Getting Started with Amazon DynamoDB

Julien Simon, Principal Technical Evangelist, AWS
@julsimon

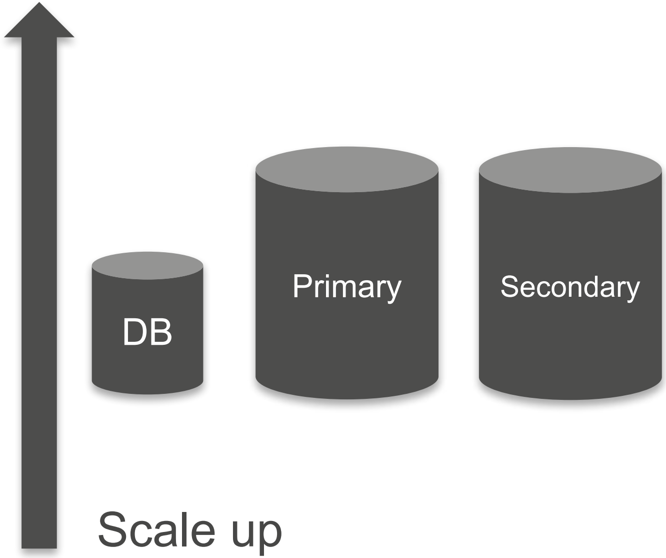
Agenda

- Relational (SQL) *vs.* non-relational (NoSQL)
- Fully managed features of DynamoDB
- Customer use cases
- Tables & secondary indexes
- Latest features
    - DynamoDB Time-to-live (TTL) – 02/2017
    - DynamoDB Auto Scaling – 06/2017
    - DynamoDB Accelerator (DAX) – 06/2017
    - DynamoDB VPC Endpoints (VPC-E) – 08/2017

Relational (SQL) *vs.* non-relational (NoSQL)

Relational vs. non-relational databases

Traditional SQL



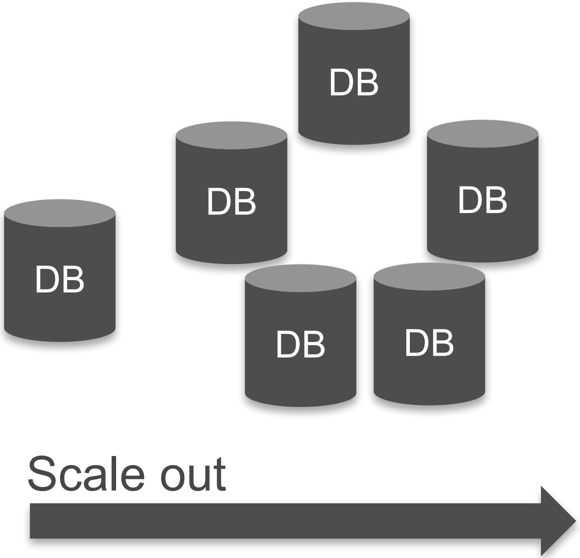
DB

Primary

Secondary

Scale up

NoSQL



DB

DB

DB

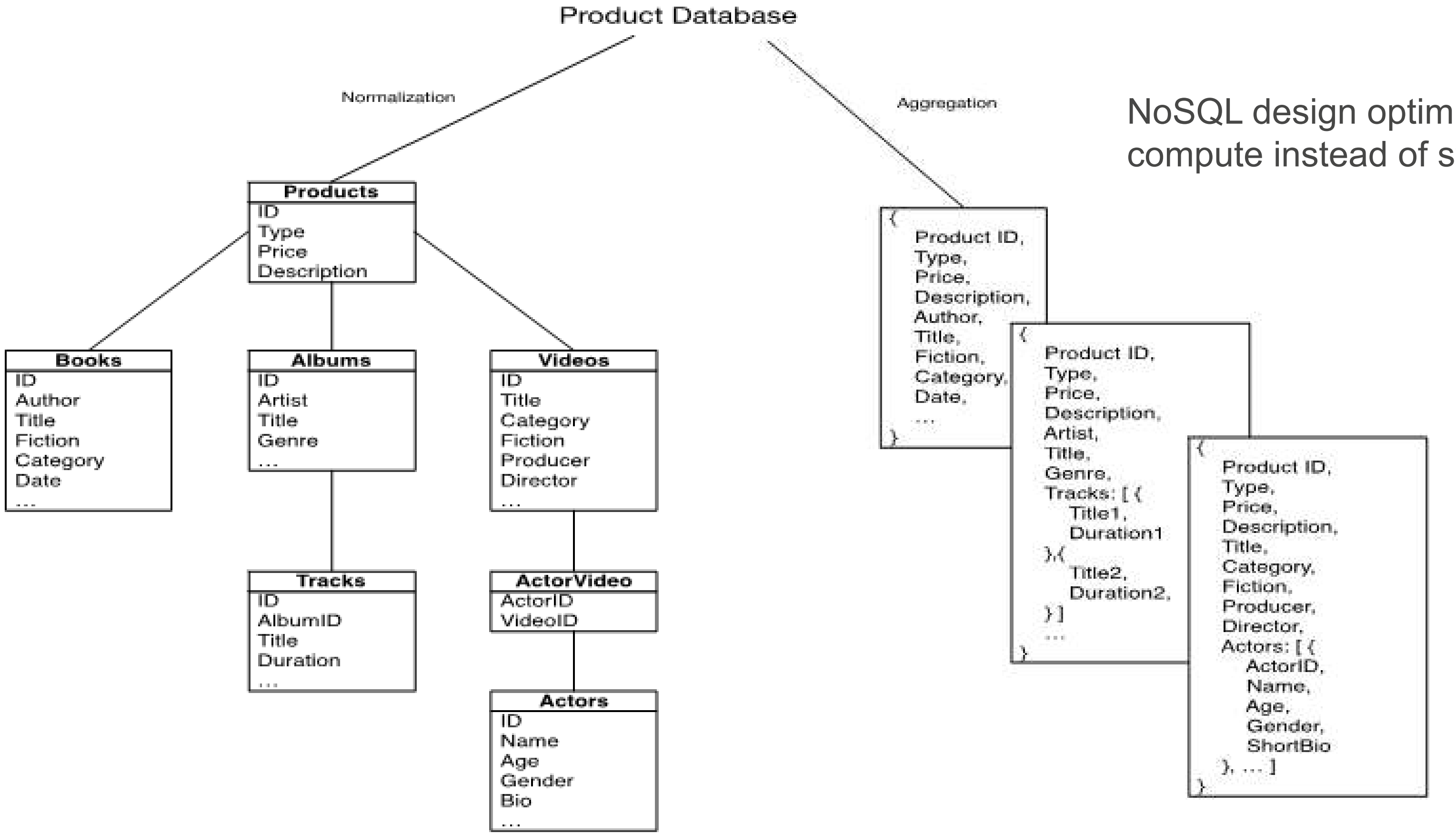
DB

DB

DB

Scale out

SQL vs. NoSQL schema design



Product Database

Normalization

Aggregation

Products
ID
Type
Price
Description

Books
ID
Author
Title
Fiction
Category
Date
...

Albums
ID
Artist
Title
Genre
...

Videos
ID
Title
Category
Fiction
Producer
Director
...

Tracks
ID
AlbumID
Title
Duration
...

ActorVideo
ActorID
VideoID

Actors
ID
Name
Age
Gender
Bio
...

```
{
    Product ID,
    Type,
    Price,
    Description,
    Author,
    Title,
    Fiction,
    Category,
    Date,
    ...
}
```

```
{
    Product ID,
    Type,
    Price,
    Description,
    Artist,
    Title,
    Genre,
    Tracks: [ {
        Title1,
        Duration1
    },{
        Title2,
        Duration2,
    } ]
    ...
}
```

```
{
    Product ID,
    Type,
    Price,
    Description,
    Title,
    Category,
    Fiction,
    Producer,
    Director,
    Actors: [ {
        ActorID,
        Name,
        Age,
        Gender,
        ShortBio
    }, ... ]
}
```

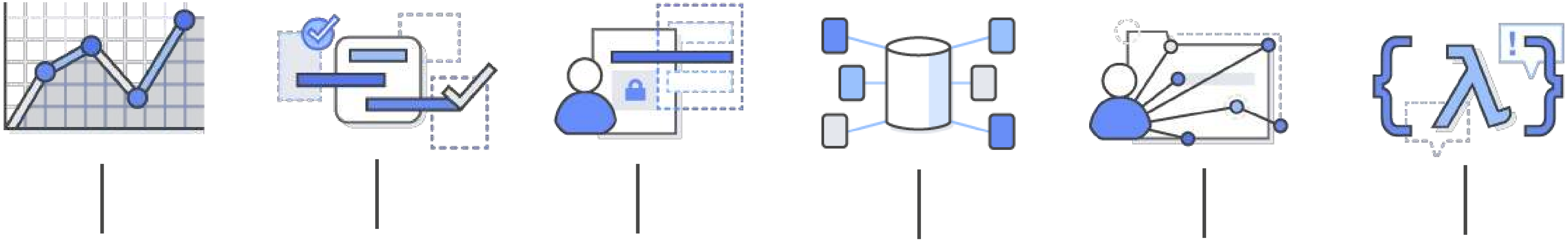
NoSQL design optimizes for compute instead of storage

Why NoSQL?

