OPENFABRICS ALLIANCE

Evolving OFS – A Tops Down Approach

Paul Grun
Cray Inc.
April 21, 2013

Technology is like a shark…it's always moving.

It if stands still, it dies.

This session is about how to keep the state of the art in I/O moving

This session…

…will be completely different from any other session you'll see at the workshop this year.

It comes in two parts; this is part one. Part two comes on Wednesday.

Every session being presented this year is relevant to the key question for the workshop, which is:

"Figure out how to keep the state of the art in I/O moving forward"

Your challenge is to actively engage in addressing this question.

We'll reconvene on Wednesday to work on the answer.

“Change you can count on”

- The scale of HPC systems is increasing dramatically
- New compute models (e.g. heterogeneous computing) are becoming common
- Many-, multi-core processors core counts expanding
- The never-ending discussion of programming models – message passing, shared memory
- Basic technology (e.g. memory, network speeds…) is not standing still

Not only that, but…

- The world is demanding new ways of analyzing avalanches of data
  → Big Data

- People want to store and access data in new ways
  → the Cloud

- Larger, complex problems are requiring new collaboration methods
  → Data access over long distances

Clearly, change is afoot

In (at least) three areas:

1. Scalability
2. The way that computer systems are built
3. The way that users interact with each other and with data

All three may impact the way that we look at and implement I/O

Our workshop goal

“Figure out how to keep the state of the art in I/O moving forward”

Challenges	The Ask	Who is the ‘Askee’?
Scalability	(What needs to change in the area of I/O to expand scalability?)	??
Technology	(How does changing technology impact I/O?)	??
Usage	(How do changing usage models impact I/O requirements?)	??

Our challenge is to fill in this table

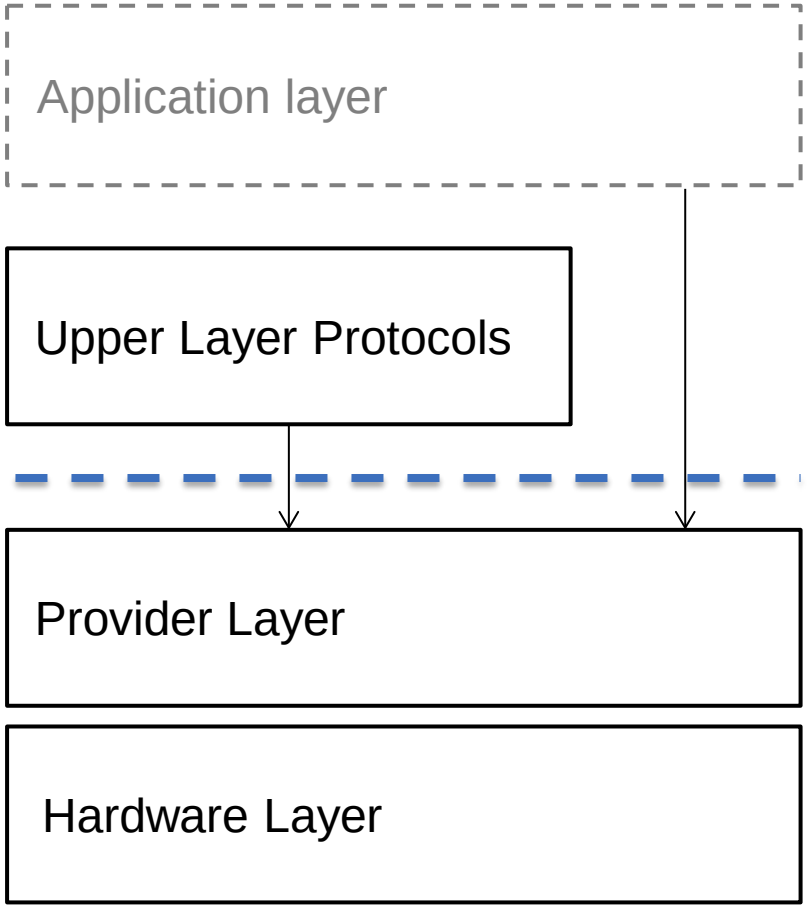
To fill in the table…

Open Fabrics Software, OFS, uniquely fills a vital need for I/O services, especially in certain key computing environments.

Sometimes, things change.

Which means that OFS needs to evolve as well.

The foundation of OFS



OFS is built on top of RDMA. (Not exclusively, but pretty much).

Applications are either coded to the Verbs API, or they rely on a ULP

So evolving OFS may also mean evolving the network infrastructure that underlies it

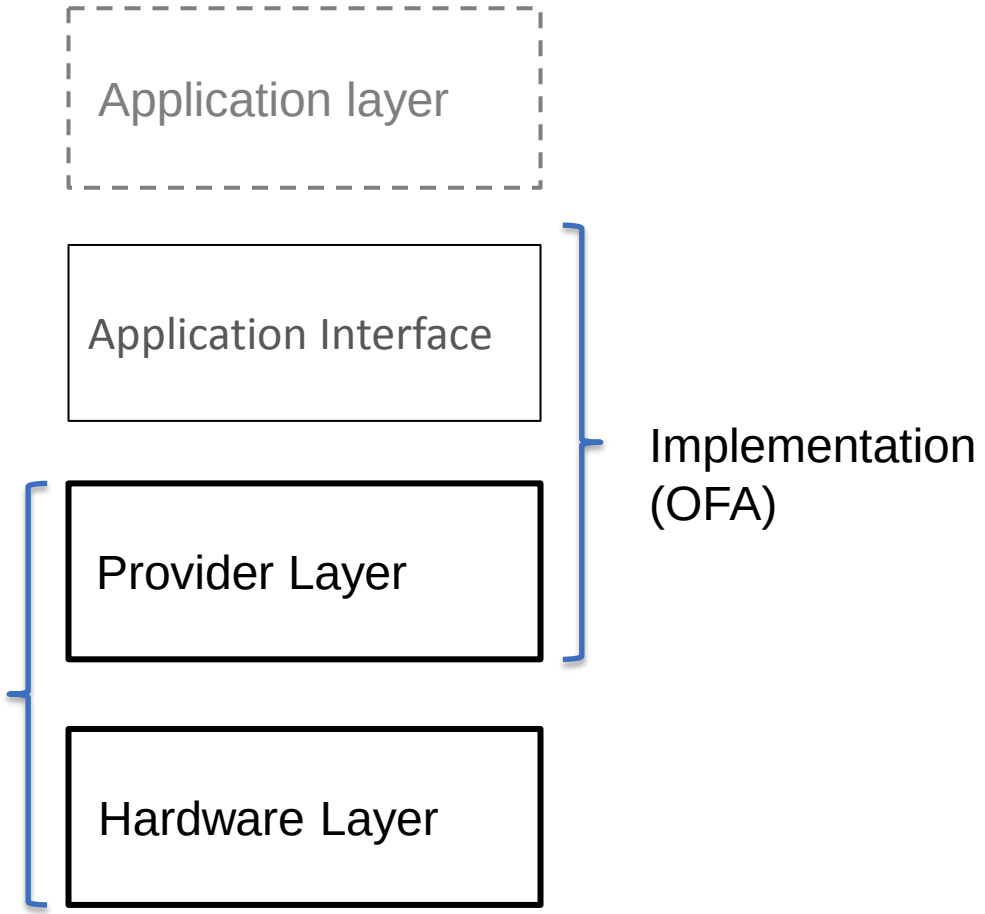
In other words, this isn't solely an OFA problem.

Specification, implementation

OFS is the implementation of an I/O stack whose behavior is specified by an industry standard specification.

The spec defines the underlying network.

Specification (InfiniBand Architecture™, iWARP, RoCE)



Network b/w roadmap

As far as the network goes, it's not just a bandwidth issue…

…it's really about how applications communicate.

(Actually, it always has been)

So amping up the signaling rate, while important, isn't likely to completely solve the problem.

Challenges ahead

1. **Scalability**: InfiniBand is probably the most scalable standards-based I/O solution in the world today. Yet we know now that classical RDMA-based applications will soon reach unimaginable levels of scale.

2. **Technology**: We must account for changing technologies and architectures

3. **Usages**: I/O has to support the ways that people interact with their data, and with each other. This is changing too

Things have changed over the past 15 years

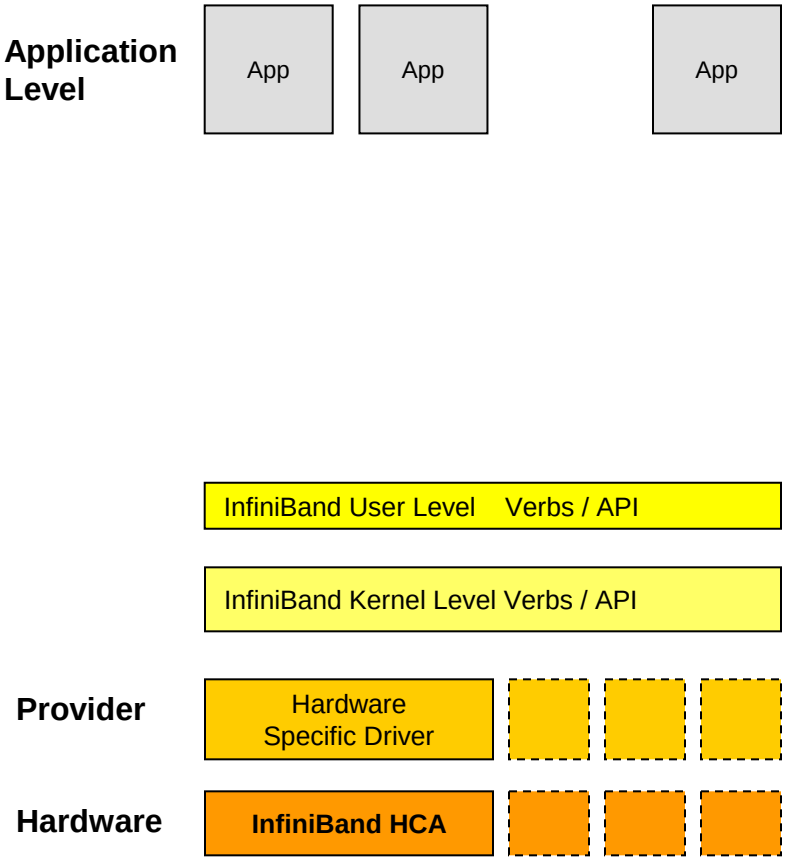
The obvious answer

So how do we move forward from here?

