SciDAC PDSI Update (part 2)
CS/VIS PI Meeting, October 23, Germantown, MD

Garth Gibson
Carnegie Mellon University and Panasas Inc.

SciDAC Petascale Data Storage Institute (PDSI)
www.pdsi-scidac.org

w/ LANL (G. Grider), LBNL (W. Kramer), SNL (L. Ward),
ORNLS (P. Roth), PNNL (E. Felix),
UCSC (D. Long), U.Mich (P. Honeyman)
Clearing Path thru Petascale to Exascale

- Scaling 100%/yr given disk realities is hard
  - Disk BW @ 20%/yr, IO/s @ 5%/yr
  - Storage problem renews itself each year

Everything Must Scale with Compute

- Memory TeraBytes: 2,500
- Computing Speed: 15,000 TFLOP/s
- Disk PetaBytes: 50
- Parallel I/O 5,000 GigaBytes/sec
- Network Speed Gigabits/sec: 500
- Archival Storage GigaBytes/sec: 500
- Metadata Inserts/sec: 20,000
- Application Performance: 200
- Year 2012

Projected Performance Development

- 2.5 TFLOP/s in 2003
- 11.7 TFLOPs in 2011
- 102.6 TFLOps in 2012
- 890.6 GFLOps in 2013

Roadrunner
First to break the "petaflop" barrier

Carnegie Mellon
Parallel Data Laboratory
www.pdsi-scidac.org
• PETASCALE DATA STORAGE INSTITUTE 06-11
  • 3 universities, 5 labs, G. Gibson, CMU, PI
  • Enabling HEC storage to meet SciDAC needs
• SciDAC @ Petascale storage issues
  • Community building: ie. PDSW @ SCxy
  • APIs & standards: ie., Parallel NFS, POSIX
  • Failure data collection, analysis: ie., cfdr.usenix.org
  • Performance trace collection & benchmark publication
  • IT automation applied to HEC systems & problems
  • Novel mechanisms for core (esp. metadata, wide area)
Annual PDSI Sponsored Workshops

HEF FSIO '07
HEF FSIO R&D Workshop/HECURA FSIO PI Meeting '07 AGENDA

Workshop Location: National I.
Session:
Monday 6/6/2007
Welcome Review of HEF FSIO 06 successes F 2007 Workshop Overview
Welcome from NSF
NSF Vision
Research Session 1: QS
Quality of Service Guarantee for Scalable File Systems
Parallel Processing Technologies
Parallel Processing and Storage Systems
End-to-End Performance Management for Large Distributed Storage
Open forum of applications
LFS, ISCBM, and DETP
LFS, New Data Analysis
Research Session 2: Understanding, Interfacing, Cache, Hw
File System Design, Relaying, Profiling, and Analysis on HPC
Memory caching and prefetching
Open forum of applications
Research Session 3: HCC
Peta-scale I/O for High-End Computing
Techniques for Streaming File Systems and Databases
Interconnect Storage Systems for High-End Computing
SMASH Tools: Scalable and Adaptive Metadata Management for High-End Computing
Open review of progress
Research Session 4: Security and Archiving
Assessment in Performance and Security Requirements for I/O in HEF
Integrated Infrastructure for Security and Efficient Long-Term Data Management
Open review of progress
Posters for All Day 1 talks
Tuesday 6/7/2007
Use of KVM for Testing File Systems at Scale
Research Session 5: Next Generation I/O Architectures
Demonstrating Clusters for High End Biomed

Supercomputing '07
Petascale Data Storage Workshop
Session Chair: Garth Gibson, CMU
Sunday, November 11, 2007
Reno, Nevada

WORKSHOP ABSTRACT
Petascale computing infrastructures make petascale demands on information and management. The last decade has shown that parallel file systems are not sufficient: this poses a critical challenge when near-future petascale data storage problems and emerging solutions found in petascale storage will see new community collaboration can be crucial. Problem identification, workload analysis, and designs of shared tools.

Petascale Data Storage Workshop Introduction
Garth Gibson

SESSION I: Scalable Systems

SESSION II: Scalable Services
Jonathan Koren (presenter), E. Zhang, Univ. of California, Santa Cruz

SESSION III: Scalable Performance
T. Wang, M. McFall, Univ. of California, Santa Cruz

POSTER SESSION 1 - see info below

POSTER SESSION II: Scalable Services
Sarah W. H. Lee, Carnegie Mellon University

POSTER SESSION III: Scalable Performance
Aziz A. Yalda, Carnegie Mellon University

POSTER SESSION IV: Scalable Systems
Brent Wehr, Pennsylvania State University

FAST '08
Wednesday, February 27, 2008
Petascale Data Storage BOF Session at FAST '08

Organizer: Garth Gibson, Carnegie Mellon University and Panasas
Co-organizers: Peter Hrynevych, U. Michigan, OTL; Daniel Long, U.C. Santa Cruz; Gary Grider, Los Alamos NL; Lee Ward, Sandia NL; Evan Fixel, Pacific Northwest NL; Phil Roth, Oak Ridge NL; BIF Kramer, Lawrence Berkeley NL

The Petascale Data Storage Institute is a DOE-funded collaboration of three universities and five national labs with the objective of anticipating the challenges of data storage for computing systems operating in the petascale regime per second to exa-operations per second and working toward the resolution of these challenges in the community as a whole. An important part of our agenda is outreach to other researchers and practitioners to share our resources and gather better understanding of the petascale issues at issue from all.

