Evaluation of FastCDC: The Effects of Chunk Sizes on Deduplication and Throughput

Miguel Alonso
Virginia Tech

Matthew Donlon
Virginia Tech

I. INTRODUCTION

Data deduplication is a technique used in storage systems to reduce data redundancy. It has been widely used by large storage cloud providers due to the drastic increase in digital data. It works by identifying and removing duplicate data blocks, leaving only unique data to be stored. Each block is identified by a cryptographically secure hash signature, which is used to identify identical blocks. This can be achieved at two granularities, at the file or chunk level, each indicating the size of the block it will be considering. The main difference between these two approaches is flexibility and performance overhead. When considering larger blocks, this will then induce better performance due to the reduced number of comparisons. On the other hand, variable size chunking considers different-sized chunks, which will inherently identify more chunks to be deduplicated but will have a much larger performance overhead.

A. Chunking

There are two main categories of chunk-level deduplication: Fixed-Size Chunking and Content-Defined Chunking. Both provide deduplication benefits in different scenarios through performance and complexity. Fixed-Size Chunking is the simplest approach, which cuts the file or data stream into fixed-size chunks. Content-Defined Chunking was introduced to fix the problems that arise from Fixed Size Chunking. These problems mainly include the boundary-shift problem, such as when a small number of bytes have been modified at the beginning of the chunk. Content-Defined Chunking declares chunk boundaries around the contents of the data stream instead of a fixed interval. Since it is proactively defining chunk sizes, there is a greater chance to detect data redundancies. According to recent studies [4], Content Defined Chunking is able to deduplicate 10–20% more data than Fixed Size Chunking.

There are many current popular Content Defined Chunking approaches, such as Rabin, Gear, QuickSync, SampeByte, and many others. The different approaches leverage different customizations built for different workloads. Some are meant to take advantage of GPUs, while others use optimized chunking.

B. FastCDC

The CDC implementation we will be considering in this paper is FastCDC. It was described in "FastCDC: a Fast and Efficient Content-Defined Chunking Approach for Data Deduplication" by Wen Xia, Yukon Zhou, Hong Jiang, Dan Feng, Yu Hua, Yuchong Hu, Yucheng Zhang, and Qing Liu. They represent the School of Computer, Huazhong University of Science and Technology, Sangfor Technologies Co., Ltd, the University of Texas at Arlington, and WNLO, Huazhong University of Science and Technology.

FastCDC is implemented on top of the Gear-based Content Defined Chunking with a focus on performance. It has three significant improvements that contribute to its performance. Firstly, it enhanced hash judgment by implementing a zero-padding scheme to enlarge the sliding window side when calculating different chuck hashes. Secondly, it ignores a minimum chunk size boundary that many other algorithms have. This increases chunking speed but has the adverse effect of decreasing the deduplication ratio due to smaller chunks. Thirdly, it introduces normalized chunking, which reduces the range of chunk sizes. Therefore, by eliminating the outlier chunk sizes, it removes the negative effect of removing the minimum chunk size. This increases deduplication since chunks have similar sizes. Overall, these improvements have shown 10x faster chunking than Rabin-based Content Defined Chunking algorithms with nearly the same deduplication ratio. Further, it is 3x faster than Gear which is claimed to be the most efficient approach.

C. Contributions

In this paper, we wish to evaluate FastCDC over a variety of files and workloads to analyze its deduplication ratio and throughput.

- How does the performance of FastCDC compare in terms of deduplication ratio and chunking speed over multiple workloads?

We hope to provide helpful insights into how defining chunk ranges affects deduplication. This can be used as a valuable evaluation of different workloads to determine what chunk sizes are best per file type. Eventually, when memory deduplication is implemented in hardware, researchers can reevaluate this study to determine if our evaluation was accurate.

II. METHODOLOGY

The data was collected from a variety of sources that we deemed would aid our evaluation of FastCDC. These include files, images, videos, and memory dumps, which cover a broad range of data that is common in large-scale storage systems. We were also interested in compression, which led us to evaluate our compressed images and videos. A further breakdown of each workload can be read below.

