



Summary of the Experiments Data Management Inputs

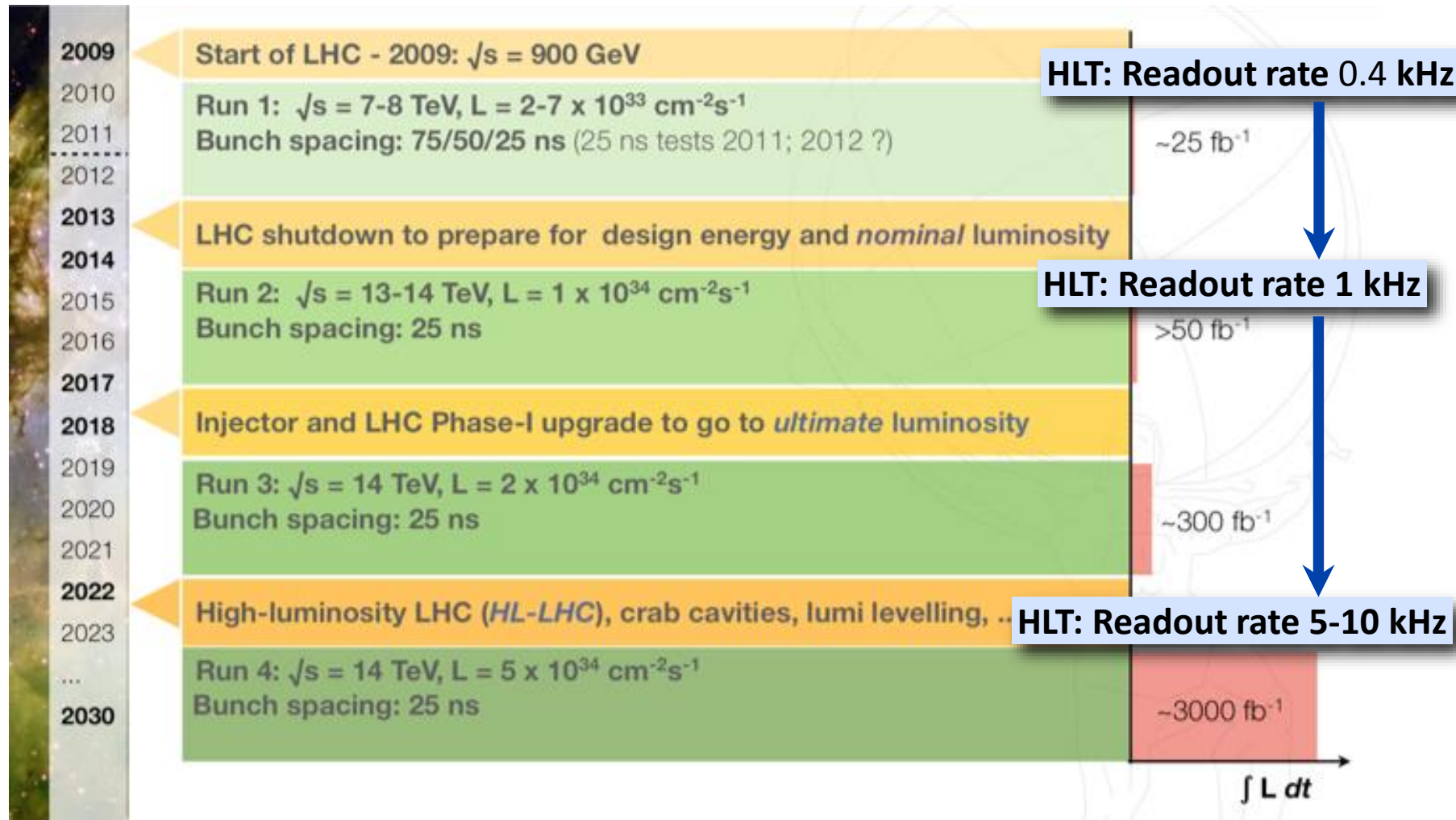
L. Betev, D. Cameron, S. Campana, P.
Charpentier, D. Duellmann, A. Filipcic, A.
Di Girolamo, N. Magini, C. Wissing



Medium Term

- Will focus of the progress that we should aim for in the next 3 years
 - In production for LHC Run-3
- We will introduce some topics for the longer term (LHC Run-4) which should drive some discussion tomorrow.

LHC Upgrade Timeline - the ATLAS and CMS Computing Challenge





- Run2 conditions:
 - 12.5 ++ kHz of HLT output rate
 - Small pileup ($\mu=1.1$)
 - Throughput just below 1 GB/s
 - ↳ In 2015: up to 1.3 GB/s
 - Online reconstruction = offline reconstruction
 - ↳ Allows direct analysis from online data (TURBO stream)
- LHCb major upgrade is for Run3 (2020 horizon!)
 - Luminosity $\times 5$ ($2 \cdot 10^{33}$)
 - Trigger rate... $\times 5$ (at least)
 - Throughput between 6 and 10 GB/s!
 - Trigger (SW only) = offline selection
 - ↳ Stripping is no longer effective (all events are for physics!)
 - ↳ Possibly directly export reconstructed data only



