Storing high-performance computing data using Storj Decentralised Cloud Storage

Dr. Antonin Portelli (The University of Edinburgh) - 13th of October 2021

Disclaimer: This report was written as a personal research initiative from the author, in discussion with Storj. Storj contributed bandwidth and storage free of charge to perform the benchmarks detailed below. This report cannot be shared or modified without explicit consent from the author. This report does not constitute in any way an endorsement of Storj or services provided by Storj by the funding agencies and institutions supporting the author. The results in this report should not be interpreted as an absolute or reliable measurement of the total bandwidth available from the systems and services involved in the data transfers.

Acknowledgements: The author warmly thanks the Storj team for the numerous interesting technical discussions and insights around this study. The author would also like to thank Dr. Oliver Witzel from the University of Siegen for running the download tests on systems in the USA.

Introduction and objectives

A rapidly increasing number of fields in fundamental science are generating large datasets for their research, and questions around the availability of this data on the long term are becoming more and more critical. Two important underlying aspects are the resilience and geographical availability of the data. Regarding the former, data related to published scientific work generally have a significant cost to be generated, and it is crucial to insure that it can be stored safely for the next decade and possibly more. Additionally, scientific collaborations generally have an important international dimension, and there is some interest for the data to be available as independently as possible from the original upload location.

Emerging decentralised technologies, where data is fragmented and distributed across a decentralised network of nodes, are appealing to address the aforementioned challenges. Indeed, they can in principle allow for a much higher redundancy and geographical spread of the data, allowing strong resilience and wide availability compared to traditional, centralised datacentre-based infrastructures.

Over the past years Storj developed an enterprise-grade decentralised cloud storage (DCS) service based on a decentralised network of voluntary *node operators* containing the data fragments, coordinated by Storj *satellites* hosting the metadata. The storage node network is entirely trust-free, i.e. the data is fully encrypted using keys belonging to the owner, and cannot be accessed in any way by Storj or the node operators. At the time of writing this report, the Sort DCS network was made of around 12,000 nodes offering around 30 PB of storage (cf. dashboard here).

The main objective of this study is to qualitatively address the efficiency of Storj DCS for synthetic data simulating typical large datasets used in HPC, using lattice quantum chromodynamics (QCD) as a representative example. All the code used to perform the benchmark described below is available freely (GPL v3) on GitHub. This code only rely on a cloud service being configured in Rclone, and can be used to benchmark any cloud service compatible with Rclone (cf. complete list here).

Procedure

The test procedure goes as follows. First, one generates collections of files of equal sizes filled with random data. File sizes are generally above 1GB, simulating the typical size of gauge field configurations in lattice QCD. The files are then uploaded to Storj DCS from the DiRAC Tesseract supercomputer at the University of Edinburgh. The files are then downloaded from a number of other supercomputing centres in the USA and the download speed is measured. The experiment is repeated by manually splitting the files in chunks of 64MB and reconstructing them after download. During all the tests, file checksums are also uploaded and downloaded, and data file integrity is verified after download.

One important detail to mention is that the systems where the tests are performed constitute in no way an ideal benchmark environment. The tests are executed from user space with no control on potential limits implemented by the computing centres and possible contention with other transfers. The aim is not to investigate the ideal limits of Storj DCS in term of performances, but rather to provide information on possible performances in a realistic HPC scenario. Finally, all centres used in the test all have access to state-of-the-art multi-GB/s access to the Internet.

Synthetic data generation

A large file filled with random data can be created efficiently by generating a random passphrase, and then encrypting a stream of zeros using this passphrase. This can be achieved using OpenSSL and the `dd` command in the following way

```
PASS=$(dd if=/dev/urandom bs=128 count=1 2>/dev/null | base64)
dd if=<(openssl enc -aes-256-ctr -pass pass:"${PASS}" -nosalt </dev/zero 2>/dev/null) \
   of="${FILENAME}" bs=1M count=${SIZEMB} iflag=fullblock
```

where `FILENAME` is the generated file name and `SIZEMB` its size in MB. A checksum for the file is then computed using the high-performance XXH128 hash

```
xxh128sum ${FILENAME} > ${FILENAME}.xxh128
```

File chunking

In order to have more manual control on file transfer parallelisation, data files can be split into chunks manually before upload using the `split` command

```
split --verbose -a 4 -b ${SIZE} -d "${FILE}" "${FILE}."
```

where `SIZE` is the size of one chunk. A data file can be reconstructed from its chunks simply using the `cat` command

```
cat $(find . -name "${FILE}.*" | sort) > ${FILE}
```

assuming only chunks are captured by the pattern in the `find` command, more filtering might be needed in practice. The XXH128 checksum is only considered on the full file, therefore also checking that the file reconstruction did not corrupt the data.

