Xrootd Tutorial
T2 workshop

CERN, June 15

Artem Trunov

CC-IN2P3

artem.trunov@cern.ch

What is xrootd

- Xrootd is extended ROOT daemon
  - Now a part of ROOT release, since v4.02
  - Replacement for rootd, backward compatible.
- Developers call it “data access system”
  - Not to confuse with data management or disk management system.
- Originally developed for BaBar at SLAC by Andy Hanushevsky when BaBar switched to ROOT based event store in 2003.
- Some features come from BaBar use cases.
- Load balancing part of xrootd is older than xrootd itself, and was developed for BaBar Objectivity/DB events store.
- Xrootd protocol is open, and there is even one more implementation of it, made by dCache team (xrootd door)

What is xrootd - II

- Featuring
  - Light-weight
  - Supports many clients
  - Uses one network connection per client for many open files
  - Allows to reuse files descriptors between clients
  - Load balancing clients between servers
  - Files replication for load balancing (from MSS backend)
  - Transparent staging from a MSS
    - If a file is not on disk, but in MSS, a client waits for the file to be staged in
  - Hierarchically scalable to hundreds of servers
  - Rather simple configuration – one config file for all servers!
- Non-featuring
  - SRM
    - Experimenting with FANL SRM
  - File database
    - Xrootd doesn’t keep a record of where the files are, a file is discovered when it’s requested
  - Great deal of data management function
    - File replication between pools
    - Space reservation, pinning etc
    - Useful admin interface.
    - Multistream file transfer (work in progress)

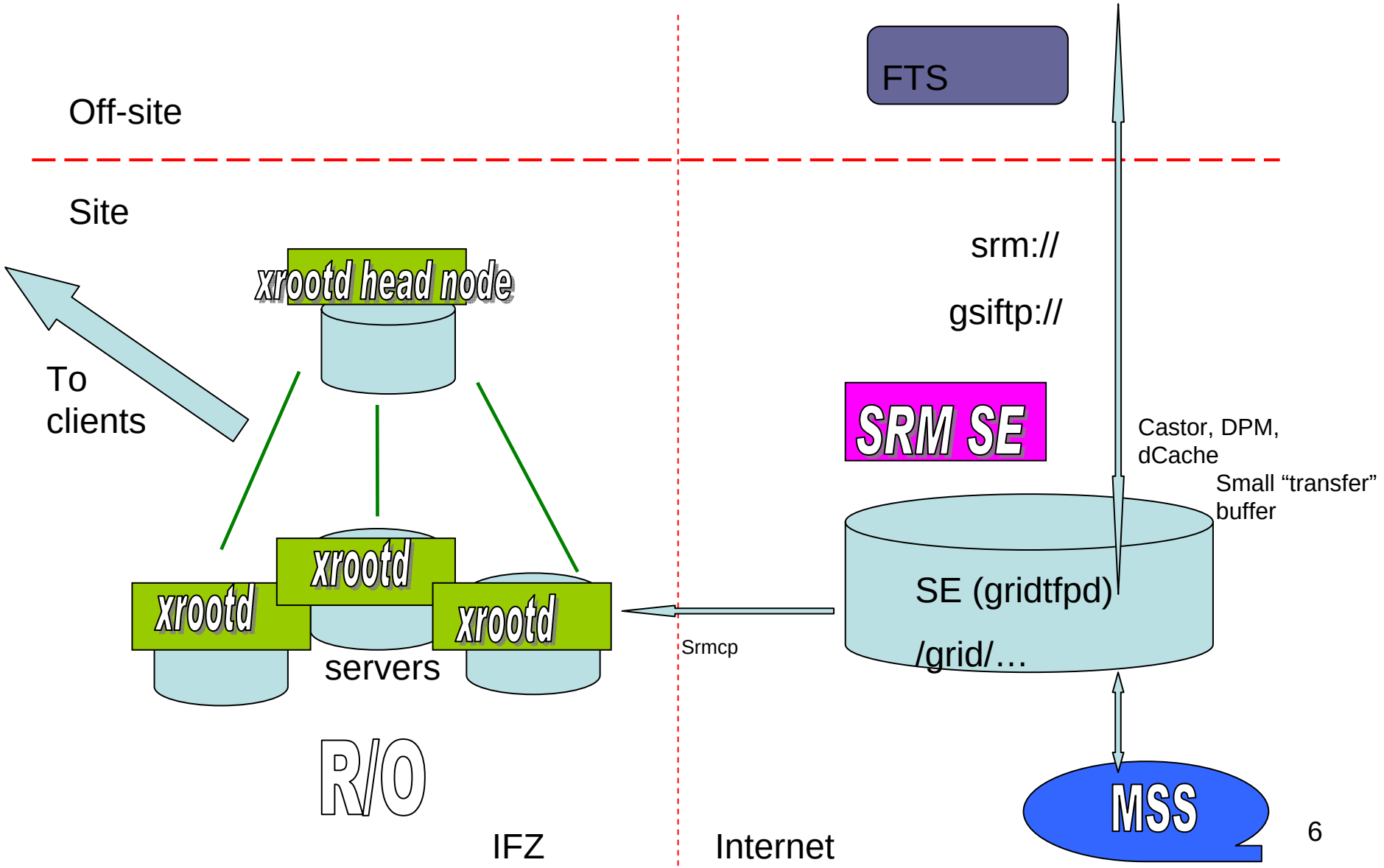
Why use xrootd

- Xrootd is getting more popular.
  - Alice effectively supports xrootd only
  - All other LHC experiments expressed interest
  - CMS supports xrootd in the new data model
- Even Castor and DMP will soon run xrootd on their nodes, so you may need to know how to configure it.
- *It's not a straight replacement for Castor, dCache, DPM!*
  - Due to lack of an SRM interface mainly – a requirement of all the experiments.

