JUMMP: Job Uninterrupted Maneuverable MapReduce Platform

W. Clay Moody*, Linh B. Ngo*, Edward Duffy†, Amy W. Apon*

Computer Science Division of the School of Computing*
Clemson Computing and Information Technology†
Clemson University

custom illustration by @dpbays
Motivation [1 of 2]

- Hadoop is the de-facto implementation of the MapReduce programming model and used in many different disciplines

- Academic shared environment clusters accommodate a wide variety of research applications within a set of financial, technical, and administrative constraints.

- The Job Uninterrupted Maneuverable MapReduce Platform is an automated scheduling platform
  - Enables the integration of Hadoop into the existing large scale computational infrastructure
  - Supports high availability and continuous computing for research and education
• User Considerations
  – Research that uses Hadoop as a tool for the research
  – Study on the Hadoop eco-system that consider different hardware and software configurations
  – Academic course project requiring individual clusters
Motivation

• Environmental Considerations
  – **Cost** of hardware, space, power and cooling
  – **Technology** differences of Hadoop and traditional HPC configurations
  – Support of the Hadoop runtime environment

![Diagram showing Traditional HPC Environments vs Hadoop Clusters](image-url)
Design Objectives

- Provision individual Hadoop clusters within an academic HPC environment:
  - As *dynamic* execution environment on demand by users
  - In *user* space without root privileges
  - Minimal interactions from system administrators
  - Exist with environment beyond reservation limitations in *semi-persistent* manner
Hadoop Overview [1 of 3]

• Hadoop is Apache Foundation’s open-source implementation Google projects and research

• Hadoop Distributed File System (HDFS)
  – Implementation of the Google File System
  – Highly fault-tolerant distributed file system

• MapReduce Framework
  – Implementation of Google MapReduce
  – Application framework for process big data in-parallel across clusters
Hadoop Overview [2 of 3]

- **HDFS Nodes:**
  - *NameNode (NN)*
  - *Data Node (DN)*

- **MapReduce Nodes**
  - *JobTracker (JT)*
  - *TaskTracker (TT)*

- Traditionally **single** NN and JT in cluster

- **Multiple** DNs and TTs

Source: Apache Foundation
• Map Phase
  – Key-value pair input mapped to key-value pair outputs

• Shuffle Phase
  – Transfer of map output to reducers as input

• Reduce Phase
  – Operation on set of values for each key in map out

Source: Yahoo
JUMMP Design

• JUMMP is a Hadoop cluster where DN/TT continuously move throughout HPC environment while allowing survivability of Hadoop cluster

• Configured with two variables:
  – \( n \): number of DataNodes / TaskTracker
  – \( t_j \): scheduled time between jumps

• Built with Palmetto HPC Cluster at Clemson
  – Uses PBS Professional Scheduler, compatible with open source versions of PBS.

• Dedicated NN/JT node outside of scheduler

• Each DN/TT reserved with its own PBS job
Jumping Node Actions

• Launch DN/TT daemons and join Hadoop Cluster

• Perform Hadoop cluster duties as normal DN/TT would

• Await trigger to “jump” (time or event based)
  
  1. Schedule replacement DN/TT PBS job
  2. Stop Hadoop daemons
  3. Decommission and blacklist itself from cluster
  4. End PBS job
Performance Analysis

- DataNode and TaskTracker cause distinct degradation on performance due to jumps

- DataNode jumps cause re-replication of data blocks located in the jumping node

- TaskTracker jumps cause MapReduce tasks to be rescheduled that were executing on jumping node

- TaskTracker jumps cause cluster to be “undersized” until replacement is fully operational
Experiment Design

• Pool of 96 homogenous nodes within Palmetto (shown in Table I)

• Benchmarks and Datasets from Purdue’s PUMA project

• 100 runs of each experiment (shown in table II) on consistent dataset

• Baseline and three different jump times

• Recorded task times, job times, and jump times

---

**TABLE I: Node Configuration**

<table>
<thead>
<tr>
<th>Node</th>
<th>HP SL250s</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Xeon E5-2665 (2)</td>
</tr>
<tr>
<td>Cores</td>
<td>16</td>
</tr>
<tr>
<td>Memory</td>
<td>64 GB</td>
</tr>
<tr>
<td>Local Storage Capacity</td>
<td>900 GB</td>
</tr>
<tr>
<td>Networking</td>
<td>Infiniband</td>
</tr>
</tbody>
</table>

**TABLE II: Experiment Parameters**

<table>
<thead>
<tr>
<th>Application</th>
<th>Wordcount</th>
<th>Terasort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset Size</td>
<td>50 GB</td>
<td>300 GB</td>
</tr>
<tr>
<td>Node Count</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>Jump Times [mins]</td>
<td>7/10/15</td>
<td>20/40/60</td>
</tr>
</tbody>
</table>
Characteristics of Impact

Wordcount on 50 GB with 8 nodes

(a) No jump
(b) Jump during map phase
(c) Jump during reduce phase
Characteristics of Impact

Terasort on 300 GB with 32 nodes

(a) No jump
(b) Jump during shuffle phase
(c) Jump during reduce phase
Performance Measurements

Wordcount

Terasort
Performance Measurements

Map Task – Insignificant
Fails to add doomed task execution time

Wordcount

Terasort
Performance Measurements

Jobs - Affected by jump frequency

Wordcount

Terasort
Optimizing Jump Time

• Upper Bound
  – Maximum jump times to decrease overhead
  – Maximize nodes within cluster to increase parallel processing
  – Jump time must be less than maximum reservation window divided by the number of nodes

• Lower Bound
  – Hadoop behavior contributes to jump times
  – Must be greater than time to re-replicate all datablocks
  – Must be greater than the average execution time of the reduce tasks of currently executing MapReduce jobs
Related Works

- **SDSC myHadoop and Apache Hadoop-on-Demand (HOD)**
  - Dynamically in userspace
  - HOD requires access to static external HDFS cluster
  - Limited to reservation limitations
  - Does not provide pseudo-persistent interactive capability

- **FutureGrid and Amazon Elastic MapReduce Cloud**
  - Use virtualization
  - Degraded I/O performance for data-intensive applications

- **Mesos and Apache Hadoop YARN**
  - New resource management mechanisms
  - Handle Hadoop and non-Hadoop processes
  - Still in development
Conclusion

• Interactive pseudo-persistent MapReduce platform within the existing administrative structure of an academic high performance computing center

• As efficient as a persistent Hadoop cluster on dedicated computing resources, depending on the jump time

• Cluster remains stable, with good performance, in the presence of jumps that occur as frequently as the average length of reduce tasks
Questions?

W. Clay Moody, Linh B. Ngo, Edward Duffy, & Amy W. Apon
Computer Science Division of the School of Computing
Clemson Computing and Information Technology
Clemson University, Clemson, SC

https://sourceforge.net/projects/jummp/