**OFI WG telecon – 10/09/2018**

**Agenda:**

1. Opens
2. Accelerators

**Opens**

RPM stuff – Doug assembling next draft of the HA Whitepaper. Asking for a 1:1 to discuss some comments. The document needed a significant re-write because RDMA is shot through the entire document and one of our key comments was that the whitepaper should be generalized to be more independent of today’s RDMA.

**Discussion – Introduction on accelerators**

No slides available today.

Possible overlap between accelerators and Smart NICs/FPGAs

Only thing obvious so far is related to memory registration. Need a way to indicate that the target is not on the CPU.

CPU would not use libfabric to communicate with the GPU.

Libfabric is not a CUDA replacement.

Do you want to allow a GPU to launch a data transfer via libfabric?

Any overlap with the RPM stuff? Not clear at this point, such an overlap may emerge. Whatever is being done for the current HA Whitepaper should not preclude that from occurring, and a GPU should be able to access remote persistent memory.

One case: a GPU accessing remote persistent memory. The RPM API enhancements should allow a GPU to do this.

Second case: consider a GPU as an accelerator performing a higher-level function e.g. mirroring. The GPU should be able to use libfabric to execute its higher-level function just like any other ‘application’.

What is the scope of libfabric w.r.t. a GPU?

Three cases:

1. An operation launched by a host CPU targeting a GPU for data transfer, and
2. An operation launched by the GPU itself where GPU memory is the target, and
3. An operation launched by the GPU where host memory is the target.

Is there a requirement (limitation) that the target address space be memory-mapped into the process’ memory space?

**“Hook Provider” in libfabric is a layer on top of a provider that allows e.g. the collection of debug or performance data**

Request to have more than one hook installed at a time?

Is there a mechanism whereby an application can discover what hooks are available? Can the hooking capability be exposed to an application?

Originally, hooks were intended to support the developer. But making them accessible to an application may enable new uses for them. At the moment, hooks are transparent to the application, they are enabled via an environment variable.

To do this, should hooks be looked at as a new kind of utility provider? Should they be looked at as an extended capability of an existing provider (including a utility provider)?

Consider the idea of a ‘hook registration’ API, allowing a hook to register itself, exposing the hooks outside of libfabric. This would also enable the development of additional ‘custom’ hooks.

**Next meeting**

Tuesday, October 23, 2018

9:00 – 10:00AM PST

**Recording:**

|  |
| --- |
| **Every 2 week OFIWG/libfabric meeting (2018)-20181009 1610-1** |
| Tuesday, October 9, 2018 |
| 12:10 pm  |  Eastern Daylight Time (New York, GMT-04:00) |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| [**Play recording**](https://cisco.webex.com/cisco/lsr.php?RCID=dc060b8e7b7e4f2f8517fa70ff3d1094) (39 min 39 sec) | |
| Recording password: xEuC9dvS |  |

|  |
| --- |
|  |

**Webex link:** See the OFA central calendar for meeting logistics. <https://openfabrics.org/index.php/ofa-calendar.html>

**OFIWG Download Site:** [www.openfabrics.org/downloads/OFIWG](http://www.openfabrics.org/downloads/OFIWG)

**Github:** <https://github.com/ofiwg/libfabric>

**OFI Software Download Site:** [www.openfabrics.org/downloads/OFI](http://www.openfabrics.org/downloads/OFIWG)