SQL	NoSQL
Optimized for storage	Optimized for compute
Normalized/relational	Denormalized/hierarchical
Ad-hoc queries	Instantiated views
Scale vertically	Scale horizontally
Good for OLAP	Built for OLTP at scale

Amazon DynamoDB

Run your business, not your database



Scalability

Performance

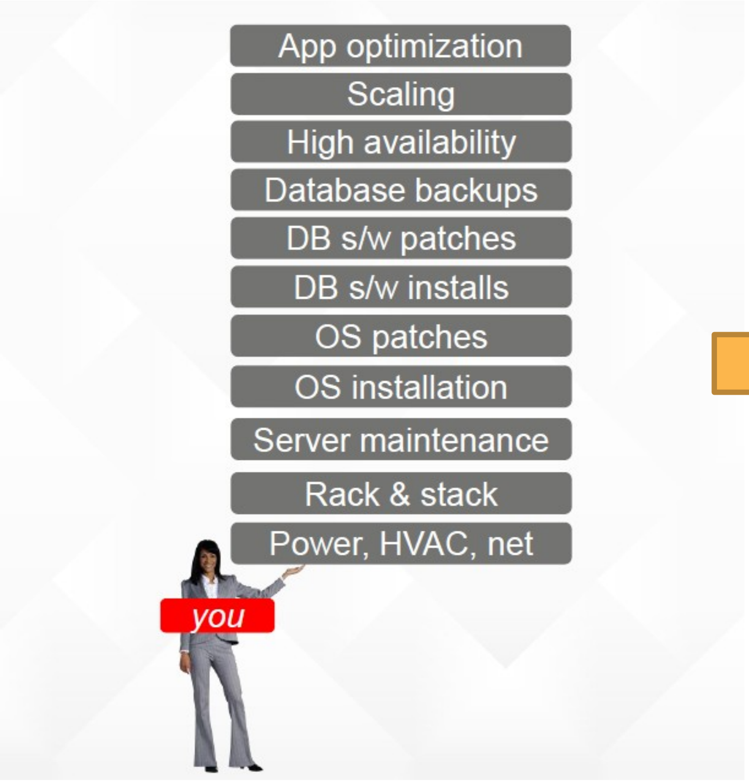
Security

Availability

Manageability

Developer Platform

Fully managed service = automated operations



App optimization

Scaling

High availability

Database backups

DB s/w patches

DB s/w installs

OS patches

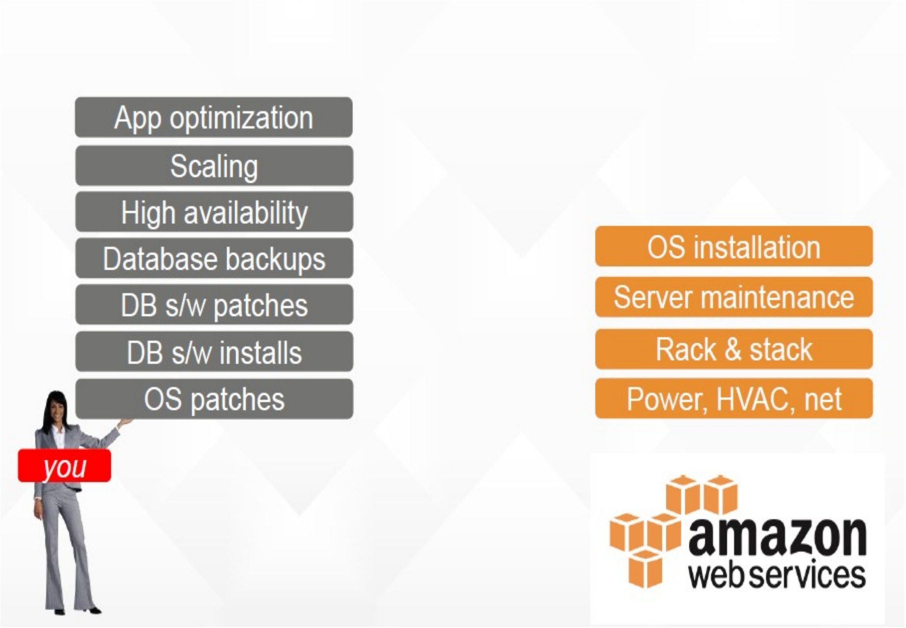
OS installation

Server maintenance

Rack & stack

Power, HVAC, net

DB hosted on premises



App optimization

Scaling

High availability

Database backups

DB s/w patches

DB s/w installs

OS patches

OS installation

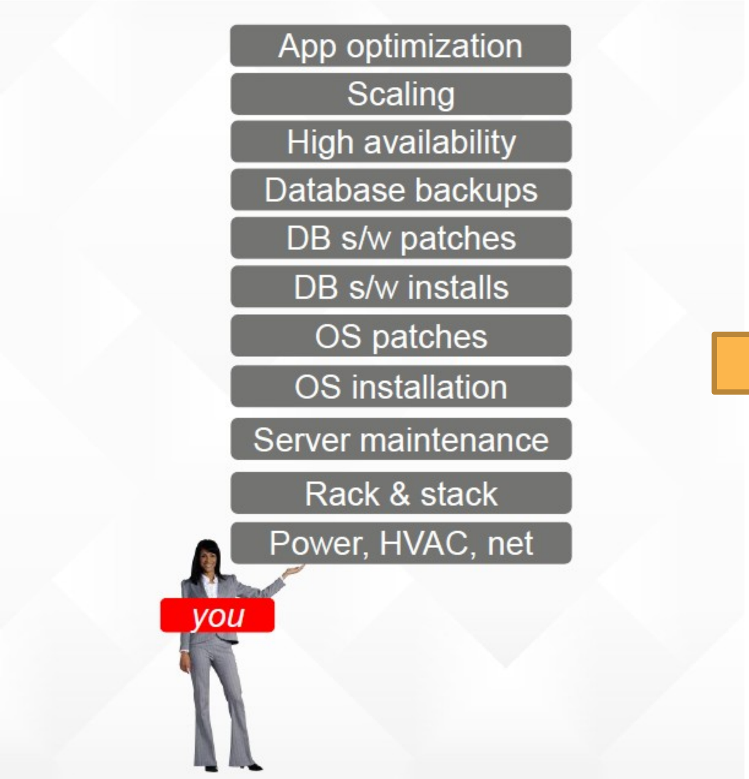
Server maintenance

Rack & stack

Power, HVAC, net

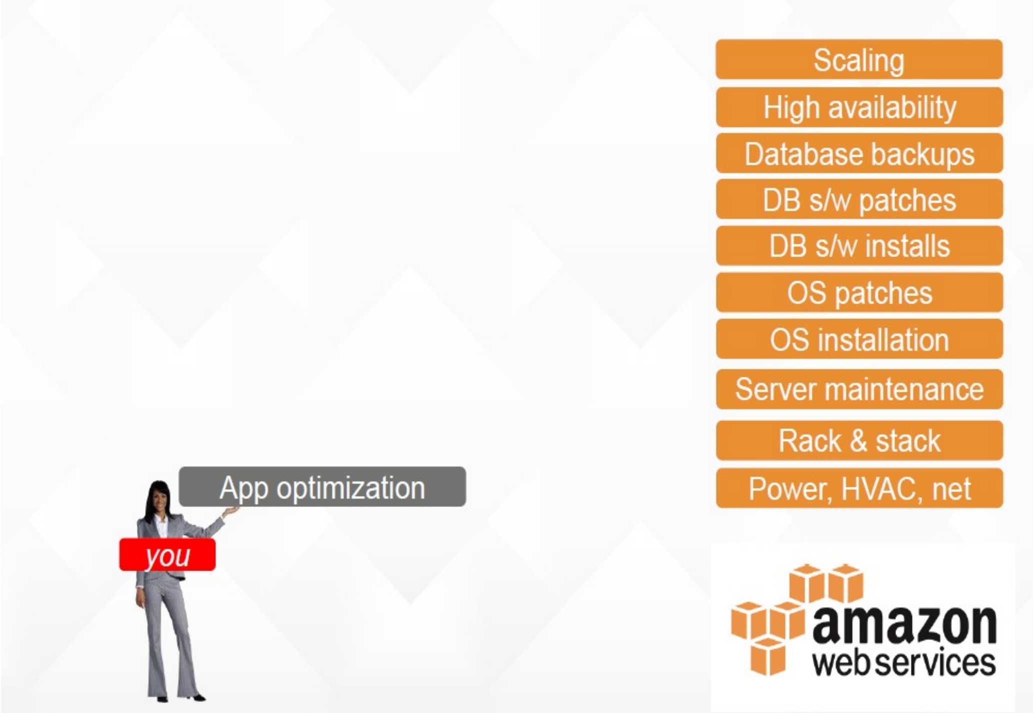
DB hosted on Amazon EC2

Fully managed service = automated operations



App optimization
Scaling
High availability
Database backups
DB s/w patches
DB s/w installs
OS patches
OS installation
Server maintenance
Rack & stack
Power, HVAC, net
you

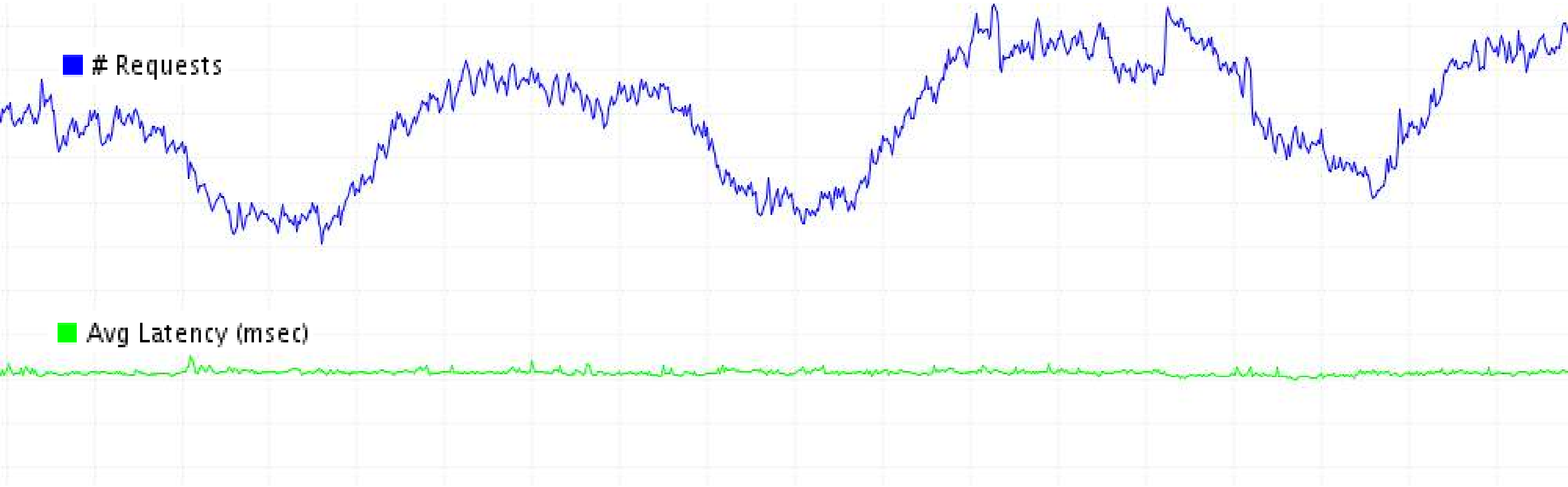
DB hosted on premises



Scaling
High availability
Database backups
DB s/w patches
DB s/w installs
OS patches
OS installation
Server maintenance
Rack & stack
Power, HVAC, net
App optimization
you
amazon webservices

