Jockey
Guaranteed Job Latency in Data Parallel Clusters

Andrew Ferguson, Peter Bodik, Srikanth Kandula, Eric Boutin, and Rodrigo Fonseca
DATA PARALLEL CLUSTERS
DATA PARALLEL CLUSTERS
DATA PARALLEL CLUSTERS
DATA PARALLEL CLUSTERS

Deadline
DATA PARALLEL CLUSTERS
VARIABLE LATENCY
VARIABLE LATENCY
VARIABLE LATENCY
VARIABLE LATENCY
VARIABLE LATENCY
VARIABLE LATENCY
VARIABLE LATENCY
VARIABLE LATENCY

4.3x
Why does latency vary?

1. Pipeline complexity
2. Noisy execution environment
Cosmos

MICROSOFT’S DATA PARALLEL CLUSTERS
• CosmosStore

Cosmos
MICROSOFT’S DATA PARALLEL CLUSTERS
• CosmosStore
• Dryad

Cosmos
MICROSOFT’S DATA PARALLEL CLUSTERS
• CosmosStore
• Dryad
• SCOPE

Cosmos
MICROSOFT’S DATA PARALLEL CLUSTERS
• CosmosStore
• Dryad
• SCOPE

Cosmos
MICROSOFT’S DATA PARALLEL CLUSTERS
DRYAD’S DAG WORKFLOW
DRYAD’S DAG WORKFLOW

Cosmos Cluster
DRYAD’S DAG WORKFLOW
DRYAD’S DAG WORKFLOW
DRYAD’S DAG WORKFLOW
DRYAD’S DAG WORKFLOW
DRYAD’S DAG WORKFLOW
Priorities?

EXPRESSING PERFORMANCE TARGETS
Priorities? Not expressive enough
Weights?

EXPRESSING PERFORMANCE TARGETS
Priorities? Not expressive enough
Weights? Difficult for users to set
Utility curves?

EXPRESSING PERFORMANCE TARGETS
EXPRESSING PERFORMANCE TARGETS

- Priorities? Not expressive enough
- Weights? Difficult for users to set
- Utility curves? Capture deadline & penalty
OUR GOAL
Maximize utility

OUR GOAL
Maximize utility while minimizing resources

OUR GOAL
Maximize utility while minimizing resources by dynamically adjusting the allocation

OUR GOAL
Jockey
• Large clusters

Jockey
• Large clusters
• Many users

Jockey
• Large clusters
• Many users
• Prior execution

Jockey
\[ f(\text{job state}, \text{allocation}) \rightarrow \text{remaining run time} \]
\( f(\text{job state}, \quad ) \rightarrow \)
\[ f(\text{job state, allocation}) \rightarrow \]
\[ f(\text{job state, allocation}) \rightarrow \text{remaining run time} \]
JOCKEY – CONTROL LOOP
JOCKEY – CONTROL LOOP
Jockey – Control Loop
\[ f(\text{job state, allocation}) \rightarrow \text{remaining run time} \]
\[ f(\text{job state, allocation}) \rightarrow \text{remaining run time} \]

\[ f(\text{progress, allocation}) \rightarrow \text{remaining run time} \]

**Jockey – Model**
JOCKEY – PROGRESS INDICATOR
Jockey – Progress Indicator
JOCKEY – PROGRESS INDICATOR
total running

JOCKEY – PROGRESS INDICATOR
total running
+
total queuing

JOCKEY – PROGRESS INDICATOR
JOCKEY – PROGRESS INDICATOR

stage

total running +
total queuing
JOCKEY – PROGRESS INDICATOR

Stage 1

\[
\text{total running} + \text{total queuing}
\]

Stage 2

\[
\text{total running} + \text{total queuing}
\]

Stage 3

\[
\text{total running} + \text{total queuing}
\]
Jockey – Progress Indicator

Stage 1

# complete total tasks

total running + total queuing

Stage 2

# complete total tasks

total running + total queuing

Stage 3

# complete total tasks

total running + total queuing
JOCKEY – PROGRESS INDICATOR
JOCKEY – PROGRESS INDICATOR
JOCKEY – PROGRESS INDICATOR
JOCKEY – PROGRESS INDICATOR
Jockey – Progress Indicator
Jockey – Control Loop
## Job model

