Reservation-based I/O Performance Guarantee for MPI-IO Applications using Shared Storage Systems

Yusuke Tanimura[1)], Rosa Filgueira[2)], Isao Kojima[1)] and Malcolm Atkinson[2)]

1) National Institute of Advanced Industrial Science and Technology (AIST), Japan, 2) The University of Edinburgh, UK

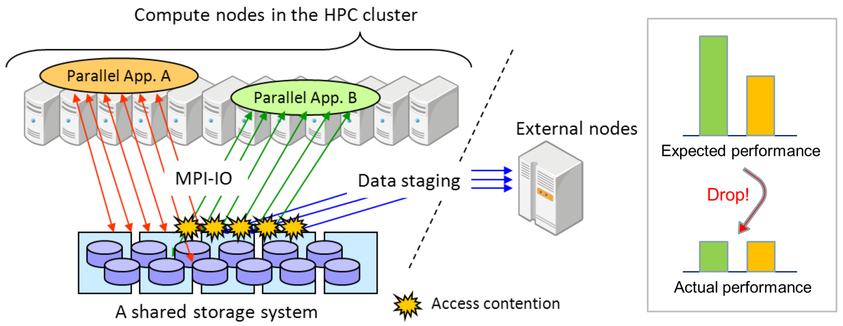
Background

There is a big problem in concurrent use of the parallel storage systems on HPC Clusters. This might spoil the performance improvement that might otherwise be obtained by optimizations of MPI-IO, such as data sieving, two-phase collective I/O and etc.

- MPI-IO is important for large data analysis.
  > Initial input, final output, and checkpointing
- Total throughput degradation and instability by various, internal and external interferences occur in HPC clusters. [J.Lofstead et al. in SC10]
- For example, access contention may happen,

a) when more than two parallel programs call MPI-IO.

b) when external access (for pre/post processing) is executed.



Advance Reservation Approach

Our approach to achieve performance guarantees is to allow users / applications to explicitly reserve I/O throughput of the storage system in advance, with start and end time of the access.

- Neither overprovitioning nor reactive QoS mechanisms
- SLO (Service Level Objectives) :
  > Read or write throughput (e.g., MB/sec) in a single access (open ~ close).
  - Striping access is automatically enabled if necessary.
  > Measurement granularities are user-defined.
- Integration with the batch scheduler

Users tell necessary I/O throughput (and job execution or I/O time) to a batch scheduler (BS) when they submit a job. Then BS reserves I/O throughput,

a) during entire execution of the job.

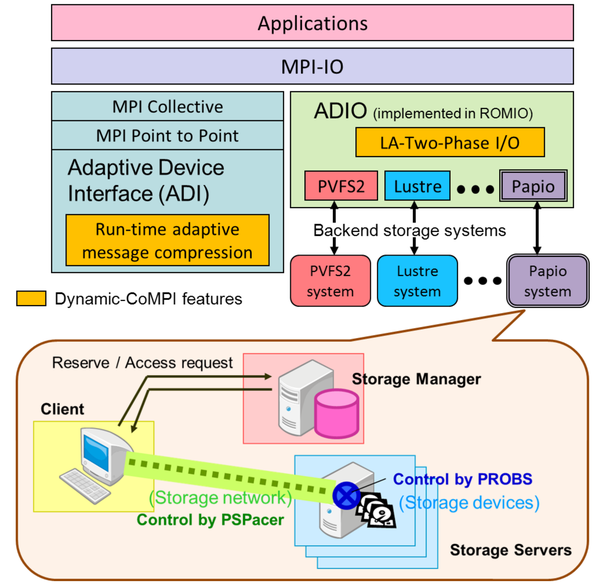
b) during specific I/O time.

We expect that iterated jobs and system-level checkpointing may have steady execution which allows users to estimate execution time of the job and its I/O time.

