dCache configuration in the WLCG Tier-2, in practice

(for v2.2.4 on SL6.5 64bit, for the ATLAS experiment)

Gen Kawamura

II. Physikalisches Institut, Georg-August-Universität Göttingen

Summary

- This slide shows you minimum configuration of workable dCache in the ATLAS WLCG
- Important points
  - VO role configuration
  - Space reservation token (for ATLAS)
  - Information provider
  - Tuning of PostgreSQL server
  - Tuning of RAID devices in disk pool

Structure of minimum SE and Pool

- Example: 1 SE + 2 Disk pools
  - SE = head + doors, Pools = Physical RAID disk nodes

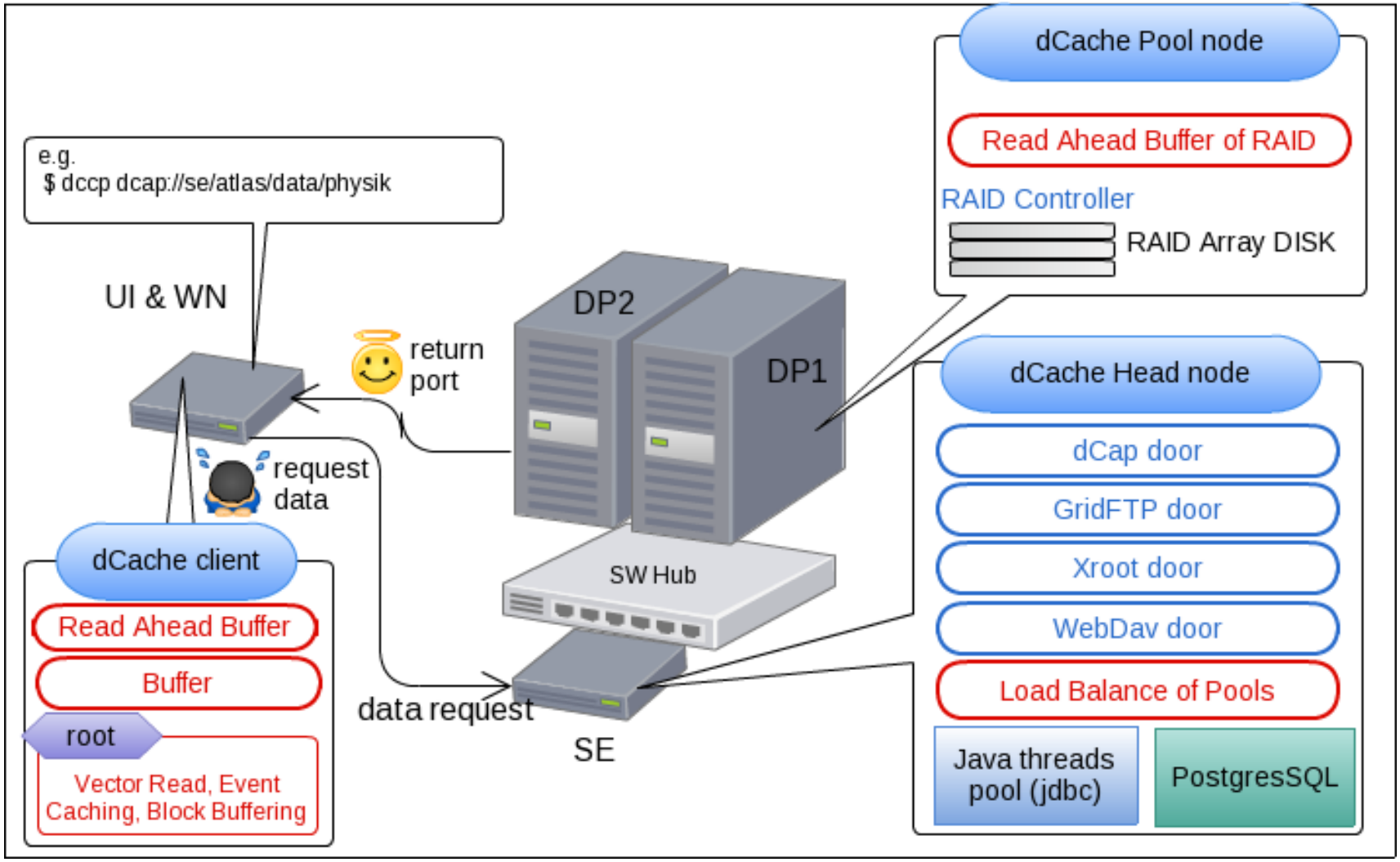


Table of contents

- Installing dCache
- Postgresql configuration
- Information provider in WLCG
- ATLAS VO role configuration
- dCache disk pool and pool configuration
- Space reservation takens in ATLAS
- Performance tuning of RAID cards and etc.

Before installing dCache

- Disabling YUM, to avoid unnecessary corruption or dependency problems in the future
- Installing UMD-2
- Installing LCG CA certificate authority files

Disabling YUM and installing UMD2

- Run “disable_yum.sh” (download from *here*)
- Deleting dag, Installing epel and UMD2

```
EPEL_RPM=epel-release-6-8.noarch.rpm

UMD_RPM=umd-release-2.0.0-2.el6.noarch.rpm

[ -e /etc/yum.repos.d/dag.repo ] && rm -v /etc/yum.repos.d/dag.repo

rpm -e epel-release umd-release

wget http://dl.fedoraproject.org/pub/epel/6/x86_64/$EPEL_VERSION -O
$HOME/$EPEL_RPM

yum -y install $HOME/$EPEL_RPM

wget http://repository.egi.eu/sw/production/umd/2/sl6/x86_64/updates/
$UMD_VERSION -O $HOME/$UMD_RPM

yum -y install $HOME/$UMD_RPM

yum clean all
```

Installing CA

- Activate certificate authority public keys