Run-3 computing models and workflows

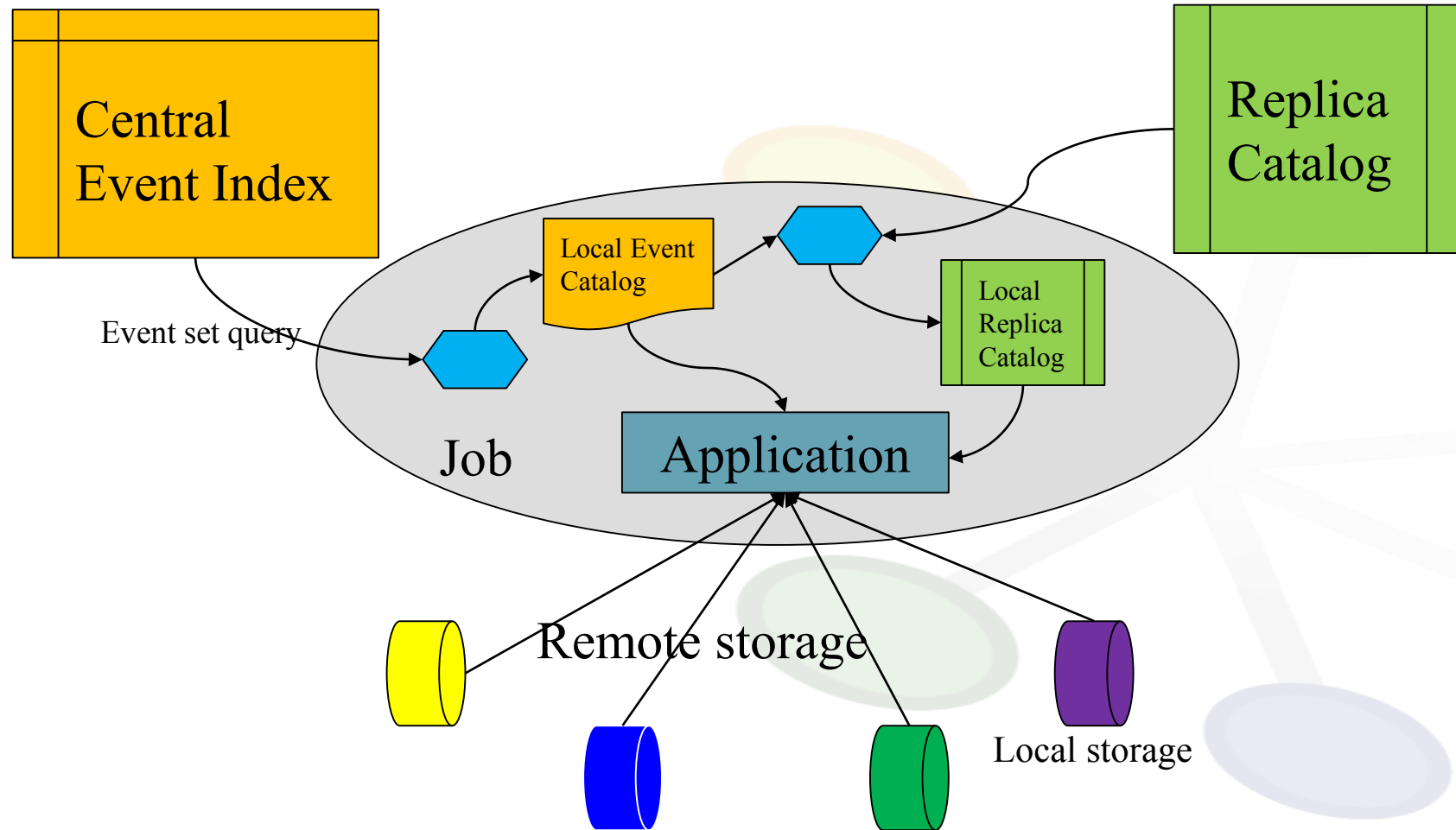


- Reduce number of user jobs on the Grid
 - Centralise ntuple / μ DST creation as “train analysis” (c.f. ALICE)
- Using indices for analysis
 - Replace “stripping + streaming” with “selection + indexation”
 - ↳ Because stripping retention will be high (more selective trigger)
 - Event set query to central (or local) index
 - ↳ Download a local event collection (i.e. direct access addresses)
 - Random access to local or remote data
 - ↳ Using a local replica catalog (Gaudi Federation)
- R&D can start now (2016/17) for:
 - Setting up train analyses
 - ↳ framework similar to stripping
 - Data indexing
 - ↳ Select technology (central vs distributed, DB vs files)
 - ↳ Index content to be defined
 - ↳ Event set queries to be defined for jobs
 - Optimizing random access through ROOT

