Hyper-Scaling Xrootd Clustering

Andrew Hanushevsky
Stanford Linear Accelerator Center
Stanford University
29-September-2005
http://xrootd.slac.stanford.edu

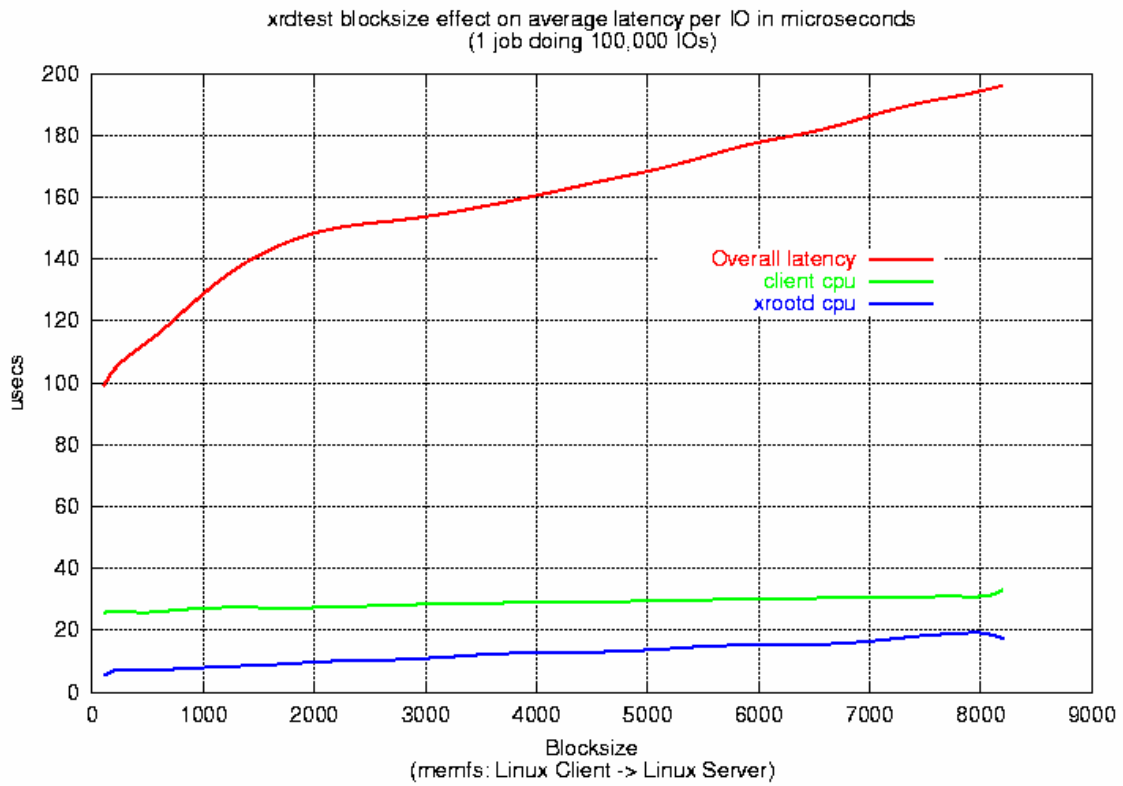
Root 2005 Users Workshop
CERN
September 28-30, 2005

