Netflix's Cloud Data Architecture

Siddharth "Sid" Anand

@r39132

QCon London
March 2011

NETFLIX

2 Talks @ QCon London

- Whitepaper : Netflix's Transition to High-Availability Storage Systems
- Title : Netflix's Cloud Data Architecture
  - Track : Architectures You've Always Wondered About
    - General Overview
    - Data Replication Deep Dive
- Title : NoSQL @ Netflix
  - Track : NoSQL : Where and How
    - SimpleDB, S3, and Cassandra

What is Netflix?

NETFLIX

What is Netflix?

Circa 1997

- **Rent a Movie**
  - Right of first sale

- **Buy a Movie**
  - Any retailer (e.g. Walmart) or e-tailer (e.g. Amazon)

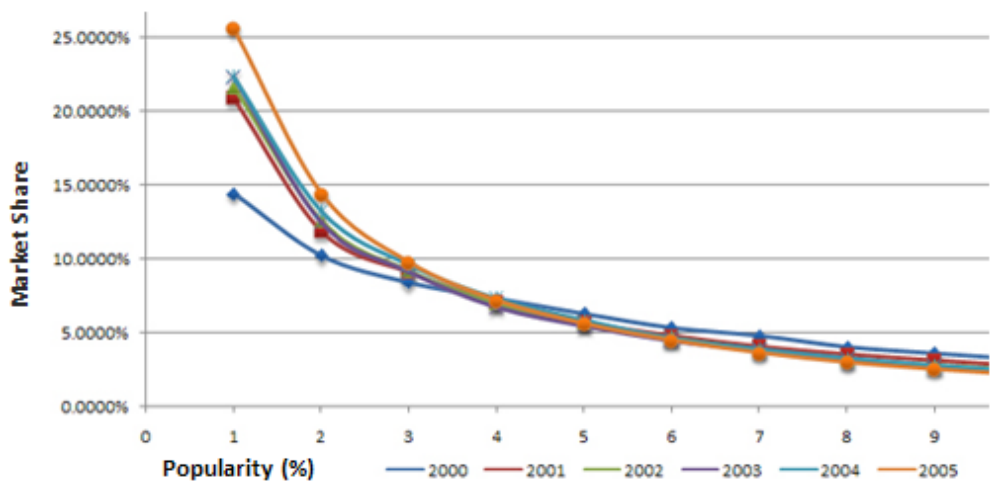
What is Netflix?

- A brick-and-mortar store can only hold ~1k-2k titles
  - Which DVD titles do they pick?
    - Brand New Releases
    - Long-standing Hits (e.g. The Godfather)
  - Brand New Releases are expensive
  - Stores profit by re-renting the same video within a short time frame (i.e. DVD Availability Window)
    - Must impose steep rental fees and steeper late fees

What is Netflix?

Circa 1997

- Netflix saw opportunity in the long-tail business
  - We store >120K titles in our 50+ shipping hubs
  - We recommend movies in the long tail, personalized to the customer, lowering costs



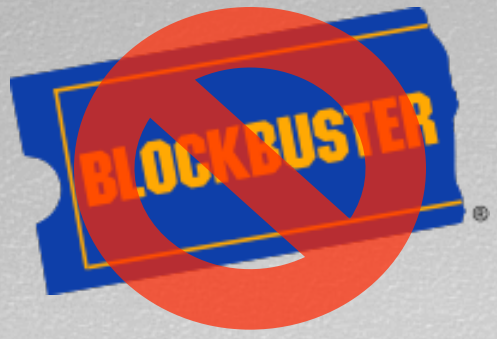
What is Netflix?

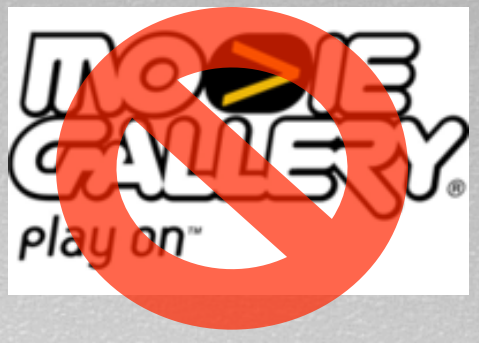
In 1999

- Netflix launches DVD rentals-by-mail
  - Unlimited rentals for a flat monthly fee
  - No due dates
  - No late fees
  - No shipping or handling fees

What is Netflix?

