IBM Research

On Developing BlueGene/L MPI-IO with High Performance

Hao Yu (yuh@us.ibm.com)

Feb. 23, 2005

On developing BG/L MPI-IO with high performance

- What is MPI-IO and BG/L MPI-IO?
- Status of BG/L MPI-IO
  - Functionalities
  - A preliminary performance
- On-going efforts
- Summary

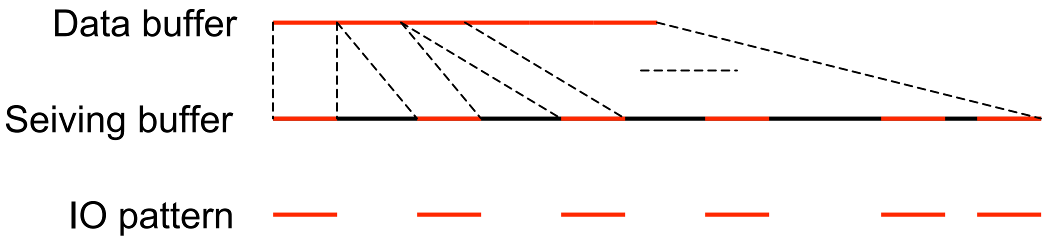
What is MPI-IO

- Parallel I/O interface specified in MPI-2 standard
  - Supports portable high performance file IO
  - File view functions and MPI datatypes allow user to express complex IO patterns
  - 3 orthogonal aspects of data access functions
    - File positioning: explicit offset, individual file pointer, shared file ptr;
    - Synchronism: blocking, non-blocking;
    - Coordination: non-collective (independent), collective
    - Among these, collective file accesses allow MPI-IO to optimize the interactions with storage devices.
  - File consistency: atomic/non-atomic access mode
  - File manipulation: open, pre-allocate, resize, etc.

Example #1: non-contiguous IO

```
int blocksize[4] = {2,2,2,2};
int indices[4] = {0,3,9,18};
char buf[8];

MPI_Type_indexed( 4, blocksize, indices, MPI_BYTE, filetype )
MPI_Type_commit( &filetype );

MPI_File_open(… &fhandle);
MPI_File_set_view( fhandle,offset,MPI_BYTE, filetype, "native", info);

MPI_File_read( fhandle, buf, 8, MPI_BYTE, &status );
```

MPI-IO may optimize the non-contiguous read by data sieving or using GPFS prefetch hints.



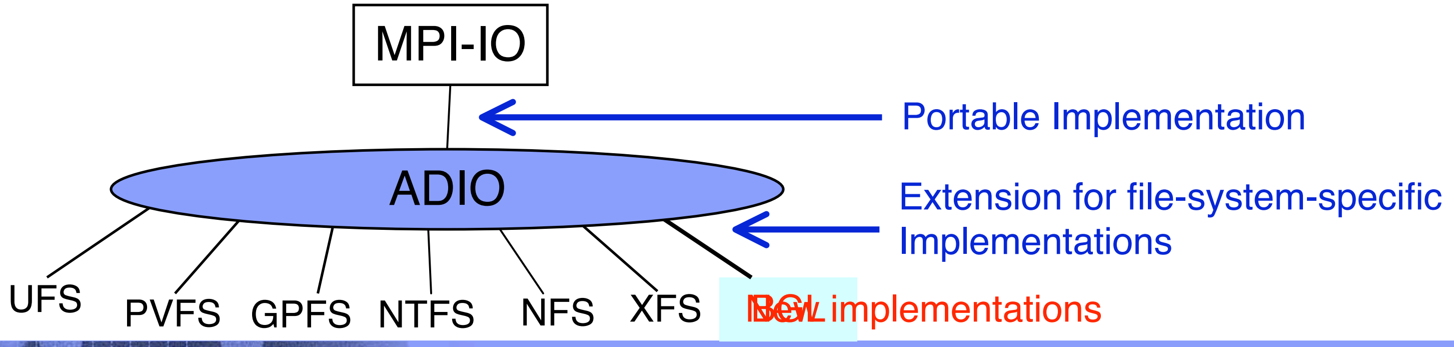
Example #2: collective non-contiguous IO

```
int blocksize[4] = {2,2,2,2};
char buf[8];

if      (myrank == 0) indices[4] = {0,4,8,12};
else if (myrank == 1) indices[4] = {2,6,10,14};

MPI_Type_indexed( n, blocksize, indices, MPI_BYTE, filetype )
MPI_Type_commit( &filetype );

MPI_File_open(… &fhandle);
MPI_File_set_view( fhandle,offset,MPI_BYTE, filetype, "native", info);

/* read from 4 disjoint regions from file */
MPI_File_read_all( fhandle, buf, 8, MPI_BYTE, &status );
```

