Cornelis Networks Omni-path Express (OPX) Libfabric provider – Observability and Tuning

Tim Thompson - Cornelis Networks

August 2023

Notices and Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH CORNELIS NETWORKS PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN CORNELIS NETWORKS'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, CORNELIS NETWORKS ASSUMES NO LIABILITY WHATSOEVER, AND CORNELIS NETWORKS DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF CORNELIS NETWORKS PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. CORNELIS NETWORKS PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.

Cornelis Networks may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Cornelis Networks reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice. Roadmap not reflective of exact launch granularity and timing. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Any code names featured are used internally within Cornelis Networks to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Cornelis Networks to use code names in advertising, promotion or marketing of any product or services and any such use of Cornelis Networks' internal code names is at the sole risk of the user.

All products, computer systems, dates and figures specified are preliminary based on current expectations and are subject to change without notice. Material in this presentation is intended as product positioning and not approved end user messaging.

Performance tests are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Cornelis Networks technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Performance varies depending on system configuration.

Agenda

- Who is Cornelis Networks?
- What is OPX and how to use it?
- OPX features and topology
- How to assess and tune Fabric performance
- Observability with OPX

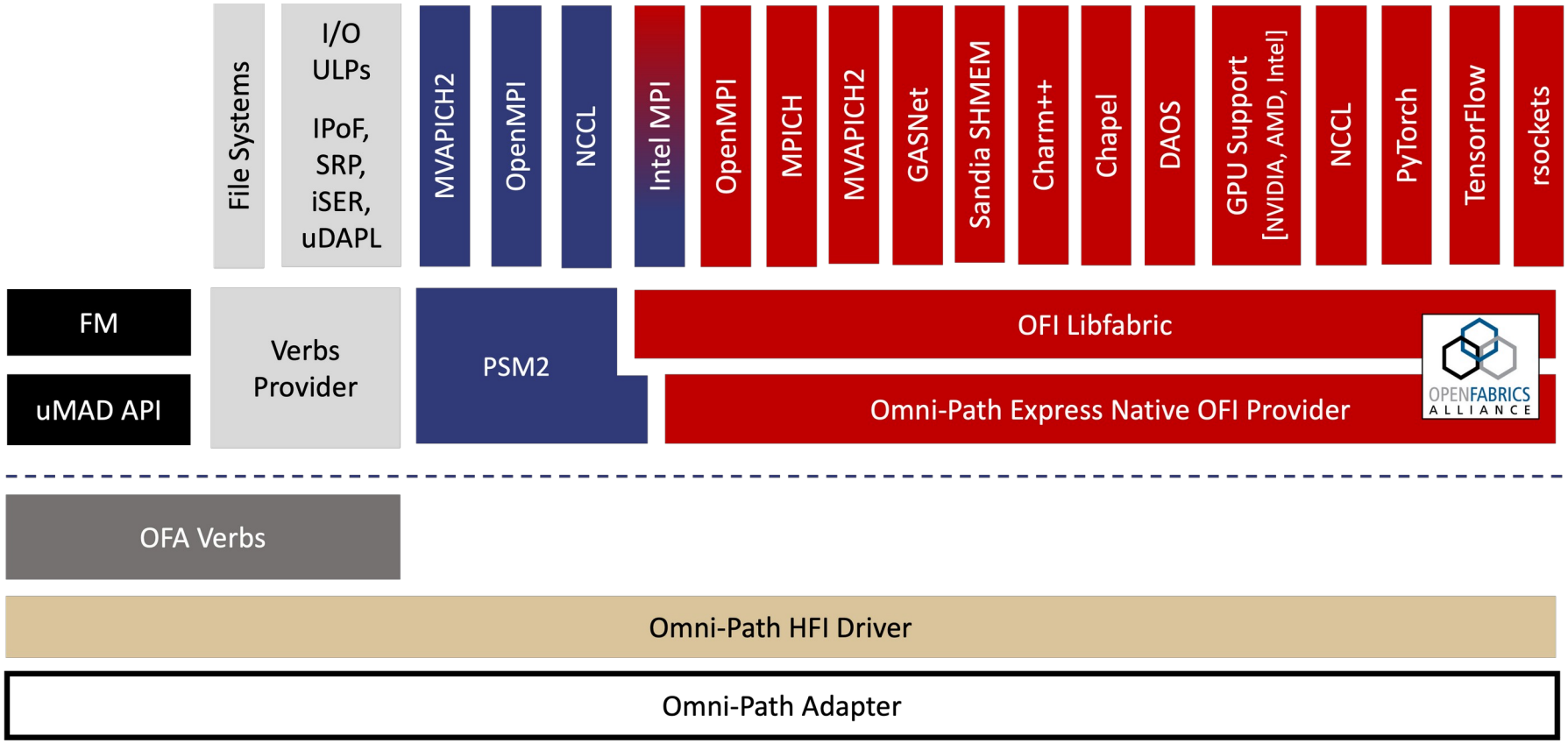
Cornelis Networks

- From Startup -> Silverstorm -> QLogic (IB) -> Intel (OPA) -> Cornelis (OPX)
- Omni-Path Architecture (OPA) 100 Gbps Fabric
- Spun out of Intel 2+ years ago
  - Bring the customers and technology
- Carry OPA flag forward and advance the technology
- Next Generation is 400Gbps (CN 5000)
- New fabric topologies

What is OPX?

- OPX is a libfabric (OFI) provider
- Replacing PSM2
  - Not a re-write or refactor of PSM2
  - Origins in BGQ (Blue Gene) provider
- Highly optimized
  - Support for RDMA and GPU RDMA
  - MVAPICH3.0b w/libfabric
- Does not require changes to hfi1 driver or Fabric Manager
- Really is a drop in replacement for PSM2

