Karlsruher Institut für Technologie

xrootd in HEP today

STEINBUCH CENTRE FOR COMPUTING - SCC

Outline

- xrootd
  - What?
  - Why?
  - How?
- xrootd in HEP experiments
  - ALICE, ATLAS, CMS

What xrootd is ...

- a file access and data transfer protocol
  - defines POSIX-style byte-level random access for
    - arbitrary data organized as files of any type identified by a hierarchical directory like name
- a reference software implementation
  - Embodied as the xrootd and cmsd daemon
    - xrootd daemon provides access to data
    - cmsd clusters xrootd daemons together
- a.k.a Scalla
  - ...but nobody uses that name

Slide Courtesy of Andy Hanushevsky

... and what xrootd is not!

- it's not a POSIX file system
  - FUSE based implementation called xrootdFS
    - xrootd based client simulating a mountable file system
    - no full POSIX file system semantics
- it's not an Storage Resource Manager (SRM)
  - SRM functionality can be provided via BeStMan
- it's not aware of file internals
  - can serve any file not only root files
  - it's distributed with root and proof because of it's unique features and efficiency

Slide Courtesy of Andy Hanushevsky

What makes xrootd so special?

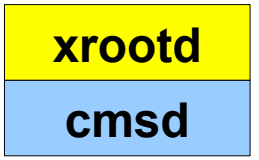
- resilience to errors
  - built into the protocol
  - easy to avoid single point of failures
  - pull the network cable on any component: client will recover
- clustering
  - cluster thousands of servers w/o any performance hit
  - cluster any storage architecture
  - federated clusters: build clusters of clusters via WAN
- plugin architecture
  - security, mass storage backends .... whatever you need
- low support requirements
  - hardware and sysadmin friendly

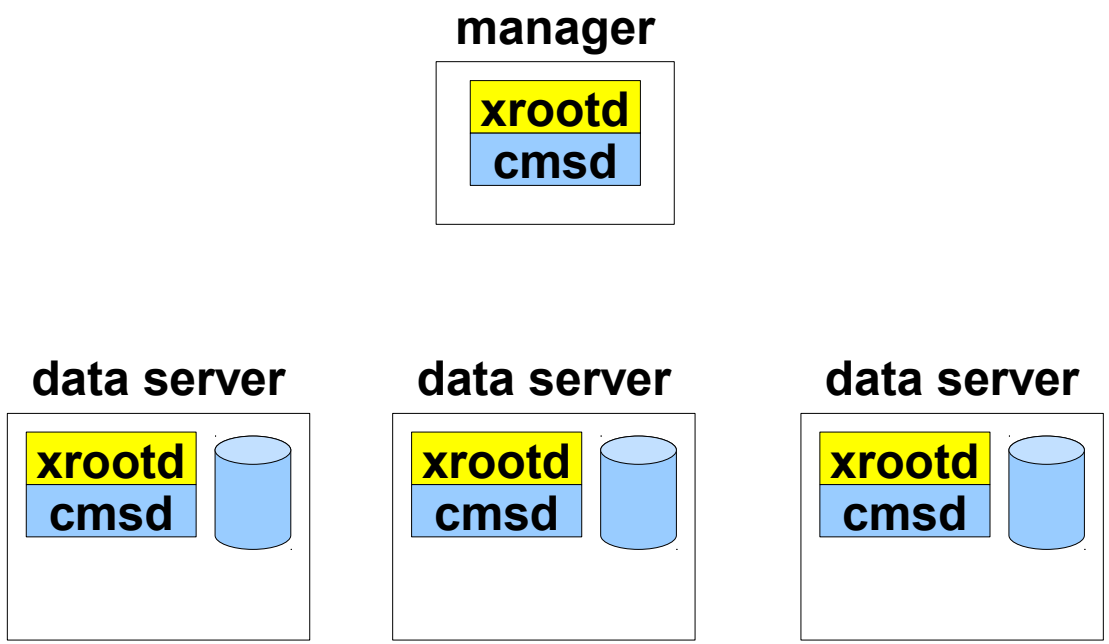
But how can you do it?

