Argonne NATIONAL LABORATORY

... for a brighter future

IOFSL

Scalable HEC I/O Forwarding Layer

Rob Ross, Pete Beckman, Dries Kimpe, Kamil Iskra (Argonne)

James Nunez, John Bent, Gary Grider, Sean Blanchard, Latchesar Ionkov, Hugh Greenberg (Los Alamos)

Steve Poole, Terry Jones (Oak Ridge)

Lee Ward (Sandia)

iskra@mcs.anl.gov

U.S. Department of Energy

UChicago Argonne LLC

Office of Science
U.S. DEPARTMENT OF ENERGY

A U.S. Department of Energy laboratory managed by UChicago Argonne, LLC

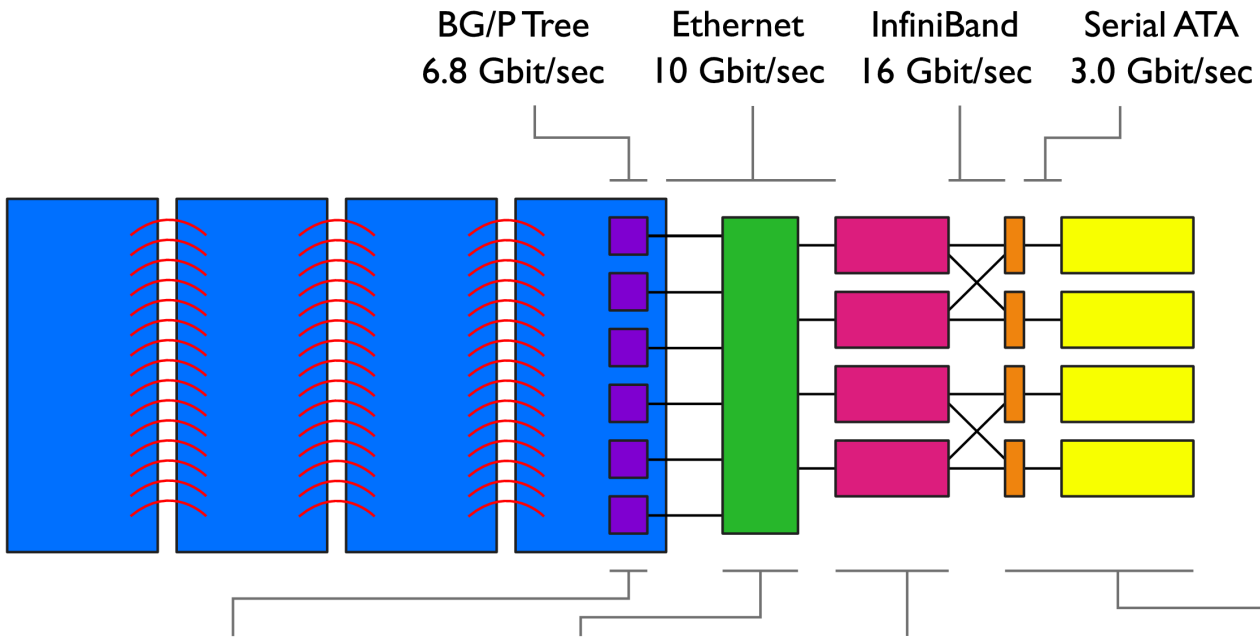
Contents

Motivation

Research into I/O infrastructure for petascale architectures:

- Today: 100K nodes (LLNL BG/L), 300K cores (Juelich BG/P)
  - will filesystems be able to handle an order of magnitude more?
- Argonne's 557 TF Blue Gene/P (Intrepid):
  - 20% of the money spent on I/O
  - full memory dump takes over 30 minutes
- I/O quickly becoming *the* bottleneck

Hardware Overview



Gateway nodes run parallel file system client software and forward I/O operations from HPC clients.

640 Quad core PowerPC 450 nodes with 2 Gbytes of RAM each

Commodity network primarily carries storage traffic.

900+ port 10 Gigabit Ethernet Myricom switch complex

Storage nodes run parallel file system software and manage incoming FS traffic from gateway nodes.

136 two dual core Opteron servers with 8 Gbytes of RAM each

Enterprise storage controllers and large racks of disks are connected via InfiniBand or Fibre Channel.

