AWS Graviton Essentials

Processor Architecture & Features, Performance, Migration and Use Cases

SoonBeom Kwon
Efficient Compute Specialist Solutions Architect
Amazon Web Services

Agenda

ARM Intro

Into the AWS Graviton

Performance Benchmark

Migration Strategy

Performance Profiling / Tunning

Customer Case Study

Appendix

ARM Intro

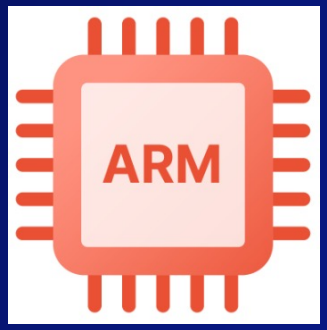
What is ARM ?

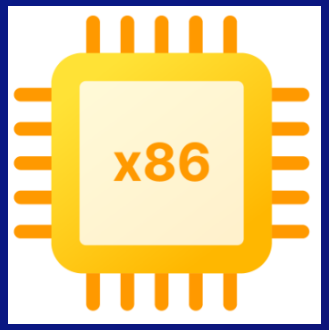
- ARM stands for Advanced RISC Machine.
- A Reduced Instruction Set Computer (RISC) was developed by a company called Acorn Computers Ltd.
- ARM is the world leader company in IP offerings consisting of a world range of microprocess cores, Architectural extensions, development tools, peripheral IP and SoC Solutions. All of these solutions are supported by ARM and a global network of design and engineering partners.

The history of ARM

- Arm was officially founded as a company in November 1990 as Advanced RISC Machines Ltd, which was a joint venture between Acorn Computers, Apple Computer (now Apple Inc.), and VLSI Technology (now NXP Semiconductors N.V). The company was founded by 12 Arm architecture designers.
- Acorn, developer of the world's first commercial single-chip RISC processor and Apple intend on advancing the use of RISC technology in its own systems, chartered ARM with creating a new microprocessor standard.
- With the introduction of its first embedded RISC core, the ARM6 family of processors, in 1991, ARM signed with VLSI and sharp as its initial license.

ARM vs X86 Instructions





RISC (Reduced Instruction Set Computing)

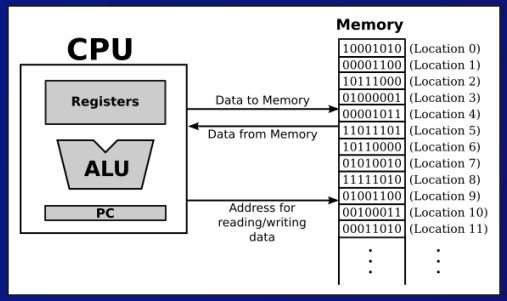
- a small, highly optimized instructions
- around 354 base instructions
- generally fixed-length (typically 32 bits or 16 bits in Thumb mode)
- save energy and makes instructions fast and easy

CISC (Complex Instruction Set Computing)

- a large set of complex instructions
- approximately 981 base instructions
- variable lengths (1 to 15 bytes)
- a lot of tasks at once but makes the processor more complicated and expensive

RISC emphasizes efficiency by taking into account cycles per instructions whereas CISC emphasizes efficiency by the number of instructions in a program.

ARM vs X86 Registers



ARM

- **Number of Registers:** 31 general-purpose registers in the 64-bit
- **Register Size:** All registers are 64-bit.
- **Register Usage:** Primarily used for storing data, addresses, and operands, with some registers having specific roles like the LR for function return addresses.
- **Memory Access:** Uses a load/store architecture, meaning data must be explicitly loaded into registers from memory and stored back.
- **Flexibility:** Registers have specific purposes and names, and their use is more rigidly defined.

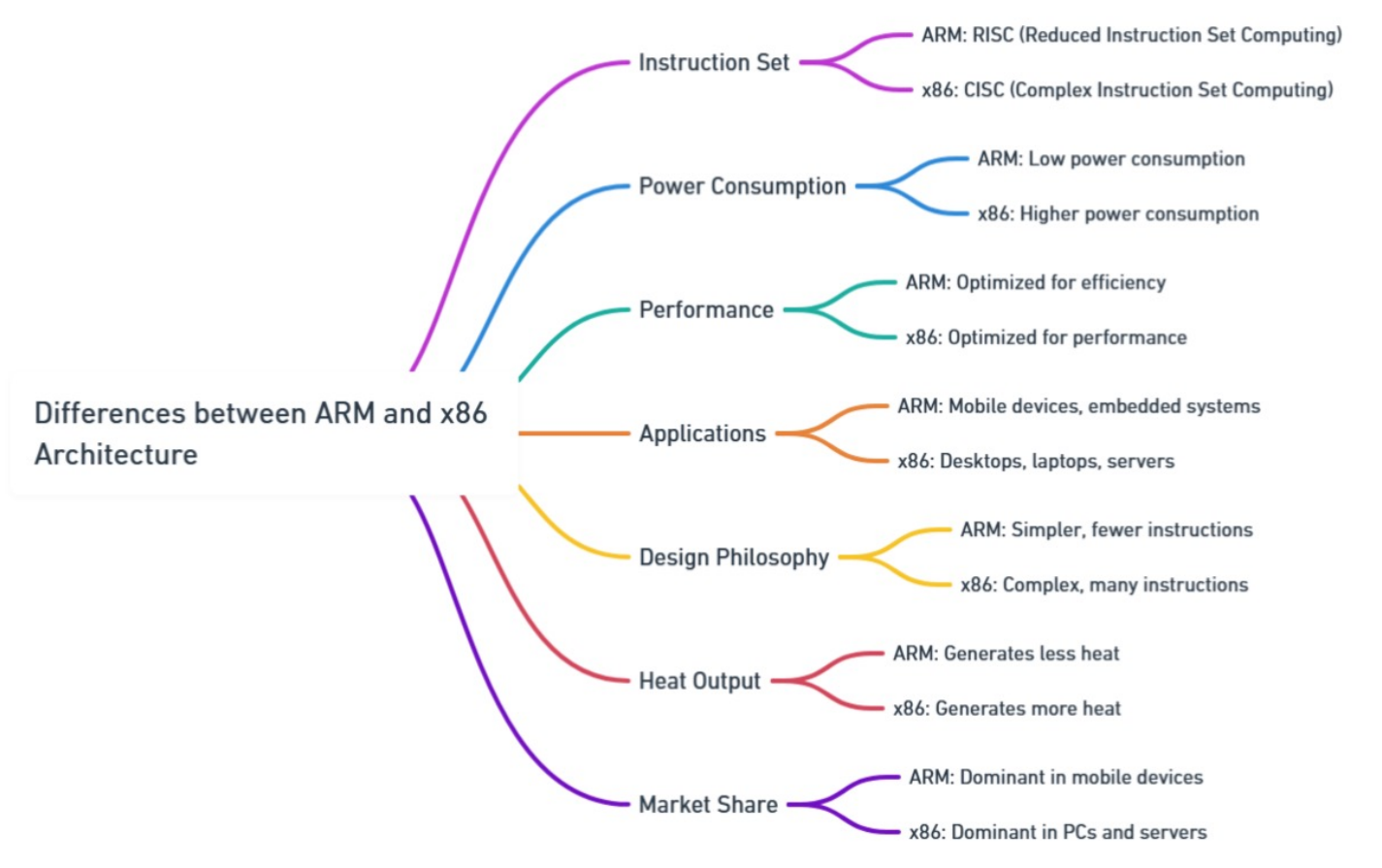
x0 to x30 64 bits general register w0 to w30 32bits general register

X86

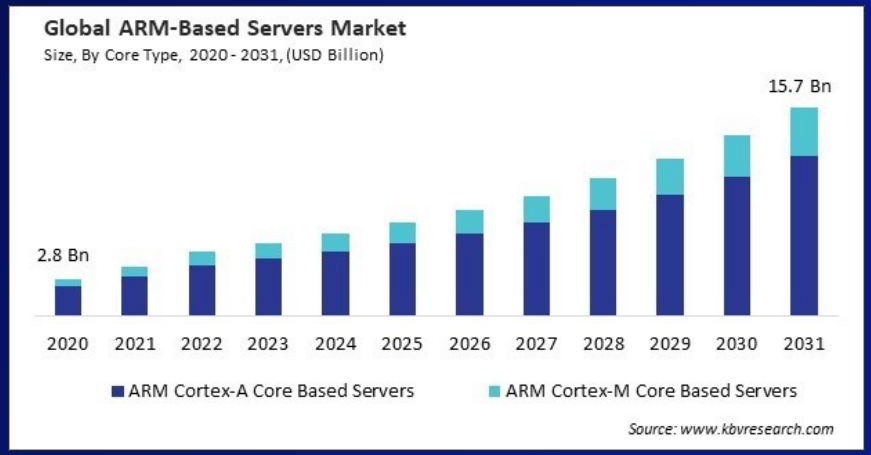
- **Number of Registers:** 16 general-purpose registers in the 64-bit
- **Register Size:** Registers are 64-bit, but some can be treated as smaller quantities (16-bit or 8-bit).
- **Register Usage:** Primarily used for calculations, address calculations, and storing operands.
- **Memory Access:** Allows for more direct memory interaction and operations directly on memory.
- **Flexibility:** Registers can be used for various purposes, and the architecture offers more complex instructions that can directly manipulate memory.

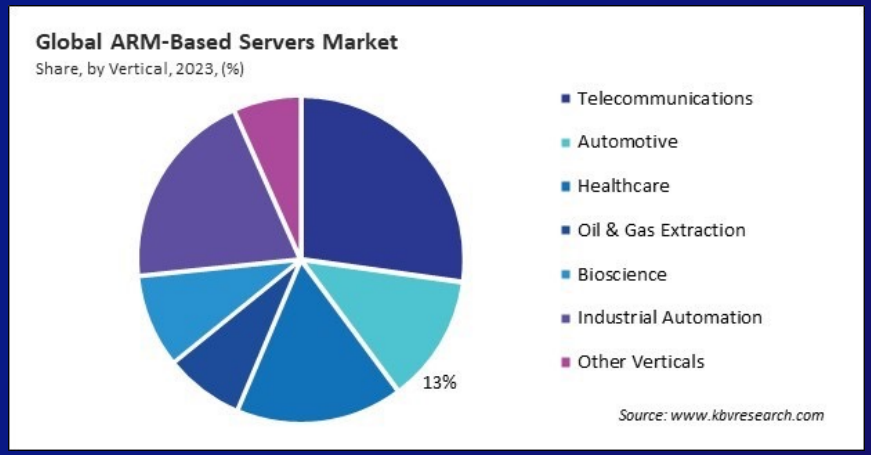
32 bits : EAX EBX ECX EDX 16 bits : AX BX CX DX 8 bits : AH AL BH BL CH CL DH DL

ARM vs X86 Key Differences



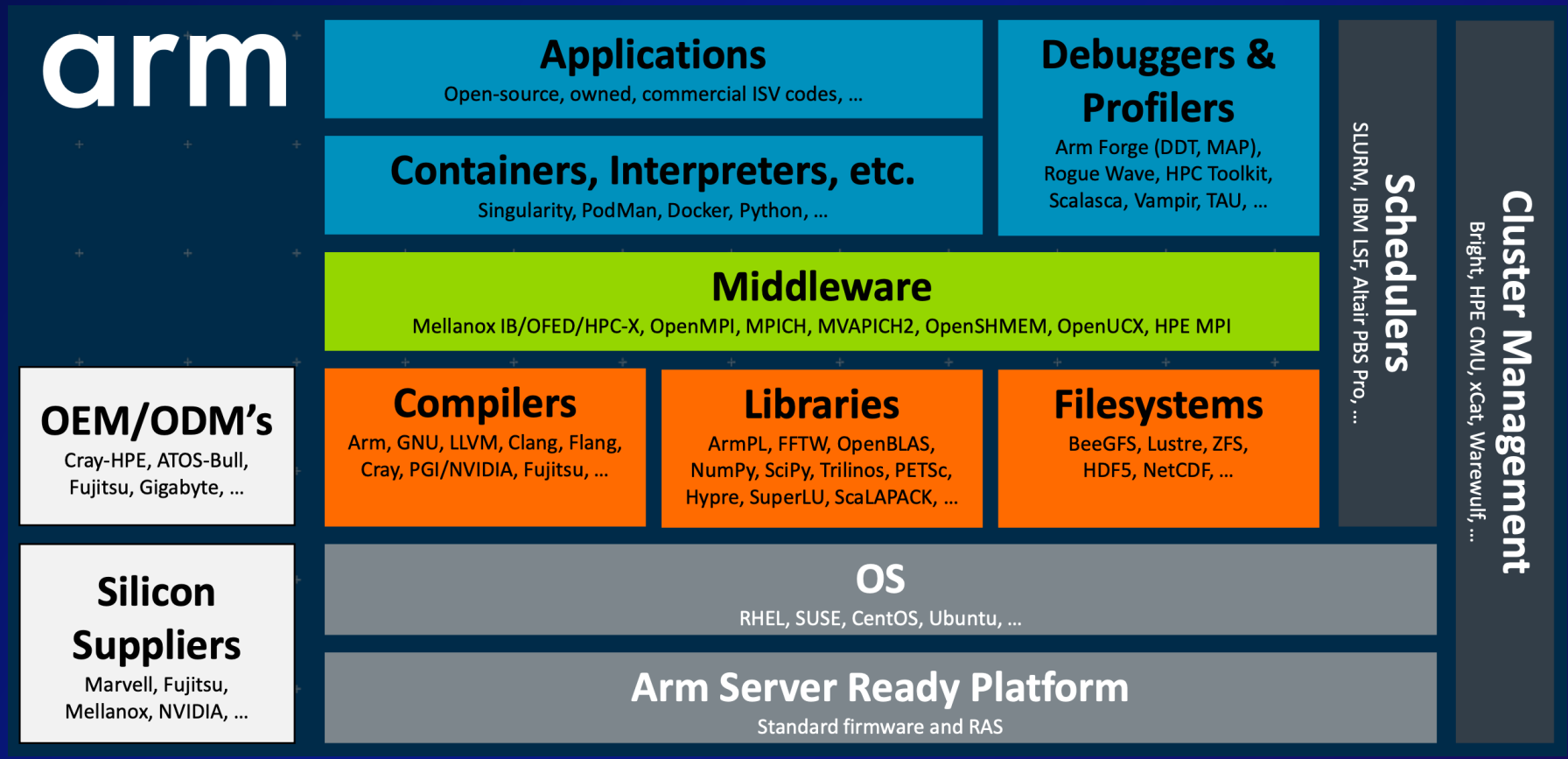
ARM Market Share





In 2024, Arm Holdings held a significant market share in various technology sectors. Specifically, Arm dominated the mobile applications market with nearly 100% share and a 54% share in the Internet of Things and embedded computing markets globally according to Statista. In the data center CPU market, Arm's market share was approximately 15%, but projections for 2025 indicate a surge to 50%, driven by the increasing demand for AI applications. Arm also has a noticeable presence in the PC market, with reports suggesting it reached 10.9% of the PC market share by the end of 2024, and is expected to grow further in the coming months.

ARM Software Echosystem



ARM Software Echosystem

arm

A Rich and Growing Application Ecosystem

GROMACS	LAMMPS	CESM2	MrBayes	Bowtie	DeepBench
NAMD	TensorFlow	ParaView	SIESTA	UM	AMBER
WRF	Quantum ESPRESSO	VASP	Torch	MILC	GEANT4
OpenFOAM	GAMESS	Mahout	VisIt	DL-Poly	NEMO
Weka	BLAST	NWCHEM	Abinit	BWA	QMCPACK
Chem/Phys	*Weather*	*CFD*	*Visualization*	*Genomics*	*AI/ML*

GNU and LLVM Toolchains

GNU Toolchain (compilers, debuggers, libraries, etc.)

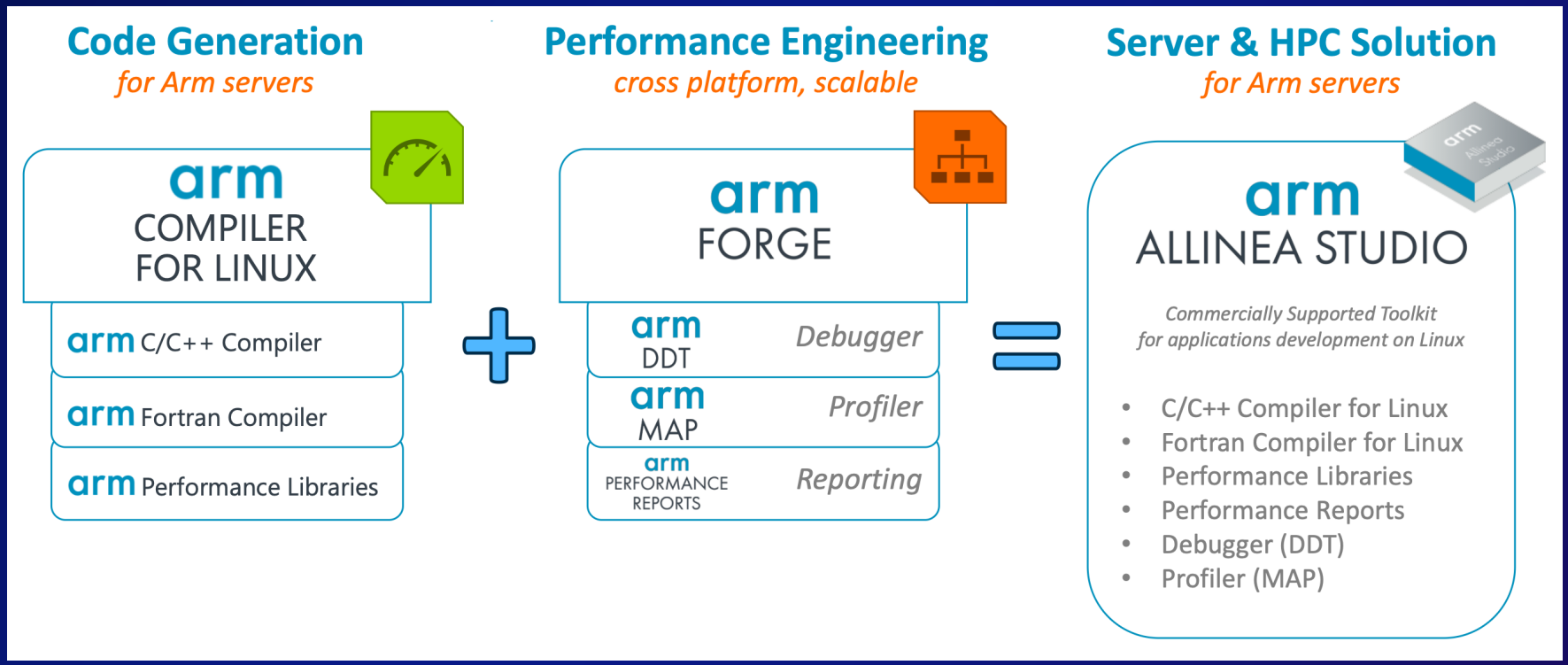
- Default compiler in Linux distributions like RedHat, SUSE, Ubuntu
- Key segments: Cloud, networking and HPC

LLVM Toolchain (compilers, debuggers, libraries, etc.)

