

Dcache Monitoring Overview Update

Alex Kulyavtsev

9/28/2009

v1.2

Intro

I've been charged by dCache project leader Timur Perelmutov to provide current description of dcache monitoring facilities based on recent experience migrating of dcache services to new node as follows:

collect the following information about each of the component: purpose, names of the executables, their location in cvs, their location once installed in monitoring node, dependence on the other components and files, configuration files and other sources of configuration information used by these programs, inputs and outputs, ways to gather information (i.e. does the component connects to other services or nodes to gather information or run other programs or scripts, etc), manner in which the components are executed (scripts or servlets in tomcat or apache httpd, run by crond or at, or just continuously running services, etc.)”

The dcache infrastructure on fndca including monitoring has been described in great detail by Rob Kennedy at [1]. I take advantage of his description of the scripts whenever is possible. The former description is based on “pre-rpm” dcache installation style. I preserve language of this memo and the language of task charge.

1. Fndca Infrastructure and Organization, Rough Draft 2, 19 Dec 2006, RDK

<https://plone4.fnal.gov/P0/DCache/dcachedoc/fndca-components/fndca-infrastructure-20061219.txt/view>

Directories

The rpm based dcache admin installation installs vanilla dcache distribution into /opt/d-cache area and copies some required configuration files to that area. The bulk of the monitoring scripts still runs from ~enstore/dcache-deploy/dcache-fermi-config area, thus the following description is still valid.

Historical FNAL Deployment Organization

=====

- 1) Overall structure broken down along the file system organization

1.1) ~enstore is the home of dcache installations, based on the history of dCache being brought to FNAL by Enstore developers. It contains a collection of directories to support deployment, a few utilities to support developers, and

some accumulated junk. Note-worthy are:

1. dcache-deploy is the main body of dcache SCRIPTS deployment, described below.
2. dcache-log contains the dCache log files
3. dcache-billing, dcache-ftp-tlog*, dcache-logins, dcache-queues, ... contain plain text records and logs used to support web pages developed by FNAL to track billing, FTP transfers, etc.
4. .bashrc, .bash_profile: setup the base environment for scripts (true for ~root as well)
5. unix.uid.list: manually updated mapping of usernames to uids at FNAL

1.2) ~enstore/dcache-deploy is a symlink to the base of FNAL dcache software infrastructure. In all known instances, it points to ~enstore/dcache-code.

~enstore/dcache-deploy contains:

```
lrwxrwxrwx 1 root  enstore  20 May 19 16:21 classes -> /opt/d-cache/classes
lrwxrwxrwx 1 root  enstore  19 May 19 16:21 config -> /opt/d-cache/config
drwxr-sr-x 28 enstore enstore 12288 Jun 16 15:31 dcache-fermi-config
lrwxrwxrwx 1 root  enstore  17 May 19 16:21 jobs -> /opt/d-cache/jobs
lrwxrwxrwx 1 root  enstore  27 Apr  3 19:07 scripts -> dcache-fermi-config/scripts
```

Historically these were links to the local subtree. Now several links redirected to dcache-core installation dir /opt/d-cache .

dcache-fermi-config: base directory of most dcache service contents

1. the directory itself

contains a disorganized ad hoc collection of scripts and utilities used in dcache operation in addition to what is in its scripts sub-directory (7).

2. fndca, fndcat, cdfst, cdfen, cdfstest, cdf

Base directories for per-instance configuration. Note that cdfen = CDFDCA, cdfstest = CDFDCAT.

Cdfst was intended for rpm installation tests. The way dcache-admin rpm generated, is to use cdfst as a common base for CDFDCA and FNDCA installation, which has led to several configuration conflicts. The deltas for subsystems to files located at cdfst are in system specific directories, like cdfen and fndca. Cdf as a base directory for rpm-based installation is now obsolete.

3. A LOT of other stuff, some ref'd by symlinks from elsewhere

4. scripts -> dcache-fermi-config/scripts/

palliative and administrative support scripts

5. status

more scripts and .awk scripts to collect pools status

1.3) /etc/rc.d/init.d contains symlinks and boot scripts for dcache components such as dcache-core, dcache-pool, postgres-boot, tomcat-boot, monitoring-boot.

