PUGNÈRE Denis
CNRS / IN2P3 / IPNL

Méthodologie de tests de performance de systèmes de stockage

ANF User Support Tools 4 HPC : 14-18 mai 2018, Fréjus




Qui suis-je ?

- Ingénieur système à l'Institut de Physique Nucléaire de Lyon (IPNL : laboratoire CNRS/IN2P3 + Université Claude Bernard Lyon 1, ~230 personnes)
- Responsable technique Tier3 IPNL (1700 coeurs, 800TB, collaboration WLCG)
- Architecte du système de stockage et de traitement « Online » des données de l'expérience ProtoDUNE Dual Phase @ CERN

Plan de la présentation

- Quel est l'objectif des tests ?
- Méthodologie proposée :
  - Définir le besoin
  - Connaitre la capacité de son infrastructure
  - Indicateurs mesurés et outils (µ-benchmarks)
  - Mettre en œuvre les tests (système complet)
  - Comparaison des résultats
- Nouveaux paradigmes
- Critères de choix d’un système de ~~stockage~~ traitement de données
- Conclusion

Quel est l'objectif des tests ?

- Monter un PoC ?
- Réaliser des tests fonctionnels ?
- Comparer différents systèmes / technologies ?
- Résoudre un problème de performance ?
  Comprendre les I/O bottlenecks dans le service :
  - revient à chercher quelle est la cause de l'I/O WAIT !
  - donne des pistes pour tourner le bon bouton

Méthodologie proposée (1/5) :
Définir le besoin

- Les tests sont à adapter selon la finalité du système de stockage :
  - Stockage :
    - POSIX : $HOME ? /scratch ? <- les besoins sont différents
    - De fichiers non POSIX,
    - Stockage d'objets (key <-> value),
    - Stockage de blocks, VM.
  - Stockage massif : Acquisition de données ? Archivage ?
  - Distribution géographique des données ?
- Selon les types de traitements à réaliser sur ces données :
  - HTC (High Thougput Computing), data mining, « analytics » : I/O dominant
  - HPC (High Performance Computing) : CPU dominant
  - Streaming : traitement des données au fil de l'eau, déclenchement de traitements sur événements ...
- Selon le pattern d'accès :
  - get / put, fetch / store / update
  - SQL query versus NoSQL
  - Map-reduce (traitement -> agrégation -> réduction) : pré-indexé dans les workers
- Tenir compte : Du type, du volume, de la criticité des données et du type d'accès à ces données

=> Décrire le workflow : construire la matrice de flux et les contraintes matérielles et temporelles

=> Mettre en place les scénarios de tests

Méthodologie proposée (2/5) :
Connaitre la capacité de son infrastructure

Principe des tests de base :

- Commencer par le plus bas niveau

- µ-benchmarks sur chaque élément du système : client(s), serveur(s), réseau

- **Tests individuels de chaque client :**
  - I/O : comportement en R, R/W, W, sequentiel, random, mixed
  - Réseau : receive, transmit, protocole (UDP, TCP), % drop, % retrans
    - si nécessaire en utilisant **+sieurs serveurs de stockage** pour tester 1 client

- **Tests individuel de chaque serveur de stockage :**
  - Configuration : RAID type (JBOD, RAID 0, 1, 5, 6, software, hardware...), block size, filesystem (EXTx, XFS, ZFS...)
  - I/O : comportement en R, R/W, W, sequentiel, random, mixed
  - Réseau : receive, transmit, protocole (UDP, TCP), %drop, % retrans,
    - si nécessaire en utilisant **+sieurs clients** pour tester 1 serveur de stockage

- **Tests réseau :**
  - Test de saturation de l'infrastructure réseau : tous clients <-> tous serveurs

- **Tests de charge système :**
  - charge (CPU, interruptions, IOPS, IOWAIT…) du|des client(s) : en fonction nb threads / client
  - charge (CPU, interruptions, IOPS, IOWAIT…) du|des serveur(s) : en fonction nb clients / serveur

Méthodologie proposée (3/5) :
Indicateurs mesurés et exemples d'outils

Type	Scénarios	Paramètres mesurés	Utilitaires
Entrèes / Sorties (block devices, systèmes de fichiers...)	comportement en R, R/W, W, sequentiel, random, mixed	Débit, IOPS, latence	fio, iozone, dd, iops, blktrace, seekwatcher, btrecord, btreplay, dstat...
Réseau	Réception, émission,% perte, protocoles (TCP, UDP)	Débit, nb paquets, protocole (% UDP drop, % TCP retrans)	iperf, nuttcp, dstat, systemtap...
Paramètres système	Charge client, charge serveur	% CPU user, % CPU system, IOWAIT, interrupts, context swiches	dstat, systemtap...

Remarques générales :

- Instrumenter tous les systèmes en oeuvre permet d'avoir une vue globale
- Lancer l'enregistrement des paramètres sur tous les systèmes en même temps permet d'obtenir une corrélation temporelle des événements
- **Relancer les tests plusieurs fois**

Exemple : Méthodologie adoptée pour un test de type « écriture intensive »

1 : Tests réseau

+

2 : Tests du stockage

+

3 : Tests de la chaine complète en fonction du workflow de l'expérience

Tests principalement réseau et protocolaires : <u>clients et serveurs</u>

- Protocoles TCP / UDP : iperf, nuttcp...
- Saturation des interfaces : Algorithmes de contrôle de congestion : cubic, reno, bic, htcp...
- UDP : % de perte de paquets
- TCP : nombre de retransmissions
- Drop de paquets
- Puis reste des tests en TCP, flux en écriture
- Optimisations :
  - Bonding réseau : test des algorithmes LACP, balance-alb, balance-tlb
  - Optimisation buffers réseau : modif /etc/sysctl.conf
  - Utilisation de Jumbo frames (MTU 9216)
  - Charge CPU : répartition des IRQ sur tous les coeurs
    - chkconfig irqbalance off ; service irqbalance stop
    - Mellanox : set_irq_affinity.sh p2p1
- Test initiaux => optimisations => caractérisation

Tests individuels des éléments de stockage :

