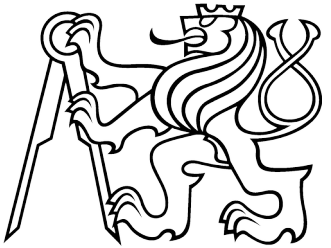


DPM to dCache migration tools

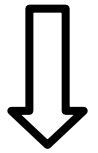
Petr Vokáč (CTU, EGI)

GDB

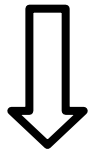
8th June 2022

DPM Evolution

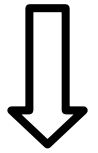
LCGDM



dmlite

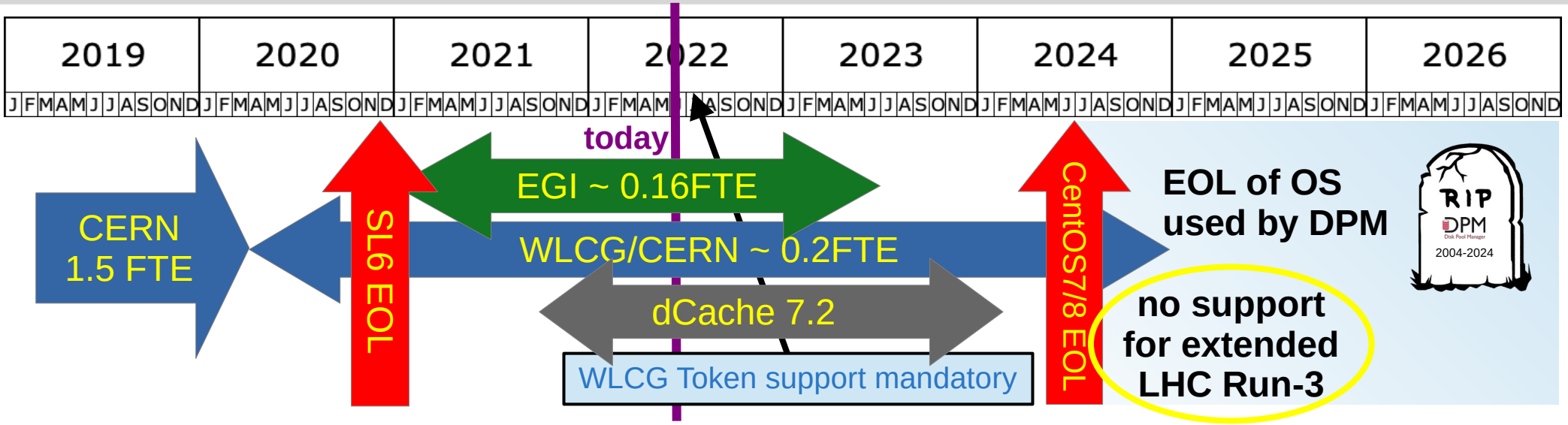


DOME



“dCache”

DOME DPM support



- **WLCG / CERN – March 2020 GDB** The DPM Collaboration
  - support till the **end of LHC Run-3**, but since this announcement
    - CentOS8 EOL changed
    - LHC Run-3 extended
    - no DPM for supported OS
    - no new features
- **EGI – provide dCache (7.2) migration tools**
  - help sites to migrate **DPM to dCache**
  - interested DPM sites should ***move before summer 2023***
  - reduced time for solving DPM issues not related to migration

DPM migration strategy

- Plans presented in past meetings
  - Spring 2021 HEPiX - migration strategy survey
  - December 2021 GDB - DPM sites in France (status and plans)
  - January 2022 GDB - DPM sites in UK (status and plans)
  - February 2022 GDB - DPM migration plans in Switzerland
  - April 2022 GDB - GRIF plans for DPM replacement (EOS)
  - private communication with clouds / DPM site administrators
- Different **migration** strategies
  - disk-less, consolidate, **dCache**, EOS, CEPH
  - different plans even within same country / cloud

small sites without dedicated storage admin(s) prefers simple solution

- DPM sites already started migration or plan to finish migration within ~ year time-frame
  - most of (LHC) sites is aware of DPM EOL
  - be done with majority of sites by 2023 DC feasible