<table>
<thead>
<tr>
<th>1% complete</th>
<th>2% complete</th>
<th>3% complete</th>
<th>4% complete</th>
<th>5% complete</th>
</tr>
</thead>
</table>

**Jockey – Control Loop**
### Job model

<table>
<thead>
<tr>
<th></th>
<th>10 nodes</th>
<th>20 nodes</th>
<th>30 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1% complete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2% complete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3% complete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4% complete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5% complete</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Jockey – Control Loop**
## Job model

<table>
<thead>
<tr>
<th></th>
<th>10 nodes</th>
<th>20 nodes</th>
<th>30 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1% complete</td>
<td>60 minutes</td>
<td>40 minutes</td>
<td>25 minutes</td>
</tr>
<tr>
<td>2% complete</td>
<td>59 minutes</td>
<td>39 minutes</td>
<td>24 minutes</td>
</tr>
<tr>
<td>3% complete</td>
<td>58 minutes</td>
<td>37 minutes</td>
<td>22 minutes</td>
</tr>
<tr>
<td>4% complete</td>
<td>56 minutes</td>
<td>36 minutes</td>
<td>21 minutes</td>
</tr>
<tr>
<td>5% complete</td>
<td>54 minutes</td>
<td>34 minutes</td>
<td>20 minutes</td>
</tr>
</tbody>
</table>
## Job model

<table>
<thead>
<tr>
<th></th>
<th>10 nodes</th>
<th>20 nodes</th>
<th>30 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1% complete</td>
<td>60 minutes</td>
<td>40 minutes</td>
<td>25 minutes</td>
</tr>
<tr>
<td>2% complete</td>
<td>59 minutes</td>
<td>39 minutes</td>
<td>24 minutes</td>
</tr>
<tr>
<td>3% complete</td>
<td>58 minutes</td>
<td>37 minutes</td>
<td>22 minutes</td>
</tr>
<tr>
<td>4% complete</td>
<td>56 minutes</td>
<td>36 minutes</td>
<td>21 minutes</td>
</tr>
<tr>
<td>5% complete</td>
<td>54 minutes</td>
<td>34 minutes</td>
<td>20 minutes</td>
</tr>
</tbody>
</table>

**Completion:**
1%

**Deadline:**
50 min.
## Job model

<table>
<thead>
<tr>
<th></th>
<th>10 nodes</th>
<th>20 nodes</th>
<th>30 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1% complete</td>
<td>60 minutes</td>
<td>40 minutes</td>
<td>25 minutes</td>
</tr>
<tr>
<td>2% complete</td>
<td>59 minutes</td>
<td>39 minutes</td>
<td>24 minutes</td>
</tr>
<tr>
<td>3% complete</td>
<td>58 minutes</td>
<td>37 minutes</td>
<td>22 minutes</td>
</tr>
<tr>
<td>4% complete</td>
<td>56 minutes</td>
<td>36 minutes</td>
<td>21 minutes</td>
</tr>
<tr>
<td>5% complete</td>
<td>54 minutes</td>
<td>34 minutes</td>
<td>20 minutes</td>
</tr>
</tbody>
</table>

**Completion:**
1%

**Deadline:**
50 min.

**Jockey – Control Loop**
## Job model

<table>
<thead>
<tr>
<th>Completion</th>
<th>10 nodes</th>
<th>20 nodes</th>
<th>30 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1% complete</td>
<td>60 minutes</td>
<td>40 minutes</td>
<td>25 minutes</td>
</tr>
<tr>
<td>2% complete</td>
<td>59 minutes</td>
<td>39 minutes</td>
<td>24 minutes</td>
</tr>
<tr>
<td>3% complete</td>
<td>58 minutes</td>
<td>37 minutes</td>
<td>22 minutes</td>
</tr>
<tr>
<td>4% complete</td>
<td>56 minutes</td>
<td>36 minutes</td>
<td>21 minutes</td>
</tr>
<tr>
<td>5% complete</td>
<td>54 minutes</td>
<td>34 minutes</td>
<td>20 minutes</td>
</tr>
</tbody>
</table>

