Scalla/xrootd 2009 Developments

Andrew Hanushevsky
SLAC National Accelerator Laboratory
Stanford University
12-October-2009
CERN Update

http://xrootd.slac.stanford.edu/

Outline

- System Component Summary
- Recent Developments
- Scalability, Stability, & Performance
  - ATLAS Specific Performance Issues
- Faster I/O
  - The SSD Option
- Future Developments

SLAC NATIONAL ACCELERATOR LABORATORY

Recap Of The Components

- xrootd
  - Provides actual data access
- cmsd
  - Glues multiple xrootd’s into a cluster
- XrdCnsd
  - Glues multiple name spaces into one name space
- BeStMan
  - Provides SRM v2+ interface and functions
- FUSE
  - Exports xrootd as a file system for BeStMan
- GridFTP
  - Grid data access either via FUSE or POSIX Preload Library

SLAC NATIONAL ACCELERATOR LABORATORY

Recent 2009 Developments

- April: File Residency Manager (FRM)
- May: Torrent WAN transfers
- June: Auto summary monitoring data
- July: Ephemeral files
- August: Composite Name Space rewrite
  - Implementation of SSI (Simple Server Inventory)
- September: SSD Testing & Accommodation

File Residency Manager (FRM)

- Functional replacement for MPS[1] scripts
  - Currently, includes…
    - Pre-staging daemon **frm_pstgd** and agent **frm_pstga**
      - Distributed copy-in prioritized queue of requests
      - Can copy from any source using any transfer agent
      - Used to interface to real and virtual MSS's
    - **frm_admin** command
      - Audit, correct, and obtain space information
        - Space token names, utilization, etc.
      - Can run on a *live* system
  - Missing **frm_migr** and **frm_purge**

