ifarm: Implementing Inline Deduplication to a Distributed File System

Ryo Matsumiya*, Shin Sasaki*, Kazushi Takahashi*‡, and Yoshihiro Oyama*‡
*The University of Electro-Communications
‡JST, CREST

In the field of HPC, storage systems have been designed with a hierarchy of storage devices. Some clusters that adopt a distributed file system such as Gfarm [1] save data to and load data from huge storage, such as tape devices, according to circumtances. Computation nodes read data from a distributed file system during their computation. Computation results are stored to a distributed file system, which usually consist of multiple I/O servers.

We consider the case in which the computation nodes read and write too large data to store to disk spaces provided by the distributed file system. In this case, the distributed file system must load (store) from (into) the huge storage multiple times, and these operations can cause overheads on file access performance. Therefore, it is required to reduce the amount of data stored in I/O servers and decreases data transfer operations between a distributed file system and huge storage.

A widely used approach to reduce disk space consumption is *deduplication*. Deduplication systems enable to store only one physical copy of duplicate data when duplicate data exist among different files or within a single file. Deduplication systems were implemented to many storage systems. Meister et al. measured the effect of deduplication to reduce data of HPC applications managed by practical storage [2]. However, Meister et al. did not implement any deduplication system working with actual file systems.

We are developing *ifarm*, which is an implementation of Gfarm with deduplication of file data. Gfarm is a distributed file system used in the field of HPC. Gfarm builds distributed file systems by a single metadata server and multiple I/O server(s). The metadata server stores the metadata of Gfarm files, including the list of I/O server(s) that each file is placed on. I/O servers are stored the contents of files.

There are two methods to introduce deduplication to a file system. One is the offline method. The offline method performs deduplication while no client is accessing the storage. The other method is the inline method, in which clients can access the storage during deduplication. As we described above, we consider the case in which computation nodes read and write too large data to store to a distributed file system. The offline method is not advantageous in this case since computation nodes running HPC applications must continue to access files even while deduplication is performed. Hence, this work focuses on the inline method.

Some deduplication systems adopt content defined chunking (CDC), which is a deduplication method that divides a file into variable-size chunks based on their content. ifarm also adopts CDC. Although CDC enables to use storage efficiently, CDC takes a long time to divide large files because it calculates many fingerprints of file data. Fast execution of CDC is required to achieve practical inline deduplication. Although many researchers attempted to parallelizing chunking operations, their methods do not address a problem of overheads proportional to sizes of divided files. Therefore, the overheads incurred in chunking large data are not sufficiently small. To hide this overhead, ifarm calculates the fingerprints of file data asynchronously with operations for handling requests from clients.

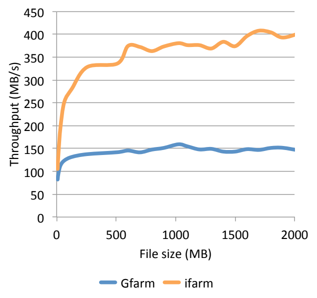

Fig. 1. Result of reading zero-data files

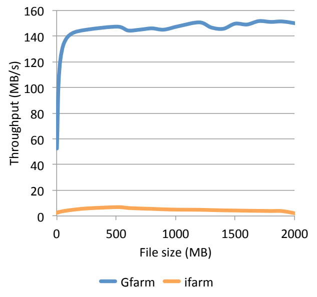

Fig. 2. Result of reading random-data files

We evaluated ifarm through two programs. One program sequentially reads a zero-data file, which indicates a workload to which ifarm shows the best performance. The other program sequentially reads a random-data file which indicates a workload to which ifarm shows the worst performance. Compared with original Gfarm, ifarm was 117% faster in the best case (Fig. 1), and 99.8% slower in the worst case (Fig. 2). In the future, we must improve the performance of ifarm, particularly the one in the worst case. We will also conduct more benchmarks to analyze the performance of ifarm.

References

[1] Osamu Tatebe, Kohei Hiraga and Noriyuki Soda, "Gfarm grid file system," no., 3, 2010.

[2] Dirk Meister, Jügen Kaiser, Andre Brinkmann, Toni Cortes, Michael Kuhn and Julian Kunkel, "A Study on Data Deduplication in HPC Storage Systems," in *the Proc. of IEEE/ACM SC '12*, 2012.