File transfer

Upload and download of files on Storj DCS are performed using the Rclone utility. A native Storj DCS remote is configured in Rclone (using the storage type with the legacy name `tardigrade`). Files are uploaded to a `cloud-benchmark` bucket with high verbosity and a linear log for further data analysis.

```
rclone mkdir "${REMOTE}":cloud-benchmark
rclone -vv --stats 1000ms --stats-one-line ${RCLONE_EXTRA} copy ${DIR} \
  "${REMOTE}":cloud-benchmark/ensemble
```

where `DIR` is the directory containing the files to upload, `REMOTE` is the name of the Storj DCS Rclone remote, and `RCLONE_EXTRA` are potential user-provided options for Rclone.

Geographical location analysis

NB: The procedure described here is not functional anymore after the spring 2021 major upgrade of Storj DCS. The share links now only contain an image of the distribution map, and the locations are absent from the page's source code.

Storj does not provide an explicit API to access the geographical location of the decentralised file fragments [TODO: is that right?]. However, the Uplink utility can generate a URL to share a given file, and the associated web page display a map of the fragment locations. The explicit array of latitudes and longitudes of the fragments are given explicitly in the Javascript code of the map. One can then automatise the retrieval of the locations by parsing directly the page code and converting them into a space-separated value file.

```
URL=$(uplink share --url  sj://cloud-benchmark/${FILE} | grep -E '^URL' | awk '{print
$3}')
curl -sL ${URL} | grep 'var routes' | grep -Eo '\[[^ ]+\]' > ${LOC}
sed -E 's/\[?\{"Latitude":([0-9.-]+),"Longitude":([0-9.-]+)\},?\]?/\1 \2\n/g' ${LOC} >
${FILE}.loc.dat
```

Data transfer performances

Adopting the strategy described above we generated 10 files of 3.8 GB each filled with random data. The dataset was duplicated by splitting each file into 60 chunks of 64MB. This chunk size was communicated as optimal by Storj engineers. For every transfer test, the data was transferred in its original form, and then the transfer was repeated using the chunked versions to take advantage of parallelisation.

In all transfers described in this section, we used the option `--transfers ${CPU}` for Rclone (substituting `RCLONE_EXTRA` above), where `CPU` is the number of CPU cores on the server (generally 24 or 32). This option allows to transfer several files in parallel and increase the throughput.

All the tests described here were performed between January and February 2021.

Upload

The synthetic data was uploaded from the DiRAC Tesseract supercomputer at the University of Edinburgh, using the Storj EU1 satellite. The results are plotted in Fig. 1. We observed a significant benefit from parallelising the transfer over a chunked dataset, leading to higher upload rates by more than a factor of 6 (roughly 650 MB/s vs. 100 MB/s).

