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. Additionally, we wanted to explore the effects of erasure coding and various file retrieval strategies on download performance within the Swarm platform. Finally, for Swarm only, we also wanted to get a picture of how the speed of file uploads scales with their size.

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, then 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. The factors were as follows:

- `platform`: Whether the target platform is Arweave, IPFS, or Swarm. Additionally, Swarm 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`.
- `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. *Note:* 500 MB file uploads were limited to Swarm, and even there to nonzero erasure coding.
- `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. 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.
- 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).
- All uploads to Swarm were direct, instead of the default "deferred" upload type. This was a better fit with the spirit of the experiment.
- We needed to make sure that no download started before the system has properly stored the data. Since our files are relatively small, uploading should be done in about 10-20 minutes at most (except 500 MB files, which take about half an hour). 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 shown in Table 1. A quick note: we just calculated that the experimental design leads to 7293 downloads, yet here we have 6966 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. The distribution of these failures across the various parameterizations is shown in Table 2.

On the other hand, all the other 7017 downloads succeeded, and all but two of them on the first attempt. (One download of a 100 MB file on Arweave completed on the 8th attempt, one 500 MB file on Swarm on the 2nd attempt, and two more 500 MB files on Swarm on the 3rd attempt. Before analyzing the data, we remove the 276 failed downloads from the table and work only with the rest.

platform	size_kb	server	erasure	strategy	time_sec	sha256_match	attempts
Arweave	1	Server 1	NONE	NONE	1.445	TRUE	1
Arweave	1	Server 2	NONE	NONE	1.806	TRUE	1
Arweave	1	Server 3	NONE	NONE	1.442	TRUE	1
Arweave	1	Server 1	NONE	NONE	1.536	TRUE	1
Arweave	1	Server 2	NONE	NONE	1.525	TRUE	1
Arweave	1	Server 3	NONE	NONE	2.199	TRUE	1
Arweave	1	Server 1	NONE	NONE	1.417	TRUE	1
Arweave	1	Server 2	NONE	NONE	2.006	TRUE	1
Arweave	1	Server 3	NONE	NONE	1.745	TRUE	1
Arweave	1	Server 1	NONE	NONE	1.490	TRUE	1

Table 1: First ten rows of the raw download data.

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

Table 2: Failed download attempts.

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 aid; the only possible file sizes are still 1 KB, 10 KB, 100 KB, 1 MB 10 MB, 100 MB, and 500 MB.

Some immediate 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.
- Swarm increases the fastest, so for large files it is the least efficient. This was expected, given its underlying DISC model.
- The DATA and RACE retrieval strategies lead to visible differences in download times, but this difference appears context-dependent. For small files RACE is faster, but for larger files and higher levels of erasure coding, DATA appears more efficient.
- It is difficult to glean out the effect of erasure coding on download times from this figure alone.

To address the last item above, we can visualize the data separately for each size category (Figure 2). This removes the distortion caused by the fact that files of different size download at vastly different speeds. What we see is that except for very small (1 KB) files, there is a non-monotonic relationship between erasure levels and download times. Intermediate erasure levels offer the best speed gains, which then diminish and sometimes even turn into a speed loss when reaching the highest erasure levels. This is especially true for the RACE strategy at larger file sizes (10-100 MB), where downloading files under the PARANOID erasure coding is generally slower not just than the scenario without erasure coding, but also slower than the corresponding DATA strategy.

Modeling download times

We now turn to building simple predictive models for these data. Their purpose is to be able to extrapolate how long it would take to download larger data files than the ones used in the data. The models are phenomenological, and therefore should be used with care: they will not extrapolate well for file sizes much larger than the ones tested. But they can give an idea of how each parameter is expected to influence the results.

We will build separate models for the three platforms. Since they are simpler in terms of the structure of the data, let us start with Arweave and IPFS, and then move on to Swarm.

Arweave and IPFS By looking at the data in Figure 1, we see that log download times increase nonlinearly with log file size. The relationship may be quadratic or cubic; here we assume it is cubic (this assumption does have limitations, as we will see later). To avoid bias arising from the fact that the log function compresses large values more than small values, we want to use a generalized linear model with a log link function to predict download times. Also,

