Optimization of dCache

1. Introduction

The aim of this exercise is to test your ideas on how performance of your storage systems can be improved. Usually, it is not possible to test your every idea on a production systems. Also, use cases are different(e.g. files sizes) and estimation of an impact of a parameter change may not be obvious. This exercise is not aimed to test any possible configuration but just some of them.

What we provide here is a toy model of a grid site with a storage element already populated with data and a computing element with worker nodes assigned to it. And the whole idea is the foolowing:

* you submit **parallel** jobs which read data from the SE(extreme conditions)
* what you get in return is MB/s from dCache instance
* the higher the value, the better your configuration is

The tool is available at https://twiki.cern.ch/twiki/bin/view/Atlas/DCacheLoadTest

Just to remind you that in a real situation before you go to optimize the software, you have to optimize your hardware: RAIDs, network configuration,etc. Try bonnie++ and WireShark instead.

2. Requirements

Everything is installed(SEs, CE with a batch system) and the only thing you need is just ideas. There is one thing, however, which is optional. The tool which performs the measurement also produces plots(ROOT, MB/s vs number of jobs). These plots are very illustrative but they require an installed X server on your laptop. Or just scp.

3. How to submit jobs. Step by step.

All you were given sheets with

Job submission:
`atlas000@grid-ce.physik.uni-wuppertal.de`

So,

```
ssh -X atlas000@grid-ce.physik.uni-wuppertal.de
cd $USER
ls
```

`run_dctest.sh` -- the main script which launches parallel jobs. executed on grid-ce .
`dc_test.py` -- python script which runs dc_read(== dccp read from SE). WN.
`series_anaysis.py` -- python analysis script. executed on grid-ce .
`file.list_SFSWN50_100M` -- list of 100 MB files on the pool sfswn050_big.

