XrootD/Scalla suite Status

xrootd / olbd / cmsd

Fabrizio Furano
CERN IT
09-Apr-08

http://xrootd.slac.stanford.edu

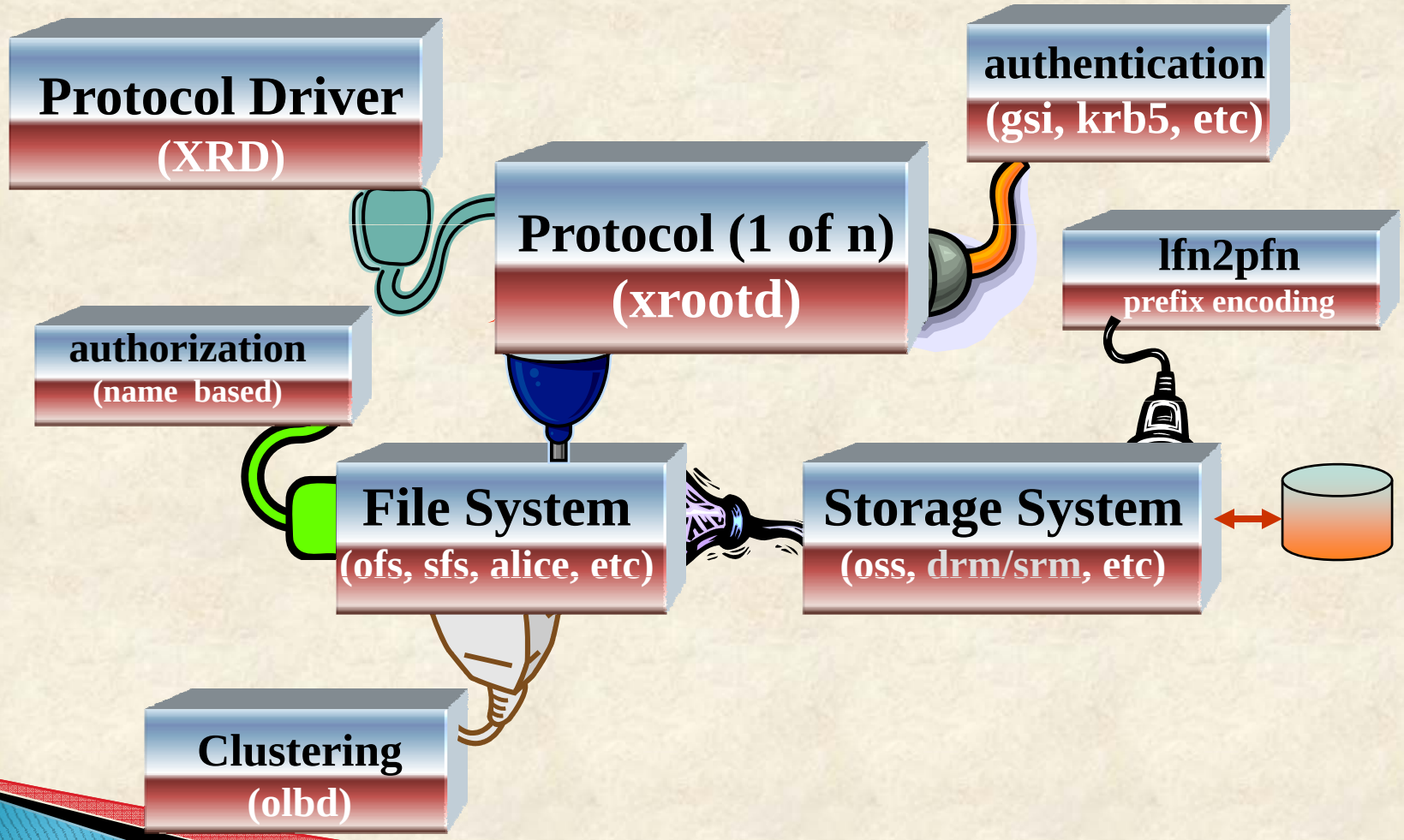
What is Scalla?