- avoid components that don't scale well
  - xrootd doesn't have a static file catalog
    - xrootd asks the cluster where the file is and caches the location for a while
- don't do what other specialized systems do better anyway
  - xrootd doesn't implement its own storage solution
    - xrootd serves files from any storage system
- keep data and control traffic and functions separate
  - xrootd daemon serves data
  - cmsd daemon finds files, does clustering

Simple xrootd Cluster

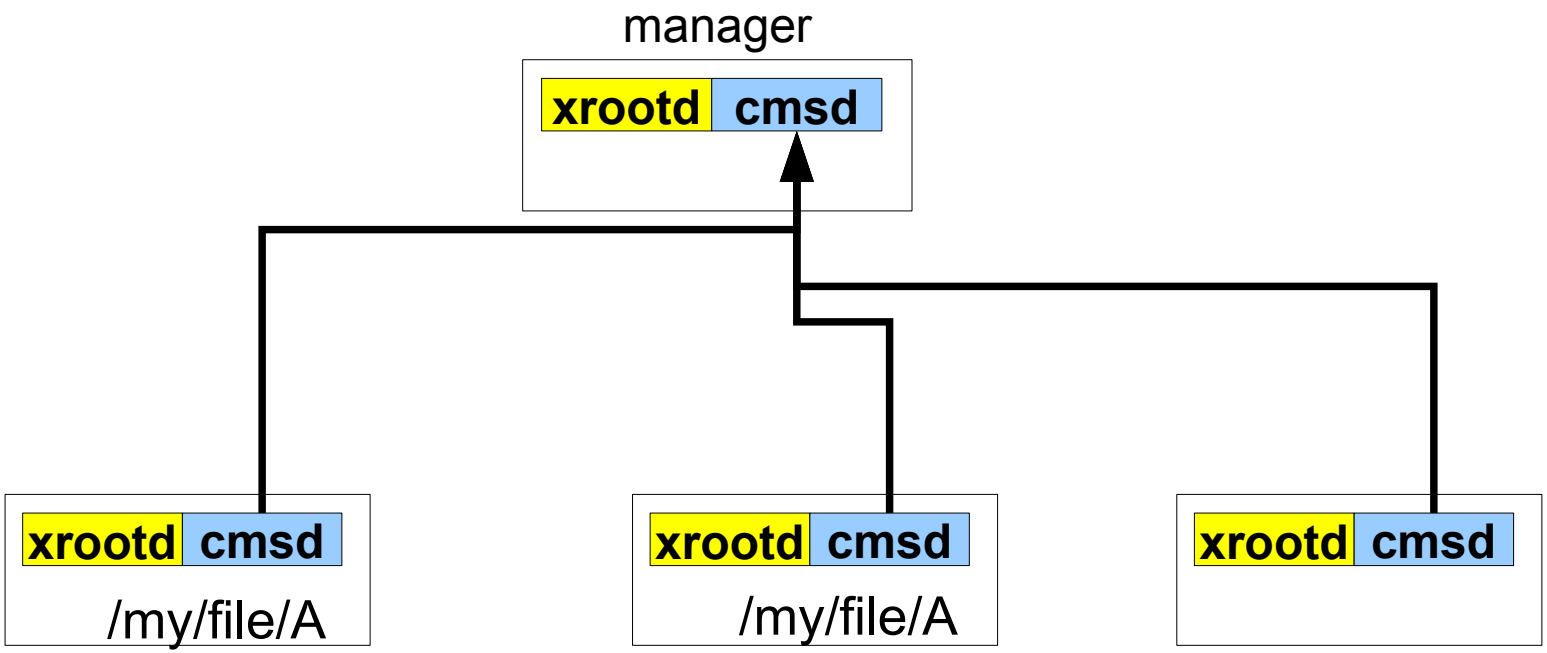
- single building block: cmsd+xrootd
- two incarnations
  - data server: where the client gets the data
  - manager: guides the client to the data server or to another manager





