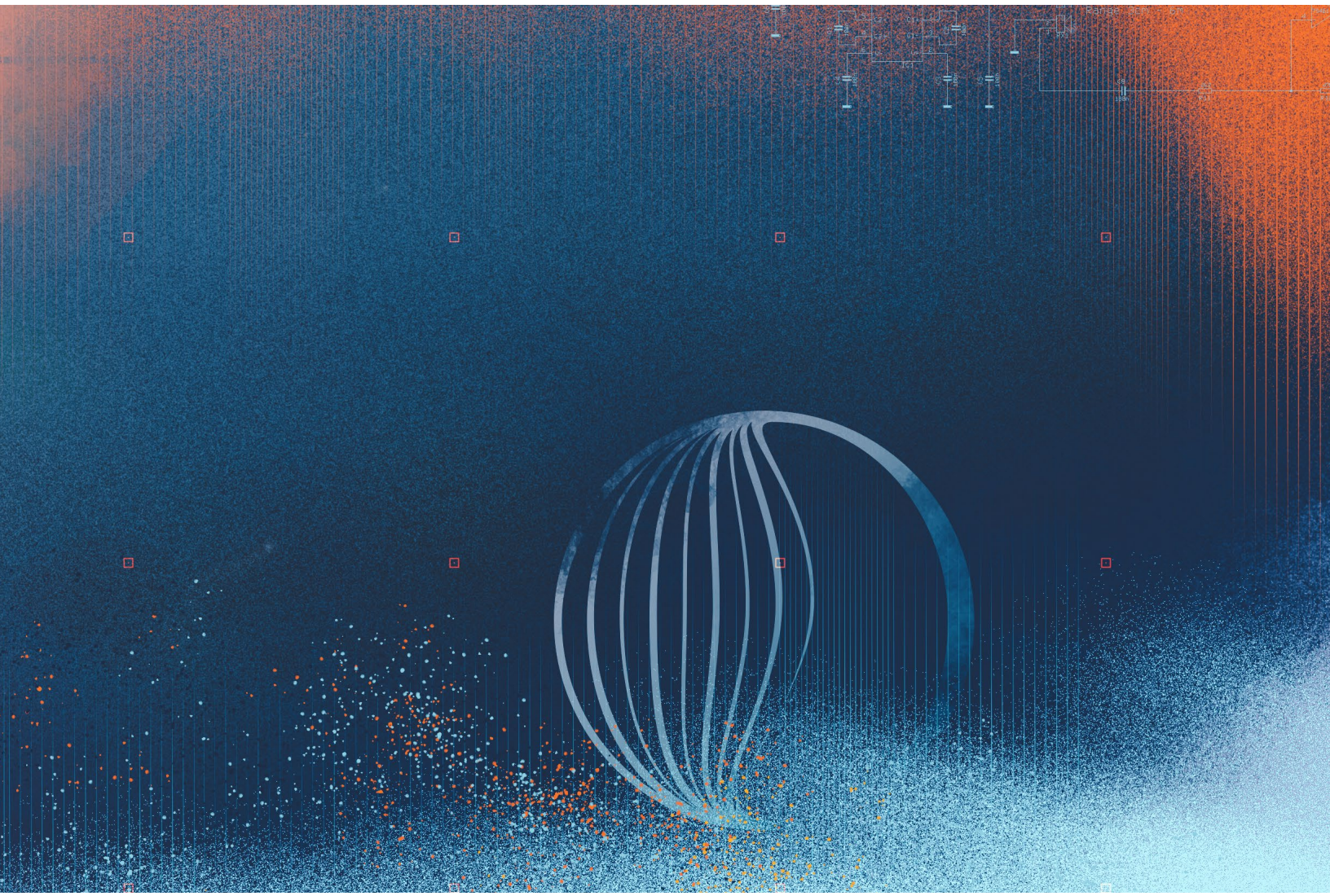
signal65



SIGNAL65 LAB INSIGHT

Arm Neoverse Enables Leading Cloud Performance and Cost Efficiency with AWS Graviton4

AUTHORS

Mitch Lewis
Performance Analyst | Signal65

Jonathan Fellows
Performance Validation Engineer | Signal65

IN PARTNERSHIP WITH

arm

AUGUST 2025

Executive Summary

Modern IT organizations are responsible for supporting a wide range of complex workloads, from traditional web and database workloads, to emerging AI and machine learning applications. Yet with this compute diversity comes an infrastructure challenge: balancing the high performance that applications demand with the cost constraints of operating at scale. Increasingly, enterprise IT organizations are turning to cloud services to meet these demands.

Major cloud service providers, such as AWS, Microsoft Azure, and Google Cloud, offer organizations flexibility and access to a wide range of compute resources to meet the diverse requirements of modern workloads in the cloud. CPUs based on Arm's Neoverse architecture, including AWS Graviton, Microsoft Azure Cobalt, and Google Cloud Axion, have emerged as compelling options for supporting key enterprise workloads. In fact, according to AWS, over the last two years, more than 50% of new CPU capacity added by AWS has been powered by Arm-based Graviton processors .

This report explores the performance and cost-efficiency benefits of Arm processors on AWS, specifically examining Arm Neoverse-powered AWS Graviton4 processors in comparison to the latest available generation AMD and Intel based AWS EC2 alternatives. As detailed in this Lab Insight Report, Signal65 conducted hands on performance testing and cost efficiency analysis across four distinct workloads to represent real world data center and AI scenarios and to showcase the broad performance and economic advantages of Arm Neoverse.

Across all workloads tested, Arm-Neoverse based Graviton4 instances consistently delivered higher performance and better price/performance than both AMD and Intel.

Key takeaways from testing:

Generative AI - Llama-3.1-8B

Arm Performance Advantage:

168% vs. AMD

162% vs. Intel

Arm Price/Performance Advantage:

220% vs. AMD

195% vs. Intel

Arm c8g.16xlarge, AMD c7a.16xlarge, Intel c7i.16xlarge

Machine Learning - XGBoost

Arm Performance Advantage:

53% vs. AMD

34% vs. Intel

Arm Price/Performance Advantage:

64% vs. AMD

49% vs. Intel

Arm r8g.16xlarge, AMD r7a.16xlarge, Intel r7i.16xlarge

Database - Redis

Arm Performance Advantage:

93% vs. AMD

41% vs. Intel

Arm Price/Performance Advantage:

149% vs. AMD

59% vs. Intel

Arm r8g.2xlarge, AMD r7a.2xlarge, Intel r7i.2xlarge

Networking - Nginx

Arm Performance Advantage:

43% vs. AMD

53% vs. Intel

Arm Price/Performance Advantage:

81% vs. AMD

73% vs. Intel

Arm m8g.2xlarge, AMD m7a.2xlarge, Intel m7i.2xlarge

[1] https://www.youtube.com/watch?v=vx36tyJ47ps&t=1041s

Cloud Infrastructure Considerations: Performance and Cost Efficiency

For many organizations, cloud resources now represent a substantial share of their overall IT infrastructure footprint. On major cloud platforms such as AWS, x86 architecture has been the only available option for years. However, cloud scale economics demand cutting edge performance, power efficiency, and a software ecosystem that's modern, agile and interoperable. Ongoing innovation from Arm, particularly its Neoverse family of CPUs, has challenged the x86 status quo, beginning with AWS Graviton in 2018.

For IT organizations leveraging the public cloud, selecting the right compute platform means finding an optimal balance between performance and cost efficiency for their workloads. On AWS, Arm Neoverse-based Graviton4 processors, make a strong case as the ideal platform to balance these requirements. In this analysis, Arm-based Graviton4 instances didn't just outperform AMD and Intel on key workloads, but they did so at a much better price-performance ratio. This testing shows that Arm is well positioned to become a leader in CPUs for cloud computing workloads.

Testing Overview

To evaluate the performance of AWS Graviton4 instances, Signal65 was asked to conduct testing across a range of real-world workload scenarios using AWS instances powered by a variety of CPUs. Signal65 evaluated the performance of Arm Neoverse-based AWS Graviton4 processors versus the latest available generation AMD and Intel EC2 instances. Testing spanned both AI/ML workloads, including inferencing on different Llama models and XGBoost inferencing and training, as well as general compute workloads on Redis databases and Nginx servers.

These four workloads were selected to represent a broad set of applications and to highlight how CPU architecture impacts both performance and efficiency. By testing diverse, commonly deployed workloads, this study presents a broad overview of Arm's competitive performance in the marketplace.

While workload performance is a critical aspect of selecting a cloud compute platform, for modern IT organizations, often operating on strict budgets, performance alone is not sufficient. In addition to performance benchmarking, Signal65 also conducted a comprehensive cost analysis of each workload to evaluate the price-performance of each platform, a critical metric for organizations looking to optimize infrastructure investments at scale.

General Purpose Workloads: Web Server and In-memory Database

General purpose workloads will continue to play a critical role in operations even as emerging AL/ML workloads gain momentum. Among these, Redis and Nginx represent two of the most widely deployed components in cloud-native and enterprise architectures. These are frontline workloads in nearly every modern application stack from streaming platforms to e-commerce, financial services, SaaS, and beyond. By benchmarking Redis and Nginx, this study evaluates how well each CPU platform handles real-world, latency-sensitive workloads that demand both speed and scalability. This testing shows that Arm outperforms both AMD and Intel across these general compute workloads.

Redis

Redis in-memory databases are widely used to meet the requirements of applications that need low latency storage, such as web applications. Having CPUs that can keep up with the high demands of such applications is key to ensuring they run at the most optimal level.

Using the memtier_benchmark tool[2], Signal65 tested r.2xlarge instance types to evaluate the impact that CPUs have on Redis database performance. One instance was set up as the Redis database, and another to generate the workload. Look in the appendix for additional details on the setup and testing, as well as a table with results summarized.

