Fermilab's Scientific Computing Storage Architecture (for HTC)

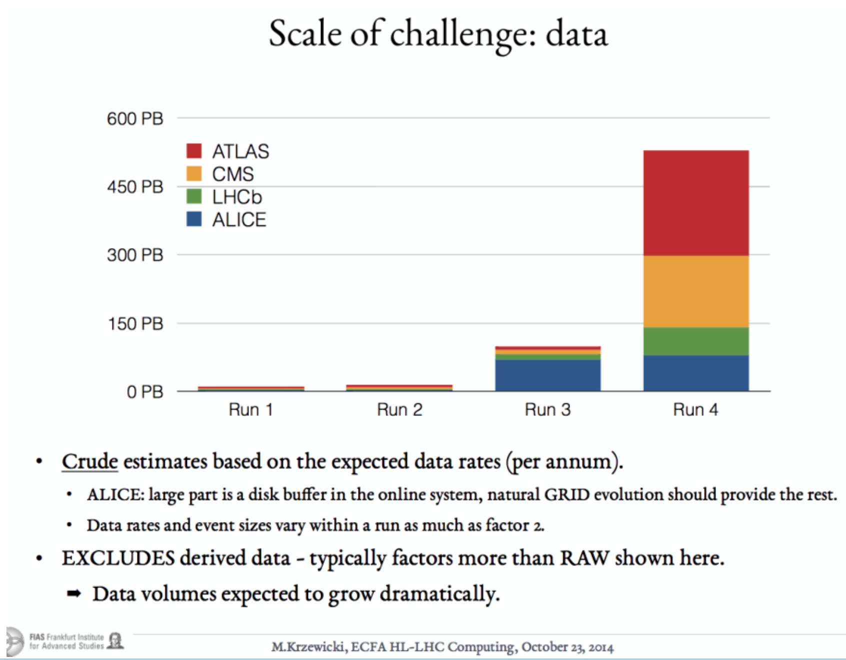
Gerard Bernabeu Altayo (on behalf of the SCSA committee)

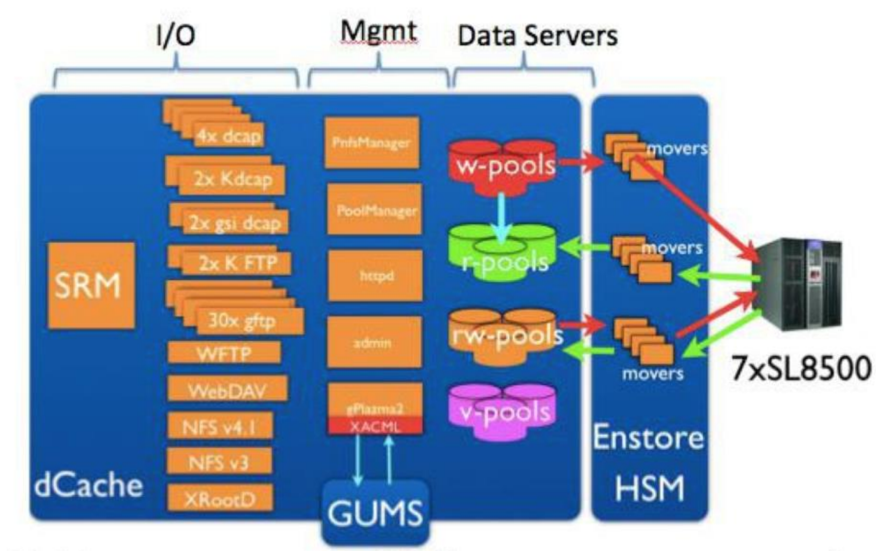
HEPIX Spring 2016

20 April 2016

Introduction I

- High Energy Physics experiments record and simulate **very large volumes of data** and the **trend** in the future is only **going up**. All this data needs to be archived, and accessed by central processing workflows as well as a diverse group of scientists to extract physics results.
- Fermilab **supports a wealth of storage technologies** for the experiments for very different tasks, from NFS mounted appliances to mass storage systems handling transparent access to files on tape through large disk caches.