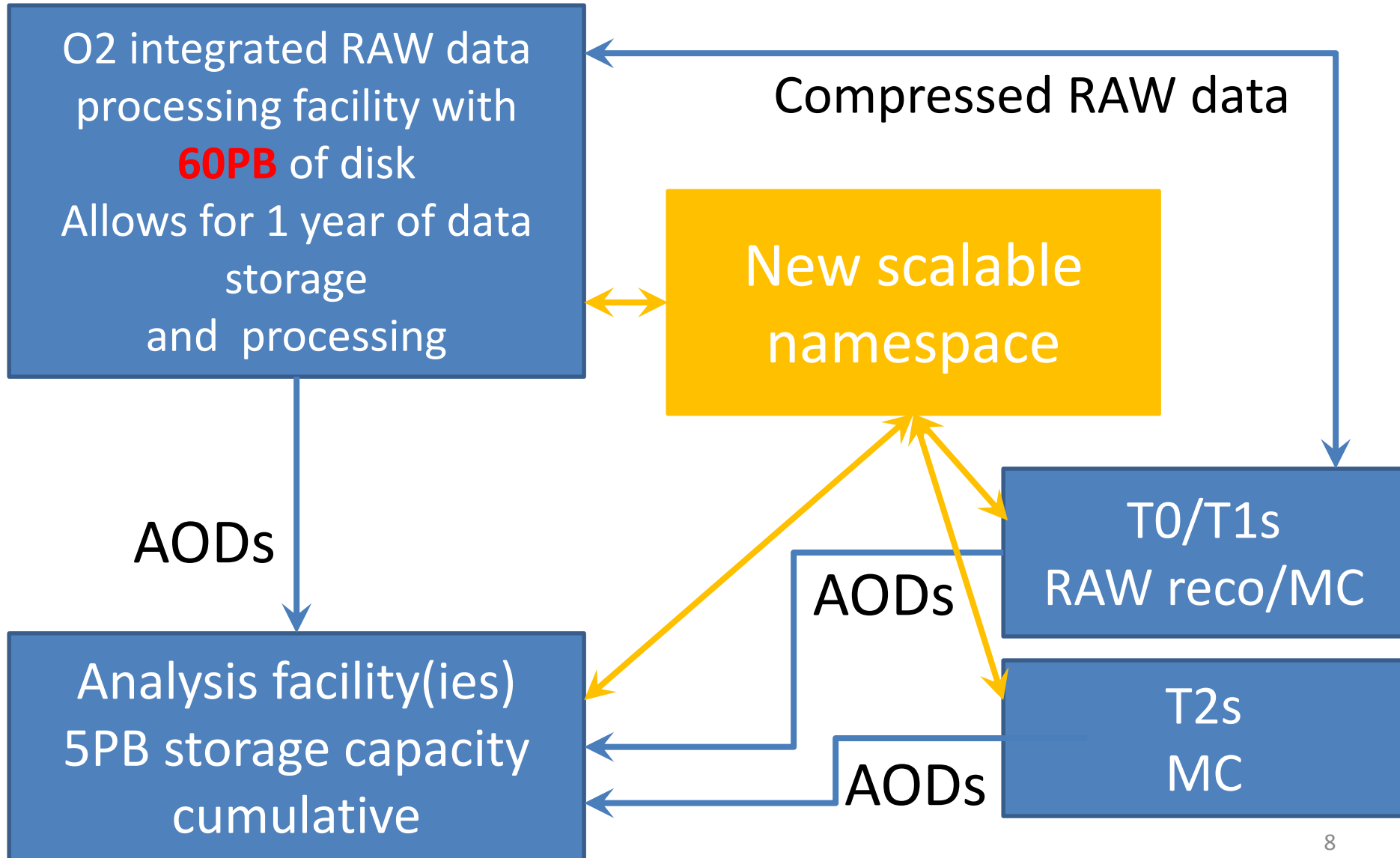


Analysis job using event index

LHCb medium term





ALICE upgrade TDR




CMS and ATLAS Computing Model medium term

RunII (and probably RunIII) are adiabatic changes with respect to current situation - this drives us to ~2023

- resources should stay within (or close to) the flat budget we are externally imposed to
- Changes in the analysis model can impact the resource needed, but for the moment no “miracle” to be expected:
 - CMS introduced recently the MiniAOD analysis data format, which is much smaller than previous AOD. Successful with caveats
 - ATLAS introduced recently the Train Analysis Model and the xAOD format. Successful with caveats



Storage Protocols: data access, data transfer, storage management



Protocols – Local Access

- Download to Worker Node (Production for ATLAS and LHCb)
 - xrdcp/SRM+gridftp (http/dccp)
- Direct access from storage
 - Anything supported by ROOT, whatever is more efficient
 - Mostly xrootd and posix (dcap still important)
 - Some work on http (especially for cloud e.g. AWS)
 - ATLAS and CMS delegate mostly to site
- Tendency to consolidate or at least no strong objection
 - As long as performance/efficiency is not heavily penalized

Protocols – Remote Access

- Remote data access does not imply a storage federation
 - Wait a couple of slide for storage federations
- Experiments optimized workflows and I/O for WAN access
 - Performance penalties now under control and acceptable for limited amount (order 10%) of WAN access
- Considerable fraction of experiments' I/O today is over WAN
 - 10% for ATLAS, 15% for CMS
- Xrootd is the solution used in production today
 - Strong wish to have it properly supported at all sites
- Some experience with HTTP from all experiments

Transfer Protocols (storage to storage)

- In principle, everything supported by FTS/GFAL2
 - SRM+gridftp, xrootd, http
- In practice, SRM+gridFTP
 - Supported at every T1/T2 in WLCG
- Can move to gridFTP only (no SRM) with caveats
 - Storage needs to support redirection properly or offer very “fat” gateway
 - No more “service class” for destination, rely on namespace
- http/xroot is possible but with more caveats
 - Dedicated FTS server to relay data, at least at some stage
 - Protocol “zoo” becomes a “zoo”**2
 - Performance? We know gridFTP since > 10 years and tuned it.

What is left of SRM?

- Needed for tapes
 - SRMBringOnline and polling of the request
- No need elsewhere, with caveats
 - See previous slide
 - TURLS need to be built by string manipulation
 - Some ancillary useful functionality (used/free space)
 - You need to deal with the legacy

□ No easy way to specify the service class destination in tURL

☆ dCache Tier1 sites where no disk/tape separation is done

☆ Possible solutions:

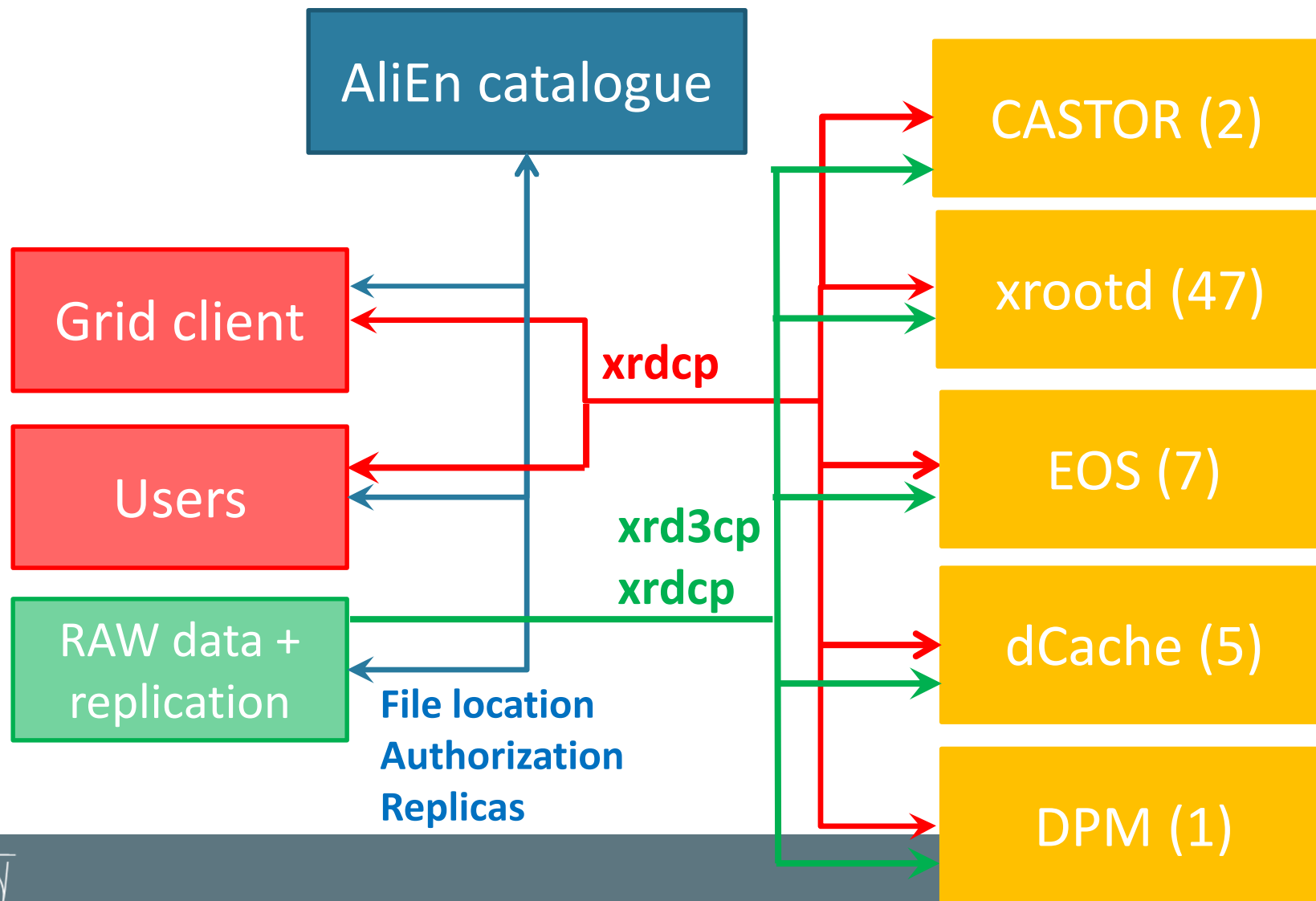
- ✱ Separate storage endpoints (gridftp) or dCache instances
- ✱ Namespace selection (already OK for StoRM @ CNAF)
 - Implies a rename of millions of files (or symlinks)
 - **Can sites help for this?**

Example from LHCb

Alice

- Single protocol for all clients/all storage solutions
 - Copy, streaming, partial access to files (important for analysis efficiency)
 - Transparent WAN/LAN file transfer and access
- Fine-grained monitoring (almost everywhere)
 - From client and storage elements
- Good understanding of internals
 - Tuned client, user-guided development and updates
 - Excellent support from developers and site operators
 - Realtime tracking of storage occupancy, cleanup and migration of data
- Comfortable situation for RUN2
 - Need for more disk space is a separate matter

Storage types, protocol and interactions



A possible medium term plan

- SRM: progress with decommissioning, apart for tapes
- Data access, upload, download:
 - Consolidate around the xrootd protocol (mainstream)
 - Progress with HTTP support, valuable both in the short and medium/long term
- Data Transfer
 - Investigate possible alternatives to gridFTP (e.g. xrootd like Alice, HTTP)
 - Do not forget that data deletion is as challenging as data transfer

Storage types and technologies

- Today's storages
 - dCache, DPM most broadly deployed
 - Castor and EOS on a few sites
 - pure xrootd or gridftp in US (with bestman on top for srm)
 - ceph@RAL under testing
- Object Stores: interesting solution for most use cases and experiments are gaining experience
 - CEPH used by ATLAS for log files and event service outputs, by CMS for CMS@HOME
 - Amazon S3 used in US for production activities on AWS
- Object Store integration:
 - Efficient 3rd party transfer needs to be sorted out
 - Access protocols: s3/http is ok for most uses
 - Authentication is via keys, different possibilities
- Can they can fully replace a large site's storage?
 - But they look very appealing, especially if mostly of what you do is put/get and you do not care about a namespace

Read-Only Storage Federations

- AAA and FAX in CMS/ATLAS
 - Xrootd based federation
- LHCb implemented a xrootd-based redirector-less federation
 - Using Gaudi+Dirac File Catalog
 - ATLAS looking into this as well
- Xrootd-based federations used today in production
 - Overlay the existing storages
- HTTP federations: some experience, possible future option, no concrete plan

Federated storage – today

(input most from Alice, shared by many)

- dCache@NDGF is a clear, long established example
 - ToDo - file location related to CE location
 - Federation possible due to excellent network, collaboration between sites, strong expert support
 - RTT penalty is still there (important for analysis)
- Other dCache federated examples e.g. in US T2s
- EOS federated storage demonstrated to work
 - Same issue (file location) to be solved
 - Expect any federated storage to have this sorted out
 - Or there will be no gain in performance (RTT penalty)

Organization of Sites - CMS

(less individual sites, more federations)

- Experiment and site overhead could be reduced by “federating” sites, but only if it’s transparent
 - Thanks to work done for WAN access, most CMS workflows could run equally well on a federated site
 - The problem is the exceptions - even between Meyrin-Wigner we need special treatment of file merging jobs (enable LazyDownload)
- Nevertheless we’d be happy if we didn’t need to deal with individual sites smaller than X

