SÉCURITÉ, INTÉGRITÉ, BACKUPS

Marc SCHAEFER

22 mai 2004

Programme

- Dangers
- Politiques
- Intégrité
- Haute disponibilité / redondance
- Détection d'intrusion
- Surveillance
- Sauvegarde
- Reprise après désastre
- Conclusion générale
- Questions

Motivations

- la sécurité est souvent négligée
- au minimum mal connue
- faire confiance à des produits est dangereux
- des outils éprouvés existent
- l'administrateur système est le centre de compétence
- *long-terme*
- un conseiller externe peut cependant vous aider

Administrateur système Un conseiller externe ne peut jamais suffire : l'administrateur système est le seul qui connaît les besoins de l'entreprise, les risques, les problèmes en cas de compromission, les applications installées, les usages des utilisateurs, etc.

La sécurité est un processus

– qui ne s'arrête jamais
– doit s'adapter aux nouvelles situations (nouveaux besoins, nouveaux logiciels, nouveau personnel, nouvelles menaces, etc)

Buts

– exposer les problèmes
– montrer une vue d'ensemble des concepts
– citer quelques logiciels
– donner quelques conseils pratiques
– insister sur l'aspect du *suivi*

Des domaines vastes

Sécurité, intégrité, backup :

... des domaines vastes et souvent négligés.

... nécessitent une bonne vue d'ensemble

... mais aussi des connaissances pratiques précises.

KISS

Domaine vaste et complexe La sécurité informatique est un domaine vaste, presque sans limite : on peut toujours repousser plus loin les définitions d'un système informatique sûr. Souvent, pour pouvoir assurer la sécurité d'un système il faut s'en éloigner, pour n'en voir que l'ensemble plutôt que les constituants. Cependant, sans une connaissance précise des interactions entre les constituants et du fonctionnement interne de ceux-ci, la sécurisation ne peut aboutir.

KISS Par contre, une constante est qu'un système simple est souvent le meilleur : les systèmes compliqués, en particulier ceux qui dépendent d'une multitude d'autres systèmes, sont souvent difficiles à diagnostiquer et à corriger de façon correcte : `Keep It Simple, Stupide`.

$Id: sib.tex,v 1.6 2004/06/22 12:38:06 schaefer Exp $

Quelques cours

– Cours ESNIG sécurité, partie sécurité interne

http://cvs.alphanet.ch/cgi-bin/cvsweb/~checkout~schaefer/public/cours/ESNIG/securite/cours/interne/

– cours ESNIG SIB (410)

http://esnig5.cpln.ch/~esnig/cgi-bin/db.pl?detailed_cours_query=410

Vos questions !

N'hésitez pas à poser vos questions ou donner vos remarques

Dangers

– panne franche
– corruption silencieuse de données
– brèche de confidentialité
– prise de contrôle
– mais également : fonctionnement journalier

évaluer le niveau de sécurité

Quelques exemples de menaces

Menace	**impact**
Serveurs en panne franche	disponibilité
Corruption silencieuse de données	intégrité
Erreur de manipulation	intégrité, disponibilité, confidentialité
Feu	disponibilité
Code malicieux (trojan)	disponibilité, confidentialité, intégrité
Sabotage	disponibilité, intégrité
Surveillance passive	confidentialité
Logiciel mal conçu	disponibilité, confidentialité, intégrité
Format de données ou de stockage obsolète	disponibilité

Panne franche Le “meilleur” type de panne est celui qui peut être immédiatement détecté : dans le pire des cas il suffit de réparer le matériel, restaurer les dernières sauvegardes du système, voire de rejouer les *transaction logs*.

Corruption silencieuse de données Dans le cas de corruption silencieuse de données, il faut tout d’abord détecter le problème : p.ex. par *regression testing* en exploitant un environnement de test dans le même système informatique. Ensuite, il faut pouvoir revenir assez tôt en arrière pour réparer les données endommagées : cela est bien sûr plus facile dans le cas de fichiers indépendants que dans celui d’une base de données.

Brèche de confidentialité Si des données confidentielles sont communiquées à des tiers, on ne peut en général pas réparer le dommage. On doit donc agir de façon préventive.

Prise de contrôle Si des pirates informatiques (*crackers*) prennent le contrôle du système (p.ex. en injectant leur logiciel dans le système, pas forcément par connexion

interactive), ils peuvent rendre celui-ci inopérant (premier cas), altérer silencieusement des données (en annulant vos contre-mesures, effaçant vos logs, voire vos backups s’ils sont on-line), consulter vos données confidentielles (mêmes lorsqu’elles sont encryptées si le décryptage est aussi effectué parfois sur cette machine), voire attaquer d’autres systèmes depuis le vôtre. Ces pirates peuvent prendre le contrôle de votre système en raison de l’installation d’un logiciel vérolé (cela inclut votre système d’exploitation et le BIOS de votre ordinateur ou de ses cartes d’extension !).

C’est pour cette raison que l’approche anti-virus est vouée à l’échec.

Besoins de sécurité Bien sûr, suivant que vous êtes une association locale ou une multinationale active dans l’aéronautique, les conséquences ne seront pas les mêmes, et la protection nécessaire (niveau de sécurité) sera différent.

Malgré cela, beaucoup d’attaques sont aujourd’hui automatiques, aveugles et à large échelle. De plus, beaucoup de problèmes sont plutôt dus à des problèmes de gestion, sauvegarde, plutôt que d’attaques externes. Tout le monde a avantage à s’en prémunir, d’autant que les méthodes de base sont simples.

Autres références

`http://www.backupcentral.com/` Informations sur les techniques et matériels de sauvegarde.

`http://berdmann.dyndns.org/zwicky/testdump.doc.html` Torture-testing of UNIX backup utilities

`http://www.linux-gull.ch/cours/securite.ps.gz` Conseils de base pour l’installation et après.

`http://www.mikerubel.org/computers/rsync_snapshots/` Sauvegardes multiples sur disque avec rsync

`http://oss.sgi.com/projects/failsafe/` Système HA logiciel pour serveur NFS

`http://www.mikerubel.org/computers/rsync_snapshots/` Sauvegardes multiples sur disque avec rsync

Conclusion générale

- il existe de nombreuses solutions, parfois complexes, qui permettent d'améliorer dans une certaine mesure la sécurité d'un système informatique
- de manière à déterminer l'intérêt de ces techniques il faut évaluer leur complexité en fonction des besoins et les tester ! ne pas sous-estimer le test !
- j'espère que cette présentation vous aura donné quelques idées sur des améliorations que vous pourriez apporter à votre système informatique

Plus d'information

`1-56592-148-8` Practical UNIX & Internet security, Simson GARFINKEL and Gene SPAFFORD, O'Reilly, 2e édition
`1-56592-642-0` UNIX Backup & Recovery
Securing Debian Manual
Linux Magazine HS-18 Hors série Haute disponibilité

Solutions

- responsabilité et sensibilisation
- politique de sécurité
- méthodes de conception et acceptation
- redondance
- sauvegardes
- cryptographie (authentification, chiffrement, etc)
- anti-virus, pare-feux, filtrage
- détection d'intrusion
- formats

assurer : **disponibilité**, **confidentialité**, **intégrité**.

Responsable, politiques et procédures Principes de base :
- désignation de *responsable(s)*
- définition de politiques et procédures

Il est très important de nommer une personne responsable de la sécurité d'une entreprise : dans le cas de grands réseaux plusieurs personnes peuvent partager cette tâche et se spécialiser (p.ex. dans le réseau interne, dans les dommages aux personnes, etc).

De même, un document tenu à jour devrait préciser les règles de fonctionnement de l'entreprise et détailler également les procédures en cas de problème voire de désastre. Ce document devrait être connu et approuvé dans ses grandes lignes par la direction de l'entreprise. Un tel document pourrait décrire une procédure de basculement vers un site de secours, ou simplement détailler la fréquence des sauvegardes et de leur vérification.