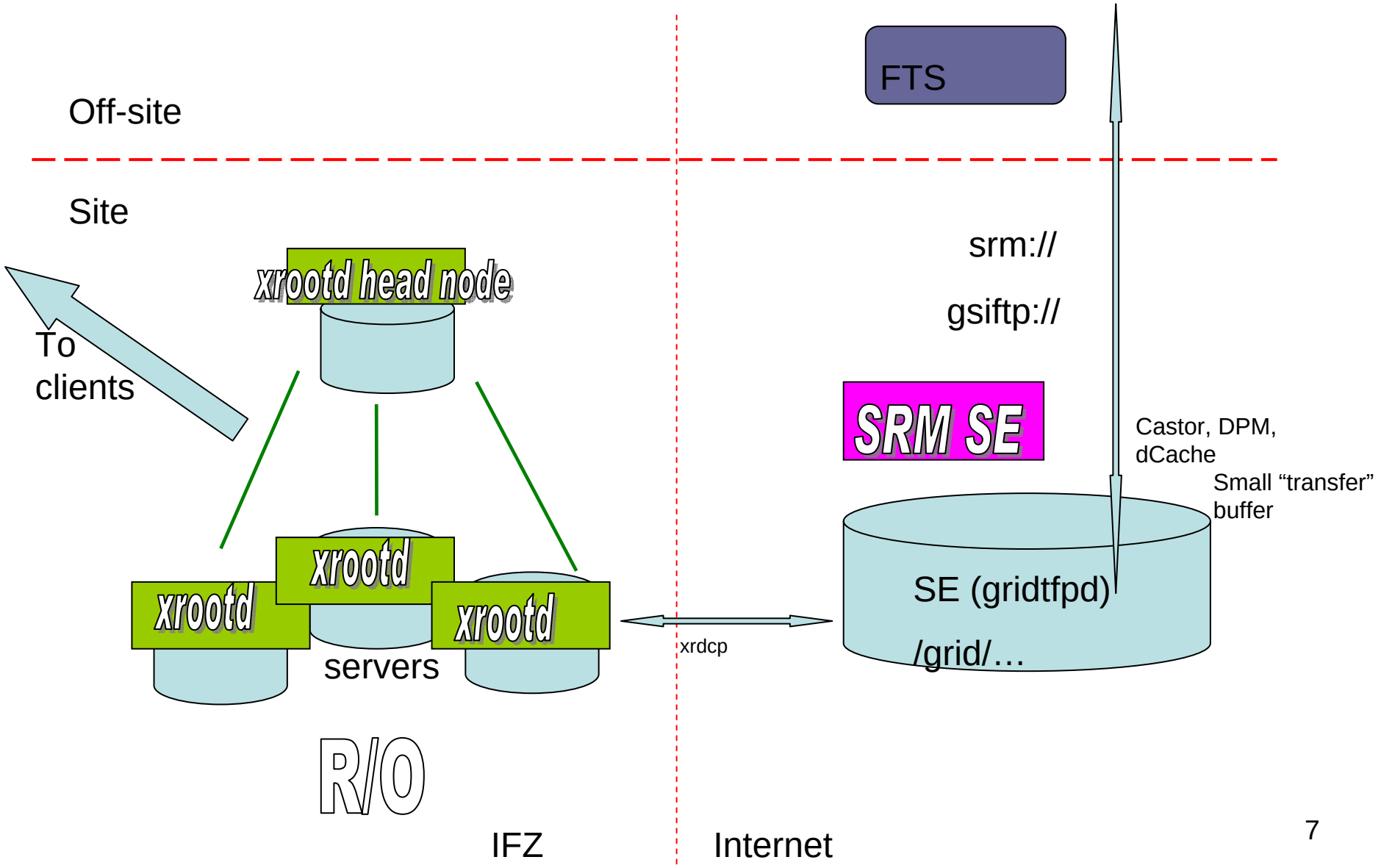
Where to use xrootd

- Xrootd is most suitable for read-only data access by jobs from a large disk cache in front of SRM-enabled storage
  - Bypasses heavy storage machinery for file access
  - No authentification, authorization overhead
  - Scales to hundreds of servers
  - Less maintenance too

Xrootd with backend SRM and MSS



Xrootd as a backend to SRM without MSS



Notes on these setups

- SRM is a small buffer.
  - It's capacity is determined by the to/from site transfer throughput.
  - 2 servers with 3 TB each is sufficient to manage 1GBs link.
  - System managed with garbage collector
  - Files are never access from jobs to this buffer
- Xrootd is a big read-only cache – this is your main disk storage
  - Job's output goes to SRM directly (presumably for future transfers)
- Security
  - You SRM storage is on public network, but very few IPs needed
  - Your xrootd pool is on private network or behind the firewall
  - No NAT, proxy or other complications
- You need to do (*you always need to do some customization work, right?*):
  - Have a way to let you application know how to access the data in xrootd
  - Adjust xrootd installation to common practices at your site, i.e. how you disctribute xrootd bins/libs to servers, how you manage updates, start and stop (default scripts are no good),
  - *Not everything here is straight-forward or implemented, but if you chose to go the xrootd way, I will be happy to assist.*

