Migrating from Cassandra to Amazon DynamoDB

Edin Zulich
NoSQL Solutions Architect

May 2018

aws

Agenda

- Cassandra/DynamoDB overview
- Why customers migrate from Cassandra
- Benefits of migrating to DynamoDB
- Migration process
- FAQ
- Summary
- References

Cassandra/DynamoDB Overview

- Both NoSQL, inspired by Dynamo
- Both used for similar use cases and requirements
  - Scalable NoSQL store
  - Performance
  - Time-series data
  - Global reach
- Both API-driven
- Different Capacity and cost model
  - Instance based vs. request based
  - Manage your cluster vs. consume a fully-managed service

aws

Why Customers Migrate from Cassandra

gumgum

"…when the company achieved a moderate traffic scale, the database started becoming unstable and presented us with occasional mood swings (here I refer to long running heavy compactions, infrastructure failures, buggy software patching etc.) leading to outages."

Anirban Roy
GumGum
https://techblog.gumgum.com/articles/moving-to-amazon-dynamodb-from-hosted-cassandra

"The database maintenance overhead slowed down the overall progress. Additional time spent scaling, upgrading and maintaining a database cluster removed time away from adding features or improving service code."

Zack Owens
Cloud Architect, Nike
https://medium.com/nikeengineering/becoming-a-nimble-giant-how-dynamo-db-serves-nike-at-scale-4cc375dbb18e

aws

Why Customers Migrate from Cassandra

Operational challenges grow with scale

- Resulting in increased complexity, resources and cost
- This is true for all distributed/clustered databases

aws

Why DynamoDB? Here's an example:

AWS Blog

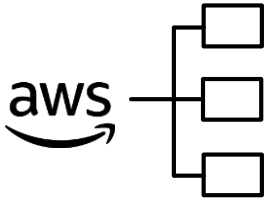
Prime Day 2017 – Powered by AWS

by Jeff Barr | on 14 SEP 2017 | in Case Studies | Permalink | Share

NoSQL Database – Amazon DynamoDB requests from Alexa, the Amazon.com sites, and the Amazon fulfillment centers totaled 3.34 trillion, peaking at 12.9 million per second. According to the team, the extreme scale, consistent performance, and high availability of DynamoDB let them meet needs of Prime Day without breaking a sweat.

Amazon DynamoDB

Fully managed nonrelational database for any scale



Fully managed

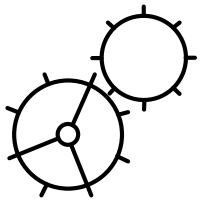
Maintenance-free

Serverless

Auto scaling

Backup and restore

Global tables

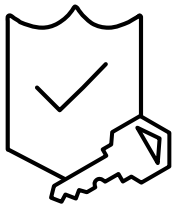


Scale and performance

Fast, consistent performance

Virtually unlimited throughput

Virtually unlimited storage



Secure

Encryption at rest and in transit

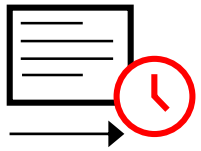
VPC Endpoints

Fine-grained access control

PCI, HIPAA, FIPS140-2 eligible

DynamoDB over the last year

February 2017



Time to Live (TTL)

April 2017

VPC Endpoints

April 2017

DynamoDB Accelerator (DAX)

June 2017

Auto Scaling

December 2017

Global Tables

December 2017, March 2018

Backup and recovery

February 2018

Encryption at rest

+ Adaptive Capacity

DynamoDB Global Tables

Fully managed, multi-master, multi-region database

Build high performance, globally distributed applications

Low latency reads & writes to locally available tables

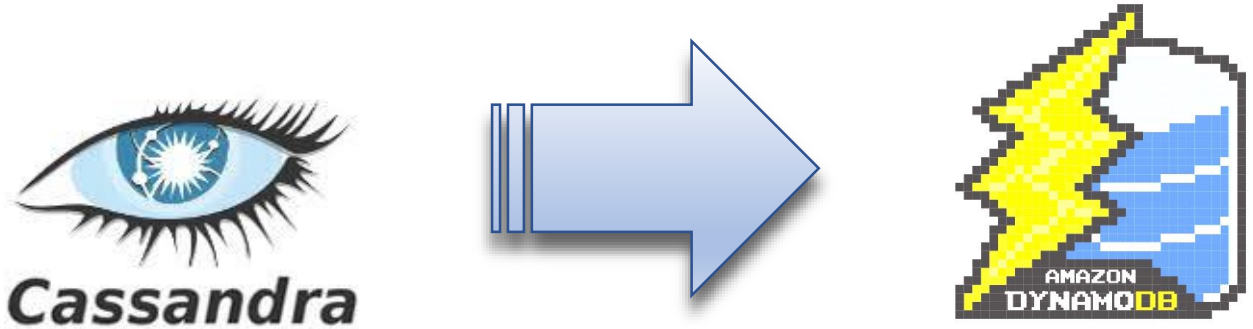
Disaster proof with multi-region redundancy

