GPFS at Tier1 and Bari

nuove tecnologie hardware ed esperienza su file system distribuito su WAN.

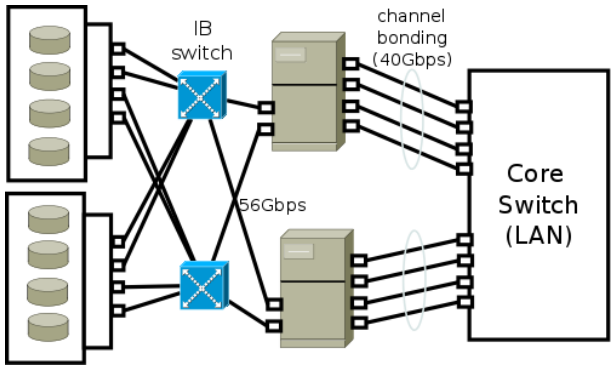
Vladimir Sapunenko (INFN-CNAF),
Giacinto Donvito (INFN-Bari, ReCas),
Alessandro Italiano (INFN-Bari, Recas),
Domenico Diacono (INFN-Bari, ReCas)

Outline

- News from Tier-1
  - New HW - new experience
    - DDP
    - Infiniband
    - 4x10 Gbps ethernet
    - 4Kn HDD
- GPFS at Bari
- How elastic Tier-1 can be
  - Farm extension
  - Remote data access
- GPFS or not GPFS?
  - 4.1 experience

News from Tier-1 storage (CNAF)

- Tender 2015 (10 PB) won by DDN
  - Storage: 2x SFA12K
  - Interconnect:
    - SAN: Mellanox FDR IB (56 Gbps)
    - 16 servers
    - LAN: 4x10 Gbps channel bonding
  - HDD: 8 TB HGST
    - Near line SAS
    - He filled
    - 4Kn sectors
- 10 PB in 3 racks!
- >50 GB/s read or write



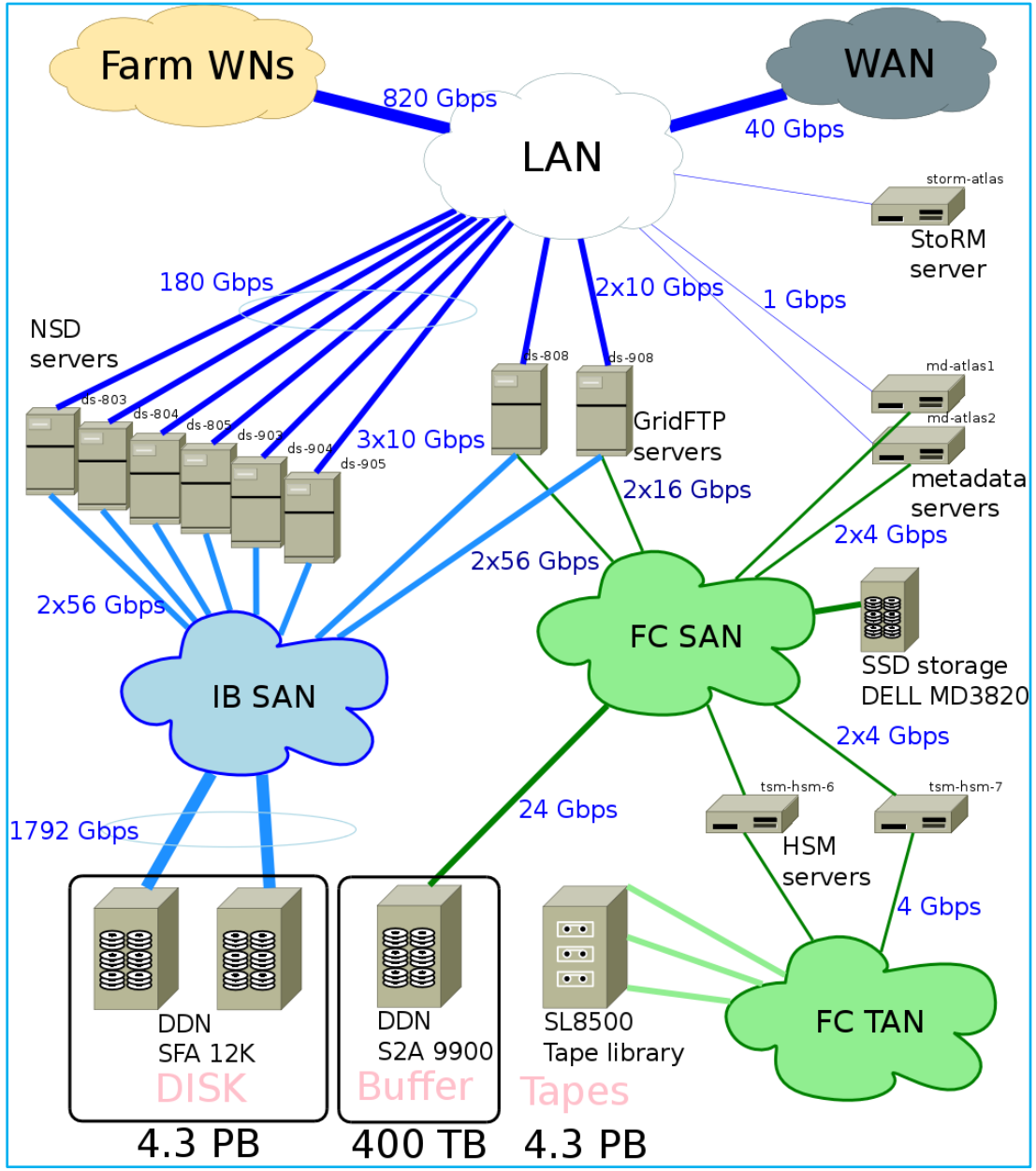
8 TB HGST disks

4Kn HDD

- **Bad news**: GPFS 3.5 (as many other “old” file systems and OS) is not compatible with 4KB sectors; need migrate to 4.1
- **Good (?) news**: In GPFS 4.1.1-3 existing (pre 4.1) file system can be extended with 4Kn disks, but
  - Only as a new Storage Pool
  - The File system will not be 4KB aligned
- **Bottom line**: Need to create new File system To get max performance from 4Kn disks and to migrate ~6 PB of data!