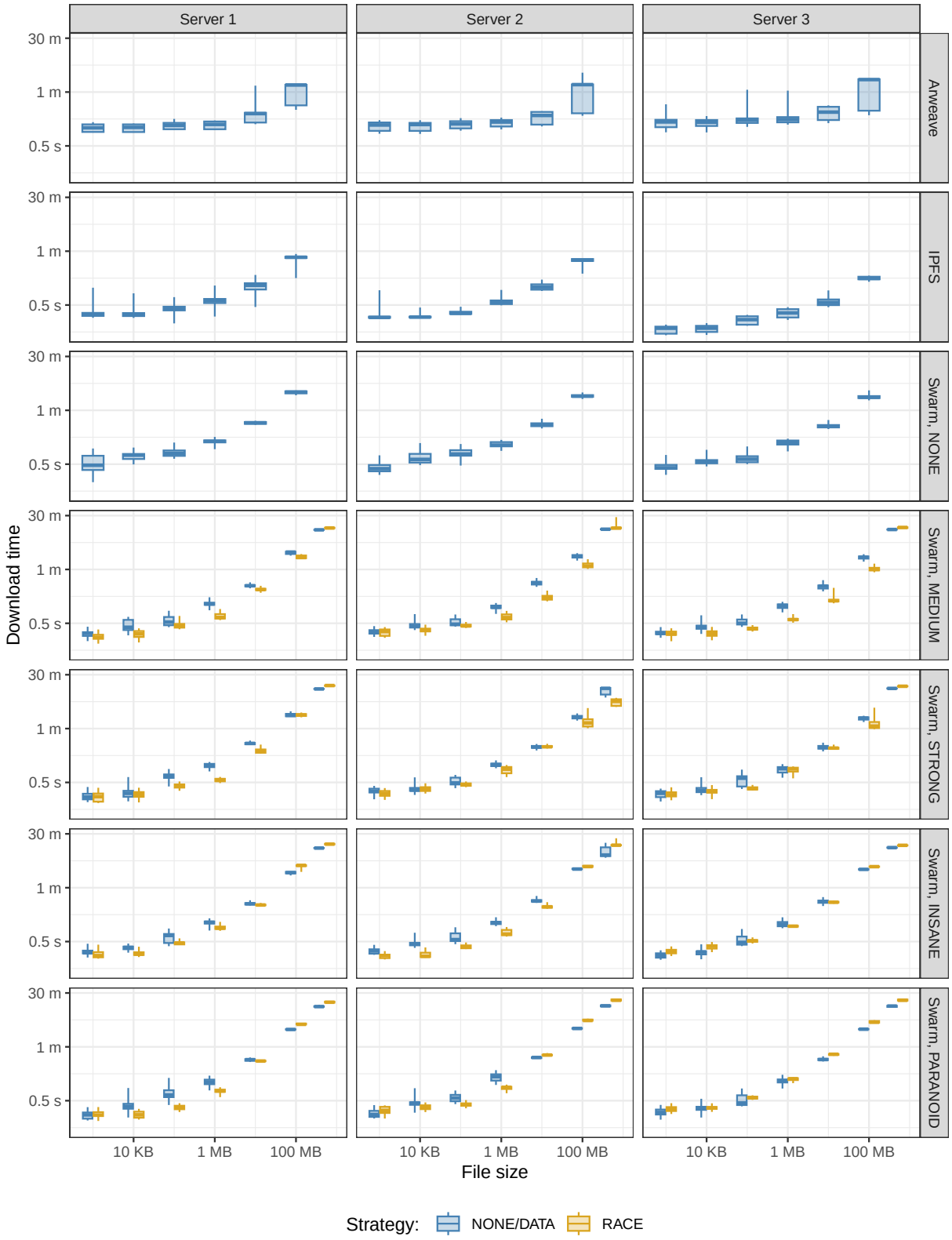

Figure 1: Empirical results from the benchmarking experiment. Box plots are standard except no outliers are shown—that is, the thick horizontal line is the median (point that separates the top and bottom half of the data), the box around it encompasses the middle 50% of all data points, and the top/bottom whiskers show where the top/bottom 25% of the data are.

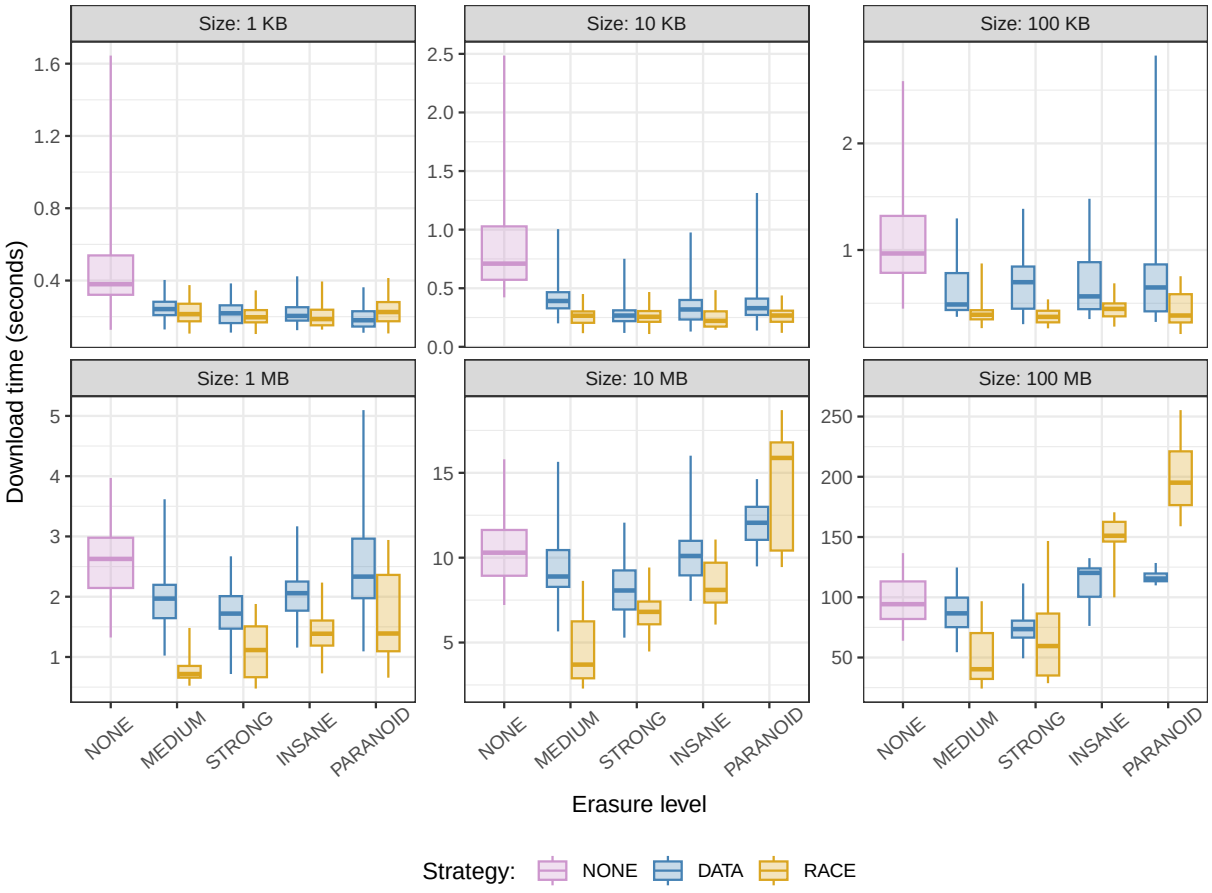

Figure 2: Download times (y-axis) as a function of erasure level (x-axis), file size (panels), and retrieval strategy (colors). Note that the y-axis is individually scaled for each panel. Data for 500 MB files are omitted, since no such files were up- or downloaded in the absence of erasure coding.

since the data come from three different servers, we will take the influence of server identity into account as a random effect. The model therefore reads

$$(\text{time})_i = \exp\left[\beta_0 + \beta_1 \log^3(\text{size})_i + \mu_{0,\text{v}(i)} + \mu_{1,\text{v}(i)} \log^3(\text{size})_i\right]. \tag{1}$$

Here $(\text{time})_i$ and $(\text{size})_i$ are respectively the ith download time (in seconds) and file size (in kilobytes) in the data, β_0 is an intercept, β_1 is a slope, and $\mu_{\text{v}(i)}$ is a random effect, where $\text{v}(i)$ returns 1, 2, or 3, depending on which server the ith data point was downloaded from.