Dans certains cas – et cela se fait de plus en plus en milieu académique – les règles de bases seront acceptées par les utilisateurs finaux, qui seront sensibilisés à la problématique de la sécurité. En entreprise, les rapports de travail sont différents mais une sensibilisation à travers l'acceptation d'une charte ou d'un standard minimal de sécurité, semble également recommandé.

Disaster recovery Une procédure manuelle ad-hoc est décrite dans *Unix Backup & Recovery*.

Sur un système à demi-installé, si l'on utilise tar (notamment pour le *disaster-recovery*), l'option `--numeric-owner`, en plus de l'option `-p`, est nécessaire pour restaurer les bons utilisateurs et groupes. La commande `chroot` peut également être utile. L'option `--incremental` est quant à elle nécessaire si l'on désire l'effet d'effacement pour restaurer un incrémental exactement.

Taux de fiabilité

disponibilité	arrêt du service sur 1 an
99.9%	environ 9 heures
99.99%	environ 1 heure
99.999%	environ 5 minutes
99.9999%	environ 30 secondes

MTTF Mean Time To Failure

MTBF Mean Time Between Failure

MTTR Mean Time To Repair

Conclusion

- complexe
- nécessaire de tester et de valider
- adapter sur le long terme
- démarche très utile
- quelques logiciels simples existent déjà pour préparer !

Mondo Rescue

http://www.microwerks.net/~hugo/

- génération de CD(s)
- démarrables (tester !)
- on peut séparer système et données
- on peut évaluer le temps de reprise

Manuellement

On a oublié de préparer ... que faire ?

- monter une machine de secours (recommandé)
- possibilités
  - soit installer un OS de manière usuelle, puis restaurer la partie données spécialement
  - soit démarrer avec les CD d'installations, préparer celle-ci, puis restaurer toutes les sauvegardes dans le point de montage temporaire

c'est mieux de se préparer ... notamment pour mesurer la durée et la disponibilité des outils et sauvegardes.

Fiabilité La plupart des valeurs de fiabilité sont obtenus par des calculs à partir d'estimations statistiques. Par exemple, si un disque a un MTBF de 40'000 heures, si l'on arrange deux disques en miroir on obtient théoriquement un doublement de la fiabilité ... sauf si on prend en compte la fiabilité d'autres systèmes en série : notamment le contrôleur RAID, ou pire une partie mécanique comme un ventilateur ou de puissance comme une alimentation.

En règle générale, on ne fera pas de véritables calculs de MTBF mais des estimations et l'on cherchera les points de failles (*single point of failures*) que l'on tentera d'éliminer (ou on prendra des composants de qualité ...).

Eliminer les pannes Eliminer les pannes totalement n'est pas possible. La stratégie est plutôt d'en limiter l'impact (p.ex. par des sauvegardes hors site régulières en cas de désastre, par des serveurs en *fail-over* de manière à garantir de la disponibilité, ou par des mesures techniques simples comme de la mémoire ECC, des disques en RAID, etc).

Vouloir diminuer le taux de panne est très coûteux au-delà des mesures simples et ne

résiste pas forcément aux forces de la nature, aux erreurs humaines ou aux attaques délibérées.

Politiques

- mises à jour régulières
- audits du système (logs)
- mailing-list de sécurité
- seulement les services indispensables (firewall non suffisant !)
- sauvegardes régulières et correctes
- évaluation des risques et plans

Journal Dans certains cas l'entreprise ou l'employé pourrait être retenu responsable de ne pas avoir suivi le plan prévu, en particulier si l'entreprise doit rendre des comptes au plan légal à un client par exemple.

Outils

- sécurité
  - des tas d'outils UNIX
  - SANTA, `nessus` et divers outils de test, alarme, etc.
- intégrité / intrusion (IDS)
  - outils UNIX portables : `md5sum`, `gpg`, kernel
  - Logiciels plus complexes OSS : `tiger`, `cops`, `integrit`
  - log-checkers : `logcheck`
- backup
  - outils UNIX portables : `tar`, `cpio`, `dd`, `gzip`
  - logiciels de backup réseau OSS : `Amanda`, `afbackup`

pas d'outils magiques

Outils en pratique

- CD démarrables et utilisables (knoppix, distribution, etc)
- outils de base : `fdisk`, `dd`, `tar`, `mke2fs`, `e2fsck`, `debuge2fs`, `mount`, `lilo`, `gzip`, `chroot`, etc.
- autres outils / pilotes : accès cassette, réseau, etc.
- il existe des CDs pour le DR

Validation Disaster Recovery

– évaluation par les personnes concernées
– diffusion
– test

à renouveler (changements !)

Usage du plan

– suivre point à point
– distance critique : noter les points à problèmes, les points effectués différemment
– importance du journal !

Intégrité

Buts :

– assurer la détection de modifications sur des données statiques (corruption, piratage, etc)

```
for i in lib bin sbin
do
   for j in / /usr
   do
      find $j/$i -type f -print0 | xargs -0 md5sum | sort
   done
done
```


(stocké sur et vérifié depuis média en lecture seulement, p.ex. **knoppix**)

Place à la paranoïa ... Pour faire une réelle vérification d'intégrité, il faudrait débrancher les disques, les rebrancher sur des machines sûres, démarrer depuis un medium non modifiable, et vérifier les signatures ou sauvegardes (préalablement générées par cette méthode) stockées sur des media non modifiables. Il n'est pas toujours nécessaire d'aller si loin, mais après un piratage il est fortement recommandé de réinstaller le système, puis de comparer les données et configurations douteuses à des données ou signatures sûres.

Rappelons cependant que les derniers processeurs d'Intel ont la capacité, qui n'est pas déconfigurable par matériel, de reprogrammer le microcode interne aux processeurs. Aucun virus n'exploitant cette fonctionnalité par exemple pour ajouter des instructions *backdoors*, n'est connu, cependant.

Cas d'étude 2

- entreprise avec serveur *intranet*
- système *fail-over* pur
- sauvegarde une fois par jour
- sauvegarde hors site une fois par semaine
- historique de 6 mois

Mais au préalable

- rendre immutables ou en ajout seulement certains fichiers (ext2/3 `chattr`)
- changer le niveau de sécurité du système (*securelevel*)
- diminuer les privilèges de root (*capabilities*), comme p.ex. interdire `/dev/kmem` et le chargement de modules
- tourner certains services en `chroot` et pas sous root (DNS p.ex.)

Proposition Disaster Recovery

1. recensement
2. risques, probabilités et coûts
3. procédures *accessibles*
   - au jour le jour
   - en cas de sinistre (réactions)

Risques A partir des informations collectées, on peut déjà déterminer les parties critiques du système et leur associer des risques. On peut également établir leur coût (relatif ou absolu si connu), la durée de panne maximum acceptable ainsi que la perte inacceptable (ou acceptable : p.ex. 1 heure, 1 jour, etc).

Volatilité Malheureusement, tout système informatique est volatile : des services sont déplacés, un système se voit installer une base de données, etc.

Cas d'étude 1

- entreprise avec serveur *extranet*
- hébergé chez un hébergeur
- application centrale de vente
- connexion Internet rapide dans l'entreprise
- sauvegarde différentielle par Internet deux fois par jour
- pas d'historique

Support kernel pour la sécurité Certaines fonctionnalités du kernel Linux peuvent être utilisées en vue d'intégrité : p.ex. rendre certains fichiers non modifiables (immutables), ou en ajout seulement grâce aux attributs d'ext2 (`man chattr`). Mais n'oublions pas que ces protections ne sont pas absolues (supprimer le chargement de modules kernel, l'accès à la mémoire physique, aux périphériques bruts sont des prérequis). On peut aussi envoyer tous les logs systèmes sur une machine plus sécurisée, sur une imprimante, dans un buffer série, etc.

