Installation, configuration, testing and optimization of a dCache Storage Element in a LHC Tier2.

Giacinto Donvito, *Vincenzo Spinoso*

(INFN-Bari)

Giorgio Maggi

(Politecnico e INFN-Bari)

Outline

- **Introduction to the problem**
- **Starting from the bottom:**
  - Raid technologies
    - Test and final configuration
  - Operating Systems
    - Linux Kernel
    - SolarisOS and ZFS
  - Data Pools configuration
  - dCache characteristics
    - Used configuration and optimization
- **Conclusion**

(CMS) LHC Tier2 in 2008

- Roles:
  - ➨**Monte Carlo Production** ➨**Analysis**
- CPU:
  - **1MSi2k**
  - **~ 500 batch slot**
- Storage:
  - **Size: ~ 200TB**
  - **WAN transfer (T1<->T2):**
    - **> 50MB/sec import**
    - **> 10 MB/sec export**
  - **LAN transfer (Storage<->WN):**
    - ***> 1GB/sec aggregate read from WNs***
    - **> 10 MB/sec write to storage**
    - **Interactive data read/write access from the user (mainly belonging to the "Tier2 local community")**
      - **Size is small compared to other kind of data but requires low latency access**

Preliminary thoughts

- A “DAS-like” solution has been considered
  - **No SAN configuration are shown here**
- try to exploit the capabilities of each server,
  - several different servers with different performance characteristics
- try to focus on reliability of the servers (not only on the performances)
- look for a vertical configuration that starts from the hardware and goes up to the service

Starting from the bottom: which RAID??

- With the disk size today, RAID5 seems not enough resilient
  - **We always try to use RAID6**
- We tested both the hardware and software (Linux Kernel 2.6.x) RAID
  - **SW RAID:**
    - Good reliability
    - Easy to manage
    - Poor performance
  - **HW RAID:**
    - Good reliability
    - Better performance related to SW RAID
      - If WriteCache is enabled (BBU needed!)
    - Monitoring and managing each controller using specific CLI provided by vendor

Starting from the bottom: which FileSystem??

- We did a lot of test and production experience with few file-systems:
  - **ReiserFS**
    - Good Stability
    - High load -> bad performance
  - **Ext3**
    - Good Stability
    - High load -> bad performance
  - **XFS**
    - The stability depends a lot on the kernel
    - Good performance
  - **ZFS**
    - Good stability
    - Easy to manage
    - Provide good RAID functionalities
    - Good performance

Starting from the bottom: which SO??

- We have tried to use as much open source software as possible
  - **But ... the level and the period of "support" is important!**
- We tested and used in production several OS:

- **SLC4**
  - Bit better stability than SLC3 -> not enough!
  - High load -> bad performance
- **SL5**
  - Not enough stability
  - Increased performance
- **Ubuntu 6.10**
  - Poor stability
  - Good hw support
  - Good performance
- **Debian stable "etch"**
  - Good stability
  - Good hw support
  - Good performance with XFS

Starting from the bottom: kernel optimization

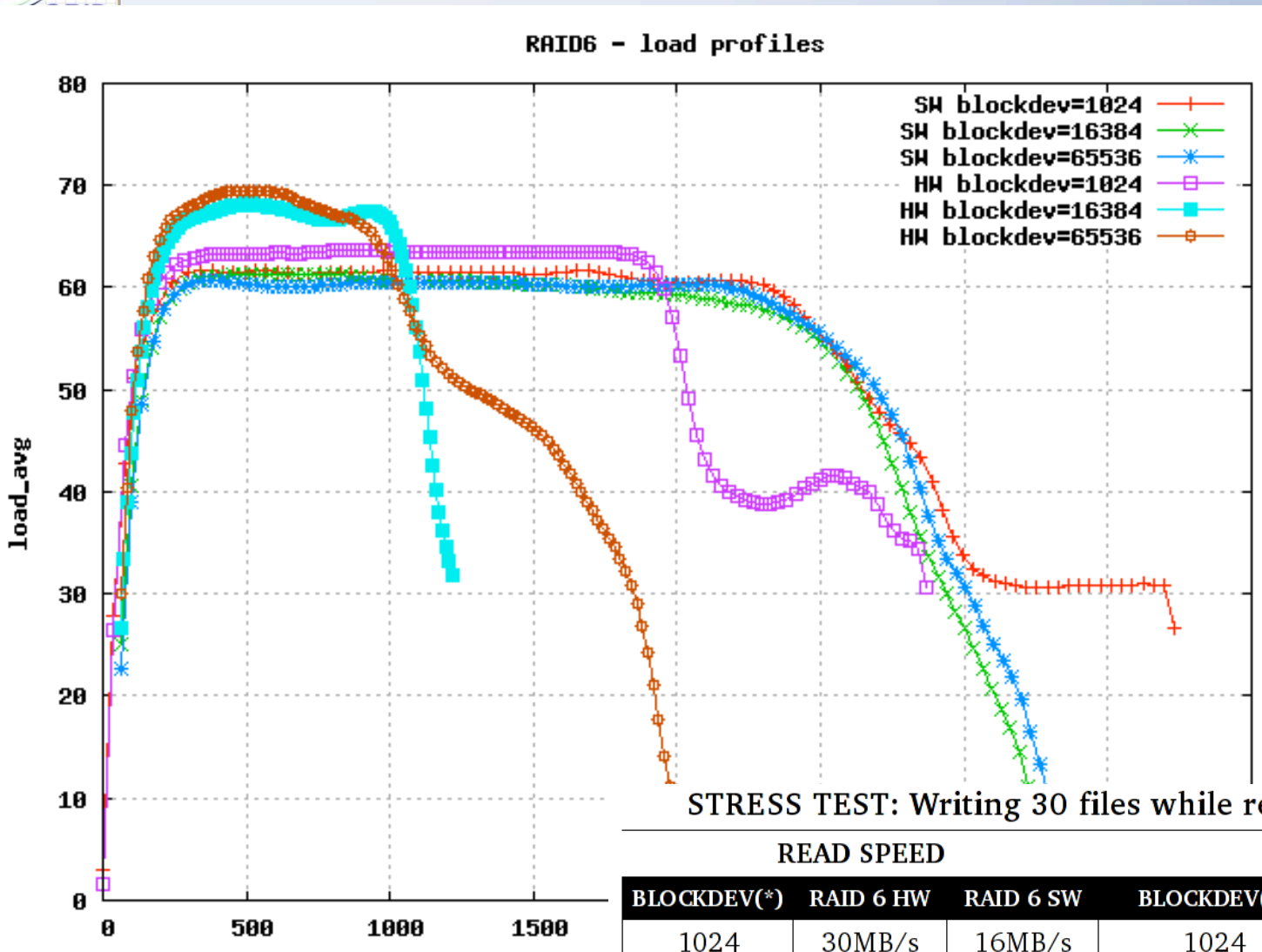
- Avoid “extreme” kernel tuning (the goal is always preserve the reliability)

