PIC
port d'informació
científica

Managing small files in Mass Storage systems using Virtual Volumes

E. Acción, M. Delfino (Port d'Informació Científica, Spain),
E. Hernández (Univ. Simón Bolívar, Venezuela),
A. Pacheco (Institut de Física d'Altes Energies, Spain)

Contributed talk to the
Computing Facilities and Networking track of the
Computing in High Energy Physics '06 conference,
Mumbai, India, 13-17 February 2006

Generalitat de Catalunya
Departament d'Universitats, Recerca i Societat de la Informació

MINISTERIO DE EDUCACIÓN Y CIENCIA
Ciemat
Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

UAB
Universitat Autònoma de Barcelona

IFAE

PIC
port d'informació
científica

Outline

- The "small file problem" : storing small files to tape is very inefficient
- *Operating Systems 101*: Volumes and Files
- Nested Volume-File architecture
- The ISO 9660 standard and the *mkisofs* tool:
  a solution for creating virtual volumes
- *Operating Systems 201*: Automounters
- The *amd* automounter and volume type "program"
- *Systems Engineering 210*: read/write segregation
- Example implementations:
  - Medical Images (Hospital Parc Taulí, Sabadell, Spain)
  - MAGIC Gamma-ray telescope
- *Systems Engineering 510*: Catalogs, coupling, etc.
- Conclusions and Outlook

PIC
port d'informació
científica

The "small file problem" : storing small files to tape is very inefficient

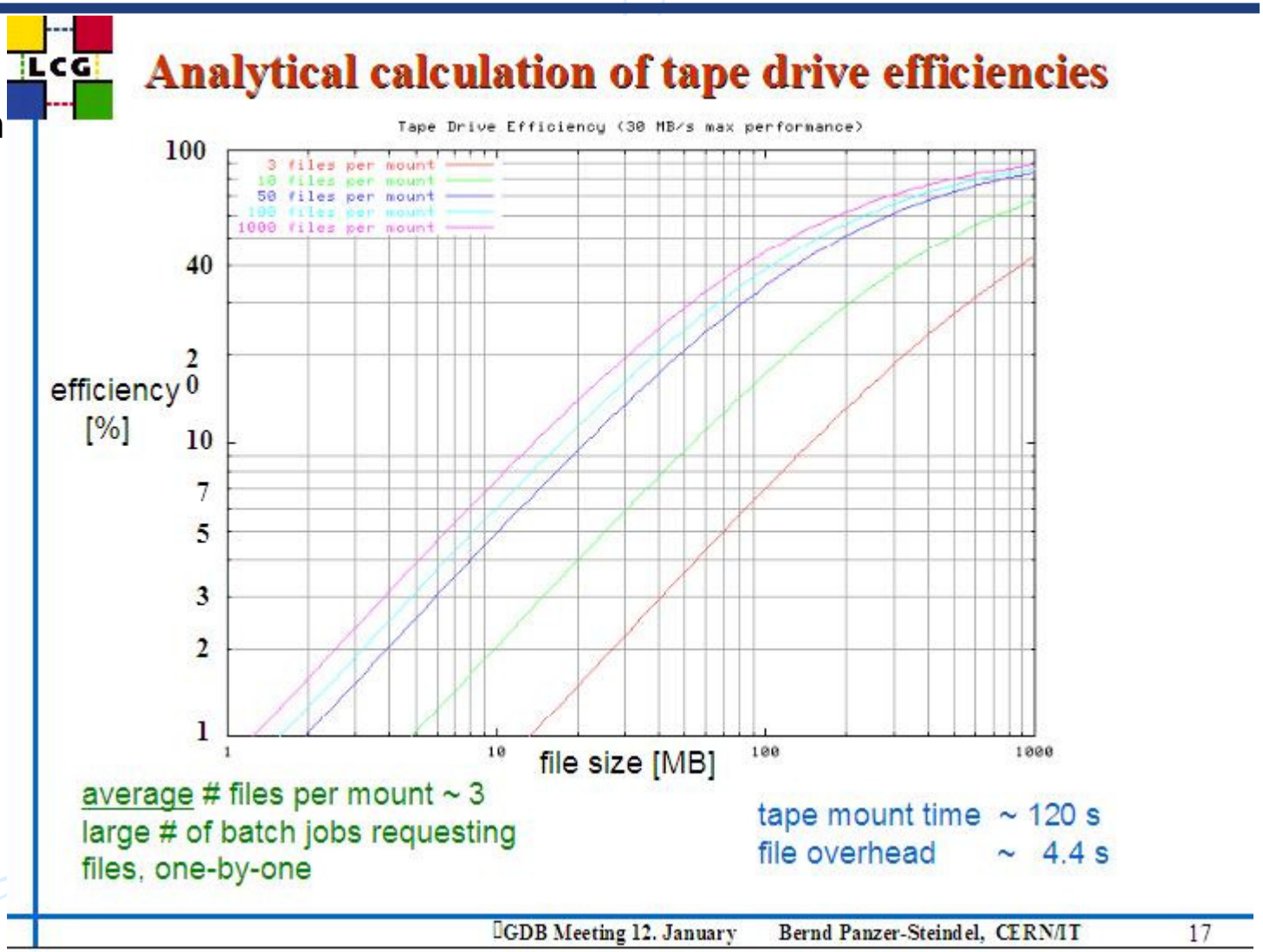
Thanks to Bernd Panzer-Steindel from CERN for the diagram.