Exemple :

```
# apt-get install e2fsprogs

% cd /tmp
% touch immutable
% touch append_only
% su
Password:
# chattr +i immutable
# chattr +a append_only
```



```
# rm append_only
rm: cannot unlink 'append_only': Operation not permitted
# touch immutable
touch: creating 'immutable': Permission denied
# chmod a+w immutable
chmod: changing permissions of 'immutable': Operation not permitted
# date >> immutable
bash: immutable: Permission denied
# date >> append_only
# chattr -i immutable
# chattr -a append_only
# exit
% rm immutable append_only
```

Attention, ces attributs étendus ne sont pas sauvegardés par `tar` !

Pas de solution universelle Il n'existe pas d'outils magiques qui peuvent décider de la sécurité ou de l'intégrité d'un système : s'ils existaient, les pirates les auraient aussi et testeraient leurs attaques à l'avance de façon à éviter d'être détectés. De plus, KISS s'applique aussi : un programme complexe de détection de failles est lui-même une faille. Par exemple, il faut vérifier qu'un programme de détection de failles fonctionnant en réseau n'est pas lui-même vulnérable (p.ex. serveur HTTP interne propriétaire).

Intégrité : Tripwire

http://www.tripwire.org/

– produit commercial, version open source
– pas seulement logiciel d'intégrité
– multi-plateforme
– principe : signatures

autres solutions, parfois plus avancées, p.ex. integrit, cops, tiger, etc

Recensement

1. quelle infrastructure informatique ?
   – serveurs
   – logiciels
   – applications/services
   – sauvegardes
   – connexions
   – accessibilité
2. quels risques ?
3. quels coûts ?

complexe, volatile

Recensement La partie la plus systématique du travail est d'établir la liste des machines à sauvegarder, leurs configurations et les logiciels et services qui y sont installés ainsi que la localisation des données. Il est également important de qualifier le type des données (p.ex. données actives comme base de données)

On peut également profiter du recensement pour indiquer quels services pourraient être déplacés sur quelles machines en cas de panne.

La fréquence des sauvegardes et leur type est également à préciser par machine.

La nature des audits (journaux (logs) systèmes) est une information importante pour tracer la cause de problèmes.
Dans certains cas, la nature des connexions à Internet est à préciser (en cas de panne, p.ex. faillite ou panne technique).

L'accessibilité des systèmes doit être définie (intervention en site, mais également vol de données ou de systèmes).

Reprise après désastre

avant :

– recensement
– risques : évaluer leurs probabilités et leurs coûts
– développer des procédures pour *éviter* ou atténuer en respectant le rapport coût/bénéfice
– planifier les réactions possibles
– publier de manière appropriée et faire critiquer et approuver
– validation et mise à jour à long terme

Haute disponibilité / redondance

– garantir la disponibilité des systèmes en choisissant des composants redondants ou des configurations à haute disponibilité
– assurer également la disponibilité de données (ECC, RAID)

Reprise après désastre (2)

pendant :
– suivre les procédures
– documenter les opérations
– sauvegarder les traces / preuves
– travailler sur copies : `dd`

but : améliorer les procédures, prouver la bonne réaction

Problèmes

– disponibilité
  – panne franche
  – performance
– confidentialité
– intégrité
  – corruption de données (invisible !)
  – tronquage, crash
  – perte de données (*bad-block* p.ex.)

La disponibilité

- conception, voire redondance
  - réseau
  - bus
  - stockage
  - serveurs
- reprise après panne (et *disaster recovery*)
  - parfois liée à l'intégrité (historique)

Disponibilité et intégrité des données

outils

- mémoire ECC, parité
- méthodes de détection d'erreur (CRC), fonctions de hâchage (MD5), signatures électroniques
- systèmes de détection de modification
  - logiciels : système de *packaging*
  - logiciels/configurations/données : `cops`, `tiger`.
  - comparaison de sauvegardes
  - problématique pour les bases de données
- journalisation (fs, SGBD)

Conclusion

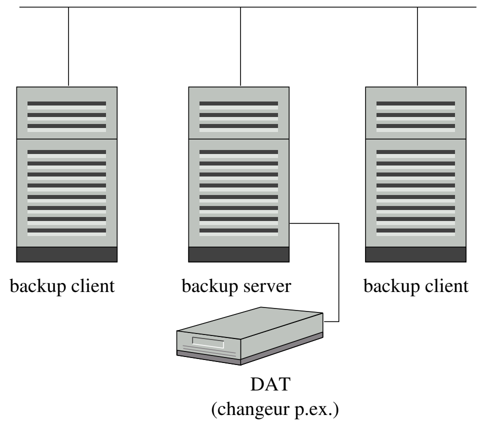
- la sauvegarde correcte nécessite des connaissances détaillées du système à sauvegarder (logiciels, licences, besoins)
- malgré tout des méthodes simples et largement disponibles permettent d'assurer une bonne sécurité à coût acceptable.
- enfin, il s'agit à nouveau d'un processus à long terme de surveillance et d'adaptation.

Sinistre

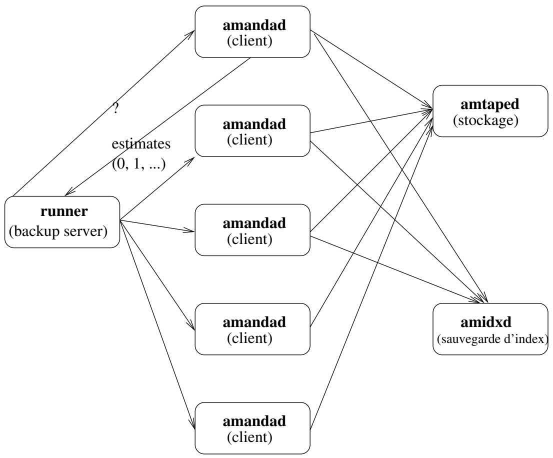
- événement naturel ou non, intentionnel ou non, qui peut mettre en danger les vies humaines, les équipements, les données ou l'entreprise
- nous allons nous concentrer sur les problèmes liés aux données IT

ce qui importe : prévenir mais aussi savoir *réagir*

Amanda : architecture



Amanda : fonctionnement interne



Intégrité (2)

– la plupart des méthodes de redondance n'offrent pas de protection.
– comment évaluer l'effet d'un problème d'intégrité ?

difficile, **complexe**, **dépendant de l'application**, **historique**.

Intégrité Assurer l'intégrité des données est un problème complexe. En assurant déjà une certaine qualité du matériel (mémoire ECC, parités et CRC sur les bus système et de stockage) on peut éviter certaines corruptions dues au matériel.

Malheureusement, dans les systèmes actuels, il est rare que toute corruption due au matériel puisse être évitée dans tous les cas. Il peut être intéressant dès lors d'exécuter des logiciels, en production, de détection de corruption.

De plus, pour certaines applications, il est difficile de détecter les modifications légitimes de corruptions. Une méthode peut être d'assurer l'intégrité du format par des vérifications automatiques, éventuellement incluses dans le logiciel concerné.

La programmation défensive et les journaux sont également utiles pour détecter et corriger les problèmes en particuliers dus au logiciel.

Journalisation Un SGBD avec contraintes d'intégrité n'assure pas forcément ces contraintes en cas de corruption incontrôlée de la base. En règle générale, toute panne franche permettra de rejouer le journal et d'atteindre un état cohérent. Mais pas une perte de données ou une corruption dans des enregistrements non touchés par la transaction. Par panne franche, on suppose qu'il n'y a pas de données écrites et que les caches sont soit vides, soit auraient dû s'écrire dans l'ordre prévu (respect de l'ordre des écritures du journal !).

Exemple de vérification Par exemple, un logiciel de gestion d'archivage utilisant une base de données ainsi que des fichiers simples pourrait régulièrement vérifier que la base de données contient tous les fichiers simples existants et aucun autre.