Easy to setup and no application re-writes required

Benefits of migrating to DynamoDB

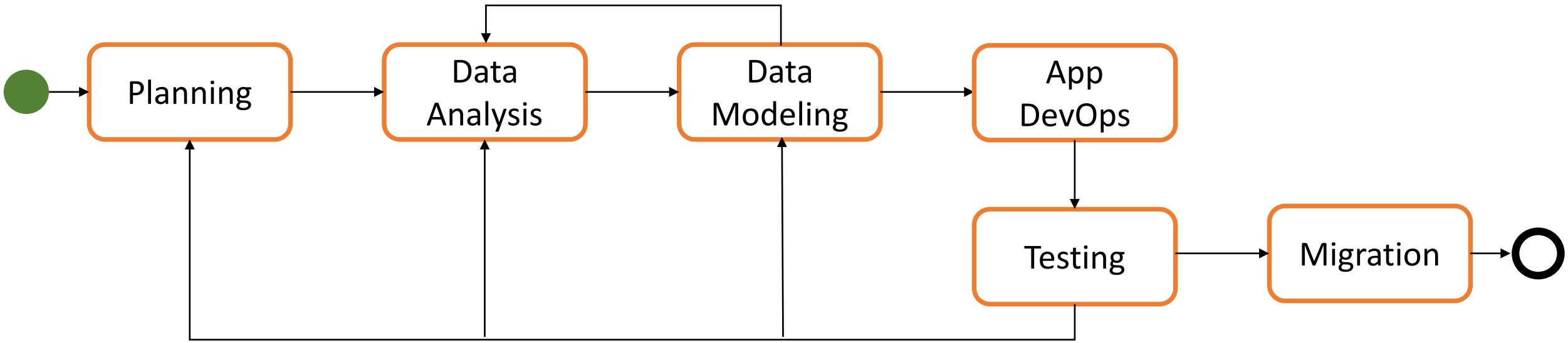
- Stability, performance at scale
  - Single-digit millisecond latency for reads and writes at any scale
- DynamoDB elastic provisioning
- Zero maintenance and operations overhead
- 30% to 70% cost savings over Cassandra

aws



Migration Process

A Phased Approach to Migration



It’s more of an application migration than a data migration

Example: SAMSUNG

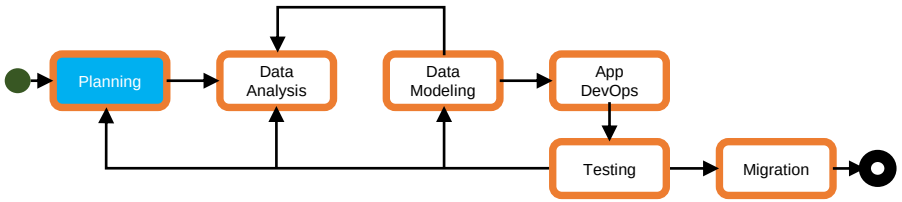
"DynamoDB provided consistent high performance at a drastically lower cost than Cassandra."

—Seongkyu Kim
Samsung

Samsung Cloud Service

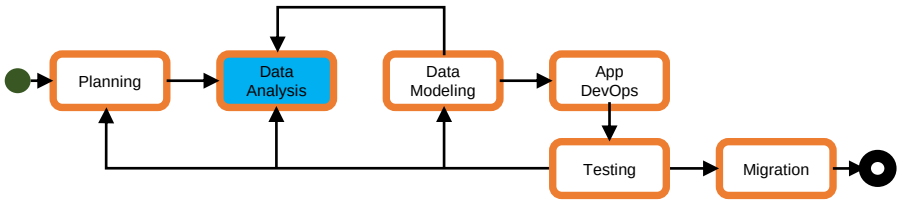
- Backup and restore and key value store for mobile app
- 300 million users
- Entire migration process took ~12 mo.
- Almost 1 PB in DynamoDB, 130 million daily API requests
- Consistent performance and 70% cost savings (TCO)