Stanford Linear Accelerator Center

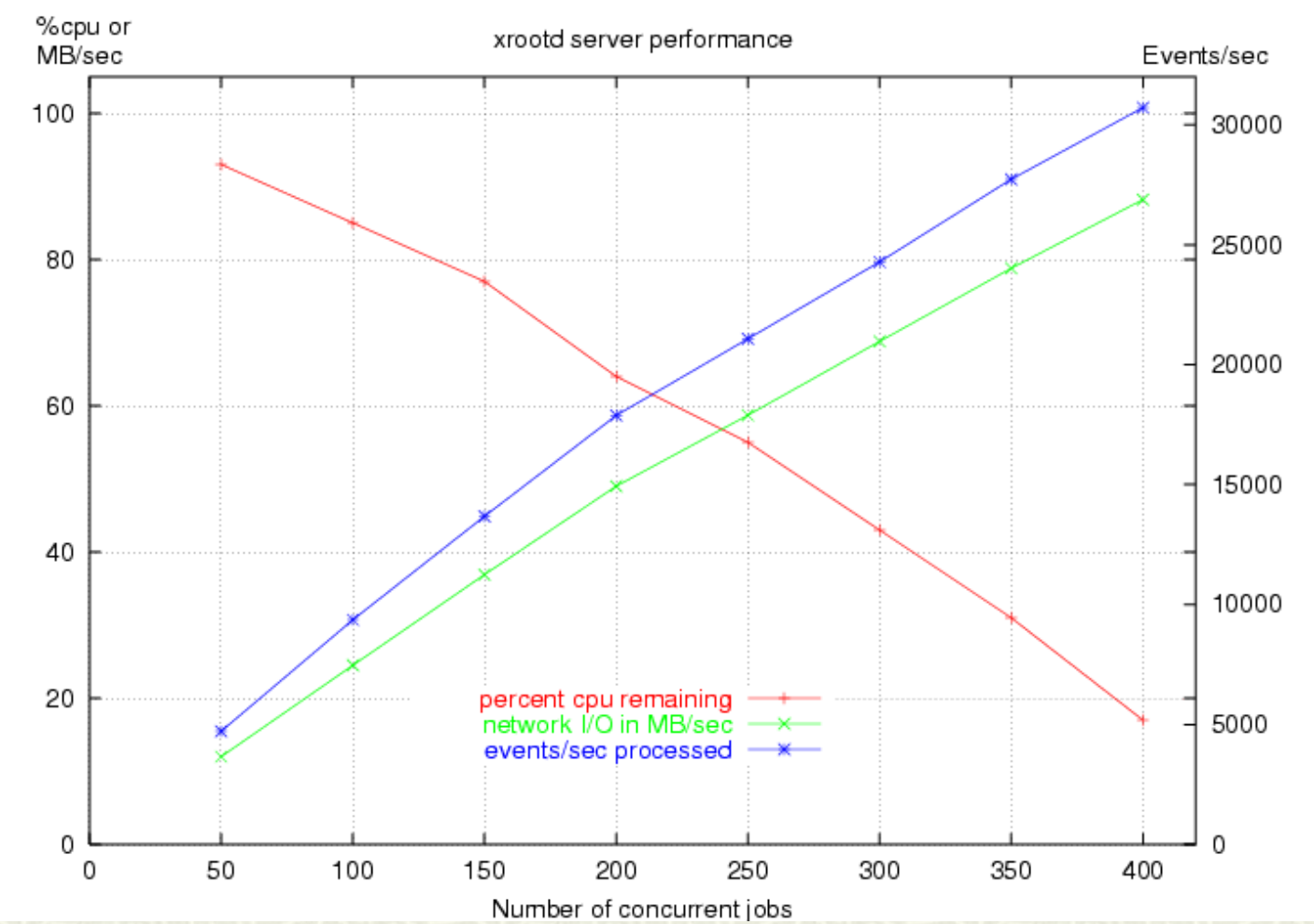
Outline

- Xrootd Single Server Scaling
- Hyper-Scaling via Clustering
  - Architecture
  - Performance
- Configuring Clusters
  - Detailed relationships
  - Example configuration
  - Adding fault-tolerance
- Conclusion

Latency Per Request (xrootd)



Capacity vs Load (xrootd)



xrootd Server Scaling

- Linear scaling relative to load
  - Allows deterministic sizing of server
    - Disk
    - NIC
    - Network Fabric
    - CPU
    - Memory
- Performance tied directly to hardware cost