- Structured Cluster Architecture for Low Latency Access
  - 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 cmsd servers (formerly olbd)
    - *Exponentially scalable and self organizing*

Scalla 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 dispersement (fast concurrent opens)
- Wide usability
  - Generic Mass Storage System Interface (HPSS, RALMSS, Castor, etc)
  - Full POSIX access
  - Server clustering (up to 200Kper site) for scalability
- 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 (the jobs recover failures - no crashes! – any factor of redundancy possible on the srv side)
  - Self-organizing servers remove need for configuration changes
  - No database requirements (high performance, no backup/recovery issues)

xrootd Plugin Architecture



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
    - Reconstructs everything to present it as a simple interface
    - *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, the overall functionality ***can*** be fully preserved
    - *Redundancy, MSS staging of replicas, …*
    - *Can means that weird deployments can give it up*
      - 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

- 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*

Courtesy of Gerardo Ganis (CERN IT

Available protocols

- 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
- ALICE security tokens
  - Emphasis on ease of setup and performance

Courtesy of Gerardo Ganis (CERN PH-SFT)

Authorization

- Authentication creates a security context with (user, group, role)
- Abstract authorization interface: XrdAccAuthorize
  - Can implement ad hoc schema as a dynamic plug-in
    - *Everything can be done*
    - *But scalable choices are not an option*
- Default implementation:
  - Capability list oriented based on host, user name
  - Privileges set-up à la Windows XP
  - Directory prefix based
    - *Support templates and fungible capabilities*

Courtesy of Gerardo Ganis (CERN IT

Is that all?

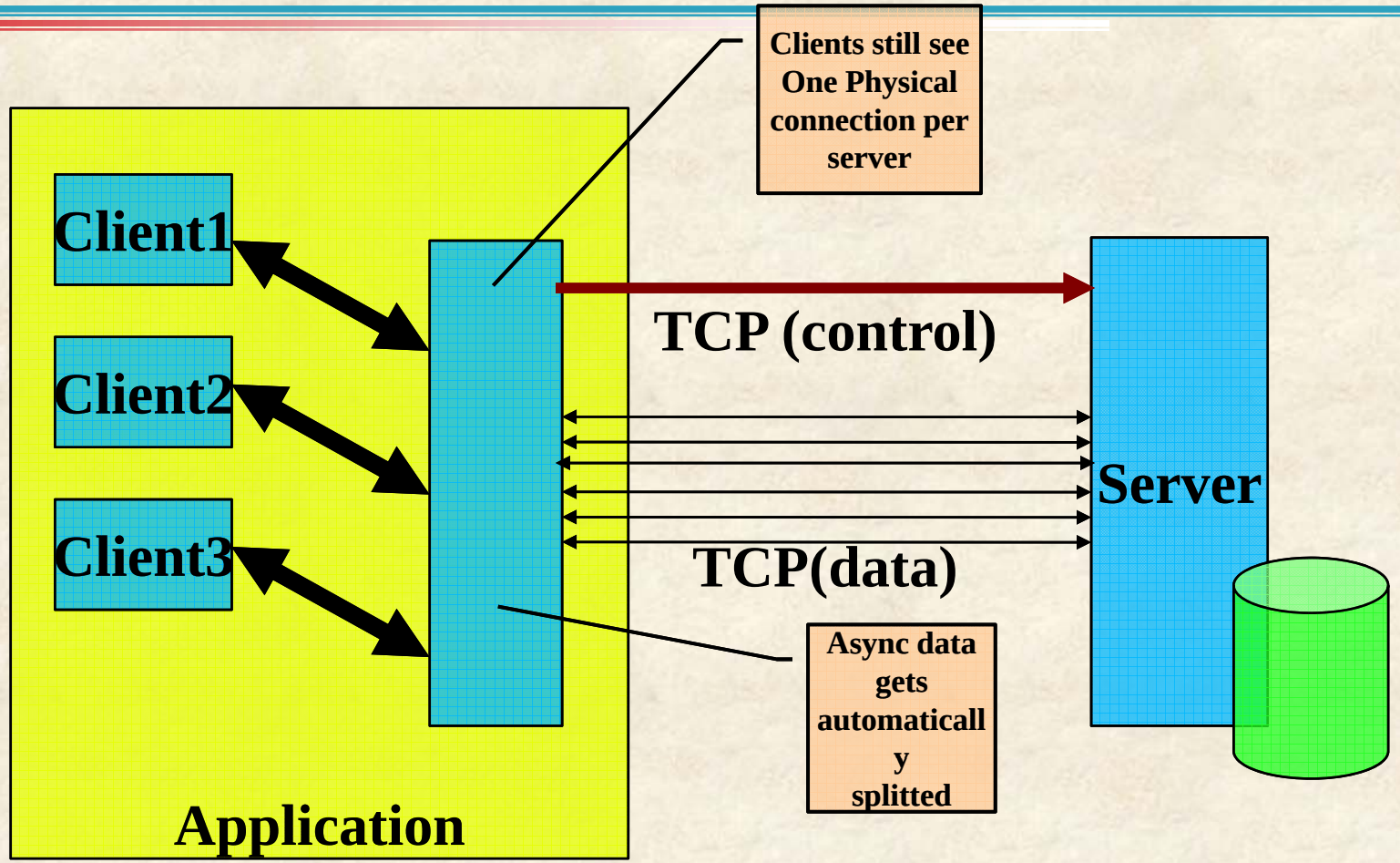
- Very efficient and scalable server side
  - Clusterizable to multiply performance and capacity
- Robust client-server interaction

