Grid data access on widely distributed worker nodes

(using Xrootd/Scalla and SRM)

Pavel Jakl[2] Jérôme Lauret[1] Andrew Hanushevsky[4]
Arie Shoshani[5] Alex Sim[5] Junmin Gu[5]

[1]Brookhaven National Laboratory, United states of America

[2]Nuclear Physics Institute, Academy of Science, Czech Republic

[4]Stanford Linear Accelerator Center, United states of America

[5]Lawrence Berkeley National Laboratory, United states of America

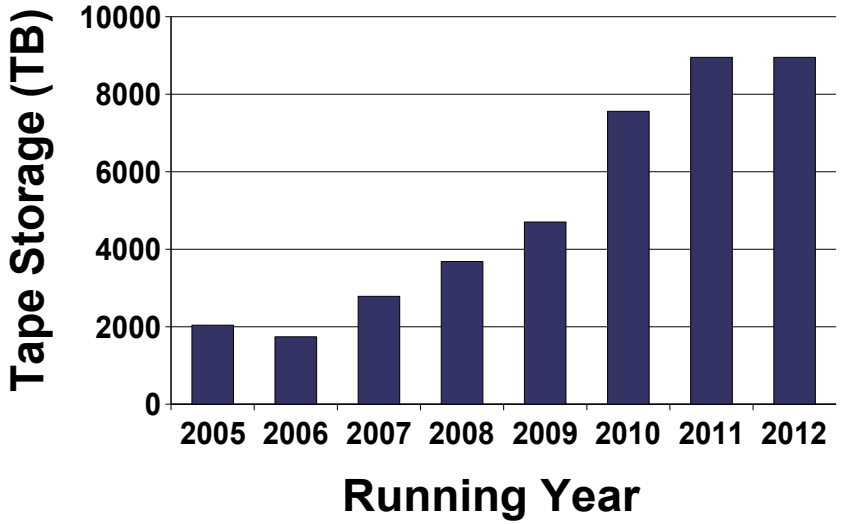
BROOKHAVEN NATIONAL LABORATORY

CHEP 2007
Victoria, Canada

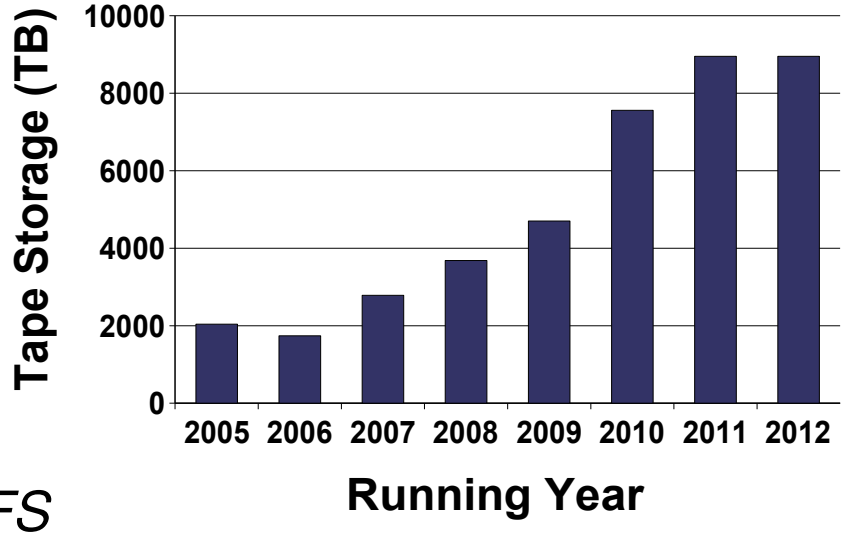
STAR

Outline

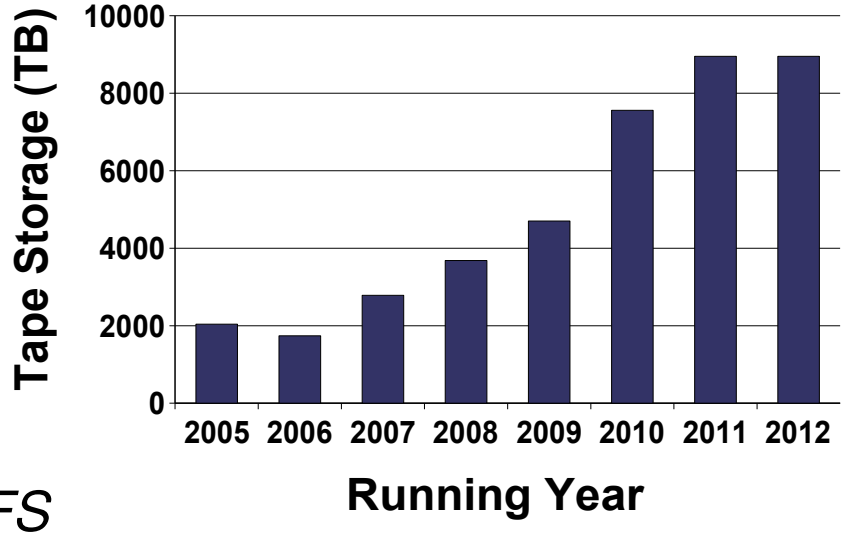
- over 1PB data per year at STAR



- over 1PB data per year at STAR
- *Permanent* location:
  - tape system (HPSS): offers several **PBs**
- *Temporary* locations:
  - centralized disk space: **75 TB** *via NFS*
  - distributed disk space: **350 TB** *spread over 1000 nodes*

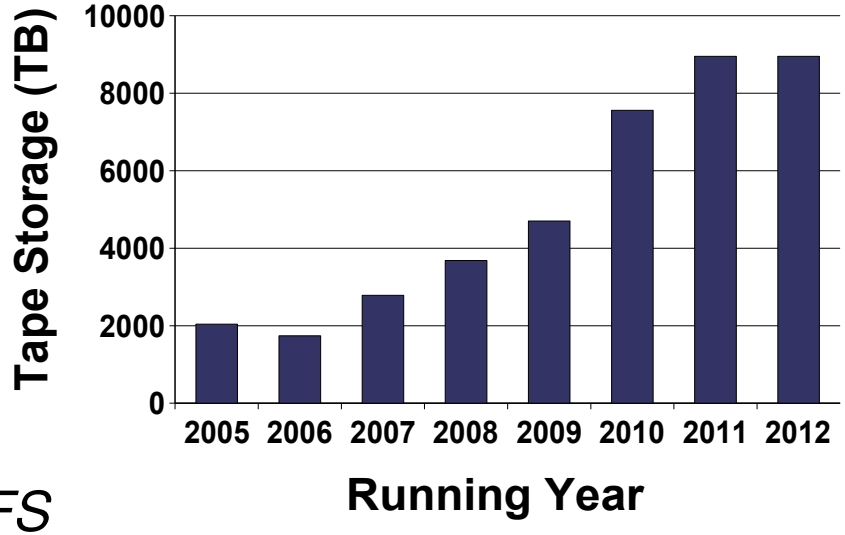


- over 1PB data per year at STAR
- *Permanent* location:
  - tape system (HPSS): offers several **PBs**
- *Temporary* locations:
  - centralized disk space: **75 TB** *via NFS*
  - distributed disk space: **350 TB** *spread over 1000 nodes*