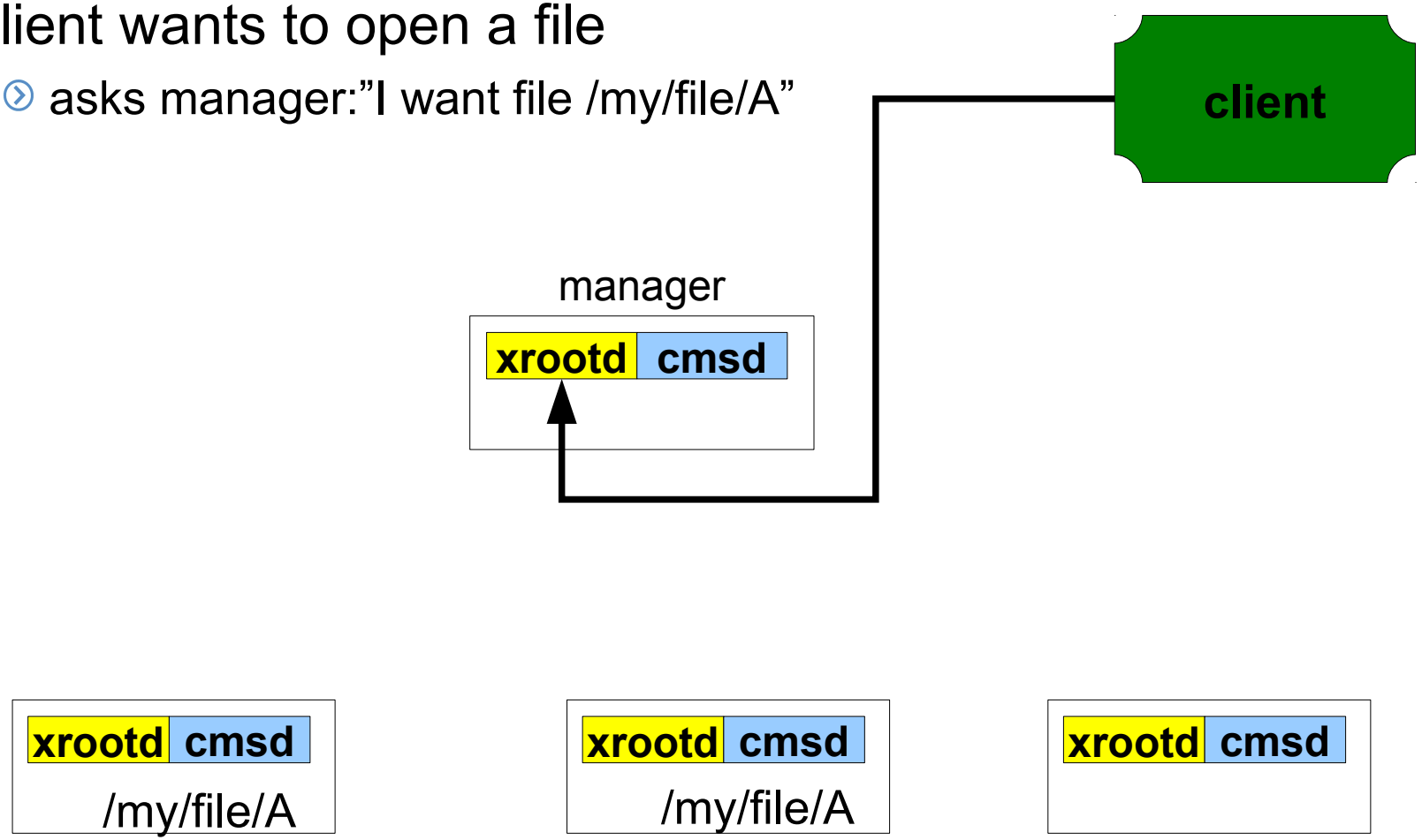
Simple xrootd Cluster

- cmsd form cluster
  - data servers register with manager



