Methoden zur Datensicherung
— Strategien und Techniken für NetBSD —

Chemnitzer Linux-Tage 2006

Stefan Schumacher, <stefan@{net-tex,cryptomancer}.de>, PGP Key <0xB3FBAE33>

http://www.net-tex.de/netbsd/backup.html

$Header: /home/daten/cvs/backup-howto/backup-folien.tex,v 1.34 2006/03/02 14:44:09 stefan Exp $

Literatur

- http://www.net-tex.de/netbsd/backup.html
- W. Curtis Preston
  „*Unix Backup & Recovery*“, O'Reilly 1999
- www.backupcentral.com (Storage Mountain)
- Æleen Frisch
  „*Unix System-Administration*“, O'Reilly 2003, (online)
- http://www.bacula.org
- http://www.amanda.org
- http://www.postgresql.org/docs/
- Systemhandbücher / `man ...`

Motivation

- Systeme fallen aus oder werden beschädigt.
- Ein Backup ist die Sicherung relevanter Daten um diese vor Verlust oder Beschädigung zu schützen.
- Alternative für den Verlust des Originalsystems.
- Kosten für Datenverlust ./. Kosten für Sicherung

Motivation

- Systeme fallen aus oder werden beschädigt.
- Ein Backup ist die Sicherung relevanter Daten um diese vor Verlust oder Beschädigung zu schützen.
- Alternative für den Verlust des Originalsystems.
- Kosten für Datenverlust ./. Kosten für Sicherung

FdI125: Backup:
Niemand will Backup. Alle wollen Restore.
(Kristian Köhntopp zitiert einen Vertriebler)

Teil I.
Strategie und Organisation

Strategiekonzept

1. Inventur
2. Bedrohungsszenarien analysieren
3. Wichtige Daten finden
4. Alles sichern
5. Dokumentation
6. Testen, Testen, Testen

1.) Inventur

- Überblick über Geräte und Programme
- Datenbank
- eingesetzte Hardware, Betriebssysteme, Dienste erfassen
- zu sichernde Daten (Mengenmäßig)

2.) Bedrohungsszenarien 1/2

- Benutzerfehler
  Sicherung / Spiegelung mit Archivierung

- Administratorenfehler
  Sicherung / Spiegelung des *gesamten* Systems

- Festplatte defekt
  Sicherung, Spiegelung, RAID

2.) Bedrohungsszenarien 2/2

- Dateisystemkorruption
- elektronischer Einbruch / Vandalismus
- herkömmlicher Einbruch / Vandalismus / Diebstahl
- Naturkatastrophen / Höhere Gewalt
  Sicherung / Spiegelung des *gesamten* Systems, evtl. Ersatzsysteme an anderen Orten

3.) Daten finden

- Daten die nicht wiederherstellbar sind
  (z.B. zeitgebunden wie Protokolle/Logs)
- Daten deren Verlust Kosten verursacht
  (`/home`, Datenbanken, Repositorys)
- Betriebssysteme / Konfigurationsdaten
- Eventuell Trennung der Daten (Code $\leftrightarrow$ Fotos)
- Quellen statt Binaries sichern

3.) Daten finden

- Daten die nicht wiederherstellbar sind
  (z.B. zeitgebunden wie Protokolle/Logs)
- Daten deren Verlust Kosten verursacht
  (`/home`, Datenbanken, Repositorys)
- Betriebssysteme / Konfigurationsdaten
- Eventuell Trennung der Daten (Code $\leftrightarrow$ Fotos)
- Quellen statt Binaries sichern
- nach Aktualisierung (unregelmäßig, benutzergesteuert)
- in günstigen Zeitabständen (cron, anacron)
- sofort (Replikation/Synchronisation)

4.) Alles Sichern

- Lohnt es sich Rechner nicht *komplett* zu sichern?
  Reproduzierbarkeit des gesamten Systems
  Speicherbedarf vernachlässigbar

- Sicherungssysteme sichern
  Katalog (Datenbank, Logdateien)

