Main Memory Map Reduce (M3R)

PL Day, September 23, 2013

Avi Shinnar, David Cunningham, Ben Herta, Vijay Saraswat, Wei Zhang

In collaboration with:
Yan Li*, Dave Grove, Mikio Takeuchi**, Salikh Zakirov**, Juemin Zhang

* IBM China Research Lab
** IBM Tokyo Research Lab

Otherwise IBM TJ Watson Research Lab, New York

© 2013 IBM Corporation
M3R (engine)

- performance
- low latency

Hadoop

- resilience
- scalability

X10

- Java
- out-of-core

in-memory
X10

• “Java”-like language designed for performance and productivity at scale
• Asynchronous Partitioned Global Address Space programming model
  – async S: run S as a separate activity
  – at (P) S: run S at place P
  – finish S: wait for termination of children activities
    – MPI style barriers, local “atomic” synchronization…
• Advanced type system
  – reified generics, closures, dependent types…
• Multiple backends: Java, C++, CUDA

http://x10-lang.org/
Main-Memory Map Reduce in X10

- Sorting (keys)
- Secondary sorting (values)
- Combiners
- Map-only jobs
- Out of core shuffle
- Multi-threading
- Iterative jobs
- Debugging
- Profiling
- User controlled serialization

© 2013 IBM Corporation
M3R/Hadoop Architecture

Java Hadoop App

M3R/Hadoop adaptor

Hadoop Map Reduce Engine

M3R Engine

X10 M3R jobs

Java M3R jobs

HDFS data

HDFS

© 2013 IBM Corporation
Hadoop Job

File System
(HDFS)

Input
(InputFormat/
RecordReader/
InputSplit)

Map
(Mapper)

Reduce
(Reducer)

Output
(OutputFormat/
RecordWriter/
OutputCommitter)

Shuffle

File System

File System

© 2013 IBM Corporation
M3R/Hadoop Job: cache

File System (HDFS)

Cache

Input (InputFormat/RecordReader/InputSplit)

Map (Mapper)

Reduce (Reducer)

Output (OutputFormat/RecordWriter/OutputCommitter)

Shuffle

File System

File System

© 2013 IBM Corporation
M3R/Hadoop Job: in-memory

File System (HDFS)

Cache

Input
(InputFormat/
  RecordReader/
  InputSplit)

Map
(Mapper)

Reduce
(Reducer)

Output
(OutputFormat/
  RecordWriter/
  OutputCommitter)

Shuffle

© 2013 IBM Corporation
M3R/Hadoop Job: co-location

File System
(HDFS)

Input
(InputFormat/
RecordReader/
InputSplit)

Map
(Map)

Reduce
(Reducer)

Output
(OutputFormat/
RecordWriter
OutputCommitter)

Shuffle

© 2013 IBM Corporation
Iterated Matrix Vector multiplication

- Algorithm ("standard HPC")
  - Row block partition $G$
  - Replicate $V$
  - In parallel, at each place, multiply each row of $G$ with $V$.
  - In parallel, each place broadcasts its segment of $V$ to all others
    - This reassembles $V$ for next phase.

- Performance key
  - Read the appropriate part of $G$ once, never communicate it.
  - Reassembly is local.
M3R/Hadoop Job: locality

File System (HDFS)

Input (InputFormat/RecordReader/InputSplit)

Map (Mapper)

Reduce (Reducer)

Output (OutputFormat/RecordWriter/OutputCommitter)

Partitioner

Cache

Shuffle

© 2013 IBM Corporation
Partition Stability in M3R

• The reducer associated with a given partition number will always be run at the same place
  – Assuming the number of reducers and the number of places remains the same,
    • The number of reducers is determined by the application.
    • The number of places is fixed for the duration of the M3R server.
Sparse Matrix Vector Multiplication

![Graph showing the comparison of M3R/Hadoop and Hadoop in terms of time and size M. The graph illustrates the performance of matrix multiplication for different sizes of matrices.](image)
shinnar M3R/Hadoop Map/Reduce Administration

State: RUNNING  
Started: Thu May 09 13:42:46 EDT 2013  
Version: N/A  
Compiled: N/A  
Identifier: m3r20130509134246676

Cluster Summary (Heap Size is 122.02 MB/3.98 GB)

<table>
<thead>
<tr>
<th>Running Map Tasks</th>
<th>Running Reduce Tasks</th>
<th>Total Submissions</th>
<th>Nodes</th>
<th>Occupied Map Slots</th>
<th>Occupied Reduce Slots</th>
<th>Reserved Map Slots</th>
<th>Reserved Reduce Slots</th>
<th>Map Task Capacity</th>
<th>Reduce Task Capacity</th>
<th>Avg. Tasks/Node</th>
<th>Blacklisted Nodes</th>
<th>Graylisted Nodes</th>
<th>Excluded Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Scheduling Information

<table>
<thead>
<tr>
<th>Queue Name</th>
<th>State</th>
<th>Scheduling Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>running</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Filter (JobId, Priority, User, Name)  
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

<table>
<thead>
<tr>
<th>JobId</th>
<th>Priority</th>
<th>User</th>
<th>Name</th>
<th>Map % Complete</th>
<th>Map Total</th>
<th>Maps Completed</th>
<th>Reduce % Complete</th>
<th>Reduce Total</th>
<th>Reduces Completed</th>
<th>Job Scheduling Information</th>
<th>Diagnostic Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>job_m3r20130509134246676_0001</td>
<td>NORMAL</td>
<td>shinnar</td>
<td>Rebalancer for /hadoop_matrixes/pr/50k/G</td>
<td>100.00%</td>
<td>50</td>
<td>0</td>
<td>100.00%</td>
<td>160</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

