Deployment Planning and Optimization for Big Data & Cloud Storage Systems

Bianny Bian
Intel Corporation
Outline

- System Planning Challenges
- Storage System Modeling w/ Intel CoFluent Studio
  - Simulation Methodology
  - Cloud: Swift Simulator
  - Big Data: HDFS and HBase Simulator
- Use Cases
Why System Planning

1. Planning
   - H/W Component Selection
   - Cluster Setup

2. Deployment
   - System Deploy

3. Optimization
   - Software /JVM /OS Settings
   - Biz Code

4. Operation
   - With Biz Expansion
     - Larger Data Volume
     - More Data Sources
     - Quicker Data Injection
     - Higher Concurrency
     - More analytics
     - ...
   - System Needs Scale

Bad Planning = Long Provision Time or Waste Resources
System Planning Challenges

- Great Hardware Diversity
- Increasing Software Complexity
- Providers are commonly addressing these challenges

How to minimize cost?

How to get the most out of the hardware?

How to predict system scalability?

How to offer the best user experience?
Outline

- System Planning Challenges
- Storage System Modeling w/ Intel CoFluent Studio
  - Simulation Methodology
  - Cloud: Swift Simulator
  - Big Data: HDFS and HBase Simulator
- Use Cases
Simulation with Intel® CoFluent™ Studio

Enables fast “What if?” analysis with a virtual system
Simulation Architecture

What-If Analysis for
- S/W stack optimization
- Predict perf on varies node, network and disk configuration
- Explore against users number and cluster size

Details @ ICPP-2014 paper “Simulating Big Data Clusters for System Planning, Evaluation and Optimization”
Cloud Storage: Swift Simulator

Logic View of a Swift Cluster

Frontend model in iCoF

Backup model in C++
Ring parsing, Node mapping, Co-routine scheduling ...
Big Data Storage: HDFS Simulator
Big Data Storage: HBase Simulator

- RPC queue
- Flush queue
- Compact/Split queue
Output: Workload and H/W metrics

- Throughput
- Latency
- Resource usage
  - CPU
  - Network
  - Disk I/O
  - ....
Output: Software States & Metrics

- Rich metrics for detailed system behavior analysis over time.

Charts from HBase Simulation
Simulation vs. Measurements - Overall Performance

Charts from Swift Simulation

Consistent High Accuracy with Cluster, H/W, S/W or Workload Changes

Workload: Currency Changes

Cluster: Storage Node # Changes

HW: Network Changes
Simulation vs. Measurements - System S/W State

Charts from HBase Simulation

Measurement

Simulation

IOPS

HFiles

Compaction Queue

Flush Queue

High Fidelity on System S/W states
Simulation Speed

- Real cluster environment: 4 node (SNB DP) cluster
- Simulation environment: UP Desktop
- Speed:
  - Swift Model: 1x ~ 10x slowdown
  - HDFS Model: >2x faster than real execution
  - HBase Model: 10x slowdown ~ 10x faster than real execution

<table>
<thead>
<tr>
<th>Cluster Size</th>
<th>Data Volume</th>
<th>Concurrency</th>
<th>Simulation Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 nodes</td>
<td>400GB</td>
<td>200</td>
<td>29 min (vs. 71 min)</td>
</tr>
<tr>
<td>40 nodes</td>
<td>4TB</td>
<td>2K</td>
<td>4.6 hour</td>
</tr>
<tr>
<td>100 nodes</td>
<td>10TB</td>
<td>5K</td>
<td>6.5 hour</td>
</tr>
</tbody>
</table>
Outline

- System Planning Challenges
- Storage System Modeling w/ Intel CoFluent Studio
- Use Cases
  - HDFS (IDF’13 presentation)
  - HBase
  - Swift
Use case: HDFS Simulation

- Planning a Video Streaming System (Details @ IDF’13)
  - 1K concurrency, 2.5MB/s/viewer
  - Baseline: 1GbE, 4 Drivers/Node, 30 Nodes

Planning Result: 30 Nodes => 20 Nodes => 10 Nodes
Use case: HBase Simulation

- Planning a Video Analytics System
  - Estimate ratio of “HD cameras : Xeon servers”
  - Determine the best HW/SW configurations for typical scenarios:
    - Best Insert Performance || Best Query Latency || Balanced

AVG: 1GB video/hour/camera
Peak: ~ 4GB /hour/camera

AVG: 200MB/h/c
unstructured vector feature data + structured meta data + thumbnail pics
## Simulation Results