- alles sichern und den Rest ausschließen

6.) Dokumentation & 7.) Testen

- Alles Dokumentieren (auch administrative Aufgabe)
- Für $Andere$ dokumentieren $\rightsquigarrow$ Korrekturlesen lassen
- Einweisung anderer Administratoren

6.) Dokumentation & 7.) Testen

- Alles Dokumentieren (auch administrative Aufgabe)
- Für $Andere$ dokumentieren $\rightsquigarrow$ Korrekturlesen lassen
- Einweisung anderer Administratoren

- Das System muss unter $realen$ Bedingungen getestet werden
- unbeteiligter Administrator startet Testlauf mit der Dokumentation

Sicherungsarten

- Point in Time: verbreitet in der Datenbanken-Welt
  sämtliche Transaktionen werde protkolliert, Protokolle können zu einem beliebigen Punkt wieder eingespielt werden
- Replikation auf anderes System, Hot-Standby
- Komplett: sichert alles
- Inkrementell: sichert alles was seit der letzten Komplettsicherung geändert wurde
- Differentiell: sichert alles was seit der letzten Differentiellsicherung geändert wurde

Komprimierung

- spart Platz, kostet Rechenzeit
- Hardwareunterstützung in Streamern
- Option oder Pipe an Kompressionsprogramm

Verschlüsselung

- Pseudogeräte: einbinden und unverschlüsselt sichern
- Container/Dateisysteme: verschlüsselt sichern
- Sicherung symmetrisch verschlüsseln (mcrypt, GnuPG)
- nur gängige Programme verwenden

- Verifikation!
- Robustheit sinkt
- besser: Komression und Verschlüsselung auf Dateiebene statt auf Volumebasis

Medien

1. Verlässlichkeit
2. Geschwindigkeit
3. Dauer der Rücksicherung (sog. $Time\ to\ Data$)
4. Kapazität
5. Kosten

Lagerung/Archivierung

- korrekte Lagerung der Medien (siehe Verpackung)
- Schutz vor Diebstahl/Manipulation/Feuer/Wasser ...
- gegebenenfalls Auslagerung
- Katalogisierung der Medien (Datenbank)

  Schlüssel, Name, Zweck, Typ, Dateien, Ort, Datum ...

Lagerung/Archivierung

- korrekte Lagerung der Medien (siehe Verpackung)
- Schutz vor Diebstahl/Manipulation/Feuer/Wasser ...
- gegebenenfalls Auslagerung
- Katalogisierung der Medien (Datenbank)
  Schlüssel, Name, Zweck, Typ, Dateien, Ort, Datum ...
- Archivierung der Bänder (Level-0-Bänder)
- Archivierung im System (CVS/SVN/...)
- Rotation der Medien (Verfall der Medien/Lesegeräte)

Teil II.
Sicherung einzelner Systeme

Sicherung im Basissystem

- /etc/daily und /etc/security
- legen Sicherungskopien wichtiger Daten in /var/backups an
- halten ein RCS-Repository von /etc in /var/backups/etc vor

Dateisystem-Snapshots

- konsistentes Abbild eines Dateisystems als Pseudogerät
- Schreibvorgänge stoppen, Blöcke synchronisieren, Abbild erstellen, weitermachen (Zeitansatz: $< 1s$)

Dateisystem-Snapshots

- konsistentes Abbild eines Dateisystems als Pseudogerät
- Schreibvorgänge stoppen, Blöcke synchronisieren, Abbild erstellen, weitermachen (Zeitansatz: $< 1s$)
- Snapshot erstellen
- Snapshot mit Dump sichern
- Snapshot zerstören
- echtes Dateisystem bleibt dabei im Einsatz

dump

- wichtigstes Backupprogramm, Testsieger
- liest Daten direkt vom raw-Device $\rightsquigarrow$ extrem robust
- sichert komplette Dateisysteme, benötigt `root`-Rechte
- Ausgabe in Datei oder auf Netzwerkgerät möglich (SSH!)
- berechnet Bandlänge automatisch oder aus Attributen
- beherrscht inkrementelle Sicherung via „Levels“
- Metadaten werden in `/etc/dumpdates` abgelegt
- Fileflag `nodump` und `-h0`

restore

restore verfügt über verschiedene Rücksicherungsmodi:

