Scalla/xrootd

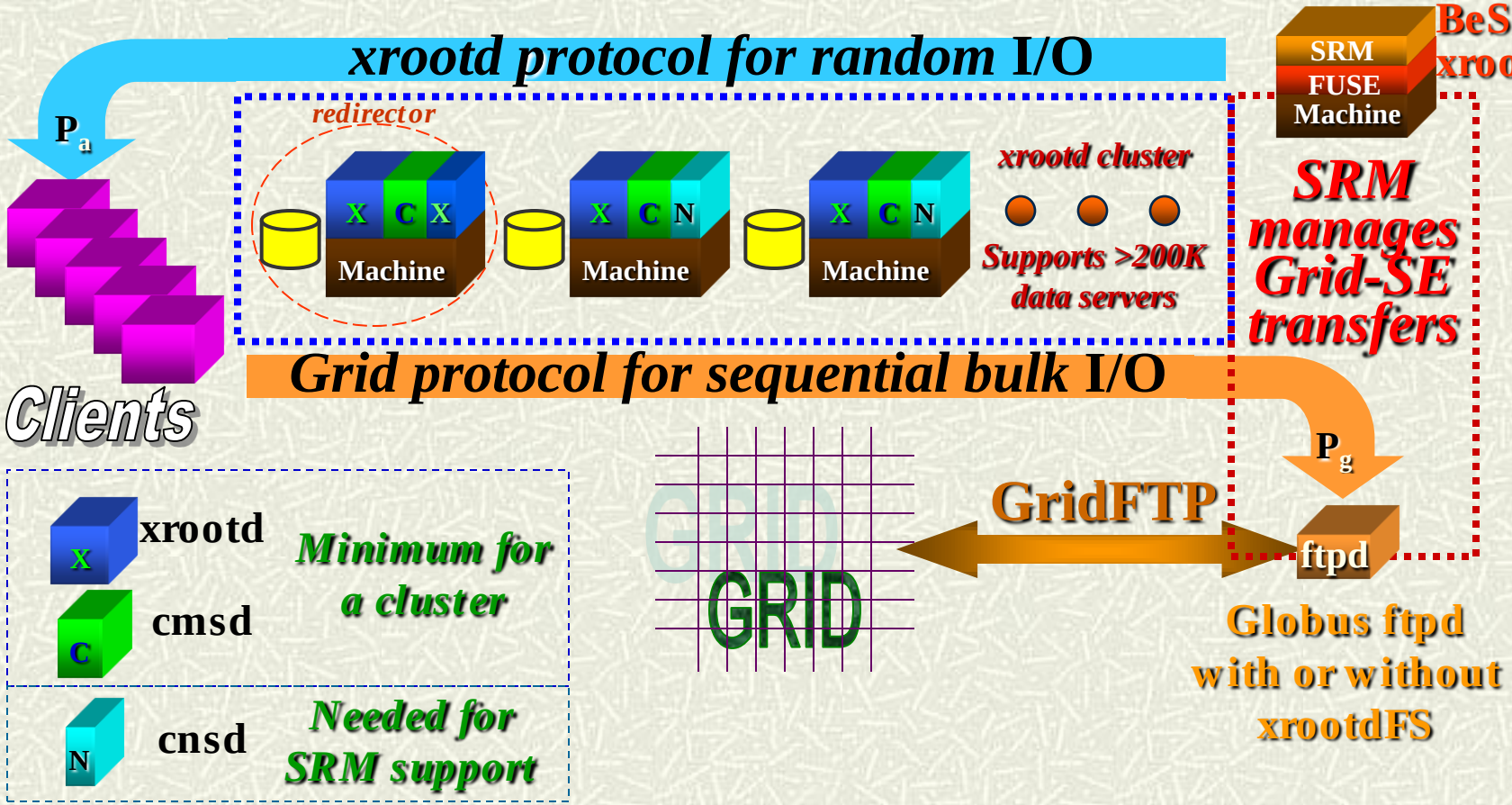
Andrew Hanushevsky
SLAC National Accelerator Laboratory
Stanford University
1-July-09
OSG Storage Forum

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

Outline

- System Overview
  - What's it made of and how it works
- Hardware Requirements
- Typical Configuration Files
- Performance, Scalability, & Stability
- Monitoring & Troubleshooting
- Recent and Future Developments

