



DSS

XROOTD

Towards an unique protocol and storage
management on the GRID

Fabrizio Furano



- HEP experiments are very big data producers
- The HEP community is a very big data consumer
 - Analyses rely on statistics on complex data
 - Scheduled (production) processing and user-based unscheduled analysis
- Performance, usability and stability are the primary factors
 - The infrastructure must be able to guarantee access and functionalities
 - The softwares must use the infrastructures well
- Having NO native interoperability is just asking for troubles
 - Or asking for super-complex glue systems just to do simple things

- In 2002 there was the need of a data access system providing basically:
 - Compliance to the HEP requirements
 - “Indefinite” scaling possibility (up to 200Kservers)
 - Maniacally efficient use of the hardware
 - Accommodate thousands of clients per server
 - Great interoperability/customization possibilities
- In the default config it implements a non-transactional distributed file system
 - Efficient through LAN/WAN
 - Multi-site federations make ONE huge storage
 - Or make single SEs able to cooperate
 - Not linked to a particular data format
 - Particularly optimized for HEP workloads



- WWW-like interactivity for the HEP case
 - Absolutely need a file? Download it!
 - Need to r/w data? Read/Write it as it is where it is!
 - Need to interface with an external system (e.g. an SRM server) ? Keep it external, through simple mechanisms.
- More than WWW, HEP is tough
 - Adapt natively to all the subtle HEP requirements
 - Avoid troubles since the very beginning

- Many sites, exposing the XROOTD protocol
 - Native XROOTD
 - A few with DPM+XROOTD
 - One with CASTOR+XROOTD
 - A few with Dcache's Xrootd protocol implementation
- Native XROOTD + 2 plugins + MonALISA
 - In a simple bundled setup
- Alien points directly to single SEs
 - Privileges local data according to the catalogue's content
 - OCDB accessed via WAN

Towards an unique protocol and storage management on the GRID

- In general, an user will:
 - Decide which analyses to perform for his new research
 - Write the code which performs it
 - Typically a high level macro or a “simple” plugin of an experiment-based software framework
 - Ask a system about the data requirements
 - Which files contain the needed information
 - Ask another system to process his analysis
 - Collect the results

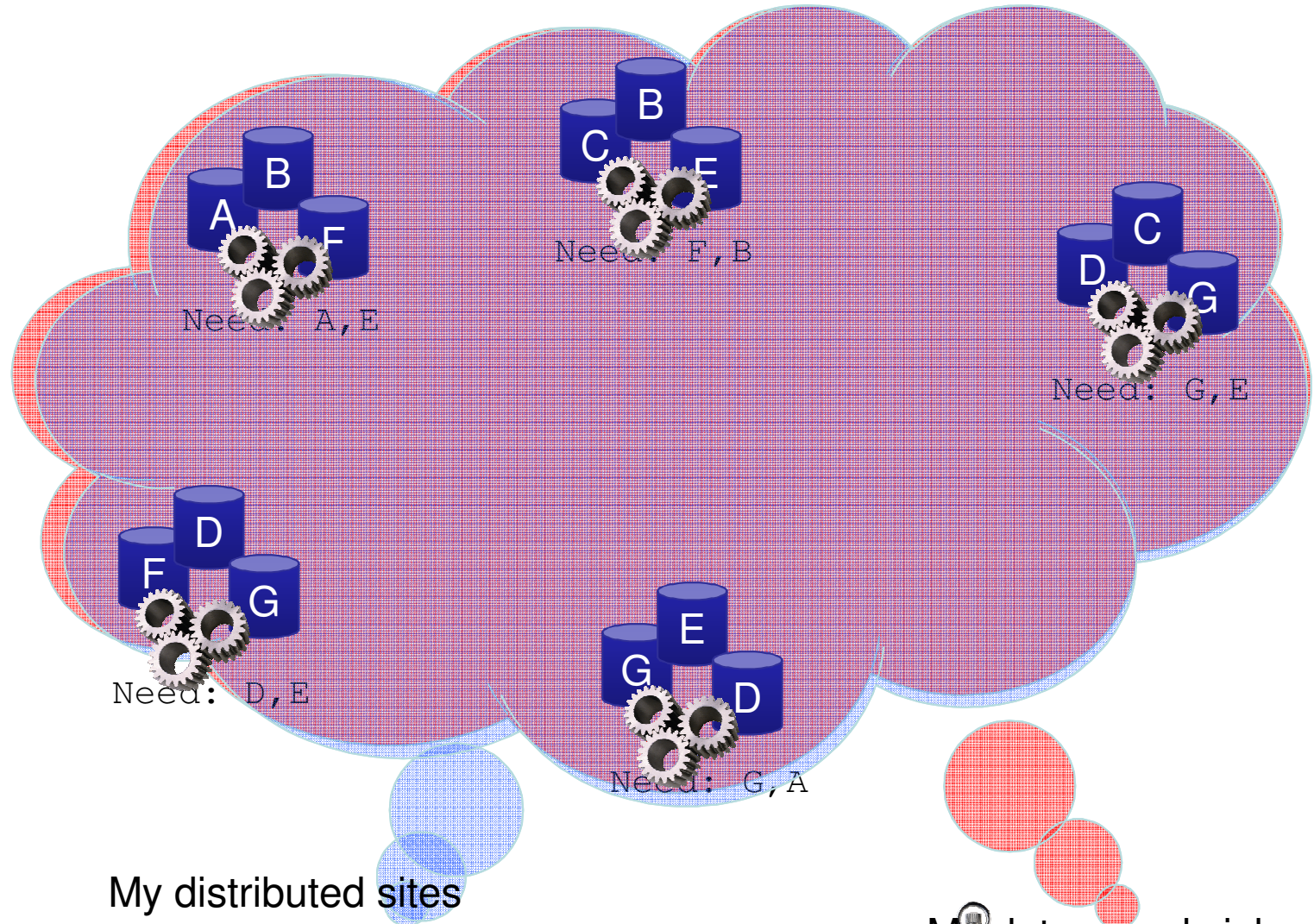
Typically an experiment-based metadata repository and/or file catalogue

