Facebook Tao

NANDITA SATHE[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding author: nsathe@iu.edu

paper-2, April 30, 2017

Facebook Tao is a graph database currently serving Facebook Inc. to manage data of its billions of users. Graph database stores data in a graph structure and establishes relationships between data using nodes and edges. Graph databases are ideal for systems that require data to be represented as a graph or hierarchical structure and need to establish connections between data points. Storing relationships between data points as a first class entity helps to draw insights from the data. For example, real time recommendation engine is possible because of the ability of graph database to connect to the masses of buyers to product data, thereby enabling insights on customer needs and product trends. A graph database system like Tao is apt for relationship driven data requirements of Facebook. Main aim of Tao system is to achieve lowest read latency, timeliness in writes and efficiently scaling the data.

Keywords: Facebook Tao, Graph Database, memcache

https://github.com/cloudmesh/sp17-i524/raw/master/paper2/S17-IO-3017/report.pdf

1. INTRODUCTION

Graph databases like Facebook Tao are a response to data needs that traditional RDBMSs like MySQL do not meet. For example, Facebook realized however efficient relational database it would use, it is not sufficient to manage the enormous data challenge Facebook had. The data was a social graph. Another mismatch, which relational database or block cache had was, most of the data that would be read into cache did not belong to any relation. For example, 'If user likes that picture". In most records the answer would be 'No' or 'False'. Storing and reading this unwanted data was a burden. Meanwhile Facebook users' base was increasing daily. Ultimately Facebook came up with Facebook Tao, a distributed social graph data store.

Facebook TAO (The Association and Objects) is a geographically distributed data store that provides timely access to the social graph for Facebook's demanding workload using a fixed set of queries [1]. It is deployed at Facebook for many data types that fit its model. The system runs on thousands of machines, is widely distributed, and provides access to many petabytes of data. TAO represents social data items as Objects (user) and relationship between them as Associations (liked by, friend of). TAO cleanly separates the caching tiers from the persistent data store allowing each of them to be scaled independently. To any user of the system it presents a single unified API that makes the entire system appear like 1 giant graph database [2]. Key advantages of the system include [2]:

- Provides a clean separation of application/product logic from data access by providing a graph API and data model to store and fetch data.
- By implementing a write-through cache TAO allows Facebook to provide a better user experience and preserve the all important read-what-you-write consistency semantics even when the architecture spans multiple geographical regions.
- By implementing a read-through write-through cache TAO also protects the underlying persistent stores better by avoiding issues like thundering herds without compromising data consistency.

2. TAO'S GOAL

Main goal of implementing Tao is efficiently scaling the data. Facebook handles approximately a billion requests per second [3]. So obviously data store has to be scalable. More than that, scalability should be efficient otherwise scaling data across machines would be extremely costly.

Second goal is to achieve lowest possible read latency. So that if a user has commented on a post, the original post writer should be able to read it immediately. Efficiency in Scaling and low Read latency is achieved by (i) separating cache and data storage, (ii) Graph specific caching and (iii) Sub-dividing data centers [3].

Third goal is to achieve timeliness of writes. If a web server has written something and it sends a read request, it should be able to read the post. Write timeliness is achieved by (i) Write through cache and (ii) Asynchronous replication [3].

Lastly, the goal is also to have high read availability which is achieved by using alternate data sources.

3. TAO DATA MODEL AND API

Facebook Inc. explains TAO data model and API associated with using an example [4]. Figure 1 depicts TAO data model.

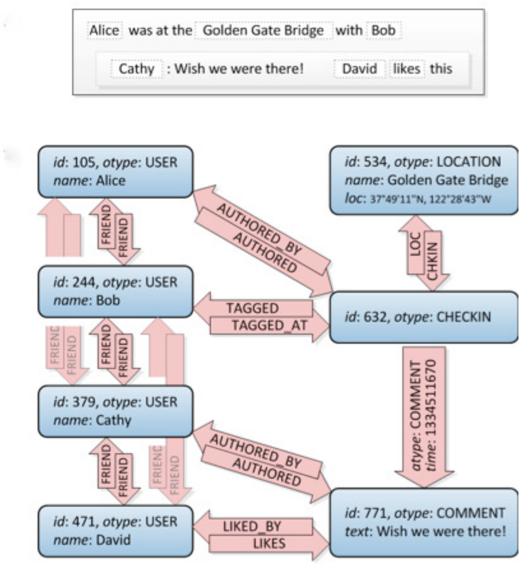

Fig. 1. TAO Data Model [4].