Parameters are for Storagetek 9940B tape drives

Note that even a disk-based MSS would have:

- "file overhead" (processing of the open statement)

- "tape mount time" (power mgmt. spinup of disk)



PIC
port d'informació
científica

Operating Systems 101: Volumes and Files

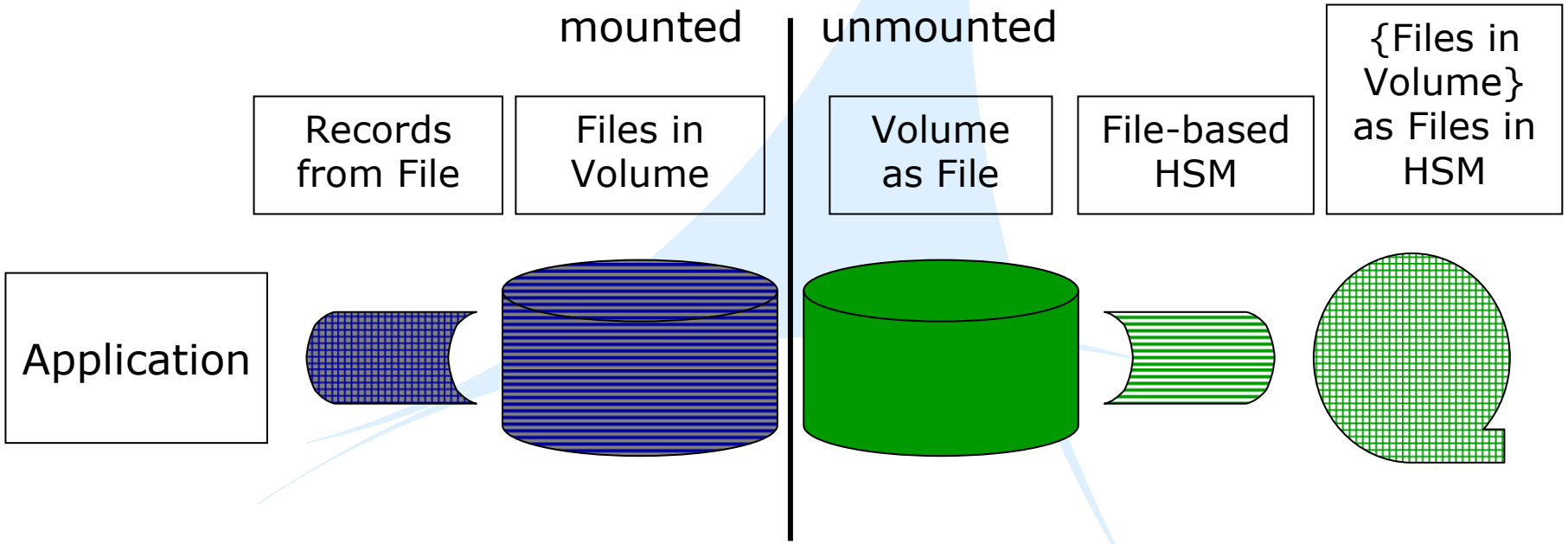
Record:	A sequence of bytes representing coherent info
HEP Event Record:	*Bytes recorded at an "event" (collision, beam crossing, etc.)*
File:	A sequence of records representing coherent info
HEP Data File	*HEP Records bunched together by physicists to please SysAdmins*
Volume:	A physical or logical container of files (low-level)
Filesystem:	A logical container of files (high-level)
Namespace:	Translation table linking high-level to low-level