Scalla Overview



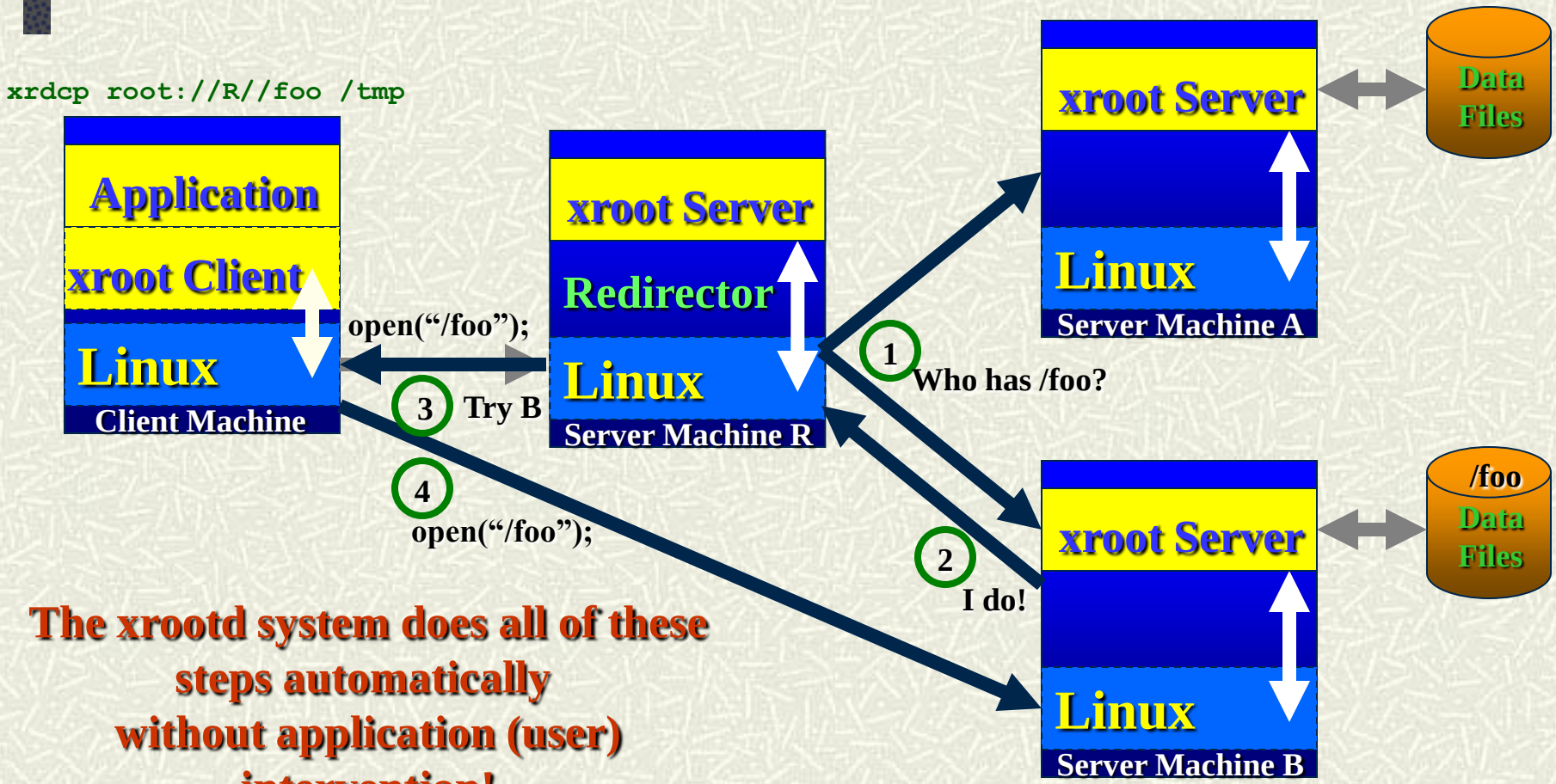
The Components

- xrootd
  - Provides actual data access
- cmsd
  - Glues multiple xrootd's into a cluster
- cnsd
  - 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

Getting to xrootd hosted data

- Via the root framework
  - Automatic when files named root://....
  - Manually, use TXNetFile() object
    - Note: identical TFile() object will not work with xrootd!
- xrdcp
  - The native copy command
- BeStMan (SRM add-on)
  - srmcp, gridFTP
- FUSE
  - Linux only: xrootd as a mounted file system
- POSIX preload library
  - Allows POSIX compliant applications to use xrootd

Cluster Maneuvering



The xrootd system does all of these steps automatically without application (user) intervention!

Corresponding Configuration File

```
# General section that applies to all servers
#
all.export /atlas

if redirector.slac.stanford.edu
all.role manager
else
all.role server
fi
all.manager redirector.slac.stanford.edu 3121

# Cluster management specific configuration
#
cms.allow *.slac.stanford.edu

# xrootd specific configuration
#
xrootd.fslib /opt/xrootd/prod/lib/libXrdOfs.so
xrootd.port 1094
```

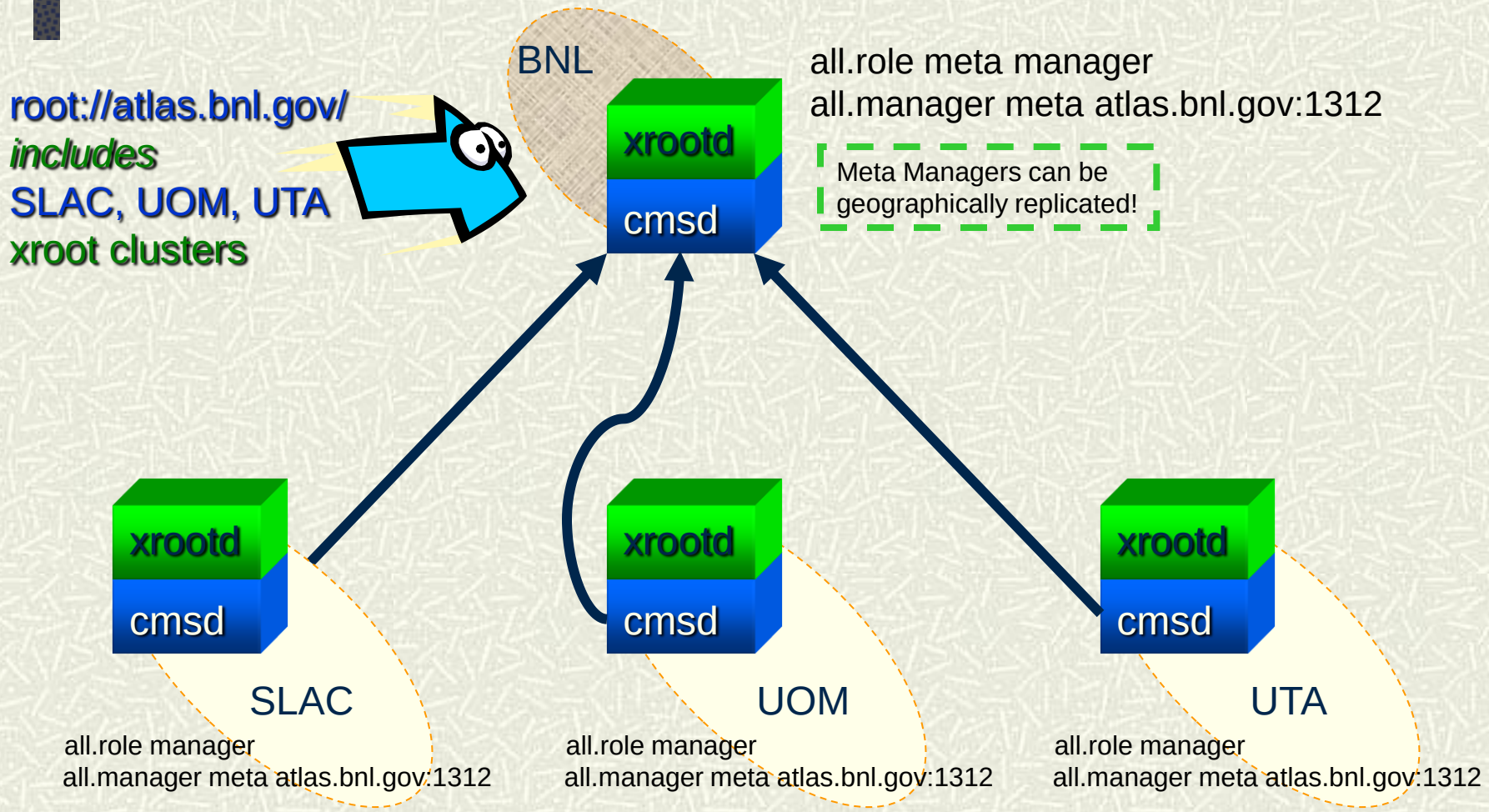
File Discovery Considerations

- The redirector does not have a catalog of files
  - It always asks each server, and
  - Caches the answers in memory for a “while”
    - So, it won’t ask again when asked about a past lookup
- Allows real-time configuration changes
  - Clients never see the disruption
- Does have some side-effects
  - The lookup takes less than a millisecond when files exist
  - Much longer when a requested file does not exist!

