Size Matters: Improving the Performance of Small Files in HDFS *

Salman Niazi †, Seif Haridi, Jim Dowling

KTH - Royal Institute of Technology

{smkniazi, haridi, jdowling}@kth.se

The Hadoop Distributed File System (HDFS) [2] is widely used as a storage platform for large volumes of data. A metadata server, called the namenode, manages all of HDFS' metadata. HDFS is primarily designed to provide streaming access to large files, from many megabytes to gigabytes in size. For large files, the cost of the metadata operations in HDFS is amortized over the relatively long periods of time spent reading or writing large files. However, for small files, metadata processing time becomes a larger fraction of total filesystem operation time (given the minimal time required to read data blocks for small files from the datanodes). In many production deployments of the HDFS, it has been observed that 20% of the files in the system are less than *4 kilobytes (KBs)*, and these files receive 20% of all the file system operations, see Figure 1. As a result, HDFS clients experience poor performance for small files in moderately sized/loaded clusters, as the namenode becomes a bottleneck, increasing metadata processing latency. We propose optimizing the file system operations on small files by using *stuffed inodes* for small files, in a version of HDFS with distributed metadata, called HopsFS [1]. Inode stuffing is a technique that embeds the data blocks of small files in the inodes' metadata for low latency file system operations.

1. Small Files in HopsFS

HopsFS is a drop-in replacement for HDFS that stores file system metadata in a highly available, in-memory, distributed relational database, called the MySQL Cluster. HopsFS supports multiple stateless namenodes with concurrent access to the in-memory file system metadata, enabling greater than an order of magnitude higher throughput for metadata operations as well as lower latency for large numbers of concurrent clients.

In ongoing work on HopsFS, we are storing the data blocks for small files in the database (on-disk) alongside the inodes' metadata (in-memory). When a client contacts a HopsFS namenode to read a small file, the namenode fetches the metadata along with its data blocks from the database and returns the file blocks to the client. This reduces read latency as it removes a network round-trip to datanodes for each block from the HDFS client protocol. While HopsFS' metadata is stored in-memory in the database, the data blocks are stored on-disk in the database, ideally on Solid State Drives (SSDs). SSDs enable lower latency and higher throughput access to on-disk (database) data, as well as more cost-effective and larger storage capacity for blocks compared to main memory.

* Funded by Swedish Foundation for Strategic Research project *E2E-Clouds*

† Salman Niazi is a PhD student and he will present the poster

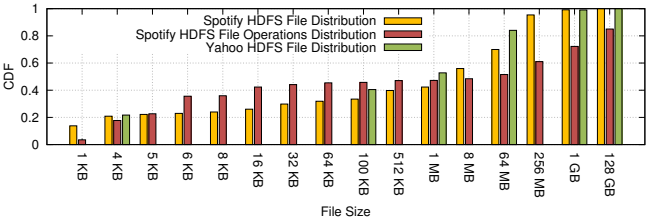


Figure 1: Distribution of file sizes in HDFS at Spotify and Yahoo where ~20% of files are smaller than 4KB.

Non-volatile memory, such as, Intel's 3D XPoint is another candidate technology for storing small files. To evaluate our solution, we are working with operational traces for a HDFS cluster at Spotify containing 357 million files, of which 71 million are smaller that 4KB in size. In Apache HDFS, these files consume ≈ 800 GB of disk space with triple replication. In HopsFS all the data blocks for the small files can easily be stored in the distributed database along with the file system metadata to provide low latency file system operations for the small files. Following are some challenges introduced by using embedded inodes for small files in HopsFS.

- **HDFS Compatibility:** Changes for handling small files should not break HopsFS' compatibility with HDFS clients (which expect the data to reside on the datanodes). To this end, HopsFS namenodes can act as datanodes, serving blocks for small files to the file system clients (block location transparency).
- **Migrating Data between Different Storage Types:** When the size of a small file that is stored in the database exceeds some threshold then the file is reliably and safely moved to the datanodes and vice versa.

2. Preliminary Results

In our micro benchmarks we tested the performance of our metadata back-end, a two-node MySQL Cluster database, as a block service. Each database node had one PCI Express SSD, that stored the on-disk tables. Despite our small setup we were able to perform 25,000 4KB random file block writes and 70,000 4KB random file block reads per second. With a bigger database cluster and more SSDs the performance is expected to increase linearly. These results give us confidence to further develop HopsFS to store both small files for a Spotify-Yahoo type workload and the in-memory metadata in the same database back-end.