- Key points about volumes:
  - They have to be "initialized" (FORMAT A:)
  - They have to be made available (mount)
  - They can be made not available (umount)
  - Organized according to fixed criteria (block size, overall size..)
  - I/O is done at a low-level (usually block-level)

PIC
port d'informació
científica

Nested Volume-File architecture

- A volume can be represented as a file
  - Backup "saveset" from Veritas, MS Backup, etc.
  - tar, gzip, ZIP compressed "archives"
  - Partitions treated as files
  - Volume+Filesystem as file (ISO9660 and others)
- Nested architecture:



PIC
port d'informació
científica

The ISO 9660 standard and the mkisofs tool: a solution for creating virtual volumes

- Probably the most widely used (removable) storage standard in the world. All data CDs and some DVDs. (See http://en.wikipedia.org/wiki/ISO_9660)
- Advantage over tar, ZIP, etc.: Directly mount, no copying.
- Other advantages:
  - "Naturally" defragmeted, excellent file reading speed (but driver uses CPU)
  - Mount "caches" directory structure, excellent for opening many small files
- The ISO standard carefully maintained, updated once (ISO 9660:1999).
- Generation of ISO 9660 image files supported by *mkisofs* command (see http://freshmeat.net/projects/mkisofs/)
- "Rockridge" and "Joliet" extensions allow nearly unlimited preservation of Unix and Windows file attributes, respectively.
- Chained structure:
  - No predetermined limit on volume size. ☺
  - Hard limit on maximum file size: 4 GB ☹
  - We are still trying to understand if there are other limits.
- Very robust. We have written and verified many volumes. Examples:
  - One application uses many volumes above 100 GB
  - Another application has 750,000 files of 100 Bytes in a volume

PIC
port d'informació
científica

Operating Systems 201: Automounters

- An "automounter" is a process that handles I/O traps ("faults") to predefined graft points in a namespace, making the corresponding filesystem/volume appear ("mounting it").
- This allows transparent completion of I/O to volumes which are offline but are known and available.
- Historically used to get around O/S overload on NFS.
- Best known automounters: *autofs* and *amd.*
- *autofs* is commonly used in Linux/Unix installations.
- *amd* comes from Solaris, now commonly available in Linux. Simple, but very robust and debugged.
- The use of an automounter allows handling the I/O faults at the system level and avoids having to link application with special libraries. This is desired by many projects.

The *amd* automounter and volume type "program"

