Microsecond-scale Preemption for Concurrent GPU-accelerated DNN Inferences

Mingcong Han, Hanze Zhang, Rong Chen, Haibo Chen

Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University
Shanghai AI Laboratory
Engineering Research Center for Domain-specific Operating Systems, Ministry of Education, China
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China
Motivation

**DNNs** are widely adopted by modern intelligent applications
**Motivation**

Real-time tasks
- Latency critical

Best-effort tasks
- No hard real-time requirement

Obstacle Detection

Fatigue Detection
Motivation

Real-time tasks

√ Low Inference Latency
× Low Resource Utilization

Best-effort tasks

Real-time tasks

? Low Inference Latency
√ High Resource Utilization

Best-effort tasks
GPU-accelerated DNN inference

Task (a list of kernels)

Kernel

Block

Time

GPU

Compute Unit (CU)
Existing GPU Task Scheduling

- Best-effort Task #1
- Real-time Task #1
- Best-effort Task #2

**Preemption Latency**

**Overall Latency**

**(RT)Task Latency**
Existing GPU Task Scheduling

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

BE Task#1 Arriving

Sequential Execution
Existing GPU Task Scheduling

Best-effort Task#1

Real-time Task#1

Best-effort Task#2

BE Task#1 Arriving

Sequential Execution

GPU
Existing GPU Task Scheduling

Sequential Execution

BE Task#1 Arriving

GPU

Block

Kernel
Existing GPU Task Scheduling

![Diagram showing the scheduling of tasks on a GPU]

- **Best-effort Task#1**
- **Real-time Task#1**
- **Best-effort Task#2**

**Preemption**

**Latency**

**Overall Latency**

**(RT)Task Latency**

**Preemption Latency**

**Sequential Execution**

**GPU**
Existing GPU Task Scheduling

Best-effort Task#1

Real-time Task#1

Best-effort Task#2

RT Task#1 Arriving

Sequential Execution

GPU
Existing GPU Task Scheduling

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

Sequential Execution

- RT Task#1 Arriving
- BE Task#2 Arriving

Latency

Overall Latency

(task Latency

Preemption Latency

Time
Existing GPU Task Scheduling

- **Best-effort Task #1**
  - Block
  - Kernel

- **Real-time Task #1**
  - Sequential Execution

- **Best-effort Task #2**
  - Overall Latency
  - (RT) Task Latency
  - Preemption Latency

- **RT Task #1 Arriving**
- **BE Task #2 Arriving**

**Sequential Execution**

**GPU**

**Time**
Existing GPU Task Scheduling

Sequential Execution

RT Task#1 Arriving

BE Task#2 Arriving

Time

Overall Latency

(RT)Task Latency

Preemption Latency
Existing GPU Task Scheduling

Sequential Execution

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

- RT Task#1 Arriving
- BE Task#2 Arriving

- GPU

- No preemption

- Overall Latency
- (RT) Task Latency
- Preemption Latency

- Time
Existing GPU Task Scheduling

Sequential Execution

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

Time

RT Task#1 Arriving

BE Task#2 Arriving

no preemption

Preemption Latency
Existing GPU Task Scheduling

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

Sequential Execution

- Block
- Kernel

RT Task#1 Arriving

BE Task#2 Arriving

no preemption
Existing GPU Task Scheduling

Sequential Execution

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

RT Task#1 Arriving
BE Task#2 Arriving

GPU

Preemption Latency

Overall Latency

(RT) Task Latency

Time

no preemption
Existing GPU Task Scheduling

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

- Preemption
- Latency
- Overall Latency
- (RT)Task Latency
- Preemption Latency

Sequential Execution

- RT Task#1 Arriving
- BE Task#2 Arriving
- no preemption
- resource wasted

- GPU

- Time
Existing GPU Task Scheduling

- **Best-effort Task#1**
- **Real-time Task#1**
- **Best-effort Task#2**

- **Block**
- **Kernel**

- **Preemption**
- **Latency**

Time:
- **RT Task#1 Arriving**
- **BE Task#2 Arriving**

Sequential Execution

- **GPU**
- **Resource wasted**

- **Overall Latency**
- **(RT)Task Latency**
- **Preemption Latency**

- **High latency for RT tasks**
- **Low throughput** (not work-conserving)
Existing GPU Task Scheduling

