Scalla

Andrew Hanushevsky

Stanford Linear Accelerator Center
Stanford University
8-December-06
http://xrootd.slac.stanford.edu

Stanford Linear Accelerator Center

Outline

- Introduction
  - History
  - Design points
  - Architecture
- Data serving
  - xrootd
- Clustering
  - olbd
- SRM Access
- 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
      - By default, arbitrary data organized as files
      - Hierarchical directory-like name space
    - Protocol includes high performance features
  - Structured Clustering provided by olbd servers
    - Exponentially scalable and self organizing

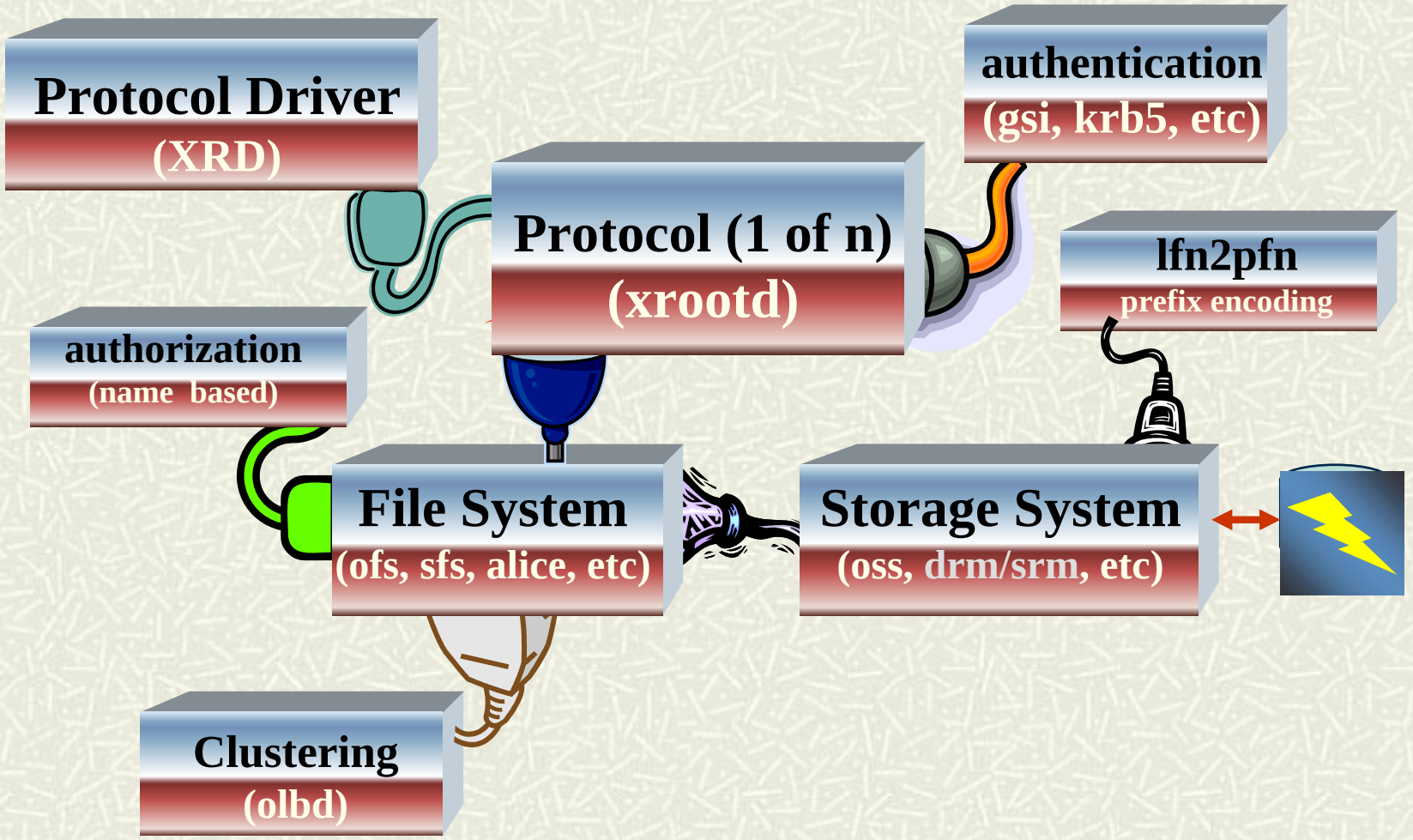
Brief History

- 1997 – Objectivity, Inc. collaboration
  - Design & Development to scale Objectivity/DB
    - First attempt to use commercial DB for Physics data
    - Very successful but problematical
- 2001 – BaBar decides to use root framework
  - Collaboration with INFN, Padova & SLAC
    - Design & develop high performance data access
    - Work based on what we learned with Objectivity
- 2003 – First deployment of xrootd system at SLAC
- 2005 – Collaboration extended
  - Root collaboration & Alice LHC experiment, CERN
  - CNAF, Bologna, It; Cornell University, FZK, De; IN2P3, Fr; INFN, Padova, It; RAL, UK; SLAC are current production deployment sites

The Scalla Design Point

- Write once read many times processing mode
  - Can capitalize on simplified semantics
- Large scale small block sparse random access
  - Needs very low latency per request
- Large compute investment
  - Needs high degree of fault-tolerance
- More data than disk space
  - Must accommodate offline storage (Mass Storage System)
- Highly distributed environment
  - Component based system (replaceable objects)
  - Simple setup with no 3rd party requirements

xrootd Plugin Architecture



Making the Server Perform I

- Protocol is a key component in performance
  - Compact & efficient protocol
    - Minimal request/response overhead (24/8 bytes)
    - Minimal encoding/decoding (network ordered binary)
    - Parallel requests on a single client stream
  - High degree of server-side flexibility
    - Request & response reordering
    - Dynamic transfer size selection
  - Rich set of operations
    - Allows hints for improved performance
      - Pre-read, prepare, client access & processing hints
      - Especially important for accessing offline storage
  - Integrated peer-to-peer clustering
    - Inherent scaling and fault tolerance

Making the Server Perform II

- Short code paths critical
  - Massively threaded design
    - Avoids synchronization bottlenecks
    - Adapts well to next generation multi-core chips
  - Internal wormhole mechanisms
    - Minimizes code paths in a multi-layered design
      - Does not flatten the overall architecture
  - Use the most efficient OS-specific system interfaces
    - Dynamic and compile-time selection
      - Dynamic: aio_read() vs read()
      - Compile-time: /dev/poll or kqueue() vs poll() or select()