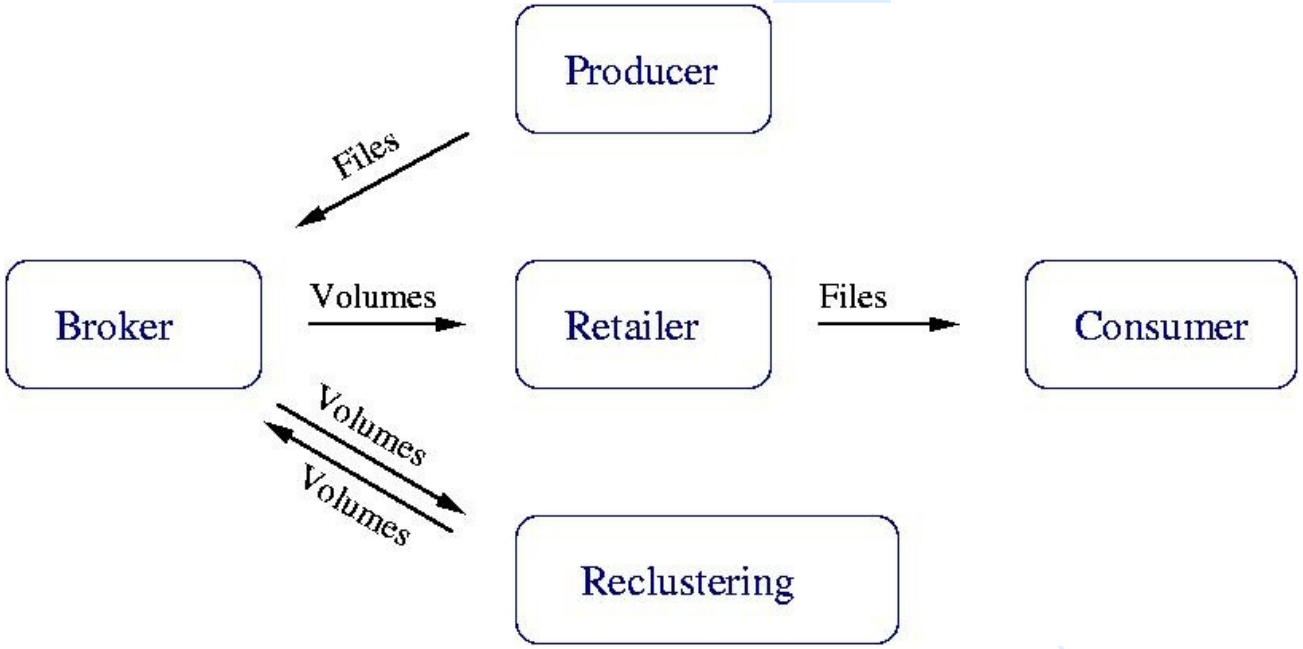
- *amd* implements a hook by which a graft point can be declared of type "program", giving a script/program name which is called automatically to perform a custom *mount* / *umount*.
- We have used this hook to implement the sequence:
  - Application "faults" while doing I/O to part of a namespace
  - *amd* is triggered and calls our script
  - The script
    - finds space in its assigned disk buffer
    - recalls from CASTOR the file for corresponding ISO 9660 volume
    - mounts the file as a volume at the graft point