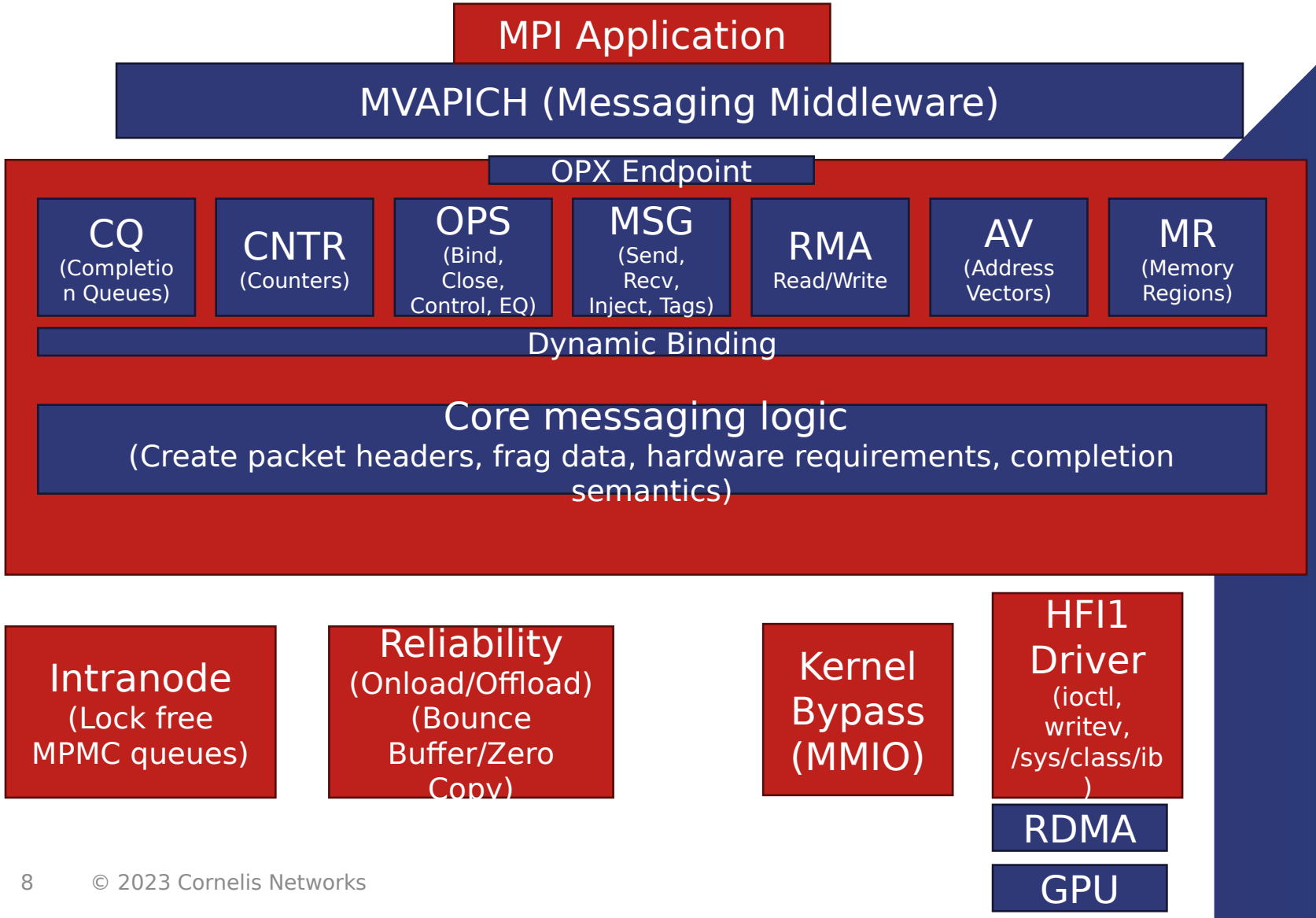
Software Stack



How to use OPX?

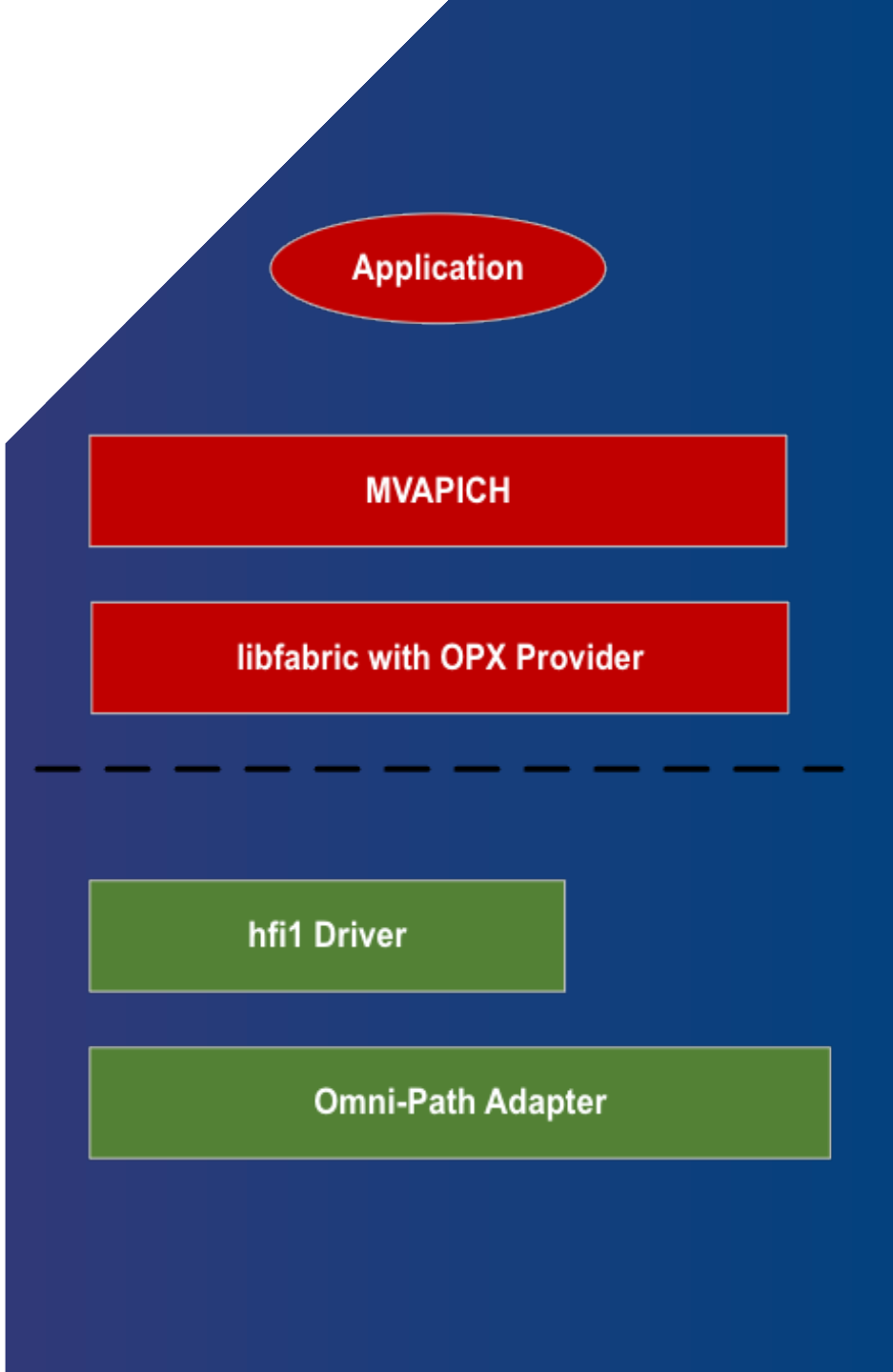
- Best-practice, install Cornelis OPXS package
- Optional: Download and compile Libfabric upstream and upstream Linux kernel with newer hfi1 driver
- Build Libfabric
- Make sure MVAPICH is not configured for internal-only provider
- Set LD_LIBRARY_PATH=/mylib/libfabric.so: $LD_LIBRARAY_PATH
- Set FI_PROVIDER=opx
- First run check, set FI_LOG_LEVEL=info and look for "**HFI1 PIO credits**". If you see this, you're running the OPX provider