- Default compiler in Android and the basis for commercial compilers (including Arm and Cray compilers)
- Key segments: Mobile (Android/iOS), Cloud

Server & HPC Development Solutions from Arm

Commercially supported tools for Linux and high performance computing

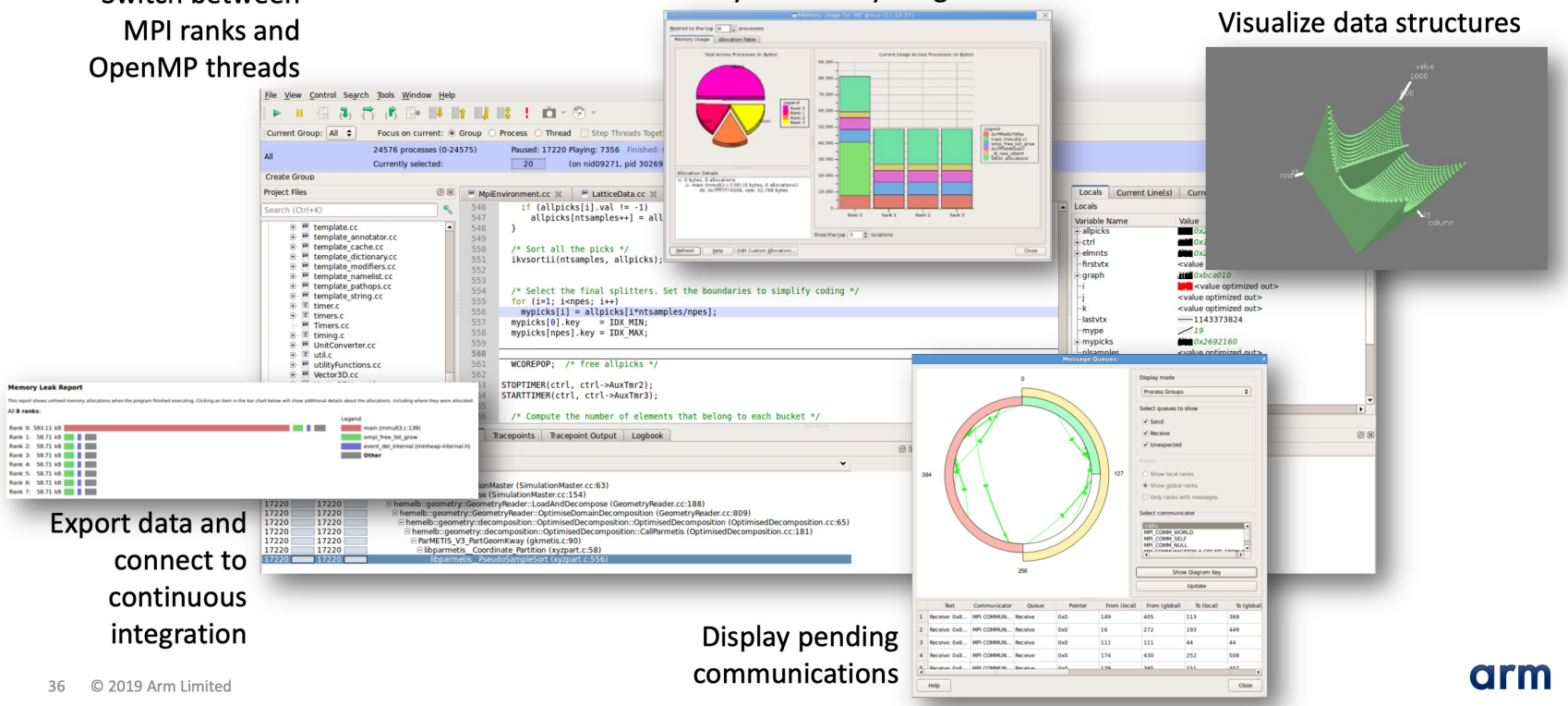


Arm Forge – DDT Parallel Debugger

Switch between MPI ranks and OpenMP threads

Analyze memory usage

Visualize data structures



Export data and connect to continuous integration

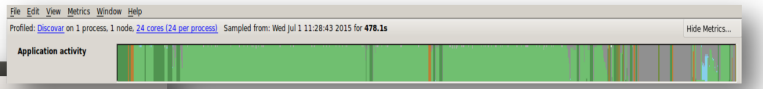
Display pending communications

arm

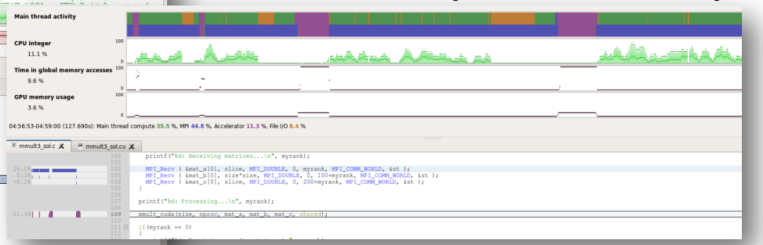
Arm Forge – MAP Multi-node Low-overhead Profiler

Understand **MPI**/**CPU**/**IO** operations thanks to timelines and metrics

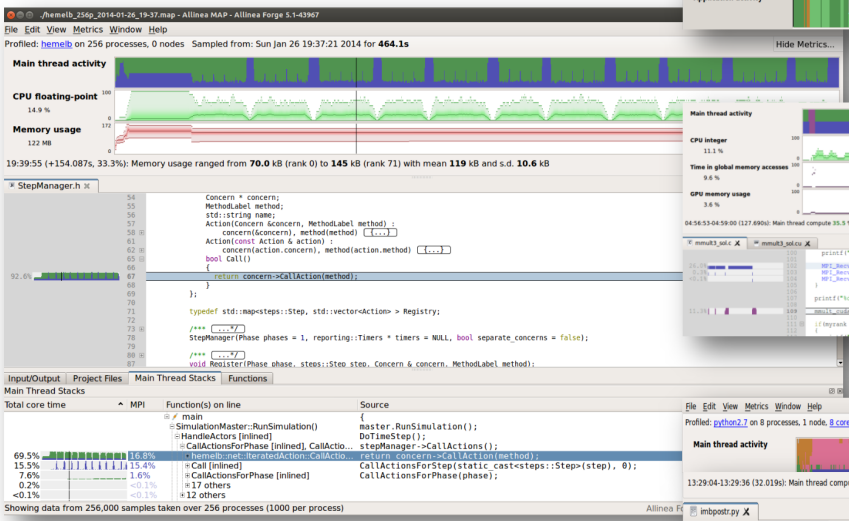
Inspect **OpenMP** activity



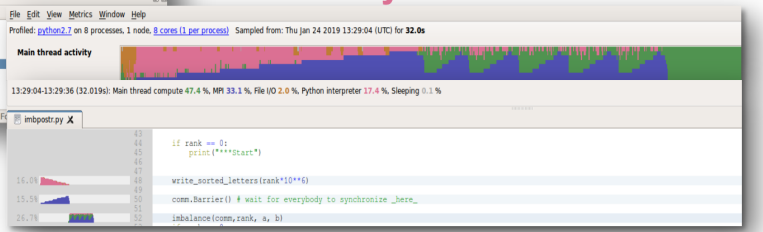
Analyze **GPU** efficiency



Investigate annotated source code and stack view



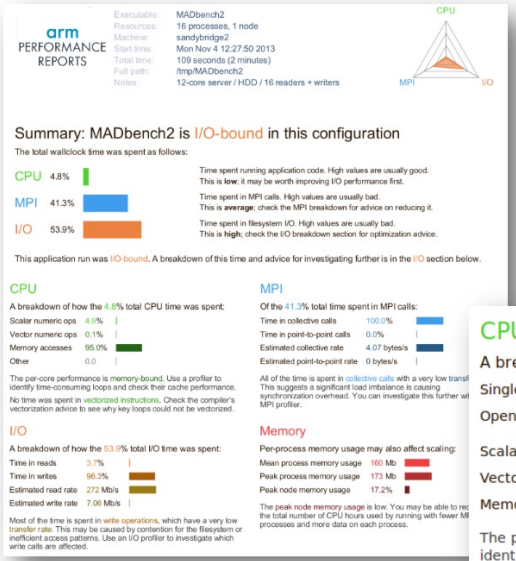
Profile **Python**-based workloads



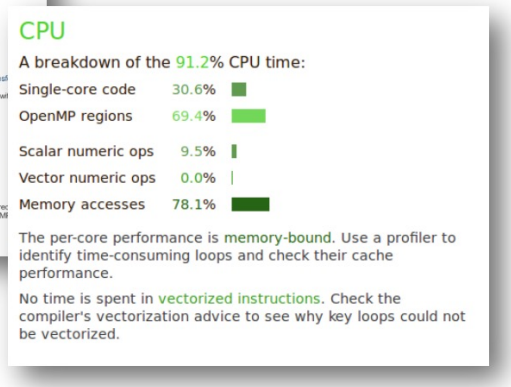
arm

Arm Performance Reports Application Analysis Tool

Analyze all performance aspects in a single HTML or TXT file



Qualify the type of workload



Inspect key metrics on SIMD, multithreading, IO, MPI efficiency and many more…

Follow guidance advices for your next steps and maximize output

arm

Into the AWS Graviton

Broadest choice of processors

Intel® Xeon Scalable processors

AMD EPYC processors

Apple processors

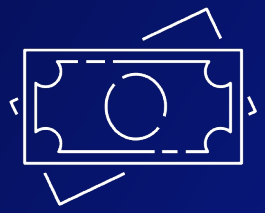
AWS Graviton processors

x86(64)

arm64

Why AWS Graviton ?

Best price-performance in Amazon EC2 for a broad array of workloads



Costs up to 20% less than comparable EC2 instances*

Uses up to 60% less energy than comparable EC2 instances

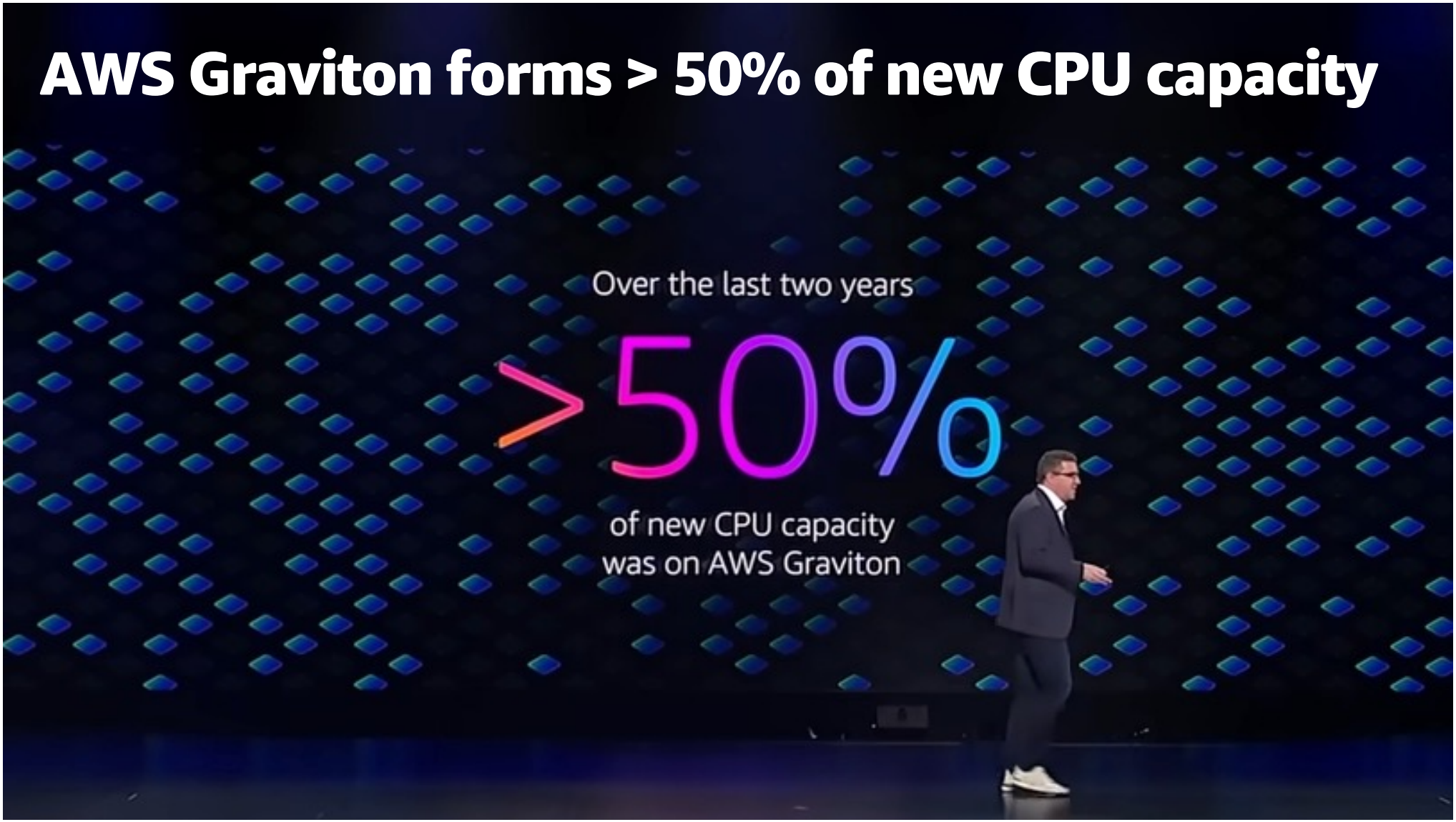
* Based on public on-demand pricing

AWS Graviton Adoption – 98 out of 100 Top Customers

"Today, more than 50,000 customers use AWS' Graviton chips in AWS compute instances, including 98 of our top 100, Amazon EC2 customers. And these chips have about 40 percent better price performance than other leading x86 processors."

Andy Jassy, Amazon CEO
03rd Aug'23, Amazon's second fiscal quarter 2023 financial analyst conference call

AWS Graviton forms > 50% of new CPU capacity



AWS Graviton Processors

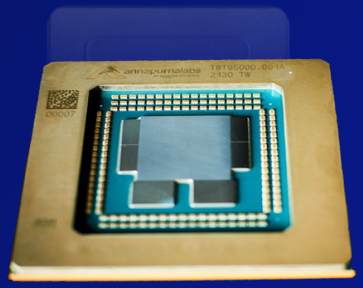
Graviton2

2019

Get up to 40% better price performance over comparable current generation x86-based instances with Graviton2
ARM Neoverse-N1 (2500MHz, 64Core)

Graviton3

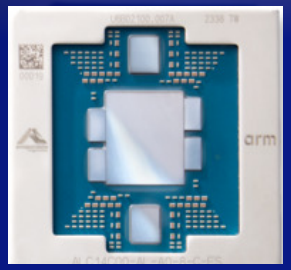
2021



Graviton3 processors provide up to 25% better compute performance than Graviton2
Up to 60% more energy efficient vs. comparable x86-based instances
ARM Neoverse-V1 (2600MHz, 64Core), SVE, bfloat16

Graviton4

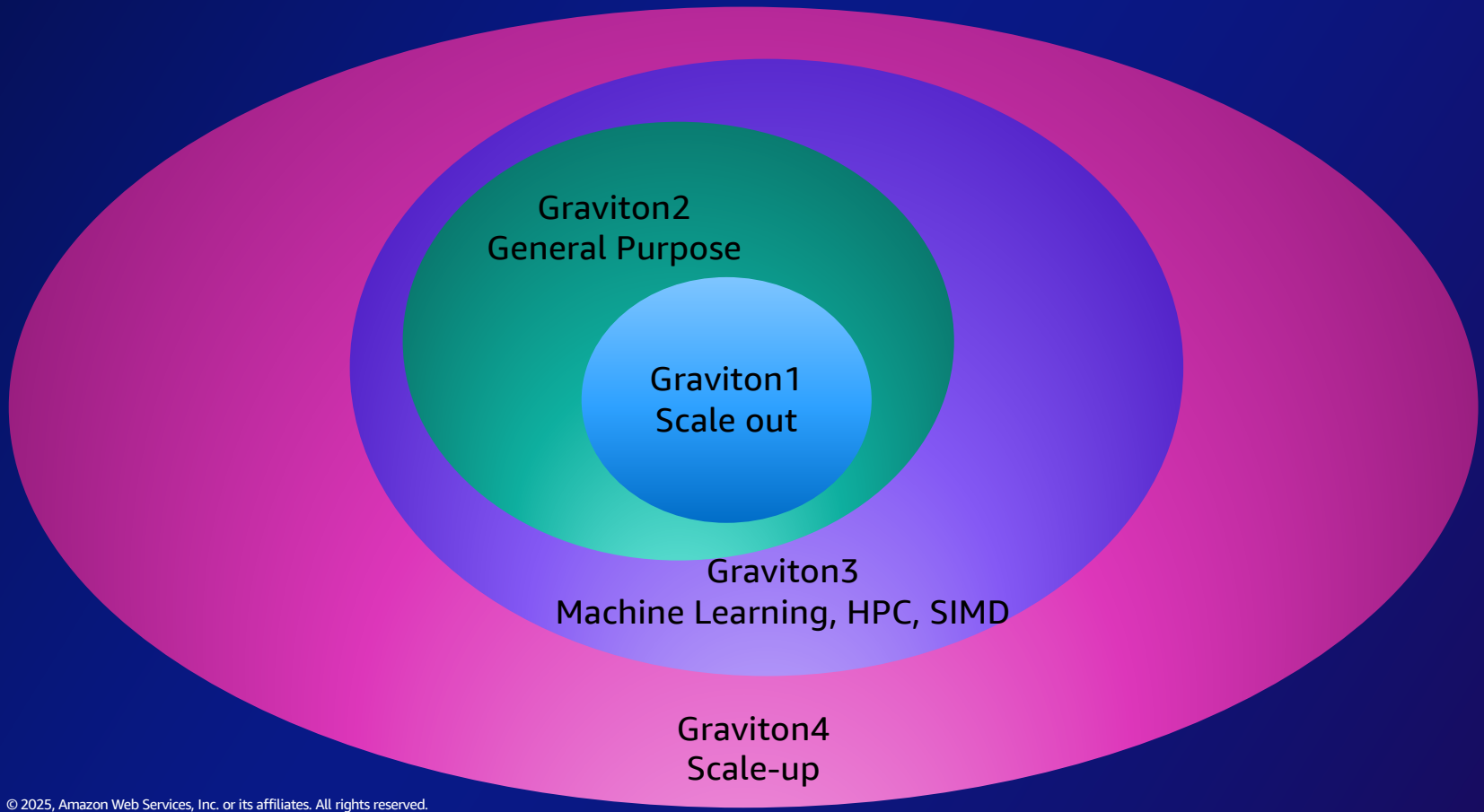
2023



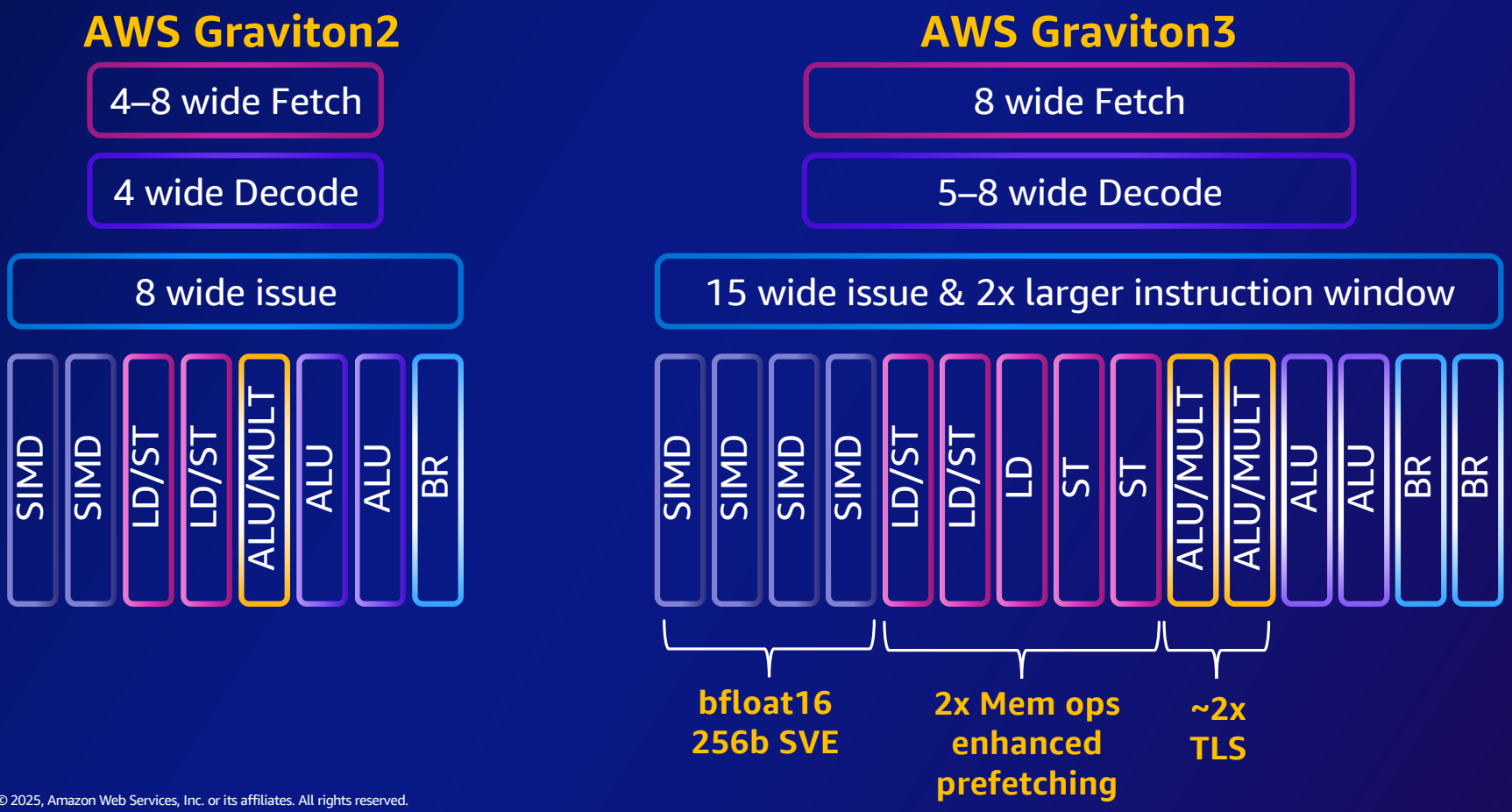
Graviton4 processors provide up to 30% better compute performance, 50% more cores than Graviton3
ARM Neoverse-V2 (2800MHz, 96Core, 192Core/2Socket for 48xlarge)

aws

Expanding workload applicability



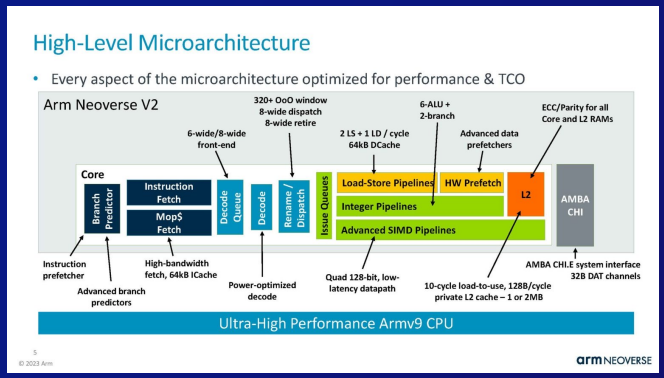
Graviton3 CPU enhancements



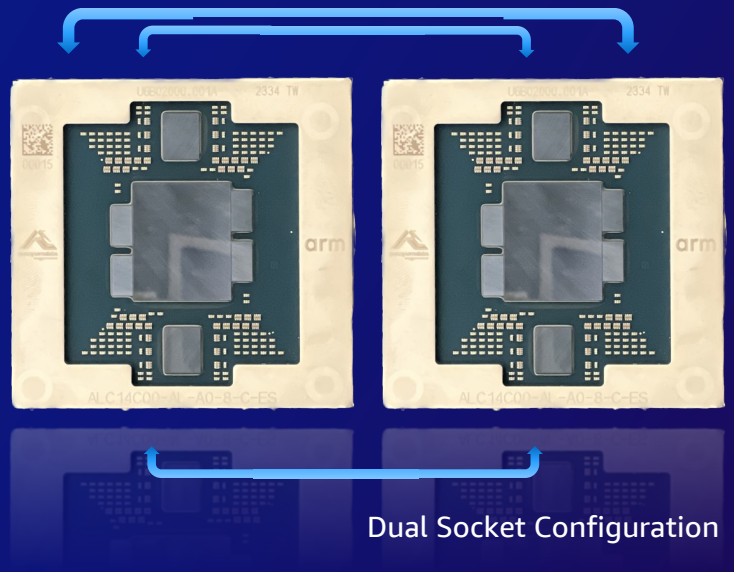
Graviton4 enhancements

SCALE-UP / EXPANDING WORKLOADS GRAVITON APPLIES TO

- 2MB L2 Cache per Core (graviton 3 - 1MB)
- LLC (L3 Cache) 36MB (graviton 3 – 32MB)
- Clock Speed 2800MHz (2700MHz for 48xlarge) (graviton 3 – 2600MHz)
- Single socket 24xl (96 vCPUs)
  - 50% more cores-per-socket than Graviton3 (64 cores)
- Support for coherent multi-socket
- Single systems with up to:
  - 192 Cores (3x more cores than Graviton3 64 vCPUs)
  - 1,536 GB (3x more DRAM than Graviton3 512 GB)