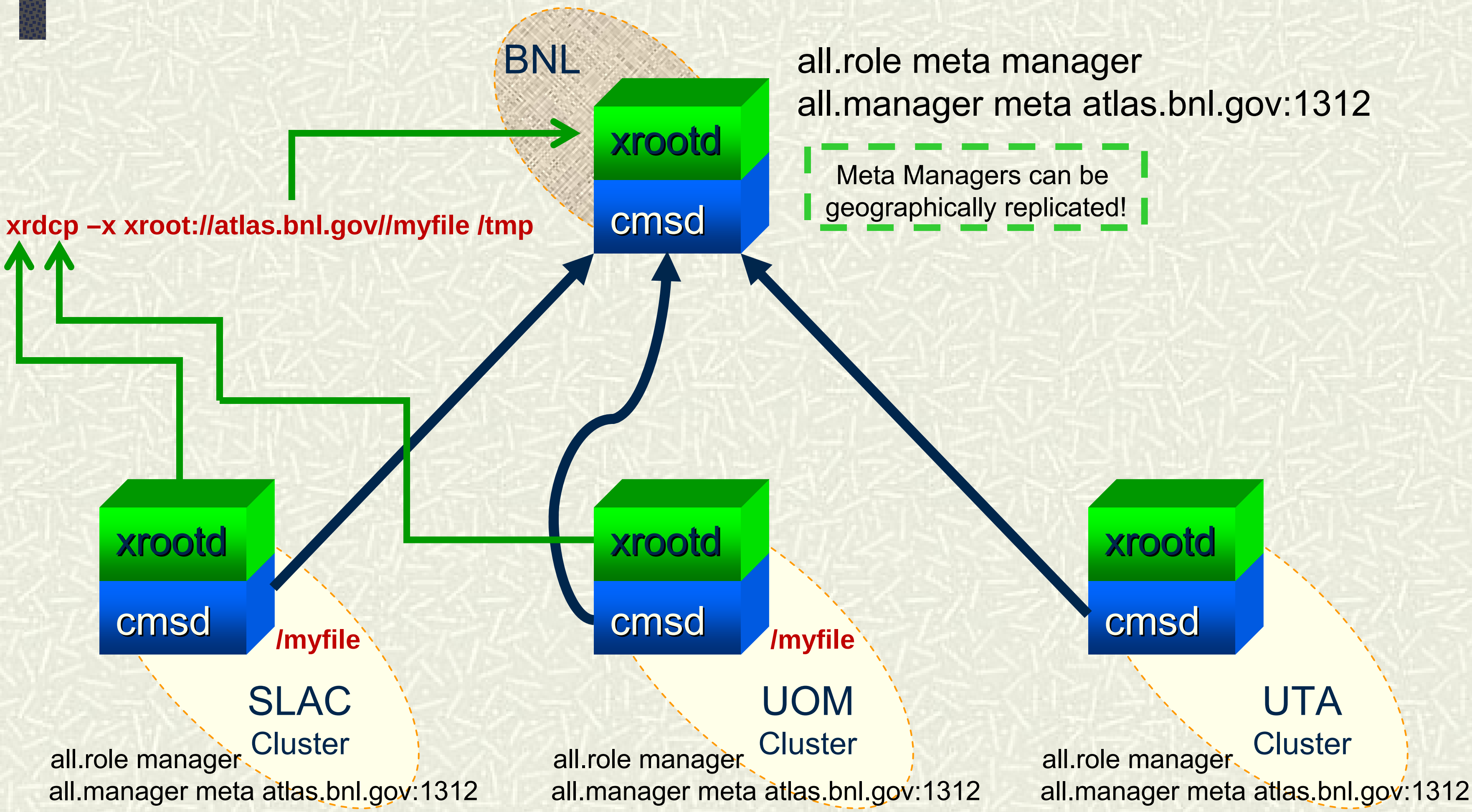
[1]Migration Purge Staging

SLAC NATIONAL ACCELERATOR LABORATORY

Torrent WAN Transfers

- The xrootd already supports parallel TCP paths
  - Significant improvement in WAN transfer rate
    - Specified as xrdcp –S *num*
- New Xtreme copy mode option
  - Uses multiple data sources bit torrent-style
    - Specified as xrdcp –x
  - Transfers to CERN; examples:
    - 1 source (.de): 12MB/sec (1 stream)
    - 1 source (.us): 19MB/sec (15 streams)
    - 4 sources (3 x .de + .ru): 27MB/sec (1 stream each)
    - 4 sources + || streams: 42MB/Sec (15 streams each)
    - 5 sources (3 x .de + .it + .ro): 54MB/Sec (15 streams each)

Torrents With Globalization



SLAC NATIONAL ACCELERATOR LABORATORY

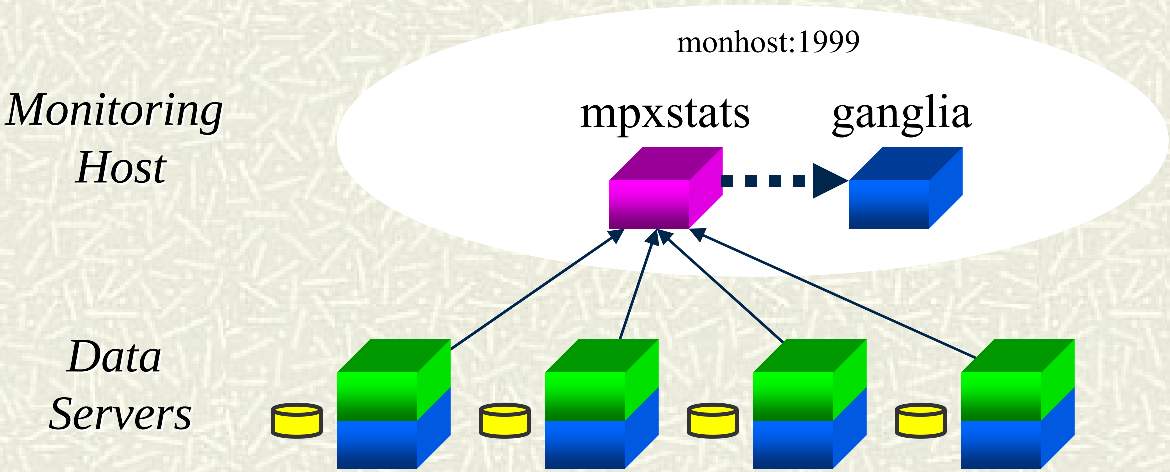
Manual Torrents

- Globalization simplifies torrents
  - All real-time accessible copies participate
    - Each contribution is relative to each file's transfer rate
- Will be implementing manual torrents
  - Broadens the scope of xrdcp
    - Though not as simple or reliable as global clusters
- xrdcp –x xroot://*host1*,*host2*,…/*path* . . .
  - Future extended syntax

Summary Monitoring

- xrootd has built-in summary & detail monitoring
- Can now auto-report summary statistics
  - Specify **xrd.report** configuration directive
- Data sent to one or two locations
  - Accommodates most current monitoring tools
    - Ganglia, GRIS, Nagios, MonALISA, and perhaps more
  - Requires external xml-to-monitor data convertor
    - Can use provided stream multiplexing and xml parsing tool
      - **mpxstats**
        - Outputs simple key-value pairs to feed a monitor script

Summary Monitoring Setup



xrd.report monhost:1999 all every 15s

SLAC NATIONAL ACCELERATOR LABORATORY

Ephemeral Files

- Files that persist only when successfully closed
  - Excellent safeguard against leaving partial files
    - Application, server, or network failures
      - E.g., GridFTP failures
  - Server provides grace period after failure
    - Allows application to complete creating the file
      - Normal xrootd error recovery protocol
      - Clients asking for read access are delayed
      - Clients asking for write access are usually denied
        - Obviously, original creator is allowed write access
  - Enabled via xrdcp **–P** option or **ofs.posc** CGI element

