PUGNÈRE Denis
CNRS / IN2P3 / IPNL

Analyse de performances de systèmes de stockage distribués

ANF "Des données au bigdata: exploitez le stockage distribué !"
12-16 December 2016 - Complexe d'accueil de Gif-sur-Yvette

Plan de la présentation

- Quel est l'objectif ?
- Méthodologie proposée :
  - Quel est le besoin ?
  - Connaitre la capacité de son infrastructure
  - Mettre en œuvre les tests
  - Indicateurs mesurés et outils
  - Comparaison des résultats
- Conclusion

Quel est l'objectif ?

- Monter un PoC ?
- 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
- Optimiser un système déjà en production ?

Méthodologie proposée (1/5) :
Quel est le besoin ?

- Les tests sont à adapter à la finalité du système de stockage :
  - $HOME ? /scratch ?
  - Acquisition de données ? Archivage ?
  - HTC (High Thougput Computing) ? HPC (High Performance Computing) ?
  - Distribution géographique des données ?
  - Stockage d'objets ? Stockage de VM ? PRA (Plan de Reprise d'Activité) ?

- Définir :
  - Le type de données
  - La quantité des données
  - la criticité des données
  - Le type d'accès à ces données : **immaginer les scénarios**

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

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

Principe :

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

- Commencer par le plus bas niveau

- **Tests individuels de chaque client :**
  - E/S : 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...)
  - E/S : comportement en R, R/W, W, sequentiel, random, mixed
  - Réseau : receive, transmit, protocole (UDP, TCP), %drop, % retrans,
    - si nécessaire en utilisant +sieurs client 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 client : en fonction nb threads
  - charge CPU serveur : en fonction nb clients

Méthodologie proposée (3/5) :
Mettre en œuvre les tests

–

– Les scénarios pouvant se combiner :
  - X clients <-> Y serveurs
  - nb clients, nb threads / client, nb dataflows / client
  - taille des meta-données (nb fichiers, nb répertoires...)
  - taille fichiers (et distribution de la taille des fichiers)
  - différentes configurations du stockage distribué :
    – block size, chunck size,
    – nb replicas fichiers, nb replicas meta-datas,
    – quotas, snapshots,
    – types de synchronisations…

– Créer le jeu de tests : Faire varier tous les paramètres nécessaires

  => Pour faire ressortir les points forts / faibles de chaque solution

  => En fonction de son besoin (**différents profils d'E/S pouvant se présenter**)

– Tests de la chaine complète

Méthodologie proposée (3/5) :
Mettre en œuvre les tests

- Jeu de tests pour déterminer le comportement d'un système 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 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 (4/5) :
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 corellation temporelle des évennements
- Relancer les tests plusieurs fois

Méthodologie proposée (4/5) :
Outils réseau

Exemples avec nuttcp et iperf :

```
Serveur nuttcp
$ nuttcp -S
test TCP vers le serveur 10.3.3.4, taille de paquet de 8972 octets, taille de fenêtre de 4Mo, durant 60 sec
$ nuttcp -T60 -i1 -l8972 -w4m 10.3.3.4
test TCP vers le serveur 10.3.3.4, 6 streams, durant 60 sec
$ nuttcp -T60 -N6 10.3.3.4
test TCP depuis le serveur 10.3.3.4 vers le client, durant 60 sec
$ nuttcp -T60 -r -i1 10.3.3.4
test UDP vers le serveur 10.3.3.4, durant 60 sec
$ nuttcp -T60 -u -i1 10.3.3.4

Serveur iperf3 (8 threads)
$ iperf3 -s -P8
Test TCP vers le serveur 10.3.3.4, client 8 threads
$ iperf -c 10.3.3.4 -P8 -b0
Test UDP vers le serveur 10.3.3.4
$ iperf -u -c 10.3.3.4 -b0

Test RAM client vers DISK serveur
Serveur :
$ iperf3 -s -F /scratch/testiperf
Client :
$ iperf3 -c 10.3.3.x -b0
```

