Pawsey

Using Data Throughout the Project Lifecycle

Setonix Phase 1 Release

21 June 2022. Version 1.01

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU

The University of Western Australia

Focus for This Training

Learning outcomes:

- The importance of managing the entirety of the data lifecycle
- How to incorporate Acacia (Project Object Store) into your workflow
- Which data should be migrated into Acacia
- Key operations with S3 clients and data in Acacia
- Data management and storage strategies for workflows on Setonix
- Data storage considerations and best practices on Setonix

Core Migration Training Modules:

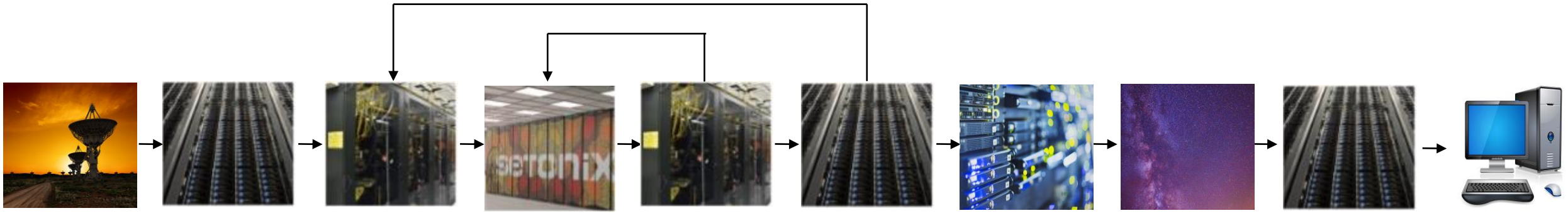
1. Getting Started with Setonix
2. Supercomputing Filesystems
3. Using Modules and Containers
4. Installing and Maintaining Your Software
5. Submitting and Monitoring Your Job
6. Using Data Throughout the Project Lifecycle

pawsey

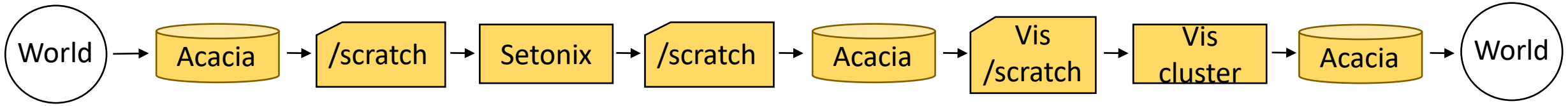
Section 1

Managing the Data Lifecycle

"Typical" Data Management

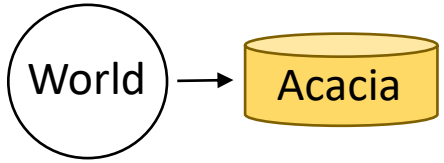


Create → Store → Stage → Process → Results output → Store → Stage → Visualise → Store → Archive



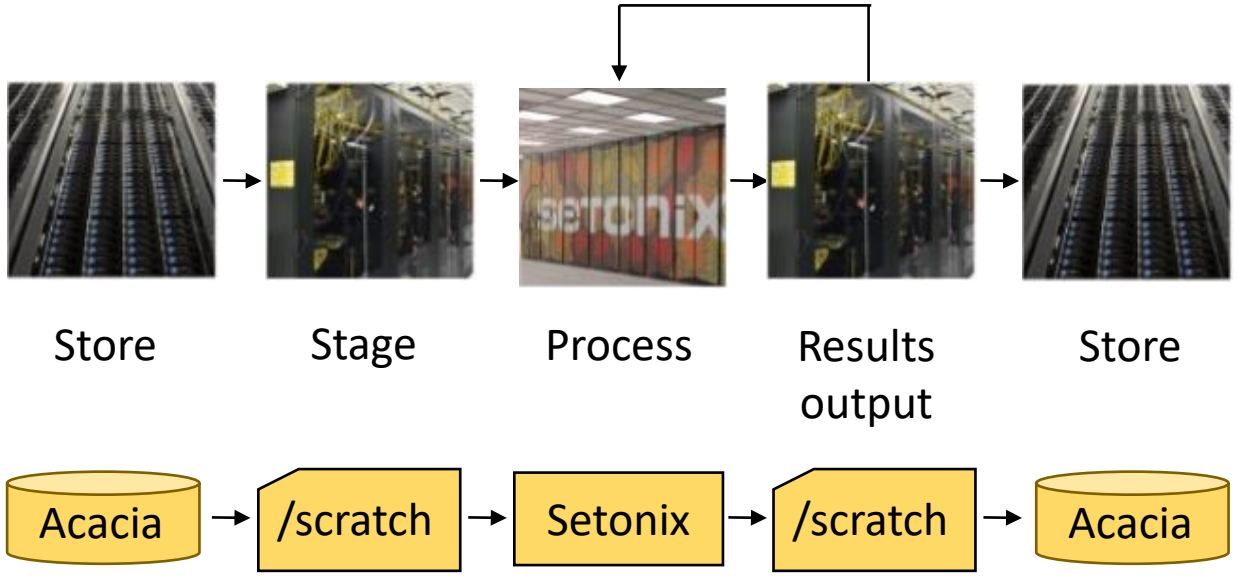
"Typical" Data Management: Transfer Data Into Acacia

Create Store



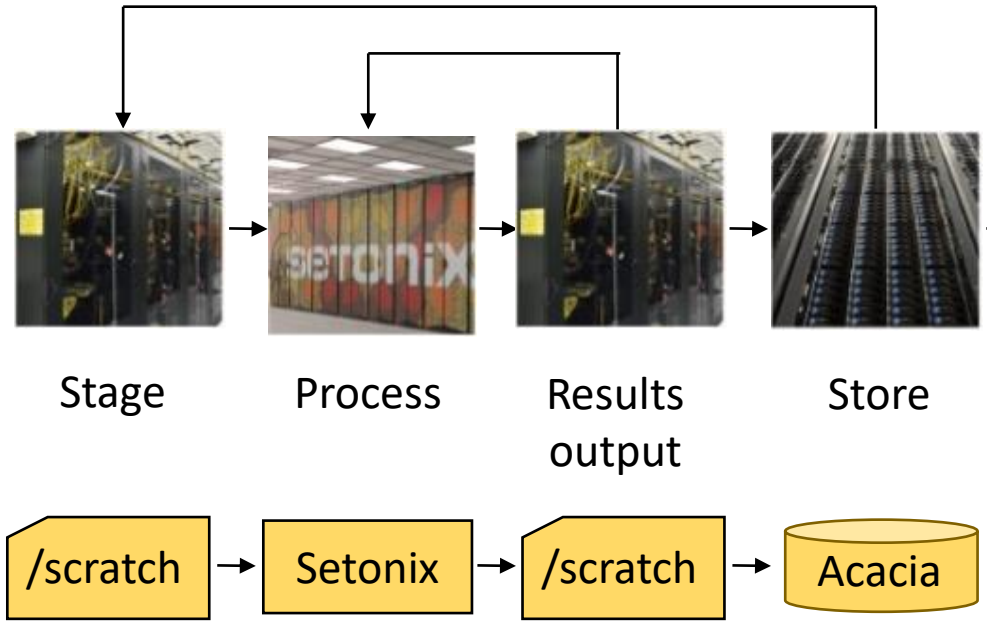
- Data is created/collected outside Pawsey from observations or from some previous research.
- Data to be used at Pawsey systems is stored into Acacia from users' local/institutional computers.

"Typical" Data Management: Using Data in Supercomputing Jobs (1)



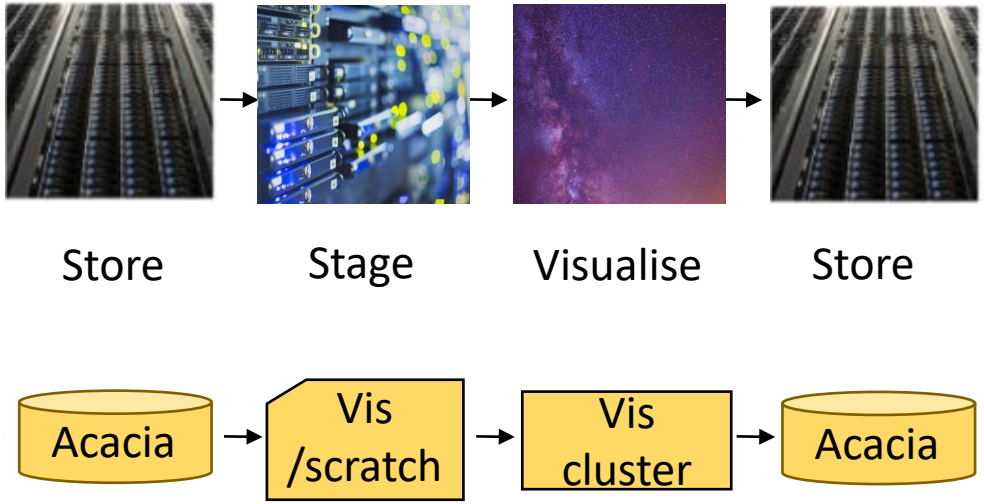
- Data needed for supercomputing calculations is transferred from Acacia and "staged" into Setonix filesystem (`/scratch`).
- Data is used in the supercomputing process.
- New results are written into the Setonix filesystem (`/scratch`), but cannot be kept there permanently.
- Results can be checkpoints, which may then be used to continue calculations in a cycle of supercomputing jobs.
- At any point, the generated results/data can be stored in Acacia.