Evaluation de l'effet d'un problème d'intégrité Comme nous le verrons lorsque nous traiterons de sauvegardes, l'existence d'un historique de modifications (journal, snapshots, sauvegardes régulières conservées) permet d'assurer le retour en arrière ou du moins une évaluation des corruptions éventuelles.

Suivant les applications il sera cependant difficile de trouver les données modifiées ou cela pourrait être très coûteux.

Disponibilité

buts :

- empêcher la visibilité des pannes
- assurer une bonne performance également en cas de charge importante
- diminuer le *downtime* en cas de panne (également désastre)

besoins

Amanda

- client-serveur-serveur
- support changeur
- lissage automatique des sauvegardes (niveaux)
- restauration facilitée (fichiers simples et répertoires notamment)
- support `tar` ou `dump`, y compris incrémental
- rapport détaillé en cas de problème (erreurs, performance, ...)

limites

Limites Les limites d'Amanda peuvent être relativement ennuyeuses suivant l'utilisation, en particulier avec peu de serveurs et des systèmes de sauvegardes peu performants. Citons notamment :

- la sécurité
- l'impossibilité d'ajouter des données à une cassette (possible avec patches)
- l'impossibilité de découper une sauvegarde sur plusieurs cassettes
- le lissage automatique est parfois un inconvénient
- l'orientation cassettes / changeur
- quelques problèmes d'optimisation
- l'absence de support *disaster-recovery* en standard (supporté par des ajouts)

La version actuelle d'Amanda supporte cependant pas mal de possibilités et je la recommande pour toute sauvegarde sur bandes de serveurs multiples.

```
        --blocking-io \
        --partial \
        --timeout=${RSYNC_TIMEOUT:-500} \
        /data backupsrv:/rsync/data/backup.0

backupsrv# cd /rsync/data
backupsrv# rm -rf backup.7
backupsrv# for i in `seq 6 1`
           do
              mv backup.$i backup.$(($i + 1))
           done
backupsrv# cp -al backup.0 backup.1
```

Problèmes de cette solution `rsync` utilise beaucoup de mémoire s'il y a beaucoup de fichiers (p.ex. 200 MB pour 700'000 fichiers) et devient en conséquence relativement lent. De plus, les sauvegardes incrémentales telles que décrites ne peuvent pas conserver l'historique des permissions des fichiers. Les numéros d'*inodes* ne sont pas conservés. Certains programmes comme Postfix qui dépendent de ceux-ci détectent

Disponibilité On tente d'assurer ici le fonctionnement (du point de vue des utilisateurs) des serveurs. Il s'agit d'assurer la disponibilité des applications et des données *en fonction des besoins et des risques envisagés*.

la modification au prochain démarrage.

Enfin, comme cette solution est purement incrémentale, il est important de vérifier les données de temps en temps (`tar --compare`, `md5sum`, etc).

Quelques définitions (1)

fail-over le serveur est doublé. Le maître est actif. L'esclave est en attente. Les données sont dupliquées entre le maître et l'esclave. En cas de panne du maître, l'esclave prend le relais.

HA *High Availability* : haute disponibilité.

load-balancing répartition de charge, en général sans HA

RAID Ensemble de disques gérés comme un seul disque logique, arrangé selon différentes méthodes suivant que l'on désire de la performance, de la fiabilité ou les deux.

Quelques définitions (2)

NAS Network-Attached Storage : serveur de fichiers sur réseau permettant de séparer les serveurs de données des serveurs d'application. Fonction de verrouillage pour accès multiple.

SAN Storage Area Network : réseau de stockage où les serveurs de données peuvent être partagés ou/et dédiés et peuvent communiquer entre eux. Fonction de verrouillage pour accès multiple.

Quelques définitions (3)

synchronisation de données Assurer que deux serveurs possèdent en tous temps les mêmes données. La réalisation nécessite un protocole spécifique. En règle générale, on adapte de manière à assurer une bonne performance ou à diminuer les coûts.

Sauvegarde en-ligne rsync

- pratique pour accéder rapidement aux données
- avantageux (disque bon marché)
- fonctionne par réseau (p.ex. via SSH)
- simple à mettre en place, y compris incrémental
- contrôle des données : `simple-data-sync`.

aussi : `rdiff-backup`.

Principes de base On installe sur la même machine un deuxième disque, par exemple amovible, et régulièrement on lance la commande `rsync` qui va synchroniser les données actives avec la sauvegarde. Cela peut également être fait via réseau sur une machine distante. Comme seules les différences sont transmises (y compris à l'intérieur d'un gros fichier p.ex.), la performance sera bonne.

Historique Comme `rsync` ne travaille jamais sur les fichiers existants mais crée des copies de travaille puis remplace l'existant, il est possible d'utiliser les liens durs. Lorsqu'un fichier est changé par `rsync`, le lien est cassé. On peut voir cela comme un *copy-on-write* des données (mais pas des permissions !).

```
rsync -aHx \
      --progress --verbose \
      -B 8192 \
      -e 'ssh -e none' \
      --delete \
      -z \
      --stats \
```

Comment vérifier ?

- media lisible, aussi dans d'autres conditions/lecteurs
- taux d'erreurs corrigées
- intégrité des données (md5sum, comparaisons)
- tous les fichiers nécessaires
- tous les programmes / licences / pilotes / matériels nécessaires
- problème des fichiers très actifs (SGBD)
- durée

documenter et tester la restauration

Sauvegarde : média et méthode

- disque (amovible) : `rsync`, `rdiff-backup`, `tar` incrémental
- réseau : `rsync`, `tar` incrémental (sur SSH), éventuellement chiffré
- CD/DVD : `cdbackup` (`tar` incrémental, `cdrecord`, `dvdrecord`)
- cassette/changeur : `tar`, Amanda, `afbackup`.
- attention à la compression

critères : prix, rapidité (création, lecture 1 fichier, lecture totale), durée du média, lecteur, **format**, historique, inventaire.

Disponibilité : exemple 1

modèle thin client

- panne du serveur implique panne de tous les terminaux de l'entreprise
- toutes les données non sauvegardées sont perdues
- il est impossible d'empêcher la visibilité des pannes
- il est probablement assez facile de doubler le serveur (*fail-over* visible) et de synchroniser régulièrement ou de façon permanente les données
- un *load-balancing* est également possible si la synchronisation des données est permanente

Load-balancing Le load-balancing peut être fait sur les accès aux données et même sur les programmes lancés. Le serveur étant alors un serveur d'application particulier. On peut imaginer deux granularités : chaque *thin client* se connecte à un serveur de session selon la charge de ceux-ci et toutes les applications sont lancées depuis ce serveur. Une granularité meilleure pouvant amener à une meilleure performance – mais une résistance au pannes plus faible peut être obtenue en lançant chaque application depuis un serveur d'application différent (possible avec X11).

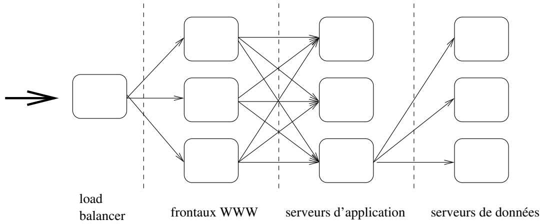
Disponibilité : exemple 2

modèle serveur de fichiers

- panne du serveur ne concerne que le stockage de données
- il est possible d'empêcher la visibilité des pannes (NFS)
- il est probablement assez facile de doubler le serveur (*fail-over* visible) et de synchroniser régulièrement ou de façon permanente les données
- un *load-balancing* est également possible si la synchronisation des données est permanente mais n'apporte pas forcément d'amélioration de performance

Disponibilité : exemple 3

modèle Web Services 3-tiers



populaire et non limité à l'HTTP

Hot backup Le principe est de passer toutes les *tablespaces* en mode *backup*, ce qui garantit l'absence de modification directe, puis de copier ces fichiers de données à un autre endroit. Ensuite on sort du mode backup, Oracle peut appliquer les modifications. On change de *redolog* ce qui en mode *archive* signifie que les précédents redolog sont toujours disponibles. On vérifie qu'on a tous les redologs nécessaires et on sauvegarde. On peut également sauver le redolog actif, même s'il est postérieur à la sauvegarde et sera partiel.