This is M3R/Hadoop
Counters for task_m3r20130509134906363_0001_m_000000

Job Counters
- Launched map tasks: 1
- Other local map tasks: 1

Map-Reduce Framework
- Map input records: 50
- Map output records: 50

Profiler Information for task_m3r20130509134906363_0001_m_000000

Profiler Graph - close

<table>
<thead>
<tr>
<th>Name</th>
<th>seconds (nanoseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>total_measured</td>
<td>0.042 (42,327,000)</td>
</tr>
<tr>
<td>map.setup</td>
<td>0.002 (1,704,000)</td>
</tr>
<tr>
<td>map_setup:setup</td>
<td>0.001 (1,327,000)</td>
</tr>
<tr>
<td>map.setup:create</td>
<td>0 (68,000)</td>
</tr>
<tr>
<td>map.setup:configure</td>
<td>0 (22,000)</td>
</tr>
<tr>
<td>map.setup:other</td>
<td>0 (287,000)</td>
</tr>
<tr>
<td>map.open_input</td>
<td>0.004 (4,180,000)</td>
</tr>
<tr>
<td>map.open_input:Recordreader_create</td>
<td>0.004 (4,124,000)</td>
</tr>
<tr>
<td>map.open_input:other</td>
<td>0 (58,000)</td>
</tr>
<tr>
<td>map.run</td>
<td>0.034 (34,171,000)</td>
</tr>
<tr>
<td>map.run$run</td>
<td>0.034 (34,022,000)</td>
</tr>
<tr>
<td>map.run$run$create$key</td>
<td>0.001 (574,000)</td>
</tr>
<tr>
<td>map.run$run$create$value</td>
<td>0 (290,000)</td>
</tr>
<tr>
<td>map.run$run$Input</td>
<td>0.009 (9,114,000)</td>
</tr>
<tr>
<td>map.run$run$write</td>
<td>0.023 (23,439,000)</td>
</tr>
<tr>
<td>map.run$run$other</td>
<td>0.001 (605,000)</td>
</tr>
<tr>
<td>map.run$other</td>
<td>0 (149,000)</td>
</tr>
<tr>
<td>map.close_input</td>
<td>0 (46,000)</td>
</tr>
<tr>
<td>map.commit</td>
<td>0.002 (2,226,000)</td>
</tr>
<tr>
<td>map.commit$commit</td>
<td>0.002 (2,078,000)</td>
</tr>
<tr>
<td>map.commit$other</td>
<td>0 (148,000)</td>
</tr>
</tbody>
</table>

Go back to the job
Go back to JobTracker
M3R/Hadoop Cache Contents on shinnar

- hdfs://shinnar:8000
  - user
    - shinnar
      - hadoop_matrices
        - pr
          - 50K
  - perm_rebalance_temp_scratch_m3r_G
    - part-00000
    - part-00001
    - part-00002
    - part-00003
    - part-00004
    - part-00005
## M3R/Hadoop Cache Contents on shinnar

**hdfs://shinnar:9000/user/shinnar/hadoop_matrices/pr/50k/perm_rebalance_temp_scratch_m3r_G/part-00000**

### Splits

<table>
<thead>
<tr>
<th>Place</th>
<th>Machine</th>
<th># Blocks</th>
<th># Entries</th>
<th>Breakdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>shinnar</td>
<td>1</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>1</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

### Data at place 0, block 0, chunk 0

<table>
<thead>
<tr>
<th>Index</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2013 IBM Corporation
M3R / Hadoop

Compatibility

Performance

© 2013 IBM Corporation
## DML results (Nov. 2011)

<table>
<thead>
<tr>
<th>Model</th>
<th>Hadoop</th>
<th>M3R</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GNNMF</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100K</td>
<td>1489s</td>
<td>115s</td>
<td>13x</td>
</tr>
<tr>
<td>200K</td>
<td>1492s</td>
<td>185s</td>
<td>8.1x</td>
</tr>
<tr>
<td>400K</td>
<td>1481s</td>
<td>300s</td>
<td>4.9x</td>
</tr>
<tr>
<td><strong>Linear Regression</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000K</td>
<td>1272s</td>
<td>120s</td>
<td>10.6x</td>
</tr>
<tr>
<td>3000K</td>
<td>1438s</td>
<td>185s</td>
<td>7.8x</td>
</tr>
<tr>
<td>5000K</td>
<td>1473s</td>
<td>275s</td>
<td>5.4x</td>
</tr>
<tr>
<td><strong>PageRank</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100K</td>
<td>880s</td>
<td>452s</td>
<td>1.9x</td>
</tr>
<tr>
<td>200K</td>
<td>885s</td>
<td>574s</td>
<td>1.5x</td>
</tr>
<tr>
<td>400K</td>
<td>872s</td>
<td>530s</td>
<td>1.7x</td>
</tr>
</tbody>
</table>
Pig unit tests

2499 tests, 6 failures, 6 errors
Current Status / Future Work

• VLDB ‘12

• Things generally work quite well

• Working on out-of-core shuffle
  – Performance degradation instead of crashing

• Working on dynamic class loading

© 2013 IBM Corporation