Using Kerberized Lustre Over the WAN for High Energy Physics Data

Josephine Palencia
Robert Budden
Kathy Benninger
Pittsburgh Supercomputing Center
300 South Craig St
Pittsburgh , PA 15223
(412) 268 4960
josephin,rbudden,
benninge@psc.edu

Dimitri Bourilkov
Paul Avery
Mengxing Cheng
Yu Fu
Bockjoo Kim
University of Florida
Gainesville, FL 32611
(353) 392 3261
bourilkov, avery, cheng, yfu,
rkim@uf.edu

Dave Dykstra
Nirmal Seenu
Fermi National Laboratory
P.O. Box 500
Batavia, IL 60510
(630) 840 3000
dykstra, seenu@fnal.gov

Jorge Rodriguez
John Dilascio
Florida International University
Miami, FL 33199
(305) 348 2000
jrodrig, jdilascio@fiu.edu

Drew Oliver
Daniel Majchrzak
University of Southern Florida
Tampa, FL 33620
(813) 974 201
drewoliver,dan@usf.edu

Donald Shrum
Jim Wilgenbusch
Florida State University
Tallahassee, FL 32306
(850) 644 8630
dcshrum, wilgenbusch@fsu.edu

ABSTRACT

This paper reports the design and implementation of a secure, wide area network, distributed filesystem by the ExTENCI project (Extending Science Through Enhanced National Cyber Infrastructure) based on lustre. The filesystem is used for remote access to analysis data from the Compact Muon Solenoid (CMS) experiment at the Large Hadron Collider (LHC), and from the Lattice Quantum ChromoDynamics (LQCD) project. Security is provided for by kerberos and reinforced with additional fine-grained control using lustre ACLs and quotas. We show the impact of using kerberized lustre on the IO rates of CMS and LQCD applications on client nodes, both real and virtual. Pre-configured images of lustre virtual clients containing the complete software stack ease the difficulty of managing these systems.

Categories and Subject Descriptors

D.4.3 [File Systems Management]: Distributed FileSystem;
D.4.6 [Security and Protection]: Authentication;
D.4.8 [Performance] Measurements;
J.2 [Physical Sciences and Engineering] Physics.

General Terms

Filesystems, Experimentation, Security, Reliability, Physics Applications Benchmark, Virtualization.

XSEDE12, July 16 - 20 2012, Chicago, IL, USA

Keywords

Kerberos, realms, lustreWAN, distributed OST pools, LHC, CMS, ROOT, LQCD, PKINIT, virtual lustre clients, virtualization, XEN, VMware, KVM, VirtualBox, OSG, Tier3.

1. INTRODUCTION

Kerberos is a network authentication protocol designed to provide a strong authentication mechanism for client-server applications through secret-key cryptography as well as user access to these systems. When users request access to the GSS-protected filesystem, kerberos nominally performs user and service credential checks over the network. Within kerberos, realms are created, and principals (systems and users) are placed in secure realms where the principals authenticate to other principals within the same realm. Principals in one realm can also authenticate to principals in other realms to achieve cross-realm security. The capability for cross-realm authentication is a necessity in the insecure wide area network. Other security standards such as OpenID and InCommon are more oriented towards web applications. Globus, which uses the X509 certificates, has the similar capability for cross-realm, federated user ID however, does not provide security to the internals of the lustre filesystem. One can enable interoperability between kerberos and Globus through PKINIT standards within kerberos so users can authenticate in the kerberos framework using their X509 certificates.

Lustre is a high performance, parallel filesystem for massive storage with good scalability for applications. The distributed filesystem has three functional units: a single metadata server, (MDS) with a single metadata target, (MDT) that stores namespace metadata; one or more object storage servers (OSSes) that store file data on one or more object storage targets (OSTs); and the clients that access and use these data.

The incorporation of kerberos into lustre is a unique endeavor that extends the authentication to all these lustre functional units as well as user access to the protected filesystem. Site firewalls become optional as the lustre network now has a blanket security over all its components which are each given unique keytabs specifying their functions. System components within the lustre network can operate with RPC message and bulk data protection enforcing data checksum, privacy, or integrity. Furthermore, secure, distributed OSTs and OST pools decentralize the lustre storage across organizations and benefit from the faster local IO. Previously unsupported, kerberos functionality is now being incorporated into the test suite of the lustre mainstream code.

Two main high energy physics applications, CMS, and LQCD are ported into the secure lustre framework and their IO evaluated across the WAN. CMS is one of LHC's general-purpose detectors, capable of studying many aspects of proton collisions at 14TeV. See **Figure 1**. Designed to measure the energies, and momenta of hadron residuals after collisions, it aims to confirm the existence of the Higgs boson, explore physics at the TeV energy scale, study dark matter and dark energy, and look for evidence of physics beyond the Standard Model such as supersymmetry and extra dimensions. At present, close to 3600 physicists from over 38 countries and 183 institutes collaborate on the CMS experiment.

