Transition story

x64 to ARM

RDS MySQL to Aurora MySQL

#12

VilniusPHP 2023-06-01

DARIUS LESKAUSKAS

CTO @ frontu.com

linkedin.com/in/darius-leskauskas

Frontu

Keep your field service operations running smoothly

Technician-first service execution software for maintenance management

Nestlé · Vilniaus vandenys · Lietuvos paštas · CAPS group Maximising Customer Experience · Circle K · Coffee Address

Migration to ARM

There was a need to choose new instance types for migration to web / worker architecture.

AWS Graviton

“AWS Graviton2 instances, which are ARM-based, have shown improved performance in various workloads and have resulted in cost savings for customers. They offer up to 40% better price performance compared to x86-based instances, enabling customers to significantly reduce their compute costs”

AWS Graviton

“AWS Graviton processors are designed to be more energy efficient, using up to 60% less energy for the same performance compared to comparable EC2 instances. This helps customers reduce their carbon footprint”

x86/x64 vs ARM

- **(Intel vs AMD) vs ∞**
  ARM = License != Chip

- **CISC vs. RISC**
  Complex Instruction Set Computer vs Reduced Instruction Set Computer

CISC

- Complex and diverse instruction set.
- Instructions can perform multiple operations in a single instruction.
- Emphasizes code density and supports a wide range of addressing modes.
- Execution time per instruction can vary.
- Typically found in older processors and offers backward compatibility.
- Suitable for applications that require complex operations and variable-length instructions.

RISC

- Simple and standardized instruction set.
- Instructions typically perform only one operation per instruction.
- Focuses on executing instructions quickly and efficiently.
- Fixed execution time per instruction.
- Often found in modern processors and offers better performance.
- Suitable for applications that require fast and efficient execution, such as embedded systems and mobile devices.

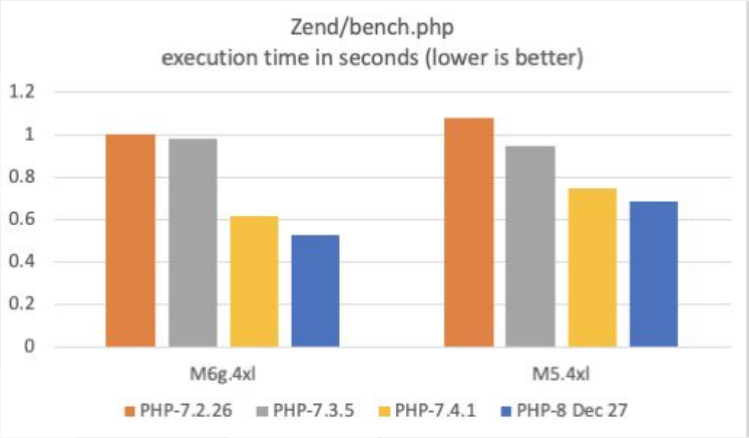
Cloud goes ARM

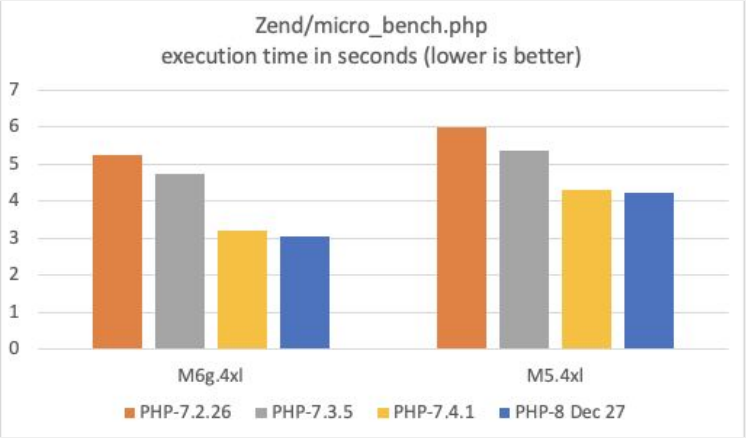
- **Graviton2, Graviton3**
  AWS
- **Tau T2A**
  Google Cloud
- **Ampere Altra**
  Azure, Oracle
- **Yitian 710**
  AlibabaCloud will have 20% of its server fleet by 2025

AWS Graviton 3

“AWS Graviton3 provide up to 25% better compute performance, up to 2x higher floating-point performance, and up to 2x faster cryptographic workload performance compared to AWS Graviton2 processors. AWS Graviton3 processors deliver up to 3x better performance compared to AWS Graviton2 processors for ML workloads, including support for bfloat16. They also support DDR5 memory that provides 50% more memory bandwidth compared to DDR4.”

