AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

XNT401

Build high-performance .NET serverless architectures on AWS

Andy Hopper (he/him)
Principal Specialist Solutions Architect, Microsoft Technologies
Amazon Web Services

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

How AWS Lambda executes your code

Using instrumentation to identify performance “hot spots”

Optimizing your code to get the best price/performance

Minimizing the impact of cold starts

Tips and tricks

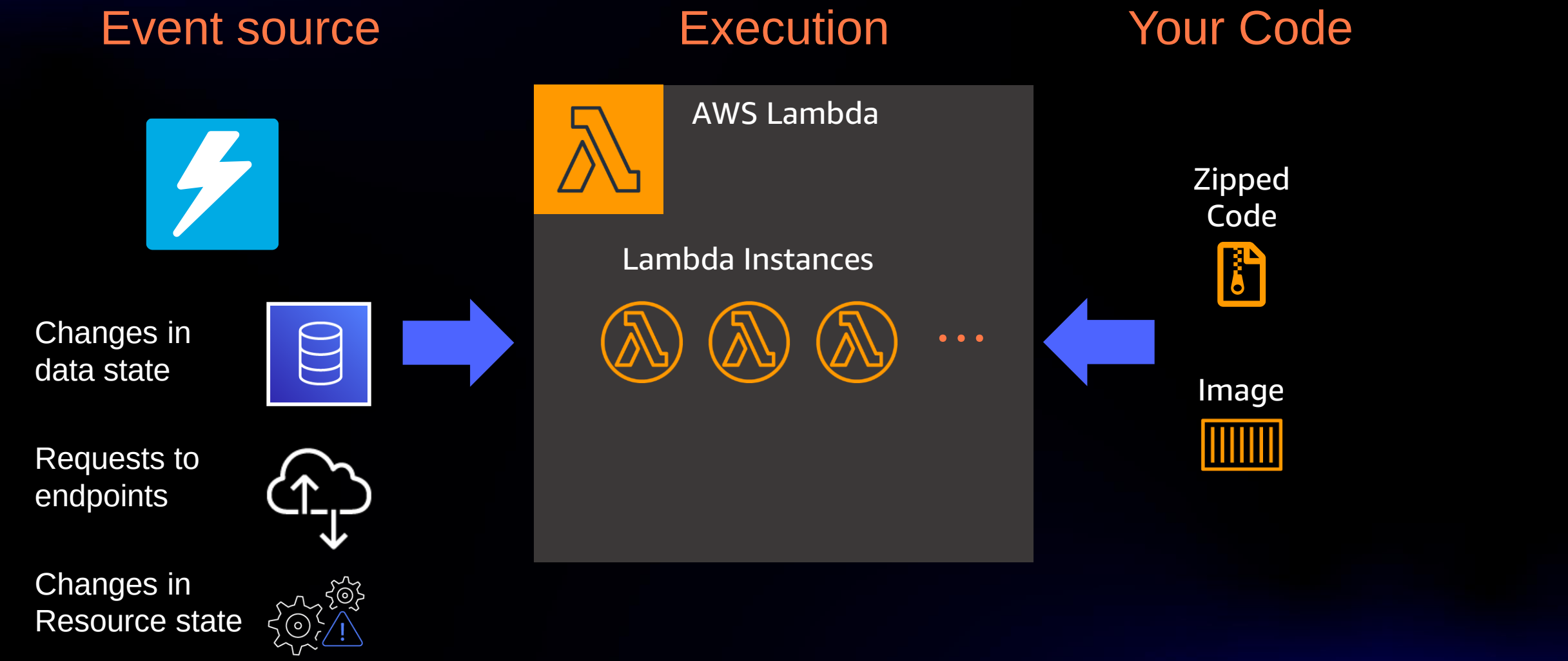
Demos

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How does AWS Lambda run your code?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda architecture



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Firecracker microVM

MicroVM technology

Minimalist dependencies

Low-memory footprint

Container support



Firecracker

Read more at: https://firecracker-microvm.github.io/

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Code delivery approaches

.zip file

- Uses an AWS-supplied runtime environment
- Provide a .zip file containing your application's code
- Indicate which assembly/class/method implements your handler