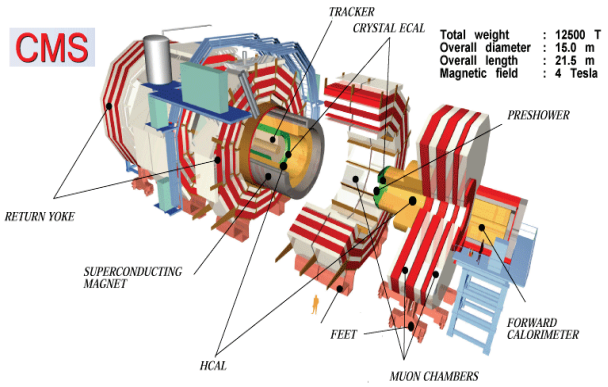


Figure 1. The Compact Muon Solenoid (CMS) Detector The detector contains subsystems such as its silicon-based tracker, the scintillating crystal calorimeter, and the large muon detectors; the tracker is designed to follow the trajectories of high-energy particles after collisions.

Lattice QCD data is primarily based on studies of the confinement of quark-gluons plasma via the lattice gauge theory formulated on a grid, or lattice of points in spacetime. The discretized formulation of QCD rather than continuous spacetime circumvents the highly nonlinear nature of the strong nuclear force. Other frameworks implemented include CERN ROOT which is included in CMSSW (CMS Software).

2. GENERAL SETUP

We incorporate kerberos into the source lustre code by enabling GSS-support in the builds. **Figures 2** and **3** show a simple and a more detailed overview of the ExTENCI lustre WAN filesystem, respectively. The KDC and the lustre servers are based at the University of Florida (UF), an Open Science Grid (OSG) Tier3 site. The secure filesystem is accessed by other Tier3 sites such as Florida International University (FIU), University of South Florida (USF) and Florida State University (FSU), Fermilab, a (Tier1) site and PSC. Virtual lustre clients running XEN, VMware, VirtualBox and KVM mount the /extenci filesystem after being authorized and granted unique keytabs by UF.

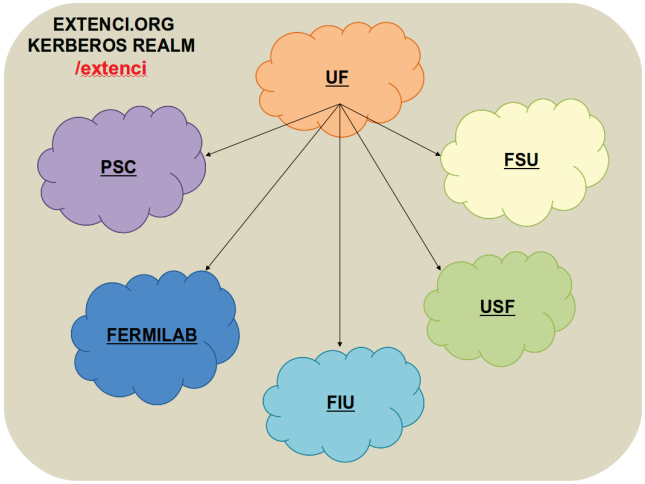


Figure 2. ExTENCI LustreWAN General Overview Kerberos realm, EXTENCI.ORG was established to create the secure lustre network for UF, PSC, Fermilab (FNAL), FIU, USF and FSU, allowing access only to authorized systems and users. UF manages the kerberos, MDS and OSS storage servers. The lustre wan filesystem is called /extenci.

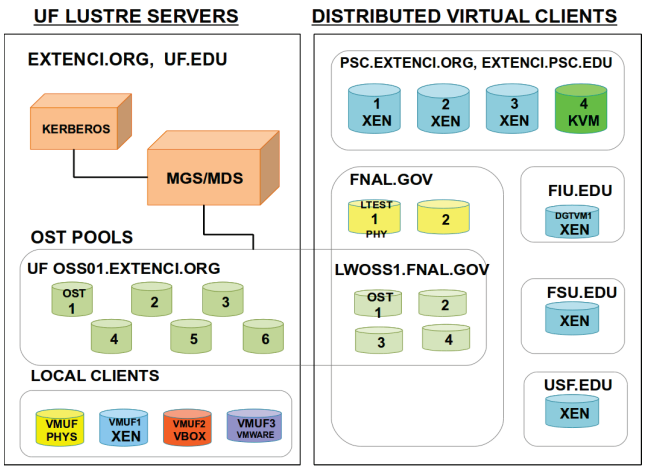


Figure 3. ExTENCI with OST Pools and Virtual Clients OST pools were created at UF and FNAL in addition to the virtual lustre clients running XEN, VirtualBox (VBOX), VMware and KVM; the VMs are booted from pre-configured images and securely mount /extenci after being authorized and given unique keytabs by UF; site firewalls are optional.

