Amazon DynamoDB

Deep dive into NoSQL and serverless scaling

Shwetang Oza, "Oza" (he/him)
Sr. Database Specialist SA
Amazon Web Services

Robert McCauley (he/him)
Sr. Database Specialist SA
Amazon Web Services

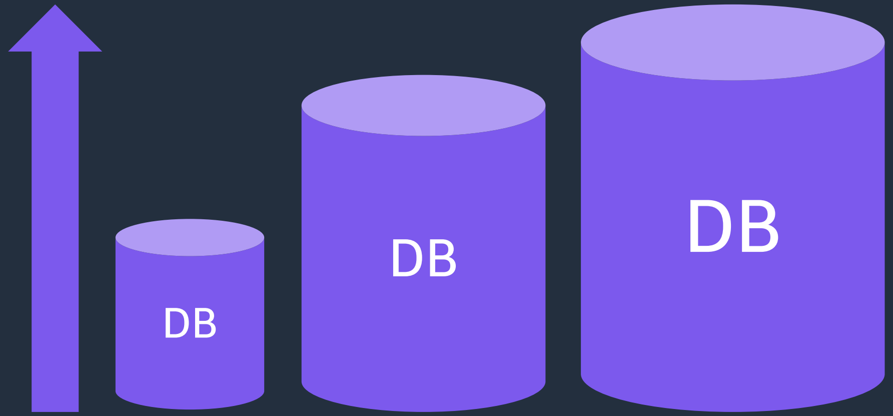
Agenda

- The NoSQL way
- DynamoDB serverless database
- Serverless adaptive capacity
- Global tables and storage
- Demo
- Q&A

The NoSQL way

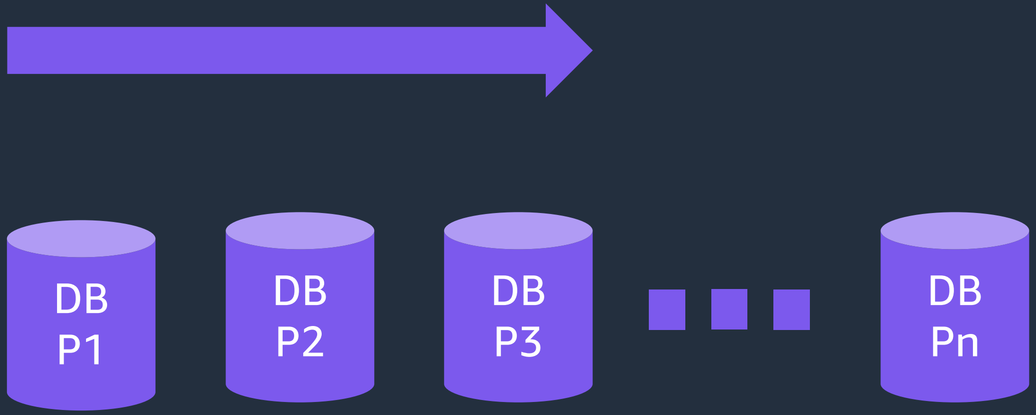
Scaling databases

Traditional SQL



Scale up

NoSQL



Scale out to many shards

There is a way to design data that’s horizontally scalable

SQL and NoSQL side-by-side

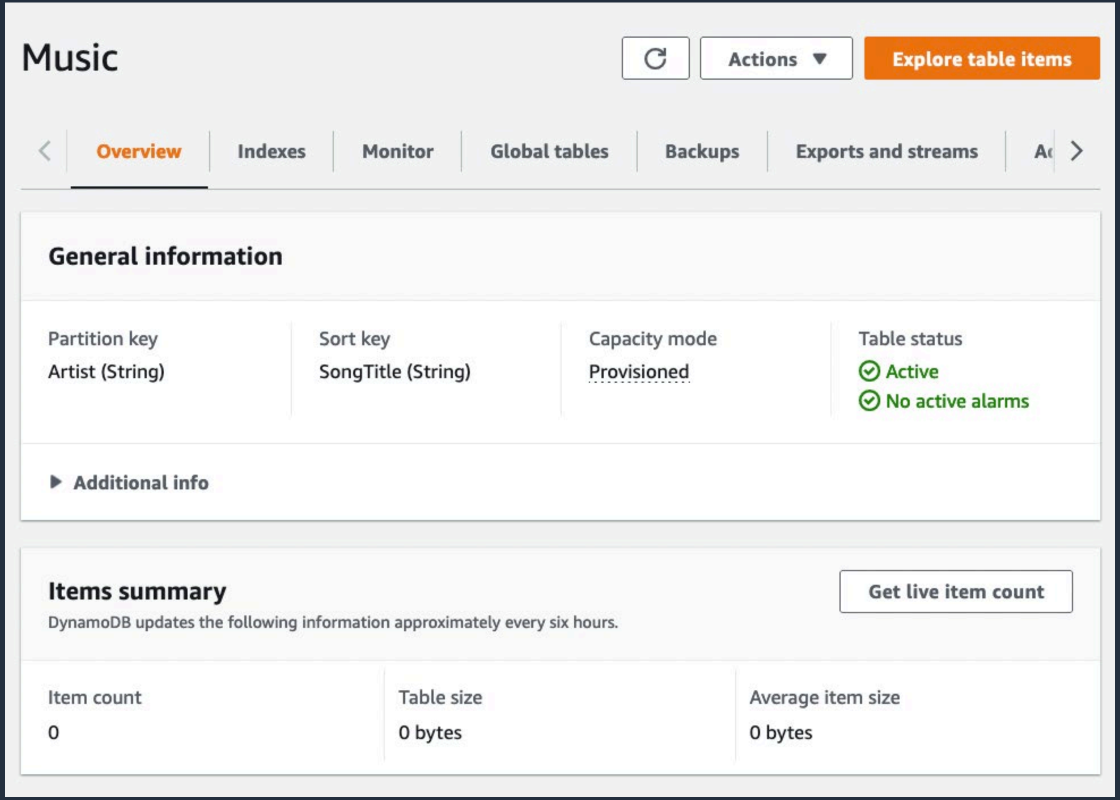
SQL	NoSQL
Optimized for storage	**Optimized for compute**
Normalized and relational	Denormalized and hierarchical
Undefined access patterns	Instantiated views
Scale vertically	Scale horizontally
Good for OLAP	Built for OLTP at scale

DynamoDB serverless database

DynamoDB

And the DBA or developer said, let there be a database called Music….

```
aws dynamodb create-table\
 --table-name Music \
--attribute-definitions \
AttributeName=Artist,AttributeType=S \
AttributeName=SongTitle,AttributeType=S \
 --key-schema \
AttributeName=Artist,KeyType=HASH \
AttributeName=SongTitle,KeyType=RANGE \
--provisioned-throughput \
ReadCapacityUnits=5,WriteCapacityUnits=5 \
--table-class STANDARD
```



General information

Partition key	Sort key	Capacity mode	Table status
Artist (String)	SongTitle (String)	Provisioned	Active No active alarms

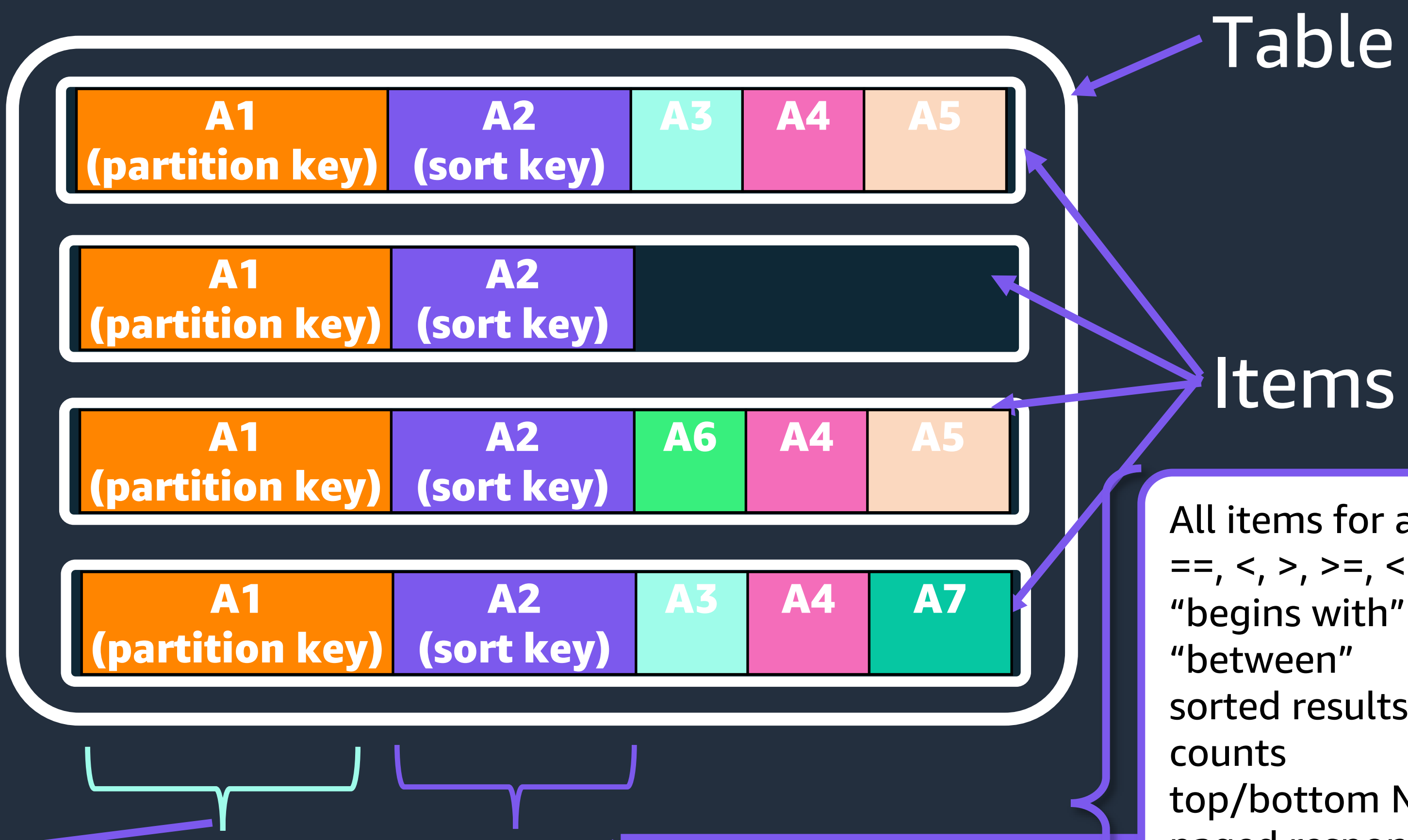
▶ Additional info

Items summary

DynamoDB updates the following information approximately every six hours.

Item count	Table size	Average item size
0	0 bytes	0 bytes

DynamoDB table

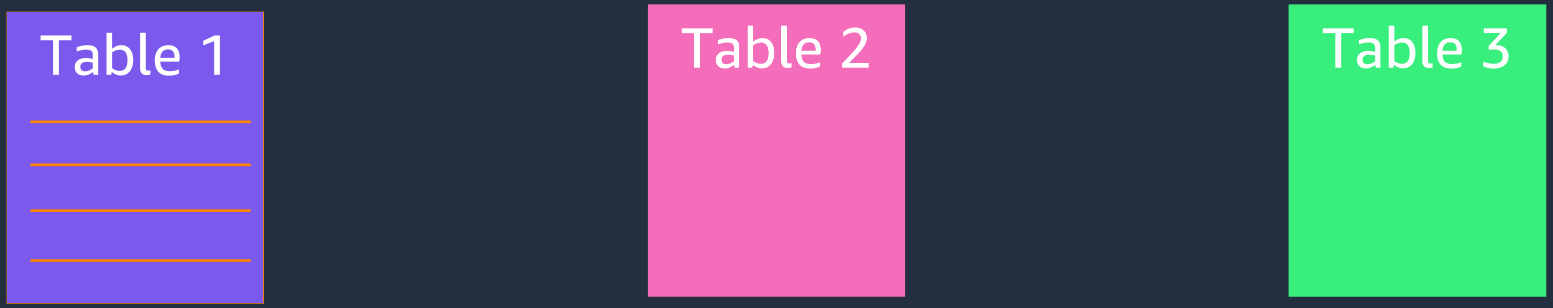


Mandatory
Key-value access pattern
Determines data distribution

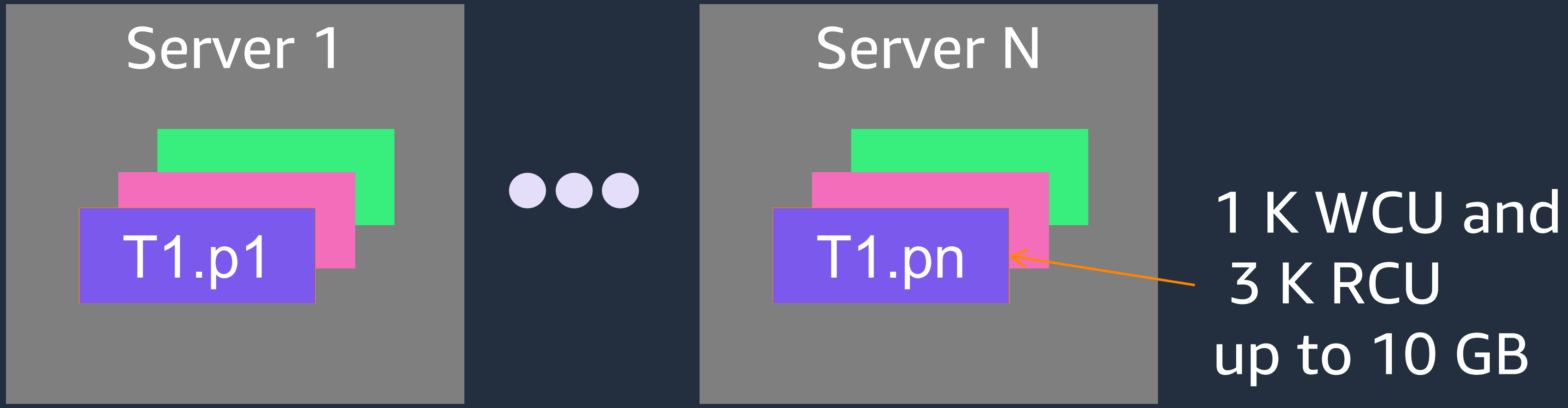
Optional
Model 1:N relationships
Enables rich query capabilities

All items for a partition key
==, <, >, >=, <=
“begins with”
“between”
sorted results
counts
top/bottom N values
paged responses

