NoSQL @ Netflix

Siddharth "Sid" Anand

@r39132

QCon London 2011
March 2011

Coordinates

- Twitter : @r39132 #qconlondon2011
- Blog : practicalcloudcomputing.com
- White Paper : Netflix's Transition to High Availability Storage Systems

Netflix Intro

Netflix Intro

- Paid subscription service delivering video streaming and DVDs by mail
- 20M+ paying subscribers
- Fast becoming #1 video subscription business in the US
- ~200-300 engineers in Los Gatos, CA
- ~$2B revenue in 2010
- Expanding globally in the years to come

Motivation

Netflix's motivation for moving to the cloud

Motivation

- Circa late 2008, Netflix had a single data center
  - Single-point-of-failure (a.k.a. SPOF)
  - Approaching limits on cooling, power, space, traffic capacity
- Alternatives
  - Build more data centers
  - Outsource the majority of our capacity planning and scale out
    - Allows us to focus on core competencies

Motivation

- **Winner** : Outsource the majority of our capacity planning and scale out
  - Leverage a leading Infrastructure-as-a-service provider
    - Amazon Web Services

- **Footnote** : As it has taken us a while (i.e. ~2+ years) to realize our vision of running on the cloud, we needed an interim solution to handle growth
  - We did build a second data center along the way
  - We did outgrow it

Cloud Migration Strategy

What to Migrate?

NETFLIX

Cloud Migration Strategy

- Components
  - Applications and Software Infrastructure
  - Data
- Migration Considerations
  - Avoid sensitive data for now
    - PII and PCI DSS stays in our DC, rest can go to the cloud
  - Favor Web Scale applications & data

Cloud Migration Strategy

Examples of Data that can be moved

- Video-centric data
  - Critics' and Users' reviews
  - Video Metadata (e.g. director, actors, plot description, etc…)
- User-video-centric data – some of our largest data sets
  - Video Queue
  - Watched History
  - Video Ratings (i.e. a 5-star rating system)
  - Video Playback Metadata (e.g. streaming bookmarks, activity logs)

Cloud Migration Strategy

How and When to Migrate?