This example shows a subgraph of objects and associations that is created in TAO after Alice checks in at the Golden Gate Bridge and tags Bob there, while Cathy comments on the check-in and David likes it. Every data item, such as a user, check-in, or comment, is represented by a typed object containing a dictionary of named fields. Relationships between objects, such as “liked by" or “friend of," are represented by typed edges (associations) grouped in association lists by their origin. Multiple associations may connect the same pair of objects as long as the types of all those associations are distinct. Together objects and associations form a labeled directed multigraph.

For every association type a so-called inverse type can be specified. Whenever an edge of the direct type is created or deleted between objects with unique IDs *i*d1 and *i*d2, TAO will automatically create or delete an edge of the corresponding inverse type in the opposite direction (*i*d2 to *i*d1). The intent is to help the application programmer maintain referential integrity for relationships that are naturally mutual, like friendship, or where support for graph traversal in both directions is performance critical, as for example in “likes” and “liked by."

3.1. Objects and Associations

TAO objects are typed nodes, and TAO associations are typed directed edges between objects. Objects are identified by a 64-bit integer (id) that is unique across all objects, regardless of object type (otype). Associations are identified by the source object (*i*d1), association type (atype) and destination object (*i*d2). At most one association of a given type can exist between any two objects. Both objects and associations may contain data as key→value pairs. A per-type schema lists the possible keys, the value type, and a default value. Each association has a 32-bit time field, which plays a central role in queries [1].

Object: (id) -> (otype, (key -> value)⋆)

Assoc.: (id1, atype, id2) -> (time, (key -> value)⋆)

Figure ’Tao Data Model’ shows how TAO objects and associations might encode the example, with some data and times omitted for clarity. The example’s users are represented by objects, as are the checkin, the landmark, and Cathy’s comment. Associations capture the users’ friendships, authorship of the checkin and comment, and the binding between the checkin and its location and comments.

The set of operations on objects is of the fairly common create/set-fields/get/delete variety. All objects of a given type have the same set of fields. New fields can be registered for an object type at any time and existing fields can be marked deprecated by editing that type’s schema.

Associations are created and deleted as individual edges. If the association type has an inverse type defined, an inverse edge is created automatically. The API helps the data store exploit the creation-time locality of workload by requiring every association to have a special time attribute that is commonly used to represent the creation time of association. TAO uses the association time value to optimize the working set in cache and to improve hit rate [1].

4. TAO ARCHITECTURE

TAO is separated into layers: two caching layers and a storage layer.

4.1. Storage Layer

The data is persisted using MySQL. The API is mapped to a small number of SQL queries. TAO needs to handle a far larger volume of data than can be stored on a single MySQL server and so, the data is divided into logical shards. Each shard is contained in a logical database. Database servers are responsible for one or more shards. The number of shards far exceeds the number of servers. The shard to server mapping is tuned in to balance load across different hosts. By default all object types are stored in one table, and all association types in another. Every ’object-id’ has a corresponding ’shard-id’. Objects are bounded to a single shard throughout their lifetime. An association is stored on the shard of its *i*d1, so that every association query can be served from a single server. [3].

4.2. Caching Layer

TAO’s cache implements the API for clients, handling all communication with databases. A region/tier is made of multiple closely located Data centers. Multiple Cache Servers make up a tier (set of databases in a region are also called a tier) that can collectively capable of answering any TAO Request. Each cache request maps to a server based on sharding. Write operations on an association with an inverse may involve two shards, since the forward edge is stored on the shard for *i*d1 and the inverse edge is on the shard for *i*d2. Handling writes with multiple shards involve: Issuing an RPC (Remote Procedure Call) call to the member hosting *i*d2, which will contact the database to create the inverse association. Once the inverse write is complete, the caching server issues a write to the database for *i*d1. TAO does not provide atomicity between the two updates. If a failure occurs the forward may exist without an inverse, these hanging associations are scheduled for repair by an asynchronous job [3].

4.3. Leaders and Followers

There are two tiers of caching clusters in each geographical region. Clients talk to the first tier, called followers. If a cache miss occurs on the follower, the follower attempts to fill its cache from a second tier, called a leader. Leaders talk directly to a MySQL cluster in that region. All TAO writes go through followers to leaders. Caches are updated as the reply to a successful write propagates back down the chain of clusters. Leaders are responsible for maintaining cache consistency within a region. They also act as secondary caches [4].

