facebook research

October 22, 2014

Facebook's Top Open Data Problems

By: Janet Wiener, **Nathan Bronson**

Facebook's Top Open Data Problems

Facebook hosted a data faculty summit on September 16, 2014. Top database faculty from around the country joined Facebook researchers at their headquarters in Menlo Park, California, to discuss the key open challenges around data storage and access.

The focus was on three broad topics: small data, big data, and hardware trends. Small data refers to OLTP-like queries that process and retrieve a small amount of data, usually 1-1000 objects requested by their id. Indexes limit the amount of data accessed during a single query, regardless of the total volume of data. Big data refers to queries that process large amounts of data, usually for analysis: trouble-shooting, identifying trends, and making decisions. The total volume of data is massive for both small and big data, ranging from hundreds of terabytes in memory to hundreds of petabytes on disk.

Small data challenges

Facebook's News Feed requires 1000s of objects to render and it is personalized for everyone on the service. The requirements for these object fetches (multiplied by 1.3 billion people) lead to many challenges in designing and distributing the underlying caches and databases. Facebook's social graph store TAO, for example, provides access to tens of petabytes of data, but answers most queries by checking a single page in a single machine.

TAO currently sustains billions of queries per second. Like traditional relational database updates, TAO updates must be durable and immediately visible. However, there are no joins or large scans in TAO. For most data types, we favor read and write availability over atomicity or consistency,

TAO and Memcache are the primary systems that handle caching our small data workload.

Nathan Bronson presented an overview of our current small data infrastructure and led a discussion of three challenges. First, we talked about how to modify Facebook's data infrastructure to optimize for mobile devices, which have intermittent connectivity and higher network latencies than most web clients. Next, we discussed how to reduce the database footprint in data centers without increasing user-visible latency or reducing system availability. Finally, we explored the efficiency/available tradeoffs of RAM caches.

Big data challenges

Big data stores are the workhorses for data analysis at Facebook. They grow by millions of events (inserts) per second and process tens of petabytes and hundreds of thousands of queries per day. The three data stores used most heavily are:

1. ODS (Operational Data Store) stores 2 billion time series of counters. It is used most commonly in alerts and dashboards and for trouble-shooting system metrics with 1-5 minutes of time lag. There are about 40,000 queries per second.

2. Scuba is Facebook's fast slice-and-dice data store. It stores thousands of tables in about 100 terabytes in memory. It ingests millions of new rows per second and deletes just as many. Throughput peaks around 100 queries per second, scanning 100 billion rows per second, with most response times under 1 second.

3. Hive is Facebook's data warehouse, with 300 petabytes of data in 800,000 tables. Facebook generates 4 new petabyes of data and runs 600,000 queries and 1 million map-reduce jobs per day. Presto, HiveQL, Hadoop, and Giraph are the common query engines over Hive.

ODS, Scuba, and Hive share an important characteristic: none is a traditional relational database. They process data for analysis, not to serve users, so they do not need ACID guarantees for data

Janet Wiener led the discussion of current big data challenges at Facebook, including anomaly detection, sampling accuracy, resource scheduling, and quantities of data too big for a single data center.

New hardware for data management

New hardware promises better performance at a lower cost. Data servers must adapt to get many of the benefits, but it takes years to build them and make them robust. Mark Callaghan and Vijay Rao described some of the new hardware changes at Facebook and led a discussion on new software adaptations.

Top Open Data Problems

Here's an overview of the top open data problems covered during the summit amidst lively discussion.

Small data open challenges:

Mobile – How should the seismic shift toward mobile devices affect Facebook's data infrastructure? Our current system is mostly pull-based; data fetching, privacy checks, and content generation happen as late as possible. This works well for the web, since it minimizes wasted work and makes it easy to incorporate the latest data in the results. It's more challenging for mobile devices, though, as they don't have continuous good network connectivity.Prefetching and data pushing can keep relevant content available even if the network is flaky, but they may also decrease efficiency. If someone never reads a specific story in News Feed, the effort to prepare it and transmit it to their device is wasted. Another concern is the interaction between early rendering and our privacy rules. To build such a system, we must be able to efficiently determine whether any subsequent changes to the social graph have affected the visibility of prefetched data.