"Typical" Data Management: Using Data in Supercomputing Jobs (2)



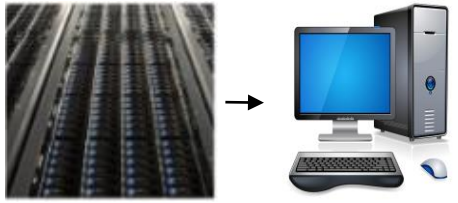
- Checkpoints stored in Acacia can be staged back into `/scratch` to continue calculations.
- Data in Acacia can be stored for the duration of your project to be used for further analysis.
- Note that `/group` does not exist anymore.

"Typical" Data Management: Visualisation

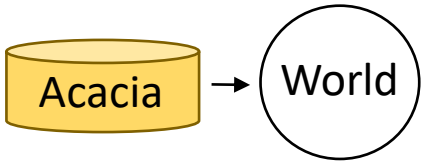


- Staging is performed into the visualisation cluster's own filesystem (Vis /scratch).
- Nebula (Windows-based) and Topaz (Linux-based) are the visualisation clusters available during Setonix Phase–1.
- Visualisation products (movies, images, etc.) can also be stored in Acacia if needed.

"Typical" Data Management: Archive & Backup



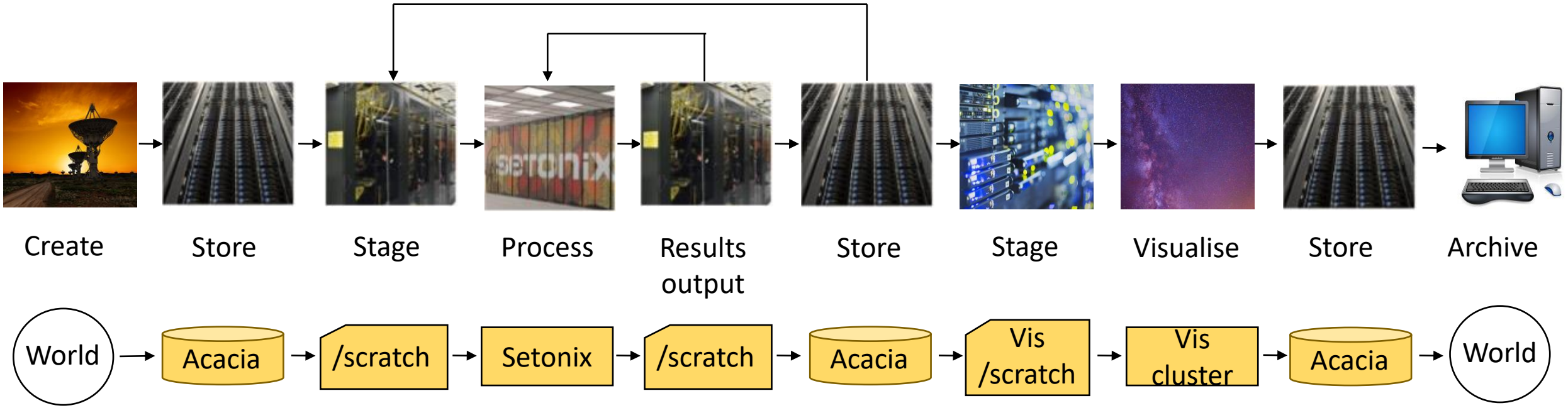
Store Archive



- Storage in Acacia is not permanent, that is, not beyond the lifetime of the project.
- Once analyses are finished, data should be archived at your own institution.

⚠ **IMPORTANT:** Pawsey does not provide backups for data in Acacia or filesystems. You are responsible for backing up your data at your own institution.

"Typical" Data Management: Your Workflow



What about your own workflow?

- How many intermediate **storage** operations do you need?
- Do you need to **stage** large amounts of data before supercomputing processing?
- Do you need to **store** large amounts of data for further postprocessing/analysis?

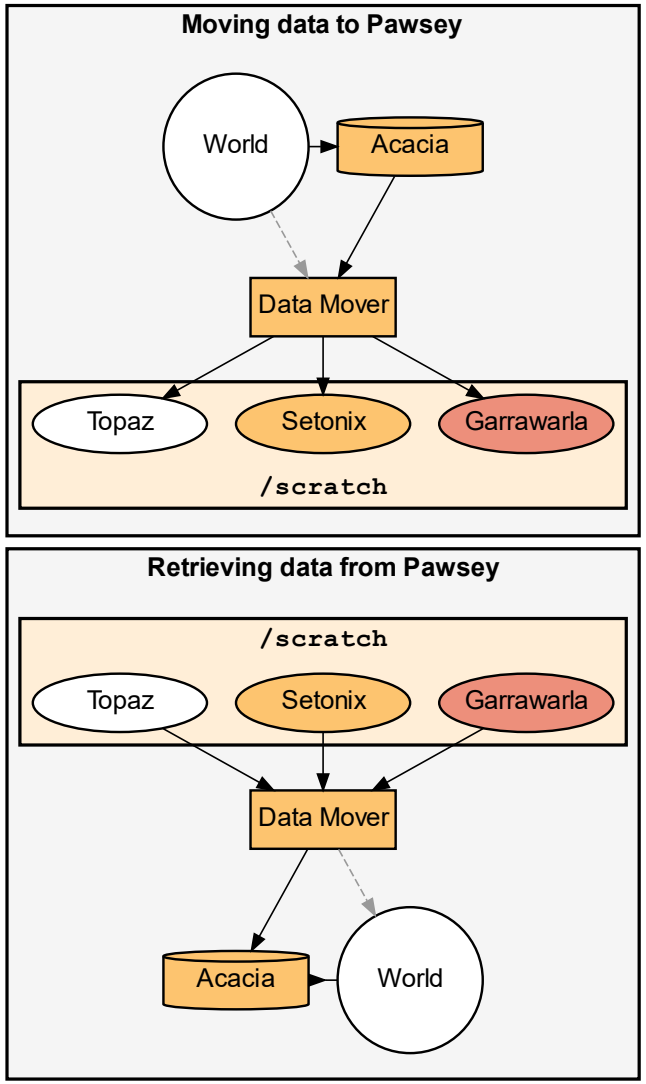
Section 2

Transferring Data to and from Pawsey

Transferring Data to and from Acacia

- Transfers into /out of Acacia are performed via tools with the S3 protocol.
- The following S3 command line clients are commonly utilised: **mc** (**minio client)**, **AWS client**, **rclone**.
  - These clients are already accessible on Pawsey clusters via modules.
  - Install your chosen client/s on your computer for communication from outside Pawsey.
- Many GUI clients also offer S3 capabilities and can be used on your computer.

