What Is the Real Impact of SHA-256? A Comparison of Checksum Algorithms

by alex duryee
digital & metadata preservation specialist
avpreserve

Revised: October 2014

AVPreserve | Media Archiving & Data Management Consultants
350 7th Avenue, Suite 1605 | New York, New York | 10001
917.475.9630 | www.avpreserve.com | info@avpreserve.com

Introduction

One of the most critical aspects of digital preservation is ensuring the ability to ascertain whether digital assets have been altered, regardless of if such alteration occurs through intention, accident, or chance. Checking files for potential alteration is usually achieved by monitoring *file fixity*, which is "the assurance that a digital file has remained unchanged" over time[1]. Typically, verifying file fixity is accomplished by generating, recording, and monitoring the *checksum* of a file, which provides a unique value based on the bit-level contents of the object. There are a variety of algorithms that can be used for generating checksums, with two in particular—MD5 and SHA-256—being the most common[2]. The comparative benefits and drawbacks of both are well-understood: while MD5 is weaker against random and deliberate collisions, it is faster to generate than SHA-256[3]. However, there are no published empirical estimates for the difference in time-to-generate between MD5 and SHA-256 in archival and repository environments, leading to difficulty in making an informed decision as to which algorithm to implement for preservation monitoring.

Testing Procedure

In August 2014, AVPreserve tested the difference in the time required to generate checksums using MD5 and SHA256. For the test, we used a collection of 131,870 files (comprised of pseudo-random data), totaling 123 gigabytes in size, located on a network attached storage (NAS) device. This scenario was selected based on its similarity to most archival repositories, where assets are stored on a dedicated server and accessed over a network. The client machine (the computer generating the checksums) was a MacBook Air (model MacBookAir6,2) with an Intel Core i7 processor, and the storage environment was a QNAP TS-669 Pro mounted via SMB. All devices and cabling used on the local area network in which the tests were performed were gigabit.

In testing, we used the UNIX `time` command to record the time elapsed while hashing the data:

```
$ time (perl hasher-[algorithm].pl) 2>>results-[algorithm].txt
```

This command was selected for its ability to separate *real time* from *processor time*. Real time is the time elapsed in the real world while the process executes, which may be impacted by data transfer rates or other factors; processor time, on the other hand, records how long the processor was working during the process. Thus, the time required for data to move over the network was recorded in real time, but not processor time; time spent actually generating checksum values is recorded in both real and processor time. By separating the two values, we isolated the amount of time spent generating checksum values within the complete process.

The Perl scripts we used to generate checksums can be downloaded at http://www.avpreserve.com/wp-content/uploads/2014/10/checksum-scripts.zip. In testing, these scripts provided more valuable results via the `time` command than programs such as hashdeep and md5sum. For example, md5sum would report that more time was spent calculating checksums than elapsed in real time. This is due to these programs using multiple processors, something the `time` command does not strongly support and has trouble parsing. Using a single-threaded Perl script allowed us to use the `time` command to accurately capture data about the checksum process.

Using Perl scripts, the test was performed 16 times for each algorithm, with series of tests

1 http://blogs.loc.gov/digitalpreservation/2014/04/protect-your-data-file-fixity-and-data-integrity/

2 The primary differences between MD5 and SHA-256 are the length of the hash value (32 characters and 64 characters, respectively) and the complexity of the cryptographic function that generates them. For more information, please see http://blogs.loc.gov/digitalpreservation/2011/11/hashing-out-digital-trust/

3 http://www.mathstat.dal.ca/~selinger/md5collision/

occurring on different days and times. This was done in order to avoid network traffic at a given time interfering with the results.

Results & Discussion

The results of AVPreserve's tests are in Appendix A below. Three initial conclusions are:

- SHA-256 does require considerably more processor time—in our testing, there was an additional 514.5 seconds for SHA-256 to complete hashing compared to MD5, or 4.18 additional seconds per gigabyte. In follow-up tests that we ran in different environments for verification of our initial results, we found that although the total processing time differed for each environment, the average 30% additional processor time required per gigabyte remained consistent.

- There is a tremendous discrepancy between the amount of time spent by the processor calculating checksums and the amount of time spent in total execution of the process, roughly on the scale of 9 "idle" seconds to 1 processor second.

- The considerable difference in total runtimes for SHA-256 and MD5 (1,674 seconds on average) is due to factors external to the algorithms.

Based on the first conclusion the table below illustrates how this difference scales to collections of varying sizes. Note that these calculations assume that the checksum process is being performed on a single processor thread; as additional threads are introduced, the durations would reduce accordingly. Note that this only includes processor time and not real time.

Storage	MD5	SHA-256
1 TB	3h 5m	4h 14m
25 TB	76h 56m	105h 58m
100 TB	307h 47m	423h 53m

Comparison of processor time needed to generate checksums for various storage sizes (based on test environment results)

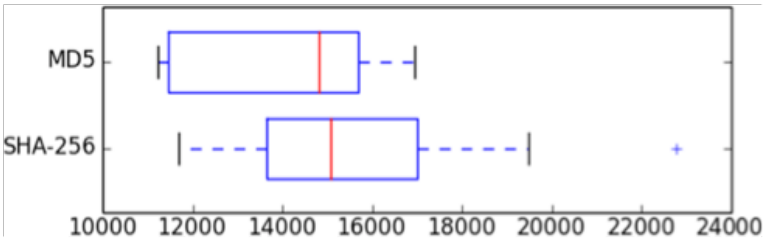
The second conclusion suggests that the traditional comparisons of checksum algorithms, which focus entirely on processor use, are insufficient for analysis of archival use cases. Using the metric of processor time, the MD5 hashes of our corpus completed in 72% of the time that SHA-256 required (average of 1,362.52 seconds versus 1,877.02 seconds). Taken by itself, this could serve as a strong argument for the implementation of MD5 over SHA256. However, this is not an accurate reflection of the actual time elapsed when computing hash values for assets in a digital environment. During our testing, checksum generation represented only 9.8% to 12% of time spent by the process, thus rendering any comparison of algorithms using total runtime impossible. The other 88% to 91.2% was spent performing other tasks, and thus renders the difference in algorithm time negligible.