After a few years…





NETFLIX

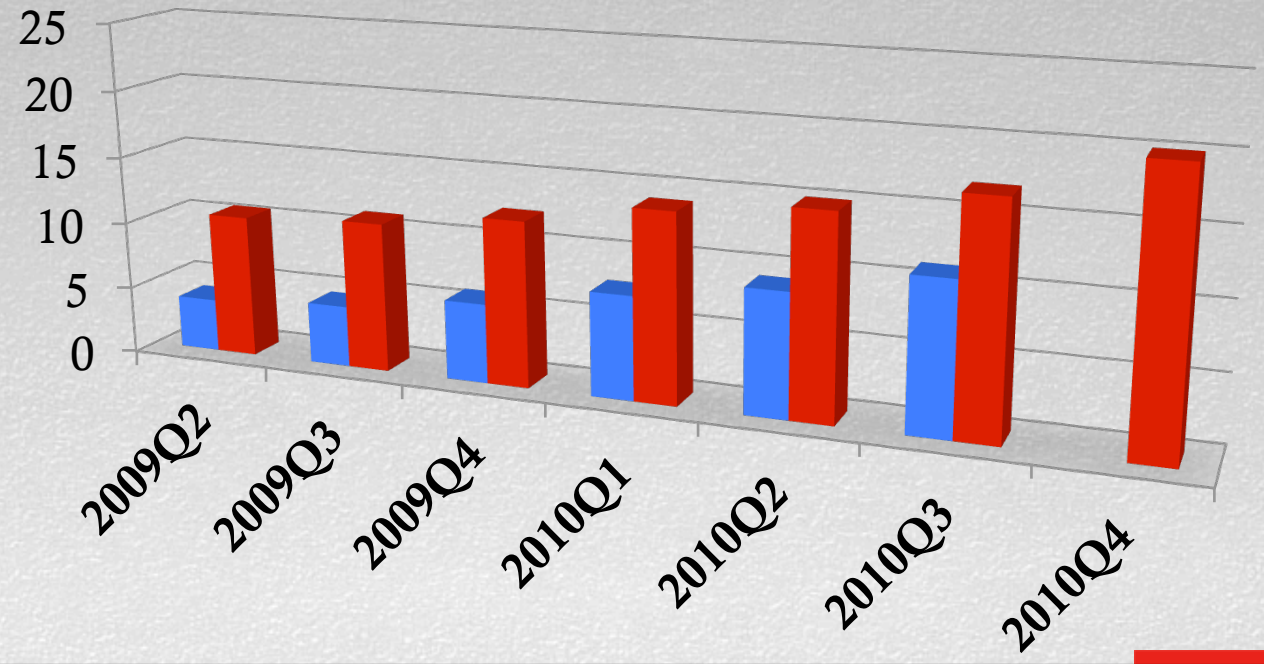
What is Netflix?

Fast Forward to 2008

- Netflix forecasts the gradual end of the DVD and starts switching to Movie Streaming
- Upside?
  - We spend $500MM to $600MM annually on US Postage for DVD mailing
  - Streaming a movie is 1/100th the cost of shipping a DVD
  - Easier to craft a business model for international expansion

What is Netflix?

- We have 20M+ subscribers in the US and Canada
- Uses 20% of US peak downstream bandwidth
- 1st or 2nd largest CDN user in the US



Source: http://ir.netflix.com

NETFLIX

What is Netflix?

Grew subscribers 2008-2010 by partnering with device makers

- Game Consoles
  - Xbox 360, Wii, PS3
- Mobile Devices
  - Apple IPad, IPhone, IPod Touch, Windows 7

Our Cloud Story

Migration to AWS

NETFLIX

Our Cloud Story

- 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 our capacity planning and scale out
    - Allows us to focus on core competencies

Our Cloud Story

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

- **Footnote** : A short 2 years later, we serve >90% of our traffic out of AWS
  - excluding the video streaming bits, which come from CDNs