DynamoDB

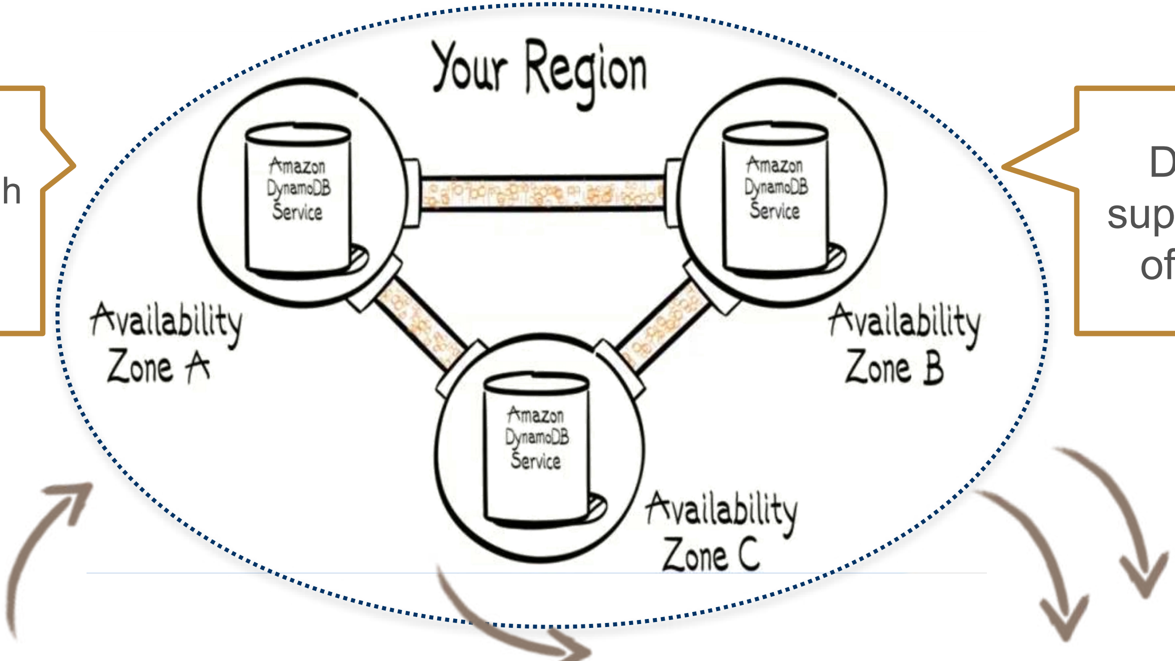
Consistently low latency at scale



PREDICTABLE PERFORMANCE!

High availability and durability

Built for high durability



Designed to support 99.99% of availability

WRITES
Replicated continuously to 3 Availability Zones
Persisted to disk (custom SSD)

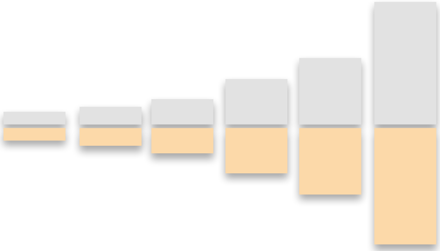
READS
Strongly or eventually consistent
No latency trade-off

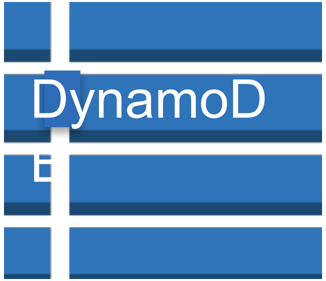
DynamoDB pricing & Free Tier

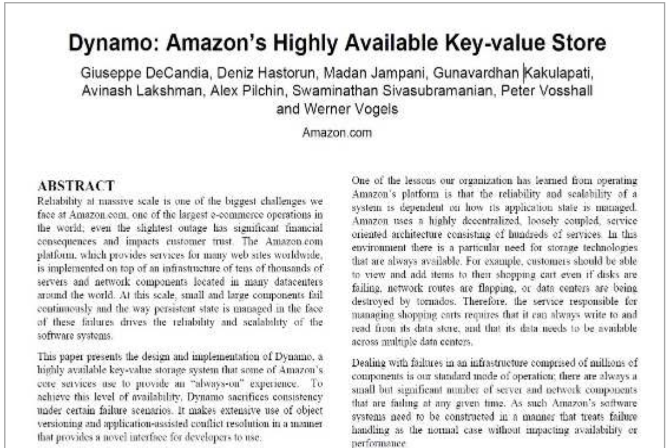
- Free Tier
  - ❑ 25 GB of storage
  - ❑ 25 reads per second
  - ❑ 25 writes per second
- Pricing for additional usage in US East (N. Virginia)
  - ❑ $0.25 per GB per month
  - ❑ Write throughput: $0.0065 per hour for every 10 units of Write Capacity
  - ❑ Read throughput: $0.0065 per hour for every 50 units of Read Capacity

Customer use cases

Amazon’s Path to DynamoDB







Dynamo: Amazon’s Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels

Amazon.com

ABSTRACT

Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters around the world. At this scale, small and large components fail continuously and the way persistent state is managed in the face of these failures drives the reliability and scalability of the software systems.

This paper presents the design and implementation of Dynamo, a highly available key-value storage system that some of Amazon’s core services use to provide an “always-on” experience. To achieve this level of availability, Dynamo sacrifices consistency under certain failure scenarios. It makes extensive use of object versioning and application-assisted conflict resolution in a manner that provides a novel interface for developers to use.

One of the lessons our organization has learned from operating Amazon’s platform is that the reliability and scalability of a system is dependent on how its application state is managed. Amazon uses a highly decentralized, loosely coupled, service oriented architecture consisting of hundreds of services. In this environment there is a particular need for storage technologies that are always available. For example, customers should be able to view and add items to their shopping cart even if disks are failing, network routes are flapping, or data centers are being destroyed by tornados. Therefore, the service responsible for managing shopping carts requires that it can always write to and read from its data store, and that its data needs to be available across multiple data centers.

Dealing with failures in an infrastructure comprised of millions of components is our standard mode of operation; there are always a small but significant number of server and network components that are failing at any given time. As such Amazon’s software systems need to be constructed in a manner that treats failure handling as the normal case without impacting availability or performance

Major League Baseball fields big data, excitement with Amazon DynamoDB

> “For the first time, we can measure things we’ve never been able to measure before.”
>
> **Joe Inzerillo**
> Executive Vice President and CTO, MLBAM

MLBAM (MLB Advanced Media) is a full service solutions provider, operating a powerful content delivery platform.

- MLBAM can scale to support many games on a single day.
- DynamoDB powers queries and supports the fast data retrieval required.
- MLBAM distributes 25,000 live events annually and 10 million streams daily.

Redfin is revolutionizing home buying and selling with Amazon DynamoDB

> “We have billions of records on DynamoDB being refreshed daily or hourly or even by seconds.”

Yong Huang
Director, Big Data Analytics, Redfin

REDFIN

Redfin is a full-service real estate company with local agents and online tools to help people buy & sell homes.

- Redfin provides property and agent details and ratings through its websites and apps.
- With DynamoDB, latency for “similar” properties improved from 2 seconds to just 12 milliseconds.
- Redfin stores and processes five billion items in DynamoDB.

Expedia's real-time analytics application uses Amazon DynamoDB

> "With DynamoDB we were up and running in a less than day, and there is no need for a team to maintain."
>
> **Kuldeep Chowhan**
> Principal Engineer, Expedia

Expedia

Expedia is a leader in the $1 trillion travel industry, with an extensive portfolio that includes some of the world's most trusted travel brands.

- Expedia's real-time analytics application collects data for its "test & learn" experiments on Expedia sites.
- The analytics application processes ~200 million messages daily.
- Ease of setup, monitoring, and scaling were key factors in choosing DynamoDB.

Nexon scales mobile gaming with Amazon DynamoDB

> By using AWS, we decreased our initial investment costs, and only pay for what we use.
>
> **Chunghoon Ryu**
> Department Manager, Nexon

NEXON

Nexon is a leading South Korean video game developer and a pioneer in the world of interactive entertainment.

- Nexon used DynamoDB as its primary game database for a new blockbuster mobile game, HIT.
- HIT became the #1 Mobile Game in Korea within the first day of launch and has > 2M registered users.
- Nexon's HIT leverages DynamoDB to deliver steady latency of less than 10ms to deliver a fantastic mobile gaming experience for 170,000 concurrent players.

