# ZFS : la prochaine génération de système de fichiers

Jacques Foury

Institut de Mathématiques de Bordeaux

Journées *Mathrice* octobre 2007

# Plan

# Plan

# Constats

- La taille des systèmes de fichiers enfle, d'année en année.
- les résultats de calculs prennent de plus en plus de place (2D-3D), les fichiers multimedia aussi
- les systèmes de fichiers classiques ne supportent pas facilement des tailles > TeraOctet
- Exemple : un fsck (ext3) sur 1 To dure très longtemps (> 4 heures)

# Constats

- La taille des systèmes de fichiers enfle, d'année en année.
- les résultats de calculs prennent de plus en plus de place (2D-3D), les fichiers multimedia aussi
- les systèmes de fichiers classiques ne supportent pas facilement des tailles > TeraOctet
- Exemple : un fsck (ext3) sur 1 To dure très longtemps (> 4 heures)

# Constats

## Que recherche-t'on ?

- **NE RIEN PERDRE**
- Retrouver vite les fichiers perdus (effacement, problème matériel)
- Retrouver de vieux fichiers
- Gérer l'inflation : à l'IMB on estime les besoins pour 2010 à 10 To

# Constats

## Que recherche-t'on ?

- **NE RIEN PERDRE**
- Retrouver vite les fichiers perdus (effacement, problème matériel)
- Retrouver de vieux fichiers
- Gérer l'inflation : à l'IMB on estime les besoins pour 2010 à 10 To

# Constats

## Que recherche-t'on ?

- **NE RIEN PERDRE**
- Retrouver vite les fichiers perdus (effacement, problème matériel)
- Retrouver de vieux fichiers
- Gérer l'inflation : à l'IMB on estime les besoins pour 2010 à 10 To

# Ext3 ?

- multiplier les baies de disques
- faire de la redondance
- multiplier les systèmes de fichiers
- ne règle pas le problème des fsck
- problèmes de saturation sur iSCSI

# Ext3 ?

- multiplier les baies de disques
- faire de la redondance
- multiplier les systèmes de fichiers
- ne règle pas le problème des fsck
- problèmes de saturation sur iSCSI

# Solutions

## Baies NetApp

- bien adaptées
- très robustes
- boîte noire
- très chères au TeraOctet

# Solutions

## Baies NetApp

- bien adaptées
- très robustes
- boîte noire
- très chères au TeraOctet

| | WAFL | ZFS |
|---|---|---|
| **Devices** | | |
| Raid (0,1,0+1,4,5,6) | Limited to raid-4 | Yes, does not support raid-4 |
| USB storage device support | No | Yes |
| Include Raw devices | Limited | Yes |
| Prevents silent data corruption | No | Yes |
| | | |
| **On Disk(s) Features** | | |
| Copy on Write design | Yes | Yes |
| Block size | Fixed 4KB | Real time variable 512 byte to 128KB |
| File system size | 10's of gigabytes maybe terabytes | Billions of gigabytes |
| Built in Compression | No | Yes |
| Built in Encryption | No | Being developed |
| End to end checksum | No | Yes |
| Quotas | Yes | Yes |
| Volume support | Yes | Yes |
| NFS | Yes | Yes |
| NFSv4 ACL support | Yes | Yes |
| | | |
| **Snapshots** | | |
| Unlimited Snapshots | Limited to 255 | Yes |
| Read/Write Copies (clones) | No | Yes |
| Snapshots accessible over NFS | Yes | Yes |
| | | |
| **Administrative Details** | | |
| Administrative GUI interface | Yes | Yes |
| Command Line interface | Yes | Yes |
| | | |
| Pricing | Not free | Free |
| License Type | Commercial | CDDL |
| Source code available | No | Yes |

# ZFS / Solaris

## système de fichiers robuste

- correction automatique des corruptions de données
- capacité gigantesque (toutes les données de la terre seraient gérables par 1 FS)
- administration simple
- performant
- robuste, sécurisé...

# Plan

# Problèmes actuels

- Pas de correction de la corruption des données
- gestion peu souple : redimensionnements, partitionnements, volumes...
- limitations : tailles et nombre de fichiers, snapshots...
- souvent lents (accès, débit, formattage, fsck...)

# Les buts de la conception de ZFS

- gestion des données simple, puissante, sûre, rapide
- oubliez tout ce que vous savez, depuis 20 ans, on recommence à 0
- gestion du stockage = on oublie les volumes
- intégrité des données, souvent considérée comme coûteuse
- opérations transactionnelles :
  - données toujours consistantes
  - enlève les contraintes sur l'ordre des E/S
  - énorme gain en performance !