1. Look at raw scalability
2. Understand emerging technology and how it impacts the network
3. Look at classes of application to get a sense for the I/O requirements

Which is exactly how the workshop is organized, but not necessarily in this order

A brief history of OFA (simplified)

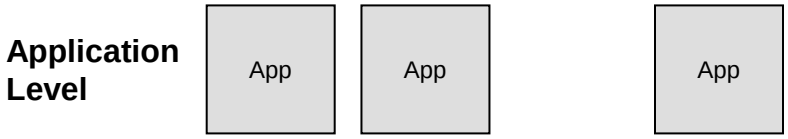


The Open Fabrics Alliance emerged as a way to keep the nascent IB industry from fracturing.

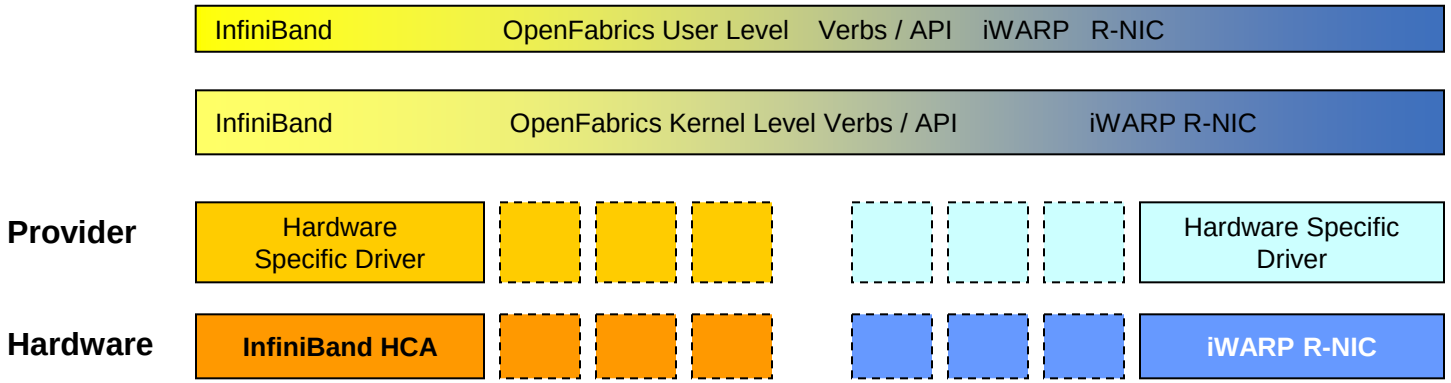
It did this by providing a common, open source Verbs API

You could say that the Verbs API is the heart of OFS

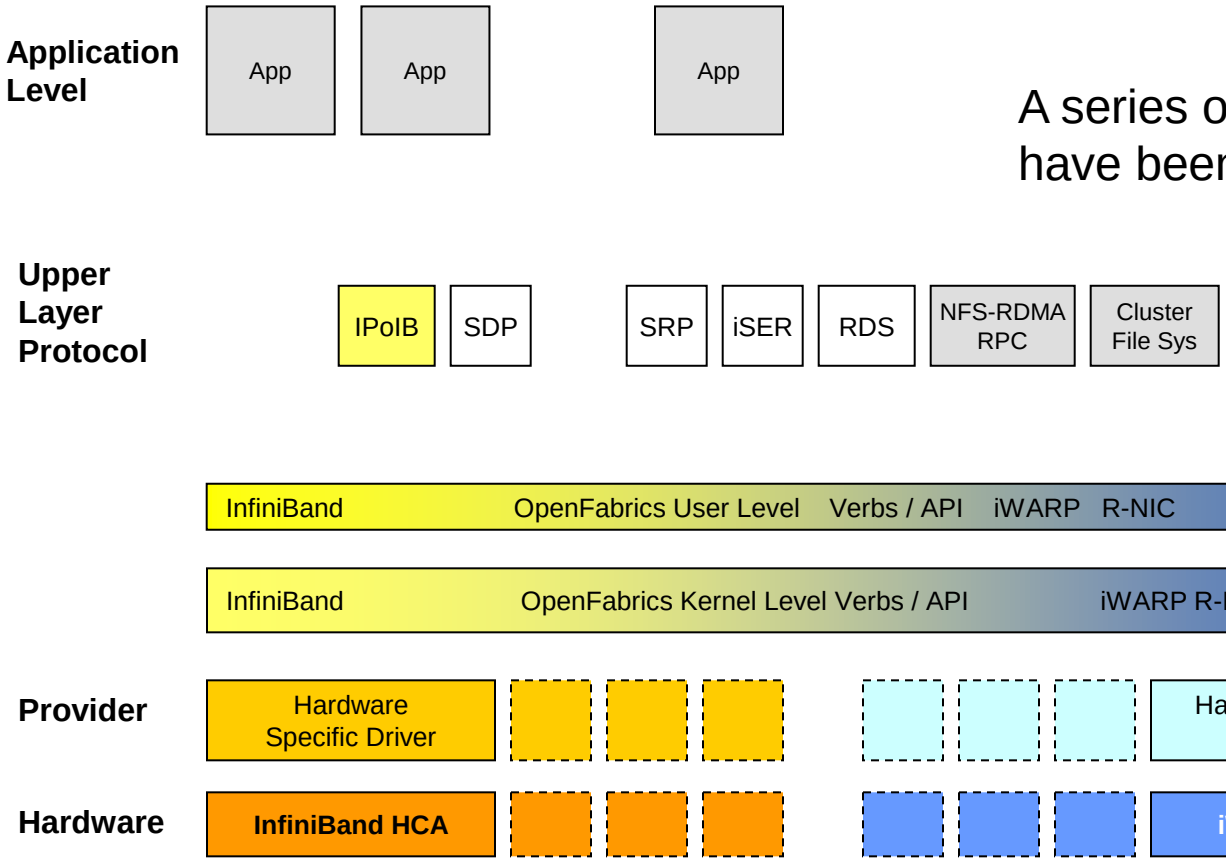
A brief history of OFA (simplified)



New underlying networks have been added over time (iWARP, RoCE), but the basics of OFS are still in the Verbs APIs



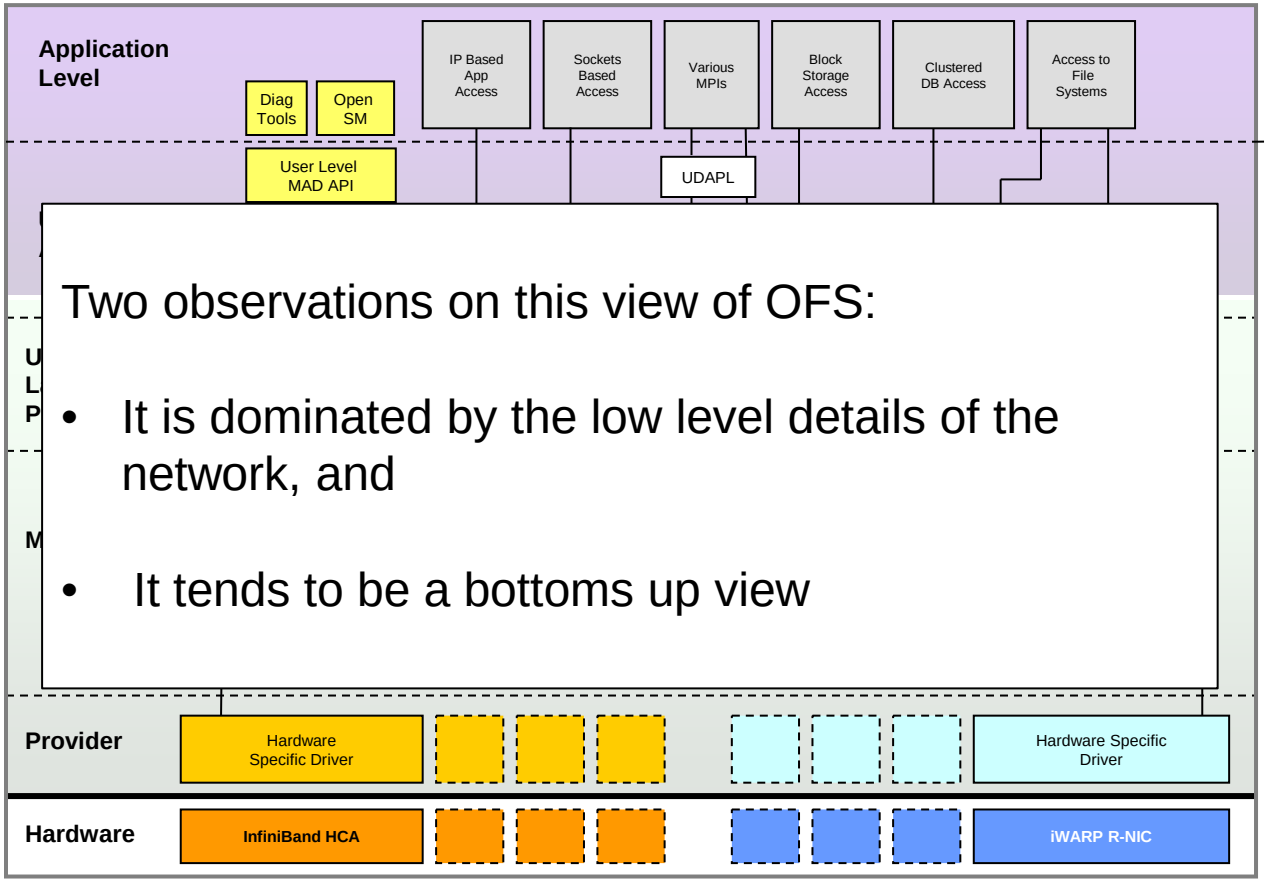
A brief history of OFA (simplified)

