Storage virtualization @ CC-IN2P3: SRB & iRODS

Jean-Yves Nief

dapnia
cea
saclay

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

What is CC-IN2P3 ?

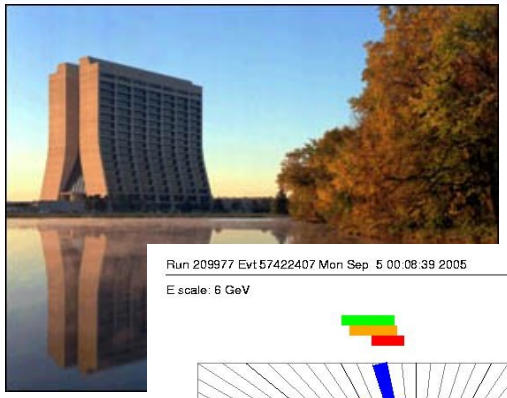
- Federate computing needs of the french community:
  - Particle and nuclear physics
  - Astroparticles and astrophysics.
  - Opening to biology, Arts & Humanities.

CNRS IN2P3
Institut National de Physique Nucléaire et de Physique des Particules

DSM cea dapnia

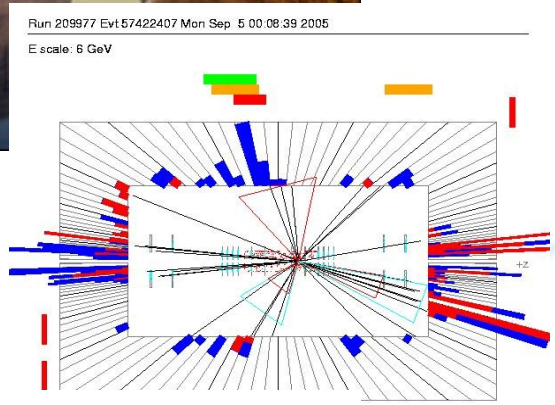
CCIN2P3

CC-IN2P3 mission



Physics experiments

- *Particle & nuclear physics*
- *Astroparticles*

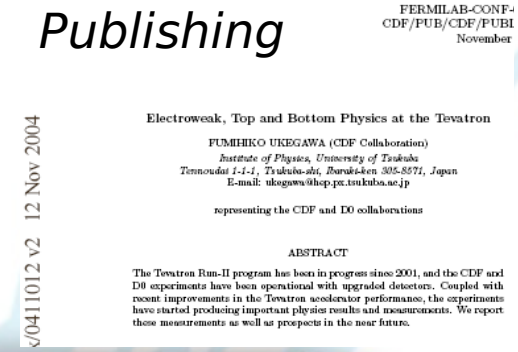


Fundamental research

Data

101000 100111
0001001010001 00011101
100010001111 00010 101000
100111 0001001010001
00011101 100010001111
00010 101000 100111 110001
111010
0001001010001101000
100111 0001001010001
00011101 1110 100010001111
00010 101000 00 11
0001100111 0001001010001
00011101 100010001111
00010 101000 100111
0001001010001

Publishing



Data analysis

Computing needs at CC-IN2P3

- 5000 users, 80 groups:
  - Users can be also foreign collaborators.
- Access also through the grid (LCG/EGEE).
- Around 10000 cores.
- Two batch farms (PCs):
  - Serial analysis.
  - Parallel analysis (MPI, PVM).
- CPU power doubles every year.

Storage needs at CC-IN2P3

- Multiple data storages:
  - Hardware:
    - Disks (3 PB).
    - Tapes (5 PB).
  - Software:
    - HPSS (mass storage system): up to 70 TB/day (read/write).
    - Parallel filesystem: GPFS.
    - Global filesystem: AFS.
    - « HEP home made filesystems » (dCache, xrootd).
    - Relational databases (Oracle, mySQL etc..).
    - First step towards virtualization.

Storage virtualization: why ?

- Scientific collaborations spread world-wide:
  - Data can also be spread among different sites.
- Using heterogeneous:
  - storage technologies.
  - operating systems.
- Virtual organization needed:
  - Authentication and access rights to the data.
- Storage virtualization:
  - To be independent from technology and hardware evolution.
  - To be independent of local organisation of the files (servers, mount point etc…).

➔ Logical view of the data independent of the physical location.

Solutions ?

- Need for a « grid » middleware.
- SRB (Storage Resource Broker) is anwswering these needs:
  - Developed by SDSC: start in 1998.
  - Under the license of General Atomics.
  - Developers in constant contact with the user community:
    - Functionnalities asked by the users.
  - Portable on many OS and platforms.
  - Support of a vast number of storage system, no limit.
  - Large user community.