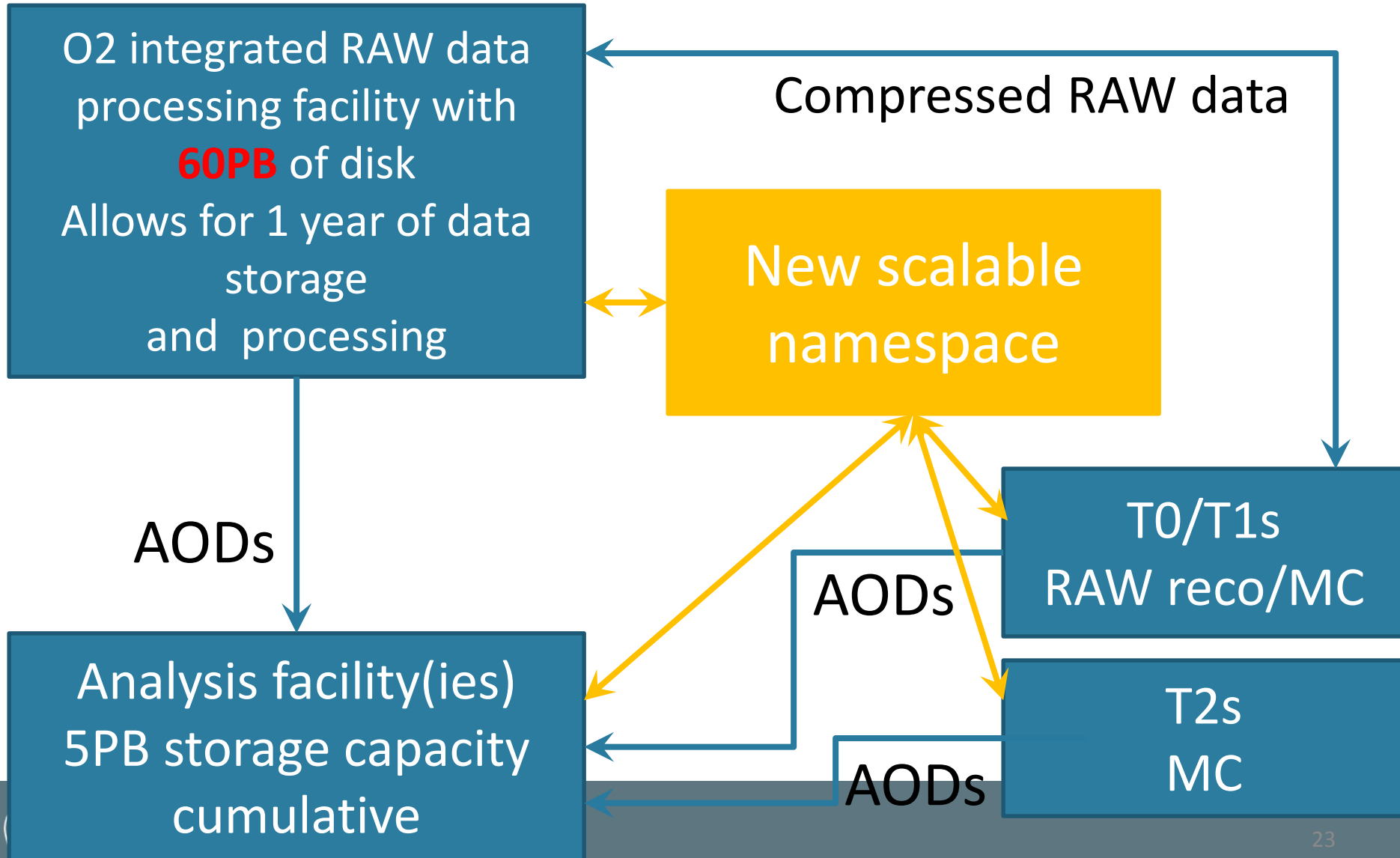
Storage Consolidation – ATLAS

- Fewer sites are better from operational point of view, while manpower always very difficult to quantify!
- Possible evolution can be in two directions:
 - a) Funding for storage consolidates in fewer sites, smaller sites (e.g. <400TB) do not invest more in disk but rather on CPU. They become disk-less or cache-only
 - b) Deploy or move to distributed storages per region/country/funding body, see previous slide

Both solutions will require work

- Different Funding Model
- Technical solutions (caches vs direct access, cache technologies, storage implementations not suitable to be distributed)

Back to Alice upgrade TDR



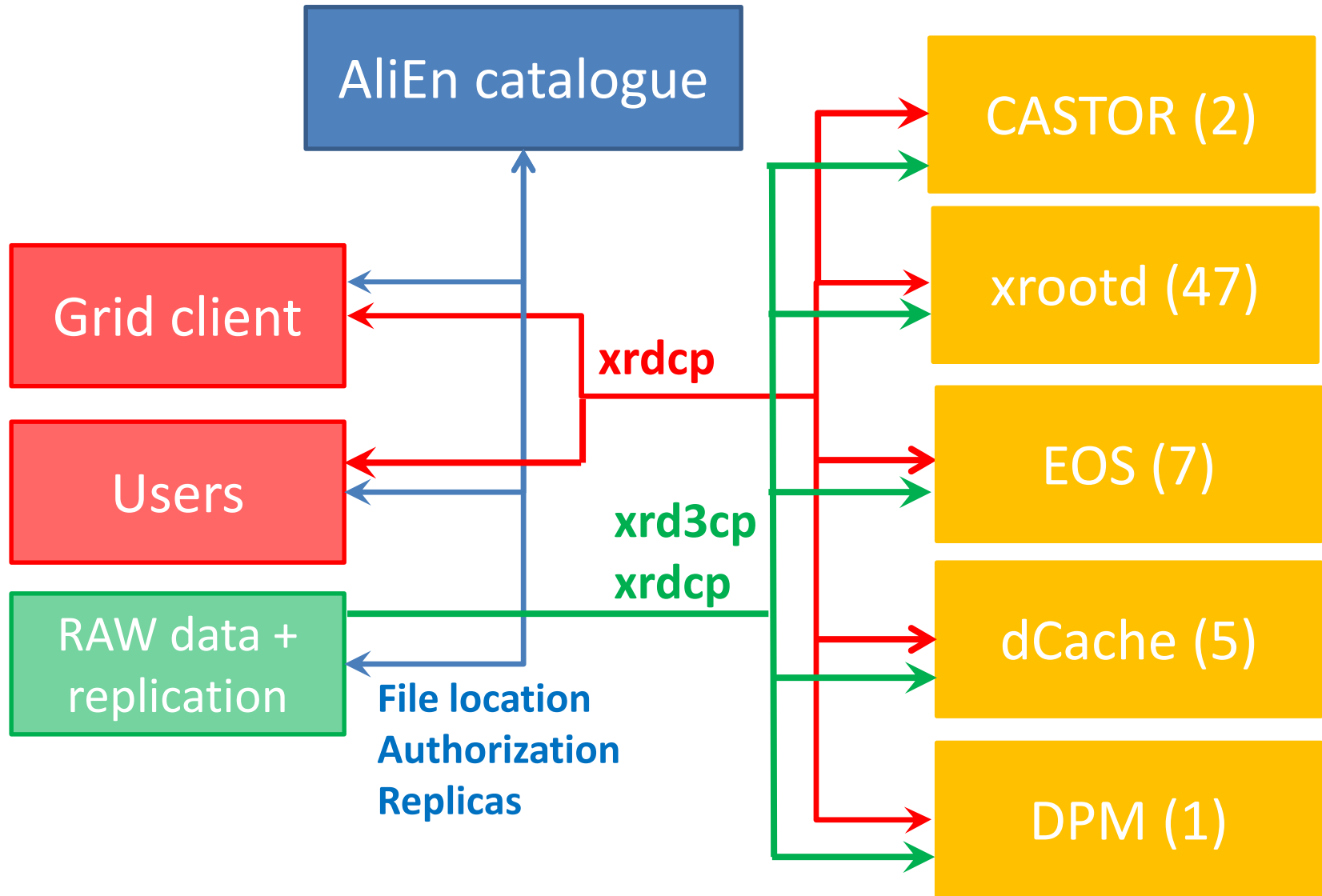
Storage Consolidation is border line
between medium and long term planning

