A Case Study of MapReduce Speculation for Failure Recovery

Huansong Fu, Yue Zhu, Weikuan Yu
Florida State University

Presenter: Huansong Fu
Outline

• Background
• Motivation
  – Issues with existing speculation and breakdown of job performance
• Design
  – FARMS and FAS
• Experiments
• Conclusion
Outline

• Background

• Motivation
  – Issues with existing speculation and breakdown of job performance

• Design
  – FARMS and FAS

• Experiments

• Conclusion
Big Data Era

- Big data capacity is massive
  - The volume of the digital universe is stupendous.
  - The value of the big data has massive potential.

Source:
Hadoop MapReduce for Big Data

- Hadoop is the representative implementation of MapReduce.
  - Aim to expedite the big data processing.
  - Harness the computing power from commodity machines.
- It has many advantages in…
  - Scalability
  - Availability
  - Data locality
  - Programming model diversity
- Already with strong foundations, and still grows in popularity.
- It has been evolved into its next generation called YARN.
MapReduce Programming Model

- Simple but enables good distributed computing
- Mainly consists of the phase of *map* and *reduce*
  - Also has *sort*, *shuffle* and *merge*
The Next Generation Hadoop (YARN)

- **ResourceManager, NodeManager and AppMaster**
  - A global ResourceManager orchestrates all resources.
  - Per-node NodeManager reports to ResourceManager with local resource and status.
  - Per-job AppMaster requests resources for the job.
Fault Tolerance of YARN

• Received far less attention than other topics
  - Such as performance improvement, scheduling optimization, data availability, etc.

• Failures are norm rather than exception
  - 3% failure rate with jobs [1]
  - 5 node failures per job (a job has an average of 268 nodes) [2]
  - 8% annual failure rate with hardware [3]

• YARN suffers serious performance degradation from failures.
  - Speculation cannot eliminate negative impacts of failures.
  - It affects small jobs the most significantly, which are the majority of jobs that are in real use. [4]

References:
Outline

• Background
• Motivation
  – Issues with existing speculation and breakdown of job performance
• Design
  – FARMS and FAS
• Experiments
• Conclusion
Speculation Mechanism in YARN

- Although YARN has data replication and regeneration to ensure the availability of data upon failures, they cannot guarantee optimal job performance in the heterogeneous environment.
- Thus, YARN also has speculation mechanism to speed up the job turnaround time.
  - Basically, it is to make a copy of the slowest task (a.k.a. straggler) intermittently.
  - Any one copy of the straggler will let the job proceed.
- However, we found that the existing speculation has some major problems in the presence of failures, especially for small jobs.
  - Speculation does not guard against node failures very well.
  - Small jobs are more susceptible to a given slowdown.
Issues with Existing Speculation

• I. Intra-node only
  – When many tasks are converged on a single node, a node failure can make every task equally slow, and no task is regarded as straggler.

• II. Prospective only
  – If tasks are completed, they are excluded from speculation candidates.
The Performance Breakdown

• 1GB jobs are seriously slowed down
  – 0% ~ 50% has intra-node issue, 50% ~ 100% has prospective issue.
• 10GB jobs are less affected
  – Free from intra-node issue, but still suffer from prospective issue.

(a) 1 GB Wordcount job
(b) 10 GB Wordcount job
Can Lower Timeout Help?

• Intuitively, to decrease timeouts can help reduce failure penalty.
• But it also produces a lot of false negative decisions that can stall the job progress.
  – We conduct test to show that it is not feasible in the heterogeneous environment.
Proposed Solutions

• Make YARN failure-cognizant.
  – YARN does not know that failure occurs, let alone the countermeasures.

• A new speculation mechanism.
  – Launch speculative copies in batch upon node failures.
  – Take completed tasks for speculation as well.

• Enhanced scheduling upon failure.
  – Limit the false positive & false negative decisions on failure occurrence.
  – Schedule tasks wisely based on failure decision.
Outline

• Background
• Motivation
  – Issues with existing speculation and breakdown of job performance
• Design
  – FARMS and FAS
• Experiments
• Conclusion
Centralized Failure Analyzer (CFA)

- A new component in YARN that is responsible for globally collect and analysis of failure information.
  - It requests application info from the RM, e.g., job IDs, task IDs, container assignments, etc.
  - Node status is reported from NM via heartbeat to RM and then CFA.
  - The failure analytics results are supplied through HDFS to each AppMaster, which schedules tasks accordingly in a job.
  - The extra I/O is lightweight and incurs minimal overheads.
Optimized Speculation Mechanism

- **Failure-Aware, Retrospective and Multiplicative Speculation.** (FARMS)
  - Tasks are speculated upon the loss of node.
  - Completed tasks are speculated based on node status and fetch need.
  - Tasks are speculated in a multiplicative manner.
Workflow of FAS

- The Fast Analytics Scheduling (FAS) redesigns the scheduling policy in the presence of failures.
  - Use a dynamic threshold instead of fixed timeouts.
  - The threshold needs to be aggressive enough to gain performance improvement, but also conservative enough to adjust to environment.
Outline

• Background
• Motivation
  – Issues with existing speculation and breakdown of job performance
• Design
  – FARMS and FAS
• Experiments
• Conclusion
Experimental Setup

• Hardware Setup:
  – 21 server nodes featuring with four 2.67 GHZ hex-core Intel Xeon X5650 CPUs, 24GB memory and one 500GB hard disk.
  – Nodes are connected through 1 Gigabit Ethernet.

• Software Setup:
  – YARN version is 2.6.0.
  – One master node of the cluster is dedicated to run ResourceManager and NameNode.

• Benchmarks:
  – Terasort, WordCount, and Secondarysort.
Evaluation of FARMS

- Test against node failure.
- FARMS leverages the failure analytics information and provides much faster job recovery performance.
  - Smaller jobs benefits more than larger jobs.
  - Performance variations are almost eliminated.
Evaluation of FAS

- Test against an unstable cluster (network & node problems).
- FAS can adapt to heterogeneous environment well.
  - It provides near-original performance in the presence of network congestions & failures.
  - The false decision for speculation seldom occurs.

![Graph showing performance comparison between different methods]

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>Average execution time (s)</th>
<th>Additional speculated tasks rate per job</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>0.02</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>0.03</td>
</tr>
<tr>
<td>4</td>
<td>300</td>
<td>0.04</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0.05</td>
</tr>
</tbody>
</table>
Overall Performance

- Test a combination of benchmarks against node failure and networking problems.
- Test set is generated based on real-world production use.
  - In overall, we can achieve 15.3% performance improvement.
Conclusion

• Revealed the issues with the existing speculation mechanism in MapReduce framework.

• Demonstrated how those issues can cause the performance breakdown of MapReduce applications, especially for small jobs.

• Brought about a combination of techniques to solve the issues.

• Conducted experiments whose results show that our solution can achieve much better performance upon failures.
Sponsors of Our Research

[NSF logo]

[Intel logo]

[Auburn University logo]
Thank You and Questions?