International Journal of Electronics Engineering (ISSN: 0973-7383)
Volume 10 • Issue 2 pp. 921-926 June - Dec 2018 www.csjournals.com

Data Deduplication Testing Tool: Destor

Naresh Kumar
Department of Computer Sc. and Engg, UIET, Kurukshetra University, Kurukshetra, INDIA

Abstract: Data redundancy is becoming big challenge for the organizations. Many researchers are doing research to provide better solutions for storing unique data contents by removing the duplicate data. Data deduplication is very important aspect for storing unique non-redundant data and to maintain data integrity with consistency. There are different deduplication chunking techniques like whole file chunking, fixed size chunking and content defined chunking. Destor tool provides a platform to evaluate the performance of these chunking techniques. In this paper configuration of Destor tool is described with performance matrices. This tool can be installed on Linux 64 bit system. To install the Destor tool there are some dependent packages those are needed to be installed on the Linux 64 bit system. These dependencies are also discussed with Destor tool installation commands. Different deduplication techniques are also tested with the help of Destor tool. Experimental results on different datasets are also presented on whole file chunking, fixed size chunking and content defined chunking.
Keywords: Data deduplication, chunking, fixed-size chunking, variable-size chunking, Destor.

1. Introduction

Today the growth of digital data in computer world is the cause of problem of inefficient storage space utilization. In this big world of data, data comes from variant sources in variant forms. An individual computer, tablet, mobiles or servers are the different data sources producing data in structured (relational data), semi-structured (XML data) and in unstructured (PDF, media logs, word, text) forms. This increase in digital data motivates storage optimization. "Storage Optimization" is a term that indicates to any wise technique which reduces the space utilization required to pile up a dataset. These techniques have a worthy role to play in enhancing data storage efficiently.

Deduplication [1] also known as Single instance storage (SAS) is a lossless technique that has become very convenient in large scale storage systems for space optimization. It eliminates the redundant data by physically storing only data that is unique. It reduces storage capacity effectively by storing only one copy. Deduplication is performed in three ways named as Inline, Post process, Concurrent either on source side or target side [2]. Inline method is implemented in memory before it is sent to the target unlike the Post process where the entire data backup reaches the target memory. The further classification of source deduplication [3] is based on deduplication granularities i.e Local chunk level and Global chunk level deduplication. In local chunk level, the elimination of duplicate chunks is performed before transferring them to backup destination within the same client. In global chunk level, the removal of redundant chunks is performed globally across various clients. Concurrent method comes in execution while data is being ingested to the target.

2. Deduplication Techniques

There are various data deduplication techniques including content defined chunking, static fixed size chunking and whole file chunking. Though static fixed size chunking is robust and efficient, it has a limitation of "boundary shift problem". Content Defined Chunking (CDC)[4] approach of data deduplication achieve high deduplication ratio, but it is too much time consuming as compared with the other approaches. CDC prevents the boundary shift problem of the static chunking approach by partitioning the input data stream according to the contents of the data but not to the local boundary. A CDC based network file system LBFS [5] eliminates redundant data in low bandwidth networks. It is the first file system that partitions data stream into variable sized chunks by finding proper cut point for each chunk. Chunking techniques like Sliding Window approach,Two Thresholds Two Divisor (TTTD) [6] and Asymmetric Extremum (AE) content defined chunking [7] provide much more improved results with variable sized chunks.

1. Destor Tool

Destor deduplication tool is an open source tool which is freely available at GITHUB [8]. Destor is basically a platform which provides performance evaluation of data deduplication techniques. This Destor tool was developed byMin Fu [8]. This tool can run on 64-bit linux operating system.

Instructions for installing the Destor tool : First download library package Glib [9] on Linux 64-bit (Ubuntu 14.04) operating system. At next step, install packages those have dependencies on Glib Zliblg-dev, libffi-dev, libssl-dev, autotools-dev and automake. These packages are required to be installed on linux operating system for making Glib in working state.

These are the steps for installing Destor tool on Linux 64 bit system

Step 1 and Step 2 for installing dependencies which are required to run the Destor tool.

```
Step 1: sudo apt-get install zlib1g-dev
Step 2: sudo apt-get install libffi-dev
```

Download the Glib first to install the Destor. Glib is the library which is required to run the Destor tool. Now change the directory where Glib is downloaded and then run these following commands to install Glib on the system.

```
Step3: cd [PATH_TO_GLIB]
Step 4: ./configure
Step5 : make
Step 6: sudo make install
```

Now run these commands to copy the required Glib files into the /include directory.

```
Step 7: cd /usr/local/
Step 8: sudo cp include/glib-2.0/* include/
Step 9: sudo cp lib/glib-2.0/include/glibconfig.h include/
Step 10: cd lib
Step 11: sudo link libglib-2.0.so libglib.so
```

