Use of the dCache Resilient Pool in Grid Jobs

Shreyas Bhat & Ken Herner

6 May 2019

dCache Storage Areas

	Quota/Space	Retention Policy	Retention Lifetime on Disk	Use case	Tape backed?	Path
Persistent	None/~100 TB/expt	Expt-managed	Until manual deletion	Immutable files w/ long lifetime	N	/pnfs/<expt>/persistent
Scratch	None	LRU	Varies (~30 d)	Immutable files w/ short lifetime	N	/pnfs/<expt>/scratch
Tape-backed	None/(O(4 PB))	LRU (evicted from disk)	~30 d	Long-term archive	Y	/pnfs/<expt>/<path>
Resilient	None	None/jobsub managed	~30 d	Tarballs w/ custom code for grid jobs	N	/pnfs/<expt>/resilient

Adapted from https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Understanding_storage_volumes

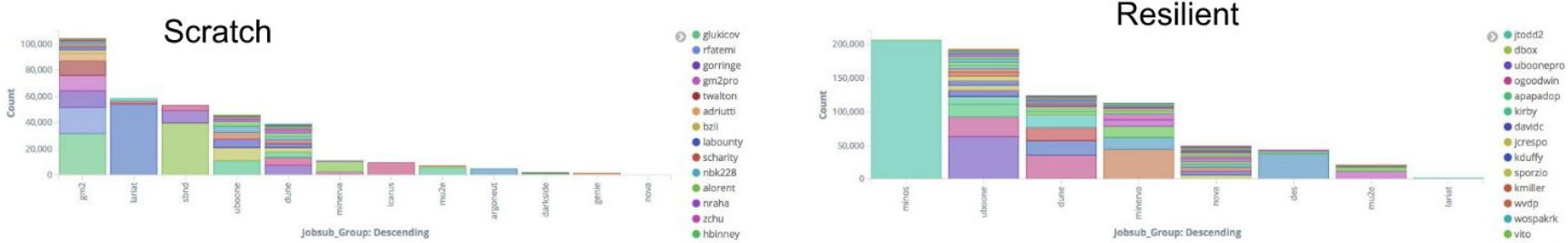
Before Resilient

- User tarballs with custom code uploaded to scratch dCache
- Access these tarballs from hundreds, sometimes thousands of jobs
- Slowed down access for everyone
- Actual example: 5 GB tarball (compressed) uploaded to scratch dCache, accessed by over 1000 jobs

