



SURF

Moving data

on HPC systems

Rohit Goswami

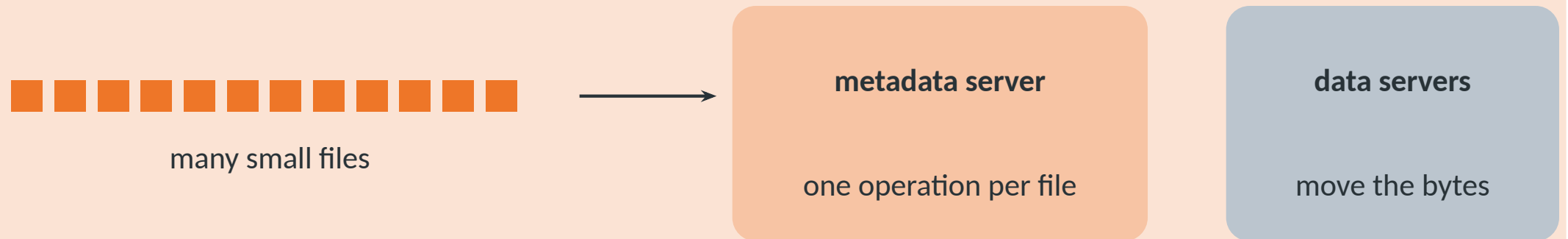
Pack, copy, and stage on a parallel file system

October 26, 2026

rgoswami@ieee.org

Why copying hurts

A parallel file system splits the work. **Data servers** move bytes. A **metadata server** records each create, open, and close. Copying a million small files asks that server for a million operations, and the data servers wait. Pack the files first.



One archive is a single metadata update. The data servers then stream that one file.

Where data lives

Keep code and notes in **home**. Do the heavy copy on **scratch**. Keep the group's long-lived data on the **project** file system. Use the **node disk** only inside a job: it is the fastest, and it disappears when the job ends.

home

/home/\$USER
backed up

scratch

/scratch-shared
not backed up

project

/projects
shared quota

node disk

\$TMPDIR
job lifetime

Pack, then checksum

Before the move

Turn a directory of small files into one archive, then record a checksum. Check it again after the copy. A squashfs image is one file you can mount read-only.

GNU tar, sha256sum, and mksquashfs are the tools.
- - sort t=name gives archive entries a stable order.

Commands

```
tar --sort=name -cf run.tar run/  
sha256sum run.tar > run.tar.sha256  
sha256sum -c run.tar.sha256  
mksquashfs run run.sqfs -comp zstd
```

rsync, done well

One careful copy

- `-a` keeps links, times, and permissions.
- `--partial` leaves a broken file in place so the next run continues.
- `--info=progress2` is one progress line for the whole transfer.
- `--checksum` reads every byte when size and time are not enough.
- A trailing slash on the source copies the contents, not an extra directory level.

Resume by running it again

```
rsync -a --partial  
--info=progress2  
run/ /scratch-shared/$USER/run/
```

When you need a full read

```
rsync -a --checksum --partial  
run/ /scratch-shared/$USER/run/
```

Several streams

Parallel rsync

Four streams help when each shard is a large file. They do not fix a million tiny files: pack those first. `xargs -P` and `fpsync` (from the `fpart` package) both run several `rsync` processes.

`--exclude` skips build products. The destination tree must already exist for the `xargs` form, because each call writes one shard.

`xargs`

```
printf '%s\0' shard-*/ |  
xargs -0 -P 4 -I{}  
rsync -a --partial {}/  
/scratch-shared/$USER/copy/{}/
```

`fpsync`, and `excludes`

```
fpsync -n 4 -o '-a --partial'  
/scratch-shared/$USER/run/  
/scratch-shared/$USER/copy/  
rsync -a --partial --exclude '*.o'  
--exclude '.git/' run/ dest/
```

mpiFileUtils

Inside a Slurm job

These are MPI programs from the mpiFileUtils suite. srun starts one of them across the tasks in the job. Options below are from the project manual.

- `dwalk -v` counts files and bytes.
- `dcp -p` copies and keeps permissions, group, and timestamps.
- `dsync` makes the destination match the source.
- `drm --dryrun` lists what a recursive delete would remove.
- `dtar -c` writes a pax archive.