Swarm Since Swarm additionally has erasure coding and different retrieval strategies, the fitted model is also more complex, although it is still a Gaussian generalized linear mixed model with a log-link function. It has the form

$$\begin{aligned}(\text{time})_i = \exp[&\beta_0 + \beta_1 \log^2(\text{size})_i + \beta_2(\text{erasure})_i + \beta_3(\text{strategy})_i \\ &+ \beta_4 \log^2(\text{size})_i(\text{erasure})_i + \beta_5 \log^2(\text{size})_i(\text{strategy})_i + \beta_6(\text{erasure})_i(\text{strategy})_i \\ &+ \mu_{0,\text{v}(i)} + \mu_{1,\text{v}(i)} \log^2(\text{size})_i + \mu_{1,\text{v}(i)}(\text{erasure})_i].\end{aligned} \tag{2}$$

In words, we predict log(time) using the main effects of $\log^2$(size), erasure level, and strategy, and all their possible two-way interactions. Plus we add a random intercept and slope (for both $\log^2$(size) and erasure level) based on server identity. In this model, erasure level is not a factor but a covariate reflecting the corresponding baseline probability of a faulty chunk retrieval event: it is taken as 0 for no erasure coding, as 1 for MEDIUM, 2 for STRONG, 3 for INSANE, and 4 for PARANOID. (Model selection indicated this as the best choice among a few other candidates, such as using the probability of individual chunk failure or using the fraction of chunks that are parity chunks.)

Before doing anything else, let us confirm that the models do capture the data well. To do so, we plot the raw data (points) together with the model predictions (lines) in Figure 3.

The models generally fit well, though they are sometimes off at small file sizes. Note however that, since the y-axis is on the log scale, a difference that appears large at small download times is not actually a relevant difference. For example, the average download time for Swarm on Server 1 without erasure coding is 0.648 seconds for files of 1 KB. The model's prediction is 0.185 seconds—which, although more than threefold larger, is less than half a second off the empirically-observed value.

Now that we have more confidence in the models, we can use them to make out-of-sample predictions. Figure 4 shows model-predicted extrapolations made from the data, averaged over the random effects.

Three remarks about these results are in order. First of all, the modeled curves for Arweave and IPFS eventually increase faster than the curves for Swarm. This would mean that for very large files, Swarm becomes the most efficient platform. This is definitely an artifact and reflects the limitations of the models, rather than telling us something about how these systems work in reality. In particular, this is the result of having modeled Swarm with a quadratic and Arweave/IPFS with a cubic curve. Since cubic functions increase faster than quadratic ones for large arguments, the model's predictions will only be reliable up to some limited file size. Second, on Swarm and for large files, RACE is on average not a better retrieval strategy than DATA: the dashed lines lie above the solid ones for files over 1GB. And third, while for large files the RACE curves are only slightly above the DATA curves, one must remember that the y-axis is on the log scale. This means that the actual difference is substantial. For example: for 10 GB files, the DATA strategy takes around 8 hours and the RACE strategy around 18 hours to download, with erasure levels playing only a minor role in shaping these figures.