- -t Inhaltsverzeichnis (ehem. dumpdir)
- -r stellt gesamtes Dateisystem wieder her
- -R wie -r, um unterbrochen Lauf fortzusetzen
- -i interaktiver Modus, erlaubt Auswahl der Dateien
- -x spielt angegebene Dateien/Verzeichnisse zurück

restore

restore verfügt über verschiedene Rücksicherungsmodi:

- -t Inhaltsverzeichnis (ehem. dumpdir)
- -r stellt gesamtes Dateisystem wieder her
- -R wie -r, um unterbrochen Lauf fortzusetzen
- -i interaktiver Modus, erlaubt Auswahl der Dateien
- -x spielt angegebene Dateien/Verzeichnisse zurück
- `restore -r -f home.0 && restore -r -f home.1`
  inkrementelle Sicherung komplett zurückspielen

restore

restore verfügt über verschiedene Rücksicherungsmodi:

- -t Inhaltsverzeichnis (ehem. dumpdir)
- -r stellt gesamtes Dateisystem wieder her
- -R wie -r, um unterbrochen Lauf fortzusetzen
- -i interaktiver Modus, erlaubt Auswahl der Dateien
- -x spielt angegebene Dateien/Verzeichnisse zurück
- `restore -r -f home.0 && restore -r -f home.1`
  inkrementelle Sicherung komplett zurückspielen
- `restore -x -f home.0 www/ postgresql1/ postgresql2/`
  www/, postgresql1/ und postgresql2/ zurückspielen

tar, cpio, pax

- `tar`/`cpio` schreiben Dateien auf Band
- fehleranfällig und wenig robust, teilweise veraltet
- `pax` wurde als Ersatz erfunden
- `/bin/tar` == `/bin/cpio` == `/bin/pax`
- benötigen Puffer zur Glättung des Stroms (`buffer`/`team`)
- verschiedene Varianten existieren (star, afio)

rsync & rdiff-backup

- rsync synchronisiert *(spiegelt)* Verzeichnisstrukturen via diffs
- kopiert Dateieigenschaften, löscht optional, exclude-Liste (RegEx)
- arbeitet lokal, im Netzwerk, SSH-getunnelt *(Schlüssel)*

rsync & rdiff-backup

- rsync synchronisiert *(spiegelt)* Verzeichnisstrukturen via diffs
- kopiert Dateieigenschaften, löscht optional, exclude-Liste (RegEx)
- arbeitet lokal, im Netzwerk, SSH-getunnelt *(Schlüssel)*
- übernimmt Änderungen sofort $\rightsquigarrow$ schlecht bei Korruption

rsync & rdiff-backup

- rsync synchronisiert *(spiegelt)* Verzeichnissstrukturen via diffs
- kopiert Dateieigenschaften, löscht optional, exclude-Liste (RegEx)
- arbeitet lokal, im Netzwerk, SSH-getunnelt *(Schlüssel)*
- übernimmt Änderungen sofort $\rightsquigarrow$ schlecht bei Korruption
- rdiff setzt auf rsync auf und fügt inkrementelle Sicherungen hinzu
- hält aktuellen Spiegel und ältere Archivversionen vor
- kann Daten beliebigen Datums zurückspielen

rsync & rdiff-backup

- rsync synchronisiert *(spiegelt)* Verzeichnisstrukturen via diffs
- kopiert Dateieigenschaften, löscht optional, exclude-Liste (RegEx)
- arbeitet lokal, im Netzwerk, SSH-getunnelt *(Schlüssel)*
- übernimmt Änderungen sofort $\rightsquigarrow$ schlecht bei Korruption
- rdiff setzt auf rsync auf und fügt inkrementelle Sicherungen hinzu
- hält aktuellen Spiegel und ältere Archivversionen vor
- kann Daten beliebigen Datums zurückspielen
- `rsync -a /etc/ /usr/back/etc`
  spiegelt `/etc/` nach `/usr/back/etc/`

