Brock's Guide to Data Archiving

This is a collection of tips and tricks and is not meant to address curated archiving, for that contact the Library about Deep Blue, or domain specific repositories such as ICPSR.

The goal in this document is to lower the cost of preserving digital data by making trade offs in ease of use and / or performance.

Finding files on Linux & Mac with find

Linux and Mac provide a very flexible tool `find`. This tool can be used find objects that match specific criteria of the object such as:

- Is a file, directory, socket, etc.
- User or group that owns the file
- Permissions of the file
- Last access, modify, or change of the file
- Size of the file

Basic Find

Find all files in current directory are owned by user labmanager

```
find . -user labmanager -type f
```

Find all files greater than 100MBytes

```
find . -type f -size +100M
```

Find all files last accessed less than 60 days ago and owned by group comp-lab

```
find . -type f -atime -60 -group comp-lab
```

Note on greater than, less than and equal

Find has many options where you may want to select all files larger than X or last accessed in last 30 days etc. This is a common mistake as the greater and less than options are not inclusive:

- `-size +60M  # Over 60 Megabytes`
- `-size 60M    # Exactly 60 Megabytes`
- `-size -60M   # Less than 60 Megabytes`

Using find to run a command on each matched file

Find provides the `-exec` command to run on each file. Files are not passed as a list but each match is passed one at time. The file is substituted where `{}` appears. The command needs to be terminated with `\;`

Delete all files owned by `olduser`,

```
find . -type f -user olduser -exec rm -f {} \;
```

Delete empty directories owned by `olduser`,

```
find . -type d -user olduser -empty -exec rmdir {} \;
```

Compression

The simplest way to save on storing data that is currently not needed is to compress it. Compared to other options compression is trading time & complexity for size.

There are many different compression methods. Some are specific to the sort of data (eg cram for genomics) and can achieve better compression or speed but only for a specific data type. When selecting a compressor think about the ubiquity of the algorithm for anyone who may need to access the data in the future, as well as how well tested it is to trust your data. Corrupted compressed data behaves like encrypted data and often cannot be recovered.

Linux / Mac Compression - gzip, bzip2, xz, lz4

Command	Suffix	Availability	Speed
`gzip`	`.gz`	High	Fast
`bzip2`	`.bz2`	High	Medium

The above are the most common compression tools and are available by default on most Linux and Apple systems. Other tools gaining favor but are not as ubiquitous are `xz/lzma` which is slower but provide better compression than `bzip2` and `lz4` which is much faster but gives worse compression.

Decompress `myseq.fq.gz` with `gzip`,

```
gzip -d myseq.fq
```

Compress `myseq.fq` creating `myseq.fq.bz2` and leave the original file (default deletes original).

```
bzip2 --keep myseq.fq
```

Parallel Versions of gzip, bzip2, lzma/xz, lz4

The most popular compression programs in large scale research are `gzip`, `bzip2`, `lzma/xz`, and `lz4`. Most versions of these you will find installed on common platforms only support single CPU cores. The following are fully forwards and backwards compatible implementations of the

classic single threaded compressor that in many cases provides significant time saving on any modern multi-core system.

Command	Compatible	Parallel Compress	Parallel Decom.	[1]Speed vs. Gzip	Gzip Size 153 G
`gzip`	gzip	No	No	1x	153G
`pigz`	gzip	Yes	No	32x	153G
`lbzip2`	bzip2	Yes	Yes	23x	151G
`mpibzip2`	bzip2	Yes	Yes	*	151G
`xz -T0`	xz/lzma	Yes	No	5.5x	137G
`pixz`	xz/lzma	Yes	Yes	5.5x	137G
`zstd`	zst	Yes	Yes	67x	155G
`lz4`	lz4	No	No	42.2x	171G

Notes:

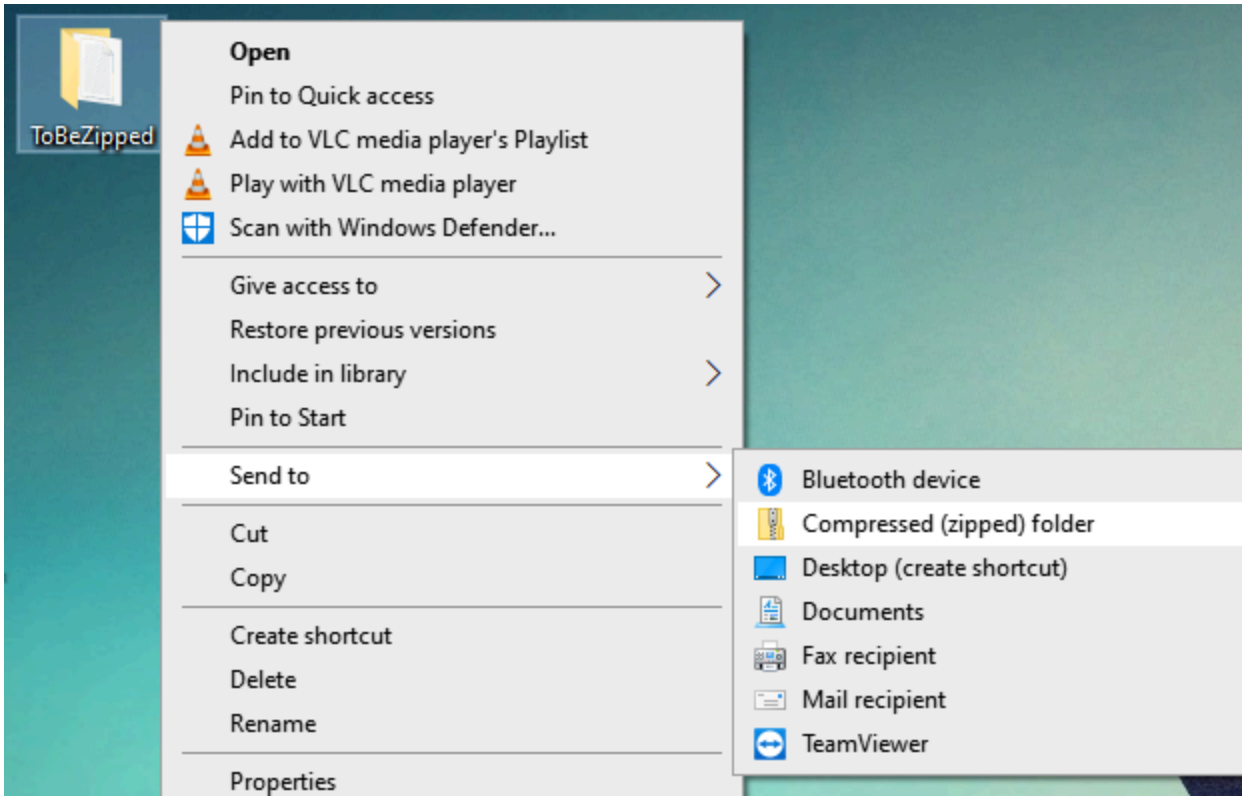
- `pigz` can only compress in parallel with very minimal speedup on decompression
- `xz` requires `-T0` option to use all cores in the system or will default to 1
- `xz` cannot decompress files in parallel but `pixz` can
- `lbzip2` and `mpibzip2` can only decompress in parallel if the archive was compressed with a parallel aware compressor
- `lz4` is not parallel aware but is by far the fastest compressor of all, but with the least space savings

Zip Archive/Compression Tool (Windows)

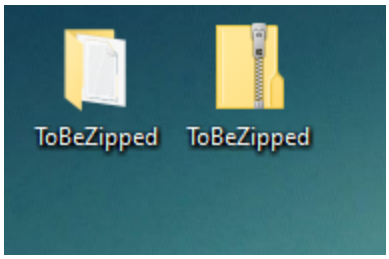
Windows has a built in zip archive tool. You can use it to take a folder and turn it into a single .zip file which can then be moved to an archive like Data Den.

[1] Example Data 222 GByte backup of a Box Account on 36 core Great Lakes node
Speedups are relative to serial gzip, parallel versions of slower compressors (xz, bzip2, etc) will be faster compared to their serial version

Use is simple and only provides a single operation of zipping an entire folder by right clicking on the folder to archive selecting “Send to” → Compressed (zipped) folder



This will create a new file that will have the icon of a zipper and a folder. It is a single zipped file, that can be opened on Linux systems with the `unzip myzip.zip` command or double click on Windows.



Tar Archive Tool (Linux, Mac)

Tar is the ubiquitous archive tool used in research. It provides a multitude of options. It can take a directory or list of files and turn them into a single optionally compressed file.

We will use cover the following options

Long form	Short Form	Description
`--create`	`-c`	Make a new tar archive
`--extract`	`-x`	Extract existing archive
`--file <file>`	`-f <file>`	Name of archive to create or extract
`--verbose`	`-v`	Verbose output (print files as tar goes along)
`--gzip`	`-z`	Compress/Decompress with Gzip
`--bzip2`	`-j`	Compress/Decompress with Bzip2

Basics - Create Archive

Use the `--create`, `--file` and `--verbose` options to create an archive out of `to_archive` folder and print what's happening

```
tar --create --file myarchive.tar to_archive
```

Basics - Extract Compressed Archive

Use the `-x`, `-f` and `-z` options to extract archive `myarchive.tar.gz` into the current folder not printing output

```
tar -xzf myarchive.tar.gz
```

Advanced Tar

Users tend to default to using a compressor with tar such as `-z` or `-j`. These options often if used on binary or already compressed data (eg. video or lossy images like jpeg) will drastically increase archive time with minimal savings in space. On very fast storage systems and networks such as those at ARC it can at times be better to not use compression with tar for the bulk of your data as it takes the most time.

Tar with Parallel Compressors

It is possible to tell tar to use compression programs it does not know about as long as they support common conventions. Parallel compressors provide significant performance advantages over the serial versions. All of the parallel compressors except `mpibzip2` can be used as below:

Compress with parallel bzip2 compression

```
tar -I lbzip2 -cf myarchive.tar.bz2 <directory>
```

Decompress with parallel bzip2 compression

```
tar -I lbzip2 -xf myarchive.tar.bz2
```

Saving an Index of Files in an Archive

Often we need to find where a specific file is. This can be solved two ways, printing the index of an existing tar, or saving a list when the tar is created. The first is very slow as tar does not store an index but must actually read every block of data. The second requires planning ahead.