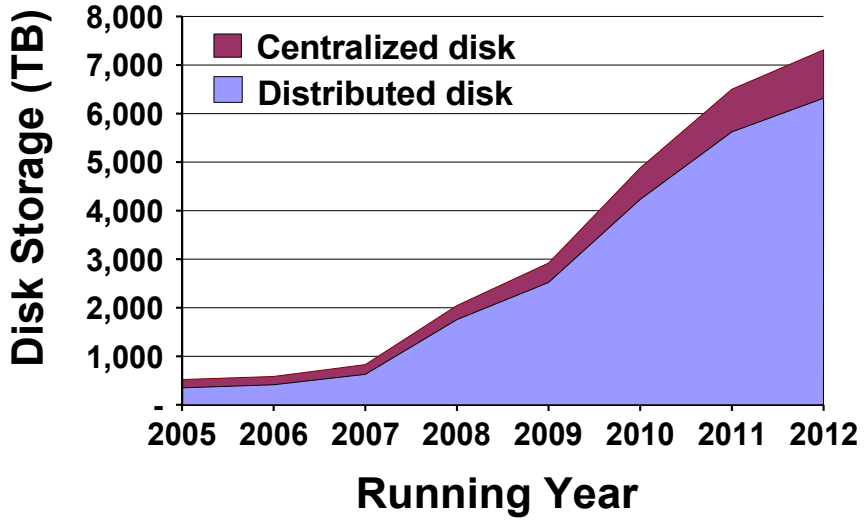


- **distributed** vs centralized disk:
  - ⊕ very low cost (factor of ∼10)
  - ⊕ less human resources to maintain
  - ⊖ worse manageability (one has to build aggregation)
  - ⊖ no native OS/system provides scalable/workable solution

- over 1PB data per year at STAR
- *Permanent* location:
  - tape system (HPSS): offers several **PBs**
- *Temporary* locations:
  - centralized disk space: **75 TB** *via NFS*
  - distributed disk space: **350 TB** *spread over 1000 nodes*



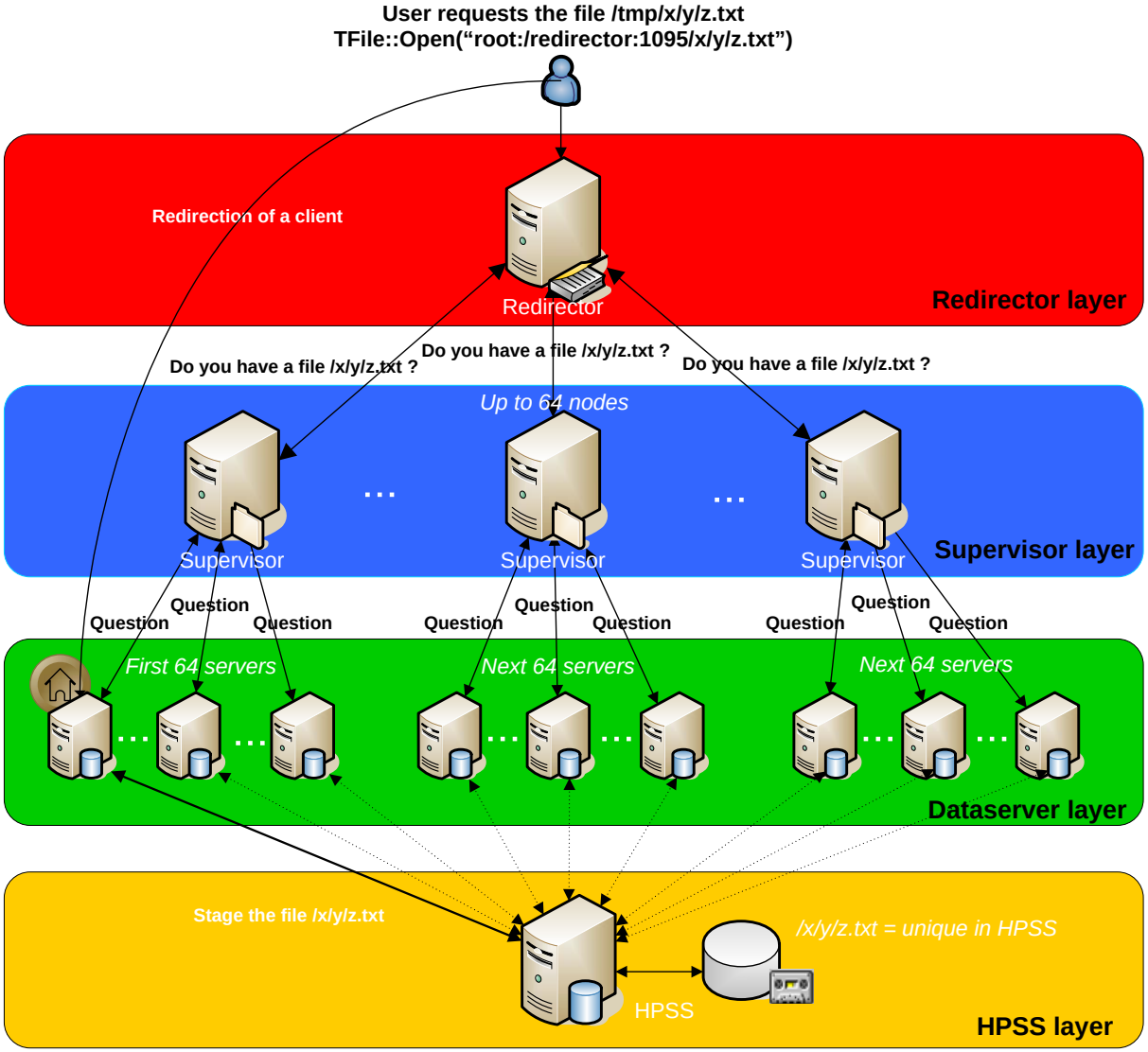
- **distributed** vs centralized disk:
  - ⊕ very low cost (factor of ∼10)
  - ⊕ less human resources to maintain
  - ⊖ worse manageability (one has to build aggregation)
  - ⊖ no native OS/system provides scalable/workable solution



What are the required features?

- distributed system providing high performance file-based access
- main goals:
  - Scalability - can serve thousands of clients
  - Reliability - should eliminate single point of failure (SPOF)
  - Fault-tolerant - adaptation to server crash or missing data
  - Flexible security - allowing to run any security protocol
  - Load balancing - sharing the load among multiple servers
  - MSS integration - accessing files from permanent storage (such as HPSS)
  - Single global unique name-space - span single name-space across multiple servers
  - Replica management - determination of the location and multiplicity of data
  - Grid integration - Consistent data management strategy: possibly talk to other DM tools, local or distributed on Grid
- solution: Scalla (Structured Cluster Architecture for Low Latency Access) aka Xrootd