1.4) /usr/local/bin contains symlinks to utility programs. Most important

ENSTORE_HOME – this script is called mostly in all other scripts to set environment var E_H pointing to ~enstore home directory. In some scripts written later “~enstore” idiom is used. This approach fails when yp server is unavailable.

1.5) /usr/local/etc/farmlets/

- has links to current farmlet in /opt/d-cache/admin/ area.

1.5) crontabs. The config areas contain crontabs for users root and enstore on the head and monitoring node, user enstore on door nodes, and possibly more.

These are covered in more detail in section 2.2.

1.6) /fnal/ups/prd/www_pages is the "home page" area of the dcache apache server (dcache home WWW page). It exists on the head node serving the traditional "outer" dcache web page created and supported by FNAL and run under apache httpd. The similar directory is on monitoring node. Some pages (transfers.html) are collected to this area on monitoring node (login.list) and then copied to the head node, to be described below. This area is saved in cdcvs under “www_pages”. Scripts generating plots and lists put the results at directory /fnal/ups/prd/www_pages/dcache

Much [TODO]

1. HTTPD non-default configuration settings

– point home location to /fnal/ups/prd/www_pages

- allow to follow symlinks

2. index.html, robots.txt are the usual httpd basics: the default page and web page indexing block respectively.

3. dcache

3.1 *.gif, *.jpg - graphics and images from DESY

3.2 billing

3.3 diskList

3.4 files

3.5 lifetime

3.6 logins

3.7 queue

3.8 running.html - "Running dCache"

3.9 Mapping of monitoring generated files to web pages:

"Daily Billing" <http://fndca3a.fnal.gov/dcache/billing.html>
"File Lifetime Plots" http://fndca3a.fnal.gov/dcache/dc_lifetime_plots.html
"Login Plots" http://fndca3a.fnal.gov/dcache/dc_login_plots.html
"Queue Plots" http://fndca3a.fnal.gov/dcache/dc_queue_plots.html
"Login List" <http://fndca3a.fnal.gov/dcache/DOORS.html>
"P2P Queue" (unlinked) <http://fndca3a.fnal.gov/dcache/p2p.html>
"Restore Queue" <http://fndca3a.fnal.gov/dcache/RC.html>
"Active Transfers" <http://fndca3a.fnal.gov/dcache/transfers.html>

3.10 <http://fndca2a.fnal.gov:8090/dcache/lplots>

3.11 <http://fndca3a.fnal.gov/dcache/files>

3.12 http://fndca3a.fnal.gov/cgi-bin/dcache_files.py

3.13 <http://fndca3a.fnal.gov:2288/statistics>

3.14

1.7) ~enstore/enstore contains the Enstore code distribution from CVS. It was used to build Enstore utilities in place that are used by dCache such as: encp, ecrc, config_server, log_server, and a few minor scripts.

On cdf nodes at present :

- pool nodes : use ups/upd installation to setup encp . Otherwise cvs checkout of "production" tag shall be enough. Setup-enstore script is used and must be kept current.
- Monitoring node: cvs checkout of "production" tag shall be enough. Setup-enstore script is used and must be kept current.
- Head node : we used rpm installation on head nodes. Setup-enstore is in site-specific derectory.

2) Break-down by functional organization

2.1) ~enstore/dcache-deploy/config/cold-start and cold-stop.

These recent scripts define how to do a cold start and cold stop of dCache instance. For cdf these scripts are current and dcache-cold-start is not supported. They are maintained in each instance's config area to allow specialization to be captured in CVS as soon as possible, with efforts to make the scripts general again coming later when time permits. This is a break-down of what the scripts execute as another dimension along which to break down the dCache system description.

1. dgang, rgang

linked from /usr/local/bin

rgang.py

1. dgang is a script layer on top of rgang that adds dcache-specific functionality and ease-of-use. This is the expected administrator interface to rgang.

2. rgang is a frozen python program built from rgang.py

3. rgang.py is a means to execute commands across multiple nodes in distributed dcache system. It can be thought of as rsh coupled with text files describing which nodes are to be contacted.