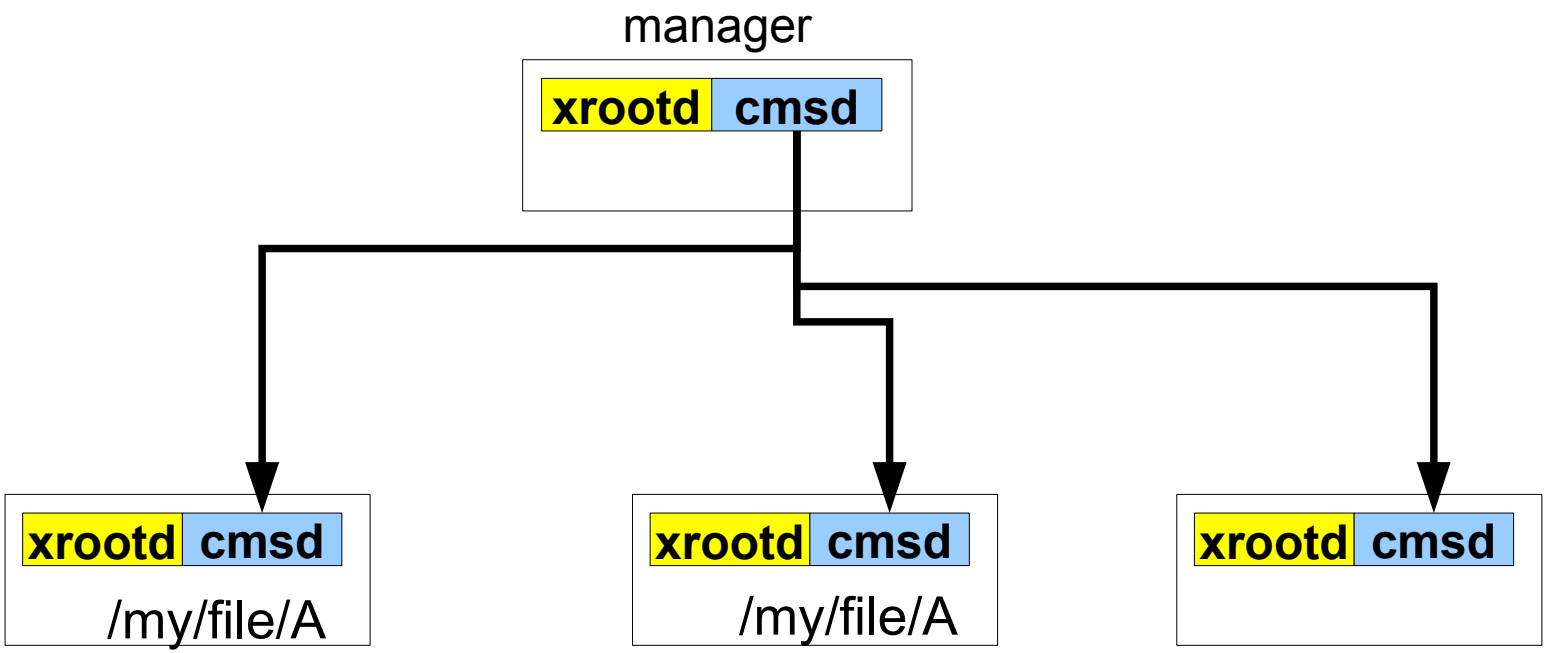
Simple xrootd Cluster

- client wants to open a file
  - asks manager:”I want file /my/file/A”