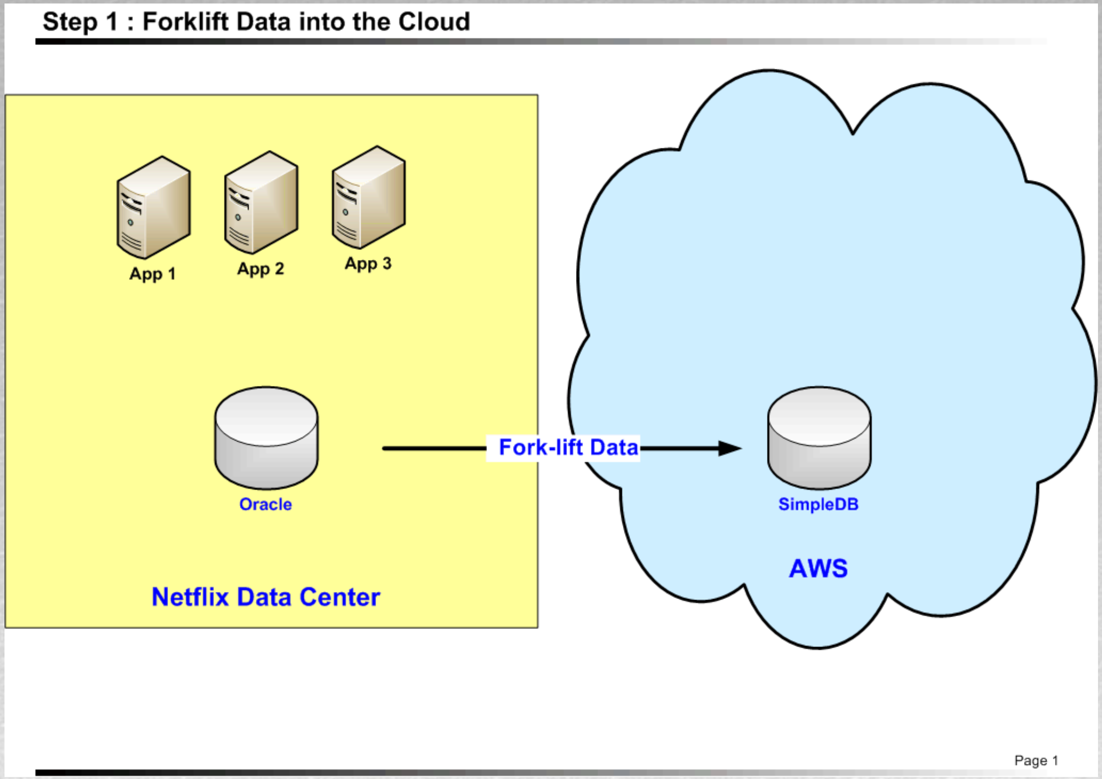
NETFLIX

Cloud Migration Strategy

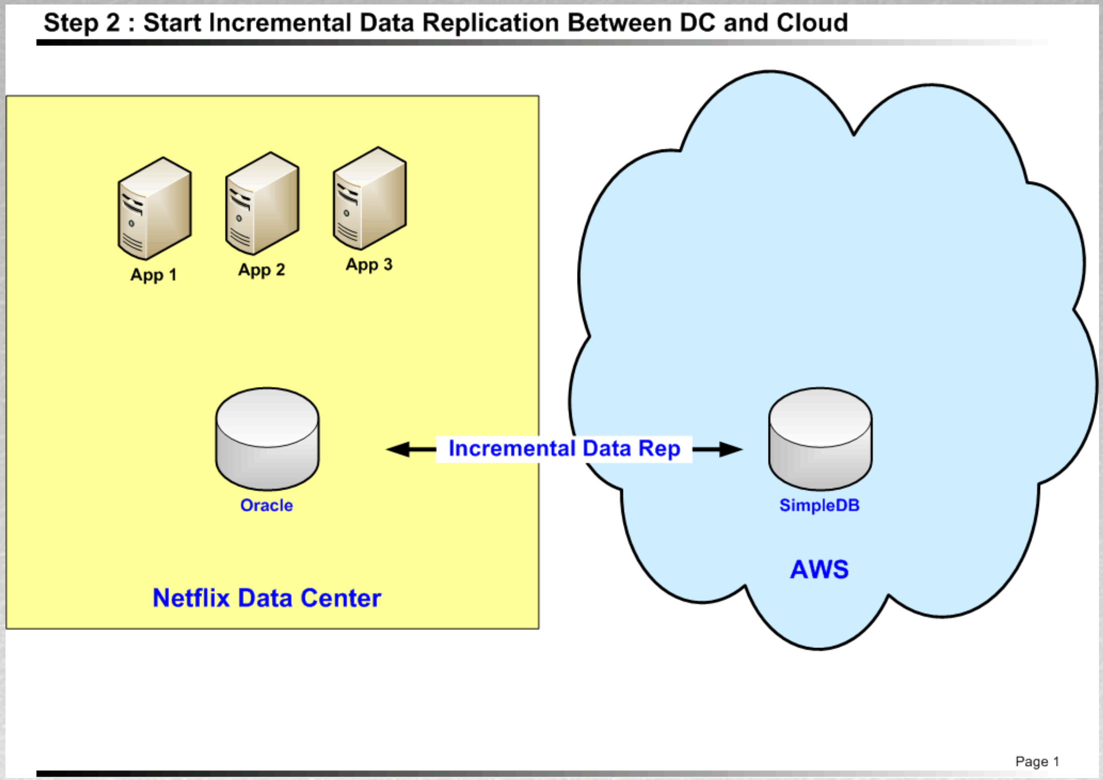
- High-level Requirements for our **Site**
  - No big-bang migrations
  - New functionality needs to launch in the cloud when possible

- High-level Requirements for our **Data**
  - Data needs to migrate before applications
  - Data needs to be shared between applications running in the cloud and our data center during the transition period

