CSCI 6360: Parallel Computing Lecture Summary - 12

Anirban Das (dasa2@rpi.edu)

February 27, 2018

Summary on MPI Parallel I/O topics :

The first part of the lecture looks into parallel I/O in MPI. Traditional sequenctial I/O though has data locality and support of some libraries, lacks in the parallelism mainly in scalability and performance due to a single node bottleneck. Similarly writing files in multiple nodes is a metadata nightmare, and also it is dificult to read back from these MPI tasks. Parallel I/O provides high performance solution to this problem, where it is possible for multiple processes to read/write data from a single file.

MPI is a natural setting to achive this because writing and reading is basically like sending and receiving a message. A single file can be read by multiple MPI tasks either using individual file pointer (MPI_File_read) or using explicit offsets (MPI_File_read_at), and similar functions exists for writing to a file including thread safe functions as well.

For non contiguous accesses users must create derived datatypes and file views.It is also possible to do collective I/O where the requests of multiple processes is optimised and fused into a single request and service them efficiently, and is mostly used when the acesses of different processes are non contiguous and interleaved. It is also possible to pass hints to the implementation.

However to ensure correctness of read write, the user either has to set atomicity true, or close and reopen file or ensure no write sequence on any process is in concurrence with any sequence on another process. Finally optimizations such as data sieving, improving prefetching and caching can also be employed alongside collective I/O to improve performance.

The second part of the lecture looks into performance of 4 different scalable parallel I/O alternatives for massive parallel systems using PHASTA CFD as a workload. They look into 1PFPP, PMPIO, syncIO and rbIO strategies. All strategies perform two types of work, equation formation and implicit iterative equation solution which requires a MPI_Allreduce. The experiments are performed on a BG/L and BG/P installations. A parallel I/O system needs to provide scalable performance even with increasing processor counts but at the same time it should be able to hide the I/O latency somehow. It is found syncIO provides a peak speed of 11.6GBps read and 25GBps write on BG/P where processors write as group to different files. Whereas rbIO at the same time reserves 3-6% of compute nodes as dedicated writers to hide latency and achieve actual 18GBps writing speed and perceived writing performance of 21TBps on BG/P ! Both were significantly better off related to PMPIO and 1PFPP. Also it is found that in syncIO and rbIO , collective IO operations vary w.r.t. number of writers and underlying file system performance and architecture.