Planning



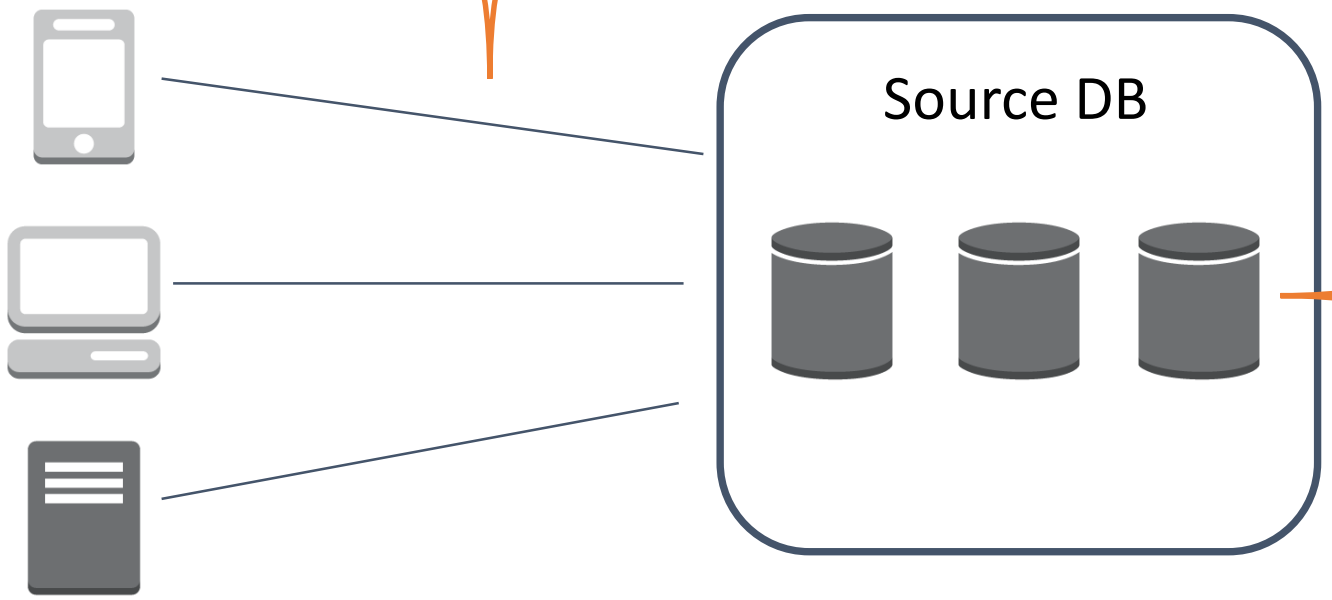
- Define requirements and goals of the migration
  - Application specific
  - E.g.: Is downtime allowed for cutover? If so, how long?
  - Opportunity for “spring cleaning”
- Document per-table requirements and challenges
- Define and document backup and restore strategies

Data Analysis



Application access patterns

- Writes and updates
- LightWeight transactions
- Queries

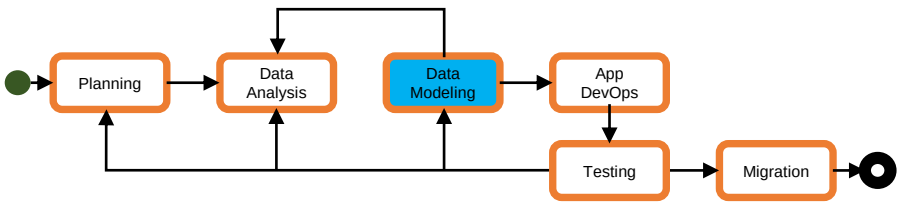


Source Data Analysis

Key data attributes:

- Number of items to be migrated
- Distribution of the item sizes
- Multiplicity of values to be used as partition or sort keys
- Data lifecycle