Some SRB features

- Logical organization of the data decoupled from the physcal organization:

Collection:	prod02
Collection Parent:	/in2p3/mc/neutrino/nc
Propriétaire:	ant_write@ccin2p3

Nom Donnée	Date Creation	Propriétaire	Num. Replica	Num. Version	Taille	Type Donnée
nu01.evt.gz	2006-10-24-14.15.30	ant_write@ccin2p3	0	0	27808157	generic
nu02.evt.gz	2006-10-24-14.15.30	ant_write@ccin2p3	0	0	27778433	generic
nu03.evt.gz	2006-10-24-14.15.28	ant_write@ccin2p3	0	0	27209758	generic

- Various browser tools: GUI, Web, APIs, shell commands called *Scommands* (Scd, Smkdir, Sput …).
- Authentication: password, certificate X509.
- Organization of the users' space by:
  - Type (sysadmin, domainadmin, simple user…).
  - Zones, domains, groups.
- ACL on the objects and directories.
- Tickets: temporary rights on a file.

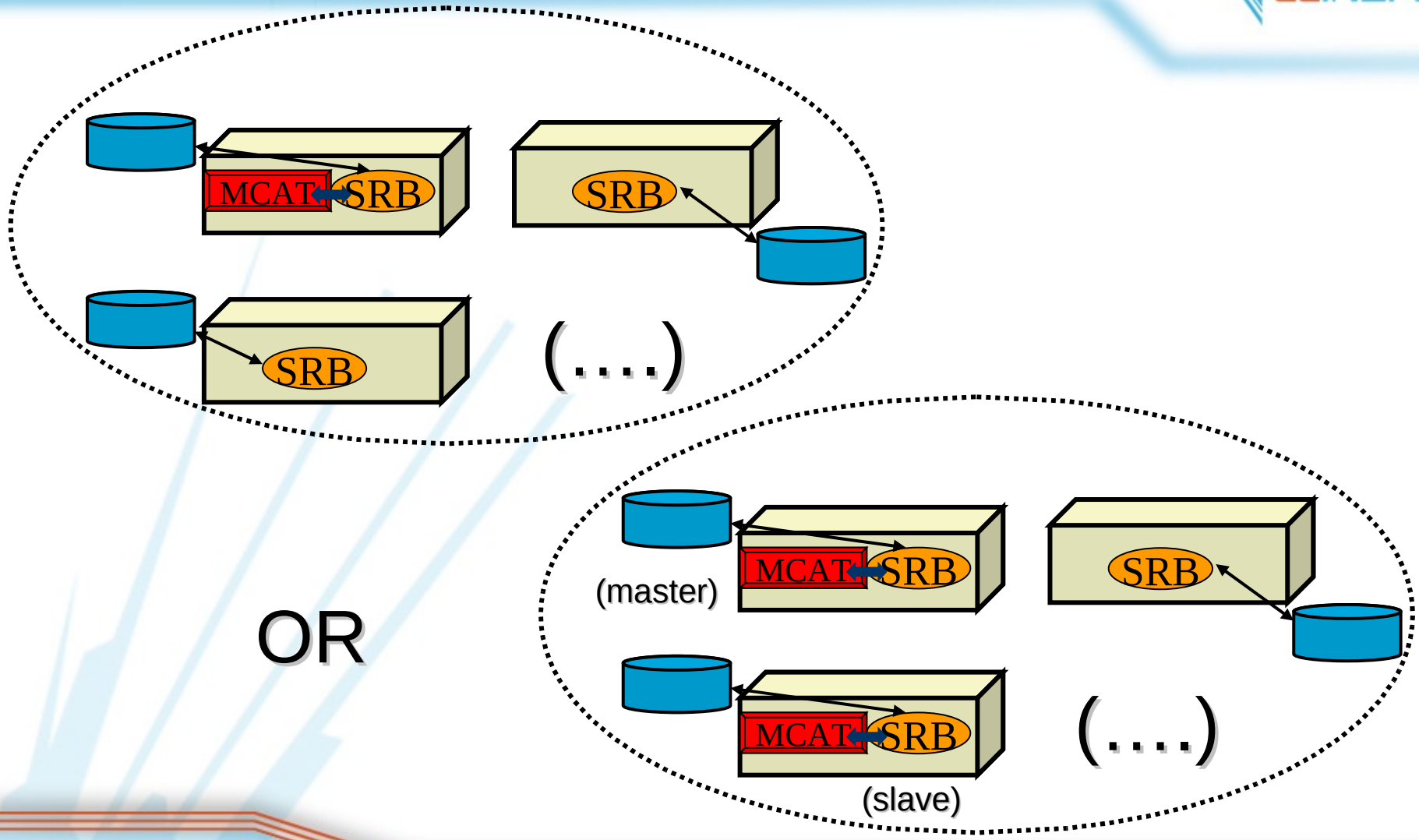
Some other SRB features

- Replica and versions handling.
- Access to the data from their attributes instead of their name and physical location.
- File search by metadata associated to them.
- Files annotations.
- Auditing: record of all the actions on the SRB.
- Storage resources hierarchy:
  - Logical resources: set of physical resources.
  - Compound resources: cache resource (temporary eg: disks) + 1 archival resource (ex: tapes).