17 DataDirect S2A9900 controller pairs with 480 1 Tbyte drives and 8 InfiniBand ports per pair

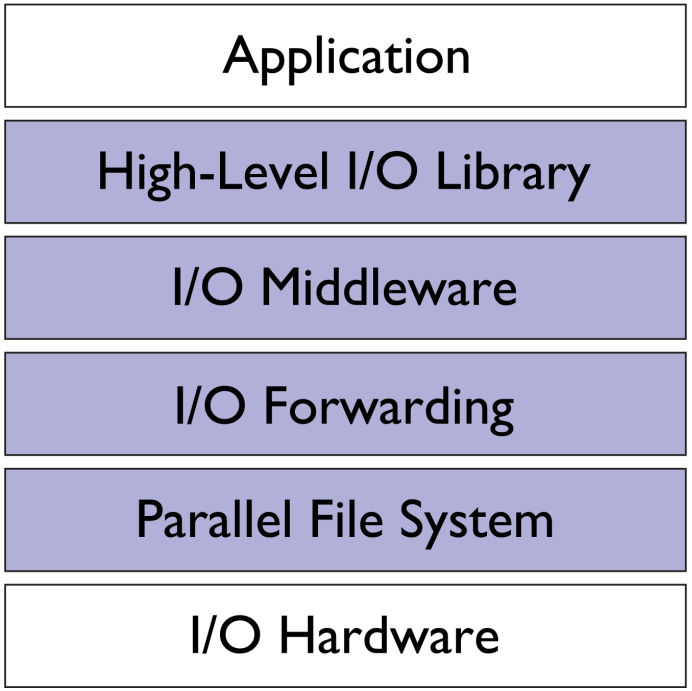
Software Overview

High-Level I/O Library
maps application abstractions onto storage abstractions and provides data portability.

HDF5, Parallel netCDF, ADIOS

I/O Forwarding
bridges between app. tasks and storage system and provides aggregation for uncoordinated I/O.

IBM ciod



I/O Middleware
organizes accesses from many processes, especially those using collective I/O.

MPI-IO

Parallel File System
maintains logical space and provides efficient access to data.

PVFS, PanFS, GPFS, Lustre

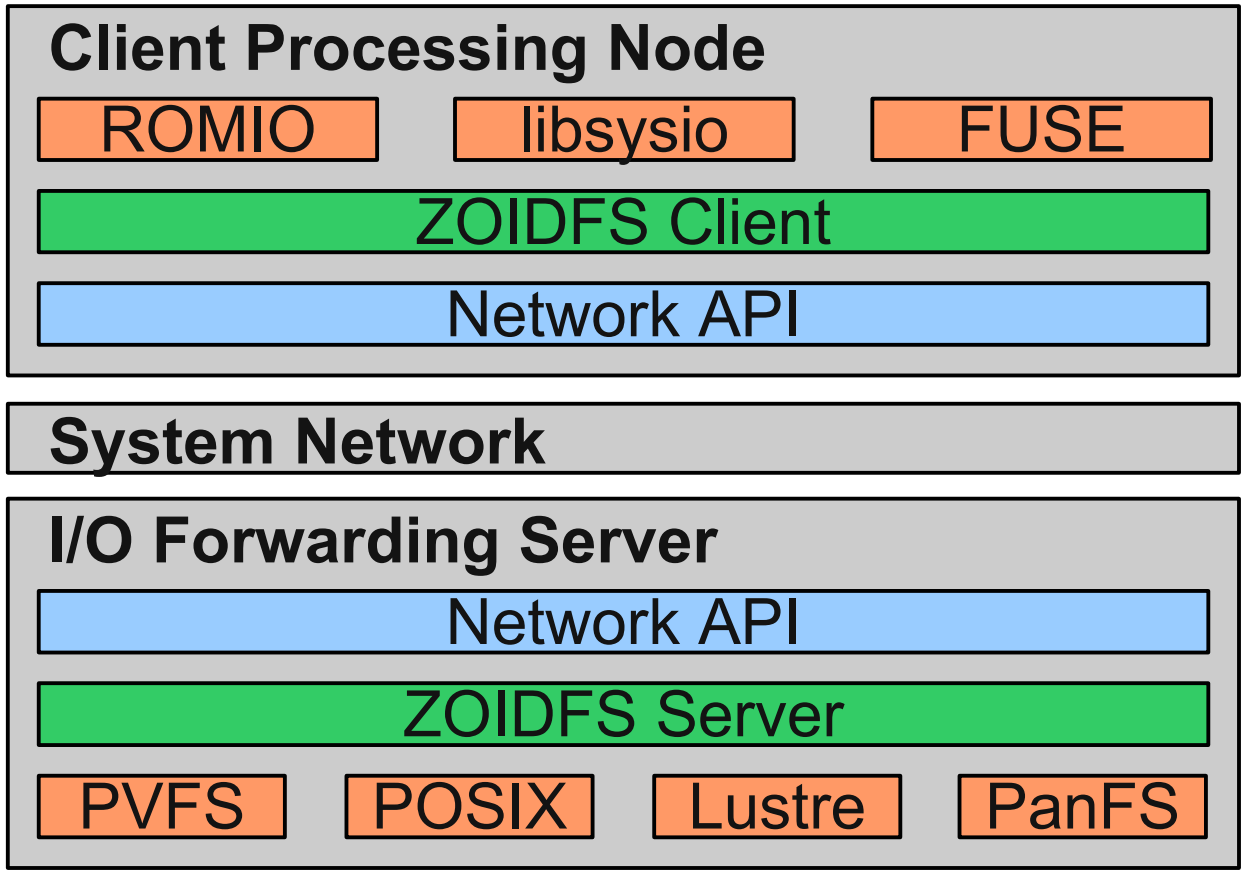
- We need to make I/O software as efficient as possible