**Completion:** 3%

**Deadline:** 40 min.

**Jockey – Control Loop**
### Job model

<table>
<thead>
<tr>
<th></th>
<th>10 nodes</th>
<th>20 nodes</th>
<th>30 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1% complete</td>
<td>60 minutes</td>
<td>40 minutes</td>
<td>25 minutes</td>
</tr>
<tr>
<td>2% complete</td>
<td>59 minutes</td>
<td>39 minutes</td>
<td>24 minutes</td>
</tr>
<tr>
<td>3% complete</td>
<td>58 minutes</td>
<td>37 minutes</td>
<td>22 minutes</td>
</tr>
<tr>
<td>4% complete</td>
<td>56 minutes</td>
<td>36 minutes</td>
<td>21 minutes</td>
</tr>
<tr>
<td>5% complete</td>
<td>54 minutes</td>
<td>34 minutes</td>
<td>20 minutes</td>
</tr>
</tbody>
</table>

**Completion:** 5%

**Deadline:** 30 min.

**Jockey – Control Loop**
f(progress, allocation) -> remaining run time

JOCKEY – MODEL
$f(\text{progress, allocation}) \rightarrow \text{remaining run time}$

analytic model?

**Jockey – Model**
\( f(\text{progress, allocation}) \rightarrow \text{remaining run time} \)

analytic model

machine learning?

JOCKEY – MODEL
\( f(\text{progress, allocation}) \rightarrow \text{remaining run time} \)

- analytic model
- machine learning
- simulator

**JOCKEY - MODEL**
<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
</table>

JOCKEY
<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline complexity</td>
<td></td>
</tr>
</tbody>
</table>

**JOCKEY**
<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline complexity</td>
<td>Use a simulator</td>
</tr>
<tr>
<td>Problem</td>
<td>Solution</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Pipeline complexity</td>
<td>Use a simulator</td>
</tr>
<tr>
<td>Noisy environment</td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Solution</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Pipeline complexity</td>
<td>Use a simulator</td>
</tr>
<tr>
<td>Noisy environment</td>
<td>Dynamic control</td>
</tr>
</tbody>
</table>

**Jockey**
Jockey in Action
• Real job

Jockey in Action
• Real job
• Production cluster

Jockey in Action
• Real job
• Production cluster
• CPU load: ~80%

Jockey in Action
Jockey in Action
Jockey in Action
Jockey in Action
Jockey in Action

Initial deadline: 140 minutes
New deadline: 70 minutes
Jockey in Action

Release resources due to excess pessimism

New deadline: 70 minutes

allocation

0 10 20 30 40 50 60 70
0 12 24 36 48 60

time

New deadline: 70 minutes

allocation

# running
Jockey in Action

"Oracle" allocation:
Total allocation-hours
Deadline

<table>
<thead>
<tr>
<th>allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>36</td>
</tr>
<tr>
<td>24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

[Graph showing allocation over time with labels]

- allocation
- # running
- oracle tokens
Jockey in Action

“Oracle” allocation:
Total allocation-hours
Deadline

Available parallelism less than allocation

allocation

# running
oracle tokens
Jockey in Action

“Oracle” allocation:
- Total allocation-hours
- Deadline

Allocation above oracle

Time

Allocation

Oracle tokens

# running

Jockey in Action

93
Evaluation
• Production cluster

Evaluation
• Production cluster
• 21 jobs

Evaluation
• Production cluster
• 21 jobs
• SLO met?

Evaluation
Evaluation

- Production cluster
- 21 jobs
- SLO met?
- Cluster impact?
Evaluation
Evaluation

Jobs which met the SLO
Jobs which met the SLO
Evaluation

Missed 1 of 94 deadlines

Jobs which met the SLO
Jobs which met the SLO
Evaluation

CDF

job completion time relative to deadline

100%
80%
60%
40%
20%
0%

10% 20% 30% 40% 50% 60% 70% 80% 90% 100% 110% 120% 130%

Jockey
deadline

Jobs which met the SLO
Evaluation

Jobs which met the SLO

Job completion time relative to deadline

1.4x

Jockey

deadline

CDF

0%

10%

20%

30%

40%

50%

60%

70%

80%

90%

100%
Evaluation

Missed 1 of 94 deadlines

Allocated too many resources

Jobs which met the SLO
Evaluation

Simulator made good predictions:
80% finish before deadline

Allocated too many resources

Jobs which met the SLO

Missed 1 of 94 deadlines

CDF

job completion time relative to deadline

max allocation

Allocation from simulator

Jockey

deadline

Simulator made good predictions:
80% finish before deadline

Allocated too many resources

Jobs which met the SLO

Missed 1 of 94 deadlines
Simulator made good predictions: 80% finish before deadline