- We started looking for some similar experiences already reported (also outside LHC environment):
  - **We have used, in all this tests, Debian stable “etch” (kernel: 2.6.18-5 x86_64)**
  - **Not seen so much improvements for most of the reported parameters**
  - **We focused mainly on three parameters (as we found that the performance and reliability are highly affected):**
    - blockdev: with few concurrent access, a greater value of the parameter gives a faster performance
    - nr_requests: (# of queue I/O request in the queue of the kernel): if it is too big the kernel memory gets overloaded
    - I/O scheduler: it seems that the “deadline” is the the useful for high i/o rates. We found it very stable

Local tests description

- Trying to execute the tests in the worst conditions that we see in production:
  - **Lots of concurrent accesses highly I/O demanding**
  - **Both read and write operation in the same time**
    - 30 reads while 30 writes
  - **Using big (2Gbyte) files in the test**
    - This will overcome bias due to caching in RAM memory
  - **The test were executed changing "blockdev" and SW/HW RAID**
  - **The tests were executed using an home made bash script that takes care of creating files, running of the tests, keep note of the execution time and make the plot of load_average.**

Cross tests



STRESS TEST: Writing 30 files while reading 30 other files

READ SPEED			WRITE SPEED		
BLOCKDEV(*)	RAID 6 HW	RAID 6 SW	BLOCKDEV(*)	RAID 6 HW	RAID 6 SW
1024	30MB/s	16MB/s	1024	21MB/s	21MB/s
16384	50MB/s	19MB/s	16384	55MB/s	21MB/s
65536	31MB/s	18MB/s	65536	32MB/s	21MB/s

(*) readahead in 512-byte sectors: 16384→8KB

Thumper (SUN Fire 4500) and Solaris

- HW Spec: 2 Dual Opteron, 16GB Ram, 48 1TB SATA disks, 8 controller, 4 Gbit/s ethernet card in “bonding” mode
- SW Spec: SolarisOS 5.10
- FileSystem and RAID:
  - **No HW Raid implemented**
  - **ZFS: 46 disks configured using RAIDZ2 (two parity discs -> typical RAID6 behavior) in two different Zpools**
  - **We performed several tests before choosing the final ZFS configuration**
    - This configuration seems a good balance between performances and fault tolerance (we can survive to a failure of up to 4 of 46 disks)

Local performance (with 20 concurrent operations):

550 MB/s WRITE

660 MB/s READ

Thumper (SUN Fire 4500) and Solaris

- HW Spec: 2 Dual Opteron, 16GB Ram, 48 1TB SATA disks, 8 controller, 4 Gbit/s ethernet card in "bonding" mode
- SW Spec: SolarisOS 5.10
- FileSystem and RAID:
  - **No HW Raid implemented**
  - **ZFS: 46 disks configured using RAIDZ2 (two parity discs -> typical RAID6 behavior) in two different Zpools**
  - **We performed several tests before choosing the final ZFS configuration**
    - This configuration seems a good balance between performances and fault tolerance (we can survive to a failure of up to 4 of 46 disks)

No deep tuning needed in order to achieve these performances

Local performance (with 20 concurrent operations):

