MODEL
LEARNING RATE $l$, AND WEIGHT DECAY $w$

AZURE

AWS

GOOGLE

MODEL LEARNING RATE $l$, AND WEIGHT DECAY $w$

TIME

$\begin{align*}
\text{Azure} & : \\
\text{AWS} & : \\
\text{Google} & :
\end{align*}$
DISTRIBUTED, PARALLEL HYPERPARAMETER TUNING

MODEL
LEARNING RATE $l$, AND WEIGHT DECAY $w$

<table>
<thead>
<tr>
<th>Job</th>
<th>Time</th>
<th>Trial</th>
<th>Job</th>
<th>Time</th>
<th>Trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l=0.01, w=5E-4$</td>
<td>0.609</td>
<td>$l=0.01, w=5E-4$</td>
<td>0.565</td>
<td>$l=0.01, w=5E-4$</td>
<td>0.833</td>
</tr>
<tr>
<td>$l=0.01, w=5E-4$</td>
<td>0.565</td>
<td>$l=0.05, w=5E-4$</td>
<td>0.609</td>
<td>$l=0.05, w=5E-4$</td>
<td>0.818</td>
</tr>
<tr>
<td>$l=0.01, w=5E-6$</td>
<td>0.577</td>
<td>$l=0.01, w=5E-6$</td>
<td>0.623</td>
<td>$l=0.01, w=5E-6$</td>
<td>0.821</td>
</tr>
<tr>
<td>$l=0.05, w=5E-6$</td>
<td>0.594</td>
<td>$l=0.01, w=5E-5$</td>
<td>0.610</td>
<td>$l=0.01, w=5E-5$</td>
<td>0.807</td>
</tr>
<tr>
<td>$l=0.01, w=5E-5$</td>
<td>0.583</td>
<td>$l=0.01, w=5E-4$</td>
<td>0.833</td>
<td>$l=0.01, w=5E-4$</td>
<td>0.940</td>
</tr>
<tr>
<td>$l=0.01, w=5E-4$</td>
<td>0.594</td>
<td>$l=0.05, w=5E-4$</td>
<td>0.609</td>
<td>$l=0.05, w=5E-4$</td>
<td>0.818</td>
</tr>
<tr>
<td>$l=0.01, w=5E-6$</td>
<td>0.577</td>
<td>$l=0.01, w=5E-6$</td>
<td>0.623</td>
<td>$l=0.01, w=5E-6$</td>
<td>0.821</td>
</tr>
<tr>
<td>$l=0.05, w=5E-6$</td>
<td>0.594</td>
<td>$l=0.01, w=5E-5$</td>
<td>0.610</td>
<td>$l=0.01, w=5E-5$</td>
<td>0.807</td>
</tr>
<tr>
<td>$l=0.01, w=5E-5$</td>
<td>0.583</td>
<td>$l=0.01, w=5E-4$</td>
<td>0.833</td>
<td>$l=0.01, w=5E-4$</td>
<td>0.940</td>
</tr>
</tbody>
</table>

Distributed, Parallel Hyperparameter Tuning

Model
Learning Rate $l$, and Weight Decay $w$
DISTRIBUTED, PARALLEL HYPERPARAMETER TUNING

SUCCESSIVE HALVING ALGORITHM (SHA)
Distributed, Parallel Hyperparameter Tuning

Stage-Level Parallelism

Stage 1
- GPU 1: T1
- GPU 2: T2
- GPU 3: T3
- GPU 4: T4
- GPU 5: T5
- GPU 6: T6
- GPU 7: T7
- GPU 8: T8

Stage 2
- Stage 1 outputs feed into Stage 2.
- GPU 2: T2
- GPU 4: T4
- GPU 6: T6
- GPU 8: T8

Stage 3
- Stage 2 outputs feed into Stage 3.
- GPU 2: T2
- GPU 4: T4
- GPU 6: T6
- GPU 8: T8

