STATUS OF OPEN FABRICS OVER VERBS BASED FABRICS

Sayantan Sur, Intel

Presenting work done by Arun Ilango, Dmitry Gladkov, Dmitry Durnov and Sean Hefty and others in the OFIWG community

6th Annual MVAPICH User Group (MUG) 2018

Legal Disclaimer & Optimization Notice

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, 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. For more complete information visit www.intel.com/benchmarks.

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Open Fabric Interfaces

User-centric interfaces lead to innovation and adoption

Open Source

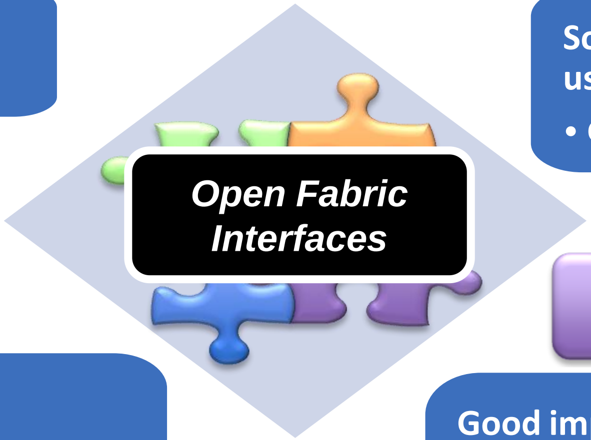
Inclusive development effort

- App and HW developers

User-Centric

Software interfaces aligned with user requirements

- Careful requirement analysis



Scalable

Optimized SW path to HW

- Minimize cache and memory footprint
- Reduce instruction count
- Minimize memory accesses

Implementation Agnostic

Good impedance match with multiple fabric hardware

- InfiniBand, iWarp, RoCE, raw Ethernet, UDP offload, Omni-Path, GNI, BGQ, …
- Works on Linux, Windows and MacOS

OFI – State of the Union

OFI Insulates applications from wide diversity of fabrics underneath

Intel® MPI Library

MPICH*

Charm++*

Open MPI*

GASNet*

Sandia SHMEM*

NetIO*

Intel® MLSL#

libfabric Enabled Middleware

OFI

Advanced application oriented semantics

Tag Matching

Scalable memory registration

Triggered Operations

Remote Completion Semantics

Multi-Receive buffers

Shared Address Vectors

Unexpected Message Buffering

Streaming Endpoints

Reliable Datagram Endpoints

Sockets TCP, UDP

Verbs

Cisco usNIC*

Intel® OPA PSM

Cray GNI*

Mellanox*

IBM Blue Gene*

Exciting new providers in development!

Exploration

OFI Implementation Update

- ✓ *OFI Provider Infrastructure*
- ✓ *OFI API Exploration*
- ✓ *Companion APIs (Bonus!)*

1.5 API Updates
- RxM provider
- SOCK endpoint types
- Memory registration
- API optimizations

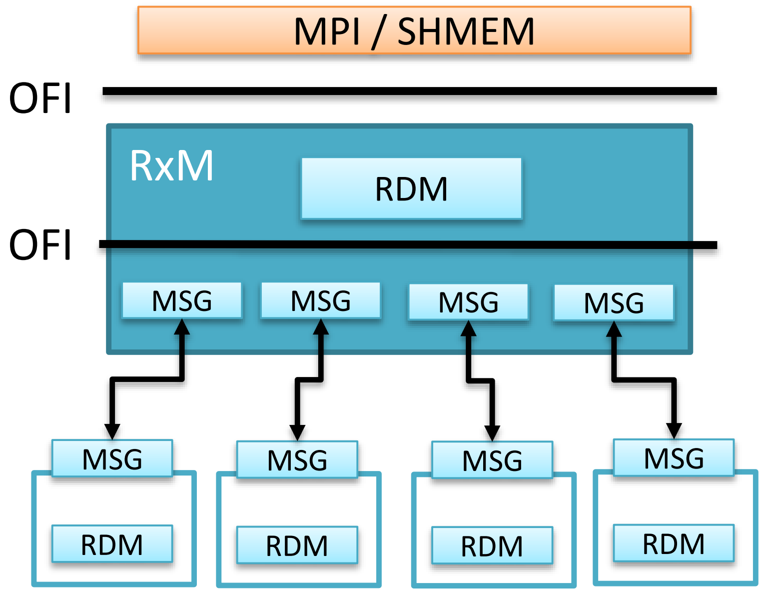
2017	v1.4.0..	..1.4.2	v1.5.0..	..1.5.3
2018	v1.6.0..	*v1.6.1*	*v1.6.2*	*v1.7.0*