In this BOF we will:
1. Introduce the Petascale Data Storage Institute (PDSI).
2. Advertise PDSI gathered and released sources of useful data, including
   - data sets of node and storage failures in large scale computing
   - file access times of non-trivial petascale computing applications
   - collections of file systems statistics gathered from petascale computing systems and other systems
3. Discuss requirements for one or more petascale data storage systems and applications.
4. Lead an open discussion of these and other issues for large scale data storage systems.

PRESENTATIONS
PDSI FAST 08 BOF Introduction - Garth Gibson, CMU
The Computer Failure Data Repository (CFDR) - Bianca Schroeder, University of Toronto
File System Statistics - Shohini Dey, CMU, Garth Gibson, CMU, Merc Unangst, Panasas
PPNL - Petascale Data Storage Institute Data release Update - Evan Fixel, PNNL
NERSC Reliability Data - Bill Kramer, Jason Hick, Akbar Mokhtarzadeh, NERSC
LANL SuDAC Petascale Data Storage Institute Operational Data Release - James Nunez, Gary Grider, John Bent, HB Chen, Megan Quinn, Aziz Toma, Los Alamos National Lab
Cegeh: An Open-Source Petabyte-Scale File System - Ethan Miller, Storage Systems Research Center, UC Santa Cruz

Special Presentation on NPC User Requirements:
IG Requirements for HPC Applications: A User Perspective
John Shalt, National Energy Research Scientific Computing Center (NERSC), LBNL

PDSI POSTER AT THE FAST '08 POSTER SESSION
PDSI Data Releases and Repositories

February
PDSW07 papers published & online

THE ACM DIGITAL LIBRARY

Conference on High Performance Networking and Computing

Proceedings of the 2nd international workshop on Petascale data storage

2007, Reno, Nevada November 11 - 11, 2007

Additional Information: full citation

Conference Chair: Garth A. Gibson Carnegie Mellon University and Panasas Inc.

Front matter

Table of Contents

SESSION: Scalable systems

On application-level approaches to avoiding TCP throughput collapse
Elie Krevat, Vijay Vasudevan, Amar Phanishayee, David G. Andersen
Pages 1-4
Full text available: pdf (115 KB)
Additional Information: full citation, abstract, references, index terms

pNFS/PVFS2 over InfiniBand: early experiences
Lei Chai, Xiangyong Cuiyang, Ranjit Noronha, Dhabaleswar K. Panda
Pages 5-11
Full text available: pdf (128 KB)
Additional Information: full citation, abstract, references, index terms

Integrated system models for reliable petascale storage systems
Brent Welch
Pages 12-16
Full text available: pdf (105 KB)
Additional Information: full citation, abstract, references, index terms

Scalable locking and recovery for network file systems
Peter J. Braam
Pages 17-20
Full text available: pdf (317 KB)
Additional Information: full citation, abstract, index terms

SESSION: Scalable services

SESSION I: Scalable Systems

9:00am - 10:20am

E. Krevat (presenter), V. Vasudevan, A. Phanishayee, D. Andersen, G. Ganger, G. Gibson, S. Seshan, Carnegie Mellon University
On Application-level Approaches to Avoiding TCP Throughput Collapse in Cluster-Based Storage Systems
Paper / Slides / Poster

Lei Chai, Xiangyong Cuiyang, Ranjit Noronha (presenter) and Dhabaleswar K. Panda, Ohio State University
pNFS/PVFS2 over InfiniBand: Early Experiences
Paper / Slides

Brent Welch (presenter), Panasas, Inc.
Integrated System Models for Reliable Petascale Storage Systems
Paper / Slides

Peter Braam, Byron Neitzel (presenter), Sun/Cluster File Systems
Scalable Locking and Recovery for Network File Systems
Paper / Slides

10:30am - 11:00am

POSTER SESSION 1 - see info below

SESSION II: Scalable Services

11:00am - 12:20pm

Jonathan Koren (presenter), Yi Zhang, Sasha Ames, Andrew Leung, Carlos Maltzahn, Ethan Miller, Univ. of California, Santa Cruz
Searching and Navigating Petabyte Scale File Systems Based on Facets
Paper / Slides

Swapnil V. Patil (presenter), Garth A. Gibson, Sam Lang, Milo Poiie, Carnegie Mellon University
GIGA+: Scalable Directories for Shared File Systems
Paper / Slides / Poster

D. Bigelow, S. Iyer, T. Kaldewey, R. Pineiro, A. Povzner, S. Brandt, R. Golding (presenter), T. Wong, C. Maltzahn, Univ. of California, Santa Cruz, IBM-Almaden
End-to-end Performance Management for Scalable Distributed Storage
Paper / Slides

