LHC Data Analysis Using NFSv4.1 (pNFS): A Detailed Evaluation

Journal of Physics: Conference Series **331** (2011) 052010 doi:10.1088/1742-6596/331/5/052010

LHC Data Analysis Using NFSv4.1 (pNFS): A Detailed Evaluation.

Johannes Elmsheuser[1], Patrick Fuhrmann[2], Yves Kemp[3], Tigran Mkrtchyan[2], Dmitry Ozerov[3], Hartmut Stadie[4]

[1]LMU Munich, Am Coulombwall 1, D-85748 Garching
[2]dcache.org, Notkestrasse 85, D-22607 Hamburg
[3]DESY, Notkestrasse 85, D-22607 Hamburg
[4]Universität Hamburg, Luruper Chaussee 149, D-22761 Hamburg

E-mail: yves.kemp@desy.de

Abstract. NFSv4.1 (pnfs) is a new industry standard which allows for access to data distributed on different file servers. The dCache storage system has implemented this protocol. The aim of this paper is to evaluate this implementation and compare it to traditional HEP protocols like dCap. We perform synthetic and simple ROOT tests as well as real analysis tests using the ATLAS HammerCloud framework and an analysis use case from CMS. The tests are done on a cluster comparable to a small analysis cluster.
Our tests have shown that the dCache NFSv4.1 (pnfs) implementation is very stable and that analysis speed is at least comparable if not superior to dCap in general. In addition, our test have shown that no change to analysis were needed to make use of the new protocol.

1. Introduction to NFSv4.1 (pNFS) and dCache

The network file system NFS is well established and well known. However, even until version 4, it relied on one single data server as being the source of the data. It is only with the protocol specification NFSv4.1 and the pNFS protocol extension that this limitation is overcome[1]. This new protocol is implemented by many vendors in their server products, and the vendors also take care of providing clients for a diversity of operating systems. Today, clients for Linux and Windows e.g. are available. The Linux client is even part of the kernel.
Distributing the data over many server and making a distinction between metadata and data access is one of the fundamental concepts of dCache[2]. It was therefore natural for the dCache developers to add the NFSv4.1 protocol to the list of supported protocols. Since version 1.9.3, NFSv4.1 is part of the dCache standard distributions and has matured over time up to a level where one can do a large scale evaluation of stability and performance using LHC data analysis.

2. Description of the test setup

In order to perform realistic LHC data analysis, a separate cluster has been set up. In the following the different components will be explained in more detail.

2.1. File server and dCache setup

The dCache system is formed by one head node (2xIntel Xeon 5160 CPU with 8 GB RAM) attached with Gbit to the central Force 10 switch and five pool nodes (Dell R510 with 2xIntel

Xeon E5520 CPU, 12 GB RAM, 12x2 TB SATA disks with 10 disks forming a RAID-6 dCache pool area with each 16 TB net capacity) attached with 10 GE to Arista Switches. Head node and pool nodes all run SL 5 with the most recent kernel provided by SL. No special patches or tunings needed to be applied to the kernel as all dCache and NFS processes are Java processes running in user space.
The dCache version used for the final setup is 1.9.10-2. The head node also played the role of dCache NFS and dcap doors whereas the pool nodes only serve as pools. No special tuning was applied to the dCache setup. The Storage Element endpoint is not published and not known to the experiment's framework.
The performance of different subsystems was evaluated before performing any dCache installation. The file system holding the dCache data (XFS with default setting on RAID-6 described above) was able to deliver an aggregated data rate of around 300 MByte/s when reading 100 files of 1 GByte each stored locally in parallel using /bin/cat to local /dev/null. A maximum aggregated data rate of around 500 MByte/s was observed when only eight files were read in parallel.
Using iperf, the point-to-point bandwidth between two file server was measured to be 5.6 Gbit/s.

2.2. Worker nodes and batch/Computing-Element (CE) setup

The clients reading from dCache are standard DESY-HH Grid WorkerNode (WN) that were taken out of the production batch environment and put into a newly created CREAM CE and Maui/Torque batch system. 32 WNs with each 2xIntel Xeon E5420 (8 cores in total) and 16 GB RAM are installed with SL 5.3 64bit and glite-WN 3.2.7-0 version. The only special setup made was to exchange the kernel and nfs-utils. The current one is a custom patched kernel 2.6.36-rc3.pnfs.3 available for download from dcache.org[5].
The CREAM CE is not published through standard Grid mechanisms as we do not want to attract normal user work load. Instead, the endpoint had to be manually entered into the test submission framework used by ATLAS and CMS. Besides this, the CE and WN looked perfectly identical to the standard DESY-HH production Grid - with the noticeable exception that dCache was mounted and accessible via "/pnfs/desy.de/...".