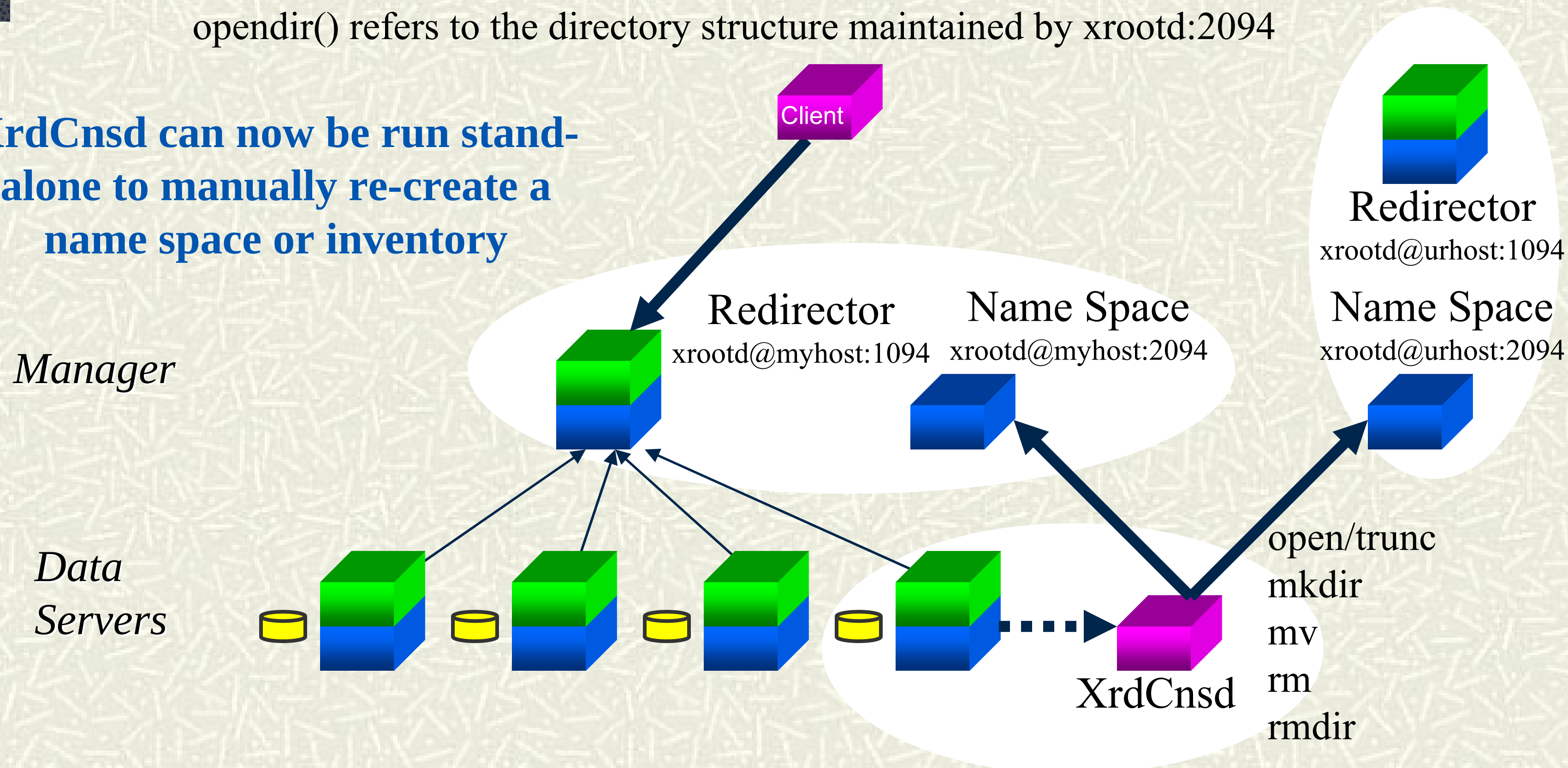
Composite Cluster Name Space

- Xrootd add-on to specifically accommodate *users* that *desire* a full name space "ls"
  - XrootdFS via FUSE
  - SRM
- Rewrite added two features
  - Name space replication
  - Simple Server Inventory (SSI)

Composite Cluster Name Space

opendir() refers to the directory structure maintained by xrootd:2094

XrdCnsd can now be run stand-alone to manually re-create a name space or inventory



ofs.notify closew, create, mkdir, mv, rm, rmdir |/opt/xrootd/etc/XrdCnsd

SLAC
NATIONAL ACCELERATOR LABORATORY

Replicated Name Space

- Resilient implementation
  - Variable rate rolling log files
    - Can withstand multiple redirector failures w/o data loss
      - Does not affect name space accuracy on working redirectors
- Log files used to capture server inventory
  - Inventory complete to within a specified window
- Name space and inventory logically tied
  - But can be physically distributed if desired

SLAC
NATIONAL ACCELERATOR LABORATORY

Simple Server Inventory (SSI)

- A central file inventory of each data server
  - Does *not* replace PQ2 tools (Neng Xu, Univerity of Wisconsin)
    - Good for uncomplicated sites needing a server inventory
  - Can be replicated or centralized
  - Automatically recreated when lost
    - Easy way to re-sync inventory and new redirectors
  - Space reduced flat ASCII text file format
    - LFN, Mode, Physical partition, Size, Space token

The cns_ssi Command

- Multi-function SSI tool
  - Applies server log files to an inventory file
    - Can be run as a cron job
  - Provides ls-type formatted display of inventory
    - Various options to list only desired information
  - Displays inventory & name space differences
    - Can be used as input to a “fix-it” script

Performance I

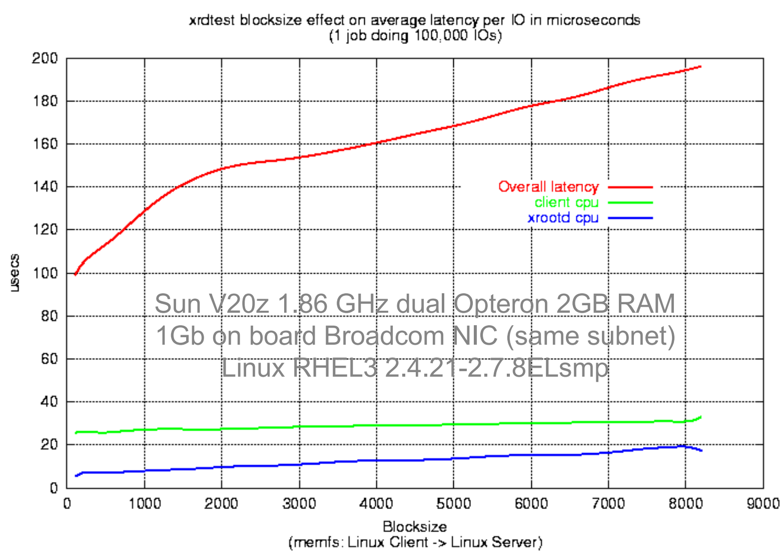
- Following figures are based on actual measurements
  - These have also been observed by many *production* sites
    - E.G., BNL, IN2P3, INFN, FZK, RAL , SLAC

CAVEAT!

