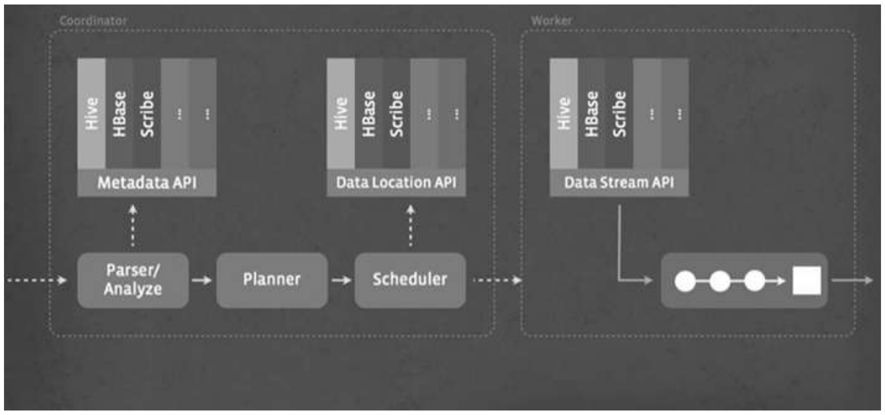
POSTED ON NOVEMBER 6, 2013 TO CORE DATA

Presto: Interacting with petabytes of data at Facebook



By Martin Traverso

Background

Facebook is a data-driven company. Data processing and analytics are at the heart of building and delivering products for the 1 billion+active users of Facebook. We have one of the largest data warehouses in the world, storing more than 300 petabytes. The data is used for a wide range of applications, from traditional batch processing to graph analytics, machine learning, and real-time interactive analytics.

For the analysts, data scientists, and engineers who crunch data, derive insights, and work to continuously improve our products, the performance of queries against our data warehouse is important. Being able to run more queries and get results faster improves their productivity.

Facebook's warehouse data is stored in a few large Hadoop/HDFS-based clusters. Hadoop MapReduce and Hive are designed for large-scale, reliable computation, and are optimized for overall system throughput. But as our warehouse grew to petabyte scale and our needs evolved, it became clear that we needed an interactive system optimized for low query latency.

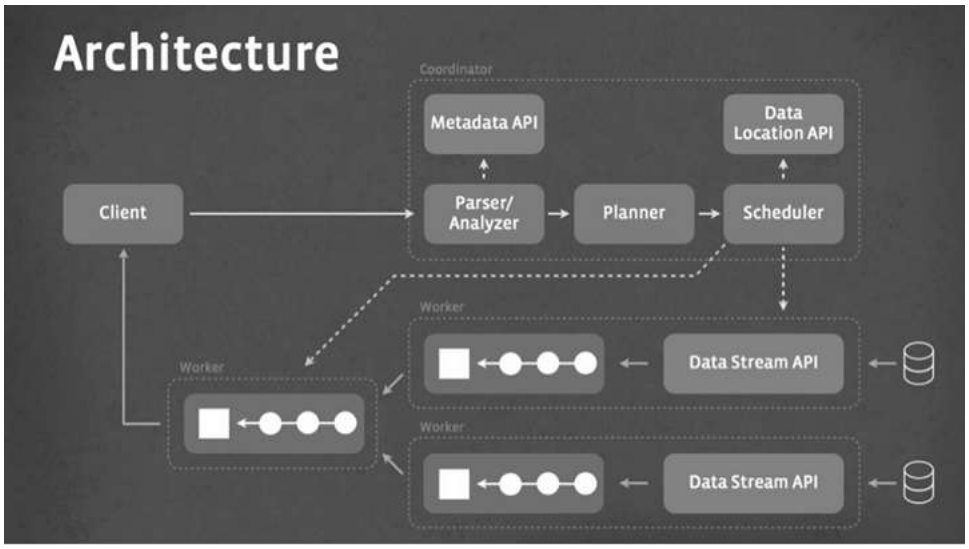
In Fall 2012, a small team in the Facebook Data Infrastructure group set out to solve this problem for our warehouse users. We evaluated a few external projects, but they were either too nascent or did not meet our requirements for flexibility and scale. So we decided to build Presto, a new interactive query system that could operate fast at petabyte scale.