1.6 Provider Enhancements
- PSM2 – native
- RxM performance
- SHM – shared memory support
- Persistent memory

1.7 Predictions
- New providers
  - RxD, multi-rail, new vendors
- SHM – xpmem support
- API enhancements

Provider Infrastructure Updates

RXM – Reliable Datagrams over Connections



Connection multiplexing

Primary path for HPC apps accessing verbs hardware

Verbs

NetworkDirect

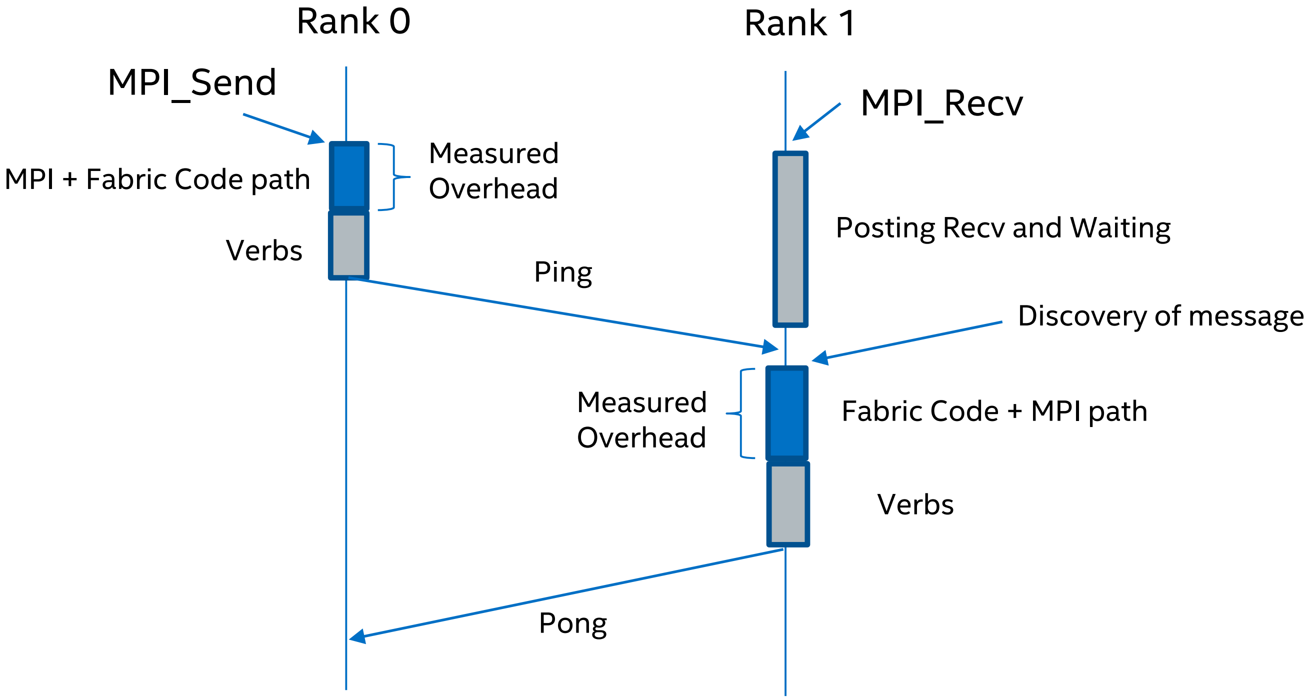
TCP

TCP will replace sockets

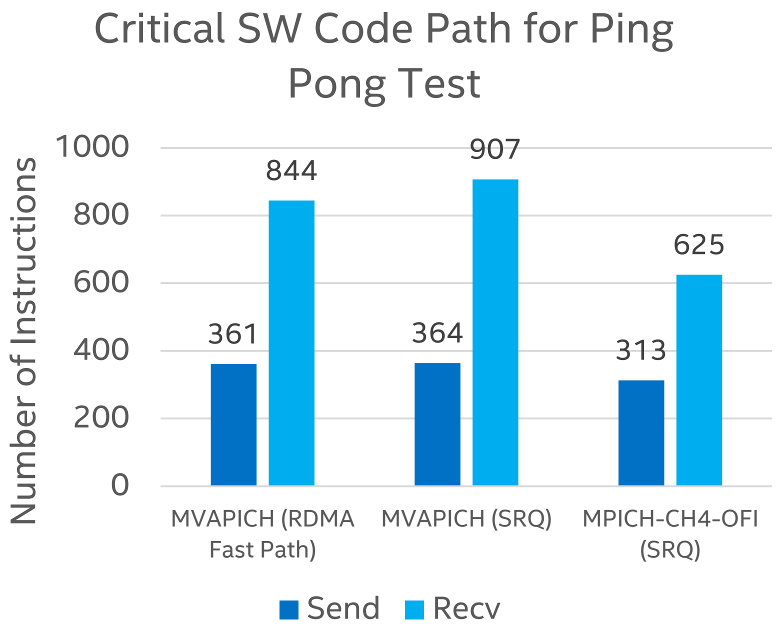
Optimizes for hardware features

- **Strong MPI performance**
- **Evaluating tighter provider coupling**

MPI Critical Path Software overhead Analysis



MPI Critical path software overhead



MVAPICH 2.3 (default configuration)
GCC 4.8.5, OFI master (@585919d)
-O3, -DNDEBUG

Gains in total code path primarily coming from combination MPICH-CH4 and OFI RXM provider

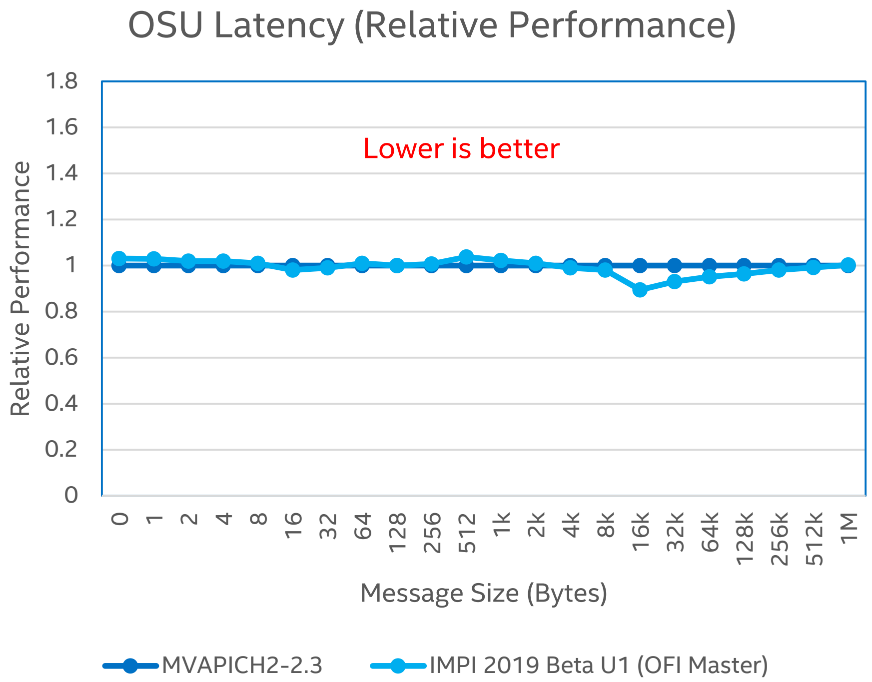
Instruction counts are an indirect measure help us gauge semantic fit

Ongoing optimization

- Aiming to reduce send path to about 250 instructions, and receive path to 450-480 instructions

Similar optimizations are possible in MVAPICH

MPI Performance Analysis - Latency



Platform:

