HDFS+: Concurrent Writes Improvements for HDFS

Kun Lu, Dong Dai, Mingming Sun

INTRODUCTION

Hadoop Distributed File System (HDFS) [3] is an open source for Google File System (GFS) [1] implementation, has been adopted by Amazon, Yahoo!, Facebook and other companies for large-scale data storage. With HDFS, data intensive programs could get high-throughput with simple operations.

However, in many situations, records which application needed were not native in HDFS [2]. Consider applications running on thousands of cluster nodes generating log records respectively in local disk. It is difficult using the log data spreading in numerous nodes by the log analysis programs.

Traditional method is that making long-running processes called agent on each machine for collecting log from disk sending to a unique collector. Writes these records to HDFS file through a collector node.

Traditional method is that making long-running processes called agent on each machine for collecting log from disk sending to a unique collector. Writes these records to HDFS file through a collector node.

We propose a HDFS based distributed file system named HDFS+ which can accept concurrent writes with multi data sources besides the basic capacity of HDFS. The HDFS+ does not guarantee the restrict order of records generated in different nodes. We introduce a new concept of fragment. Many independent files can be seen as a single file. HDFS+ does not provide any write functions to clients. From the viewpoint of log analysis program, it can not write records to the log file, they just read from it. Even so, the log file becomes larger due to log records are written to local disk by applications.

MOTIVATION

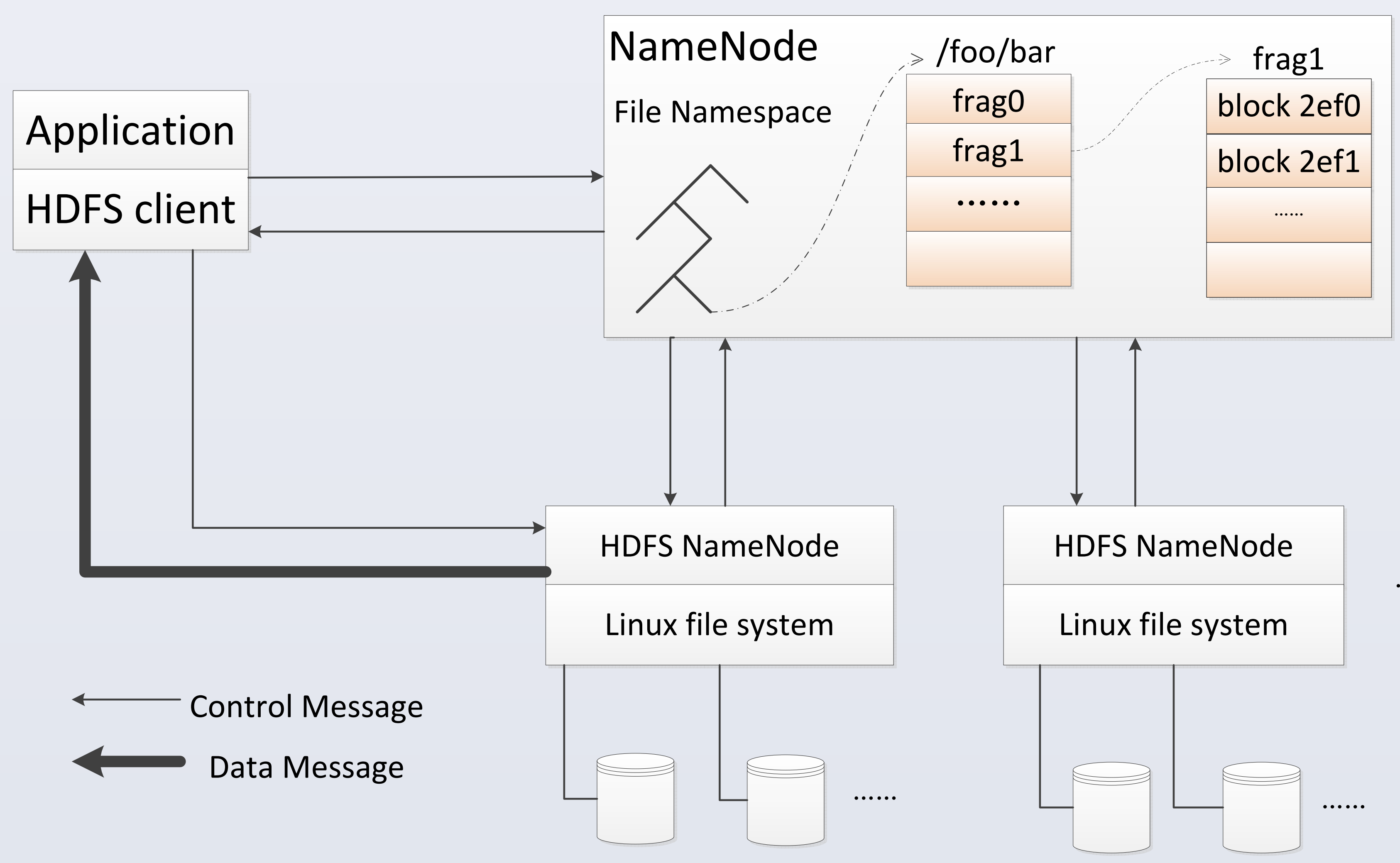
We intend to use HDFS+ to improve programs like log analysis in cluster of several thousands of hosts. There is a lot of difference in file access mode between cloud environment and traditional mode. More applications only need to obtain a set of data, without the need to obtain the sequential relationship between the data. Two prerequisites are needed in our solution:

Firstly, there are numerous data producers appending data to a file in the clusters and the data consumers won't execute mutating operations.