Quick Scalla architecture overview at STAR



Each CE hosts SE = sharing of resource

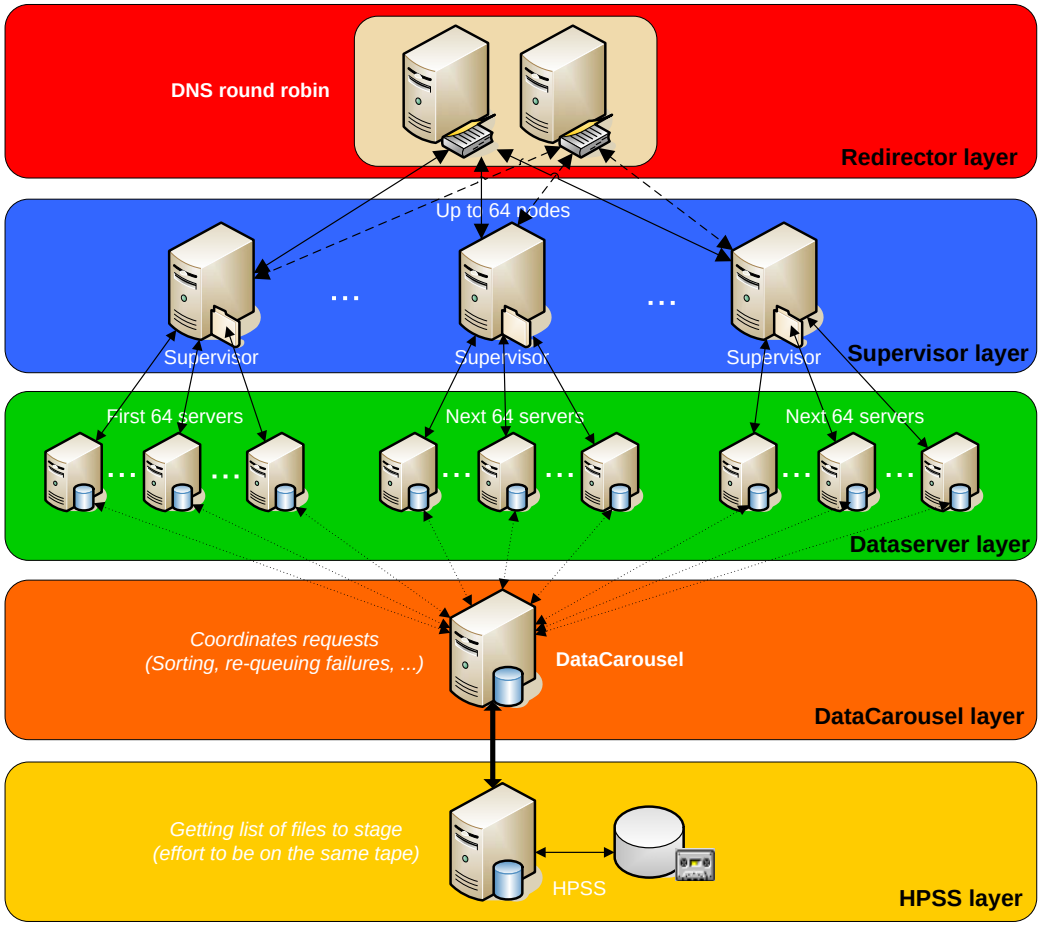
Outline

Introducing coordination solution

- coordination issues within MSS access in large deployment of Scalla was presented at last CHEP conference[1]
- an interimn solution called **DataCarousel** has been introduced with main features as:
  - coordinating and queuing requests
  - sorting of requests according to their location on tapes
  - sharing access with other data management tools involving policy based authorization with different priorities per user or group
  - keeping track of requests and re-queuing them in case of failure
  - advance techniques as: request expiration time, cancellation of request etc.
- **Question**: How this system was integrated into Scalla/Xrootd architecture ?

[1] P. Jakl *et al.*, "From rootd to Xrootd, from physical to logical files: experience on accessing and managing distributed data", *in Proc. of CHEP 2006*

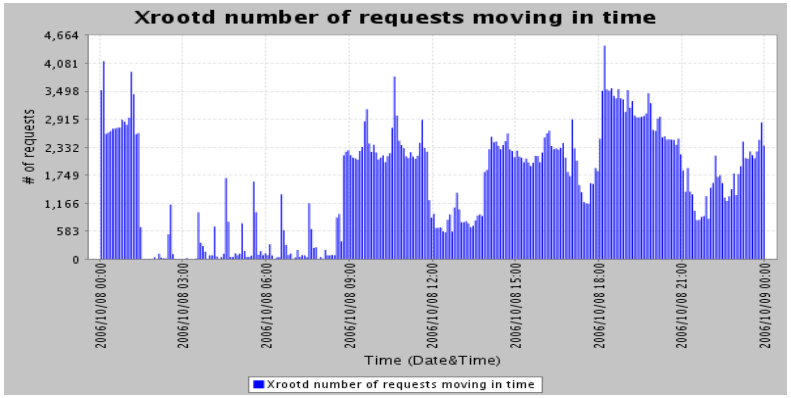
Scalla with HPSS request coordination

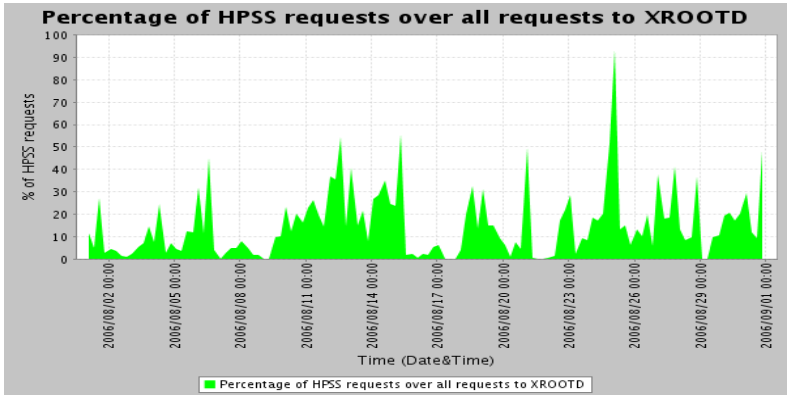


- ***How this widely distributed system is being monitored ?***

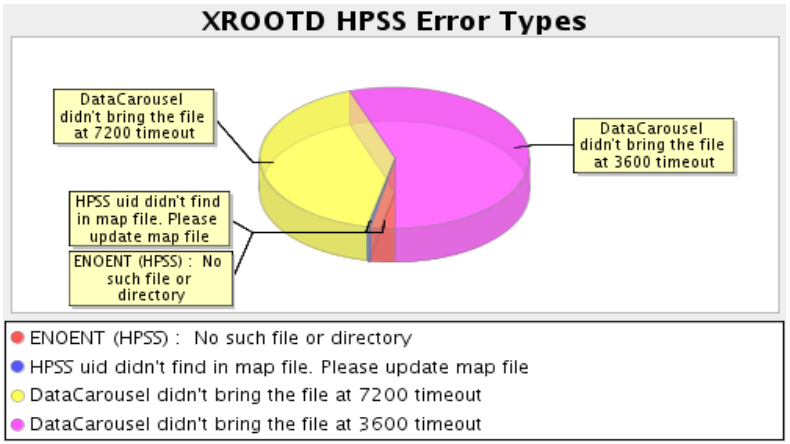
Scalla in production and real analysis scenario

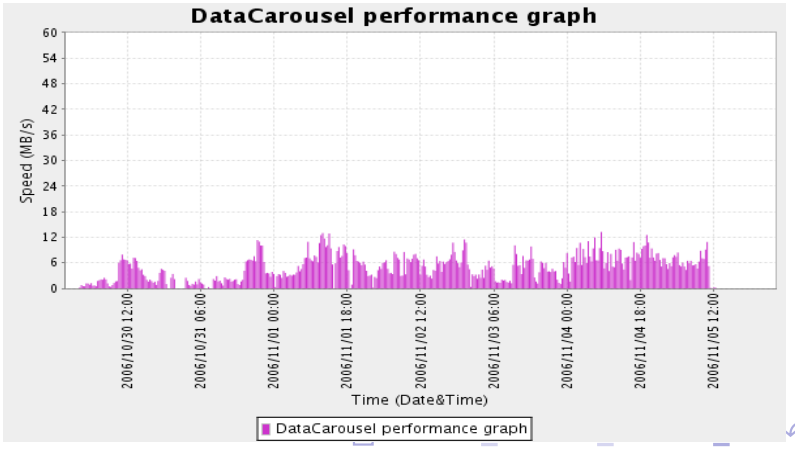
- possible to see up to 35 requests/sec to open files, users use Scalla to access HPSS data-sets