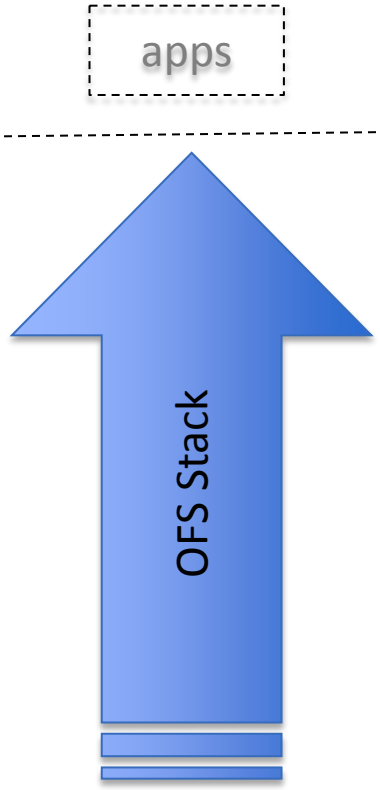


A series of interface definitions have been crafted over time

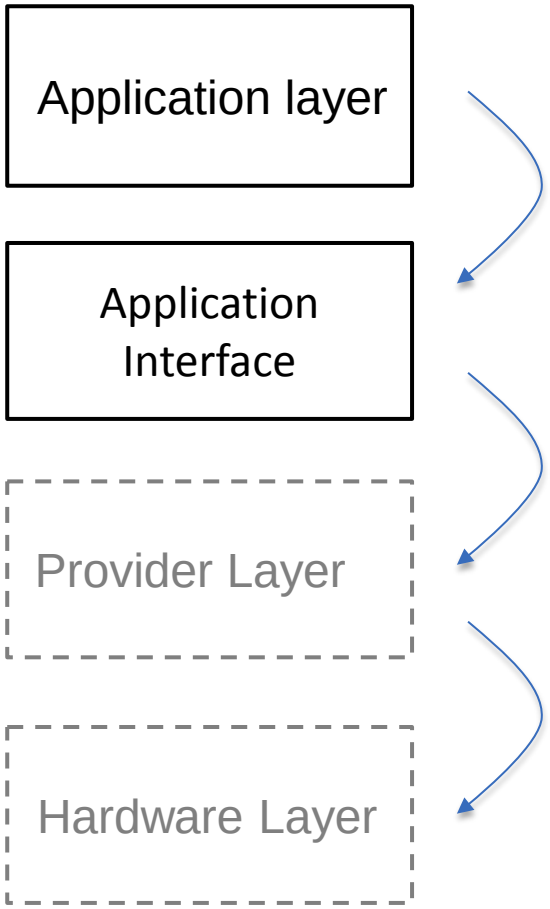
This is the familiar architecture we all know nowadays

Forest for the trees?





Application as driver

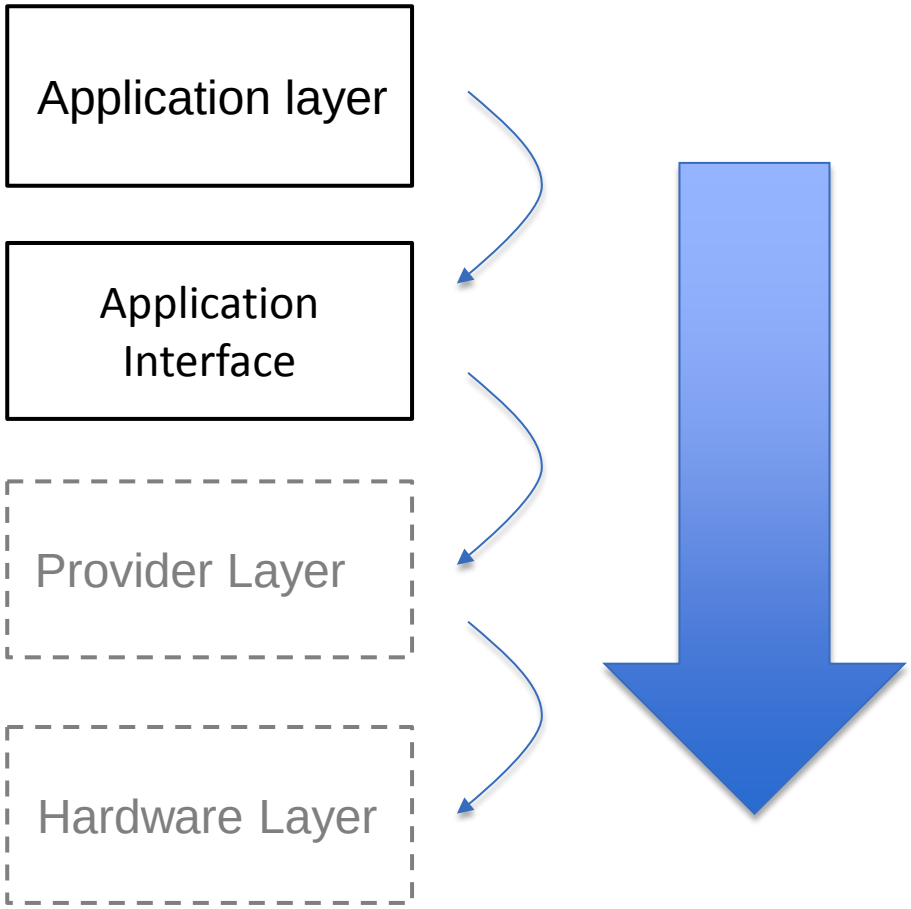


The application layer's I/O requirements tend to drive the definition of the interface layer

…which in turn drives the definition of the underlying network

So it makes sense to start by looking at the applications of interest

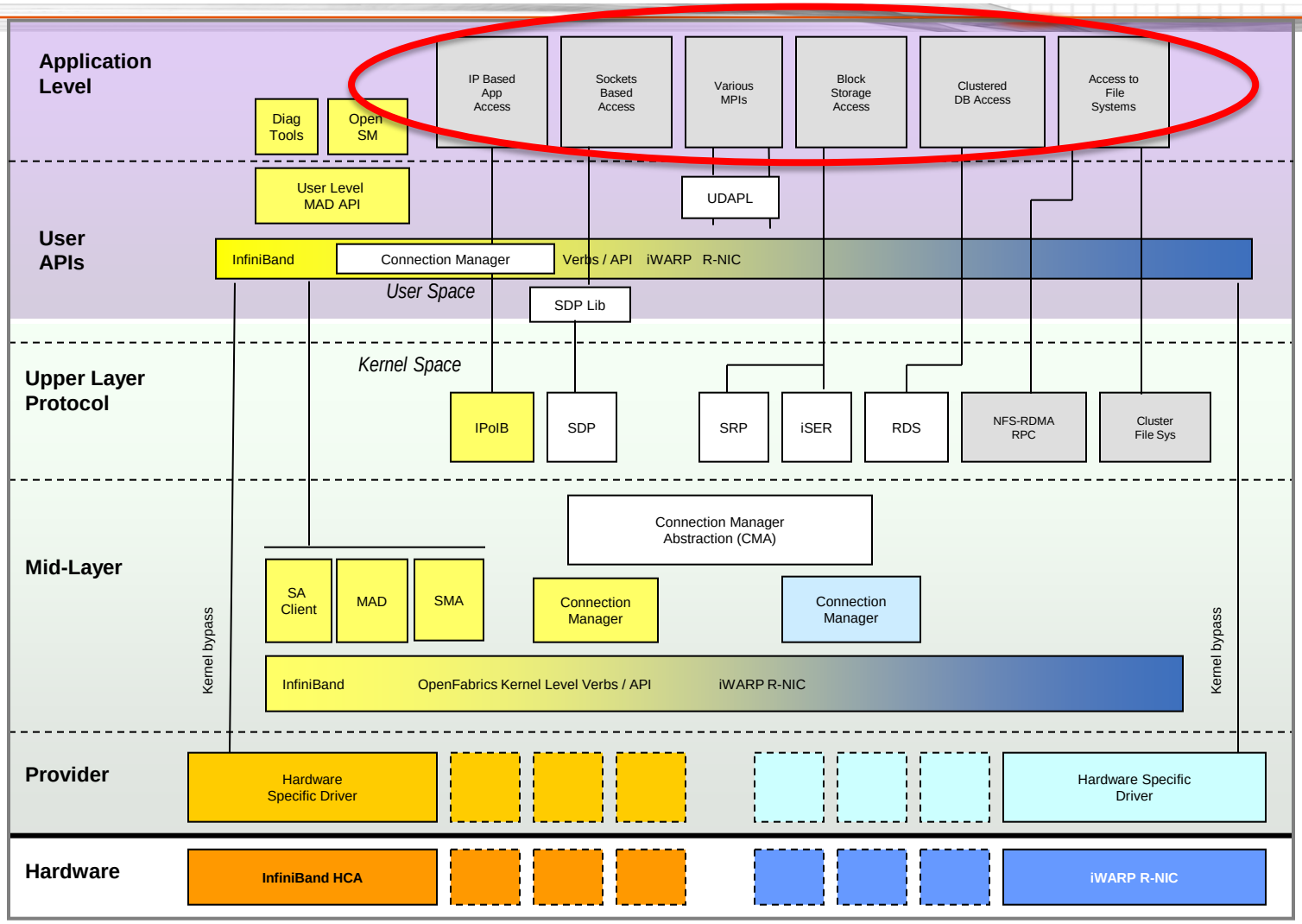
Tops down



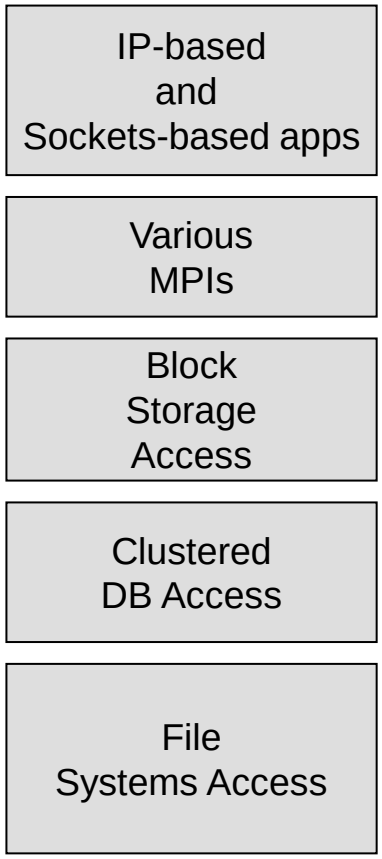
Which tends to recommend a tops down look, if the goal is improving support for applications of interest.