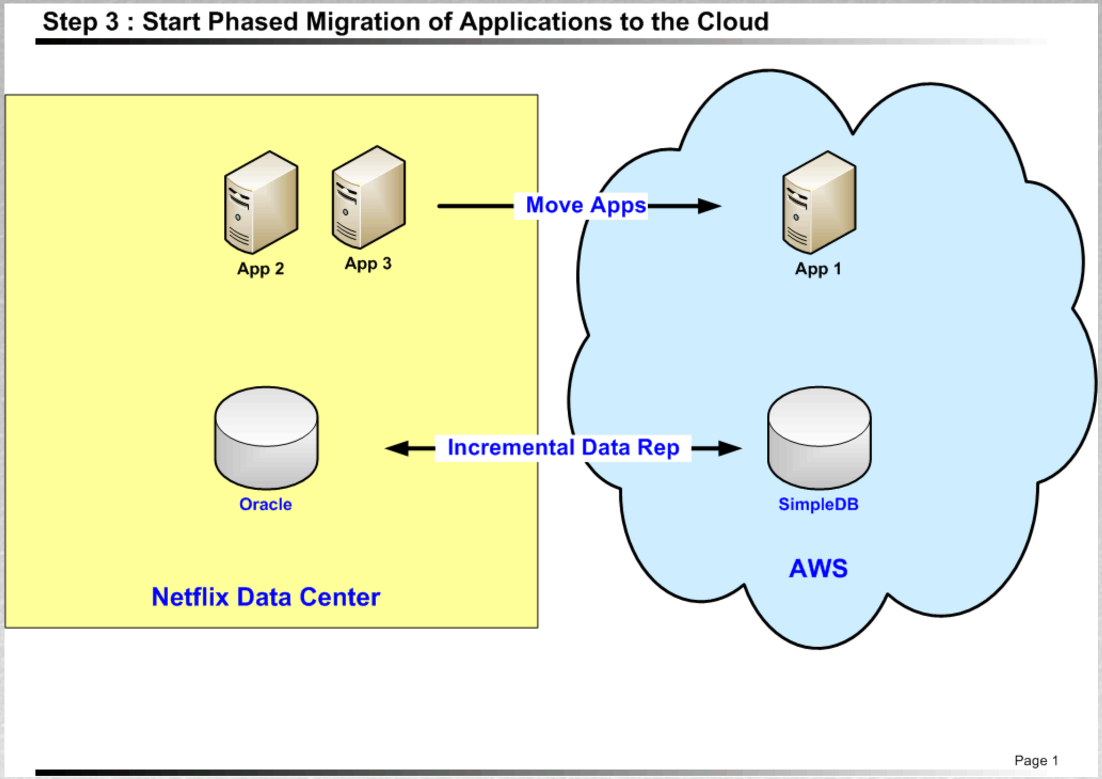
Cloud Migration Strategy



Cloud Migration Strategy



Cloud Migration Strategy



Cloud Migration Strategy

- Pick a (key-value) data store in the cloud
  - Challenges
    - Translate RDBMS concepts to KV store concepts
    - Work-around Issues specific to the chosen KV store
    - Create a bi-directional DC-Cloud data replication pipeline

Pick a Data Store in the Cloud

NETFLIX

Pick a Data Store in the Cloud

An ideal storage solution should have the following features:

- ☑ Hosted
- ☑ Managed Distribution Model
- ☑ Works in AWS
- ☑ AP from CAP
- ☑ Handles a majority of use-cases accessing high-growth, high-traffic data
  - ☑ Specifically, key access by customer id, movie id, or both

Pick a Data Store in the Cloud

- We picked SimpleDB and S3
  - SimpleDB was targeted as the AP equivalent of our RDBMS databases in our Data Center
  - S3 was used for data sets where item or row data exceeded SimpleDB limits and could be looked up purely by a single key (i.e. does not require secondary indices and complex query semantics)
    - Video encodes
    - Streaming device activity logs (i.e. CLOB, BLOB, etc…)
    - Compressed (old) Rental History

Technology Overview

SimpleDB

Technology Overview : SimpleDB

Terminology

SimpleDB	Hash Table	Relational Databases
Domain	Hash Table	Table
Item	Entry	Row
Item Name	Key	Mandatory Primary Key
Attribute	Part of the Entry Value	Column

Technology Overview : SimpleDB

Soccer Players				
Key	Value			
ab12ocs12v9	First Name = Harold	Last Name = Kewell	Nickname = Wizard of Oz	Teams = Leeds United, Liverpool, Galatasaray
b24h3b3403b	First Name = Pavel	Last Name = Nedved	Nickname = Czech Cannon	Teams = Lazio, Juventus
cc89c9dc892	First Name = Cristiano	Last Name = Ronaldo		Teams = Sporting, Manchester United, Real Madrid

SimpleDB's salient characteristics

- SimpleDB offers a range of consistency options
- SimpleDB domains are sparse and schema-less
- The Key and all Attributes are indexed
- Each item must have a unique Key
- An item contains a set of Attributes
  - Each Attribute has a name
  - Each Attribute has a set of values
  - All data is stored as UTF-8 character strings (i.e. no support for types such as numbers or dates)

Technology Overview : SimpleDB

What does the API look like?

- **Manage Domains**
  - CreateDomain
  - DeleteDomain
  - ListDomains
  - DomainMetaData
- **Access Data**
  - Retrieving Data
    - GetAttributes – returns a single item
    - Select – returns multiple items using SQL syntax
  - Writing Data
    - PutAttributes – put single item
    - BatchPutAttributes – put multiple items
  - Removing Data
    - DeleteAttributes – delete single item
    - BatchDeleteAttributes – delete multiple items

Technology Overview : SimpleDB

- Options available on reads and writes
  - Consistent Read
    - Read the most recently committed write
    - May have lower throughput/higher latency/lower availability
  - Conditional Put/Delete
    - i.e. Optimistic Locking
    - Useful if you want to build a consistent multi-master data store – you will still require your own anti-entropy
    - We do not use this currently, so we don't know how it performs

Challenge 1

Translate RDBMS Concepts to Key-Value Store Concepts

NETFLIX

Translate RDBMS Concepts to Key-Value Store Concepts

- Relational Databases are known for relations
- First, a quick refresher on Normal forms

Normalization

NF1 : All occurrences of a record type must contain the same number of fields -- variable repeating fields and groups are not allowed

NF2 : Second normal form is violated when a non-key field is a fact about a subset of a key

Violated here

Part	Warehouse	Quantity	Warehouse-Address

Fixed here

Part	Warehouse	Quantity

Warehouse	Warehouse-Address

NETFLIX

Normalization

- **Issues**
  - Wastes Storage
    - The warehouse address is repeated for every Part-WH pair
  - Update Performance Suffers
    - If the address of a warehouse changes, I must update every part in that warehouse – i.e. many rows
  - Data Inconsistencies Possible
    - I can update the warehouse address for one Part-WH pair and miss Parts for the same WH (a.k.a. update anomaly)
  - Data Loss Possible
    - An empty warehouse does not have a row, so the address will be lost. (a.k.a. deletion anomaly)

Normalization

- RDBMS → KV Store migrations can't simply accept denormalization!
  - Especially many-to-many and many-to-one entity relationships
- Instead, pick your data set candidates carefully!
  - Keep relational data in RDBMS
  - Move key-look-ups to KV stores
- Luckily for Netflix, most Web Scale data is accessed by Customer, Video, or both
  - i.e. Key Lookups that do not violate 2NF or 3NF