- NO…. We are missing a lot of things
  - new trends
  - mixed features
  - The users' creativity has been much higher than expected
    - *Ours too…*

New trends

- Client-server WAN interactions
  - Is it true that through WAN we can only hope to create file replicas?
    - *Even if we don't need them?*
- Through-WAN cluster aggregations
  - What about inter-site interactions?
  - A unique meta-cluster or several sites collaborating?
- Xrootd based file system
  - Nice way to put at work external apps
    - *For those who require a file system at any cost*

Multiple streams (2/3)



Multiple streams (3/3)

- It is not a copy-only tool to move data
  - Can be used to speed up access to remote repos
  - Transparent to apps making use of *_async reqs
- xrdcp uses it (-S option)
  - results comparable to other cp-like tools
- For now only reads fully exploit it
  - Writes (by default) use it at a lower degree
    - *Not easy to keep the client side fault tolerance with writes*
- Automatic agreement of the TCP windowsize
  - You set servers in order to support the WAN mode
    - *If requested… fully automatic.*

Motivation

- We want to make WAN data analysis convenient
  - A process does not always read every byte in a file
  - The typical way in which HEP data is processed is (or can be) often known in advance
    - *TTreeCache does an amazing job for this*
  - xrootd: fast and scalable server side
    - *Makes things run quite smooth*
    - *Gives room for improvement at the client side*
      - About WHEN transferring the data
      - There might be better moments to trigger a chunk xfer
        - with respect to the moment it is needed
      - Better if the app has not to pause while it receives data
      - Many chunks together
        - Also raise the throughput in many cases

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*

- 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

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 ”slides through” keeping the network pipeline full
    - *Via asynchronous requests*
- Result: Data transfer goes in parallel with the computation
  - Throughput comparable to “batch” file-copy tools
    - *70-80% of it and we are doing a live analysis, not a file copy!*