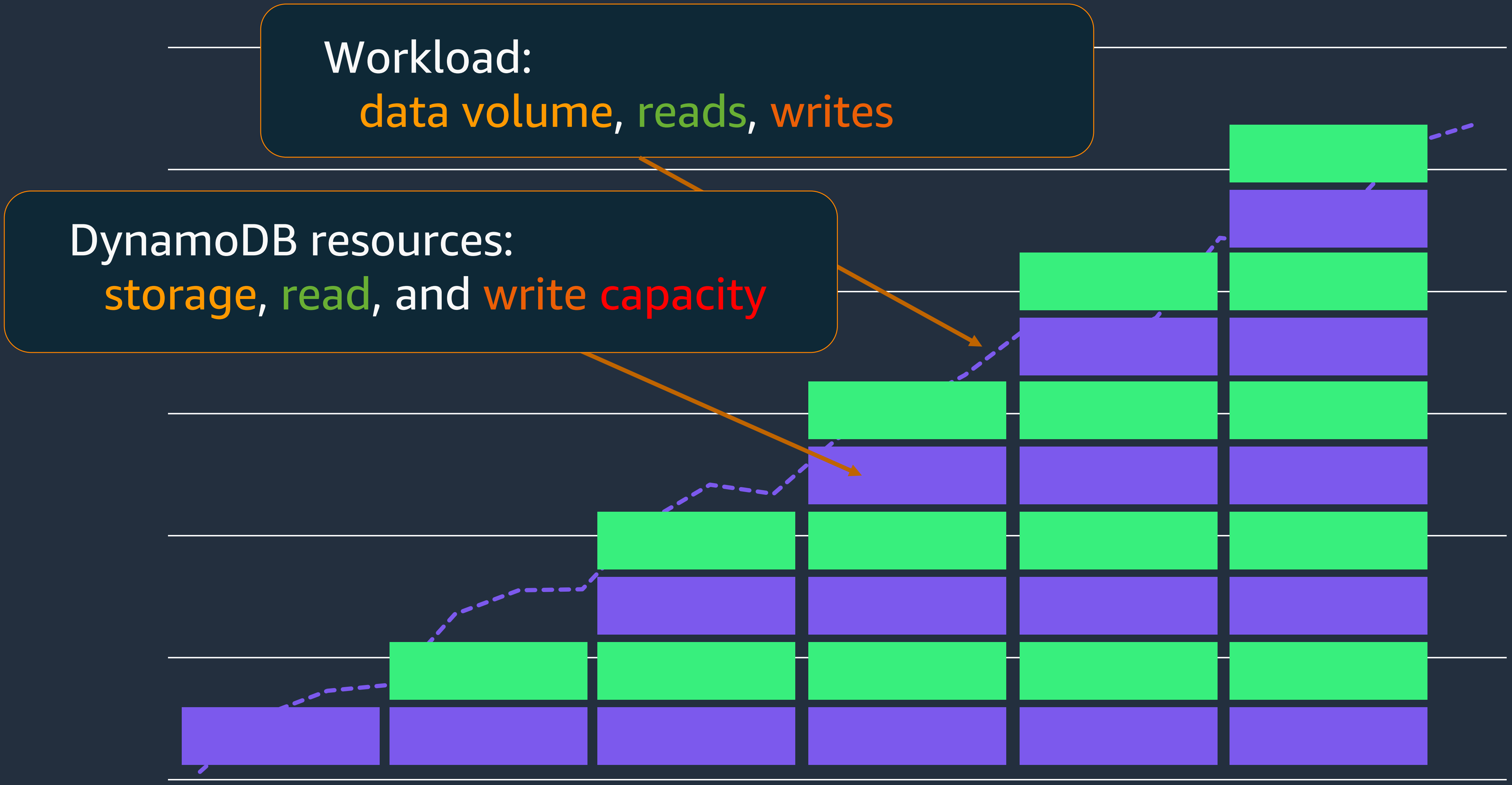
You work with tables…



DynamoDB does the rest under the hood…

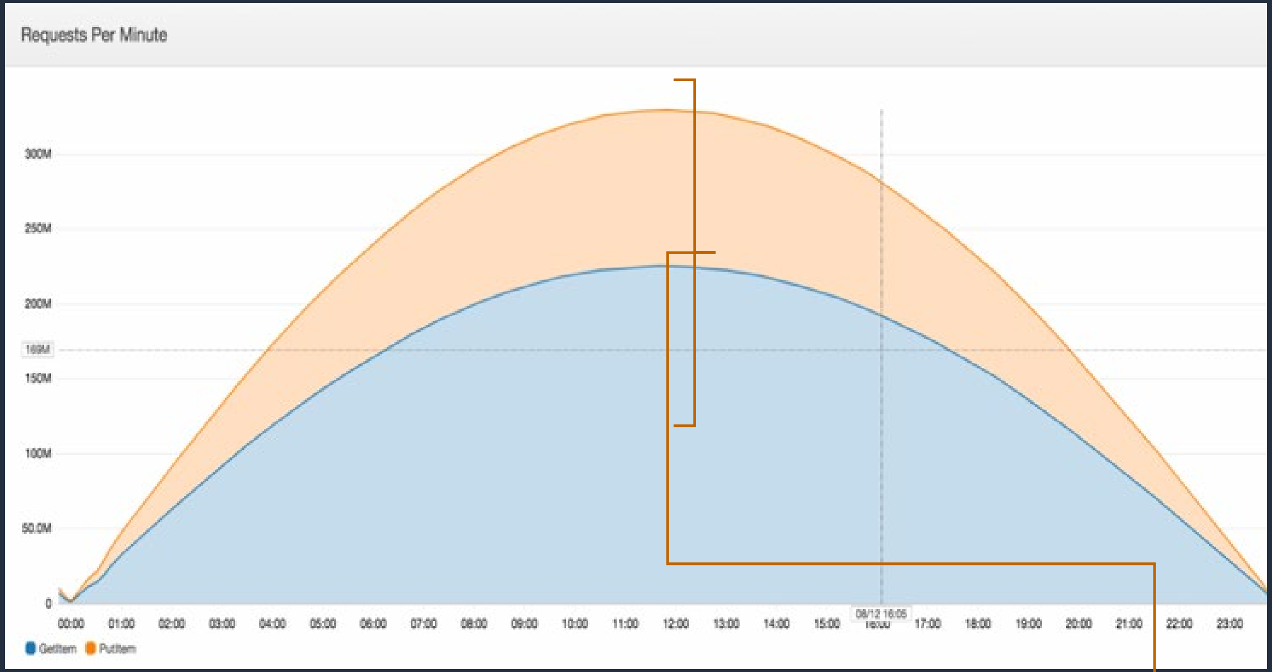


Horizontal scaling with DynamoDB



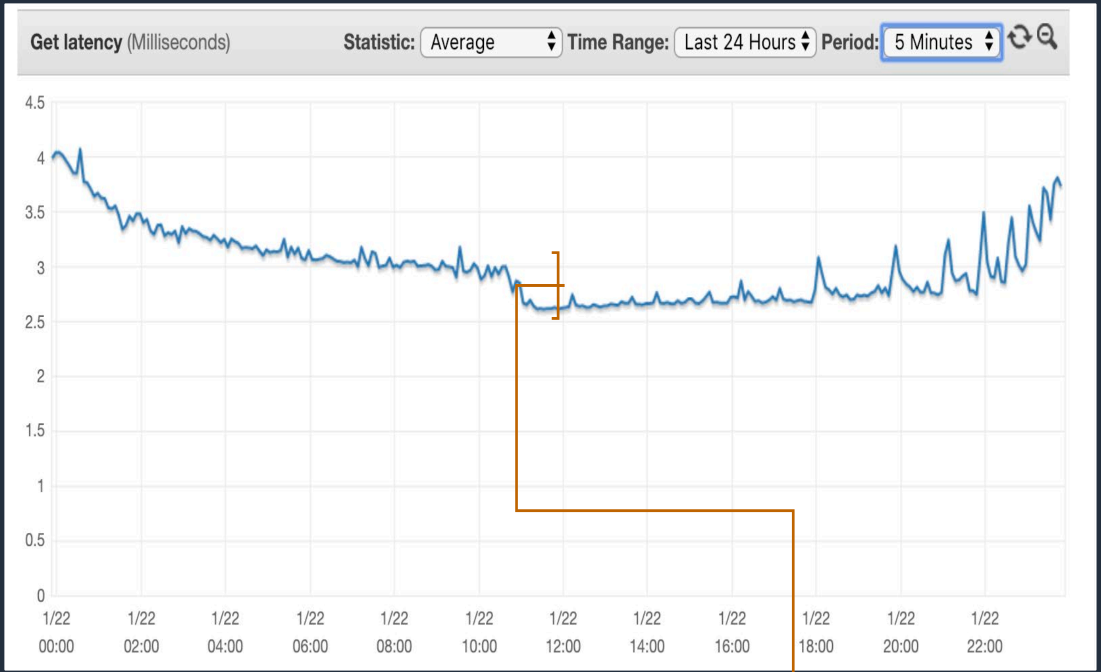
Performance at any scale

High request volume



Many **millions of requests** per second per table

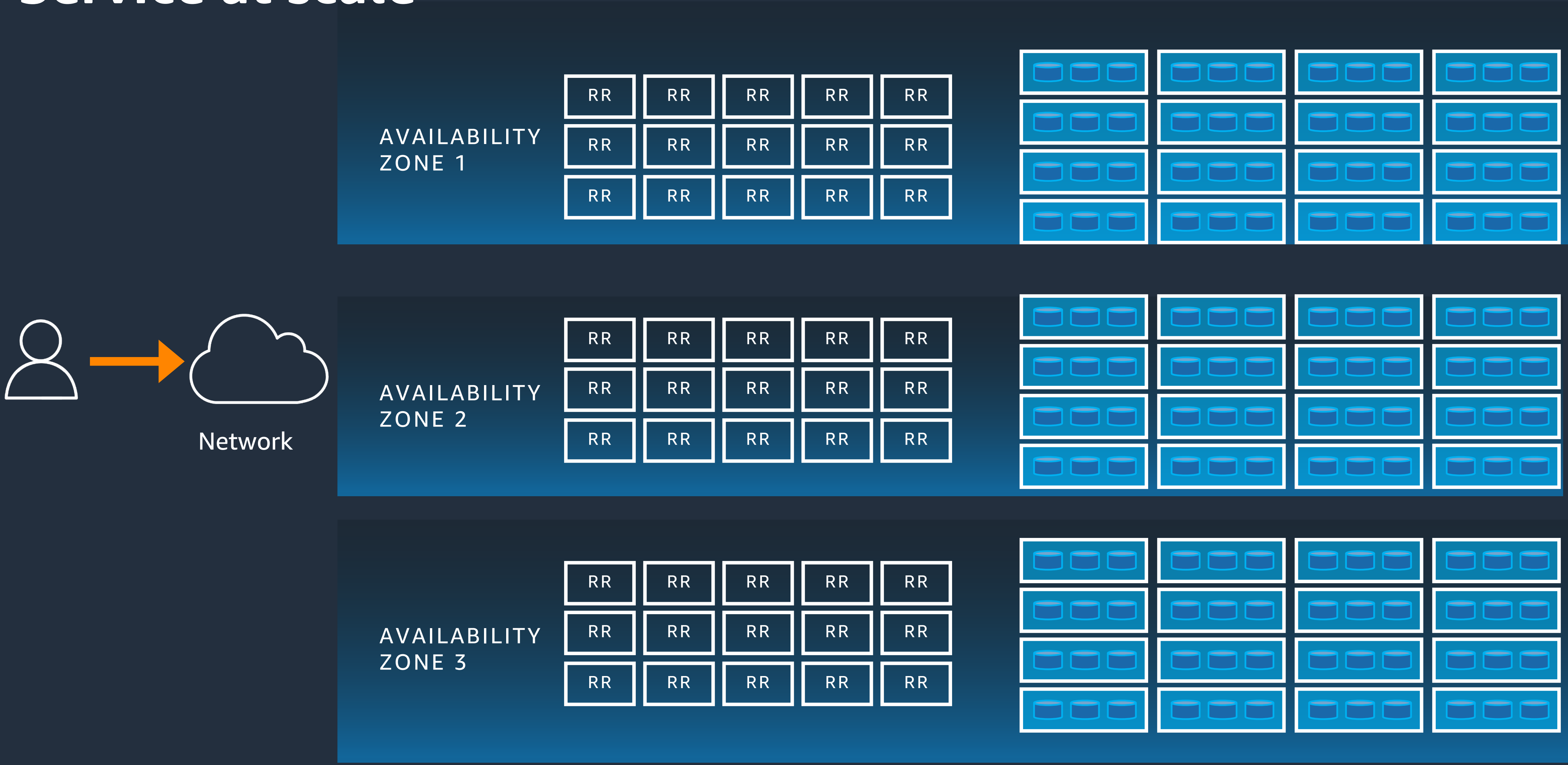
Consistent low latency



Statistic: Average Time Range: Last 24 Hours Period: 5 Minutes

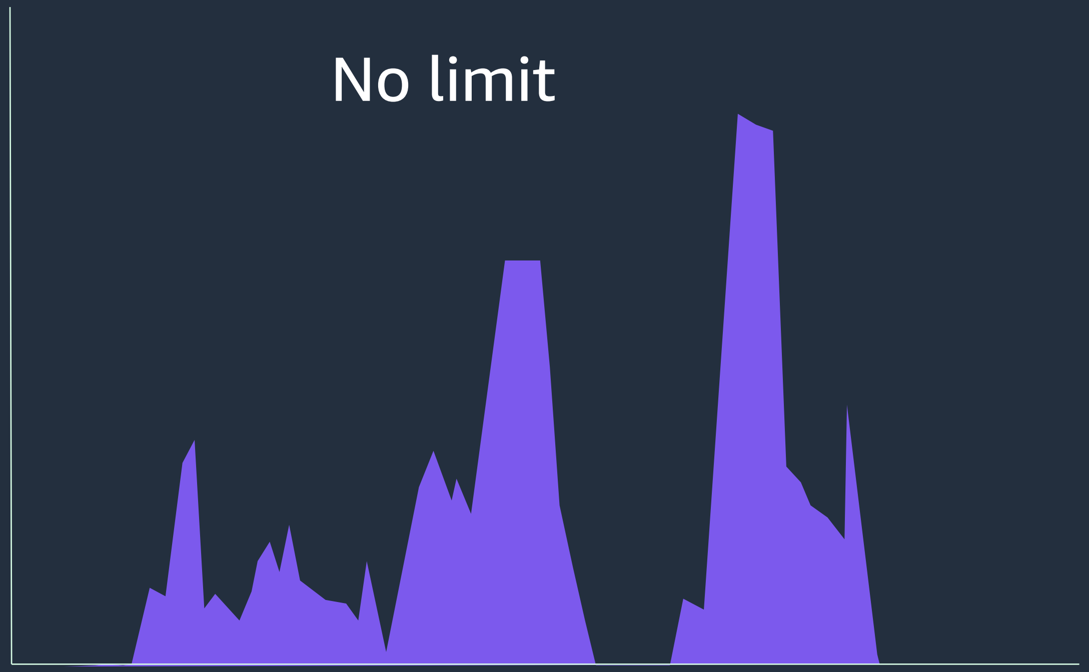
Millisecond variance

Service at scale



On-demand capacity

On-demand capacity mode



Start at zero

Features

- No capacity planning, provisioning, or reservations
- Pay only for the reads and writes you perform

