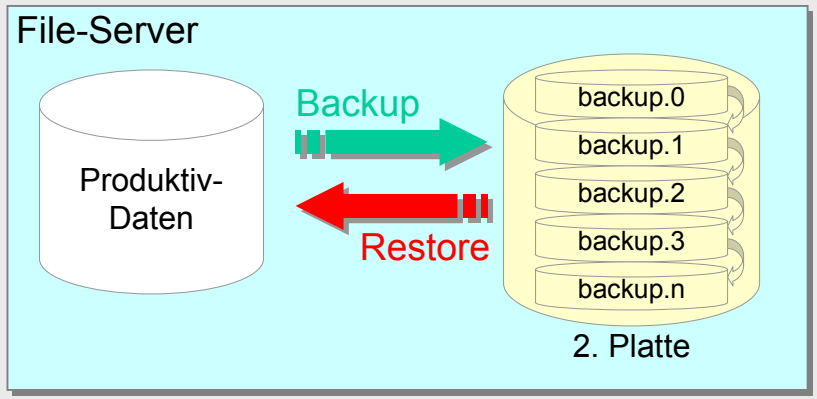
Automatische Datensicherung für Home- und Workgroup-Server mit rsync und hard links

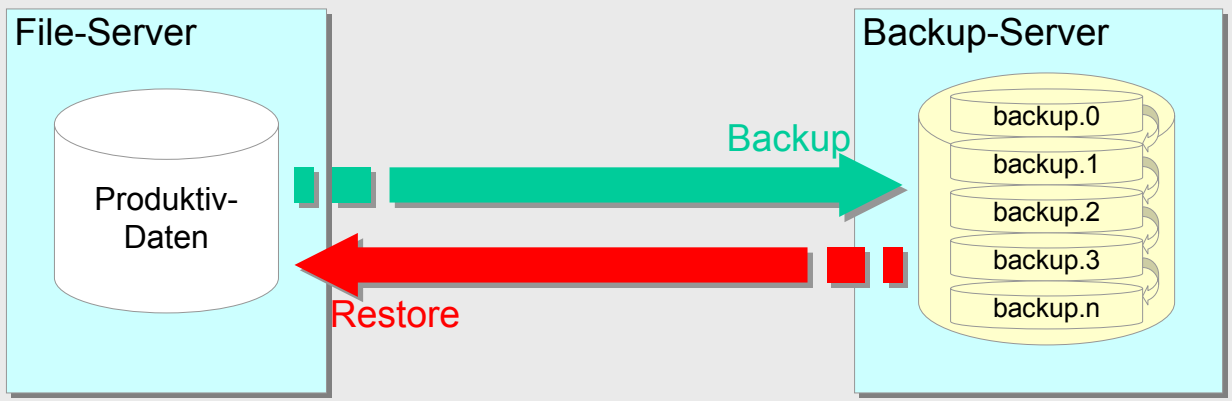
Hans-Jürgen Beie <hjb@pollux.franken.de>

Inhalt

Szenario 1: Einzelner (Home-) Server



Szenario 2: Getrennte Produktiv- und Backup-Server (Workgroup)



Einführung – Ziele / Voraussetzungen

Worum geht es?

- regelmäßige Sicherung von Daten (täglich, wöchentlich, ...)
- kleines bis mittleres Datenvolumen (≤ 100 GB)
- schnelle Sicherung auf preiswertes Medium (Festplatte)
- Verwendung von Standard-Software
- flexibel konfigurierbar
- einfache Restore-Funktion
- ... etwas für faule Admins =;-)

Worum geht es nicht?