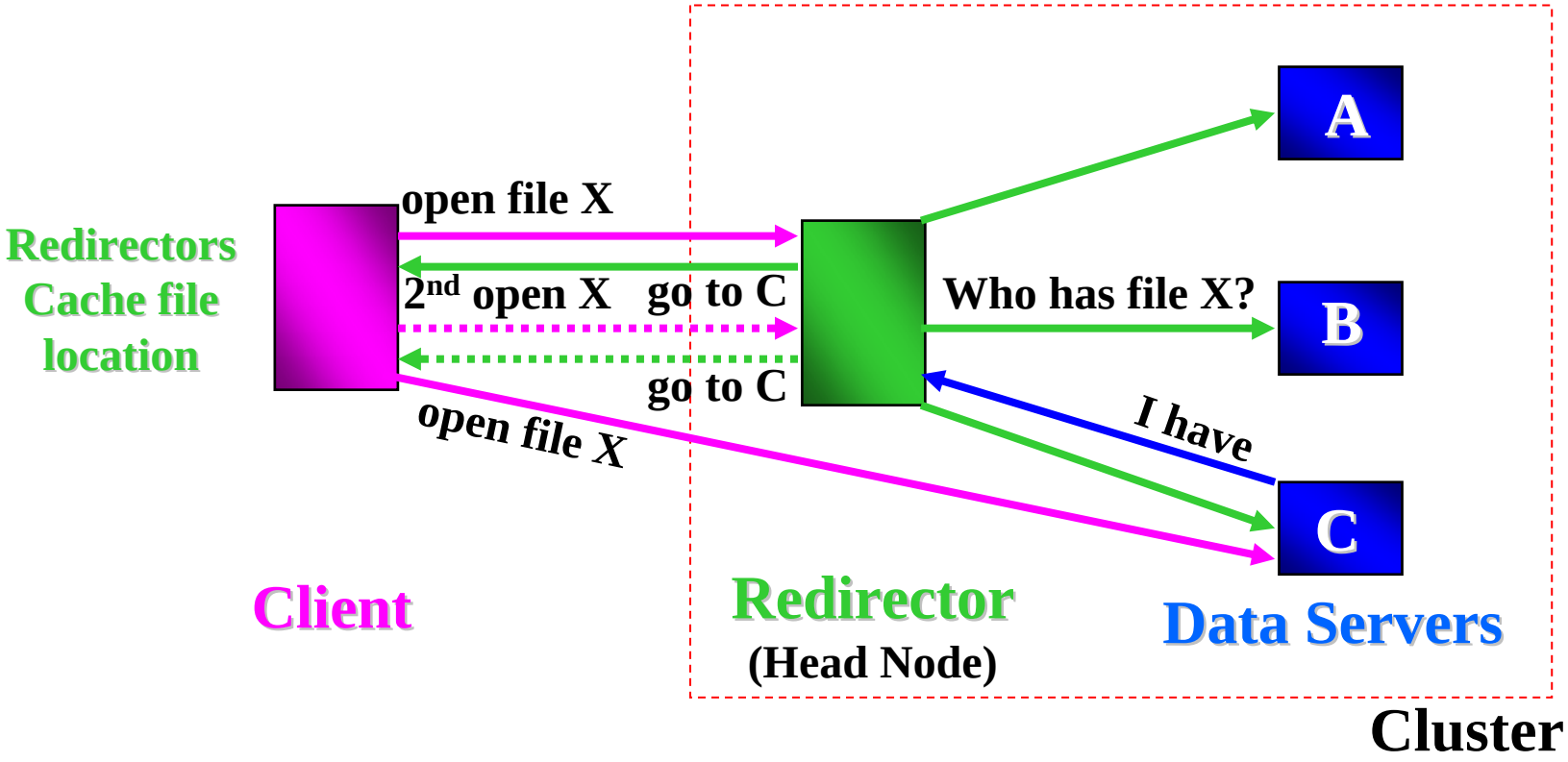
Other ideas on xrootd setups

- “Resilent” xrootd
  - i.e. xrootd managing disk space on worker nodes
  - Popular setup, especially in the US T2 sites (with dCache)
  - With xrootd already used by LBL
  - Planned by some British sites.
  - Ease of xrootd management is a big factor
  - Though, no automatic replica management etc, unlike dCache.
- Xrootd and PROOF
  - At least Alice and CMS are interested in using PROOF for analysis
  - Since xrootd servers are not CPU-loaded, you can run PROOF jobs on their spare CPU cycles!
  - Buy powerful servers (dual-cpu dual-core Opterons) + 6-20 TB
  - Try it out – you are the first! ☺

In This Tutorial

- How it works
- Terminology
- Getting, installing, testing…
- Various configurations (we will work our way through various configurations, expanding list of employed options)
  - Single server
  - Multiple servers (disk)
  - *Configuration with mass storage not covered*
  - *Authentification methods not covered*
- Non-mainstream options
  - Proxy mode
  - Redundant redirectors
  - Running more than one xrootd servers on a host
- Debugging, monitoring
- Extras
  - Xrootd and FNAL SRM
  - Xrootd for Alice
  - Xrootd for CMS: setting up CMSSW+trivial file catalog to use xrootd
  - Patching your older ROOT installations
- Getting more info

How it works



Courtesy of A. Hanushevsky

Terminology

- xrootd server
  - referes to both xrootd daemon and the machine running it. Be careful, look for the context. We will try to use "xrootd" for the daemon and "xrootd node“ and “xrootd server” to refer to the machine.
- redirector
  - xrootd running in a special mode (and again the machine it’s running on). This is a "head node", where all clients make an initial connect. It does not server the data itself. Will try to use “redirector node” to refer to the machine.
- olbd
  - load balancing daemon. It works in pair with xrootd to provide file location and load balancing.
- xrdcp
  - a program used to copy files in/out of xrootd servers using xrootd (root://) protocol.