Discuss more tomorrow

Backup: the original inputs from experiments

Alice

Storage types, protocol and interactions



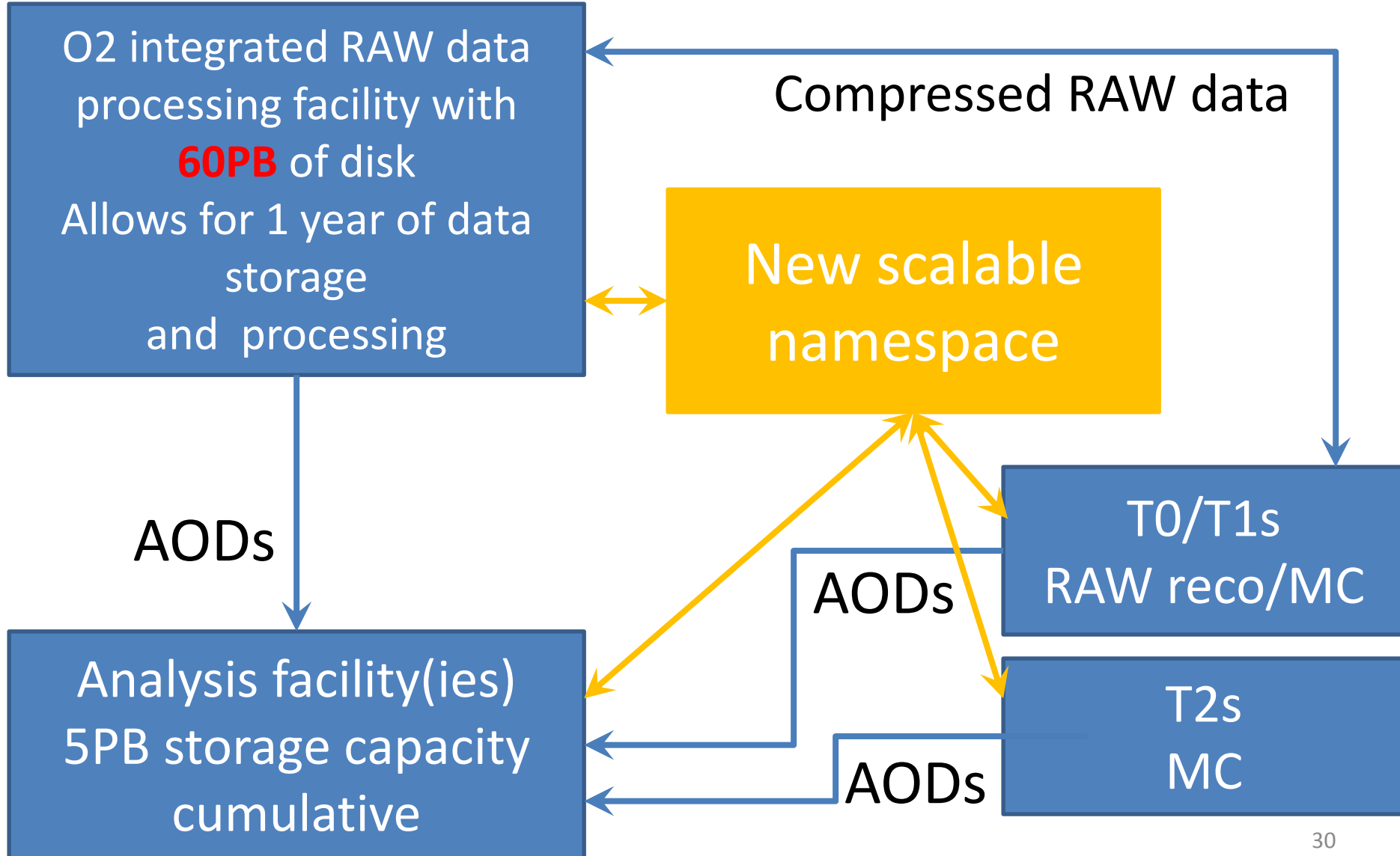
Status now

- Single protocol for all clients/all storage solutions
 - Copy, streaming, partial access to files (important for analysis efficiency)
 - Transparent WAN/LAN file transfer and access
- Fine-grained monitoring (almost everywhere)
 - From client and storage elements
- Good understanding of internals
 - Tuned client, user-guided development and updates
 - Excellent support from developers and site operators
 - Realtime tracking of storage occupancy, cleanup and migration of data
- Comfortable situation for RUN2
 - Need for more disk space is a separate matter

