Parallelizing Skewed Hbase regions using Splittable DoFn

Prathap Reddy
Google
Agenda

- HBase and BigTable Overview
- HBase Snapshot Storage Structure
- Import Snapshots Pipeline
- Challenges & Resolutions
**HBase**

- Open Source Distributed Scalable Big Data Store
- Random read/write access patterns
- Automatic sharding of tables across regions
- Server side processing using Coprocessors

**Bigtable**

- Fully managed by Google
- High availability and automatic replication
- Auto Scaling based on application traffic
- Enterprise grade security and control
Hbase Snapshots

- Representation of table at point in time
- Zero Data Copying
- Minimal impact on region servers
- Creating Snapshot
  ```
  hbase> snapshot 'tableName', 'snapshotName'
  ```
- Export Snapshot to Google Cloud Storage
  ```
  hbase> hbase \n  org.apache.hadoop.hbase.snapshot.ExportSnapshot \n  -snapshot $SNAPSHOT_NAME \n  -copy-to $BUCKET_NAME$SNAPSHOT_EXPORT_PATH/data \n  -mappers $NUM_MAPPERS
  ```
# Hbase Storage Structure

<table>
<thead>
<tr>
<th>Table</th>
<th>(HBase table)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td>(Regions for the table)</td>
</tr>
<tr>
<td>Store</td>
<td>(Store per ColumnFamily for each Region for the table)</td>
</tr>
<tr>
<td>MemStore</td>
<td>(MemStore for each Store for each Region for the table)</td>
</tr>
<tr>
<td>StoreFile</td>
<td>(StoreFiles for each Store for each Region for the table)</td>
</tr>
<tr>
<td>Block</td>
<td>(Blocks within a StoreFile within a Store for each Region for the table)</td>
</tr>
</tbody>
</table>

* Region represents a key range (startKey - endKey) and may live on a different region server

* Store Files are also known as Hfiles
Importing to BigTable (v1)

- Build Snapshot Config
- Read Snapshot (HadoopFormatIO)
- Create Mutation
- Write to Bigtable

* Pipeline Source
Challenges

❖ Skewed regions

❖ Single Table Snapshots
Importing to BigTable (v2)

- Read multiple Snapshot Configs
- List Regions
- Read Region Splits (in parallel)
- Create mutation
- Write to multiple tables in Bigtable

* Snapshot config provides snapshot name, source path and target table name
Splittable Dofn

- Powerful abstraction with support to split each element of work
  
  \[(element, restriction) \rightarrow (element,\text{restriction}_1) + (element, \text{restriction}_2)\]

- Dynamic rebalancing to avoid stragglers
Splittable DoFn

- Restriction represents a portion of work (e.g.: OffsetRange, ByteKeyRange)

- Similar Syntax as DoFn with an additional `RestrictionTracker` parameter to `@ProcessElement` method

- `@GetInitialRestriction` - Represents the complete work for a given element

- `@SplitRestriction` (Optional) - Supports pre-splitting initial restriction
Execution of Splittable Dofn
@GetInitialRestriction

public ByteKeyRange getInitialRange(@Element RegionConfig regionConfig) {
    return ByteKeyRange.of(
        ByteKey.copyFrom(regionConfig.getRegionInfo().getStartKey()),
        ByteKey.copyFrom(regionConfig.getRegionInfo().getEndKey()));
}
@SplitRestriction
public void splitRestriction(@Element RegionConfig regionConfig,
@Restriction ByteKeyRange range,
OutputReceiver<ByteKeyRange> outputReceiver) {
    int numSplits = (int) Math.ceil((double) regionConfig.getRegionSize() / BYTES_PER_SPLIT);
    if (numSplits > 1) {
        RegionSplitter.UniformSplit uniformSplit = new RegionSplitter.UniformSplit();
        byte[][] splits =
            uniformSplit.split(
                range.getStartKey().getBytes(),
                range.getEndKey().getBytes(),
                getSplits(regionConfig.getRegionSize()),
                inclusive: true);
        IntStream.range(0, splits.length - 1).forEach((int i) ->
            outputReceiver.output(
                ByteKeyRange.of(ByteKey.copyFrom(splits[i]), ByteKey.copyFrom(splits[i + 1]))));
    } else {
        outputReceiver.output(range);
    }
}
@ProcessElement

public void processElement(
    @Element RegionConfig regionConfig,
    OutputReceiver<KV<SnapshotConfig, Result>> outputReceiver,
    RestrictionTracker<ByteKeyRange, ByteKey> tracker)
        throws Exception {
            try (ResultScanner scanner = newScanner(regionConfig, tracker.currentRestriction())) {
                for (Result result : scanner) {
                    if (tracker.tryClaim(ByteKey.copyFrom(result.getRow()))) {
                        outputReceiver.output(KV.of(regionConfig.getSnapshotConfig(), result));
                    } else {
                        break;
                    }
                } 
            }
            tracker.tryClaim(ByteKey.EMPTY);
Dynamic Splitting

- Splits current processing element into primary and residual parts
- Runners schedules residual part onto another instance
public class HbaseRegionSplitTracker extends RestrictionTracker<ByteKeyRange, ByteKey> implements RestrictionTracker.HasProgress {

    public HbaseRegionSplitTracker(boolean enableDynamicSplitting) {
        this.enableDynamicSplitting = enableDynamicSplitting;
    }

    public SplitResult<ByteKeyRange> trySplit(double fractionOfRemainder) {
        return enableDynamicSplitting ? this.byteKeyRangeTracker.trySplit(fractionOfRemainder) : null;
    }
}
Pipeline Graph

- **Read Snapshot Configs**
  - Succeeded
  - 1 sec
  - 1 of 1 stage succeeded

- **Restore Snapshots**
  - Succeeded
  - 17 sec
  - 1 of 1 stage succeeded

- **List Regions**
  - Succeeded
  - 1 sec
  - 2 of 2 stages succeeded

- **Read Regions**
  - Succeeded
  - 17 min 41 sec
  - 2 of 2 stages succeeded

- **Wait.OnSignal**
  - Succeeded
  - 3 sec
  - 3 of 3 stages succeeded

- **Write to BigTable**
  - Succeeded
  - 2 min 34 sec
  - 1 of 1 stage succeeded

- **Clean restored files**
  - Succeeded
  - 1 sec
  - 1 of 1 stage succeeded
Benchmark Tests

❖ Snapshot Datasets

➢ 104 GB with 19 regions (6 regions of 3.5 GB in size and remaining 13 regions are approximately 7 GB)

➢ 875 GB with 14 regions (Mixed region sizes varying from 30GB to 98 GB)

❖ Enabled and Disabled Dynamic Splitting

❖ 10 - 30% improvements in Job Duration with reduced VCPU Consumption

* Beyond Initial splits enabling further splitting didn’t yield significant differences