Custom runtime

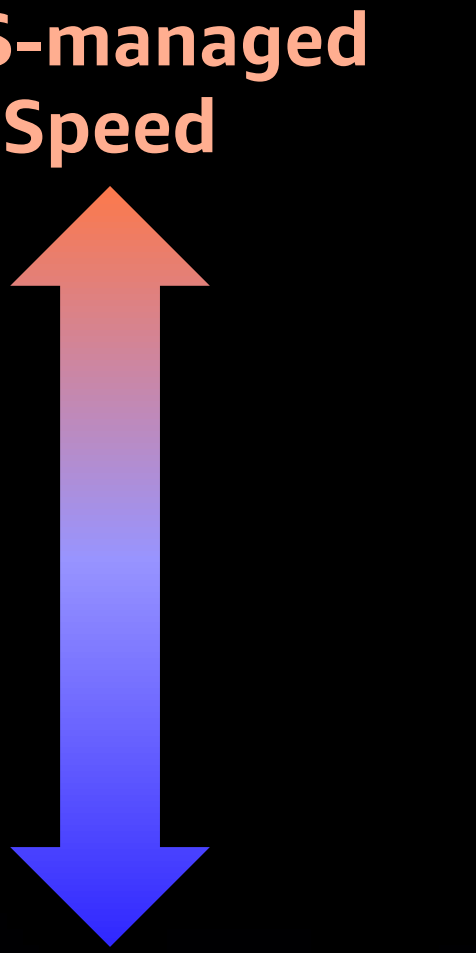
- You provide the runtime and code
- Provide a .zip file with an executable file named "bootstrap"

Layers

- Helps with code reuse
- Can be used to ship custom runtimes, libraries, data

Images

- Create a Docker image containing your code
- You provide the runtime



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Monitoring your AWS Lambda performance

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Instrument your code to track performance

- Amazon CloudWatch metrics
- AWS X-Ray
- Amazon CodeGuru Profiler

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using Amazon CloudWatch

Amazon CloudWatch

- Log stream support included by default
- Automatically collected metrics track duration/performance

Amazon CloudWatch Lambda Insights

- Enabled as an AWS Lambda layer
- Captures detailed runtime data

Amazon CloudWatch Synthetics

- Uses “canaries” to track behavior/latency
- Executes requests against API endpoints

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS X-Ray to analyze performance

AWS X-Ray

- Automatically participates in existing traces
- Can start a trace by enabling active tracing
- Use custom segments to gather detailed data about your code paths



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Optimizing your code for AWS Lambda

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Structuring your code

AWS Lambda has three lifecycle phases

- Init – ensures the execution environment is ready
  - Extension init
  - Runtime init
  - Function init
- Invoke – the execution of your handler
- Shutdown – clean shutdown of the execution environment

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Taking advantage of the init phase

Init only executes once for the lifetime of your function's instance

- Loads your handler's assemblies
- Ideal for loading configuration, secrets, etc.
- Has a temporary CPU "boost"

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Maximizing throughput

Leverage async/await

- Database queries
- Disk I/O
- AWS SDK calls
- Identify code that can wait concurrently

Parallelize your code

- QueueUserWorkItem, Parallel.ForEach
- Benefits the most from >1 vCPU

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon CodeGuru Profiler

Amazon CodeGuru Profiler

- Currently Java and Python only
- Uses runtime data collection to pinpoint bottlenecks

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Minimizing cold starts

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ReadyToRun

Can avoid the need to JIT through ahead-of-time compilation

- Can dramatically reduce startup time due to JIT compilation
- In versions prior to .NET 6, you must compile on the same OS as the target deployment platform (e.g., AWS Lambda == Linux)
- Results in larger assembly sizes

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Provisioned concurrency for AWS Lambda

- Simple config to avoid cold starts
- No more manual prewarming
- Predictable performance during spikes
- Good fit for latency-sensitive applications
- No code changes required
- Bound to version or alias

- AWS CloudFormation support
- Scheduling with AWS Application Auto Scaling
- From 1 to account concurrency limit
- Two new metrics
- Ramp-up time (500 per minute)
- Initialization code is executed automatically

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tips and tricks

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Caching data

Amazon ElastiCache

- Submillisecond latency
- Note: requires VPC connectivity

Leverage the init phase by preloading ”lookup tables”

Use in-memory caching

- An AWS Lambda environment spans requests
- Beware of concurrency – don’t assume the next request uses the same instance!

