PUGNÈRE Denis

*CNRS / IN2P3 / IPNL*

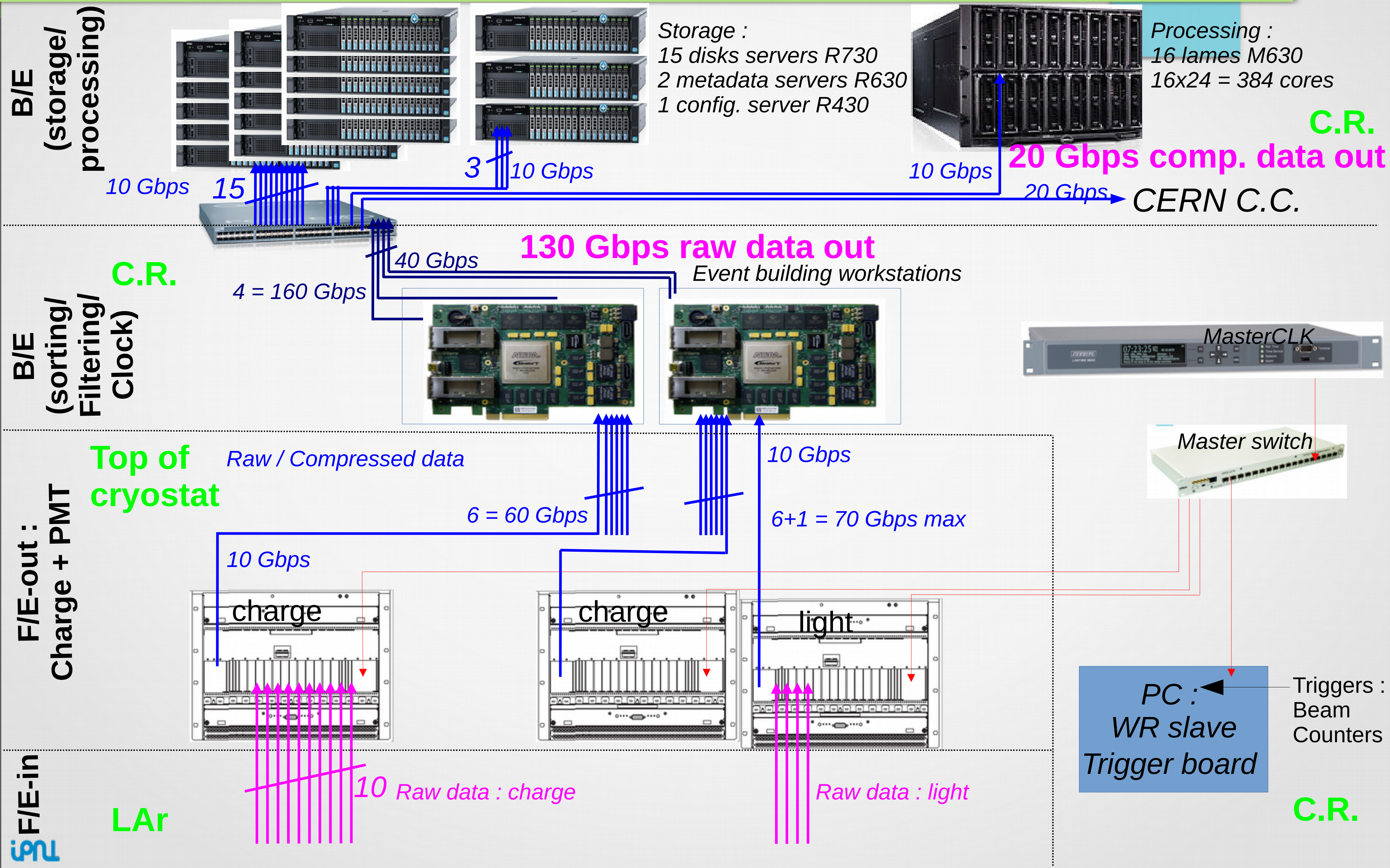
*D.Autiero, D.Caiulo, S.Galymov, J.Marteau, E.Pennacchio*

*E.Bechetoille, B.Carlus, C.Girerd, H.Mathez*

# Comparison between different online storage systems

*WA105 Technical Board Meeting, June 15th, 2016*

# WA105 data network



**B/E (storage/ processing)**

Storage :
15 disks servers R730
2 metadata servers R630
1 config. server R430

Processing :
16 lames M630
16x24 = 384 cores

**C.R.**

**20 Gbps comp. data out**

10 Gbps
15
3
10 Gbps
10 Gbps
20 Gbps
CERN C.C.

40 Gbps

**130 Gbps raw data out**

Event building workstations

**C.R.**

4 = 160 Gbps

**B/E (sorting/ Filtering/ Clock)**

MasterCLK

Master switch

**F/E-out : Charge + PMT**

**Top of cryostat**

Raw / Compressed data

10 Gbps

6 = 60 Gbps

6+1 = 70 Gbps max

10 Gbps

charge

charge

light

PC :
WR slave
Trigger board

Triggers :
Beam
Counters

**F/E-in**

**LAr**

10 Raw data : charge

Raw data : light

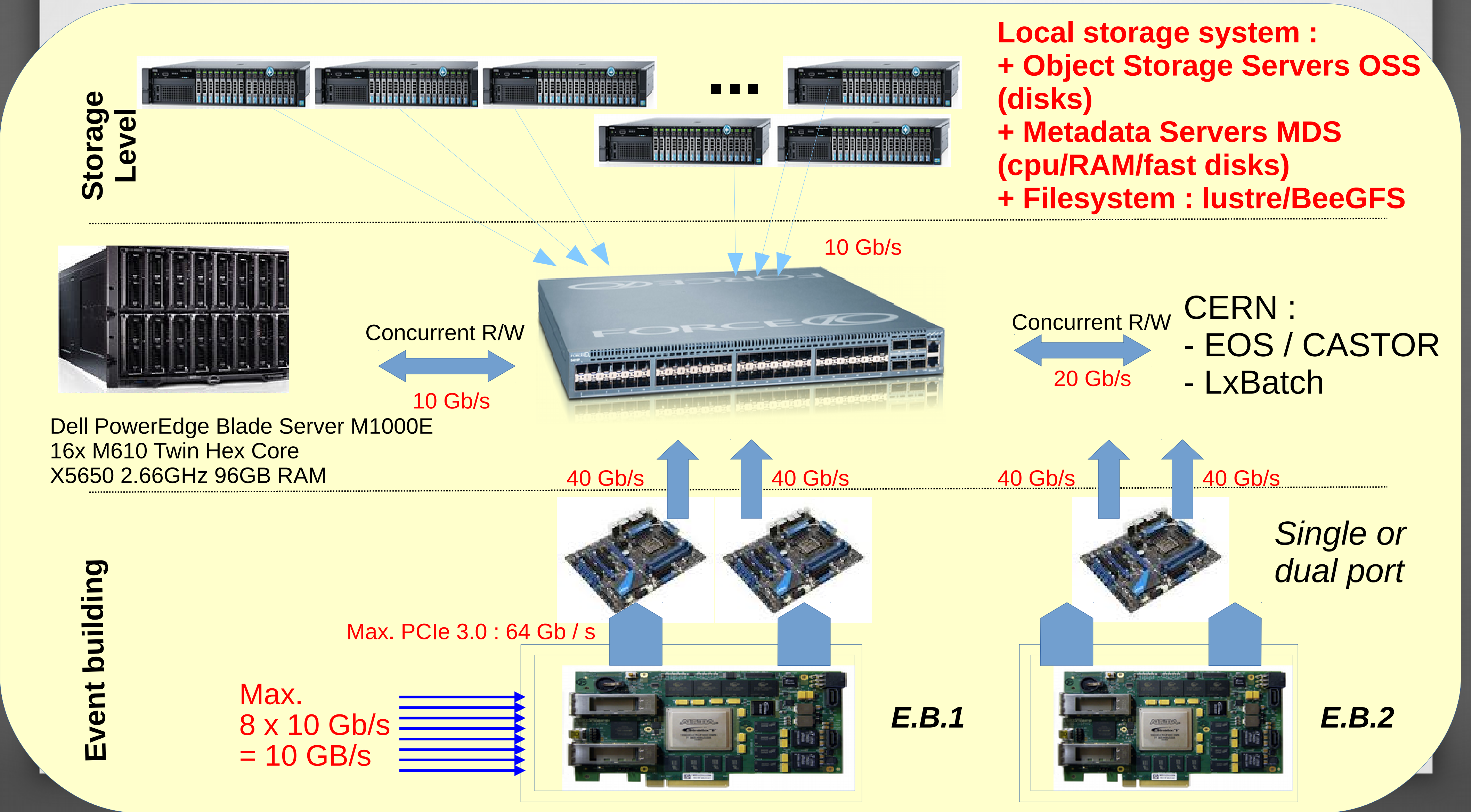
**C.R.**

# Data flow

- AMC charge R/O event size :
  - 12 bits x 64 ch x 10 ksamples ~ 10 Mbits/AMC
- Charge (+ Light) R/O data flow :
  - 100Hz events = 10 x 100 Mbits/s/AMC = 1 Gbps/AMC
  - **10 Gbps/crate**
  - 130 Gbps (16 GB/s) total
- Requirements for online storage :
  - 16 GB/s x 1 day x 50 % duty cycle = 700 TB per day !
  - No compression / reduction factor included : raw data
  - Assumption : factor 10 reduction with Huffman

# Distributed storage solution

CERN requirements : ~3 days autonomous data storage for each experiment : ~1PB
WA105 ~ LHC-experiment requirements



**Storage Level**

**Local storage system :**
**+ Object Storage Servers OSS (disks)**
**+ Metadata Servers MDS (cpu/RAM/fast disks)**
**+ Filesystem : lustre/BeeGFS**