Print all files in an existing tar file with `-tv`

```
tar -ztvf myarchive.tar.gz
```

Save Index when tar is created with `-v` and `tee`

```
tar -cvf myarchive.tar.gz <directory> | tee myarchive.txt
```

You can then search for files using simple `grep -i pattern myarchive.txt`

Expand tar file to a Different Location

Often it would be desirable to expand a tar that lives in one location (eg slower bulk storage) to a different location (fast storage). Often both locations are not big enough to hold the tar and the expanded version. Using the `-C` flag you can tell where to expand a tar

```
tar -xf myarchive.tar -C /fast/dir/
```

Extracting Specific Files from Tar

It is possible to extract specific files rather than expand an entire archive. This can avoid needing the full space to store the full expanded archive. This generally does not speed up the extraction process as the archive must be read at the start of the archive until the data is found.

Extract file `run2/data1`

```
tar -xvzf myarchive.tar.gz run2/data1
```

Extract entire directory `run2/`

```
tar -xvzf myarchive.tar.gz run2
```

Wildcards and complex pattern matching is possible.

Tar and Archives Bigger than Source Data (Sparse Files)

If you ever find a tar being significantly larger than the source data this is most often caused by sparse files. This normally only happens when using tar without a compression option as compressors will address the holes in sparse files.

If you don't want to use a compressor eg. to save time on a fast large filesystem, you can use the `--sparse` option to GNU tar the default on most systems to have tar not create empty space for sparse files. This does not in any way damage the data.

Archivetar - All In One Archive Tool

Archivetar is a tool that aims to address many of the common bottlenecks when archiving projects at scale. It's intended to prep an entire folder for upload to Data Den or AWS Glacier or similar low cost, but sensitive to file size / file count. It addresses performance by supporting most of the parallel compression tools as well as parallelism across lists of files. By avoiding single extremely large archives it makes management or extracting subsets of data much easier.

Great Lakes Three Line Archive to Data Den

```
module load archivetar
cd <folder>
archivetar --prefix my-archive --destination-dir
/<dataden-volume>/<folder>/

# wait for globus transfers on globus.org to complete, before
deleting originals.
```

Archivetar Examples

Create Backup of all Files, Compress with Gzip Leave Originals

```
# Prefix tar's with my-backup
# Each tar before compression should be at least 100G
# To include all files don't set --size
archivetar --prefix my-backup --tar-size 100G --gzip
```

Prep Directory for Data Den

While not required, often the largest files in the data set are already binary and benefit minimally from compression. But reading TBytes of data through a compressor for no space savings can be much slower than raw network & disk speed. This example includes only files less than 10GB in the archives, and deletes them after they are in the tar. We will also compress those small files. This will leave tar's of "small" files and the large files. The resulting directory can be directly uploaded to Data Den. The total number of files often reduced by 90% or more, but reduces the time to compress by 50% or more.

Steps:

1. Bundle small files into tars and save list of files with `archivetar`
2. Delete files in bundles with `archivepurge`
3. Upload entire folder archives + large files

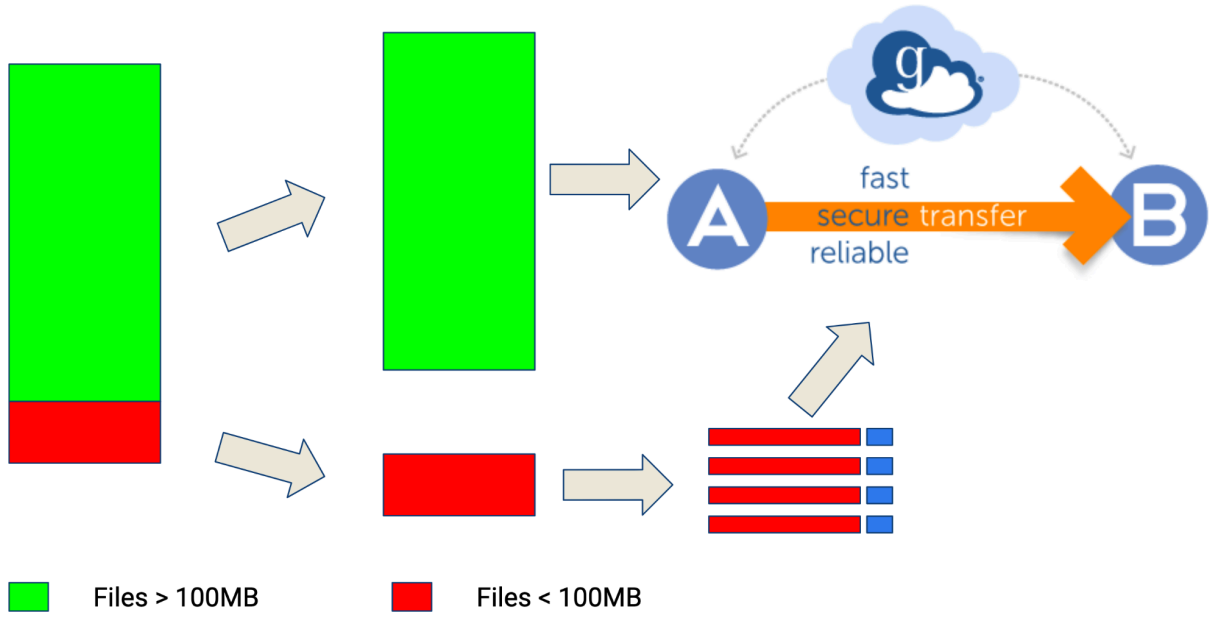
```
archivetar --prefix my-archive --size 10G --tar-size 100G --gzip
--save-purge-list

# use my-archive-<timestamp>.cache to delete files in archive
archivepurge --purge-list my-archive-*.cache
```

Auto Upload to Data Den

This example uses the Globus API to directly upload from Great Lakes to Data Den an already prepped version. When complete if just a backup you can delete all `my-archive*` files or if an actual archive, delete when the globus transfers finish.
NOTE you must wait for the transfers to finish. The tool will complete before the transfers do so you need to monitor them at globus.org.