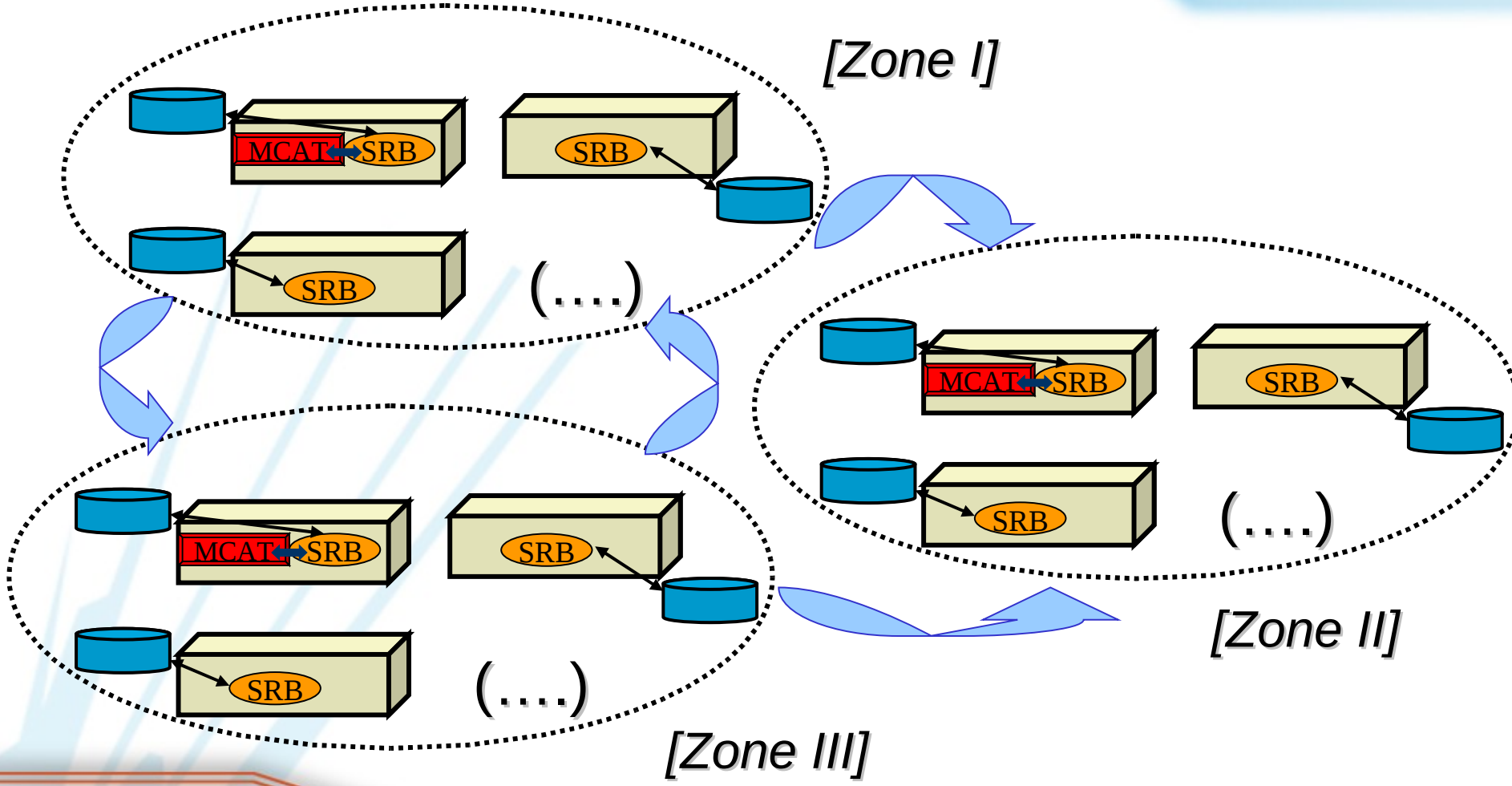
Yet some other SRB features

- Databases:
  - Extensible MCAT: interface of your database schema with the one from SRB.
  - Shadow objects: database access through SRB (behave as a proxy server).
- Possible to interface SRB with any systems which provides POSIX APIs.

SRB architecture (1 federation)



SRB architecture (n federations)



SRB in Lyon

HEP	*BaBar*	*SLAC « mirror » site*
	CMOS, Calice	*Data archival*
	Indra	*Data distribution and archival*
	Lattice QCD	*hundreds of TB / y*
Astroparticle	Antares	*Main center: ~200 TB / y*
	Auger, Virgo	*Main center: tens of TB / y*
	Edelweiss	*Main center: tens of TB / y*
	SN Factory	*Part of the online: ~GB / d*
Biomedical	*BioEmergence*	*European project ~ TB/y*
	Mammography	*Project with a computing lab*
	Neuroscience	*Lyon and Strasbourg hospital*