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

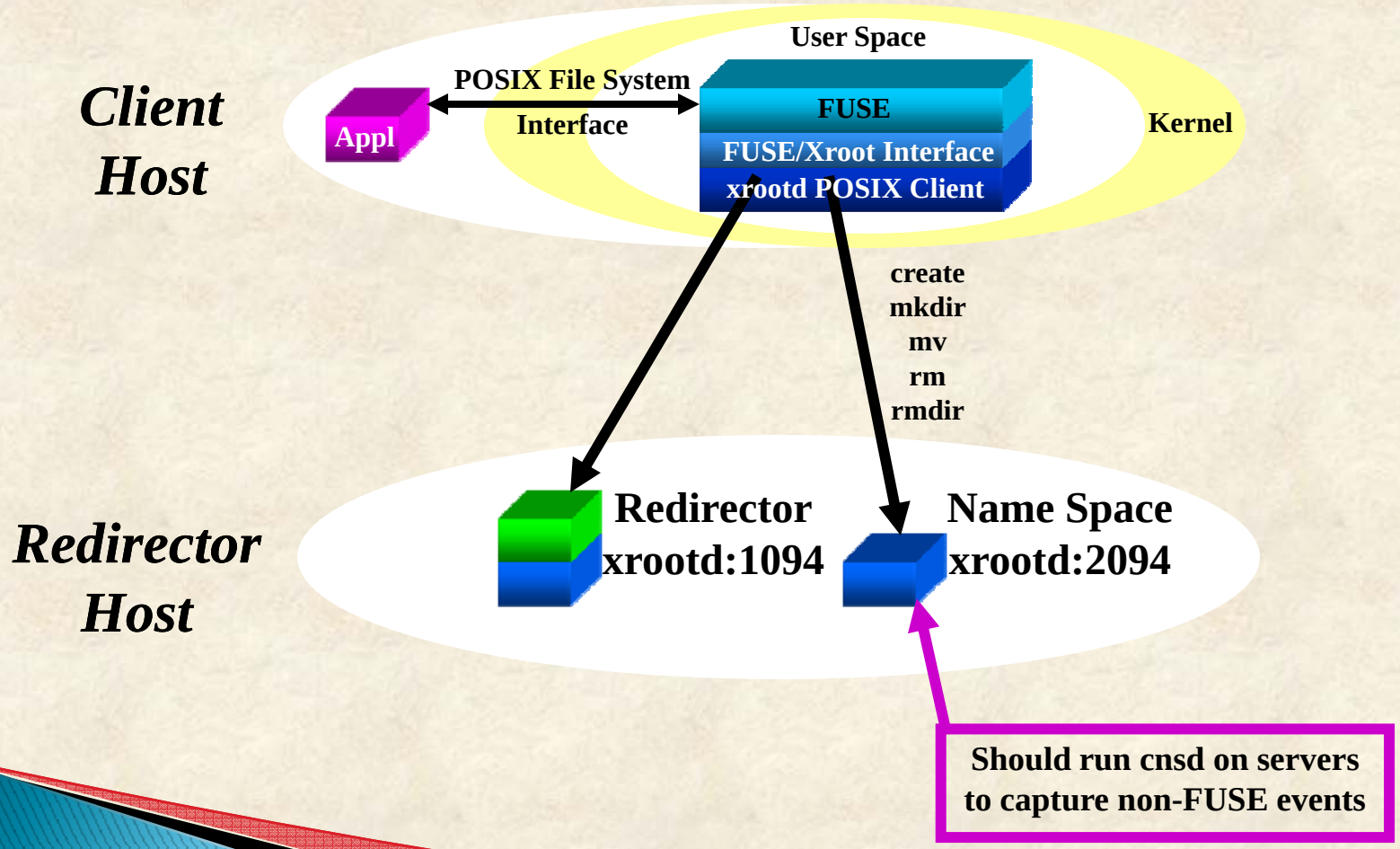
Data System vs File System

- 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?
- The answer is ….
  - It can to a degree that may be good enough
- We relied on FUSE to show how

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 fully functional (used to provide SRM access for ATLAS)

XrootdFS (Linux/FUSE/Xrootd)



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)*
- So, it's good for the SRM-side of a repo
  - But not for the job side

Cluster globalization

- Up to now, xrootd clusters could be populated
  - With xrdcp from an external machine
  - Writing to the backend store (e.g. CASTOR)
- FTD in ALICE now uses the first
  - Load problems, all the traffic through one external machine
    - *Close to the dest cluster*
- If a file is missing or lost
  - For disk and/or catalog screwup
  - Job failure
    - *... manual intervention needed*

Virtual MSS

- Purpose:
  - A request for a missing file comes at cluster A,
    - *A assumes that the file ought to be there*
      - And tries to get it from the collaborating clusters
      - Through a request to the ALICE Global redirector
  - If a pre-stage (prepare) request comes,
    - *A does the same*
- Note that A itself is subscribed to the ALICE global redirector
- In theory, Alien/Aliroot will never request files that should not be there
    - *But DBs go out of sync with the reality sometimes*

