Hewlett Packard
Enterprise

Introduction To Libfabric Environment Variables

Jesse Treger & Ian Ziemba, HPE

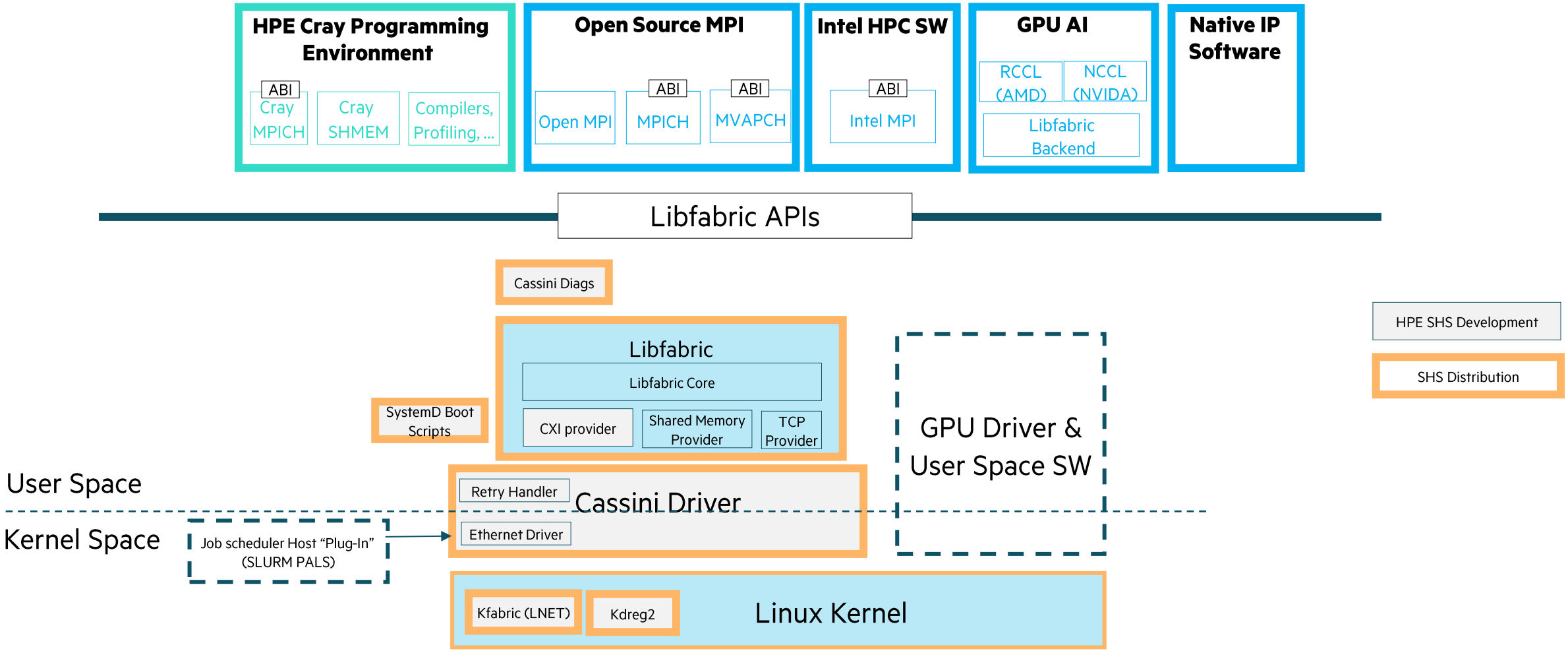
May 4, 2025

Agenda

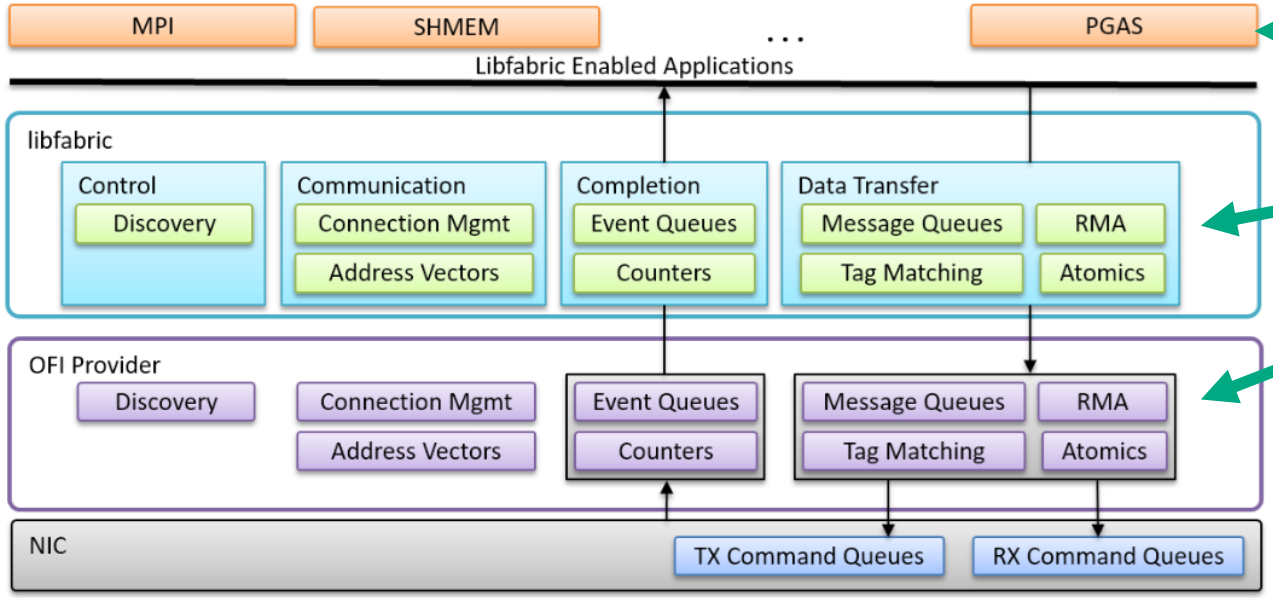
- Comprehensive but accelerated overview of Libfabric variables that may need user/site configuration
  - This material will eventually make its way into user documentation
  - Man pages are today the primary place the development team documents these
  - Refer to downloadable presentation