Key benefits

- Eliminates tradeoffs of overprovisioning or underprovisioning
- Instantly accommodates your workload as traffic ramps up and down

On-demand scaling properties

Base throughput

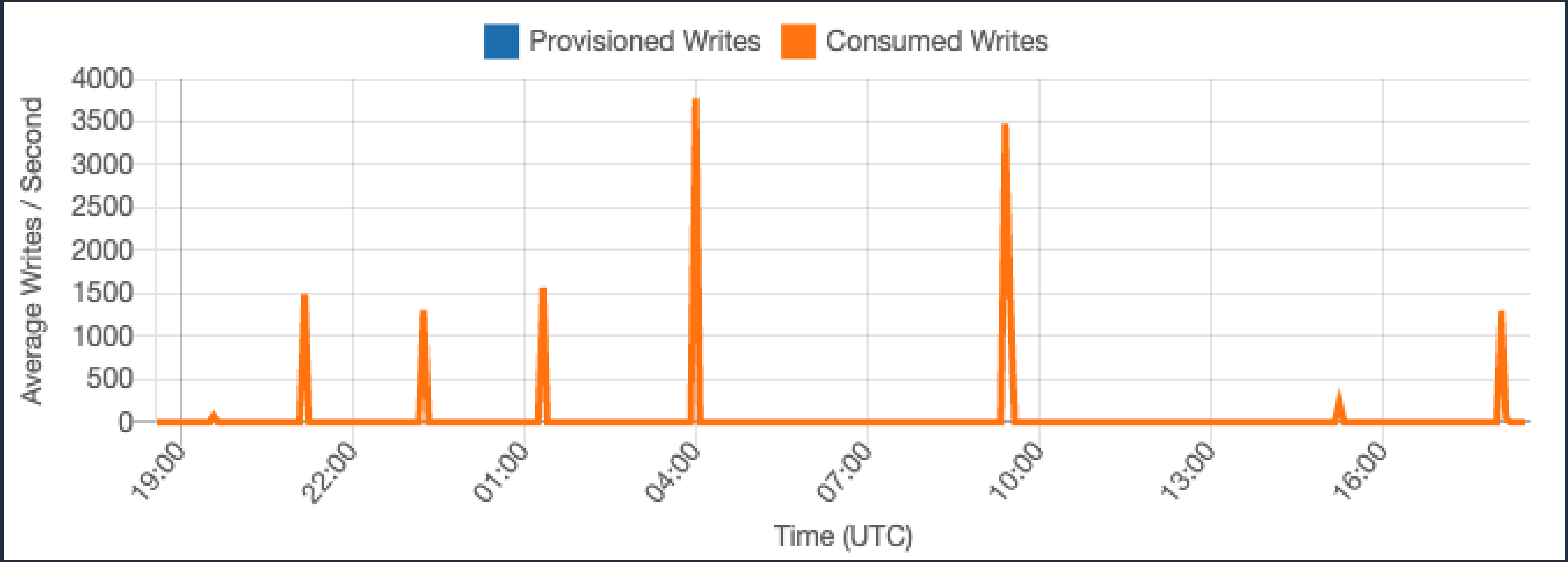
- Up to 4,000 WCU per second
- Up to 24,000 default RCU per second
- Any linear combination of the two

Maximum throughput

- Unlimited!

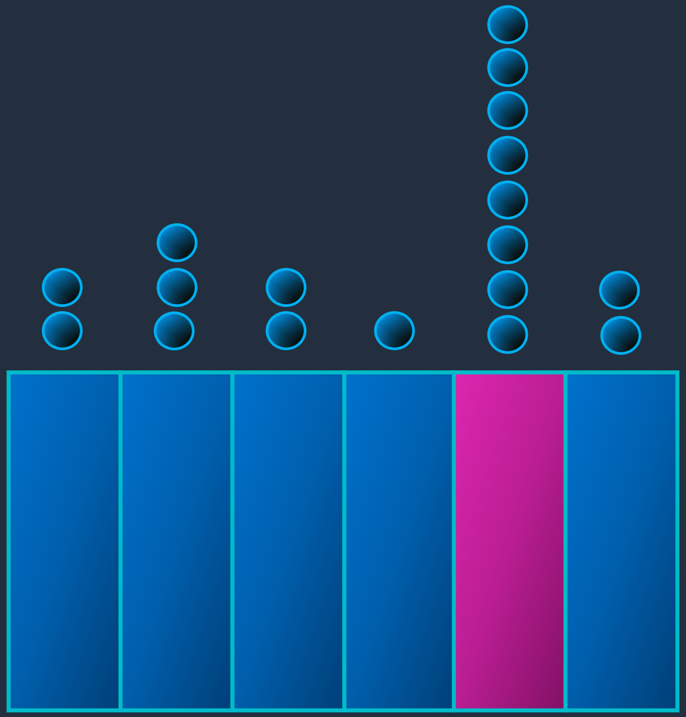
Pay per request: Use nothing, pay nothing