550 MB/s WRITE

660 MB/s READ

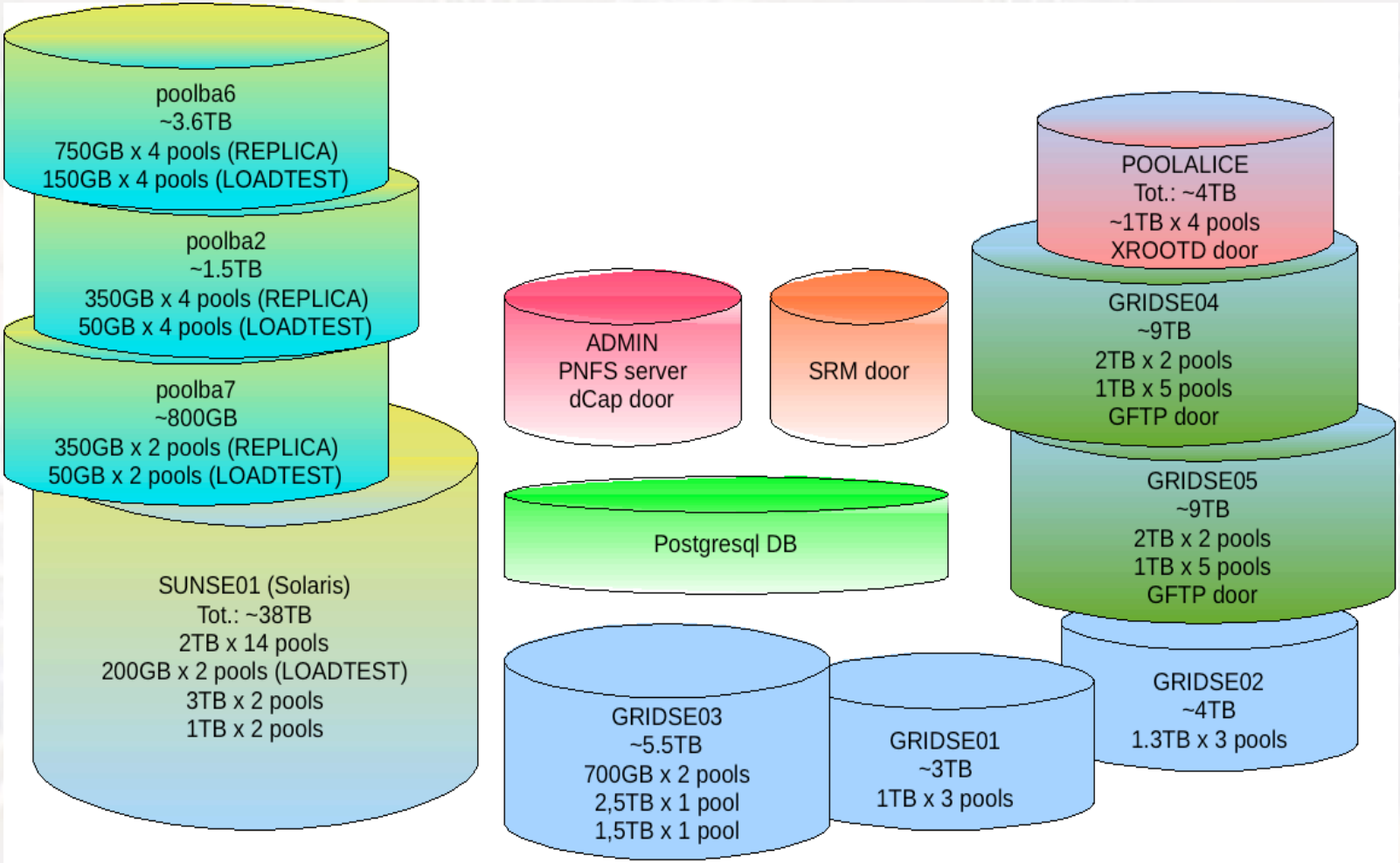
Storage Software

- Why did we use dCache as storage manager:
  - **Widely distributed especially in CMS community**
  - **Highly customizable and flexible**
  - **Proved scalability up to T1 size**
  - **Java based, then highly portable and platform independent:**
    - We succeeded to use every OS and platform we liked/needed (debian, solaris...)
  - **Integrated solution for both SRM and file-system functionalities**
- How do we use to the storage manager:
  - **Look to the system in order to understand where the bottleneck could be, or where there is room for improvements in order to optimize the configuration to the experiment needs**

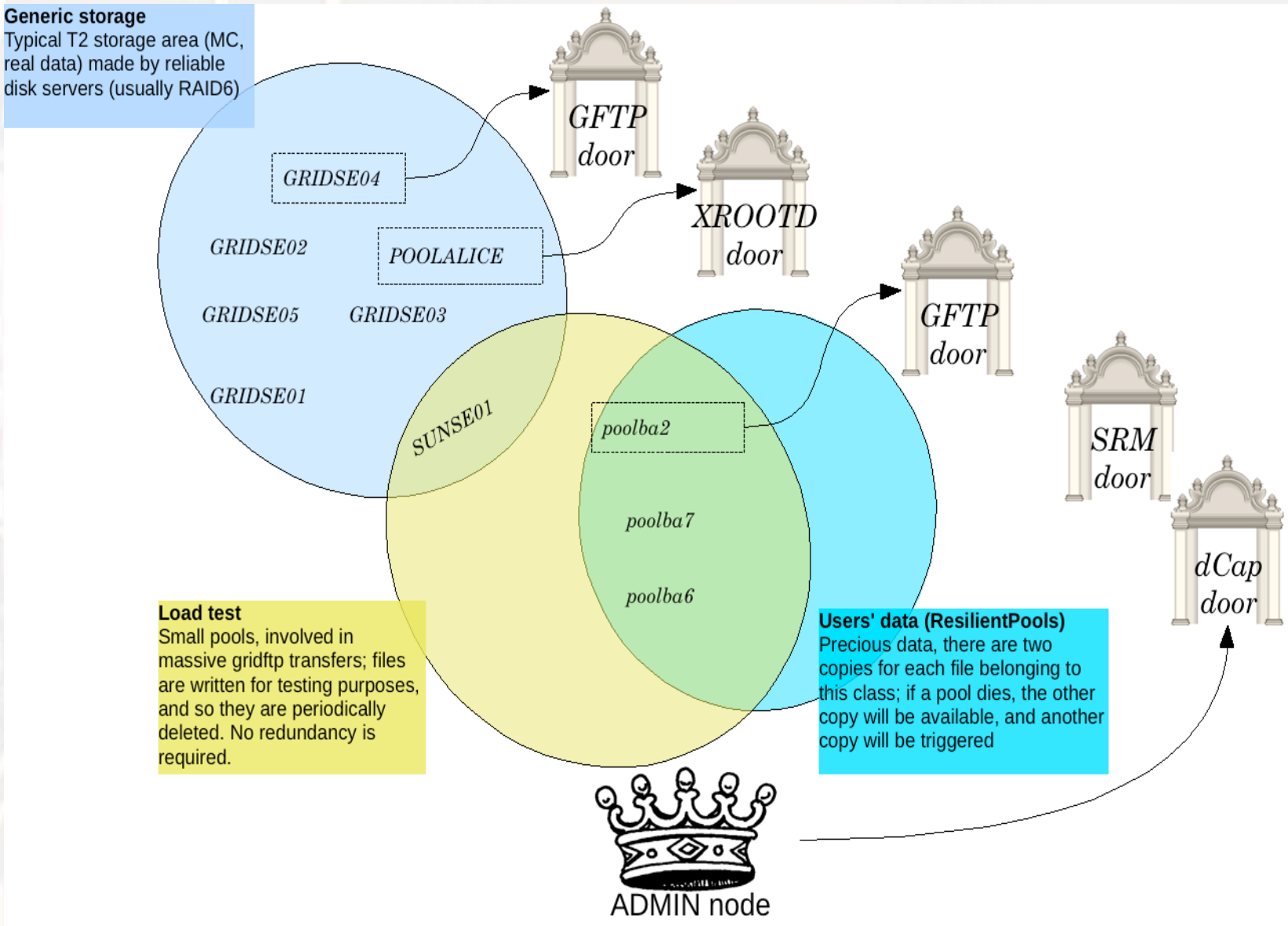
Typical scenario

- More than one VO per site (could co-exists to reduce the man power needed)
- Several and very different storage "boxes"
- Different "kind" of data with different usage pattern and "retention policy"
  - **"short living" and test data**
    - there are constant transfers between CMS sites in order to test the link "healthiness"
    - performance and stability of the rate achieved are key parameters
    - the data can be deleted as soon as they are transferred
  - **"MC produced data"**
    - Data produced locally at the tier2
    - they cost CPU to be reproduced (fairly precious)
  - **Experimental/MC data (replica of T1 data)**
    - Data transferred from the T1 -> can be retransferred easily if the T1 has enough "bandwidth"
  - **User's data**
    - Require a lot of human work to be reproduced (and CPU time) -> Very precious data