OPX provider topology



Many tuning knobs split across many levels

- Direct
  - Application tuning knobs and #define, MPI_THREAD_MODEL
  - MVAPICH ENVs (prefix MV2_*), #define, anything else?
  - Libfabric and OPX (prefix FI_*, FI_OPX_*), #define
  - Hfi1 device driver (/sys/module/hfi1/parameters)
  - Fabric Service Level, Virtual Lane, and FGAR
- Indirect
  - Linux kernel/distro tuning and de-jittering, hugepages
  - HPC job topology <-> Fabric topology
  - Compute Node BIOS settings (c-state, NUMA, PCIe, etc)
  - Package versions (Spack, Ansible, Docker, …)



OPX Libfabric Endpoint config

- Locking vs Non-locking
  - FI_THREAD_DOMAIN, FI_THREAD_ENDPOINT, FI_THREAD_COMPLETION – most performant
  - OPX has high-level locking for thread safety.
- Manual vs Auto progress
  - Onload-type reliability makes all flows RX/TX. Polling for progress via cq_read allows OPX to do progress work. Better if MPI calls this (this is the manual progress model)
  - Auto progress works, will run threadsafe, and spawn an external thread to call cq_read to ‘progress’ the provider. Not as fast
- AV_TABLE vs AV_MAP
  - AV_MAP is more tested and faster (maybe). Libfabric2 might remove
- CQ_COMPLETION – When is an op ‘complete’ ?
  - FI_INJECT_COMPLETE vs FI_TRANSMIT_COMPLETE vs FI_DELIVERY_COMPLETE
  - FI_SELECTIVE_COMPLETION
- Onload vs Offload reliability
  - Onload model only supported right now. Offload model might be coming
- FI_CONTEXT_2 – Only provider that requires this?

OPX ENVs

- Documentation in OPXS docs and Libfabric man page: fi_opx(7) (ofiwg.github.io)
- Best-practice: Run fi_info with your target Libfabric to print the OPX provider ENVs:
  fi_info -g FI_OPX
- **FI_OPX_HFI_SELECT** – How to deal with multiple HFIs and NUMA
- **FI_OPX_DELIVERY_COMPLETION_THRESHOLD** – Bounce buffers
- **FI_OPX_EXPECTED_RECEIVE_ENABLE** - RDMA
- **FI_OPX_RELIABILITY_SERVICE_USEC_MAX** – Reliability Ping
- **FI_OPX_RELIABILITY_SERVICE_PRE_ACK_RATE** – Send window

Hfi1 driver inspection/tuning

- Hfi1 device driver has many tunable module parms
- List them all with this command (on a system that has OPXS installed and hfi1 loaded):
- **grep . /sys/module/hfi1/parameters/***
  - Many parms, psm2 tuning guide has advice on tuning these, and it works pretty well for OPX
  - **rcvhdrcnt**: Depth of the Eager Rx rings for each context. Default 2048. Recommend 8192 for larger HPC jobs. The larger value will make the memory footprint larger
  - **num_user_contexts**: Maximum number of ranks per hfi1 adapter in this compute node. Default is -1, which means assume max ranks. Setting this value to a number that is equal to the number of ranks you expect to launch (probably the number of cores?) MIGHT increase performance by allocating extra TX send buffer. The lower the number, the more credits.