- most of errors are caused by timeouts $\Rightarrow$ slow performance per a HPSS tape drive (14 at STAR) $\Rightarrow$ high latencies





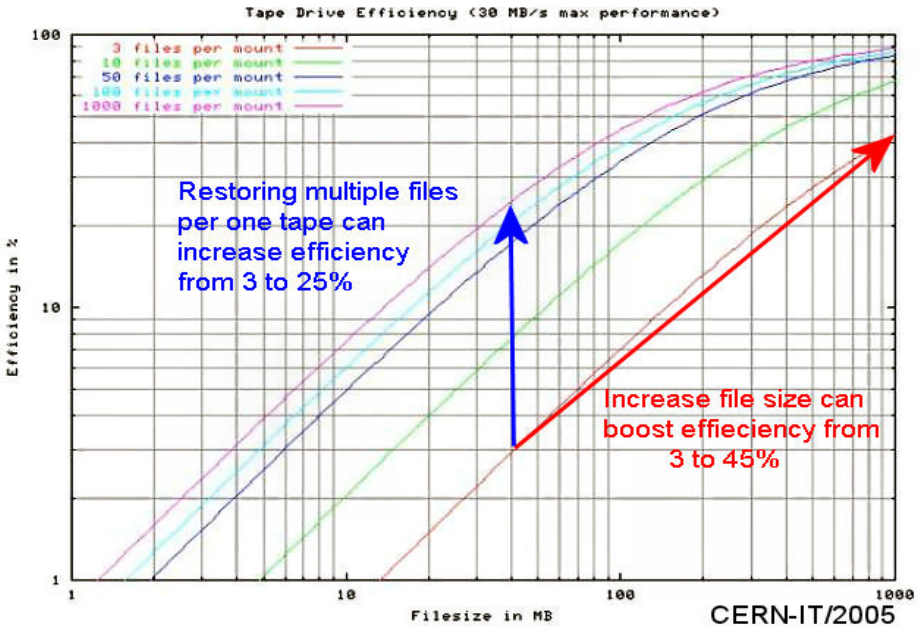
Analysis scenario (Facts at beginning)

- users defines their jobs(=analysis) using job's description of SUMS (STAR Unified Meta Scheduler)
  - SUMS resolves their meta-data query (energy, collision etc.) into particular physical data-sets by handshaking with STAR FileCatalog
  - SUMS orders data-sets, splits them into specific sub-jobs and submits into batch system queue (~100-1000 files per one sub-job)
- STAR files are compressed and structured ROOT files (events sorted into tree structure)

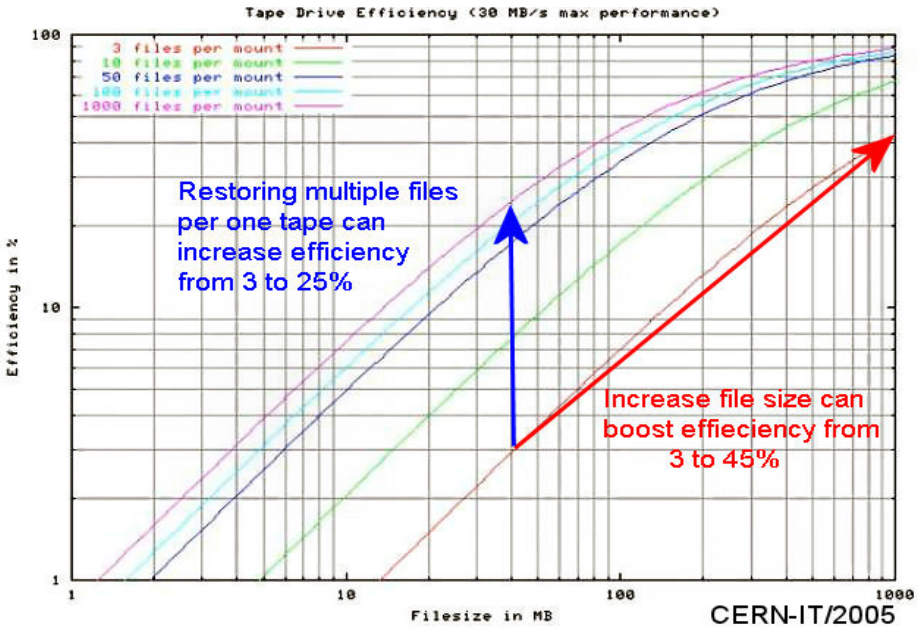
	avg size of file	file's description
an event file	~284 MB	same size as daq files
an MuDST file	~88 MB	mostly used for analysis

- one file could be restored in average of 24 minutes
- drawback for user's side:
  - job hangs waiting for a file to be restored from HPSS and eventually is killed by batch system's wall clock (e.g 24*1000=400 hours)

