
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

SVS305-R

Understanding AWS Lambda concurrency

Brian Zambrano (he/him)

Sr. Specialist Solutions Architect, Serverless
Amazon Web Services

Justin Pirtle (he/him)

Principal Solutions Architect
Amazon Web Services

“We want a serverless architecture that supports 10,000 requests per second.”

The business

While keeping costs under control,

with good performance,

and integration with our relational database

AWS Lambda concurrency affects all of this

Agenda

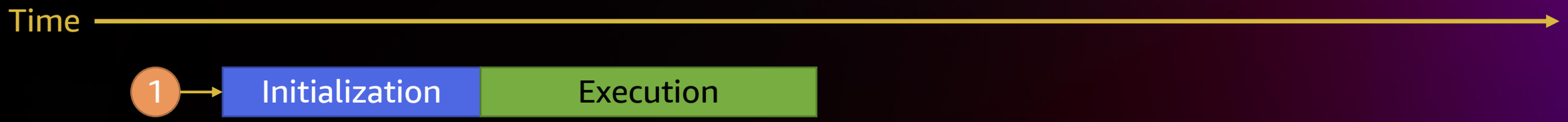
What we will cover

- AWS Lambda concurrency and concurrency controls
- Applying concurrency to a sample serverless API architecture
- Best practices for scaling synchronous and asynchronous workloads

What we will not cover

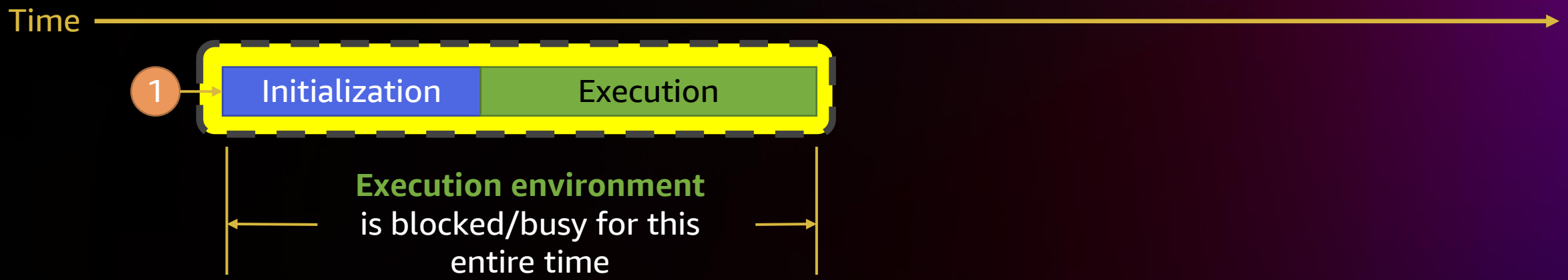
- What is serverless?
- What are Amazon API Gateway, Amazon DynamoDB, AWS Lambda?