Which it is.

OFS application support today



OFS application support



Support for various types of legacy apps

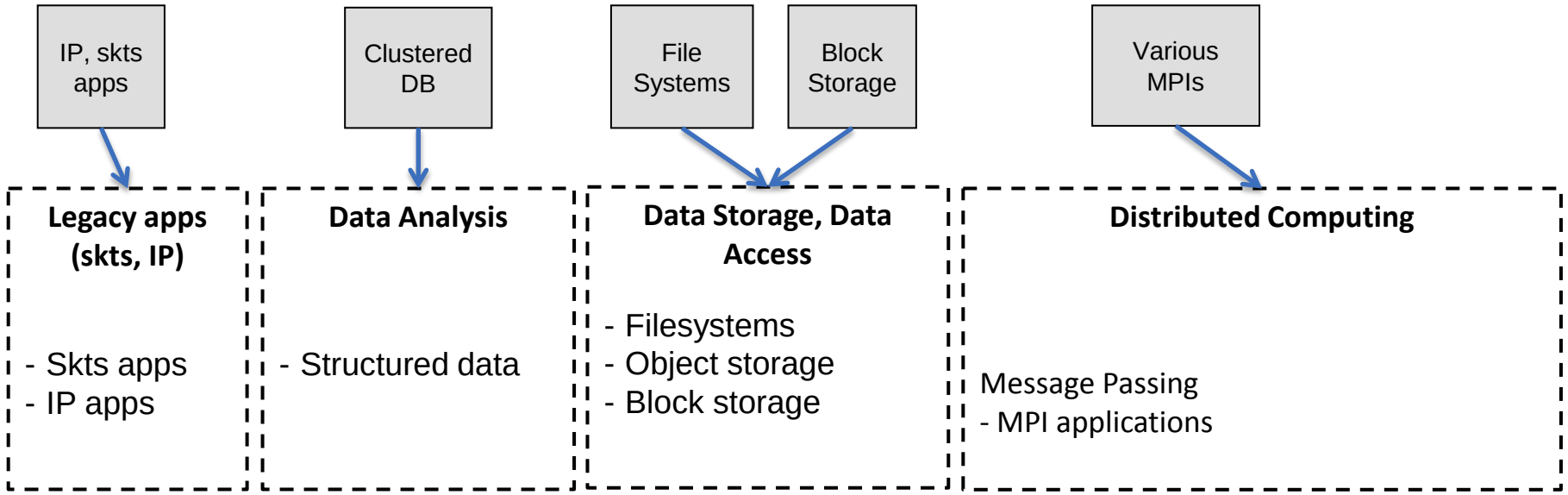
Distributed computing via message passing

Network-attached block storage

Extracting value from structured data

Network-attached file or object storage

Application support

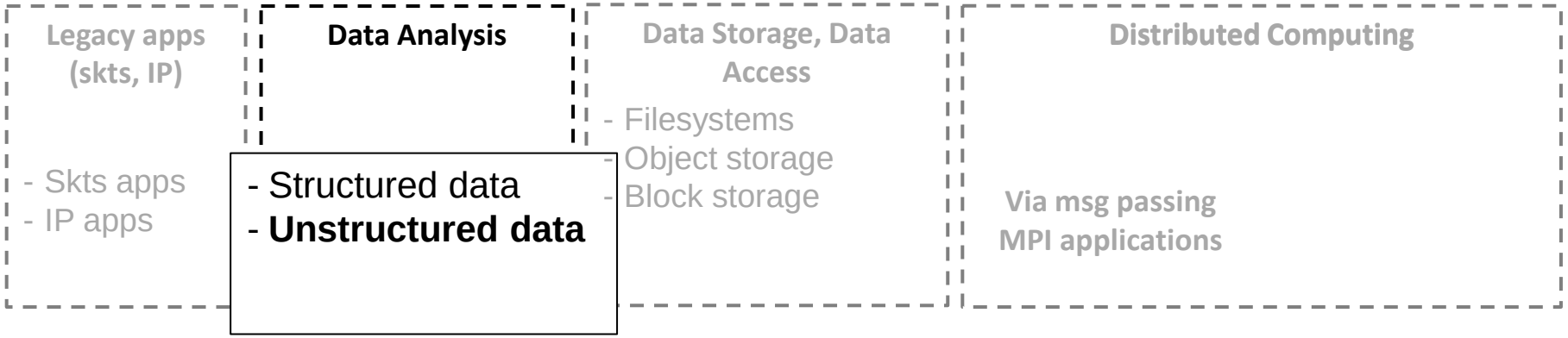


The ways that data is organized, so value can be extracted.

The ways that users store and access data, and the ways that users collaborate through data.

Programming models for processing data

Broadening support

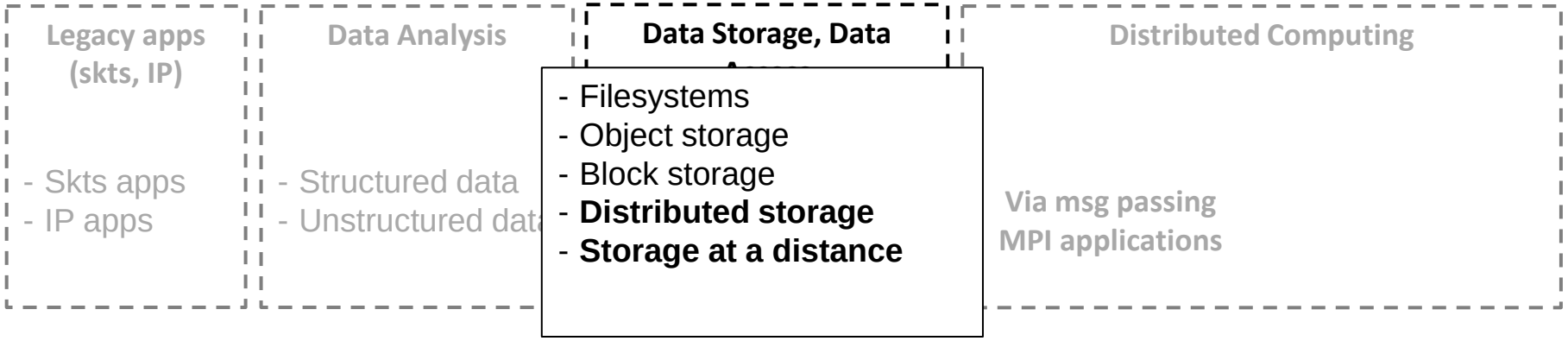


Add to the Data Analysis category:
- Unstructured data, because people want to extract value from avalanches of unorganized data. (which is the essence of Big Data).

Hadoop, for example.

What are the I/O requirements to support tools for accessing and analyzing both structured and unstructured data?

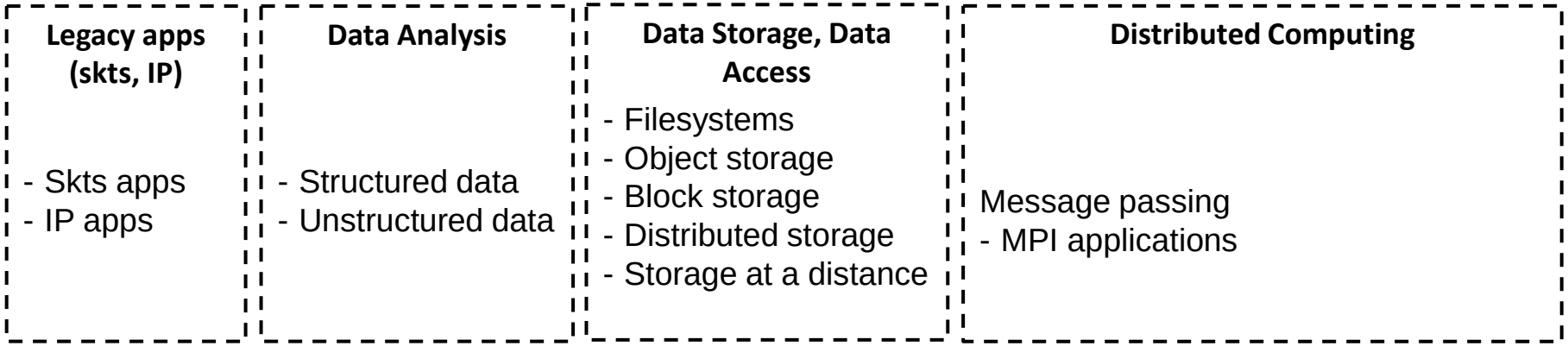
Broadening support



- Distributed storage because of the shift in how data is accessed (e.g. from anywhere) → Cloud storage

- Storage at a distance because distributed teams of users demand the ability to collaborate through common, shared data

