**Characteristics of Deep Learning Jobs**

**Feedback-driven exploration**
- Deep learning experiments today use manual or automatic (AutoML) trial-and-error techniques to find the best model.

**Model Sensitivity**
- DL jobs have different sensitivities to resource affinity, due to network architecture or hyperparameters (e.g., batch-size).
- Other cases: Inter-server locality, 1-GPU interference, NIC interference.

**Intra-job Predictability**
- GPU Memory usage follows a cyclic pattern aligned with mini-batch boundaries, usually with more than 10x difference in utilization within a mini-batch.

**Scheduling Mechanisms for Deep Learning**

**Traditional GPU Allocation**
- Allocation reactively at job arrival and departure.
- Dedicated GPUs for a job in its whole lifetime.
- Jobs queued if no qualified resources.

**Migration**
- Generic GPU processes: Use CRIU to dump and restore process state across machines.
- Checkpoint-aware processes: Repurpose TF checkpointing APIs to save and restore state. Pre-warm libraries for fast migration.

**Grow-Shrink**
- Opportunistically scale jobs to idle GPUs.
- Vacate GPUs on-demand.
- Depends on job capabilities to utilize additional GPUs.

**Introspective Policies**

**Over-subscription**
- Time-slice to allow multiple jobs to run simultaneously with a weighted time-share.
- Pack multiple jobs in the same server if jobs have light-weight resource requirements.

**Runtime adjustment**
- Migrate jobs at mini-batch boundary if better resources appear.
- Defrag GPUs to better compact resources for multi-GPU jobs.
- Grow to more resources when available and shrink when required.

**Profiling for introspection**
- Monitor resource utilization (e.g., GPU utilization and memory).
- Non-invasive progress rate estimation for scheduling decisions.

**Experimental Results**

**Hyperparameter Search with Time-slicing**
Search across 12 dimensions - LeNet on CIFAR-10

<table>
<thead>
<tr>
<th>Position</th>
<th>95th (25%)</th>
<th>18th (50%)</th>
<th>2080 (75%)</th>
<th>3659 (98%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 GPUs</td>
<td>691.5</td>
<td>1373.0</td>
<td>2067.2</td>
<td>2726.4</td>
</tr>
<tr>
<td>Baseline</td>
<td>125.5</td>
<td>213.8</td>
<td>302.4</td>
<td>397.1</td>
</tr>
<tr>
<td>Speedup</td>
<td>5.51x</td>
<td>6.44x</td>
<td>6.64x</td>
<td>7.04x</td>
</tr>
<tr>
<td>16 GPUs</td>
<td>253.0</td>
<td>492.7</td>
<td>731.7</td>
<td>970.0</td>
</tr>
<tr>
<td>Baseline</td>
<td>74.4</td>
<td>103.7</td>
<td>135.4</td>
<td>162.6</td>
</tr>
<tr>
<td>Speedup</td>
<td>3.40x</td>
<td>4.73x</td>
<td>5.40x</td>
<td>5.96x</td>
</tr>
</tbody>
</table>

**Cluster Experiment: Time-slicing + Packing**
Mixed PyTorch jobs on 180 Tesla GPUs

26% increase in cluster GPU utilization
4.5x faster job feedback

**Low overhead Suspend/Resume & Migration**

**Cluster Experiment: Time-slicing + Migration**
9-day trace from Microsoft servers on 100 GPUs

27% reduction in job completion time
13.6x faster hyperparameter search