Federated storage – today

- dCache@NDGF is a clear, long established example
 - ToDo - file location related to CE location
 - Federation possible due to excellent network, collaboration between sites, strong expert support
 - RTT penalty is still there (important for analysis)
- EOS federated storage demonstrated to work
 - Same issue (file location) to be solved
 - Expect any federated storage to have this sorted out
 - Or there will be no gain in performance (RTT penalty)

ALICE upgrade TDR



2016 WLCG Workshop - CMS input on storage for the medium term

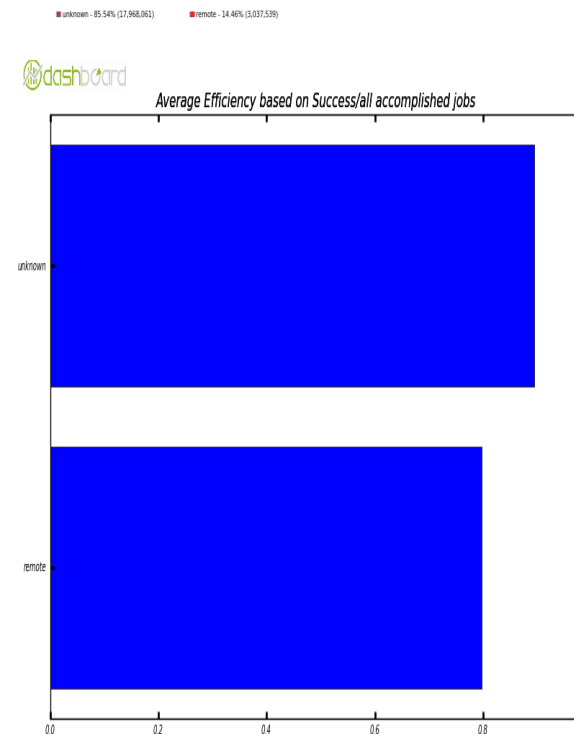
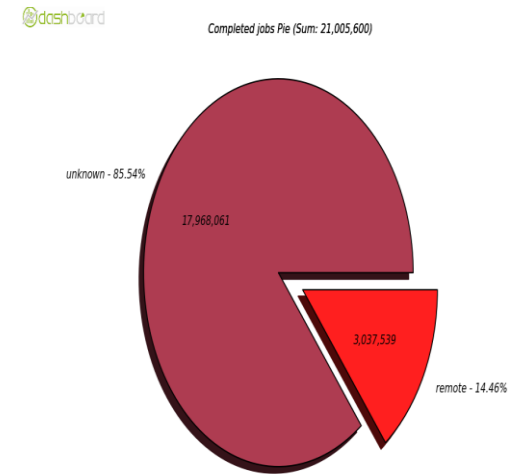
Nicolò Magini, Christoph Wissing

Protocols

- Local access:
 - Reminder: CMS delegates to sites the choice of local protocol
 - “Whatever is supported by ROOT” - in practice
 - Currently root or posix; dcap also still widely deployed at sites but we wouldn’t object at phasing it out like we (almost) did for RFIO
 - Started to gain experience with http in AWS project
- Remote access:
 - Will remain xrootd in the medium term
 - Testing of http based federation is an option
- Transfers:
 - In principle: “Whatever is supported by gfal2/FTS3”
 - srm, gsiftp, root, http, ...
 - In practice: in the medium term, we are OK with phasing out srm for disk but we need interoperability with srm, i.e.
 - gsiftp - OK, we already have some gsiftp endpoints in PhEDEx, e.g. eoscms
 - root, http - OK, as long as the storage also has srm/gsiftp as fallback protocol OR the site uses a “dedicated” FTS3 server to relay data to srms

WAN Access

- Spent a lot of effort to improve application efficiency for WAN access
- Fraction of jobs (~15%) routinely access data over WAN
- Most workflows suited for WAN access
 - Very reasonable performance penalties
 - Present exceptions:
 - MC-DIGI-RECO 1/MB/s/core for large pile-up
 - Pre-mixing (under validation) 0.01MB/s/core
 - Merge jobs
- WAN access via Federation based on xrootd
 - Fully deployed and expected to be operated over next few years
 - HTTP based federation considered as future option



Organization of Sites

(less individual sites, more federations)

- Experiment and site overhead could be reduced by “federating” sites, but only if it’s transparent
 - Thanks to work done for WAN access, most CMS workflows could run equally well on a federated site
 - The problem is the exceptions - even between Meyrin-Wigner we need special treatment of file merging jobs (enable LazyDownload)
- Nevertheless we’d be happy if we didn’t need to deal with individual sites smaller than X

Adaption to new Technology Trends (e.g. object stores)

- Started to gain experience using object stores as “traditional” Storage Element.
- Transfer files between grid SEs and OSs (using FTS3)
- Jobs reading input files from OSs and/or straging output back
 - Examples: S3 during AWS project, Ceph during CMS@home project

Funding situation medium term

For CMS, RunII (and probably RunIII) are adiabatic changes with respect to current situation - this drives us to ~2023

- resources should stay within (or close to) the flat budget we are externally imposed to
- Changes in the analysis model can impact the resource needed, but for the moment no “miracle” to be expected:
 - CMS introduced recently the MiniAOD analysis data format, which is much smaller than previous AOD, but
 - more copies needed, and in more versions (one of the drivers for MiniAOD is indeed the reprocessing capabilities CMS had to abandon on larger data formats)
 - Not ok for all the analysis (target ~ 75%), so we cannot get rid of adequate access also to AOD
 - All in all, initially probably even a small resource increase ...