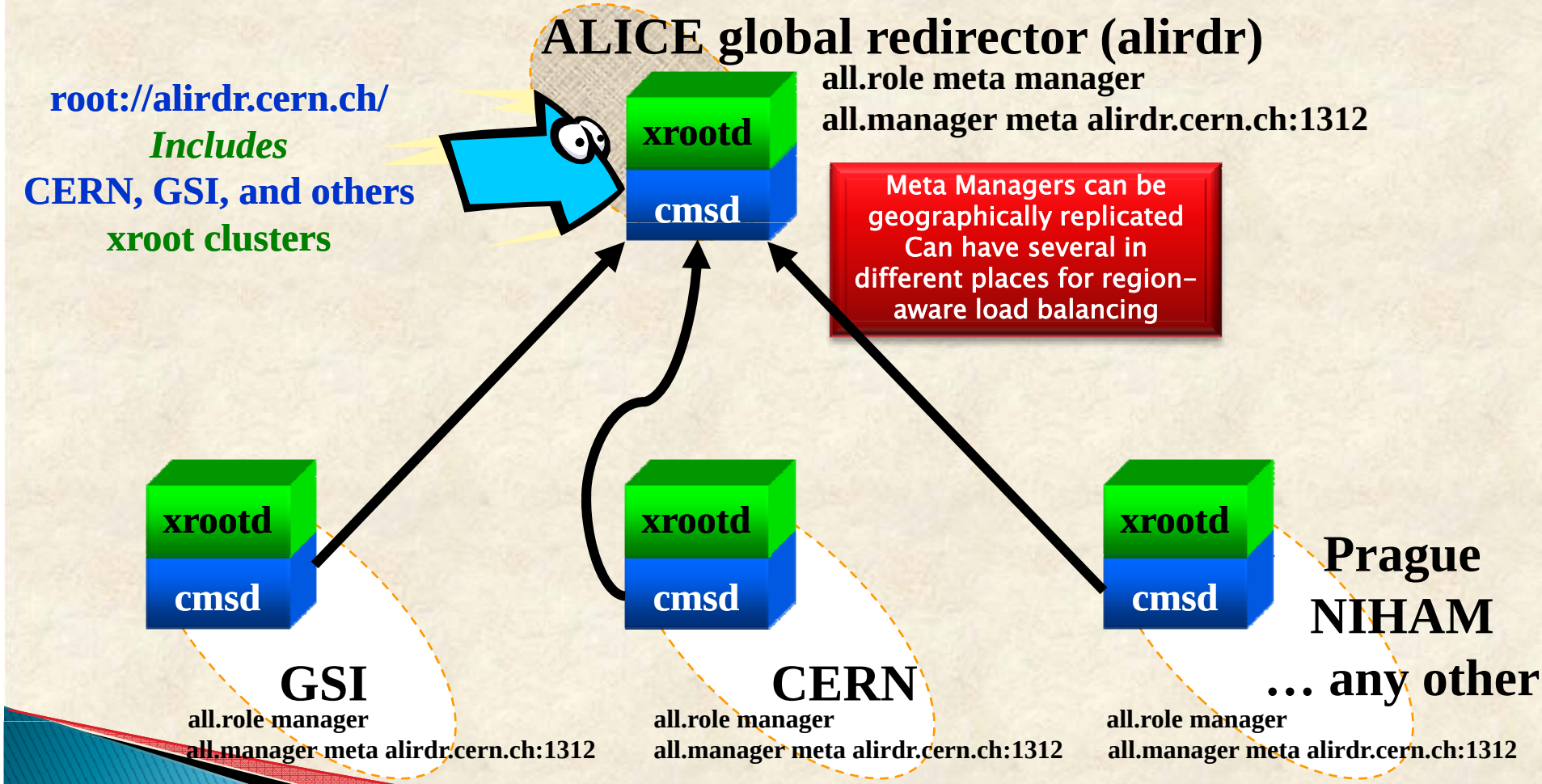
Why Globalize?

- Uniform view of participating clusters
  - Can easily deploy a virtual MSS
    - *Included as part of the existing MPS framework*
    - *A subcluster can get data from the metacluster (e.g. missing files)*
  - Try out real time WAN access
    - *You really don't need data everywhere!*
- ALICE is moving in this direction
  - The non-uniform name space is not an obstacle anymore
    - *For historical reasons, a file had different path prefixes and name suffixes*
      - Suffix has been removed
      - Site-dependent path prefixes are translated correctly now
  - xrootd-based sites in ALICE will support both
    - *The local historical name space*
    - *The uniform global one*