rsync & rdiff-backup

- rsync synchronisiert *(spiegelt)* Verzeichnisstrukturen via diffs
- kopiert Dateieigenschaften, löscht optional, exclude-Liste (RegEx)
- arbeitet lokal, im Netzwerk, SSH-getunnelt *(Schlüssel)*
- übernimmt Änderungen sofort $\rightsquigarrow$ schlecht bei Korruption
- rdiff setzt auf rsync auf und fügt inkrementelle Sicherungen hinzu
- hält aktuellen Spiegel und ältere Archivversionen vor
- kann Daten beliebigen Datums zurückspielen
- `rsync -a /etc/ /usr/back/etc`
  spiegelt `/etc/` nach `/usr/back/etc/`
- `rsync -a stefan@fenris:/etc/ /usr/back/fenris-etc`
  spiegelt fenris' `/etc/` ins lokale `/usr/back/fenris-etc`

rsync & rdiff-backup

- rsync synchronisiert *(spiegelt)* Verzeichnisstrukturen via diffs
- kopiert Dateieigenschaften, löscht optional, exclude-Liste (RegEx)
- arbeitet lokal, im Netzwerk, SSH-getunnelt *(Schlüssel)*
- übernimmt Änderungen sofort $\rightsquigarrow$ schlecht bei Korruption
- rdiff setzt auf rsync auf und fügt inkrementelle Sicherungen hinzu
- hält aktuellen Spiegel und ältere Archivversionen vor
- kann Daten beliebigen Datums zurückspielen
- `rsync -a /etc/ /usr/back/etc`
  spiegelt `/etc/` nach `/usr/back/etc/`
- `rsync -a stefan@fenris:/etc/ /usr/back/fenris-etc`
  spiegelt fenris' `/etc/` ins lokale `/usr/back/fenris-etc`

Images mit g4u

- Ein-Disketten-NetBSD
- `dd` → `gzip` → `ftp` oder Datei
- Betriebssystem- / Dateisystemunabhängig
- sichert *gesamtes* System
- geeignet für Cluster

```
uploaddisk benutzer@ftp.server.local Imagename.gz wd1
slurpdisk benutzer@ftp.server.local Imagename.gz wd0
copydisk sd0 sd1
```

Teil III.
Sicherung verteilter Systeme

Amanda

- Ziel: n Rechner sichern auf ein Bandlaufwerk
- Verwaltet Sicherungsprogramme im Netzwerk

Amanda

- Ziel: n Rechner sichern auf ein Bandlaufwerk
- Verwaltet Sicherungsprogramme im Netzwerk
- Führt Statistiken und legt Sicherungslevel selbst fest
- $dumpcycle \rightsquigarrow$ gibt Zeitraum für eine Komplettsicherung an
- Gesamtdaten werden durch Tage in $dumpcycle$ geteilt und portionsweise gesichert
- zusätzlich werden geänderte Daten inkrementell gesichert

Amanda

- kann auf Bänder, Wechsler, Platten und RAIT schreiben
- führt Index für die Bänder in Logdateien
- kann bestimmte Dateien im Index wiederfinden
- hat eigenes Programm zur Wiederherstellung, das mit dem Index die passenden Bänder sucht
- funktioniert aber auch ohne Index

Bacula

- Datensicherungssystem speziell für verteilte Systeme
- Client-Server-Betrieb mit Dæemons

Bacula

- Datensicherungssystem speziell für verteilte Systeme
- Client-Server-Betrieb mit Dæemons
- Director (bacula-dir, Steuerungsprozess)
- Storage (bacula-sd, Schreibt Daten auf Band)
- File (bacula-fd, liefert zu sichernde Daten an)
- Catalog (PostgreSQL, Index für gesicherte Dateien)
- Console (bconsole, Konsole)

Bacula

- kommuniziert im Netz (Cram-MD5, TSL/SSL)
- unterstützt Bandlaufwerke, Bandwechsler und Sicherungen auf Festplatte
- klicki-bunte Konsolen existieren
- Betrieb als Client möglich
- Clients für MS-Windows und fast jedes Unix existieren
- verwendet keine Levels, stattdessen *Full*, *Incremental* und *Differential* mit Datumsangabe im Schedule