Lustre kerberos supports point-to-point, and blanket security protection for network negotiations between lustre components-servers and clients so that site firewalls are no longer mandatory. Table 1 lists the different lustre kerberos flavors that can be specified among systems within a lustre network. Iozone benchmarks from both local and remote clients show that krb5{n,a} checksum kerberos protection on bulk data introduce

no sizable lustre IO overhead while krb5{i,p} integrity and privacy stricter protection exact a high IO penalty of 50-80%. (**Figure 4**). Hence, the kerberos default security, krb5n is adopted to provide the adequate security for the realm without sacrificing performance.

Table 1. Lustre kerberos security flavors naming the various types of authentication and the protection provided.

KERBEROS FLAVOR	null	krb5n	Krb5a	krb5i	krb5p
RPC message protection	null	null	header integrity	integrity	privacy
Bulk data protection	null	checksum	checksum	integrity	privacy

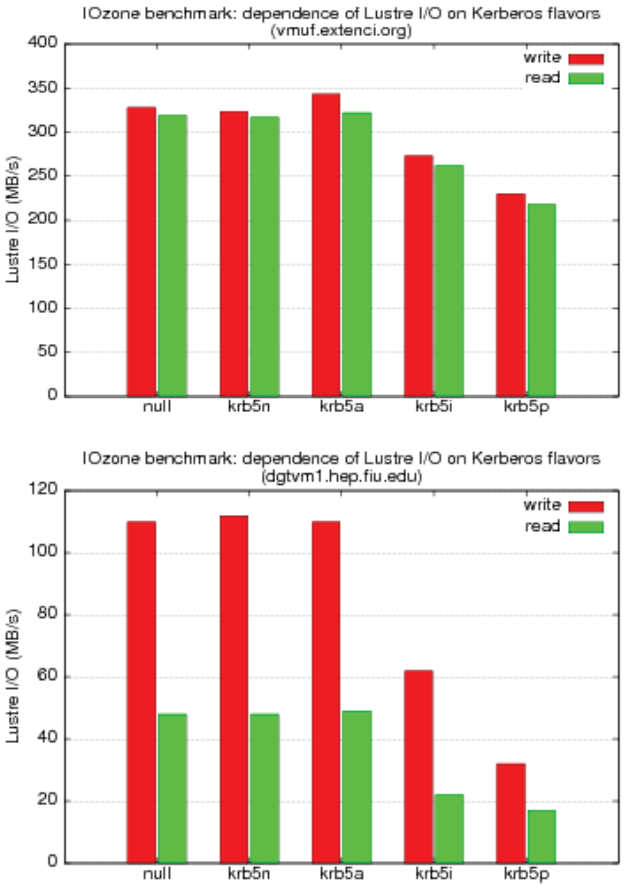


Figure 4. Dependence of Lustre IO on Kerberos Flavors
An instance of iozone on a non-striped file stored in lustre shows that krb5{n,a} with checksum protection has no impact on IO while krb5{i,p} with stricter integrity, privacy protection slows down the filesystem IO considerably. Clients: vmuf (10GigE, UF) and dgtvm1 (1GigE, FIU).

User accounts are centralized and regularly synced from UF. Current lustre kerberos requires that the user database be consistent on the metadata server and all its clients. A three-fold check (UID, GID, and kerberos credentials) is performed when a user logs in and requests access to the protected filesystem. This shouldn't be the case for a purely kerberos deployment. To aid in federated ID, LDAP/PAM is also being enabled within kerberos.

2.1 Secure OST Pools

OST pools enable the grouping of OSTs for file striping purposes with automated free-space leveling occurring within the pool. OST pool usage is specified and stored along with other striping information such as stripe count and size for directories, or individual files. Iozone benchmarks at UF and Fermilab OST pools show marked improvement on the lustre IO when local storage is used. Tier-3 universites (FIU, USF, FSU) and Fermilab can use the nearest OST pool nearest to them. OST pools created at UF and Fermilab contribute to a total storage of 60TB. See **Figure 5.**