ATLAS example

- 4.3 PB disk space
- 6 NSD servers (3x10 Gbps)
- 2 metadata servers (1Gbps)
- 2 GridFTP (XrootD) (2x10 Gbps)
- 2 HSM servers
- Metadata on SSD (mirrored)
- VM as Storm server
- Throughput required (5 MB/s/TB) = 21.5 GB/s
- Throughput available (6 NSD x 30 Gbps) = 22.5 GB/s



Tier-1 extension

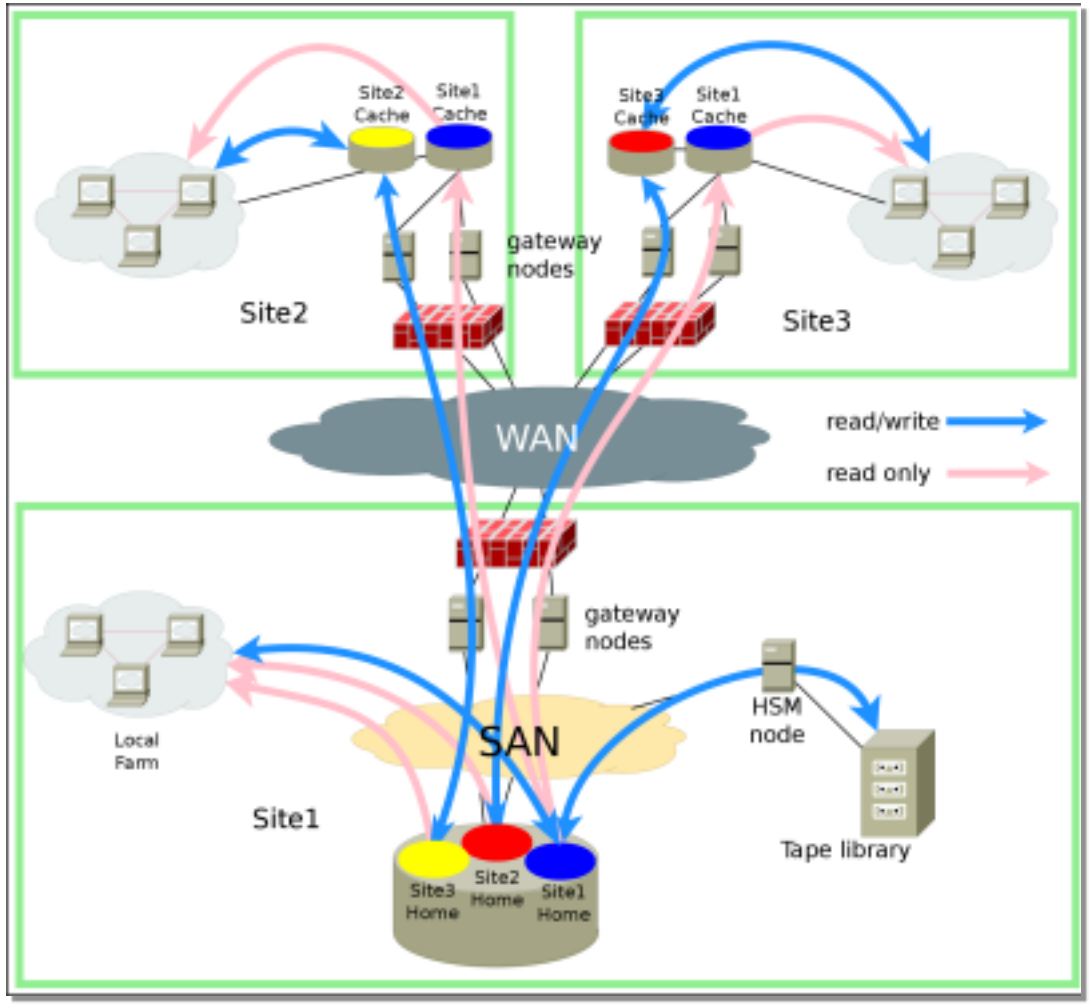
- At Tier-1 we are expecting huge increase in CPU demand in the next years
- To be prepared we are testing possibility to use remote resources to (dynamically) extend Tier-1 farm
- First production use case: part of 2016 pledged resources for WLCG experiments at CNAF are in Bari-ReCaS
  - 40 WNs (~21 kHS06) and ~330 TB of disk allocated to Tier-1 farm for WLCG experiments
  - 64 cores per mb(546 HS06/WN)
  - 1 core/1 slot, 4GB/slot, 8,53 HS06/slot
  - ~10% of CNAF total resources, ~13% of resources pledged to WLCG experiments
- Goal: direct and transparent access from CNAF
- Similar to CERN/Wigner extension

