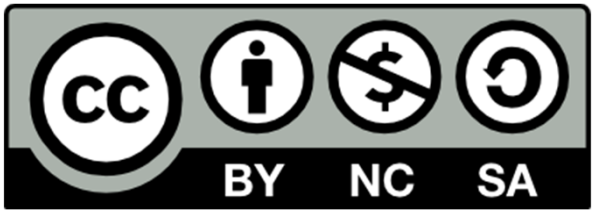
Good practice for transferring data

Caoimhín Laoide-Kemp
Andy Turner
EPCC, The University of Edinburgh

Reusing this material



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.

archer

epcc | THE UNIVERSITY OF EDINBURGH

Useful Links

- Data Management Guide:
  - http://www.archer.ac.uk/documentation/data-management/
- User Guide – ARCHER file systems:
  - http://www.archer.ac.uk/documentation/user-guide/resource_management.php#sec-3.3
- Globus Online:
  - https://www.globus.org/

Spoilers!

- Combine small files into single larger archive files before transferring
- Use the right tool:
  - Do you really need to use rsync?
  - Is a parallel data transfer tool really required?
- Watch out for compression/encryption overheads
- Be aware of the weakest link in the transfer chain

Overview

- ARCHER/RDF file systems and layout
- Combining files – archiving
- Copying data: ARCHER to/from RDF
- Transferring data: on/off the RDF

ARCHER/RDF file systems

ARCHER/RDF file systems

`/home`: backed-up, NFS, available on login, serial and service nodes.

`/work`: **not backed-up**, Lustre parallel file system, available on login, serial, service and compute nodes.

RDF: backed-up only for disaster proofing **(accidental deletion recovery not supported)**, GPFS, available on login nodes (and serial nodes).

Accessing the RDF

Directly mounted on ARCHER login and serial nodes at:

```
/epsrc
/nerc
/general
```

RDF additionally has its own Data Transfer Nodes (DTNs): **dtn01.rdf.ac.uk**, **dtn02.rdf.ac.uk**. Should be used when transferring between the RDF and a remote machine.

RDF also has a Data Analytic Cluster (DAC): **login.rdf.ac.uk.** Can use the scheduler here for long-running archiving and compression tasks

Combining files: archiving

Archiving – Motivation

More efficient use of the file system – single file requires fewer metadata operations to move/copy/access.

Can dramatically improve performance, especially with a large number of small files.

Example, 23GB of data = ~13000 32KB-5MB files:

```
$> time cp -r mydata /general/z01/z01/user/

real    59m47.096s
user    0m0.148s
sys     0m37.358s
```

Archiving – Motivation

Same files in an archive:

```
$> time cp mydata.tar /general/z01/z01/user/

real    3m3.698s
user    0m0.008s
sys     0m33.958s
```

Some initial overhead required for archive creation (~15 mins) but time saved on subsequent accesses.

Serial queues on ARCHER or RDF DAC should be used for any long running tasks.

Archiving – Utilities

Common archiving utilities on ARCHER/RDF:

- tar
- cpio
- zip

Some technical differences but choice mostly personal preference.

Generally recommend forgoing compression to speed up process but there is a compression/transfer time trade-off.

Archiving – tar creation

Ubiquitous “tape archive” format.

Common options:

-c create a new archive
-v verbosely list files processed
-W verify the archive after writing
-l confirm all file hard links are included in the archive
-f use an archive file

Example command:

```
tar -cvWlf mydata.tar mydata/
```

Archiving – tar extraction and verification

-x extract from an archive

```
tar -xf mydata.tar
```

-d "diff" archive file against a set of data

```
$> tar -df mydata.tar mydata
mydata/damaged_file: Mod time differs
mydata/damaged_file: Size differs
```

Note: tar archives do not store file checksums

Original data must be present during verification.

Archiving – zip creation

Widely used and supported by most major systems, including current versions of Windows.

Common options:

-r recursively archive files and directories

-0-9 compression level (-0 recommended on ARCHER)

Example command:

```
zip -0r mydata.zip mydata
```

Note: zip files **do not preserve hard links** (data is copied).

Archiving – zip extraction and verification

Uses a separate utility for extraction.

```
unzip mydata.zip
```

-t test archive (zip file stores CRC values by default)

```
$> unzip -t mydata.zip
Archive: mydata.zip
      testing: mydata/  OK
      testing: mydata/file    OK
No errors detected in compressed data of mydata.zip.
```

Copying data: ARCHER to/from RDF

Copying – Local Copy

```
cp –r source /epsrc/gid/gid/destination
```

Copying to the mounted RDF filesystem exactly the same as a normal copy between directories.

```
rsync –r source /epsrc/gid/gid/destination
```

Pro: rsync will not attempt to transfer files that already exist.

Con: this “mirroring” requires a large number of metadata operations, slowing performance.

Recommend rsync over cp when resynchronising a previously copied directory containing large files.

