Data Infrastructure @ LinkedIn (v2)

Sid Anand

QCon NY (June 2012)

What Do I Mean by V2?

V2 == version of this talk, not version of our architecture.

Version 1 of this talk

- Presented at QCon London (March 2012)
  - http://www.infoq.com/presentations/Data-Infrastructure-LinkedIn

Version 2 – i.e. this talk

- Contains some overlapping content
- Introduces Espresso, our new NoSQL database

For more information on what LinkedIn Engineering is doing, feel free to follow @LinkedInEng

About Me

Current Life…

- LinkedIn
  - Site Ops
  - Web (Software) Engineering
    - Search, Network, and Analytics (SNA)
      - Distributed Data Systems (DDS)
        - Me (working on analytics projects)

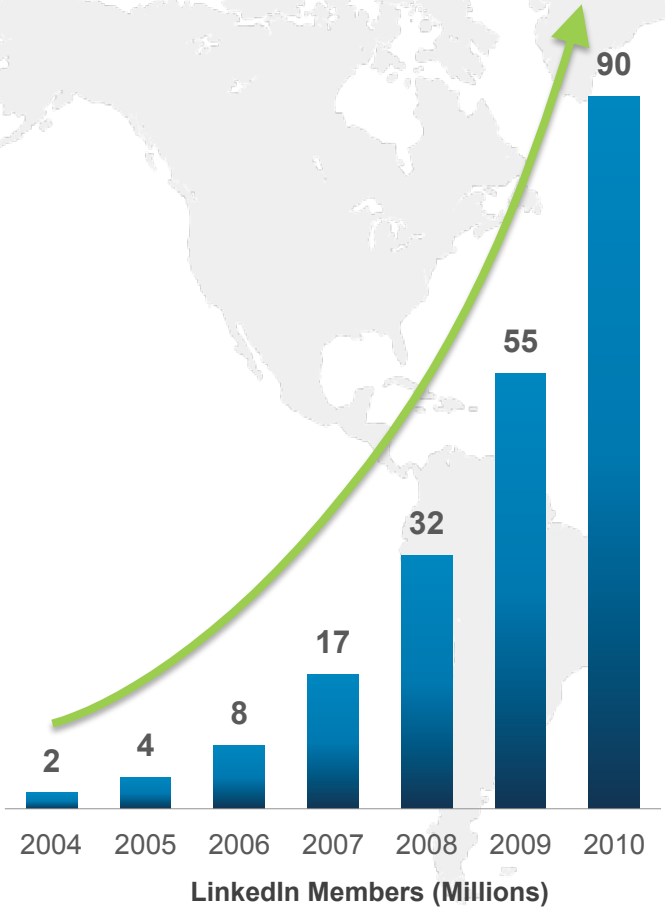
In Recent History…

- **Netflix**, Cloud Database Architect (4.5 years)
- **eBay**, Web Development, Research Labs, & Search Engine (4 years)

Let’s Talk Numbers!

The world's largest professional network

Over 60% of members are outside of the United States



82%*
Fortune 100 Companies use LinkedIn to hire

>2M*
Company Pages

17*
Languages

~4.2B
Professional searches in 2011

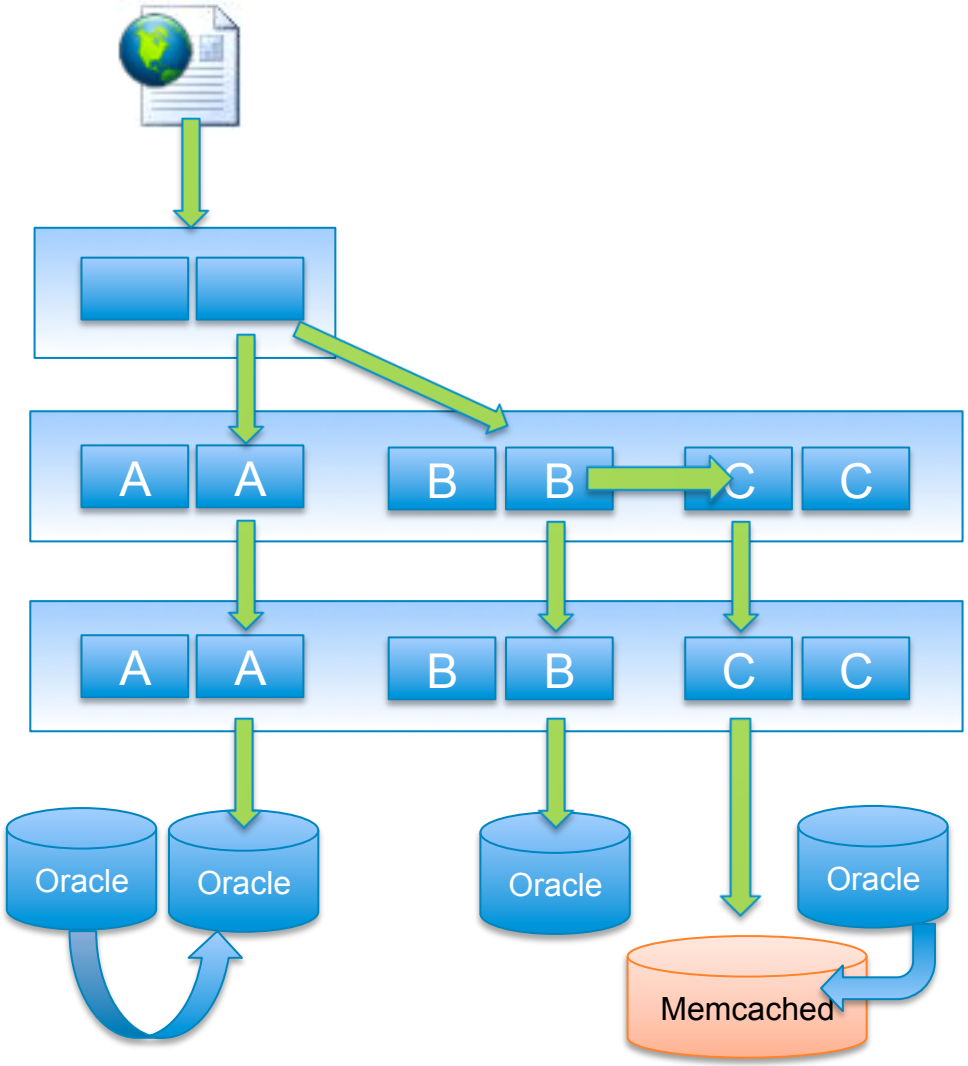
*as of March 31, 2012

Our Architecture

Overview

- Our site runs primarily on Java, with some use of Scala for specific infrastructure
- What runs on Scala?
  - Network Graph Service
  - Kafka
- Most of our services run on Apache Traffic Server + Jetty

