
AWS
re:Invent

SVS405-R

A Serverless Journey: Under the Hood of AWS Lambda

Holly Mesrobian
Director of Engineering
Amazon AWS Lambda
Amazon Web Services

Marc Brooker
Senior Principal Engineer
Amazon AWS Serverless
Amazon Web Services

AWS re:Invent

aws

SERVERLESS AT SCALE IS THE NEW NORM

THOMSON REUTERS

processes **4,000 requests** per second

mlbam 

ingests, analyzes and stores **17+ petabytes of data** per season

processes **half a trillion** validations of stock trades daily

API traffic to register and license more than **47 million driver records** in Great Britain,

executes **16 million** requests a month

processes **tens of billions** of data points monthly

Running Highly Available Large Scale Systems Is a Lot of Work

AWS re:Invent

aws

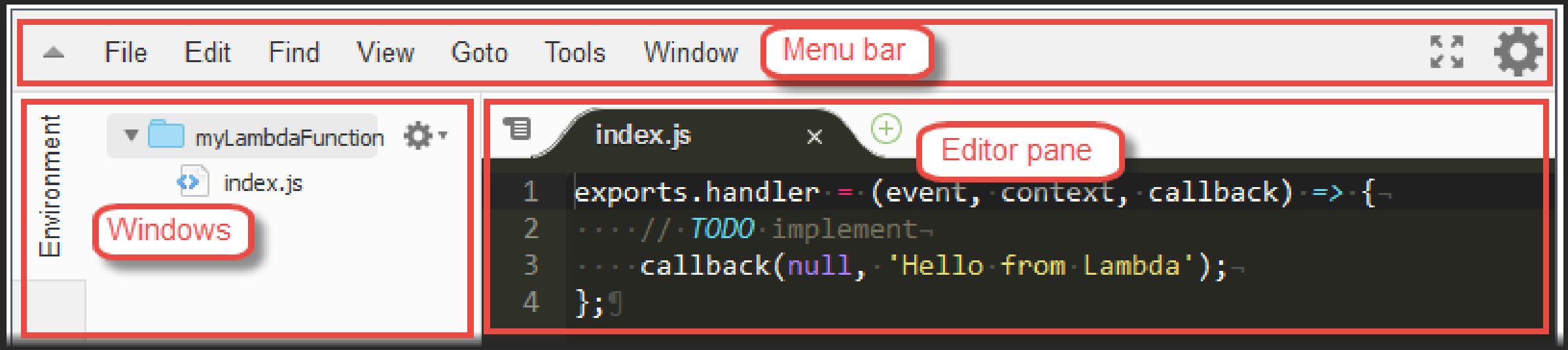
Load Balancing

Scaling Up and Down

Handling Failures

Predictable Low Latency


STOP
UNDER CONSTRU TION


File
Edit
Find
View
Goto
Tools
Window
Menu bar
Environment
myLambdaFunction
index.js
Windows
index.js
Editor pane
1 exports.handler = (event, context, callback) => {
2 ····//·TODO·implement
3 ····callback(null, 'Hello from Lambda');
4 };

