm)	$0.1540	94 096	612 k	1.78 ×
C4-standard-4	Intel Emerald Rapids (x86)	$0.1977	≈131 000 **(~1.95 × N2)** †	663 k	1.92 ×
C4A-standard-4	Google Axion V2 (Arm)	$0.1796	≈183 000 **(~2.7 × N2)** †	**1 021 k**	**2.96 ×**

N2 vs Axion vs Xeon vs T2A

FinOps & Sustainability

Has x86 lost the data center battle? Arm claims victory as it declares close to 50 percent of compute shipped to top hyperscalers in 2025 will be Arm-based

News By Wayne Williams published April 5, 2025

AI demand is behind the Neoverse architecture's meteoric rise

Comments (1)

When you purchase through links on our site, we may earn an affiliate commission. Here's how it works.

(Image credit: Shutterstock / gguy)

- Arm says Neoverse powers nearly half of 2025 hyperscaler compute shipments
- AI growth makes power-efficiency essential for hyperscale data centers
- AWS, Google, Microsoft are leading shift from x86 to Arm infrastructure

SIMPSONZ

ARM processors typically offer a higher number of CPU cores per socket compared to traditional x86 processors.

ARM processors also work efficiently in multi-socket environments. This allows a single server to support multiple sockets, each of which can host many ARM cores.

With more cores per socket, cloud providers can **allocate more virtual machines (VMs) or containers on each physical server**. This allows them to maximize the utilization of hardware, ultimately leading to lower per-instance costs for customers.

More CPU Cores per Socket

ARM licenses its architecture rather than selling chips directly.

This **licensing model** tends to be cheaper for cloud providers when compared to **x86** processors, which often have higher upfront costs due to both **chip pricing** and **licensing fees** from companies like Intel and AMD.

As cloud providers like AWS, Google Cloud, and Microsoft Azure continue to expand their ARM-based offerings (e.g., AWS Graviton), they gain the **ability to scale up ARM chip production to achieve economies of scale**.

Lower Total Cost of Ownership

ARM processors are highly customizable, and cloud providers can design their own specialized ARM chips (e.g., AWS Graviton) optimized for specific workloads like web services, databases, or machine learning.

By tailoring ARM chips to specific use cases, cloud providers avoid over-provisioning resources and **reduce unnecessary costs**, which they can pass on to users through more affordable pricing.

ARM’s Customization and Optimizations

ARM cores, such as those in the Neoverse family, are designed with a strong focus on **performance-per-watt**. They deliver impressive computing power while consuming significantly less power compared to x86 processors.

Since cloud providers charge for power usage (especially in large-scale data centers), the lower energy consumption of ARM chips means that **data center operational costs** are reduced, which translates to **cheaper pricing** for customers.

More Compute Power per Watt

	Produkt	Wycena za dzień	Wycena za miesiąc	Uwagi / linki
2	**Web**			
	A100 40GB (1 year commitment price)	$226,64	$6 893,63	Wymagania: PCIe 5.0, 94GB, HBM, 456 Tensor, 14592 CUDA, 51 TFLOP a2-highgpu-4g (1xCPU + 4xGPU) 48vCPU, 340GB RAM, 1GPU, 40GB HBM2, 432 Tensor, 6912 CUDA, 156 TF32 Tensor Core, PCI Gen4 GPU w sumie: 4 GPU, 160 GB, 1728 Tensor, 27648 CUDA, 624 TFLOPS Nie spełania PCIe https://www.pny.com/nvidia-a100
18	A100 40GB (3 year commitment price)	$125,95	$3 830,89	Wymagania: PCIe 5.0, 94GB, HBM, 456 Tensor, 14592 CUDA, 51 TFLOP a2-highgpu-4g (1xCPU + 4xGPU) 48vCPU, 340GB RAM, 1GPU, 40GB HBM2, 432 Tensor, 6912 CUDA, 156 TF32 Tensor Core, PCI Gen4 GPU w sumie: 4 GPU, 160 GB, 1728 Tensor, 27648 CUDA, 624 TFLOPS Nie spełania PCIe https://www.pny.com/nvidia-a100
19	V100 GPU	$531,84	$16 176,80	Wymagania: PCIe 5.0, 94GB, HBM, 456 Tensor, 14592 CUDA, 51 TFLOP 2x n1-standard-16 (16 vCPU, 60GB) 2x 4x nvidia-tesla-v100 (1GPU, 16GB HBM2, 640 Tensor, 5120 CUDA, 112 TFLOPS, PCI Gen3) CPU w sumie: 32 vCPU, 120GB GPU w sumie: 8 GPU, 128 GB, 5192 Tensor, 40960 CUDA, 896 TFLOPS Nie spełania PCIe
20	P100 GPU	$322,56	$9 811,20	Wymagania: PCIe 5.0, 94GB, HBM, 456 Tensor, 14592 CUDA, 51 TFLOP 2x n1-standard-16 (16 vCPU, 60GB) 2x 4x nvidia-tesla-p100 (1GPU, 16GB HBM2, BRAK Tensor, 3584 CUDA, 9,3 TFLOPS, PCI Gen3) CPU w sumie: 32 vCPU, 120GB, GPU w sumie: 8 GPU, 128 GB, BRAK Tensor, 28672 CUDA, 74,4 TFLOPS Nie spełania PCIe, brak Tensor cores https://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-PCIe-datasheet.pdf

