facebook

Petabyte Scale Data at Facebook

Dhruba Borthakur ,
Engineer at Facebook,
SIGMOD, New York, June 2013

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?

December 2010

Four major types of storage systems

- **Online Transaction Processing Databases (OLTP)**
  - The Facebook Social Graph
- **Semi-online Lightweight 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	Hundreds of petabytes	Haystack	storage capacity
Data Warehouse	Hundreds of petabytes	Hive, HDFS and Hadoop	storage capacity

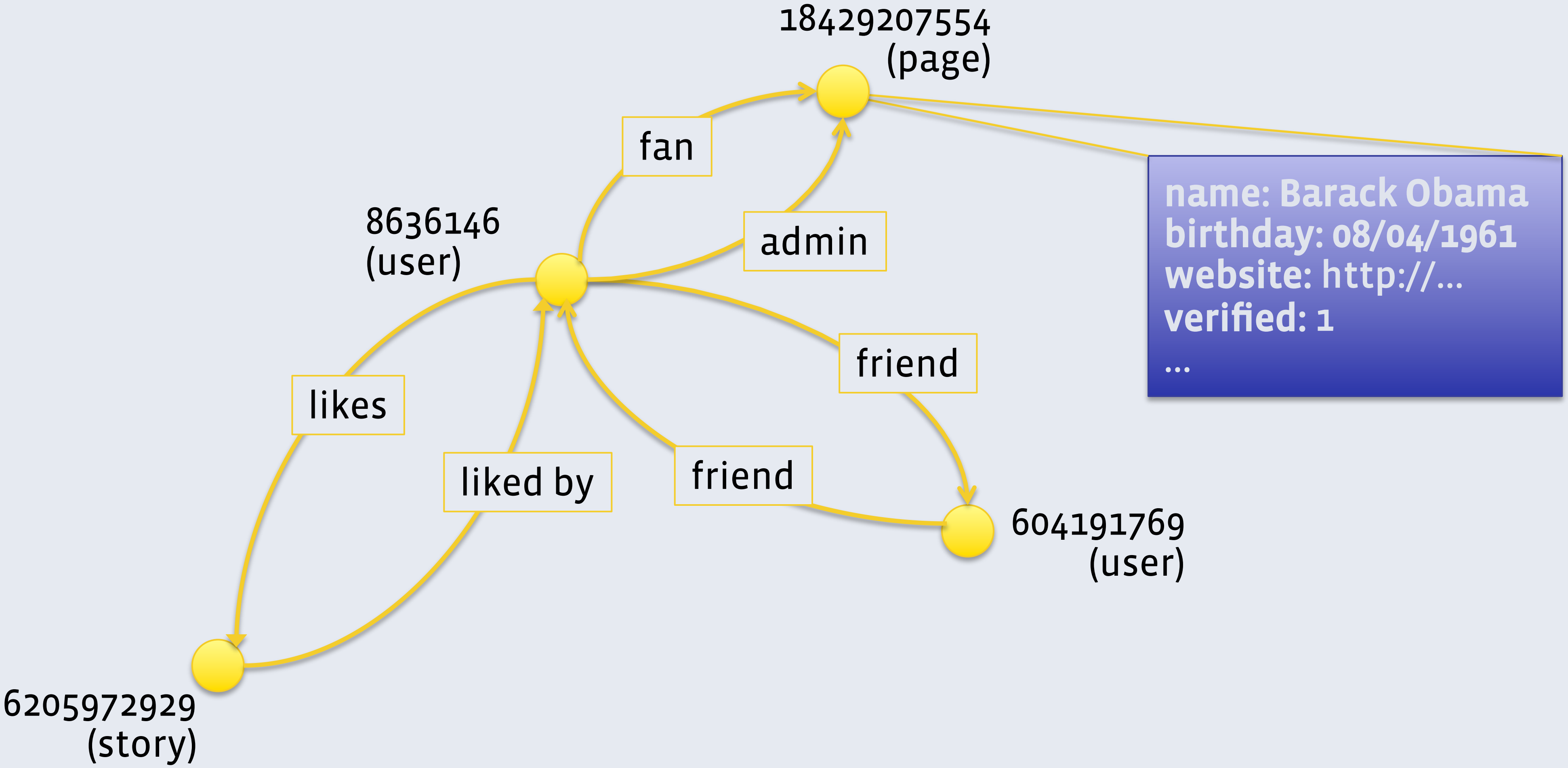
Characteristics

	Query Latency	Consistency	Durability
Facebook Graph	< few milliseconds	quickly consistent across data centers	No data loss
Facebook Messages and Time Series Data	< 100 millisec	consistent within a data center	No data loss
Facebook Photos	< 100 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



name: Barack Obama
birthday: 08/04/1961
website: http://...
verified: 1
...