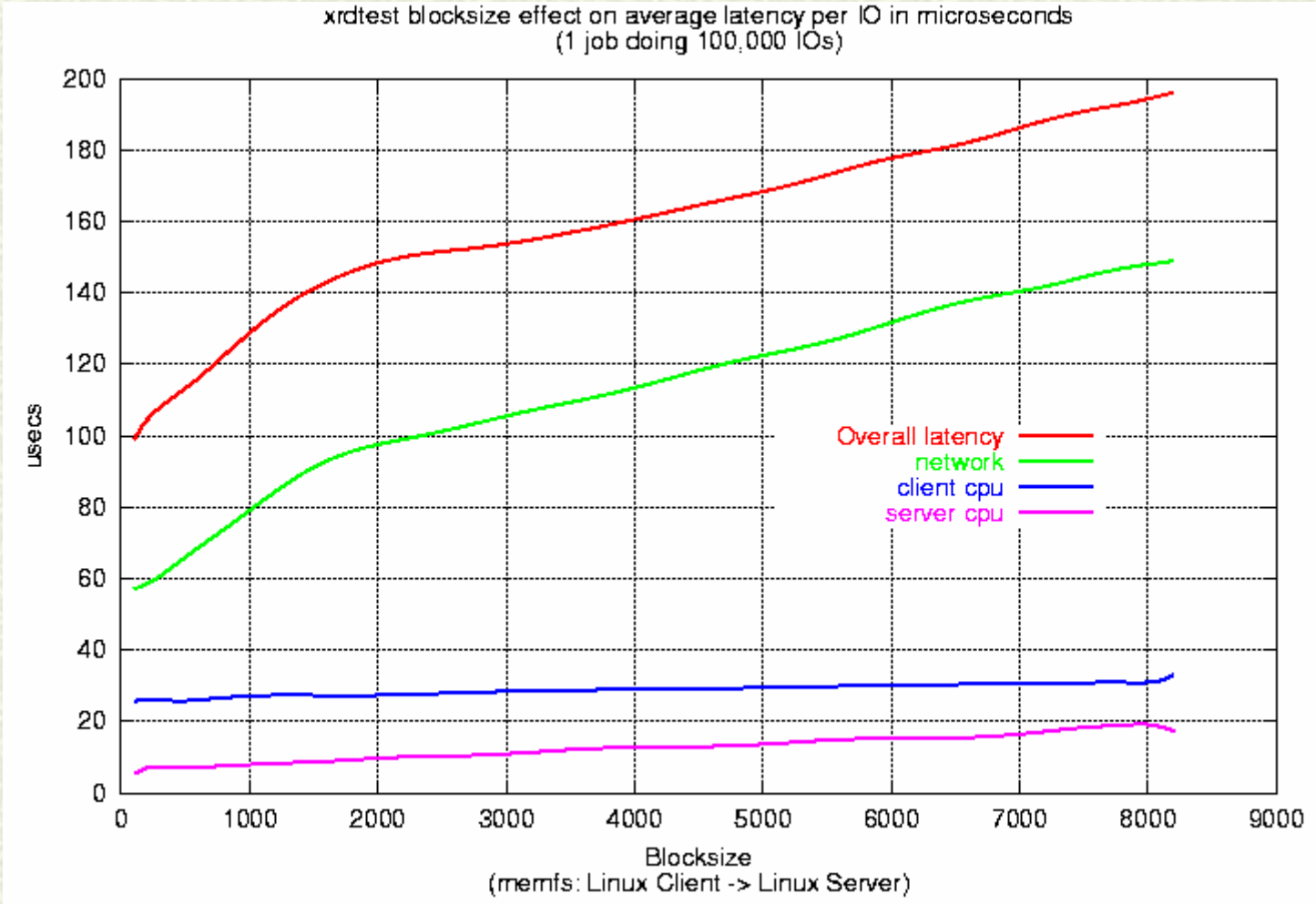
Stanford Linear Accelerator Center

Making the Server Perform III

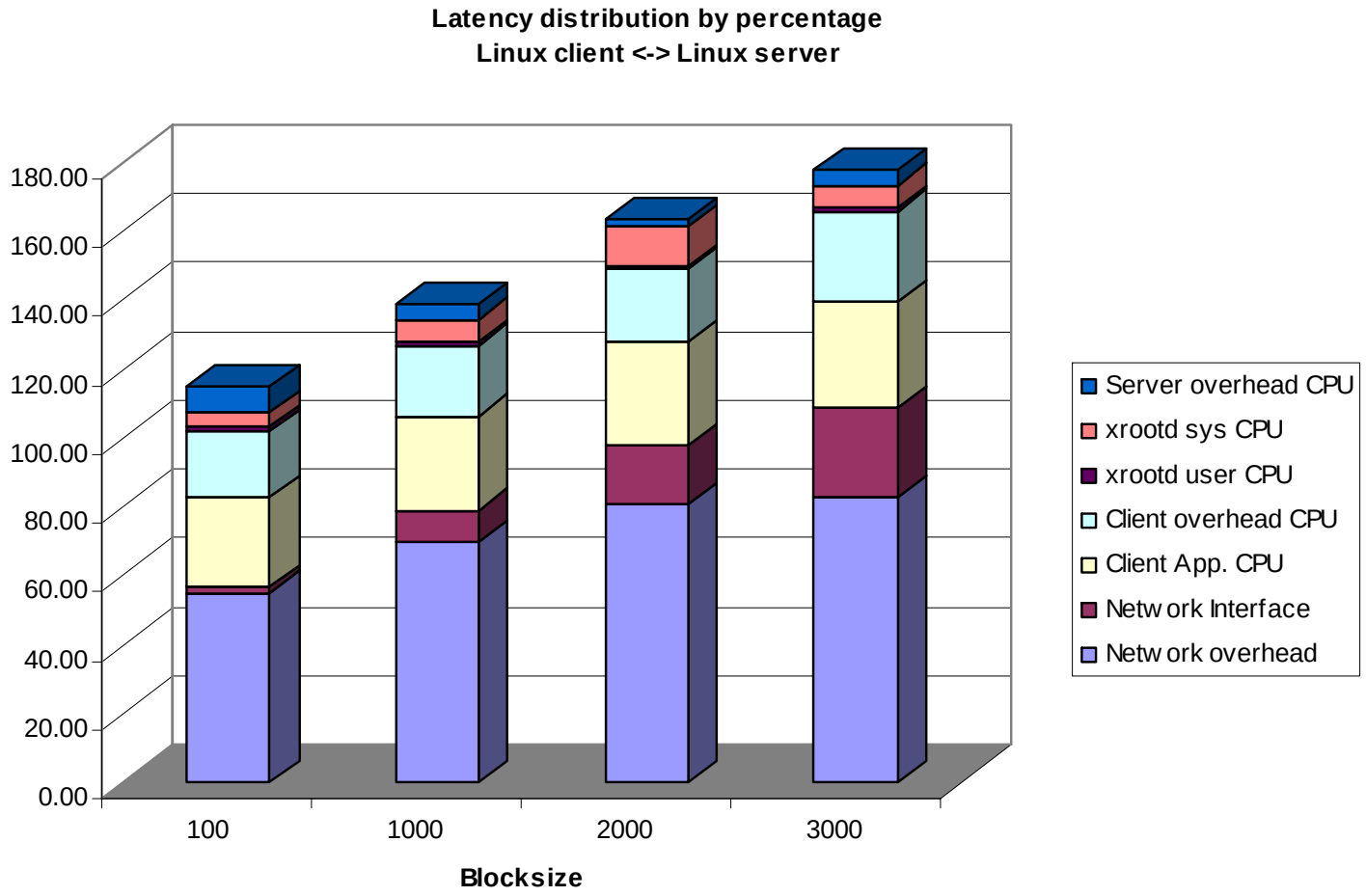
- Intelligent memory management
  - Minimize cross-thread shared objects
    - Avoids thrashing the processor cache
  - Maximize object re-use
    - Less fragmenting the free space heap
    - Avoids major serialization bottleneck (malloc)
  - Load adaptive I/O buffer management
    - Minimize server growth to avoid paging

