Petaflop 슈퍼컴퓨터 스토리지의 데이터 중복 잠재성 분석 연구

조용현°[1], 사프다 자밀[1], 김기현[1], 이재국[2], 안도식[2], 홍태영[2], 김영재[1]
[1]서강대학교 컴퓨터공학과, [2]한국과학기술정보연구원
{yongcho, safdar, realltd16, youkim}@sogang.ac.kr, {jklee, dsan, tyhong}@kisti.re.kr

An Analysis of Data Deduplication Potential on Storage System of Petaflop Supercomputer

Yonghyeon Cho°[1], Safdar Jamil[1], Kihyun Kim[1], Jae-Kook Lee[2], Dosik Ahn[2], Taeyoung Hong[2], Youngjae Kim[1]
[1]Department of Computer Science and Engineering, Sogang University, Seoul, Republic of Korea
[2]Korea Institute of Science and Technology Information

Abstract

The petaflop supercomputers have provided optimal resources for numerous scientific domains. Specifically, the storage nodes in supercomputers support tens to hundreds of petabytes of capacity and they are managed by parallel file system. Several studies have explored storage space optimization techniques, such as data deduplication (dedup). The dedup reduces the storage space by identifying and eliminating the duplicate data. However, these studies only considered the supercomputers at *tera-scale*. Therefore, in this work, we perform data dedup analysis on the Nurion petaflop supercomputer operated by Korea Institute of Science and Technology (KISTI) using an in-house dedup tool, named Deduplication Analyzer (Danzer). We observe that on average 25% of the data can be removed by adopting data dedup.

1. Introduction

The National Supercomputing Center at the Korean Institute of Science and Technology Information (KISTI) in South Korea has a long history of deploying and operating world class supercomputer. KISTI is home of the Nurion [1] petaflop supercomputer that is ranked 11^{th} in 2018 on the Top500 list [2], with a peak performance of 25.3 Petaflops (PF). Nurion is a Cray CS500 with 8,305 compute nodes, each compute node with 64 cores Intel Xeon Phi processor.

Nurion contains 797.3 TB of aggregated memory and 21 PB of storage capacity managed by Lustre parallel file system (PFS) [3]. Furthermore, the storage layer of Nurion also employs burst buffers (BB) to absorb high I/O demand. The BB acts as an I/O buffer by storing bursty I/O request traffic, which occurs instantly in the application program, between the application and the PFS. In addition, to store the huge amount of data produced by HPC simulation, Nurion maintains a peta-scale storage system based on Luster. However, in order to store the constantly produced simulation data, Nurion adopts a purging policy that deletes files older than two weeks. The purging process releases the resources acquired by clients and make them available for new users. However, the purge policy is usually implemented based on time and only triggered once the required time duration is met, which can lead to exhaustive utilization of PFS. To address this, a few studies considered adopting the storage optimization techniques, such as data deduplication [4, 5].

Data deduplication (dedup) reduces the storage space by identifying and eliminating the duplicate data. Meister et. al [4] estimated the data dedup potentials in the storage systems of supercomputer and analyzed more than 1,000 TB of data stored at PFS and observed that 20% to 30% of the data can be removed by adopting data dedup. However, this study only considered supercomputers at *tera-scale* while in current era supercomputers are of *peta-scale*. The increase in resources and induction of new application domains such as machine learning at supercomputing facilities make it necessary to once again analyze the potentials of storage optimization techniques.

Therefore, in this work, we perform data dedup analysis atop Nurion supercomputer and present our observations on the datasets we collected during a span of a year. To perform the analysis, we developed an in-house dedup tool, named Deduplication Analyzer (Danzer), which estimates the benefits of employing data dedup on supercomputers. To the best of our knowledge, this is the first work to perform dedup analysis on peta-scale supercomputer. Danzer is a two phase tool, *trace* and *analysis* phase. In the *trace* phase, Danzer traverses the filesystem, performs dedup steps (fixed-size chunking and fingerprinting), and writes the fingerprints to a *trace-file*. In the *analysis* phase, Danzer performs dedup analysis and extracts insights based on the *trace* phase. We observed that on average 25% of the data can be removed by adopting data dedup at supercomputers and reduce the space utilization.

This work was supported by the Korea Institute of Science and Technology Information (Grant No.J-22-NB-C03-S01).

