Invocare API esterne in maniera efficiente
con Amazon SQS e AWS Lambda

Gabriele Postorino

Principal Technical Account Manager

Amazon Web Services

Gabriele Postorino

Technical Account Manager
AWS Enterprise Support

- Sicurezza
- Reliability
- Operations
- Performance
- Costi
- …

- Migrazioni
- Go Live
- Eventi

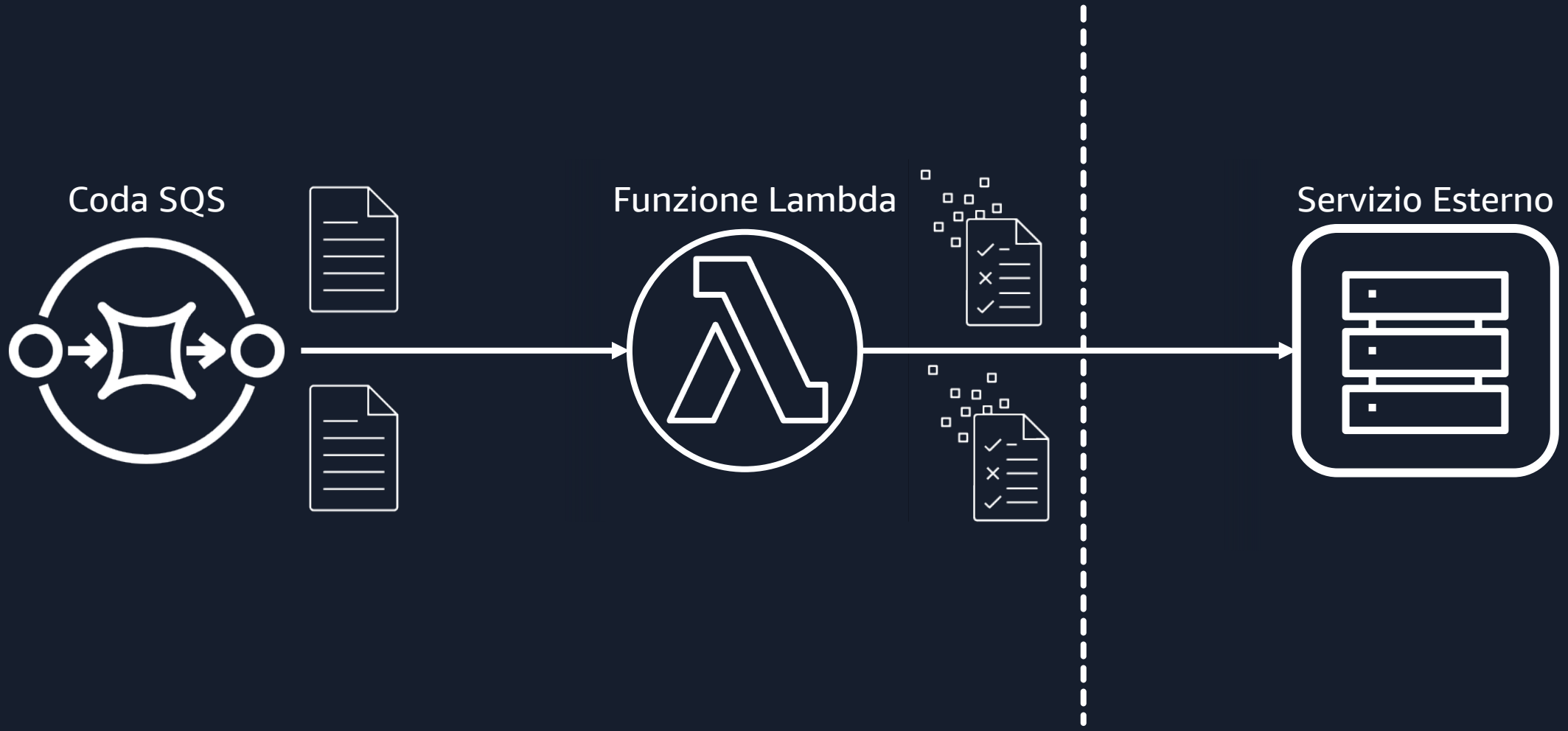
Focus

Scaling best practices
Resilience

Invocare API esterne in maniera efficiente

con Amazon SQS e AWS Lambda

Lo scenario



SQS event source mapping

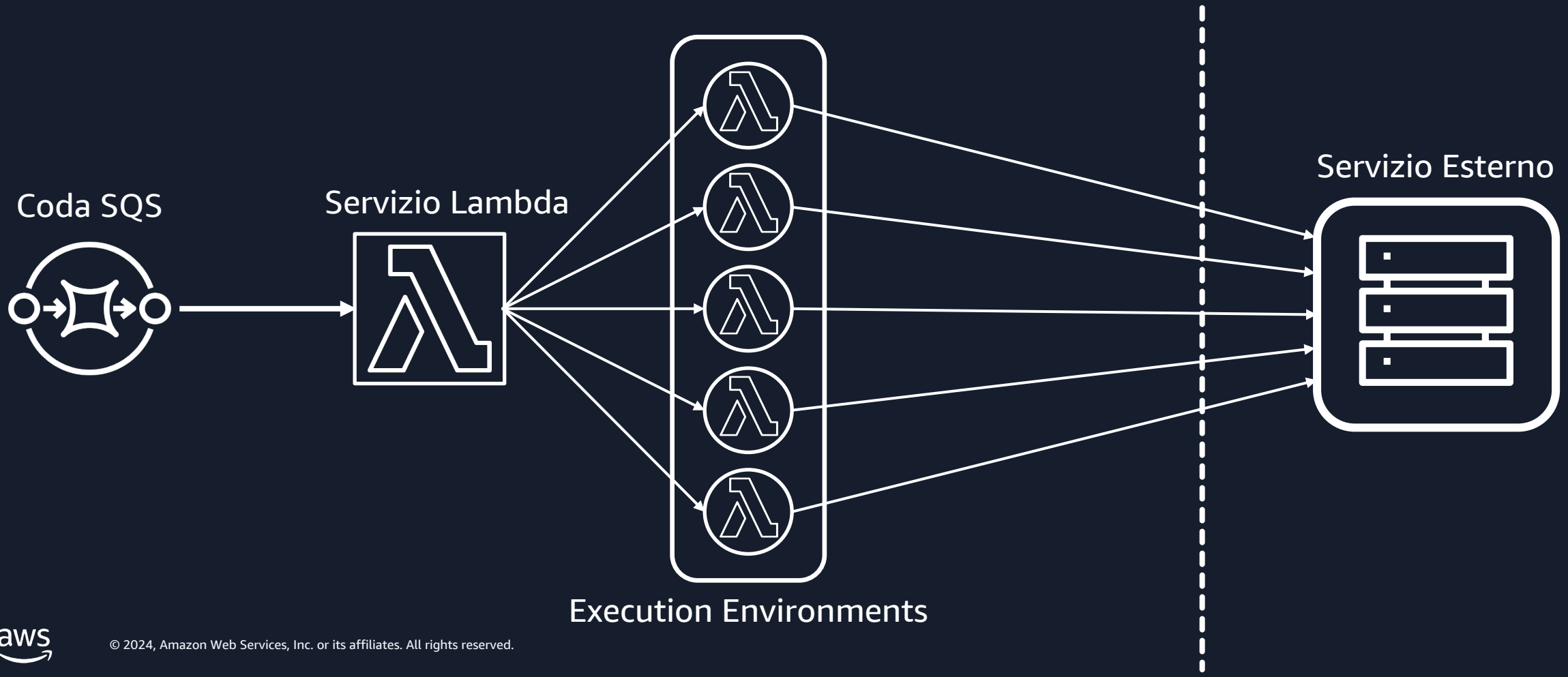
POLLING AND BATCHING

- Lambda legge i messaggi in coda e invoca la funzione attraverso un evento
- Ogni evento contiene un batch di messaggi di dimensione configurabile
- Quando l'esecuzione termina con successo i messaggi sono eliminati dalla coda