Stanford Linear Accelerator Center

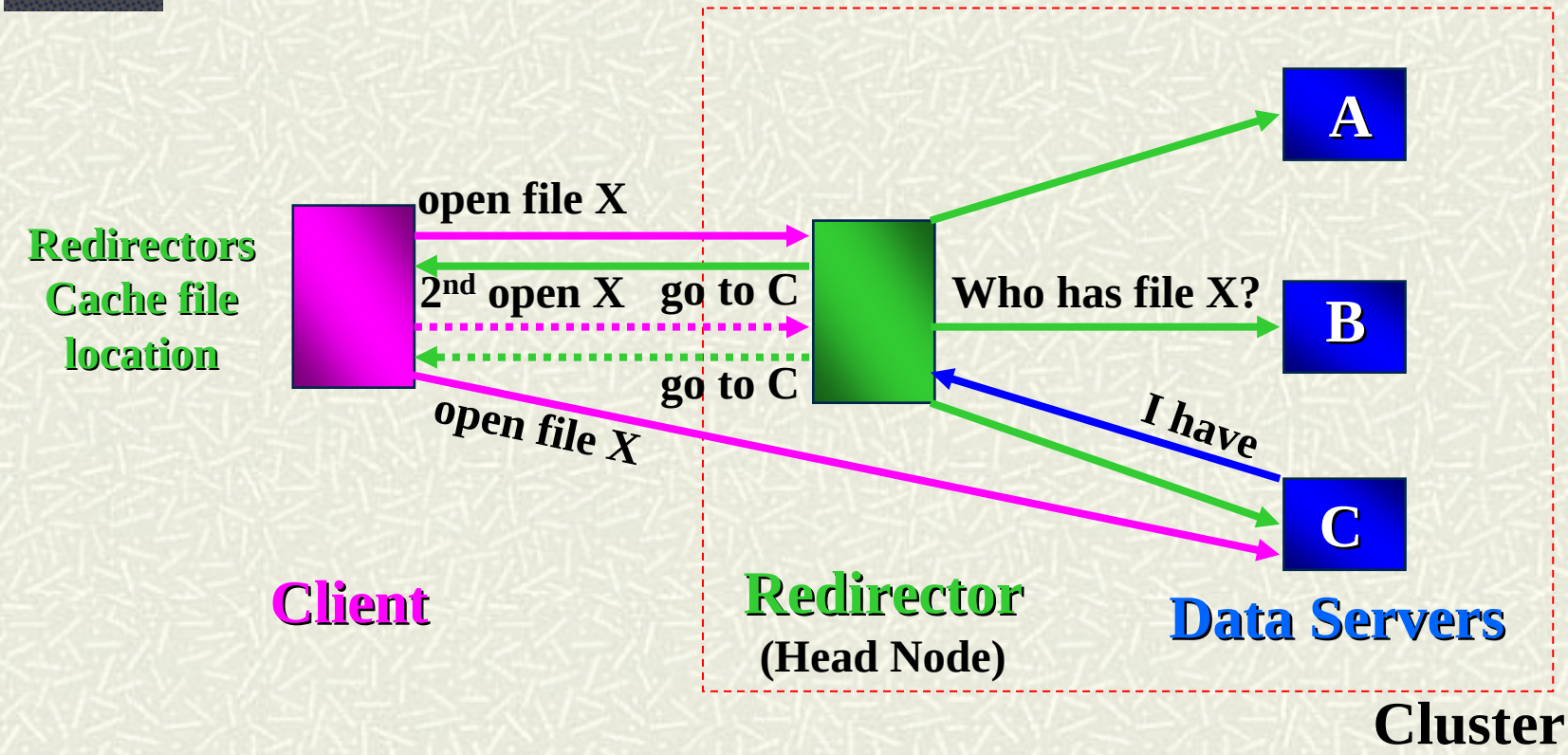
Hyper-Scaling

- xrootd servers can be clustered
  - Increase access points and available data
    - Complete scaling
  - Allow for automatic failover
    - Comprehensive fault-tolerance
- The trick is to do so in a way that
  - Cluster overhead (human & non-human) scales linearly
    - Allows deterministic sizing of cluster
  - Cluster size is not artificially limited
  - I/O performance is not affected