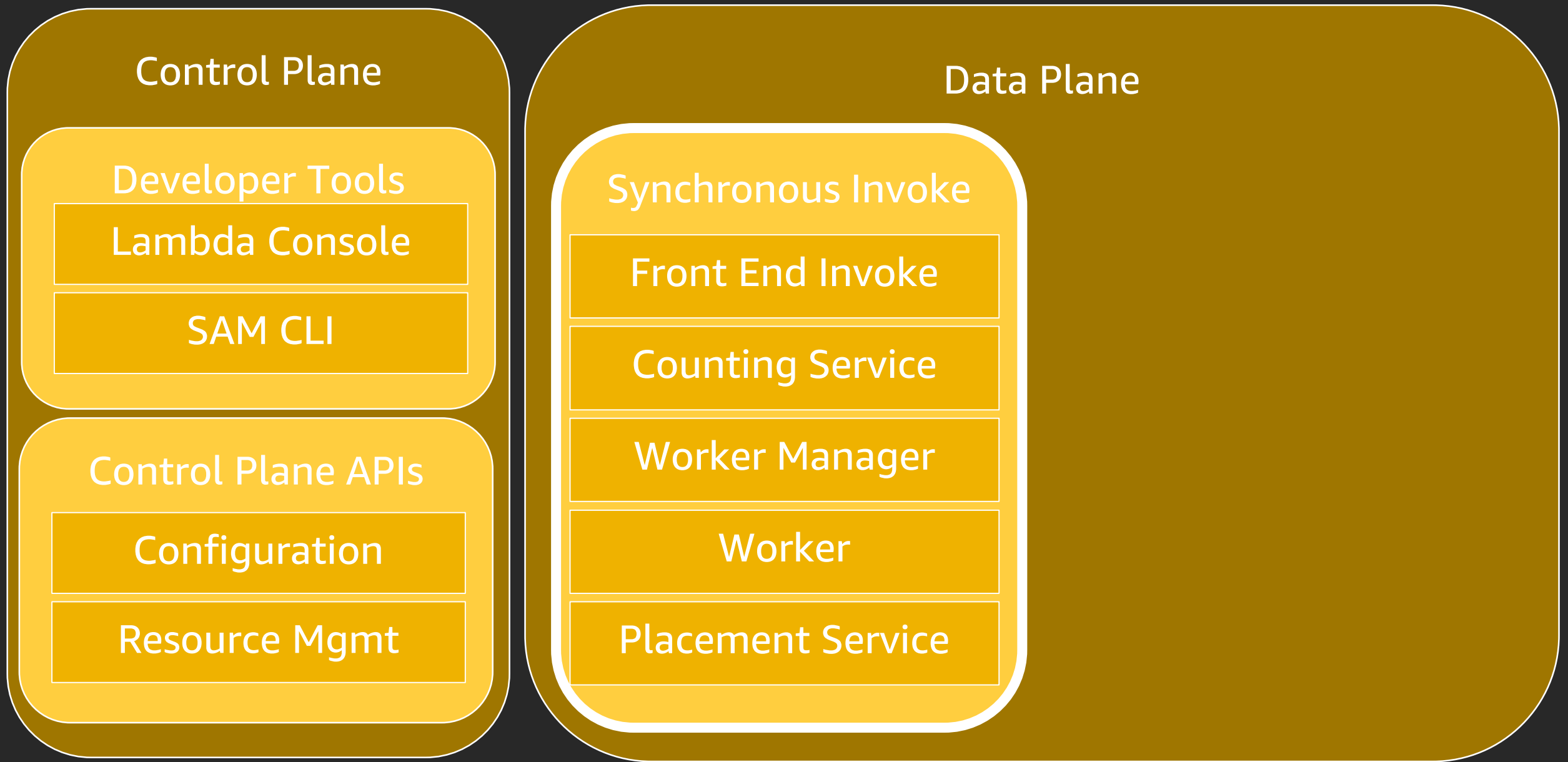
AWS Lambda Handles

Event Processing

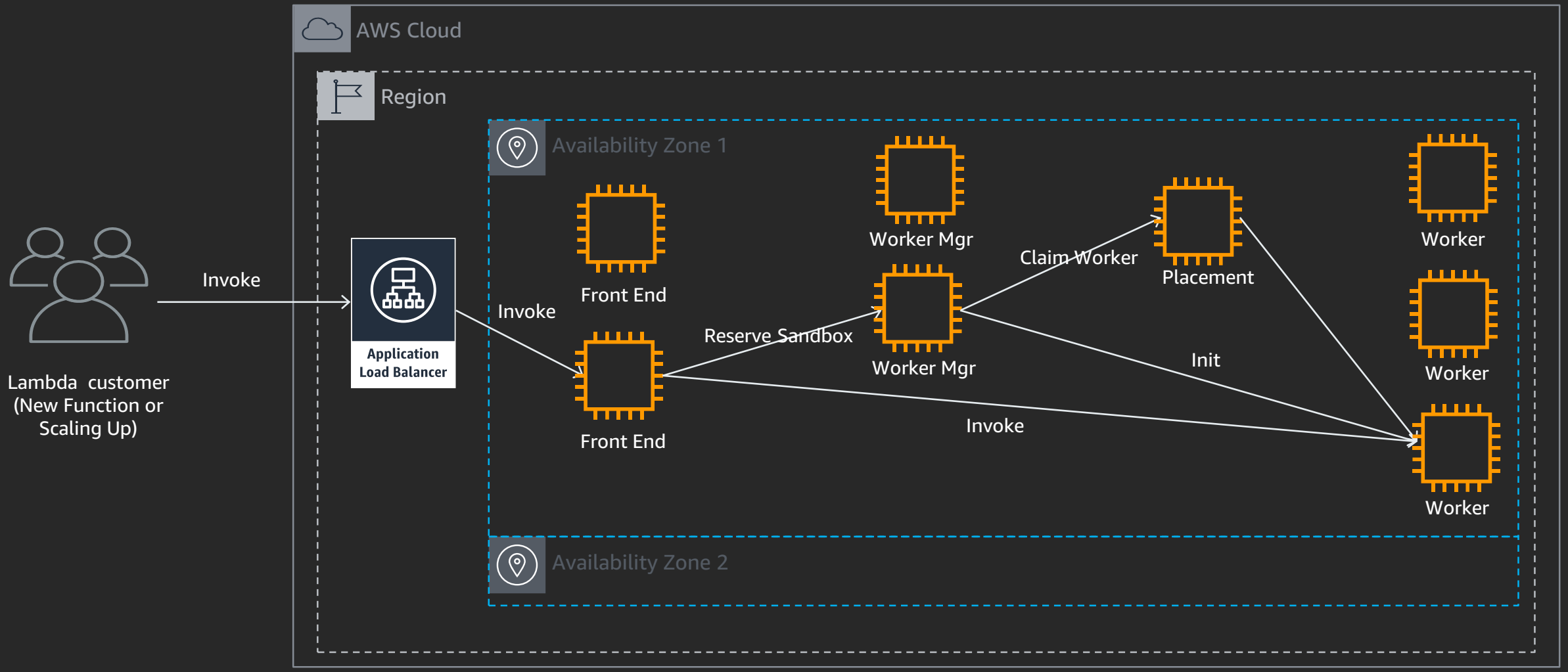
Stream Processing

Predictable Performance

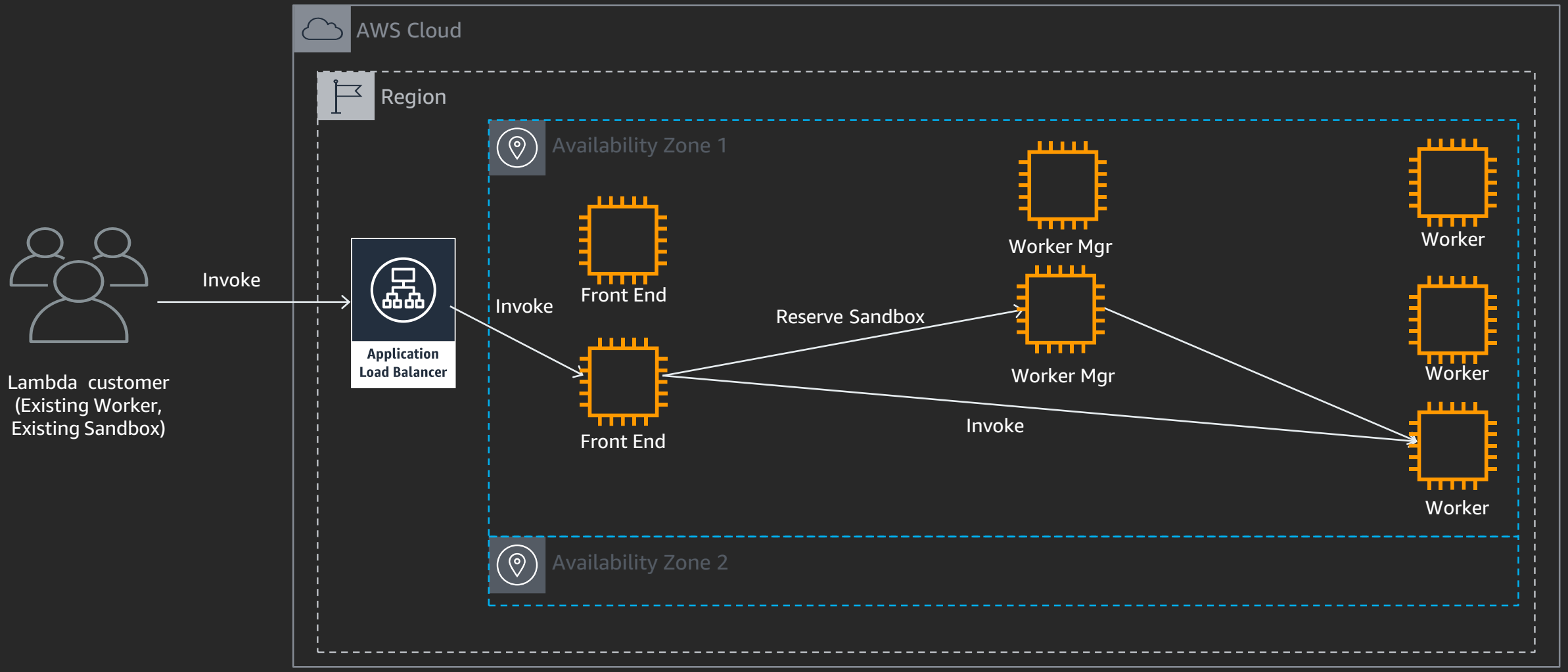
Innovations in Isolation

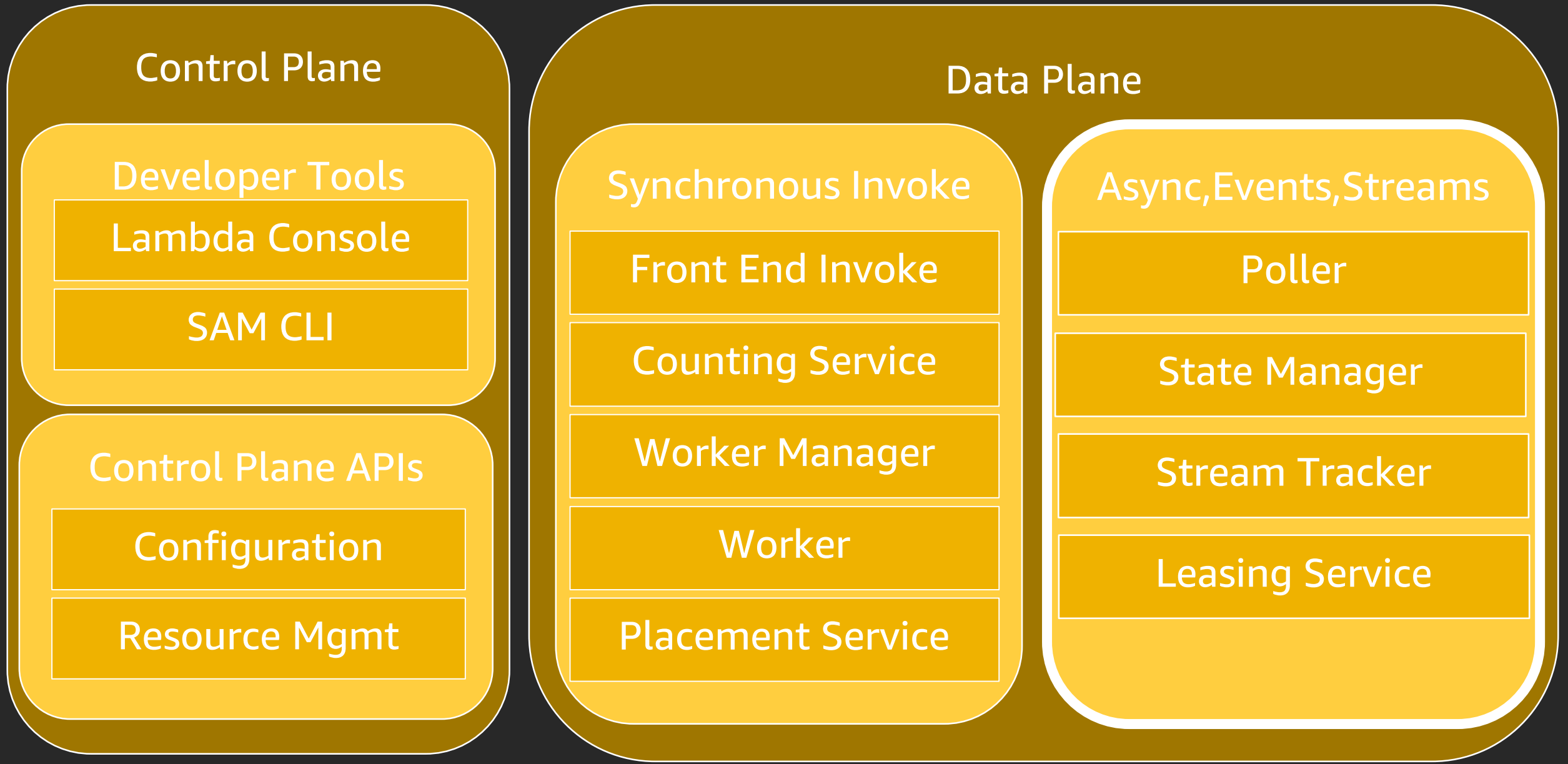

Control Plane
Developer Tools
Lambda Console
SAM CLI
Control Plane APIs
Configuration
Resource Mgmt
Data Plane
Synchronous Invoke
Front End Invoke
Counting Service
Worker Manager
Worker
Placement Service

Synchronous First Time Invoke or Scale Up



Synchronous Repeat Invoke




Control Plane
Developer Tools
Lambda Console
SAM CLI
Control Plane APIs
Configuration
Resource Mgmt
Data Plane
Synchronous Invoke
Front End Invoke
Counting Service
Worker Manager
Worker
Placement Service
Async,Events,Streams
Poller
State Manager
Stream Tracker
Leasing Service

Poller

Consumes events and ensures they are processed

State Manager or Stream Tracker

Handles scaling by managing Pollers and event or stream source resources

Leasing Service

Assigns Pollers to work on a specific event or streaming source

Event Processing

Handling Asynchronous and Event Based Invokes

AWS re:Invent

aws

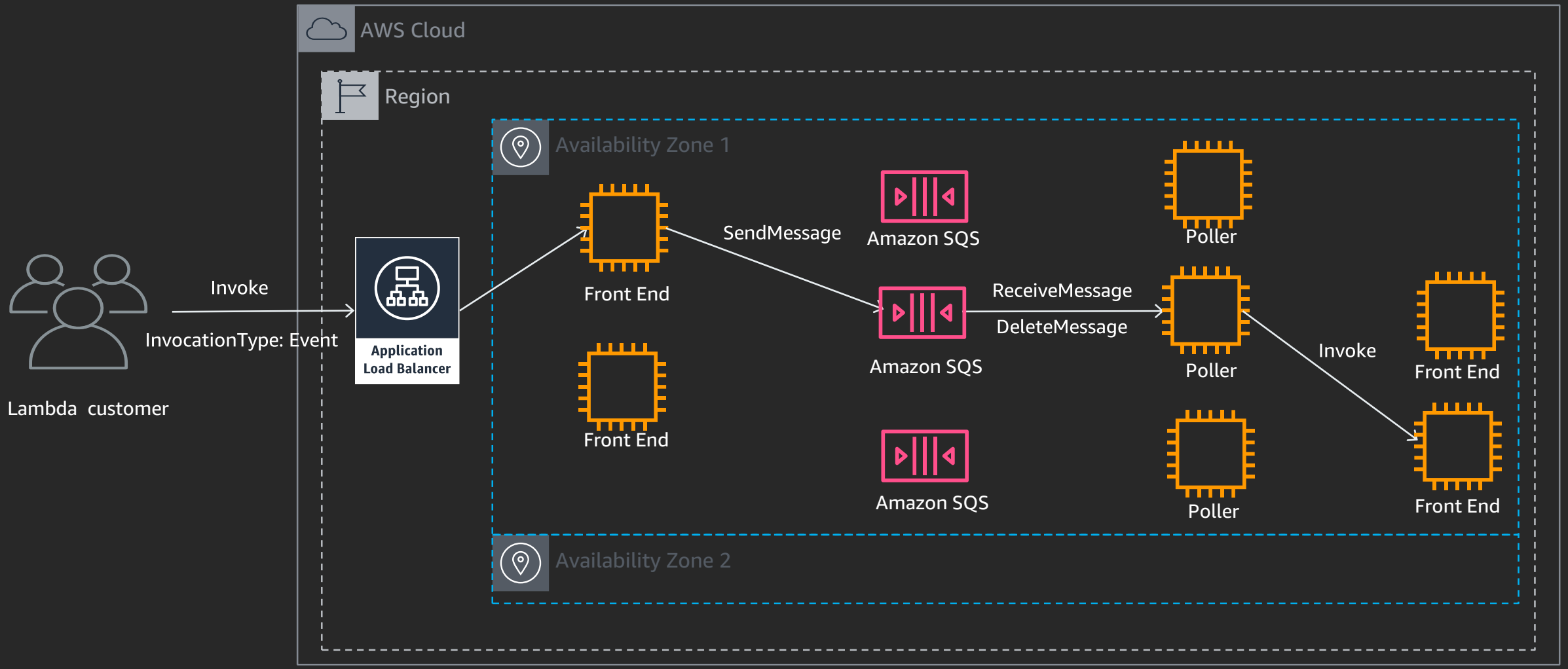

airbnb
mlbam
COMCAST
Experian
Coca-Cola
LexisNexis
New York Public Library
Benchling
Pentair
DBRS
instacart
Zillow
ALTS
VOGUE
23andMe
SCHOLASTIC
CITRIX
McGraw Hill
The Seattle Times
ADITYA BIRLA GROUP
VidRoll
Localytics
asurion
Expedia
Atlassian
FireEye
vevo
DAZN
EDIZIONI CONDÉ NAST S.P.A.
BUSTLE
THOMSON REUTERS
ViewLift
SQUARE ENIX
Nextdoor
AirAsia
ZAPPROVED
FannieMae
THE HONEST CO.
Vertafore
Unleash your potential
FUNNY OR DIE
Genome Institute of Singapore
MORNINGSTAR
Haier
PINPOINT
ISCS
la Mutuelle Générale
LA NACION
rachio
Discovery COMMUNICATIONS
FICO
twilio
finra
VERACODE
MADSACK MEDIENGRUPPE
BEACHBODY ON DEMAND
AUTODESK
WARBY PARKER
doppler labs
Cornelius
WHALEROCK INDUSTRIES
cimpress
Nishitetsu
Optimizely
ASTRO DIGITAL
SAISON INSURANCE
Serenova shine bright
nextel
TÁ NAS SUAS MÃOS.
INTERCOM
COUNTY OF ALAMEDA CALIFORNIA
NATIONAL GEOGRAPHIC
AdRoll
DBS
edmunds
teralytic
HomeAway
Fender
Ringier
Blissfully
Dunelm
CSIRO
Mercury
Accolade
A CLOUD GURU
iRobot
openfit
STRAVA

"We believe 'once you go serverless you never go back,' hence all of our new developments are serverless by default."