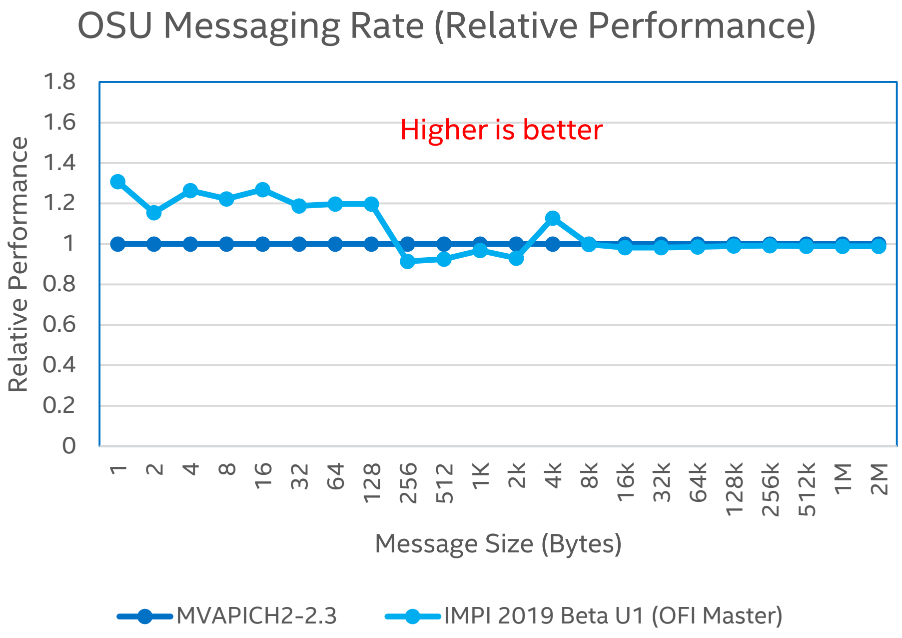
Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
Infiniband controller: Mellanox Technologies MT27700 Family [ConnectX-4]
intel_pstate on/turbo on
RHEL 7.4
mlnx1-OFED.4.3.0.2.1.43101.x86_64

Run details:

$ mpirun -hosts nnlmpibdw01,nnlmpibdw02 -n 2 -ppn 1 numactl --physcpubind=7 osu_latency -i 40000

$ FI_OFI_RXM_SAR_LIMIT=8192 FI_VERBS_MR_CACHE_ENABLE=1 mpirun -hosts nnlmpibdw01,nnlmpibdw02 -n 2 -ppn 1 numactl --physcpubind=7 osu_latency -i 40000

MPI Performance Analysis – Message Rate



Platform:

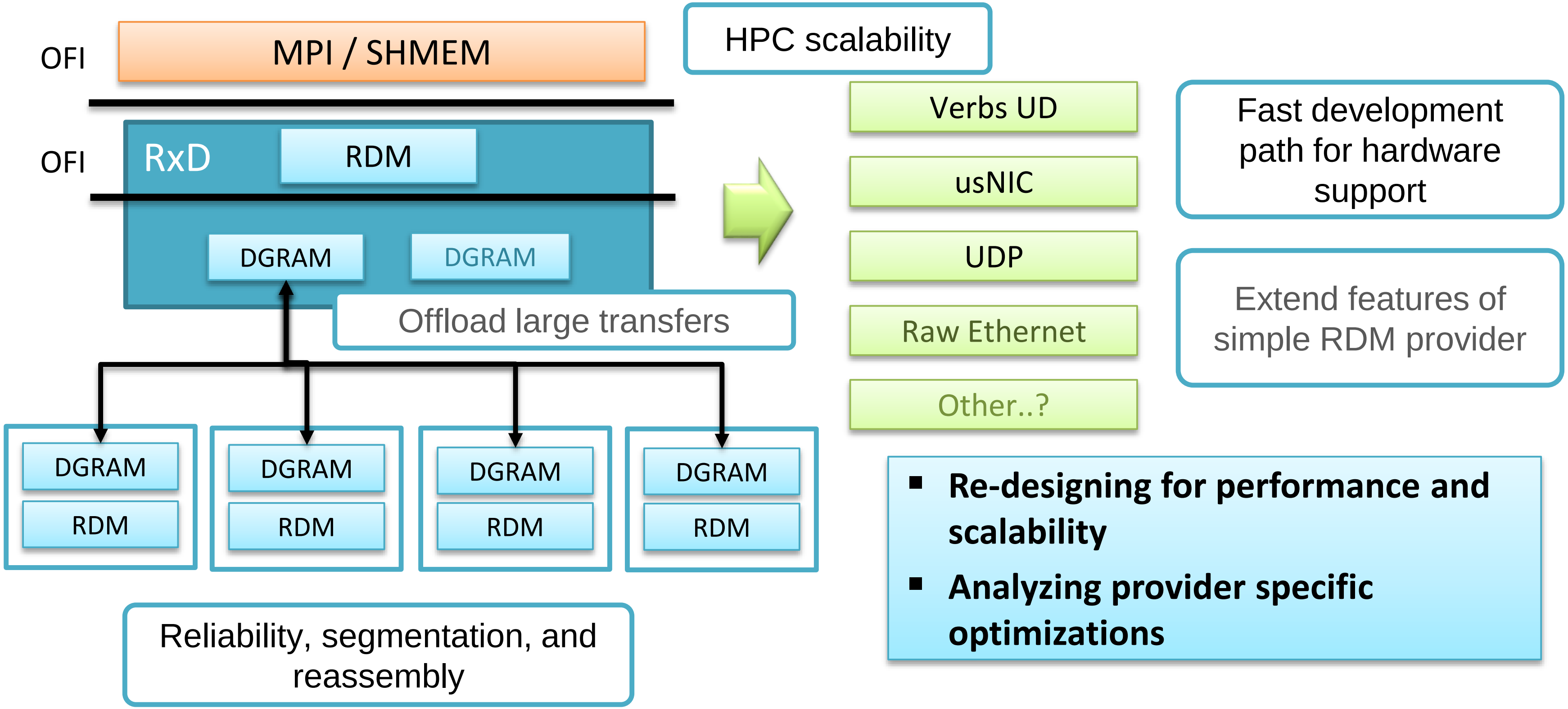
Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
Infiniband controller: Mellanox Technologies MT27700 Family [ConnectX-4]
intel_pstate on/turbo on
RHEL 7.4
mlnx1-OFED.4.3.0.2.1.43101.x86_64

Run details:

$ mpirun -hosts nnlmpibdw01,nnlmpibdw02 -n 2 -ppn 1 numactl --physcpubind=7 osu_mbw_mr