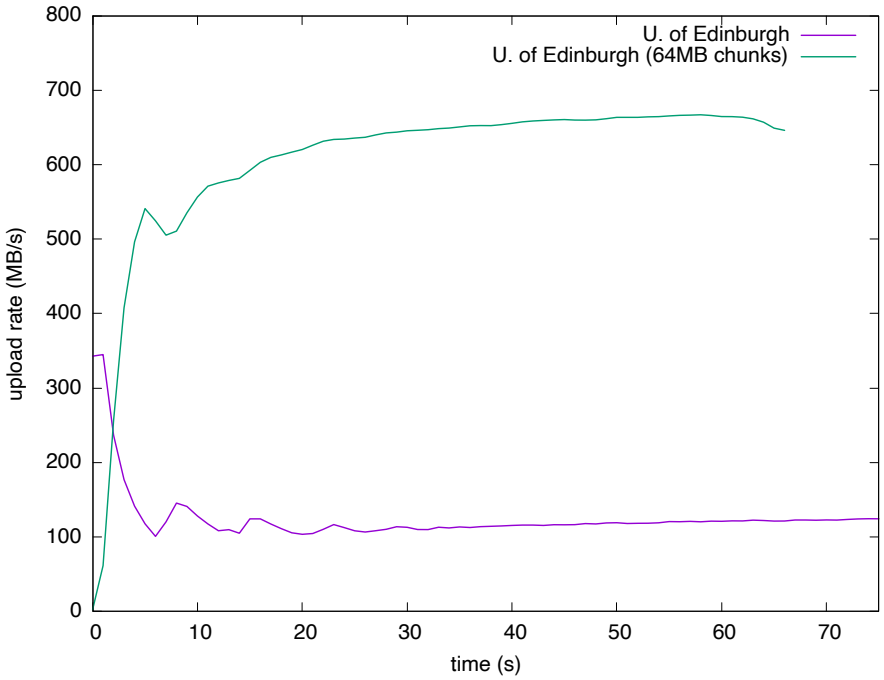


Figure 1 - Upload rates to Storj DCS. For better visibility the time interval was zoomed to the duration of the faster chunked transfer. The plain transfer maintained stable performances until it was completed.

Download

The data uploaded from Edinburgh was then downloaded from various supercomputing systems in the USA, and also in Edinburgh. The results for are plotted in Fig. 2 for the original dataset, and in Fig. 3 for the chunked dataset. Unsurprisingly, the best rates were obtained from Edinburgh, likely benefitting from some locality of the data in Europe. The impact of chunking appears to be more marginal than for upload rates. More precisely, chunking seems to improve download rates by 10%-30% depending on the location. Overall, maybe at the exception of Oak Ridge National Laboratory, we obtained healthy rates of several of hundreds MB/s.

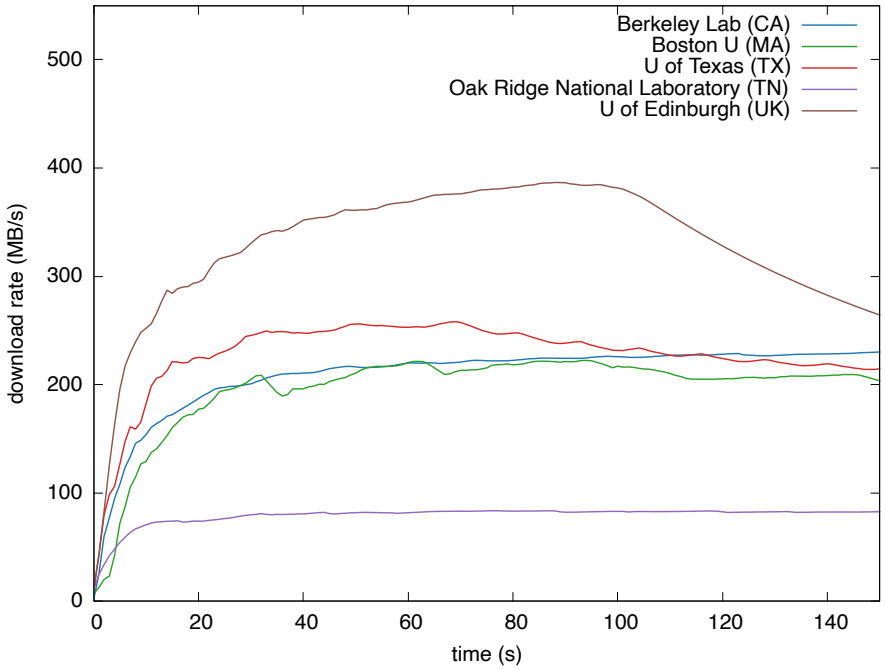


Figure 2 - Download rates from Storj DCS for the original dataset. A download test was executed in Fermilab but failed without being repeated.