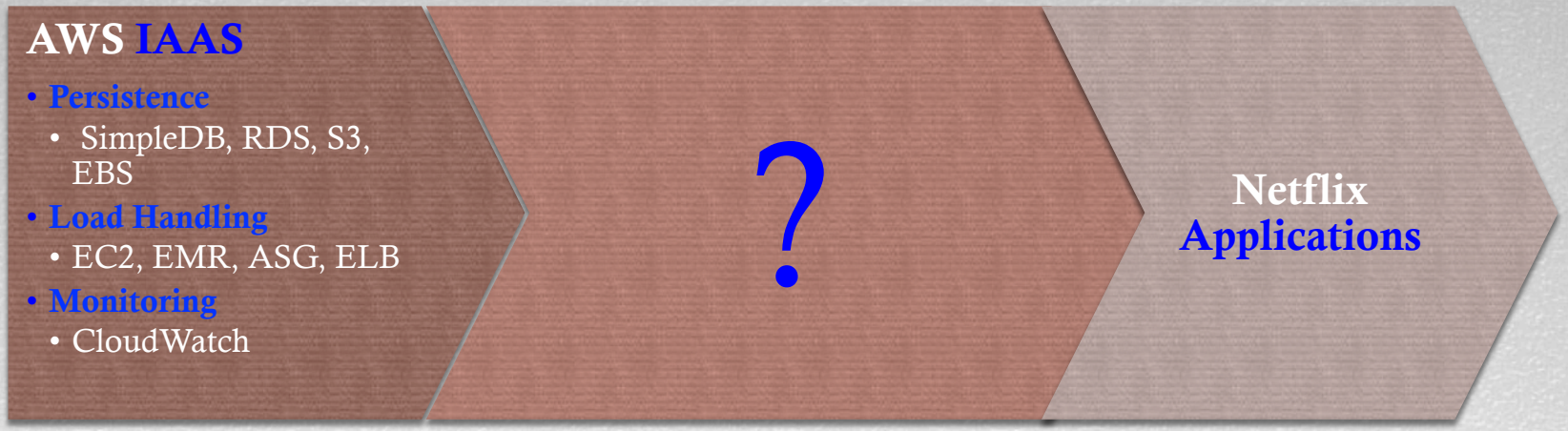
How Netflix Uses AWS

The Netflix Application Platform

NETFLIX

How Netflix Uses AWS

AWS provides various IAAS offerings, but applications need more!