Mission

Design, build, and distribute a scalable, unified high-end computing I/O forwarding software layer that would be adopted and supported by DOE Office of Science and NNSA.

- Reduce the number of file system operations/clients that the parallel file system sees
- Provide function shipping at the file system interface level
- Offload file system functions from simple or full OS client processes to a variety of targets
- Support multiple parallel file system solutions and networks
- Integrate with MPI-IO and any hardware features designed to support efficient parallel I/O

I/O Forwarding Framework



ZOIDFS Protocol

- Stateless
- NFSv3-like protocol
- opaque, 32-byte **`zoidfs_handle_t`** (no file descriptors)
- **`zoidfs_lookup`**, **`zoidfs_create`** (no open, close)
- **`zoidfs_getattr`** retrieves only the requested attributes
- **`zoidfs_readdir`** can do **`getattr`** if requested
- Maximally flexible **`zoidfs_read`**, **`zoidfs_write`**:

```
int zoidfs_read(const zoidfs_handle_t *handle,
                size_t mem_count,
                void *mem_starts[],
                const size_t mem_sizes[],
                size_t file_count,
                const uint64_t file_starts[],
                uint64_t file_sizes[]);
```

BMI

- Buffered Message Interface
- Designed for PVFS2
- Asynchronous
- Thread-safe
- Support for multiple networks:
  - TCP
  - IB
  - GM, MX
  - Portals
  - (need to write one for Blue Gene)
- XDR-encoded metadata between clients and servers
  - data payload sent using expected messages

ZOIDFS Server

- Two server designs (basic and advanced)
- Multi-threaded or state machine
- “native” PVFS and POSIX drivers
- Will also use libsysio as file system abstraction layer
- Planning to leverage a cooperative caching layer from a related NSF project
- Planning to use pipelining for large requests
  - instead of fragmenting of requests on the client side
  - requires careful buffer management