Successive Halving Algorithm (SHA)
DISTRIBUTED, PARALLEL HYPERPARAMETER TUNING

Stage 1:
- GPU 1: T1
- GPU 2: T2
- GPU 3: T3
- GPU 4: T4
- GPU 5: T5
- GPU 6: T6
- GPU 7: T7
- GPU 8: T8

Stage 2:
- T1 → T2
- T2
- T4
- T4

Stage 3:
- T2 → T4
- T4 → T6
- T6

Trial-Level Parallelism
DISTRIBUTED, PARALLEL HYPERPARAMETER TUNING

Stage 1

Stage 2

Stage 3

GPU 1

GPU 2

GPU 3

GPU 4

GPU 5

GPU 6

GPU 7

GPU 8

T1

T2

T3

T4

T5

T6

T7

T8

IDLE

T2

IDLE

T4

IDLE

T6

IDLE

TIME
DISTRIBUTED, PARALLEL HYPERPARAMETER TUNING

Stage 1
- GPU 1: T1
- GPU 2: T2
- GPU 3: T3
- GPU 4: T4
- GPU 5: T5
- GPU 6: T6
- GPU 7: T7
- GPU 8: T8

Stage 2
- Stage 1 results are sent to GPUs:
  - GPU 1: T1
  - GPU 2: T2
  - GPU 3: T4
  - GPU 4: T4
  - GPU 5: T5
  - GPU 6: T6
  - GPU 7: T7
  - GPU 8: T8

Stage 3
- Final stage of tuning:
  - T2 is sent to GPU 2
  - T4 is sent to GPU 4
  - T6 is sent to GPU 6

Time progresses from left to right.
DISTRIBUTED, PARALLEL HYPERPARAMETER TUNING

Stage 1
- GPU 1: T1
- GPU 2: T2
- GPU 3: T3
- GPU 4: T4
- GPU 5: T5
- GPU 6: T6
- GPU 7: T7
- GPU 8: T8

Stage 2
- GPU 1: T1
- GPU 2: T2
- GPU 3: T3
- GPU 4: T4
- GPU 5: T5
- GPU 6: T6
- GPU 7: T7
- GPU 8: T8

Stage 3
- T4
- T6

TIME
Distributed, Parallel Hyperparameter Tuning

Stage 1

Stage 2

Stage 3

GPU 1
GPU 2
GPU 3
GPU 4
GPU 5
GPU 6
GPU 7
GPU 8

T1
T2
T3
T4
T5
T6
T7
T8

T2
T4
T6
T4
T6
T7

GPU 1
GPU 2
GPU 3
GPU 4
GPU 5
GPU 6
GPU 7
GPU 8

T1
T2
T3
T4
T5
T6
T7
T8

Time
DISTRIBUTED, PARALLEL HYPERPARAMETER TUNING

Stage 1
Stage 2
Stage 3

GPU 1
GPU 2
GPU 3
GPU 4
GPU 5
GPU 6
GPU 7
GPU 8

T1
T2
T3
T4
T5
T6
T7
T8

Time
DISTRIBUTED, PARALLEL HYPERPARAMETER TUNING

Stage 1

- GPU 1: T1
- GPU 2: T2
- GPU 3: T3
- GPU 4: T4
- GPU 5: T5
- GPU 6: T6
- GPU 7: T7
- GPU 8: T8

Stage 2

- DEPROVISION

- GPU 2: T2
- GPU 3: T3
- GPU 4: T4
- GPU 5: T5
- GPU 6: T6
- GPU 7: T7
- GPU 8: DEPROVISION

Stage 3

- DEPROVISION
- T4
- DEPROVISION
- T6
- DEPROVISION

Time
Given a time constraint, minimize the cost of executing a hyperparameter tuning job.

<table>
<thead>
<tr>
<th>STAGE</th>
<th>EPOCHS</th>
<th>TRIALS</th>
<th>GPUs/TRIAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0-4</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5-12</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>13-28</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Allocation plan
**Challenges**

**How can we model the job completion time and cost of the given allocation plan?**

**How can we generate a low cost allocation plan that completes on time?**

**How can we schedule said allocation to optimize worker co-location + cluster utilization?**
<table>
<thead>
<tr>
<th>Challenges</th>
<th>Rubberband</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>How can we model the job completion time and cost of the given allocation plan?</strong></td>
<td><strong>Cost/performance model via profiling</strong>&lt;br&gt;DL model training latency and provisioning overheads</td>
</tr>
<tr>
<td><strong>How can we generate a low cost allocation plan that completes on time?</strong></td>
<td><strong>DAG-based execution model which finds feasible and cost-efficient resource allocations</strong></td>
</tr>
<tr>
<td><strong>How can we schedule said allocation to optimize worker co-location + cluster utilization?</strong></td>
<td><strong>Full-stack system for placement, scheduling, and scaling</strong></td>
</tr>
</tbody>
</table>
**Issue 1: Modeling Job Completion Time and Cost**