Basic Cluster Architecture

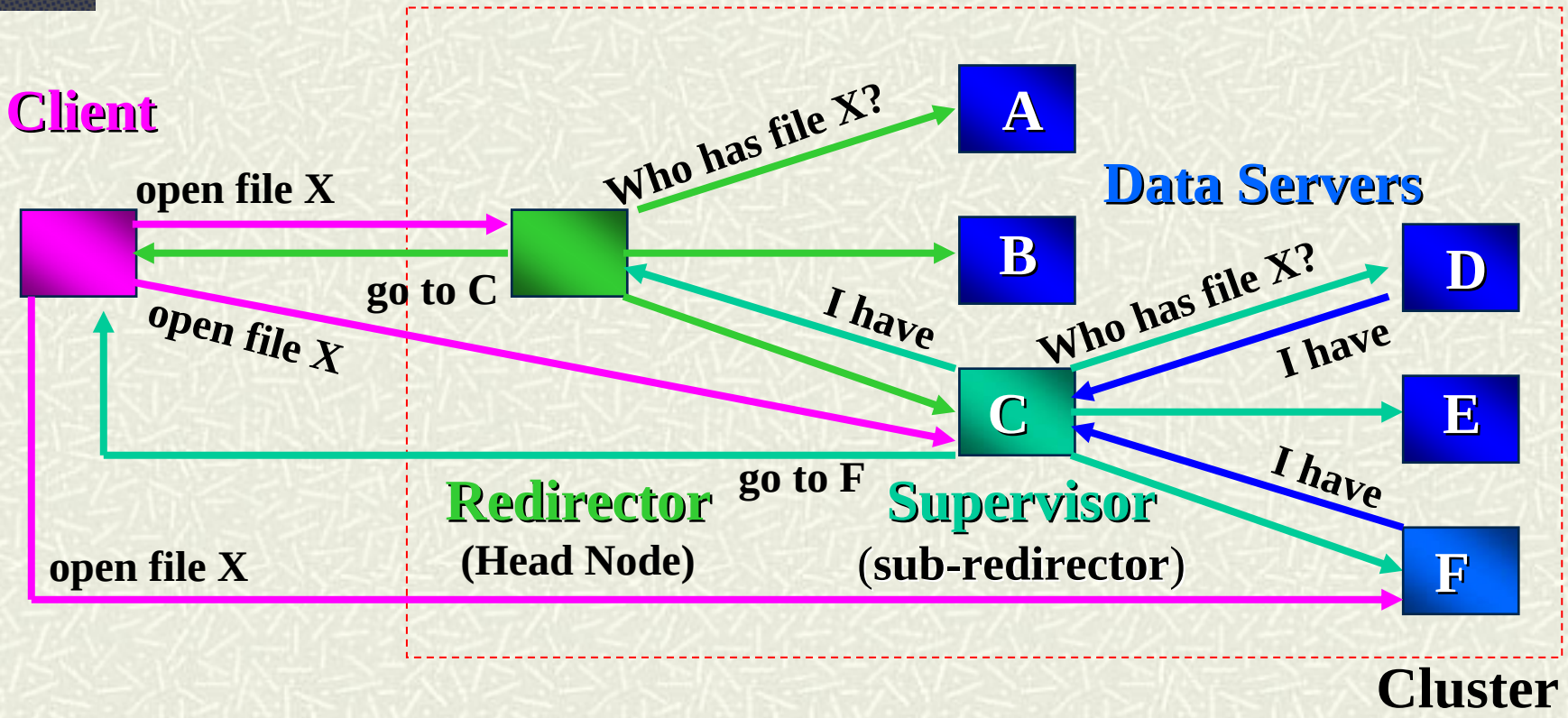
- Software cross bar switch
  - Allows point-to-point connections
    - Client and data server
  - I/O performance not compromised
    - Assuming switch overhead can be amortized
- Scale interconnections by stacking switches
  - Virtually unlimited connection points
    - Switch overhead must be very low

Single Level Switch



Client sees all servers as xrootd data servers

Two Level Switch



Client sees all servers as xrootd data servers

Making Clusters Efficient

- Cell size, structure, & search protocol are critical
  - Cell Size is 64
    - Limits direct inter-chatter to 64 entities
    - Compresses incoming information by up to a factor of 64
    - Can use very efficient 64-bit logical operations
  - Hierarchical structures usually most efficient
    - Cells arranged in a B-Tree (i.e., B64-Tree)
    - 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 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 size increases