Journal Cette méthode a l'avantage de ne pas nécessiter trop de redologs et donc de ne pas risquer une restauration trop longue. Une autre méthode est de ne sauver que la base initiale puis les redologs, mais cela peut causer des durées de restauration très longues.

Logical backups Cette terminologie Oracle correspond à un dump binaire de la base, avec cohérence table par table. Il n'y a pas de logs de transactions, d'historique et l'opération est lente. Le backup logique complet nécessite l'arrêt de la base (ou un mode sans modification).

Dans le cas d'Oracle l'utilitaire supporte également des niveaux de sauvegarde avec granularité de table.

Il faut en général que l'objet concerné soit dans un état stable au moment où le snapshot est pris :

`http://www.linux.org/docs/ldp/howto/LVM-HOWTO/snapshots_backup.html`

Par exemple, avec le système de fichiers xfs, il faut utiliser la commande `xfs_freeze` pour entrer puis sortir du mode freeze. Avec certaines bases de données il faut entrer en mode backup, voire arrêter certaines applications, etc.

Exemple : Oracle

- cold backup
- hot backup (avec mode archive)
  - passer en mode *backup*
  - copier les fichiers
  - sortir du mode *backup*
  - changer le *redolog*
  - sauvegarder les fichiers de contrôle à travers Oracle
  - copier les redologs (archivés et actifs) et vérifier que tous les logs sont présents
- hot backup (mode mirror/replication)
- logical backups

Granularité En règle générale, la granularité de fiabilité est la transaction HTTP. Entre deux transactions la panne peut être cachée aux utilisateurs. Dans une transaction, des mécanismes logiciel client peuvent également être implémentés (atomicité, idempotence) de manière à assurer une transition transparente.

Principes de base L'idée – souvent appliquées dans les applications Web (WWW) à haute disponibilité et répartition de charge – est de séparer les diverses parties d'un serveur WWW en 3 parties :

1. *les frontaux WWW*, qui se chargent de la prise en charge de requêtes HTTP et de leur traitement de base (erreurs, fichiers statiques, éventuellement cache de requêtes). En général un serveur HTTP de base avec fonction proxy/cache peut suffire à réaliser cette partie. Ils utilisent principalement les ressources de mémoire et de réseau de la/les machine(s).
2. *les serveurs d'application*, qui se chargent du traitement des requêtes complexes, de la validation de données : ils exécutent des scripts en divers langages (Perl, Java, etc). Ils utilisent principalement les ressources de CPU, voire de mémoire, de(s) la/les machine(s).

3. *les serveurs de données*, qui se chargent du stockage des données, voire des bases de données. Ces serveurs sont principalement chargés en réseau et stockage. Il peut-être tentant de les implémenter en NAS.

Au niveau du modèle OSI :

couche présentation dans le client léger (client WWW).

couche transport HTTP, serveur WWW

couche session stockage des données et du contexte d'application.

couche application serveurs d'applications et de stockage.

De manière à ce que cette solution puisse fonctionner, il faut en règle générale un répartiteur de requêtes, ce qui peut s'implémenter par un *load-balancing firewall*, voire par un logiciel spécifique sur chacune des stations ou plus simplement des entrées DNS *round-robin*.

Les clients de ce système sont considérés *légers* : il ne s'agit pas de stations sans disques mais elles tournent au plus un système d'exploitation de base et un client

WWW (navigateur) à l'exclusion de tout logiciel d'application, ce qui est un avantage pour la gestion du poste de travail (mais un inconvénient de performance et de répartition de charge).

Certaines implémentations peuvent également résister aux pannes d'un des composants doublés du système. Tout dépend du type de synchronisation de données et de base de données, par exemple.

En déplaçant le problème (en stockant les données sur des serveurs dédiés) on se rend compte que si tout le contexte est stockés sur ces derniers, les frontaux WWW et les serveurs d'applications sont relativement statiques (à part leur logiciel) et sont donc facilement interchangeables. Il est donc facile d'implémenter une haute fiabilité et une répartition de charge sur ces serveurs.

Le problème des *serveurs de données* et de leur synchronisation reste. Nous allons traiter plus loin des différentes méthodes et de leur qualité, simplicité et coût.

Problèmes du modèle 3 tiers

- nécessite la réécriture des applications
- données exclusivement dans les serveurs de données (sauf modèle simplifié)
- plus lourd qu'une solution *fail-over* simple
- dépend de la qualité des implémentations de synchronisation
- *identifier le goulet*
  - performance
  - besoins en synchronisation

adaptable

Il est évident qu'il faut plus de place disque en mode spécial.

Sauvegarde spécifique Une interface particulière à l'application permet d'effectuer une sauvegarde cohérente sous divers formats (binaire, sortie texte SQL par exemple). L'application est soit suspendue soit travaille en mode spécial pendant l'exécution de la sauvegarde. S'il y a assez de place disque, un fichier temporaire – par exemple compressé – peut être créé de manière à limiter le temps de sauvegarde.

La sauvegarde spécifique est cependant en règle générale moins sûre que la sauvegarde arrêtée, et moins efficace que le mode spécial.

Snapshots Un snapshot est la photographie cohérente du système de fichiers à un instant. Dans la mesure où toutes les applications sont en suspension (selon la définition ci-dessous) lors du snapshot, celui-ci est alors également cohérent pour les applications. Un snapshot est pris très rapidement et donc la suspension des applications ne dure qu'un instant très bref. Une fois le snapshot pris, le système de fichiers effectue les modifications similairement au mode spécial ci-dessus décrit. Plusieurs

snapshots peuvent être actifs simultanément et sont souvent accessibles comme un système de fichiers usuel (en lecture seulement).

Un snapshot peut (doit) être également annulé lorsqu'il n'est plus nécessaire. Il peut éventuellement servir d'historique dans certains cas. Un snapshot peut se faire en logiciel ou à l'aide de matériel dédié (*split* RAID1 p.ex., gestionnaire de volume logique sur baie de stockage externe).

Il est évident qu'il faut plus de place disque si l'on utilise des snapshots.

Les snapshots sont possibles sous GNU/Linux avec l'aide des outils LVM, notamment `lvcreate`. Il y a de la documentation également dans `lvm(8)` :

```
lvcreate --size 100m --snapshot --name snap /dev/vg1/lvol1
```

La taille à spécifier est environ celle des modifications au volume original pendant que le snapshot est actif, on peut l'étendre si nécessaire avec `lvextend`. On le supprime avec `lvremove`. Si elle est trop petite, le snapshot deviendra alors inaccessible.

Données actives A propos des sauvegardes correctes : de façon générale, sous UNIX, on peut sauvegarder des fichiers qui sont ouverts. Cela signifie qu'il faut faire attention à l'intégrité des données. En effet, faire une sauvegarde du répertoire de données de PostgreSQL (lorsque cette base de données est active) ne donnera probablement pas de bons résultats.

En règle générale, il faut arrêter les **daemons** qui peuvent modifier des fichiers avant de sauvegarder ceux-ci, ou utiliser des méthodes garanties, spécifiques à l'application, pour faire une sauvegarde. Par exemple avec PostgreSQL il s'agit de l'utilitaire `pg_dump` (ou `pg_dumpall`). L'interfaçage de ces utilitaires est en général assez facile avec les logiciels de sauvegarde simples.