- Today's outline
  - SHS host software stack and Libfabric
  - Environment settings
    - –Memory cache monitor options
    - –Queue sizes
    - –Matching modes & unexpected messages
    - –Rendezvous protocol
  - Common scenarios/Debugging/Future Considerations

HPE Slingshot NIC Host Software Stack (Conceptual View)



Libfabric Overview



Libfabric Applications

Libfabric Core/APIs

Libfabric "Providers"

- **CXI provide (Cassini)**
- Verbs (for SS10 Systems/IB)
- TCP/IP (for Ethernet)
- Shared Memory Provider
- LinkX Provider ("Experimental")
- Other networks (Omnipath, Elastic Fabric, Adapter, Cisco, …)
- Future: Ultra Ethernet

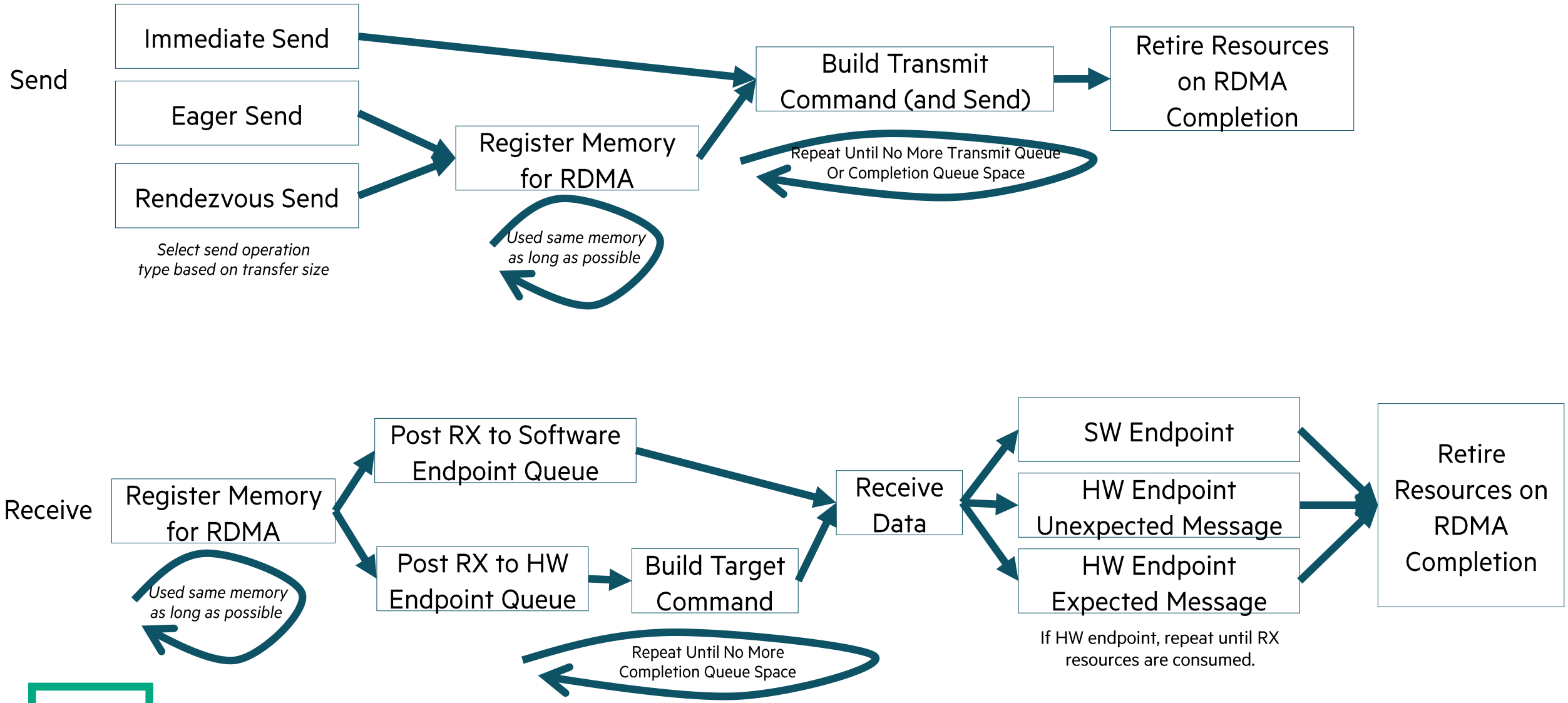
- Low-level communication library that abstracts diverse networking technologies.
- Developed by the OFI Working Group (subgroup of OpenFabrics Alliance – OFA)
- Defines interfaces that enable a tight semantic map between applications and underlying fabric services
- Co-designed with fabric HW providers and application developers with a focus on needs of HPC users.
- Supports multiple communication semantics, is fabric and HW implementation agnostic, leverages the existing RDMA open-source community.
- Designed to minimize the impedance mismatch between apps including middleware such as MPI, SHMEM, data storage - and fabric hardware.
- Target high-bandwidth, low-latency NICs, with a goal to scale to tens of thousands of nodes

