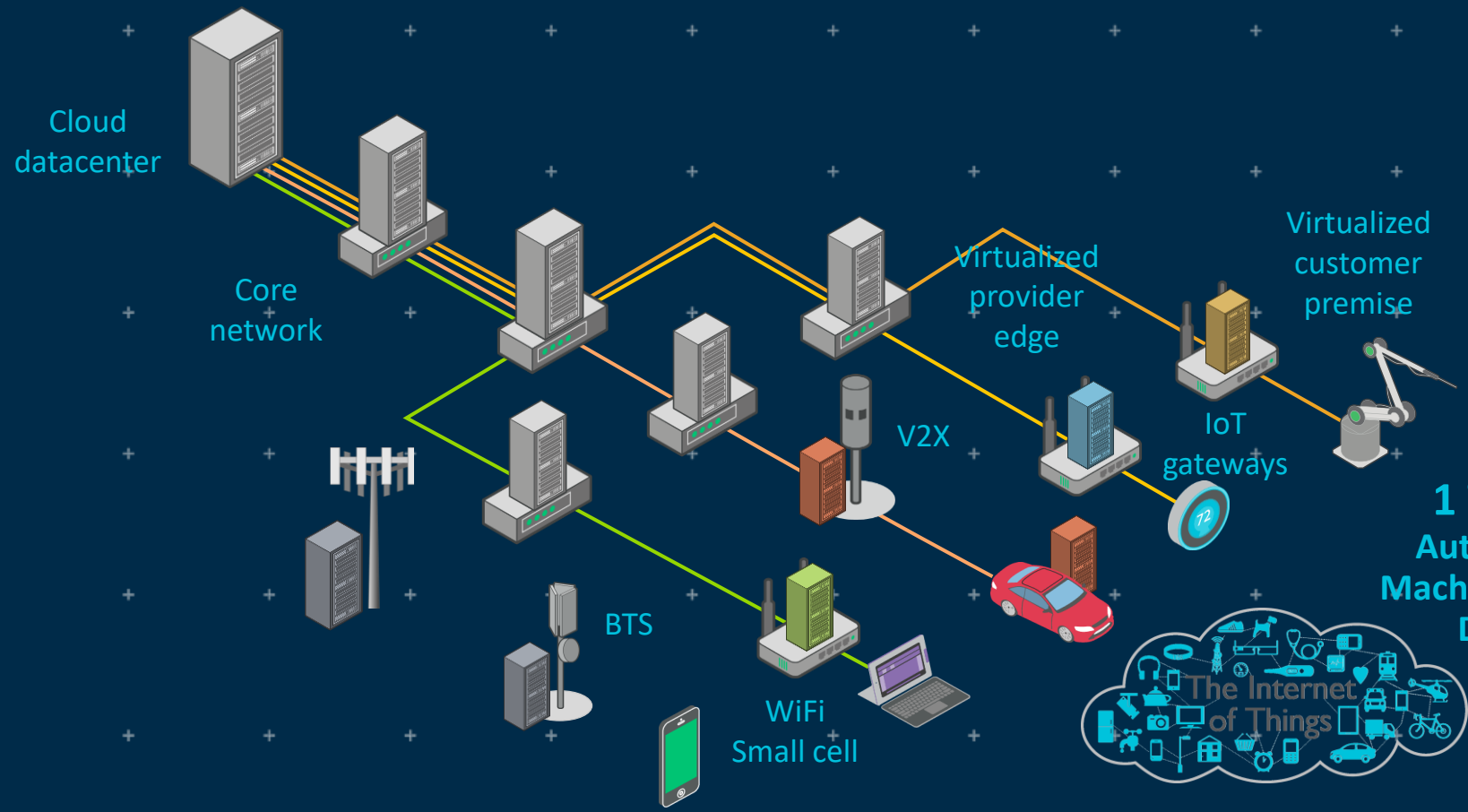
# Methodology and challenges assessing performance of systems with different ISAs

2022/04/01

Andr
Arm, Distinguish

# New infrastructure for support of 1T connected devices




# Arm Neoverse scalable compute platforms

GIC: Generic Interrupt Controller
MMU: Memory Management Unit
* CCIX – Cache Coherent Interconnect (www.ccixconsortium.org)