- **Estimate Ratio for typical scenarios**

<table>
<thead>
<tr>
<th>RS number</th>
<th>200 Cameras</th>
<th>1K Cameras</th>
<th>10K Cameras</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT</td>
<td>3</td>
<td>11</td>
<td>106</td>
</tr>
<tr>
<td>QUERY</td>
<td>5</td>
<td>22</td>
<td>215</td>
</tr>
<tr>
<td>BALANCE</td>
<td>4</td>
<td>14</td>
<td>135</td>
</tr>
</tbody>
</table>

- **Recommend Configurations**

<table>
<thead>
<tr>
<th></th>
<th>Flusher Handler</th>
<th>Blocking StoreFiles</th>
<th>Memstore Block Mul</th>
<th>Compact min</th>
<th>Large Compact Thread</th>
<th>Small Compact Thread</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT</td>
<td>24</td>
<td>3000</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>QUERY</td>
<td>24</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>14</td>
<td>10</td>
<td>...</td>
</tr>
<tr>
<td>BALANCE</td>
<td>8</td>
<td>100</td>
<td>4</td>
<td>10</td>
<td>6</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

- **Choice of Production Environment**
  - Multiple Medium Size Clusters instead of one Large Cluster

- **Explore different scenario to determine best HW and SW configuration**
Simulation Results – System Status

Options | IOPS | Cameras | Blocking Freq | CL Buffer @ Blocking | + HFiles/Region @ Peak | + Compact Queue @ Peak
--- | --- | --- | --- | --- | --- | ---
Baseline | 174 | 250 | 210 sec | 2.3GB | 30 | 160:0
INSERT | 285 | 400 | 100 sec | 460MB | 137 | 580:300
QUERY | 133 | 190 | 240 sec | 3.3GB | 17 | 85:0
BALANCE | 220 | 300 | 230 sec | 1.9GB | 42 | 100:0

Rich Metrics to Know System Behavior before Provisioning
Use case: Swift Simulation

- Optimizing a Swift Cluster
  - Goal: Double system uploading throughput
  - Baseline: 1 proxy + 3 storage, 8 Disks / node, 1Gb network

Identify and Fix Bottlenecks Very Quickly
Summary

- Big Data and Cloud Storage introduces new challenges in planning and optimizing
- Traditional analysis solutions not practical
- Intel® CoFluent™ Studio enables fast system analysis and optimization before hardware/software provisioning
  - Anticipate user experience
  - Reduce operating costs with optimal use of hardware and software
For More Information

cof-info@intel.com & cofluent.intel.com
Backup
Swift Simulation Input

- Workload Parameters
  - Concurrency
  - Request Size
- S/W Settings
  - Role setting (proxy, storage) for each node
  - Object ring
  - Proxy/object worker number
  - Object size
- H/W Settings:
  - Cluster size
  - System Components (CPU, Disks, Memory, Network)
  - Network topology
HBase Simulation Input: Workload

Consult Customers

Tables
- Two tables, share row key
  - Feature table (F): Two CF
  - Thumbnail table (T): One CF, LOB, no compaction/split
- Row key design: Put random hash value at the front for load balance
  - Format: [Hash value (001~099)] # [Camera ID (6 digital)] # [Timestamp (17 digital)]
  - Raw key % Region number -> Region

Table Design

Record Pattern

Records
- Feature table
  - CF1: 3 vector columns
    - Un-structured, byte array
    - Per column: 50-100-500KB
  - CF2: 3 meta columns
    - Structured
    - Per column: 9K-11K evenly distributed
- Thumbnail table, 1CF, six columns (3 Pic + 3 Info)
  - Picture columns: 209KB-2MB (even distribution)
  - LOB: write link (500KB) only in Region after Flush
- Arrival pattern – all 1/1 mapping
  - Row key: F-CF1-pic1, F-CF1-pic2, F-CF1-pic3, F-CF2-m1, F-CF2-m2, F-CF2-m3
  - Row key: T-CF1-pic1, T-CF1-pic2, T-CF1-pic3, T-CF1-inf1, T-CF1-inf2, T-CF1-inf3
- Per row key, insert (100KB + 10KB + 3KB + 500KB) x 3 = 1.8MB

Simulation Input

- <ClientSettings>
  - <Mode>: MapRed/Threaded
  - <Concurrency>: 1000
  - <ThreadPerProcess>: 10
  - <ProcessPerServer>: 1
- <ClientSettings>