Lambda concurrency

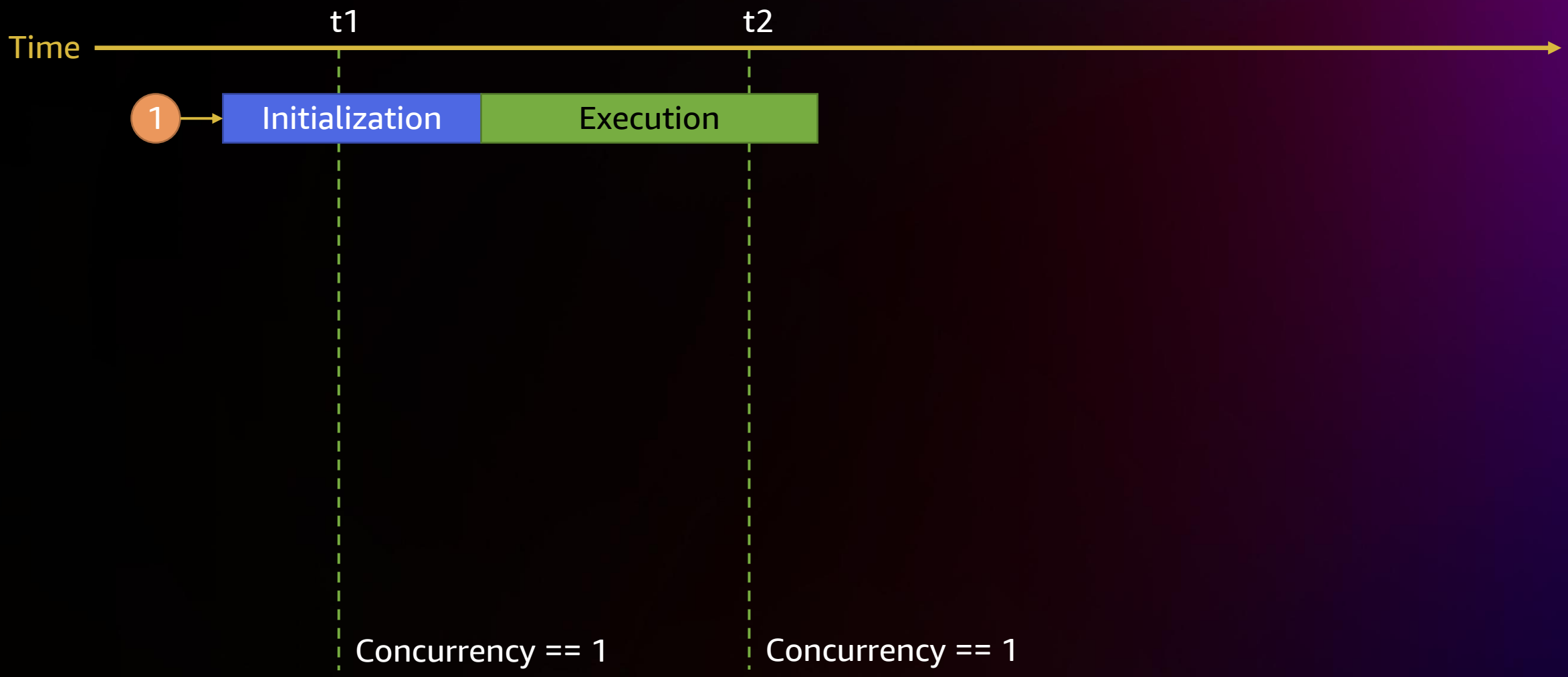


Lambda concurrency

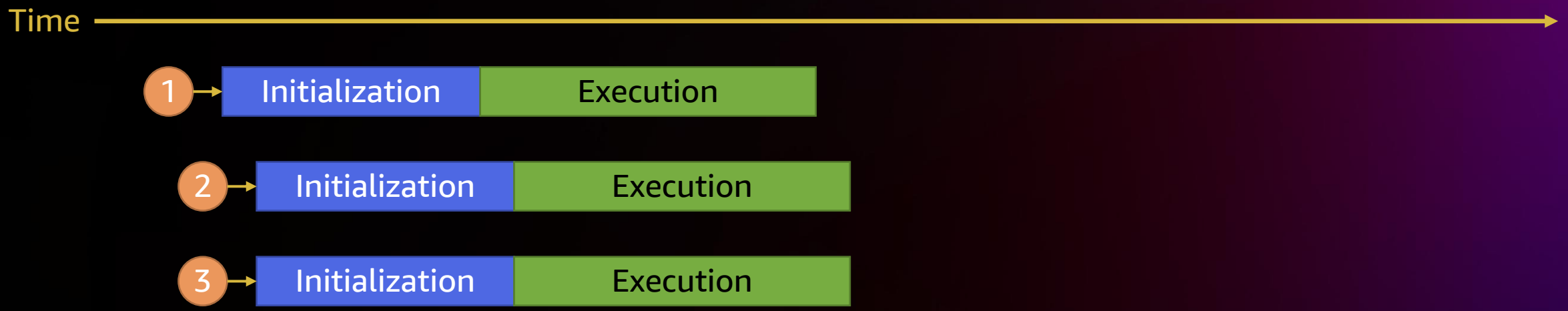
Lambda concurrency



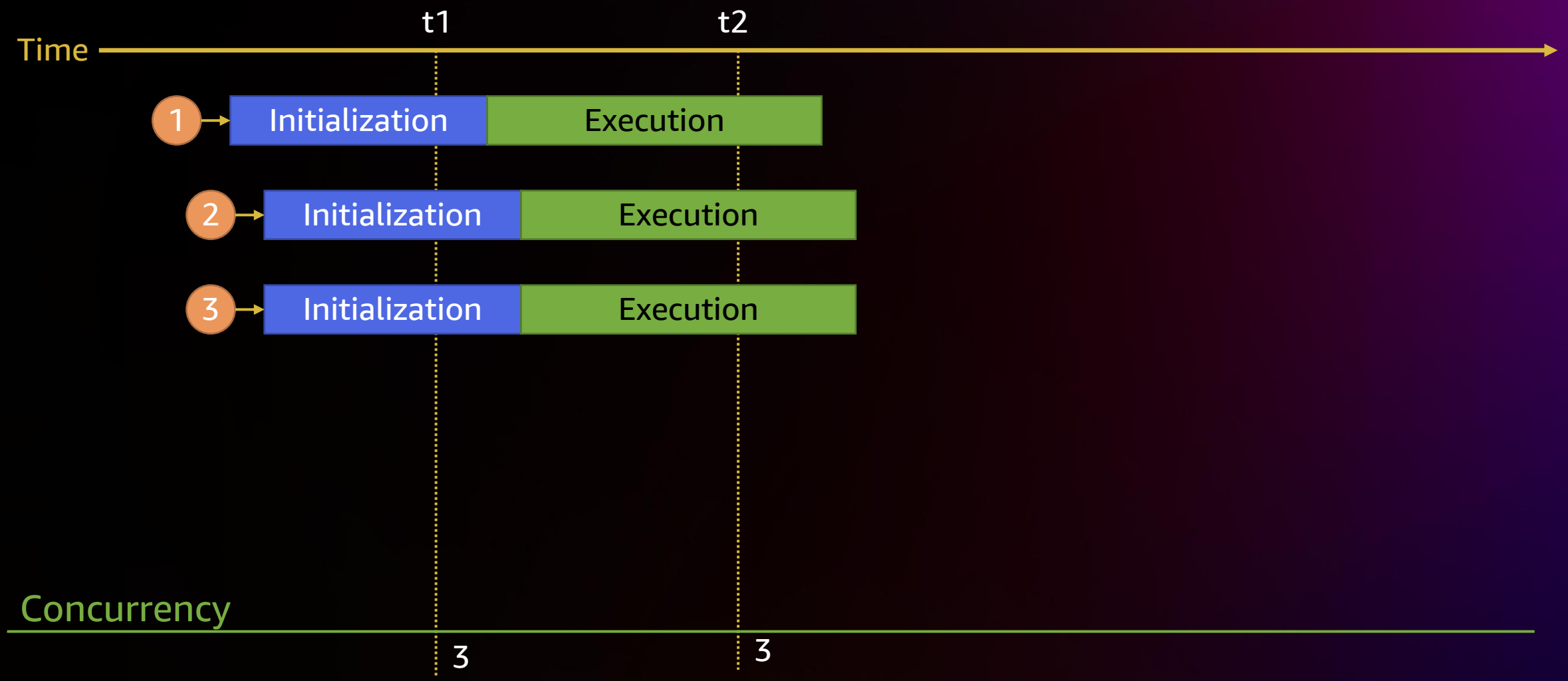
Lambda concurrency



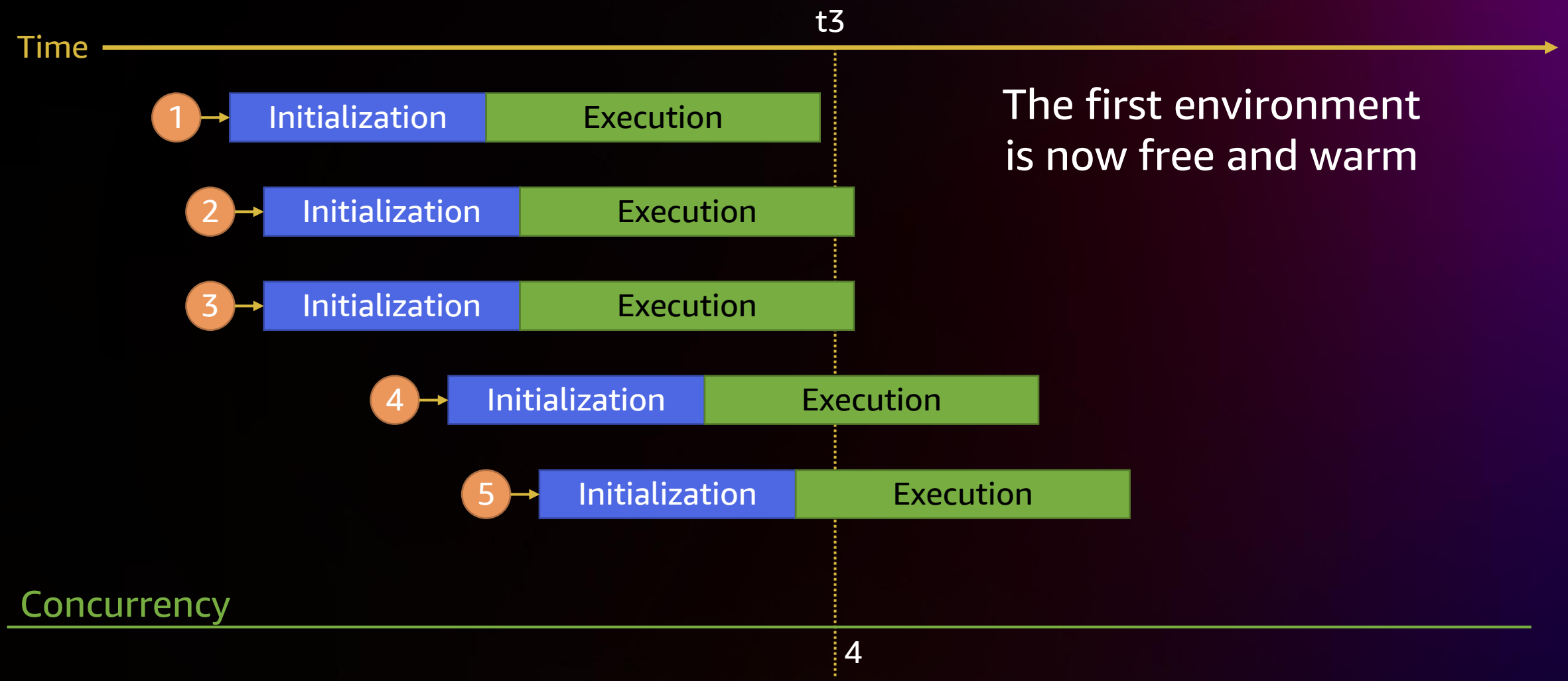
Lambda concurrency



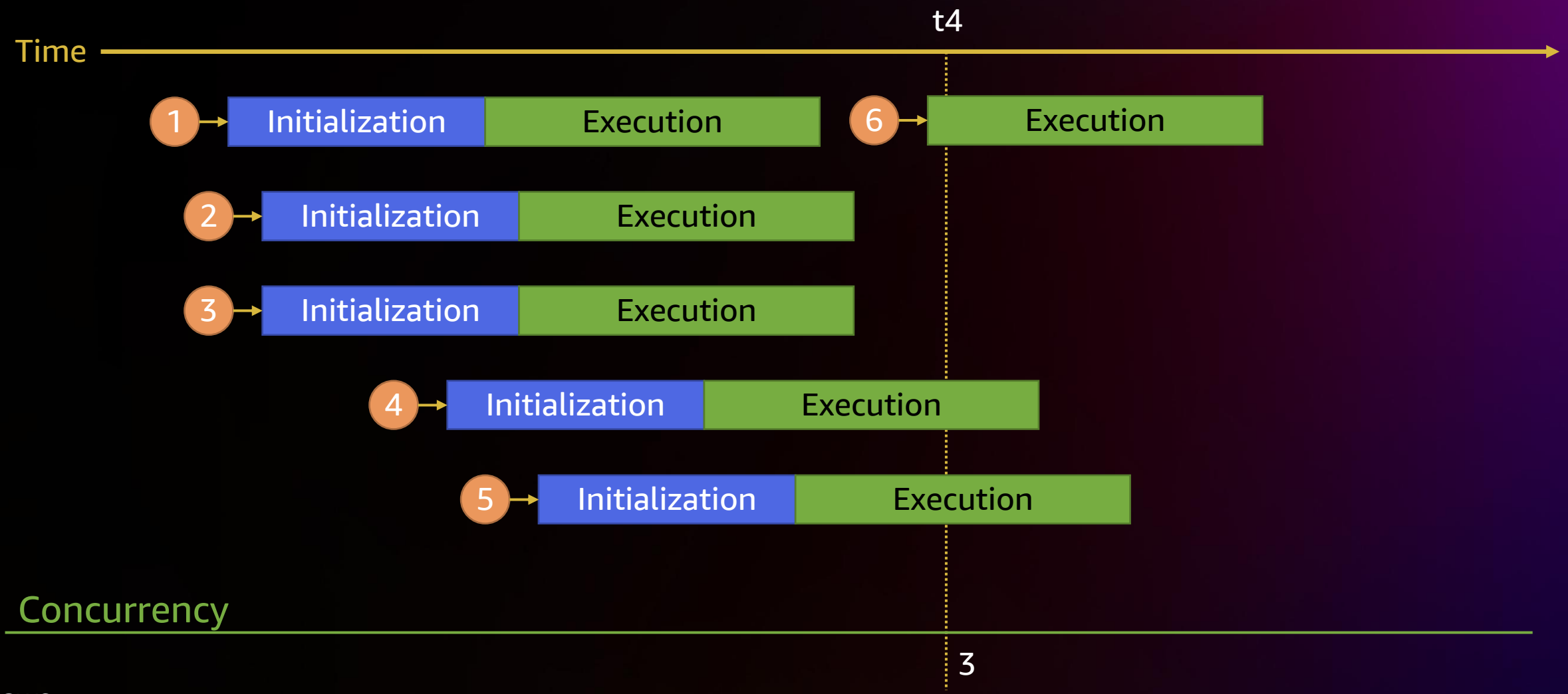
Lambda concurrency



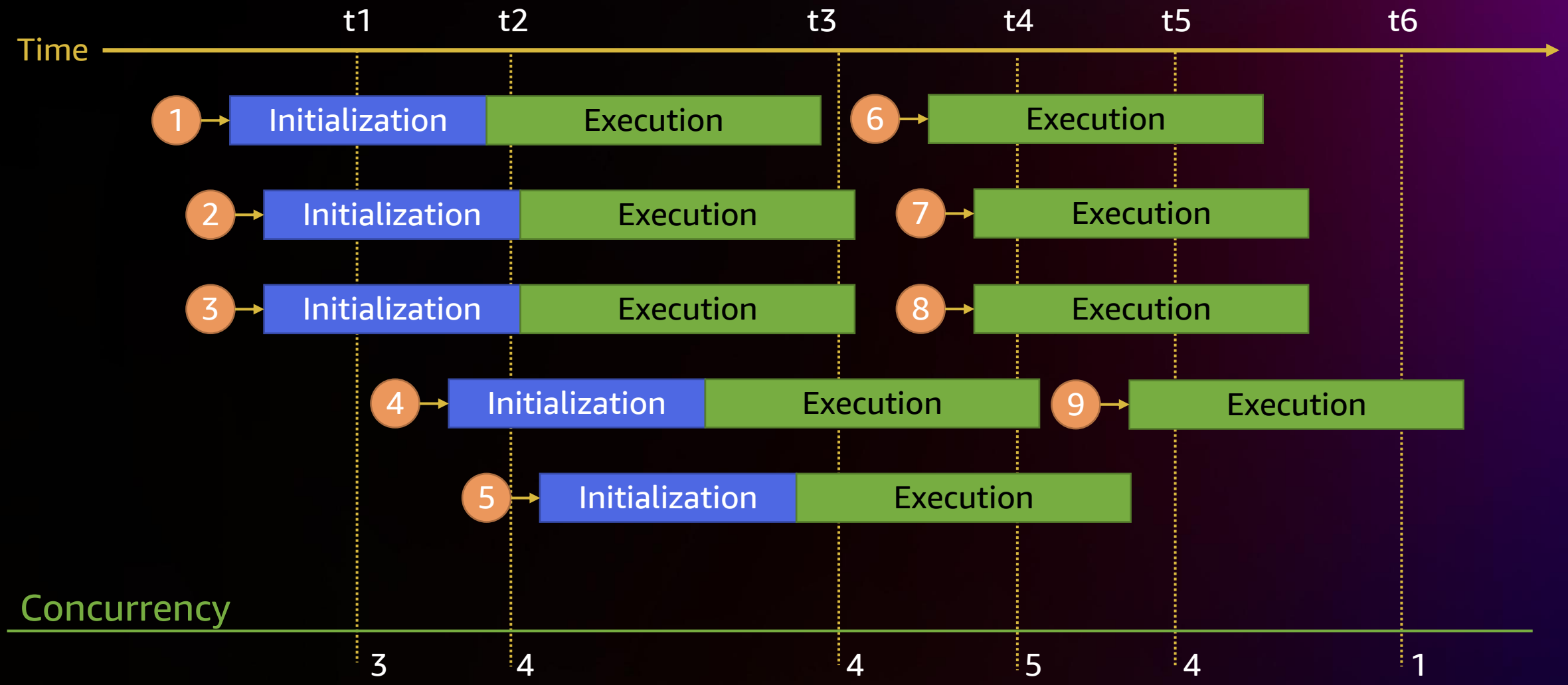
Lambda concurrency



Lambda concurrency



Lambda concurrency



Concurrency is a point-in-time measurement

Concurrency is a
point-in-time measurement

and not the same as
requests per second

How to calculate concurrency requirements

Lambda concurrency

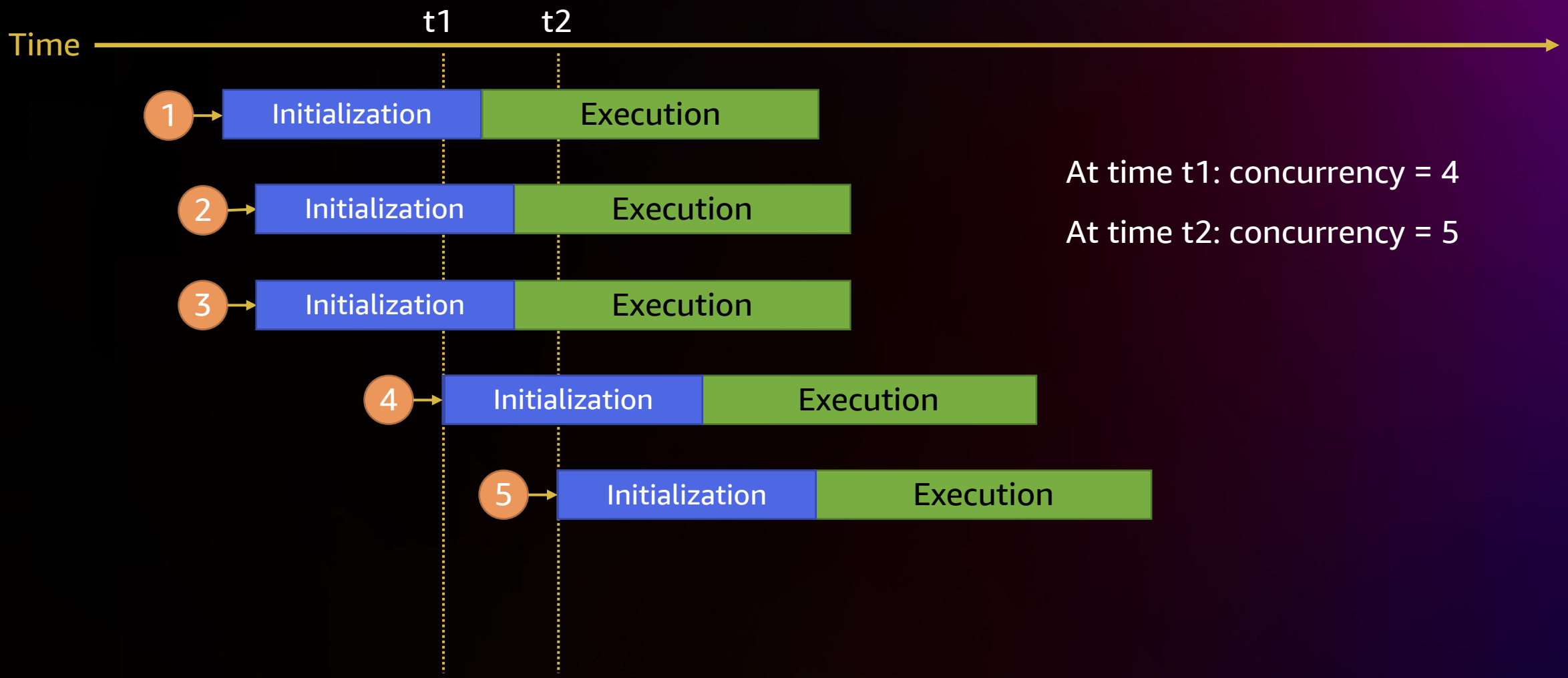
Lambda concurrency = requests per second (RPS) x duration in seconds