2.3. The network

The central back bone of the DESY-HH Grid network is a large Force10 switch. All WNs, the head node and the CREAM CE are attached via 1 Gbit to it. The dCache pool nodes are connected via 10 GE to two Arista switches. These are in turn connected to the central Force10 switch using two 10 GE uplinks each.

2.4. Theoretical performance expectations

A maximum of 32 Gbit aggregated performance cannot be exceeded, this limit is induced by the worker nodes. However, it is unlikely that this limit can be reached. The RAID arrays of the file server have been measured to provide 550 MByte/s with a few streaming reads, whereas their performance degrades to 300 MByte/s when accessed by 100 concurrent streaming reads. The random read performance depends on the exact pattern and can reach values well below these numbers. Therefore, an aggregated bandwidth of 1.5 Gbyte/s to 2.5 GByte/s is unlikely to be exceeded when reading from disks. We have tested scenarios where data was read from the file server cache. In these cases, the local network connectivity will be limiting: Using iperf, the network card was measured to deliver around 5.6 Gbit/s. The maximum aggregated bandwidth is around 28 GBit/s. In all cases, the two layered network setup is not expected to be a bottleneck.
It is easy to see that the storage system is underpowered compared to the worker nodes. This is done on purpose to uncover limitations of our dCache setup.

3. Synthetic tests with dCache

As a first test, files with one GByte of random content are copied to the dCache system such that each pool contains 4096 files. This way one can guarantee that during one test run a file is only read once, so no caching on the file server or on the worker node can occur. Between test runs all file system caches are flushed. These files are then read using "/bin/cat /pnfs/the/file > /dev/null" and "dccp /pnfs/the/file /dev/null" and the total elapsed time is recorded. Different numbers of readers run in parallel to test the scaling. Each reader processes eight different files, of which only the timing information of the four central ones are taken into account. This excludes possible artifacts from slightly asynchronous job starts and tails occurring from one busy pool node. The overall throughput is computed as a function of the numbers of concurrent reads.

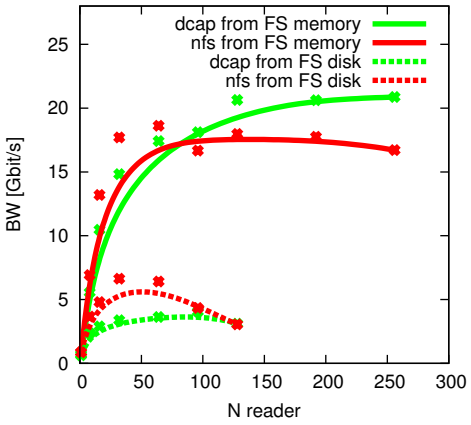

Figure 1. Results of the simple file reading.

A first run has shown effects of the disks system in the file server. The throughput is smaller than expected in the theoretical expectations. The reason is under investigation by experts. A second run was performed using a special setup in which the files are kept in memory on the file servers, bypassing the disks. The performance is shown in figure 1.

3.1. Remarks on stability and WAN performance

The stability of the dCache system and the clients cannot be expressed by a quantitative measure, but we can make same qualitative statements.

- Populating the dCache system with data worked without a server or client hick-up.
- Wide-area-writes were performed, 13 TB of data were written over 10 days with 100 GByte average file size, no crash observer.
- Untaring the Linux kernel into the NFS-mounted dCache concurrently from different clients did not lead to any server or client interruptions.
- Over a DSL line we could mount the dCache and perform a recursive file listing. As expected, this was orders of magnitude slower than local access, but no client or server problem was observer.
- 128 clients were writing into the same file. This lead to stuck clients, without however a dCache server interruption.
- On very rare occasions, some clients got stuck during normal reading and needed to be rebooted. These errors were however not reproducible, and did not happen with later versions of the Kernel. dCache server operation was not affected.

In general, the dCache server proved very stable during the testing period. On rare occasions, clients got stuck during normal operation. Whenever possible, the kernel developers were provided with debug data to improve stability.

4. ROOT reads

The ROOT framework[3] is at the base of most current HEP applications. Therefore understanding the read performance of ROOT files is crucial and needs to be investigated. The ROOT team added many new features in newer releases, among others the TTreeCache and an optimized basket structure. For this reason, we used ROOT 5.27.06, the most recent