OPX Observability - Logs

- FI_LOG_LEVEL=WARN or TRACE or INFO or DEBUG or MAX
- Optimized or Debug builds, these are valid for both. Optimized builds skip much logging
- OPX is a potential data firehose of text

```
libfabric:34638:1680714711::opx:fabric:opx_hfi_wait_for_device():163<info> Found /dev/hfi1_0 after 0.0 seconds
libfabric:34638:1680714711::opx:fabric:_hfi_cmd_ioctl():352<info> command OPX_HFI_CMD 0XC, HFI1_IOCTL 0X80041BEE
libfabric:34638:1680714711::opx:fabric:_hfi_cmd_ioctl():352<info> command OPX_HFI_CMD 0, HFI1_IOCTL 0XC01C1BE1
libfabric:34638:1680714711::opx:fabric:_hfi_cmd_ioctl():352<info> command OPX_HFI_CMD 0X1, HFI1_IOCTL 0X40281BE2
libfabric:34638:1680714711::opx:fabric:opx_hfi_userinit_internal():548<info> CONTEXT INIT ctxtinfo: active 1, unit 0, ctxt 11, subctxt 0
libfabric:34638:1680714711::opx:fabric:opx_hfi_userinit_internal():550<info> CONTEXT INIT ctxtinfo: rcvtids 1400, credits 361
libfabric:34638:1680714711::opx:fabric:opx_hfi_userinit_internal():552<info> CONTEXT INIT ctxtinfo: numa 0, cpu 0, send_ctxt 139
libfabric:34638:1680714711::opx:fabric:opx_hfi_userinit_internal():554<info> CONTEXT INIT ctxtinfo: rcvhdrq_cnt 2048, rcvhdrq_entsize 128
libfabric:34638:1680714711::opx:fabric:opx_hfi_userinit_internal():556<info> CONTEXT INIT ctxtinfo: egrtids 32, sdma_ring_size 128
libfabric:34638:1680714711::opx:fabric:_hfi_cmd_ioctl():352<info> command OPX_HFI_CMD 0X2, HFI1_IOCTL 0X40781BE3
libfabric:34638:1680714711::opx:fabric:opx_hfi_userinit_internal():573<info> CONTEXT INIT baseinfo: hwver 3020710, swver 60003, jkey 59371, qp 128
libfabric:34638:1680714711::opx:fabric:opx_hfi_userinit_internal():576<info> CONTEXT INIT baseinfo: credit_addr dabbad00030b02d8, sop dabbad00020b0000, pio dabbad00010b0000
libfabric:34638:1680714711::opx:fabric:opx_hfi_userinit_internal():579<info> CONTEXT INIT baseinfo: hdrbase dabbad00040b0000, egrbase dabbad00050b0000, sdmabase dabbad000d0b0000
libfabric:34638:1680714711::opx:fabric:opx_hfi_userinit_internal():583<info> CONTEXT INIT baseinfo: ureg dabbad00060b0000, eventbase dabbad00070b0200, statusbase dabbad00080b0000, tailaddr 0
libfabric:34638:1680714711::opx:fabric:fi_opx_hfi1_context_open():505<info> Selected HFI is 0; caller NUMA domain is 0; HFI NUMA domain is 0
libfabric:34638:1680714711::opx:fabric:fi_opx_hfi1_context_open():515<info> Selected HFI unit 0 in the same numa node as this pid.
libfabric:34638:1680714711::opx:fabric:_hfi_cmd_ioctl():352<info> command OPX_HFI_CMD 0X9, HFI1_IOCTL 0X40021BEB
libfabric:34638:1680714711::opx:core:fi_param_get_():279<info> variable selinux=<not set>
libfabric:34638:1680714711::opx:fabric:fi_opx_hfi1_context_open():656<info> Context configured with HFI=0 PORT=1 LID=0x2 JKEY=59371
libfabric:34638:1680714711::opx:domain:fi_opx_timer_init():118<info> Cycle timer is not available due to cpu affinity, using clock_gettime
libfabric:34638:1680714711::opx:core:fi_param_get_():279<info> variable reliability_service_pre_ack_rate=<not set>
libfabric:34638:1680714711::opx:ep_data:fi_opx_reliability_service_init():2244<trace> FI_OPX_RELIABILITY_SERVICE_PRE_ACK_RATE not specified, using default value of 64
libfabric:34638:1680714711::opx:core:fi_param_get_():279<info> variable reliability_service_usec_max=<not set>
libfabric:34638:1680714711::opx:ep_data:fi_opx_reliability_service_init():2261<trace> FI_OPX_RELIABILITY_SERVICE_USEC_MAX not specified, using default value of 500
libfabric:34638:1680714711::opx:core:fi_param_get_():279<info> variable reliability_service_nack_threshold=<not set>
libfabric:34638:1680714711::opx:ep_data:fi_opx_reliability_service_init():2281<trace> FI_OPX_RELIABILITY_SERVICE_NACK_THRESHOLD not specified, using default value of 1
libfabric:34638:1680714711::opx:ep_data:fi_opx_open_command_queues():1349<info> HFI1 PIO credits: 361
libfabric:34638:1680714711::opx:ep_data:fi_opx_ep_tx_init():792<info> Credits_total is 361, so set pio_max_eager_tx_bytes to 8192
libfabric:34638:1680714711::opx:ep_data:fi_opx_ep_tx_init():810<info> Set pio_flow_eager_tx_bytes to 8192
libfabric:34638:1680714711::opx:core:fi_param_get_():279<info> variable delivery_completion_threshold=<not set>
libfabric:34638:1680714711::opx:ep_data:fi_opx_ep_tx_init():821<info> FI_OPX_DELIVERY_COMPLETION_THRESHOLD not set.  Using default setting of 16385
libfabric:34638:1680714711::opx:ep_data:fi_opx_ep_tx_init():834<info> Multi-packet eager max message length is 16384, chunk-size is 4160.
libfabric:34638:1680714711::opx:core:fi_param_get_():279<info> variable sdma_disable=<not set>
libfabric:34638:1680714711::opx:ep_data:fi_opx_ep_tx_init():849<info> sdma_disable parm not specified; using SDMA
libfabric:34638:1680714711::opx:core:fi_param_get_():279<info> variable expected_receive_enable=<not set>
libfabric:34638:1680714711::opx:ep_data:fi_opx_endpoint_rx_tx():1968<info> expected_receive_enable parm not specified; disabled expected receive rendezvous
```