**Performance Modeling**
- Training latency
- Provider queuing delay
- Instance initialization latency

**Cost Modeling**
- Compute price
- Billing granularity
- Data price
ISSUE 2: FINDING A LOW-COST ALLOCATION PLAN
ISSUE 2: FINDING A LOW-COST ALLOCATION PLAN

STEP 1: GENERATE CANDIDATES
ISSUE 2: FINDING A LOW-COST ALLOCATION PLAN

**STEP 1:** GENERATE CANDIDATES

**STEP 2:** USE SIMULATOR TO PREDICT JOB COMPLETION TIME
ISSUE 2: FINDING A LOW-COST ALLOCATION PLAN

**STEP 1:** GENERATE CANDIDATES

**STEP 2:** USE SIMULATOR TO PREDICT JOB COMPLETION TIME

**STEP 3:** GREEDILY SELECT BEST CANDIDATE
ISSUE 2: FINDING A LOW-COST ALLOCATION PLAN

**Step 1:** Generate candidates

**Step 2:** Use simulator to predict job completion time

**Step 3:** Greedily select best candidate

Maximize **cost-marginal benefit**:

\[ M = \frac{\text{Cost of current best plan} - \text{Cost of proposed plan}}{\text{JCT of proposed plan} - \text{JCT of current best plan}} \]
ISSUE 2: FINDING A LOW-COST ALLOCATION PLAN

**STEP 1:** GENERATE CANDIDATES

**STEP 2:** USE SIMULATOR TO PREDICT JOB COMPLETION TIME

**STEP 3:** GREEDILY SELECT BEST CANDIDATE

**STEP 4:** ITERATE WITH NEW BEST CANDIDATE

Maximize **COST-MARGINAL BENEFIT:**

\[
M = \frac{\text{Cost of current best plan} - \text{Cost of proposed plan}}{\text{JCT of proposed plan - JCT of current best plan}}
\]
 ISSUE 3: EFFECTIVELY EXECUTE ALLOCATION PLAN

END OF STAGE

SCHEDULER

ALLOCATION PLAN

# RESOURCES NEEDED

CLUSTER MANAGER

RESOURCES NEEDED

MODIFIED CLUSTER

PLACEMENT CONTROLLER

RESOURCE ALLOCATION

PHYSICAL RESOURCE ASSIGNMENTS

Issue 3: Effectively execute allocation plan

- Resource allocation
- Physical resource assignments

Cluster manager

Allocation Plan

Resources needed

Modified cluster

End of stage
### Issue 3: Effectively Execute Allocation Plan

<table>
<thead>
<tr>
<th>Stage</th>
<th>Epochs</th>
<th>Trials</th>
<th>GPUs/Trial</th>
<th>Cluster Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0-4</td>
<td>8</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5-12</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Cluster:**
- **Stage 1:** T1, T1, T2, T2, T3, T3, T4, T4
- **Stage 2:** T5, T5, T6, T6, T7, T7, T8, T8

End of Stage
ISSUE 3: EFFECTIVELY EXECUTE ALLOCATION PLAN

SCHEDULER

STOP - T2, T3, T5, T7
CONTINUE - T1, T4, T6, T8

CLUSTER
ISSUE 3: EFFECTIVELY EXECUTE ALLOCATION PLAN

Cluster Manager → Deprovision Node 4

Cluster: T1 T1 T3 T3 T5 T5 T7 T7 T8 T8

Deprovision Node 4: T8 T8
ISSUE 3: EFFECTIVELY EXECUTE ALLOCATION PLAN

PLACEMENT CONTROLLER

MOVE T8 TO NODE 1 AND 2

CLUSTER
System

1. **Profiler** and **Simulator** model job completion time + cost of potential allocations

2. **Planner** generates a low cost allocation plan that completes on time

3. **Scheduler**, **Placement Controller**, and **Cluster Manager** execute the allocation plan such that worker co-location and cluster utilization are maximized
End-to-End Results
Simulation Quality

![Graph showing JCT (s) for Static and Rubberband]

![Graph showing Cost ($) for Static and Rubberband]
Across Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Static</th>
<th>Rubberband</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>CIFAR100</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>RTE</td>
<td>35</td>
<td>25</td>
</tr>
</tbody>
</table>
ACROSS DEADLINES

![Diagram showing JCT (s) and Cost ($) for Static and Rubberband across different values of 1200, 1800, and 2400. The JCT (s) diagram shows a decrease in JCT for Rubberband compared to Static as the value increases. The Cost ($) diagram shows an increase in Cost for Rubberband compared to Static as the value increases.]
Thank You!