If using the defaults `--source` and `--destination` are not required on ARC-TS systems unless not using the default values.

```
archivetar --size 10G --tar-size 100G --lz4 --prefix my-archive \
--source umich#greatlakes --destination umich#flux \
--destination-dir /flux-support-dd/brockp/archive-test/
```



Archive Full or Close to Full folders

By defult archivetar will create tars as fast as possible and leave the created tar. Thus ignoring compression archiving 10TB of data will require 20TB of space (origonal plus tars). This is not desirable for very large archives or when limited space is available. There are two ways to address this:

`--rm-at-files`
This tells archivetar to wait for a tar to be uploaded and then delete the local copy. Thus archivetar won't move onto the next tar until the current one is complete. This often means you can archive with less than 1TB of free space. This also is useful for automation because archivetar will leave a folder as though it was never there.

`--bundle-path <path>`
This tells archivetar to create all files (tar, index.txt etc) in an alternative location. A popular one is/scratch on the HPC clusters or /tmp/ on linux workstations. Thus allowing archiving without ever writing any data to the current directory being archived.

Unarchive Examples

`Archivetar` doesn't do anything special other than wrap regular `tar`. Thus all archives can be expanded using commands available on any workstation. `Unarchivetar` is a simplification similar to `archivetar` and uses parallelism to speed the process but is not required.

`Unarchivetar` has several options that allow it to be used as a backup. This means skipping files that already exist, or are newer, when expanding an archive.

Unarchive

```
# unarchive prefix my-archive
unarchivetar --prefix my-archive
```

Unarchive over top of existing data skipping files that already exist

```
unarchivetar --prefix my-archive --skip-old-files
```

Multiple tar/compressors in parallel -- GNU Parallel

If CPU is not limited, often invoking multiple compress/tars at once can reduce total runtime. The simplest way to invoke multiple at a time, is with GNU Parallel. The format of `parallel` is to create a list, a template and invoke that multiple times. By default it will invoke one template per core.

Compress every file in the current directory limiting to only 10 copies at a time with `pigz`.

```
ls | parallel -j 10 --progress pigz {}
```

Tar the directories `A B C` named `A.tar, B.tar`, etc.

```
parallel --progress tar -cf {}.tar ::: A B C
```

Parallel File Utilities, cp, rm, find, chmod - mpiFileUtils

Installed on ARC-TS Systems

```
module spider mpifileutils
```

`Mpifileutils` provides parallel versions of many common file manipulation utilities. As many modern filesystems are highly multi-threaded, having multiple IO requests can mask network latency and metadata locking issues. In addition because the tools use MPI they can be run in parallel across multiple systems aggregating their network capacity for shared file systems.

There are several tools of interest `dwalk, dfind, dcp, drm, dchmod`. They mostly work the same and you can use the `--output` and `--input` options to chain the results for `dfind` to `dcp` for example.

Basic find, copy, delete (dfind, dcp, drm)

Find all files owned by user `brockp` in `/home` not accessed in 180 days copy to new location, then, delete them using 12 cores at a time

```
mpirun -np 12 dfind --user brockp --atime +180 --output
filtered.cache /home
mpirun -np 12 dcp --input filtered.cache --progress 10
/destdir/
mpirun -np 12 drm --input filtered.cache
```

Find all files larger than 1G in ~ and print

```
mpirun -np 12 dfind --size +1GB --print ~
```

Finding Duplicate Files

Often it is useful to find if data has been duplicated. This is very intensive as all data in a path will be read. The parallel `ddup` will speed this up significantly by splitting the hash calculation across multiple cores at a time, possibly across many hosts. Eg. at ARC-TS we are able to sustain comparing well over 2 GB/s.

```
mpirun -np 12 ddup /path/to/walk/
```

Data Transfer & Sharing

Transfer -- Globus

We highly recommend using Globus.org for data transfer the benefits include:

- Full Mac, Linux, Windows Support
- Extreme performance
- High Reliability including server reboot and network changes mid-transfer
- No Firewall Ports Required

TODO Video Tutorial

Campus Globus Endpoints

ARC maintains a list with links: https://arc.umich.edu/globus/#document-4

Endpoint Name	Supports	Network Configuration

umich#hits-research	Adv. Genomics Core / HITS NAS	1 x 10 Gbps

Sharing -- Globus

Globus supports international data sharing. This does not require administrator intervention and supports fine grained user and group permissions based on email address or globus username. This functionality is similar to sharing folders in Google Drive where the recipient will be notified by email that a folder is now shared with them.

TODO Video Tutorial

Other Options

RClone

RClone is supported on all platforms and supports all the major cloud storage providers. This can often allow bridging data from services such as Google Drive and Box.