  - Control is returned to amd and the I/O completes transparently
- We also implement a dismount procedure: leave the ISO file in cache for a specified timeout, then delete.
- We call this "Virtual Volumes" or **ViVo** (street-smart in Spanish)
- But ViVo is more general, using *amd* is just one implementation

Systems Engineering 210: read/write segregation

- One thing we have learned "loud and clear" in operating PIC:

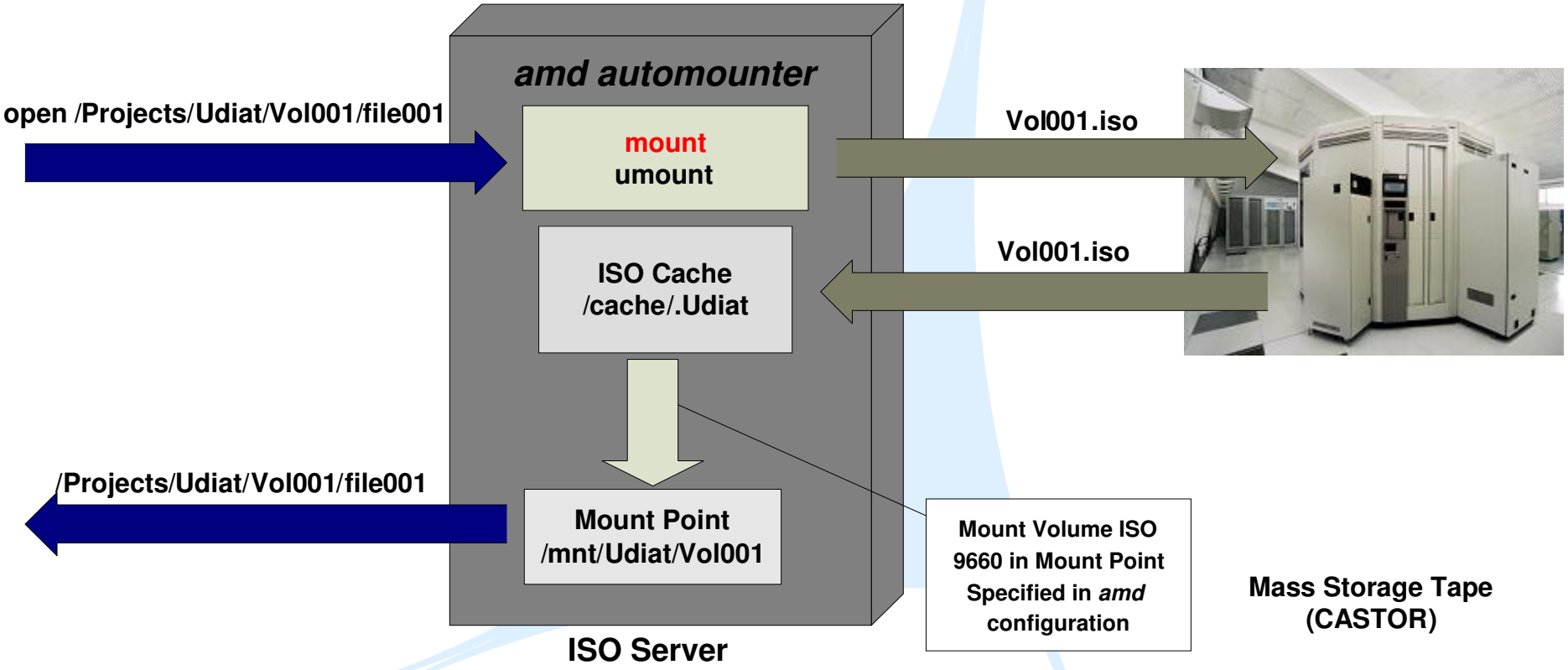
> Treat ***writing*** and ***reading*** as two completely separate channels, only coupled via "the catalog".