2. /usr/local/etc/farmlets/*.farmlet - the means to describe what nodes play which roles in a dCache system. All are linked like

admin.farmlet -> /opt/d-cache/admin/admin.farmlet

Typical files are:

1. admin.farmlet = head node, door nodes (NOT monitor nodes)
 2. dcache.farmlet = all nodes in system
 3. head.farmlet = head node only
 4. monitoring.farmlet = monitor node(s) only
 5. pnfs.farmlet = node where PnfsManager runs (NOT pnfsd)
 6. pool.farmlet = any node hosting one or more pools
-
3. [obsolete, TODO] postgresql-boot - starts a postgresql database server. At least the following aspects of dcache use postgresql:
 1. billing
 2. SRM
 3. future monitoring plots
 4. "service httpd start" - FNDCA uses a httpd rpm deployment of Apache httpd, which can be started easily by the "service" interface.
 5. tomcat-boot - starts a tomcat servlet container. At least the following aspects of dcache use postgresql:
 1. Vladimir's old-style monitoring plots
 2. SRM
 3. Lazlo monitoring plots
 6. kdcmux-boot - starts the FNAL-developed KDC multiplexer service. This service allows Java programs to spread their kerberos authentication requests across a suite of KDCs to break the limitation to exactly one KDC imposed by the JAVA Kerberos API. This is crucial to support the load of large kerberos-oriented dCache instances such as CDF dCache.

[TODO] file(s) involved, description of operation.
-
7. General start-up
see cold-start, cold-stop above

8. monitoring-boot - starts the FNAL-specific watchdogs and monitoring support system on the monitor node. This system uses the Unix "at" daemon to execute actions at regular intervals. Exactly which watchdogs or monitoring are executed varies between dcache instances and is defined in this boot script itself. Those which may be applicable to FNDCA include:

Categories: according to primary deliverable of script

Plotter - creates a web page for viewing

Watchdog - send e-mail if a condition is found

Palliative - reduces impact of a weakness in dCache

Notes:

location in cvs:

- I omit prefix dcache-fermi-config until otherwise stated, listed path is relative to this dir.

Common configuration patterns:

- typically script calls /usr/local/bin/ENSTORE_HOME to setup script installation base (always ~enstore home ddir) and set its value at environment var E_H .
- node name is defined through "uname -n" and used as "case \$node" to switch through system name or change script behaviour : marked as *nodename*
- more recent scripts source

scripts/setup-enstore

the script setup-enstore

- sets E_H (enstore home)
- finds and sources setups.sh in set of locations
- depending on system (hardcoded) locates and sources enstore's "setup-enstore" or fakes it's settings when it used only to define system (cdf, fndca) and scripts do not actually use anything from ~enstore/enstore area :
 - for rpm installation of enstore (cdf head node) it will use $E_H/site_specific/config/setup-enstore$
 - otherwise try $E_H/enstore/setup-enstore$
 - on public dcache and newer cdf pool nodes (ups encp installed) it

```
sets ENSTORE_DIR="${E_H}"/enstore,  
setup encp ...  
export ENSTORE_CONFIG_HOST=`"$  
{ENCP_DIR}"/chooseConfig`
```

- on cdf admin nodes encp not required, and it fakes ENSTORE_CONFIG_HOST setting.

See Appendix A for short summary of scripts.

name:

monitoring-boot

purpose:

Master script to start, stop or show monitoring scripts

location in cvs:

monitoring-boot

location once installed in monitoring node:

/usr/local/bin/

/etc/rc.d/init.d/

dependence on the other components and files:

various monitoring scripts it executes (see below)

configuration files and other sources of configuration information:

`$E_H`

system name by parsing link `$E_H/dcache-deploy/config`

or hardcoded

the script has elaborated case statement to select services to run based on the name of the system (defined through node name it runs on).

inputs and outputs:

ways to gather information

connects to other services or nodes to gather information

run other programs or scripts

showat

etc

manner in which the components are executed

(scripts

servlets in tomcat or apache httpd

run by crond or at

at

just continuously running services

onetime

etc

Monitoring-boot script starts several "at" jobs. The set of jobs depends on node name to define hardcoded system name. The common pattern for the wrapper monitoring script is to submit (call) child script and wait until it finish and reschedule itself with "at" command. Thus frequency of execution is hardcoded at the child script (2-15 minutes after completion).