Denis Bauer

Head of Cloud Computing & Bioinformatics

Event Invoke



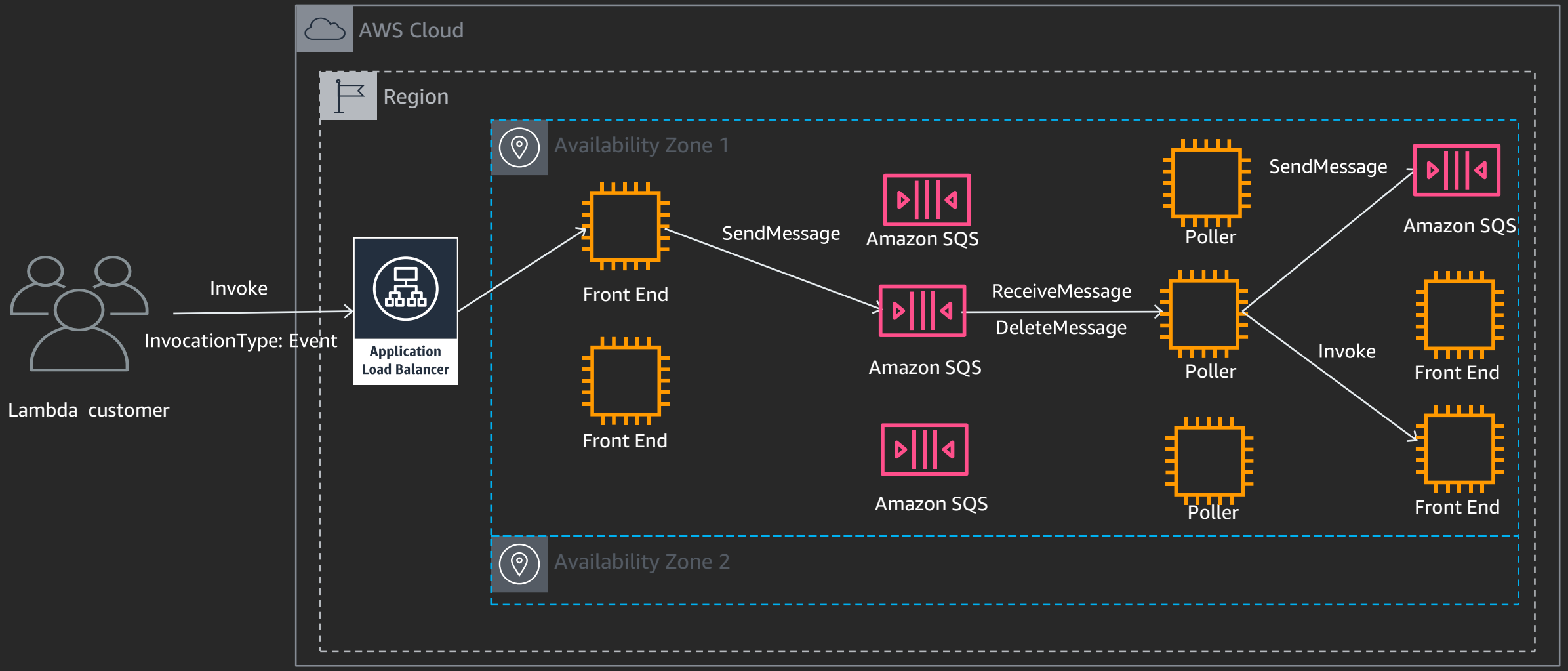
Event Destinations

Providing Callbacks After Processing

AWS re:Invent

aws

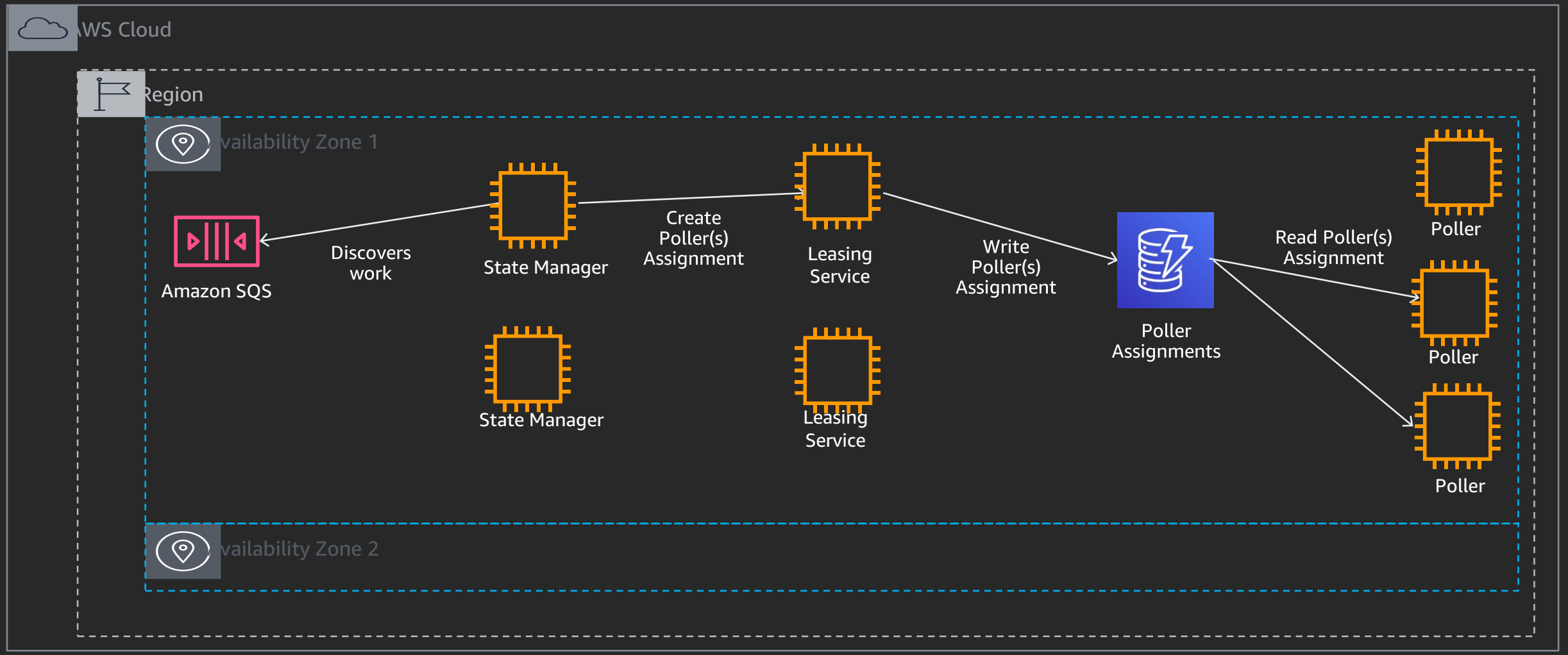
Event Invoke Destination



Scaling Up and Down

Dynamically Adjusting to Process Incoming Event Volume

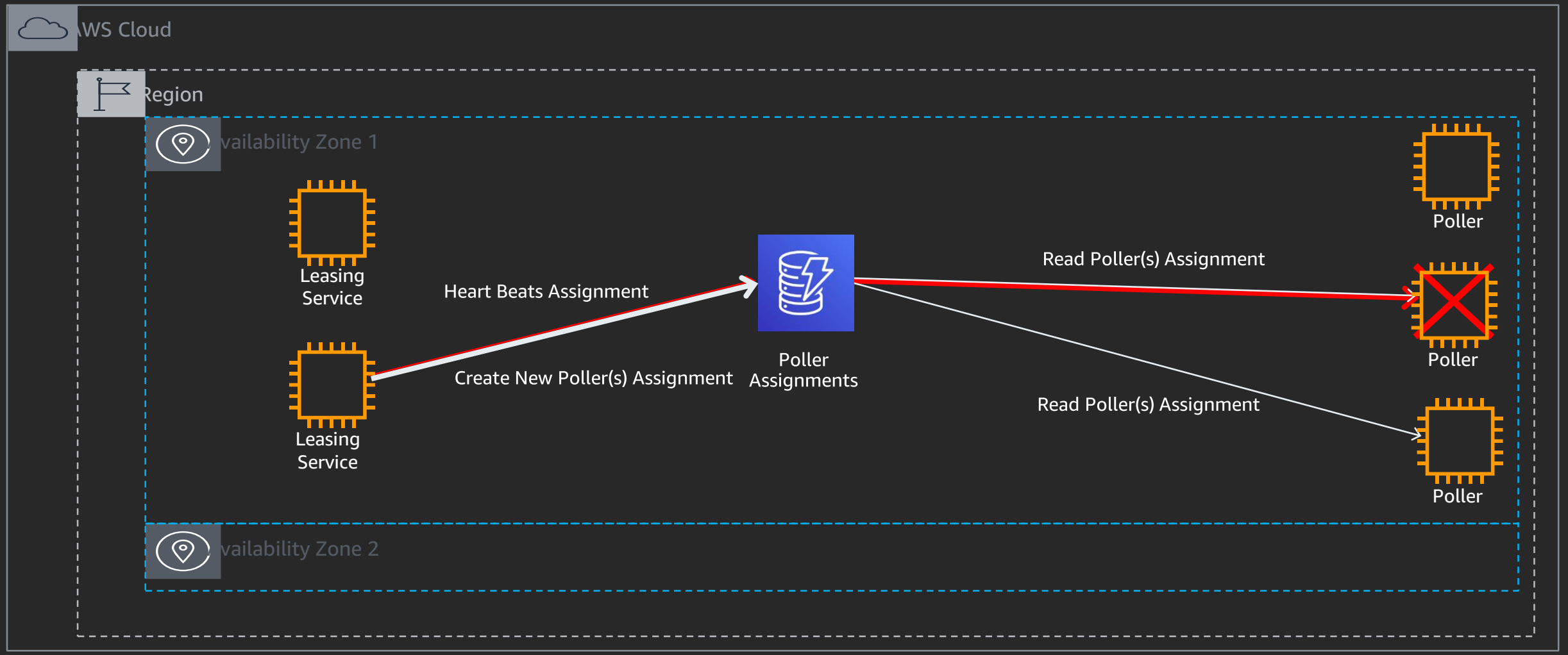
Event Invoke Scaling Up and Down



Handling Errors

Detecting and Recovering From Event Based Error Conditions

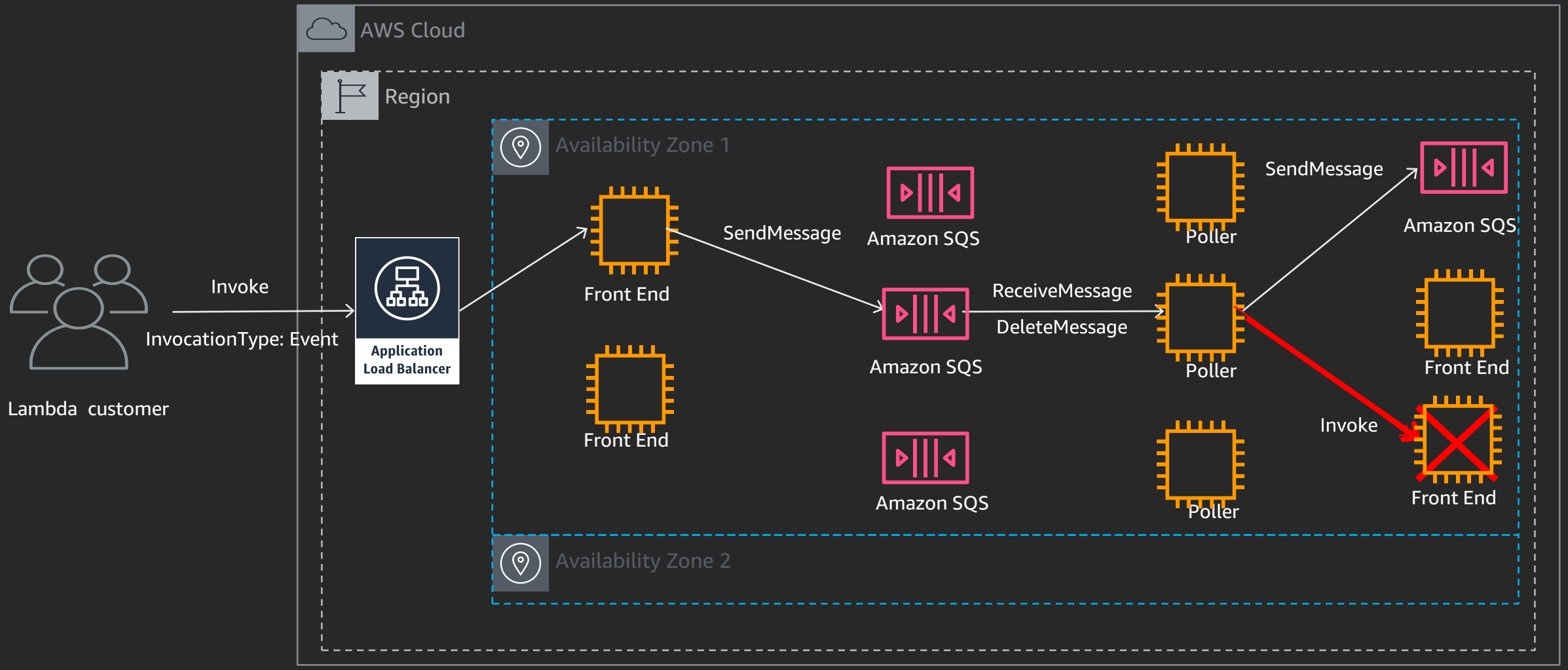
Event Invoke Error Handling



Retry Policies

Flexibility on how to try to process events in failure conditions

Event Invoke Error Handling Options



Stream Processing

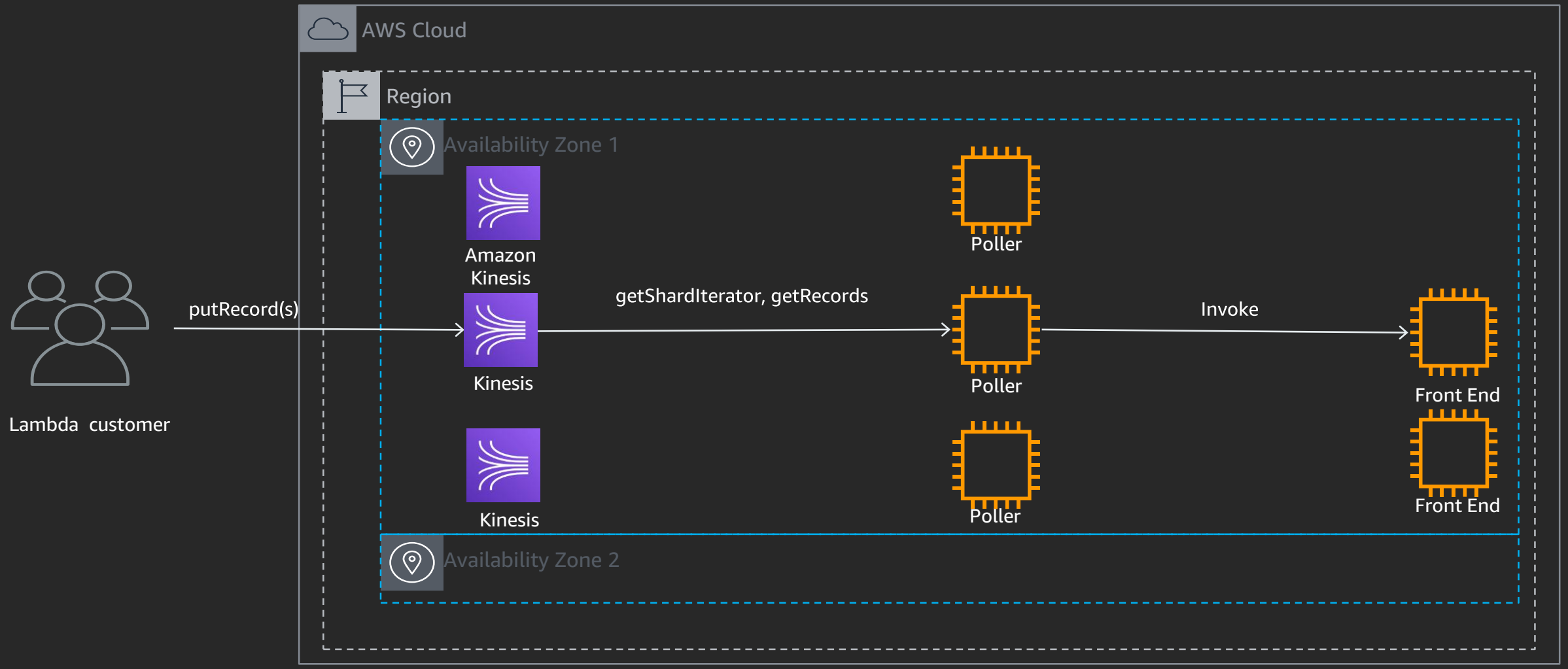
Handling High Scale Data Source Invokes

AWS re:Invent

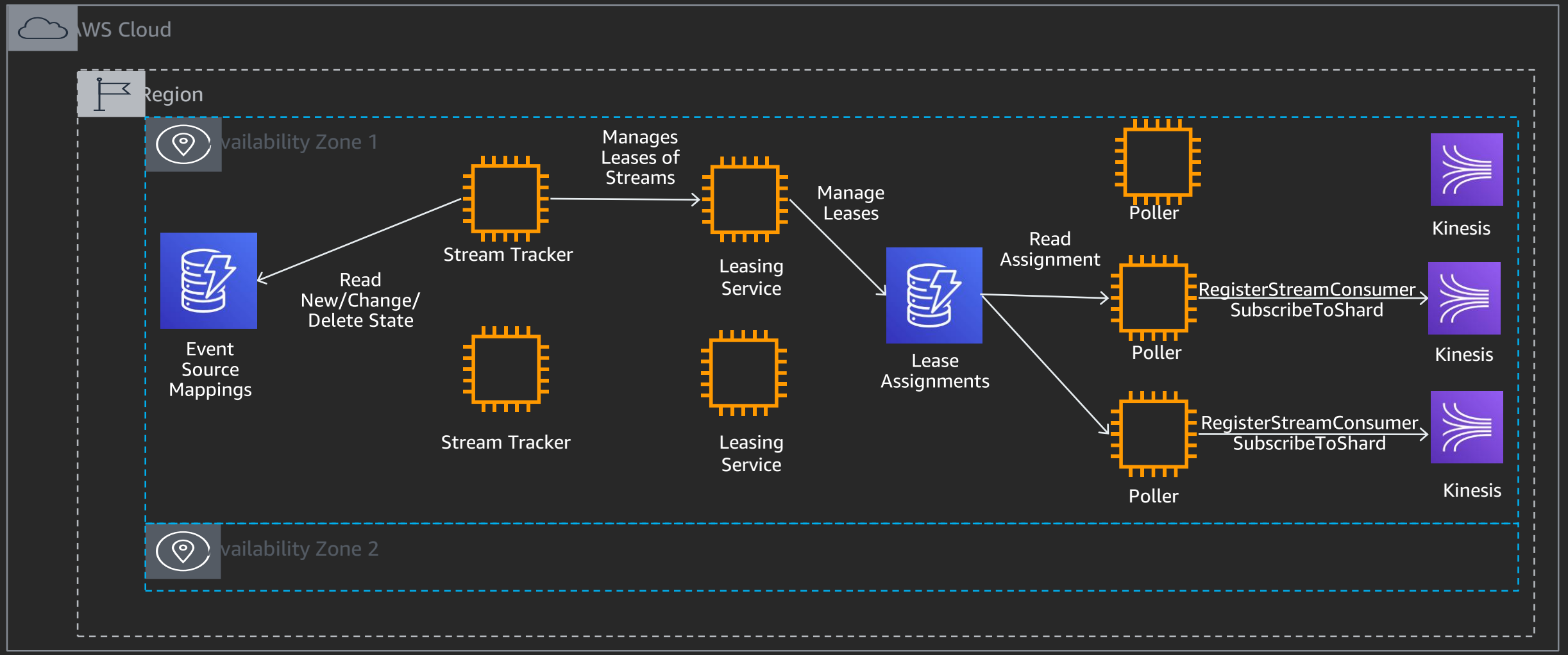
aws


airbnb
mlbam
COMCAST
Experian
Coca-Cola
LexisNexis
New York Public Library