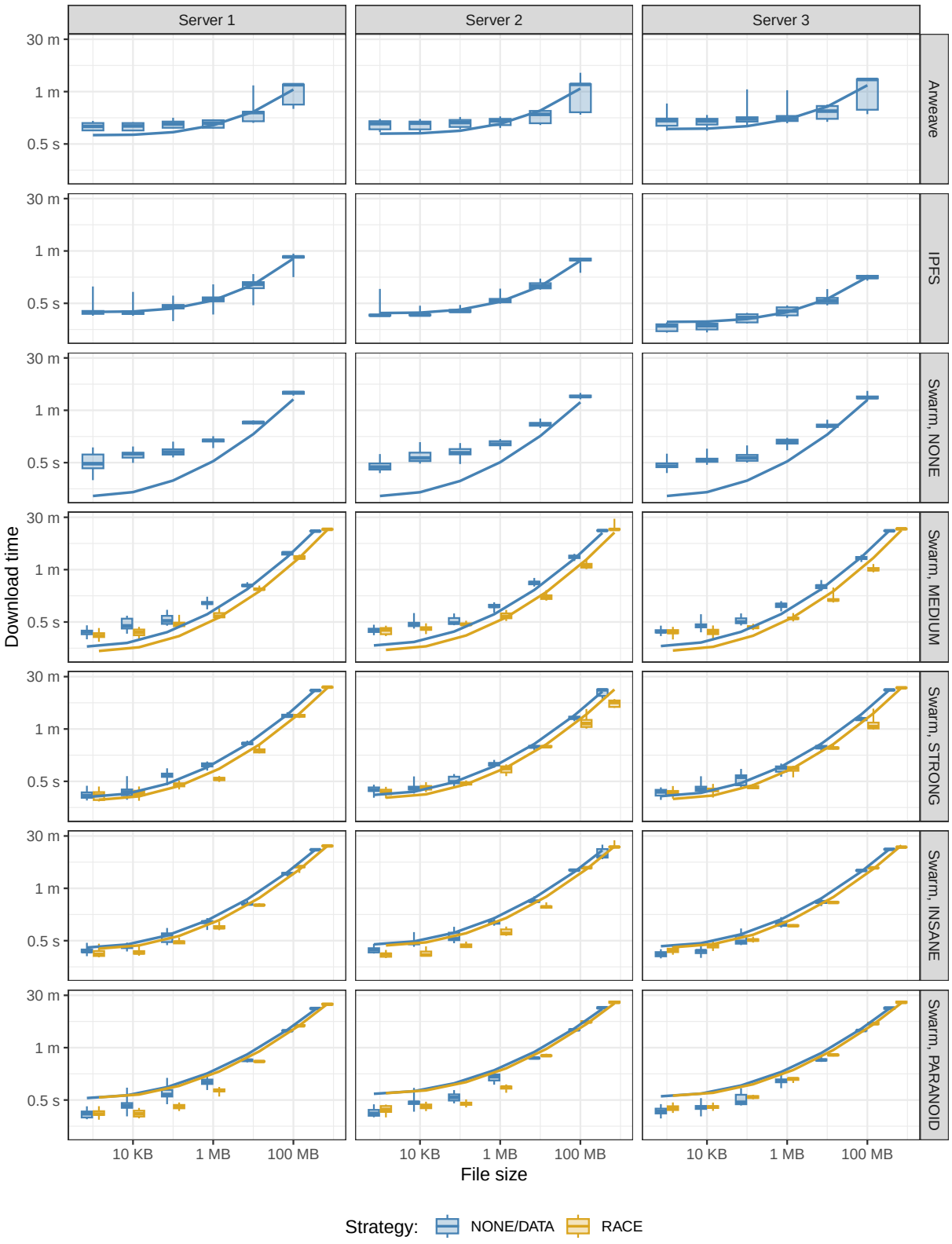

Figure 3: As Figure 1, but with the data (box plots) plotted together with predictions from the fitted models (lines).

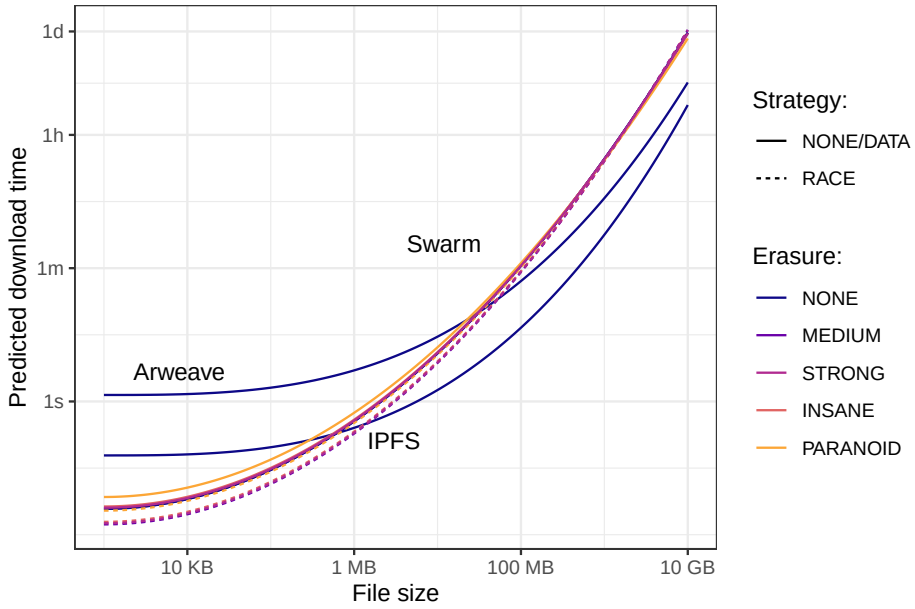

Figure 4: Model-based predictions for mean file retrieval times. Random effects are averaged over, so server identity no longer plays a role.

Modeling the effect of erasure codes and retrieval strategies on download times

We start from the observations made in Figure 2: except for very small files, erasure coding has a non-monotonic, U-shaped effect on download times, and while the RACE strategy is generally faster than DATA, this is no longer the case when file sizes are large and erasure coding is very strong.

A clearer and statistically more interpretable version of the same figure is obtained after standardizing download times. Within each size category, we subtract from every observation the mean download time, and divide this difference by the standard deviation of the download times. This results in the *z-score* for download times: the original data, but scaled so that (i) they are dimensionless, (ii) their mean is zero, and (iii) their standard deviation is equal to one. The summary of these z-scores, broken down by erasure level and retrieval strategy, is shown in Figure 5.

The U-shaped relationship is not a mirage: it is supported by models. We will fit three different models to these data to show that this is the case. Each of them relies on treating erasure levels as an ordinal variable, ranging from 0 (NONE) to 4 (PARANOID). The first model is the crudest and simplest, but also the most direct: we predict z_i, the download time z-score for observation i within a given file size category, as a function of $((\text{erasure})_i - 2)^2$. The subtraction of 2 ensures that the quadratic curve is centered at the STRONG erasure level, and the squaring is there to fit a U-shaped curve. The model, which we will call `sqr`, reads

$$z_i = \beta_0 + \beta_1((\text{erasure})_i - 2)^2 + \epsilon_i, \tag{3}$$

where the β_i are regression coefficients and the ϵ_i are normally-distributed residuals. The second model is a more flexible version of the same: instead of fixing the minimum of the function at the STRONG (=2) erasure level, we let the model decide, by adding an explicit first-order term as well:

$$z_i = \beta_0 + \beta_1(\text{erasure})_i + \beta_2(\text{erasure})_i^2 + \epsilon_i. \tag{4}$$

This model will be abbreviated `quad`. Finally, the third model aims at not just making inference easier, but providing a model that predicts the mean behavior accurately. We do this by also

