facebook

Petabyte Scale Data at Facebook

Dhruba Borthakur ,
Engineer at Facebook,
XLDB Conference at Stanford University, Sept 2012

Agenda

1. Types of Data
2. Data Model and API for Facebook Graph Data
3. SLTP (Semi-OLTP) and Analytics data
4. Immutable data store for photos, videos, etc
5. Why Hive?


facebook
December 2010

Four major types of storage systems

- **Online Transaction Processing Databases (OLTP)**
  - The Facebook Social Graph
- **Semi-online Light Transaction Processing Databases (SLTP)**
  - Facebook Messages and Facebook Time Series
- **Immutable DataStore**
  - Photos, videos, etc
- **Analytics DataStore**
  - Data Warehouse, Logs storage

Size and Scale of Databases

	Total Size	Technology	Bottlenecks
Facebook Graph	Single digit petabytes	MySQL and TAO	Random read IOPS
Facebook Messages and Time Series Data	Tens of petabytes	HBase and HDFS	Write IOPS and storage capacity
Facebook Photos	High tens of petabytes	Haystack	storage capacity
Data Warehouse	Hundreds of petabytes	Hive, HDFS and Hadoop	storage capacity

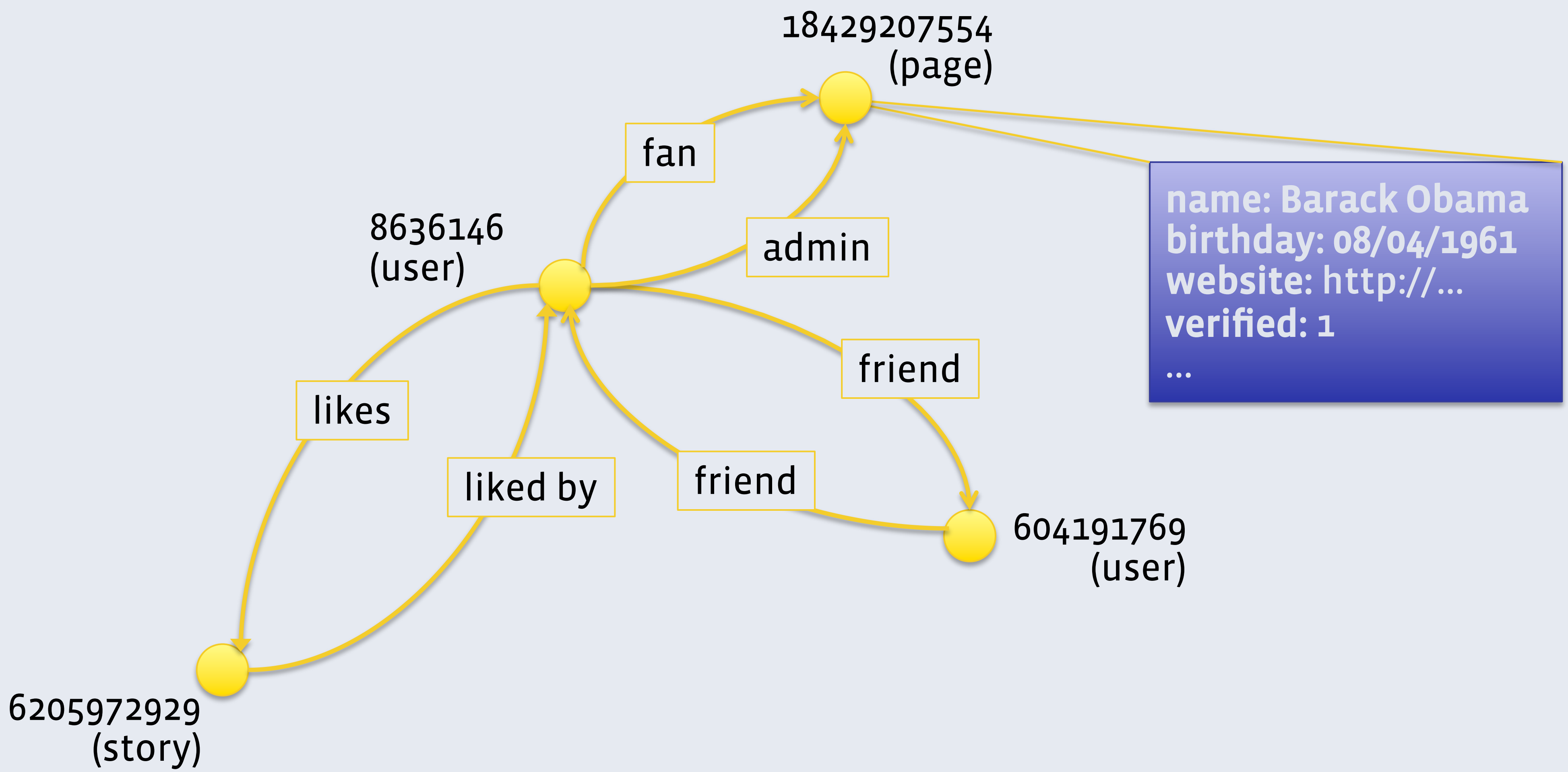
Characteristics

	Query Latency	Consistency	Durability
Facebook Graph	< few milliseconds	quickly consistent across data centers	No data loss
Facebook Messages and Time Series Data	< 200 millisec	consistent within a data center	No data loss
Facebook Photos	< 250 millisec	immutable	No data loss
Data Warehouse	< 1 min	not consistent across data centers	No silent data loss