Latency

Sun V20z
Dual 1.8 GHz
Opterons
1Gb NIC

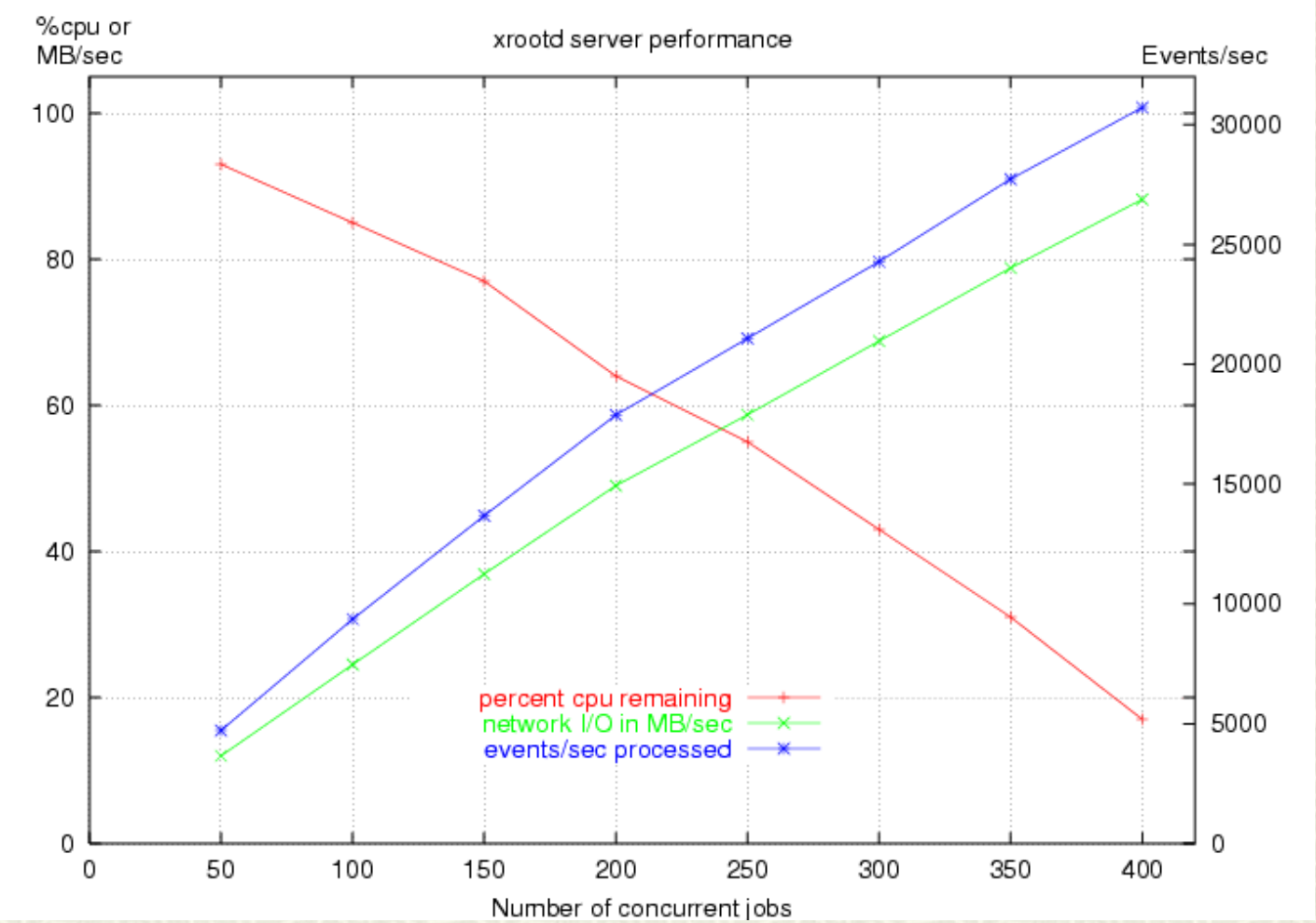


Overhead Distribution



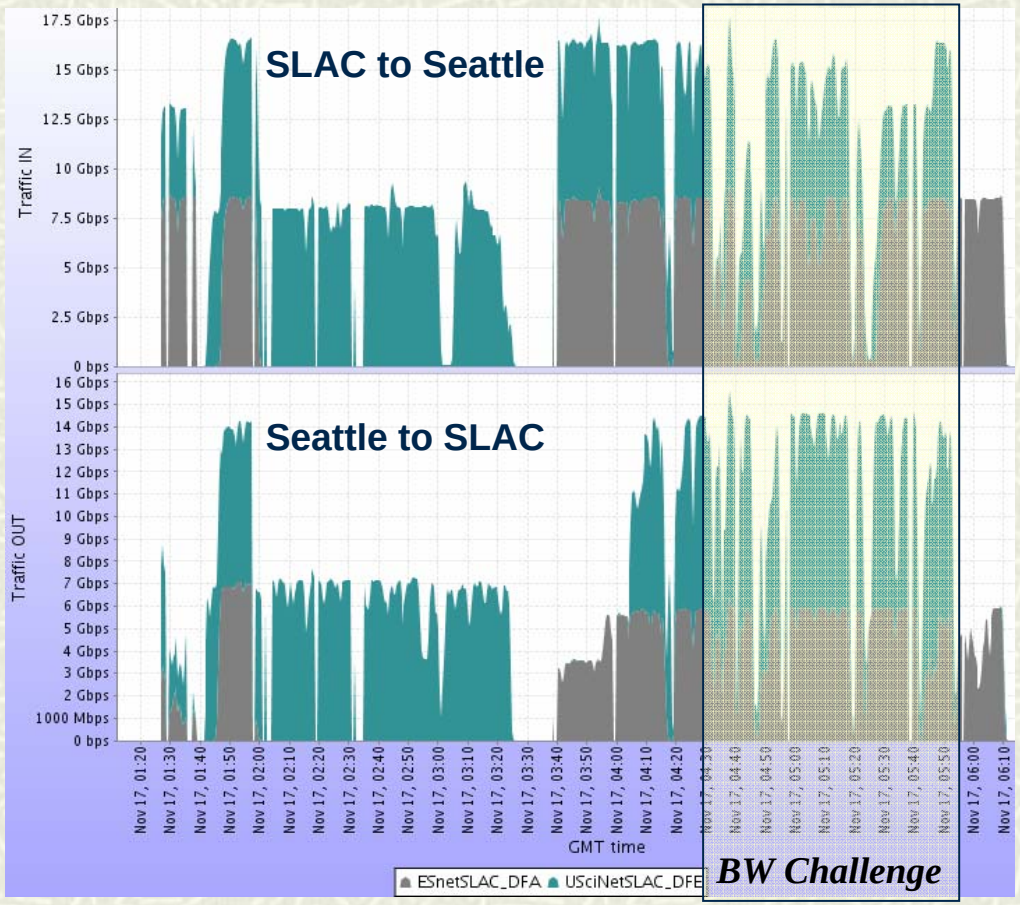
Scaling

Sun V20z
Dual 1.8 GHz
Opterons
1Gb NIC



I/O Bandwidth (wide area network)

- SC2005 BW Challenge
  - Latency ⇔ Bandwidth
- 8 xrootd Servers
  - 4@SLAC & 4@Seattle
  - Sun V20z w/ 10Gb NIC
  - Dual 1.8/2.6GHz Opterons
  - Linux 2.6.12
- 1,024 Parallel Clients
  - 128 per server
- 35Gb/sec peak
  - Higher speeds killed router
  - 2 full duplex 10Gb/s links
  - Provided 26.7% overall BW