Examples

- 10,000 RPS x 1,000 ms = 10,000 concurrency
- 10,000 RPS x 500 ms = 5,000 concurrency
- 10,000 RPS x 100 ms = 1,000 concurrency

Long-running functions require more concurrency

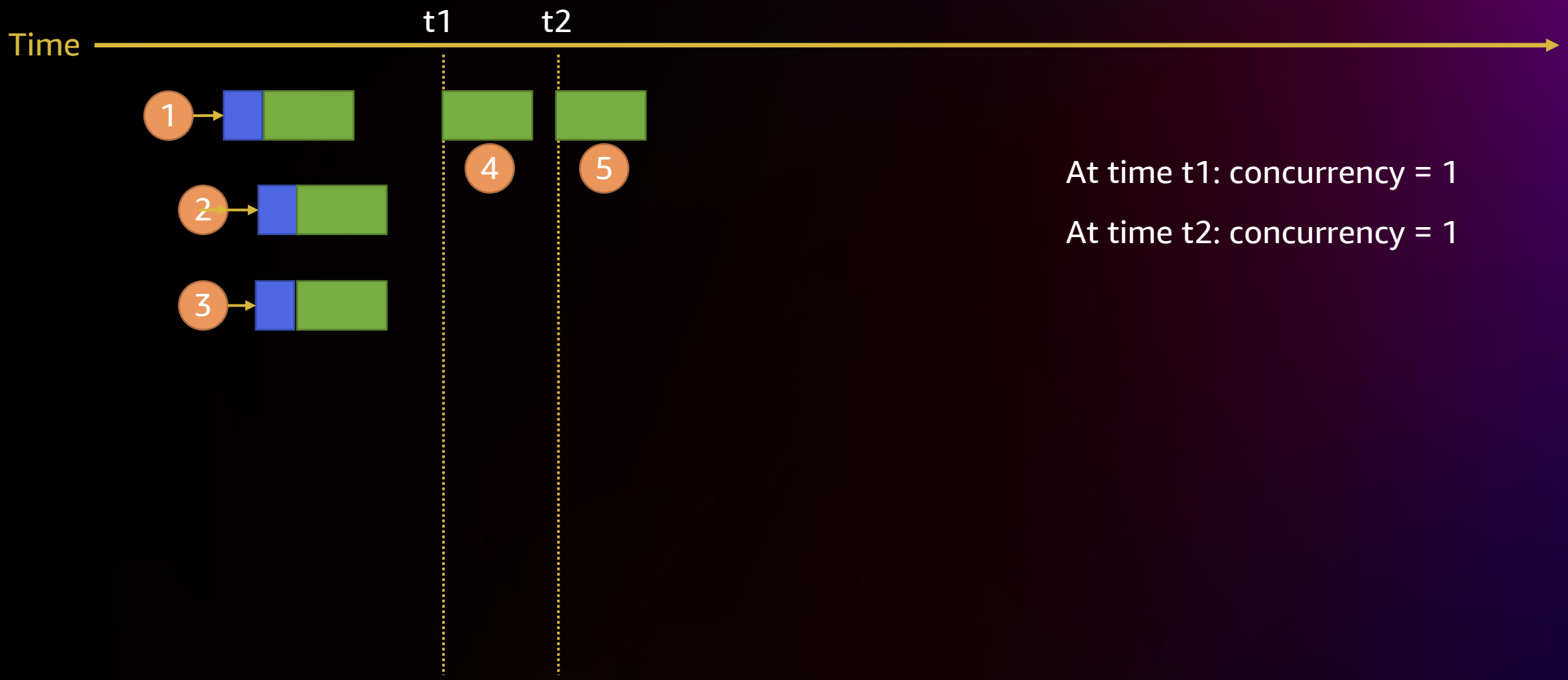
Decreasing concurrency



At time t1: concurrency = 4

At time t2: concurrency = 5

Decreasing concurrency

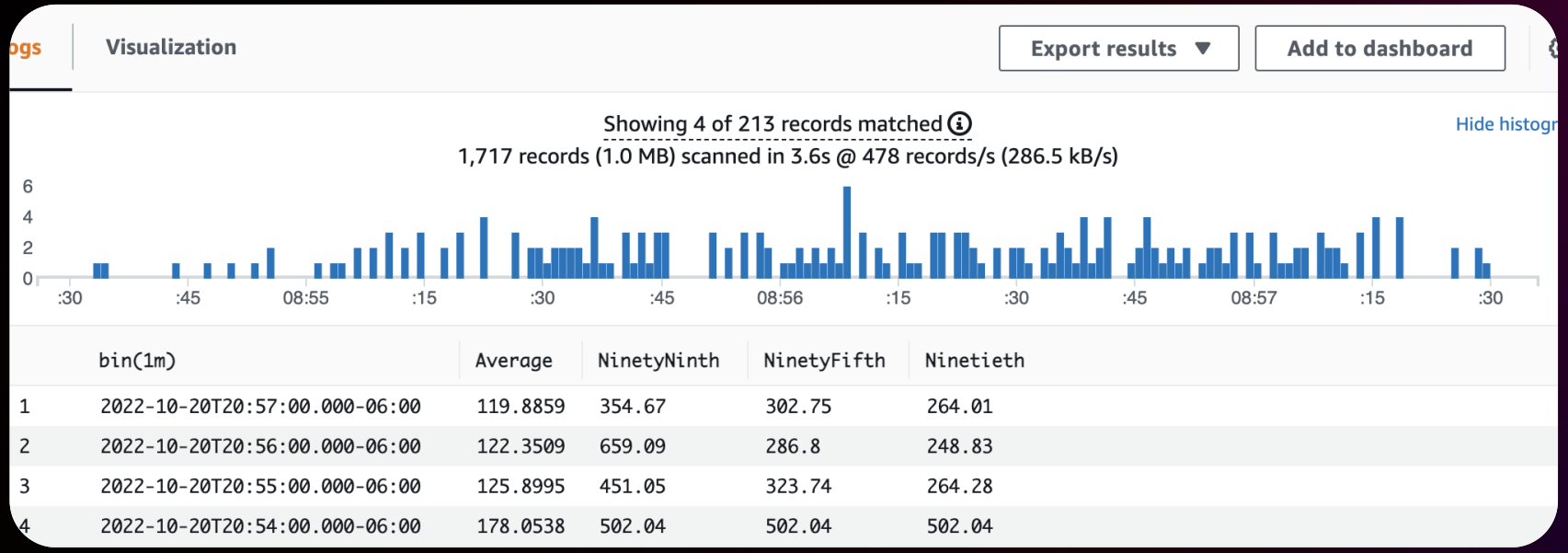


At time t1: concurrency = 1

At time t2: concurrency = 1

Decreasing concurrency: Warm start durations

```
filter @type = "REPORT" and @message not like /(?i)(Init Duration)/
| stats
  avg(@duration) as Average,
  pct(@duration, 99) as NinetyNinth,
  pct(@duration, 95) as NinetyFifth,
  pct(@duration, 90) as Ninetieth
by bin(1m)
```