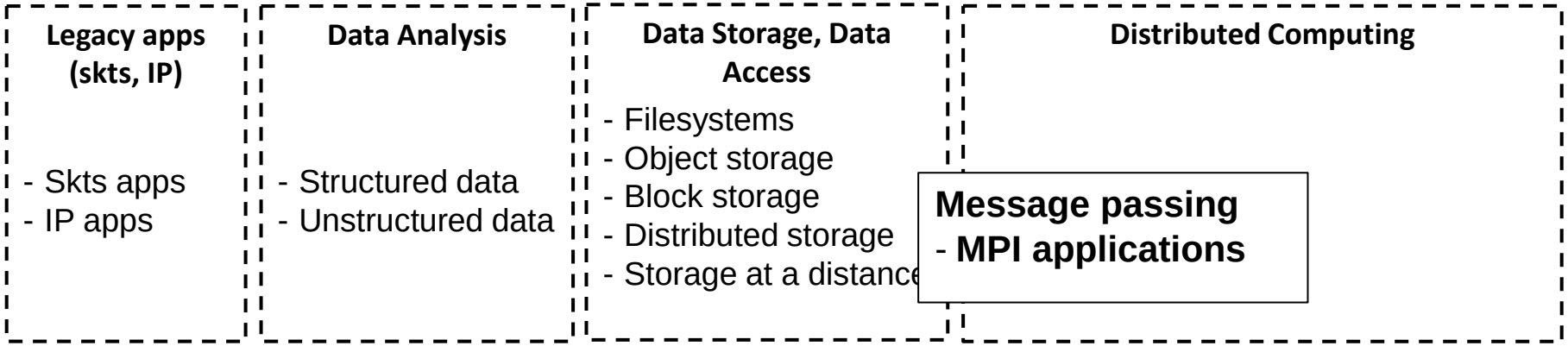
Same question – what are the I/O requirements to support data storage and access?



These examples were mainly about how systems are used...

1. Scalability
2. Technology
3. **Usages -** I/O has to support the ways that people interact with their data

Broadening support

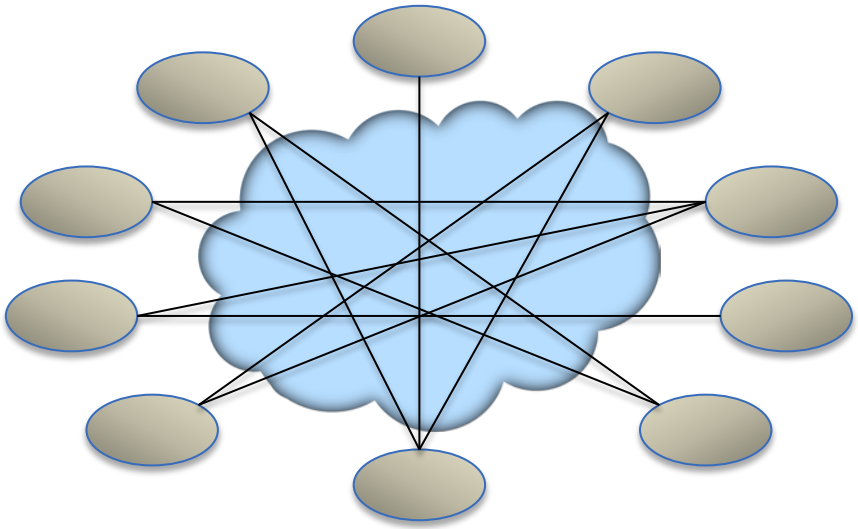


Support for Distributed Computing – improved support for MPI

(Yes, we already have PSM (and MXM) to address this.)

Can we improve the scalability of message passing programming models?

Msg passing scalability

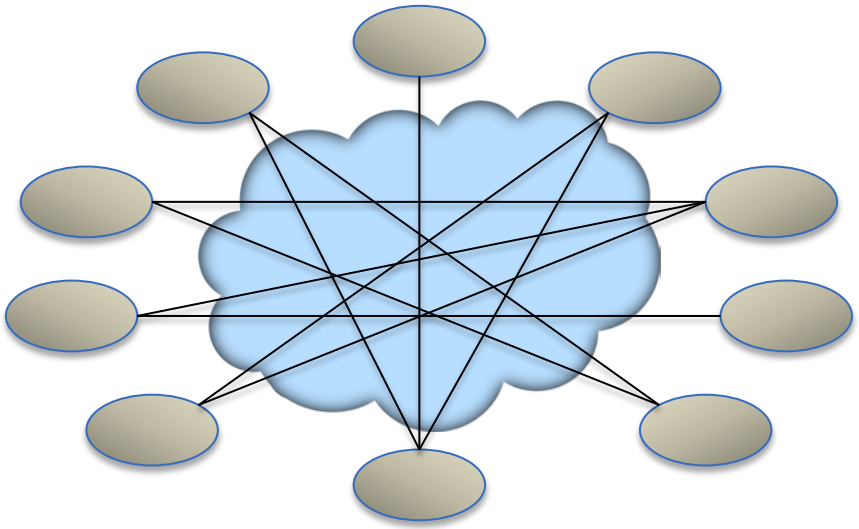


Each instance may require a connection to many other instances.

Especially troublesome using RC mode.

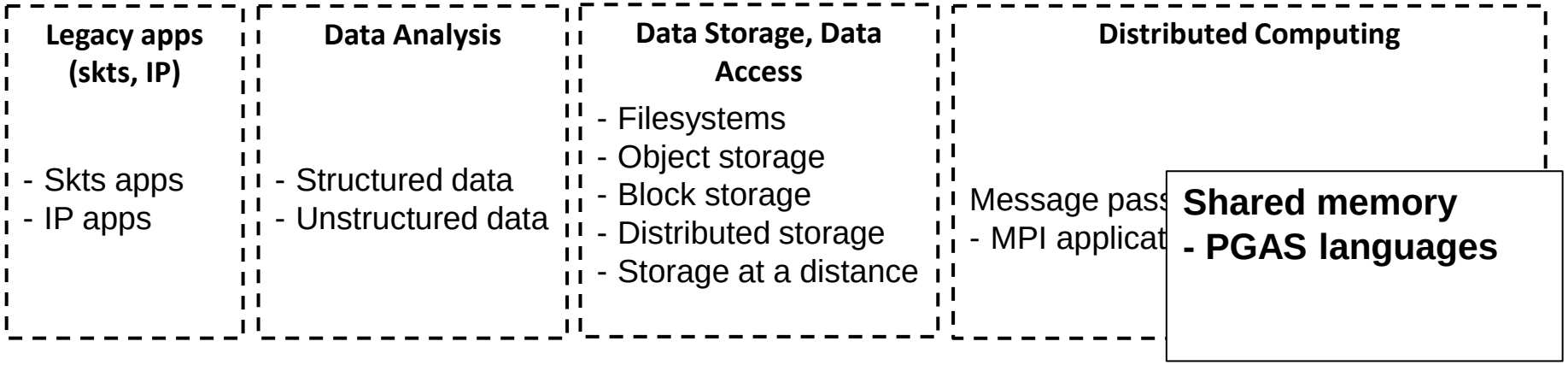
RDMA *as implemented today**, is essentially point-to-point message passing

MPI scalability opportunities



- RC mode consumes QP resources
- RDMA-CM scalability, overhead
- Addressing scalability (LIDs)
- Addressing scalability (LIDs)
- resources per queue pair
- memory registration overhead
- Code bloat due to multiple MPI support
- ‘well-known QPs’ for MPI

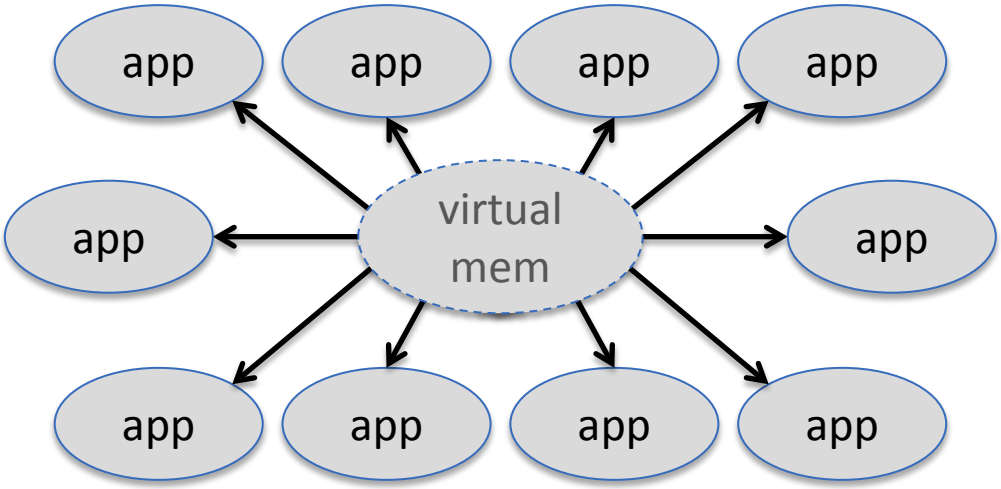
Broadening support



Support for Distributed Computing

Add support for shared memory programming models

Shared mem prog model



Each execution instance creates a single channel to the virtual shared memory

Shared memory systems are desirable because of their scalability characteristics.

MP vs shared memory

There are endless debates over the value of one vs the other.

Some say you can implement shared memory over a MP architecture, and some say the reverse.