On ARC-TS Systems:

```
module load rclone
https://arc-ts.umich.edu/flux/software/rclone/setup-box/
```

On fast networks and storage increase number of parallel transfers:

```
rclone copy --transfers=32
```

Storage Solutions

Locker

Locker is considered ‘warm’ archive. It’s an all disk system capable of 1GByte/s often called near-line. Performance is not suitable for use with large scale computation or small file operations.

Common use cases for Locker are:

- Capturing new data from devices that will not be acted on immediately
- “Low” performance lab share
- Holding data that will be quickly moved to another location

Locker provides 1 Million file objects / 1 TByte of provisioned capacity. Minimum allocation 10 TByte

Data Den

Data Den provides a massive scale ‘cold’ archive. It’s a disk cached - tape system providing offline encrypted copies of data. Data Den is capable of ingesting data at 1GB/s, but recall performance is highly dependent on file size if data is not in cache.

Common use cases for Data Den are:

- Maintaining secondary copies of data of high value
- Maintaining data for long term preservation or publication
- Sharing data sets

Data Den access is only via Globus Transfer and Sharing
Data Den provides 10,000 File objects / 1 TByte of provisioned capacity
Data Den requires 99%+ of data to be in files > 100 MBytes Each
Minimizing the number of files is key for performance

Scanning Existing Data for Expected Archive Behavior

The tool `archivescan` produces a report of how much data is over 100MB or under. The goal for Data Den is that the total data volume (Bytes) is 98-99% on tape, rather than on disk. Active Archive relaxes this requirement.

Archivescan on ARC Systems

```
module load archivetar
cd <folder>
archivescan

----- Results ------
Data Den Candidates:
Files: 125
Size: 206.2GiB

Cache (Locker) Candidates:
Files: 6510
```

```
Size: 8.2GiB

Total Time 0.97 Seconds
Fraction Offline: 96.17703 %
```

Archivescan on non-ARC Systems

Requires Python3

```
curl -O https://raw.githubusercontent.com/brockpalen/archivetar/master/bin/archivescan
python3 ./archivescan
```

Data Den Optimal File Size

Data Den performance is heavily impacted by the number of files as the tape drives have to seek which can take up to 50 seconds per file. Thus most users will use `tar`, or other tool to bundle many files together into a single archive before uploading to Data Den.

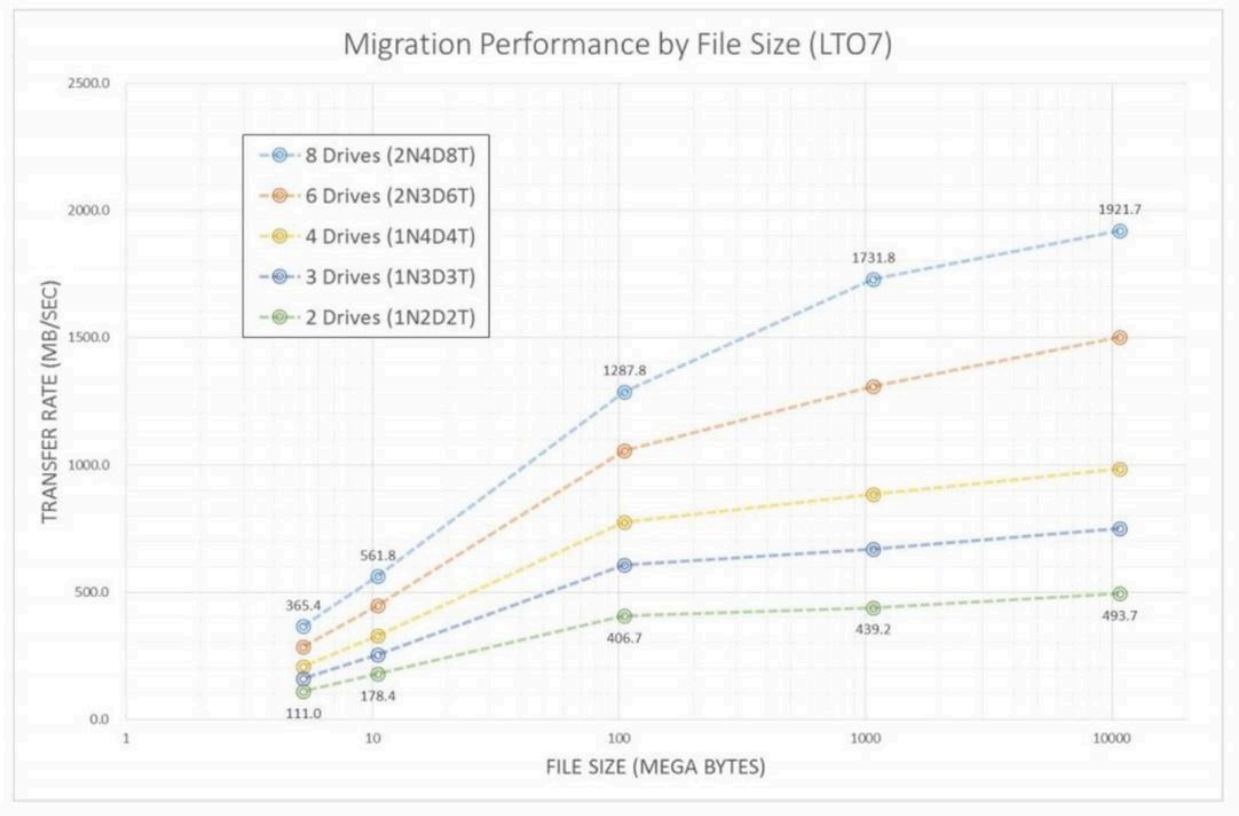
You may be tempted to create a single many TeraByte file. This is also sub optimal as it does not allow the system to use parallel drives to recall multiple files. Single file will only ever use 1 drive and a single 12 TByte tape may take up to 10 hours to read. Thus 12 1 TeraByte files will come back much faster.