Carnegie Mellon
Parallel Data Laboratory

www.pdsi-scidac.org
Petascale Data Storage Workshop 08

• Monday Nov 17, 8:30-5, room 14, SC08
• www.pdsi-scidac.org/events/PDSW08
• IEEE Digital Library publication
• Tentative program
  • Input/Output APIs and Data Organization for High Performance Scientific Computing
  • Fast log-based concurrent writing of checkpoints
  • Scalable Full-Text Search for Petascale File Systems
  • Zest: Reliable Terabytes Per Second Storage for Petascale Systems
  • Performance of RDMA-capable Storage Protocols on Wide-Area Network
  • Comparing performance of solid state devices and mechanical disks
  • Arbitrary Dimension Reed-Solomon Coding & Decoding for Extended RAID on GPUs
  • Pianola: A script-based I/O benchmark
  • Introducing Map-Reduce to High End Computing
  • Logan: Automatic Management for Evolvable, Large-Scale, Archival Storage
  • Just-in-time Staging of Large Input Data for Supercomputing Jobs
  • Revisiting the Metadata Architecture of Parallel File Systems
pNFS: Scalable NFS Standard & Code Soon

- Open source & competitive offerings!
- NetApp, Sun, IBM, EMC, Panasas ....

From: Tigran Mkrtchyan <tigran.mkrtchyan@desy.de>
Date: July 16, 2008 4:18:13 AM PDT
To: pnfs@linux-nfs.org
Subject: [pnfs] pnfs becomes real!

today we ran the first real physics analysis job using dCache-pnfs server and linux pnfs client:

tigran@nairi:/work/linux-pnfs> git show --oneline 6aa5e52464ba2c77f1b2f365e415305d19b51dd20
Author: Benny Halevy <bhalevy@panasas.com>
Date: Tue Jul 15 20:22:51 2008 +0300

Anyway first time we can show that NFSv4.1 is something real (and not my hobby only).

From: Spencer Shepler <Spencer.Shepler@Sun.COM>
Date: August 1, 2008 4:34:46 PM GMT-04:00

2. IETF status

All of the current working group internet drafts are moving forward for publication. This means that they have submitted to the area director and will start their way through the process (IETF last call and IESG review).

- SC-08 BOF Wed Nov 19 5:30pm
Tools for Understanding IO in Apps

**NEWEST TRACE DATA, REDSTORM, SANDIA NAT’L LAB**
- A physics simulation problem for a common Sandia application, Alegra
- Runs were performed alongside regular user runs
- Each run generated 4 restart dumps, and ran for 20 simulation cycles
- Both single core per node, and 2 core (virtual node mode) per node
  - Repeated with and without tracing enabled
- The single core per node jobs ran at a client size of 2744 processes
  - Non-tracing elapsed run time 10:42 minutes
  - Tracing elapsed run time 11:07 minutes
- The 2 core per node jobs ran at 2916 nodes, 5832 processes.
  - Non-tracing elapsed run time 15:52 minutes
  - Tracing elapsed run time 16:37 minutes
- Raw trace file sizes 30K-50K per MPI rank, except rank zero (600KB-700KB)
  - Rank 0 I/O to terminal records progress in the job.

sourceforge.net/projects/libsysio
PDSI distributes parallel workloads

**NEWEST TRACE DATA, REDSTORM, SANDIA NAT’L LAB**

- A physics simulation problem for a common Sandia application, Alegra
- Runs were performed alongside regular user runs
- Each run generated 4 restart dumps, and ran for 20 simulation cycles
- Both single core per node, and 2 core (virtual node mode) per node
  - Repeated with and without tracing enabled
- The single core per node jobs ran at a client size of 2744 processes
  - Non-tracing elapsed run time 10:42 minutes
  - Tracing elapsed run time 11:07 minutes
- The 2 core per node jobs ran at 2916 nodes, 5832 processes.
  - Non-tracing elapsed run time 15:52 minutes
  - Tracing elapsed run time 16:37 minutes
- Raw trace file sizes 30K–50K per MPI rank, except rank zero (600KB–700KB)
  - Rank 0 I/O to terminal records progress in the job.

sourceforge.net/projects/libsysio
PDSI distributes parallel workloads

**MPI-IO Test**

Although there are a host of existing file system and I/O test programs, most are not designed with parallel I/O in mind and are not useful at the clusters at Los Alamos National Lab (LANL). LANL’s MPI-IO Test was built with parallel I/O and scale in mind. The MPI-IO test is built on top of MPICH2 and is used to gather timing information for reading from and writing to files using a variety of I/O profiles; N processes writing to N files, N processes writing to one file, N processes sending data to M processes writing to M files, and N processes sending data to M processes to one file. These diagrams illustrate various I/O access patterns. A data aggregation capability is available and can pass down MPI-IO, ROMIO and file system specific hints. The MPI-IO test should be used for performance benchmarking and, in some cases, to diagnose problems with file systems or I/O networks.

The MPI-IO Test is open sourced under LA-CC-05-013.