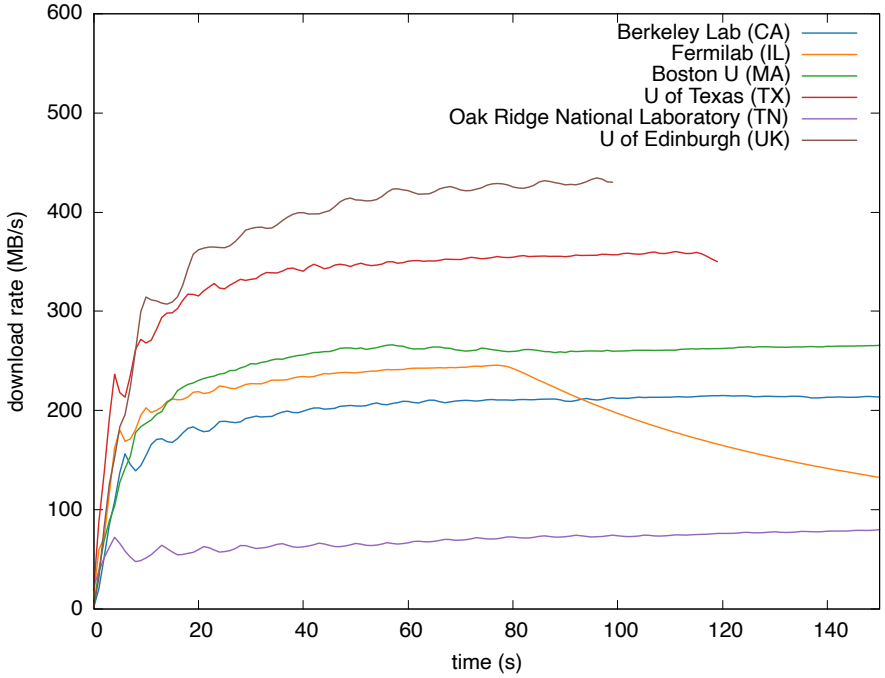


Figure 3 - Download rates from Storj DCS for the chunked dataset.

Parallelisation strategies

During the course of this study, Storj engineers implemented native parallelism as part of the Uplink library provided with their services. This allows to transfer a single file using parallel transfers without needing to split the file as done above. Using a pre-release version of the command-line tool `uplink` provided by Storj, we performed new benchmarks and compared to the manual file splitting described above. These tests were performed form the DiRAC Tursa supercomputer in Edinburgh.

The server used has a dual-socket AMD EPYC 7H12 which is a total of 128 cores and 1TB of RAM. The server and network activity were essentially idle at the moment of the test, constituting a clean benchmark environment. We generated a single 128 GB file filled with random data as described above. It was then transferred to and from Storj DCS using the `uplink` command and the native parallelism flag `--parallelism`, and then chunking the file in 64 MB fragments and parallelising through multiple transfers using Rclone as done previously. The upload performances are shown in Fig. 4. One can observe similar performances between native parallelism and manual splitting of the file, with marginally higher figures for manual splitting. When using a high number of transfers with the `--parallelism` flag, we observed the transfer becoming unstable and crashing, likely due to a disruptively large CPU activity. Finally, the download performances are displayed in Fig. 5. In that case, splitting the file manually still provides about two times the performances of native parallelism.

In conclusion, the native parallelism implemented in the Storj software stack allows for impressive upload & download rates in the 300-400 MB/s range for a single file transfer. In the case of download rates, splitting the file manually is still more performant and reach rates above 700 MB/s. However, it is important to say that splitting the file requires I/O intensive splitting and reconstruction operations which have to be systematically executed when a file is sent or retrieved from the cloud. Our conclusion is that it is very likely that optimal performances and convenience will be obtained using Storj native parallelism in combination with multi-file parallel transfers. This will be possible to achieve when native parallelism options will be exposed in Rclone, which is currently being implemented.