References

[1] S. Niazi, M. Ismail, S. Haridi, J. Dowling, S. Grohsschmiedt, and M. Ronström. Hopsfs: Scaling hierarchical file system metadata using newsql databases. In *15th USENIX Conference on File and Storage Technologies (FAST 17)*, pages 89–104, Santa Clara, CA, 2017. USENIX Association.

[2] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, Washington, DC, USA, 2010.

Size Matters: Improving the Performance of Small Files in HDFS

Salman Niazi[1], Seif Haridi[1,2], Jim Dowling[1,2]

[1] KTH - Royal Institute of Technology
[2] RISE SICS - Swedish Institute of Computer Science

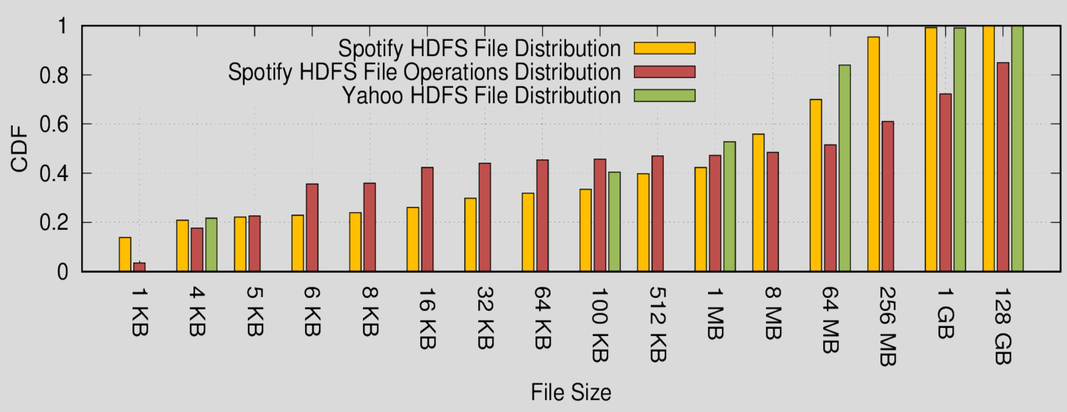
HopsFS is a next generation distribution of the Hadoop Distributed File System (HDFS). HopsFS can perform more than a **million file system operations** per second. Contrary to **HDFS**, which is only suitable for storing **large files**, **HopsFS** can efficiently handles both **large and small files**. HopsFS uses **Inode stuffing** technique to embed small files data with the metadata that significantly improves the performance of the small files**.**

Introduction

The Hadoop Distributed File System (HDFS) [1] is the most popular open-source platform for storing large volumes of data. HDFS splits files to large blocks that are replicated across multiple datanodes. The file to block mapping is stored on single metadata server called the namenode. HDFS is suitable for storing large files. File system operations such as reading a writing files, require the file system clients to first contact the namenode to fetch the location of the data blocks, and then establish communication channels with the datanodes to read/write the files. For large files, the overhead of fetching the metadata and establishing communication channels with the datanodes is amortized over the relatively long periods of time spend in reading and writing large files. For small files this overhead is significantly more than the time required to read/write the actual file blocks. For HDFS the problem is compounded by the fact that it only supports single active namenode which becomes performance bottleneck in the presence of large number of small files.

In many production deployments of the HDFS, it has been observed that 20% of the files in the system are less than 4 kilobytes (KBs), and these files receive 20% of all the file system operations, see figure below. As a result, HDFS clients experience poor performance for small files in moderately sized/loaded clusters, as the namenode becomes a bottleneck, increasing metadata processing latency.