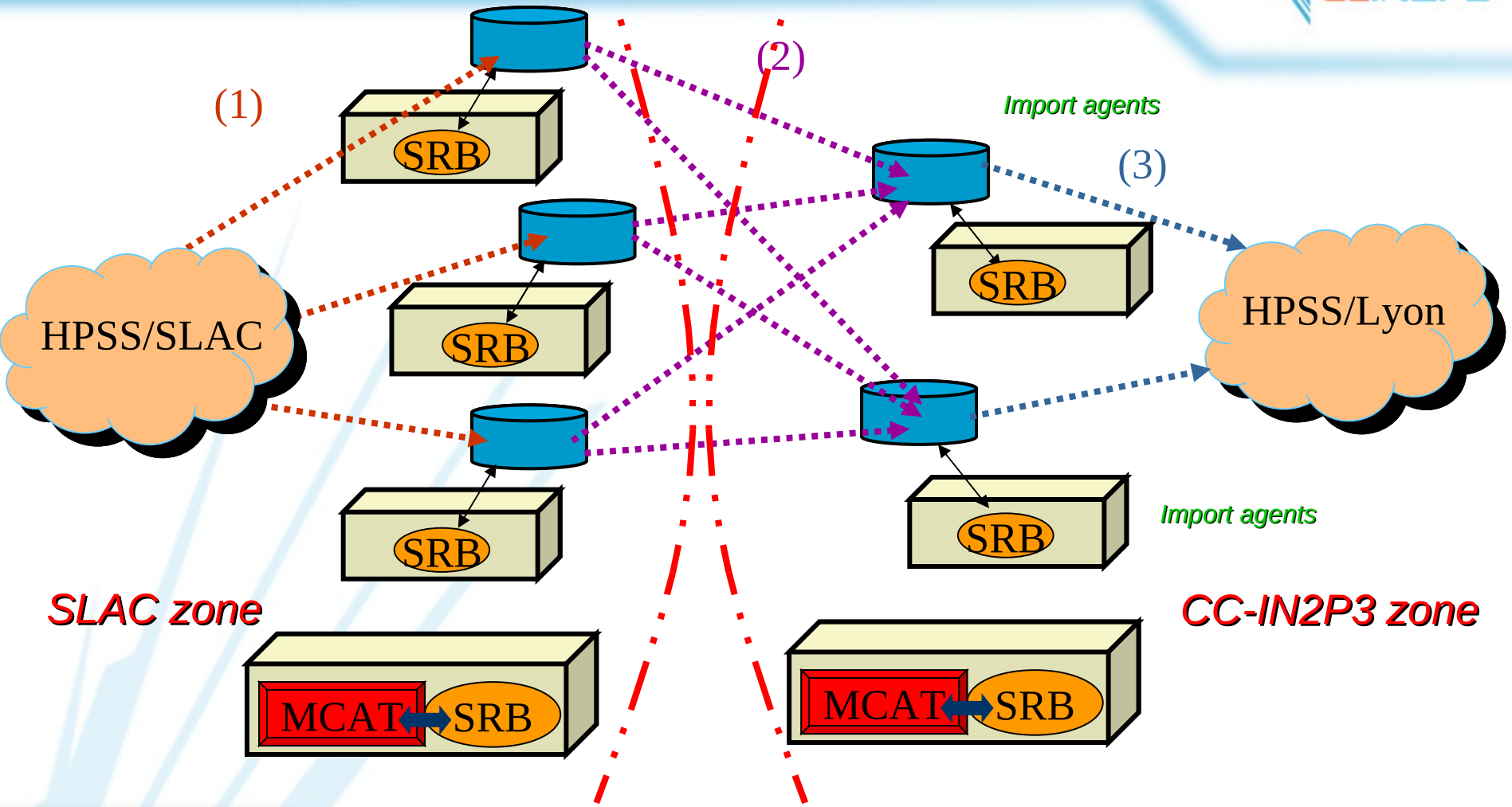
SRB hardware and software @ CC-IN2P3

- 15 SRB servers (130 TB of disk space):
  - Sun v480, v20z, v440 (Sparc III) and Thumpers x4500 (AMD Opteron).
  - OS: Solaris 9, Solaris 10 and Linux RHEL4.
- Interfaced with our Mass Storage System HPSS:
  - Code developed to handle automatic migration/purge of compound resources (disk cache/tape archive).
- MCATs database: using Oracle 11g.

Example in HEP: BaBar

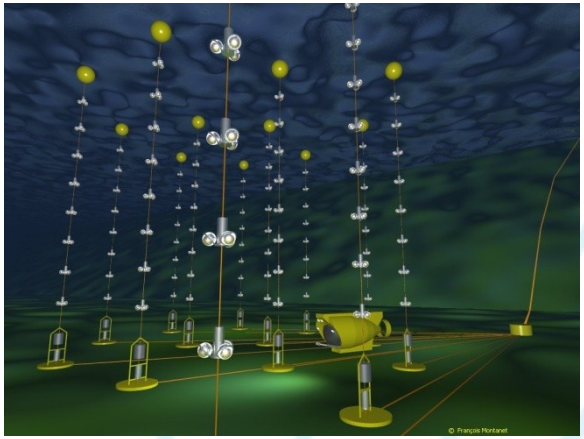
- Data import from SLAC to Lyon.
- SRB being used since 2004 in production.
- **Fully automated:**
  - New files created are registered in the SLAC catalog database.
  - Client application in Lyon: detection of files missing in the Lyon catalog database + transfer of these files.
  - Automated error recovery.
- Up to **5 TB / day** (max. rate observed).
- Usual rate: 2-3 TB / day (during production periods)
- 900 TB imported so far (since 2004), 2 million files.

