RLScheduler: An Automated HPC Batch Job Scheduler Using Reinforcement Learning

Di Zhang\textsuperscript{1}, Dong Dai\textsuperscript{1}, Youbiao He\textsuperscript{2}, Forrest Sheng Bao\textsuperscript{2}, Bing Xie\textsuperscript{3}

\textsuperscript{1}University of North Carolina at Charlotte
\textsuperscript{2}Iowa State University
\textsuperscript{3}Oak Ridge National Laboratory
Motivation & Background
RLScheduler Design
Evaluation & Analysis
Conclusion
Motivation & Background

• Introduction of HPC batch job schedulers
• Challenges of existing schedulers
• Background of Reinforcement Learning
HPC Batch Job Scheduler

Motivation & Background

Job Queue (Waiting Jobs)

- Job 1
- Job 2
- Job 3
- ...

- s_1
- n_1
- r_1
- ...

- s_2
- n_2
- r_2
- ...

- s_3
- n_3
- r_3
- ...

- s_i
- n_i
- r_i
- ...

- s_j
- n_j
- r_j
- ...

HPC Batch Job Scheduler

- computing nodes

- s: job submission time

- n: the number of processors that a job requests

- r: job's runtime estimation (or upper bound) from users

Computing Nodes
HPC Batch Job Scheduler

Job Queue (Waiting Jobs)

First Come First Serve (FCFS)

HPC Batch Job Scheduler

Computing Nodes

Motivation & Background

s: job submission time

n: the number of processors that a job requests

r: job's runtime estimation (or upper bound) from users
HPC Batch Job Scheduler

Job Queue (Waiting Jobs)

Job 1

s_1
n_1
r_1

Job 2

s_2
n_2
r_2

Job 3

s_3
n_3
r_3

……

HPC Batch Job Scheduler

Smallest Job First (Small)

Job Queue (Waiting Jobs)

s: job submission time

n: the number of processors that a job requests

r: job’s runtime estimation (or upper bound) from users

Motivation & Background

Computing Nodes
HPC Batch Job Scheduler

Job Queue (Waiting Jobs)

Job 1

s1
n1
r1

Job 2

s2
n2
r2

Job 3

s3
n3
r3

......

s: job submission time
n: the number of processors that a job requests
r: job’s runtime estimation (or upper bound) from users

Shortest Job First (SJF)

HPC Batch Job Scheduler

Computing Nodes

Motivation & Background
Motivation & Background

HPC Batch Job Scheduler

Job Queue (Waiting Jobs)

\[
\text{Score}(w, n, r) = -(w_t/r_t)^3 \times n_t
\]

Computing Nodes

- **w**: waiting time
- **n**: the number of processors that a job requests
- **r**: job's runtime estimation (or upper bound) from users
HPC Batch Job Scheduler

Job Queue (Waiting Jobs)

Score(s, n, r)

\[ score(t) = \log_{10}(r_t) \times n_t + 870 \times \log_{10}(s_t) \]

**Motivation & Background**

- **s**: job submission time
- **n**: the number of processors that a job requests
- **r**: job’s runtime estimation (or upper bound) from users
Motivation & Background

For a Given Scheduler:

- Different Job Traces
- Different Scheduling Goals
- Complicated Scheduling Goals

Amvrosiadis, et. al. On the diversity of cluster workloads and its impact on research results, USNIX ATC'18

From https://www.cs.huji.ac.il/labs/parallel/workload/L_ricc/index.html

Slurm classic Fair Share https://slurm.schedmd.com/classic_fair_share.html
Impact of Different Job Traces

Motivation & Background

Job Schedulers behave differently on Different Job Traces

Different Job Traces

Average bounded slowdown

Best!

Worst!
Impact of Scheduling Goals

Job schedulers behave differently toward different goals

Motivation & Background
Impact of Complicated Goals

Motivation & Background

What Scheduling Policy?

Minimize Average Bounded Slowdown & Maximize User Fairness

Complicated Goals require new schedulers

Maximize Resource Utilization

Minimize Average Bounded Slowdown

Maximize User Fairness

Minimize Average Turnaround time

Minimize Average Waiting time
Reinforcement Learning

Motivation & Background


Motivation & Background

• Overview of RLScheduler
• Challenges and Our Solutions

RLScheduler Design

Evaluation & Analysis

Conclusion
Our Contributions

• The first reinforcement learning based batch job scheduler for HPC systems

• New neural network and trajectory filtering mechanism to enable efficient RL training

• Extensively evaluations on efficiency, usability, and stability of RLScheduler.
RLScheduler Design

Job Queue (Waiting Jobs)

Policy Network

Value Network

Baseline

Reward

Computing Nodes

Actor-Critic Model

State $s_t$

Reward $r_t$

Action $a_t$

Environment

Agent

R value

m: the request memory per processor

u: the user's ID.