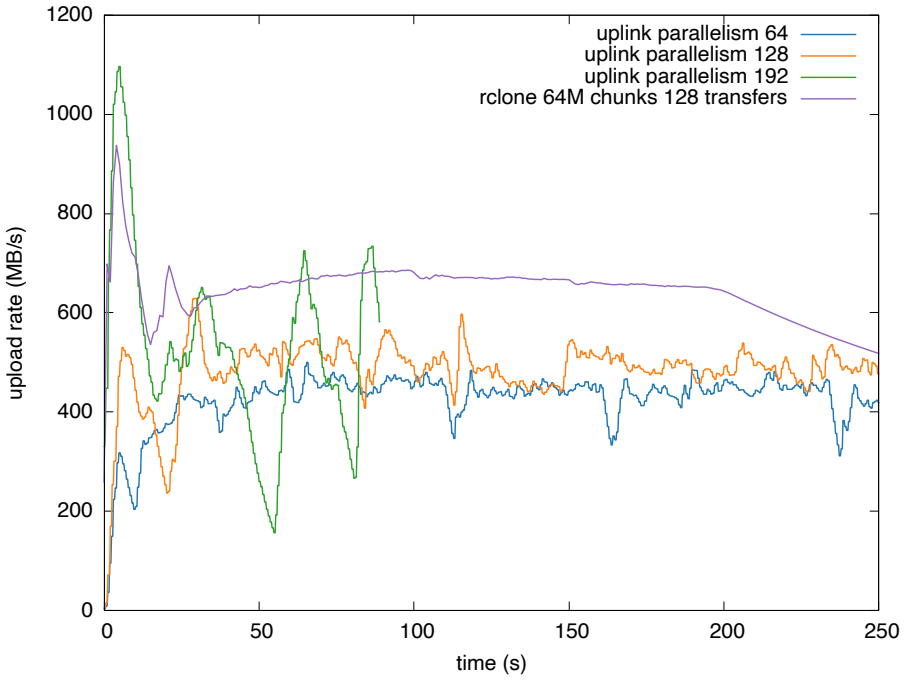


Figure 4 - Upload rates to Storj DCS comparing the Storj Uplink program with native parallelisation to chunking the file as described previously.

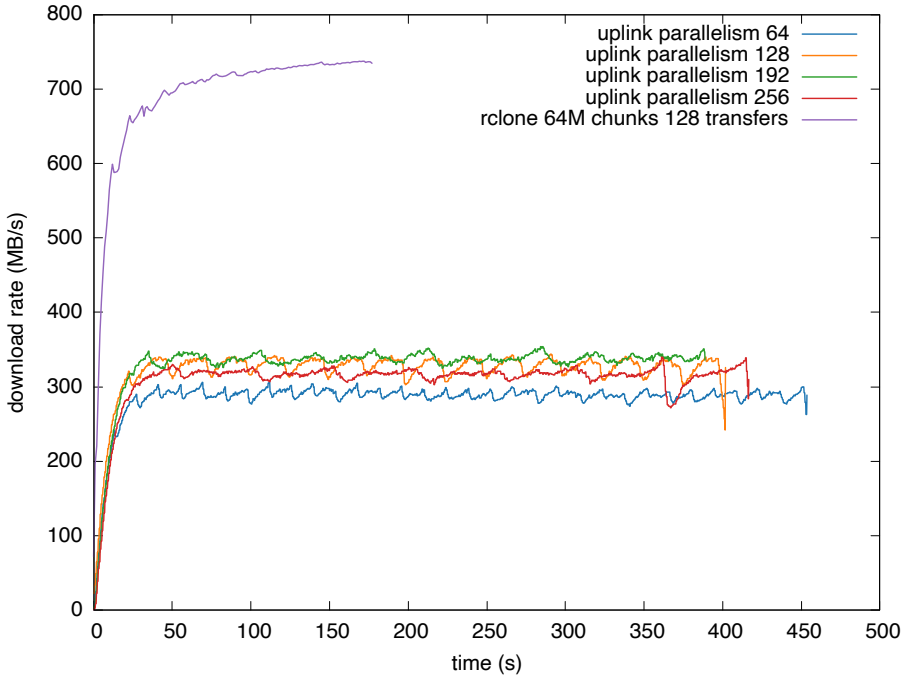


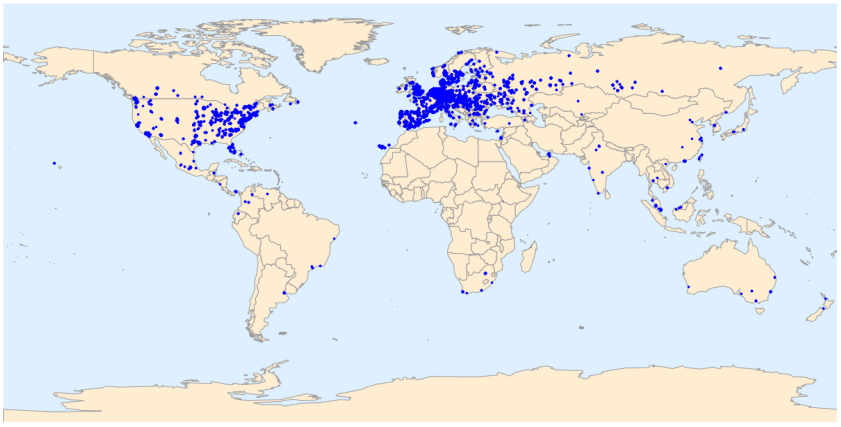
Figure 5 - Download rates from Storj DCS comparing the Storj Uplink program with native parallelisation to chunking the file as described previously.