Why Do It This Way?

- Simple, lightweight, and ultra-scalable
  - Easy configuration and administration
- Has the R^3 property
  - Real-Time Reality Representation
    - Allows for ad hoc changes
      - Add and remove servers and files without fussing
      - Restart anything in any order at any time
- Uniform cookie-cutter tree architecture
  - Fast-tracks innovative clustering

Cluster Globalization



What's Good About This?

- Uniform view of participating clusters
  - Can easily deploy a virtual MSS
    - Fetch missing files when needed with high confidence
  - Provide real time WAN access as appropriate
    - You don't necessarily need data everywhere all the time!
  - Significantly improve WAN transfer rates
    - Must use torrent copy mode of xrdcp (explained later)
  - Keeps clusters administratively independent
- Alice is using this model
  - Handles globally distributed autonomous clusters

Hardware Requirements

- Xrootd *extremely* efficient of machine resources
  - Ultra low CPU usage with a memory footprint 20 ≈ 80MB
- Redirector
  - At least 500MB – 1GB memory and 2GHZ CPU
- Data Servers
  - I/O or network subsystem will be the bottleneck
    - CPU should be fast enough to handle max disk and network I/O load
  - The more memory the better for file system cache
    - 2 – 4 GB memory recommended
      - But 8 GB memory per core (e.g., Sun Thumper or Thor) works much better
  - Performance directly related to hardware
    - You get what you pay for!

ATLAS xrootd/cmsd Configuration File (1/2)

```
# Typically two variables and a listing of valid hosts are needed to allow for auto-configuration
#
set xrdr  = atl-prod04.slac.stanford.edu
set space = /atlas/xrdcache
set xpath = /opt/xrootd/prod
set apath = /var/adm/xrootd

cms.allow host data004.slac.stanford.edu
cms.allow host wain*.slac.stanford.edu
```

Who the redirector is
Where virtual partitions are mounted
Where binaries & libs are located (usually here)
Where admin information can be stored (usually here)

Specific data servers (zero or more) allowed for cmsd and/or a range of data servers (recommended method)

```
# Generally, everything below the line need not be changed as it used as a template with above settings
#=============================================================================================
#

all.export /atlas/xrootd

all.manager $xrdr 3121

all.adminpath ${apath}

cms.allow host $xrdr

xrootd.chksum max 3 adler32 ${xpath}/bin/xrdadler32

xrootd.fslib ${xpath}/lib/libXrdOfs.so
```

Exported root path (usually one but can have more

Who the redirector is and what cmsd port it uses

The admin path everyone is to use

Redirector always allowed to connect to cmsd

ATLAS uses adler32 which comes packaged with xrootd

The extended filesystem plug-in (should be default)

ATLAS xrootd/cmsd Configuration File (2/2)

```
# This section configures managers, data servers, and request routing all of which can be inferred
#
if $(xrdr) && named anon
  all.role manager
  ofs.forward 3way $(xrdr):1095 mv rm rmdir trunc
else if named anon
  all.role server

  ofs.notify closew create mkdir | ${xpath}/bin/XrdCnsd -l ${apath}/cnsd.log root://$(xrdr):1095

  ofs.notifymsg create $TID create $FMODE $LFN?$CGI
  ofs.notifymsg closew $TID closew $LFN $FSIZE
fi

oss.cache public         $(space) xa
oss.cache ATLASDATADISK  $(space) xa
oss.cache ATLASMCDISK    $(space) xa
oss.cache ATLASPRODDISK  $(space) xa
oss.cache ATLASUSERDISK  $(space) xa
oss.cache ATLASGROUPDISK $(space) xa

if $(xrdr) && named cns
oss.usage log $(space)/.logs quotafile $(space)/.quotas
xrd.port 1095
fi
```

If we're the redirector our role is a manager
How name space requests are routed via manager

Otherwise we must be a plain data server
Route server name space requests to the cnsd

Do not modify these two lines as they describe the format of various xrootd/cnsd messages.

ATLAS space token to virtual partition mapping
Each virtual partition will be named "${space}/*token_name*". So, substituting below this will be "/atlas/xrdcache/ATLASPRODDISK".

cns xrootd maintains usage and quotas
and uses port 1095 instead of default of 1094

Configuration for Start/Stop Scripts

```
#!/bin/sh

# Set XRDUSER to be the username to "su" to if the script is run as root (usually the only thing to be set)
XRDUSER=atldq2

# Set XRDBASE to be the base directory where xrootd version/architecture directories have been installed.
XRDBASE=/opt/xrootd/prod
XRDARCH=''

# Set XRDCONFIG the default config file name. The start script uses '$XRDBASE/etc/$XRDCONFIGN'
XRDCFG=$(XRDBASE)/etc
XRDCONFIG=xrootd.cf

# Set 'XRDLOGDIR' to be the directory where log files are placed and
# Set 'XRDLOGFN'  to be the base log file name (XRDLOGFN for xrootd and CMSLOGFN for cmsd)
XRDLOGDIR=/var/adm/xrootd/logs
XRDLOGFN=xrdlog
CMSLOGFN=cmsdlog

# Specify the working directory (XRDHOMEDIR for xrootd and CMSHOMEDIR for cmsd). Core files go there!
XRDHOMEDIR=/var/adm/xrootd
CMSHOMEDIR=/var/adm/cmsd

# Set 'time' to be number of seconds to wait for a required file to appear (only for AFS or NFS)
time=60

# Set 'count' to be the maximum number of times to wait for a required file (only for AFS or NFS)
count=30
```