Thus you want a file large enough that the 50 second seek time per-file is small compared to total time recalling the data, but not so large you are limited to a single drive only for all your data.

The general rules of file sizes are:

- Default Minimum 100 MByte[2]
- Optimal 20 GByte - 500 GByte (100G Recommended)
- Maximum 10 TByte

[2] The value can be tuned with consultation about appropriate use and expectations



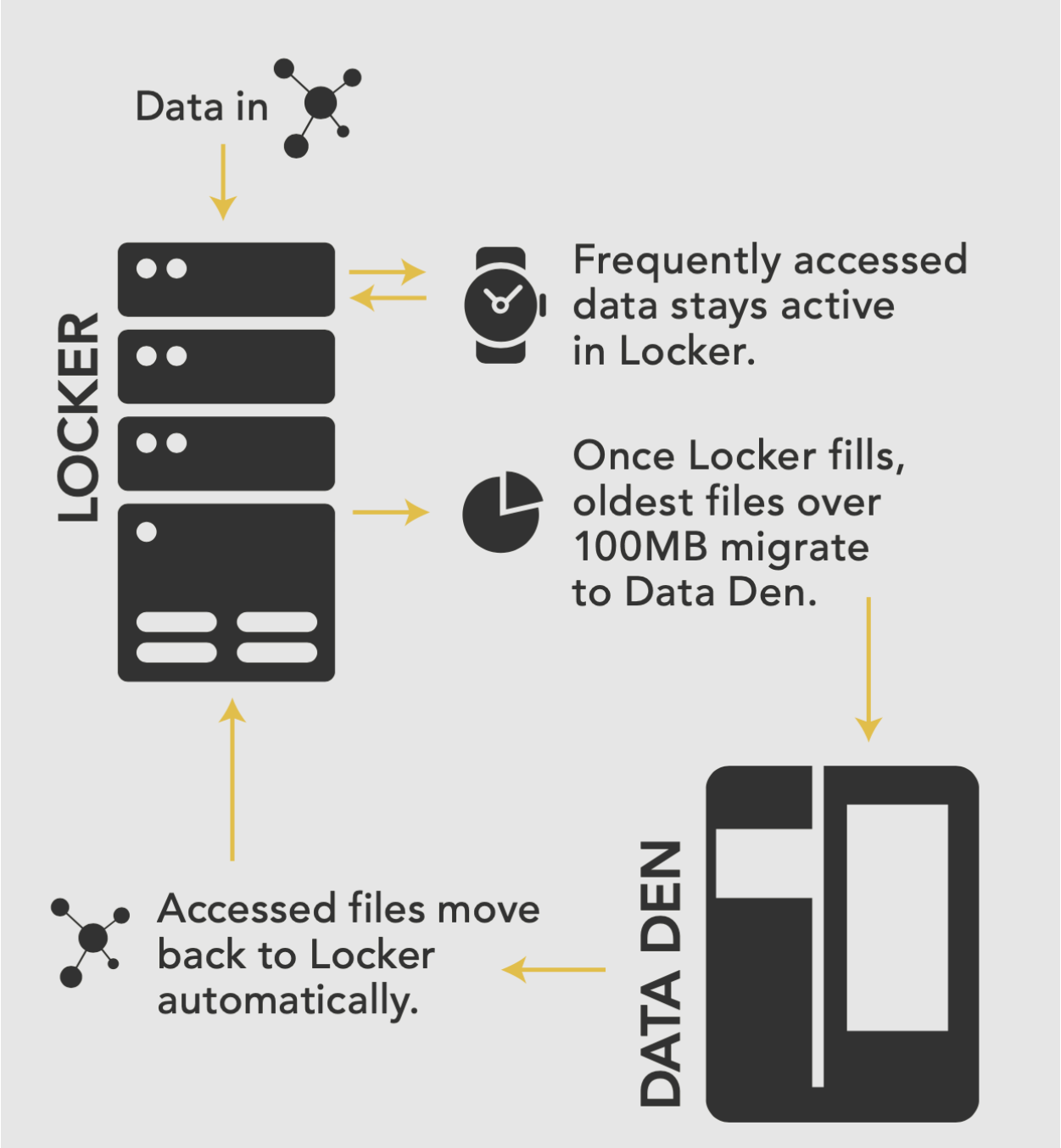
Data Den currently has 9 LTO8 Drives/Site

Advanced Data Den + Locker Active Archive

For the largest volumes of data (100 TByte minimum) it is possible to provision dedicated disk space in Locker with Data Den and provide support for small files along with large files for Active Archive This allows Data Den to be used as an active mounted filesystem via NFS or CIFS for direct data capture and manipulation. This can simplify data management for extreme scale projects with disk performance for recent data, and Data Den costs for capacity.

100 TByte minimum volume size

See: Advanced Uses of Locker & Data Den
See: Cost Estimator



Google Drive / DropBox etc.

