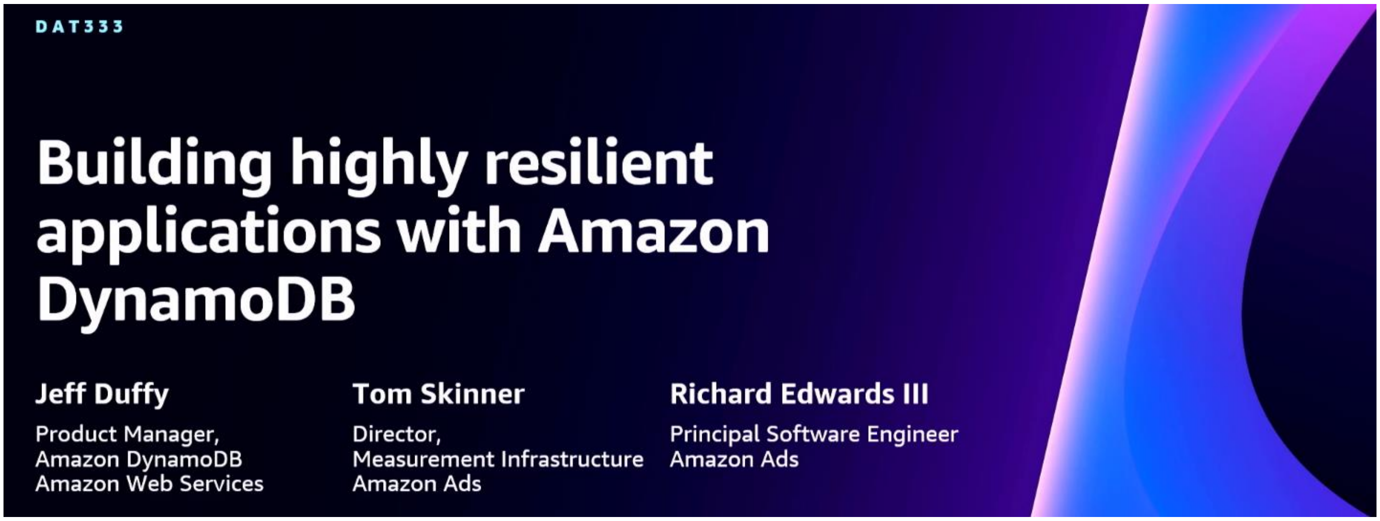


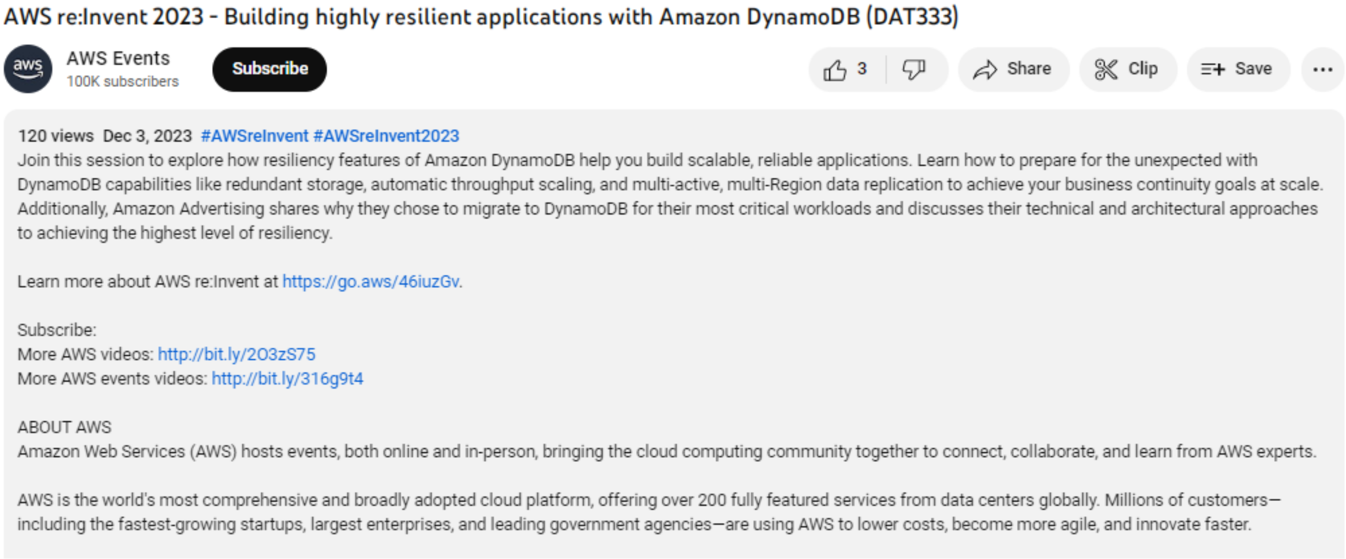
DAT333

Building highly resilient applications with Amazon DynamoDB

Jeff Duffy
Product Manager,
Amazon DynamoDB
Amazon Web Services

Tom Skinner
Director,
Measurement Infrastructure
Amazon Ads

Richard Edwards III
Principal Software Engineer
Amazon Ads



AWS re:Invent 2023 - Building highly resilient applications with Amazon DynamoDB (DAT333)

120 views Dec 3, 2023 #AWSreInvent #AWSreInvent2023

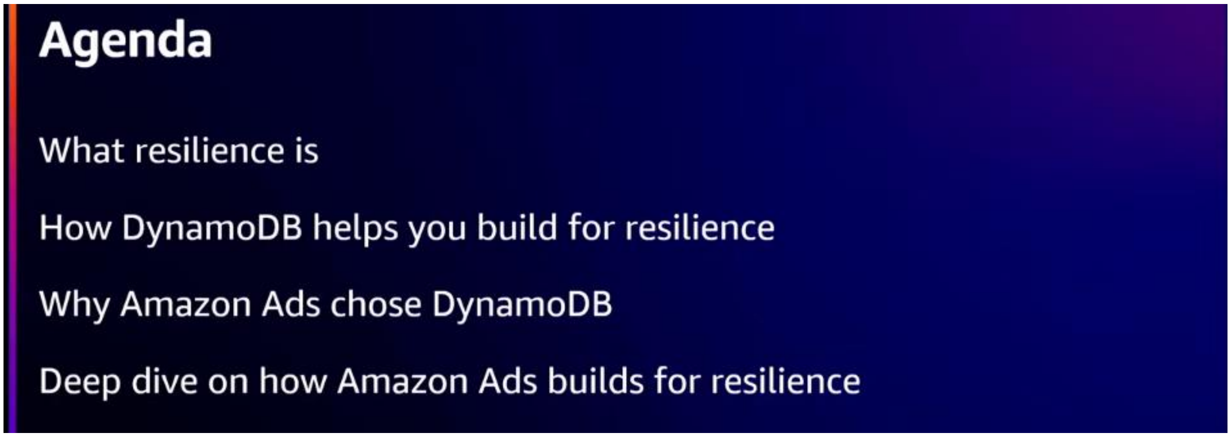
Join this session to explore how resiliency features of Amazon DynamoDB help you build scalable, reliable applications. Learn how to prepare for the unexpected with DynamoDB capabilities like redundant storage, automatic throughput scaling, and multi-active, multi-Region data replication to achieve your business continuity goals at scale. Additionally, Amazon Advertising shares why they chose to migrate to DynamoDB for their most critical workloads and discusses their technical and architectural approaches to achieving the highest level of resiliency.

Learn more about AWS re:Invent at https://go.aws/46iuzGv.

Subscribe:
More AWS videos: http://bit.ly/2O3zS75
More AWS events videos: http://bit.ly/316g9t4

ABOUT AWS
Amazon Web Services (AWS) hosts events, both online and in-person, bringing the cloud computing community together to connect, collaborate, and learn from AWS experts.

AWS is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.



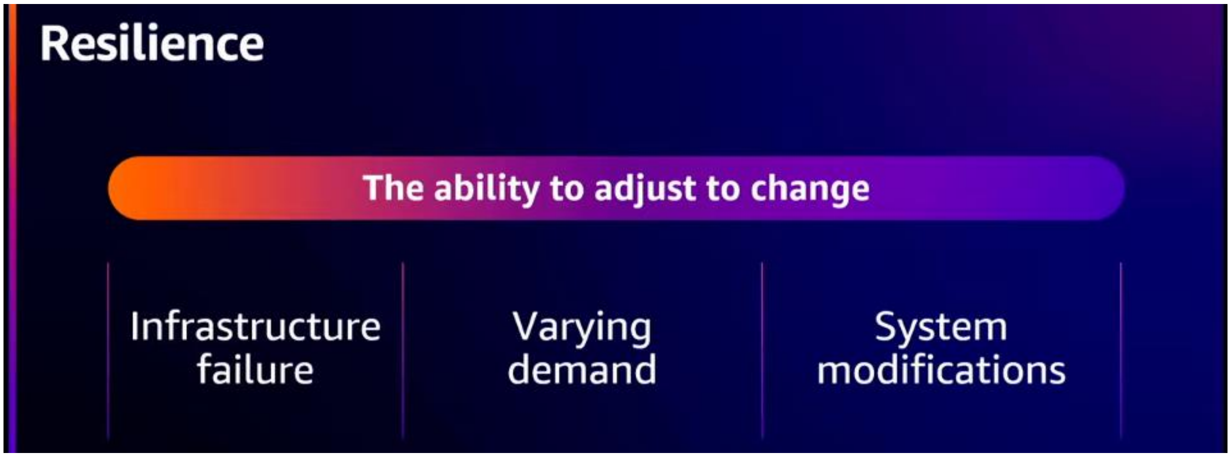
Agenda

What resilience is

How DynamoDB helps you build for resilience

Why Amazon Ads chose DynamoDB

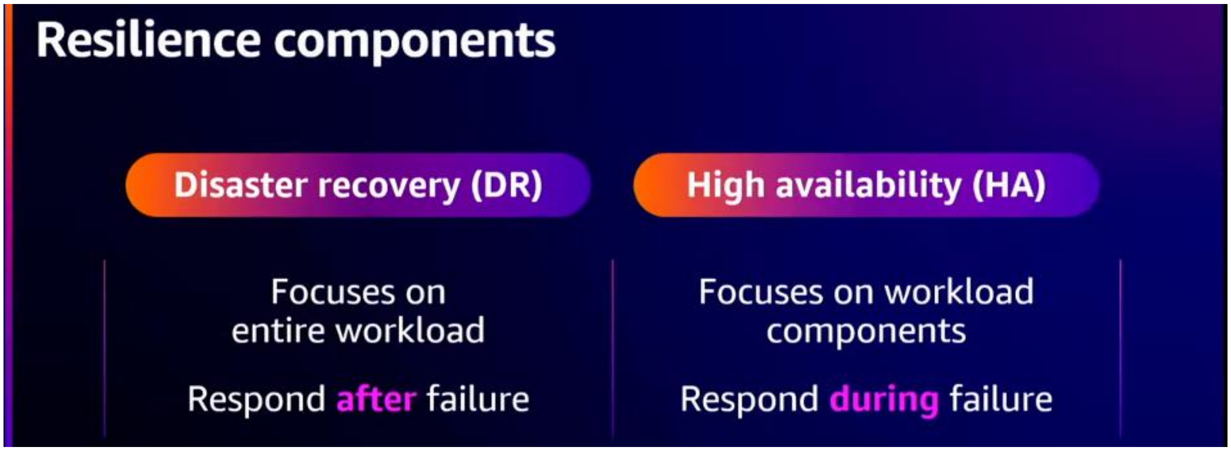
Deep dive on how Amazon Ads builds for resilience



Resilience

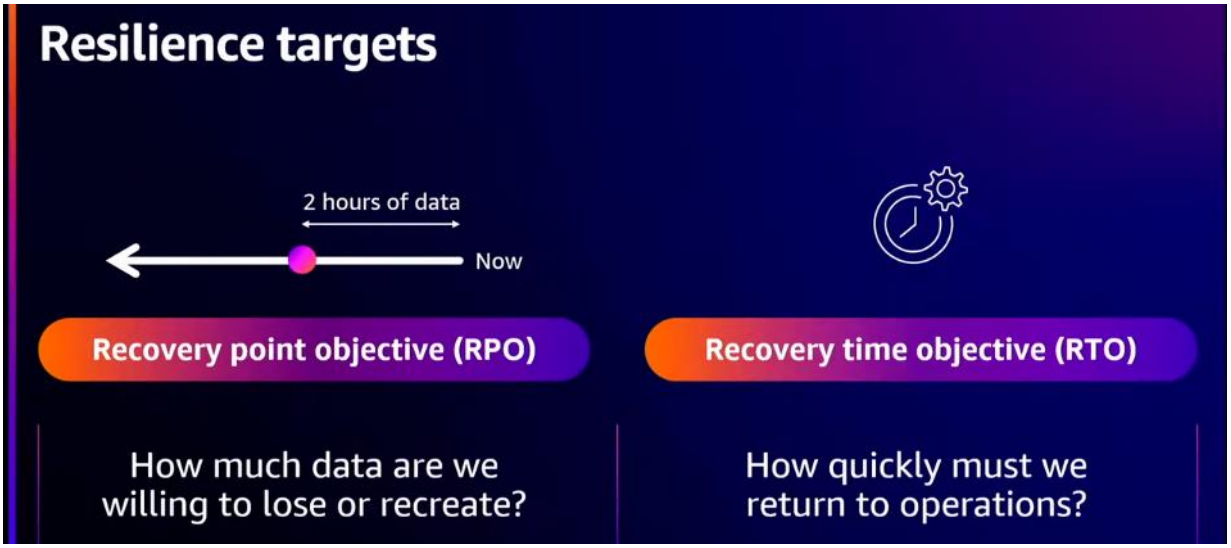
The ability to adjust to change

Infrastructure failure	Varying demand	System modifications



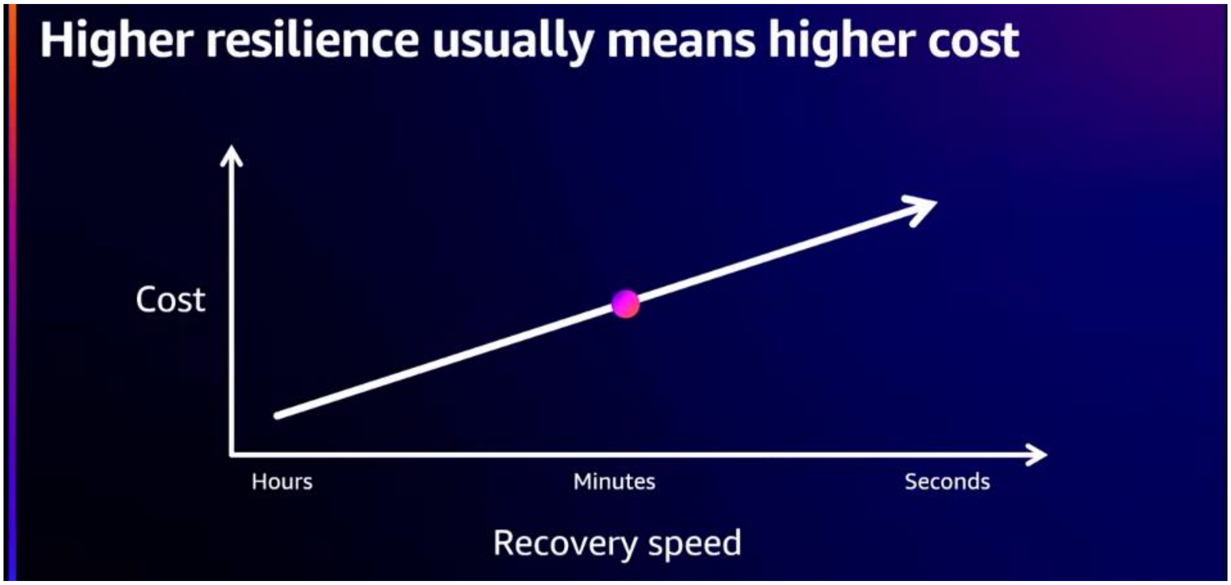
Resilience components

Disaster recovery (DR)	**High availability (HA)**
Focuses on entire workload	Focuses on workload components
Respond **after** failure	Respond **during** failure

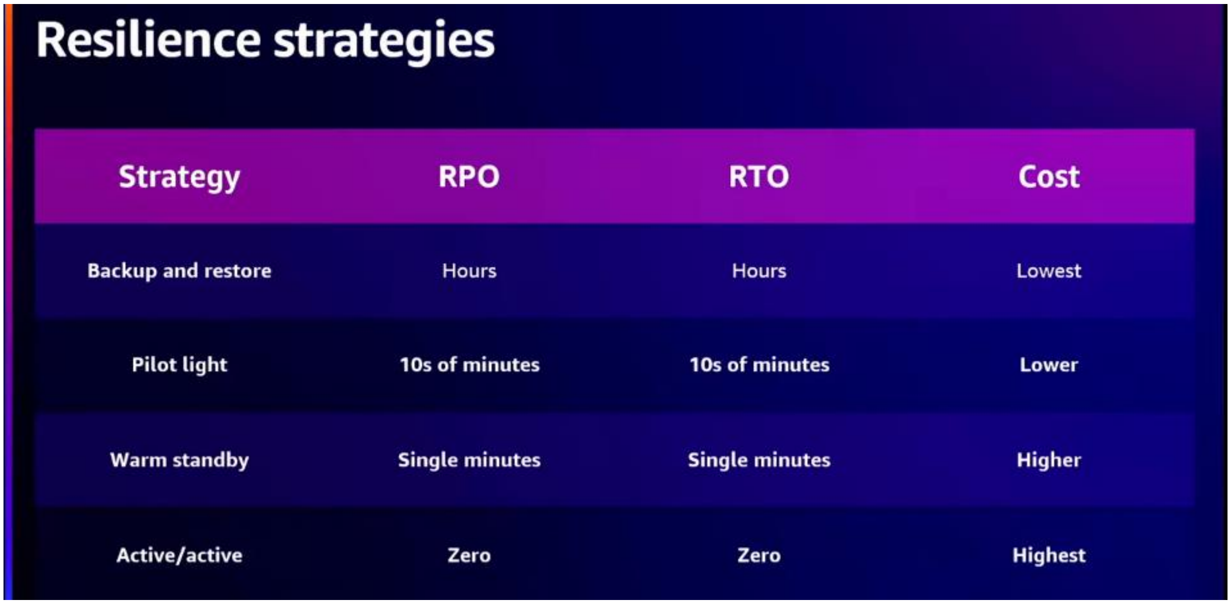


Resilience targets

Recovery point objective (RPO)	**Recovery time objective (RTO)**
How much data are we willing to lose or recreate?	How quickly must we return to operations?



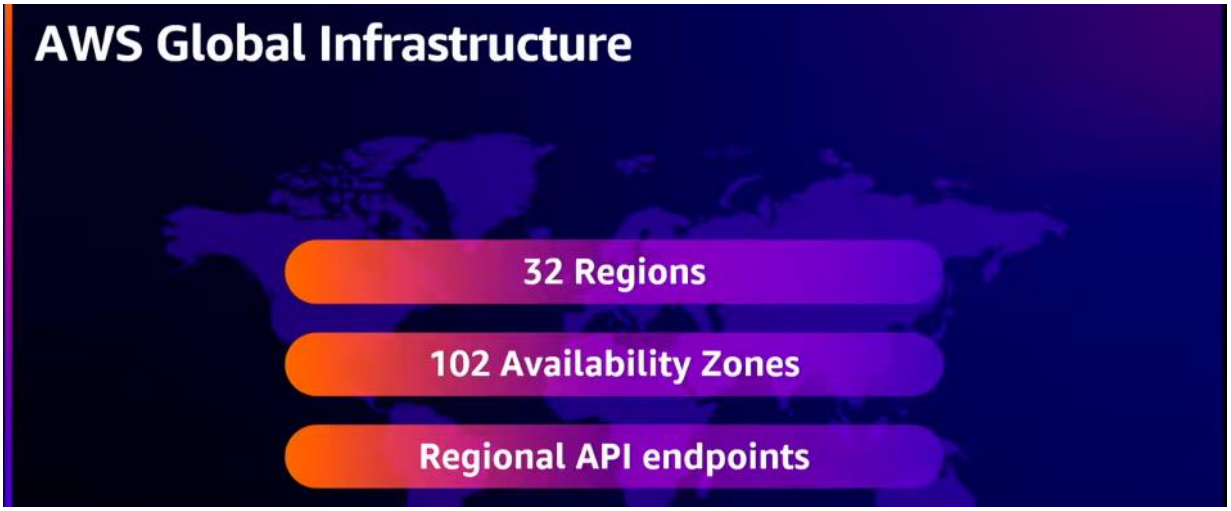
Higher resilience usually means higher cost



Resilience strategies

Strategy	RPO	RTO	Cost
Backup and restore	Hours	Hours	Lowest
Pilot light	10s of minutes	10s of minutes	Lower
Warm standby	Single minutes	Single minutes	Higher
Active/active	Zero	Zero	Highest

We have 4 strategies in our well architected framework

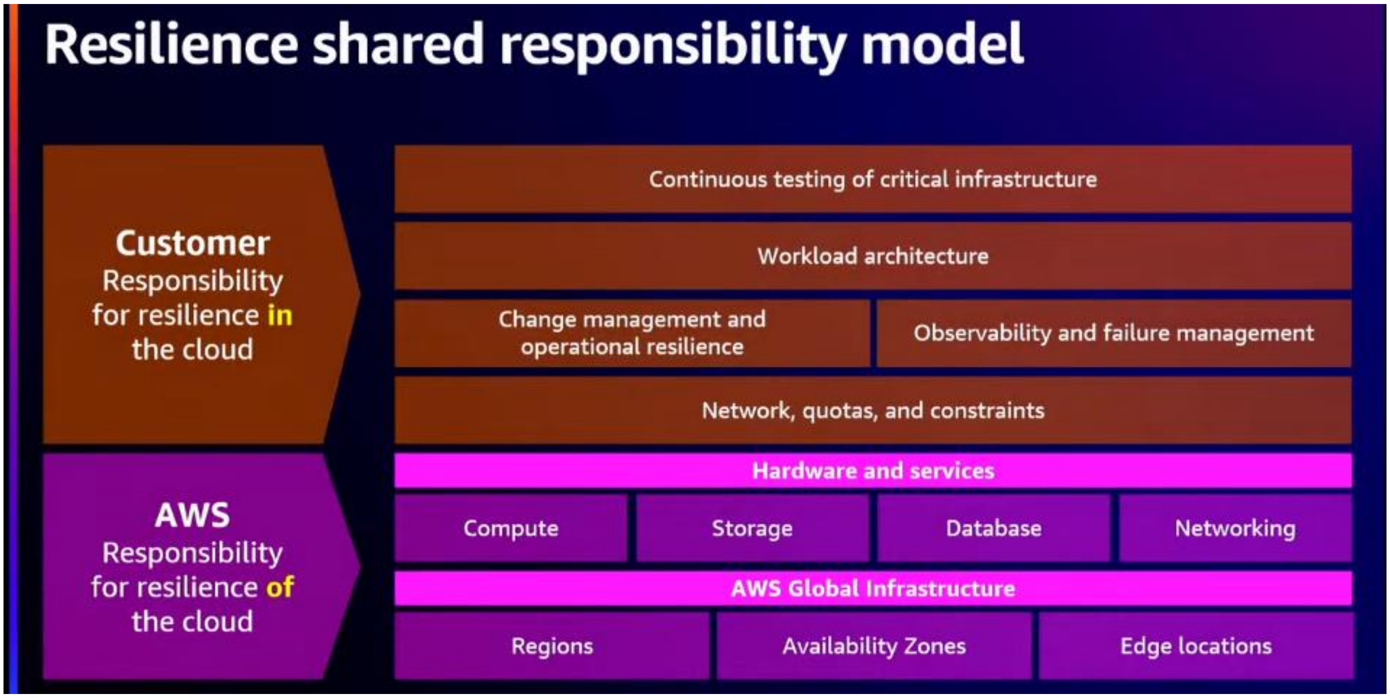


AWS Global Infrastructure

32 Regions

102 Availability Zones

Regional API endpoints



Resilience shared responsibility model

Customer Responsibility for resilience **in** the cloud

- Continuous testing of critical infrastructure
- Workload architecture
- Change management and operational resilience
- Observability and failure management
- Network, quotas, and constraints

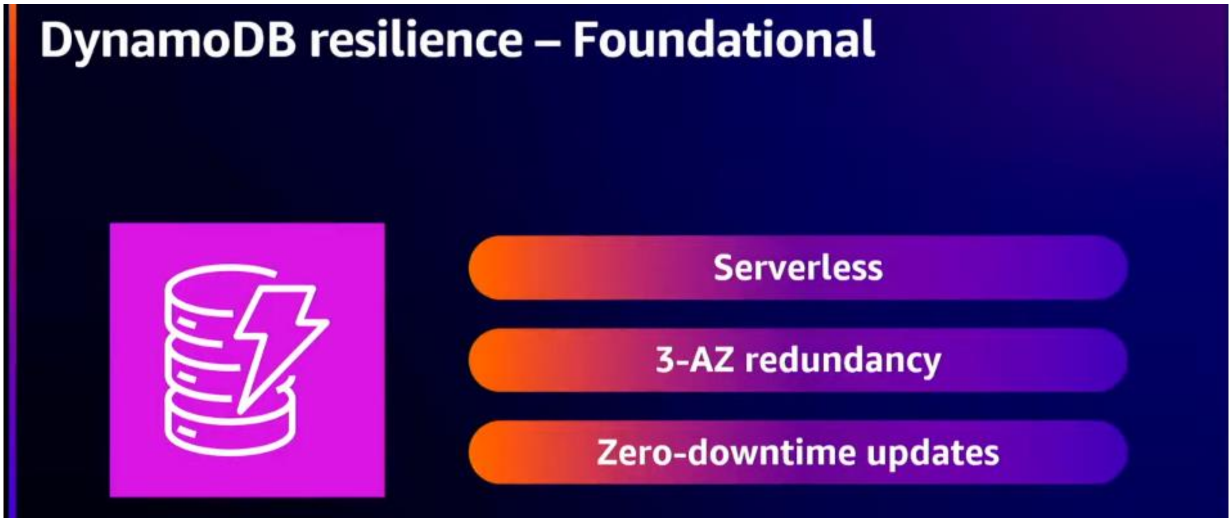
AWS Responsibility for resilience **of** the cloud

Hardware and services

Compute | Storage | Database | Networking

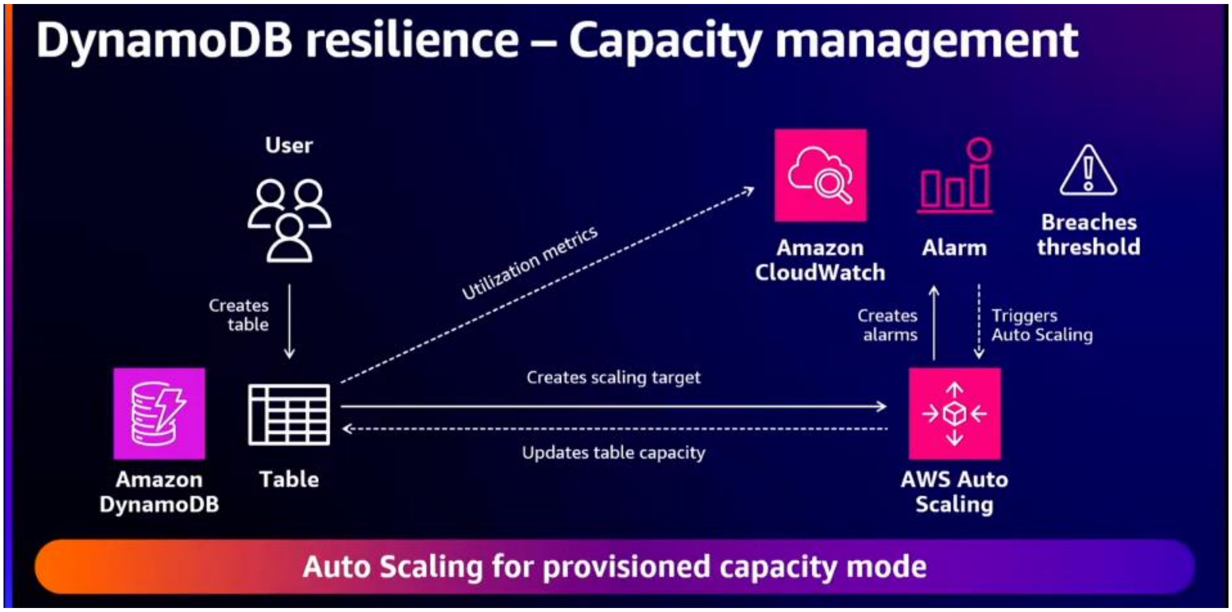
AWS Global Infrastructure

Regions | Availability Zones | Edge locations



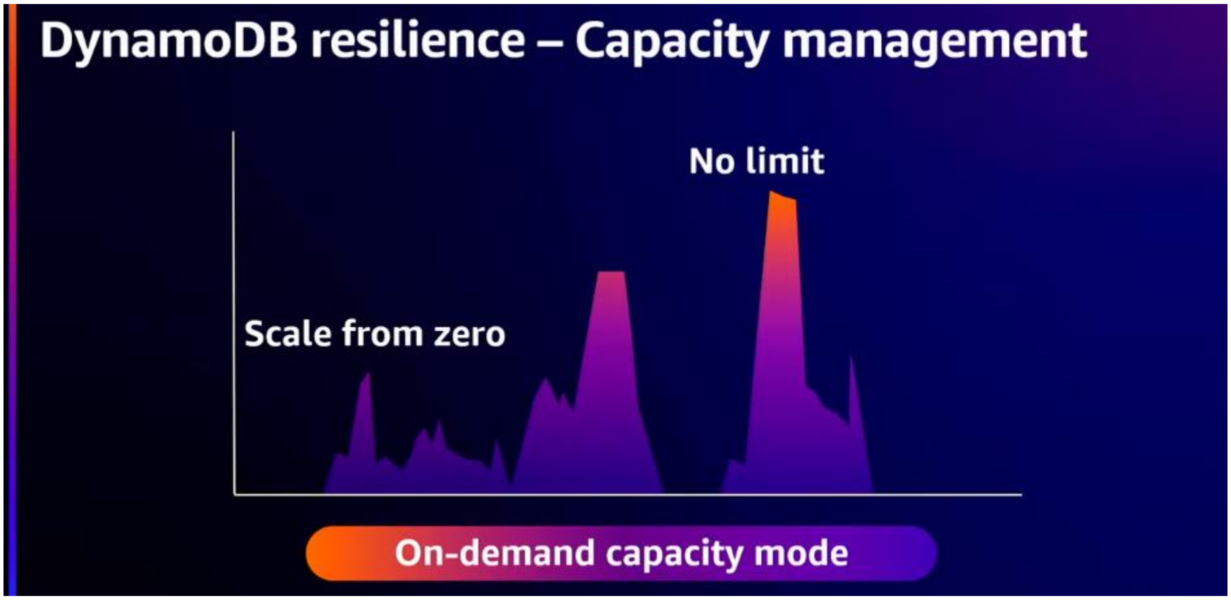
DynamoDB resilience – Foundational

- Serverless
- 3-AZ redundancy
- Zero-downtime updates



DynamoDB resilience – Capacity management

Auto Scaling for provisioned capacity mode



DynamoDB resilience – Capacity management

No limit

Scale from zero

On-demand capacity mode

DynamoDB resilience – Recovery

Point-in-time recovery (PITR)

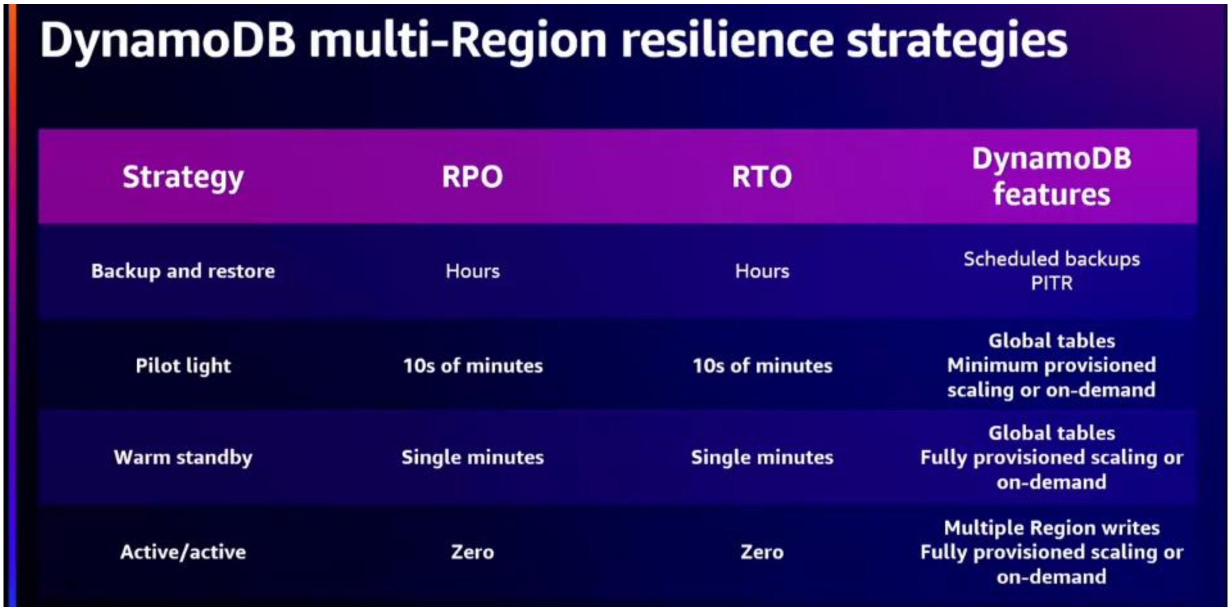
Scheduled backups

DynamoDB resilience – Global tables

Multi-active, multi-Region

99.999% availability

No failover required

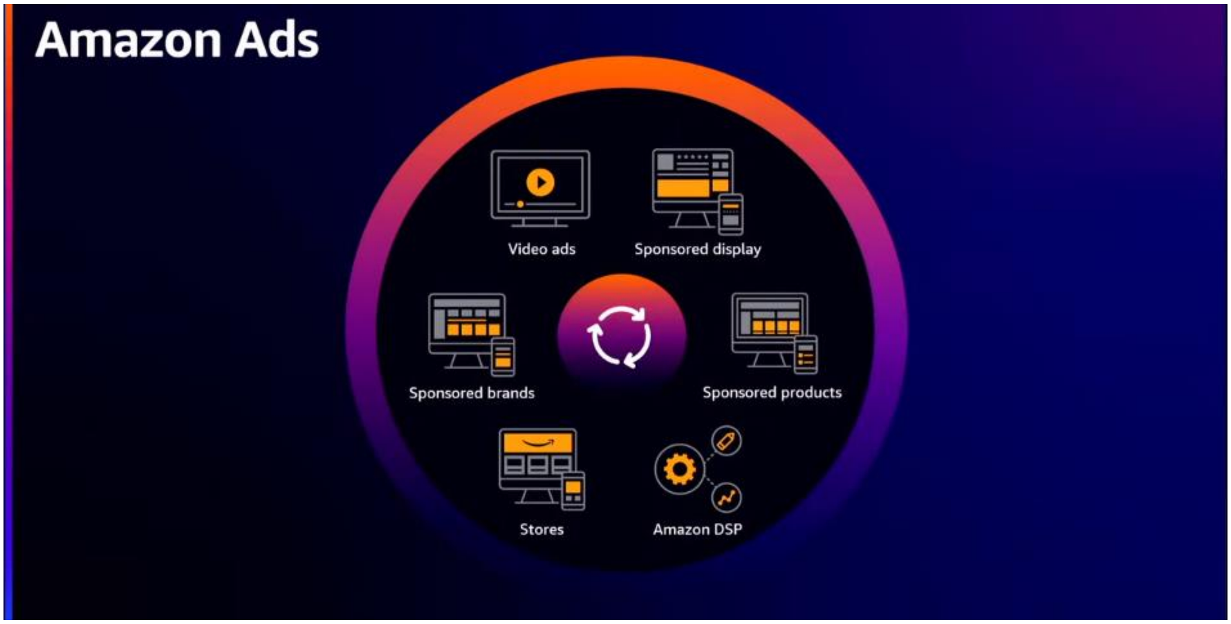


DynamoDB multi-Region resilience strategies

Strategy	RPO	RTO	DynamoDB features
Backup and restore	Hours	Hours	Scheduled backups PITR
Pilot light	10s of minutes	10s of minutes	Global tables Minimum provisioned scaling or on-demand
Warm standby	Single minutes	Single minutes	Global tables Fully provisioned scaling or on-demand
Active/active	Zero	Zero	Multiple Region writes Fully provisioned scaling or on-demand

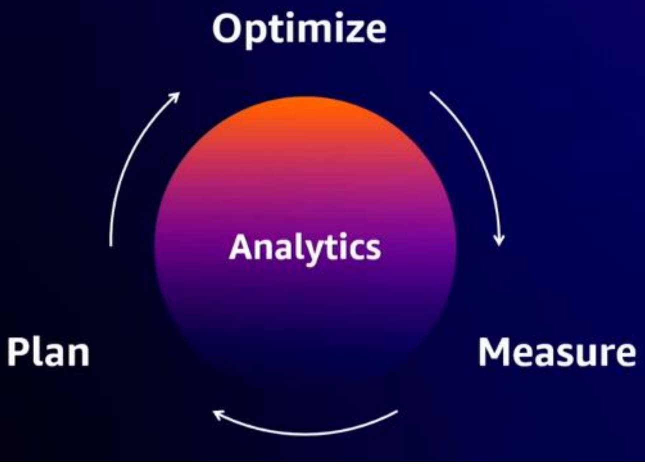
Ads measurement

Tom Skinner



Amazon Ads

Advertising analytics – Lifecycle



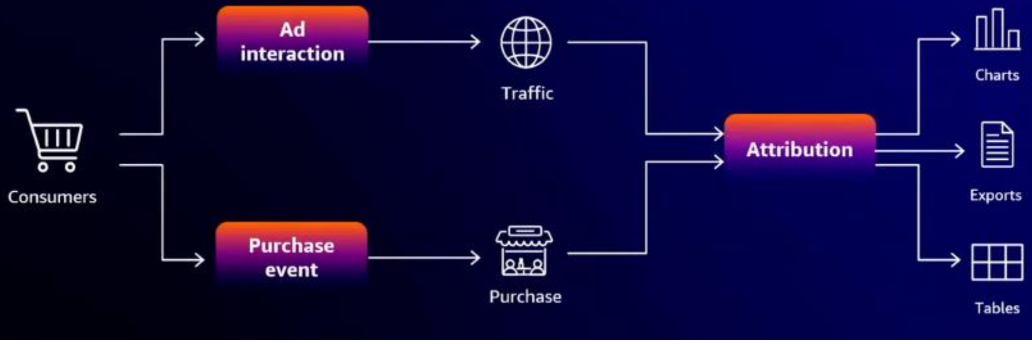
Optimize

Analytics

Plan

Measure

Attribution data flow



Consumers

Ad interaction

Traffic

Purchase event

Purchase

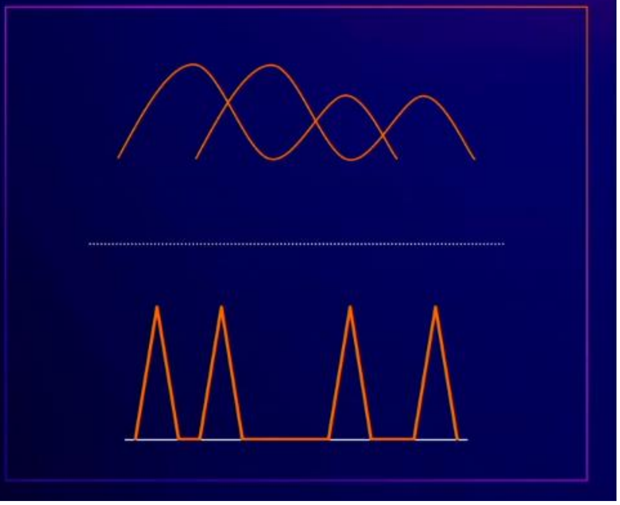
Attribution

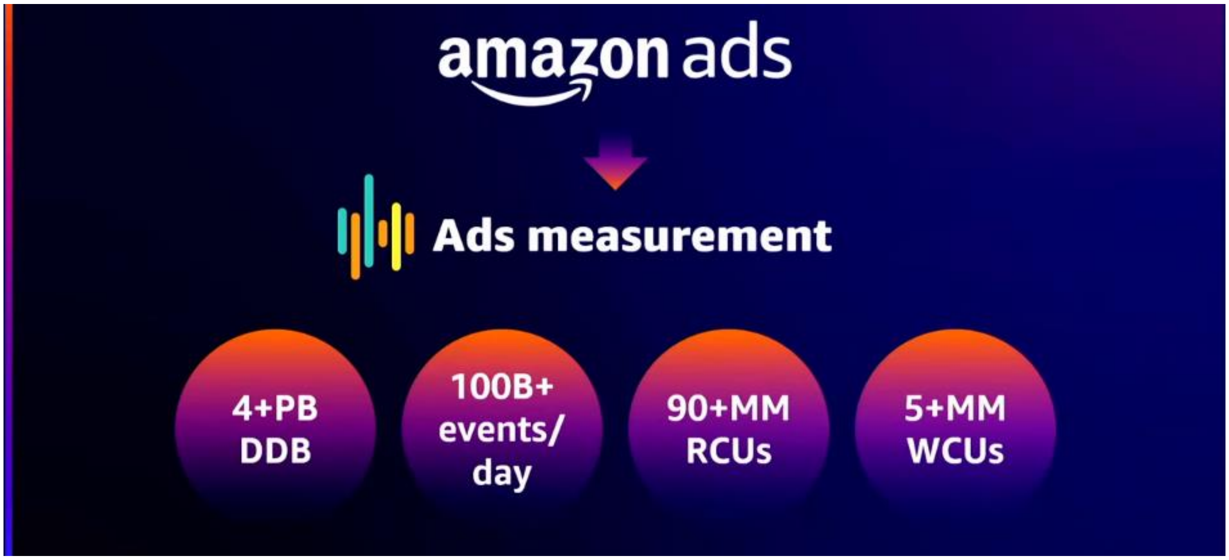
Charts

Exports

Tables

Workload demands





amazon ads

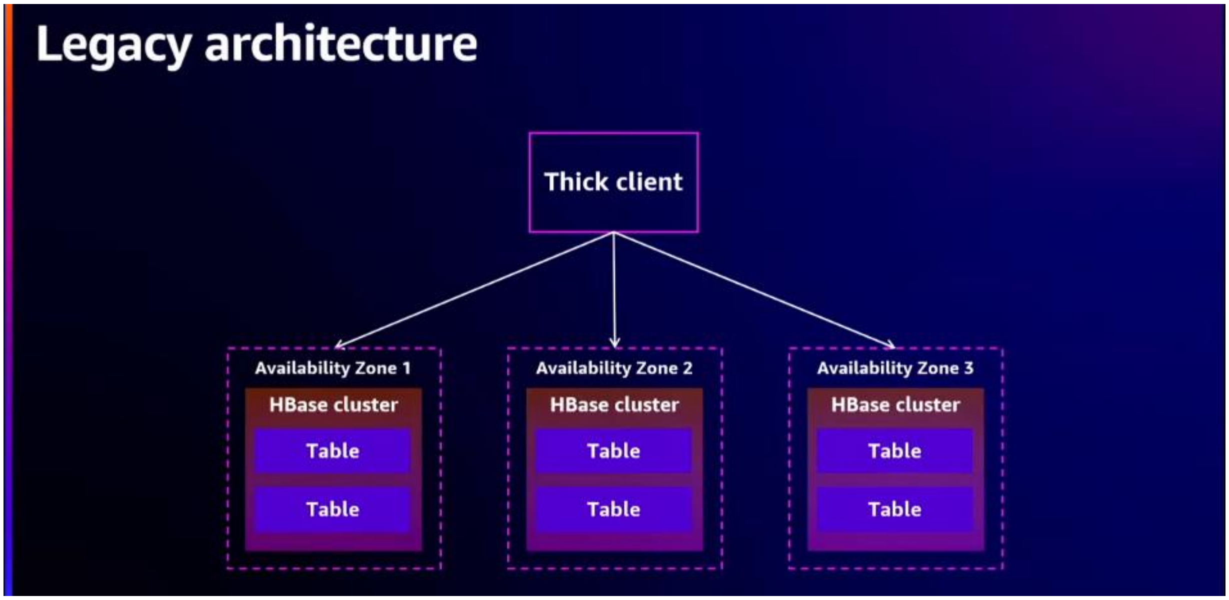
Ads measurement

4+PB DDB

100B+ events/ day

90+MM RCUs

5+MM WCUs



Legacy architecture

Thick client

Availability Zone 1

HBase cluster

Table

Table

Availability Zone 2

HBase cluster

Table

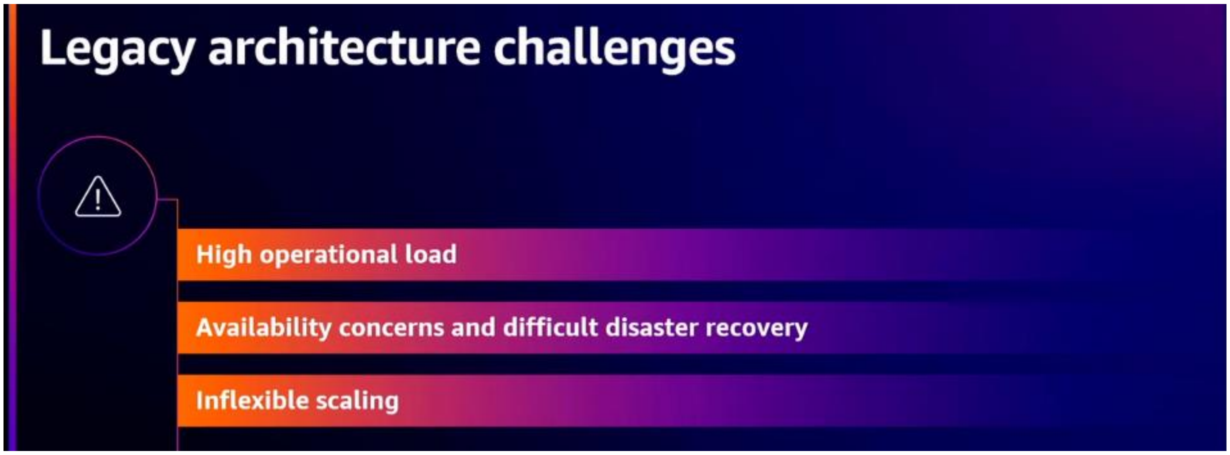
Table

Availability Zone 3

HBase cluster

Table

Table

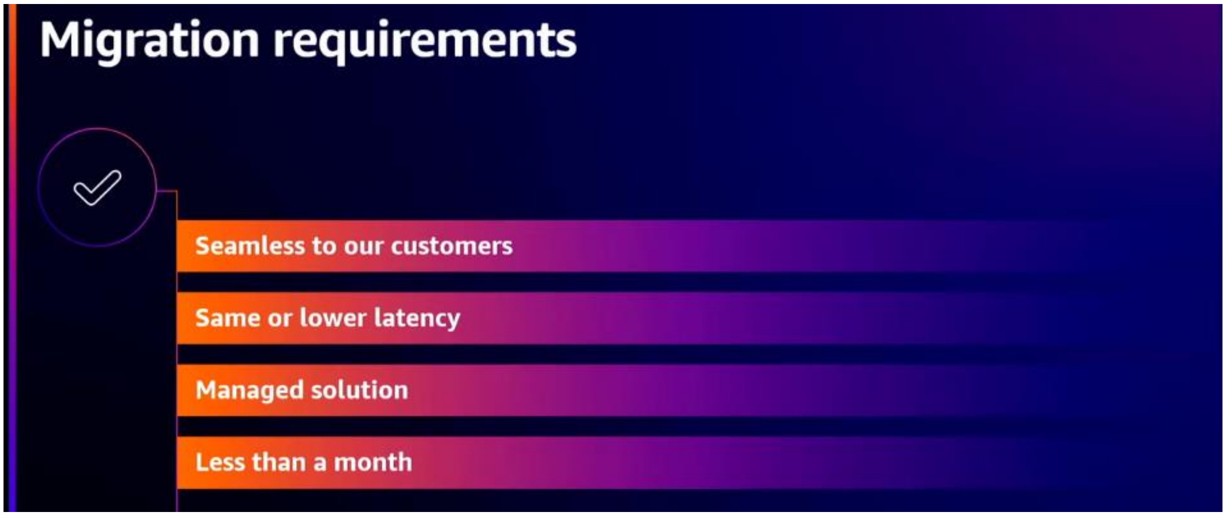


Legacy architecture challenges

High operational load

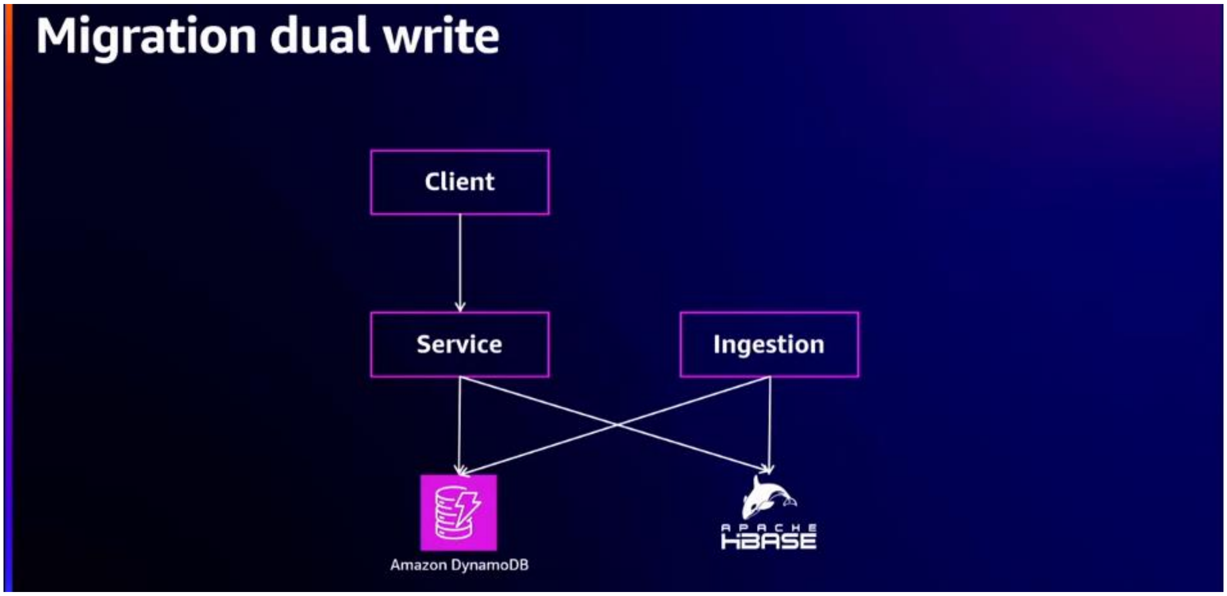
Availability concerns and difficult disaster recovery

Inflexible scaling



Migration requirements

- Seamless to our customers
- Same or lower latency
- Managed solution
- Less than a month



Migration dual write

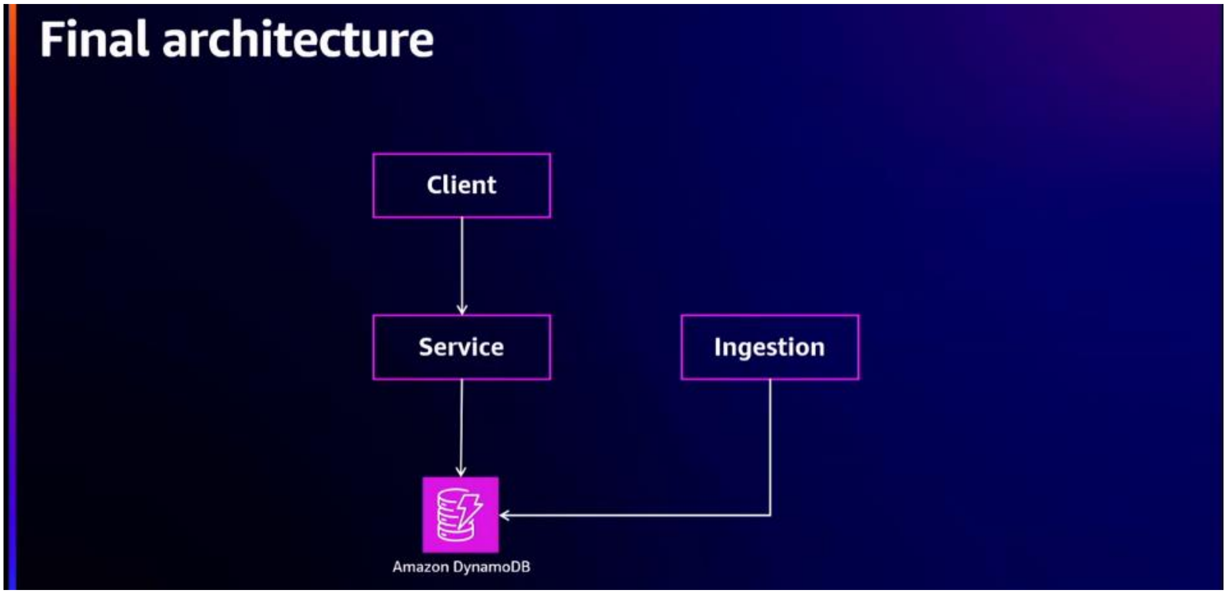
Client

Service

Ingestion

Amazon DynamoDB

APACHE HBASE



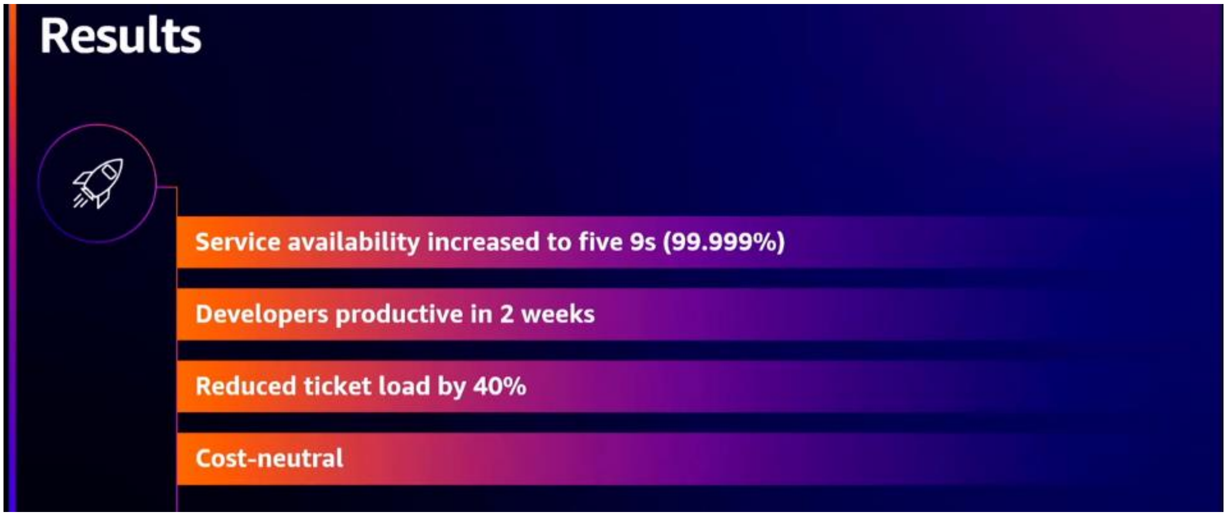
Final architecture

Client

Service

Ingestion

Amazon DynamoDB

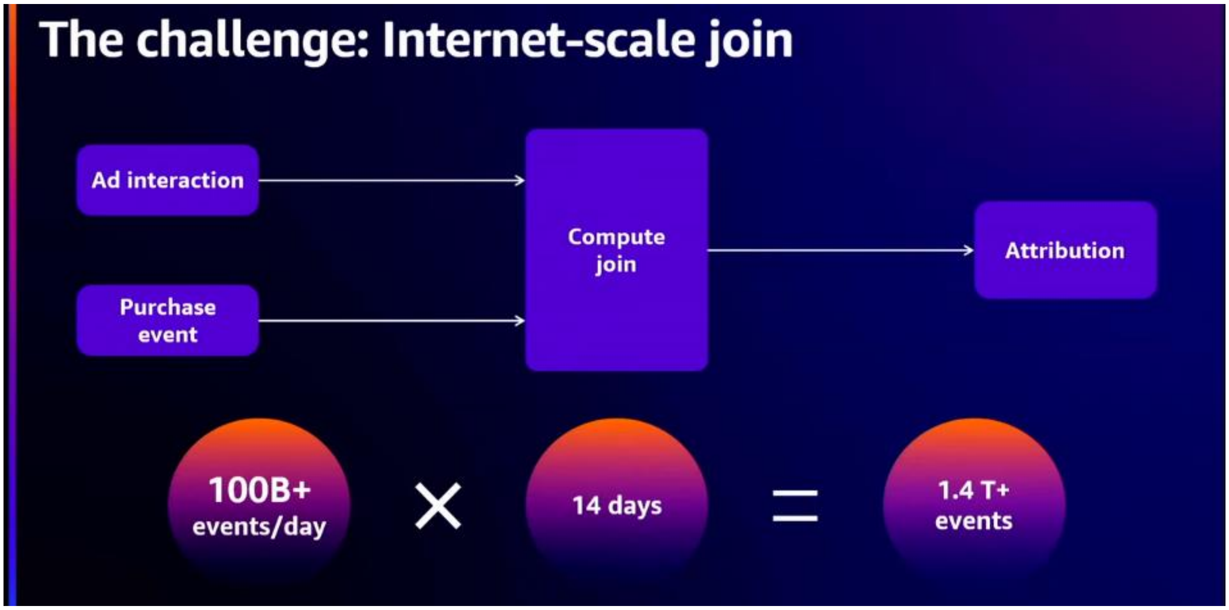


Results

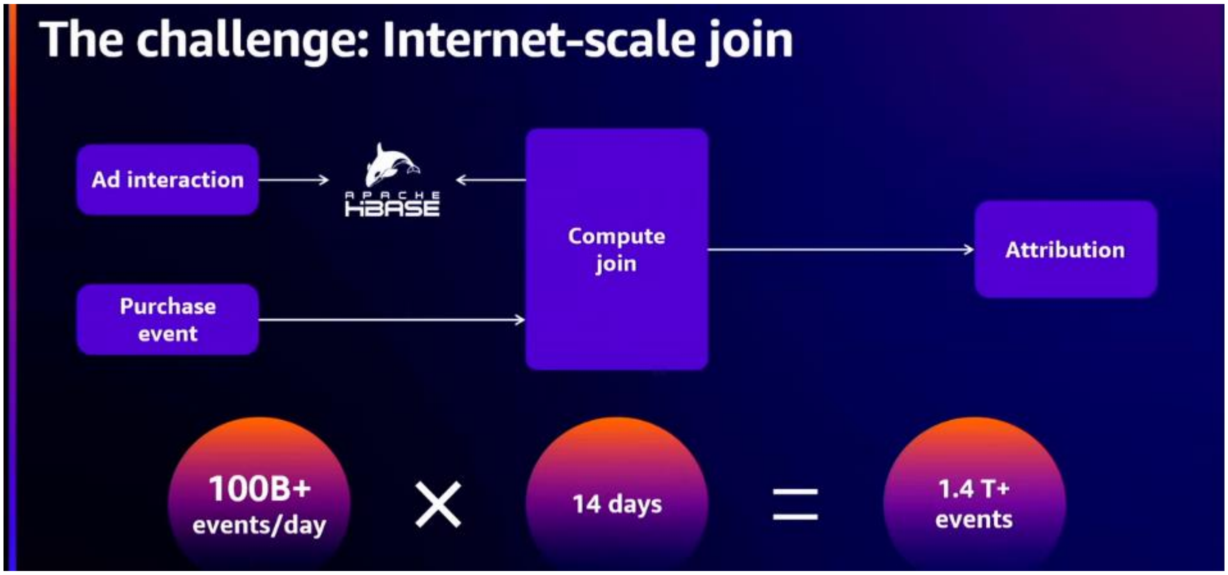
- Service availability increased to five 9s (99.999%)
- Developers productive in 2 weeks
- Reduced ticket load by 40%
- Cost-neutral

Principal Engineer

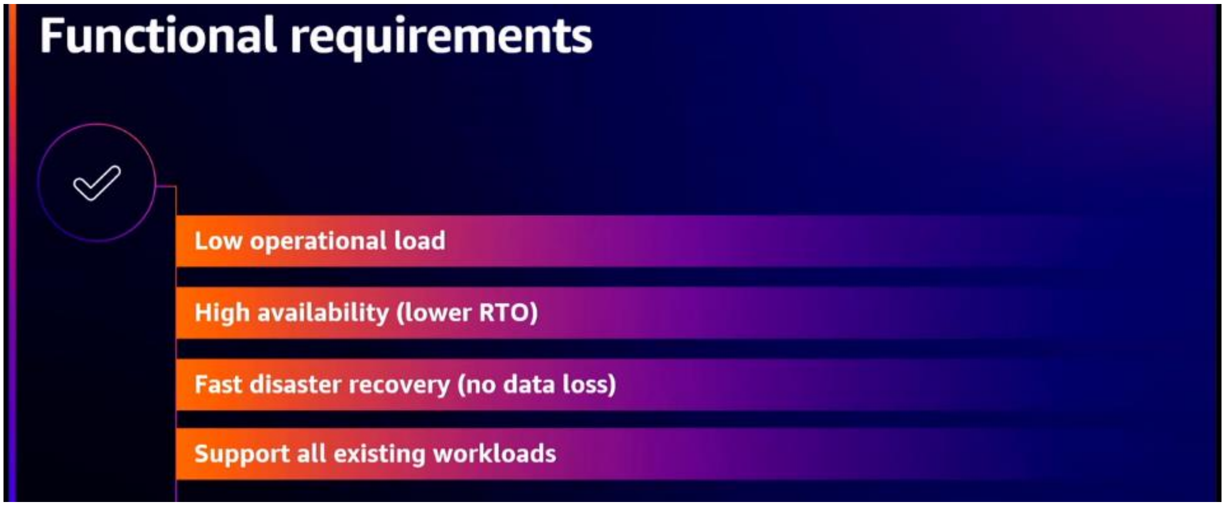
Rich Edwards



The challenge: Internet-scale join



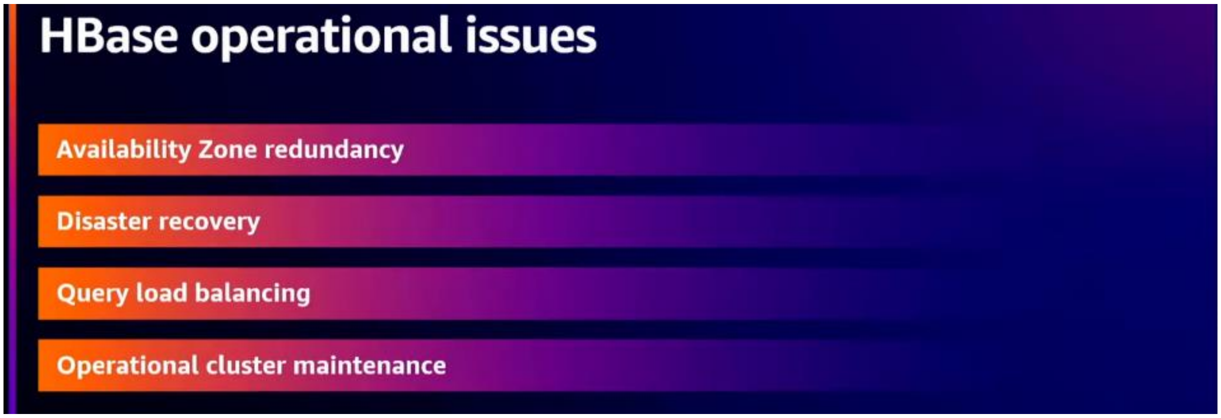
The challenge: Internet-scale join



Functional requirements

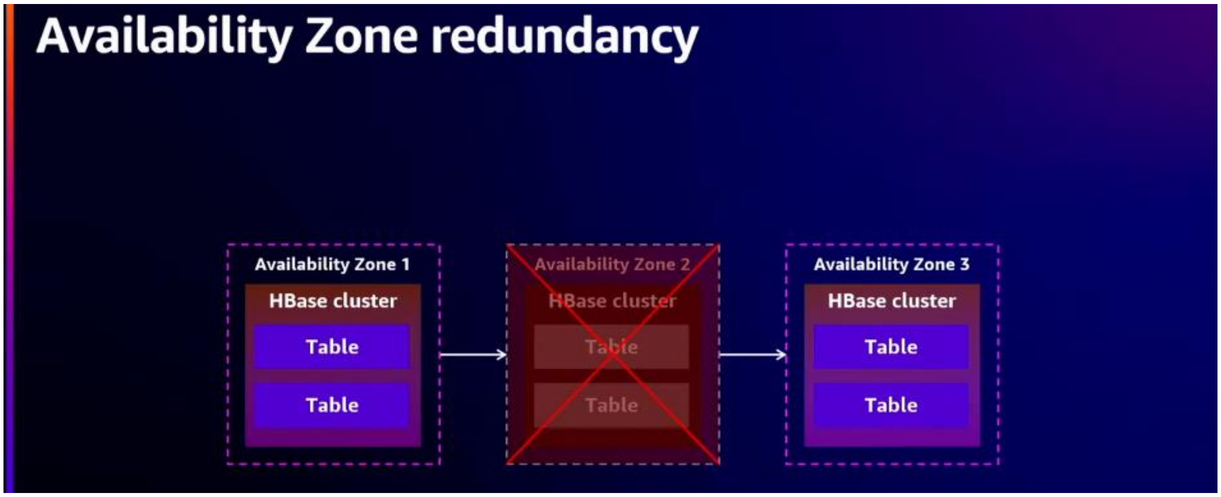
- Low operational load
- High availability (lower RTO)
- Fast disaster recovery (no data loss)
- Support all existing workloads

Our solution



HBase operational issues

- Availability Zone redundancy
- Disaster recovery
- Query load balancing
- Operational cluster maintenance

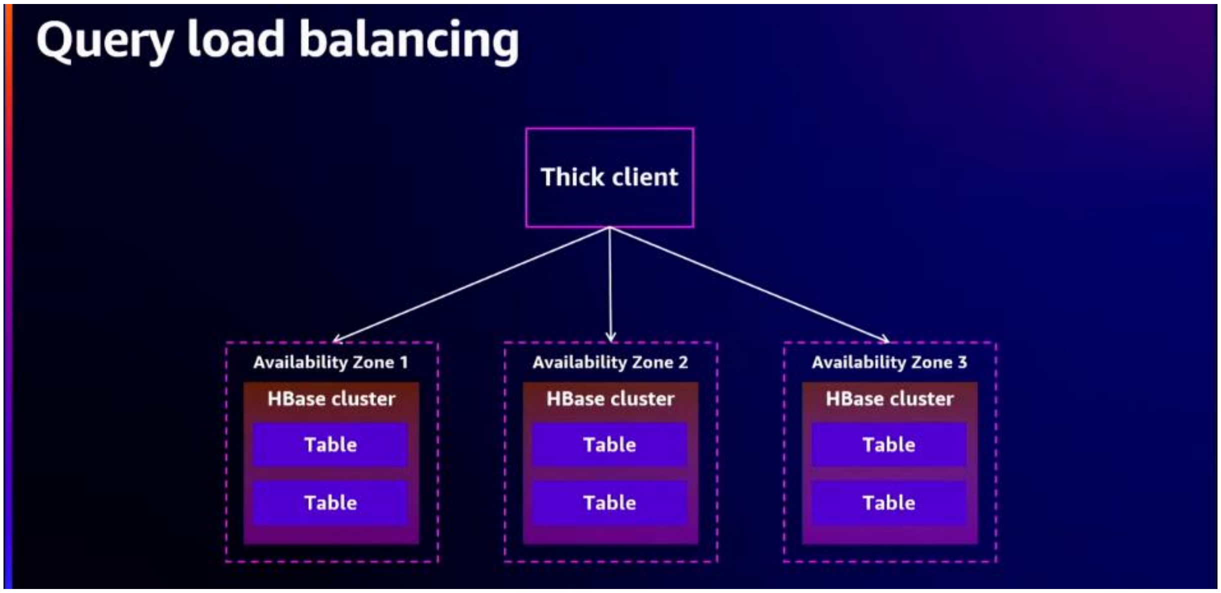


Availability Zone redundancy

Availability Zone 1
HBase cluster
Table
Table

Availability Zone 2
HBase cluster
Table
Table

Availability Zone 3
HBase cluster
Table
Table



Query load balancing

Thick client

Availability Zone 1

HBase cluster

Table

Table

Availability Zone 2

HBase cluster

Table

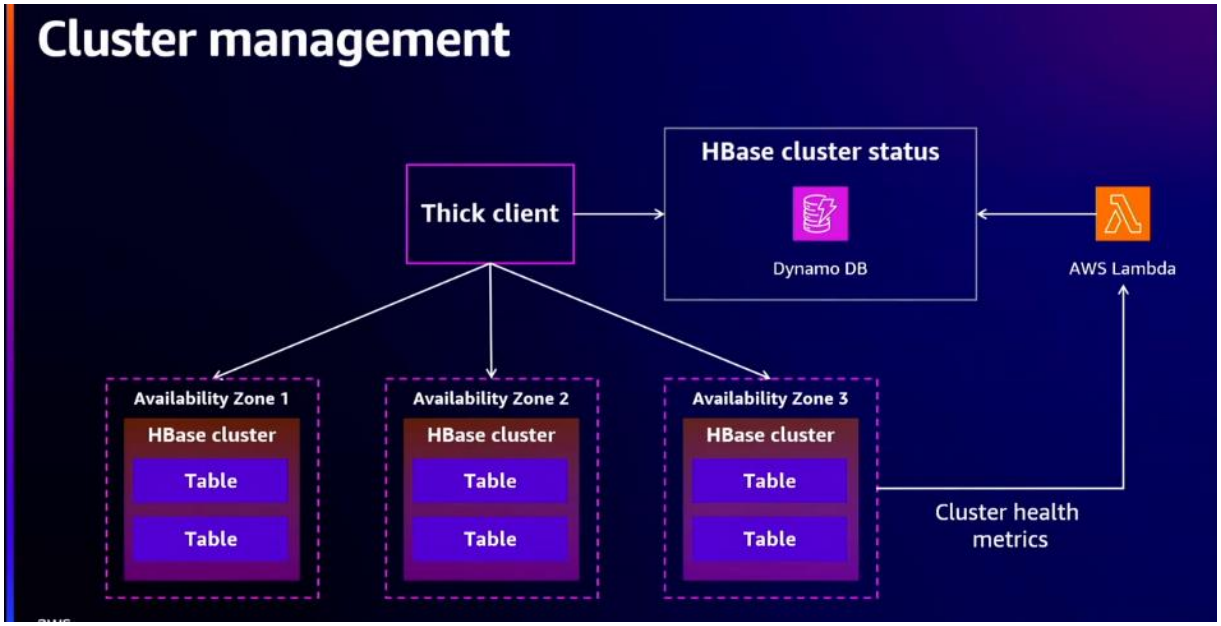
Table

Availability Zone 3

HBase cluster

Table

Table



Cluster management

Thick client

HBase cluster status

Dynamo DB

AWS Lambda

Availability Zone 1

HBase cluster

Table

Table

Availability Zone 2

HBase cluster

Table

Table

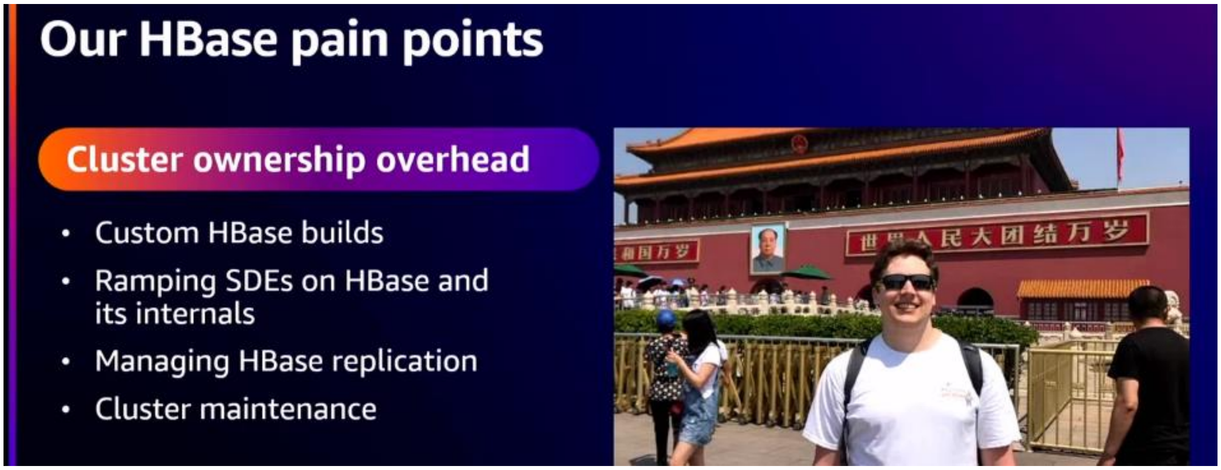
Availability Zone 3

HBase cluster

Table

Table

Cluster health metrics

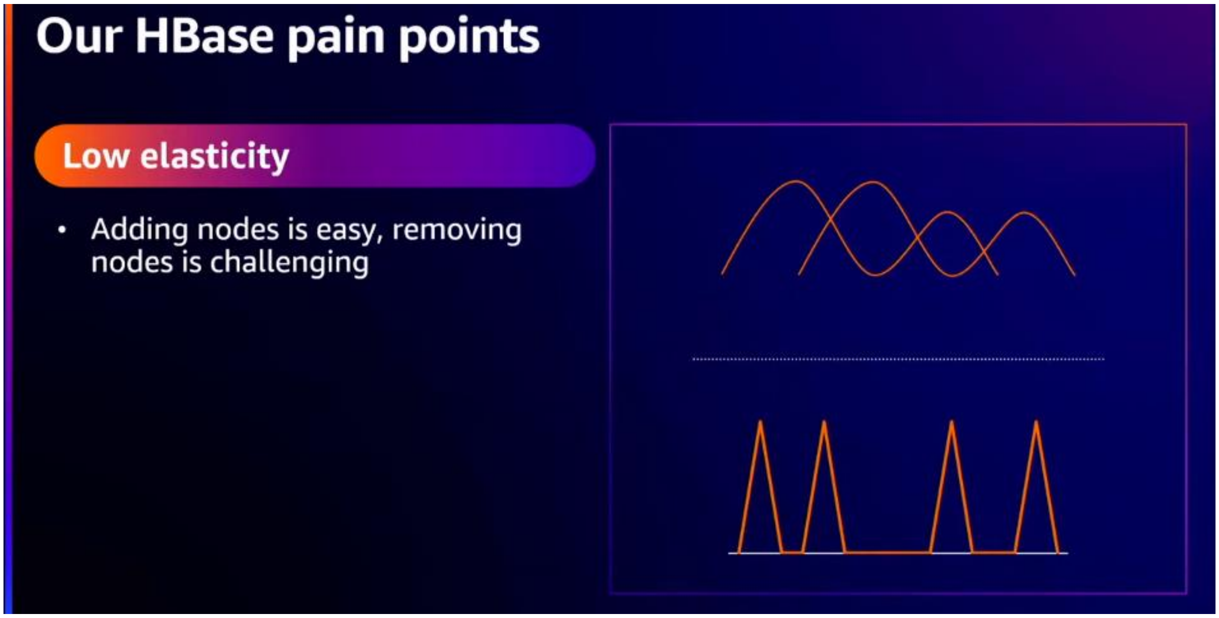


Our HBase pain points

Cluster ownership overhead

- Custom HBase builds
- Ramping SDEs on HBase and its internals
- Managing HBase replication
- Cluster maintenance

和国万岁
世界人民大团结万岁



Our HBase pain points

Low elasticity

- Adding nodes is easy, removing nodes is challenging

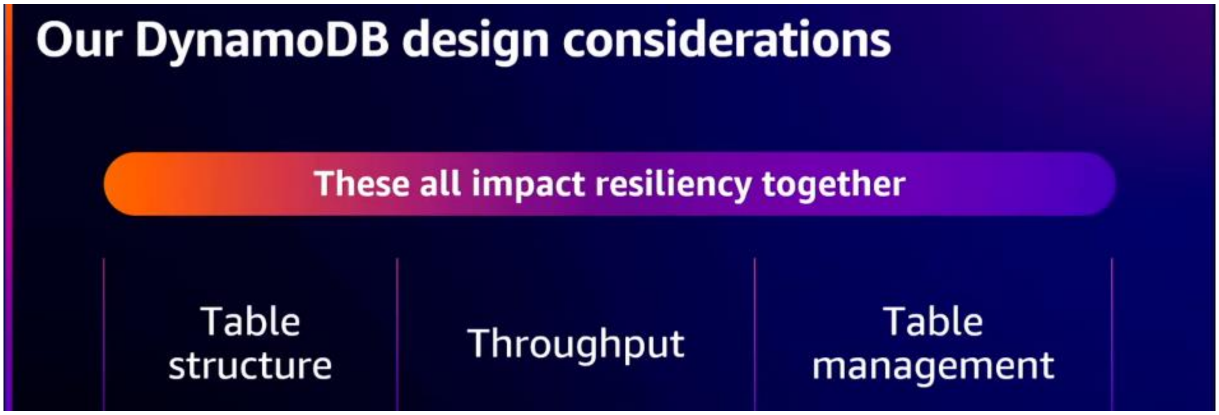
DynamoDB solved our pain points

Fully managed

- No OS/security patching
- No redundancy management

Elastic

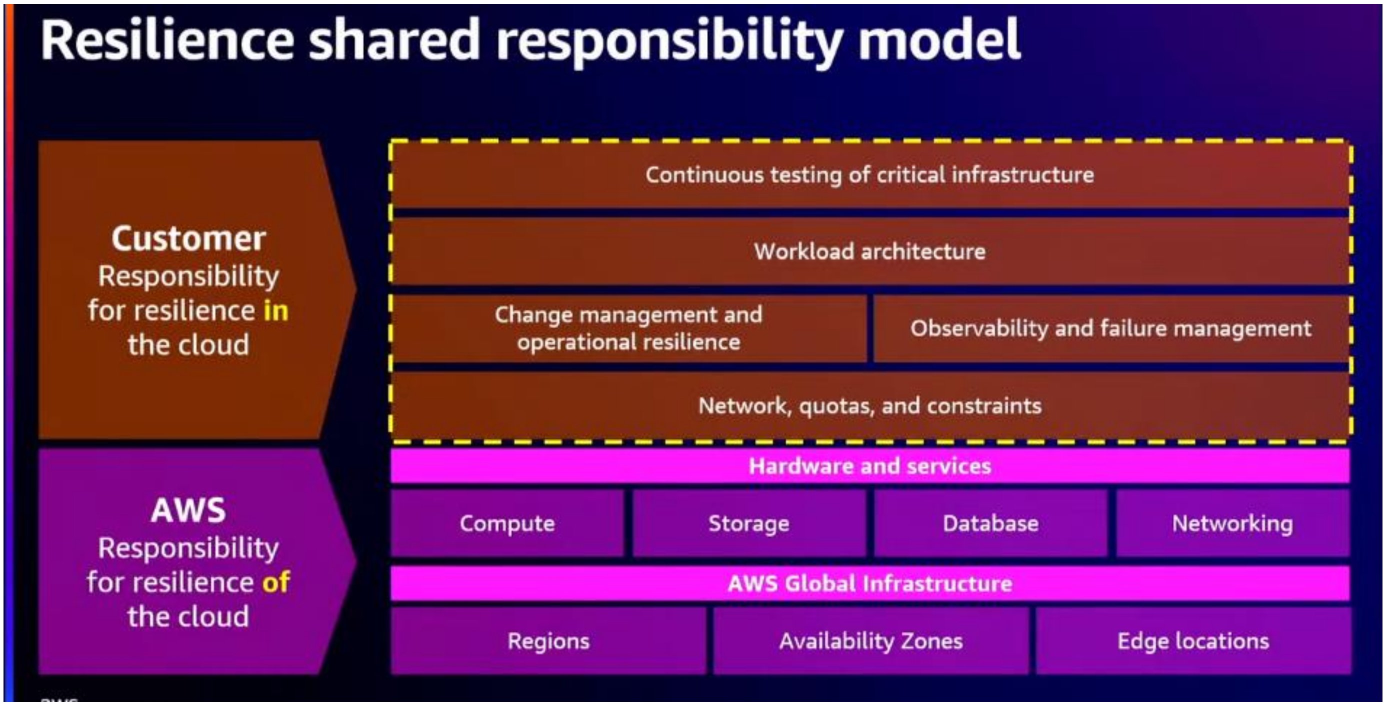
- Full auto scaling



Our DynamoDB design considerations

These all impact resiliency together

Table structure	Throughput	Table management



Resilience shared responsibility model

Customer Responsibility for resilience **in** the cloud

- Continuous testing of critical infrastructure
- Workload architecture
- Change management and operational resilience
- Observability and failure management
- Network, quotas, and constraints

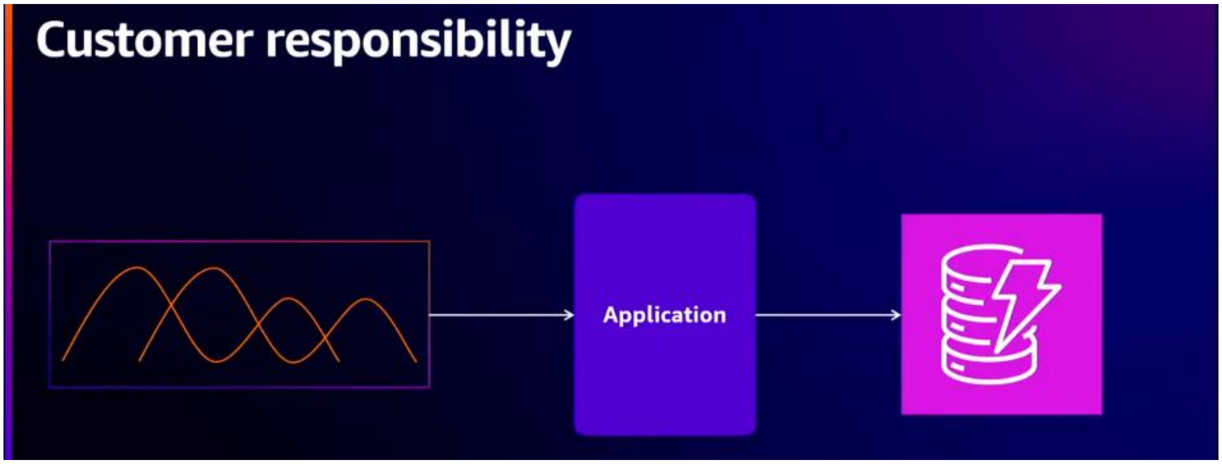
AWS Responsibility for resilience **of** the cloud

Hardware and services

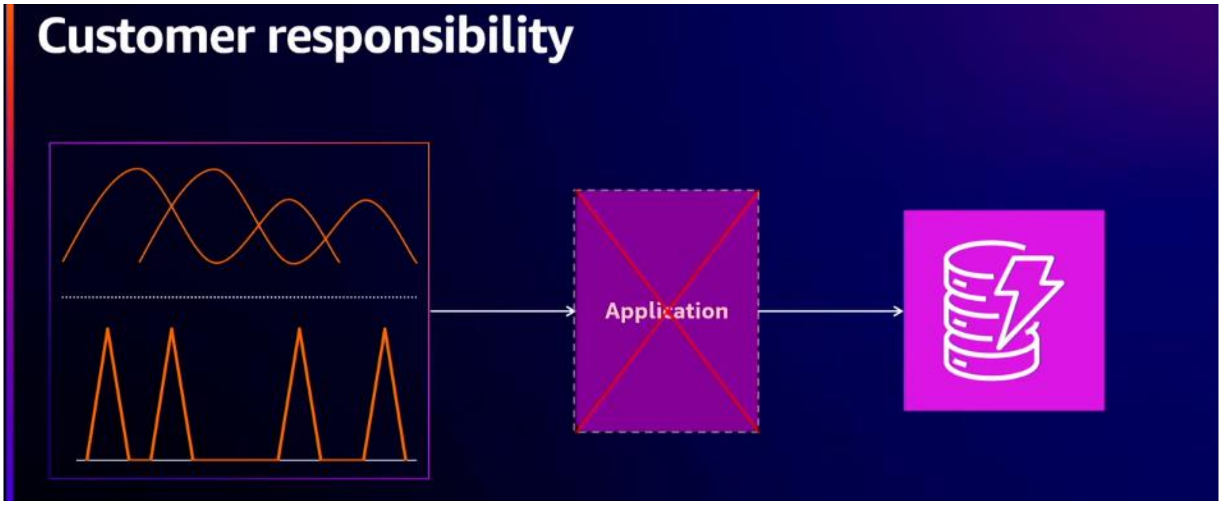
- Compute
- Storage
- Database
- Networking

AWS Global Infrastructure

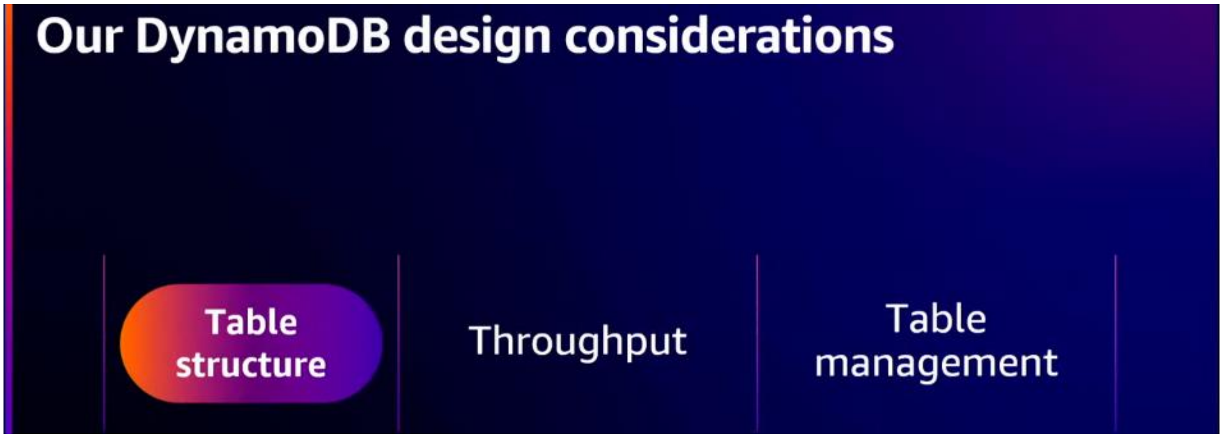
- Regions
- Availability Zones
- Edge locations



Customer responsibility

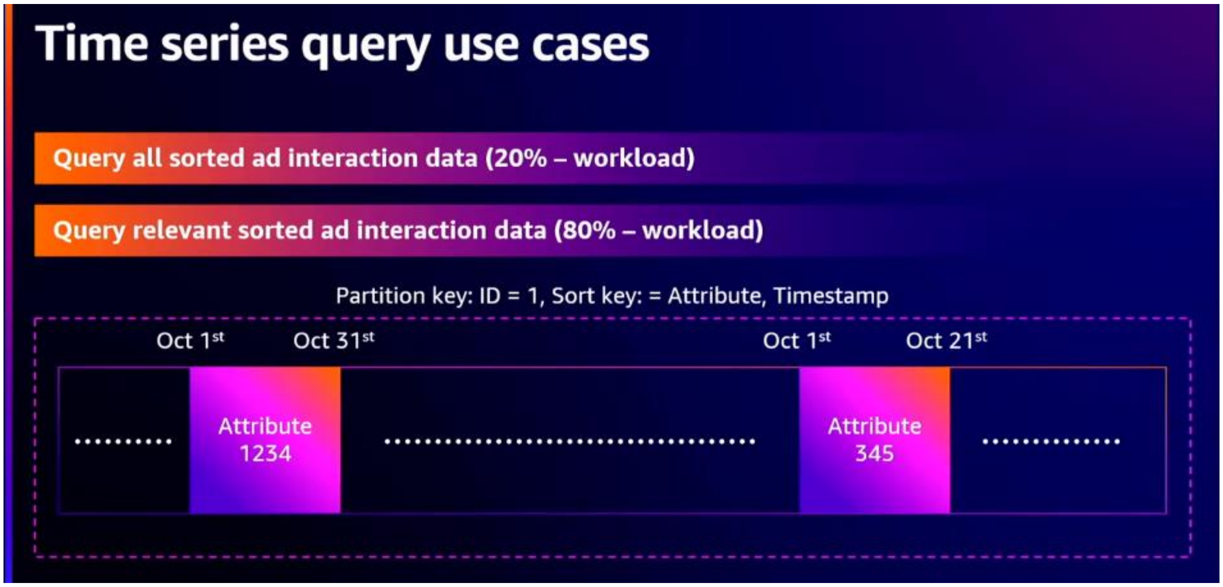


Customer responsibility



Our DynamoDB design considerations

Table structure | Throughput | Table management



Time series query use cases

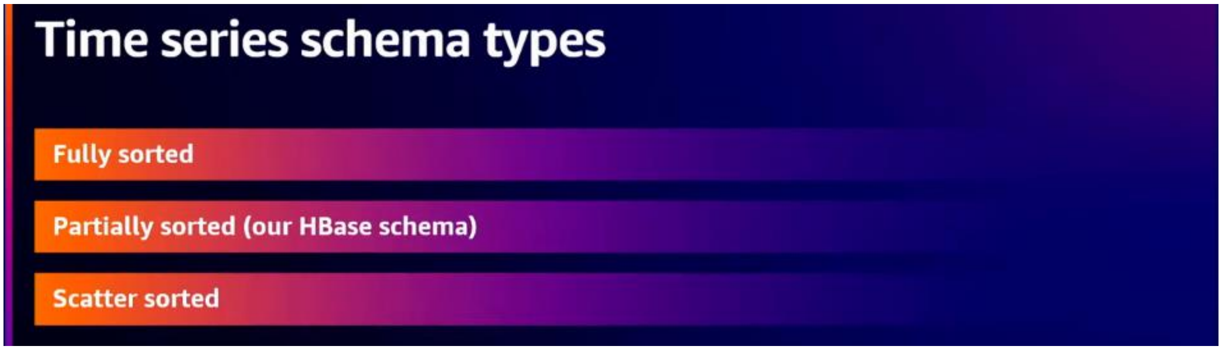
Query all sorted ad interaction data (20% – workload)

Query relevant sorted ad interaction data (80% – workload)

Partition key: ID = 1, Sort key: = Attribute, Timestamp

Oct 1st Oct 31st Oct 1st Oct 21st

.... Attribute 1234 .... Attribute 345 ....



Time series schema types

Fully sorted

Partially sorted (our HBase schema)

Scatter sorted

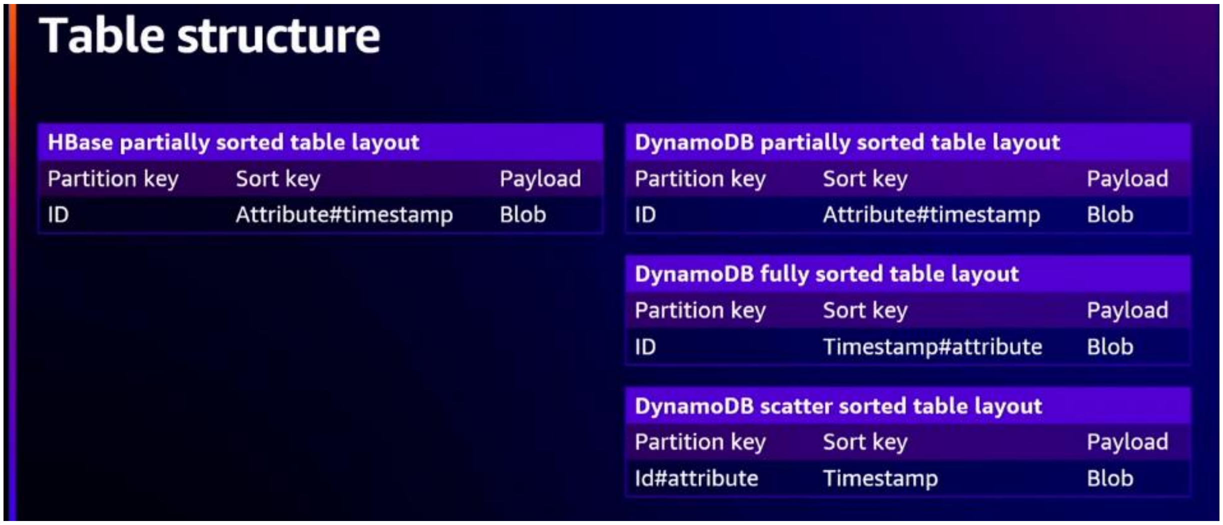


Table structure

HBase partially sorted table layout		
Partition key	Sort key	Payload
ID	Attribute#timestamp	Blob

DynamoDB partially sorted table layout		
Partition key	Sort key	Payload
ID	Attribute#timestamp	Blob

DynamoDB fully sorted table layout		
Partition key	Sort key	Payload
ID	Timestamp#attribute	Blob

DynamoDB scatter sorted table layout		
Partition key	Sort key	Payload
Id#attribute	Timestamp	Blob

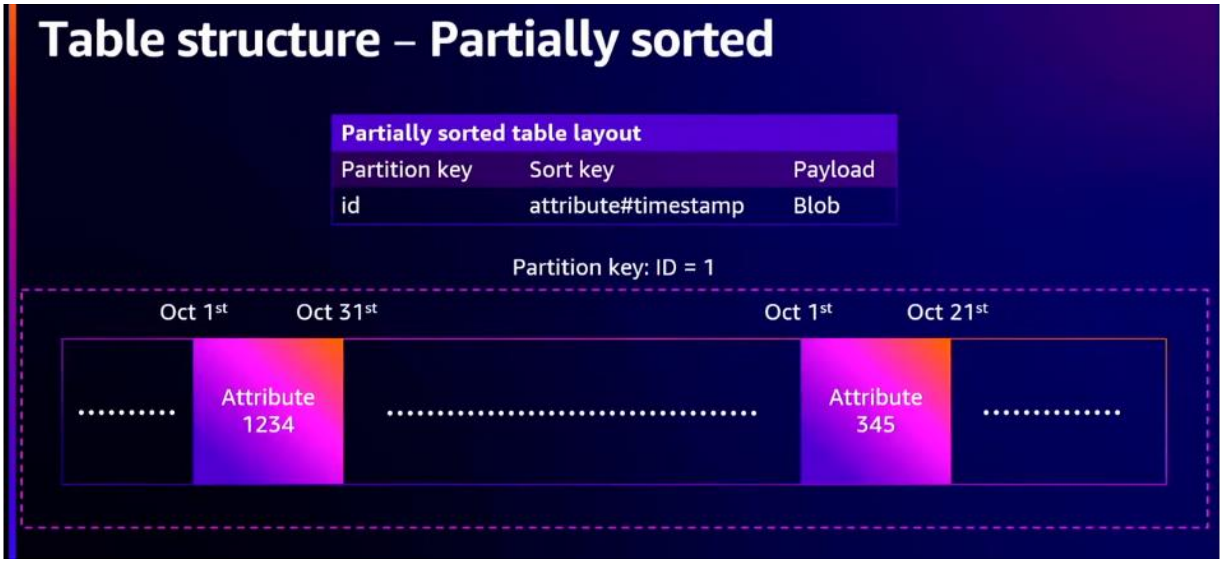


Table structure – Partially sorted

Partially sorted table layout		
Partition key	Sort key	Payload
id	attribute#timestamp	Blob

Partition key: ID = 1

Oct 1st Oct 31st Oct 1st Oct 21st

Attribute 1234

Attribute 345

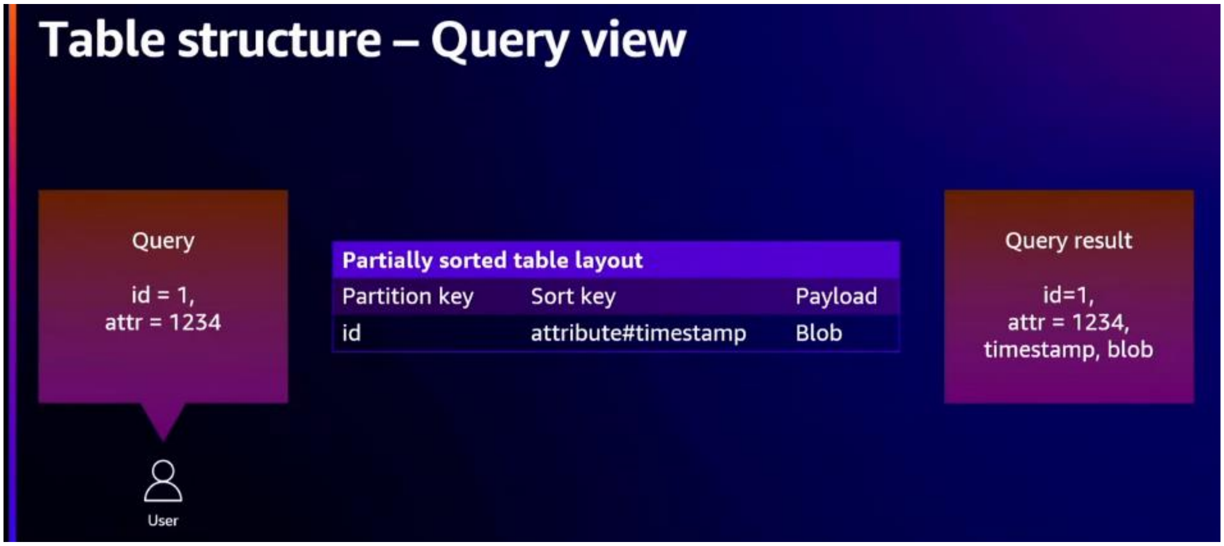


Table structure – Query view

Query

id = 1,
attr = 1234

User

Partially sorted table layout		
Partition key	Sort key	Payload
id	attribute#timestamp	Blob

Query result

id=1,
attr = 1234,
timestamp, blob

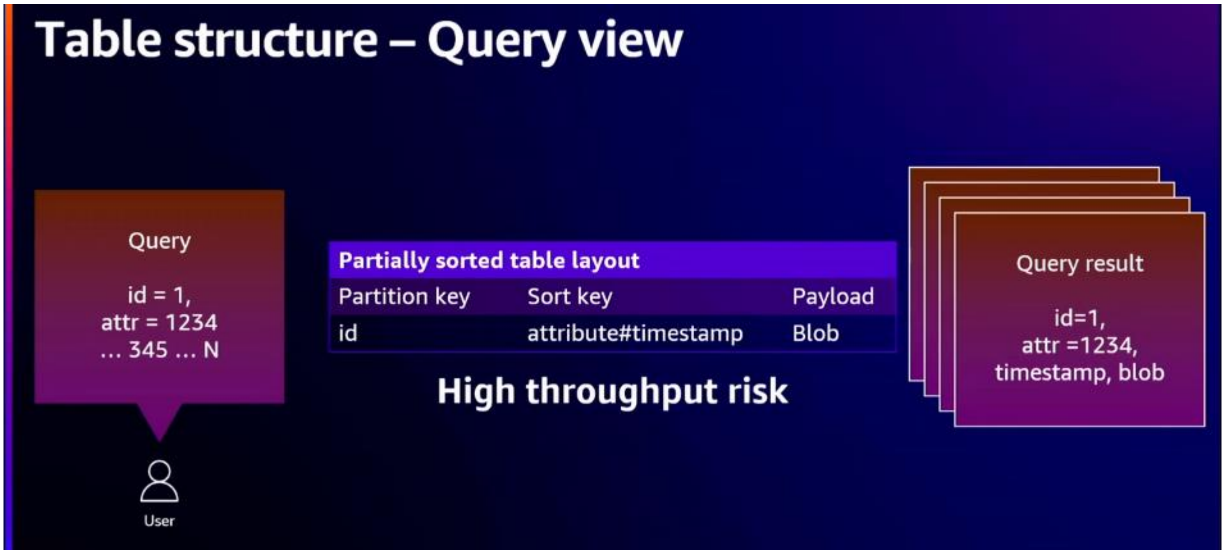


Table structure – Query view

Query

id = 1,
attr = 1234
… 345 … N

User

Partially sorted table layout		
Partition key	Sort key	Payload
id	attribute#timestamp	Blob

High throughput risk

Query result

id=1,
attr =1234,
timestamp, blob

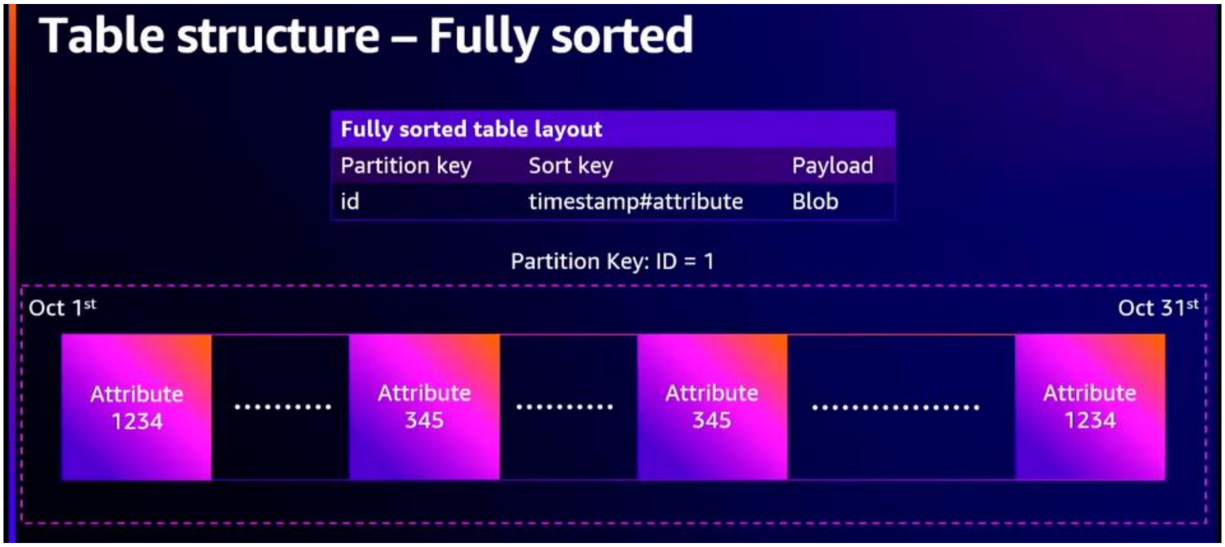


Table structure – Fully sorted

Fully sorted table layout		
Partition key	Sort key	Payload
id	timestamp#attribute	Blob

Partition Key: ID = 1

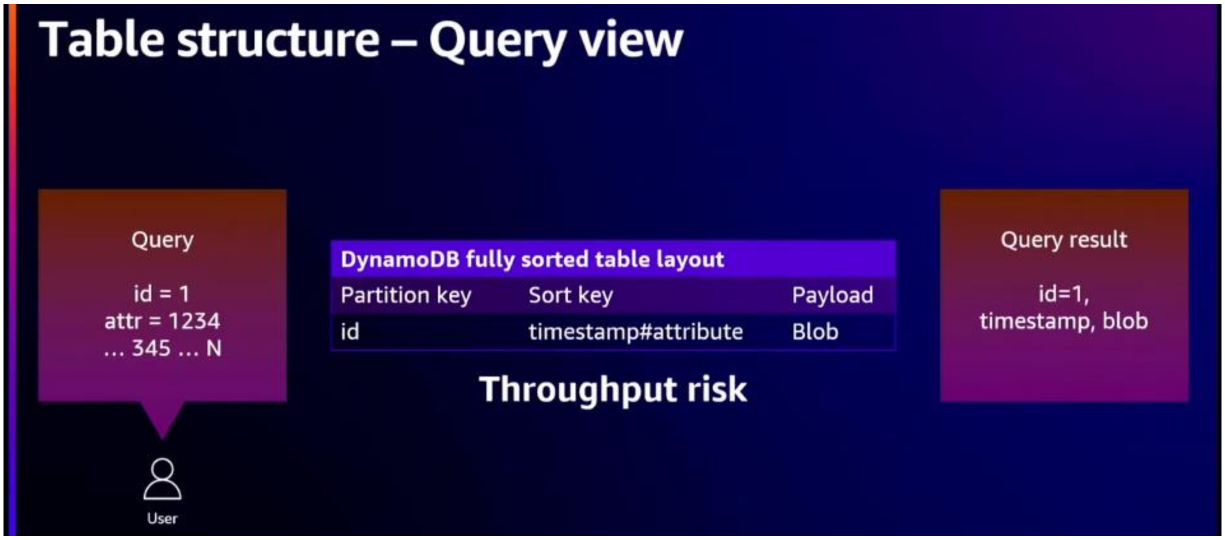


Table structure – Query view

Query

id = 1
attr = 1234
… 345 … N

User

DynamoDB fully sorted table layout		
Partition key	Sort key	Payload
id	timestamp#attribute	Blob

Throughput risk

Query result

id=1,
timestamp, blob

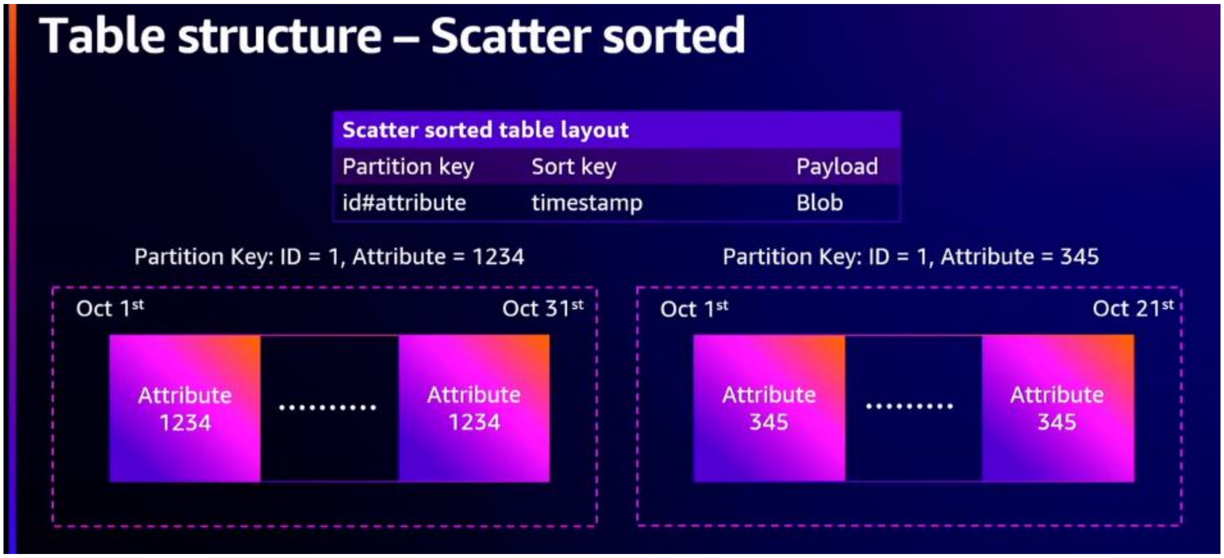


Table structure – Scatter sorted

Scatter sorted table layout		
Partition key	Sort key	Payload
id#attribute	timestamp	Blob

Partition Key: ID = 1, Attribute = 1234

Oct 1st Oct 31st

Attribute 1234 .... Attribute 1234

Partition Key: ID = 1, Attribute = 345

Oct 1st Oct 21st

Attribute 345 .... Attribute 345

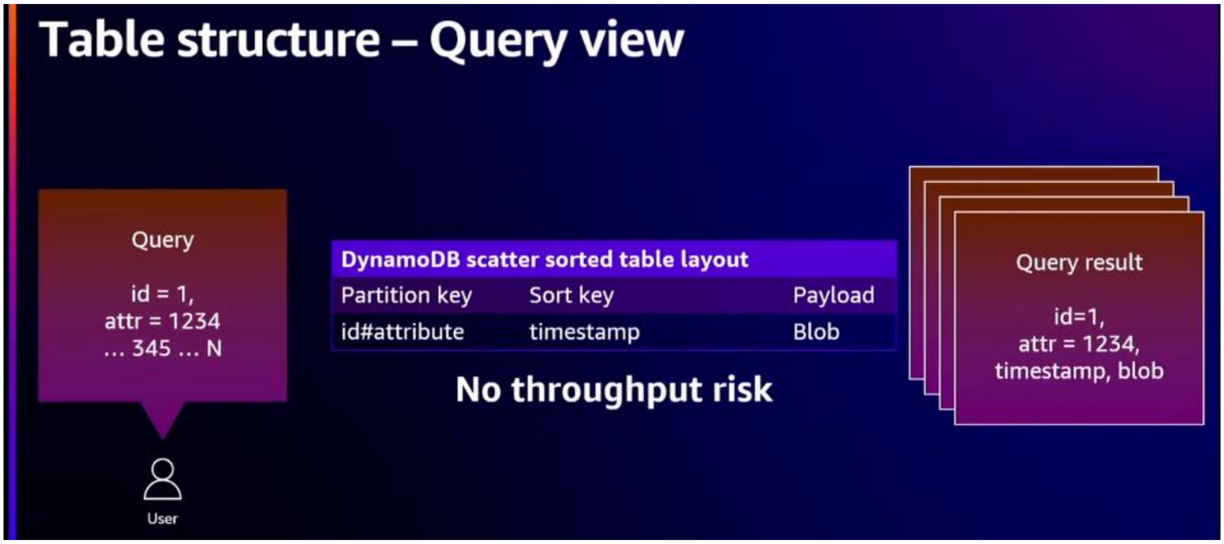


Table structure – Query view

Query

id = 1,
attr = 1234
... 345 ... N

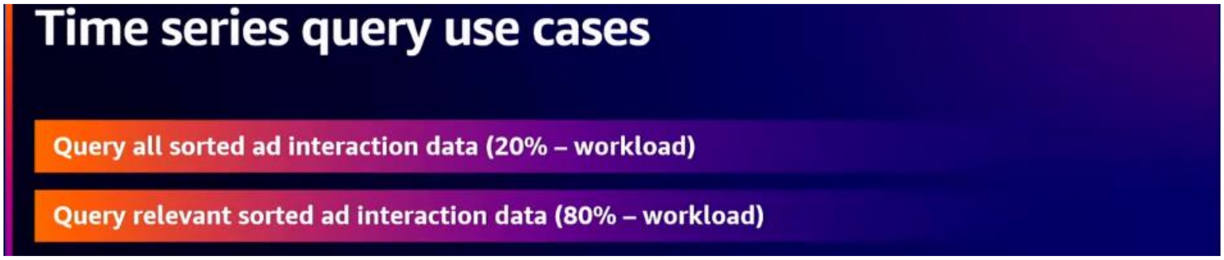
User

DynamoDB scatter sorted table layout		
Partition key	Sort key	Payload
id#attribute	timestamp	Blob

No throughput risk

Query result

id=1,
attr = 1234,
timestamp, blob



Time series query use cases

Query all sorted ad interaction data (20% – workload)

Query relevant sorted ad interaction data (80% – workload)

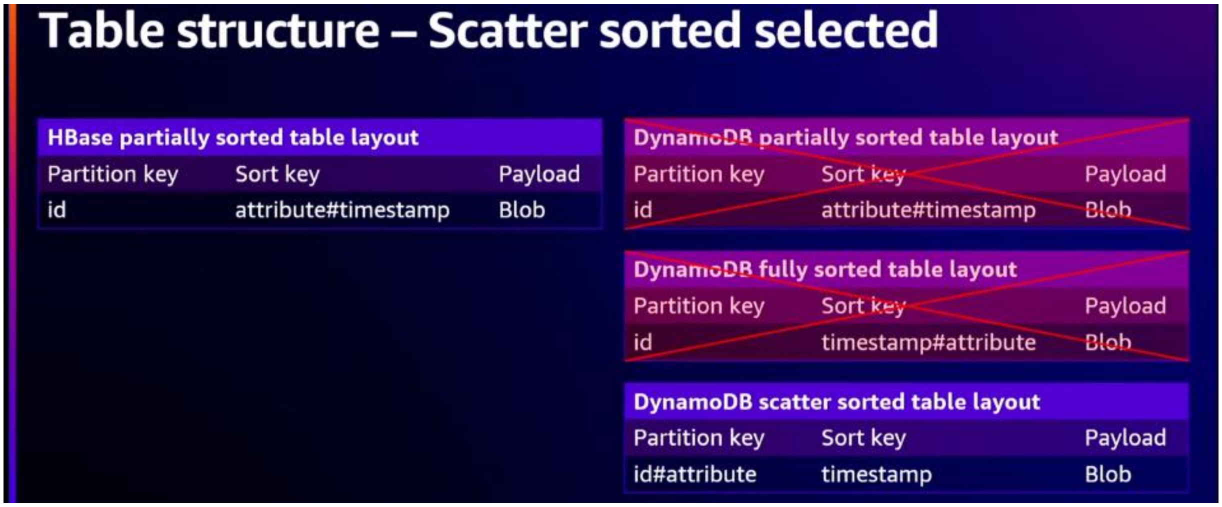


Table structure – Scatter sorted selected

HBase partially sorted table layout		
Partition key	Sort key	Payload
id	attribute#timestamp	Blob

~~DynamoDB partially sorted table layout~~		
~~Partition key~~	~~Sort key~~	~~Payload~~
~~id~~	~~attribute#timestamp~~	~~Blob~~

~~DynamoDB fully sorted table layout~~		
~~Partition key~~	~~Sort key~~	~~Payload~~
~~id~~	~~timestamp#attribute~~	~~Blob~~

DynamoDB scatter sorted table layout		
Partition key	Sort key	Payload
id#attribute	timestamp	Blob

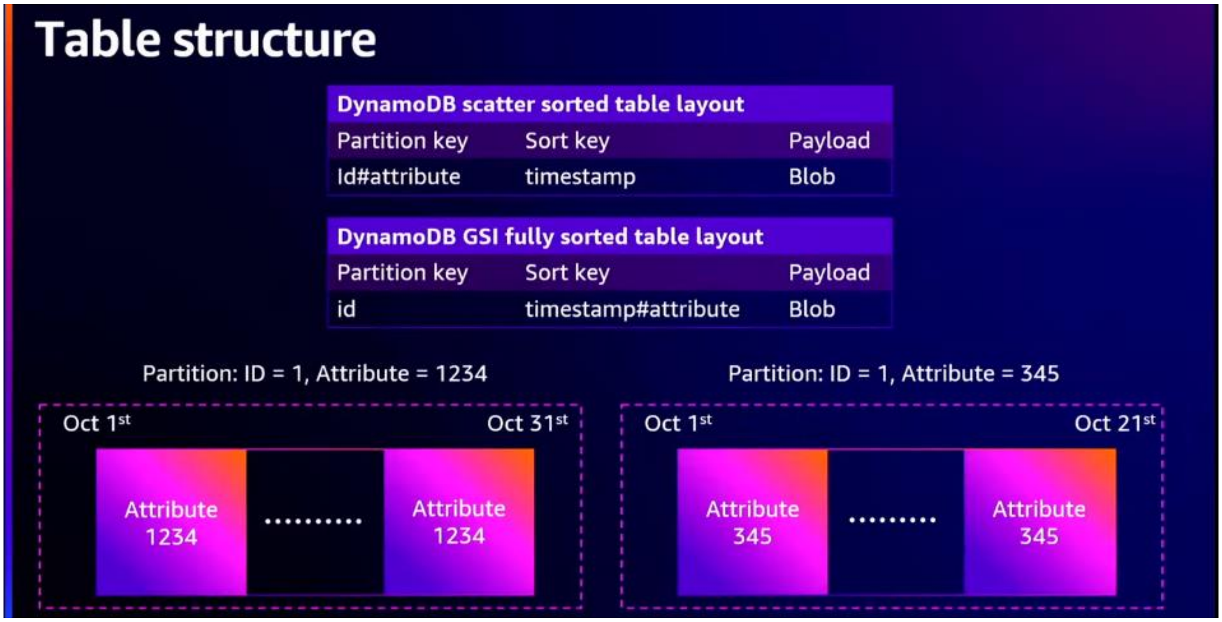


Table structure

DynamoDB scatter sorted table layout		
Partition key	Sort key	Payload
Id#attribute	timestamp	Blob

DynamoDB GSI fully sorted table layout		
Partition key	Sort key	Payload
id	timestamp#attribute	Blob

Partition: ID = 1, Attribute = 1234

Oct 1st — Oct 31st

Attribute 1234 .... Attribute 1234

Partition: ID = 1, Attribute = 345

Oct 1st — Oct 21st

Attribute 345 .... Attribute 345

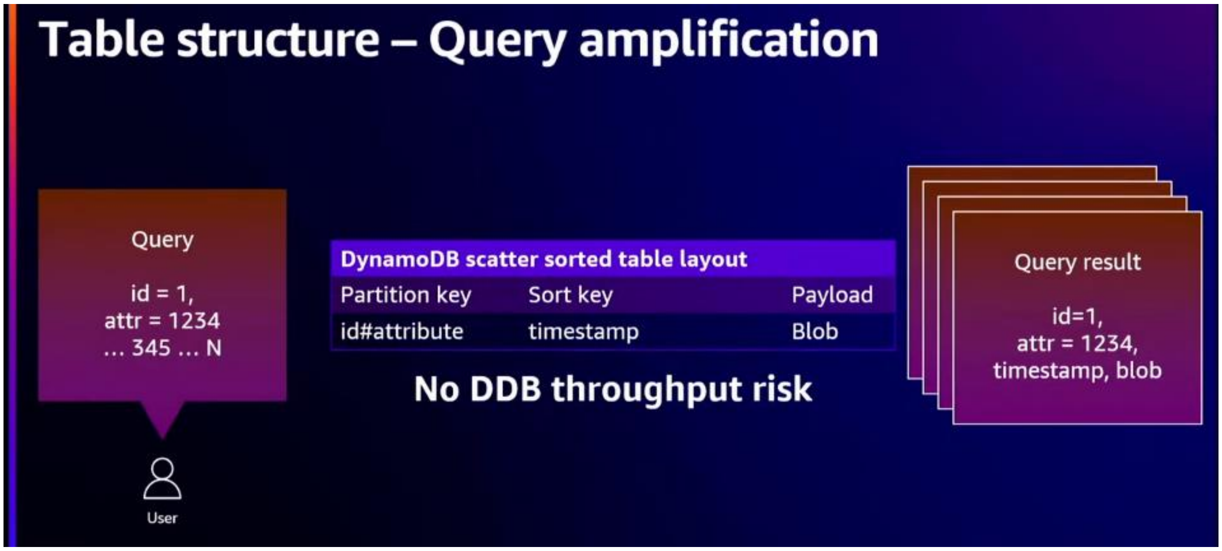


Table structure – Query amplification

Query

id = 1,
attr = 1234
… 345 … N

User

DynamoDB scatter sorted table layout		
Partition key	Sort key	Payload
id#attribute	timestamp	Blob

No DDB throughput risk

Query result

id=1,
attr = 1234,
timestamp, blob

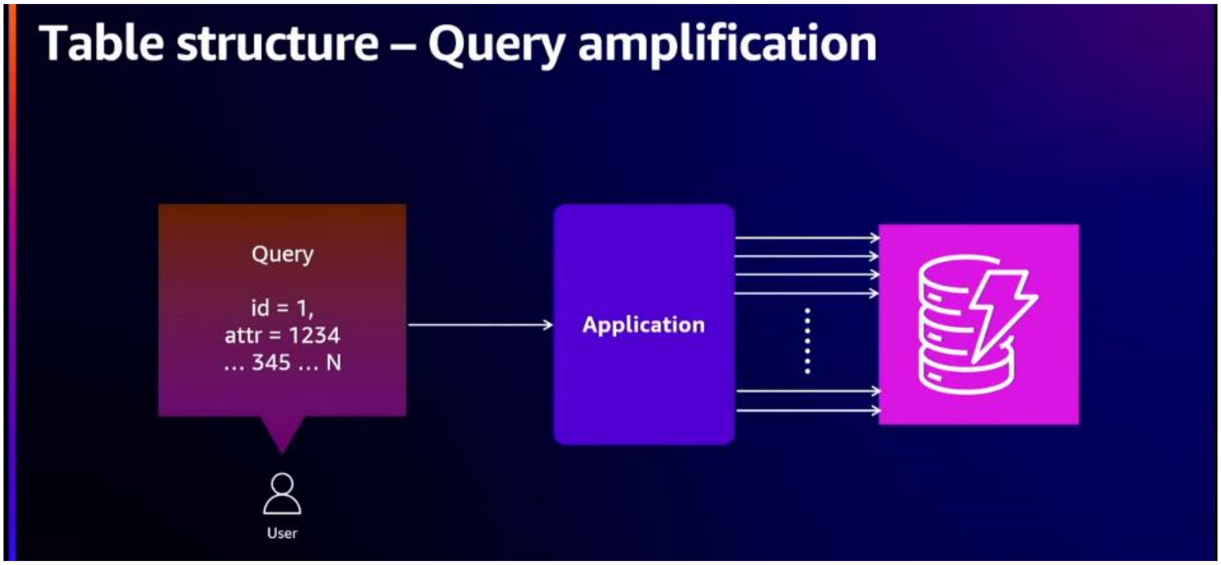


Table structure – Query amplification

Query

id = 1,
attr = 1234
… 345 … N

User

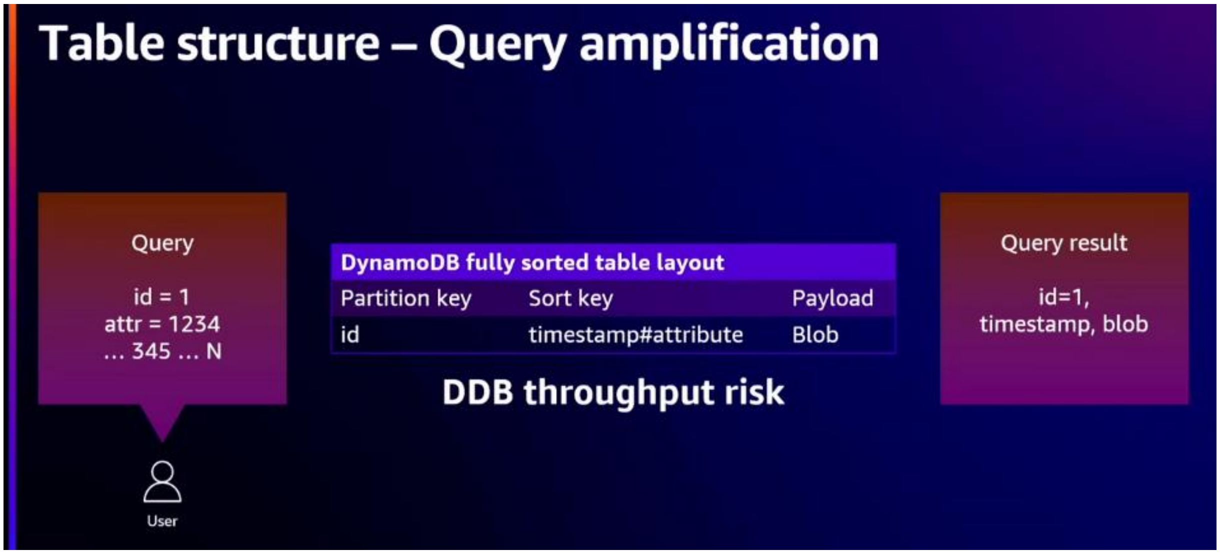


Table structure – Query amplification

Query

id = 1
attr = 1234
… 345 … N

DynamoDB fully sorted table layout		
Partition key	Sort key	Payload
id	timestamp#attribute	Blob

DDB throughput risk

Query result

id=1,
timestamp, blob

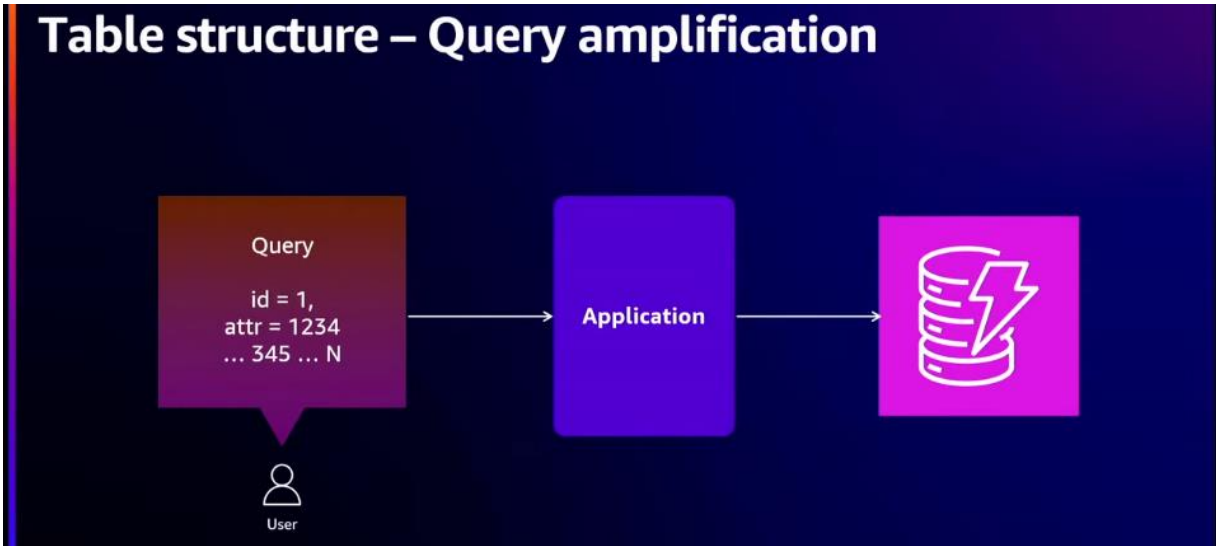


Table structure – Query amplification

Query

id = 1,
attr = 1234
… 345 … N

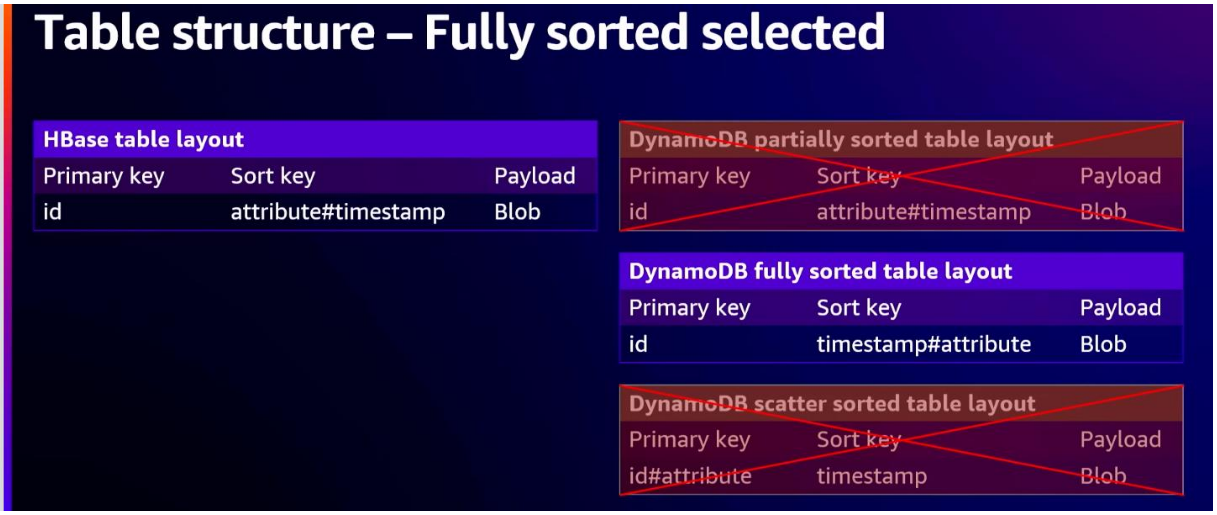


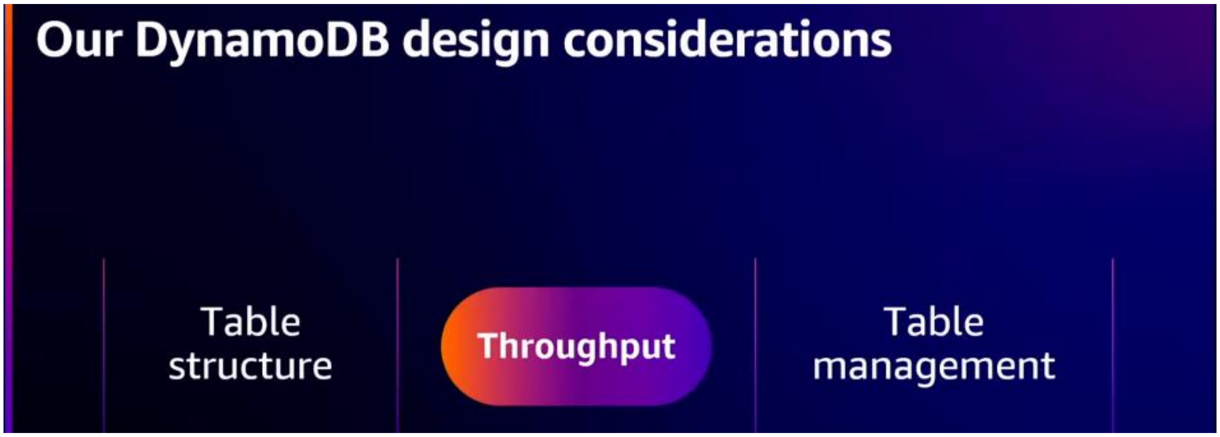
Table structure – Fully sorted selected

HBase table layout		
Primary key	Sort key	Payload
id	attribute#timestamp	Blob

~~DynamoDB partially sorted table layout~~		
~~Primary key~~	~~Sort key~~	~~Payload~~
~~id~~	~~attribute#timestamp~~	~~Blob~~

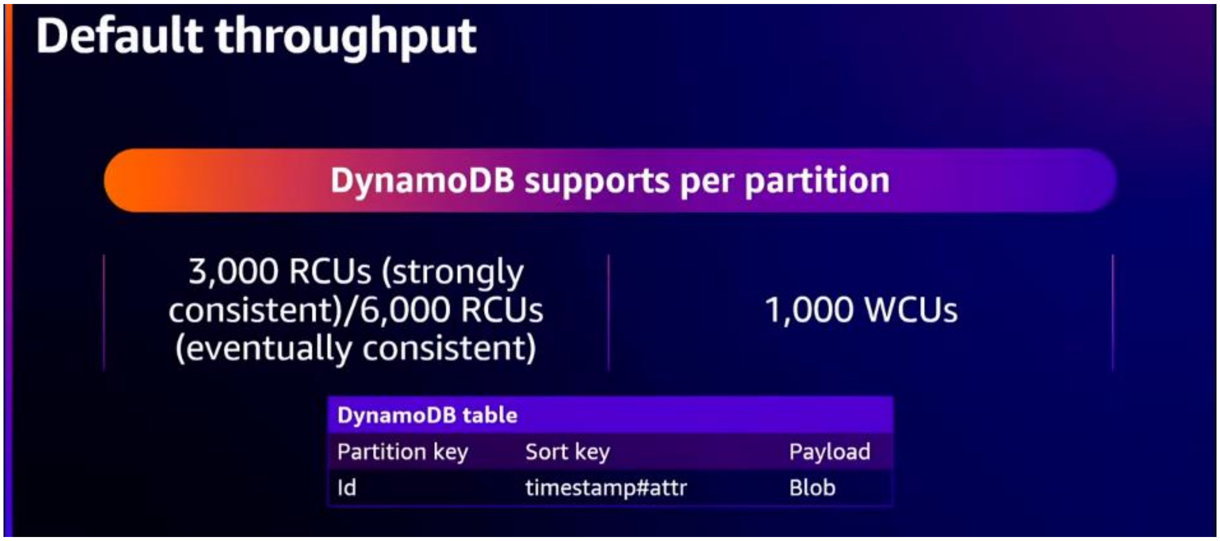
DynamoDB fully sorted table layout		
Primary key	Sort key	Payload
id	timestamp#attribute	Blob

~~DynamoDB scatter sorted table layout~~		
~~Primary key~~	~~Sort key~~	~~Payload~~
~~id#attribute~~	~~timestamp~~	~~Blob~~



Our DynamoDB design considerations

Table structure | **Throughput** | Table management



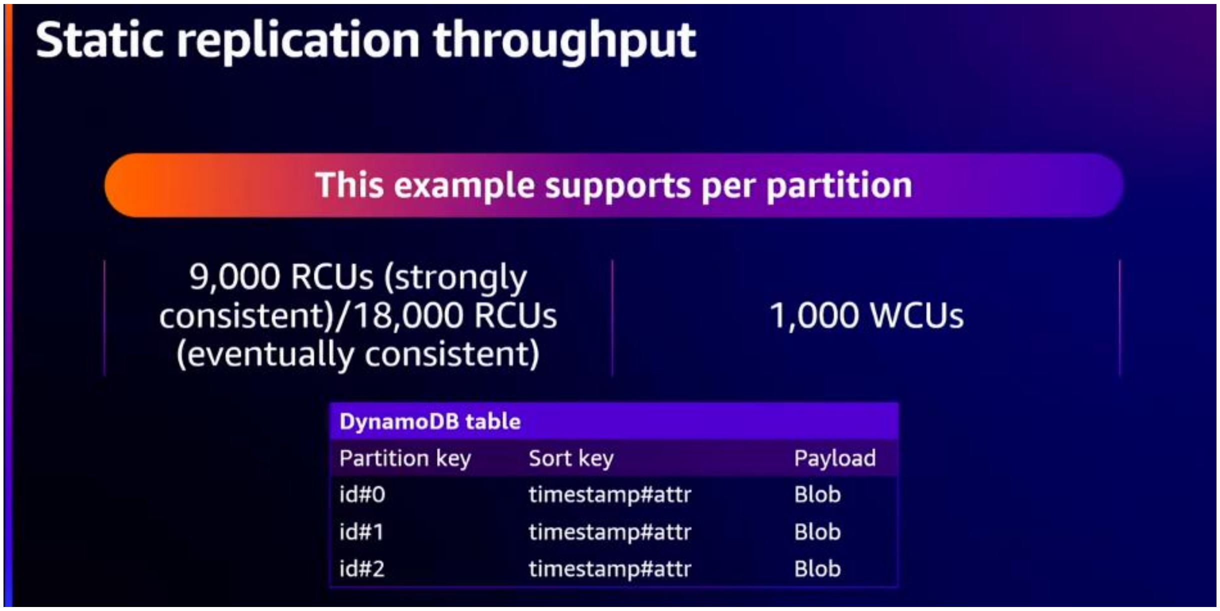
Default throughput

DynamoDB supports per partition

3,000 RCUs (strongly consistent)/6,000 RCUs (eventually consistent)

1,000 WCUs

DynamoDB table		
Partition key	Sort key	Payload
Id	timestamp#attr	Blob



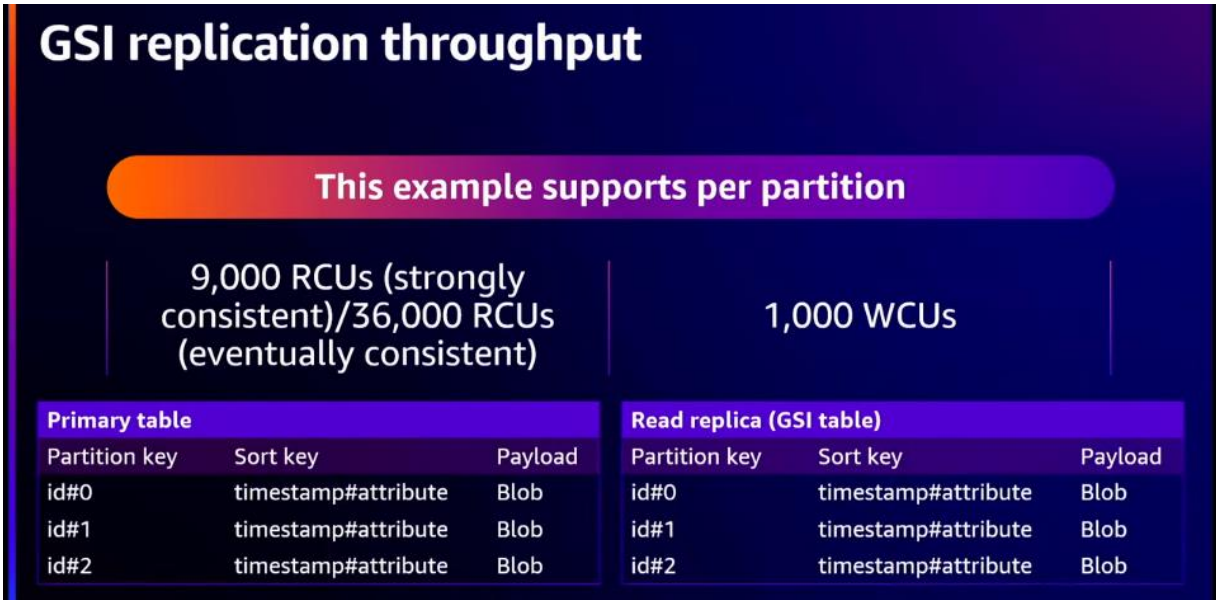
Static replication throughput

This example supports per partition

9,000 RCUs (strongly consistent)/18,000 RCUs (eventually consistent)

1,000 WCUs

DynamoDB table		
Partition key	Sort key	Payload
id#0	timestamp#attr	Blob
id#1	timestamp#attr	Blob
id#2	timestamp#attr	Blob



GSI replication throughput

This example supports per partition

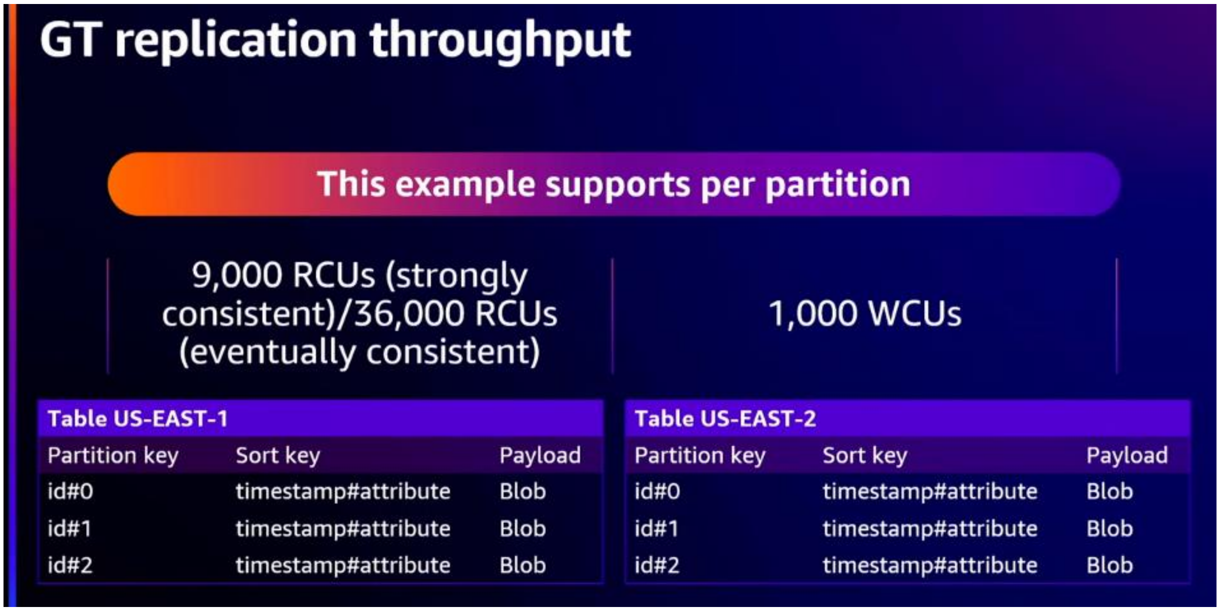
9,000 RCUs (strongly consistent)/36,000 RCUs (eventually consistent)	1,000 WCUs

Primary table

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob

Read replica (GSI table)

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob



GT replication throughput

This example supports per partition

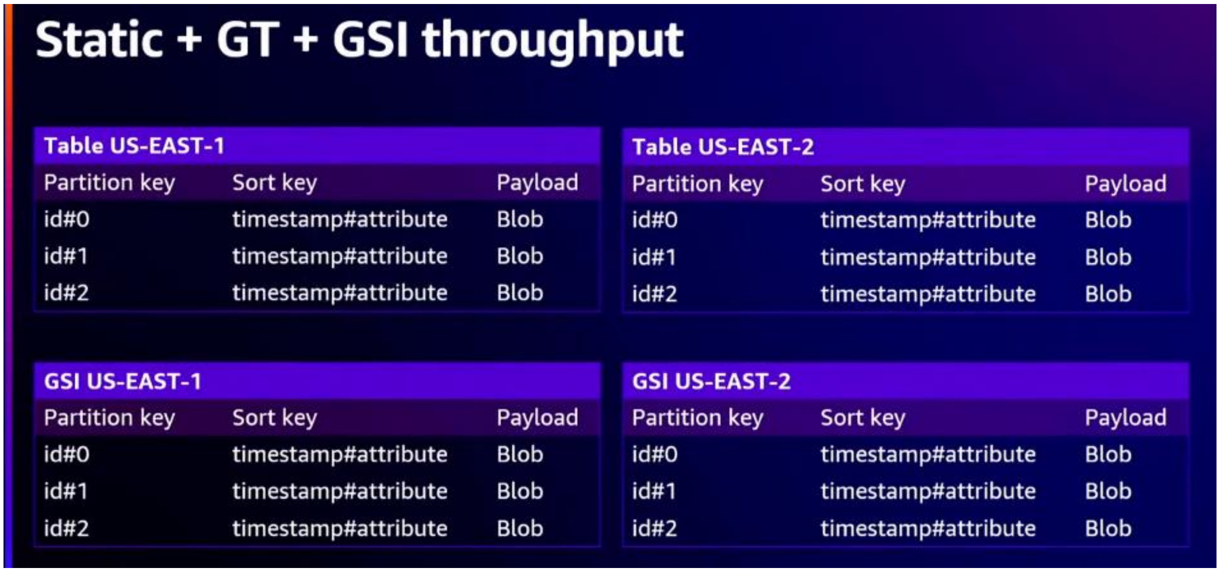
9,000 RCUs (strongly consistent)/36,000 RCUs (eventually consistent)	1,000 WCUs

Table US-EAST-1

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob

Table US-EAST-2

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob



Static + GT + GSI throughput

Table US-EAST-1

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob

Table US-EAST-2

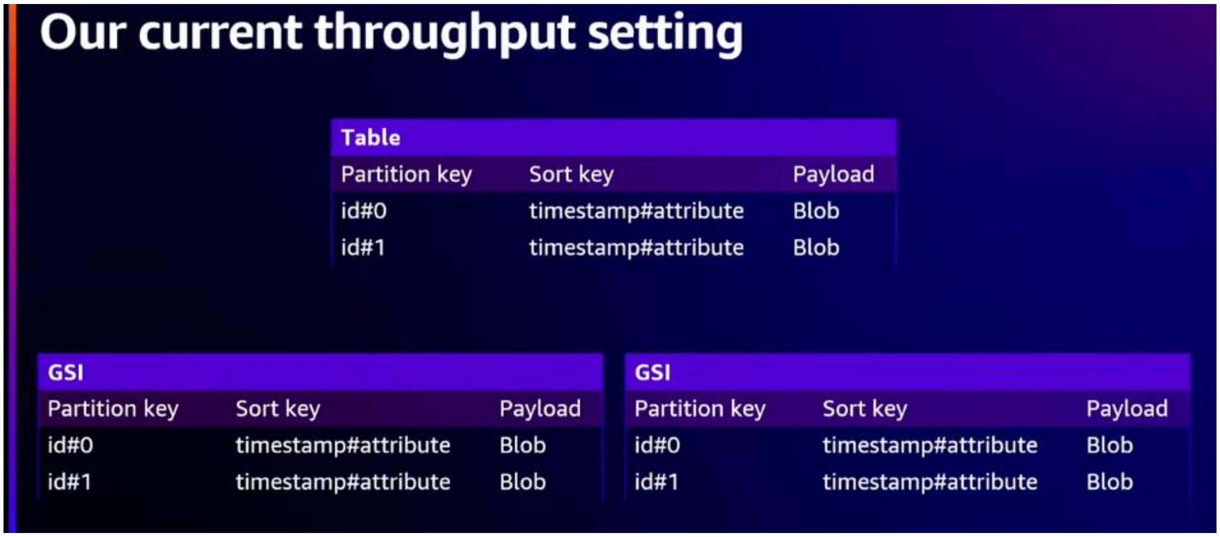
Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob

GSI US-EAST-1

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob

GSI US-EAST-2

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob

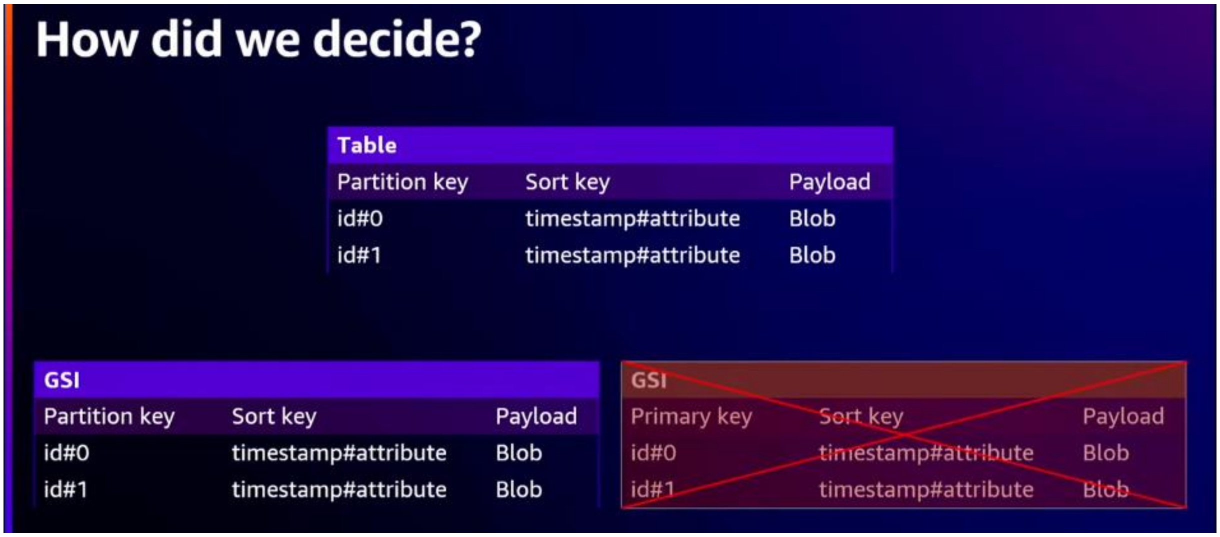


Our current throughput setting

Table		
Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob

GSI		
Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob

GSI		
Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob

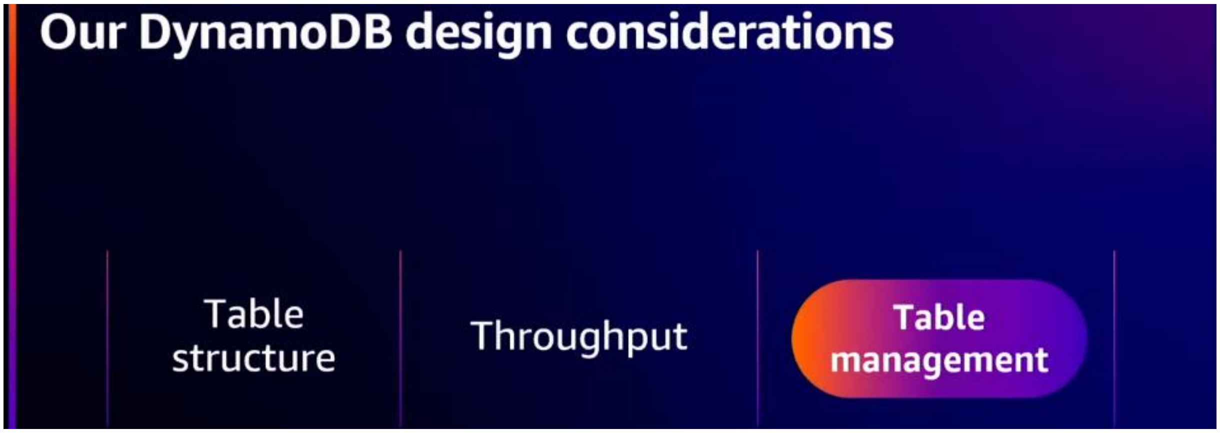


How did we decide?

Table		
Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob

GSI		
Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob

~~GSI~~		
~~Primary key~~	~~Sort key~~	~~Payload~~
~~id#0~~	~~timestamp#attribute~~	~~Blob~~
~~id#1~~	~~timestamp#attribute~~	~~Blob~~



Our DynamoDB design considerations

Table structure

Throughput

Table management

Is this static?

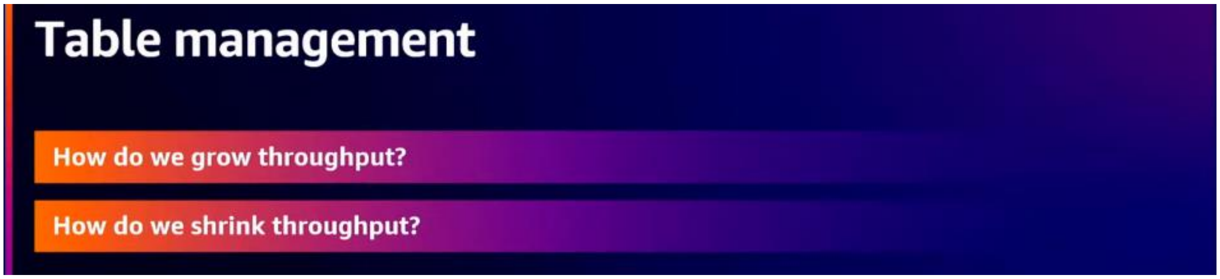


Table management

- How do we grow throughput?
- How do we shrink throughput?

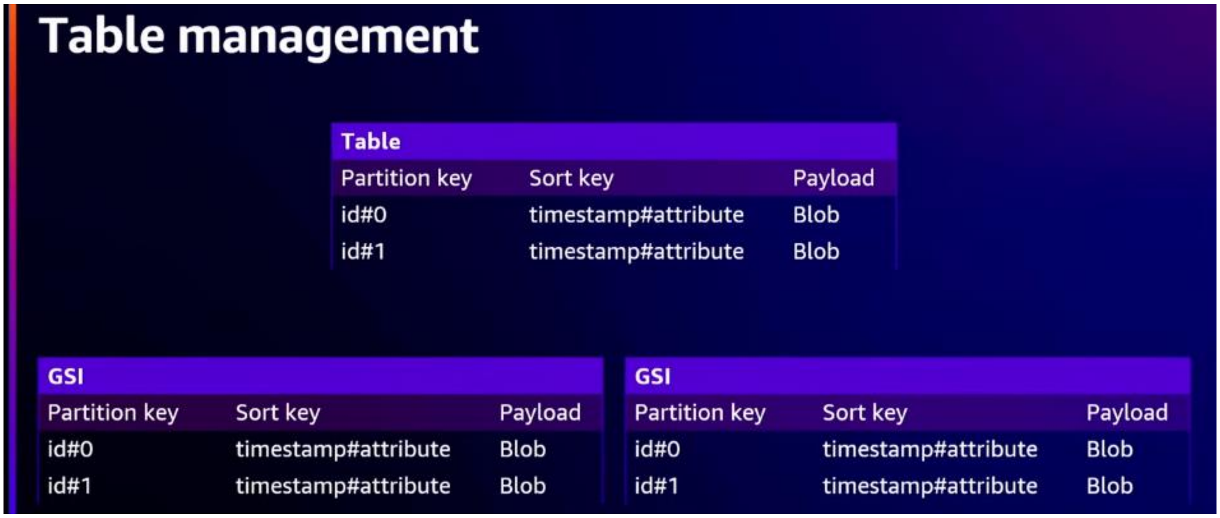


Table management

Table

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob

GSI

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob

GSI

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob

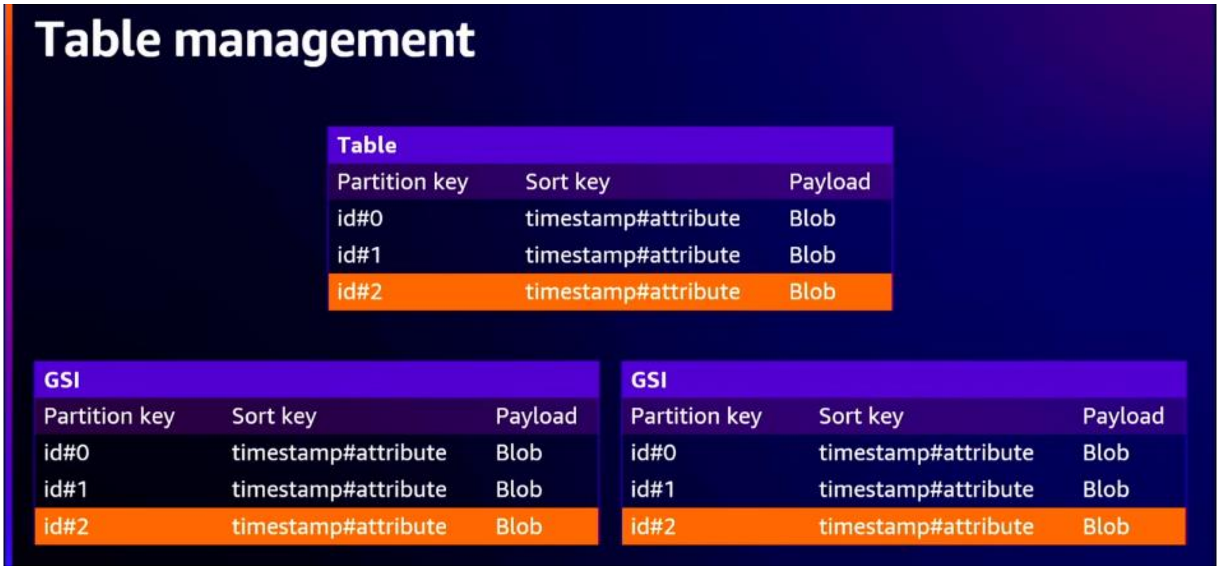


Table management

Table

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob

GSI

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob

GSI

Partition key	Sort key	Payload
id#0	timestamp#attribute	Blob
id#1	timestamp#attribute	Blob
id#2	timestamp#attribute	Blob

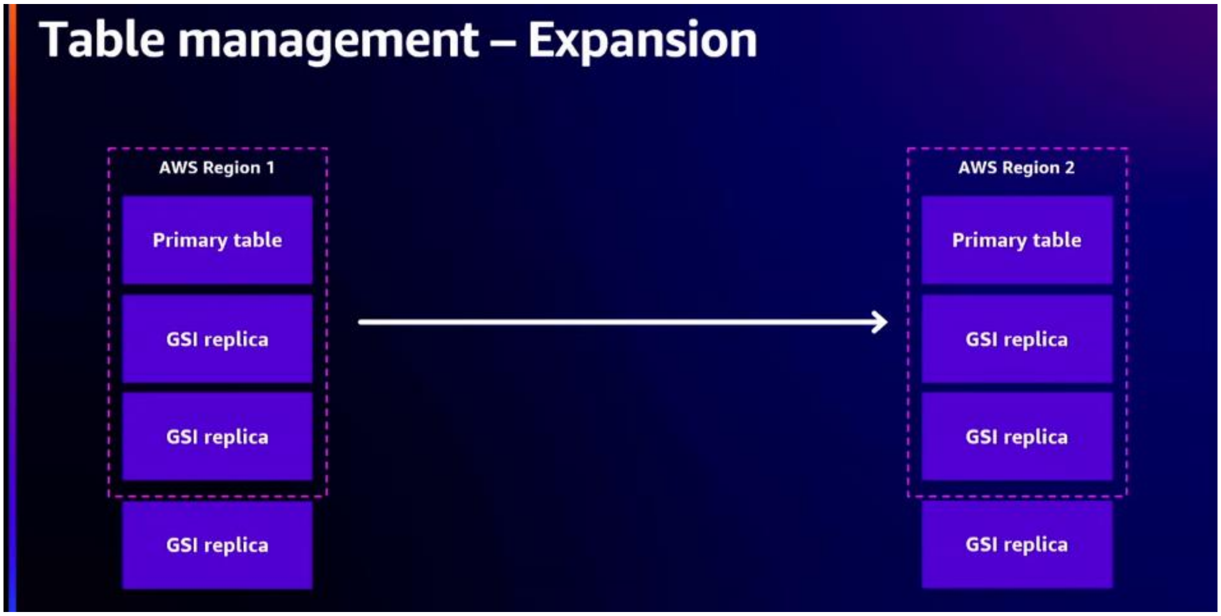


Table management – Expansion

AWS Region 1

Primary table

GSI replica

GSI replica

GSI replica

AWS Region 2

Primary table

GSI replica

GSI replica

GSI replica

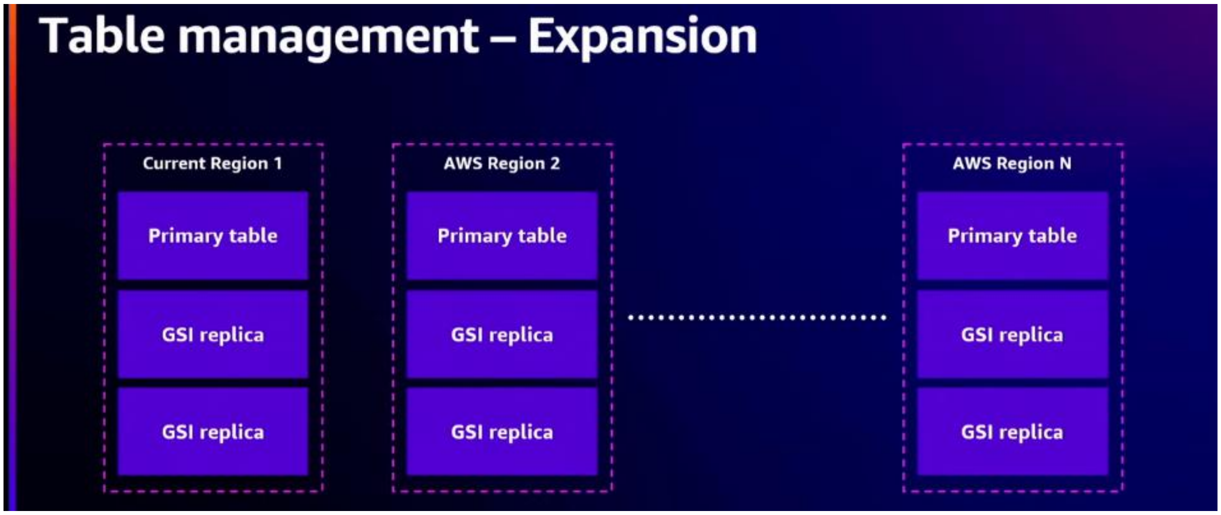


Table management – Expansion

Current Region 1

Primary table

GSI replica

GSI replica

AWS Region 2

Primary table

GSI replica

GSI replica

AWS Region N

Primary table

GSI replica

GSI replica

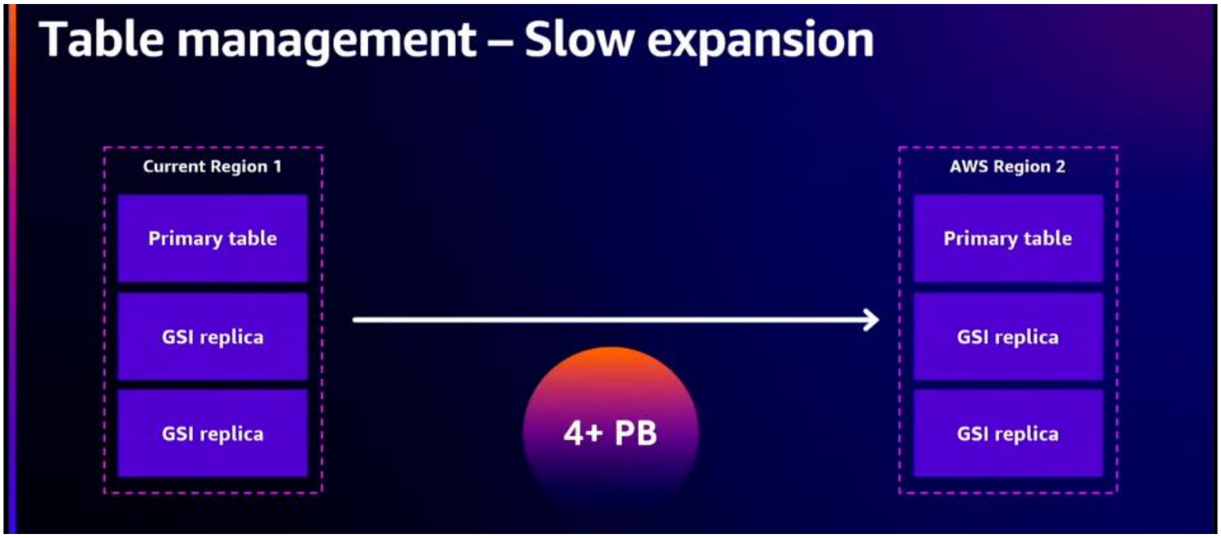


Table management – Slow expansion

Current Region 1

Primary table

GSI replica

GSI replica

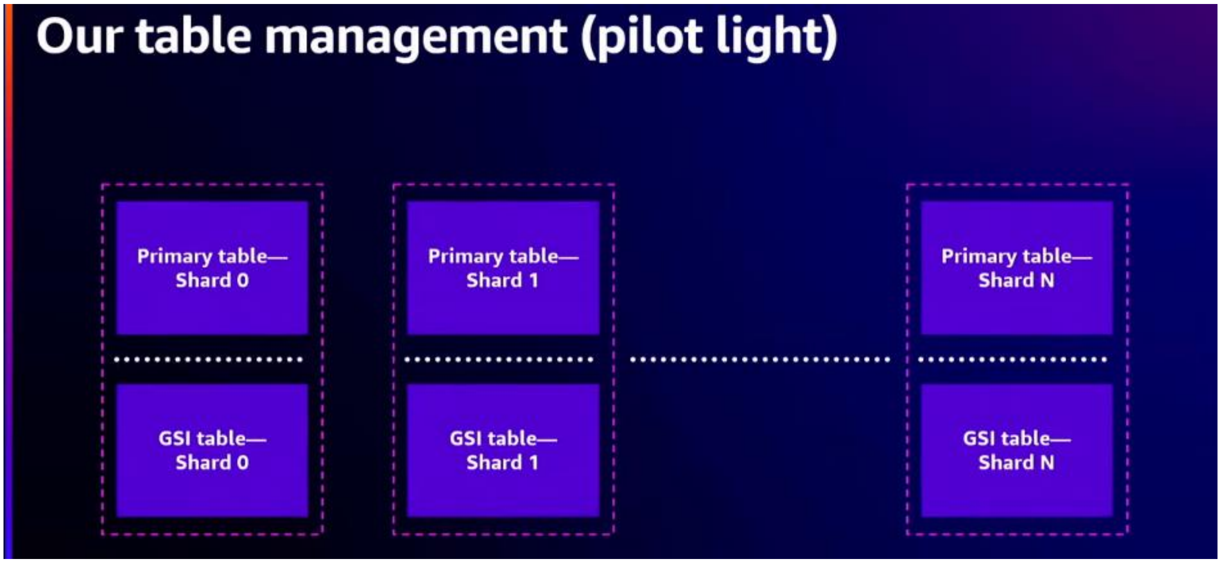
4+ PB

AWS Region 2

Primary table

GSI replica

GSI replica



Our table management (pilot light)

Primary table—
Shard 0

GSI table—
Shard 0

Primary table—
Shard 1

GSI table—
Shard 1

Primary table—
Shard N

GSI table—
Shard N

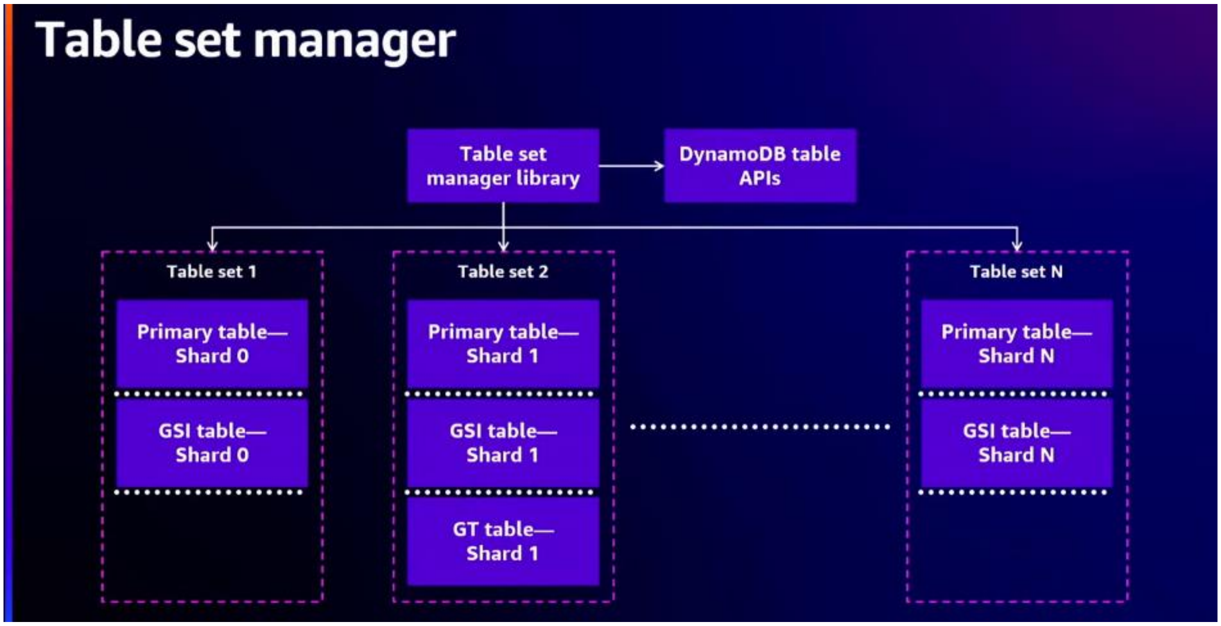


Table set manager

Table set
manager library

DynamoDB table
APIs

Table set 1

Primary table—
Shard 0

GSI table—
Shard 0

Table set 2

Primary table—
Shard 1

GSI table—
Shard 1

GT table—
Shard 1

Table set N

Primary table—
Shard N

GSI table—
Shard N

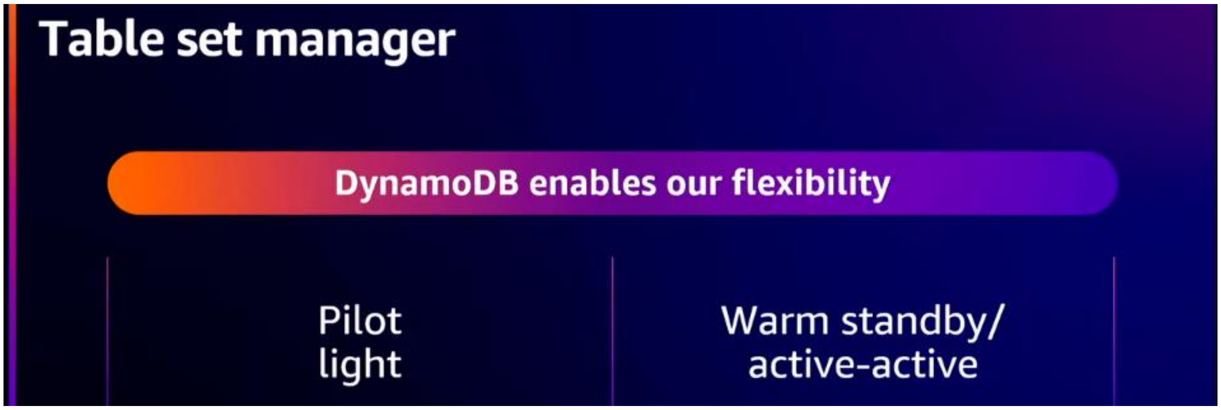
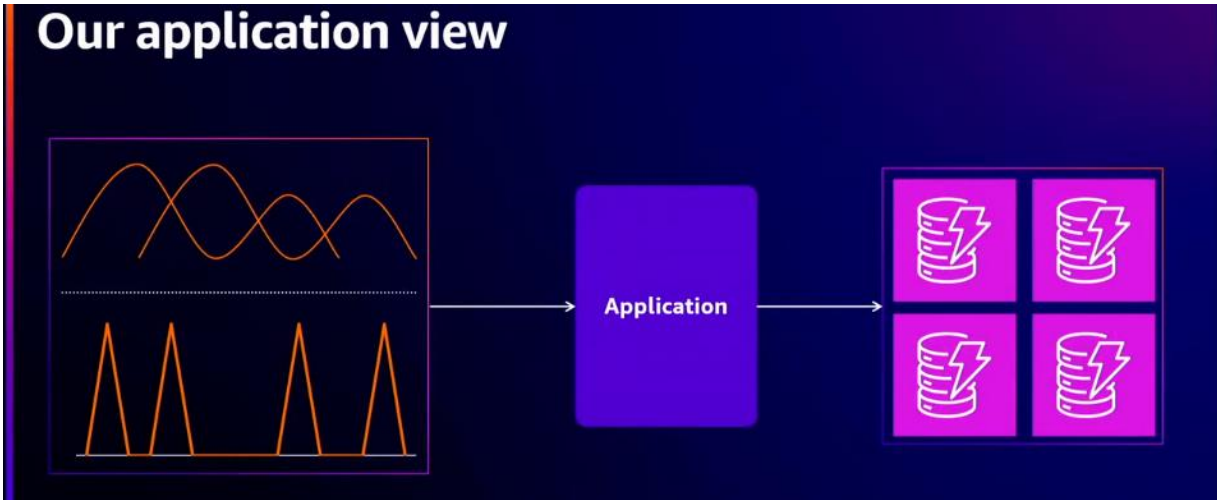


Table set manager

DynamoDB enables our flexibility

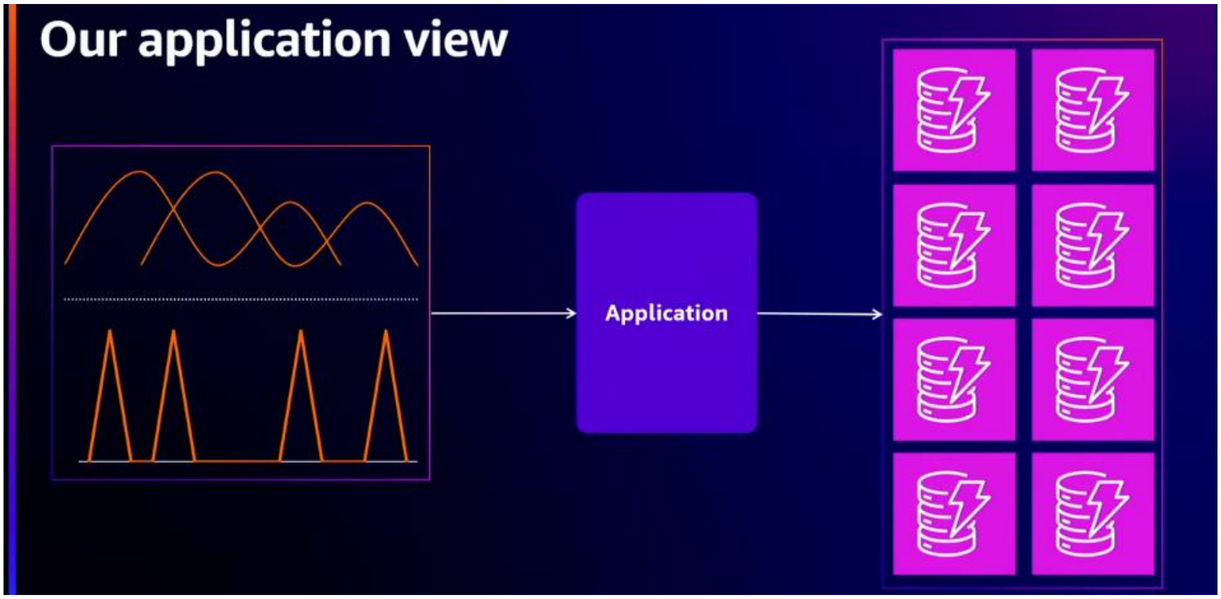
Pilot
light

Warm standby/
active-active



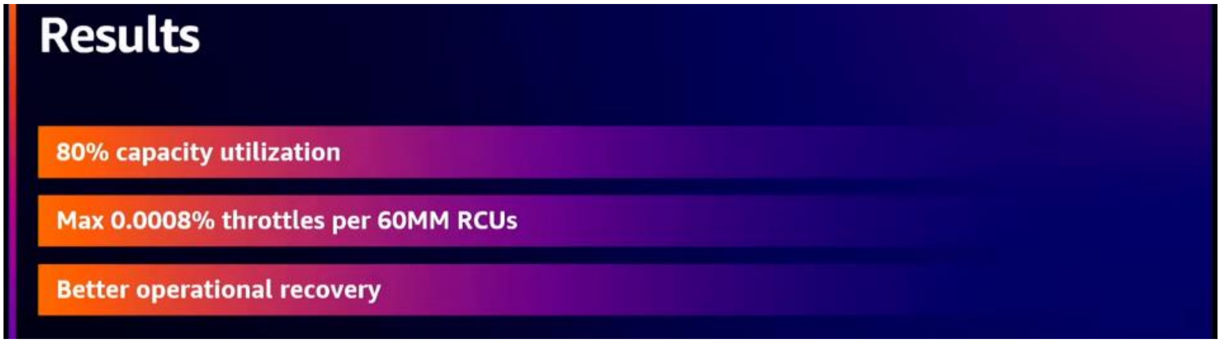
Our application view

Application



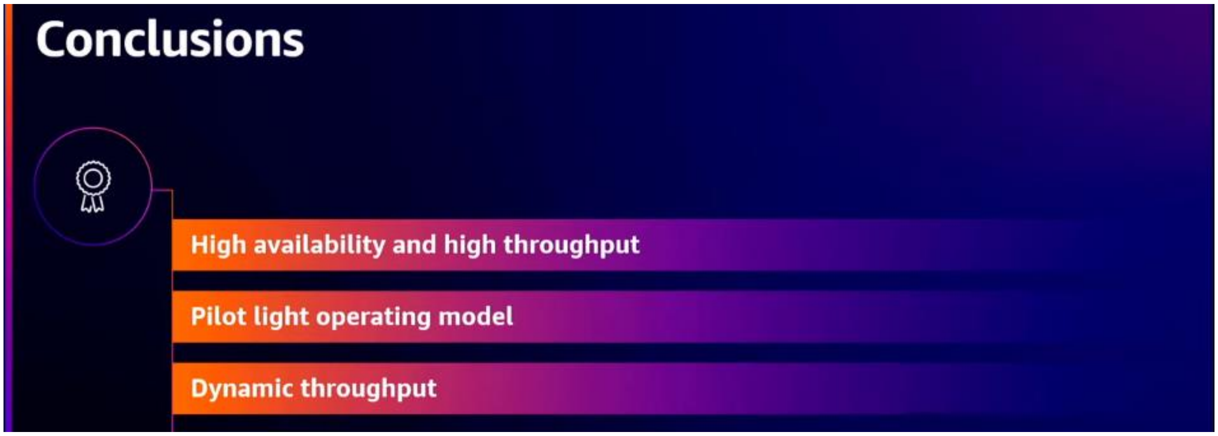
Our application view

Application



Results

- 80% capacity utilization
- Max 0.0008% throttles per 60MM RCUs
- Better operational recovery



Conclusions

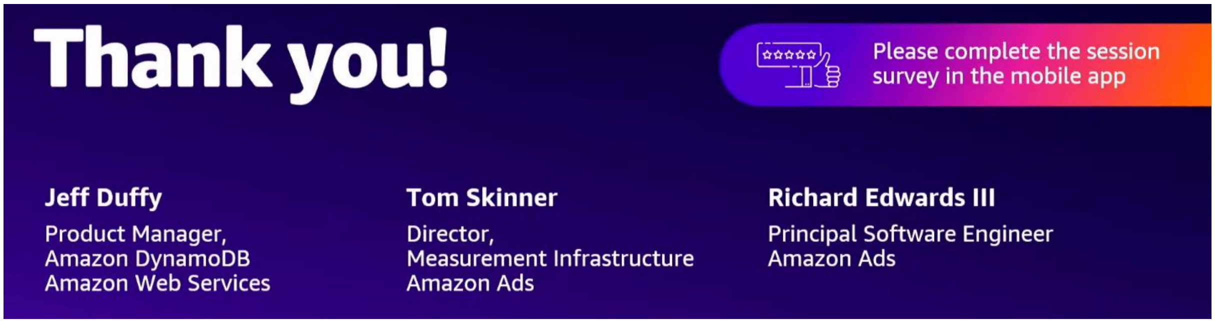
- High availability and high throughput
- Pilot light operating model
- Dynamic throughput

What's next

- Evaluate pilot light to warm standby/active-active
- Double down on our design pattern

If you take away just three things:

- Resilient applications require a resilient database
- DynamoDB offers a rich set of resilience features
- AWS helps you build for resilience



Thank you!

Please complete the session survey in the mobile app

Jeff Duffy
Product Manager,
Amazon DynamoDB
Amazon Web Services

Tom Skinner
Director,
Measurement Infrastructure
Amazon Ads

Richard Edwards III
Principal Software Engineer
Amazon Ads