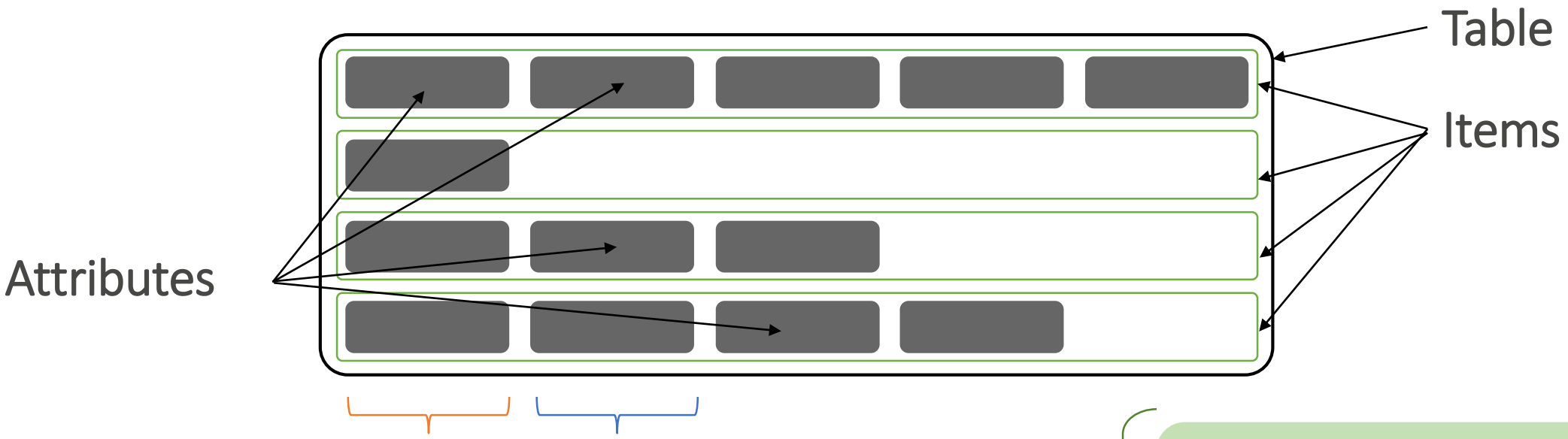
Data Modeling and Capacity Planning



Data Modeling

- Work from access patterns
  - Define instantiated views for your access patterns
- Define a partitioning (scaling) scheme
  - Likely the same as in Cassandra
- Define data structure
  - What are records (items in DynamoDB) going to look like?
  - May not be the same as in Cassandra
- Streams and Triggers

DynamoDB fundamentals: Table



Mandatory
Key-value access pattern
Determines data distribution

Optional
Model 1:N relationships
Enables rich query capabilities

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

DynamoDB Sort key ⇔ Cassandra clustering key

DynamoDB fundamentals: data types

Type	DynamoDB Type
String	String
Integer, Float	Number
Timestamp	Number or String
Blob	Binary
Boolean	Bool
Null	Null
List	List
Set	Set of String, Number, or Binary
Map	Map

DynamoDB: provisioned throughput capacity

Per table/GSI

Capacity is per second, rounded up to the next whole number

Read Capacity Unit (RCU)

1 RCU returns 4KB of data for strongly consistent reads, or double the data for eventually consistent reads

Write Capacity Unit (WCU)

1 WCU writes 1KB of data, and *each item consumes 1 WCU minimum*

Modeling hierarchical data: item hierarchies…

- Use composite sort key to define a hierarchy
- Highly selective result sets

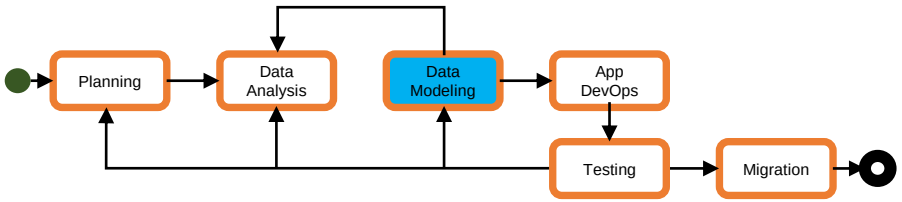
	Primary Key		Attributes						
	ProductID	type							
Items	1	bookID	title	author	genre	publisher	datePublished	ISBN	
			Some Book	John Smith	Science Fiction	Ballantine	Oct-70	0-345-02046-4	
	2	albumID	title	artist	genre	label	studio	released	producer
			Some Album	Some Band	Progressive Rock	Harvest	Abbey Road	3/1/73	Somebody
	2	albumID:trackID	title	length	music	vocals			
			Track 1	1:30	Mason	Instrumental			
	2	albumID:trackID	title	length	music	vocals			
			Track 2	2:43	Mason	Mason			
	2	albumID:trackID	title	length	music	vocals			
			Track 3	3:30	Smith	Johnson			
	3	movieID	title	genre	writer	producer			
			Some Movie	Scifi Comedy	Joe Smith	20th Century Fox			
	3	movieID:actorID	name	character	image				
			Some Actor	Joe	img2.jpg				
	3	movieID:actorID	name	character	image				
			Some Actress	Rita	img3.jpg				
	3	movieID:actorID	name	character	image				
			Some Actor	Frito	img1.jpg				

… or documents (JSON)

- JSON data types (M, L, BOOL, NULL)
- Document SDKs available
- 400 KB maximum item size (limits hierarchical data structure)

	Primary Key	Attributes							
	ProductID								
Items	1	id	title	author	genre	publisher	datePublished	ISBN	
		bookID	Some Book	Some Guy	Science Fiction	Ballantine	Oct-70	0-345-02046-4	
	2	id	title	artist	genre	Attributes			
		albumID	Some Album	Some Band	Progressive Rock	{ label:"Harvest", studio: "Abbey Road", published: "3/1/73", producer: "Pink Floyd", tracks: [{title: "Speak to Me", length: "1:30", music: "Mason", vocals: "Instrumental"},{title: ”Breathe", length: ”2:43", music: ”Waters, Gilmour, Wright", vocals: ”Gilmour"},{title: ”On the Run", length: “3:30", music: ”Gilmour, Waters", vocals: "Instrumental"}]}			
	3	id	title	genre	writer	Attributes			
		movieID	Some Movie	Scifi Comedy	Joe Smith	{ producer: "20th Century Fox", actors: [{ name: "Luke Wilson", dob: "9/21/71", character: "Joe Bowers", image: "img2.jpg"},{ name: "Maya Rudolph", dob: "7/27/72", character: "Rita", image: "img1.jpg"},{ name: "Dax Shepard", dob: "1/2/75", character: "Frito Pendejo", image: "img3.jpg"}]			