Getting, installing, testing

- Visit http://xrootd.slac.stanford.edu
  - Download prebuilt binaries for your Server platform
    - Linux and Solaris, including v.10 on Opterons are best supported)
  - Or download source code and compile
  - Versioning: 20050920-0008 – just a build date/time
- Relocate to where you want to use it, i.e. perhaps /opt/xrootd (let's assume you do so)
- Add /opt/xrootd/lib to the $LD_LIBRARY_PATH and /opt/xrootd/bin to the $PATH (1,2)
- Start xrootd: (3)
- Run xrdcp (xrootd copy program) to and from the server you've set up (4,5)

```
(1)$ setenv PATH /opt/xrootd/bin:$PATH
(2)$ setenv LD_LIBRARY_PATH /opt/xrootd/lib:LD_LIBRARY_PATH
(3)$ xrootd -l /tmp/xrootd.log &
(4)$ xrdcp root://server.domain.edu:1094//tmp/xrootd.log /tmp/test1
(5)$ xrdcp /tmp/test1 root://server.domain.edu:1094//tmp/test2
```

- Notes:
  - By default (no config file) xrootd can read/write only to /tmp. You can see the configuration warning in the xrootd.log file.
  - Default port is 1094
  - Two slashes // after the port is mandatory: one will not work (don't ask ☺)

Single server configuration - minimal

- It requires a configuration file (assume it's xrootd.cf)
- Configuration directives are specified for each subsystem separately
- Will try to naturally advance from simplest to more complex
- xrootd directives are for xrootd (protocol) layer itself
- xrd directives are for the abstract xrd layer, that is actually handling connections, threads etc

```
xrootd.export /data

## This is to enable xrootd to serve certain path(s).

## Multiple such directives can be specified.


xrd.port 51234

## default is 1094
```

Single server configuration - extending

- OFS plug-in – whole set of options for file system management

```
xrootd.export /data

xrootd.fslib /opt/xrootd/lib/libXrdOfs.so

## This enable whole set of directive related to disk and
directory management (ofs and oss)
```

Overview of subsystems (from the manual)

- **Xrd** E**x**tended **R**equest **D**aemon
- **Xrootd** The **xrootd** protocol implementation
- **Olb** **O**pen **L**oad **B**alancing
- **Ofs** **O**pen **f**ile **s**ystem
  - coordinating **acc**, **odc**, & **oss** components
- **Acc** **Ac**cess control (i.e., authorization)
- **Odc** **O**pen **d**istribute **c**ache
- **Oss** **O**pen **s**torage **s**ystem
  - i.e., file system implementation
- **Sec** **Sec**urity authentication

Oss for the case of multiple file systems on the server

- Very common case, you have /data1 and /data2 on your server.
  - Select a “primary” partition, say /data1
  - Make “cache” directories in both partitions:
    - /data1/cache1, /data2/cache2
  - Your config files looks like:

```
xrootd.export /data1/cms

oss.cache public /data1/cache1

oss.cache public /data2/cache2
```

- Files will be created as soft links from the “namespace” (the xrootd.export) to the actual files in the cache{1,2} directories:

```
$ ls /data1/cms/testfile1.root
/data1/cms/testfile1.root -> /data1/cache1/%data1%cache%cms%testfile1
```

Oss layer – for more advanced file system management

- It’s most useful when you have a MSS backend
- However few options are still used without MSS (will discuss later)

```
oss.path /data/cms r/o

## for the case you don’t want to allow jobs (and xrdcp client) to write to
certain paths.

oss.forcero /data/cms

## forces all r/w requests to be r/o, error only given on actual write
attemtp

oss.inplace /data/cms/dir1

## files under this path will be created without soft links
```

Oss.localroot – nicer namespace management

- So far our server config file looks like:

```
xrootd.export /data1/cms
oss.cache public /data1/cache1
oss.cache public /data2/cache2
```

- But what if you have /data3 and /data4 on your second server? And you want to keep a single namespace for your files? Use oss.localroot directive

```
xrootd.export /cms
oss.localroot /data1
oss.cache public /data1/cache1
oss.cache public /data2/cache2
```

- Real PFN = oss.localroot + LFN
  - Where xrootd LFN is /cms/your/file:
  - $ `xrdcp root://server:1094//cms/your/file /tmp/test2`
  - Note, how we hided the mount point and exposed only abstract names space!