Reasons for slow HPSS performance

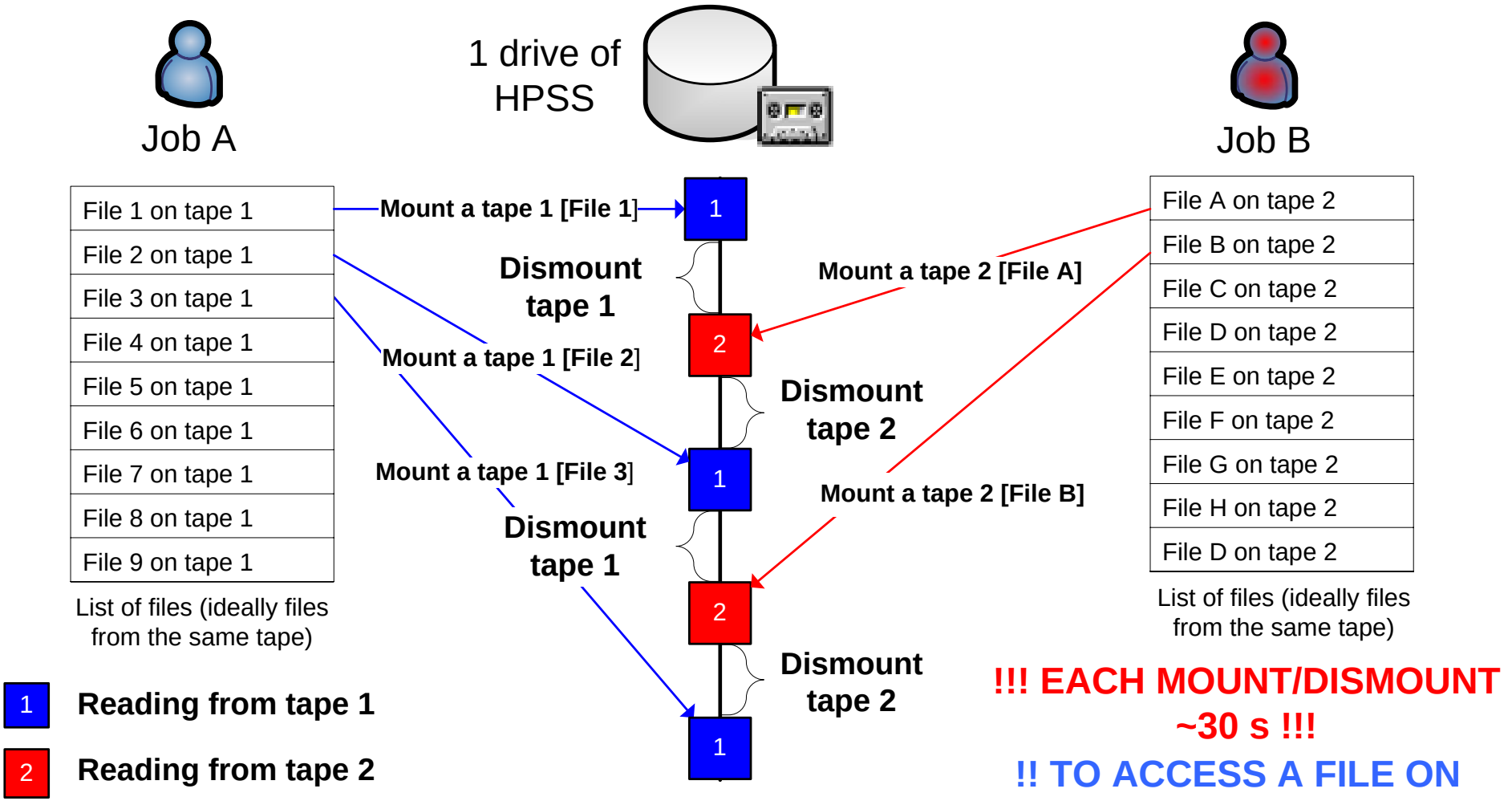


Reasons for slow HPSS performance

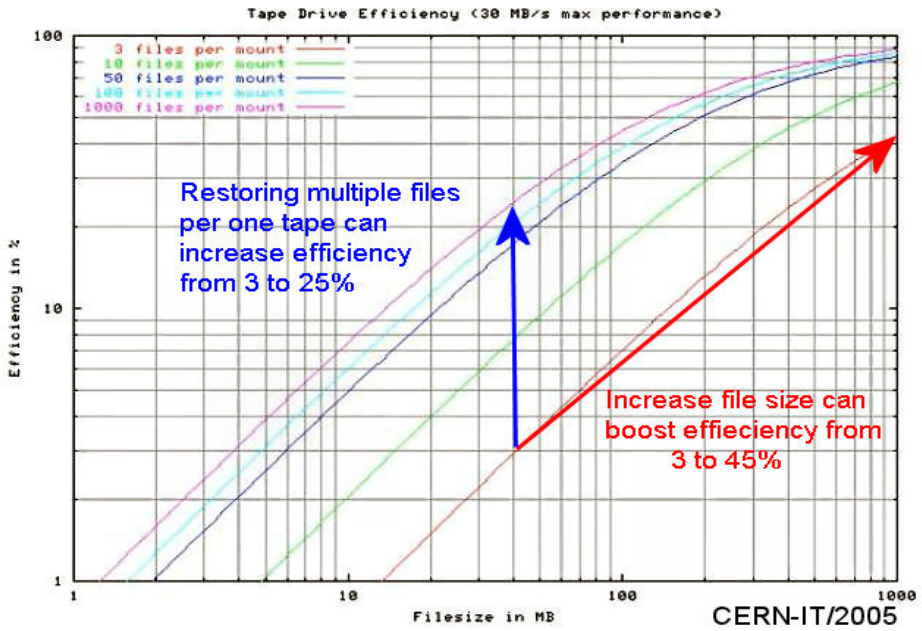


1. the performance can be boosted by optimizing the size of a file (currently ∼90MB)
2. sequential processing of files causes excessive mounting of tapes

Impact of sequential processing on a HPSS drive



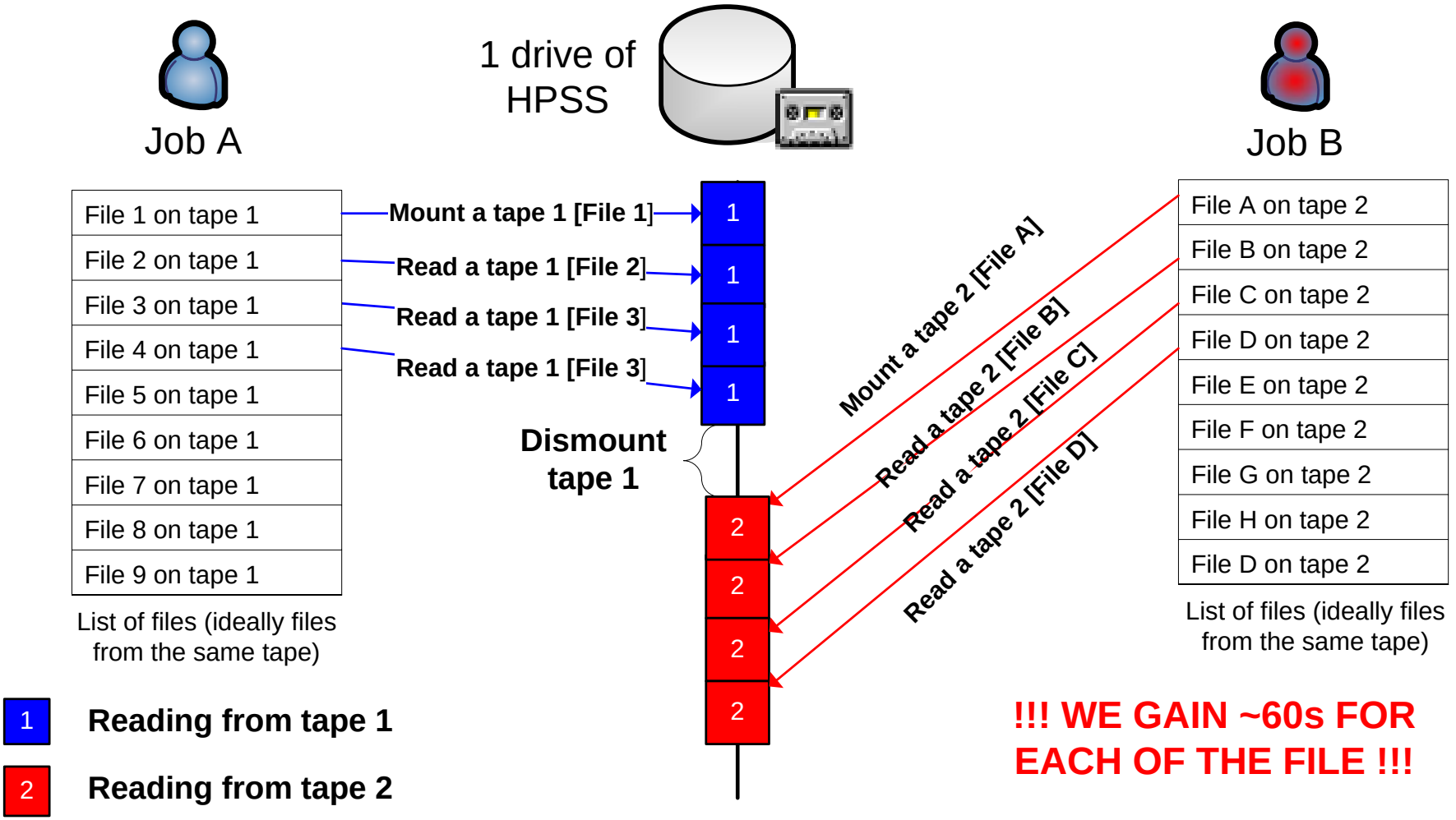
Reasons for slow HPSS performance (cont.)