Typically the GRID (WLCG) or batch farms or PROOF

This will likely become also his own computer as the hw performance increases and the sw uses it efficiently

Towards an unique protocol and storage management on the GRID

- Files and datasets are stored into Storage Elements, hosted by sites
 - The decision is often taken when they are produced
- Processing jobs are very greedy
 - Up to 15-20 MB/s
- The GRID machinery (ev. Together with some service of the experiment) decides where to run a job
 - The service can also be human-based (!)
- Matching the locations of the data with the available computing resources is known as the “GRID Data Management Problem”.



My distributed sites
With pre-filled storage
With computing farms

My data greedy jobs



Towards an unique protocol and storage management on the GRID

- Having an unique WAN+LAN compliant protocol allows to do the right thing
 - Exploit locality whenever possible (=most of the times)
 - Do not worry too much if a job accesses some data file which is KNOWN not to be in the same site.
 - Explicitly creating a new replica just for a job takes much more time and risk.
 - Access condition data ONLY via WAN

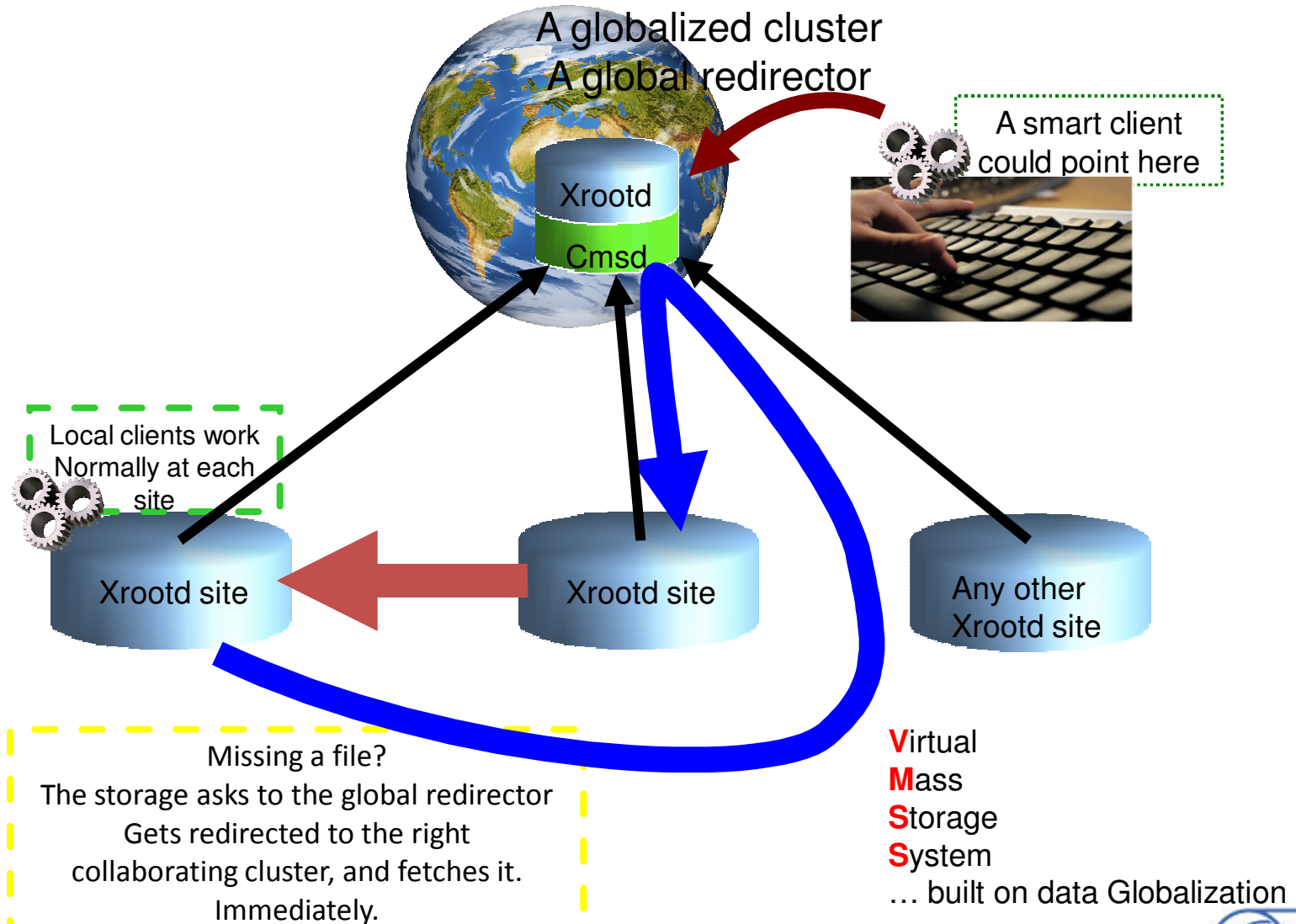
- The historical approach was to implement a “catalogue” using a DBMS
 - This “catalogue” knows where the files are
 - Or are supposed to be
- This can give a sort of “illusion” of a worldwide file system
 - It must be VERY well encapsulated, however
 - One of the key features of the AliEn framework
 - But 101% encapsulation is just impossible
 - It would be nicer if this functionality were inside the file system/data access technology
 - No need for complex systems/workarounds

- Each server manages a portion of the storage
 - many servers with small disks, or
 - fewer servers with huge disks
- Low overhead aggregation of servers
 - Gives the functionalities of an unique thing
 - A non-transactional file system
- Efficient LAN/WAN byte-level data access
- Protocol/architecture built on the tough HEP requirements