```
# Installing EGI CA

yum -y install yum-protectbase ca-policy-egi-core fetch-crl


# activate periodic CRL update

chkconfig fetch-crl-boot on

chkconfig fetch-crl-cron on
```

Installing dCache on head node

- Installing required packages and dCache

  yum -y install java-1.6.0-openjdk-devel ruby

  yum -y install postgresql-server postgresql

  yum -y install bdii glue-schema

  yum -y install dcache-server

- Make standard environments with dCache book
  - Samples for this slide can be found
    - */etc/dcache/dcache.conf*
    - /etc/dcache/layouts/*$(hostname).conf*
    - /var/lib/dcache/config/*poolmanger.conf*

PostgreSQL configuration

- Basic configurations
  - Tuning on auto vacuum
    - In /var/lib/pgsql/data/postgresql.conf
      - autovacuum = on
  - Configuring access right for localhost
    - Add following lines to /var/lib/pgsql/pg_hba.conf

```
# configuration for dCache chimera and SRM database
local   all         all                               trust
host    all         all         127.0.0.1/32          trust
host    all         all         ::1/128               trust
```

Information provider

- An example of “info-provider.xml” is *here*
  - Add Grid site name and host name to dcache.conf

```
SITE_NAME=DE-ATLAS-T2

echo “info-provider.site-unique-id=$SITE_NAME
info-provider.se-unique-id=$(hostname -f)
info-provider.se-name=dcache@$(hostname -f)" >>
/etc/dcache/dcache.conf

ln -s /usr/sbin/dcache-info-provider
/var/lib/bdii/gip/provider/dcache-info-provider
```

ATLAS VO configuration

- Creating a proper *gplasma.conf* in /etc/dcache
  - Activate voroemap
  - Generate /etc/grid-security/grid-vorolemap