Translate RDBMS Concepts to Key-Value Store Concepts

- Aside from relations, relational databases typically offer the following:
  - Transactions
  - Locks
  - Sequences
  - Triggers
  - Clocks
  - A structured query language (i.e. SQL)
  - Database server-side coding constructs (i.e. PL/SQL)
  - Constraints

Translate RDBMS Concepts to Key-Value Store Concepts

- Partial or no SQL support (e.g. no Joins, Group Bys, etc…)
  - **BEST PRACTICE**
    - Carry these out in the application layer for smallish data
- No relations between domains
  - **BEST PRACTICE**
    - Compose relations in the application layer
- No transactions
  - **BEST PRACTICE**
    - SimpleDB : Conditional Put/Delete (best effort) w/ fixer jobs
    - Cassandra : Batch Mutate + the same column TS for all writes

NETFLIX

Translate RDBMS Concepts to Key-Value Store Concepts

- No schema - This is non-obvious. A query for a misspelled attribute name will not fail with an error
  - **BEST PRACTICE**
    - Implement a schema validator in a common data access layer

- No sequences
  - **BEST PRACTICE**
    - Sequences are often used as primary keys
      - In this case, use a naturally occurring unique key
      - If no naturally occurring unique key exists, use a UUID
    - Sequences are also often used for ordering
      - Use a distributed sequence generator or rely on client timestamps

Translate RDBMS Concepts to Key-Value Store Concepts

- No clock operations, PL/SQL, Triggers
  - BEST PRACTICE
    - **Clocks** : Instead rely on client-generated clocks and run NTP. If using clocks to determine order, be aware that this is problematic over long distances.
    - **PL/SQL, Triggers** : Do without
- No constraints. Specifically,
  - No uniqueness constraints
  - No foreign key or referential constraints
  - No integrity constraints
  - BEST PRACTICE
    - Applications must implement this functionality

Challenge 2

Work-around Issues specific to the chosen KV store

SimpleDB

NETFLIX

Work-around Issues specific to the chosen KV store

- Missing / Strange Functionality
  - No back-up and recovery
  - No native support for types (e.g. Number, Float, Date, etc…)
  - You cannot update one attribute and null out another one for an item in a single API call
  - Mis-cased or misspelled attribute names in operations fail silently. Why is SimpleDB case-sensitive?
  - Neglecting "limit N" returns a subset of information. Why does the absence of an optional parameter not return all of the data?
  - Users need to deal with data set partitioning
  - Beware of Nulls
  - Write throughput not as high as we need for certain use-cases

Work-around Issues specific to the chosen KV store

No Native Types – Sorting, Inequalities Conditions, etc…

- Since sorting is lexicographical, if you plan on sorting by certain attributes, then
  - zero-pad logically-numeric attributes
    - e.g. –
      - 0000000000000000111111 ← this is bigger
      - 0000000000000000011111
  - use Joda time to store logical dates
    - e.g. –
      - 2010-02-10T01:15:32.864Z ← this is more recent
      - 2010-02-10T01:14:42.864Z

Work-around Issues specific to the chosen KV store

- Anti-pattern : Avoid the anti-pattern Select SOME_FIELD_1 from MY_DOMAIN where SOME_FIELD_2 is null as this is a full domain scan
  - Nulls are not indexed in a sparse-table
  - **BEST PRACTICE**
    - Instead, replace this check with a (indexed) flag column called IS_FIELD_2_NULL: Select SOME_FIELD_1 from MY_DOMAIN where **IS_FIELD_2_NULL = 'Y'**
- Anti-pattern : When selecting data from a domain and sorting by an attribute, items missing that attribute will not be returned
  - In Oracle, rows with null columns are still returned
  - **BEST PRACTICE**
    - Use a flag column as shown previously

Work-around Issues specific to the chosen KV store

- **BEST PRACTICE** : Aim for high index selectivity when you formulate your select expressions for best performance
  - SimpleDB select performance is sensitive to index selectivity
  - Index Selectivity
    - Definition : # of distinct attribute values in specified attribute **/** # of items in domain
      - e.g. Good Index Selectivity (i.e. 1 is the best)
        - A table having 100 records and one of its indexed column has 88 distinct values, then the selectivity of this index is 88 / 100= **0.88**
      - e.g. Bad Index Selectivity
        - lf an index on a table of 1000 records had only 5 distinct values, then the index's selectivity is 5 / 1000 = **0.005**

Work-around Issues specific to the chosen KV store

Sharding Domains

- There are 2 reasons to shard domains
  - You are trying to avoid running into one of the sizing limits
    - e.g. 10GB of space or 1 Billion Attributes
  - You are trying to scale your writes
    - To scale your writes further, use BatchPutAttributes and BatchDeleteAttributes where possible