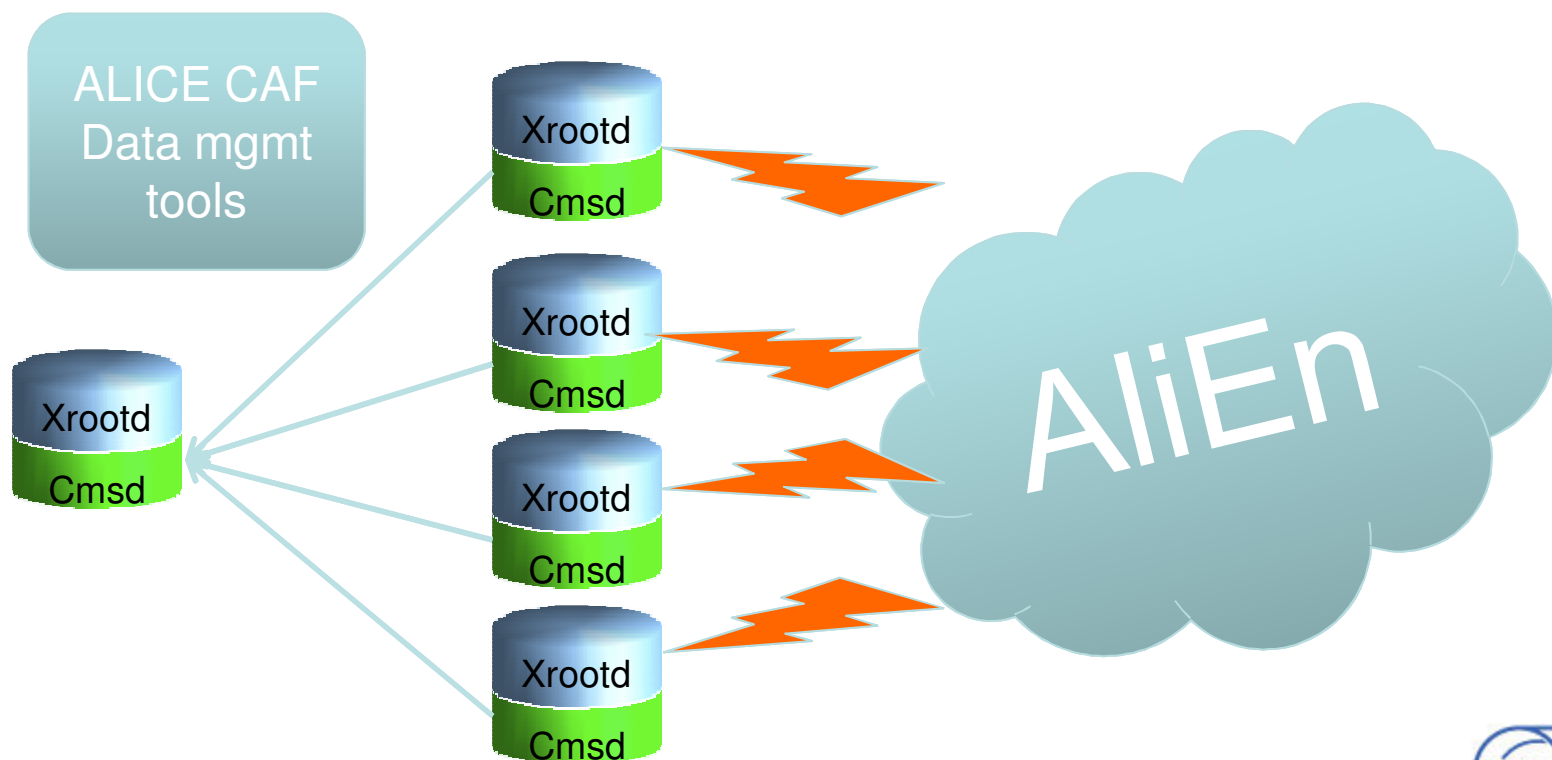
- Analysis clients work at a site
- The local storage, accessed through URLs acts as a proxy of the worldwide storage
 - A local r/w cache
 - In practice, if a file is missing, it is ‘fetched’ from an external system
 - Or a file can be ‘requested’ to appear
 - Must have a sufficient size to reduce the “miss rate”
 - Efficient data movement tools can populate it as well

- Build efficient storage clusters
- Aggregating storage clusters into WAN federations
- Access efficiently remote data
- Build proxies which can cache a whole repository
 - And increase the data access performance (or decrease the WAN traffic) through a decent ‘hit rate’
- Build hybrid proxies
 - Caching an official repository while storing local data locally

- Suppose that we can easily aggregate sites
 - And provide an efficient entry point which “knows them all natively”
- We could use it to access data directly
 - Somebody tried, with very good results
- We could use it as a building block for a proxy-based structure
 - If site A is asked for file X, A will fetch X from some other ‘friend’ site, though the unique entry point
 - A itself is a potential source, accessible through the entry point



- Data is proxied locally to adequately feed PROOF
- From the 91 AliEn sites



Towards an unique protocol and storage management on the GRID

- Take a PROOF cluster, with XROOTD storage, make it easily installable
- Put the xrd-dm plugin by M.Vala
 - Transform your AF into a proxy of the ALICE globalized storage, through the ALICE GR
 - If something needed is not present, it will be fetched in FAST
 - Also support sites not seen by the GR, through internal dynamic prioritization of the AliEn sites.
- Data management: how does the data appear?
 - Manual (or supervised) pre-staging requests, or:
 - Suppose that an user always runs the same analysis several times
 - Which is almost always true
 - The first round will be not so fast but working, the subsequent will be fast
- And you have the ALICE SKAF (Kosice, Slovakia)

