Data Infrastructure at LinkedIn

Shirshanka Das
XLDB 2011
Me

- UCLA Ph.D. 2005 (Distributed protocols in content delivery networks)
- PayPal (Web frameworks and Session Stores)
- Yahoo! (Serving Infrastructure, Graph Indexing, Real-time Bidding in Display Ad Exchanges)
- @ LinkedIn (Distributed Data Systems team): Distributed data transport and storage technology (Kafka, Databus, Espresso, ...)
Outline

- LinkedIn Products
- Data Ecosystem
- LinkedIn Data Infrastructure Solutions
- Next Play
LinkedIn By The Numbers

- 120,000,000+ users in August 2011
- 2 new user registrations per second
- 4 billion People Searches expected in 2011
- 2+ million companies with LinkedIn Company Pages
- 81+ million unique visitors monthly*
- 150K domains feature the LinkedIn Share Button
- 7.1 billion page views in Q2 2011
- 1M LinkedIn Groups

* Based on comScore, Q2 2011
Member Profiles

LinkedIn

Earn Your M.S. in Finance - 15 month program designed for busy working professionals. Click for Info.

Jeff Weiner
CEO at LinkedIn
Mountain View, California | Internet

Jeff Weiner @dtunkelang adds great examples to recent talk re: the importance of understanding mutual fit when recruiting.

Dream. Fit. Passion. thenoly channel.com | via DunkleTunkelang
A few days ago, our CEO Jeff Weiner led a session at LinkedIn on how to "close" candidates — that is, how to persuade candidates to join your team once you have found and interviewed them. Since not everyone has the opportunity...

5 hours ago · Like (1) · Comment · Send a message · Share · See all activity

Current
CEO at LinkedIn
Member, Board of Directors at DonorsChoose
Member, Board of Directors at Malaria No More

Past
Executive in Residence at Acceleration Partners
Executive in Residence at Greylock
Executive Vice President Network Division at Yahoo

Education
University of Pennsylvania - The Wharton School

Recommendations
8 people have recommended Jeff

Connections
500+ connections

Websites
Company Website

Social Media
Follow @jeffweiner

Public Profile
http://www.linkedin.com/in/jeffweiner08

Summary
Internet executive with over 18 years of experience, including general management of mid to large size organizations, corporate development, product development, business operations, and strategy.

Get Hired Faster
with Job Seeker Premium

- Get noticed with a Job Seeker Badge
- Move to the top as a Featured Applicant
- Contact recruiters directly with InMail

LinkedIn Premium

Jeff Weiner likes this update:

Mike Gamson Another step in the right direction for the Khan Academy... No more
Khan Academy Integrates With Digital Textbooks maasnable.com

The 12-minute lectures that Bill Gates has called "the start of a revolution" will now be linked with the material in some digital textbooks.

4 hours ago · Like (4) · Comment · Share

Jeff Weiner @dtunkelang adds great examples to recent talk re: the importance of understanding mutual fit when recruiting.

Dream. Fit. Passion. thenoly channel.com | via DunkleTunkelang
A few days ago, our CEO Jeff Weiner led a
Signal - faceted stream search
Outline

- LinkedIn Products
- Data Ecosystem
- LinkedIn Data Infrastructure Solutions
- Next Play
Three Paradigms: Simplifying the Data Continuum

**Online**
- Member Profiles
  - Company Profiles
  - Connections
  - Communications

**Nearline**
- Signal
  - Profile Standardization
  - News
  - Recommendations
  - Search
  - Communications

**Offline**
- People You May Know
  - Connection Strength
  - News
  - Recommendations
  - Next best idea

Activity that should be reflected immediately
Activity that should be reflected soon
Activity that can be reflected later
## Data Infrastructure Toolbox (Online)

<table>
<thead>
<tr>
<th>Capabilities</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key-value access</td>
<td>Voldemort</td>
</tr>
<tr>
<td>Rich structures (e.g. indexes)</td>
<td>Espresso, Oracle</td>
</tr>
<tr>
<td>Change capture capability</td>
<td>Zoie, Bobo, Sensei</td>
</tr>
<tr>
<td>Search platform</td>
<td></td>
</tr>
<tr>
<td>Graph engine</td>
<td>D-Graph</td>
</tr>
</tbody>
</table>
### Data Infrastructure Toolbox (Nearline)

