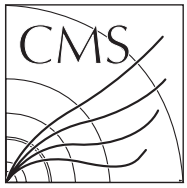
Available on CMS information server

CMS CR -2009/097



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

14 May 2009

Utilizing Lustre filesystem with dCache for CMS analysis

Y Wu, B Kim, J Rodriguez, Y Fu, D Bourilkov and P Avery

Abstract

This paper presents our new experimentations to utilize Lustre filesystem for CMS analysis with direct POSIX file access while keeping dCache as the frontend for data distribution and management. We describe our implementations that integrate dCache with Lustre filesystem and how to enable user data access without going through the dCache file read protocol. Our initial CMS analysis job measurement and transfer performance results are shown and the advantages of difference implementations are briefly discussed.

Presented at *17th International Conference on Computing in High Energy and Nuclear Physics,21 - 27 March 2009,Prague,Czech Republic,15/05/2009*

Utilizing Lustre filesystem with dCache for CMS analysis

Y Wu[1], **B Kim**[1], **J Rodriguez**[2], **Y Fu**[1], **D Bourilkov**[1] **and P Avery**[1]
[1]Department of Physics, University of Florida, Gainesville, FL 32611, USA
[2]Department of Physics, Florida International University, Miami, FL 33199, USA

Abstract. This paper presents our new experimentations to utilize Lustre filesystem for CMS analysis with direct POSIX file access while keeping dCache as the frontend for data distribution and management. We describe our implementations that integrate dCache with Lustre filesystem and how to enable user data access without going through the dCache file read protocol. Our initial CMS analysis job measurement and transfer performance results are shown and the advantages of difference implementations are briefly discussed.

1. Introduction

The CMS experiment is expected to produce a few Peta Bytes of data per year and distribute them globally. Within the CMS computing infrastructure, most user analyses and Monte Carlo event production will be carried out at some 50 CMS Tier-2 sites [1]. How to store the data and allow efficient access by physicists has been a challenge, especially for Tier-2 sites with limited storage resources. The CMS experiment, including other LHC experiments, has been successfully using dCache [2] for managing and distributing large amount of data. CMS users generally access the data stored in dCache through dCache dcap protocol. However, it is observed that file access is relatively slow through the dCache dcap protocol, especially when large numbers of simultaneous jobs are trying to access the data at the same time.

Lustre filesytem [3], a cluster filesystem developed by Sun, has been used among various high performance computing systems in many government laboratories, universities and corporations, like Lawrence Livermore National Laboratory, Oak Ridge National Laboratory, Indiana University, and Texas Advanced Computing Centre (TACC), etc due to its high performance IO feature. In fact, Lustre file system was said to manage data on more than 50 percent of the Top 50 Supercomputers by November 2008 according to Sun [4]. The HPC Centre at the University of Florida (UF) has been using Lustre since 2007. It now has over 81TB disk storage managed by Lustre filesystem. UF CMS Tier-2 site has been collaborating with UF HPC Centre since the very beginning and is a shareholder of UF HPC Centre. It is nature for us to explore the possibility to utilize the large amount of storage available in UF HPC Centre, and try to look at the IO access performance when running CMS applications on it.

In this paper, we describe our experimentations to utilize dCache as the frontend for data management and distribution and Lustre filesystem as the backend to provide users with the direct POSIX (portable operating system interface) file access without going through dCache file read protocol for fast IO access performance. Utilizing our new implementations, we are able to store files into Lustre filesystem and retrieve them through the existing dCache interface using the well defined OSG/EGEE

client tools. In the meantime, we are able to run user jobs more efficiently by directly accessing data through mounted Lustre filesystem.

2. Implementations

We tried to find the optimum way to keep the existing data transfer and management functionalities for dCache and allow the users to access the data without going through the dcap doors in the meantime. To do so, we experimented with two possible implementations while trying to fully utilize the existing UF CMS Tier-2 and UF HPC Centre infrastructures. One implementation is to directly use mounted Lustre filesytem as dCache pool storage; another one is to use dCache configurable HSM (hierarchical storage management) interface to tertiary storage system with customized dCache/Lustre storage handling scripts for storing and retrieving files to/from Lustre filesystem.

2.1. Using Lustre filesystem directly as dCache pool

The implementation to directly use Lustre filesystem is pretty straightforward. So we only briefly describe it here. Using the standard dCache installation procedure, we created 12 dCache pools on a server node at UF CMS Tier-2 where each pool has 1TB of disk space. The <*poolDirectory*> is set in the dCache configuration to point to the mounted Lustre path. We then created two cronjobs: one makes a soft link for each pnfsid file under the Lustre pools with their original user defined filename in the same Lustre filesystem, the entry was also put in a postgres database (not related to the existing dCache pnfs postgres database) for tracking; another cronjob is used to clean up the soft links when the corresponding pnfsids are removed. File transfer and storage will be handled by the existing dCache system. User data access is provided through the linked filenames in Lustre filesystem.