Now it is required some more dependencies packages for running the Destor tool properly. Run these commands on Terminal for installing these packages.

```
Step 12: sudo apt-get install libssl-dev
Step 13: sudo apt-get install autotools-dev
Step 14: sudo apt-get install automake
```

All packages are installed for Destor. Now change the directory where Destor is downloaded and run the following commands for installing Destor on the system.

```
Step 15: cd [DIR_OF_DESTOR]
Step 16: ./configure
Step 17: automake --add-missing
Step 18: make
Step 19: sudo make install
```

2. Working of Destor Tool

Destor can be run by using following commands:

1. sudo destor /path/to/data/ -p
   this command is used to create backup for the specified data.

2. sudo destor r<JobId> /path/to/store –p
   This command is used to restore the backed up data.

3. destor –h
   this command can be executed for the help.

4. sudo destor –t /path/to/data/
   this command is used to create a trace for the data on which backup job is executed.
   There are various techniques in destor tool like file level chunking, fixed size chunking, variable sizeRabin CDC, TTTD, and AE. These techniques can be executed by modifying destor.config file. This destor.config file should be placed in working directory of the destor.
   Working directory: /home/data/working.
   In destor.config there are various chunking techniques; by default it runs rabin based cdc technique for backup job. To run different techniques using Destor tool, remove hash(#) from the technique name and save file as shown in Fig. 1.

```
# Specify the chunking algorithm.
# It can be rabin, fixed, "normalized rabin", tttd, file, and
ae.
# "file" indicates an approximiate file-level deduplication.
# For example,
# chunk-algorithm "normalized rabin"
# chunk-algorithm file
# chunk-algorithm fixed
# chunk-algorithm tttd
# chunk-algorithm ae
chunk-algorithm rabin

# Specify the average/maximal/minmal chunk size in bytes.
#chunk-avg-size 4096
chunk-avg-size 4096
chunk-max-size 65536
chunk-min-size 512
```

Fig. 1: destor.config file

3. Testing on Destor Tool

Testing the performance of different deduplication techniques on various datasets, it has been installed with Destor tool on Dell Inspiron i5 having 500 GB Hard Disk and 4 GB RAM using Ubuntu 14.04 Operating system. Experimental results are presented for whole file chunking, fixed size chunking and variable size chunking.

Experimental Results:

In this section, it has been tested with the performance on structured, semi-structured and unstructured datasets using Destor [8] tool. Data deduplication is tested at file based, fixed size blocks based and variable sized chunks based. When the duplicity is checked over multiple files, the entire file is treated as single one and then duplicate data filesare removed to maximize storage space. In fixed size chunking the file is partitioned into fixed size blocks and

redundancy is removed. Finally, the variable size chunking that partitions the datastream according to the contents with variable size chunks. It has performed the testing on a machine with configuration Dell Inspiron i5 CPU with installed memory 4GB on 64bit operating system in Ubuntu version 14.04. The testing performed on different datasets with the help DESTOR tool. Table 1. shows result on various datasets with different deduplication techniques and results are shown graphically from Fig. 2 to Fig. 4.

Input Datasets	Deduplication Techniques	Before Dedupe Input Data Size(Bytes)	After Dedupe Output Data Size(Bytes)
Semi-structured	File	4164058712(4.16GB)	736723424(0.73GB)
Semi-structured	Fixed	4164058712	219059884
Semi-structured	Rabin CDC	4164058712	367212926
Semi-structured	TTTD	4164058712	100698326
Structured	File	2007217590(2.007GB)	577872144(0.57GB)
Structured	Fixed	2007217590	118842424
Structured	Rabin CDC	2007217590	197072522
Structured	TTTD	2007217590	103982642
Unstructured	File	2870495530(2.87GB)	594362064(0.59GB)
Unstructured	Fixed	2870495530	455055824
Unstructured	Rabin CDC	2870495530	296722216
Unstructured	TTTD	2870495530	256145918

Table 1: Experimental results of different deduplication techniques on DESTOR tool