2. Data Deduplication (Dedup)

Data dedup is a key storage space optimization technique adopted in various distributed storage systems like cloud. Dedup reduces the storage space by identifying and eliminating the duplicate data. Dedup consists of several steps: partitioning the data into chunks, fingerprinting data chunks using a cryptographic hash function such as SHA-1 and MD-5, removing the duplicated data and managing the metadata of dedup. Analyzing dedup potential is to estimate how much data can be removed via dedup. We define a metric to quantitatively measure the dedup potential, referred as dedup ratio.

The dedup ratio is defined as $\frac{capacity_{redundant}}{capacity_{total}}$, which is equal to $1 - \frac{capacity_{unique}}{capacity_{total}}$. For example, a dedup ratio of 25% denotes that 25% of the data could be removed by the dedup and only 75% of the original data capacity would be actually stored.

D. Meister et. al. [4] studied the deduplication potential and found that 20% and 30% of the data in the HPC filesystem was duplicated [4]. J. Kaiser et. al. [5] showed the potential for significant space savings ranging from 37% to 99% [5] in system-level checkpoint data of numerous HPC applications.

3. Danzer: Deduplication Analyzer

Figure 1 shows the design overview of our proposed dedup analysis tool, Danzer. Similar to FS-C [4] tool, Danzer is also composed of two phases, a trace phase and an analysis phase. Dur-

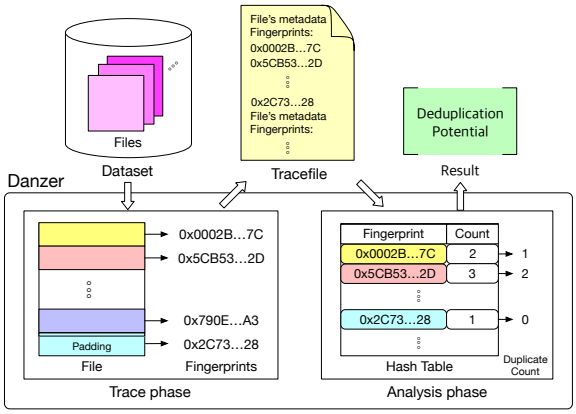

Figure 1: The Architecture of Danzer.

ing the trace phase, Danzer traverses the given dataset where it recursively reads all the files, chunks them based on the chunking parameter, computes the fingerprint and writes it to the *tracefile*. Once the whole dataset is traversed, the second step, analysis phase, is triggered to extract the deduplication insights from the *tracefile*. Each component of Danzer is explained in detail below.

- **Trace phase:** Danzer takes the uppermost directory path as an input parameter and traverses whole files under that directory. Once Danzer starts running, first, Danzer reads data file by file. Second, Danzer partitions a file into a series of chunks and performs fingerprinting on each chunk to deduplicate it. The size of the chunk determines the deduplication accuracy. Chunk size can be configured before the Danzer starts up. In order to perform fingerprinting, each data chunk is put into the SHA-1 hash function to get a 20 Bytes unique fingerprint value. Despite concerns about the hash collision, the probability of it occurring for the SHA-1 algorithm is orders of magnitudes smaller than other sources of data loss such as undetected memory corruption. This series of processes is repeated for all files under the input directory path.
- **Tracefile:** Information about all traced files during *trace phase* is saved in one output file, which is called *tracefile*. In the *tracefile*, metadata and fingerprint information for each data chunk are sequentially recorded for each file. Metadata of the traced file is written on the first line: file name, file size. Below that, the fingerprint values of all chunks from that file are recorded consecutively. Since the information of a given dataset is stored in one single file, it serves as a simple interface between two phases and provides high portability.
- **Analysis phase:** The analysis phase calculates dedup ratio by making use of the result of the trace phase, *tracefile*. First, Danzer reads the fingerprint values of all data chunks under the input directory path from the tracefile. After reading, Danzer constructs a hash table with the fingerprint value as the key and the number of data chunks with that value as the value. Then, based on that hash table, Danzer calculates how much data is redundant among the entire data chunk, and finds the dedup ratio. The execution time of the analysis phase is less than 5% of the *trace phase*. However, if we integrate this task with the *trace phase*, we lose the possibility of parallelizing the *trace phase* due to the shared property of the hash table. By separating this step from the *trace phase*, we leave room to optimize the *trace phase*, which is a bundle of the same tasks, using multithreaded parallelism.

Table 1: Testbed Machine Specifications.