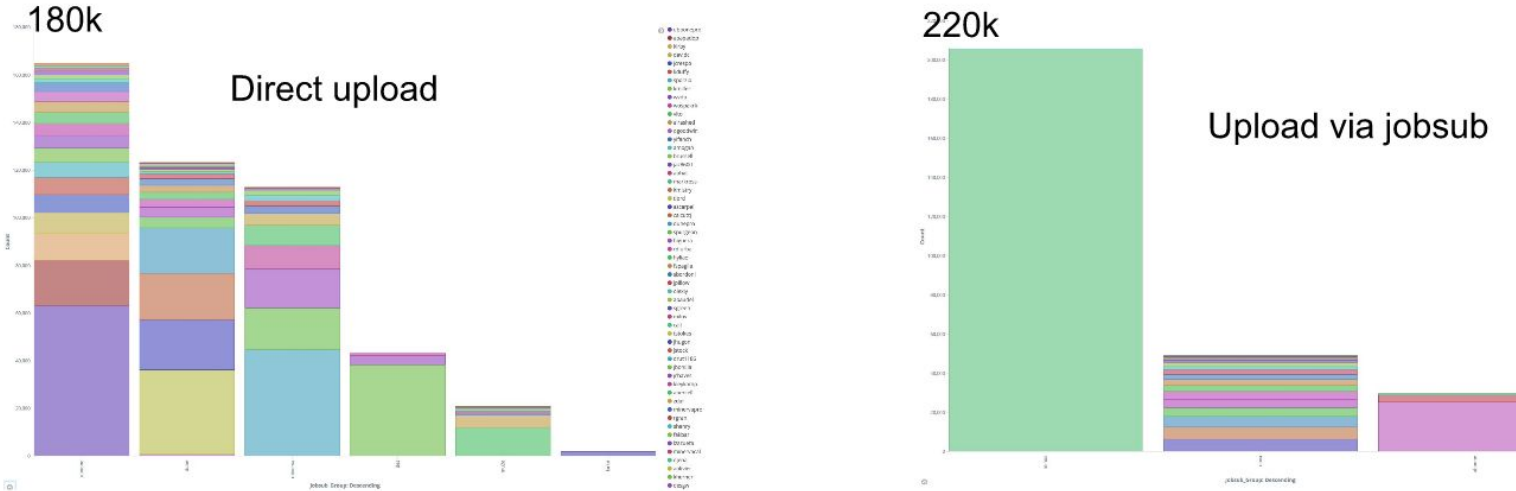
Resilient dCache Pool

- Solution: Come up with new dCache pool that shares hardware with scratch dCache
- Replicate every file in new pool by some to-be-determined factor (20x)
- Since most FIFE users interact with the grid through jobsub_client, create a feature that uploads files and tarballs directly to resilient dCache, and handle cleanup
- Note that for standard experiment-wide code, CVMFS is still the way to go!

Current Status



Number of tar files (code) pulled from dCache during last 7 days by experiment (scratch vs resilient)



Number of tar files (code) pulled from dCache during last 7 days by experiment (direct upload vs jobsub upload)

Cleanup?

Table 1

Experiment	# of Tar Files	Size(GB)	# Old Files	Size of Old Files(GB)
mu2e	496	344	347	263
uboone	5638	260	2377	137
dune	2395	95	1091	25
nova	57	12	57	12

Tar files in resilient areas by experiment (old == didn't access during last 3 months)

Factor of 20!

Using Resilient dCache with Jobsub

- This is why we recommend people use jobsub_client feature to upload files and tarred directories to resilient dCache: *we handle the cleanup!*

Specifics: When given flag to upload a file to the jobsub-configured dropbox (within resilient dCache):

- jobsub_client calculates the hash of the file
- Creates a directory in /pnfs/dune/resilient/jobsub_stage/<hash>
- Uploads file to that directory
- Copies that file to job and in some cases, untars it in the initial working directory
- All file transfer done using IFDHC

Due to hashing mechanism, we won't re-upload the same tarball each time jobsub_submit is run.

Cleanup policy: 30 days old AND not being used by any job in queue

Jobsub file options at submit time

Options:

- -f (NO CLEANUP - not preferred for resilient dCache use)
- -f dropbox:// (Upload files to resilient)
- -f tardir:// (Upload directories to resilient as tarballs)
- --tar_file_name dropbox:// (Upload tarball to resilient, have it unpacked in job)
- --tar_file_name tardir:// (Upload directory to resilient as tarball, have it unpacked in job)

Notes:

- For most cases, use the *--tar_file_name* options for tarballs
- For any of the dropbox:// options, exclude unneeded files (.root, .svn, .git files)
- tardir:// automatically includes a default tar exclusion file that excludes .tar, .git, .jpg, etc. files
    - Make your own: https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Jobsub_submit#tarball-exclusion-file-syntax

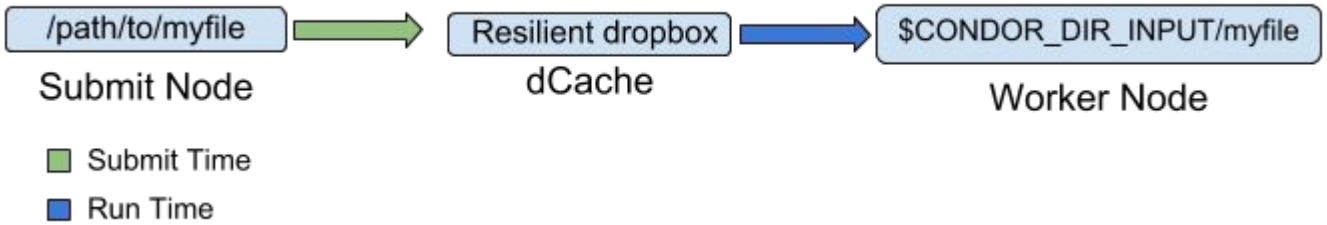
jobsub_submit -f /path/to/myfile

- *Not recommended for use with resilient dCache paths*
- /path/to/myfile must be grid-accessible
- jobsub wrapper will transfer the file from /path/to/myfile to $CONDOR_DIR_INPUT in the job
- No cleanup! Don’t put /path/to/myfile in resilient space and use this.
- Example:

jobsub_submit -G dune -f /path/to/myfile <other args> file://path/to/my/executable <executable_args>

jobsub_submit -f dropbox:///path/to/myfile

- *Best for multiple file uploads to resilient space*
- /path/to/myfile must be accessible *locally* (on the submit node)
- Jobsub will clean this up

