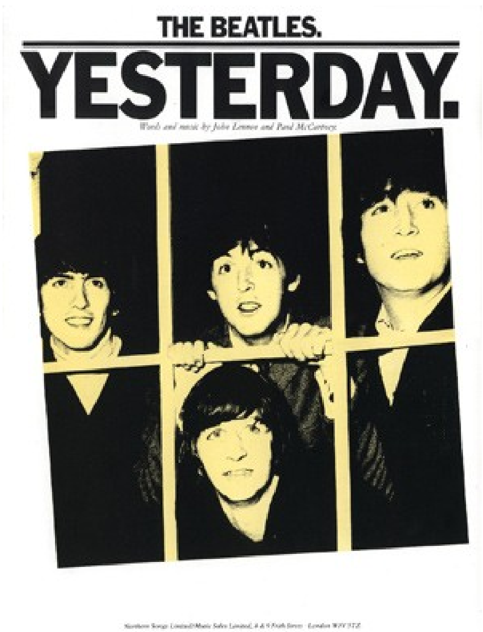
DE LA RECHERCHE À L'INDUSTRIE

Future of IO: a long and winding road

Philippe DENIEL (philippe.deniel@cea.fr)
CEA, DAM, DIF, F-91297, Arpajon, France

www.cea.fr

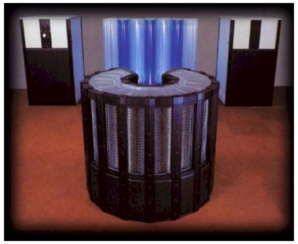
Yesterday



A look at the rear-view mirror

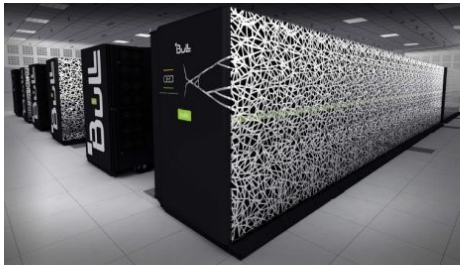
Small local FS on Cray machines

- Cray machine were MPP machines
- Logically equivalent to a single node with many cores and a large memory
- Storage was local and attached to the MPP machines
  - **All Cray machines at TERA produced 80TB in their whole life**



Then came SMP clusters

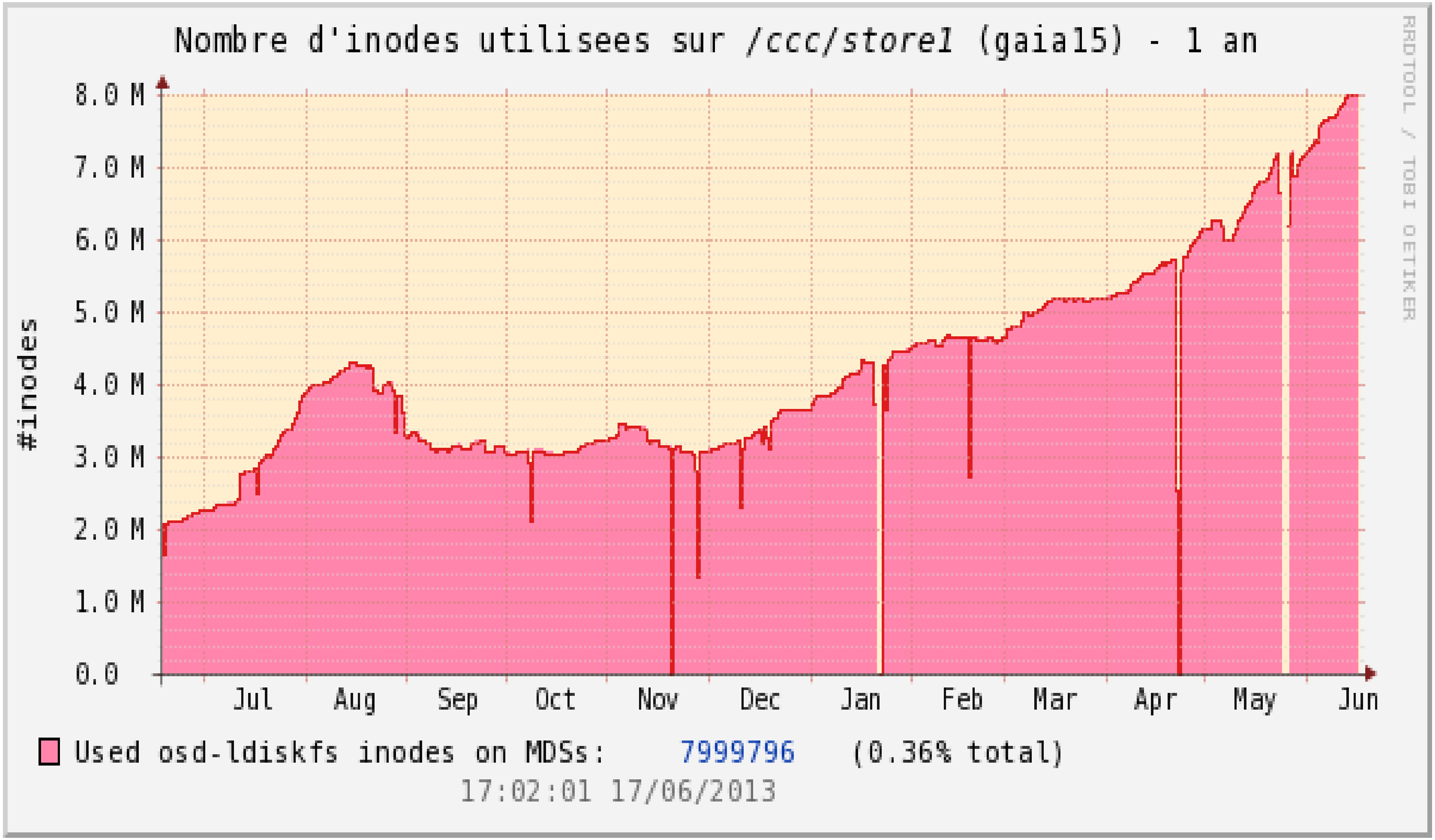
- Many small machines federated by a high performance network to build a big one
- File System is accessible consistently from all nodes in the cluster
- Parallel File System were born with SMP
  - **Volume stored (early 2000): a few petabytes**
  - **Production (early 2000): ~100TB/year**