- Logical File Name: a filename as it is seen by the application
 - E.g. /data/furano/myfile
- Physical File Name: the filename as it is physically stored
 - E.g. with the mount point prefix before
 - /mnt/sda/raw/data/furano/myfile
 - Or a completely different file name
 - /mnt/sda/data/2763-5427-4527-4573-2572-452.001.dat
- This translation has to be done “internally” in the site’s storage
- It’s a powerful idea to implement location transparency
- This distinction can make things easier or difficult.
 - A simple rule (e.g. a local directory prefix) can make it easier to aggregate servers into clusters, exposing a common name space
 - An older idea was to completely detach them, and assign a number (or something totally unrelated) as a PFN
 - Doing so, the association pieces must be kept in a DB

- Often a source of misunderstandings
 - AliEn has LFNs
 - They are the user-readable names
 - AliEn converts them to PFNs
 - The ugly filenames with numbers
 - An AliEn PFN is considered by XROOTD as an XROOTD LFN
 - XROOTD takes care internally of its PFN translation
 - Hiding the internal mount points
 - At the end:
 - USERS see Alien LFNs
 - SYSADMINS see XROOTD PFNs (= Alien PFNs with a prefix)

- Historically, the *AF admins didn't like to deal with the AliEn PFNs
 - The ugly filenames made by numbers
 - They wanted to store only LFNs (i.e. the human-readable filenames)
 - So, there already are some places storing native LFNs
- If these XROOTD-based storages get aggregated by the Global Redirector:
 - Their content will be accessible as a whole, with no need of translating names through AliEn.
 - Interesting futuristic experiment (pioneered by SKAF)
 - The *AFs could give data each other, by using the VMSS
 - So, a part of the ALICE storage could be accessed directly with nice names, skipping the ROOT-side AliEn xlation.

- GRID WNs to start PROOF interactive workers
 - Ongoing interesting developments, e.g. PoD by Anar Manafov
- Data globalization/proxying seems ideal to feed them with data
 - Thanks to the “unique protocol” for direct access
 - Ideas are welcome...
- The purpose is:
 - Give handles to build a lightweight/dynamic Data Management structure
 - Whose unique goal is to work well
 - Enable interactivity for users

- The XROOTD usage is gaining importance
 - Many kinds of components to design massively distributed data access systems
 - Challenge: create/evolve the newer ones, e.g. :
 - chunk-based and file-based proxies
 - What about a personal cache/proxy ?
 - Bandwidth/queuing managers
 - Goal: a better user experience with data
 - Without the right data in the right moment nothing can work
 - “moment” does not mean “place”

- Proxying is a concept, there are basically two ways it could work:
 - Proxying whole files (e.g. the VMSS)
 - The client waits for the entire file to be fetched in the SE
 - Proxying chunks (or data pages)
 - The client's requests are forwarded, and the chunks are cached in the proxy as they pass through
- In HEP we do have examples of the former
- It makes sense to make also the latter possible
 - Some work has been done (the original XrdPss proxy or the newer, better prototype plugin by A.Peters)

- In the data access frameworks (e.g. ROOT) many things evolve
- Applications tend to become more efficient (=greedier)
- Applications exploiting multicore CPUs will be even more
 - An opportunity for interactive data access (e.g. from a laptop)
 - A challenge for the data access providers (the sites)
 - The massive deployment of newer technologies could be the real challenge for the next years

- “There are MANY things to do”
 - 3 years ago the direct WAN access for HEP was considered Sci-fi, now it is only advanced
 - Thinking about single sites now is very reductive
 - A lot of benefit can come from smarter integrations with the GRID, PROOF, PoD (see Anar’s presentation)
 - We must consider computations AND data management together
 - Multicores are approaching also in “small” computers, making the things more demanding for the data sources and the Data Management infrastructure
 - New ideas can give performance and the illusion of locality
 - For fast-paced analysis, needing fast-paced access

