

facebook

Petabyte Scale Data at Facebook

Dhruba Borthakur,
Engineer at Facebook,
UC Berkeley, Nov 2012

Agenda

1 Types of Data

2 Data Model and API for Facebook Graph Data

3 SLTP (Semi-OLTP) and Analytics data

4 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

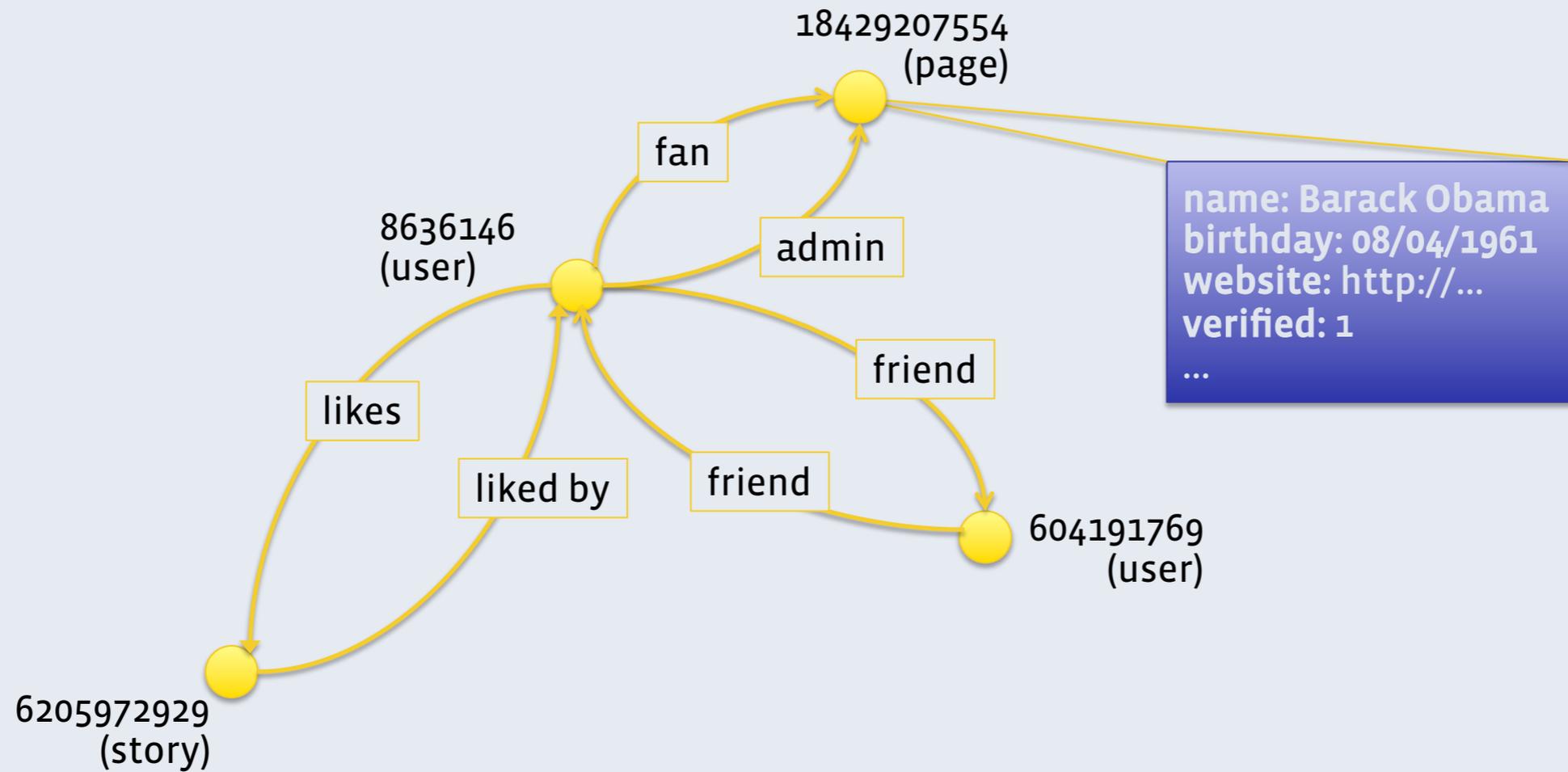
Characteristics

	Query Latency	Consistency	Durability
Facebook Graph	< few milliseconds	quickly consistent across data centers	No data loss
Facebook Messages and Time Series Data	< 200 millisecc	consistent within a data center	No data loss
Facebook Photos	< 250 millisecc	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 : $(id1, atype, id2) \rightarrow (time, (key \rightarrow value)^*)$
- `assoc_add(id1, atype, id2, time, (k->v)^*)` : adds/updates the given assoc
- `assoc_delete(id1, atype, id2)` : deletes the given association

