Multi-Level Hashing Dedup in HPC Storage Systems

Eric Valenzuela (Undergraduate)
Department of Computer Science
California State University, Channel Islands
Camarillo, USA
eric.valenzuela798@myci.csuci.edu

Yin Lu (Advisor), Susan Urban (Advisor), Yong Chen (Advisor)
Department of Computer Science
Texas Tech University
Lubbock, USA
{yin.lu, susan.urban, yong.chen}@ttu.edu

***Abstract*— Reaching high ratios of data deduplication in High Performance Computing (HPC) is highly achievable. Prior art demonstrates magnitudes of reduction possible and 15 to 30 percent of redundant data can be removed on average using deduplication techniques. The objective of this research study is to design and experiment a dedup system to provide 100% data integrity without a possibility of losing data while reducing the need of costly byte-by-byte comparisons. Because data deduplication uses hashing algorithms, hash collisions will occur. Prior systems ignore byte-by-byte comparisons that are needed to handle collisions citing the probability is low. Our research focuses on investigating a multi-level dedup method to reduce byte-by-byte comparisons while providing 100% data integrity, and the implementation of multi-level hash functions while talking advantage of Xeon Phi many-core architecture to compute cryptographic fingerprints concurrently. Our current proof-of-concept evaluations with a deduplication file system, Lessfs, show promising results.**

Keywords— Deduplication; File Systems; Multi-Level Hashing; Performance; Storage

I. INTRODUCTION

A. Data Deduplication

Data deduplication is the process of removing redundant data and leaving only one single copy of data behind. It is a method that eliminates duplicate copies of identical data blocks. Storage systems save their capacity when applying this technique by simply creating a pointer to the data block that was already present on the system.

B. Types of Data Deduplication

There are generally two types of data deduplication: inline and offline. Inline handles data deduplication while data is being written to the storage volume, and offline handles data deduplication after data has been transferred into the storage volume.

Both methods have its advantages and disadvantages. The inline method requires less storage volume because data deduplication is being processed in real time, but there is a degradation in performance because more CPU overhead is required as it tries to identify redundant data simultaneously. On the contrary, the offline method requires additional storage capacity because data deduplication has not yet been applied until all the files have been transferred which may become a problem if the storage capacity is low.

C. Deduplication Methods

Note that data deduplication can be applied at the file, block, and bit level. Our research focuses on the block and bit level of data deduplication. The potential of achieving high data deduplication ratios are lost when full file data deduplication is applied. For instance, if two files contain 99.9 percent of identical data and the only difference is a single change of a bit, the system will consider the files as completely different pieces of data while not taking full advantage of the possible identical data within the file.

D. HPC Data Deduplication

There are noticeable file statistics in HPC. For example, 10 percent of the files in HPC occupy 90 percent of storage [1]. Some of the most common file types with the highest data deduplication ratios are NetCDF files, nc and ncf. Small files have also demonstrated to have large data deduplication results, but unfortunately, contribute less to the overall savings. In fact, only a small fraction of all chunks often contribute the most to data deduplication results.

II. RESEARCH OBJECTIVES

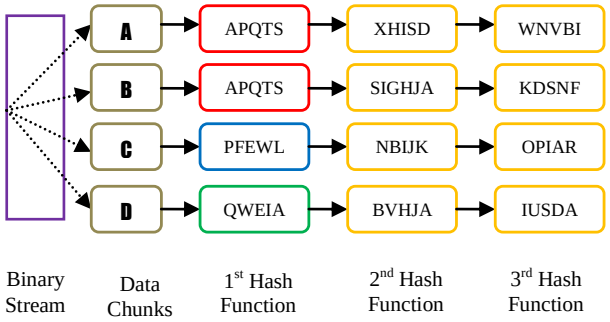
As data deduplication uses hashing algorithms, hash collisions will occur, and prior systems ignore byte-by-byte comparisons that are needed to handle collisions citing the probability is low, which is translated to the possibility of losing data. Throughout our current research, our goal is to design a data deduplication system that provides 100 percent data integrity without the possibility of losing data. We introduce a multi-level data deduplication method to reduce costly byte-by-byte comparisons while maintaining 100 percent data integrity. The implementation of the multi-level deduplication technique is designed to take advantage of Xeon Phi many-core parallel architecture to compute cryptographic fingerprints concurrently.