Neoverse v2 Core enhancement

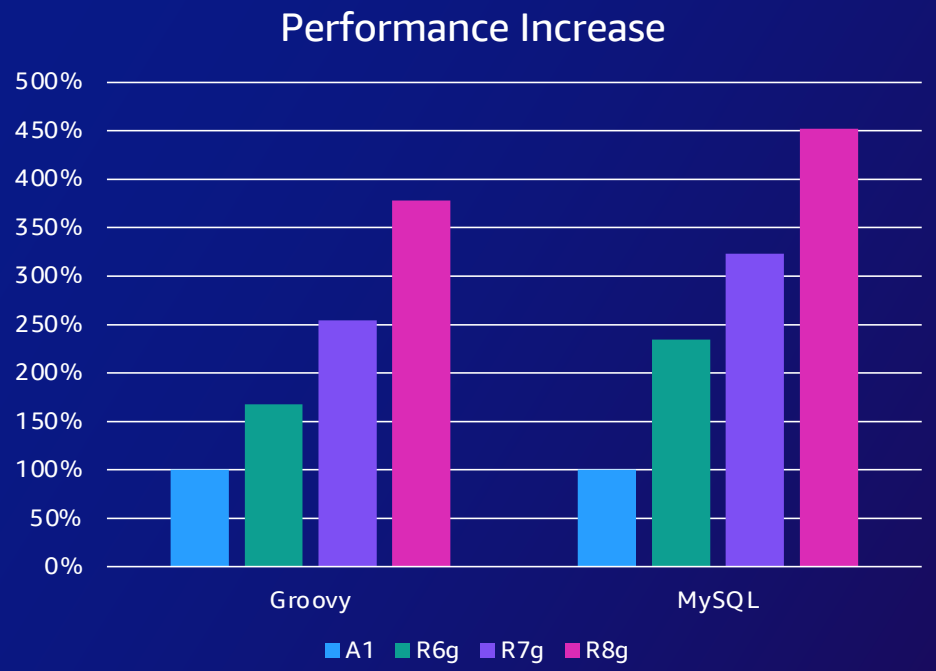


Dual Socket Configuration

Four times the performance in four generations

Leap in performance with Graviton4-based R8g

Consistently delivering big increases for 4 generations



Workloads for AWS Graviton

AWS managed services

- Amazon Aurora
- Amazon RDS
- Amazon ElastiCache
- Amazon EKS
- Amazon ECS
- Amazon EMR
- Amazon OpenSearch

Self-managed workloads on EC2

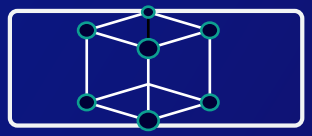
- Web + gaming servers

- Analytics

- Open source databases

- Microservices/Containers

- HPC and AI/ML



Workload characteristics

- ✓ Open source, Linux-based
- ✓ Running newer versions of software (e.g. programming language, OS, database engine)
- ✓ Written in higher-level programming languages (e.g. Java, PHP, node.JS)
- ✓ Written in compiled languages (e.g. C/C++, Python, Go)

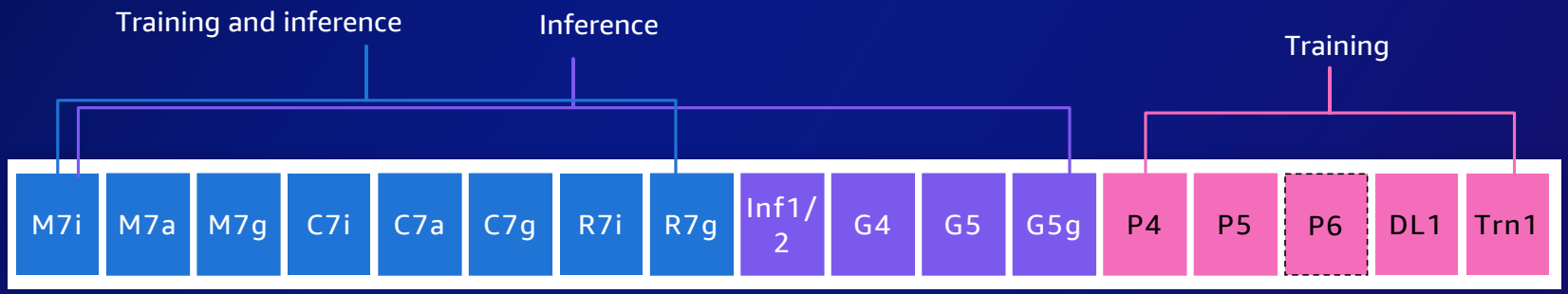
aws

Compute infrastructure for AI/ML

CPU, GPU, and Custom Accelerator EC2 Instances

Traditional machine learning (ML)

Deep learning (DL)



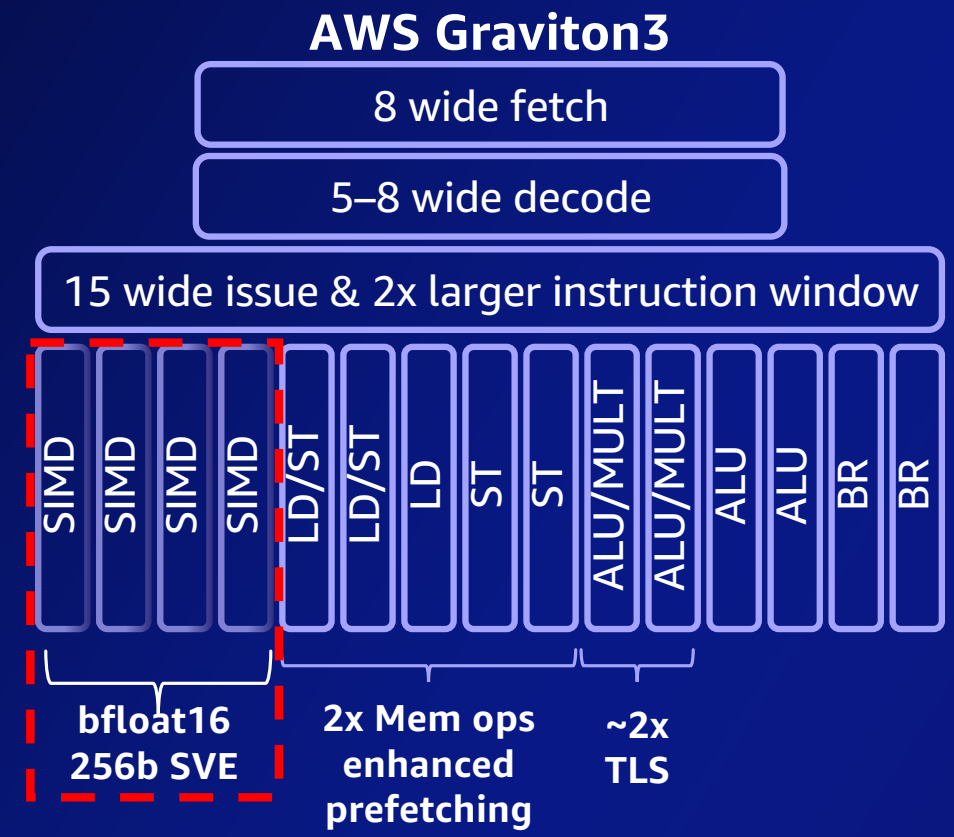
Xeon CPU
Habana Gaudi accelerator

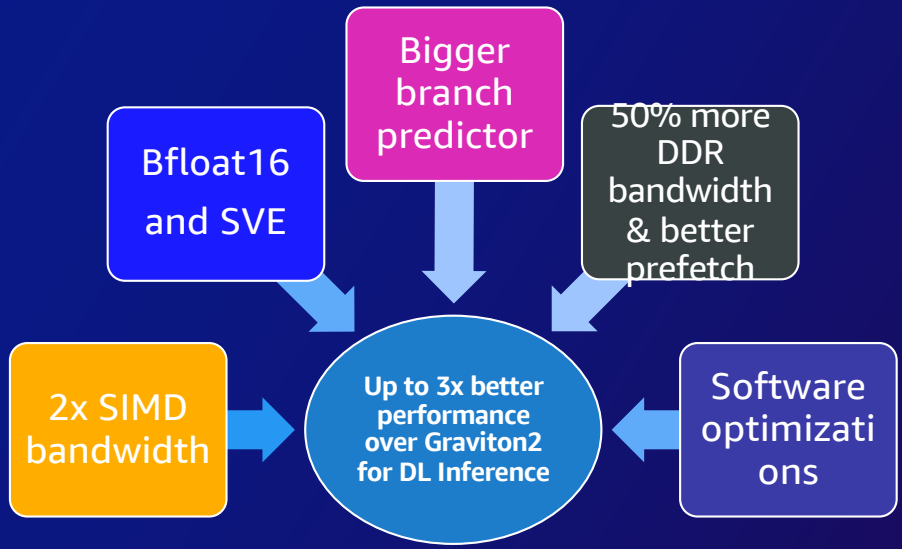
EPYC CPU
Radeon GPU

Graviton CPU
Inferentia accelerator
Trainium accelerator

A100, H100, Blackwell
T4 GPUs

Why is Graviton3 compelling for CPU AI/ML?





Machine Learning Performance

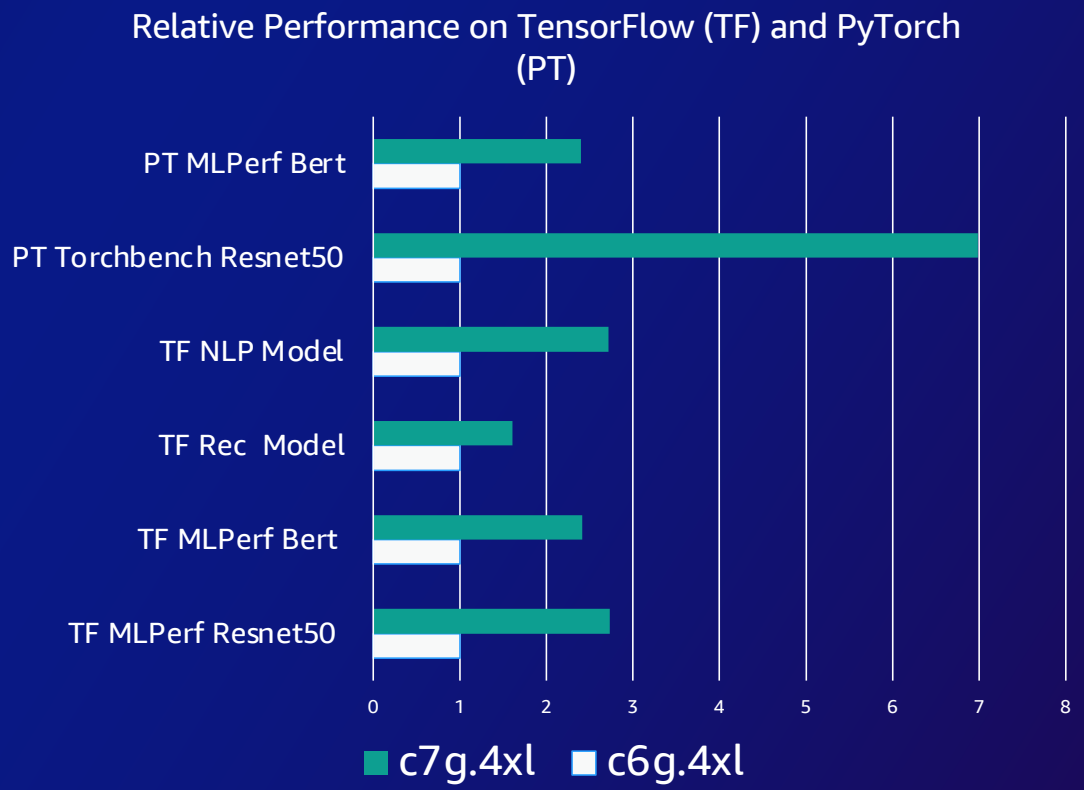
Leap in Performance with Graviton3: 2x vector width, 1.5x memory bandwidth, and bfloat16

+

Large amount of improvements in TensorFlow, PyTorch, OneDNN, and Arm Compute Library

=

Fastest CPU-based machine learning inference with TensorFlow and PyTorch across many models



When to use AWS Graviton3 for ML

ML inference

Traditional ML scenarios for classification, ranking, and clustering use cases

Deep learning for:

- Single stream real-time inference
- Natural language processing (NLP) use cases like recommendations, automatic speech recognition or speech to text, sentiment analysis, masked text predictions, etc.
- Generative AI use cases with smaller-to-medium input contexts (e.g., chatbots)
- Scenarios where ML service needs to be collocated with the existing non-ML services

ML training

- Data preparation phase
- Traditional ML models (tabular data)
  - scikit-learn
  - AutoGluon (AutoML) https://go.aws/3EkPD4k
- Sparse neural networks
- Fine-tuning (transfer learning) smaller-to-medium sized networks

EC2 Naming Explained

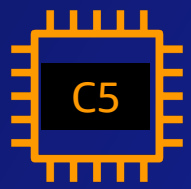
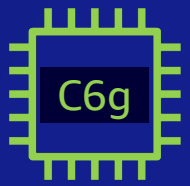
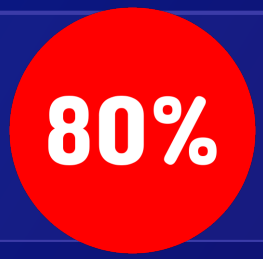
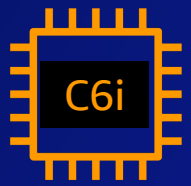
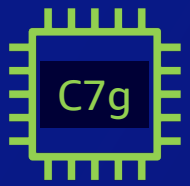
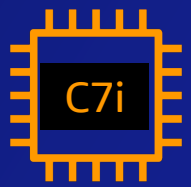
https://aws.amazon.com/ko/ec2/instance-types/

AWS Graviton EC2 instances

	Graviton2	Graviton3	Graviton3E	Graviton4
General Purpose	M6g, M6gd, T4g	M7g, M7gd		M8g
Compute Optimized	C6g, C6gd, C6gn	C7g, C7gd	C7gn, Hpc7g	C8g
Memory Optimized	R6g, R6gd, X2gd	R7g, R7gd		R8g, X8g
Storage Optimized	Im4gn, Is4gen, I4g			I8g
Accelerated Computing	G5g			

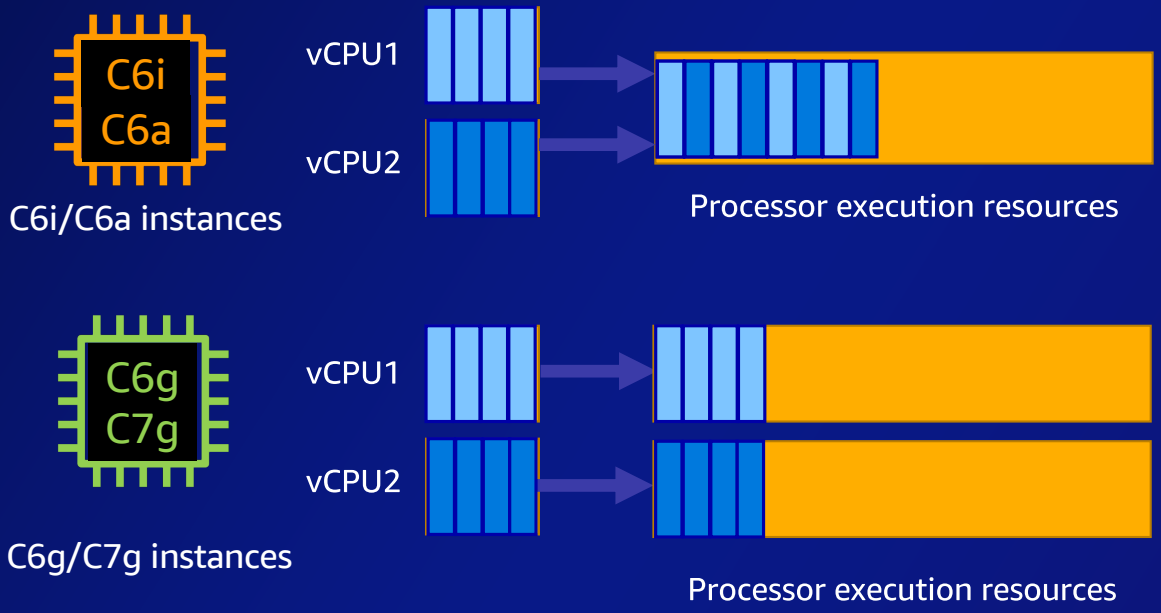
A common ratio for M instances is 1:4, meaning 1 vCPU for every 4 GiB of RAM.
C family might have a ratio closer to 1:2 or 2GB per vCPU.
R instances generally have a higher proportion of memory compared to CPU such as 1:8 or 1:16

Mapping x86 to Graviton

x86	ARM	Price Ratio(%)
intel  Skylake	 graviton2	
intel  Ice Lake	 graviton3	
intel  Sapphire Rapids	 graviton4	

Price discount ratio is on demand basis

Every vCPU is a physical core for Graviton

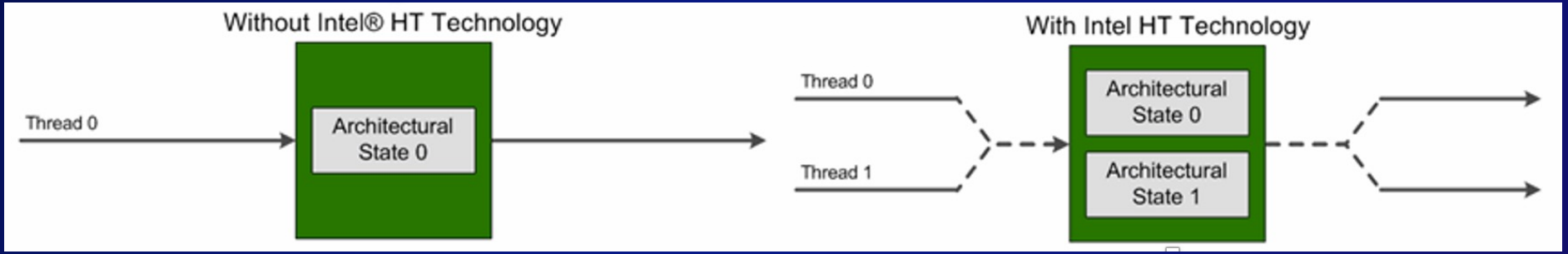


Every vCPU is a physical core

No simultaneous multithreading (SMT)

- Higher overall performance, runs well under heavier load conditions
- More consistent performance in a multi-tenancy environment

Intel Hyper Threading (SMT)

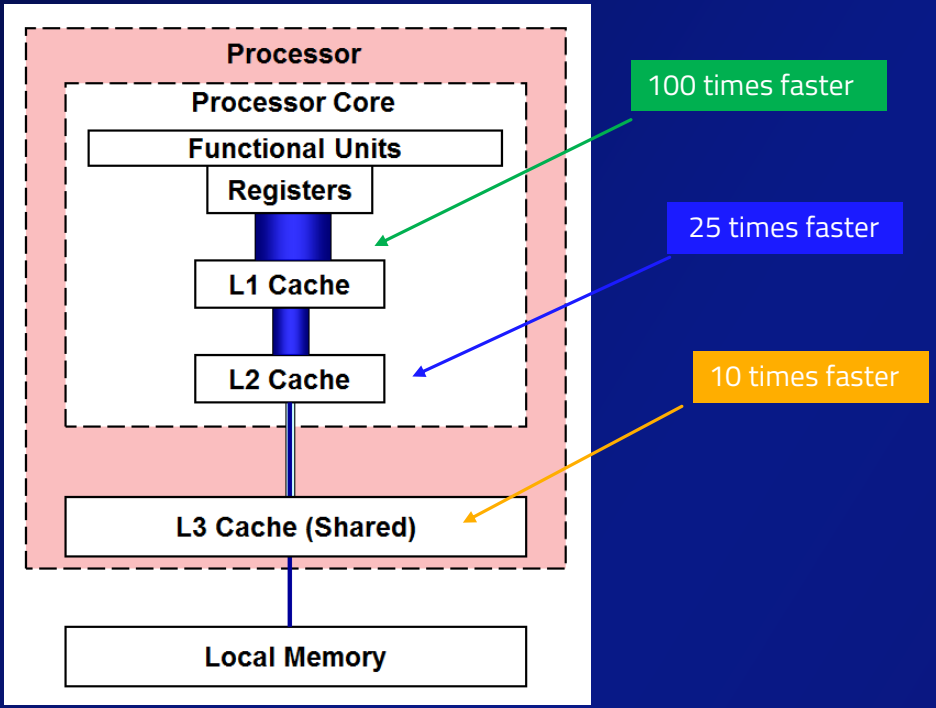


https://www.dasher.com/will-hyper-threading-improve-processing-performance/

Hyper-threading is an Intel technology which provides a second set of registers (i.e. a second architectural state) on a single physical processor core. This allows a computer's operating system or hypervisor to access two logical processors for each physical core on the system.
As threads are processed, some of the internal components of the core (called execution units) are frequently idle during each clock cycle. By enabling hyper-threading, the execution units can process instructions from two threads simultaneously, which means fewer execution units will be idle during each clock cycle.