Benchling
Pentair
DBRS
instacart
Zillow
ALTS
VOGUE
23andMe
SCHOLASTIC
CITRIX
McGraw Hill
The Seattle Times
ADITYA BIRLA GROUP
VidRoll
Localytics
asurion
Expedia
Atlassian
FireEye
vevo
DAZN
EDIZIONI CONDÉ NAST S.P.A.
BUSTLE
THOMSON REUTERS
ViewLift
SQUARE ENIX
Nextdoor
AirAsia
ZAPPROVED
FannieMae
THE HONEST CO.
Vertafore
Unleash your potential
FUNNY OR DIE
Genome Institute of Singapore
MORNINGSTAR
Haier
PINPOINT
ISCS
la Mutuelle Générale
LA NACION
rachio
Discovery COMMUNICATIONS
FICO
twilio
finra
VERACODE
MADSACK MEDIENGRUPPE
BEACHBODY ON DEMAND
AUTODESK
WARBY PARKER
doppler labs
Cornelius
WHALEROCK INDUSTRIES
cimpress
Nishitetsu
Optimizely
ASTRO DIGITAL
SAISON INSURANCE
Serenova shine bright
nextel
TÁ NAS SUAS MÃOS.
INTERCOM
COUNTY OF ALAMEDA CALIFORNIA
NATIONAL GEOGRAPHIC
AdRoll
DBS
edmunds
teralytic
HomeAway
Fender
Ringier
Blissfully
Dunelm
CSIRO
Mercury
Accolade
A CLOUD GURU
iRobot
openfit
STRAVA