Remote data access

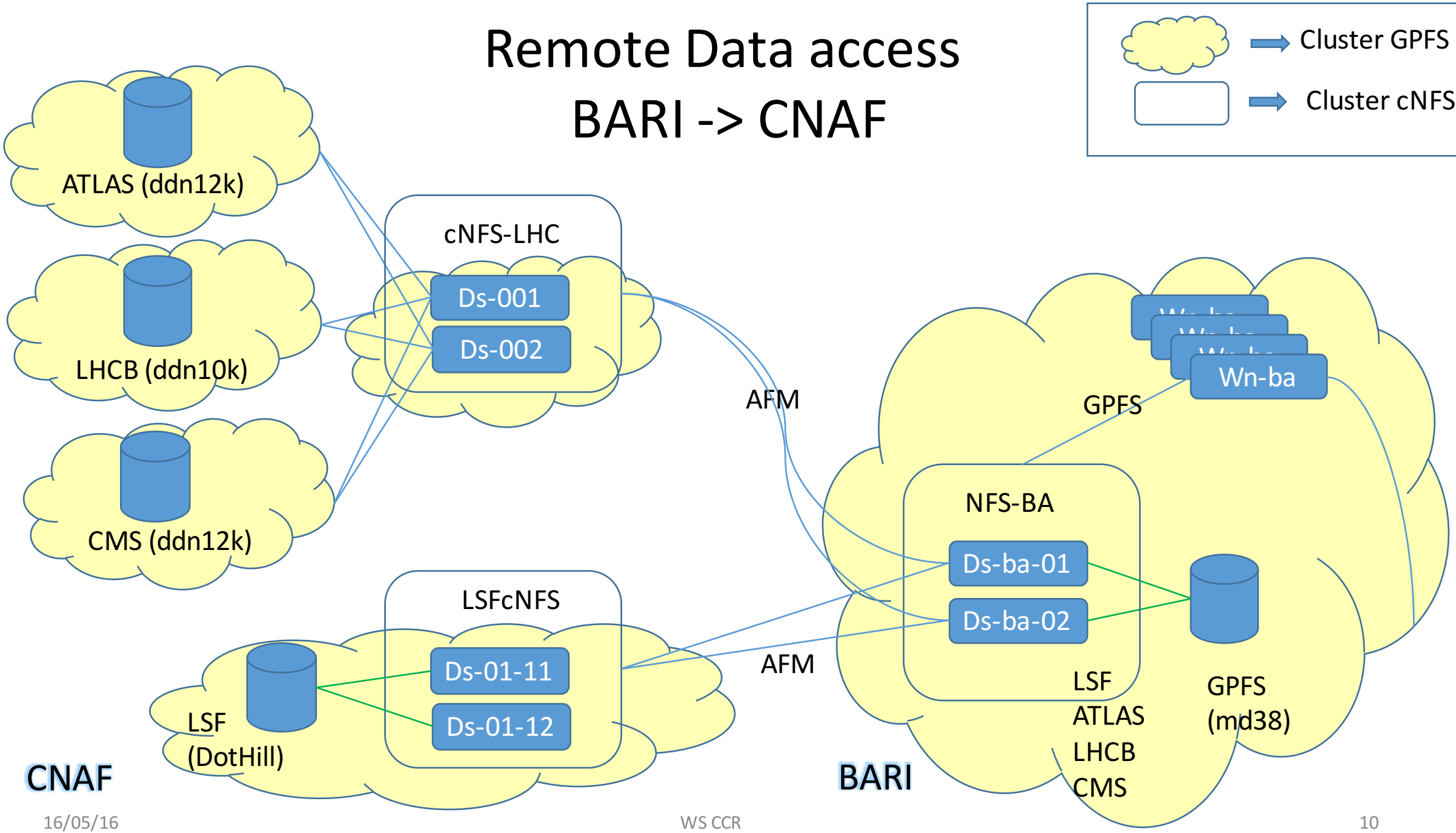
- The goal is to provide the transparent access to T1 data
  - The same name space
  - The same protocols
- Jobs expect to access data the same way as at CNAF
  - Jobs unaware of “Bari connection” ☺
  - Not all experiments able to use a fallback protocol
- Remote mount via GPFS is not considered because of latency
- Local (@Bari) Posix cache for data needed
  - GPFS native feature (AFM)

Remote data access via GPFS AFM

- GPFS AFM
  - A cache providing geographic replica of a file system
  - manages RW access to cache
- Two sides
  - Home - where the information lives
  - Cache
- Data written to the cache is copied back to home as quickly as possible
- Data is copied to the cache when requested
- Configured as read-only for site extension



Remote Data access
BARI -> CNAF



Storage used for Cache @Bari

- Using one standard storage box:
  Dell PowerVault MD3860f
  - Dual controller (active-active)

    Front-end: 4xFC16 ports per controller

    16GB/s total
  - Back-end: 3x6Gbps SAS bus per controller (1 int + 2 ext) - 4.5GB/s total
  - Disks: 120 NLSAS 4TB
  - Base with 60 HDDs
  - One expansion box with 60 HDDs
- Plus 2 servers (10 Gbps)