Highly recommend only using for small data volumes (< 1 TByte). While possible these services are not designed for extreme scale.

For larger data volume transfers or Linux clients use RClone, to avoid issues with large transfers with browsers or automated transfers.

HDFS / Spark Hadoop

Specific to HDFS on ThunderX.

Importing/Exporting data in Parallel

Standard `hdfs -get` and `-put` has performance limitations. You can work around this using `parallel` to upload/download multiple files at a time. Speeds up to 1GB/s are possible.

Upload all files in the current directory that starts with `decahose-2018` to hdfs folder `decahose`

```
ls decahose-2018* | parallel --progress hdfs dfs -put {} decahose/
```

Download all files from hdfs folder `results` to local directory

```
hdfs dfs -ls -C results | parallel  --progress hdfs dfs -get {} .
```

Native Compressed Data

Most tools in the big data ecosystem can natively read compressed data. Thus you do not need to expand data before uploading and most tools can open/read without any extra action. The underlying HDFS libraries wll auto detect and decompress the data dynamically. Most compression types are supported, and special considerations should be made for uploading extremely large files due to not all compression formats being splittable.

Read compressed json into data frame:

```
reddit = sqlContext.read.json("reddit_full*.json.gz")
```

A1 * B1 16bytes : 1 Flop, 3hz, * 2 = 6 Gflops

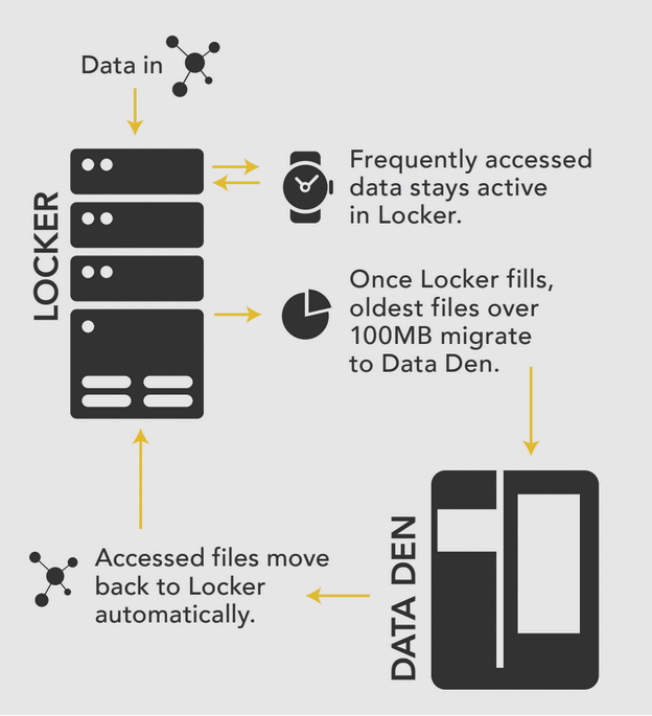
Advanced Tools

Data Den Active Archive

Active Archive sits between active storage (defined as storage with responsive latency) and cold storage such as default Data Den. Specifically, it allows:

1. Automating data lifecycle through cache sizing (Locker Size)
2. Separating the cost and behavior of small files from large files in a single export
3. NFS or SMB access to Data Den managed storage
4. Ability to manage what data is currently cached though archivescan

Technical Overview



Active Archive combines two ARC services Data Den deep archive (extremely low cost storage for large files > 100MB) and Locker (nearline storage system). A single filesystem tree is presented to the user but the actual data of each file could reside in ether system depending on the size of the file. This threshold defaults to 100MB and can be modified but unlikely to be lowered.

1. Files < 100MB never leave Locker disk storage, remaining responsive
2. Most recently accessed files > 100MB, reside on Locker until Locker quota. Additional data resides on Data Den until first access.

Once a file > 100MB is accessed that is only on Data Den the command (eg cp, globus, R) will block like hung storage mount until the entire file is copied from Data Den back to Locker. Further access to that file will be at Locker's performance until it is pushed out by newer data.

Archivescan is a tool for archive users

Archivescan (part of the Archivetar module) provides several additional functions to Active Archive Users:

Size Cache Requirements for migrated data

`cd <folder>`
`archivescan`
 	
`----- Results ------`
`Data Den Candidates:`
`Files: 24393`
`Size: 45.5TiB`
 	
`Cache (Locker) Candidates:`
`Files: 10324244`
`Size: 2.1TiB`
 	
`Scan Time 3185.097 Seconds`
`Fraction Offline: 95.6 %`
	This example would require at least
	
	● 11TB of Locker to hold the 10+ Million small files. (minimum recommended is 20TB)
	● They will only consume 2.1TB of the 11TB of space.
	● The remaining space will cache up to 9TB of the 45TB of data on Data Den.

List files on Locker Cache or Only on Data Den

```
cd <folder>
module load archivetar
archivescan --print-cached
archivescan --print-offline
```

Add `--quiet` to the above when being used in scripts to avoid extra messages.