Méthodologie proposée (4/5) :
Outils entrées / sorties

Exemples avec dd, fio et iozone :

```
dd, 1 fichier de 10Go contenant 10000 blocs de 1Mo, I/O direct
$ 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

dd, 1 fichier de 10Go contenant 10000 blocs de 1Mo, I/O synchrone
$ dd if=/dev/zero of=test10G.dd bs=1M count=10000 oflag=sync
10485760000 bytes (10 GB) copied, 22,6967 s, 462 MB/s

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

$ fio --name=fio_test --ioengine=sync --rw=${profile} --bs=1m --direct=1 --size=${size} \
      --numjobs=${threads} --runtime=300 --time_based --ramp_time=5  --group_reporting

# iozone
 # -e  Include flush (fsync,fflush) in the timing calculations
 # -r 1m : block 1m
 # -s 1G : file size = 1G
 # -o : Writes are synch (O_SYNC)
 # -H : Use POSIX async I/O with # async operations
 # -k : Use POSIX async I/O (no bcopy) with # async operations
 #  tests (-i) : 0=write/rewrite, 1=read/re-read, 2=random-read/write, 8=random_mix,
 #  -l 1 -u 8 : 8 tests réalisés, de 1 a 8 proccess
 # -t 6 : 6 processus (uniquement)
 #  -+u : report cpu utilisation
 # +p # Percentage of mix to be reads
$ iozone -e -o -+u -r 1m -s ${size} -i 0 -i 1 -i 2 -i 8 -t ${threads} -+p ${percent} -Rb  ~/${fichier}.xls
$ iozone -e -+u -r 1m -s ${size} -i 0 -t ${threads} -Rb ~/${fichier}.xls
```

Méthodologie proposée (4/5) :
Outils entrées / sorties

Exemples avec blktrace / seekwatcher:

```
Nécessaire pour blktrace :
$ mount -t debugfs debugfs /sys/kernel/debug

Lacement du test, enregistrement des traces et rendu en une seule commande :
$ seekwatcher -t trace-scratch -o dd-scratch.png -d /dev/mapper/vg.01-scratchvol \
              -p 'dd if=/dev/zero of=/scratch/file bs=1M count=4096 oflag=sync'


Ou en plusieurs commandes :
1. Enregistrement des traces :
$ blktrace -d /dev/mapper/vg.01-scratchvol -o trace-scratch

2. Lancement du test :
$ dd if=/dev/zero of=/scratch/file bs=1M count=4096 oflag=sync

3. Rendu à partir des traces :
$ seekwatcher -t trace-scratch -o dd-scratch.png
```