Job Queue (Waiting Jobs)
Challenge 1: Impact of Input Order

Job Queue (Waiting Jobs)

Job 1: s1, n1, r1, u1, m1
Job 3: s3, n3, r3, u3, m3
Job 2: s2, n2, r2, u2, m2

......

Computing Nodes

RLScheduler Design

RL-DNN

Job 1, Job 2, Job 3, .......

Job 2
Solution: Kernel-based Policy Network

Kernel-based Policy Network is insensitive to the order of jobs

Policy network: Kernel-based network
Challenge 2: High Variance in Samples

The average bounded slowdown of scheduling sequence of 256 jobs in PIK-IPLEX-2009 job trace.
Solution 1: Value Network

Value Network helps reduce the variance by estimating the value of different states.
Solution 2: Trajectory Filter

Filter out jobs and retrain jobs with average bounded slowdown in between Mid and 2*Mean.
Evaluation & Analysis

• Efficiency Evaluations
• Usability Evaluations
• Stability Evaluations
Efficiency
Usability
Stability

Evaluation & Analysis

- How is the performance of kernel-based neural network?
- How well can value network and trajectory filtering reduce the variance?

- How is the performance on various job traces?
- How is the performance for different optimization goals?
- How is the performance of complex metrics?

If RLScheduler works well in above scenarios, is it stable?
Evaluation & Analysis

- Efficiency
- Usability
- Stability

Evaluation Outline

Graphs and charts illustrate performance metrics such as Lublin-1, utilization, and job traces.
Compare Different Neural Networks

<table>
<thead>
<tr>
<th>Name</th>
<th>Layers</th>
<th>Size of each layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP_v1</td>
<td>3</td>
<td>128, 128, 128</td>
</tr>
<tr>
<td>MLP_v2</td>
<td>3</td>
<td>32, 16, 8</td>
</tr>
<tr>
<td>MLP_v3</td>
<td>5</td>
<td>32, 32, 32, 32</td>
</tr>
<tr>
<td>LeNet [33]</td>
<td>6</td>
<td>2x(conv2d, maxpooling2d), dense</td>
</tr>
<tr>
<td>RLScheduler</td>
<td>3</td>
<td>32, 16, 8</td>
</tr>
</tbody>
</table>

Converge

The horizontal axis shows the total number of training epoch; the vertical axis shows the performance of the agent. The larger vertical axis value indicates a smaller average bounded job slowdown and is better.
The training curves of RLScheduler on PIK-IPLEX2009 job trace with and without trajectory filtering.

With/Without Trajectory Filtering

With trajectory enabled, RLScheduler converges within 100 epochs.
Evaluation & Analysis

Evaluation Outline

Efficiency

Usability

Stability

Graphs and charts showing performance metrics for various scenarios, including average, worst, and best job traces.
Training on Different Job Traces:

RLScheduler converges in all of the workloads within 100 training epochs and different job traces have different converge pattern.
Testing on **Different Job Traces:**

<table>
<thead>
<tr>
<th>Trace</th>
<th>FCFS</th>
<th>WFP3</th>
<th>UNI</th>
<th>SJF</th>
<th>F1</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scheduling without Backfilling</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lublin-1</td>
<td>7273.8</td>
<td>19754</td>
<td>22275</td>
<td>277.35</td>
<td>258.37</td>
<td>254.67</td>
</tr>
<tr>
<td>SDSC-SP2</td>
<td>1727.5</td>
<td>3000.9</td>
<td>1848.5</td>
<td>2680.6</td>
<td>1232.1</td>
<td>466.44</td>
</tr>
<tr>
<td>HPC2N</td>
<td>297.18</td>
<td>426.99</td>
<td>609.77</td>
<td>157.71</td>
<td>118.01</td>
<td>117.01</td>
</tr>
<tr>
<td>Lublin-2</td>
<td>7842.5</td>
<td>9523.2</td>
<td>11265</td>
<td>787.89</td>
<td><strong>698.34</strong></td>
<td>724.51</td>
</tr>
<tr>
<td><strong>Scheduling with Backfilling</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lublin-1</td>
<td>235.82</td>
<td>133.87</td>
<td>307.23</td>
<td>73.31</td>
<td>75.07</td>
<td><strong>58.64</strong></td>
</tr>
<tr>
<td>SDSC-SP2</td>
<td>1595.1</td>
<td>1083.1</td>
<td>548.01</td>
<td>2167.8</td>
<td>1098.2</td>
<td><strong>397.82</strong></td>
</tr>
<tr>
<td>HPC2N</td>
<td>127.38</td>
<td>97.39</td>
<td>175.12</td>
<td>122.04</td>
<td><strong>71.95</strong></td>
<td>86.14</td>
</tr>
<tr>
<td>Lublin-2</td>
<td>247.61</td>
<td>318.35</td>
<td>379.59</td>
<td><strong>91.99</strong></td>
<td>148.25</td>
<td>118.79</td>
</tr>
</tbody>
</table>