A. Videos

We consider a YouTube video at different resolutions to determine if there is any duplication between them. The video, "Conan Makes NYC Pizza — CONAN on TBS", was posted on August 22, 2019 to the Team Coco YouTube Channel. The video has a total length of 4 minutes and 48 seconds. There are four videos that each have a different resolution: 240, 360, 480, and 720. The total amount of space these videos take up is 63 MB. These files were chosen to analyze the impact of differing video resolutions on data deduplication.

B. Textbooks

We chose different versions of a large textbook to determine if, through different versions of similar content, there were any impacts on deduplication. The textbook we chose was "Computer Architecture: A Quantitative Approach" by John L. Hennessy and David A. Patterson. We acquired the 1990s Edition, the 3rd Edition, the 4th Edition, the 5th Edition, and the 6th Edition. The total amount of space these textbooks take up is 74.2 MB. These files were chosen to analyze the impact of evolving textbook editions on data deduplication.

C. Images

There are lots of images on the internet that take up lots of space on personal computers and in the cloud. We were curious to see the impact of deduplication on pictures that share common features. Therefore, we selected an image dataset that contained images of vehicles. The total amount of space these images take up is 351.6 MB. These files were chosen to analyze the possibility of data deduplication on similar images.

D. Compressed Images

Many times, users may have many images or videos on their devices and wish to save space. Therefore, they may turn to compression to solve their problem. This workload consists of compressed images and uncompressed images. This was chosen to investigate whether, after compression, there may still be a possibility for deduplication. The total amount of space for the compressed images and regular images is 703.0 MB. These files were chosen to analyze the possibility of data deduplication between compressed and decompressed images.

E. Compressed Videos

This evaluation choice follows the compressed image approach described above. The differences include the type of data being considered, which is the video described above as well. The total amount of space for the compressed images and regular images is 125.5 MB. These files were chosen to analyze the possibility of data deduplication between compressed and decompressed videos.

F. Memory Dumps

Virtual machines are very prevalent in large-scale storage systems and data centers, as they allow users to run different environments on shared hardware. Reducing redundancy on such a large scale could dramatically improve memory efficiency. We wished to evaluate deduplication across different C programs. These memory dumps have had their all-zero pages removed to ensure correctness. The total amount of space for the C workloads is 1.5 GB. These files include performance benchmarks such as perlbench, omnetpp, fluidanimate, and freqmine. These files were chosen to analyze the possibility of data deduplication between different C workloads.

G. Approach

The FastCDC deduplication algorithm provides multiple metrics for evaluation. It provides information such as chunk sizes, unique chunks, total data, dupe data, dedupe ratio, and throughput. The two we deemed most important were deduplication ratio and throughput, which will be what our evaluation is based upon. The deduplication ratio is recorded as a percentage out of 100

The implementation we used in our evaluation has many customizable features. The features we focused on focused mainly on chuck sizing. We were able to set specific bounds for chunking, such as the minimum, average, and maximum allowed chuck sizes. These will set hard bounds so that we can control what will be considered a chunk.

Firstly, we considered a fixed chunk size for all workloads. This served as a baseline evaluation for our other methods. Since it is relatively the same as fixed-size chunking, it is not taking advantage of adjusting the block size to the data. Secondly, our next approach included setting ranges for each base chunk size. There were three ranges we evaluated: 2048, 4096, and 8192 bytes. For example, if we are considering a base chunk size of 8192 bytes and a range of 4096 bytes, The possible chunk sizes range from 4096 to 12288 bytes. This does not indicate what the average chunk size should be. As the byte range increases, so does the number of chunks it is considering and its impact on performance. These were chosen to show the benefits and limitations of smaller and larger ranges.

Therefore, over these different ranges of chunk sizes and their comparison against the fixed chunk sizing, we are able to identify the impact the base chunk size has on the deduplication ratio and throughput.

III. RESULTS

The FastCDC implementation was open-sourced and published on GitHub [5]. These graphs are sectioned by file type into their visualization of deduplication and throughput over a range of chunk sizes. As described above, the range of chunk sizes indicates the spread of available chunk sizes from the base chunk size on the x axis. Each graph contains the labels: fixed, 2048, 4096, and 8192 bytes. These indicate the range of chunk sizes, which is why they are distinguished separately.