Typical scenario



Typical scenario



Configuration “Hacks”

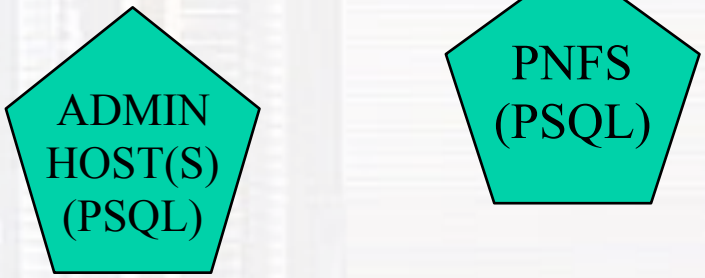
- In a system like dCache, there are several “central processes”
- they can be executed in different machine in order to achieve the needed performances and scalability
- For a Tier2 sized farm a good configuration could be similar to what we report here:

Configuration “Hacks”

- In a system like dCache, there are several “central processes”
- they can be executed in different machine in order to achieve the needed performances and scalability
- For a Tier2 sized farm a good configuration could be similar to what we report here:

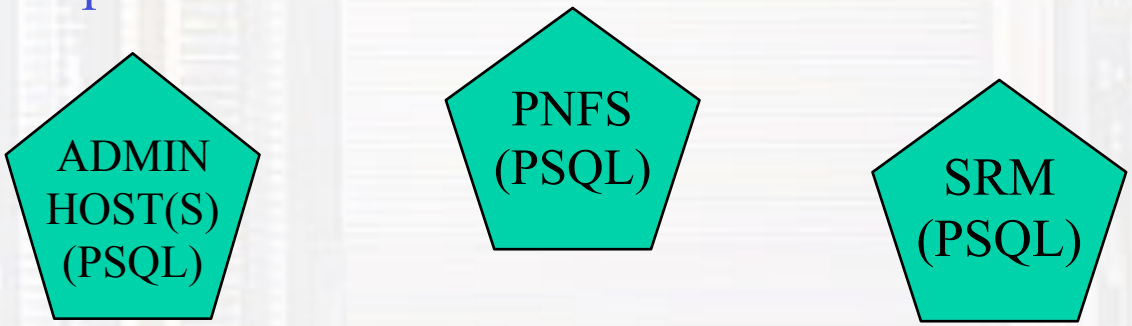
Configuration “Hacks”

- In a system like dCache, there are several “central processes”
- they can be executed in different machine in order to achieve the needed performances and scalability
- For a Tier2 sized farm a good configuration could be similar to what we report here:



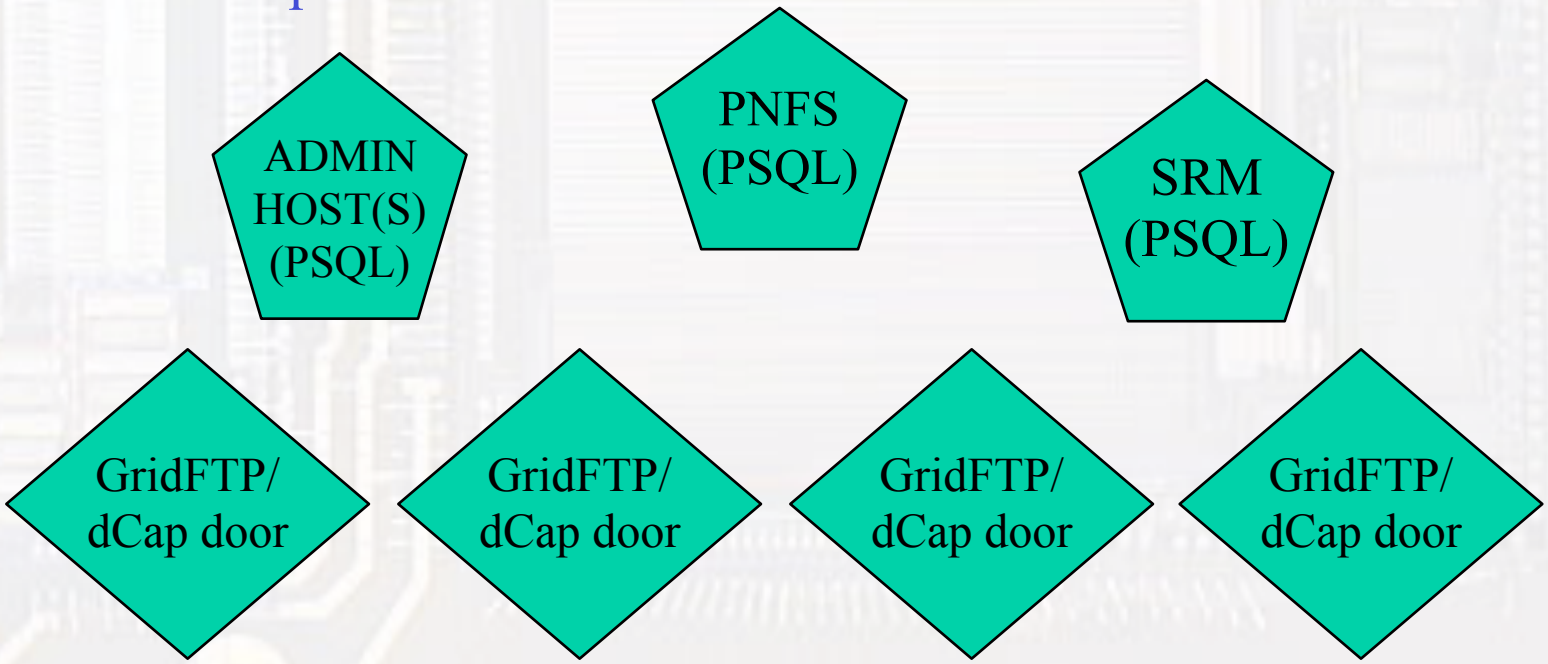
Configuration “Hacks”

