Hadoop Map-Reduce Tuning and Debugging

Arun C Murthy

acmurthy@apache.org

Yahoo! Grid Team, CCDI
Existential Angst: Who Am I?

- Lowly Engineer, CCDI Yahoo!
  - Design, review, and implement features in Hadoop, specifically Map-Reduce (and security)
  - Working on Hadoop full time since March 2006

- Apache Software Foundation
  - Hadoop Core Committer
  - Member of Hadoop Program Management Committee
Topical Matters

- Peek inside your MR application
- Tuning
- Debugging (god forbid!)
Counters …

• Often MR applications have countable ‘events’

• For e.g. the Map-Reduce framework ‘counts’ the bytes read/write on HDFS and the local filesystem

• To define your own:
  – static enum Counter {C1, C2}
  – reporter.incrCounter{Counter.C1, 1}
Counters continued...

<table>
<thead>
<tr>
<th>Counter</th>
<th>Map</th>
<th>Reduce</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Systems</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local bytes read</td>
<td>15,436,320,026</td>
<td>8,575,518,710</td>
<td>24,011,838,736</td>
</tr>
<tr>
<td>Local bytes written</td>
<td>17,333,083,926</td>
<td>8,575,518,710</td>
<td>25,908,602,636</td>
</tr>
<tr>
<td>HDFS bytes read</td>
<td>5,093,892,056</td>
<td>0</td>
<td>5,093,892,056</td>
</tr>
<tr>
<td>HDFS bytes written</td>
<td>0</td>
<td>31,139,543,728,535</td>
<td>31,139,543,728,535</td>
</tr>
<tr>
<td>Job Counters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Launched map tasks</td>
<td>0</td>
<td>0</td>
<td>727</td>
</tr>
<tr>
<td>Launched reduce tasks</td>
<td>0</td>
<td>0</td>
<td>724</td>
</tr>
<tr>
<td>Data-local map tasks</td>
<td>0</td>
<td>0</td>
<td>602</td>
</tr>
<tr>
<td>Rack-local map tasks</td>
<td>0</td>
<td>0</td>
<td>96</td>
</tr>
<tr>
<td>Map-Reduce Framework</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Map input records</td>
<td>57,300,102</td>
<td>0</td>
<td>57,300,102</td>
</tr>
<tr>
<td>Map output records</td>
<td>57,300,102</td>
<td>0</td>
<td>57,300,102</td>
</tr>
<tr>
<td>Map input bytes</td>
<td>5,093,891,958</td>
<td>0</td>
<td>5,093,891,958</td>
</tr>
<tr>
<td>Map output bytes</td>
<td>8,005,032,069</td>
<td>0</td>
<td>8,005,032,069</td>
</tr>
<tr>
<td>Combine input records</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Combine output records</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Reduce input groups</td>
<td>0</td>
<td>33,668,268</td>
<td>33,668,268</td>
</tr>
<tr>
<td>Reduce input records</td>
<td>0</td>
<td>57,061,253</td>
<td>57,061,253</td>
</tr>
<tr>
<td>Reduce output records</td>
<td>0</td>
<td>833,247,066,047</td>
<td>833,247,066,047</td>
</tr>
</tbody>
</table>
Tuning Map-Reduce Applications

• Where do I start?
  – User code (the less said, the better!)
    • Use configure and/or close
    • Use the OutputCommitter and setup/cleanup tasks
  – The framework
    • Input
    • Data-path
    • Output
Tuning – A Diversion

• Tell HDFS and Map-Reduce about your network!
  – Rack locality script: `topology.script.file.name`

• Number of maps
  – Data locality

• Number of reduces
  – You do not, I repeat, do not, need a single output file!
Tuning – Step One of Three: The Input

• Amount of data processed per Map
  – Consider fatter maps
  – Custom InputFormat:
    • InputSplit
    • RecordReader
Tuning – Step Two of Three: The Data-Path
Tuning – The Mapper

- User’s Map Method
- User’s Record Reader
- Input Split

Sort Buffer

- User’s Partitioner/Comparator/Combiner

Sort/Combine & Spill

Local Disk

- Spill file
  - R0
  - R1
  - R2
  - R3

- In-Mem Spill index
  - Offset, uncompressed, compressed length

- io.sort.mb

- Record info (K,V offsets and Partition) maintained

Every map task does a small sort
Tuning – The Mapper

- Index
- Wrap around buffer
- Sort buffer, bytes array
- io.sort.record.percent * io.sort.mb
- io.sort.mb * io.sort.spill.percent
- Index and partition buffers (int arrays)
Tuning – The Mapper

- `io.sort.mb`
  - Controls the sort buffer size

- `io.sort.factor`
  - Controls the number of files simultaneously merged (recommendation: 100)