### MPI.IO TEST traces

These traces were collected using LANL-Trace (V 1.0.0) on the LANL MPI.IO test (V 1.00.020) application. These traces are all from system data machine number 25 on this computer systems table. Here is the README and FAQ that explains how LANL-Trace works and what the output files look like:

TRACE README,
TRACE FAQ.

#### N-to-N

<table>
<thead>
<tr>
<th>Release</th>
<th>Date</th>
<th>Source</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000.21</td>
<td>8 July 2008</td>
<td>mpi_io_test_21.tgz</td>
<td>README</td>
</tr>
<tr>
<td>1.000.20</td>
<td>13 November 2007</td>
<td>mpi_io_test_20.tgz</td>
<td>README</td>
</tr>
<tr>
<td>1.000.09</td>
<td>15 December 2006</td>
<td>mpi_io_test_09.tgz</td>
<td>README</td>
</tr>
<tr>
<td>1.000.08</td>
<td>2 March 2006</td>
<td>mpi_io_test_08.tgz</td>
<td>README</td>
</tr>
</tbody>
</table>
PDSI distributes parallel workloads

**MADBench**: Microwave Anisotropy Dataset Computational Analysis Package Benchmark
The benchmark code MADBench is a "stripped-down" version of MADCAP, a Microwave Anisotropy Dataset Computational Analysis Package ...

IPM benchmarks: Medium, Large and X-large datasets.

**MILC**: MIND Lattice Computation
The benchmark code MILC represents part of a set of codes written by the MIND Lattice Computation (MILC) collaboration used to study chromodynamics (QCD), the theory of the strong interactions of subatomic physics ...

IPM benchmarks: Medium and Large datasets.

**PMEMD**: Particle Mesh Ewald Molecular Dynamics
The benchmark code PMEMD (Particle Mesh Ewald Molecular Dynamics (MD), NMR Refinement and minimizations ...

IPM benchmarks: Medium and Large datasets.

**IO Benchmarks with IPM**
The new version of IPM integrates the standard POSIX IO calls. All runs are made with this new feature on Jacquard (courtesy of IBM).  

**MADBench**:
- 256 tasks, POSIX one file per task [plots] [stats]
- 64 tasks, POSIX one file per task [plots] [stats]
- 16 tasks, POSIX shared file [plots] [stats]

**Chombo**:
- 256 tasks, 2 components [plots] [stats]
- 32 tasks, 2 components [plots] [stats]
- 32 tasks, 10 components [plots] [stats]

**AMRScalingXfer**: 128 tasks, small run [plots] [stats]

*Note: This is development software, and the runs/plots are not profiling in IPM.

**Trace Data**
Here are files containing trace data for some of the applications. These traces are generated by invoking the "trace" utility on every task and piping the data for each task to a separate file. Process ID is used to create unique file names. All applications where run on Jacquard.

- **PMEMD**: 16 tasks small dataset run
- **MADBench**: 64 tasks medium dataset run
- **MILC**: 16 tasks medium dataset run

---

**I/O Benchmark and Characterization Links**:

**I/O Performance for HPC Platform using IOR PDF**
This study analyzes the I/O practices and requirements of current HPC applications and use them as criteria to select a subset of microbenchmarks that reflect workload requirements.

**FLASH I/O Benchmark PDF**
This code from 'The Center for Astrophysical Thermonuclear Flashes' can test either HDFS, Parallel NetCDF, or a direct Fortran write. The I/O benchmarks are compared for Seaborg and Bassi systems.

**Performance Effect of Multi-core on Scientific Applications (PDF) paper slides**
Prepares performance measurements of several complete scientific applications on single and dual core Cray XT3 and XT4 systems.

**MADBench - IPM of a Cosmology Application on Leading HEC Platforms PDF**
Prepares MADBench, a lightweight version of MADCAP CMB power spectrum estimation code, and uses the Integrated Performance Monitoring (IPM) package to extract MPI message-passing overheads

**MADBench2 PDF**
Prepares I/O analyses of modern parallel filesystems and examines a broad range of system architectures and configurations. It also describes use of Luster striping to improve concurrent file access performance.

**Effective I/O Bandwidth Benchmark PDF**
This paper describes the design and implementation of a parallel I/O benchmark useful for comparing filesystem performance on a variety of architectures, including, but not limited to cluster systems.

**Efficient Parallel I/O on the Cray XT3/XT4 PDF**
Provides an overview of I/O methods for three different applications

---

---

---
PDSI distributes convenient packages

How do I get pvfs installed on my system?

Follow the instructions above for installing the pdsi ubuntu repository on your ubuntu system. Then there are several packages to choose from to install pvfs depending on what you want to do with the box. Currently pvfs isn't built to use infiniband networking, however, you can run ip over infiniband and use these packages.