Facebook Graph: Objects and Associations

Data model

Objects & Associations



Facebook Social Graph: TAO and MySQL

An OLTP workload:

- Uneven read heavy workload
- Huge working set with creation-time locality
- Highly interconnected data
- Constantly evolving
- As consistent as possible

Data model

Content aware data store

- Allows for server-side data processing
- Can exploit creation-time locality
- Graph data model
  - Nodes and Edges : Objects and Associations
- Restricted graph API

Data model

Objects & Associations

- Object -> unique 64 bit ID plus a typed dictionary
  - (id) -> (otype, (key -> value)*)
  - ID 6815841748 -> {‘type’: page, ‘name’: “Barack Obama”, … }
- Association -> typed directed edge between 2 IDs
  - (id1, atype, id2) -> (time, (key -> value)*)
  - (8636146, RSVP, 130855887032173) -> (1327719600, {‘response’: ‘YES’})
- Association lists
  - (id1, atype) -> all assocs with given id1, atype in desc order by time

Data model

API

- Object : (id) -> (otype, (key -> value)*)
  - obj_add(otype, (k->v)*) : creates new object, returns its id
  - obj_update(id, (k->v)*) : updates some or all fields
  - obj_delete(id): removes the object permanently
  - obj_get(id) : returns type and fields of a given object if exists

Data model

API

- Association : (id1, atype, id2) -> (time, (key -> value)*)
  - assoc_add(id1, atype, id2, time, (k->v)*) : adds/updates the given assoc
  - assoc_delete(id1, atype, id2) : deletes the given association

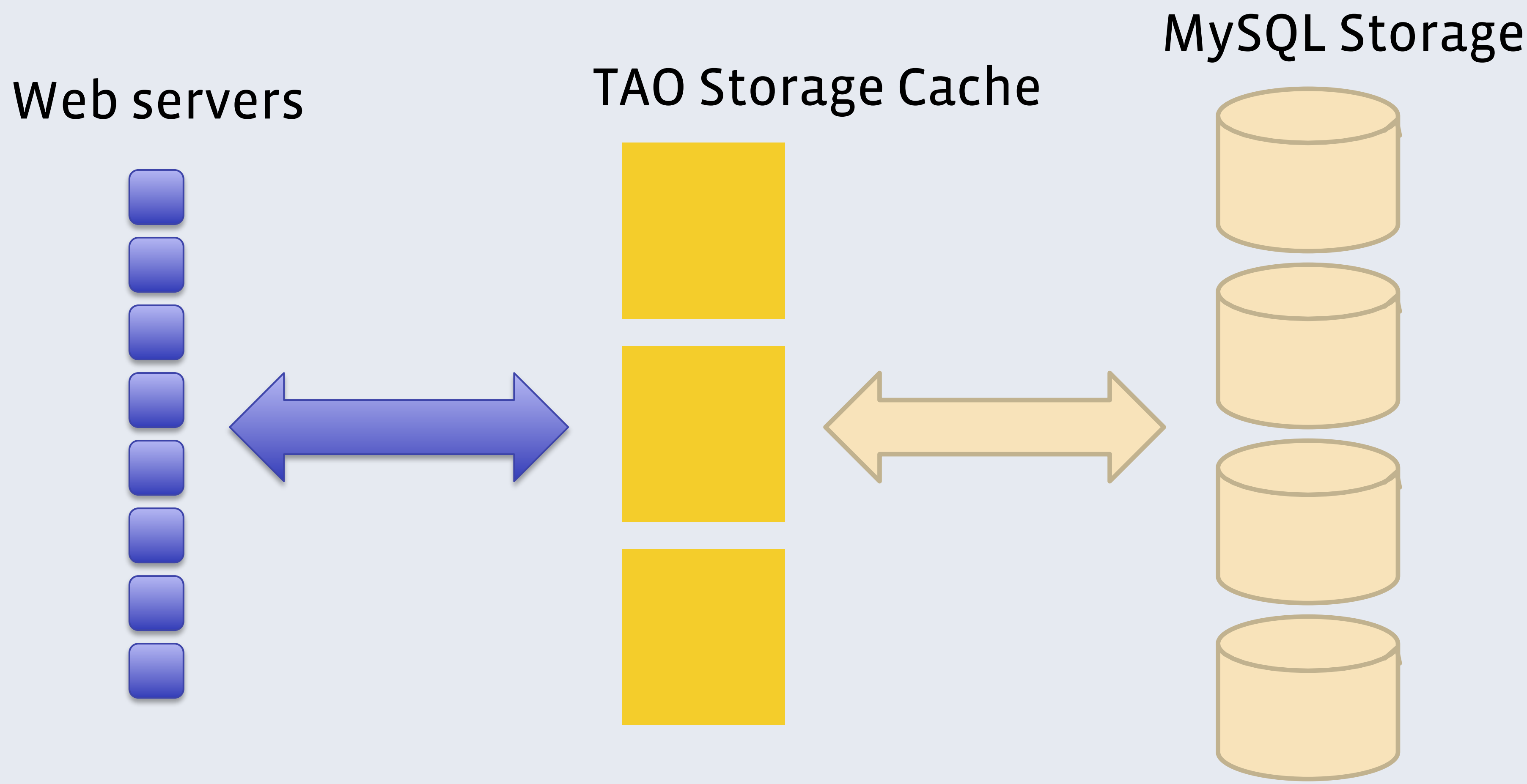
Data model

API

- Association : (id1, atype, id2) -> (time, (key -> value)*)
  - assoc_get(id1, atype, id2set) : returns assocs where id2 $\in$ id2set
  - assoc_range(id1, atype, offset, limit, filters*): get relevant matching assocs from the given assoc list
  - assoc_count(id1, atype): returns size of given assoc list