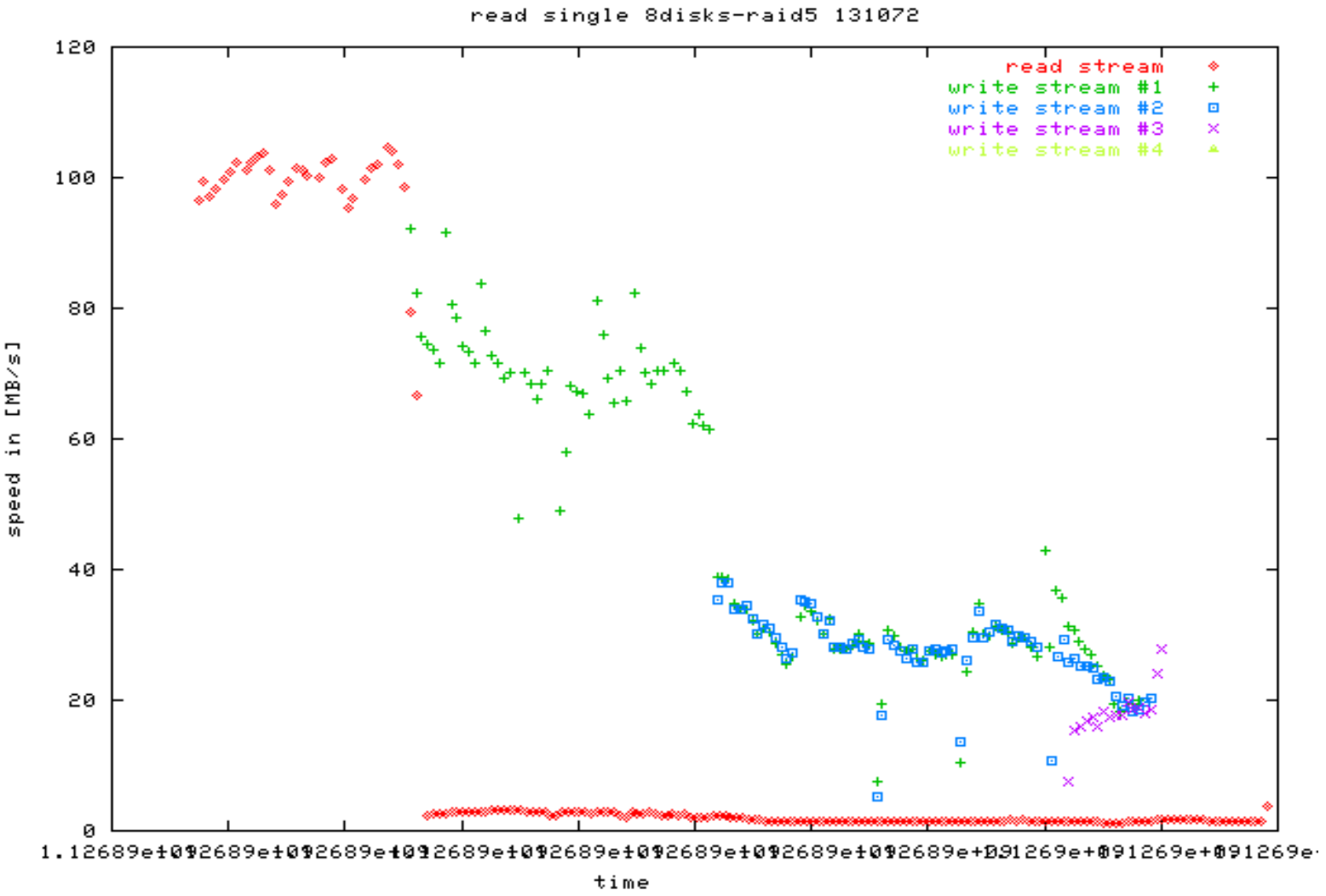
Méthodologie proposée (4/5) :
Outils systèmes

Exemples avec dstat :

```
# dstat options
 # -a = -cdngy
 # -c : enable cpu stats (system, user, idle, wait, hardware interrupt, software interrupt)
 # -d : enable disk stats (read, write)
 # -g : enable page stats (page in, page out)
 # -n : enable network stats (receive, send)
 # -N x,y : x and y network interfaces stats
 # -i : enable interrupt stats
 # -p : enable process stats (runnable, uninterruptible, new)
 # -r : enable I/O request stats (read, write requests)
 # -y : enable system stats (interrupts, context switches)
 # --aio : enable aio stats (asynchronous I/O)
 # --fs  : enable filesystem stats (open files, inodes)
 # --net-packets : show the number of packets received and transmitted

$ dstat -N p1p1,p2p1,em1,em2,bond0 -a -i -r -p --aio --fs --net-packets --output $fichier
```