    - BW averaged 106Gb/sec
    - 17 Monitored links total



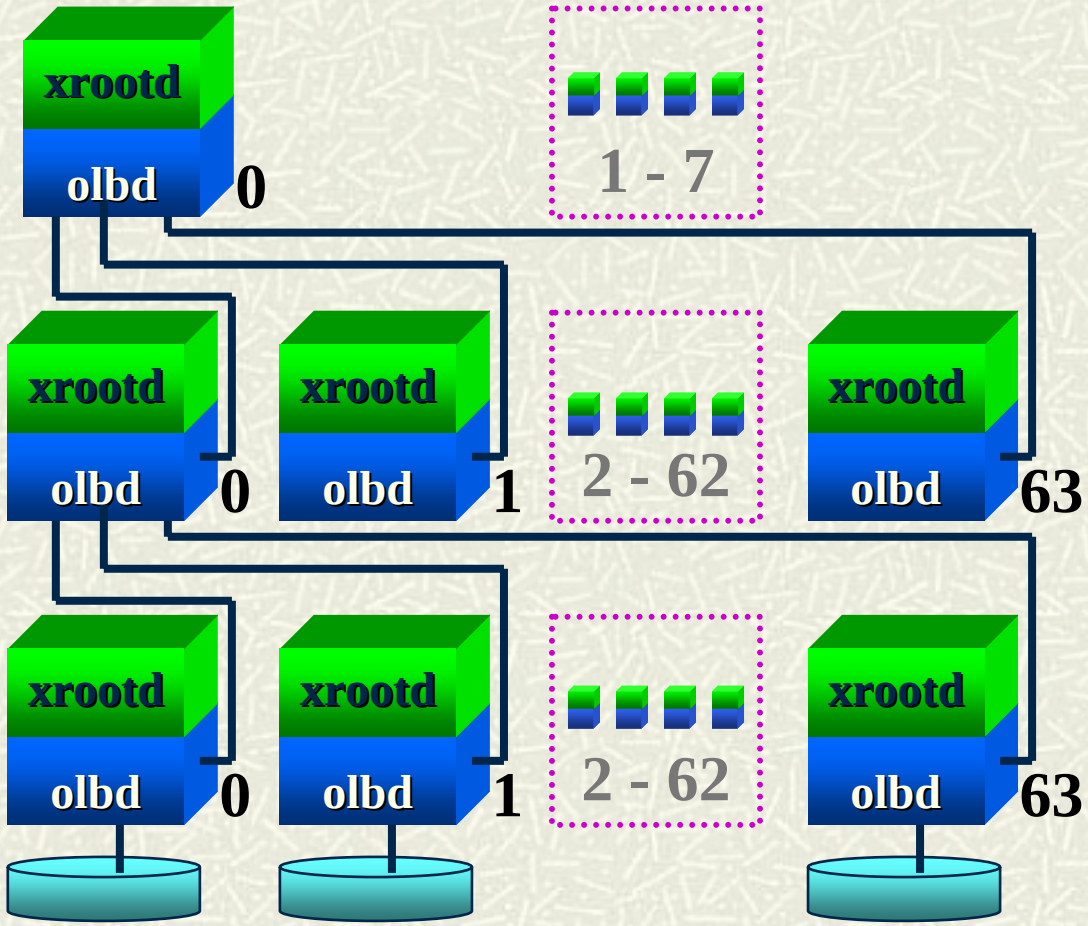
ESnet routed ESnet SDN layer 2 via USN

http://www-iepm.slac.stanford.edu/monitoring/bulk/sc2005/hiperf.html

Clustering

- xrootd servers can be clustered
  - Increase access points and available data
  - Allows for automatic failover
- Structured point-to-point connections
  - Cluster overhead (human & non-human) scales linearly
  - Cluster size is not limited
  - I/O performance is not affected
- Always pairs xrootd & olbd servers
  - Symmetric cookie-cutter arrangement