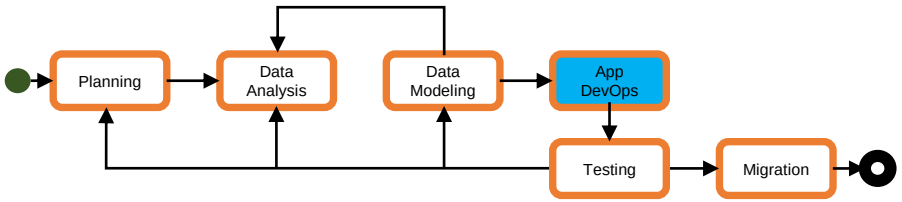
Data Modeling and Capacity Planning



Capacity Planning

- Reads, writes, and storage for tables and GSI's
- Cost considerations
- Streams
- Migration vs. post-migration capacity
  - Is there an initial data import phase?
  - Provision capacity for import during migration, then for normal operation

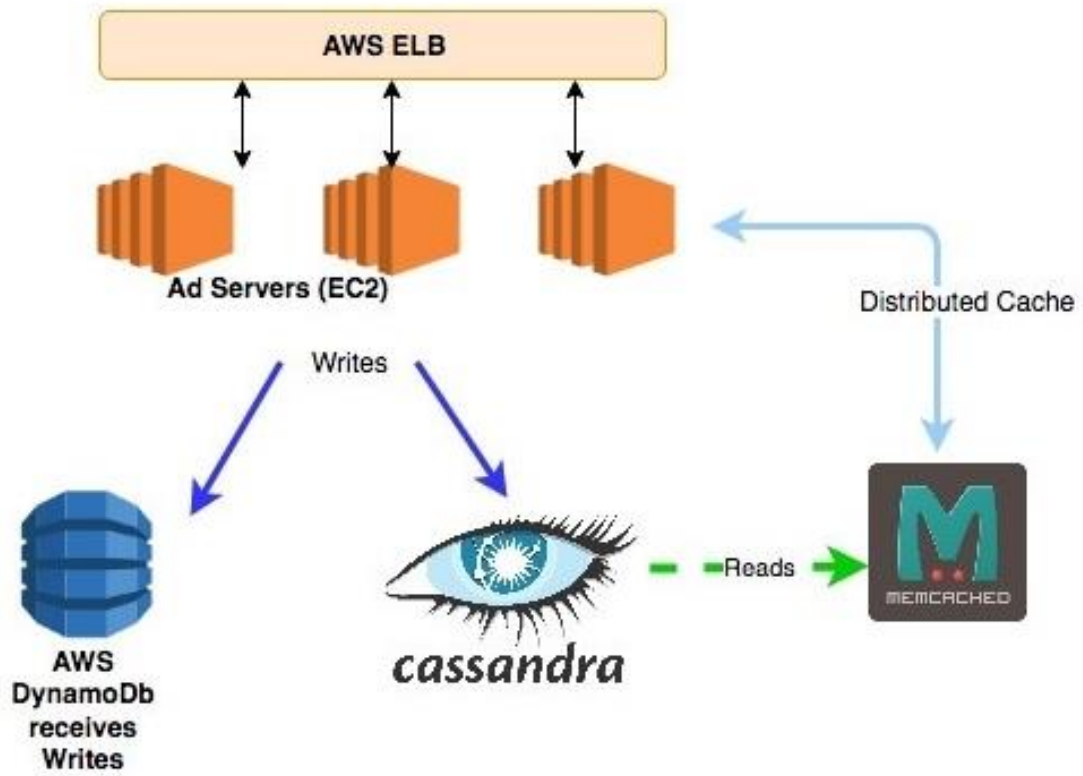
Application Development and Operations



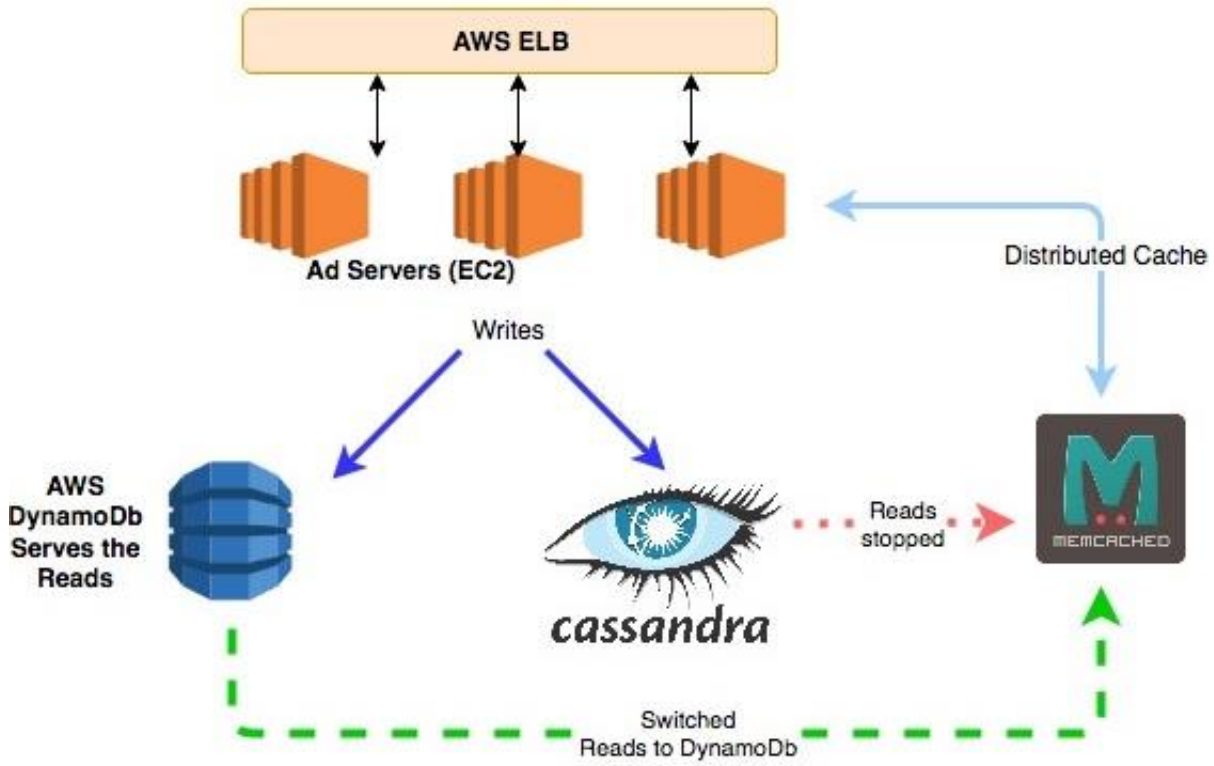
Development and Testing

- Data access layer for DynamoDB
- Dynamic application configuration
  - Write/read to/from both source and target database
  - Support gradual switchover
- Test rollback, backup/restore

Example: gumgum



Migration strategy 2: Read data from Cassandra, but write in parallel to both Cassandra and DynamoDb for at least 30 days.



Migration strategy 2: After ~40 days of writes, reads to Cassandra are stopped and started from DynamoDb.

Source: https://techblog.gumgum.com/articles/moving-to-amazon-dynamodb-from-hosted-cassandra