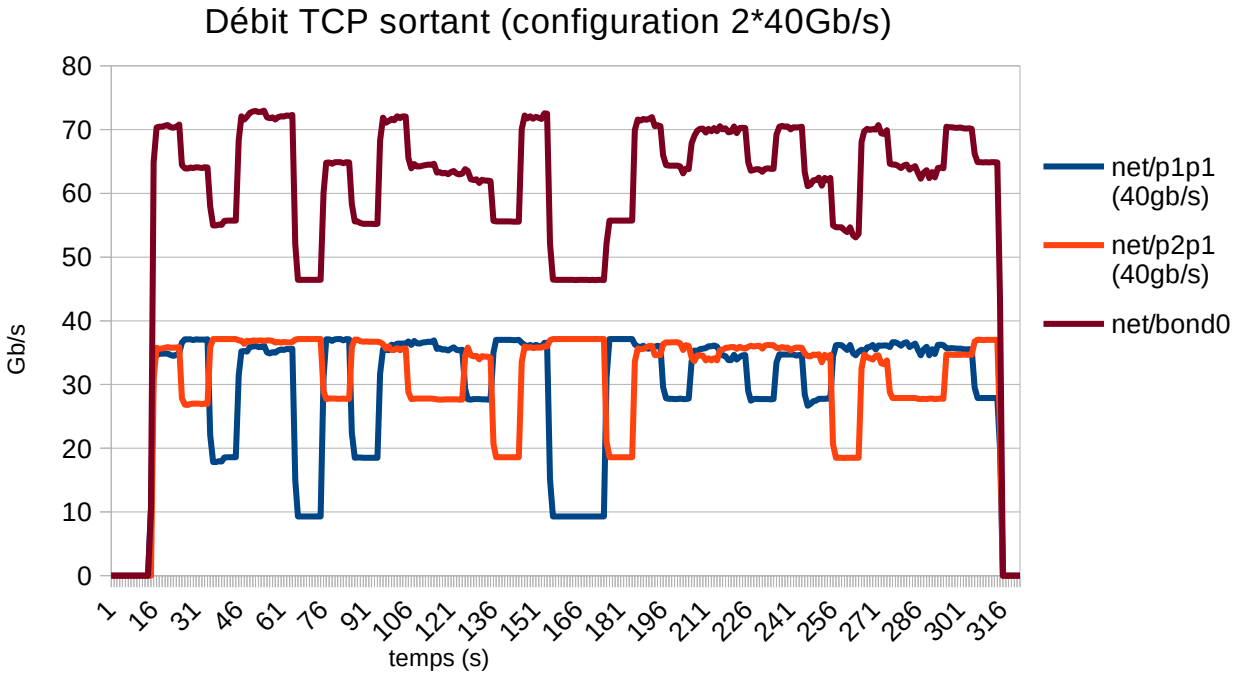
- Iozone / fio / dd : benchmark du système de fichier local
- Benchmark du software de stockage (dans le cas d'un SDS)

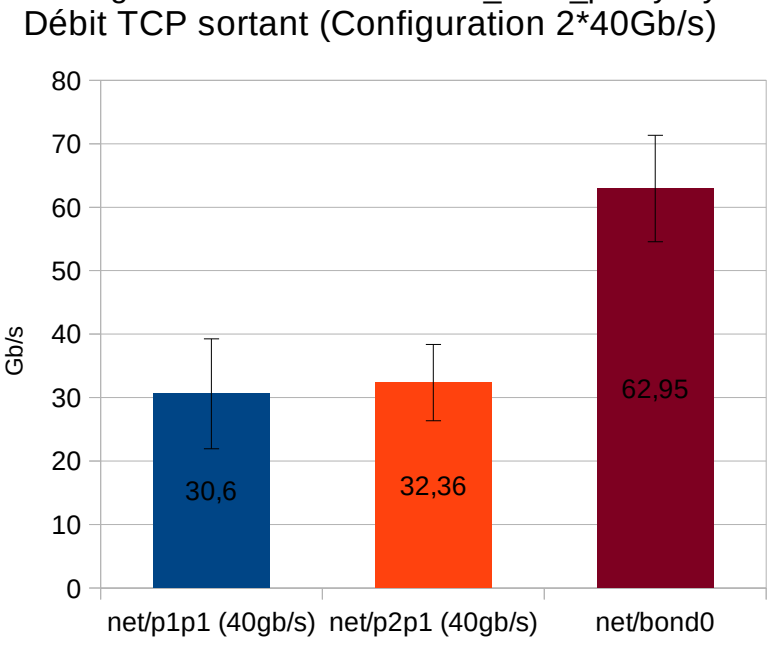
Tests de la chaine complète :

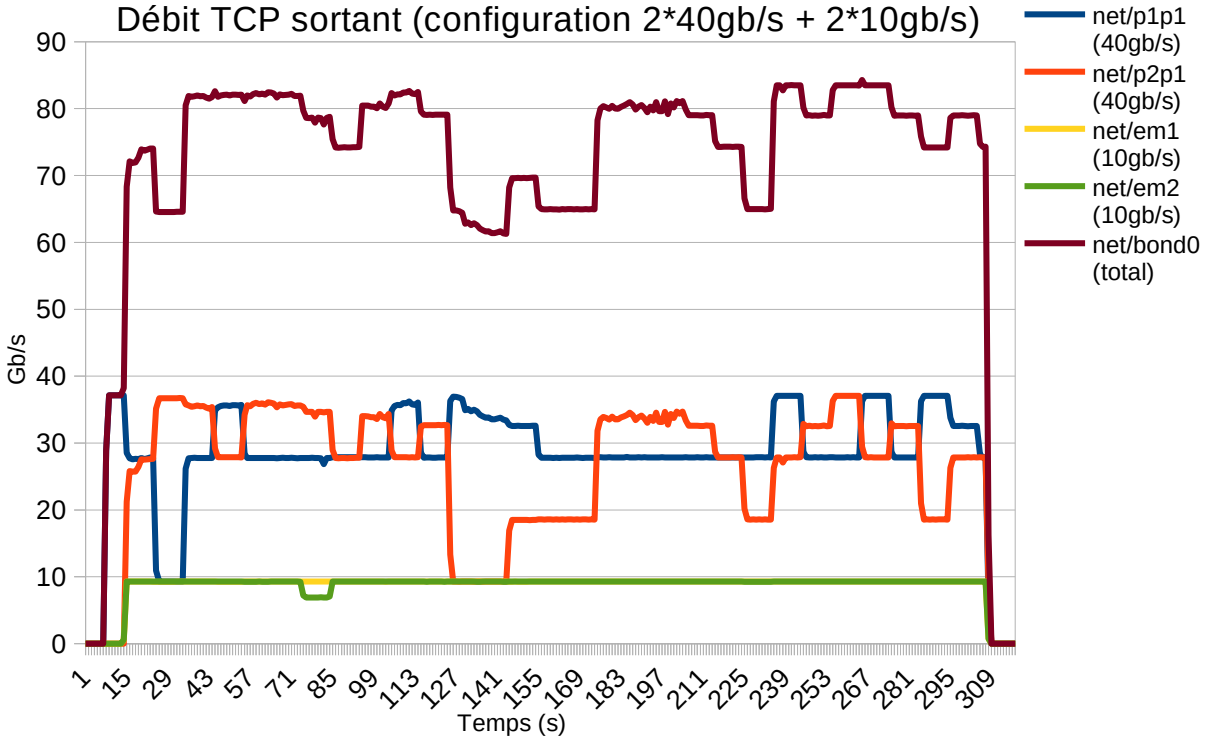
- Sur le client
  - Stockage : Iozone, fio, dd, xrdcp
  - Réseau / système : dstat
- Sur les éléments de stockage : dstat