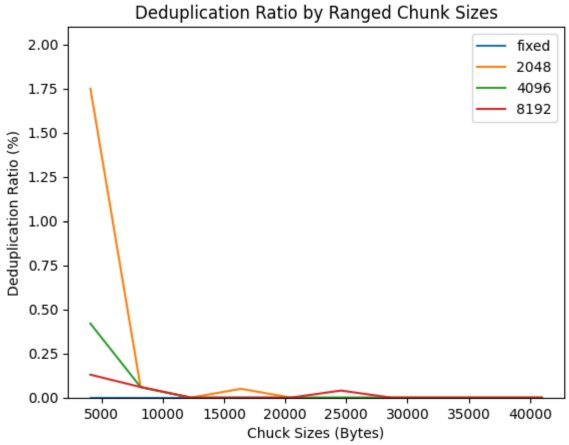

Fig. 1: Video Deduplication

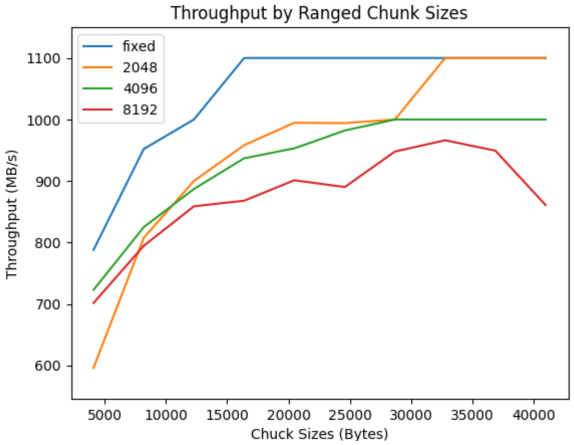

Fig. 2: Video Throughput

A. *Videos*

The deduplication ratio of our video files benefits from chunk sizes between 4096 and 8192 bytes. After 10,000 bytes, it can be seen that there is no increased benefit. The highest deduplication is attributed to chunk size 2048. Even though the fixed chunk size had the greatest throughput, it had the worst deduplication.

B. *Textbooks*

The deduplication ratio of our textbook files benefits from chunk sizes between 4096 and 10,000 bytes. It can be seen that there is a gradual decrease up to 35,000 bytes, which is in contrast to the sharp decline of our video files. The highest deduplication is attributed to the ranged chunk size of 2048 with smaller chunk sizes, where the ranged chunk size of 8192 overtakes large base chunk sizes. This may be due to large sections of each textbook containing similar data, which allows for higher deduplication at larger chunk sizes.

C. *Images*

The deduplication ratio of our image files benefits from all tested chunk sizes. It can be seen that there are similar ratios between all ranges of chunk sizes up until 20,000