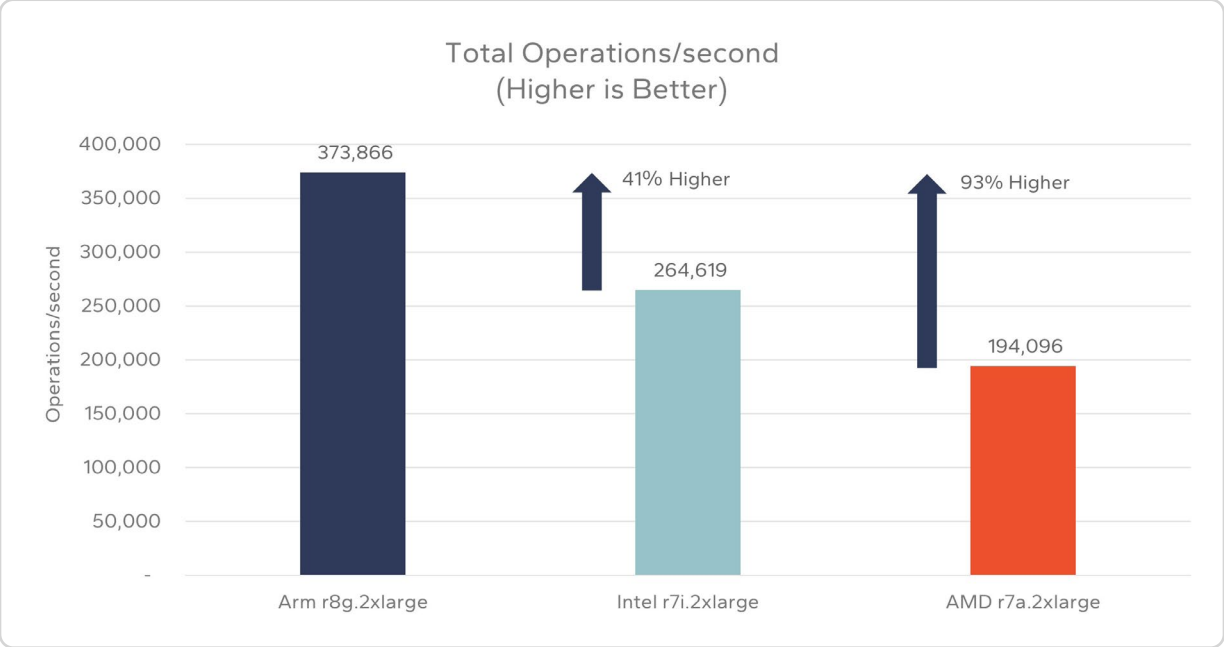


Figure 1: *Redis Operations/Second with Arm Improvement %*

Arm r8g instances performed significantly better in total operations per second than both Intel r7i (41% higher) and AMD r7a (93% higher) instances.

[2] https://github.com/RedisLabs/memtier_benchmark

With latency a key metric of how a Redis database performs, the lowest latency is the most desirable outcome.

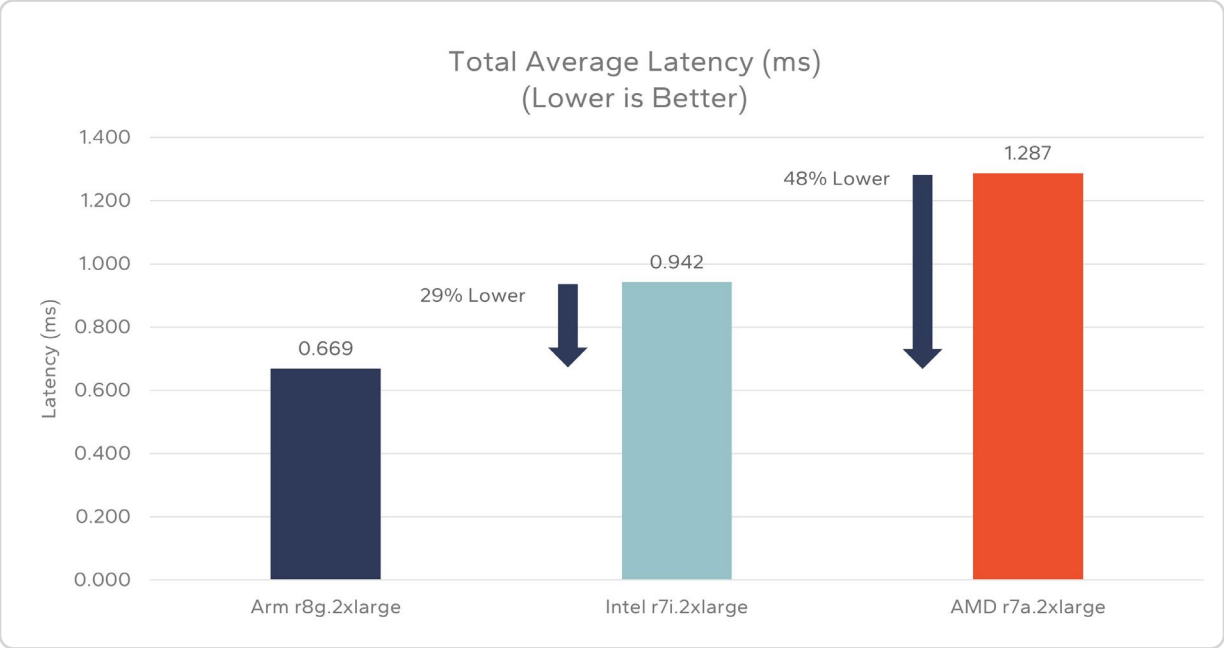


Figure 2: *Redis Average Latency*

Arm CPUs provided 29% lower latency than Intel and 48% lower latency than AMD CPUs.

While performance is an important metric to consider when making a choice for what CPUs to use in an instance, so too is the cost of running those instances. Arm starts out with a significant price advantage vs Intel and AMD as seen in the table below.

Instance Type	On Demand Hourly Price
Arm r8g.2xlarge	$0.47128
Intel r7i.2xlarge	$0.52920
AMD r7a.2xlarge	$0.60860

Figure 3: *Redis Instance Pricing (Source: AWS[3])*

While the price is lower, the performance/$ is the key metric to consider in these scenarios.

[3] Pricing data sourced from AWS On-Demand EC2 plans as of June 10, 2025. All pricing uses US East(Ohio) region.

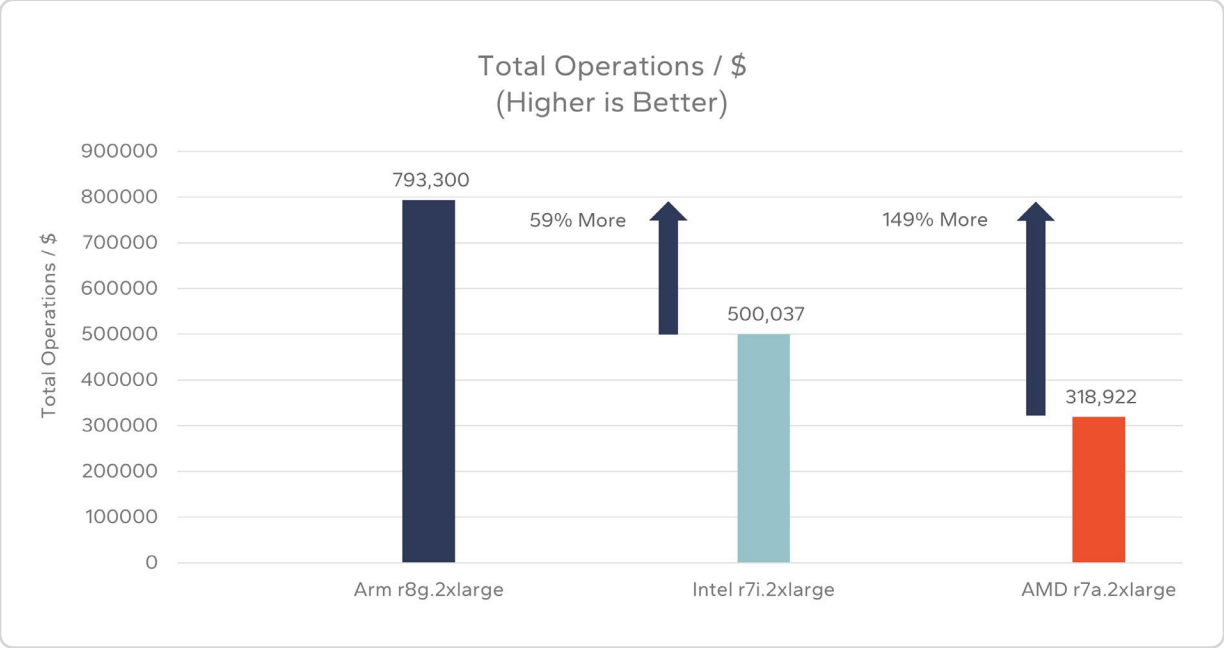


Figure 4: *Redis Operations per Dollar*

As can be seen above, the performance/$ ratio puts Arm in an even better position with 59% more ops/$ than Intel and 149% more ops/$ than AMD.

Signal65 Comment: *Overall Arm significantly outperformed Intel and especially AMD in the Redis memtier_ benchmark workload. These performance gains are quite impressive and not something that is commonly seen when comparing competitors' products. Having even a 10% performance cost advantage is enough to make many companies choose one product over the other, here we see 59% vs Intel and 149% vs AMD.*

Nginx

Nginx continues to be a very popular choice for scale-out web application servers. For organizations that need high performance for their web servers, several types of instances are necessary to achieve that goal. A load balancer is an important piece of this setup, and one that significantly benefits from optimal compute performance. Signal65 tested both m.xlarge and m.2xlarge instances for this workload, and found Arm provided superior performance at a lower cost than both AMD and Intel instances.

To show the benefit of CPUs in the load balancer instance, six instances were used for file servers and one instance was used as a workload generator using the workload generator tool wrk2[4]. Each file server instance hosted a 1KB file that the load balancer would receive over a secure connection using SSL certificates. To keep things consistent, the same instance types were used for the workload generator and file server instances across all tests leaving only the load balancer instance as the one that changed. See the appendix for more details on the set up and workload, as well as a table with results summarized.

Arm showed significant a performance lead versus AMD and Intel with both the m8g.xlarge and m8g.2xlarge compared to m7a and m7i instance types.

[4] https://github.com/giltene/wrk2

As seen in the chart below, using m.xlarge instance types Arm achieved a 34% and 40% performance gain vs AMD and Intel respectively. Similarly, Arm saw a 43% and 53% gain vs AMD and Intel with the m.2xlarge instance types.