StartXRD.cf for StartCMS and StartXRD

Typical FUSE Setup (1/3)

```
#!/bin/sh

# chkconfig: 345 99 10
# chkconfig(sun): S3 99 K0 10
# description: start and stop XrootdFS
#

fuseDir="/afs/slac/package/fuse/@sys"

status() {
    df
}

start() {
    if [ X$LD_LIBRARY_PATH != X ]; then
        LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/afs/slac.stanford.edu/package/xrootd/curr/lib/@sys
    else
        LD_LIBRARY_PATH=/afs/slac.stanford.edu/package/xrootd/curr/lib/@sys
    fi
    LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/afs/slac.stanford.edu/package/fuse/@sys/lib
    export LD_LIBRARY_PATH

    ulimit -c unlimited
    cd /
    export XROOTDFS_OFSFWD=0
#   export XROOTDFS_USER='daemon'
    export XROOTDFS_FASTLS="RDR"
    insmod $fuseDir/kernel-modules/`uname -r`/kernel/fs/fuse/fuse.ko 2> /dev/null
```

Typical FUSE Setup (2/3)

```
MOUNT_POINT="/xrootd/atlas/usr"
    export XROOTDFS_RDRURL="root://atl-xrdr:1094//atlas/xrootd/usr"
    export XROOTDFS_CNSURL="root://atl-xrdr:1095//atlas/xrootd/usr"
    $fuseDir/bin/xrootdfsd $MOUNT_POINT -o allow_other,fsname=xrootdfs,max_write=131072,direct_io
MOUNT_POINT="/xrootd/atlas/dq2"
    export XROOTDFS_RDRURL="root://atl-xrdr:1094//atlas/xrootd/dq2"
    export XROOTDFS_CNSURL="root://atl-xrdr:1095//atlas/xrootd/dq2"
    $fuseDir/bin/xrootdfsd $MOUNT_POINT -o allow_other,fsname=xrootdfs,max_write=131072,direct_io
MOUNT_POINT="/xrootd/atlas/atlasdatadisk"
    export XROOTDFS_RDRURL="root://atl-xrdr:1094//atlas/xrootd/atlasdatadisk"
    export XROOTDFS_CNSURL="root://atl-xrdr:1095//atlas/xrootd/atlasdatadisk"
    $fuseDir/bin/xrootdfsd $MOUNT_POINT -o allow_other,fsname=xrootdfs,max_write=131072,direct_io
MOUNT_POINT="/xrootd/atlas/atlasmcdisk"
    export XROOTDFS_RDRURL="root://atl-xrdr:1094//atlas/xrootd/atlasmcdisk"
    export XROOTDFS_CNSURL="root://atl-xrdr:1095//atlas/xrootd/atlasmcdisk"
    $fuseDir/bin/xrootdfsd $MOUNT_POINT -o allow_other,fsname=xrootdfs,max_write=131072,direct_io
MOUNT_POINT="/xrootd/atlas/atlasproddisk"
    export XROOTDFS_RDRURL="root://atl-xrdr:1094//atlas/xrootd/atlasproddisk"
    export XROOTDFS_CNSURL="root://atl-xrdr:1095//atlas/xrootd/atlasproddisk"
    $fuseDir/bin/xrootdfsd $MOUNT_POINT -o allow_other,fsname=xrootdfs,max_write=131072,direct_io
MOUNT_POINT="/xrootd/atlas/atlasuserdisk"
    export XROOTDFS_RDRURL="root://atl-xrdr:1094//atlas/xrootd/atlasuserdisk"
    export XROOTDFS_CNSURL="root://atl-xrdr:1095//atlas/xrootd/atlasuserdisk"
    $fuseDir/bin/xrootdfsd $MOUNT_POINT -o allow_other,fsname=xrootdfs,max_write=131072,direct_io
MOUNT_POINT="/xrootd/atlas/atlasgroupdisk"
    export XROOTDFS_RDRURL="root://atl-xrdr:1094//atlas/xrootd/atlasgroupdisk"
    export XROOTDFS_CNSURL="root://atl-xrdr:1095//atlas/xrootd/atlasgroupdisk"
    $fuseDir/bin/xrootdfsd $MOUNT_POINT -o allow_other,fsname=xrootdfs,max_write=131072,direct_io
}
```

Typical FUSE Setup (3/3)

```
stop() {
    umount /xrootd/atlas/usr
    umount /xrootd/atlas/dq2
    umount /xrootd/atlas/atlasdatadisk
    umount /xrootd/atlas/atlasmcdisk
    umount /xrootd/atlas/atlasproddisk
    umount /xrootd/atlas/atlasuserdisk
    umount /xrootd/atlas/atlasgroupdisk
}
case "$1" in
start)
    start
    ;;
stop)
    stop
    ;;
restart)
    stop
    start
    ;;
status)
    status
    ;;
*)
    echo "Usage: $0 {start|stop|restart|status}"
    exit 1
esac
```

BeStMan Configuration (1/3)

```
publicPort=8080
securePort=8443
FactoryID=server
supportedProtocolList=gsiftp://atl-prod07.slac.stanford.edu;gsiftp://atl-prod08.slac.stanford.edu

# lcg-utils doesn't accept service certificate
#
CertFileName=/etc/grid-security/srm/hostcert.pem
KeyFileName=/etc/grid-security/srm/hostkey.pem

# Enable GUMS: note GUMSCurrHostDN need to be identical to
#              openssl x509 -in CertFileName -subject -noout | cut -f2 -d' '
#
GridMapFileName=/opt/bestman/conf/grid-mapfile.empty
GUMSserviceURL=https://osgserv02.slac.stanford.edu:8443/gums/services/GUMSAuthorizationServicePort
GUMSCurrHostDN=/DC=org/DC=doegrids/OU=Services/CN=osgserv04.slac.stanford.edu

# To use sudo to manage file system (mkdir, rmdir, rm, mv, cp, ls)
# Assuming user 'daemon' run bestman, need the following in /etc/sudoers
#    Cmnd_Alias SRM_CMD = /bin/rm, /bin/mkdir, /bin/rmdir, /bin/mv, /bin/cp, /bin/ls
#    Runas_Alias SRM_USR = ALL, !root
#    daemon     ALL=(SRM_USR) NOPASSWD: SRM_CMD
# Also need "Defaults !requiretty" for some sudo implementation.
#
# To enable, set accessFileSysViaSudo=true. To use sudo for srmLs as well, set noSudoOnLs=false
# (Please watch out the overhead of doing so)
#
accessFileSysViaSudo=false
noSudoOnLs=true
```