DynamoDB Customers

Mobile

- UNDER ARMOUR
- airbnb
- lyft
- REDFIN
- duolingo

Web

- Adobe
- Expedia
- amazon marketplace
- JustGiving
- jobandtalent

Gaming

- SUPERCELL
- zynga
- NEXON
- FanDuel
- FRONTIER

IoT

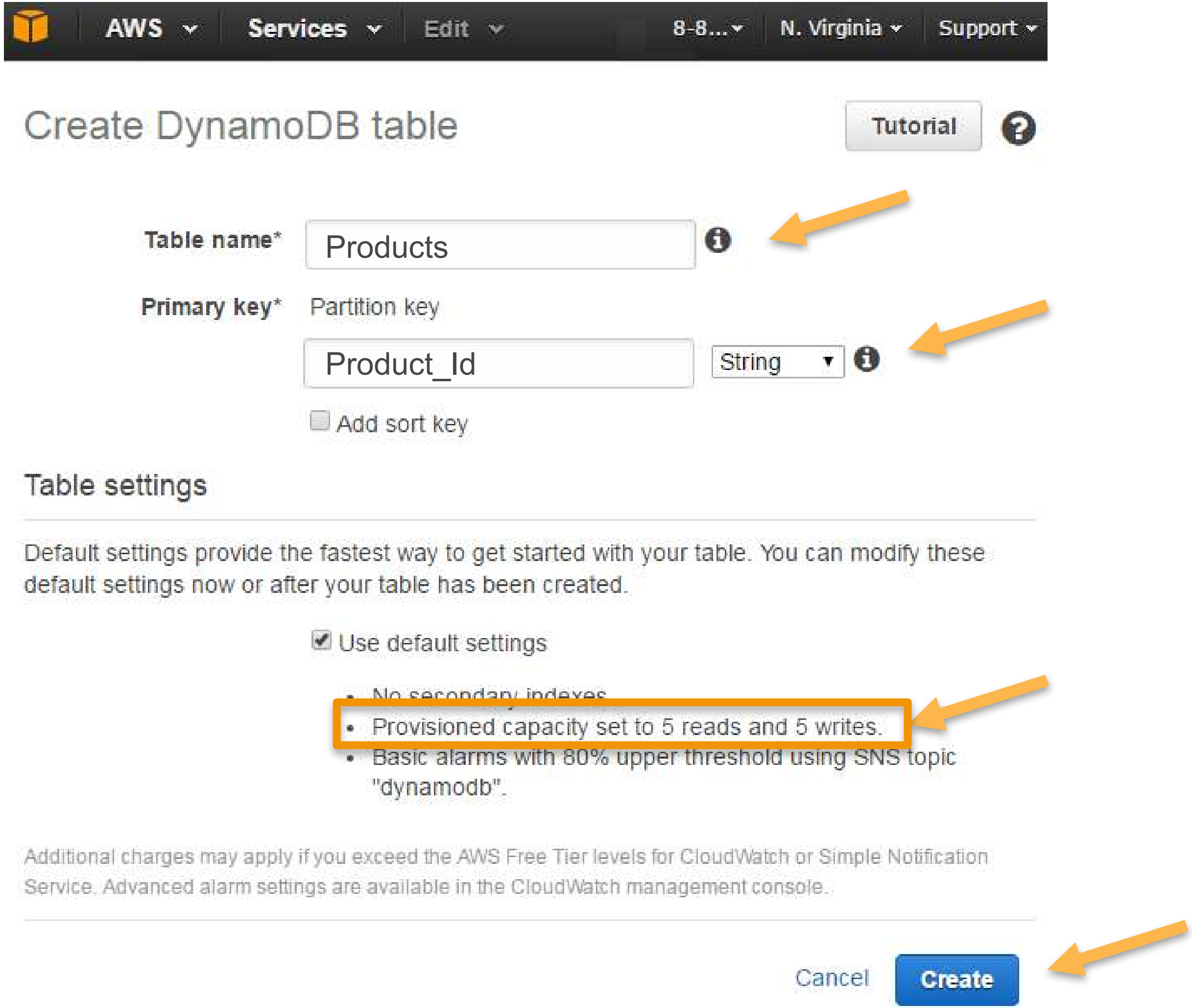
- mlbam
- canary
- dropcam
- ACTi Connecting Vision
- MEDIATEK

Ad Tech

- DataXu
- has offers by TUNE
- eyeview
- ADBRAIN
- jampp

That sounds really good. How do I get started?

Let’s create a table…



Create DynamoDB table

Table name* Products

Primary key* Partition key

Product_Id String

□ Add sort key

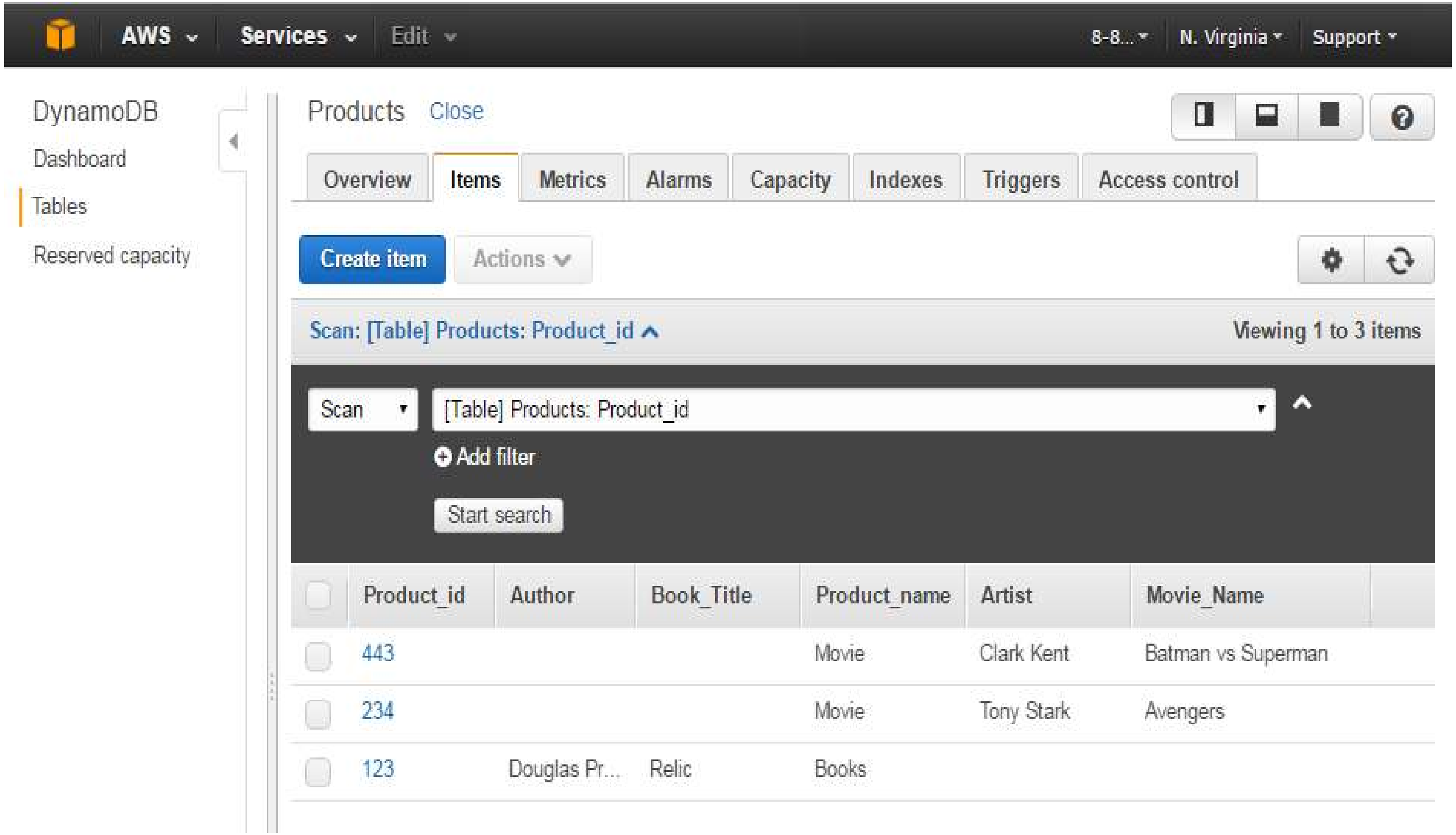
Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☑ Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.



Scan: [Table] Products: Product_id

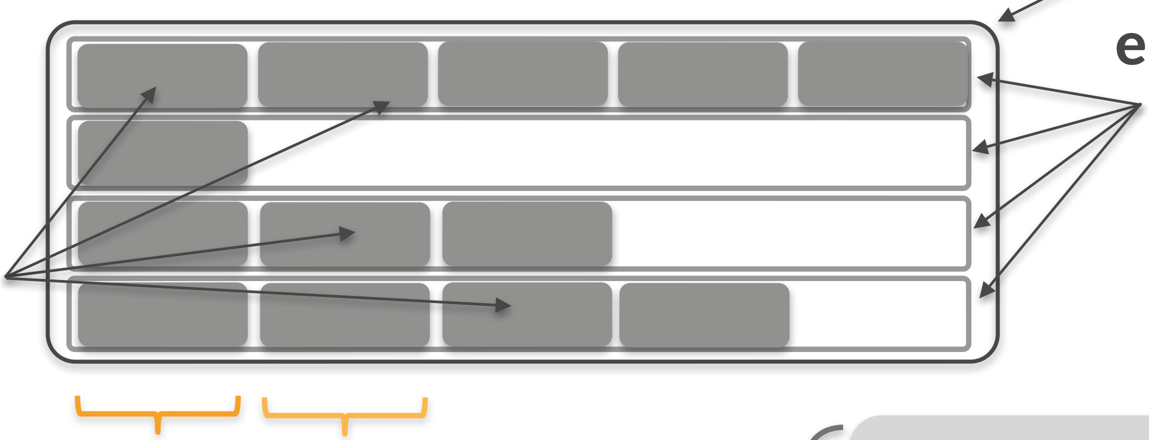
Viewing 1 to 3 items

Scan

[Table] Products: Product_id

Product_id	Author	Book_Title	Product_name	Artist	Movie_Name
443			Movie	Clark Kent	Batman vs Superman
234			Movie	Tony Stark	Avengers
123	Douglas Pr...	Relic	Books		

DynamoDB table structure



Tabl
e

Item
s

Attribute
s

Partitio
n
key

Sor
t
key

Mandatory
Key-value access pattern
Determines data distribution

Optional
Enables rich query capabilities

All items for key
==, <, >, >=, <=
“begins with”
“between”
“contains”
“in”
sorted results
counts
top/bottom N values

Local secondary index (LSI)

Alternate sort key attribute

Index is local to a partition key

10 GB maximum per partition key; LSIs limit the number of range keys!

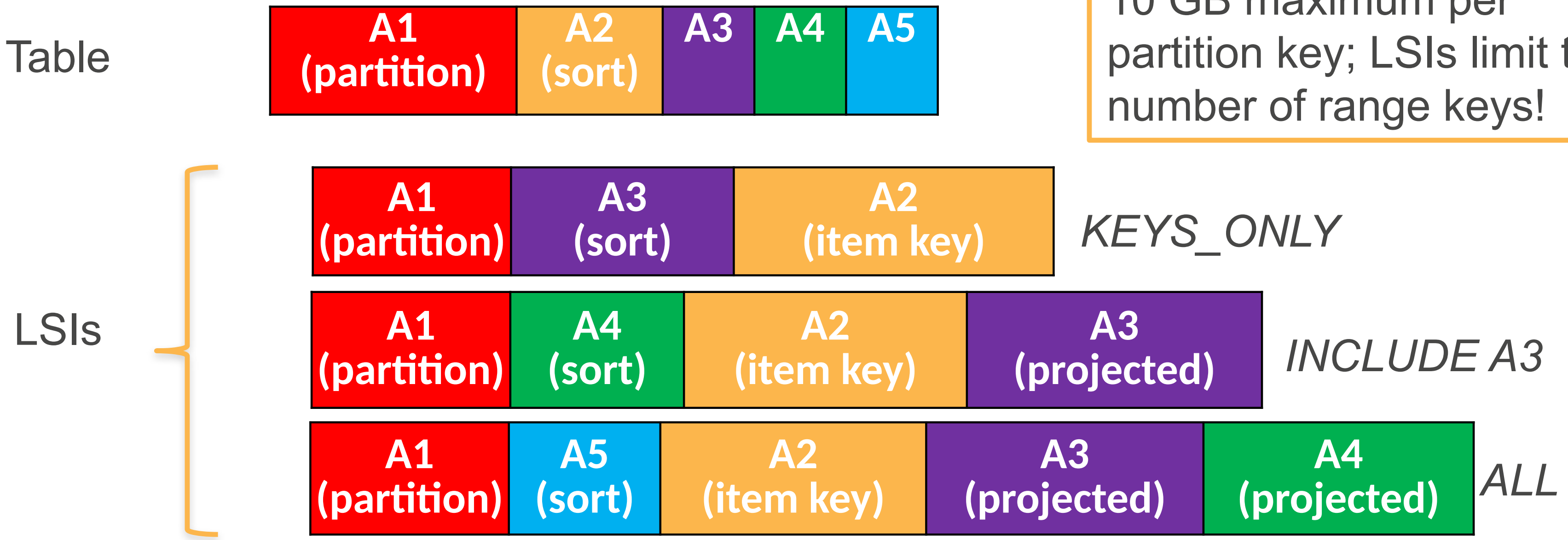


Table: A1 (partition) | A2 (sort) | A3 | A4 | A5

LSIs:

A1 (partition) | A3 (sort) | A2 (item key) — *KEYS_ONLY*

A1 (partition) | A4 (sort) | A2 (item key) | A3 (projected) — *INCLUDE A3*

A1 (partition) | A5 (sort) | A2 (item key) | A3 (projected) | A4 (projected) — *ALL*

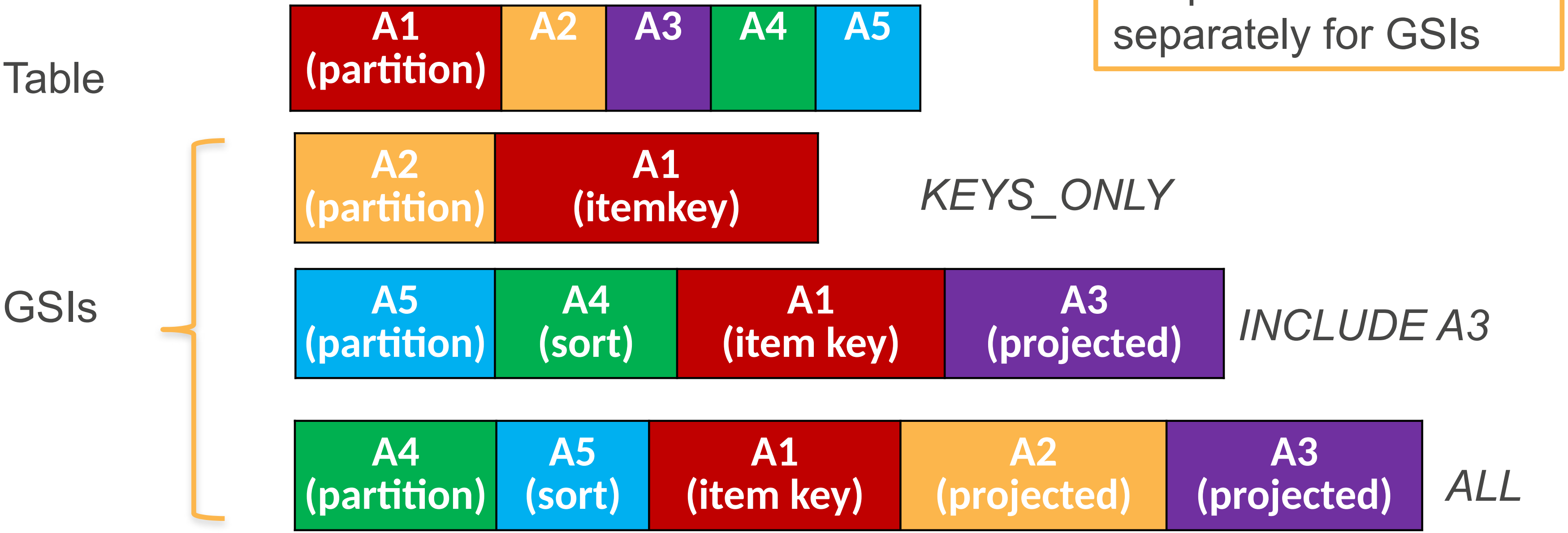
Global secondary index (GSI)

Alternate partition and/or sort key

Index is across all partition keys

Online indexing

Read capacity units (RCUs) and write capacity units (WCUs) are provisioned separately for GSIs



Table

A1 (partition) | A2 | A3 | A4 | A5

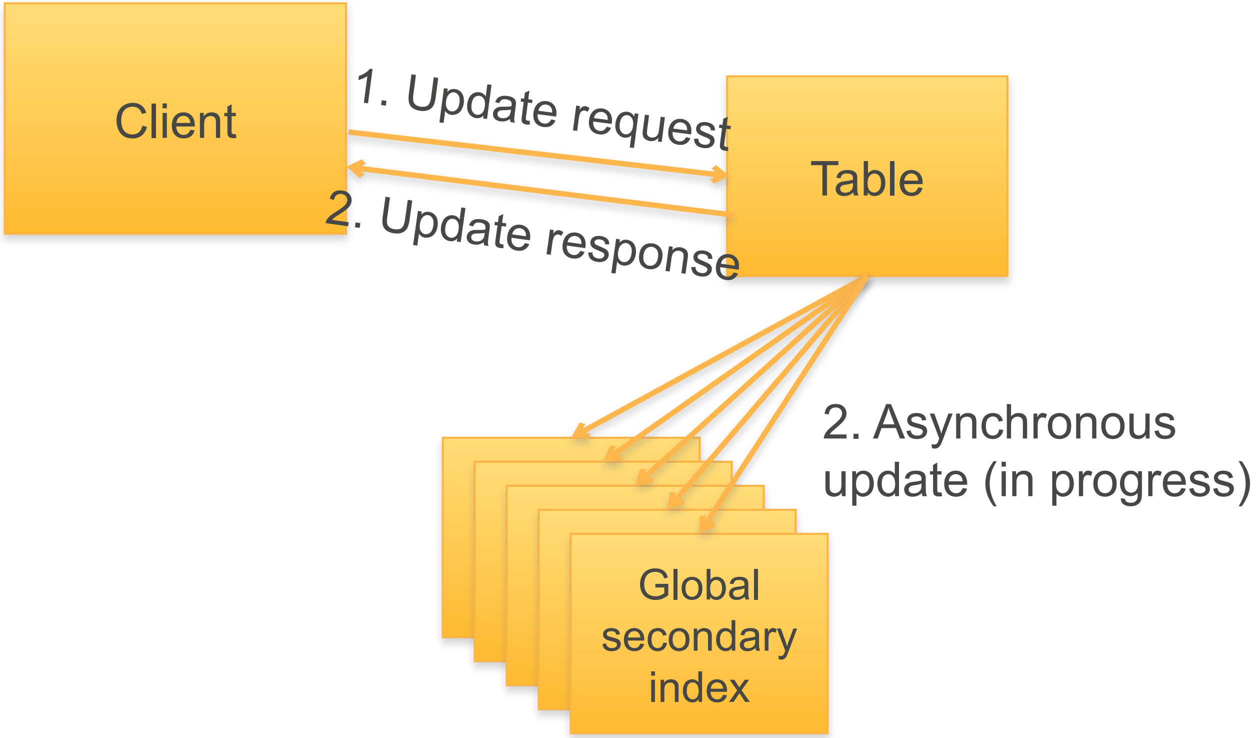
GSIs

A2 (partition) | A1 (itemkey) — *KEYS_ONLY*

A5 (partition) | A4 (sort) | A1 (item key) | A3 (projected) — *INCLUDE A3*

A4 (partition) | A5 (sort) | A1 (item key) | A2 (projected) | A3 (projected) — *ALL*

How do GSI updates work?



If GSIs don’t have enough write capacity, table writes are throttled!

LSI or GSI?

Use LSI if you absolutely need strong consistency

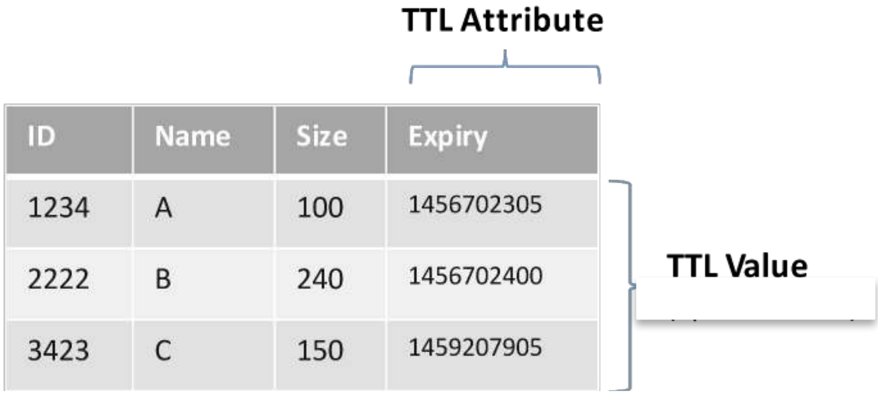
GSI works for 99% of scenarios and is more flexible

- LSI must be created at the same time the table is created. GSI can be created later
- Different partition key
- Dedicated read & write capacity

If data size in an item collection > 10 GB, you must use GSI

DynamoDB Time-To-Live

Time-to-Live (TTL)


TTL Attribute
TTL Value

ID	Name	Size	Expiry
1234	A	100	1456702305
2222	B	240	1456702400
3423	C	150	1459207905

Features

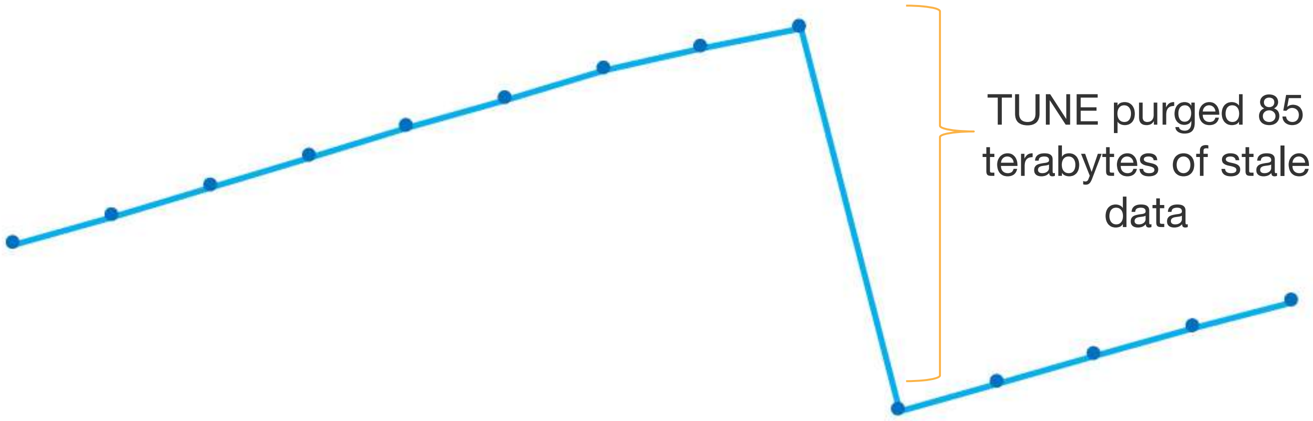
- **Automatic:** Deletes items from a table based on expiration timestamp
- **Customizable:** User-defined TTL attribute in epoch time format
- **Audit Log:** TTL activity recorded in DynamoDB Streams