2.2. Using dCache HSM interface

dCache provides an open interface to store and retrieve data from tertiary storage system, e.g., enstore tape system [5]. Our dCache+Lustre implementation fully takes advantage of this interface with some modification. Files are first put onto stageout pools after transfer, and then migrated to Lustre, just as

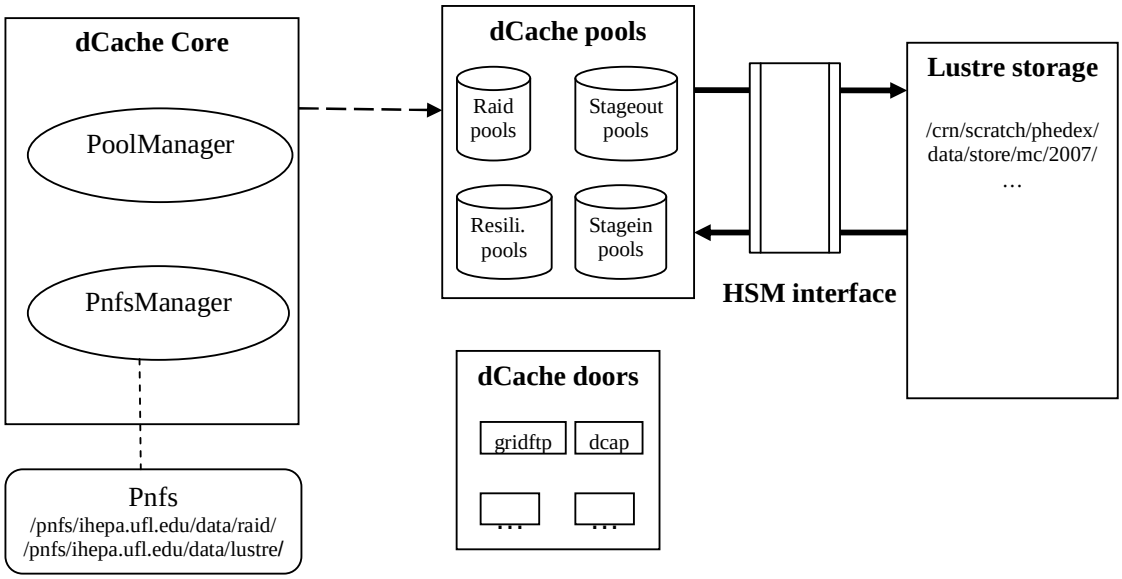


Figure 1. Use dCache HSM interface for storing/retrieving data in Lustre filesystem

being done in standard dCache with tape backend. However, before files are migrated into Lustre using their pnfsids, we map them to their original physical filenames and put the files into Lustre. Files will be mapped back to their pnfsids before staging back to stage-in pools for dCache access if missing from any dCache pools. HSM interface with callout script provides store/retrieve operation to/from Lustre and mapping between physical filenames and pnfsids. Figure 1 shows the interaction between different dCache components.

2.3. PoolManager and pnfs configuration

As the heart of dCache system is its pool manager, the pool manager decides how to handle a user read/write requests and which dCache pool to use when dCache receives a request. We need to make sure the files we intend to put into Lustre filesystem are put into the right pools for the corresponding Lustre filesystem. In order to fulfill this, we defined special directory tags within pnfs so that we can put files under a single top directory, e.g., /pnfs/ihepa.ufl.edu/data/lustre/, for controlling which dCache pools to use. Special dedicated dCache pools along with pgroup and link are defined in PoolManager.conf file for use with the corresponding pnfs tags. The configurations of pnfs and dCache PoolManager need to be modified for dCache to work properly with our new implementations.

3. Experiments and discussions

We tested both our implementations through CMS data transfer system PhEDEx [6] and CMS analysis jobs NtupleDumper. Our tests focused on two aspects of the CMS Tier-2 functionalities: one is data access for running CMS applications; another is data transfer to our CMS Tier-2 site.

In all these tests, we used the UF HPC /crn Lustre filesystem which is mounted on all the UF CMS Tier-2 worker nodes and test nodes. UF/HPC /crn Lustre filesystem has one MDS (metadata server) and two OSSs (object storage server) that manage multiple OSTs (object storage target). 10Gb network connection was in place for connecting OSSs with the UF CMS Tier-2 nodes. The following are the test results and some discussions.