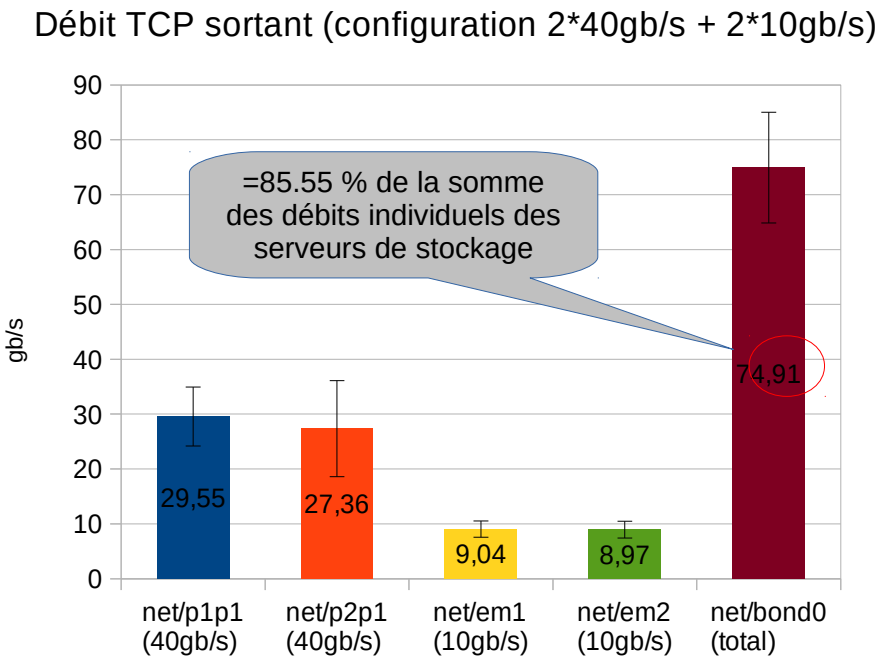
Exemple de résultats de tests réseaux obtenus

Configuration bonding : mode=balance-alb xmit_hash_policy layer2+3









Méthodologie proposée (4/5) :
Mise en œuvre les tests complets

- Jeu de tests pour déterminer le comportement **d'un système de stockage complet** en fonction de la taille des fichiers et du nombre de flux // :

```
for size in 100M 1G 10G 20G 100G
do
  for profile in write read randwrite randread readwrite randrw
  do
    for threads in 1 6 8 10
    do
      # start recording statistics
      # launch 5 minutes test
      # stop recording statistics
      # flush caches
    done
  done
done
```

- Comportement **d'un système de stockage complet** en fonction du nombre d'accès concurrents :

```
for write_flow_number in $(seq 1 5)
do
  for read_flow_number in $(seq 0 10)
  do
      # start recording statistics
      # launch 5 minutes test
      # stop recording statistics
      # flush caches
  done
done
```

Méthodologie proposée (5/5) :
Mise en œuvre les tests complets (non POSIX)

Exemple pour stockage de type XrootD comme EOS : utilisation du client xrdcp (client XrootD). Méthode similaire pour irods, webdav, GridFTP...

Méthode de préparation sur le(s) client(s) :

- Création d'un RAMDISK de 21Go (et montage sur `/test-ramdisk`)
- Création du fichier de la taille souhaitée dans le RAMDISK

- Boucles avec copie du fichier avec la commande xrdcp :

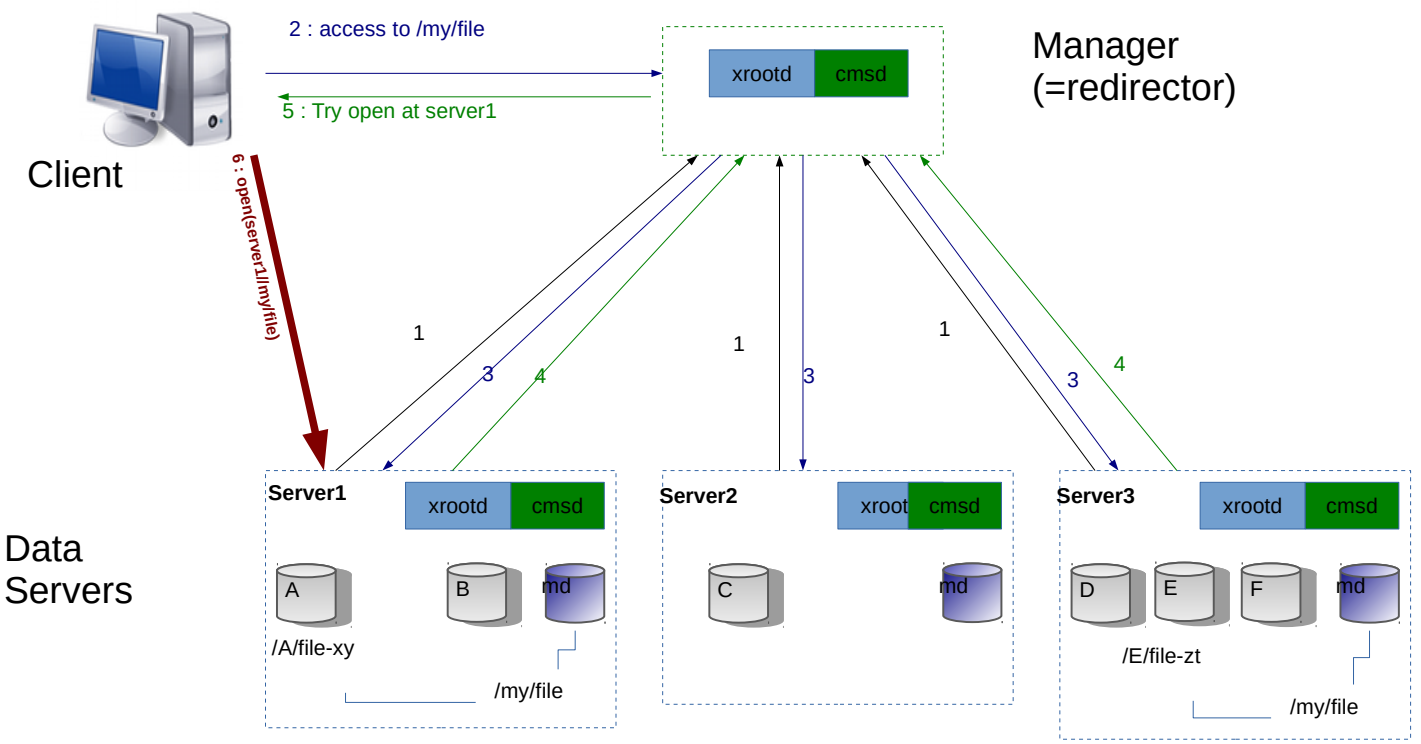
```
$ xrdcp /test-ramdisk/fichier root://${server}/${repertoire}/file${i}_thread${thread}
```

- Tests de copies de fichiers selon les scenarios 1/6/8 threads concurents d'un seul client) vers le système de stockage :
  - Copie de 300 fois le même fichier de 100Mo vers le système de stockage
  - Copie de 300 fois le même fichier de 1Go vers le système de stockage
  - Copie de 30 fois le même fichier de 10Go vers le système de stockage
  - Copie de 15 fois le même fichier de 20Go vers le système de stockage

Exemple du fonctionnement du protocole XRootD

```
$ xrdcp root://manager/my/file /tmp/fichier
```

Se traduit par :



1 → Data server to manager : Hi, I'm alive, I have $x TB free, my CPU load is 0.00...