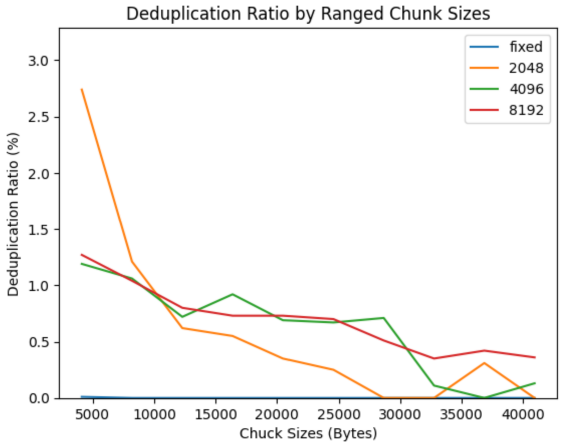

Fig. 3: Textbook Deduplication

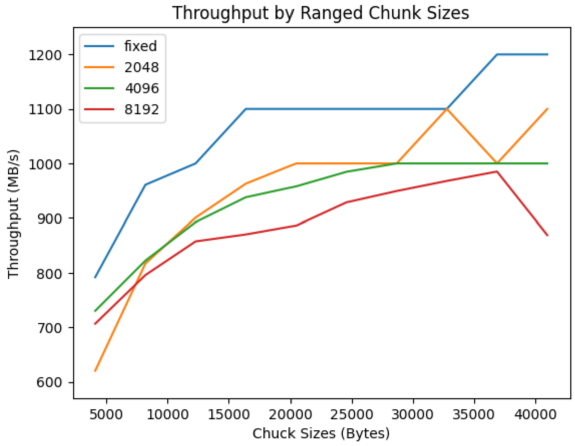

Fig. 4: Textbook Throughput

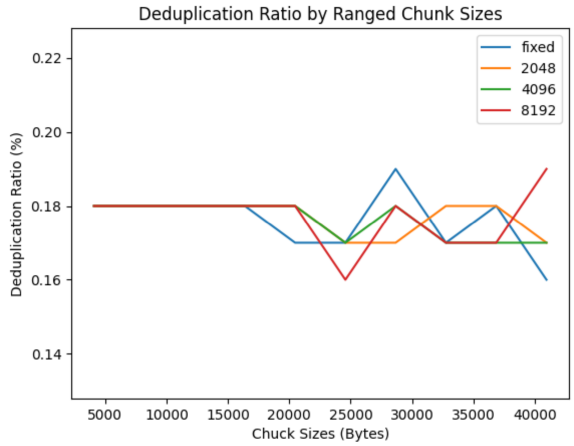

Fig. 5: Images Deduplication

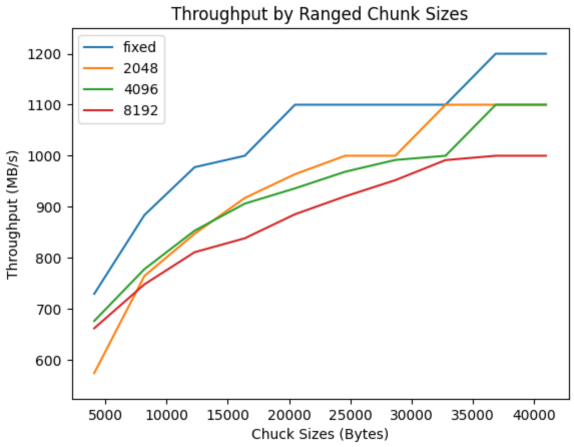

Fig. 6: Images Throughput

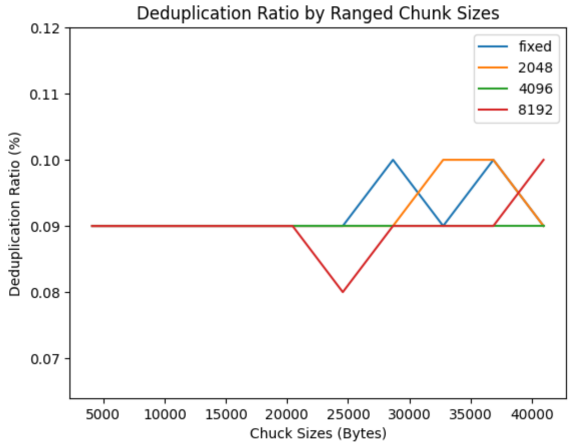

Fig. 7: Compressed Images Deduplication

bytes. This may be due to images being very large files, and the advantages of larger chunking benefit the range of 8192 around 40,000 bytes.

D. Compressed Images

The deduplication ratio of our compressed images and regular image files benefits from all tested chunk sizes. It can be seen just as in the Figure 5 and 6 that there are similar ratios between all ranged chunk sizes up to 20,000 bytes. Additionally, the deduplication ratio has halved, which indicates that there was minimal additional deduplication between compressed and regular files.

E. Compressed Videos

The deduplication ratio of our compressed videos and regular video files benefits from base chunk sizes before 20,000. It can be seen just as in the Figure 1 and 2 that their deduplication ratio only topped out at 1.75ore 20,000. It can be seen just as in the video graphs [!!!!!!!] that their deduplication ratio only topped out at 1.75%, while this graph indicates 30%. This shows that deduplication may benefit from compressed and regular videos.