- **Best-effort Task#1**
- **Real-time Task#1**
- **Best-effort Task#2**

**Preemption**

**Latency**

**Overall Latency**

- **(RT)Task Latency**
- **Preemption Latency**

**Sequence**

- **RT Task#1 Arriving**
- **BE Task#2 Arriving**

**Execution**

- **Sequential Execution**
- **Block-level Preemption**

**No Preemption**

- **No concurrency**
Existing GPU Task Scheduling

Sequential Execution

Block-level Preemption

Best-effort Task#1

Real-time Task#1

Best-effort Task#2

block

kernel

no preemption

no concurrency

RT Task#1 Arriving

BE Task#2 Arriving

Preemption

Latency

Overall Latency

(RT)Task Latency

Time
Existing GPU Task Scheduling

- **Best-effort Task#1**
- **Real-time Task#1**
- **Best-effort Task#2**

**Preemption**

**Latency**

**Overall Latency**

**RT Task Latency**

**Preemption Latency**

**Time**

**Sequential Execution**

**Block-level Preemption**

- **RT Task#1 Arriving**
- **no preemption**
- **wait for the running block**
- **no concurrency**
Existing GPU Task Scheduling

**Sequential Execution**

- **Best-effort Task#1**
- **Real-time Task#1**
- **Best-effort Task#2**

**Block-level Preemption**

- **Real-time Task#1 Arriving**
- **Best-effort Task#2 Arriving**

**Preemption**

- Overall Latency
- (RT)Task Latency
- Preemption Latency

**Time**

- No concurrency
- Wait for the running block
- Preemption + execution
Existing GPU Task Scheduling

Best-effort Task#1

Real-time Task#1

Best-effort Task#2

Block-level Preemption

Sequential Execution

RT Task#1 Arriving

BE Task#2 Arriving

wait for the running block

no preemption

no concurrency

preemption + execution

Preemption

Execution

Task Latency (ms)

Number of Best-effort Tasks

Overall Latency

(RT)Task Latency

Preemption Latency

GPU

Task Time

RT Task#1 Arriving

no preemption

GPU

Sequential Execution

Block-level Preemption
Existing GPU Task Scheduling

Sequential Execution

Block-level Preemption

Task Latency (ms)

0 1 2 3 4 5 6 7 8

Preemption

Execution

Number of Best-effort Tasks

0 1 2 4 8

3ms 4ms

RT Task#1 Arriving

BE Task#2 Arriving

no preemption

wait for the running block

no concurrency

preemption + execution

Overall Latency

(RT) Task Latency

Preemption Latency
Existing GPU Task Scheduling

- **Best-effort Task#1**
- **Real-time Task#1**
- **Best-effort Task#2**

**Preemption**

**Latency**

**Overall Latency**

**RT Task**

**BE Task**

**Arriving**

**Time**

**Sequential Execution**

**Block-level Preemption**

**GPU**

**no preemption**

**wait for the running block**

**no concurrency**
Existing GPU Task Scheduling

**Sequential Execution**
- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

**Block-level Preemption**
- Real-time Task#1 Arriving
- Best-effort Task#2 Arriving
- Time

**GPU**
- Block-level Preemption
- Starvation

**Graph**
- BE Throughput (reqs/s)
- RT Frequency (reqs/s)
- Starvation

**Legend**
- Overall Latency
- (RT)Task Latency
- Preemption Latency
Existing GPU Task Scheduling

- **Best-effort Task#1**
- **Real-time Task#1**
- **Best-effort Task#2**

**Preemption**

**Latency**

- **Overall Latency**
- **(RT)Task Latency**
- **Preemption Latency**

**Time**

- **RT Task#1 Arriving**
- **BE Task#2 Arriving**

**Sequential Execution**

- **GPU**
- **no concurrency**

**Block-level Preemption**

- **GPU**
- **wait for the running block**

- (Not always) low latency for **RT** tasks
- **Low throughput** (not work-conserving)
Existing GPU Task Scheduling