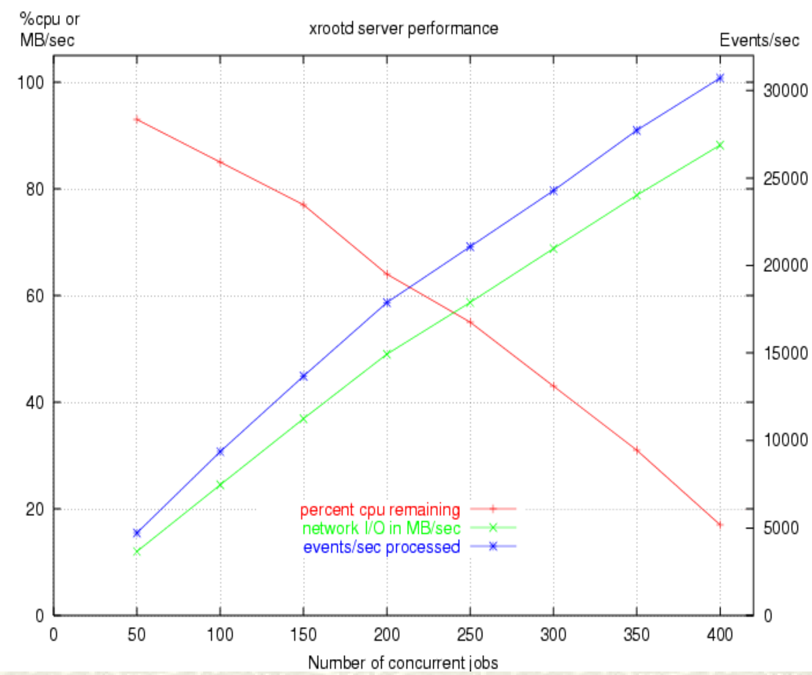
  - Figures apply only to the *reference* implementation
  - Other implementations vary significantly
    - Castor + xrootd protocol driver
    - dCache + native xrootd protocol implementation
    - DPM + xrootd protocol driver + cmsd XMI
    - HDFS + xrootd protocol driver

Performance II

Latency



Capacity vs. Load



xrootd latency < 10µs → network or disk latency dominates

Practically, at least ≈100,000 Ops/Second with linear scaling

xrootd+cmsd latency *(not shown)* 350µs →»2000 opens/second

Performance & Bottlenecks

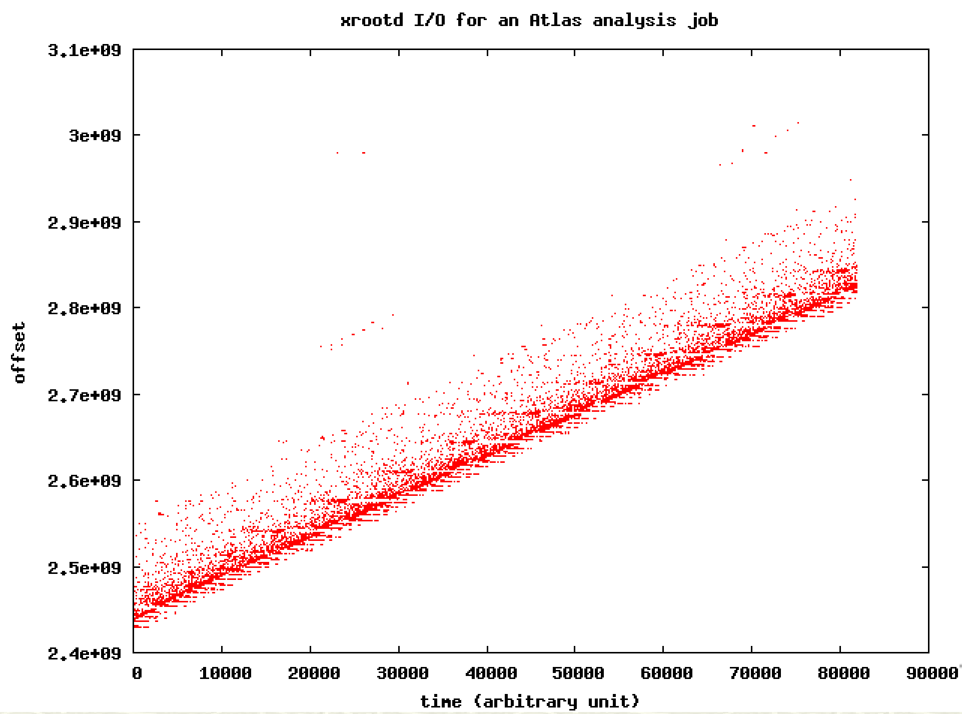
- High performance + linear scaling
  - Makes client/server software virtually transparent
    - A 50% faster xrootd yields 3% overall improvement
    - Disk subsystem and network become determinants
      - This is actually excellent for planning and funding