- In a system like dCache, there are several “central processes”
- they can be executed in different machine in order to achieve the needed performances and scalability
- For a Tier2 sized farm a good configuration could be similar to what we report here:



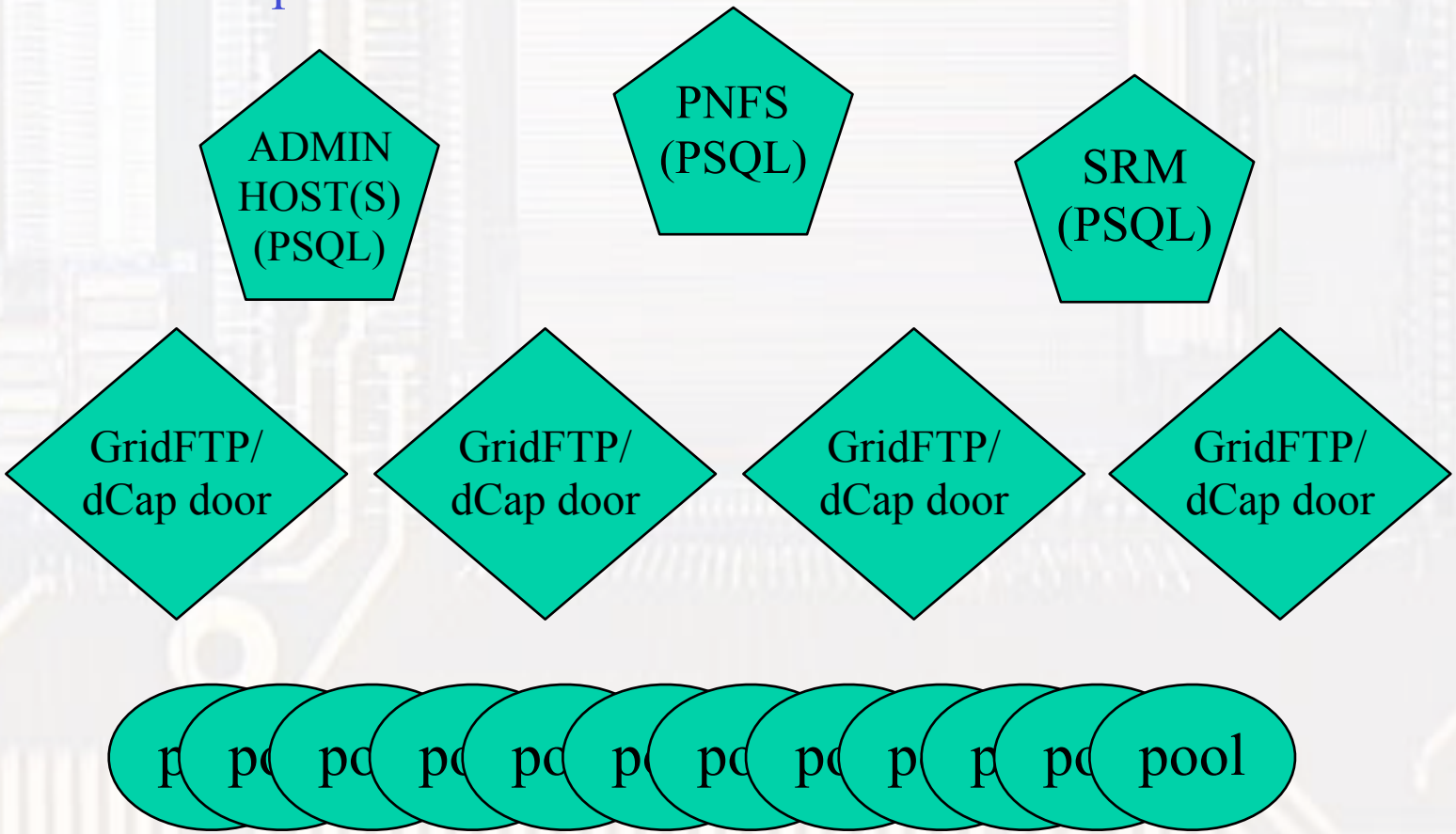
Configuration “Hacks”

- In a system like dCache, there are several “central processes”
- they can be executed in different machine in order to achieve the needed performances and scalability
- For a Tier2 sized farm a good configuration could be similar to what we report here:



Configuration “Hacks”

- In a system like dCache, there are several “central processes”
- they can be executed in different machine in order to achieve the needed performances and scalability
- For a Tier2 sized farm a good configuration could be similar to what we report here:



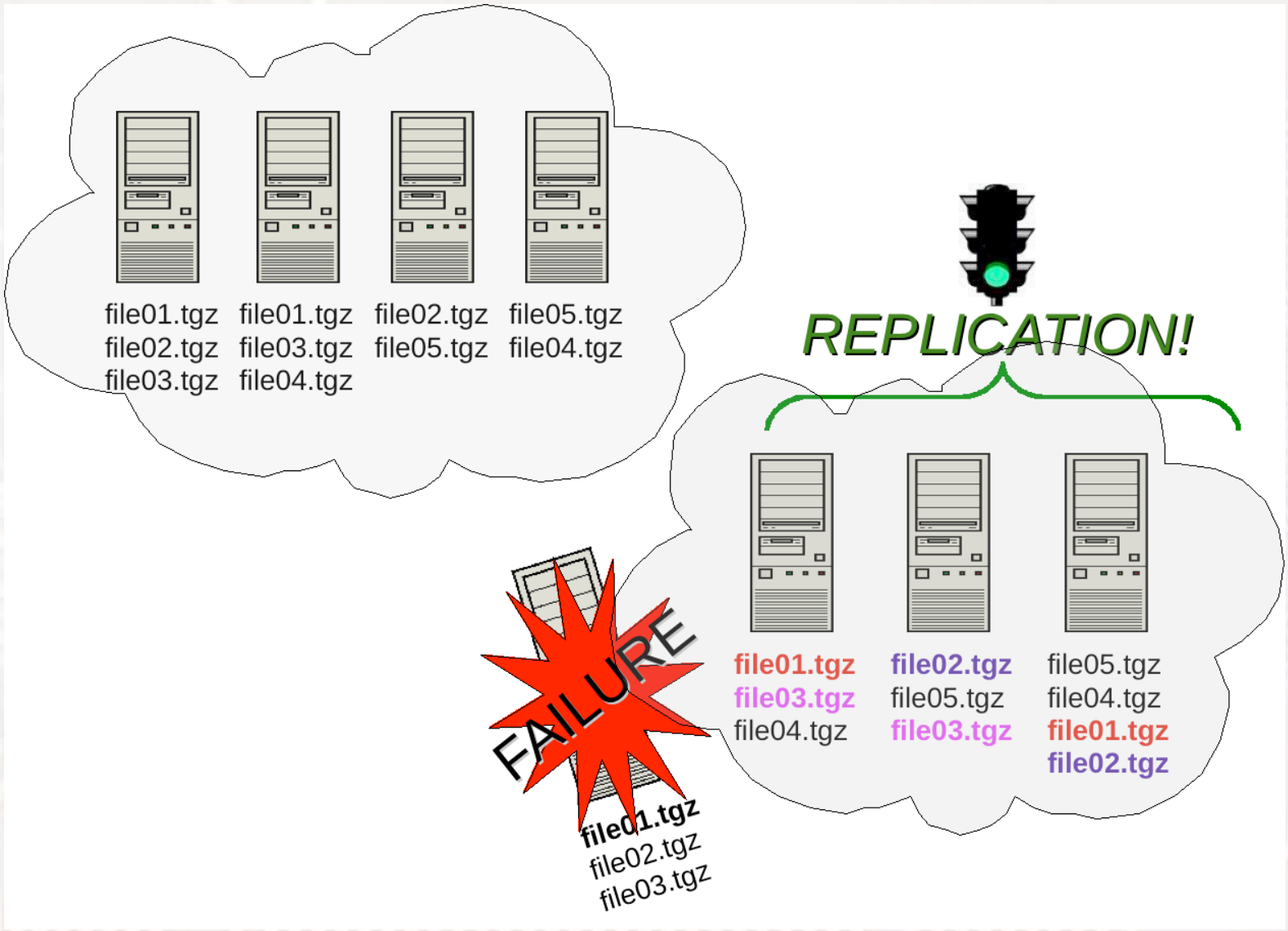
Configuration “Hacks” (2)

- We specialized also the HW configuration of the servers devoted to each “storage area”
  - **Test data:**
    - Based on several “small” servers without raid sub-system:
      - They are fast enough and really cheap
  - **MC/Real Data:**
    - Use large boxes with raid6 (or similar)
      - These machine should be reliable enough (depending on how powerful is the “T1 connection”)
      - The performance and cost are not the only parameters to take into account
  - **User Data:**
    - Raid is not enough here
      - Tried cheap machines without raid solution, but...
      - using software mirror (using dCache ReplicaManager)