Stream Invoke Processing



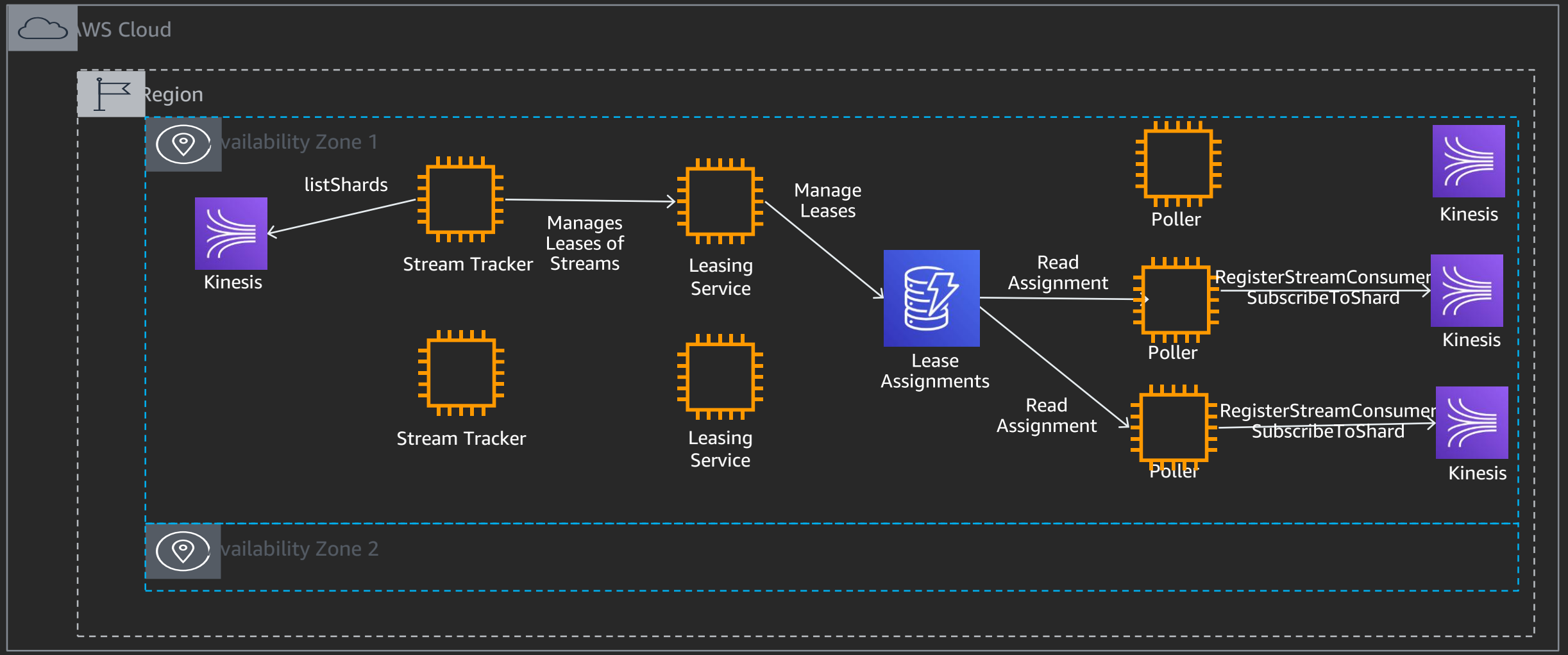
Stream Invoke Establishment



Scaling Up and Down

Dynamically Adjusting to Process Incoming Stream Volume

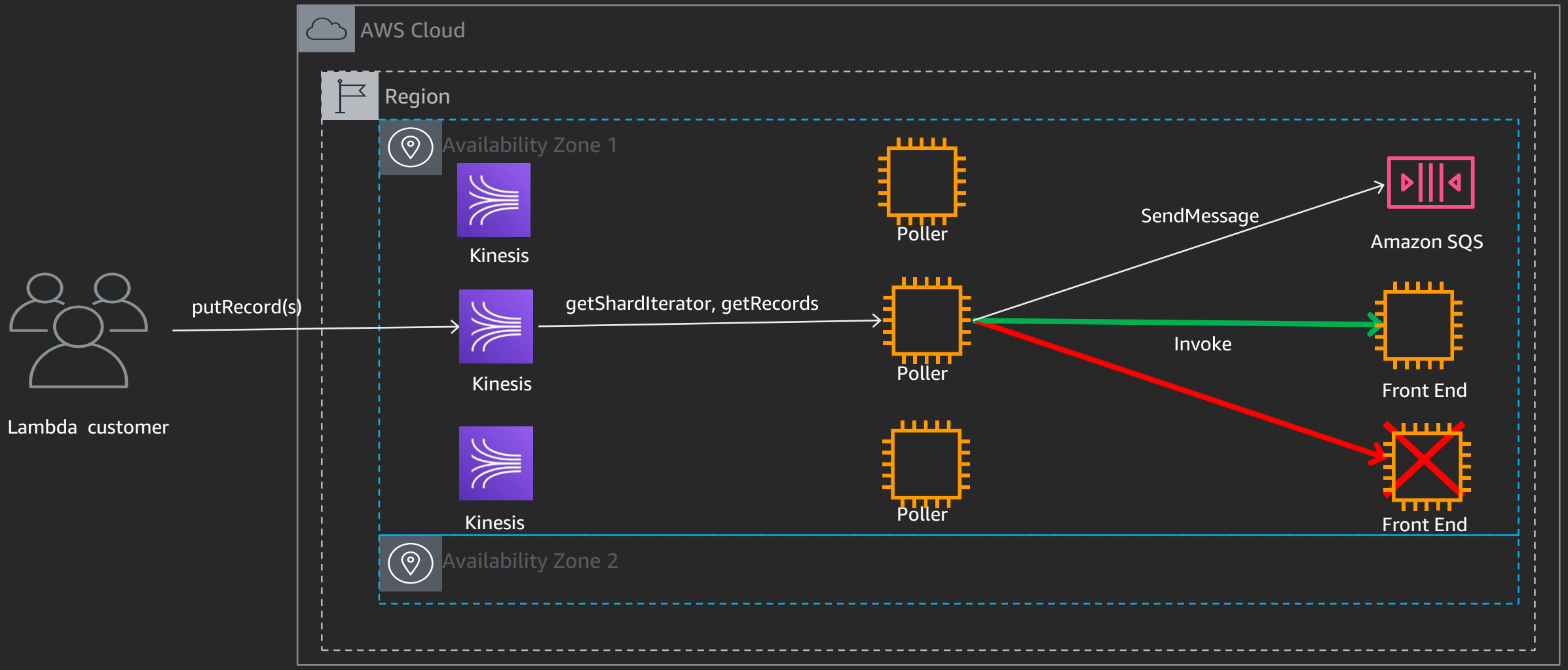
Stream Invoke Scaling Up and Down



Handling Errors

Detecting and Recovering From Stream Based Error Conditions

Stream Invoke Error Handling



Predictable Performance

Handling low latency interactive workloads

AWS re:Invent

aws

Lambda Provisioned Concurrency

Enable provisioned concurrency to keep your functions initialized and hyper-ready to respond within double-digit milliseconds.

Provisioned **Concurrency**

Why Not Provisioned Capacity?

- Predictability
- Fault Tolerance
- Customer Behavior

Capacity: Depends on hardware performance.

Concurrency: Does not depend on hardware.

Capacity: Need to model host and AZ failure.

Concurrency: Host and AZ failure tolerance built in.

Capacity: Customer and business changes can effect how efficiently you use capacity.

Concurrency: Unit of work done.

Why Not Provisioned Rate (Requests per Second)?

- Scale factors and contention
- Batching and work-per-unit
- Heterogeneous requests

RPS: Does not consider work-per-unit (batching, etc).

Concurrency: Considers work-per-unit.

RPS: Does not consider cost-per-unit (e.g. *list* vs *get* APIs)

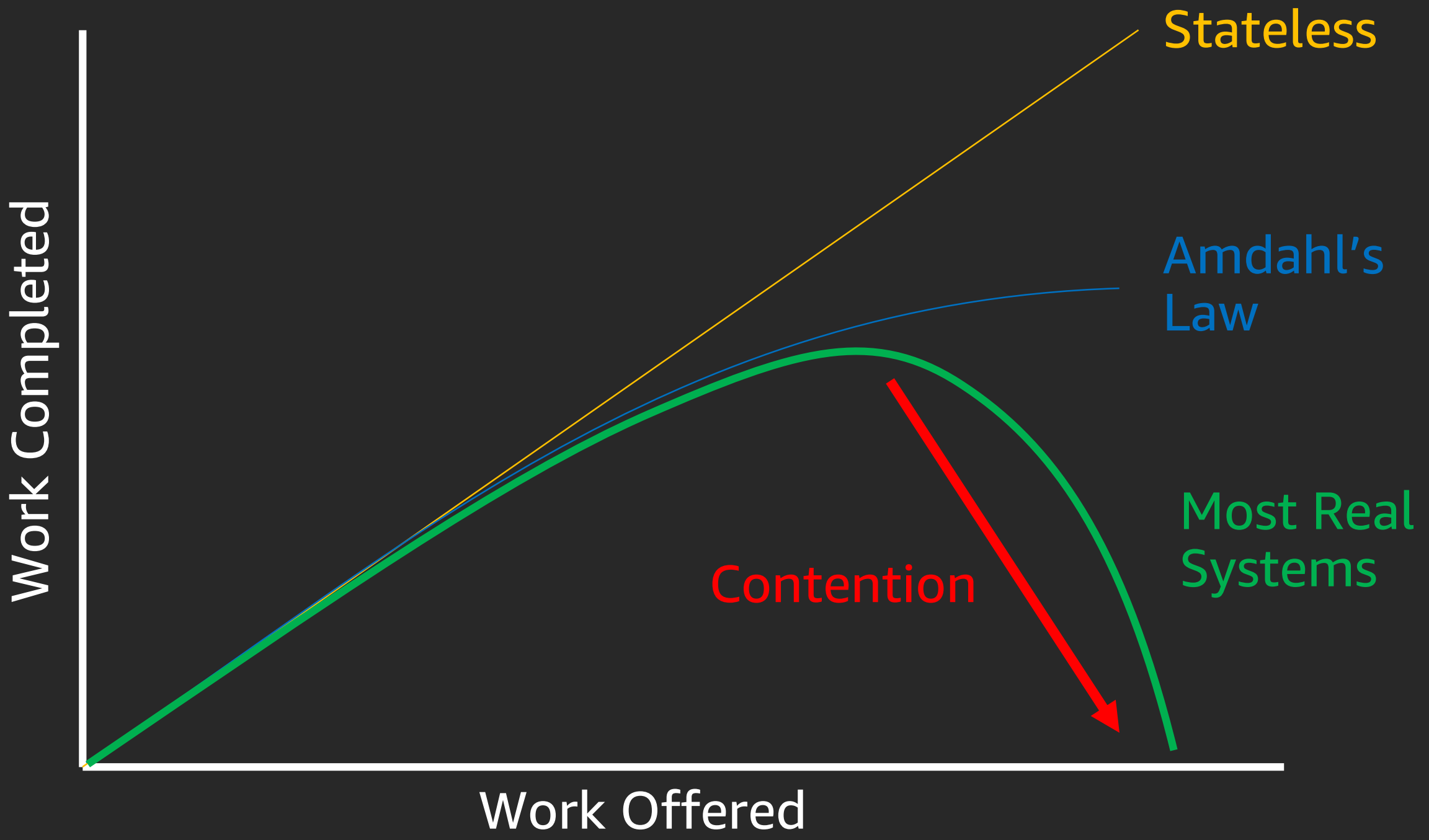
Concurrency: Considers cost-per-unit.

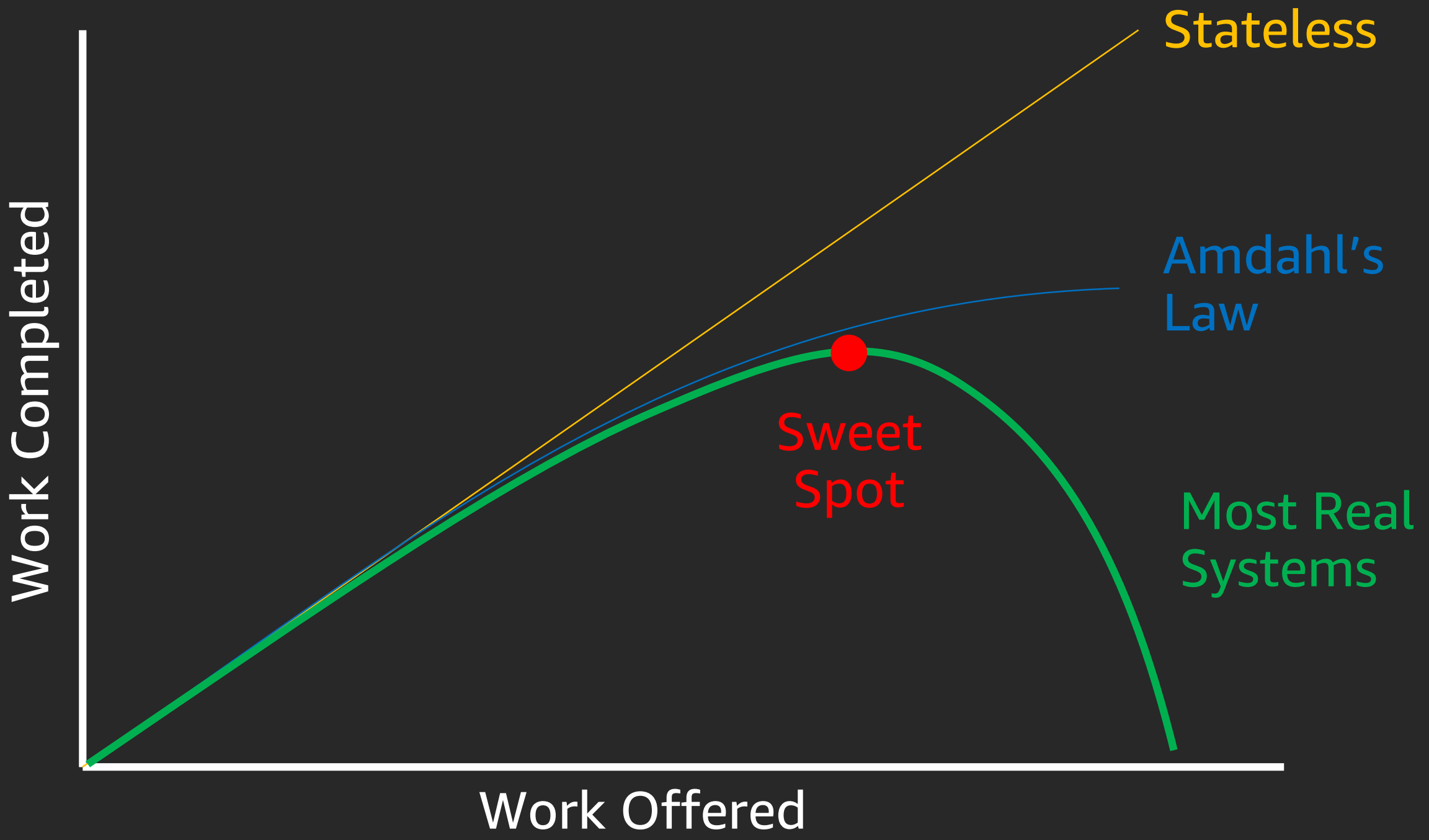
RPS: Does not consider effects of contention.

Concurrency: Contention and concurrency are directly linked.