Challenge 3

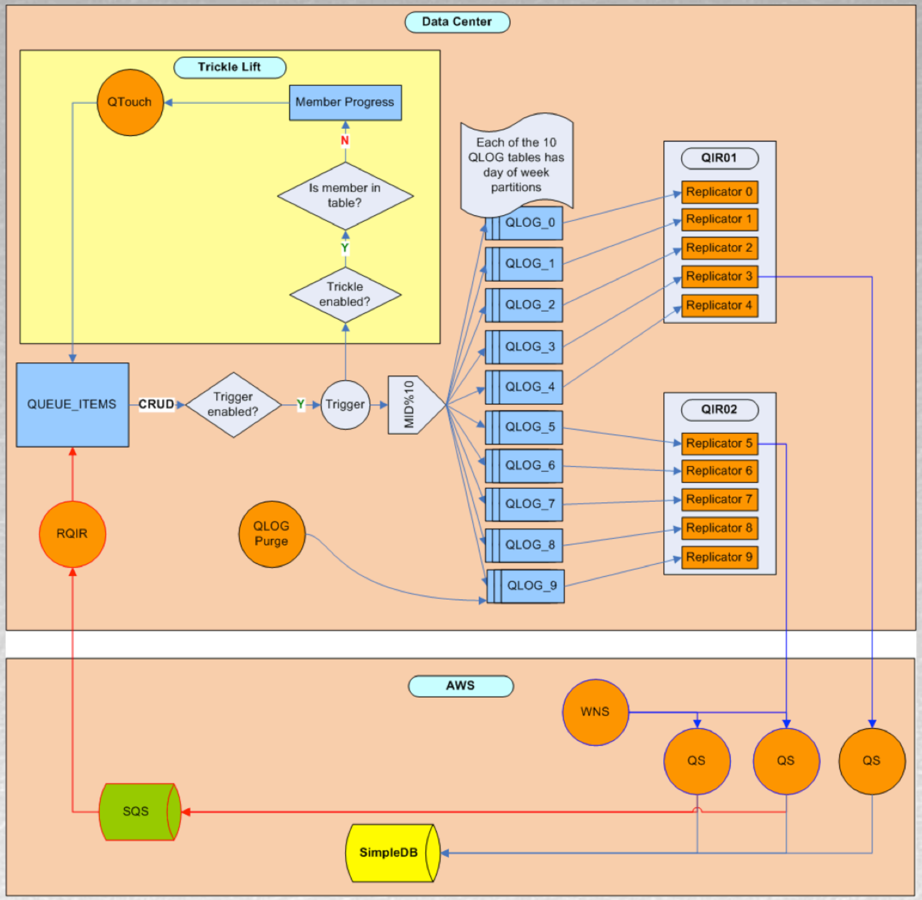
Create a Bi-directional DC-Cloud Data Replication Pipeline

NETFLIX

Create a Bi-directional DC-Cloud Data Replication Pipeline

- Home-grown Data Replication Framework known as IR for Item Replication
- 2 schemes in use currently
  - Polls the main table (a.k.a. Simple IR)
    - Doesn't capture deletes but easy to implement
  - Polls a journal table that is populated via a trigger on the main table (a.k.a. Trigger-journaled IR)
    - Captures every CRUD, but requires the development of triggers

Create a Bi-directional DC-Cloud Data Replication Pipeline



Create a Bi-directional DC-Cloud Data Replication Pipeline

- How often do we poll Oracle?
  - Every 5 seconds

- What does the poll query look like?
  - select *
    from QLOG_0
    where LAST_UPDATE_TS > :CHECKPOINT ← Get recent
    and LAST_UPDATE_TS < :NOW_MINUS_30s ← Exclude most recent
    order by LAST_UPDATE_TS ← Process in order

Create a Bi-directional DC-Cloud Data Replication Pipeline

- **Data Replication Challenges & Best Practices**
  - SimpleDB throttles traffic aggressively via 503 HTTP Response codes ("Service Unavailable")
  - With Singleton writes, I see 70-120 write TPS/domain
  - IR
    - Shard domains (i.e. partition data sets) to work-around these limits
    - Employs Slow ramp up
    - Uses BatchPutAttributes instead of (Singleton) PutAttributes call
    - Exercises an exponential bounded-back-off algorithm
    - Uses attribute-level replace=false when fork-lifting data