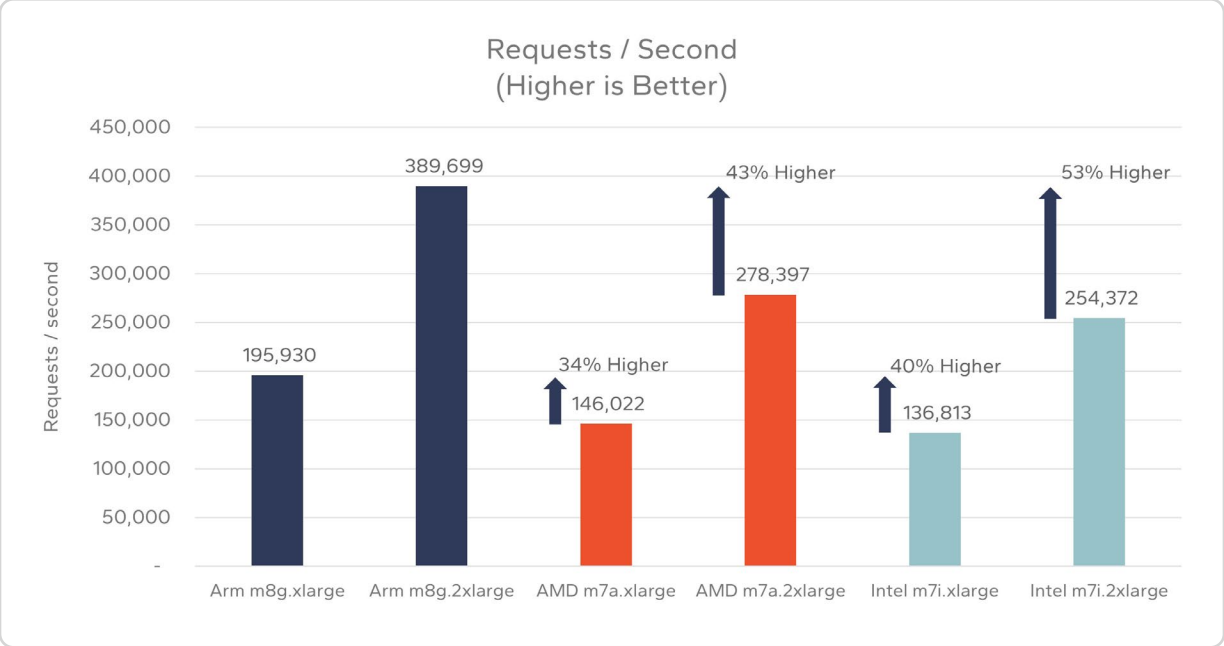


Figure 5: *Nginx Requests per Second with Arm Improvement %*

Another important concept to consider is performance scaling with increasing instance size. By doubling the instance size from xlarge to 2xlarge, ideally the performance gain would be 2x. In the chart below we can see that the Arm instance essentially scaled by 2x, coming in at a 1.99x scale factor. AMD and Intel instances did not scale as well however, with a 1.91x scale factor for AMD and 1.86x for Intel.

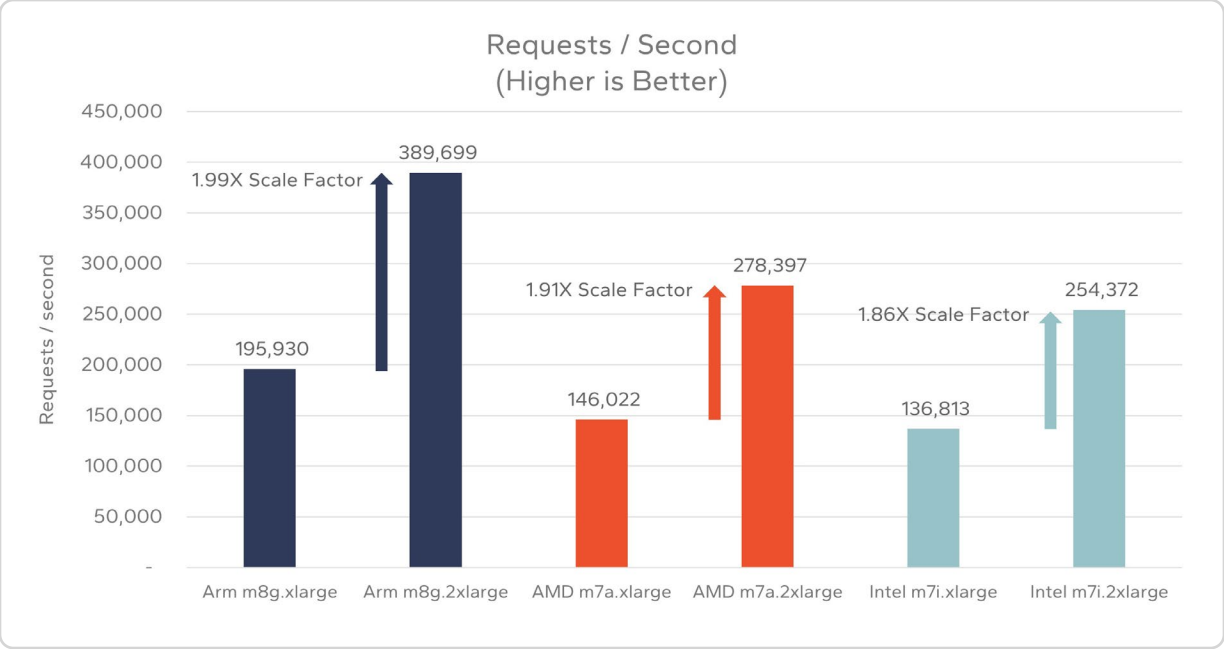


Figure 6: *Nginx Requests per Second with Scale Factor*

As discussed previously, Arm has lower prices for its instances vs AMD and Intel. This alone makes a compelling argument for choosing Arm instances.

Instance Type	On Demand Hourly Price
Arm m8g.xlarge	$0.17952
Arm m8g.2xlarge	$0.35904
AMD m7a.xlarge	$0.23184
AMD m7a.2xlarge	$0.46368
Intel m7i.xlarge	$0.20160
Intel m7i.2xlarge	$0.40320

Figure 7: *Nginx Instance Pricing (Source: AWS[5])*

Impressively, the performance gains for Arm did not come at the expense of cost. Arm has a lower price/performance ratio than both AMD and Intel by significant margins.

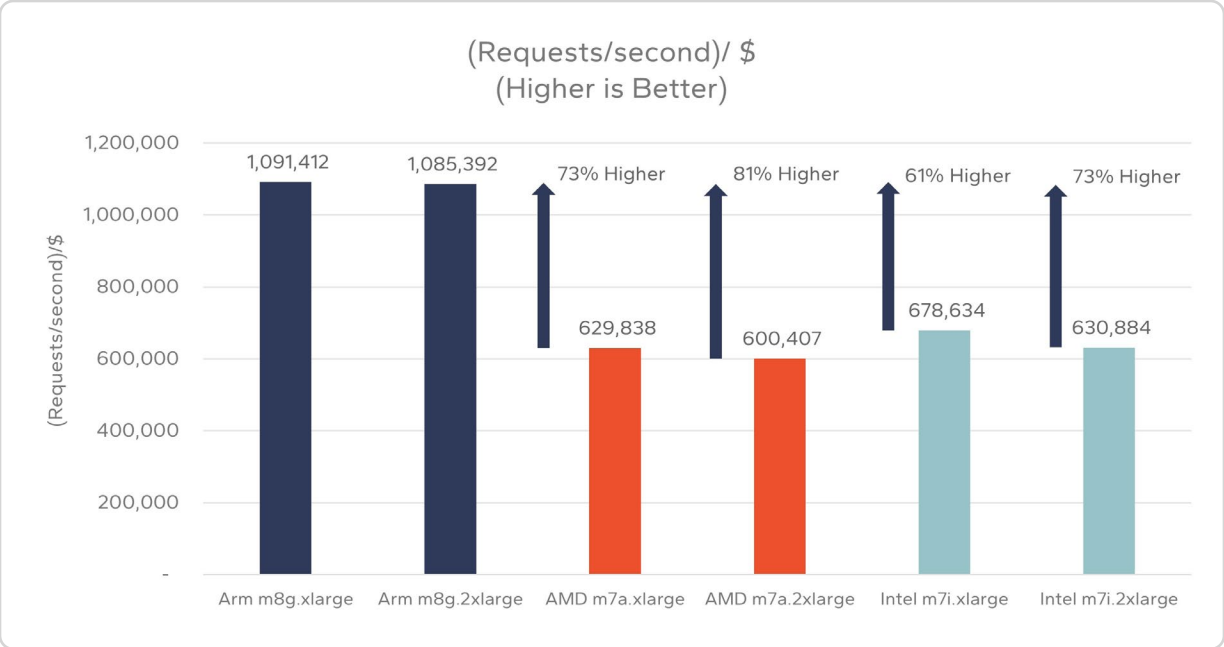


Figure 8: *Nginx Price Performance*

Arm has a 73% and 61% better perf/$ than AMD and Intel with the m.xlarge instance types, and 81% and 73% better perf/$ with the m.2xlarge instance types.

[5] Pricing data sourced from AWS On-Demand EC2 plans as of June 10, 2025. All pricing uses US East(Ohio) region.

Running the Nginx workload showcases how Arm can perform better for a much lower cost than either AMD or Intel. Based on our testing, Arm based instances will also deliver high performance whether they are xlarge or 2xlarge size, unlike the competitors where the performance of the 2xlarge instances does not scale by a factor of 2 compared to the xlarge size. For organizations that rely on fast web applications such as Nginx servers, Arm-based AWS Graviton4 CPUs are the clear choice for instance type.

Signal65 Comment: *When it comes to optimal web server applications, such as Nginx, fast performance is key. Arm has shown it outperforms its competitors by very large margins, which is again uncommon. The performance cost ratio being even better for Arm make these Nginx results all the more impressive.*

AI and ML on Arm Neoverse

When considering AI and ML workloads, much of the focus over the past few years has gravitated towards generative AI and large language models. While generative AI represents a promising avenue for new AI innovation, many traditional machine learning algorithms are still core enterprise workloads. Unlike large-scale generative AI model training, which is often reliant on large GPU clusters, traditional ML workloads are typically well suited for CPUs. Similarly, CPUs can be leveraged for many generative AI inferencing use cases.

As generative AI workloads grow in importance, organizations will need to make strategic decisions about where to spend their resources. AI workloads using GPUs can be resource intensive and quickly accumulate high costs in the cloud. To remain cost effective, organizations must evaluate their AI/ML workloads to understand when to leverage GPUs, and when they can instead optimize price and performance with CPUs.

To evaluate the performance of traditional machine learning workloads on Arm-based AWS Graviton4 CPUs, Signal65 conducted performance tests for XGBoost training and inferencing. To further explore how Arm Neoverse technology can be leveraged for emerging generative AI use cases, additional AI inference testing was performed using different Llama large language models and input/output scenarios based on real-world use cases. Across every scenario tested, Arm-based Graviton4 instances consistently outperformed AMD and Intel while also delivering significantly better cost-efficiency. These results clearly position Arm as a leading CPU option for running AI and ML workloads in the cloud.

XGBoost

XGBoost (eXtreme Gradient Boosting) is a popular machine learning algorithm that is widely used for both regression and classification tasks. Performance testing of XGBoost was completed for both training and inferencing workloads for a classification problem using the Higgs dataset[6]. Testing was completed on the following AWS EC2 instances:

- r8g.16xlarge (Arm)
- r7a.16xlarge (AMD)
- r7i.16xlarge (Intel)

[6] https://archive.ics.uci.edu/dataset/280/higgs

Testing was achieved using the xgboost-bench[7] benchmark to measure the average time to complete a series of 15 training or inferencing runs. To examine the performance with different dataset sizes, tests were executed with various subsets of the Higgs dataset, including between 10k and 10M instances.