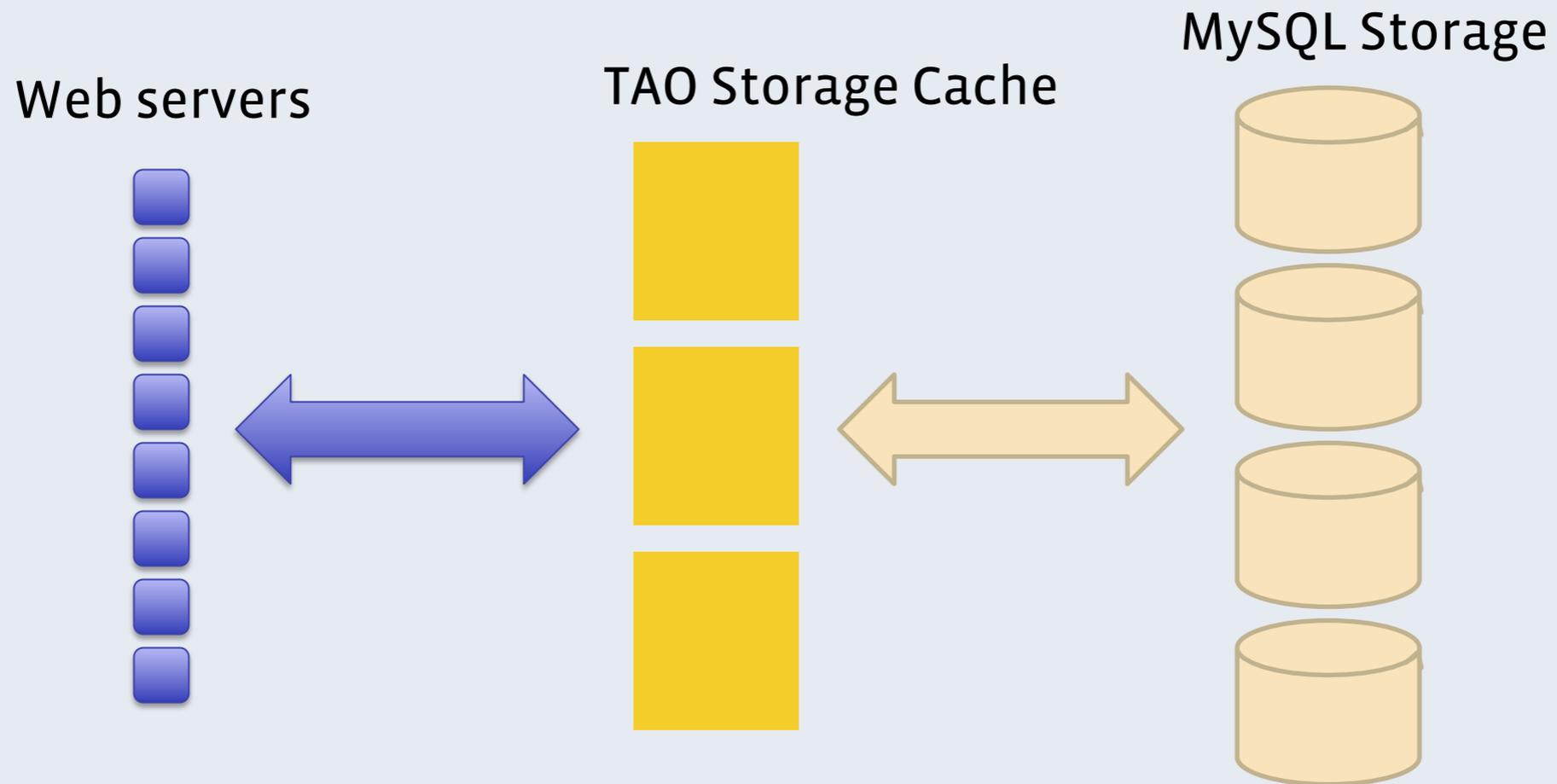
Data model

API

- Association : $(id1, atype, id2) \rightarrow (time, (key \rightarrow value)^*)$
 - `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