Cause du problème Les fichiers actifs en ajout seulement sont déjà un problème car toutes les données ne seront pas forcément sauvées. Lorsque plusieurs fichiers forment un tout cohérent, en ajout seulement, il n'y aura pas cohérence de l'ensemble. De plus lorsqu'un fichier peut être modifié partout, la sauvegarde ne sera pas cohérente.

La cause du problème est la non atomicité de la sauvegarde. Les moyens de résoudre ce problème s'attaquent tous à la non atomicité.

Arrêt de l'application (cold backup) L'arrêt de l'application garantit dans tous les cas la cohérence des données sauvegardées. Dans certains cas cependant l'arrêt de l'application, en particulier pendant tout le temps de sauvegarde, n'est pas acceptable. C'est la solution la plus sûre, cependant.

Suspension La suspension de l'activité de l'application – qui avant d'accepter synchronise ses données et les rend cohérentes – permet de mitiger certains des problèmes de l'arrêt de l'application.

Mode spécial Un mode spécial peut être appliqué, durant lequel l'application procède comme lors d'une suspension, mais continue en suite à fonctionner, en sauvegardant les modifications aux données dans un fichier séparé. Une fois la sauvegarde terminée, l'application applique toutes les modifications effectuées aux fichiers et supprime le fichier séparé.

Modèle 3 tiers et stockage de données Le modèle 3 tiers est très *scalable* mais peut souvent être simplifié de manière à baisser les coûts. Il faut faire attention à éviter de simplement déplacer le problème de la fiabilité ou de la performance sur le serveur de données ou de SGBD. En effet, en règle générale pour assurer une véritable haute disponibilité, la base de données doit être synchronisée en temps réel. Pour assurer une bonne performance un support *load-balancing* réel est nécessaire dans la base de données.

Identifier le goulet Avant de se lancer dans la réécriture de toutes vos applications, déterminez si celles-ci sont plutôt dépendantes du réseau, du CPU, de la mémoire ou du stockage. Cela vous permettra de dimensionner votre solution correctement. Egalement, déterminez si vous pouvez tolérer des pertes de données et leur étendue. Par exemple, si votre système n'a pas une grande charge sur la base de données, une simple solution de réplication très écomomique sera suffisante et assurera un *fail-over* de celle-ci sans pertes de données.

Besoins

déterminer les bornes acceptables pour :

– interruption de service ou non
– type de reconfiguration / recâblage à effectuer
– durée de l'interruption de service
– quantité de données perdue
– performance
– coût acceptable

planification

Méthodes

- redondance
  - duplication de composants à tous les niveaux
    - RAID
    - *multipath*
  - synchronisation de données
- arrangement pour la performance
- surveillance de services (y.c. intégrité)

Données spéciales Les *forks* (MacOS, Windows 2000), les données brutes (*raw devices*), les ACLs (Access Control Lists) ainsi que les attributs étendus ext2/3 (`chattr`) ne sont souvent pas supportés par les logiciels de sauvegarde ! Il faut utiliser d'autres méthodes (dump texte, voire patches).

Fichiers qui changent Les fichiers qui changent peuvent être un problème pour le logiciel de sauvegarde. En particulier si un fichier change en répertoire pendant la sauvegarde. Une façon de corriger cela est de passer par les *snapshots* que nous verrons, également pour la sauvegarde de tout le système.

Réalisation des besoins

- pas d'interruption de service
  - protocoles conçus pour : NFS, évt. HTTP
  - prise d'adresse IP et MAC (ou NAT)
- petite interruption de service
  - protocoles : HTTP, SMB, X11, etc
  - *fail-over* simple
  - prise d'adresse IP

Fichiers actifs

non atomicité de la sauvegarde

- *arrêt* de l'application
- *suspension*
- *mode spécial*
- *sauvegarde spécifique*

logging, **snapshots**

Une convention possible ...

/	système
/data	données entreprise (/data/home, /data/www, etc)
/scratch	données temporaires non sauvegardées
/tmp	données temporaires peu volumineuses non sauvegardées, effacées au redémarrage
/var	données systèmes variant souvent

Données problématiques

- données volumineuses
- fichiers actifs
  - ouverts
  - verrouillés
  - en cours de modification
  - d'ajout
  - bases de données
  - qui changent de type
- données spéciales

OpenVMS LanManager Il s'agit de la seule implémentation d'un serveur de fichiers SMB supportant le fail-over *sans* interruption de service. Elle n'est probablement plus disponible aujourd'hui.

Réalisation des besoins (2)

- interruption moyenne
  - protocoles : HTTP, SMB, X11, etc
  - prise d'adresse IP, voire MAC (ou NAT)
  - changement de disque
  - activation manuelle du serveur de secours, recâblages
- grande interruption
  - restauration de sauvegardes
- très grande : *disaster recovery*

toujours prévoir le *disaster recovery* au minimum

RAID

Redundant(?) Array of Inexpensive/Independant Disks

– performance et/ou fiabilité (panne franche disque, voire contrôleur)
– logiciel, matériel, intermédiaire
– granularité
– bas coût aujourd'hui
– *partiel*
– parfois couplé aux *snapshots*, *volume-managers*

pas une sauvegarde !

Partiel Le RAID ne protège ni contre des erreurs humaines ni contre la défaillance d'un système ou d'un logiciel. Il ne protège que de pannes franches de disques, voire de sous-systèmes disques (câbles, contrôleurs).

Granularité On entend par granularité l'unité minimale de traitement aussi appelée parfois *stripe* ou *chunk size*. Suivant sa taille et l'application considérée, les performances seront bonnes ou moins bonnes.

Archiviste Un manque dans les entreprises modernes est celui de l'archiviste, qui tenait les archives et structurait les données passées de l'entreprise. Il avait également pour rôle de détruire des archives n'ayant plus d'intérêt.

Dans les entreprises modernes, les informations utiles sont souvent noyées dans la masse de diverses informations sans intérêt, des copies de données identiques, etc.

Certains outils logiciels permettent de mitiger ce problème (p.ex. liens *copy on write*), mais pas de le résoudre.

Séparation des données

conseil simple : séparer les données en
– système
– données entreprise
– données temporaire
– évt. données actives

définir les politiques de sauvegarde, test, réinstallation et historique en conséquence !

Problèmes On suppose le taux de compression, celui-ci peut changer. Le media est déjà utilisé à sa limite ou presque.

En cas de destruction de la cassette de sauvegarde complète, pertes de données (ajouter une sauvegarde différentielle à la dernière sauvegarde différentielle par exemple).

Ne comprend ni l'installation ni les licences logicielles si nécessaire.

En DDS-4, la sauvegarde de 15 GB de données compressées prend de 2 à 3 heures.

Enfin, en cas de panne totale, il faut compter sur la restauration d'au plus 6 cassettes incrémentales, voire l'achat d'un lecteur et des licences des programmes.

Choix des données

- toutes les données ne sont pas nécessaires
- roulement des anciennes données
- structuration des données
- *exclude* ou *include* lists
- souvent on préfère tout sauvegarder

manque l'archiviste (*politique d'archivage*)

Niveaux de RAID courants

0 accès parallèle à plusieurs disques, les données y sont réparties. Pas de redondance. Perte des données en cas de perte d'un seul disque. Chaque disque supplémentaire augmente en théorie la performance. Capacité N. *striping*. Exemple : LVM

pas *redondant* ... (performance)

Niveaux de RAID courants (2)

1 écriture simultanée sur plusieurs disques des mêmes données. Survit à la panne franche d'au plus $N-1$ disques. Performance sans changement, voire abaissée légèrement en écriture, possible augmentation de performance en lecture en cas de plusieurs flux de données. Mode dégradé utilisable pour snapshot partiel ou pour sauvegarde de données si synchronisé avec l'OS/les applications. Capacité 1. *mirroring*

performant, sûr, peu économique

RAID1 split and binding L'idée est d'utiliser un array RAID1 pour la sauvegarde de données. On indique au système d'exploitation et aux applications de se mettre en mode de sauvegarde de données (*cache flush* ou mode journal / logging p.ex.). Une fois cela fait on sort un des disques de l'array RAID1 (*split*) et on donne l'autorisation à l'OS et aux applications de reprendre le mode normal. Le disque sorti du RAID1 est une sauvegarde cohérente du système. On remplace ce disque par un autre qui est reconstruit.