For a single socket system, hyper-threading can boost system performance by up to 30%. For dual socket systems, hyper-threading can boost performance by up to 15%. (10% OS overhead exists)

CPU Cache



C6g
C7g

L1 – 64k(D) / 64k(I)
L2 – 1MB
LLC – 32MB

C5

L1 – 32k(D) / 32k(I)
L2 – 1MB
LLC – 35.75MB

C6i

L1 – 48k(D) / 32k(I)
L2 – 1.25MB
LLC – 54MB

C8g

L1 – 64k(D)/ 64k(I)
L2 – 2MB
LLC – 36MB

C7i

L1 – 48k(D) / 32k(I)
L2 – 2MB
LLC – 105MB

ARM focuses on L1 cache however Intel is on LLC (Last Level Cache, L3)

aws

Throughput Per Core

tested with game bot having 20 threads

DENO 2.0

Implemented by Rust

concurrent users (instance size is xlarge)

Expected Throughput

X 4 *X 2 + 30%(SMT)*

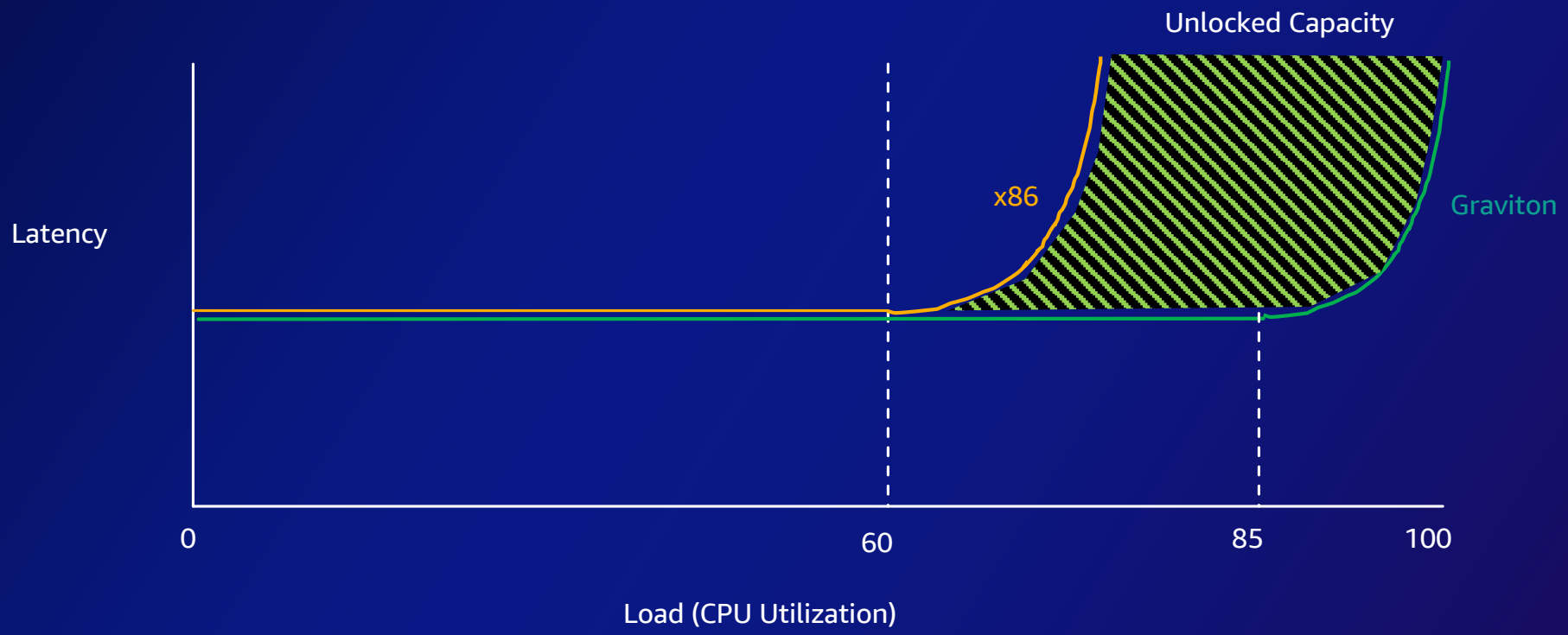
50 people	60 people	70 people	80 people	90 people	100 people
C6g	Huge performance enhancement			C7g	C6i
80% CPU				83% CPU	84% CPU
DENO based JS Game Server (tested only with 1 core / single thread server)					

1 Gacha and 5 times battle with enemy and a boss might appear randomly. Go into a lobby in the middle. Do this scenario a total of 5 times and takes merely 40 minutes.
With 3 Deno Servers, both graviton and intel shows quite similar performance (270P)

C7g.xlarge(4 VCPU, $0.1632) is 85% on-demand price against C6i.xlarge(4 VCPU, $0.192) at SEOUL region.

aws

Latency x86(SMT) vs Graviton



AWS Graviton software momentum

Operating Systems

Commercial

- Amazon Linux 2
- Amazon Linux 2023
- Red Hat Enterprise Linux 8.2+
- SLES 15 SP2
- 20.04LTS, 22.04LTS

Community

- AlmaLinux
- alpine linux
- CentOS
- Debian
- fedora
- Rocky Linux
- FreeBSD
- NetBSD

AWS tools and software

- AWS Marketplace
- AWS Systems Manager
- Amazon CloudWatch
- Amazon CodeCatalyst
- AWS CodeDeploy
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- AWS Command Line Interface
- AWS EC2 Image Builder
- AWS Auto Scaling (Mixed-arch)
- Amazon Inspector
- AWS X-Ray
- Amazon Corretto OpenJDK
- AWS Fluent Bit
- AWS Batch

Broad Graviton Support for Containers

Orchestrators

- Amazon ECS
- Amazon EKS
- Docker Swarm
- Kubernetes

Image registries

- Amazon ECR
- Docker Hub
- JFrog ARTIFACTORY
- QUAY

Container-optimized Linux distros

- Bottlerocket
- FLATCAR

Serverless

- AWS Fargate
- AWS Lambda

Broad Graviton support in DevOps Ecosystem

Fully managed

- AWS CodeBuild
- circleci
- Cirrus CI
- Travis CI
- GitHub

Hybrid (Hosted/bring-your-own-runner)

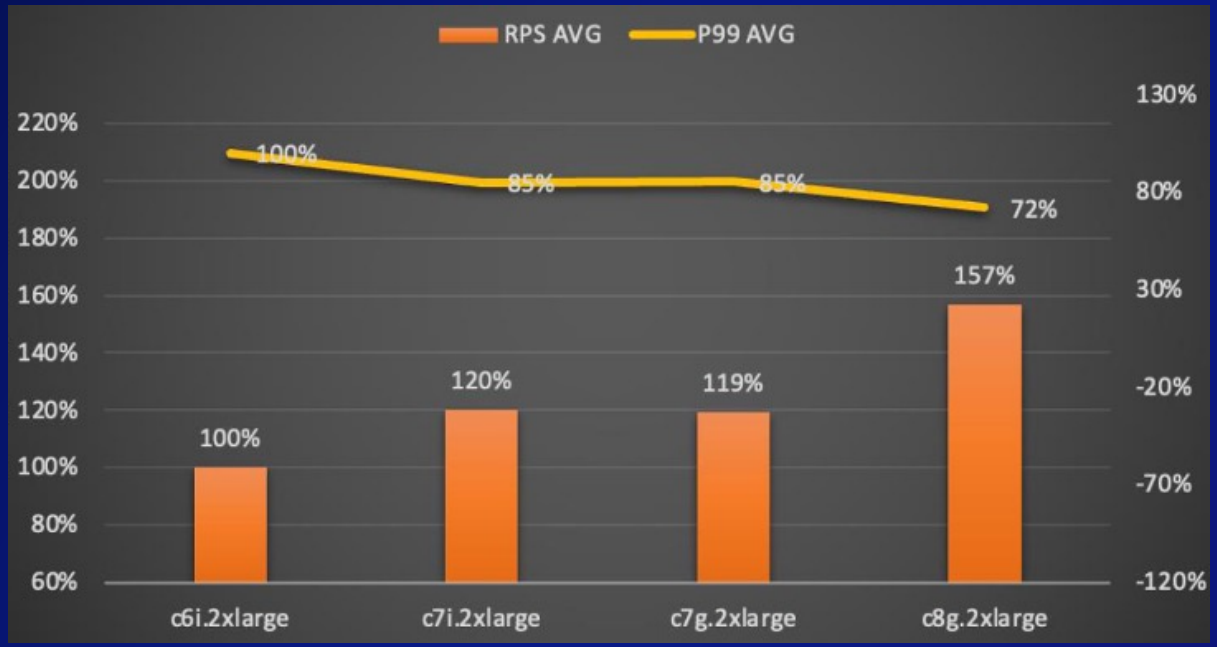
- GitHub
- GitLab

Self-managed

- Jenkins

Performance Benchmark

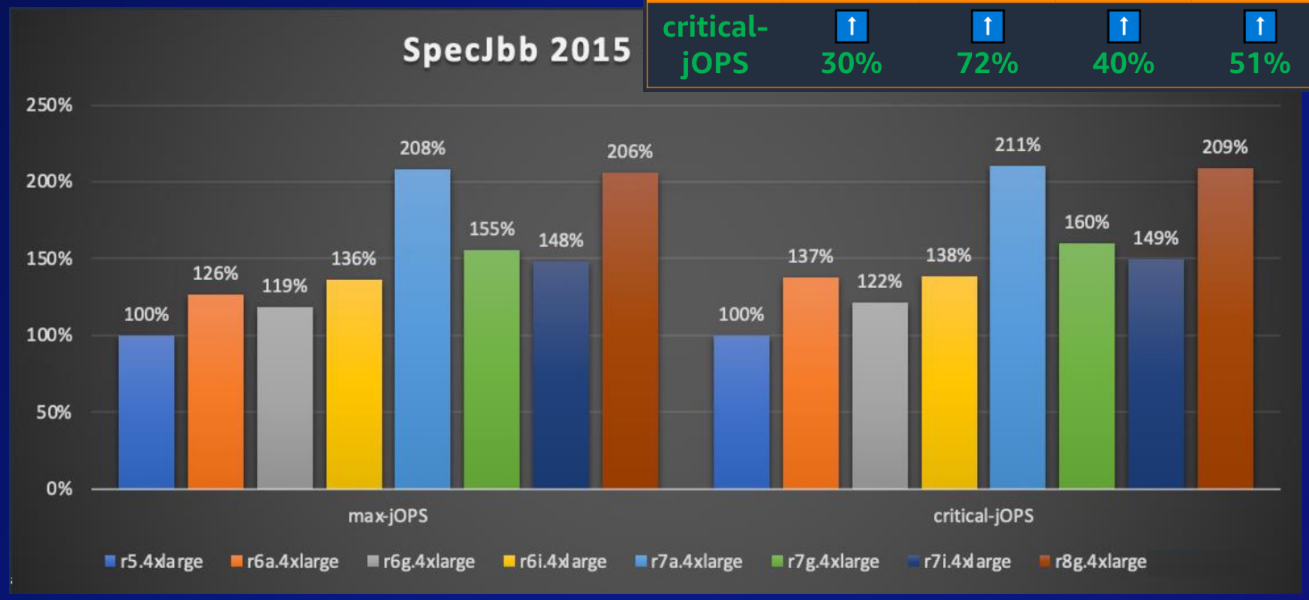
Nginx Benchmark



EC2 – C family / 2xlarge 3EA
OS - Amazon Linux 2023
TOOL – wrk2
8 threads, 10~300 connections (https)

Java Benchmark

	R8g	R7g	R6g	R7i	R6i	R5	R7a	R6a
critical-jOPS		↑ 30%	↑ 72%	↑ 40%	↑ 51%	↑ 109%	↓ 1%	↑ 52%

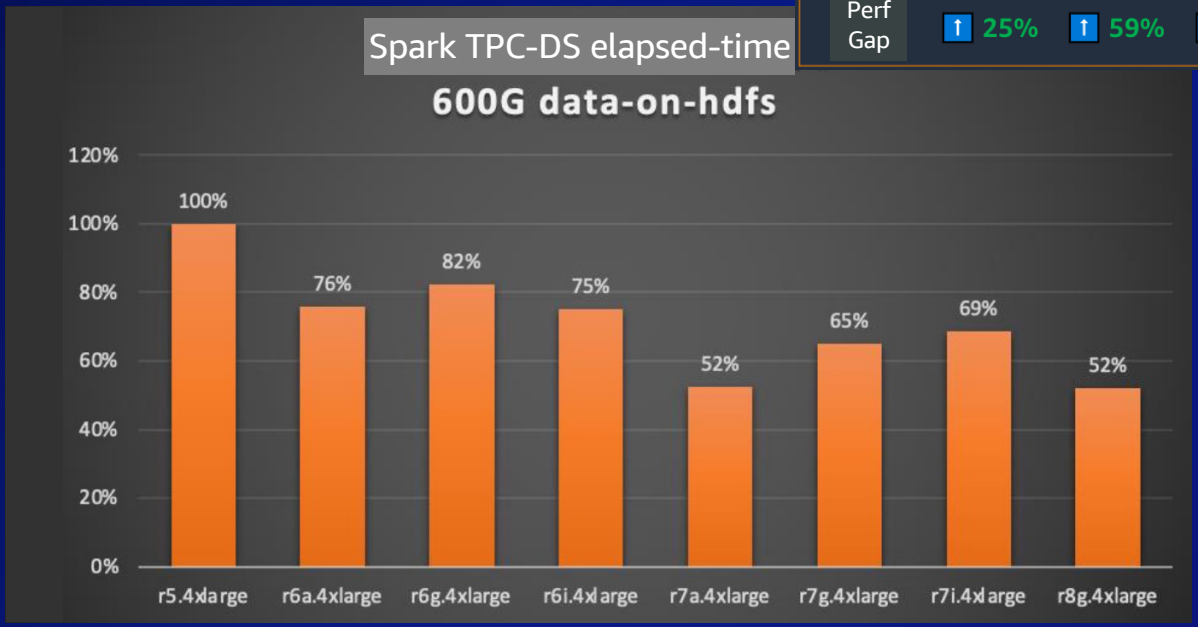


EC2 – R family / 4xlarge
OS - Amazon Linux 2023
JDK - Amazon Corretto 11

The **SPECjbb® 2015 benchmark** has been developed from the ground up to measure performance based on the latest Java application features. It is relevant to all audiences who are interested in Java server performance, including JVM vendors, hardware developers, Java application developers, researchers and members of the academic community.
max-jOPS represents the maximum throughput a system can achieve, while **critical-jOPS** measures the system's ability to maintain a certain level of **throughput under service level agreement** constraints with response times ranging from 10ms to 100ms

Spark Benchmark

	R8g	R7g	R6g	R7i	R6i	R5	R7a	R6a
Perf Gap		↑ 25%	↑ 59%	↑ 32%	↑ 44%	↑ 93%	↑ 1%	↑ 46%



EC2 – R family / 4xlarge
OS - Amazon Linux 2023
EBS – 1,000G GP3 / 8,000 IOPS

TPC-DS with hive-Benchmark (99 SQL)
JDK 8 / Scala 2.12 / Hadoop 3.3.1 / Hive 3.1.3 / Spark 3.3.1

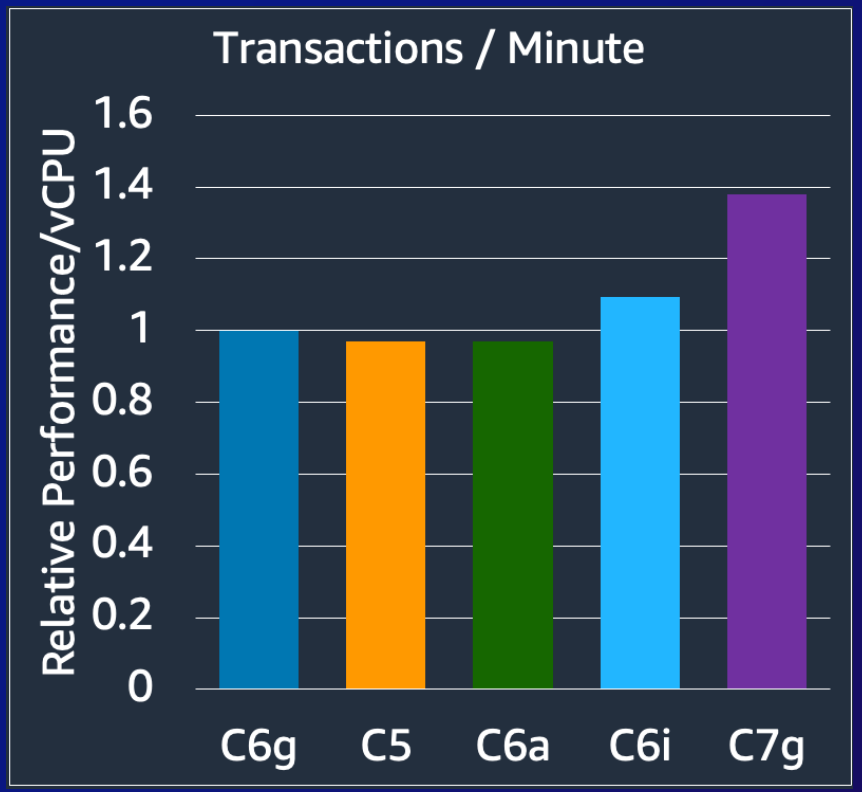
TPC-DS is a decision support benchmark created by the Transaction Processing Performance Council (TPC). It's used to measure the performance of big data systems like Hive. TPC-DS uses complex SQL queries and large datasets to simulate a retail environment, allowing developers to assess the performance of systems like Hive on large, realistic workloads

MMORPG Server FPS Benchmark (Rust)

c7g < c7i < c8g (4xlarge / 8000 active user)

MySQL Benchmark

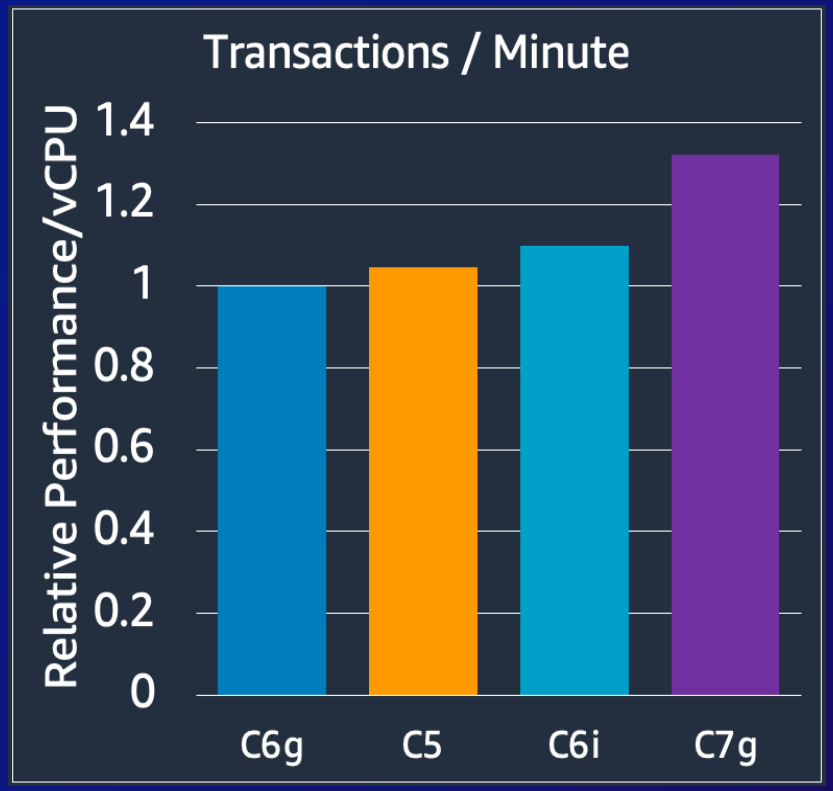
- Benchmarks the MySQL8 RDBMS engine using Hammerdb to execute a TPC-C like workload.
- Hammerdb simulates 64 virtual users against a database of ~24 warehouses. The database is sized to fit entirely in memory.
- Performance is measured as new-orders/minute.



All instances were 4xlarge
Al2, Kernel v4.14.x, MySQL v8.0.30, hammerdb v3.3, Transparent Huge Pages : always

PostgreSQL Benchmark

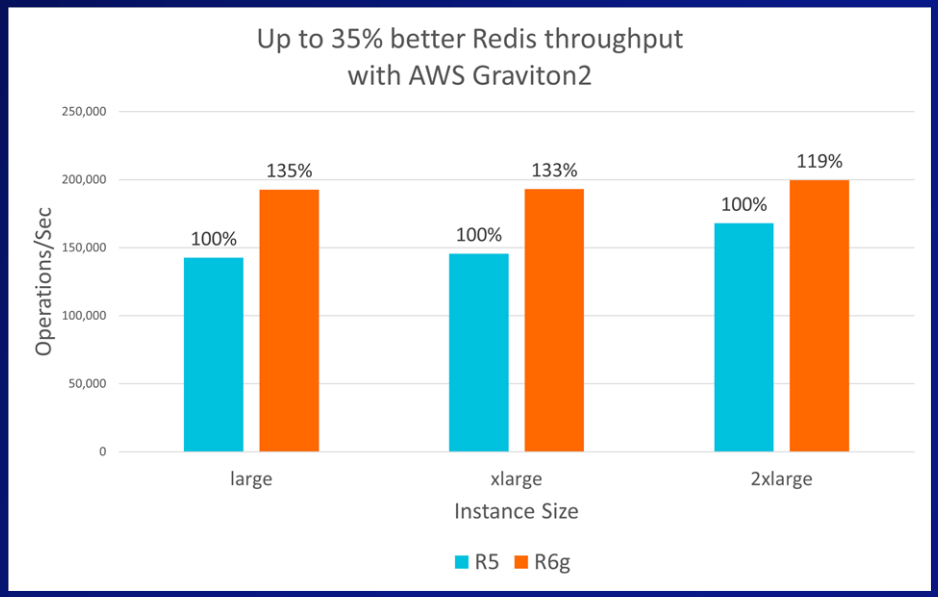
- Benchmarks the Postgresqlv14 RDBMS engine using Hammerdb to simulate a TPC-C like workload.
- Hammerdb simulates 64 virtual users against a database of ~24 warehouses. The database is sized to fit entirely in memory.
- Performance is measured as new-orders/minute.