LinkedIn : Architecture



Presentation Tier

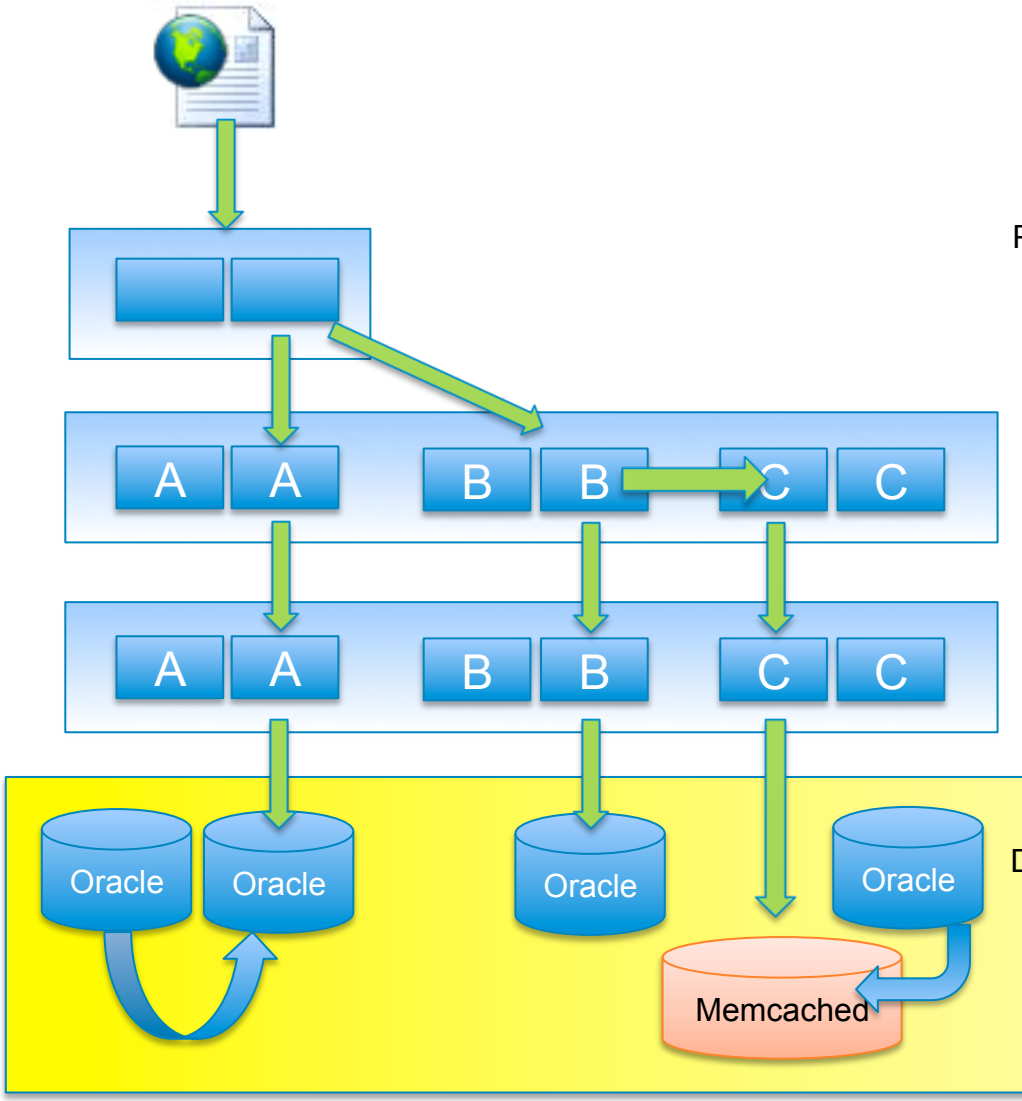
Business Tier

Data Access Tier

Data Infrastructure

- A web page requests information A and B
- A thin layer focused on building the UI. It assembles the page by making parallel requests to the BT
- Encapsulates business logic. Can call other BT clusters and its own DAT cluster.
- Encapsulates DAL logic
- Concerned with the persistent storage of and easy access to data

LinkedIn : Architecture



Presentation Tier

Business Tier

Data Access Tier

Data Infrastructure

- A web page requests information A and B
- A thin layer focused on building the UI. It assembles the page by making parallel requests to the BT
- Encapsulates business logic. Can call other BT clusters and its own DAT cluster.
- Encapsulates DAL logic
- Concerned with the persistent storage of and easy access to data

Data Infrastructure Technologies

LinkedIn Data Infrastructure Technologies

Oracle: Source of Truth for User-Provided Data

Oracle

- Until recently, all user-provided data was stored in Oracle – our source of truth
  - Espresso is ramping up
- About 50 Schemas running on tens of physical instances
- With our user base and traffic growing at an accelerating pace, how do we scale Oracle for user-provided data?

Scaling Reads

- Oracle Slaves
- Memcached
- Voldemort – for key-value lookups

***

Scaling Writes

- Move to more expensive hardware **or** replace Oracle with something better

LinkedIn Data Infrastructure Technologies

Voldemort: Highly-Available Distributed Data Store

Voldemort : Overview

- A distributed, persistent key-value store influenced by the Amazon Dynamo paper

- Key Features of Dynamo
  - Highly Scalable, Available, and Performant
  - Achieves this via Tunable Consistency
    - Strong consistency comes with a cost – i.e. lower availability and higher response times
    - The user can tune this to his/her needs
  - Provides several self-healing mechanisms when data does become inconsistent
    - Read Repair
      - Repairs value for a key when the key is looked up/read
    - Hinted Handoff
      - Buffers value for a key that wasn't successfully written, then writes it later
    - Anti-Entropy Repair
      - Scans the entire data set on a node and fixes it
  - Provides means to detect node failure and a means to recover from node failure
    - Failure Detection
    - Bootstrapping New Nodes

Voldemort : Overview

API

VectorClock<V> **get** (K key)