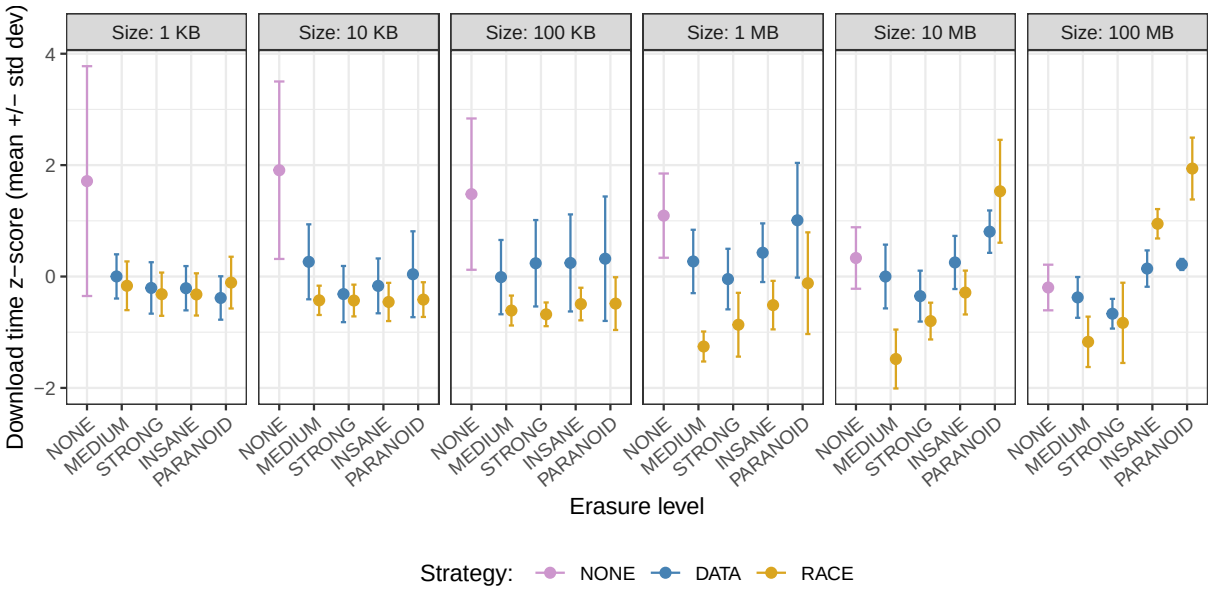

Figure 5: Download time z-scores (y-axis; the z-score is the download time minus the mean, divided by the standard deviation) per file size category (panels), as a function of erasure level (x-axis) and retrieval strategy (colors). Data for 500 MB files are omitted, since no such files were up- or downloaded in the absence of erasure coding. Points represent the mean z-score, whishers represent the plus/minus one standard deviation range around those means. The points and error bars for the different retrieval strategies have been slightly moved sideways to reduce visual overlap.

fitting a cubic term:

$$z_i = \beta_0 + \beta_1(\text{erasure})_i + \beta_2(\text{erasure})_i^2 + \beta_3(\text{erasure})_i^3 + \epsilon_i. \tag{5}$$

This model will be called `cubic`. Note that all these models are for one file size category only. While it would be possible to fit a joint model with file size providing an extra term, it is more to the point to instead fit one model per file size category. Since there are 6 sizes (1 KB, 10 KB, 100 KB, 1 MB, 10 MB, and 100 MB) and 3 models (`sqr`, `quad`, and `cubic`), we will fit 18 models. Actually, it will be two times that, for 36 models, because we will fit separately for the DATA and RACE download strategies.

Let us first of all check how well these models actually fit the data. As seen in Figure 6, none of the models are completely off, but `quad` and `cubic` are generally much more accurate than `sqr`.

It is not obvious from this though which model is "best", because the better-fitting models also have more free parameters. To check model quality, we calculate the AIC score of each model and compare them within each file size category. We do this separately for DATA (Figure 7) and RACE (Figure 8) models. As seen, `cubic` is almost always superior to `quad`, which in turn is superior to `sqr`. The exceptions are for the DATA fits, for 1 MB and 10 MB files. In the former case, all three models perform about equally well, while in the latter, `quad` and `cubic` are similarly good but both are better than `sqr`. (The reason for this is visible in the blue curve for 1 an 10 MB files in Figure 6: in these cases, even the simplest `sqr` model happens to fit the data quite well.)

Importantly, regardless of which model we use, Figure 6 shows that the local minimum of the predicted curves always lies in the range 0 to 4 that corresponds to the permissible erasure

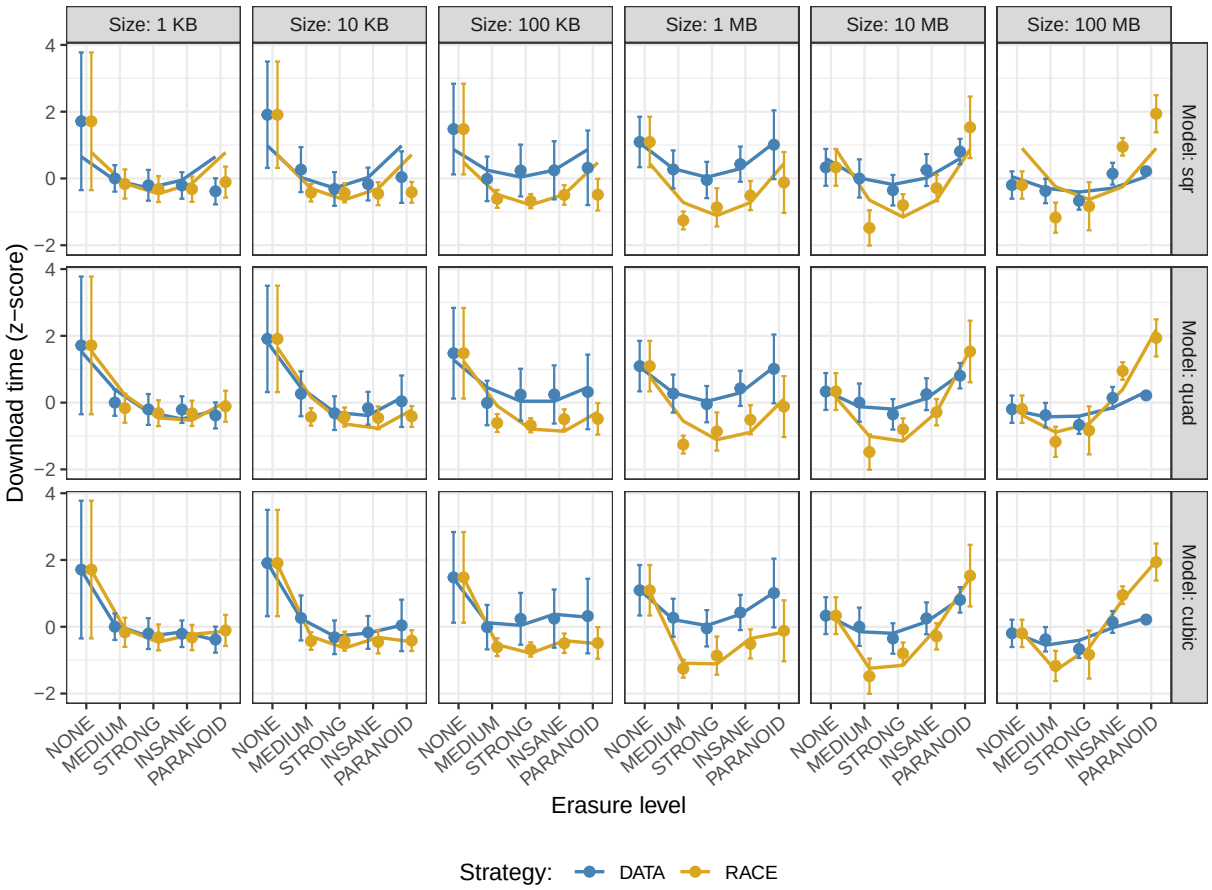

Figure 6: Model fits to the standardized (z-score) download time data, for different download strategies (color). Points and error bars are as in Figure 5; the curves are the predictions from the fitted models. As in Figure 5, the points and error bars for the different retrieval strategies have been slightly moved sideways to reduce visual overlap.