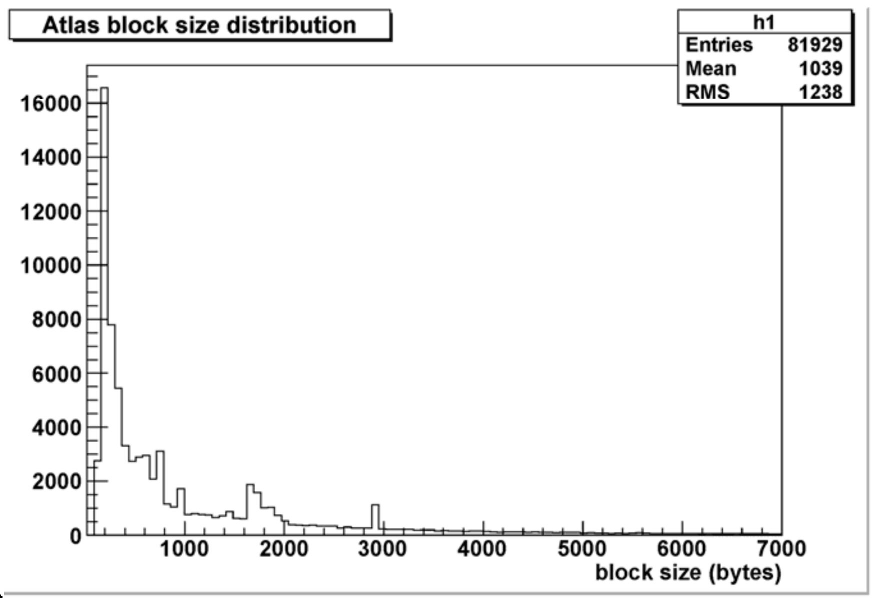
HOWEVER

  - Transparency makes other bottlenecks apparent
    - Hardware, Network, Filesystem, or Application
      - Requires deft trade-off between CPU & Storage resources
    - But, bottlenecks usually due to unruly applications
      - Such as ATLAS analysis