B-Tree Organization



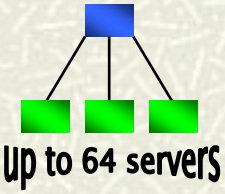
Manager
(Root Nodes)

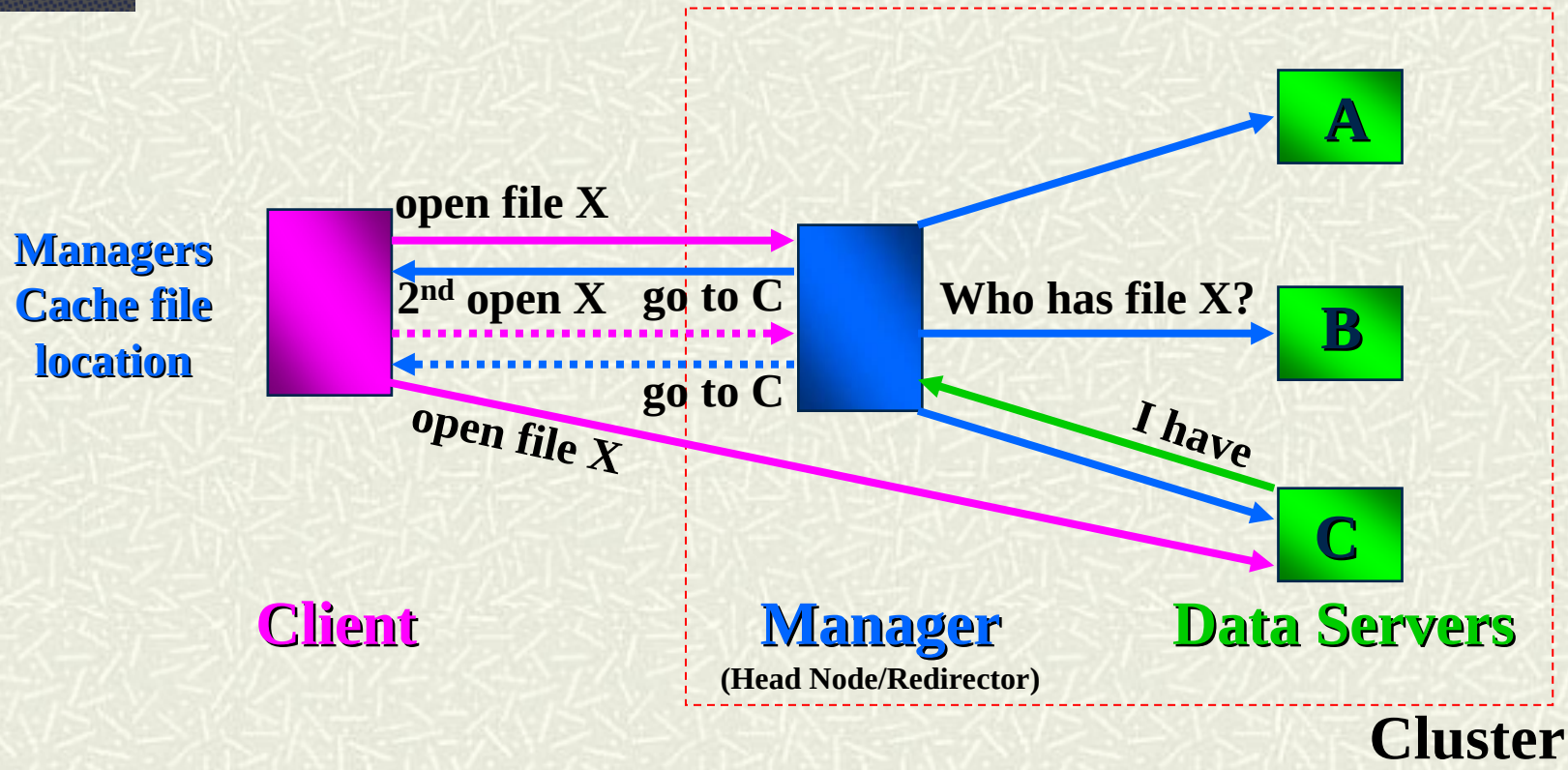
Supervisor
(Intermediate Nodes)

Data Server
(Leaf Nodes)