GGUS storage sites, ask for production token support at the beginning of 2023

DPM to dCache migration

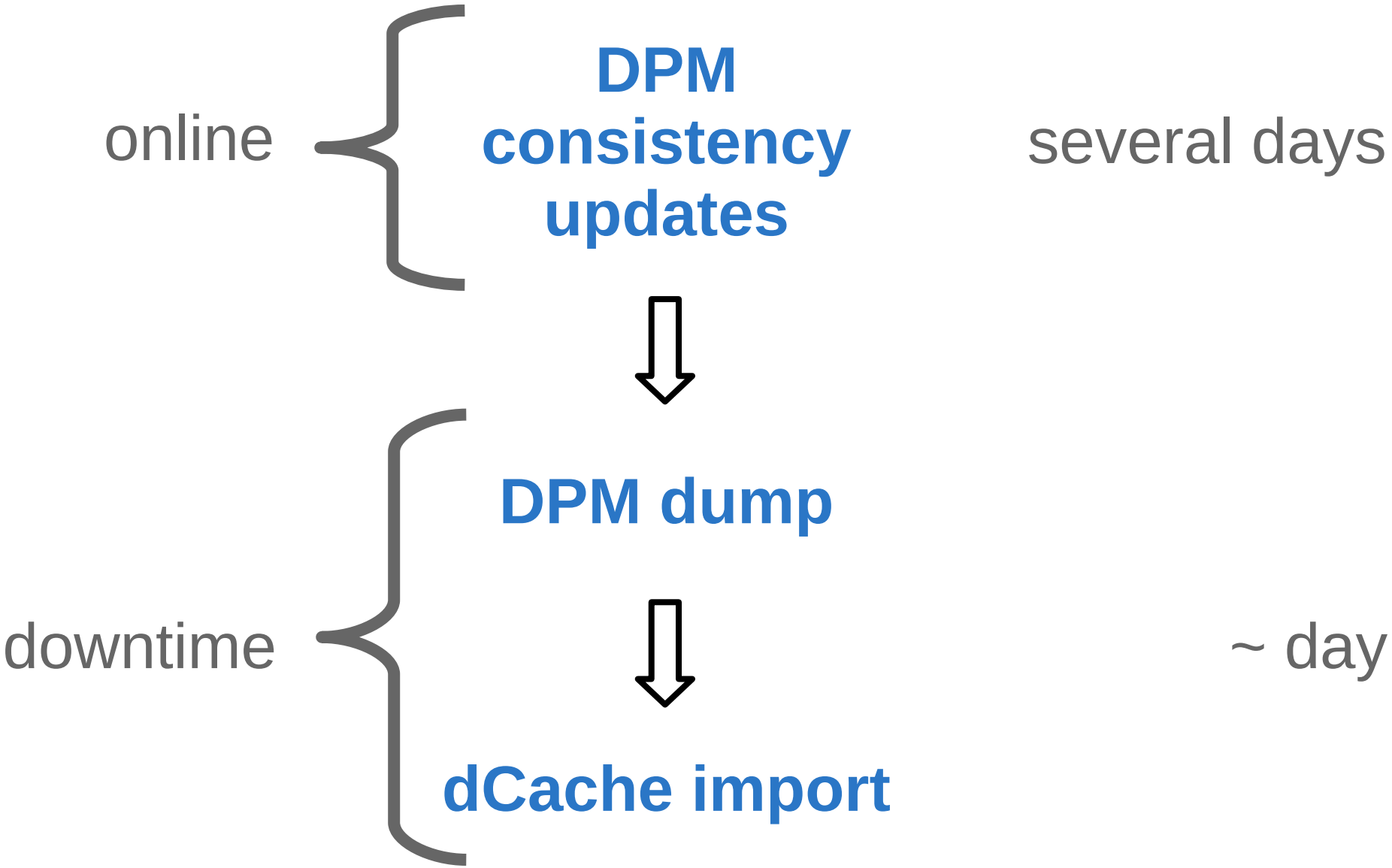
- In-place dCache migration is just **one of available options**
  - sites should consider their future plans first
  - provide easy migration path to the compatible storage
- Transparent migration
  - ***Migrate*** just ***catalog*** (database) and keep ***files untouched***
    - both SE store files on ***posix filesystem***
- No visible difference for clients (Rucio, FTS, gfal2, …)
  - dCache can easily match DPM features & provide more
    - sometimes with slightly more complex configuration
  - same protocols (HTTPS, xroots, gsiftp, SRM)
  - same hosts:ports / firewall configuration (almost)
  - same authentication (X.509, +tokens), different WLCG SRR location
- ***dCache DPM replacement transparent for end users / VOs***
  - **Goal: 1 day downtime DPM → dCache migration**