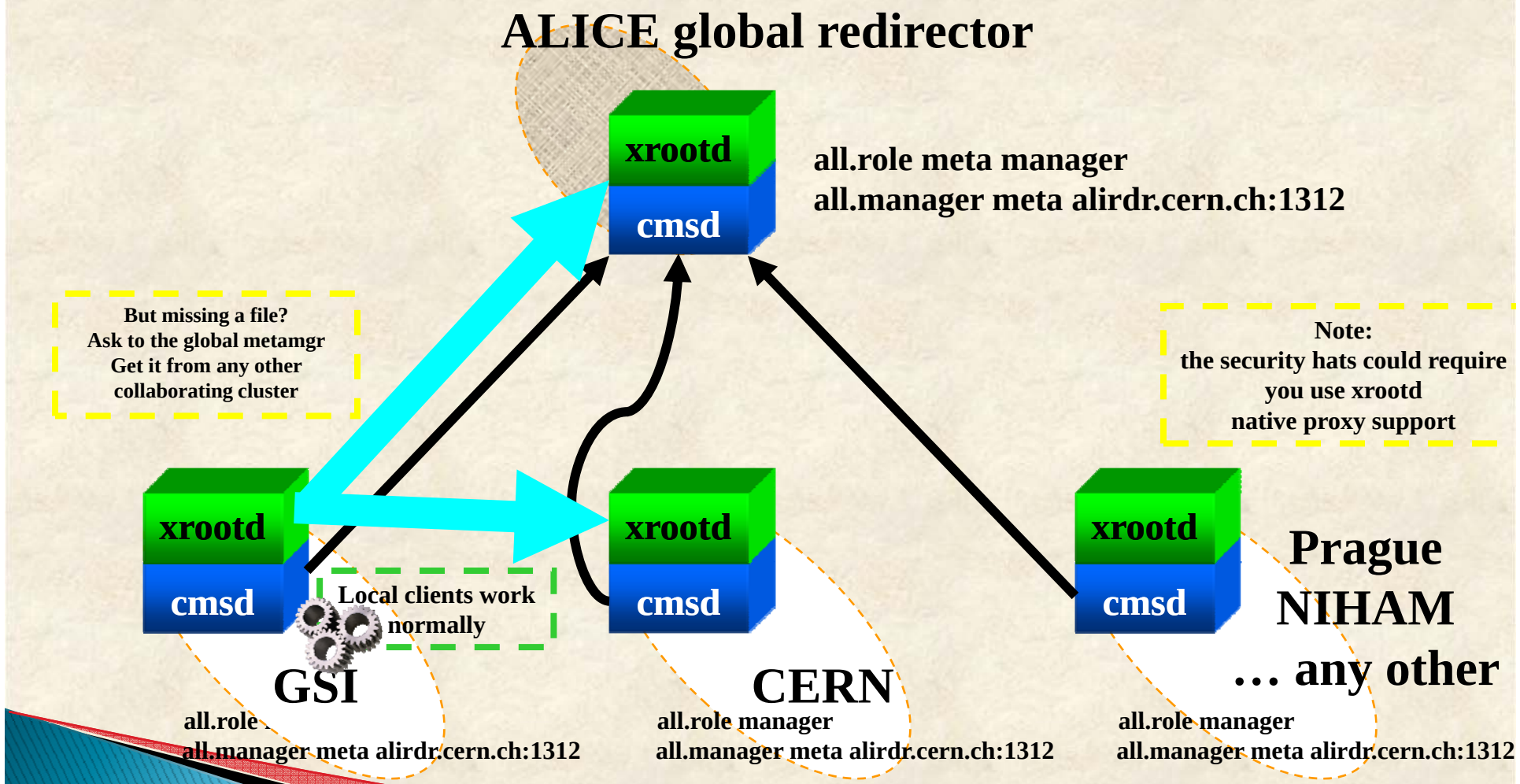
Many pieces

- Global redirector acts as an xrootd meta-manager
- Local clusters subscribe to it
  - And declare if they
    - *Export to the global mechanism only the online files or from their local MSS too*
- Local clusters (without local MSS) treat the globality as a very big MSS
  - Coordinated by the ALICE Global redirector
    - *Load balancing*
    - *Priority to files which are locally online*
    - *Fast file location*
- True, robust, realtime collaboration between storage elements!
  - Expecially tier-2s