HDFS Files and FS Operations Distribution

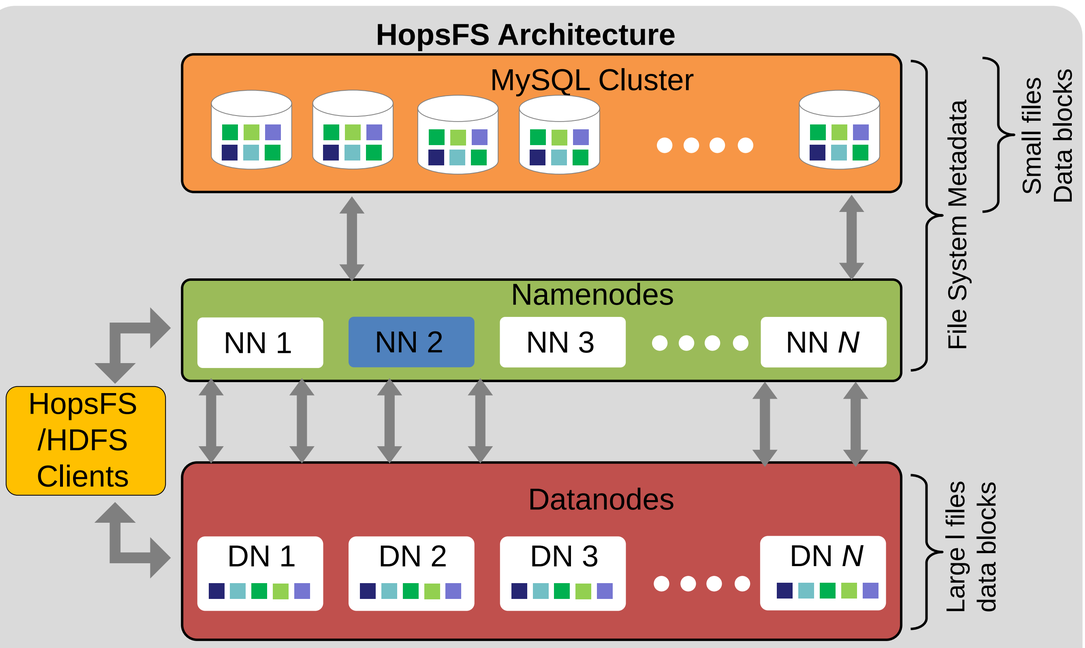


Distribution of file sizes and file system operations in HDFS clusters at Spotify and Yahoo. At Spotify and Yahoo ≈20% of files are smaller than 4KB and these small files receive ≈20% of all the file system operations.

HopsFS

HopsFS [2] is a drop-in replacement for HDFS that stores file system metadata in a highly available, in-memory, distributed relational database, called the MySQL Cluster. HopsFS supports multiple stateless namenodes with concurrent access to the in-memory file system metadata, enabling greater than an order of magnitude higher throughput for metadata operations as well as lower latency for large numbers of concurrent clients.

HopsFS Architecture



HopsFS supports multiple stateless namenodes. In HopsFS large files are split into blocks that are replicated across multiple datanodes. The metadata for the entire file system along with the data blocks for all the small files is stored in the distributed database. Storing the file blocks of small files with the metadata improves the performance of the file system operations.

By storing the file system metadata in an external distributed database, HopsFS can easily handle large amount of file system metadata. In ongoing work on HopsFS, we are using inode stuffing technique, that is, storing the data blocks for small files in the database (on-disk) alongside the inodes' metadata (in-memory). When a client contacts a HopsFS namenode to read a small file, the namenode fetches the metadata along with its data blocks from the database and returns the file blocks to the client. This reduces read latency of the file system operation as it removes multiple network round-trip to datanodes for each block from the HDFS client protocol. While HopsFS' metadata is stored in-memory in the database, the data blocks are stored on-disk in the database, ideally on Solid State Drives (SSDs) . Non-volatile memory, such as, Intel's 3D XPoint is another candidate technology for storing small files.

Results

In our micro benchmarks we tested the performance of our metadata backend, a two-node MySQL Cluster database, as a block service. Each database node had one PCI Express SSD, that stored the on-disk tables. Despite our small setup we were able to perform 25,000 4KB random file block writes and 70,000 4KB random file block reads per second. With a bigger database cluster and more SSDs the performance is expected to increase linearly.

References

[1] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. **The hadoop distributed file system.** In Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), MSST '10, Washington, DC, USA, 2010. IEEE Computer Society.

[2] Salman Niazi, Mahmoud Ismail, Seif Haridi, Jim Dowling, Steffen Grohsschmiedt, and Mikael Ronström. **HopsFS: Scaling hierarchical file system metadata using newsql databases.** In 15th USENIX Conference on File and Storage Technologies (FAST 2017), pages 89–104, Santa Clara, California, 2017. USENIX Association.

Register on http://hops.site to try Hops Hadoop distribution on our cloud for free.