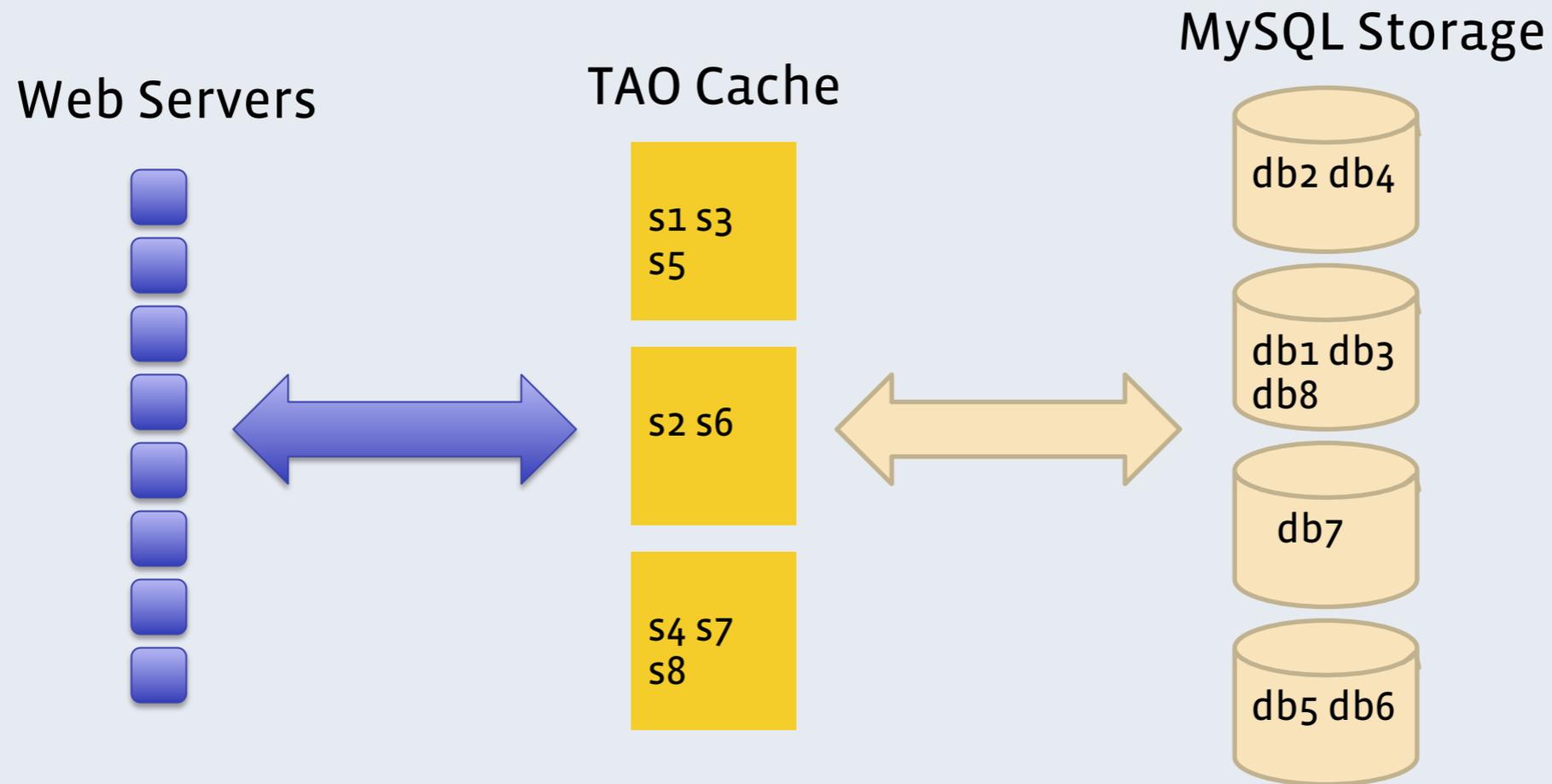
Cache & Storage



Architecture

Sharding

- Object ids and Assoc id1s are mapped to shard ids



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

Messages



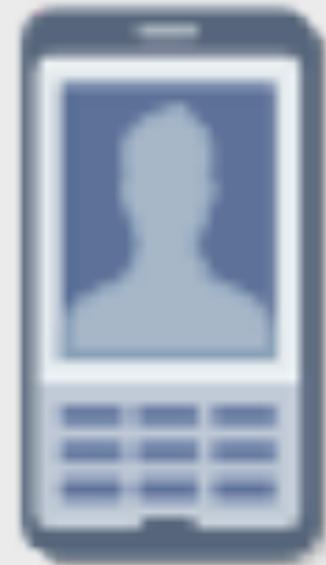
Chats



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
 - 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