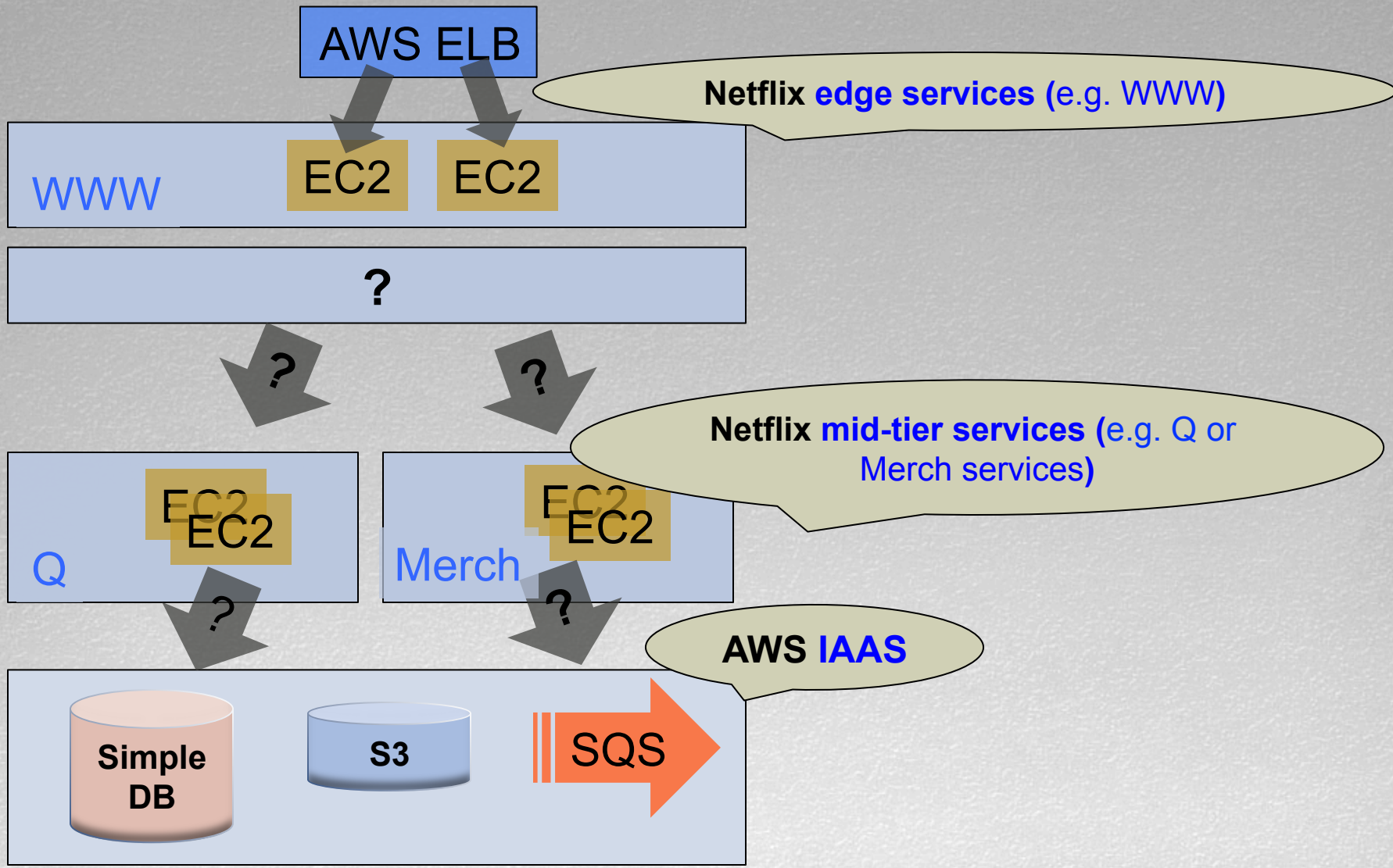


How Netflix Uses AWS

AWS provides various IAAS offerings, but applications need more!

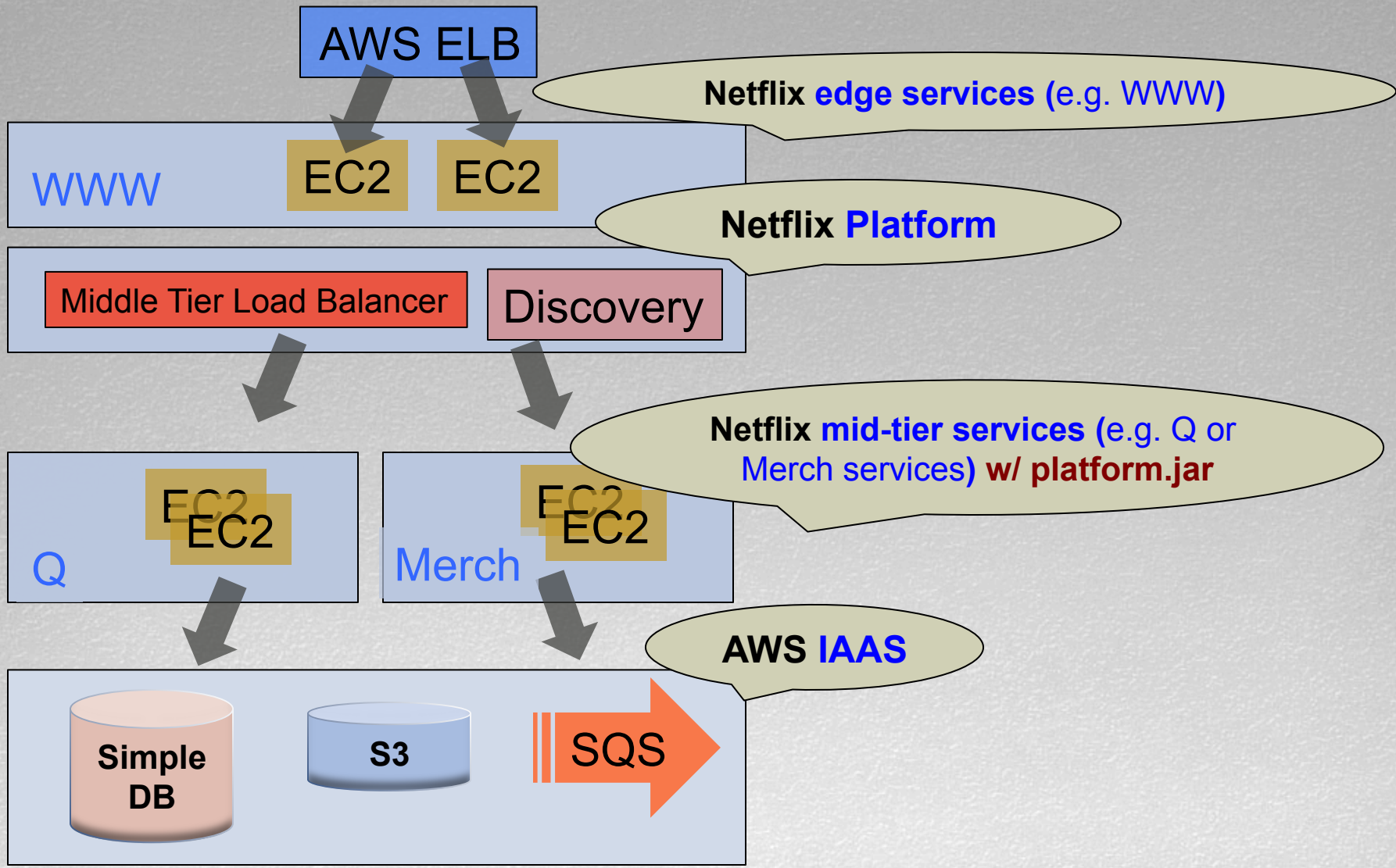
Hence the need for Netflix's infrastructure team to bridge the gap!