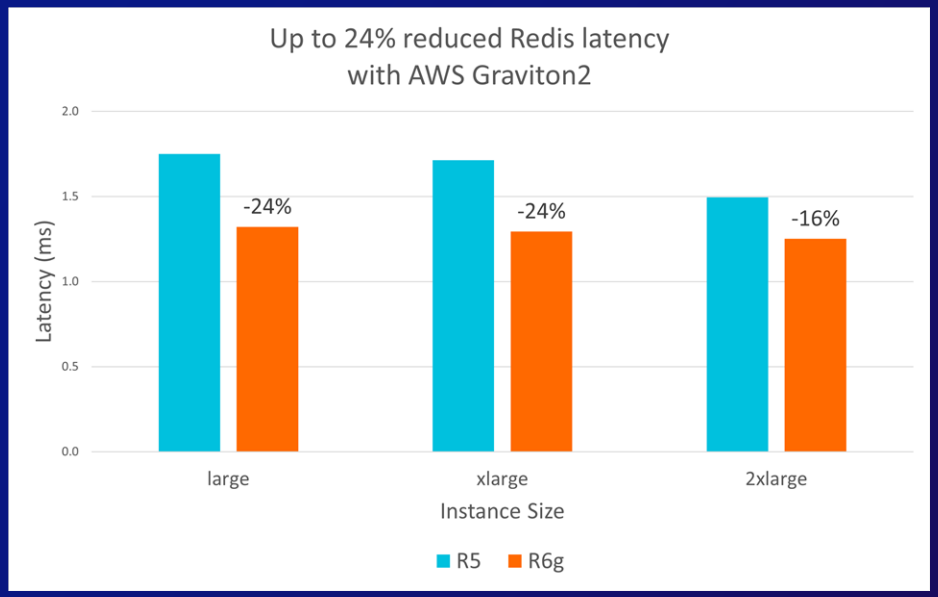


All instances were 4xlarge
Al2, Kernel v4.14.x, Postgresql v14, hammerdb v4.2, Transparent Huge Pages: never

Redis Benchmark

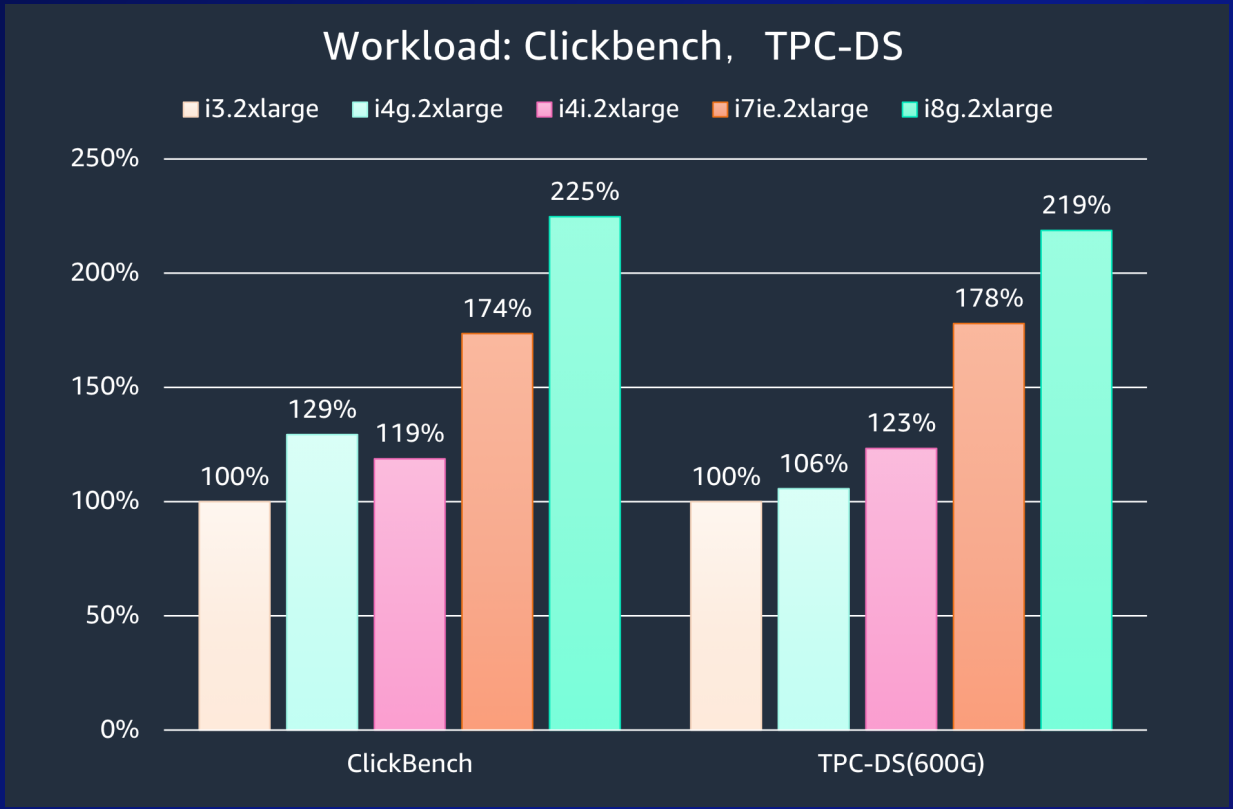
Redis 6.0.9 / Ubuntu 20.04 / Memtier benchmarking tool 1.3.0
5 threads / Sequential Key





https://community.arm.com/arm-community-blogs/b/servers-and-cloud-computing-blog/posts/redis-on-aws-graviton2

Clickhouse Benchmark

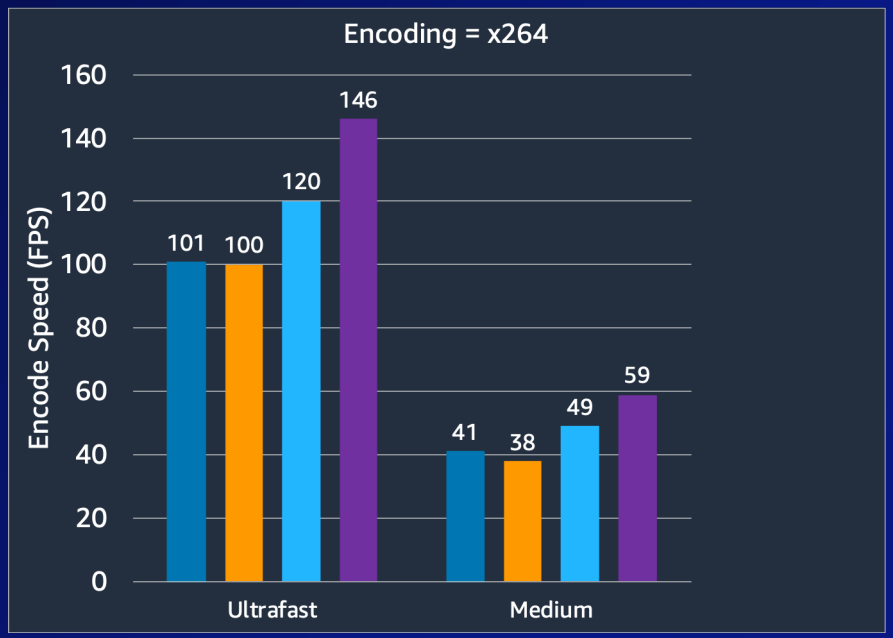


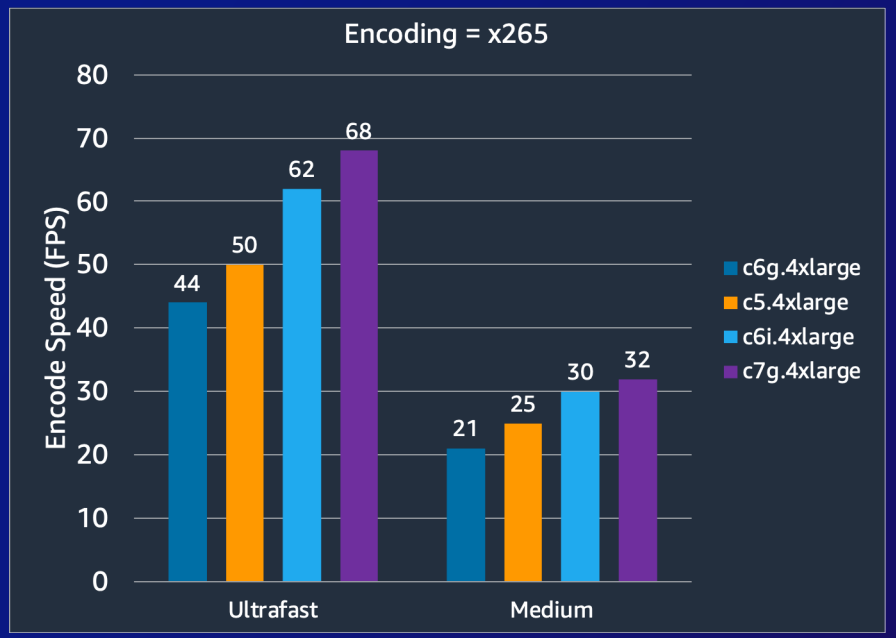
ClickHouse is a column-oriented database management system (DBMS) designed for online analytical processing (OLAP), meaning it's optimized for handling large volumes of data and complex queries, particularly in real-time

EC2	QPS	
r7i.8xlarge	2.800	
r7g.8xlarge	3.500	+25%
r8g.8xlarge	4.595	+64%

https://presentations.clickhouse.com/2024-aws/

Ffmpeg Benchmark



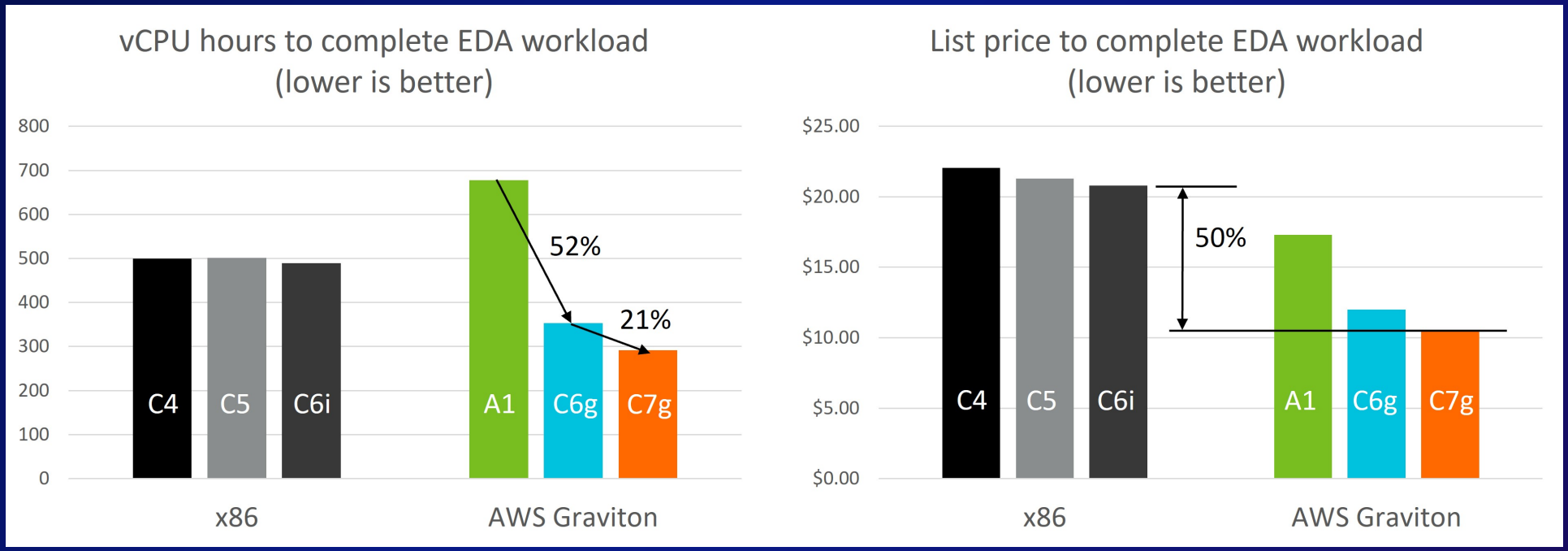


https://community.arm.com/arm-community-blogs/b/servers-and-cloud-computing-blog/posts/thirty-six-percent-better-video-encoding-with-aws-graviton2_2d00_based-c6g

https://community.arm.com/arm-community-blogs/b/servers-and-cloud-computing-blog/posts/reduce-h-265-high-res-encoding-costs-by-over-80-with-aws-graviton2-1207706725

aws

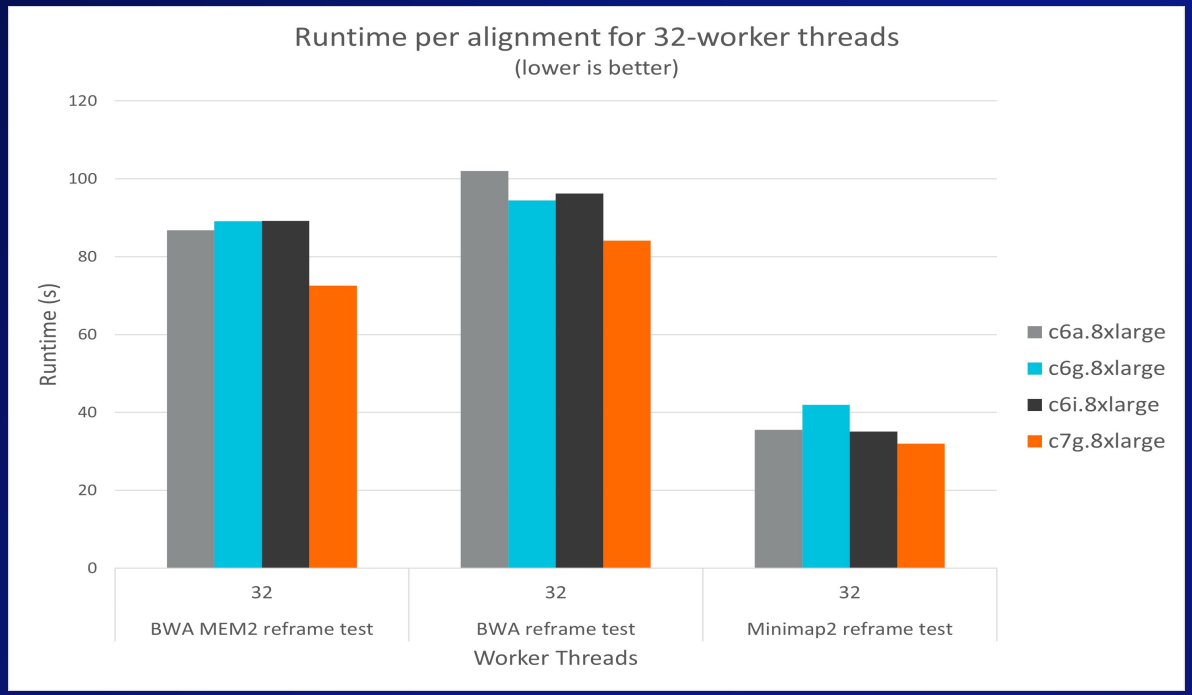
Cadence EDA Benchmark



vCPU hours and cost to complete EDA workload

Workflow tested on Cadence tools, including Xcelium, JasperGold, circuit simulation Spectre and physical characterization with Liberate.
See link https://community.arm.com/arm-community-blogs/b/high-performance-computing-blog/posts/aws-graviton3-improves-cadence-eda-tools-performance-for-arm

Genomics Benchmark (BWA/Minimap2)



Compared to the previous generation AWS Graviton2 (c6g.8xlarge) the performance is between 12% and 31% higher. However, AWS Graviton3 also demonstrates between 10% and 23% more performance compared to Intel Icelake (c6i.8xlarge) and between 11% and 21% more performance than AMD Milan (c6a.8xlarge).

This result means that AWS Graviton3 saves up to 20% over AMD Milan and up to 30% compared Intel Ice Lake

BWA is a software package for mapping DNA sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM
Minimap2 is a versatile sequence alignment program that aligns DNA or mRNA sequences against a large reference database
https://community.arm.com/arm-community-blogs/b/high-performance-computing-blog/posts/aws-graviton3-reduces-time-and-cost-for-genomics

All the others

Improving performance of PHP for Arm64 and impact on AWS Graviton2 based EC2 instances

AWS Graviton3 delivers leading AES-GCM encryption performance

Comparing data compression algorithm performance on AWS Graviton2

Amazon EMR launches support for Amazon EC2 C7g (Graviton3) instances to improve cost performance for Spark workloads by 7–13%

Gain up to 30% Cost-Performance benefits for Apache Kafka on AWS Graviton2 Processors

Optimize your Elasticsearch deployment with Arm-based Amazon EC2 M6g instances

MongoDB performance on Arm Neoverse based AWS Graviton2 processors

Apache Arrow optimization on Arm

23% Cost savings and 36% performance gain by deploying GitLab on Arm-based AWS Graviton2

Making your Go workloads up to 20% faster with Go 1.18 and AWS Graviton

Increase performance by up to 30% by deploying Apache Cassandra on AWS Graviton2

Improve ClickHouse Performance up to 26% by using AWS Graviton3

Migration Strategy

Start with managed services

Compute


AWS Batch


AWS Lambda


Amazon EKS


Amazon ECS


AWS Elastic Beanstalk


AWS Fargate

Databases


Amazon RDS


Amazon Aurora


Amazon ElastiCache


Amazon MemoryDB


Amazon DocumentDB


Amazon Neptune

ML & Analytics


Amazon SageMaker


Amazon EMR


Amazon OpenSearch

Storage


Amazon FSx for Lustre


Amazon FSx for OpenZFS

Ease of adoption

Difficulty	Workload	Actions
Virtually no effort	RDS, Aurora, ElastiCache, OpenSearch, MemoryDB, & Neptune	Upgrade to latest and enjoy
Super easy	EMR	Typically, just works
Pretty easy	AWS Lambda	Typically, just works with Lambda managed runtimes or base images. ⚠ Check for JNI, shared objects, or native modules
Quite easy	Linux – Interpreted and JIT'd languages (e.g., Java, PHP, Python, Node.js, .NET Core)	Select Arm64 AMI and Install Bonus if containerized ⚠ Check for JNI, shared objects, or native modules
More involved	Linux – Compiled languages (e.g., C/C++, Go, Rust)	Select Arm64 AMI and compile ⚠ Port any intrinsics, assembly, or native modules
Some work, high reward	Microsoft Windows – .NET	Migrate to Linux + .NET core on Arm64

As a rule, the more current your software stack the better

Migration plan

Before Starting

Use Porting Advisor for Graviton!