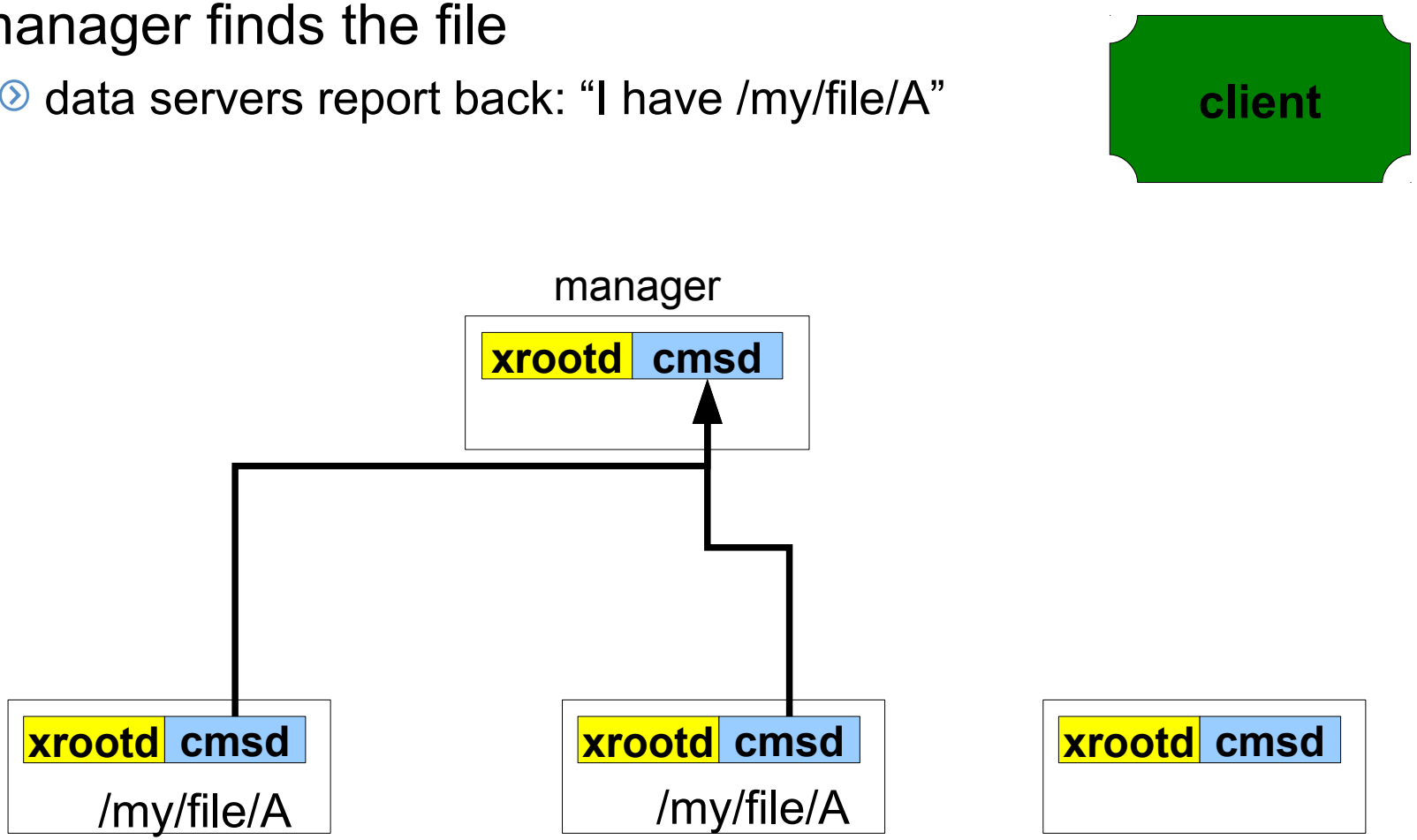
Simple xrootd Cluster

- manager finds the file
  - asks data servers: “who has file /my/file/A”