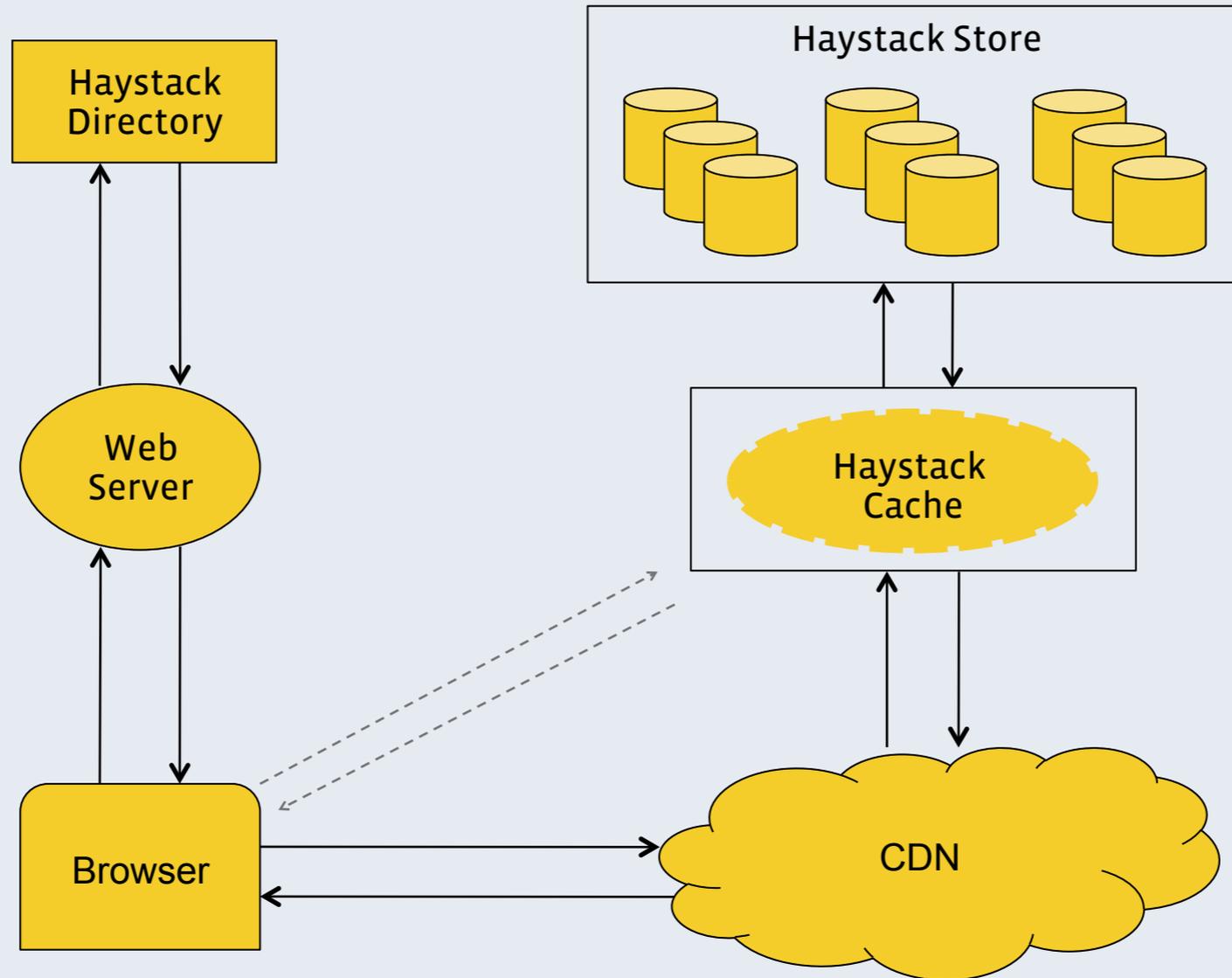
- 6 Billion messages/day
- 74 Billion operations/day
- At peak: 1.5 million operations/sec
- 55% read, 45% write operations
- Average write operation inserts 16 records
- All data is lzo compressed
- Growing at 8 TB/day