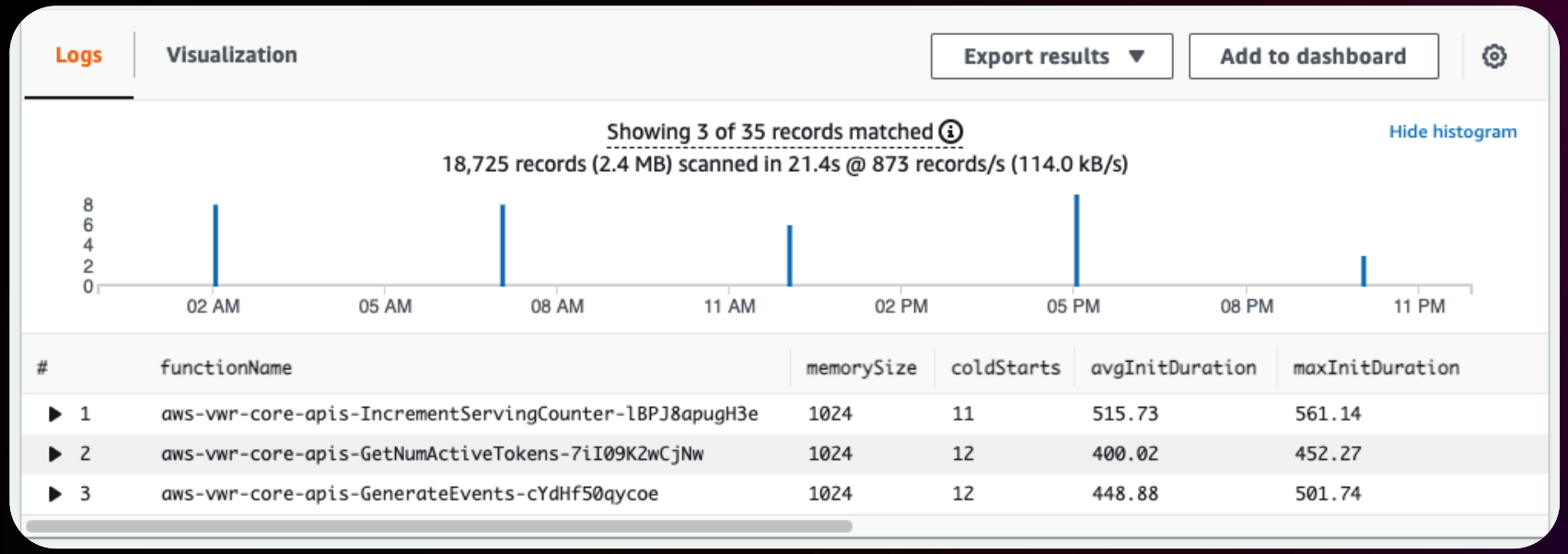


	bin(1m)	Average	NinetyNinth	NinetyFifth	Ninetieth
1	2022-10-20T20:57:00.000-06:00	119.8859	354.67	302.75	264.01
2	2022-10-20T20:56:00.000-06:00	122.3509	659.09	286.8	248.83
3	2022-10-20T20:55:00.000-06:00	125.8995	451.05	323.74	264.28
4	2022-10-20T20:54:00.000-06:00	178.0538	502.04	502.04	502.04

Lambda optimizations: https://s12d.com/Lambda-Optimizations-2021

Decreasing concurrency: Cold start durations

```
filter @type="REPORT" and @message like /(?i)(Init Duration)/
| parse @message /^REPORT.*Init Duration: (?<initDuration>.*) ms.*/
| parse @log /^.*\/aws\/lambda\/(?<functionName>.*)/
| fields @memorySize / 1000000 as memorySize
| stats count() as coldStarts, avg(initDuration) as avgInitDuration, max(initDuration) as maxInitDuration
by functionName, memorySize
```



#	functionName	memorySize	coldStarts	avgInitDuration	maxInitDuration
▶ 1	aws-vwr-core-apis-IncrementServingCounter-lBPJ8apugH3e	1024	11	515.73	561.14
▶ 2	aws-vwr-core-apis-GetNumActiveTokens-7iI09K2wCjNw	1024	12	400.02	452.27
▶ 3	aws-vwr-core-apis-GenerateEvents-cYdHf50qycoe	1024	12	448.88	501.74

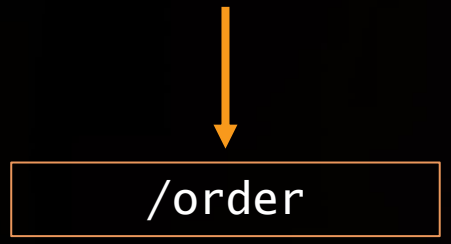
Lambda optimizations: https://s12d.com/Lambda-Optimizations-2021