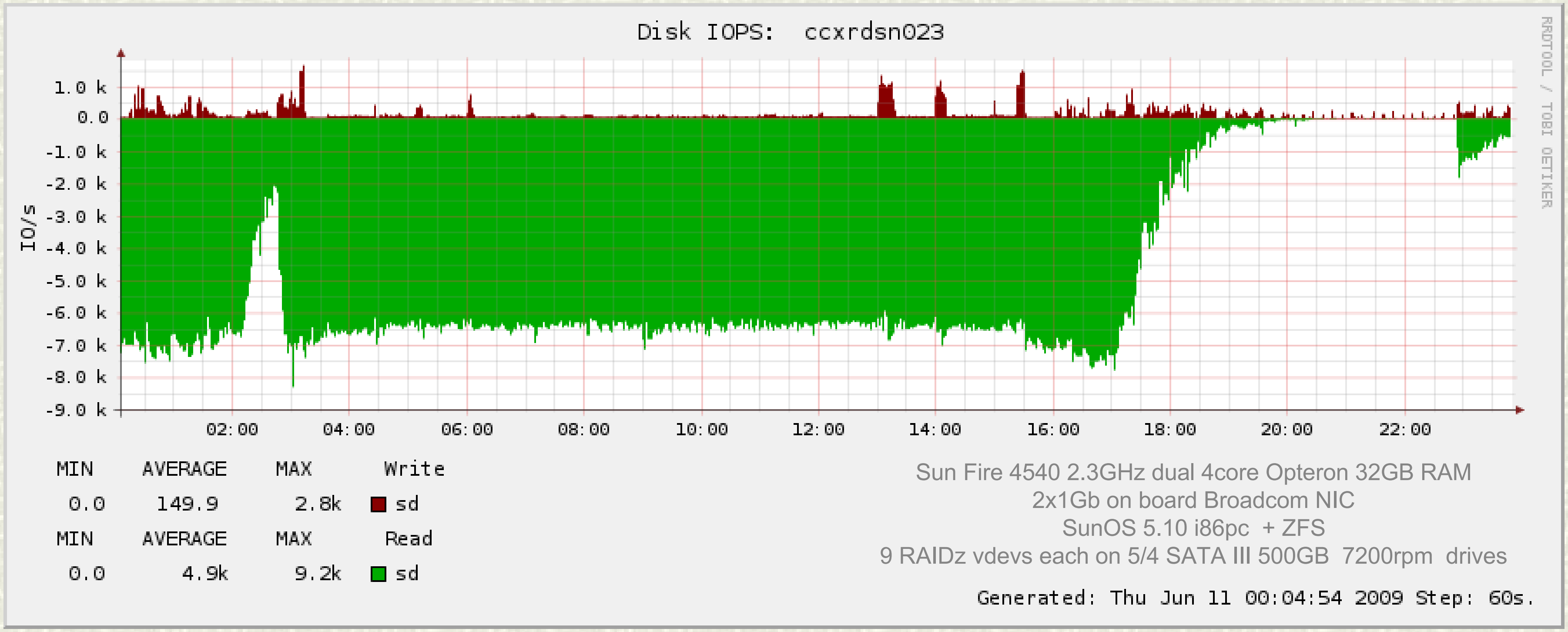
SLAC
NATIONAL ACCELERATOR LABORATORY

ATLAS Data Access Pattern





ATLAS Data Access Impact



350 Analysis jobs using simulated & cosmic data at IN2P3

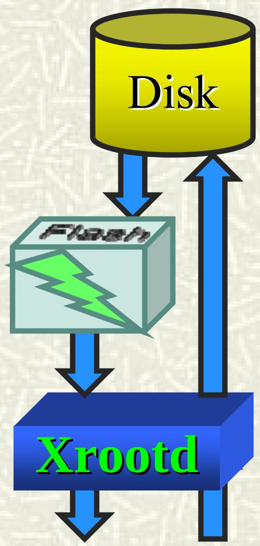
ATLAS Data Access Problem

- Atlas analysis is fundamentally indulgent
  - While xrootd can sustain the load the H/W & FS cannot
- Replication?
  - Except for some files this is not a universal solution
    - The experiment is already disk space insufficient
- Copy files to local node for analysis?
  - Inefficient, high impact, and may overload the LAN
  - Job will still run slowly and no better than local cheap disk
- Faster hardware (e.g., SSD)?
  - This appears to be generally cost-prohibitive
    - That said, we are experimenting with smart SSD handling

Faster **Scalla** I/O (The SSD Option)

Latency only as good as the hardware (xrootd adds < 10µs latency)

- **Scalla** component architecture fosters experimentation
  - Research on intelligently using SSD devices



ZFS Specific

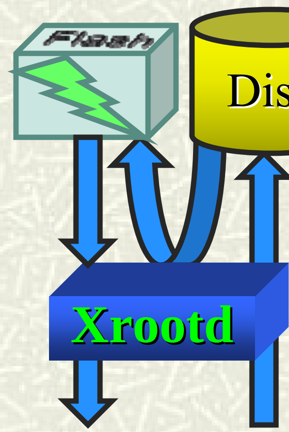
ZFS caches disk blocks via its ARC[1]

Xrootd I/O:

Data sent from RAM/Flash
Data received sent to Disk

R/O Disk Block Cache

[1]Adaptive Replacement Cache



R/O Disk File Cache

FS Agnostic

Xrootd caches files

Xrootd I/O:

Data sent from RAM/Flash
Data received sent to Disk

Will This Be Effective?