Dell PowerEdge Blade Server M1000E
16x M610 Twin Hex Core
X5650 2.66GHz 96GB RAM

CERN :
- EOS / CASTOR
- LxBatch

*Single or dual port*

**Event building**

Max.
8 x 10 Gb/s
= 10 GB/s

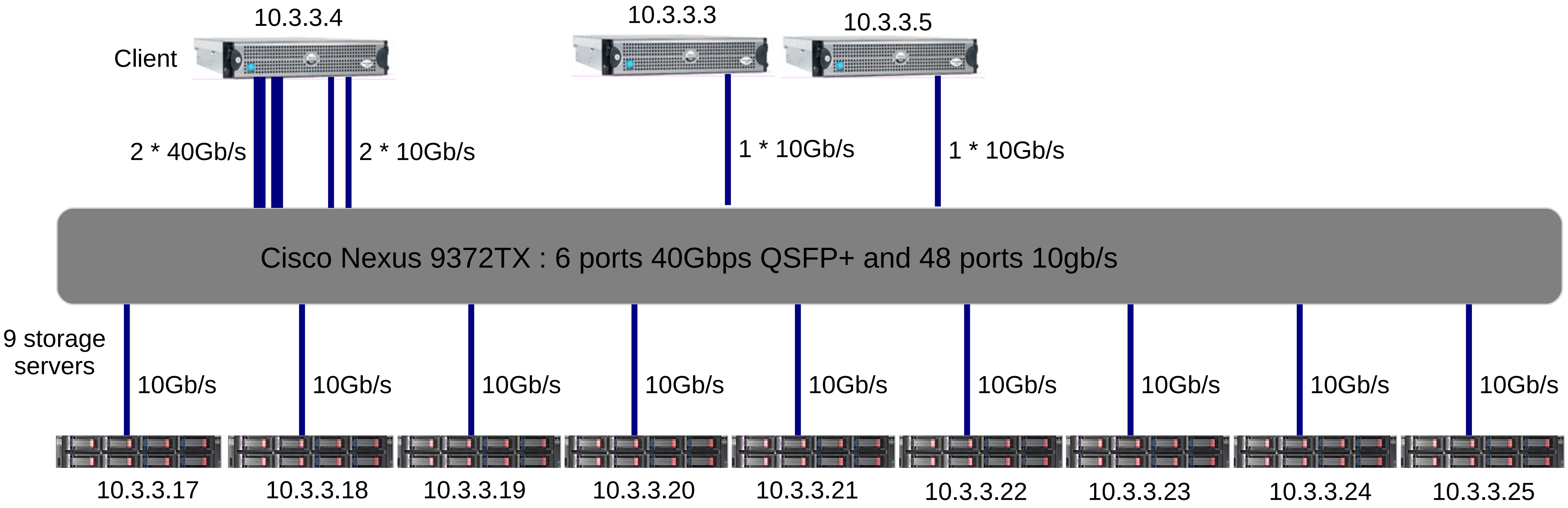
***E.B.1***

***E.B.2***

# Tests benchmarks

**Client** : Dell R630
- 1 CPU E5-2637 @ 3.5Ghz (4c, 8c HT),
- 32Go RAM 2133 Mhz DDR4
- 2 * Mellanox CX313A 40gb/s
- 2 * 10Gb/s (X540-AT2)
- CentOS 7.0

**MDS / Managment** : 2 * Dell R630
- 1 CPU E5-2637 @ 3.5Ghz (4c, 8c HT),
- 32Go RAM 2133 Mhz DDR4
- 2 * 10Gb/s (X540-AT2)
- Scientific Linux 6.5 et Centos 7.0



10.3.3.4

Client

10.3.3.3

10.3.3.5

2 * 40Gb/s

2 * 10Gb/s

1 * 10Gb/s

1 * 10Gb/s

Cisco Nexus 9372TX : 6 ports 40Gbps QSFP+ and 48 ports 10gb/s

9 storage servers

10Gb/s

10Gb/s

10Gb/s

10Gb/s

10Gb/s

10Gb/s

10Gb/s

10Gb/s

10Gb/s

10.3.3.17

10.3.3.18

10.3.3.19

10.3.3.20

10.3.3.21

10.3.3.22

10.3.3.23

10.3.3.24

10.3.3.25

**9 Storage Servers** : (9 * Dell R510 : bought Q4 2010)
- 2 * CPU E5620  @ 2.40GHz  (4c, 8c HT), 16Go RAM
- 1 carte PERC H700 (512MB) : 1 Raid 6 12HDD 2TB (10D+2P) = 20TB
- 1 Ethernet intel 10Gb/s (X520/X540)
- Scientific Linux 6.5

# Storage systems tested

Given the data flow constraints, research for storage systems candidates :

- Which can fully exploit hardware capacity
- Which are very CPU efficient on the client

=> Tests objectives : Characterization of the acquisition system and the storage system on the writing performance criteria

| | Lustre | BeeGFS | GlusterFS | GPFS | MooseFS | XtreemFS | XRootD | EOS |
|---|---|---|---|---|---|---|---|---|
| Versions | v2.7.0-3 | v2015.03.r10 | 3.7.8-4 | v4.2.0-1 | 2.0.88-1 | 1.5.1 | 4.3.0-1 | Citrine 4.0.12 |
| POSIX | Yes | Yes | Yes | Yes | Yes | Yes | via FUSE | via FUSE |
| Open Source | Yes | Client=Yes,<br>Serveur=EULA | Yes | No | Yes | Yes | Yes | Yes |
| Need for MetaData Server ? | Yes | Metadata + Manager | No | No | Metadata + Manager | | Yes | Yes |
| Support RDMA / Infiniband | Yes | Yes | Yes | Yes | No | No | No | No |
| Striping | Yes | Yes | Yes | Yes | No | Yes | No | No |
| Failover | M + D (1) | DR (1) | M + D (1) | M + D (1) | M + DR (1) | M + DR (1) | No | M + D (1) |
| Quota | Yes | Yes | Yes | Yes | Yes | No | No | Yes |
| Snapshots | No | No | Yes | Yes | Yes | Yes | No | No |
| Integrated tool to move data over data servers ? | Yes | Yes | Yes | Yes | No | Yes | No | Yes |

(1) : M=Metadata, D=Data, M+D=Metadata+Data, DR=Data Replication

# Storage systems tested

- Notes on the storage systems choices :
  - All are in the class « software defined storage »
  - Files systems :
    - GPFS, Lustre and BeeGFS are well known on the HPC (High Performance Computing) world : they are parallel file systems which perform well when there are many workers and many data servers
    - I wanted also to test GlusterFS, MooseFS, XtreemFS to see they caracteristics
  - Storage systems :
    - XrootD is a very popular protocol for data transfers in High Energy Physics, integrating seamlessly with ROOT, the main physics data format
    - EOS : large disk storage system (135PB @CERN), multi-protocol access (http(s), webdav, xrootd…)
  - All these systems has they strengths and weaknesses, not all discussed here

**Attention : I’ve tuned only some parameters of these storage systems, but not all, so they are not optimal.**
**Not all technical details are shown in this slideshow, contact me if you need them**

# Tests strategy

1 : Network-alone tests

+

2 : Client tests

+

3 : Storage tests

+

4 : Complete chain tests

**Protocol tests including :**

- TCP / UDP protocols (tools used : iperf, nuttcp...)
- Network interface saturation : congestion control algorithms cubic, reno, bic, htcp...
- UDP : % packets loss
- TCP : retransmissions
- Packets drops
- Rates in writting

**What type of flux may be generated by the client :**

Initial tests => optimizations => characterization

- Optimizations :
  - Network Bonding : LACP (IEEE 802.3ad), balance-alb, balance-tlb
  - Network buffers optimization : modif /etc/sysctl.conf
  - Jumbo frames (MTU 9216)
  - CPU load : IRQ sharing over all cores
    - chkconfig irqbalance off ; service irqbalance stop
    - Mellanox : set_irq_affinity.sh p2p1

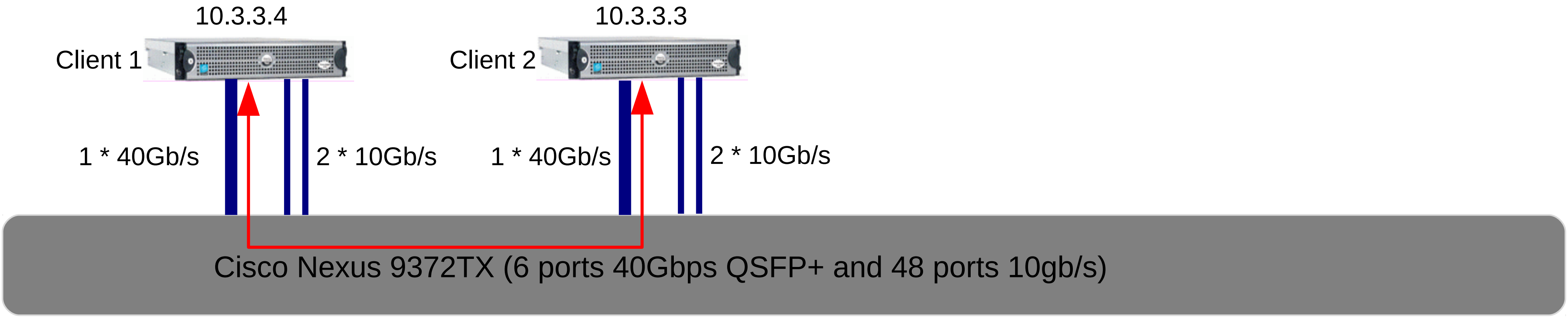
**Individual tests of the storage elements :**

- benchmark of the local filesystem (tools used : Iozone, fio, dd)

**Tests of the complete chain :**

- On the client
  - Storage : Iozone, fio, dd, xrdcp
  - Network/ System : dstat
- On the storage elements : dstat

# 1-a. Network tests between 2 clients



10.3.3.4

Client 1

1 * 40Gb/s

2 * 10Gb/s

10.3.3.3

Client 2

1 * 40Gb/s

2 * 10Gb/s

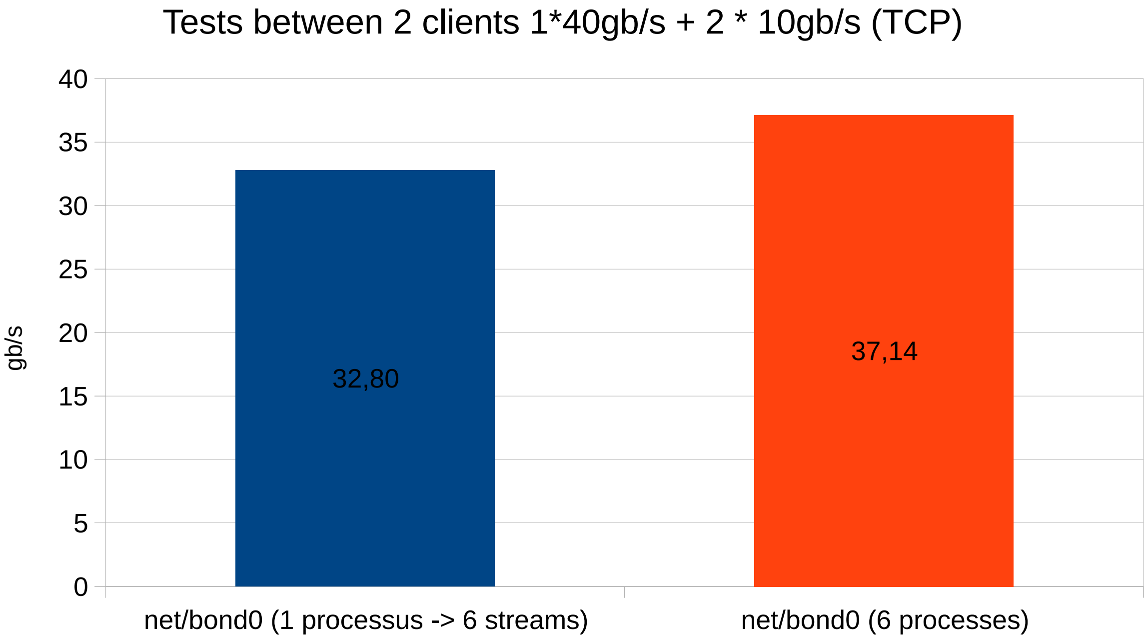
Cisco Nexus 9372TX (6 ports 40Gbps QSFP+ and 48 ports 10gb/s)

