Persistent Memory over Fabrics
An Application-centric view

Paul Grun
Cray, Inc
OpenFabrics Alliance Vice Chair
Agenda

OpenFabrics Alliance Intro

OpenFabrics Software

Introducing OFI - the OpenFabrics Interfaces Project

OFI Framework Overview – Framework, Providers

Delving into Data Storage / Data Access

Three Use Cases
The OpenFabrics Alliance (OFA) is an open source-based organization that develops, tests, licenses, supports and distributes OpenFabrics Software (OFS). The Alliance’s mission is to develop and promote software that enables maximum application efficiency by delivering wire-speed messaging, ultra-low latencies and maximum bandwidth directly to applications with minimal CPU overhead.

https://openfabrics.org/index.php/organization.html
OpenFabrics Alliance – selected statistics

- **Founded in 2004**

- **Leadership**
  - Susan Coulter, LANL – Chair
  - Paul Grun, Cray Inc. – Vice Chair
  - Bill Lee, Mellanox Inc. – Treasurer
  - Chris Beggio, Sandia – Secretary (acting)
  - 14 active Directors/Promoters (Intel, IBM, HPE, NetApp, Oracle, Unisys, Nat'l Labs…)

- **Major Activities**
  1. Develop and support open source network stacks for high performance networking
     - OpenFabrics Software - OFS
  2. Interoperability program (in concert with the University of New Hampshire InterOperability Lab)
  3. Annual Workshop – March 27-31, Austin TX

- **Technical Working Groups**
  - OFIWG, DS/DA, EWG, OFVWG, IWG
  - [https://openfabrics.org/index.php/working-groups.html](https://openfabrics.org/index.php/working-groups.html) (archives, presentations, all publicly available)
OpenFabrics Alliance – selected statistics

- **Founded in 2004**

- **Leadership**
  - Susan Coulter, LANL – Chair
  - Paul Grun, Cray Inc. – Vice Chair
  - Bill Lee, Mellanox Inc. – Treasurer
  - Chris Beggio, Sandia – Secretary (acting)
  - 14 active Directors/Promoters (Intel, IBM, HPE, NetApp, Oracle, Unisys, Nat’l Labs…)

- **Major Activities**
  1. Develop and support open source network stacks for high performance networking
     - OpenFabrics Software - OFS
  2. Interoperability program (in concert with the University of New Hampshire InterOperability Lab)
  3. Annual Workshop – March 27-31, Austin TX

- **Technical Working Groups**
  - OFIWG, DS/DA, ZWG, OFVWG, IWG
  - [https://openfabrics.org/index.php/working-groups.html](https://openfabrics.org/index.php/working-groups.html) (archives, presentations, all publicly available)

today’s focus
OpenFfabsics Software (OFS)

Open Source APIs and software for advanced networks

Emerged along with the nascent InfiniBand industry in 2004

Soon thereafter expanded to include other IB-based networks such as RoCE and iWARP

Wildly successful, to the point that RDMA technology is now being integrated upstream. Clearly, people like RDMA.

OFED distribution and support: managed by the Enterprise Working Group – EWG

Verbs development: managed by the OpenFabrics Verbs Working Group - OFVWG
Historically, network APIs have been developed ad hoc as part of the development of a new network.

To wit: today’s Verbs API is the implementation of the verbs semantics specified in the InfiniBand Architecture.

But what if a network API was developed that catered specifically to the needs of its consumers, and the network beneath it was allowed to develop organically?

What would such a resulting API look like?
Introducing the OpenFabrics Interfaces Project

- OpenFabric Interfaces Project (OFI)
  - Proposed jointly by Cray and Intel, August 2013
  - Chartered by the OpenFabrics Alliance, w/ Cray and Intel as co-chairs

- Objectives
  - ‘Transport neutral’ network APIs
  - ‘Application centric’, driven by application requirements

OFI Charter

Develop, test, and distribute:
1. An extensible, open source framework that provides access to high-performance fabric interfaces and services
2. Extensible, open source interfaces aligned with ULP and application needs for high-performance fabric services

OFIWG will not create specifications, but will work with standards bodies to create interoperability as needed
OpenFabrics Software (OFS)

Result:
1. An extensible interface driven by application requirements
2. Support for multiple fabrics
3. Exposes an interface written in the language of the application
OFI Framework

OFI consists of two major components:
- a set of defined APIs (and some functions) – libfabric, kfabric
- a set of wire-specific ‘providers’ - ‘OFI providers’

Think of the ‘providers’ as an implementation of the API on a given wire.
OpenFabrics Interfaces - Providers

All providers expose the same interfaces, None of them expose details of the underlying fabric.