- `io.sort.record.percent`
  - Controls the number of records that can be collected

- `io.sort.spill.percent`
  - Soft limit that controls when sort/spill starts
Tuning – The Shuffle

Shuffle: MxR number of intermediate file transfers to copy map-outputs to the reducers
Tuning – The Shuffle

• Map-side
  – Partitioner
  – Use combiners: the faster way to copy data is to do less of it!
  – Compression for map-outputs
    • `mapred.compress.map.output`
    • `mapred.map.output.compression.codec`
    • Native compression libraries (lzo)
Tuning – The Shuffle

• TaskTracker
  – Jetty threads on the TaskTracker
    • tasktracker.http.threads
  – In-memory Index Cache
    • mapred.tasktracker.indexcache.mb
Tuning – The Shuffle

- Stall shuffle
  - Can fit in memory
- Main Memory
  - Small Map Output
  - Large Map Output
  - Cannot fit in memory
  - Happens in parallel (copier threads)
- Disk
  - Spill merged data
  - Merge
  - Merge/Combine
  - Spill merged data
Tuning – The Shuffle

1st level (raw on-disk map outputs or merged map outputs)
io.sort.factor = 3

2nd level,
3rd level,...
(merge three files to one)

Main Memory

User's Reduce method

Output Filesystem
Tuning – The Shuffle

• **Reduce-side**
  - `mapred.reduce.parallel.copies`
  - `mapred.reduce.copy.backoff`
  - `mapred.job.shuffle.input.buffer.percent`
  - `mapred.job.shuffle.merge.percent`
  - `mapred.inmem.merge.threshold`
  - `mapred.job.reduce.input.buffer.percent`
Tuning – Step Three of Three: The Output

- OutputCommitter
- MultipleOutputs / MultipleOutputFormat

- Do you really need 3 replicas?
Tuning - Miscellaneous

• Speculative execution
• Heap size for the child
  – mapred.child.java.opts
• Re-use jvm for maps/reduces
  – mapred.job.reuse.jvm.num.tasks
• Last, not least: Raw Comparators
Tuning - RawComparator

```java
public class MyKeyClass implements WritableComparable {
    // Some data
    private int counter;
    private Text bigText;

    public void write(DataOutput out) throws IOException {
        out.writeInt(counter);
        bigText.write(out);
    }

    public void readFields(DataInput in) throws IOException {
        counter = in.readInt();
        bigText.readFields(in);
    }

    public int compareTo(MyKeyClass o) {
        int thisCounter = this.counter;
        int thatCounter = o.counter;
        return (thisCounter < thatCounter ? -1 : (thisCounter==thatCounter ? 0 : 1));
    }
}

public static class Comparator extends WritableComparator {
    public Comparator() {
        super(MyKeyClass.class);
    }

    public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2) {
        int n1 = WritableComparator.readInt(b1, s1);
        int n2 = WritableComparator.readInt(b2, s2);
        return (n1 < n2) ? -1 : (n1 == n2) ? 0 : 1;
    }
}
```
Profiling

• Set `mapred.task.profile` to `true`
• Profile a small range of maps/reduces
  – `mapred.task.profile.{maps|reduces}`
• hprof support is built-in
• Use `mapred.task.profile.params` to set options for the debugger
• Possibly `DistributedCache` for the profiler’s agent
Debugging – Oh no!

- Advanced technology – stderr – Hold on! Where do we find it?

Job **job_200810142005_0045**

<table>
<thead>
<tr>
<th>Task Attempts</th>
<th>Machine</th>
<th>Status</th>
<th>Progress</th>
<th>Start Time</th>
<th>Finish Time</th>
<th>Errors</th>
<th>Task Logs</th>
<th>Counters</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>task_200810142005_0045_m_000000_0</td>
<td>gs201394.inkтомсерарс.com</td>
<td>SUCCEEDED</td>
<td>100.00%</td>
<td>19-Oct-2008 04:22:22</td>
<td>19-Oct-2008 04:24:01 (1 mins, 39sec)</td>
<td></td>
<td>Last 5KB Last SKB All</td>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>

Go back to the job
Go back to JobTracker

Hadoop, 2008.
Debugging continued...

- Run job with ‘Local Runner’
  - Set `mapred.job.tracker` to `local`
  - Runs application in single process/thread

- Run on a single-node cluster i.e. your dev-box, with sampled data

- Set `keep.failed.task.files` to true and use the IsolationRunner
Questions?

• For more information:
  – Website: [http://hadoop.apache.org/core](http://hadoop.apache.org/core)
  – Mailing lists:
    • core-dev@hadoop.apache.org
    • core-user@hadoop.apache.org
  – IRC: #hadoop on irc.freenode.org