SQS Event Source Mapping

SCALING BEHAVIOUR

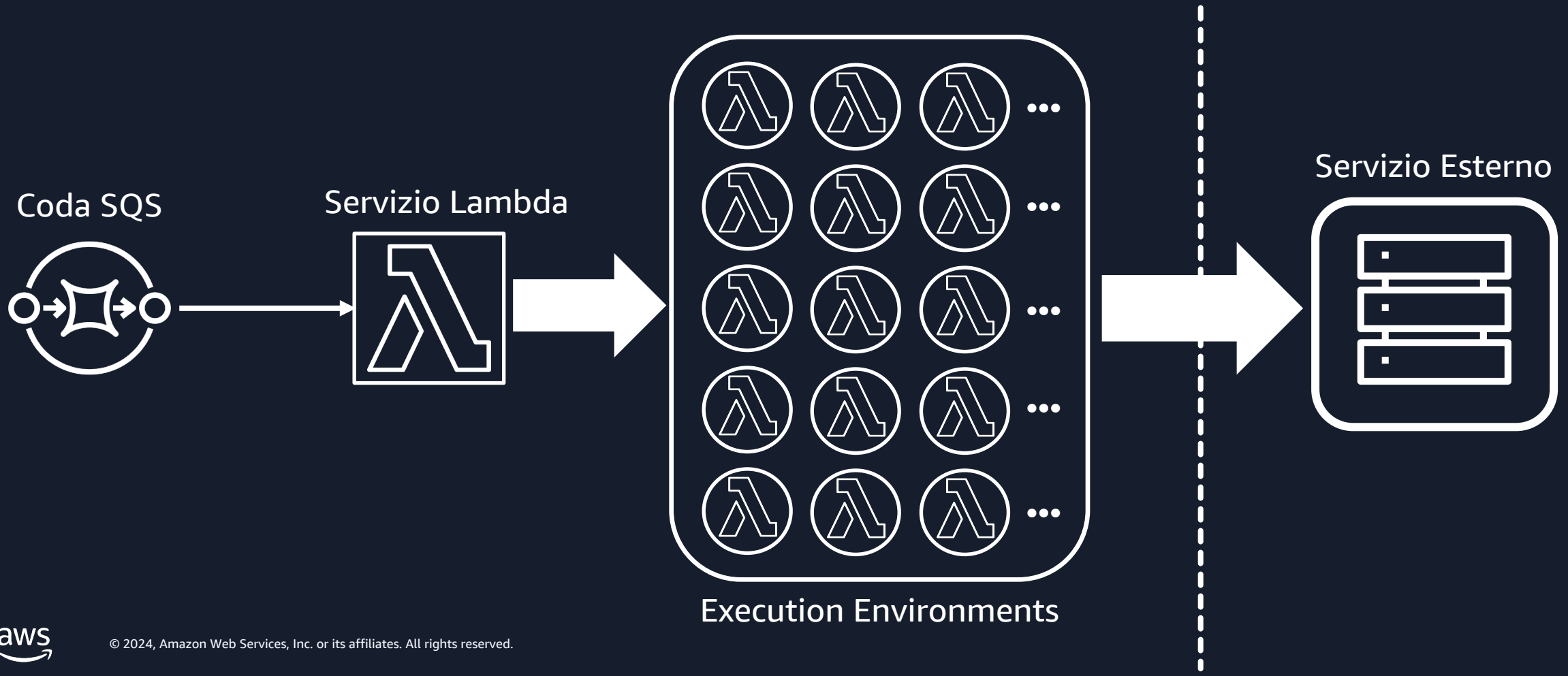
Lambda legge i messaggi in coda e invoca fino a 5 esecuzioni concorrenti