How Netflix Uses AWS



NETFLIX

Our Cloud Story



NETFLIX

How Netflix Uses AWS (EC2 Stack)

Application

Platform & Other Infrastructure jars

Discovery Client

MT Load Balancer

Crypto Client

S3 Client

SimpleDB Client

SQS Client

EVCache Client

Configuration Client

Guest OS / VM / Host OS / Machine

How Netflix Uses AWS

The Netflix Data Platform

NETFLIX

How Netflix Uses AWS

- Persistence in the Cloud (c.f. NoSQL @ Netflix talk)
  - SimpleDB
  - S3
  - Cassandra
- Data Replication
  - IR (a.k.a. Item Replicator)
  - HAProxy + Squid + Oracle → temporary

Persistence

SimpleDB

NETFLIX

Persistence : 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

Persistence : 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)

Persistence : SimpleDB

- Moved some of our largest transactional data sets to SimpleDB in 2010
  - e.g. **RENTAL HISTORY** : Everything that anyone has watched at Netflix, including streaming and DVD
- We have
  - ~ thousands of domains
  - ~ 1TB of OLTP/transactional data (i.e. no CLOBS or BLOBS)
  - ~ billions of rows of data (a.k.a. items)
- We execute billions of SimpleDB statements per day

Persistence

S3

NETFLIX

Persistence : S3

- Simple KV store organized as objects in buckets
  - Each AWS account is given a max of 10 buckets
  - Each bucket can hold an unlimited number of objects
- We use S3 to store data that does not fit into SimpleDB
  - Logs from streaming devices
  - Files used in movie encoding
  - Truncated-tail of Rental History
    - Good pattern introduced by Greg Kim

Data Replication

Item Replicator

Data Replication between Oracle, SimpleDB, S3, and Cassandra

NETFLIX

Data Replication : IR

- Home-grown Data Replication Framework known as IR for Item Replication
- Keeps data in sync between RDBMS (DC) and NoSQL (Cloud)
  - Mostly unidirectional: DC → Cloud
- 2 data capture schemes in use currently
  - Trigger-oriented IR
    - All CRUD operations on source table X are copied via a trigger to XLog_i journal tables (i is the shard index)
    - IR reads (polls) from a journal table
  - Simple IR
    - IR reads (polls) from source table X directly

