TAO
Facebook’s Distributed Data Store for the Social Graph

Nathan Bronson, Zach Amsden, George Cabrera, Prasad Chakka, Peter Dimov, Hui Ding, Jack Ferris, Anthony Giardullo, Sachin Kulkarni, Harry Li, Mark Marchukov, Dmitri Petrov, Lovro Puzar, Yee Jiun Song, Venkat Venkataramani

Presented at USENIX ATC – June 26, 2013
The Social Graph

Nathan
Yesterday at 5:30pm via iPhoto
At the summit — at Charlotte Dome.

Carol
how was it? is it worth the long approach?
Like · More · Just now

Alice
Like · More · Just now
Write a comment...

(hypothetical encoding)
Dynamically Rendering the Graph

Web Server (PHP)
Dynamically Rendering the Graph

• 1 billion queries/second
• many petabytes of data
What Are TAO’s Goals/Challenges?

- Efficiency at scale
Dynamic Resolution of Data Dependencies

- LOCATION: Charlotte Dome.
- COMMENT: how was it? is it worth the $!
  Like · More · Just now
- PHOTO
- USER: Carol
- USER: Nathan
- UPLOAD FROM
- APP: iPhoto

1. POST At the summit
2. 
3. 
What Are TAO’s Goals/Challenges?

▪ Efficiency at scale
▪ Low read latency
▪ Timeliness of writes
▪ High Read Availability
Graph in Memcache

Web Server (PHP)

Obj & Assoc API

memcache
(nodes, edges, edge lists)

mysql

Web Server (PHP) communication with Memcache and MySQL databases, utilizing an Object & Associate API.
Objects = Nodes

- Identified by unique 64-bit IDs
- Typed, with a schema for fields

id: 308 =>
  type: USER
  name: “Alice”

id: 1807 =>
  type: POST
  str: “At the summ…"

id: 2003 =>
  type: COMMENT
  str: “how was it …"

Associations = Edges

- Identified by <id1, type, id2>
- Bidirectional associations are two edges, same or different type

<1807, COMMENT, 2003>  
  time: 1,371,704,655

<308, AUTHORED, 2003>  
  time: 1,371,707,355

<2003, AUTHOR, 308>  
  time: 1,371,707,355
Association Lists

- `<id1, type, *>`
- Descending order by time
- Query sublist by position or time
- Query size of entire list

id: 1807 =>
type: POST
str: “At the summ…"

<1807,COMMENT,4141>
time: 1,371,709,009
str: “Been wanting to do …"

<1807,COMMENT,8332>
time: 1,371,708,678
str: “The rock is flawless, …"

<1807,COMMENT,2003>
time: 1,371,707,355
str: “how was it, was it w…"

id: 4141 => type: COMMENT
str: “Been wanting to do …"

id: 8332 => type: COMMENT
str: “The rock is flawless, …"

id: 2003 => type: COMMENT
str: “how was it, was it w…"
Objects and Associations API

**Reads – 99.8%**

- Point queries
  - `obj_get` 28.9%
  - `assoc_get` 15.7%
- Range queries
  - `assoc_range` 40.9%
  - `assoc_time_range` 2.8%
- Count queries
  - `assoc_count` 11.7%

** Writes – 0.2%**

- Create, update, delete for objects
  - `obj_add` 16.5%
  - `obj_update` 20.7%
  - `obj_del` 2.0%
- Set and delete for associations
  - `assoc_add` 52.5%
  - `assoc_del` 8.3%
What Are TAO’s Goals/Challenges?

- Efficiency at scale
- Low read latency
- Timeliness of writes
- High Read Availability
Independent Scaling by Separating Roles

- **Web servers**
  - Stateless
  - Sharded by id
  - Servers → read qps
  - Servers → bytes

- **Cache**
  - Objects
  - Assoc lists
  - Assoc counts

- **Database**

**TAO**
Subdividing the Data Center

- Web servers
  - Inefficient failure detection
  - Many switch traversals

- Cache
  - Many open sockets
  - Lots of hot spots

- Database
Subdividing the Data Center

Web servers

Cache

- Distributed write control logic

Database

- Thundering herds
What Are TAO’s Goals/Challenges?

▪ Efficiency at scale
▪ Low read latency
▪ Timeliness of writes
▪ High Read Availability
Write-through Caching – Association Lists

Web servers

Follower cache

Leader cache

Database

- Web servers
- Follower cache
- Leader cache
- Database

refill X

range get

X → Y

ok

Y, A, B, C

refill X

Y, A, B, C

X → Y

ok

ok

Y, A, B, C

X → Y

ok

Y, A, B, C

X → Y

ok
Asynchronous DB Replication

Web servers

Master data center

Writes forwarded to master

Inval and refill embedded in SQL

Replica data center

Delivery after DB replication done
What Are TAO’s Goals/Challenges?

- Efficiency at scale
- Low read latency
- Timeliness of writes
- High Read Availability
Improving Availability: Read Failover

- Web servers
- Follower cache
- Leader cache
- Database

Master data center

Replica data center
Efficiency at scale
  • Separate cache and DB
  • Graph-specific caching
  • Subdivide data centers

Read latency

Write timeliness
  • Write-through cache
  • Asynchronous replication

Read availability
  • Alternate data sources
Inverse associations

- Bidirectional relationships have separate $a \rightarrow b$ and $b \rightarrow a$ edges
  - $\text{inv}_\text{type}(\text{LIKES}) = \text{LIKED}_\text{BY}$
  - $\text{inv}_\text{type}(\text{FRIEND}_\text{OF}) = \text{FRIEND}_\text{OF}$
- Forward and inverse types linked only during write
  - TAO assoc_add will update both
- Not atomic, but failures are logged and repaired
Single-server Peak Observed Capacity

Hit rate

Operations/second
Write latency
More In the Paper

- The role of association time in optimizing cache hit rates
- Optimized graph-specific data structures
- Write failover
- Failure recovery
- Workload characterization