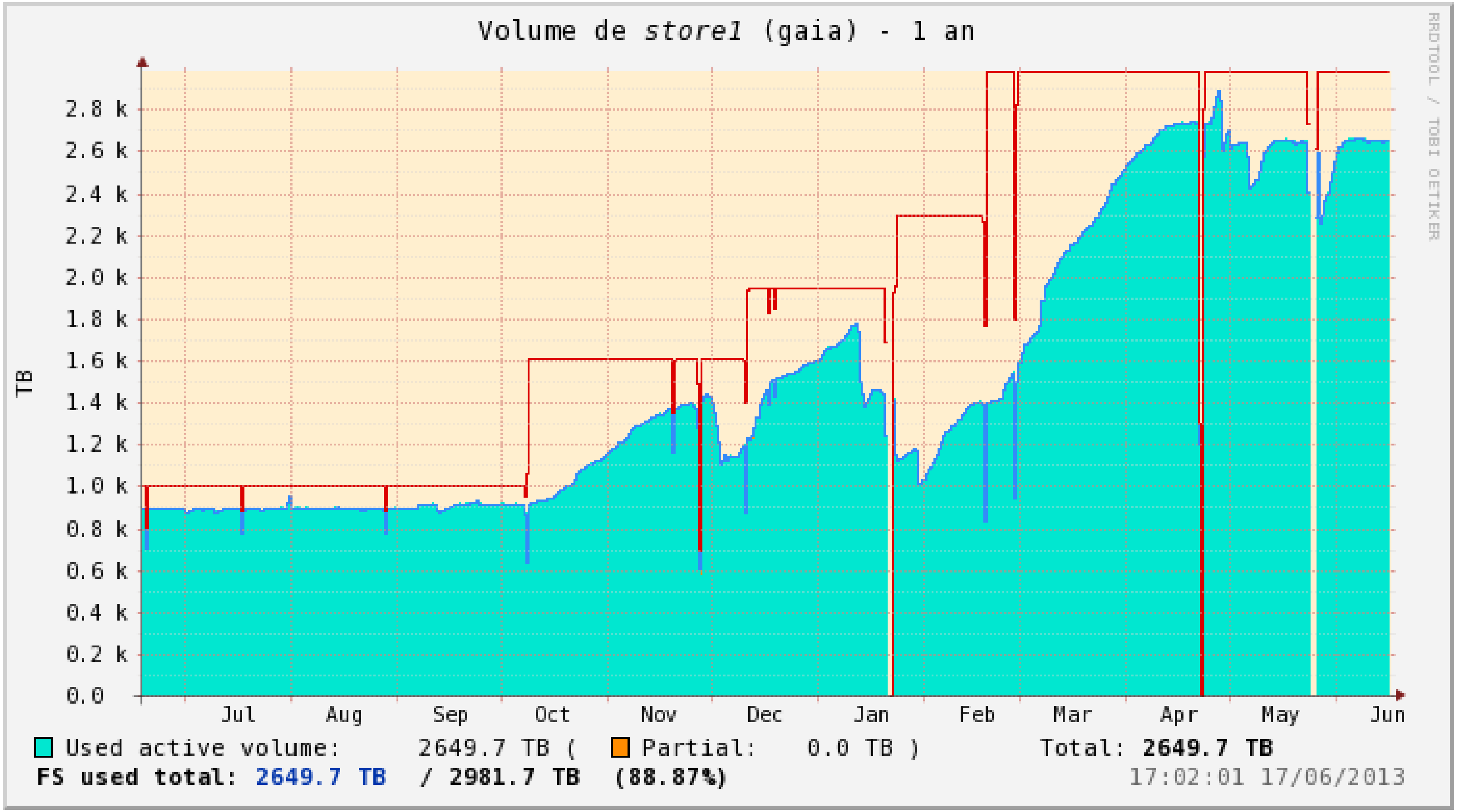
... and even bigger SMP clusters

- SMP clusters prove to be reliable supercomputers
- They become bigger and bigger and stored more ar
- Lustre and GPFS became *de facto* standards
  - **Volume stored : ~40PB**
  - **Production ~6PB/year**

STOREDIR@TGCC: Volume (from last COMUT)



STOREDIR@TGCC: inodes (from last COMUT)