<table>
<thead>
<tr>
<th>Capabilities</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change capture streams</td>
<td><strong>Databus</strong></td>
</tr>
<tr>
<td>Messaging for site events, monitoring</td>
<td><strong>Kafka</strong></td>
</tr>
<tr>
<td>Nearline processing</td>
<td><strong>Coming Soon!</strong></td>
</tr>
</tbody>
</table>
## Data Infrastructure Toolbox (Offline)

<table>
<thead>
<tr>
<th>Capabilities</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine learning, ranking, relevance</td>
<td>Hadoop, Hive, Pig Azkaban, RDBMS</td>
</tr>
<tr>
<td>Analytics on Social gestures</td>
<td>Coming Soon!</td>
</tr>
</tbody>
</table>
Laying out the tools
Outline

- LinkedIn Products
- Data Ecosystem
- LinkedIn Data Infrastructure Solutions
- Next Play
Focus on four systems in Online and Nearline

- **Data Transport**
  - Kafka
  - Databus

- **Online Data Stores**
  - Voldemort
  - Espresso
LinkedIn Data Infrastructure Solutions

Kafka: High-Volume Low-Latency Messaging System
**Kafka: Architecture**

**Scale**
- Billions of Events
- TBs per day
- Inter-colo: few seconds
- Typical retention: weeks

**Guarantees**
- At least once delivery
- Very high throughput
- Low latency
- Durability

---

**Broker Tier**
- Push Events
  - Sequential write
  - sendfile
  - Topic 1
  - Topic 2
  - Topic N

**Web Tier**
- Push Event
  - 100 MB/sec

**Consumers**
- Pull Events
  - Kafka Client Lib
  - Iterator 1
  - Iterator n
  - Offset Management

**Zookeeper**
- Topic, Partition Ownership
- Offset Management

---

- 100 MB/sec
- 200 MB/sec

- Typical retention: weeks
LinkedIn Data Infrastructure Solutions

Databus: Timeline-Consistent Change Data Capture
Databus at LinkedIn

Features
- Transport independent of data source: Oracle, MySQL, …
- Portable change event serialization and versioning
- Start consumption from arbitrary point

Guarantees
- Transactional semantics
- Timeline consistency with the data source
- Durability (by data source)
- At-least-once delivery
- Availability
- Low latency
LinkedIn Data Infrastructure Solutions

Voldemort: Highly-Available Distributed Data Store
Voldemort: Architecture

**Highlights**
- Open source
- Pluggable components
- Tunable consistency / availability
- Key/value model, server side “views”

**In production**
- Data products
- Network updates, sharing, page view tracking, rate-limiting, more…
- Future: SSDs, multi-tenancy
LinkedIn Data Infrastructure Solutions

Espresso: Indexed Timeline-Consistent Distributed Data Store
Espresso: Key Design Points

- Hierarchical data model
  - InMail, Forums, Groups, Companies
- Native Change Data Capture Stream
  - Timeline consistency
  - Read after Write
- Rich functionality within a hierarchy
  - Local Secondary Indexes
  - Transactions
  - Full-text search
- Modular and Pluggable
  - Off-the-shelf: MySQL, Lucene, Avro
Application View

Mailbox Database

Message Metadata Table

<table>
<thead>
<tr>
<th>MemberId</th>
<th>MsgId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>1</td>
<td>Invitation to join Linkedin</td>
</tr>
<tr>
<td>bob</td>
<td>2</td>
<td>Job opportunity</td>
</tr>
<tr>
<td>bob</td>
<td>3</td>
<td>Request for referral</td>
</tr>
<tr>
<td>tom</td>
<td>1</td>
<td>Invitation to join Linkedin</td>
</tr>
<tr>
<td>tom</td>
<td>2</td>
<td>Job opportunity</td>
</tr>
</tbody>
</table>

Mailbox Aggregates Table

<table>
<thead>
<tr>
<th>MemberId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>unread:20, total:100</td>
</tr>
<tr>
<td>tom</td>
<td>unread: 2, total: 25</td>
</tr>
</tbody>
</table>

