rdma-core Update

UCF Annual Meeting

Dec 2019
Changes this Year

- Soft iWarp provider (siw)
- EFA provider
- Incorporate `infiniband-diags` and `libibmad` into `rdma-core`
- User Azure Pipelines for CI instead of Travis
- Kernel module autoloading
- RDMA kernel device renaming and stable device names
- More pyverbs features and pyverbs based testing
- External QP for RDMA-CM
mlx5 Changes

- High rate on-chip steering rule manipulation
- PCI Atomics for RDMA
- ODP support for more scenarios
- Asynchronous devx command execution
- Asynchronous event delivery
mlx5 DevX/DV/etc

- Fully up streamed and in several distros
- Continuing to migrate software away from MLNX_OFED specific APIs
GPU Controlled QPs

- Allow the GPU to post work to a QP

- A small part of the provider driver must live in the GPU
  - GPU uses DV APIs to directly access the QP ring memory
  - Work Queue Entries are written by the GPU
  - Doorbell rings are generated by the GPU

- The QP ring memory and other objects can be placed on the GPU
  - https://patchwork.kernel.org/patch/11175987/
  - Application gets call backs when the driver needs memory
  - Callback knows what the intended memory usage is, can allocate in the appropriate way
GPU Providers

- Currently no way to provide the provider GPU ‘dv’ code as a library
  - Currently mlx5dv is the only WQE DV in rdma-core, so not an immediate problem
  - Generally a poor situation
Relaxed Ordering MR

- Allow the user to request relaxed ordering for RDMA_READ and RDMA_WRITE on a per-MR basis

- Relaxed ordering is something largely implied by the IBA, but some degenerate cases require the app to opt in:
  - Concurrent RDMA based read/write to the same address may not be fully ordered
  - Interaction with PCI peer devices becomes un-ordered
  - Data becomes visible to host memory in no defined order (i.e., memory polling for completion is not possible)

- Some CPU designs require relaxed ordering for performance

- Will be updated all appropriate verbs users to set the relaxed ordering flags