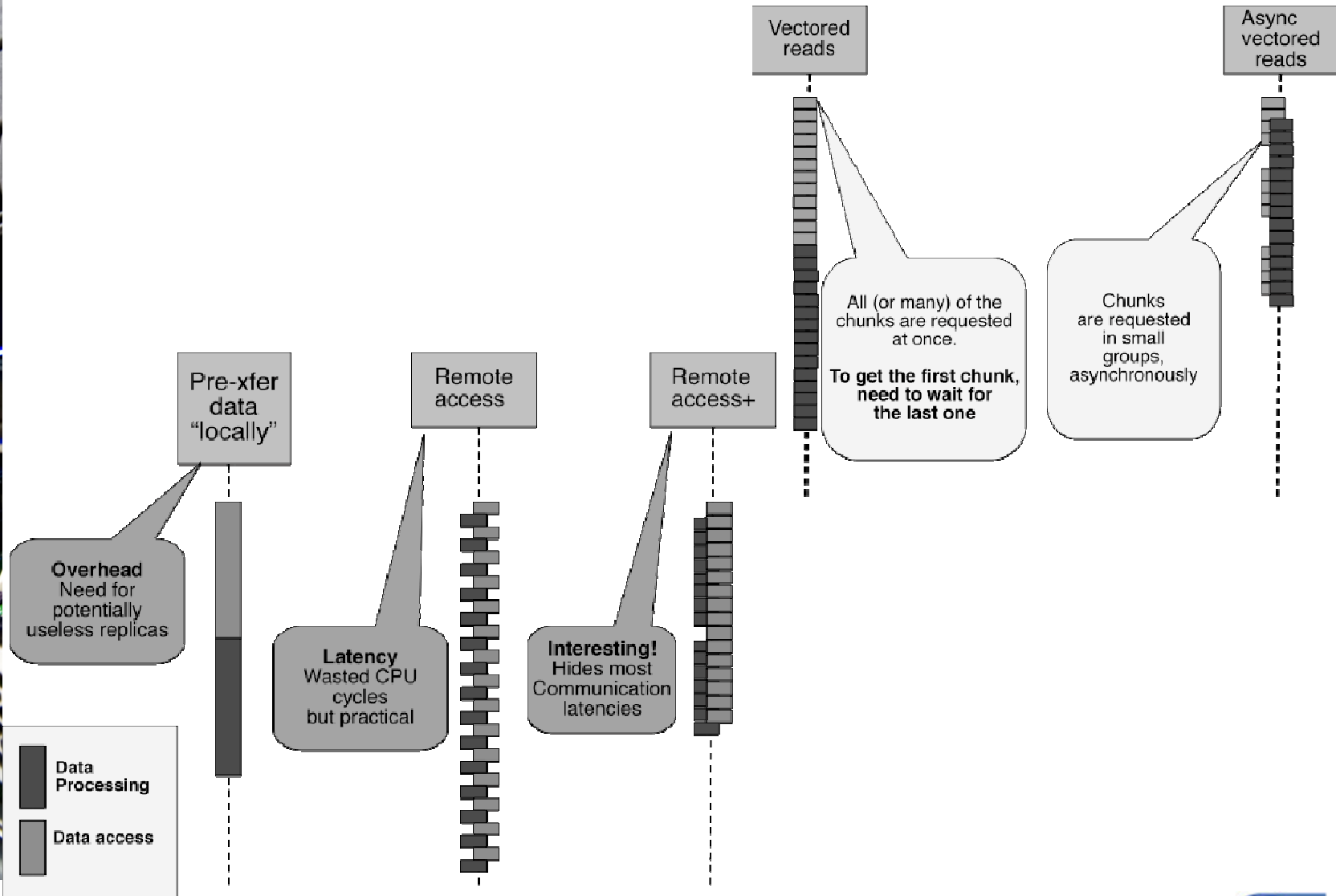


DSS

Thank you!



- The Scalla/xrootd project puts great emphasis in performance. Some items:
 - Asynchronous requests (can transfer while the app computes)
 - Optimized vectored reads support (can aggregate efficiently many chunks in one interaction)
 - Exploits the ‘hints’ of the analysis framework to annihilate the network latency
 - And reduce the impact of the disks’ one by a big factor
 - Allows efficient random-access-based data access through high latency WANs