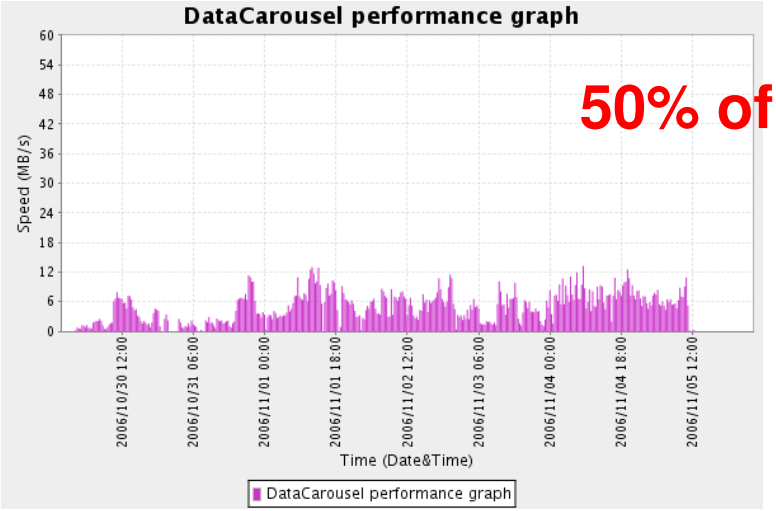
1. the performance can be boosted by optimizing the size of a file (currently ∼90MB)
2. sequential processing of files causes excessive mounting of tapes

- DataCarousel already does a sorting of requests per tape, but not enough efficient in case of sequential processing
- we need bigger list for sorting ⇒ Pre-staging of files
  - job publishes its whole intend for processing ⇒ all files to be processed in near future
  - files from the same tape will end up in DataCarousel system ⇒ bigger list for selection

Influence of the pre-staging on a HPSS drive



Did the pre-staging help ?



50% of improvement

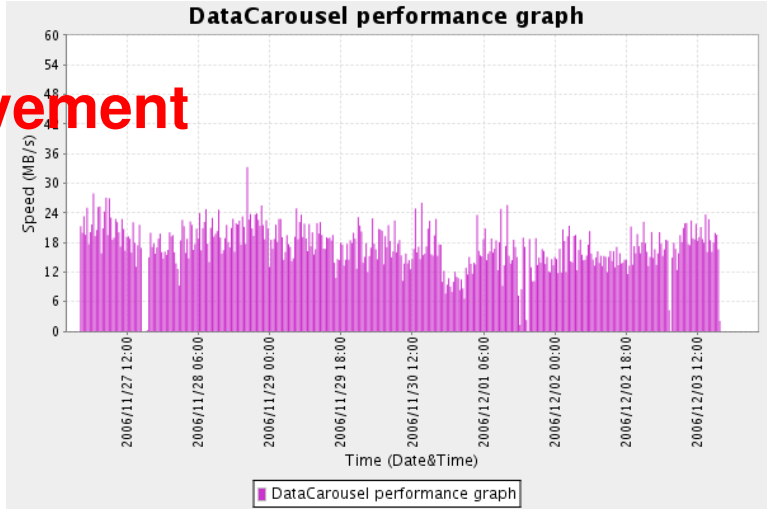
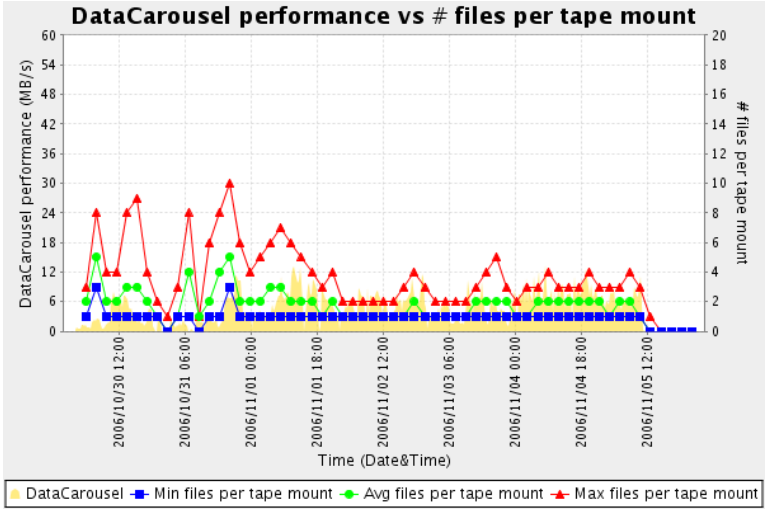
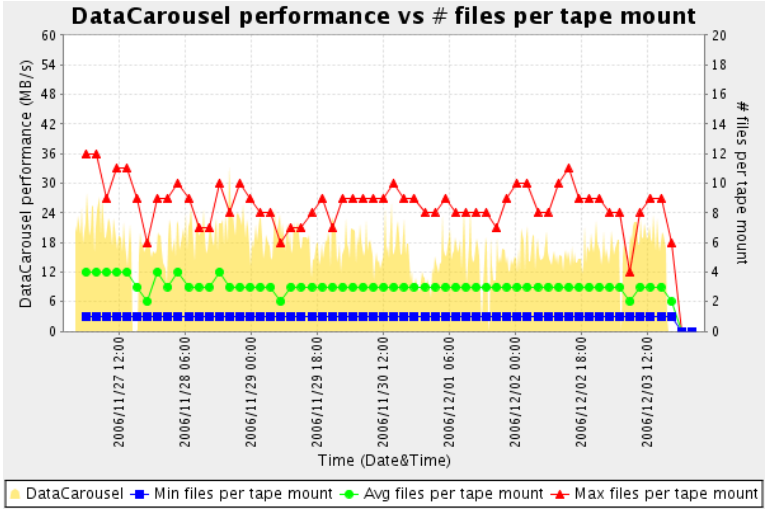


Figure: Before

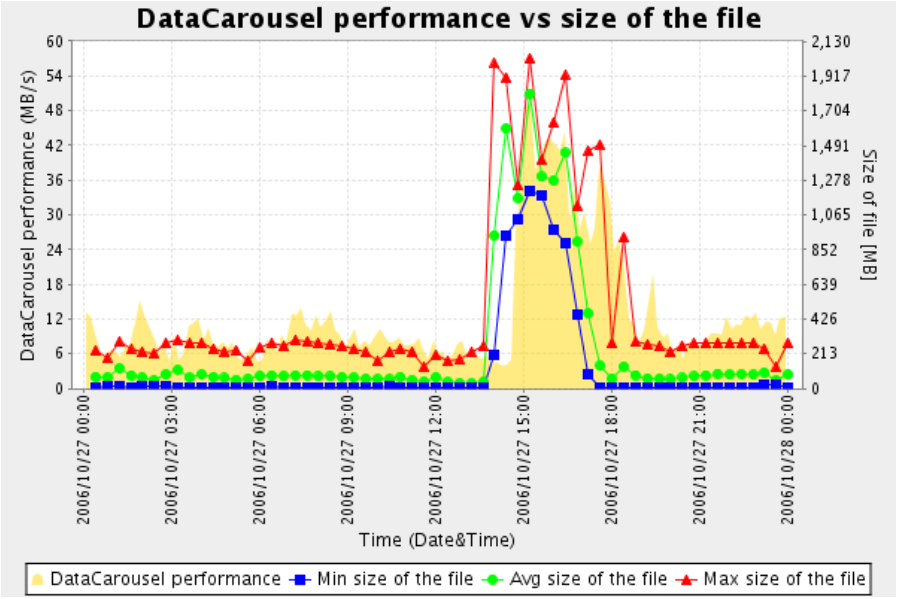
Figure: After





File size impact on HPSS performance

- size of file has the biggest impact on HPSS performance
- we can boost efficiency up to 45% when increasing the current file size 10x times
- peak is showing test with STAR DAQ files (10 times bigger than MuDST files)



- plan to merge our MuDST files (all files with the same "RUN" number together)

Outline

Motivation . . .