- More details about S3 clients @ [Acacia User Guide]

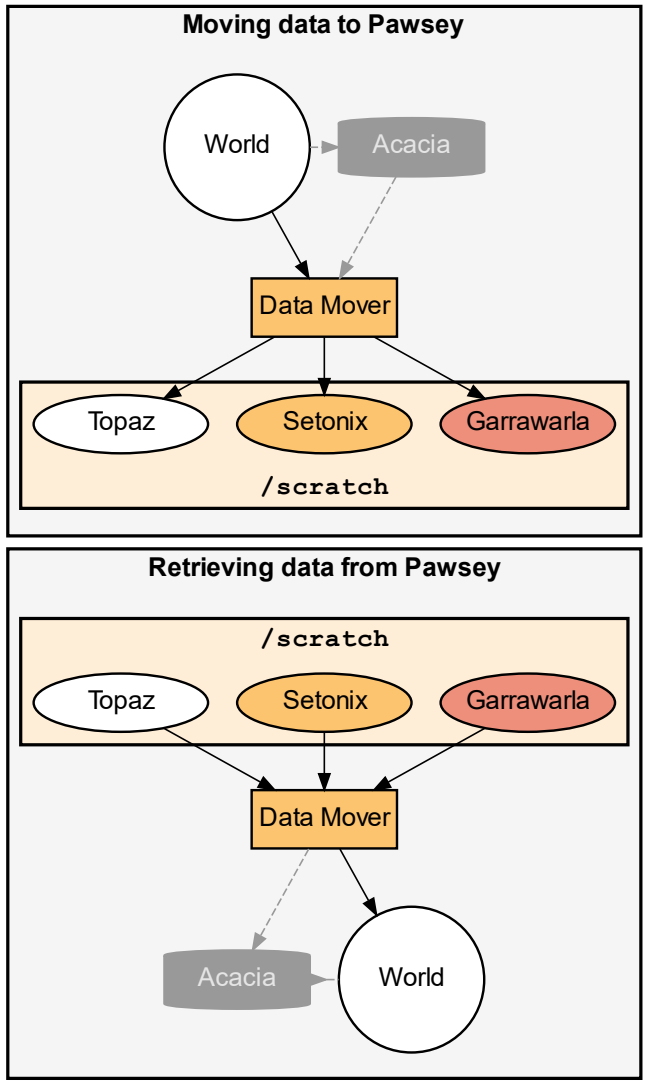


Direct Transfers to and from `/scratch`

- The Setonix filesystems are still available externally via the login and data mover nodes.
- Use **`data-mover.pawsey.org.au`** for large transfers (~>100Mb) to avoid congesting the login nodes.
- Available command line clients will depend on your own terminal environment and should use the ssh protocol: **scp**, **sftp**, **rsync**.
- Commonly used GUI clients include MobaXTerm, FileZilla, WinSCP, and Cyberduck.
- Pawsey filesystems are not designed for storing data.

More details @

- Migration Training Module 2: Supercomputing Filesystems (Materials, Recordings)
- Transferring files



pawsey

Section 3

Acacia: The New Project Object Store

Acacia: The Project Object Store

- Acacia is a high-performance, large-volume CEPH Object Storage system with S3 interface (similar to Amazon S3).
- Acacia has a total capacity of 60 Petabytes.
- Supercomputing projects will have an initial 1 TB allocation (data is shared among all members).
- Users will have additional 100 GB individual allocation (not shared by default).
- The new larger storage capability replaces `/group`.

Acacia is an Object Store NOT a Filesystem

- Acacia is a CEPH Object Storage system with its own authorization access.
- Interaction is through a S3 compatible client, e.g., mc (minio client).
- Objects are immutable and cannot be opened or edited within Acacia.
- For working with the data, it should be first "staged" into `/scratch`.
    - New technologies are trying to overcome this barrier (not ready yet).

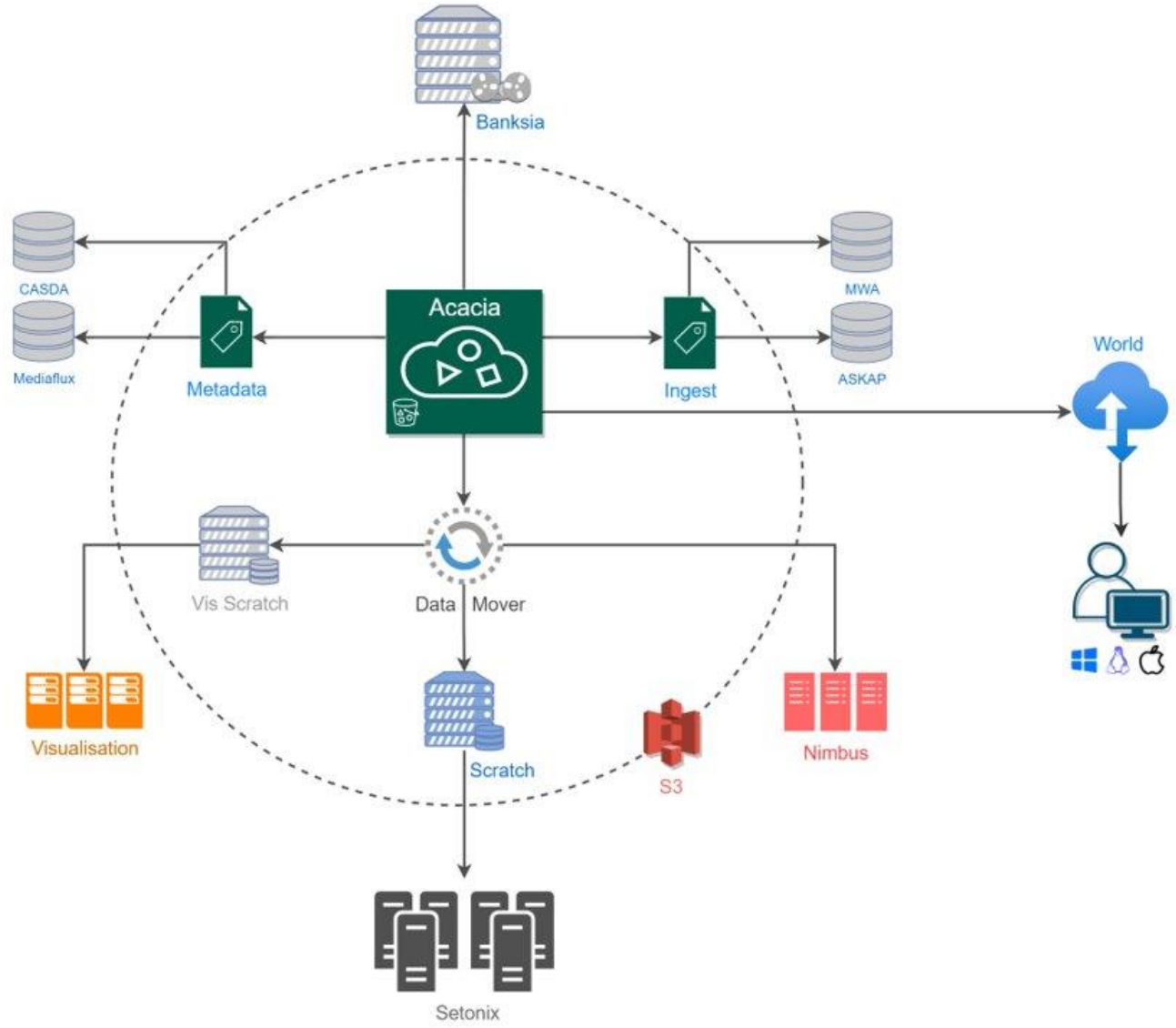
Key Concepts in the Use of Object Storage

- **Object:** atomic unit of data formed by "file + metadata"
  - The **metadata** is additional information describing the object itself.
- **Bucket:** structure that stores objects
  - Buckets and directories have similar purposes, but they are not the same.
  - Buckets cannot be created inside buckets, that is, sub-buckets cannot exist.
  - S3 clients organise objects within a bucket with the use of prefixes.
  - There is no limit on the numbers of objects that a bucket can contain, but we do not recommend to store more than 100,000 objects in a single bucket.

More details @ [Acacia User Guide]

Acacia Connectivity

- The visibility of Acacia from many platforms creates a more practical working environment.
- Acacia allows easier integration of hybrid workflows (Supercomputing + Cloud + Visualisation)
- Data can be stored and retrieved from your host institution/home and the many Pawsey systems.



Benefits of Acacia

Old:	New:	Benefits
Limitations of /group quota	Larger project-storage quota on Acacia	• This allows projects to keep a larger amount of important data at Pawsey. • This may improve dramatically the time spent on "staging" of data into /scratch (avoiding slow transfers to/from your own institution).
Supercomputing filesystems not directly accessible from other systems	Every Pawsey system now has uniform access to Acacia (even from your own computer)	• This allows easier integration of hybrid workflows (Supercomputing + Cloud + Visualisation) through their common access to Acacia.
	Access to Acacia from computers outside Pawsey use the same protocol	• Access to Acacia from your own institution use the same protocol (via S3 clients), making your access routines more portable.
Managed Data Storage	Your data on Acacia can be shared with colleagues outside Pawsey	• Acacia enables easier interaction with colleagues around the world.

Differences Between Filesystems and Acacia Object Store (1)

Existing: Filesystems	New: Acacia Object Store	Differences
	Needs its own access authorization	• Requires the use of credentials for authorising access.
Responds to operating system commands	Responds to S3 commands	• Interactions through a S3 compatible client application (mc [minio], aws-cli, rclone, etc.) • Main operations: create-bucket, put-object, get-object
Modifiable data files	Immutable data objects	• Acacia has immutable stored objects. • Acacia does not allow open/read/write/execute operations on stored objects. • To use your data, copy it from Acacia and stage it into `/scratch` (or your own local filesystem). • When work is finished, you will need to put files back into Acacia if you want to preserve them for "future use".

