Scalla Advancements

xrootd /cmsd (f.k.a. olbd)

Fabrizio Furano
CERN – IT/GS

Andrew Hanushevsky
Stanford Linear Accelerator Center

Cern Seminar
Stanford University/SLAC
9-May-08

http://xrootd.slac.stanford.edu

Stanford Linear Accelerator Center

Outline

- Introduction
- Current Developments
  - Composite Cluster Name Space
  - POSIX file system access via FUSE+xrootd
    - SRM support
  - Cluster Management Service (cmsd)
    - Cluster globalization
    - Virtual MSS
  - Bandwidth Scheduling
    - Directed Support Services
- Announcements
- Conclusion

Stanford
Linear
Accelerator
Center

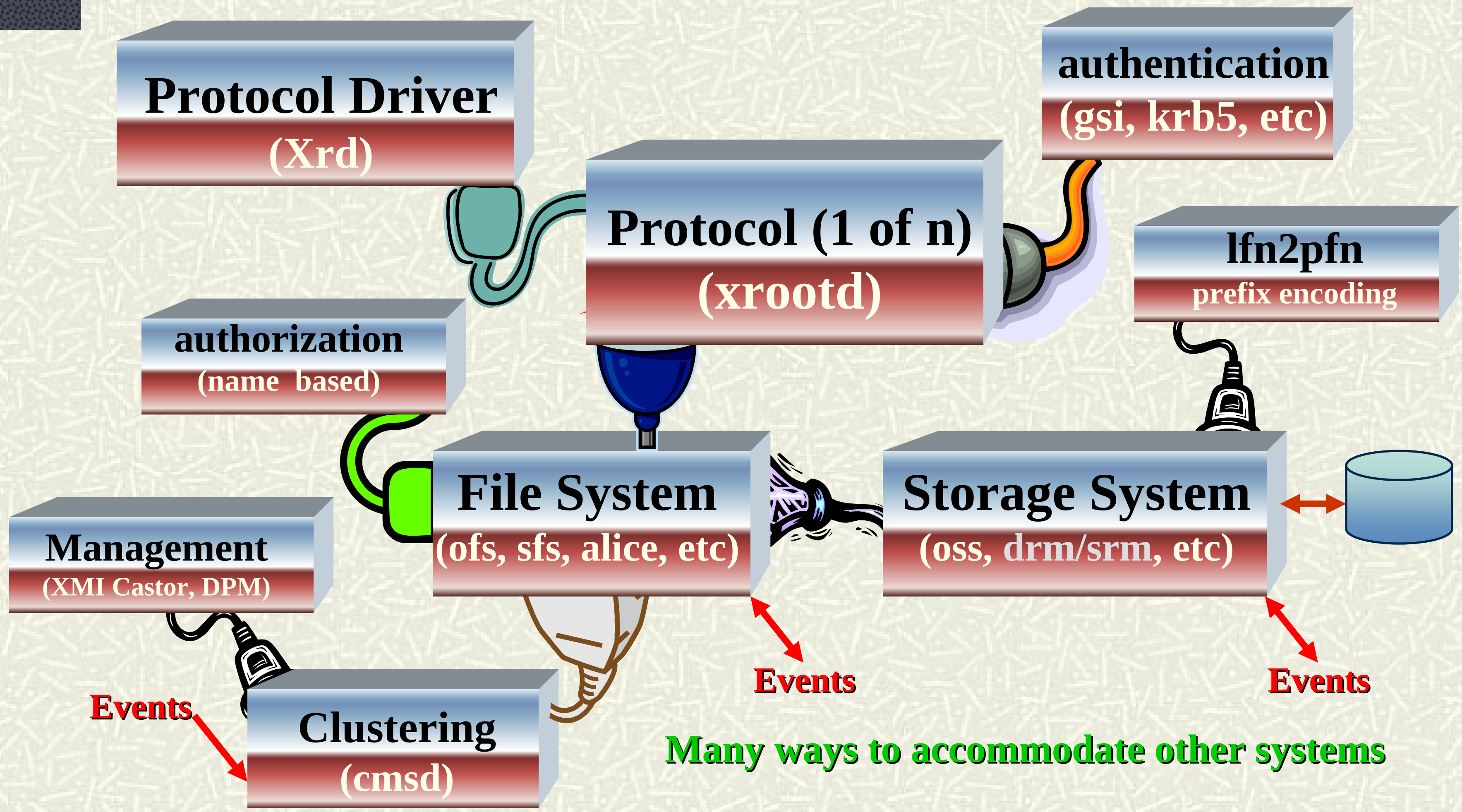
What is **Scalla**?