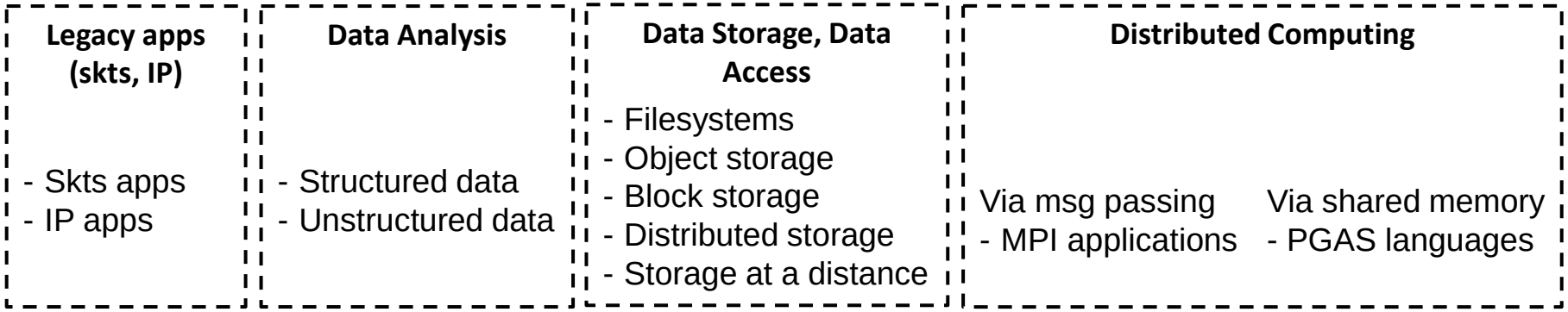
The question is not message passing ***VS*** shared memory.

Rather than debate the merits, we could discuss:

- Where does a message passing architecture make sense and how can it be improved?

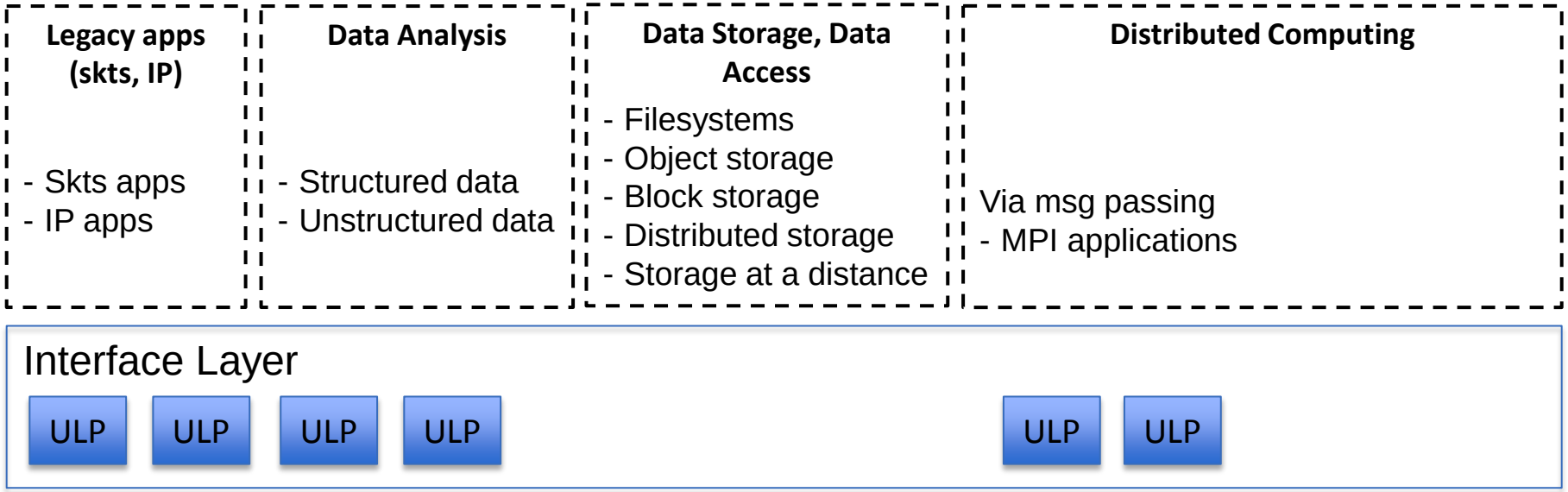
- Should OFS improve its support for shared memory models (e.g. PGAS), and if so, how?

Expanded taxonomy



1. **Scalability** – OFS should support scalable programming models
2. Technology
3. **Usages -** I/O has to support the ways that people interact with their data

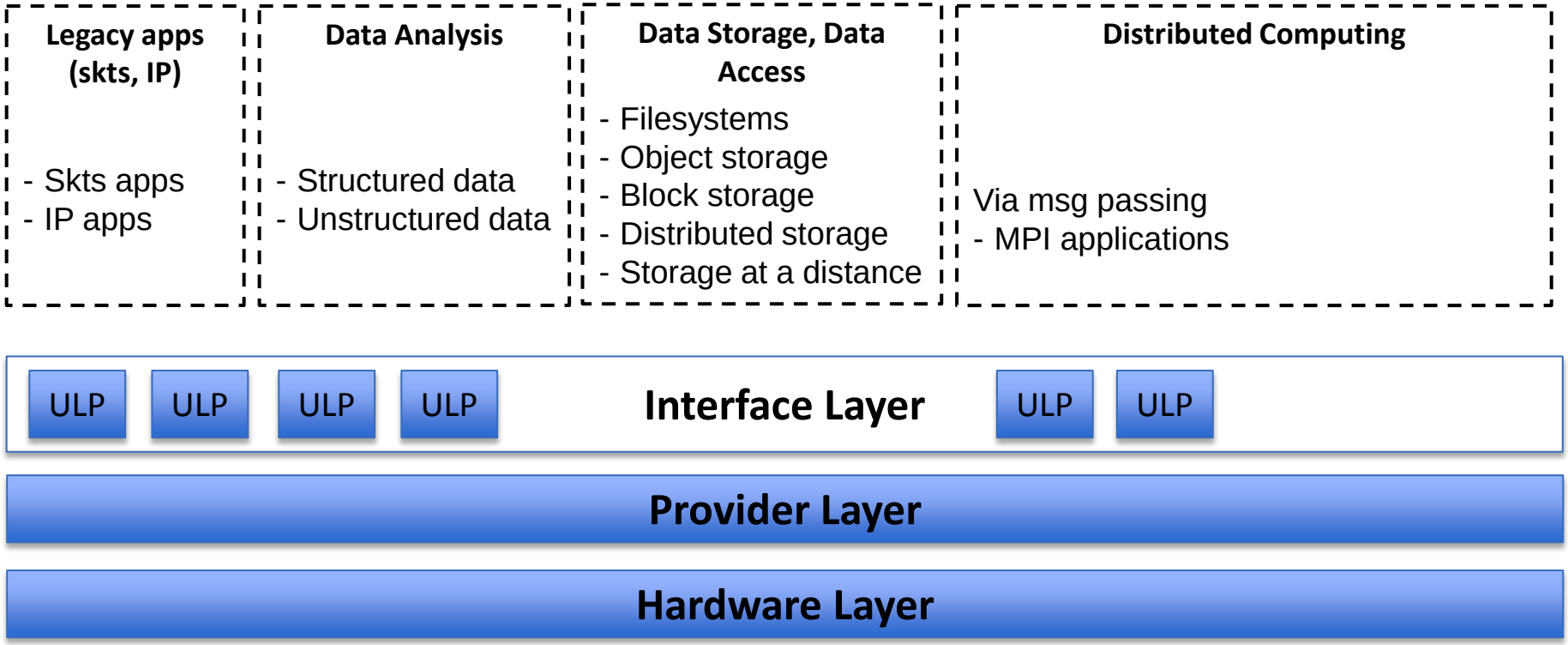
Impacts on I/O architecture



What do the characteristics of each class tell us about the desired interface?
Today, it's verbs, but…

- can the verbs i/f be improved?
- are there classes of apps for which verbs may not be the proper interface? (
- We've already seen the emergence of non-verbs APIs such as PSM, MXM.)

Impacts on I/O architecture



In turn, what do the characteristics of the various interfaces tell us about the required properties of the underlying network?

So far…

1. Scalability
?? 2. The way that computer systems are built
3. The way that users interact with each other and with data

What of the technology?

And technology marches on

A few examples:

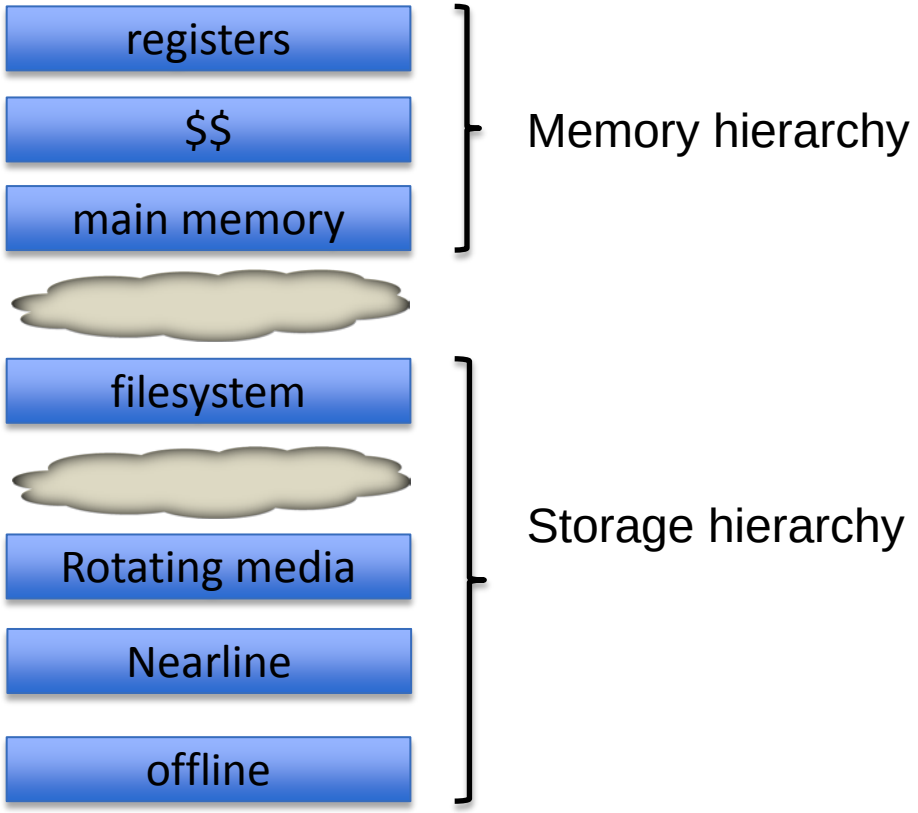
- Solid state memory has the potential to re-make the way that processors store and access data and thus re-define the classical memory hierarchy

- Multicore processing may change the way threads are allocated to cores and thus may change the basic demands placed on the interconnect

- Heterogeneous processing may change the profile of the traffic passing between cores. Small vs large messages?