Cluster Scale Management

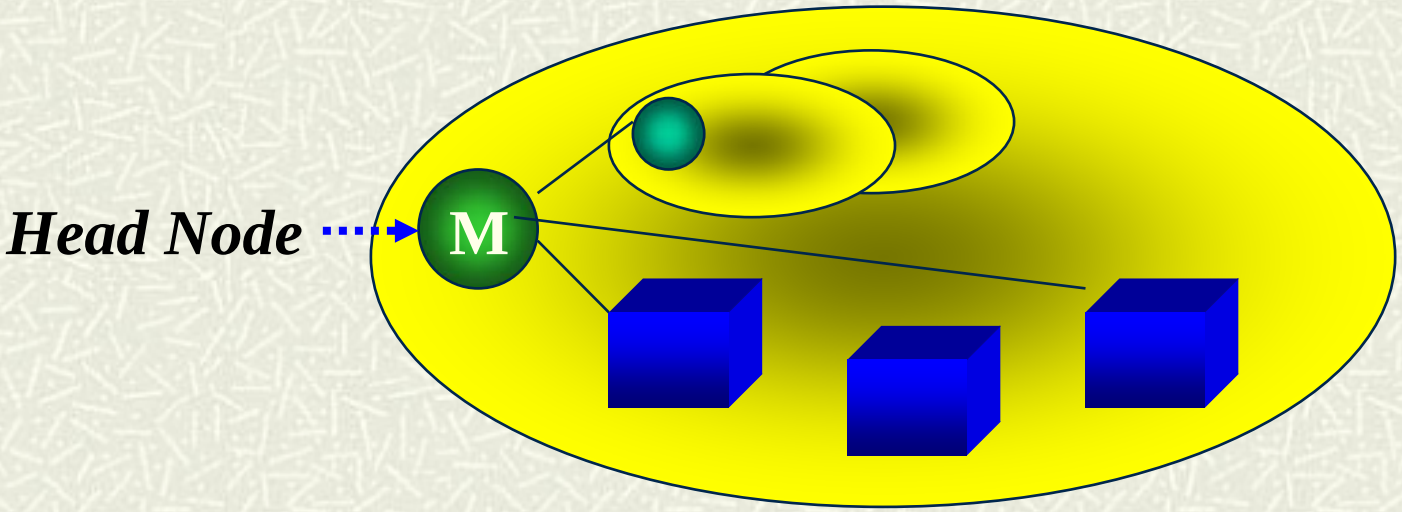
- Massive clusters must be self-managing
  - Scales 64^n where n is height of tree
    - Scales very quickly ($64^2 = 4096$, $64^3 = 262{,}144$)
    - Well beyond direct human management capabilities
  - Therefore clusters self-organize
    - Single configuration file for all nodes
    - Uses a minimal 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

Clustering Impact

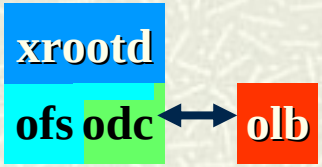
- Redirection overhead must be amortized
  - This is deterministic process for xrootd
    - All I/O is via point-to-point connections
    - Can trivially use single-server performance data
  - Clustering overhead is non-trivial
    - 100-200us additional for an open call
    - Not good for very small files or short “open” times
  - However, compatible with the HEP access patterns

Detailed Cluster Architecture

A cell is 1-to-64 entities (servers or cells)
clustered around a cell manager
The cellular process is self-regulating and creates a
B-64 Tree