🔅Fermilab

Introduction II

- To prepare for the future, Fermilab recently developed a Scientific Computing Storage Architecture to consolidate and to prepare for a future evolution of the supported storage forms. A big aspect was **to remove POSIX access to the storage systems from the worker nodes**, necessary to be able to transparently support workflows on different resources like the **local** batch systems, **grids** and commercial **clouds**.
- This will **enable Fermilab experiments to get more resources** (see Tony's Fermilab HEP Cloud: an elastic computing facility for High Energy Physics. talk from Tuesday morning).
- In this presentation, we will discuss the architecture and describe how experts categorizes workloads accessing storage, files and access patterns. We are currently in the process of implementing this **strategy together with Fermilab's experiments** and projects and will report on the progress.

Job Categories

- **Production** batch jobs (centrally submitted and managed, well defined codebase)
- **Analysis** batch jobs (user defined code)
- **Interactive** applications

Next will categorize the phase space of work vs.

- Storage Categories (where is the data stored)
- File Access Mechanisms (how is data accessed)
- File Access Protocols (by which means)

Storage Categories

- **tape-backed MSS -** *Mass Storage System (MSS) orchestrating disk servers and tape robots with tape drives*
- **non-tape-backed MSS -** *Mass Storage System (MSS) orchestrating disk servers. Two varieties:*
  - *Garbage collected, least accessed file replica on disk gets removed if space is needed*
  - *Persistent, when running out of space, writes fail*
- **OSG StashCache -** *Public Xrootd cache infrastructure on OSG*
- **CVMFS -** *Read-only distributed file system based on HTTP caches*
- **Sandbox -** *Group of files or tarball with files needed for the execution of a batch job*
- **Blob DB -** *DB infrastructure including REST APIs to retrieve stored blobs of data*
- **local file system on worker nodes -** *usually accessible as scratch space*
- **network attached storage –** *Shared Disk system providing full POSIX access*

File Access Mechanisms

- **Data streaming:**
  - access files through native protocol of storage, remotely without copying it to the local job environment.
  - Example: xrootd, http, dcap protocol for dCache, etc.
  - This is typically the most efficient solution access large amounts of data (with proper servers and TCP&cache tuning)
- **Copy In:**
  - copy files to local file system on worker node or interactive node
- **Copy Out:**
  - copy files from worker or interactive node to storage
- **Blob DB read:**
  - Access blobs from Blob DB through REST APIs

File Access Protocols

- **Xrootd:**
  - Access files through xrootd protocol (requires xrootd server infrastructure)
  - Both copy-in/out (xrdcp) or streaming (native root file open with xrootd url)
- **dcap**
  - dCache specific protocol, preferably use XrootD instead.
- **Submission infrastructure:**
  - Use file transport protocol of the submission infrastructure (eg Condor sandbox)
  - This is a copy-in/out mechanism
- **HTTP:**
  - Download files through HTTP protocol
  - Both copy-in/out (wget) or streaming (native root file open with HTTP url)
- **POSIX-like:**
  - Access files through POSIX-like interfaces to storage
  - Interface does not provide full POSIX functionality
  - Examples: dCache-NSF4, EOS-FUSE
- **POSIX (read):**
  - (Read-only) access to files through fully POSIX compliant interface to storage
  - Example: reading from CVMFS or accessing files from network attached storage

File Input Categories

- **code** files - *run time executable, libraries and code files; two classes:*
  - immutable base releases of experiment code (N GB)
  - user-specific code as add-on to base release or stand-alone (< 100MB/job)
- **support** files - *job specific inputs*
  - Examples: configuration files, txt files. < 10 MB per job
- **auxiliary** files - additional inputs to processing and analysis jobs that have high reusability rates
  - Examples: flux files, pile-up files
  - 10 MB to N GB (N ≥ 1)
- **data** files:
  - *holding data from data taking and MC simulation, low reusability rates*
  - Examples: RAW detector files
  - 100 MB to N GB (N ≥ 1)

File Output Categories

- **log** files:
  - *text files holding status and error outputs produced during execution of applications, accessed for problem debugging*
  - Examples: stdout, stderr
- **histogram** files:
  - *Output generated by applications that can directly be used for end-analysis and is limited in size*
  - Example: histograms in root files, small ntuples in root files
  - 10 MB to 100 MB
- **output** files:
  - *holding data from data taking and MC simulation, low reusability rates*
  - Examples: RAW detector files
  - 100 MB to N GB (N ≥ 1)