In this post, we will briefly describe the architecture of Presto, its current status, and future roadmap.

Architecture

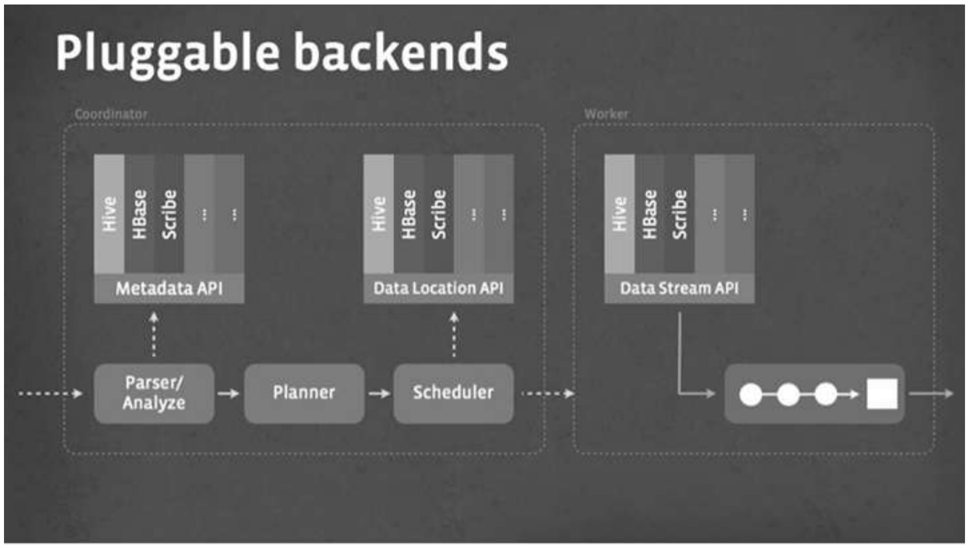
The diagram below shows the simplified system architecture of Presto. The client sends SQL to the Presto coordinator. The coordinator parses, analyzes, and plans the query execution. The scheduler wires together the execution pipeline, assigns work to nodes closest to the data, and monitors progress. The client pulls data from output stage, which in turn pulls data from underlying stages.

The execution model of Presto is fundamentally different from Hive/MapReduce. Hive translates queries into multiple stages of MapReduce tasks that execute one after another. Each task reads inputs from disk and writes intermediate output back to disk. In contrast, the Presto engine does not use MapReduce. It employs a custom query and execution engine with operators designed to support SQL semantics. In addition to improved scheduling, all processing is in memory and pipelined across the network between stages. This avoids unnecessary I/O and associated latency overhead. The pipelined execution model runs multiple stages at once, and streams data from one stage to the next as it becomes available. This significantly reduces end-to-end latency for many types of queries.



The Presto system is implemented in Java because it’s fast to develop, has a great ecosystem, and is easy to integrate with the rest of the data infrastructure components at Facebook that are primarily built in Java. Presto dynamically compiles certain portions of the query plan down to byte code which lets the JVM optimize and generate native machine code. Through careful use of memory and data structures, Presto avoids typical issues of Java code related to memory allocation and garbage collection. (In a later post, we will share some tips and tricks for writing high-performance Java system code and the lessons learned while building Presto.)

Extensibility is another key design point for Presto. During the initial phase of the project, we realized that large data sets were being stored in many other systems in addition to HDFS. Some data stores are well-known systems such as HBase, but others are custom systems such as the Facebook News Feed backend. Presto was designed with a simple storage abstraction that makes it easy to provide SQL query capability against these disparate data sources. Storage plugins (called connectors) only need to provide interfaces for fetching metadata, getting data locations, and accessing the data itself. In addition to the primary Hive/HDFS backend, we have built Presto connectors to several other systems, including HBase, Scribe, and other custom systems.



Current status

As mentioned above, development on Presto started in Fall 2012. We had our first production system up and running in early 2013. It was fully rolled out to the entire company by Spring 2013. Since then, Presto has become a major interactive system for the company's data warehouse. It is deployed in multiple geographical regions and we have successfully scaled a single cluster to 1,000 nodes. The system is actively used by over a thousand employees, who run more than 30,000 queries processing one petabyte daily.