The Internal Details

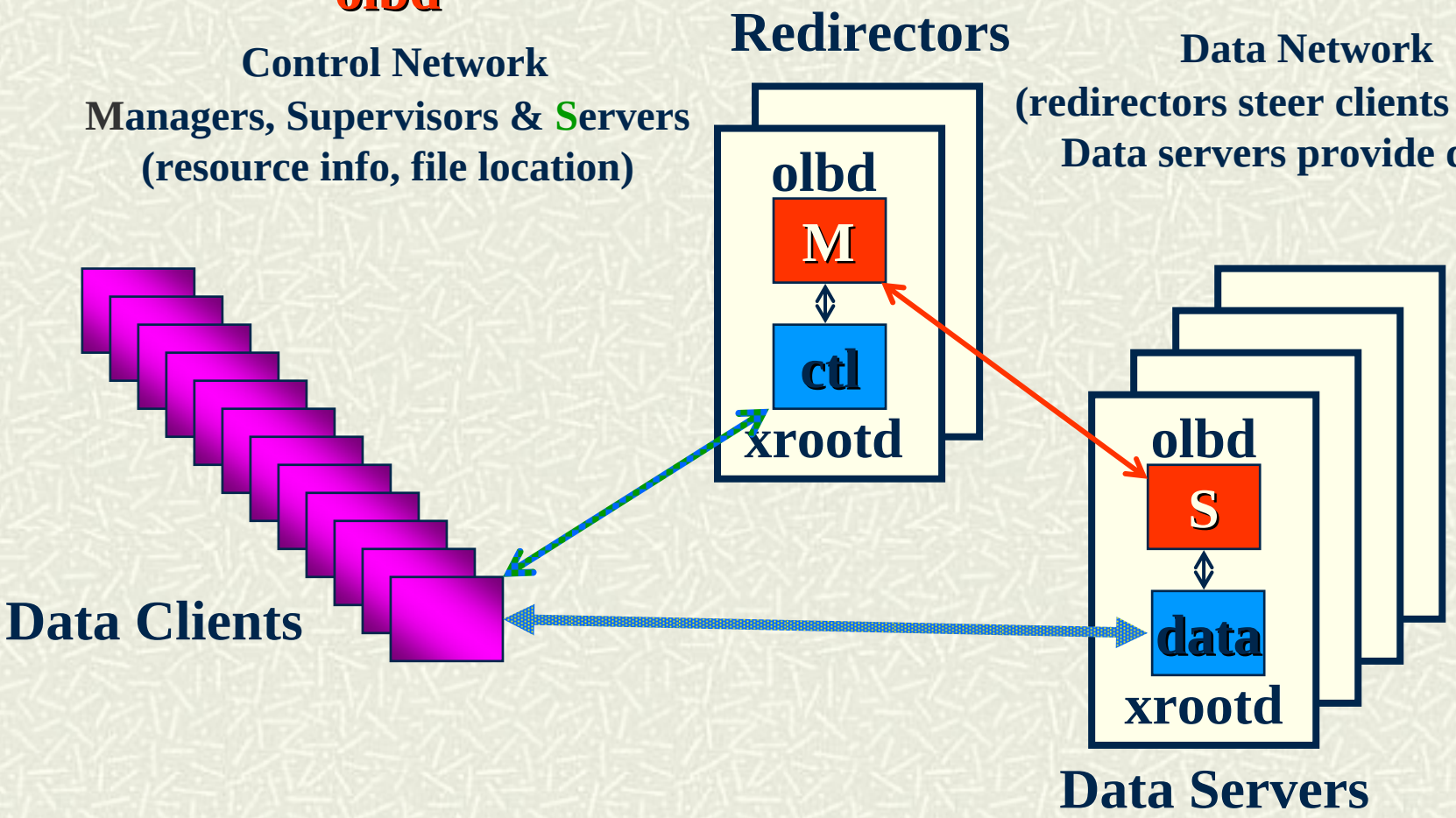


olbd

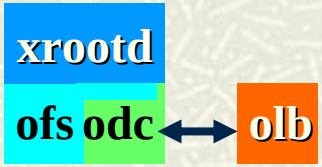
Control Network
Managers, Supervisors & Servers
(resource info, file location)

xrootd

Data Network
(redirectors steer clients to data
Data servers provide data)