On-demand workload

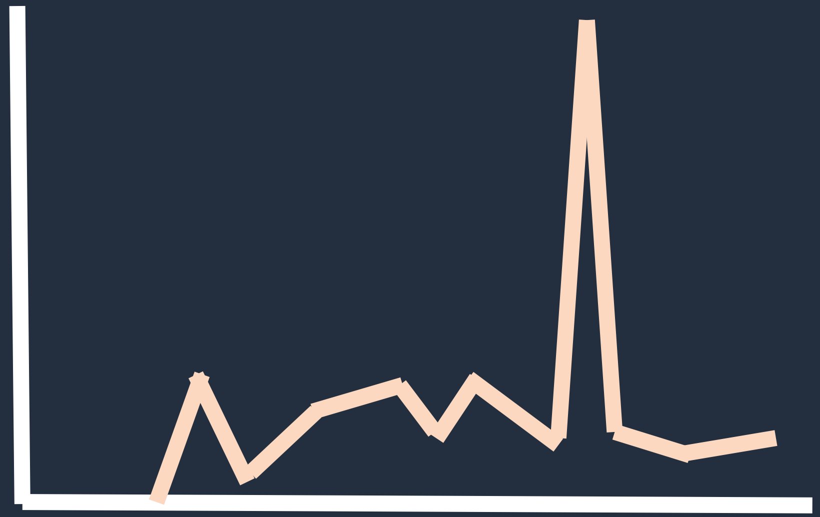


Serverless adaptive capacity

The uneven access problem common to NoSQL



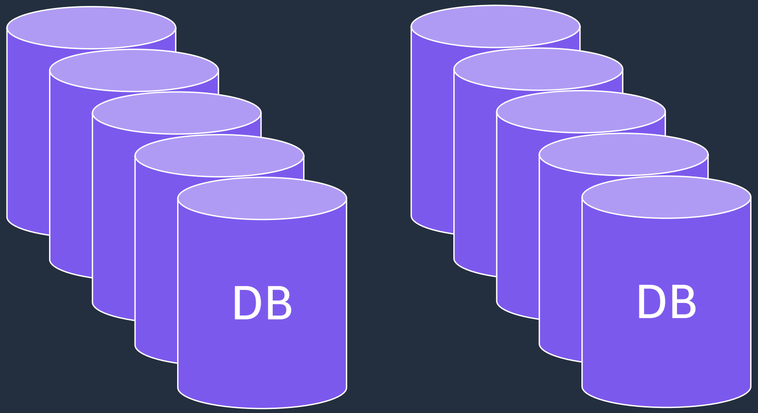
Across data



and time

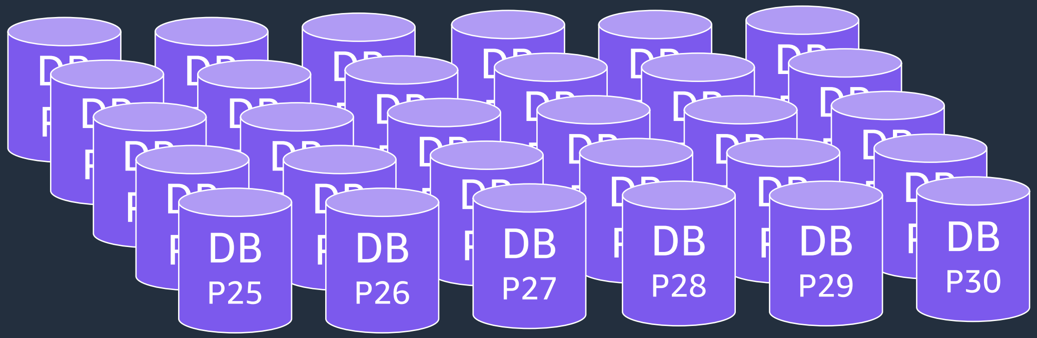
Scaling NoSQL databases

Most NoSQL databases



Servers and clusters

DynamoDB



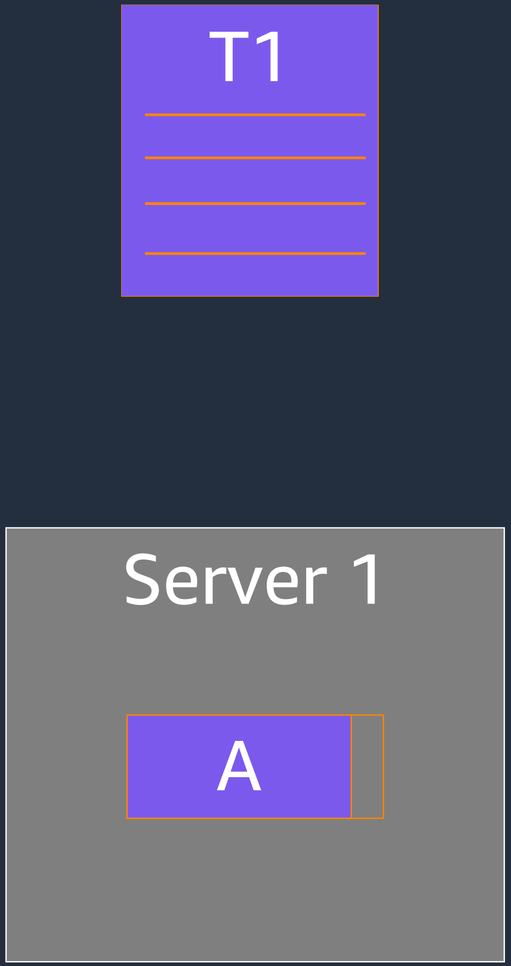
DynamoDB: partitions

Basic premise: There is a way to shard data that’s horizontally scalable

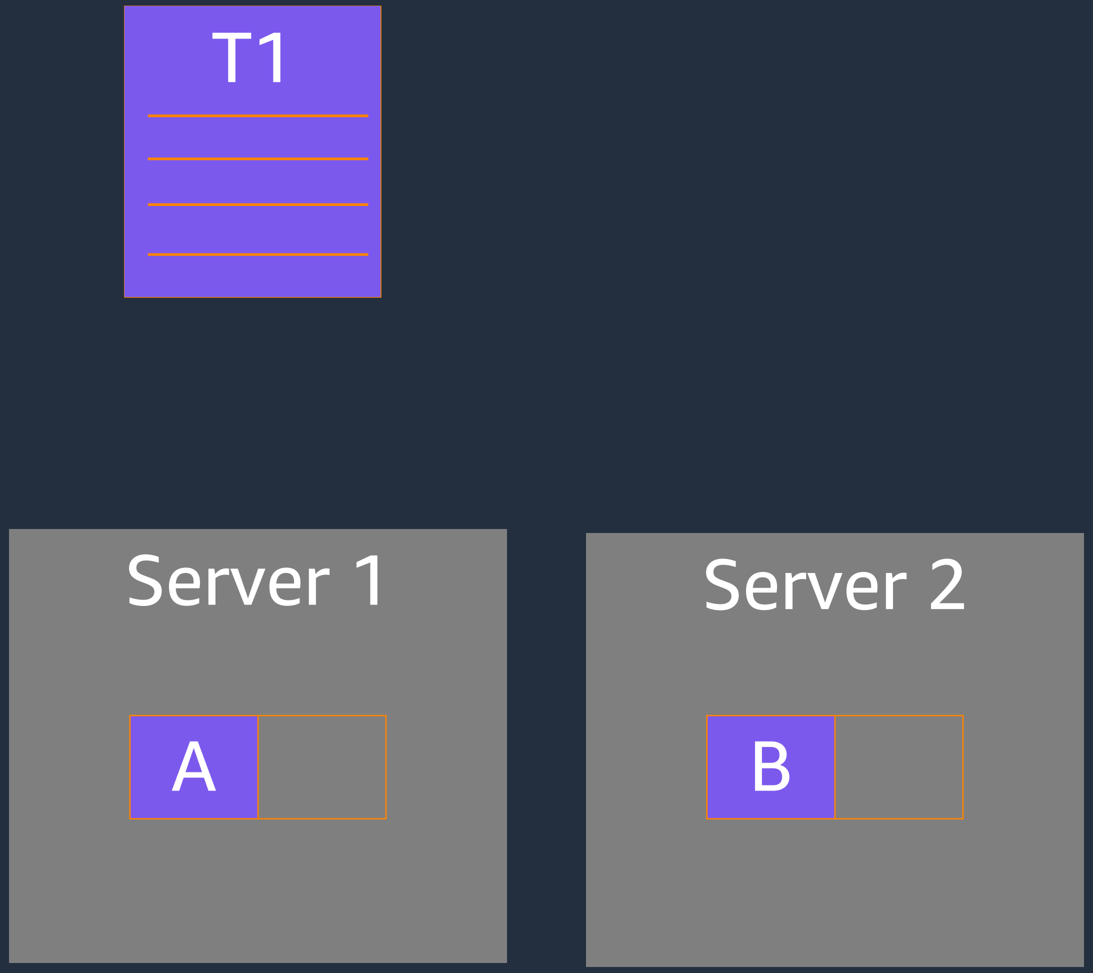
Adaptive capacity – Core functions

- Dynamic partitioning
- High-traffic item isolation
- Throughput boosting

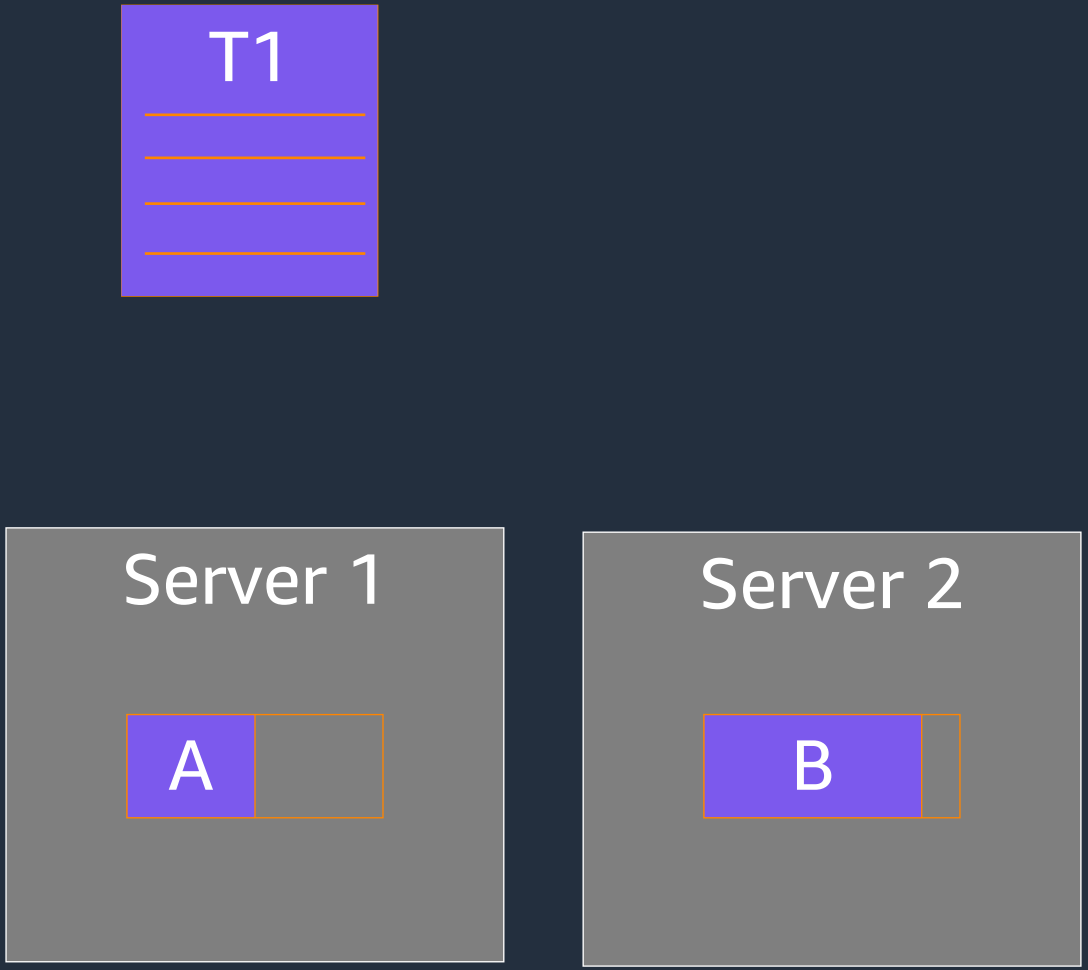
Dynamic partitioning – Storage growth



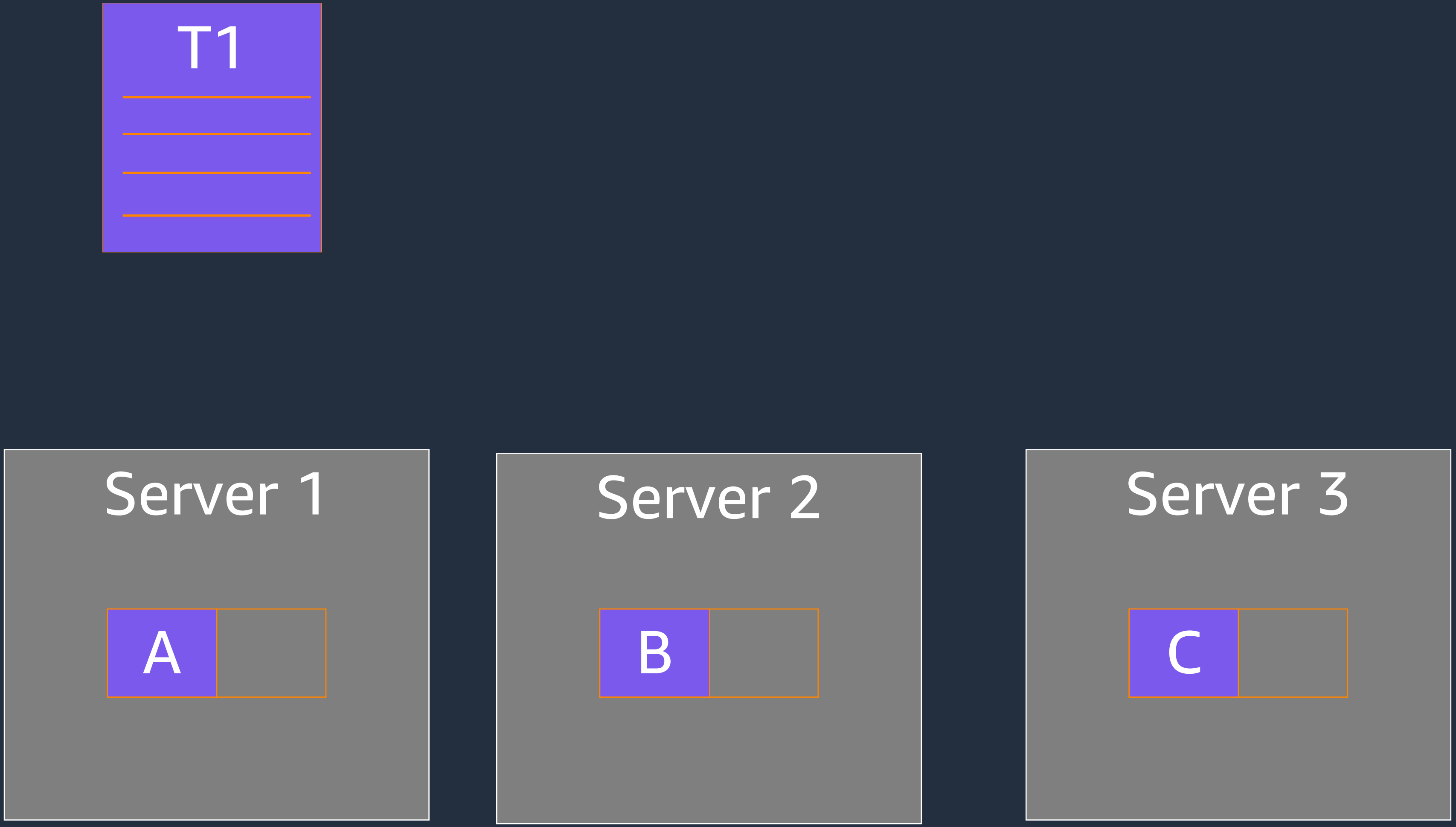
Dynamic partitioning – Storage growth



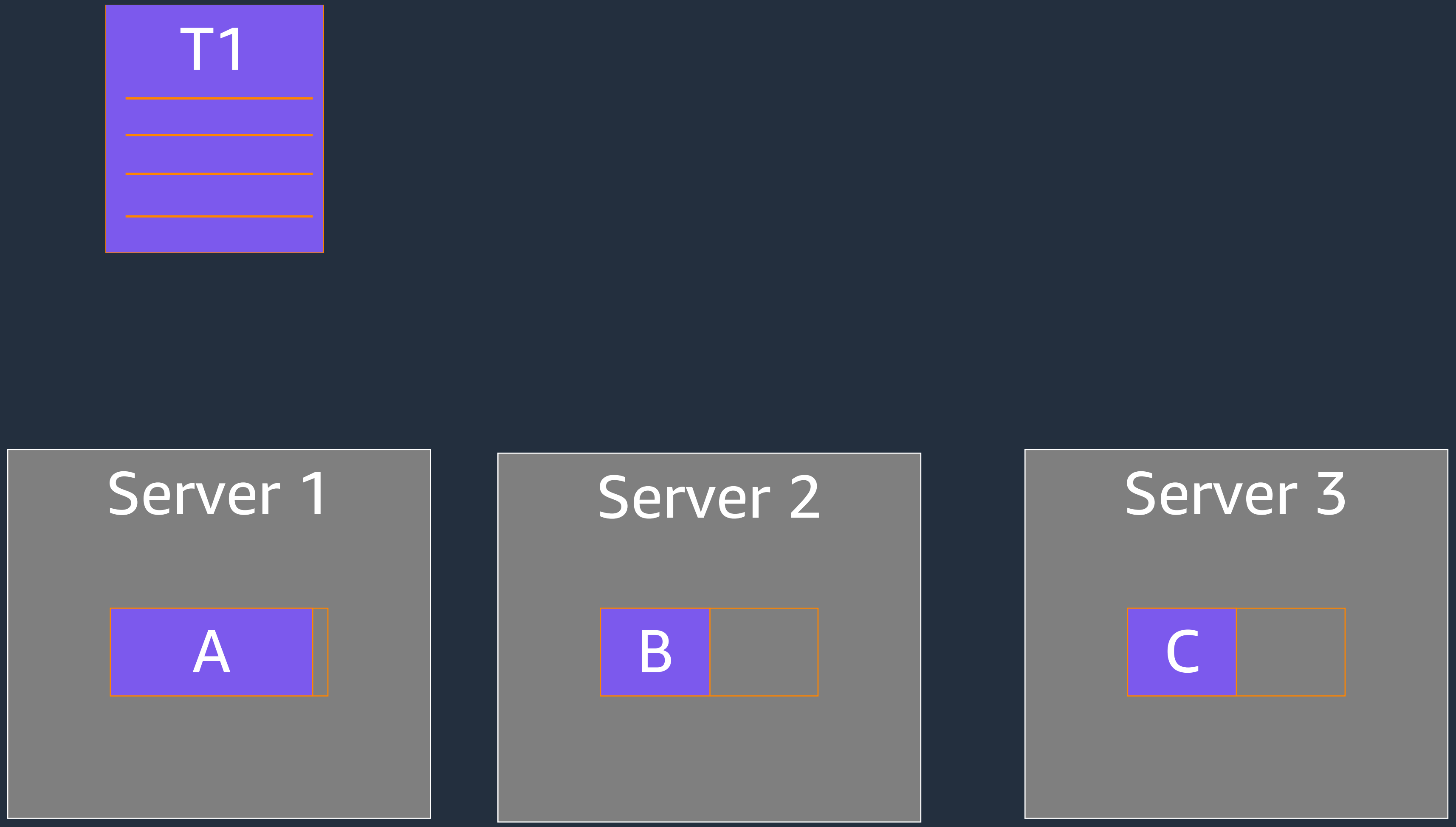
Dynamic partitioning – Storage growth



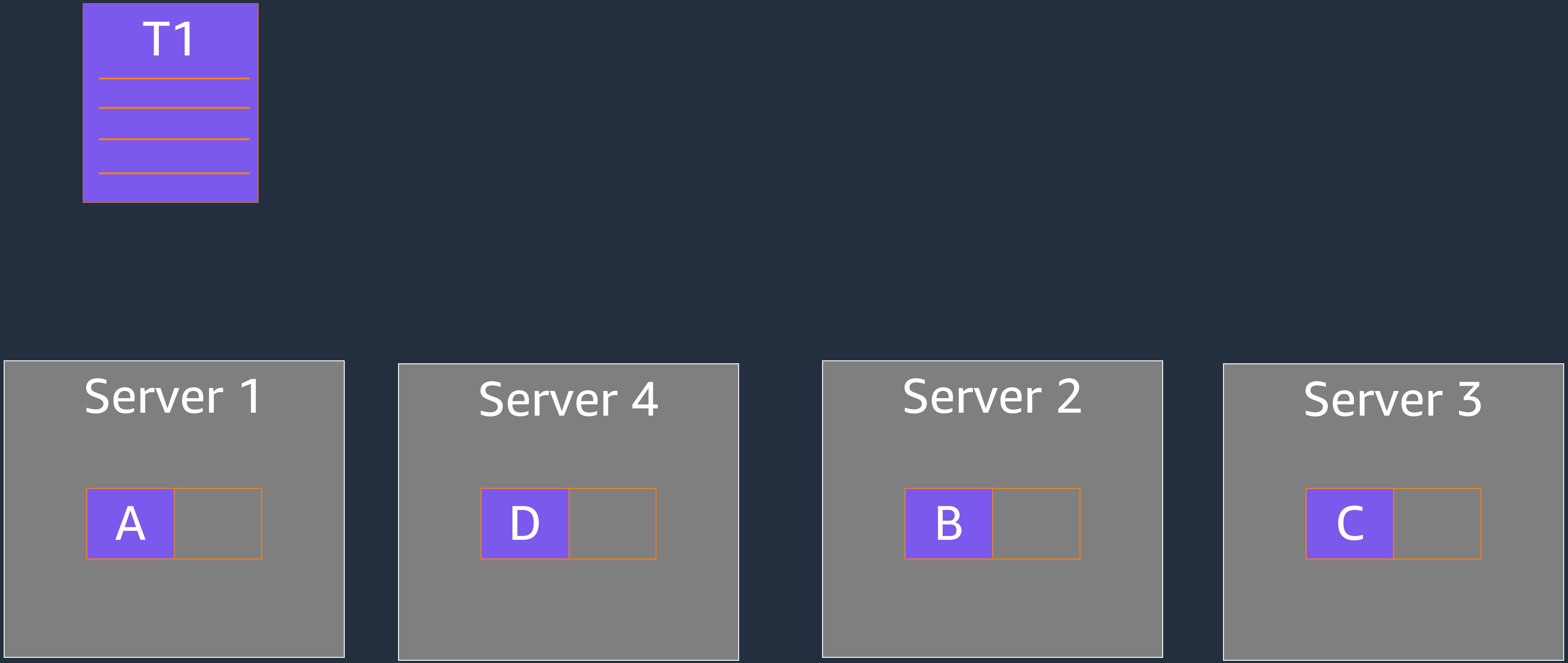
Dynamic partitioning – Storage growth



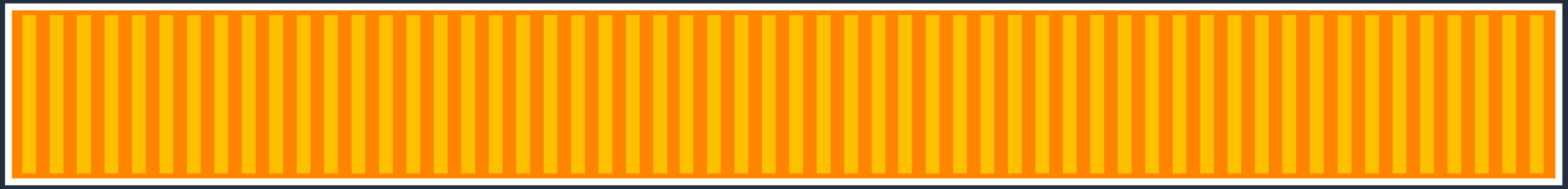
Dynamic partitioning – Storage growth



Dynamic partitioning – Storage growth

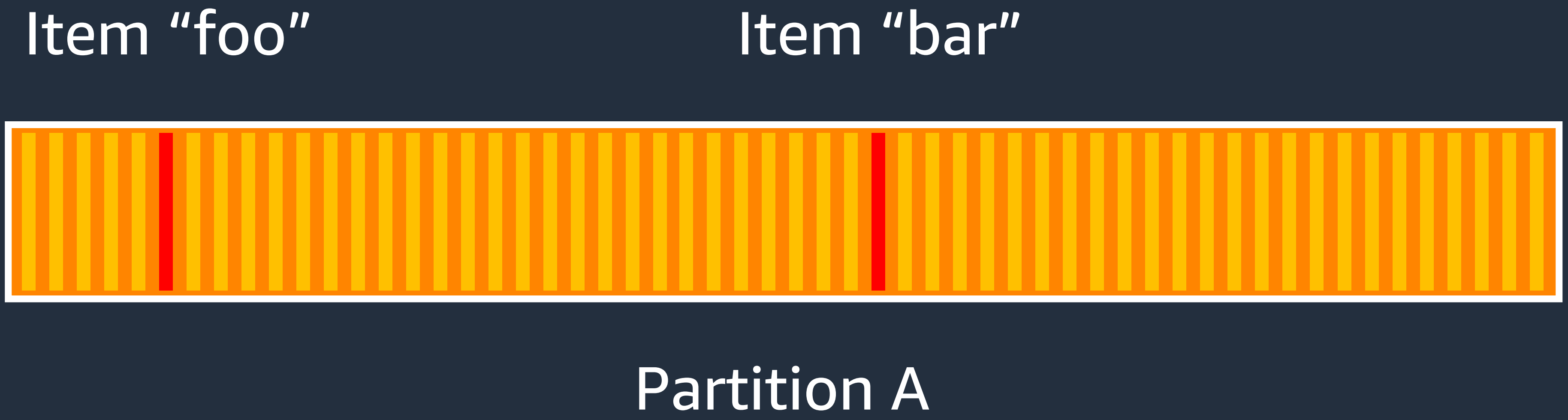


High-traffic item isolation



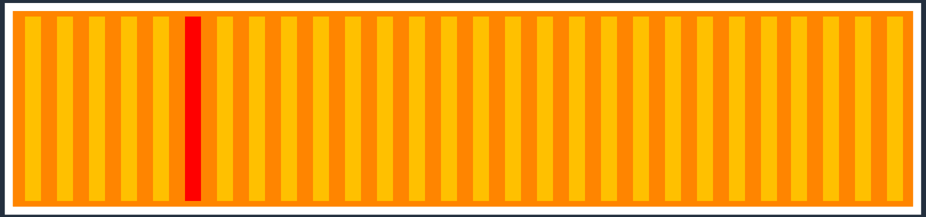
Partition A

High-traffic item isolation



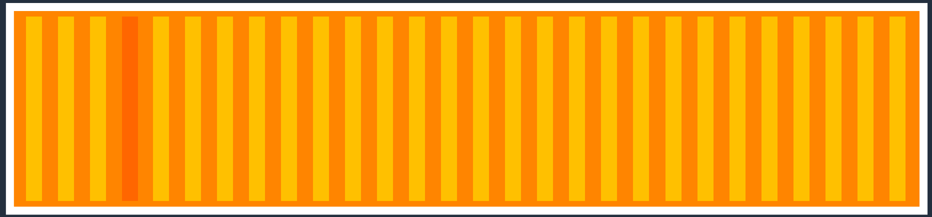
High-traffic item isolation

Item “foo”



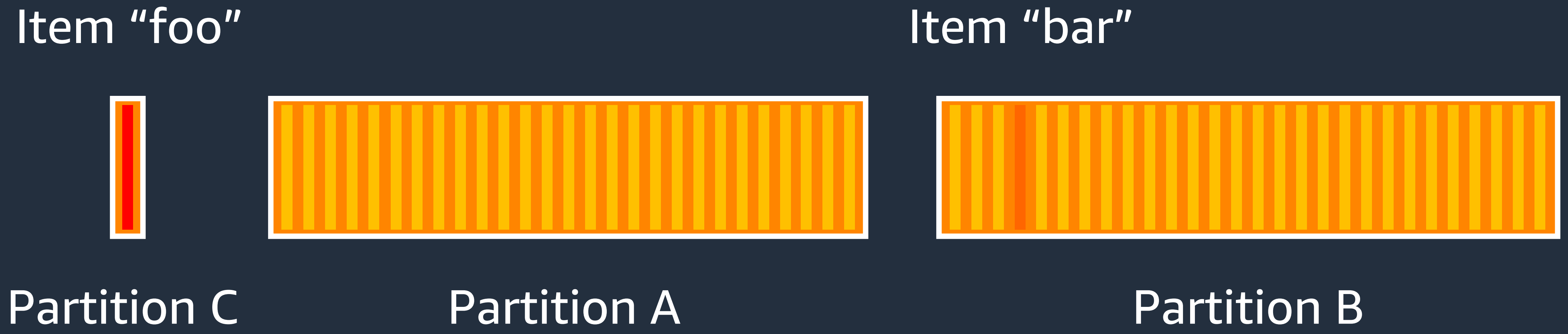
Partition A

Item “bar”

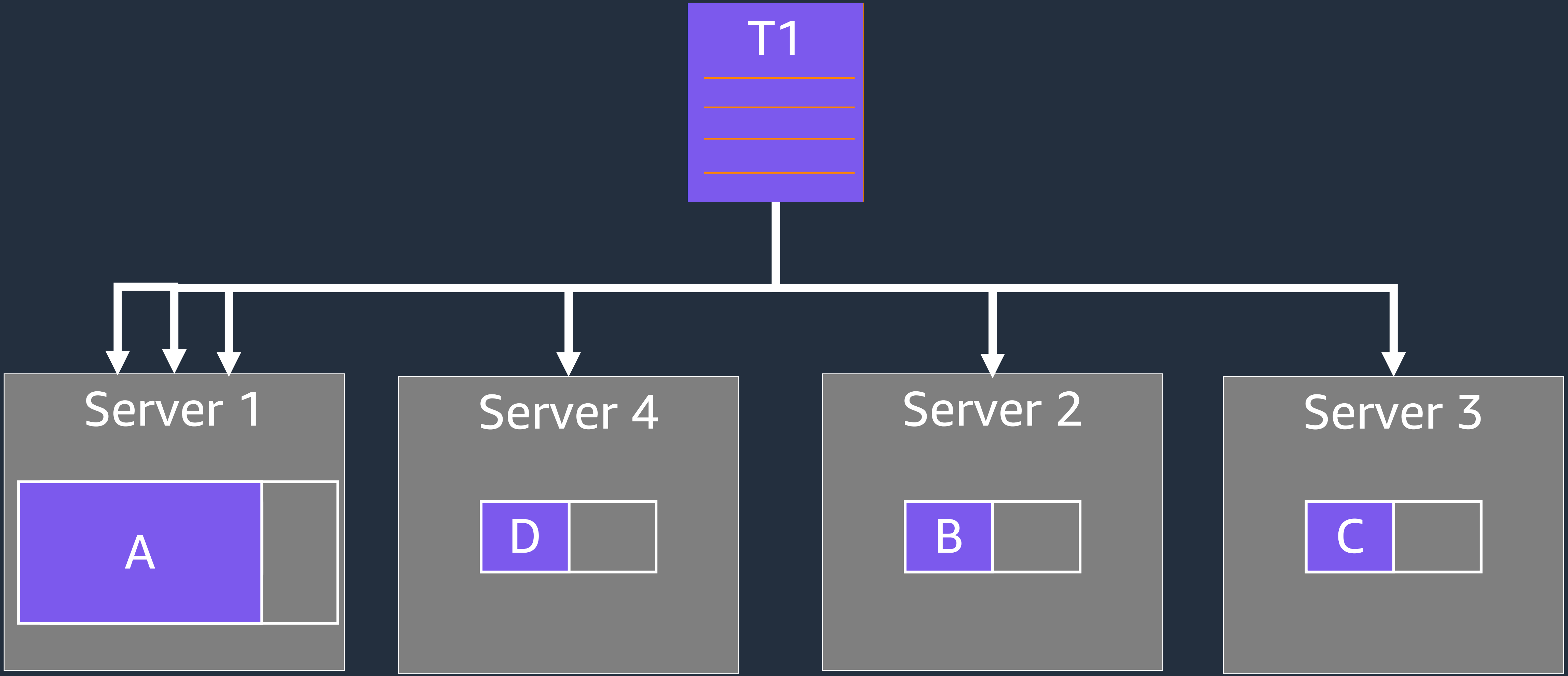


Partition B

High-traffic item isolation



Adaptive capacity throughput boosting

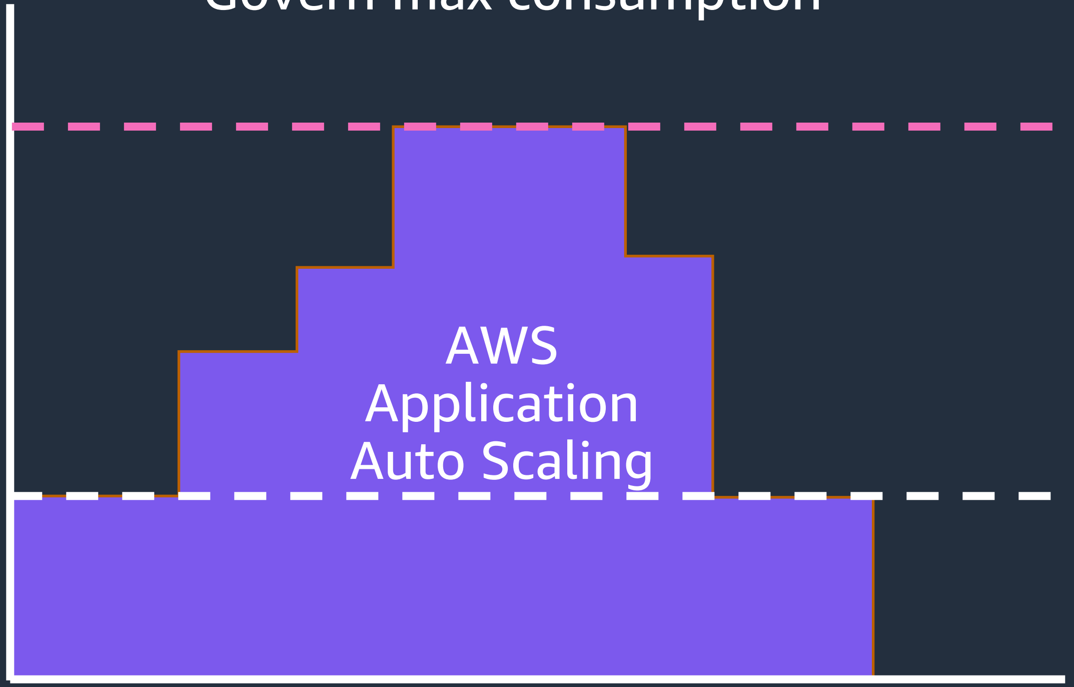


Provisioned capacity

DynamoDB capacity modes provide flexibility

Provisioned

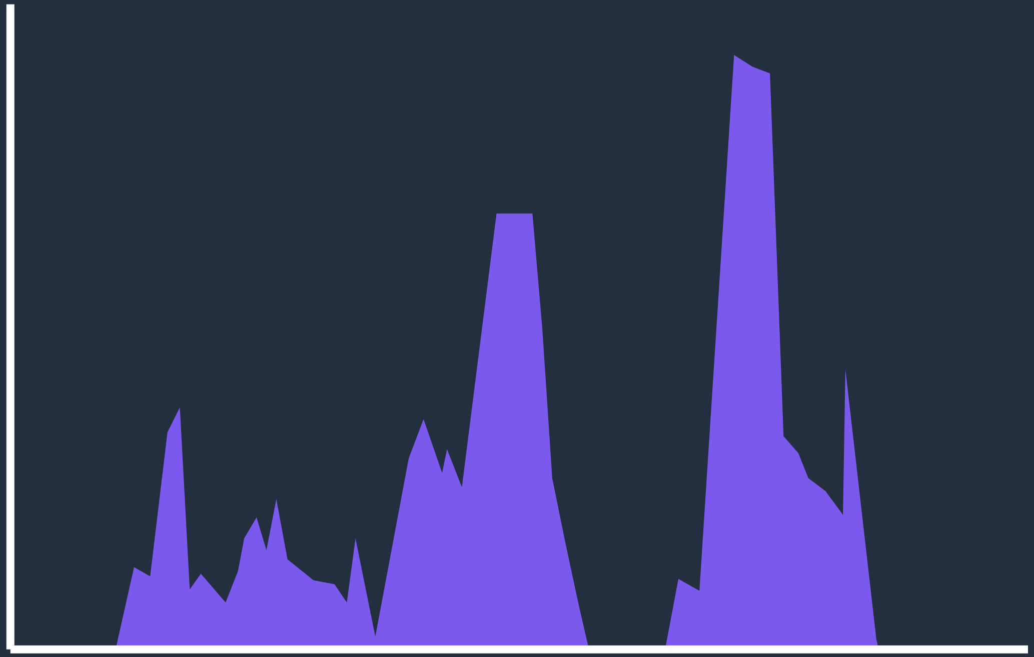
Govern max consumption



Set a minimum

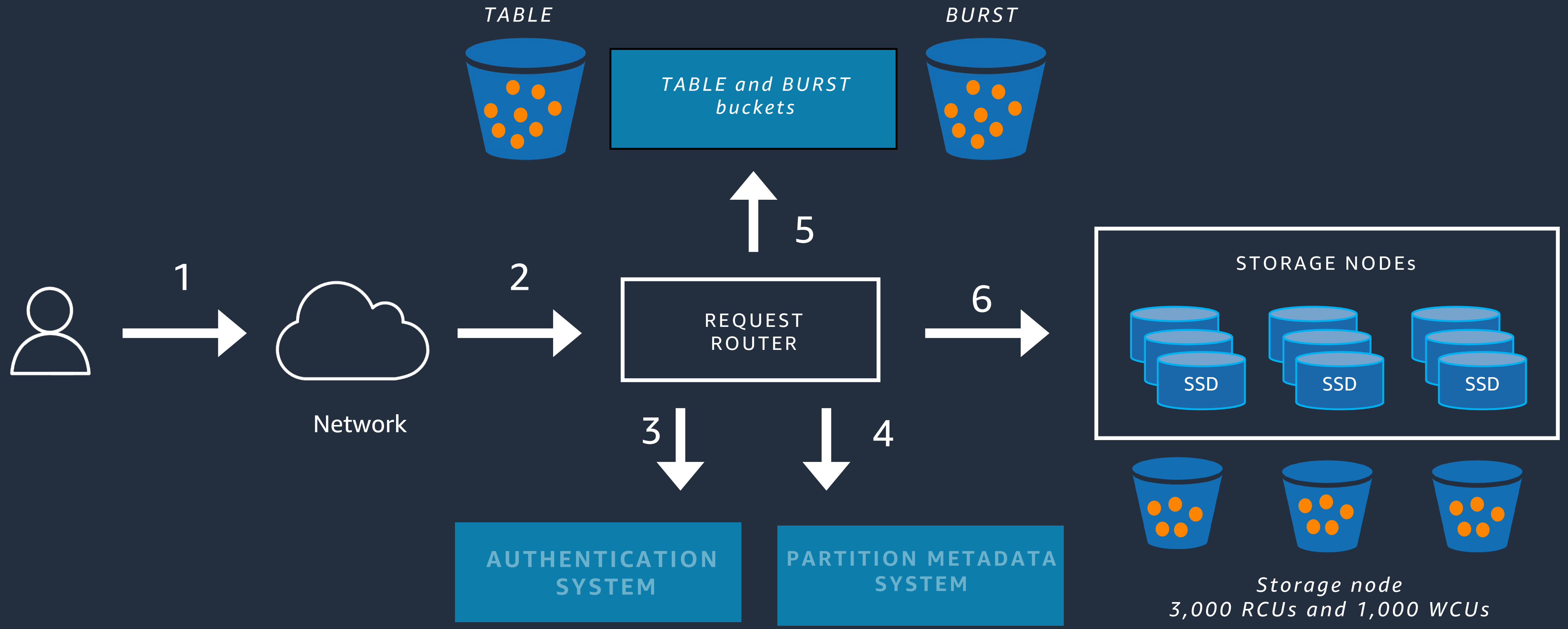
On-demand

No limit



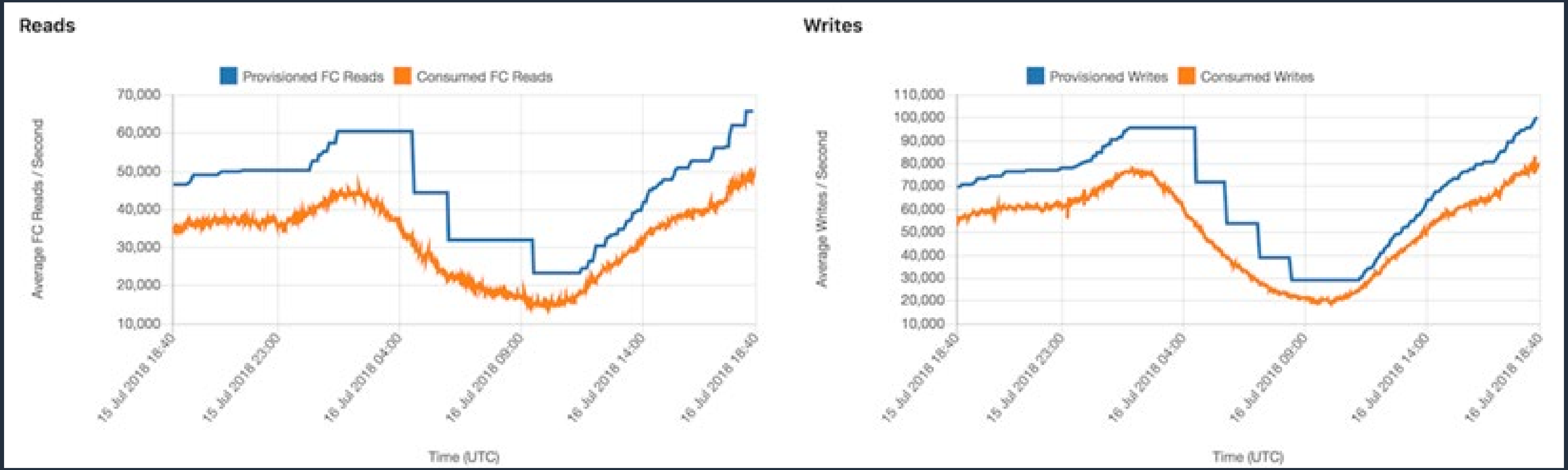
Start at zero

Token buckets manage provisioned throughput

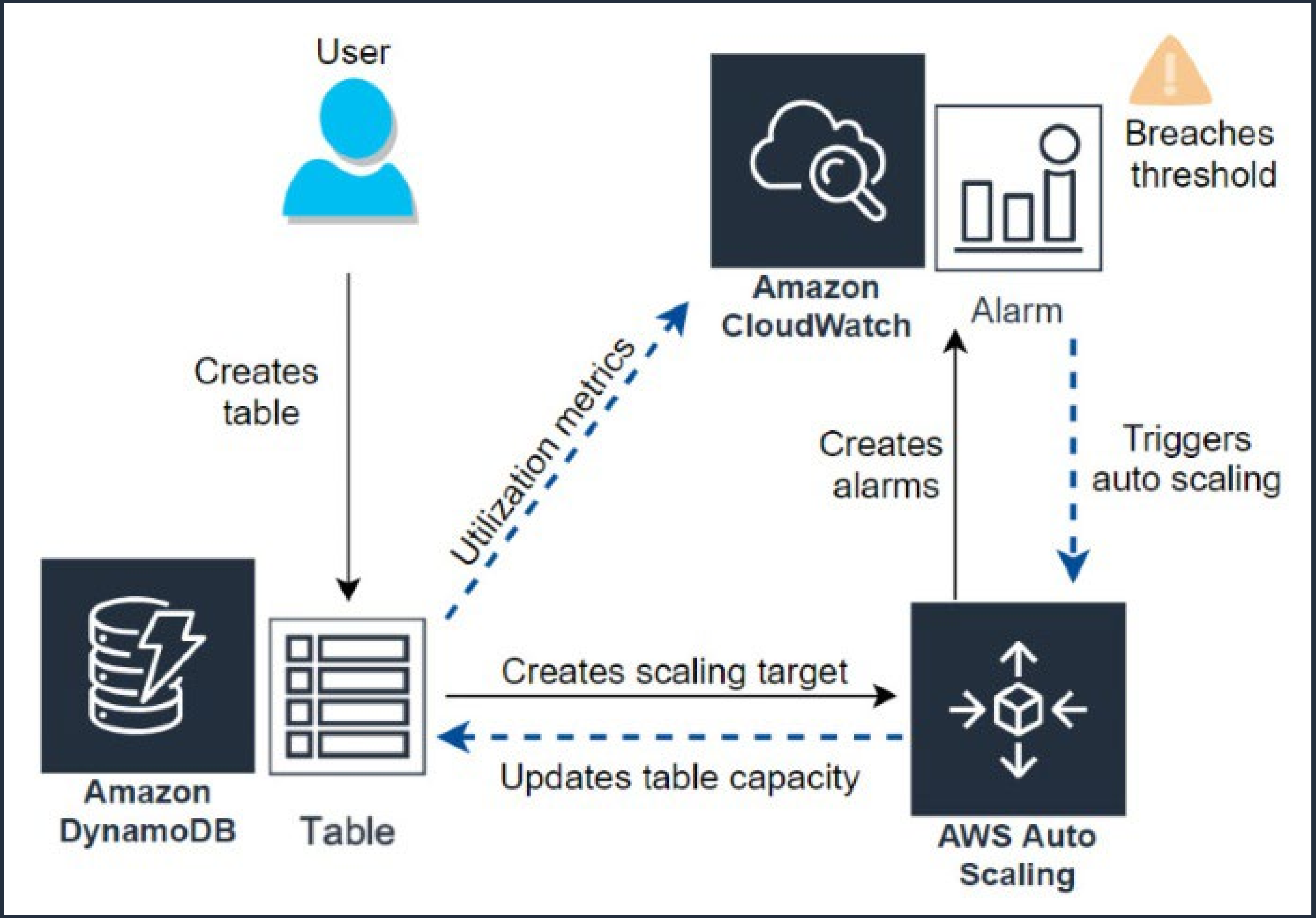


Provisioned capacity with auto scaling

RCU and WCU levels adjusted automatically



Auto scaling



Choosing a capacity mode

Use provisioned mode

- Steady workloads
- Gradual ramps
- Events with known traffic
- Ongoing monitoring

Use on-demand mode

- Unpredictable workloads
- Frequently idle workloads
- Events with unknown traffic
- Automatic scaling (up and down to zero)

Consider your tolerance for operational overhead and overprovisioning

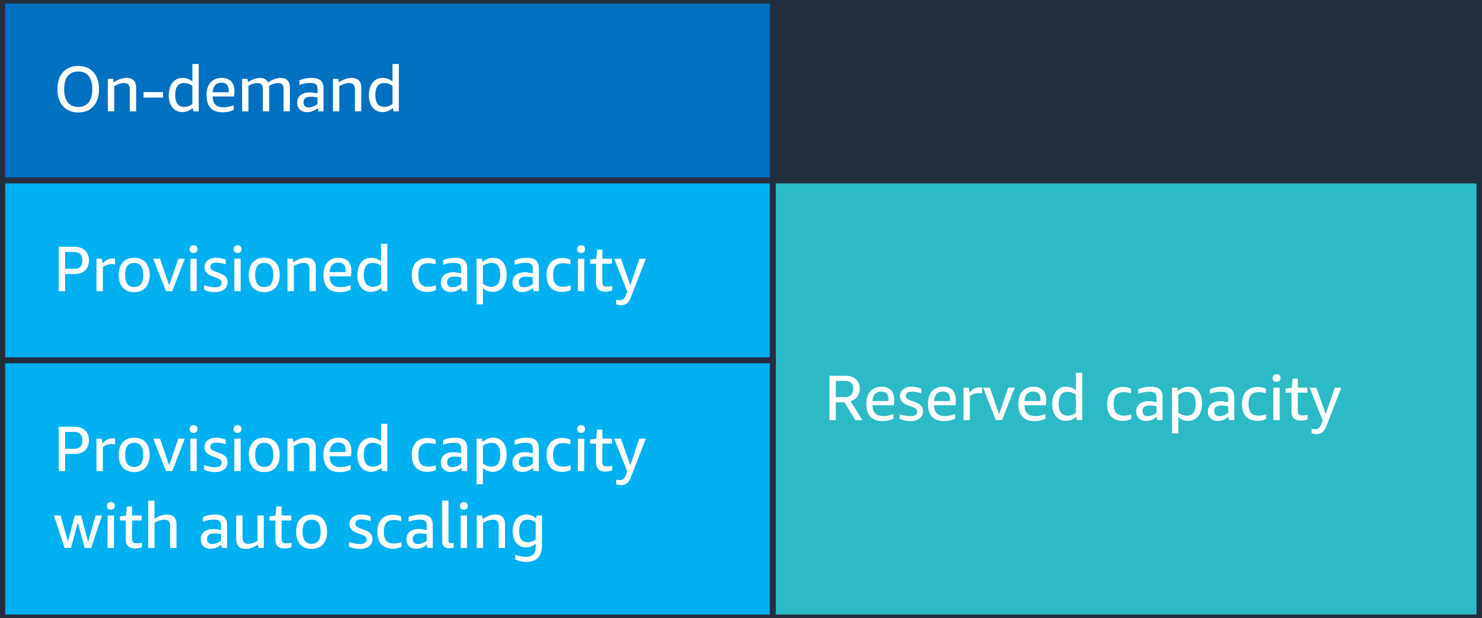
Quantify the provisioned throughput needed for the event

- 1 RCU = One 4 KB strongly consistent read
  - Or, two 4 KB eventually consistent reads
- 1 WCU = One 1 KB write

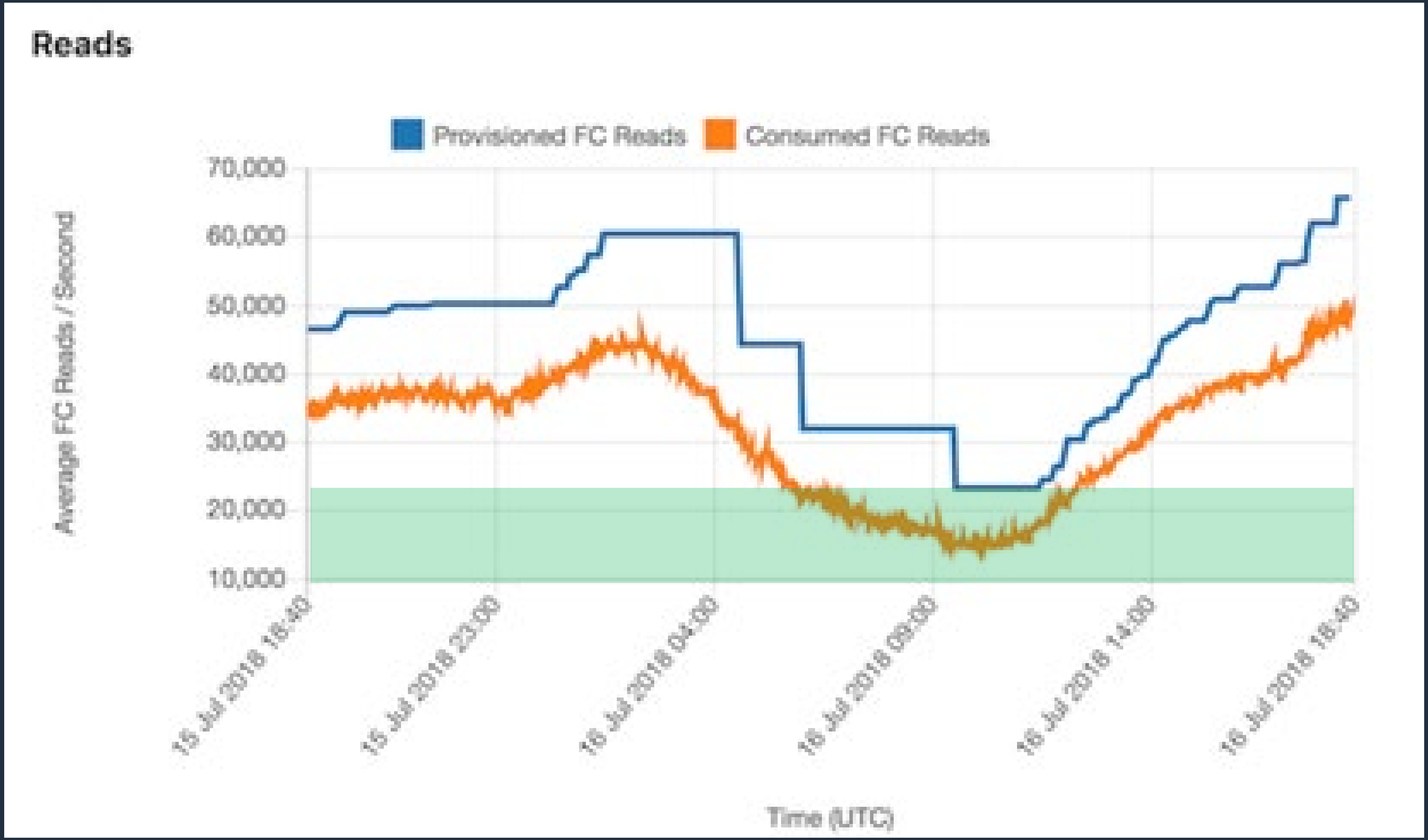
- RCU needed = Round up (item size in KB/4 KB) x reads per second
- WCU needed = Round up (item size in KB/1 KB) x writes per second

** Single partition can handle 3,000 RCUs and 1,000 WCUs

Reserved capacity clarification

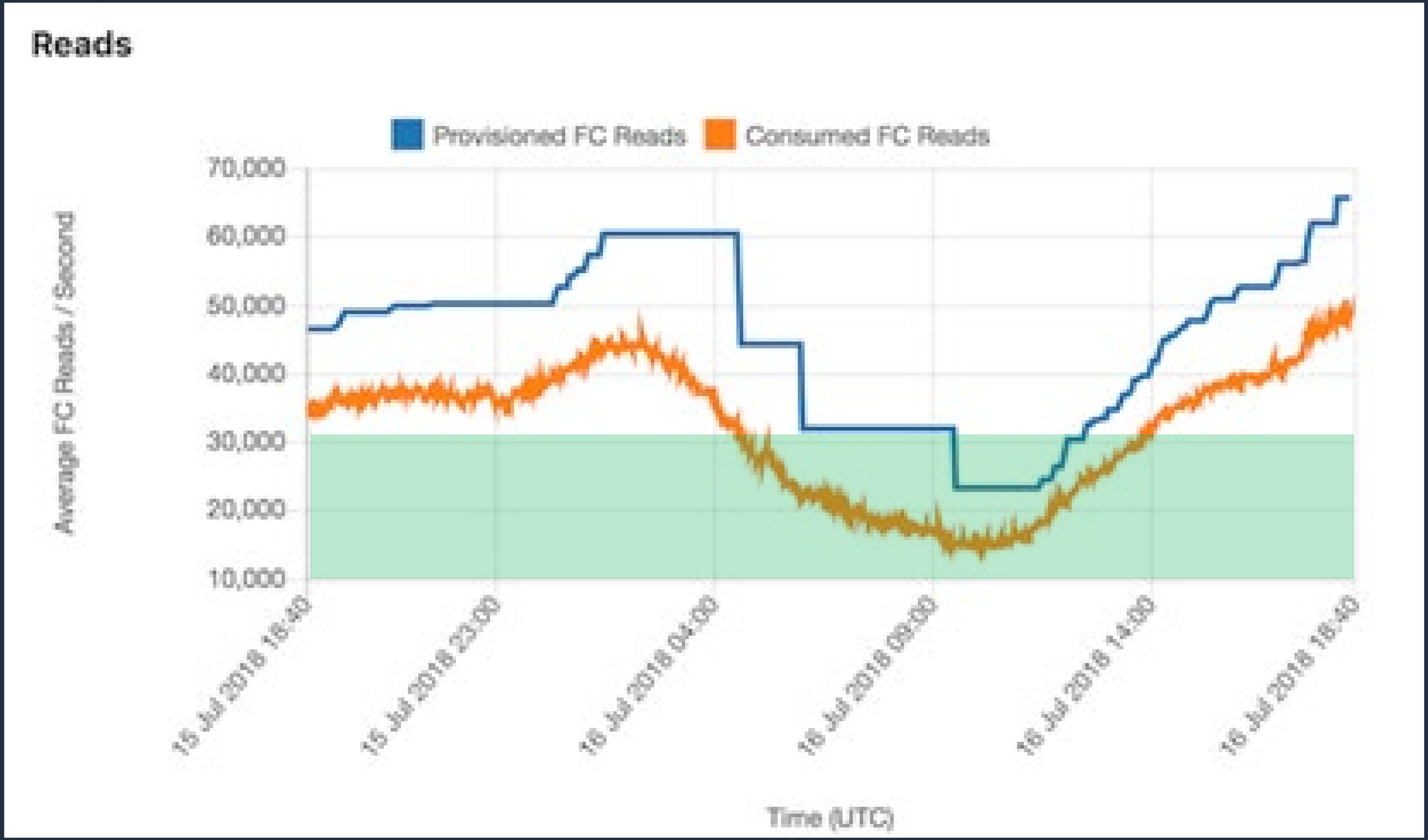


Reserved capacity – FLOOR provisioned level

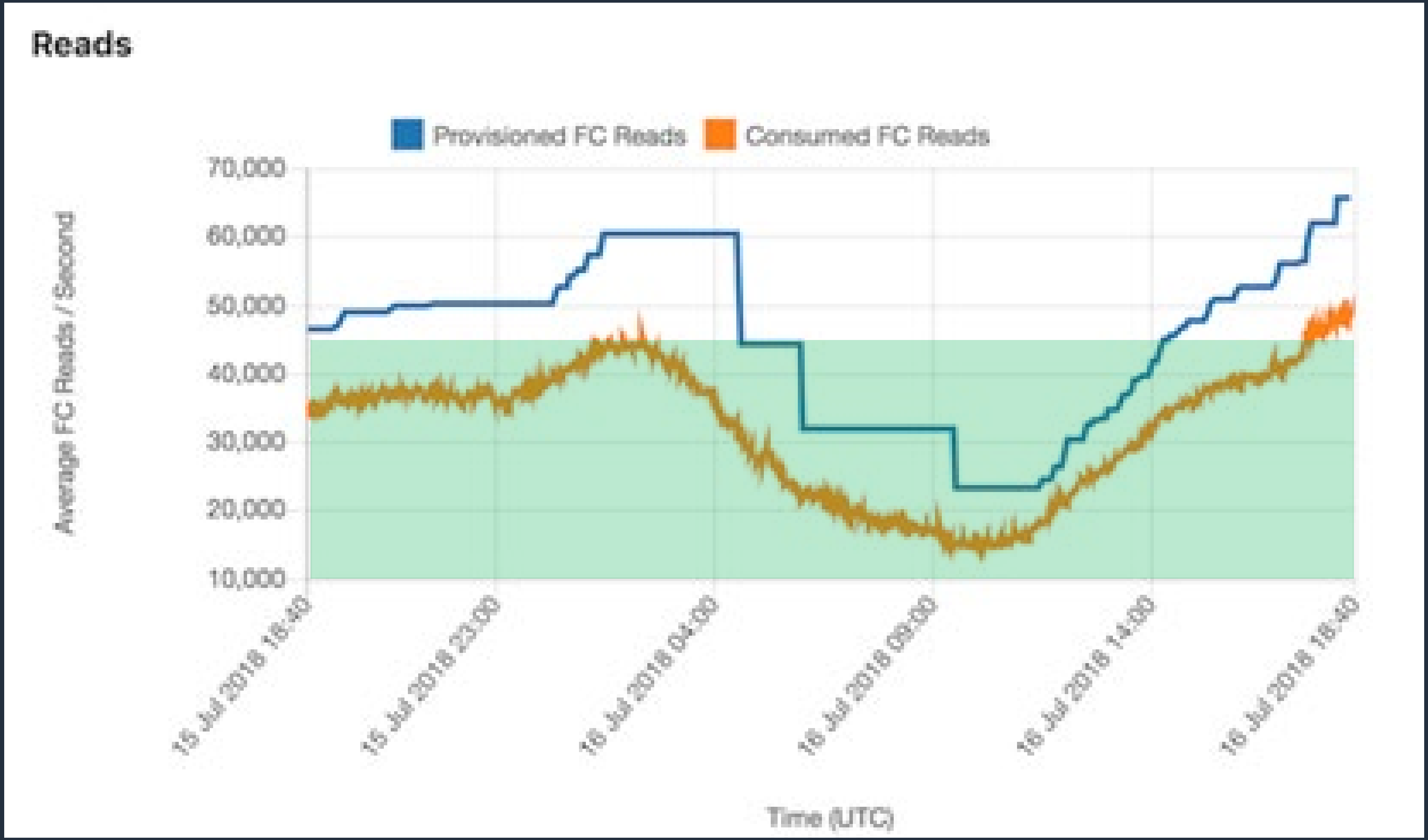


Reserved capacity – FLOOR plus a bit

✓

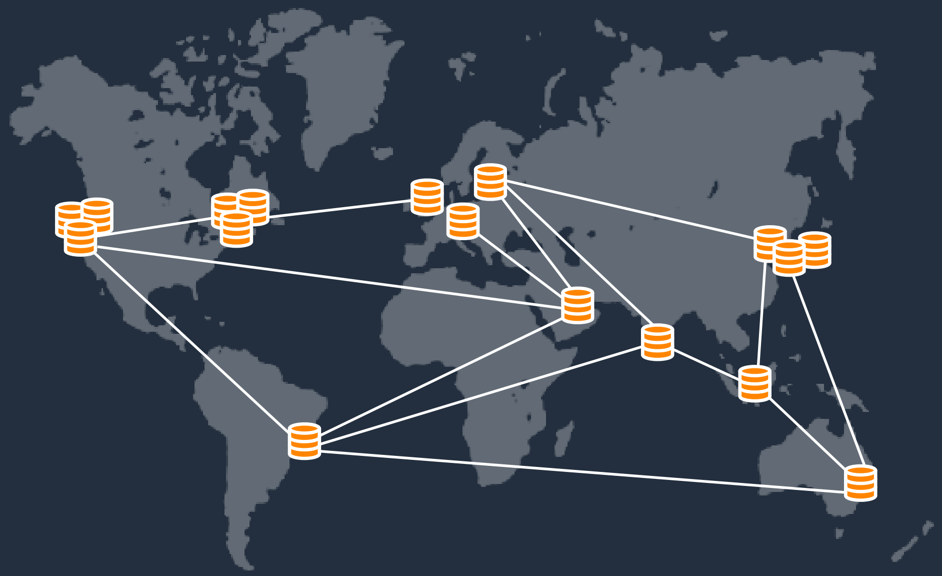


Reserved capacity – AVERAGE level for month



Global tables

DynamoDB global tables



Build high-performance, globally distributed applications

Low-latency reads and writes to locally available tables

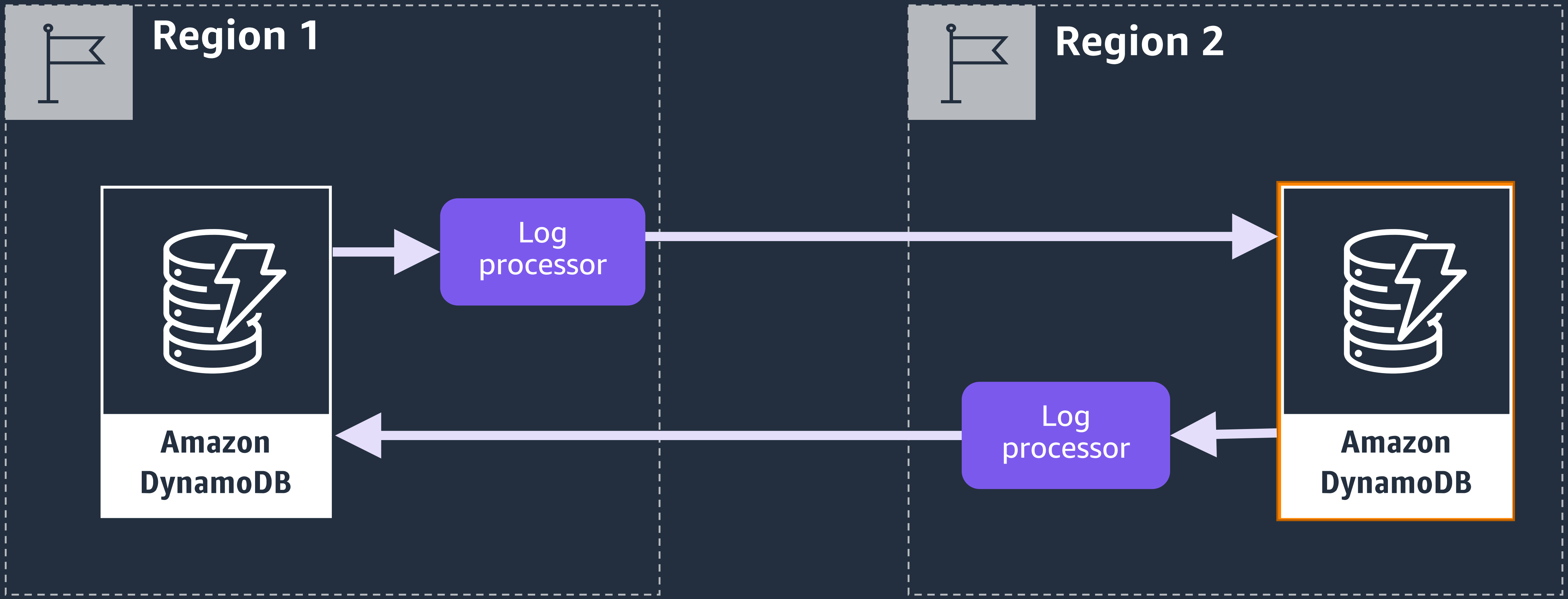
Multi-Region redundancy and resiliency and 99.999% availability

Multi-active writes from any Region

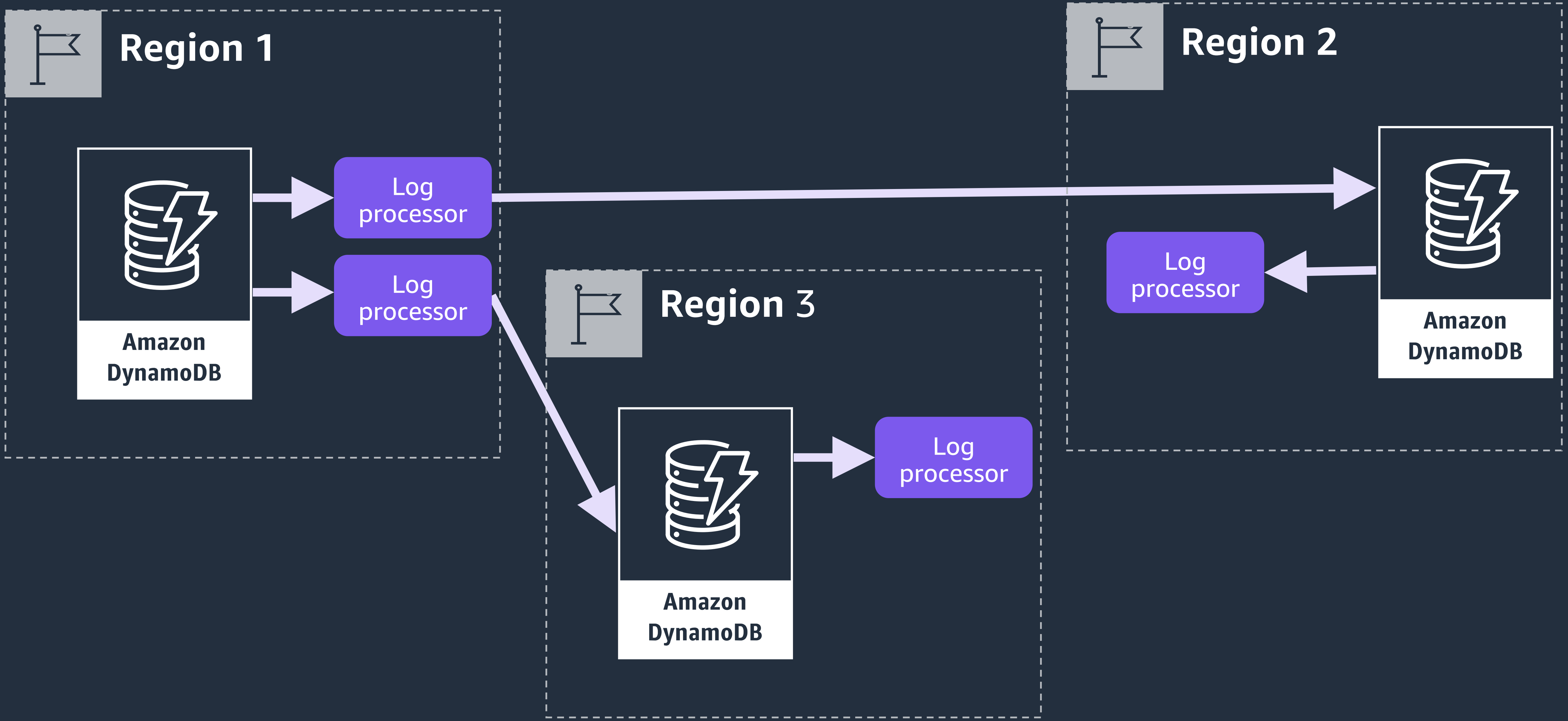
Easy to set up and no application rewrites required

Implement hot/hot DR solutions with global table

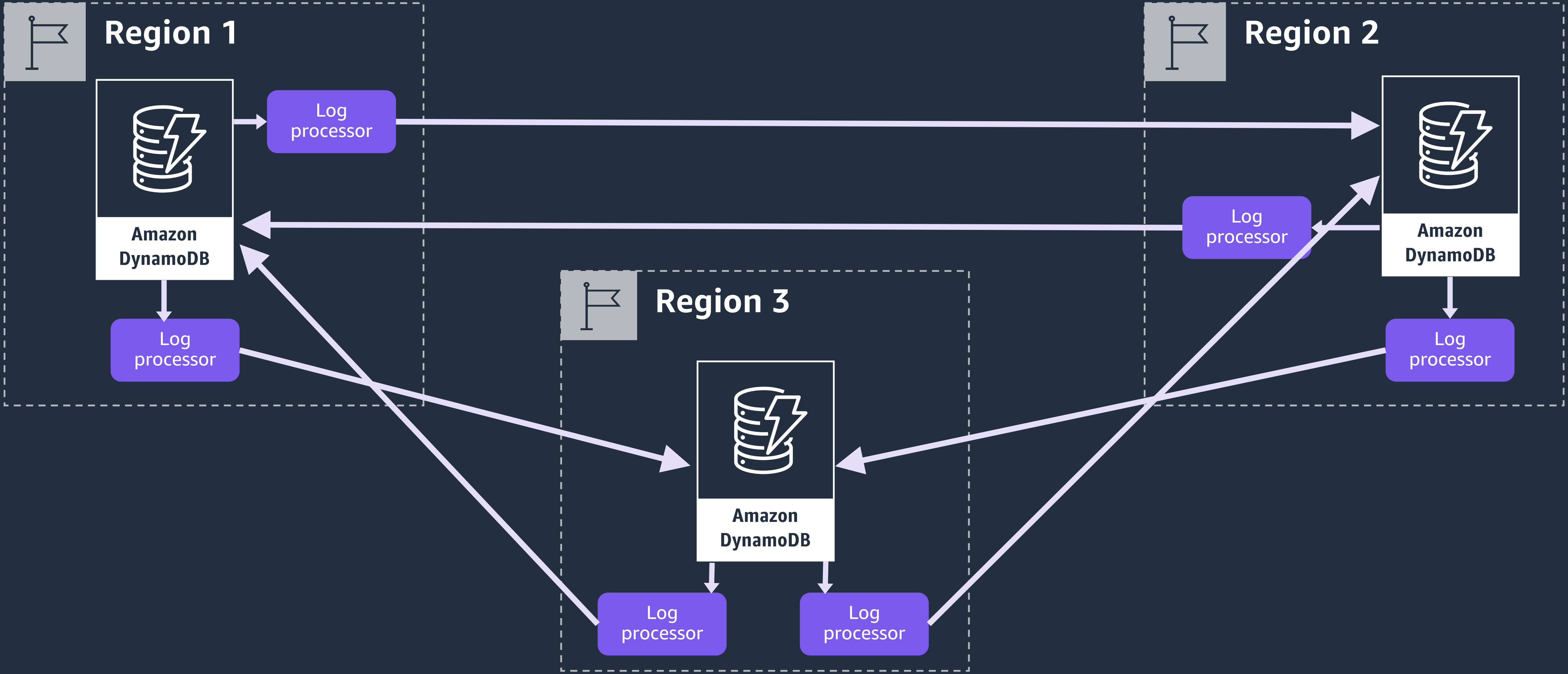
Multi-active replication



Multi-Region replication

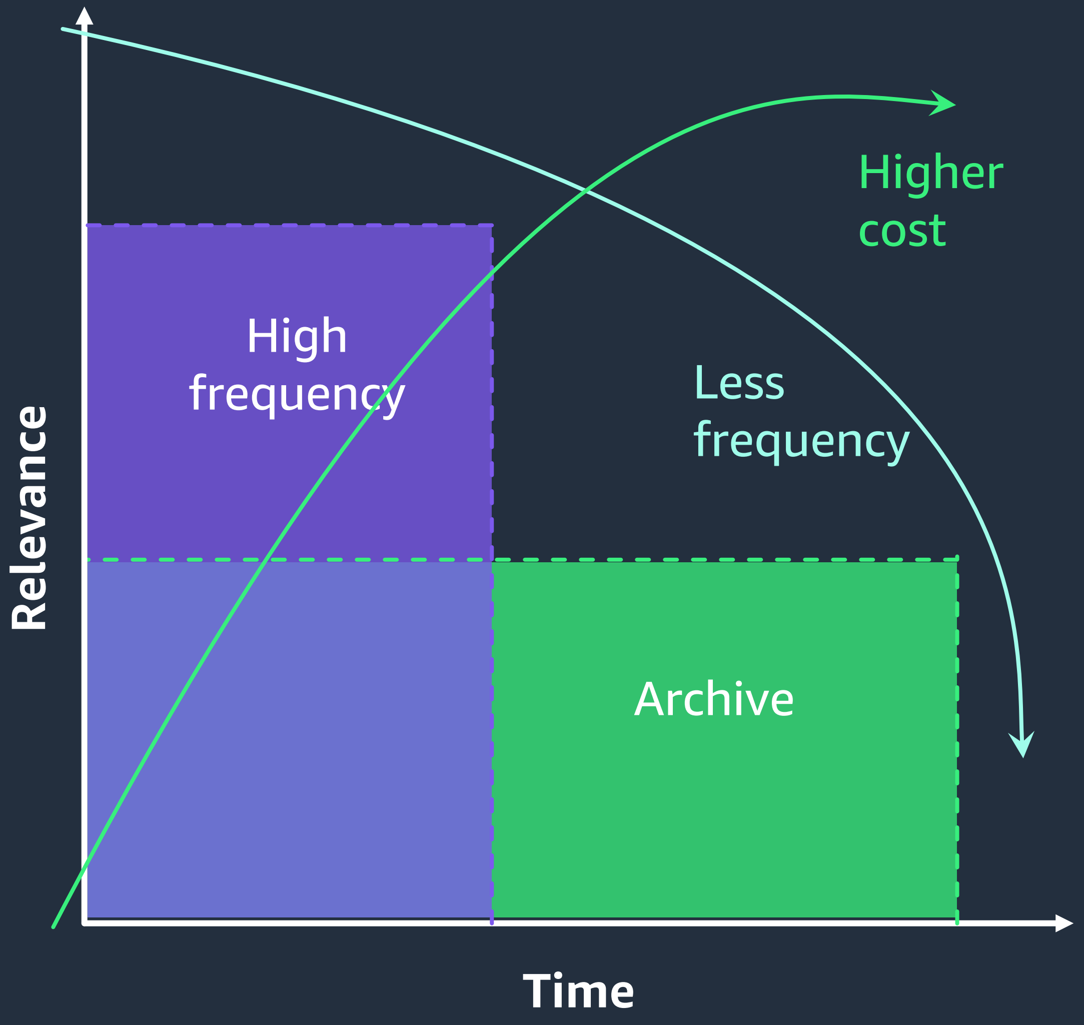


Multi-Region



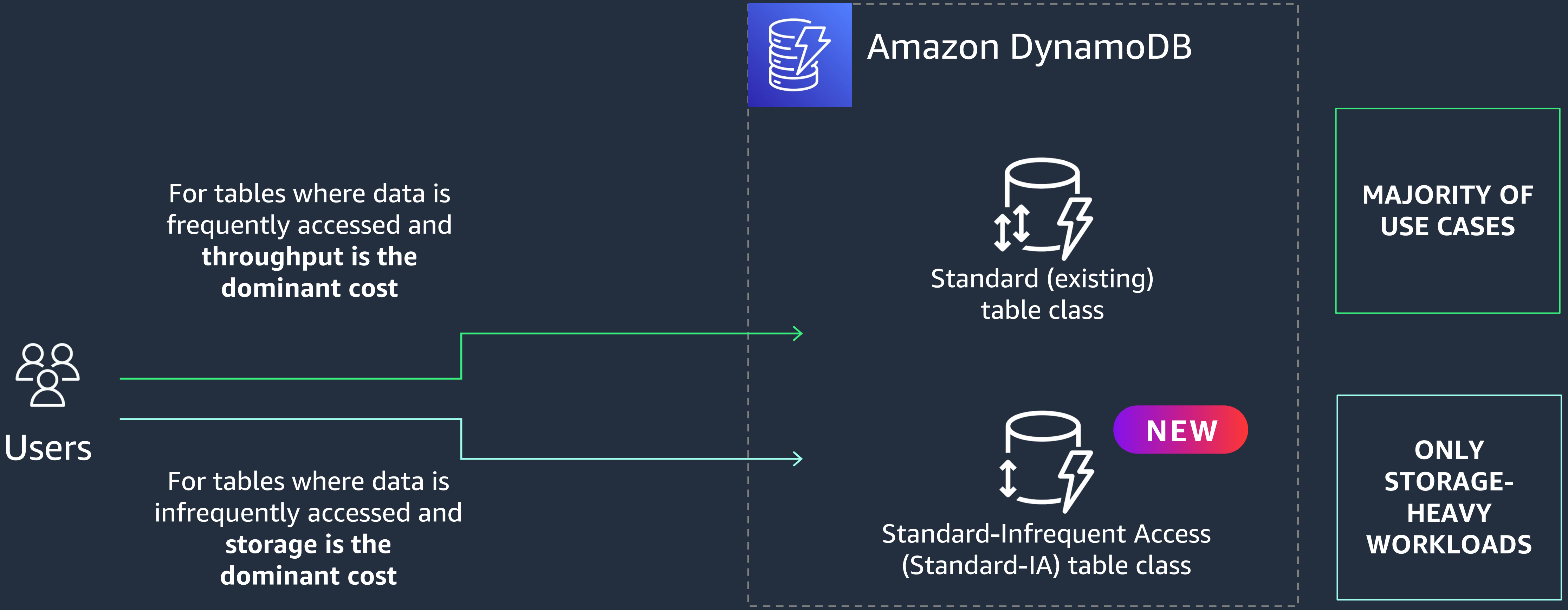
Storage

Data lifecycle



- Data volume is growing fast
- Data relevance decreases over time
- Older data is less frequently accessed
- Storing data can get more expensive at scale

Flexibility to manage your data with a new table class


Users
For tables where data is frequently accessed and **throughput is the dominant cost**
For tables where data is infrequently accessed and **storage is the dominant cost**
Amazon DynamoDB
Standard (existing) table class
NEW
Standard-Infrequent Access (Standard-IA) table class
MAJORITY OF USE CASES
ONLY STORAGE-HEAVY WORKLOADS

Demo

Thank you!

Shwetang Oza Robert McCauley