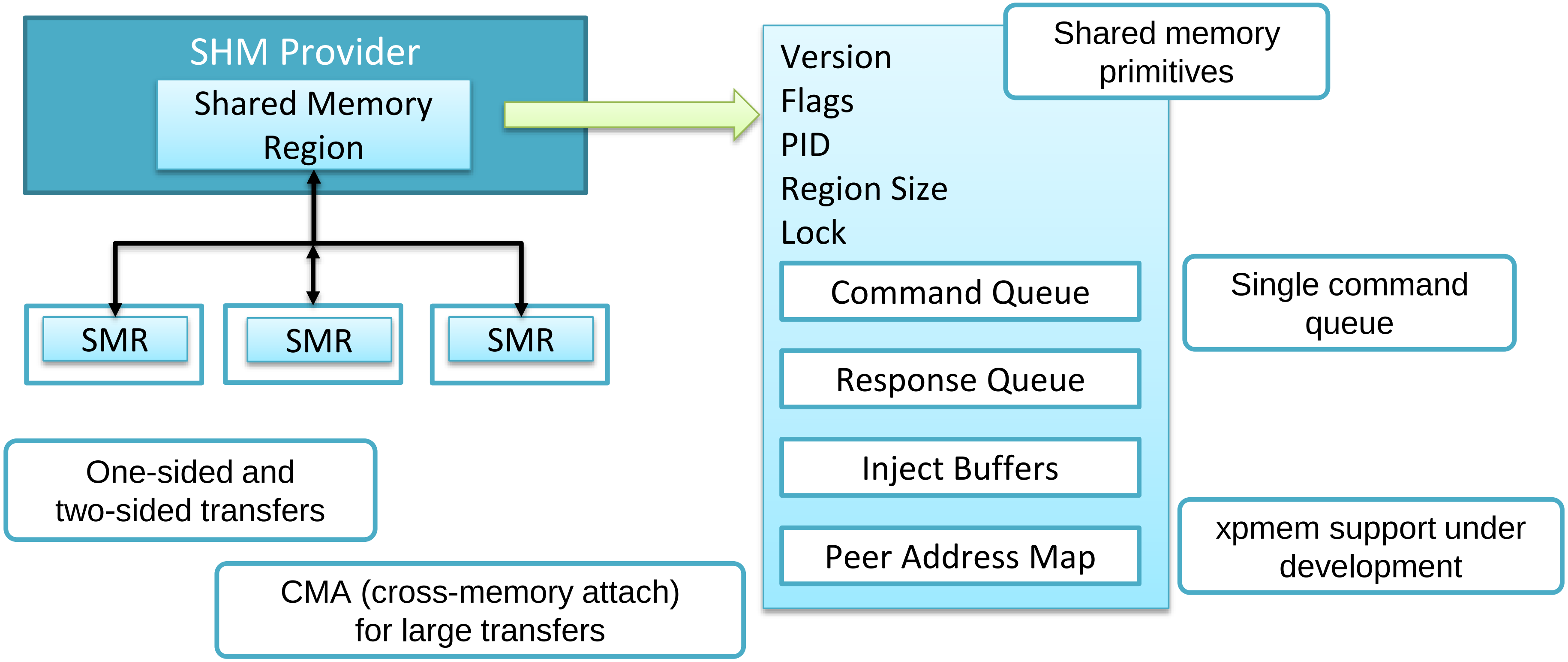
$ FI_OFI_RXM_SAR_LIMIT=8192 FI_VERBS_MR_CACHE_ENABLE=1 mpirun -hosts nnlmpibdw01,nnlmpibdw02 -n 2 -ppn 1 numactl --physcpubind=7 osu_mbw_mr

RXD – Reliable Datagram over Unreliable Datagram

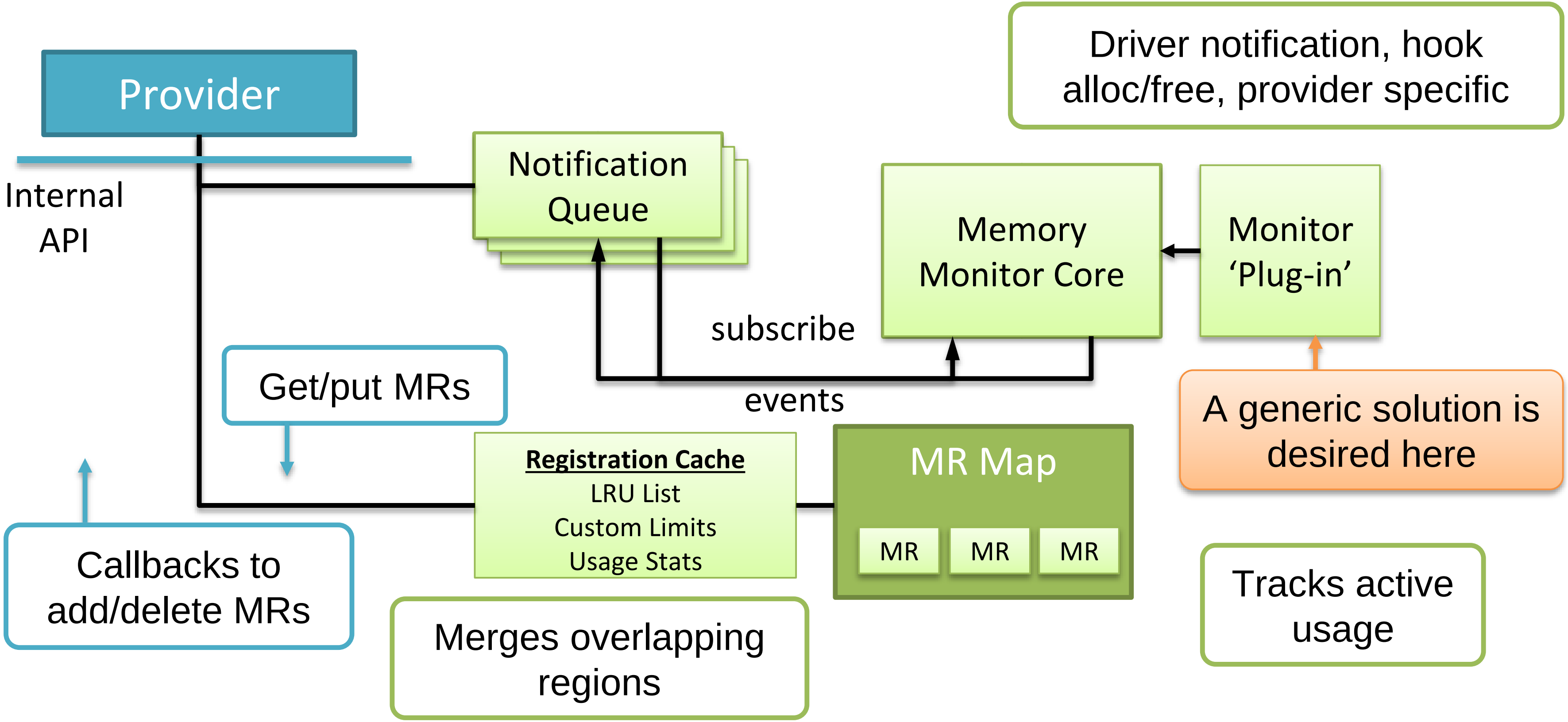


- **Re-designing for performance and scalability**
- **Analyzing provider specific optimizations**