Bacula

- Konfiguration in Direktiven
- FileSet: was
- Client: welcher Rechner
- Schedule: wann
- Pool: wohin
- Job: fasst alles zusammen

Bacula: Optionen

Einige Optionen:

- Fileset: Include, Exclude, signature, compression, verify
- JobDefs: Level, Client, FileSet, Schedule, Storage, Pool
- Schedule:

```
Run = Full sun-sat at 21:00
Run = Differential 2nd-5th sun at 21:00
Run = Incremental mon-sat at 21:00
```

- allgemeine Steuerung über Konsole

Bacula: Rücksicherung

1. Die letzten 20 Jobs können restauriert werden
2. Es werden alle Jobs angezeigt, die eine bestimmte Datei gesichert haben. Einer der Jobs kann restauriert werden.
3. Eine kommaseparierte Liste mit JobIds wird übergeben.
4. Man kann SQL-Anfragen stellen um die passende JobId zu finden. Diese ID kann dann mit Punkt 3 restauriert werden.
5. Sichert den letzten Stand eines Clients zurück, in dem alle notwendigen Sicherungen identifiziert werden.
6. Wie 5.), allerdings kann man ein Datum spezifizieren
7. Eine Datei mit zurückzusichernden Dateinamen wird eingelesen
8. Man kann Dateinamen und ein korrespondierendes Datum angeben

Teil IV.
PostgreSQL

Sicherungsvarianten

- Datenbankcluster sichern
- Logisches Backup (pg_dump)
- Point-in-Time-Recovery mit Write-Ahead-Logs
- Replikation (hier: Pgpool)

PostgreSQLs Cluster sichern

- Konfiguration, Daten und Metadaten werden als normale Dateien im Cluster abgelegt
- können wie normale Datei gesichert werden
- Hohe Schreibaktivität $\rightsquigarrow$ Inkonsistenzen
- Snapshots verwenden, Tablespaces beachten
- kann nur mit der selben PostgreSQL-Version gelesen werden

Logisches Backup

- pg_dump, pg_dumpall und pg_restore
- erzeugen ASCII-Textdatei mit SQL-Befehlen zum Erstellen der Datenbank(en)

```
$ /usr/pkg/bin/vacuumdb -Upgoperator -f -z meineDB

$ /usr/pkg/bin/pg_dump -Fc -Upgoperator -meineDB >
/home/pgsql/meineDB_'date +%y%m%d'

$ /usr/pkg/bin/pg_dumpall > pg_'date +%y%m%d'
```

Point-in-Time-Recovery

- Point-in-Time-Recovery: Datenbankzustand kann auf beliebigen, konsistenten Zeitpunkt zurückgesetzt werden.
- Write-Ahead-Logs protokollieren alle Datenbankaktionen mit
- Standard für WALs: Drei Dateien mit je 16MB im Cluster
- werden normalerweise reihum überschrieben $\rightsquigarrow$ sichern
- `archive_command` in postgresql.conf setzen (cp/scp/rsync ...)
- darf nicht fehlschlagen, da sonst Rotation abbricht

Point-in-Time-Recovery: Rücksicherung

- PostgreSQL installieren und einrichten
- Cluster zurückspielen, Rechte anpassen
- recovery.conf erstellen und `restore_command` und `recovery_target_time`
- PostgreSQL starten und Kaffee trinken gehen, denn es werden alle Datenbankaktionen erneut durchlaufen

Replikation

- synchronisieren Datenbestand
- synchron (sofort) oder asynchron (nacheinander)
- ermöglicht sog. Hot-Standby-System das auf aktuellem Stand ist
- asynchron: COMMIT auf Server, Verteilung auf Replikanten
  nur ein Master (wegen Konsistenz)
- synchron: verteilt Daten sofort bei Beginn der Transaktion, erwartet COMMIT von allen Replikanten (Zwei-Phasen-Commit)

Pgpool

- Connection Pool mit synchroner Replikation
- cachet Verbindungen zu Datenbankservern
- kann zwischen Servern umschalten
- ermöglicht „Write-Only“-Replikation
- Erwartet als Optionen Adresse und Port der Server
- kann transparent agieren (aber Deadlock-Gefahr!)