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

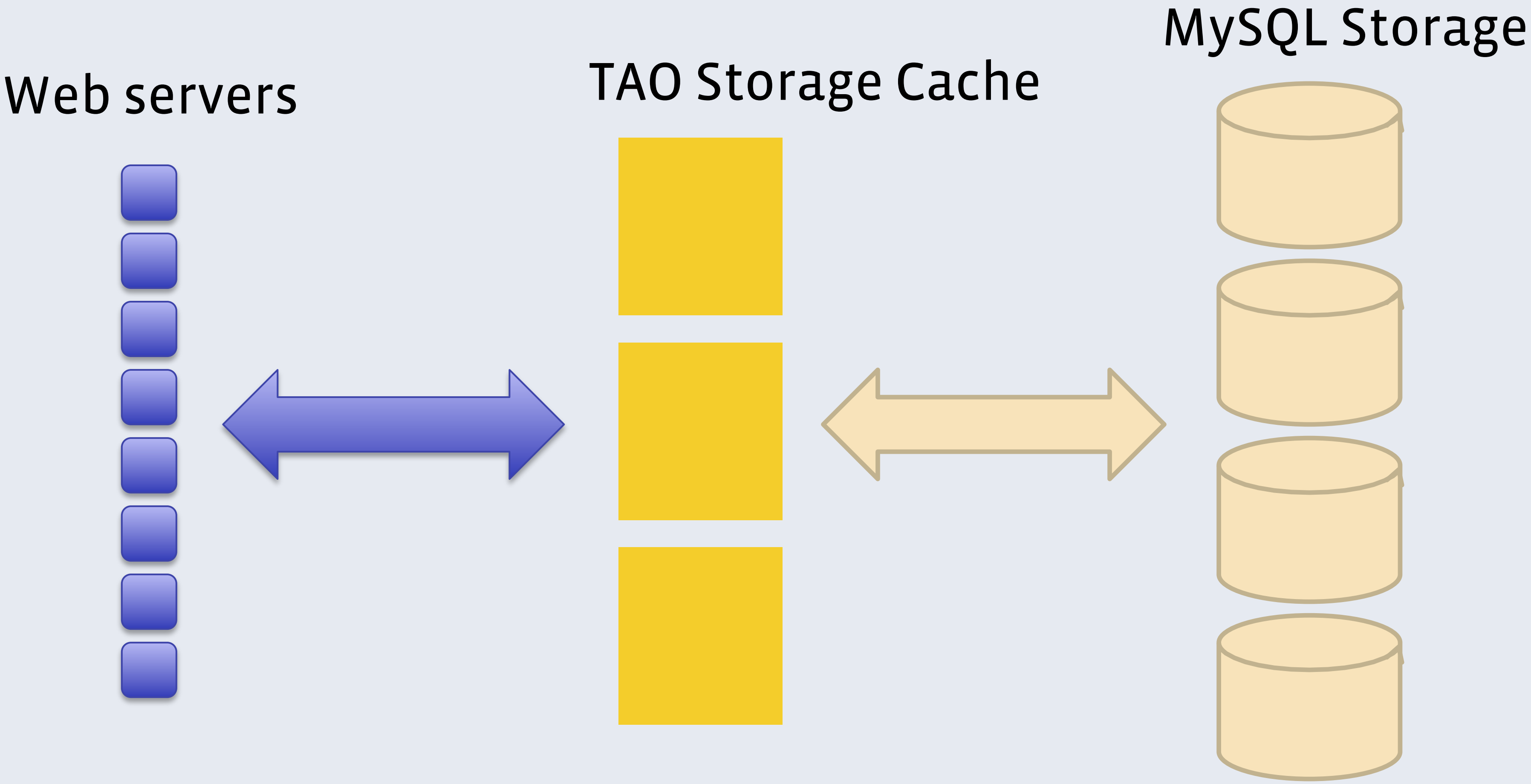
Data model

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

Architecture