3.1. Analysis job performance

To compare the analysis job performance using data on Lustre, dCache and dCache with Lustre pools, we ran CMS application NtupleDumper with CMSSW_2_1_12 used by UF analysis group against recent Cosmic run data. We first ran the jobs accessing the data located on Lustre filesystem, then accessing them through dCache dcap protocol, and lastly accessing the same data through dCache dcap protocol with Lustre backend pools.

We first put two copies of the same file: one on dCache, another one on the HPC /crn Lustre filesystem. We first ran the IO intensive NtupleDumper program on the same node using files on dCache and Lustre separately. The job running with the file on Lustre was able to finish in 386.88 seconds, and the job running with the file access through dCache dcap protocol completed in 1019.82 seconds. We then put 50 files from another Cosmic run dataset on both dCache and Lustre filesystem and re-ran the same IO intensive CMS application NupleDumper across multiple worker nodes at UF CMS Tier-2 site (each worker node got one job). We also ran the same application through dCache dcap protocol on our setup using Lustre filesystem as dCache pools. Figure 2 shows the average execution time with different number of jobs using different storage access technologies (diamond: direct Lustre access; square: dcap access xfs dCache pools; triangle: dcap access Lustre dCache pools).

When the jobs were running, we tried to make sure the worker nodes, HPC Lustre system and dCache pool nodes were not in use if possible. One thing we want to point out is that the dCache pool server node and underlying storage at UF CMS Tier-2 are the same as the one used at UF HPC for their

Lustre filesystem. As you can see, CMS jobs are generally 2.56 times faster when running with data on HPC Lustre filesystem. Also shown on Figure 2 is the running result with the same jobs for our setup with Lustre filesystem as dCache pools through dCache dcap protocol. Comparing with direct access the data with mounted Lustre filesystem, the job execution time is about 60% longer than the same job execution time on the same worker node. And the job execution time generally increases with the increasing number of jobs. Re-running the same jobs accessing the same data generally provides better performance due to cache effect. It is possible to further improve the IO related performance by using file striping in Lustre filesystem. Further investigation is needed in this area.

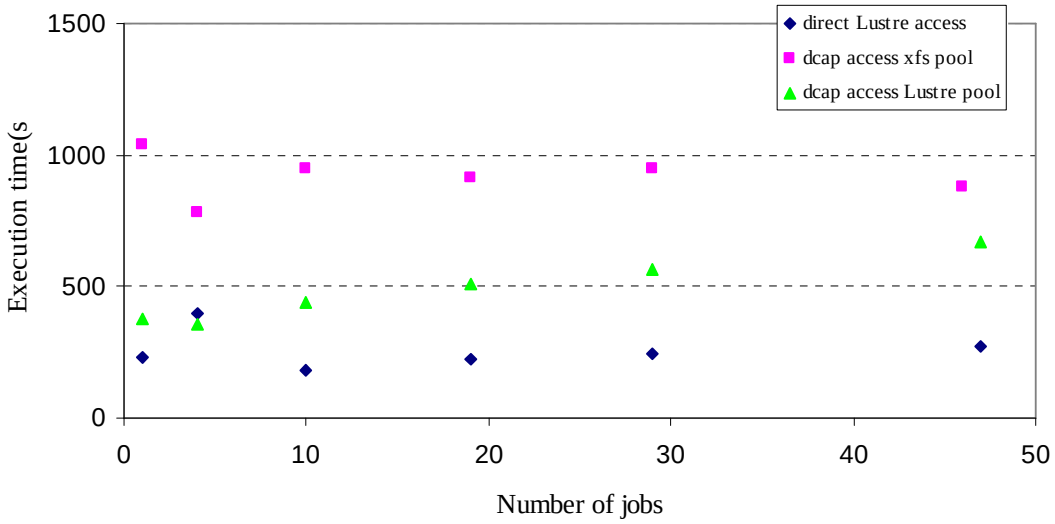


Figure 2. CMS analysis job execution time with number of jobs using different storage access technologies (diamond: direct Lustre access, square: dcap access with xfs backend dCache pool; triangle: dcap access with Lustre backend dCache pool)

3.2. Transfer performance

3.2.1. Transfer performance using dCache HSM interface

To test how well the system works, we configured 3 nodes as stageout dCache pools and 1 node as the stagein dCache pool node. We then subscribed a 9.6TB Cosmic RECO dataset (3999 files) to move the files from FNAL USCMS Tier-1 centre to the UF Tier-2 centre through PhEDEx. Initially, we only used the two computer nodes which didn't have good IO performance as dCache pools for data transfer; the rate was relatively low, less than 20MB/s. We later found out that the transfer rate was limited by the local disk rate, especially during the simultaneous read/write operations (files on the dCache pools were transferred to Lustre backend while more new files were being put on the pool disks). Later, we configured a much more powerful server node as the new Lustre pool node. And we were able to reach a rate of 50MB/s continuously. When the transfer to the pool node completed, we were able to achieve over 100MB/s from Lustre pool node to the HPC Lustre file storage system. Figure 3 shows the transfer performance using dCache HSM interface.