CentOS7 and newer required for headnode and diskhodes **after migration** (~20% still rely on SL6)

no direct migration from **legacy DPM**

make ***migration*** as ***simple*** as legacy to DOME DPM transition

Storage migration in three steps



https://twiki.cern.ch/twiki/bin/view/DPM/DpmDCache

Migration steps in more details

- Online DPM consistency checks / fixes with dmlite-shell (4 steps)
  - meanwhile install (minimal) python3 **`migrate.py`** dependencies
- Stop DPM & export namespace and config
  - meanwhile install dCache 7.2 on all DPM machines
  - install PostgreSQL database and create user and databases
- Generate dCache configurations from exported **`config.csv`**
  - optionally modify user & group mapping and path attributes
    - useful for cleaner dCache configuration
  - distribute generated dCache configuration files
  - temporarily start dCache on headnode → init. db tables
- Import **`namespace.csv`** dump in dCache PostgreSQL
  - Import create data files used later on disknodes to link (move) file from DPM to dCache physical location on posix filesystem
- Start dCache services + cleanup DPM (files, MySQL db, packages)

!!! terminated import can't be re-tried without cleanup

Migration commands for hosts

```
dpmheadnode$ dmlite-shell --log-level=INFO --log-file=/tmp/dpm-lost-and-dark.log -e 'dbck lost-and-dark-show script' > /tmp/dpm-lost-and-dark.sh
dpmheadnode$ sh /tmp/dpm-lost-and-dark.sh
dpmheadnode$ dmlite-shell --log-level=DEBUG --log-size=104857600 --log-file=/tmp/dpm-dbck.log -e 'dbck dpm-dbck update'
dpmheadnode$ dmlite-shell --log-level=DEBUG --log-size=104857600 --log-file=/tmp/dpm-dbck.pool-file.log -e 'dbck pool-file update nthreads=8'
dpmheadnode$ dmlite-shell --log-level=INFO --log-size=104857600 --log-file=/tmp/dpm-dbck.fill-checksum.log -e 'dbck fill-checksum update nthreads=25'
# declare downtime & stop DPM services
dpmheadnode$ systemctl stop httpd rfiod srmv2.2 dpnsdaemon dpm dpm-gsiftp xrootd@dpmredir
dpmheadnode$ systemctl disable httpd rfiod srmv2.2 dpnsdaemon dpm dpm-gsiftp xrootd@dpmredir
dpmdisknodes$ systemctl stop httpd rfiod dpm-gsiftp xrootd@dpmdisk
dpmdisknodes$ systemctl disable httpd rfiod dpm-gsiftp xrootd@dpmdisk
dpmdbnode$ python3 migrate.py --log-level=DEBUG --log-file=dpm-dump.log --dpm-export --dpm-dbhost=dpmdb.fqdn --dpm-dbuser=dpmdb_user --dpm-dbpasswd=dpmdb_secret
# generate dCache configuration files from config.csv created in previous step
dcachedbnode$ python3 migrate.py --log-level=DEBUG --log-file=migrate-dcache-config.log --dcache-config
# install dCache and its database & distribute dCache configs to /etc/dcache on all storage nodes
dcacheheadnode$ yum install -y https://www.dcache.org/downloads/1.9/repo/7.2/dcache-7.2.16-1.noarch.rpm
dcacheheadnode$ alternatives --set java $(alternatives --display java | grep 'family java-11-openjdk' | cut -d' ' -f1)
dcacheheadnode$ chown dcache /etc/grid-security/hostcert.pem /etc/grid-security/hostkey.pem
dcacheheadnode$ mkdir /etc/systemd/system/dcache@.service.d
dcacheheadnode$ cat > /etc/systemd/system/dcache@.service.d/capabilities.conf <<EOF
[Service]
AmbientCapabilities=CAP_NET_BIND_SERVICE
EOF
dcacheheadnode$ ssh-keygen -C admin@localhost -t rsa -N '' -f id_rsa
dcacheheadnode$ cat /root/.ssh/id_rsa.pub > /etc/dcache/admin/authorized_keys2