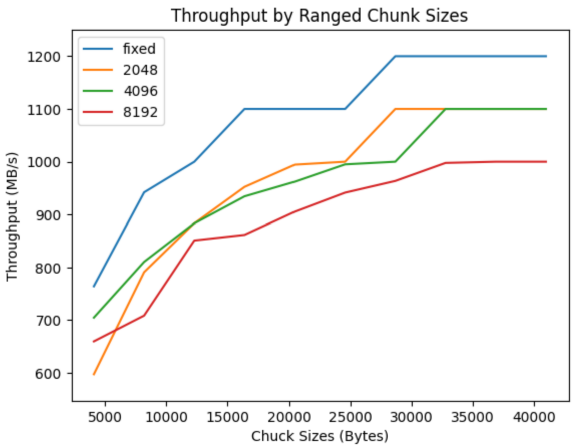

Fig. 8: Compressed Images Throughput

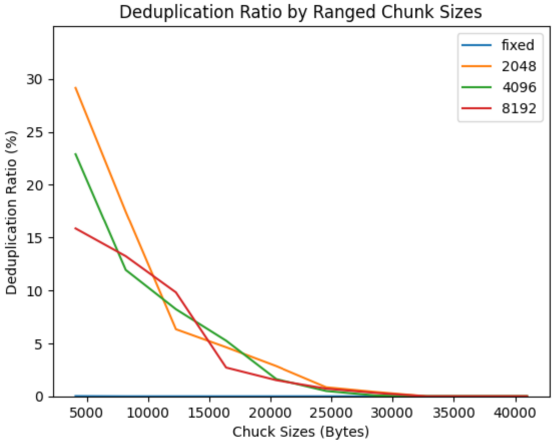

Fig. 9: Compressed Videos Deduplication

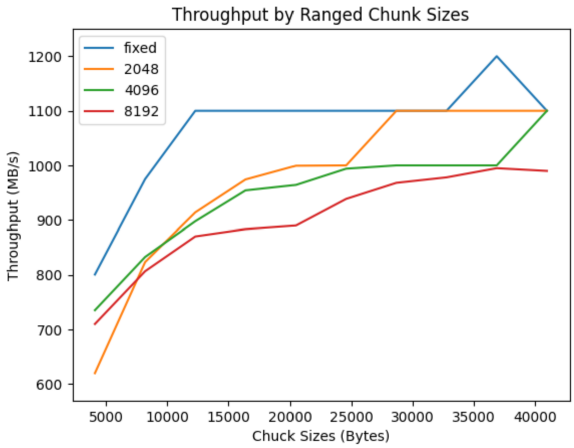

Fig. 10: Compressed Videos Throughput

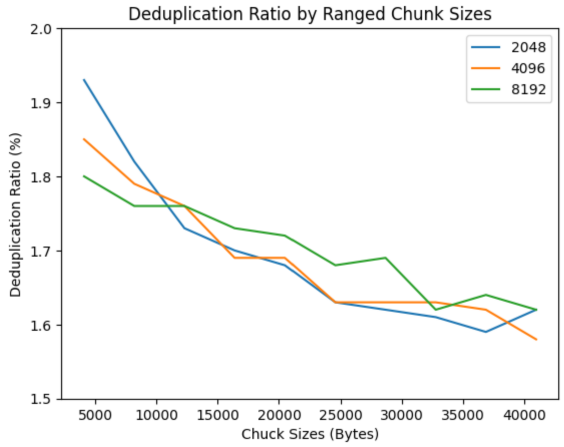

Fig. 11: Memory Dump Deduplication

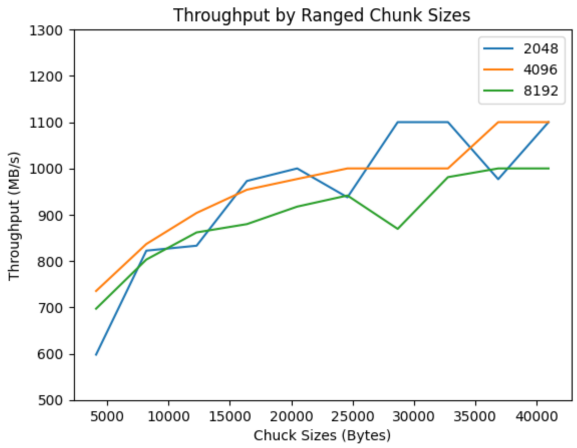

Fig. 12: Memory Dump Throughput

F. Memory Dumps

The deduplication ratio of our memory dumps indicates a gradual decrease as the base chunk size increases. Therefore, they benefit from smaller chunk sizes. The ranged chunk size 2048 starts out with the highest deduplication ratio but ends as the worst, while the opposite works for the ranged chunk size 8192. These indicate that for larger base chunk sizes, it benefits from a larger chunk range. But there is an interesting ending where range 2048 overtakes range 4096. This overtake occurs at a base chunk size of 40,000 bytes. Additionally, throughput indicates that there is variability between ranges 2048 and 4096, where range 8192 remains the slowest.

IV. LIMITATIONS

One limitation we came across was the number of memory dumps we had access to. The TA was only able to give us four memory dumps to analyze, limiting our ability to produce results. We limited our base chunk size as well as our range of chunk sizes such that we are only evaluating a subset. This work could be extended to consider much larger chunks as well as a different variety of chunk ranges.

V. FUTURE WORK

Our plans for future work will first be to compare this deduplication algorithm to other existing software deduplication algorithms. The research and testing we did only included one algorithm, but there are many others out there that this process could be done on. Valuable information can be learned to find the best algorithm for specific purposes and file types if this type of research was done. Another step for future work would include evaluating Java workloads. We evaluated over C workloads, and comparing these results to evaluating over Java workloads could also provide new insight. Along with testing on Java workloads, it would also be beneficial if we tested on more memory dumps. We were only limited to a few memory dumps that we were given by the TA, but having a few more to test on could also improve the results. The last step for future work would be to determine a relationship between what workloads can be best deduplicated.