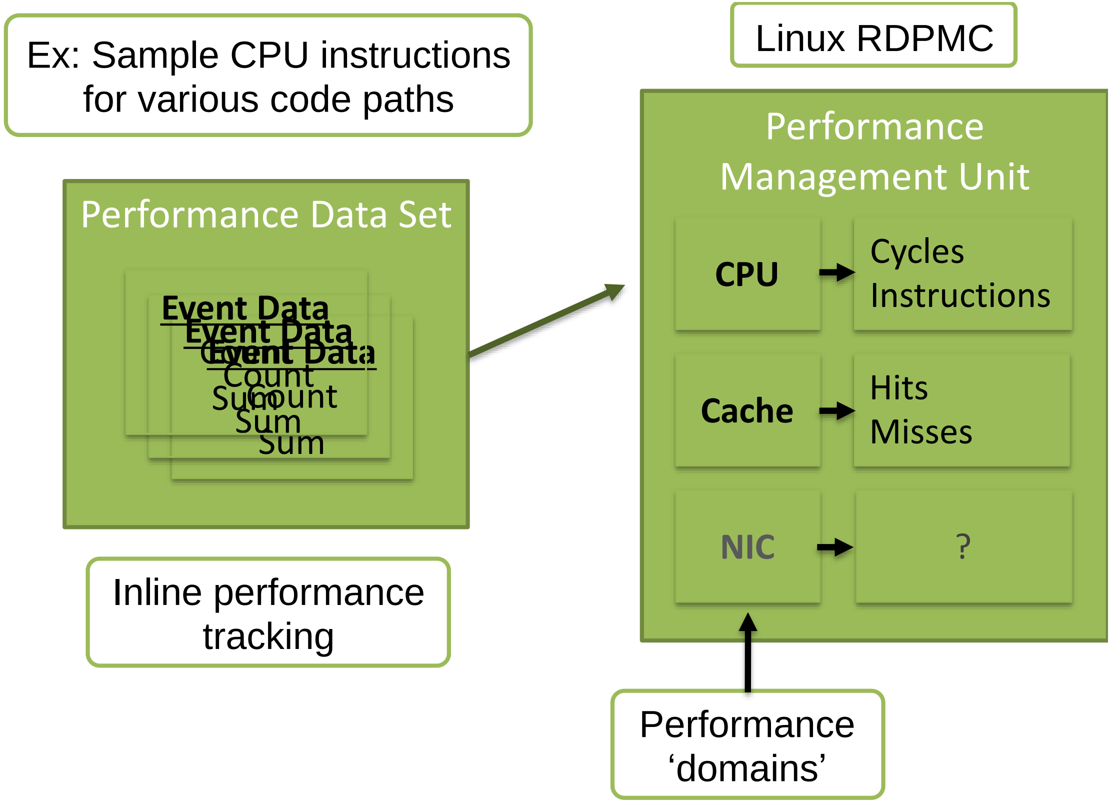
Shared Memory Provider



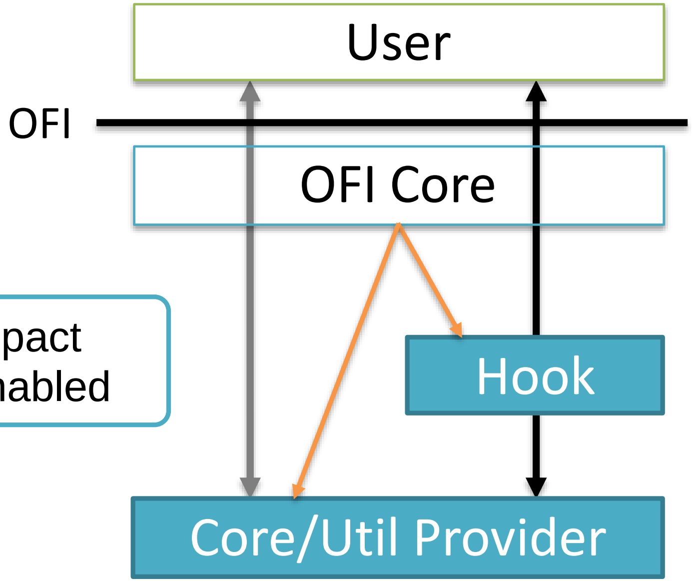
Memory Monitor and Registration Cache



Performance Monitoring



Hooking Provider



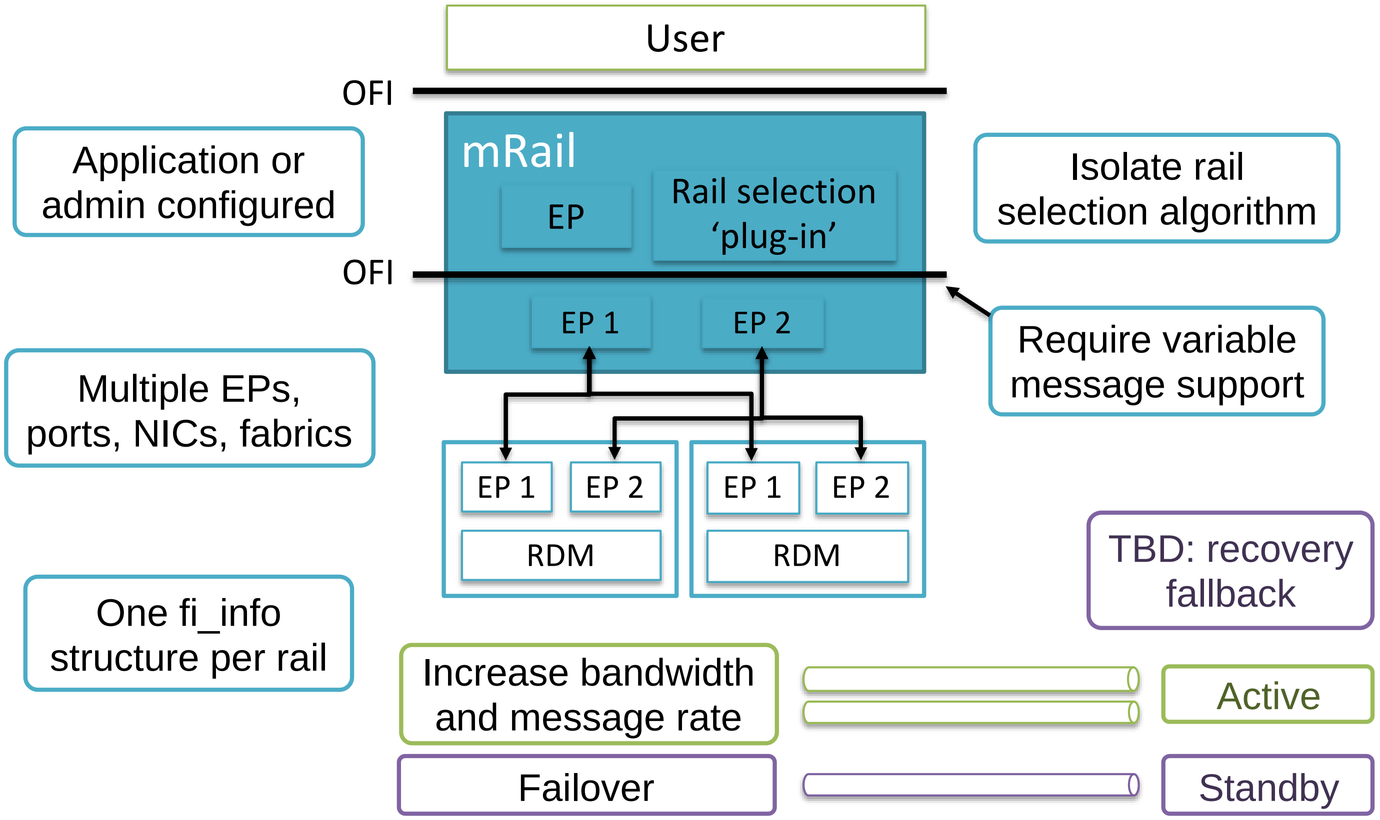
Always available – release and debug builds

