Description and Analysis of the Web3 Storage Benchmarking Experiment

György Barabás and Marko Zidarić

Purpose and experimental design

The purpose of this project was to compare data download speeds and reliability across three different Web3 storage platforms: Arweave, IPFS, and Swarm. The data were gathered by first uploading files of various sizes to all three platforms, and then downloading them under different circumstances. We then compared the times needed for data retrieval to see whether and when some platform allowed for shorter download times than the others.

Our goal was to implement this speed comparison as a proper, repeatable, well-designed digital experiment. This meant that we always uploaded unique, randomly generated files of fixed size, which were then downloaded from the same server to remove the confounding influence of server identity. (The experiment as a whole can, and was, repeated over several different servers; see below.) We up- and downloaded the exact same number of files of specified sizes on each platform. We additionally varied some other parameters as well to gauge their influence on download speeds. We varied all parameters in a fully factorial way, to get at all their possible combinations (with minor exceptions mentioned below). The factors were as follows:

- `size`: The size of the uploaded random file. We had 7 distinct factor levels: 1 KB, 10 KB, 100 KB, 1 MB, 10 MB, 100 MB, and 500 MB. Importantly, every single upload was a unique random file, even if the file sizes were otherwise equal.
- `platform`: Whether the target platform is Arweave, IPFS, or Swarm. *Note:* For Arweave and IPFS, no 500 MB files were uploaded. That size category is reserved for Swarm only. Additionally, Swarm itself breaks up into several sub-factors depending on:
  - the strength of erasure coding employed (`0` = None, `1` = Medium, `2` = Strong, `3` = Insane, and `4` = Paranoid);
  - the used redundancy strategy (we use `NONE` whenever erasure coding is set to 0; otherwise we use `DATA` and `RACE`).

  Together, these lead to 11 distinct factor levels for `platform`, namely `Arweave`, `IPFS`, and all combinations of the above for Swarm: `Swarm_0_NONE`, `Swarm_1_DATA`, `Swarm_1_RACE`, `Swarm_2_DATA`, `Swarm_2_RACE`, `Swarm_3_DATA`, `Swarm_3_RACE`, `Swarm_4_DATA`, and `Swarm_4_RACE`.
- `server`: The identity of the server initiating the downloads might influence download speeds. For a fair comparison, servers should be identical within an experiment, but it makes sense to perform the whole experimental suite over multiple different servers—which is what we have done. This means that we have an extra experimental factor, with as many distinct levels as the number of distinct servers used. Here we have used three distinct servers.

- `replicate`: To gain sufficient sample sizes for proper statistical analysis, every single combination of the above factors was replicated 30 times. For example, given the unique combination of 1MB files uploaded to IPFS on Server 1, we actually up- and downloaded at least 30 such files (each uniquely and randomly generated, of course).

The above design leads to (7 filesizes) x (11 platforms) x (3 servers) x (30 replicates) = 6930 unique download experiments—or, rather, somewhat fewer because IPFS and Arweave did not have any 500 MB files up- and downloaded.

Additionally, here are some further notes and points about the experimental design outlined above:

- For Swarm (and only Swarm), upload speeds were also measured, using the tags API to make sure that all chunks have been properly placed and uploaded to the system before declaring a file fully uploaded.
- All uploads to Swarm were direct, as opposed to deferred.
- The reason for insisting on unique, randomly generated files for uploading was to eliminate the possibility of cached downloads (if supported by the target platform).
- We needed to make sure that no download started after the system has properly stored the data. Since our files are relatively small, uploading should be done in about 10-20 minutes at most. So we opted for a crude but reliable way of eliminating any syncing issues: we waited exactly 2 hours after every upload, and began downloading only then.
- When testing IPFS, the data were uploaded from one server but downloaded from another.
- All Swarm downloads were done using nodes with an active checkbook. (Eventually we might also repeat the experiment with no checkbook, but for now we are not dealing with this issue.)
- Every download was re-attempted in case of a failure. In total, 15 attempts were made before giving up and declaring that the file could not be retrieved.

Preliminary data analysis

After we load and compile the data, we get a table with 7293 rows and 8 columns. Each row corresponds to a single download. The first ten rows are:

platform	size_kb	server	erasure	strategy	time_sec	sha256_match	attempts
Arweave	1	Server 1	NONE	NONE	1.4445	TRUE	1
Arweave	1	Server 2	NONE	NONE	1.8055	TRUE	1
Arweave	1	Server 3	NONE	NONE	1.4425	TRUE	1
Arweave	1	Server 1	NONE	NONE	1.5356	TRUE	1
Arweave	1	Server 2	NONE	NONE	1.5249	TRUE	1
Arweave	1	Server 3	NONE	NONE	2.1991	TRUE	1
Arweave	1	Server 1	NONE	NONE	1.4169	TRUE	1
Arweave	1	Server 2	NONE	NONE	2.0065	TRUE	1
Arweave	1	Server 3	NONE	NONE	1.7447	TRUE	1
Arweave	1	Server 1	NONE	NONE	1.4898	TRUE	1