Oss.localroot – nicer namespace management (2)

```
xrootd.export /cms
if exceptional.server.domain.edu
oss.localroot /data3
oss.cache public /data3/cache1
oss.cache public /data4/cache2
else
oss.localroot /data1
oss.cache public /data1/cache1
oss.cache public /data2/cache2
fi
```

- Could have **if** directive for each server, that are different, and keep single config file for them all!
- *Note, the there is apparently a bug, and **writing** via redirector is not working when oss.localroot specified on servers. But you can still use it in r/o mode which is recommended to LCH experiments. And watch for updates.*

Notes on using conditional directives

```
if [ hostpat [. . .] ]
      [ other directives when if is true ]
[ else
      [ other directives when if is false ]
]
fi
```

Alternatively:

```
directive if hostpat
```

hostpat: *host* | *host+* | *pfx** | **sfx* | *pfx*sfx*

is the pattern of the host to which subsequent directive applies. All non-applicable hosts ignore all directives until the fi.

Host patterns are:

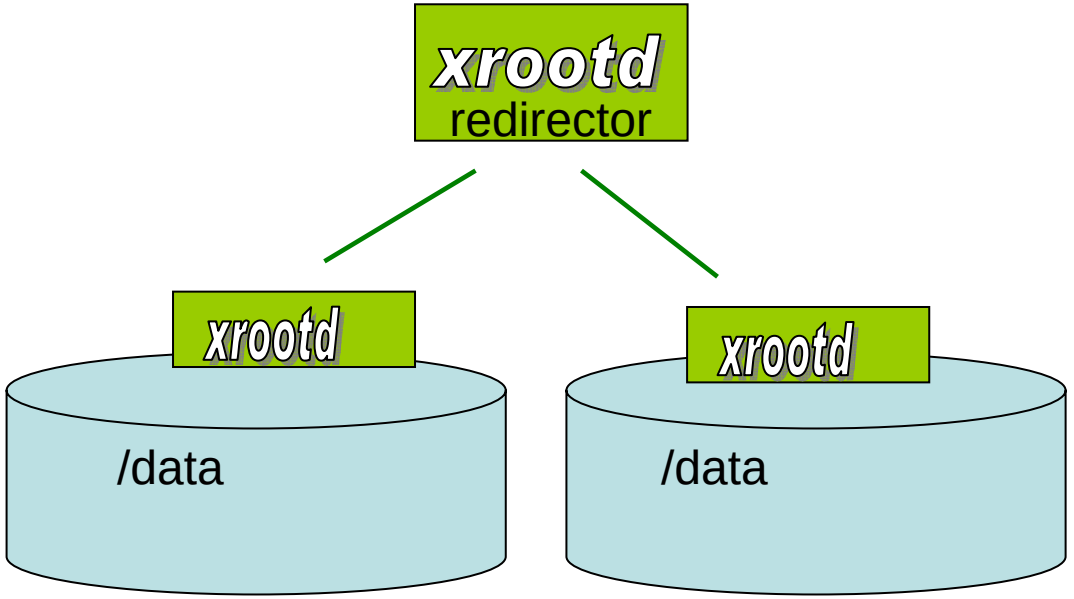
- Host
  - Any host that matches the specified DNS name.
- Host+
  - Any host that has an address that matches any of the addresses assigned to host.
- Pfx*, *sfx, pfx*sfx
  - * is for any characters

Nested if directives are not allowed

If you see conditional directives are not working for you – it's most likely related to FQHN resolution. I.e. check the output of "'hostname –f" and use the same for value in conditional directive

Multiple servers – setup with a redirector

- Client contacts the redirector (“head node”)
- Redirector locates a file, and passes the location to the client
- Client reconnects to a server directly
- Redirector doesn’t need a disk, neither fast CPU
- Two daemons need to be run on either redirector or servers
  - Xrootd
  - Olbd
- Start up order not important
- It is olbd that locates files and selects a host for redirection
- They all still use single config file