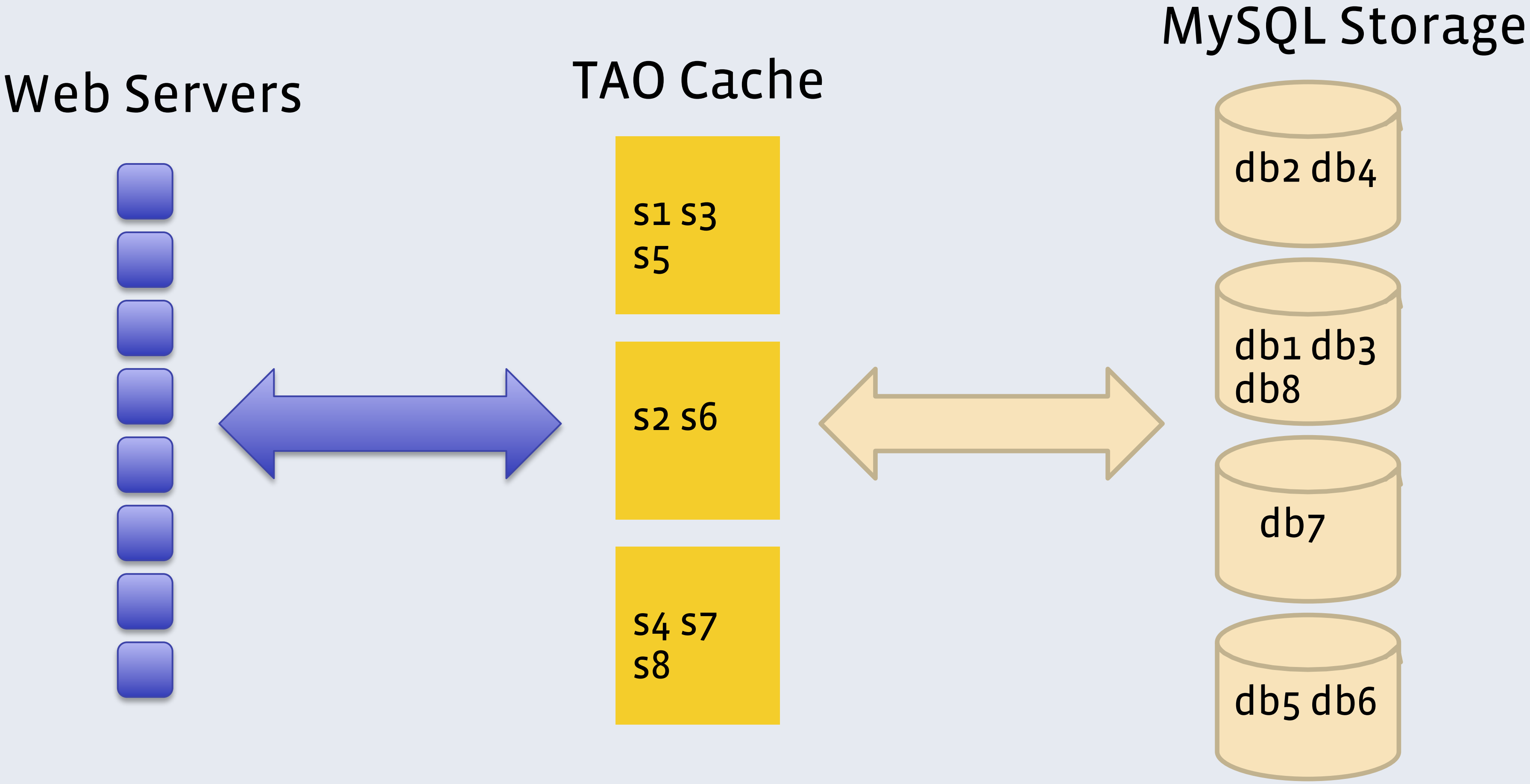
Cache & Storage



Architecture

Sharding

- Object ids and Assoc id1s are mapped to shard ids

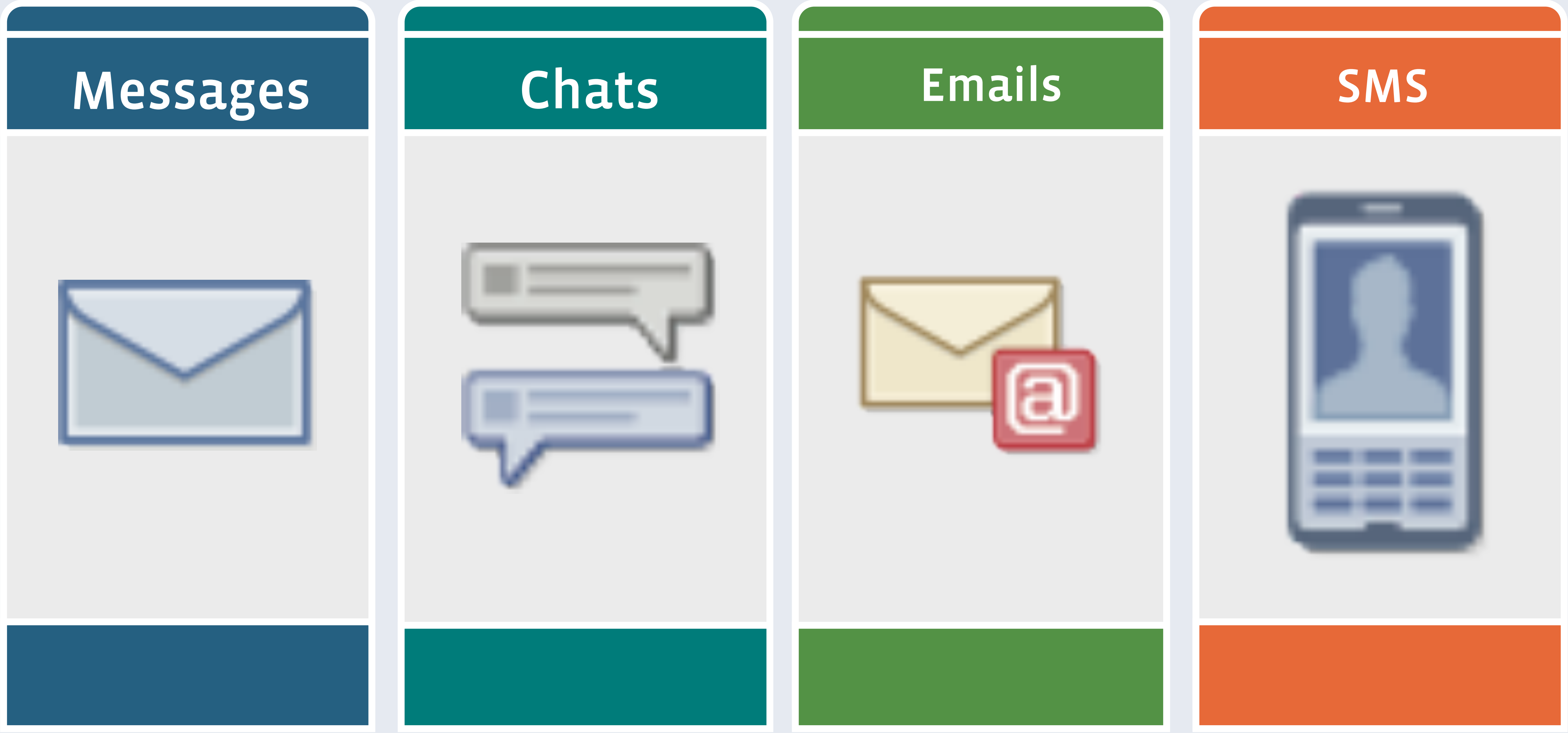


Workload

- **Read-heavy workload**
  - Significant range queries
- **LinkBench benchmark SIGMOD 2013 paper**
  - http://www.github.com/facebook/linkbench
  - Real distribution of associations and access patterns