Usually best not to use “-z” (compression) option to rsync

Copying – Local Copy

Remember: must be done on a node where the two file systems are mounted:

- ARCHER login nodes
- ARCHER serial nodes

Transferring data: on/off RDF

Transfer – Utilities

Via SSH

- scp
- rsync

For very large transfers

- Globus Online
- (bbcp)

Copying – SSH Tools

For remote transfers DTNs should be used.

```
scp –r source user@dtn01.rdf.ac.uk:[destination]
```

Analogue of standard cp.

```
rsync –r –e ssh source
user@dtn01.rdf.ac.uk:[destination]
```

Same utility for both local and remote transfers.

Can also transfer data directly off ARCHER (without RDF) but need to use the serial queues/PP nodes as no DTNs available.

Copying – SSH Performance

All traffic encrypted – secure but performance penalty.

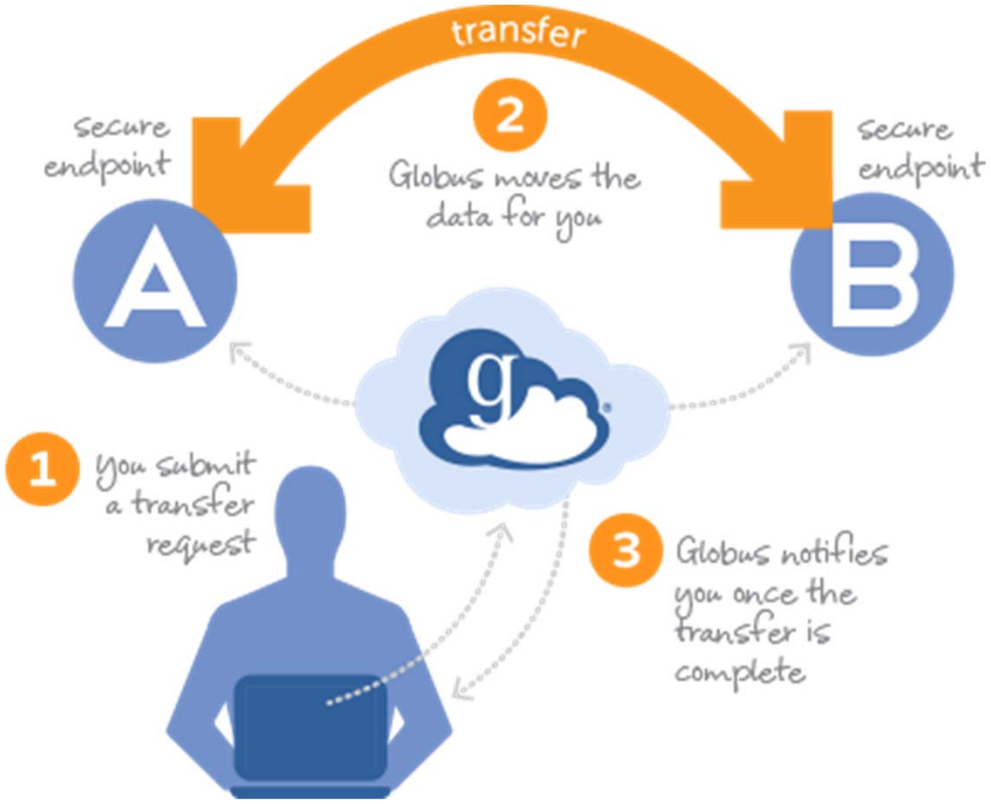
Different ciphers can be used to improve speed.

Algorithm “arcfour” usually fastest but least secure:

```
scp -c arcfour …
rsync -e “ssh -c arcfour” …
```

Lots of files also introduce a large overhead so combine using and archiving tool wherever possible.

Large Transfers – Globus Online



Register for an account at: https://www.globus.org/

- Endpoint for RDF is called **Archer RDF** or **archer#rdf**
- Use your RDF username and password to activate the endpoint

archer

epcc | THE UNIVERSITY OF EDINBURGH

Large Transfers – Globus Online – Performance

Uses GridFTP parallel file transfer to get best performance. Performance is limited by:

- Network bandwidth between two endpoints
  - Often large for two servers at different locations
  - Can be limited for transfers to local laptop/workstation (e.g. wifi, 1 Gpbs ethernet)
- Storage access bandwidth
  - Large for large files on parallel file system
  - Small for many small files
  - Can be small for local storage (e.g. single disk, over USB)

Summary

Summary

- RDF mounted directly on ARCHER login nodes. DTNs available for remote transfers
- Archiving improves performance for copying and transfer. Be aware of metadata operation bottleneck with lots of (small) files.
- Beware compression in rsync – can lead to bottleneck on CPU performance (avoid "-z" rsync option to mitigate)
- Beware encryption in ssh – can lead to bottleneck on CPU performance (use *arcfour* to mitigate)
- Globus Online can access best performance for large data transfers
- Be aware of the weakest link in your data transfer chain (e.g. low network bandwidth, low storage bandwidth)

For advice contact: support@archer.ac.uk