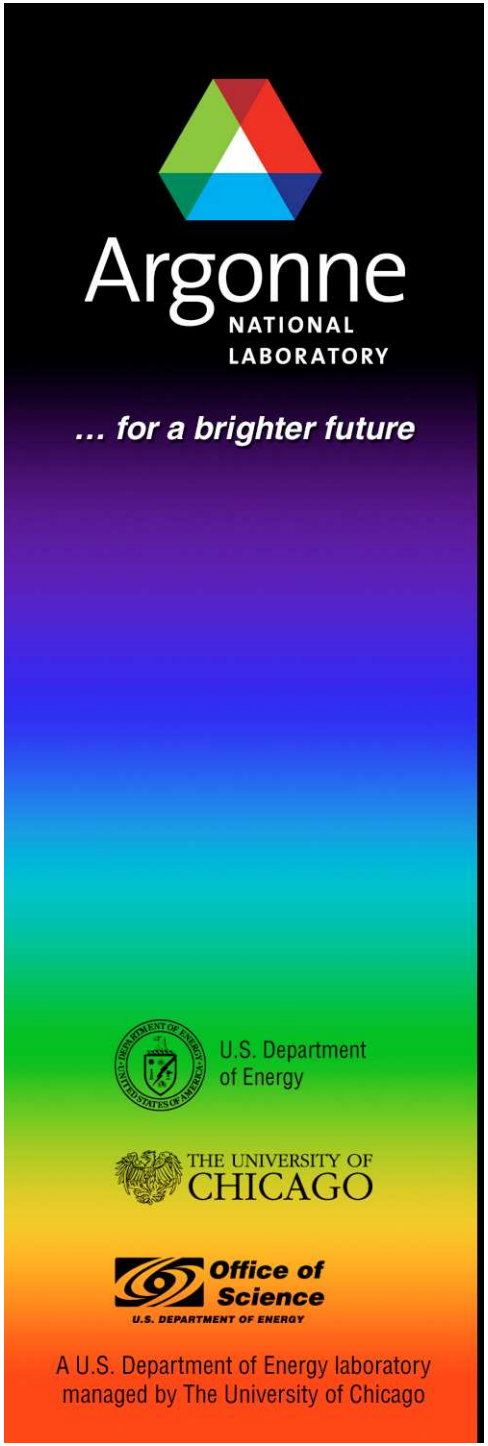


Can MPI Be Used for Persistent Services?

Rob Latham (*presenting)*

Robert Ross, Rajeev Thakur

Argonne National Laboratory

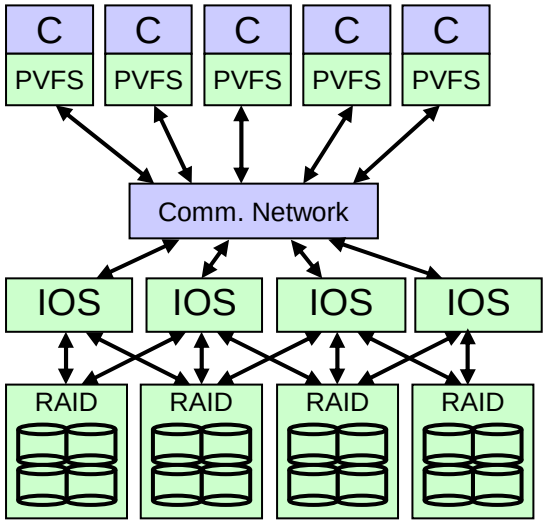
{robl,rross,thakur}@mcs.anl.gov

Challenges: What Makes System Services so Hard?

- System services provide long-running resources to applications
  - resource manager, job scheduler, file system
- Computational resources growing larger
- Software complexity increasing
- Application guys solved this
  - Gaussian, FFTW, GROMACS, pNetCDF.
- Language guys solved this
- Faster development -> more time for research:
- Consider specific example of PVFS

Case study: PVFS

- Communication performed over existing cluster network
  - TCP/IP, InfiniBand, Myrinet
- Servers store data in local file systems (e.g. ext3, XFS)
  - Local files store PVFS file stripes
  - Berkeley DB currently used for metadata (rather than files)
- Mixed kernel-space, user-space implementation
  - VFS module in kernel with user-space helper process
  - User-space servers, interface for kernel bypass
- Designed for hundreds of servers, tens of thousands of clients



PVFS configured with redundancy

PVFS Challenges

- Long development:
  - 5 years for recent redesign
  - Relatively small group of developers, but time typical for others (comparable with Luster, GPFS)
- Tricky code
  - Coordinating large numbers of processes (clients/servers, data/metadata)
  - Coordinating network and disk operations
- File system bugs particularly unpopular, unexpected

MPI Benefits

- Heterogenous communication
- Portable source
- Well-defined features and interfaces to those features
- Active research community
- Implementations likely to contain optimizations
- Debugged
  - Or at least, someone else's problem

Service Discovery: Current

- Clients need to know which machines host what resources
- Configuration items: one per service and one per client
- Change in system (planned, unplanned, or automated): need new config files