Processor	Intel Xeon Phi 7250 (KNL) processor, 1.4 GHz # of CPU Nodes: 8305, # of Cores per Node: 68
Main Memory	Multi-Channel DRAM (6 channel), DDR4-2400, 16 GB x 6 (= 96 GB) per Node, Bandwidth per CPU: 115.2 GB/s, Total size: 778.6 TB
Parallel File System	Lustre 2.7.21.3, /scratch: 21 PB, /home: 0.76 PB, /apps: 0.5 PB Bandwidth: 0.3 TB/s
OS Kernel	Linux version 3.10.0-1062.el7.x86_64

4. Evaluation Results

4.1 Experimental Setup

We developed Danzer using C++. Danzer was run on a single server and the server's specs are given in Table 1. We used 25 datasets for deduplication study. One dataset is composed of data which was used for 1 HPC application. For each dataset, we have collected the data used by the top 5 users in the criteria of the amount of Nurion usage from June 1, 2021 to May 31, 2022. When there were less than 5 users, data of all users was used as the dataset which might have led to a relatively low amount of capacity.

4.2 File Size Statistics

In order to observe how the file size statistics of a given dataset are related to the dedup ratio and Danzer's execution time, we evaluated the file size distribution and the cumulative file size distribution as seen Table 2. For these results, we present 10%, 25%, 50%, 75%, and 90% percentiles for the number of files and capacity. When $\boldsymbol{x}$ is the median which is the 50% percentile point on the file size distribution, it means that 50% of files are smaller than $\boldsymbol{x}$. For the cumulative file size distribution in the same condition, it means that 50% of the capacity is occupied by files which are smaller than $\boldsymbol{x}$. If there is no file for that condition, we represent it as '-'. We also measured the number of files, capacity and mean file size for each dataset.

First, we observe that in the HPC storage system there are a large number of tiny files while they do not occupy a significant capacity. For most of the datasets, which are 19 out of 25, a fourth of the files are smaller than 8 KB. Specifically in the VASP dataset, even three quarters of files were smaller than 3 KB. In those all 19 datasets, even though a considerable amount of files are smaller than 8 KB, the capacity is occupied less than 10% by them. On the other hand, the largest top 10% files occupied more than 90% of the total capacity except for 3 datasets. In particular, for the Nastran dataset 90% of files were smaller than 50 MB, while the mean file size is 1.74 GB. This is an extreme case where a few huge files occupy more than 90% of the capacity.

4.3 Deduplication Potential

Second, we observe that there is a dedup potential of 24.85% in our data collection, which means that 1.59 TB is redundant among the total 6.4 TB when using 4 KB chunk size. In the case of 8 KB chunk size, the dedup ratio is 24.06% slightly less than the case of 4 KB chunk size, which means that 1.54 TB is redundant among the total 6.4 TB. When we use 4KB chunking, 15 out of 25 datasets show dedup ratio greater than 10%. The highest dedup ratio is 66.57% in MPAS which is the only dataset whose dedup ratio is over 50% and the lowest is 0.51% in Gromacs. We found that the deviation of dedup ratio for the given data is quite large and does not appear close to each other. This suggests that when measuring the dedup potential in the HPC system, there is a possibility of misjudgment by sampling only the data of a few applications.

Furthermore, we also captured the change pattern of the dedup potential when increasing the chunk size from 4 KB to 8 KB. Depending on the size of the data chunk, there was a trade-off be-

Table 2: File size statistics and dedup ratios (DR) of Nurion supercomputer datasets. The dedup ratios were measured both using 4 KB and 8 KB chunking. Two sets of percentiles were measured in 10%, 25%, 50%, 75%, 90%, respectively. Mean FS. and Q.E. denote mean file size and Quantum Espresso.