Zero-impact unless enabled

Intercept calls to any provider

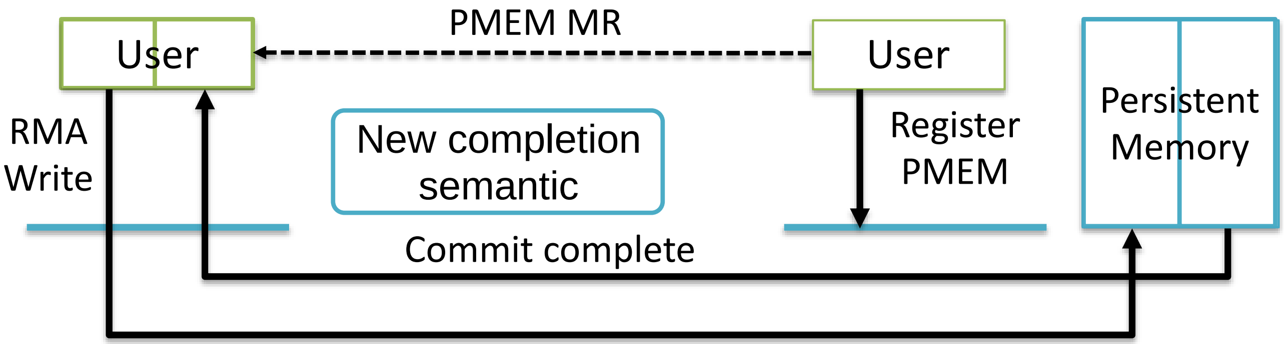
Debugging, performance analysis, feature enhancements, testing

Multi-rail provider



API Exploration

Persistent Memory



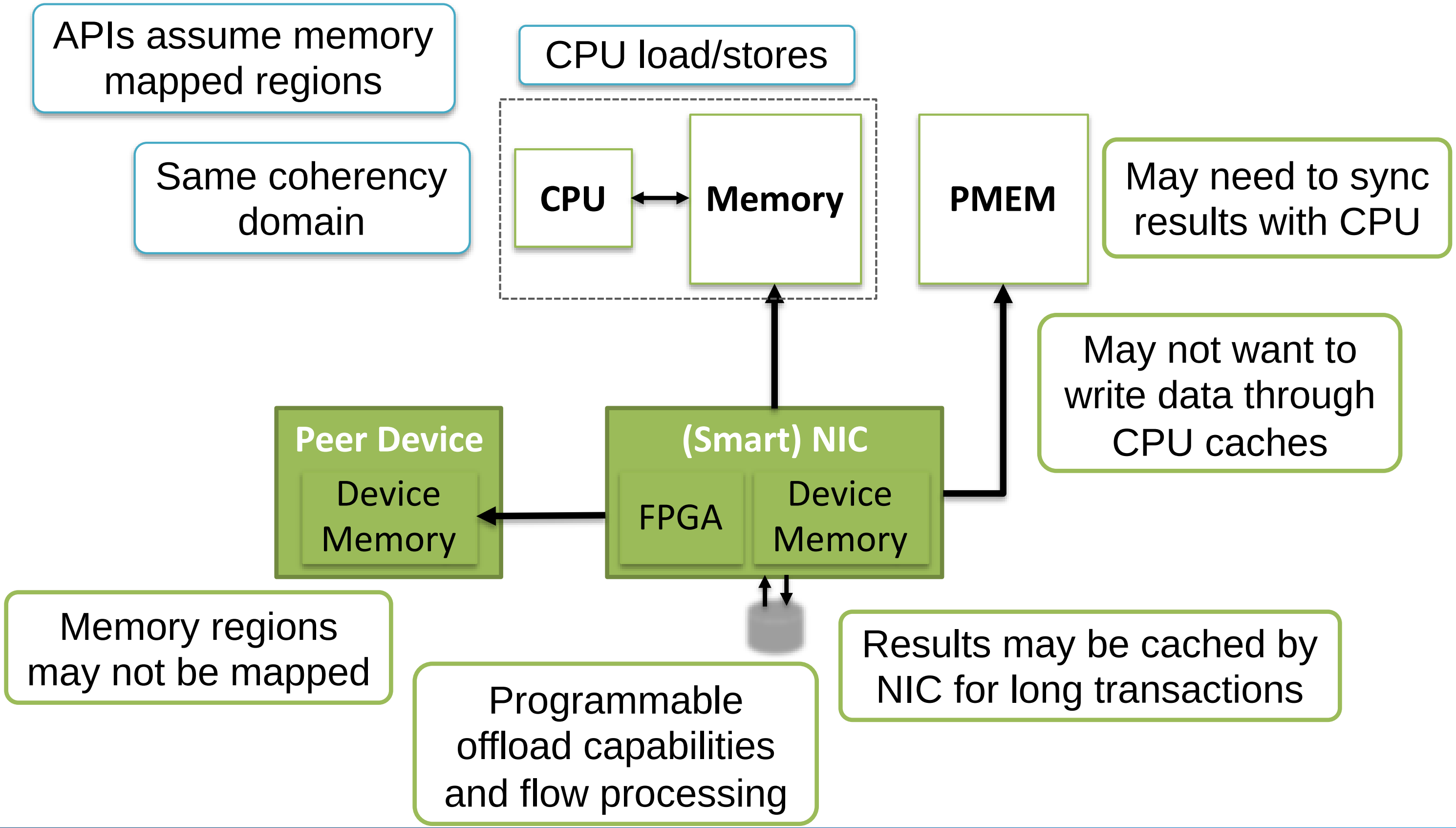
Work with SNIA (Storage Networking Industry Association)