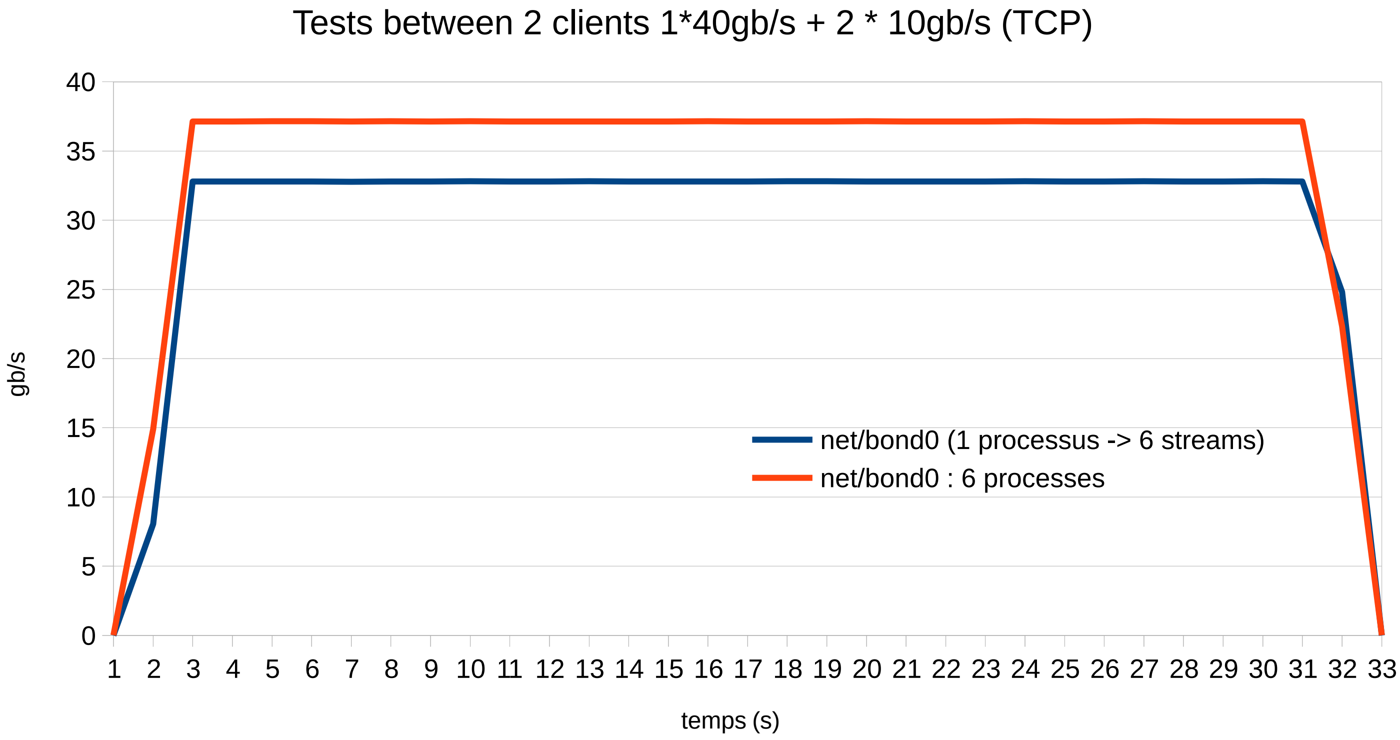
**How behave the flows between 2 clients with each 1 * 40gb/s + 2 * 10gb/s cards ?**

- Tests of many network configuration parameters :
  - net.ipv4.tcp_congestion_control = cubic, MTU=9216, irq affinity on all CPU cores, tuning mellanox,
  - Bonding, tests of several algorithms : mode=balance-tlb, balance-alb (xmit_hash_policy=layer2 ou layer3+4), but not LACP (IEEE 802.3ad)
- Network bandwidth tests between **only 2 « clients » with 1*40Gb/s + 2*10Gb/s each** (nuttcp)
  - **Only 1 flow** between 10.3.3.3 et 10.3.3.4 : TCP = **34746 Mb/s (0 TCP retrans)**, UDP = 38561Mb/s (2.74 % UDP packet losses)
  - **1 processus, 6 flows** between 10.3.3.3 et 10.3.3.4 : TCP = **35058 Mb/s (0 TCP retrans)**
  - **6 processus (1 flow / processus)** between 10.3.3.3 et 10.3.3.4 : TCP = **39532 Mb/s (0 TCP retrans)**

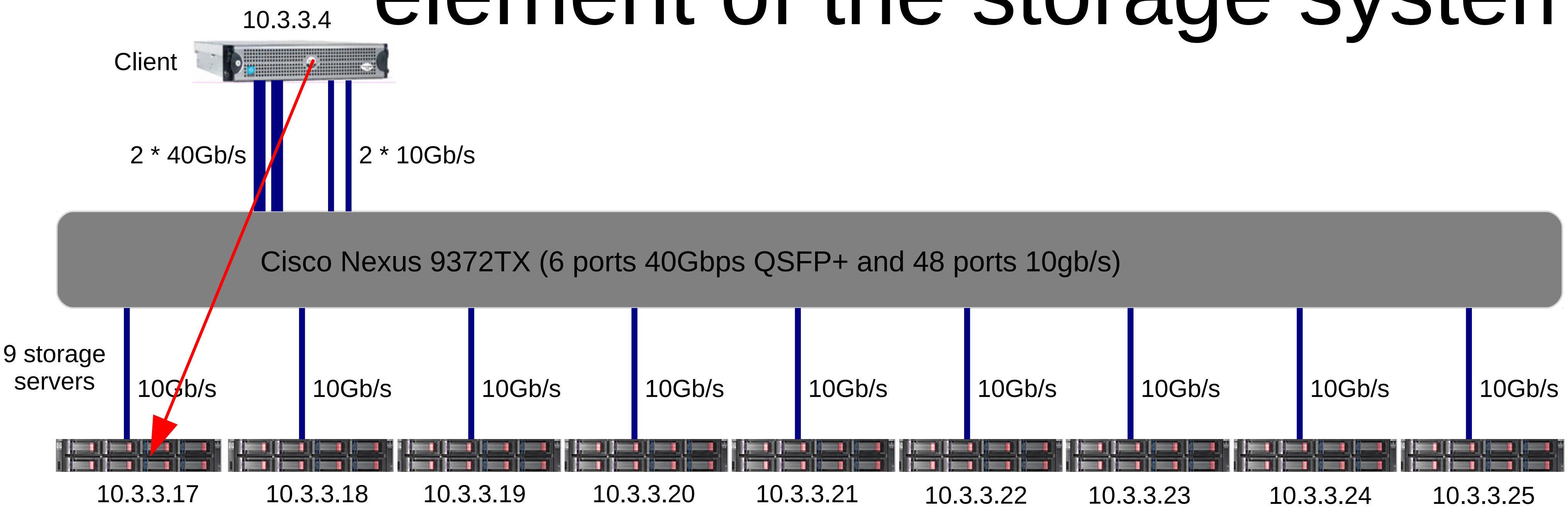
# Comparison 1 vs 6 processes :

- Bandwidth comparison beween :
  - 1 process which generate 6 streams
  - 6 process, 1 stream / process
- 30 secondes test
- Near saturation of the 40Gb/s card
- the flow doesn't pass thru the 2*10Gb/s cards (all bonding algorithms tested)
- +12.7 % when the flows are generated by 6 independent process





# 1-b. Network tests to individual element of the storage system



10.3.3.4

Client

2 * 40Gb/s

2 * 10Gb/s

Cisco Nexus 9372TX (6 ports 40Gbps QSFP+ and 48 ports 10gb/s)

9 storage servers

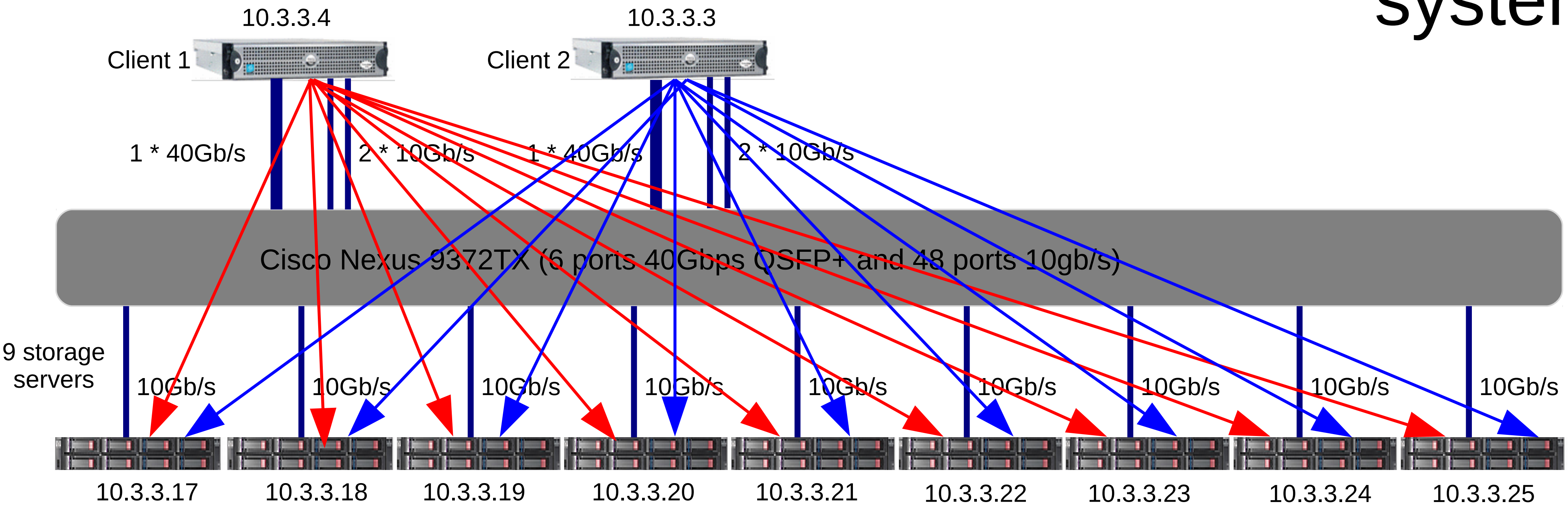
10Gb/s 10Gb/s 10Gb/s 10Gb/s 10Gb/s 10Gb/s 10Gb/s 10Gb/s 10Gb/s

10.3.3.17 10.3.3.18 10.3.3.19 10.3.3.20 10.3.3.21 10.3.3.22 10.3.3.23 10.3.3.24 10.3.3.25

**What is the maximum network bandwidth we can achieve using all the storage servers ?**

- Network bandwidth tests to each storage server (client : 100Gb/s max, storage 90Gb/s max)
  - Individually : 1 flow (TCP or UDP) to 1 server (nuttcp) :
    - TCP client → server : sum of the 9 servers = 87561.23 Mb/s **(7k à 8k TCP retrans / server)**
    - TCP server → client : sum of the 9 servers = 89190.71 Mb/s (0 TCP retrans / serveur)
    - UDP client → server : sum of the 9 servers = 52761.45 Mb/s (83 % à 93 % UDP drop)
    - UDP server → client : sum of the 9 servers = 70709.24 Mb/s (0 drop)
- Needed step : Helped to identify problems not detected until now : bad quality network cables..., servers do not have exactly the same bandwidth, within about 20 %

# 1-c. Network tests with 2 clients and the storage system



**How behave the concurrent flows from 2 clients to the storage ?**

- **Each client sends data to the 9 servers, no writing on disk, only network transmission**
- 2 clients :network cards installed on each client : **1 * 40gb/s + 2* 10gb/s, 120Gb/s max**
  - Simultaneous sending 9 network flows from each 2 clients to the 9 storage servers