Recall a Folder to Disk Cache

Note if there is not sufficient cache other cached files will be pushed off to make space. If the amount of data being recalled is more than the total cache size, the last file recalled will be pushed off again creating undesired results.

```
cd <folder>
archivescan --recall
```

Active Archive Policies

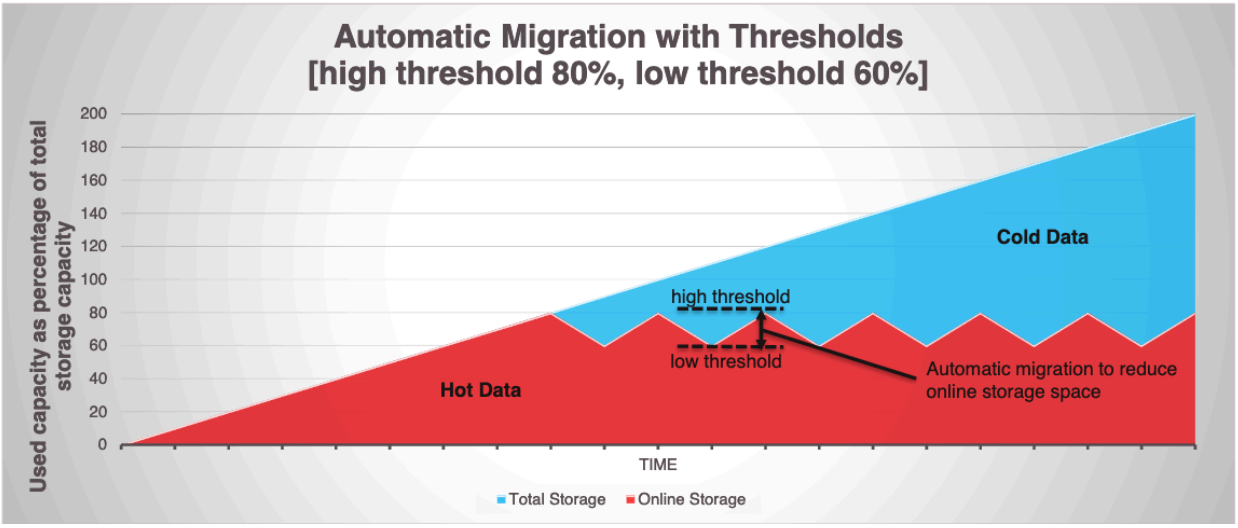
1. Minimum 20TB Cache Space
2. 1 Million Inodes (file & folder count quota) for each 1TB of Locker Cache

3. Volumes with excessive tape drive consumption will be required to adjust usage or modify cache settings
4. The 100TB of Data Den capacity from UMRCP can apply to the Data Den portion of an Active Archive Deployment
5. Sparse files should be avoided. These are uncommon.

All Active Archive Polices

Example Active Archive Use Cases

1. Primary data source which most recent data is significantly more active. Eg observational data, genomics etc.
    a. Set the Locker/Cache size to match active data volume
    b. Set Data Den size for total data
2. Reduce/eliminate the need for tar/zip for most users
    a. Set Locker/Cache size to match file counts and data volume < 100MB
    b. Set Data Den size to match data volume files > 100MB



Active Archive Cost Calculator

Does not include 100TB of free Data Den, Cost Calculator.

Notes for IT Staff

Active Archive exports show up as replicated Locker volumes. Commands like `df` and `du` will show twice the space consumed than the actual file sizes reported by `ls -l` or expected by the user. Eg a 20TB volume will appear as 40TB.

When the cache/Locker space is full in `df` output you will get an out-of-space message even if there is a remaining Data Den quota.

Files only on Data Den/tape (migrated state) `du` will report their size as 2MB while `ls -l` will show the appropriate size. To calculate the size of files on tape and on cache use `du --apparent-size` this also correctly accounts for replication and does not double report. Not using that option reports actual Locker disk blocks consumed including replication.

`archivescan` uses the sparse file nature of data on data den to infer what is offline when using `--current-state` or `--recall`

Data Den evaluates data not accessed for 24 hours every midnight. Files are then copied to tape (but not removed from Locker Cache). Locker cache can **only** be freed for new data by data that we premigrated through this process. Thus it may take as long as 48 hours to get data moved to Data Den. Tickets are required to migrate data early, e.g. for large one-time data imports.

Using find to find files likely not on tape but could be.

```
find . -type f -size +100M -size -10000G -amin +60 -printf "%S\t%p\n"
|  gawk '$1 > 1.0 {for (i=2; i<NF; i++) printf $i " "; print $NF}'
```

Find all files that are only on tape

```
IFS=$'\n' find . -type f  -printf "%S\t%p\n" | gawk '$1 < 1.0 {for
(i=2; i<NF; i++) printf $i " "; print $NF}'
```

Grand Unified File Index

Grand Unified File Index (GUFI) (gooo-phi) provides a high-performance filesystem scanner with the intention of providing user-friendly search. Built on top of SQL, it can be used as a high-performance metadata query. Because GUFI mirrors the filesystem folder structure it is possible to scan once and run multiple reports on different paths.

ARC Provided GUFI Container and Reports

A pre-built container with several reports is provided by ARC and usable on any system with Singularity.

Scanning a Filesystem / Build Index

The container sets the GUFI index to `/tmp/GUFI` and is only required for the `gufi_*` commands and not by the SQL-based reports.

```
module load singularity
singularity pull --arch amd64 library://brockp/gufi/gufi:master
singularity exec gufi_master.sif gufi_dir2index -n <#threads> <inputdir> /tmp/GUFI
```

ARC Reports

See https://github.com/umich-arc/gufi-archive/blob/master/SCRIPTS.md