L'avantage est principalement en cas de grands volumes de données. Le disque ainsi sorti peut être ensuite sauvegardé sur bandes dans de bonnes conditions.

Niveaux de RAID courants (3)

5 Arrangements de N disques dont $N-1$ fois la capacité est utile. Le dernier disque est utilisé comme disque de parité (XOR) (RAID 3), ou on a une parité tournante (RAID 5) pour éviter le goulet d'étranglement du disque de parité. En mode dégradé, un array RAID 5 est similaire à un array RAID 0.

impact du ***read-modify-write*****.**

Exemple Une sauvegarde complète (niveau 0) est faite le lundi matin. En supposant qu'un fichier est changé le lundi après-midi, la sauvegarde niveau 3 du mardi matin le sauvegardera. Mais également la sauvegarde niveau 2 du mercredi matin.

En cas de modification mercredi après-midi, la sauvegarde du jeudi matin et du vendredi matin sauvegardera le fichier.

Exemple

- 30 GB à sauvegarder sur DDS-4
- compression : données 50% compressibles
- une sauvegarde complète par semaine, déposée hors site
- une sauvegarde incrémentale par jour (1 GB/jour), déposée dans armoire protégée
- 1 mois d'historique
- 1 sauvegarde par mois conservée pendant 12 mois
- disque avec sauvegarde incrémentales
- un audit par 6 mois
- coût matériel : 2000.- + 120.- (amorti sur 3 ans)
- coût récurrent : environ 1200.- / an

Méthode simple et efficace Cette méthode permet de restaurer n'importe quel fichier de la semaine en restaurant la sauvegarde complète (0) suivie de la sauvegarde incrémentale de niveau 1 correspondante.

Niveaux de RAID courants (4)

10, 01 RAID 1 sur du RAID 0 ou le contraire

autres combinaisons

Méthode des Tours de Hanoï

but : assurer plusieurs chemins de restauration.

La *plupart* des fichiers modifiés sont sauvés exactement deux fois.

Exemple : `0 3 2 5 4 7 6 9 8 9`

lundi	mardi	mercredi	jeudi	vendredi	samedi	dimanche
0	3	2	5	4	7	6

complexe, optionnelle, utile seulement si plusieurs volumes

Types de RAID

- *matériel* : sur bus SCSI ou IDE, accessible comme disque sans installation de pilotes, démarrable
- *logiciel* : pilote logiciel, pas d'aide matérielle, démarrable ou non suivant l'OS et le niveau de RAID
- intermédiaire : pilote logiciel dans le BIOS, pilote dans l'OS, seul avantage : plus facilement démarrable que logiciel

performance, flexibilité, disponibilité, firmware

Synchronisation de données

- copie de données régulière (`rsync`, différentiel)
- envoi de journaux (réplication de SGBD)
- réplication de SGBD transactionnelle et load-balancing
- disque partagé dormant (SCSI, Fibre Channel)
- disque partagé actif (SCSI, Fibre Channel ; GFS)
- synchronisation par réseau permanente et fiable (système de fichiers, couche disque virtuelle, p.ex. **Veritas**, **Evidian**, **SGI/Debian FailSafe**, etc)
- NAS

Restauration Une vraie sauvegarde par niveau, lorsque restaurée, peut également supprimer les fichiers qui ont été supprimés dans l'intervalle.

GlobalFileSystem (GFS)

- SAN (SCSI, Ethernet, FC)
- écriture multiple
- support HA
- scalable
- journalisé
- support : **GNU/Linux**, **Solaris**, **Oracle**.

version libre existe

Résistance aux défaillances du media

méthode simple et efficace (media séparé)
- sauvegarde complète (0) le vendredi midi
- sauvegarde de niveau 1 par rapport au dernier 0, chaque jour

perte du dernier niveau 0
- perte d'un jour de modifications
- solution : incrémental de niveau 1 au niveau 0 *précédent* (voire de niveau 2 au niveau 1 précédent)

Pérennité des formats

- format
  - de sauvegarde
  - (mais également de données !)
- parfois choix support / compatibilité
- exemples de formats de sauvegarde ouverts
  - POSIX tar
  - ISO-9660 + Rock Ridge
  - cpio, afio, pax : bien répandus
  - dump : spécifique ! évoluant ! danger !
  - évt. : **Arkeia** (source du logiciel de désarchivage fourni)

souvent les formats disparaissent plus vite que le média !

Niveaux de sauvegarde

0 Sauvegarde complète (*full backup*)

1 Sauvegarde incrémentale : sauvegarde uniquement des différences par rapport à la dernière sauvegarde complète.

N+1 Sauvegarde des différences par rapport à la sauvegarde de niveau N correspondante

granularité : **fichier**, **bloc** ou **byte**

Une solution simple de fail-over

- par exemple : serveur NFS haute disponibilité
- monitoring de service (p.ex. `heartbeat`)
- synchronisation de données permanente : RAID1 sur `nbd`
- en cas de panne, prise d'adresses IP et MAC, directe ou via NAT intelligent
- *hot standby*

Debian/SGI FailSafe

Conclusion

- il existe différentes méthodes pour assurer l'intégrité et la disponibilité
- il faut définir les besoins
- toutes les applications ne sont pas simples à intégrer de manière efficace.

Détection d'intrusion

- alarmer après l'arrivée d'un intrus
- parfois alarmer avant le piratage proprement dit
- basé sur :
  - analyse de logs
  - analyse de trafic réseau
  - ajout de méchanisme dans les applications et le kernel
  - services ou machines fictives

Linux Intrusion Detection System

http://www.lids.org/

- patch kernel
- augmente la sécurité
  - fichiers/processus cachés/protégés
  - ACLs
  - capabilities
  - portscan detector
- alarmes

autres solutions *hardened kernel*, p.ex. grsecurity

Chiffrement des sauvegardes Il s'agit ici d'un choix difficile : d'un côté l'existence de media de sauvegarde non forcément strictement répertorié peut mettre en danger la confidentialité des données, de l'autre, le chiffrement peut créer un risque de déni d'accès aux données en cas de besoin.

Le chiffrement peut s'avérer nécessaire notamment lorsque les données sont déposées chez un hébergeur de sauvegarde (par exemple via Internet) ou de manière générale sur un support partagé ou si les données sont très confidentielles. Dans ce cas, une solution peut être que la machine effectuant les sauvegardes dipose de la clé de chiffrement *et* de la clé de déchiffrement et puisse vérifier ainsi la bonne qualité de la sauvegarde. Mais l'hébergeur de sauvegarde, lui, ne dispose pas de la clé. De manière à assurer l'accès aux données, la clé de déchiffrement est déposée chez un notaire (*key escrow*) avec mandat d'autoriser l'accès à un fondé de pouvoir de l'entreprise.

Coûts On parle ici de coût de la solution, mais aussi d'autres coûts potentiels comme le coût du *downtime*, de la perte de données, de la recherche de problèmes, de l'appel à une entreprise de *data recovery* et bien sûr des coûts d'exploitation de la solution.