One tool per srun

```
srun dwalk -v "$SRC"  
srun dcp -p "$SRC" "$DST"  
srun dsync -v "$SRC" "$DST"  
srun drm --dryrun "$OLD"  
srun dtar -c -f run.tar run/
```

A real job script

Two Rome nodes

Snellius uses the rome partition for these CPU nodes. The script walks the source and copies it to the destination. EESSI 2023.06 and 2025.06 do not provide mpiFileUtils yet, so you use the module your site provides or build it with EasyBuild.

Submit with `sbatch dcp.slurm` from a directory on scratch, not from home.

dcp.slurm

```
#!/bin/bash
#SBATCH --job-name=dcp
#SBATCH --partition=rome
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --time=01:00:00
#SBATCH --output=dcp-%j.out
set -eu
SRC=/scratch-shared/$USER/run
DST=/scratch-shared/$USER/run-copy
srun dwalk -v "$SRC"
srun dcp -p "$SRC" "$DST"
```

From outside the cluster

scp starts again from the first file if the transfer stops. rsync continues a partial file. Globus moves between endpoints you registered. On Snellius, the iRODS iCommands on the next slide are the documented 2023 modules. The Data Archive is /archive/\$USER on login and staging nodes only. daget stages tape, and compute nodes do not mount /archive. Research Drive works as an rclone WebDAV remote; SURF documents the setup.

Commands from outside

SRC and DEST are Globus endpoints you registered. RD is the rclone remote in the SURF Research Drive example.

```
scp -r run/ USER@snellius.surf.nl:/scratch-shared/$USER/run
rsync -a --partial --info=progress2 run/
USER@snellius.surf.nl:/scratch-shared/$USER/run/
globus transfer SRC:run/ DEST:run/
module load 2023
module load iRODS-iCommands/4.3.0
iput -r run
daget /archive/$USER/run
srun -t 1:00:00 -p staging cp -r /archive/$USER/run /scratch-shared/$USER/
rclone copy run/ RD:run/
```

snellius.surf.nl is the SSH host. archive.surf.nl is the Data Archive login host.

Stage in, stage out, clean up

Around the job

Copy the inputs from scratch to \$TMPDIR, which is the node disk. Write outputs there too. Copy the results back to scratch before the job ends: the node disk is removed for you. Then delete the scratch tree you no longer need.

Use `srun drm` instead of `rm -rf` when the tree is large enough to stall one process. Run `drm --dryrun` first.

In the job script

```
rsync -a --partial  
/scratch-shared/$USER/run/ $TMPDIR/run/  
rsync -a --partial  
$TMPDIR/out/ /scratch-shared/$USER/out/
```

After the results are back

```
rm -rf /scratch-shared/$USER/run  
srun drm --dryrun "$OLD"  
srun drm "$OLD"
```

Checklist

1. **Pack** small files with tar or mksquashfs before a wide copy.
2. **Checksum** the archive before the move and again after it.
3. Put the heavy copy on **scratch**, not in home.
4. Use `rsync -a --partial`, and run the same command again to resume.
5. Open several streams only for **large** files.
6. Stage into `$TMPDIR`, copy results out, then delete the scratch copy.
7. The Data Archive is not mounted on compute nodes. `daget` on a login or staging node first.

| Hands-on

SURF

Predict, then measure

Write the order first

Use bash. `time` prints real, user, and sys: write down real.

1. Make 20,000 small files.
2. Time `cp -r`, copying a packed tar, and `rsync -a`.
3. Checksum the archive, copy it, and check it again.
4. Interrupt one `rsync` and resume it.
5. Stage into `$TMPDIR` in a short Slurm job, then copy the result back.

On a scratch directory

```
mkdir -p src out
yes data | head -n 20000 |
split -l 1 -a 5 -d - src/f
ls src | wc -l
{ time cp -r src out/by-cp; }
tar --sort=name -cf src.tar src
{ time cp src.tar out; }
{ time rsync -a src/ out/by-rsync/; }
sha256sum src.tar > src.tar.sha256
cp src.tar.sha256 out/
( cd out && sha256sum -c src.tar.sha256 )
```

Resume a broken copy

Catch it midway

Predict a short `part/big.bin` after `timeout`, then equal sizes. `--bwlimit` slows the first copy so two seconds is enough to stop it. `--partial` keeps the broken file, and the next `rsync` continues.

`cmp` exits 0 when the bytes match, and it prints nothing.