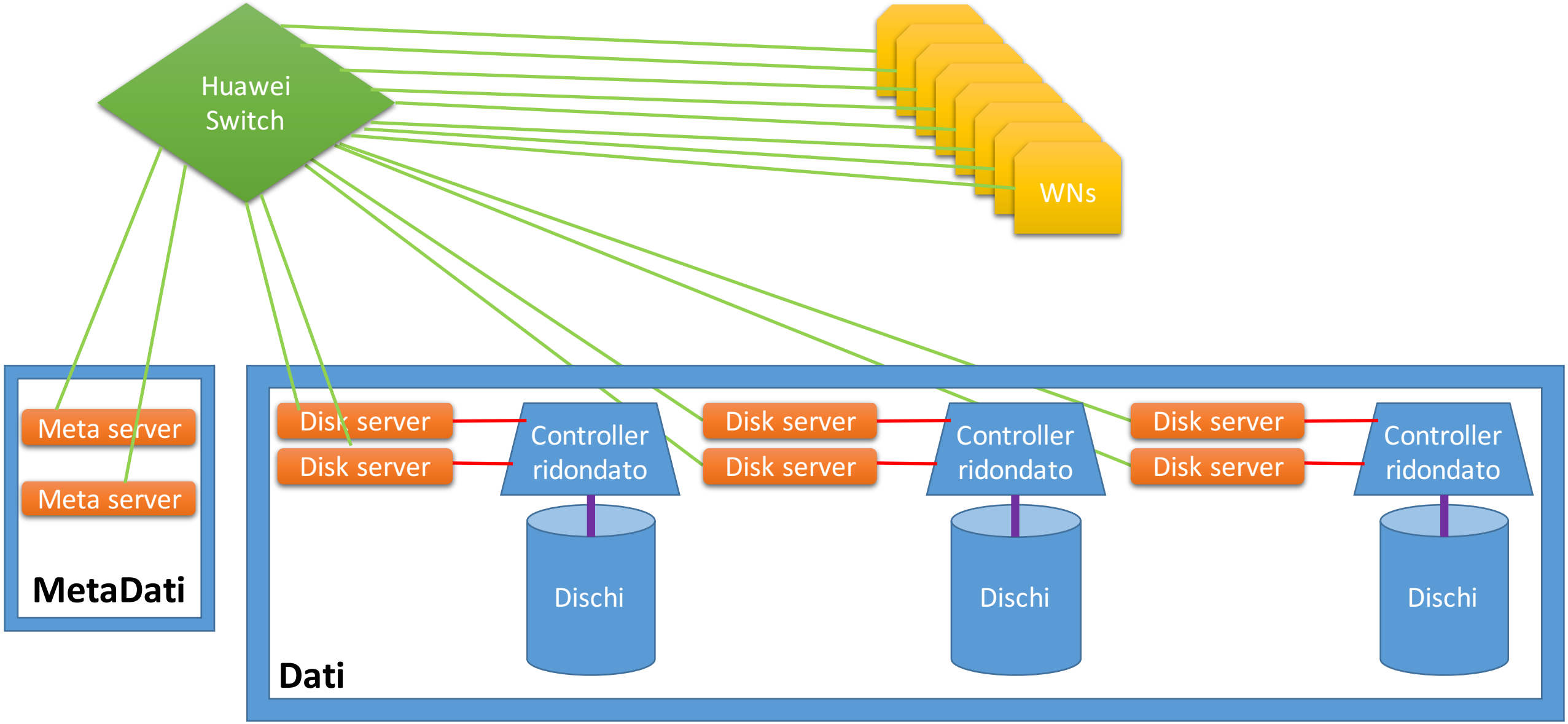
Dell MD3800 and Dynamic Disk Pool

- DDP
  - Best results when all disks in a single pool
- Pro and Contro
  - Significantly reduced Recovery time in case of disk failure
    - Very important when using large and slow disks (> 4 TB)
  - Not good for GPFS
    - GPFS expects to have 1 LU in 1 RAIDset (it's the base of GPFS philosophy)
  - Doing well with a steady unidirectional workload
  - Became a disaster with concurrent read/write access load

GPFS at Bari

- 5 doppi controller **MD3860f** esportati da due **server DELL** ciascuno. Ogni server si collega allo storage tramite due controller FC, e vede ciascun disco tramite due path distinti gestiti da multipath.
- Ogni **server DELL** è collegato allo **Switch Huawei (centrale) con una doppia connessione 10Gbit/s**
- Uno storage **MD3860f** include al suo interno 120 dischi da 4TB, raggruppati in un unico **Distributed Disk Pool**, organizzati in 11 LUN da 30 TB ciascuna
- Ogni LUN viene usata come un singolo NSD per GPFS, ed usa uno dei server come primario, l'altro come secondario. Il carico delle 11 partizioni è diviso tra i due server.
- I dischi sugli **MD3860f** contengono SOLO DATI: I metadati sono ospitati su due server a parte, su dischi SSD

Schema logico



Configurazione dei FileSystem

- Sui cinque MD3860f sono ospitati due filesystem differenti:

1. gpfsHome
    1. I metadati sono in replica 2 su due server, su dischi SSD in RAID1
    2. I dati sono in replica 2
    3. Size Dati: 240TB
        1. A breve verranno aggiunti circa altri 200TB
    4. Questo file system viene usato dagli utenti per la home e quindi contiene dati più critici