Critères de choix du/des media

- temps de sauvegarde totale
- temps de sauvegarde incrémentale
- temps nécessaire pour récupérer un fichier isolé
- temps nécessaire pour récupérer tous les fichiers
- prix au mégabyte
- prix total de la solution
- durée de vie du support (media) et du lecteur
- taille, capacité, compacité

Besoins (2)

- *downtime* : rapidité de restauration dans les divers cas
- qualité des audits
- besoins spécifiques
- compression, chiffrement
- *coûts*

Qualité des audits Une sauvegarde qui ne peut être restaurée au besoin ou qui n'est pas complète ou cohérente n'a pas d'intérêt. Dans ces conditions, un audit régulier des supports de données (avec comparaison des données à sauvegarder) est recommandé.

Besoins spécifiques Ici, l'on doit traiter des cas particuliers comme les bases de données (sauvegarde *hot* ou *cold*, journalisation, réplication, etc) ou des applications particulières.

Compression La compression peut aider dans la plupart des cas (données compressibles) à réduire les besoins en capacité de sauvegarde, et parfois également le temps de sauvegarde. La plupart du temps, cela va également augmenter le risque de perte de données. Certains systèmes permettent de compresser fichier par fichier, ou d'au moins diminuer l'impact en cas d'erreur de compression. La compression matérielle doit être désactivée si une compression logicielle est effectuée. En général, la compression matérielle est moins efficace et moins facilement récupérable.

snort

- analyse le trafic au niveau interface et détecte des comportements à risque, des programmes automatiques ou des attaques directes sur des vulnérabilités connues
- est apparemment utilisé dans certaines *appliances* CISCO
- peut être sujet d'attaques de sécurité ! mettre à jour et de préférence tourner sur une machine dédiée et monitorée

solutions plus simples existent

Surveillance

- logs systèmes
- place disque
- charge de la machine (historique)
- alarmes
  - classifier
  - moyens (e-mail, SMS, téléphone)
  - quittances

IDS, **tests proactif**, ***honey-pot***

Honey-pot Un honey-pot est une machine (réelle ou virtuelle, voir UML ou LVS p.ex.) qui simule une cible donnée. On peut y laisser tourner des versions vulnérables de logiciels. La machine ne contient aucune information importante, n'est jamais réellement utilisée (pas de logins, etc), ne peut se connecter à des resources importantes et se trouve de préférence derrière un firewall (si possible *bridging firewall silencieux*) et un outil de détection d'intrusion.

En cas d'intrusion, une alarme est envoyée. Elle permettra, si le honey-pot est judicieusement placé (p.ex. DMZ inaccessible de l'extérieur, réseau interne, etc) de détecter un intrus assez rapidement.

Sauvegarde

motivations

- être capable d'établir une politique adaptée
- choisir les méthodes de sauvegardes suivant les besoins

ce n'est pas la sauvegarde qui importe, c'est la **restauration** !

Nombre de duplicats Dans certains cas, il faut créer plusieurs copies d'une sauvegarde (soit en dupliquant la sauvegarde, soit en effectuant à la fois une sauvegarde complète et une incrémentale par rapport à la dernière sauvegarde complète par exemple). Ces duplicats permettent de restaurer toutes les données même en cas de défectuosité d'un media.

Durée et mode de l'historique On différenciera ici la sauvegarde pure de l'archivage de données. En règle générale il n'est pas nécessaire de garantir un historique sur une longue période. Il suffit de pouvoir restaurer le système en l'état et peut-être celui d'il y a une semaine, un mois et un an. Il n'est donc en général pas nécessaire de stocker pour toujours, ou même pour 2 ans, des supports de données. En effet, dans beaucoup de cas, les données sont obsolètes assez rapidement et les données à conserver le sont sur les systèmes en production et sont donc sauvegardées à chaque sauvegarde complète.

Dans certains cas, par exemple des données sans cesse rajoutées, on passe à de la migration. Par exemple, une grande base de données de facturation peut être remise à zéro chaque année, et seule la sauvegarde de cette année contient les données. Dans ce

cas il faut faire extrêmement attention à avoir plusieurs copies (duplicat) des données, sous forme complète ou incrémentale et à la vérification des données.

Dans certains cas également il faut assurer la restaurabilité du système à l'état qui avait à une date quelconque. Ces besoins sont plus rares et peuvent souvent être réalisés par des sauvegardes incrémentales (coûteuses en temps de restauration et sensibles aux défectuosité du media).

L'historique est aussi disponible si nécessaire (au plus 6 mois, voire 1 an). Par contre, il est plus difficile d'effectuer des comparaisons régulières.

Dans cette stratégie la durée de vie du media n'est pas un des critères fondamentaux.

Buts des sauvegardes

- récupération de fichiers isolés (*day-to-day operation*)
- historique
- base de comparaison
- *disaster recovery*

divers poids et méthodes

Besoins

- quantité de données à sauvegarder (évt. données non sauvegardées)
- temps de sauvegarde et de restauration (totale, fichier)
- fréquence des sauvegardes
- nombre de duplicats
- durée et mode de l'historique

Les divers buts des sauvegardes Les sauvegardes sont en général effectuées dans le but de rendre possible la restauration de fichiers isolés (très utile dans le fonctionnement journalier d'un système informatique), dans le but d'avoir une base de comparaison si le système venait à souffrir de problème divers (intégrité de données générale, piratage, bugs logiciels), comme historique pour assurer la temporalité (suivi légal par exemple, pour récupérer des données plus anciennes ou plus simplement pour déterminer la cause d'un problème). Enfin, un dernier but est celui qui consiste à assurer, en cas de panne totale d'un système, de vol ou de désastre naturel par exemple, la restaurabilité des données.

Ces divers buts ne sont pas en règle générale atteints de la même manière, n'ont pas le même coût, ne nécessite pas les mêmes précautions et ne sont pas forcément associés à une importance identique dans différents contextes.

Principes de base

- nommer un *responsable*
- définir les *besoins*
- sauvegarder toutes les données nécessaires
- prêter attention aux cas particuliers
- *tester* : ce sont les **restaurations** qui comptent !
- *surveiller* à long terme et *adapter*

Surveiller La surveillance signifie consulter les rapports systèmes des sauvegardes, vérifier les sauvegarde par exemple en listant leur contenu et en les restaurant régulièrement (un fichier, voire tout avec comparaison intégrale moins régulièrement), consulter les compteurs d'erreurs corrigées des lecteurs de bande et des disques, vérifier les rapports de nettoyage, etc.

Mais il peut également s'agir d'adapter la politique de sauvegarde aux nouveaux besoins.

Exemple

- données centrales
- sauvegarde sur disque toutes les 2 heures (incrémentale)
- sauvegarde sur site externe incrémentale deux fois par jour, via Internet, chiffrée
- sauvegarde complète sur bandes stockées hors site une fois par semaine
- vérification des bandes par rotation, réutilisation des bandes tous les 6 mois, une bande gardée tous les 6 mois.

Données non centrales Attention à la sauvegarde de données sur les postes de travail, par exemple les ordinateurs portables, les assistants personnels (PDAs). Travailler par sauvegarde d'image si nécessaire. Ne pas oublier d'avoir les licences et clés de licences des logiciels en lieu sûr.

Certains systèmes de fichiers ou protocoles permettent le travail en mode déconnecté et la resynchronisation par la suite (**Coda**, **Intermezzo**, l'attaché-case de **Microsoft Windows** et bien sûr les systèmes de contrôle de version (**CVS** par exemple)).

Types de sauvegardes Dans ce cas, les sauvegardes sur disque toutes les deux heures assurent une restauration en cas d'effacement ou d'écrasement accidentel avec peu de pertes. Une donnée perdue peut être retrouvée en quelques secondes.

La sauvegarde sur bande est coûteuse en temps de sauvegarde et de restauration et n'est qu'effectuée une fois par semaine. Couplée à la sauvegarde deux fois par jour incrémentale sur un site distant via Internet (sauvegarde chiffrée avec une clé non accessible au serveur de sauvegarde) elle assure une restauration, en cas de désastre, de toutes les données avec perte d'au plus 4 heures. Cette restauration prendra du temps.