- karma - gks gui for looking at a pvfs filesystem graphically
- llbpfvs - library for pvfs
- llbpfvs-dev - libraries and headers for developing with pvfs
- pvfs-fs-utils - pvfs file system utils for talking directly to pvfs
- pvfs-server - pvfs server binary
- pvfs-server-utils - pvfs helper utils for generating configs, ping/en the servers, etc.
- pvfs-source - source for the pvfs client driver
- pvfs-modules-2.6.21-1-686 - pvfs modules and client for Linux (kernel 2.6.21-1-686)

Then simply:

```bash
apt-get install pvfs-server pvfs-modules-2.6.21-1-686
```

NOTE: The architecture used in this example is a 686 compatible box. So if you are using amd64 or ia64 please replace 686 with the appropriate architecture.

How do I get lustre installed on my system?

Follow the instructions above for installing or for unstable for installing the pdsi ubuntu repository on your ubuntu system. Then there are several packages to choose from to install lustre depending on what you want to do with the box. Currently lustre isn't built to use infiniband or quadrics, however you can run IP over both quadrics and infiniband and still use these packages.

- linux-doc-2.6.18-4-lustre-686 - linux kernel specific documentation for version 2.6.18-4-lustre-686
- linux-headers-2.6.18-4-lustre-686 - common header files for linux 2.6.18-4-lustre on 686
- linux-image-2.6.18-4-lustre-686 - linux 2.6.18-4-lustre image on 686
- linux-manual-2.6.18-4-lustre-686 - linux kernel section 9 manual pages for version 2.6.18-4-lustre
- linux-patch-lustre - linux kernel patch for the lustre filesystem
- linux-source-2.6.18-4-lustre-686 - linux kernel source for version 2.6.18 with debian and lustre patches
- lustre-dov - development files for the Lustre filesystem
- lustre-modules-2.6.18-4-lustre-686 - lustre filesystem driver modules for linux 2.6.18 on 686
- lustre-source - source for lustre filesystem client kernel modules
- lustre-utils - userspace utilities for the lustre filesystem
- liblustre - liblustre library for the lustre filesystem

Then simply:

```bash
apt-get install linux-image-2.6.18-4-lustre-686 lustre-utils lustre-modules-2.6.18-4-lustre-686
```
Newest: File Statistics
### Results

<table>
<thead>
<tr>
<th>Uploaded File in CSV format</th>
<th>Organization</th>
<th>Date</th>
<th>Data Size</th>
<th>System Name</th>
<th>Form Questions</th>
<th>Formatted Result</th>
<th>Graphs</th>
<th>Graphs</th>
<th>fsstats Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>BradHavel_nano1.csv</td>
<td>ARSC</td>
<td>Mar12008 69TB</td>
<td>SAMQFS</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>BradHavel_sea2.csv</td>
<td>ARSC</td>
<td>Mar13008 115TB</td>
<td>SAMQFS</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>BradHavel_sea1.csv</td>
<td>ARSC</td>
<td>Mar13008 305TB</td>
<td>SAMQFS</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>BradHavel_nbprojects.csv</td>
<td>ARSC</td>
<td>Mar13008 32TB</td>
<td>SAMQFS</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>JamesNunez_panscratch1.csv</td>
<td>LANL</td>
<td>Apr012008 9TB</td>
<td>Panasas</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>JamesNunez_yellowscratch2.csv</td>
<td>LANL</td>
<td>Apr02008 25TB</td>
<td>Panasas</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>JamesNunez_yellowscratch3.csv</td>
<td>LANL</td>
<td>Apr042008 26TB</td>
<td>Panasas</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>AkbarMokhtarani_NGIPFsstats.csv</td>
<td>NERSC</td>
<td>Apr082008 107TB</td>
<td>GPFS</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>PhilRoth_fsstats.csv</td>
<td>ORNL</td>
<td>Oct102007 305GB</td>
<td>Panasas</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>MichaelStroucken_pdf2.csv</td>
<td>PDL</td>
<td>Apr092008 1TB</td>
<td>WAFL</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>MichaelStroucken_pdf1.csv</td>
<td>PDL</td>
<td>Apr092008 4TB</td>
<td>WAFL</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>EvanFelix_dtemp.csv</td>
<td>PNNL</td>
<td>Mar172008 23TB</td>
<td>Lustre</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>EvanFelix_rwfs.csv</td>
<td>PNNL</td>
<td>Mar172008 265TB</td>
<td>Lustre</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>EvanFelix_home.csv</td>
<td>PNNL</td>
<td>Mar172008 5TB</td>
<td>ADVFS</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>EvanFelixmpp2dtemp.csv</td>
<td>PNNL</td>
<td>Oct102007 12TB</td>
<td>ext3</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>EvanFelix_mpp2rwfs.csv</td>
<td>PNNL</td>
<td>Oct102007 23TB</td>
<td>ext3</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>EvanFelix_mpp2home.csv</td>
<td>PNNL</td>
<td>Oct102007 4TB</td>
<td>advs</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>Katie_scratch1.csv</td>
<td>PSC</td>
<td>Mar272008 32TB</td>
<td>Lustre</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>Katie_bessgmer1.csv</td>
<td>PSC</td>
<td>Mar272008 4TB</td>
<td>Lustre</td>
<td>Form</td>
<td>Histograms</td>
<td>PS</td>
<td>PDF</td>
<td></td>
<td>1.4</td>
</tr>
</tbody>
</table>