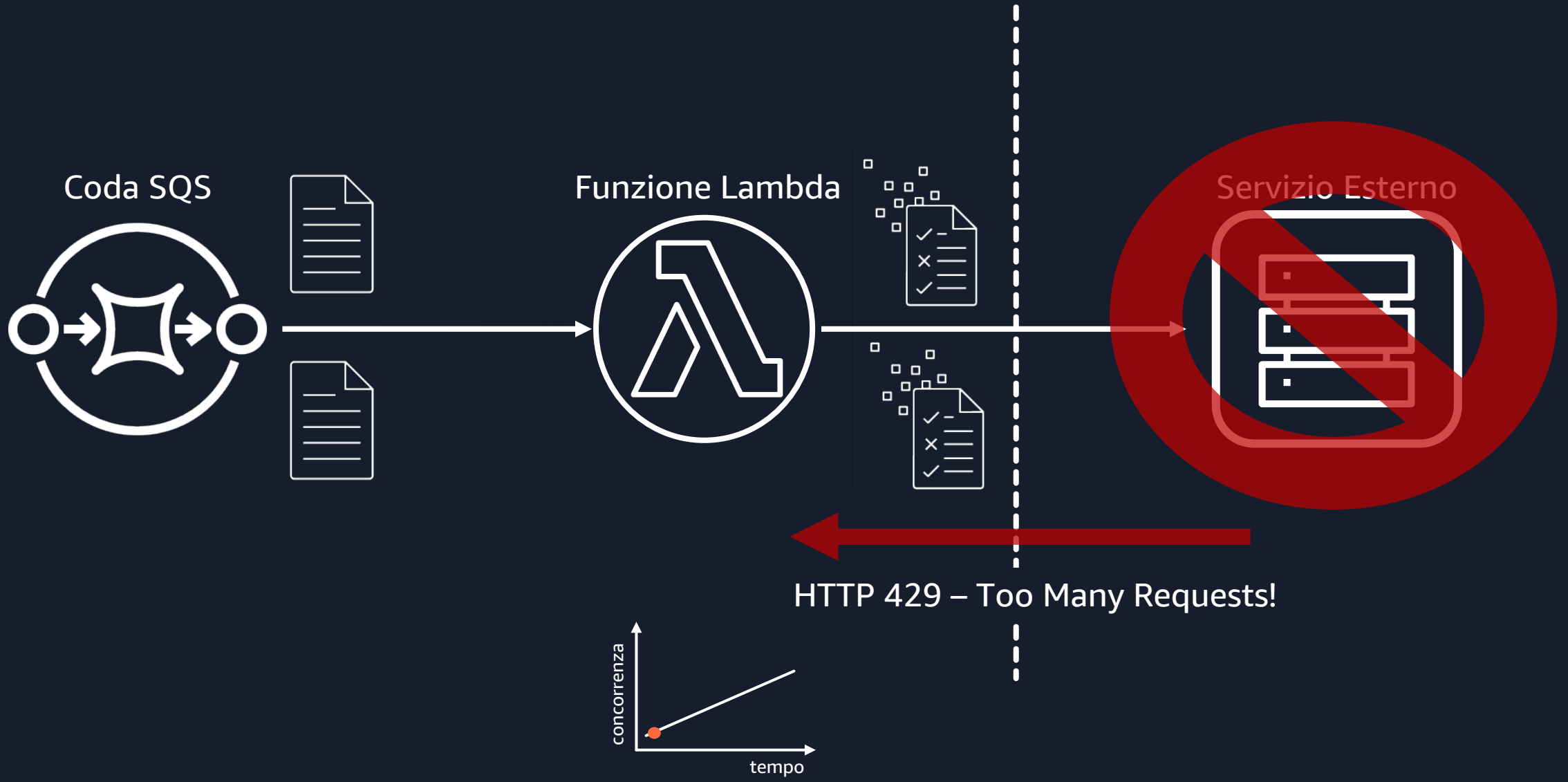
SQS Event Source Mapping

SCALING BEHAVIOUR

Può scalare fino a 1000 esecuzioni concorrenti gradualmente (+300/min)



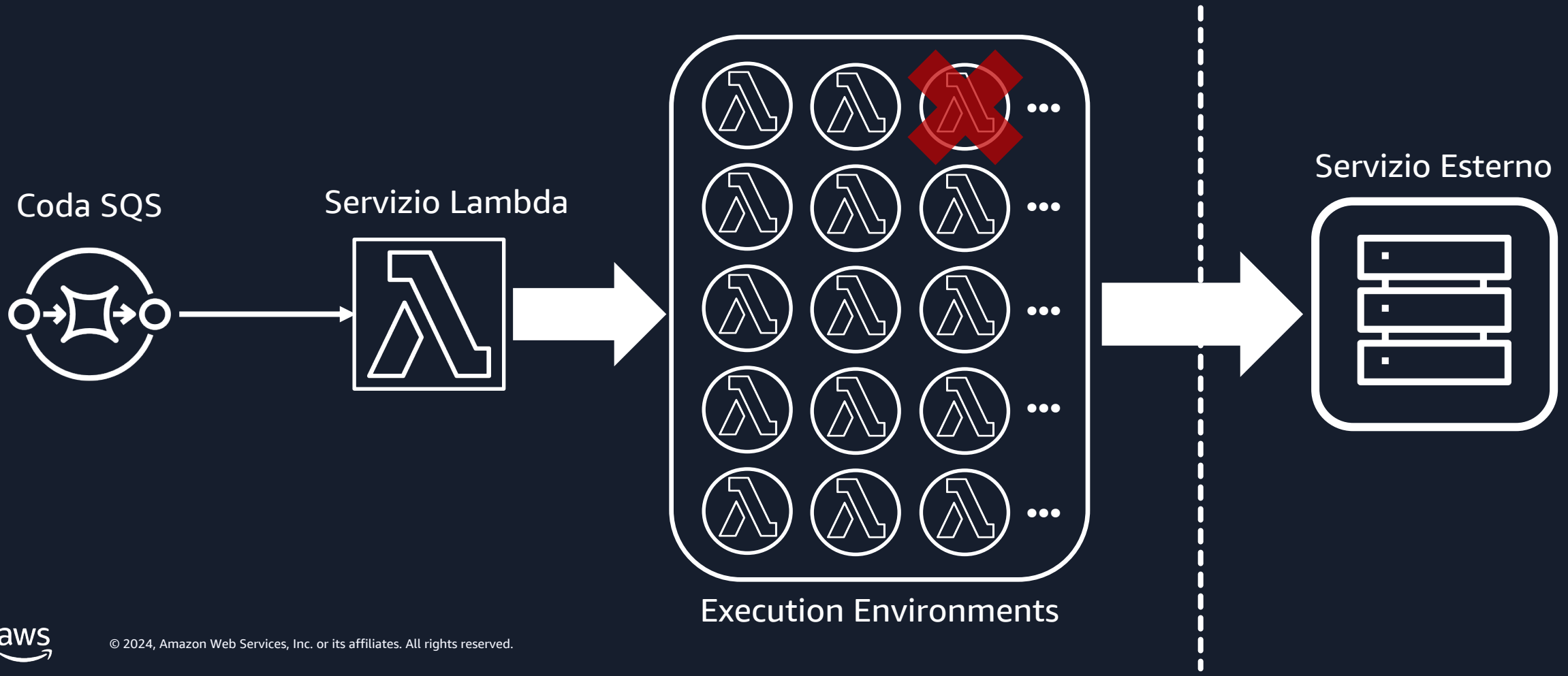
Lo scenario



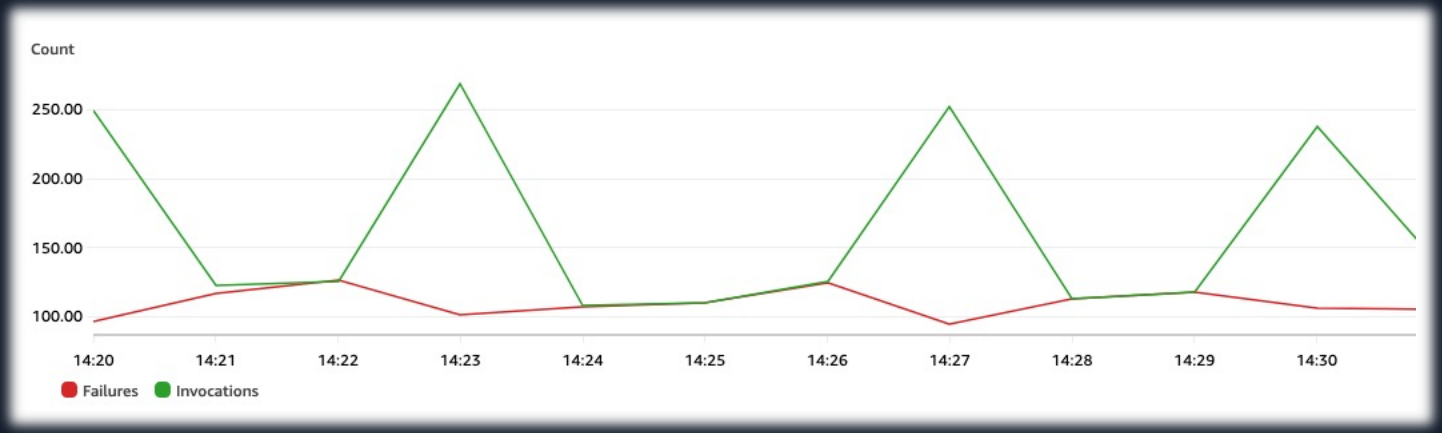
SQS Event Source Mapping

SCALING BEHAVIOUR

Quando una esecuzione fallisce, Lambda riduce la concorrenza in automatico



Quindi, tutto ok?