Geographical distribution analysis

Using the method described above, we extracted the geographical locations of the file pieces on Storj DCS network. In all this section we focus on the original dataset (i.e. not the chunked data). The chunked dataset showed similar properties and is significantly longer to analyse due to the higher number of pieces on the network. The geographical analysis presented here was performed using Wolfram Mathematica.

The aggregated locations of all the pieces for all the files is plotted in Fig. 4. We counted a total of 30284 pieces on the network. We observe that a large number of pieces are distributed fairly uniformly over Central Europe. The USA distribution features a large number of pieces highly concentrated on urban zones. In Fig. 5 we plot the distribution of these pieces by countries, and by states for the USA.

Finally, in Fig. 6 we plot the average download rate (after stabilisation) against the average geodesic distance between the download location and the locations of all the file pieces on the Storj DCS network. One would expect anti-correlations to be observed, i.e. low average geodesic distances implies more file pieces available locally and therefore higher download rates. A trend can be marginally be observed in the plot, however higher statistics would be necessary to eliminate uncontrolled systematic deviations.



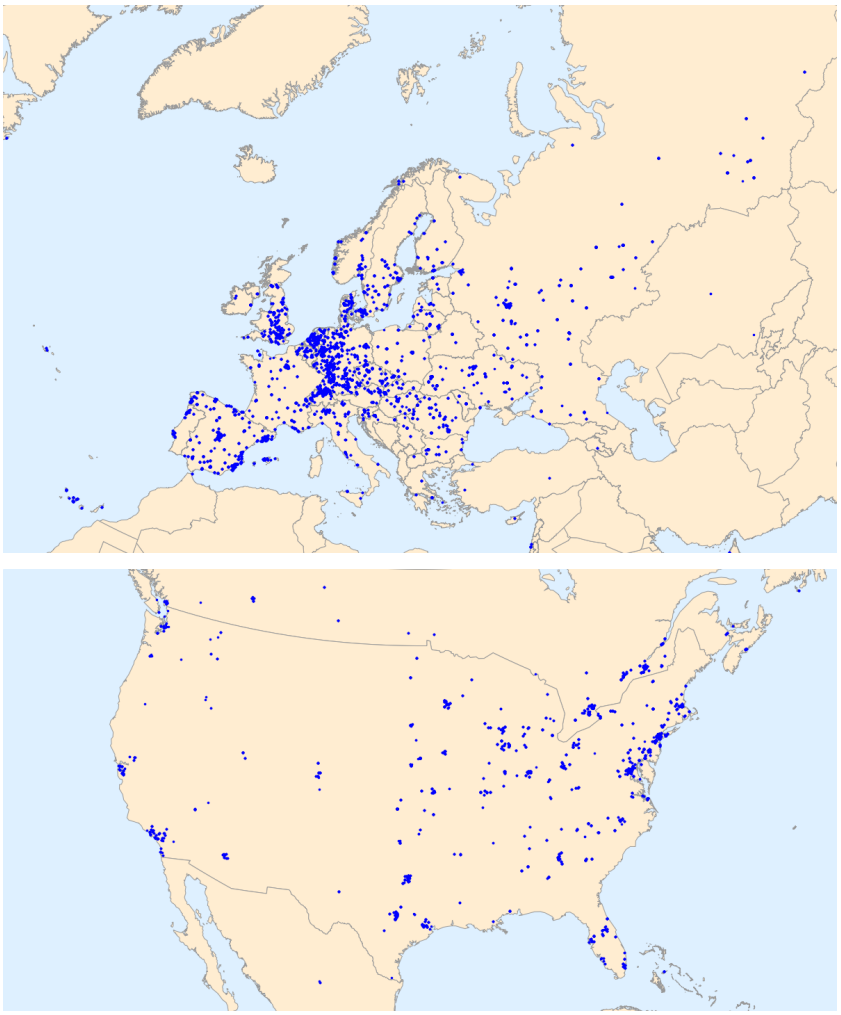
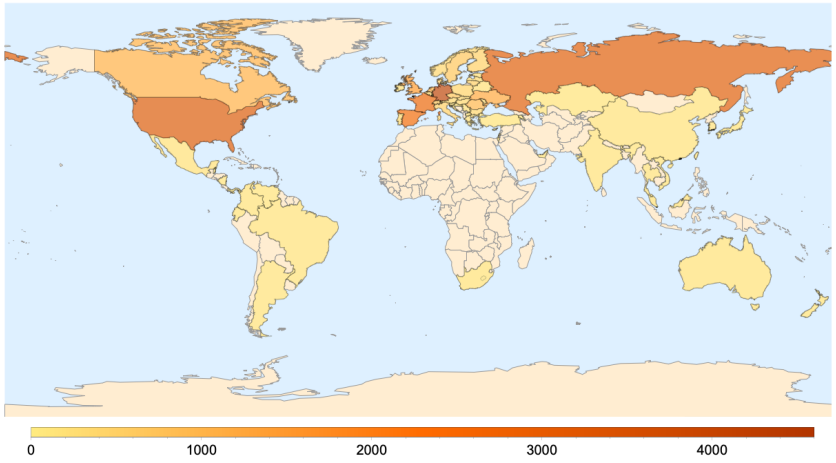