Jobs which met the SLO

Control loop is stable and successful

Control loop only

Missed 1 of 94 deadlines

Allocated too many resources
Evaluation
Evaluation

fraction of allocation above oracle
Evaluation

fraction of deadlines missed

fraction of allocation above oracle
Evaluation

fraction of deadlines missed

fraction of allocation above oracle

Jockey

max allocation
Evaluation

- Allocation from simulator
- Control loop only

Fraction of deadlines missed vs. fraction of allocation above oracle.

- Jockey
- Max allocation

Graph shows a comparison between allocation strategies, with the y-axis representing the fraction of deadlines missed and the x-axis showing the fraction of allocation above the oracle.
Conclusion
Data parallel jobs are complex,
Data parallel jobs are complex, yet users demand deadlines.
Data parallel jobs are complex, yet users demand deadlines. Jobs run in shared, noisy clusters,
Data parallel jobs are complex,

yet users demand deadlines.

Jobs run in shared, noisy clusters,

making simple models inaccurate.
Jockey
simulator
control-loop
Questions?

Andrew Ferguson
adf@cs.brown.edu
Questions?

Andrew Ferguson
adf@cs.brown.edu

Co-authors

- Peter Bodík (Microsoft Research)
- Srikanth Kandula (Microsoft Research)
- Eric Boutin (Microsoft)
- Rodrigo Fonseca (Brown)

Questions?

Andrew Ferguson
adf@cs.brown.edu
Backup Slides
Utility Curves

For single jobs, scale doesn’t matter

For multiple jobs, use financial penalties
Jockey

Resource allocation control loop

\[ U_a = U(t_r + C(p, a)) \]

\[ A^r = \arg \min_a \{ a : U_a = \max_b U_b \} \]

1. Slack
2. Hysteresis
3. Dead Zone
Cosmos

Resource sharing in Cosmos

- Resources are allocated with a form of fair sharing across business groups and their jobs. (Like Hadoop FairScheduler or CapacityScheduler)
- Each job is guaranteed a number of tokens as dictated by cluster policy; each running or initializing task uses one token. Token released on task completion.
- A token is a guaranteed share of CPU and memory
- To increase efficiency, unused tokens are re-allocated to jobs with available work
Jockey

Progress indicator

• Can use many features of the job to build a progress indicator

• Earlier work (ParaTimer) concentrated on fraction of tasks completed

• Our indicator is very simple, but we found it performs best for Jockey’s needs

\[
\sum_{\text{stage } s} f_s (Q_s + T_s)
\]
Comparison with ARIA

• ARIA uses analytic models
• Designed for 3 stages: Map, Shuffle, Reduce
• Jockey’s control loop is robust due to control-theory improvements
• ARIA tested on small (66-node) cluster without a network bottleneck
• We believe Jockey is a better match for production DAG frameworks such as Hive, Pig, etc.
Jockey

Latency prediction: $C(p, a)$

- Event-based simulator
  - Same scheduling logic as actual Job Manager
  - Captures important features of job progress
  - Does not model input size variation or speculative re-execution of stragglers
  - Inputs: job algebra, distributions of task timings, probabilities of failures, allocation

- Analytic model
  - Inspired by Amdahl's Law: $T = S + P/N$
  - $S$ is remaining work on critical path, $P$ is all remaining work, $N$ is number of machines
Jockey

Resource allocation control loop

• Executes in Dryad’s Job Manager

• Inputs: fraction of completed tasks in each stage, time job has spent running, utility function, precomputed values (for speedup)

• Output: Number of tokens to allocate

• Improved with techniques from control-theory
Jockey

offline
during job runtime

job profile
Jockey

offline

job profile

simulator
during job runtime
The Jockey simulator operates offline during job runtime, using a job profile and job stats.
Jockey

offline

simulator

job profile

during job runtime

latency predictions

job stats

utility function
Jockey

offline

job profile

simulator

during job runtime

latency
predictions

job stats

utility
function

resource allocation control loop

running
job