This large amount of time was spent performing non-checksum related tasks necessary for the process to complete. Most of it was spent transferring data over the network. Checksum generation for on-disk assets requires that data be passed from the storage environment to the processor, which computes the hash based on incoming data and stores it in RAM. As such, during the checksum generation/validation process, every bit of every file that is analyzed must be passed from storage to the machine generating the hashes. If the assets are stored on a

remote server, this data transfer serves as a bottleneck—because the processor generates hashes faster than it can receive data from storage, it spends most of its time waiting to receive data. In our tests, this waiting period represented 90% of the time spent by the checksum process; in other words, very little of the checksum process involved actual checksum generation.

The third conclusion stems from an analysis of the total runtimes compared to processor time used. The processor time used by each algorithm had very little variance—each value was within 3% of the mean—whereas the total runtime had up to 46% deviance from the mean. Given the wide variance of total runtime compared to the highly consistent processor time, and the considerable overlap between the total runtimes of the algorithms, there is strong evidence that factors other than the algorithm determine the amount of time elapsed. The most likely factors include available network bandwidth and server disk input/output speed at time of testing.

The chart below graphs the median real times (red line) and the quartiles (blue lines) for each algorithm to scan the test data. Note that the medians are very close to each other, and that there is considerable overlap between the times elapsed. This indicates that any difference in performance between MD5 and SHA-256 was minor compared to the total time elapsed. In other words, it is possible that the extra time necessary for SHA-256 to calculate may be lost in the noise of network bandwidth, storage input/output, and other factors.



Comparison of seconds elapsed of real time

For comparison, we also ran checksum tests against 100 gigabytes of data locally on the MacBook Air used as the client machine. Since this machine uses a solid-state drive with very high bandwidth, the effect of storage-to-processor time on the test was minimized. When testing both MD5 and SHA-256 in this environment, we found that the processor time and real time were within 15% of each other[4]. This confirms that most of the time “checking fixity” is actually spent waiting for data to move from one system to another. As such, performing fixity checks as close to the storage environment as possible can greatly reduce the time necessary to validate assets.

Note that the time to generate checksums will vary depending on environment, as demonstrated in our network test and local tests. In our testing, we found that generating MD5 checksums on remote storage required 11.08 seconds of processor time, and SHA-256 required 15.62 seconds. On local storage, MD5 required 8.9 seconds of processor time, and SHA-256 required 12.33 seconds. Despite the difference in real time required to generate checksums on different environments (likely caused by factors such as network overhead), the proportional difference between the algorithms remains approximately constant around 72%.

Conclusions

Taking the results of these tests and conclusions into consideration, AVPreserve recommends the following when planning to implement file fixity generation and verification:

4 The average real time for MD5 was 1,049 seconds, and average processor time was 890 seconds; the average real time for SHA-256 was 1,393 seconds, and the average processor time was 1,233 seconds.

- **Assess repository architecture for potential bottlenecks in the fixity workflow.** If the machine generating checksums has limited bandwidth to the storage environment, this will have a dramatic effect on the time required to perform fixity tests.

- **In cases where speed is an issue (for example, the sheer quantity of data would make fixity verification tests run for very long periods), various optimizations should be performed to allow for regular fixity checks.** Discussions with IT specialists may be fruitful in finding ways to maximize fixity throughput—for example, decreasing latency between devices, or parallelizing checks to maximize processor throughput.

- **Use empirical data to determine which checksum algorithm to implement in repositories.** Analyzing data and extrapolating results will help in determining which algorithm is best suited for particular infrastructure and preservation needs.

Acknowledgements

The author would like to thank Kam Woods for detecting an error in the original data used for this paper, and for sending along revisions to the data generation process.

Appendix A

Test Results (In Seconds)					
SHA-256		MD5		Time Difference	
Real Time	CPU Time	Real Time	CPU Time	Real Time	CPU Time
12016.772	1938.277	11556.661	1361.648	460.111	576.629
12008.204	1927.289	11444.962	1350.49	563.242	576.799
14555.803	1938.349	14933.721	1370.436	-377.918	567.913
13743.785	1938.591	11446.803	1346.633	2296.982	591.958
13400.161	1862.212	11444.177	1349.705	1955.984	512.507
19485.504	1845.571	15428.809	1366.617	4056.695	478.954
14339.282	1832.346	11353.914	1339.73	2985.368	492.616
14433.376	1842.451	11211.001	1343.941	3222.375	498.51
18473.928	1848.278	11956.528	1358.08	6517.4	490.198
11689.236	1825.854	15212.994	1361.727	-3523.758	464.127
15876.057	1849.811	15602.471	1363.056	273.586	486.755
16724.46	1855.241	15886.136	1375.957	838.324	479.284
22772.095	1907.015	16897.595	1372.317	5874.5	534.698
15581.72	1866.037	16942.971	1403.573	-1361.251	462.464
17854.547	1883.278	14718.193	1366.574	3136.354	516.704
16531.463	1871.738	16662.786	1369.817	-131.323	501.921
Averages Times					
15592.9	1877.0	13918.7	1362.5	1674.2	514.5

*Raw data is packaged with the Perl scripts available at http://www.avpreserve.com/wp-content/uploads/2014/10/checksum-scripts.zip.