**Arm CPU**

CPU — Neoverse N1, N2, V1

**Arm SoC Backplane**

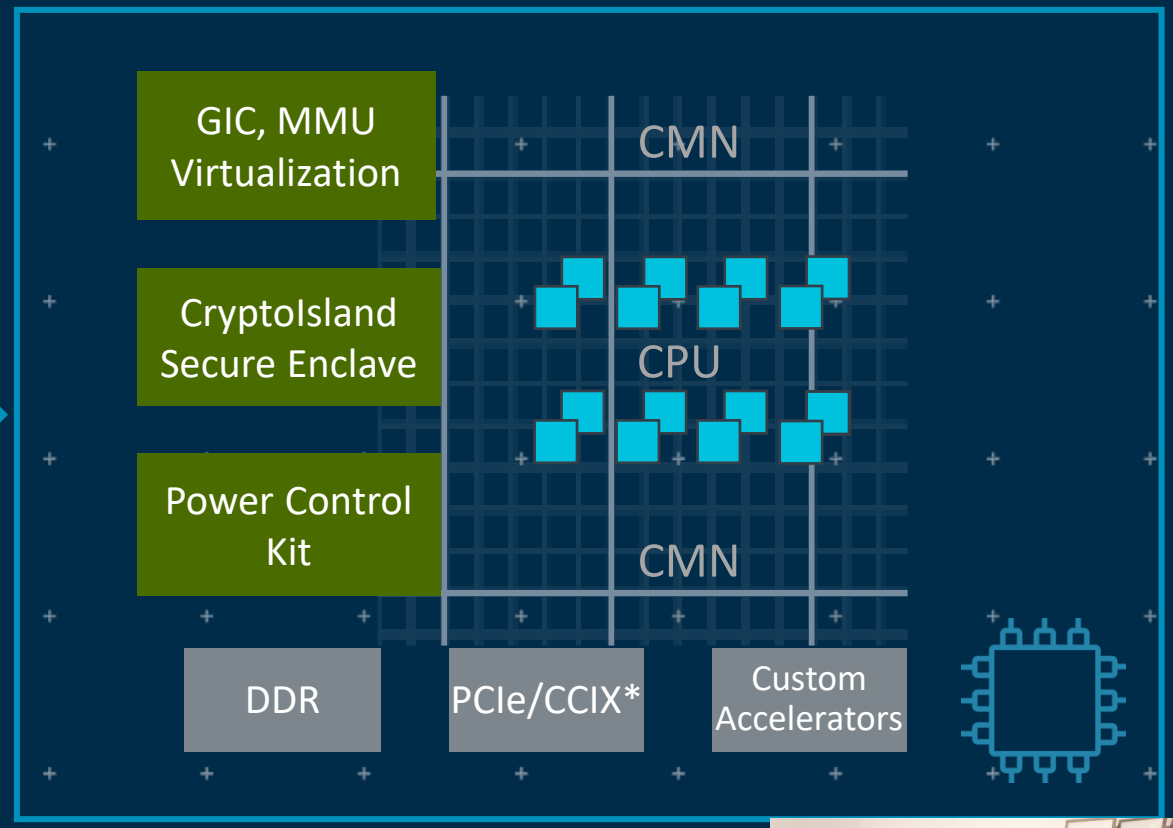
- Coherent Mesh Network (CMN)
- GIC, MMU Virtualization
- Power Control Kit
- CryptoIsland Secure Enclave

**Memory**

- DDR4 DDR5
- HBM

**IO**

- PCIe
- CCIX
- GbE




Common Software Platform, SBSA, SBBR, Arm ServerReady
Arm Architecture v8.x-A, AMBA