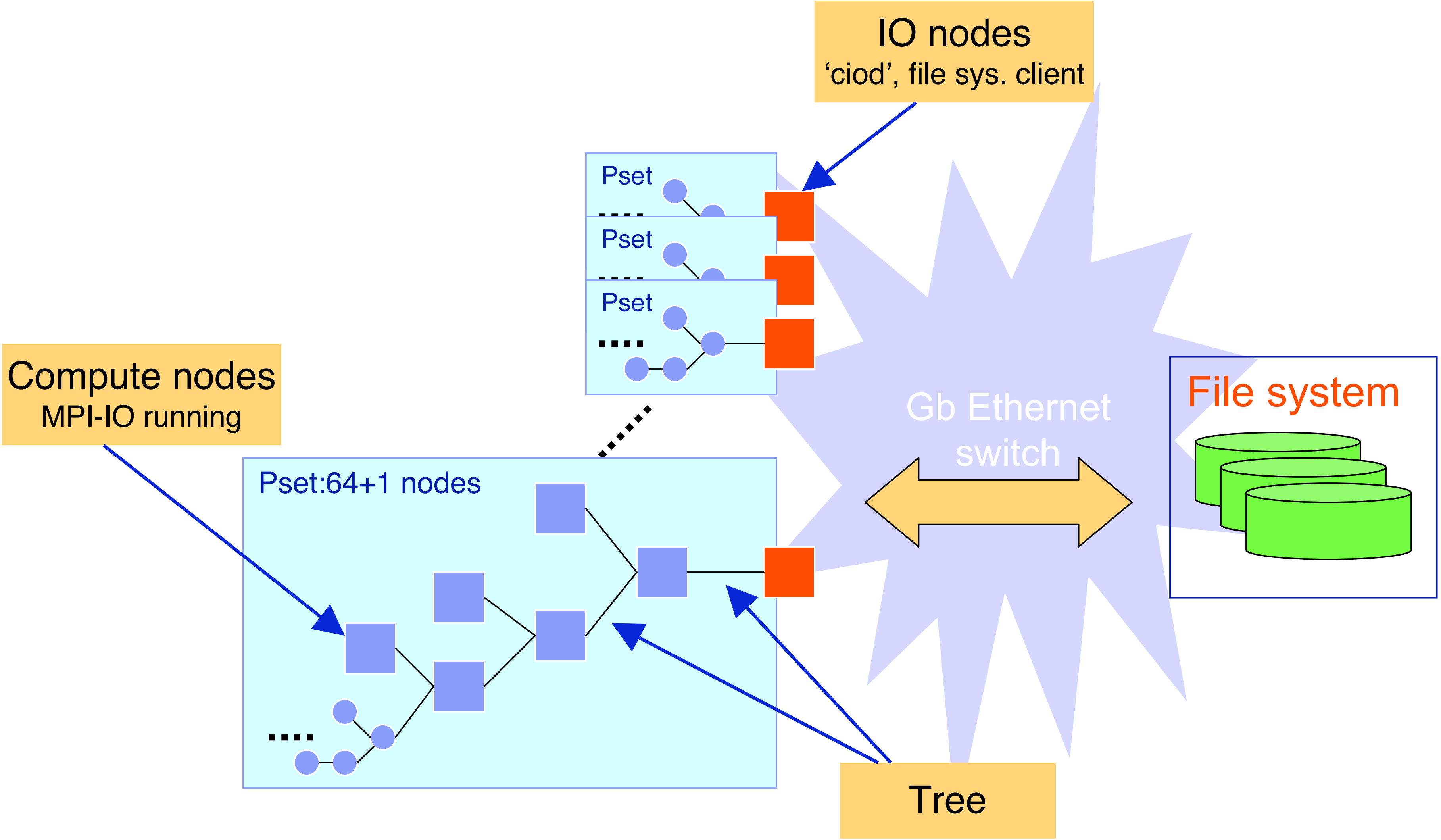
MPI-IO may aggregate the read requests from 2 processes and issues contiguous IO operations to the file system.