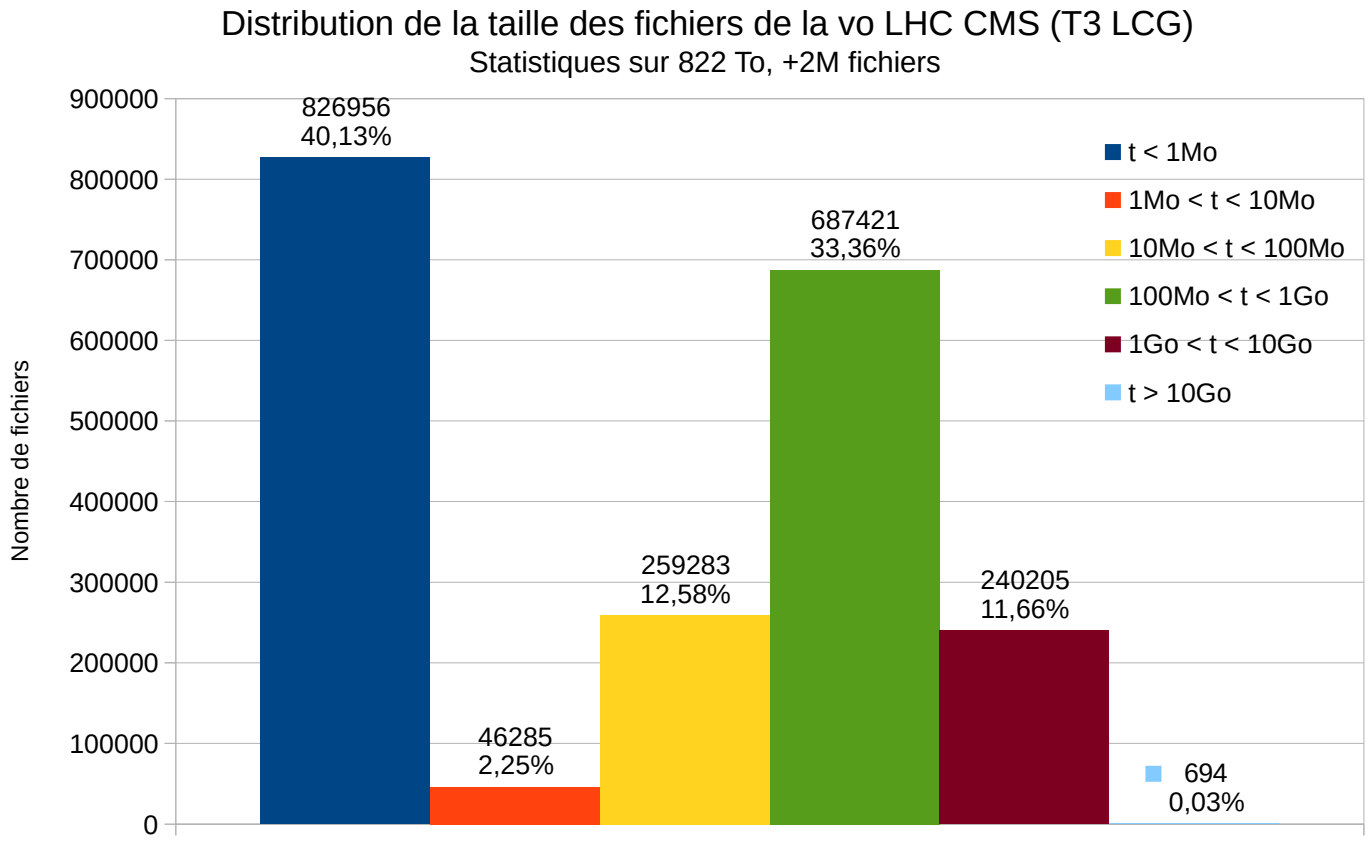
Exemples de profils d'E/S



CDR, reconstruction and data export in the T0 system,
Bernd Panzer-Steindel, CERN/IT, 2005

La vraie vie

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



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

Conclusion

- 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.
- Un grand nombre d'éléments matériels et logiciels sont en œuvre (y compris firmware)
- Ne pas se focaliser sur un test de benckmark sur un seul critère => mettre en oeuvre plusieurs scénarios

Bibliographie

- fio : http://git.kernel.dk/?p=fio.git;a=summary
- Iozone : http://www.iozone.org
- iops : https://github.com/cxcv/iops
- nuttcp : http://nuttcp.net/nuttcp/nuttcp.html
- iperf : http://software.es.net/iperf/
- dstat : http://dag.wiee.rs/home-made/dstat/
- blktrace, blkparse : http://www.cse.unsw.edu.au/~aaronc/iosched/doc/blktrace.html
- seekwatcher : https://oss.oracle.com/~mason/seekwatcher