- Langzeitarchivierung
- Versions-Management
- Spezielle Backup-Hardware/Software

System-Voraussetzungen (für Backup-Server)

- Unix (beliebiges Derivat)
- rsync
- GNU File Utilities (**`cp, mv, rm, mkdir`**)

Einige Grundlagen, für unser Backup-Karussell

- rsync
  - Ein Standard-Tool zur Remote-Synchronisation von Dateien
- hard links
  - Was hat es damit auf sich?
  - Wozu ist das gut?
- cp und hard links
  - Verknüpfen von Dateien und Verzeichnissen
  - Wie man “Kopien“ erzeugt, die keinen Speicherplatz benötigen.

rsync

- Ein Tool zur Remote-Synchronisation von Dateien und Verzeichnissen
- Ist auf den meisten Unix-basierten System bereits vorhanden
- Stammt ursprünglich von Andrew Tridgell (http://rsync.samba.org)
- Quellverzeichnis rekursiv in ein Zielverzeichnis kopieren

```
rsync -a source_dir dest_dir/
```

  Dabei werden nur die Dateien kopiert, die im Zielverzeichnis nicht vorhanden sind oder sich von denen im Zielverzeichnis unterscheiden.
- Lokales Verzeichnis mit Daten von einem anderen Rechner synchronisieren

```
rsync -a remote.host:/source_dir /backup/
```

  Das geht auch via secure shell

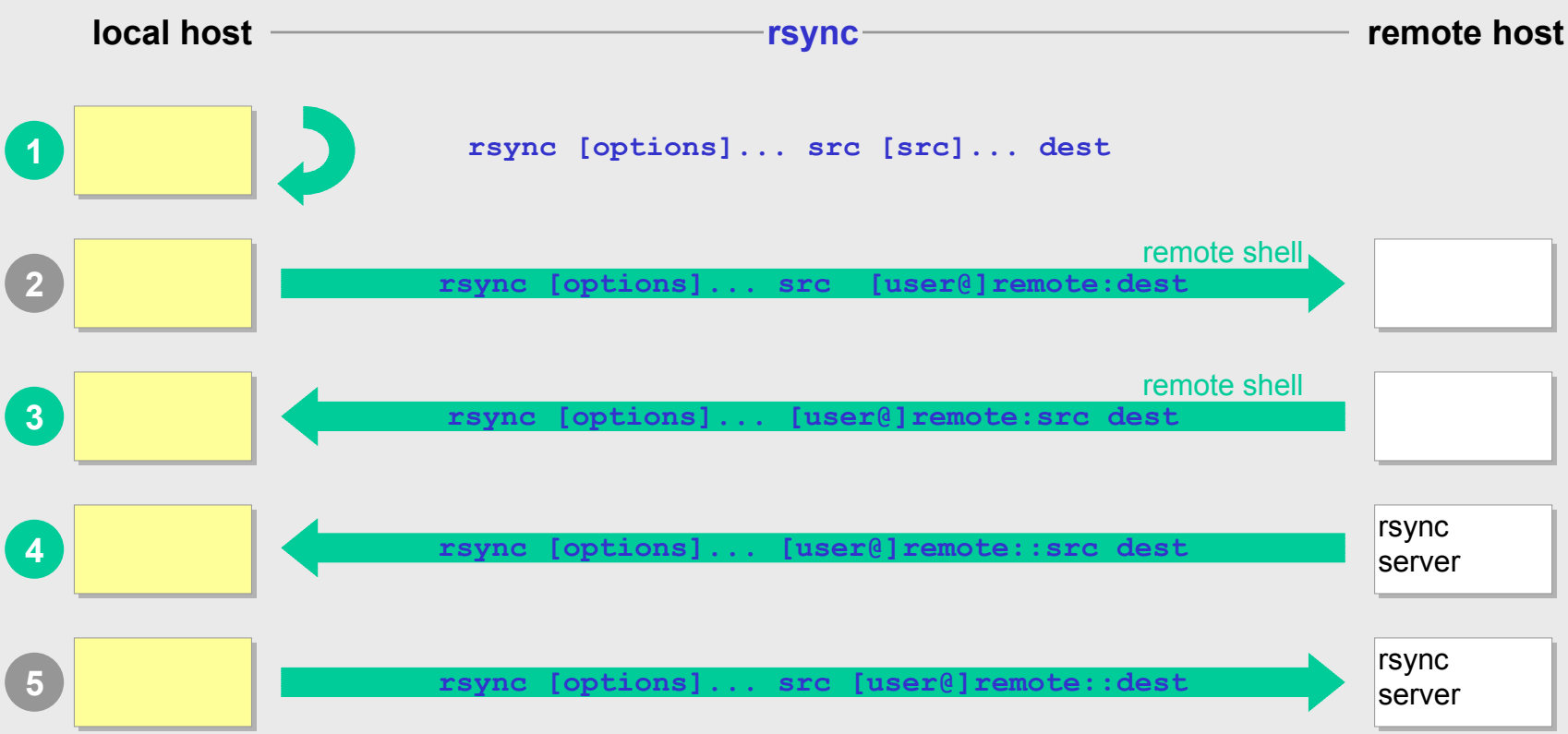
```
rsync -a -e ssh user@remote.host:/source_dir /backup/
```