Sources: https://ofiwg.github.io, https://www.cs.mtsu.edu/~waderholdt/6430/papers/07312664.pdf

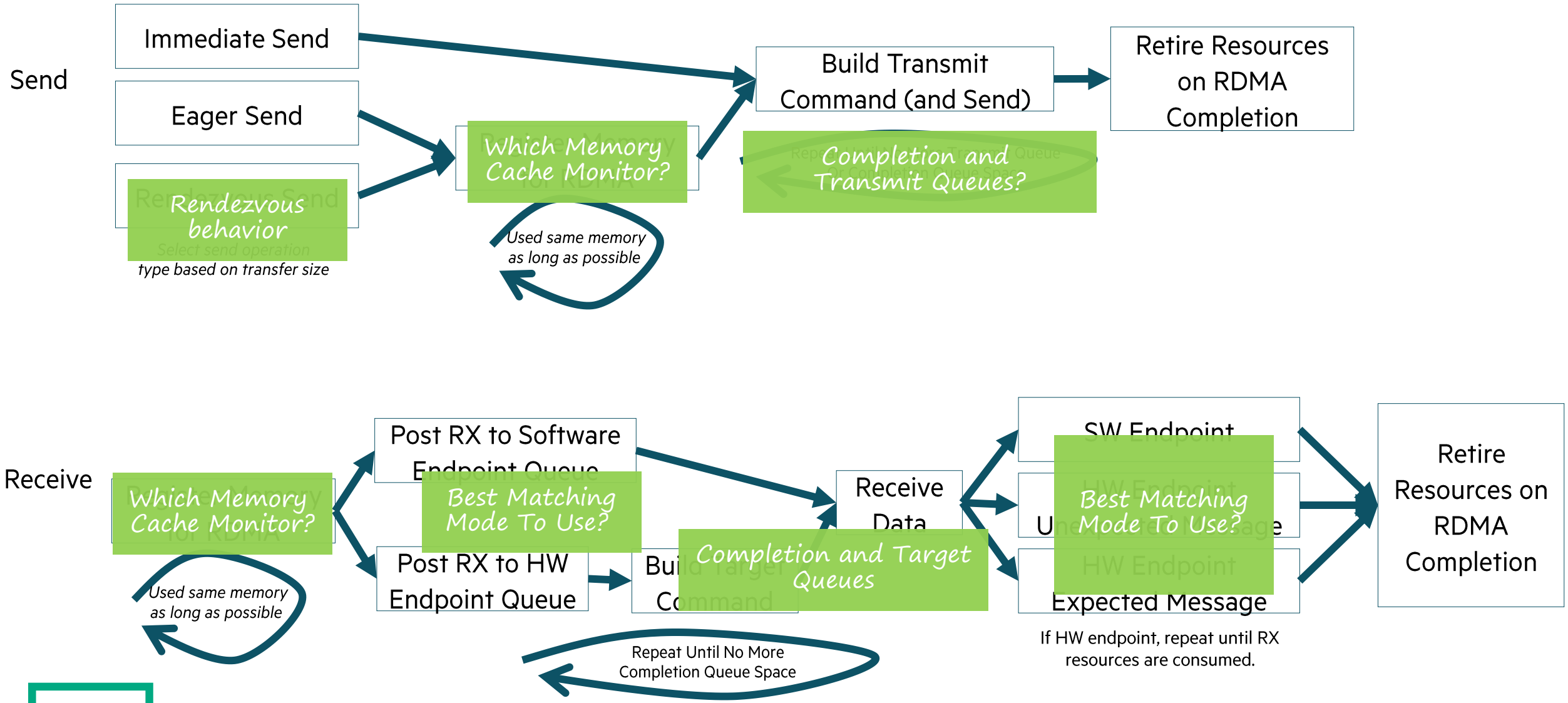
Why Users/Sites Might Set Libfabric Environmental Variables?