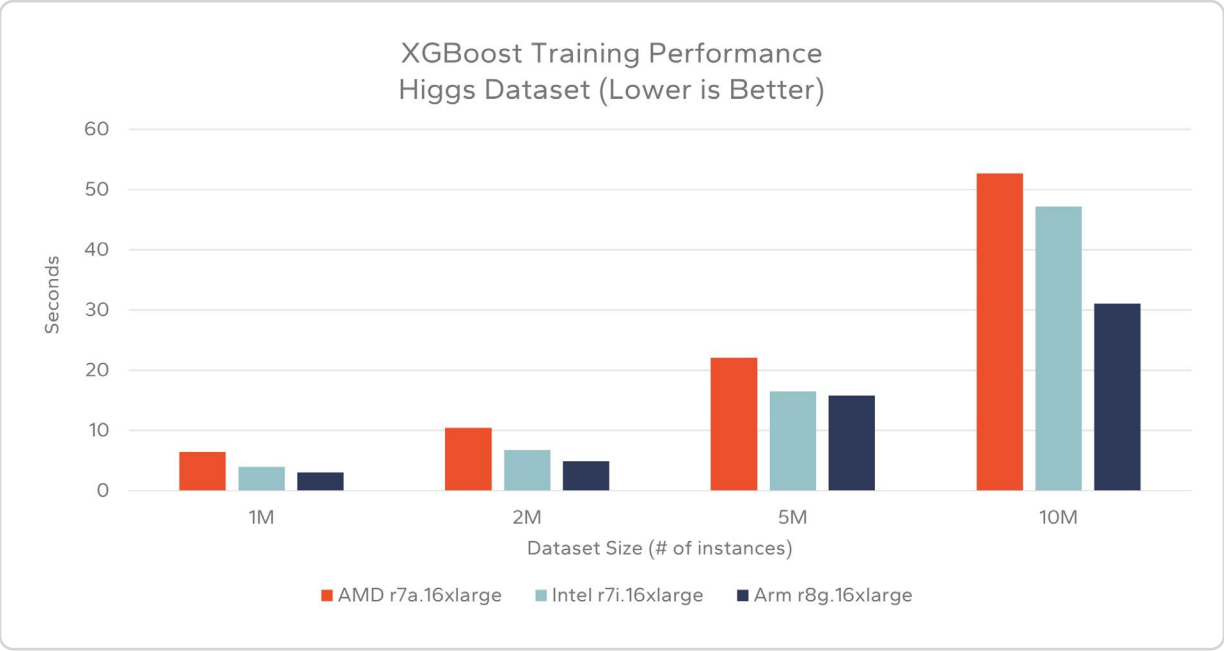


Figure 9: *XGBoost Training Performance*

XGBoost training tests showed AWS Graviton4, powered by Arm Neoverse, to achieve faster training performance than both Intel and AMD instances at each dataset size tested. When looking at the 1M, 2M, 5M, and 10M dataset sizes, the AWS Graviton4 instance achieved between 53% and 29% lower training time than the AMD instances, which were found to have the highest overall training times. Compared to Intel, the Graviton4 instances achieved between 4% and 34% lower training times.

Signal65 Comment: *For machine learning algorithms such as XGBoost, model training performance is critical. Faster training times not only help reduce cloud or infrastructure costs but also enable greater experimentation, which can lead to more accurate and robust models. In the case of XGBoost, improved performance allows organizations to run more extensive hyperparameter tuning, perform additional boosting rounds, and retrain more frequently on fresh data. These time savings become increasingly impactful when training with large datasets. For enterprises leveraging machine learning, the performance advantages achieved by Arm during XGBoost training can improve efficiency and result in more impactful machine learning models.*

Similarly, the AWS Graviton4 instance was found to achieve faster inferencing performance as well. Inferencing was achieved with test datasets sized at 20% of each training dataset. Examining results for the corresponding 1M to 10M training dataset sizes, Arm achieved between 14% and 17% faster inferencing than AMD, and between a 40% and 41% advantage compared to Intel.

[7] https://github.com/dmlc/xgboost-bench

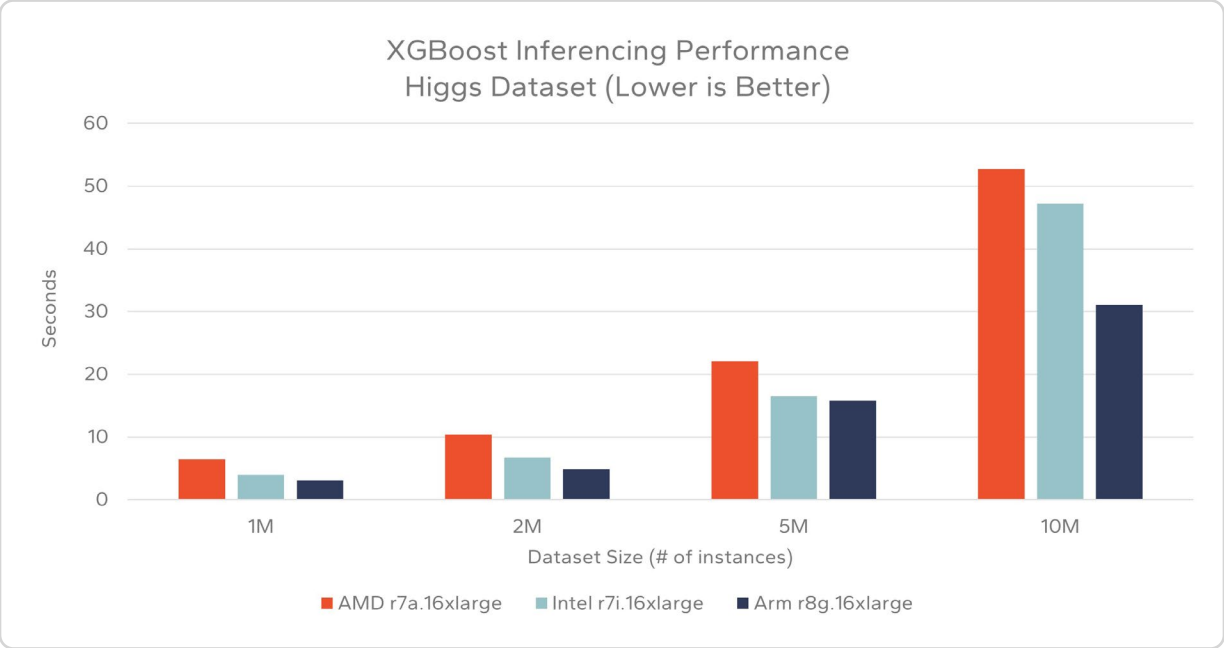


Figure 10: *XGBoost Inferencing Performance*

Utilizing public pricing data for each EC2 instance tested, these results can additionally be examined as the cost to run each workload. Pricing for each instance at the time of testing can be seen in Figure 11 below:

Instance Type	On Demand Hourly Price
r8g.16xlarge (Arm)	$3.77024
r7a.16xlarge (AMD)	$4.86880
r7i.16xlarge (Intel)	$4.23360

Figure 11: *XGBoost Instance Pricing (Source: AWS[8])*

Before considering performance, Graviton4 holds a significant advantage in hourly pricing alone, costing 23% less than the AMD instance and 13% less than the Intel instance. When combining the pricing data with the performance data, the Arm-based Graviton4 instance is shown to have an even larger price advantage, both for training and inferencing.

This price-performance analysis shows that Arm can perform XGBoost training at between 45% and 64% lower cost than AMD and between 15% and 49% lower cost than Intel.

[8] Pricing data sourced from AWS On-Demand EC2 plans as of June 10, 2025. All pricing uses US East(Ohio) region.

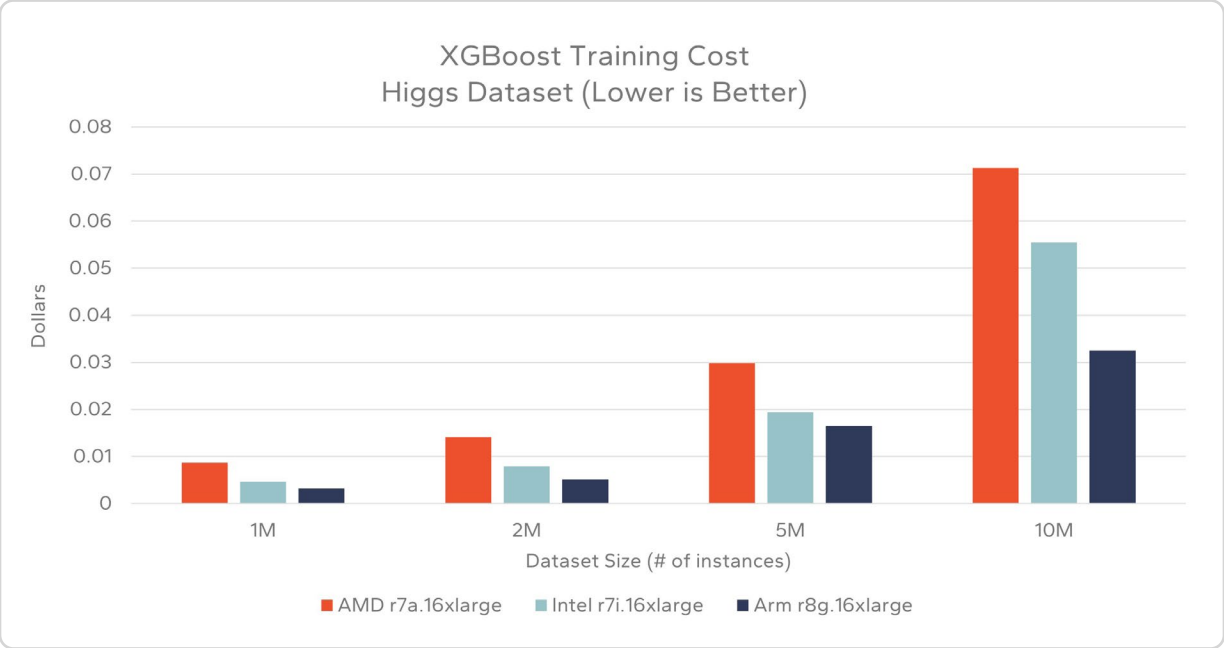


Figure 12: *XGBoost Training Cost*

For inferencing, cost was evaluated as the price required to predict each dataset as a batch process, based on the measured performance. Arm-based Graviton4 was found to achieve between a 34% and 35% cost advantage compared to AMD, and between a 46% and 47% advantage compared to Intel.