- **Scalla** could be extended:
  - does not bring files over from other space management systems (dCache, Castor etc.)
  - always bring files from MSS, not from cache of neighboring node
  - has no advanced reservation of space, no extended policies per users or role based
  - no guarantee for stored files (no lifetime, no pinning of files)
  - only access files (what about event-based access ?)
- other middle-ware is designed for disk cache management

Leveraging on other projects and targeted re-usable components ?

SRM functionality

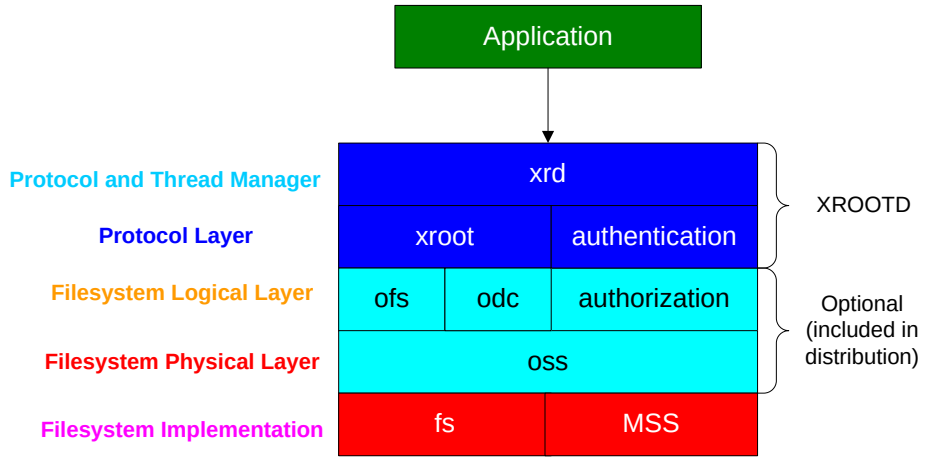
- **Storage managers:** the grid middle-ware component whose function is to provide dynamic space allocation and file management on shared distributed storage systems
  - Manage **space**
    - Negotiate and assign space to users and manage *lifetime* of spaces
  - Manage **files** on behalf of user
    - Pin files in storage till they are released
    - Manage *lifetime* of files
  - Manage file sharing
    - Policies on what should reside on a storage or what to evict
  - Bring the files from remote locations
  - Manage multi-file requests
    - a brokering function: queue file requests, pre-stage
- various storage managers with implementation of SRM interface exists:
  - DRM/HRM (LBNL), BeStMan (LBNL), dCache (Fermilab+DESY), DPM (CERN), CASTOR (CERN)

Scalla+SRM integration overview

- Scalla+SRM integration activity is performed on LBNL storage managers
- LBNL storage managers comes in 2 versions corresponding to different SRM interface specification version
- SRM interface specification version 1.1
  - Disk Resource Manager (DRM)
    - Manages one or more disk resources
  - Tape Resource Manager (TRM)
    - Manages the tertiary storage system (e.g. HPSS)
  - Hierarchical Resource Manager (HRM=TRM+DRM)
    - An SRM that stages files from tertiary storage into its disk cache
- SRM interface specification version 2.2
  - Berkeley Storage Manager (BeStMan)
    - Manages one or more disk resources and also has ability to bring files from tertiary storage

Scalla components architecture

- xrootd architecture is very amenable to extensions (divided into several components and plug-ins)



- xrd - provides networking support, thread management and protocol scheduling
- ofs - provides enhanced first level access to file data (responsible for coordinating activities of oss, odc, auth)
- oss - provides access to underlying storage system (controlled by ofs and invokes meta-data operations)

Scalla+SRM components architecture

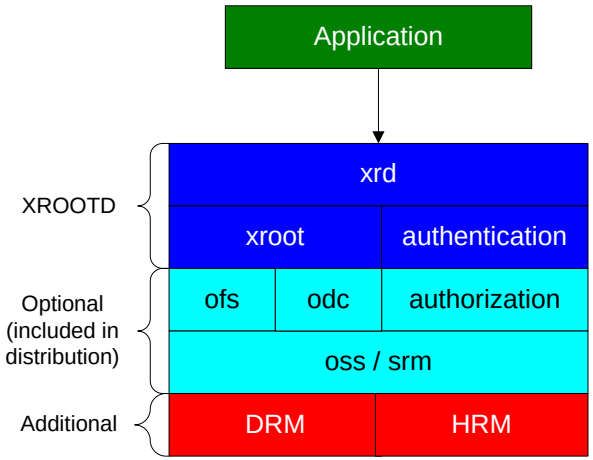
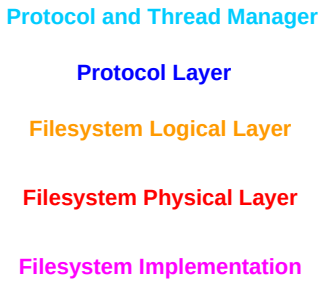


Figure: SRM version 1.1



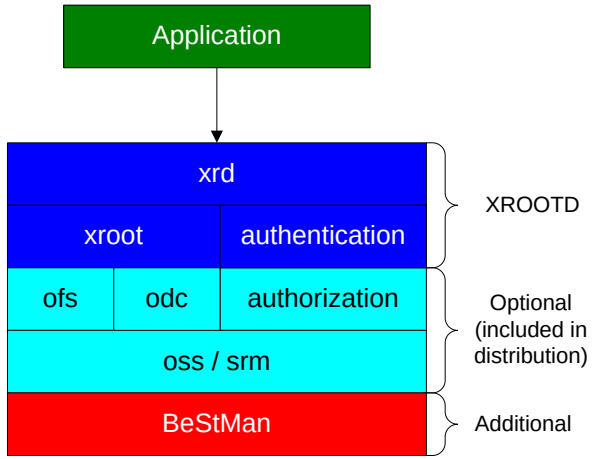


Figure: SRM version 2.2

- oss component was externalized as a plugin
- several existing methods were virtualized (Create, Open, Close, Stage)
  - easy wrapping of concrete class implementation (for example: calling HRM or BeStMan client API)
- for example:
  - Create() uses DRM or BeStMan to create a file
  - Close() informs that the file is no longer in use

Current status ?

- implemented use-cases:
  - Getting files from archive
  - Put in cache only
  - Put in cache and archive ASAP (delay is possible)
  - Modify existing file in cache only
  - Modify existing file in cache and archive ASAP (delay is possible)
  - Migrate existing file from cache to archive ASAP
- SRM v1.1
  - working version supports read, write mode into HPSS with all SRM functionalities as pinning, allocation
  - currently supports one cache partition per a node
  - further development has been closed and effort moved into the version 2.2
- SRM v2.2
  - supports multiple cache partitions per a node
  - alpha prototype available for testing
  - need also testing in large scale (possible further development to increase stability and reliability)

Outline

Summary

- Scalla/Xrootd is currently deployed on almost 500 nodes (the biggest production deployment of xrootd) serving over 350TB of distributed disk space
  - load balancing and handshake with tape system make the system resilient to failures
  - the system is used by users for daily analysis
  - HPSS performance study helped us to increase performance and also decrease the ratio of failed requests (still ongoing work on merging files)
- future plans:
  - large scale testing of SRM 2.2 and possible further development to increase stability and reliability
  - planning a regression tests $\Rightarrow$ performance comparison with home-brewed version)
  - planning enhancements such as cache-to-cache transfers and interoperability with other SRM-aware tools (such as DataMovers etc.)