Key Benefits

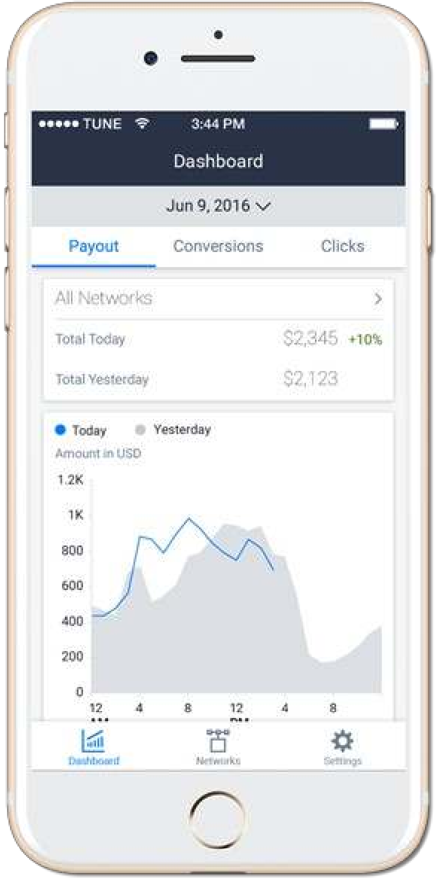
- **Reduce costs:** Delete items no longer needed
- **Performance:** Optimize application performance by controlling table size growth
- **Extensible:** Trigger custom workflows with DynamoDB Streams and Lambda

Time-to-live (TTL)

TUNE



TUNE purged 85 terabytes of stale data



●●●●● TUNE 3:44 PM

Dashboard

Jun 9, 2016

All Networks

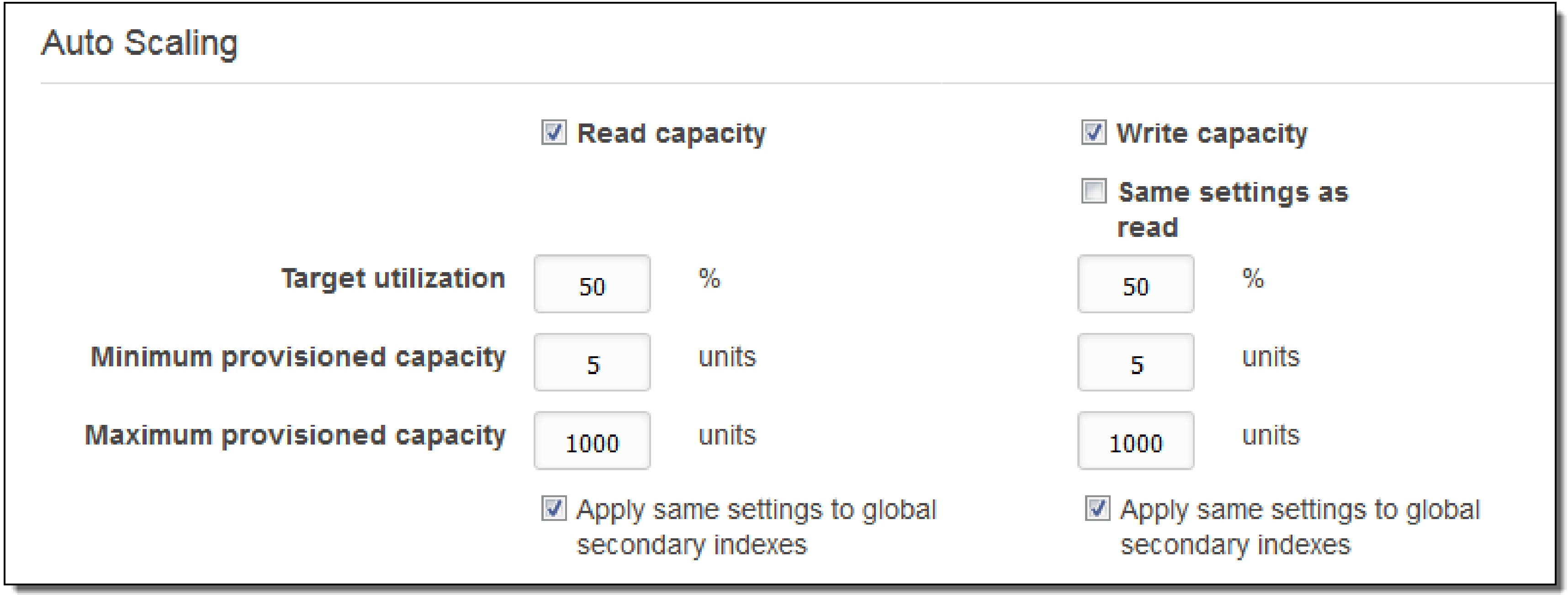
Total Today $2,345 +10%

Total Yesterday $2,123

Reduced their costs by over **$200K per year**, while also simplifying their application logic.

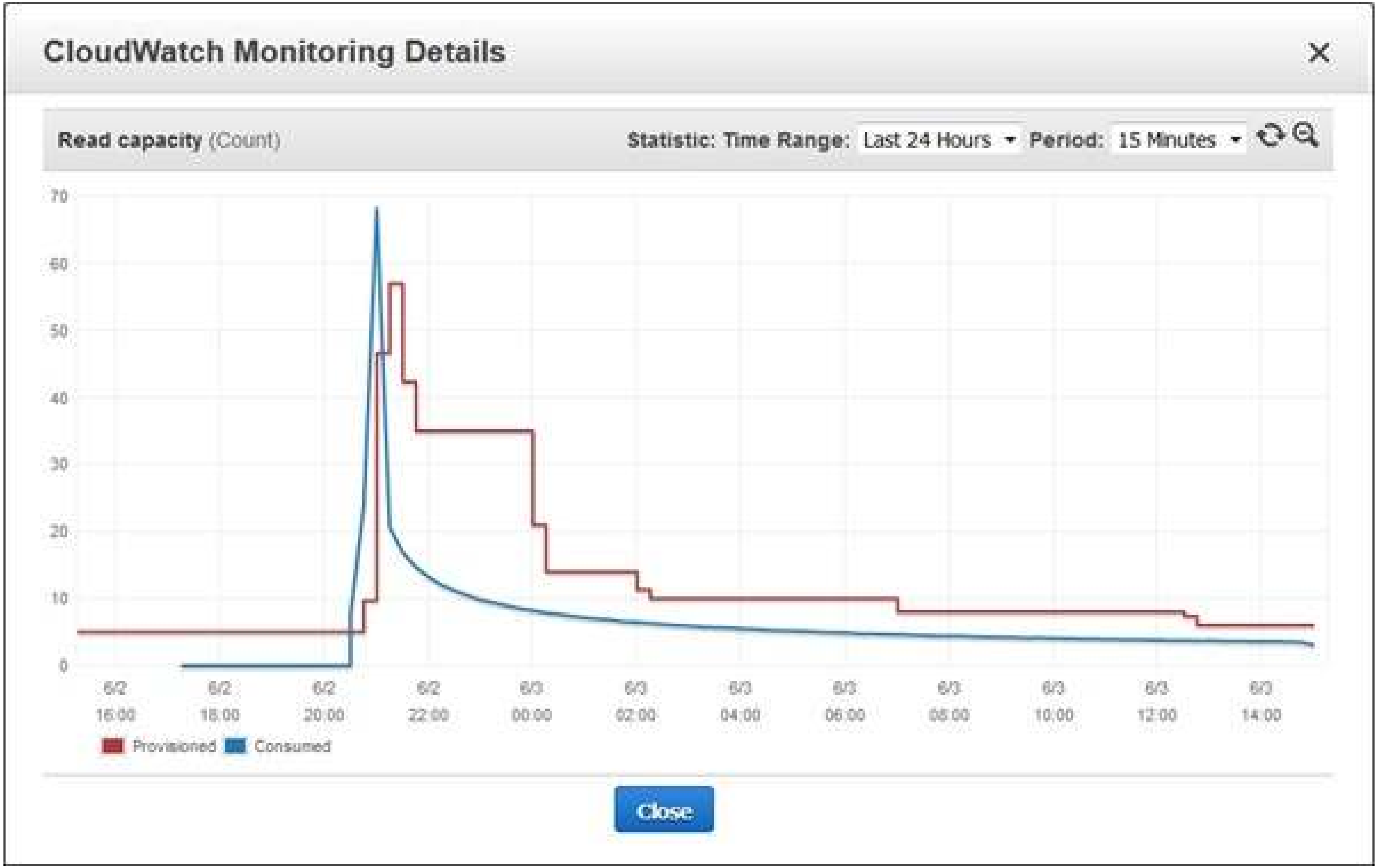
DynamoDB Auto Scaling

Adapting Read and Write Capacity to load



Auto Scaling

	☑ Read capacity		☑ Write capacity	
			□ Same settings as read	
Target utilization	50	%	50	%
Minimum provisioned capacity	5	units	5	units
Maximum provisioned capacity	1000	units	1000	units
	☑ Apply same settings to global secondary indexes		☑ Apply same settings to global secondary indexes	