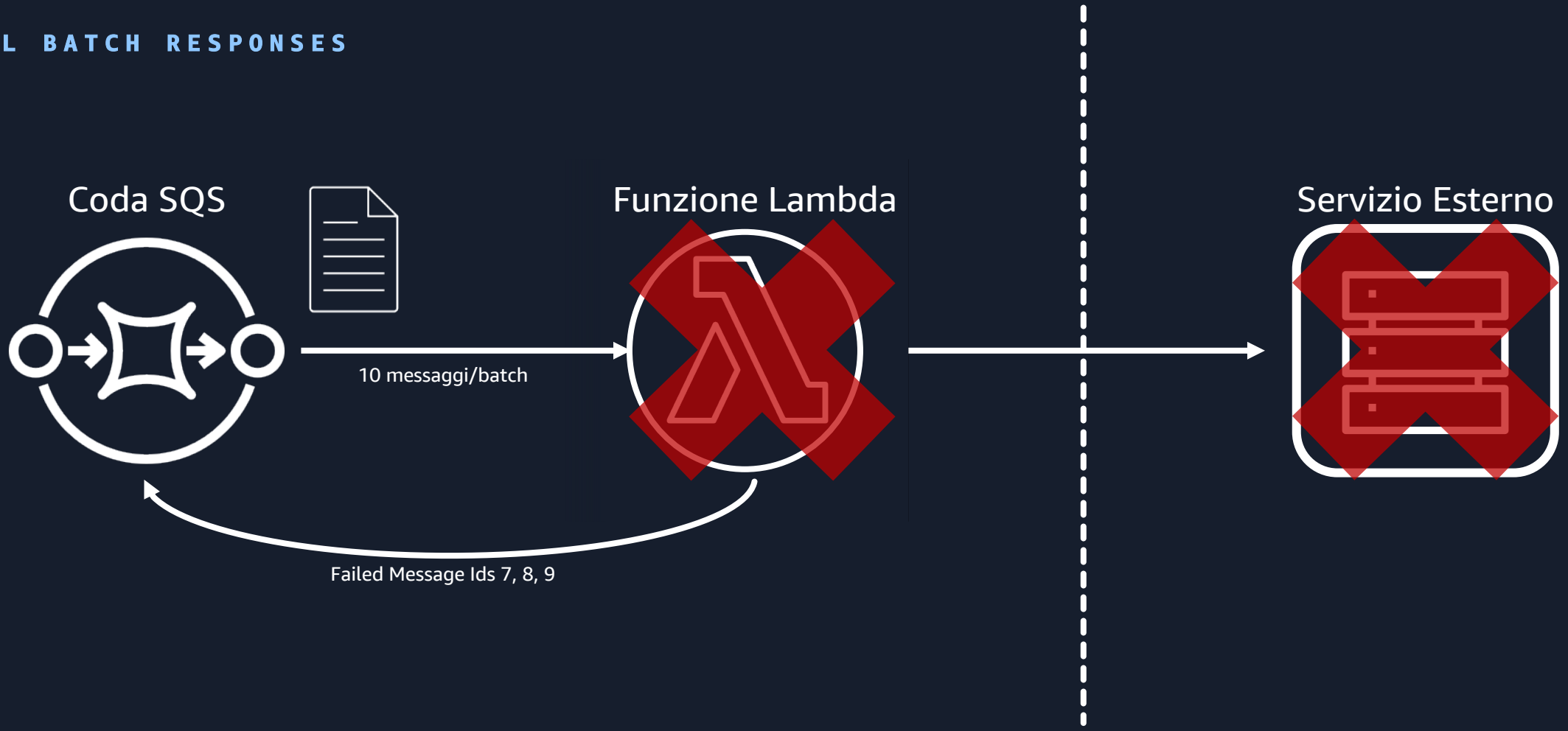


Non proprio…

Se una esecuzione della funzione fallisce, tutti i messaggi del batch tornano in coda!

Gestione degli errori

PARTIAL BATCH RESPONSES



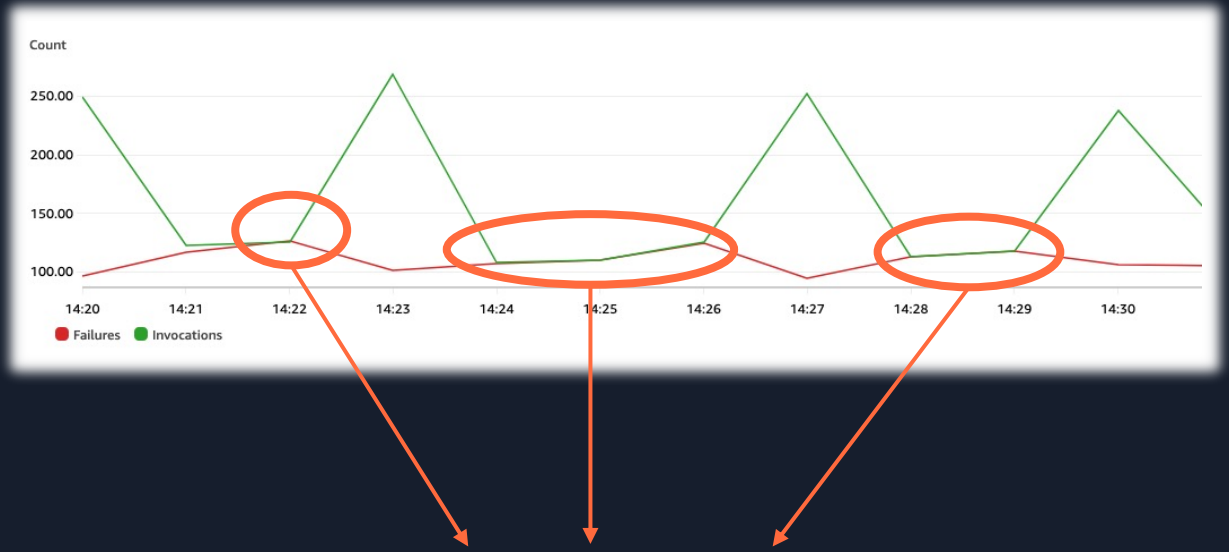
Gestione degli errori

RETRY

Riprovare automaticamente le operazioni che falliscono a causa di errori transitori migliora la resilienza dell'applicazione

Tentativi troppo frequenti possono però:

1. sovraccaricare i sistemi a valle
2. fallire nuovamente

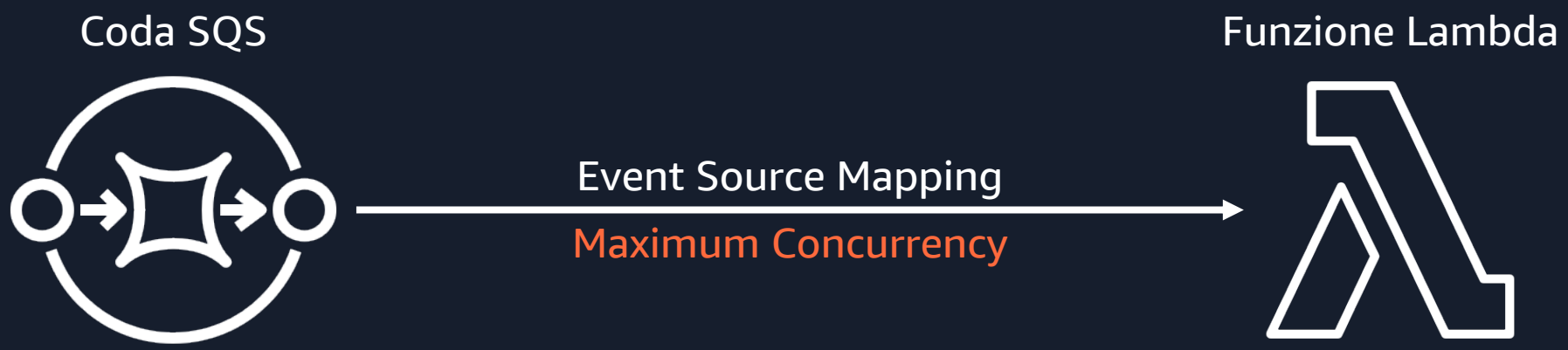


Tutte le esecuzioni sono fallite

Gestione degli errori

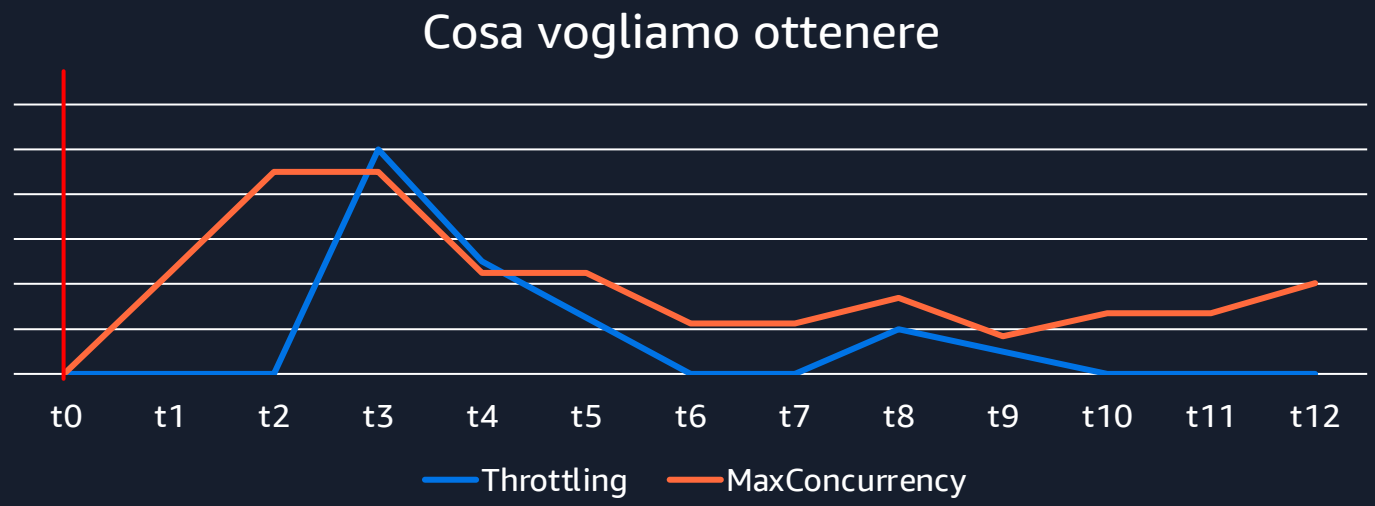
RETRY WITH BACKOFF

HTTP 429 - Too Many Requests → Slow down!



limita il numero di esecuzioni concorrenti di una funzione Lambda che una coda SQS può invocare

Adaptive scaling



Vediamo come può essere implementato…

Implementazione

Metrica Cloudwatch per monitorare il livello di throttling

CW PutMetricData

```
namespace = 'myapp'
try:
    response = self.cloudwatch.put_metric_data(
        Namespace=namespace,
        MetricData=[
            {
                'MetricName':name,
                'Value': value,
                'Unit': unit
            }
        ]
    )
    logger.info(f"Successfully published metric {name} to {namespace}")
except ClientError as e:
    logger.error(f"Error publishing metric: {e}")
    raise
```

1. Richiede una chiamata verso CloudWatch
2. Aumenta il tempo di esecuzione della funzione
3. Genera costi aggiuntivi

   (API Call a CloudWatch + tempo di esecuzione lambda)

Implementazione

Metrica Cloudwatch per monitorare il livello di throttling

CW Logs metric filter

```
logger = logging.getLogger(__name__)
logger.setLevel("INFO")

namespace = 'myapp'

def send_metric(name, value, unit):
    logger.info(f"MONITORING|{value}|{unit}|{name}|{namespace}")
    return
```

```
2024-11-06T17:19:19.224Z            START RequestId: e361f182-13f6-50ae-87cc-ae27b4a1c0d0 Version: $LATEST

START RequestId: e361f182-13f6-50ae-87cc-ae27b4a1c0d0 Version: $LATEST

2024-11-06T17:19:19.505Z            [WARNING] 2024-11-06T17:19:19.505Z e361f182-13f6-50ae-87cc-ae27b4a1c0d0 Messages failed to send to target Lambda function: Too Many Requests

[WARNING]       2024-11-06T17:19:19.505Z        e361f182-13f6-50ae-87cc-ae27b4a1c0d0    Messages failed to send to target Lambda function: Too Many Requests

2024-11-06T17:19:19.505Z            [INFO] 2024-11-06T17:19:19.505Z e361f182-13f6-50ae-87cc-ae27b4a1c0d0 MONITORING|1|Count|ThrottledRequests|myapp

[INFO]  2024-11-06T17:19:19.505Z        e361f182-13f6-50ae-87cc-ae27b4a1c0d0    MONITORING|1|Count|ThrottledRequests|myapp
```

ThrottledRequests

Filter pattern
"MONITORING|1|Count|ThrottledRequests|myapp"

Metric
myapp / ThrottledRequests

Metric value
1

Default value
-

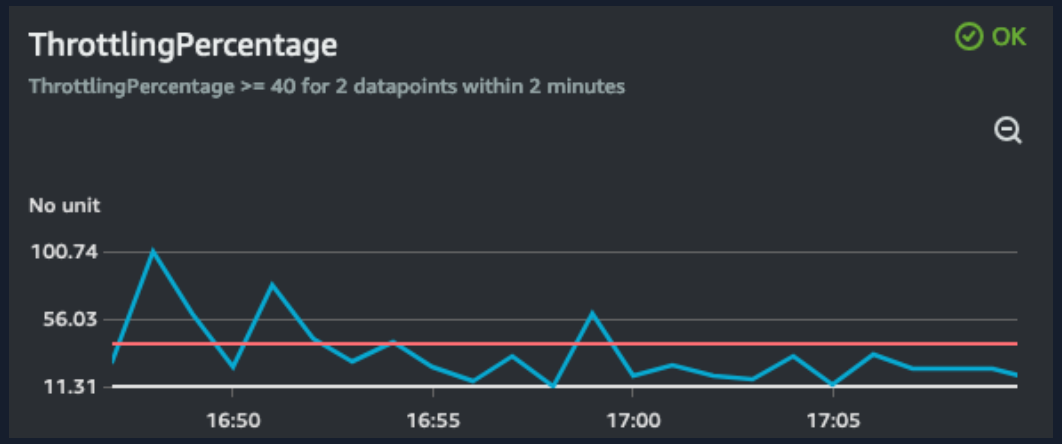
Unit
Count

Dimensions
-

Alarms
None.