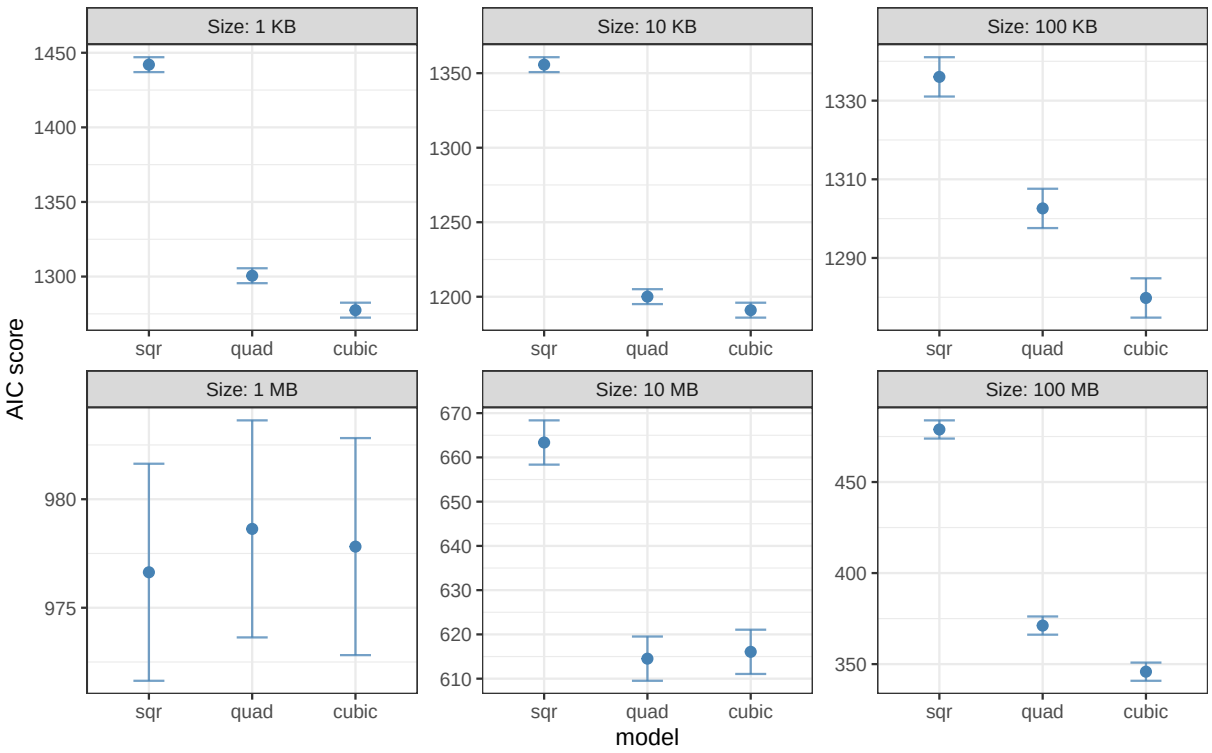

Figure 7: AIC (Akaike Information Criterion) scores for model fits, restricted to the DATA retrieval strategy. Points show the actual AIC scores; the error bars are always indicating a plus/minus 5 AIC point range. This makes it visually easy to see if two intervals overlap. In case they do not, the models are separated by more than 10 AIC points, which is a good rule of thumb for preferring the model with the lower AIC score.

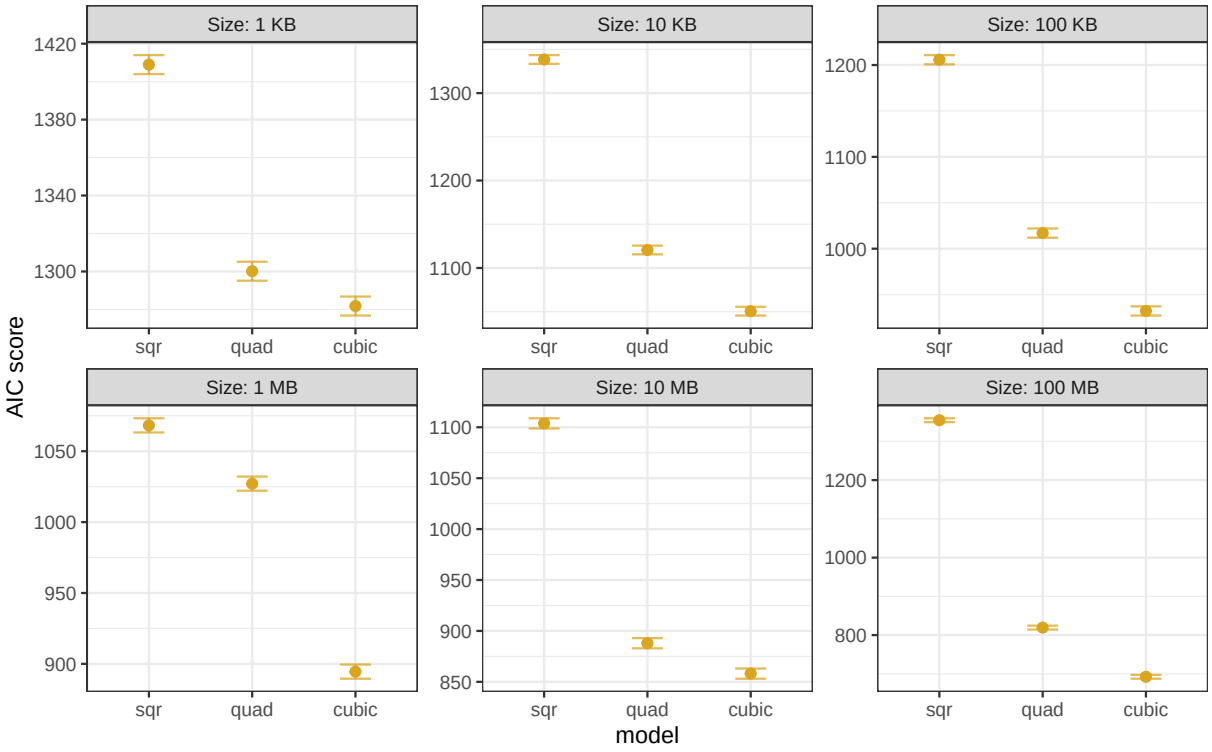

Figure 8: As Figure 7, but restricted to the RACE retrieval strategy.