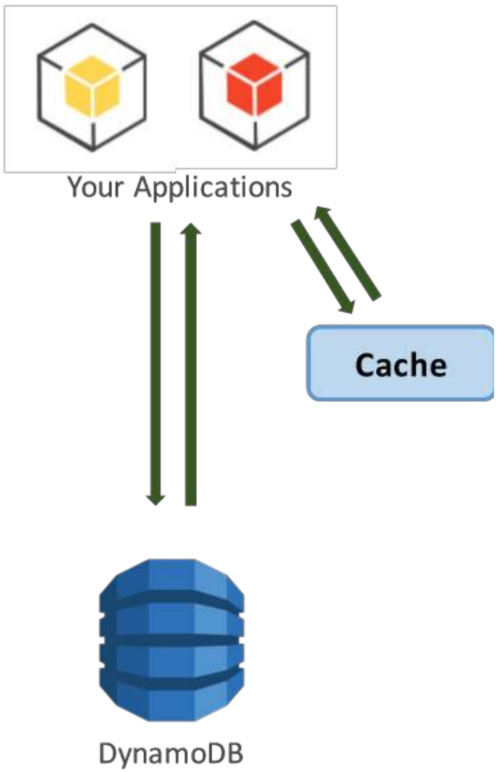
CloudWatch Monitoring Details

Read capacity (Count)

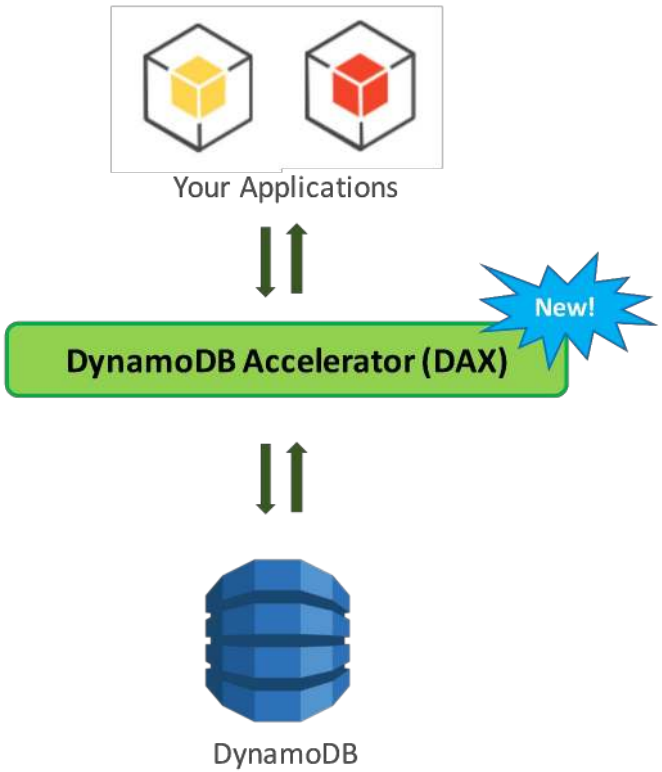
Statistic: Time Range: Last 24 Hours Period: 15 Minutes

DynamoDB Accelerator (DAX)

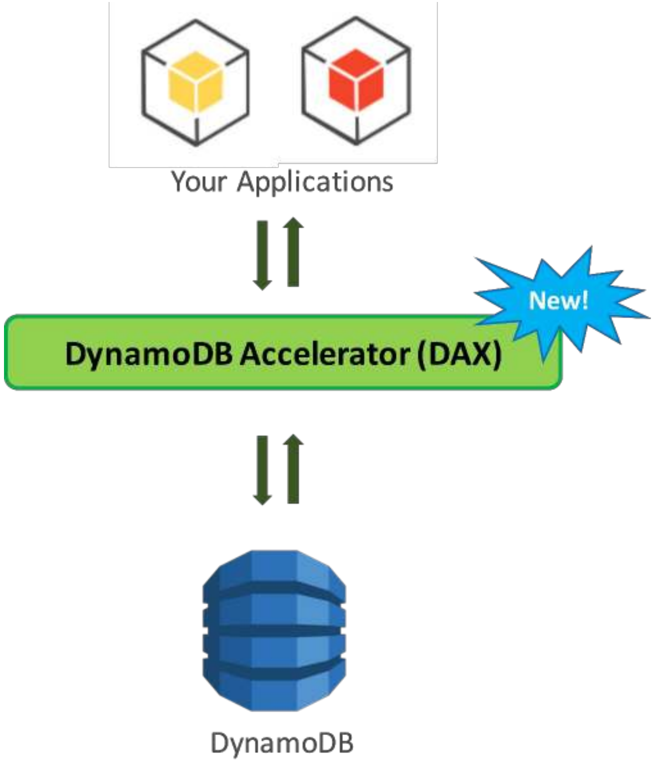
Traditional side cache



Caching made simple



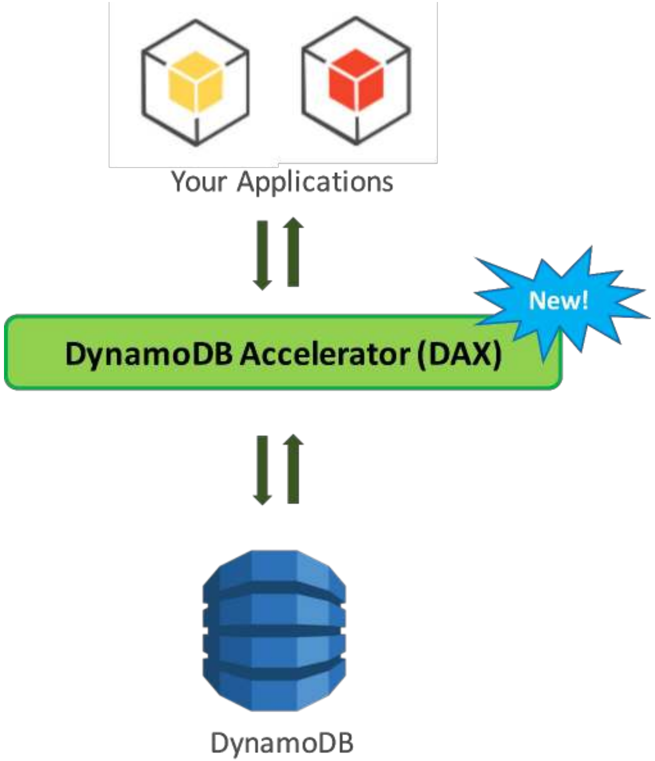
DynamoDB Accelerator (DAX)



Features

- **Fully managed:** handle all of the upgrades, patching, and software management
- **Flexible**: Configure DAX for one table or many
- **Highly available:** fault tolerant, replication across multi-AZs within a region
- **Scalable**: scales-out to any workload with up to 10 read replicas
- **Manageability:** fully integrated AWS service: Amazon CloudWatch, Tagging for DynamoDB, AWS Console
- **Security:** Amazon VPC, AWS IAM, AWS CloudTrail, AWS Organizations

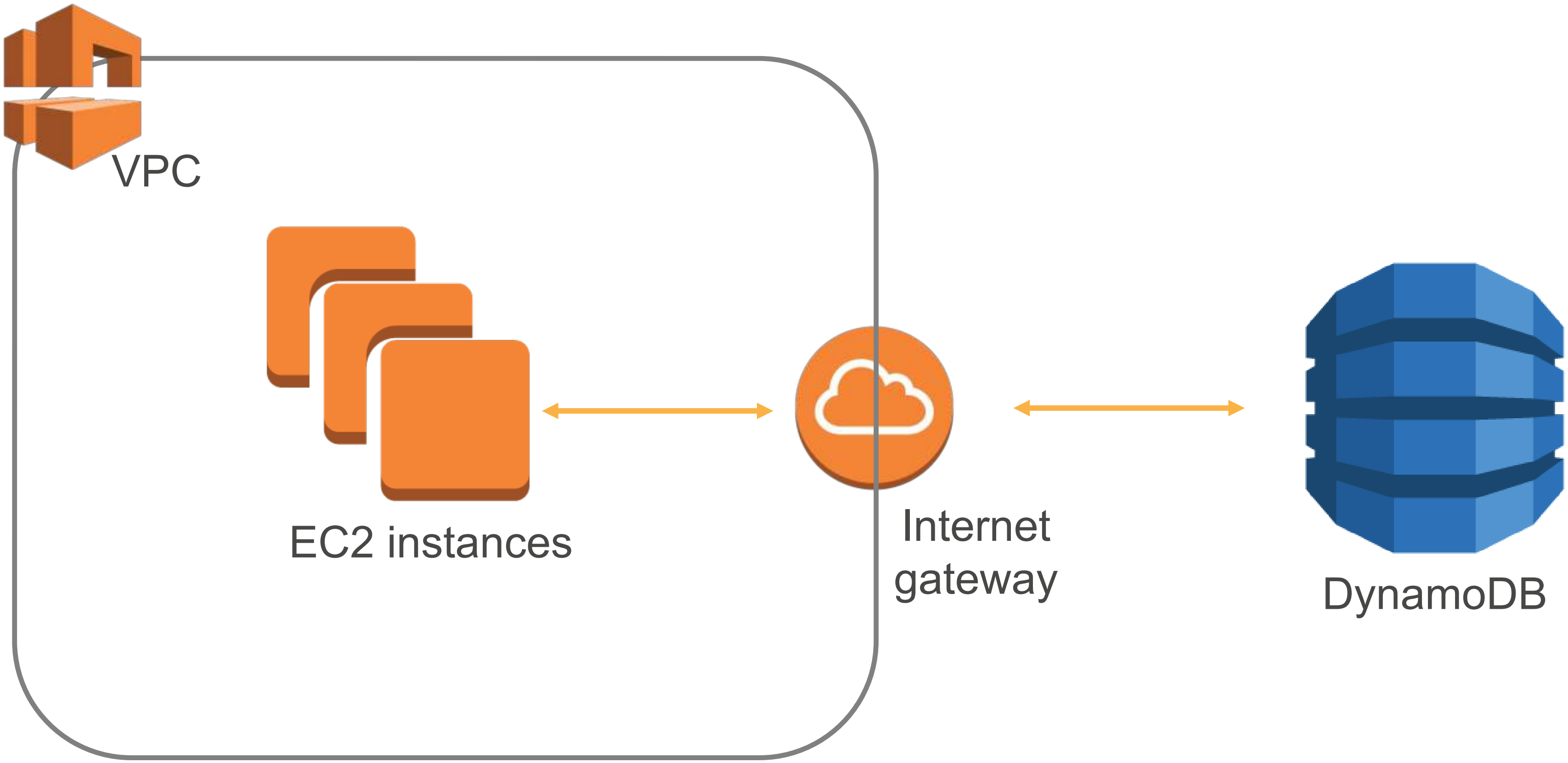
DynamoDB Accelerator (DAX)



