Table of Contents

HBASE-18715 Isolate Large Query

Large queries are very expensive than the general operation, and often delay insensitive. if not isolate, Handlers may be all used by the large query, lead to Put or Get slowly, so there need be an mechanism to isolate the large query, for example we can use a separate RpcExecutor to handle this.

For the following type of queries we can think it as a large query:

1. Scan has no startkey or endkey

2. Client call Scanner's next more than a certain threshold

   the threshold can be determined through hbase.ipc.server.largequery.maxseq which default value is 20 (according to our query experiences, maybe not suitable), and the value can also be changed dynamically (Based on HBASE-12147 Feature).

3. Client call custom Coprocessor which involves large queries(such as AggregateService)

And for large queries, RegionServer should meet the following requirements:

1. Decide whether to isolate the large query dynamically

   Can be achieved through hbase.ipc.server.largequery.isolate which default value is true, can also be changed dynamically.

2. Setup certain Handlers to deal with large queries only.

   The number of Handlers can be setup through hbase.regionserver.largequery.handler.count which default value is 0, in this case the isolate feature is disabled, Even if hbase.ipc.server.largequery.isolate is set to true.

3. Dynamic decide if we should CacheBlocks when executing a large query

   In some Offline scenarios, there is a need to scan lots of data from HBase, but those data are not used frequently, and no need to cache them. so we can bypass the CacheBlock feature if hbase.regionserver.largequery.cacheblock is set to false.

4. In offpeak hours, large queries are not isolate

Implementation Details

Isolation control is mainly achieved through the extension of SimpleRpcScheduler. in the original implementation, Handlers are divided into 3 different groups:

(1) priorityExecutor to deal with the high priority request

(2) replicationExecutor to deal with the replication request

(3) callExecutor to deal with request from user tables

Based on this, we can introduce a largeQueryExecutor to handle with large query request, and bind some Handlers on it. when request come in, do the following judgment:

1. if hbase.ipc.server.largequery.isolate is enabled

2. if hbase.regionserver.largequery.handler.count is larger than 0

3. if the request priority is marked as LARGEQUERY

   The logic is mainly defined through getPriority of AnnotationReadingPriorityFunction class, we should do some extensions of this method.

   - if the come in request is a ScanRequest but has no startKey or endKey, mark it as LARGEQUERY priority.

   - if the come in request is a ScanRequest and the nextCallSeq value is larger than hbase.ipc.server.largequery.maxseq, also mark it as LARGEQUERY, no matter what startKey or endKey defined.

   - if the request is a CoprocessorServiceRequest, we should use CoprocessorJudger to determine if the coprocessor call involves large queries. for example if we use org.apache.hadoop.hbase.coprocessor.AggregateImplementation, we can define a Judger like this:

```
public class AggregateServiceJudger implements CoprocessorJudger {
    @Override
    public boolean isLargeQuery(ByteString msg) {
        try {
            AggregateRequest request = AggregateRequest.parseFrom(msg);
            Scan scan = request.getScan();
            String startRow = scan.getStartRow().toStringUtf8();
            String stopRow = scan.getStopRow().toStringUtf8();
            if ("".equals(startRow) || "".equals(stopRow)) {
                return true;
            }
        } catch (InvalidProtocolBufferException e) {
            LOG.info(e.getMessage(), e);
        }
        return false;
    }
}
```

   And introduction it through hbase.regionserver.largequery.judger.class, so HBase can use it.

If the request satisfy any of the above conditions, and current hour is not in offpeak hours, use largeQueryExecutor to handle it, achieve large query isolated.

Isolation Effect

In our test cluster, after isolate large queries, cpu resources used by large query can be limit below 40%. but if not isolate, cpu uses may be up to 100% which affect the general operation.

As shown in the figure, hbase09.nh enabled isolate feature.

Configure Parameter

1. hbase.ipc.server.largequery.isolate

   Whether or not isolate large query, default values is ture, can be changed dynamically.

2. hbase.ipc.server.largequery.maxseq

   when call Scaner's next more than this threshold, think it as a large query, default value is 20, can be changed dynamically.

3. hbase.regionserver.largequery.handler.count

   Number of Handlers to deal with large query, default value is zero which means large query isolate is not enabled, even if hbase.ipc.server.largequery.isolate is set to true.

4. hbase.regionserver.largequery.cacheblock

   when doing a large query, if enable the cacheblock feature, default value is true, can be changed dynamically.

5. hbase.regionserver.largequery.judger.class

   Judger to decide if the coprocessor call involves large query.