Current providers: sockets, verbs, usnic, gni, mlx, psm, psm2, udp, bgq
OFI Framework – a bit more detail

Network consumers

Framework
Control Services
Discovery

Communication Services
Connection Management
Address Vectors

Completion Services
Event & Completion Queues
Counters

Data Transfer Services
Msg queues
RMA
Tag Matching
Atoms

Provider
Discovery
Connection Management
Address Vectors

NIC
Application-centric design means that the working groups are driven by use cases:

Data Storage / Data Access, Distributed and Parallel Computing…
OFI Project Overview – work group structure

Legacy Apps
- Sockets apps
- IP apps

Data Analysis
- Structured data
- Unstructured data

Data Storage, Data Access
- Data Storage
  - object, file, block
  - storage class memory
- Data Access
  - remote persistent memory

Distributed and Parallel Computing
- Msg Passing
  - MPI applications
- Shared memory
  - PGAS languages

kernel mode
‘kfabric’

user mode
‘libfabric’
Since the topic today is Persistent Memory…
Data Storage, Data Access?

**Data Storage / Data Access Working Group**

- **DS**
  - object, file, block
  - storage class mem

- **DA**
  - persistent memory

**Key Use Cases:**
1. Lustre
2. NVMe
3. Persistent Memory

Reminder:
- libfabric: User mode library
- kfabric: Kernel modules

Non-volatile memory (storage class memory, persistent memory)
adding a new network type may require creating a new LND
Data Storage – Lustre LND Architecture

drawing courtesy of Intel
Data Storage – NVMe over fabrics

NVMe is a block storage protocol designed for operation with storage class memory devices.
NVMe/F is an extension, allowing access to an NVMe device over a fabric.

NVMe/F leverages the characteristics of verbs to good effect for verbs-based fabrics.
Data Storage – Future Lustre LND Architecture

drawing courtesy of Intel
Data Storage – enhanced APIs

- kfabric as a second native API for NVMe
- kfabric as a possible ‘LND’ for LNET

A kfabric verbs provider exists today (not upstream), meaning that kfabric can be run over a conventional verbs-based fabric. No modifications are necessary.

Waiting for a kfabric provider to emerge
A look at Persistent Memory

Applications tolerate long delays for storage, but assume very low latency for memory
- Storage systems are generally asynchronous, target driven, optimized for cost
- Memory systems are synchronous, and highly optimized to deliver the lowest imaginable latency with no CPU stalls

Persistent Memory over fabrics is somewhere in between:
- Much faster than storage, but not as fast as local memory (usually)

How to treat PM over fabrics?
- Build tremendously fast remote memory networks, or
- Find ways to hide the latency, making it look for the most part like memory, but cheaper and remote
Data Access – PM operations

For ‘normal’ fabrics, the responder returns an ACK when the data has been received by the end point.

- It may not be globally visible, and/or

- It may not yet be persistent

Need an efficient mechanism for indicating when the data is globally visible, and is persistent
Data Access – key fabric requirements

Objectives:
1. Give the client app control over commits to persistent memory
2. Return to the client distinct indications of
   • global visibility, persistence
3. Optimize completion protocols to avoid round trips

Caution: OFA does not define wire protocols, it defines the semantics seen by the consumer
Possible converged I/O stack

- **Kernel application**
- **User app**

**VFS / Block Layer**
- **SCSI**
- **NVMe**
- **ulp**

**Local I/O**
- **HBA**
- **SSD**
- **NVDIMM**

**Remote I/O**
- **SRP, iSER, NVMe/F, NFSoRDMA, SMB Direct, LNET/LND,…**
- **iSCSI**

**Local byte addressable**
- **byte access**

**Remote byte addressable**
- **byte access**

**Remote block/file I/O**
- **fabric-specific device**
- **NIC, RNIC**
- **HCA**

**VFS / Block I/O / Network FS / LNET**
- **kfabric**
- **kverbs**

* Verbs exists as a kfabric provider today
Discussion - Between Two Ferns

Doug Voigt - SNIA NVM PM TWG Chair
Paul Grun – OFA vice Chair, co-chair OFIWG, DS/DA

Are we going in the right direction?
Are we aligned? Perfectly aligned? Almost aligned?
Are we focusing on the right use cases?
How to go forward from here?
Thank You
Backup
kfabric API – very similar to existing libfabric

Consumer APIs
- kfi_getinfo() kfi_fabric() kfi_domain() kfi_endpoint() kfi_cq_open() kfi_ep_bind()
- kfi_listen() kfi_accept() kfi_connect() kfi_send() kfi_recv() kfi_read() kfi_write()
- kfi_cq_read() kfi_cq_sread() kfi_eq_read() kfi_eq_sread() kfi_close() …

Provider APIs
- kfi_provider_register()
  During kfi provider module load a call to kfi_provider_register() supplies the kfi api with
  a dispatch vector for kfi_* calls.

- kfi_provider_deregister()
  During kfi provider module unload/cleanup kfi_provider_deregister() destroys the kfi_*
  runtime linkage for the specific provider (ref counted).

http://downloads.openfabrics.org/WorkGroups/ofiwg/dsda_kfabric_architecture/
Data Storage – Current Lustre LND Architecture

Adding a new network type requires introducing a new LND
One option is to define a new LND for each new network type.
Data Storage – Current Lustre LND Architecture

Or, backport LNET to kfabric, allowing use of any kfabric provider
By analogy, storage protocols include an ‘ending status’ phase, signaling to the client that the data has been safely stored.

No such client/server model exists for a remote persistent memory device.

Hence, there is no convenient way to know if data has been safely stored within the persistence domain.

Synchronization is achieved by the underlying I/O protocol.