Wrangler
Predictable and Faster Jobs using Fewer Resources

Neeraja J. Yadwadkar,
Ganesh Ananthanarayanan and Randy Katz
Parallel Data Analytics

Job queue

Master

Slaves

Job completed
Job completed
Defining Stragglers

Tasks of a job

Median

Threshold

Task Execution Time
## Impact of Stragglers

### Presence of Stragglers in real-world production level traces

When replayed using SWIM\(^+\) on a 50 node EC2 cluster….

<table>
<thead>
<tr>
<th>Workload</th>
<th>Stragglers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook 2009 (FB2009)</td>
<td>24 %</td>
</tr>
<tr>
<td>Facebook 2010 (FB2010)</td>
<td>23 %</td>
</tr>
<tr>
<td>Cloudera’s Customer b (CC(_b))</td>
<td>28 %</td>
</tr>
<tr>
<td>Cloudera’s Customer e (CC(_e))</td>
<td>22 %</td>
</tr>
</tbody>
</table>

\(^+\)Chen Y., et al., The Case for Evaluating MapReduce Performance Using Workload Suites, MASCOTS’11

\(*\)captured for over 6 months from about 4000 machines in total

\(\text{Threshold} = 1.3 \times \text{median}\)
Impact of Stragglers: We measure the potential in speeding up jobs in the trace using the following crude analysis: replace the progress rate of every task of a phase that is slower than the median task with the median task’s rate. If this were to happen, the average completion time of jobs improves by 47%, 29% and 36% in the Facebook, Bing and Yahoo! traces, respectively; small jobs (≤ 10 tasks) improve by 49%, 38% and 41%.

Dolly, NSDI’13
Speculative Execution

Job queue

Replicating

Job completed

Wasted Resources

Wasted Time in detecting stragglers

$T_s$: In progress
Existing Approaches

- Wasted Resources
- Wasted Time in detecting stragglers

Speculative Execution
- LATE
- Mantri
- Dolly
- Wrangler

Replicate

OSDI’04
OSDI’08
NSDI’13
OSDI’10
NSDI’13
Design Goals

1. Identify stragglers as early as possible
   - Avoid Wasting Time in detecting stragglers

2. Schedule tasks for improved job finishing times
   1. To avoiding creation of stragglers
   2. To avoid replication
      - Avoid Wasting Resources
Defer potential stragglers

Reduce

Map\textsubscript{3}

Map\textsubscript{2}

Map\textsubscript{1}

Job Submitted

Time

Map finished

Map finished

Job finished

Net Gain

Deferred Assignments

Reduce Deferred Assignments

Time

Map finished

Map finished

Job finished

Job finished
Our proposal: Wrangler

"Input" Features

Model Builder ➔ Predictive Scheduler

Workers

Scheduling Decisions
Selecting “Input” Features

<table>
<thead>
<tr>
<th>CPU</th>
<th>Memory</th>
<th>Disk</th>
<th>Network</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu_idle</td>
<td>mem_buffers</td>
<td>swap_free</td>
<td>bytes_in</td>
<td>Jvm. memHeapUsedM</td>
</tr>
<tr>
<td>cpu_system</td>
<td>mem_cached</td>
<td>swap_total</td>
<td>bytes_out</td>
<td>Jvm. threadsBlocked</td>
</tr>
<tr>
<td>cpu_user</td>
<td>mem_free</td>
<td>disk_free</td>
<td></td>
<td>Jvm. gcTimeMillis</td>
</tr>
<tr>
<td></td>
<td>mem_shared</td>
<td>disk_total</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>mem_total</td>
<td></td>
<td></td>
<td>Jvm. threadsBlocked</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>datanode.blocks_written</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rpc.callQueueLen</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rpc.OpenConnections</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

MapReduce: Dean, et. al, OSDI’04

LATE: Zaharia, et. al, OSDI’08

Mantri: Ananthanarayanan, et. al, OSDI’10

Note: The text contains additional footnotes and references, which are not directly transcribed here.
Selecting “Input” Features

Key observation using Subset selection methods

Features of importance vary across nodes and across time!

Why?

