DREAMS: Dynamic Resource Allocation for MapReduce with Data Skew

Zhihong Liu**, Qi Zhang†, Mohamed Faten Zhani‡, Raouf Boutaba†, Yaping Liu* and Zhenghu Gong*

*National University of Defense Technology, China
†University of Waterloo, Canada
Outline

• Introduction

• Our solution

• Evaluation

• Conclusion
Introduction
Introduction

How much data is created every minute?

4,000,000 search queries

2,460,000 pieces of new content are shared

72 hours of new videos are uploaded

Introduction

- MapReduce is a popular framework for big data analytics

- Data skew in MapReduce

Hash (Key) mod Num. ofReducers

Map Stage

Reduce Stage
Introduction

- Resource management schemes in Hadoop
  - Slot-based
  - Container-based

- Limitations
  - Assume the same kind of tasks (map or reduce) in a job has uniform resource requirement
  - Do not support dynamic resource allocation to each task

Incur

1) Prolonging the job completion time
2) Reducing the resource utilization
Introduction

- Existing solutions
  - Rebalance the key-value pairs among reduce tasks based on the key distribution
    - cause a synchronization barrier
  - Run speculative tasks on other machines
    - may waste resource while omitting the correlation between task load and progress rate
  - Repartition the unprocessed load of slow tasks to another tasks
    - incur large overhead to repartition the load
Our solution
DREAMS

**Idea**: Dynamically adjusting the container size based on the load of each reduce task, thereby mitigating the negative impact of data skew.

**Benefits**:
- Eliminates the overhead of rebalancing the load
- Mitigates data skew at run-time
- Simple to implement

**Limitation**:
- Needs job profiles
Challenges

• How to predict the load of each reduce task at run-time?

• How much amount of resources should be allocated to each reduce task?
Challenge One
How to predict the load of each reduce task

- Using linear regression

- \( F^j \) is the percentage of map tasks that have completed

- \( S_i^j \) is the size of the partitions generated by the completed map tasks for reduce task \( i \)

- Once a threshold \( \delta \) (e.g. 5\%) is reached, we finalize the linear model.

InvertedIndex on Wikipedia dataset

\[ \text{Load of the reduce task} \]

\[ \text{InvertedIndex on Wikipedia dataset} \]

\[ \text{Fraction of map tasks (\%)} \]

\[ \text{size of partitions (MB)} \]
Challenge Two
How much resource should be allocated?

Task duration \( = f(\text{Task load, Amount of resource}) \)

- We need to know:
  - What is the relationship between the task duration and the task load?
  - What is the relationship between the task duration and the resource allocation?
The relationship between task duration and task load

The task duration is linearly correlated with the task load

(a) InvertedIndex 10G
(b) InvertedIndex 10 and 20G

The task duration is linearly correlated with the task load
The relationship between task duration and CPU allocation is inverse proportionally correlated with the CPU allocation.
The relationship between task duration and memory

(a) Sort10G
(b) InvertedIndex 10G

Memory is not the bottleneck resource for this workload
Reduce task performance model

\[ T_i = \begin{cases} 
\alpha + \beta P_i + \gamma D + \frac{\zeta}{\text{Alloc}_{i}^{\text{cpu}}} + \frac{\eta P_i}{\text{Alloc}_{i}^{\text{cpu}}} + \frac{\xi D}{\text{Alloc}_{i}^{\text{cpu}}} & \text{if } \text{Alloc}_{i}^{\text{cpu}} \leq \varphi \\
\alpha' + \beta' P_i + \gamma' D + \frac{\zeta'}{\text{Alloc}_{i}^{\text{cpu}}} + \frac{\eta' P_i}{\text{Alloc}_{i}^{\text{cpu}}} + \frac{\xi' D}{\text{Alloc}_{i}^{\text{cpu}}} & \text{if } \text{Alloc}_{i}^{\text{cpu}} > \varphi 
\end{cases} \]

Task duration \[= f(\text{Task load}, \text{Amount of resource}) \]

<table>
<thead>
<tr>
<th>(T_i)</th>
<th>task duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_i)</td>
<td>task load</td>
</tr>
<tr>
<td>(D)</td>
<td>sum of all reduce loads</td>
</tr>
<tr>
<td>(\text{Alloc}_{i}^{\text{cpu}})</td>
<td>CPU allocation</td>
</tr>
</tbody>
</table>

- Use non linear regression to determine the coefficient factors
- Each tuple of \((T_i, P_i, D, \text{Alloc}_{i}^{\text{cpu}})\) is a training data
- This performance model is used as a **job profile** for allocating resource
Architecture of DREAMS