Differences Between Filesystems and Acacia Object Store (2)

Existing: Filesystems	New: Acacia Object Store	Differences
Contains files	Contains objects	• Object = [file + metadata] • The metadata may contain a description and any desired information about the file.
Uses directories for collecting files	Uses buckets for collecting objects	• Buckets store a collection of objects. • Buckets cannot have sub-buckets inside, so organisation structure becomes flat. This contrast with the tree-like structure of directories in filesystems.
Shared through group permissions	Shared through policies	• Data in project-storage is shared with all members of the project by default. • Data in user-storage is not shared by default. • Additional sharing can be set through policies.

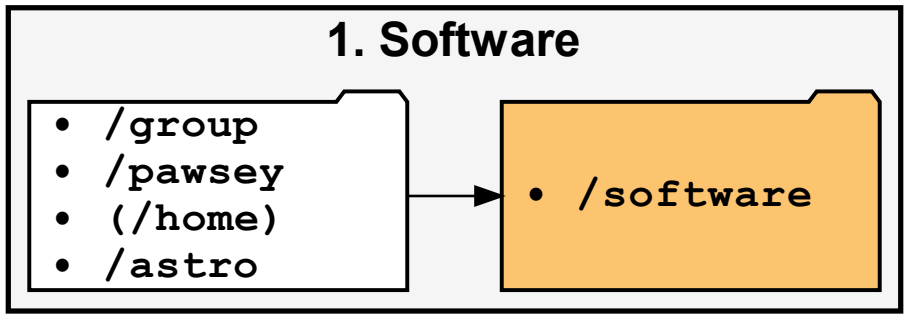
- More details @ Acacia User Guide

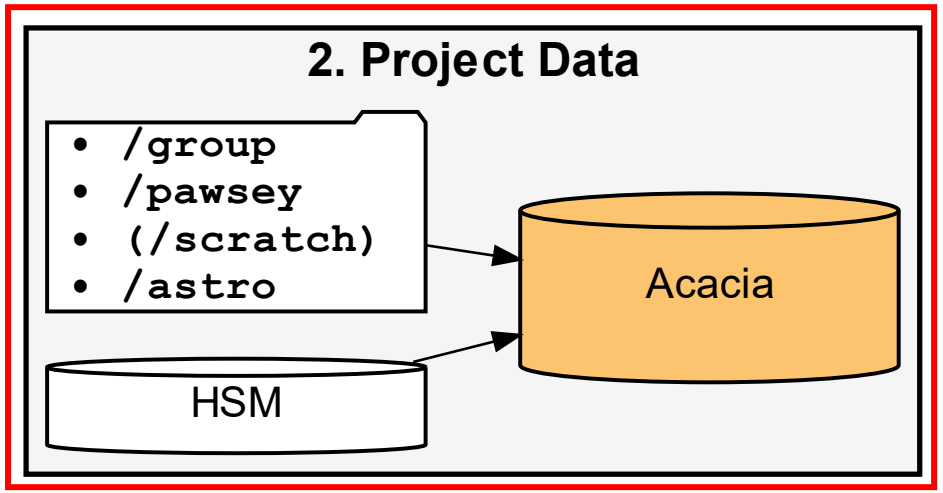
pawsey

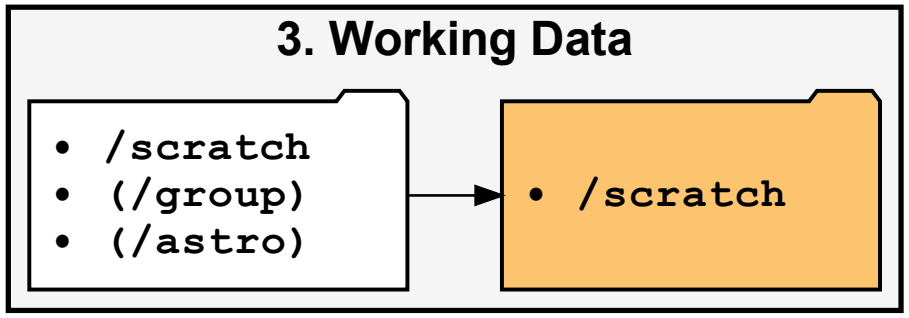
Section 4

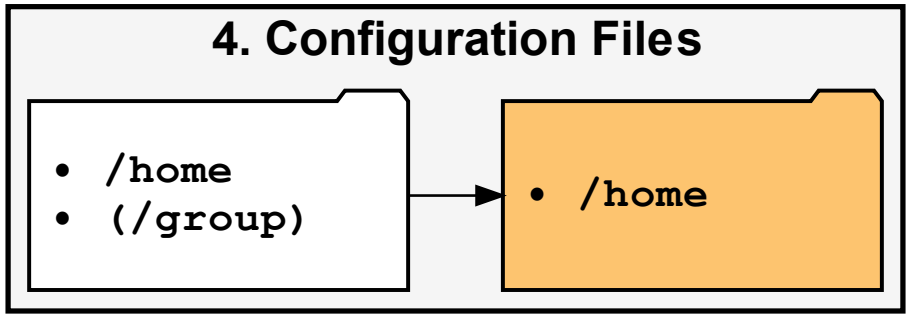
Migrating Your Data from /group

Reorganising Data into New Infrastructure







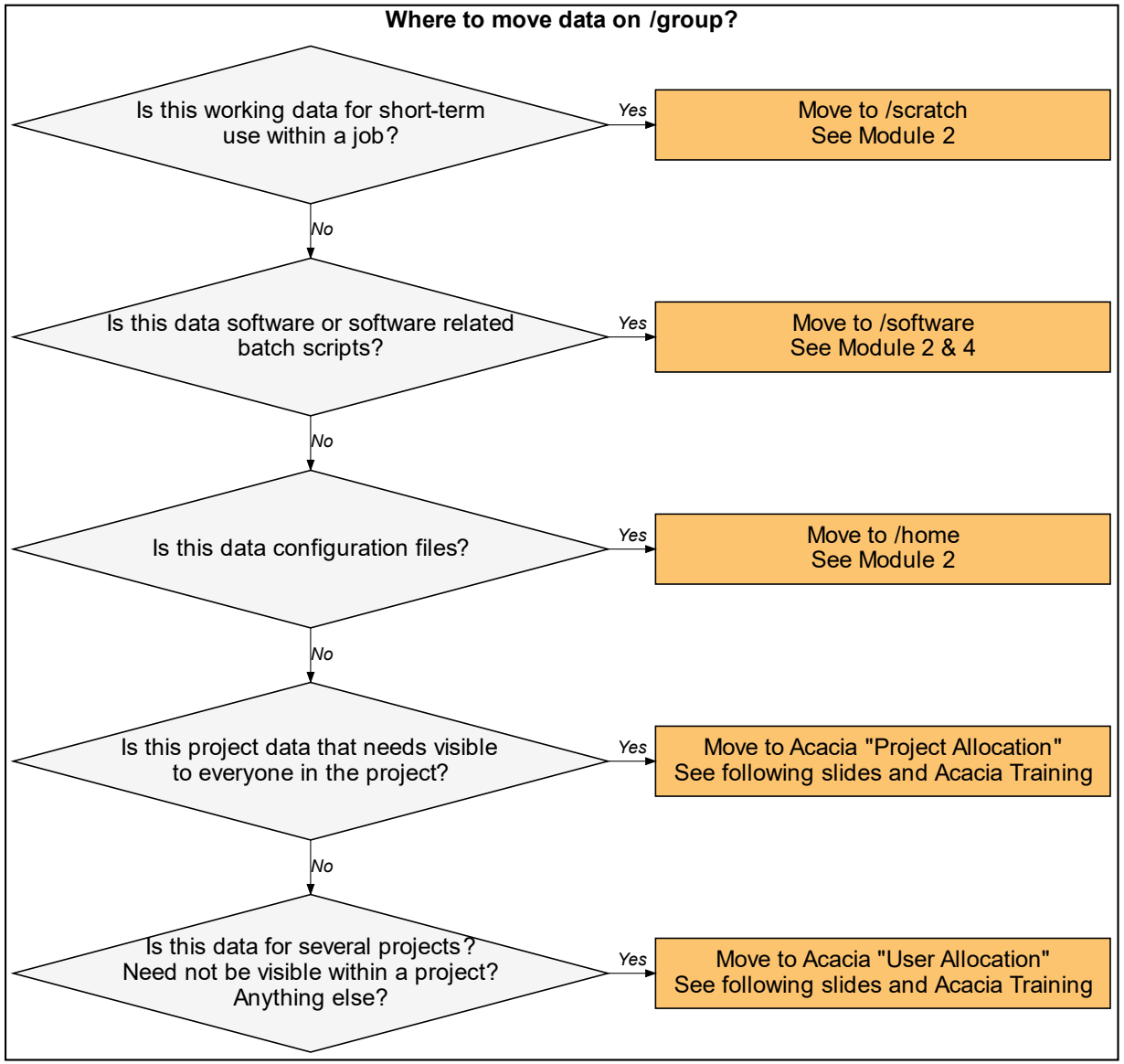


Legend
- Gold = New infrastructure
- White = Existing infrastructure
- (Brackets) = Not recommended usage

