Result Integrity Check for MapReduce Computation on Hybrid Clouds

Yongzhi Wang, Jinpeng Wei  Florida International University
Mudhakar Srivatsa  IBM T.J. Watson Research Center
Agenda

- Problem Statement
- System Design
- Theoretical Analysis
- Experiment Result
Agenda

- Problem Statement
- System Design
- Theoretical Analysis
- Experiment Result
MapReduce

- A highly parallel computing model designed for Big Data computation.
- Computation is performed on a cluster consists of a master and many slave workers.
- Data are stored and processed in \(<\text{key}, \text{value}\>\) tuples.
- One MapReduce job can be divided into map phase and reduce phase.
Word Count, a MapReduce Example
Problem

- How to provide high integrity MapReduce computing service on an untrusted public cloud.

Our Solution: Cross Cloud MapReduce (CCMR)

Trusted private cloud + Untrusted public cloud
Assumptions and Attacker model

- **Assumptions**
  - The private cloud is trusted.
  - The MapReduce storage (DFS) is trusted [5][6].
  - The tasks in MapReduce jobs are deterministic

- **Attacker Model**
  - Certain portion \((m)\) of workers are malicious \((0 \leq m \leq 1)\).
  - The malicious workers are controlled by an adversary.
  - The adversary directs malicious workers to collaborate with each other to inject as many errors as possible without being detected.

---

Agenda

- Problem Statement
- System Design
- Theoretical Analysis
- Experiment Result
System Design

- Hybrid cloud architecture
  - Trusted private cloud with verifiers
  - Untrusted public cloud
- Core Techniques
  - Two-layer check
  - Credit based trust management
- Different design for different phases.
Map Phase Integrity Check

- Two-layer check for each map task
  - Replication with replication probability $r$.
  - For task passed first-layer check, verification with verification probability $v$.
- Credit accumulation for each mapper
  - Increment credit of the worker.
  - Buffer the task result on the worker.
  - Accept results in batch only when worker achieves credit threshold $T$. 
Map Phase Integrity Check

Two-layer check for each map task
Credit accumulation for each mapper
Reduce Phase Integrity Check

- Straightforwardly applying aforementioned techniques to Reduce phase may encounter difficulties
  - In some job, reduce task number is smaller while the processed records in one task is huge. (e.g., word count: 1 reduce task, 2.7 M of records to be processed, 262 seconds to finish).
- We wish
  - To divide such reduce task into many sub-tasks.
  - To apply two-layer check and credit based trust management to each reduce sub-task.
Reduce Phase Integrity Check

Map 0 Output
- Apple, 1
- Banana, 1
- Cat, 2
- Egg, 2
- Fish, 2
- ... 7

Map 0 Key Table
- 0,5,9
- 10,6,10
- 21,3,7
- 29,3,7
- 37,4,8
- ... 1

Map 1 Output
- Apple, 2
- Cat, 1
- Driver, 1
- Good, 3
- ... 7

Map 1 Key Table
- 0,5,9
- 10,3,7
- 18,6,10
- 28,4,8
- ... 1

Reduce 0 Output
- Apple, 3
- Banana, 1
- Cat, 3
- Driver, 1
- Egg, 2
- Fish, 2
- Good, 3
- ... 3

Report 2:
- Driver, Good, hash#2

Report 1:
- Apple, Driver, hash#1

Sub-task 1
- YES

Replicate
- 6

Master
- 11

Verify
- 10

Report 1:
- Apple, Driver, hash#1
Reduce Phase Integrity Check
Agenda

- Problem Statement
- System Design
- Theoretical Analysis
- Experiment Result
Model the System

- Model map and reduce phase with same set of parameters
- Model the CCMR
  - T: Credit Threshold
  - r: Replication probability
  - v: Verification probability
- Model the Attacker
  - m: Malicious worker fraction
  - c: Cheat probability
- Measurement Metrics
  - Accuracy: Job Error Rate
    - Fraction of incorrect task/sub-task accepted by the master in one job.
  - Performance: Overhead & Verifier Overhead
    - For each task/sub-task, the expected number of extra executions performed on public/private cloud.
- Theoretical analysis conclusion is presented in Theorem 1.
Accuracy and Performance Trade Off

Job Error Rate vs Verifier Overhead
$m = 0.5$, $c = 0.1$, $\nu = 0.15$

- To decrease Job Error Rate
  - Increase $T$
  - A higher overhead
  - A higher verifier overhead
• What’s the upper bound of job error rate.
• When $m$ is 0.5, $r$ is 0.5, job error rate upper bound is less than 1%.
• When $m$ is 1.0, $r$ is 0.5, job error rate upper bound is 9%.
Agenda

- Problem Statement
- System Design
- Theoretical Analysis
- Experiment Result
Environment Setup

• **CCMR**
  • **Private Cloud:**
    • 1 Linux server (2.93 GHz, 8-core Intel Xeon CPU and 16 GB of RAM)
    • Running as a master and the verifier
  • **Public Cloud**
    • 12 Amazon EC2 micro instances (Amazon Linux AMI 32-bit, 613 MB memory, Shared ECU, Low I/O performance)
    • Running as 12 slave workers.

• **Baseline (Original MapReduce)**
  • 13 Amazon EC2 micro instances (Amazon Linux AMI 32-bit, 613 MB memory, Shared ECU, Low I/O performance)
  • Running as 1 master and 12 slave workers.
Experiment Applications

- **Word Count:**
  - Compute the frequency of each word in a batch of text files.
  - 800 text files as input, total input size as 653M.
  - 800 map task, 1 reduce task.

- **Mahout 20 news group classification:**
  - MapReduce implementation of Naïve Bayes classification
  - Classify news text files into 20 different categories.
  - 5 jobs in one application.
When $r$ is 0.3, job error rate ranges from 1.08% to 2.25%.
When $r$ is 1.0, job error rate ranges from 0.14% to 0%.
Map Phase (Performance)

Running time (s) of Word Count (Map Phase with CCMR) Compared with Baseline, \( m=0 \)

<table>
<thead>
<tr>
<th>( r )</th>
<th>Running time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>1728</td>
</tr>
<tr>
<td>0.5</td>
<td>2069</td>
</tr>
<tr>
<td>1.0</td>
<td>2323</td>
</tr>
<tr>
<td></td>
<td>3167</td>
</tr>
</tbody>
</table>

Extra execution times are 19%, 34% and 83%, when \( r \) increases from 0.3 to 1.0. The extra execution times are proportional to the replication probability \( r \).
Reduce Phase (Performance)

Running time (s) of Mahout 20 Class. & Word Count (Reduce phase with CCMR) compared with baseline

\[ m = 0, \ r = 0.167, \ v = 0.07, \ T = 600 \]

<table>
<thead>
<tr>
<th></th>
<th>Baseline Execution time (s)</th>
<th>CCMR Execution time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mahout 20 news Group Classification</td>
<td>1304</td>
<td>1892</td>
</tr>
<tr>
<td>Word Count</td>
<td>979</td>
<td>1398</td>
</tr>
</tbody>
</table>

Extra execution time are 45% in Mahout classification, 43% in word count application. The extra execution time are attributed to the vast number of sub-tasks for replication and verification. E.g., Word Count application consists of 88 replication sub-tasks and 6 verification sub-tasks.
Conclusion and Future Work

- Cross Cloud MapReduce (CCMR), a hybrid cloud MapReduce framework is proposed.
- By utilizing the trusting base gained from private cloud, a high integrity assurance MapReduce service can be achieved with majority of computation performed on public cloud.
- Reduce performance overhead and reasoning the optimal system parameter would be the next step work.
Thank you!