List price is the start of the discussion - it is good to asses options

Alphabet says Spotify, Paramount sign to use new cloud chip

By **Stephen Nellis**

October 30, 2024 5:03 PM GMT+1 · Updated 6 months ago

A screen displays the logo of Spotify on the floor at the New York Stock Exchange (NYSE) in New York City, U.S., December 4, 2023. REUTERS/Brendan McDermid/File Photo Purchase Licensing Rights

Oct 30 (Reuters) - Google Cloud, a unit of Alphabet Inc (GOOGL.O), said on Wednesday that Spotify Technology (SPOT.N) and Paramount Global (PARA.O) are using a new chip that it designed with technology from Arm Holdings to power their streaming services.

Three Mile Island nuclear plant will reopen to power Microsoft data centers

SEPTEMBER 20, 2024 · 1:40 PM ET

By C Mandler

Power Efficiency of Java on Arm-based server

Advantages of Arm-based servers in power consumption with Java application

Verification – impact on CO_2 emission reduction

The potential CO_2 emission reduction over 3 years by replacing x86 servers by Arm-based servers was estimated.

Assumptions

- One rack can hold 21 x86 servers. We assume those 21 x86 servers replace to Arm servers.
- Servers operate 24 hours a day, 365 days a year.
- Electricity charges and CO_2 emission factors are as follows.
  - Electricity charge (JPY/kWh) :
    Referred by TEPCO's rate averaging the difference from seasons and time zone of a day (See Reference #2)
  - CO_2 Emission Factor (t-CO_2/kWh):
    Referred by CO_2 Emission coefficient which TEPCO provides on their web (See Reference #3).

Results

Replacing x86 servers with Arm servers can reduce annual CO_2 emissions by about 10t-CO_2 and reduce CO_2 emissions by about 30t-CO_2 after 3 years of use.

CO_2 Emission per Rack (kg-CO_2/Rack)			
Year	x86 Server	Arm Server	Savings
1yr	26,350	16,601	9,750
2yr	52,700	33,201	19,499
3yr	79,050	49,802	29,249

Conclusion

The word "green application" first comes to mind about power saving by improving processing efficiency in terms of software such as algorithms. However, the program is ultimately executed on hardware. So both software and hardware should be considered to avoid local power saving. Arm server will make a solid contribution to hardware power savings. From the results of NTT DATA's verification, the use cases for Arm server can be expanded not only to HPC but also to the enterprise domain. Ampere Altra® and Altra® Max processors used in the RL300 are specifically designed to improve performance and efficiency for cloud workloads, but given the findings, the RL300 will be a critical building block to improve performance and reduce carbon footprint of IaaS, PaaS, and SaaS offerings. In terms of combining performance and power saving, Arm servers are to be used in a wider range of areas.

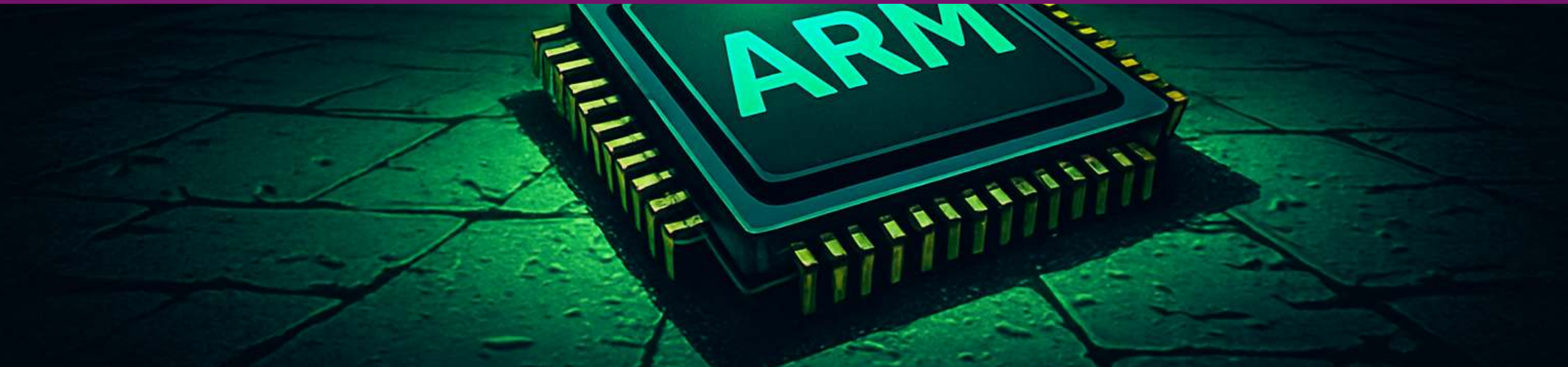
https://www.nttdata.com/global/ja/-/media/nttdataglobal-ja/files/news/topics/2023/112400/112400-01.pdf