Evolve APIs to support other usage models

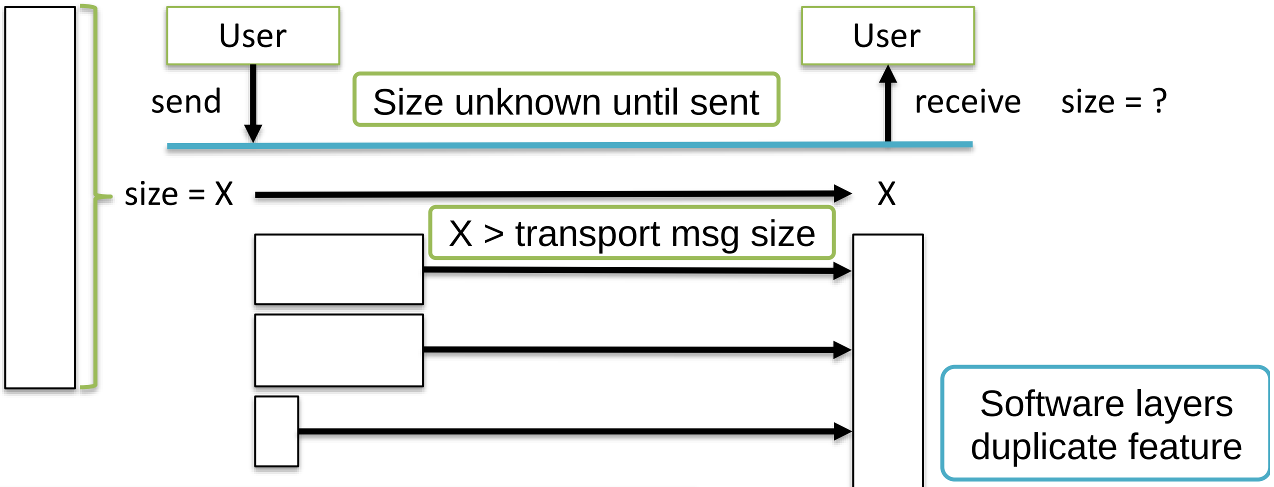
- **Exploration**
  - Byte addressable or object aware
  - Single or multi-transfer commit
  - Advanced operations (e.g. atomics)

- **Keep implementation agnostic**
  - Handle offload and on-load models
  - Support multi-rail
  - Minimize state footprint

Data Domains



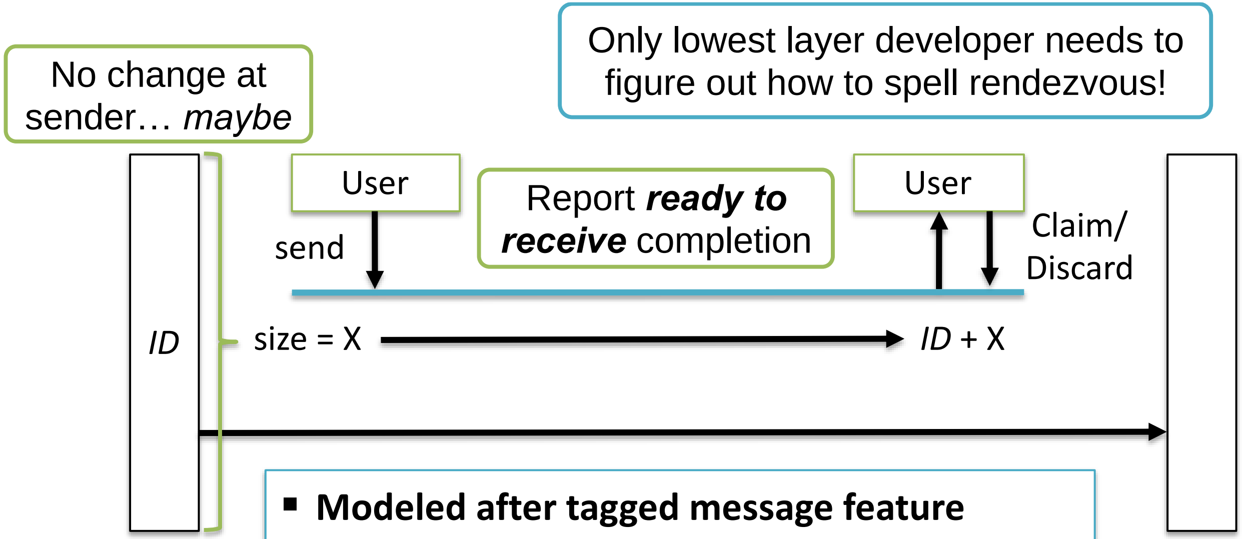
Variable Length Messages



- **Eager message→ ←rendezvous→**
  - RMA read or tagged message
- **MTU→ ←ack remaining transfer→**
  - RMA write, tagged send, send
- **RTS→ ←CLS transfer→**

Similar wire protocols – different implementations

Variable Length Messages (continued)



- **Modeled after tagged message feature**
- **Opt-in –impacts protocol**
- **Provider optimizes around hardware abilities**
- **Opportunity: report *discard* to sender**
  - Application flow control and load balancing
  - Dynamically disable receive processing (e.g. EBUSY)