III. APPROACH

This study is based on an open source tool Lessfs [7], which is an inline data deduplication filesystem for Linux, known as Lessfs – Data Deduplication for Less. It has great specifications including built in compression and strong

encryption, supports master/slave replication, and offers the choice of "back end" systems.

Lessfs is implemented at the user space, requires the Tokyocabinet Database to be built, and also uses FUSE libraries [7]. Lessfs even uses the QuickLZ algorithm to apply additional compression on top of data deduplication [7]. With the combination of this top-notch compression algorithm [4], Lessfs is one of the greatest open source file systems for data deduplication. Lessf's most common hashing algorithm is Tiger (cryptography), along with seven other hashing algorithms including Whirlpool, Haval, Shefru, Ripemd, SHA1, SHA2, and Midnight Blue.



A. *Multi-Level Hashing*

The focus of this study is not in re-discovering the idea of using multiple hash functions to avoid hash collisions [2], but to specifically apply the technique in data deduplication for HPC storage systems.

By applying multi-level hashing we maintain 100 percent data integrity and reduce the need of costly byte-by-byte comparisons that ensure the data integrity when performing data deduplication. For instance, when two data chunks produce the same cryptographic fingerprint, current dedup file systems cannot immediately determine if both blocks of data are truly redundant or if a collision has occurred. Instead, Lessfs, as most dedup file systems, rely on the statistical argument that the probability for a hash collision to occur is very low and treat both blocks as identical, thus increasing the potential of losing data if a hash collision occurs.

By-by-byte comparisons are needed to ensure the removed block is indeed identical to an existing block, not a hash collision scenario. However, because byte-by-byte comparisons are an expensive process to ensure integrity, several data deduplication file systems avoid the implementation. Our motivation is to ensure no unique data is discarded due to hash collisions, which is certainly important, especially in mission-critical systems such as aircrafts, space flight, military, or medical systems.

In this research investigation, we conduct byte-by-byte comparisons for ensuring 100 percent data integrity in data deduplication systems. We also introduce multi-level dedup to reduce costly comparisons and leverage Xeon Phi many-core parallel architecture to speed up hash calculations. Figure 1 demonstrates the concept of multi-level hashing. If several cryptographic fingerprints map to the same hash index, multiple hash functions are used as needed in data deduplication until we ensure the current data blocks being analyzed are in fact redundant.

B. *Proof of concept prototype*

Before implementing the multi-level hashing technique into Lessfs, we coded a sequential and parallel program that performs multi-level hashing for large climate data sets. To demonstrate the concept and evaluate the potential, we compared the performance on Xeon Phi nodes in which cryptographic fingerprints are concurrently hashed

IV. RESULTS

We gathered three unique climate data sets and compared the performance results for applying multi-level hashing. Climate data sets were used for testing purposes because they tend to have the highest data deduplication ratio in HPC storage systems. Two main outcomes from the performance test results demonstrate that a 90 percent of time reduction can be achieved when hashing large files using a 1GB buffer, than compared to at the sequential level.

V. CONCLUSION AND FUTURE WORK

In this study, we investigated and experimented a dedup system to provide 100 percent data integrity while reducing the need of costly byte-by-byte comparisons via a multi-level dedup method. Prior systems ignore byte-by-byte comparisons to handle collisions, and even the probability of hash collision. While data loss is low, there is still a possibility. Our study provides a multi-level dedup method to reduce byte-by-byte comparisons while providing data integrity. The multi-level dedup is also designed to leverage many-core architecture (Xeon Phi) to compute cryptographic fingerprints concurrently. Our current proof-of-concept evaluations have confirmed promising results. We intend to continue the evaluation with the Lessfs dedup system and evaluate with more high performance scientific computing workloads.

REFERENCES

[1] Meister, D., Kaiser, J., Brinkmann, A., Cortes, T., Kuhn, M., & Kunkel, J. (2012, November). A study on data deduplication in HPC storage systems. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (p. 7). IEEE Computer Society Press.

[2] Broder, A., & Mitzenmacher, M. (2001). Using multiple hash functions to improve IP lookups. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*(Vol. 3, pp. 1454-1463). IEEE

[3] Mark Ruijter. lessfs - open source data de-duplication. http://www.lessfs.com, February 2012. [Online; Consulted on June 23, 2014]

[4] Quicklz - official website. Website, http://www.quicklz.com.

[5] Lessfs. Lessfs web page. http://www.lessfs.com/wordpress/. June 2014.

[6] Koutoupis, P. (2011). Data deduplication with Linux. *Linux Journal*, *2011*(207), 7.