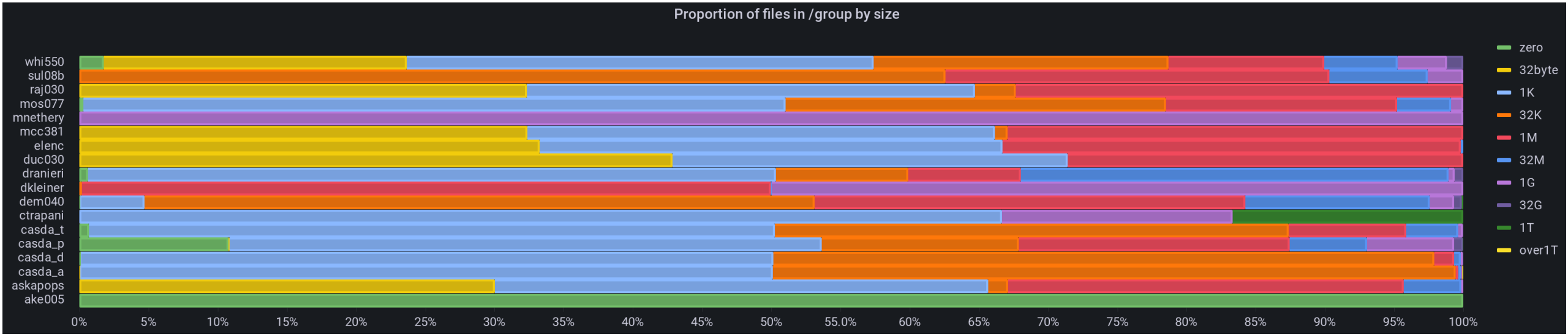
More details @ Migration Training Module 2: Supercomputing Filesystems (Materials, Recordings)

Migrating Data from /group

- /group will soon be read-only, and eventually decommissioned.
- We strongly recommend that you migrate your data into the new filesystems and/or Acacia storage system as soon as possible.
- Before updating your workflows, migrate data from /group.

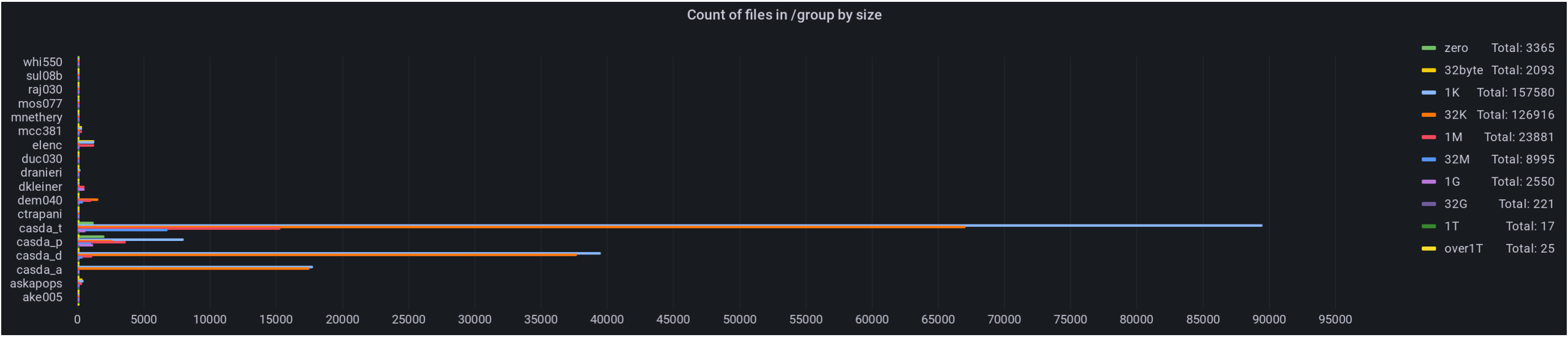


Check Your File Size Distribution



- Principal Investigators have received this analysis as part of a report.
- This shows the proportion of the size of the files that each user in the project has in `/group`.

Check How Many Files You Have



- Another part of this report is the number of files that each user in the project has in `/group`.
- Directories with small number of files are good candidates to be “S3-mirrored” into a bucket in Acacia with exactly the same objects.
- Directories with thousands of files are good candidates for being packed into .tar files before storing into Acacia.

Migration Recommendations

1. Check your data in `/group` and define what needs to be kept at Pawsey and what can be deleted or archived at your own institution.
2. Define the part of your data to be packed and stored as .tar files, such as, directories containing thousands of files, complete simulation cases, etc.
    - You can create more than one .tar file per original directory (and they can be packed in parallel).
3. Define the part of your data to be stored as mirrored , such as, data directories from which workflows will select objects to be staged in each job.
4. Start small and test your workflows before embarking on a large migration process.
5. Ask Pawsey staff for help if needed.

pawsey

Section 5

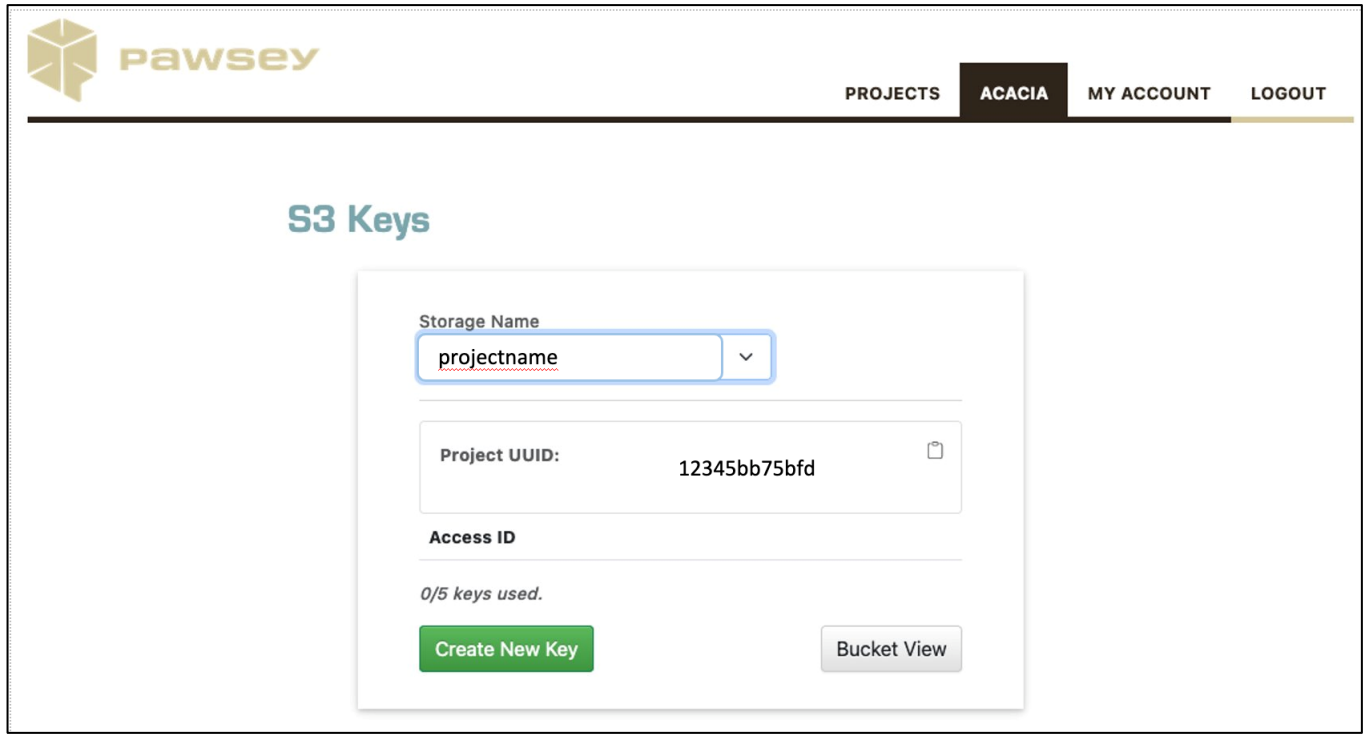
Examples of S3 Client Operations

Obtaining Access Keys to Acacia (1)