Today

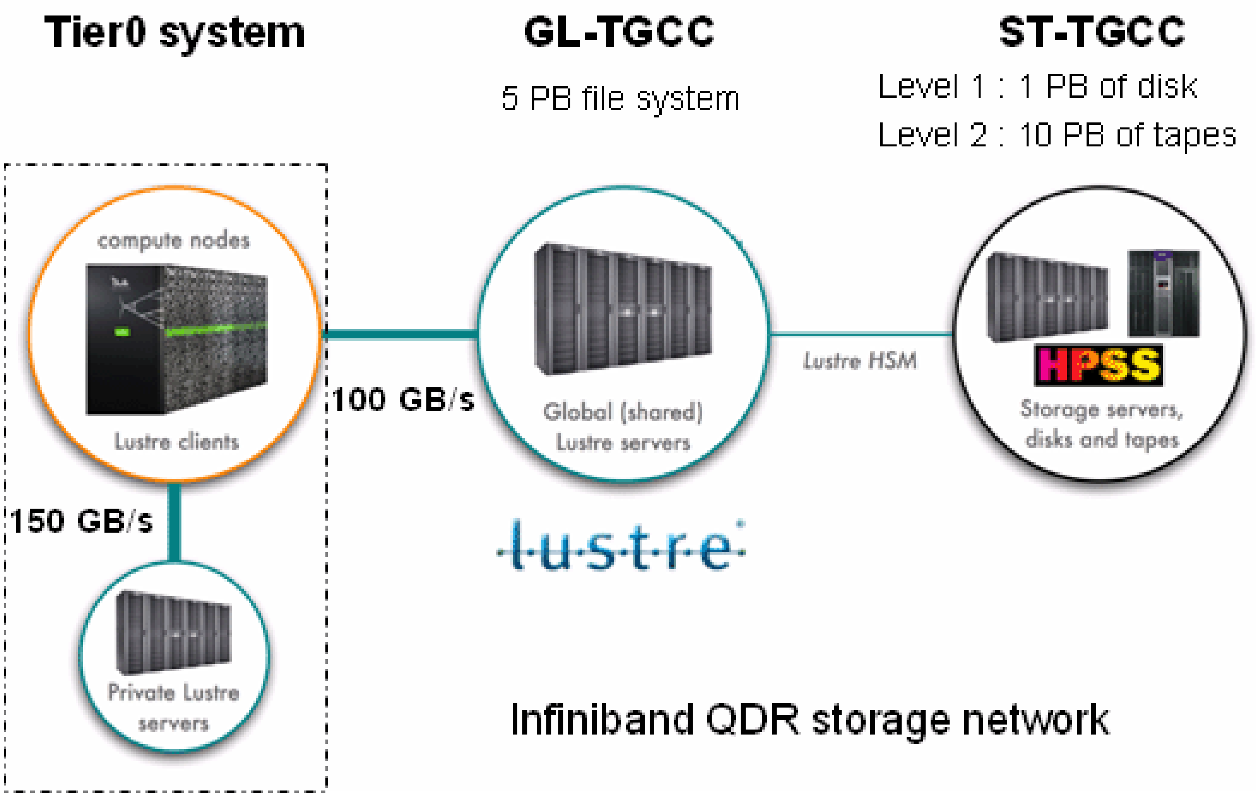
A data centric architecture

GL-TGCC

- Fast access to data,
- Gateway to files
- post-processing

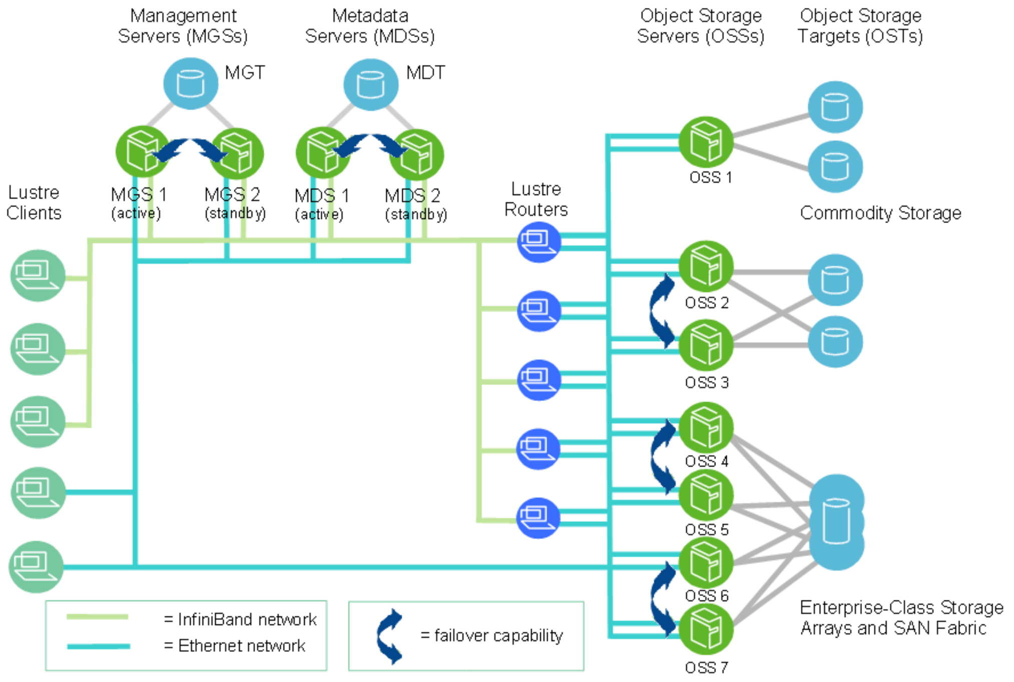
ST-TGCC

- Long term storage
- Keeps simulation results

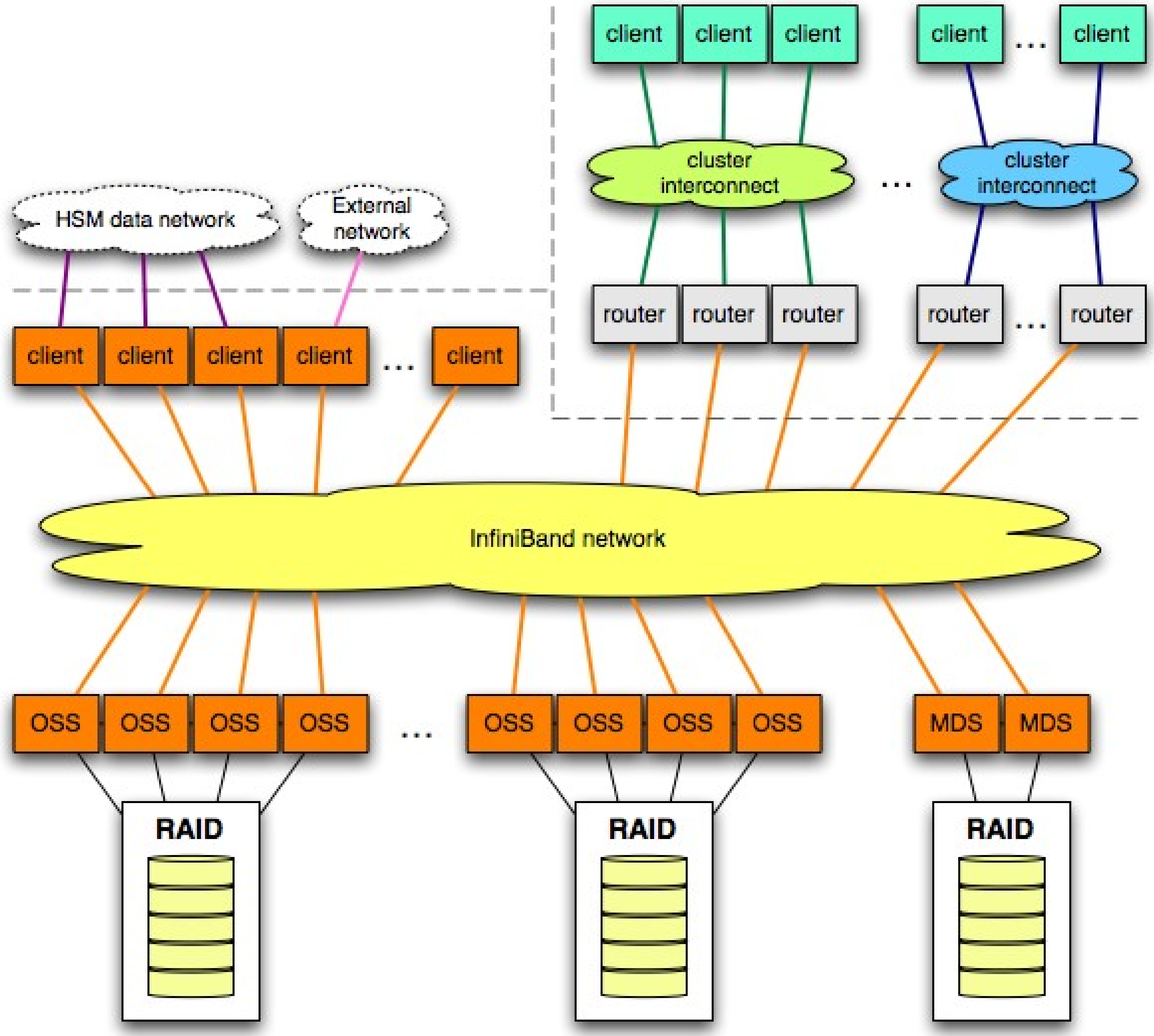


Parallel Filesystem

- Developed by Intel Data Division(formerly WhamCloud), with support form international labs and organizations
  ·OpenSource Product. Half of Top500 machines use it
  ·CEA is part of the development.
- Components
  ·MGS (*Management Server*)
  ·MDS (*Metadata Server*)
  ·OSS (*Object Storage Server*)
  ·Routers
  ·Clients
- GL-TGCC
  ·Two filesystems : *work* and *store*
  ·Interconnexion Infiniband QDR
  ·1x *metadata cells*
    - 2x MDS, 1x DDN SFA10K
  ·10x cells I/O
    - 4x OSS, 1x DDN SFA10K



GL-TGCC ARCHITECTURE (data-centric)



Large FS are cool but won't scale forever

Keeping Large FS consistent is expensive and complex

- The complexity grows drastically as the number of stored object increases
- There is a limit to the number of clients a server or a cluster of servers can manage
- File systems naturally creates dependencies between objects
  - hardlink makes it possible for a file to exists in more than one directory
  - Symlink's usage can result in "file systems" mazes
  -

Scalability issues

- Todays feedback shows that filesystems with billions of objects are unstable
- Such a creature is hard to administrate with Today's technology

Big Files are beautiful, small ones are not

- Small Files Problem is definitely the first plague of ~~Egypt~~ HPC
- MPI makes program running on ~1000 to ~100 000 nodes a reality
- Each node can produce per-process files
- Interesting data resides in all files
  - File are spread on many different resources with an atomic, unavoidable cost for accessing each of them
  - Situation becomes nightmarish when tapes are involved
    - accessing 1000 files means mounting dozens of tapes => long delays

POSIX sucks

Locks

- POSIX was born in a world with no threads, only processes, treated as atomic lock owners
  - Thread 1 has a lock
  - Thread2 requests another lock
    - No new lock is created
    - Lock owned by Thread1 is modified by Thread2's request
- All locks held by a process are dropped any time the process closes any file descriptor that corresponds to the locked file, even if those locks were made using a still-open file descriptor.

