A Brief Introduction to OpenFabrics Interfaces’ libfabric

Dave Goodell, Paul Grun, Sean Hefty, Howard Pritchard, Bob Russell, Jeff Squyres, Sayantan Sur
OpenFabrics libibverbs middleware

• **Widely adopted low-level RDMA API**
• **Ships with upstream Linux**
• **Intended as unified API for RDMA**

• Designed around InfiniBand architecture
  – Targets specific hardware implementation

• Hardware, not network, abstraction
  – Too low level for most consumers
  – Interfaces not designed around HPC

• Hardware and fabric features are changing
  – Divergence is driving competing APIs
  – PSM, MXM, PAMI, uGNI …

• More applications require high-performance fabrics
  – Cloud systems, data analytics, virtualization, big data …

but…
Solution

OpenFfabsics Interfaces Working Group

Open Source

Leverage existing open source community
• Inclusive development effort
• App and HW developers

Application-Centric

Software interfaces aligned with application requirements
• 168 requirements from MPI, PGAS, SHMEM, DBMS, sockets, NVM, ...

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, others
Design

- Enable simple, basic usage
- Move functionality under libfabric
  - E.g. reliability over unreliable transports, SW atomic support

- Advanced application constructs
- Expose abstract HW capabilities
  - Data and message ordering, progress model, resource mgmt constraints, …

Range of usage models
Architecture

Note: current implementation focused on enabling applications

Libfabric Enabled Applications

- MPICH (Netmod)
- Open MPI (MTL / BTL)
- Clang UPC
- GASNet
- ES-API rsockets

Libfabric

- Control Services
  - Discovery
  - fi_info
- Communication Services
  - Connection Management
  - Address Vectors
- Completion Services
  - Event Queues
  - Counters
- Data Transfer Services
  - Message Queues
  - RMA
  - Tag Matching
  - Atomics

- Triggered Ops

Scalable addressing support (minimal memory footprint)

Lightweight completion mechanisms

Multiple data transfer semantics: point-to-point, one-sided, collective building blocks
Application Configured Interfaces

App specifies comm model

Communication type

Capabilities

Data transfer flags

Endpoint

Provider directs app to best SW path

Usage mode

NIC

Message Queue Ops

RMA Ops

RMA

sm. msg

inject send

send

write

read

lg. msg

App specifies comm model

Communication type

Capabilities

Data transfer flags

Endpoint

Provider directs app to best SW path

Usage mode

NIC

Message Queue Ops

RMA Ops

RMA

sm. msg

inject send

send

write

read

lg. msg
**API Performance Analysis**

Issues apply to many APIs: Verbs, AIO, DAPL, Portals, NetworkDirect, …

<table>
<thead>
<tr>
<th><strong>libibverbs with InfiniBand</strong></th>
<th><strong>libfabric with InfiniBand</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>sge</td>
<td>void *</td>
</tr>
<tr>
<td>send_wr</td>
<td>size_t</td>
</tr>
<tr>
<td>next</td>
<td>void *</td>
</tr>
<tr>
<td>num_sge</td>
<td>fi_addr_t</td>
</tr>
<tr>
<td>opcode</td>
<td>void *</td>
</tr>
<tr>
<td>flags</td>
<td></td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>76+8 = 84</strong></td>
</tr>
</tbody>
</table>

Generic entry points result in additional memory reads/writes

Interface parameters can force branches in the provider code

Move operation flags into initialization code path for optimal SW paths
# Memory Footprint

## Per peer addressing data

<table>
<thead>
<tr>
<th></th>
<th>libibverbs with InfiniBand</th>
<th>libfabric with InfiniBand</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td><strong>Data</strong></td>
<td><strong>Size</strong></td>
</tr>
<tr>
<td>struct *</td>
<td>ibv_ah</td>
<td>8</td>
</tr>
<tr>
<td>uint32</td>
<td>QPN</td>
<td>4</td>
</tr>
<tr>
<td>uint32</td>
<td>QKey</td>
<td>4 [0]</td>
</tr>
<tr>
<td>ibv_ah</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>36</strong></td>
</tr>
</tbody>
</table>

### Map Address Vector:
- encodes peer address
- direct mapping to HW command data

### Index Address Vector:
- minimal footprint
- requires lookup/calculation for peer address

<table>
<thead>
<tr>
<th>IB Data:</th>
<th>DLID</th>
<th>SL</th>
<th>QPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size:</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Thank You
OpenFabrics Interfaces
Working Group

Develop an extensible, open source framework and interfaces aligned with ULP and application needs for high-performance fabric services

ofiwg@lists.openfabrics.org

github.com/ofiwg
Architecture

Libfabric Enabled Applications

libfabric

Control Services
- Discovery

Communication Services
- Connection Management
- Address Vectors

Completion Services
- Event Queues
- Counters

Data Transfer Services
- Message Queues
- RMA
- Tag Matching
- Atomics

OFI Provider
- Discovery
- Connection Management
- Address Vectors
- Event Queues
- Counters
- Message Queues
- RMA
- Tag Matching
- Atomics

NIC
- TX Command Queues
- RX Command Queues

MPI
SHMEM
PGAS