Data Capture Schemes

IR

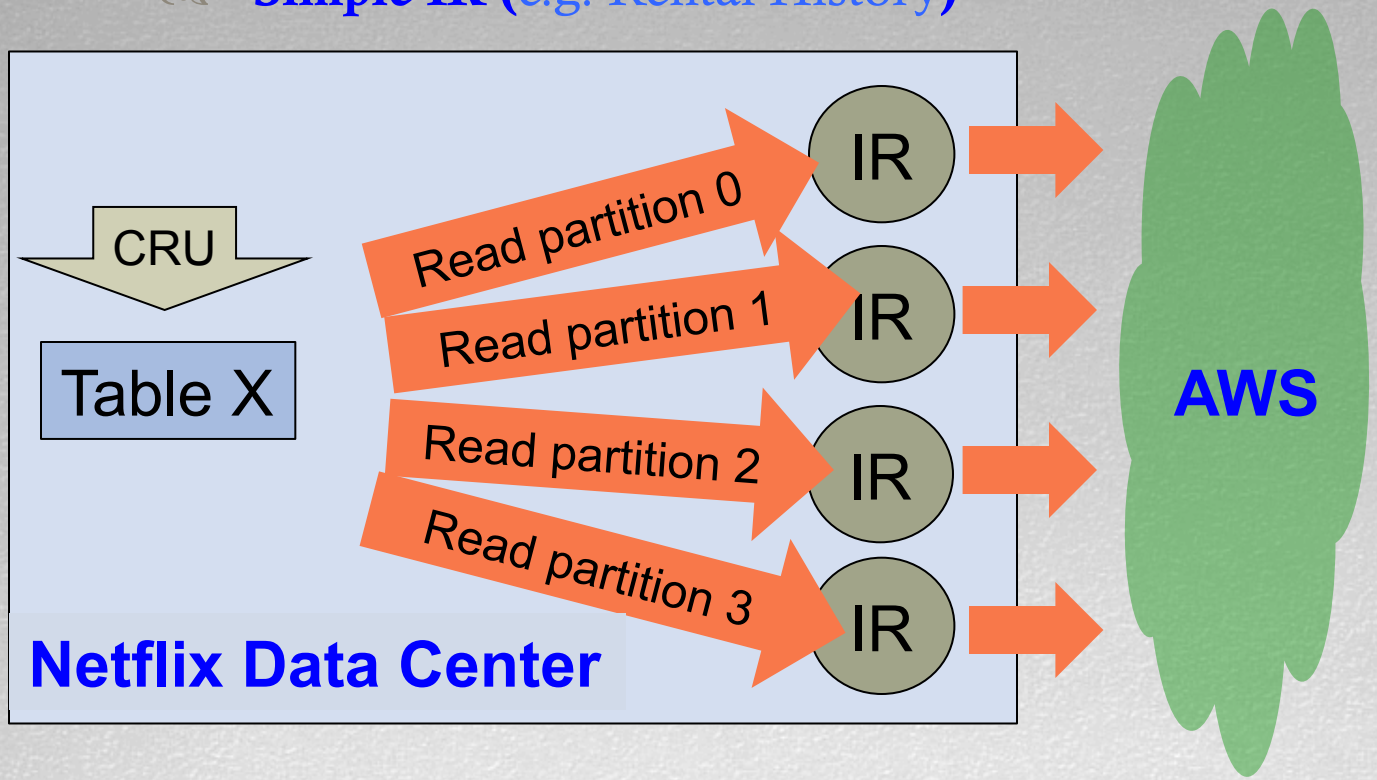
NETFLIX

Data Capture Schemes : IR

- **Trigger-oriented IR (**e.g. Movie Q**)**

Data Capture Schemes : IR

- **Simple IR (**e.g. Rental History**)**



IR's Polling Select

NETFLIX

IR's Polling Select

- **Anatomy of an IR Select Query**
  - **Execute the SQL below as a recurring poll**

```
select * from RENTAL_HISTORY r
where r.LAST_MODIFIED_TS > checkpoint value        ← (A)
and r.LAST_MODIFIED_TS < (now-10 seconds)          ← (B)
order by r.LAST_MODIFIED_TS asc                    ← (C)
```

  - **(A)** – preserves progress in case IR crashes and restarts
  - **(B)** – handles interesting race condition… will explain in later slide
  - **(C)** – in order read for repeatability as in **(A)**

IR's Polling Select

- **Anatomy of an IR Select Query**
  - Assume previous checkpoint is T0
  - @ time = T4, only the top 2 records are visible and will be replicated
    - (B) hides The Town and Duma
    - {1, The Machinist} and then {2, Blood Diamond} are replicated to the cloud
  - After IR replicates this data, the checkpoint is now T2

Customer_ID	Movie_ID	last_modified_ts	
1	The Machinist	T1	←
2	Blood Diamond	T2	←
4	The Town	T4 - 3sec	
1	Duma	T4	

IR's Polling Select

- **Anatomy of an IR Select Query**
  - Assume previous checkpoint is T2
  - @ time = T4+12 sec, The 3 records pointed to by arrows are visible and replicated to the cloud
    - Of these 1 new record, The King's Speech, became visible in time to be replicated – Commit-delayed

	Customer_ID	Movie_ID	last_modified_ts	
	1	The Machinist	T1	
	2	Blood Diamond	T2	
	4	The Town	T4 – 3 sec	←
new	3	The King's Speech	T4	←
	1	Duma	T4	←
new	5	Rescue Dawn	T4 + 11 sec	

IR's Polling Select

- "Commit-delayed" Race Condition
- Consider 2 update transactions
  - Transaction 1

```
update RENTAL_HISTORY r
set r.LAST_MODIFIED_TS = systimestamp,
r.MOVIE_ID = 'The Machinest good'
where r.CUSTOMER_ID = 1;
commit;
```

  - Transaction 2

```
update RENTAL_HISTORY r
set r.LAST_MODIFIED_TS = systimestamp,
r.MOVIE_ID = 'Blood Diamond good'
where r.CUSTOMER_ID = 2;
commit;
```

IR's Polling Select

- One possible schedule
  - Transaction 1

    update RENTAL_HISTORY r

    set r.LAST_MODIFIED_TS = systimestamp **@T1**

    r.MOVIE_ID = 'The Machinest good'

    where r.CUSTOMER_ID = 1;

    commit; **@T4**
  - Transaction 2

    update RENTAL_HISTORY r

    set r.LAST_MODIFIED_TS = systimestamp, **@T2**

    r.MOVIE_ID = 'Blood Diamond good'

    where r.CUSTOMER_ID = 2;

    commit; **@T3**

IR's Polling Select

Customer_ID	Movie_ID	last_modified_ts
1	The Machinist bad	T0
2	Blood Diamond bad	T0

-- Transaction 2 commits at T3, but records T2

Customer_ID	Movie_ID	last_modified_ts
1	The Machinist bad	T0
2	Blood Diamond good	T2