Work Completed
Work Offered
Naïve,
Stateless &
Non-
Coordinating


Stateless
Amdahl's Law
Most Real Systems
Work Completed
Work Offered


Stateless
Amdahl's Law
Most Real Systems
Contention
Work Completed
Work Offered


Stateless
Amdahl's Law
Sweet Spot
Most Real Systems
Work Completed
Work Offered

Concurrency

=

Rate

x

Latency

Concurrency

=

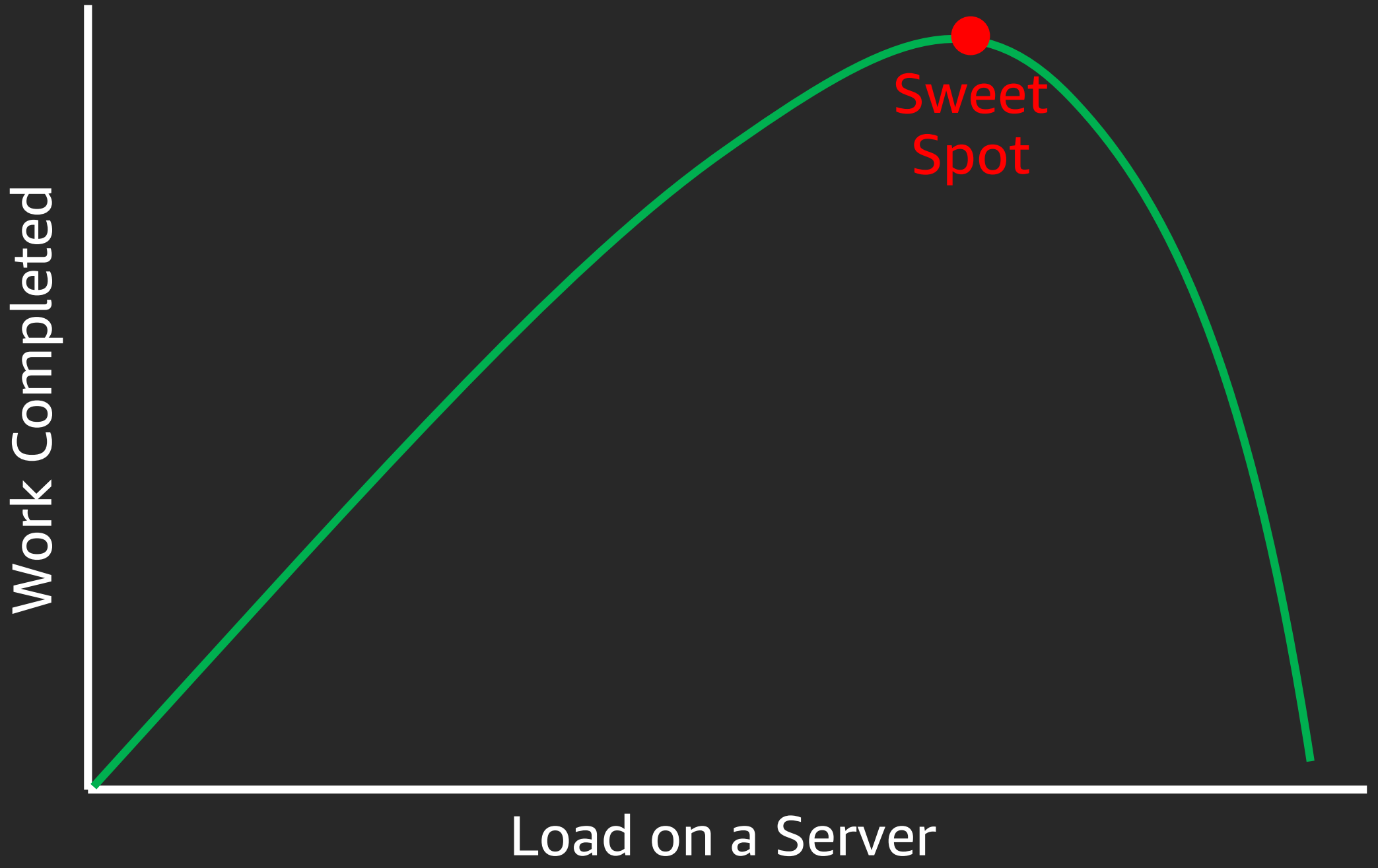
Rate

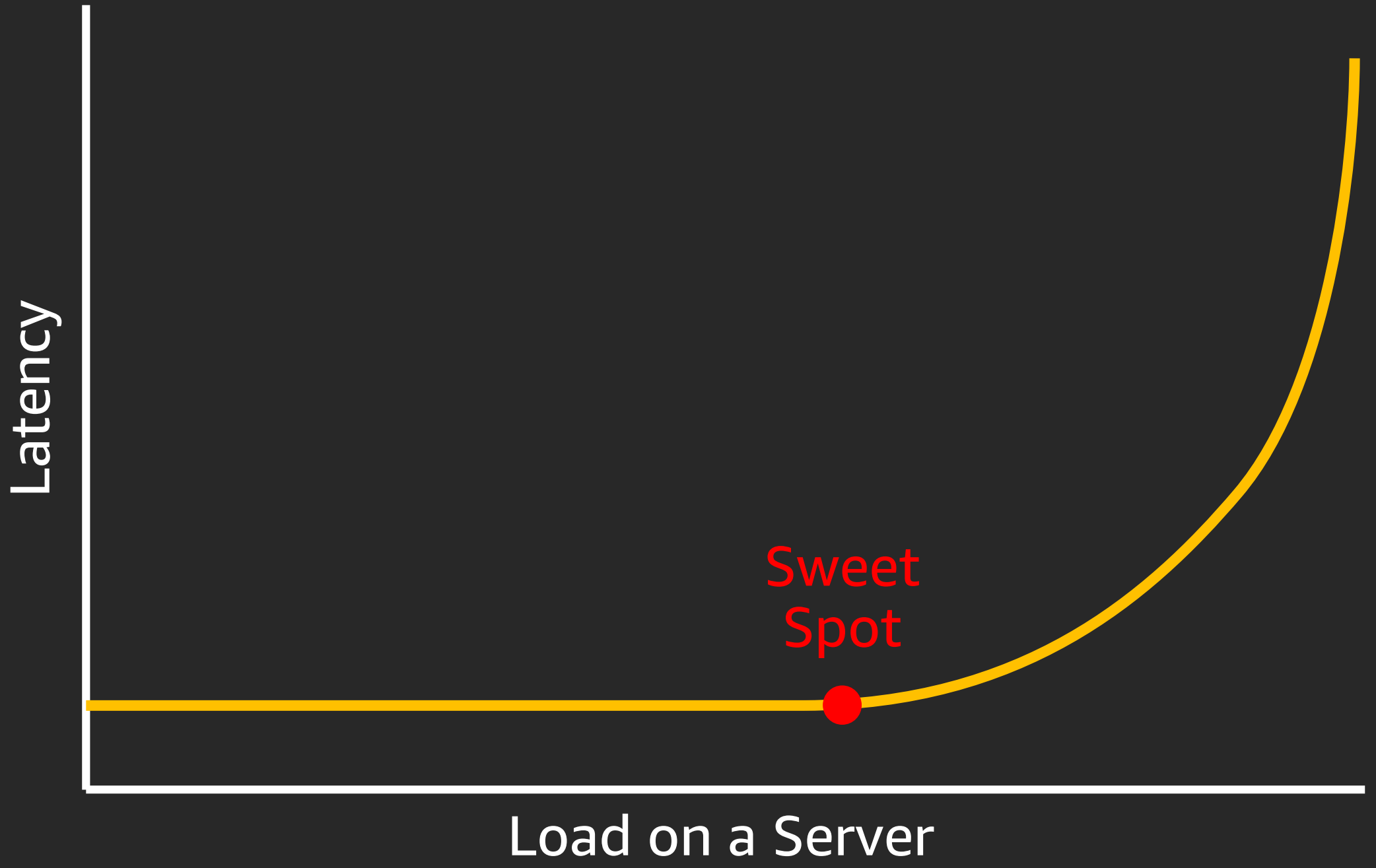
x

Latency

(which depends on concurrency)

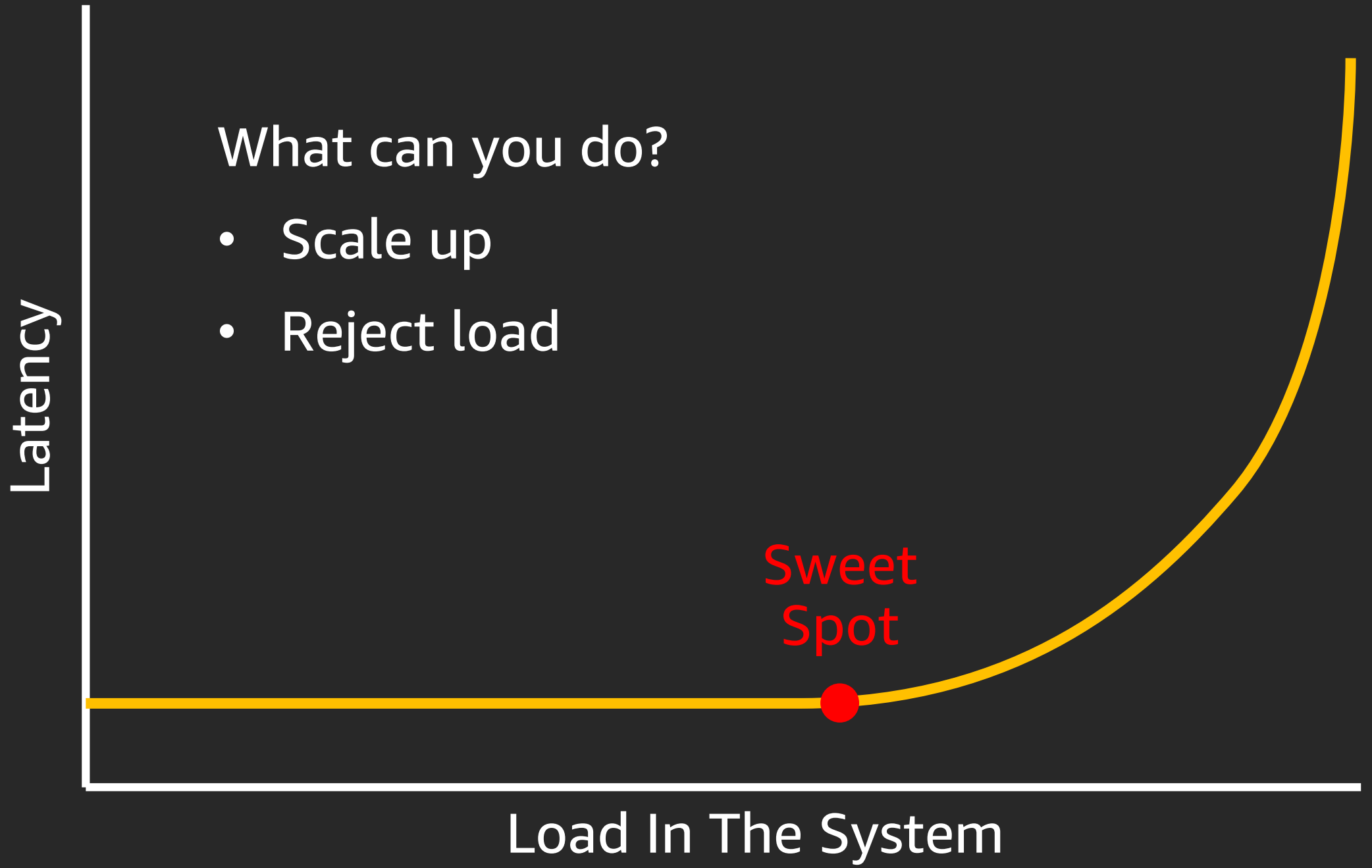
Provisioned Concurrency


Work Completed
Sweet
Spot
Load on a Server


Latency
Sweet
Spot
Load on a Server

What can you do?