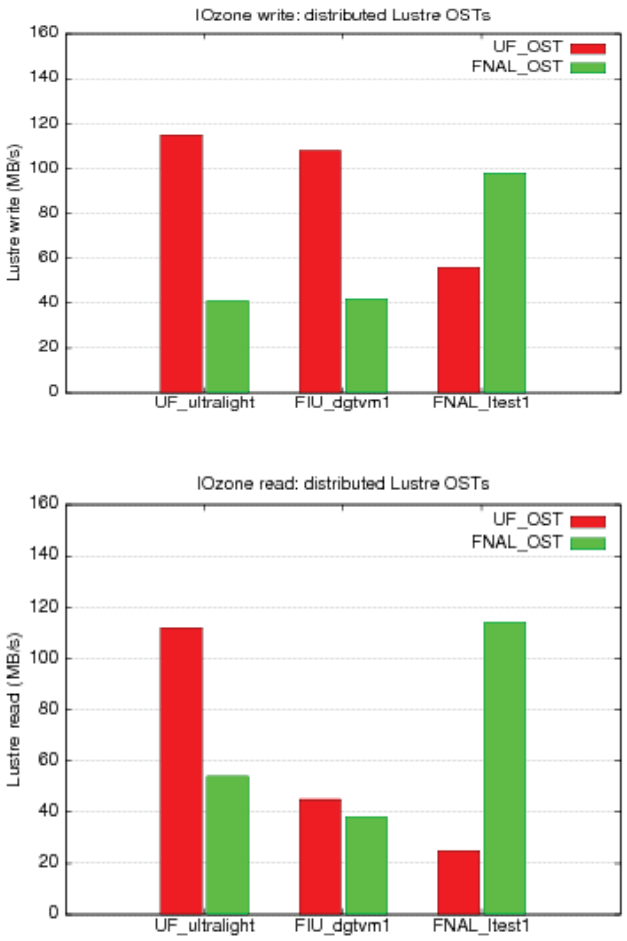


Figure 5. Distributed OST Pools at UF and Fermilab
OST pools at UF and FNAL provide fast local IO as shown by the results from iozone read and write measurements. Tier3 universites (FIU, USF, FSU) and Fermilab can use the closest OST pool available to them and saturate the full network bandwidth achieving close to 120MB/s with kerberos enabled. Clients: ultralight (1GigE, UF), dgtvm1 (1GigE, FIU), ltest1 (1 GigE, FNAL).

2.2 Packaged Virtual Lustre Clients

System images preconfigured with kerberos and lustre (quota, ACLs) as well as the application software stack (CMS, LQCD, ROOT) make the setup and administration of the systems easier. This is especially true for sites with limited, or no resources and system support. The various virtualization appliances supported in the image include XEN, VirtualBox, VMplayer and KVM. The client images are downloaded by users, and booted up to mount /extenci. In **Table 2**, VMUF is a physical virtualization server at UF that's also use as the client testbed for XEN (VMUF1), Virtualbox (VMUF2), VMplayer (VMUF3) and KVM (VMUF4), not on the table; VMUF, and the OSS server both have 10GigE cards. Iperf measurements between the VM clients and the OSS show that the network IO for VirtualBox and VMplayer are not fully optimized availing only of 15-20% (1.5-1.9Gb/s) of the full bandwidth while lustre on a XEN client comes close to the full network 10GigE capabilility of the server (9.6Gb/s).

Table 2. Network IO of Various Virtualized Systems

CLIENT	CLIENT-> OSS (Gbps)	OSS -> CLIENT (Gbps)
VMUF Physical	9.9	9.9
VMUF1 XEN	9.6	7.2
VMUF2 VirtualBox	1.9	1.5
VMUF3 VMPlayer	1.5	.78

3. APPLICATIONS

We evaluate the performance of two high energy physics applications- CMS, and LQCD on the ExTENCI lustre filesystem.

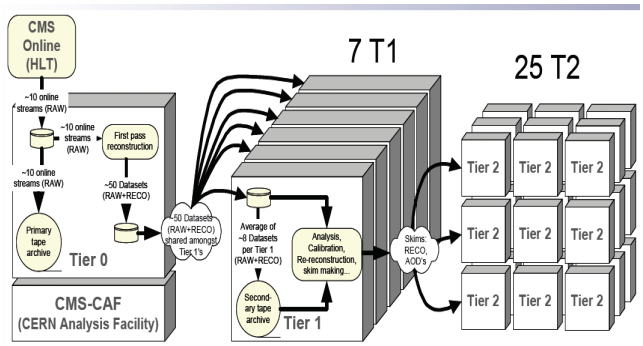


Figure 6. CMS Monte Carlo Data Flow

From the CMS detector, data goes online and cascades down various tier levels (T0,.., T3) where it undergoes processing and reconstruction. At T0, RAW data is received from the CMS detector, and repacked (RECO) where unsorted streams are sorted into physics streams of events with similar characteristics. Reconstruction algorithms are ran to produce AOD. RAW, RECO and AOD are exported to T1. At T1, data is further redistributed after reprocessing with improved algorithms. At T2, Monte Carlo events are generated. At T3, the filtered AOD is ready for analysis.