BeStMan Configuration (2/3)

```
# Event log (not the bestman log)
#
noEventLog=false
EventLogLocation=/tmp

# BeStMan Gateway Mode only: Xrootd only, use xrootdTokenCompName=oss.cgroup to pass space token to GridFTPs
#
xrootdTokenCompName=oss.cgroup

# BeStMan Gateway Mode only: Associate a space token usage with a path. Setting it to true will
#                            enable space availability checking when receiving a srmPrepareToPut() request
#
pathForToken=false

# BeStMan Gateway Mode only: Do srmLs/Rm/Mkdir/Rmdir/Mv via filesystem
#
checkSizeWithFS=true
checkSizeWithGsiftp=false

# Static space token list: Token[space in BG][desc:TokenDescription] or an empty list.
#
staticTokenList=ATLASDATADISK[desc:ATLASDATADISK][101312][owner:atlas][retention:REPLICA][latency:ONLINE]
                [usedBytesCommand:/afs/slac/package/xrootd/atlasutils/space.sh -used ATLASDATADISK];
                ATLASMCDISK[67104][desc:ATLASMCDISK][owner:atlas][retention:REPLICA][latency:ONLINE]
                [usedBytesCommand:/afs/slac/package/xrootd/atlasutils/space.sh -used ATLASMCDISK];
                ATLASPRODDISK[40480][desc:ATLASPRODDISK][owner:atlas][retention:REPLICA][latency:ONLINE]
                [usedBytesCommand:/afs/slac/package/xrootd/atlasutils/space.sh -used ATLASPRODDISK];
                ATLASUSERDISK[5120][desc:ATLASUSERDISK][owner:atlas][retention:REPLICA][latency:ONLINE]
                [usedBytesCommand:/afs/slac/package/xrootd/atlasutils/space.sh -used ATLASUSERDISK];
                ATLASGROUPDISK[1024][desc:ATLASGROUPDISK][owner:atlas][retention:REPLICA][latency:ONLINE]
                [usedBytesCommand:/afs/slac/package/xrootd/atlasutils/space.sh -used ATLASGROUPDISK
```

BeStMan Configuration (3/3)

```
# noCacheLog must be true for BeStMan Gateway mode, and false otherwise.
#
noCacheLog=true
CacheLogLocation=/var/log

# disableSpaceMgt must be true for BeStMan Gateway mode
#
disableSpaceMgt=true

# Support adler32, md5 and crc32
defaultChecksumType=adler32

# default is false. when set to true, then srmLs on file shows its checksum
# i.e. srmLs(file or files) would show checksum with fullDetailedList option
# srmLs(dir) will not show checksum on its files.
showChecksumWhenListingFile=false

# Grid map caching values 0 (cache it permanently as long as MaxMappedIDCached permits)
#                       < 0 (never cache it)
#                       > 0 (stay in cache for that much seconds)
# default is 1800 seconds.
LifetimeSecondsMappedIDCached=1800
```

Conclusion on Requirements

- Minimal hardware requirements
  - Anything that natively meets service requirements
- Configuration requirements vary
  - xrootd/cmsd configuration is trivial
    - Only several lines have to be specified
  - FUSE configuration wordy but methodical
    - Varies little from one environment to the next
  - BeStMan configuration is non-trivial
    - Many options need to be considered and specified
    - Other systems involved (GUMS, sudo, etc.)

Scalla Performance

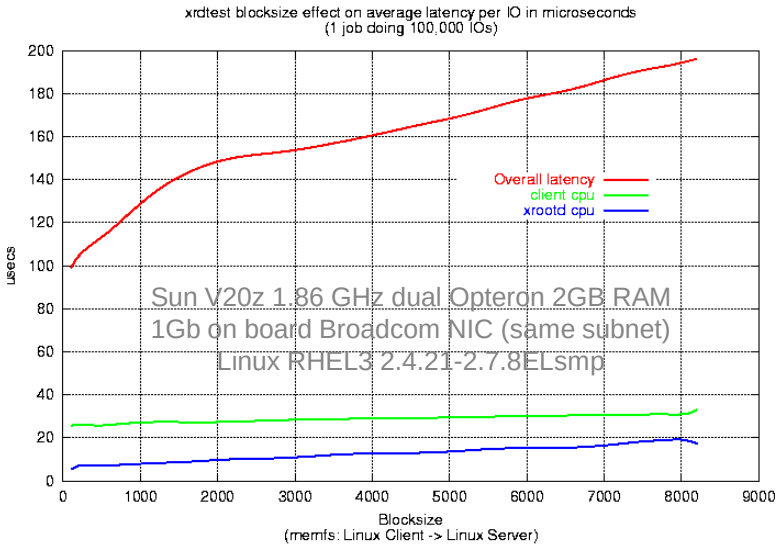
- Following figures are based on actual measurements
  - These have also been observed by many *production* sites
    - E.G., BNL, IN2P3, INFN, FZK, RAL , SLAC

CAVEAT!

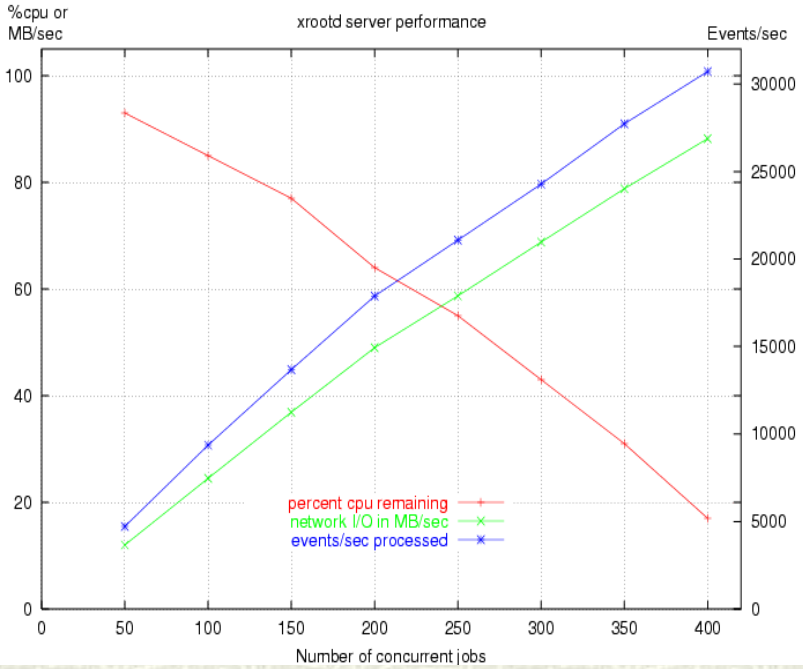
  - Figures apply only to the ***reference*** implementation
  - Other implementations vary significantly
    - Castor + xrootd protocol driver
    - dCache + native xrootd protocol implementation
    - DPM + xrootd protocol driver + cmsd XMI
    - HDFS + xrootd protocol driver