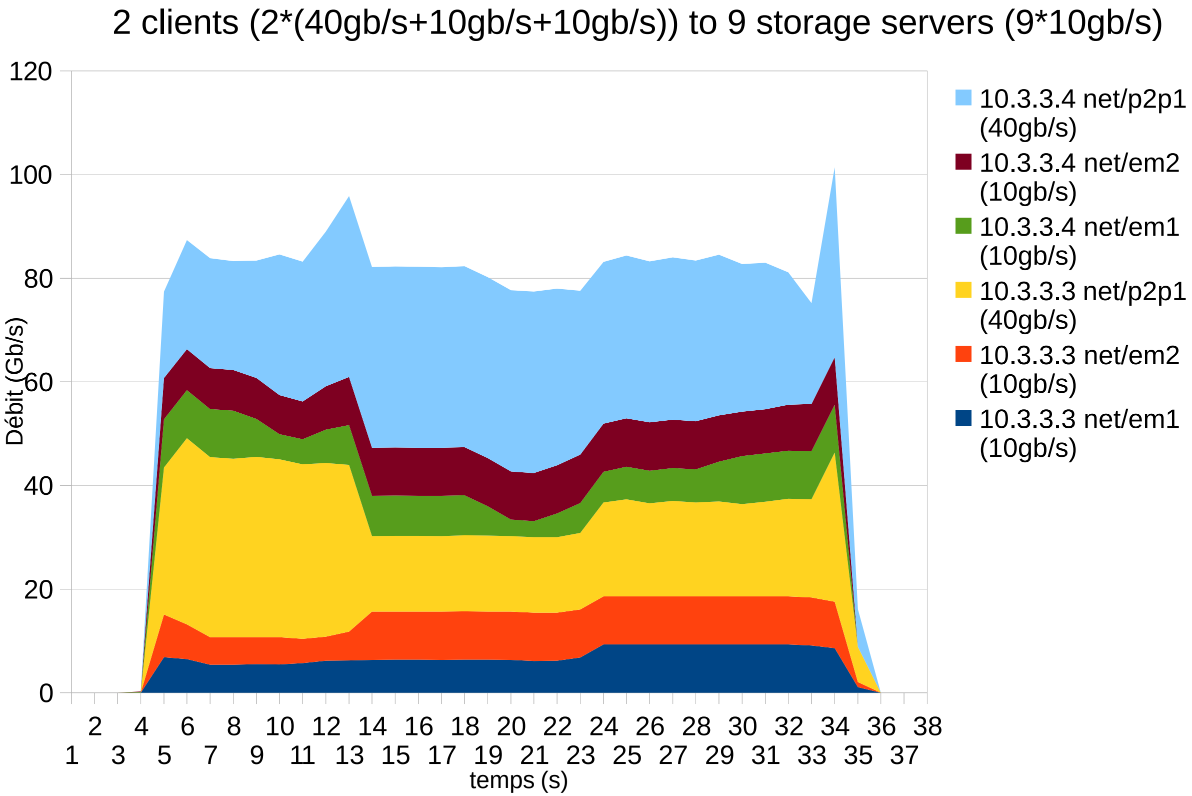
    => the flows pass thru all clients network interfaces (the 40gb/s and the 10gb/s)

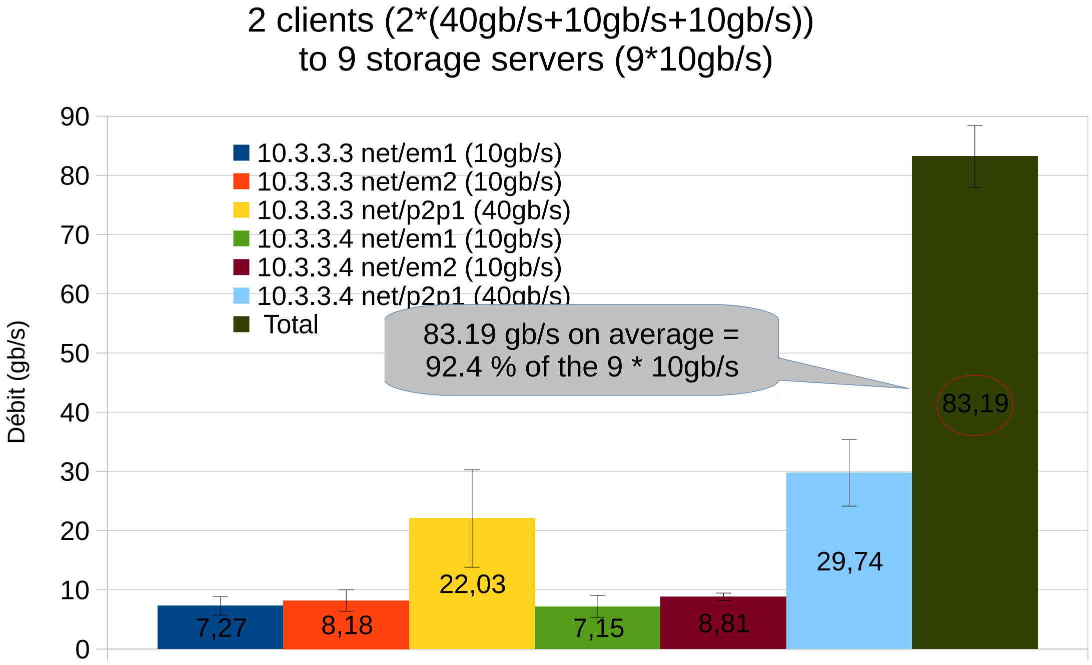
    => 5k à 13k TCP retrans / client and / serveur

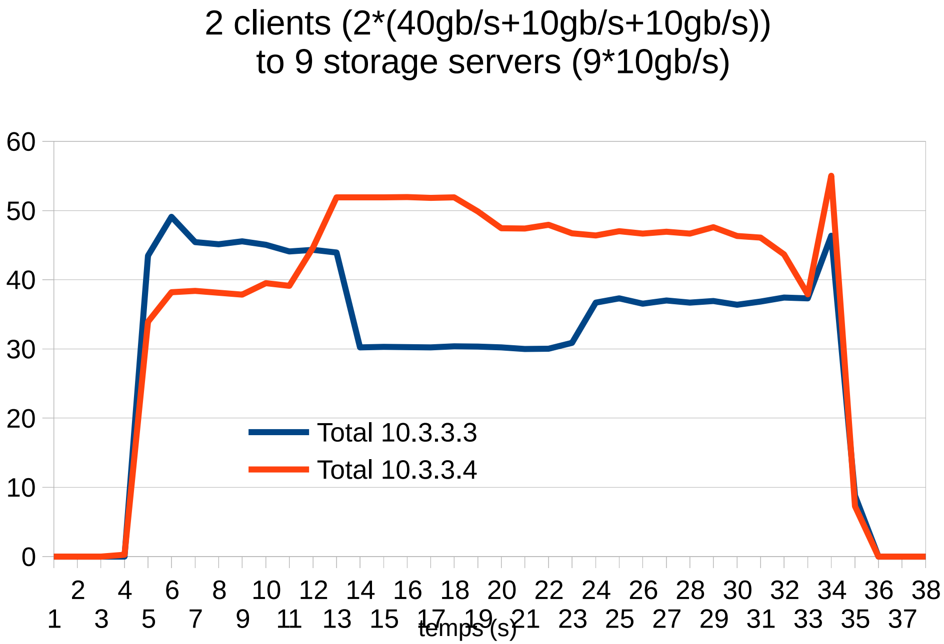
    => the cumulated bandwith of the all 9 storage servers is used at 92.4 % (normalized to total bandwidth in individual transmission of slide 11 in TCP mode)

# Small asymmetry observed for a short period among the two clients

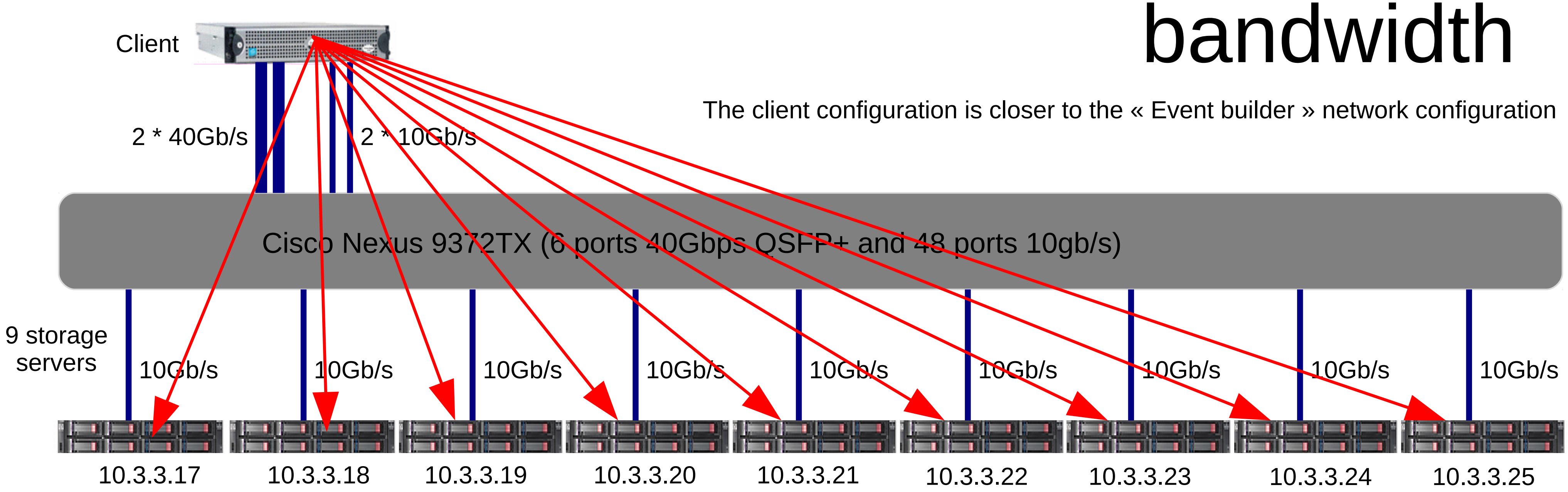
- Traffic distribution test from 2 clients to 9 storage servers : each client is equiped with 1*40gb/s + 2*10gb/s
- mode=balance-alb xmit_hash_policy=layer3+4
- 30 seconds test
- The flows are ditributed on all the network interfaces of the clients
- Client 1 : 37.49 gb/s on average
- Client 2 : 45.7 gb/s on average
- Sum = 83.19 gb/s on average
  = 92.4 % des 9 * 10gb/s
- The trafic distribution between clients (during the time) is no uniform