Stanford Linear Accelerator Center

Single Level Switch



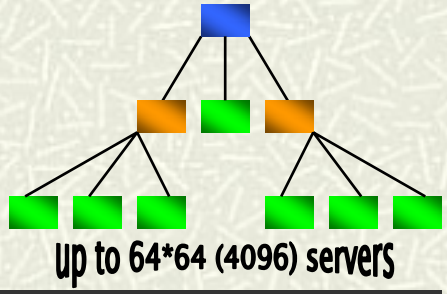


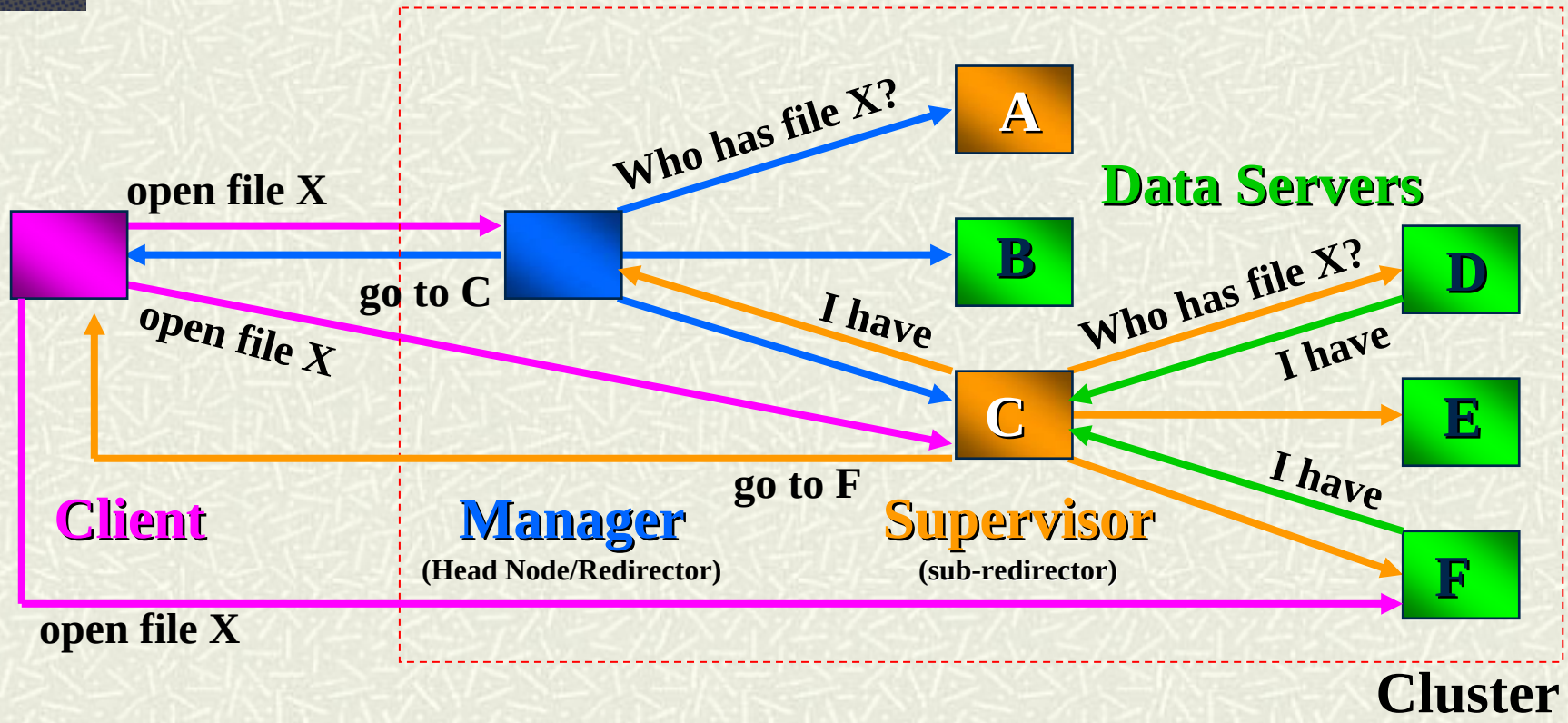
Client sees all servers as xrootd data servers

Efficient Clustering

- Cell size, structure, & search protocol are critical[Ref]
  - Cell Size is 64
    - Limits direct inter-chatter to 64 entities (64x compression)
    - Deterministic bound on amount of work per transaction
    - Can use very efficient 64-bit logical operations
  - Hierarchical B64-Tree structure for efficient scaling
    - Scales 64^h (where h is the tree height)
      - Client needs h-1 hops to find one of 64^h servers (2 hops for 262,144 servers)
      - Number of responses is bounded at each level of the tree
  - Search is a directed broadcast query/rarely respond protocol
    - Provably best scheme[Ref] if less than 50% of servers have the wanted file
      - Generally true if number of files >> cluster capacity
      - Cluster protocol becomes more efficient as cluster grows

Two Level Switch





Client sees all servers as xrootd data servers

Reference I

- Bryan Horling, Roger Mailler, and Victor Lesser; “A Case Study of Organizational Effects in a Distributed Sensor Network”; Proceedings of the 2004 IEEE/WTC/ACM International Conference on Intelligent Agent Technology; Beijing, China; September 20-24, 2004.

Reference II

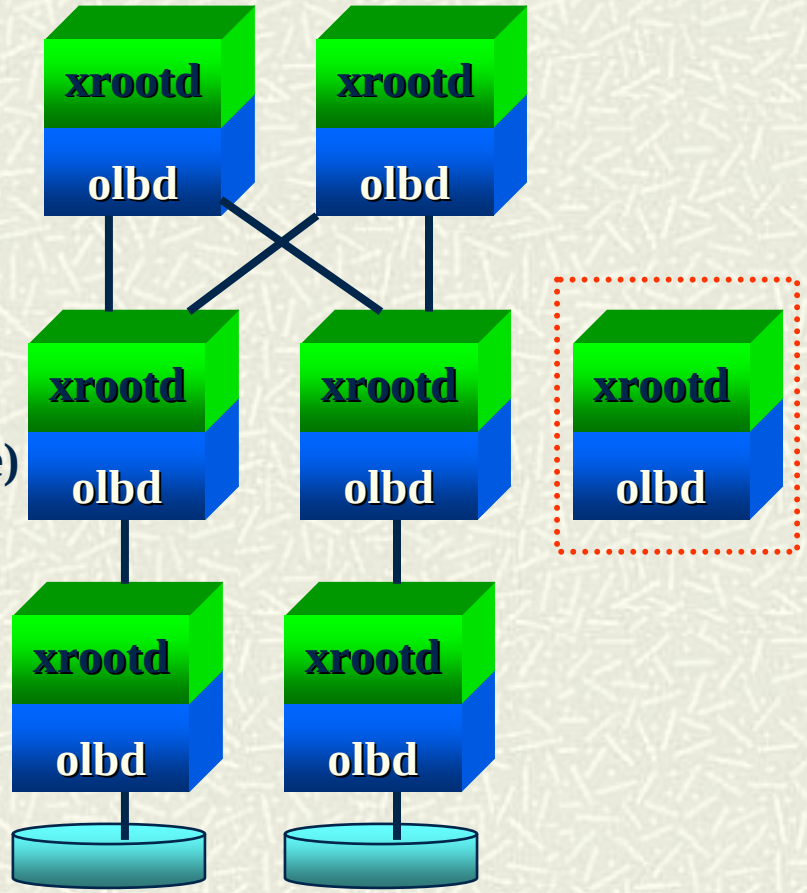
- Fabrizio Furano and Andrew Hanushevsky; “Managing Commitments in a Multi-Agent System using Passive Bids”; Proceedings of the 2005 IEEE/WTC/ACM International Conference on Intelligent Agent Technology; Compiegne, France; September 19-22, 2005.