2. gpfsData
    1. I metadati sono in replica 2 su due server, su dischi SSD
    2. I dati sono in replica 1
    3. Size Dati: 1.4PB
        1. A breve verrà aggiunto circa 1PB
    4. Requisito essenziale: la perdita di uno storage MD38 non deve rendere inutilizzabile il FS
        1. Ne permanentemente ma neanche temporaneamente

gpfsData Placement policy (1pool=1MD38)

```
RULE 'DATA10' SET POOL 'data1' LIMIT (99) WHERE INTEGER(RAND()*50)<10
RULE 'DATA20' SET POOL 'data2' LIMIT (99) WHERE INTEGER(RAND()*40)<10
RULE 'DATA30' SET POOL 'data3' LIMIT (99) WHERE INTEGER(RAND()*30)<10
RULE 'DATA40' SET POOL 'data4' LIMIT (99) WHERE INTEGER(RAND()*20)<10
RULE 'DATA50' SET POOL 'data5' LIMIT (99)

RULE 'DATA11' SET POOL 'data1' LIMIT (99) WHERE INTEGER(RAND()*40)<10
RULE 'DATA21' SET POOL 'data2' LIMIT (99) WHERE INTEGER(RAND()*30)<10
RULE 'DATA31' SET POOL 'data3' LIMIT (99) WHERE INTEGER(RAND()*20)<10
RULE 'DATA41' SET POOL 'data4' LIMIT (99)

RULE 'DATA12' SET POOL 'data1' LIMIT (99) WHERE INTEGER(RAND()*30)<10
RULE 'DATA22' SET POOL 'data2' LIMIT (99) WHERE INTEGER(RAND()*20)<10
RULE 'DATA32' SET POOL 'data3' LIMIT (99)

RULE 'DATA13' SET POOL 'data1' LIMIT (99) WHERE INTEGER(RAND()*20)<10
RULE 'DATA23' SET POOL 'data2' LIMIT (99)

RULE 'DATA14' SET POOL 'data1' LIMIT (99)
```

Cluster parameters

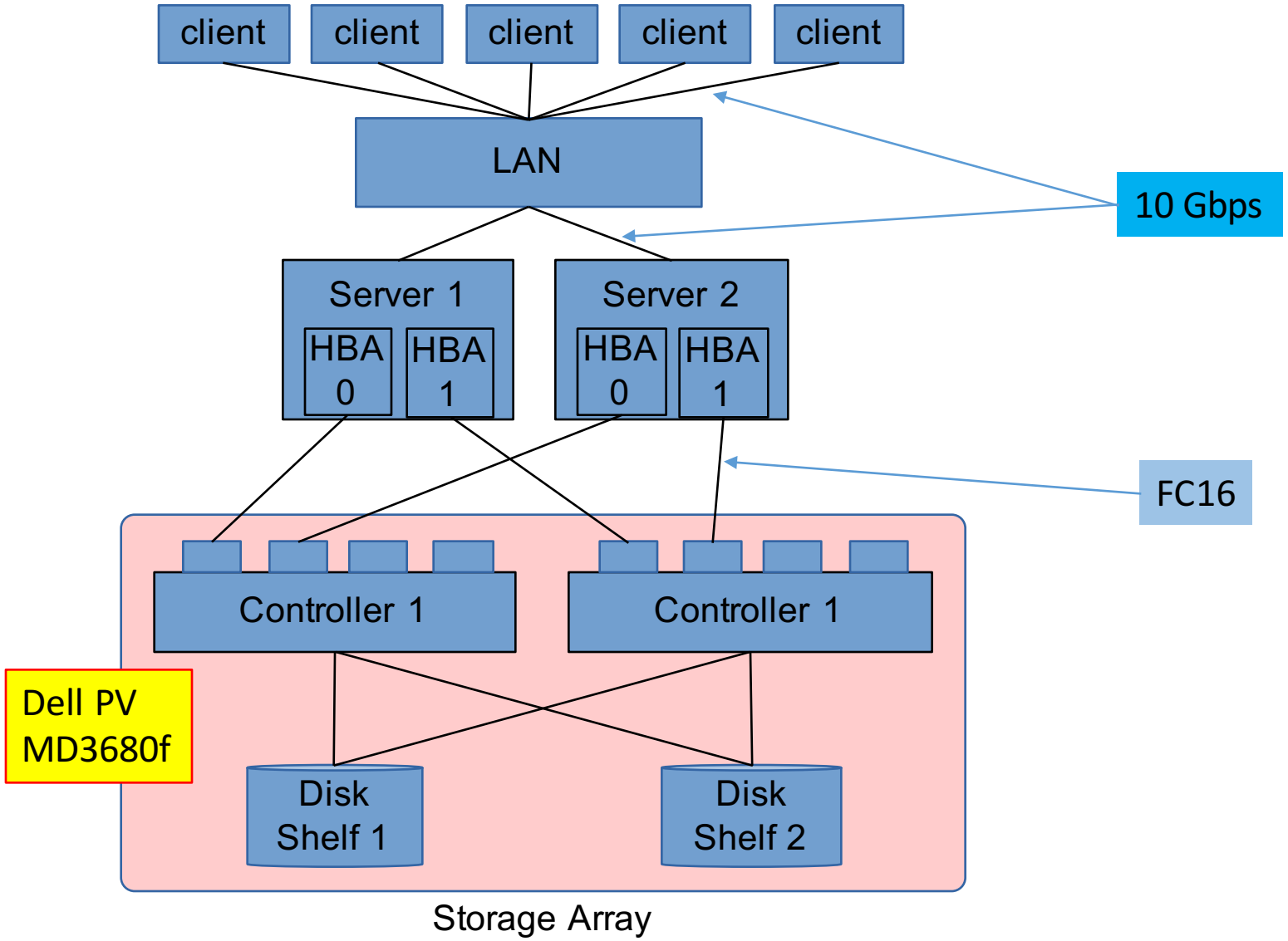
ccrEnabled no
deadlockDetectionThreshold 0
unmountOnDiskFail meta
autoload yes
adminMode central

L'opzione *meta* fa si che la perdita di uno storage renda inaccessibili solo i file di quel pool: il fs non viene smontato e i pool rimanenti sono operativi.