Applied concurrency: Lambda invocation models

Lambda invocation methods

Synchronous

Amazon API Gateway

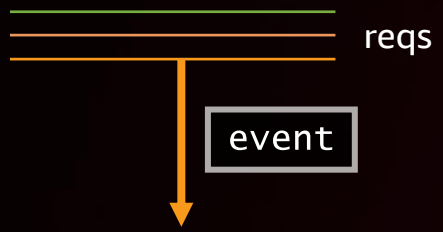


Lambda function

Asynchronous

Amazon SNS

Amazon S3



Lambda function

Poll based

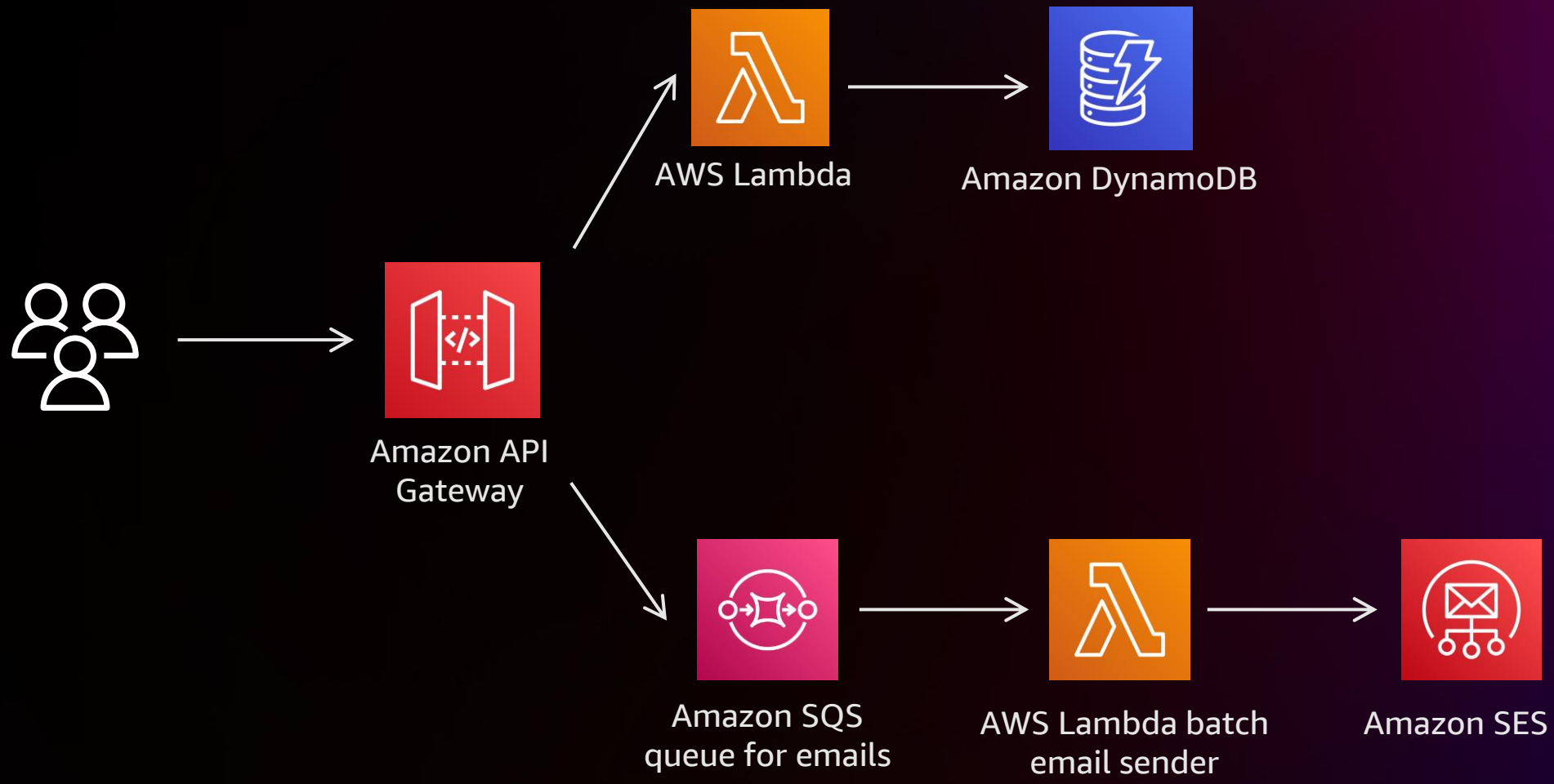
Amazon SQS

Amazon Kinesis

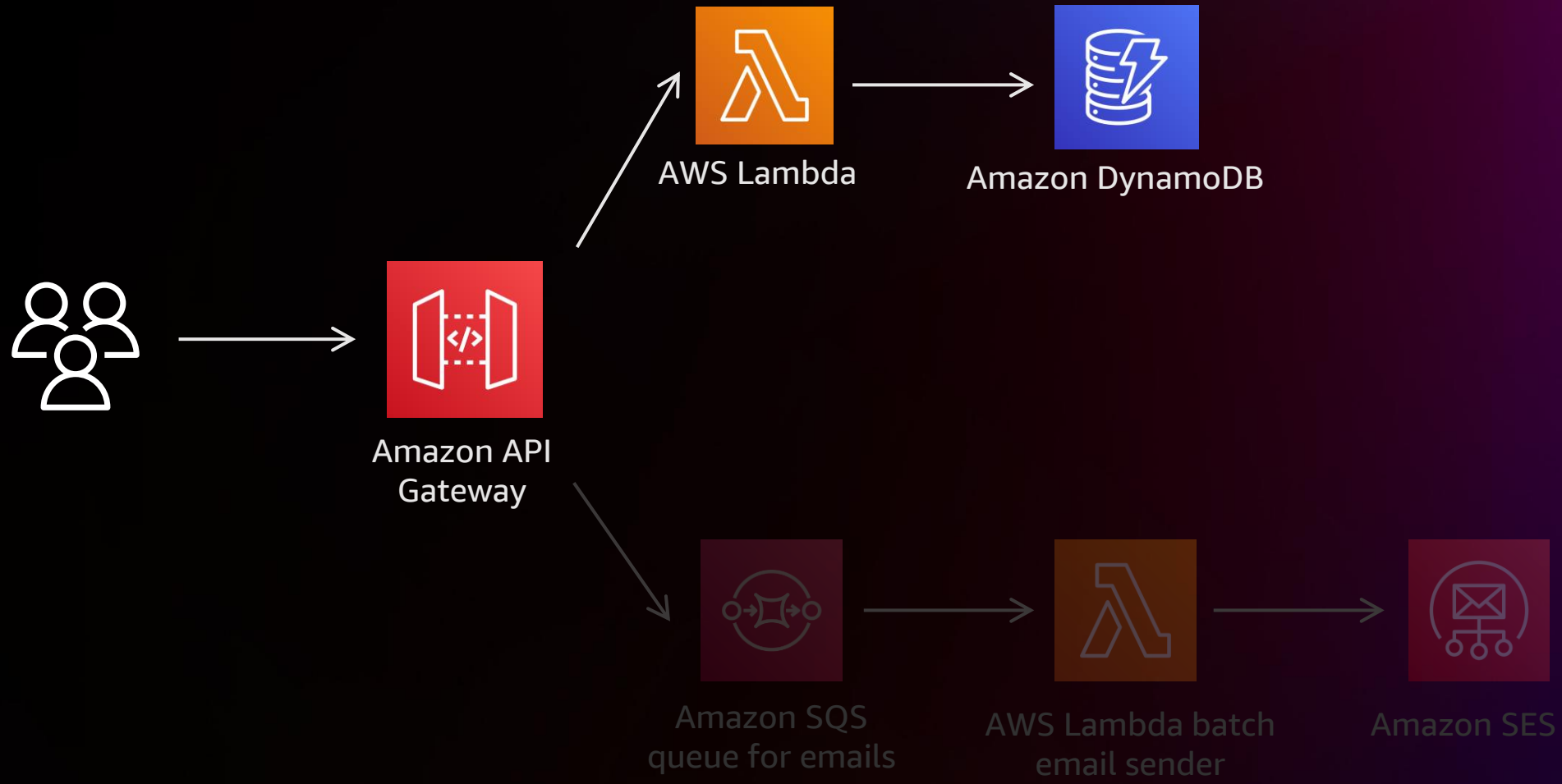
AWS Lambda service

Lambda function

Today’s application



Synchronous invocations



AWS Lambda function scaling quotas

Account concurrency quota

Maximum concurrency in a given Region
across all functions in an account

1,000 per Region by default

This can be increased to customer needs

AWS Lambda function scaling quotas

Burst concurrency quota

Maximum increase in concurrency for an initial burst of traffic

3,000 in Oregon, N. Virginia, and Ireland

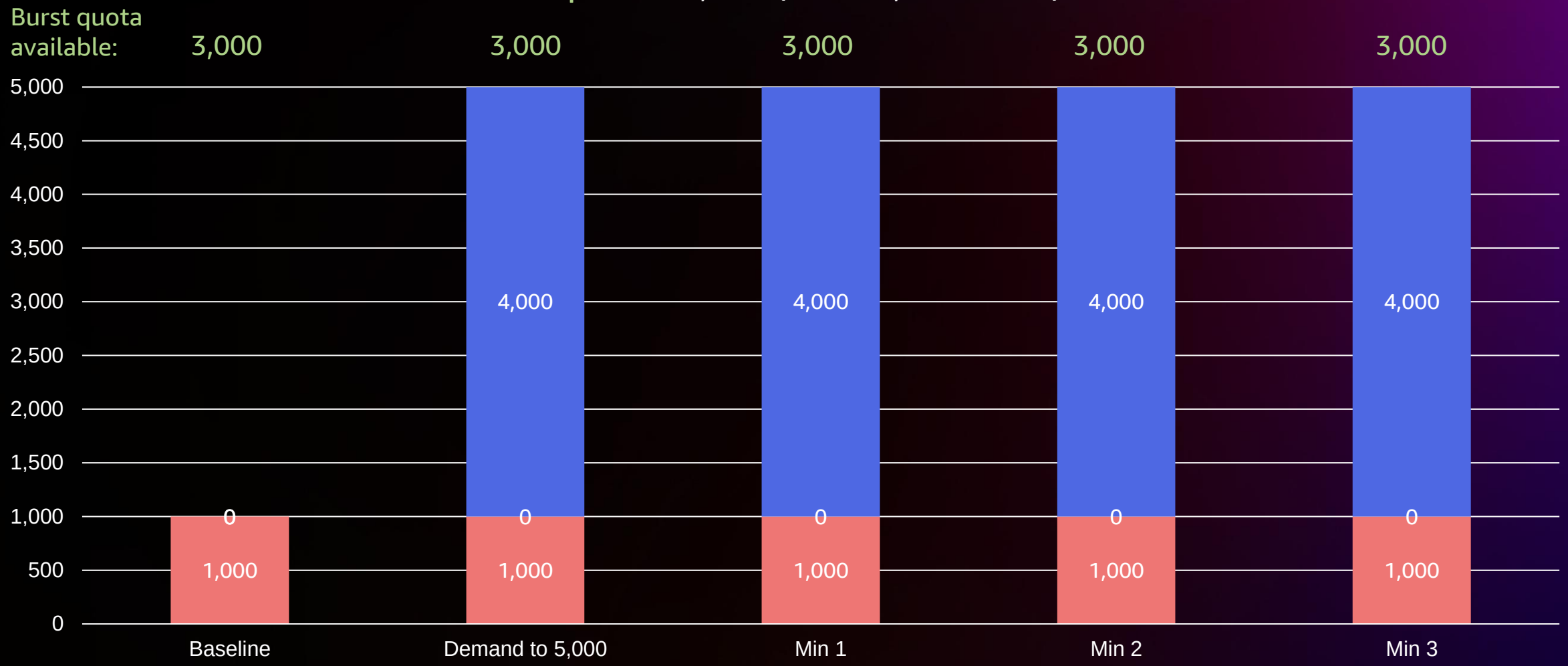
1,000 in Tokyo, Frankfurt, and Ohio

500 in all other Regions

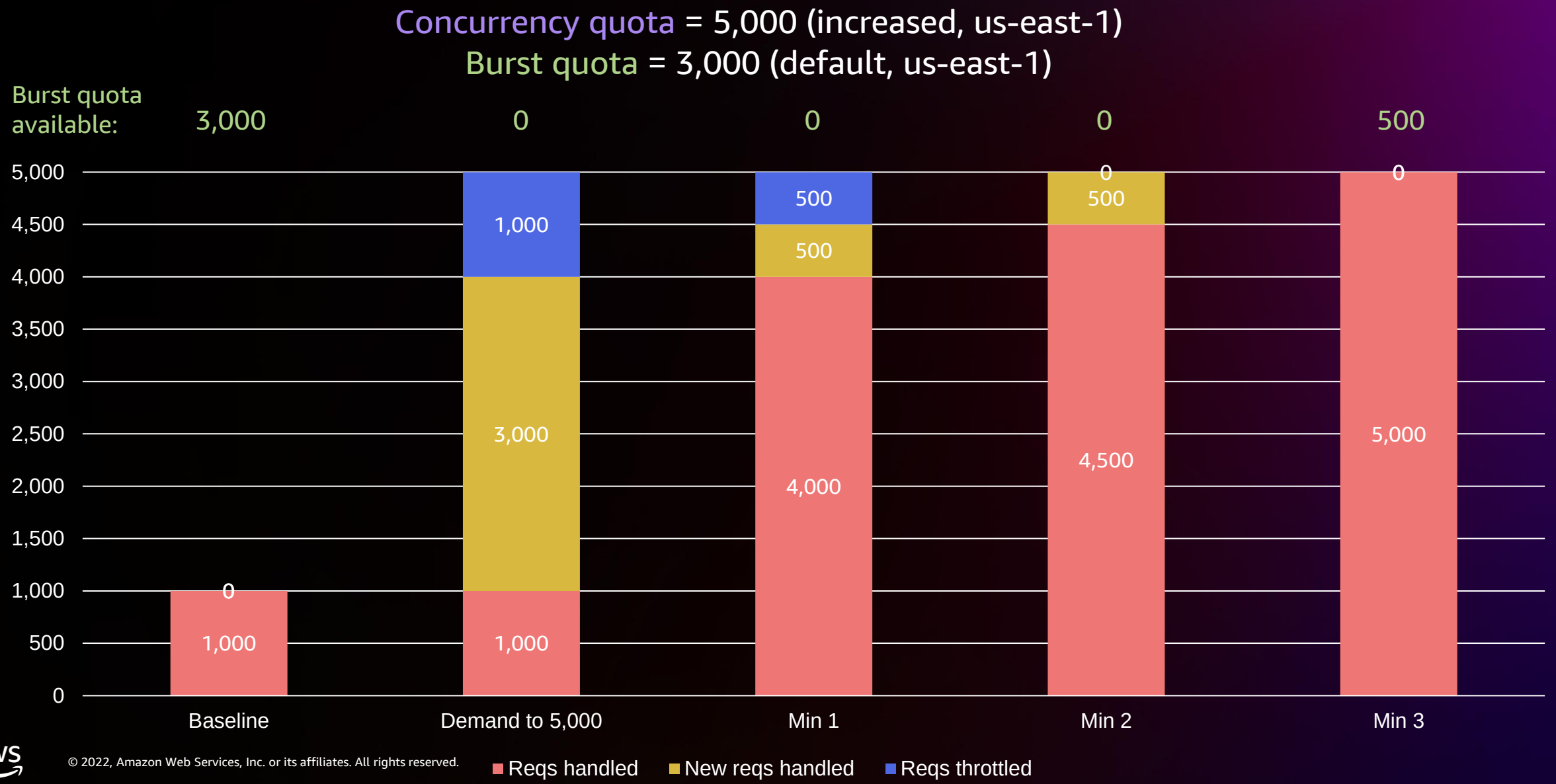
After the initial burst, your function concurrency can scale by an additional **500 instances each minute**

Lambda bursting and ramp up

Concurrency quota = 1,000 (default, us-east-1)
Burst quota = 3,000 (default, us-east-1)



