Apache Phoenix Joins Cloudera Labs

by Srikanth Srungarapu May 06, 2015 **no comments**

We are happy to announce the inclusion of Apache Phoenix in Cloudera Labs.

Apache Phoenix is an efficient SQL skin for Apache HBase that has created a lot of buzz. **Many companies** are successfully using this technology, including Salesforce.com, where Phoenix first started.

With the news that Apache Phoenix integration with Cloudera's platform has joined **Cloudera Labs**, let's take a closer look at a few key questions surrounding Phoenix: What does it do? Why does anyone want to use it? How does it compare to existing solutions? Do the benefits justify replacing existing systems and infrastructure?

In this post, we'll try to answers those questions by briefly introducing Phoenix and then discussing some of its unique features. I'll also cover some use cases and compare Phoenix to existing solutions.

What is Apache Phoenix?

Phoenix adds SQL to HBase, the distributed, scalable, big data store built on Hadoop. Phoenix aims to ease HBase access by supporting SQL syntax and allowing inputs and outputs using standard JDBC APIs instead of HBase's Java client APIs. It lets you perform all CRUD and DDL operations such as creating tables, inserting data, and querying data. SQL and JDBC reduce the amount of code users need to write, allow for performance optimizations that are transparent to the user, and opens the door to leverage and integrate lots of existing tooling.

Internally, Phoenix takes your SQL query, compiles it into a series of native HBase API calls, and pushes as much work as possible onto the cluster for parallel execution. It automatically creates a metadata repository that provides typed access to data stored in HBase tables. Phoenix's direct use of the HBase API, along with coprocessors and custom filters, results in performance on the order of milliseconds for small queries, or seconds for tens of millions of rows.

Use Cases

Phoenix is good for fast HBase lookups. Its secondary indexing feature supports many SQL constructs and make lookup via non-primary key fields more efficient than full table scans. It simplifies the creation and management of typed row-centric data by providing composite row keys and by enforcing constraints on data when written using the Phoenix interfaces.

Phoenix provides a way to transparently salt the row key, which helps in avoiding the RegionServer **hotspotting** often caused by monotonically increasing rowkeys. It also provides multi-tenancy via a combination of multi-tenant tables and tenant-specific connections. With tenant-specific connections, tenants can only access data that belongs to them, and with multi-tenant tables, they can only see their own data in those tables and all data in regular tables.

Regardless of these helpful features, Phoenix is not a drop-in RDBMS replacement. There are some limitations:

- Phoenix doesn't support cross-row transactions yet.
- Its query optimizer and join mechanisms are less sophisticated than most COTS DBMSs.
- As secondary indexes are implemented using a separate index table, they can get out of sync with the primary table (although perhaps only for very short periods.) These indexes are therefore not fully-ACID compliant.
- Multi-tenancy is constrained—internally, Phoenix uses a single HBase table.

Comparisons to Hive and Impala

The other well known SQL alternatives to Phoenix on top of HBase are Apache Hive and Impala. There is significant overlap in the functionality provided by these products. For example, all of them follow SQL-like syntax and provide a JDBC driver.

Unlike Impala and Hive, however, Phoenix is intended to operate exclusively on HBase data; its design and implementation are heavily customized to leverage HBase features including coprocessors and skip scans.

Some other considerations include:

- The main goal of Phoenix is to provide a high-performance relational database layer over HBase for low-latency applications. Impala's primary focus is to enable interactive exploration of large data sets by providing high-performance, low-latency SQL queries on data stored in popular Hadoop file formats. Hive is mainly concerned with providing data warehouse infrastructure, especially for long-running batch-oriented tasks.
- Phoenix is a good choice, for example, in CRUD applications where you need the scalability of HBase along with the facility of SQL access. In contrast, Impala is a better option for strictly analytic workloads and Hive is well suited for batch-oriented tasks like ETL.
- Phoenix is comparatively lightweight since it doesn't need an additional server.
- Phoenix supports advanced functionality like multiple secondary-index implementations optimized for different workloads, flashback queries, and so on. Neither Impala nor Hive have any provision for supporting secondary index lookups yet.

The following table summarizes what we've discussed so far:

Oozie (26)
Ops And DevOps (23)
Parquet (14)
Performance (13)
Pig (36)
Project Rhino (5)
QuickStart VM (6)
Search (25)
Security (32)
Sentry (2)
Spark (40)
Sqoop (24)
Support (5)
Testing (9)
This Month In The Ecosystem (16)
Tools (9)
Training (46)
Use Case (69)
Whirr (6)
YARN (15)
ZooKeeper (24)
Archives by Month

	Apache Phoenix	Impala	Apache Hive
Syntax	SQL	SQL	HiveQL
Key goal	High-performance SQL queries over HBase for low-latency applications	Interactive exploratory analytics on large data sets	Batch processing (e.g., ETL)
Secondary indexes	Yes (non-ACID compliant)	No	No
Dedicated daemons	No	Yes	Yes
HBase specific?	Yes	No	No

Future Work

The Phoenix project is investigating **integration with transaction managers** such as **Tephra** (from Cask). It is also trying to incorporate **query optimizations** based on the size and cardinality of the data. Although Phoenix supports **tracing** now, more work is needed to round out its monitoring and management capabilities.

Conclusion

Phoenix offers some unique functionality for a certain set of HBase use cases. You can begin playing with Phoenix by following the **installation instructions** (you will need HBase 1.0, which ships as part of **CDH 5.4**), and you can view the source code **here**.

As with everything else in Cloudera Labs, Phoenix integration is not supported yet and it's for experimentation only. If you have any feedback or comments, let us know via the **Cloudera Labs discussion forum**.

Srikanth Srungarapu is a Software Engineer at Cloudera, and an HBase committer.

Filed under:

Cloudera Labs

HBase

No Responses

Leave a comment

Name REQUIRED

Email REQUIRED
(WILL NOT BE PUBLISHED)

Website

Comment

Leave Comment

Prove you're human! *

4 – two =