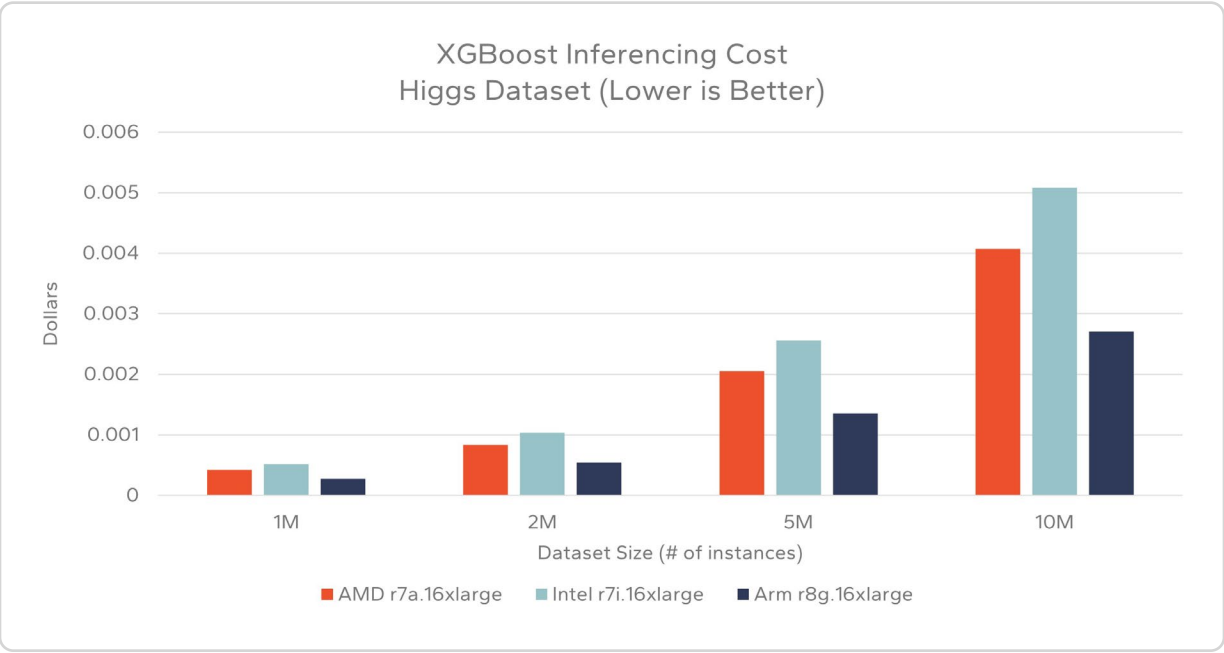


Figure 13: *XGBoost Inference Cost*

Signal65 Comment: *Testing showed that Arm delivers significant advantages in both XGBoost training and inference, driven by clear architectural strengths. The Graviton4 r8g.16xlarge pairs 64 physical cores (no SMT) with large 2 MB L2 caches, a unified 36 MB system cache, and 12 channels of high-bandwidth DDR5-5600, reducing latency and maximizing per-core throughput. SVE2 vectorization, advanced branch prediction, and superior memory bandwidth further give it a strong edge over AMD's chiplet-based design, which suffers from NUMA-related latency, and Intel's architecture, which relies on fewer physical cores and SMT, leading to potential resource contention. These architectural advantages make Graviton well suited for end-to-end machine learning workflows.*

Llama

While discussion regarding large language models is often centered around GPU acceleration, many practical AI workloads can be run on CPUs as well. GPUs are typically required for model training and very large scale AI implementations, beyond the scope of many enterprises. AI inferencing, however, can be achieved on CPUs – providing a practical, cost effective way for enterprises to leverage smaller scale AI inferencing.

To test the AI inferencing capabilities of different CPU providers on AWS, Signal65 conducted performance testing of Llama-3.1-8B and Llama-3.3-70B using the llama.cpp[9] batched-bench benchmark.

The Llama model family represents one of the most popular open-source LLMs currently available. Llama-3.1-8B is a relatively small model, making it well suited for CPU-based inferencing. While Llama-3.3-70B is a much larger model, testing of such a large model showcases the overall AI capabilities of each CPU. Both models were tested with two distinct context sizes and a range of batch sizes to measure the throughput in tokens per second. The context sizes tested can be seen in Figure 14, below.

Description	Input / Output Tokens	Use Case Examples
Small input / Small output	256 / 256	• Chatbot • Code completion
Large input / small output	2048 /256	• Document summarization • RAG

Figure 14: *Context Size Configurations*

Across the two models tested, the Intel and AMD instances achieved relatively comparable performance, with performance advantages depending on the specific configuration. The Arm-based Graviton4 instance, however, achieved a significant throughput advantage against both competitors across all test configurations.

For Llama-3.1-8B, Graviton4 achieved between 127% and 137% higher throughput than AMD for the small input / small output test configuration (256 / 256). Compared to Intel, Graviton4 achieved between 135% and 180% advantage for this configuration.

[9] https://github.com/ggml-org/llama.cpp

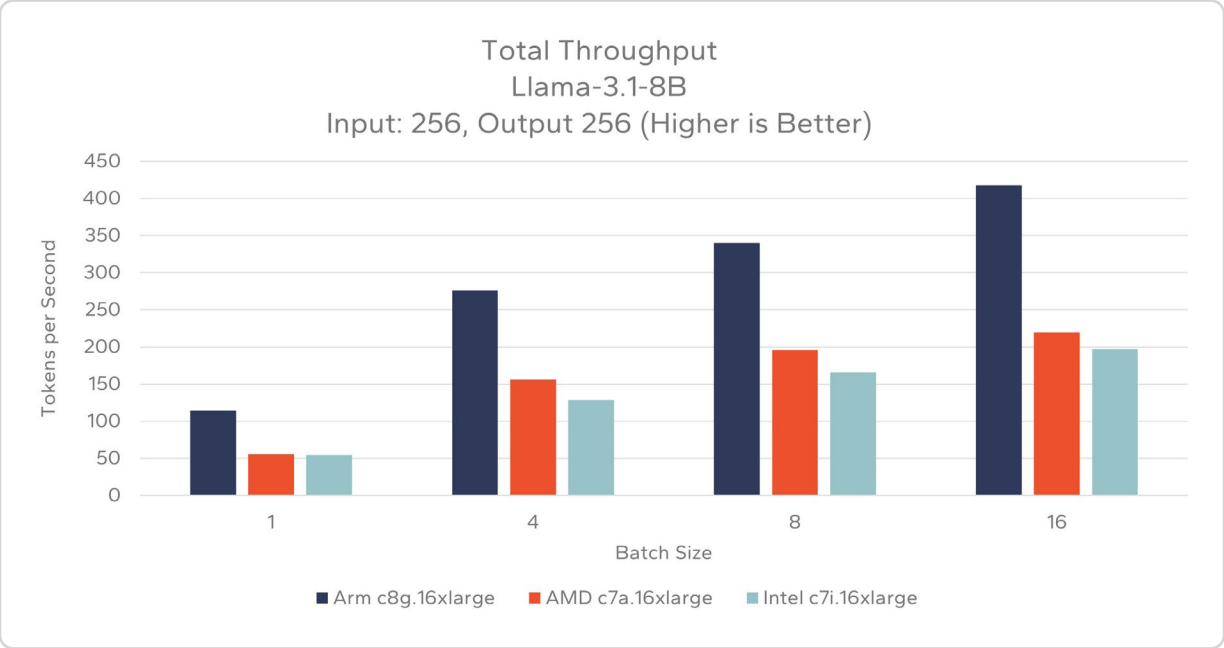


Figure 15: *Llama-3.1-8B Throughput (Input: 256 / Output: 256)*

For the large input / small output configuration (2048/256), Graviton4 achieved between 102% and 168% higher throughput than AMD and between 135% and 162% higher throughput than Intel.

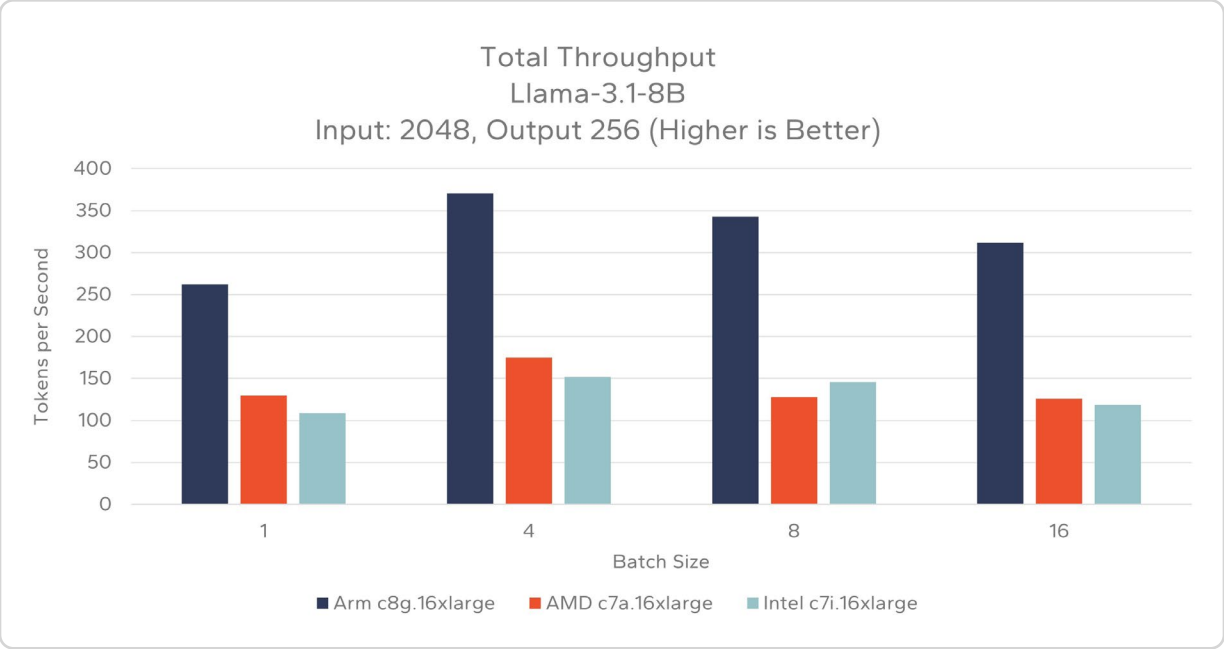


Figure 16: *Llama-3.1-8B Throughput (Input: 2048 / Output: 256)*

Just as with Llama-3.1-8B model, Graviton4 was found to have a significant performance advantage for the larger Llama-3.3-70B model. For the small input / small output testing, Graviton4 achieved between 116% and 120% higher throughput than AMD and between 108% and 117% higher throughput than Intel.

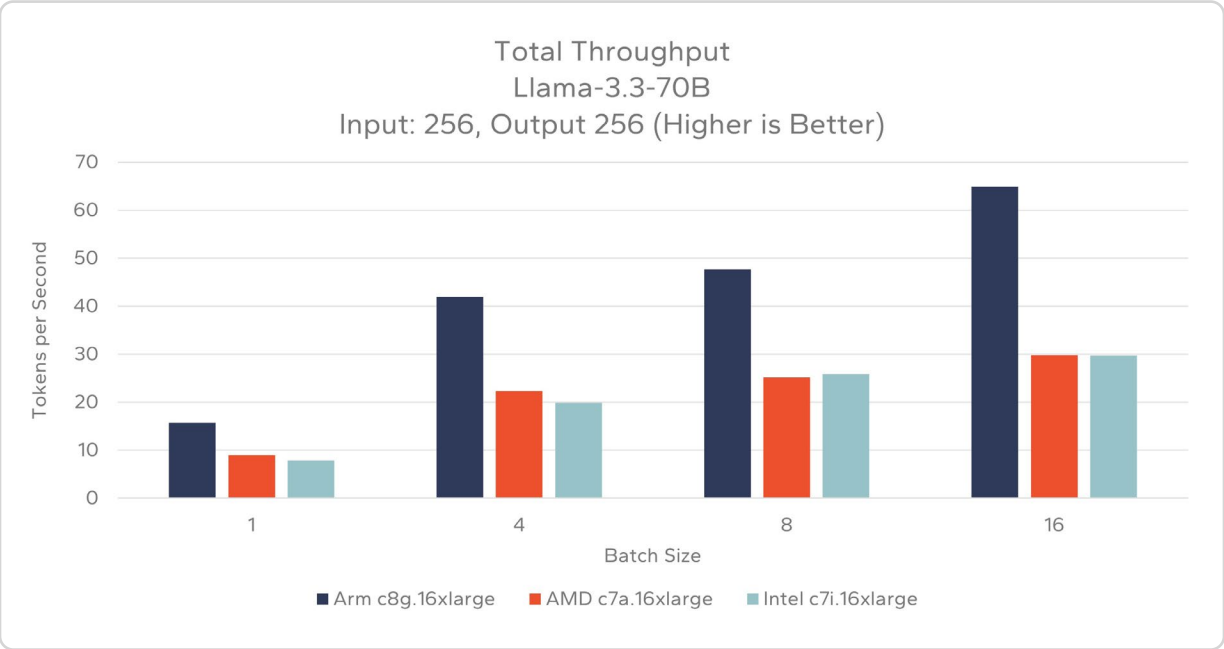


Figure 17: *Llama-3.3-70B Throughput (Input: 256 / Output: 256)*

For the large input / small output testing of Llama-3.3-70B, Graviton4 achieved between 98% and 154% higher throughput than AMD and between 119% and 148% higher throughput than Intel.

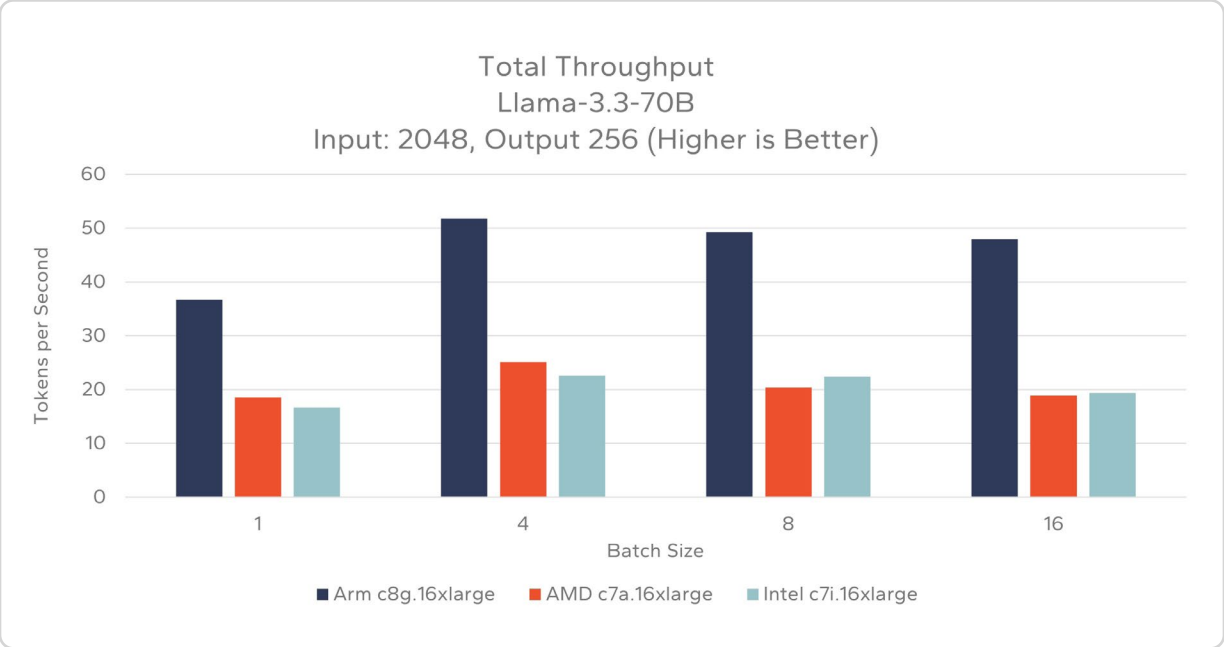


Figure 18: *Llama-3.3-70B Throughput (Input: 2048 / Output: 256)*

Signal65 Comment: *Arm's large performance advantage across all Llama inferencing tests is quite notable. Arm achieved leading performance for both model sizes at all context window and batch size configurations. This consistent performance advantage is indicative of notable architectural differentiation and showcases Arm as a practical choice for CPU-based AI deployments capable of meeting various model and use case requirements.*

While Arm achieves a significant performance advantage compared to both competitors, it additionally holds a significant economic advantage. By utilizing the performance testing results alongside publicly available pricing data, each instance can be compared by the tokens-per-dollar that it achieves. Pricing for each instance at the time of testing can be found below:

Instance Type	On Demand Hourly Price
c8g.16xlarge (Arm)	$2.54336
c7a.16xlarge (AMD)	$3.28448
c7i.16xlarge (Intel)	$2.85600

Figure 19: *Llama Instance Pricing (Source: AWS[10])*

When factoring in price, Arm-based Graviton4 achieves an even more significant advantage in each test scenario. Compared to AMD, Graviton4 achieved up to 205% more tokens per dollar running Llama-3.1-8B with a small input / small out, and up to 220% more tokens per dollar with a large input / small output configuration. Compared to Intel, Graviton4 achieved up to 214% and 195% more tokens per dollar for these configurations.

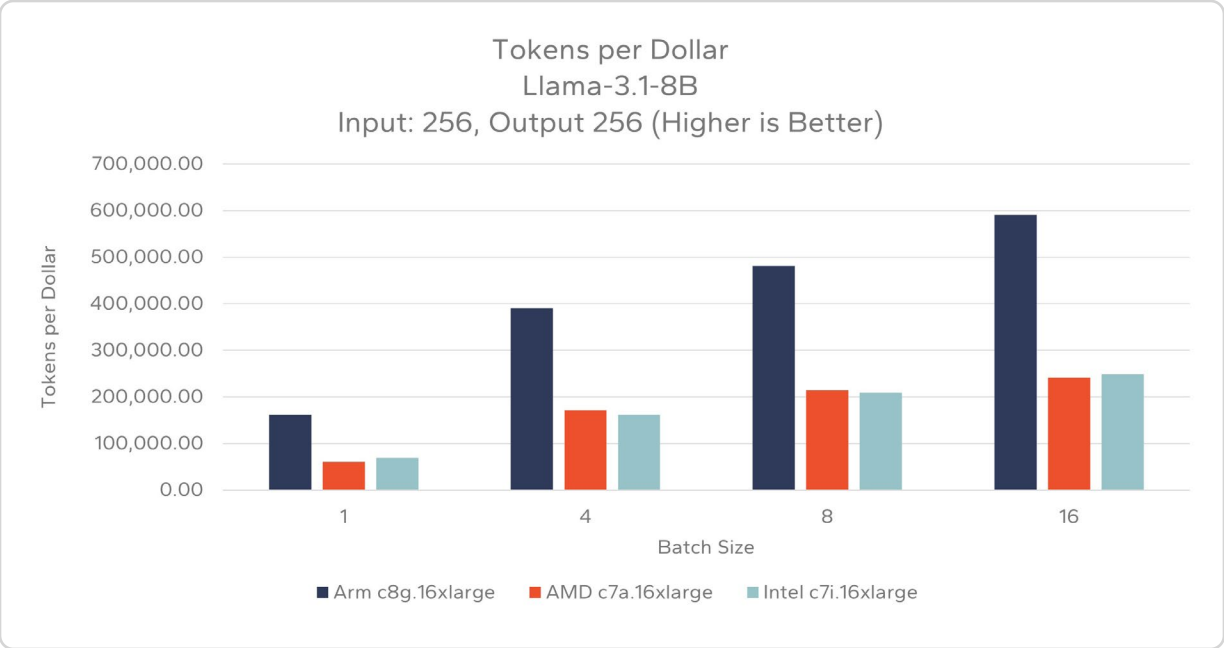


Figure 20: *Llama-3.1-8B Tokens per Dollar (Input: 256 / Output: 256)*

[10] Pricing data sourced from AWS On-Demand EC2 plans as of June 10, 2025. All pricing uses US East(Ohio) region.

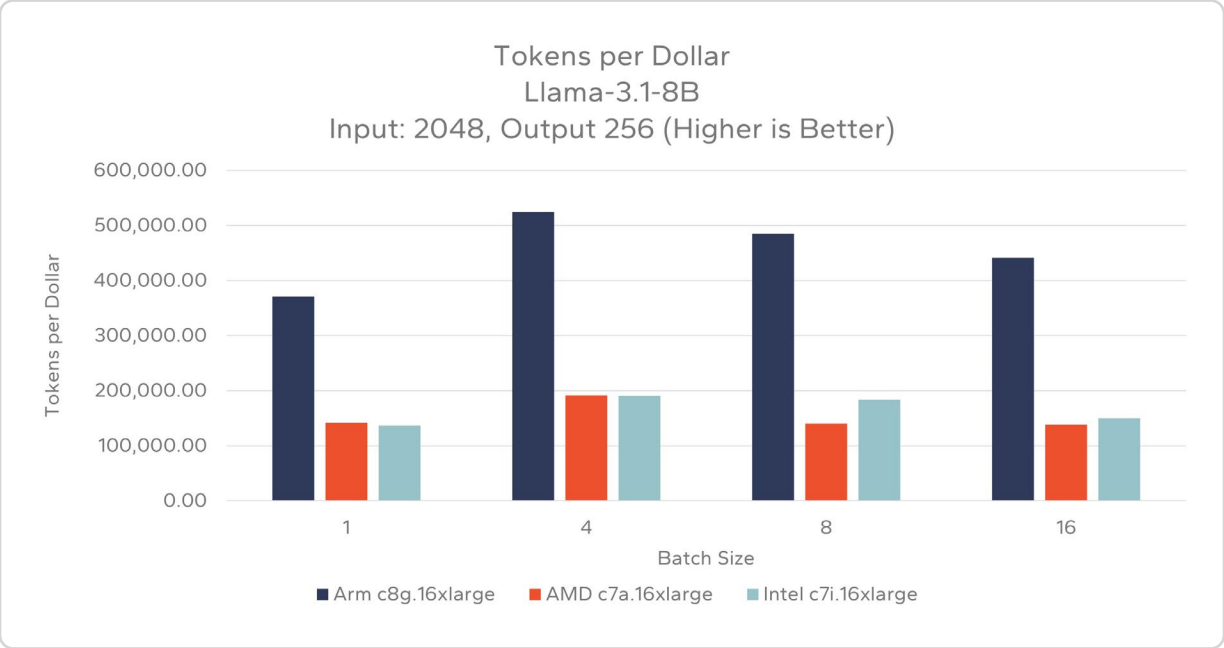


Figure 21: *Llama-3.1-8B Tokens per Dollar (Input: 2048 / Output: 256)*

Graviton's price-performance advantage was also significant when running Llama-3.3-70B. For the small input / small output configuration, Graviton4 achieved up to a 184% token per dollar advantage compared to AMD and up to a 144% token per dollar advantage compared to Intel. For the large input / small output test configuration Graviton4 achieved up to a 227% token per dollar advantage over AMD and up to a 178% token per dollar advantage compared to Intel.

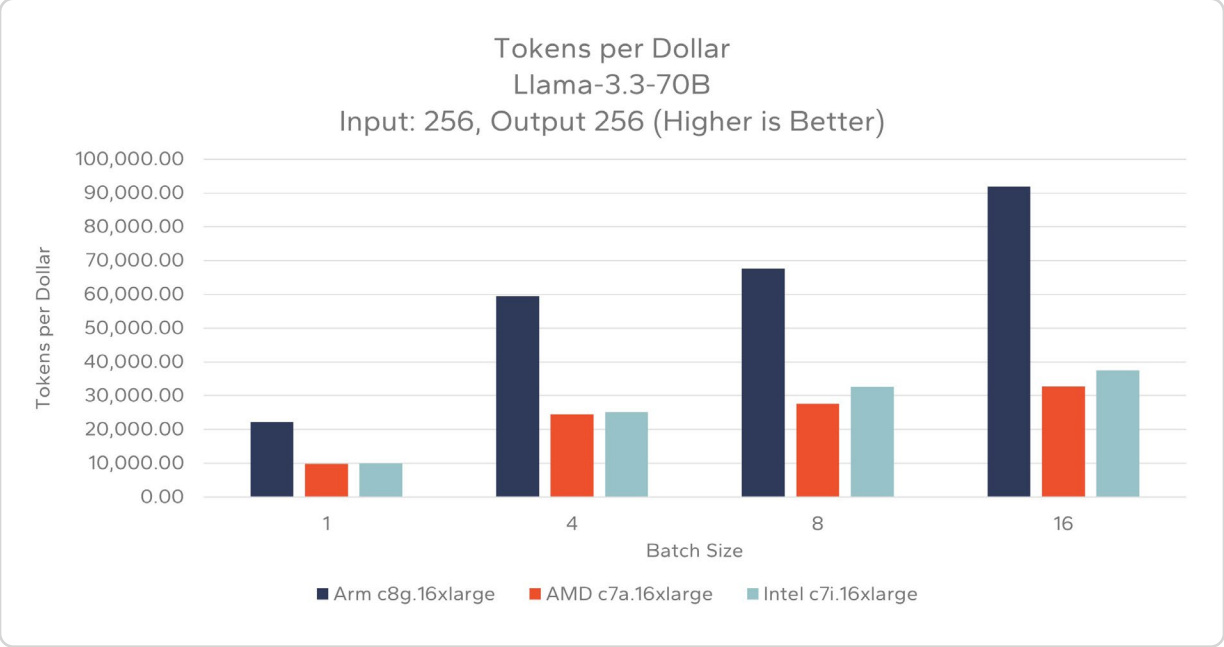


Figure 22: *Llama-3.3-70B Tokens per Dollar (Input: 256 / Output: 256)*

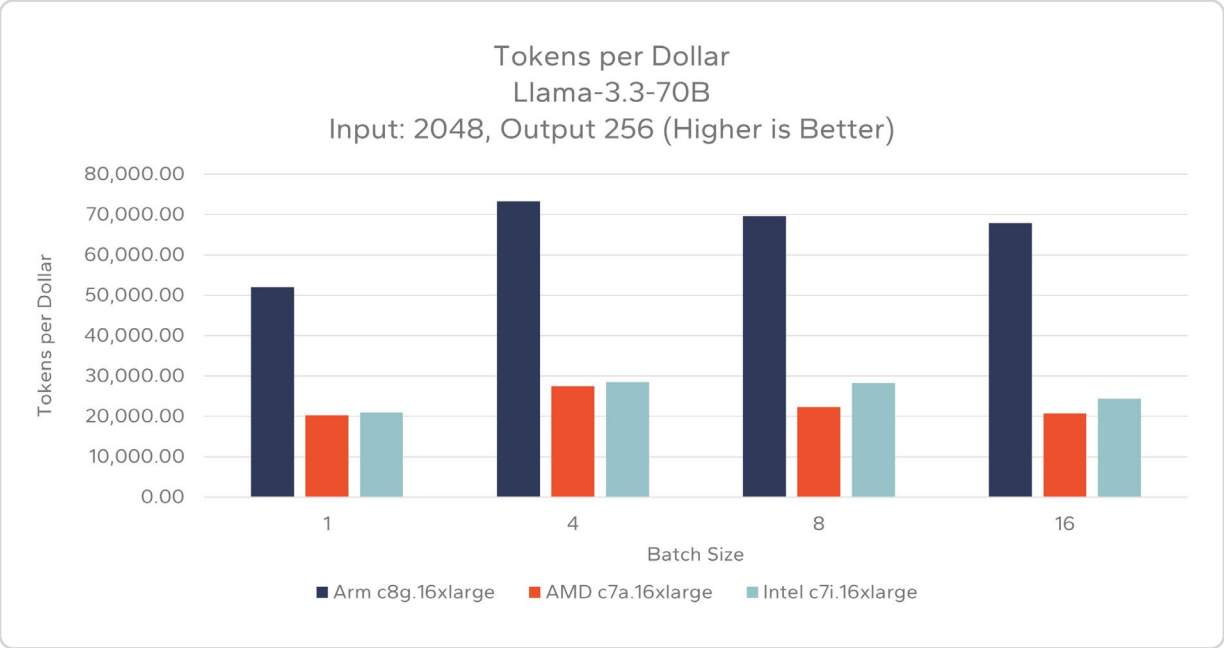


Figure 23: *Llama-3.3-70B Tokens per Dollar (Input: 2048 / Output: 256)*

Analysis

The results of this testing show that Arm-based AWS Graviton4 instances produce faster and better performance at a much lower cost than either AMD or Intel based AWS EC2 instances. This is the case across a variety of workload types tested, both AI/ML and general compute, highlighting Arm's broad applicability for enterprise workloads deployed on AWS.

When examining the Redis database workload, Arm-based AWS Graviton4 instances performed 41% and 93% better than Intel and AMD respectively in operations per second. The Graviton4 instances also had 29% and 48% lower latencies than Intel and AMD respectively. This superior performance also costs less, with 59% and 149% more operations/$ than AMD and Intel. Arm's large performance lead and lower cost than the competitors combine to make Graviton4 instances a compelling choice for this workload.

A similar story is seen with the Nginx web server workload. Arm-based AWS Graviton4 instances again strongly outperform both AMD and Intel in terms of requests per second at 34% and 43% respectively with xlarge size instances. The gains are even higher with 2xlarge size instances, 40% and 53% vs AMD and Intel respectively. Arm based instances' performance also scaled with increasing instance size (1.99x) while AMD and Intel did not scale as well (1.91x and 1.86x respectively). Cost analysis shows even better results for Graviton4 CPUs with a range of 61% to 81% more operations/$ depending on instance size. For companies that run web server applications such as Nginx, Arm-based AWS Graviton4 CPUs provide the performance and price point to meet a wide range of enterprise requirements.

While Arm's performance gains vs AMD and Intel with general compute workloads are impressive, they also carried over to AI/ML workloads. The performance and cost/performance advantages achieved in both XGBoost training and inferencing highlight AWS Graviton4 as a highly efficient platform for end-to-end XGBoost processes. Meanwhile, Graviton4 held a clear advantage over its competitors on Llama inferencing tests, showcasing its abilities to run generative AI applications.

With the XGBoost training workload, AWS Graviton4 instances achieved up to 54% lower training times than AMD and up to 34% lower training times compared to Intel. When factoring in cost, these advantages rose to 64% and 49% respectively. This type of performance advantage not only saves IT organizations significant money, but enables data scientists to iterate quicker resulting in more accurate models and better outcomes.

Arm was also found to achieve faster XGBoost inferencing performance, with a performance advantage of up to 16% over AMD and up to 40% over Intel. For ML inferencing, especially in applications requiring real time or low latency predictions, this level of performance advantage is highly impactful. Faster inference enables quicker insights, improves responsiveness, and gives organizations greater agility to quickly leverage their ML models to make impactful decisions.

When considering newer generative AI applications, many organizations will choose to strategically deploy these AI workloads on CPUs due to the cost and resource limitations of GPUs. This testing showed Arm-based Graviton4 instances are capable of achieving faster inferencing of two distinct Llama AI models at a lower cost than AMD or Intel. On average across test cases, Arm achieved 2.7x more tokens per dollar than AMD and 2.56x more tokens per dollar than Intel when running Llama-3.1-8B. The cost and performance advantages achieved by Arm opens the door for organizations to deploy practical LLM inferencing applications on CPUs, while reserving GPU resources for where they are most needed.

Final Thoughts

The significance of these results ultimately translates into cost savings for IT organizations. By leveraging the cost-per-dollar advantages of Arm-based AWS Graviton4 instances, organizations can achieve the same amount of production at a lower cost, or alternatively, achieve greater production for the same cost that would be required for AMD or Intel based instances.

IT organizations are commonly tasked to do more with less, managing both core enterprise and emerging AI workloads while operating within shrinking budgets. To meet these demands, it's essential to make infrastructure choices that deliver both performance and cost efficiency. While cloud services, such as EC2, offer a wide array of options, IT decision-makers should carefully evaluate the cost and performance characteristics of each instance type, and their underlying CPUs, in relation to specific workloads.

This testing of AWS Graviton4 instances revealed substantial performance improvements and cost savings across a broad range of enterprise workloads. Notably, Arm outperformed both competitors across all four workloads tested, showcasing its flexibility to support several key enterprise applications. These results demonstrate the technical innovation of Arm Neoverse CPUs, along with the economically compelling positioning of Graviton4 instances on AWS.

Additionally, Signal65 found no meaningful differences in software compatibility or ease of deployment between Arm-based instances and alternative AMD and Intel instances throughout its testing. This indicates that the performance and cost-efficiency advantages found across these workloads can be achieved by IT organizations with minimal operational disruption.

IT decision makers should consider AWS Graviton4 instances as part of their AWS cloud strategy, both for general compute and AI/ML workloads. By leveraging the architectural advantages of Arm Neoverse CPUs, organizations can achieve measurable gains in cost efficiency and performance, helping to meet the requirements of both traditional and newly emerging workloads. The impressive performance and cost characteristics demonstrated by this testing highlight the changing cloud infrastructure landscape and establish Arm as a leading option for organizations deploying AWS EC2 instances.

Appendix

AWS Environment

Signal65 conducted testing in our private AWS environment for the entirety of the testing. The environment included all infrastructure needed for the workloads including subnet, security group/rules, internet gateway, etc. Testing was completed on AWS EC2 instances available as of June 10, 2025.

Pricing Information

All pricing information used in this study was sourced from publicly available AWS EC2 On-Demand pricing, using the US East(Ohio) region. Pricing was sourced as of June 10, 2025.

Additional Results and Configuration Details

Llama

EC2 Instance	Processor	vCPU	Memory
C8g.16xlarge	AWS Graviton4	64	128 GiB
c7a.16xlarge	AMD EPYC Genoa (SMT-off)	64	128 GiB
c7i.16xlarge	Intel Xeon Sapphire Rapids	64	128 GiB

Figure 24: *EC2 Instances Tested (Llama)*

Inference testing of Llama models measured prompt processing, token generation, and total token processing capabilities. Full results for each model, context length, and batch size configuration are available in the charts below.

Batch Size	1	4	8	16
Prompt Processing (tokens/s)				
c8g.16xlarge	585.06	700.44	733.87	730.22
c7a.16xlarge	255.05	295.59	323.62	309.03
c7i.16xlarge	209.22	289.92	305.32	310.21
Token Generation (tokens/s)				
c8g.16xlarge	63.23	171.69	221.11	292.26
c7a.16xlarge	31.04	106.22	140.33	170.24
c7i.16xlarge	31.41	82.31	113.62	144.66
Total (tokens/s)				
c8g.16xlarge	114.13	275.78	339.83	417.44
c7a.16xlarge	55.35	156.28	195.77	219.53
c7i.16xlarge	54.62	128.21	165.61	197.3

Figure 25: *Llama-3.1-8B (256/256) Inferencing Results*

Batch Size	1	4	8	16
Prompt Processing (tokens/s)				
c8g.16xlarge	558.37	497.15	418.29	360.19
c7a.16xlarge	267.73	224.17	143.48	138.53
c7i.16xlarge	210.74	210.29	191.54	141.24
Token Generation (tokens/s)				
c8g.16xlarge	50.02	122.35	140.12	150.17
c7a.16xlarge	25.27	63.5	68.26	72.35
c7i.16xlarge	22.29	46.98	50.08	52.28
Total (tokens/s)				
c8g.16xlarge	262.25	370.91	342.7	311.75
c7a.16xlarge	129.58	174.98	127.83	125.75
c7i.16xlarge	108.67	151.7	145.79	118.78

Figure 26: *Llama-3.1-8B (2048/256) Inferencing Results*

Batch Size	1	4	8	16
Prompt Processing (tokens/s)				
c8g.16xlarge	74.48	78.36	78.86	78.36
r7a.16xlarge	34.51	36.04	36.03	35.69
r7i.16xlarge	34.26	37.58	37.75	37.59
Token Generation (tokens/s)				
c8g.16xlarge	8.78	28.66	34.2	55.47
r7a.16xlarge	5.16	16.19	19.31	25.63
r7i.16xlarge	4.45	13.56	19.67	24.59
Total (tokens/s)				
c8g.16xlarge	15.71	41.97	47.71	64.95
r7a.16xlarge	8.98	22.34	25.15	29.83
r7i.16xlarge	7.88	19.93	25.86	29.73

Figure 27: *Llama-3.3-70B (256/256) Inferencing Results*

Batch Size	1	4	8	16
Prompt Processing (tokens/s)				
c8g.16xlarge	68.32	63.31	56.44	51.58
r7a.16xlarge	31.34	29.84	22.45	20.06
r7i.16xlarge	29.26	28.6	27.08	21.96
Token Generation (tokens/s)				
c8g.16xlarge	7.81	21.03	24.26	30.7
r7a.16xlarge	4.34	11.04	11.66	12.99
r7i.16xlarge	3.75	8.38	9.49	9.95
Total (tokens/s)				
c8g.16xlarge	36.73	51.75	49.19	47.95
r7a.16xlarge	18.54	25.09	20.36	18.91
r7i.16xlarge	16.67	22.55	22.45	19.36

Figure 28: *Llama-3.3-70B (2048/256) Inferencing Results*

XGBoost

EC2 Instance	Processor	vCPU	Memory
r8g.16xlarge	AWS Graviton4	64	512 GiB
r7a.16xlarge	AMD EPYC Genoa (SMT-off)	64	512 GiB
r7i.16xlarge	Intel Xeon Sapphire Rapids	64	512 GiB

Figure 29: *EC2 Instances Tested (XGBoost)*

booster	gbtree
objective	binary:logistic
eval_metric	logloss
num_trees	500
max_depth	8
max_leaves	256
eta (learning_rate)	0.10
reg_alpha (L1)	0.90
reg_lambda (L2)	1
subsample	1.0
colsample_bytree	1.0
tree_method	hist
scale_pos_weight	2
seed / random_state	0
version	2.1.1

Figure 30: *XGBoost Hyperparameters*

XGBoost Training Time (s)			
	r7a.16xlarge	**r7i.16xlarge**	**r8g.16xlarge**
higgs1m	6.4481529	3.9501672	3.053839
higgs2m	10.427143	6.7625724	4.8953356
higgs5m	22.098804	16.485834	15.781319
higgs10m	52.712483	47.181538	31.057467

Figure 31: *XGBoost Training Results*

XGBoost Inference Time (s)			
	r7a.16xlarge	**r7i.16xlarge**	**r8g.16xlarge**
higgs1m	0.3148779	0.4413019	0.2626021
higgs2m	0.6189278	0.8790217	0.5203919
higgs5m	1.5178429	2.1793192	1.2931957
higgs10m	3.0093134	4.3251077	2.5837367

Figure 32: *XGBoost Inference Results*

Redis

Redis was deployed and ran following the steps outlined in the following github link - https://github.com/ARM-software/developer/tree/master/projects/redis_benchmark

One r.2xlarge instance was used as a load generator and one r.2xlarge instance running the Redis database. The AMD/ Intel instances were created with ami-0c3b809fcf2445b6a and the Arm instances with ami-068da067c3c4ef0d3.

The results collected were the median values of three separate runs.

EC2 Instance	Processor	vCPU	Memory
r8g.2xlarge	AWS Graviton4	8	64 GiB
r7a.2xlarge	AMD EPYC Genoa (SMT-off) – need to validate	8	64 GiB
r7i.2xlarge	Intel Xeon Sapphire Rapids	8	64 GiB

Figure 32: *EC2 Instances Tested (Redis)*

EC2 Instance	Total Ops/second	Total Avg. Latency (ms)	(Total Ops /s) / $
r8g.2xlarge	373,866	0.669	793,300
r7a.2xlarge	194,096	0.942	318,922
r7i.2xlarge	264,619	1.287	500,037

Figure 33: *Redis Results*

Nginx

Nginx was deployed using six m7i.16xlarge (ami-0c3b809fcf2445b6a) instances used as file servers, one m.xlarge or m.2xlarge instance as the load balancer (AMD/Intel ami-0c3b809fcf2445b6a, Arm ami-068da067c3c4ef0d3), and one m7i.24xlarge (ami-0c3b809fcf2445b6a) instance as the load generator running wrk2.

Configuration of the load balancer and file servers was done in accordance with guidelines provided by Arm - https://learn.arm.com/learning-paths/servers-and-cloud-computing/nginx_tune/

The results collected were the median values of three separate runs.

EC2 Instance	Processor	vCPU	Memory
m8g.xlarge	AWS Graviton4	4	16 GiB
m8g.2xlarge	AWS Graviton4	8	32 GiB
m7a.xlarge	AMD EPYC Genoa (SMT-off) – need to validate	4	16 GiB
m7a.2xlarge	AMD EPYC Genoa (SMT-off) – need to validate	8	32 GiB
m7i.xlarge	Intel Xeon Sapphire Rapids	4	16 GiB
m7i.2xlarge	Intel Xeon Sapphire Rapids	8	32 GiB

Figure 34: *EC2 Instances Tested (Nginx)*

EC2 Instance	Requests/second	Req/sec Scale Factor	(Requests/sec)/$
m8g.xlarge	195,930	-	1,091,412
m8g.2xlarge	389,699	1.99x	1,085,392
m7a.xlarge	146,022	-	629,838
m7a.2xlarge	278,397	1.91x	600,407
m7i.xlarge	136,813	-	678,634
m7i.2xlarge	254,372	1.86x	630,884

Figure 35: *Nginx Results*

Important Information About this Report

CONTRIBUTORS

Mitch Lewis
Performance Analyst | Signal65

Jonathan Fellows
Performance Validation Engineer | Signal65

PUBLISHER

Ryan Shrout
President and GM | Signal65

INQUIRIES

Contact us if you would like to discuss this report and Signal65 will respond promptly.

CITATIONS

This paper can be cited by accredited press and analysts, but must be cited in-context, displaying author's name, author's title, and "Signal65." Non-press and non-analysts must receive prior written permission by Signal65 for any citations.

LICENSING

DISCLOSURES

Signal65 provides research, analysis, advising, and consulting to many high-tech companies, including those mentioned in this paper. No employees at the firm hold any equity positions with any companies cited in this document.

IN PARTNERSHIP WITH

arm

ABOUT SIGNAL65

Signal65 is an independent research, analysis, and advisory firm, focused on digital innovation and market-disrupting technologies and trends. Every day our analysts, researchers, and advisors help business leaders from around the world anticipate tectonic shifts in their industries and leverage disruptive innovation to either gain or maintain a competitive advantage in their markets.

signal65

signal65

CONTACT INFORMATION
Signal65 I signal65.com