- **S**tructured **C**luster **A**rchitecture for **L**ow **L**atency **A**ccess
  - Low latency access to data via xrootd servers
    - POSIX-style byte-level random access
      - Hierarchical directory-like name space of arbitrary files
        - Does not have full file system semantics
        - This is *not* a general purpose data management solution
    - Protocol includes high performance & scalability features
  - Structured clustering provided by cmsd servers
    - Exponentially scalable and self organizing

General Design Points

- High speed access to *experimental* data
  - Write once read many times processing mode
  - Small block sparse random access (e.g., root files)
  - High transaction rate with rapid request dispersal (*fast* opens)
- Low setup cost
  - High efficiency data server (low CPU/byte overhead, small memory footprint)
  - Very simple configuration requirements
  - No 3rd party software needed (avoids messy dependencies)
- Low administration cost
  - Non-assisted fault-tolerance
  - Self-organizing servers remove need for configuration changes
  - No database requirements (no backup/recovery issues)
- Wide usability
  - Full POSIX access
  - Server clustering for scalability
  - Plug-in architecture and event notification for applicability (HPSS, Castor, etc)

xrootd Plugin Architecture



Stanford Linear Accelerator Center

Architectural Significance

- Plug-in Architecture Plus Events
  - Easy to integrate other systems
- Orthogonal Design
  - Uniform client view irrespective of server function
    - Easy to integrate distributed services
    - System scaling always done in the same way
- Plug-in Multi-Protocol Security Model
  - Permits real-time protocol conversion
- System Can Be Engineered For Scalability
  - Generic clustering plays a significant role

Stanford Linear Accelerator Center

Single point performance

- Very carefully crafted, heavily multithreaded
  - Server side: promote speed and scalability
    - High level of internal parallelism + stateless
    - Exploits OS features (e.g. async i/o, polling, selecting)
    - Many many speed+scalability oriented features
    - Supports thousands of client connections
  - Client: Handles the state of the communication
      - Constructs a simple interface from complex interactions
    - Fast data path
    - Network pipeline coordination + latency hiding
    - Supports connection multiplexing + intelligent server cluster crawling
- Server and client exploit multi core CPUs natively

Fault tolerance

- Server side
  - If servers go down, the overall functionality ***can*** be fully preserved
    - Redundancy, MSS staging of replicas, …
    - Means that static deployments can be avoided
      - E.g. storing in a DB the physical endpoint addresses for each file
- Client side (+protocol)
  - The application never notices errors
    - Totally transparent, until they become fatal
      - i.e. when it becomes really impossible to get to a working endpoint to resume the activity
  - Typical tests (try it!)
    - Disconnect/reconnect network cables
    - Kill/restart servers

Authentication

Courtesy of Gerardo Ganis (CERN PH-SFT)

- Flexible, multi-protocol system
  - Abstract protocol interface: XrdSecInterface
    - Protocols implemented as dynamic plug-ins
    - Architecturally self-contained
      - NO weird code/libs dependencies (requires only openssl)
      - High quality highly optimized code, great work by Gerri Ganis
- Embedded protocol negotiation
  - Servers define the list, clients make the choice
  - Servers lists may depend on host / domain
- One handshake per process-server connection
  - Reduced overhead:
  - # of handshakes ≤ # of servers contacted
    - Exploits multiplexed connections
    - no matter the number of file opens

Authentication Protocols

Courtesy of Gerardo Ganis (CERN PH-SFT)

- Password-based (pwd)
  - Either system or dedicated password file
    - User account not needed
- GSI (gsi)
  - Handle GSI proxy certificates
  - VOMS support should be OK now (Andreas, Gerri)
  - No need of Globus libraries (and super-fast!)
- Kerberos IV, V (krb4, krb5)
  - Ticket forwarding supported for krb5
  - Fast ID (unix, host) to be used w/ authorization
- Unix (unix)
  - Simple nfs-like protocol to supply uid/gid
- ALICE security tokens
  - Emphasis on ease of setup and performance