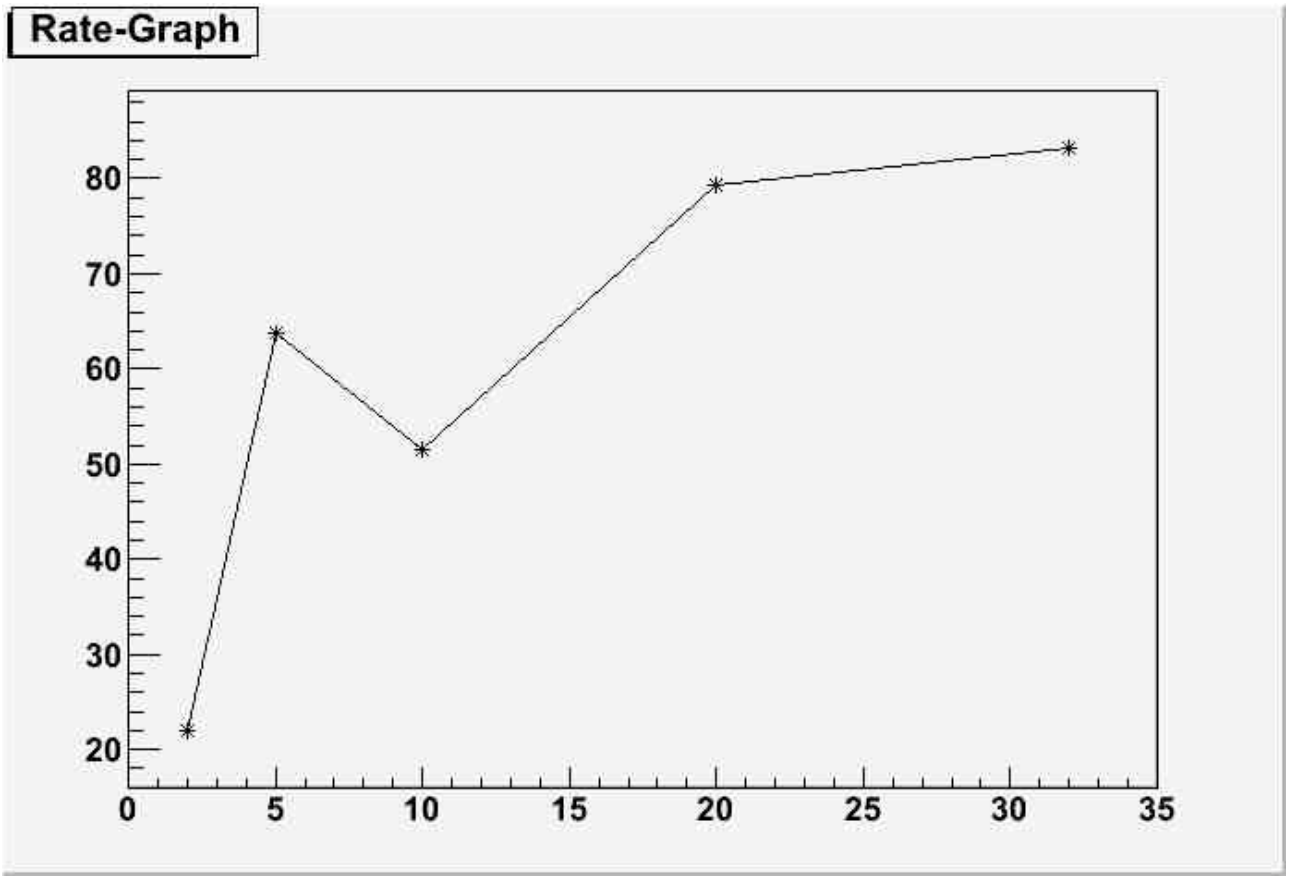
4. ./run_dctest.sh

There are three parameters in this file to be changed:

a. testname=Test-VMWARE_SFS050-1 -- you will run many tests. For every submission of sequence of bunches of jobs, there will be a directory created with the this name. If there is already such directory, **it will be removed and recreated**. There will be a lot of bookkeeping work. So, choose the name wisely for every test.
b. testseq="2 5 10 15" -- as the name suggests, the script will submit first 2 parallel jobs, then after the first 2 will be finished, 5 parallel jobs, 10 and, finally 15.
c. FILELIST=$DCTESTDIR/file.list_SFSWN50_100M -- this is the list of files which will be read from the dCache. The files which will actually be read on worker nodes are chosen randomly from this list. You can make a mixture of files by combining different lists of different pools. If the files are small enough, then they may be cached at the system level. On the other hand, if the files are bigger then the time to establish a connection is saved. This file lists represents different use cases. A vast area for playground. The files in file.list* refer to the host sfswn050 which does not exist. You should convert them with

```
cat file.list_XXXXX | sed 's/050.p/(*your SE*).p/'
cat file.list_XXXXX | sed 's/050.p/075.p/'
```

Launch run_dctest.sh and it will show you a plot:



It shows MB/s vs number of jobs. The higher it is, the better.

If do not have an X server, then grep for the lines in your $testname.log:

```
### Peformance 2 12 17 0.682494 1.364989
```

the first number is the number of jobs(2 here), the last parameter is the rate you are looking for(1.364989 MB/s here).

5. dCache tuning

You are the kings. Change any parameter you like and see the effect. Some ideas:

a. Study systematics.
b. Test different file.lists_s. Generate your own file.list_XXXXXXX with files on different pools.
c. Replication(see talk by Christopher).
d. Number of active movers(see your instance's webpage or GUI).
e. Transfer parameters
f. OS level(e.g. readahead)
g. Postgres(memory: may be related to kernel parameters, query tuning). Also see http://trac.dcache.org/projects/dcache/wiki/HandsOnPostgreSQL . Postgres is restarted with
`/etc/init.d/postgresql restart`

6. **Things to remember**

a. Needless to say, if you change a relevant parameter in dCache configuration, you have to restart it. It will be necessary sometimes even to run `install/install.sh` .
b. The batch queue has about 400 slots for jobs. Do not submit more than 30 jobs at a time: testseq<"30".
c. The storage elements are virtual machines but CE and WNs are real. What you learn here about the performance may have a negative impact in the sense that your local site performance may look very different(e.g. hardware).
d. There are backup copies of the virtual machines.
e. We are working in groups of two for one dCache instance. All your job submissions are logged and all the instances are equal. The group with the highest transfer rate for **15** jobs will get a prize.

APPENDIX

Short introduction to root:

```
root file.root
new TBrowser();
```

click in the left panel "`ROOT Files`", then click file.root, then click "`Rate-Graph; 1`".

This will show the plot.