Messages & Time Series Database SLTP workload

Facebook Messages



Messages

Chats

Emails

SMS

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
  - Lightweight 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

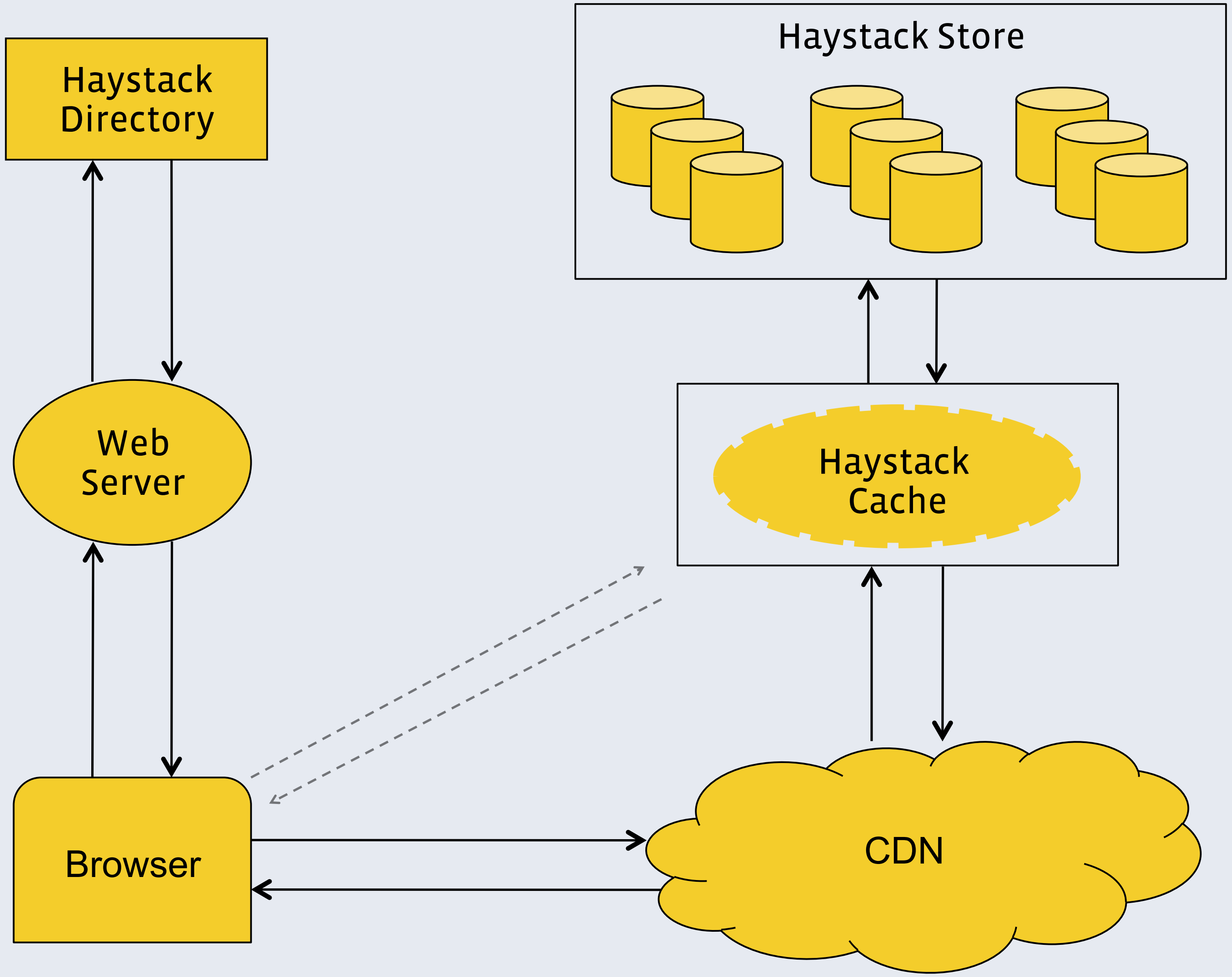
- 6 Billion messages/day
- 74 Billion operations/day
- At peak: 1.5 million operations/sec
- 55% read, 45% write operations
- Average write operation inserts 16 records
- All data is lzo compressed
- Growing at 8 TB/day