Presto is 10x better than Hive/MapReduce in terms of CPU efficiency and latency for most queries at Facebook. It currently supports a large subset of ANSI SQL, including joins, left/right outer joins, subqueries, and most of the common aggregate and scalar functions, including approximate distinct counts (using HyperLogLog) and approximate percentiles (based on quantile digest). The main restrictions at this stage are a size limitation on the join tables and cardinality of unique keys/groups. The system also lacks the ability to write output data back to tables (currently query results are streamed to the client).

Roadmap

We are actively working on extending Presto functionality and improving performance. In the next few months, we will remove restrictions on join and aggregation sizes and introduce the ability to write output tables. We are also working on a query "accelerator" by designing a new data format that is optimized for query processing and avoids unnecessary transformations. This feature will allow hot subsets of data to be cached from backend data store, and the system will transparently use cached data to "accelerate" queries. We are also working on a high performance HBase connector.

Open source

After our initial Presto announcement at the Analytics @ WebScale conference in June 2013, there has been a lot of interest from the external community. In the last couple of months, we have released Presto code and binaries to a small number of external companies. They have successfully deployed and tested it within their environments and given us great feedback.

Today we are very happy to announce that we are open-sourcing Presto. You can check out the code and documentation on the site below. We look forward to hearing about your use

Presto on GitHub

The Presto team within Facebook Data Infrastructure consists of Martin Traverso, Dain Sundstrom, David Phillips, Eric Hwang, Nileema Shingte and Ravi Murthy.

TAGS: BACKEND INFRA PERFORMANCE

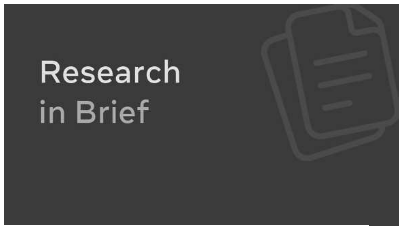
◂ **Prev**
Under the Hood: Building posts search

Next ▸
The Mature Optimization Handbook

Read More in Core Data

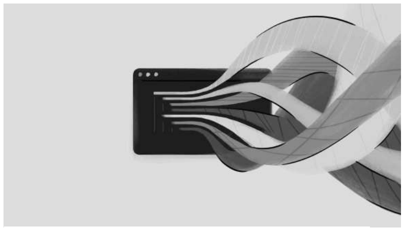
View All ▸

MAR 7, 2022
Augmenting Flexible Paxos in LogDevice to improve read availability



OCT 26, 2021
Kangaroo: A new flash cache optimized for tiny objects

SEP 13, 2021
Superpack: Pushing the limits of compression in Facebook's mobile apps



SEP 2, 2021
CacheLib, Facebook's open source caching engine for web-scale services

AUG 18, 2021
RAMP-TAO: Layering atomic transactions on Facebook's online graph store

AUG 6, 2021
How we built a general purpose key value store for Facebook with ZippyDB

Related Posts

Related Positions

REMOTE, US

Data Scientist, Product Analytics
REMOTE, NETHERLANDS

Data Scientist, Product Analytics
REMOTE, ITALY

Data Scientist, Product Analytics
REMOTE, FRANCE

See All Jobs

Available Positions

Business Analyst
MENLO PARK, US
Business Analyst
REMOTE, US
Data Scientist, Product Analytics
REMOTE, NETHERLANDS
Data Scientist, Product Analytics
REMOTE, ITALY
Data Scientist, Product Analytics
REMOTE, FRANCE

See All Jobs

Stay Connected

Engineering at Meta
Like

Meta Open Source
Follow

Meta Research
Like

Meta for Developers
Like

RSS
Subscribe

Open Source

Meta believes in building community through open source technology. Explore our latest projects in Artificial Intelligence, Data Infrastructure, Development Tools, Front End, Languages, Platforms, Security, Virtual Reality, and more.

ANDROID
iOS
WEB
BACKEND
HARDWARE

Learn More

Meta

Engineering at Meta is a technical news resource for engineers interested in how we solve large-scale technical challenges at Meta.

Home
Company Info
Careers