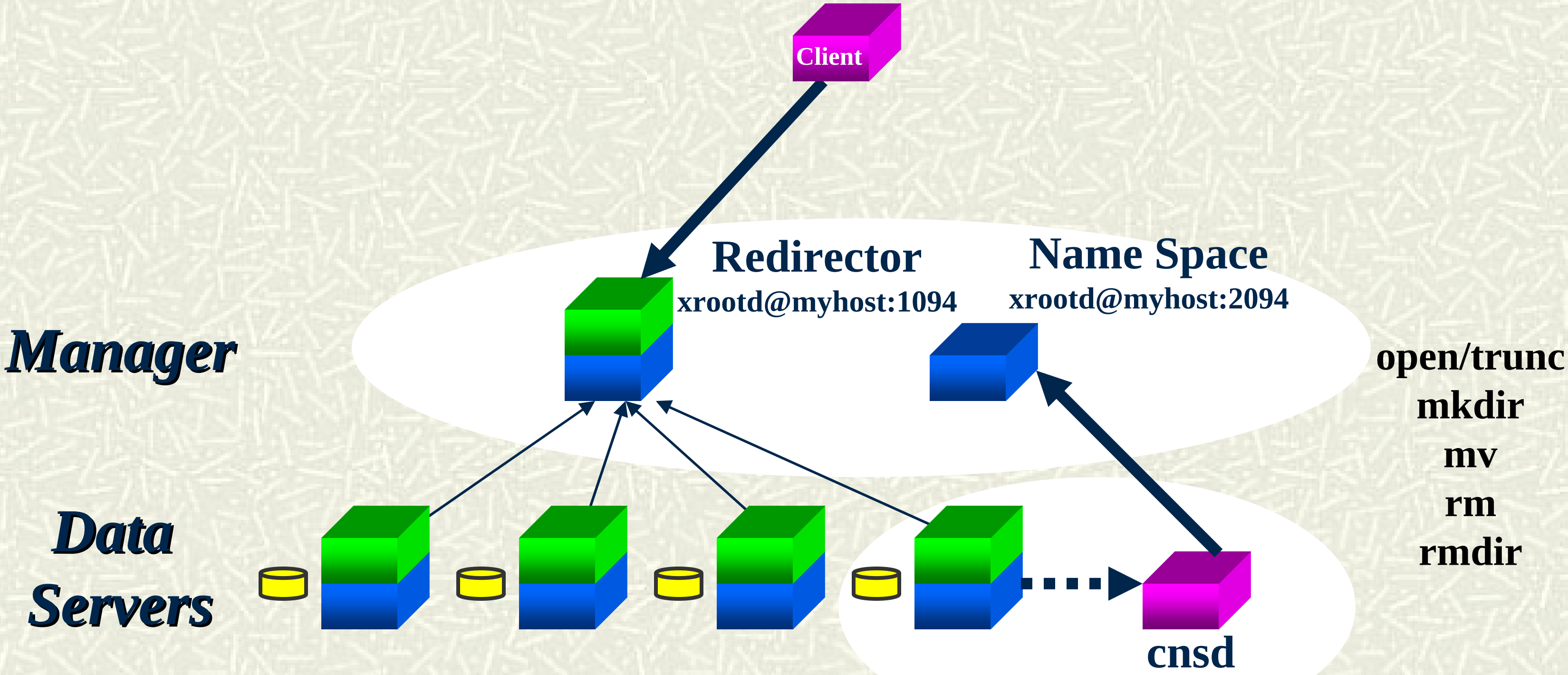
Stanford Linear Accelerator Center

The Distributed Name Space

- **Scalla** implements a distributed name space
  - Very scalable and efficient
  - Sufficient for data analysis
- Some users and applications (e.g., SRM) rely on a centralized name space
  - Spurred the development of a Composite Name Space (cnsd) add-on
    - Simplest solution with the least entanglement

Composite Cluster Name Space

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



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

Stanford Linear Accelerator Center

cnsd Specifics

- Servers direct name space actions to common xrootd(s)
  - Common xrootd maintains composite name space
    - Typically, these run on the redirector nodes
  - Name space replicated in the file system
    - No external database needed
    - Small disk footprint
  - Deployed at SLAC for Atlas
    - Needs synchronization utilities, more documentation, and packaging
      - See Wei Yang for details
    - Similar mySQL based system being considered by CERN/Atlas
      - Annabelle Leung <annabelle.leung@cern.ch>

