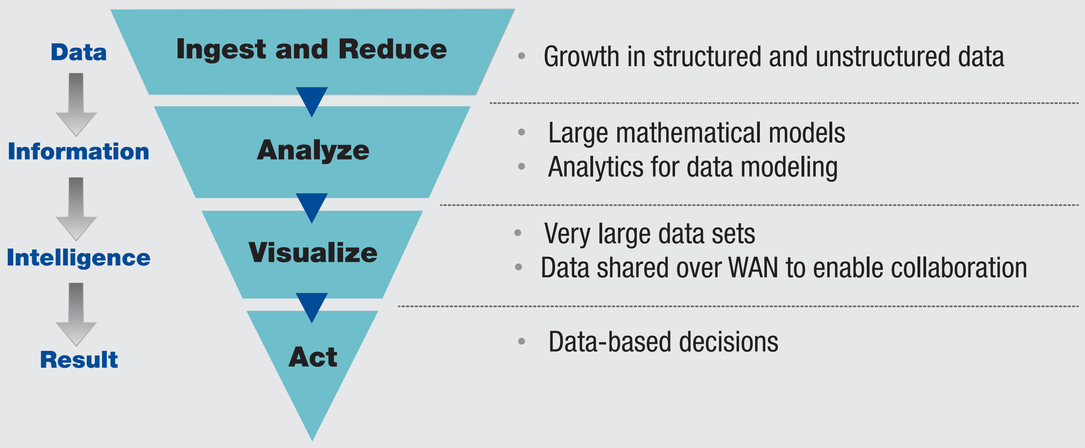
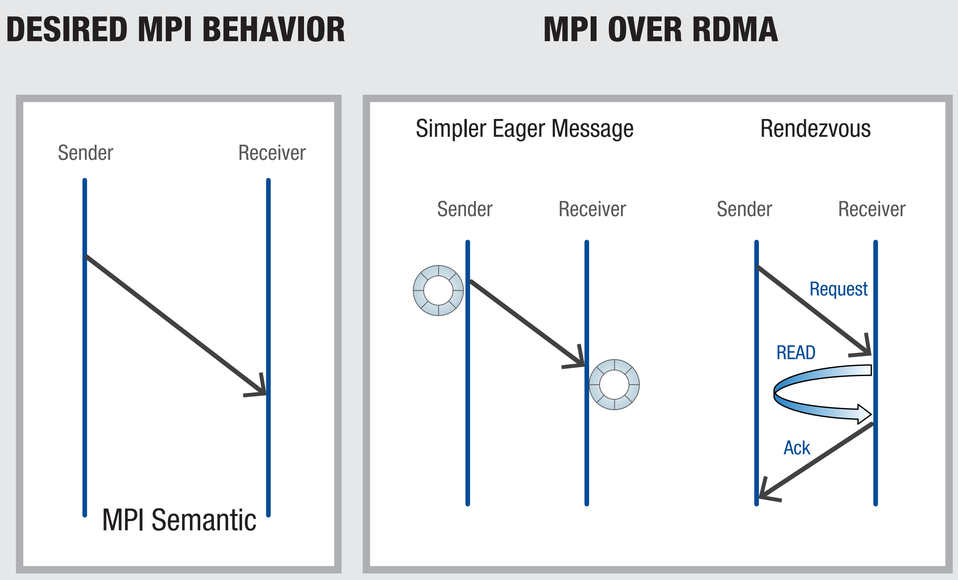
Accelerating Improvements in HPC Application I/O Performance and Efficiency

Problem: Expanding usage models, evolving technology leads to re-think I/O model



New usages (big data, clouds, storage at a distance) combined with evolving technology (non-volatile memory, solid state storage, heterogeneous processors, multi-core and more) is creating new types of processes with very different communications requirements. This is creating an imperative to re-think the I/O model.

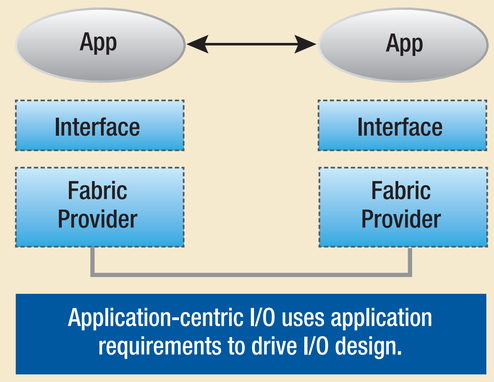
Problem: Scaling MPI for HPC



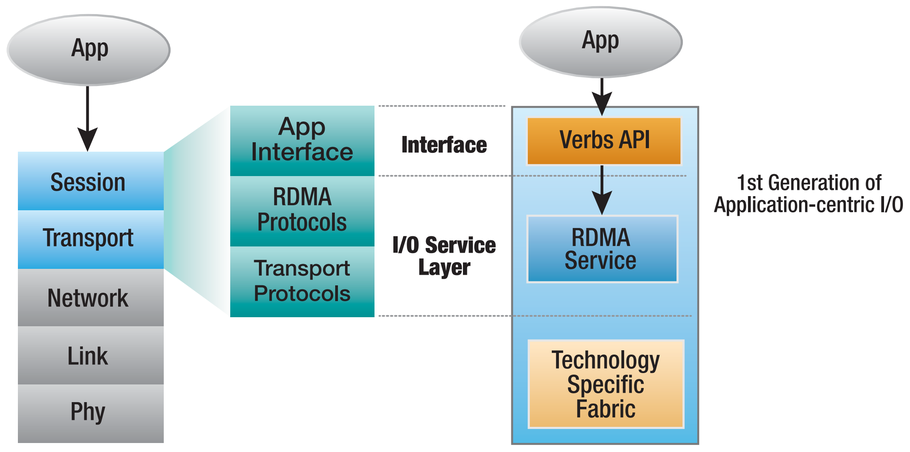
RDMA advanced the state of the art in scalable MPI applications but there needs to be a better approach to implementing MPI