Server Performance

Latency



Capacity vs. Load



xrootd latency < 10µs → network or disk latency dominates

Practically, at least ≈10,000 Ops/Second with linear scaling

xrootd+cmsd latency *(not shown)* 350µs →»1000 opens/second

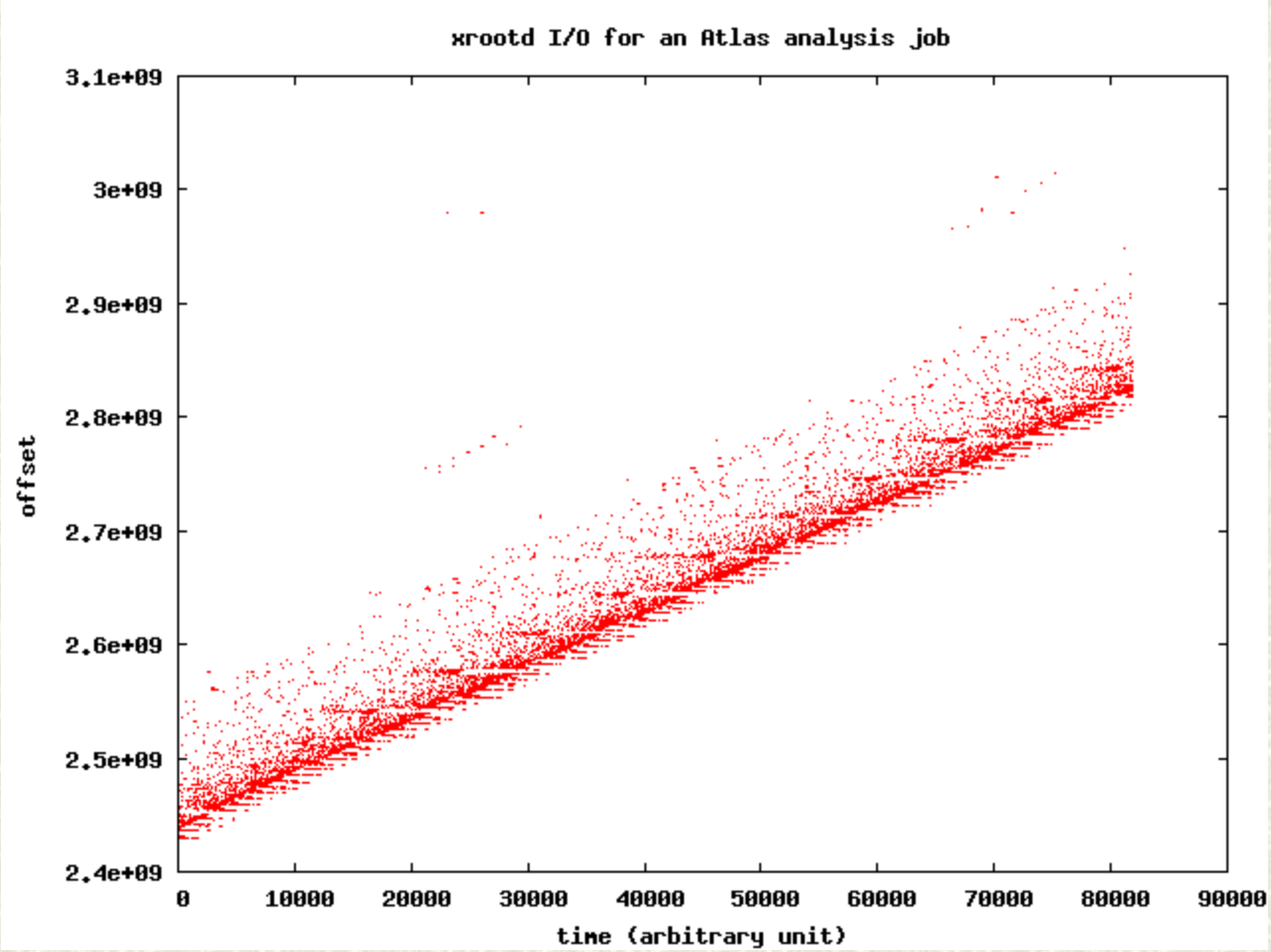
High Performance Side Effects

- High performance + linear scaling
  - Makes client/server software virtually transparent
    - Disk subsystem and network become determinants
      - This is actually excellent for planning and funding