Know Your:

- Software & Dependencies
- Coding Languages
- AWS Services

Start with Easy Wins

Migrate AWS Managed Services first

Devise a Strategy

Prioritize Workloads with Biggest Impact and Lowest Friction

Execute Migrations

Try Proof-of-Concept First

Have a Testing Strategy

DevOps Skill is Key to Success

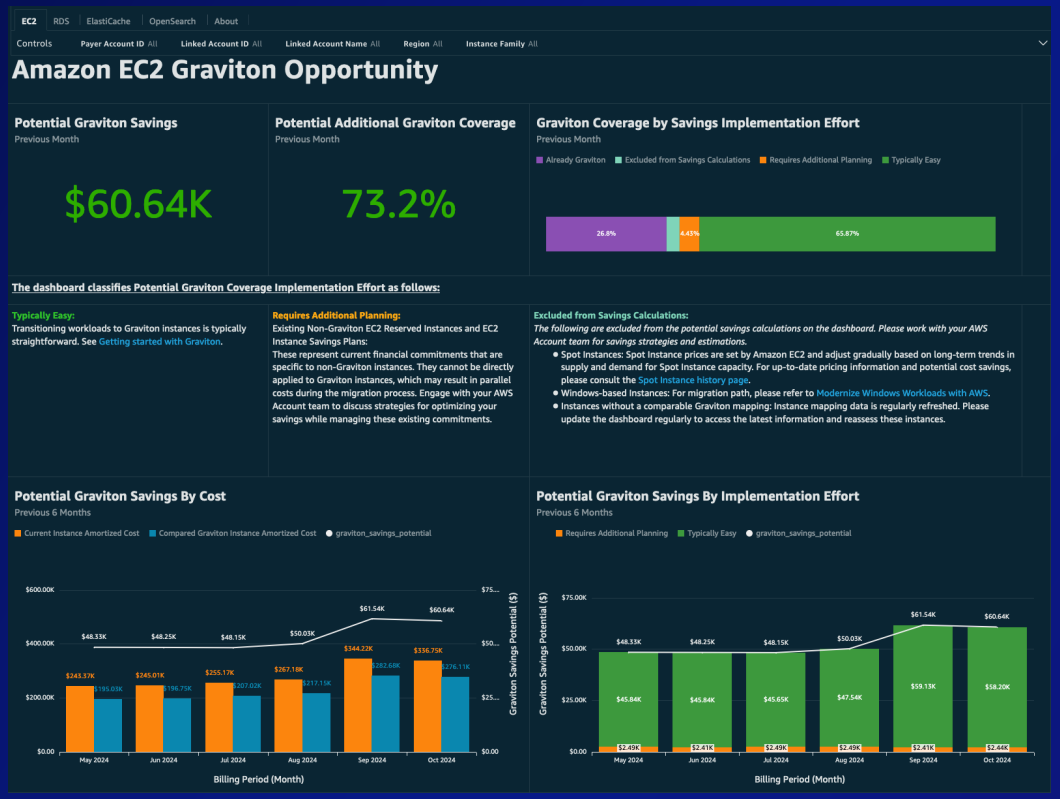
Observe and Report

Cost Savings

Performance Improvements

Carbon Footprint Reduction

AWS Graviton Savings Dashboard



See how much you can save with AWS Graviton

The Graviton Savings Dashboard is a visualization tool that helps you understand the impact of current and future AWS Graviton usage on your workloads. In as little as 1 hour, the Graviton Savings Dashboard helps you evaluate and find the potential savings of moving workloads to AWS Graviton.

your AWS Cost and Usage Reports contain all the details necessary for the Graviton Savings Dashboard.

https://aws.amazon.com/ec2/graviton/graviton-savings-dashboard/

More recent software gives better performance

11+

5+

18+

1.18+

7.4+

11+

3.0+

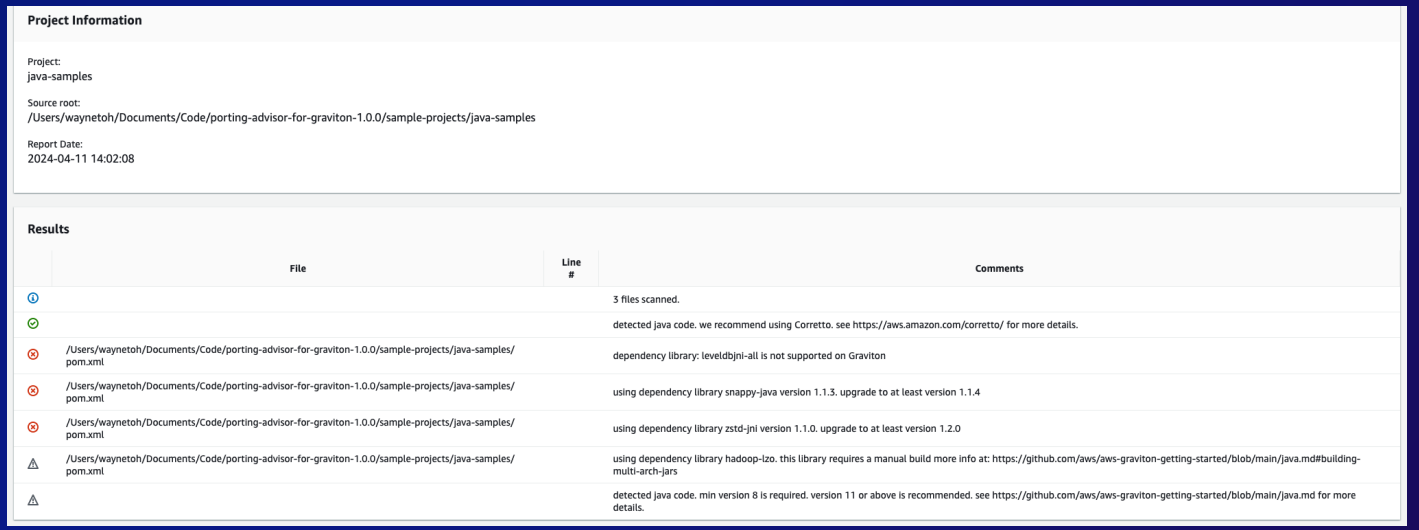
19.3+

1.0+

The minimum version supporting ARM architecture , each latest would be more better.

Porting Advisor

- An open-source tool that **provides guidance** on getting applications Graviton ready
- **How?** Analyzes application source code and generates recommendations of minimum required or recommended versions for a subset of language runtime and library dependencies https://github.com/aws/porting-advisor-for-graviton
- Currently supports
- Python 3+
- Java 8+
- Go 1.11+
- C, C++, Fortran



Project Information

Project:
java-samples

Source root:
/Users/waynetoh/Documents/Code/porting-advisor-for-graviton-1.0.0/sample-projects/java-samples

Report Date:
2024-04-11 14:02:08

Results

	File	Line #	Comments
ⓘ			3 files scanned.
✓			detected java code. we recommend using Corretto. see https://aws.amazon.com/corretto/ for more details.
⊗	/Users/waynetoh/Documents/Code/porting-advisor-for-graviton-1.0.0/sample-projects/java-samples/pom.xml		dependency library: leveldbjni-all is not supported on Graviton
⊗	/Users/waynetoh/Documents/Code/porting-advisor-for-graviton-1.0.0/sample-projects/java-samples/pom.xml		using dependency library snappy-java version 1.1.3. upgrade to at least version 1.1.4
⊗	/Users/waynetoh/Documents/Code/porting-advisor-for-graviton-1.0.0/sample-projects/java-samples/pom.xml		using dependency library zstd-jni version 1.1.0. upgrade to at least version 1.2.0
⚠	/Users/waynetoh/Documents/Code/porting-advisor-for-graviton-1.0.0/sample-projects/java-samples/pom.xml		using dependency library hadoop-lzo. this library requires a manual build more info at: https://github.com/aws/aws-graviton-getting-started/blob/main/java.md#building-multi-arch-jars
⚠			detected java code. min version 8 is required. version 11 or above is recommended. see https://github.com/aws/aws-graviton-getting-started/blob/main/java.md for more details.

Amazon Q Code Transformation

- Complete Java language upgrades in a fraction of the time
- Preview for .NET Framework to .NET Core

Internal Amazon results

1,000
production applications upgraded from Java 8 to Java 17 in just two days

10
minutes on average to upgrade each application

Less than 1
hour to complete the longest upgrade

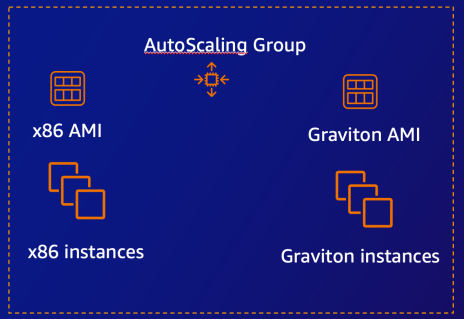
2+
days previously taken to upgrade each application

https://aws.amazon.com/ko/blogs/korea/upgrade-your-java-applications-with-amazon-q-code-transformation-preview/

EC2 Migration Consideration

EC2 AutoScaling Group Migration

1. Create a Graviton-based AMI
2. Create a new ASG launch template version
3. Update with new AMI id and Graviton instance type
4. Use an *instance refresh* to update the instances in your Auto Scaling group



https://ec2spotworkshops.com/efficient-and-resilient-ec2-auto-scaling/lab1/110-instance-refresh.html

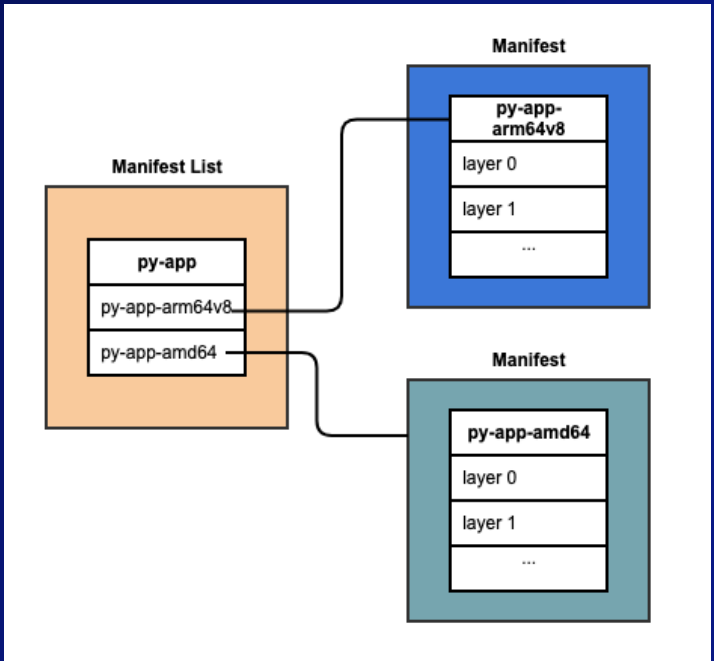
https://aws.amazon.com/ko/blogs/aws/new-application-load-balancer-simplifies-deployment-with-weighted-target-groups/

Canary Releases

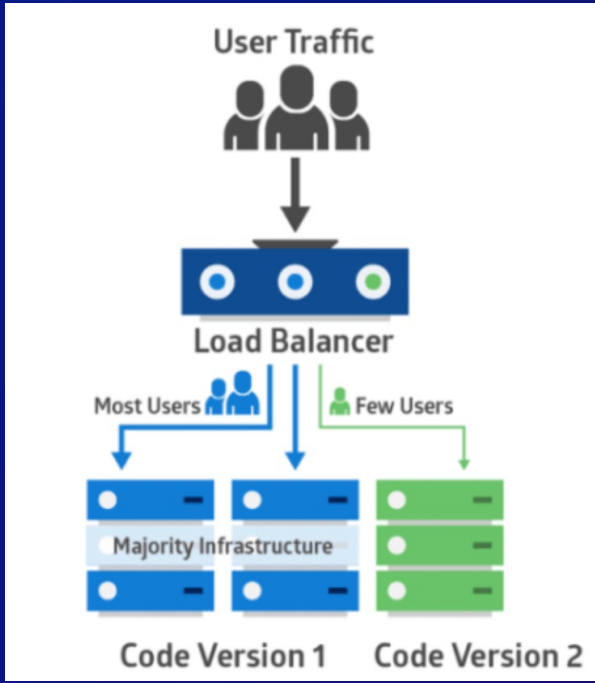
Introduce Graviton servers on a few instances and increase the footprint over time

Container Migration Consideration

Multi Architecture & Continuous Delivery/Deployment

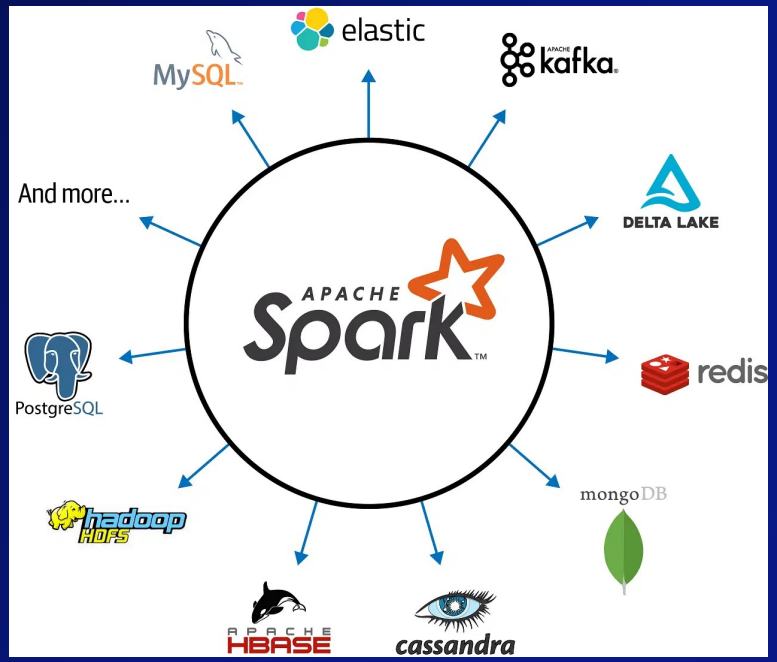


Docker Multi Platform Image



Rolling / BlueGreen / Canary

Bigdata Migration Consideration



python

Java

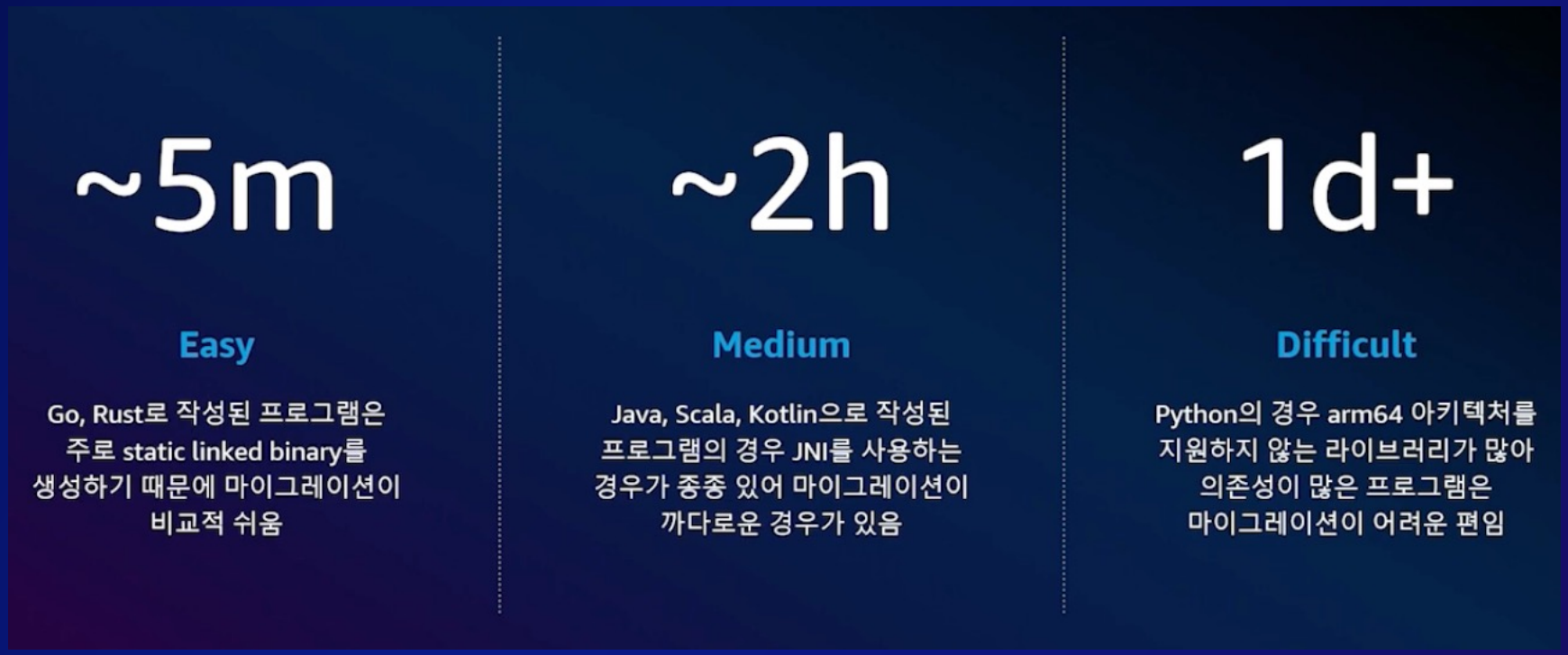
Scala

GO

SQL

Most of bigdata engine have been developed with JAVA/Scala and a little with C/C++.
Decoupling between Computing and Storage is standard. Data is stored in S3
We're focusing on computing engine / node / application code
Need strong check for data pipeline, especially implemented with python. (native library dependency)

Migration difficulty by programming language



External library dependency using other language.

Performance Profiling / Tunning

Factors affecting Performance

OS / Kernel Parameter
Hardware Architecture / Spec
Development Language
Library dependency
Runtime
Application Code Quality
Parallelism
Workload Type / Characteristics

...

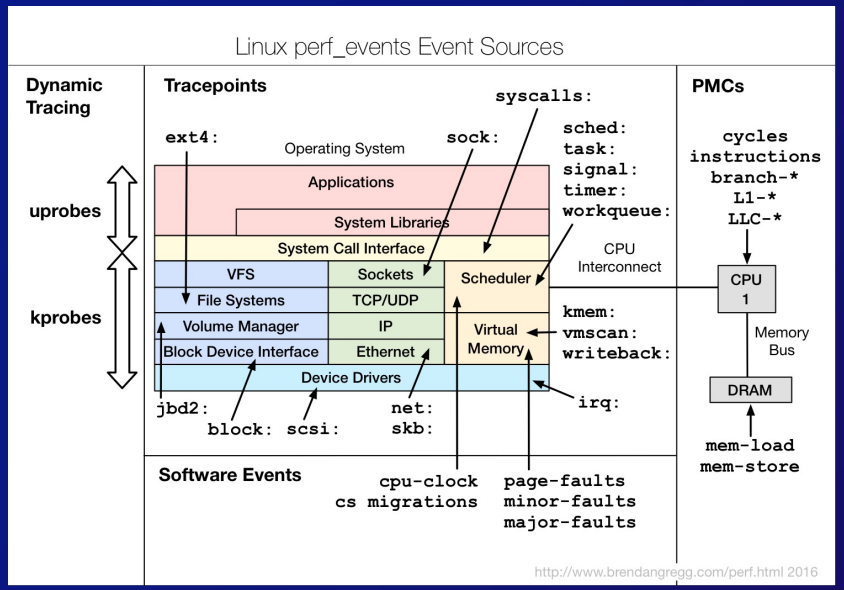
Linux Perf

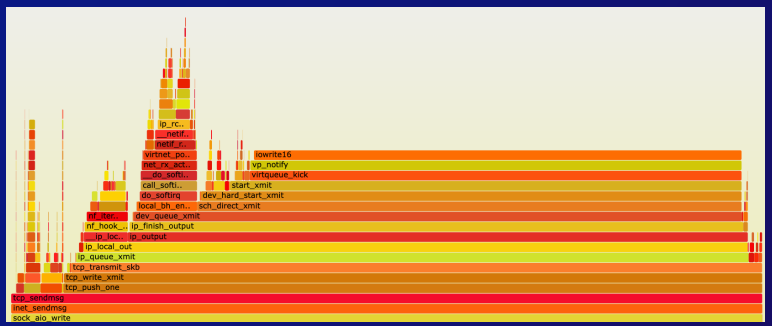
"Perf" most commonly refers to perf, a performance analysis tool in Linux. It's part of the Linux kernel and is used to profile both user-space applications and kernel code. The name "perf" is short for "Performance Counters for Linux", originally known as PCL (Performance Counters for Linux).

$ perf top -p <pid>

```
Samples: 108  of event 'cycles:ppp', Event count (approx.): 2648224
Overhead  Shared Object                      Symbol
  19.36%  libpython3.11.so.1.0               [.] _PyEval_EvalFrameDefault
  13.57%  libpython3.11.so.1.0               [.] _PyObject_Malloc
   7.40%  [kernel]                           [k] rseq_get_rseq_cs.isra.0
   7.40%  libpython3.11.so.1.0               [.] tupledealloc
   5.67%  libpython3.11.so.1.0               [.] PyObject_Size
   4.96%  [kernel]                           [k] complete_walk
   4.34%  libpython3.11.so.1.0               [.] list_dealloc
   4.34%  libpython3.11.so.1.0               [.] cfunction_vectorcall_NOARGS
   4.04%  libpython3.11.so.1.0               [.] structseq_dealloc
   3.84%  [kernel]                           [k] __check_object_size.part.0
```

$ perf record -F 99 -p <pid> -g -- sleep 60
https://github.com/brendangregg/FlameGraph





DATADOG Application Profiler

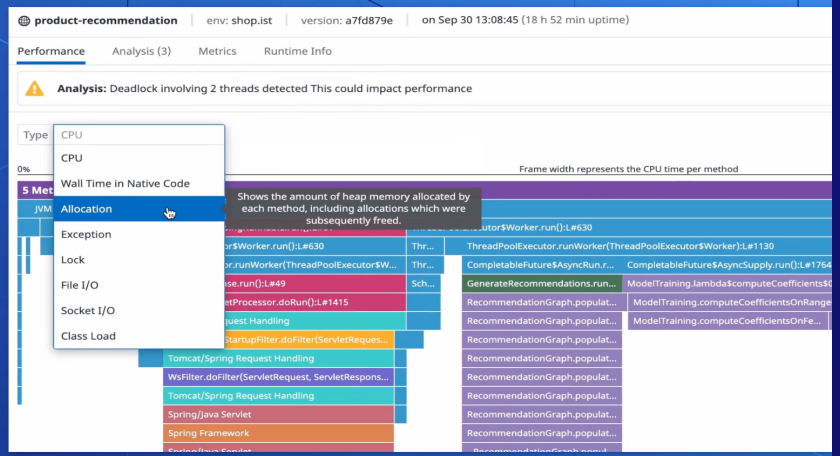
DATADOG

GET STARTED FREE

Continuous Application Profiler

Rapidly identify and optimize the most resource-consuming parts of your application code. Analyze code performance in production, at any scale, with neglible overhead.

