LIBFABRIC INTRANODE DEVICE SUPPORT

Alexia Ingerson

Intel Corp.
OVERVIEW

Is this memory on the host or device?

Is this memory going to the host or another device?

memcpy

CMA

shared memory

memcpy

IPC

shared memory

memcpy

CPU/Host memory

GPU/device memory

CPU/Host memory

GPU/device memory
OFI API CHANGES

- **info->caps: Fl_HMEM**
  - Requests support for transfers to and from device memory

- **domain_attr->mr_mode: Fl_MR_HMEM**
  - Specifies that the application should register all device memory with proper interfaces
  - Eliminates the need for a provider to query devices in order to determine memory location (expensive)

- **fi_mr_attr->iface**
  - Indicates software interface used to manage memory region
    ```c
    enum fi_hmem_iface {
      FI_HMEM_SYSTEM = 0, // system/host memory
      FI_HMEM_CUDA, // Nvidia/CUDA memory (libcuda)
      FI_HMEM_ZE, // Intel/Ze memory (libze_loader)
      ...
    }
    ```
  - Tells provider which API calls to use when copying to and from device

- **fi_mr_attr->device**
  - Device identifier for HMEM memory
  - Indicates on which device the memory is located on (type varies by interface) when multiple devices are present
Applications with GPU support

HMEM Hook

Device software interface

MR Cache for HMEM

Provider (verbs;ofi_rxm, shm, etc)

Register HMEM, Request transfers

Host or device address?

For existing applications or middlewares that are not GPU aware and need to rediscover location of buffers

Optimal when memory location is known and carried through stack

May optimize host vs HMEM transfers differently

OFI w/ HMEM Support

© OpenFabrics Alliance
**smr_region**
EP initialized info / resources

- Command Queue
- Response Queue
- Inject Pool

- Single command queue for incoming messages
- Response queue for messages requiring an ACK
- Pool of bounce buffers for medium-sized messages
**SHM PROTOCOLS**

- **Rx Command Queue**
  - Device memcpy
  - Inline

- **Inject**
  - Device->host/host->device copies tend to be expensive
  - Rx Command Queue
  - BUFF

- **IOV**
  - Rx Command Queue
  - RESP
  - Get/open IPC handle

© OpenFabrics Alliance

OpenFabrics Alliance Workshop 2020
DEVICE BOUNCE BUFFERS

**smr_region**
EP initialized info / resources

- Command Queue
- Response Queue
- Inject Pool
- IPC Handle Pool

```c
smr_ipc_handle {
    void    *buf;
    uint8_t  handle[64];
}
```

- Single command queue for incomings messages
- Response queue for messages requiring an ACK
- Pool of bounce buffers for medium-sized messages
- Pool of device buffer IPC handles
DEVICE BOUNCE BUFFER PROTOCOL

- Rx Command Queue
- Save bounce buffer offset into cmd
- Use IPC buffer::handle to do IPC copy into peer device buffer
struct ofi_hmem_ops {
    (*init)();
    (*cleanup)();
    (*copy_to_hmem)();
    (*copy_from_hmem)();
    (*get_handle)();
    (*open_handle)();
    (*close_handle)();
    (*alloc)();
    (*free)();
};
<table>
<thead>
<tr>
<th>Sample HMEM Ops</th>
<th>Intel GPU Level 0 API (Ze)</th>
<th>Nvidia GPU CUDA API</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INIT</strong></td>
<td>zeInit()</td>
<td>cuInit()</td>
</tr>
<tr>
<td></td>
<td>zeDriverGet()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>zeCommandQueueCreate()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>zeCommandListCreate()</td>
<td></td>
</tr>
<tr>
<td><strong>COPY</strong></td>
<td>zeCommandListAppendMemoryCopy()</td>
<td>cudaMemcpy()</td>
</tr>
<tr>
<td></td>
<td>zeCommandQueueExecuteCommandLists()</td>
<td></td>
</tr>
<tr>
<td><strong>IPC</strong></td>
<td>zeDriverGetMemIpcHandle()</td>
<td>cudaMemcpy()</td>
</tr>
<tr>
<td></td>
<td>zeDriverOpenMemIpcHandle()</td>
<td>cudaIpcGetMemHandle()</td>
</tr>
<tr>
<td></td>
<td>zeDriverCloseMemIpcHandle()</td>
<td>cudaIpcOpenHandle()</td>
</tr>
<tr>
<td><strong>MEM</strong></td>
<td>zeDriverAllocSharedMem()</td>
<td>cudaMalloc()</td>
</tr>
<tr>
<td></td>
<td>zeDriverFreeMem()</td>
<td>cudaFree()</td>
</tr>
</tbody>
</table>