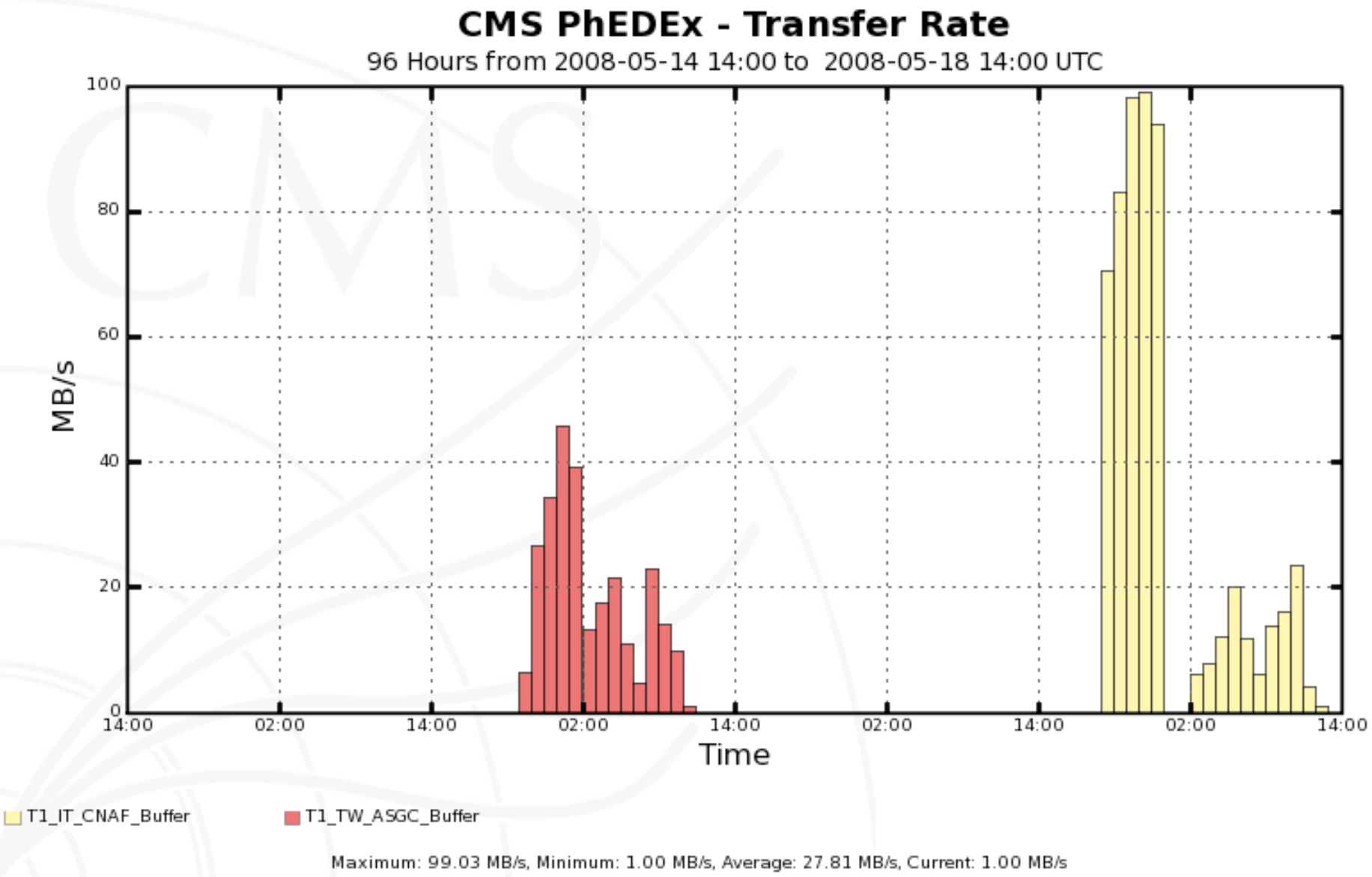
dCache ReplicaManager

- It is based on a process that aims to guarantee the High Availability of files stored in a sub-set of pools (pool groups)
- The basic idea is to enforce the number of "available" copy of the same file
  - **This parameter could be configured by the administrator**
- The process loops at given time interval in order to check for the number of available copy of the files
- The process is driven by information stored into a DB
- It is possible to configure dCache in order to have one or more directory tree associated to one or more pool groups
  - **In this way it is possible to have HA on a given directory tree**
- We "overloaded" this system in order to have more than one Replica process.
  - **This could be useful to guarantee HA of different storage area imposing something like quotas per different use case**