2 → Client to manager : I want to access to /my/file = open('root://manager//my/file')

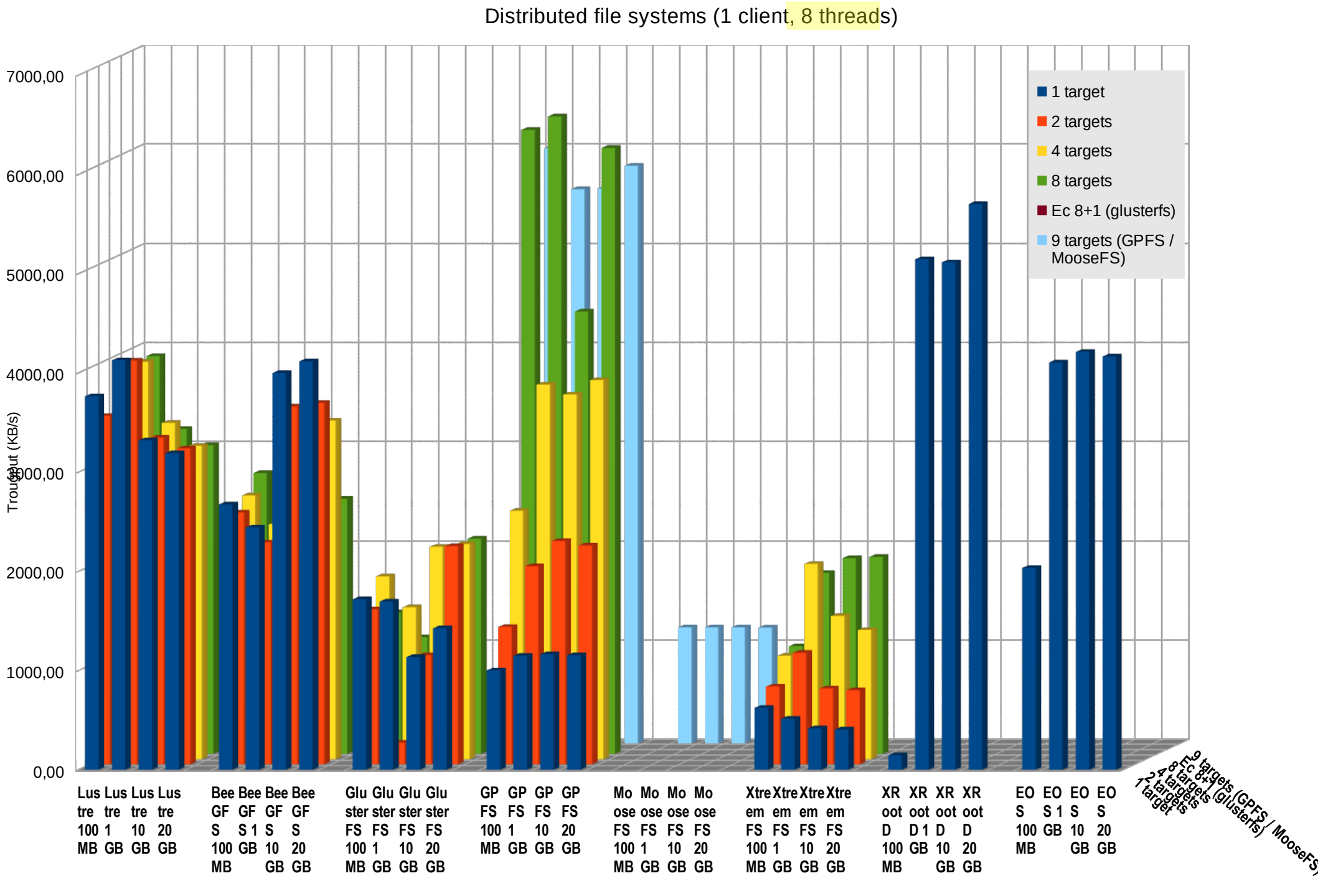
3 → Manager to data servers : who has /my/file ?

4 → Data server to manager : I have /my/file

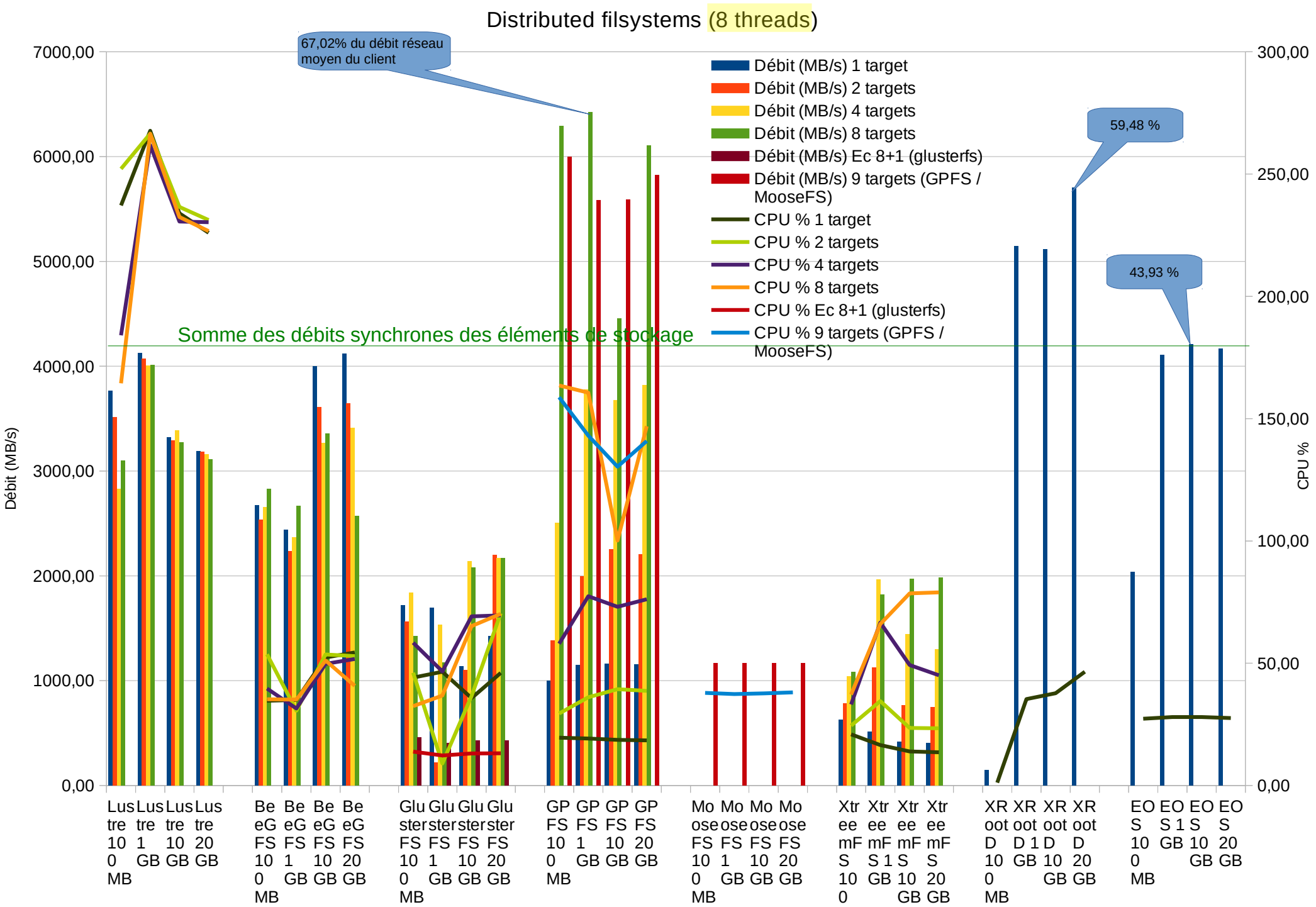
5 → Manager to client : Try open /my/file at server1

