IOFSL – I/O Forwarding Scalability Layer

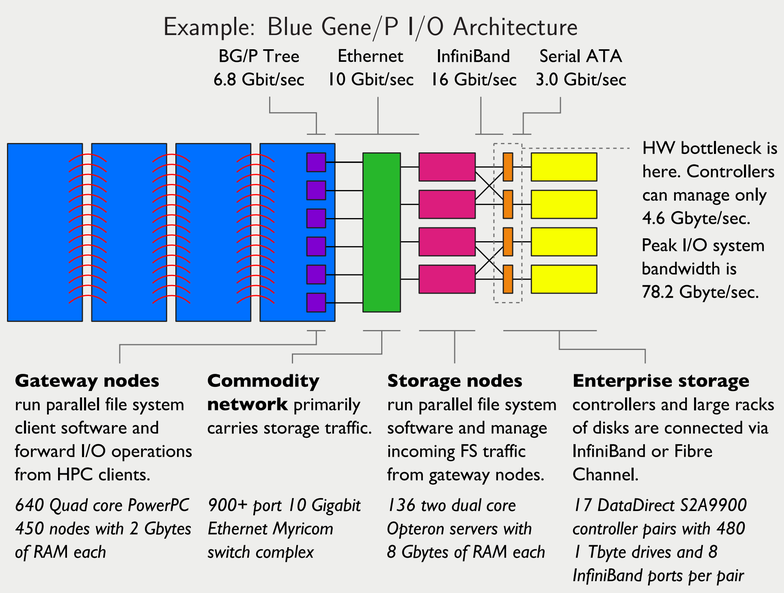
R. D'Amore, P. Beckman, J. Bent, J. Cope, G. Grider, K. Iskra, T. Jones, **D. Kimpe**, S. Poole, J. Nunez, K. Ohta, C.M. Patrick, R. Ross, L. Ward, B. Welton

Argonne National Laboratory, Los Alamos National Laboratory, Oak Ridge National Laboratory, Sandia National Laboratories, University of Chicago, University of Tokyo

Introduction

Modern massively parallel systems exhibit unique I/O architectures and I/O requirements. For example, compute nodes might not have direct outside access or might be running microkernels incapable of fully supporting all I/O functionality. To handle this, *I/O forwarding* was introduced. The basic idea?

> *Instead of performing your own I/O, have it done by some other entity that might be better suited or located.*



Architectural diagram of the 557 TFlop IBM Blue Gene/P system at the Argonne Leadership Computing Facility.

IOFSL

IOFSL – I/O Forwarding Scalability Layer

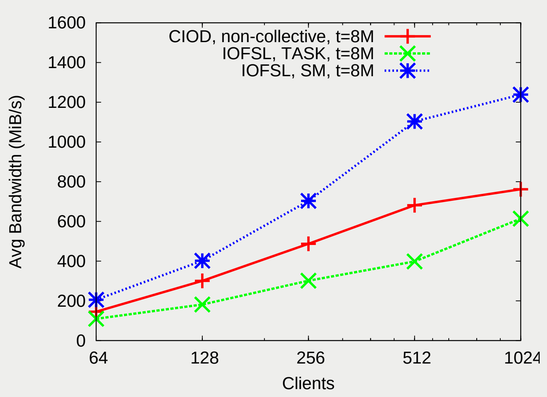
- Portable I/O Forwarding Implementation
- Production Quality – not just a research project
- Currently provides I/O forwarding on most leadership class machines

IOFSL provides features not commonly found in other forwarding implementations:

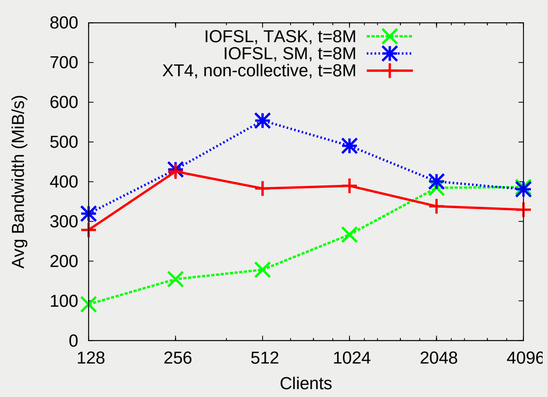
- Flexible extensible design: easy to adapt to new systems and to add support for new filesystems.
- Accelerates I/O research by providing a customizable, open source implementation.
- Manipulation of I/O requests instead of merely forwarding them; used to implement optimizations such as request merging and request scheduling.