*Everything in green and blue boxes provided by Arm*

# Arm Neoverse Platform Roadmap

In planning

| IP Date | Production | Now | | 2022+ |
|---|---|---|---|---|
| **Platform features** | 7nm<br>PCIe Gen4, DDR4, HBM2<br>CCIX 1.0 | 7/5nm<br>PCIe Gen5, DDR5, HBM2e<br>CCIX 1.1 | 5nm<br>PCIe Gen5, DDR5, HBM3<br>CCIX 2.0, CXL 2.0 | 5/3nm<br>PCIe Gen5/6, DDR5, HBM3<br>CCIX next, CXL next |
| **V-series**<br>Max, ST Perf, ML | | **V1 Platform (Zeus)**<br>N1+50% ST perf. uplift<br>SVE 2x256b, bFloat16 | | **Poseidon Generation Platforms**<br>+30% infra WL perf.<br>ML/Vector uplift<br>Greater core density |
| **N-series**<br>Perf/W, Scale out, 5G | **N1 Platform**<br>A72+60% ST perf. uplift | | **N2 Platform (Perseus)**<br>N1+40% ST perf. uplift<br>SVE2 2x128b, bFloat16 | |
| **E-series**<br>Data efficiency | **E1 Platform**<br>A53+2.7x throughput | | | |

# Neoverse N1 Performance Leadership

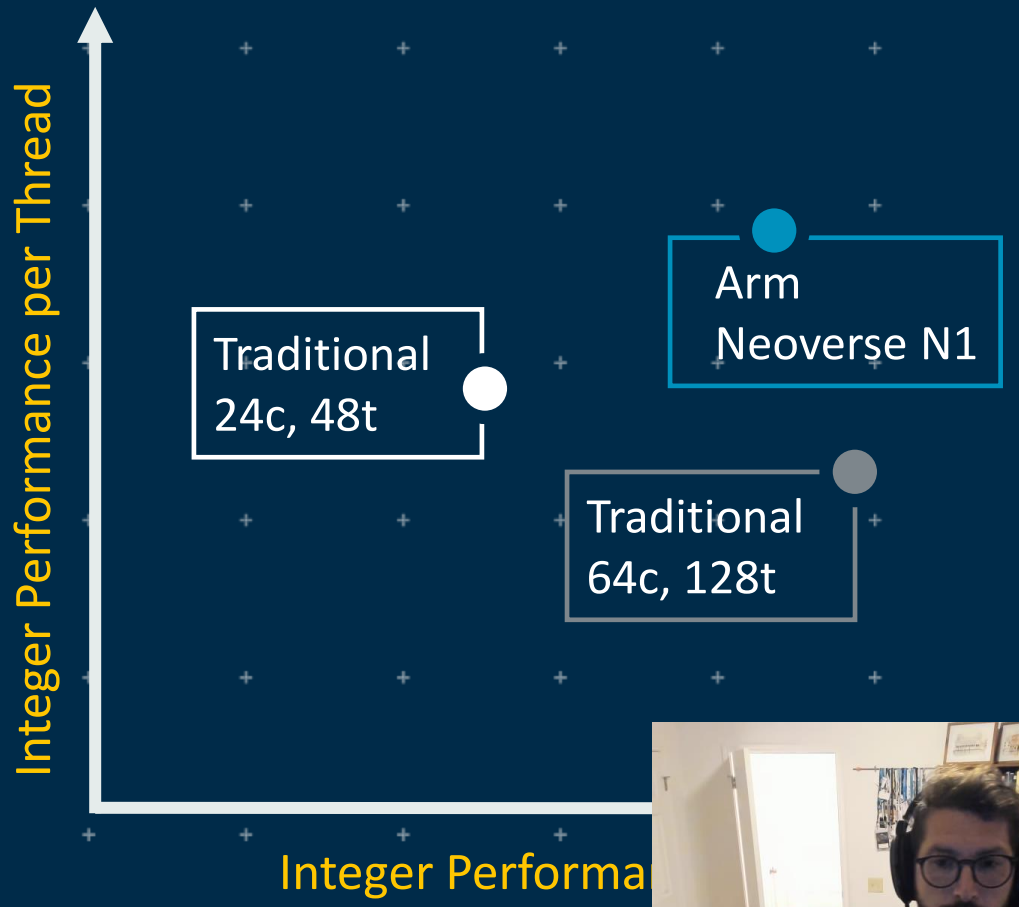
- Architected to deliver the best performance to the end user

Customers on Neoverse N1 already see significant benefits on many applications:

- **>1.4x** on general compute
- **> 1.26x** on x264 media encoding
- **>1.5x** on EDA and simulation tools
- **>1.5x** on web servers and reverse proxies
- **>1.65x** key value databases
- **130,899** max-JOps on SPECjbb2015®, single socket

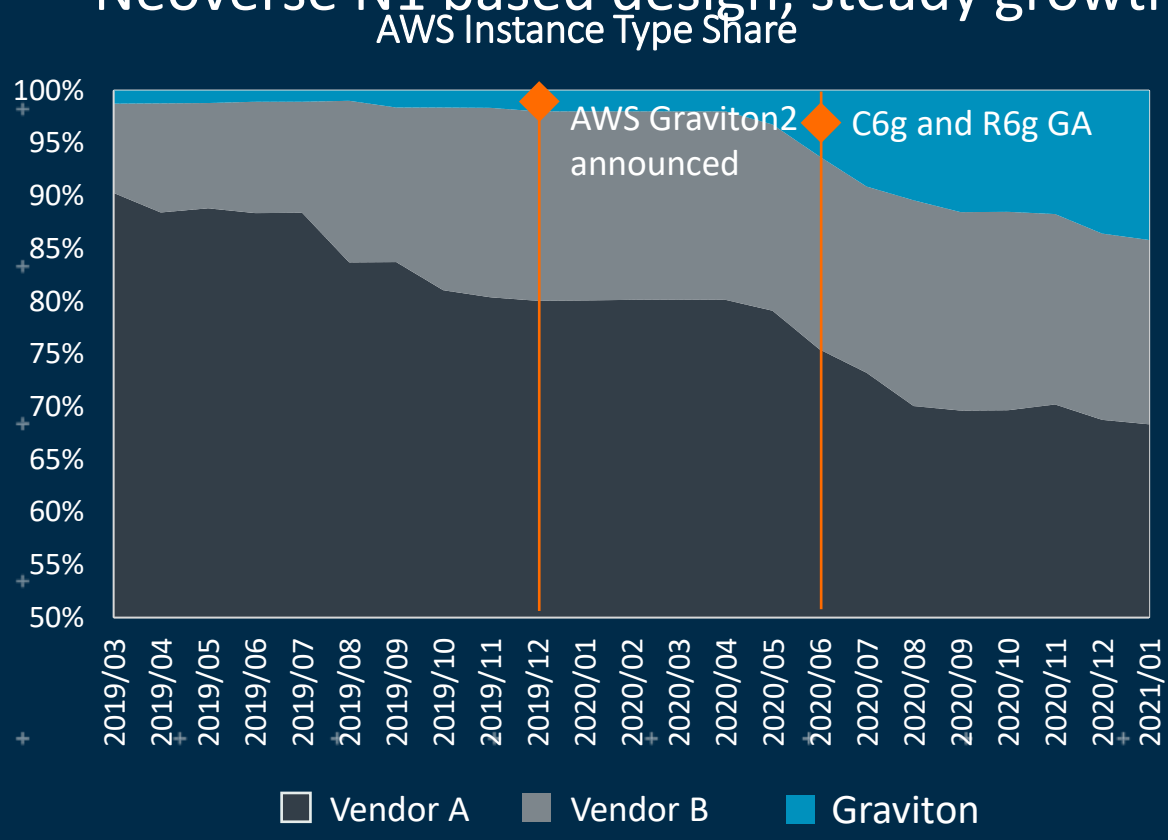
https://www.alibabacloud.com/blog/alibaba-dragonwell-powers-java-applications-in-alibaba-cloud_597564