levels, from NONE to PARANOID. This is good indication that the relationship between download times and erasure levels is indeed U-shaped, but to make this conclusion stronger, we should check whether the quadratic term in the models, responsible for the minima, is in fact significantly different from zero. This does happen to be the case for *all* the considered models, yielding strong evidence that, as suspected, erasure levels have a non-monotonic effect on download times (Table 3). All p-values are extremely low, with $p < 10^{-9}$ even after the most stringent (Bonferroni) correction for multiple testing.

After establishing that erasure coding tends to have the greatest benefit on download times at intermediate strengths, we can look at the effect of using the DATA vs. the RACE download strategies. This can be done in two steps. First, we perform a two-way ANOVA for each file size category on the data in Figure 5, with erasure level and download strategy varying as factors. We then look at the ANOVA tables to see whether and when these factors (or their interaction) appear to play a role. The results are in Table 4. What we see is that for 1 KB files, strategy doesn't make a difference, and the interaction of erasure level and strategy probably doesn't (the Bonferroni-corrected p-value of 0.088 is not too far above the conventional cutoff of 0.05, so it is difficult to tell). This interaction term also appears to play no role for 100 KB files. Apart from these exceptional cases however, all main and interaction effects are always significant.

Now that we know it is meaningful to look at the effect of retrieval strategy on download times (except for 1 KB files), we can perform a post-hoc Tukey test to actually gauge the overall quantitative effect that switching from the DATA to the RACE strategy has. Table 5 shows only the DATA-to-RACE overall contrasts. As seen, there is no significance for 1 KB files. Otherwise, the estimated differences are always negative, except for 100 MB files. This is not surprising: we can see in Figure 5 that the effect of strategy is ambiguous and depends on the interaction term—at least for the file sizes that were tested.

In conclusion, unless file sizes are large and erasure levels are simultaneously very strong, the RACE strategy boosts download speeds compared with the DATA strategy.

The raw data for upload times to Swarm

Upload times to Swarm were also measured. Uploads have been performed using the `direct` method, instead of the default `deferred` method. The data are visualized in Figure 9. Predictably, upload times increase with file size. Also not surprisingly, there is a more or less regular trend for higher levels of erasure coding to lead to longer upload times.

This within-size trend becomes clearer if we calculate the effective size of each file—that is, explicitly account for the extra file size that erasure coding induces for uploading. Table 6 summarizes how many chunks out of every unencrypted 128-chunk sequence are parities. Additionally, one must account for the packed-address chunks used to reference the Swarm hash tree. Let the number of packed-address chunks be P and the number of chunks in the file N. Then, with a branching factor of 128, P can be given in terms of a geometric series:

$$P = \frac{N}{128}\left(1 + \frac{1}{128} + \frac{1}{128^2} + ...\right) = \frac{N}{128}\frac{128}{127} = \frac{N}{127}. \tag{6}$$

So the size of each file should be multiplied by $1 + 1/127 = 128/127$ to account for the presence of packed-address chunks.

Taking into account both the parity chunks and packed-address chunks in inflating file sizes, we can calculate the exact file size for each combination of base size / erasure coding combination. The results are in Figure 10, where all box plots are placed exactly where they should be along the x-axis. From this figure, the more or less linear trend of increase within each size category is obvious.

size	model	quad.term	std.error	adj.r.squared	adj.p.value
1 KB	sqr	0.234	0.034	0.095	<1e-9
1 KB	quad	8.302	1.021	0.341	<1e-9
1 KB	cubic	8.302	0.994	0.375	<1e-9
10 KB	sqr	0.317	0.031	0.191	<1e-9
10 KB	quad	11.240	0.913	0.429	<1e-9
10 KB	cubic	11.240	0.903	0.441	<1e-9
100 KB	sqr	0.206	0.030	0.094	<1e-9
100 KB	quad	7.325	1.023	0.161	<1e-9
100 KB	cubic	7.325	0.996	0.204	<1e-9
1 MB	sqr	0.257	0.020	0.266	<1e-9
1 MB	quad	9.130	0.714	0.265	<1e-9
1 MB	cubic	9.130	0.712	0.268	<1e-9
10 MB	sqr	0.195	0.015	0.283	<1e-9
10 MB	quad	6.925	0.497	0.363	<1e-9
10 MB	cubic	6.925	0.498	0.362	<1e-9
100 MB	sqr	0.115	0.012	0.179	<1e-9
100 MB	quad	4.076	0.363	0.355	<1e-9
100 MB	cubic	4.076	0.353	0.392	<1e-9
1 KB	sqr	0.309	0.032	0.166	<1e-9
1 KB	quad	10.977	1.020	0.347	<1e-9
1 KB	cubic	10.977	0.999	0.374	<1e-9
10 KB	sqr	0.338	0.030	0.219	<1e-9
10 KB	quad	12.004	0.836	0.520	<1e-9
10 KB	cubic	12.004	0.772	0.590	<1e-9
100 KB	sqr	0.318	0.026	0.250	<1e-9
100 KB	quad	11.279	0.745	0.508	<1e-9
100 KB	cubic	11.279	0.677	0.593	<1e-9
1 MB	sqr	0.389	0.022	0.405	<1e-9
1 MB	quad	13.814	0.753	0.458	<1e-9
1 MB	cubic	13.814	0.649	0.597	<1e-9
10 MB	sqr	0.507	0.023	0.516	<1e-9
10 MB	quad	17.988	0.645	0.701	<1e-9
10 MB	cubic	17.988	0.624	0.721	<1e-9
100 MB	sqr	0.384	0.031	0.259	<1e-9
100 MB	quad	13.620	0.598	0.775	<1e-9
100 MB	cubic	13.620	0.519	0.831	<1e-9