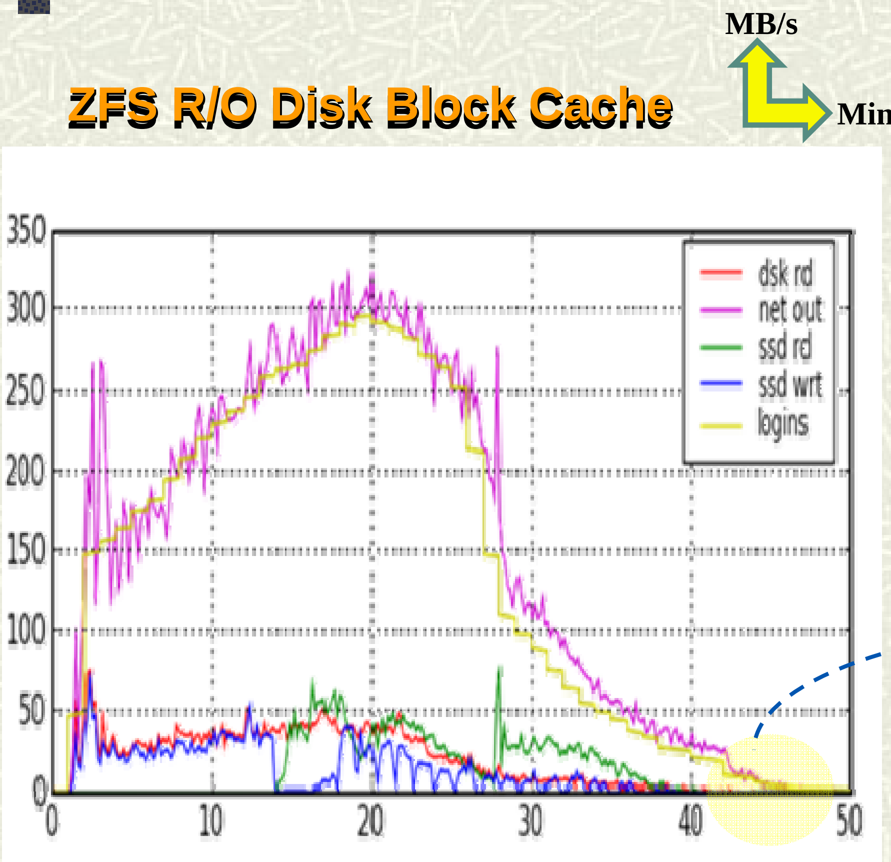
SLAC NATIONAL ACCELERATOR LABORATORY

ZFS Disk Block Cache Setup

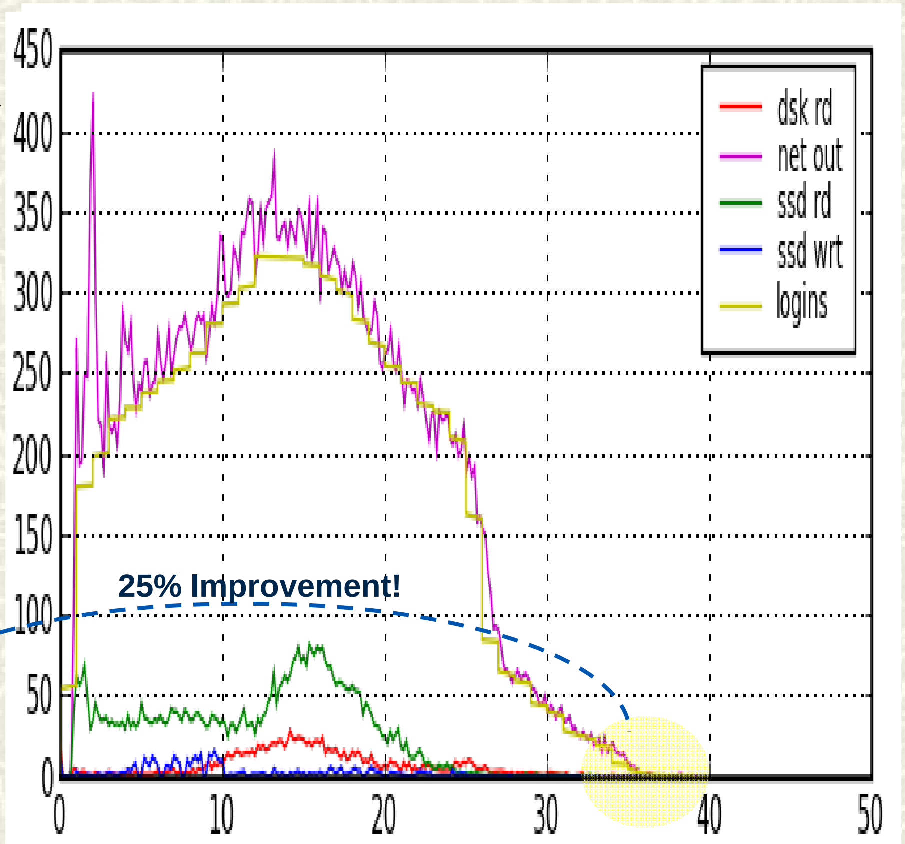
- Sun X4540 Hardware
  - 2x2.3GHz Qcore Opterons, 32GB RAM, 48x1TB 7200 RPM SATA
- Standard Solaris with temporary update 8 patch
  - ZFS SSD cache not support until Update 8
- I/O subsystem tuned for SSD
  - Exception: used 128K read block size
    - This avoided a ZFS performance limitation
- Two FERMI/GLAST analysis job streams
  - First stream after reboot to seed ZFS L2ARC
  - Same stream re-run to obtain measurement

Disk vs SSD With 324 Clients

ZFS R/O Disk Block Cache



Cold SSD Cache I/O



Warm SSD Cache I/O

SLAC NATIONAL ACCELERATOR LABORATORY

If Things Were So Simple!

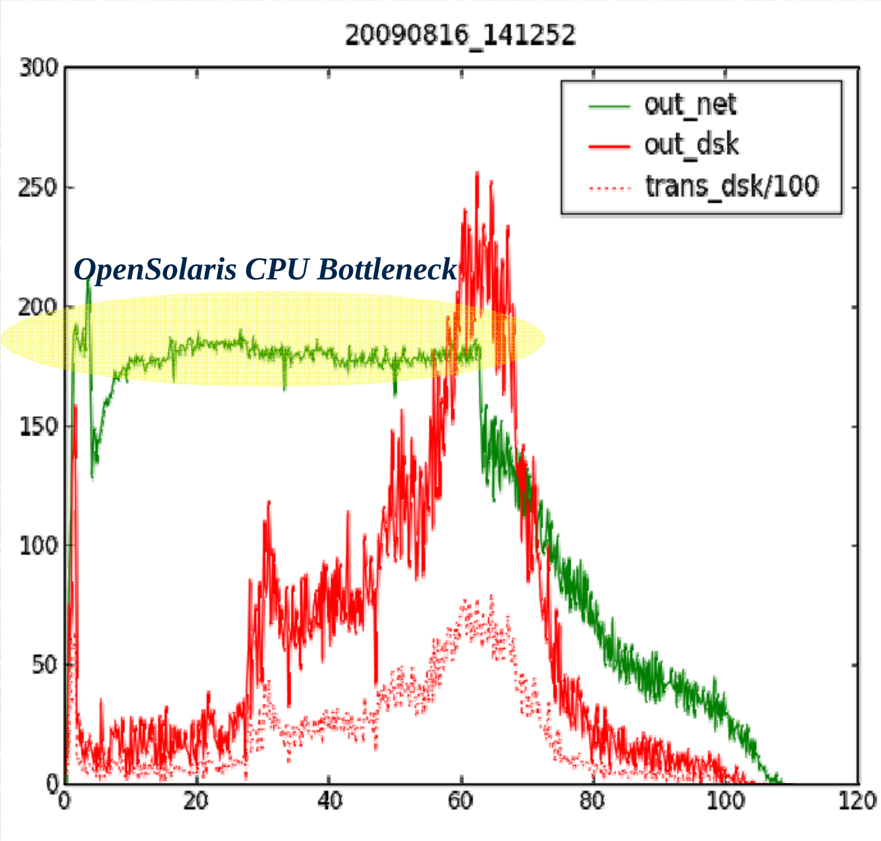
- ZFS Disk Block Cache is workflow sensitive
  - Test represents a specific workflow
    - Multiple job reruns (happens but …)
  - But we could not successfully test the obvious
    - Long term caching of conditions-type (i.e., hot) data
      - Not enough time and no proper job profile
- Whole file caching is much less sensitive
  - At worst can pre-cache for a static workflow
- However, even this can expose other problems