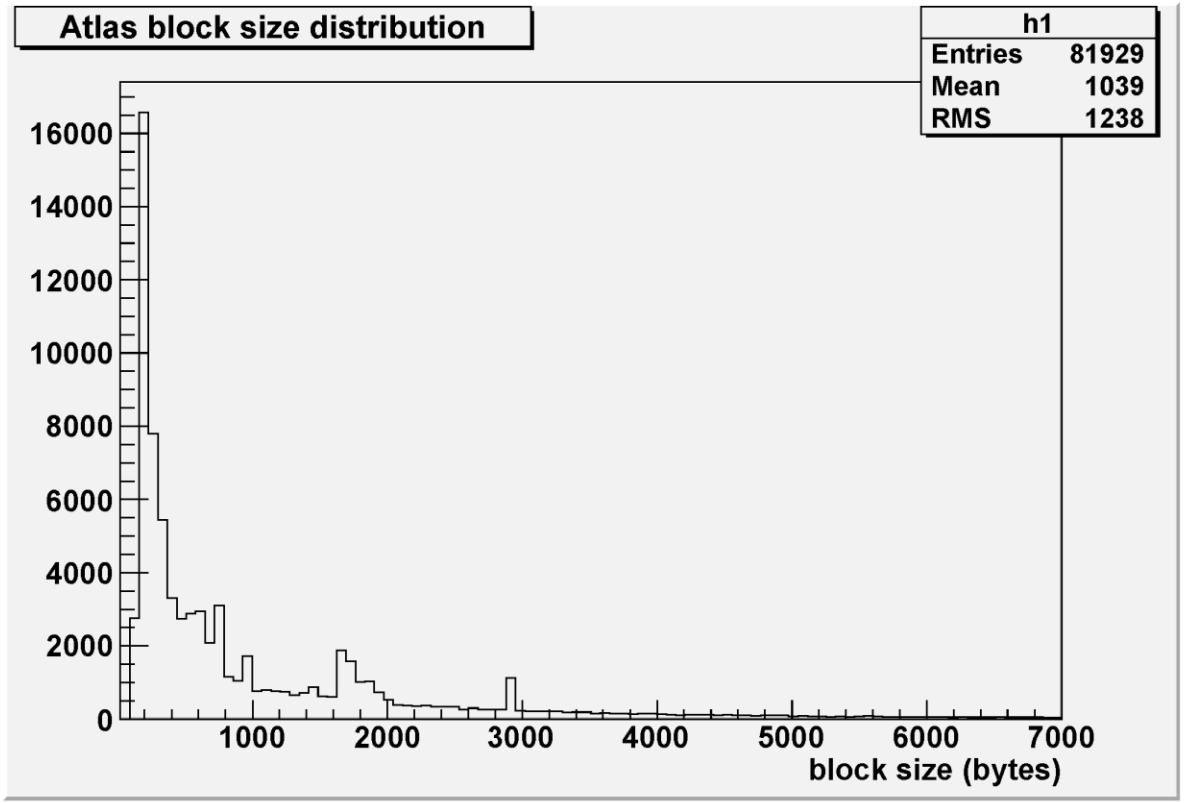
HOWEVER

  - Transparency allows applications to over-run H/W
    - Hardware/Filesystem/Application dependent
      - Requires deft trade-off between CPU & Storage resources
    - Over-runs usually due to unruly applications
      - Such as ATLAS analysis

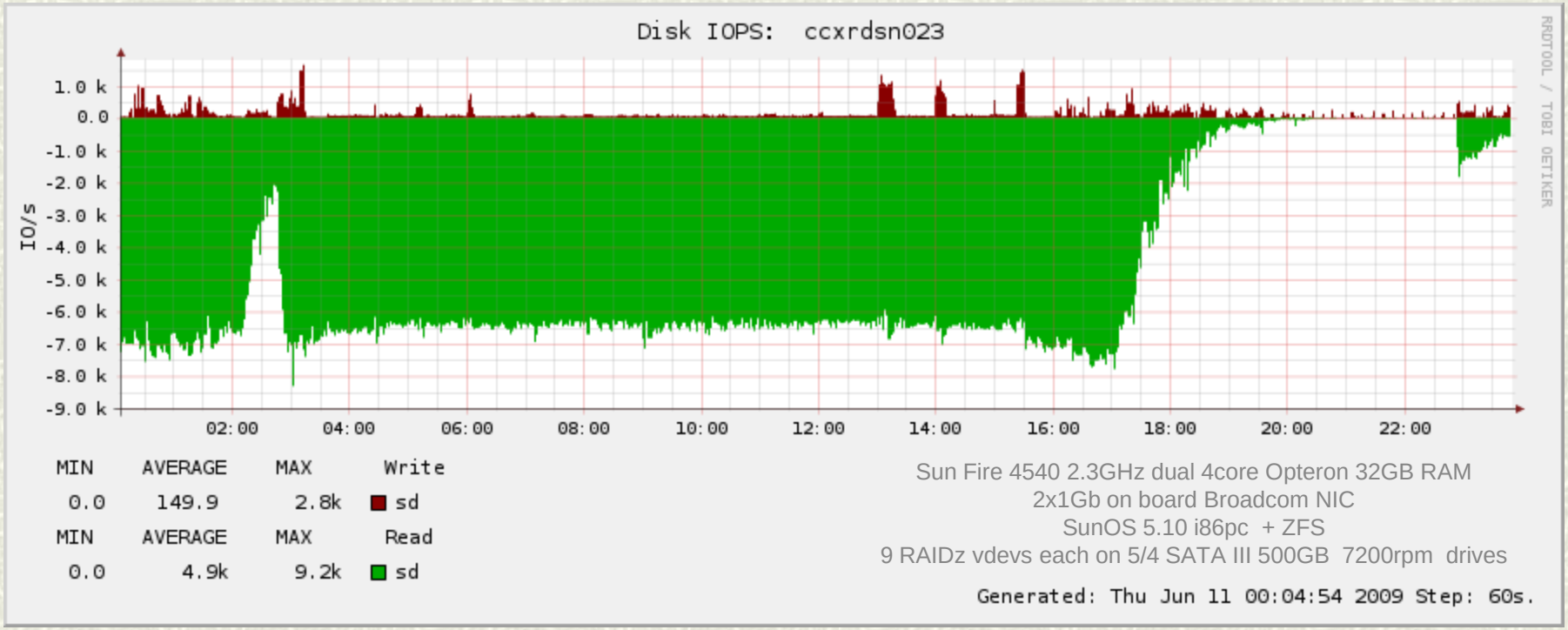
ATLAS Data Access Pattern



ATLAS Data Access Block Size



ATLAS Data Access Result



350 Analysis jobs using simulated & cosmic data at IN2P3

SLAC NATIONAL ACCELERATOR LABORATORY

Data Access Problem

- Atlas analysis is fundamentally indulgent
  - While xrootd can sustain the request load the H/W 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
- Faster hardware (e.g., SSD)?
  - This appears to be generally cost-prohibitive
    - That said, we are experimenting with smart SSD handling

Some "Realistic" Approaches

- Code rewrite is the most cost-effective
  - Unlikely to be done soon due to publishing pressure
- Minimizing performance degradation
  - Recently implemented in xrootd (see next slide)
- Tight file system tuning
  - Problematic for numerous reasons
- Batch job throttling
  - Two approaches in mind
    - Batch queue feedback (depends on batch system)
    - Use built-in xrootd reactive scheduling interface