Dataset	# Files	Capacity	DR (4K)	DR (8K)	Mean FS.	File Size Percentile					Cumulative File Size Percentile				
Abaqus	6,829	385.07 G	40.27%	39.33%	57.74 M	1.29 K	8.58 K	251.99 K	251.99 K	2.13 M	983.25 M	1.55 G	4 G	11.13 G	17.01 G
ANSYS	7,602	544.39 G	19.51%	17.89%	73.33 M	31	366	25.52 K	1.02 M	178.87 M	178.85 M	384.62 M	604.58 M	11.39 G	60.26 G
BWA	2,803	121.75 G	10.50%	9.90%	44.48 M	510	510	510	1.13 K	101.29 K	215.74 M	4.01 G	35.71 G	36.31 G	-
CESM	2,340	273.34 G	24.58%	23.77%	119.61 M	234.29 K	5.63 M	14.81 M	21.87 M	105.34 M	105.34 M	444.48 M	3.67 G	3.67 G	3.67 G
Charmm	4,936	381.56 G	17.74%	15.80%	79.16 M	424	5.08 K	176.75 K	3.81 M	10.88 M	1.94 G	3.41 G	5.79 G	6.62 G	9.87 G
Gaussian	2,549	293.72 G	33.23%	33.07%	117.99 M	0	754	4.43 K	4.78 M	88.22 M	137.15 M	4.11 G	21.35 G	33.61 G	33.61 G
grims	4,293	322.65 G	8.36%	8.30%	76.96 M	1.05 K	3.04 K	15.97 K	1018.04 K	1018.04 K	3.18 G	68.09 G	218.84 G	-	-
Gromacs	4,278	231.4 G	0.51%	0.50%	55.39 M	143	24.18 K	1.74 M	12.17 M	45.78 M	45.78 M	312.47 M	1.22 G	1.27 G	1.28 G
in-house	8,847	172.35 G	0.73%	0.15%	19.95 M	261	1.22 K	6.22 K	20.97 K	157.23 K	494.25 M	494.25 M	494.25 M	10.19 G	12.72 G
LAMMPS	5,085	23.24 G	14.93%	14.67%	4.68 M	909	11.25 K	18.86 K	845.75 K	6.8 M	8.81 M	27.87 M	74.99 M	120.18 M	166.86 M
MOM	2,428	322.32 G	41.80%	39.53%	135.94 M	101	1.13 K	8.83 K	73.52 K	6.72 M	978.7 M	2.93 G	5.69 G	28.47 G	60.26 G
MPAS	10,829	219.06 G	66.57%	62.05%	20.71 M	752	2.46 K	24.68 K	1.37 M	1.37 M	15.96 G	15.96 G	15.96 G	15.96 G	42.19 G
msc	322	190.65 G	1.91%	1.88%	606.28 M	361	25.39 M	244.14 M	1.82 G	1.82 G	244.14 M	1.82 G	1.82 G	1.82 G	1.82 G
NAMD	13,049	2.79 G	5.55%	3.00%	223.85 K	443	1.35 K	4.27 K	21.73 K	107.92 K	283.86 K	9.05 M	380.25 M	389.3 M	389.3 M
Nastran	125	217.3 G	6.66%	6.34%	1.74 G	307	3.5 K	177.47 K	10.69 M	45.85 M	19.09 G	195.15 G	-	-	-
OpenFoam	32,900	31.8 G	4.15%	4.09%	1013.48 K	879	1.15 K	4.5 K	258.51 K	3.08 M	1.21 M	6.38 M	6.6 M	15.99 M	1.65 G
Pytorch	9,239	472.51 M	2.81%	2.52%	52.37 K	145	459	1.3 K	7.24 K	37.01 K	71.26 K	799.79 K	1.65 M	1.85 M	9.79 M
Qchem	7,865	319.67 G	28.13%	27.02%	41.62 M	0	167	1.82 K	110.66 K	7.06 M	287.71 M	10.68 G	228.51 G	-	-
Q.E.	10,661	199.93 G	9.98%	9.91%	19.2 M	248	2.27 K	102.96 K	3.5 M	17.58 M	17.62 M	579.5 M	579.64 M	579.72 M	2.1 G
QMCpack	1,793	5.61 G	0.90%	0.83%	3.2 M	61	1.38 K	8.61 K	109.64 K	1.15 M	13.22 M	40.03 M	121.37 M	130.17 M	377.47 M
RAMSES	2,280	12.4 G	23.79%	17.31%	5.57 M	27.4 K	4 M	4 M	4 M	4 M	4 M	4 M	5.19 M	60.8 M	70.6 M
ROMs	18,017	318.54 G	12.33%	12.32%	18.1 M	62	234	4.86 K	25.92 K	29.91 M	77.7 M	113.33 M	549.52 M	4.95 G	8.62 G
SIESTA	8,118	565.6 G	45.58%	45.54%	71.34 M	72	1.04 K	13 K	209.65 K	790.34 K	2 G	5.02 G	7.84 G	11.86 G	45.64 G
VASP	158,369	1.05 TB	22.39%	22.38%	6.92 M	0	13	618	2.85 K	147.92 K	49.33 G	49.33 G	49.33 G	49.33 G	49.33 G
WRF	16,352	326.61 G	42.51%	41.18%	20.45 M	212	287	9.08 K	1.37 M	3.16 M	1.11 G	1.11 G	1.79 G	1.79 G	34.19 G

Table 3: Dedup ratio when using 8KB chunking of 18 HPC datasets measured in 2012 [4].