POSIX does not fit parallel file systems

- The same structure mixes different types of metadata
  - “last offset in file” (aka st_size) is a pure file metadata
  - “space used” (aka st_blocks) is a storage related metadata
  - Modern FS handle storage metadata separately
- POSIX “Exactly Once Semantics” (EOS) do not fit distributed parallel FS

Corner cases

- POSIX is full of “corner cases” where actual behavior is not what people would expect
- **<u>POSIX TRIVIA:</u>** What's happening if you call rename() with both file arguments referring to the same inode ?

From Today to 2020

IO Challenges for 2020 : the IO Proxy

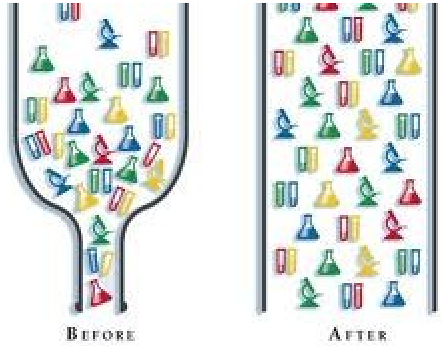
Manycore processors => reduction of ratio memory/core

The operating system will have less memory buffers for its own needs

- Less room in the OS for the file systems
- Even the TCP/IP network stack may become too expensive and be replaced by lower level but faster paradigm (like RDMA)

Kill the bottleneck!!

- Need for mechanisms to manage larger data without generating bottlenecks
- The former approach used in SMP is not valid anymore and would lead to an explosion of the number of clients



Data View: Study's closer look at data

Data usage of compute clusters

- In most cases, a large machine run many studies (sets of correlated jobs) at the same time.
- Each study has its own set of required input and produces its own set of data.
- It's useless to expose everyone's data to everyone, just show what's required

- The Data View is a consequence of the notion of Study
  - We can define data views for study
  - Each view contains subset of data to be used by the study, and has related areas to store results
  - The data view is tied to the study
  - Each study is agnostic to the others
  - Intersection of different data views is made of read-only data

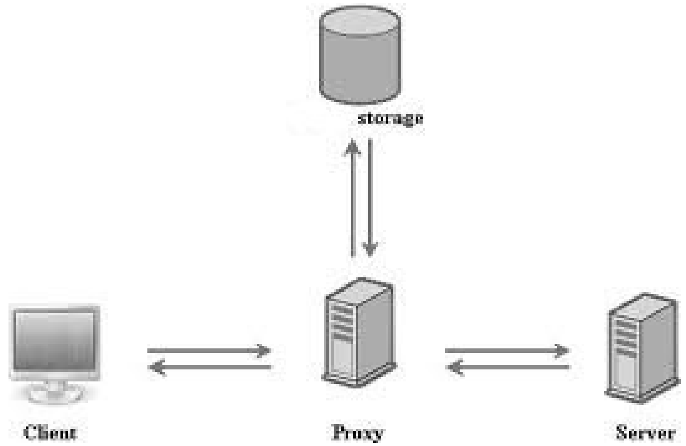
Data View means IO Proxy

The data view will be served by a dedicated IO Proxy

- The proxy is the only “data view provider” to the study

- The proxies are the only actual clients of storage resources

- The proxy is the natural place for optimization based on hints provided by upper layers (IO Libraries and simulation code)
    - Impact on data cache policies (keep only what is tagged as essential)
    - Impact on metadata cache policies (do not flush what will be used soon)