**Links**
- A Comparative graph of some of the above results EPS
- A Comparative graph of some of the above results PDF
- A Comparative graph of archival file system EPS
- A Comparative graph of archival file systems PDF

---

**Carnegie Mellon Parallel Data Laboratory**

www.pdsi-scidac.org
PDSI Targeted Apps

• P. Roth (ORNL) led
  • Parallel Ocean Program (POP) with PERI
  • Turbulent Combustion (S3D) with PERI
  • IO characterization and modeling
• L. Ward (SNL) led
  • Climate Change (CCSM) with M. Taylor
  • Trace-based performance debugging
• G. Gibson (CMU) starting
  • Astrophysics (Flash), reaching out to P. Hovland
Failure Data Collection

- Los Alamos root cause logs
  - 22 clusters & 5,000 nodes
  - covers 9 years & continues
  - cfdr.usenix.org publishes this and many other failure datasets

<table>
<thead>
<tr>
<th>HW</th>
<th>ID</th>
<th>Nodes</th>
<th>Procs</th>
<th>Procs/node</th>
<th>Production Time</th>
<th>Mem (GB)</th>
<th>NICs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>8</td>
<td>N/A - 12/99</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>1</td>
<td>32</td>
<td>32</td>
<td>N/A - 12/03</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>N/A - 04/01</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>164</td>
<td>328</td>
<td>2</td>
<td>04:01 - now</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>256</td>
<td>1024</td>
<td>4</td>
<td>12/01 - now</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>128</td>
<td>512</td>
<td>4</td>
<td>09/01 - 01/02</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>1024</td>
<td>4096</td>
<td>4</td>
<td>05/02 - now</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1024</td>
<td>4096</td>
<td>4</td>
<td>10/02 - now</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>128</td>
<td>512</td>
<td>4</td>
<td>09/03 - now</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>128</td>
<td>512</td>
<td>4</td>
<td>09/03 - now</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>128</td>
<td>512</td>
<td>4</td>
<td>09/03 - now</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>32</td>
<td>128</td>
<td>4</td>
<td>09/03 - now</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>128</td>
<td>256</td>
<td>2</td>
<td>09/03 - now</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>256</td>
<td>512</td>
<td>2</td>
<td>09/03 - now</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>256</td>
<td>512</td>
<td>2</td>
<td>09/03 - now</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>256</td>
<td>512</td>
<td>2</td>
<td>09/03 - now</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>256</td>
<td>512</td>
<td>2</td>
<td>09/03 - now</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>512</td>
<td>1024</td>
<td>2</td>
<td>09/03 - now</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>03/05 - 06/05</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>03/05 - 06/05</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>16</td>
<td>2048</td>
<td>128</td>
<td>12/96 - 09/02</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
<td>12/96 - 09/02</td>
<td>64</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
<td>12/96 - 09/02</td>
<td>64</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>49</td>
<td>6152</td>
<td>128</td>
<td>01/97 - now</td>
<td>128</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
<td>01/97 - 11/05</td>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>80</td>
<td>06/05 - now</td>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
<td>10/08 - 12/04</td>
<td>128</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
<td>11/02 - now</td>
<td>64</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
<td>11/05 - 12/04</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>5</td>
<td>544</td>
<td>32</td>
<td>01/98 - 12/04</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
<td>11/02 - now</td>
<td>64</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
<td>11/05 - 12/04</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>H</td>
<td>22</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>11/04 - now</td>
<td>1024</td>
<td>0</td>
</tr>
</tbody>
</table>
Revisiting checkpoint: Log representation

- Fastest checkpoint just a series of “variable=value” (ie. PSC Zest)
  - Instead of seeking to serialized location, just append operation to log
  - Each thread writes strictly sequential log of operations
  - “Meaning” of set of logs is applying log to (possibly null) initial database
- Prior: Gatech/ORNL ADIOS, ANL summer project
  - Discussing with SciDAC SDM on application to pHDF5/netCDF
- Embed in storage software as general, transparent service
  - Optimize writing and reading representations separately
  - Defer serializing by just storing changelogs for later application
  - Some checkpoints never read, so never serialized
  - If read before serialized, trigger serialization (or something smarter)
- Opens up embedded indices (FastBit, B-trees, database)
CMU class project: log-structured PVFS files

- HPC checkpoints
  - AMR apps are non-sequential concurrent writers
    - Lousy BW
  - Store file as log of writes
    - Good BW

N-to-1 strided example

Each process writes each element in a single shared "stride" within a single shared file. The file consists of one region per element (not one region per process as in N-1 non-strided). Each region contains "strided" data from each process.

