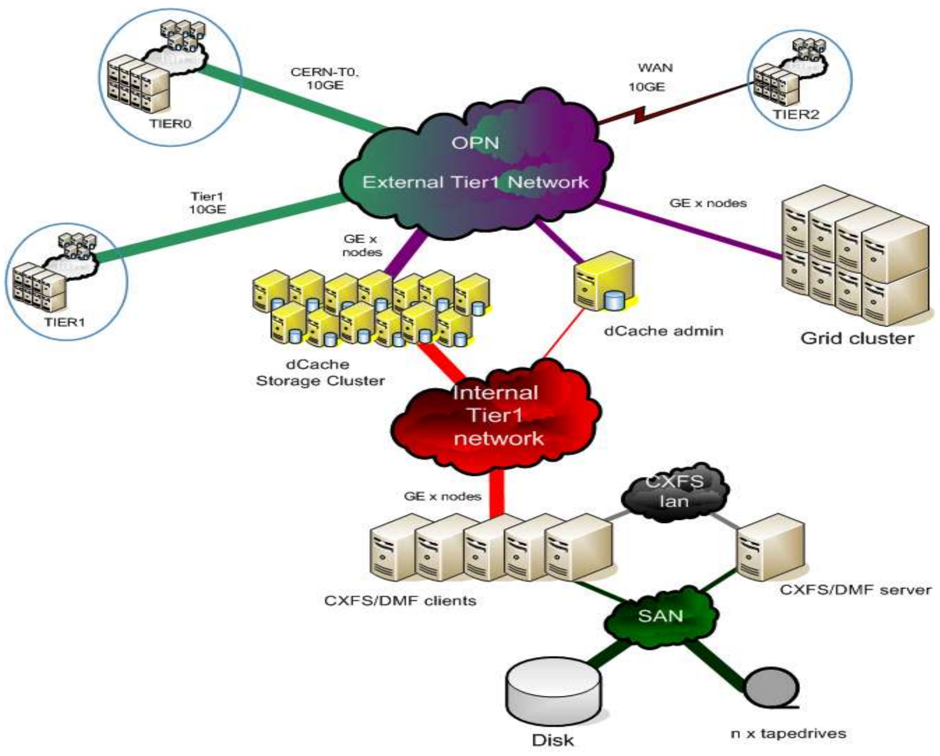
SARAdCachesetup

for the Tier II dCache workshop, June 2006
by Ron Trompert, SARA

1. General setup



2. Hardware

- Pool nodes
  - 5x dual Opteron's, 4GB memory, 2x 1GE
  - 2.2TB disk cache (1.7 TB RAID6),XFS, 12x 250GB SATA, 3ware RAID controller
- Admin node
  - dual Xeon, 4GB memory, 2x 73GB internal disk, 2x 1GE
- MSS gateway nodes (CXFS/DMF clients)
  - 2x dual Xeon, 4GB memory, 2x 73GB internal disk, 2x 1GE, dual HBA FC, 1.6 TB CXFS file-system (SAN shared file-system)
  - runs CXFS client, read/write data directly to/from CXFS file-system
  - and rfio daemon to put/get data to/from pool nodes
- MSS server (CXFS/DMF)
  - 16 cpu R16K MIPS, 16GB memory, 1.6 TB CXFS file-system (SAN shared file-system), 6x STK 9940B tape drives
  - CXFS Metadata server, regulates access to CXFS file-system

- DMF (Data Migration Facility = HSM system), migrates data from disk to tape and back

Currently, we have 5 pool nodes that is going to be increased with another 10 nodes with 7 TB SATA storage each (NAS) soon. The pool nodes run SL4.2 with the 2.6 kernel that comes with it.

3. Setup

On the dCache admin node runs a PNFS server, a gridftp door, a srm door and a gsidcap door. Due to the fact that we are going to scale up our infrastructure we will run the dCache admin software and the PNFS server on one node and the srm,gsidcap and gridftp door on another node.

All pool nodes run a gridftp door. During the throughput tests we found that doing this, the throughput rate scales linearly with the number of pool nodes.

4. Operation

Incoming data is written to the pool nodes. Every 60 seconds precious files are flushed to the CXFS/DMF client nodes using the hook in dCache that allows you to copy data to your mass storage system. These client nodes share a cxfs-based file system. On these nodes runs a DMF client that migrates data to and from tape. Data is copied from/to the pool nodes to/from these nodes using RFIO.