RLScheduler performs either comparably well to the best or is the best among the presented schedulers.
Training on **Different Goals:**

RLScheduler converges towards this new goal but with different patterns.
Testing on Different Goals:

<table>
<thead>
<tr>
<th>Trace</th>
<th>FCFS</th>
<th>WFP3</th>
<th>UNICEP</th>
<th>SJF</th>
<th>F1</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lublin-1</td>
<td>0.657</td>
<td>0.747</td>
<td>0.691</td>
<td>0.762</td>
<td>0.816</td>
<td>0.714</td>
</tr>
<tr>
<td>SDSC-SP2</td>
<td>0.670</td>
<td>0.658</td>
<td>0.688</td>
<td>0.645</td>
<td>0.674</td>
<td>0.671</td>
</tr>
<tr>
<td>HPC2N</td>
<td>0.638</td>
<td>0.636</td>
<td>0.636</td>
<td>0.640</td>
<td>0.637</td>
<td>0.640</td>
</tr>
<tr>
<td>Lublin-2</td>
<td>0.404</td>
<td>0.543</td>
<td>0.510</td>
<td>0.562</td>
<td>0.478</td>
<td>0.562</td>
</tr>
</tbody>
</table>

**Scheduling without Backfilling**

<table>
<thead>
<tr>
<th>Trace</th>
<th>FCFS</th>
<th>WFP3</th>
<th>UNICEP</th>
<th>SJF</th>
<th>F1</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lublin-1</td>
<td>0.868</td>
<td>0.864</td>
<td>0.883</td>
<td>0.778</td>
<td>0.840</td>
<td>0.850</td>
</tr>
<tr>
<td>SDSC-SP2</td>
<td>0.682</td>
<td>0.681</td>
<td>0.706</td>
<td>0.661</td>
<td>0.677</td>
<td>0.707</td>
</tr>
<tr>
<td>HPC2N</td>
<td>0.639</td>
<td>0.637</td>
<td>0.638</td>
<td>0.641</td>
<td>0.638</td>
<td>0.642</td>
</tr>
<tr>
<td>Lublin-2</td>
<td>0.587</td>
<td>0.583</td>
<td>0.587</td>
<td>0.593</td>
<td>0.552</td>
<td>0.593</td>
</tr>
</tbody>
</table>

**Scheduling with Backfilling**

<table>
<thead>
<tr>
<th>Trace</th>
<th>FCFS</th>
<th>WFP3</th>
<th>UNICEP</th>
<th>SJF</th>
<th>F1</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lublin-1</td>
<td>0.868</td>
<td>0.864</td>
<td>0.883</td>
<td>0.778</td>
<td>0.840</td>
<td>0.850</td>
</tr>
<tr>
<td>SDSC-SP2</td>
<td>0.682</td>
<td>0.681</td>
<td>0.706</td>
<td>0.661</td>
<td>0.677</td>
<td>0.707</td>
</tr>
<tr>
<td>HPC2N</td>
<td>0.639</td>
<td>0.637</td>
<td>0.638</td>
<td>0.641</td>
<td>0.638</td>
<td>0.642</td>
</tr>
<tr>
<td>Lublin-2</td>
<td>0.587</td>
<td>0.583</td>
<td>0.587</td>
<td>0.593</td>
<td>0.552</td>
<td>0.593</td>
</tr>
</tbody>
</table>

RLScheduler has good performance among all the presented schedulers.

Average bounded slowdown

Best!