3.1 CMS Data Analysis

CMS data is arranged into a hierarchy of data tiers. See **Figure 6**. Each physics event is written into each data tier containing different levels of information about the event. The RAW data contains the full event information from Tier0 (T0) and has the raw detector information such as detector element hits; it is not used for analysis. The RECO, or reconstructed data is the output from the first processing by T0. This layer contains highly detailed reconstructed physics objects that is still too large for analysis. The AOD (Analysis Object Data) is the distilled version of the RECO event information that represents a good balance between event size and complexity; AOD can be used for data analysis. Following the data workflow, (RAW) data is received from the CMS detector experiment at T0 where it is repacked and reconstructed (RECO) to produced AOD. RAW, RECO and AOD are exported to Tier1 (T1) and redistributed. At Tier2 (T2), Monte Carlo events are generated. Finally, at Tier3 (T3), users can prepare their analysis code, run on the data and collect results. There are two stages in the CMS data analysis- SCRAM and RUN. SCRAM involves the CMS environment building time for the compilation and linking of various libraries; RUN consists of actual data processing. We tested CMSSW-3.9-7 muon analysis code at four remote lustre clients. See **Figures 7**. Prelimary results of CMS data processing on the lustreWAN gave poor client performance at PSC and FNAL. Network latency greatly lengthened the roundtrip time for every access back to the servers. CMS SCRAM encountered over 8.5x10 4 file accesses (2.4 x 104 opens, 2.4 x 104 stats, 3.6 x 104 lstats and 1.5x103 readlinks) from 2 x 104 directories, 1.8x105 files with sizes ranging from a few KB to several MB. The LOSF (lots of small files) scenario compounded the effects of large network latencies during CMSSW SCRAM. To resolve the situation, the 12GB CMSSW software was made resident on the lustre VM client's local partition, /local/cmssw while the input or data files were placed on the lustre filesystem. See **Figure 8**.

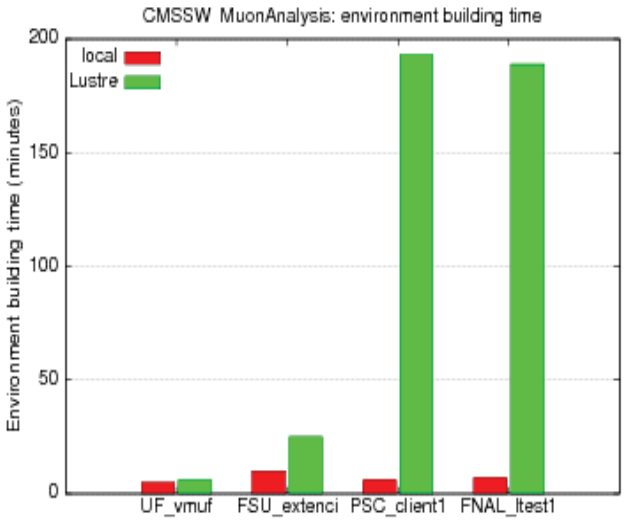


Figure 7. CMS SCRAM on Lustre and Local Partition

CMS SCRAM's environment building time is adversely affected by larger network latencies. What normally takes a minute for SCRAM to complete in the local partition lengthened to 200min at more remote sites such as PSC (client1, 47ms RTT-round trip time) and FNAL (ltest1, 60ms RTT) compared to UF (vmuf, .01ms RTT), FSU (extenci, 5ms RTT) which finished within 25 minutes.

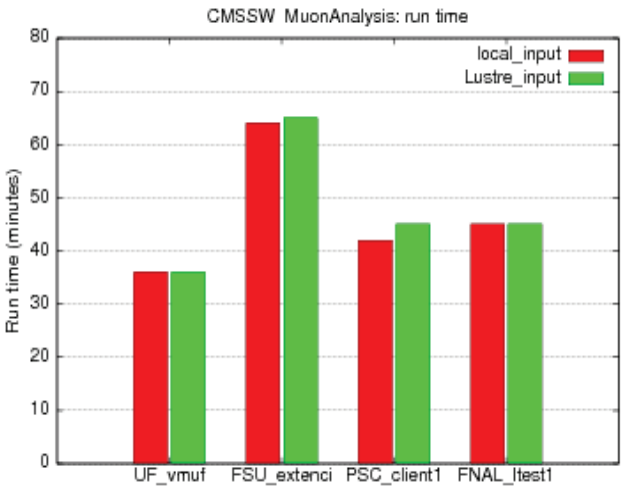


Figure 8. CMS SCRAM on Client's Local Partition
CMSSW was made resident on the client's local partition while the CMS input data was placed on the lustre filesystem to overcome the larger network latencies over the WAN. Once this arrangement was adopted, CMS data analysis could be done on the WAN with performance similar to that solely running on the local partition. FSU network had problems during this test, hence the slightly higher run time.

3.2 ROOT

ROOT, born at CERN is a huge, object-oriented framework aimed at tackling some of the data analysis and processing challenges in high-energy physics. It provides a data structure that is extremely powerful for fast access to huge amounts of data with orders of magnitude faster than any database. Contained within ROOT are mathematical and statistical tools that can also be used to simultaneously operate on the data in parallel. Data can be generated following any statistical distribution, making it possible to simulate complex systems. ROOT graphics can show results via histograms, scatter plots, fitting functions, which can be adjusted real-time by few mouse clicks. It also allows interactive sessions to write macros, or compile programs.