- **Best-effort Task#1**
- **Real-time Task#1**
- **Best-effort Task#2**

**Preemption**

**Latency**

**Overall Latency**

**RT Task Latency**

**Preemption Latency**

**RT Task#1 Arriving**
- **BE Task#2 Arriving**

**Sequential Execution**
- **Block-level Preemption**
- **Multi-Streams**
Existing GPU Task Scheduling

Best-effort Task#1

Real-time Task#1

Best-effort Task#2

RT Task#1 Arriving

BE Task#2 Arriving

Time

Sequential Execution

Block-level Preemption

Multi-Streams

Overall Latency

(RT)Task Latency

Preemption Latency

no preemption

wait for the running block

no concurrency

no concurrency
Existing GPU Task Scheduling

- **Best-effort Task #1**: Sequential Execution
  - Block-level Preemption
    - Wait for the running block
    - No concurrency
  - Multi-Streams

- **Real-time Task #1**: No preemption
  - Overall Latency
  - (RT)Task Latency

- **Best-effort Task #2**: No concurrency
  - Preemption Latency

- **RT Task #1 Arriving**: No preemption
- **BE Task #2 Arriving**: Wait for the running block
Existing GPU Task Scheduling

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

**Sequential Execution**
- No preemption
- No concurrency

**Block-level Preemption**
- Wait for the running block
- No concurrency

**Multi-Streams**
- Concurrent
Existing GPU Task Scheduling

Sequential Execution

Block-level Preemption

Multi-Streams

RT Task#1 Arriving ⭐

BE Task#2 Arriving ⭐

Time

Throughput (reqs/s)

Number of Best-effort Tasks

Best-effort Task

Real-time Task
Existing GPU Task Scheduling

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

<table>
<thead>
<tr>
<th>Preemption</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>(RT)Task</td>
<td>Overall</td>
</tr>
<tr>
<td>Latency</td>
<td>Preemption</td>
</tr>
</tbody>
</table>

**Sequential Execution**
- No preemption
- No concurrency
- Wait for the running block

**Block-level Preemption**
- No preemption
- No concurrency

**Multi-Streams**
- Concurrent
- Interference
Existing GPU Task Scheduling

- **Best-effort Task#1**
  - Block
  - Kernel
- **Real-time Task#1**
  - Block
  - Kernel
- **Best-effort Task#2**
  - Block
  - Kernel

**Preemption**
- Overall Latency
- (RT) Task Latency
- Preemption Latency

**Time**
- RT Task#1 Arriving
- BE Task#2 Arriving

**Sequential Execution**
- No preemption
- No concurrency
- Wait for the running block

**Block-level Preemption**
- No concurrency
- Concurrent
- Interference

**Multi-Streams**
- Concurrent
- Interference

Graph:
- Real-time Task:
  - 40ms
- Best-effort Task:
  - 4ms

Number of Best-effort Tasks
Existing GPU Task Scheduling

Sequential Execution

Block-level Preemption

Multi-Streams

- High latency for RT tasks
- High throughput (work-conserving)
Existing GPU Task Scheduling

### Sequential Execution
- High latency for RT tasks
- Low throughput (not work-conserving)

### Block-level Preemption
- (Not always) low latency for RT tasks
- Low throughput (not work-conserving)

### Multi-Streams
- High latency for RT tasks
- High throughput (work-conserving)
### Existing GPU Task Scheduling

**Challenge:** Achieve both
- **low-latency** for RT tasks and
- **work-conserving** for BE tasks

<table>
<thead>
<tr>
<th>Best-effort Task#1</th>
<th>Real-time Task#1</th>
<th>Best-effort Task#2</th>
</tr>
</thead>
<tbody>
<tr>
<td>block</td>
<td>kernel</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **RT Task#1 Arriving**: no preemption
- **BE Task#2 Arriving**: concurrent

**Sequential Execution**

- Block-level Preemption
- Multi-Streams

**Overall Latency**
- (RT)Task Latency
- Preemption Latency
REEF: GPU-accelerated DNN Inference System

Design Goal 1:
- Low-latency for real-time tasks

Design Goal 2:
- Work-conserving for best-effort tasks

Reset-based Preemption:
- μs-scale preemption based on idempotence

Dynamic Kernel Padding:
- controlled concurrent execution based on latency predictability
**REEF overview: architecture**

- **Model Compiler**
- **Code Transformer**
- **Kernel Profiler**
- **Model Pool**
  - esNet
  - gg
  - bert
- **Dynamic Kernel Padding**
- **GPU Runtime**
- **Scheduler**
  - polling
- **Task Queues**
  - BE
  - RT

