Opass: Analysis and Optimization of Parallel Data Access on Distributed File Systems

Jiangling Yin, Jun Wang, Dan Huang, Tyler Lukasiewicz, and Jian Zhou
Department of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL
{jyin, dhuang, jwang}@eecs.ucf.edu

Conventionally, in parallel data analysis, multiple processes running on different cluster nodes share a separate dedicated storage system. Once a data processing task is scheduled to a process, the data will be transferred from the shared storage to the process. However, in today's big data era, large amounts of data movement over the shared network could incur an extra overhead during parallel execution, especially during iterative data analysis, which involves moving data from storage to processes repeatedly.

Distributed file systems, such as GFS, HDFS, QFS or Ceph, could be directly deployed on the disks of cluster nodes to reduce data movement. When storing a data set, distributed file systems will usually divide the data into smaller *chunk files* and randomly distribute them with several identical copies (for the sake of reliability). When retrieving data from HDFS, a client process will first attempt to read the data from the disk that it is running on. If the required data is not on the local disk, the process will then read from another node that contains the required data. In this paper, the data read requests from the parallel processes are referred as *parallel data requests*.

Unfortunately, most parallel data requests will be served remotely due to the lack of consideration of data distribution in HDFS and the task assignment among parallel processes. These remote data requests can cause a serious imbalance of data access on the cluster nodes. Because of this, the overall execution time will be prolonged due to the synchronization requirements in parallel execution. In reality, the I/O performance could be further degraded as the size of the cluster and the data increase.

In this paper, we propose novel matching-based algorithms for optimizing parallel read access. Our goal is to reduce remote data accesses on HDFS for parallel data analysis and thus achieve a higher balance of data read requests between cluster nodes. To achieve this goal, we retrieve the data layout information from the underlying distributed file system and model the assignment of processes to data as a one-to-many matching in a *Bipartite Matching Graph*. We then use the matching-based algorithms to compute a solution that enables parallel data access to be served on HDFS in a local and balanced fashion.

To verify our method, we run MPI parallel processes on Marmot to read a dataset from HDFS via two methods. The first method, in which the data assignment of each process is mainly decided by its process rank. The second method is our proposed method: Opass. We plot the I/O time taken to read every chunk on a 64-node cluster, which contains 640 chunks and the size of each chunk is approximately 64 MB. The execution results are shown in Figure 1. The figure shows that without the use of Opass, the I/O time increases dramatically after the initiation of the execution. In contrast, with the use of Opass, the I/O time during the entire execution is approximately one to two seconds.

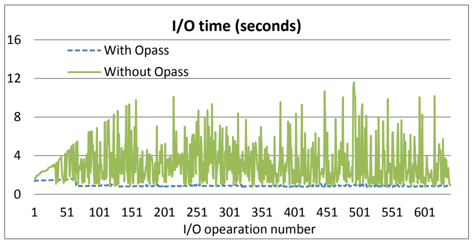

Figure 1: I/O times comparison on Parallel Access.

we also plot the amount of data served by each node on a 64 node cluster in Figure 2. We find that the amount of data served per node varies greatly without the use of Opass. Some nodes, such as node-44, serve more than 1400 MB of data while some node serves 64 MB. Such an imbalance will cause the disk head to become a bottleneck, and thus the I/O read time could increase, as shown in Figure 1. In contrast, with the use of Opass, every storage node serves approximately 640 MB.

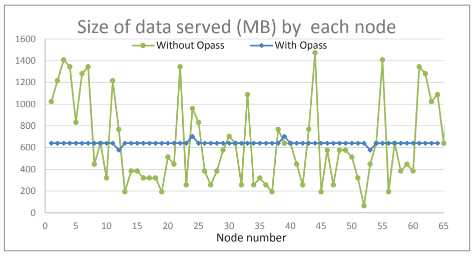

Figure 2: Access patterns comparison on Parallel Access.