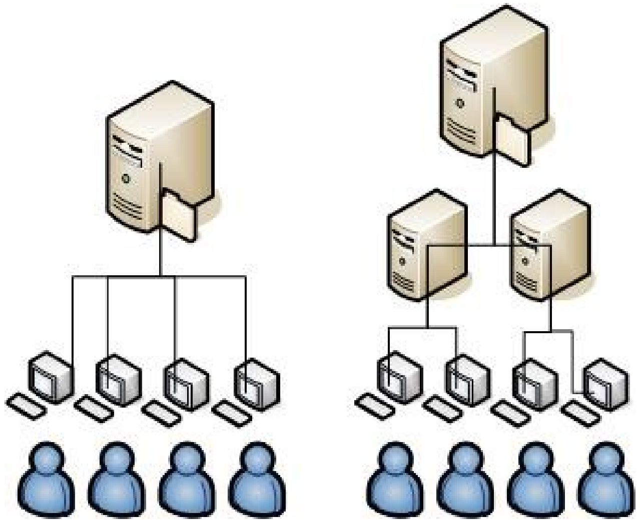
IO Proxies kill bottlenecks

The proxy algorithm is a bottleneck killer

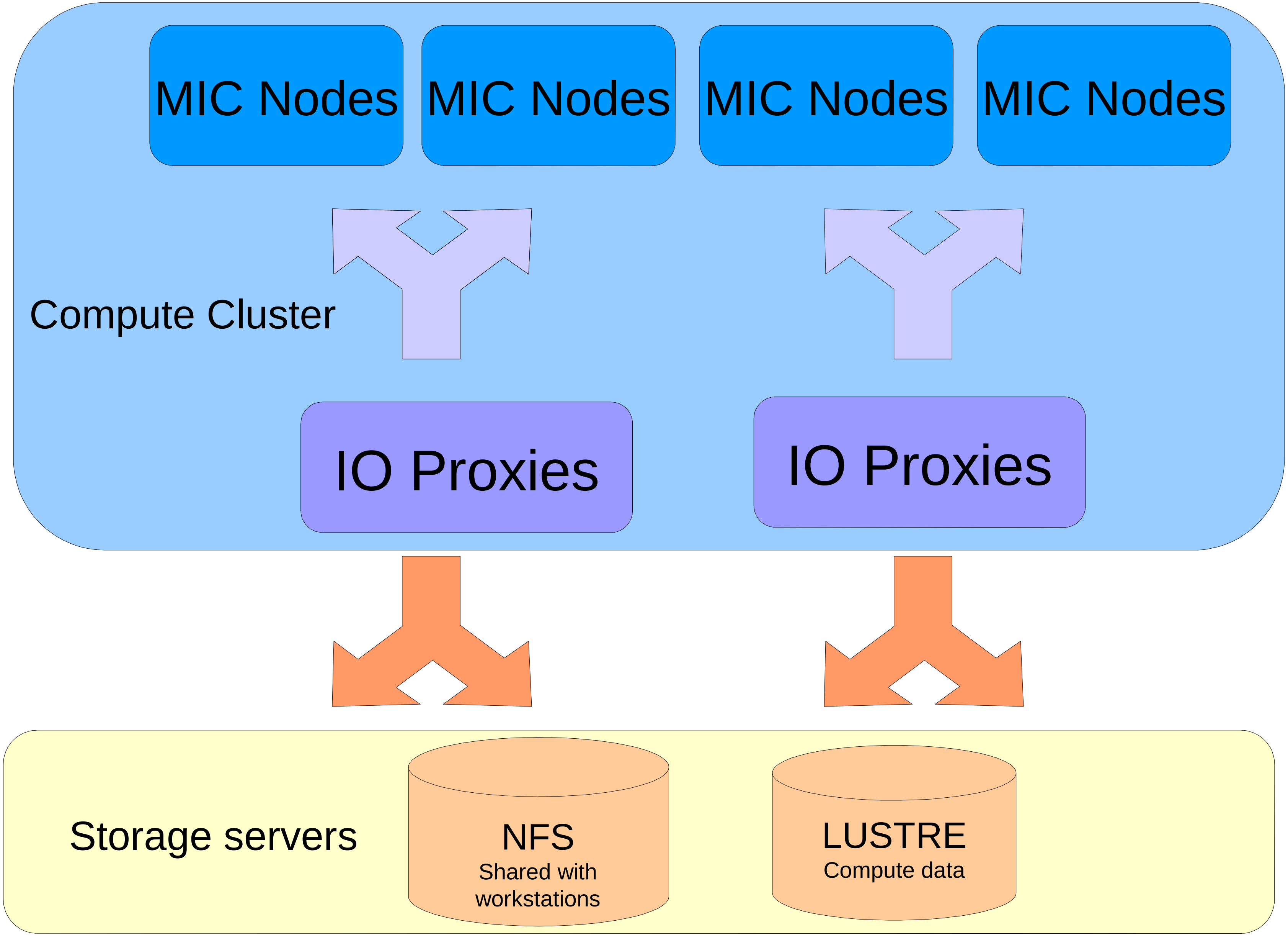
- Proxy brings more flexibility to clustered architecture
- IO streams can be controlled at the Proxy's level

The proxy approach has been successfully used in situations involving lots of clients

- The HTTP across the Internet makes intensive use of proxies, most of them invisible to the end user (your web browser at your workstation)
- Sending mail over the Internet goes through multiple SMTP proxies



IO proxies inside future architecture



IO Proxies

IO Proxies are internal to the future compute machine

- Single path for computes nodes to access data
  - Lustre Filesystems
  - NFS remote servers

IO Proxies as “fuse”