1. Log into Pawsey Users Portal (Origin):

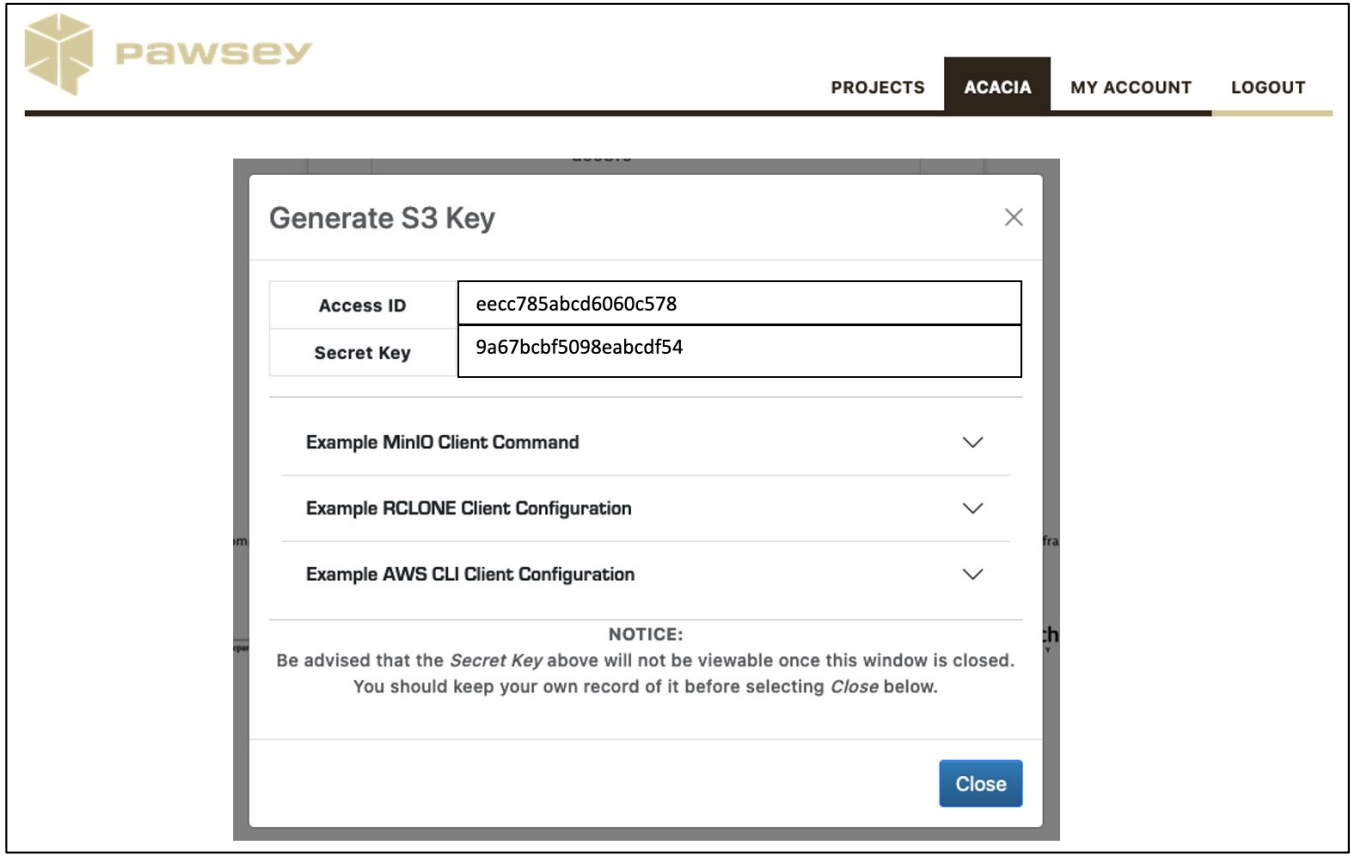
   https://portal.pawsey.org.au

2. From the portal, you can:
   - Manage the access keys to your different storages
   - Have a view of the existing buckets within the storages
3. Click **Create New Key** to get a new pair of access keys.



Obtaining Access Keys to Acacia (2)

4. Copy the "Access ID" and the "Secret Key" into a safe place.
    - Never share your access keys.
    - Read the "Notice": keys are not stored in the portal.
5. It is a good idea to take note of the provided "Configuration Command" examples for the different S3 clients.



More details @ Acacia User Guide: [Managing an Allocation]

Endpoint

- Access to both, user storage and project storage, is through the same endpoint:

```
https://projects.pawsey.org.au
```

- Then, what determines the specific buckets/data to be accessed are the access keys.
- Both, endpoint and access keys are details to be included in the client's configuration.

Which S3 client should I use?

- Minio client (mc) is recommended for basic interaction with Acacia.
- Rclone is recommended for synchronising full directories and is very good for scripting (low verbosity).
- AWS client is recommended for advanced functionality, such as setting policies for sharing buckets.
- Our documentation and this training use mc as the starting point client.
- Modules for the three S3 clients are available in our clusters:

```
$ module load miniocli/version
$ module load rclone/version
$ module load awscli/version
```

- You can use the same S3 clients from your own laptop or computer at your institution for direct access to Acacia. **But** you will need to perform your own installations of the clients.

Client Configuration

- The mentioned clients save access configuration in their respective configuration files
  - mc (minio client) in:

    `$HOME/.mc/config.json`
  - `rclone in:`

    `$HOME/.config/rclone/rclone.conf`
  - `aws client in:`

    `$HOME/.aws/config`

    `$HOME/.aws/credentials`
- Clients have their own commands for setting configurations, but you can edit the configuration files directly with the same effect.
- Keys are stored in the configuration files.

⚠ **IMPORTANT:** Ensure that the configuration files are only accessible and readable by you.

Client Configuration: mc (minio client)

- mc configuration command creates an alias for the endpoint address and its access keys:

  `mc alias set` *`aliasName`* `https://projects.pawsey.org.au` *`accessID secretKey`*

- Use an alias name that causes no confusion with local directories. In this training, for example, we suggest to put the word "`acacia-`" before **your** "*`projectname`*" for the alias name of your project storage (but the whole alias name is arbitrary and up to each user):

```
$ mkdir -p $HOME/.mc
$ set +o history   #This command turns off history recording for safety
$ mc alias set acacia-projectname https://projects.pawsey.org.au 12345abcd 67890efgh
Added `acacia-projectname` successfully.
$ set -o history   #This command turns on history again
$ ls -l $HOME/.mc/config.json
-rw------- 1 user user 2367 Mar 28 15:22 /home/username/.mc/config.json
```

- Note the use of `set +o history` … `set -o history` to avoid recording of the keys in the history.
- More details @ Acacia User Guide: Installing and Configuring an S3 client

Client Configuration: rclone

- Here we edit directly the configuration file. Note the use of the `tee` command and <<EOF redirection):

```
$ mkdir -p $HOME/.config/rclone
$ set +o history   #This command turns off history recording for safety
$ tee -a $HOME/.config/rclone/rclone.conf <<EOF

[acacia-projectname]
type = s3
provider = Ceph
endpoint = https://projects.pawsey.org.au
access_key_id = 12345abced
secret_access_key = 67890efgh
EOF
$ set –o history    #This command turns on history again
```

```
$ rclone listremotes
acacia-projectname:
```

- We recommend to use the same “profile name” as the "alias name" for `mc` (previous slide). But the client settings are independent from each other.

Functionality Check

1. Check that the alias/profile settings exist.
   - Use "`mc alias list`" to check the alias settings:

     `mc alias list` *`aliasName`*

2. Check that key operations work.
   - For example, check if you can create, list and delete a test bucket in a storage where you have writing access:

     `mc mb` *`aliasName/testBucketName`*

     `mc ls` *`aliasName/`*

     `mc rb` *`aliasName/testBucketName`*

```
$ mc alias list acacia-projectname
acacia-projectname
  URL       : https://projects.pawsey.org.au
  AccessKey : 12345abcd
  SecretKey : 67890efgh
  API       : s3v4
  Path      : auto
```

```
$ mc mb acacia-projectname/projectname-abc-test01
Bucket created successfully `acacia-projectname/projectname-
  abc-test01`.

$ mc ls acacia-projectname/
[2022-02-18 13:41:40 AWST]     0B projectname-abc-test01/

$ mc rb acacia-projectname/projectname-test01
Removed `acacia-projectname/projectname-abc-test01`
  successfully.
```

Bucket Names have Restrictions

- Bucket names on Acacia must be **UNIQUE** across **ALL** projects and users.
  - You will receive an error if a bucket name already exists for another user or project.
- **Hint:** You can use your project name at start of your bucket naming to create a unique identifier:

  projectname-sometext

- Do not include any confidential information in the bucket name:

> ⚠ **IMPORTANT:** Never use your username to create a unique identifier.

