Enable Apache Phoenix to Support SQL Better

Abstract

Apache Phoenix is a relational database layer over HBase delivered as a client-embedded JDBC driver targeting low latency queries over HBase data. It compiles SQL query into HBase scans and coprocessors, then returns query results in the format of regular JDBC result sets. Although Phoenix has al- ready had an almost perfect support for SQL, there is still some space for im- provement. The aim of this project is to improve performance by adding some key missing optimisations and add new SQL operators for Phoenix. This pro- posal is based on these jiras(PHOENIX-2405, PHOENIX-505, PHOENIX- 258, PHOENIX-671) created by Phoenix community. After this project, all of these jiras should be resolved. Apache Phoenix will gain a better performance when it executes SQLs contain MIN, MAX and DISTINCT and also become more stable when the server-side performs the computation of ORDER BY. Moreover, some new SQL operators will be added for Apache Phoenix.

Benefits to Community

Performance and stability always have a hight priority for large-scale data processing system. This project will replace the buffer data strategy of ORDER BY leading Phoenix to become more stable. The project also plans to apply efficient optimisations on MIN, MAX and DISTINCT reducing the time complexity from O(n) to O(1), which can save numerous time when the dataset is significantly large.

Moreover, SQL has already been a universal standard for all the software engineers. Missing any operator of SQL would generate numerous trouble for users. This project can reduce the inconvenience caused by missing SQL operators leading Apache Phoenix to become more user-friendly.

Brief Design and Roadmap

Apache Phoenix has two parts: the client-side and the server-side. The server-side is in charge of receiving RPC invokes from client-side and then generating coprocessors which query the local data in each HBase region server. The client-side collects the query result from the server-side and the perform merge, aggregation computations. As users communicate with Apache Phoenix in the format of SQL, the client-side is also responsible for parsing SQL and creating a query plan.

All the SQL operators which involved in this proposal are UNION oper- ator or aggregation operators. It means that this project will mainly focus on the client-side adding support for new SQL syntax and implement the computation component. In addition, to reduce the load of network improving the performance, coprocessors are needed to product semi-aggregation result for aggregation operators. Therefore, there is also some work on the server-side adding or modifying aggregator classes.

Based on this analysis, the process of adding or modifying each operator can be divided into two phases:

- Enable PhoenixSQL to support new SQL syntax
  – Add support to PhoenixSQL.g for new operators.
  – Modify QueryCompiler to adapt new SQL syntax.

- Implement the computation components
  – Create or modify aggregator classes which are executed in the server-side.
  – Implement the computation parts based on the meaning of SQL on the client side.

As the key of this project is the optimisation of computation, the main workload will focus on the second parse. For PHOENIX-2405, I will change the buffer data strategy from ’memory mapped files’ to ’memory + spill to disk’. Although this modify can increase the stability of Phoenix, it may reduce the performance. Therefore, a suite of performance should be provided. For PHOENIX-505 and PHOENIX-258, Apache Phoenix always performs a full scan for MIN, MAX, and DISTINCT, even when they are on a prefix of a compound key. The main effort to resolve the two jiras is to convert the full scan to other optimised computations, such as skip scan or select the first or last row directly.

Scheduling

- Before the GSoC term: All my examinations will be held before the end of April. After that, I’m going to start to read codes and test cases of Apache Phoenix and try to fix small bugs and resolve sample issues.

- 1st ~ 3rd weeks: I plan to change the buffer data strategy of Spoolin- gResultIterator from ’memory mapped files’ to to ’memory + spill to disk’, and then I will implement the performance tests comparing the performance of two strategies.

- 4rd ~ 5th weeks: I will change the MIN and MAX computation from a full scan to a scan with LIMIT 1 if the only aggregation is MIN(value) when the index is on value ASC and MAX(value) when the index is on value DESC.

- 6th ~ 7th weeks: I will optimise the execution plan of DISTINCT using skip scan. As this optimisation is not suitable for all the situa- tion, it also necessary to detect when the optimisation can be applied automatically.

- 8th ~ 9th weeks: I plan to work on UNION operator. As UNION ALL has been supported by Apache Phoenix and the process of supporting UNION and UNION ALL is very similar to each other, I will following the roadmap of developing UNION ALL to support UNION.

- 10th ~ 11th weeks: The two weeks are buffer weeks, just in case of some unexpected things happen. If everything goes well, I will work on PHOENIX-2062, PHOENIX-424, PHOENIX-423.

- 12th week: Code review, bug fixing and modifications based on community feedback. I will also try to analyse the bottleneck and improve the performance.

Deliverables

- The strategy of buffer data in order by can switch to 'memory + spill to disk', and performance tests are provided.

- MIN, MAX, DISTINCTION operator should be executed more quickly in some particular conditions.

- Apache Phoenix acquires the ability to support the SQL syntax: query UNION query.

- A suite of well-designed unit and integration test cases.

About Me

I'm an MSc(Computing) student at Imperial College London. I had worked for about three years at Alibaba Group in the area of distributed computing. Although I don't have much experience in Apache Phoenix, Im quite familiar with HBase. When HBase didnt support coprocessors, I implemented a very simple one with my colleagues. Here are my technical skills:

- Rich experience in developing distributed systems. • Proficient in Java as well as JVM tuning, profiling, and JVM memory management.

- Owning the ability of managing large-scale data with experience of writing programs which process tens of TB size data every day.

- Participating in operation and maintenance of clusters which consist of over 100 nodes. Familiar with Linux and its toolkits.

- Skilled in Hadoop, HBase, and Zookeeper.

Further details, you can get from my CV.