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

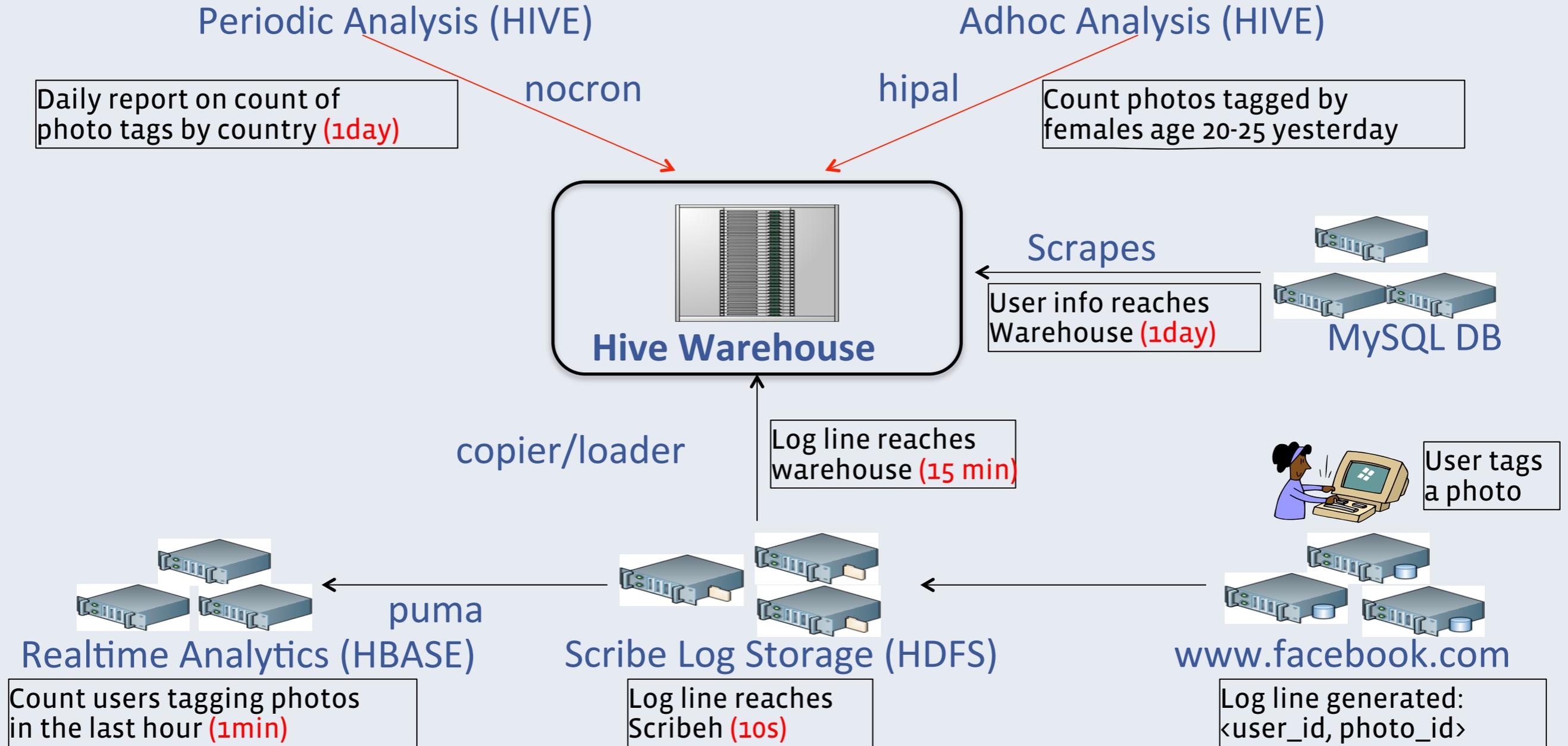


Haystack Internals

- 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/>

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

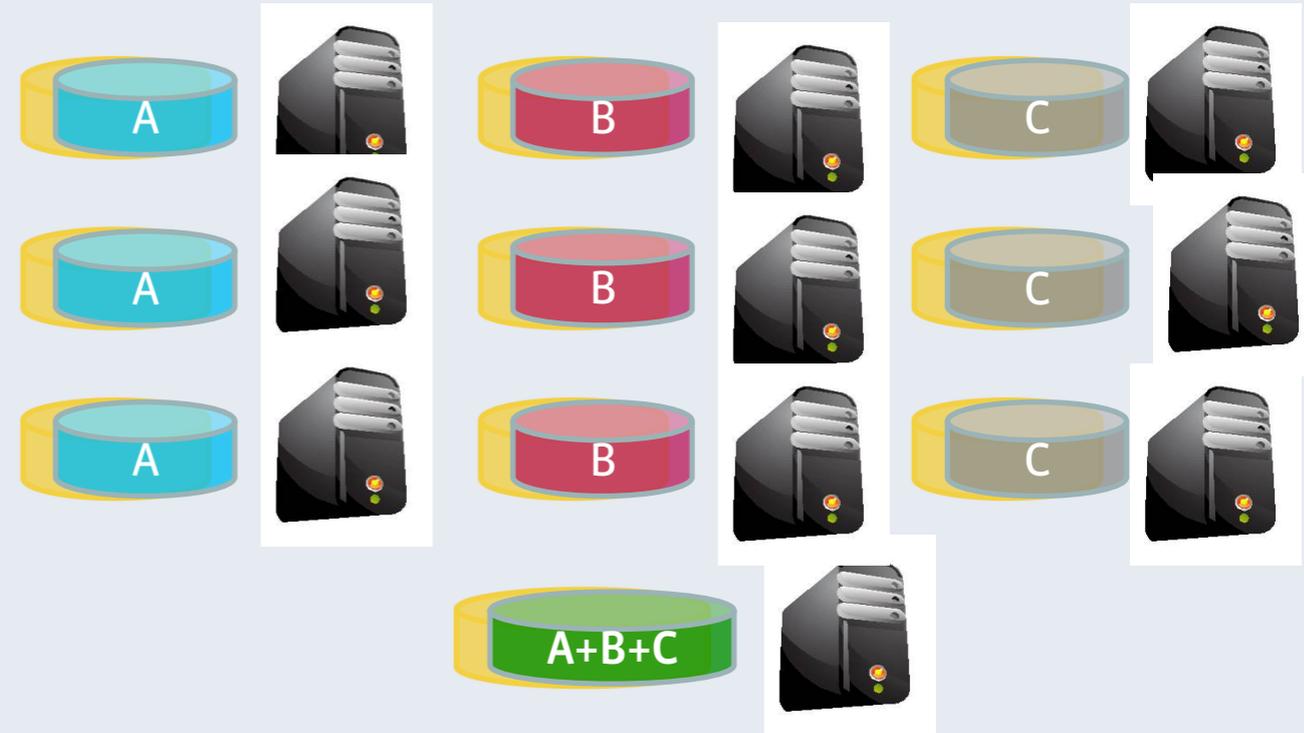


Measure Network Encoding

Start the same: triplicate every data block

Background encoding

- Combine third replica of blocks from a single file to create parity block
- Remove third replica
- Reed Solomon encoding for much older files



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 Administrators themselves
- Questions about a table are directed to users of that table
- Automatic query lineage tools



Measuring Testability