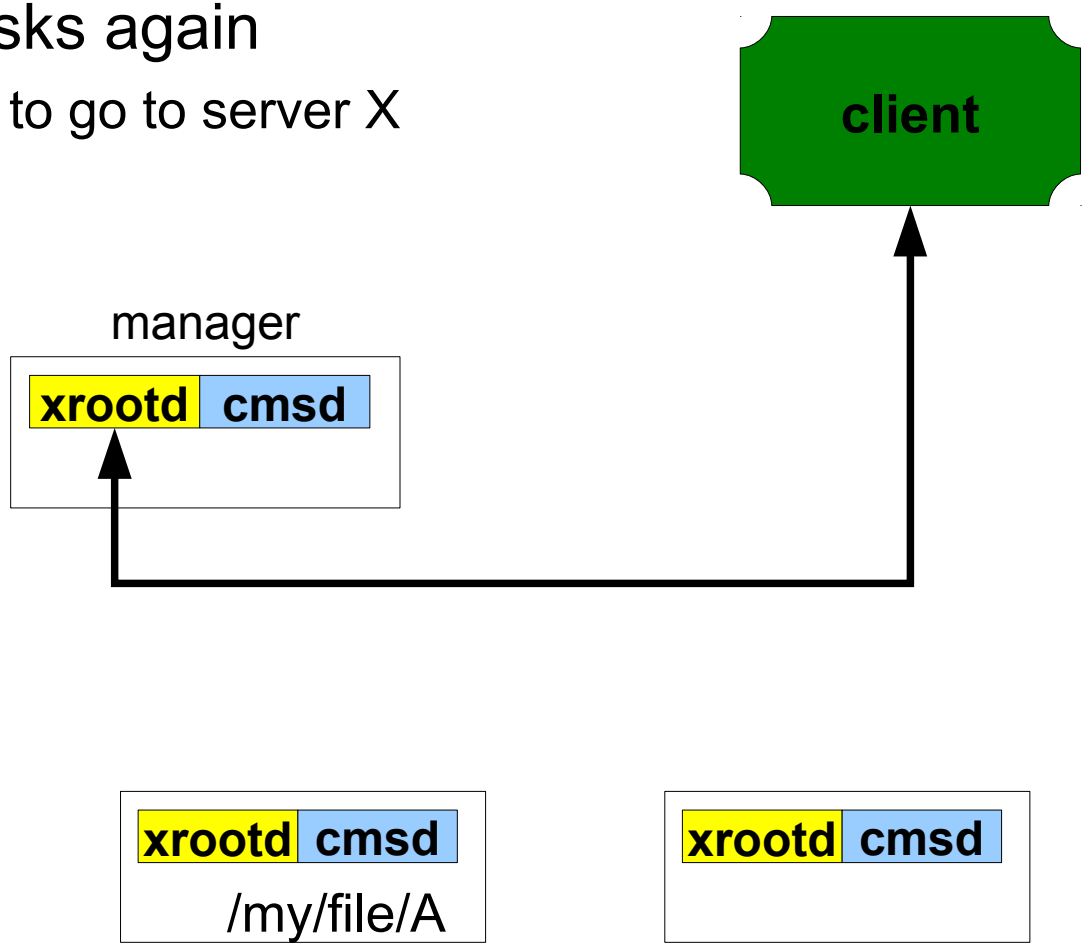
Simple xrootd Cluster

- manager finds the file
  - data servers report back: “I have /my/file/A”