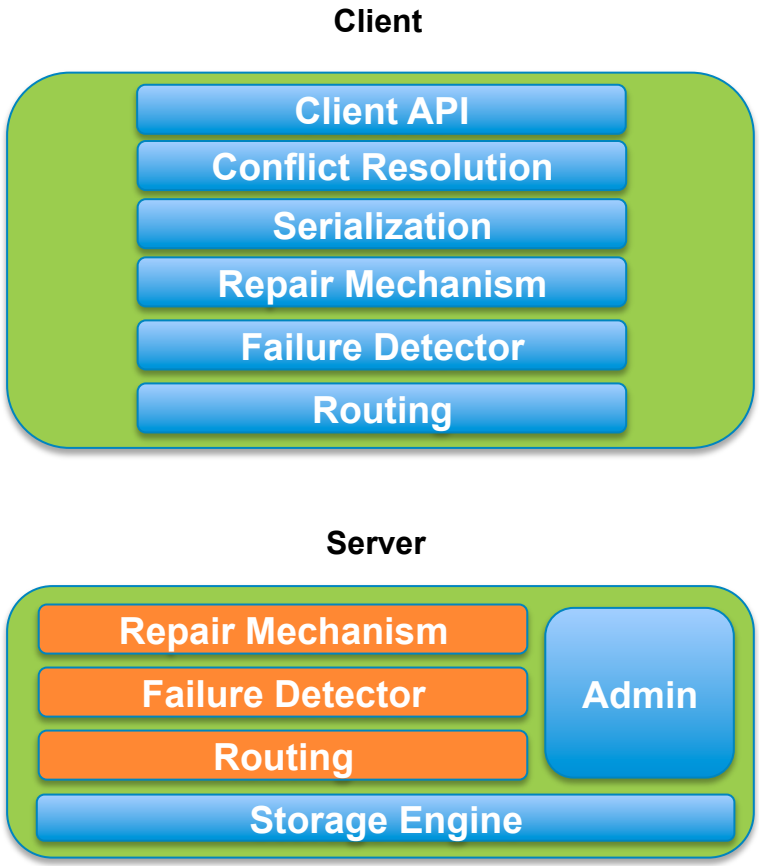
put (K key, VectorClock<V> value)

applyUpdate(UpdateAction action, int retries)

Voldemort-specific Features

- Implements a layered, pluggable architecture
- Each layer implements a common interface (c.f. API). This allows us to replace or remove implementations at any layer
  - Pluggable data storage layer
    - BDB JE, Custom RO storage, etc…
  - Pluggable routing supports
    - Single or Multi-datacenter routing

Layered, Pluggable Architecture

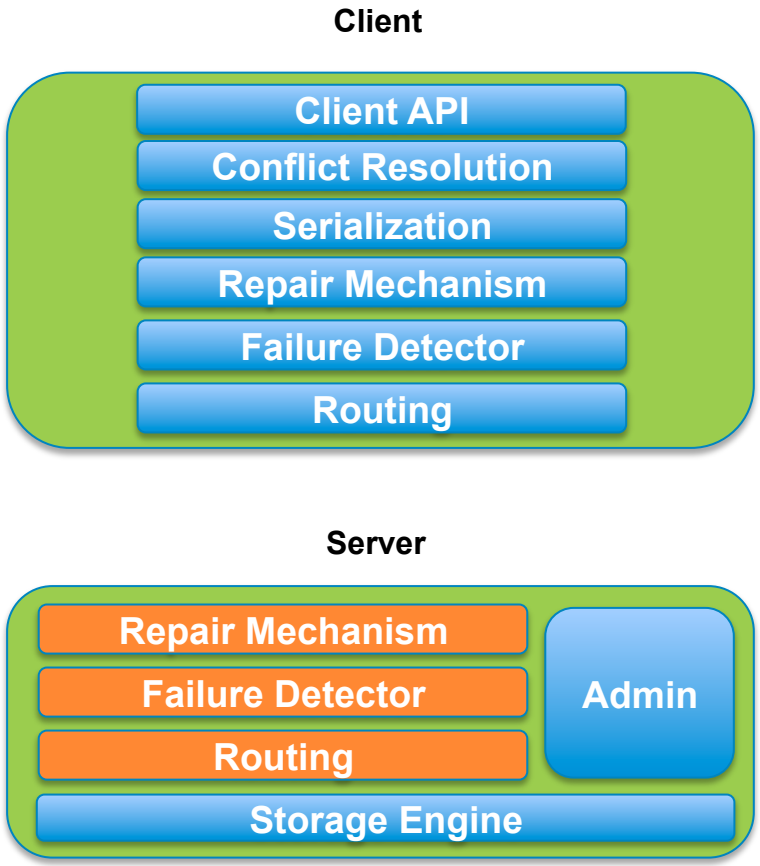


Voldemort : Overview

Voldemort-specific Features

- Supports Fat client or Fat Server
  - **Repair Mechanism** + **Failure Detector** + **Routing** can run on server or client
- LinkedIn currently runs Fat Client, but we would like to move this to a Fat Server Model

Layered, Pluggable Architecture



Where Does LinkedIn use Voldemort?

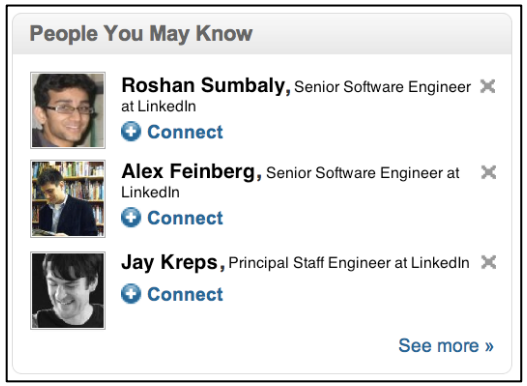
Voldemort : Usage Patterns @ LinkedIn

2 Usage-Patterns

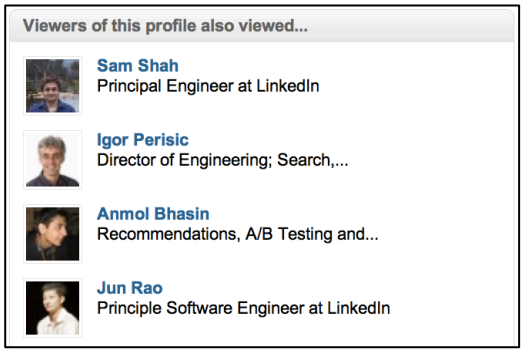
- Read-Write Store
  - A key-value alternative to Oracle
  - Uses BDB JE for the storage engine
  - 50% of Voldemort Stores (aka Tables) are RW
- Read-only Store
  - Uses a custom Read-only format
  - 50% of Voldemort Stores (aka Tables) are RO
- Let's look at the RO Store