NodeManager

Map 1
Map 2
...
P1 Pn...P2
P1 Pn...P2

Application Master

Task Duration Estimator

Resource Allocator

Partition Size Predictor

Resource Manager

Fine-grained Container Scheduler

Partition Stats Report

Resource Request

Resource Response

Container Launch

Job Profile

NodeManager

Partition Size Monitor

Map 1
Map 2
...
P1 P2 ... Pn

NodeManager

Partition Size Monitor

Map 3
Map 4
...
P1 P2 ... Pn

NodeManager

Partition Size Monitor

Map 5
Map 6
...
P1 P2 ... Pn

Partition Size Monitor
Evaluation
Evaluation

- Accuracy of reduce task load prediction
  - Metric
    
    $$ARE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{P_{i}^{pred} - P_{i}^{measrd}}{P_{i}^{measrd}} \right|$$

- Results
  - Different datasets
  - Different *slowstart* settings

<table>
<thead>
<tr>
<th>APP</th>
<th>Input Size(GB)</th>
<th>$\delta = 0.05$</th>
<th>$\delta = 0.06$</th>
<th>$\delta = 0.07$</th>
<th>$\delta = 0.08$</th>
<th>$\delta = 0.09$</th>
<th>$\delta = 0.10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort</td>
<td>10</td>
<td>2.28%</td>
<td>2.09%</td>
<td>1.94%</td>
<td>1.81%</td>
<td>1.71%</td>
<td>1.71%</td>
</tr>
<tr>
<td>Sort</td>
<td>20</td>
<td>1.60%</td>
<td>1.43%</td>
<td>1.32%</td>
<td>1.26%</td>
<td>1.17%</td>
<td>1.13%</td>
</tr>
<tr>
<td>Sort</td>
<td>50</td>
<td>1.1%</td>
<td>1.01%</td>
<td>0.94%</td>
<td>0.90%</td>
<td>0.84%</td>
<td>0.78%</td>
</tr>
<tr>
<td>lvIndex</td>
<td>9.01</td>
<td>8.2%</td>
<td>7.63%</td>
<td>7.05%</td>
<td>7.05%</td>
<td>6.43%</td>
<td>5.87%</td>
</tr>
<tr>
<td>lvIndex</td>
<td>21.02</td>
<td>5.62%</td>
<td>5.25%</td>
<td>5.08%</td>
<td>4.79%</td>
<td>4.53%</td>
<td>4.38%</td>
</tr>
<tr>
<td>lvIndex</td>
<td>49.04</td>
<td>4.73%</td>
<td>4.43%</td>
<td>4.21%</td>
<td>4.07%</td>
<td>3.90%</td>
<td>3.70%</td>
</tr>
</tbody>
</table>
Accuracy of reduce task performance model

**Metric**

\[
ARE = \frac{1}{k} \sum_{l=1}^{k} \left| \frac{T_{l}^{\text{pred}} - T_{l}^{\text{measrd}}}{T_{l}^{\text{measrd}}} \right|
\]

**Results**

<table>
<thead>
<tr>
<th>Application</th>
<th>Input Data Type</th>
<th>Input Data Size(GB)</th>
<th>Test-on-training</th>
<th>Test-on-unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort</td>
<td>Synthetic</td>
<td>10</td>
<td>5.44%</td>
<td>9.36%</td>
</tr>
<tr>
<td>Sort</td>
<td>Synthetic</td>
<td>20</td>
<td>7.91%</td>
<td>10.62%</td>
</tr>
<tr>
<td>Sort</td>
<td>Synthetic</td>
<td>30</td>
<td>12.28%</td>
<td>16.38%</td>
</tr>
<tr>
<td>Sort</td>
<td>Synthetic</td>
<td>50</td>
<td>11.09%</td>
<td>19.57%</td>
</tr>
<tr>
<td>InvertedIndex</td>
<td>Wikipedia</td>
<td>9.01</td>
<td>11.67%</td>
<td>13.97%</td>
</tr>
<tr>
<td>InvertedIndex</td>
<td>Wikipedia</td>
<td>21.02</td>
<td>12.89%</td>
<td>13.31%</td>
</tr>
<tr>
<td>InvertedIndex</td>
<td>Wikipedia</td>
<td>31.03</td>
<td>14.67%</td>
<td>16.44%</td>
</tr>
<tr>
<td>InvertedIndex</td>
<td>Wikipedia</td>
<td>49.04</td>
<td>14.56%</td>
<td>17.06%</td>
</tr>
</tbody>
</table>

Different datasets

Two kinds of validations
Job performance evaluation

20.3% speedup

(a) Sort

(b) InvertedIndex
Task execution timeline

The straggling tasks prolong the job completion

(a) Sorting 10G with Native

(b) Sorting 10G with DREAMS
Resource utilization

(a) Sorting 10G with Native

(b) Sorting 10G with DREAMS

Better utilization
Conclusion
Conclusion

- We present DREAMS, a framework that mitigates the data skew for MapReduce by **adjusting the container size** at run-time

**Our contributions**

- We develop an partition size prediction model
  - Perform at run-time
  - The error rate is less than 8.2%

- We design a reduce task performance model
  - The worst error rate is 19.57%

- We demonstrate the benefits of leveraging resource-awareness for data skew mitigation
  - Eliminate the overhead of rebalancing the load
  - Improve the job running time by up to 20.3%
Thank you 😊

Questions?