- Optionen, die wir häufig brauchen

-a	Archiv-Modus (entspricht **-rlptgoD**)
--delete	Lösche Dateien im Ziel, die auf der Quellseite nicht existieren.
--exclude-from=FILE	Ignoriere Dateien, die in FILE beschrieben sind.
--delete-excluded	Lösche ausgeschlossenen Dateien auch auf der Empfängerseite.
-v	Verbose: sage, was Du tust.
--stats	Zeige eine Zusammenfassung nach der Aktion.

Mehr dazu mit: **man rsync.**

Die fünf Arten mit rsync zu Synchronisieren



1 3 und 4 sind die Varianten, die wir für das Karussell verwenden.

rsync ist speicherhungrig

rsync verwaltet eine Liste der zu synchronisierenden Dateien im RAM (ca. 100 Byte pro Datei).
Diese Liste wird vor dem eigentlichen Kopiervorgang unter Berücksichtigung der Include/Exclude-Muster und der Dateien im Quell-/Ziel-Verzeichnis aufgebaut.
Also Kopiervorgänge in “handliche“ Einheiten aufteilen!

rsync ist schnell

Remote-Kopien mit rsync sind deutlich schneller als solche über NFS (nicht selbst getestet).
Im Zweifelsfall sollte man also direkt mit rsync und nicht über NFS kopieren.
Sichern über Samba-Shares von Windows-Systemen bremst auch sehr deutlich!

rsync ist mächtig

Vor den ersten größeren Kopieraktionen (nicht mit Produktivdaten) die Doku lesen!

rsync läuft auch unter Windows (Cygwin)

aber aufgrund einiger Eigenarten von Windows kommt es immer wieder zu Problemen, z.B.

- unterschiedliche Auflösung bei den Datei-Zeitstempeln
- I/O-Engpässe bei der Übertragung großer Datenmengen

Man sollte sich (noch) nicht auf automatische, unbeaufsichtigte Datensicherungen von Windows-Systemen mit rsync verlassen. Wer von den beiden das Sorgenkind ist, lassen wir hier offen!

hard links

- Eine Datei wird im File-System durch einen Block an Informationen - den sogenanten **inode** - repräsentiert.
- Ein **hard link** ist ein Zeiger auf eine Datei. Ein **hard link** zeigt auf einen **inode** und gilt nur innerhalb ein und desselben File-Systems (innerhalb der gleichen Partition).
- Jeder Dateiname ist ein **hard link**.

- Mit dem Kommando **`ln`** (wie link)

```
ln old_name new_name
```

  wird der Datei **`old_name`** ein neuer (zusätzlicher) Name **`new_name`**, also ein weiterer **hard link**, zugewiesen.
- Der Speicherplatz, den eine Datei beansprucht, wird erst dann wieder freigegeben, wenn alle hard links, die auf die Datei verweisen, gelöscht sind.

Kopien ohne Speicherplatz

- Das Kommando **cp** zum Kopieren von Dateien kann ebenfalls **hard links** erzeugen:

  ```
  cp -al source_dir dest_dir
  ```

  kopiert das Quellverzeichnis **`source_dir`** rekursiv unter Beibehaltung aller Dateiattribute in das Zielverzeichnis **`dest_dir`**.