- Admins have tools to keep all in sync
  - But we're looking for a standard way to achieve this on all platforms
  - Ease of setup a big deal for PVFS end users

Service Discovery with MPI

- Utilize name publishing interface
  - Servers start up, call MPI_PUBLISH_NAME
  - Clients call MPI_LOOKUP_NAME
  - well-known service name
- Offload configuration to MPI implementation
  - MPI is likely already available
- Does demand quite a lot from interface
  - Standard lets implementations decide scope of key/value pairs

Portability in PVFS

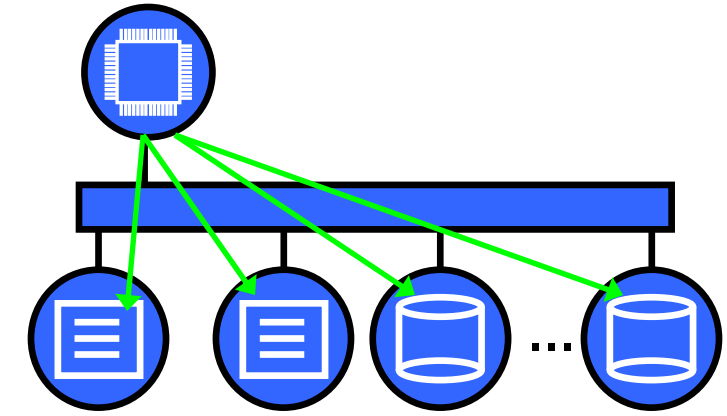
- System software particularly tied to underlying hardware
- Lots of different interconnects in HPC systems
  - TCP emulation usually available
    - *myri0, IPoIB*
  - Write an abstraction package
    - *Portals, BMI*
  - What about new interconnects?
    - *Vendor buy-in or do it yourself*
- Heterogeneity
  - We wrote our own request processor
- Operating systems

Portability with MPI

- Let MPI be the network abstraction layer
  - MPI_COMM_ACCEPT instead of interconnect-specific method
  - Or MPI_COMM_JOIN for bootstrapping over ubiquitous TCP
- Vendor buy-in: done (why are you in Bonn again?)
  - New interconnects and protocols get MPI implementations quickly: if not vendors then eager grad students
- MPI standard allows for heterogeneity

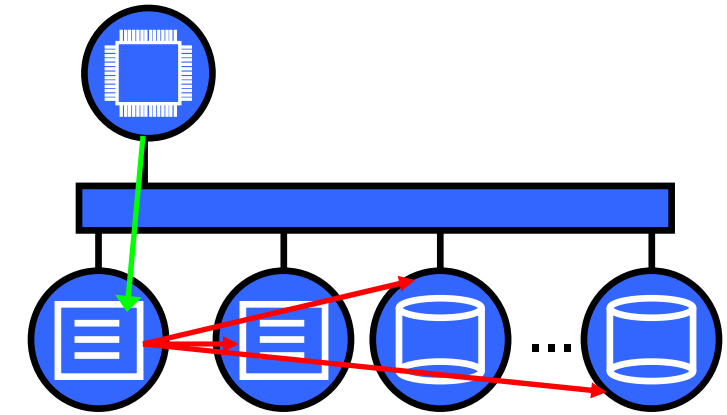
Collective and Aggregate operations in PVFS

- Certain PVFS API routines require several steps:
  - Create: datafile and metadata entries on servers
  - Remove: same, but reversed
  - Stat: needs partial size information from each server
- Call one function, but multiple messages on the network.



Aggregate and Collective operations with MPI

- Server-to-server communication
  - One all-encompassing communicator
  - Manage a collection of intercommunicators
- Single message from client
  - Could be sent to any server
- Servers sort out what has to happen
- Further optimization: efficient collectives
  - Runtime efficiency of O(log n) instead of O(n) to send n messages
  - Get this for free from many implementations
- Being able to pre-post non-blocking collectives would be helpful here



Challenges

- Custom MPI error handlers
- Fault tolerance
  - MPI faults
  - Hardware/software faults (everything else)
    - *MPI can help here (distributed checksum)*
- Can applications be made tolerant of failures?
  - PVFS servers and clients can be restarted w/o other nodes caring
  - How do we get the benefit of collective communication without introducing too much state?

Today's Implementations

Feature	MPICH2-1.0.3	OpenMPI-1.0.1	BGL MPI V1R1M1
Published name appears to other singleton processes	No	No	No
CONNECT ACCEPT works under singeton MPI_INIT	No	No	No
MPI datatype processing supports heterogeneous architectures	No	No	No

- Clearly, some work to be done

Alternate Approaches

- PVM:
  - heterogeneity, dynamic process management:
- CORBA:
  - not as widely deployed as other options
- Shorter answer
  - Don’t let me stop you from implementing something on top of other libraries

Next Steps

- Prototype PVFS in MPI
- Prototype MPI extensions
- Evangelize benefits to MPI implementers
- Come up with satisfying answers to failure modes

The End. Questions?

- Vielen Dank