- Bucket names **cannot have CAPITAL letters, underscores, spaces, nor special characters.**
- More details @ Acacia User Guide: [Acacia - Introduction]

```
$ mc ls acacia-mine/

$ mc mb acacia-mine/thesis
mc: <ERROR> Unable to make bucket `acacia-mine/thesis`. Error
  response code BucketAlreadyExists.

$ mc mb acacia-mine/projectname-abc-thesis
Bucket created successfully `acacia-mine/projectname-abc-
  thesis`

$ mc ls acacia-mine/
[2022-02-23 11:36:25 AWST] 4.0KiB projectname-abc-thesis/
```

```
$ mc mb acacia-mine/“projectname_abc rawData”
mc: <ERROR> Unable to make bucket `acacia-mine/
projectname_abc rawData`.
                ⬆   ⬆   ⬆    Bucket name contains invalid characters
```

⚠ Warning: Clients may operate inadvertently on your file system.

Example: `mc mb` works as `mkdir -p` when the alias does not correspond to a storage in Acacia:

- If the alias does not exist (possibly due to a typo) `mc` will create a directory in your local filesystem using the wrong alias as name:

  `mc mb` *wrongAlias/bucketName* => `mkdir -p` *wrongAlias/bucketName*

- No error message is sent to the user as this is a valid command acting on your filesystem!

To avoid this problem:

- Use <tab> autocomplete, which "types" the correct names for you.
- Use an environment variable to avoid typos (especially useful in Slurm batch scripts).

  `theStore=`*aliasName*

- Check for correct behaviour and that directories were not created locally.

```
$ find . -type d
.
$ mc mb akazia-pine/projectname-abc-test01
Bucket created successfully `akazia-pine/projectname-
  abc-test01`.
$ find . -type d
.
./akazia-pine
./akazia-pine/projectname-abc-test01 :
```

```
$ find . -type d
.
$ theAlias="acacia-mine"
$ mc mb $theAlias/projectname-abc-test02
Bucket created successfully `acacia-mine/projectname-
  abc-test02`.

$ mc ls $theAlias
[2022-02-23 11:36:25 AWST] 4.0KiB projectname-abc-
  test02/
$ find . -type d
.
```

Storing Data into Acacia

Case 1: Copy a small number of files into a bucket

- When just a small amount of data is to be transferred
- Useful for testing

Case 2: Mirror or synchronisation of a full directory into a bucket

- Useful when your workflow needs to select individual objects from the storage
- This method stores each file from a directory as an object with the path as prefix to the name

⚠ **NOTE:** We do not recommend to store more than 100,000 objects per bucket.

Case 3: Collect multiple files into a single .tar file before storing in Acacia

- Preferred method for:
  - dealing with large amount of files
  - files related to the same application/case that need to stay together
- Packing of data into a .tar file is performed in the local filesystem before transfer into Acacia.
- Data stays in the .tar file when in Acacia (no extraction can happen in Acacia).
- Extraction of content of the stored .tar file requires the file to be "staged" back into the local filesystem.

Storing Data into Acacia: Copy of a Small Number of Files

Copy a file/s into a bucket

- Using `mc cp` to copy single file:

`mc cp --quiet --md5 localFile aliasName/bucketName/pathPrefix/objectName`

- Using `mc cp` to copy several files:

`mc cp --quiet --md5 localFiles aliasName/bucketName/pathPrefix/`

  - Note the use of "/" at the end for the prefix to be understood as prefix and not as an object name

⚠ **Note:** Use the command `help: mc cp --help`

```
$ ls ./
kangaroo  koala  platypus   quokka  wombat
```

```
$ mc cp --quiet --md5 ./quokka acacia-projectname/projectname-abc-downunder/fauna/quokka
`/yourlocalpath/emu` -> `acacia-projectname/projectname-abc-aus/fauna/quokka`
Total: 0 B, Transferred: 48.25 GiB, Speed: 124.01 MiB/s
```

```
$ mc cp --quiet --md5 ./k* acacia-projectname/projectname-abc-downunder/fauna/
`/yourlocalpath/kangaroo` -> ` acacia-projectname/projectname-abc-downunder/fauna/kangaroo`
`/yourlocalpath/koala` -> ` acacia-projectname/projectname-abc-downunder/fauna/koala`
Total: 0 B, Transferred: 48.25 GiB, Speed: 124.01 MiB/s
```

```
$ mc ls acacia-projectname/projectname-abc-downunder/fauna/
[2022-04-27 09:33:45 AWST]     0B STANDARD kangaroo
[2022-04-27 09:33:45 AWST]     0B STANDARD koala
[2022-04-27 09:33:16 AWST]     0B STANDARD quokka
```

Storing Data into Acacia: Synchronise a Full Directory (rclone sync)

Synchronise full directories

⚠ **Note:** We do not recommend synchronising directories with a large number of files (>~100,000 files).

- We recommend to use `rclone` for syncronisation in preference to `mc mirror`
- With `rclone` you can use the command `sync`:

  `rclone sync --transfers` *`N`* `--checksum` *`localDir profileName:bucketName/prefixPath/`*

  - Note the use of the colon (:) after the profile name
  - *`prefixPath`* contains the intended naming (see example)

Important Notes:

- `rclone sync` seems to recover better from problems during execution
- The `--transfers` option indicates the number of files to be processed at the same time (we recommend to use 12)
- `--checksum` verifies the copy by comparing the mdf5 hashes of the objects

```
$ rclone sync --transfers 12 --checksum ./flora acacia-
  mine:projectname-abc-downunder/flora/


$ rclone lsl acacia-mine:projectname-abc-downunder/flora/
        0 2022-04-27 11:42:25.000000000 NSW/bluebell
        0 2022-04-27 11:42:25.000000000 NSW/bottleBrush
        0 2022-04-27 11:42:28.000000000 NT/waratah
        0 2022-04-27 11:42:28.000000000 NT/wattle
        0 2022-04-27 11:42:39.000000000 TAS/spiderFlower
        0 2022-04-27 11:42:44.000000000 VIC/gumTree
        0 2022-04-27 11:42:44.000000000 WA/everlasting
        0 2022-04-27 11:42:44.000000000 WA/kagarooPaw

$ rclone lsd acacia-mine:projectname-abc-downunder/flora/
          0 2022-06-13 14:03:24        -1 NSW
          0 2022-06-13 14:03:24        -1 NT
          0 2022-06-13 14:03:24        -1 TAS
          0 2022-06-13 14:03:24        -1 VIC
          0 2022-06-13 14:03:24        -1 WA
```

Storing Data into Acacia: Pack Data into .tar Files Before Transfer

First pack with tar command:

`tar –czvf tarFile listToTar`

- Plan a strategy to pack your data (for example by case, or by results at different times)
  - Several .tar files can be created in parallel and faster than a single big .tar file containing all data

Then transfer the files(s):

`mc cp --quiet --md5 tarFiles aliasName/bucketName/prefixPath/`

Important Notes:

- For critical data, reliability checks of the .tar file should be performed before transfer, for example:
  - Use of "`tar –tf`" to list and compare contents
  - Re-extract and compare with original

```
$ ls -d */
case0/  case1/  case2/
$ tar -czvf case0.tar.gz case0
case0/
…
$ tar -czvf case1.tar.gz case1
case1/
…
$ tar -czvf case2.tar.gz case2
case2/
…
```

```
$ ls *.tar.gz
case0.tar.gz   case1.tar.gz   case2.tar.gz
$ mc cp --quiet --md5 *.tar.gz acacia-mine/projectname-abc-tarcases/2022/
`/yourlocalpath/case0.tar.gz` -> `acacia-mine/projectname-abc-tarcases/2022/case0.tar.gz`
`/yourlocalpath/case1.tar.gz` -> `acacia-mine/projectname-abc-tarcases/2022/case1.tar.gz`
`/yourlocalpath/case2.tar.gz` -> `acacia-mine/projectname-abc-tarcases/2022/case2.tar.gz`
Total: 0 B, Transferred: 12.00 GiB, Speed: 140.09 MiB/s
```

Have files been transferred correctly?