Netflix's Transition to NoSQL

Cassandra

Data Model

Cassandra

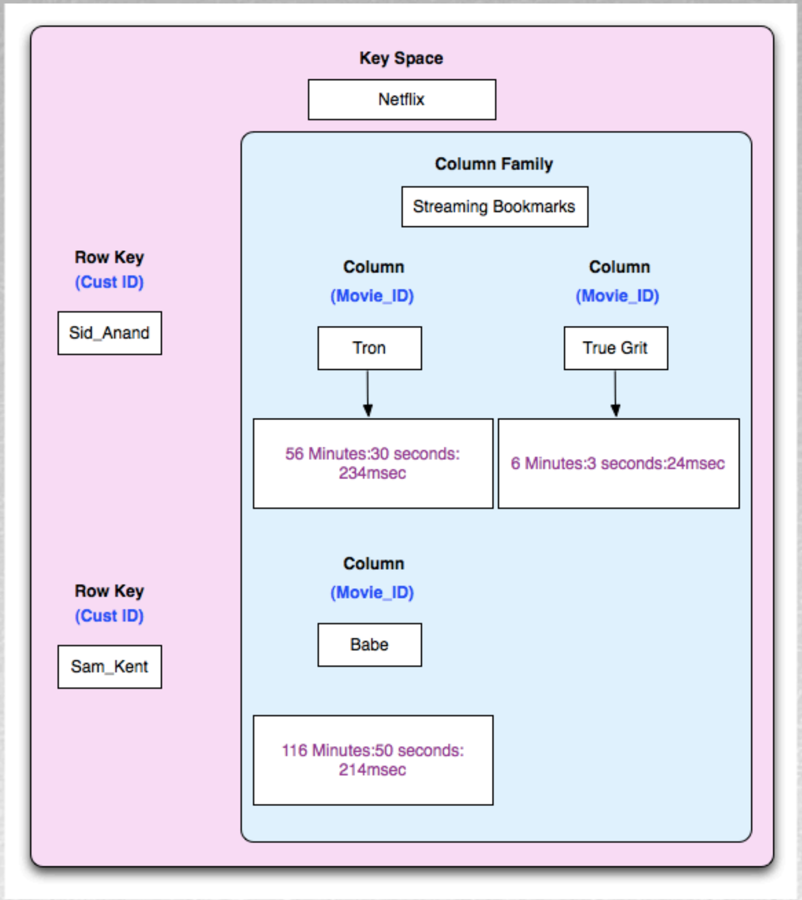
NETFLIX

Data Model : Cassandra

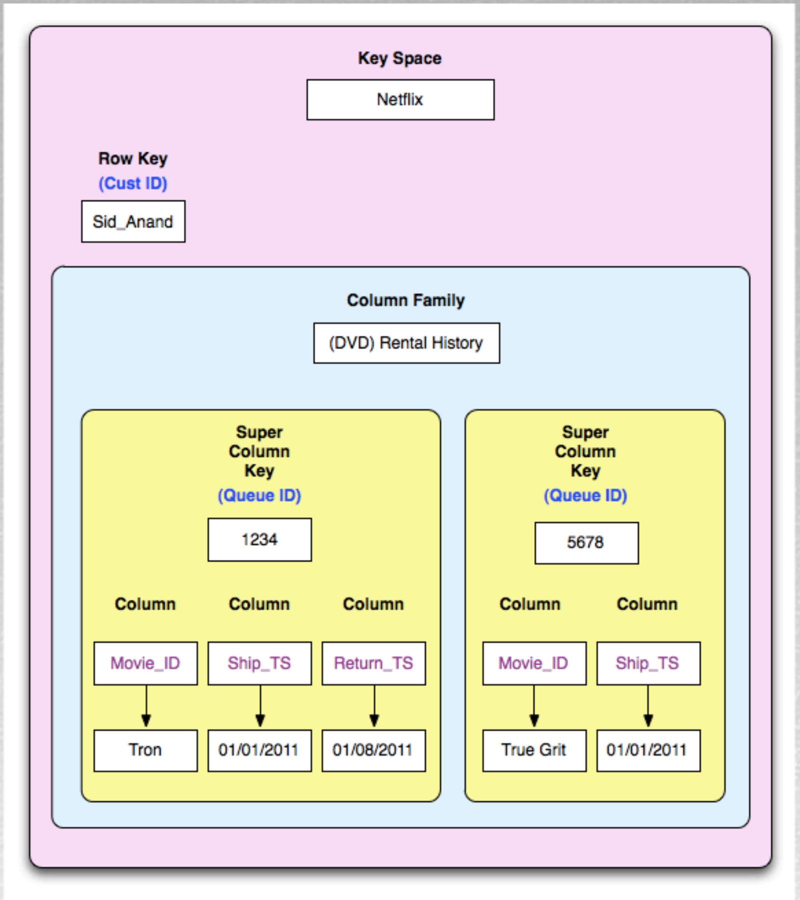
Terminology

SimpleDB	Cassandra	Relational Databases
	Key Space	“Schema”
Domain	Column Family	Table
Item	Row	Row
Item Name	Row Key	Mandatory Primary Key
	Super Columns	
Attribute	Column	Column

Data Model : Cassandra



Data Model : Cassandra



NETFLIX

API in Action

Cassandra

NETFLIX

APIs for Reads

- **Reads**
  - I want to continue watching Tron from where I left off (quorum reads)?
    - datastore.get(“Netflix”, ”Sid_Anand”, Streaming Bookmarks → Tron , ConsistencyLevel.QUORUM)
  - When did the True Grit DVD get shipped and returned (fastest read)?
    - datastore.get_slice(“Netflix”, ”Sid_Anand”, (DVD) Rental History → 5678, [“Ship_TS”, “Return_TS”], ConsistencyLevel.ONE)
  - How many DVD have been shipped to me (fastest read)?
    - datastore.get_count(“Netflix”, ”Sid_Anand”, (DVD) Rental History, ConsistencyLevel.ONE)

NETFLIX

APIs for Writes

- **Writes**
  - Replicate Netflix Hub Operation Shipments as Batched Writes : True Grit and Tron shipped together to Sid
    - datastore.batch_mutate
      (“Netflix”, **mutation_map**, ConsistencyLevel.QUORUM)