- Scale up
- Reject load



Scale Up

- Scaling up takes time
  - Milliseconds to seconds on Lambda
  - Seconds to minutes on servers
- Autoscaling is great, but approximate
  - Works *by proxy*
  - Predicting the future is hard

Reject Load

- Traditional infrastructure does this *by proxy*
- Lambda does it directly on concurrency!
  - Per-function concurrency limits

Scale Fast

Control Load

Provision Ahead

Control Load

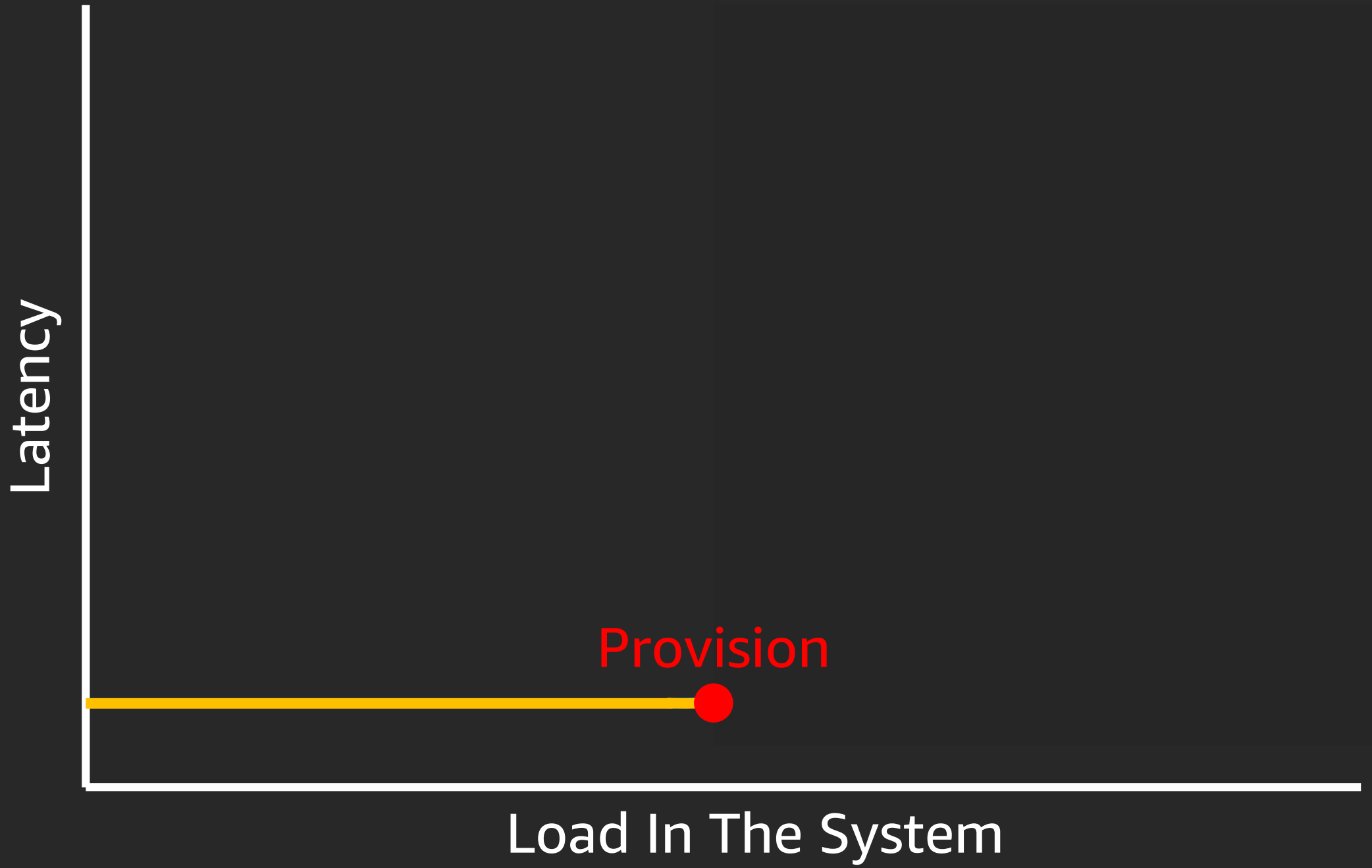
With per-function concurrency

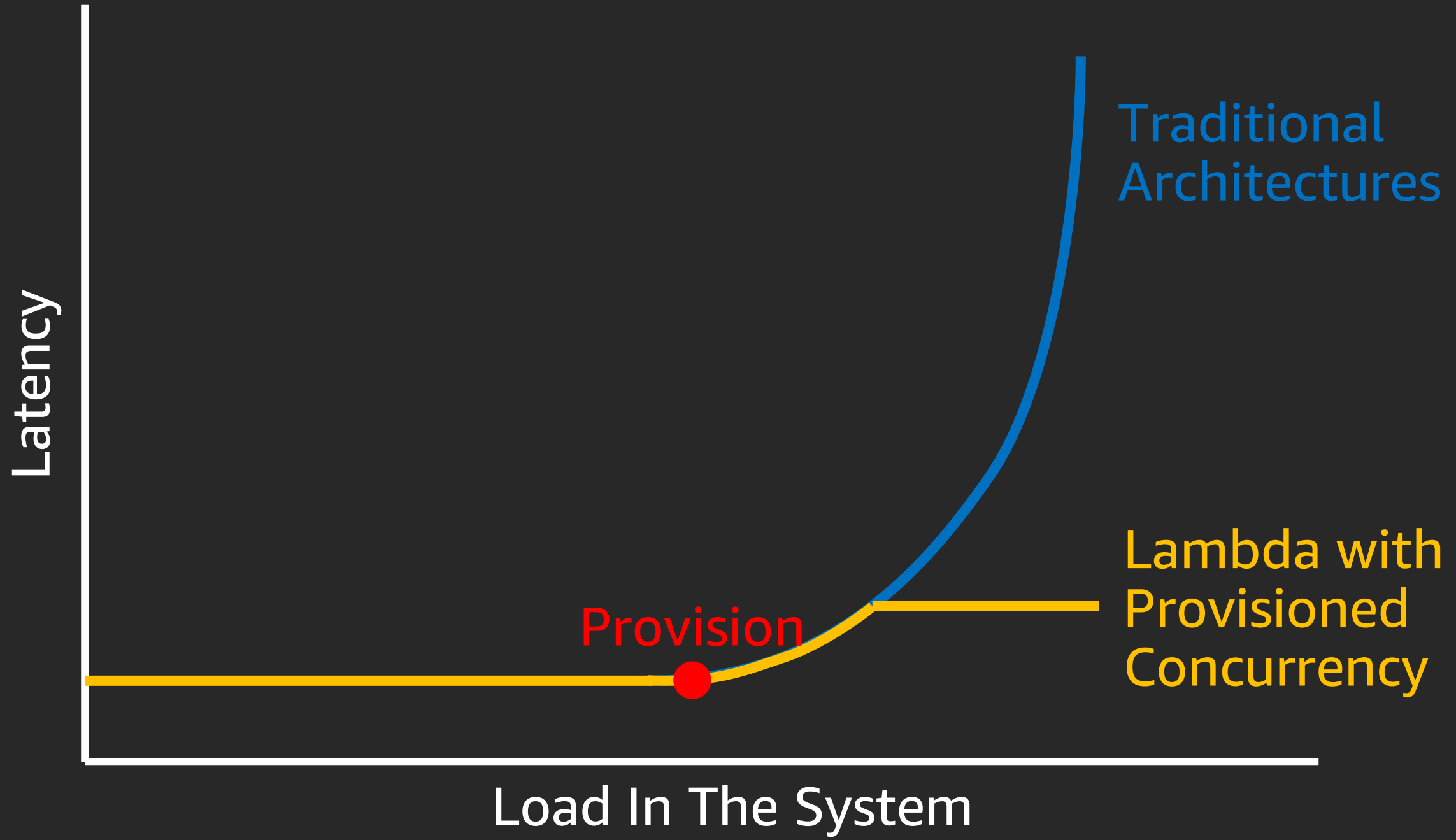
Scale Fast

Built into Lambda

Provision Ahead

Available Now!


Latency
Provision
Load In The System

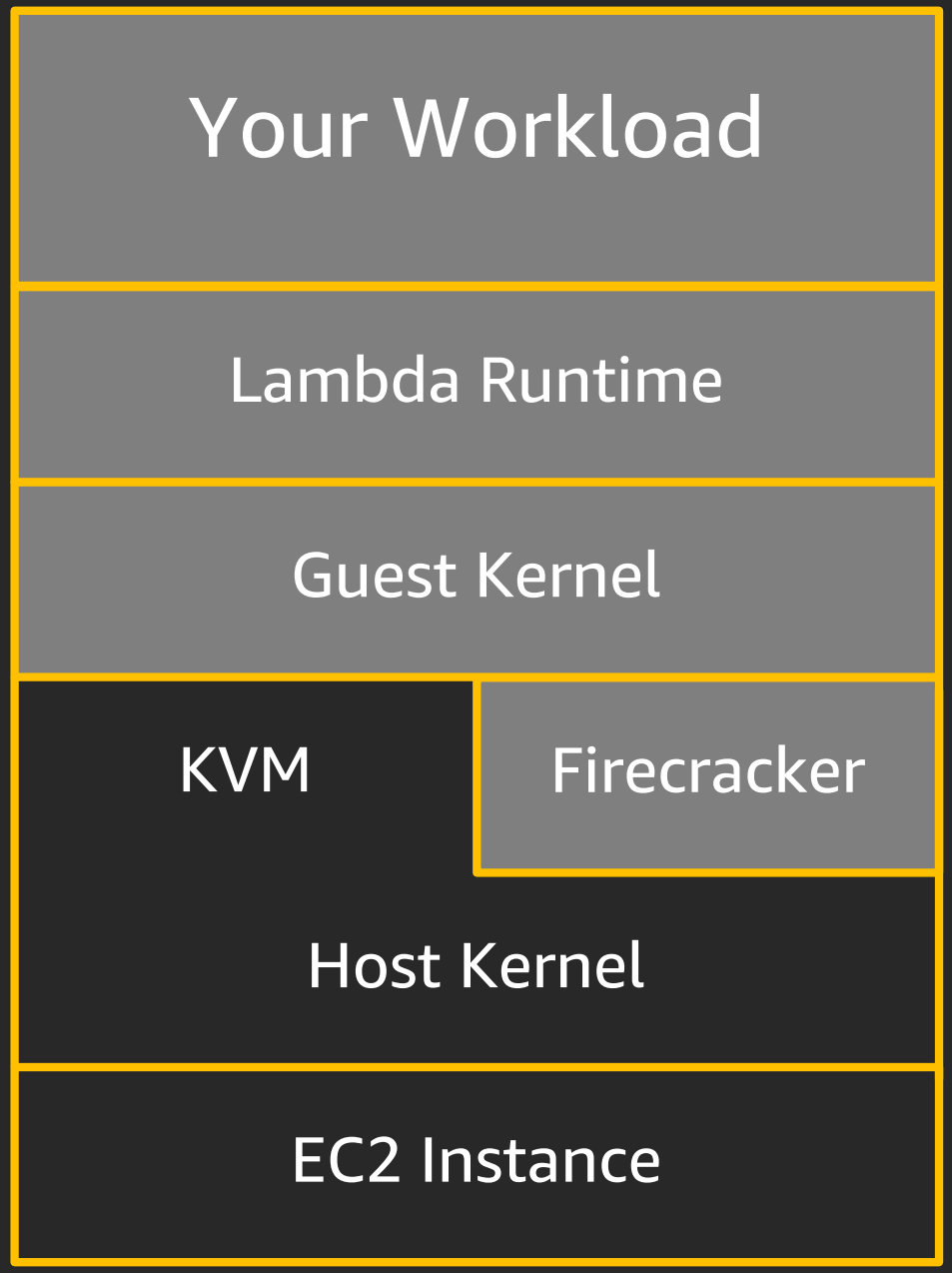

Latency
Traditional Architectures
Lambda with Provisioned Concurrency
Provision
Load In The System