Storage vs. File Category Matrix

	Production		Analysis		Interactive	
	Input	Output	Input	Output	Input	Output
tape-backed MSS (going through DM solution)	auxiliary data	output	auxiliary data	output	auxiliary data	output
non-tape-backed MSS	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output
OSG StashCash	auxiliary		auxiliary		auxiliary	
network attached storage					code support auxiliary data	log histogram output
local file system on worker nodes						
CVMFS	code		code		code	
Sandbox	code support		code support			
Blob DB	support		support		support	

Recommending data streaming for all Input data.

Fermilab

Storage vs. File Category Matrix

	Production		Analysis		Interactive	
	Input	Output	Input	Output	Input	Output
tape-backed MSS (going through DM solution)	auxiliary data	output	auxiliary data	output	auxiliary data	output
non-tape-backed MSS	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output
OSG StashCash	auxiliary		auxiliary		auxiliary	
network attached storage					code support auxiliary data	log histogram output
local file system on worker nodes						
CVMFS	code		code		code	
Sandbox	code support		code support			
Blob DB	support		support		support	

POSIX supported but very limited CPU/storage available

Recommending data streaming for all Input data.

Storage vs. File Category Matrix

	Production		Analysis		Interactive	
	Input	Output	Input	Output	Input	Output
tape-backed MSS (going through DM solution)	auxiliary data	output	auxiliary data	output	auxiliary data	output
non-tape-backed MSS	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output
OSG StashCash	auxiliary		auxiliary		auxiliary	
network attached storage					code support auxiliary data	log histogram output
local file system on worker nodes						
CVMFS	code		code		code	
Sandbox	code support		code support			
Blob DB	support		support		support	

ReadOnly solutions scaling out with intensive caching use

Recommending data streaming for all Input data.

Storage vs. File Category Matrix

	Production		Analysis		Interactive	
	Input	Output	Input	Output	Input	Output
tape-backed MSS (going through DM solution)	auxiliary data	output	auxiliary data	output	auxiliary data	output
non-tape-backed MSS	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output
OSG StashCash	auxiliary		auxiliary		auxiliary	
network attached storage					code support auxiliary data	log histogram output
local file system on worker nodes						
CVMFS	code		code		code	
Sandbox	code support		code support			
Blob DB	support		support		support	

Scale out distributed storage solutions, proven to scale when streaming data (eg: CMS@FNAL averages 30GiB/s read with thousands of open files)

Recommending data streaming for all Input data.

Process is experiment per experiment

1. Meet with the experiment to
   - explain the project goal: stop using ‘mounted’ resources (eg NFS)
   - Request data flow diagrams (online and offline).
2. Analyze data flow diagrams and propose changes. Priorities:
   1. Removing NAS/POSIX
   2. Consolidate data management and access (F-FTS, IFDH) across experiments and aligning with SCD portfolio
   3. Data streaming (vs copy in/out)
3. Experiment implements the changes with support from the SCD.
   - Experiment is motivated to get more resources. Removing POSIX enable the jobs to run on OSG and other remote resources!