Objectives

1. Improve scalability and performance of HPC systems for mathematical modeling applications by focusing on I/O for
   - Message passing apps (e.g. MPI)
   - Shared memory apps (e.g. PGAS)
   - Storage and visualization apps
2. Increase support for data driven modeling systems by focusing on I/O for
   - Large unstructured datasets (Big Data analytics)
   - Data access and storage at a distance
   - Cloud-based storage and computing infrastructures

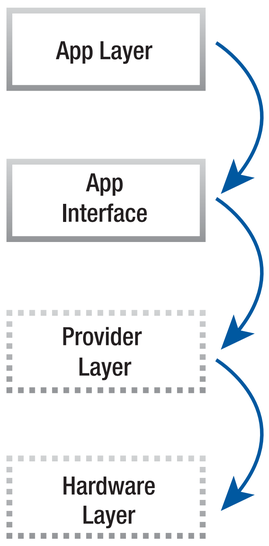


Start with RDMA Architecture



- RDMA improved app performance and scalability by applying the principles of "application-centric I/O"
- RDMA architecture consists of an application interface and an I/O services layer
- Use application requirements to drive an update to the app interface and I/O services layer

Define the Application Interface by Focusing on Application Requirements



Understand application I/O characteristics (MPI, PGAS, storage, WAN, Big Data…)

Let those characteristics drive the interface definition(s)

This drives the fabric feature set(s)

And the choice of underlying fabric

The long list of application requirements
- datagram – streaming
- connected, unconnected
- client-server, point to point
- multicast
- tag matching
- active messages
- reliable datagram
- strided transfers
- one-sided reads/writes
- send-receive transfers
- triggered transfers
- atomic operations
- collective operations
- Synch / asynch transfers
- QoS
- ordering, flow control xfers

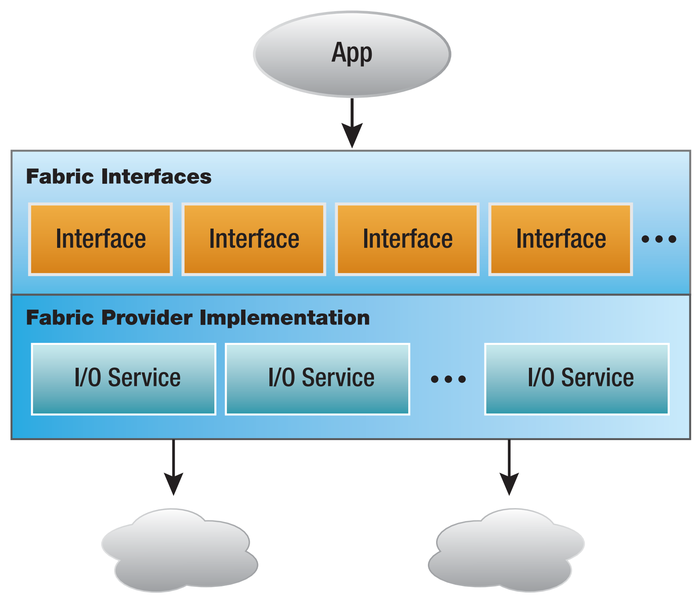
Work to be Done

Observations

– Application requirements are broad and divergent
– A single API cannot meet all functional and performance requirements
– Any particular app is likely to need only a subset from the broad API space

Proposal

1. Create a set of application interfaces (APIs) driven by application requirements
   – Examples: large block transfer, small message passing, collectives, connection management, tag matching
   – Each I/F provides an abstracted I/O service to the application
2. Let I/F definitions drive the description of required I/O services.
   – Vendors will provide optimized implementations of these I/O services.
3. Create a flexible and extensible framework to contain the set of APIs and I/O services



Important point!
The framework is fabric independent. Choice of fabric is driven by customer demand and vendor implementation.

The OpenFabrics Alliance OpenFramework Working Group (OFWG) was created to move the state of the art in high performance networks to the next level.

Its charter is:

Develop, test, and distribute:

1. Extensible, open source interfaces aligned with application demands for high-performance fabric services.
2. An extensible, open source framework that provides access to high-performance fabric interfaces and services.

More information

SC'13 BoF – Discussing an I/O Framework to Accelerate Improvements in Application I/O Performance
November 21, 12:15PM, Room 601/603

OpenFabrics Alliance – www.openfabrics.org
OpenFramework Working Group - http://lists.openfabrics.org/cgi-bin/mailman/listinfo
OpenFramework Working Group Co-chairs

Paul Grun (Cray, Inc.) grun@cray.com
Sean Hefty (Intel, Inc.) sean.hefty@intel.com

OPENFABRICS ALLIANCE