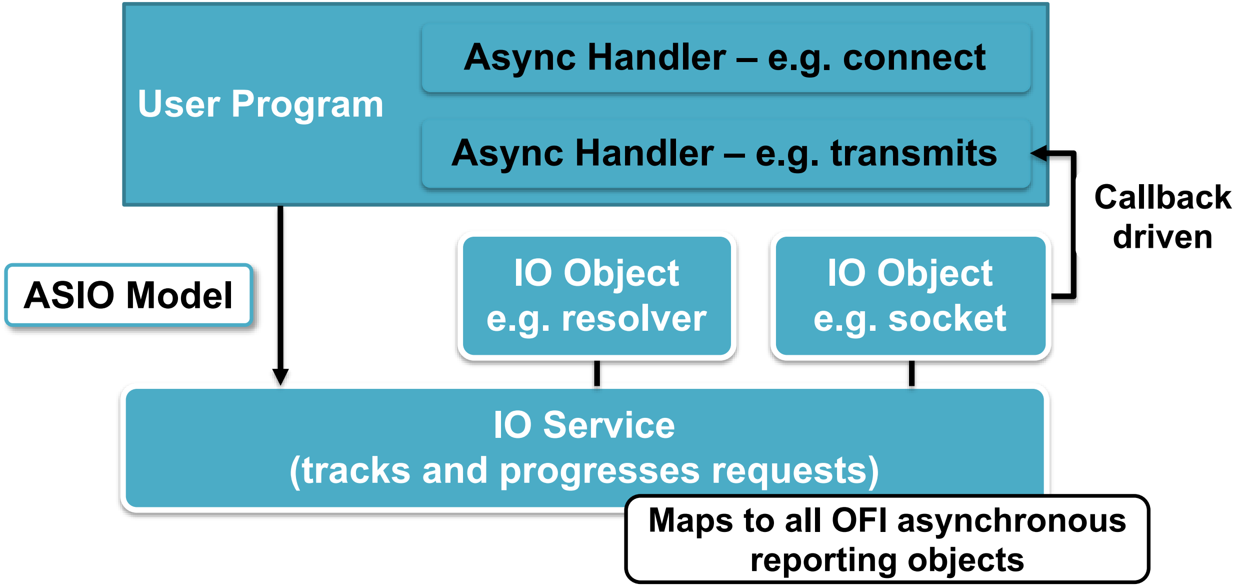
Companion APIs

C++ Standardization

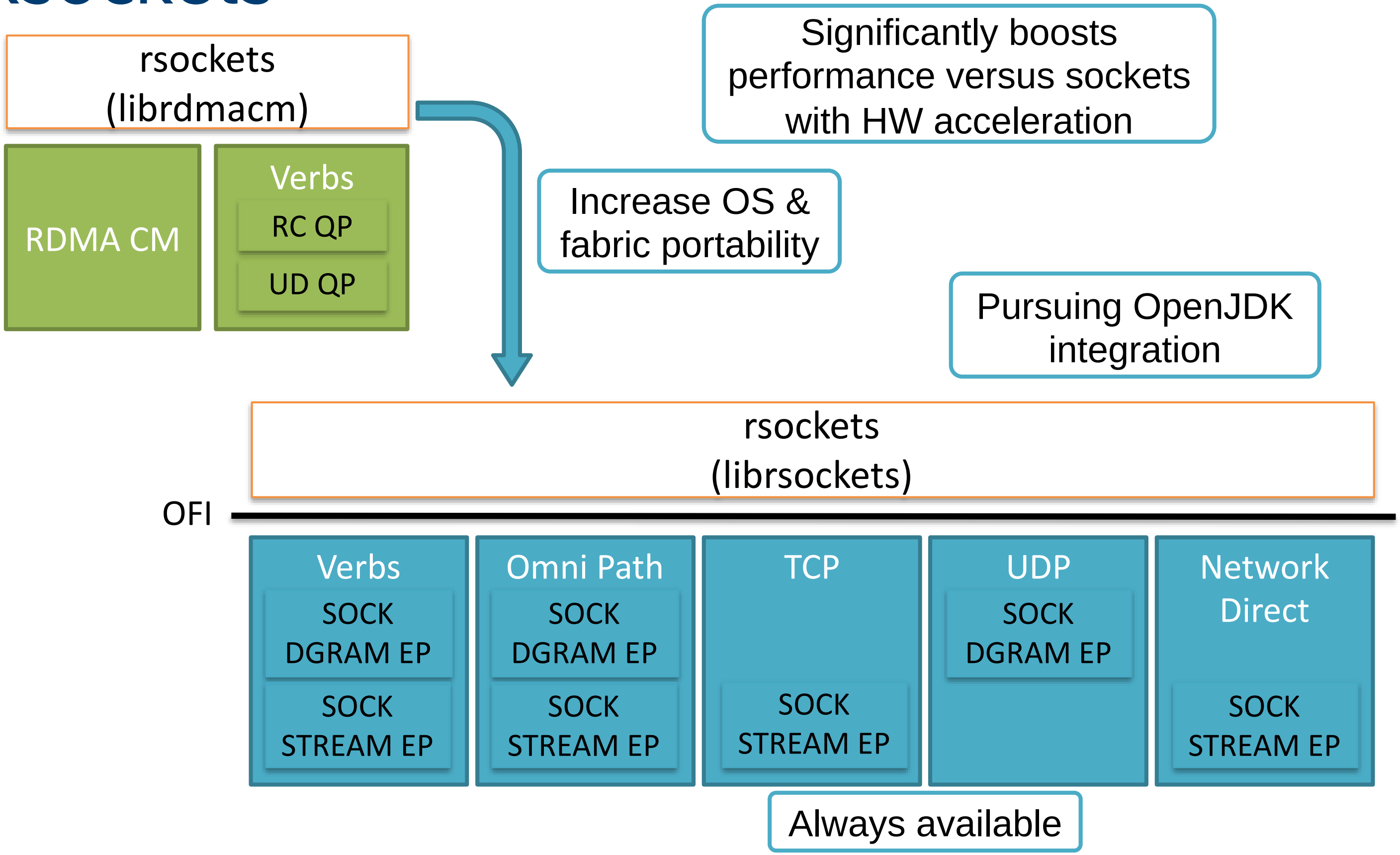
Add support for fabrics directly to the C++ language

Feedback from C++ community
- Implement proposal
- Detail alternatives
- Justify extensions

Proposal
- Extend ASIO
- Implement over libfabric



Rsockets


rsockets
(librdmacm)
RDMA CM
Verbs
RC QP
UD QP
Significantly boosts performance versus sockets with HW acceleration
Increase OS & fabric portability
Pursuing OpenJDK integration
rsockets
(librsockets)
OFI
Verbs
SOCK DGRAM EP
SOCK STREAM EP
Omni Path
SOCK DGRAM EP
SOCK STREAM EP
TCP
SOCK STREAM EP
UDP
SOCK DGRAM EP
Network Direct
SOCK STREAM EP
Always available

Summary

Significant software work ongoing to implement full set of OFI features on Fabric providers that lack native support

Components developed are generic and re-usable across Fabrics

Fabric vendors can implement subset of features and get access to wide OFI software ecosystem by leveraging utility components

As newer features are added to OFI, provide a pathway to quickly enable those features in older providers – applications can track latest OFI APIs

Participation in OFIWG is free, simple, no associations or boards to join

http://libfabric.org

experience
what's inside™