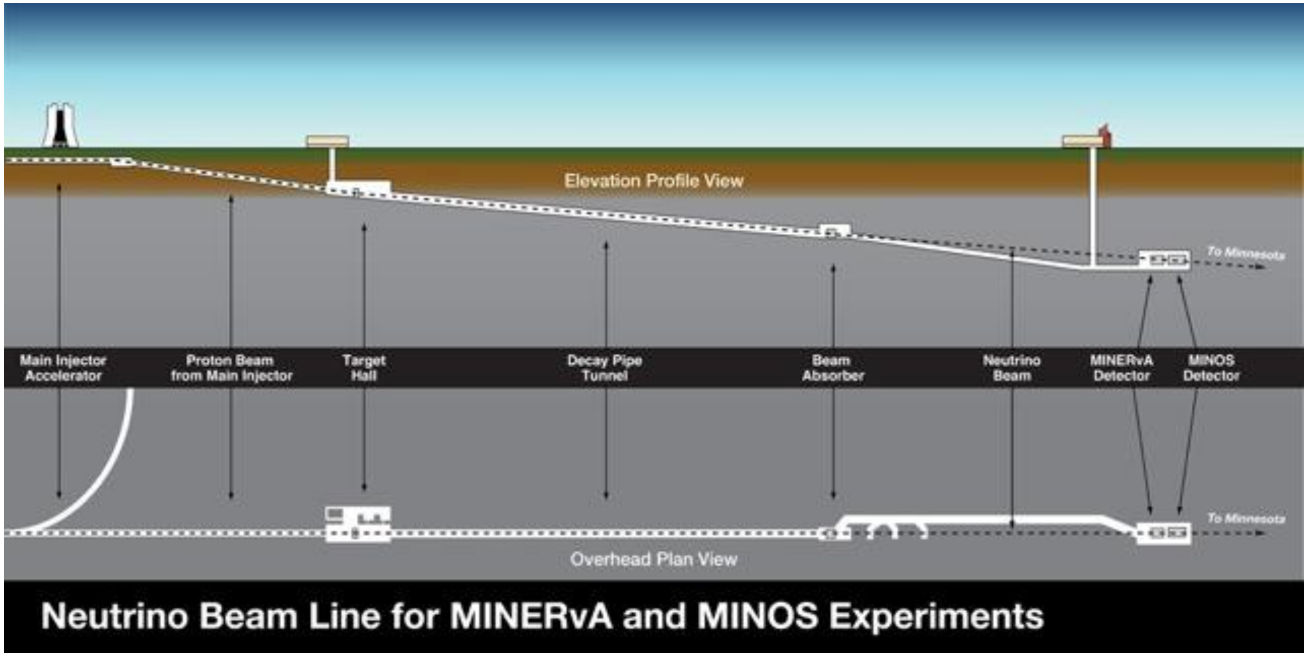
https://cdcvs.fnal.gov/redmine/projects/sam/wiki/File_Transfer_Service_Information

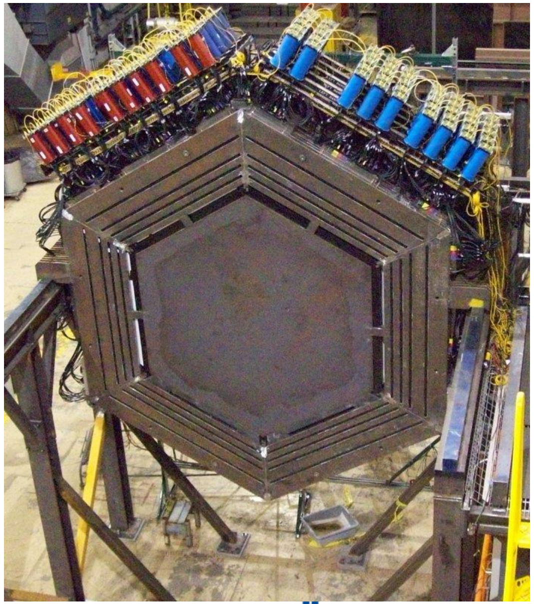
IFDH (Intensity Frontier Data Handling), is a suite of tools for data movement tasks

Example: The Minerva case

http://minerva.fnal.gov/

MINERvA (E938) is the first neutrino experiment in the world to use a high-intensity beam to study neutrino reactions with five different nuclei, creating the first self-contained comparison of interactions in different elements. While this type of study has previously been done using beams of electrons, this is a first for neutrinos.