DMF schedules tape accesses so that the available amount of tape drives is optimally used. Therefore, the script that copies the data is very simple and only does the rfcp and writes storage info into pnfs. The cxfs/dmf client node to copy the data to is selected by the script based on load information, number of rfiod daemons currently running and heath and status info like, “is the cxfs file-system mounted?”.
After the copy, the file is fsync-ed by the script on the cxfs client to be sure that the file is physically on disk and then the script instructs DMF to write the file to tape (dmput). DMF normally works with thresholds to write files to tape but telling DMF to write a file to tape as early as possible optimizes the rate of writing to tape. When you do a dmput DMF does not start writing to tape right away but it collects these requests until it is worthwhile to mount a tape and write the data to tape. In this setup we have been able to get a rate that was very close to the maximum rate possible. We could get a rate of more than 28 MB/s per 9940B drive where 30 MB/s is the maximum. In order not to overload the pool node we limit the number of stores per pool to two which we found to be a good value.

To optimize read accesses from tape we run a cron job every minute that prestages data. It runs on de dCache admin node and does a dmget (instructs

DMF to get a file from tape) of every file of the restores that are in the queue. So while these restores are waiting to be executed, DMF is already fetching the data from tape. This means that when the restore is executed the data may already be (or will soon be) on disk of the CXFS/DMF clients. We have also limited the amount of restore at each HSM pool to two. We haven't done any throughput tests reading from tape yet so I cannot present any numbers.

5. Network

The pool nodes are connected with the cxfs/dmf clients through an internal network. No filtering takes place here. The pool nodes and the node with an srm door are part of a subnet with a dedicated 10 Gb link to CERN. On the dedicated link with CERN there is no firewall. On this subnet there is also a router that takes care of the connectivity with the outside world. On this router there are ACL's to filter the network traffic. We still need to see the impact of that for the performance. The CXFS/DMF client nodes are connected to the CXFS/DMF server with an other internal network on which no filtering takes place and of course they are connected through the SAN.

The dache nodes communicate with each other over the external subnet. The reason for this is that if congestion takes place here. It will probably lead to problems like transfers into or out of the dCache being slow or failing but not to problems in our dCache internally, like pools filling up with precious data because the flushing to the cxfs clients is not fast enough.

All nodes involved have 1 GigE connectivity to the networks.

6. dCache Configuration

We have switched off the space costs because taking only cpu costs into account during continuous heavy inbound network traffic gives a better load balancing over the pools.

The number of I/O movers on each pool was set to a rather low value like 4-6. Using the default of 100 crashes the pool node when a significant number of gridftp transfers are running. The number of restores and stores is set to two for each pool. Two rfcp's gives a rate of approximately 100 MB/s which is about the maximum for a 1 GiGE link. Of course values like this depend on the number of pools on each pool node and how heavy they are used.

We have typically 6 pools per pool node. 2 dedicated pools for CERN-SARA disk-disk and disk-tape throughput tests. One generic pool for all VOs to use. This pool has a hsm backend. This pool is used far more than the others. On top of this there are 3 disk-only pools for ATLAS,ALICE and LHCB. In the future we intend to reduce the number of pools on a pool nodes because we anticipate that other pools will be used more heavily and we want to keep the load on the pool nodes under control.

In normal operation, the generic pool is used most heavily but I expect that in the future the use of the disk-only pools will increase. We dedicate the disk-only pools to specific VOs in order to impose “disk quota”. In the future we intend to do the same with the HSM pools. The advantage of this approach is that it is then easy to show how much disk space is used by what VO without having to scan the pnfs directories or the pools. In addition, if, for example, an LHCb user writes a lot of data then as a result cached LHCb files will be removed. Hence, the VOs will not get in each others way.

At the moment we do not (yet) distinguish between gridftp pools and local analysis read pools. The read, write, cache preference for all our pools is the same. So far we haven’t encounter any problems with this setup yet but maybe we will have a different opinion in the future. Our dCache pools are all NAS boxes, for now, we have no plans to use worker nodes as read/cache pools. The reason for this is that we use the second disk on the worker nodes to host home directories for single CPU jobs.

We have a postgres-based pnfs that resides on a RAID1 system. In addition, this is backed up every day. But in the near future we will move to a slony-based setup where the databases will be on a SAN-RAID infrastructure.

7. Problems

All in all we did not have that many problems. Maybe, the biggest problem is the learning curve of getting to use dCache. The documentation has greatly improved with “The Book” which is a great help.

In conclusion, we are very happy with our choice of hardware and software, dual opterons/2.6 kernel/dCache.