- ViVo penalizes write completion latency to benefit read performance
- The "big picture" system diagram:



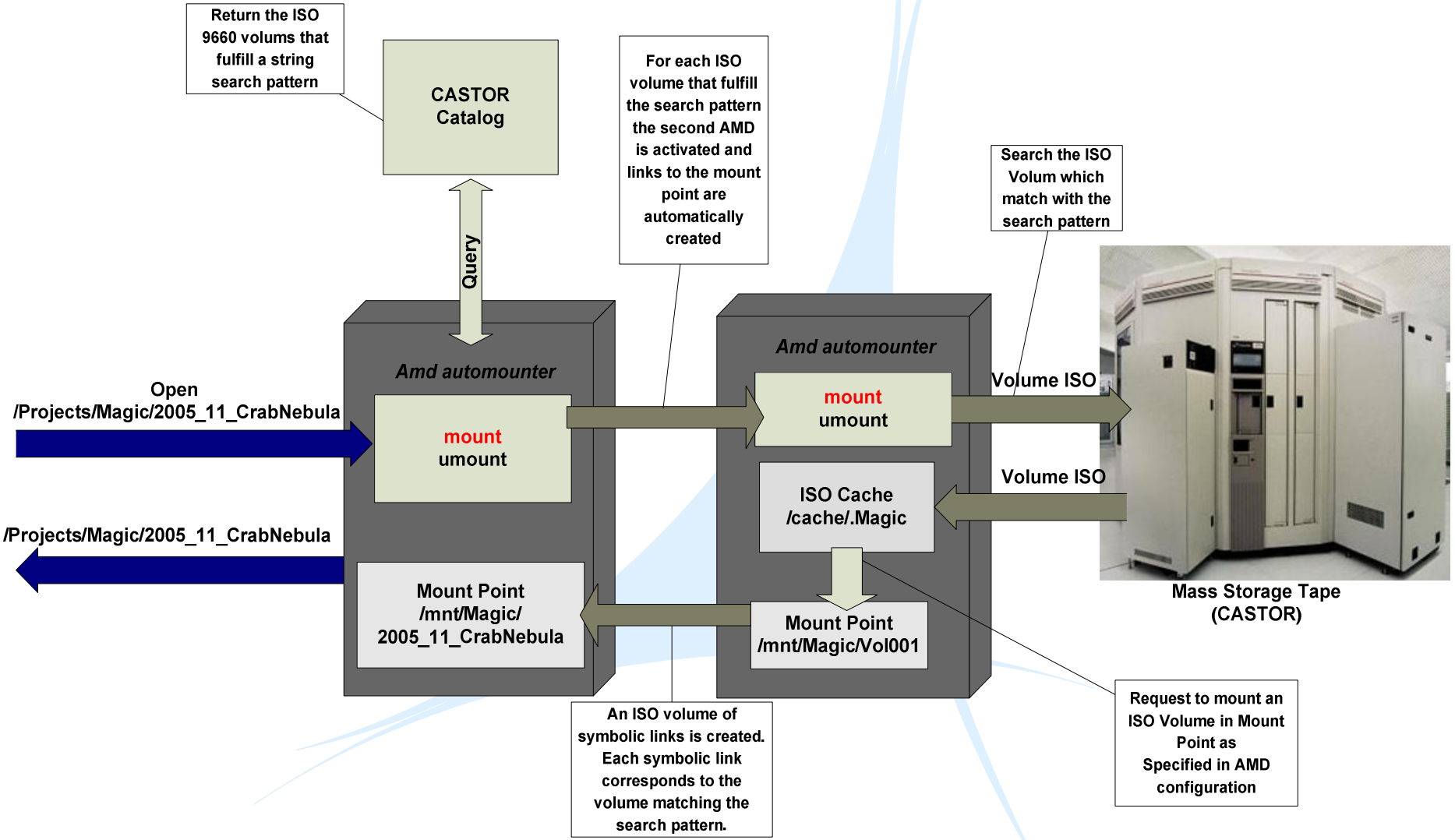
PIC
port d'informació
científica

Example implementations: Medical Images (Hospital Parc Taulí, Sabadell, Spain)



PIC
port d'informació
científica

Example implementations: MAGIC Gamma-ray telescope



PIC
port d'informació
científica

ViVo is being used in production

- Hospital Parc Taulí
  - Nightly transfer of all medical images acquired during the day
  - 2004 (part of year)
    - One directory per day = One ISO volume stored to MSS
    - Variable size, some as large as 15 GB/volume
    - Separate data path from normal recording to DVD at hospital
  - 2005 (full year)
    - Fully integrated with hospital image management system
    - One directory per day = Several ISO volumes (cutoff=4,7 GB)
    - Common data path for DVD recording and PIC transmission
  - 9.2 TB, almost 3000 ISO volumes, average filesize < 1 MB
- MAGIC
  - Started quasi-online data acquisition from La Palma 11/11/05
  - One directory per night = n ISO volumes (n sources observed)
  - 2.5 TB, 272 ISO volumes on tape
  - Starting to discuss with collaboration:
    - Reclustering of data by observed source
    - "DVD on demand" service: Collaborator fills out menu, data catalog is queried, custom ISO volume file is returned to collaborator(s).

Systems Engineering 510: Catalogs, coupling, etc.

- Where is/are the catalog(s)? Do you need it/them?
- Some have gone "catalog-crazy", the same "file" is catalogued in multiple places:
  - In Experiment-specific catalog and in Mass storage catalog
  - In Grid catalog and Local Mass storage catalog
  - In all of the above
- Note that ***catalogs***, or any centralized information access, is dangerous for embarrasing-parallelism, as it introduces ***coupling*** between datasets.
- In real implementations, you cannot get around it completely. We have made a minimalistic implementation of ViVo where:
  - Each volume has to be checked into *amd* (a kind of catalog)
  - We reuse the CASTOR namespace to generate a directory full of symbolic links to ViVo volumes, using simple lexical rules
- We think more attention should be given to efficient catalogs

PIC
port d'informació
científica

Conclusions and Outlook

- We have successfully deployed a combination of common O/S tools (*mkisofs* and *amd*) in order to handle large numbers of (small) files in "containers" which are (large) ISO 9660 files which are handled through PIC's Castor MSS.
- We have given this the name "Virtual Volumes" (acronym **ViVo**).
- This **vastly** improves the efficiency of the MSS at very little cost, as long as the volumes are populated according to file access patterns.
- Only suitable for a WORM environment.
- In production for Parc Taulí Hospital and MAGIC.
- Work-in-progress:
  - How to share out automounted ISO volumes to a whole farm without all nodes running *amd*.
  - Monitoring of access patterns and re-clustering of files in volumes.
  - Better/easier integration with catalogs used by projects.
  - Transmitting ISO images between centers instead of individual files.
- Extensions under study:
  - ISO volumes served by Apache, GridSite, dCache or xrootd.
  - Connection between SRM and ViVo: Direct? Via dCache or xrootd?