Providing Fault Tolerance

Manager
(Root Node)

Supervisor
(Intermediate Node)

Data Server
(Leaf Node)

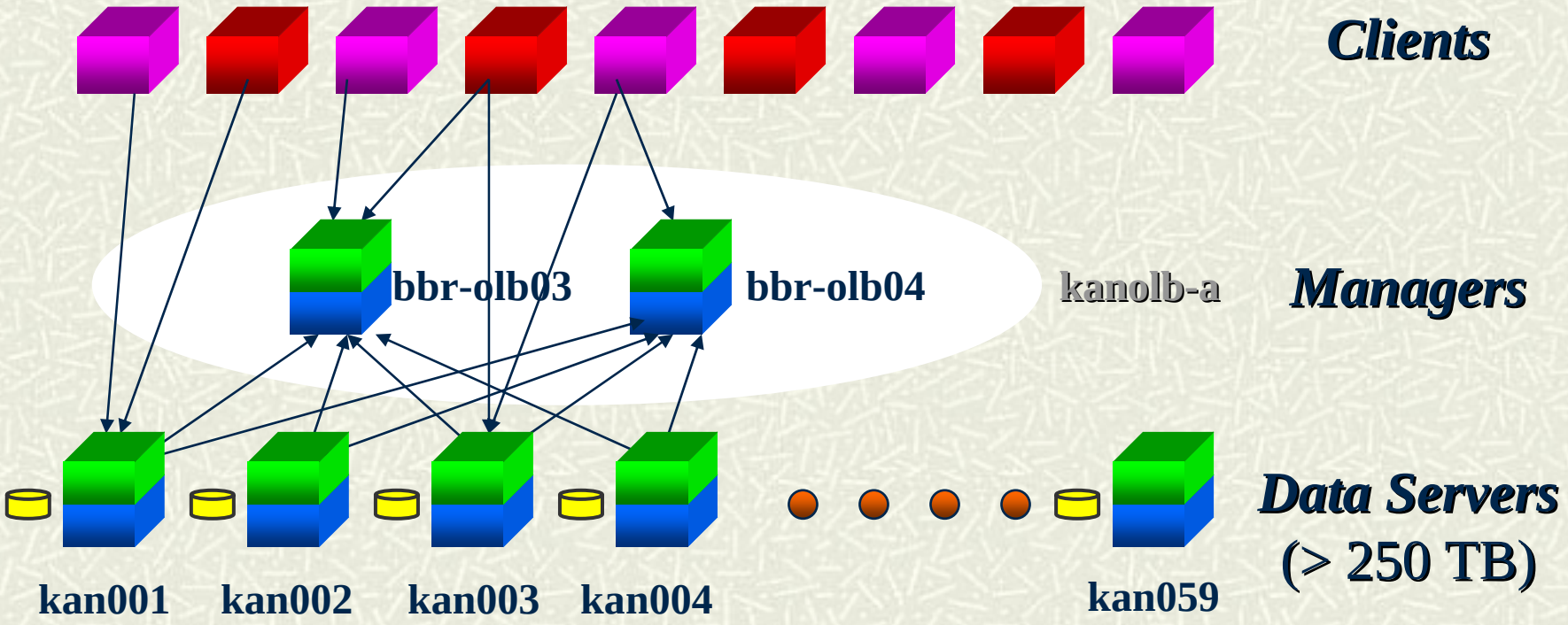


Fully Replicate

Hot Spares

Data Replication
Restaging
Proxy Search

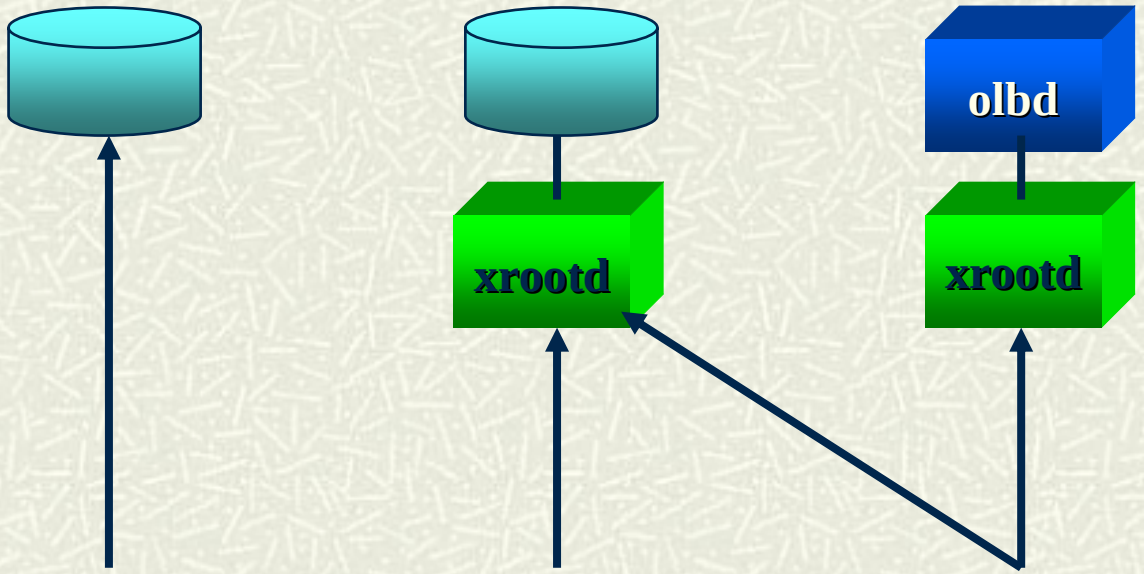
The SLAC ¼PB "kan" Cluster



Scalable Cluster Management

- Massive clusters must be self-managing
  - Scales 64^n where n is height of tree
    - Scales very quickly ($64^2 = 4096$, $64^3 = 262{,}144$)
    - Well beyond direct human management capabilities
  - Therefore clusters self-organize
    - Uses a minimal spanning tree algorithm
      - 280 nodes self-cluster in about 7 seconds
      - 890 nodes self-cluster in about 56 seconds
    - Most overhead is in wait time to prevent thrashing
      - Time can be substantially reduced by ordering connections