Relationship between xrootd and olbd

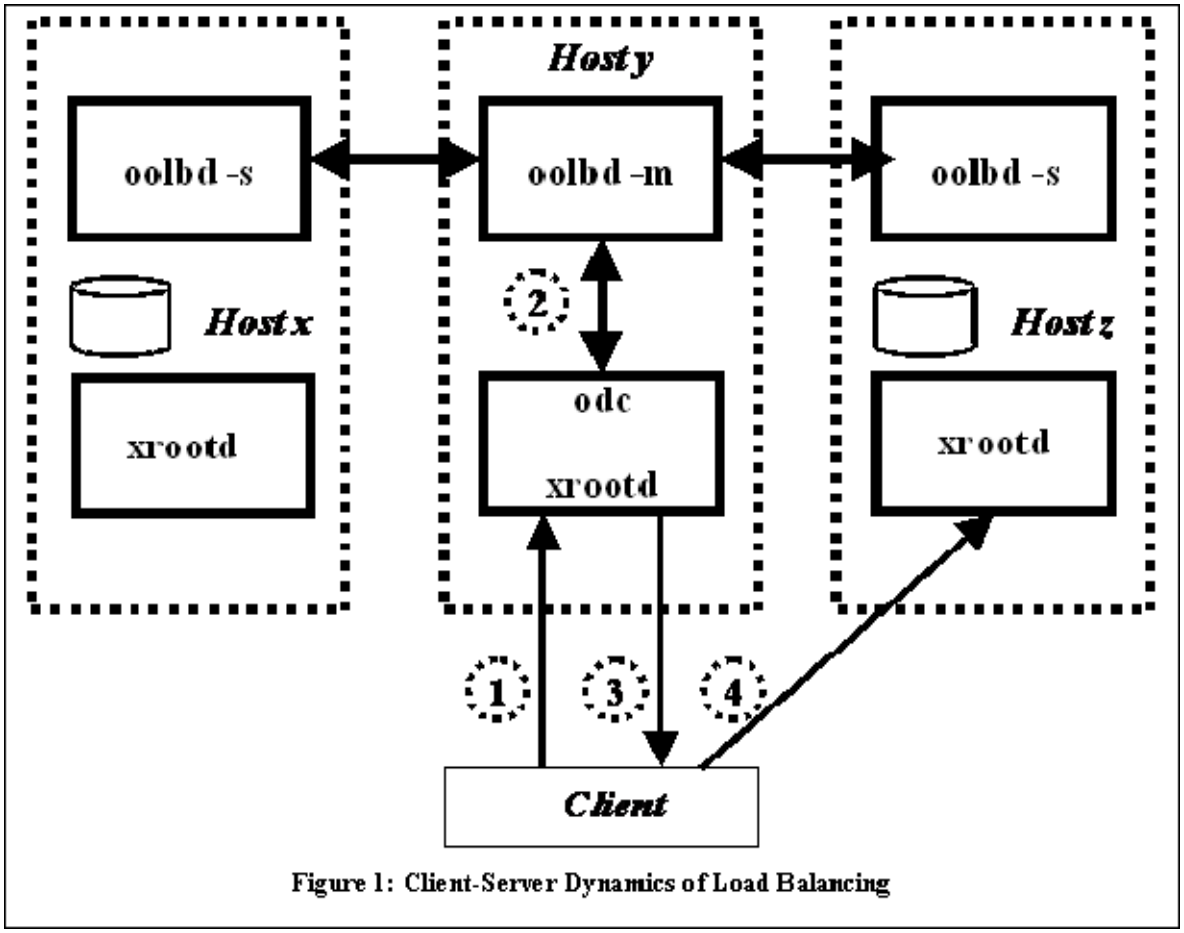
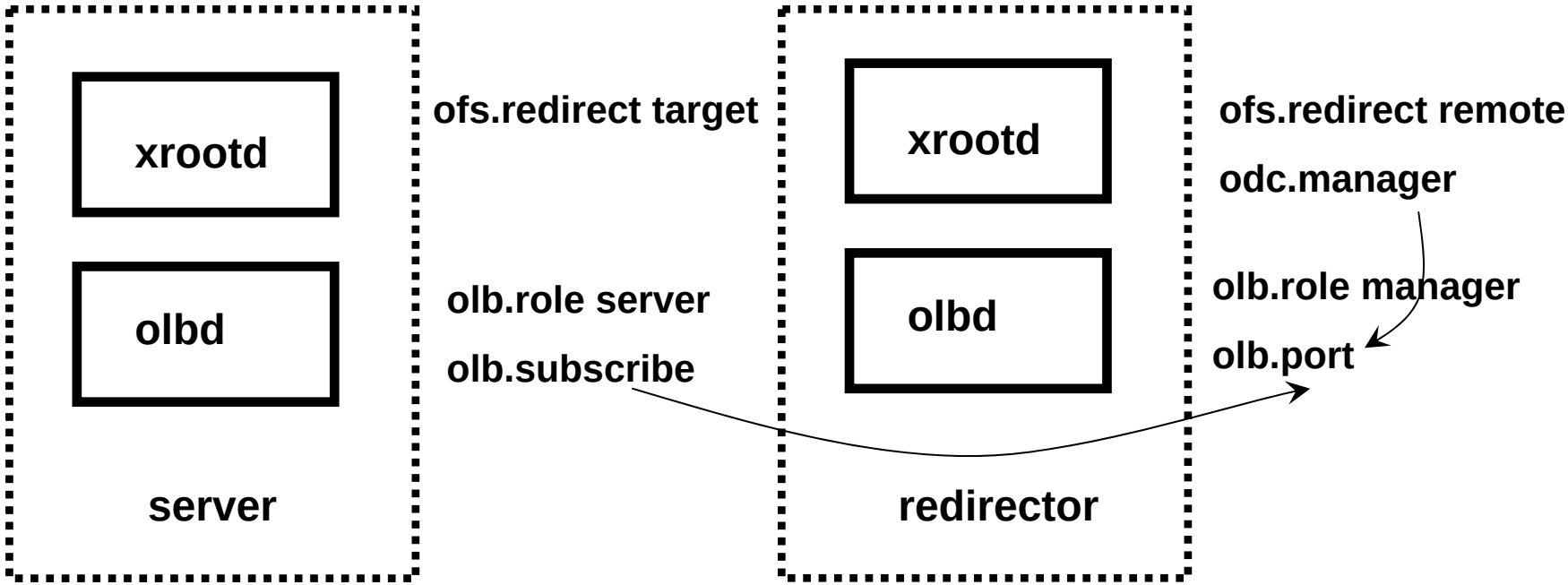


Figure 1: Client-Server Dynamics of Load Balancing

Picture: courtesy of A.Hanushevsky

Components and config directives for setup with redirector

- ofs, olb, odc components involved



Note: the manual has an old wrong picture

Multiple servers – config directives

- The option that “turns on” redirection is ofs.redirect
  - ofs.redirect remote on the redirector
  - ofs.redirect target on servers
- Olbd on the redirector accepts connections from server’s olbd
  - Olbd.role manager
  - Olbd.port 1095
- Odc of the redirector xrootd talks to redirector olbd
  - Odc.manager redirector.domain.edu 1095
- Server’s olbd subscribe to redirector’s olbd
  - Olbd.role server
  - Olbd.subscribe redirector.domain.edu 1095