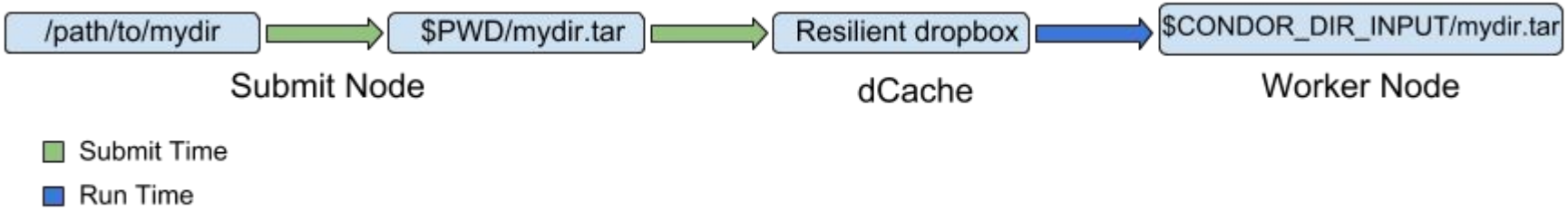


Example:

jobsub_submit -G dune -f dropbox:///path/to/myfile -f dropbox:///path/to/anotherfile <other args> file://path/to/my/executable <executable_args>

jobsub_submit -f tardir:///path/to/mydir

- /path/to/mydir must be accessible *locally* (on the submit node)
- Jobsub will clean this up

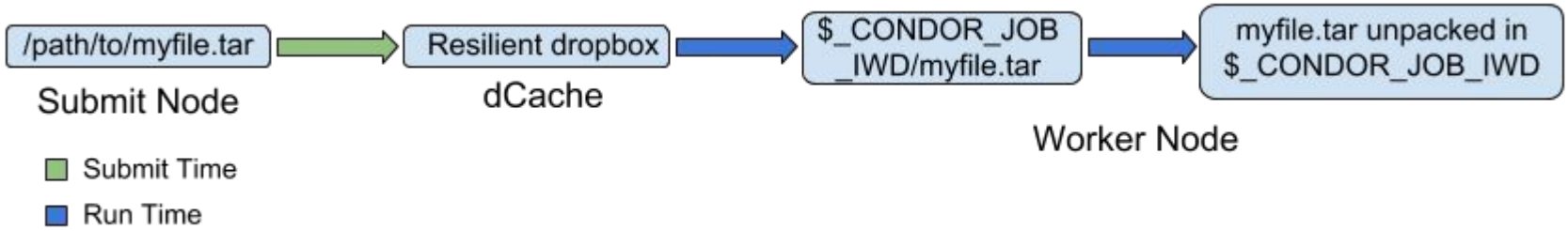


Example:

jobsub_submit -G dune -f tardir:///path/to/mydir <other args> file://path/to/my/executable <executable_args>

jobsub_submit --tar_file_name dropbox:///path/to/myfile.tar

- *Best for single-tarball upload*
- /path/to/myfile.tar must be accessible *locally* (on the submit node)
- Note: $_CONDOR_JOB_IWD is just the directory your job will always start in
- Jobsub will clean up

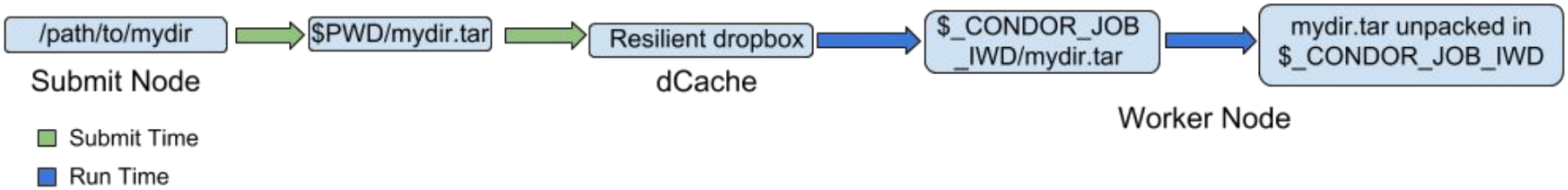


Example:

jobsub_submit -G dune --tar_file_name dropbox:///path/to/myfile.tar <other args> file://path/to/my/executable <executable_args>

jobsub_submit --tar_file_name tardir:///path/to/mydir

- *Best for uploading a single directory*
- /path/to/mydir must be accessible *locally* (on the submit node)
- Jobsub will clean this up



Example:

jobsub_submit -G dune --tar_file_name tardir:///path/to/mydir <other args>
file://path/to/my/executable <executable_args>

Challenges with jobsub and resilient space

- If jobsub_client is tarring up a big directory and uploading it, jobsub_submit takes more time
- Filling up /tmp space
  - This can be worked around by exporting $TMPDIR somewhere else that has more space

Future: jobsub and resilient space

- Filling up /tmp space while creating tarball: proposed --tar_out_dir flag (Redmine 22347) - no need for $TMPDIR workaround
- Sample submit command might look like

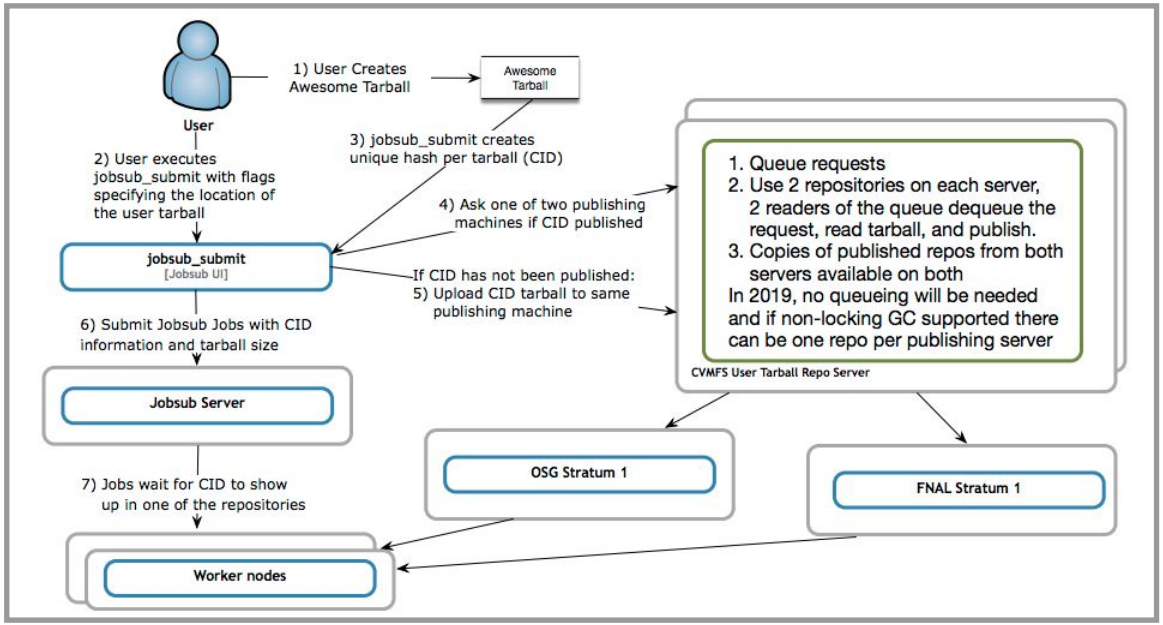
  jobsub_submit -G dune --tar_file_name tardir:///path/to/tarball --tar_out_dir /path/to/all/of/my/job/tarballs_directory

- Jobsub_client would create the tarball and compress it in --tar_out_dir value

Challenges with dCache resilient space

- Using huge amounts of dCache read/write pools (scratch and resilient share this)
- LRU policy for scratch space is invoked a lot more than before (average lifetime decreases)

Future: Rapid Code Distribution on CVMFS (D. Dykstra)



- jobsub_client commands would look exactly the same
- In development phase (diagram above may change)

Thank you!