Voldemort : RO Store Usage at LinkedIn

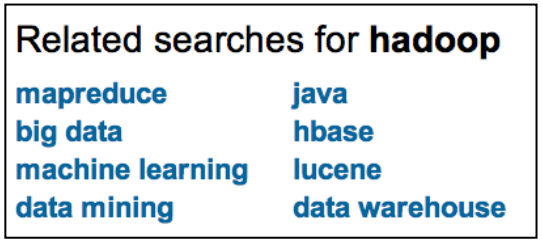
People You May Know



Viewers of this profile also viewed

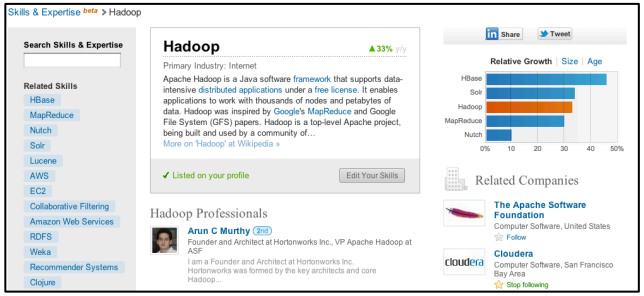


Related Searches

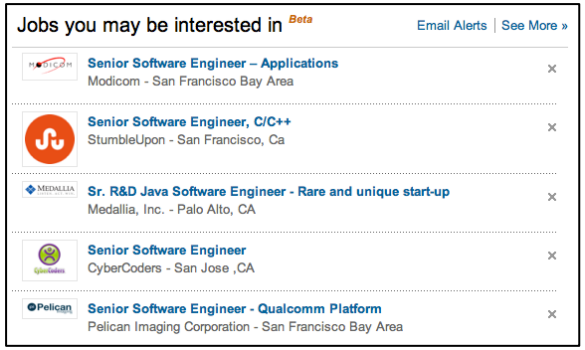


Events you may be interested in

LinkedIn Skills



Jobs you may be interested in



Voldemort : Usage Patterns @ LinkedIn

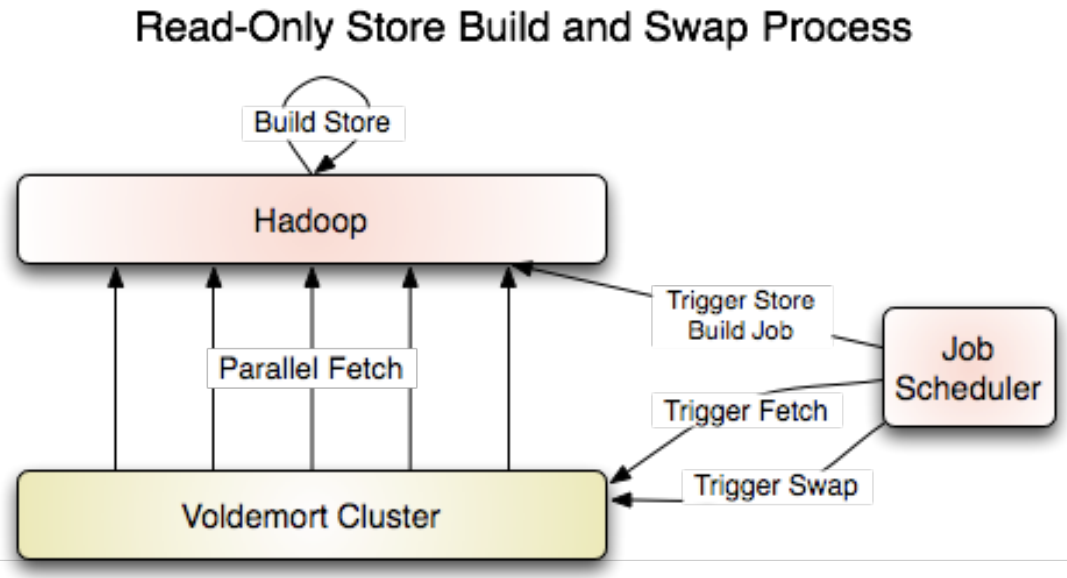
RO Store Usage Pattern

1. We schedule a Hadoop job to build a table (called “store” in Voldemort-speak)

2. Azkaban, our Hadoop scheduler, detects Hadoop job completion and tells Voldemort to fetch the new data and index files

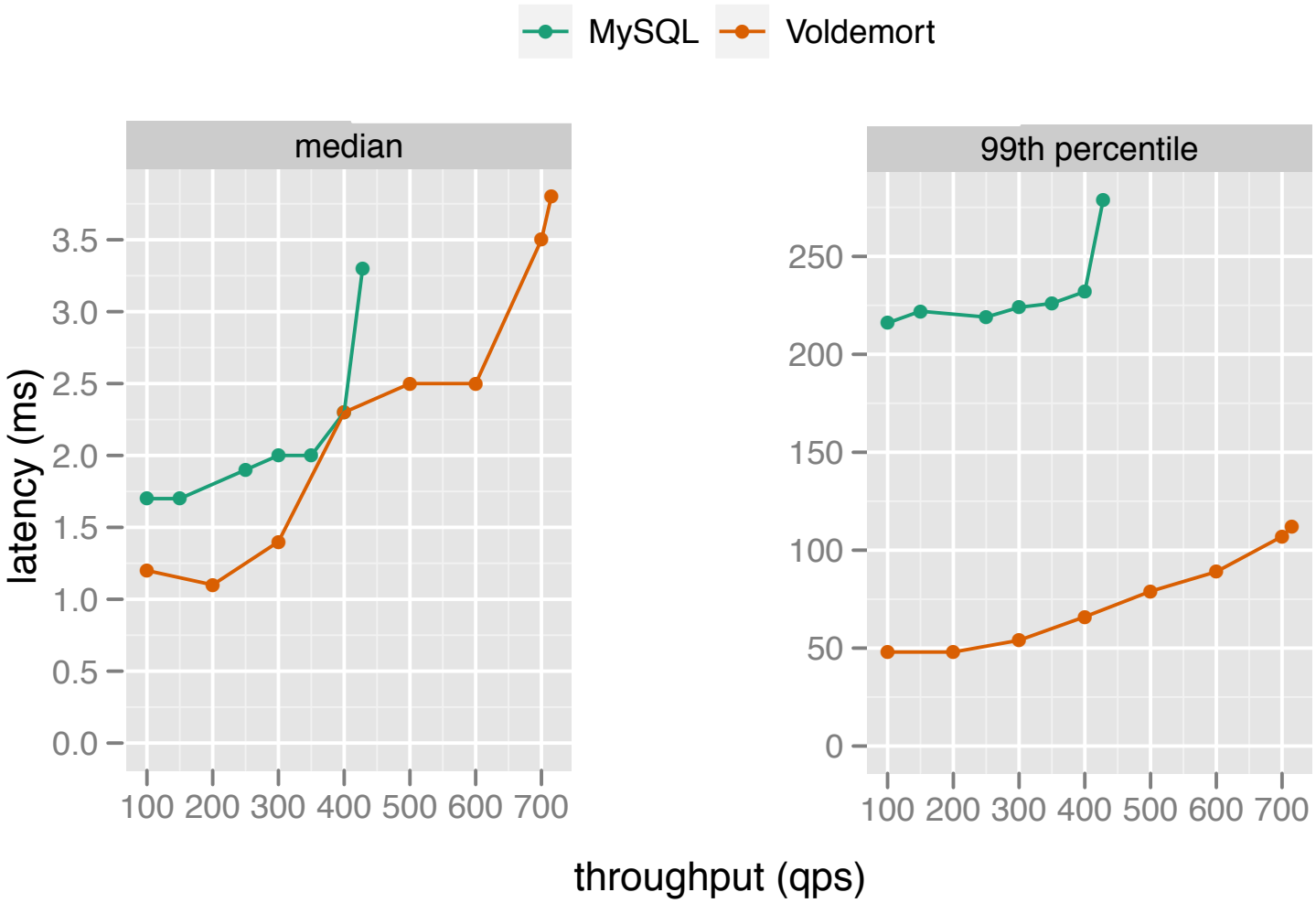
3. Voldemort fetches the data and index files in parallel from HDFS. Once fetch completes, swap indexes!