Haystack: The Photo Store

Facebook Photo DataStore

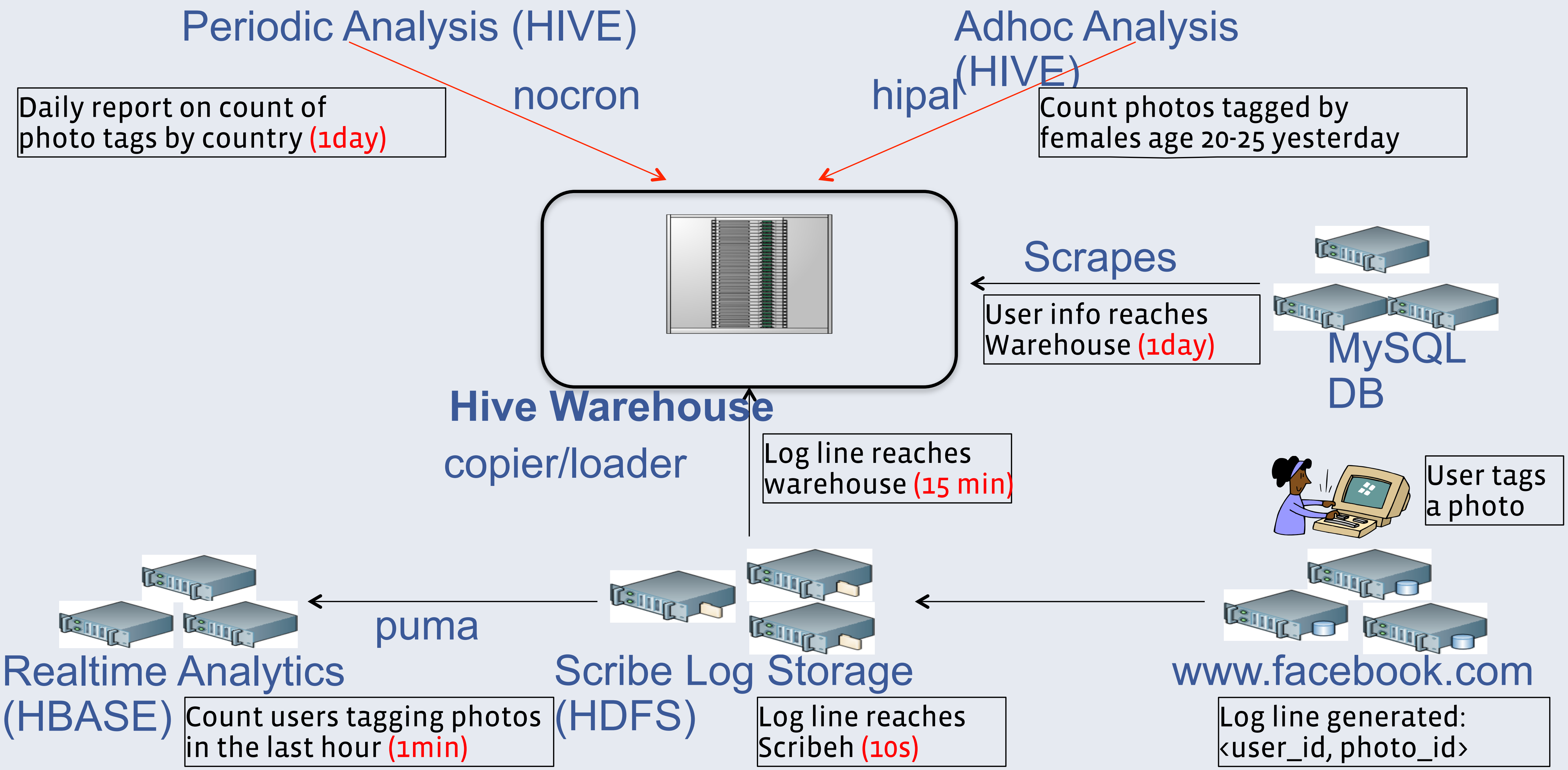
	2009	2012
Total Size	15 billion photos 1.5 Petabyte	hundred petabytes
Upload Rate	30 million photos/day 3 TB/day	300 million photos/day 30 TB/day
Serving Rate	555K images/sec	