Schema Configuration



Redirectors
(Head Node)

ofs.redirect remote
odc.manager *host port*

olb.role manager
olb.port *port*
olb.allow *hostpat*

Supervisors
(sub-redirector)

ofs.redirect remote
ofs.redirect target

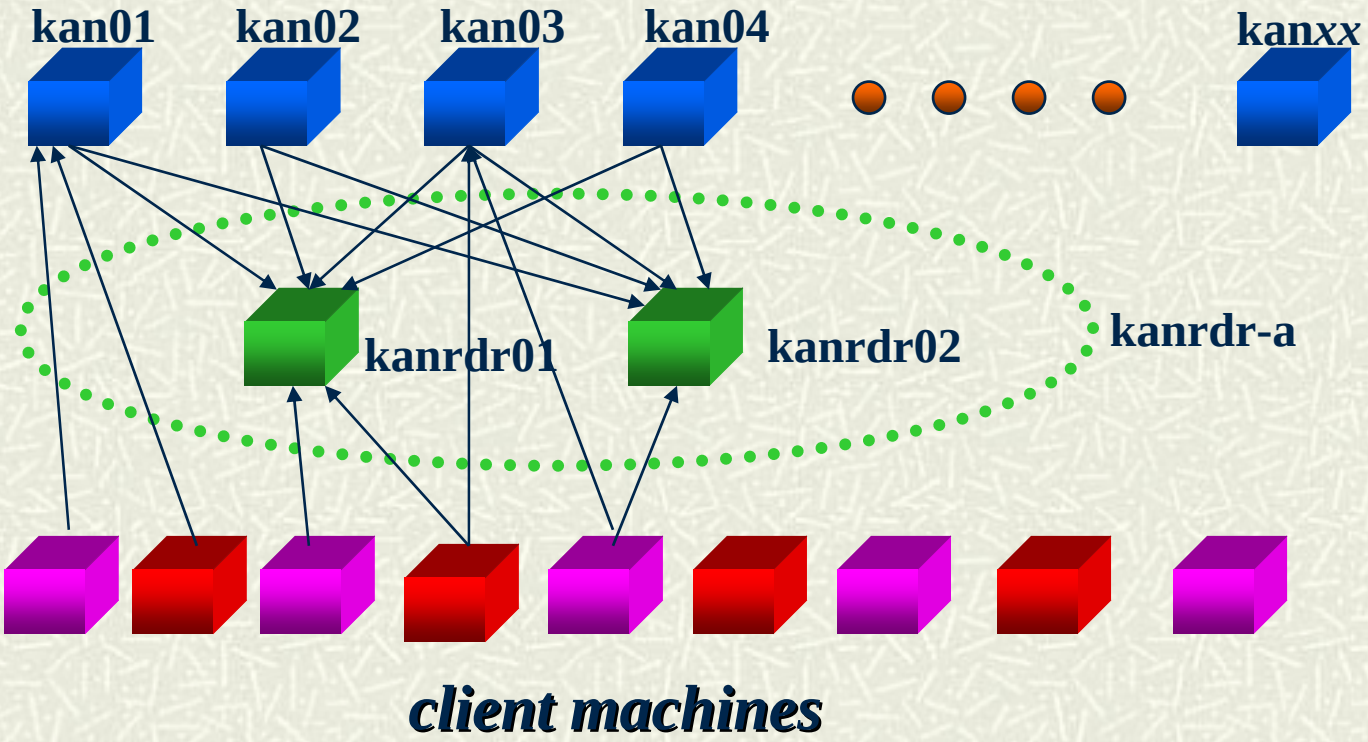
olb.role supervisor
olb.subscribe *host port*
olb.allow *hostpat*

Data Servers
(end-node)

ofs.redirect target

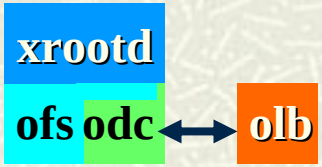
olb.role server
olb.subscribe *host port*

Example: SLAC Configuration



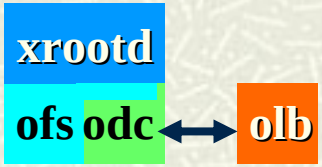
Hidden Details

Configuration File




```
if kanrdr-a+
    olb.role manager
    olb.port 3121
    olb.allow host kan*.slac.stanford.edu
    ofs.redirect remote
    odc.manager kanrdr-a+ 3121
else
    olb.role server
    olb.subscribe kanrdr-a+ 3121
    ofs.redirect target
fi
```