Commands

```
dd if=/dev/zero of=big.bin bs=1M count=48
mkdir -p part
timeout 2 rsync -a --partial
--bwlimit=2000 big.bin part/
ls -l big.bin part/big.bin
rsync -a --partial big.bin part/
cmp big.bin part/big.bin
```

Stage into the job

A ten-minute job

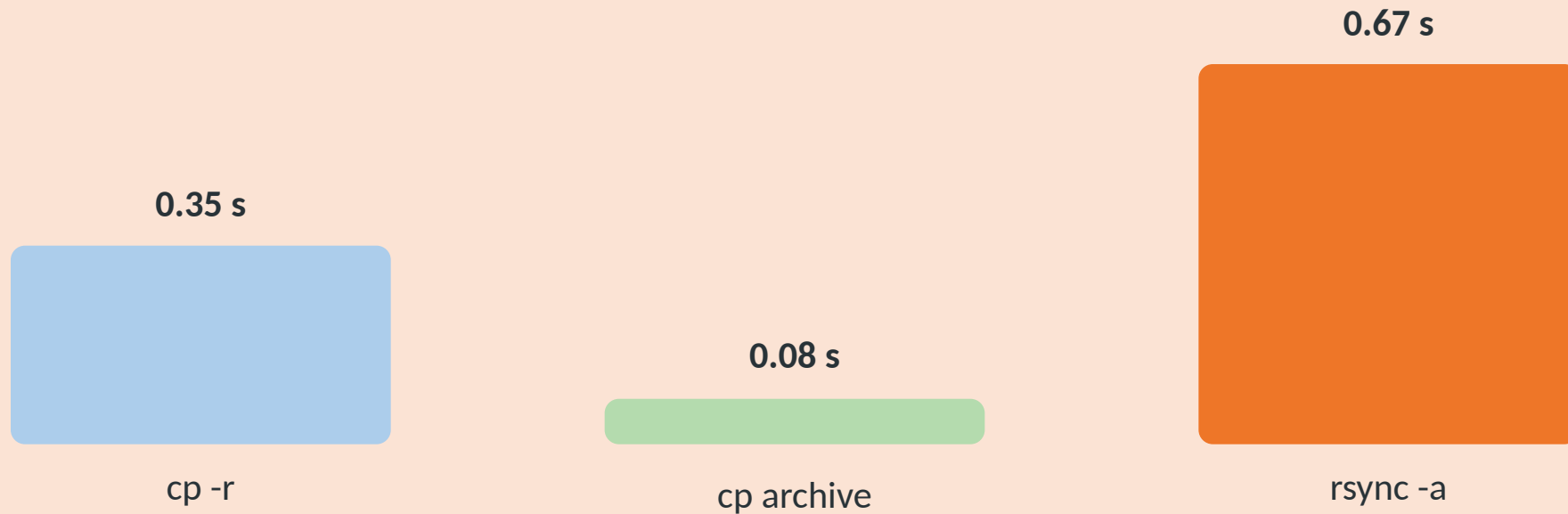
Submit from scratch with `sbatch stage.slurm`.
--constraint=scratch-node selects a node where `$TMPDIR` is the node disk. That disk is removed when the job ends, so the script copies the tree back to scratch before it exits.

`stage.slurm`

```
#!/bin/bash
#SBATCH --job-name=stage
#SBATCH --partition=rome
#SBATCH --constraint=scratch-node
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --time=00:10:00
#SBATCH --output=stage-%j.out
set -eu
rsync -a --partial
/scratch-shared/$USER/src/ "$TMPDIR/src/"
rsync -a --partial
"$TMPDIR/src/" /scratch-shared/$USER/out/
```

An example

Local ext4 example, 26 October 2026; measure your own Snellius times. The archive bar measures copying an existing tar, excluding packing. The other copies create 20,000 files. The interrupted copy held 4128768 of 50331648 bytes; rerunning rsync passed cmp, and sha256sum reported src.tar: OK.



Further reading

- `man rsync`, `man tar`, `man sha256sum`, and `man mksquashfs`.
- mpiFileUtils manual: <https://mpifileutils.readthedocs.io/> (dwalk, dcp, dsync, drm, dtar).
- SURF User Knowledge Base: Connecting to the system (`snellius.surf.nl`).
- SURF: Data Archive from Snellius (`daget`, `/archive`, partition staging) and login host `archive.surf.nl`.
- SURF: How to use Rclone with Research Drive (remote RD), and Snellius filesystems (`$TMPDIR`, `scratch-node`).
- EasyBuild: an mpiFileUtils easyconfig, when your site provides no module.