Figure 4 - Geographical locations of the datafile pieces on Storj DCS for the original dataset, with zooms over Europe and the continental USA.



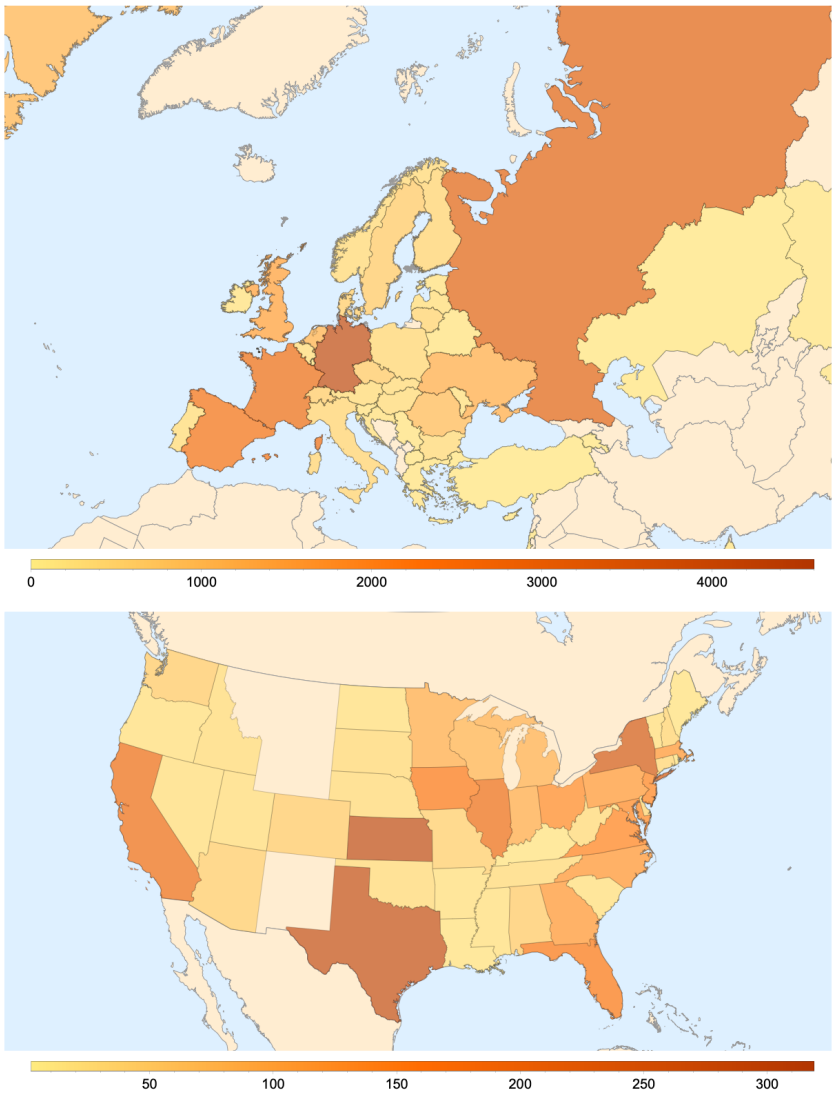


Figure 5 - Geographical location count of the datafile pieces on Storj DCS for the original dataset by country, and by state for the USA.