GPFS AFM as a Cache of T1 for Bari

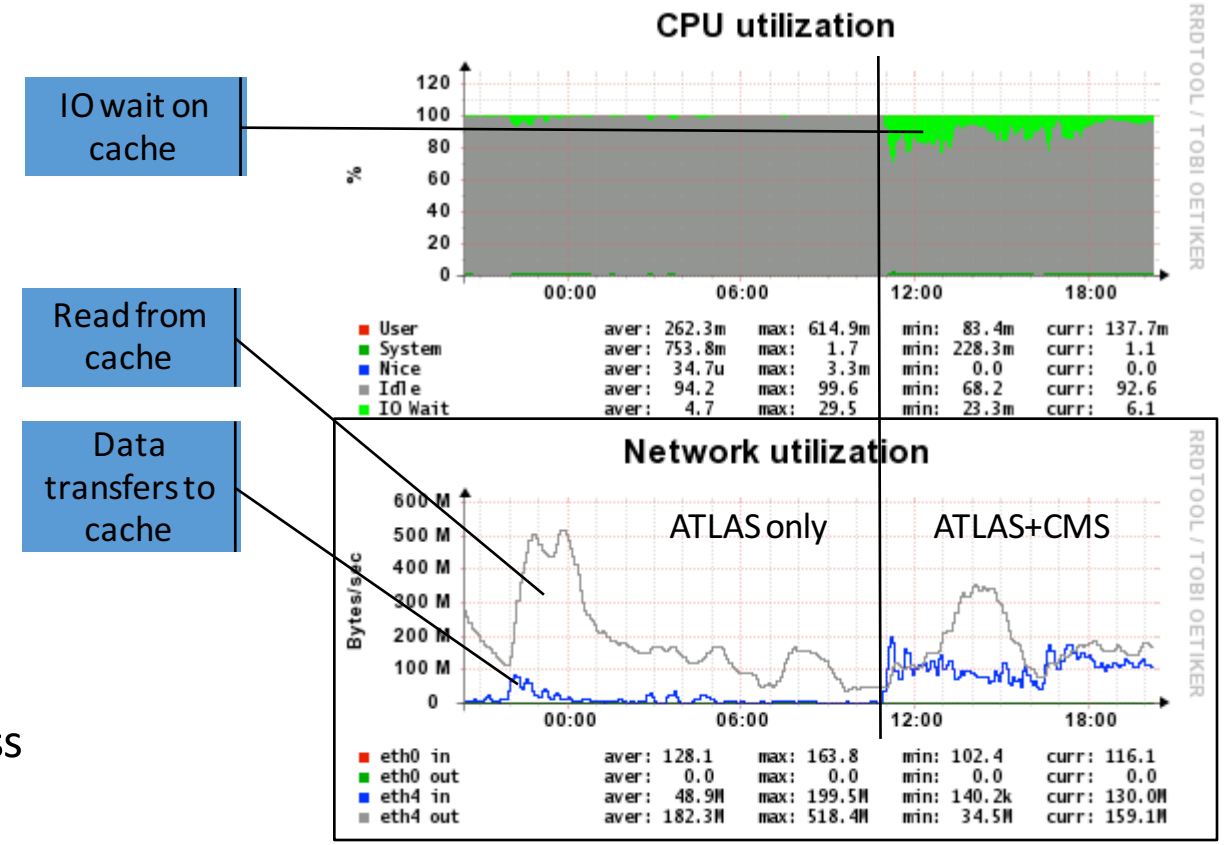
- Cache storage GPFS/AFM
  - 2 server, 10 Gbit
  - 120 TB → 330 TB (Atlas, CMS, LHCb) as cache for data
- Alice experiment does not need cache
  - Remote Xrootd access to data in any case
- CMS able to fallback to Xrootd protocol in case of posix access failure
- (Small) AFM cache also for LSF shared fs
  - On the same storage system
  - Would be better to to place it on a separated system to avoid interferences due to I/O intensive jobs

Cache @Bari


client
client
client
client
client
LAN
10 Gbps
Server 1
HBA 0
HBA 1
Server 2
HBA 0
HBA 1
FC16
Controller 1
Controller 1
Dell PV MD3680f
Disk Shelf 1
Disk Shelf 2
Storage Array

First observations:

- Local cache access critical
  - Potential bottleneck
- First "incarnation" of cache
  - 120 TB of disk space
  - Max 1 GB/s r or w
  - Concurrent r/w degrade performances to 100 B/s
  - 20 TB/experiment
    - CMS fills space in 12h
    - Atlas, LHCb use only 10% of the space
- Very low efficiency for CMS jobs
  - Emergency solution: disable cache access
  - Xrootd fallback