OPX Observability – Debug Counters

- Requires debug build of Libfabric AND `#define OPX_DEBUG_COUNTERS`
- Will affecting timing/race conditions

```
(1346422) ### DEBUG COUNTERS ###
(1346422) ### mp_eager.send_first_packets                          347136
(1346422) ### mp_eager.send_nth_packets                            1041408
(1346422) ### mp_eager.send_first_force_cr                         0
(1346422) ### mp_eager.send_nth_force_cr                           0
(1346422) ### mp_eager.send_fall_back_to_rzv                       0
(1346422) ### mp_eager.send_full_replay_buffer_rx_poll             2072
(1346422) ### mp_eager.recv_max_ue_queue_length                    0
(1346422) ### mp_eager.recv_max_mq_queue_length                    0
(1346422) ### mp_eager.recv_first_packets                          347140
(1346422) ### mp_eager.recv_nth_packets                            1041412
(1346422) ### mp_eager.recv_completed_process_context              0
(1346422) ### mp_eager.recv_completed_eager_first                  0
(1346422) ### mp_eager.recv_completed_eager_nth                    347140
(1346422) ### mp_eager.recv_truncation                             0
(1346422) ### mp_eager.recv_nth_no_match                           0
(1346422) ### mp_eager.recv_nth_match                              1041412
(1346422) ### mp_eager_recv_total_completed ----------> 347140
(1346422) ### mp_eager_recv_truncation                  0
(1346422) ### reliability_ping.acks_sent                           74446
(1346422) ### reliability_ping.acks_preemptive_sent                1243853
(1346422) ### reliability_ping.acks_received                       1310648
(1346422) ### reliability_ping.acks_ignored                        31752
(1346422) ### reliability_ping.nacks_sent                          133
(1346422) ### reliability_ping.nacks_preemptive_sent               0
(1346422) ### reliability_ping.nacks_received                      192
(1346422) ### reliability_ping.nacks_ignored                       0
(1346422) ### reliability_ping.pings_sent                          92072
(1346422) ### reliability_ping.pings_received                      74579
```

Current Status

- MVAPICH2 numbers look performant, good out of box experience
- OPX Code is upstream, recommend v1.19 of Libfabric.
- New features
  - Large message improvements – upstream but not default on (update hfi1 driver first)
  - GPU support – Almost up streamed
- Available on GitHub, Distro*, OPXS Software Suite
  - Checkout Libfabric ‘main’ branch
- Get involved
  - Happy to take patches via GitHub

Thank You

www.cornelisnetworks.com