Haystack based Design



Hive Analytics Warehouse

Life of a photo tag in Hadoop/Hive storage



Periodic Analysis (HIVE)

nocron

Daily report on count of photo tags by country (1day)

Adhoc Analysis (HIVE)

hipal

Count photos tagged by females age 20-25 yesterday

Scrapes

User info reaches Warehouse (1day)

MySQL DB

Hive Warehouse

copier/loader

Log line reaches warehouse (15 min)

User tags a photo

puma

Realtime Analytics (HBASE)

Count users tagging photos in the last hour (1min)

Scribe Log Storage (HDFS)

Log line reaches Scribeh (10s)

www.facebook.com

Log line generated: <user_id, photo_id>

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/

What is BigData? Prospecting for Gold..

- “Finding 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

How to measure performance

- Traditional database systems:
  - Latency of queries
- Big Data systems:
  - How much data can we store and query? (the '**Big**' in BigData)
  - How much data can we query in parallel?
  - What is the value of this system?

Measure Cost of Storage

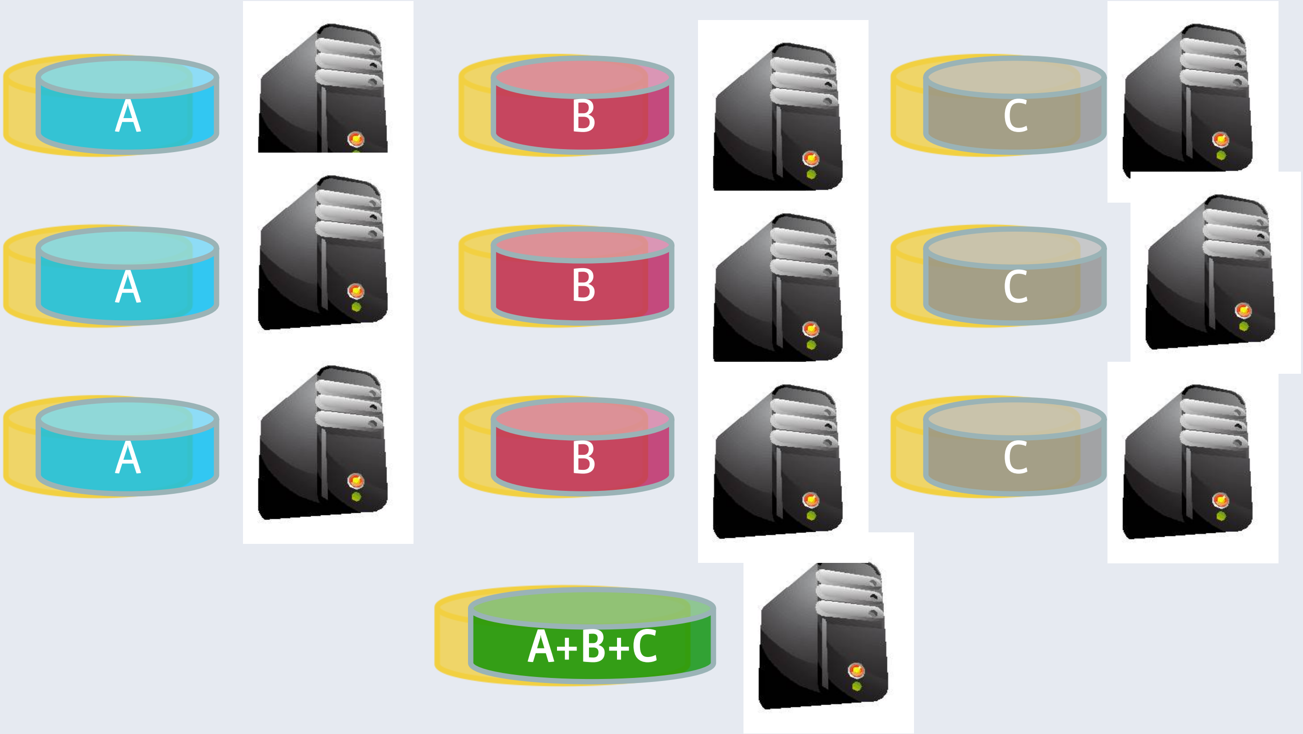
- **Distributed Network Encoding of data**
  - Encoding is better than replication
  - Use algorithms that minimize network transfer for data repair
- **Tradeoff cpu for storage & network**
  - Remember lineage of data, e.g. record query that created it
  - If data is not accessed for sometime, delete it
  - If a query occurs, recompute the data using query lineage