Most of the discussion around this challenge focused on how to capture the dependencies of privacy rules in an analyzable format. This is a surprisingly difficult problem. There is a long tail of tricky corner cases, legacy privacy rule types, and dynamic checks. A specific tricky example we

accurate analysis (either static or dynamic) difficult.

Reducing Replication – Facebook's application code and data infrastructure were written with the assumption that queries are fast. Our data centers contain regional database replicas so that even in the case of a cache miss, we can fetch the nodes and edges of the social graph quickly. Our data and query abstractions in PHP currently rely on the ability to make many round-trips to the data layer — sometimes several dozens of them to render a single endpoint. This execution model is fundamentally different from a typical SQL query, where a single complex query can process interlinked data from many tables.Small data centers with a complete database replica are not hardware efficient. Each replica must support a large set of application servers, caches, and other backend services. Therefore, there is a large lower bound on the size of data centers. Few potential sites have enough good electrical power to meet that bound.

We would like to be able to serve the Facebook application from a data center that is not geographically close to a complete replica. There are two general strategies: ensure that enough data is present in the local data center to limit cross-data-center accesses; or reduce the number of round trips between the (local) PHP and the (remote) data layer.

Two primary difficulties exist with holding only a subset of data locally: outliers and negative information. A partitioning or caching scheme that worked well on average might still result in unacceptable performance for some of our use cases, which would prevent us from deploying it. Also, one of the challenging features of our workload is that it contains many queries for which no edges match. Negative results are challenging to cache, since they are harder to associate with versions or other conflict resolution information.

Reducing the number of round trips between the application and data layers might also be done with a more powerful query language. Our discussion of this idea ended up circling back to the problem of analyzing privacy rules, because our fine-grained privacy checks are applied to all of the intermediate results of a query as well as to the final result.

The Impact of Caches on Availability – Facebook's read-heavy workload relies on caches for efficiency. Data centers are subdivided into clusters, each of which has its own caches. This

scenarios. If we try to capitalize on the efficiency gains presented by a high cache hit rate, we risk a capacity crunch when something goes wrong.We had an interesting discussion on the difference between capacity misses and cold misses, and on how we could denormalize data to encode negative information into the data that was present. There were also some incomplete ideas brought up about the possibilities of upcoming non-volatile memories to allow a data store that provides extremely high read capacity without a large in-memory cache.

Big data open challenges:

Sampling at logging time in a distributed environment – Far too many events happen on Facebook's web tier to log them all for data analysis. Instead, the events are sampled at logging time, usually with a random function mod N, for some sample rate N. Each web server makes its own decisions. (There is no time or space for coordination.) Although the sample rates are also logged, what is the accuracy of query results over sampled data? The entirety of the data is not available for comparison. What techniques can we apply to estimate accuracy? Do they work if the query contains very selective predicates?

Trading storage space and CPU – ODS, Scuba, and Hive all use standard compression techniques to save space, often a combination of dictionary compression, bit packing, and delta encoding, with a final pass by gzip or lz4. This combination reduces space requirements by a factor of 10 or more, but adds CPU overhead at both ingestion and query time. Ingestion overhead is significant; approximately half of all Scuba data is replaced every two weeks. Which combinations of compression methods are best?What about replication? Is it better to store three copies or use Reed-Solomon encoding, which is only about 2.2 copies. More copies reduce hot spots and make it easier to recover from the loss of a data center — but the storage cost is high. Can the number of copies be chosen dynamically based on usage?

Reliability of pipelines – Not only are these data stores themselves complex distributed systems, they also rely on other systems to deliver the data, to perform transformations, and to load the data. Furthermore, they have dependencies on the operating system, the file system, the deployment software, and a bunch of other lower-level systems. A failure in any of these systems results in missing data or failed queries. Finally, many queries are themselves part of a scheduled pipeline of queries to deliver a result table, including over half of all Hive queries. Even if every

we can take from the map-reduce framework without introducing too much overhead?