Event-based Processing

> *Instead of having a thread working on the completion of a single forwarding request, threads work on a request until forced to wait, at which point the thread switches to another request. When a request becomes runnable again, the process repeats until the request is completed.*



IBM BG/P (@ANL)



Cray XT4 (@ORNL)

Advantages:

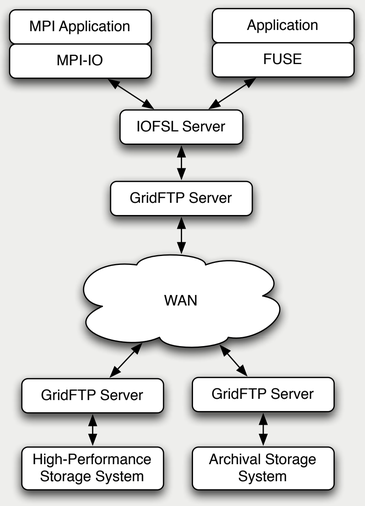
- Number of threads can be tuned independent of the number of requests
- Less memory and CPU overhead (context switches, thread stacks)

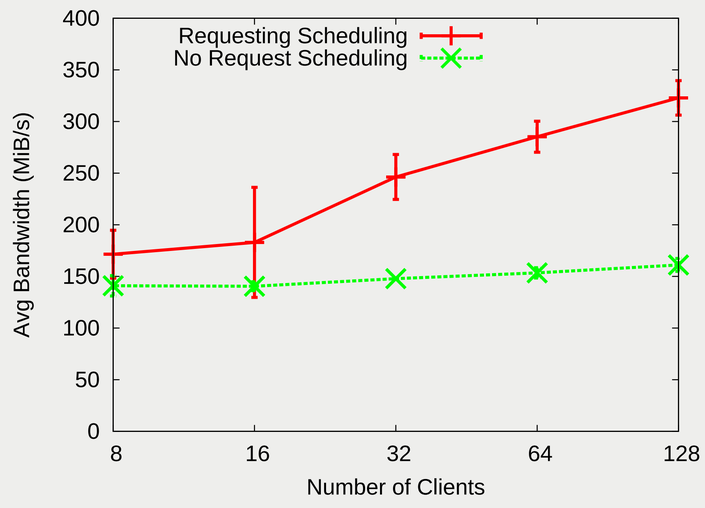
Disadvantages:

- Programming model is non-intuitive
- Debugging is hard
- Increased threshold to contribute code

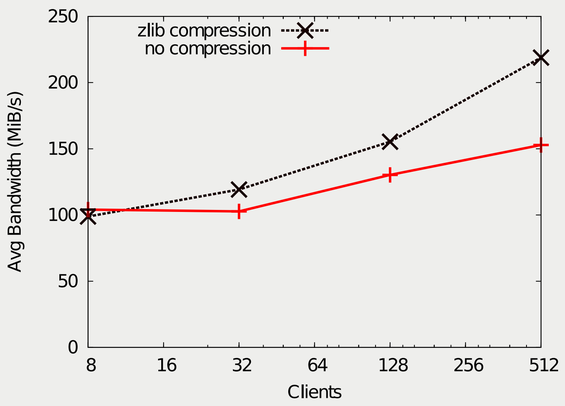
Request Scheduling and Merging

Example with our GridFTP backend driver:





Data Compression



Observation:

- Many clients talk to a single I/O forwarder
- Clients generally are idle until the forwarding server responds
- Compression is typically much slower than decompression

By using spare CPU time at the clients to compress data sent to the forwarding server (i.e. *writes*), the effective network bandwidth increases.

The forwarding server either decompresses before performing the I/O operation, or (ideally) indicates to the I/O system that the data is already compressed and passes it on unmodified.

Ongoing Work

We're working on improving IOFSL. Some of our current projects:

- server side compression
  Researching fast compression functions, trading CPU time for a lower compression ratio, in order to enable compression by the forwarding server.
- collaborative caching
  A jointly maintained cache between the forwarders enables new optimizations (such as request merging between forwarders) and improves the efficiency of existing optimizations.
- network protocol improvements
  Packing data and request information in the same message to reduce the number of exchanges between clients and forwarding servers. This is particulary important for applications making many small accesses (such as FUSE).
- Security Infrastructure
  Currently, an IOFSL forwarder executes requests for a single user. Our new security infrastructure will enable a single forwarder to securely serve multiple users.
- I/O Tracing and Visualization
  Integrating IOFSL with end-to-end I/O tracing and visualization tools, such as those developed for the NSF HECURA IOVIS / Jupiter project.

Contributing

We welcome all contributions and collaborations:

- IOFSL Project website: `http://www.iofsl.org/`
- IOFSL Wiki and Developer website: `http://trac.mcs.anl.gov/projects/iofsl/wiki`
- IOFSL Public Git repository: `http://git.mcs.anl.gov/iofsl`

Contact us at `io-fwd-devel@lists.mcs.anl.gov`