Used by physicists to analyze their data, or to perform simulations, ROOT is fundamental to all LHC experiments and is directly called upon by CMS for IO. ROOT can save the CMS data in a compressed binary form called the ROOT file along with the object format. With intelligent data compression, the ROOT file can have similar events in trees and leaves clustered next to each other. The ROOT trees spread over several files can be chained and accessed as a unique object, allowing for loops over huge amounts of data. Parameters can be optimized (e.g. read-ahead) and processing can be very fast in reading a few variables from each event and slow with complete full events. Consequently, IO can become CPU-intensive depending on the occurrence and frequency of (de)compression. The data saved into one or several ROOT files can be accessed from the PC, the web, or the GRID.

To study the scalability of CMS data analysis for multiple ROOT instances on lustre, we designed and customized ROOT files with a few 10^3 lines of code and fed it into the ROOT framework. With the tuning of the ROOT file, full control of the data format is regained. Numerous tree structures can be tested by varying the number of branches and leaves from the very simple to that closely resembling complex CMS data.

Using ROOT-v-5.30, we performed lustre IO reads on a non-striped file. See **Figure 9**. Choosing a ROOT file with 20 branches and 5000 leaves, we obtained results indicating that in contrast to IO performed on the local partition (plateaus at 60MB/s), lustre has very good scalability with increasing number of ROOT instances. The 1st root instance clocked reads at 64MB/s, the 2nd at 130MB/s, and so forth reaching almost 400MB/s for 8 instances. We were not able to test beyond 8 instances because we discover that ROOT reads can also be CPU-intensive in proportion to the frequency of data (de)compression. We're designing a ROOT file that will be more IO-intensive to saturate the lustre filesystem and also testing beyond 8 CPU cores to achieve rates over 400MB/s once resources become available.

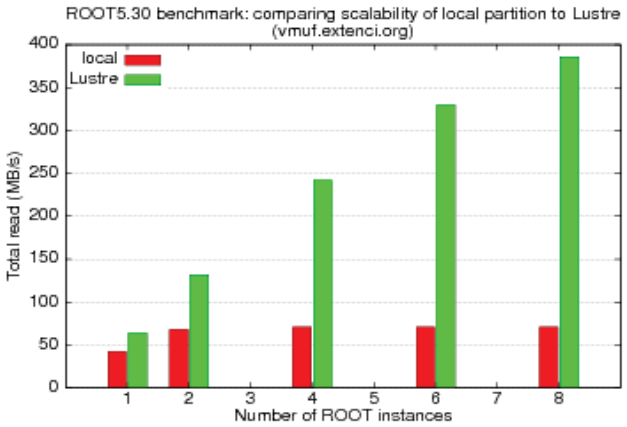


Figure 9. CMS: Lustre Scalability of Multiple Root Instances
In constrast to IO performed on the local partition (saturates at 60MB/s), Lustre exhibits very good scalability with increasing number of ROOT instances. The 1st root instance yielded reads at 64MB/s and almost reached 400MB/s for 8 instances. A ROOT file with 20 branches and 5000 leaves was used with vmuf, (10GigE, UF).

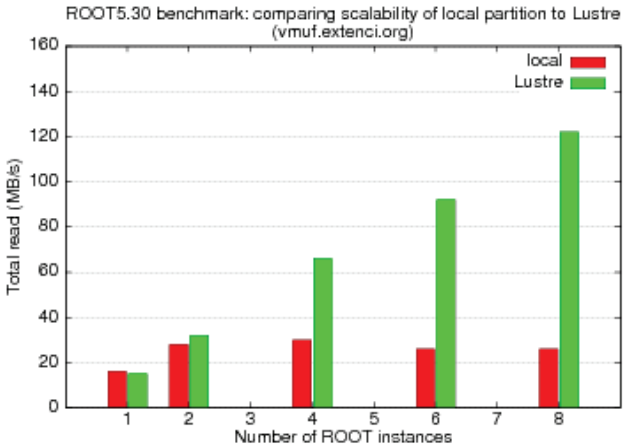


Figure 10. Perfect Linear Scalability with Lustre with Increasing ROOT Instances. Using a 2nd ROOT file with 2 branches, 2 leaves per branch and 20GB of random numbers, we achieve the perfect linear scalability with lustre. Though compared with the 1st ROOT file, the throughput per instance was lower with 15MB/s for the 1st instance, and 120MB/s with 8 instances; client vmuf (10GigE, UF).

We scanned through many other CMS data tree structures and found that the ROOT file that achieved the perfect linear scalability with lustre corresponds to 2 branches, 2 leaves for each branch and 20GB of random numbers (**Figure 10**). This is in spite of the low 15MB/s for the 1st root instance, 35MB/s for the 2nd root instance, and 120 MB/s for 8 instances. We continue to investigate all the optimizable parameters in the ROOT file, then couple them with the lustre tunables to achieve the best IO throughput. We also investigated CMS ROOT lustre scalability with multiple lustre clients (**Figure 11***)*. We ran separate serial and parallel iozone writes and ROOT reads using 7 clients. Our results show that the stacked summation from the serial runs was very close to that of the simultaneous IO throughput on the OSS storage when all the clients are running in parallel. This proves that lustre is not only is scalable with the increasing number of ROOT instances but also gives scalable total IO throughput with the increasing number of clients.