DynamoDB API

Table and Item API

Admin	CRUD
Create Table	Put/Get Item
Update Table	Batch Put/Get Item
Delete Table	Update Item
Describe Table	Delete Item
	Query
	Scan

Streams API

DynamoDB Streams
ListStreams
DescribeStream
GetShardIterator
GetRecords

DynamoDB API Notes

- Conditional writes/updates
- *ConditionalCheckFailedException*
- *ConsistentRead* parameter
  - On a per request basis when using *GetItem, Scan, Query*
- Filtering (*FilterExpression*)
- Sort order (*ScanIndexForward*)
- *Limit*
- Pagination (*LastEvaluatedKey, ExclusiveStartKey*)
- *ThrottlingException* and *ProvisionedThroughputExceededException*
- *ReturnConsumedCapacity*

aws

Rich Expressions

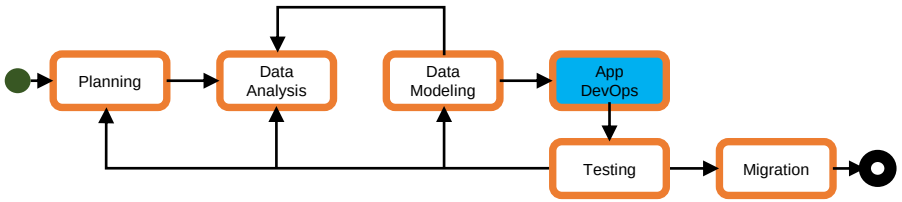
- Projection expression
  - Query/Get/Scan: ProductReviews.FiveStar[0]
- Filter expression
  - Query/Scan: #V > :num (#V is a place holder for keyword VIEWS)
- Conditional expression
  - Put/Update/DeleteItem: attribute_not_exists (#pr.FiveStar)
- Update expression
  - UpdateItem: set Replies = Replies + :num