- Application specific requirements
  - Application specific nuances ∩ avoiding interconnect-specifics in codes
  - Usually to fix failure, but sometimes tune performance including “hard-core” sophisticated users
- HW specific divergence
  - Varying cost impact of memory overhead (e.g. HBM = $$$)
  - Scale impact
- Different expectations *who* gives “hints” (Cray SW vs 3rd party stacks/applications)
- Workarounds
  - Software bugs, including undoing old workarounds to now-fixed bugs
  - Current Si limitations (fixed in future generations)
- To enable debug information

Conceptual Flow Of What's Going On Under the Covers



Conceptual Flow Of What's Going On Under the Covers



Memory Registration

Memory Registration Cache Monitor

- To use RDMA memory must be “registered”
- Usually done by Libfabric provider (but can be done by application)
  - Cached because each registration action takes time away from computing
- The “memory cache monitor” watches for re-use (or eviction)
  - Breadcrumbs like memory allocation calls and page faults - can have app dependencies
  - Monitor for GPUs memory - CUDA, ROCM, Xi - plus 3 options for CPU memory:

memhooks	userfaultfd (uffd)	kdreg2
Libfabric core	**Part of Linux distributions (kernel module)**	**HPE Developed – ships with HPE Slingshot Host Software and HPE Github as open source**
Track memory allocation/free	**Handle page faults in user space**	**Enable applications where memhooks & uffd don’t work and be a single monitor standard**
Today’s default (version 12.0.x and older)	**Required with NCCL and RCCL to avoid memhooks/CUDA deadlock**	**Optionally installed prior to release 12.0 SHS. Possible default in future releases.**
Memory registered by provider as it is passed by application, with intercepts to know if present in the cache	Lazy, on-demand registration when memory is accessed. Operates at the page level (fine-grained)	Uses the kernel to atomically notify user-space of invalidation memory registrations. Relies on kernel MMU notifier functionality.

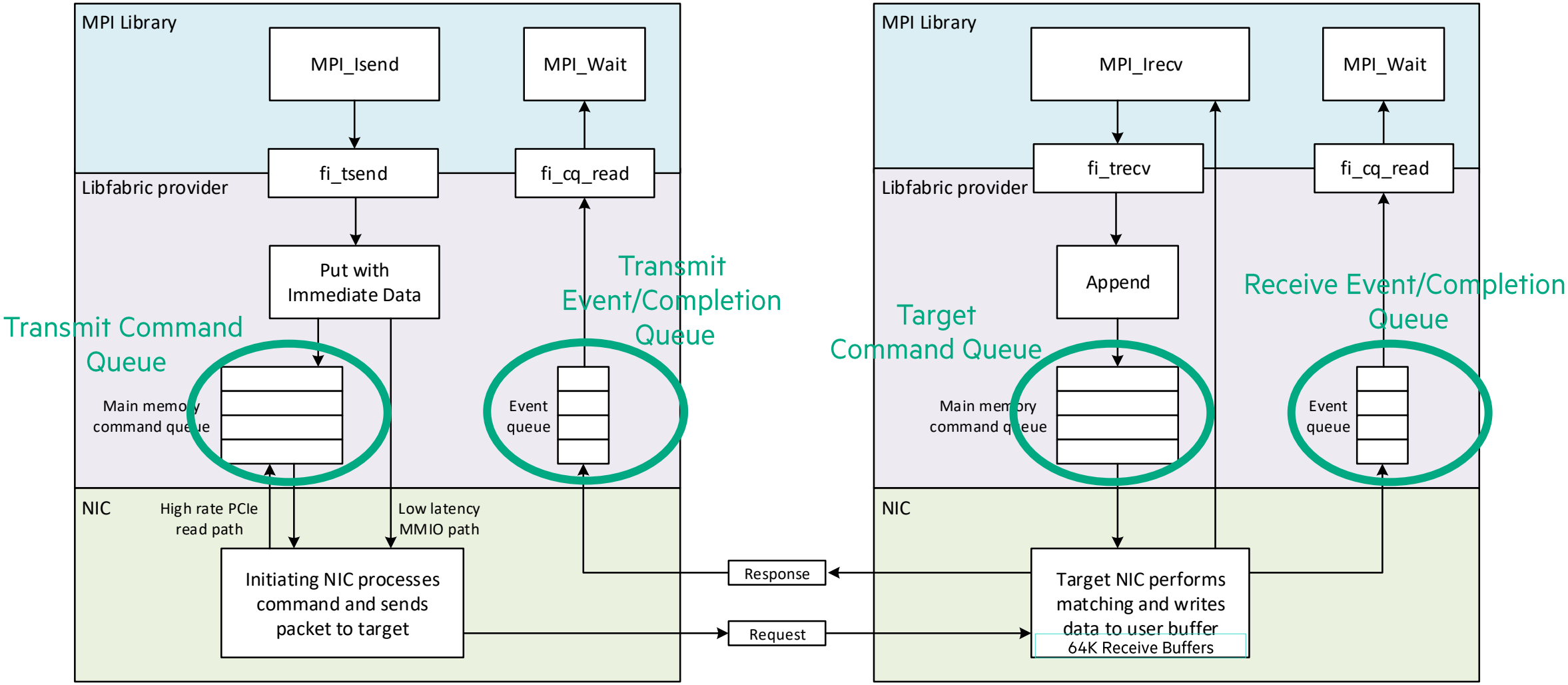
Memory Cache Monitor Environment Settings