dcacheheadnode$ scp dcache.conf gplazma.conf ban.conf ban.conf multi-mapfile.group multi-mapfile.user multi-mapfile.vo multi-mapfile.unmapped vo-group.json vo-user.json omnisession.conf LinkGroupAuthorization.conf
dcacheheadnode:/etc/dcache
dcacheheadnode$ scp layout-HEADNODE_FQDN.conf dcacheheadnode:/etc/dcache/layout
dcachedisknodes$ yum install -y https://www.dcache.org/downloads/1.9/repo/7.2/dcache-7.2.16-1.noarch.rpm
dcachedisknodes$ alternatives --set java $(alternatives --display java | grep 'family java-11-openjdk' | cut -d' ' -f1)
dcachedisknodes$ chown dcache /etc/grid-security/hostcert.pem /etc/grid-security/hostkey.pem
dcachedisknodes$ scp dcache.conf dcachedisknodes:/etc/dcache
dcachedisknodes$ scp layout-DISKNODE_FQDN.conf dcachedisknodes:/etc/dcache/layout
dcachedbnode$ yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
dcachedbnode$ yum install -y postgresql14-server
dcachedbnode$ /usr/pgsql-14/bin/postgresql-14-setup initdb
dcachedbnode$ cat > /var/lib/pgsql/14/data/pg_hba.conf <<EOF
# database on headnode
local   all             all                                     trust
host    all             all             127.0.0.1/32            trust
host    all             all             ::1/128                 trust
# database on dedicated dbnode
#host   chimera         dcache          192.0.2.123/32          md5
#host   spacemanager    dcache          192.0.2.123/32          md5
#host   pinmanager      dcache          192.0.2.123/32          md5
#host   srm             dcache          192.0.2.123/32          md5
EOF
dcachedbnode$ # enable connections to postgresql database from remote machines in case you use dedicated dbnode
dcachedbnode$ #perl -p -i -e "s/^#.*listen_addresses *= *'localhost'/listen_addresses = '*'/" /var/lib/pgsql/14/data/postgresql.conf
dcachedbnode$ systemctl enable postgresql-14
dcachedbnode$ systemctl start postgresql-14
dcachedbnode$ createuser -U postgres --no-superuser --no-createrole --createdb --pwprompt --no-password dcache
dcachedbnode$ createdb -U dcache chimera
dcachedbnode$ createdb -U dcache spacemanager
dcachedbnode$ createdb -U dcache pinmanager
dcachedbnode$ createdb -U dcache srm
# start dCache on headnode + configure linkgroups and space reservations + stop dCache on headnode
dcacheheadnode$ systemctl daemon-reload
dcacheheadnode$ systemctl start dcache.target
dcacheheadnode$ sleep 300 # wait for dCache start
dcacheheadnode$ cat admin-cli.psu | grep -v ^# | ssh -p 22224 -l admin localhost
dcacheheadnode$ sleep 300 # wait till PSU configuration gets propagated between dCache services
dcacheheadnode$ cat admin-cli.reserve | grep -v ^# | ssh -p 22224 -l admin localhost
dcacheheadnode$ systemctl stop dcache.target
dcachedbnode$ python3 migrate.py --log-level=DEBUG --log-file=migrate-dcache-import.log --dcache-import
dcachedisknodes$ python migrate.py --log-level=INFO --log-file=migrate-dcache-link.log --link --link-file=data-dpmdisk1.example.com.csv
# start dCache services on all storage nodes
# cleanup physical files from DPM locations and keep just new "dcache" subdirectory
```

Consistency

DPM dump

dCache import

DPM host cmds

- headnode (5)
- disknodes (2)
- dbnode (1)

dCache host cmds

- headnode (16)
- disknodes (6)
- dbnode (14)

Technically it is possible to revive DPM at any migration stage except after last cleanup step