References

P. Jakl, J. Lauret, A. Hanushevsky, A. Shoshani, A. Sim

From rootd to xrootd: From physical to logical file

Proc. of Computing in High Energy and Nuclear Physics (CHEP'06)

P. Jakl, J. Lauret, M. Šumbera

Managing widely distributed data−sets

Czech Technical University, FNSPE, Research report, 2006

http://www.star.bnl.gov/∼pjakl

A. Romosan, A. Hanushevsky

XROOTD−SRM

SRM−Collaboration meeting, CERN, 2006

P. Jakl, J. Lauret, A. Hanushevsky, A. Shoshani, A. Sim, J. Gu

Data access on widely distributed worker nodes

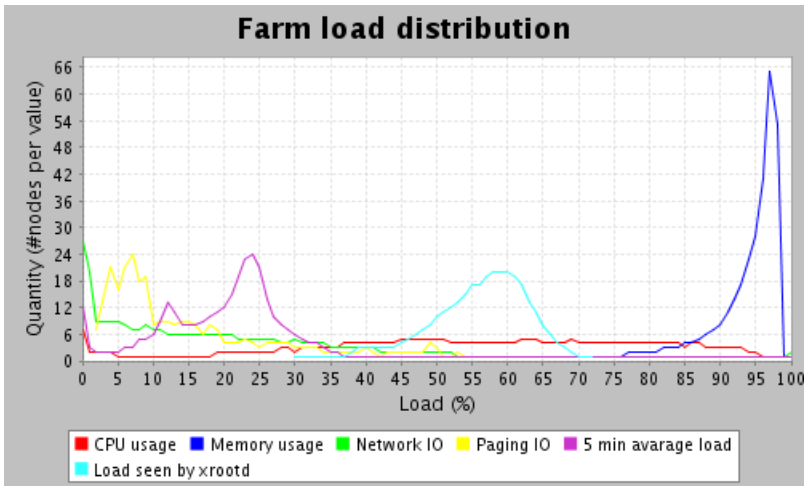
ROOT workshop 2007, CERN, 2007

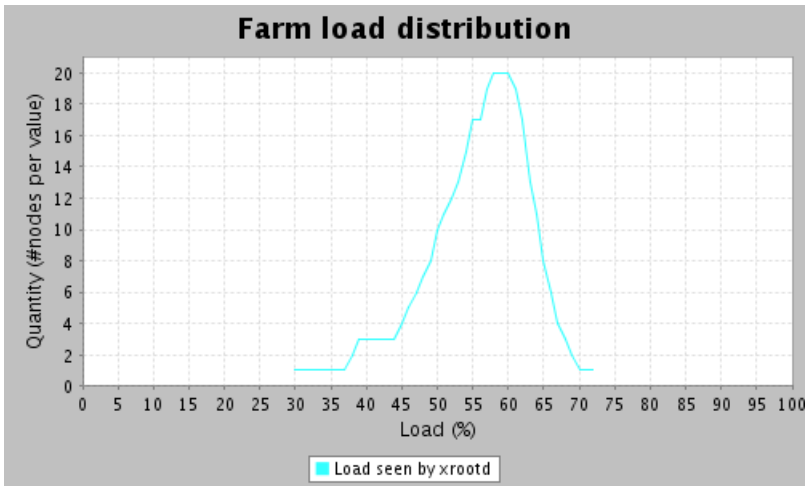
Outline

Backup slides

Monitoring load profile of the cluster

- distributed system can have several choices to fulfill a incoming request (more replicas of a file etc.)
- the system needs to balance the load among many collaborating servers
- Scalla offers computation of the server workload as a flexible formula:
  - it is a combination of 5 main factors (cpu, memory etc.)
- how to setup the thresholds to represent STAR's environment ?
  - is it CPU-bound, memory-bound environment?





Studying load distribution stability

- load distribution illustrated to be pretty stable over longer period of time

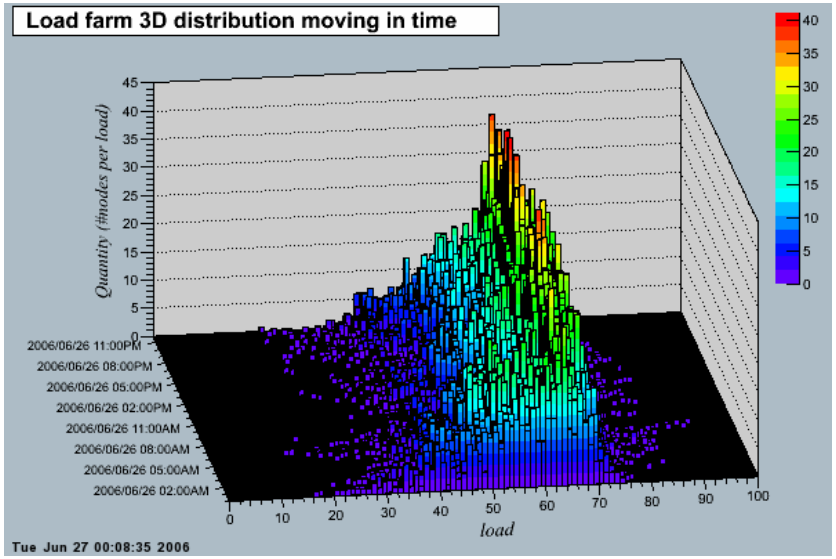


Figure: Week 26, Monday

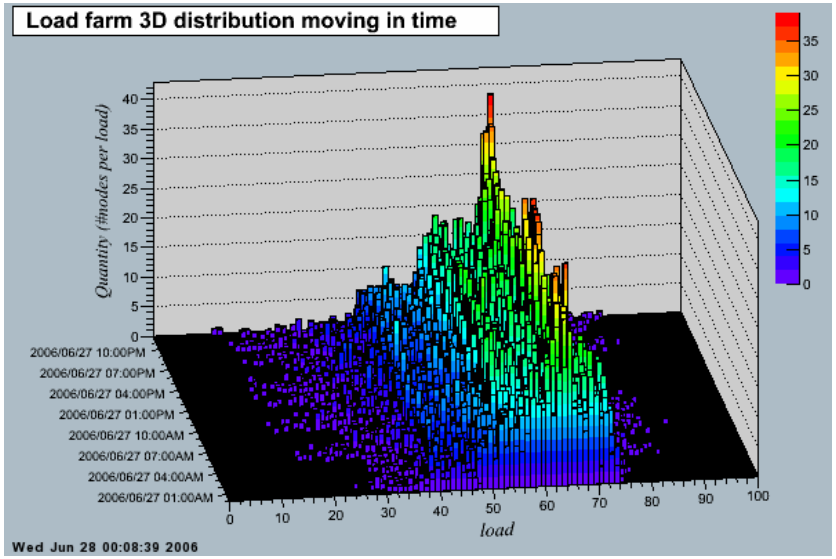


Figure: Week 26, Tuesday