Function	Environment Variable	Default Value	Comments	Symptom/Error (example)
System Mem Monitor	FI_MR_CACHE_MONITOR	memhooks	Options are uffd or kdreg2. NCCL and RCCL require uffd or kdreg2	“MR cache disabled for %s memory” "Failed (ret = %d) to monitor addr=%p len=%zu\n"
Cache Size	FI_MR_CACHE_MAX_SIZE	Sys memory÷cpu_cnt÷2 *	-1 means “unlimited” – (little downside to setting that)	Unexpected poor RDMA performance
Cache Count	FI_MR_CACHE_MAX_COUNT	1024 *	Setting to 0 disables caching: Useful as a debug tool. If set to unlimited, can use to govern memory consumption	Unexpected poor RDMA performance
NVIDIA CUDA Mem Monitor	FI_MR_CUDA_CACHE_MONITOR_ENABLED	1		"Failed to get CUDA buffer ID for buffer %p len %lu\n“ "cuPointerGetAttribute() failed: %s:%s\n"
AMD ROCR Mem Monitor	FI_MR_ROCR_CACHE_MONITOR_ENABLED	1		"Failed to perform hsa_amd_pointer_info: %s\n“ "Cannot monitor non-HSA allocated memory\n“ "Failed to perform hsa_amd_register_deallocation_callback: %s\n"
Intel Xi Mem Monitor	FI_MR_ZE_CACHE_MONITOR_ENABLED	1		"Could not get memory id\n"
GPU Specific	FI_CXI_DISABLE_HOST_REGISTER	0	Disables registration of overflow buffers with GPUs to overcome excessive memory allocation on GPUs with many processes	"cudaErrorMemoryAllocation:out of memory"

* Set differently by Cray MPI

Queue Sizes

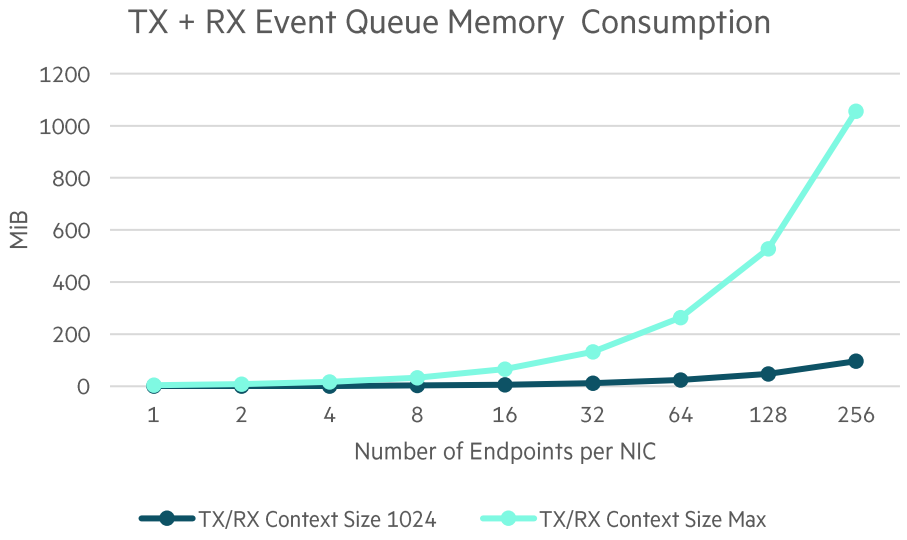
Illustration Of MPI Send Execution

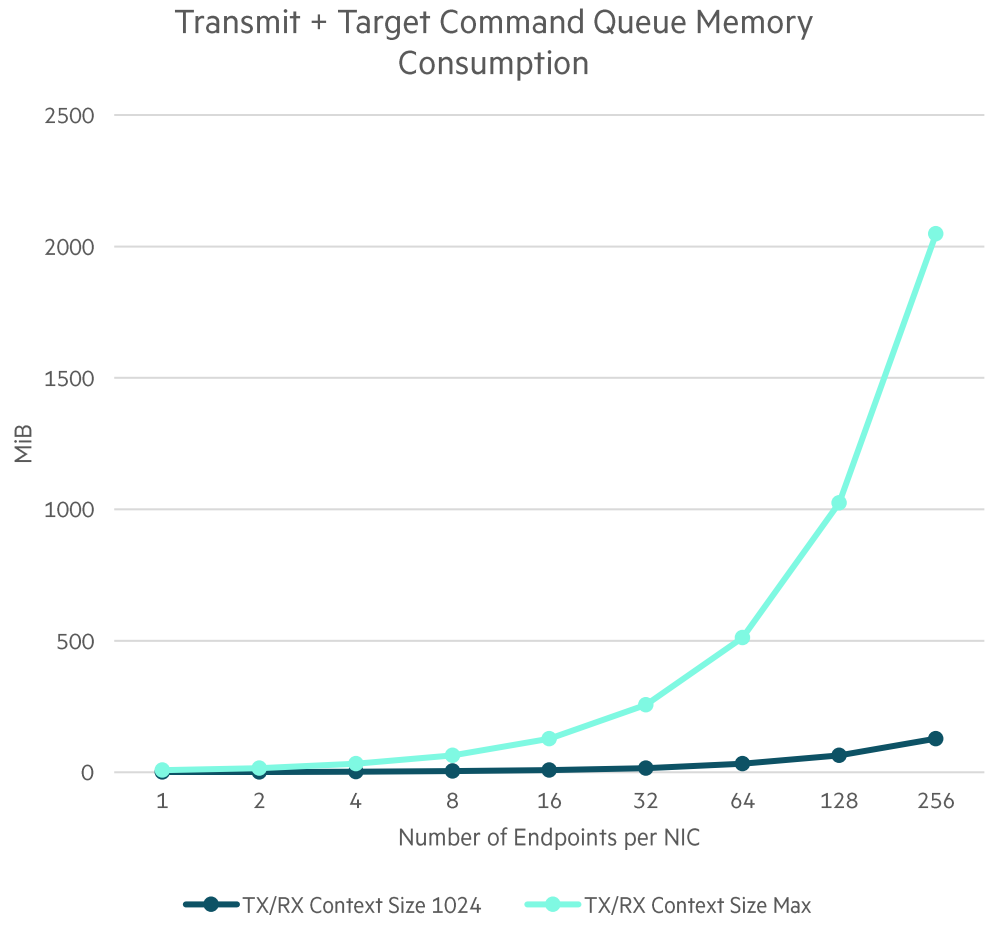


Four Queues Configured Per “Endpoint”