Message Details Table

<table>
<thead>
<tr>
<th>MemberId</th>
<th>MsgId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>1</td>
<td>&quot;Dear Bob,...&quot;</td>
</tr>
<tr>
<td>bob</td>
<td>2</td>
<td>&quot;Hello there,...&quot;</td>
</tr>
<tr>
<td>bob</td>
<td>3</td>
<td>&quot;Good morning,&quot;</td>
</tr>
<tr>
<td>tom</td>
<td>1</td>
<td>&quot;Hi Tom,...&quot;</td>
</tr>
<tr>
<td>tom</td>
<td>2</td>
<td>&quot;Interesting opportunity&quot;</td>
</tr>
</tbody>
</table>
### Partitioning

#### Mailbox Database

<table>
<thead>
<tr>
<th>MemberId</th>
<th>MsgId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>1</td>
<td>Invitation to join LinkedIn</td>
</tr>
<tr>
<td>bob</td>
<td>2</td>
<td>Job opportunity</td>
</tr>
<tr>
<td>bob</td>
<td>3</td>
<td>Request for referral</td>
</tr>
<tr>
<td>tom</td>
<td>1</td>
<td>Invitation to join LinkedIn</td>
</tr>
<tr>
<td>tom</td>
<td>2</td>
<td>Job opportunity</td>
</tr>
</tbody>
</table>

#### Mailbox Aggregates Table

<table>
<thead>
<tr>
<th>MemberId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>unread 20, total: 100</td>
</tr>
<tr>
<td>tom</td>
<td>unread 2, total: 25</td>
</tr>
</tbody>
</table>

#### Message Details Table

<table>
<thead>
<tr>
<th>MemberId</th>
<th>MsgId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>1</td>
<td>&quot;Dear Bob,...&quot;</td>
</tr>
<tr>
<td>bob</td>
<td>2</td>
<td>&quot;Hello there,...&quot;</td>
</tr>
<tr>
<td>bob</td>
<td>3</td>
<td>&quot;Good morning,&quot;</td>
</tr>
<tr>
<td>tom</td>
<td>1</td>
<td>&quot;Hi Tom,...&quot;</td>
</tr>
<tr>
<td>tom</td>
<td>2</td>
<td>&quot;Interesting opportunity&quot;</td>
</tr>
</tbody>
</table>

#### Mailbox Database - Partition 1

<table>
<thead>
<tr>
<th>MemberId</th>
<th>MsgId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>1</td>
<td>Invitation to join LinkedIn</td>
</tr>
<tr>
<td>bob</td>
<td>2</td>
<td>Job opportunity</td>
</tr>
<tr>
<td>bob</td>
<td>3</td>
<td>Request for referral</td>
</tr>
</tbody>
</table>

#### Mailbox Aggregates Table

<table>
<thead>
<tr>
<th>MemberId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>unread 20, total: 100</td>
</tr>
</tbody>
</table>

#### Message Details Table

<table>
<thead>
<tr>
<th>MemberId</th>
<th>MsgId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>1</td>
<td>&quot;Dear Bob,...&quot;</td>
</tr>
<tr>
<td>bob</td>
<td>2</td>
<td>&quot;Hello there,...&quot;</td>
</tr>
<tr>
<td>bob</td>
<td>3</td>
<td>&quot;Good morning,&quot;</td>
</tr>
</tbody>
</table>

#### Mailbox Database - Partition 2

<table>
<thead>
<tr>
<th>MemberId</th>
<th>MsgId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>tom</td>
<td>1</td>
<td>Invitation to join LinkedIn</td>
</tr>
<tr>
<td>tom</td>
<td>2</td>
<td>Job opportunity</td>
</tr>
</tbody>
</table>

#### Mailbox Aggregates Table

<table>
<thead>
<tr>
<th>MemberId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>tom</td>
<td>unread 2, total: 25</td>
</tr>
</tbody>
</table>