Migration documentation

- Migration process described in the DpmDCache twiki
- Always use most recent dmlite packages from EPEL repository
  - current release dmlite 1.15.2-5, available for CentOS7 & CS8
  - even minor update may safe your time
    - fixed issues with corner cases discovered during prod. site migration
    - more robust with respect to variations in DPM configuration
      - legacy DPM → define quotatokens → consistency → dCache
- Documentation improved since first migration
  - initially written based on steps done with testbed
    - migration tool development & testing
    - important details missing / not written in right order
  - integrated experience with migration of production storages
  - less manual steps, less error prone, more details
  - many steps – even I followed by me while migrating our prod. DPM

no SL6 support in dCache 7.2

special characters correctly escaped

Migrated production sites

- At least four sites already successfully used dCache migration tools
- Different level of support during migration
- Namespace dump & restore takes significant time
  - can increase total downtime in case of problems
  - depends mainly on number of objects and PgSQL core performance
    - biggest known DPM instance at TOKYO – 8PB with 67M objects

GOCDB site name	Date	Storage size [PB]	Storage objects [M]	Consistency updates [days]	DPM export [h]	dCache Import [h]	Downtime [h]
prague_cesnet_lcg2	February	0.6	2				~ 30*
TW-NTU-HEP (BelleII)	May						
praguelcg2	May 14	3.0	47	15	2.5	20*	< 24
RO-07-NIPNE	May 31	5.0	17	5	11	14.5	~ 48

First site – prague_cesnet_lcg2

- Took some time to find right window for downtime
- Done by my colleagues to validate documentation
  - significant communication and related improvements
  - few migration steps too complicated → migration code improvements
    - set right spacetoken size from beginning
    - avoid WriteToken updates after migration (writetag & pushtag)
      - “complicated”, slow and error prone
        - necessary to run in right order (lost almost a day during migration)
      - dCache don’t provide tool to migrate existing files between spacetoken without copying data (implemented in **`migrate.py`**)
- After successful migration we were not able to read files with checksum validation
  - DPM can calculate missing checkum on demand
  - dCache checksum must be calculated / stored during file upload
- Missing functionality that correctly associate files with spacetoken

Taiwan DPM migration

- Successfully migrated their BelleII DPM at the beginning of May
  - just followed documentations
  - no support support from our side
- Issues with CMS instance migration
  - DPM was never fully converted in the DOME mode
  - still relied on legacy DPM stack with SRM protocol
  - no quotatokens defined for stored VO data files
  - migration tools ignored files without spacetoken
    - warning logged during migration
    - only directory structure imported in dCache namespace
    - Workaround using **`config.csv`** path attributes overwrite functionality
  - recent dmlite comes with more robust migrate.py
    - import immediately fails with no spacetoken defined
    - valid spacetoken assigned to the files with missing spacetoken
      - directory structure and spacetoken size

“big DPM” migration – praguelcg2

- So far biggest storage migrated to dCache with ~ 5PB & 50M objs.
- DPM consistency updates took quite some time
  - a lot of issues found, always using most recent dmlite
    - files stored in a wrong DPM pool moved – 1M files / 60TB / 2 days
  - one of supported VO did not use checksums – 10M files / 0.5PB / 2 weeks
- Extensive testing of migration tools
  - export / import validation
  - performance
    - export ~ 50GB MariaDB → 20GB namespace dump (SSD, 2.5hours)
      - export on DB node 2x faster than remote export from headnode
    - imported PostgreSQL size ~ 50GB
      - import from HDD only 2x slower than import from NVMe/SSD
      - import 2x faster on Intel i7-8700K compared to Intel Xeon E5-2630 v4
- Tuning default 100 movers limit per dCache pool (low for ATLAS)

Recent RO-07-NIPNE migration

- Found two minor issues but with no impact on migration downtime
  - lost and dark data consistency updates stuck
  - lost database connection during DPM **`config.csv`** export
    - fortunately discovered by during test export
- Fixed in latest EPEL dmlite release
- Very slow DPM export
  - MySQL database on physical good hw but with spinning disks
  - order of magnitude slower export compared to praguelcg2 site
- Confirmed that HDD have negligible impact on PostgreSQL dCache namespace import performance
- Problems with periodic dCache service restarts
  - big 1PB diskserver caused OutOfMemoryError with default 4GB limit
    - dCache use “excessive” amount of memory to cache metadata
  - details in “troubleshooting section” of the migration documentation