**Execution modes**:
- **normal mode**:
  - no RT task in RT queue
  - execute BE tasks concurrently
- **real-time mode**:
  - at least one RT task in RT queue
  - execute one RT task a time

**Task queues**:
- one FIFO queue for RT tasks
- several FIFO queues for BE tasks

- **Model Loader**
- **generate DNN models**
REEF overview: scheduling example
REEF overview: scheduling example
REEF overview: scheduling example

Best-effort Task#1

Real-time Task#1

Best-effort Task#2

Block of BE Task

Block of RT Task

☆ Task Arriving

Time

r₁

b₁

GPU
REEF overview: scheduling example

![Diagram of scheduling example]

- **Best-effort Task**
  - Task #1
  - Task #2
- **Real-time Task**
  - Task #1
- **GPU**

- **Block of BE Task**
- **Block of RT Task**
- **Task Arriving**

**Time**
REEF overview: scheduling example

In normal mode, kernels are executed concurrently in **multiple GPU streams**.
REEF overview: scheduling example

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

Block of BE Task
Block of RT Task
Task Arriving

Time
REEF overview: scheduling example

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

Block of BE Task
Block of RT Task
Task Arriving

Time

r₁
b₁
v₁
REEF overview: scheduling example

- **Best-effort Task#1**
- **Real-time Task#1**
- **Best-effort Task#2**

- Block of BE Task
- Block of RT Task
- Task Arriving

**Time**

**Reset-based preemption**
REEF overview: scheduling example

- Block of BE Task
- Block of RT Task
- Task Arriving

Switch to real-time mode
REEF overview: scheduling example

Block of BE Task
Block of RT Task
Task Arriving

Time
REEF overview: scheduling example

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

Block of BE Task
Block of RT Task
Task Arriving

Time

GPU

Block
Kernel

Task Arriving

r_1
b_1
v_1
REEF overview: scheduling example

- Best-effort Task#1
- Real-time Task#1
- Best-effort Task#2

- Block of BE Task
- Block of RT Task
- Task Arriving

dynamic kernel padding
REEF overview: scheduling example

Block of BE Task
Block of RT Task
☆ Task Arriving

switch back to normal mode
REEF overview: scheduling example

- **Low latency** for real-time tasks
  - **Normal Mode**: preempt best-effort tasks in a few μs.
  - **Real-time Mode**: get the GPU resources as many as possible.

- **Work conserving** for best-effort tasks
  - **Normal Mode**: fully utilize GPU resources by using GPU streams.
  - **Real-time Mode**: use the GPU resources leftover by real-time tasks.
REEF overview: scheduling example

- **Low latency** for real-time tasks
  - **Normal Mode**: preempt best-effort tasks in a few μs
  - **Real-time Mode**: get the GPU resources as many as possible

- **Work conserving** for best-effort tasks
  - **Normal Mode**: fully utilize GPU resources by using GPU streams
  - **Real-time Mode**: use the GPU resources leftover by real-time tasks
Reset-Based Preemption

Design Goal:
Preempt concurrent BE tasks in a few μs
# device codes

```python
__global__ void conv_relu(in, weight, out):
1    sum = 0;
2    for i in range(0,3)
3        for j in range(0,3)
4            sum += in[..] * weight[..]
5    out[..] = ReLU(sum)

__global__ void dense(in, weight, bias, out):
6    sum = 0;
7    for i in range(0,512)
8        sum += in[..] * weight[..]
9    out[..] = sum + bias[..]
```
Key Observation

# device codes

__global__ void conv_relu(in, weight, out):
1   sum = 0;
2   for i in range(0,3)
3       for j in range(0,3)
4           sum += in[...].x weight[...]
5       out[...] = ReLU(sum)

__global__ void dense(in, weight, bias, out):
6   sum = 0;
7   for i in range(0,512)
8       sum += in[...].x weight[...]
9   out[...] = sum + bias[...]
Key Observation

Idempotence

Basic idea:
• **Preempt** a task by *killing* the running kernels
• **Restore** a task by *re-executing* the preempted kernels

```
# device codes
__global__ void conv_relu(in, weight, out):
1    sum = 0;
2    for i in range(0,3)
3        for j in range(0,3)
4            sum += in[..] * weight[..]
5    out[..] = ReLU(sum)

__global__ void dense(in, weight, bias, out):
6    sum = 0;
7    for i in range(0,512)
8        sum += in[..] * weight[..]
9    out[..] = sum + bias[..]
```
Reset-based Preemption