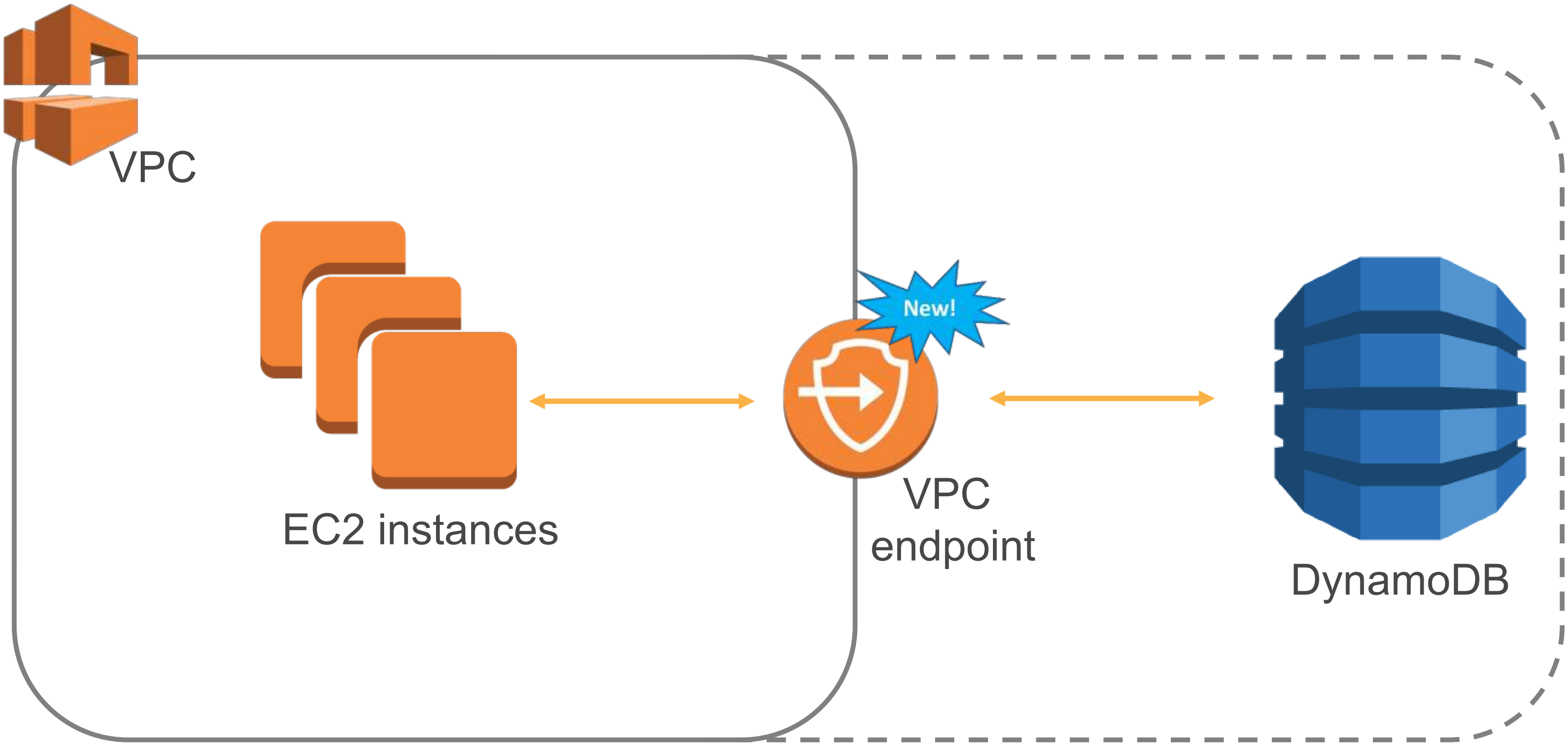
Key Benefits

- **Fast performance:** Microseconds response times at millions of reads/sec from single DAX cluster
- **Ease of use:** DynamoDB API compatible - requires minimal code change for existing applications, simplifying developer experience
- **Lower costs:** Reduce provisioned read capacity for DynamoDB tables for tables with hot data

DynamoDB VPC Endpoints

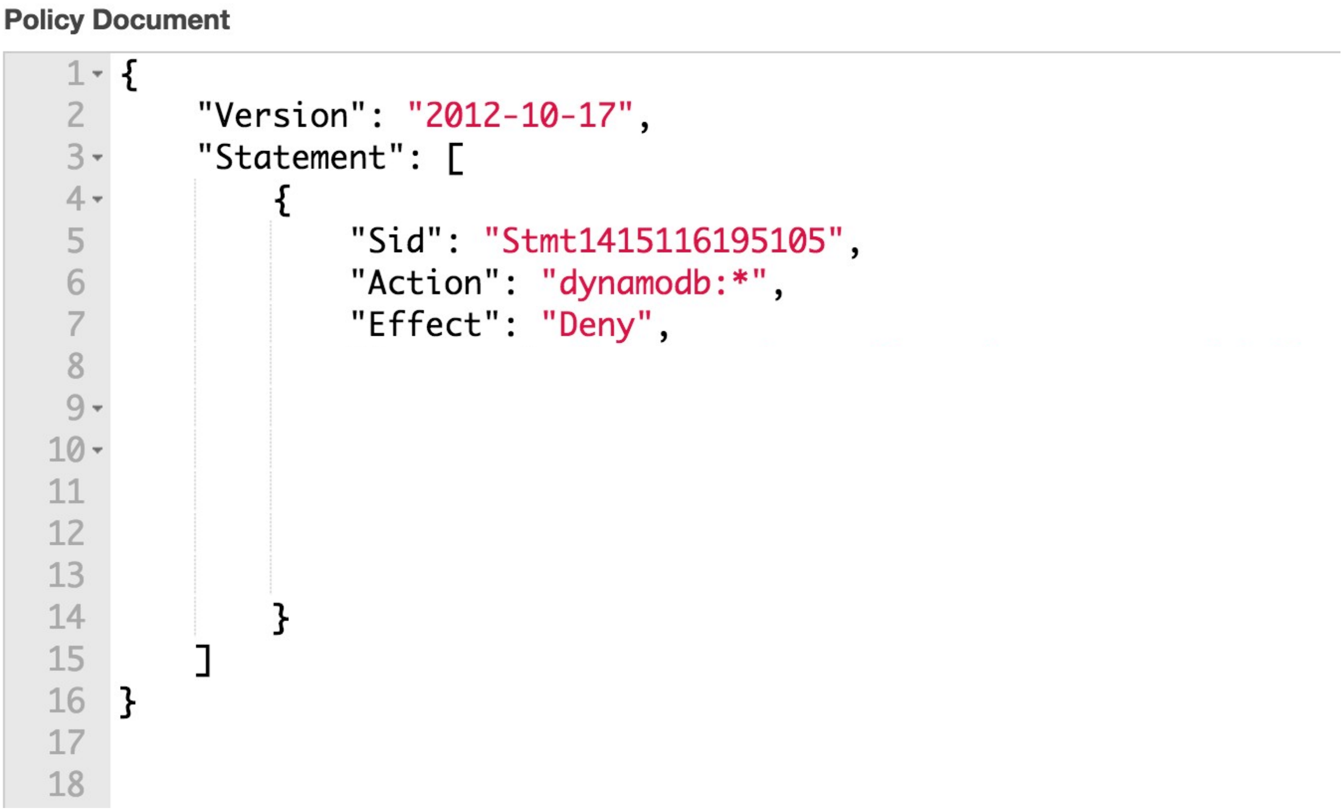

VPC
EC2 instances
Internet gateway
DynamoDB

VPC Endpoints for DynamoDB (VPC-E)



VPC Endpoints for DynamoDB (VPC-E)

Use an IAM policy to restrict an IAM user, group, or role to a particular VPC-E for DynamoDB tables



Policy Document

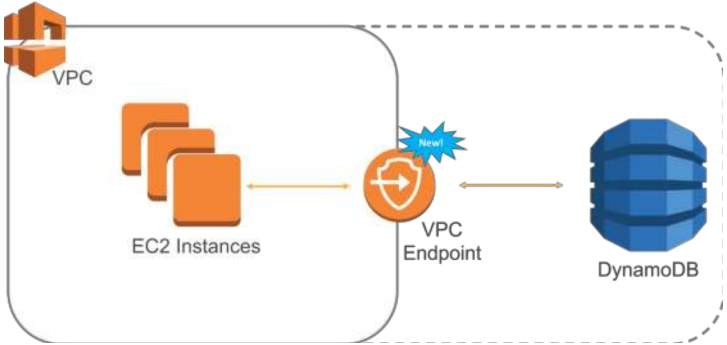
```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Sid": "Stmt1415116195105",
                        "Action": "dynamodb:*",
                        "Effect": "Deny",






                }
        ]
}
```

VPC Endpoints for DynamoDB (VPC-E)



Features

- **VPC:** Access DynamoDB via secure Amazon VPC endpoint
- **Access Control:** restrict table access for each VPC endpoint with a unique IAM role and permissions

Key Benefits

- Turn off access from public Internet gateways enhancing privacy and security
- Fast, secure data transfer between Amazon VPC and DynamoDB

DynamoDB in a nutshell

Scalability

- Millions of reads/sec from single DAX cluster
- Unlimited items and storage

Performance

- Consistent, single digit millisecond latency
- Optimized for analytics workloads with native indexing
- Microsecond response times with DynamoDB Accelerator (DAX)

Security

- Control user access at items and attributes level
- SOC, PCI, ISO, FedRAMP (Mod & High), HIPAA BAA
- Monitor with CloudWatch metrics & logging with CloudTrail
- Client-side encryption library
- Secure, private VPC endpoints

Availability

- Designed for 99.99% high availability (HA)
- Built-in replication across 3 zones

Manageability

- Fully-managed
- Perpetual free tier
- Pay-as-you-grow for capacity and storage independently
- Track table level spending with Tagging
- Purge data automatically (Time To Live)
- DMS connector for DynamoDB

Developer Platform

- Event-driven programming with Triggers & Lambda
- Advanced analytics with EMR & Amazon Redshift
- Full-text query support with Amazon Elasticsearch Service
- Real-time stream processing with Amazon Kinesis

Additional resources

Deep Dive on DynamoDB
https://fr.slideshare.net/AmazonWebServices/srv404-deep-dive-on-amazon-dynamodb-78419698

AWS blog https://aws.amazon.com/blogs/aws/category/amazon-dynamo-db/

AWS Database blog https://aws.amazon.com/blogs/database/category/dynamodb/

Picking the right AWS backend for your Java application
https://www.youtube.com/watch?v=u_LaoaJH9Jo

AWS SUMMIT

Thank you!

Julien Simon, Principal Technical Evangelist, AWS
@julsimon

amazon web services