6 → Client to data server1 : open('root :server1//my/file')

Exemple de résultats obtenus sur 1 scenario (8 threads) sur un ensemble de systèmes de stockage, selon 3 parametres : FS, nb targets, taille fichiers) : mise en évidence du striping des données

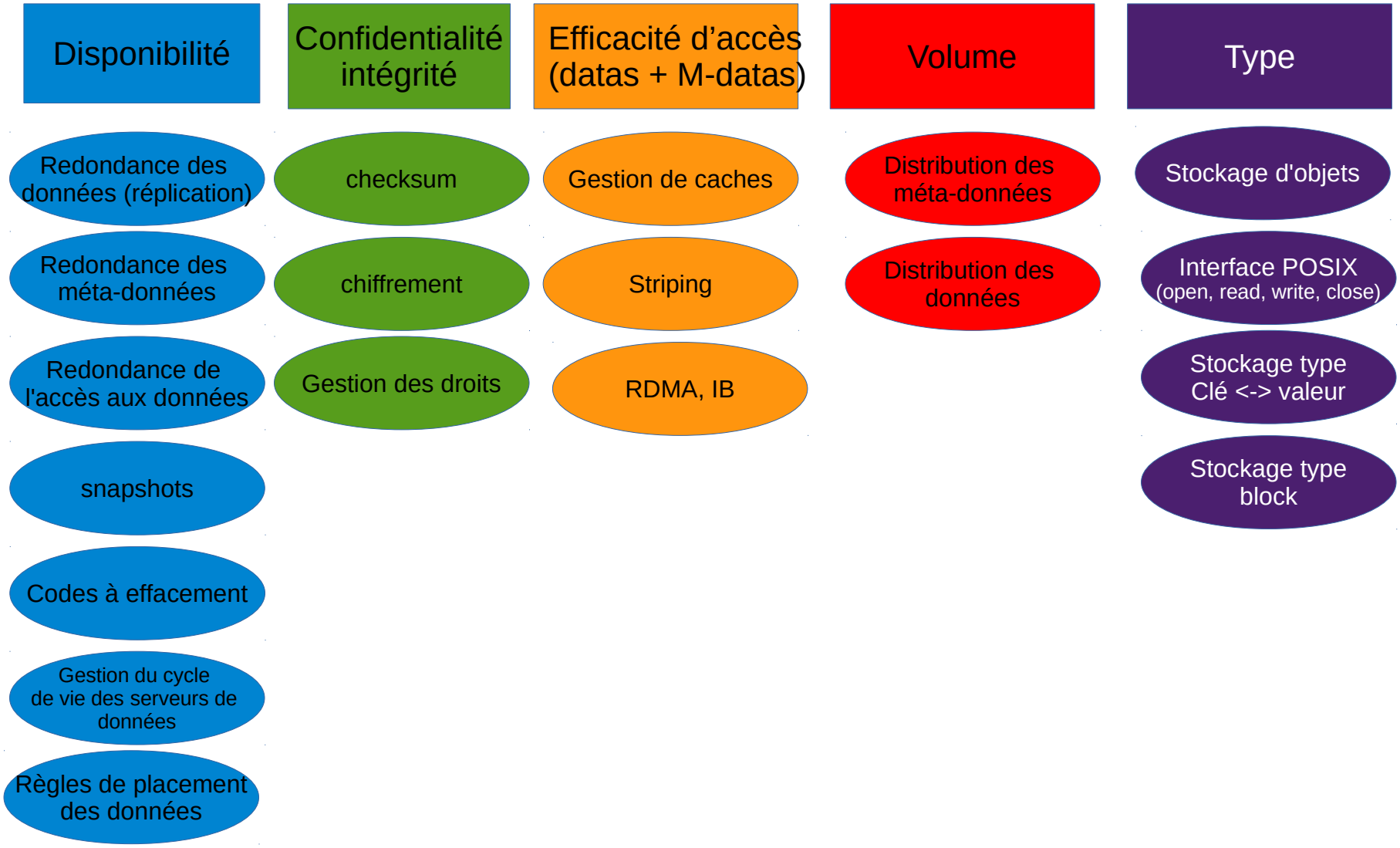


Exemple de résultats obtenus sur 1 scenario (8 threads) sur un ensemble de systèmes de stockage, selon 3 parametres : FS, nb targets, taille fichiers) : mise en évidence de la charge des clients