-- IR replicates "Blood Diamond good" and sets checkpoint = T2

-- Transaction 1 commits at T4, but records T1

Customer_ID	Movie_ID	last_modified_ts
1	The Machinist good	T1
2	Blood Diamond good	T2

-- IR will never see "The Machinist good" because checkpoint already advanced past T1 to T2

IR's Polling Select

- Solving the "Commit-delayed" Race Condition
  - Transaction 1

```
update RENTAL_HISTORY r
set r.LAST_MODIFIED_TS = systimestamp,
r.MOVIE_ID = 'The Machinest good'
where r.CUSTOMER_ID = 1
and r.last_modified_ts < (now – 10 sec);
commit;
```

  - Transaction 2

```
update RENTAL_History r
set r.LAST_MODIFIED_TS = systimestamp,
r.MOVIE_ID = 'Blood Diamond good'
where r.CUSTOMER_ID = 2;
and r.last_modified_ts < (now – 10 sec);
commit;
```

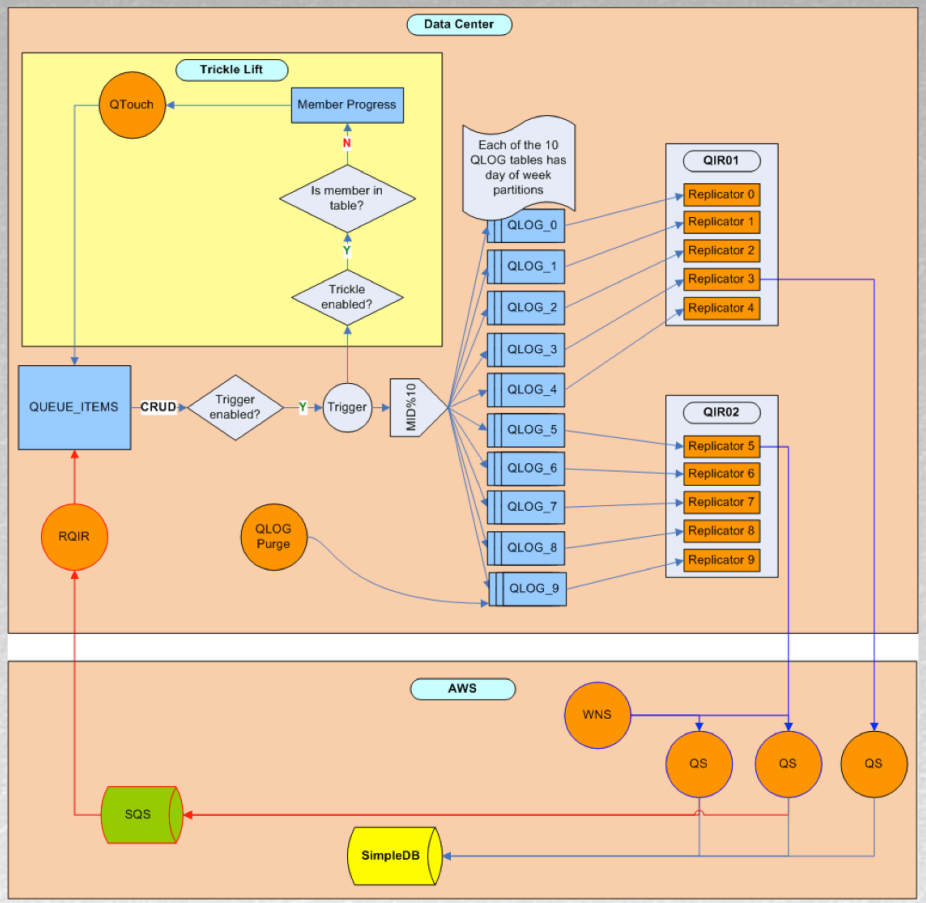
Forklifting Historical Data

NETFLIX

Forklifting Historical Data

- Fork-lifting Data – TRICKS!
  - Trigger-oriented IR
    - Trickle-lifting
      - Parallel fork-lifting and incremental replication
  - Simple IR
    - Typically need to forklift by recovering a snapshot in a second database
      - Execute forklift of that data
      - Assume database snapshot @ time = T1
    - In the primary database, buffer modifications until forklift complete
      - One forklift is complete, replicate data changed after time=T1

Forklifting Historical Data



Forklifting Historical Data

- Trickle-lifting Data for the movie Q
  - User with id=6 adds a movie to his queue
  - Trigger fires to replicate that data to QLOG_6 (e.g. 6 % 10 shards = 6)
  - Along the way, the trigger checks whether user with id=6 is found in the member_progress table
    - If not, the user id is inserted into the member_progress table
    - If so, no action is taken
  - A separate program called QTouch will notice a new record while polling the member_progress table : user id = 6
    - QTouch will execute a harmless update of all records for the user to fork-lift that user's data into the cloud