These Queues Consume Host Memory – Ideally Want To Balance With Number of Messages In Flights

Queue Size Memory Illustration





Can consume substantial memory – that's why this is not today set globally high by default

Command and Event/Completion Queues

- Event / Completion Queues (RX and TX)
  - Reports asynchronous RDMA completion events
  - Impacts permitted credits to control inflight operations
  - Low TX event queue space will result in CXI provider returning –FI_EAGAIN (performance impact)
- Command Queues (TX and Target)
  - Stores commands to queue to the NIC
    - e.g. transmit command queue used to issue RDMA operations (e.g., fi_send)
    - Target command queue posts receives to the NIC
  - Full command queue results in –FI_EAGAIN
    - Could be undersized TX Queue - or network backpressure

Cued by default RX and TX size environment variables if not specified by the application

Note prior to release 11.0 the Completion Queue Size variable was used to set a shared queue for RX and TX

Event Queue and Command Queue Related Environment Variables

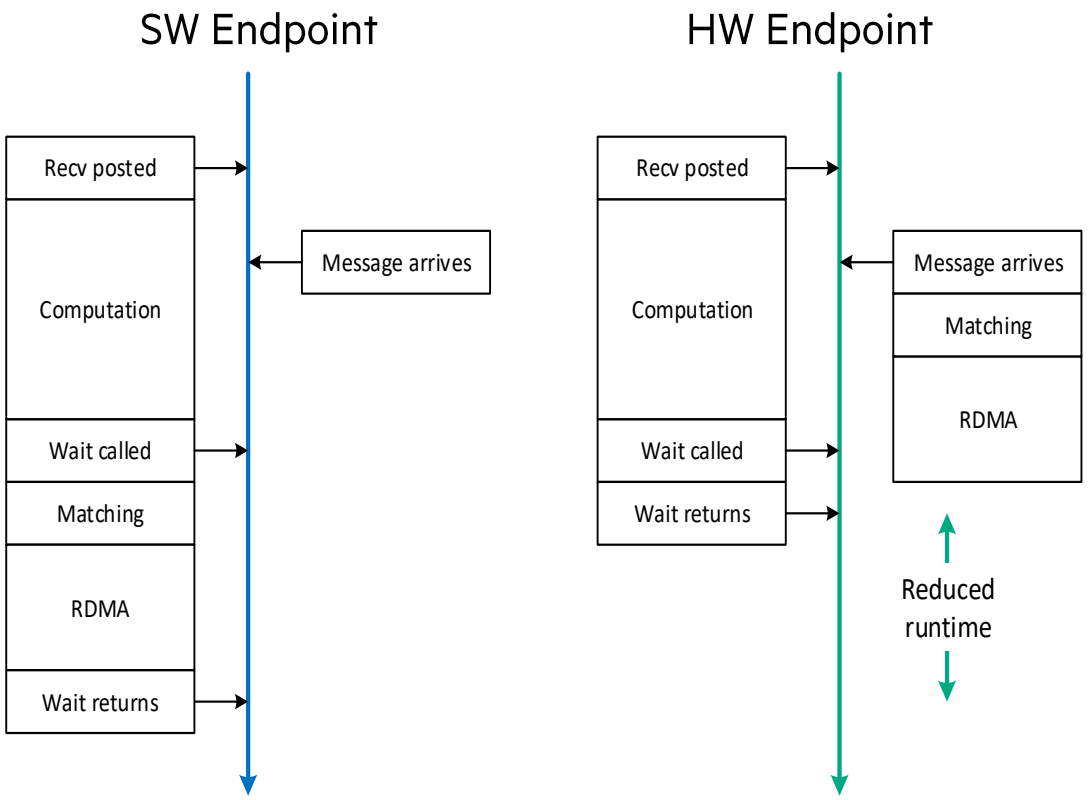
What	Variable	Default	Max	Comments/Symptom/Error (example)
Transmit Context	FI_CXI_DEFAULT_TX_SIZE	512 *	16384	Used to set the transmit command queue and completion queue. Applications which rely on significant unexpected rendezvous messaging may require this to be increased to the size of the expected outstanding rendezvous operations FI_EAGAIN being returned may mean this value needs to be increased.
Receive Context	FI_CXI_DEFAULT_RX_SIZE	512	15360	Used to set the receive event completion queue and target command queue. `cxi:core:cxip_recv_pte_cb():2432<warn> Flow control EQ full`
Completion Queue	FI_CXI_DEFAULT_CQ_SIZE	1024 *	---	In older releases (prior to 11.0) this was used to size the transmit and receive command queue structure. It can still be used as the internal calculation will use a formula that looks at the maximum of the TX size calculation and the CQ divided by two. But sites may wish to evaluate whether this is now setting structure too large and consuming unneeded memory. The following is an example error which may requiring increasing this value. `libfabric:88194:cxi:core:cxip_cq_eq_progress():544<warn> Cassini Event Queue overflow detected.`