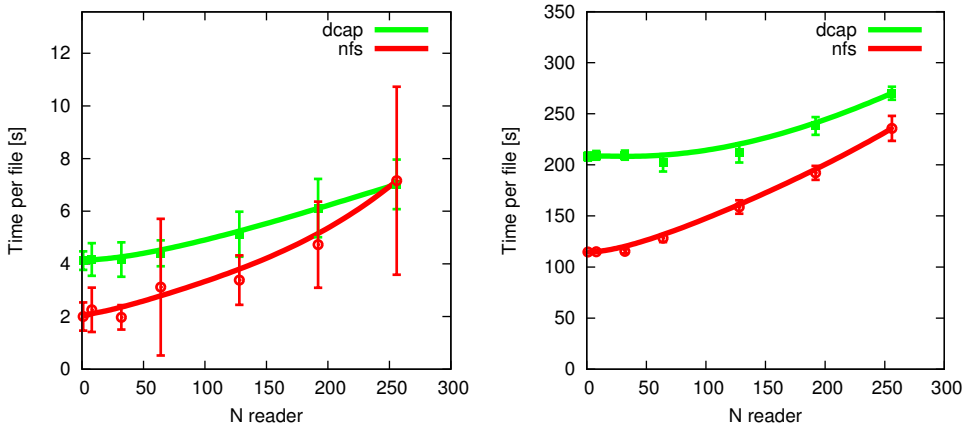

Figure 2. ROOT tests: Left: Optimized files, no TTreeCache, two branches. Right: Original file, 60 MB TTreeCache, all branches. (Accuracy of measurement is 1 second, resulting in larger error bars when transfers are faster like in the left plot.)

version available during the test period. The ROOT team provided us with two example files: An ATLAS file with the original, unoptimized basket structure(`AOD.067184.big.pool_4.root`), and another ATLAS file with reorganized baskets (`atlasFlushed.root`). We were also provided with a ROOT macro, taodr.C. This macro was slightly modify by us to allow easy change of reading parameters: a) File type. b) dcap or NFS. c) read all branches or only two branches. d) A TTreeCache of 60 MByte, or no TTreeCache. e) Reading 1, 8, 16, 32, 64, 128, 192 or 256 files in parallel.

The general picture found was that reading via NFS is faster than via dCap. Depending on the exact reading parameters, the differences vary. Two exemplary figures are shown in 2. Investigations have shown that this is due to use of the file system cache on the client side. To quantify this effect, table 1 shows for each category the ratio of read size over file size. It is clear that without TTreeCache buffer, this ratio is higher than with TTreeCache enabled. However, even with TTreeCache enabled, reading via NFS can profit to large extents from the client file system cache. We have not tested the XROOTD protocol, for a detailed investigation, the reader should refer to [4].

type	branches	NFS	dCap	NFS +TreeCache	dCap +TreeCache
optimized	two	0.07	0.08	0.06	0.03
optimized	all	1.0	11.5	1.0	1.0
original	two	0.04	0.52	0.05	0.03
original	all	0.98	52.4	0.98	0.92

Table 1. Ratio Transferred size over File Size (source dCache billing log).

5. ATLAS HammerCloud tests

HammerCloud was developed by ATLAS as a distributed analysis stress testing system[7]. It is used to deliver large numbers of real analysis jobs to Grid sites to stress test their storage and network setup.

Small modifications needed to be done in order to enable the submission of HammerCloud jobs to our test environment: Usually, the HammerCloud system can only send jobs to official Grid sites

registered in the Grid Information System. This was not the case for our computing element, the CE URL needed to be defined manually. As for the storage element, it also is not registered in the Grid Information System, hence the test data is not contained in the usual ATLAS data management system DQ2. The HammerCloud test had to be modified to allow for a custom list of files. No modification was needed in the analysis code itself for the different protocols used.

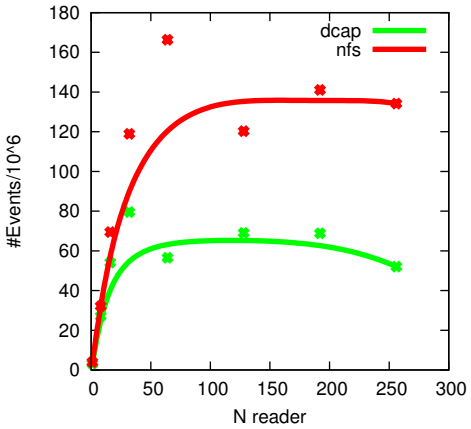

Figure 3. Results of ATLAS HammerCloud.

The analyzed data consisted of official ATLAS MC samples. The events were simulated 7 TeV events, preferably no minimum bias events. The data was reconstructed with ATHENA version 15.6.8 in the AOD format. The total amount of data consisted of 33 TByte. The analysis itself was performed using ATHENA version 15.6.6 and 16.0.2.3, respectively. The ROOT version used are 5.22/00h (15.6.6) and 5.26/00e (16.0.2.3). The analysis is a standard AOD one reading trigger variables and muon variables.
Different configurations have been benchmarked on the test system: the number of concurrent readers was varied from one to 256 and with either reading from dCap or via the NFS 4.1 mounted dCache system. Each configuration was tested during 24 hours, allowing for the system to be fully loaded over a long period. Figure 3 show the number of events processed by jobs finished within the 24 hour period. One can see that up to around 32 concurrent processes, the scaling behavior of both reading methods is good, with a slight advantage for the NFSv4.1 reading. Above 64 concurrent readers, one can observe a break in the performance which affects both protocols, again with the NFSv4.1 performing better than the dCap protocol. This is an effect of our undersized storage, which puts too much stress on the pools when too many clients connect to them. We attribute the performance difference between the two protocols to client caching.