• Complex task-to-node interactions
• Complex task-to-task interactions
• Heterogeneous clusters
• Heterogeneous task requirements
Our proposal: Wrangler

“Input” Features

Workers

Model Builder

Predictive Scheduler

Scheduling Decisions

Worker 1
Our proposal: **Wrangler**

Master

- Model Builder
- Predictive Scheduler

Workers

Utilization Counters

Scheduling Decisions
Our proposal: **Wrangler**

![Diagram of Wrangler system](image)

- **Model Builder**
- **Predictive Scheduler**
- **Workers**
  - **Worker 1**
  - **Utilization Counters**
  - **Scheduling Decisions**
A Challenge in Model Building

Unknown and dynamic causes behind stragglers

Desired: An ability of predicting stragglers without needing domain-specific knowledge

Approach: Classification!
Techniques that learn to adapt to changing causes automatically
Classification for Predicting Stragglers

For every node in a cluster,

Build a model:

\[\{\text{Utilization Counters, Straggler/Non-Straggler}\} \rightarrow \text{Learning} \rightarrow \text{Classifier}\]

Predict:

\[\{\text{Utilization Counters}\} \sim 100 \rightarrow \text{Classifier} \rightarrow \begin{array}{c} \text{Straggler} \\ \text{Non-Straggler} \end{array}\]
Classification Technique: SVM

Non-Stragglers

Stragglers

Separating Hyper-plane

feature_2

feature_1
Classification Accuracy using SVM

Is this accuracy good enough?
Our proposal: **Wrangler**

![Diagram](Diagram.png)

The diagram illustrates the flow of data and processes within the Wrangler system. It shows the interaction between the Master and Workers, with the Model Builder generating data that flows to the Predictive Scheduler. The Utilization Counters provide feedback to the Model Builder, and the Scheduling Decisions are updated based on the feedback from the Workers.

**Master**
- Model Builder
- Predictive Scheduler

**Workers**
- Utilization Counters
- Scheduling Decisions

**Model Builder**
- Generates data

**Predictive Scheduler**
- Makes scheduling decisions

**Workers**
- Provide utilization counters

**Utilization Counters**
- Feed back to Model Builder

**Scheduling Decisions**
- Updated based on feedback from Workers
Predictive Scheduler (Naïve)

Defer scheduling a task on a node that is predicted to create a straggler
Intuition....

Key: Better load-balancing

1. Improved job completion
2. Reduced resource consumption
Does the Naïve approach improve job completion?

Workload: CC_b

Percentage Reduction

Baseline: Speculative Execution
Our proposal: **Wrangler**

Master

- Model Builder
- Confident?
- Predictive Scheduler

Workers

- Utilization Counters

Model Builder -> Confident?

Confident? Yes -> Predictive Scheduler

Predictive Scheduler -> Scheduling Decisions

Scheduling Decisions -> Workers

Utilization Counters -> Worker 1

Worker 1
Our proposal: Wrangler

Only confident predictions influence scheduling decisions
Confidence Measure

Non-Stragglers

Stragglers

Separating Hyper-plane

Corresponds to $P$: confidence threshold

Low-confidence zone
$p$: Confidence Threshold

- Value calculated via cross-validation
- Verified via empirical sensitivity analysis

(refer to paper for details)
Evaluation

• Aim I: Does Wrangler Improve Job Completion Times?

• Aim II: Does Wrangler Reduce Resources Consumed?
Aim 1: Does Wrangler Improve Job Completion Times?

Workload: CC_b

Baseline: Speculative Execution

Confidence measure is the key!

<table>
<thead>
<tr>
<th>Percentage Reduction</th>
<th>Prediction without confidence measure</th>
<th>Prediction with confidence measure (p=0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg</td>
<td>43.59</td>
<td>43.13</td>
</tr>
<tr>
<td>95p</td>
<td>58.87</td>
<td></td>
</tr>
<tr>
<td>97p</td>
<td>62.10</td>
<td></td>
</tr>
<tr>
<td>99p</td>
<td>11.34</td>
<td></td>
</tr>
<tr>
<td>99.9p</td>
<td>22.46</td>
<td></td>
</tr>
</tbody>
</table>
Aim II: Does Wrangler Reduce Resources Consumed?

Percentage Reduction in Total Task-Seconds

Baseline: Speculative Execution

- FB2009: 55.09%
- FB2010: 24.77%
- CC_b: 40.15%
- CC_e: 8.24%
Load-Balancing with Wrangler

Workload: FB2010

Without Wrangler

With Wrangler ($p=0.7$)

Few highly loaded nodes
Sophisticated schedulers exist....why Wrangler?

• Difficult to
  – Anticipate the dynamically changing causes
  – Deal with over 100 resource utilization counters
  – Build a generic and unbiased scheduler

Wrangler statistically learns to predict **stragglers** without needing domain-specific knowledge
Status and Future Directions

• Status:
  – Improving accuracy using lesser training data
  – Improving job completions further

• Future Directions:
  – Generalization to other problems in cluster computing environments
    • Fault Detection
    • Resource Allocation
Conclusions

• ML techniques enable early prediction of tasks that are likely to straggle.

• Use of confidence measure makes straggler predictions useful

• Wrangler:
  – Enables faster jobs
  – Uses fewer resources

Contact: neerajay@eecs.berkeley.edu