Lambda bursting and ramp up

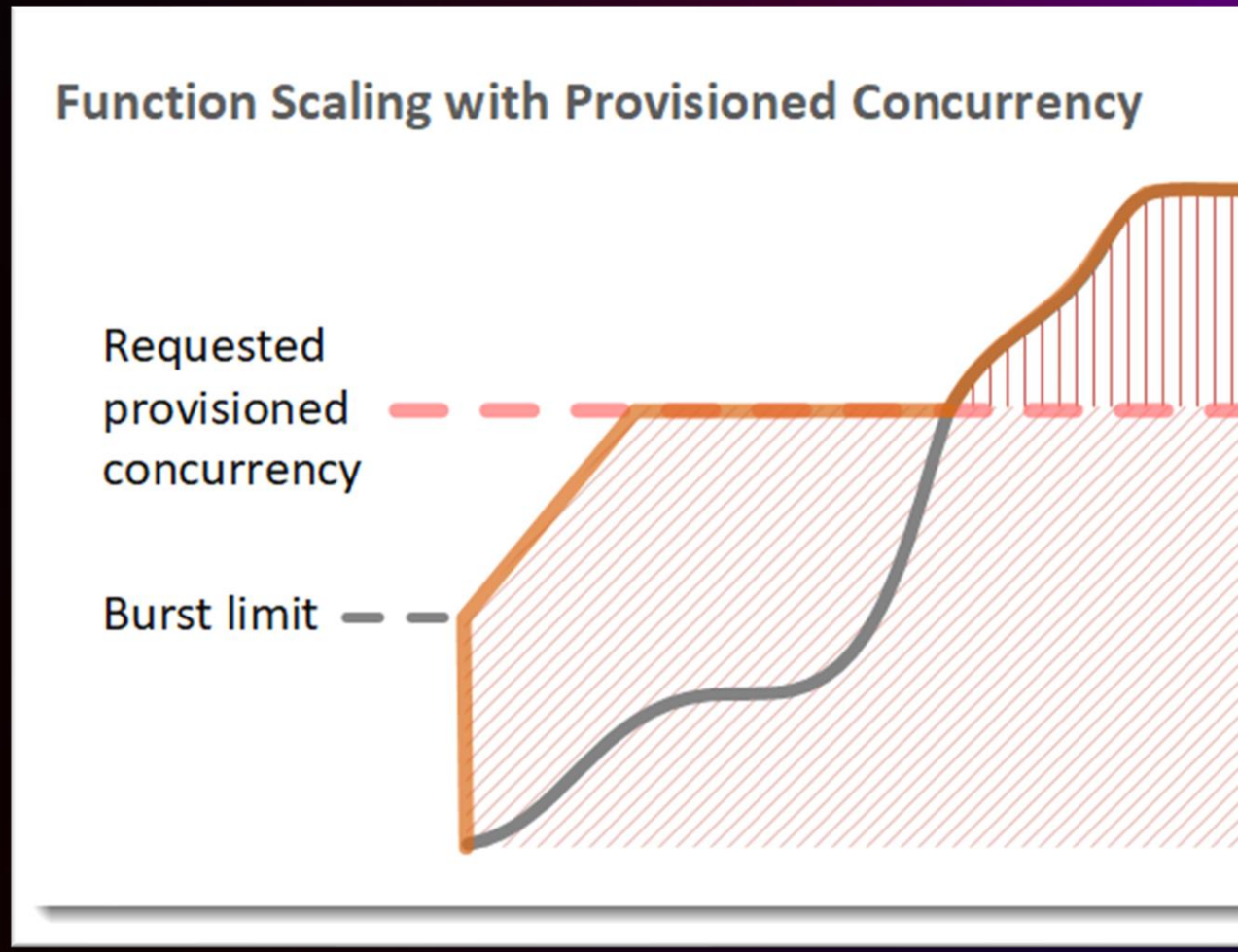


What if you need more burst than the burst quota allows?

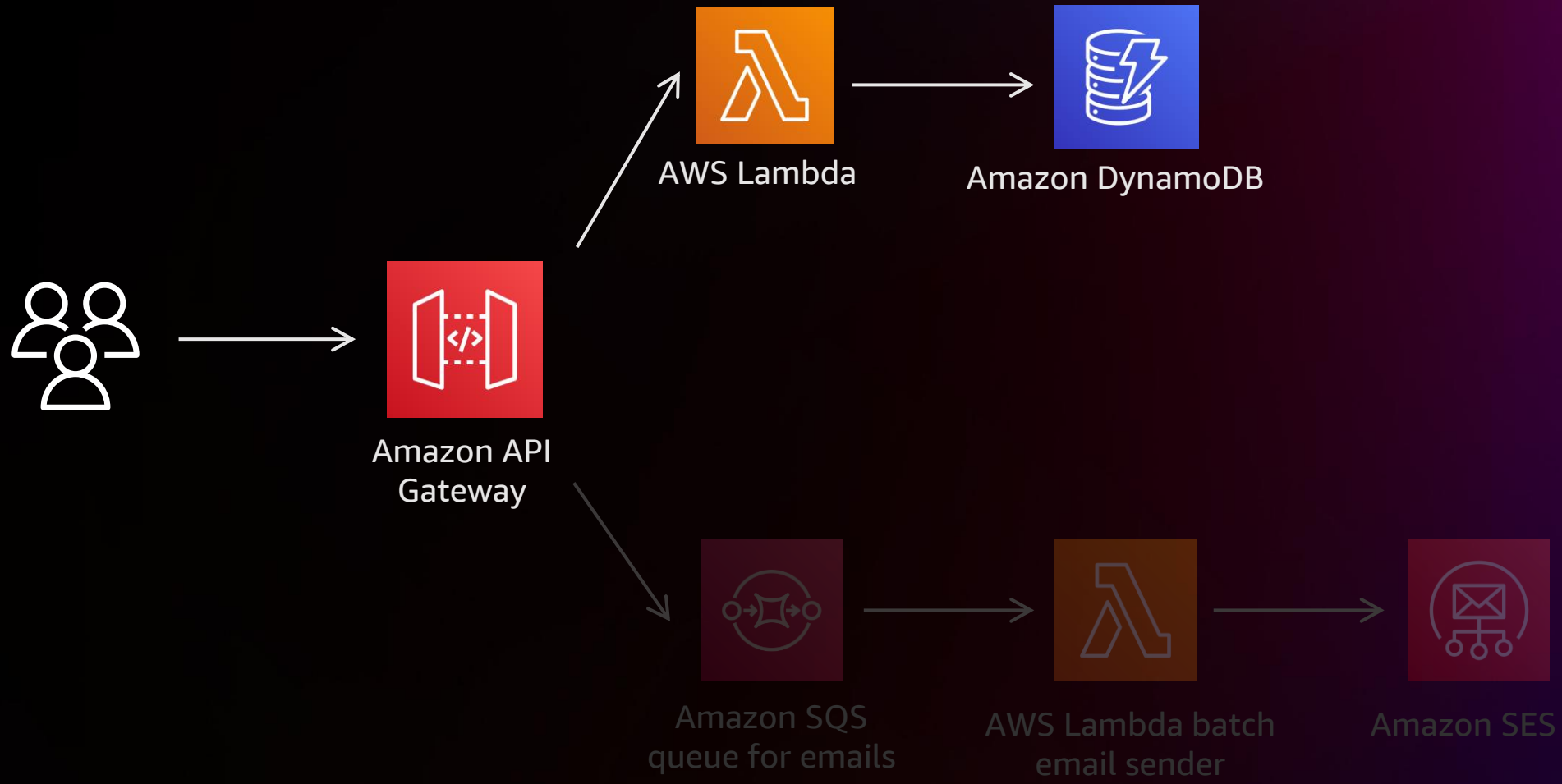
Lambda concurrency controls

Provisioned concurrency

- Sets floor on minimum number of execution environments
- Pre-warm execution environments to reduce cold-start impact
- Burst to use standard concurrency if desired
- Can save costs in certain situations