Implementazione

Allarme su percentuale di throttling vs esecuzioni totali



Details

Name
ThrottleBreached

Type
Metric alarm

Description
No description

Label - optional
ThrottlingPercentage

Math expression
100*(m3/m1)

Metrics
m3 | myapp | ThrottledRequests • Statistic: Sum
m1 | AWS/Lambda | Invocations | FunctionName : source-function • Statistic: Sum

Period
1 minute

State
OK

Threshold
ThrottlingPercentage >= 40 for 2 datapoints within 2 minutes

Last state update
2024-11-06 16:54:59 (UTC)

Actions
No actions

Datapoints to alarm
2 out of 2

Missing data treatment
Treat missing data as good (not breaching threshold)

Percentiles with low samples
evaluate

ARN
arn:aws:cloudwatch:eu-[redacted]:alarm:ThrottleBreached

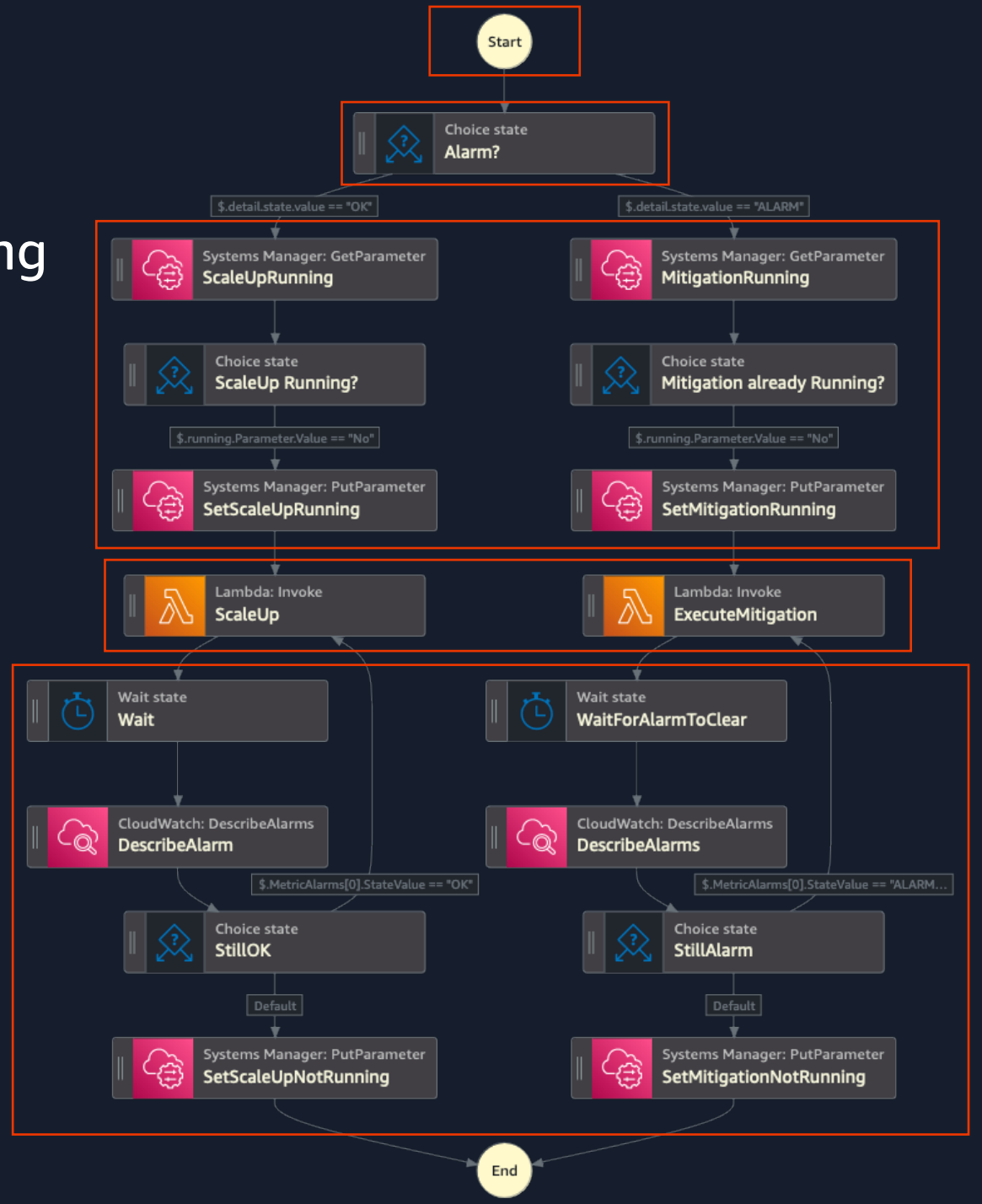
Implementazione

Step function per orchestrare lo scaling

Invocata da una regola di EventBridge
sul cambiamento di stato del CloudWatch Alarm

Switch sullo stato dell'allarme:
- In allarme -> avvia scale down
- OK -> avvia scale up