A quick note: we just calculated that the experimental design leads to somewhat fewer than 6930 downloads, yet here we have 7293 rows of data. This is because for Arweave and IPFS we had 60 replicates per factor combination instead of 30 (with one exception: we only had 51 replicates for files of size 100 MB). Instead of discarding the extras, we kept them to lend

further statistical power down the line. This was also important to do because some of the downloads failed: 276 out of the 7293 downloads could not finish even in 15 attempts. This is how these failures are distributed across the parameterizations:

platform	server	size_kb	erasure	strategy	number of fails
IPFS	Server 1	1e+05	NONE	NONE	38
IPFS	Server 2	1e+00	NONE	NONE	30
IPFS	Server 2	1e+01	NONE	NONE	30
IPFS	Server 2	1e+02	NONE	NONE	30
IPFS	Server 2	1e+03	NONE	NONE	30
IPFS	Server 2	1e+04	NONE	NONE	30
IPFS	Server 2	1e+05	NONE	NONE	45
IPFS	Server 3	1e+05	NONE	NONE	38
Swarm	Server 1	5e+05	INSANE	DATA	1
Swarm	Server 2	5e+05	INSANE	DATA	1
Swarm	Server 2	5e+05	STRONG	DATA	1
Swarm	Server 2	5e+05	STRONG	RACE	1
Swarm	Server 3	5e+05	INSANE	DATA	1

On the other hand, all the other 7017 downloads succeeded, and all but four of them on the first attempt. For reference, here are the data for these four downloads:

platform	size_kb	server	erasure	strategy	time_sec	attempts
Arweave	1e+05	Server 2	NONE	NONE	130.9	8
Swarm	5e+05	Server 2	MEDIUM	RACE	1261.6	3
Swarm	5e+05	Server 2	MEDIUM	RACE	1599.8	3
Swarm	5e+05	Server 2	INSANE	DATA	911.0	2

Before analyzing the data, we remove the 276 failed downloads from the table and work only with the rest.

Visualizing and interpreting the raw download data

The data from the experiment are shown in Figure 1. We have a grid of panels, with 7 rows and 3 columns. The columns are the three different servers used for downloading. The rows indicate storage platform and level of erasure coding (if applicable). Within each panel, file size is along the x-axis and retrieval time along the y-axis, both on the log scale. Colors show retrieval strategies: when erasure coding is absent, blue means NONE, whereas in the presence of erasure coding, it means DATA. Yellow indicates the RACE strategy. Each point corresponds to one download event. The points have been jittered sideways in a way that reflects the general shape of their distribution. This is just for visual help; the only possible file sizes are still 1 KB, 10 KB, 100 KB, 1 MB 10 MB, 100 MB, and 500 MB.

Some important take-aways from this figure:

- Arweave's download times increase the slowest, though it also takes longer to download small files from it.
- IPFS download times increase somewhat faster.