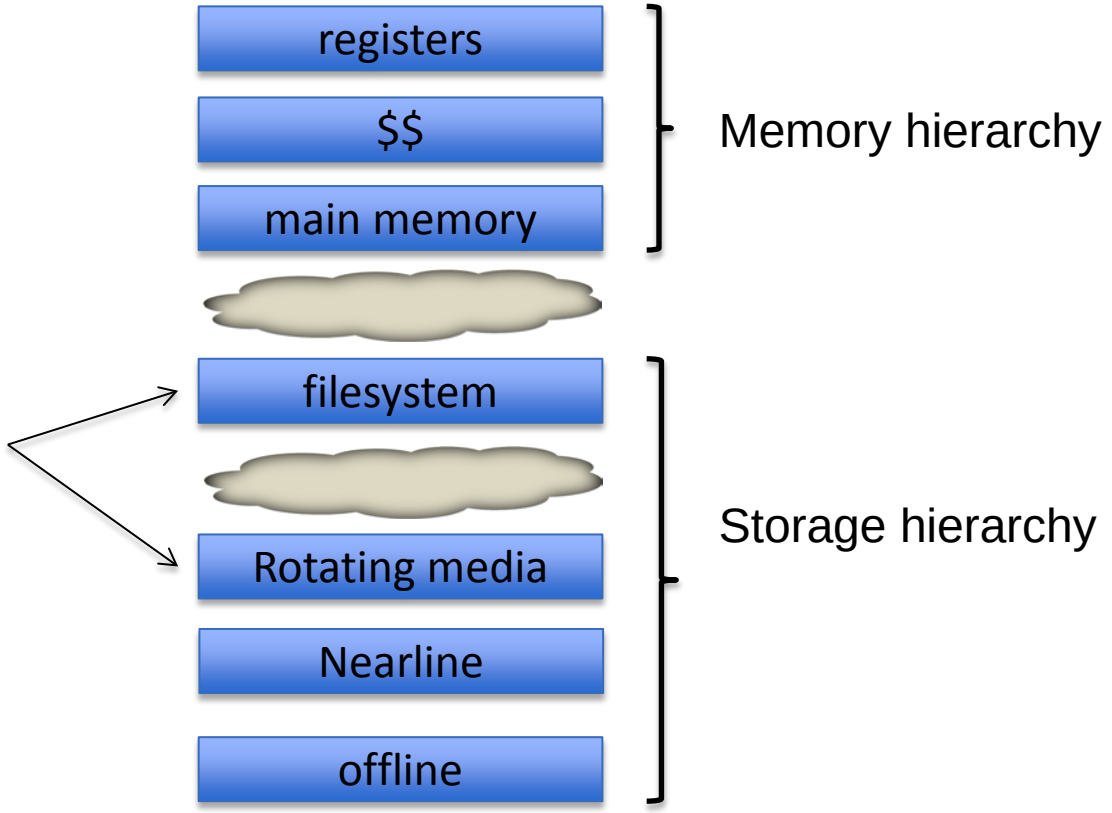
Memory hierarchy

Classical OFS assumes the existence of a traditional memory hierarchy



Memory hierarchy

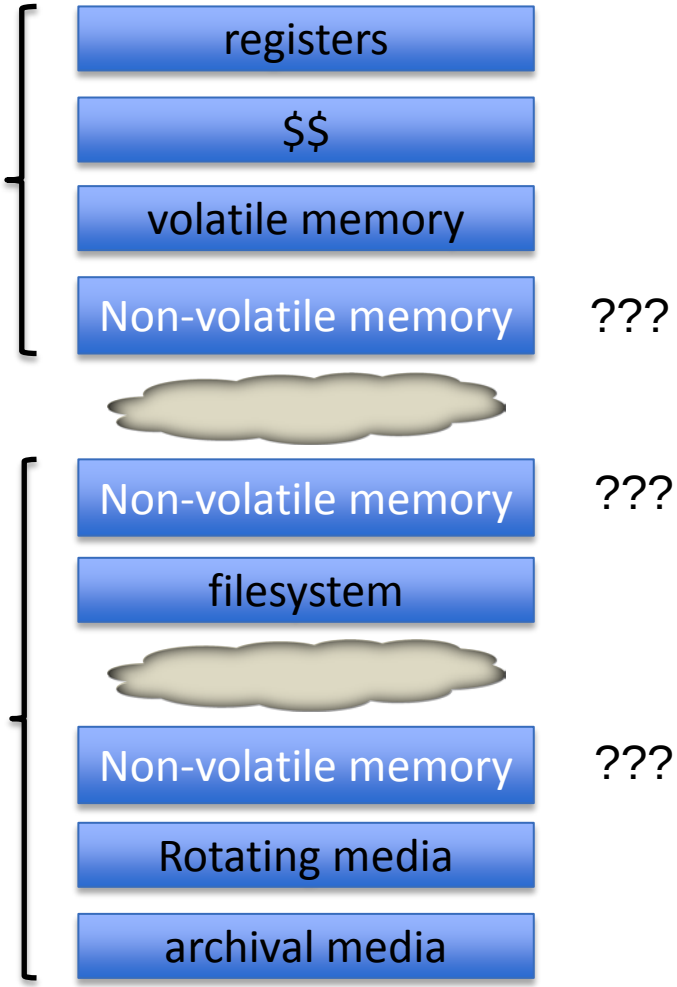
Classical OFS assumes the existence of a traditional memory hierarchy



Data may be buffered here in volatile memory.

Cost of this memory is on par with the cost of main memory, i.e. expensive

Re-factoring the hierarchy



Low cost non-volatile memory may end up forcing a re-factoring of the hierarchy

- A re-factored memory hierarchy?
- A new view to the file system?
- A new I/O protocol?

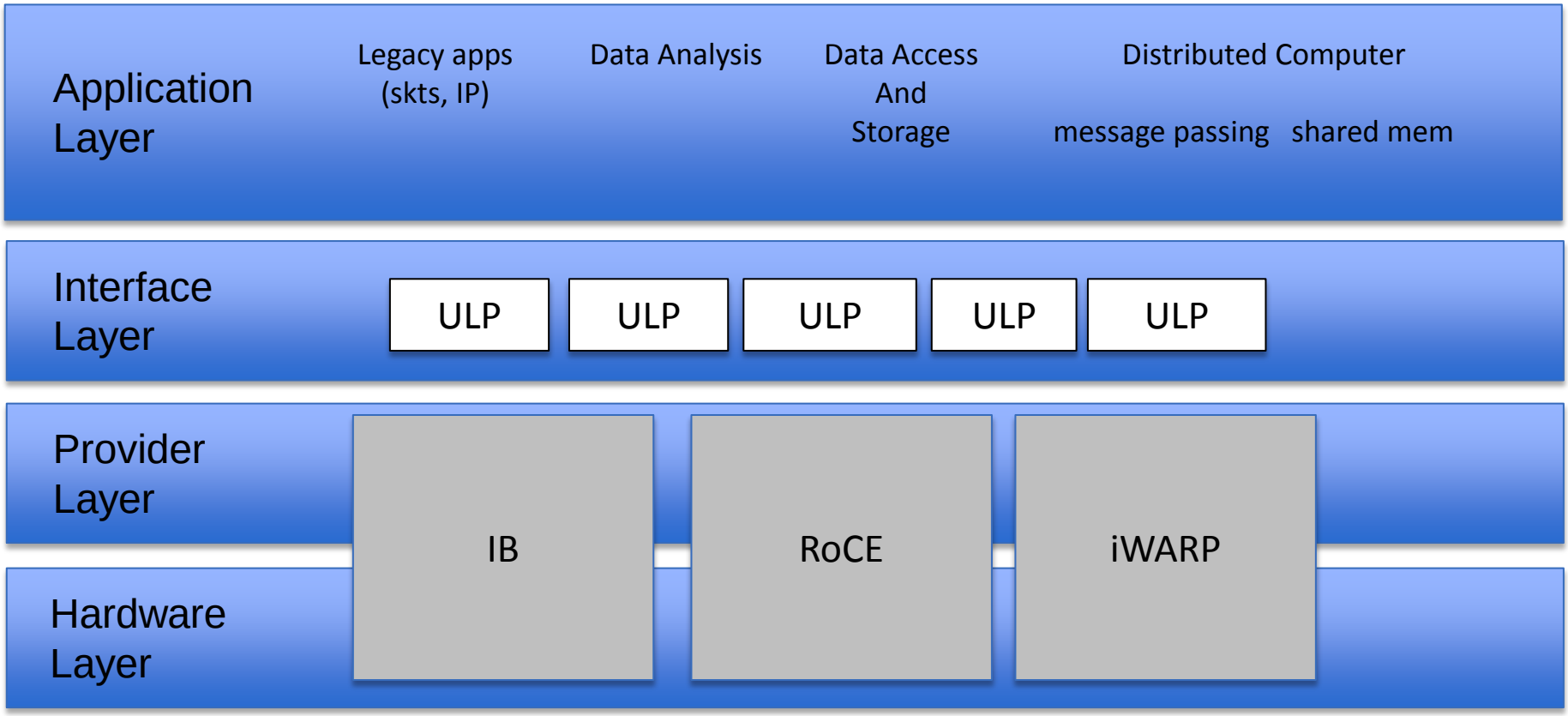
Challenges ahead

1. **Scalability**:
   - Programming model support
   - Inherent network scalability

2. **Technology**:
   - SSDs
   - Multicore
   - Heterogenous processing

3. **Usages**:
   - Big Data – analysis of unstructured data
   - Cloud – user access to his data
   - Storage at a distance – tools for collaboration among distributed teams

Fair game


Application Layer
Legacy apps (skts, IP)
Data Analysis
Data Access And Storage
Distributed Computer
message passing
shared mem
Interface Layer
ULP
ULP
ULP
ULP
ULP
Provider Layer
IB
RoCE
iWARP
Hardware Layer

Capturing results

“Figure out how to keep the state of the art in I/O moving forward”

Challenges	**The Ask**	**Who is the ‘Askee’?**
Scalability	(What needs to change in the area of I/O to expand scalability?)	??
Technology	(How does changing technology impact I/O?)	??
Usage	(How do changing usage models impact I/O requirements?)	??