- Set differently by Cray MPI

Match Modes

HPE Slingshot NIC Can Offloads Expected & Unexpected Msg Tag Matching

The Libfabric Provider Provides Three Match Modes Offered to Help Manage HW Offload Uses



Mode	Use
Hardware Endpoint	Take advantage of Cassini offloads. Allocates an overflow buffer for unexpected messages.
Software Endpoint	Avoid exhausting HW resources or want to target HW offload specifically. Allocates a request buffer. .
Hybrid Endpoint	If don't know traffic patterns. Switches from HW to SW if resources exhausted. Allocate both a request and overflow buffer.

Match Mode - Summary

Function	Environment Variable	Default	Max Value	Comments/Symptom/Error
Match Mode	FI_CXI_RX_MATCH_MODE	HARDWARE		Options are HARDWARE SOFTWARE and HYBRID
Overflow Buffer	FI_CXI_OFLOW_BUF_SIZE	2097152 (2MB) *		Overflow buffers should be sized based on unexpected message rate and eager data sizes, and scale. E.G. increasing FI_CXI_RDZV_THRESH or FI_CXI_RDZV_EAGER_SIZE would suggest increasing FI_CXI_OFLOW_BUF_SIZE. Request buffers should be sized on message rates and eager data sizes, and scale – e.g. increasing (Multiple overflow buffers are created to prevent race conditions)
Request Buffer	*FI_CXI_REQ_BUF_SIZE*	2097152 (2MB)		(Multiple buffers are created to prevent race conditions)
Hybrid Subconfiguration	FI_CXI_HYBRID_PREEMPTIVE FI_CXI_HYBRID_RECV_PREEMPTIVE FI_CXI_HYBRID_POSTED_RECV_PREEMPTIVE, FI_CXI_HYBRID_UNEXPECTED_MSG_PREEMPTIVE			For advanced users or debugging – not typical
Overflow Subconfiguration	FI_CXI_OFLOW_BUF_MIN_POSTED FI_CXI_OFLOW_BUF_COUNT FI_CXI_OFLOW_BUF_MAX_CACHED :			
Request Subconfiguration	*FI_CXI_REQ_BUF_MIN_POSTED* *FI_CXI_REQ_BUF_MAX_CACHED* *FI_CXI_REQ_BUF_MAX_COUNT*			

* Set differently by Cray MPI

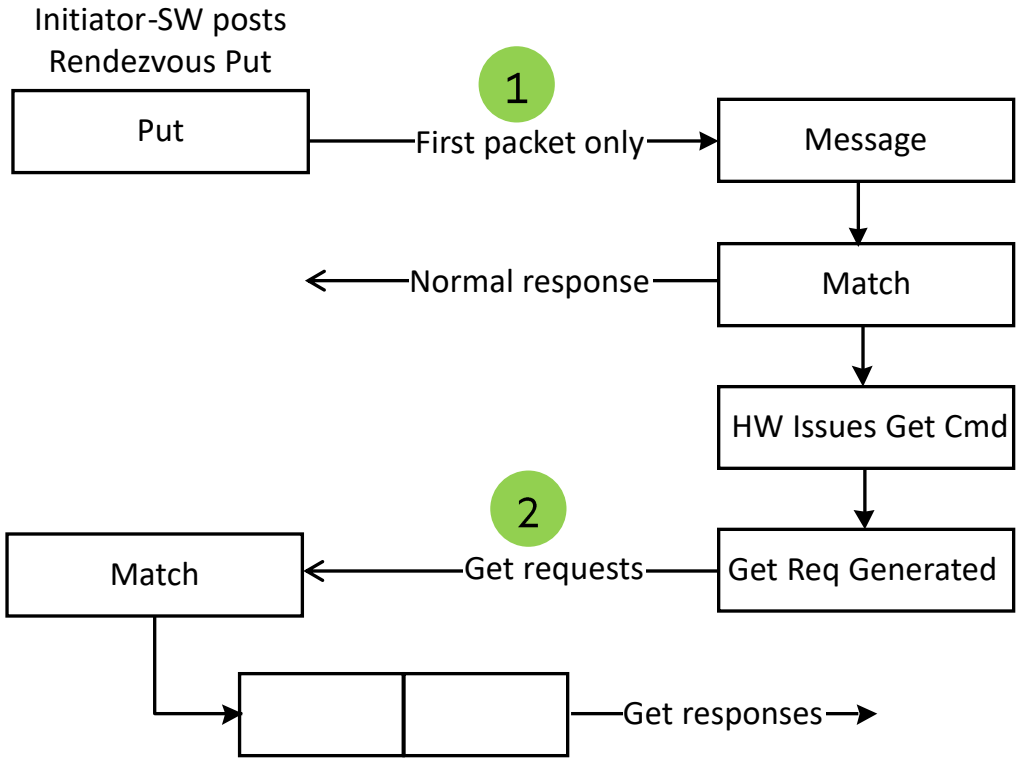
Rendezvous Protocol

Rendezvous offload

- Highly efficient especially for larger message sizes
  - Typical for all GPU applications, many CPU applications
  - Enables communications and computation overlap
- HPE Slingshot NICs is highly optimized for this
  - Rendezvous protocol offloading
  - Tag matching offloading
  - Unexpected message handling offloading
  - “Bulk payload” data adaptively routed packet-by-packet (no need for expensive DPU to order packets)