SLAC NATIONAL ACCELERATOR LABORATORY

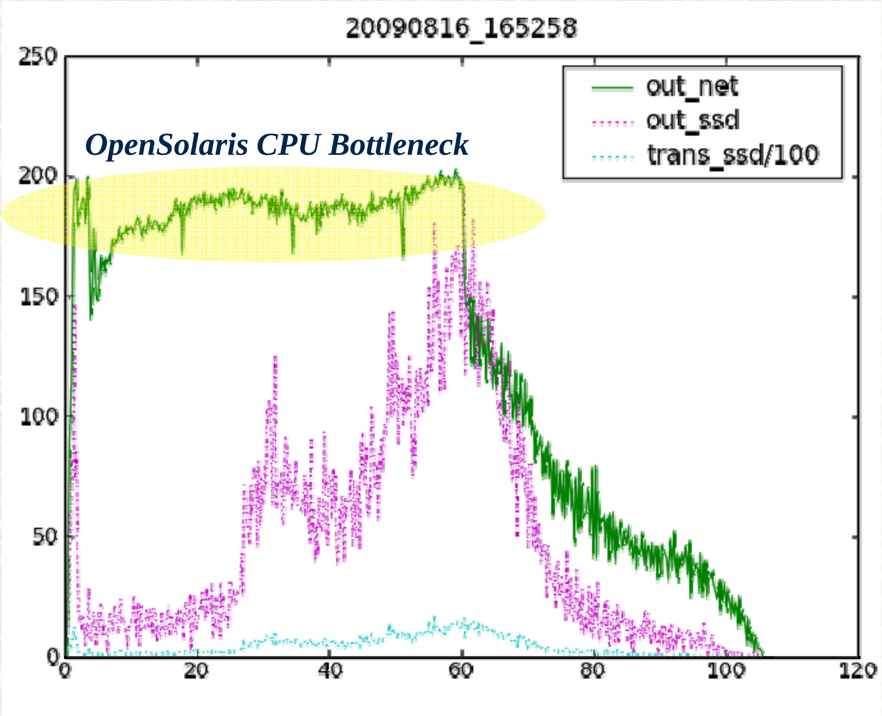
Same Job Stream: Disk vs SSD

MB/s

Min Xroot R/O Disk File Cache



Disk I/O



SSD I/O

SLAC NATIONAL ACCELERATOR LABORATORY

Xrootd R/O Disk File Cache

- Well tuned disk can equal SSD Performance?
  - Yes, when number of well-behaved clients < *small* **n**
    - 324 Fermi/GLAST clients probably not enough *and*
    - Hitting an OS bottleneck
      - OpenSolaris vectors all interrupts through a single CPU
  - Likely we could have done much better
    - *System software* issues proved to be a roadblock
      - This may be an near-term issue with SSD-type devices
- Increasing load on high performance H/W appears to reveal other software problems ….

SLAC NATIONAL ACCELERATOR LABORATORY

What We Saw

- High SSD load can trigger FS lethargy
  - ZFS + 8K blocks + high load = Sluggishness
    - Sun is aware of this problem
- Testing SSD to scale is extremely difficult
  - True until underlying kernel issues resolved
    - This is probably the case irrespective of the OS
    - We suspect that current FS's are attuned to high latency
    - So that I/O algorithms perform poorly with SSD's

The Bottom Line

- Decided against ZFS L2ARC approach (for now)
  - Too narrow
    - Need Solaris 10 Update 8 (likely late 4Q09)
    - Linux support requires ZFS adoption
      - Licensing issues stand in the way
    - Requires substantial tuning
      - Current algorithms optimized for small SSD's
      - Assumes large hot/cold differential
        - Not the HEP analysis data access profile

SLAC NATIONAL ACCELERATOR LABORATORY

The xrootd SSD Option

- Currently architecting appropriate solution
  - Fast track → use the staging infrastructure
    - Whole files are cached
    - Hierarchy: SSD, Disk, Real MSS, Virtual MSS
  - Slow track → cache parts of files (i.e., most requested)
    - Can provide parallel mixed mode (SSD/Disk) access
    - Basic code already present
      - But needs to be expanded
  - First iteration will be fast track approach

SLAC
NATIONAL ACCELERATOR LABORATORY

Future Developments

- Smart **SSD** file caching
- Implement **frm_purge**
  - Needed for new-style XA partitions and SSD's
- Selectable client-side caching algorithms
- Adapting **Scalla** for **mySQL** clusters
  - To be used for **LSST** and perhaps **SciDB**
- Visit the web site for more information
  - http://xrootd.slac.stanford.edu/

Acknowledgements

- Software Contributors
  - Alice: Derek Feichtinger
  - CERN: Fabrizio Furano , Andreas Peters
  - Fermi/GLAST: Tony Johnson (Java)
  - Root: Gerri Ganis, Beterand Bellenet, Fons Rademakers
  - SLAC: Tofigh Azemoon, Jacek Becla, Andrew Hanushevsky, Wilko Kroeger
  - LBNL: Alex Sim, Junmin Gu, Vijaya Natarajan (BeStMan team)
- Operational Collaborators
  - BNL, CERN, FZK, IN2P3, RAL, SLAC, UVIC, UTA
- Partial Funding
  - US Department of Energy
    - Contract DE-AC02-76SF00515 with Stanford University

SLAC
NATIONAL ACCELERATOR LABORATORY