Example in HEP: BaBar



Examples in astrophysics and astroparticles

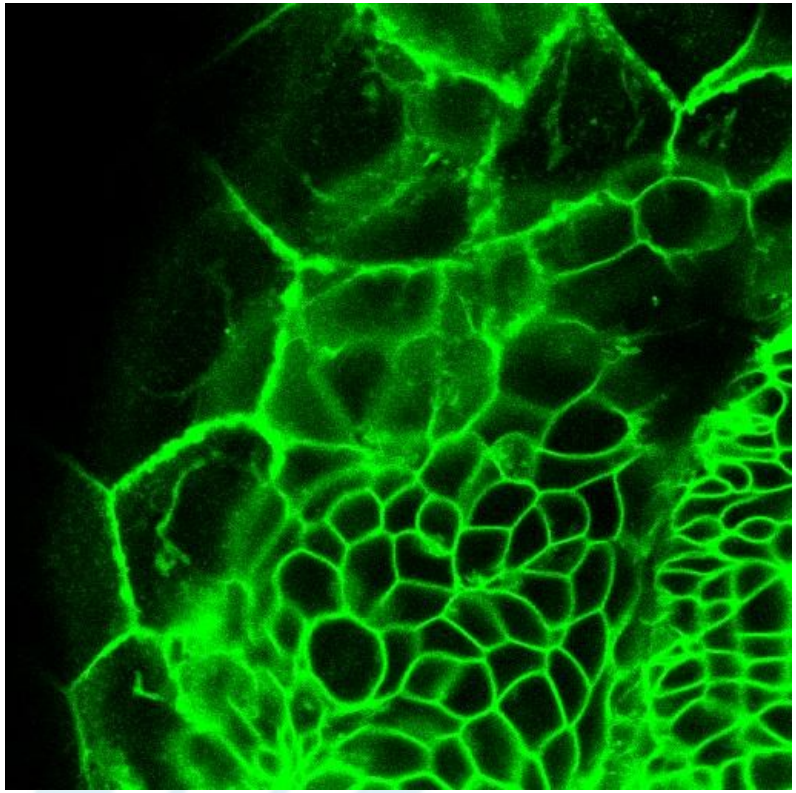
- Underwater: Antares



- in the pampa: Pierre Auger Observatory

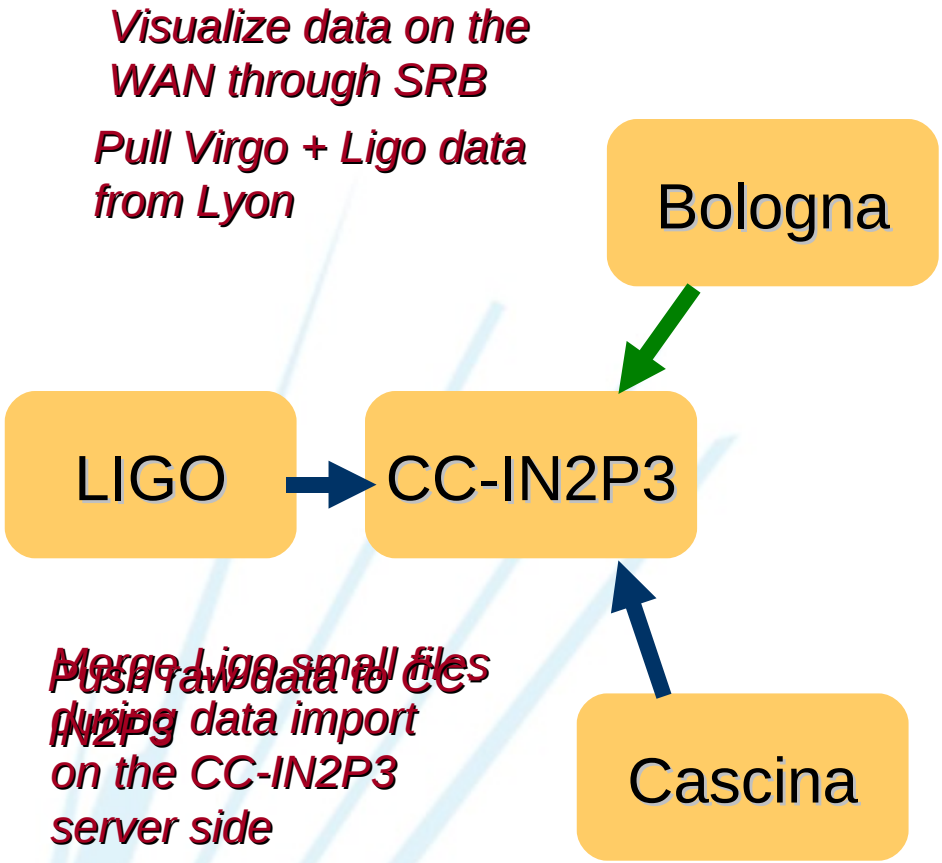
- At the top of the mountain: SuperNovae Factory in Hawaii

Example in biology: BioEmergence



- European projects involving 5 countries.
- Embryogenesis: zebra fish.
- 2 microscopes now (several in the future): amount of data could be huged (PB scale).
- Data pushed from the microscopes into the SRB.
- SRB integrated within their workflow.
- CC-IN2P3: core of the system.