Not the best
Minimizing maximal average bounded slowdown among users is a complicated metrics considering Performance and Fairness at the same time.
RLScheduler with Fairness

<table>
<thead>
<tr>
<th>Trace</th>
<th>FCFS</th>
<th>WFP3</th>
<th>UNICEP</th>
<th>SJF</th>
<th>F1</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Scheduling without Backfilling</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDSC-SP2</td>
<td>7257</td>
<td>14858</td>
<td>12234</td>
<td>12185</td>
<td>8260</td>
<td>4116</td>
</tr>
<tr>
<td>HPC2N</td>
<td>2058</td>
<td>5107</td>
<td>5145</td>
<td>1255</td>
<td>1310</td>
<td>1147</td>
</tr>
<tr>
<td><strong>Scheduling with Backfilling</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDSC-SP2</td>
<td>7356</td>
<td>8464</td>
<td>3840</td>
<td>10121</td>
<td>7799</td>
<td>2712</td>
</tr>
<tr>
<td>HPC2N</td>
<td>1502</td>
<td>2125</td>
<td>2081</td>
<td>1491</td>
<td>583</td>
<td>519</td>
</tr>
</tbody>
</table>

Results of scheduling different job traces towards average bounded slowdown with Maximal Fairness.

RLScheduler can consider multiple metrics at the same time: minimizing average bounded slowdown and keeping fairness among users together.
Evaluation & Analysis

- Efficiency
- Usability
- Stability

Graphs and charts illustrating performance metrics, including:
- Lublin-1 performance over episodes
- Utilization patterns
- Job traces comparisons with average benchmarks.
Stability Evaluation

Job Trace X → RL Model Trained on X (RL-X)

Job Trace Y → RL Model Trained on Y (RL-Y)

Result of the Worst heuristic scheduling policies

Result of the Best heuristic scheduling policies

Average bsld

Never seen by RL-X

Never seen by RL-Y

X

Y

Z

Job Traces

Worst

Best

RL-X

RL-Y

Evaluation & Analysis
Stability Evaluation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling without Backfilling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lublin-1</td>
<td>258.37 (F1)</td>
<td>22274.74 (UNICEP)</td>
<td>254.67</td>
<td>482.62</td>
<td>283.00</td>
<td>334.73</td>
</tr>
<tr>
<td>SDSC-SP2</td>
<td>1232.06 (F1)</td>
<td>3000.88 (WFP3)</td>
<td>1543.40</td>
<td>466.44</td>
<td>1016.83</td>
<td>1329.41</td>
</tr>
<tr>
<td>HPC2N</td>
<td>118.01 (F1)</td>
<td>660.77 (UNICEP)</td>
<td>169.91</td>
<td>300.43</td>
<td>186.42</td>
<td>236.00</td>
</tr>
<tr>
<td>Lublin-2</td>
<td>698.34 (F1)</td>
<td>11265.3 (UNICEP)</td>
<td>665.49</td>
<td>805.16</td>
<td>648.52</td>
<td>724.51</td>
</tr>
<tr>
<td>ANL Intrepid</td>
<td>8.39 (F1)</td>
<td>35.11 (FCFS)</td>
<td>9.91</td>
<td>9.61</td>
<td>8.93</td>
<td>9.75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling with Backfilling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lublin-1</td>
<td>73.31 (SJF)</td>
<td>307.23 (UNICEP)</td>
<td>58.64</td>
<td>93.16</td>
<td>54.65</td>
<td>64.45</td>
</tr>
<tr>
<td>SDSC-SP2</td>
<td>548.01 (UNICEP)</td>
<td>2167.84 (SJF)</td>
<td>1364.43</td>
<td>397.82</td>
<td>746.65</td>
<td>1192.97</td>
</tr>
<tr>
<td>HPC2N</td>
<td>71.95 (F1)</td>
<td>175.12 (UNICEP)</td>
<td>115.93</td>
<td>128.73</td>
<td>115.79</td>
<td>144.54</td>
</tr>
<tr>
<td>Lublin-2</td>
<td>91.99 (SJF)</td>
<td>379.59 (UNICEP)</td>
<td>172.15</td>
<td>183.98</td>
<td>139.80</td>
<td>118.79</td>
</tr>
<tr>
<td>ANL Intrepid</td>
<td>2.73 (F1)</td>
<td>4.12 (UNICEP)</td>
<td>3.63</td>
<td>4.56</td>
<td>3.99</td>
<td>3.58</td>
</tr>
</tbody>
</table>

A learned RLScheduler model, regardless of which job trace it was trained on, can be safely applied to other job traces.
Motivation & Background
RLScheduler Design
Evaluation & Analysis
Conclusion
Summary

• We designed and implemented the first RL-based HPC batch job scheduler.
  • https://github.com/DIR-LAB/deep-batch-scheduler

• We introduced new network design and trajectory filtering mechanism in RLScheduler to stabilize and speedup the training.

• We conducted extensive evaluations to show the efficiency, usability, and stability of RLScheduler across various HPC job traces and scheduling goals.
Thank you! & Questions?