What is BlueGene/L MPI-IO

- BlueGene/L MPI-IO started as a direct port of Argonne National Lab’s MPI-IO implementation, ROMIO.
- What is ROMIO
  - Aportable MPI-IO implementation
  - Its portability is achieved mainly because that it was built on top of MPI and an abstract-device interface called ADIO
  - Emphasizing on optimizing collective IO and non-contiguous IO
- BG/L MPI-IO took ROMIO implementation for NFS



BG/L I/O subsystem



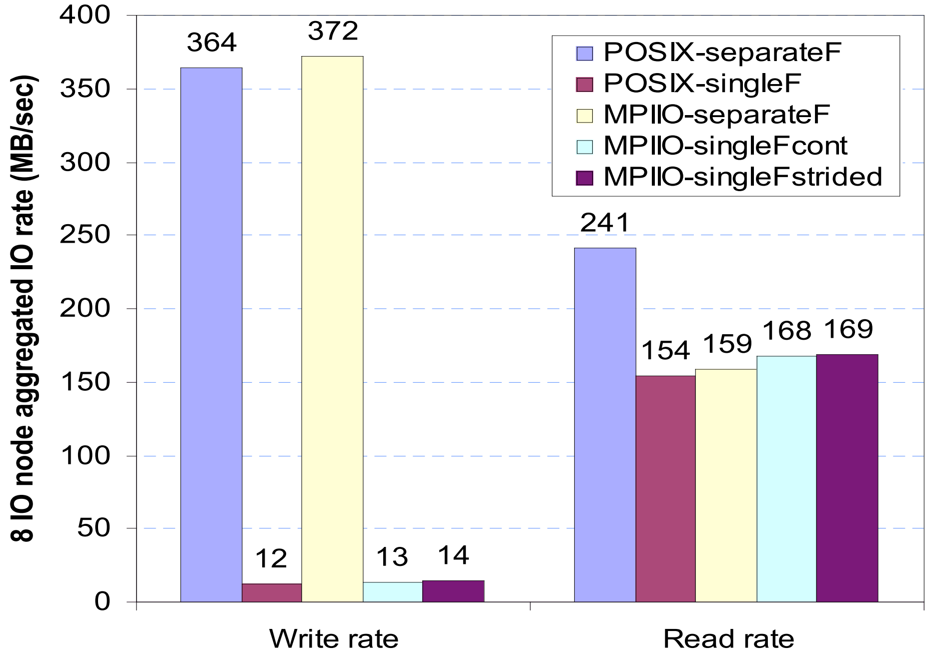
Status of BG/L MPI-IO

- Ported most MPI-IO functionalities
  - MPI-IO functionalities are tested for ROMIO tests, MPICH2 IO tests, LLNL MPIO-test, parallel HDF5, PnetCDF. FLASH_bench
  - Exchanged many emails with ROMIO team at ANL.
  - Enhanced BG/L with fcntl() file locking function (can be easily extended for supporting MPI-IO atomic access mode for other file systems)
- Started work on performance optimization for BG/L MPI-IO
  - Optimization for collective IO
  - GPFS specific developments
  - Collaborations:
    - ANL ROMIO team (optimization for PVFS2)
    - Northwestern U: Choudhary, Coloma, Ching