    /usr/sbin/dcacheVoms2Gplasma.py -r -a -c /etc/dcache/dcacheVoms2Gplasma.conf

    - An example of dcacheVoms2Gplasma.conf for ATLAS is *here*
  - Appendix: how to generate grid-mapfile

    yum -y install edg-mkgridmap

    edg-mkgridmap --output=/etc/grid-security/grid-mapfile --safe

dCache pool and pool configuration

- Has mounted a physical pool in /pool1 and generating a 100GB pool

  pool_dir=/pool1

  pool_name=dp1.dcache.german-t2.de__dpool1_atlas

  pool_domain=$(hosnname)Domain

  pool_size=100

  /usr/bin/dcache pool create --lfs=precious --size=${pool_size}000000000 ${pool_dir}/${pool_name} $pool_name $pool_domain

- The pool information needs to be defined in head node
  - /var/lib/dcache/conf/*poolmanager.conf*

    psu create pool dp1.dcache.german-t2.de__dpool1_atlas

    psu addto pgroup atlas dp1.dcache.german-t2.de__dpool1_atlas

dCache pool and pool configuration

- Making a minimum dCache pool node

  yum -y install java-1.6.0-openjdk-devel ruby

  yum -y install dcache-server

  - Edit following files
    - */etc/dcache/dcache.conf*
    - /etc/dcache/layouts/*$(hostname).conf*

Space reservation token in ATLAS

- Creating a directory structure. The typical structure is the followings
  - domain=german-t2.de
  - vo=atlas, or vo=ops
  - dir=/pnfs/${domain}/data/$vo
  - Generating atlas dir by chimera-cli

```
vo=atlas
chimera-cli Mkdir $dir
chimera-cli Chown $dir atlas001
chimera-cli Chgrp $dir atlas
chimera-cli Chgrp $dir 775
```

Space reservation token in ATLAS

- Generating tag for pnfs directoy

```
echo "STATIC" | chimera-cli Writetag $dir sGroup
echo "StoreName $vo" | chimera-cli Writetag $dir OSMTemplate
echo "ONLINE" | chimera-cli Writetag $dir AccessLatency
echo "REPLICA" | chimera-cli Writetag $dir RetentionPolicy
```

- Making 100GB of a spacetoken “ATLASLOCALGROUPDISK” in admin interface

```
ssh -c blowfish -p 22223 admin@localhost <<EOF
cd SrmSpaceManager
reserve -vog=$vog -vor=* -acclat=ONLINE -retpol=REPLICA -lg=atlas-link-group -desc=ATLASLOCALGROUPDISK 100GB "-1"
```

Space reservation token in ATLAS

- Generating tag for pnfs directory

  echo "STATIC" | chimera-cli Writetag $dir/atlasgroupdisk sGroup

  echo "StoreName $vo" | chimera-cli Writetag $dir/atlasgroupdisk OSMTemplate

  echo "ONLINE" | chimera-cli Writetag $dir/atlasgroupdisk AccessLatency

  echo "REPLICA" | chimera-cli Writetag $dir/atlasgroupdisk RetentionPolicy

  - Adding an explicit spacetoken ID to spacetoken directory

    echo "$spacetoken_id" | chimera-cli Writetag $dir/atlaslocalgroupdisk WriteToken

- In head node, a SRM configuration is applied to */etc/dcache/LinkGroupAuthorization.conf*
- Sample scripts can be found *here*