"at" jobs active on cdfcam :

root :

`$E_H/dcache-deploy/dcache-fermi-config/enabled.list`

enstore :

`$E_H/dcache-deploy/dcache-fermi-config/login.list`

`$E_H/dcache-deploy/dcache-fermi-config/login.plot`

`$E_H/dcache-deploy/dcache-fermi-config/pool.stats`

`$E_H/dcache-deploy/dcache-fermi-config/postgres.list`

`$E_H/dcache-deploy/dcache-fermi-config/queue.plot`

`$E_H/dcache-deploy/dcache-fermi-config/restore.list`

`$E_H/dcache-deploy/dcache-fermi-config/retry.Pool`

1. login.list - plotter

Calls

`scripts/listioalldoors.sh`

This script creates the "Login List" web page.

<http://fndca3a.fnal.gov/dcache/DOORS.html>

2. moverls.list – plotter (not acive on cdf)

Calls

`scripts/moverls.sh`

This script creates the "Pool Mover List" web page that is not linked by from the main FNDCA web page.

<http://fndca3a.fnal.gov/dcache/moverls.html>

3. restore.list – plotters

Calls scripts/kill_restore_butincache.sh (todo – check it)

This script creates the "Restore Queue" web page.

<http://fndca3a.fnal.gov/dcache/RC.html>

4. enabled.list - watchdog

Calls scripts/check_poolenabled.sh

This script checks for pools reported offline for at least 6 iterations of checking, and sends e-mail if any.

5. postgres.list - watchdog

Calls

check_postgres

This script checks for postgres database instances needed are running, and sends e-mail if none found.

Configuration:

IP is hardcoded depending on system

6. queue.plot - plotter

Calls scripts/queues.sh

This scripts gathers information and creates plots of the queue levels for each pool in dcache: number of movers active or queued, number of restores active or queued, etc. After final processing done by a cronjob, these plots are shown on the web page:

http://fndca3a.fnal.gov/dcache/dc_queue_plots.html

7. login.plot - plotter

Calls scripts/logins.sh

This script gathers information and creates plots of the login levels for each door in dcache. After final processing done by a cronjob, these plots are shown on the web page:

http://fndca3a.fnal.gov/dcache/dc_login_plots.html

8. pool.stats - watchdog, plotter

Calls

status/updatePoolStatus.sh

status/updateDirectory.sh

Involved in the processing of billing information to create a detailed statistics breakdown by pool or storage class with intermediate information stored in the sub-directory `fndca:~enstore/dcache-statistics`.

Displayed on the web page:

<http://fndca3a.fnal.gov:2288/statistics>

9. retry.Pool - watchdog

Calls

scripts/retry.waiting

This script checks for excessive retries due to, for instance, Enstore backlogs, and sends e-mail if some are found.

10. retry.P2P – plotter (not used on cdf)

Calls

scripts/retry.p2p

This script creates the "dCache P2P Queue" web page that is not linked by from the main FNDCA web page.

<http://fndca3a.fnal.gov/dcache/p2p.html>

11. retry.no-mover-found - watchdog, palliative

Calls

scripts/retry.NoMoverFound

This script checks for cases where no mover is found for a request (?), and sends e-mail if some are found.

12. kill_close_wait.sh - palliative

Calls

scripts/kill_close_wait

CLOSE_WAIT is a socket state usually associated with a socket that has not been properly closed per protocol.

This script cleans up unclosed sockets that can accumulate in a large dcache system, exhausting file descriptors. In principle, should not be needed with ideal dcache code or a low load dcache system (since a JVM does clean these up eventually).

2.2) Cronjobs

Categories: according to primary deliverable of script

Admin - manages logs, cleans up disk space, or makes back-ups

Plotter - creates a web page for viewing

Watchdog - send e-mail if a condition is found

Palliative - reduces impact of a weakness in dCache

1. crontab.enstore.fndca

1. make_queue_plots.sh - plotter

Runs dcache_make_queue_plot_page.py

This script takes the information gathered by an "at" job as input to formally create the web page:

http://fndca3a.fnal.gov/dcache/dc_queue_plots.html

2. make_login_plots.sh - plotter

Runs dcache_make_login_plot_page.py

This script takes the information gathered by an "at" job as input to formally create the web page:

http://fndca3a.fnal.gov/dcache/dc_login_plots.html

3. Billing.summary - plotter

This script creates the FNAL "Billing" web page, listing actions per day (distinct from DESY Billing web page).