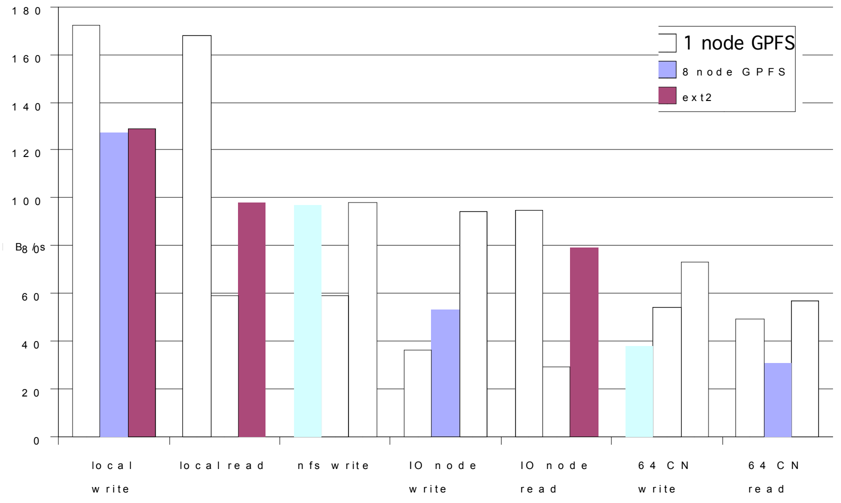
Preliminary MPI-IO performance…

IOR 2.8.1, 8 IO nodes, NFS mount, 8-node GPFS, 1.7TB

IO-rich conf. - 64 CN, 8 ION, 32 GB file, 1MB/op



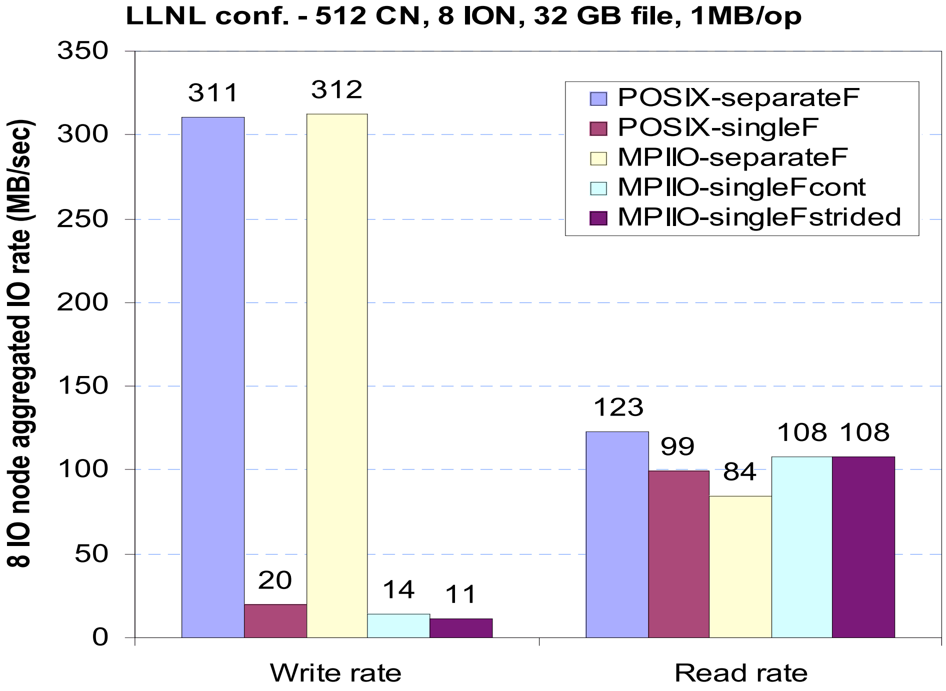
32 GB, 4MB, 32k wsize, 64 CN, 9000M



- Reason for the poor single file writing performance: we did not specify “noac” for NFS mount. Every wsize NFS write invokes metadata lock.
- Reason for the 160MBps read performance is not clear.