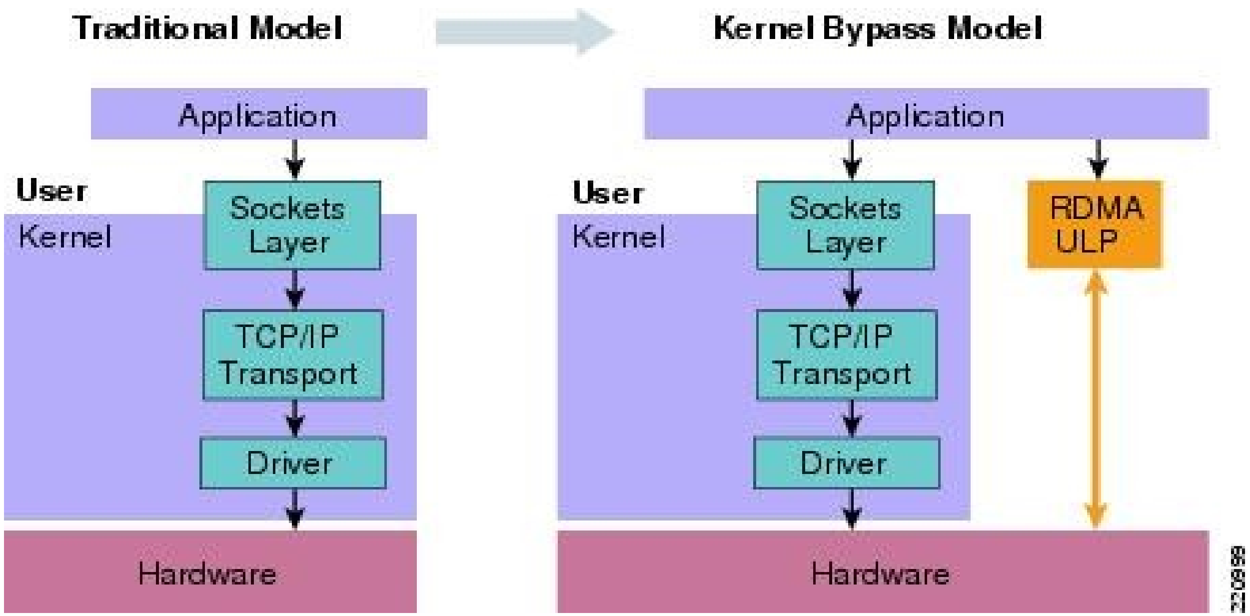
- A single “evil” command can easily collapse a storage system
- A “rogue study” will only mess its own proxy
  - Use of internal metrics will help identifying toxic behaviors
  - In such a case, the proxy would slow pause, pause or even stop to protect the back-end
- A major failure on the IO Proxy will crash it, preventing the trouble to contaminate the whole machine

Efficient networking: the RDMA transport layer

RDMA: Remote Direct Memory Access

A machine allows another to write directly in a few windows in its own memory

- Simpler implementation compared to TCP/IP
- A *de facto* standard via Infiniband, iWARP and RoCE technologies
- LAN dedicated but fast network model
- Bypass several OSI layers to optimize performances



File Server's protocol: the 9P Protocol

IO Proxies will export data to compute nodes via the 9P Protocol

- 9P was originally designed by Bell Labs for Plan9
  - A standardized protocol
  - A living protocol
    - Several enhancements in the protocol since its birth
    - Latest is 9p.2000L (designed for Linux)

Through its design, 9P fits IO Proxy's requirements

- 9P is a very lightweight protocol
- 9P is fast to interpret
  - Use little-endianess making XDR-like marshaling unnecessary
- 9P is buffer oriented, which fits well RDMA transport
- 9P make zero-copy based implementation easy and requires less memory
- 9P has all you need to implement full POSIX semantics
- 9P is quite complete
  - Distributed flocks are supported
  - Extended attributes are supported

New storage related API

File private locks

- A new model of lock (non compliant to POSIX), enthusiastically adopted by the community. Integration in the kernel's mainline is under progress
- The file owns its locks
- Locks are revoked as the last opened file descriptor is closed
- Compliant with process using many threads

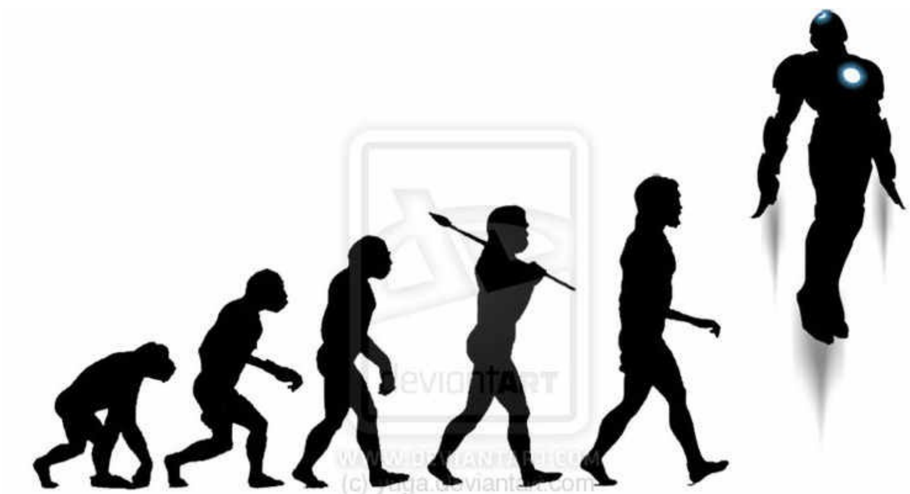
EIOWG/E10

- The Exascale IO Work Group tries to define new models and API for exascale compatible IO
- EIOWG's creed: Database had big successes because they were well standardized, let's do the same with data storage
- The Work Group tries to bring up new model and new API to comply with exascale's needs for storage accesses.

