WIP-FAST 2008: View-based collective I/O for MPI-IO

Javier García Blas, Florin Isailă and J. Carretero
Computer Architecture Group, and Communications Systems
University Carlos III, Spain
{fjblas,florin,jcarrete}@arcos.inf.uc3m.es

Keywords: parallel file system, high performance computing, data staging

This report presents work in progress at a new file system-independent collective I/O optimization based on file views: view-based collective I/O. View-based collective I/O has been implemented inside ROMIO [1] implementation of MPI-IO standard. In the previous work [2] we have shown that view-based I/O outperforms the original two-phase collective I/O from ROMIO due to a smaller cost of scatter/gather operations, a reduction of the metadata overhead, and a smaller number of collective communication and synchronization primitives used in the implementation. View-based collective I/O reuses the file access information from the preceding MPI I/O operations in order to guide the subsequent I/O to the processes that hold the most up-to-date cache. In ROMIO, the life of the file domains only spans a single MPI collective I/O call.

View-based I/O uses a compute node cache that caches collective buffers across collective calls. In the first implementation the collective buffers are transferred into the cache on-demand for reads and when the cache is full or when the file is closed for writes.

The aim of this on-going work is to include in view-based collective I/O two data staging strategies for prefetching and flushing the collective I/O buffer cache. The prefetch and flush aim to overlapping computation and I/O and to improve collective buffer I/O read and write performance by making better use of the network to increase effective I/O bandwidth.

The prefetching strategy is based on MPI-IO views. In MPI-IO, views are contiguous windows that map on not-contiguous file regions. They disclose potential future accesses, and allow aggressive prefetching strategies to be employed. In our approach, the prefetch is done in coordinate manner, by aggregating the view information of several processes and reading ahead whole blocks from the file system.

The flushing strategy allows for overlapping the computation and I/O, by gradually transferring the data from the collective buffer cache to the file system. This approach distributes the cost of the final flush (at file close) over the computation time, given the fact that many scientific applications alternate computation and I/O phases. This approach should reduce also the rates at which the buffer cache becomes full with dirty file blocks, which may clog the computation to go on.

Currently, we have already implemented the mechanisms for enforcing these two strategies and are estimating the efficiency of this approach for large scale scientific parallel application. We are investigating the tradeoff between the contradictory goals of promoting data by prefetching, demoting the data by flushing and temporal locality.

References

[1] R. Thakur, W. Gropp, and E. Lusk, "*Data Sieving and Collective I/O in ROMIO*," in Proc. of the 7th Symposium on the Frontiers of Massively Parallel Computation, February 1999, pp. 182–189.

[2] Javier García Blas, Florin Isaila, David E. Singh and J. Carretero, "*View-based collective I/O for MPI-IO*," to appear in Proceedings of CCGRIG, 2008.