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Higher performance doesn’t always cost more

Compute all prime numbers **<= 1M** **100 times**

256 MB	14,675 msec	$0.0061146
512 MB	7,233 msec	$0.0060275
1,024 MB	3,655 msec	$0.0060917
2,048 MB	3,096 msec	$0.0103200

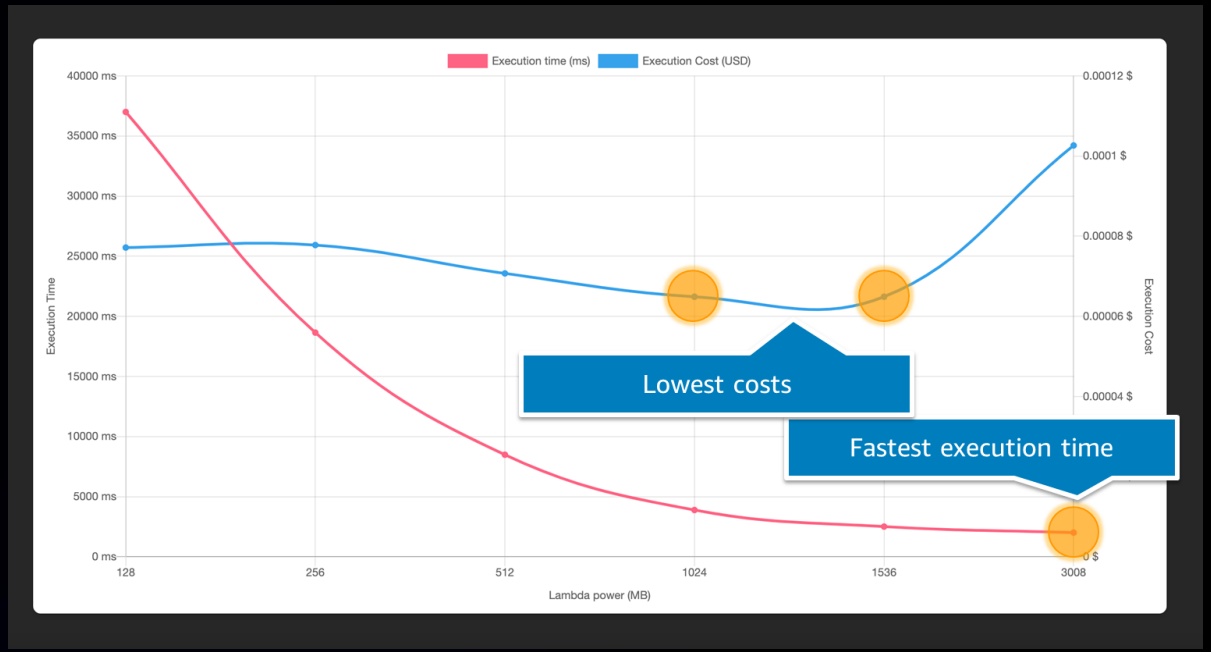
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda Power Tuning

Uses AWS Step Functions to run multiple concurrent versions of an AWS Lambda function

Compares the performance of different memory allocations

Can be integrated into your CI/CD process



Available at https://github.com/alexcasalboni/aws-lambda-power-tuning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Running your code on ARM64 with AWS Graviton2

- 20% savings over x86
  - x86: $0.0000166667 per GB/s
  - ARM64: $0.0000133334 per GB/s
- Test your workload
  - Choose the processor type to match the use case

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Compare duration costs

Compute all prime numbers **<= 1M** **100 times**

x86	1,024 MB	3,655 msec	$0.0060917
	2,048 MB	3,096 msec	$0.0103200
ARM64	1,024 MB	4,910 msec	$0.0065467
	2,048 MB	2,840 msec	$0.0075734

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Key takeaways

- You can achieve significant performance improvements by understanding the execution environment
- Improve your function’s throughput using parallelism
- Instrument your code to monitor changes over time
- ARM64 with AWS Graviton2 CPUs can give you better price/performance
- Test, test, test!

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Related sessions

XNT301	What’s new with .NET development and deployment on AWS
XNT303	Accelerate .NET 6 performance with ARM64 on AWS Graviton2
SVS401	AWS Lambda function performance tuning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.