… Preliminary MPI-IO performance

IOR 2.8.1, 8 IO nodes, NFS mount, 8-node GPFS, 1.7TB

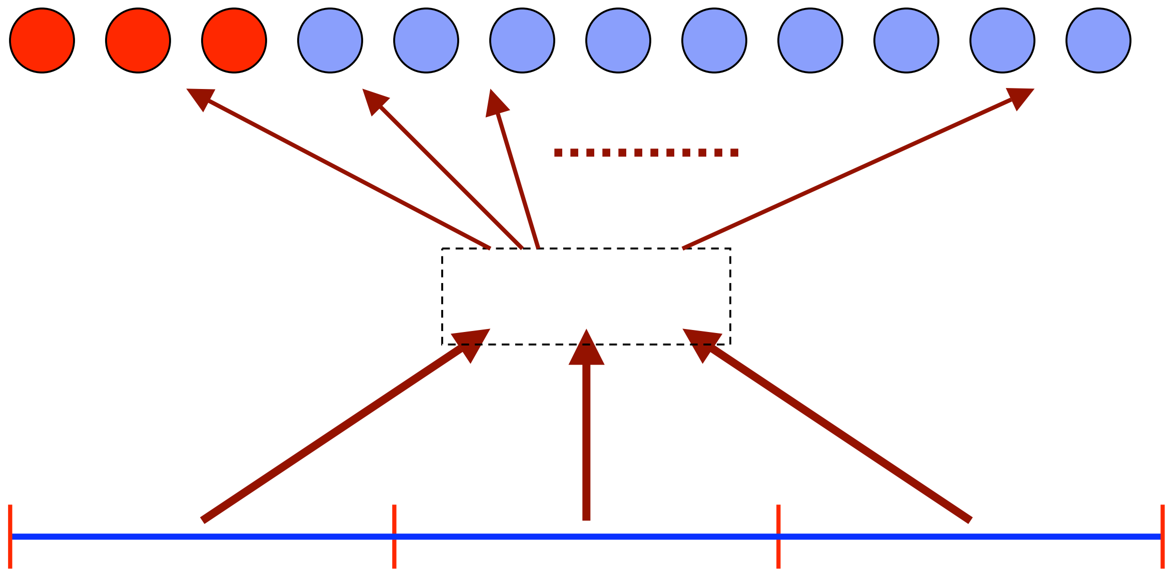


- Reason for the 100 MBps read rate is one-day old driver.
- MPI-IO keeps up with the POSIX-IO perf. for separate file writing and reading.

Collective IO

- ROMIO emphasizes on collective IO optimization.
- 2-phase framework
  - Phase 1 aggregates, distributes, and/or redirects IO requests onto a list of IO aggregators by building the communication graph (execution schedule) of the IO requesters and the IO aggregators.
  - Phase 2 carries out the schedule (including data shipping among MPI processes and IO operations from IO aggregators)
  - In this framework, MPI-IO can perform optimizations such as
    - aggregate fine-grain IO requests;
    - balance, distribute IO load among MPI processes.
- For BG/L, we recommend use of collective IO
  - BG/L does not have means to optimize non-collective MPI-IO ops
    - IO node should not be loaded
    - BG/L MPI does not have one-sided comm. Mechanism
    - Look-aside will hurt MPI performance.