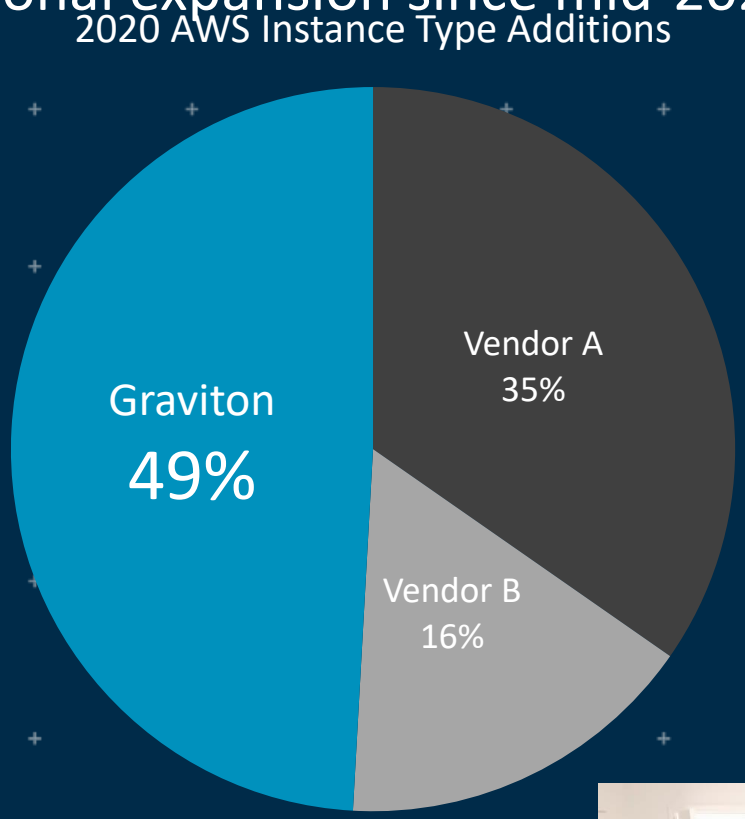
*Performance measurements collected by Arm and its partners on similarly configured, generally available, Neoverse N1 and traditional cloud instances*




# AWS Graviton2 is Quickly Expanding its EC2 Footprint

- Neoverse N1 based design, steady growth and regional expansion since mid-2020




AWS Graviton2-based EC2 instances make up 14% of the installed base within AWS

49% of AWS EC2 instance additions based on AWS Graviton2

*Source: Liftr Insights*

# Processor Performance Equation

$$\text{Perf of machine B vs A} = \frac{\text{time performed to accomplish task on machine A}}{\text{time performed to accomplish task on machine B}}$$

$$\text{time performed to accomplish work on machine} = \frac{\text{Instructions}}{\text{Task}} \times \frac{\text{Clock cycles}}{\text{Instructions}} \times \frac{\text{Seconds}}{\text{Clock cycles}} = \frac{\text{Seconds}}{\text{Task}}$$

SpecCPU scores match these equations, but datacenter workloads are a bit tricker…
…will discuss them later.

# ISA and task 1/2

- Arm is very RISC
  - All instructions are 32 bits, always go through registers
  - Load and store architecture – no memory operations (i.e. there is no "`add <mem>,<reg>`" instruction in Arm)
  - Instruction count might be higher than for x86, does not mean performance is lower!
    - A closer count is x86 uops vs Arm instructions
    - Arm implementations can fuse or split instructions (current Arm Neoverse cores only fuse, no splits)
- Arm implementations might vary from each other
  - Definitely true in terms of performance
  - SW compatibility is guaranteed (if they have the same ISA version)
- Arm has a relaxed memory model
  - SW targeting Arm requires explicit memory barriers to guarantee observability of memory operations (unlike x86, which implements Total Store Ordering)
  - This mostly matters for complex SW libraries and frameworks, but it's worth keeping
- Sometimes optimizing for code size (i.e. "`-Os`" on gcc) also improves perf