Key Idea:
Reset kernels in everywhere
Dynamic Kernel Padding

**Design Goal:**
Allow RT/BE tasks to execute concurrently without interference to RT tasks.
Key Observation

Latency Predictability

Basic Idea:
Allow shorter BE kernels to co-execute with longer RT kernels
Dynamic Kernel Padding

Key Idea:
Dynamically pad RT kernels with BE kernels
Dynamic Kernel Padding

Key Idea:
Dynamically pad RT kernels with BE kernels

RT kernel cannot utilize all CUs

Time

GPU
Dynamic Kernel Padding

Key Idea:
Dynamically pad RT kernels with BE kernels

Pad BE kernels with RT kernel
Dynamic Kernel Padding

Key Idea:
Dynamically *pad* RT kernels with BE kernels

The BE kernel must be *shorter* than RT kernel
Dynamic Kernel Padding

Key Idea:
Dynamically pad RT kernels with BE kernels

Latency Predictability
The BE kernel must be shorter than RT kernel
Dynamic Kernel Padding

Key Idea:
Dynamically pad RT kernels with BE kernels

Latency Predictability
The BE kernel must be shorter than RT kernel
Dynamic Kernel Padding

Key Idea:
Dynamically pad RT kernels with BE kernels
Dynamic Kernel Padding

Key Idea: Dynamically pad RT kernels with BE kernels

Latency Predictability

The BE kernel must be shorter than RT kernel
Dynamic Kernel Padding

Key Idea:
Dynamically pad RT kernels with BE kernels
Dynamic Kernel Padding

Key Idea:
Dynamically pad RT kernels with BE kernels
Evaluation

- **Hardware Environments**
  - AMD Instinct MI50 GPU (60 CUs and 16 GB memory)
  - Intel Core i7-10700 CPU (8 cores) + 16 GB of DRAM

- **Software Environments**
  - ROCm 4.3.0
  - Apache TVM 0.8.0
Evaluation

- **DNN Inference Serving Benchmark (DISB)**
  - A new benchmark for DNN inferences in real-time scenarios
  - Five representative DNN models:
    - ResNet-152 (RNET), DenseNet-201 (DNET), VGG-19 (VGG), Inception-v3 (IN3), DistilBert (BERT)
  - Five workloads

<table>
<thead>
<tr>
<th>DISB</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Num. of RT clients</strong></td>
<td>1/VGG</td>
<td>1/VGG</td>
<td>1/VGG</td>
<td>5/ALL</td>
<td>5/ALL</td>
</tr>
<tr>
<td><strong>Num. of BE clients</strong></td>
<td>1/RNET</td>
<td>1/RNET</td>
<td>5/ALL</td>
<td>5/ALL</td>
<td>5/ALL</td>
</tr>
</tbody>
</table>

Issue **RT** inference requests (fixed frequency)

Issue **BE** inference requests (close-loop)

saturate our testbed
Evaluation

- **DNN Inference Serving Benchmark (DISB)**
  - A new benchmark for DNN inferences in real-time scenarios
  - Five representative DNN models:
    - ResNet-152 (RNET), DenseNet-201 (DNET), VGG-19 (VGG), Inception-v3 (IN3), DistilBert (BERT)
  - Five workloads

- **Real-world Trace**
  - From an open autonomous driving platform (i.e., ApolloAuto)
Evaluation

• Comparing targets
  – **RT-Only**: dedicate the GPU for RT tasks
  – **SEQ**: sequentially execute tasks without preemption
  – **GPUStreams**: execute RT/BE tasks concurrently in multiple GPU streams
REEF only incurs at most 2% latency overhead for RT tasks compared with RT-Only.

REEF can improve overall throughput by $1.14 \times \sim 7.7 \times$ compared with RT-Only.
REEF can avoid starvation for BE tasks.
REEF achieves \textbf{μs-scale} preemption latency
Conclusion

• **REEF: a GPU-accelerated DNN inference serving system**
  – Achieve both **low-latency** (2% latency overhead for real-time tasks) and **work-conserving** (1.14x – 7.7x throughput improvement)
  – Reset-based preemption: μs-scale preemption based on **idempotence**
  – Dynamic kernel padding: controlled concurrent execution based on **latency predictability**

Thanks & QA