Performance Model

Cassandra

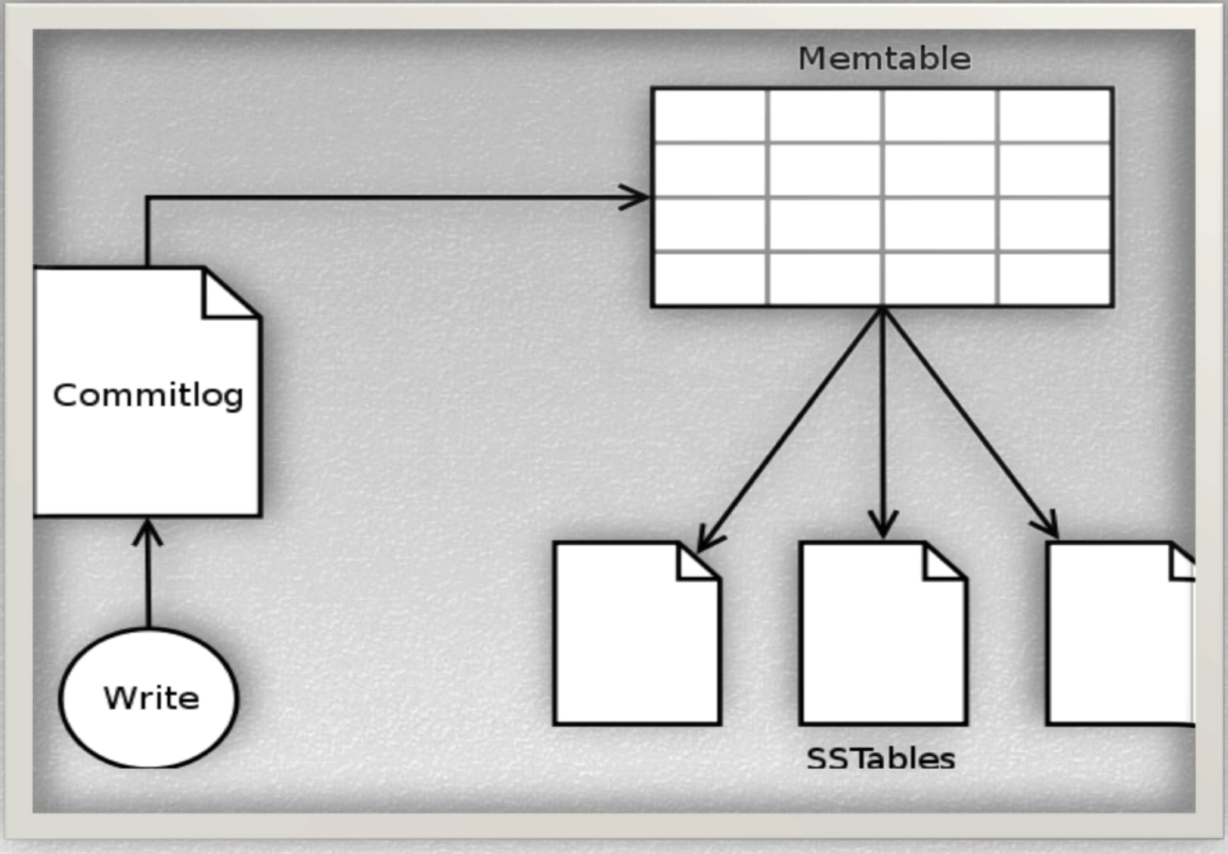
NETFLIX

The Promise of Performance

- High-Availability Writes
  - Write to Commit Log (a.k.a. Write-Ahead Log) & ACK
    - FSYNC the commit log semi-in-frequently
  - Memtable is a Hash Table in RAM → O(1) for reads and writes
  - Memtable is flushed to SSTable on disk in a background thread
    - SSTable is a sorted list on a serial device (a.k.a. disk)
  - Compensate for potential slowness at a subset of replicas by shooting a write-request to all replicas and waiting for the first success response to come back
    - Requires a lower consistency level on writes (e.g. CL=1)
    - First write to come back allows coordinator to ACK