Multiple servers – config file

```
if redirector.domain.edu
ofs.redirect remote
olb.role manager
olb.port 1095
odc.manager redirector.domain.edu 1095
else
ofs.redirect target
olb.role server
olb.subscribe redirector.domain.edu 1095
fi
```

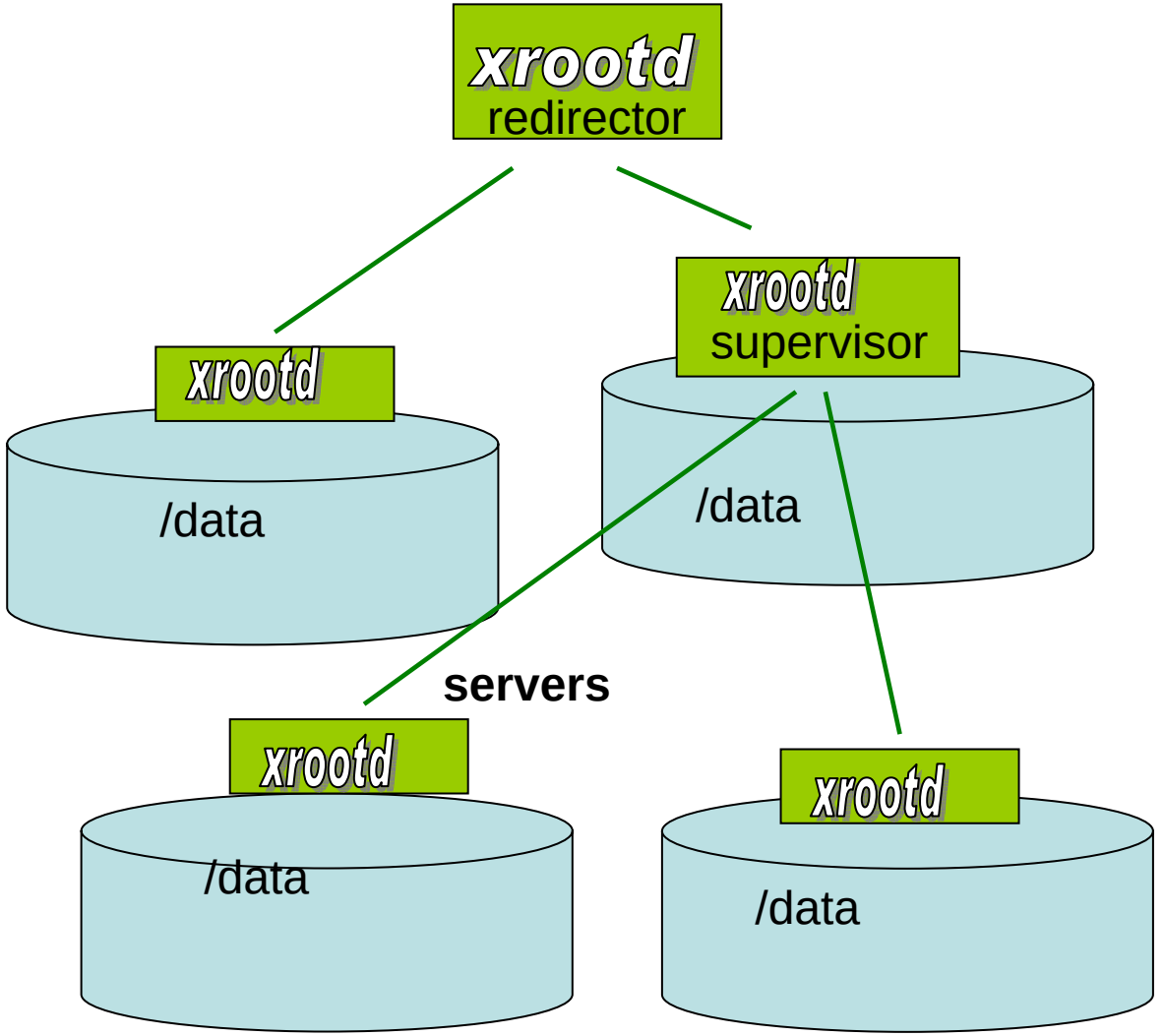
Multiple servers – more security

- Olb.allow
  - To restrict servers that can subscribe to the olbd redirector (“manager”)

```
if redirector.server.domain.edu
ofs.redirect remote
olb.role manager
olb.port 1094
odc.manager redirector.domain.edu 1095
olb.allow redirector.domain.edu
olb.allow server1.domain.edu
olb.allow server2*.domain.edu
fi
```

Multiple servers - scaling

- One redirector only can get 64 subscribers
- Hierarchical subscription in effect to scale beyond 64 server
- New role – supervisor. It can serve data as well
- No need to define hierarchy in advance – self-organizing system. You only need enough supervisors so each server will find its own.