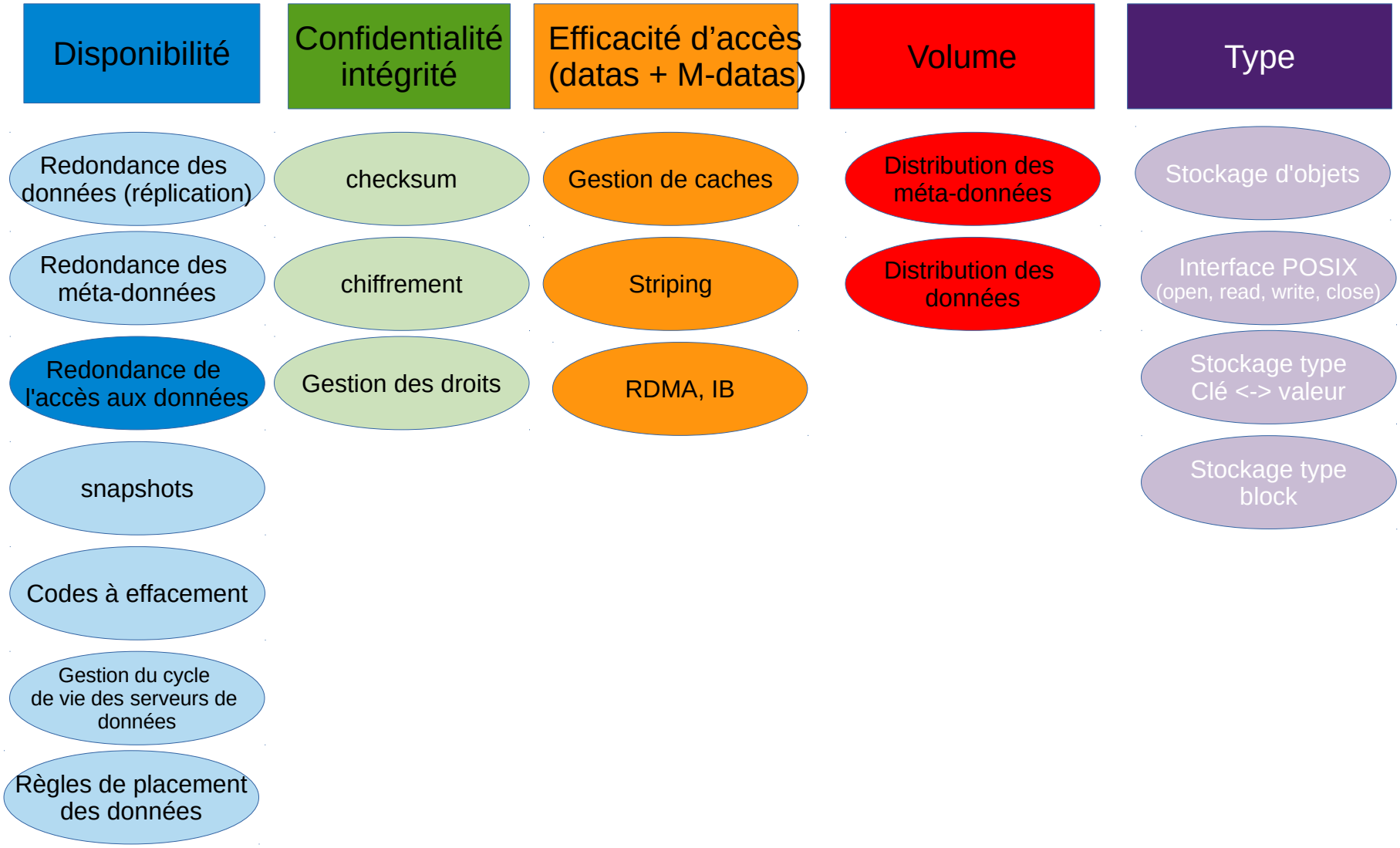
Différents types de fonctionnalités

Fonctionnalités des systèmes de fichiers/stockage distribués



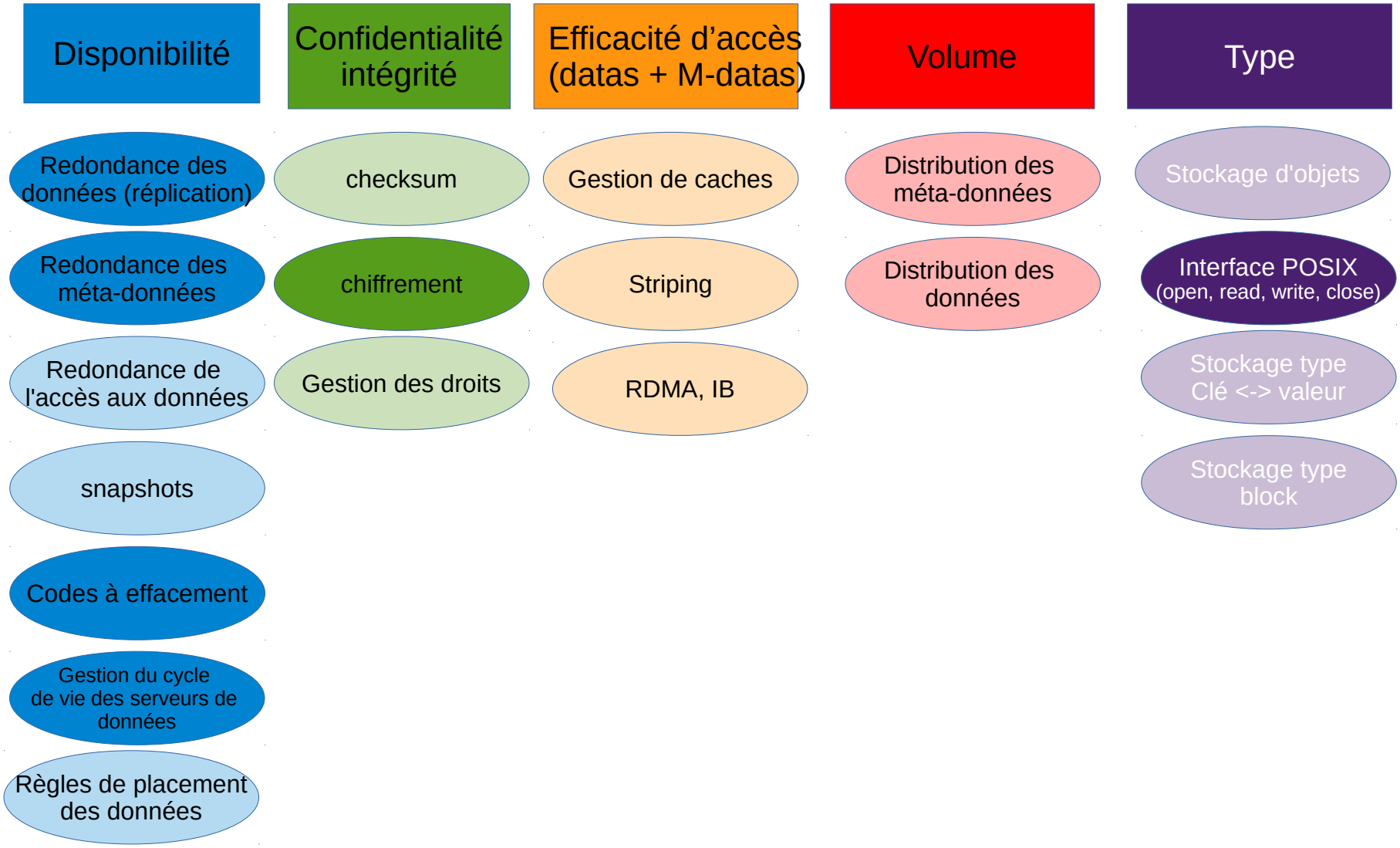
Impact des fonctionnalités sur les performances

Quelles fonctionnalités ont un impact positif sur les performances ?