Architecture

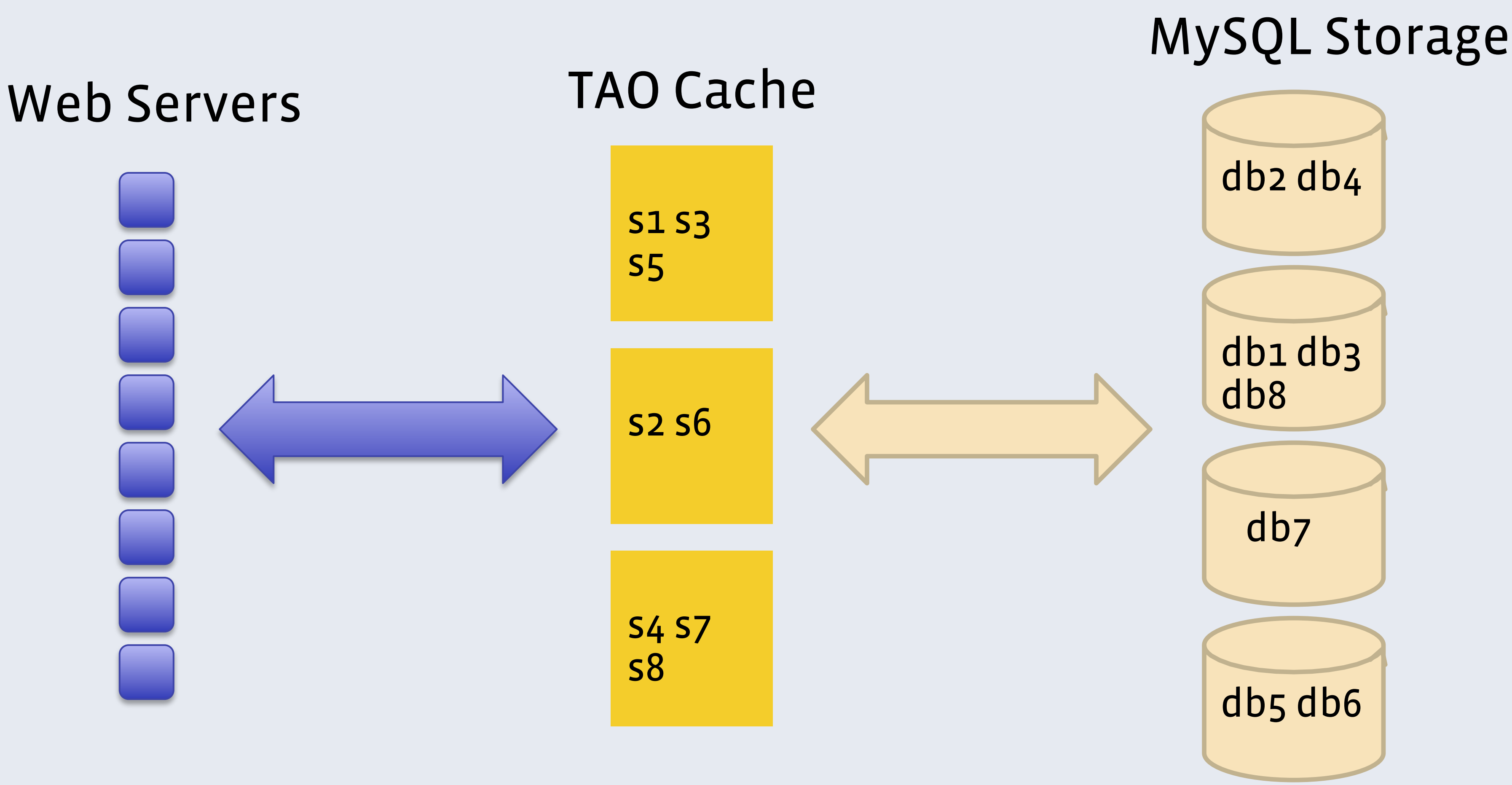
Cache & Storage



Architecture

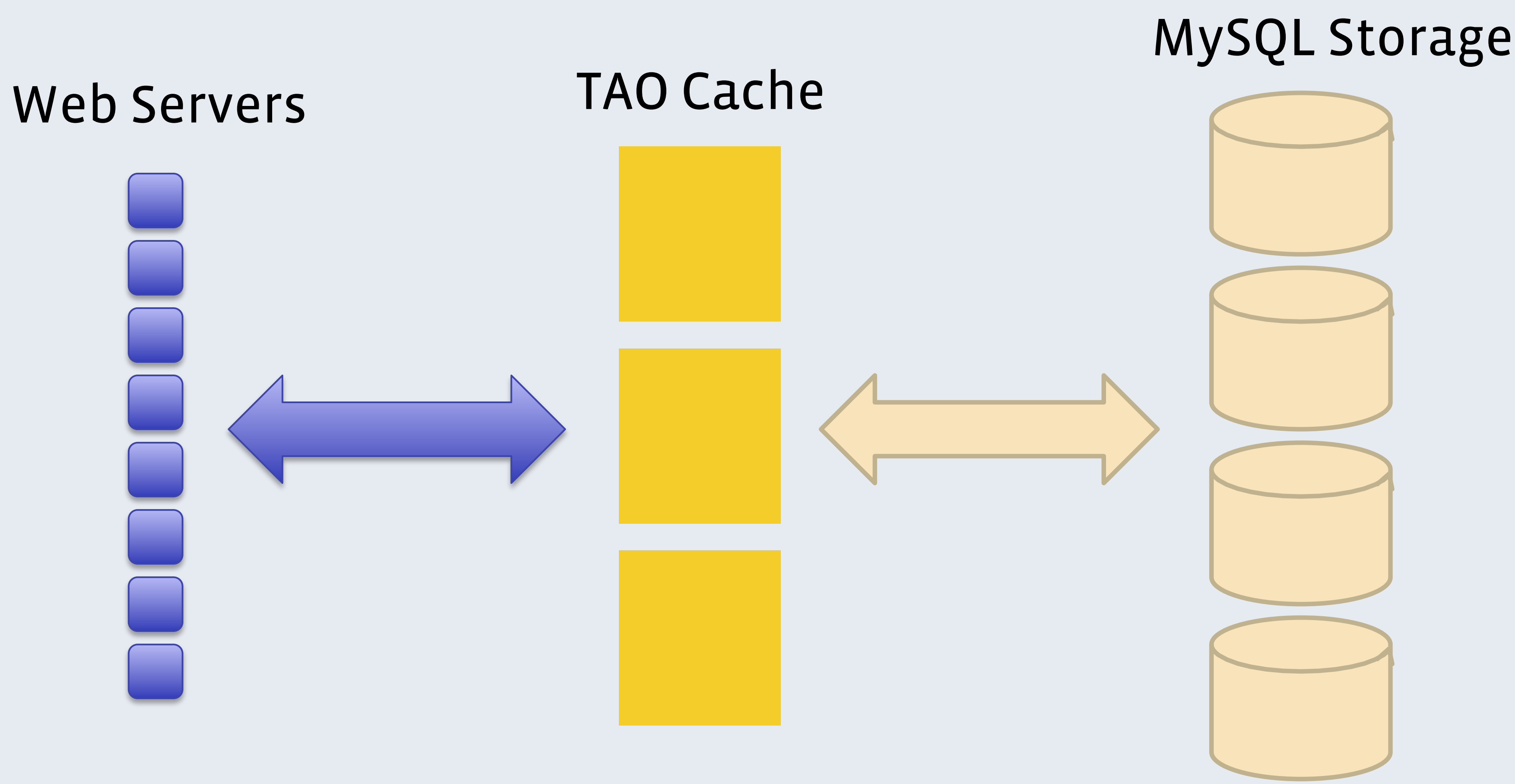
Sharding

- Object ids and Assoc id1s are mapped to shard ids



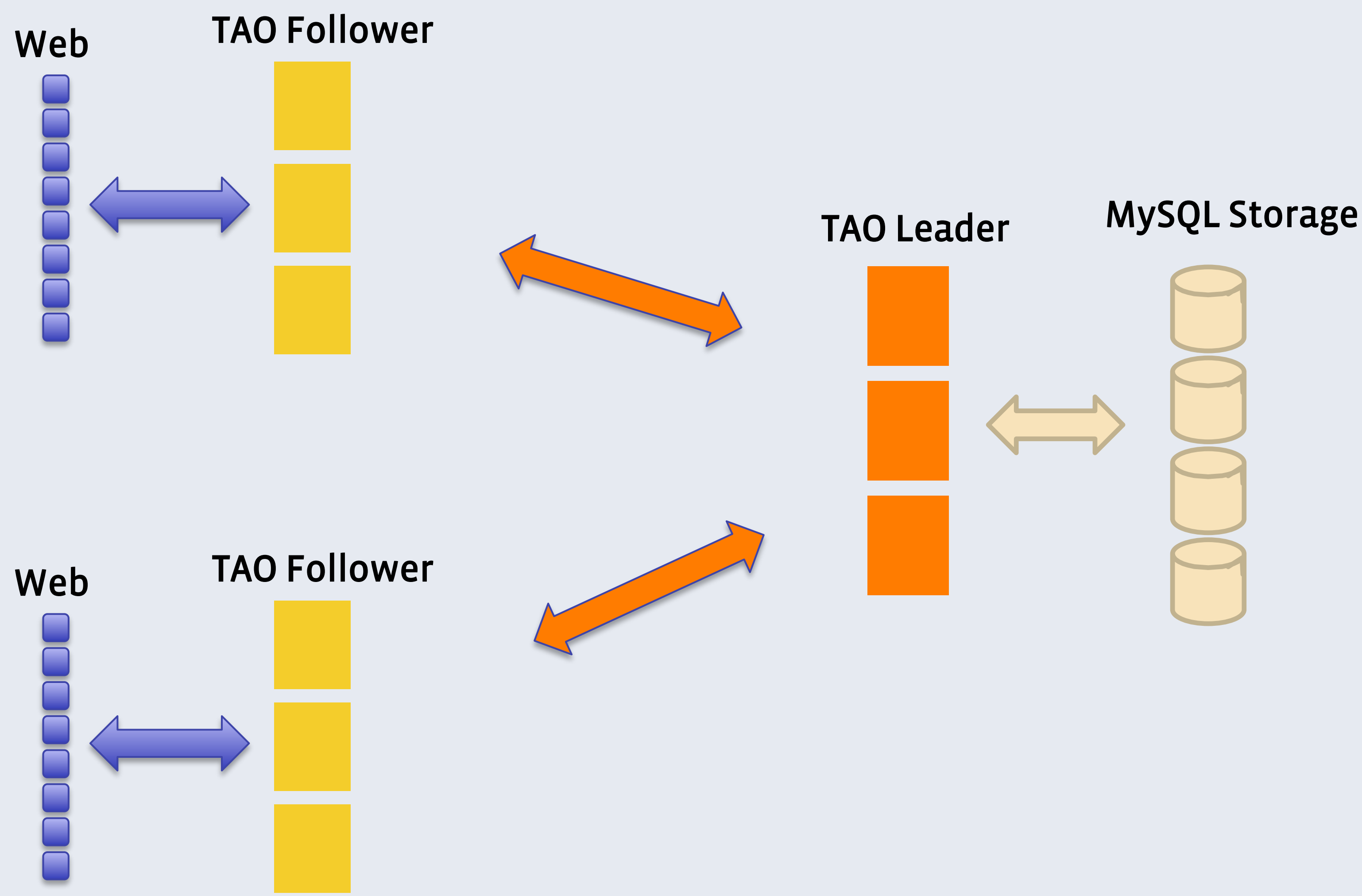
Architecture

Scale independently