- <Table ID="0" CFCnt="2" RegionNum="24" Combined="true">
  - <AutoFlush>1</AutoFlush>
  - <FlushBufferSize>0</FlushBufferSize>
  - <WriteHLog>1</WriteHLog>
  - <DeferredHLog>0</DeferredHLog>
  - <HRegionSettings>
    - <HRegion ID="0" StartKey="0" EndKey="99999999" />
    - <HRegion ID="1" StartKey="10000000" EndKey="" />
  - <HRegionSettings>
  - <ColumnFamily ID="0" ColumnCnt="3" Combined="true">
    - <Column ID="0">0</Column>
      - <DistributionTypeOfSize>0</DistributionTypeOfSize>
      - <valueseed1>10</valueseed1>
      - <valueseed2>10</valueseed2>
      - <Column>
    - <Column ID="1">1</Column>
      - <DistributionTypeOfSize>0</DistributionTypeOfSize>
      - <valueseed1>10</valueseed1>
      - <valueseed2>10</valueseed2>
      - <Column>
  - <Column>

2014 Storage Developer Conference. © Intel Corporation. All Rights Reserved.
## HBase Simulation Inputs: S/W and H/W

### S/W Settings
- JVM/OS
- HBase (>40)
- HDFS

### Cluster Settings
- Node #
- Topology
- Processor
- Network
- Storage
- Memory

### HBase parameters modeled

<table>
<thead>
<tr>
<th>HBase parameter</th>
<th>HDFS Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>HBASE_HEAPSIZE</td>
<td>hbase.regionserver.handler.count</td>
</tr>
<tr>
<td>hbase.client.write.buffer</td>
<td>hbase.regionserver.hlog.blocksize</td>
</tr>
<tr>
<td>hbase.hregion.majorcompaction</td>
<td>hbase.regionserver.hlog.enabled</td>
</tr>
<tr>
<td>hbase.hregion.majorcompaction.jitter</td>
<td>hbase.regionserver.logroll.multiplier</td>
</tr>
<tr>
<td>hbase.hregion.max.filesize</td>
<td>hbase.regionserver.logroll.period</td>
</tr>
<tr>
<td>hbase.hregion.memstore.block.multiplier</td>
<td>hbase.regionserver.maxlogs</td>
</tr>
<tr>
<td>hbase.hregion.memstore.flush.size</td>
<td>hbase.regionserver.optionallogflushinterval</td>
</tr>
<tr>
<td>hbase.hregion.preclose.flush.size</td>
<td>hbase.regionserver.region.split.policy</td>
</tr>
<tr>
<td>hbase.hstore.blockingStoreFiles</td>
<td>hbase.regionServer.regionSplitLimit</td>
</tr>
<tr>
<td>hbase.hstore.blockingWaitTime</td>
<td>hbase.regionServer.regionSplitLimit</td>
</tr>
<tr>
<td>hbase.hstore.compaction.kv.max</td>
<td>hbase.regionserver.thread.compaction.large</td>
</tr>
<tr>
<td>hbase.hstore.compaction.max</td>
<td>hbase.regionserver.thread.compaction.small</td>
</tr>
<tr>
<td>hbase.hstore.compaction.max.size</td>
<td>hbase.regionserver.thread.compaction.throttle</td>
</tr>
<tr>
<td>hbase.hstore.compaction.min.size</td>
<td>hbase.regionserver.thread.split</td>
</tr>
<tr>
<td>hbase.hstore.compaction.ratio</td>
<td>hbase.rowlock.wait.duration</td>
</tr>
<tr>
<td>hbase.hstore.compaction.ratio.offpeak</td>
<td>hbase.server.thread.wakewfrequency</td>
</tr>
<tr>
<td>hbase.hstore.compactionThreshold</td>
<td>hbase.server.thread.wakewfrequency.multiplier</td>
</tr>
<tr>
<td>hbase.regionserver.compactionChecker.majorCompactPriority</td>
<td>hbase.store.delete.expired.storefile</td>
</tr>
<tr>
<td>hbase.regionserver.flusher.count</td>
<td>ipc.server.max.callqueue.length</td>
</tr>
<tr>
<td>hbase.regionserver.global.memstore.lowerLimit</td>
<td>ipc.server.read.threadpool.size</td>
</tr>
<tr>
<td>hbase.regionserver.global.memstore.upperLimit</td>
<td></td>
</tr>
</tbody>
</table>