NETFLIX

The Promise of Performance

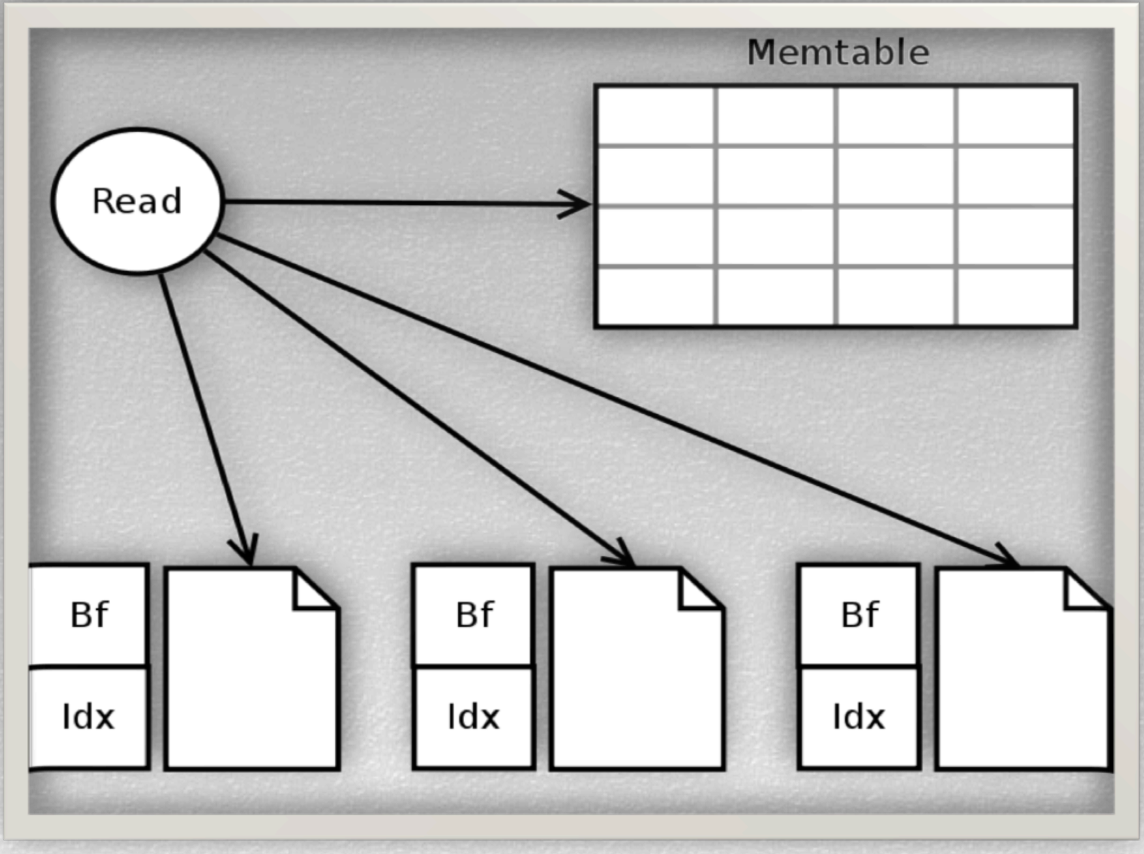


NETFLIX

The Promise of Performance

- Manage Reads
  - Rows and Top-Level Columns are stored and indexed in sorted order giving logarithmic time complexity for look up
    - These help
      - Bloom Filters at the Row Level
      - Key Cache
      - Large OS Page Cache
    - These do not help
      - Disk seeks on reads
        - It gets worse with more row-redundancy across SSTables → Compaction is a necessary evil
      - Compaction wipes out the Key Cache

The Promise of Performance



Distribution Model

Cassandra

NETFLIX

Distribution Model

- No time here.. Read up on the following:
  - Merkle Trees + Gossip ➔ Anti-Entropy
  - Read-Repair
  - Consistent Hashing
  - SEDA (a.k.a. Staged Event Driven Architecture) paper
  - Dynamo paper

Features We Like

Cassandra

Features We Like

- Rich Value Model : Value is a set of columns or super-columns
  - Efficiently address, change, and version individual columns
  - Does not require read-whole-row-before-alter semantics
- Effectively No Column or Column Family Limits
  - SimpleDB Limits
    - 256 Attributes / Item
    - 1 Billion Attributes / Domain
    - 1 KB Attribute Value Length
- Growing a Cluster a.k.a. Resharding a KeySpace is Manageable
  - SimpleDB
    - Users must solve this problem : application code needs to do the migration

Features We Like

- Better handing of Types
  - SimpleDB
    - Everything is a UTF-8 string
  - Cassandra
    - Native Support for Key and Column Key types (for sorting)
    - Column values are never looked at and are just []byte
- Open Source & Java
  - Implement our own Backup & Recovery
  - Implement our own Replication Strategy
  - We know Java best, though we think Erlang is cool, with the exception of the fact that each digit in an integer is 1 byte of memory!!!
  - We can make it work in AWS

Features We Like

- No Update-Delete Anomalies
  - Specify a batch mutation with a delete and a mutation for a single row_key/column-family pair in a single batch
  - Must use same column Time Stamp
- Tunable Consistency/Availability Tradeoff
  - Strong Consistency
    - Quorum Reads & Writes
  - Eventual Consistency
    - R=1, W=1 (fastest read and fastest write)
    - R=1 , W=QUORUM (fastest read and potentially-slower write)
    - R=QUORUM, W=1 (potentially-slower read and fastest write)

Where Does That Leave Us?

Cassandra

NETFLIX

Where Are We With This List?

- KV Store Missing / Strange Functionality
  - ~~No back up and recovery~~
  - ~~No native support for types (e.g. Number, Float, Date, etc…)~~
  - ~~You cannot update one attribute and null out another one for an item in a single API call~~
  - Mis-cased or misspelled attribute names in operations fail silently.
  - ~~Neglecting "limit N" returns a subset of information. Why does the absence of an optional parameter not return all of the data?~~
  - ~~Users need to deal with data set partitioning~~
  - ~~Beware of Nulls~~
  - ~~Write throughput not as high as we need for certain use-cases~~

Pick a Data Store in the Cloud

An ideal storage solution should have the following features:

- ☐ Hosted
- ☑ Managed Distribution Model
- ☑ Works in AWS
- ☑ AP from CAP
- ☑ Handles a majority of use-cases accessing high-growth, high-traffic data
  - ☑ Specifically, key access by customer id, movie id, or both
- ☑ Back-up & Recovery
- ☑ Multi-Region

Questions?

NETFLIX