Our challenge is to fill in this table

Call to Action

During the course of the next two and a half days there are session devoted to:

- Scalability
- Usages in HPC, the Enterprise, Cloud & Big Data.
- Technology

Think about how each of these impacts both the I/O software stack (e.g. OFS) and the underlying network (e.g. IB/RoCE/iWARP)

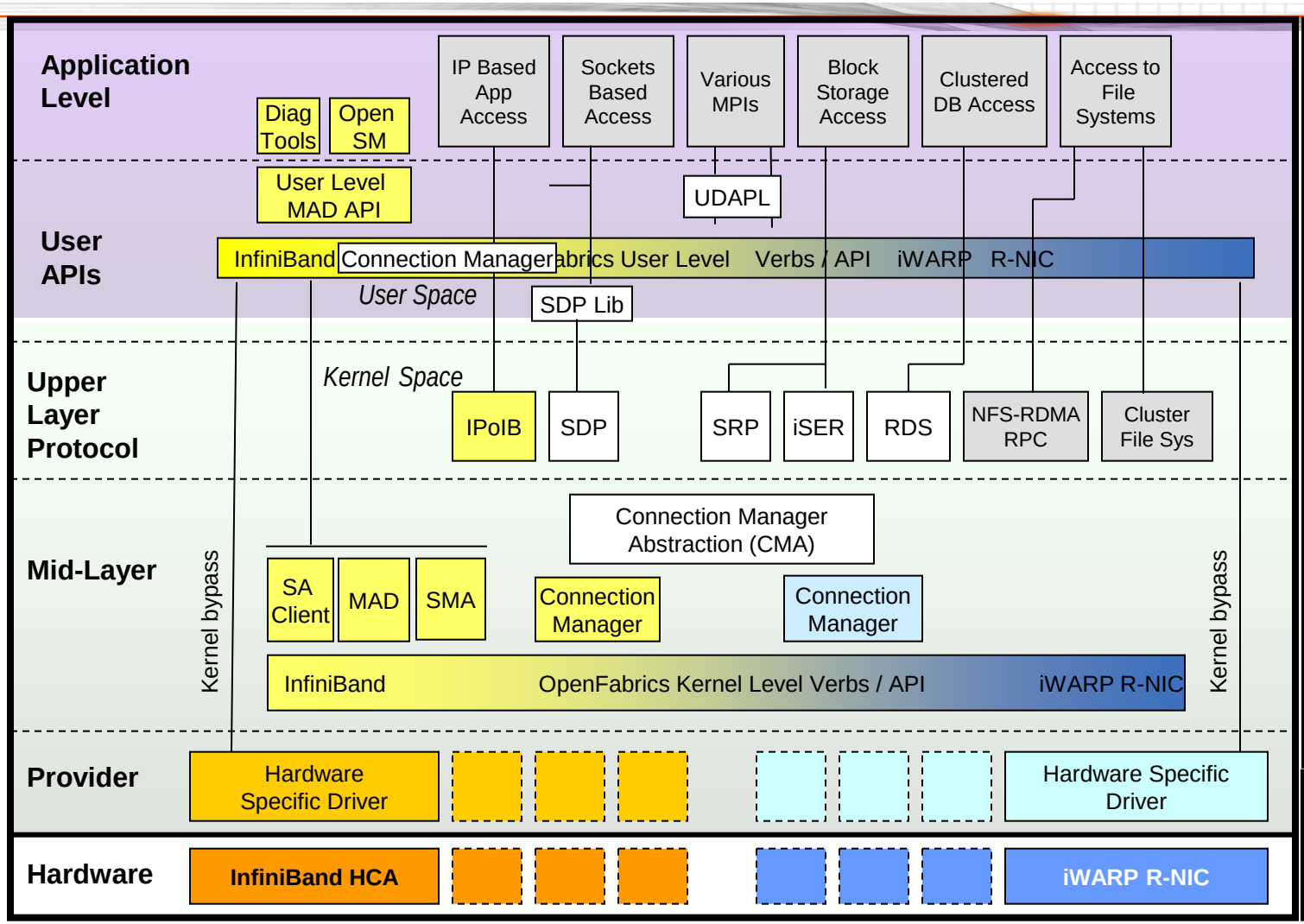
Think about improving the verbs model and ask where RDMA makes sense, and where it may not

On Wednesday, attempt to integrate all this into a course forward

See you on Wednesday

OPENFABRICS ALLIANCE

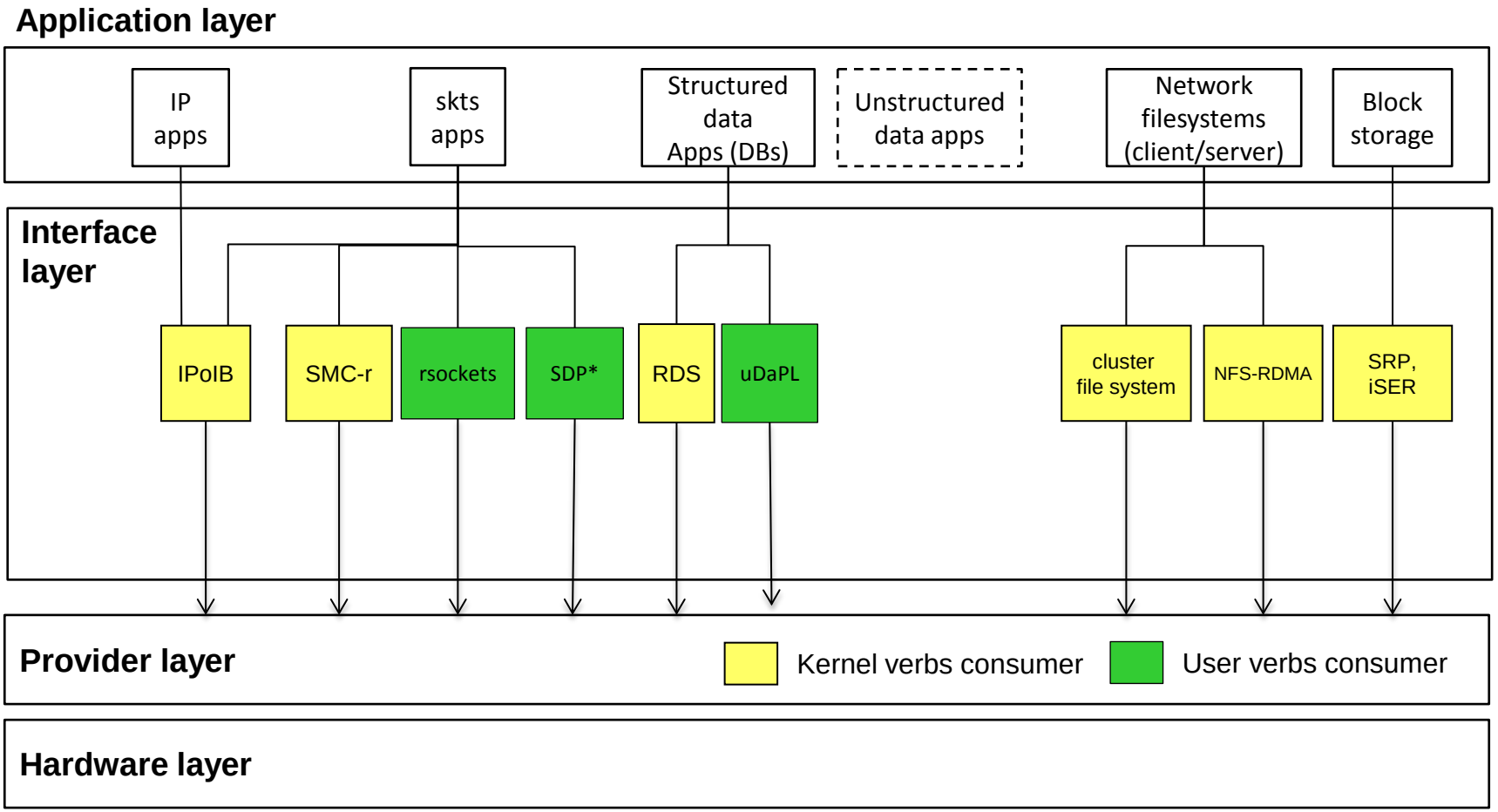
OFS Architecture Today



SA	Subnet Administrator
MAD	Management Datagram
SMA	Subnet Manager Agent
PMA	Performance Manager Agent
IPoIB	IP over InfiniBand
SDP	Sockets Direct Protocol
SRP	SCSI RDMA Protocol (Initiator)
iSER	iSCSI RDMA Protocol (Initiator)
RDS	Reliable Datagram Service
UDAPL	User Direct Access Programming Lib
HCA	Host Channel Adapter
R-NIC	RDMA NIC

Key: Common; InfiniBand; iWARP; Apps & Access Methods for using OF Stack

Application support



Distributed computing