4. Voldemort serves fast key-value look-ups on the site
   - e.g. For key=“Sid Anand”, get all the people that “Sid Anand” may know!
   - e.g. For key=“Sid Anand”, get all the jobs that “Sid Anand” may be interested in!



How Do The Voldemort RO Stores Perform?

Voldemort : RO Store Performance : TP vs. Latency



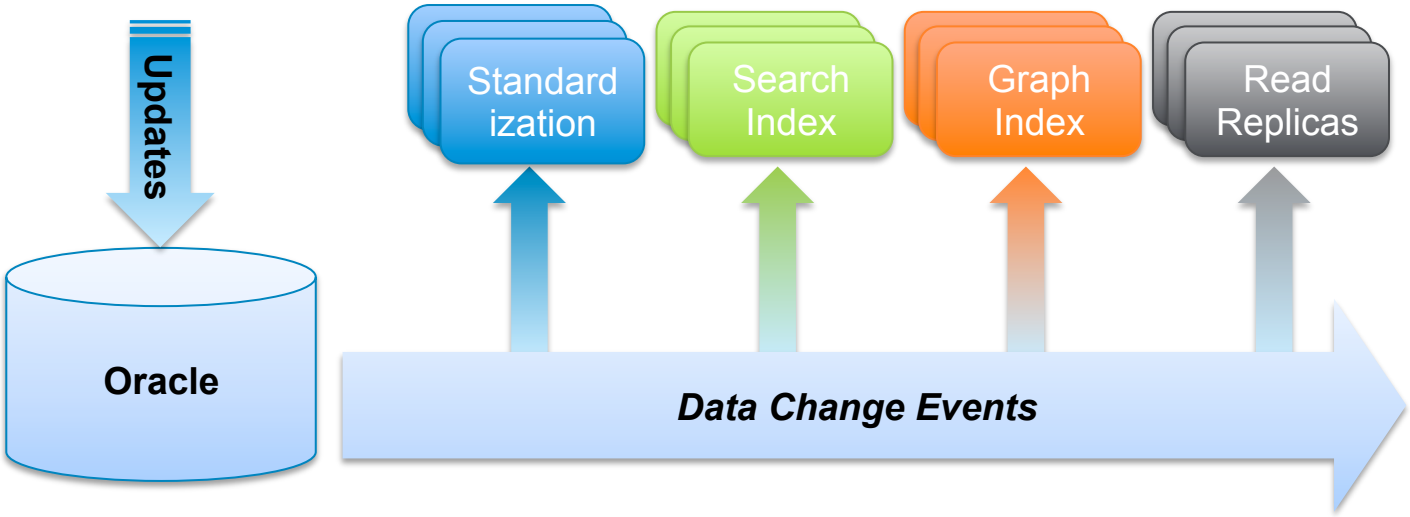
100 GB data, 24 GB RAM

LinkedIn Data Infrastructure Solutions

Databus : Timeline-Consistent Change Data Capture

Where Does LinkedIn use Databus?

Databus : Use-Cases @ LinkedIn

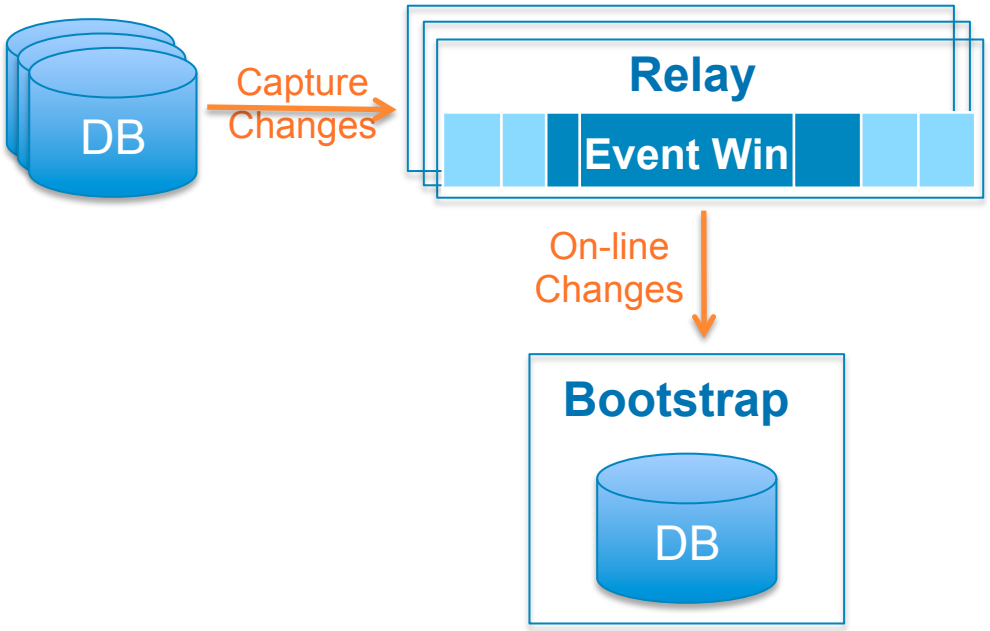


A user updates his profile with skills and position history. He also accepts a connection

- The write is made to an Oracle Master and Databus replicates:
- the profile change to the Standardization service
  - E.G. the many (actually 40) forms of IBM are canonicalized for search-friendliness and recommendation-friendliness
- the profile change to the Search Index service
  - Recruiters can find you immediately by new keywords
- the connection change to the Graph Index service
  - The user can now start receiving feed updates from his new connections immediately