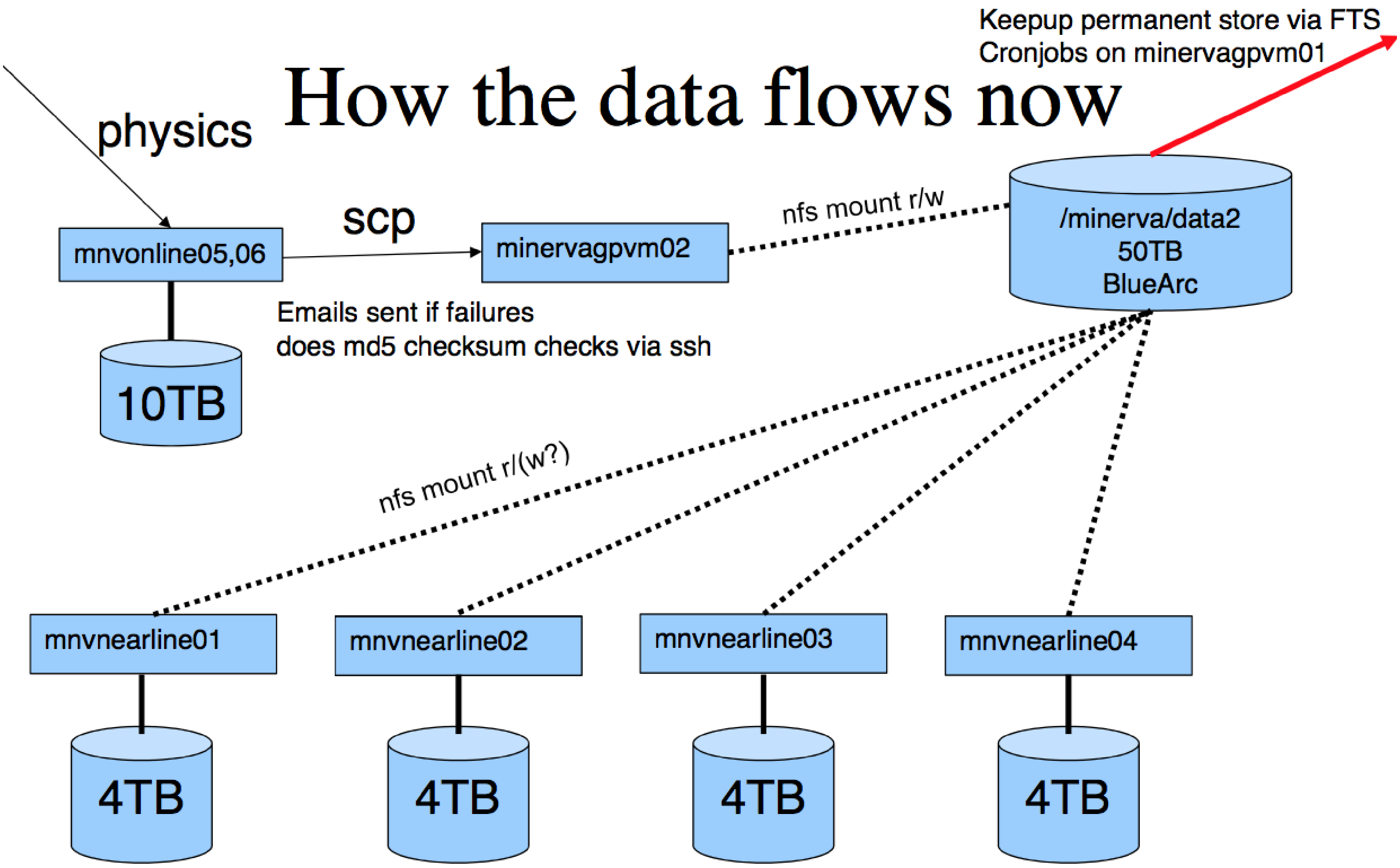
Example: The Minerva case (from D. Ruterbories)