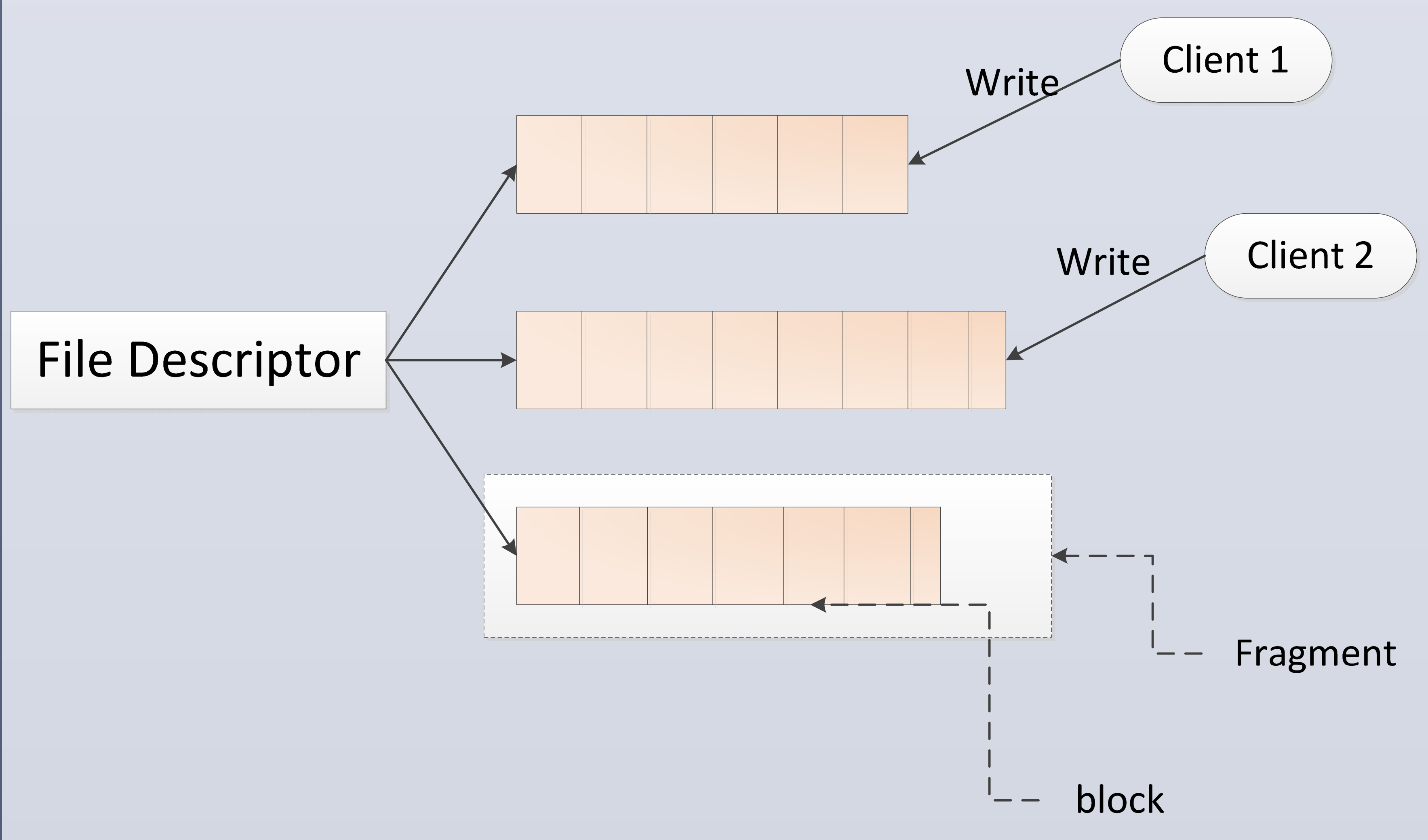
Secondly, since records are self-representable that means records contain the information such as time stamp etc. By using the information included in records, programs could aware the record's rough sequence order of all records. A few applications in data analysis may use time sequence as one of parameters in analysis. As the matter of fact, in distributed system, there is no restrict sequence order in two distinct nodes if no obvious synchronization. In HDFS+ programs could distinguish record's actual rank in the local records, and rough rank in all records distributed in all nodes by using timestamp information.

METHOD

We use fragments as interlayer connects the namespace layer and data block layer. Each fragment is responsible for a concurrent write task. The fragment is divided into HDFS original blocks and can be written by a program. Multiple fragments can be increased by multiple programs, one program to one fragment. We use snapshot technology to achieve the file read operation. When the client opens a file in HDFS+, snapshot will creates every copies of fragments belongs to this file. We can create a single HDFS+ file from many local files by exchanging a local file to a fragment.



Fragment is the basic unit of concurrent writes, one fragment can be written by one client at the same time. As in HDFS+, files are divided into length variable fragments, and fragments are composed by fixed-size blocks. Clients can read file using a sequential manner. When the client opens a file in HDFS+, snapshot will creates every copies of fragments belongs to this file.



REFERENCES

[1] Sanjay Ghemawat, Howard Gobioff and Shun-Tak Leung. The Google file system. SIGOPS Oper. Syst. Rev., 2003.

[2] Ariel Rabkin and Randy Katz. Chukwa: a system for reliable large-scale log collection. Proceedings of the 24th interna-tional conference on Large installation system administration, 2010.

[3] Shvachko, K., Hairong, Kuang, Radia, S., and Chansler, R. The Hadoop Distributed File System. Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, 2010.