MOON: MapReduce On Opportunistic eNvironments

Tudor Uricec
Outline

• Introduction
• Moon Design Rationale and Architecture
• Moon Data Management
• Moon Task Scheduling
• Performance Evaluation
• Conclusion
Introduction

• Problem solved

• Reason for solution

• MapReduce

• Hadoop
Problem solved

- In volunteer computing owners donate their computing resources to one or more projects.
- Running MapReduce on such systems results in poor performance.
- Writers propose MOON which extends Hadoop, using adaptive tasks and data scheduling algorithms.
- Moon can deliver a 3-fold performance improvement.
Reason for solution

- In a volunteer computing enterprise, the unavailability of nodes averages 0.4.
- In dedicated systems, it is much lower.

Fig. 1. Percentage of unavailable resources measured in a 7-day trace from a production volunteer computing system at San Diego Supercomputing Center [7]. The trace of each day was collected from 9:00AM to 5:00PM. The average percentage unavailability is measured in 10-minute intervals.
MapReduce

- Offers a flexible programming model for processing large sets of data

- Implements two primitives Map and Reduce
  - Map takes key-value -> intermediate key-value are applied Reduce -> Output key-value

- Also used in areas like machine learning, bioinformatics, astrophysics and cyber-security
Hadoop

• Is a popular, open-source MapReduce runtime system

• It was run on an emulated volunteer computing system and produced several problems

• Problems occurred because:
  – HDFS uses replication
  – Hadoop does not replicate intermediate results => livelock
  – Task scheduling assumes smooth task execution
Hadoop

• Two types of nodes: NameNode and DataNodes

• Replication degree is specified by a replication factor

• JobTracker runs on master and manages job status

• TaskTracker on each worker and tracks the available execution slots.(M,R)

• Tasks are prioritised
Outline

• Introduction

• **Moon Design Rationale and Architecture**

• Moon Data Management

• Moon Task Scheduling

• Performance Evaluation

• Conclusion
Moon Design Rationale and architecture

• Adopts a small number of dedicated computers

• These are used to improve availability through replication

• Long running tasks might not complete without dedicated nodes

• 99.99% availability requires 1:3 replicate ratio.

• Relies on the Hadoop heartbeat mechanism to detect PC unavailability

Fig. 2. Overview of MOON executing environments. The resources on nodes with a question mark are ephemeral.
Outline

• Introduction

• Moon Design Rationale and Architecture

• **Moon Data Management**

• Moon Task Scheduling

• Performance Evaluation

• Conclusion
Moon Data Management

• Three types of data: input, intermediate, output.

• To accomplish a cost-effective and robust storage service difference measures are taken:
  – Multi-dimensional, Cost-effective Replication Service
  – Prioritizing I/O Requests
  – Handling Ephemeral Unavailability
Multi-dimensional, cost-effective replication service

• Moon manages two types of resources

• Defines two type of workers: dedicated and volatile DataNodes

• Replication factor is defined by two numbers

• Data files are split into two categories: reliable and opportunistic
Multi-dimensional, cost-effective replication service

• Output data will first be stored as opportunistic and then reliable

• Important to control the load level of dedicated DataNodes and maximize utilization

• Unavailable dedicated nodes
  • =>
  • change of replication factor such that the availability level is met

• Replication factor not met, the file is placed in replication queue
Prioritizing I/O Requests

• It is necessary to alleviate read traffic on dedicated nodes

• Read requests are first issued on volatile nodes

• Write requests are prioritized to the dedicated nodes depending on the file type

• Reliable files are fulfilled prior to the opportunistic ones

Fig. 3. Decision process to determine where data should be stored.
Handling Ephemeral Unavailability

- Hadoop uses a NodeExpriyInterval variable to determine node failure

- This fault tolerant mechanism produces problems for opportunistic environments (variable vs mean unavailability)

- Moon introduces a hibernate state, new hibernate variable introduced

- A hibernate DataNode will not be supplied any I/O requests
Outline

• Introduction

• Moon Design Rationale and Architecture

• Moon Data Management

• **Moon Task Scheduling**

• Performance Evaluation

• Conclusion
Moon Task Scheduling

• Hadoop considers a task as a straggler if:
  – Runs for more than a minute
  – Its progress score is behind the average by 0.2 or more

• The JobTracker give priority to the ones with input data local to the TaskTracker

• Problems occur in opportunistic environments because:
  – Task do not run smoothly towards completion
  – Large percentage of tasks will be suspended
  – All instances of a task can be suspended
Moon Task Scheduling

• Moon implements three techniques:
  – Ensuring Sufficient Progress with High Node Volatility
  – Two-phase Task Replication
  – Leveraging the Hybrid Resources
Outline

• Introduction

• Moon Design Rationale and Architecture

• Moon Data Management

• Moon Task Scheduling

• Performance Evaluation

• Conclusion
Performance Evaluation

• Experimental Setup

• Speculative Task Scheduling Evaluation

• Replication of Intermediate Data

• Overall Performance Impacts of Moon
Experimental Setup

• Experiments were executed on System X at Virginia Tech, G5 compute nodes with dual 2.3 GHz PowerPC 970FX processors, 4GB of RAM, 80 GByte hard dives

• Ethernet Network was to simulate volunteer computing systems

• Each node was running the GNU/Linux OS

• Moon is developed based on Hadoop 0.17.2

• Experiments focus on two applications sort and word count

<table>
<thead>
<tr>
<th>Application</th>
<th>Input Size</th>
<th># Maps</th>
<th># Reduces</th>
</tr>
</thead>
<tbody>
<tr>
<td>sort</td>
<td>24 GB</td>
<td>384</td>
<td>0.9 x AvailSlots</td>
</tr>
<tr>
<td>word count</td>
<td>20 GB</td>
<td>320</td>
<td>20</td>
</tr>
</tbody>
</table>
Speculative Task Scheduling Evaluation

- Job response time is used as the performance metric.
- Moon intermediate data are set as reliable files such that data are always available for Reduce Tasks.
- 60 volatile nodes and 6 dedicated nodes.
- Different TrackerExpiryInterval for Hadoop.

![Graphs showing execution time for different rates and tasks.]
Speculative Task Scheduling Evaluation

• Another important metric to evaluate is the total number of duplicated tasks issued.

• The Hadoop Scheduler creates larger numbers of speculative tasks as a smaller TrackerExpiryInterval is used.

• Thus the cost of shortening this variable issues more speculative tasks.
Replication of Intermediate Data

- This section evaluates the impact of Moon’s intermediate data replication on shuffle efficiency

- 60 volatile nodes and 6 dedicated nodes

- Fixed replication factor \{1,3\}

- 61% better in sort, 32.5% in word count

- When three fetch failures are performed for a task, a new copy is immediately reissued
Overall Performance Impacts of Moon

- Hadoop is configured to store six replicas for both input and output data
- 60 volatile nodes and 3, 4, 6 dedicated nodes are used for the Moon tests
- Replication factor is set to \{1,1\}
- Best VO replication configuration used
Outline

• Introduction
• Moon Design Rationale and Architecture
• Moon Data Management
• Moon Task Scheduling
• Performance Evaluation
• Conclusion
Conclusions

• Moon is an adaptive system that supports MapReduce jobs on opportunistic environments, where existing MapReduce run-time policies fail to handle frequent node outages.

• In particular, the benefits of Moon’s data and task replication design to greatly improve the QoS of MapReduce when running on a volatile hybrid architecture.