Hierarchical Merge for Scalable MapReduce

Xinyu Que, Yandong Wang, Cong Xu,

Weikuan Yu*

Auburn University
Outline

- **Background and Motivation**
- Design of Hierarchical Merge Hadoop
- Performance Evaluation
- Conclusions and Future Work
Fast Data Analytics for Emerging Data Crisis

- IDC estimates 8 zettabytes of digital data will be generated in 2015
- Big demand from both scientific and industry sectors
  - Scalable data processing tools are needed to cope with big data
- Commercial companies, government agencies and universities are gearing up resources for efficient big data analytics
MapReduce Programming Model and Hadoop

- A popular parallel data processing model is MapReduce
- Hadoop is open-source implementation of MapReduce.
- It realizes simple *map* and *reduce* interfaces for users.
- Designed for commodity off-the-shelf hardware components.
- Widely deployed by many big data companies.
Hadoop-A: Pipelined Shuffle, Merge, and Reduce

- Hadoop Acceleration (Hadoop-A) developed at Auburn accelerates the merging of data in Hadoop.
- A complete pipeline overlaps the shuffle, merge, and reduce phases.

Network-levitated Merging (NetLev)

- Fetch a small header from each segment
- Built up a Global Priority Queue for all the segments
- GPQ then generates the final merged data for reduce function
Scalability Challenge

• Memory requirement for GPQ in the network-levitated merge
  • Application's dataset size: $S$
  • Data split size: $B$
  • Memory buffer size per segment: $M_b$
  • Total amount of memory per one GPQ: $M_t = M_b \times (S/B)$

• The memory requirement grows linearly with the number of segments

• Can we do better to increase the scalability while maintaining performance?
Outline

• Background and Motivation
• Design of Hierarchical Merge Hadoop
• Performance Evaluation
• Conclusions and Future Work
Solution: Hierarchical Merge

- Linear array is used to sort the incoming segments based on their size
- Smaller number of segments are moved into a Child Priority Queue (CPQ)
- All CPQs are then organized into a root priority queue (RPQ), which merges data for reduce
Scalability Improvement

- The segments are spread to into many small CPQs (Q), each with C segments. We can assume C=Q for convenience of reasoning.

- To keep the pipeline running with overlapped fetching, the number of segments in each active CPQ
  \[ \sqrt{S / B} \]

- The memory requirement grows in the order, which becomes
  \[ M_t = \sqrt{S / B} \ast M_b \]

- How to manage the merging process and buffers to contain the cost of the extra merging step?
Merging Orchestration and Buffer Management

- **Merging Orchestration**
  - The merging of RPQ and CPQs is overlapped through the use of multiple worker threads and double buffering
  - Priority based scheduler is used to synchronize the worker threads

- **Buffer Management**
  - Buffer manager is designed to be responsible for the buffers assignment and preemption
Outline

- Background and Motivation
- Design of Hierarchical Merge Hadoop
- Performance Evaluation
- Conclusions and Future Work
Experimental Testbed

- 21 nodes with dual-socket, quad-core 2.13 GHz Intel Xeon processors, PCI-Express Gen 2.0 bus.
- 500GB 7200RPM WD SATA hard drives, one per node.
- InfiniBand ConnectX-2 QDR Host Channel Adaptors.
- OFED 1.5.3.2 Release from Mellanox.
- Compared HM-Hadoop to Hadoop-A and the Original Hadoop 0.20.2 on IPoIB, 10GigE and RoCE (RDMA over Converged Ethernet).
Performance on Different Networks

- HM-Hadoop shows much better performance compared to the original Hadoop for all cases, improvement up to **27%**
- Compared to Hadoop-A, HM-Hadoop has a performance loss around **8%**, while provides better scalability
- HM-Hadoop also shows good scalability on different networks
Performance Benefits to Different Workloads

- HM-Hadoop shows significant improvement on reduce-heavy workloads:
  - 23.4% on InvertedIndex, 20% on TermVector, 18% on SequenceCount and 54.6% on Hive OrderBy
- Less benefits to Map-heavy applications
Benefits to I/O Access

- `vmstat` traces the disk activities on slave nodes
- HM-Hadoop significantly reduces the disk accesses:
  - 30.9% for read operations and 36.2% for write operations
  - The total reduction is up to 34.1%

<table>
<thead>
<tr>
<th></th>
<th>READ</th>
<th>WRITE</th>
<th>TOTAL (Kblocks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HM-Hadoop</td>
<td>31,088</td>
<td>40,833</td>
<td>71,921</td>
</tr>
<tr>
<td>Hadoop</td>
<td>45,031</td>
<td>64,039</td>
<td>109,070</td>
</tr>
</tbody>
</table>
Benefits to Disk Contention

- HM-Hadoop has the similar I/O service time as the original Hadoop.
- HM-Hadoop has similar or lower I/O wait time during the mapping phase of the execution.
- In the reducing phase, HM-Hadoop’s I/O requests only spend around 40% of the total time waiting in the queue, compared to 90% for the original Hadoop.
Outline

- Background and Motivation
- Design of Hierarchical Merge Hadoop
- Performance Evaluation
- Conclusions and Future Work
Conclusions and Future Work

- Proposed Hierarchical Merge as a new strategy to achieve scalable MapReduce
- Demonstrated that HM-Hadoop improves the execution time by up to 27% and reduces disk accesses by up to 34.1%.
- To investigate the benefits of Hierarchical Merge for more commercial and scientific workloads on large-scale commercial cloud computing systems