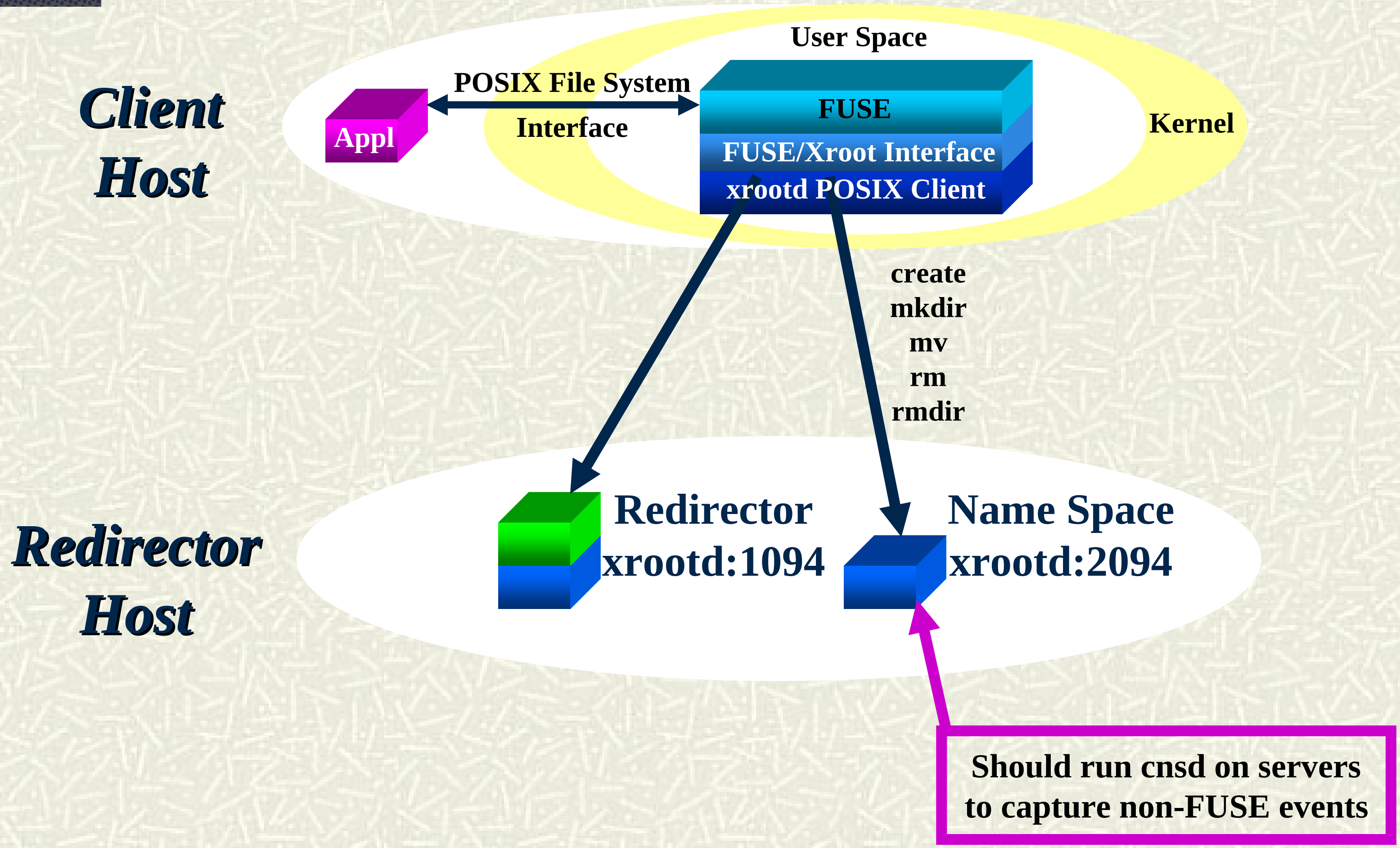
Data System vs File System

- **Scalla** is a data access system
  - Some users/applications want file system semantics
    - More transparent but many times less scalable
- For years users have asked ….
  - Can **Scalla** create a file system experience?

What is **FUSE**

- **F**ilesystem in **U**ser**s**pac**e**
  - Used to implement a file system in a user space program
    - Linux 2.4 and 2.6 only
    - Refer to http://fuse.sourceforge.net/
  - Can use **FUSE** to provide xrootd access
    - Looks like a mounted file system
  - SLAC and FZK have xrootd-based versions of this
    - Wei Yang at SLAC
      - Tested and practically fully functional
    - Andreas Petzold at FZK
      - In alpha test, not fully functional yet

XrootdFS (Linux/FUSE/Xrootd)

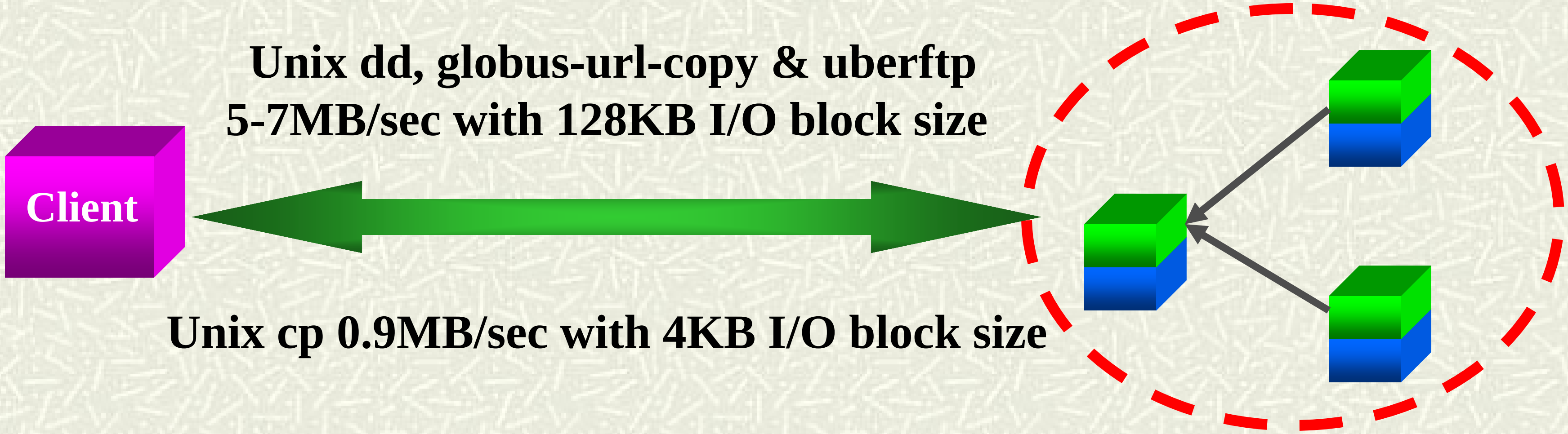


Stanford Linear Accelerator Center

XrootdFS Performance

Sun V20z
RHEL4
2x 2.2Ghz AMD Opteron
4GB RAM
1Gbit/sec Ethernet

VA Linux 1220
RHEL3
2x 866Mhz Pentium 3
1GB RAM
100Mbit/sec Ethernet



Conclusion: Better for some things than others.

Stanford Linear Accelerator Center

Why XrootdFS?

- Makes some things much simpler
  - Most SRM implementations run transparently
  - Avoid pre-load library worries
- But impacts other things
  - Performance is limited
    - Kernel-**FUSE** interactions are not cheap
    - Rapid file creation (e.g., tar) is limited
  - **FUSE** must be administratively installed to be used
    - Difficult if involves many machines (e.g., batch workers)
    - Easier if it involves an SE node (i.e., SRM gateway)

What does this buy you?

