Multi-stream Execution in Meta VM

Yaoyao Ding
Collaborated with Haichen Shen
What & Why Multi-Stream Execution

- CUDA stream: support multiple CUDA operations simultaneously
- Operations that can overlap includes:
  - Multiple computation kernels
  - Memory transfer between host and device
  - Data transfer between different CUDA devices and nodes

Multi-Stream Execution allows us to achieve better device utilization.
Multi-Stream Support in Meta VM

Stream-Schedule Operators

- **set_stream(stream_id)**
  Change the current cuda stream index

- **add_event(event_id)**
  Add an event to current stream

- **wait_event(event_id)**
  Let current stream wait given event

Schedule Policies

- **Wavefront Schedule**
  Runs available ops wave by wave.

- **As Soon As Possible (ASAP) Schedule**
  Partition the dataflow graph into chains and run each chain in a stream. Launch ops on critical path first.

- **Inter-Operator Scheduler (IOS) Schedule**
  Use dynamic-programming algorithm to search partition.
Example – Schedule Pass

Before schedule pass (GNF or BBNF)

```
def @main(%x) {
    %0 = atan(%x);
    %1 = atan(%x);
    %2 = atan(%0);
    add(%1, %2)
}
```

After schedule pass (ANF)

```
def @main(%x) {
    let %x_0 = set_stream(0);
    let %x_1 = atan(%x);
    let %x_2 = atan(%x_1);
    let %x_3 = add_event(0);
    let %x_4 = set_stream(1);
    let %x_5 = atan(%x);
    let %x_6 = wait_event(0);
    let %x_7 = add(%x_5, %x_2);
    %x_7
}
```
Example – Multi-Stream Execution

Dataflow Graph
A → B → C → D

VM Bytecode
Invoke A
Invoke B
CudaSetStream 0
Invoke C
CudaSetStream 1
Invoke D
CudaAddEvent 0
CudaWaitEvent 0

Dataflow Graph
A → B → C → D

VM Bytecode
Invoke A
Invoke B
CudaSetStream 0
Invoke C
CudaSetStream 1
Invoke D
CudaWaitEvent 0

(Color indicates stream)
Preliminary Result

Inference

Model: Inception V3
Device: NVIDIA Tesla V100
cuDNN: 7.6.5
Result: up to 1.45x speedup