Databus Architecture

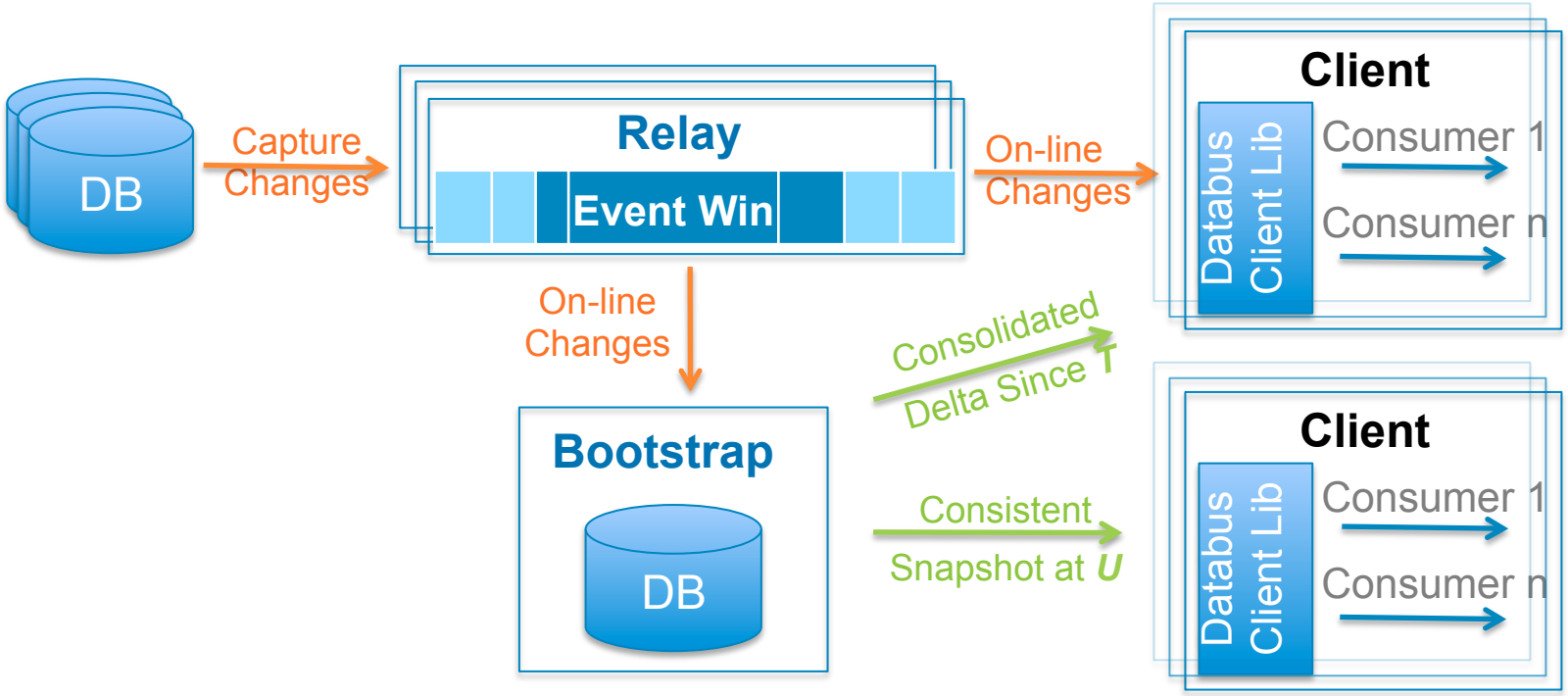
Databus : Architecture



Databus consists of 2 components

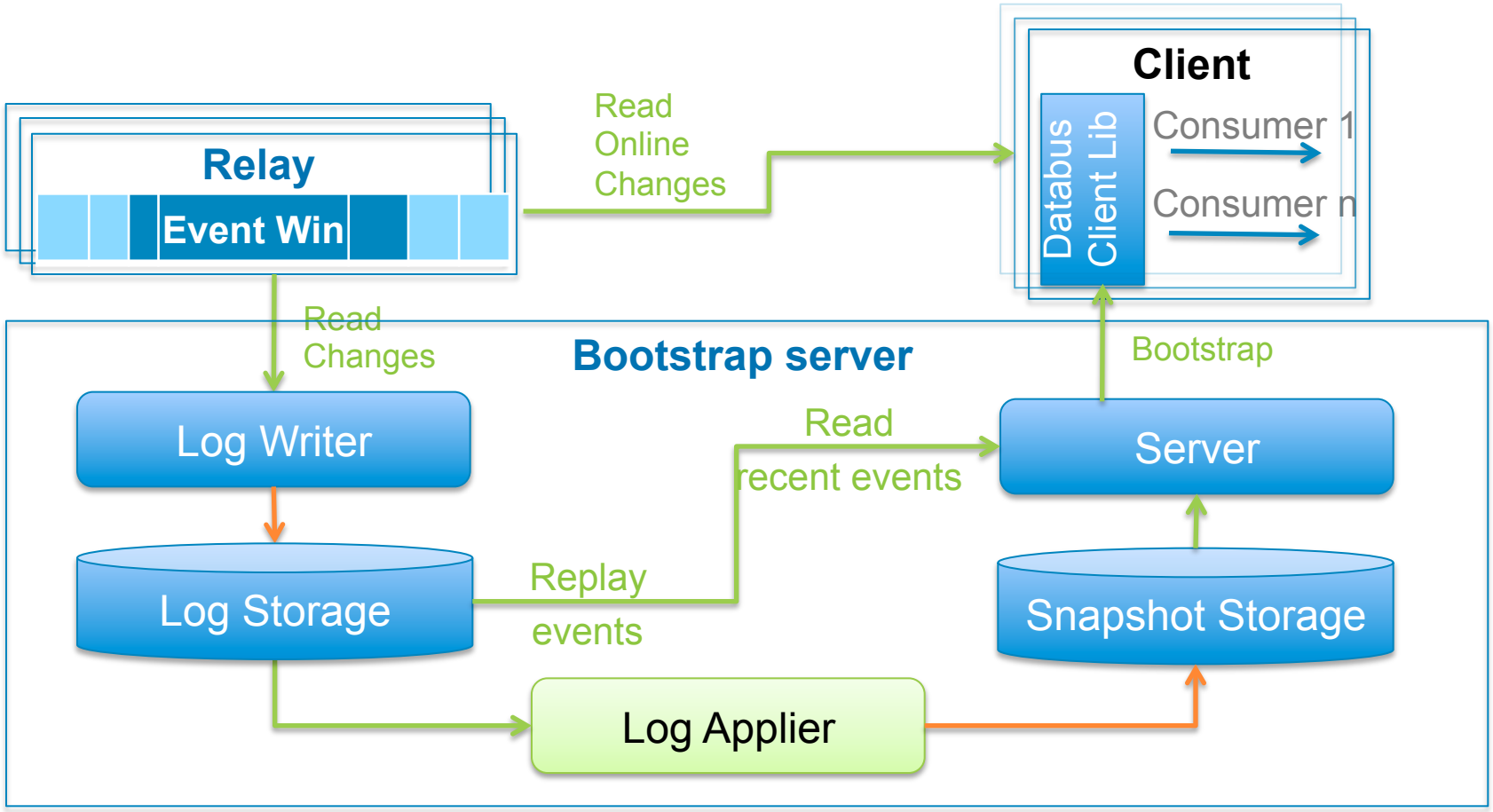
- **Relay Service**
  - "Relays" DB changes to subscribers as they happen in Oracle
  - Uses a sharded, in-memory, circular buffer
  - Very fast, but has limited amount of buffered data!
- **Bootstrap Service**
  - Serves historical data to subscribers who have fallen behind on the latest data from the "relay"
  - Not as fast as the relay, but has large amount of data buffered on disk