- Generally compatible SRM support
  - Integrated with the LBNL bestman SRM
    - Interoperable with all current SRM implementations
- Supports static SRM space tokens
  - Fully integrated with xrootd
  - Quotas applied at the Fuse layer
    - xrootd keeps track of usage by space token
- Fully operational for US Atlas

Next Generation Clustering

- Cluster Management Service (cmsd)
  - Functionally replaces olbd
    - Compatible with olbd config file
      - Unless you are using deprecated directives
    - Straight forward migration
      - Either run olbd or cmsd everywhere
  - Currently in being deployed
    - Alice & US Atlas
    - Available in CVS head
    - Documentation on web site

cmsd Advantages I

- New protocol
  - Compact binary format
    - 8-byte request/response header
  - Architected for minimal data copying
  - Eliminates data conversions between xrootd/cmsd
  - Symmetric parameterized build/parse object
    - Easy to maintain and add new request types
    - Allows central dispatching of sync/async requests

cmsd Advantages II

- Much lower latency
  - New protocol reduces processing time
  - New super fast light-weight location cache
  - State echoing to avoid repeat calculations
  - Verifiable pointers for shorter lock duration
  - Deferred server pinning for non-I/O requests
    - Fast prepare, locate, stat, etc.
  - Fully threaded architecture

cmsd Advantages III

- Better fault detection and recovery
  - Constant algorithm keeps track of needed re-queries
    - Provides linear performance w.r.t. cache size
    - Provides parallel redirect whenever possible
  - Recognition of access conflicts
    - E.g., write access when more than one copy exists
  - Suspend event forwarding
    - The xrootd redirector handles service suspensions

Stanford Linear Accelerator Center

cmsd Advantages IVa

- Added functionality
  - Global clusters
  - Authentication
    - Uses standard xrootd framework
  - Uniform handling of opaque (i.e., cgi) information
    - Available to all plug-ins
  - More meaningful space controls
    - Sensitive to server capacity
    - Allows space utilization as a selection parameter

cmsd Advantages IVb

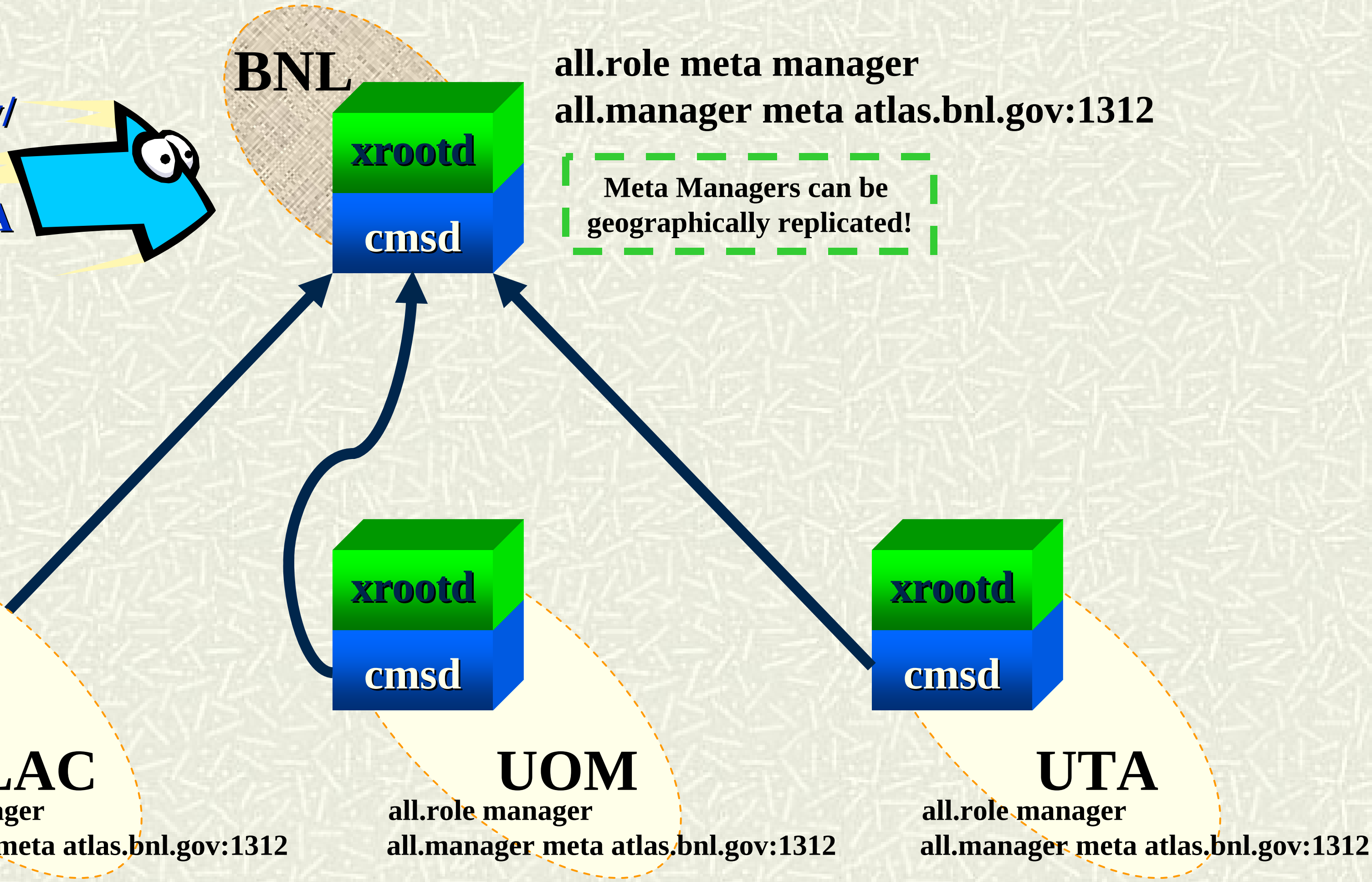
- Added functionality
  - Staging vs Online differentiation
    - Allows selection of online files while staging other copies
  - Complete request forwarding
    - rm, rmdir, mv now forwarded to the xrootd server
  - End-to-end request ID forwarding
    - Automatic whenever tracing enabled
    - Allows tracking client requests throughout the cluster