Dataset	Capacity	DR (8K)	Dataset	Capacity	DR (8K)
BSC-BD	11.2 TB	7.0%	DKRZ-B5	75.3 TB	29.5%
BSC-MOD	20.1 TB	21.3%	DKRZ-B6	47.9 TB	22.5%
BSC-PRO	9.1 TB	29.3%	DKRZ-B7	65.2 TB	14.1%
BSC-SCRA	3.0 TB	38.3%	DKRZ-B8	176.6 TB	13.9%
DKRZ-A	27.0 TB	17.9%	DKRZ-K	42.9 TB	49.3%
DKRZ-B1	114.5 TB	19.7%	DKRZ-M1	134.5 TB	15.0%
DKRZ-B2	109.1 TB	27.6%	DKRZ-M2	116.8 TB	21.1%
DKRZ-B3	126.5 TB	74.4%	RENCI	11.1 TB	23.8%
DKRZ-B4	68.3 TB	27.1%	RWTH	53.8 TB	23.2%

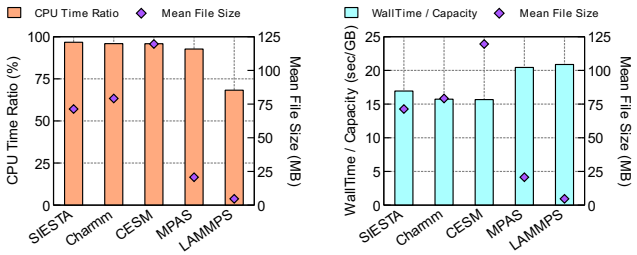

Figure 2: (Left) CPU time ratio when running Danzer's trace phase with mean file size of 5 datasets. (Right) Wall time of trace phase per capacity with mean file size of 5 datasets.

tween the analysis execution speed and the dedup ratio derived. As the chunk size increases, the amount of chunking and fingerprinting work decreases, which has the advantage of reducing the execution time for analysis. However, since the unit size of chunking has become coarser-grained from 4 KB to 8 KB, the dedup ratio decreases slightly as seen in Table 2.

4.4 Execution Time Measurement

We show an analysis of Danzer's execution time. Our experiment was conducted using a single core in Nurion's KNL computing node.

First, the *analysis phase* of Danzer's two phases has relatively little execution time, so the effect on Danzer's overall running time is insignificant. This is because the *analysis phase* has a time complexity of *O(n)* using a hash table in calculating the dedup ratio. We found that the execution time of the analysis phase is within a couple of minutes at most, and it is between 1% and 5% of the total running time.

Second, we clearly observe that running Danzer's *trace phase* is a computationally intensive task, as shown in Figure 2. This is because a significant amount of computation is required for fingerprinting using the SHA-1 hash function. We found that the ratio of **cpu-time** to **wall-time** (*CPU time ratio*) is over 90% for datasets with the mean file size of 20 MB or more among the five sampled datasets. Specifically, the LAMMPS dataset has a mean file size of 4.68 MB, which is very small compared to the other four datasets as shown in Figure 2, which leads to a relatively low *CPU-ratio*, 68.26%. This shows that the smaller the mean file size, the greater the overhead of the file open and close operations.

The above observations suggest the motivation that the *trace phase*, the bottleneck of the Danzer process, needs to be run by exploiting multi-nodes or multi-core parallelism.

5. Conclusion

In this paper, we found a high deduplication potential on the Nurion supercomputer. Specifically, when setting the chunk size to be 4 KB, the dedup ratio is 24.85% which is exactly in the same range with a decade ago study, surprisingly. Based on our observation, we claim that dedup services on HPC environments will save storage space and cost by a significant amount in petascale storage as well.

References

[1] J.-K. Lee, S.-J. Kim, and T. Hong, "Analysis of traffic and attack frequency in the nurion supercomputing service network," *KIPS Transactions on Computer and Communication Systems*, vol. 9, no. 5, pp. 113–120, 2020.

[2] "Top500 supercomputing sites, howpublished = https://www.top500.org, note = Accessed: 2022-10-28."

[3] P. J. Braam and P. Schwan, "Lustre: The intergalactic file system," in *Ottawa Linux Symposium*, vol. 8, pp. 3429–3441, 2002.

[4] D. Meister, J. Kaiser, A. Brinkmann, T. Cortes, M. Kuhn, and J. Kunkel, "A study on data deduplication in hpc storage systems," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–11, IEEE, 2012.

[5] J. Kaiser, R. Gad, T. Süß, F. Padua, L. Nagel, and A. Brinkmann, "Deduplication potential of hpc applications' checkpoints," in *Proceedings of the 2016 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 413–422, IEEE, 2016.