Cluster Globalization… an example



The Virtual MSS Realized



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
      - No costly out of time DB lookups
  - File (pre)fetching on demand
    - *Can be transformed into a 3rd-party GET (by asking for a source)*
  - Practically no need to track file location
    - *But does not stop the need for metadata repositories*

Virtual MSS++

- This architecture uses the new CMSD, the replacement for OLBD
  - Hence, we can pass additional info associated to the request
- With a little additional dev
  - We can specify in a request the preferred source for a file to be fetched from
    - *It can be a non-VMSS enabled site (e.g. a dCache-based site)*
  - People love to call this third party copy
    - *They are not right nor wrong*
    - *But it's a get-like operation*

So, what? Embryonal ALICE VMSS

- Test instance cluster @GSI
  - Subscribed to the ALICE global redirector
  - Until the xrdCASTOR instance is subscribed, GSI will get data only from voalice04 (and not through the global redirector coordination)
    - *The mechanism seems very robust, can do even better*
    - *To get a file there, just open or prestage it*
    - *Need of updating Alien*
      - Staging/Prestaging tool required (done)
      - FTD integration (done, not tested yet)
      - Incoming traffic monitoring through the XrdCpapMon xrdcp extension (which is not xrdcpapmon)… done!
        - Technically, no more xrdcpapmon, just xrdcp does the job
        - So, one tweak less for ALICE offline
    - *Many many thanks to Silvia Masciocchi, Anna Kreshuk & Kilian Schwarz*
      - For the enthusiasm and awesome support

ALICE VMSS Step 2

- Point the GSI “remote root” to the ALICE global redirector
  - As soon as the xrdCASTOR instance (at least!) is subscribed
  - Still waiting for it, an upgrade to the Scalla PROD version
  - No functional changes
    - *Will continue to 'just work', hopefully*
- This will be accomplished by the complete revision of the setup (work in progress)

ALICE VMSS Step 3 – 3 rd party GET

- Not terrible dev work on
  - Cmsd
  - Mps layer
  - Mps extension scripts
  - deep debugging and easy setup
- And then the cluster will honour the data source specified by FTD (or whatever)
  - Xrootd protocol is mandatory
    - *The data source must honour it in a WAN friendly way*
      - Technically means a correct implementation of the basic xrootd protocol
      - Source sites supporting xrootd multistreaming will be up to 15x more efficient, but the others still will work

Problems? Not yet.

- There should be no architectural problems
  - *Striving to keep code quality at maximum level*
  - *Awesome collaboration*
- BUT...
  - If or when the VMSS will be used beyond what FTD will ask it to do...
    - *Somebody will realize that everything is already there… everywhere*
      - Even the data
  - The architecture will prove itself to be ultra-bandwidth-efficient
    - *Or greedy, as you prefer*
  - We designed an enhancement, requiring some dev at some point (Andy+Fabrizio)
    - *NEW! Scheduled in May!*
    - *But first… let's finish the test setup!*

Latest news – ALICE Monitoring

- The info we see in Monalisa are somehow reductive
  - Recent additions in the ROOT side monitoring
    - *Gave very good surprises about data access @CERN*
      - Very high transaction rate (and very low load)
      - Also feeding to remote sites (data access, not bulk files)
      - Low latency and high throughput, which the actual monitoring cannot catch
    - *But evidenced weaknesses in the monitoring*
      - Both logical and functional
    - *So… TFile-TAlienFile-TXNetFile-TVirtualMonitoring-TMonaLisaWriter in ROOT are under a major revision*
      - Work in progress… NOW

Acknowledgements

- Old and new software Collaborators
  - Andy Hanushevsky, Fabrizio Furano (client-side), Alvise Dorigo
  - Root: Fons Rademakers, Gerri Ganis (security), Bertrand Bellenot (windows porting)
  - Alice: Derek Feichtinger, Andreas Peters, Guenter Kickinger
  - STAR/BNL: Pavel Jackl, Jerome Lauret
  - Cornell: Gregory Sharp
  - SLAC: Jacek Becla, Tofigh Azemoon, Wilko Kroeger, Bill Weeks
  - BaBar: Peter Elmer
- Operational collaborators
  - BNL, CERN, CNAF, FZK, INFN, IN2P3, RAL, SLAC