cmsd Migration

- Basically a rewrite of critical olbd objects
  - Better implementation for reduced maintenance cost
    - The olbd supported only for bug fixes
    - New development is cmsd-focused
- Provides basic backward compatibility
  - Read the migration guide for caveats
- Configuration file compatible
  - If you didn't use deprecated directives

Cluster Globalization

root://atlas.bnl.gov/
includes
SLAC, UOM, UTA
xroot clusters



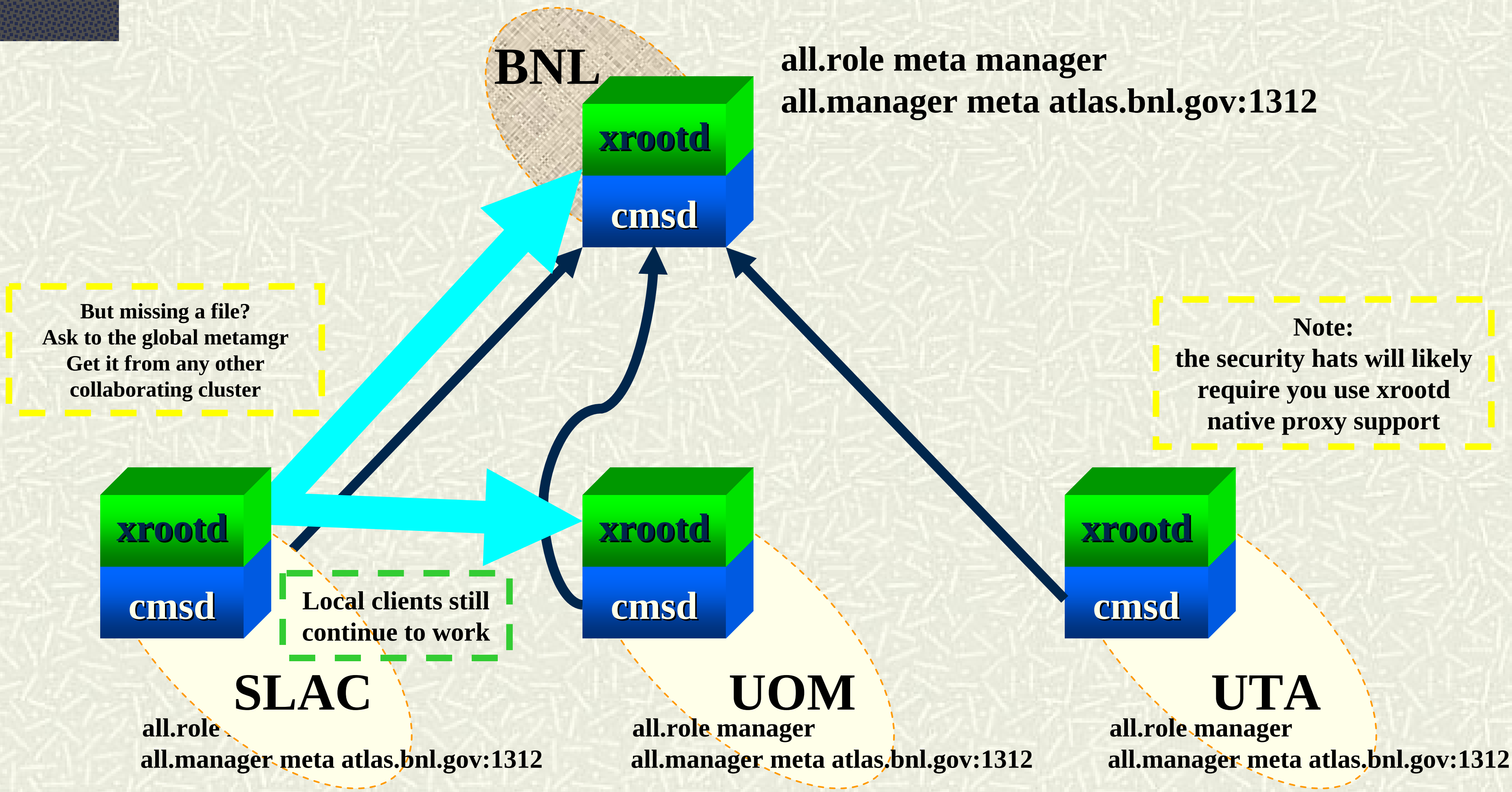
Why Globalize?