# 1. and 2. Network tests from 1 client to the storage system with increased bandwidth

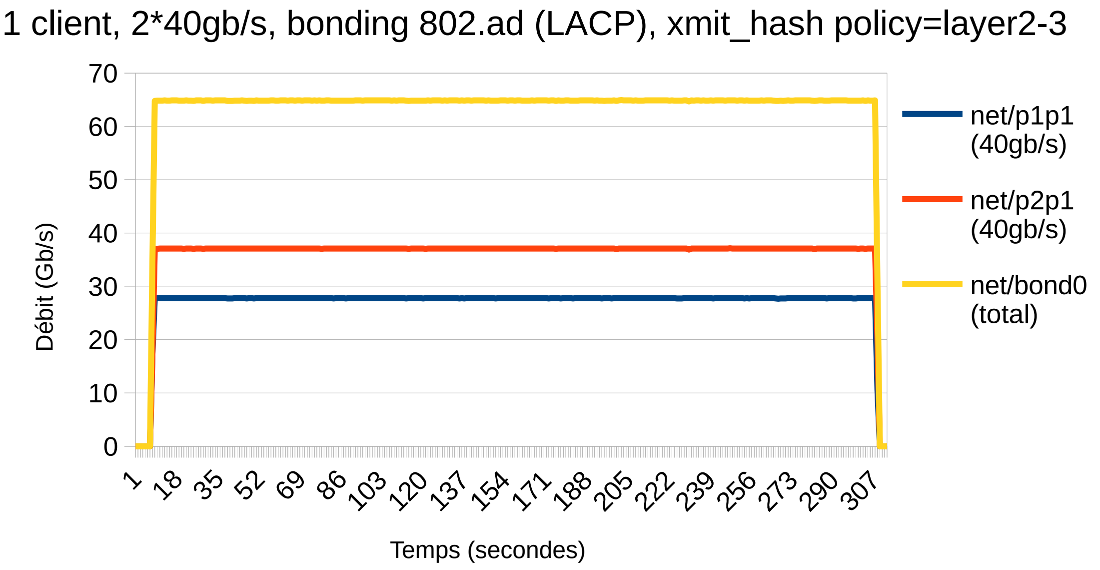


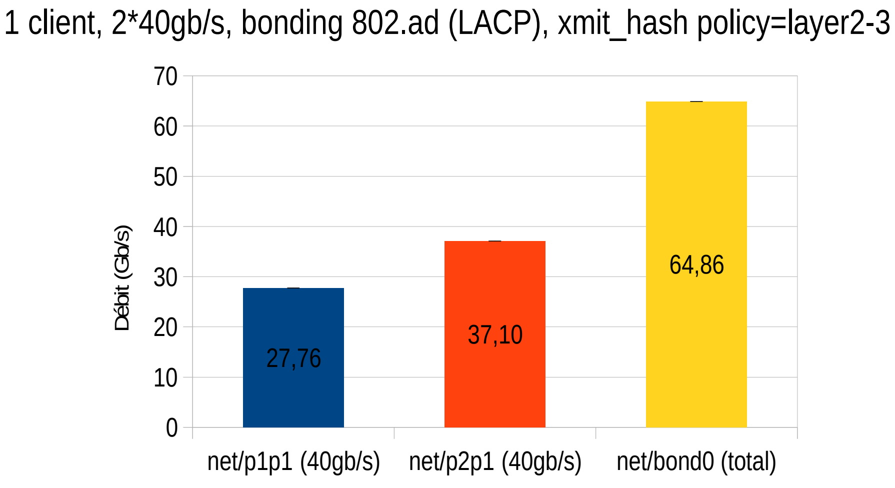
The client configuration is closer to the « Event builder » network configuration

**How behave the bonding (data repartition among different cards) algorithms ?**

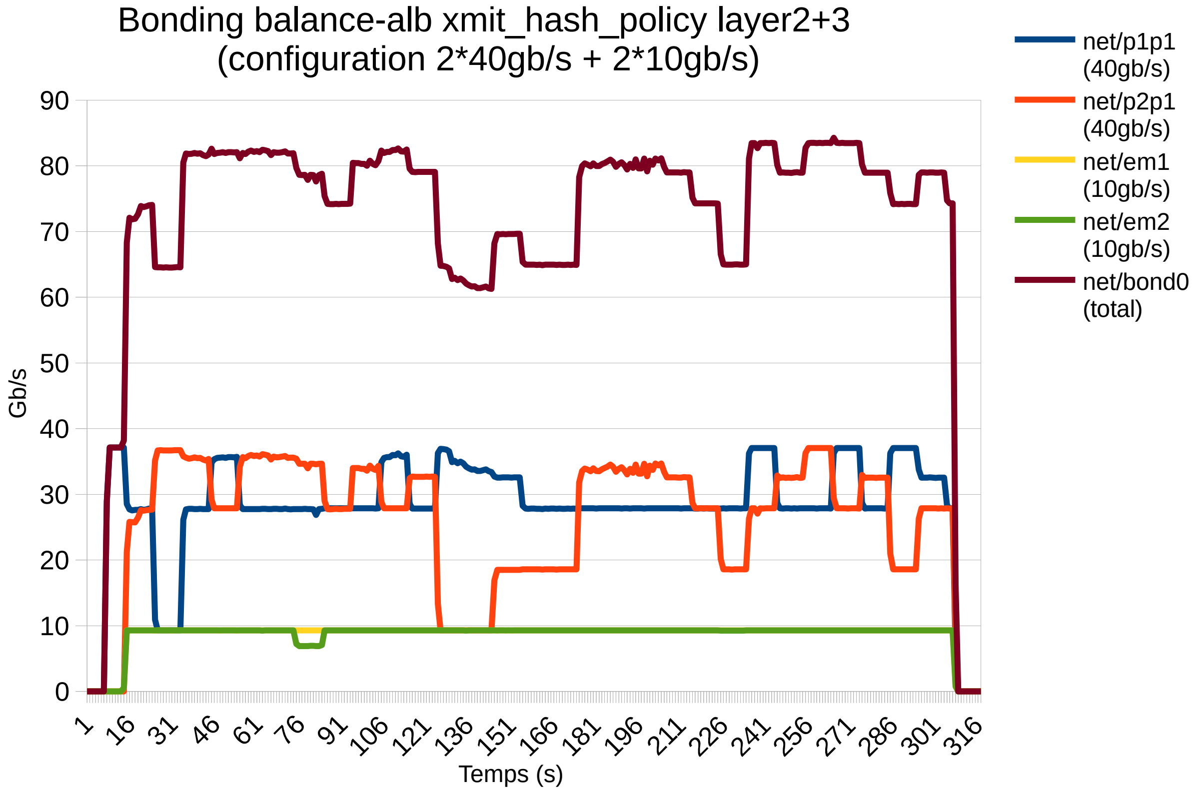
- We are sending 9 simultaneous TCP flows (1 to each server) during 5 minuts (nuttcp)
  - 1st test : we test individually each 40gb/s card → 9 serveurs : 40gb/s card saturation
  - 2nd test : Client Bonding with only 2 * 40gb/s → 9 serveurs :
    - Bonding tested : mode=balance-alb, balance-tlb, LACP
    - High variation measures (except LACP), best = LACP (802.3ad xmit_hash_policy=layer2+3)
  - 3rd test : Client bonding with 2 * 40gb/s + 2 * 10gb/s → 9 serveurs :
    - Bonding tested : mode=balance-alb, balance-tlb but not LACP
    - High variation measures, best = balance-alb xmit_hash_policy layer2+3

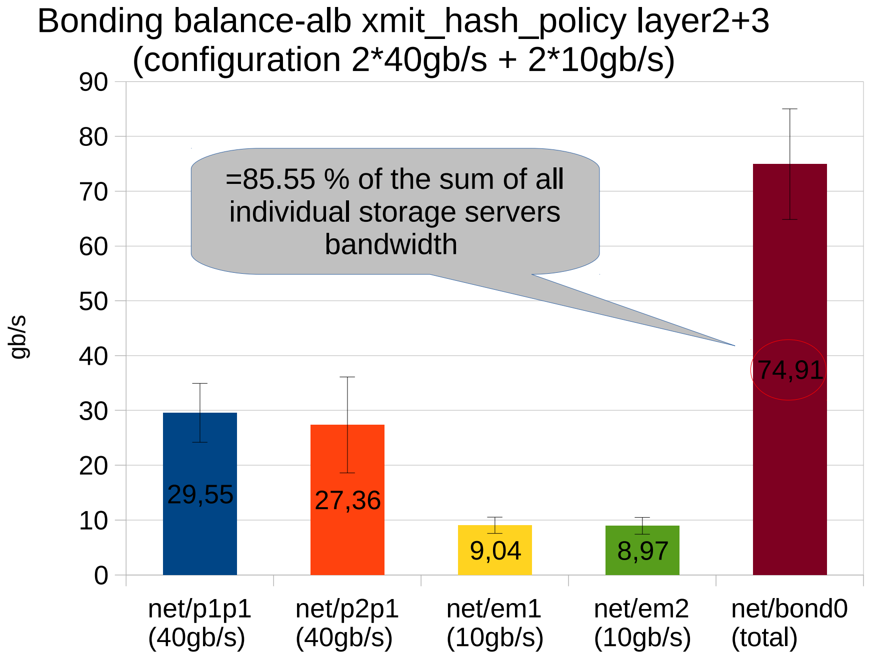
# 2nd test : bonding with 2*40gb/s, best = 802.3ad xmit_hash_policy=layer2+3


1 client, 2*40gb/s, bonding 802.ad (LACP), xmit_hash policy=layer2-3


1 client, 2*40gb/s, bonding 802.ad (LACP), xmit_hash policy=layer2-3

# 3rd test : bonding with 2*40gb/s + 2*10gb/s, best = balance-alb xmit_hash_policy=layer2+3


Bonding balance-alb xmit_hash_policy layer2+3 (configuration 2*40gb/s + 2*10gb/s)


Bonding balance-alb xmit_hash_policy layer2+3 (configuration 2*40gb/s + 2*10gb/s)

=85.55 % of the sum of all individual storage servers bandwidth

# 3. Test of the storage elements

- Storage servers configuration :
  - 1 Raid 6 on 12 2TB hard disks (10 Data + 2 Parity)
  - ~20 TB available on each server
  - Stripe size 1M
- standards tools used :
  - fio (read, write, readwrite, randread, rendwrite, randrw), we choose different size of files and different number of concurrent process
  - iozone (write, read, random-read/write, random_mix) we choose different size of files and different number of concurrent process
  - dd (sync, async, direct…)
- The present challenge is on the writing speed on the storage elements, so we test writing speed on each :
  - test dd (with and without I/O buffer) : sequential writing
    - Without I/O buffer (synchronous) :

```
# dd if=/dev/zero of=test10G.dd bs=1M count=10000 oflag=sync
10485760000 bytes (10 GB) copied, 22,6967 s, 462 MB/s
```

    - With I/O buffer (asynchronous) :

```
# dd if=/dev/zero of=test10G.dd bs=1M count=10000 oflag=direct
10485760000 bytes (10 GB) copied, 9,91637 s, 1,1 GB/s
```

  - Test fio : random write buffered

```
# fio --name=randwrite --ioengine=libaio --iodepth=1 --rw=randwrite --bs=4k
--direct=0 --size=512M --numjobs=8 –runtime=240 --group_reporting

bw=508339KB/s
```

Remember 462 MB/s is the max bandwidth which can be absorbed by a server of this kind