GET STARTED FREE



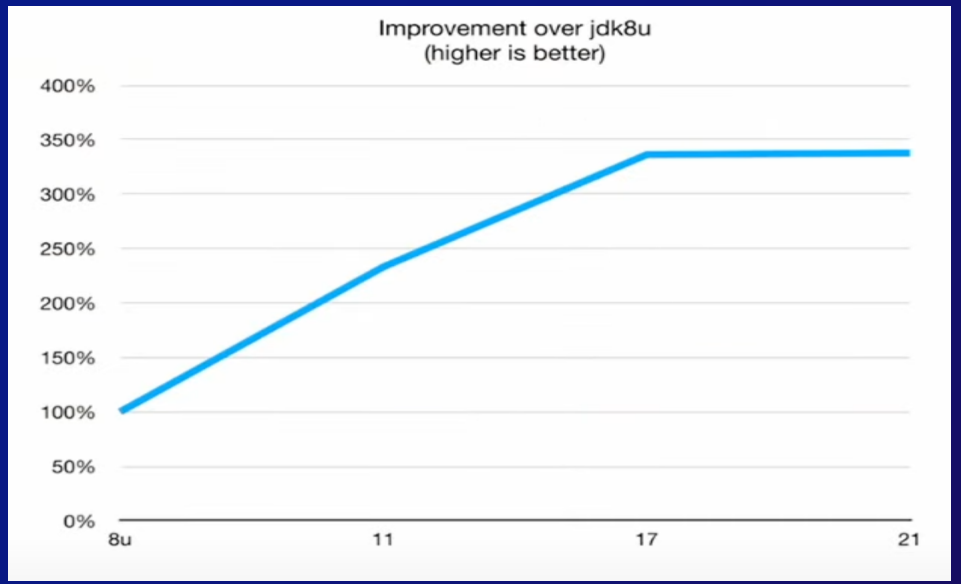
SUPPORTED LANGUAGES

.NET C C++ GO Java node.js php python Ruby Rust

Java Performance Over Time

On ARM64 specifically, latency Performance is up to 350% from Java 8 baseline.

- **General performance improvements with better GC, language features**
- **More platform-specific intrinsics**
- **Aarch64 specific optimizations such as SIMD, atomics, memory barriers**



Accelerating performance of Java applications on Arm64 by Dave Neary

For more detail about intrinsics, check Project Panama: Interconnecting JVM and native code - https://openjdk.org/projects/panama/
String and memory functions, JAVA call OS libraries such as indexOf, arraycopy, etc.
SIMD means single instruction for multiple data, executed with ARM NEON instruction.
Atomic instructions (LL/SC) exist to ensure that a value cannot change between when it's read with LD instructions and a new value is stores with ST instruction. In Java, synchronized blocks or volatile data type are one example of atomic instructions.

JVM Performance Tunning

There is no ARM specific JAVA performance tunning point, it’s just related with JVM itself

Check JVM Version & Flavor:
The more recent the better. AWS recommends at least JDK11, but ideally JDK17 or newer. JDK8 is the minimum version supporting Arm64 but likely won’t provide the best performance.

Check JARs and shared objects for architecture specific code:
JNI extensions usually exist to implement performance critical functions in a language other than Java. Without an Arm64 version the code may not work or can fall back on a slower pure Java implementation

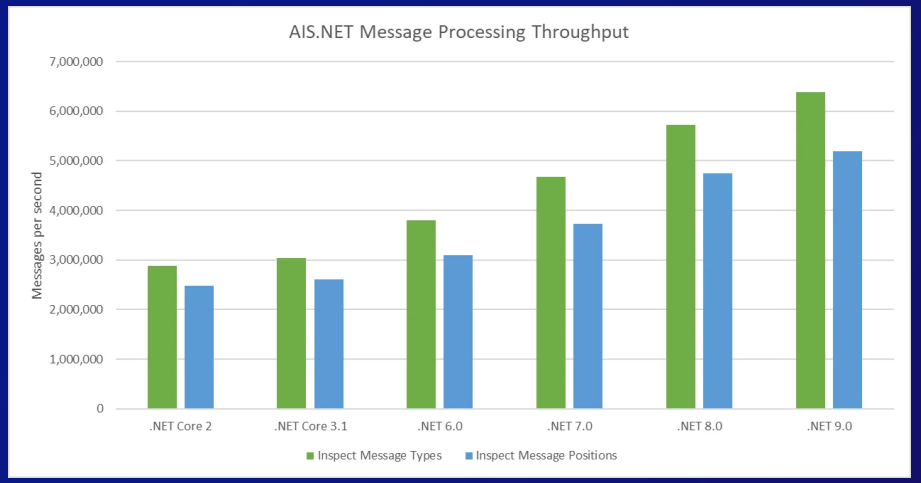
Java Crypto operations:
AES/GCM benefits when using AES hardware instructions, which can improve performance by up to 5x for this algorithm. Corretto & OpenJDK 18 support this by default and have been back-ported to Corretto & OpenJDK 11 and 17 which can be enabled
using XX:+UnlockDiagnosticVMOptions -XX:+UseAESCTRIntrinsics

Java JVM Options:
Flags -XX:-TieredCompilation -XX:ReservedCodeCacheSize= -XX:InitialCodeCacheSize=

.NET Core Performance

dotnet improves CLR performance with similar approach with JAVA

- General performance improvements with better GC, language features
- More platform-specific intrinsics
- Aarch64 specific optimizations such as SIMD, atomics, memory barriers



Check Runtime if working as emulation mode with dotnet –info or global.json file
In emulation mode, CPU usage is around 30% higher than native mode.

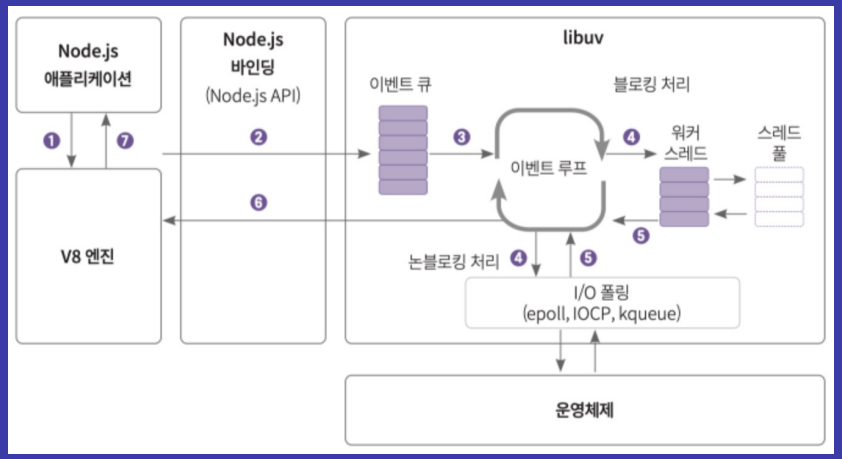
For .NET 6 and later, you can use environment variables for GC mode modification

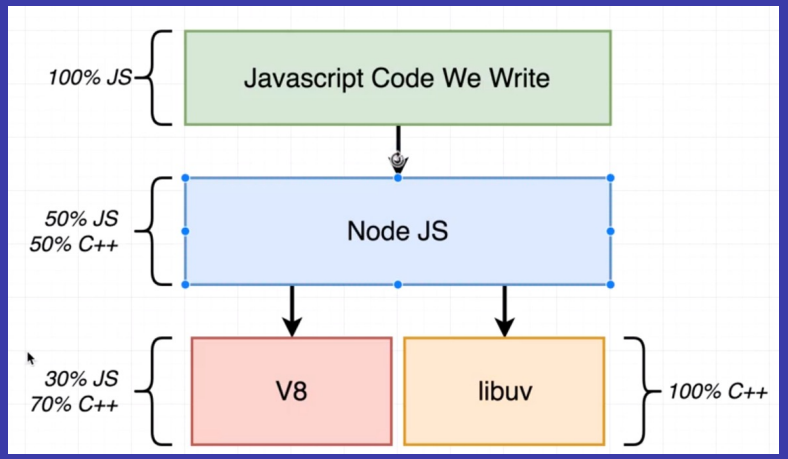
```
export DOTNET_gcServer=1  # Enable server GC
export DOTNET_GCHeapCount=12  # Number of heaps (optional)
export DOTNET_gcConcurrent=1  # Enable concurrent GC
```

https://endjin.com/blog/2024/11/how-dotnet-9-boosted-ais-dotnet-performance-by-9-percent-for-free

Node.js Performance

Node.js performance on ARM architecture has evolved significantly. While early comparisons sometimes showed x86_64 outperforming ARM in specific benchmarks, the gap has narrowed, and ARM-based systems are increasingly viable for Node.js deployments, especially in cloud environments and edge devices.





Node.js uses the "Single Threaded Event Loop" architecture to handle multiple concurrent clients. Node.js Processing Model is based on the JavaScript event-based model along with the JavaScript callback mechanism.
Node.js uses libuv for asynchronous I/O operations, which internally uses a thread pool for certain tasks like file system operations, DNS resolution, and cryptography. The size of this thread pool can be adjusted using the UV_THREADPOOL_SIZE environment variable (default 4 max 128)

Node.js Performance Tunning (1)

Use async/await (and the underlying asynchronous I/O APIs) for:

I/O-bound operations (network requests, file operations, database interactions). Tasks that involve waiting for external resources without significant computation. Does not parallelize CPU-intensive tasks; such tasks will still block the single main thread, making the application unresponsive during their execution

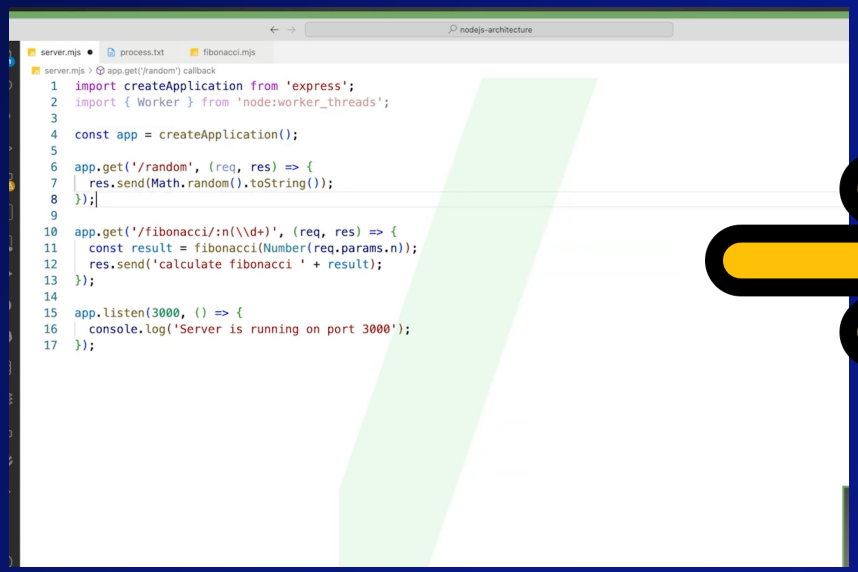
Use Worker Threads for:

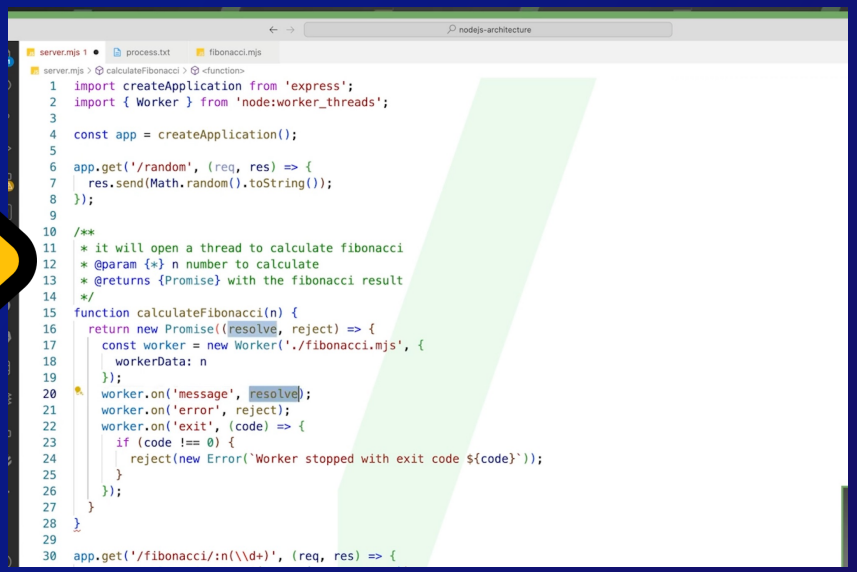
CPU-intensive tasks that would otherwise block the main event loop. Tasks that can be broken down into independent units and processed in parallel.

Others:

Enable turbo-instruction-scheduling which optimize JavaScript code for better performance by generating efficient machine code (node --turbo-instruction-scheduling index.js)

Node.js Performance Tunning (2)





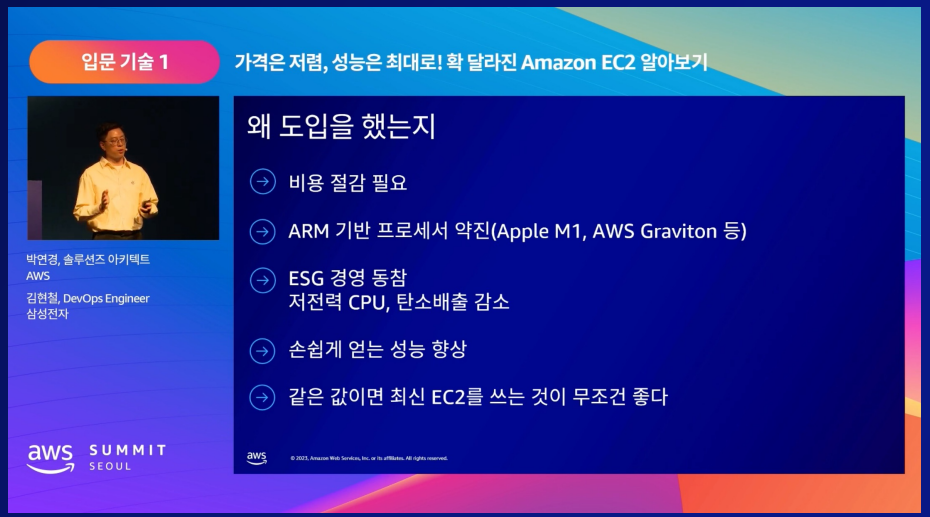
Check Addon (Native Code) performance bottleneck if you use external library explicitly and you can profile your application like this (node --prof index.js)

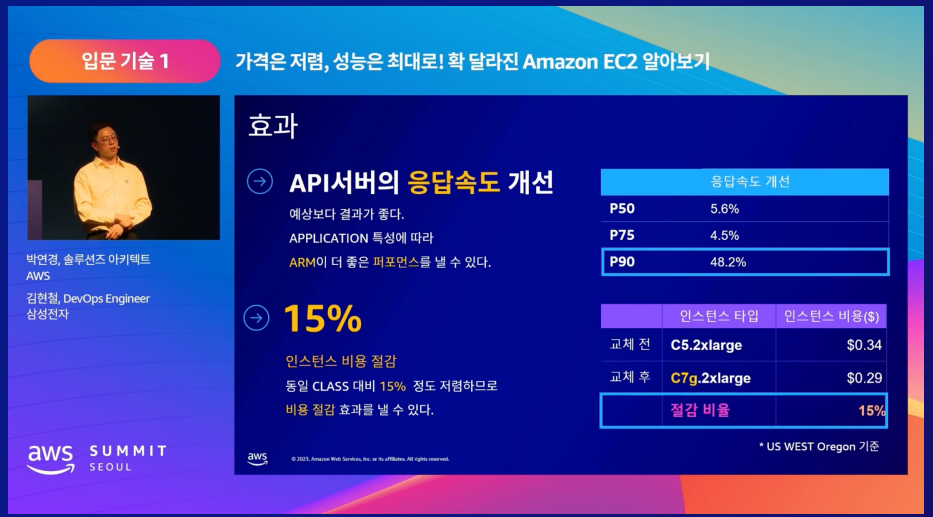
If application works as single thread and just only occupy one CPU, run the multi copies of application.

Customer Case Study

Graviton Customers

SAMSUNG EKS Graviton Adoption



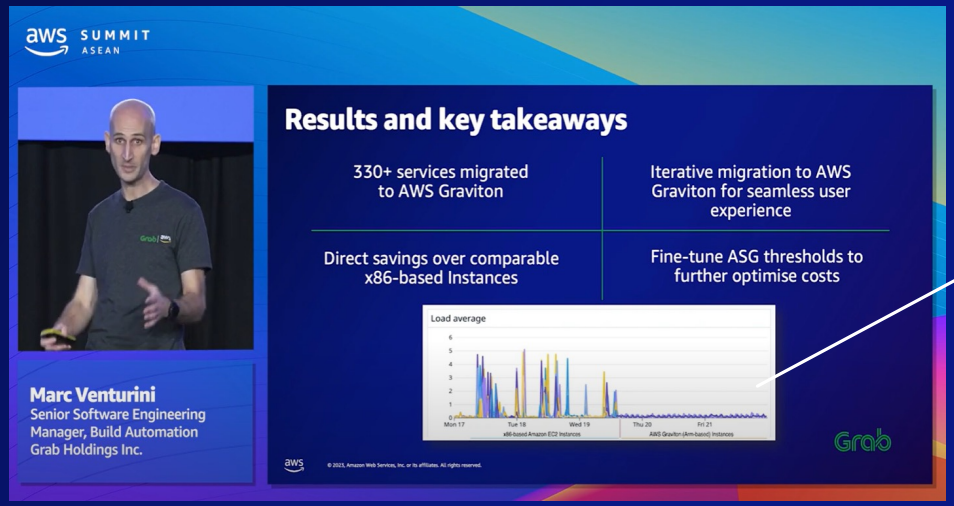


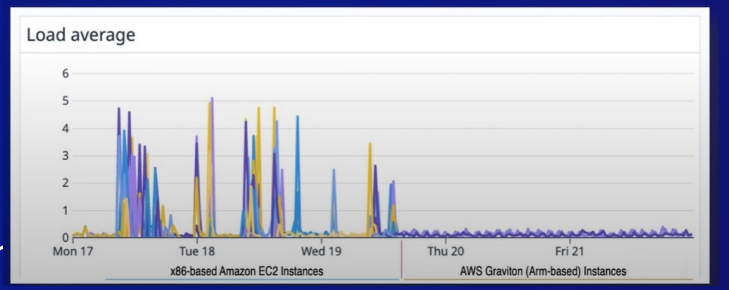
https://www.youtube.com/watch?v=kByUnvQE_U0 (26:00-)

Mobile TVPlus, Samsung Podcast

- Gained 15% cost saving by changing C5.2xlarge to Graviton3 (C7g.2xlarge)
- Upgraded JDK from V8 to V11 with latest patch
- Response Time was improved (P50 5.6%, P90 48.2%)
- Application Stack is Java / Spring / Datadog

Grab's AWS EC2 Graviton Adoption

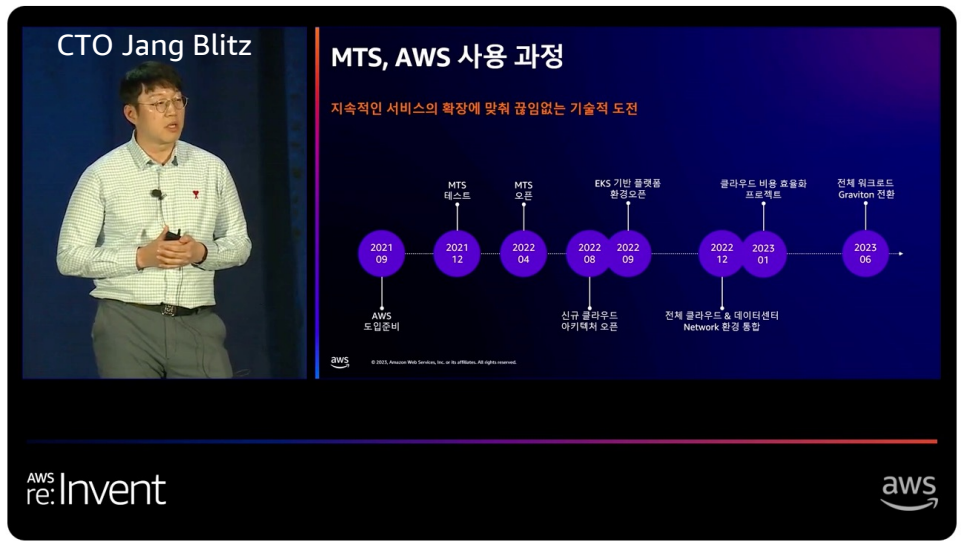




https://youtu.be/3zufC4833f8 (13:30-)

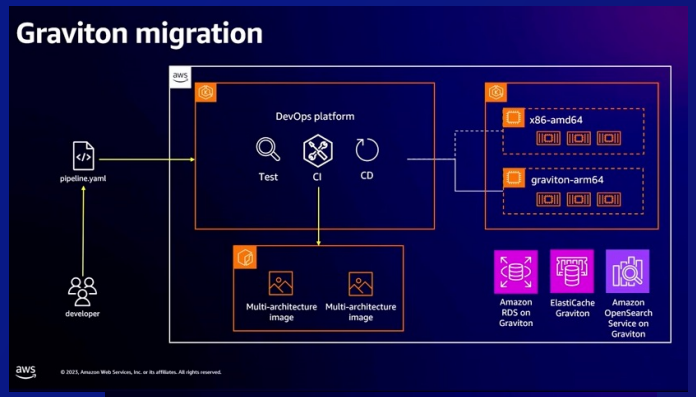
- Gained 10-20% cost saving by changing intel/AMD to Graviton2.
- Migrated 80% of workloads (330+ services) to Graviton.
- CPU usage has also stabilized.
- Application stack is Golang and docker

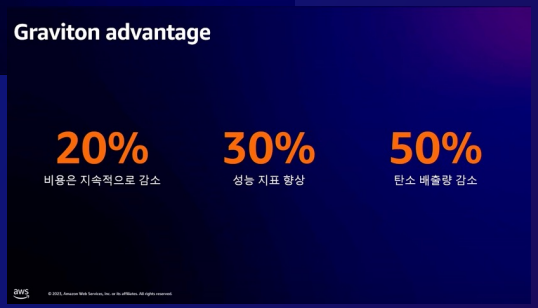
Kakaopay securities's Graviton Adoption



AWS re:Invent 2023 - How to scale at speed on AWS while addressing security and compliance (GBL205)