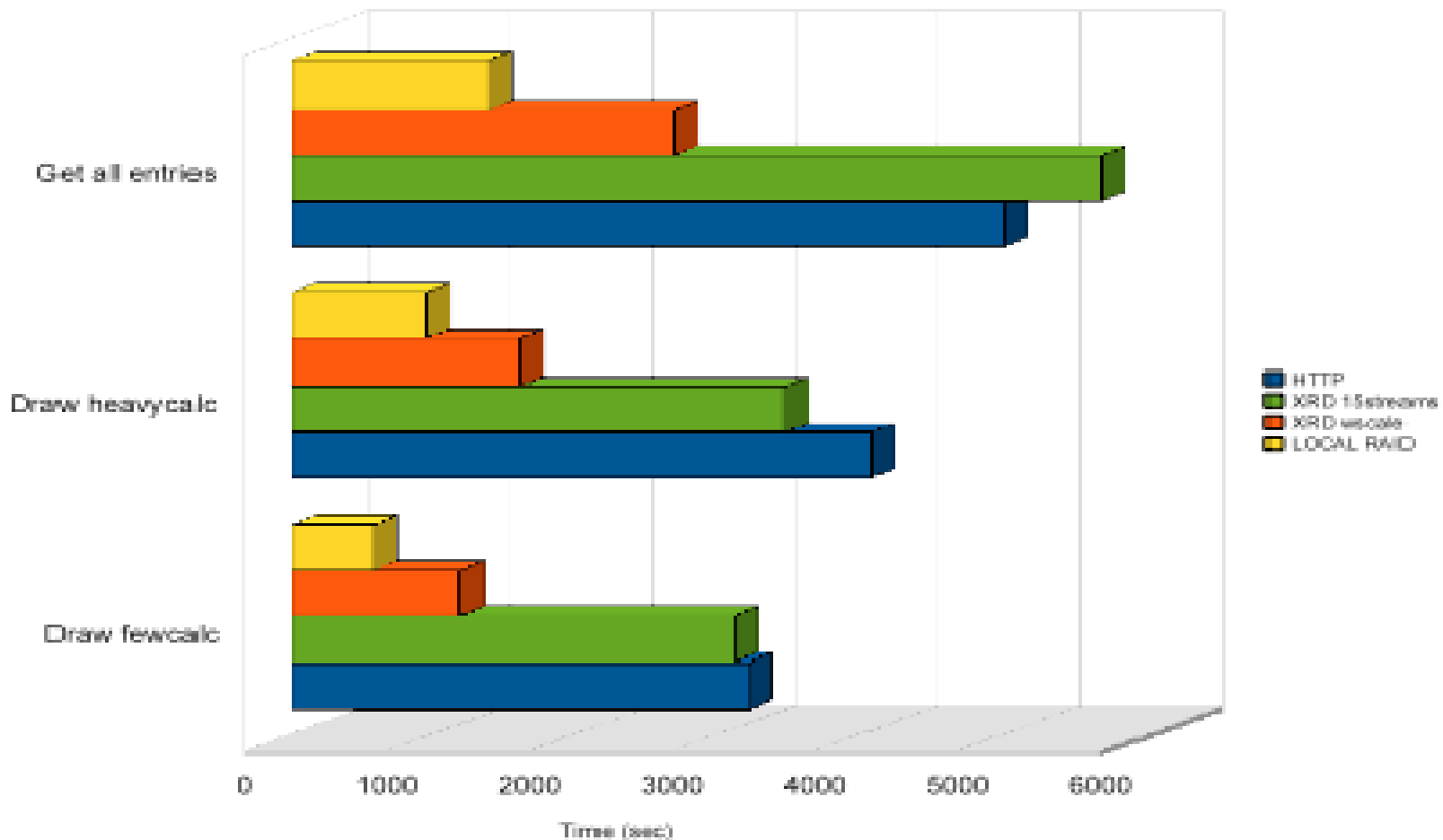
- In WANs each client/server response comes much later
 - E.g. 180ms later
- With well tuned WANs one needs apps and tools built with WANs in mind
 - Otherwise they are walls impossible to climb
 - I.e. VERY bad performance... unusable
 - Bulk xfer apps are easy (gridftp, xrdcp, fdt, etc.)
 - There are more interesting use cases, and much more benefit to get
- ROOT has the right things in it
 - If used in the right way

- Caltech machinery: 10Gb network
- Client and server (super-well tuned)
 - Selectable latency:
 - $\sim 0.1\text{ms}$ = super-fast LAN
 - $\sim 180\text{ms}$ = client here, server in California
 - (almost a worst case for WAN access)
- Various tests:
 - Populate a 30GB repo, read it back
 - Draw various histograms
 - Much heavier than the normal, to make it measurable
 - From a minimal access to the whole files
 - Putting heavy calcs on the read data
 - Up to reading and computing everything
 - Analysis-like behaviour
 - Write a big output ($\sim 600\text{M}$) from ROOT