DynamoDB Error Handling

- HTTP 400
  - A problem with request
  - Common: ProvisionedThroughputExceededException
    - Use exponential backoff to retry
  - ConditionalCheckFailedException
- HTTP 500
  - A problem on the service side
  - OK to retry immediately or after a short delay
- Retries and exponential backoff
  - Enabled and handled by SDK by default
  - Understand the backoff strategies – e.g. in Java SDK: PredefinedBackoffStrategies.java
  - Consider disabling the default and implementing your own
  - Log and monitor failed requests

aws

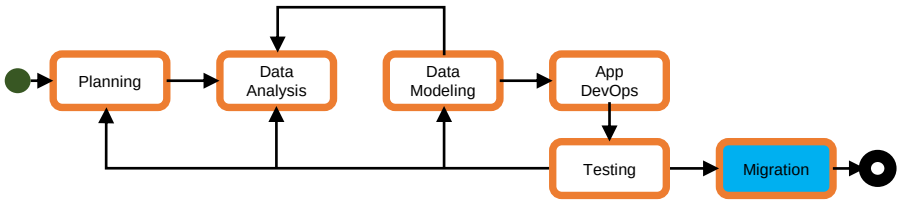
Application Development and Operations



Operations

- Security/access control via AWS IAM
- Deployment via AWS CloudFormation
- Monitoring via Amazon CloudWatch, AWS CloudTrail, AWS Config
  - Or use third-party offerings
  - Custom application metrics

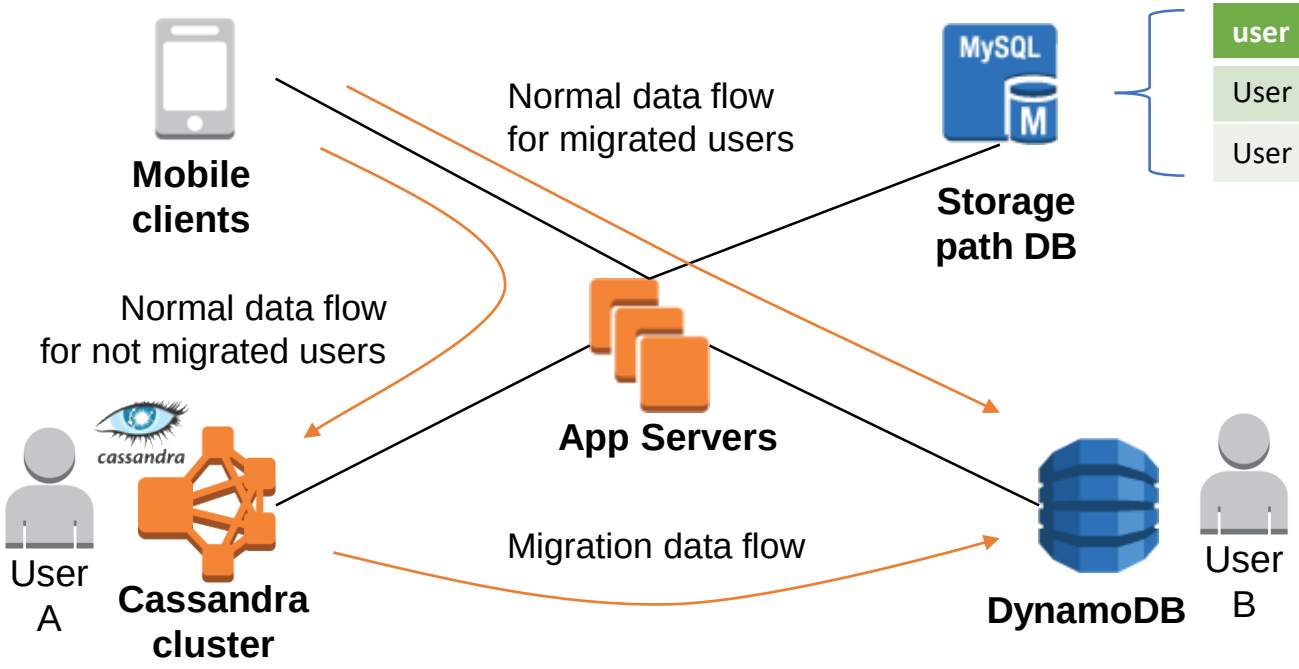
Data Migration



- Approach depends on application
- Examples
  - Data lifecycle-based: data stored for a set amount of time
    - Phase 1: write to both, read from Cassandra
    - Phase 2: after a full cycle start reading from DynamoDB
  - Incremental
    - E.g. user-based
      - Migrate users over time, each user as quickly as possible
  - Batch export/import + live updates
    - Use EMR to import data into DynamoDB
      - Provision enough write capacity
      - Keep read capacity down
      - Tune EMR cluster instance type and size, DynamoDB write ratio