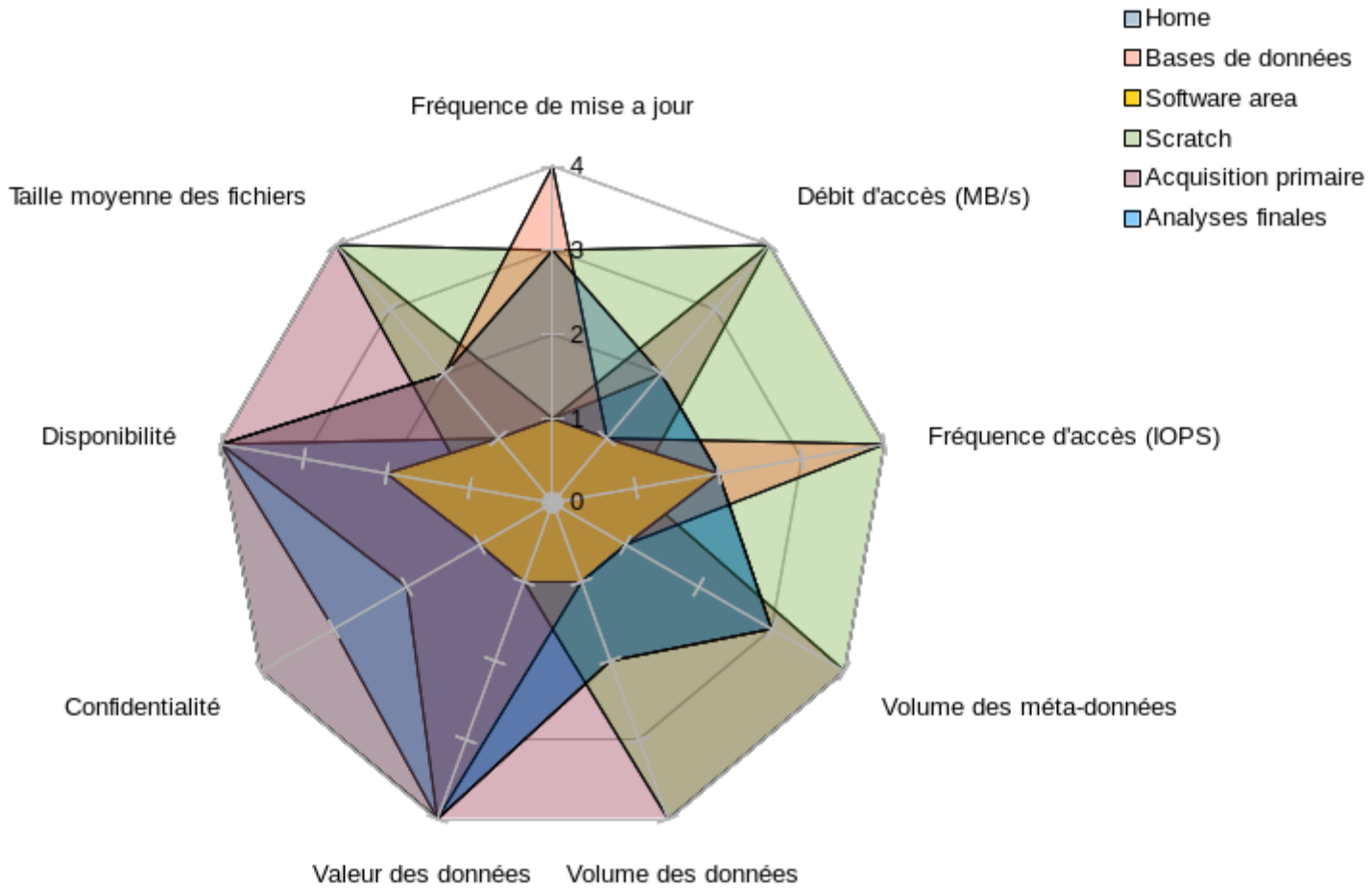
Impact des fonctionnalités sur les performances

Quelles fonctionnalités ont un impact négatif sur les performances ?



Mais existe t-il un outil idéal ?

Existe-il un système de stockage qui couvre tous ces besoins ?

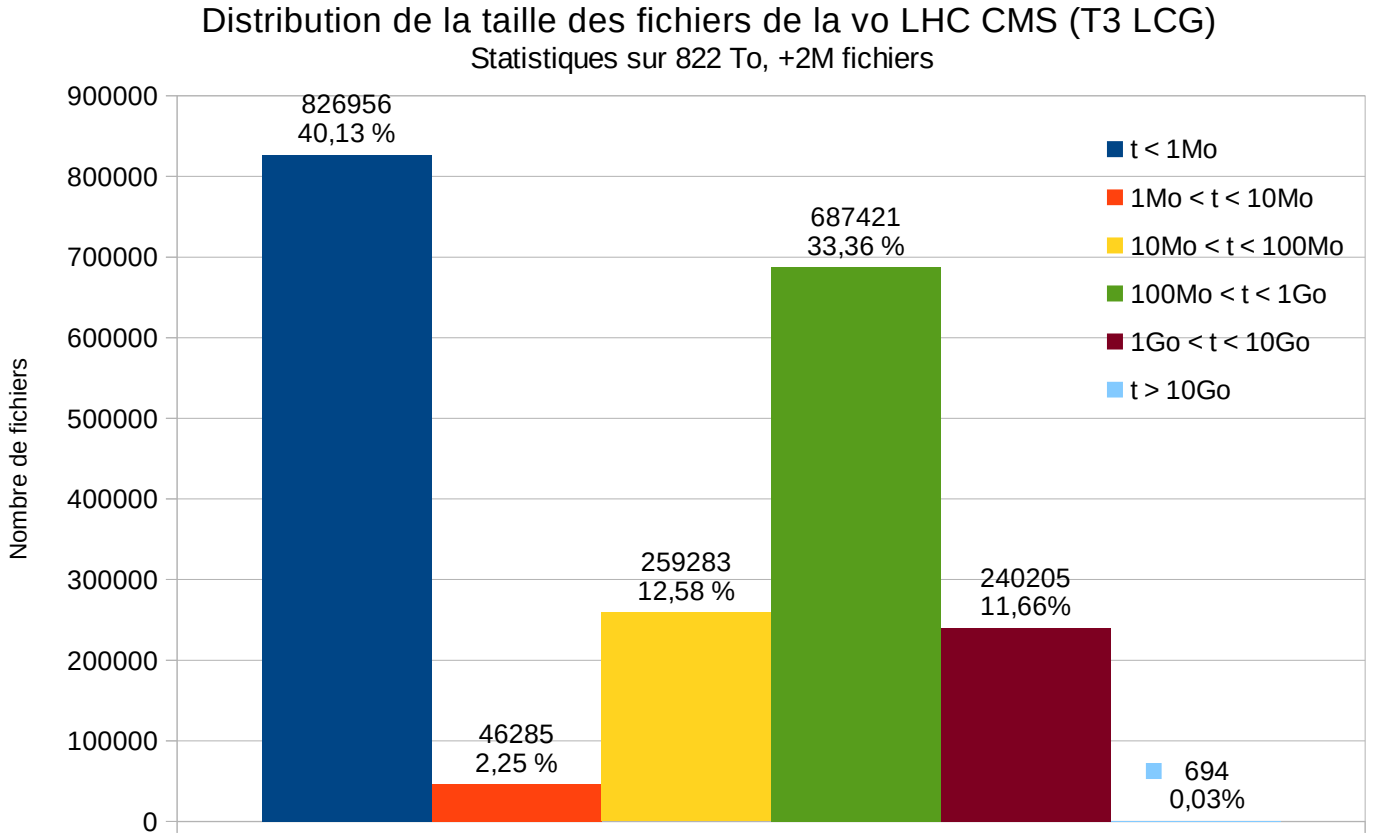


Exemple illustratif non représentatif d'un cas général

Les consignes de stockage données aux utilisateurs ↔ la réalité

Exemple de consigne[1] pour les sites CMS : « The nominal CMS file size is 5-10GB ».