Multiple servers – config with supervisor

```
if redirector.server.domain.edu
ofs.redirect remote
olb.role manager
olb.port 1094
odc.manager redirector.domain.edu 1095
else
ofs.redirect target
olb.role server
olb.subscribe redirector.domain.edu 1095
fi
if supervisor.domain.edu
olb.redirector remote
Olb.redirector target
Olb.role supervisor
olb.subscribe redirector.domain.edu 1095
fi
```

Multiple servers - Partitioning a cluster

- Suppose you want to dedicate different servers to different VOs.
  - Olb.path *on the server node* helps
  - You could tell olbd that a given host only serves certain paths
  - Same directive can separate read-only servers with read-write servers
- **Olb.path w /somepath *must be specified to be able to write to a server, since default is “r /”.***

```
xrootd.export /data
olb.path w /data/alice
olb.path w /data/cms if server1.domain.edu
## server1 can also serve CMS
```

Multiple servers - managing load balancing

- When there are few file replicas, the choice of the server can be tuned up.
- By default, round-robin scheduling is used, irrespective to the load on the servers
- In order to account for load, one has to run a load collector and define a load computing strategy.
- Refer to the following directives in the OLB manual:
  - Olb.perf
  - Olb.sched
  - Olb.ping

Non mainstream - proxy mode

- This is to access your data on xrootd servers behind firewall.
- Still sort of experimental mode
- No need to run olbd on the proxy node – it uses the one on the redirector

```
if proxy.domain.edu
ofs.redirect proxy
odc.manager proxy redirector1.domain.edu 1095
fi
```

Non-mainstream – redundant redirectors

- You may need this since redirector is a single point of entry, hence single point of failure.

```
if redirector1.domain.edu redirector2.domain.edu
ofs.redirect remote
olb.role manager
olb.port 1095
odc.manager redirector1.domain.edu 1095
odc.manager redirector2.domain.edu 1095
else
ofs.redirect target
olb.role server
olb.subscribe redirector1.domain.edu 1095
olb.subscribe redirector2.domain.edu 1095
fi
```

Non-Mainstream – Running more than one xrootd on a server node

- Can be useful for many reasons, i.e.:
  - You share disk pool between Alice and other experiments. Alice using one redirector, other using another redirector.
- Choose different ports for your instances of olbd, and xrootd.
- Use –n *name* option when you start olbd and xrootd

Debugging, Monitoring

- Trace directives
  - xrd.trace
  - xrootd.trace
  - xrootd.log
  - olb.trace
  - ofs.trace
- Monitoring with xrootd.monitor directive
  - http://xrootd.slac.stanford.edu/examples/monitoring/index.html

Extras – Alice-specific setup

- Alice developed a two step installation process, however it's only useful for Alice
  - Download using alien-installer
  - Configure using site info in Alice LDAP server
- See more on Alice setup at the Wiki page:
  - http://alien.cern.ch/twiki/bin/view/AliEn/HowToInstallXrootd
- Come to Alice Tutorial session on Fri, June 16:
  - http://indico.cern.ch/conferenceDisplay.py?confId=3718

Extras - CMS

- In xrootd.cf file:
  - Xrootd.export /store
  - Oss.localroot /yourmountpoint37
- Add in your trivial file catalog storage.xml:

```
<lfn-to-pfn protocol="root"
path-match="/+store/(.*)"
result="root://redirector.domain.edu:1094/store/$1"/>

<pfn-to-lfn protocol="root"
path-match="root://redirector.domain.edu:1094/store/(.*)"
result="/store/$1"/>
```

- In SITECONF/<YOURSITE>/JobConfig/site-local-config.xml :

```
<catalog
url="trivialcatalog_file:/x/y/z/storage.xml?protocol=root"/>
```

Extras – Other VOs

- I don’t know ☺
  - Ask your collaborators
  - Be the first to try out
  - Share your results

Extras - SRM and xrootd try out standalone FNAL SRM

- Install a standard classical SE, all together with host certificate and gridftp server.
- Download, compile FNAL SRM
  - https://srm.fnal.gov/twiki/bin/view/SrmProject/UnixFSSrm
  - Edit /etc/grid-security/grid-mapfile
  - Edit .srmconfig/dcache.kpwd in srm dir
  - Edit bin/run-unix-fs in srm dir, set GRIDFTP_HOST and GRIDFTP_PORT
- Since it's using host certificate, it must be run as root.
- Don't bother with srmclient, use srmcp from your LCG UI. Test that you can srmcp in/out of your server.
- Install xrootd, run as a non-privileged user.
  - For readonly access, it must be able to read all files
  - For r/w access, better to make xrootd user a member of the VO group you serve.
- Now, you can use both srm:// and root:// protocol to access the same files on this server.
- *This only works on one server – where SRM runs. There are some efforts to extend it to use with xrootd cluster. Interested to contribute? Let me know.*

Extras - Patching your old ROOT installations

- Xrootd is included since ROOT v.4.02f
- But your site users can still use it with older applications!
- You can download code of XTNetClient, compile it against your ROOT v3 or v4 installation, using instructions:
  - http://xrootd.slac.stanford.edu/build/rootclient.html
- Add to your system.rootrc file the line below (and test!)

```
+Plugin.TFile:       ^root:    XTNetFile     XTNetFile "XTNetFile(const
char*,Option_t*,const char*,Int_t,Int_t)"
```

Getting more info

- Xrootd web page
  - http://xrootd.slac.stanford.edu
  - See Presentations, Documentation sectins
- Mailing list – stay current, ask questions
  - Xrootd-l@slac.stanford.edu
  - Contact peter.elmer@cern.ch to sign up
- Alice
  - http://alien.cern.ch/twiki/bin/view/AliEn/HowToInstallXrootd

Last advises

- Start from the simplest config
- Test on every step
- Report all bugs and findings
- Consult before using some advanced functions

Good luck!