Measure Network Encoding

Start the same: triplicate every data block (storage overhead=3)

Background encoding

- Combine third replica of blocks from a single file to create parity block
- Remove third replica (storage overhead = 2)
- Reed Solomon encoding for much older files (storage overhead = 1.4)



A file with three blocks A, B and C (XOR Encoding)

http://hadoopblog.blogspot.com/2009/08/hdfs-and-erasure-codes-hdfs-raid.html

Measuring Data Discovery: Crowd Sourcing

- There are 50K tables in a single warehouse
- Users are Data Adminstrators themselves
- Questions about a table are directed to users of that table
- Automatic query lineage tools

Fault Tolerance and Elasticity

- Commodity machines
- Faults are the norm
- Anomalous behavior rather than complete failures
  - 10% of machines are always 50% slower than the others

Measuring Fault Tolerance and Elasticity

- **Fault tolerance is a must**
  - Continuously kill machines during benchmarking
  - Slow down 10% of machine during benchmark
- **Elasticity is necessary**
  - Add/remove new machines during benchmarking

Why use Hive instead of a Parallel DBMS?

- Stonebraker/DeWitt from the DBMS community:
  - Quote "Hadoop is a major step backwards"
  - Published benchmark results which show that Hadoop/Hive is not as performant as a traditional DBMS
  - http://database.cs.brown.edu/projects/mapreduce-vs-dbms/
  - Hive query is 50 times slower than DBMS query
- **Conclusion: Facebook's 4000 node cluster (100PB) can be replaced by a 20 node DBMS cluster**
- **What is wrong with the above conclusion?**

Hive/Hadoop instead of Parallel DBMS?

- Dr Stonebraker's proposal would put 5 PB per node on DBMS
  - What will be the io throughput of that system? **Abysmal**
  - How many concurrent queries can it support? **Certainly not 100K concurrent clients**
  - Query latency is not the only metric to make a conclusion
- Hive/Hadoop is very very slow
  - Hive/Hadoop needs to be fixed to reduce query latency
  - But an existing DBMS cannot replace Hive/Hadoop

Presto: A Distributed SQL Engine

- Low Latency, interactive usage
- Bypasses Map/Reduce
- Processes Hive/Hadoop data but has pluggable backends
- Will be open sourced soon
- Scale
    - 30K daily queries, 300 TB scanned daily
    - Growing fast

Future Challenges

New trends in storage software

- Analytics Data
  - Streaming queries, low latency queries
  - Cold Storage – very low $/GB
- OLTP Data
  - One size does not fit all: need specialized solutions
  - disk, flash, disk+flash
  - write heavy, point lookups, range scans
  - iops bound, storage bandwidth bound, memory bound

Questions?
dhruba@fb.com

http://hadoopblog.blogspot.com/