Overview

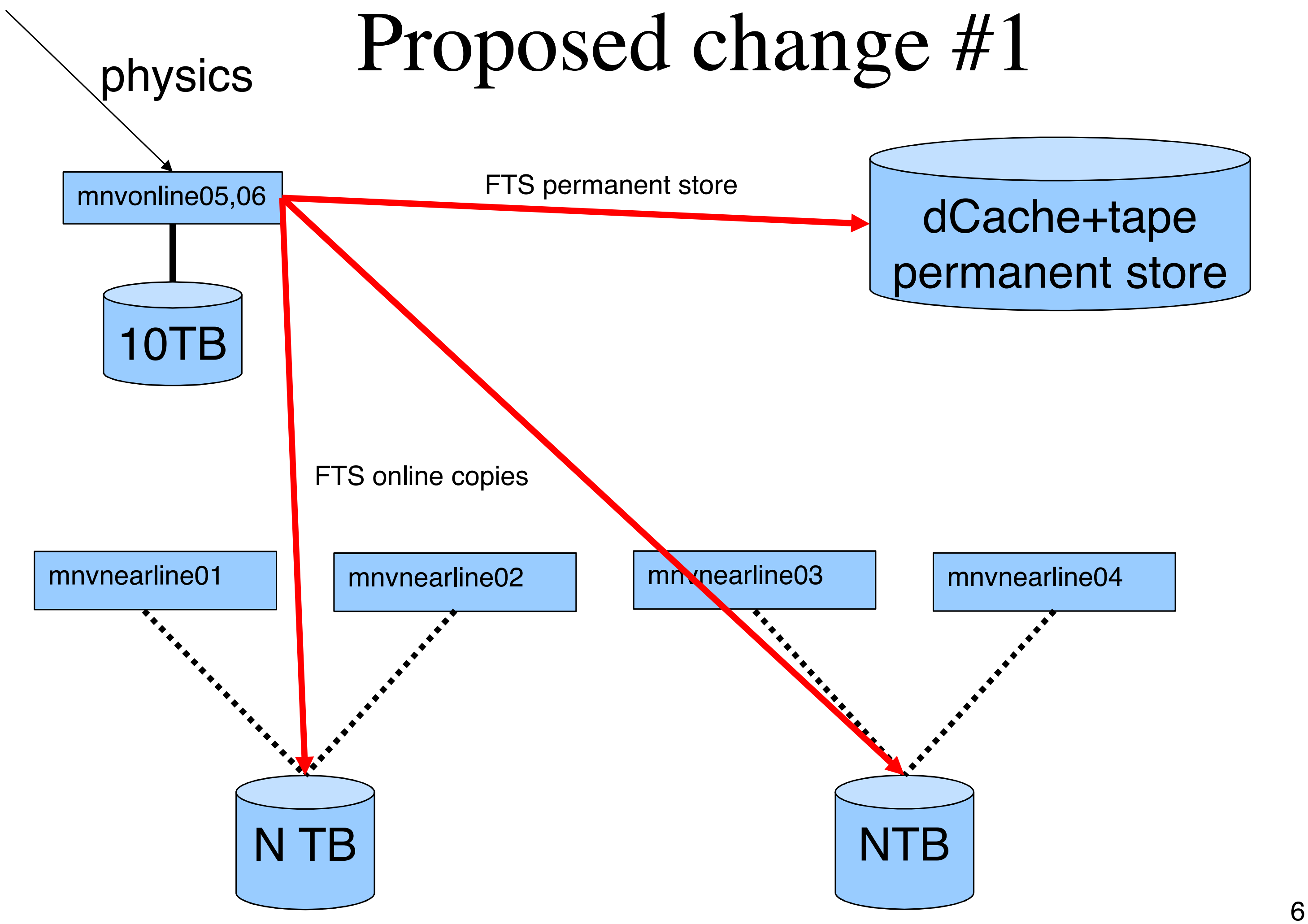
- Offline systems has to deal with BlueArc going away.
  - Grid jobs cannot access BlueArc
  - Disk space will disappear
- Online systems will also have to adapt to this change in some form or another.
  - Sensitive because this is our data
- We want to minimize person power costs wherever possible…
- Can offline experience help out?

2

Example: The Minerva case (from D. Ruterbories)



Example: The Minerva case (from D. Ruterbories)



Example: The Minerva case – Software distribution

- Mimic what other experiments are already doing (eg CMS)
  - Publish main Software releases on CVMFS (OASIS, mounted on all OSG resources)
  - Users ship their code modifications via a TAR with jobsub (condor sandbox) and overlay it (eg with CMT). – CMS equivalent to CRAB3

- The user code section is 5-100MB for Minerva. Instead of using the standard Condor *schedd* to transfer it to each WorkerNode we may need to use a scalable HTTP server as source.

Summary & Questions

Presented the Fermilab Storage Architecture
- Goal is to enable everyone to efficiently and transparently benefit from opportunistic resources and remote resources

Working with experiments to implement it now.
- First experiment to move is Minerva, gaining experience

?