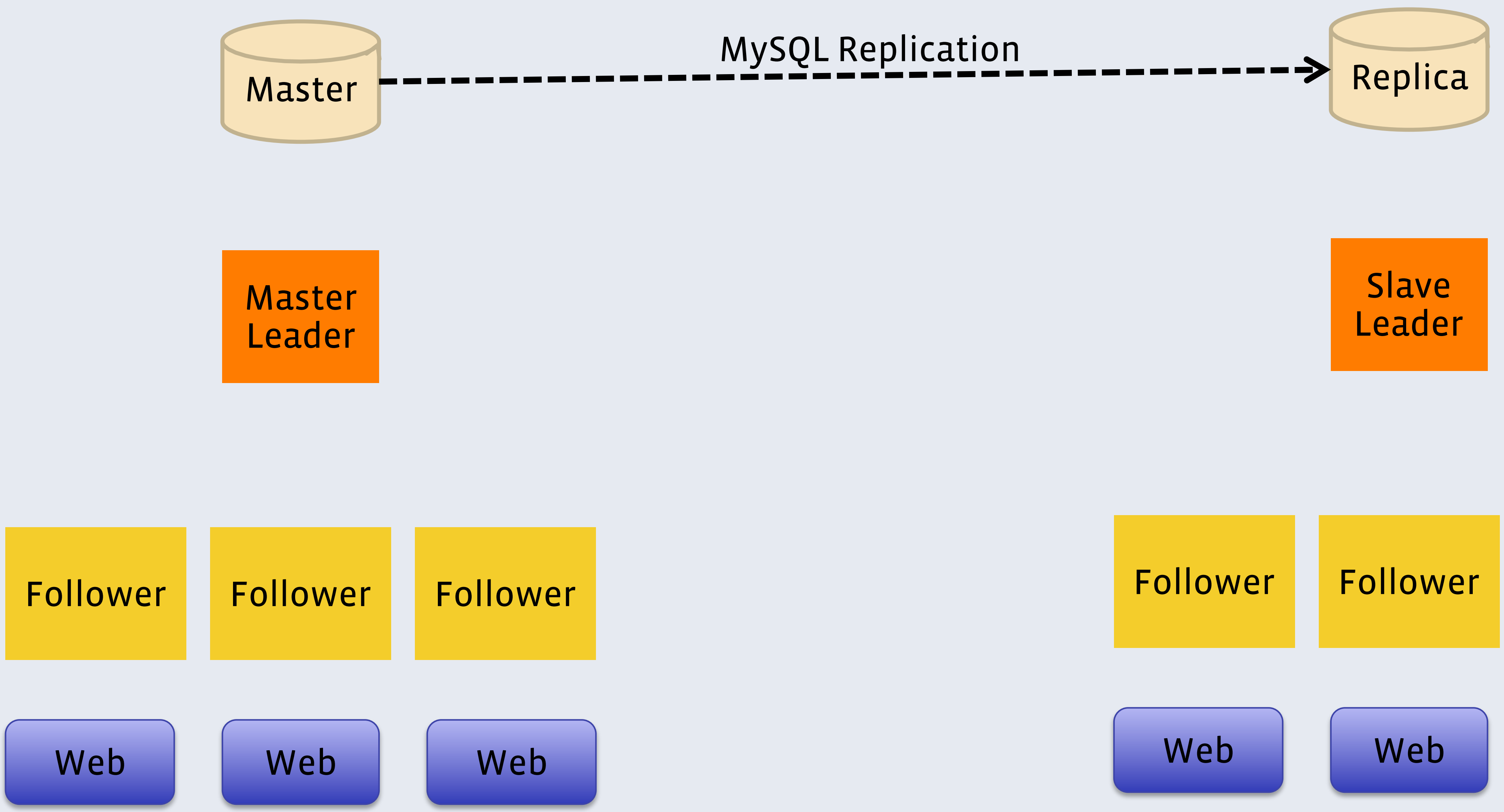
Architecture

Leaders and followers



Architecture

Multiple regions

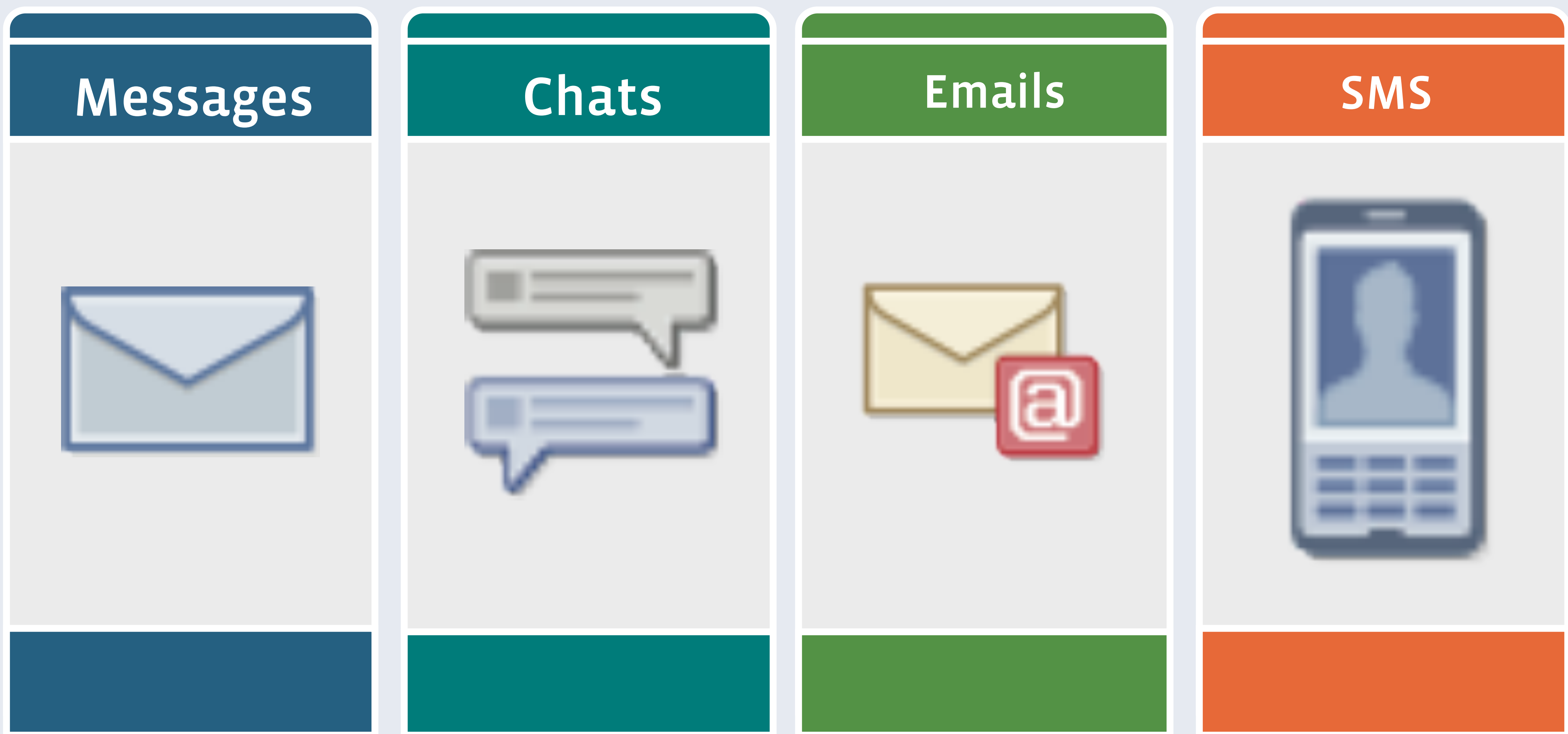


Workload

- **Read-heavy workload**
  - Significant range queries
- **LinkBench benchmark being open-sourced**
  - http://www.github.com/facebook/linkbench
- **Real data distribution of Assocs and their access patterns**

Messages & Time Series Database
SLTP workload

Facebook Messages



Why we chose HBase

- **High write throughput**
- **Horizontal scalability**
- **Automatic Failover**
- **Strong consistency within a data center**
- **Benefits of HDFS** : Fault tolerant, scalable, Map-Reduce toolset,
- **Why is this SLTP?**
  - Semi-online: Queries run even if part of the database is offline
  - Light Transactions: single row transactions
  - Storage capacity bound rather than iops or cpu bound