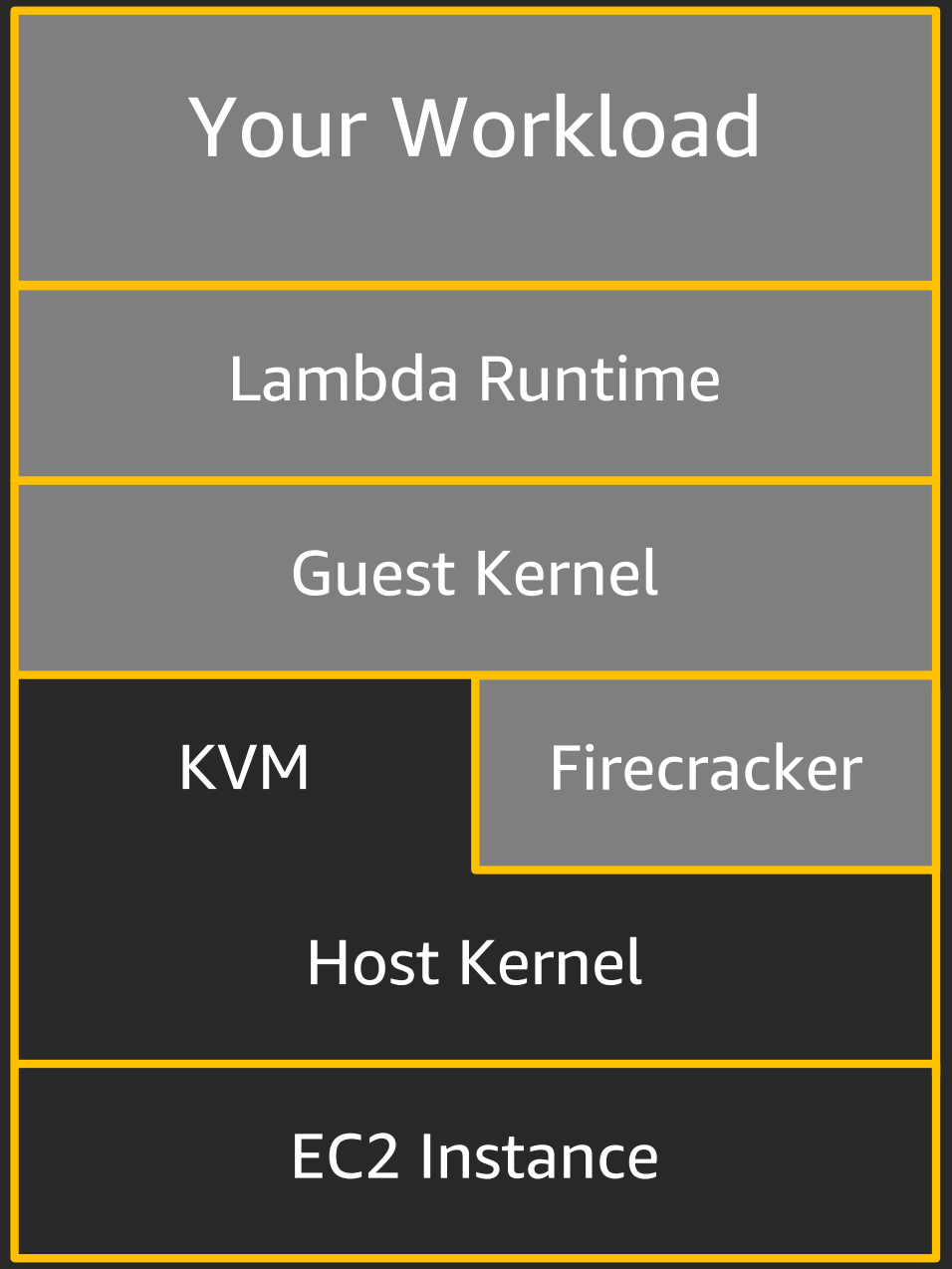
Lambda Isolation Under The Covers

AWS re:Invent

aws

Firecracker


Your Workload
Lambda Runtime
Guest Kernel
KVM
Firecracker
Host Kernel
EC2 Instance



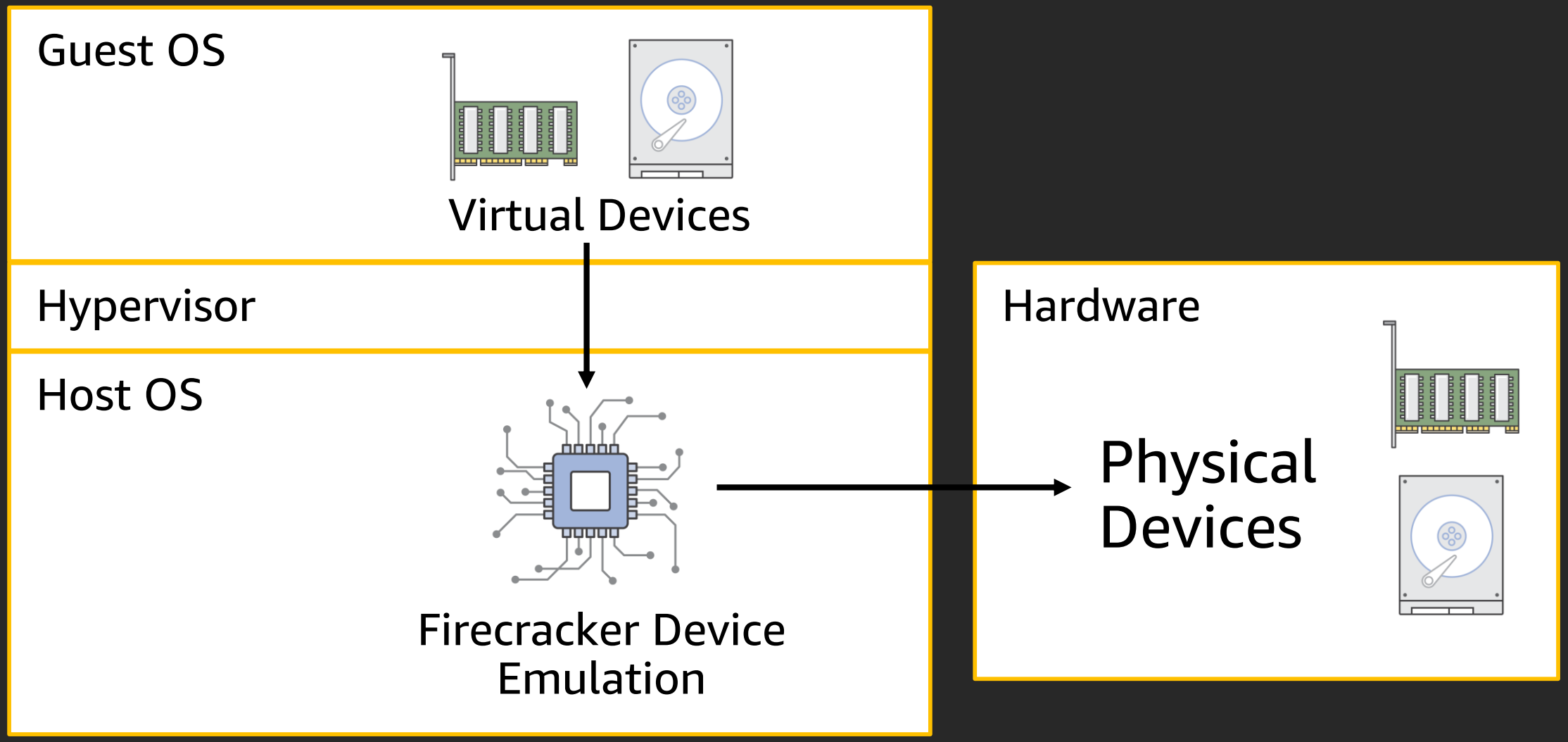
Dedicated to your function

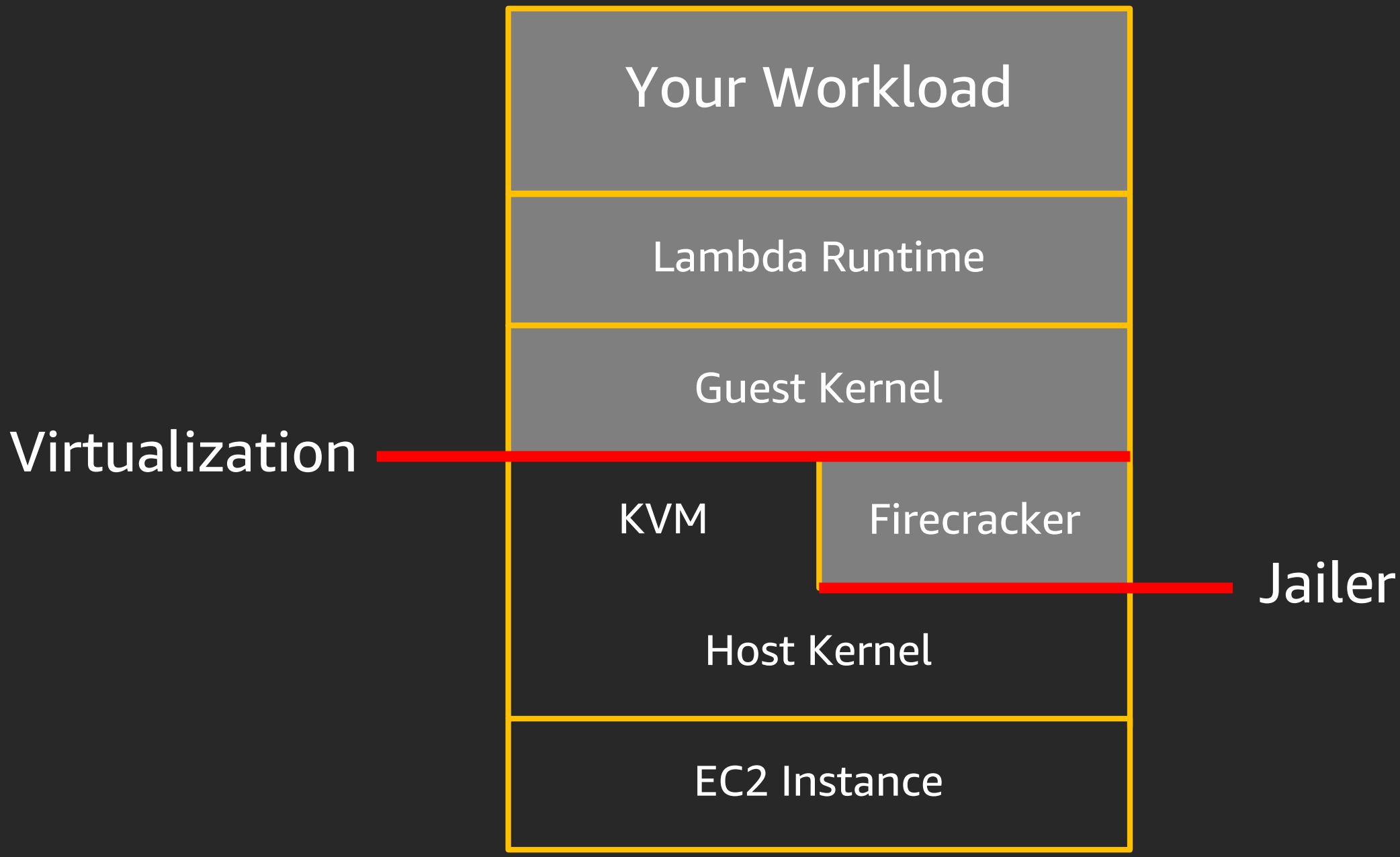
What Does KVM Do?

- Programs the host CPU to do secure hardware virtualization (HVM)
- Low-level virtualization details
  - Memory management, paging, etc
- Abstracts hardware details

What Does Firecracker Do?

- Configures KVM
- Device emulation
  - Pretends to be an SSD, NIC, etc.
- Performance isolation
- Optimized for serverless
  - Fast moving, low overhead

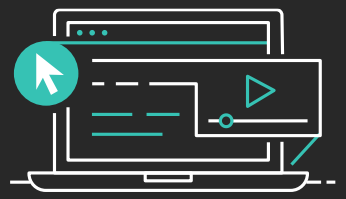

Guest OS
Virtual Devices
Hypervisor
Host OS
Firecracker Device Emulation
Hardware
Physical Devices


Your Workload
Lambda Runtime
Guest Kernel
Virtualization
KVM
Firecracker
Jailer
Host Kernel
EC2 Instance

In Conclusion

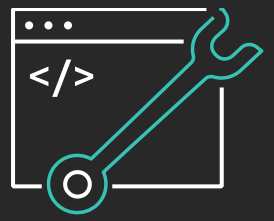
Learn serverless with AWS Training and Certification

Resources created by the experts at AWS to help you learn modern application development



Free, on-demand courses on serverless, including

- Introduction to Serverless Development
- Getting into the Serverless Mindset
- AWS Lambda Foundations
- Amazon API Gateway for Serverless Applications
- Amazon DynamoDB for Serverless Architectures



Additional digital and classroom trainings cover modern application development and computing

Visit the Learning Library at https://aws.training

aws training and certification

Thank you!

AWS re:Invent

aws

Please complete the session survey in the mobile app.