DFS on a Diet: Enabling Reduction Schemes on Distributed File Systems

Ryan Nathanael Soenjoto Widodo
Graduate School of Systems and Information Engineering,
ryannsw(at)osss.cs.tsukuba.ac.jp

Hirotake Abe
Department of Computer Science,
habe(at)cs.tsukuba.ac.jp

Kazuhiko Kato
Department of Computer Science,
kato(at)cs.tsukuba.ac.jp

Operating Systems and System Software

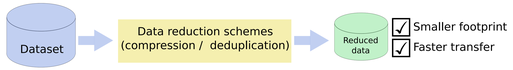
筑波大学 University of Tsukuba

SC20 Everywhere we are | more high tech

1. Background and motivation

Data reduction schemes are important for minimizing storage space.

- It can also reduce data size in Distributed FileSystems (DFS).



- Some data reduction schemes can work better than others.

Domain specific (LogZip*) vs. general purpose scheme (Gzip)

But...

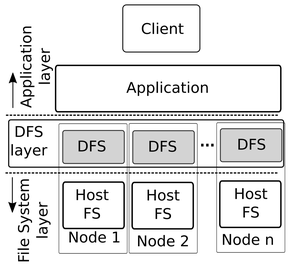
1. The schemes' selections are usually limited.

☑ Gzip ☑ Snappy
☑ Bzip2 ☑ Lz4
☑ Zlib ☐ Deduplication

2. Adding a new scheme can be both time and cost expensive.
   - Depending on user's familiarity with the platform and source code.
3. Only usable in some applications.
   - Compression codecs in Hadoop is not available in some applications like DistCP (Distributed Copy).

*Jinyang Liu, Jieming Zhu, Shilin He, Pinjia He, Zibin Zheng, and Michael R Lyu. 2019. Logzip: Extracting Hidden Structures via Iterative Clustering for Log Compression. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 863–873.

2. Problems of conventional approaches

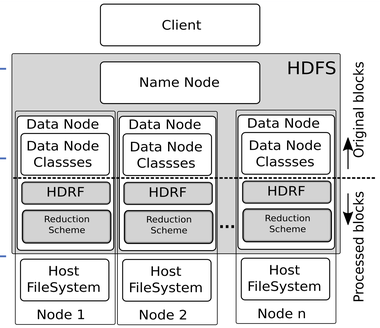


	Application layer	File System layer	DFS layer
Difficulty of adding new schemes	○ Platform libraries	△ Difficult even with source code mod.	△ Difficult and may break the DFS functionalities.
Application Source Code Transparency	✕ Application code required	○ Application code not required	○ Application code not required
Network Traffic Reduction	○ At DFS layer and below	✕ No data reduction outside FS layer	○ Possible reduction with deeper integration into the DFS code.
Example	Hadoop Compression Codecs	ZFS for HDFS** or Lustre	Currently, there is no such implementation at the DFS layer because it adds complexities to the DFS design and implementation

Adding data reduction schemes with existing approaches can be difficult and may not work with all applications.

**Ruijin Zhou, Ming Liu, and Tao Li. 2013. Characterizing the efficiency of data deduplication for big data storage management. In 2013 IEEE international symposium on workload characterization (IISWC). IEEE, 98–108.

3. Apply at the DFS layer



Our solution:

1. We create a framework for the DFS to connect the data reduction scheme.
2. The framework use programming techniques like dependency injection (DI) to connect with the data reduction scheme.
3. The framework processes the data before it is passed to the FileSystem layer to minimize the impact on the DFS's functionalities.

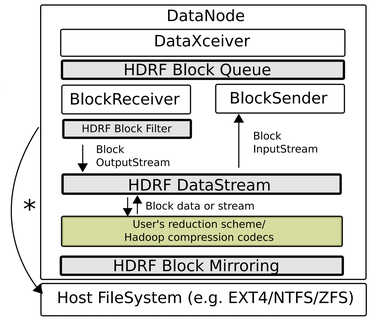
4. Hadoop Data Reduction Framework (HDRF)

General DFS

- 2 options for block transfer.
  - Stream mode.
    - For data reduction scheme with I/O stream
  - Block array/buffer mode.
    - Higher memory usage.
    - Easier implementation.

HDFS Specific optimizations

- Block queue.
  - Limit parallel block access. in HDFS to avoid IO congestion.
  - For single stream reduction schemes.
- Block filter.
  - Only process the data block.
  - Exclude MapReduce job files.
- Block mirroring.
  - Block replication through HDRF.
  - Reduces the network traffic during replication.

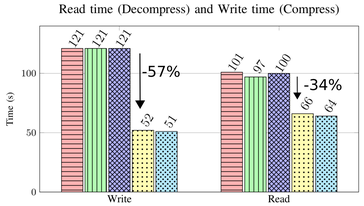


* Direct block storage for MapReduce job blocks

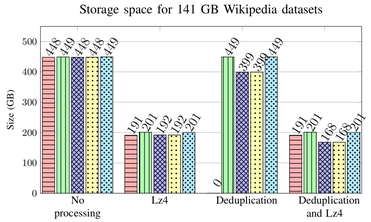
Check out our implementation details in our paper*!

*R. Nathanael, S. Widodo, H. Abe, and K. Kato, "HDRF: Hadoop data reduction framework for hadoop distributed file system," in Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems, 2020, pp. 122–129

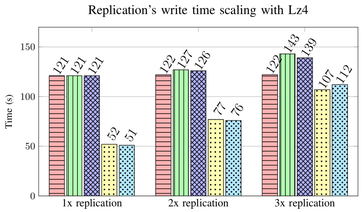
5. Experimental results



- Runtime overhead <3%.
- Halve the write time with a more efficient data transfer method (DistCP).



- Storage overhead <1%.
- Works with a custom deduplication scheme.



- Replication increases network traffic for DFS operations.
  - 14% slower at 3x replication.
  - 12% faster at 3x replication with DistCP.

Vanilla HDFS | ZFS | HDRF | HDRF with DistCP | ZFS with DistCP

6. Conclusion

- We proposed an unintrusive design for reduction scheme in a DFS.
- We Implemented a proof of concept in HDFS.
- We tested and the results show:
  - Small overhead and works on all Hadoop applications.
  - Up to half read and write time reduction when using DistCP.
  - The storage space is not compromised.

7. Future work

- Block mirroring to reduce network traffic during block replication.
- Implementation on other DFSs.
- Usage of HDRF for other applications.
  - Storing blocks in persistent memory through HDRF.