4.4. Scaling Geographically

High read workload scales with total number of follower servers. The assumption is that latency between followers and leaders is low. Followers behave identically in all regions, forwarding read misses and writes to the local region's leader tier. Leaders query the local region's database regardless of whether it is the master or slave. This means that read latency is independent of inter-region latency. Writes are forwarded by the local leader to the leader that is in the region with the master database. Read misses by followers are 25X as frequent as writes in the workload thus read misses are served locally [3]. Facebook chooses data center locations that are clustered into only a few regions, where the intra-region latency is small (typically less than 1 millisecond) [3]. It is then sufficient to store one complete copy of the social graph per region.

Since each cache hosts multiple shards, a server may be both a master and a slave at the same time. It is preferred to locate all of the master databases in a single region. When an inverse association is mastered in a different region, TAO must traverse an extra inter-region link to forward the inverse write. TAO embeds invalidation and refill messages in the database replication stream. These messages are delivered in a region immediately after a transaction has been replicated to a slave database. Delivering such messages earlier would create cache inconsistencies, as reading from the local database would provide stale data. If a forwarded write is successful then the local leader will update its cache with the fresh value, even though the local slave database probably has not yet been updated by the asynchronous replication stream. In this case followers will receive two invalidates or refills from the write, one that is sent when the write succeeds and one that is sent when the write's transaction is replicated to the local slave database [3].

5. EDUCATIONAL MATERIAL

To get started on learning Facebook TAO, following resources can prove helpful.

- Technical paper on Facebook TAO [1].
- Background, Architecture and Implementation from Facebook itself [4].
- TAO summary in a video on USENIX website [5].

6. RELATED WORK

TAO is a geographically distributed, eventually consistent graph store optimized for reads, thus combining all three techniques into one system. Eventually consistency model is based on BASE (Basically Available, Soft state, Eventual consistency) semantics. These systems typically provide weaker guarantees than the traditional ACID (Atomicity, Consistency, Isolation, Durability) semantics. Google's BigTable, Yahoo!'s PNUTS, Amazon's SimpleDB, and Apache's HBase are examples of this more scalable approach. These systems all provide consistency and transactions at the per-record or row level similar to TAO's semantics for objects and associations, but do not provide TAO's read efficiency or graph semantics [1].

The Coda file system uses data replication to improve performance and availability in case of unreliable networks. Unlike Coda, TAO does not allow writes in portions of the system that are disconnected. Google Megastore is a storage system that uses Paxos (protocol for distributed consensus) across geographically distributed data centers to provide strong consistency guarantees and high availability. TAO provides no consistency guarantees but handles comparatively many more requests [1].

Since TAO was designed specifically to serve the social graph, its features are inherited from the graph databases. Neo4j is a open-source graph database that provides ACID semantics and the ability to shard data across several machines. Twitter uses its FlockDB to store parts of its social graph, as well [1].

Instead of using existing graph systems Facebook has customised Tao to fulfill its specific needs and workload requirements.

7. CONCLUSION

Overall, this paper explains characteristics and challenges of Facebook's workload, the objects and associations data model and lastly, it details out TAO, the geographically distributed system that implements the API to work with social graph. TAO is deployed at scale inside Facebook. Its separation of cache and persistent store has allowed those layers to be independently designed, scaled, and operated, and maximizes the reuse of components across the organization [1].

8. ACKNOWLEDGEMENTS

The author would like to thank Prof. Gregor von Laszewski and his associates from the School of Informatics and Computing for providing all the technical support and assistance.

REFERENCES

[1] N. Bronson *et al.*, "Tao: Facebook's distributed data store for the social graph," in *2013 USENIX Annual Technical Conference*, 2013. [Online]. Available: http://ai2-s2-pdfs.s3.amazonaws.com/39ac/2e0fc4ec63753306f99e71e0f38133e58ead.pdf

[2] V. Venkataramani, "What is the tao cache used for at facebook," Web Page, June 2013. [Online]. Available: https://www.quora.com/What-is-the-TAO-cache-used-for-at-Facebook

[3] N. Upreti, "Facebook's tao and unicorn data storage and search platforms," Slides, April 2015. [Online]. Available: https://www.slideshare.net/nitishupreti/faceboko-tao-unicorn

[4] M. Marchukov, "Tao: The power of the graph," Web Page, June 2013. [Online]. Available: https://www.facebook.com/notes/facebook-engineering/tao-the-power-of-the-graph/10151525983993920/

[5] N. Bronson, "Tao: Facebook's distributed data store for the social graph," Slides, June 2013. [Online]. Available: https://www.usenix.org/node/174510