Redirection Overhead



Server cache search	1st Open	2nd
Linux	13us	3us
Solaris	43us	4us

1st Open	2nd
4300us	880us
4400us	920us

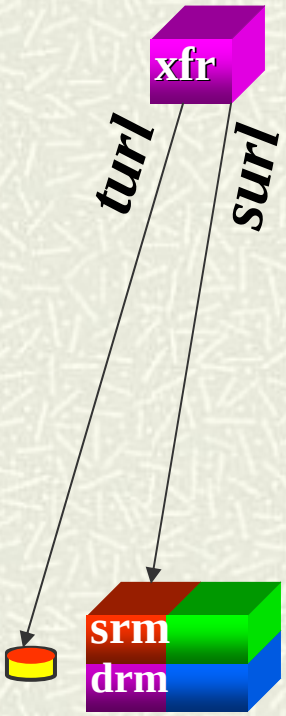
1st Open	2nd
(not tested)	
5580us	1240us

(only xrootd protocol overhead measured)

Clustering Impact

- Redirection overhead must be amortized
  - This is deterministic process for xrootd
    - All I/O is via point-to-point connections
    - Can trivially use single-server performance data
  - Clustering overhead is non-trivial
    - Not good for very small files or short “open” times
  - However, compatible with the HEP access patterns
- Complicates grid access to space
  - But is solvable

Grid Access To Single Server



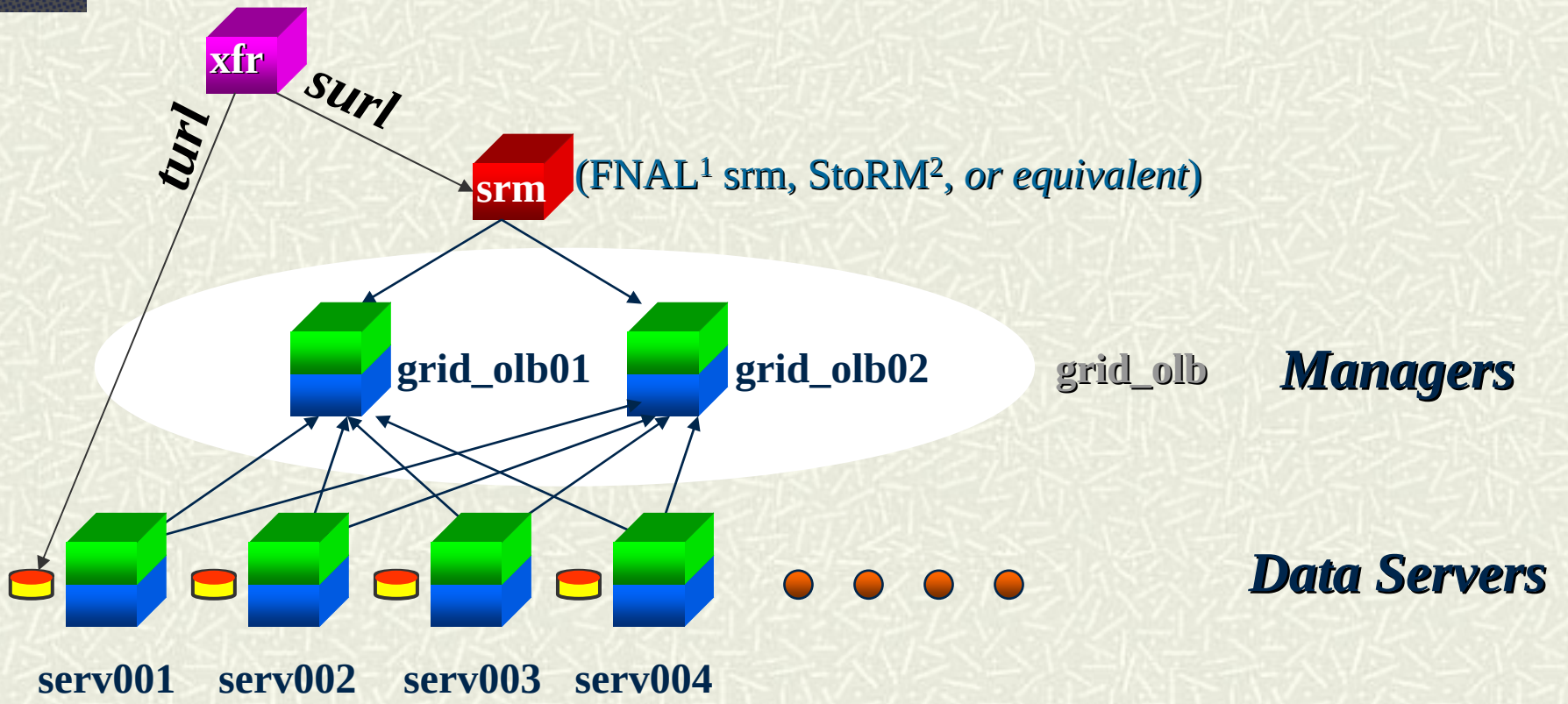
(LBNL[1] srm/drm *or equivalent*)

Data Server

GSIftp

[1]http://sdm.lbl.gov/srm-dist/

Grid Access To Clusters



[1]http://www-isd.fnal.gov/srm/
[2]http://storm.forge.cnaf.infn.it/doku.php

Grid Access to Managed Space

- LCG DPM
  - Includes SRM and Disk Pool Manager
    - xrootd integration completed
- Castor2
  - Includes SRM and Disk/Tape Manager
    - xrootd integration nearing completion
- MPS
  - Part of standard distribution
  - Makes sense when used with external SRM
    - FNAL SRM integration on-going but problematical
    - StoRM integration looks much more promising

Conclusion

- High performance data access systems achievable
  - The devil is in the details
    - Proper protocol, semantics, and engineering critical
- High performance and clustering are synergetic
  - Allows unique performance, usability, scalability, and recoverability characteristics
- SRM access for grid compatibility ongoing
  - Several solutions (none perfect)
  - Some packaging issues remain

Acknowledgements

- Software Collaborators
  - INFN/Padova: Fabrizio Furano (client-side), Alvise Dorigao
  - Root: Fons Rademakers, Gerri Ganis (security), Beterand Bellenet (windows)
  - Alice: Derek Feichtinger, Guenter Kickinger
  - STAR/BNL: Pavel Jackl
  - Cornell: Gregory Sharp
  - SLAC: Jacek Becla, Tofigh Azemoon, Wilko Kroeger, Bill Weeks
  - BaBar: Pete Elmer (packaging)
- Operational collaborators
  - BNL, CNAF, FZK, INFN, IN2P3, RAL, SLAC
- Funding
  - US Department of Energy
    - Contract DE-AC02-76SF00515 with Stanford University