Databus : Architecture



Databus : Architecture - Bootstrap

- Generate consistent snapshots and consolidated deltas during continuous updates



LinkedIn Data Infrastructure Technologies

Espresso: Indexed Timeline-Consistent Distributed Data Store

Why Do We Need Yet Another NoSQL DB?

Espresso: Overview

What is Oracle Missing?

- **(Infinite) Incremental Scalability**
  - Adding 50% more resources gives us 50% more scalability (e.g. storage and serving capacity, etc…). In effect, we do not want to see diminishing returns
- **Always Available**
  - Users of the system do not perceive any service interruptions
    - Adding capacity or doing any other maintenance does not incur downtime
    - Node failures do not cause downtime
- **Operational Flexibility and Cost Control**
  - No recurring license fees
  - Runs on commodity hardware
- **Simplified Development Model**
  - E.g. Adding a column without running DDL ALTER TABLE commands

Espresso: Overview

What Features Should We Retain from Oracle?

- **Limited Transactions : Constrained Strong Consistency**
  - We need consistency within an entity (more on this later)
- **Ability to Query by Fields other than the Primary Key**
  - a.k.a. non-PK columns in RDBMS
- **Must Feed a Change Data Capture system**
  - i.e. can act as a Databus Source
  - Recall that a large ecosystem of analytic services are fed by the Oracle-Databus pipe. We need to continue to feed that ecosystem

Guiding Principles when replacing a system (e.g. Oracle)

- **Strive for Usage Parity, not Feature Parity**
  - In other words, first look at how you use Oracle and look for those features in candidate systems. Do not look for general feature parity between systems.
- **Don't shoot for a full replacement**
  - Buy yourself headroom by migrating the top K use-cases by load off Oracle. Leave the others.

What Does the API Look Like?

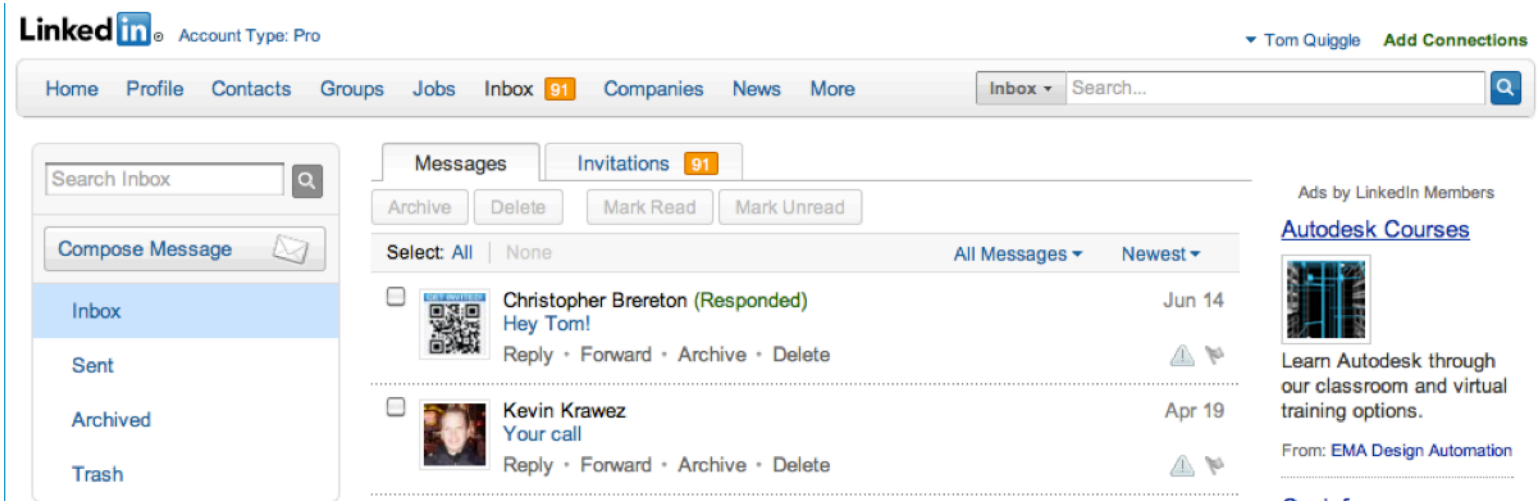
Espresso: API Example

Consider the User-to-User Communication Case at LinkedIn

- Users can view
  - Inbox, sent folder, archived folder, etc.…

- On social networks such as LinkedIn and Facebook, user-to-user messaging consumes substantial database resources in terms of storage and CPU (e.g. activity)
  - At LinkedIn 40% of Host DB CPU estimated to serve U2U Comm
  - Footnote : Facebook migrated their messaging use-case to HBase

- We're moving this off Oracle to Espresso



Espresso: API Example

Database and Tables

- Imagine that you have a Mailbox Database containing tables needed for the U2U Communications case
- The Mailbox Database contains the following 3 tables:
  - Message_Metadata – captures subject text
    - Primary Key = MemberId & MsgId
  - Message_Details – captures full message content
    - Primary Key = MemberId & MsgId
  - Mailbox_Aggregates – captures counts of read/unread & total
    - Primary Key = MemberId

Example Read Request

Espresso supports REST semantics.