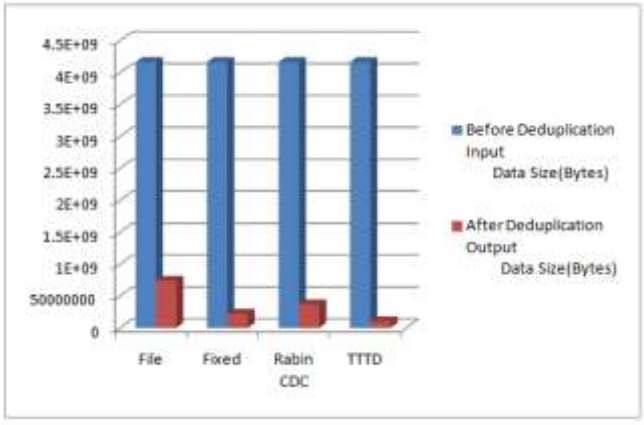

Fig. 2: For Semi-structured data

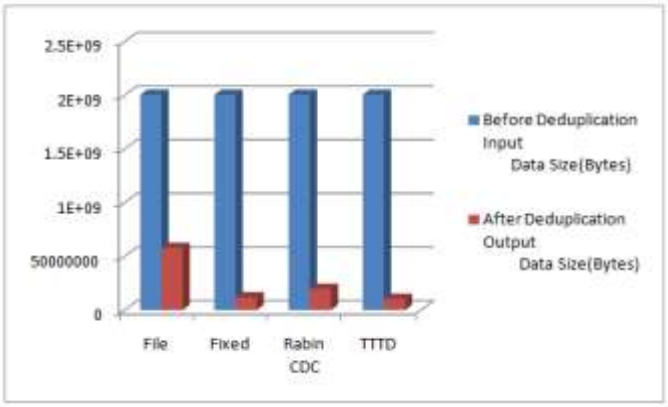

Fig. 3: Results for Structured Datasets

International Journal of Electronics Engineering (ISSN: 0973-7383)
Volume 10 • Issue 2 pp. 921-926 June - Dec 2018 www.csjournals.com

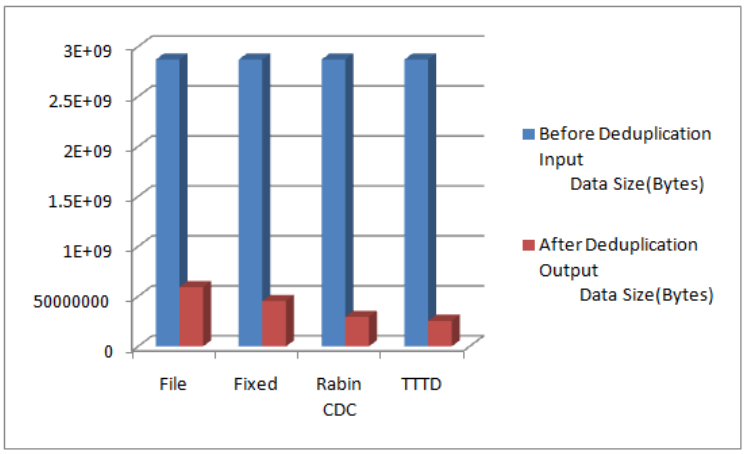

Fig. 4: Results for Unstructured Datasets

When job is executed then backup.log, trace file and restore.log file is created. Backup.log contains information like job id, original size of the file and other parameters like deduplication ratio, total time etc. Trace file contains file name and its hashes and the size of the chunks created by the selected technique. Restore.log file contains job id which can be used to restore the original data with the help of restore command. Experimental snapshots from DESTOR tool is shown from Fig. 5 to Fig 7 for backup.log, inputdata.trace and restore.log files. It is mandatory to move old backup.log and restore.log files from /home/data/working/ to any other location in hard disk to test new datasets as new backup.log and restore.log will be created for new input datasets, otherwise, deduplication results for new datasets will not be generated.

```
26 3123997744 303947041 1.0000 0.0000 8 5 4 7 1 1 7 99.02
27 3123997744 303947041 1.0000 0.0000 8 5 0 8 0 0 8 118.37
28 2705124491 303952514 1.0000 0.0000 8 5 4 7 1 1 7 85.59
29 2844723280 303957160 1.0000 0.0000 8 5 4 7 1 1 7 90.29
```