# Storage systems tested

Recall :

| | Lustre | BeeGFS | GlusterFS | GPFS | MooseFS | XtreemFS | XRootD | EOS |
|---|---|---|---|---|---|---|---|---|
| Versions | v2.7.0-3 | v2015.03.r10 | 3.7.8-4 | v4.2.0-1 | 2.0.88-1 | 1.5.1 | 4.3.0-1 | Citrine 4.0.12 |
| POSIX | Yes | Yes | | | | Yes | via FUSE | via FUSE |
| Open Source | Yes | Client=Ye<br>Serveur=EUL | | | | Yes | Yes | Yes |
| Need for MetaData Server ? | Yes | Metadata<br>Ma | No | No | Metadata + Manager | | Yes | Yes |
| Support RDMA / Infiniband | Yes | Yes | Yes | Yes | No | No | No | No |
| Striping | Yes | Yes | Yes | Yes | No | Yes | No | No |
| Failover | M + D (1) | DR (1) | M + D (1) | M + D (1) | M + DR (1) | M + DR (1) | No | M + D (1) |
| Quota | Yes | Yes | Yes | Yes | Yes | No | No | Yes |
| Snapshots | No | No | Yes | Yes | Yes | Yes | No | No |
| Integrated tool to move data over data servers ? | Yes | Yes | Yes | Yes | No | Yes | No | Yes |

Each file is divided into « chunks »
ditributed over all the storage servers
This is always at the charge of the client
CPU (DAQ back-end)

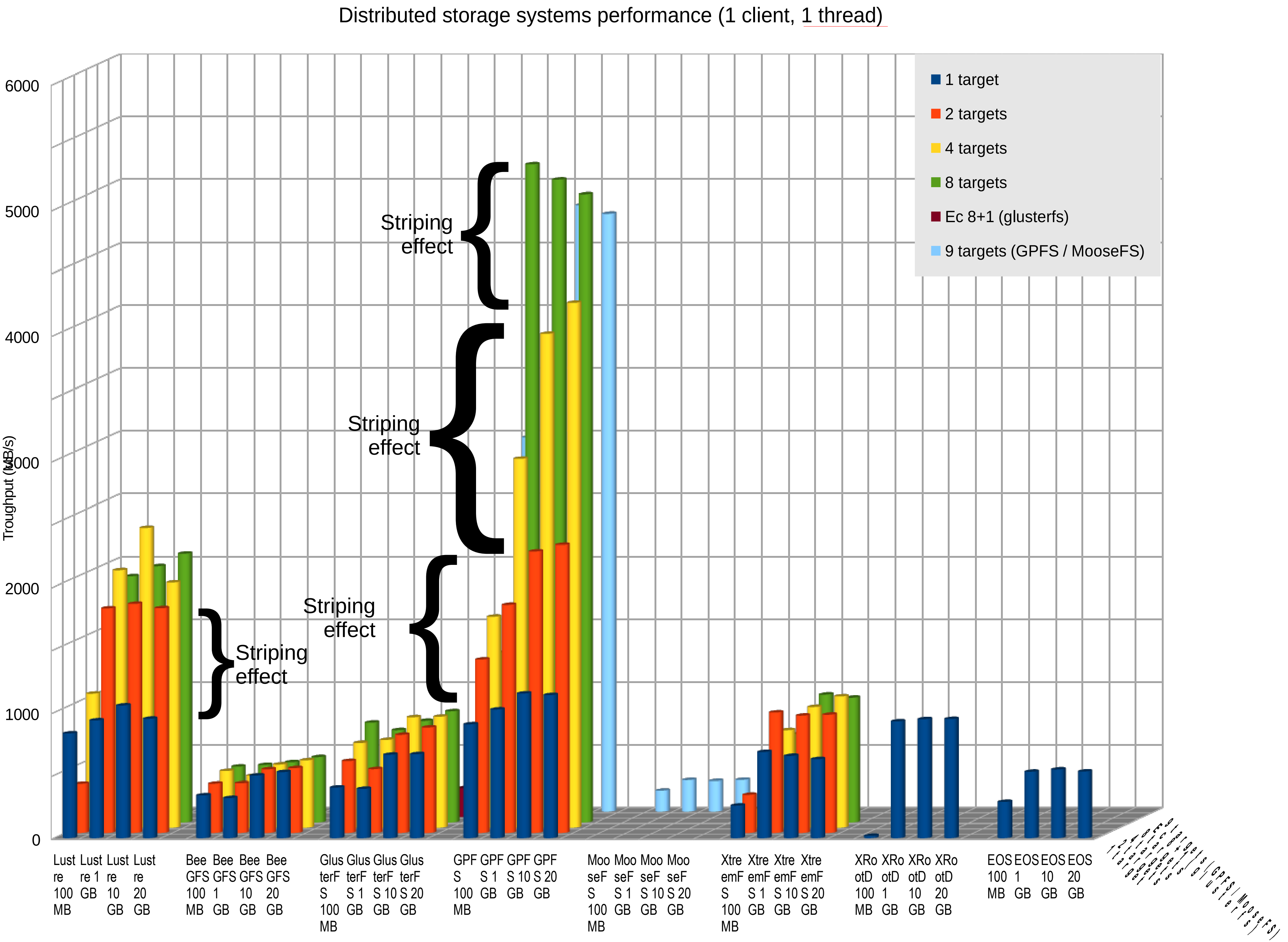
(1) : M=Metadata, D=Data, M+D=Metadata+Data, DR=Data Replication

# Tested parameters common to all storage systems

- Differents parameters :
  - **File size** to be written ? => **choice = 100MB, 1GB, 10GB and 20GB**
    - Needed to determine which file size is optimal,
    - To determine the cost of metada processing :
  - **Flows number** ? Thread(s) number ? Number of process to be launched in // to write data ?
    => **choice = 1, 6, 8**
    1 = to determine the individual flow bandwidth
    6 = number of flows received by 1 « Event Builder »
    8 = number of hyper-threaded cores of my testbed client
  - **Number of chunks** (typical of distributed FS : number of fragments used to write each file in // to multiple storage servers : needed to know the data distribution effect when more than 1 storage server is used)
    **choice => 1, 2, 4, 8**
  - **Number of targets : number of storage servers involved in the writing process of the chunks**