- Uniform view of participating clusters
  - Can easily deploy a virtual MSS
    - Included as part of the existing MPS framework
  - Try out real time WAN access
    - You really don't need data everywhere!
- Alice is moving in this direction
  - The non-uniform name space problems solved

Virtual MSS

- Powerful mechanism to increase reliability
  - Data replication load is widely distributed
  - Multiple sites are available for recovery
- Allows virtually unattended operation
  - Based on BaBar experience with real MSS
  - Automatic restore due to server failure
    - Missing files in one cluster fetched from another
      - Typically the fastest one which has the file really online
  - File (pre)fetching on demand
    - Can be transformed into a 3^{rd}-party copy
      - When cmsd is deployed
  - Practically no need to track file location
    - But does not preclude the need for metadata repositories

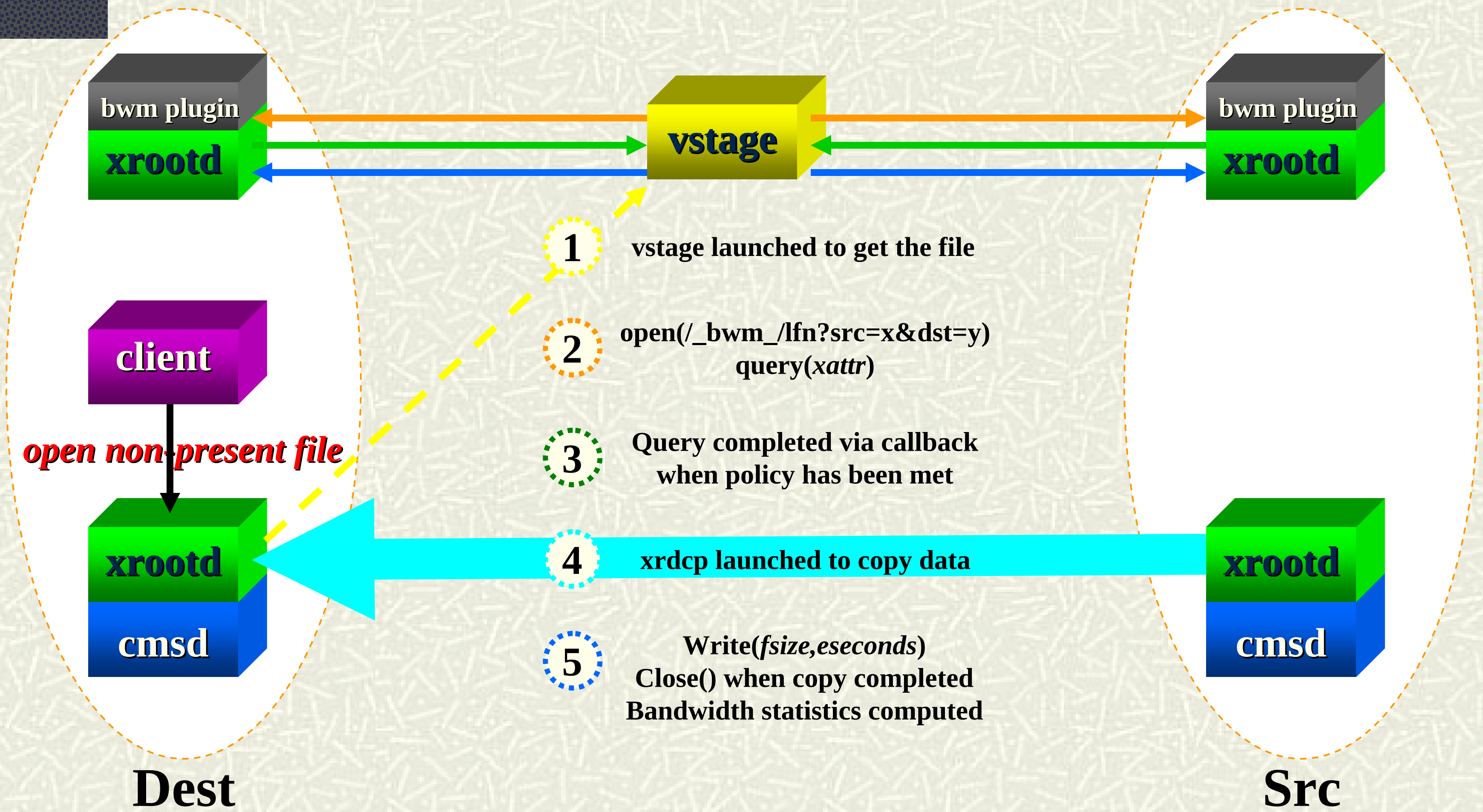
The Virtual MSS Realized



Copying Data Has It's Downside

- Network bandwidth intensive
  - xrootd can blithely use all that is available
- Need extensive bandwidth controls
  - Target domain, dynamic priority, duration, etc.
- Need extensive real-time monitoring
- Points out the need for bandwidth manager
  - Easy to robustly do with **Scalla**
    - Simply use a specialized xrootd server!