3.2.2. Transfer performance using directly mounted Lustre filesystem

In this test, we transferred another Cosmic dataset with 2.4 TB (2103 files) from FNAL USCMS Tier-1 centre to UF Tier-2 centre. The transfer performance is improved substantially as the files are put

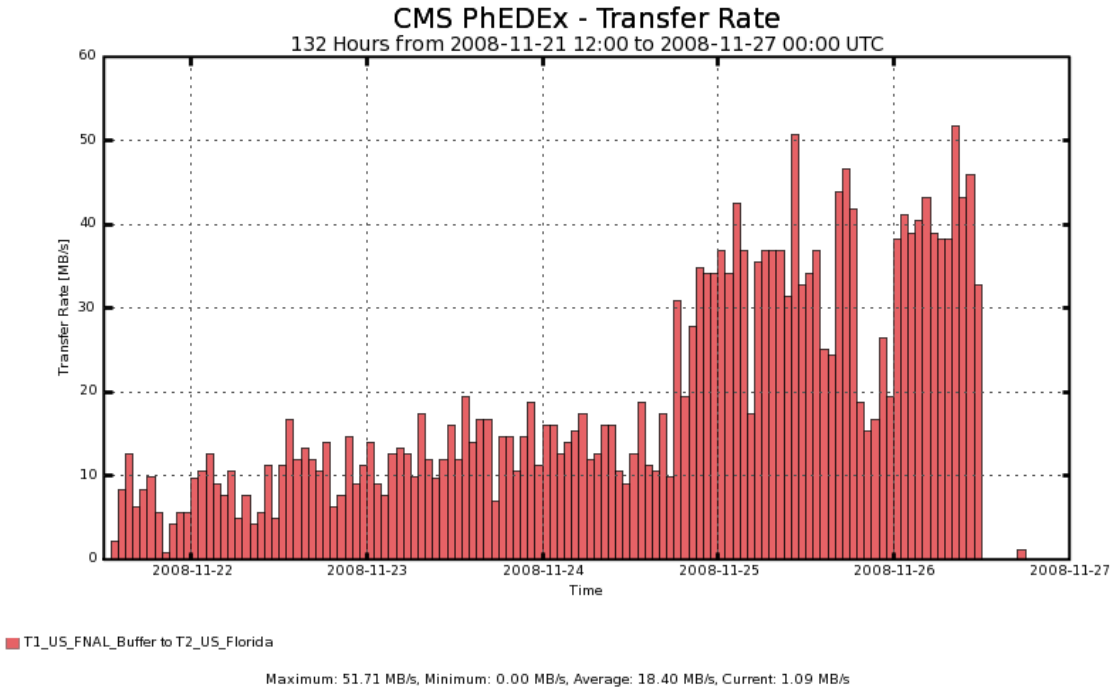


Figure 3. PhEDEx data transfer rate using dCache HSM interface implementation

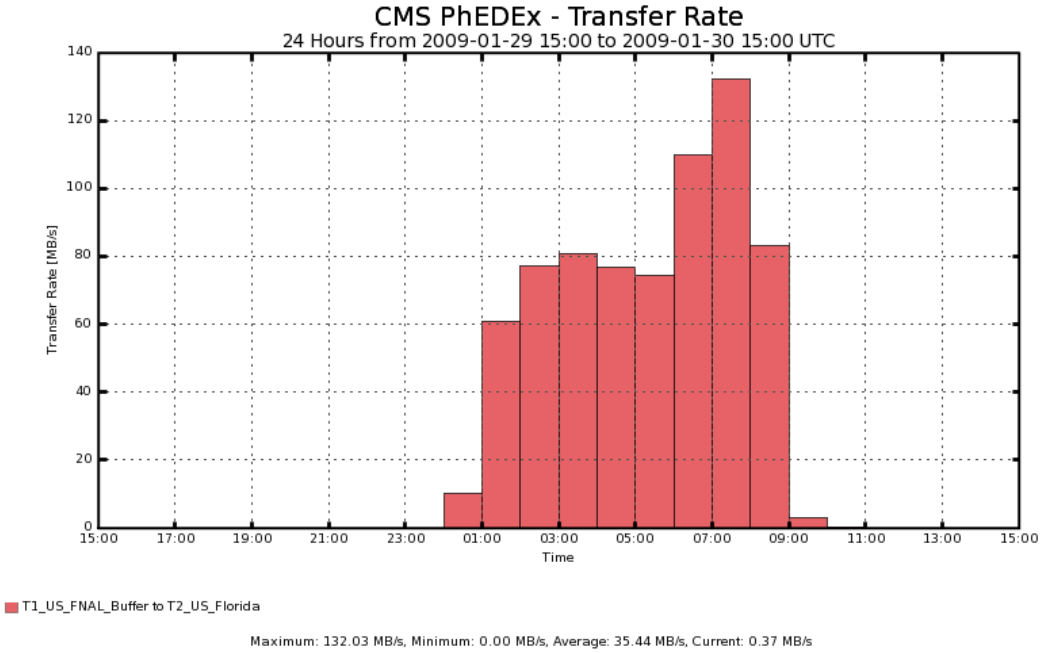


Figure 4. PhEDEx data transfer rate using directly mounted Lustre filesystem implementation

directly on Lustre filesystem without going through the local disks. As we can see from the next figure, the average transfer rate was now about 80MB/s, and the peak transfer rate was able to reach 130MB/s from the single dCache pool node we configured. Figure 4 shows the improved transfer performance using directly mounted Lustre filesystem. We want to point out that we only used one server node as dCache pool node during our test. Our intention was not to test what's the maximum transfer rate we can achieve using our current infrastructure, but to show that reasonable transfer rate can be achieved using directly mounted Lustre filesystem implementation (even with a single dCache pool node).

From the above tests, we can see that Lustre filesystem is likely suitable for using as dCache pool storage and offers reasonable transfer performance. Comparing with other distributed filesystems, e.g., NFS, we have not observed any transfer issue that happened when using NFS 3 as dCache pool storage [7]. Using directly mounted Lustre filesystem as dCache pool likely gives better transfer performance. On the other hand, the transfer rate will suffer if the connection between Lustre filesystem and a dCache pool node is not reliable. People might consider using HSM interface implementation when Lustre storage belongs to another organization that you may just want to use them as opportunity storage. Within the same organization and good network connection, using directly Lustre mounted filesystem as dCache pool is likely a better choice to achieve better transfer performance.

4. Conclusion