Virgo



- Interferometer for gravitational waves detection (in production: 60 TB / y).
- Need for a reliable data distribution system.
- Distribute Ligo data (same experiment in the US) to the european sites: CC-IN2P3 and Bologna.
- Have been using bbftp so far.
- Test of EGEE tools not successful.
- SRB has replaced bbftp:
  - Bookkeeping system.
  - Interface with HPSS.
  - Ligo: interoperability.

SRB @ CC-IN2P3

- MCATs performance enhancement:
  - Reindexing made automatically on a weekly basis.
- Issues with Oracle performances:
  - Some oddity in the way Oracle optimized requests.
  - Request analyzis done on all the MCATs on a daily basis.
- Database is one of the key component of the system.

  ➔ Now OK: Oracle servers dedicated to SRB.

SRB prospects @ CC-IN2P3

- More than 10 projects using it:
  - SRB: critical part of their computing system.
  - Clients on Linux, Mac OSX, Windows, Solaris, AIX (Blue Gene) from Europe, USA.
  - Stable and robust.
- Daily traffic very important:
  - Hundreds of thousands of connections per day.
  - Some projects with more than 200,000 connections per day at peak rate.
  - Bandwidth peak rate: several Gbits/s.
  - Read/write peak rate > 10 TB/day.
- Will reach 2 PB of data referenced and handled by SRB in 2009 (10 millions of files registered).
- Slowly move to iRODS (will start in 2010).

Assessment of the SRB usage

- Many functionalities used …
- … but not all of them ☹, for example:
  - Extensible MCAT.
- Some developpements were needed:
  - Server side (monitoring, compound resource management, …).
  - Client side (data management application for BaBar, neuroscience etc…).
- Documentation (FAQ):
  - People can be lost by the level of functionalities
- GUI applications (eg: inQ) are fancy but dangerous:
  - Too easy ➔ can be used without being cautious.
- Also true for APIs, Scommands…

Assessment of the SRB usage

- Lack of control on the number of connections to the SRB system (but true for many computing software !):
  - Can be difficult to scale the system.
- Database has to be tuned properly:
  - Need for someone having DBA expertise.
- Sociological factors: fear to have data not under his control.
- Sometimes, lack of manpower on the experiment side in order to build customized client application.

Beyond SRB

- Storage virtualization not enough.
- For client applications relying on these middlewares:
  - No safeguard.
  - No guarantee of a strict application of the data preservation policy.
- Real need for a data distribution project to define a coherent and homogeneous policy for:
  - data management.
  - storage resource management.
- Crucial for massive archival projects (digital libraries …).
- No grid tool had these features until 2007.

Virtualization of the data management policy

- Typical pitfalls:
  - No respect of given pre-established rules.
  - Several data management applications may exist at the same moment.
  - Several versions of the same application can be used within a project at the same.

  ➔ *potential inconsistency.*
- Remove various constraints for various sites from the client applications.
- Solution:
  - Data management policy virtualization.
  - Policy expressed in terms of rules.

A few examples of rules

- Customized access rights to the system:
  - Disallow file removal from a particular directory even by the owner.
- Security and integrity check of the data:
  - Automatic checksum launched in the background.
  - On the fly anonymization of the files even if it has not been made by the client.
- Metadata registration:
  - Automated metadata registration associated to objects (inside or outside the iRODS database).
- Customized transfer parameters:
  - Number of streams, stream size, TCP window as a function of the client or server IP.
- … up to your needs …

iRODS

- **iR**ule **O**riented **D**ata **S**ystems.
- Project begun in January 2006, led by DICE team (USA).
- First version official in December (v 0.5).
- **Open source.**
- Financed by: NSF, NARA (National Archives and Records Administration).
- CC-IN2P3 (France), e-science (UK), ARCS (Australia): collaborators.

iRODS

- Based on the same ideas used in SRB.
- iCAT for iRODS ⇔ MCAT for SRB.
- But goes much further:
  - Data management based on rules build on the server side.
  - System can be fully customized without modifying any single line of the iRODS code.
  - Write your own services by adding your own modules.
  - Virtualization of the data management policy.
  - Logical name space for the rules:
    - Clustering in sets of rules.
    - Chaining the rules in a complex workflow (with a C like language).
    - Versioning handling.