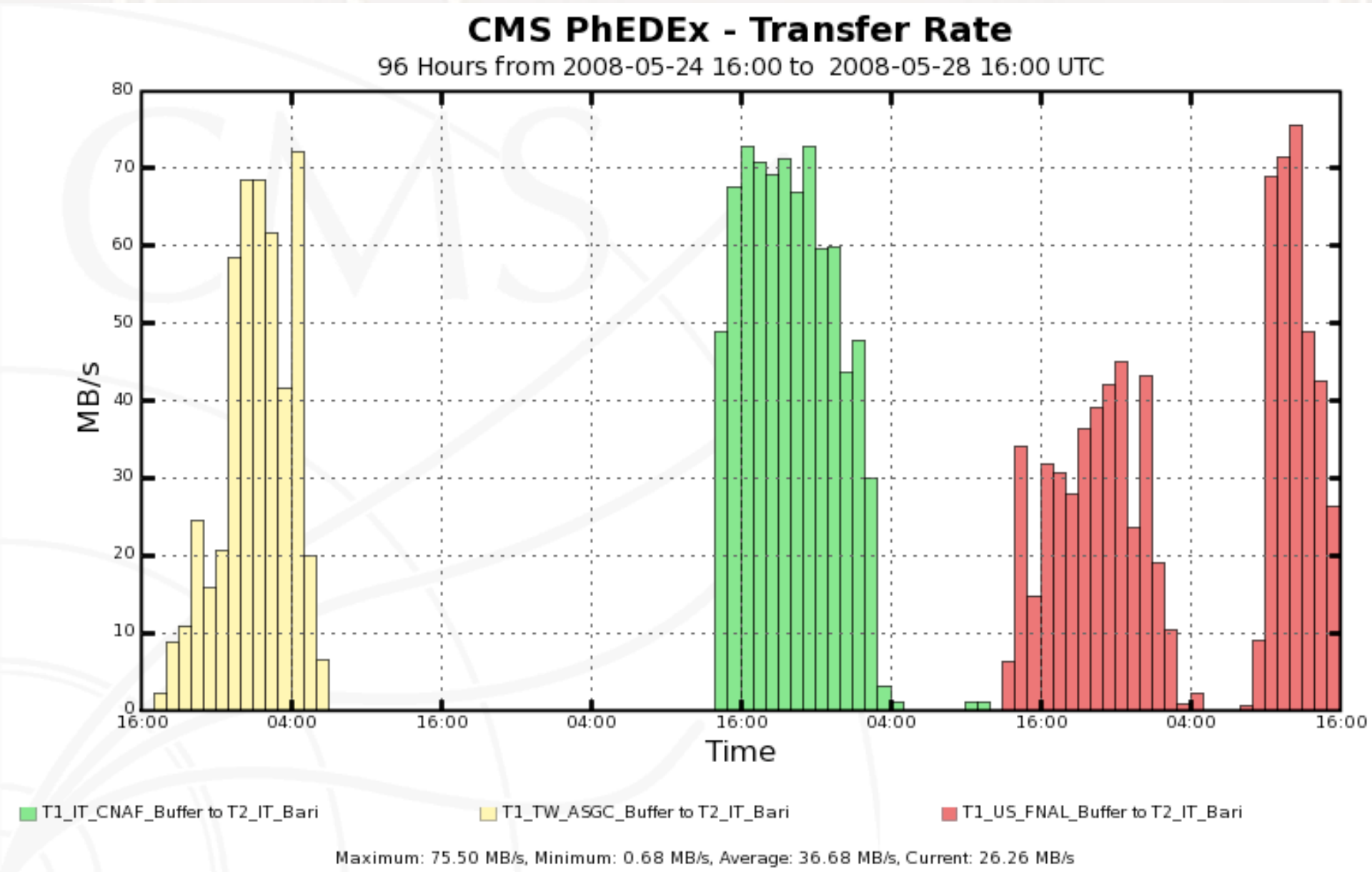
dCache ReplicaManager example



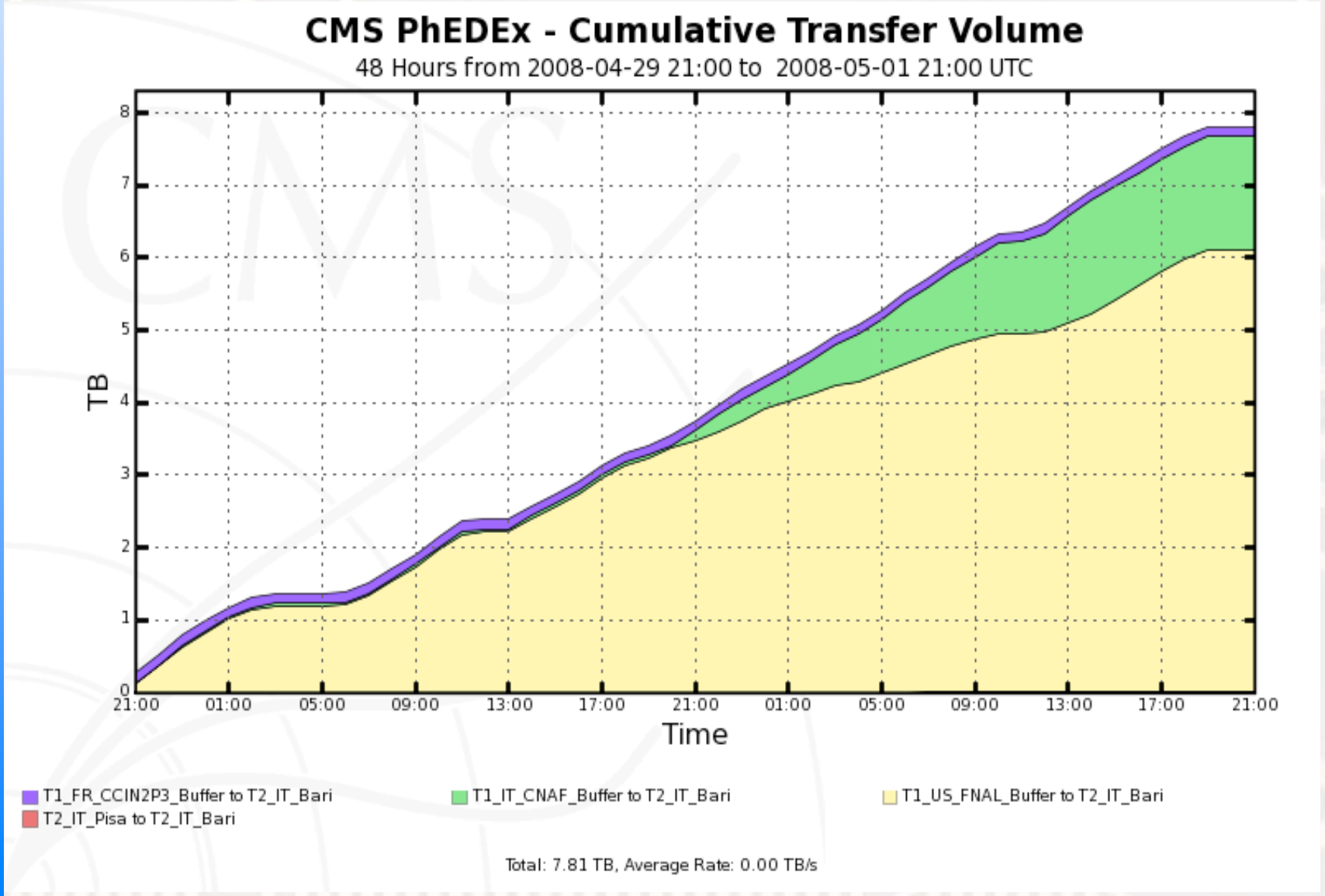
Results achieved



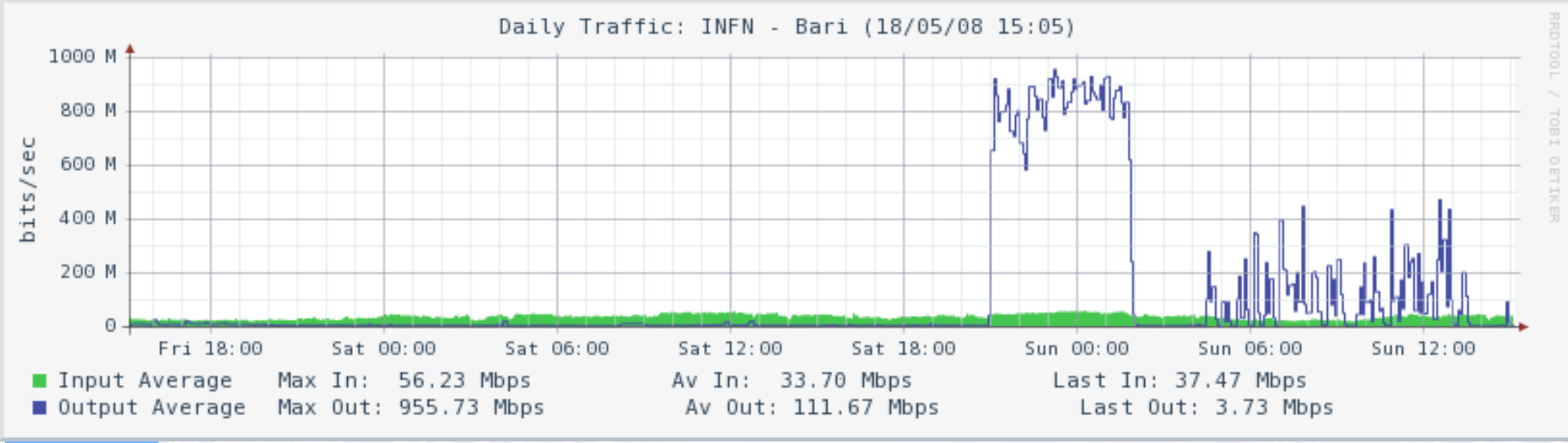
Results achieved



Results achieved



Results achieved



- ✓ Regional and non regional files transfers (WAN)
  - ✓ Already reached the nominal rate
  - ✓ The most of the time we reached a good latency in the transfer time of a "dataset"
- ✓ Several concurrent users reading different datasets
- ✓ We tried up to 90% of job slots full of analysis jobs
- ✓ Concurrency between WAN and analysis access

Conclusions

- ✓ We achieve good performance and stability of the storage system testing it in several challenges
- ✓ Now we are really close to the “nominal rate” of a CMS Tier2
- ✓ dCache proves to have the flexibility needed in order to fulfills requirements at a LCH Tier2
- ✓ We used the same installation for both Alice and CMS tier2
  - ✓ We do not need to spend too much additional effort in order to “serve” more than one VO.
- The site admin needs an effort in learning the system at the beginning

Conclusions

- ✓ Good experience using Thumper and SolarisOS
  - ✓ Fast and quite reliable
    - ✓ We tested also the Custom Care Service:
    - ✓ There was a problem in a driver of the disks controller -> in less than two working days they provided us a patch that solves the problem)
    - ✓ very good job, indeed.

- ➡ Several problems with SLC3/4 OS -> Moved successful to Debian
  - ➡ Good compatibility with hw components
  - ➡ Good compatibility with dCache software
  - ➡ Great reliability
  - ➡ Good security support (Experience with openssl bug -> solved in less than 24 hours)