(c) Write Bandwidth of modified and unmodified PVFS with various numbers of clients and block sizes

- Group 8 mpi-io write test (from LANL)
  - S. Dayal, M. Chainani, D.K. Uppugandi, W. Tantosiriroj
## Storage suffers failures too

<table>
<thead>
<tr>
<th>Pittsburgh Supercomputing Center</th>
<th>Type of drive</th>
<th>Count</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HPC1</strong></td>
<td>18GB 10K RPM SCSI</td>
<td>3,400</td>
<td>5 yrs</td>
</tr>
<tr>
<td></td>
<td>36GB 10K RPM SCSI</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HPC2</strong></td>
<td>36GB 10K RPM SCSI</td>
<td>520</td>
<td>2.5 yrs</td>
</tr>
<tr>
<td><strong>HPC3</strong></td>
<td>15K RPM SCSI</td>
<td>14,208</td>
<td>1 yr</td>
</tr>
<tr>
<td></td>
<td>15K RPM SCSI</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.2K RPM SATA</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HPC4</strong></td>
<td>250GB SATA</td>
<td>13,634</td>
<td>3 yrs</td>
</tr>
<tr>
<td></td>
<td>500GB SATA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>400GB SATA</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Various HPCs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>COM1</strong></td>
<td>10K RPM SCSI</td>
<td>26,734</td>
<td>1 month</td>
</tr>
<tr>
<td><strong>COM2</strong></td>
<td>15K RPM SCSI</td>
<td>39,039</td>
<td>1.5 yrs</td>
</tr>
<tr>
<td><strong>COM3</strong></td>
<td>10K RPM FC-AL</td>
<td>3,700</td>
<td>1 yr</td>
</tr>
<tr>
<td></td>
<td>10K RPM FC-AL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10K RPM FC-AL</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Internet services Y</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Storage failure especially painful

- Scalable performance = more disks
- But disks are getting bigger
- Recovery per failure increasing
- Hours to days on disk arrays
- Consider # concurrent disk recoveries
  e.g. 10,000 disks
  3%/year replacement rate
  1+ day recovery each
  Constant state of recovery?
- Maybe soon 100s of concurrent recoveries (at all times?)
- Design normal case
  for many failures (huge change!)
Object storage & scalable repair

- Defer the problem with parallel scalable repair
- File replication and, more recently, object RAID can scale repair - “decluster” redundancy groups over all disks (mirror or RAID)
  - use all disks for every repair, faster is less vulnerable
- Object (chunk of a file) storage architecture dominating at scale
  GFS, HDFS, … PanFS, Lustre, PVFS, … Centera, …
Developing reliable, evolvable Archives

- Evolvable, distributed network of intelligent, disk-based tomes
  - Smart enough to function independently
  - Provide inter-disk redundancy
  - Building blocks for more complex systems
  - Evolve over time: integrate new technologies
- Handle errors at multiple levels
  - Scale response to size of problem
  - Very high reliability!
- Control costs
  - Commodity low-power hardware
    - Keep disks spun down
  - Standardized interfaces
Spyglass design

- Partition file system hierarchy by subtree
  - Each subtree is an independent subindex
- Summarize contents of each subindex
  - Quickly rule out entire subindexes that can’t satisfy the query
- Log incremental changes
  - Rebuild index when there are “enough” changes
- Integrity is much easier
  - Rebuild subindex, not entire index
Use cases for huge directories

• Apps use FS as fast, lightweight database
  • Use case: All clients inserting millions of small files in a single directory as fast as possible
  • Retain VFS API: create(), lookup(), readdir(), etc.

• Creating many small files in a “burst”
  • E.g., per-process checkpoint on large clusters
  • E.g., science experimental capture

• Creating many small files “steadily”
  • E.g., “log” files from long-running apps for later post-processing (history, bio device runs,...)

• Most interested in pushing the boundaries
Extendible Hashing [Fagin79]

Hash keys for load-balancing

\[
\text{hash(“F5”)} = 1001…011
\]

RADIX increases, that uses the growing table (\(R = 2\) bits)

• Header-table doubles, if necessary
  • On splitting, the new partitions distribute their keys
• Mechanism designed for single server impln.
GIGA+ Directories (PVFS, FUSE)

- Eliminate serialization
- All servers grow directory independently, in parallel, without any co-ordinator

Local representation of huge directory in Giga

- No synchronisation & consistency bottlenecks
  - Servers only keep local “view”, no shared state

Graph showing aggregate throughput vs. number of servers and different file systems.
Whither shared storage clusters?

• Contrasted to per-application/per-machine
  • sharing allows common namespace
  • sharing allows common provision+use of spare
    – including bursty usage

• But, interference can kill storage performance
  • Disk: “context switch” = mechanical seek (slow!)
  • Cache: what does time-sharing mean?
  • Cluster: coordinating timing across nodes
R-value quantizing of disk arm (2 apps)

- Workload 1 alone
- Workload 2 alone
- Combination (Ideal) R=0.9
- Combination (Unacceptable)
But, data will be striped over servers

- Data striped for performance (esp. bandwidth)
  - each client req. translates to multiple server accesses
  - client req. is “done” when all accesses are done
    - so, overall req. waits for the slowest one
  - unsynchronized quanta can lead to significant delays
    - so, need to coordinate quanta (a la synch spindles)
Promising initial results (seek intensive)