What we store in HBase

- Small messages
- Message metadata (thread/message indices)
- Search index
- Large attachments stored in Haystack (photo store)

Size and scale of Messages Database

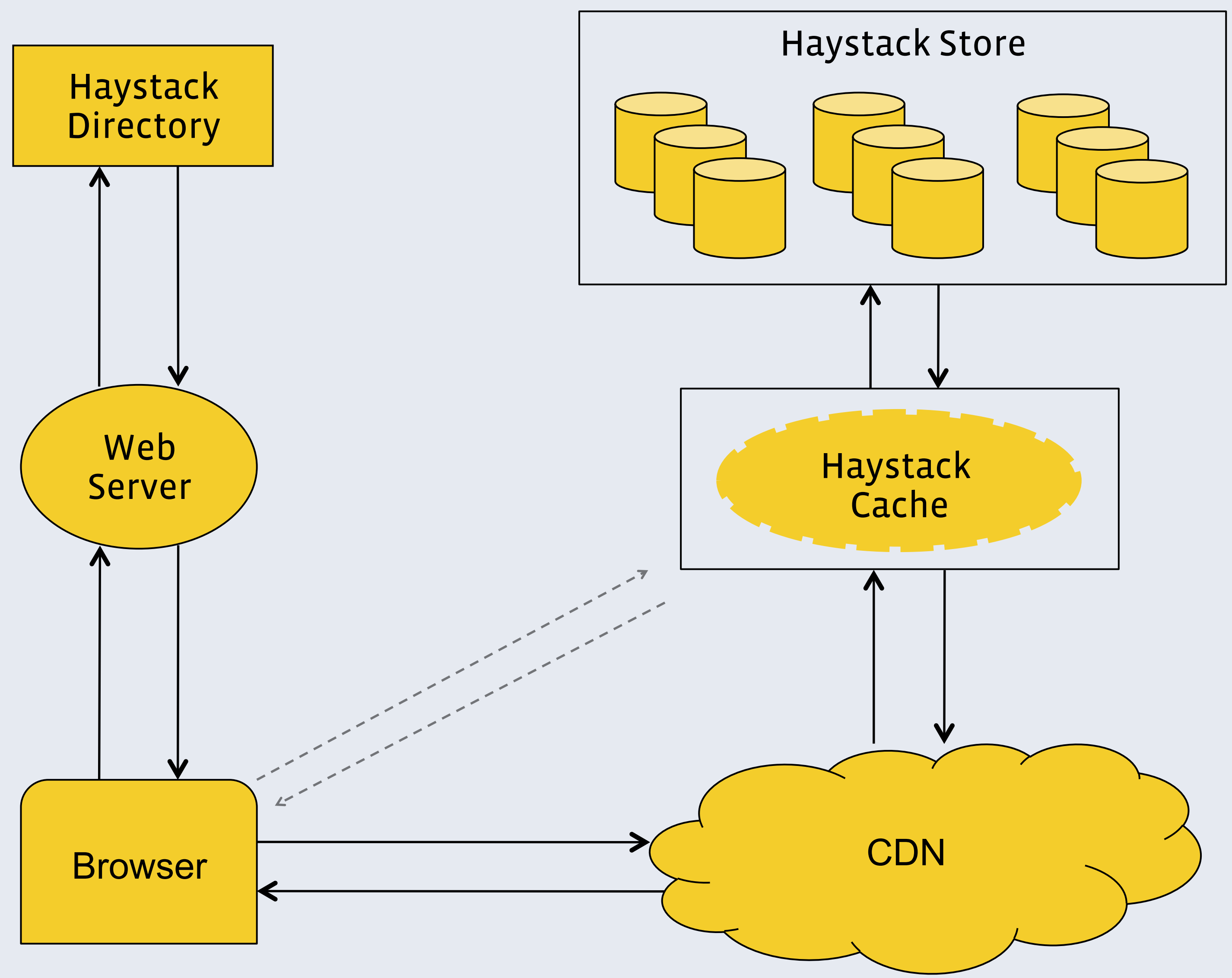
- 6 Billion messages/day
- 74 Billion operations/day
- At peak: 1.5 million operations/sec
- 55% read, 45% write operations
- Average write operation inserts 16 records
- All data is lzo compressed
- Growing at 8 TB/day

Haystack: The Photo Store

Facebook Photo DataStore

	2009	2012
Total Size	15 billion photos 1.5 Petabyte	High tens of petabytes
Upload Rate	30 million photos/day 3 TB/day	300 million photos/day 30 TB/day
Serving Rate	555K images/sec	