iRODS: the rules

- A rule (prefix *ac*) contains:
  1. Name.
  1. Condition.
  1. Function(s) call: other rule(s) or micro-services.
  1. Recovery in case of error.
- A micro-service (prefixe *msi*):
  - Does a given task, can rely on internal functionnalities of iRODS.
  - Standard interface provided for the standard fournie micro-services.
- Rule example (called when a file is removed):

  acDataDeletePolicy*||*nop*|*nop

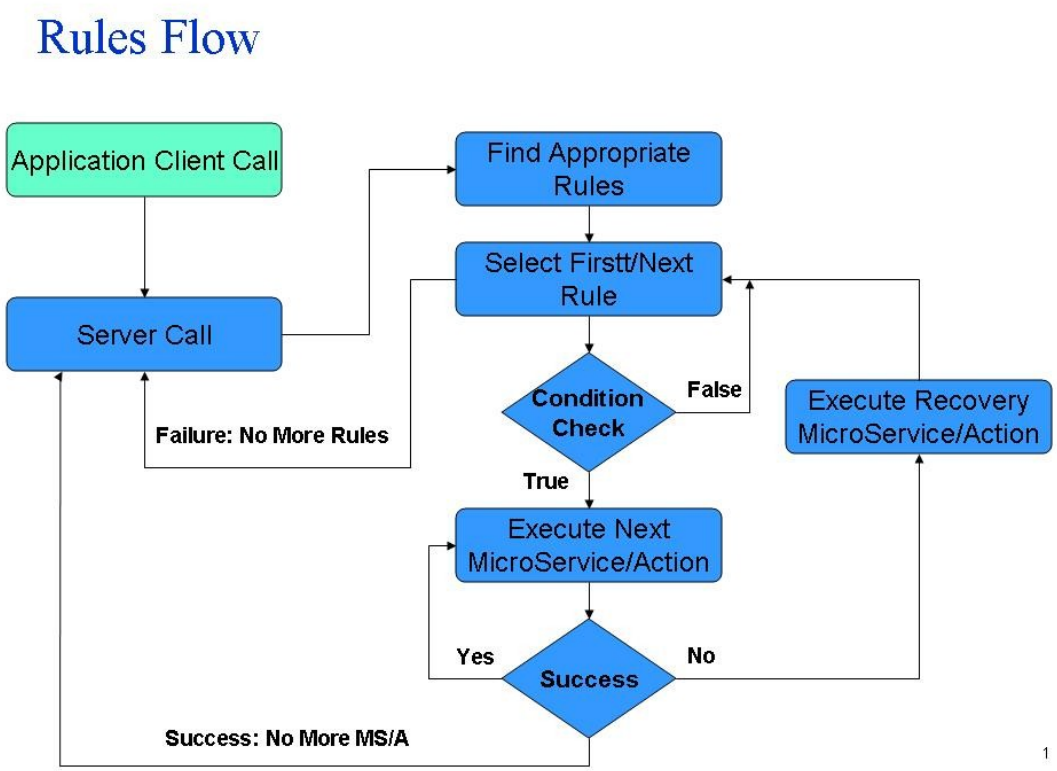
iRODS: the rules

- Last rule modification:
  - Prevent deleting data in /in2p3/RealData:
    acDataDeletePolicy|$objPath like /in2p3/RealData/*|msiDeleteDisallowed|nop

- Chain of micro-services that can be included when putting a file into the system:

```
myTestRule{
        *B = /temp/home/rods/test2;
        *C = /temp/home/rods/test3;
        msiDataObjOpen(*A,*S_FD);                       // ouverture de test1.
        msiDataObjCreate(*B,null,*D_FD);                // création de test2.
        msiDataObjLseek(*S_FD,10,SEEK_SET,*junk1); //  lecture des octets de test1 de 10000 octets à partir du …
        msiDataObjRead(*S_FD,10000,*R_BUF);             //  …. 10ème.
        msiDataObjWrite(*D_FD,*R_BUF,*W_LEN);       //  écriture du contenu dans test2.
        msiDataObjClose(*S_FD,*junk2);                  //  fermeture de test1.
        msiDataObjClose(*D_FD,*junk3);                  //   fermeture de test2.
        msiDataObjCopy(*B,*C,null,*junk4);              //  copie de test2 dans test3 dans la ressource courante.
        delay ("<A></A>") {
                        msiDataObjRepl(*C,demoResc8,*junk5);   // réplication après un délai de test3 sur un autre type de
stockage.
                        msiDataObjChksum(*C,*Operation,*ChecksumRes);     // checksum de tous les réplica de test3.
         }
        msiDataObjUnlink(*B,*junk6);                    // effaçage de test2.
}
INPUT *A="/temp/home/rods/test1",*Condition="COLL_NAME = '/tempZone/home/rods/loopTest'",*Operation=ChksumAll
```

iRODS: the rule engine



Allows the processing of rules:

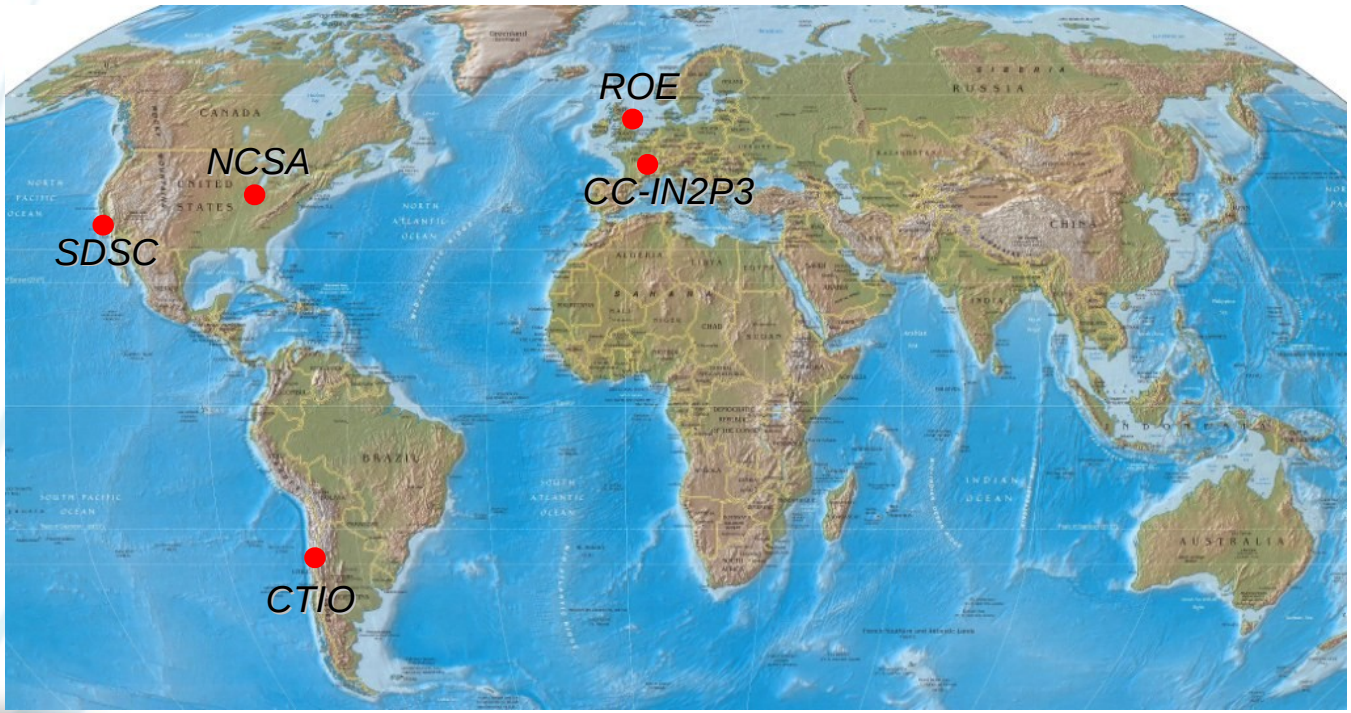
- *delayed execution*
- *interaction with external services.*

iRODS developments @ CC-IN2P3

- Tests scripts of the APIs through the shell commands.
- Stress tests.
- Micro-services:
  - Host based access control.
  - Tar and untar of files.
- Load balancing and monitoring system (next release).
- Interface with Mass Storage System.

iRODS test beds

- With KEK (Japan): data transfer at high rate.
- LSST (telescope in Chile, 2015): data replication and workflow.



Production iRODS @ CC-IN2P3

- iRODS: 6 servers with Oracle backend,120 TB.
- Adonis (Arts & Humanities projects):
  - > 10 TB of data registered so far.
  - > 2 millions of files.
  - Accessed from batch farm.
  - Micro-services needs to be used for one project (long term data preservation):
    - Data archived in CINES (Montpellier) and pushed to Lyon (tar files):
      - Automatically untar the files @ CC-IN2P3.
      - Automatically register the files in Fedora (external system).
      - Data integrity check also done (checksum).

Production iRODS @ CC-IN2P3

- Biomedical applications.
- More projects (biology, astrophysics) should start with iRODS this year.
- Expecting 100 TB of data registered in the system by the end of 2009.

iRODS assessment

- Highly scalable for data management tasks.
- Many features and customization: very attractive to potential users.
- Already a large community interested by iRODS growing world-wide in various fields, for example:
  - Long term digital preservation.
  - Astrophysics.
  - Biology.
- Considered as a key software at CC-IN2P3.
- Already stable and mature enough for production.
- DICE team very reactive in order to solve problem and open to include new features.
- More documentation needed, rules syntax could improve.

Acknowledgement

- http://cc.in2p3.fr/
- http://irods.sdsc.edu/index.php/Main_Page
- Thanks to:
  - DICE research team.
  - Pascal Calvat, Jean Aoustet.
  - Wilko Kroeger (SLAC): BaBar.
  - Adil Hasan (University of Liverpool).
  - Yoshimi Iida.