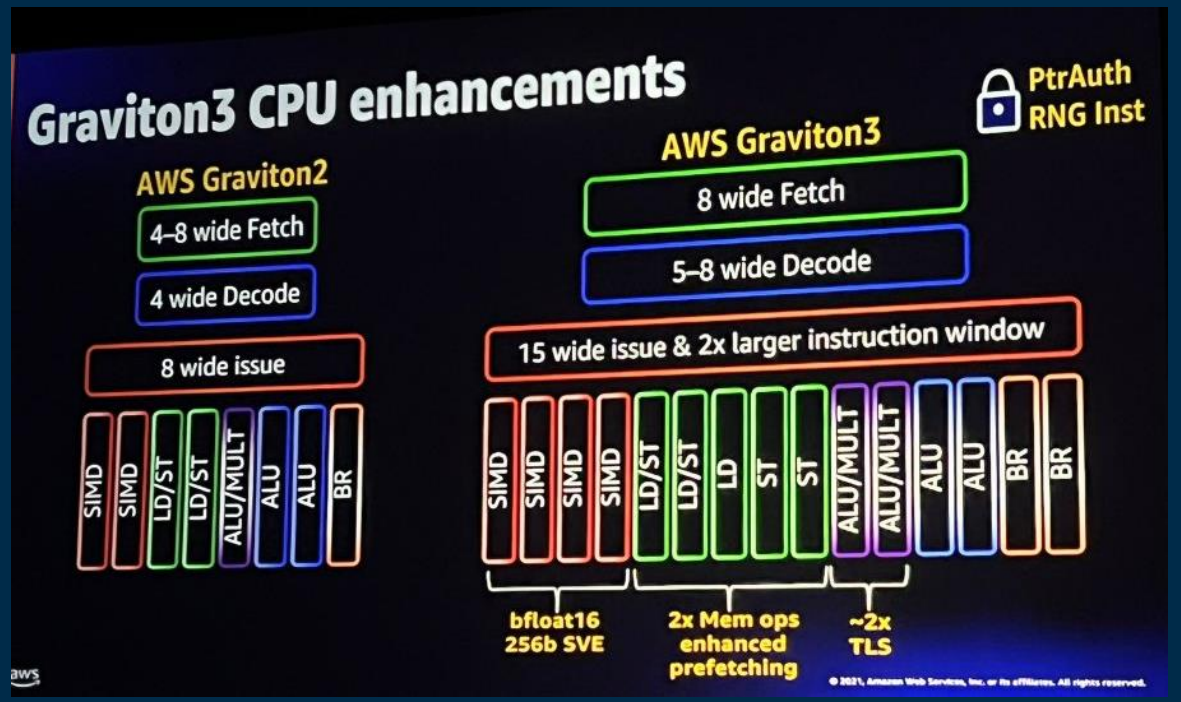
# ISA and task 2/2

- Make sure the same to understand if the right features are enabled or what are the characteristics of the machines compared, for instance:
  - Arm introduced atomic operations (i.e. Compare and Swap) in v8.2
    - Often precompiled SW targets v8.0 (i.e. Raspberry Pi 4) and uses load-exclusive/store-conditional pairs, which is lower performance for high core count systems.
    - Recompile with a new compiler to enable atomics (in gcc9, target neoverse-n1, v8.2 or add `-moutline-atomics`)
  - Neoverse vector units (Neon or SVE) are *typically* narrower than AVX512
    - SVE is vector length agnostic: code once, the machine will figure out how many times it needs to execute the loop based on how wide the vector units are in HW *(from 128 up 2048bits)*
    - Neon units are 128 bits, but systems with larger vector SVE units can also run more Neon instructions in parallel (i.e. Neoverse V1 can un 2x256 SVE or 4x128 Neon instructions in one cycle)

# CPI

$$\frac{\text{Clock cycles}}{\text{Instructions}}$$

- Current Arm implementations are quite wide
  - Front-end is often wider than the generationally equivalent x86 cores
- Larger L1 caches (VIPT vs PIPT)
- Widening the cores is often more energy efficient than increasing frequency
- How to account for SMT?
  - Increases core IPC (1/CPI) for tasks that are parallelizable and are waiting for high latency memory accesses
  - Arm Neoverse N1, N2, and V1 do not support SMT: each SW context running gets a dedicated core, L1 and L2 caches