Performance: 37% faster





Contribution to PHP Core

AWS contributions to PHP-7.4

Function	Speedup	Commits to PHP-7.4
inc/dec	1.5x	https://github.com/php/php-src/pull/4094
add/sub	1.82x	https://github.com/php/php-src/pull/4095
hash_init	1.61x	https://github.com/php/php-src/pull/4096
hash_vect	1.72x	https://github.com/php/php-src/pull/4126
crc32	29x	https://github.com/php/php-src/pull/4108
rev64	7.8x	https://github.com/php/php-src/pull/4109
base64 encode	3.5x	https://github.com/php/php-src/pull/4381
base64 decode	2.15x	https://github.com/php/php-src/pull/4381
string addslashes	2.8x	https://github.com/php/php-src/pull/4396
string stripslashes	4.9x	https://github.com/php/php-src/pull/4396

Migration result

- **No compatibility issues with standard list of modules**
  Even external libXl module is working!
- **No need to change configuration**
  Same nginx, php, php-fpm configs
- **Decreased ~ response time**
  From ~150ms -> ~110ms & -> ~100ms after PHP 8.0 -> 8.2 upgrade
- **Saved € & Lowered carbon footprint**
  Karma++

Sources

- https://www.zend.com/blog/php-arm-architecture-support
- https://aws.amazon.com/blogs/compute/improving-performance-of-php-for-arm64-and-impact-on-amazon-ec2-m6g-instances/
- https://www.toptal.com/back-end/arm-servers-armv8-for-datacentres

Migration to AWS Aurora

There was a need to migrate to MySQL 8 & choose a new instance type to scale up DB

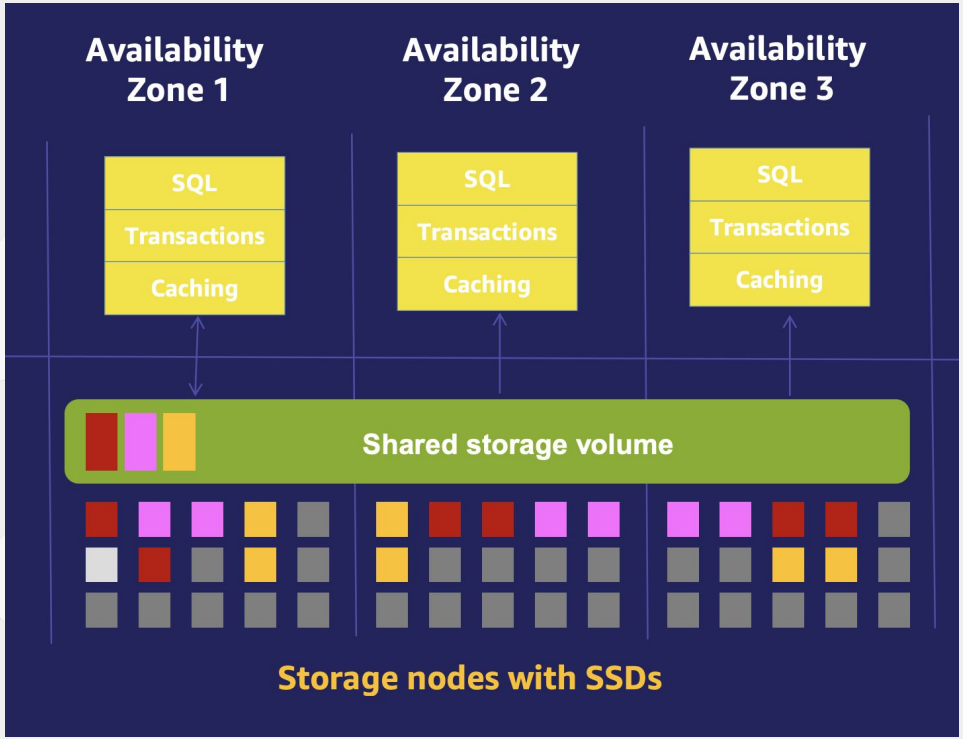
Why AWS Aurora?

01. x2 RAM
16GB vs 8GB comparing with RDS MySQL instance

02. Easy replication & configuration
Everything is done through AWS Console

03. Built for cloud
Why bother?

How it works?


Availability Zone 1
Availability Zone 2
Availability Zone 3
SQL
Transactions
Caching
SQL
Transactions
Caching
SQL
Transactions
Caching
Shared storage volume
Storage nodes with SSDs

Aurora Pros & Cons

- Fast & Automatic Failover
- x5 better throughput
- Availability and Durability

- No possibility of additional extensions
- Need to wait for ported MySQL updates
- A bit more expensive

Nice features

- **Reader endpoint as load balancer**
  No need to handle load programmatically, easy setup for Doctrine
- **Easy backup/restore**
  No impact to performance during backup process
- **Autoscaling**
  Add additional readers automatically
- **Blue/Green Deployment**
  Experiment with configuration on production data

Migration process

- **With near-zero downtime**
  Create an Aurora Read Replica from the source RDS DB instance & move traffic.

- **With downtime**
  Create from snapshot.

Serverless?

- **Much more expensive if used not by the purpose**
  Micro instances are cheaper
- **For specific cases only**
  Need to make reports periodically?

Thank you!

linkedin.com/in/darius-leskauskas