# Principes de fonctionnement

## Grands principes

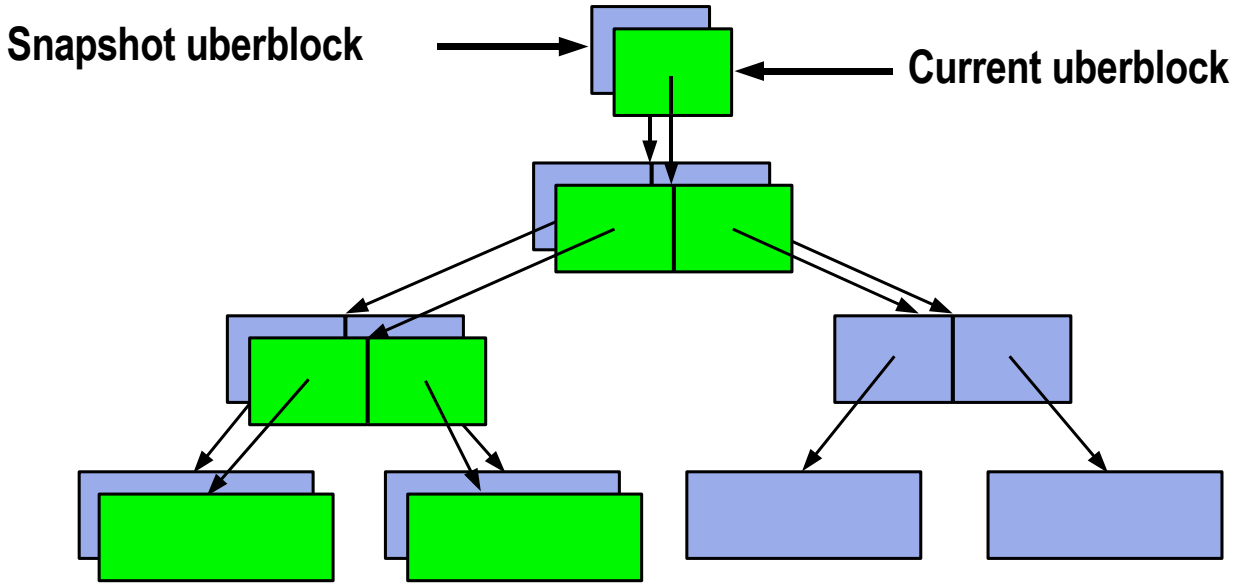
- groupes de disques mis en commun (pools)
- gestion identique à la RAM : pas de partition, redimensionnables à volonté, toute la bande passante disponible, l'espace dans le groupe est partagé par les ZFS
- pile d'E/S :
  - - transactions basées sur des objets,
  - "commit" d'un groupe de transactions,
  - toujours consistant sur le disque : pas de journal
  - E/S physiques quand on veut,
  - pas de resynchro si panne de courant,

# Principe de fonctionnement

- copy-on-write : cf transparent suivant
- transactionnel
- tout est vérifié (checksum)
- snapshots : utilise le COW : marque les blocs qui devraient être libérés : pas de ressources.
- miroirs : autocorrectif

# Bonus: Constant-Time Snapshots

- **At end of TX group, don't free COWed blocks**
  - **Actually cheaper to take a snapshot than not!**



# Plan

# RAID 4 et 5

## Faiblesse du RAID 4/5

- Rappel : RAID4 = 1 disque de parité, RAID 5 = parité répartie
- Quand on écrit, le principe est
  - de lire les données + la parité (2 lectures),
  - de calculer la nouvelle parité,
  - d'écrire la nouvelle donnée et sa parité.
- Si on a un problème sur le disque qui reçoit la parité : on corrompt la donnée !

# RAID-Z

## RAID-Z traite le problème différemment

- pas de disque dédié à la parité (vs RAID4) (6 disques minimum conseillés)
- répartition dynamique : chaque bloc *logique* se répartit indépendamment : exemple, 3 secteurs logiques = 3 blocs + 1 bloc de parité
- on ne lit pas la donnée existante, on écrit seulement une nouvelle donnée, puis on déplace le pointeur
- détecte et corrige la corruption de données
- matériel : contrôleur basique + groupe de disques de base, pas de carte spéciale, pas de NVRAM...
- A venir : RAIDZ2 (double parité)

# Scrubbing

## Vérification intégrale des données

- parcourt l'ensemble du pool pour vérifier l'intégrité des données
- en tâche de fond, pénalise peu les accès
- corrige toutes les incohérences trouvées
- remplacement de disque en panne : en live, reconstruction avec vérification transparente.
- Utilise les checksum pour vérifier la validité.

# Plan

# Virtuellement infinie

- ZettaByte = 1 million de TeraBytes, ZFS = 256 000 millions de millions de ZB...
- pas de limite de fichiers, répertoires, inodes...
- accès aux données en parallèle, à temps constant...
- rajout de disques : extension automatique et dynamique de l'espace

# Conception futée

- Copy-On-Write
- répartition dynamique sur les disques (striping)
- tailles de blocs variables, choisies en fonction de la donnée écrite
- E/S canalisées et contrôlées
- présélection intelligente des données (prefetch)

# Actions simples

- pas de volumes : tout l'espace est partagé, pas d'espace perdu (40Ko utilisés sur un FS de 18To vide)
- on met des disques de toutes sources (physiques, NAS, SAN, baies de disques)
- FS hiérarchique, avec possibilité d'héritage des propriétés
  - par arbre : snapshots, compression, backup, privilèges, partage NFS
  - usage disque : df (rapide) plutôt que du (lent)
- tout se fait en ligne...
- cloner un FS : snapshot, puis clone du snapshot offline... instantané

# Actions simples (suite)

sauvegarde incrémentale : si rapide qu'on peut la faire 1 fois par minute pour faire une réplication distante

```
zfs backup -i fs1@14:33 fs1@14:34 \
 | ssh host  zfs restore -d fs2
```

on peut déplacer physiquement les disques d'un pool et tout retrouver dans une autre machine !

```
machineA# zpool export tank
```

Puis

```
machineB# zpool import tank
```

**Testé des millions de fois en situation brutale : pas de perte de données**

# En résumé

## ZFS, c'est...

- **simple** - fait ce qu'on demande,
- **puissant** - pools, snapshots, clones, compression, scrubbing, RAID-Z,
- **sûr** - vérifie et corrige les erreurs de données,
- **rapide** - répartition dynamique, intelligent prefetch, E/S contrôlées
- **opensource** (opensolaris)
- **gratuit !**

# Plan

## OpenSolaris : AVS

(voir la vidéo de la démo)

# Plan

# Références

- http ://www.opensolaris.org/os/community/zfs
- WAFL vs ZFS
- et la discussion qui a suivi cette publication

**Questions ?**
**Voulez-vous voir un exemple ?** (si on a le temps...)