- Goal \( R = 0.9 \)
- Argon sync’d
- No Argon
- Argon unsync’d

- MB/s per server

Number of servers:
1 2 3

MB/s per server:
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6
Cosmology Simulations (A. Szalay, JHU)

Cosmological simulations have $10^9$ particles and produce over 30TB of data (Millennium, Aquarius, …)

- Build up dark matter halos
- Track merging history of halos
- Use it to assign star formation history
- Combination with spectral synthesis
- Realistic distribution of galaxy types

- Too few realizations (IO and storage limited)
- Hard to analyze the data afterwards ->need DB (Lemson)
- What is the best way to compare to real data?
Looking to another form of object storage

- HPC & Web search operate at similar scale
  - 10s of thousands of nodes & growing
  - Want to co-opt web effort/excitement for HPC

**The Google File System**

By Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung
(Presented at SOSP 2003)

**Motivational Facts**
- More than 15,000 commodity-class PC’s.
- Multiple clusters distributed worldwide.
- Thousands of queries served per second.
- One query reads 100’s of MB of data.
- One query consumes 10’s of billions of CPU cycles.
- Google stores dozens of copies of the entire Web!

**Conclusion:** Need large, distributed, highly fault-tolerant file system.
Recall pNFS: scalable NFS very soon

- Teach NFS to delegate file maps
  - Client directs parallel transfer
  - Scales bandwidth up
  - Scales metadata load down
- IETF standard near complete
  - Sun, NetApp, EMC, IBM, Panasas, BlueArc, etc
  - Open source Linux essential
    - Linux core team active
- Data servers can be clients too
  - Maps expose placement
  - Full, friendly, familiar, file systems 😊
What if PVFS Shim’d into Hadoop?

Hadoop/Mapreduce framework

Extensible file system API (org.apache.hadoop.fs.FileSystem)

Shim layer

PVFS library (talks to PVFS servers)

buf  map  rep

HDFS library

to HDFS nodes

to local disk

Local FS

Application

Application

Hadoop Internet services stack

PVFS library (talks to PVFS servers)

Wittawat Tantisiriroj © October 08
Modified PVFS

Distributed Grep (64GB over 32 nodes)

Completion Time (sec)

- Vanilla PVFS
- PVFS w/ Read-ahead
- PVFS w/ Read-ahead & File Layout
- HDFS
N clients writes to n distinct files

- Multiple copies requires HDFS and PVFS to perform more write operation
  - HPC file systems need to go this route for scalable rebuild
- HDFS writes the first copy locally (bad distribution)
  - A good trick only if real work is already subdivided
Concurrent writes to a single file

<table>
<thead>
<tr>
<th></th>
<th>HDFS</th>
<th>PVFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (MB/s)</td>
<td>24.6</td>
<td>105.5</td>
</tr>
<tr>
<td>Network Traffic In (GB)</td>
<td>49.7</td>
<td>59.0</td>
</tr>
<tr>
<td>Network Traffic Out (GB)</td>
<td>48.1</td>
<td>59.2</td>
</tr>
<tr>
<td>Completion Time (min:sec)</td>
<td>10:50</td>
<td>2:31</td>
</tr>
</tbody>
</table>

- PVFS enables concurrent writes to non-overlapping regions, so N clients, each can copy 1/N each.
- Without multiple writers to a file, HDFS can only go as fast as a single client can.
- Real issue is Internet service users have to play with data to store in lots of right-sized sub-files (ugh)
Test with Analytics benchmarks

- **Grep**: Search for a rare pattern in two million 100-bytes records
- **Sort**: Sort two million 100-bytes records
- **Never-Ending Language Learning (NELL)**: (from J. Betteridge) Count the numbers of selected phases in 37GB data-set
- **Page-Rank Application**: (from L. Zhao) Rank webpage by their reading difficulty level (aka. easy to read)
PVFS’s performance is similar to HDFS in read-intensive applications.
In write-intensive application, HDFS performs better because it writes the first copy locally.
Its All About Data, Scale & Failure (not cycles)

• Continual gathering of data on data storage
  – Failures, distributions, traces, workloads
• Nurturing of file systems to HPC scale, requirements
  – pNFS standards, benchmarks, testing clusters, academic codes
• Checkpoint specializations
  – App compressed state, special devices, special representations
• Failure as the normal case?
  – Risking 100s of concurrent disk rebuilds (need faster rebuild)
  – Quality of service (performance) during rebuild in design
• HPC vs Cloud Storage Architecture
  – Where is the storage? What traffic patterns? Common code?
• Correctness at increasing scale?
  – Testing using virtual machines to simulate larger machines
  – Formal verification of correctness (performance?) at scale
SciDAC PDSI Update (part 2)
CS/VIS PI Meeting, October 23, Germantown, MD

Garth Gibson
Carnegie Mellon University and Panasas Inc.

SciDAC Petascale Data Storage Institute (PDSI)

www.pdsi-scidac.org

w/ LANL (G. Grider), LBNL (W. Kramer), SNL (L. Ward),
ORNL (P. Roth), PNNL (E. Felix),
UCSC (D. Long), U.Mich (P. Honeyman)