Example: Samsung

- **Online migration**
- Full migration is not possible (several 100s of TB sized tables) : **Per user migration**
- Some users are in Cassandra while others are in DynamoDB : **Storage path DB**
- To minimize impact for each user, migrate as soon as possible : **Accelerate migration**



user	storage path
User A	*<path_to_cassandra>*
User B	*<path_to_dynamodb>*

Benefits of migrating to DynamoDB: SAMSUNG

- Successfully launched Samsung Cloud service supporting massive scale workloads for **Samsung Galaxy Smart Phones**
- **40% cost saving** in NoSQL infrastructure cost
- No capacity planning for **Peta-byte Scale** Storage Capacity with on-demand capacity
- Consistent performance at **10s of Millions** of operations
- Zero administration for **Hundreds of Tables** with DynamoDB Auto Scaling
- No failures during **2 years Operation**
- No data corruption or loss for **Billions of Items**
- **Enterprise Level Security and Compliance** using VPC Endpoints for DynamoDB

Benefits of migrating to DynamoDB: gumgum

- Stability, performance at scale
  - ~2-3ms read and ~4-6ms write latency
- DynamoDB elastic provisioning via auto scaling
- Zero maintenance and operations overhead
- 65-70% TCO savings over Cassandra

> This was a *big win* for us considering the fact that there will be no additional infrastructure to run and no more maintenance activities required from an engineers end. Given the current state we look forward to migrate all our existing data from Apache Cassandra to Amazon DynamoDB in the recent future.

Anirban Roy
GumGum
https://techblog.gumgum.com/articles/moving-to-amazon-dynamodb-from-hosted-cassandra

aws

Cassandra to DynamoDB FAQ

- Q: We are using Cassandra to store time-series data, and DynamoDB is a key-value store, so it's not efficient for time-series data.

  DynamoDB is as efficient at storing time-series data as Cassandra is. DynamoDB supports composite primary keys (partition and sort key), and when the sort key is a timestamp, DynamoDB stores data grouped and sorted by time, allowing for efficient access by time and time ranges.

- Q: Does DynamoDB have tombstones like Cassandra?

  No, DynamoDB does not use tombstones so there aren't performance issues arising from them.

- Q: Does DynamoDB do compactions like Cassandra?

  No, DynamoDB does not do compactions, and so there is no risk of performance degradation due to those.

- Q: Can DynamoDB latency be affected by JVM garbage collection?

  No. With respect to all three questions above, the key thing to know is that, as a fully-managed service, DynamoDB manages the resources for you to deliver consistent

Cassandra to DynamoDB FAQ

- Q: We had a hard time with data consistency with Cassandra. Won't we have the same problem with DynamoDB?

  No. DynamoDB manages consistency for you and provides straightforward options for reads, as well as conditional update API that can be used to implement transactional behavior.

- Q: We have rows in Cassandra with many columns, and they are large in size. Doesn't DynamoDB have a limit on row size?

  Yes, DynamoDB limits each row ("item") size to 400KB. Wide rows from Cassandra can be modeled as item hierarchies (multi-row collections) in DynamoDB, or JSON documents, depending on access patterns. Also, consider using compression for large fields/data that is not queried, and/or storing large objects in S3.

- DynamoDB does not have a date/time data type. What should we use instead?

  Date/time values should be stored as numbers (in epoch format) if they will be used with DynamoDB TTL. They can also be stored as strings (e.g. in ISO 8601 to allow for sorting).

From Cassandra to DynamoDB: Summary

- Trading one set of idiosyncrasies for another…
  - From thinking about instances to thinking about read/write capacity
  - From Cassandra column families to DynamoDB tables and item hierarchies
    - DynamoDB item size is limited to 400KB
  - Store date/time data as Numbers or Strings in DynamoDB

- Benefits of DynamoDB
  - No database maintenance, no clusters to manage
  - Effortless scaling
  - Stability and consistent performance at any scale
  - In all AWS regions around the world
  - Agility
  - Cost savings

aws

From Cassandra to DynamoDB: Summary

- Keys to Success
  - Understand the source data and access patterns
  - Understand capacity and cost and how scaling affects it
  - Test thoroughly and often
  - Plan on an iterative migration process
  - Learn from documented cases

aws

References

- DynamoDB Best Practices
  https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-practices.html
- Samsung Cassandra to DynamoDB Migration
  - https://www.youtube.com/watch?v=Z-2UIrI9feQ#t=01m44s
- Moving to Amazon DynamoDB from Cassandra: A Leap Towards 60% Cost Saving per Year
  - http://techblog.gumgum.com/articles/moving-to-amazon-dynamodb-from-hosted-cassandra
- Becoming a Nimble Giant: How DynamoDB Serves Nike at Scale
  - https://medium.com/nikeengineering/becoming-a-nimble-giant-how-dynamo-db-serves-nike-at-scale-4cc375dbb18e
- Why Druva Moved Away from Cassandra
  - https://www.druva.com/blog/why-druva-moved-away-from-cassandra/
  - https://aws.amazon.com/solutions/case-studies/druva/
- Why Tellybug Moved from Cassandra to Amazon DynamoDB
  - https://attentionshard.wordpress.com/2013/09/30/why-tellybug-moved-from-cassandra-to-amazon-dynamodb/

aws

Thank you