COMMUNICATION AND COMPUTATION API
COMPOSABILITY
XPU SUPPORT IN OFI

Sean Hefty
Intel Corporation
April 2022
Introduction:
Heterogeneous memory
Computational awareness

NIC-XPU coordination:
Workflow model
XPU initiated communication
NIC initiated computation

Other thoughts:
Restrictions
Exploration
INTRODUCTION
HETEROGENEOUS MEMORY SUPPORT

Conceptual SW model
Want: direct data access and placement

Computational Library
Computational Framework (e.g. oneCCL, NCCL)
Inter-process (IPC)
Communication Framework (e.g. MPI, SHMEM)
Intra-node
Inter-node
Inter/intra device

CUDA
Neuron
oneAPI L0
ROCR
Synapse

Device copies DMA setup
Setup remote DMA
Transmit/receive
Direct memory access

XPU
NIC

Communication Library (libfabric)

Heterogeneous APIs!
Used together but evolve independently

Upstream solution: dmabuf
Fallback: host bounce buffers
INTRODUCTION

COMPUTATIONAL AWARENESS

Conceptual SW flow

Want: correspond directly where execution occurs

CUDA
Neuron
oneAPI L0
ROCR
Synapse

Returning control introduces latency

Optimizations

XPU initiated communication

NIC initiated compute

Heterogeneous HW!

Do not assume implementation!
NIC-XPU COORDINATION

WORKFLOWS

Example: GPU + standard NIC

Example: GPU + RDMA NIC

Example: standard NIC + GPU

Example: RDMA NIC + GPU
NIC-XPU COORDINATION

WORKFLOWS

**Transmit**

- Manager
  - Process
  - Send results
  - Done
  - Example: need to coordinate with recipient

- ComX
  - Receive
  - Results
  - Example: GPU + Integrated NIC

**Receive**

- Manager
  - Do recv
  - Process
  - Done
  - Example: need to coordinate other data

- ComX
  - Data
  - Example: GPU + Integrated NIC

- Manager
  - Process & send results
  - Done
  - Example: can immediately send results

- ComX
  - Data
  - Example: GPU + Integrated NIC

- Manager
  - Do recv & process
  - Done
  - Example: can immediately process data

- ComX
  - Data
  - Example: GPU + Integrated NIC
Manager

Computational directive (API)

process
results

Want: common model

extremes

send results

Communication directive (API)

Manager

Comp

Comm

results

ComX

Manager
done

data

Example: GPU + standard NIC

Example: GPU + Integrated NIC

Challenge:
Find a coordinating solution without assuming HW implementation on either side!

Critical Challenge:
Define APIs that mere humans can figure out how to use!
Abstract NIC

- Model: CPU writes command
- XPU rings doorbell

Less abstract NIC

- Variable sized queue entries
- Valid flags
- NIC can prefetch entries (no doorbell)
- Valid entry count

Simple model would be easy to use – start from here

Details NIC (not vendor) specific!
XPU INITIATED COMMUNICATION

PHASE I: XPU TRIGGERED OPS

Middleware Framework

1 prepare send

Compute Library

3 request work & notify

4 schedule work

libfabric

2 deferred send

7 send done

NIC

5 data ready

6 direct memory access

XPU

3 request work & notify

1 prepare send

2 deferred send

7 send done

NIC provider setups notification

Can use dmabuf

XPU signals NIC directly

Combines triggered + heterogeneous memory APIs

Can use dmabuf

Compute Library

Middleware Framework

XPU

NIC

6 direct memory access

5 data ready

3 request work & notify

4 schedule work

2 deferred send

7 send done

 NIC provider setups notification

Can use dmabuf

XPU signals NIC directly

Combines triggered + heterogeneous memory APIs

Can use dmabuf

Compute Library

Middleware Framework

XPU

NIC

6 direct memory access

5 data ready

3 request work & notify

4 schedule work

2 deferred send

7 send done

 NIC provider setups notification

Can use dmabuf

XPU signals NIC directly

Combines triggered + heterogeneous memory APIs

Can use dmabuf

Compute Library

Middleware Framework

XPU

NIC

6 direct memory access

5 data ready

3 request work & notify

4 schedule work

2 deferred send

7 send done

 NIC provider setups notification

Can use dmabuf

XPU signals NIC directly

Combines triggered + heterogeneous memory APIs

Can use dmabuf

Compute Library

Middleware Framework

XPU

NIC

6 direct memory access

5 data ready

3 request work & notify

4 schedule work

2 deferred send

7 send done

 NIC provider setups notification

Can use dmabuf

XPU signals NIC directly

Combines triggered + heterogeneous memory APIs

Can use dmabuf
**XPU INITIATED COMMUNICATION**

**PHASE I: XPU TRIGGERED OPS**

- **Universal triggering mechanism**
  - Supports broad range of NIC implementations

**CPU queues XPU triggerable transfer**

```
fi_writemsg(endpoint, msg, FI_TRIGGER)
```

```c
struct fi_triggered_xpu {
    enum fi_hmem_iface iface;
    union {
        int cuda;
        int ze;
    } device;
    int count;
    struct fi_trigger_var var[];
};
```

**Writing to variables trigger the transfer**

```c
struct fi_triggered_var {
    enum fi_datatype data_type;
    int count
    void *addr;
    union {
        uint8_t val8;
        uint16_t val16;
        uint32_t val32;
        uint64_t val64;
        uint8_t *data
    } value;
};
```

**Likely: couple variables**

- Compute side requirement:
  - write memory
  - (possibly across PCI)
Wait! Is there an easier way?

Model: XPU waits for completion

Simple model is not viable

Can we handle errors?

RDMA doesn’t generate target completions

Predict which entry will match with specific inbound data

Multiple receive queues feed into cq

XPU needs to poll off-device memory

Typical setup
What could the app do?

- Coordinate transfer with peer
- Data (RDMA write)
- Signal (RDMA write)
- Wait for signal then process
- Direct memory access
- Data followed by signal

Manager

Comp

• Works with standard and offload NICs
• App could watch multiple signals
• XPU needs ability to poll on local memory change

App writes ‘completion’ to known location

Use atomic add to receive from multiple peers
OTHER THOUGHTS

RESTRICTIONS

- **XPU triggerable endpoint not sharable with host**
  - Increases resources
  - Significant impact for underlying connections

- **Triggers must be signaled in order**
  - *Some* NICs may allow signaling out of order, but will be processed in order

- **Want write-after-write ordering**
  - Or delay writing signal until data transfer complete

- **Defer processing until data from all peers have arrived**
  - Atomic add can help here

---

Proposal: application managed command windows

May need new network protocols – write + atomic add
struct fi_tx_window {
    void *base;
    size_t length;
};

struct fi_tx_var {
    struct fi_tx_window *window;
    size_t offset;
    size_t window_increment;
    size_t size;
    uint8_t data[];
};

memcpy(window->base + tx_var->offset, tx_var->data, tx_var->size);
window->base += tx_var->window_increment;
OTHER THOUGHTS
EXPLORATION

NIC initiated computation

I got nothing.
THANK YOU