Depiction of ROMIO collective read



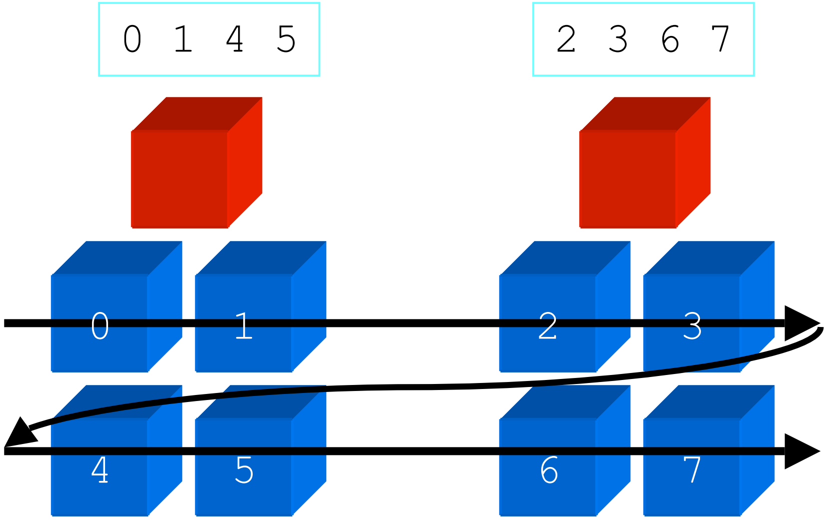
processes

aggregators

file domain

BG/L specific collective I/O optimizations…

- ROMIO only performs 2-phase IO for non-contiguous IO requests that are not overlapped across processes.
    - On BG/L, compute nodes in a Pset may not have contiguous rank
        - Contiguous and non-overlapped access pattern from application's view-point may become irregular on IO node.
    - We will apply 2-phase IO for contiguous collective IO



IO pattern from IO nodes

Default MPI ranking

… BG/L specific collective I/O optimizations

- ROMIO specifies IO aggregator via user provided hints containing a list of MPI processor names
  - On BG/L, ask a user for such a list will not work
    - Such a list for 1000 processors will take 72KB
    - It is non-trivial for user to generate such a list that is aware of BG/L Pset structures
  - We will provide a hint (bgl_cb_nodes) specifying #IO aggregators in each Pset.

GPFS specific developments

- GPFS file access mode:
  - Default mode: distributed GPFS block level locks are used to provide file consistency.
  - Data shipping mode: accessing a file block has to go through a pre-specified GPFS client node
    - User can distribute file across a set of GPFS client nodes following a cyclic pattern.
    - Need to ship Gpfs_fcntl() from CN to ION.

GPFS specific developments

- ROMIO assumes a regular file domain partition based on a run-time summary of the collective IO operation.
  - Because GPFS' file locking and file distribution are based on a fixed block-size, ROMIO's default file partition may introduce false sharing.
  - GPFS specific or more flexible file domain partitions is considered and corresponding hints shall be provided.
- GPFS only has atomic access mode
  - MPI-IO needs to support relatively efficient atomic access mode.
  - Due to limited power on IO node, such effort is considered in the framework of MPI collective IO.
- Collaborating with Northwestern on these optimizations.

Summary – BG/L MPI-IO is under development

- BG/L MPI-IO is started as a port of Argonne National Lab's MPI-IO implementation, ROMIO.
- Most BG/L MPI-IO operations are functional.
- From preliminary experiments, BG/L MPI-IO seems not introducing much overhead when comparing to POSIX IO.
- We are concentrating on optimizing BG/L MPI-IO for GPFS.
- Collective IO will be the most efficient way to use BG/L MPI-IO.
- Collaboration with ANL ROMIO team and Prof. Alok Choudhary's team at Northwestern U.

Team

- IBM BG/L I/O team: Chris, Parker, Engelsiepen, Volobuev
- IBM BG/L MPI team: Almasi
- Argonne Nation Lab ROMIO team: Ross, Thakur, Latham
- Northwestern Univ: Choudhary, Coloma, Ching
- IBM contact: Yu (yuh@us.ibm.com)