Design Considerations

Kevin Liew

May 19, 2016

1 RPC

How are ResultSets transferred? Batches contained in Thrift objects? Contiguous batches of TRowSet.

```
ie.
ResultSet{
    TRowSet batch1{
        TColumn1{row1, row2},
        TColumn2{row1, row2},
        columnCount = 2
    }
    TRowSet batch2{
        TColumn1{row3, row4},
        TColumn2{row3, row4},
        columnCount = 2
    }
}
```

except that the columns are in ‘binaryColumns‘ rather than a list of TColumn after HIVE-12049

2 Interface

Should we give priority to the server or client preference for compressors? ie. the client prefers snappy first, gzip second, but the server prefers gzip first, snappy second. Should we use gzip or snappy?

3 Compressor-Decompressor

How do we handle the case where the compressor throws an exception while handling a batch? Do we restart serialization and

serialize the batch as an uncompressed column? What do we do with prior batches that were already serialized? We may have to restart serialization of the whole result set. Is there a better solution? Prior batches will be compressed. If an exception is thrown while processing a column, it won't be compressed for that batch. The client will receive it as an uncompressed column and will know that it does not need to be decompressed. The server will still try to compress that column for all other batches and those that succeed will be compressed.

Where does decompression occur in the client? Will we operate on (Encoded)ColumnBasedSet or directly on TRowSet? Which files should I look at? We probably get a TRowSet and deserialize that to (Encoded)ColumnBasedSet from which we decompress batch(es).

hive.server2.thrift.resultset.max.fetch.size: "Max number of rows sent in one Fetch RPC call by the server to the client." If the client receives batch-by-batch, then we don't need to store the batch size in TEnColumn because the client will receive and decompress each batch separately instead of receiving one huge blob with all batches? So does the client receive the batches separately or in one blob? Batch-by-batch so we do not need a list of batch sizes.

How can we ensure that result sets are compressed for cases where the query does not invoke any map-reduce tasks (ie. select * from table)? Our SerDe is not called so if we write our compressor within the SerDe, then some queries will not be compressed.