# Frequency – and Turbo

Seconds / Clock cycles

$$P = ½ * \alpha * Cdyn * f * V^2$$

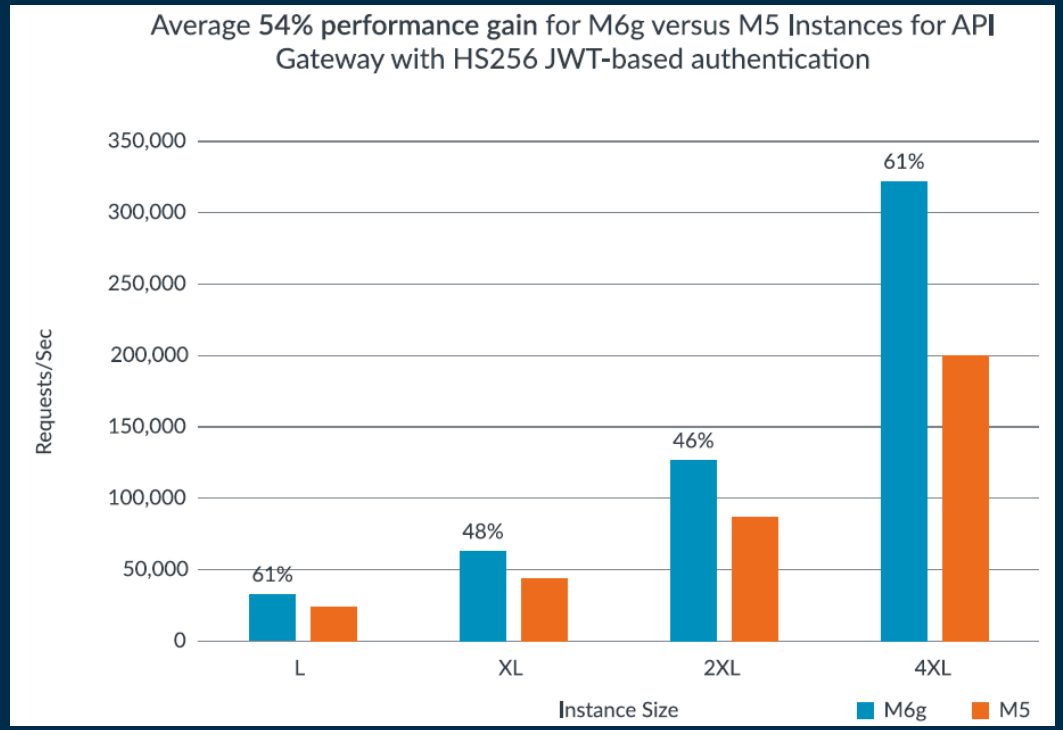
- Power consumption and cooling for one SoC keeps growing into the 350W/400W envelope – TDP determines core count and frequency

- Arm Neoverse N1, N2, and V1 are designed to go up to 3.5GHz… however, but most of our partners don't Turbo as high as our competitors:
  - Enables power efficient at sustained frequency
  - Avoids frequency jitter due to other noisy neighbors or frequency hit due to utilization of certain classes of instructions
  - Turbo capabilities can really help certain classes of workloads (i.e. aggregating data co… a map reduce query in order to compose a response web page)

# Performance in the real world

## Often the metric that matters is throughput under SLA

### Nginx




Up to **54% performance** gains on average per instance

Up to **20% cost savings** for scale-out deployments

- How many requests can be serviced by the system within a certain SLA (latency boundary)?
- Why does SLA matter?
  - Often need to perform Map-reduce tasks, which latency is bound by the slowest request coming back.
  - Cannot degrade performance as the workload increases:
    - Scale up the deployment and shard the application
- When comparing how systems spend their time on workloads like these try to n
  amount of work done – what is
  sustainable for the systems und

# Conclusions

- Within the same generation of datacenter processors, systems are very different from each other, and will likely continue to diverge:
  - System architecture (cache size and architecture, core count, frequencies, SMT, ISA)
  - Accelerators, disaggregated memory, …
- The rulebook still applies – but focus on:
  - The task performed, and how the machine is used and what matters to the user
  - If you are studying scaling or features in details, normalize what you can when you can (including the workload, if possible)
  - Tune the systems to their best capabilities if you are looking to assess overall performance

- You can try Arm Neoverse N1-based machines **today** on:

AWS: https://aws.amazon.com/ec2/graviton/

Oracle OCI: https://www.oracle.com/cloud/compute/arm

Alibaba cloud: 云服务器ECS ARM实例规格族邀测 (aliyun.co

arm

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה