

Templado System

Empowering Payments with ARM-Based Payment Gateway Technology

Table of Contents

Abstract.....	2
Background	2
The Advantages of ARM: From Smartphones to Supercomputers and Beyond.....	3
Higher core count, better performance per watt.....	3
Lower TCO, better performance per dollar	3
Cloud-nativity	3
Reduced on-premise infrastructure costs.	3
Performance	3
The Reference Architecture	4
Platform Components	5
Gateway.....	5
Transaction Engine	5
Merchant Portal.....	6
Payment Terminal.....	7
Admin Interface	7
Note on standard.....	7
Merchant Model	7
Direct Merchant.....	7
Aggregation Merchant.....	8
Hybrid Merchant.....	9
Testing.....	9
On premises systems	9
Amazon Graviton	10
ARM On Cloud	11
Database on ARM.....	14
Conclusion	17
About Templado System	18

Abstract

This white paper examines the benefits of ARM-based technology in payment gateway systems, analyzing the performance, security, scalability, and cost-efficiency of ARM architecture and its suitability for modern payment processing infrastructure.

Background

In today's digital age, online payments have become an integral part of businesses across various industries. Payment gateways play a crucial role in securely processing transactions, ensuring seamless customer experiences, and facilitating efficient business operations. This white paper focuses on the background and purpose of utilizing ARM-based technology in payment gateway systems.

Over the past decade, and especially over the past five years, ARM processors have come to dominate the smartphone and tablet landscape, offering strong performance-per-watt along with lower design, production, and deployment costs. Processors based on the ARM architecture, an alternative to the mainstream x86 architecture, are gradually making the leap from mobile devices to servers and data centers, marking an important shift in server-based computing.

A traditional x86-class server with twelve, sixteen, twenty-four, or more cores increases performance by scaling up the speed and sophistication of each processor, relying on raw per-core power to handle demanding computing workloads. By comparison, an ARM server employs many smaller, more power-efficient processors that share processing tasks across hundreds of cores rather than channeling the workload through just a few. This approach is commonly referred to as "scaling out."

All major cloud providers—AWS, Azure, Oracle, and Google—offer ARM instances. Amazon's instances are based on its Graviton ARM CPU, while the other providers offer instances built on Ampere ARM CPUs.

The purpose of this white paper is to provide an in-depth understanding of ARM-based payment gateway technology. It aims to educate businesses, payment service providers, and developers about the benefits and implications of adopting ARM architecture in their payment processing infrastructure. By exploring the performance, security, scalability, and cost-effectiveness aspects, this white paper aims to empower decision-makers to make informed choices for their payment gateway systems.

The Advantages of ARM: From Smartphones to Supercomputers and Beyond

Higher core count, better performance per watt

While the maximum number of cores in each CPU is not wholly dependent upon the architecture, mainstream x86 CPUs tend to house a smaller number of larger, more complex cores. ARM CPUs compete by having more cores inside a single processor; for example, the Ampere® Altra® Max series can contain up to 128 cores in each CPU. The workload is distributed to a large number of smaller, more efficient cores, rather than having a few powerful cores handle every task. As a result, the cores of an ARM processor can offer better performance per watt of power.

Lower TCO, better performance per dollar

ARM servers can offer a meaningfully better cost-per-dollar than traditional x86 servers. ARM CPUs are known for their energy efficiency. They typically consume less power compared to x86 CPUs, resulting in lower electricity costs over time. This is especially beneficial in server rooms or data centers where power consumption is a significant expense. The reduced power consumption also leads to lower cooling requirements, further contributing to cost savings. ARM processors are generally cheaper to design and manufacture compared to x86 processors. The licensing model of ARM architecture allows multiple companies to design and produce ARM-based chips, increasing competition and driving down costs. This cost advantage can make ARM servers more affordable, especially in large-scale deployments. ARM servers offer scalability and flexibility, making them suitable for various workloads and applications. They are commonly used in cloud computing, web hosting, and edge computing environments. The ability to optimize hardware configurations and leverage ARM's architecture for specific workloads can result in better performance per dollar compared to x86 servers.

Cloud-nativity

Price-for-performance is a deciding factor for companies that choose ARM, and for good reason: ARM is just as efficient as x86 for cloud computing while offering more attractive pricing. Organizations can save money while processing a broad spectrum of workloads, putting ARM's performance into direct competition with x86 architectures; in this area, its rivals simply cannot surpass ARM's price-to-performance ratio. AWS Graviton, Amazon's first ARM-based processor offering, can save AWS users as much as 40 percent in costs for performance equal to that of Amazon's previous x86 cloud processors.

Reduced on-premise infrastructure costs.

The infrastructure costs associated with payment gateway systems can be a significant expense for businesses. ARM-based payment gateways offer cost-effective solutions by minimizing infrastructure requirements. Upfront and ongoing infrastructure costs can be brought down by as much as 33 percent per core, which has a positive trickle-down effect on the cost of cloud-based offerings.

Performance

ARM servers deliver excellent performance for certain workloads, particularly those that are highly parallelizable or require high memory bandwidth. ARM processors also include built-in hardware security features, such as TrustZone technology, which strengthen the security of payment processing systems by protecting against malware and other threats.

The Reference Architecture

The Templado System payment platform, shown in Figure 1, supports both on-premises and cloud deployment. As a cloud-native platform, it runs on the ARM-based compute instances offered by all major cloud providers: Amazon Web Services (AWS Graviton2 and Graviton3), Microsoft Azure (Ampere Altra), Oracle Cloud, and Google Cloud Platform (Tau T2A instances, built on Ampere Altra).

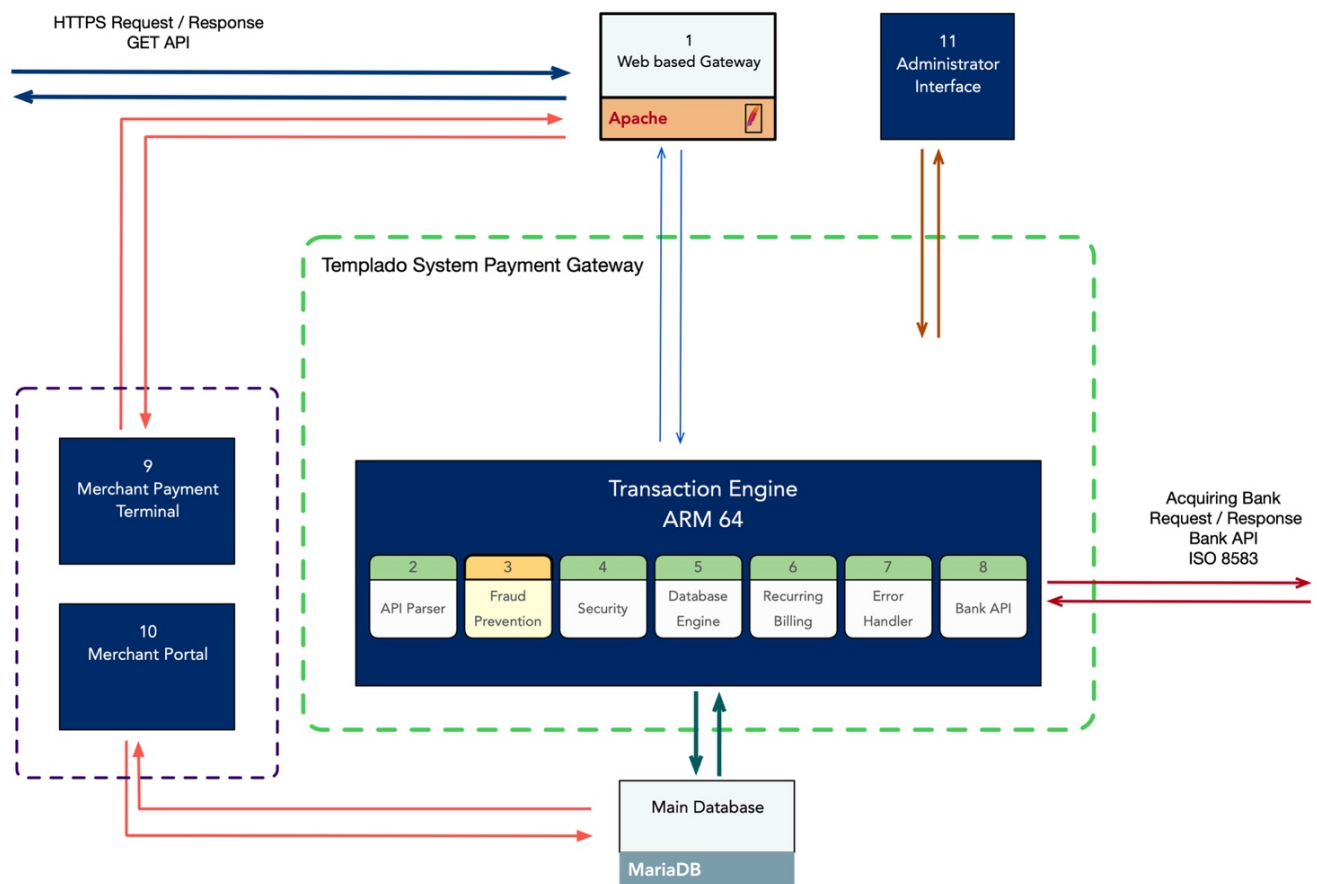


Figure 1. Payment platform reference architecture

The goal was to design the system free from third-party licensing costs, such as Microsoft Windows Server or SQL Server. As a result, the platform runs entirely on open-source software. The system uses the Linux AArch64 platform as its server operating system (Ubuntu, Fedora ARM) and MariaDB or MySQL as its primary database. For security, the platform relies on OpenSSL to perform the necessary cryptographic functions, including the SSL API and AES encryption to protect customer data and manage encryption keys.

Customers are free to use any other key-management solution to protect data, such as AWS Key Management Service (KMS) or a virtual HSM for cloud-based installations, or a physical HSM appliance (such as Thales or Futurex) for on-premises installations.

Platform Components

This is where the platform's core payment-processing functions reside, including the payment gateway, transaction engine, data storage, and merchant-related modules.

Gateway

The web gateway is implemented on the Apache web server. Using Apache offers several advantages for a payment gateway: it is open source and freely available, which helps reduce infrastructure costs for small businesses and startups; it is compatible with multiple operating systems, including Linux, Unix, Windows, and macOS, allowing deployment across a wide range of platforms; and it provides a range of built-in security features, including SSL/TLS encryption to secure communication between clients and the server, along with numerous modules and configurations to mitigate common web server vulnerabilities. Apache HTTP Server is fully cross-platform and runs natively on ARM-based systems.

The purpose of the Apache gateway, often referred to as an API gateway, is to act as an intermediary between clients and backend services. It serves as the platform's central entry point for all incoming requests and provides functionality to manage, secure, and optimize API communication. For HTTPS transactions, the gateway operates as a reverse proxy, accepting incoming HTTPS requests from clients and serving as a secure endpoint for those connections. Once the gateway receives an API request, it forwards the processed request to the Transaction Engine for further handling.

Transaction Engine

The transaction engine is responsible for authenticating transactions, verifying merchants, performing fraud detection, selecting the appropriate acquiring partner bank, and selecting the network protocol or API for bank authentication. The transaction engine submits the request to the acquiring bank and, upon receiving a response, returns the result to the merchant through the merchant's API. The transaction engine also performs all necessary database functions.

To fully take advantage of the ARM architecture, the transaction engine is designed and implemented as a low-level program combining C/C++ and assembly language. It is built to be scalable and to take advantage of multi-core ARM architectures, and it can run on anything from a 4-core ARM Cortex-A70-series processor up to a 128-core Ampere Altra CPU.

The transaction engine has a thread pool of 20 threads available by default. When the CPU has more than 20 cores, there are a few options for optimizing thread utilization and achieving more efficient processing:

Thread per Core (1:1): In this approach, each available core is assigned a single thread from the thread pool. This maximizes parallelism, allowing each core to execute a thread independently. If there are more cores than threads, some cores may remain idle.

Multi-threading: The transaction engine is designed to support multi-threading and can spawn multiple threads per core to fully utilize available resources. For example, on a 40-core CPU the engine can spawn additional threads to reach a total of 40 threads, and so on for larger core counts;

the Ampere Altra family includes 32-, 64-, and 128-core variants. This approach further enhances parallelism and takes advantage of the additional computational power.

API Parser (2)

An API parser module is responsible for parsing and extracting relevant information from incoming API requests.

Fraud Prevention (3)

A payment fraud prevention module is a component designed to detect and prevent fraudulent activities related to payment transactions. Its primary purpose is to identify and mitigate fraudulent behavior. The module utilizes a set of predefined rules and logic to detect specific fraud patterns or known fraud indicators. These rules include velocity checks, IP blacklisting, address verification, and card verification value (CVV) validation.

Security (4)

OpenSSL is used to perform several security tasks, including hashing, AES encryption/decryption, and TLS/SSL transport security.

Database Engine (5)

The transaction engine uses a C++ library to communicate with the database. Each thread maintains its own SQL connection to parallelize database requests.

Recurring Billing (6)

The Templado payment gateway supports recurring billing based on merchant configuration, serving the needs of subscription-based merchants.

Error Handler (7)

The error handler module is responsible for managing situations such as timed-out transaction responses and any other system errors related to incorrect API calls, failed merchant verification, or failed fraud checks.

Acquiring Bank communication (8)

The acquiring partner bank API is incorporated directly into the transaction engine. Since there is no industry standard for acquiring bank APIs comparable to ISO 8583, we chose to integrate the API directly into the engine. This approach offers a speed advantage, though changing acquiring partners requires updating the API code inside the transaction engine. Other payment gateways introduce an additional abstraction layer for bank APIs that avoids recompiling the engine when the partner changes; we chose to prioritize simplicity and speed instead.

Merchant Portal

A merchant portal for a payment gateway is a web-based application that allows merchants to manage their payment processing activities. It provides a centralized platform for merchants to view and manage transaction data, track sales and revenue, and perform other important tasks related to payment processing.

Payment Terminal

A web-based payment terminal is a payment processing solution accessible through a web browser. This type of terminal allows merchants to accept online payments without the need for a physical point-of-sale (POS) system. With a web-based payment terminal, customers make payments by entering their payment information directly into a secure web form, which the payment gateway then processes.

Admin Interface

A payment processor administrative interface is a web-based application that allows payment processing companies to manage their operations. It provides a centralized platform for payment processors to monitor and manage transactions, configure payment settings, add merchants, and perform other administrative tasks.

Note on standard

While banking systems operate under an international standard (ISO 8583), no equivalent standard exists for online payment processing. Every payment processor implements its own API, transport, and security model. Over the past twenty years of web commerce growth, Visa and MasterCard have not introduced a standard API for processing online transactions; the only universal requirement is PCI certification. The absence of a standard gives payment processors the flexibility to develop their own APIs based on the internal decisions that best suit their goals.

Merchant Model

The Templado System platform supports the following merchant models:

Direct Merchant

The Direct Merchant ID (DMID) model is a payment processing model in which a merchant has a direct relationship with a payment processor. Under the DMID model, the merchant is typically required to obtain their own merchant account with a payment processor. The merchant account is linked to the DMID, which serves as a unique identifier for the merchant when processing transactions and settling funds.

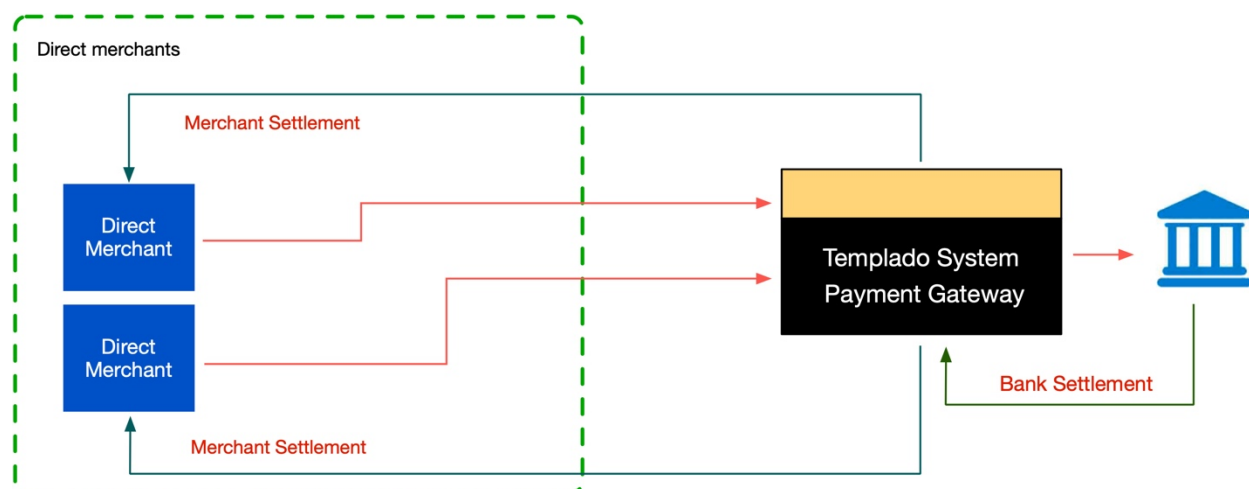


Figure 2. Direct Merchant Model

Aggregation Merchant

The merchant aggregation model is a business model in which a company acts as an intermediary between multiple merchants and payment processor. A Master Merchant ID (MMID) is a unique identifier assigned to an aggregator and all of its sub-merchants or merchants, who are working under the aggregator's umbrella. The MMID is used by the aggregator to manage the payment processing for all of its sub-merchants. When a customer makes a payment to one of the sub-merchants, the payment is collected by the aggregator, who then disburses the funds to the appropriate sub-merchant account.

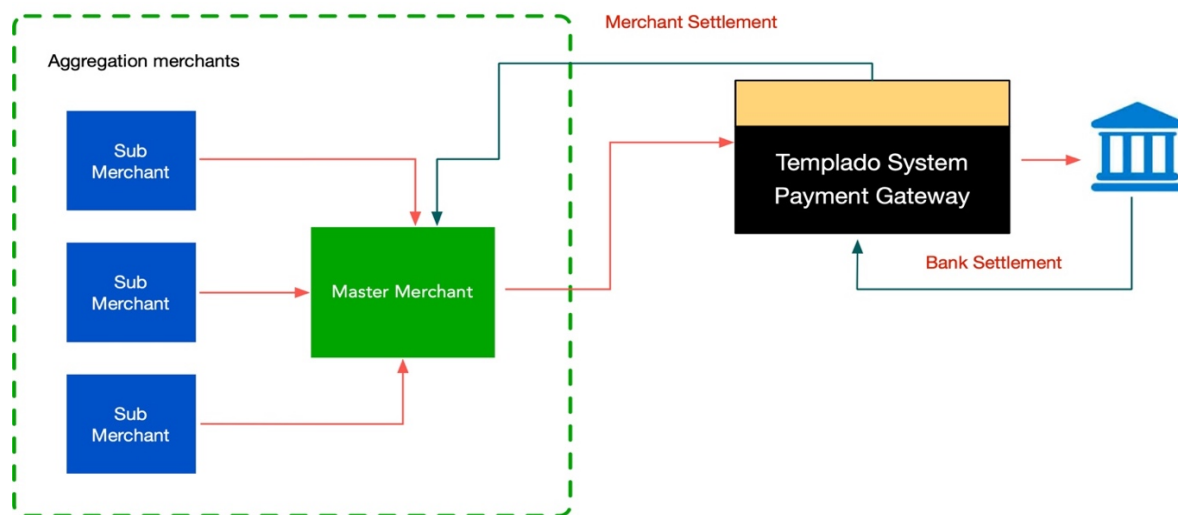


Figure 3. Aggregation Merchant Model

Hybrid Merchant

In the hybrid model, Templado System allows direct merchants and aggregation merchants to be combined within the same deployment.

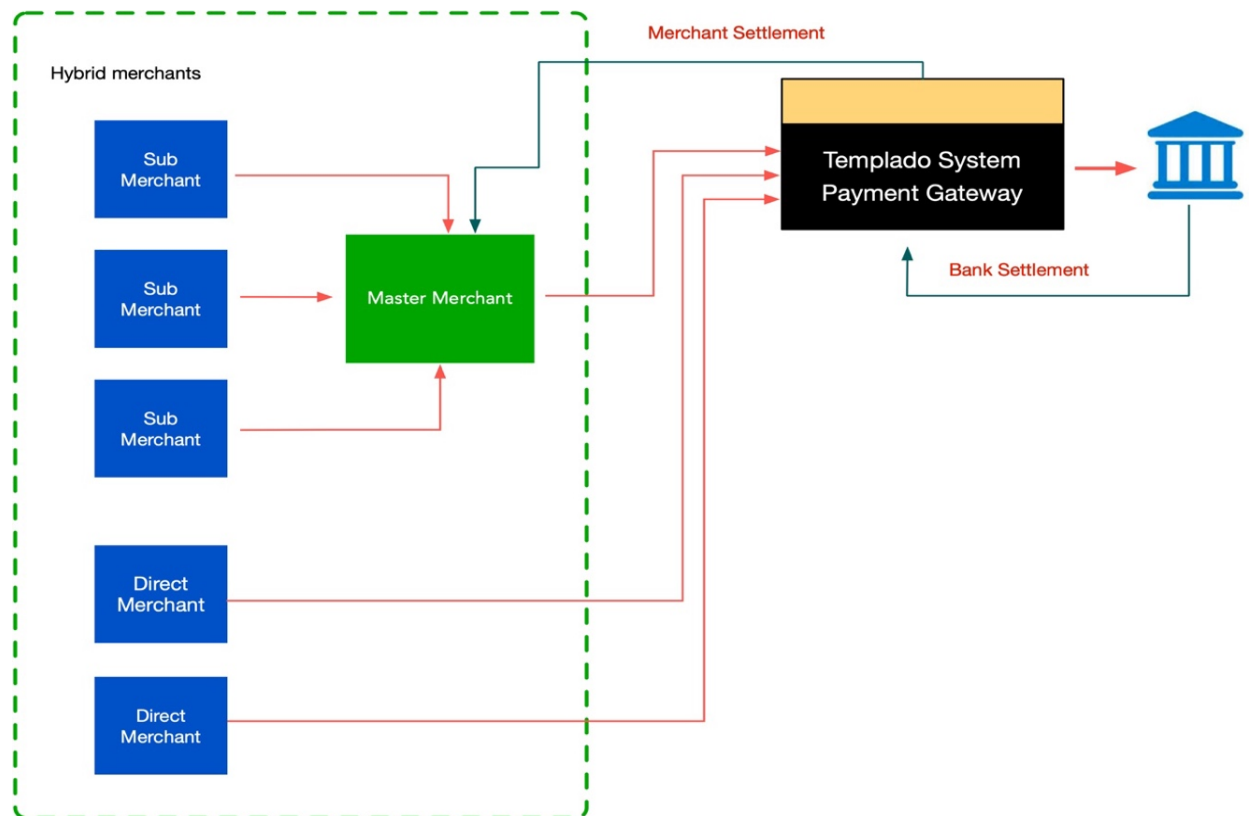


Figure 4. Hybrid Merchant Model

Testing

On premises systems

1. ARM Cortex A73 CPU, 8GB RAM
2. Ampere Ultra Ampere Altra 32 Core, 32GB RAM, Samsung 970 NVMe, Ubuntu 22.04 Server
3. Ampere Ultra Ampere Altra 64 Core, 32GB RAM, Samsung 970 NVMe, Ubuntu 22.04 Server
4. Dell Power Edge 650. Intel Xeon 6338, 32GB RAM, SATA SSD RAID 5, Microsoft 2019 Server
5. EPYC 7763, 32GB RAM, SATA SSD RAID 5, Microsoft 2019 Server

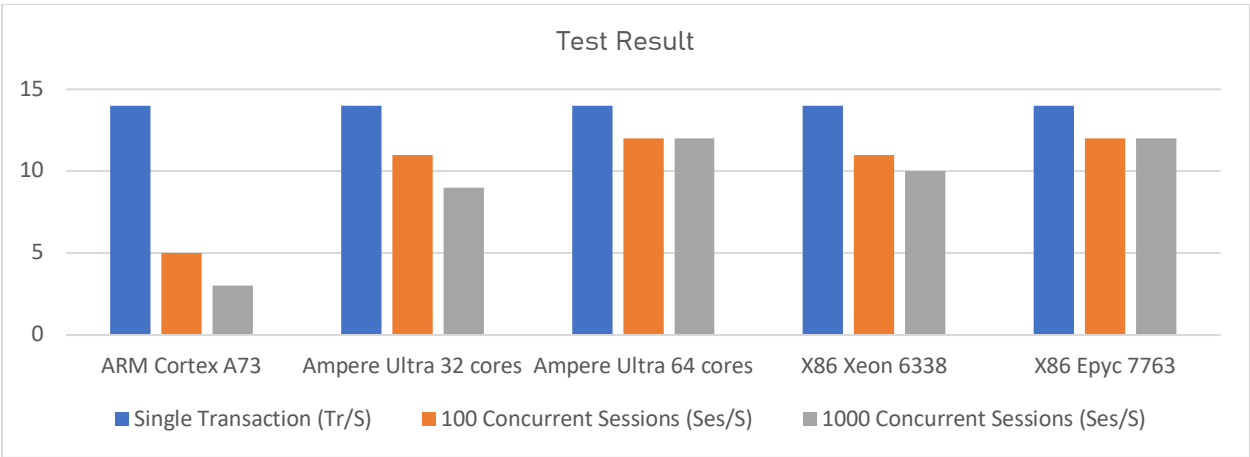
We used an Ampere Altra development workstation from Adlink Technology.

<https://www.ipi.wiki/products/ampere-altra-developer-platform?variant=41995037868194>

Cost:

1. Ampere Ultra 32 Core \$3,250
2. Ampere Ultra 64 Core \$4,200
3. Intel Xeon \$6,029

4. AMD Epyc \$7,890



A single session with a single transaction completes in approximately 64ms on any of the tested systems; the gateway takes roughly 64ms to process a single transaction, and all systems perform comparably.

Next, we simulated 100 concurrent sessions from Apache passing transactions to the transaction engine. The 32-core Ampere Altra performed on par with the Intel Xeon, though slower than the AMD EPYC. The 64-core Ampere Altra slightly outperformed the Intel Xeon and performed roughly on par with the AMD EPYC.

Finally, we simulated 1,000 concurrent sessions from Apache passing transactions to the transaction engine. In this test, the 32-core Ampere Altra was slower than both the Xeon and the EPYC, but the 64-core Ampere Altra clearly outperformed the Intel Xeon and matched the AMD EPYC.

A 128-core Ampere Altra server was not available for testing in our scenario, nor was the newer Ampere One (192 cores).

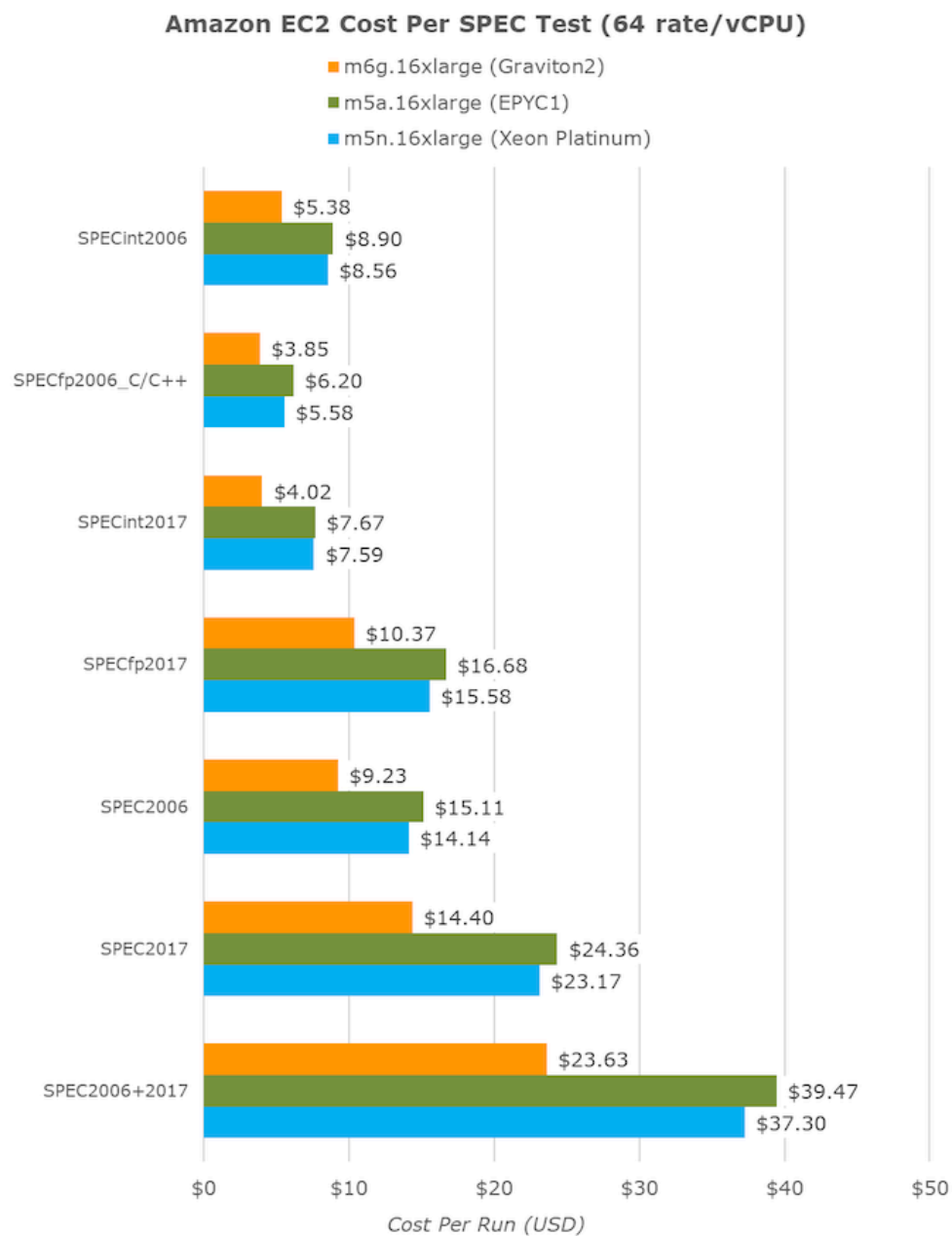
Amazon Graviton

AWS Graviton processors are designed by Amazon to deliver strong price-performance for cloud workloads running on Amazon EC2. Since Amazon does not sell Graviton CPUs for building standalone servers, this test was conducted using an AWS EC2 instance. For this test, the minimum available Graviton configuration was chosen: an EC2 t4g.small instance with 2 cores and 1GB of memory. The instance was capable of running Ubuntu 22.04 with the Apache 2 web server, where the engine received live external transactions to simulate a real merchant experience. The engine was connected to MariaDB running on a separate instance, to simulate the separation of the database server from the engine server.

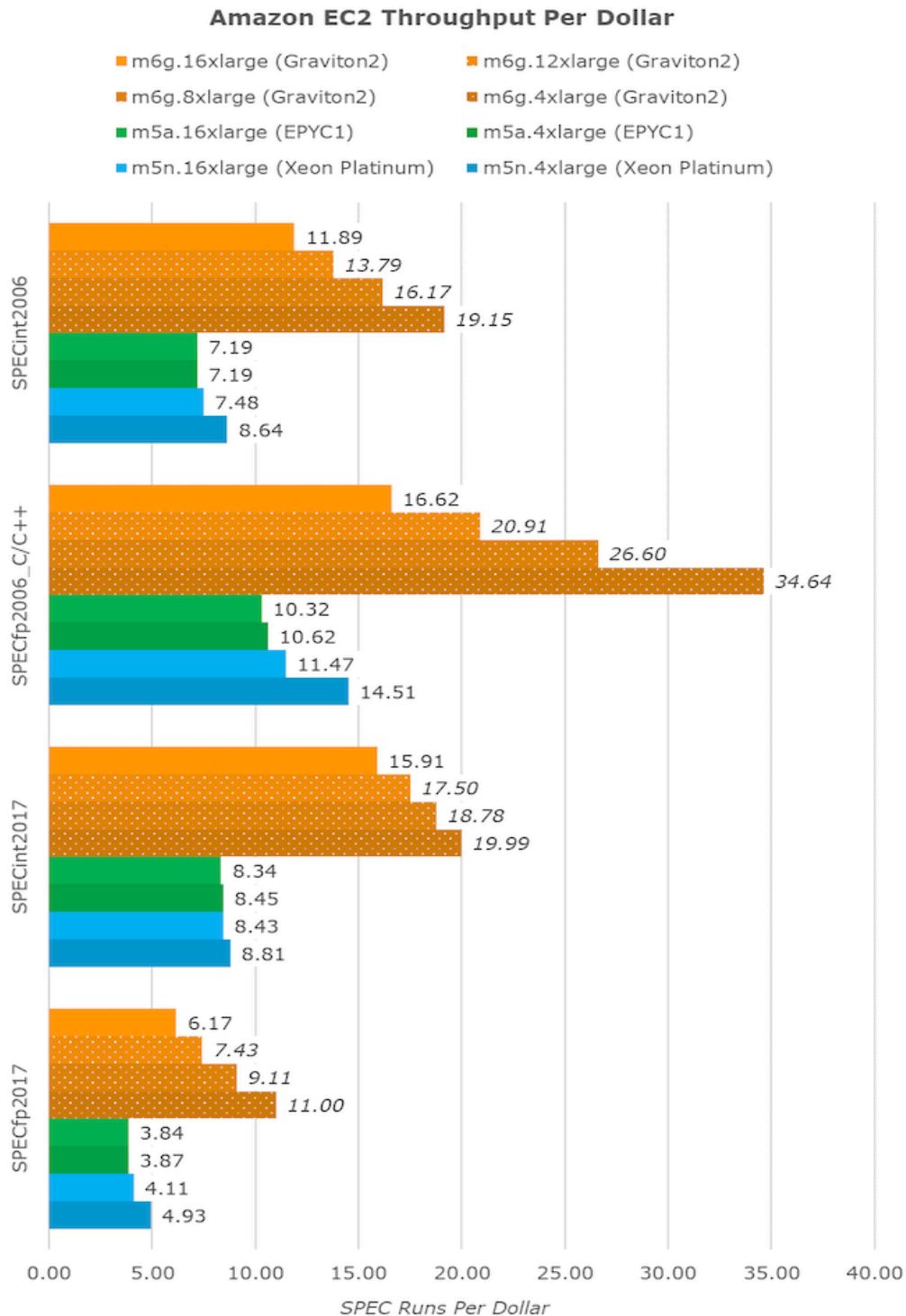
Graviton-2 result
8 transactions per second on the EC2 t4g.small ARM configuration.

ARM On Cloud

All major cloud providers—AWS, Azure, Oracle, and Google—offer ARM instances. Amazon's instances are based on its Graviton ARM CPU, while the other providers offer instances built on Ampere ARM CPUs.

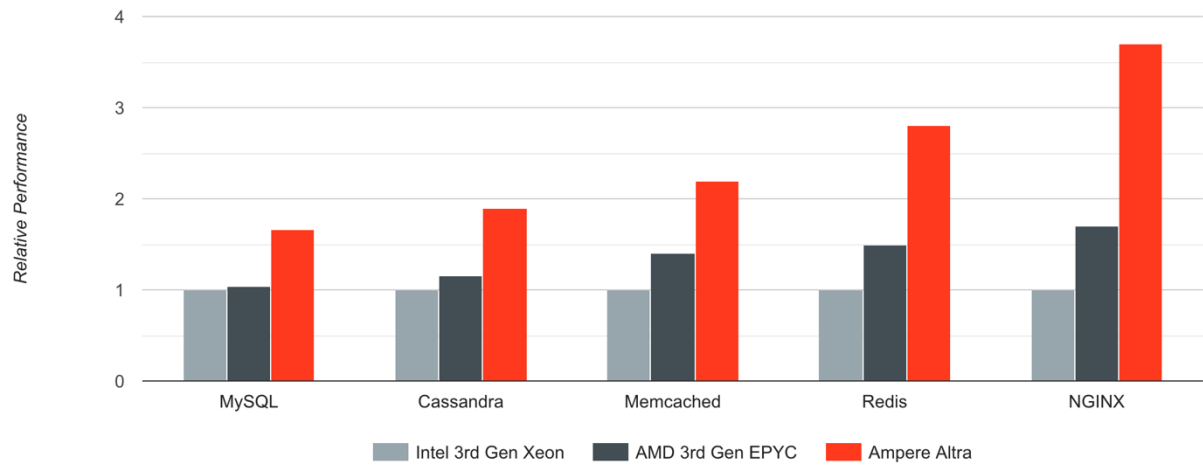


The Graviton2 demonstrated that it can keep up extremely well in terms of performance and throughput, even beating the competition in many of the tests. However, performance is not always the primary concern—sometimes the goal is simply to complete a workload in the most cost-effective way possible, at which point value becomes the deciding factor.



Amazon supports this claim, stating that the Graviton2 chip achieves 40% better performance per dollar than the competition. As covered earlier, for the 64-vCPU 16xlarge instance size, the m6g (Graviton2), m5a (EPYC1), and m5n (Xeon Cascade Lake) instances are priced at an hourly cost of \$2.464, \$2.752, and \$3.808, respectively.

Aggregating all workloads together—intended to produce a representative figure across a wide variety of real-world use cases—shows the Graviton2 coming in 40% cheaper than the competing platforms, an outstanding result.



Comparisons between the major legacy X86 architectures and the Ampere Altra Max Cloud Native Processor running on five of the most popular cloud native workloads.

Database on ARM

Platform uses MariaDB as main database. MariaDB is designed for high performance and can handle large volumes of data and transactions efficiently. This makes it a good choice for payment processing applications that require fast and reliable performance. MariaDB supports features such as replication and clustering, which can help ensure high availability and uptime for payment processing applications. This can help minimize downtime and ensure that payment processing is always available to customers. It is open source and free to use, which can help reduce the cost of payment processing for businesses. MariaDB includes built-in security features such as SSL/TLS encryption, secure password storage, and access controls. This can help ensure that sensitive payment data is protected from unauthorized access and theft.

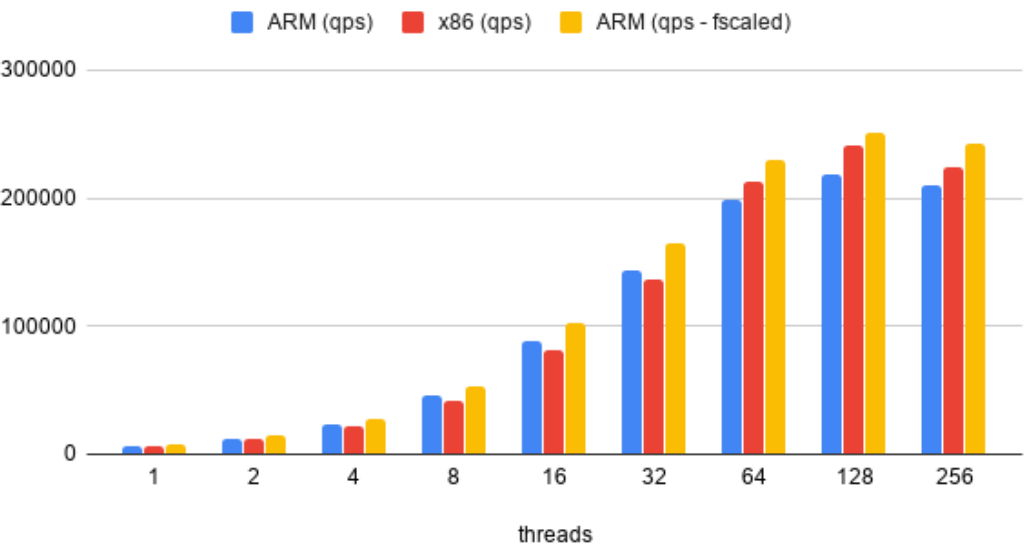
Performance

24 vCPU/48 GB Intel(R) Xeon(R) Gold 6266C CPU @ 3.00GHz for running MySQL on x86.

24 vCPU/48 GB ARM @ 2.60GHz for running MySQL on ARM

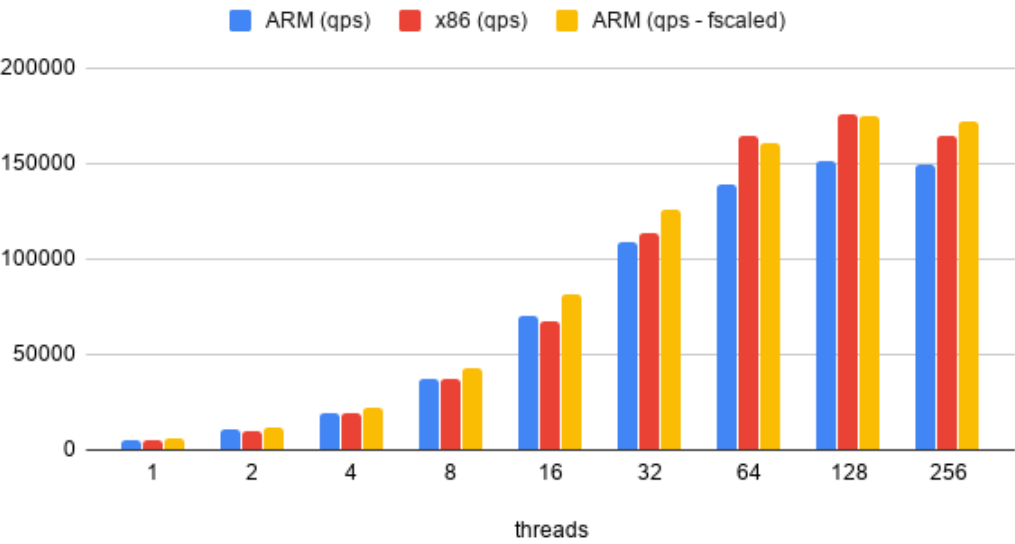
Point Select

ARM vs x86 (point-select)



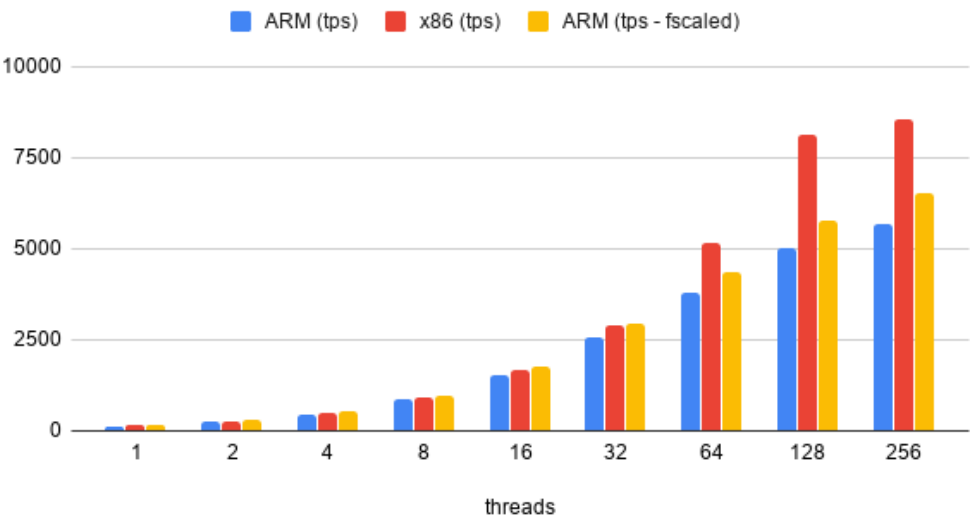
Read Only

ARM vs x86 (read-only)



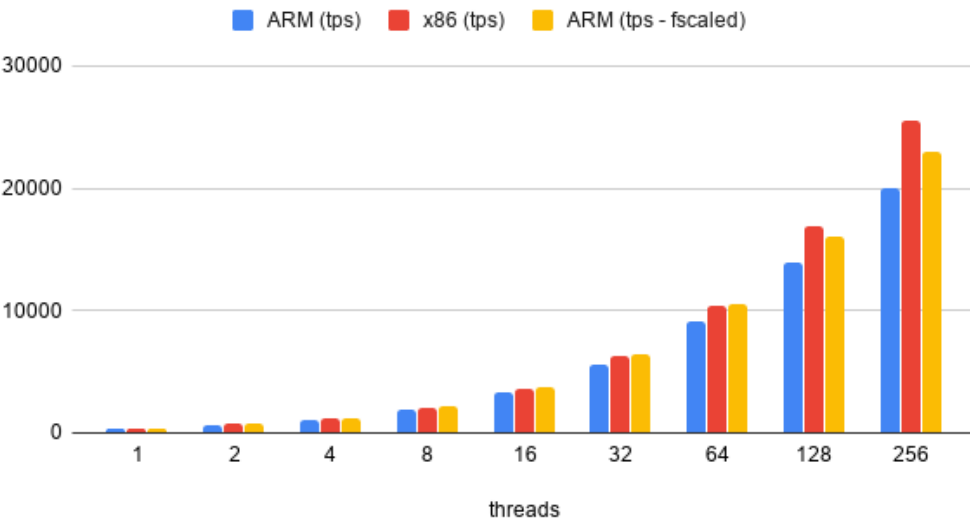
Read Write

ARM vs x86 (read-write)



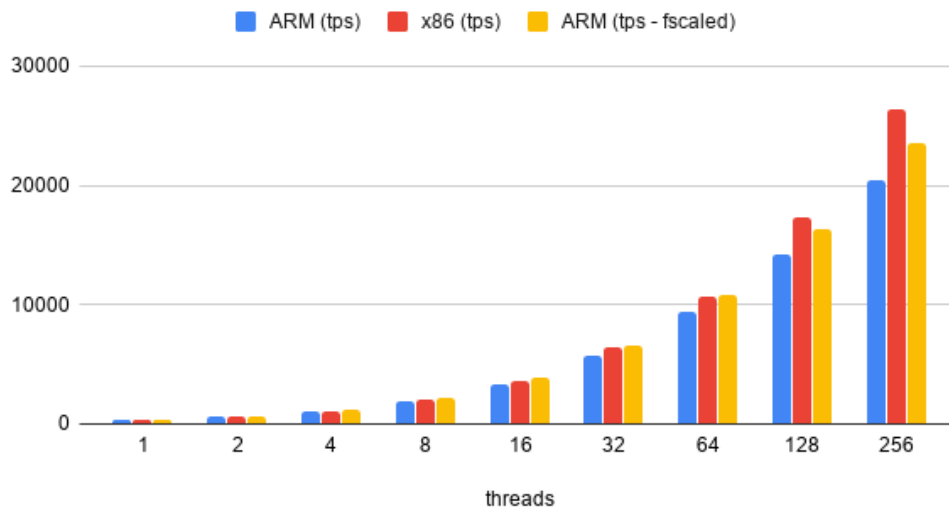
Update Index

ARM vs x86 (update-index)



Update Non-Index

ARM vs x86 (update non-index)



Conclusion

There are several important observations

For read-only workloads, MySQL on ARM continues to perform on par with MySQL on x86.

For write-involving workloads, MySQL on ARM begins to lag slightly, though frequency scaling narrows the gap considerably.

Since frequency scaling is not always a realistic deployment parameter, the price-per-performance ratio is a more useful metric to consider: the ARM instance used in this test (24 vCPU/48GB) is 66% cheaper than the equivalent x86 instance.

A clear pattern emerges from this testing: ARM workloads scale well until they approach CPU limits, at which point increasing contention causes ARM to start lagging. This is expected, since mutex and contention hot-spots (such as spin-locks) have historically been tuned for x86. Now that MySQL officially supports ARM, and the ARM ecosystem and community continue to grow, similar tuning for ARM is expected to follow.

In summary, MySQL on ARM is a compelling option to evaluate from both a cost and performance perspective.

The cost-effectiveness and resource optimization advantages of ARM-based payment gateways make them a compelling choice for businesses seeking to maximize their return on investment. By adopting ARM architecture, organizations can reduce infrastructure costs, lower energy consumption, and optimize resource utilization, leading to significant cost savings. The ARM server ecosystem has been growing steadily, with an increasing number of software vendors and operating systems providing support for ARM-based servers. This expanding ecosystem ensures compatibility and access to a wide range of software, further enhancing the value proposition of ARM servers.

This white paper has highlighted the benefits of ARM-based technology in achieving cost-effectiveness and resource optimization in payment gateway systems.

About Templado System

The Templado Payment Platform is a payment processing platform built on ARM architecture. It is designed for data processing environments that require the highest throughput and lowest latency.

Templado System is headquartered in 80 Corporate drive Suite 309, Scarborough, Ontario M1H 3G8 Canada. To learn more about Templado System visit <https://templadosystem.com> or contact us at info@templadosystem.com to get more information.