Verifica se l'azione è già in corso,
continua solo in caso negativo



Implementazione

Step function per orchestrare lo scaling

Esegue l'operazione di scale up o scale down

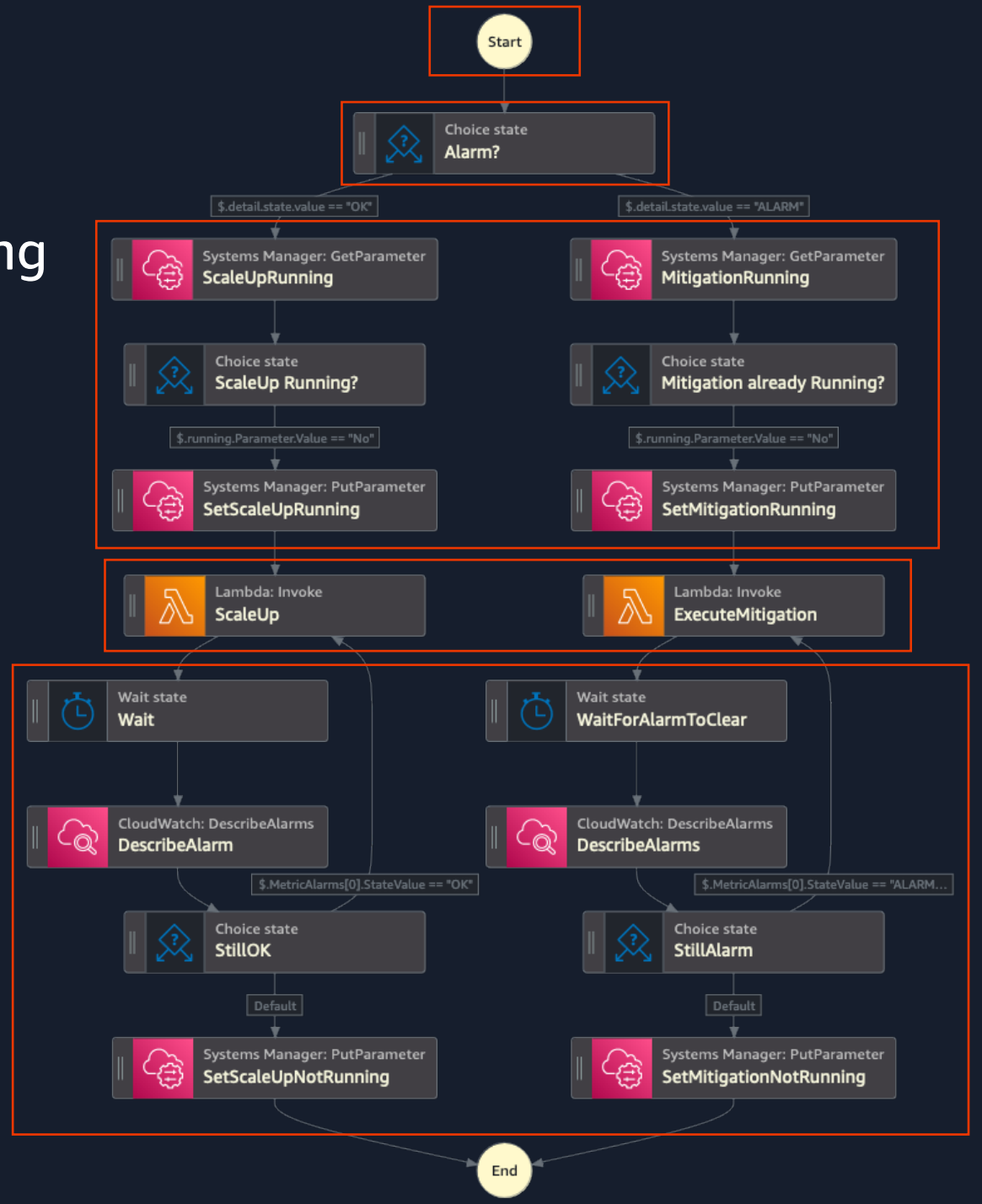
Scale down:

```
# Set new max concurrency to half

concurrency = int(max_concurrency/2) if int(max_concurrency/2) >=2 else 2

response = lambda_client.update_event_source_mapping(
    UUID=event_source_uuid,
    ScalingConfig={'MaximumConcurrency': concurrency })
```

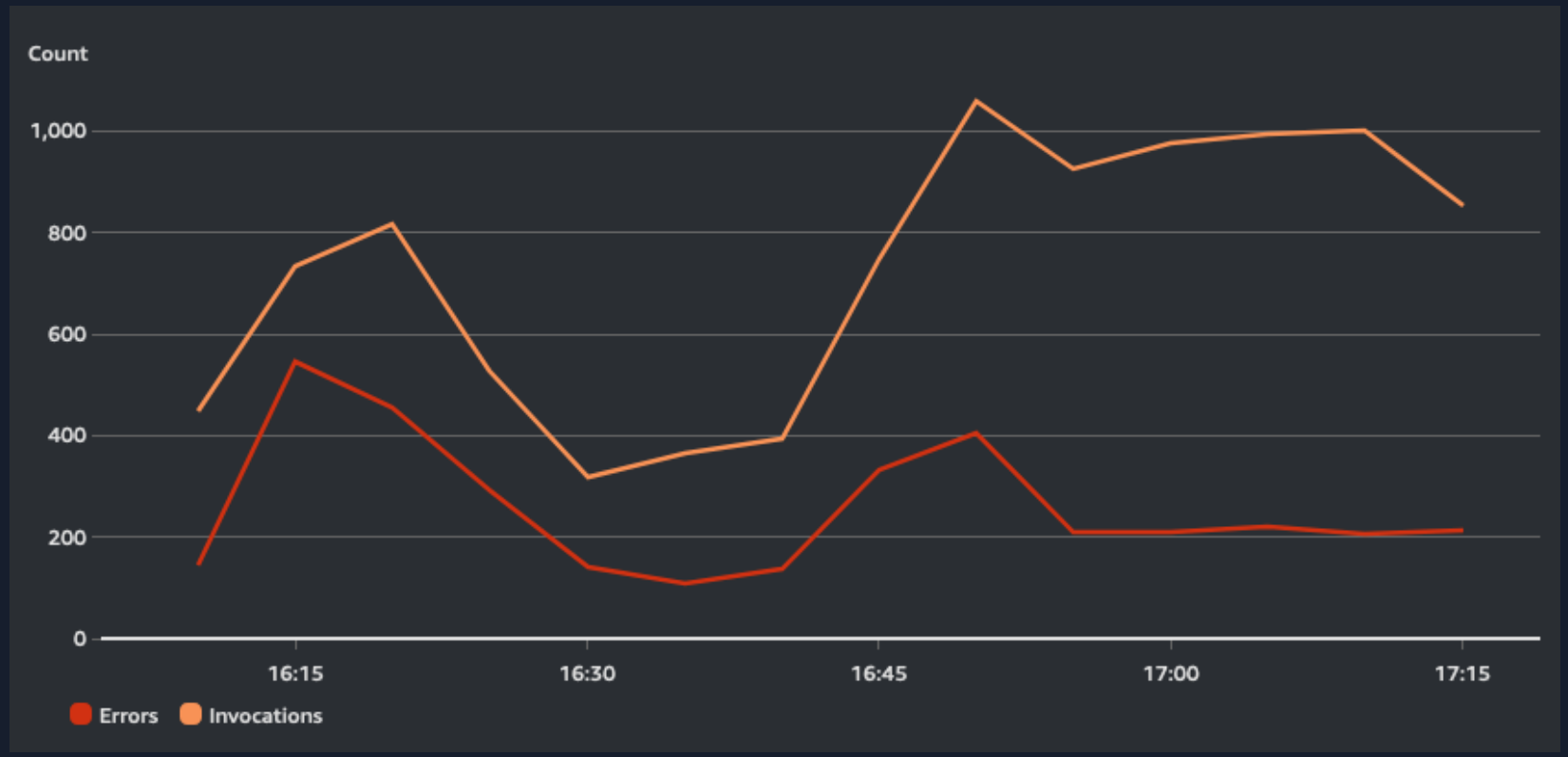
Scale up:

```
# Set new max concurrency to 50% more

concurrency = max(int(max_concurrency*1.5), 2)
concurrency = concurrency if concurrency < max_concurrency_limit else max_concurrency_limit

response = lambda_client.update_event_source_mapping(
    UUID=event_source_uuid,
    ScalingConfig={'MaximumConcurrency': concurrency })
```