The Bandwidth Managers



Why Do It This Way?

- Reuses the Scalla paradigm
  - Lightweight and database-free
    - Very little administrative effort
  - Very little new code
    - Much less to maintain
  - Recovery is automatic when *bwm* fails
    - Can easily have a hot spare
    - Transfers can even cautiously proceed w/o a *bwm*
  - Avoids desperate late-night pages

The Scalla Directed Support Services Architecture

Scalla DSS Advantages

- Advertisement is unnecessary in Scalla DSS
  - Services accessed via well known path prefixes
    - Always accessed via well-known redirectors
      - Redirectors know location of the prefix/service mapping
  - Works even if service is not deployed in a cluster
    - Dynamically deployable and changeable
    - Hot spares managed via DNS aliasing
- Authentication & Authorization available
  - Uses the standard Scalla security framework

Stanford Linear Accelerator Center

Scalla DSS Caveat!

- This is generally a co-operative model
  - It can be bypassed
  - But, it works sufficiently well for >80% of cases
- Can be made mandatory
  - Using signed cookies
    - Usually too expensive and complex to be worthwhile

Stanford Linear Accelerator Center

BWM Configuration

- Minimal Symmetric Configuration
  - In *each* redirector
    - xrootd.redirect *bwmhost:port* /_bwm_/
  - In *each* bwm xrootd
    - xrootd.export /_bwm_/ nolock
    - xrootd.ofslib *bwm_plugin.so*
    - bwm.policy { file *filepath* ***or*** lib *policy_plugin.so* }
    - bwm.log { * ***or*** |*program* }
      - qtod, resptod, endtod, id, src, dst, lfn, fsize, tsec

Stanford Linear Accelerator Center

BWM Policy

- Default simple endpoint pair scheduling
  - In real-time policy file
    - maxslots *number*
    - endpoint *domain incoming*% *outgoing*% *reserve*%
      - Repeat as necessary
  - Might add additional constraints
    - E.g., maximum bandwidth by endpoint
  - However, complex policies can be implemented
    - Write your own policy plug-in

But… do we really need to copy all this data?

Dumb WAN Access*

- Setup: client at CERN, data at SLAC
  - 164ms RTT time, available bandwidth < 100Mb/s
- Test 1: Read a large ROOT Tree
  - (~300MB, 200k interactions)
    - Expected time: 38000s (latency)+750s (data)+CPU → 10 hrs!
- Test 2: Draw a histogram from that tree data
  - (6k interactions)
    - Measured time 20min
      - Using xrootd with WAN optimizations disabled

*Federico Carminati, *The* ALICE *Computing Status and Readiness*, LHCC, November 2007

Smart WAN Access*

- Exploit xrootd WAN Optimizations
  - TCP multi-streaming: for up to 15x improvement data WAN throughput
  - The ROOT TTreeCache provides the hints on ”future” data accesses
  - TXNetFile/XrdClient ”slide through” keeping the network pipeline full
- Data transfer goes in parallel with computation
  - Throughput improvement comparable to “batch” file-copy tools
    - 70-80%, improvement and we are doing a live analysis, not a file copy!
- Test 1 actual time: 60-70 seconds
  - Compared to 30 seconds using a Gb LAN
    - Very favorable for sparsely used files
- Test 2 actual time: 7-8 seconds
  - Comparable to LAN performance
    - 100x improvement over dumb WAN access (i.e., 20 minutes)

*Federico Carminati, *The ALICE Computing Status and Readiness*, LHCC, November 2007

Announcements!

- The CERN-based Scalla web page is online!
  - http://savannah.cern.ch/projects/xrootd/
- Scalla CVS repository is going public!
  - Will be located in afs with unrestricted read access
  - Planning on providing web access

Conclusion

- Scalla is a robust framework
  - Elaborative
    - Composite Name Space
    - XrootdFS
    - SRM
  - Extensible
    - Cluster globalization
    - Bandwidth scheduling
- Many opportunities to enhance data analysis
  - Simplicity, Speed and Efficiency

Stanford Linear Accelerator Center

Acknowledgements

- Current software Collaborators
  - Andy Hanushevsky, Fabrizio Furano
  - Root: Fons Rademakers, Gerri Ganis (security), Bertrand Bellenot (windows)
  - Alice: Derek Feichtinger, Andreas Peters, Guenter Kickinger
  - STAR/BNL: Pavel Jackl, Jerome Lauret
  - SLAC: Jacek Becla, Tofigh Azemoon, Wilko Kroeger
- Operational collaborators
  - BNL, CERN, CNAF, FZK, INFN, IN2P3, GSI, RAL, SLAC
- SLAC Funding
  - US Department of Energy
    - Contract DE-AC02-76SF00515 with Stanford University