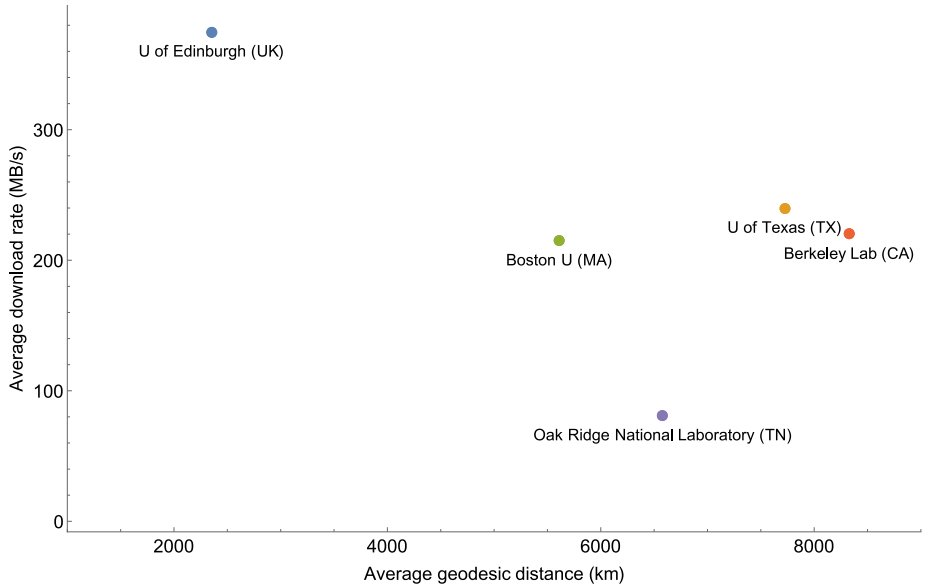


Figure 6 - Average download rate in MB/s vs. average geodesic distance between the download location and file pieces locations on Storj DCS.

Outlook and perspectives

In this study we measured the upload and download rates of large data files from major supercomputing centres using Storj DCS. The data was upload from Edinburgh in Scotland and retrieved from a number of sites in the USA. We managed to push the upload rates to above 600 MB/s by fragmenting manually the files and parallelising the transfers. Retrieving the files gave download rates between 80 MB/s and 450 MB/s depending on the location, although in most cases sustained performances above 200 MB/s are observed.

Comparing these performance figures to more traditional, centralised cloud infrastructures is delicate because of the radically different properties of both strategies. However it is safe to state that these rates are overall quite impressive, especially considering that thanks to the decentralised nature of the network, there is no need for researching an optimal network path between the source and the destination.

From an HPC infrastructure point of view, these rates might look inferior to multi GB/s obtainable with HPC data centres on dedicated national or international research networks with fast links to computing centres. However this statement needs to be balanced against a number of unique and important advantages provided by a decentralised network. Firstly, high transfer speeds on dedicated networks might fall off abruptly as soon as the data is transiting through a point outside of the network, and finding fast paths to transfer data between sites might be non-trivial and highly dependent on geographical and infrastructure constraints. Secondly, a decentralised storage infrastructure like the one deployed by Storj is highly resilient and Byzantine-fault tolerant, which is generally not the case of a traditional datacenter infrastructure. Regarding this last point, at the moment although the storage of data is indeed fully decentralised and Byzantine-fault tolerant, users still need to query the metadata to centralised satellites currently run by Storj. However, Storj is actively working to achieve full decentralisation of their network in the future. Additionally, Storj is currently working on improving transfer parallelisation in their native software stack, and the rates reported here can potentially be considerably increased in the future.

In conclusion, because of the inherent decentralised and international nature of scientific collaborations, one can reasonably expect that emergent decentralised technologies have the potential to be very suitable to support global scientific efforts. We found the decentralised storage services proposed by Storj to already demonstrates impressive availability and performances across large distances, and we expect significant improvements to be implemented as the technology evolves. Because of its high resilience and geographical availability, this type of service appear suitable for data curation, long-term storage and sharing of published scientific data, perhaps as a layer above more traditional storage focused on very-high transfer rates for large, internal data involved in the scientific calculation themselves. This study entirely focuses on performances and technical aspects, although we acknowledge that in practice cost-efficiency is an additional, crucial factor to consider, which could be quantified at a later stage.

For the future, it would be very interesting to continue studying Storj decentralised cloud as the technology evolves. Storj started integrating an S3-compatible backend in satellites which will even further increase the availability of the data. Additionally, a number of things can still be done in term of exploring opportunities for parallelising transfers, and Storj is actively exploring that. Finally, we found the perspective of a fully decentralised network to be exciting as it will allow for domain-specific satellites to be deployed, which is potentially very suitable for scientific data curation.