Dans les faits, au bout de plusieurs années d'exploitation sur le site WLCG CMS T3_FR_IPNL, 88 % des fichiers ont une taille < 1Go (statistiques relevées en dec 2016) : **seulement 11 % de fichiers > 1Go**



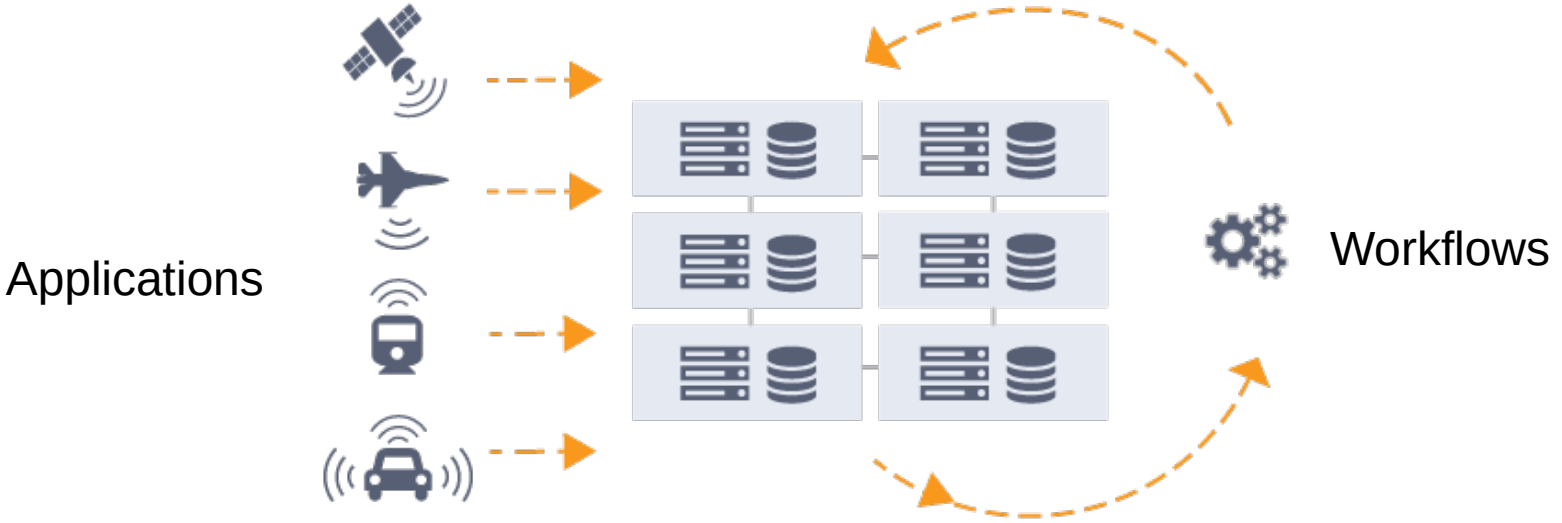
[1] : https://twiki.cern.ch/twiki/bin/view/CMSPublic/SiteOperationProcedures

Mais d'autres paradigmes existent

<u>À cause du problème "Moving Computation is Cheaper than Moving Data"</u>

On passe du monde classique « **Compute centric paradigm »** : CPU + … + CPU <=> stockage + .. + stockage

Au mode « **Data centric paradigm** » : CPU + stockage intégré <=> CPU + stockage intégré <=> ...



Clusters de type Spark/Hadoop/map-reduce/analytics

Référence des illustrations : https://www.nextplatform.com/2016/10/06/emergence-data-centric-computing/

Pour les nouveaux paradigmes

Dans le domaine du SDS (Software Defined Storage) non POSIX (Stockage d'objets, key <-> value, NoSQL, map/reduce), on observe :

- Grande entropie du fonctionnement des systèmes : stockage seulement **v.s.** computing au plus près des données (Spark, Map/Reduce, OpenIO...)
- Grande entropie du comportement des systèmes. Car c'est une combinaison des :
  - profile I/O du SDS (nb replicas (dedondance) des objets, taille de chuncks, nb de data-servers, quantité de meta-données, journal, balancing, recovering)
  - profile I/O applicative (traitements local sur les données, types d'accès...)

=> Complexité de la pile des couches logicielles

Les tests englobent de plus en plus la prise en compte de de la chaine applicative de traitement des données

Critères de choix d'un système de ~~stockage~~ traitement de données

- L'expérimentation est le meilleur moyen de déterminer le comportement d'un système
- La pile logicielle des systèmes de stockage distribués est complexe : La configuration / le tuning l'est aussi
- Un grand nombre de critères (souvent contradictoires) caractérisent les systèmes de stockage distribués :
  - Fonctionnalités,
  - Rapport coût / TB,
  - Rapport performance / coût,
  - Scalabilité de la solution,
  - Robustesse,
  - Complexité versus facilité de maintenance dans le temps,
  - Coût humain d'exploitation de la solution.
- Un grand nombre d'éléments matériels et logiciels sont en œuvre (y compris firmware)
- Ne pas se focaliser sur un test de benchmark sur un seul critère => mettre en oeuvre plusieurs scénarios
- Quantifier le service rendu :
  - En fonction du besoin
  - En fonction des fonctionnalités **réellement utilisées**

- Ne pas négliger le coût de sortie de la solution

Bibliographie

Outils de tests et de monitoring :

- Iozone : http://www.iozone.org
- nuttcp : http://nuttcp.net/nuttcp/nuttcp.html
- iperf : http://software.es.net/iperf/
- dstat : http://dag.wiee.rs/home-made/dstat/
- fio : http://git.kernel.dk/?p=fio.git;a=summary

Systèmes de stockage et d'accès :

- Xrootd : http://xrootd.org/papers/Scalla-Intro.pdf
- EOS : http://eos.web.cern.ch/
- IRODS : https://irods.org/
- GridFTP : http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/
- BeeGFS : https://www.beegfs.io
- Lustre : http://lustre.org/
- GPFS (General Parallel File System → IBM Spectrum Scale) : https://www.ibm.com/support/knowledgecenter/en/SSFKCN/gpfs_welcome.html
- Moosefs : https://moosefs.com
- GlusterFS : https://www.gluster.org
- XtreemFS : www.xtreemfs.org

Systèmes de traitement de données :

- Apache Spark : https://spark.apache.org
- Apache Hadoop : https://hadoop.apache.org