Fig. 5: Backup.log file

```
file start 16
1 (9th copy).txt
87B0322C6A30E79F34A998A96597FA61D7500F11 4646
B0A8684ADD3FD470F89B8DEBA61F7FF971200E8B 5507
1EC87FE6F0949C41DFEC8E0D7A5EAA1DF5EE40AE 29941
9AFF615807 10C281A12C5F2B6A4B5B6A0FF6AD8A 1630
A774C4F9FAA9CE53F3716F7280D85F8F04269C99 2271
01761131BE5017EEC46A34473D4042CD8E1D2283 6414
9C16FFA6ED85C139C24A5B3F8D5A1DEE29CAF392 21301
A8CD2E8483E278863283973AC9867B888F5BE807 4518
E87DBE1F046762C6DB1EF4CE2D96F486B4C75F39 2449
2731578E097D88FDAE6030CAAFCB899A21C23159 7885
C82305F25647EB8D38C6A46682B0E7CD0577E47B 5423
08D04FB6B17CFFE68A5A1B551A32981120B98129 5259
46E11510F4233443D6812F5A075C5C3739B74C7C 3902
8B0AA125F73109DF83F62411ABF9FF1B1E10A670 8914
5655C0C8F3233F16BB945A3BA953118D3C292A41 8635
FF2B1AD04B7292ED80153BF2D402F8252FC96C8B 9882
05BE49F0EDB675D2D5877622991A10098819BE9C 7727
4467189FF49A0D0784E2ADA15B1A61E456232627 23111
0ECEC8C093A57856B85684B2A5FAC685A9BFD732 2074
6CDCA41D9EFB3187602AD26E94D09E58C1294C1A 4506
02C208E90169F35556D6D6C4208D0ABBF5BAA107 9970
25C23C2F6E781AF0B7B3DC9DC69D8A3BA16C8209 8105
AECE1740905 5B75BBE674EF63A371D2BD6702676 1152
D61F3623FD2221DA8F3DFB23380768B7EF365F02 3291
7DFFC34F62D728830 69C35D4DFA7204D42481202 14611
F41C5F1178B2FF4FA6B9E14D4ECD37963D981A6D 15521
F29CB675D24A597F59A5ECD05903 5B11255DC6E7 5270
AC4275 0EAA9BBACF713C81B7480C995298A1F109 11576
```

Fig. 6: Trace file

```
29 2844723280 8 339.1174 67.8217
```

Fig. 7: Restore file

```
job id: 8
backup path: /home/rahul/Desktop/dataset/
number of files: 18
number of chunks: 208963 (9132 bytes on average)
number of unique chunks: 618
total size(B): 2007217590
stored data size(B): 577872144
deduplication ratio: 1.0000, 330.2293
total time(s): 15.011
throughput(MB/s): 121.24
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 2.327s, 782.11MB/s
chunk_time : 11.933s, 152.52MB/s
hash_time : 11.539s, 157.72MB/s
dedup_time : 0.324s, 5621.54MB/s
rewrite_time : 0.017s, 109271.19MB/s
filter_time : 0.377s, 4824.41MB/s
write_time : 0.015s, 121741.37MB/s
```

Fig. 8: Results from File based Technique.

4. Conclusion

In this paper, it has been tested with different techniques for data deduplication. For net deduplication performance and increased throughput, an effective chunking technique is required. This deduplication testing tool will help researchers to select an appropriate and feasible technique for their work in deduplication. Table 1. conclude basic differences between various deduplication techniques. In future, I will work to design testing tool for deduplication of distributed storage systems for Big Data.

7. References:

[1] Jaehong Min, Daeyoung Yoon, and Youjip Won, "Efficient De-duplication Techniques for Modern Backup Operation", IEEE Transactions on Computers, Vol. 60, No. 6, pp. 824-840, 2011.

[2] D. Harnik, B.Pinkas, A-Shulman-Peleg, "Side channels in cloud services, the case of deduplication in cloud storage", IEEE Journal of Security Privacy, No. 8, pp. 40-47. 2010.

[3] Zhu, B., K. Li and H. Patterson, "Avoiding Disk bottleneck in the data domain deduplication file system. Proceedings of the 6th USENIX Conference on File and Storage Technologies, February 26-29, San Joes, CA. USA., pp: 269-282, 2008.

[4] Kave Eshghi and Hsiu Khuern Tang,"A Framework for Analyzing and Improving Content-Based Chunking Algorithms", Hewlett Packard Development Company and Intelligent Enterprise Technologies Laboratory, Feb 2005.

[5] Athicha Muthitacharoen, Benjie Chen, and David Mazieres, "A Low-Bandwidth Network File System" ,ACM SIGOPS Operating System Review, vol. 35, no. 5, pp. 174- 187,2001.

[6] Jyoti Malhotra and Jagdish Bakal,"A Survey and Comparative Study of Data De-duplication Techniques", IEEE International Conference on Pervasive Computing (ICPC), pp. 1-5, 2015.

[7] Y. Zhang, H. Jiang, D. Feng and Y. Zhou, "AE: An Asymmetric Extreme Content Defined Chunking Algorithm for Fast and Bandwidth-Efficient Data Deduplication", IEEE Conference on Computer Communications (INFOCOM), pp. 1337-1345, 2015.

[8] Destor, https://github.com/fomy/destor Accessed on 8-November-2018.

[9] Glib, http://ftp.gnome.org/pub/GNOME/sources/glib/2.49/, Accessed on July-2018.