Best Practices : IR

- Data Model Best Practices
  - Checkpoint Column
    - Choose a Timestamp not Date data type (i.e. for microsecond granularity) or an ordered sequence
    - Need a Before Trigger to set the timestamp or sequence
      - Don't trust what is passed in by DB clients
    - Index the column

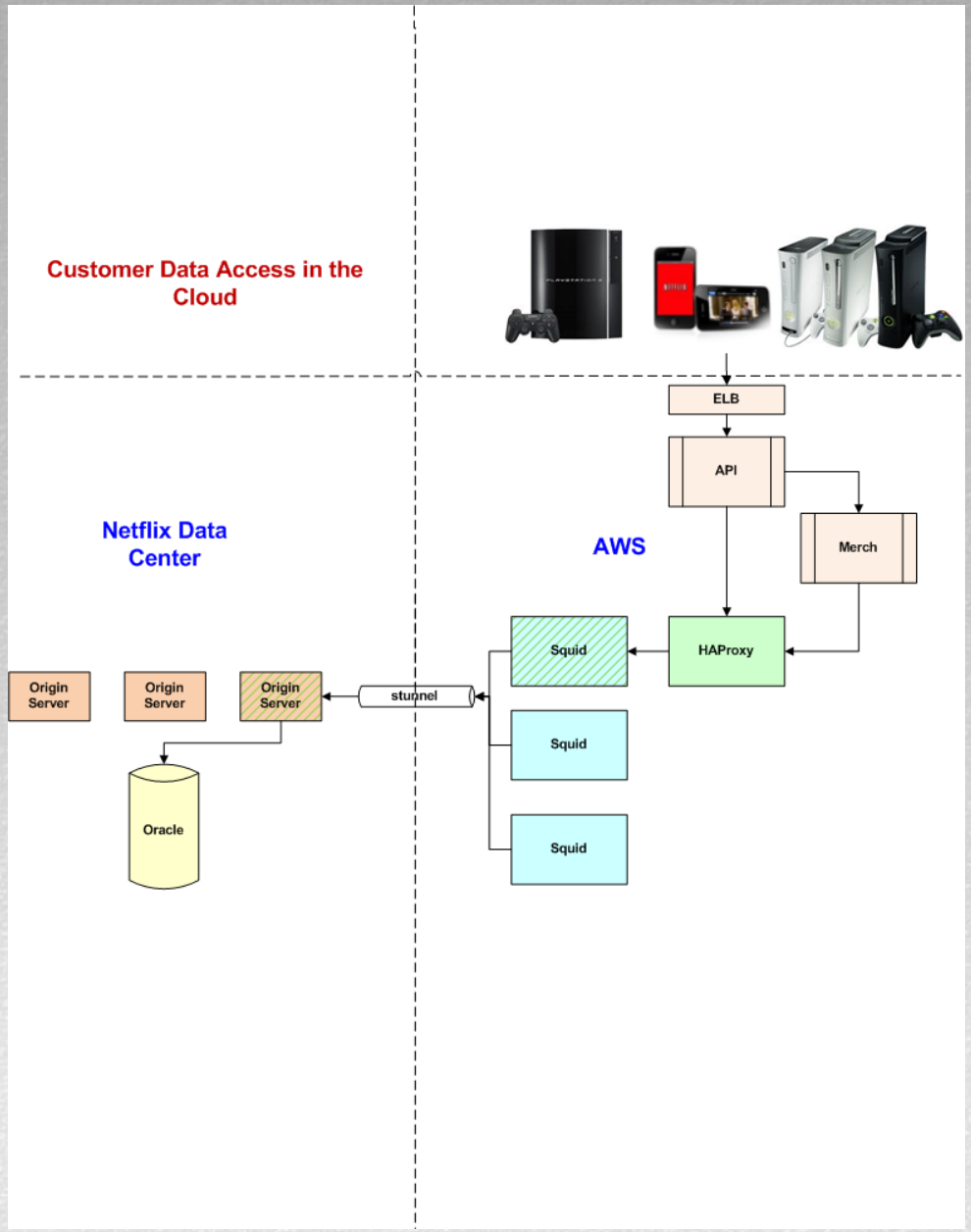
Data Replication

HAProxy + Squid + Oracle

Data lives in Oracle but is cached in the cloud

NETFLIX

Data Replication : HAProxy+Squid+Oracle



@r39132

NETFLIX

Questions?

NETFLIX