- **Traditional systems**

- Recreate load using tests
- Validate results

- **Big Data Systems**

- Cannot replicate production load on test environment
- Deploy new service on a small percentage of service
 - Monitor metrics
 - Rolling upgrades
- Gradually deploy to larger section of service



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



Measuring Value of the System

- cost /GB decreasing with time
 - So users can store more data
 - But users need a metric to determine whether this cost is worth it
- What is the **VALUE** of this system?
 - A metric that aids the user (and not the service provider)

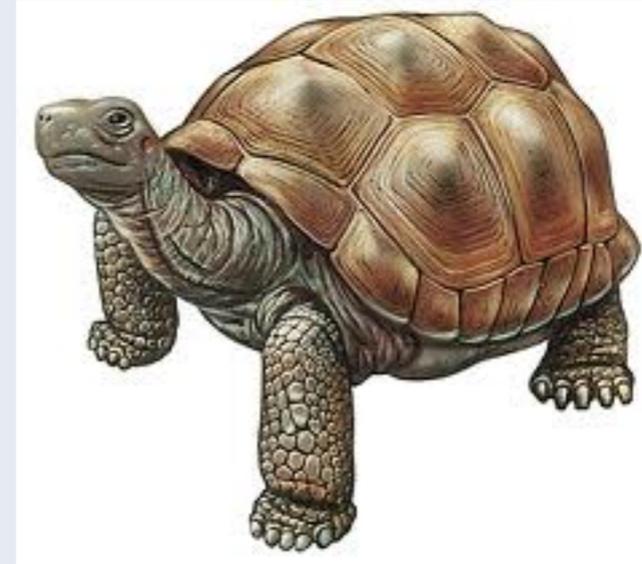
Value per Byte (VB) for the System

- A new metric named VB
 - Compare differences in value over time
 - If VB increases with time, then user is satisfied
- You touch a byte, its VB is MAX (say 100)
- System VB = weighted sum of VB of each byte in the system



VB – Even a turtle ages with time

- The VB decreases with time
 - A more recent access has more value than an older access
 - Different ageing models (linear, exponential)



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
- **Stonebraker’s 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**
 - He is using a wrong 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

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@gmail.com

<http://hadoopblog.blogspot.com/>