Globally distributed warehouse – While the small data stores are replicated globally, Facebook's data warehouse cannot be stored in any single data center, so it's partitioned globally. However, the data for any one query must be in a single data center. Today, the query writer needs to know about the data centers and copy data manually before writing a query. This process leads to lots of local decisions about replication and lots of copies of important data. Can we remove the burden from the query writer and make better decisions? Can the query optimizer choose where to replicate data and where to run the query?Resource scheduling is also complex in the globally distributed warehouse. Thousands of users, thousands of machines, and hundreds of thousands of queries means lots of scheduling decisions. But over 60% of the queries are in daily pipelines with dependencies. How should these queries be prioritized? They have different runtimes and SLAs. How do we handle varying network (sometimes cross-globe) and hardware constraints?

Time series correlation and anomaly detection – Time series generally contain patterns. A deviation from the pattern, which might be daily or weekly, often indicates a problem. Many alerts look for deviations against a fixed threshold or against a fixed percentage change from the previous time.Anomaly detection is about spotting changes without specifying a threshold. Important criteria for anomaly detection at Facebook include: False negatives are OK. False positives are not. They wake you up at 2:00 am! Detectors must handle baseline changes, such as a big jump when a service is turned up. Time series with missing points are important; some actions will be rare, e.g., because some Android device is rare.

Once a problem is spotted, other time series with correlated changes may yield the root cause, or at least a step along the way toward finding it. These algorithms must be fast: under 5 seconds on over 2 billion time series. However, we can limit the initial set of time series to those with similar names, i.e., 10,000 or so. For both anomaly detection and time series correlation, which algorithms should we use? Can we apply them to compressed data?

Data Management Hardware Open Challenges

with the trends.

Facebook's server designs are published as part of the Open Compute Project. Each storage technology is characterized by its read and write performance, its optimal access patterns, its available form factors, its write endurance, and its cost. Which technology is best for the kinds of small data and big data access patterns described above?

For example, people on Facebook can access all of their data at any time, but there is a very large variation in access frequency. A comment from three years ago is much less likely to be needed than a comment from today. Given these differences in access patterns, what storage optimizations can we apply?

At Facebook's scale, we are most concerned with the ultimate efficiency of a heterogeneous storage stack. Should we consider row-level data placement? Can we exploit application-level knowledge to optimize placement? How many levels of the storage hierarchy need to be searched before determining that a query has an empty result?

The discussion of future hardware trends centered on ways to leverage the incredible performance of non-volatile memories. There are several interesting decisions here: At what granularity should we partition data? How do we choose the storage destination when writing? Reading? Who unifies the heterogeneity?

————

We're excited about the opportunity to collaborate with academia on these challenges and hope that by sharing this list broadly, it will stimulate more interest in solving these problems and create more opportunities to work together in defining the next generation of databases.

Areas: Data Science Systems & Networking

facebook research

Related

Publication

The Effect of Computer-Generated Descriptions on Photo-Sharing Experiences of People with Visual Impairments

Yuhang Zhao, Shaomei Wu, Lindsay Reynolds, Shiri Azenkot

CSCW 2018

November 3, 2018

Publication

Neural Compositional Denotational Semantics for Question Answering

Nitish Gupta, Mike Lewis

EMNLP 2018

October 31, 2018

Publication

The effects of natural scene statistics on text readability in additive displays

Daryn R. Blanc-Goldhammer, Kevin J. MacKenzie

HFES 2018

October 1, 2018

Publication

Low-Dose-Rate Cobalt-60 Testing Results for Kaman KD-5100 Differential Inductive Position Measuring Systems

Bart McGuyer, Randall Milanowski, Slaven Moro

JOURNAL, IEEE RADIATION EFFECTS DATA WORKSHOP (REDW)

October 1, 2018

facebook research

Like Page 188K likes

RSS Feed About

Careers Privacy

Cookies Terms

Help