ATLAS

Cameron/Di Girolamo/Filipic

Sites and consolidation

- Fewer sites are better from operational point of view
 - But manpower always very difficult to quantify!
- Our recommendation:
 - Small (<400TB) T2s do not invest more in disk
- Prefer aggregation or federation per region/country/funding body
- Federation technologies:
 - dcache and xrootd proven for many years (eg NDGF, US)
 - http federation not (yet) used heavily
- What to do with new small sites?
 - Could be “storage-less” from ATLAS point of view
 - possible to select particular workflows (e.g. low I/O) if needed
 - Use cache or federation or remote storage

Summary: We foresee a split between a few “large” sites and many small cache or federated sites

Storage types and technologies

- “Classic”: dCache, DPM are used almost everywhere
 - Castor and EOS on a few sites
- Others: pure xrootd or gridftp in US (with bestman on top for srm), ceph@RAL under testing
- Object stores are used more and more
 - ATLAS currently uses them for extra log copy and event service
 - 3rd party transfer is a problem (for now) but only if we need it
 - Access protocols: s3/http is ok for most uses
 - Authentication is via keys, ATLAS services provide link to grid certificates
 - Still to be shown that they can fully replace a large site’s storage
- Caches: e.g. ARC, xrootd
 - Work well, details are hidden from ATLAS
 - Concepts of volatile storage and object stores are built into Rucio

Summary: We expect the classic storage technologies to stay but foresee object stores taking over some workflows/sites

Storage usage and protocols

- All atlas data has a lifetime (extended if data is used)
- We put everything we can (not small files or short-lived data) on tape
- Disk is managed as a cache, expired or unused data is removed if space is needed
- **Protocols:**
 - srm/gridftp still required to be fully part of atlas as of today
 - Missing: space reporting, legacy ATLAS code (still srm hardcoded here and there)
 - Details by activity as of today:
 - Data staging from tape: srm is the only option
 - Data transfer: gridftp is the only option
 - Data access: want to move to copy to scratch for production (gfal-copy whatever://), direct read for analysis (xrootd or http)
- **Summary:** gridftp is minimum requirement for any site, http or xrootd if site runs analysis, srm for tape. No real change foreseen in medium-term
 - Present ATLAS tools are flexible enough to integrate/change new protocols if needed, but work is needed (e.g. webdav is not really identical to S3)!

General Considerations

- ATLAS is trying to push hard on integrating/evaluating new technologies, new protocols, new concept of e.g. Federations
- We need to plan the evaluations carefully (not too many, limited time, clear goals)
- We need to be able to move away from legacy stuff
 - the problem is that it is often “easier” not to touch things that work!
 - data taking, conferences, special events restrict when major transitions can happen



DM medium term plans



- Run2 conditions:
 - 12.5 ++ kHz of HLT output rate
 - Small pileup ($\mu=1.1$)
 - Throughput just below 1 GB/s
 - ↳ In 2015: up to 1.3 GB/s
 - Online reconstruction = offline reconstruction
 - ↳ Allows direct analysis from online data (TURBO stream)
- LHCb major upgrade is for Run3 (2020 horizon!)
 - Luminosity $\times 5$ ($2 \cdot 10^{33}$)
 - Trigger rate... $\times 5$ (at least)
 - Throughput between 6 and 10 GB/s!
 - Trigger (SW only) = offline selection
 - ↳ Stripping is no longer effective (all events are for physics!)
 - ↳ Possibly directly export reconstructed data only



- Production jobs
 - Mostly using download to WN (using gridftp, could be replaced with xrdcp or http)
- Direct protocol access
 - User jobs and analysis productions
 - Use xroot, unless other protocol proven to be more efficient
 - ↳ file: used locally when available
 - ↳ All sites must provide xroot access over the WAN
 - ⚠ Few issues sometimes with xroot end nodes not in firewall
 - Data Federation (through Gaudi + FC)
 - ↳ No use of redirector, only use replica catalog
 - ⚠ Try local replica first, then randomly remote replicas
 - Use of http aggregation and http access
 - ↳ For the time being, work ongoing for FC/SE consistency checks
 - ↳ No plan to move to using it in jobs unless more performant
- SRM for getting xroot/file: tURLs for data access
 - Should disappear soon (build tURL by string manipulation)
 - ↳ Implies that xroot endpoints are known and stable (one per SE)!



Data transfers (FTS3 and data upload)

LHCb medium term

- SRM still mandatory for tape bringOnline!
- Service class selection for destination
 - No easy way to specify the service class destination in tURL
 - ↳ dCache Tier1 sites where no disk/tape separation is done
 - ↳ Possible solutions:
 - ⌞ Separate storage endpoints (gridftp) or dCache instances
 - ⌞ Namespace selection (already OK for StoRM @ CNAF)
 - Implies a rename of millions of files (or symlinks)
 - **Can sites help for this?**
 - Caveat of building gridftp URLs:
 - ↳ Possible performance issue for gridftp unless redirection in place
 - ⌞ On Castor and EOS for example (not possible to use disk servers gridftp)
 - Medium term plans
 - ↳ Source replica (FTS3):
 - ⌞ Use built gridftp URL (see caveat above)
 - ↳ Destination replica:
 - ⌞ Use built gridftp URL if separate service classes
 - ⌞ Use SRM URL for dCache sites



- Reduce number of user jobs on the Grid
 - Centralise ntuple / μ DST creation as “train analysis” (c.f. ALICE)
- Using indices for analysis
 - Replace “stripping + streaming” with “selection + indexation”
 - ↳ Because stripping retention will be high (more selective trigger)
 - Event set query to central (or local) index
 - ↳ Download a local event collection (i.e. direct access addresses)
 - Random access to local or remote data
 - ↳ Using a local replica catalog (Gaudi Federation)
- R&D can start now (2016/17) for:
 - Setting up train analyses
 - ↳ framework similar to stripping
 - Data indexing
 - ↳ Select technology (central vs distributed, DB vs files)
 - ↳ Index content to be defined
 - ↳ Event set queries to be defined for jobs
 - Optimizing random access through ROOT



Analysis job using event index

LHCb medium term