Summary

- Migration tools part of latest EPEL DPM/dmlite release
- At least 4 production sites already successfully migrated to dCache
  - quick and easy
  - clear enough documentation
    - working on dCache equivalents of dmlite-shell commands
  - sufficiently robust migration tools
  - no issues found after using migrated dCache in production
- ~ 60 DPM sites with ~ 90 DPM and ~ 100PB
  - DPM sites considering dCache migration tools
    - start to plan migration now
    - migrate before summer 2023 (EGI support)
  - LHC sites – production quality tokens for 2023 DC
    - DPM implementation not compliant / can’t be used with tokens
- Support for DPM will go down in coming years

BACKUP

Config – user / group maps

- DPM automatically create identity based on VOMS X509 proxy
  - dpm_user → certificate subject, dpm_group → VOMS FQAN
  - some user / group can be merged during migration
  - DPM users / groups without files not in the DPM topology export
- User mapping in DPM export
  - format: 'user',DN[,dcache_user[,dcache_uid]]
  - automatically / manually add dcache_user, dcache_uid
  - remove user line from mapping => use default user for given VO
- Group mapping in DPM export
  - format: 'user',FQAN[,dcache_group[,dcache_gid]]
  - automatically / manually add dcache_group, dcache_gid
- Generate corresponding gplazma2 and vogroup & multimap files
  - primary group from first FQAN + all other mapped FQAN groups
  - specific mapped username + uid or default VO username (group2uid)

~ 100 unique DNs
~ 30 unique FQAN
at prague DPM

overwrite dump with
'path',user,group,mode
configuration for all
subdirectories

Config – modifying permissions

- VO usually don’t need permission granularity for individual users
- dCache user / group mapping must be pre-configured
  - complicated to get info about all users in VO
  - special permissions to access VO user data with X.509 details
- For most VO it’s possible to avoid individual user mapping
  - use generic user that match VO group
  - remove all users from migration config file
- DPM permission model not same as dCache
  - optionally it is possible to overwrite permission during migration
  - cleanup owners, groups, ACLs currently used in DPM
    - format: path,directory,username,groupname,mode,ace_list,spacetoken
    - configuration applied recursively for subdirectories
      - original value used for empty arguments, e.g. path,/dir/,atlas,,,,
    - examples for ATLAS, BELLE, CMS in the documentation

Config migration – space

- Limit dCache available space per-VO / spacetoken
  - allow files to be distributed to all disknodes
  - aggregated performance
- dCache use storage pools, pool groups, links, link group
  - everything automatically generated during namespace import
  - configuration files must be distributed to headnode & disknodes
    - commented with info where to store them
  - dCache admin shell script
    - to create all pool groups, links and link groups
    - define spacetokens
- Directory layout different for dCache
  - migration script executed on disknode create hardlinks to dCache dir
  - also used to cleanup files from origiginal DPM directories
  - should be robust enough to deal with special characters (e.g. ‘\n’)

Steps to get working dCache
0) stop DPM
1) execute DPM export script
2) execute dCache import script
3) distribute dCache config files
4) run commands in admin shell
5) link existing files
6) start dCache

Additional steps

- Some configurations not enabled by default
  - commented out in generated dCache config files
  - described in documentation and briefly directly in config files
  - requires site specific data not available in DPM database
  - particular dCache service may not start with wrong configuration
- Argus
  - local argus server has to be specified before enabling in gPlazma
- StAR EGI accounting
  - configure which data to publish
- WLCG SRR
  - different URL after migration
  - needs to be communicated with VO that rely on SRR
- Telemetry
  - site specific details

Migration strategy

- Site stop providing “grid storage”
  - consolidation of storage sites
  - disk-less / cache only site (small sites)
    - Both options require really good network for data intensive sciences
- Copy all data to the new storage
  - straightforward, but require significant additional (local) space
  - slow (months) and require significant effort also from central Ops
    - e.g. migrating 30 ATLAS DPM sites would be nightmare
  - gives full flexibility choosing new storage technology
    - a lot of sites expressed interest e.g. in erasure encoding
    - HEPiX Erasure Encoding Working Group – still empty
  - operate multiple sites as one storage – NDGF style
    - all complexity handled by central team, reduced expertise at small site
- Migration without moving data transparent to the clients / VOs

HEPiX survey