TAO: FACEBOOK’S DISTRIBUTED DATA STORE FOR THE SOCIAL GRAPH

Endrias Kahssay
MOTIVATION

- The Facebook graph has more than a billion active users, and trillions of edges.

- Because of complex privacy rules and tailoring of content to individual users, content is generated each time it's viewed resulting in a extremely high demand for reads.

- Clients are likely to query similar information as other clients.
• Facebook was built by storing the social graph on MySQL.

• As facebook scaled, it was necessary to reduce load on the database.

• Memcache, a key value cache system was scaled to work at the size of Facebook.
WHY TAO?

• Memcache which is a key-value system is inefficient at representing graphs.

• Queries must always fetch entire edge list, and change to one edge require edge list reload.

• Decentralized control logic results in more failures and thundering herds.
TAO

- A graph-aware caching framework that is geographically distributed over many thousands of machines.

- TAO provides low latency for users, fault tolerance, and eventual consistency.

- Shares many of its design with memcache.
API

- Object: (id) → (otype, (key value)*)
  - Represent Vertices

- Assoc.: (id1, atype, id2) → (time, (key value)*)
  - Represent edges
API

- Association List: (id1, atype) → [anew ... aold]
  - Ordered by timestamp.

- assoc add(id1, atype, id2, time, (k→v)*)
  - Add association from id1 to id2 of type atype, and its inverse if exists.

- assoc delete(id1, atype, id2)
  Delete association from id1 to id2
**API**

- `assoc_get(id1, atype, id2set, high?, low?)`

- `Assoc_count(id1, atype)`
  
  -- “How many checkins at the GG Bridge?” ⇒ `assoc count(534, CHECKIN)`

- `assoc_range(id1, atype, pos, limit)`
  
  -- “50 most recent comments on Alice’s checkin” ⇒ `assoc range(632, COMMENT, 0, 50)`
DATA STORAGE

- TAO’s API is mapped to SQL queries.

- Data is divided into shards, and each shard is contained in a logical database and load balancing among different hosts.

- Each object is associated with a unique shard that doesn’t change, and the association list is stored with the object.
CACHING LAYER

- TAO’s cache implements the complete API for clients, handling all communication with databases.

- A tier consists of multiple cache servers that serve different shards and can collectively serve any TAO request.

- A client issues a request to an appropriate cache, which will be responsible for completing reads and writes.
Caching Layer

- Cache misses and writes have to contact database.

- Write operations might involve other shards as inverse id could be on different machine, not guaranteed to be atomic.

- Cache understand semantics of queries.
ALL-TO-ALL CONNECTION

- Common for hundreds of objects and associations to be queried when rendering a Facebook page.

- Results in ALL-TO-ALL connection.

- In theory could scale with a single cache tier, but making shards more fine grained results in quadratic growth of connections and hot spots.
LEADERS AND FOLLOWERS

• Split the cache into two levels: a leader tier and multiple follower tiers.

• Leaders behave as described before, directly communicating with the storage layer.

• Followers forward miss and writes to a leader.

• Eventual Consistency; one leader per shard, and leader notifies followers that are out of date.
**Scaling Geographically**

- Followers can be thousands of miles of apart from leader.

- Master/slave architecture with slave DB in each region.

- Followers forward write / cache miss requests to local leaders, and leaders forward reads to local db, and writes to master leader.
Social graph is tightly connected, meaning each region would have to have a copy of the entire graph.

Expensive; instead have each region have some of the shards.

A full copy of the DB can be found across close regions.
Eventual consistency; clients can receive stale data; best effort read after write consistency.

TAO embeds invalidation and refill messages in the database replication stream.

If a forwarded write is successful then the local leaders involved (including inverse), will update their cache with the fresh value from a change set before returning.
CONSISTENCY

- MySQL remains a source of truth.

- Version numbers are used to avoid races in updating data.

- Slaves suffer from rare race condition related to evicting elements post change set that are not in local DB.
FAULT TOLERANCE

- Database fail if they crash or are too behind. If master database fails, one of the slaves becomes a master.

- When a leader fails, another leader for different shards takes over until the previous leader comes back.

- Invalidation failures: when a failed leader is replaced, all of the shards of the leader are invalidated.
RESULTS

• Facebook’s workload: 99.8% read requests, and 0.2% write requests.

• A single TAO deployment which processes billions of reads and millions of writes per sec.

• Results for randomly sampled 6.5 million requests over a 40 day period.
Figure 9: Write latency from clients in the same region as database masters, and from a region 58 msec away.

- Average write latency in same region, 12.1 msec, remote region 74.4 msec
## Read Performance

<table>
<thead>
<tr>
<th>operation</th>
<th>hit lat. (msec)</th>
<th>miss lat. (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50%</td>
<td>avg</td>
</tr>
<tr>
<td>assoc_count</td>
<td>1.1</td>
<td>2.5</td>
</tr>
<tr>
<td>assoc_get</td>
<td>1.0</td>
<td>2.4</td>
</tr>
<tr>
<td>assoc_range</td>
<td>1.1</td>
<td>2.3</td>
</tr>
<tr>
<td>assoc_time_range</td>
<td>1.3</td>
<td>3.2</td>
</tr>
<tr>
<td>obj_get</td>
<td>1.0</td>
<td>2.4</td>
</tr>
</tbody>
</table>
Replicas lag by 1 second 85% of the time, by utmost 3 secs 99% of the time, and 10 secs for 99.8%.

Local leader was unavailable 0.15% of the time, and slave databases were promoted 0.25% of the time.
CONCLUSION

• TAO is a graph-aware caching framework that is geographically distributed and has great performance.

• Eventual consistency.

• Future work on guaranteeing bounded consistency, noSQL database, fast writes, and getting rid of “rare” races that result in stale data.