Two “stages” to rendezvous transaction

1. Set up the with a control message + portion of the message (“eager data”)
2. Then receiver pulls data as a “read” operation from the sender and matches/orders the data



Rendezvous Variables

- Note that it is also critically important to have set previously discussed buffer and match modes
- Below are the most important settings

Function	Environment Variable	Default	Max	Comments/Symptom/Error
Threshold	FI_CXI_RDZV_THRESHOLD	2048 *		Threshold to use rendezvous protocol. Can increase if it is more efficient to use a put operation for all the data
Eager size	FI_CXI_RDZV_EAGER_SIZE	2048		Eager data size for rendezvous protocol. Not typical to change as it can incure the unexpected message overhead.
HW Initiated Gets or Alternative	FI_CXI_RDZV_PROTO	0 (or default)		alt_read is the alternative setting that uses SW get invocation. Recommended for NCCL/RCCL applications. The symptom that implies using alt-read is glacially slow progress on a rendezvous exchange.
Additional Buffer Size for Software Gets (Alternative Rendezvous)	FI_CXI_SW_RX_TX_INIT_MAX	1024	16384	The maximum number of TX operations that can be initiated internally by the provider as part of RX processing. For instance, for rendezvous messaging this controls the number of software initiated RGet operations that can be outstanding when running as a software EP or as a hardware EP with the “alt_read” rendezvous protocol enabled. Symptom may need to be increased is rendezvous performance throttles as application scales in – SW mode or alt-read.

* Set differently by Cray MPI

Alt-Read Rendezvous Settings

- We have seen Cassini NIC HW offload unable to handle full rendezvous GET offload in some patterns
  - Manifests as very very slow progress that looks like a lockup due to retry handler activity
  - Initially seen with NCCL/RCCL – rare beyond that
- An alternative Rendezvous protocol was implemented that does not use HW initiator offload portion of "GET"
  - Performance impact minimal for large message - extra operations fall to the CPU (not GPU)
  - Expect more impactful on CPU-based applications - so do not recommend making global default
- In addition to configuration (below) may need to increase FI_CXI_SW_RX_TX_INIT_MAX to add buffer space for SW GET

Configuration	How	Pros	Cons
Software Only Endpoint	*FI_CXI_RDZV_PROTO*= alt_read *FI_CXI_RX_MATCH_MODE=SOFTWARE*	Completely in user control for one app	May leave some performance on the table by forcing CPU (not GPU) to do matching
HW Offload Enabled Endpoint	*FI_CXI_RDZV_PROTO*= alt_read FI_CXI_RX_MATCH_MODE=HARDWARE (or leave *as default as of 12.0) or F*I_CXI_RX_MATCH_MODE=H*YBRID* Set sysfs variable "rdzv_get_en" in kernel space by privileged user (such as SLURM or PALs) – refer to docs	HW matching offload may help performance	If NIC shared with other application all will be forced to use alt_read

Common Scenarios

Common Scenarios

- NCCL & RCCL
  - Follow the application node guidance
  - Requires alternative rendezvous and not memhooks
  - Settings to force GDRCopy (and not cudamemcpy) to move between buffers and GPU memory (relevant to older releases – new options with SHS 12.0+)
- Large scale jobs + 3rd party MPI
  - May require beefing up queue sizes
- Less Common
  - Specific applications

Environmental Variable	Default Setting (12.0/earlier)	Cray MPI Setting (25.3/earlier)
FI_CXI_RDZV_THRESHOLD	2048.	16384
FI_CXI_RDZV_EAGER_SIZE	2048	(same)
FI_CXI_DEFAULT_CQ_SIZE	1024	131072
FI_CXI_DEFAULT_TX_SIZE	512	1024
FI_CXI_OFLOW_BUF_SIZE	2097152	12582912
FI_CXI_OFLOW_BUF_COUNT	3	(same)
FI_CXI_RX_MATCH_MODE	hardware	(same)
FI_CXI_REQ_BUF_MIN_POSTED	4	6
FI_CXI_REQ_BUF_SIZE	2097152	12582912
(Memory cache monitor)	memhooks	(same)
FI_MR_CACHE_MAX_SIZE	sys memory ÷ cpu_cnt ÷ 2	-1
FI_MR_CACHE_MAX_COUNT	1024	524288

Debugging

- Always capture a libfabric log with FI_LOG_LEVEL=warn FI_LOG_PROV=cxi
- To see environment that is overridden or what HMEM types have been enabled use FI_LOG_LEVEL=info
- If debug is desired, build opensource with –debug-enabled and use FI_LOG_LEVEL=debug
  - This can show conditions that are not errors, operations that are posted, and memory regions that have been allocated. All of which can be used to verify if there is an application usage or provider error.
  - Superchatty and can hurt performance – lose compiler optimizations

Thank you

jesse.treger@hpe.com