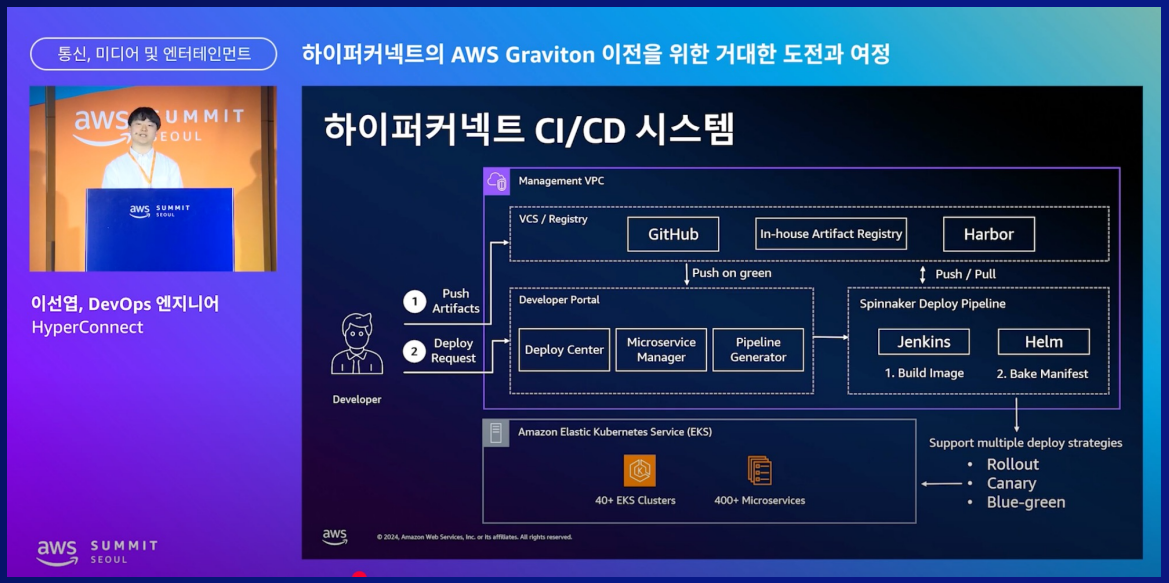
https://www.youtube.com/watch?v=WljJ-sBQLS8&list=PLORxAVAC5fUW40w3WpbSbACrHZqhoQmG6&index=8 (11:30-)

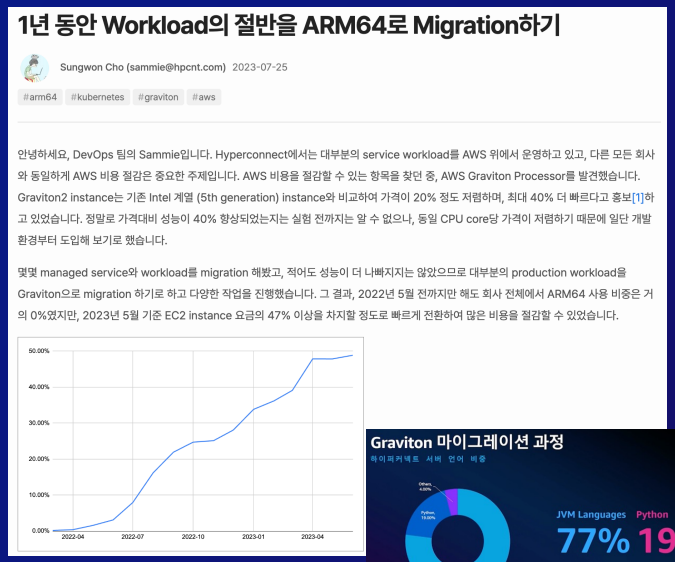




- Migrated entire service workloads to Graviton
- Reduce 20% TCO and improve 30% performance

HyperConnect Graviton Adoption at scale

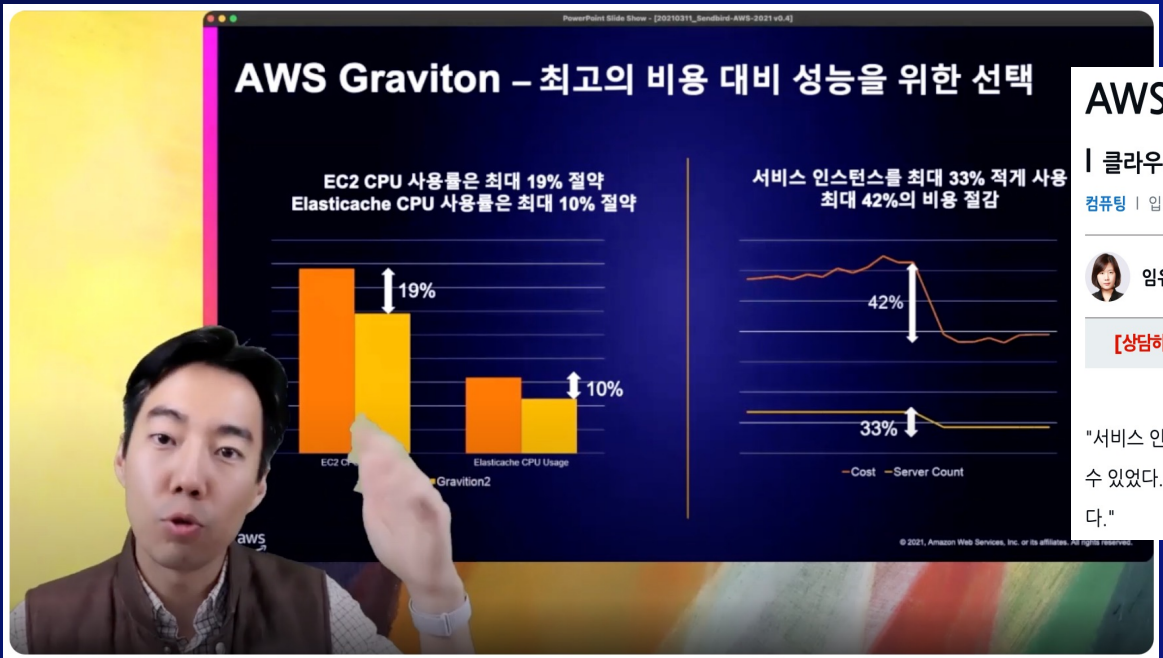




https://www.youtube.com/watch?v=AE1QHyFnZsw (10:00-)

- Cost saving by changing intel EC2/EKS Nodegroup to Graviton3.
- Has 30 AWS accounts across 10 regions and migrated 300 services to graviton3 for 2 years, which corresponds over 80% of our total EC2 servers.
- Application stack is Java / Kotlin / Python / Golang

Sendbird Graviton Adoption (42% Cost Saving)



AWS Summit Online Korea 2021 | 기조연설 - 김동신 대표(센드버드)

https://www.youtube.com/watch?v=1ybzxarCEls (5:50-)

AWS 쓰는 센드버드, ARM 인스턴스로 갈아 탄 이유

| 클라우드 비용 42% 절감...전체 프로세스 이전 중

컴퓨팅 | 입력 :2021/05/11 15:17 수정: 2021/05/11 16:16

임유경 기자 | 기자 페이지 구독 기자의 다른기사 보기

[상담하기] 2025 ICT 중소기업 정보보호 지원사업 - 보안 솔루션 도입 80% 혜택을 놓치지 마세요!

"서비스 인스턴스 사용을 최대 33% 줄이고, 전체 비용은 42%까지 절감할 수 있었다. 어떤 코드나 아키텍처 변화 없이 그래비톤2 적용만으로 거둔 성과다."

https://zdnet.co.kr/view/?no=20210511113119

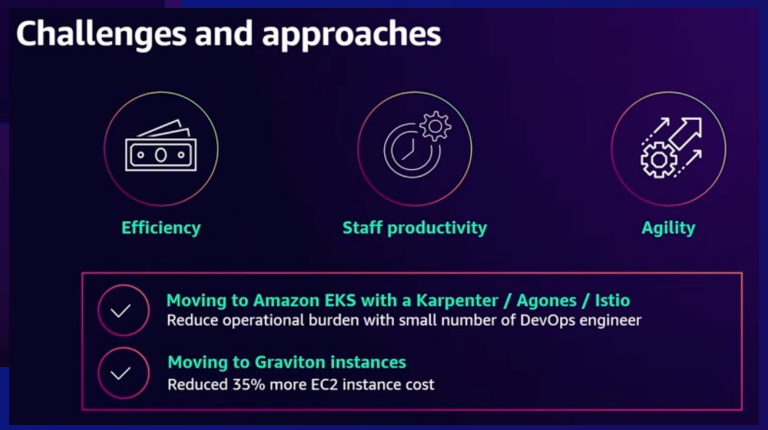
- Cost saving up to 42%
- Reduce 33% instances than before
- EC2 (Max CPU 19% decrease)
- Elasticache (Max CPU 10% decrease)

SKT OSS TANKGO Graviton Adoption

SK텔레콤은 그라비톤을 통해 비용과 성능의 균형을 맞추며 워크로드 규모가 큰 만큼 상당한 비용 절감 효과를 얻었다. 데이터베이스 서비스인 아마존 오로라를 그라비톤2로 전환했을 때 CPU 활용률과 쿼리 처리 속도 등 주요 지표에서 약 1.3배 이상 성능이 향상됐으며 비용도 약 10% 저렴해졌다. 또한 데이터 수집 및 전처리 파이프라인으로 활용 중인 MSK를 그라비톤3로 전환해 비용을 50% 절감하는 효과를 거뒀다. EKS 의 경우 그라비톤2에서 그라비톤3로의 전환, 그라비톤3 적용 후 성능 향상에 따른 최적화, 카펜터 적용 등을 통해 전년 대비 약 50% 수준의 비용을 절감했다.

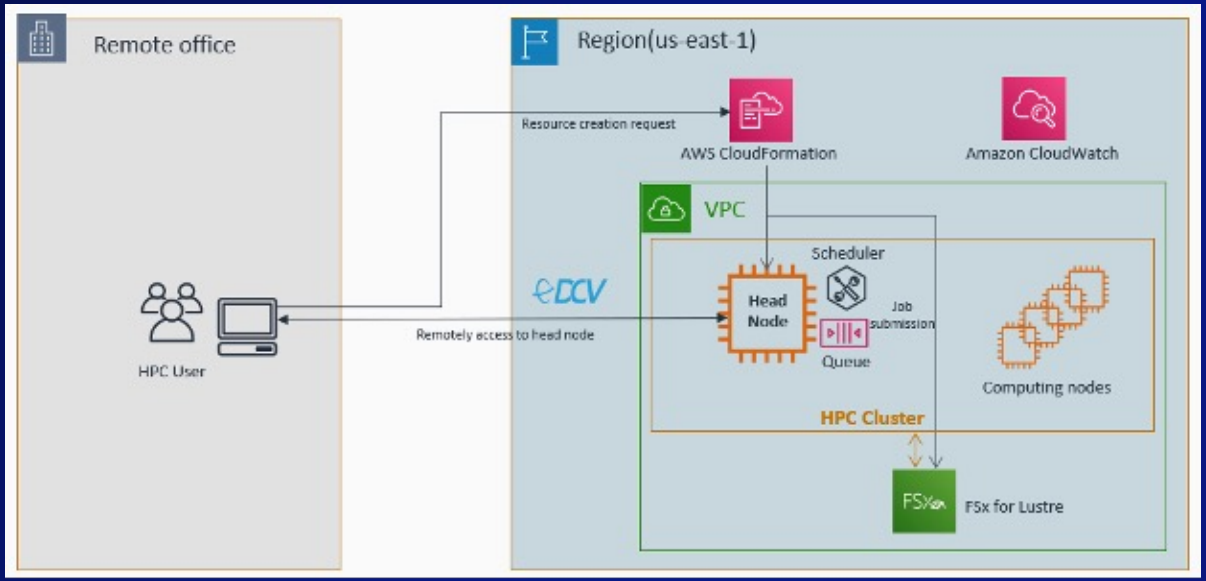
https://www.pointdaily.co.kr/news/articleView.html?idxno=247322

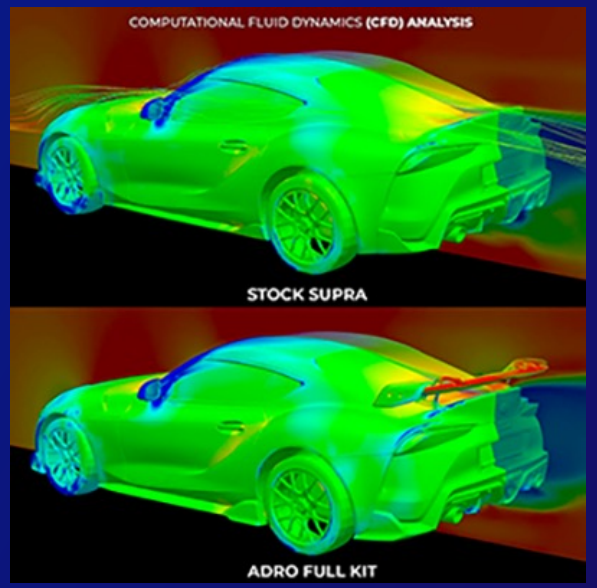
PUBG lobby server graviton adoption



https://www.youtube.com/watch?v=GhYUjTxSOkE (31:45-)

에이드로, Graviton3E 기반의 CFD 시뮬레이션 구축

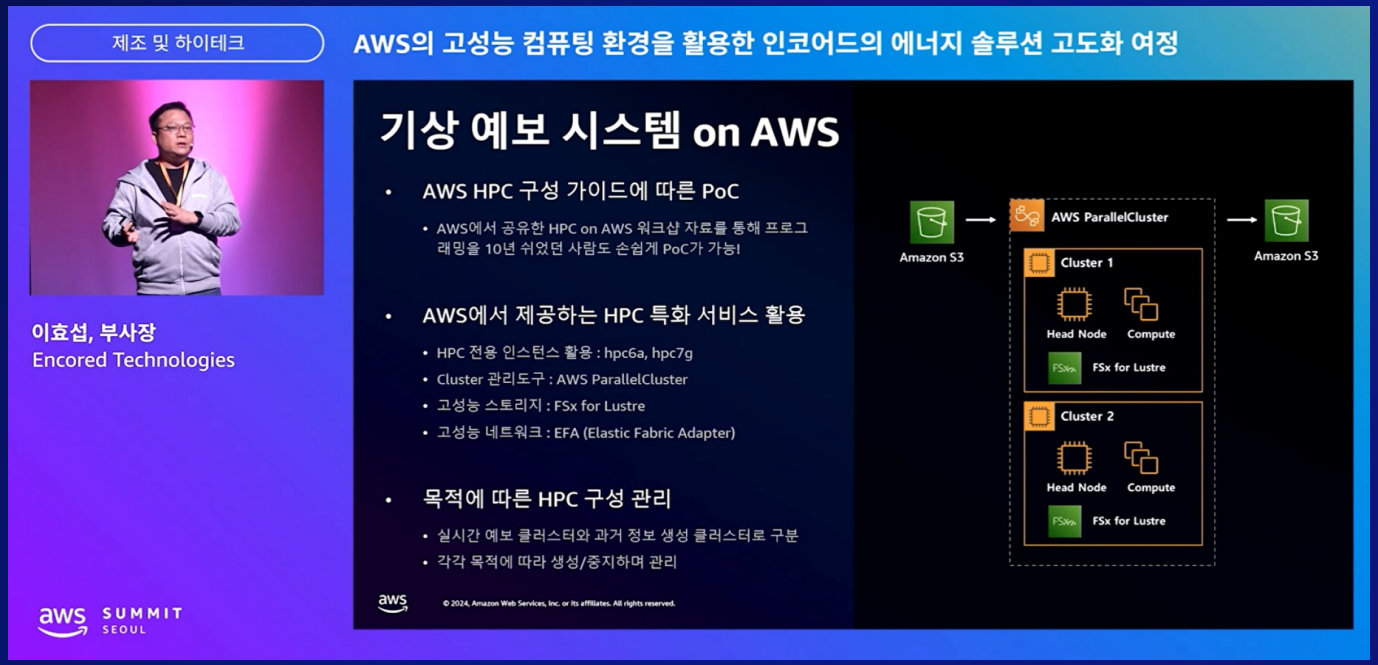




에이드로(ADRO)는 고품질의 바디키트를 제조하는 스타트업으로, 차량의 바디키트에 전달되는 공기의 흐름과 압력분포 등을 시뮬레이션함으로써 성능을 최적화하는 데, 기존의 고사양 PC로는 CFD 시뮬레이션을 실행하는 데 한계가 있었습니다. AWS의 HPC 솔루션을 통해서 정확한 분석을 위해 시뮬레이션당 1,200 코어를 사용하여 빠른 시간 내에 결과를 얻을 수 있고 분석 결과값을 신뢰할 수 있게 되었습니다. (3일이나 걸리던 CFD 시뮬레이션을 3시간으로 단축)

https://aws.amazon.com/ko/solutions/case-studies/adro-case-study/

인코어드, Graviton3E 기반의 기상 예보 시스템 구축



인코어드테크놀로지스는 AWS에서 제공하는 HPC(High Performance Computing, 고성능 컴퓨팅)에 특화된 서버 자원인 Amazon EC2 Hpc6a, Amazon EC2 Hpc7g를 활용해 기상 예보 시스템을 위한 HPC 클러스터를 구축해 신재생에너지 발전소의 발전량을 예측하고 있습니다.

날씨 예측에 필요한 대용량의 데이터는 Amazon S3 와 Amazon FSx for Lustre를 사용해 저장하고, HPC 클러스터 관리 도구인 AWS ParallelCluster를 사용했습니다. 도입초기 Graviton2 활용하였으나, 현재는 ARM 기반의 AWS Graviton3E 프로세서로 구동되는 Amazon EC2 Hpc7g로 변경하면서 비용을 약 60% 절감했습니다.

https://aws.amazon.com/ko/solutions/case-studies/encored-technologies-case-study/

Supporting Prime Day

amazon

2021: 12 core retail services, three-Region cluster composed of over 53,000 Graviton2-based C6g instances

2022: Increased adoption of Graviton2, creating more efficiencies; overall server footprint only 7% larger than Cyber Monday 2021

2023: Tens of millions of normalized Graviton instances, to power over 2,600 services

aws

Resources

- Graviton2 Getting Started Github Repo - Technical details: https://github.com/aws/aws-graviton-getting-started
- Graviton2 workshops: https://graviton2-workshop.workshop.aws/en/gettingstarted.html
- Graviton Developer workshop https://catalog.us-east-1.prod.workshops.aws/workshops/dcab7555-32fc-42d2-97e5-2b7a35cd008f/en-US

Appendix

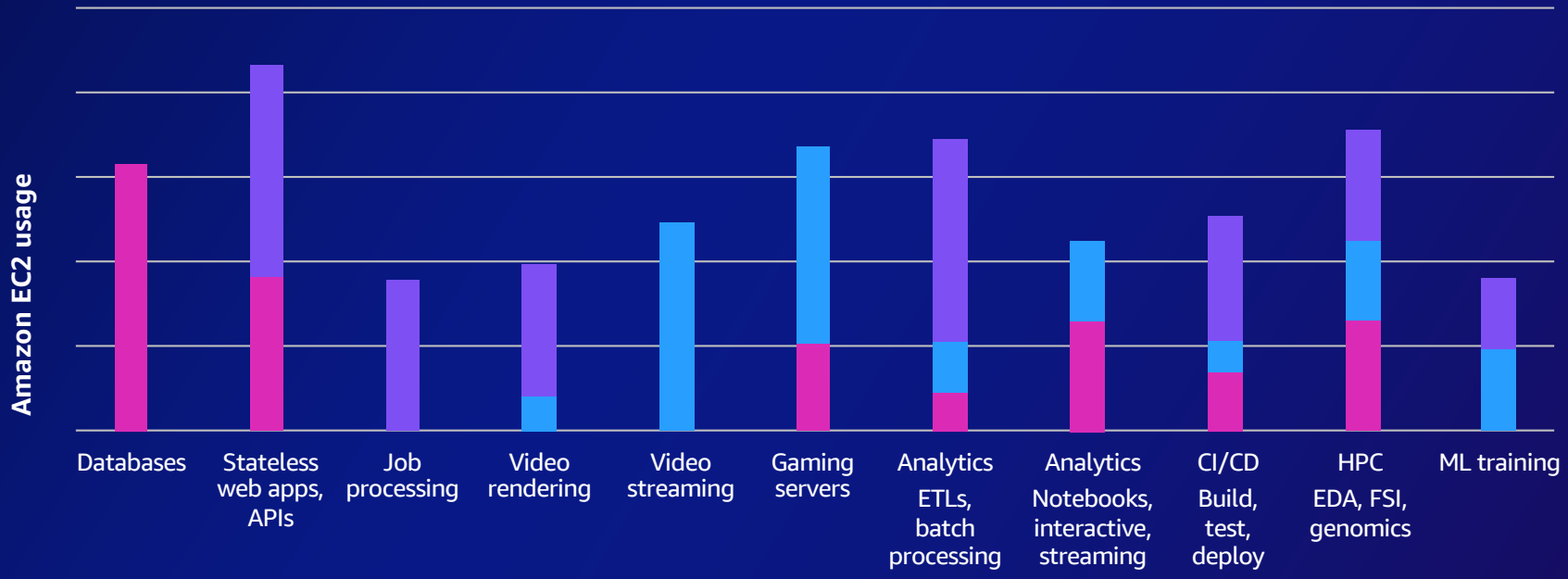
Dive into Deep Learning Compiler

Workload types and purchase options

Use **Savings Plans** for known/steady-state workloads

Scale using **On-Demand** for new or stateful spiky workloads

Scale using **Spot Instances** for flexible, fault-tolerant workloads



Example purchase option distributions

Thank you!