

# Lawrence Livermore National Laboratory

## Customer Support and I/O Applications

LLNL-PRES-403156



Richard Hedges

# Topics for today

---

- Introduce our users and applications
- Somewhat chronological discussion of Lustre at LLNL
- Emphasize events and aspects which impact usability
- Explain some of our support strategies
- Conclude with our latest and greatest success story



# Customers and Systems

---

- Multidisciplinary Scientific and Engineering Research:  
3D and nonlinear simulations
- Both scientists and computer scientists on code teams
- Current generation of ASC system (purple):
  - AIX - 100 teraflops - 50 TB memory - 2 PB RAID disk
  - 100 GB/sec I/O delivered to single app
- Many Linux clusters
  - Accomplish ASC architecture with commodity hardware and software
  - Beginnings of Lustre
  - Replicate as needs demand and funding allows
- BG/L
  - 131,172 CPUs initially, now 212,922
  - 1 ION/ 64 nodes = 1664 Lustre Clients



# High Level Application Synopsis

---

- Programming Model
  - Many shared memory nodes of modest local scale (2 – 16 processors)
  - Individual jobs running on hundreds to thousands of processors
  - Distributed memory (MPI communications), shared memory (maybe) within node
  - All nodes mount a shared parallel file system: codes can perform parallel I/O to a single shared file or a file per process in a single or multiple directories
  - Codes written in C, C++, FORTRAN
  - Writes dominate reads by about a 5 to 1 ratio
    - Changing due to moving data to and from archive



# Parallel File Systems before Lustre

---

- GPFS customer for ~ 5 years on AIX systems
  - Acceptance and performance testing of GPFS
    - IOR version 1
    - MPI/IO
- ASC I/O
  - For parallel system be able to dump 1/2 of memory in 5 minutes
  - Parallel I/O via software “layer cake” : parallel HDF5, MPI-IO, parallel file system
- Scalable I/O Project
  - Support and promote parallel I/O libraries: MPI-IO, HDF5
  - Initial development of IOR version 2



# Transition to Lustre - early Linux Clusters

- Initially 64 NFS file systems (yikes!)
  - **ddcMD** (each process writes small file for each snapshot) loved it
- Heavy internal pressure to get Lustre into production
- De-emphasis of general solution in favor of direct support
- Support of customers encountering serious Lustre bugs
  - Serious == interfering with productive use of Lustre or cluster
  - Quantify/reproduce problem: work with code team
  - Find workaround or bug or both
  - Document “better practices”
- Survey of IO models
  - Gained some more detailed understanding of I/O on key codes
  - Established working relationship with code teams



# Early Lustre Systems in Production

---

- Dedicated file system for each cluster
- Lustre team primarily involved in testing and characterizing site relevant bugs
- Early attempt to use BlueArc filers as OSTs
- **Customers frustrations:**
  - Lustre robustness not near to GPFS
  - Not used to possibility of short read/write
  - Shared file users forget manual stripe settings



# BG/L

---

- New hardware, OS, port of Lustre
- Major new performance tuning effort focused on BG/L's unique architecture
- SLIC: cluster mounting BG/L file systems with single purpose of offloading data to storage (archive)
- MPI-IO, HDF5, pNetCDF not so useable at scale
- Metadata (file creation) performance becomes obvious issue to customers

## **ddcMD**

- File per process per snapshot not going to work: creating 128k files takes ~2 minutes, snapshot about minutes of simulation
- Change of I/O Model: sharing files at lower scale (~100 files) than process count (128 k)
- Offloading data from Lustre to Storage (archive) becoming an issue



# Multi-cluster file systems: Spring 2007

---

- Had not yet achieved acceptable stability
- Adding new series of clusters (6) to existing (4)
- 3 (now 2) file systems on each network
- More storage
  
- New problems
  - Contention related:
    - BG/L can easily initiate denial of service to other clusters
    - Difficult to identify and localize source of performance issue
  - Issues moving data to Storage



# Diagnostic Work Flow

## for issue associated with individual application

- Identify individual (harder than it may sound for multi cluster)
- Verify problem
  - If broad impact to system performance, control conditions
- “strace -etrace=open,close,lseek,read,write” same or related problem(smaller scale) extracting parameters of the I/O:
  - transfer size(s)
  - file size(s)
  - shared file or file per process
  - Any stalling conditions, etc.
- Map parameters onto IOR or write small kernel
- bugzilla with kernel IOR command line