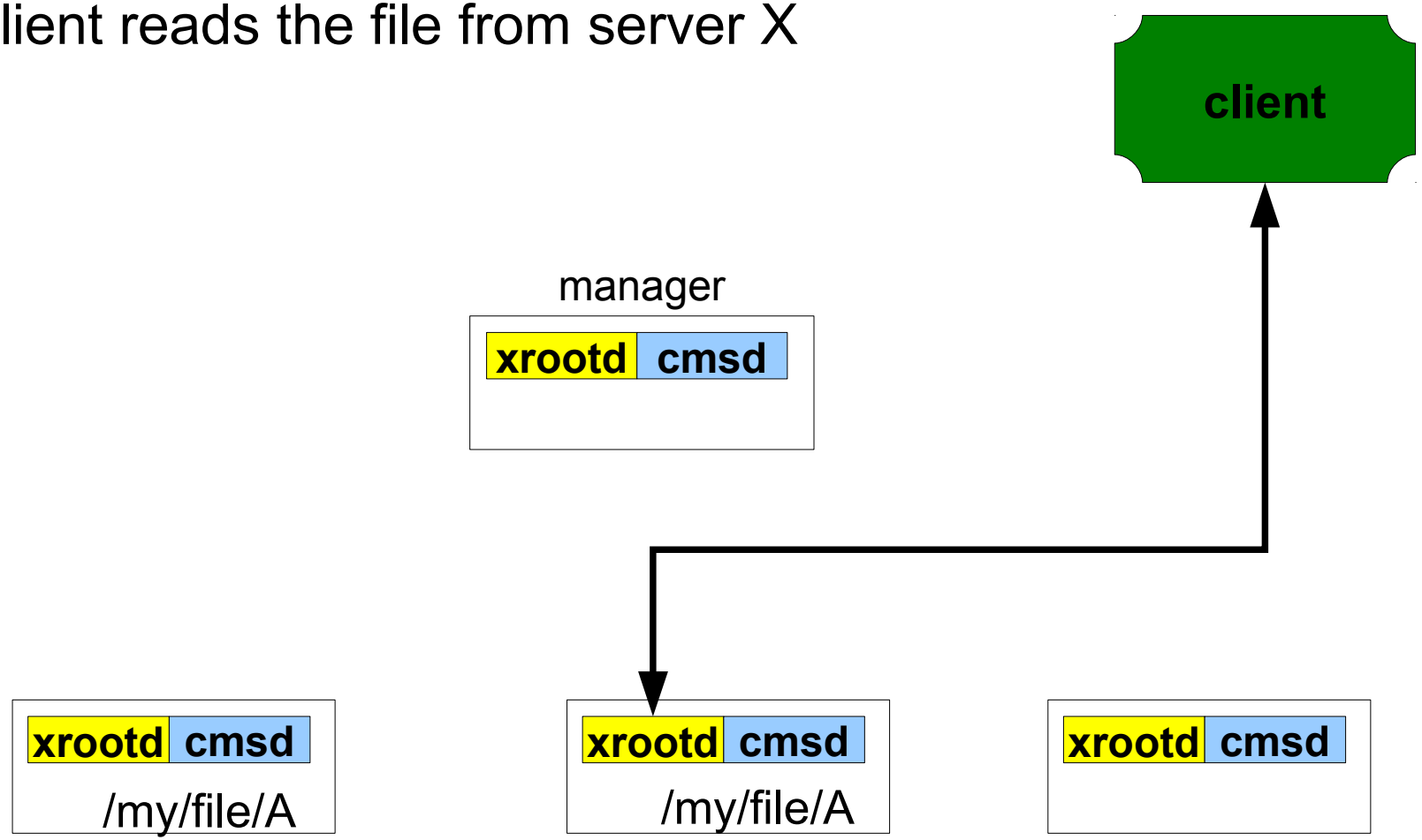
Simple xrootd Cluster

- client returns and asks again
  - manager tells client to go to server X



Simple xrootd Cluster

- client reads the file from server X

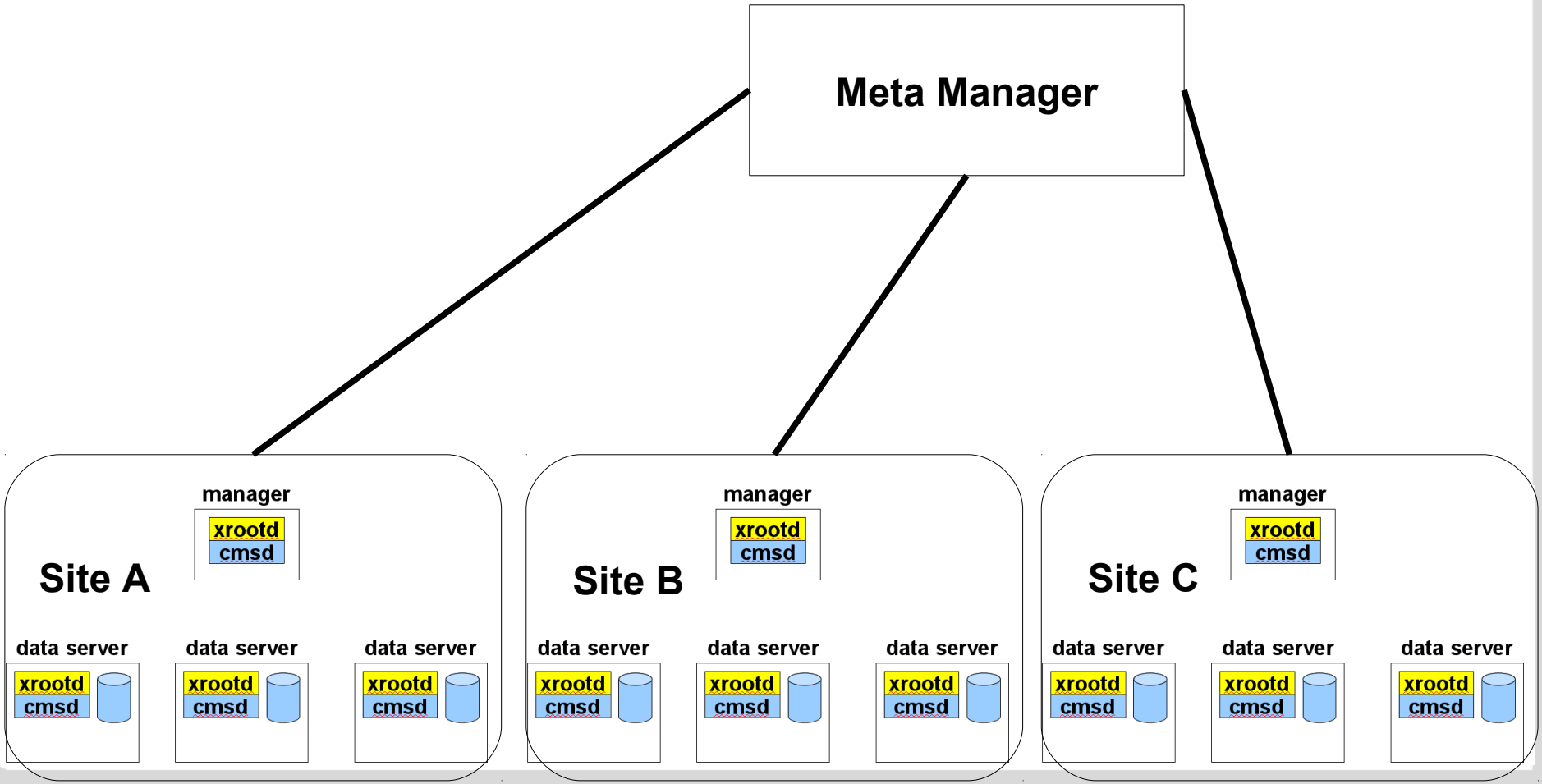


Advanced xrootd Clusters

- Massive clusters
  - another management layer required: supervisor nodes
  - manager can handle up 64 data server
  - supervisor: manager of 64 managers or data servers
  - Massive clusters possible
    - scales like 64^n where n is height tree ($64^3 = 262144$)

- File Residency Manager (FRM)
  - simple glue scripts allow easy integration of mass storage system (HPSS at SLAC, TSS/TSM at KIT)
    - file replication for load balancing
  - automatic purging, staging, on demand pre-staging

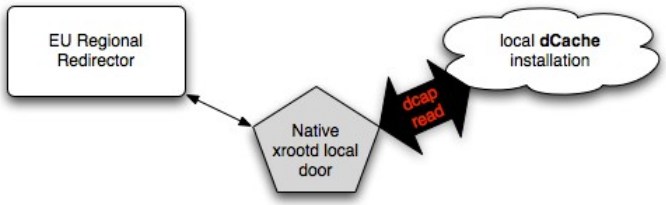
xrootd Cluster Federation



xrootd in current HEP Experiments

- BABAR xrootd pioneer, only local usage
- ALICE is using xrootd SEs
- ATLAS, CMS, LHCb are using dCache, StoRM, DPM, EOS, Castor, ...
- xrootd features very interesting
  - regional/global SE federations
  - virtual mass storage
  - provides global namespace via plugins
- requires xrootd protocol support in different components
  - many different options to tie in xrootd/cmsd block with dCache

CMS

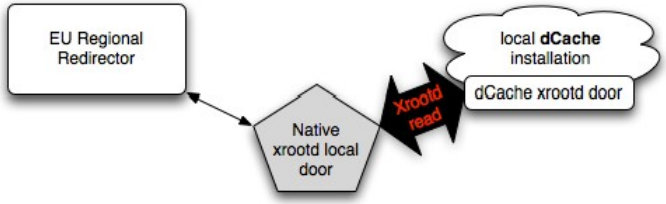


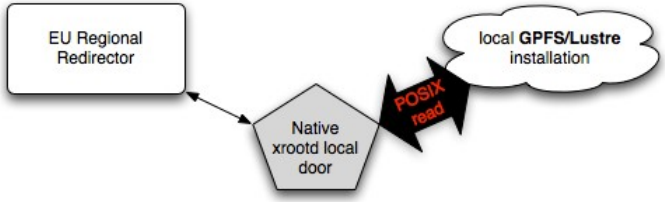
"xrootd over **dcap**"