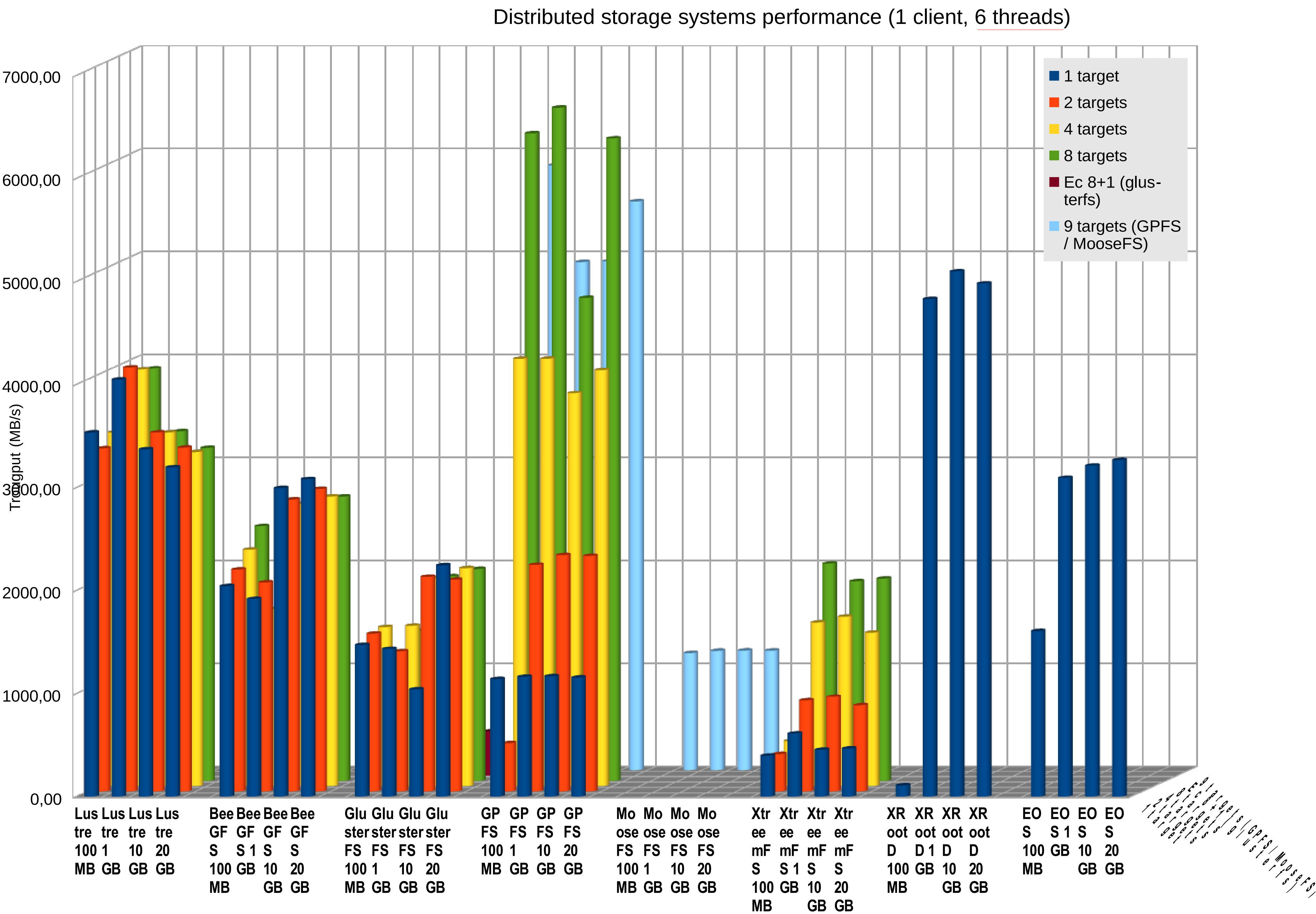
=> 4*3*4 = 48 combinations to be tested

=> 48 combinations * 8 Storage Systems = 384 tests in final

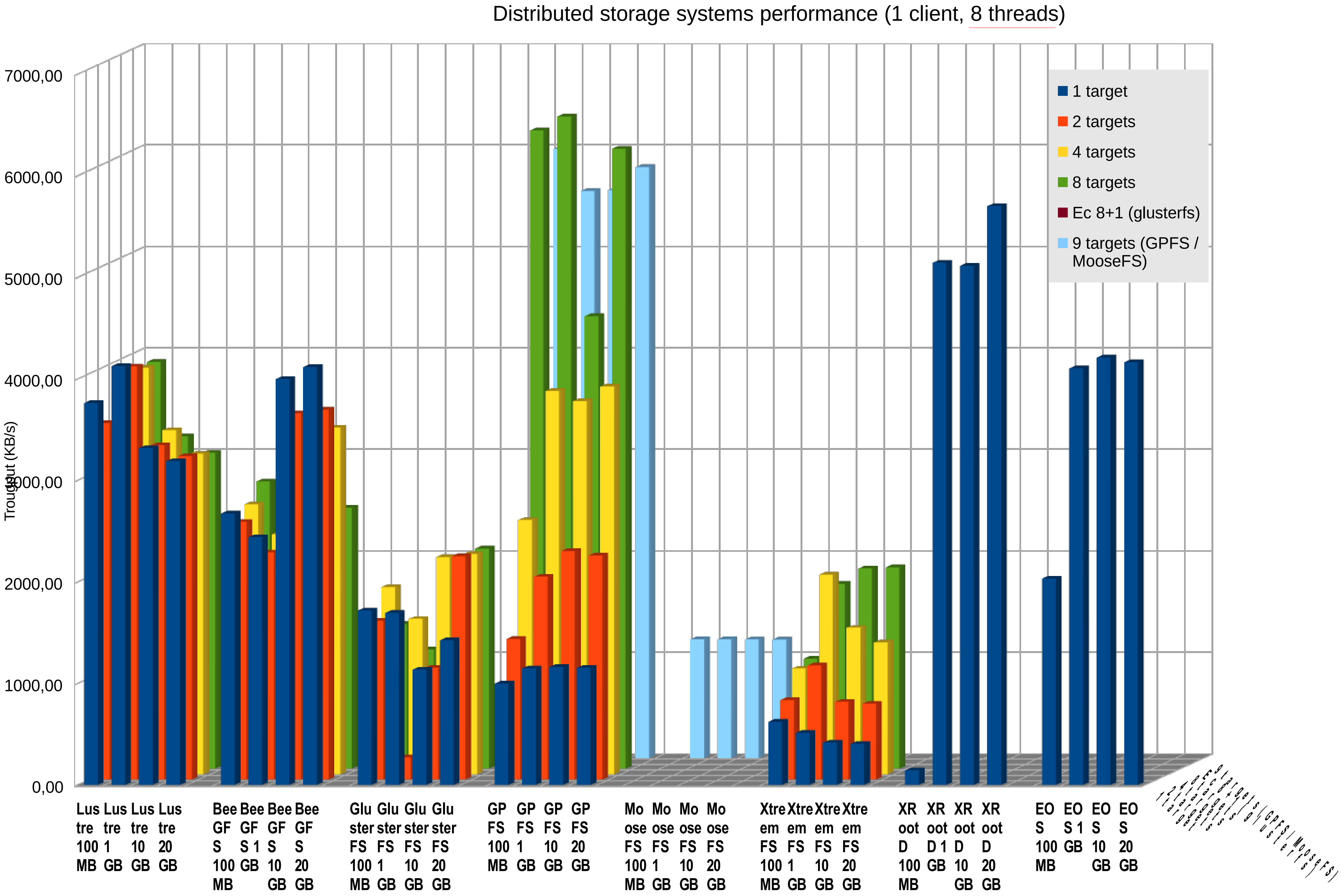


Distributed storage systems performance (1 client, 1 thread)

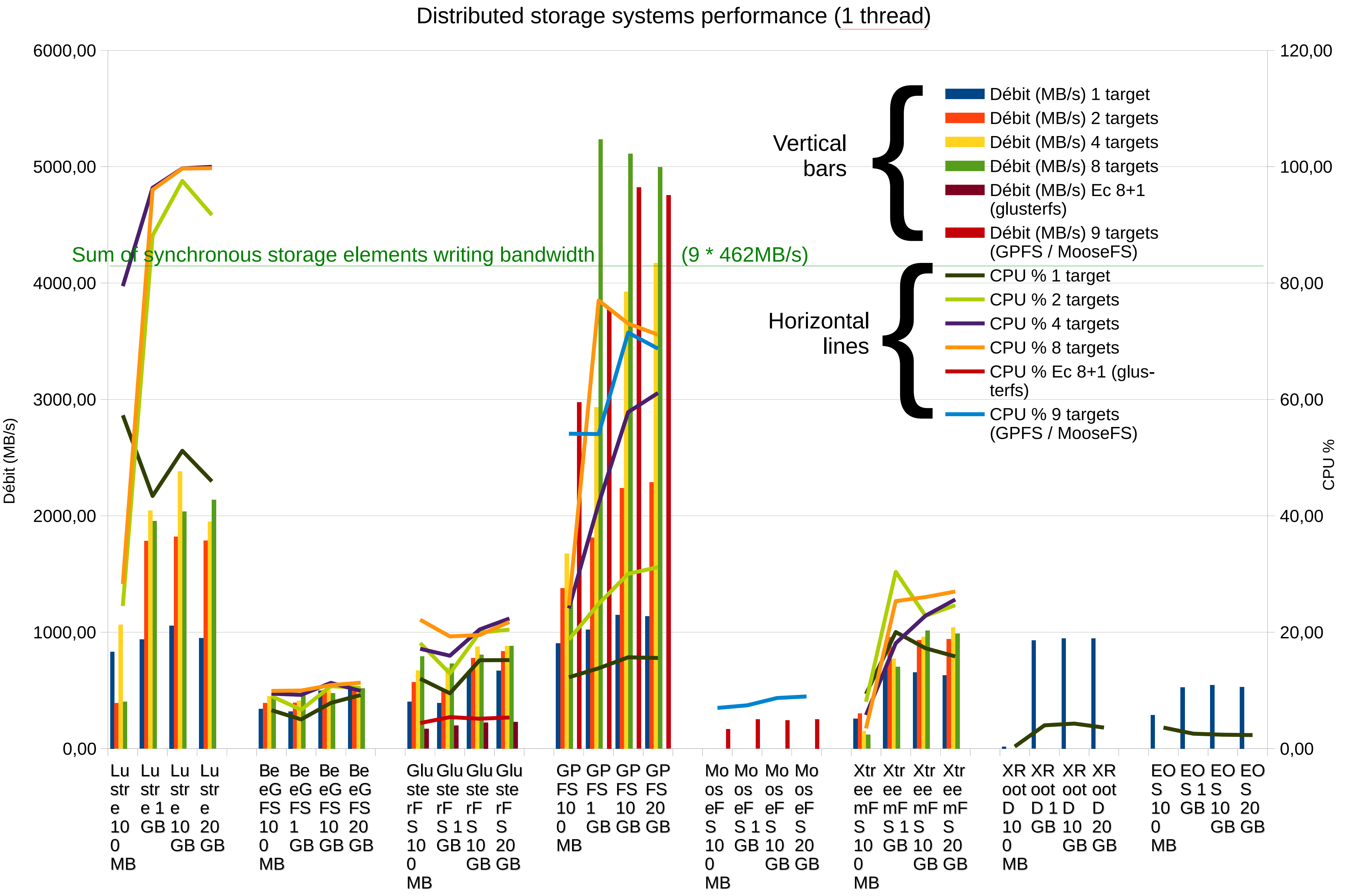
# → Bottleneck in previous slide due to serialization of wtiting by only one thread

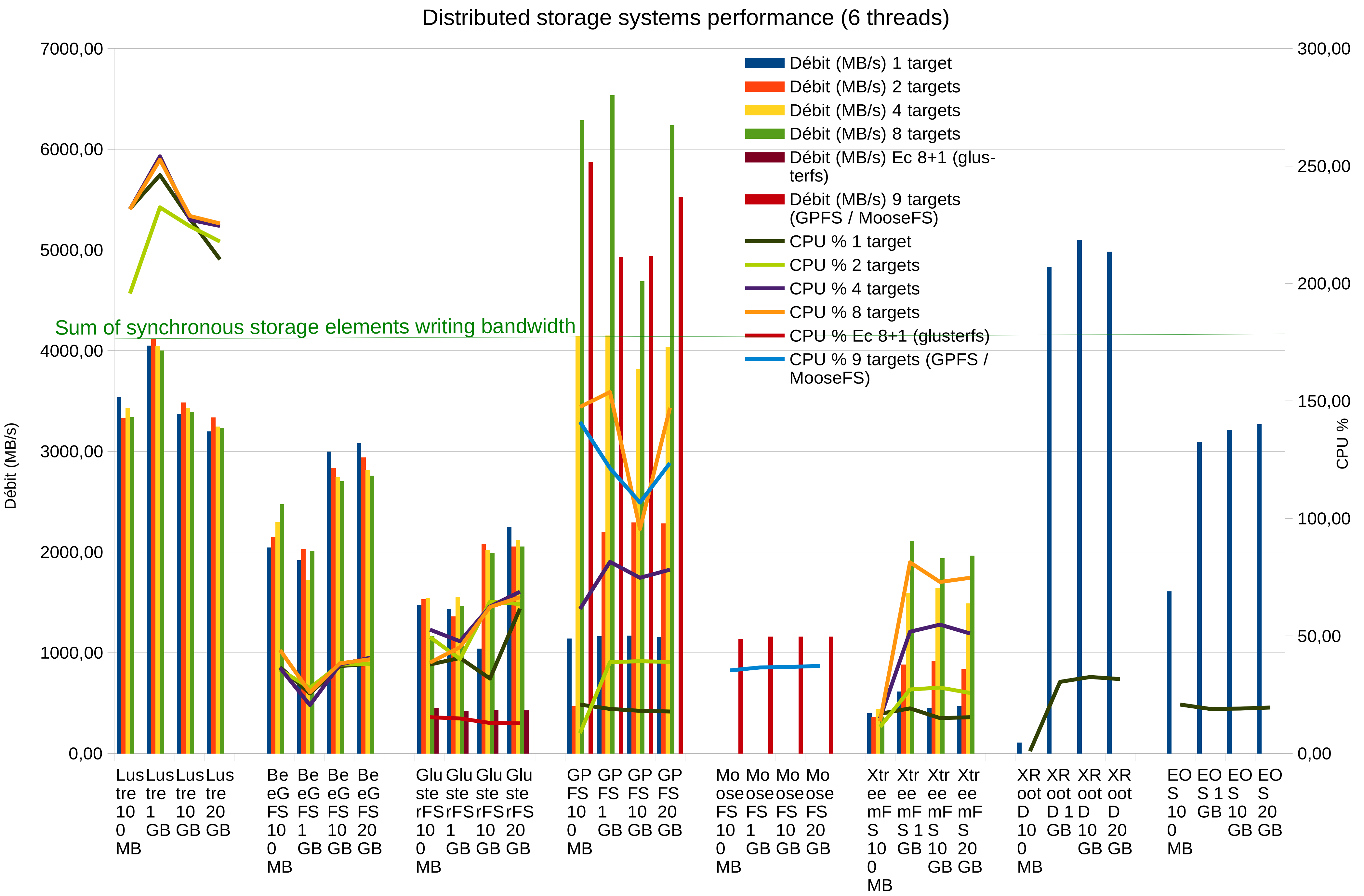

Distributed storage systems performance (1 client, 6 threads)