# Lessons Learned: general comments

---

- Focus upon individual customer situations instead of general solution was critical to success: long term involvement crucial
- Customer will never take your best advice as soon as you would hope (until desperate)
- When you can proactively identify a customer with a performance issue, you are miles ahead in credibility as well as in solving the problem: *would like to see new admin tools in this area*
- Parallel I/O libraries not performing at present scale: *some new Lustre focused optimization efforts under way*
- Have a useful methodology to approach user encountered issues



# Customer Success Story: ddcMD on BG/L again

---

## Review

- 64 NFS filers: 128k processes \* 1file \* # snapshot = a heck of a lot of small files and post-processing nightmare
- Lustre: (file creation) performance becomes issue
  - Aggregate data at lower scale (100 files) than process count

## Mid 2007:

- “How can I get the same performance as you get for your maximum throughput tests?”
- “Need comprehensive life-cycle plan for our data.”



# What is ddcMD?

---

- Molecular Dynamics code
- studying Kelvin-Helmholtz instability
  - formation of waves and vortices at fluid interfaces
  - simulating macroscopic process with with atomistic calculation !!!
- Hardware Fault tolerant MD - parity error recovery method
- 2-62.5 billion atoms
- Running on BG/L with up to 212,992 CPUs
- 103.9 Tflops sustained (115.1 Tflops computing performance)



# Kelvin-Helmholtz instability



# ddcMD: Simulation details from User

---

## Initial estimate of data requirements

- Each restart file requires 4.5 TB of storage.
- Each bxyz file requires 4.5 TB of storage.
- Each coarse grain file requires 16 GB of storage.
- Each atom subset file requires 4MB of storage.
  
- **We would like a restart interval somewhere in the range of 2-3 hours.**
- We should probably eliminate bxyz files for this run. Coarse grain files will be saved every 500-1000 time steps.
- **Hence our data rate will be approximately 1.6 TB per hour.**
  
- **At a restart interval of 3 hours and a write speed of 10 GB/sec our I/O would cost 5% of total run time.**



# ddcMD: Lustre Issues

- **Currently ddcMD read/writes data to/from Lustre at roughly 2 GB/sec.**
- At this rate it would take more than 30 minutes (20% of wall clock) to read or write a restart file.
  - This is not acceptable and we will need to increase our I/O rate.
- **Our goal is at least 10 GB/sec. To hit this mark we will**
  - **Write one file per I/O node (1664 files) per restart.**
  - **Ensure writer tasks are matched one-to-one with I/O nodes.**
  - **Stripe files 1 OST per file.**
  - **Write large blocks of data (10's to 100's of Mb per fwrite).**
- Exclusive access to a file system could help performance.
  - Perhaps we can mount lscratch1 or lscratch2 read only to all machines except BG/L. **(YES, lscratch2 was dedicated to team)**
  - Using both lscratch1 and lscratch2 could potentially double bandwidth. **(NO, we have other users.)**



# ddcMD: HPSS Issues (strategy to archive data)

- Experience has shown that we can move data from Lustre to storage at roughly 100MB/sec using a single slic node.
- We will use nft for our transfers so we expect the load balancing features of nft to use all 10 slic nodes and achieve 1GB/sec or 3.6TB/hr transfer rates.
- **This exceeds our data rate by a considerable margin so we should be ok.**
- Our plan calls for a restart "file" to consist of 1664 files (one per I/O node) with each file roughly 2.7 GB in size.
- The number of files in a coarse grain files isn't yet known. Using 64 files should in principle allow an I/O node to cache the entire write. Each file would then be 250MB in size.
- Subset files will probably be written as multiple small files, but post-processed into a single file.



## Customer Success Story: Conclusion

- **It all worked!**
- ddcMD achieved 18 GB/sec on Lustre, the best we know of for a real application, and also happened to win the 2007 Gordon Bell award
- From left to right, Jim Glosli, Fred Streitz, Robert Rudd, David Richards and Kyle Caspersen (IBM's John Gunnels is not pictured)



# Final remarks

---

- Focused on heroic effort on ddcMD I/O
- ddcMD efforts and successes of comparable magnitude on many fronts:
  - Simulation algorithms
  - Fault tolerance
  - Archiving
  - Visualization
- Lustre team perspectives
  - Long working relationship with code team
  - Supporting every day and extreme requirements
  - ***Looking forward to the next order of magnitude increase in compute and I/O capability***