Synchronous invocations

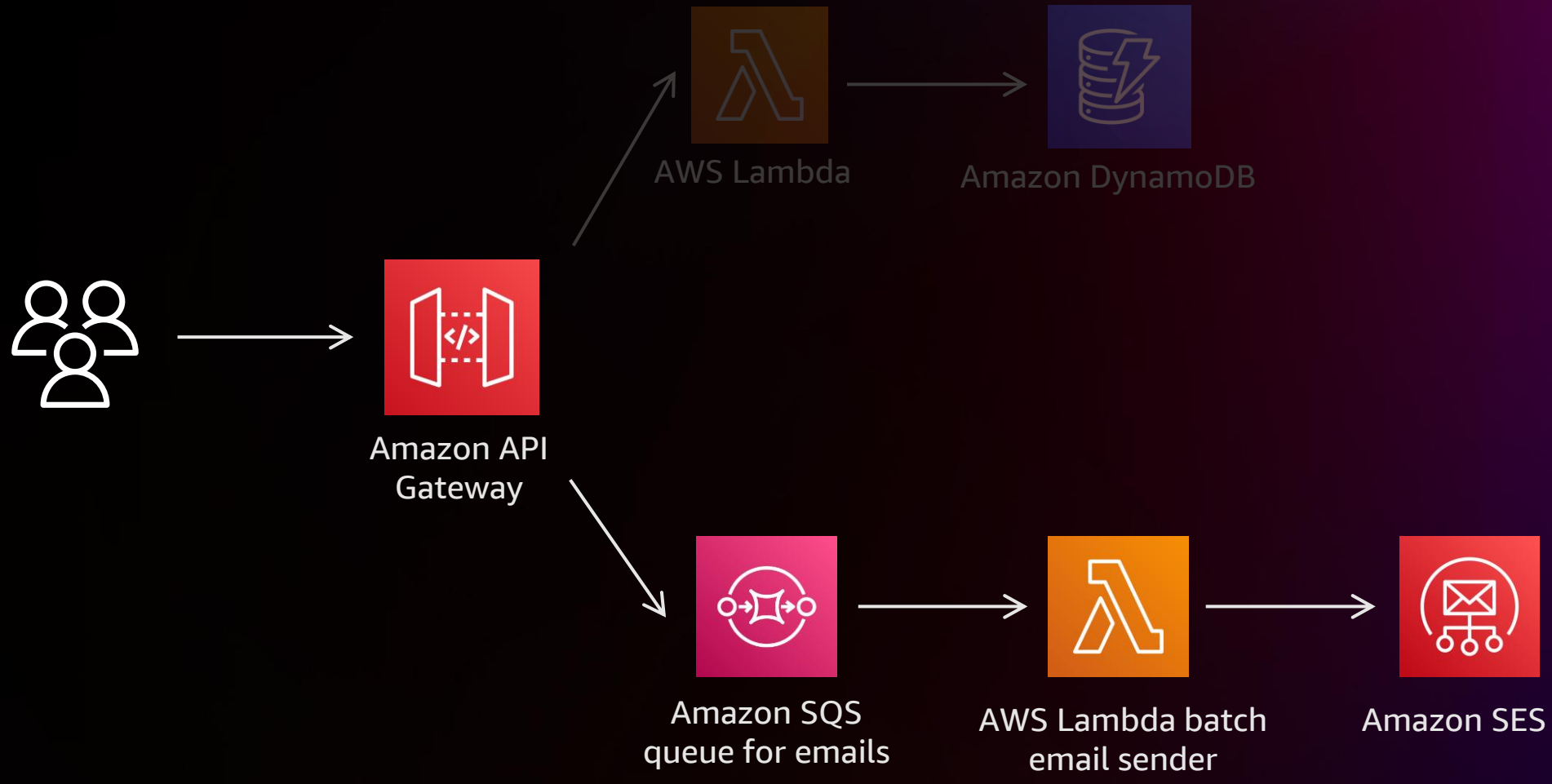


Synchronous invocations: Best practices

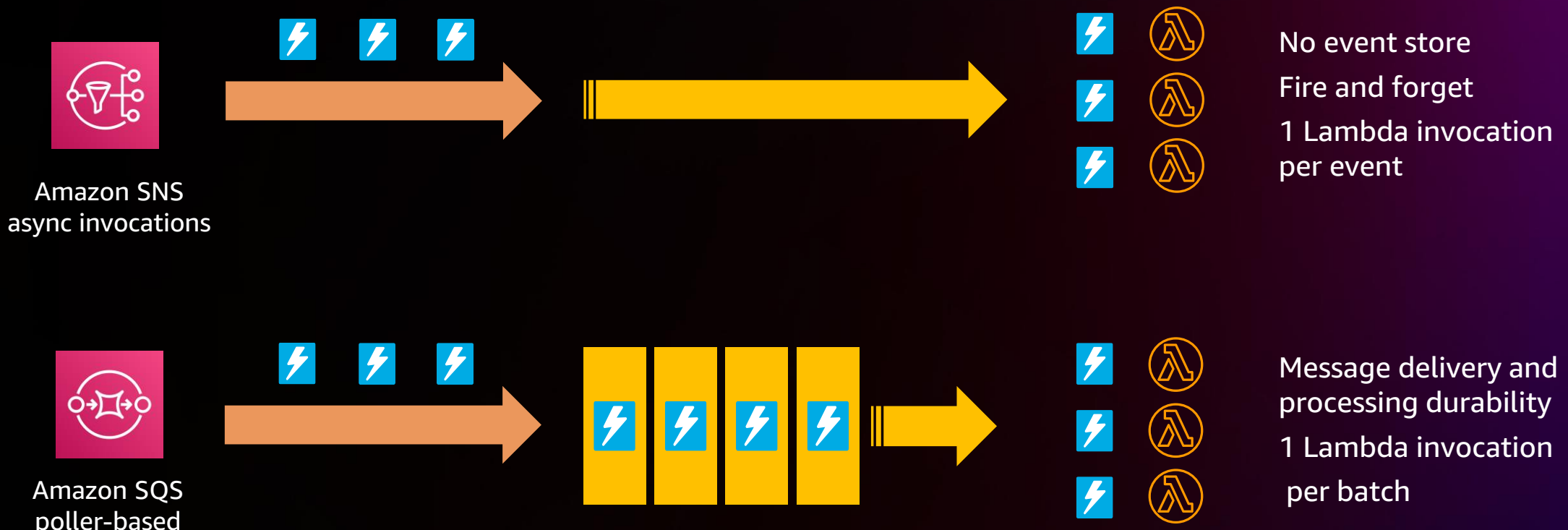
- **Load test**
  - Raise account-level concurrency quota in advance of bursts of demand
- **Set appropriate timeouts**
  - Set reasonable timeouts per Lambda function – do not set to 15 minutes
  - Configure API response timeout – default is 29 seconds
- **Retry with backoff**
  - Clients should retry, but with exponential backoff to not stampede the system
- **Implement idempotency**
  - Expect and account for retried operations
  - Application awareness and handling of scenarios is key

Dive deeper at https://s12d.com/reInvent2020-Error-Handling

Asynchronous invocations



Lambda function concurrency across models

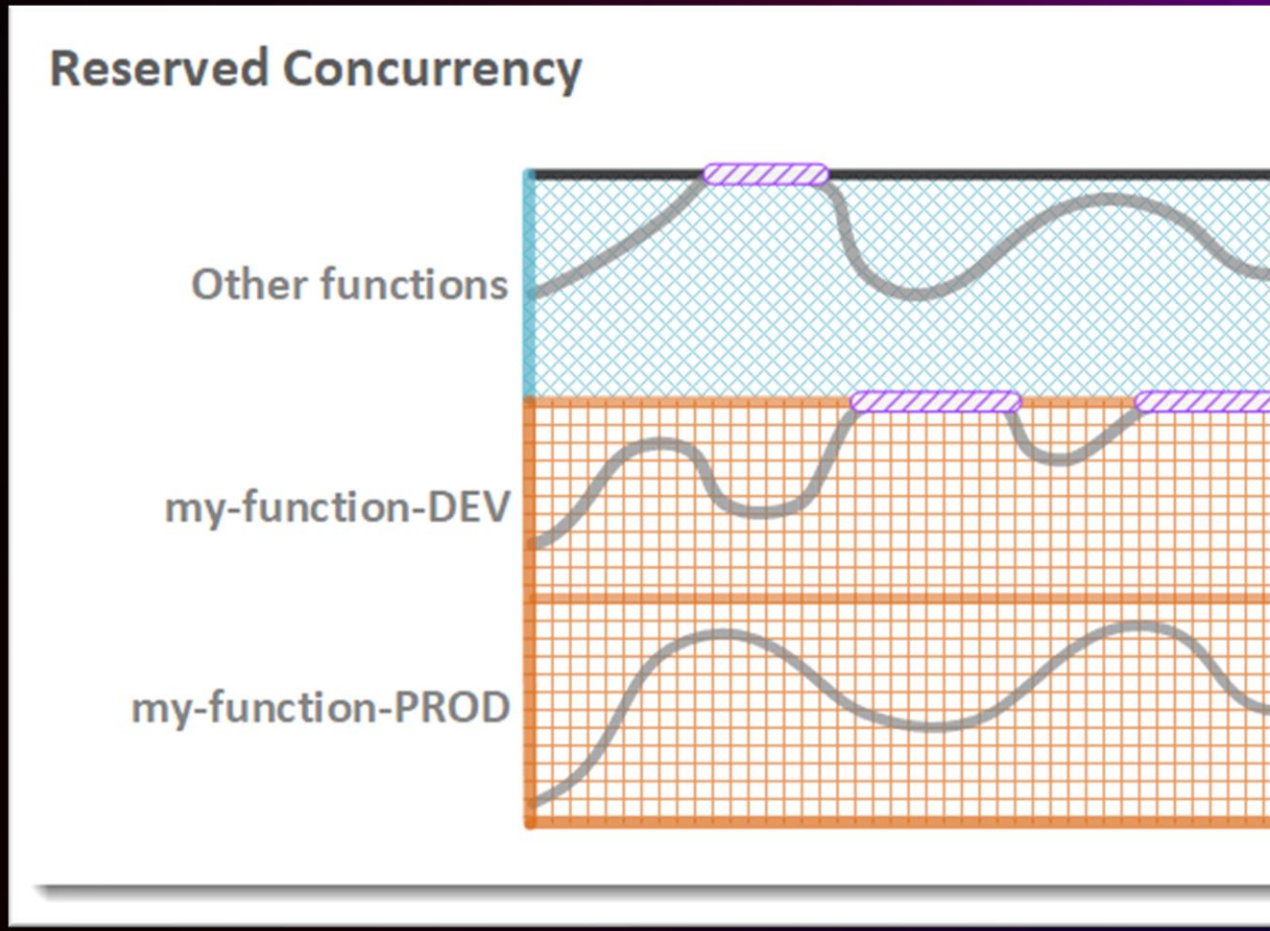


What if scaling too fast may overwhelm downstream systems?

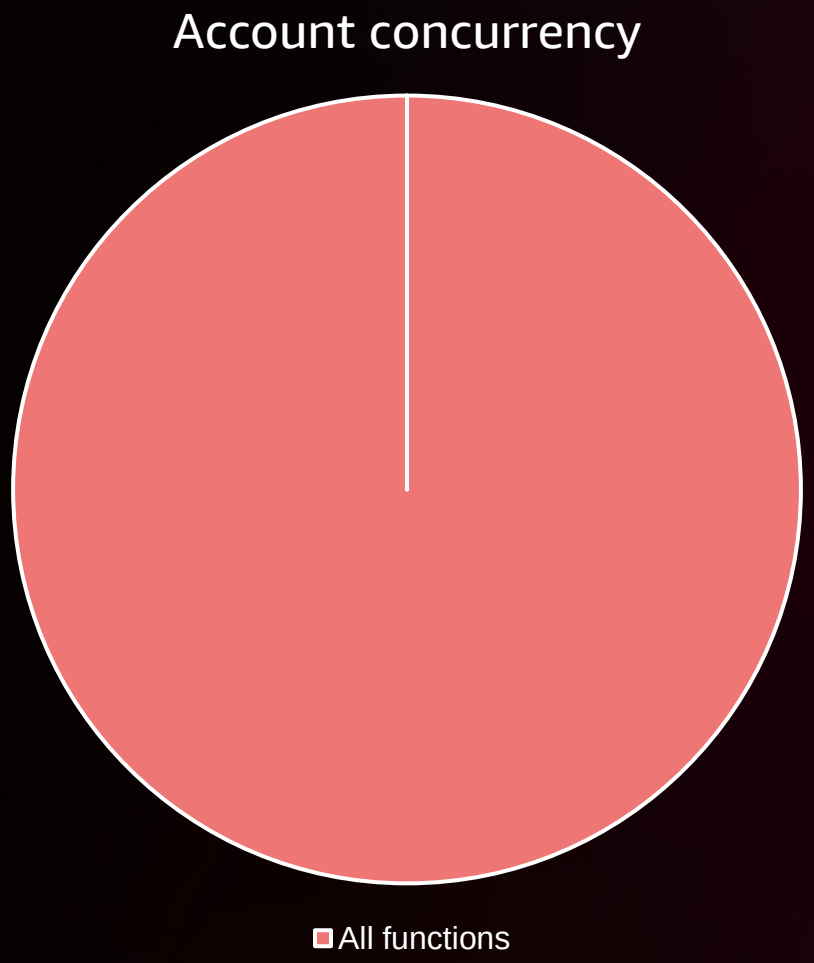
Lambda concurrency controls

Reserved concurrency

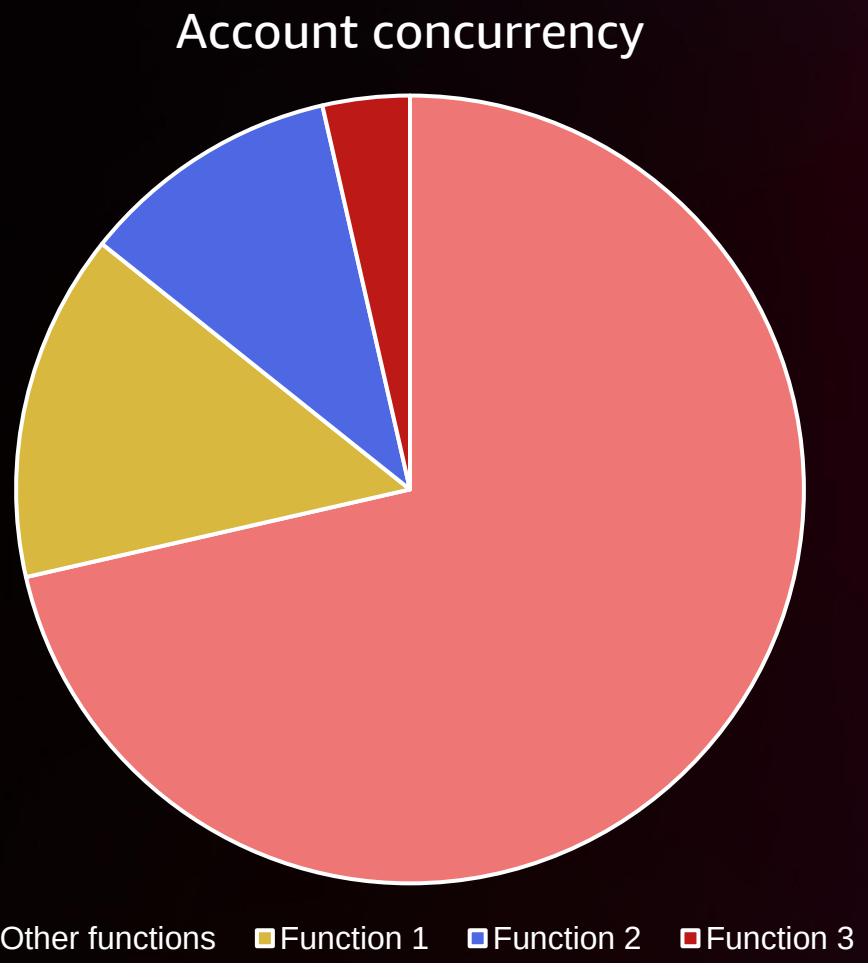
- Sets ceiling on maximum number of execution environments – upper limit on maximum concurrency for a given function
- Also reserves that concurrency from the account's quota