VI. CONCLUSIONS

After all of our testing and data collection, we came to a few conclusions. We learned the benefits and utility of memory deduplication. Computers can sometimes have large chunks of duplicated memory that can be removed to clear up storage space, which is very beneficial to performance and efficiency. We also learned about the effect of selecting different chunk sizes when deduplicating memory. The larger chunk sizes typically result in faster completion but lower deduplication percentages. This makes logical sense because as the window of memory we are searching for gets larger, the algorithm can complete faster since there are fewer iterations, but the amount of data that is being recognized as a duplicate is being reduced. Depending on the file size and type, it makes sense to choose a smaller chunk size for some and a larger chunk size for others. For example, memory dump deduplication performs better with a smaller chunk size, while compressed videos benefit from a larger chunk size. There needs to be a balance between speed and deduplication performance, which is why a variable chunk size is important.

REFERENCES

[1] XIA W., ZHOU Y., JIANG H., FENG D., HUA Y., HU Y., LIU Q., ZHANG Y. FastCDC: A Fast and Efficient Content-Defined Chunking Approach for Data Deduplication. 2016 USENIX Annual Technical Conference

[2] EL-SHIMI, A., KALACH, R., KUMAR, A., ET AL. Primary data deduplication–large scale study and system design. In Proceedings of the 2012 conference on USENIX Annual Technical Conference (USENIX'12) (Boston, MA, USA, June 2012), USENIX Association, pp. 1–12.

[3] BRODER, A. Some applications of Rabin's fingerprinting method. Sequences II: Methods in Commun

[4]] EL-SHIMI, A., KALACH, R., KUMAR, A., ET AL. Primary data deduplication–large scale study and system design. In Proceedings of the 2012 conference on USENIX Annual Technical Conference (USENIX'12) (Boston, MA, USA, June 2012), USENIX Association, pp. 1–12

[5] Iscc, "ISCC/FASTCDC-Py: Fastcdc implementation in python https://pypi.org/project/fastcdc/," GitHub. [Online]. Available: https://github.com/iscc/fastcdc-py. [Accessed: 06-May-2023].