Networking - VPN ingress bytes	392435047
Billing - has billing account	Yes
Billing - Spend during last 30 days	$342
Sustainability - est. CO_2 emissions during last 30 days	2 kg CO2
Daily avg. usage - App Engine instances	0
Daily avg. usage - Cloud SQL instances	0
Daily avg. usage - BigQuery jobs	0
Daily avg. usage - BigQuery queries	0
Daily avg. usage - Storage requests	2663548
User activity - Number of active owners	1

Recommendation

Consider deleting this project to remediate 250 high severity security risks, save est. $200/month, and reduce your carbon footprint by est. 4 kgCO_2e/month.

Highlights



Since JEP 237 merely “ran” JDK on AArch64, HotSpot has gained intrinsics, Gen-ZGC, Vector API with SVE/SVE2 support, and virtual threads.

The JVM has caught up to the hardware.

Since JEP 237 merely “ran” JDK on AArch64, HotSpot has gained intrinsics, Gen-ZGC, Vector API with SVE/SVE2 support, and virtual threads.

Now Java not only runs on ARM, but fully leverages wide registers, deep buffers, and many true cores without SMT traps...

The JVM has caught up to the hardware.

Since JEP 237 merely “ran” JDK on AArch64, HotSpot has gained intrinsics, Gen-ZGC, Vector API with SVE/SVE2 support, and virtual threads.

Now Java not only runs on ARM, but fully leverages wide registers, deep buffers, and many true cores without SMT traps...

....and with GitHub and CircleCI arm64 runners plus Docker buildx, the build-once-run-everywhere pipeline truly works - no cross-compile acrobatics (excluding JNI).

The JVM has caught up to the hardware.

- JDK itself is ready right now, but...
- Always use a modern OpenJDK (Java 11.09 or newer). Newer JDKs include Arm-specific optimizations for the latest Arm platforms, such as AWS Graviton 4.
- GraalVM is ready too! (Unless you use Windows ARM64)
- More modern ARM CPU - better results (Neoverse makes a real difference)
- If your services are I/O-heavy, highly parallel, or licensed per vCPU (physical cores mean less contention and better economics) - that is very good choice.
- If FinOps and ESG are in your KPIs: smaller clusters, smaller bills, smaller CO_2 footprints.
- Maybe, if your project uses JNI libraries - they'll need recompiling or removal
- Not always, if single-thread CPU performance or exotic AVX-512 instructions matter - there x86 may still win...
- ...however Neoverse architecture is very performant

Highlights

arm Products Markets Partners Developers Support & Training Company

Works on arm

An Initiative to Streamline Software Development on Arm with Free Cloud Resources

The Works on Arm initiative enables developers to build, test, and optimize projects on the Arm64 architecture by providing free-of-cost access to Arm based developer platforms, cloud instances, and CI/CD environments. Arm has partnered with cloud platform providers to make Neoverse-based compute available for cloud-to-edge solutions, while offering choice and flexibility.

Developer Offerings:

worksonarm.com webpage

"Optimizing the GC when Migrating Cloud Workloads to Arm" with Kieran Hejmadi – JVM Weekly vol. 128

Today, we have a guest post from Kieran Hejmadi, an Software and Academic Ecosystem Manager in Arm.

ARTUR SKOWRONSKI AND KIERAN HEJMADI

MAY 01, 2025

4 Share

Why Java on ARM is gaining momentum

Artur Skowroński
Head of Java & Kotlin Engineering

PUBLISHED: MAY 7, 2025 | 7 MINUTES READ

The evolution of ARM: from Acorn to the cloud

ARM's story starts in a wooden shed in Cambridge, 1985, where engineers built the first chip - the ARM1 - for a line of educational computers for BBC. The goal was clear: keep it simple, keep it efficient.

Short pipelines (with fewer processing stages between fetching and executing each instruction), simpler architecture with fewer transistors (requiring less power and chip area), and a RISC (Reduced Instruction Set Computing - a small, uniform set of simple commands) set the tone early. That lean design helped ARM take over the mobile world (one of the first devices powered by it was Apple Newton - a smartphone from 1992) - and it didn't stop there. Today, ARM cores show up almost everywhere, with over 300 billion sold and still climbing.

bitly

VIRTUSLAB

Thank you 🙏

@ArturSkowronski

VIRTUSLAB

jvm weekly