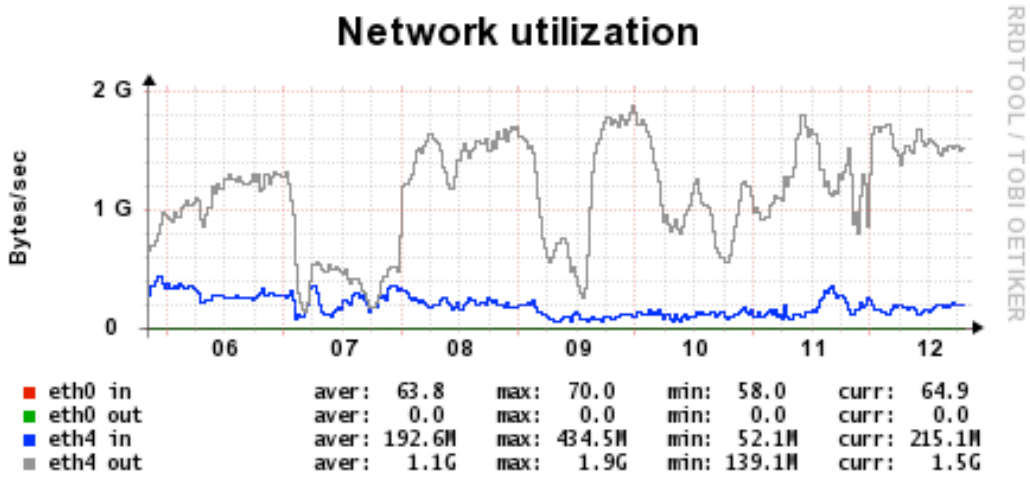


Cache tuning (1)

- Enlargement of data cache (from 120 to 330 TB)
  - ~70 TB per experiment
  - Important for CMS: to accommodate entire datasets for reprocessing
- Increase of file system size does not increase number of disks spindles
  - With a single Dynamic Disk Pool we are using ALL spindles even with a single LU!
- Investigation on GPFS/AFM configuration
  - Several tickets opened to IBM reporting bugs and performance issues

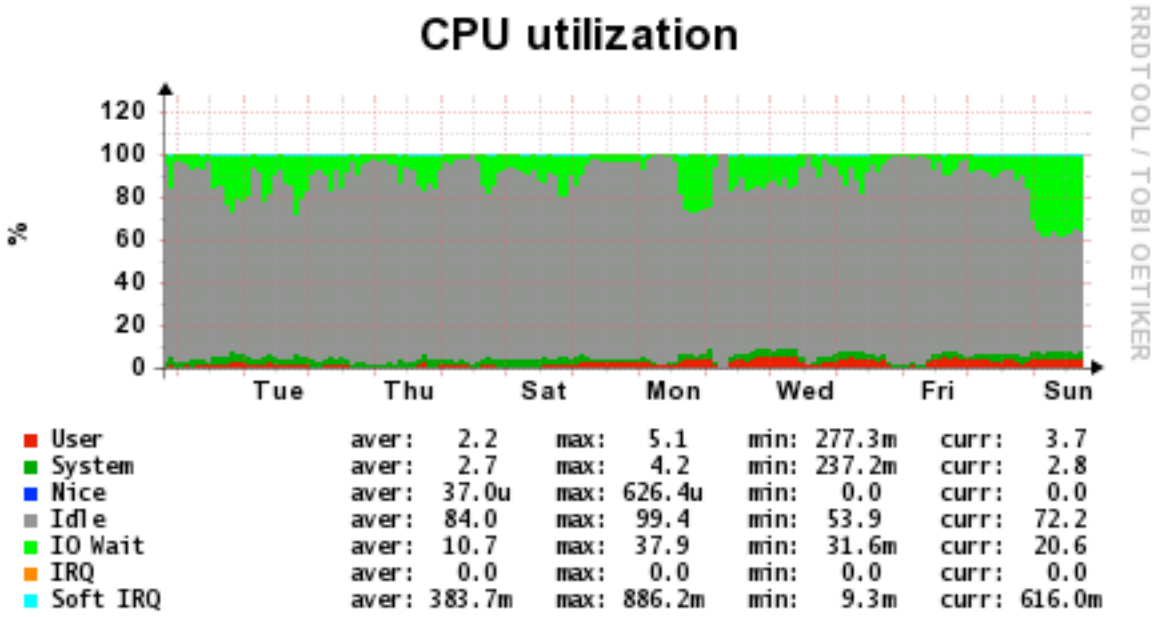
Cache tuning (2)

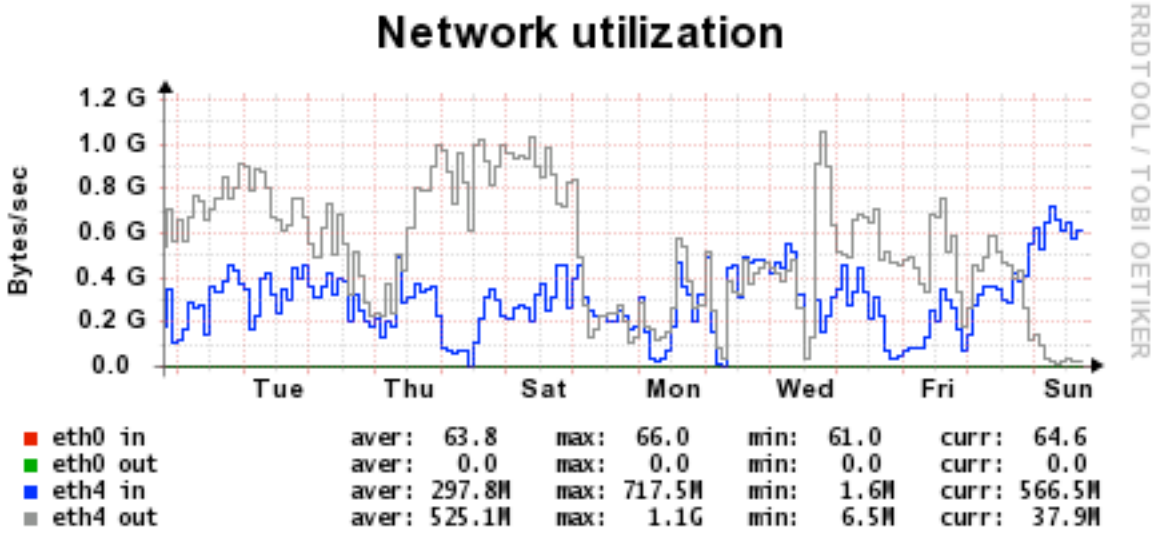
- GPFS optimization normally based on supposition that 1 RAIDset =1 LU and is done on LU level
  - In our case 1 RAIDset contains 12 LU (Virtual disk)
  - we needed to lower number of processes (threads) working with each LU by factor of 10.
- Another way to reduce number of I/O is to Increase of blocksize of the file system
  - 1MB -> 4MB
- Doing well if we have all data already in place