Table 3: Regression table for the quardatic terms in the models `sqr`, `quad`, and `cubic`.

size	term	df	sumsq	meansq	F.value	adj.p.value	signif
1 KB	erasure	4	301.328	75.332	120.235	0.000	***
1 KB	strategy	1	0.150	0.150	0.239	1.000	-
1 KB	erasure:strategy	3	5.664	1.888	3.013	0.088	-
1 KB	Residuals	801	501.858	0.627			
10 KB	erasure	4	377.922	94.481	191.174	0.000	***
10 KB	strategy	1	27.081	27.081	54.796	0.000	***
10 KB	erasure:strategy	3	8.133	2.711	5.485	0.003	**
10 KB	Residuals	801	395.864	0.494			
100 KB	erasure	4	227.138	56.784	95.946	0.000	***
100 KB	strategy	1	105.424	105.424	178.131	0.000	***
100 KB	erasure:strategy	3	2.378	0.793	1.339	0.781	-
100 KB	Residuals	801	474.060	0.592			
1 MB	erasure	4	224.734	56.184	127.908	0.000	***
1 MB	strategy	1	219.511	219.511	499.739	0.000	***
1 MB	erasure:strategy	3	12.915	4.305	9.801	0.000	***
1 MB	Residuals	801	351.840	0.439			
10 MB	erasure	4	431.403	107.851	373.493	0.000	***
10 MB	strategy	1	25.875	25.875	89.607	0.000	***
10 MB	erasure:strategy	3	101.219	33.740	116.843	0.000	***
10 MB	Residuals	774	223.502	0.289			
100 MB	erasure	4	475.064	118.766	672.019	0.000	***
100 MB	strategy	1	27.569	27.569	155.993	0.000	***
100 MB	erasure:strategy	3	164.806	54.935	310.843	0.000	***
100 MB	Residuals	801	141.561	0.177			

Table 4: ANOVA tables for fitting the two-way ANOVA model time $\sim$ erasure + strategy + erasure : strategy for each file size category. The `signif` column offers an at-a-glance overview of the p-values, with $p < 0.001$ receiving three stars, $p < 0.01$ two stars, and $p < 0.05$ one star. All p-values have been adjusted for multiple comparisons via Bonferroni correction.

size	contrast	estimate	conf.low	conf.high	adj.p.value
1 KB	RACE-DATA	-0.0288472	-0.1673791	0.1096846	1
10 KB	RACE-DATA	-0.3878784	-0.5109144	-0.2648423	<1e-9
100 KB	RACE-DATA	-0.7653045	-0.8999450	-0.6306640	<1e-9
1 MB	RACE-DATA	-1.1043122	-1.2203052	-0.9883192	<1e-9
10 MB	RACE-DATA	-0.3857145	-0.4816553	-0.2897737	<1e-9
100 MB	RACE-DATA	0.3913552	0.3177801	0.4649303	<1e-9

Table 5: Results from Tukey's honest significant difference tests for each file size category, filtered for only the contrast between the DATA and RACE retrieval strategies. The p-values have not only been corrected in the standard way for Tukey's test, but further adjusted to account for testing across six different file size categories.

Figure 9: 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. The box plots for a given file size are displayed side-by-side to reduce their overlap, but they are understood to all correspond to one file size.

Erasure level	Parity chunks (out of 128)
NONE	0
MEDIUM	9
STRONG	21
INSANE	31
PARANOID	90

Table 6: Parity chunks per every 128-chunk block, for each erasure level.

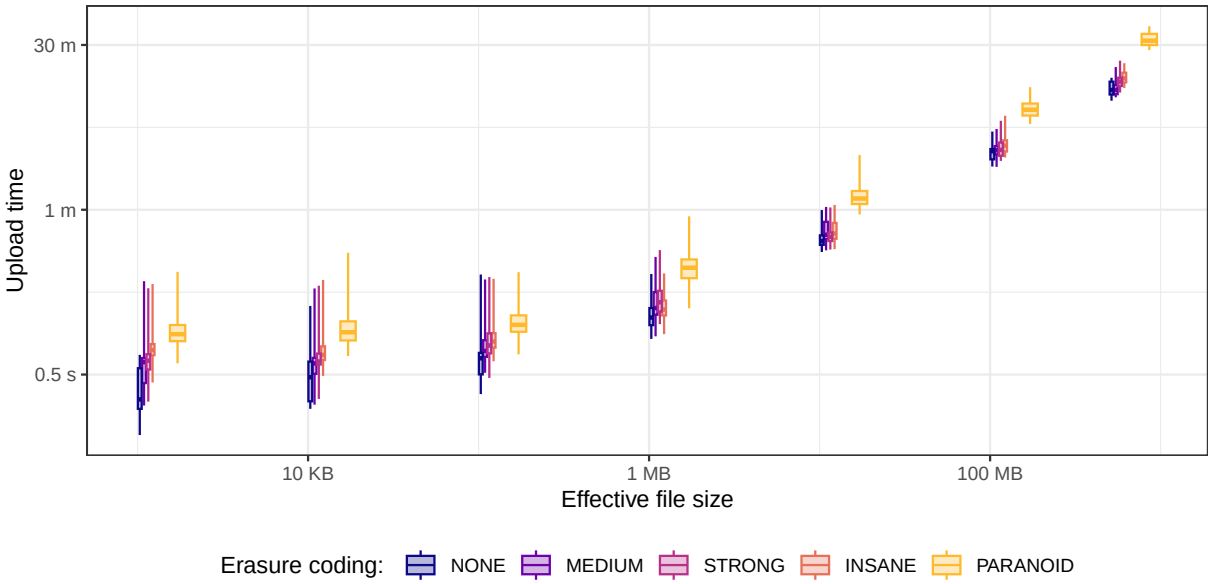

Figure 10: As Figure 9, but with the effective size of each file calculated, and the box plots placed exactly where they belong across the x-axis.