<http://fndca3a.fnal.gov/dcache/billing.html>

4. repls start or repls kickoff - plotter

This script initiates the creation of a file listing for each pool. Some processing is done and the intermediate results are stored on each pool node.

4a. repls copy - plotter

This script gathers the results from each pool node into the master web listing found at:

<http://fndca3a.fnal.gov/dcache/files>

Note: we are accepting that sometimes the "repls start" may not be done when we do the copy hours later... very rare, but possible. The alternative would be to have authorized keys for each pool node entered on the head node which is fragile (nodes come/go) and unwieldy.

4b repls [no args] – started remotely by repls on the pool node worker part of the script running on pool node.

6a. lifetime start - plotter

This script initiates the creation of a file lifetime lists for each pool. Some processing is done and the intermediate results are stored on each pool node.

6b. lifetime copy - plotter

This script gathers the results from each pool node into and creates plots for the web page at:

http://fndca3a.fnal.gov/dcache/dc_lifetime_plots.html

Note: we are accepting that sometimes the "repls start" may not be done when we do the copy hours later... very rare, but possible. The alternative would be to have authorized keys for each pool node entered on the head node which is fragile (nodes come/go) and unwieldy.

7. lifetime [no args]

Worker script running on pool node and collecting information.

8. ftp_gather - plotter

This script gathers the FTP logs from door nodes to the admin node. These logs are used to create the "Recent FTP Transfers on fndca" web page using:

http://fndca3a.fnal.gov/cgi-bin/dcache_files.py

2. crontab.enstore.fndcam

1. pg_dumpall - admin

Back up the postgresql database on monitor node.

3. crontab.root.fndca

1. move-old-logs - admin

Clean up old FTP transfer and related logs to the ftp-tlog-old directory (where they will no longer be visible on the Recent FTP Transfers web page).

[TODO] There may be a weakness here. I recall not seeing new directories created in the "old" ftp log area to accommodate new users. The script assumed only files would be moved, not directory trees. This should be confirmed and fixed.

2. move-ftp-cert-logs - admin

Clean up old FTP transfer C=US items. I am not familiar with this with what these are though.

3. tmpwatch - admin

Clean up statistics and support file areas, but do not remove the doc stored in the statistics sub-directory.

4. check_port_block - watchdog

Check that the appropriate ipchains (personal firewall) blocks on the use of certain dcache ports from offsite are in place. Example: unsecured dcap ports are supposed to be block from use by nodes outside of fnal.gov.

4. crontab.root.fndca1

1. tmpwatch - admin

Clean up old FTP logs still on door node.

5. crontab.root.fndcam

1. real-encp-cleanup kickoff - admin

Initiate the clean up of old logs left by real-encp on each of the pool nodes.

2. check_crc kickoff - watchdog

Initiate the check of CRC of almost every file in cache on each of the pool nodes. Only files older than 12 hours in cache are checked.

[TODO] more detail on what is really compared in this.

2.3) DCache Call-outs

Enstore restores/stores:

real-encp

encp.options

2.4) Vladimir's monitoring plots

1. old-style tomcat-based

cdfen/web.xml

2. Lazlo-based

2.5) pagedcache

2.6) Operational Utilities:

poolcmd,

doorcommand

pathfinder

2.7) Dmitry's PNFS consistency incremental scan

http://www-stken.fnal.gov/enstore/dcache_monitor

2.8) Miscellaneous utilities:

dropit (copied from a FNAL FUE product)

3) Supporting Infrastructure not covered elsewhere

3.1) authorized_keys

3.2) kerberos service credentials

3.3) /etc/grid-security

3.4) CVS: HPPC

Configuration

As it was implemented by Dmitry scripts to generate fermi-type tweaking and configuration settings are located at

dcache-fermi-config/cdftst/etc

this directory is used for common scripts to generate with rpms, and also holds file templates. System dependant settings are set through "case" statements in scripts.