Design and Implementation

We have been developing a performance guarantee storage software called Papio. In order to examine an effect of the performance guarantee of MPI-IO, we developed the ADIO layer of Papio for Dynamic-CoMPI.

- Papio [Y.Tanimura et al. in Grid 2010]
  > Parallel I/O storage software with performance guarantee functionality based on advance reservation (Note that the reservation is mandatory for now.)
  - Assign available resources to the reserved access: fully occupied or shared
  - If shared, control I/O throughput of the storage network and I/O scheduling of disks
- Dynamic-CoMPI [R. Filgueira et al. in J. Supercomputing 2010]
  > Implement advance features based on MPICH2
  - Locality aware strategy for Two-Phase I/O: Optimized data aggregation into contiguous buffers, and sequential transfers into the file/storage system
  - Adaptive-CoMPI: Run-time adaptive message compression
- ADIO layer of Papio (ad_papio) in ROMIO
  > Support collective calls: MPI_File_write_all(), MPI_File_read_all()



Evaluation

The performance of Dynamic-CoMPI/Papio with reservation was compared to Dynamic-CoMPI/{PVFS2 or Lustre}, during concurrent access time.

- MPI-IO Test benchmark: We ran the benchmark and an additional workload (sequential read/write). Both access the storage system at the same time. – Fig.1
  > Without the reservation, the throughput dropped 3~25%.
- Application benchmark: We used the BISP3D application which is a 3-dimensional simulator of bipolar devices. Its MPI-IO write access to the storage system was conflict with the additional workload. – Fig. 2
  > Papio provided faster (11~24%) and stable throughput.

Experiment environment: 5 clients and 4 storage servers
- 4-core Opteron CPU, 8GB memory, OCZ VERTEX SSD, 1GbE network
- The MPI program was executed in parallel over 4 clients.

[Stripe-contiguous (SC)]
- Papio (with reservation)
- Lustre (with collective comm.)

[File-contiguous (FC)]
- PVFS2
- Lustre (without collective comm.)

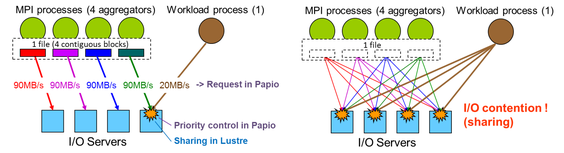
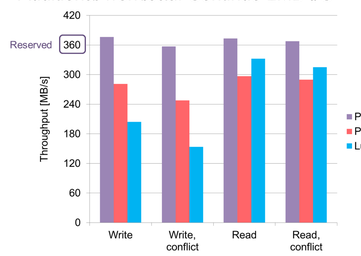


Fig.1: MPI-IO Test Result

MPI program: Continue 32MB I/O

Additional workload: Continue 1MB I/O

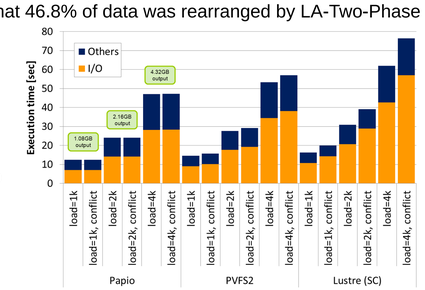


In the Papio case, each aggregate process requires (reserves) 90MB/s and another workload requires 20MB/s.

In the PVFS2 and Lustre cases, I/O throughput is shared by multiple accesses on the conflicted I/O servers.

Fig.2: BISP3D Result

Without the reservation, total time increased 6~27% (I/O time increased 8~40%), while Papio provided required throughput. Note that 46.8% of data was rearranged by LA-Two-Phase I/O.



Conclusion

- The performance guarantee achieved stable execution of MPI jobs during concurrent access time.
  > Without the reservation, I/O time increased 3~40% by conflicts.
  > Our approach is more effective for write than read.

Future work

- Further development of ad_papio – Support other collective calls and non-collective calls.
- Evaluation of the reservation-based job execution models using the integrated batch scheduler.