Tuning of RAID card in disk pool

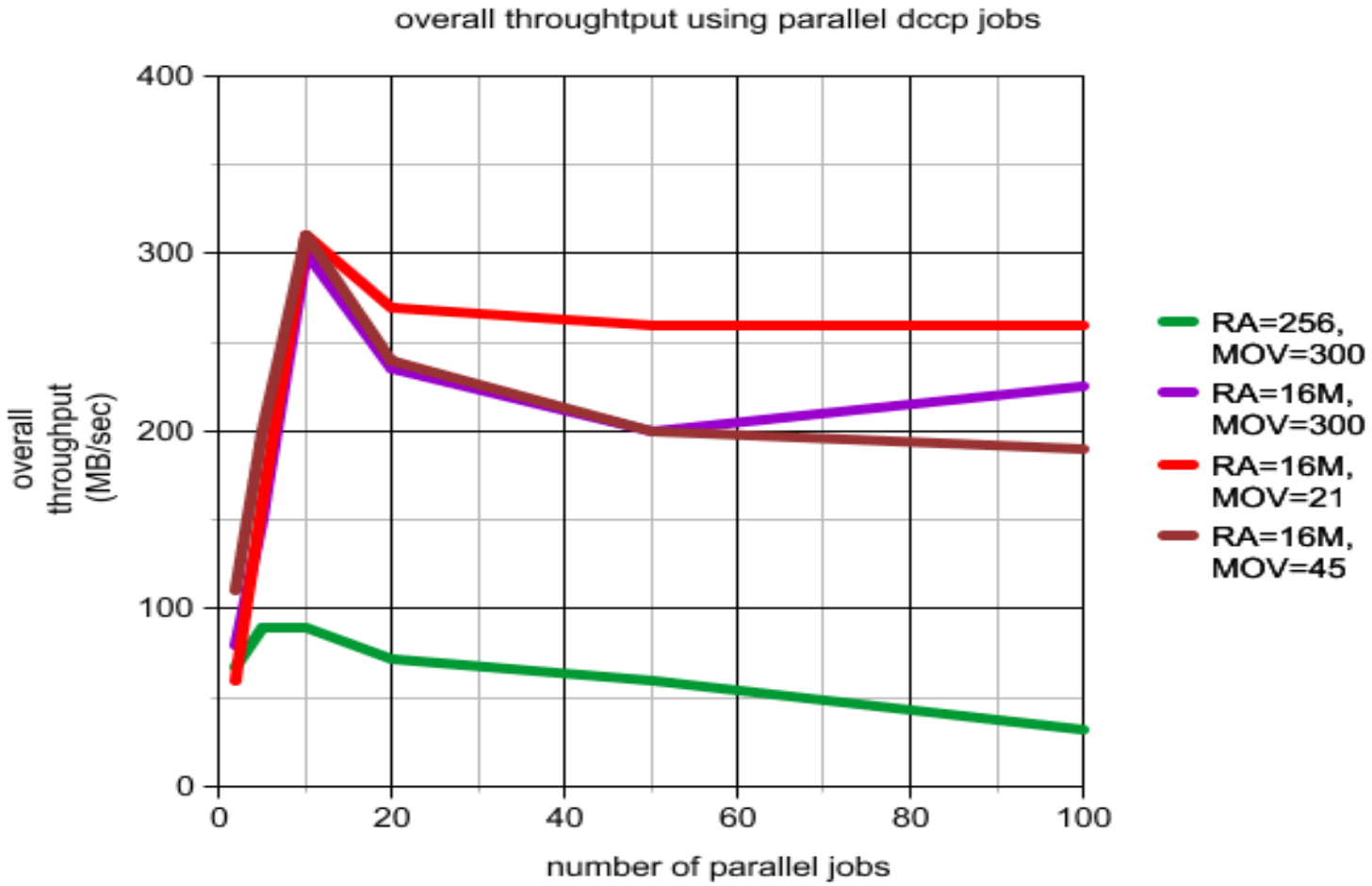
- One important parameter is “read ahead buffer size” for block devices. This tuning is effective when number of parallel requests is >20.
  - e.g. (/dev/dm-0 is a logical RAID disk)

    blockdev –getra /dev/dm-0
    - 64 MByte (512 Byte * 131072 Blocks) is sufficient

    blockdev –setra 131072 /dev/dm-0

Example of performance tuning

- Example: performance with parallel copy jobs



Additional standard tuning

- Do not forget to optimize jvm memory
  - in /etc/dcache/dcache.conf

    dcache.java.memory.heap=2048m

    dcache.java.memory.direct=2048m
- Making performance of PostgreSQL better
  - Very typical tuning parameters
    - in /var/lib/pgsql/data/postgresql.conf
      - max_connections = more than 1000
      - shared_buffer = 25% of memory
      - effective_cache_size = 50% of memory