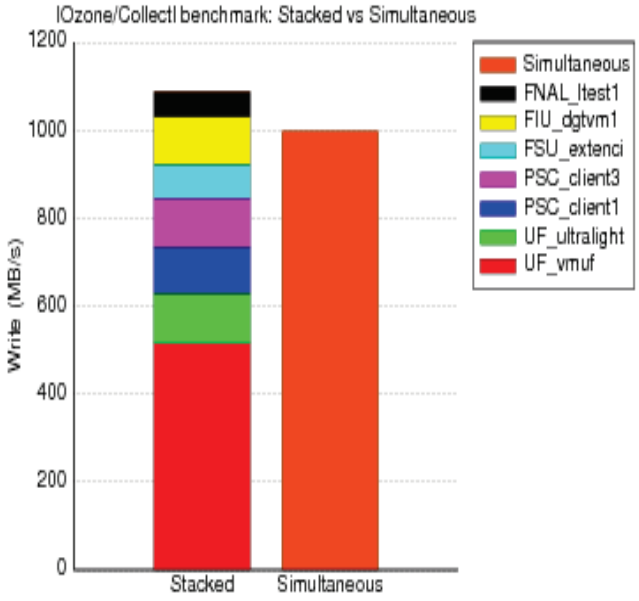
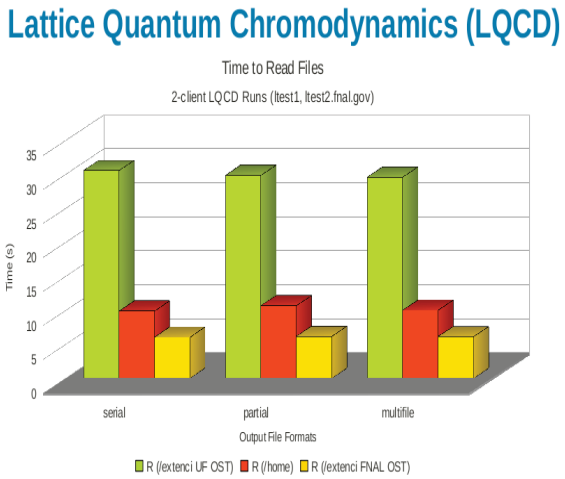


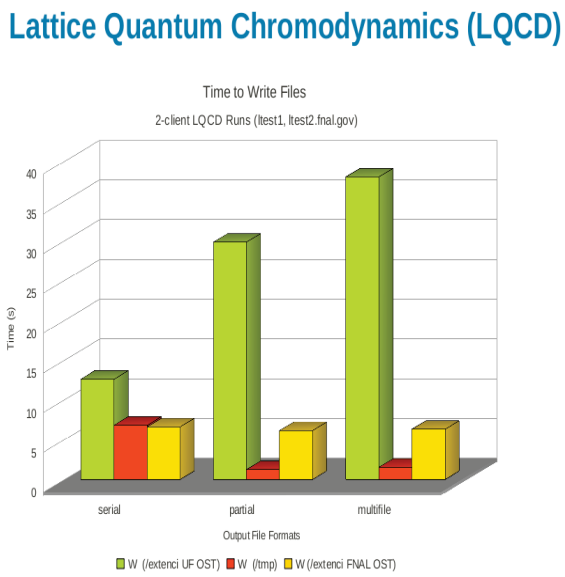
Figure 11. CMS: Lustre Scalability with Multiple Clients. Stacked summation of iozone write and ROOT reads for 7 clients (1xFNAL: ltest1; 2xPSC: client1, client3; 1xFSU: extenci; 1xFIU: dgtvm1; 2xUF: ultralight, vmuf) after sequential (W: 1.1 GB/s, R: 675 MB/s) runs is very closed to simultaneous (W: 1 GB/s, R: 600 MB/s) total IO throughput when all clients are running in parallel. All the clients have 1GigE interfaces except vmuf (10GigE). All clients are also virtual with the exception of UF's vmuf, ultralight and FNAL's ltest1 which are physical machines.