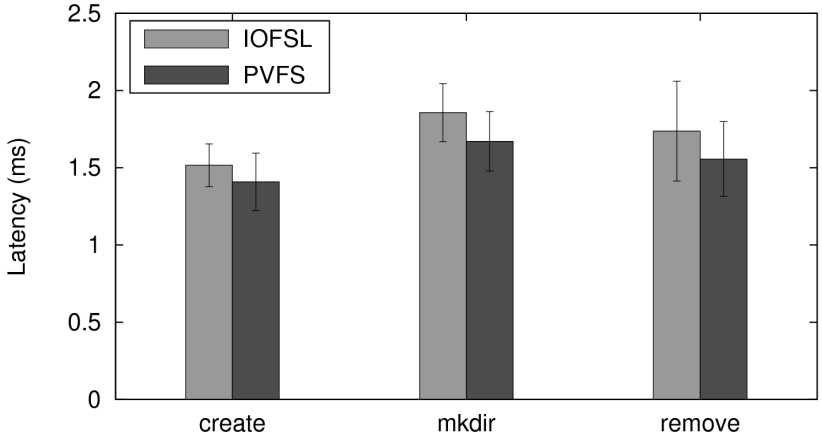
POSIX Support

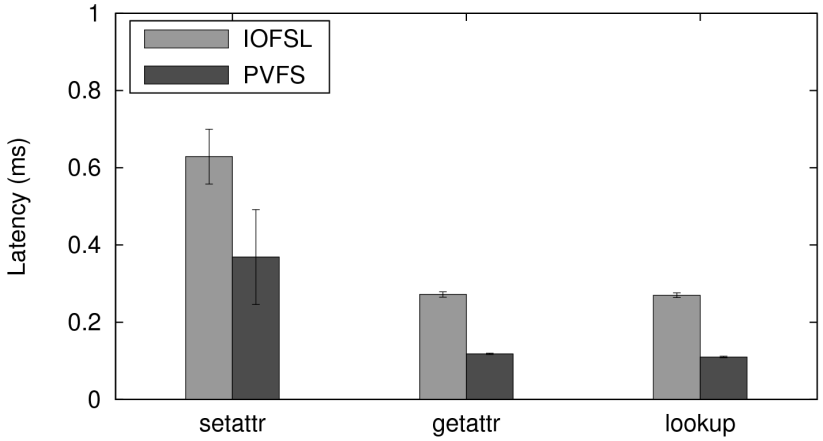
- Client-side:
  - FUSE (works, but performance not explored)
  - SYSIO (still to be implemented)
- Server-side:
  - SYSIO (still to be integrated) has a “native” POSIX driver
  - Custom ZOIDFS POSIX driver
  - How to translate between file handles and file descriptors?
    - *File handles are stateless, persistent, and global*
    - *Globally accessible database?*
    - *ESTALE for unknown handles forces a re-lookup*

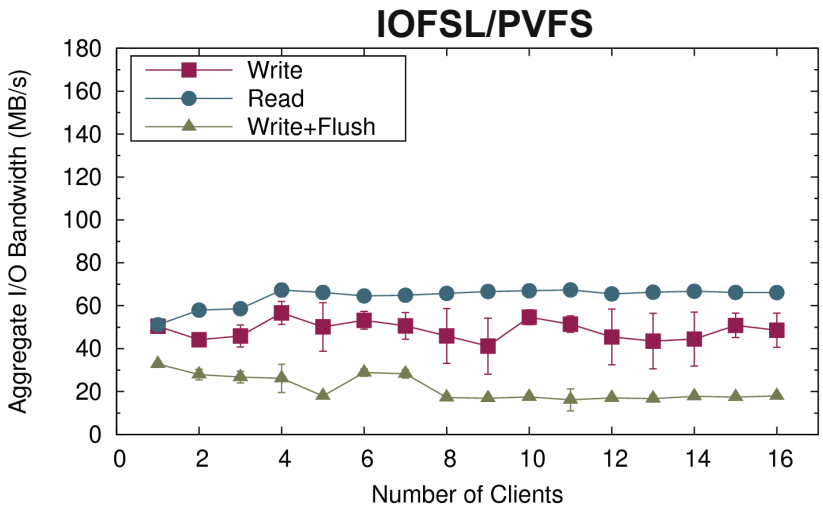
Security

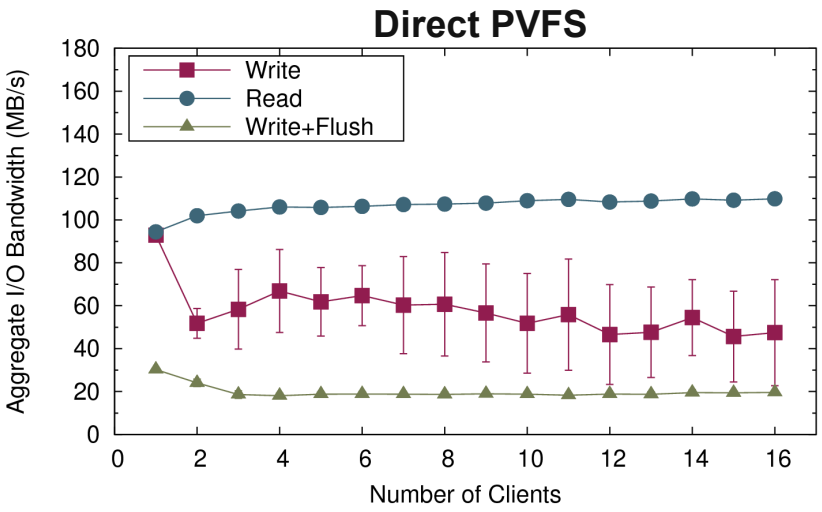
- NNSA labs care...
- Concerns:
  - authentication
  - authorization
- I/O forwarding servers might need to forward data of multiple users
- Add credentials to requests
- We will use POSIX test suite from The Open Group, which includes a security module

Early Measurements









- On this platform, IOFSL just introduces overhead

Current Status

- Client:
  - FUSE client implemented
  - Basic ROMIO driver implemented
- Networking:
  - BMI extracted from PVFS
  - ZOIDFS over BMI implemented
- Server:
  - Several server designs explored
  - libsysio file handle and credentials interfaces implemented
  - ZOIDFS to PVFS and POSIX drivers implemented

Future

- Integrate libsysio on both client and server
- Pipelining
- Test on Cray XT
- Support for Blue Gene
- Cooperative caching between servers
- Security

- http://www.iofsl.org/ (mostly a placeholder for now)

(if you see this slide, then I must have pressed End instead of PgDn)