Reserved concurrency



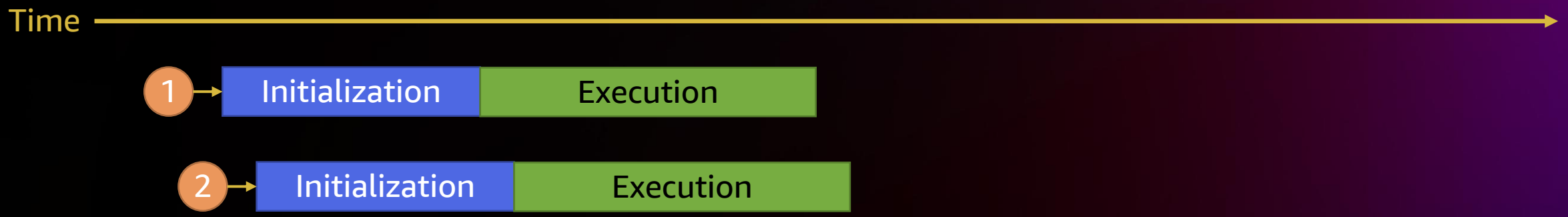
Reserved concurrency



Reserved concurrency is an upper limit for a Lambda function's concurrency

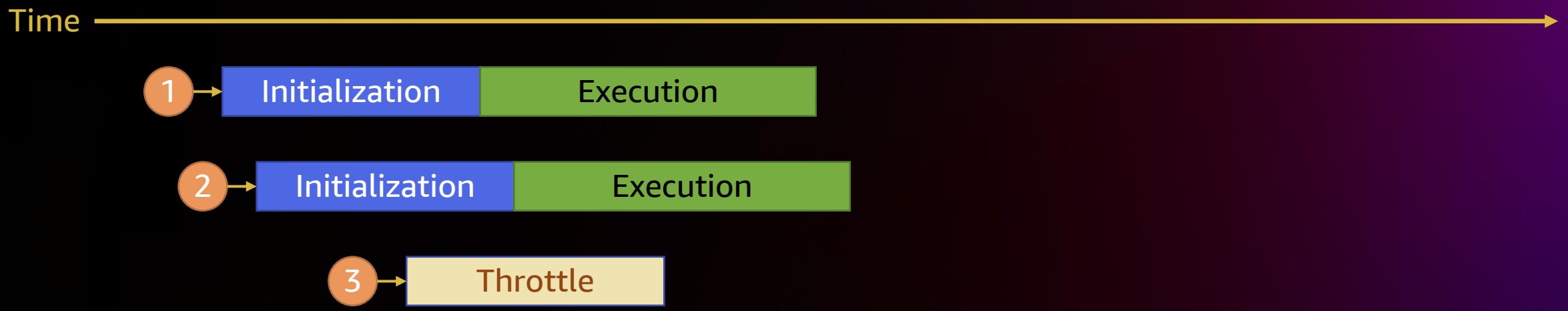
Reserved concurrency is always respected, regardless of integration

How Lambda scales: Reserved concurrency



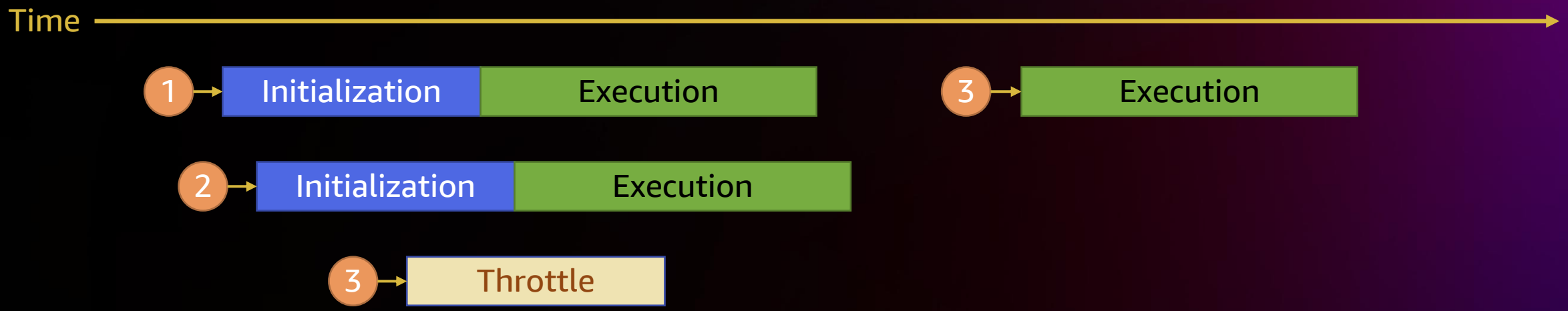
Reserved Concurrency = 2

How Lambda scales: Reserved concurrency



Reserved Concurrency = 2

How Lambda scales: Reserved concurrency



Reserved Concurrency = 2

Asynchronous invocations: Best practices

- Invocation model
  - Leverage async invocations whenever possible
- Function configuration
  - Leverage batching for cost reduction and performance gains
  - Catch errors/failures and return altogether if using a batch size greater than 1
  - Configure Lambda on-failure destination to save and review failed messages
  - Monitor function error metrics to ensure they remain at or close to 0
- Amazon SQS queue configuration
  - Set the queue visibility timeout to at least 6x function timeout
  - Setup a DLQ with maxReceiveCount of at least 5

Amazon SQS and Lambda best practices: https://s12d.com/SQS-Lambda-Best-Practices

Concurrency tips

- Understand fundamentals
- Plan ahead and load test
- Monitor, monitor, monitor
  - `ConcurrentExecutions`
  - `Errors`
  - `Throttles`
  - SQS queue depth, Kinesis iterator age, etc.

Open discussion

Check out these other sessions

SVS404: A closer look at AWS Lambda

SVS401: Best practices for advanced serverless developers

SVS314: AWS Lambda performance tuning: Best practices and guidance

References

Documentation

- Invoke Lambda functions
  https://s12d.com/lambda-invocation
- Using Lambda with Amazon SQS
  https://s12d.com/lambda-sqs

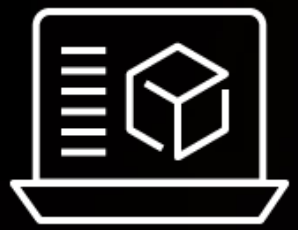
Tools and guidance

- AWS Lambda Powertools for Python
  https://s12d.com/powertools
- Serverless Applications Lens – AWS Well-Architected Framework
  https://s12d.com/serverless-wa-lens




Continue your AWS serverless learning

Learn at your own pace



Expand your serverless skills with our learning plan on **AWS Skill Builder**

Increase your knowledge

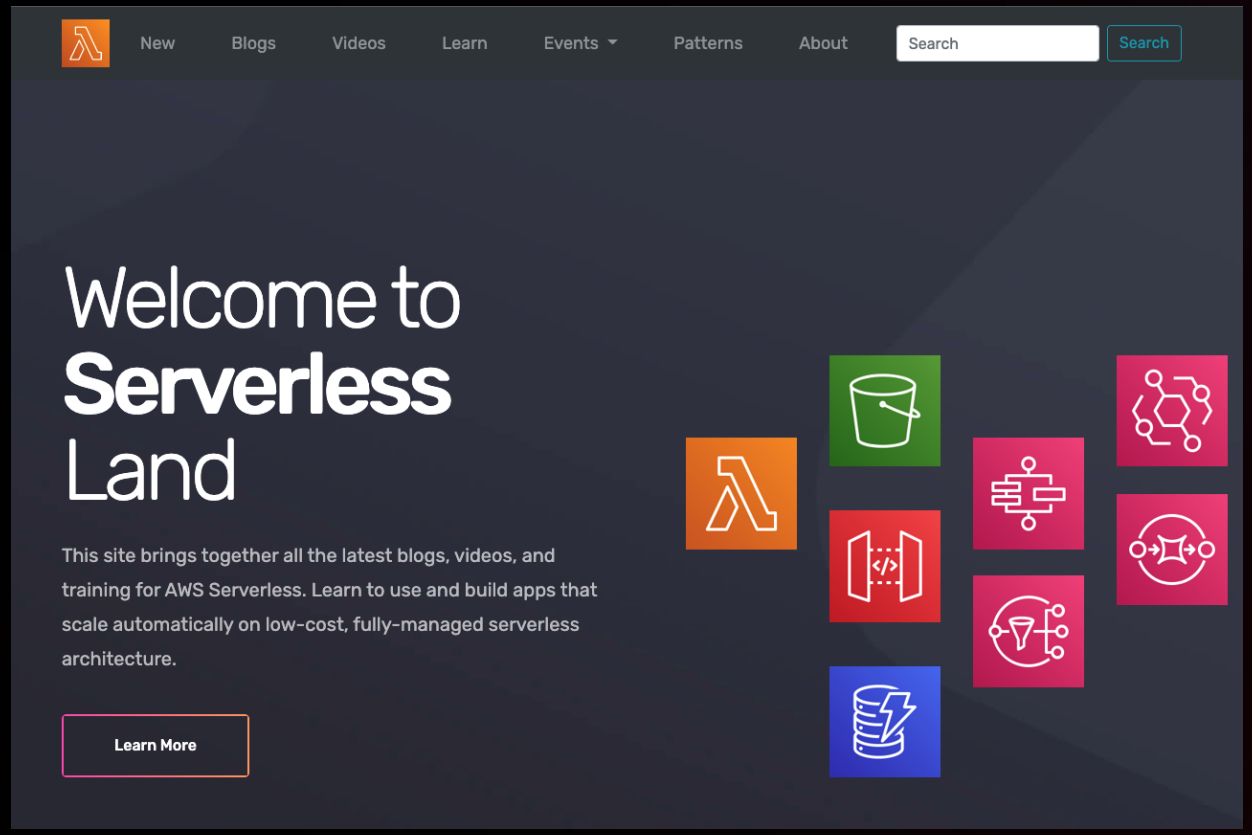
Use our **Ramp-Up Guides** to build your serverless knowledge

Earn AWS serverless badge

Demonstrate your Knowledge by achieving **digital badges**

https://s12d.com/serverless-learning

Serverless Land for more resources



Session slides and links: https://s12d.com/reInvent2022-Lambda-Concurrency

Thank you!

Brian Zambrano (he/him)
@brianzambrano

Justin Pirtle (he/him)
@justinpirtle

Please complete the session survey in the **mobile app**