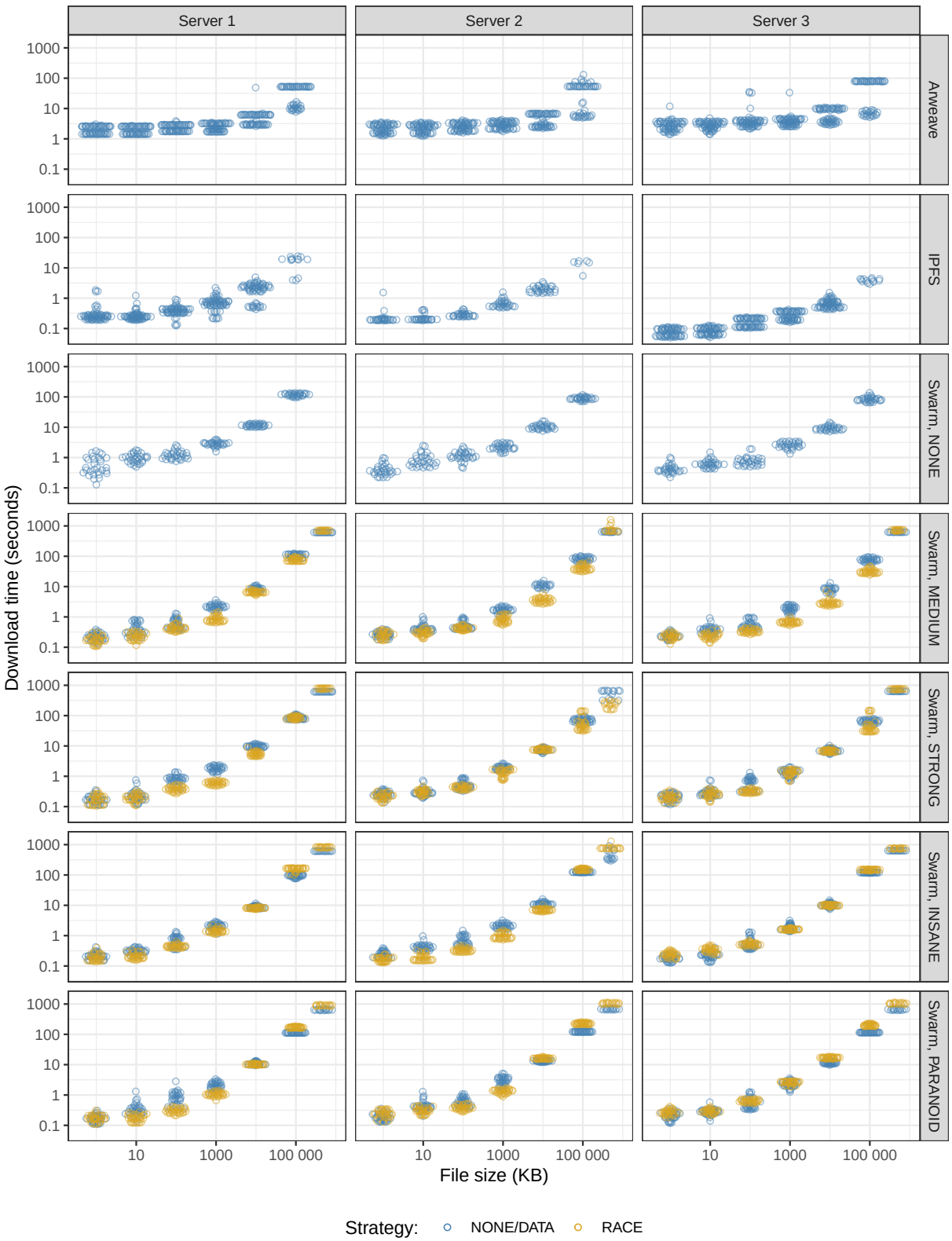

Figure 1: Empirical results from the benchmarking experiment

- Swarm increases the fastest, so for large files it is the least efficient. This was expected, given its underlying DISC model.
- The level of erasure coding does not appear to have much of an effect on download speeds, at least for very small and large files.
- The DATA and RACE retrieval strategies do lead to differences. For small files RACE is faster; for larger files, sometimes RACE is faster, sometimes DATA.

The raw data for upload times to Swarm

As mentioned before, upload times to Swarm were also measured. Uploads have been performed using the `direct` method, instead of the default `deferred` method. The raw data are in Figure 2.

Predictably, upload times increase with file size. Also not surprisingly, there is a regular trend for higher levels of erasure coding to lead to longer upload times.

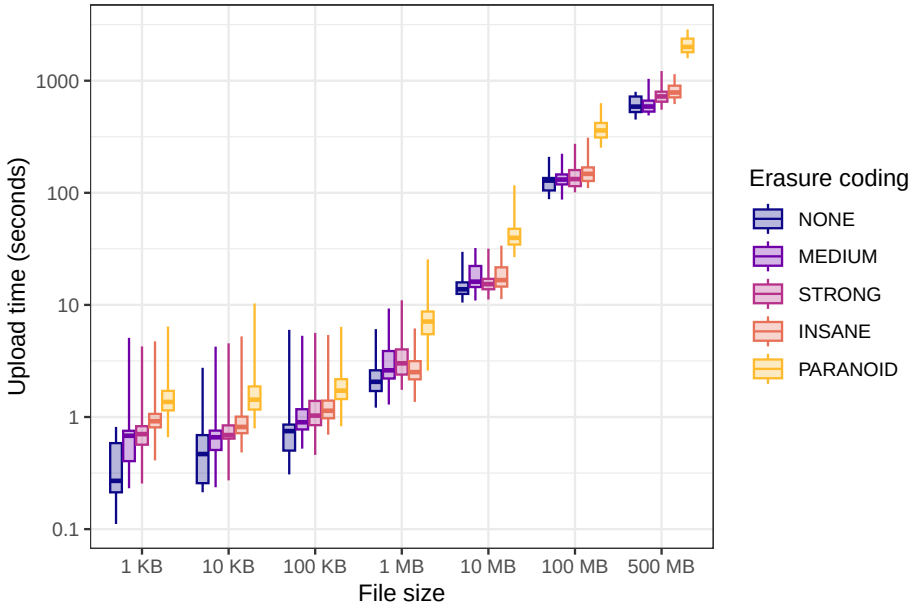

Figure 2: Upload times to Swarm (y-axis; note the log scale), as a function of file size (x-axis; also on the log scale) and level of erasure coding (color legend on the right). Brighter colors represent higher levels of erasure coding.