Irina: Accelerating DNN Inference with Efficient Online Scheduling

Xiaorui Wu, Hong Xu, Yi Wang
DNN inference in real world

Traffic Monitoring

Surveillance

AI Assistant

Garbage Classification
DNN online service pipelines

Distributed DNN Inference
Distributed DNN inference serving
Distributed DNN inference serving

- Two Perspectives

Client Perspective

Cluster Perspective
DNN inference serving

• Cluster: Improve the throughput of cluster and reduce the total makespan
• Client: Reduce query its response time
Different client experience?

- Execute task A and then execute task B
- Execute task B and then execute task A
Different client experience?

• Execute task A and then execute task B

• Execute task B and then execute task A
Existing DNN serving systems focus on offline scheduling

E.g., TensorFlow Serving, Cliper, Nexus

• Aim to improve the throughput of cluster
• Don’t consider the client experience
• Rely on offline scheduling policy, online serving is FIFO
Irina: efficient DNN online inference scheduling

• Minimizes the average job completion time (JCT)
• Improves the client experience

\[
\text{Average JCT} = \frac{\sum_{i=1}^{n} (T_i - t_i) \times B_i}{\sum_{i=1}^{n} B_i}
\]
Dynamical scheduling

1. Task-level
   • Preemption scheduling: provide limited resource for the task with the highest benefit
   • Stacking scheduling: run multiple tasks concurrently

2. Model-level
   • Batching scheduling: batch examples across tasks
Preemption scheduling

- Ongoing task occupies limited resources
- New task contains a large number of queries
- Minimize average JCT
- Satisfy the latency SLOs

<table>
<thead>
<tr>
<th>Model</th>
<th>Batch Size</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td>75</td>
</tr>
<tr>
<td>B</td>
<td>256</td>
<td>10</td>
</tr>
</tbody>
</table>
Preemption scheduling

Problems: Ongoing task occupies limited resource

\[ JCT = \frac{75 \times 8 + 84 \times 256}{8 + 256} = 83.73 \]
Preemption scheduling

Challenges:
- How to enable preemption?
- How to process the immediate results?

$JCT = \frac{10 \times 256 + 86 \times 8}{8 + 256} = 12.30$
Stacking scheduling

- Underutilization of the GPU resource
- Concurrent tasks meet the GPU resource limitation
- Framework supports to run multiple models in the same runtime

<table>
<thead>
<tr>
<th>Model</th>
<th>Batch Size</th>
<th>Latency (ms)</th>
<th>Avg. GPU (%)</th>
<th>Peak Memory (%)/On-Peak Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>10</td>
<td>200</td>
<td>70%</td>
<td>65/100</td>
</tr>
<tr>
<td>D</td>
<td>100</td>
<td>10</td>
<td>28%</td>
<td>35/5</td>
</tr>
</tbody>
</table>
Stacking scheduling

JCT = \frac{200 \times 10 + 50 \times 100}{10 + 100} = 63.64

Problem: Single task cannot saturate the GPU resource
Stacking scheduling

\[
JCT = \frac{200 \times 10 + 10 \times 100}{10 + 100} = 27.3
\]

**Challenge:** Interference among concurrent tasks
Batching scheduling

• The same model in ongoing task and new task
• Relatively low latency growth
• Sufficient GPU computing resource

<table>
<thead>
<tr>
<th>Model</th>
<th>Batch Size</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>60</td>
</tr>
<tr>
<td>A</td>
<td>8</td>
<td>75</td>
</tr>
<tr>
<td>A</td>
<td>16</td>
<td>85</td>
</tr>
<tr>
<td>A</td>
<td>32</td>
<td>150</td>
</tr>
</tbody>
</table>
Batching scheduling

App A

App A

$JCT = \frac{75 \times 8 + 149 \times 8}{8 + 8} = 112$

Problem: Sequential execution cause underutilization of GPU resource
Batching scheduling

$JCT = \frac{86 \times 8 + 85 \times 8}{8 + 8} = 85.5$

Challenges:
- How to enable preemption?
- How to process the immediate results?
Challenges:

1. How to enable preemption?
2. How to process the immediate results?
3. Interference among concurrent GPU models
Enabling preemption:

- Possible solutions:
  1. Killing/Canceling the thread ✗
  2. Multiple process share the same GPU driver ✗
  3. Modify Exit op in computing graph ✓
Immediate results

• Possible solutions:
  1. Swap in/out immediate results
     Save computing resource
  2. Recomputing immediate results
     Reduce IO operations
Interference among concurrent GPU models

- Two solutions:
  1. Stack long running tasks and short running tasks
     Reduce interference influence
  2. Profiling concurrent running tasks
     Choose models with little interference
Real data trace simulation

• Data trace is collected from testbed
• Simulate 4 applications with varied workloads
• Improve JCT by 1.3x-2.5x over Tensorflow Serving

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Batch Size</th>
<th>Letency</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLOv3</td>
<td>8</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>117</td>
</tr>
<tr>
<td>VGG-16</td>
<td>8</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>33</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>32</td>
<td>14</td>
</tr>
<tr>
<td>AlexNet</td>
<td>128</td>
<td>9</td>
</tr>
</tbody>
</table>
Conclusion

Irina serves multiple applications with efficient online scheduling at the low job completion time

- Includes three dynamic scheduling policies
- Enables preemption to support preemption scheduling and batching scheduling policies
- Analyzes interference across models and frameworks
- Uses JCT as the target to schedule online inference workloads
Thank you