In this paper, we describe our experimentations to utilize Lustre filesystem for CMS analysis with direct POSIX file access while keeping dCache as the frontend for data distribution and management. Two possible implementations are described in this paper. In each implementation, we are able to use existing UF CMS Tier-2 dCache infrastructures to put files into dCache and underlying UF HPC Lustre filesystem, and then to retrieve them if requested.

Utilizing Lustre filesystem can significantly improve the data access performance than being accessed through dCache directly with dcap protocol for the CMS applications we used. Over 60% performance improvement has been observed when running jobs with data stored on Lustre filesystem. It is likely Lustre can provide faster data access rate for CMS analysis jobs with the specific CMS applications. Also, separate study shows users can access files stored in Lustre filesystem remotely with reasonable IO performance [8]. This may have important implication on how a CMS Tier-3 site may store and access the CMS data.

dCache with directly Lustre mounted pools shows reasonable transfer performance. Utilizing Lustre filesystem directly as dCache pool will likely be useful at sites with good network connections and possible same integrated site. Using HSM interface will be a good choice if a site can get opportunity storage from a larger organization and/or the network connection is not in good shape.

References

[1] "The CMS computing model", CMS NOTE/2004-31
[2] dCache web site: http://www.dcache.org
[3] Lustre web site: http://www.lustre.org
[4] Sun Press Release: http://www.sun.com/aboutsun/pr/2008-11/sunflash.20081118.2.xml
[5] Riese M et al., "The dCache book", http://www.dcache.org/manuals/Book/

[6] Rehn J, Barrass T, Bonacorsi D, Hernandez J, Semeniouk I, Tuura L and Wu Y, "PhEDEx high-throughput data transfer management system", CHEP06, Mumbai, India, February 2006
[7] Cowan G, "Using dCache with NFS", http://www.ph.ed.ac.uk/~gcowan1/dcache/dcache_nfs.txt
[8] Rodriguez J, Avery P, Brody T, Bourilkov D, Fu Y, Kim B, Prescott C and Wu Y, "Wide area network access to CMS data using the Lustre filesystem", CHEP09, Prague, Czech Republic, March 2009

Acknowledgments
Authors wishing to acknowledge a lot of assistances and advices from the colleagues at UF HPC Centre, especially Dr. Craig Prescott.