Some fundamental scripts:

make-dcache-admin – generate dcache-admin rpm. Script declares what files, and where from will be copied to rpm (from common cdftst or system specific dir). This includes installation scripts, configuration files, template files and also some .batch files (don't expect them to be from original distro from dcache.org)

At present, only few files come from system specific dir like cdfen :

*.farmlet, PoolManager.conf , pool_files.config, dcache.kpwd, authorized_keys, few batch files.

dcache-admin.spec - .spec file for rpm generation, defines which files are installed on target system.

make-dcache-pool – similar, for generate dcache-pool rpm

dcache-pool.spec – similar to dcache-admin.spec

dcache-common-install – common set of shell functions to be called during installation in dcache-{admin,pool}-install

dcache-{admin,pool}-install – where most of configuration tuning defined. During rpm install the "uname -n" called to set node name. Based on the node name, the type of the

configuration is identified through “case \$node” statements (along with few other variables, to set node type or type of system – test or product).

Based on the system type, node type, etc. the vanilla dcache installation configuration files (*Setup*, *.batch, etc) are tweaked. The setup is not generic, and lengthy parts of the tweaking code present in case statement. It mostly work for repeated configuration generation on the same set of nodes. The creation on new configuration on new set of nodes with different distribution of services among nodes will require careful revision of “case <nodeType>” fragments.

make_links – executed during rpm installation, create links, mostly from /etc/d-cache/... to dcache-fermi-config/<some-script> .

make_pool_files - on the pool node, generate pool specific files (setup, .poollist) on the base system specific file pool_files.config

dcache.local* - files to be installed in /opt/d-cache/jobs to the to the local settings during jmv startup.

Appendix A.

Brief Summary of the dCache Monitoring Scripts

Script type legend :

p - plotter

w -watchdog

c - compensation (palliative)

a - admin

- - not used on cdf

Scripts reside at ~enstore/dcache-deploy/dcache-fermi-config/

Started by monitoring-boot as "at" jobs:

root:

w enabled.list scripts/check_poolenabled.sh

enstore :

p login.list scripts/listioalldoors.sh

login.plot scripts/logins.sh

pw pool.stats status/updatePoolStatus.sh

pw status/updateDirectory.sh

w postgres.list check_postgres

p queue.plot scripts/queues.sh

p restore.list scripts/kill_restore_butincache.sh

w retry.Pool scripts/retry.waiting

-p moverls.list scripts/moverls.sh

-p retry.P2P scripts/retry.p2

-wc retry.no-mover-found scripts/retry.NoMoverFound

-c kill_close_wait.sh scripts/kill_close_wait

cronjobs:

HEADNODE

ROOT

crontab.root.fndca

a move-old-logs

ENSTORE

crontab.enstore.fndca

p make_queue_plots.sh dcache_make_queue_plot_page.py

p make_login_plots.sh dcache_make_login_plot_page.py

p Billing.summary

p ftp_gather

Scripts call itself on pool nodes for
distributed execution on pools

p scripts/repls (kickoff/start, copy) repls

p lifetime lifetime

MONITORING NODE

ROOT

crontab.root.fndcam

a real-encp-cleanup kickoff

a check_crc kickoff - watchdog

ENSTORE

crontab.enstore.fndcam

a pg_dumpall
a move-ftp-cert-logs
a tmpwatch
w check_port_block

DOOR NODES

ROOT

crontab.root.fndca1

a tmpwatch

dcache call-outs :

POOL NODES

real-encp

encp-options

Appendix B.

Scripts accessing dcache SLM port

dcache.sshSetup – must be sourced. Defines multiple functions to be called, including functions accessing sshPort and admin Port, e.g. dcache() and adcache(). Used mostly by cms. The common script sourcing this file is “check_port_block”

scripts/doorcommand

scripts/killold.sh

scripts/listidoors.sh

scripts/ping_all_cells

scripts/ping.slow

scripts/poolcmd

scripts/queues.sh

scripts/repls

scripts/retry.waiting

scripts/sweeperFree.sh