3.3 Lattice QCD

The QCD theory describes the interactions between quarks and gluons. In lattice QCD, the fields representing the quarks are defined at lattice sites while the gluon fields are defined on the links connecting the neighboring sites. This approximation approaches continuum QCD as the spacing between lattice sites is reduced to zero. The software stack used to build the static SU3 for numerical simulations of 4D lattice gauge theory includes the milc_qcd-7.6.3, mvapich-2.1.7-r5225, scidac and scidac-mvapich.The SCIDAC code (Scientific Discovery through Advanced Computing) added functional levels with the QCD physics toolbox, QOP optimization in ASM, QDP-data parallelism, QLA-linear algebra and QMP- message passing capabilities. The milc input files are read serially while the milc output files are written using three different SciDAC formats namely, serial_scidac, partfile_scidac, and multifile_scidac. (**Figures 12***)*. LQCD was ran from two clients at Fermilab having the infiniband (IB) network back-end for message-passing while



Number of Input Files	Number of Output Files	Input File Sizes	Output File Sizes	Output File Formats
2	1	579MB, 495B	579MB	serial
2	16	"	37MB	partial
2	16	"	37MB	multifile



Figures 12. LQCD Multi-client Runs at Fermilab Two clients at Fermilab (ltest1, ltest2.fnal.gov) ran the milc code. The input files are read serially while the milc output files are written using three different SciDAC formats- serial, partfile, and multifile_scidac illustrate the fastest IO for reads were obtained if Fermilab uses its own OST pool. For the first figure, reads using UF's OST pool took 30s, IO for writes was best on local partition with the exception of the serial output format which was comparable.

performing IO on lustre. Three filesystems are compared- UF OST pool, Fermilab's client local partition (/tmp, /home), and Fermilab's OST pool. Results indicate the benefit of using Fermilab's local OSTs (versus UF's OSTs) with the 6-fold improvement on read IO across all output file formats in contrast to the 3-fold IO improvement using the local partition. Results for write IO, however showed preference for the local partition.

4. ISSUE AND FUTURE WORK

The foremost issue we encountered was the lack of support of kerberos in lustre. Clients crashing led to instabilities during the applications run and slowed down our progress. Fortunately, Naval Research Lab (NRL) is working to include the kerberos functionality into WhamCloud's test suite and finally into the lustre mainstream. We continue our efforts in applications profiling and tuning to optimize their performance on the WAN, to fully understand the CMS data tree structures and to predict which ROOT file would best be optimized on the lustre filesystem. We hope to better integrate with CERN tools and the XSEDE resources to increase the project's interoperability with other organizations. Having a functional PKINIT using grid certificates would also allow users to authenticate in the kerberos realm using their OSG X509 certificates. Finally, we continue to fine tune the packaging of client virtual images. A client management console, and a GUI interface would make it easy for users to create, boot, and launch secure virtual lustre clients as well as specify the desired memory, number of CPUs, OS, applications, and sync application versions

5. ACKNOWLEDGEMENTS

We acknowledge funding from the National Science Foundation, NSF Proposal 1007115 ExTENCI with OSG. We thank Ken Hornstein (NRL) and Chris Gearing (WhamCloud) for the work to include kerberos functionality into lustre mainstream. Our thanks to Elizabeth Albert (PSC) for creating the figure templates used in references 4 and 5.

6. REFERENCES

[1] Palencia, J., Budden, R., Benninger, K., Bourilkov, D., Avery, P., Cheng, M., Fu, Y., Kim, B., Dykstra, D., Seenu, N., Rodriguez, J., Dilascio, J., Shrum, D., Wilgenbusch, J., Oliver, D., Majchzak, D., "Using Kerberized Lustre over the WAN for High Energy Physics Data. " Proceedings from the Lustre Users Group 2012 Conference (Austin, TX , April 23-25, 2012). Available at www.opensfs.org/wp-content/uploads/2011/11/lug2012-v20.pdf

[2] D. Bourilkov, P. Avery, M. Cheng, Y. Fu, B. Kim, J. Palencia, R. Budden, K. Benninger, J. Rodriquez, D. Dykstra, N. Seenu, "Secure Wide Area Network Access to CMS Analysis Data using the Lustre Filesystem," Computing in High Energy and Nuclear Physics (CHEP) 2012, New York

[3] D. Bourilkov, P. Avery , M. Cheng, Y. Fu, B. Kim, J. Palencia, R. Budden, K. Benninger, "Using Virtual Lustre Clients on the WAN for Analysis of Data from High Energy Experiments," Computing in High Energy and Nuclear Physics (CHEP) 2012, New York

[4] Palencia, Josephine, Budden, Robert and Sullivan, Kevin, "Kerberized Lustre 2.0 over the WAN," TG10 Proceedings of the 2010 TeraGrid Conference ACM New York, NY, USA ©2010 ISBN: 978-1-60558-818-6. Available at http://dl.acm.org/citation.cfm?id=1838589&jmp=cit&coll=GUIDE&dl=GUIDE#CIT

[5] Palencia, J., Budden, R., and Sullivan, K. "Kerberized Lustre 2.0 over the WAN. " In Proceedings from the Lustre Users Group 2010 Conference (Aptos, CA, April 14-16, 2010).

[6] CMS, cms.web.cern.ch/content/cms-physics

[7] LQCD, www.physics.utah.edu/~detar/milc/milc_qcd.htm

[8] Lustre, wiki.whamcloud.com/display/PUB/Documentation

[9] MIT Kerberos, web.mit.edu/kerberos