#### Message Details Table

<table>
<thead>
<tr>
<th>MemberId</th>
<th>MsgId</th>
<th>Value Blob</th>
</tr>
</thead>
<tbody>
<tr>
<td>tom</td>
<td>1</td>
<td>&quot;Hi Tom,...&quot;</td>
</tr>
<tr>
<td>tom</td>
<td>2</td>
<td>&quot;Interesting opportunity&quot;</td>
</tr>
</tbody>
</table>
Partition Layout: Master, Slave

3 Storage Engine nodes, 2 way replication

Cluster Manager

Database

Node 1

Node 2

Node 3

Cluster

Node: 1
M: P.1 – Active
S: P.5 – Active

Partition: P.1
Node: 1

Partition: P.12
Node: 3

Partition: P.1
Node: 1
M: P.1
S: P.5

Partition: P.5
Node: 1
M: P.5
S: P.1

Partition: P.7
Node: 1
M: P.7
S: P.8

Master
Slave
Espresso: API

- REST over HTTP

- **Get Messages for bob**
  - GET /MailboxDB/MessageMeta/bob

- **Get MsgId 3 for bob**
  - GET /MailboxDB/MessageMeta/bob/3

- **Get first page of Messages for bob that are unread and in the inbox**
  - GET /MailboxDB/MessageMeta/bob/?query="+isUnread:true +isInbox:true"&start=0&count=15
Espresso: API Transactions

- Add a message to bob’s mailbox
  - transactionally update mailbox aggregates, insert into metadata and details.

```plaintext
POST /MailboxDB/*/bob HTTP/1.1
Content-Type: multipart/binary; boundary=1299799120
Accept: application/json
--1299799120
Content-Type: application/json
Content-Location: /MailboxDB/MessageStats/bob
Content-Length: 50
{"total":"+1", "unread":"+1"}

--1299799120
Content-Type: application/json
Content-Location: /MailboxDB/MessageMeta/bob
Content-Length: 332
{"from":"...", "subject":"..."}

--1299799120
Content-Type: application/json
Content-Location: /MailboxDB/MessageDetails/bob
Content-Length: 542
{"body":"..."}

--1299799120--
```
Espresso: System Components
Espresso @ LinkedIn

- First applications
  - Company Profiles
  - InMail
- Next
  - Unified Social Content PI
  - Member Profiles
  - Many more…
Espresso: Next steps

- Launched first application Oct 2011
- Open source 2012
- Multi-Datacenter support
- Log-structured storage
- Time-partitioned data
Outline

- LinkedIn Products
- Data Ecosystem
- LinkedIn Data Infrastructure Solutions
- Next Play
The Specialization Paradox in Distributed Systems

- **Good:** Build specialized systems so you can do each thing really well
- **Bad:** Rebuild distributed routing, failover, cluster management, monitoring, tooling
Generic Cluster Manager: Helix

- Generic Distributed State Model
- Centralized Config Management
- Automatic Load Balancing
- Fault tolerance
- Health monitoring
- Cluster expansion and rebalancing
- Open Source 2012
- Espresso, Databus and Search
Stay tuned for

- **Innovation**
  - Nearline processing
  - Espresso eco-system
  - Storage / indexing
  - Analytics engine
  - Search

- **Convergence**
  - Building blocks for distributed data management systems
Thanks!
Appendix
Espresso: Routing

- Router is a high-performance HTTP proxy
- Examines URL, extracts partition key
- Per-db routing strategy
  - Hash Based
  - Route To Any (for schema access)
  - Range (future)
- Routing function maps partition key to partition
- Cluster Manager maintains mapping of partition to hosts:
  - Single Master
  - Multiple Slaves
Espresso: Storage Node

- **Data Store (MySQL)**
  - Stores document as Avro serialized blob
  - Blob indexed by (partition key {, sub-key})
  - Row also contains limited metadata
    - Etag, Last modified time, Avro schema version

- Document Schema specifies per-field index constraints
- Lucene index per partition key / resource
Espresso: Replication

- MySQL replication of mastered partitions
- MySQL “Slave” is MySQL instance with custom storage engine
  - custom storage engine just publishes to databus
- Per-database commit sequence number
- Replication is Databus
  - Supports existing downstream consumers
- Storage node consumes from Databus to update secondary indexes and slave partitions