- need a native xrootd door reading files using libdcap
  - need to be integrated in dCache in terms of libs and pnfs access
    - quite similar to the installation of a gridftp door

"xrootd over **xrootd**"

- need a dCache xrootd door reading files using xroot library
  - machine configured in "proxy-mode"
- the dCache instance must already provide a dCache xrootd endpoint





"xrootd over **POSIX**"

- need one or more native xrootd doors that are clients of the local parallel fs (GPFS/Lustre)

Use Cases

- Resilient analysis
  - fetch the last missing run
    - copy only when necessary
- Adaptable analysis
  - cache files where they are needed
    - copy whatever analysis demands
- Storage starved analysis
  - directly stream data from multiple sites to jobs
    - deliver to wherever the compute cycles are

xrootd in HEP Experiments - ALICE

- xrootd used as standard data access protocol
- disk-only and tape SEs
- meta manager ties all site clusters together
  - allows virtual mass storage system
    - if file is missing locate somewhere else and copy it here

xrootd in HEP Experiments

ALICE & ATLAS

- real time placement of files at site
- locally handle missing files, new/modified files, disk full
- worldwide for ALICE
- US centric but expanding for ATLAS

CMS

- tie together US regional T2/T3 sites
- run analysis at storage starved sites
  - higher latency but at least your job runs