6. CMS analysis tests

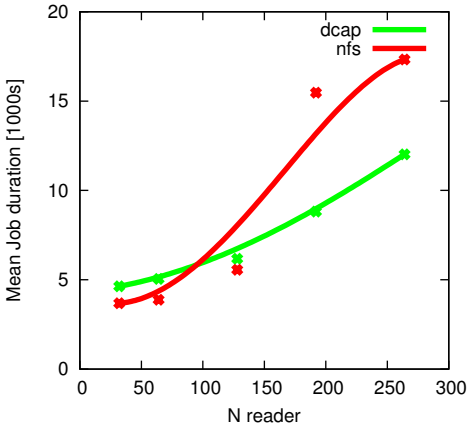

Figure 4. Results of CMS analysis test.

For performing the tests, no official CMS benchmark tool was used. The CMS group of the University of Hamburg has instrumented one particular analysis to measure the performance of the production dCache at DESY, they have kindly agreed to modify their tools to work in our test environment.
The job submission to the test computing element was done using grid-control[6], a job submission engine developed at the University of Karlsruhe. Support for non-official CE is very easy in this submission engine. CMS uses a so-called "Trivial File Catalog" which maps the logical file name to the physical file name taking into account the access protocol. The CMSSW version used for submission was slightly modified to accept a custom-made TFC. This way, switching between dCap and NFSv4.1 in our tests was rather easy. No modification needed to be done to the analysis code itself.
The analysis was using a dataset of 1.7 TByte in 308 files

in the RECO format. Using CMSSW 3.6.2, the data is read and PAT Ntuples are produced [8]. This is rather I/O intensive and is a typical use-case on the DESY National Analysis Facility.
As one can see in plot 4, the situation with up to 128 concurrent jobs is rather similar to the ATLAS situation: Jobs reading via NFS are about 20% faster than those reading via dCap. Above 128 concurrent jobs, the dCap performance gets slightly worse while the NFS performance visibly degrades. As we did expect a degradation due to the undersized storage, this behavior still needs investigation.
During the test, the benefit of the file system cache was clearly visible: According to the dCache billing logs, the dCap jobs transferred 2.5 times the data volume of the NFS jobs for the same physics content.

7. Summary and outlook

We have tested the dCache server implementation of the NFSv4.1 (pnfs) protocol against the dCap protocol using LHC analysis code. Also synthetic and stability tests were performed. We see a very good stability of the dCache server and an overall comparable if not superior performance of NFSv4.1 (pnfs). We clearly see effects and benefits from the client caching of the Linux kernel.
NFSv4.1 (pnfs) is not part of the SL5 distribution, and only has become part of the vanilla kernel recently. We were however provided with patched kernel versions which could be run on a standard SL5 installation with gLite middleware and the experiments' software stack. Also this client proved remarkably stable, with only a very few hick-ups.

The LHC experiments are in the process of migrating to newer ROOT versions with improvements in the field of data access like TTreeCache. We will follow these developments.
This paper on purpose concentrated on the performance aspects of the new protocol and the dCache implementation. Future tests must also include security aspects.
The NFSv4.1 (pnfs) patches have become part of the vanilla kernel beginning of 2011. It is probable that these patches will be backported to become part of one of the next minor releases of RHEL 6 and its derivatives. If the LHC experiments stay with RHEL 5 or similar for a longer period, a community effort to support a NFSv4.1 (pnfs) enabled kernel might be envisaged.

8. References

[1] Network File System Version 4 Minor Version 1 Protocol, RFC 5661, http://tools.ietf.org/html/rfc5661
[2] http://www.dcache.org/
[3] ROOT – A C++ framework for petabyte data storage, statistical analysis and visualization, I. Antcheva et. al., Comput. Phys. Commun. 180 , 12 (2009) 2499-2512
[4] Xrootd in dCache - design and experiences, G. Behrmann et. al., this proceedings
[5] YUM repository with kernel used for testing: http://dcache-www01.desy.de/yum/nfsv4.1/el5/nfsv41.repo
[6] https://ekptrac.physik.uni-karlsruhe.de/trac/grid-control.
[7] HammerCloud: A stress testing system for distributed analysis, D. van der Ster et. al., this proceedings
[8] PAT: Tools for Physics Analysis at CMS;Hinzmann, Andreas, this proceedings