Potential Simplification?




```
if kanrdr-a+
    olb.role manager
    olb.port 3121
    olb.allow host kan*.slac.stanford.edu
    ofs.redirect remote
    odc.manager kanrdr-a+ 3121
else
    olb.role server
    olb.subscribe kanrdr-a+ 3121
    ofs.redirect target
fi
```

```
olb.port 3121
all.role manager       if  kanrdr-a+
all.role server        if !kanrdr-a+
all.subscribe kanrdr-a+
olb.allow host kan*.slac.stanford.edu
```

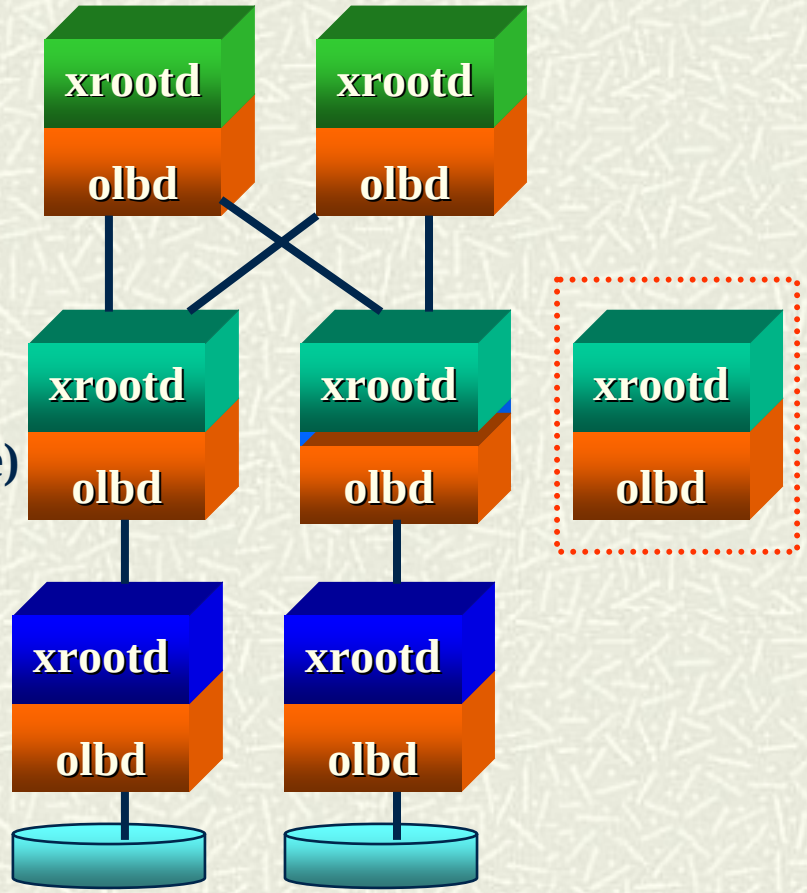
Is the simplification really better?
We're not sure, what do you think?

Adding Fault Tolerance

Manager
(Head Node)

Supervisor
(Intermediate Node)

Data Server
(Leaf Node)



Fully Replicate

Hot Spares

Data Replication
Restaging
*Proxy Search**

^xrootd has builtin proxy support today; discriminating proxies will be available in a near future release.

Stanford Linear Accelerator Center

Conclusion

- High performance data access systems achievable
  - The devil is in the details
- High performance and clustering are synergetic
  - Allows unique performance, usability, scalability, and recoverability characteristics
- Such systems produce novel software architectures
  - Challenges
    - Creating applications that capitilize on such systems
  - Opportunities
    - Fast low cost access to huge amounts of data to speed discovery

Acknowledgements

- Fabrizio Furano, INFN Padova
  - Client-side design & development
- Principal Collaborators
  - Alvise Dorigo (INFN), Peter Elmer (BaBar), Derek Feichtinger (CERN), Geri Ganis (CERN), Guenter Kickinger (CERN), Andreas Peters (CERN), Fons Rademakers (CERN), Gregory Sharp (Cornell), Bill Weeks (SLAC)
- Deployment Teams
  - FZK, DE; IN2P3, FR; INFN Padova, IT; CNAF Bologna, IT; RAL, UK; STAR/BNL, US; CLEO/Cornell, US; SLAC, US
- US Department of Energy
  - Contract DE-AC02-76SF00515 with Stanford University