Thanks to Iosif Legrand
and Ramiro Voicu

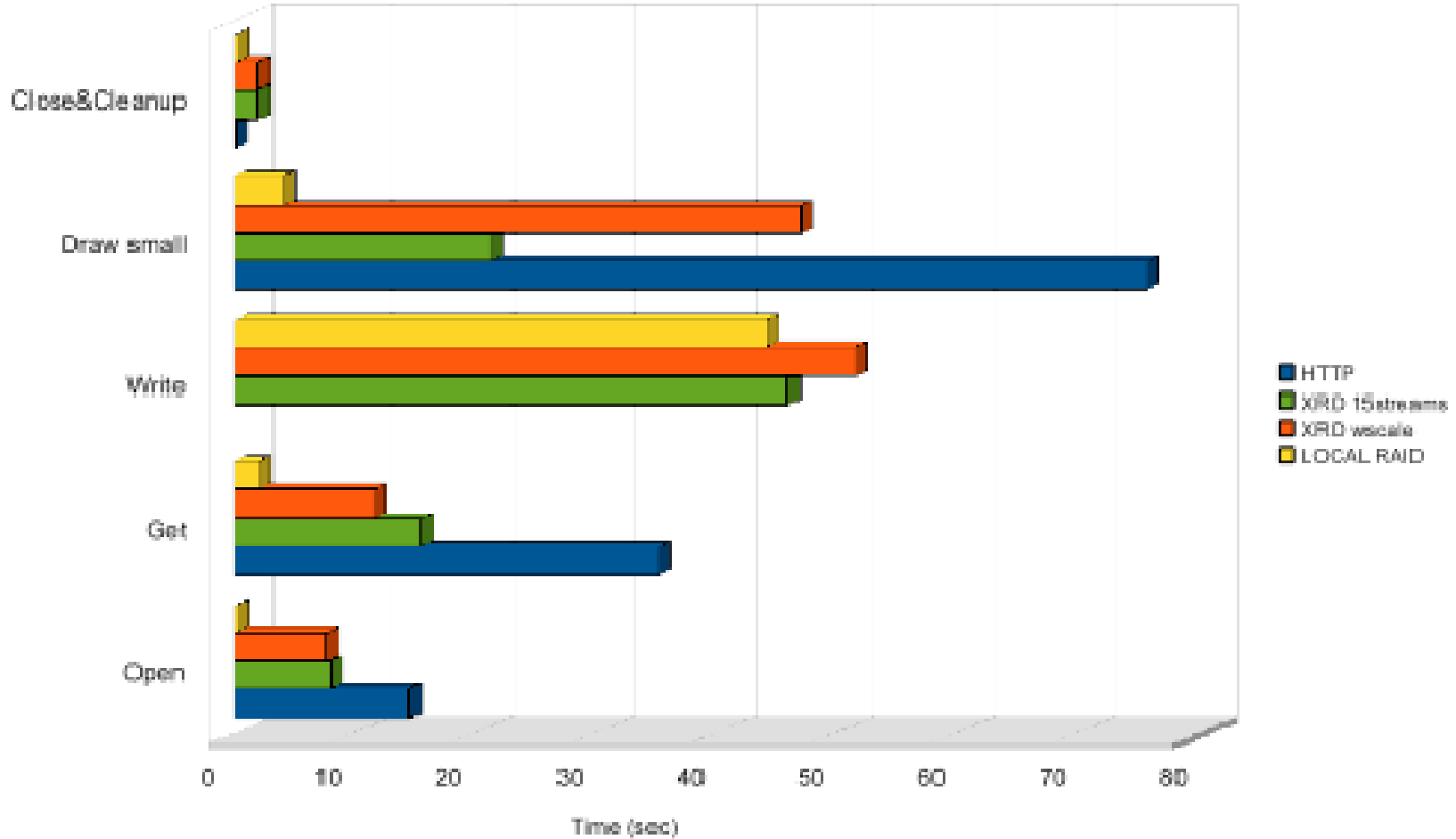


10M Cache - Analyze 10 3G files





10M Cache - Analyze 10 3G files



An estimation of Overheads and write performance

Towards an unique protocol and storage management on the GRID

- Things look quite interesting
 - BTW same order of magnitude than a local RAID disk (and who has a RAID in the laptop?)
 - Writing gets really a boost
 - Aren't job outputs written that way sometimes?
 - Even with Tfile::Cp
- We have to remember that it's a worst-case
 - Very far repository
 - Much more data than a personal histo or an analysis debug (who's drawing 30GB personal histograms? If you do, then the grid is probably a better choice.)
 - Also, since then (2009), the xrootd performance increased further by a big factor for these use cases