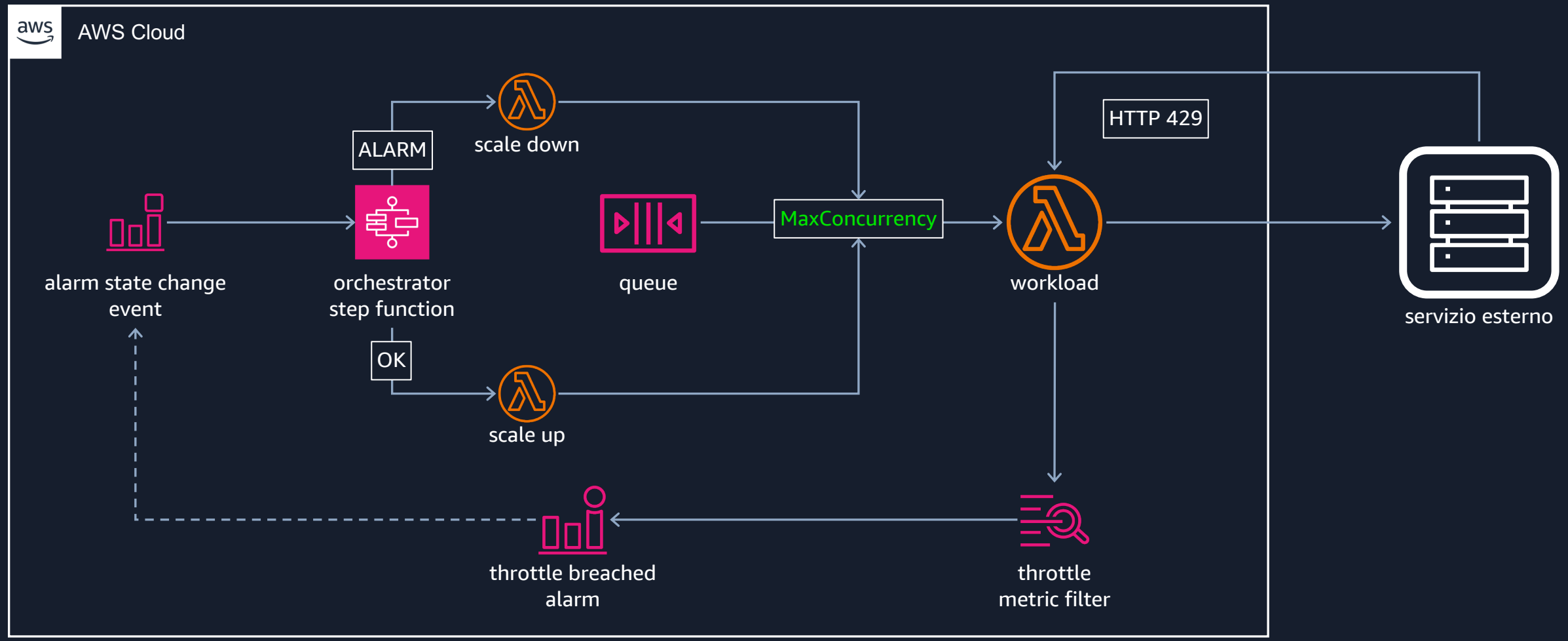
- aspetta per un determinato tempo
- verifica lo stato dell'allarme
- ripete l'azione se lo stato non è cambiato
- esce altrimenti



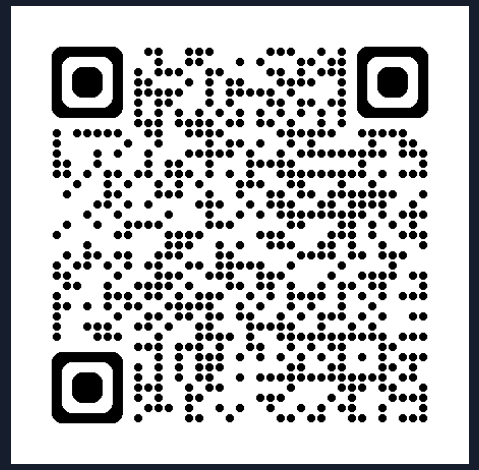
Risultato



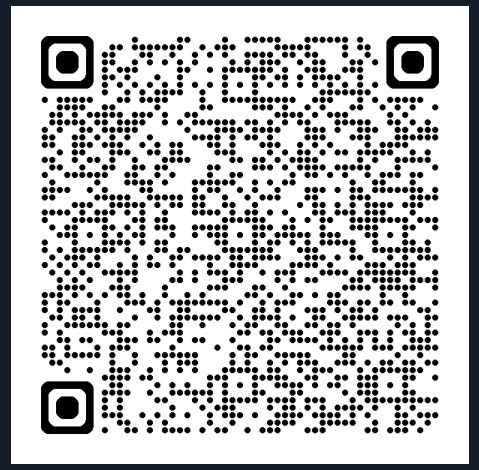
Recap



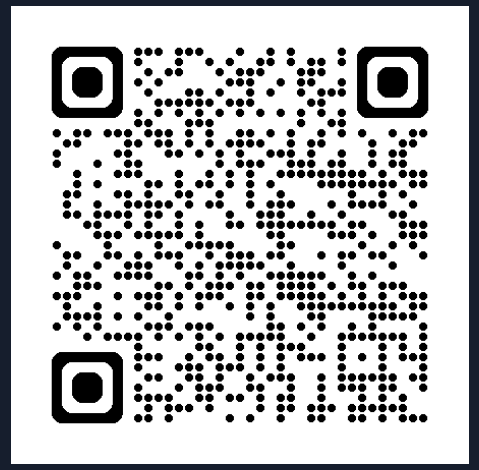
Link utili



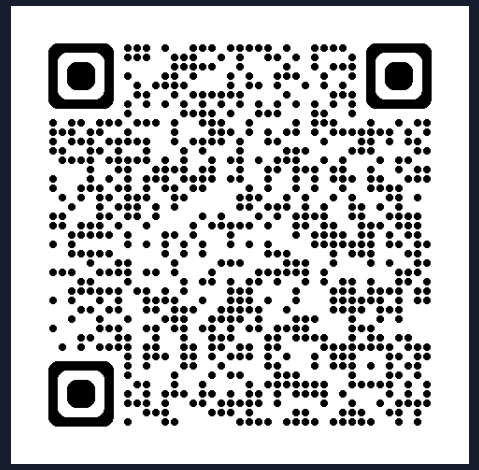
SQS Event Source Mapping
Scaling behaviour



Partial batch responses
best practices



Error Handling for
SQS event source in Lambda



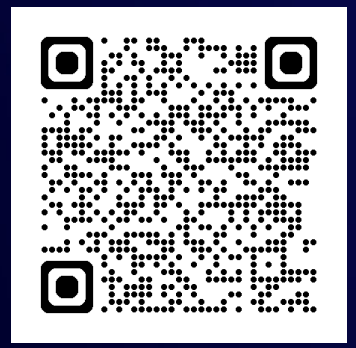
Retry with backoff
pattern

Thank you!

Gabriele Postorino

Principal Technical Account Manager

Amazon Web Services



https://www.linkedin.com/in/gabrielepostorino/