# All client cores used

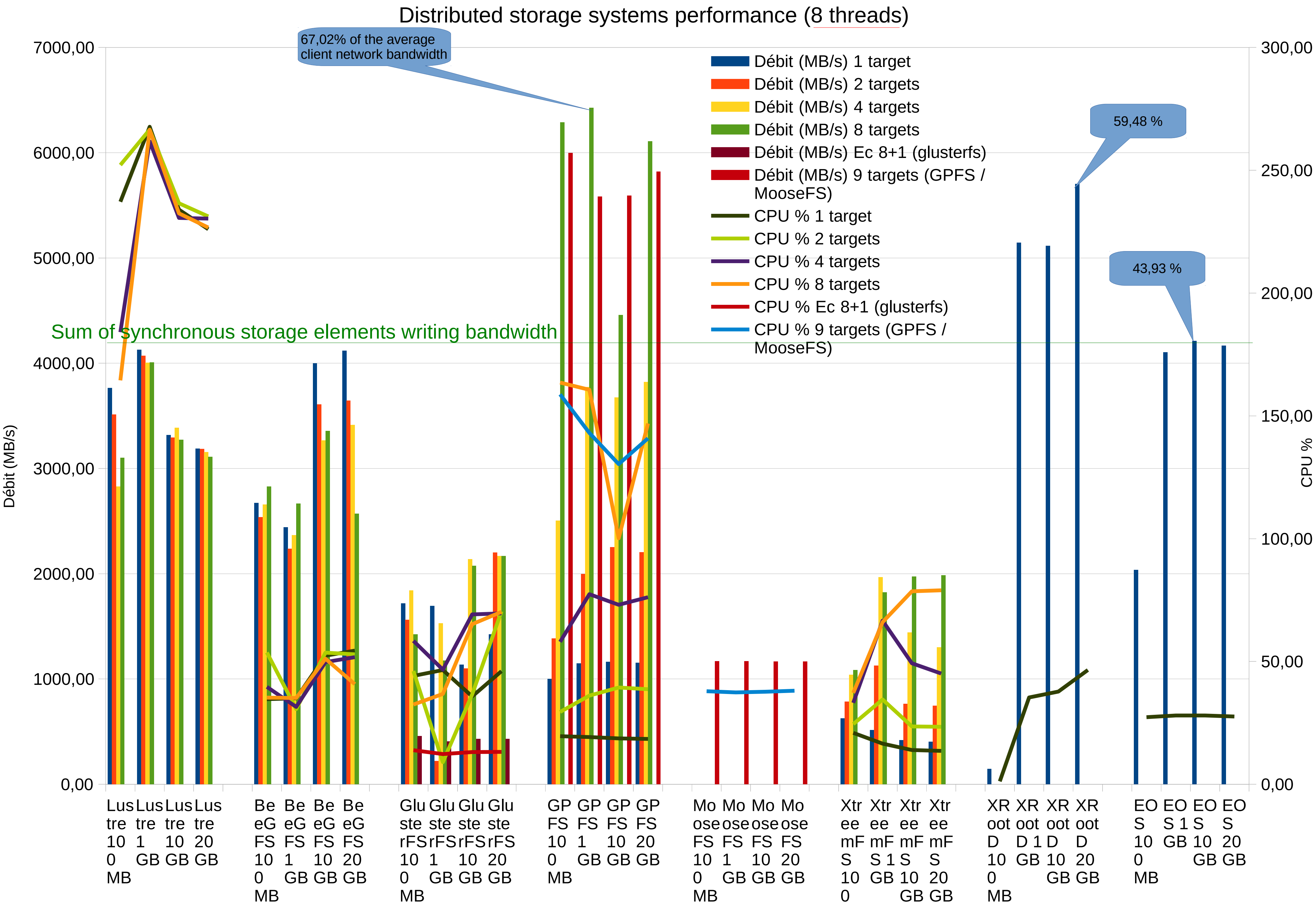


Distributed storage systems performance (1 client, 8 threads)


Distributed storage systems performance (1 thread)
Débit (MB/s)
CPU %
6000,00
5000,00
4000,00
3000,00
2000,00
1000,00
0,00
120,00
100,00
80,00
60,00
40,00
20,00
0,00
Sum of synchronous storage elements writing bandwidth (9 * 462MB/s)
Vertical bars
Horizontal lines
Débit (MB/s) 1 target
Débit (MB/s) 2 targets
Débit (MB/s) 4 targets
Débit (MB/s) 8 targets
Débit (MB/s) Ec 8+1 (glusterfs)
Débit (MB/s) 9 targets (GPFS / MooseFS)
CPU % 1 target
CPU % 2 targets
CPU % 4 targets
CPU % 8 targets
CPU % Ec 8+1 (glusterfs)
CPU % 9 targets (GPFS / MooseFS)
Lustre 100 MB
Lustre 1 GB
Lustre 10 GB
Lustre 20 GB
BeeGFS 100 MB
BeeGFS 1 GB
BeeGFS 10 GB
BeeGFS 20 GB
GlusterFS 100 MB
GlusterFS 1 GB
GlusterFS 10 GB
GlusterFS 20 GB
GPFS 100 MB
GPFS 1 GB
GPFS 10 GB
GPFS 20 GB
MooseFS 100 MB
MooseFS 1 GB
MooseFS 10 GB
MooseFS 20 GB
XtreemFS 100 MB
XtreemFS 1 GB
XtreemFS 10 GB
XtreemFS 20 GB
XRootD 100 MB
XRootD 1 GB
XRootD 10 GB
XRootD 20 GB
EOS 100 MB
EOS 1 GB
EOS 10 GB
EOS 20 GB



Distributed storage systems performance (6 threads)


Distributed storage systems performance (8 threads)
67,02% of the average client network bandwidth
59,48 %
43,93 %
Sum of synchronous storage elements writing bandwidth
Débit (MB/s) 1 target
Débit (MB/s) 2 targets
Débit (MB/s) 4 targets
Débit (MB/s) 8 targets
Débit (MB/s) Ec 8+1 (glusterfs)
Débit (MB/s) 9 targets (GPFS / MooseFS)
CPU % 1 target
CPU % 2 targets
CPU % 4 targets
CPU % 8 targets
CPU % Ec 8+1 (glusterfs)
CPU % 9 targets (GPFS / MooseFS)
Débit (MB/s)
CPU %
7000,00
6000,00
5000,00
4000,00
3000,00
2000,00
1000,00
0,00
300,00
250,00
200,00
150,00
100,00
50,00
0,00
Lustre 100 MB
Lustre 1 GB
Lustre 10 GB
Lustre 20 GB
BeeGFS 100 MB
BeeGFS 1 GB
BeeGFS 10 GB
BeeGFS 20 GB
GlusterFS 100 MB
GlusterFS 1 GB
GlusterFS 10 GB
GlusterFS 20 GB
GPFS 100 MB
GPFS 1 GB
GPFS 10 GB
GPFS 20 GB
MooseFS 100 MB
MooseFS 1 GB
MooseFS 10 GB
MooseFS 20 GB
XtreemFS 100 MB
XtreemFS 1 GB
XtreemFS 10 GB
XtreemFS 20 GB
XRootD 100 MB
XRootD 1 GB
XRootD 10 GB
XRootD 20 GB
EOS 100 MB
EOS 1 GB
EOS 10 GB
EOS 20 GB

# Technical detailed conclusions

- Classification :
  - High performance filesystems : GPFS, Lustre, BeeGFS
  - Massive storage systems : XRootD et EOS are also well adapted
- Conclusion of all the tests :
  - We hit the limits of storage system testbed : old hardware (5 years old storage servers), not a high end server for the client.
  - Not tested : acquisition phase concurrent with online analysis phase <=> high speed writing and concurrent readling files
  - Network tests :
    - 40gb/s -> 10gb/s : some inefficency : TCP retransmissions and UDP drops
    - Recommendation :
      - prefer same network interface speed on all systems 40gb/s -> 40gb/s, 56gb/s -> 56gb/s…
      - Prefer LACP (IEEE 802.3ad) more efficient than the other algorithms (when the interfaces have the same speed)
  - Acquisition :
    - To improve the client bandwidth => distribute the acquisition flow on several process
    - To distribute the I/O to all the storage elements => create several network flows to record data into the storage system
  - The I/O parallelization (chuncks distributed over all the storage servers) :
    - Provides a gain only for a small number of clients or a small number of data flows (1, 2, 3..?)
    - Has no effect for 6 or 8 independent flows
  - The POSIX distributed storage systems :
    - Large differences in performance : Negative impact of fuse (unusable in our case)
    - GPFS very effective (it use al the hardware ressources), but the problem of the cost of the license (€€€)
    - Lustre and BeeGFS are also effective, but Lustre use heavily the client CPU (at least for the version 2.7.0)
  - The POSIX layer need CPU of the client, the non POSIX storage systems :
    - Benefit forXrootD et EOS : they don't provide the POSIX layer, they need little CPU power (they just open network sockets)
    - XrootD is high performance (files > 1Go) : performance problem for small files (100Mo), metadata penalty
    - EOS was less efficient than XrootD but has more exciting features for production (lifecycle of data and of storage servers)

# Summary conclusions

- Conclusion of all the network and storage tests :
  - Acquisition :
    - When possible : distribute the acquisition flow on several independent processes (ideal ratio : 1 acquisition flow / CPU core)
    - When possible : to distribute the load on the storage system, create as many independent network flows as possible (ideal ratio : 1 network flow per storage server)
  - Network tests :
    - Prefer to use the same network interface speed on all systems : 40gb/s -> 40gb/s...
    - Prefer LACP (IEEE 802.3ad) : it is more efficient than the other algorithms (when the interfaces have the same speed)
  - 4 bests candidated shown by the performance tests : **GPFS**, Lustre, **XRootD** and EOS.
    - GPFS very effective (it use all the hardware ressources), but the problem is the cost of the annual license (€€€)
    - Lustre need far more CPU than the others
    - XrootD is very effective (as GPFS)
    - EOS is less efficient than XrootD but has features well designed for production storage systems
    - Suggestion : XrootD or EOS
  - Data files on the storage systems :
    - Do not create small files (because of metadata penalty) : create at least > 1GB / file
    - But not too big : due to storage constraints on worker nodes in the online/offline analysis phases (< 20GB / file ?)

# Thanks to

- **Telindus / SFR for the switch loan (6 weeks)**
- R. Barbier (IPNL/EBCMOS), B. Carlus (IPNL/WA105) & J. Marteau (IPNL/WA105) for the Mellanox 40gb/s loan
- The IPNL's CMS team for temporary use of the 9 Dell R510 before the LHC's RUN 2 data taking
- L-M Dansac (Univ-Lyon 1/CRAL) for temporary use of a Dell R630
- C. Perra (Univ-lyon 1/FLCHP), Y. Calas (CC-IN2P3), L. Tortay (CC-IN2P3), B. Delaunay (CC-IN2P3), J-M. Barbet (SUBATECH), A-J. Peters (CERN) for the help

# Links / bibliography

- Storage systems :
  - GPFS : https://www.ibm.com/support/knowledgecenter/SSFKCN/gpfs_welcome.html
  - Lustre : http://lustre.org/
  - BeeGFS :
    - http://www.beegfs.com/content
    - http://www.beegfs.com/docs/Introduction_to_BeeGFS_by_ThinkParQ.pdf
  - GlusterFS : https://www.gluster.org
  - MooseFS : https://moosefs.com
  - XtreemFS :
    - http://www.xtreemfs.org
    - http://www.xtreemfs.org/xtfs-guide-1.5.1.pdf
  - XrootD : http://xrootd.org
  - EOS : http://eos.readthedocs.io/en/latest
- Bonding : https://www.kernel.org/doc/Documentation/networking/bonding.txt
- System, network and Mellanox tuning :
  - http://www.mellanox.com/related-docs/prod_software/MLNX_EN_Linux_README.txt
  - http://supercomputing.caltech.edu/docs/Chep2012_40GEKit_azher.pdf
  - http://www.nas.nasa.gov/assets/pdf/papers/40_Gig_Whitepaper_11-2013.pdf
  - https://access.redhat.com/sites/default/files/attachments/20150325_network_performance_tuning.pdf
  - https://fasterdata.es.net/host-tuning/40g-tuning/
- The new CMS DAQ system for LHC operation after 2014 (DAQ2) :
  - http://iopscience.iop.org/article/10.1088/1742-6596/513/1/012014/pdf