“Cache” is not “Home”

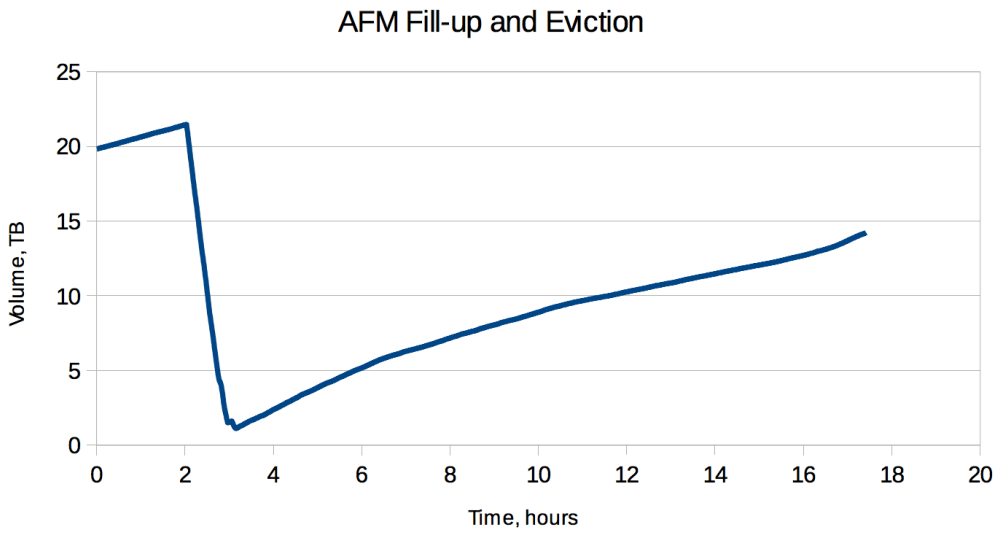
- At “home” the main I/O load comes from WN’s reads
- At “cache”, because of limited space we have reads from WNs AND writes from “home”
- CMS example:
  - Data set size ~40TB
  - Time needed to cache ~11 hours (at 1 GByte/s) if no any other activity present
- Concurrent bi-directional I/O requires more performant disks





Cache eviction

- In AFM cache *fileset* ***AutoEviction*** enabled by default, but will be effective only when
  - *fileset* quota enabled
  - “Soft” and “hard” limits are defined
  - “Soft” limit is less then “Hard”
- *AutoEviction* starts when “soft” limit is reached and (should) bring *fileset* occupation to 80% of “soft” limit.
  - Apparently doesn’t work in this way
- Had to use “manual” mode



“Manual” cache eviction

- Implemented via “callback” on filesetLimitExceeded event
- First disable *AutoEviction*

mmchfileset gpfs CMS -p afmEnableAutoEviction=no

- The callback:

```
mmlscallback
afm_purge
        command       = /var/mmfs/etc/evict.sh
        event         = filesetLimitExceeded
        node          = ds-ba-01.cr.cnaf.infn.it,ds-ba-02.cr.cnaf.infn.it
        parms          = %fsName %filesetName
```

- The script to start eviction:

```
cat /var/mmfs/etc/evict.sh
#!/bin/ksh
/usr/lpp/mmfs/bin/mmafmctl $1 evict -j $2 --safe-limit 18000000000000 --order LRU \
--log-file /storage/log/$2_evict.log
```

Safe-limit is in bytes!

GPFS 4.1 experience

- Good stuff
  - 4Kn disks support
  - Automated deadlock breakup
- Bad stuff
  - Too many bugs in AFM: 15 bug fixes since last release (2 month)
    - ACL – not visible in RO cache (resolved in 4.1.1-6)
    - Gpfs.snap – does “find” on remote filesystem (resolved in 4.1.1-6)
    - Autoeviction – purges 100% of large (>20TB) filesets
    - Metadata Prefetch – mmfs crashes (for filesystem with millions of files)

AFM tuning

- File system
  - Blocksize - defined at the time of FS creation
- Fileset
  - afmPrefetchThreshold (default=0, 100 – no prefetch) – effective immediately
- Cluster config (to be effective requires gpfs restart)
  - nsdThreadsPerQueue 3
  - prefetchThreads 48
  - nsdSmallThreadRatio 4
  - nsdMaxWorkerThreads 64
  - afmMaxWorkerThreads 128

Conclusions

- Limited Caching (with auto-cleaning) is good for limited kind of applications (i.e. NON-data intensive jobs)
- To be efficient the “cache” disk should be more performant (x2) than “home” ones (and so, more expensive!)
- Having 1-to-1 replication of data on remote site is not bad idea at all

Thank you for your attention!