The File System: a model from the Past

- File System will probably continue to exists as a way a user can see and manage its information
  - File system structure and dependences won't be present from back to top
  - File system will be kept inside larger and more adapted objects
  - New paradigms to be created to replace file systems

Beyond 2020

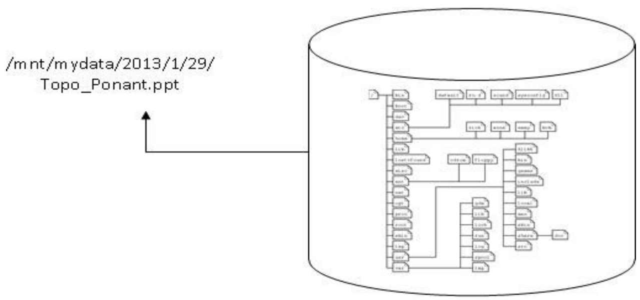


2020: A (storage) space odissey

Today's models keep data in file systems

Structure based on files and directories:

- Strong dependencies result of this structure
- The distribution of such a structure is complicated
- Addressing by path is inconsistent (you can rename)

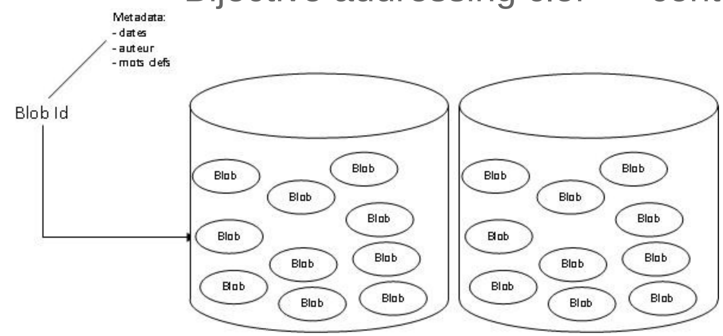


Storage blobs: data accessed via an already known key

Data are kept in weakly typed containers : the **storage blobs**

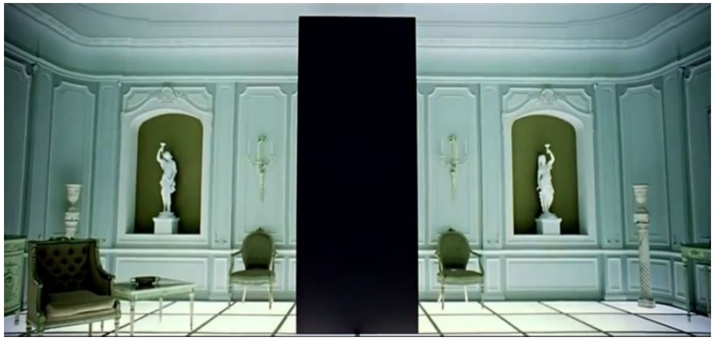
Containers and independent, billions of them exist, and they can be easily distributed

Bijective addressing clef => content



Analogy : Youtube videos are " video blobs" acessed in a key/value way

Example : Etienne Klein's video on the breach of symmetry is referenced by key "R-kLlXKjgnI".
its URL is http://www.youtube.com/watch?v=R-kLlXKjgnI



Blobs in action

Blobs are « storage baskets »

Blobs are polymorphic object, unaware of their contents. They can be stored on various media: tapes, disks, flash memory, SSDs…

Their simple structure makes eases

- Replication across different systems
- reliability

Each blob is addressed by a unique key stored in a massively distributed database

Bijective relationship in-between a key and a blob

- This very simple schema removes constraints

Needs a specific database

- with large DB storage
- massively distributed
- fault tolerant / redundant

A simple and innovative approach

- To be used intensively in HPC environment
- Generic and usable in enterprise marker (Cloud, Big Data)
- Innovative use of the media
- Strongly correlated with new flavors of Databases (Hadoop)

Why storage blobs are cool

In a HPC context

Dual speed storage

The high performance storage used by supercomputer to primarily produce data is expensive and has a limited size.

Data are stored in blobs for
•Free expensive resources
•Secure the data

Store more, for less money
•Blobs rely on inexpensive media
•They provide massive storage

Storage Blobs to replace HSMs/Backups

A disk to tape blob migration is quite simple.

Blobs are not interdependent, they can be accessed concurrently
• Better service for the end user
• Better Volume/Throughput ratio

Beyond HPC

Collaborations

A versatile model that fits many domains known as big data producers (biology, climatology, astrophysics, medecin, computer simulation)

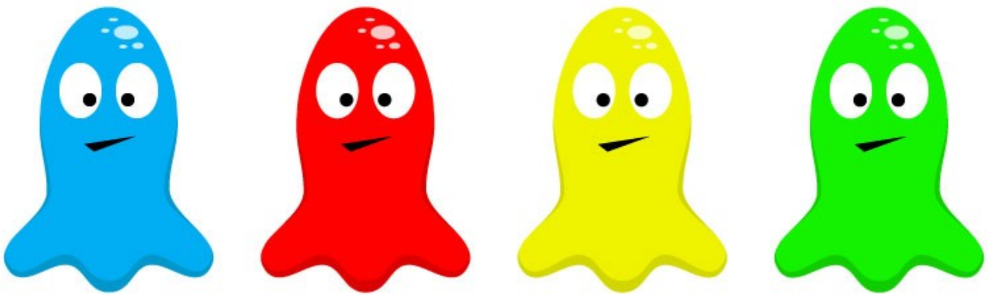
Fits the expected explosion of numerical storage in scientific and non scientific domains

A model living in the Open Source World

Blobs will rely on Open Source products

Collaboration to be set up with the industry (storage clouds) and Open Source community Collaboration (Hadoop).

A model with strong correlation to Big Data



Questions ?

Commissariat à l’énergie atomique et aux énergies alternatives
Centre DAM-Ile de France | 91297 Bruyères-le-Châtel Cedex
T. +33 (0)1 69 26 40 00 | F. +33 (0)1 69 26 70 86

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019

Direction des applications militaires