To get unread and total email count for "**bob**", issue a request of the form:

- GET /<database_name>/<table_name>/<resource_id>
- GET /Mailbox/Mailbox_Aggregates/bob

Mailbox Database

Message Metadata Table

MemberId	MsgId	Value Blob
bob	1	Invitation to join Linkedin
bob	2	Job opportunity
bob	3	Request for referral
tom	1	Invitation to join Linkedin
tom	2	Job opportunity

Message Details Table

MemberId	MsgId	Value Blob
bob	1	"Dear Bob,...."
bob	2	"Hello there,...."
bob	3	"Good morning, "
tom	1	"Hi Tom,..."
tom	2	"Interesting opportunity"

Mailbox Aggregates Table

MemberId	Value Blob
bob	unread:20, total:100
tom	unread: 2, total: 25

Espresso: API Example

Collection Resources vs. Singleton Resources

A resource identified by a *resource_id* may be either a *singleton* or a *collection*

Examples (Read)

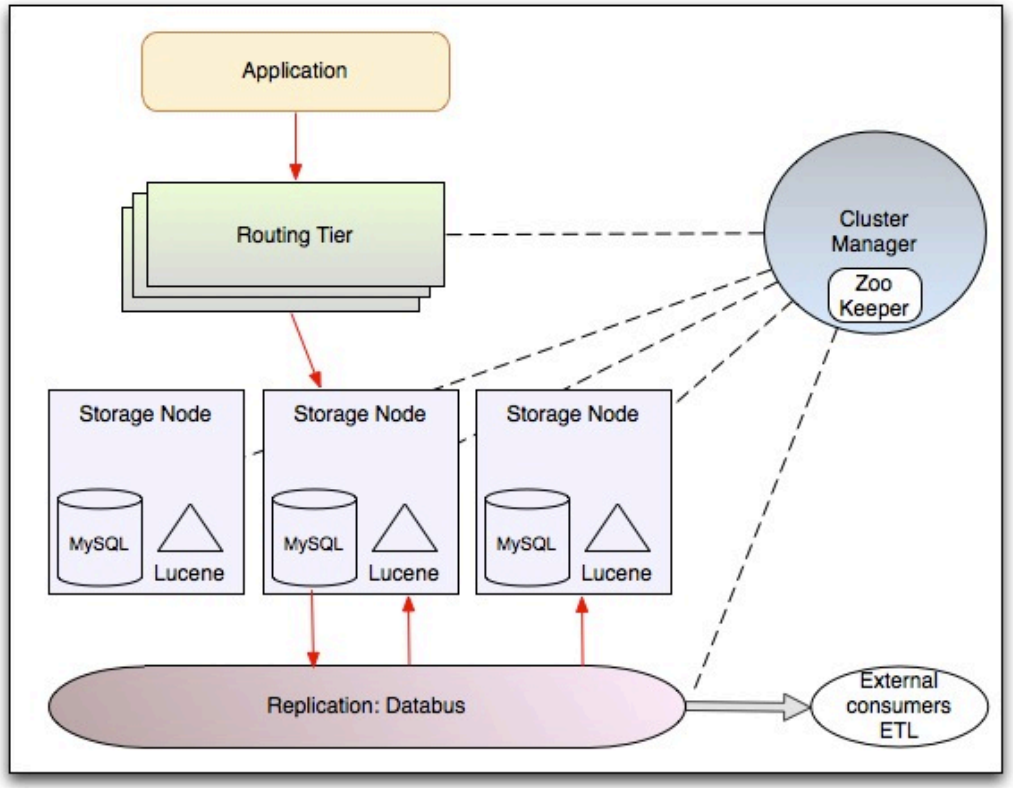
- For singletons, the URI refers to an individual resource.
  - GET /<database_name>/<table_name>/<resource_id>
  - E.g.: Mailbox_Aggregates table
    - To get unread and total email count for "**bob**", issue a request of the form:
      - GET /Mailbox/Mailbox_Aggregates/bob

- For collections, a secondary path element defines individual resources within the collection
  - GET /<database_name>/<table_name>/<resource_id>/<subresource_id>
  - E.g.: Message_Metadata & Message_Details tables
    - To get all of "**bob's**" mail metadata, specify the URI up to the <resource_id>
      - GET /Mailbox/Message_Metadata/bob
    - To display one of "**bob's**" messages in its entirety, specify the URI up to the <subresource_id>
      - GET /Mailbox/Message_Details/bob/4

What Does the Architecture Look Like?

Espresso: Architecture

- Components
  - Request Routing Tier
    - Stateless
    - Accepts HTTP request
    - Consults Cluster Manager to determine master for partition
    - Forwards request to appropriate storage node
  - Storage Tier
    - Data Store (e.g. MySQL)
    - Data is semi-structured (e.g. Avro)
    - Local Secondary Index (Lucene)
  - Cluster Manager
    - Responsible for data set partitioning
    - Notifies Routing Tier of partition locations for a data set
    - Monitors health and executes repairs on an unhealthy cluster
  - Relay Tier
    - Replicates changes in commit order to subscribers (uses Databus)



Next Steps for Espresso

Espresso: Next Steps

Key Road Map Items

- Internal Rollout for more use-cases within LinkedIn
- Development
  - Active-Active across data centers
  - Global Search Indexes : "<resource_id>?query=…" for singleton resources as well
  - More Fault Tolerance measures in the Cluster Manager (Helix)
- Open Source Helix (Cluster Manager) in Summer of 2012
  - Look out of an article in High Scalability
- Open Source Espresso in the beginning of 2013

Acknowledgments

Presentation & Content

- Tom Quiggle (**Espresso**) @TomQuiggle
- Kapil Surlaker (**Espresso**) @kapilsurlaker
- Shirshanka Das (**Espresso**) @shirshanka
- Chavdar Botev (**Databus**) @cbotev
- Roshan Sumbaly (**Voldemort**) @rsumbaly
- Neha Narkhede (**Kafka**) @nehanarkhede

A Very Talented Development Team

Aditya Auradkar, Chavdar Botev, Antony Curtis, Vinoth Chandar, Shirshanka Das, Dave DeMaagd, Alex Feinberg, Phanindra Ganti, Mihir Gandhi, Lei Gao, Bhaskar Ghosh, Kishore Gopalakrishna, Brendan Harris, Todd Hendricks, Swaroop Jagadish, Joel Koshy, Brian Kozumplik, Kevin Krawez, Jay Kreps, Shi Lu, Sunil Nagaraj, Neha Narkhede, Sasha Pachev, Igor Perisic, Lin Qiao, Tom Quiggle, Jun Rao, Bob Schulman, Abraham Sebastian, Oliver Seeliger, Rupa Shanbag, Adam Silberstein, Boris Shkolnik, Chinmay Soman, Subbu Subramaniam, Roshan Sumbaly, Kapil Surlaker, Sajid Topiwala, Cuong Tran, Balaji Varadarajan, Jemiah Westerman, Zhongjie Wu, Zach White, Yang Ye, Mammad Zadeh, David Zhang, and Jason Zhang

y Questions?