Haystack based Design

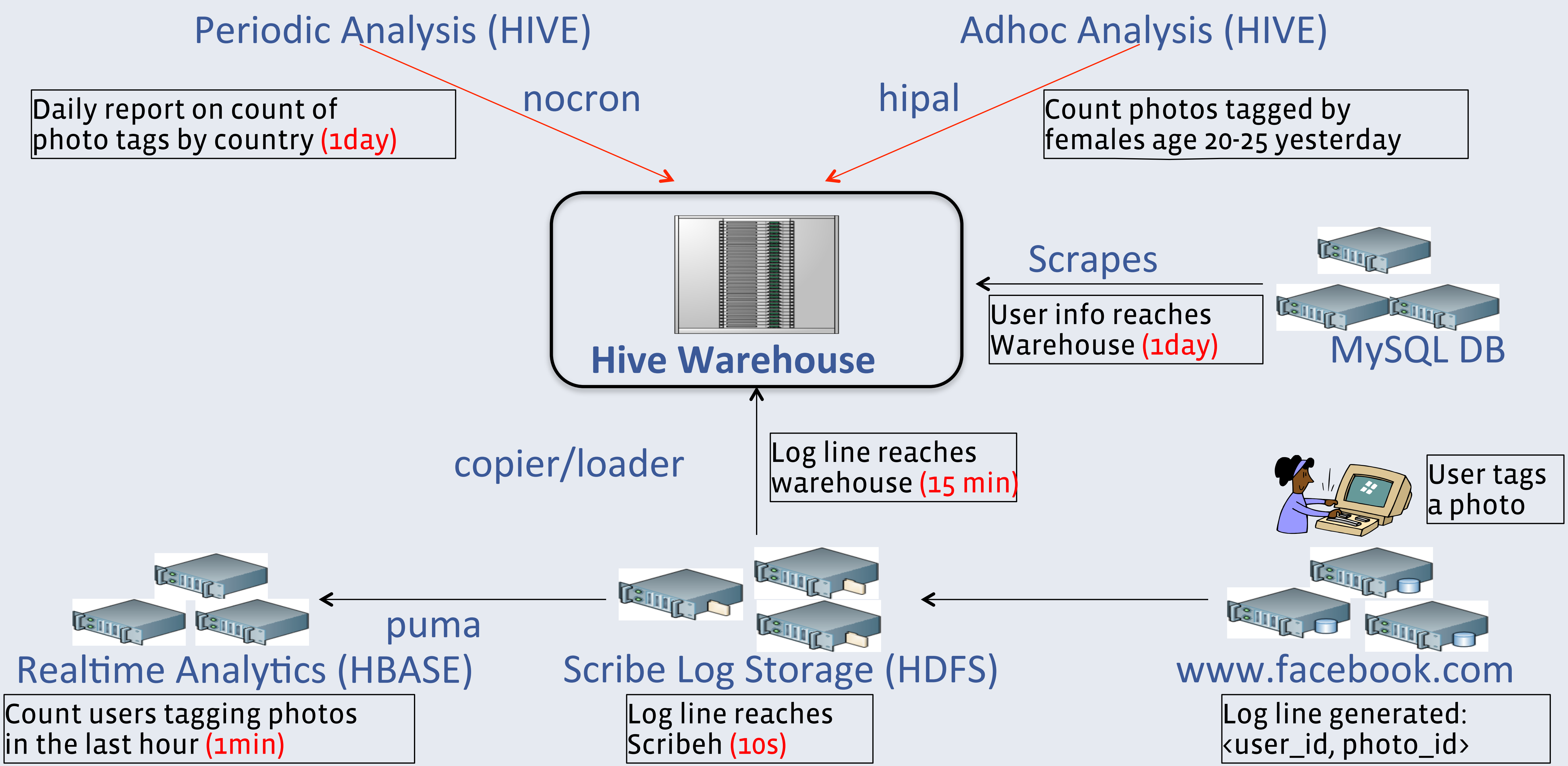


Haystack Internals

- Log structured, append-only object store
- Built on commodity hardware
- Application-aware replication
- Images stored in 100 GB xfs-files called needles
- An in-memory index for each needle file
  - 32 bytes of index per photo

Hive Analytics Warehouse

Life of a photo tag in Hadoop/Hive storage



Analytics Data Growth(last 4 years)

	Facebook Users	Queries/Day	Scribe Data/ Day	Nodes in warehouse	Size (Total)
Growth	14X	60X	250X	260X	2500X

Why use Hive instead of a Parallel DBMS?

- Stonebraker/DeWitt from the DBMS community:
  - Quote “major step backwards”
  - Published benchmark results which show that Hive is not as performant as a traditional DBMS
  - http://database.cs.brown.edu/projects/mapreduce-vs-dbms/

Why Hive?

- **Prospecting for gold in the wild-west…..**
  - A platform for huge data-experiments
  - A majority of queries are searching for a single gold nugget
  - Great advantage in keeping all data in one queryable system
  - No structure to data, specify structure at query time
- **Crowd Sourcing for data discovery**
  - There are 50K tables in a single warehouse
  - Users are DBAs themselves
  - Questions about a table are directed to users of that table
  - Automatic query lineage tools help here

Why Hive?

- **No Lock-in**
  - Hive/Hadoop is open source
  - Data is open-format, one can access data below the database layer
  - Want a new UDF? No problem, Very easily extendable

- **Shortcomings of existing DBMS benchmarks**
  - Does not test fault tolerance – kill machines
  - Does not measure elasticity – add and remove machines
  - Does not measure throughput – concurrent queries in parallel

Future Challenges

New trends in storage software

- **Trends:**
  - SSDs cheaper, increasing number of CPUs per server
  - SATA disk capacities reaching 4 - 8 TB per disk, falling prices $/GB
- **New projects**
  - Evaluate OLTP databases that scales linearly with the number of cps
  - Prototype storing cold photos on spin-down disks

Questions?
dhruba@fb.com

http://hadoopblog.blogspot.com/