Clients use mdf5 checksums to guarantee that final objects and original files are the same:

- `mc cp --quiet --md5 localFile aliasName/bucketName/pathPrefix/objectName`
- `rclone sync --checksum localDir profileName:bucketName/prefixPath/`
- These options verify the copies by comparing the mdf5 checksums of the objects

If you still want to double check equality, `rclone check` is a great tool

`rclone check localFile profileName:bucketName/prefixPath/objectName --combined report.out`

`rclone check localDir profileName:bucketName/prefixPath/ --combined report.out`

If files in Acacia are correct, then you can DELETE original data from your host filesystem.

```
$ rclone check ./fauna acacia-projectname/projectname-abc-downunder/fauna/ --combined report.out
2022/04/29 13:19:54 NOTICE: S3 bucket projectname-abc-aus path fauna: 0 differences found
2022/04/29 13:19:54 NOTICE: S3 bucket projectname-abc-aus path fauna: 5 matching files

$ cat report.out
= quokka
= wombat
= platypus
= koala
= kangaroo
```

```
$ rm -rf ./fauna
```

Staging Data from Acacia into `/scratch`

Apply the same transfer commands, but in the opposite direction.

For example:

- Using `mc cp` to stage a single file:

`mc cp --quiet --md5` *`aliasName/bucketName/pathPrefix/objectName localPath/localFile`*

- Using `rclone sync` to stage a full directory:

`rclone sync --transfers` *`N`* `--checksum` *`profileName:bucketName/pathPrefix/localPath/`*

```
$ mc ls acacia-projectname/projectname-abc-tarcases/2022/
[2022-04-29 13:51:32 AWST]   309MiB STANDARD case0.tar.gz
[2022-04-29 13:51:32 AWST]   309MiB STANDARD case1.tar.gz
[2022-04-29 13:51:32 AWST]   310MiB STANDARD case2.tar.gz
$ mc cp --quiet --md5 acacia-projectname/projectname-abc-
tarcases/2022/case0.tar.gz .
`acacia-projectname/projectname-abc-
tarcases/2022/case0.tar.gz` -> `case0.tar.gz`
Total: 0 B, Transferred: 309 MiB, Speed: 104 MiB/s
$ tar xvf case0.tar.gz
case0/
case0/file3
case0/file1
case0/file2
```

```
$ rclone sync --checksum --transfers 12  acacia-
mine:projectname-abc-downunder/fauna/ ./fauna

$ ls fauna
kangaroo  koala  platypus  quokka  wombat
```

pawsey

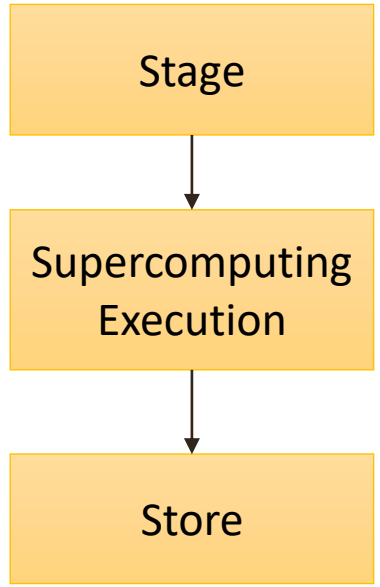
Section 6

Automating Data Management in a Workflow

Use Separate Slurm Batch Scripts for Different Steps

For example, you may need to perform a series of steps that depend on each other (shown on right) and make use of the following Slurm batch scripts:

- `stageFromAcaciaTar.sh`
  - Allocates resources in the `copy` partition
  - Copies a `.tar` file from Acacia into `/scratch`
  - Extracts the content of the `.tar` file and stages it in the working directory
- `superExecutionSetonix.sh`
  - Allocates resources in the `work` partition
  - Executes a supercomputing job using a high performance tool
    - The tool reads what is needed from the recently staged data
    - The tool writes results/checkpoints into the working directory
- `storeIntoAcaciaTar.sh`
  - Allocates resources in the `copy` partition
  - Creates a `.tar` file(s) containing what is needed to be kept from the recently created results
  - Copies the `.tar` file(s) into Acacia
- Full example scripts @ Acacia User Guide: [Acacia Workflow Examples]



Slurm Batch Script for Staging Data

stageFromAcaciaTar.sh:

- Use the `copy` partition.
- S3 clients work better with 2 or more `cpus`.
- Load the necessary modules.
- Use variables for easy reuse of the script.
- Perform checks for valid arguments (not shown here).
- `mc cp` performs the copy of the `.tar` file.
  - Always redirect the output of mc to `/dev/null` to avoid unnecessary recording of the output.
- `tar -x` extracts the content of the file, completing the staging process.

```
#!/bin/bash --login
#SBATCH --account=projectname
#SBATCH --job-name=stageTar
#SBATCH –partition=copy
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=2
#SBATCH --time=requiredTime
#SBATCH --export=NONE

module load miniocli/version

tarFileName=fileName.tar
fullPathInAcacia=aliasName/bucketName/prefixPath
workingDir=/scratch/yourCasePath

# (Checks for valid arguments go here)

srun mc cp --md5 --quiet "${fullPathInAcacia}/${tarFileName}" \
 "${workingDir}/" > /dev/null

# (Checks for correct transfer go here)

srun tar -xvf "${workingDir}/${tarFileName}" –C "${workingDir}"

# (Checks for correct staging go here)
```

Slurm Batch Script for the Supercomputing Execution

superExecutionSetonix.sh:

- Use the `work` partition.
- Request the resources that are needed.
- Load the necessary modules.
- Use variables for easy reuse of the script.
- Perform checks for valid arguments (not shown here).
- `cd` puts the process in the correct directory.
- `srun` executes the tool with the indicated resources.
  - Use `SLURM_variables` where possible for easy reuse of the script.

```
#!/bin/bash --login
#SBATCH --account=projectname
#SBATCH –partition=work
#SBATCH --job-name=superExecution
#SBATCH --nnodes=nodes
#SBATCH --ntasks-per-node=tasksPerNode
#SBATCH --time=requiredTime
#SBATCH --export=NONE

module load toolModule/toolVersion

workingDir=/scratch/yourCasePath

# (Checks for valid arguments go here)

# (Checks for valid staged data go here)

cd $workingDir

srun toolName arguments
```

Slurm Batch Script for Storing Results into Acacia

storeIntoAcaciaTar.sh:

- Use the `copy` partition.
- S3 clients work better with 2 or more `cpus`.
- Load the necessary modules.
- Use variables for easy reuse of the script.
- Perform checks for valid arguments (not shown here).
- `tar -c` packs the list of directories into the `.tar` file.
- `mc cp` copies the `.tar` file into the indicated path in Acacia.

```
#!/bin/bash --login
#SBATCH --account=projectname
#SBATCH --job-name=storeTar
#SBATCH –partition=copy
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=2
#SBATCH --time=requiredTime
#SBATCH --export=NONE

module load miniocli/version

workingDir=/scratch/yourCasePath
cd $workingDir

whatToTar=( file1 dir1/ dir2/file* ) #These are in workingDir
newTarName=newTarName.tar
fullPathInAcacia=aliasName/bucketName/prefixPath

# (Checks for valid arguments go here)

srun tar -cvf $newTarName "${whatToTar[@]}"

# (Checks for correct taring go here)

srun mc cp --md5 --quiet $newTarName \
     "${fullPathInAcacia}/" > /dev/null

# (Checks for correct transfer go here)
```

Coordinate Different Steps with Job Dependencies

masterWorkflow.sh:

- Use the `--dependency` option of the scheduler to coordinate the execution of the steps.
  - The `cut` command is used to isolate the fourth field of the output (JobID) and assign it to a variable for later use.
  - In this case the `afterok` option is used for jobs to start only after successful execution of the parent job.
- Execute the master batch script on the command line, which will submit the desired jobs with the desired dependencies.

```
#!/bin/bash

stageJobID=$(sbatch --parsable \
stageFromAcaciaTar.sh | cut –d ”;" -f 1)

superJobID=$(sbatch --parsable \
--dependency=afterok:$stageJobID \
superExecutionSetonix.sh | cut –d ”;" -f 1)

storeJobID=$(sbatch --parsable \
--dependency=afterok:$superJobID \
storeIntoAcaciaTar.sh | cut –d ”;" -f 1)

squeue -j "$stageJobID,$superJobID,$storeJobID" \
       -o "%7i %9P %8j %2t %10r %30E"
```

```
$ ./masterWorkflow.sh
JOBID   PARTITION NAME     ST REASON     DEPENDENCY
5534787 copy      storeTar PD Dependency afterok:5534786(unfulfilled)
5534786 work      superExe PD Dependency afterok:5534785(unfulfilled)
5534785 copy      stageTar PD Priority   (null)
```

How do I get help?

Migration Documentation & Migration Guides

- Setonix Migration Guide
- Setonix User Guide
- Supercomputing Documentation
- Acacia User Guide

Migration Training Materials & Video Recordings

- Upcoming Migration Training
- Recordings: Pawsey YouTube Setonix Migration Phase 1 Playlist, and Managing Data at Pawsey
- Materials: Setonix Migration Training Materials (PDFs)

Help Desk

- Help Desk
- Email: help@pawsey.org.au