Overload Detection

- The xrootd server is blazingly fast
  - Trivial to overload a server's I/O subsystem
    - File system specific, but effect generally the same
      - Sluggish response and large differential between Disk & Net I/O
- New overload recovery algorithm implemented
  - Detects overload in the session recovery path
    - Paces clients to re-achieve expected response envelope
    - Additional improvements here will likely be needed

Stability & Scalability Improvements

- xrootd has a 5+ year production history
  - Numerous high-stress environments
    - BNL, FZK, IN2P3, INFN, RAL, SLAC
  - Stability has been vetted
    - Changes are now very focused
      - Functionality improvements
      - Hardware/OS edge effect limitations
      - Esoteric bugs in low use paths
  - Scalability is already at the theoretical maximum
    - E.g., STAR/BNL runs a 400+ server production cluster

Summary Monitoring

- xrootd has built-in summary monitoring
- Can auto-report summary statistics
  - xrd.report configuration directive
    - Available in latest release
  - Data sent to up to two central locations
  - Accommodates most current monitoring tools
    - Ganglia, GRIS, Nagios, MonALISA, and perhaps more
      - Requires external xml-to-monitor data convertor
      - Will provide a stream multiplexing and xml parsing tool
      - Currently converting Ganglia feeder to use auto-reporting

Detail Monitoring

- xrootd has built-in multi-level monitoring
  - Session, service, and I/O requests
    - Detail level if configurable
    - Minimal impact on server performance
  - Current data collector/renderer is functional
    - Still needs better packaging and documentation
    - Work in progress to fund “productization” effort

Native Detailed View (one of many)

Top Performers Table

Table rows: 5 | Time Period: Last Hour | Site: SLAC | Update

Top active users

User Name	Now			Last Hour			
	Number of Jobs	Number of Files	File Size [MB]	Number of Jobs	Number of Files	File Size [MB]	MB Read
ayarritu	615	139	65,987	430	146	65,802	41,360
jregens	360	405	371,874	64	317	303,252	143,852
cschill	281	32	27,133	79	30	25,301	4,892
feltresi	149	106	167,528	70	143	218,873	74,552
torsten	72	99	83,673	184	1,532	630,092	235,327

Hottest dataTypes

dataType Name	Now				Last Hour				
	Number of Jobs	Number of Files	File Size [MB]	Number of Users	Number of Jobs	Number of Files	File Size [MB]	Number of Users	MB Read
SPskims	998	739	632,651	11	663	340	304,938	6	120,728
SP	652	1,839	1,961,610	12	981	506	474,819	7	159,512
PRskims	93	650	811,152	7	204	83	107,807	2	62,265
PR	66	600	453,640	6	265	1,454	525,498	3	174,754
cfg	0	0	0	0	8	1	7	1	10

Hottest skims

skim Name	Now				Last Hour				
	Number of Jobs	Number of Files	File Size [MB]	Number of Users	Number of Jobs	Number of Files	File Size [MB]	Number of Users	MB Read
BtoRhoGamma	591	139	65,987	1	458	146	65,802	1	41,360
DstToD0PiToVGamma	262	86	33,138	1	70	41	16,171	1	4,668
BToDlnu	115	118	186,026	2	125	145	222,200	2	74,568
AllEvents	76	394	508,309	3	210	84	108,365	3	62,268
Tau11	4	95	130,103	1	3	6	149	0	127

Hottest files

File Path	File Size [MB]	Now	Last Hour	
		Number of Jobs	Number of Jobs	MB Read
/store/PRskims/R18/18.6.3d/AllEvents/00/AllEvents_20006.04HB.root	1,690	2	15	1,630
/store/PRskims/R18/18.6.3e/AllEvents/05/AllEvents_20502.04HB.root	1,688	1	17	1,636
/store/PRskims/R18/18.6.3e/AllEvents/05/AllEvents_20502.01.root	1,689	1	17	1,635
/store/PRskims/R18/18.6.3e/AllEvents/05/AllEvents_20500.03HB.root	1,688	1	19	1,641
/store/PRskims/R18/18.6.3e/AllEvents/05/AllEvents_20500.01.root	1,689	1	19	1,640

Troubleshooting

- Troubleshooting is log based
  - Xrootd & cmsd maintain auto-rotating log files
  - Messages usually indicate the problem
    - Log is normally lean and clean
      - Easy to spot problem messages
      - Can change verbosity to further isolate problems
        - E.g., turn on various trace levels
- Simple clustering of independent file systems
  - Use OS tools to trouble shoot any individual node
    - Admin needs to know standard Unix tools to succeed

Problem Handling

- OSG is the first level contact
  - Handles obvious problems
    - E.G., configuration problems
- SLAC or CERN handle software problems
  - OSG refers software problems to SLAC
  - Problem ticket is created to track resolution
    - R/T via xrootd-bugs@slac.stanford.edu
- Use discussion list for general issues
  - http://xrootd.slac.stanford.edu/
    - List via xrootd-l@slac.stanford.edu

Recent Developments

- Auto-reporting summary data
  - June, 2009 (covered under monitoring)
- Ephemeral files
  - June, 2009
- Torrent WAN transfers
  - May, 2009
- File Residency Manager (FRM)
  - April, 2009

Ephemeral Files

- Files that persist only when successfully closed
  - Excellent safeguard against leaving partial files
    - Application, server, or network 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

Improved WAN Transfer

- 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 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)

File Residency Manager (FRM)

- Functional replacement for MPS 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, obtain space information
        - Space token names, utilization, etc.
      - Can run on a live system

Future Developments

- Simple Server Inventory (SSI) [July release]
  - Maintains a file inventory per data server
  - Does *not* replace PQ2 tools (Neng Xu, Univerity of Wisconsin)
    - Good for uncomplicated sites needing a server inventory
- Smart SSD handling [July start]
- Getfile/Putfile protocol implementation [August release]
  - Support for true 3rd party server-to-server transfers
- Additional simplifications [ongoing]

Conclusion

- Xrootd is a lightweight data access system
  - Suitable for resource constrained environments
    - Human as well as hardware
  - Rugged enough to scale to large installations
    - CERN analysis & reconstruction farms
- Readily available
  - Distributed as part of the OSG VDT
    - Also part of the CERN root distribution
- 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