- Dabei werden jedoch nur Verknüpfungen (**hard links**) mit den ursprünglichen Dateien erzeugt. Es werden nicht die Daten selbst (physikalisch) kopiert, sondern es werden nur neue Dateinamen in einem neuen Verzeichnis erzeugt, die auf die ursprünglichen **inodes** zeigen.
  Dabei wird also (fast) kein zusätzlicher Speicherplatz beansprucht.

- verwendete Optionen:
  - **`-a`** Archiv-Modus: rekursiv, Dateiattribute beibehalten, symbolischen Links nicht folgen
  - **`-l`** Quelle(n) nicht kopieren, sondern Verknüpfung(en) erstellen

alles zusammen: Das Karussell

1. Backups rotieren (umbenennen)

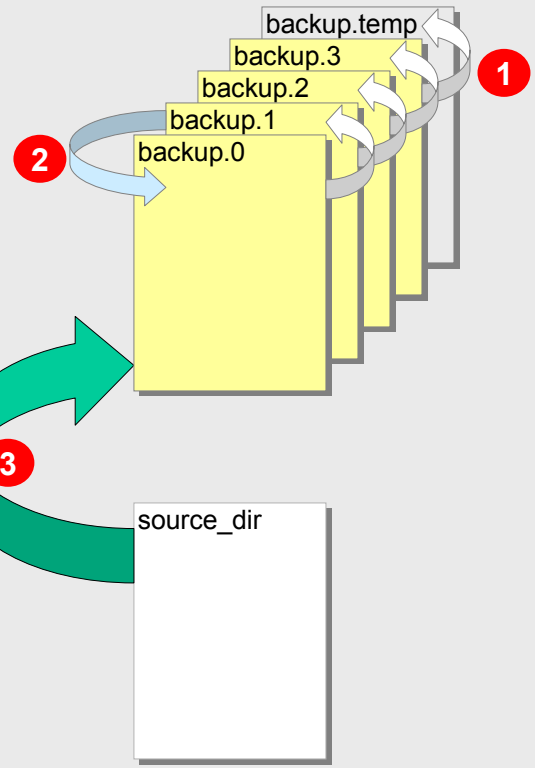
```
mv backup.3 backup.temp  (älteste Daten)
mv backup.2 backup.3
mv backup.1 backup.2
mv backup.0 backup.1 (jüngste Daten)
```

2. letztes Backup über hard links “duplizieren“

```
cp -al backup.1 backup.0
```

3. letztes Backup mit Quellverzeichnis synchronisieren

```
rsync -a --delete source_dir backup.0/
```



Am nächsten Tag, nächste Woche, ...
macht man das Gleiche wieder.

Schritte (1) und (2) laufen sehr schnell, da dabei nicht wirklich Daten kopiert werden.

Die Geschwindigkeit von (3) hängt von der Menge der neuen/geänderten Daten ab.

Backup – automatisch

Das Karussel drehen...

- zeitgesteuert – über cron jobs
- Backup-Karussel in ein Shell-Skript verpacken – die Minimallösung
- Verschiedene Backup-Aufgaben in verschiedene Shell-Skripte einpacken – besser

  ...

  Das wird leider schnell unübersichtlich und ist umständlich zu verwalten :-|

Mehrere Karussels drehen lassen...

... mit "Frontends", die dem Systemadministrator solche Aufgaben erleichtern:

- Die Backup-Aufgaben werden in Konfigurationsdateien festgelegt.
- Das Frontend kümmert sich um die Details: Verwaltet die nötigen Link- und rsync-Aktionen, überwacht die Prozesse und erzeugt Log-Files :))

Mehrere Karussels drehen lassen... z.B. mit ***rsback***

- ***rsback*** läuft auf dem Backup-Host und erledigt schematisierte Sicherungs-Aufgaben (tasks) nach dem hier beschriebenen Verfahren.
- ***rsback*** erzeugt und verwaltet rotierende Backup-Archive von lokalen Dateien und/oder von Daten auf Remote-Hosts mit variabel einstellbarer Verfallsdauer.
- ***rsback*** wir typischer weise über cron jobs gestartet

Mehr dazu unter *http://www.pollux.franken.de/hjb/rsback/*

Beispiel für ein Backup Repository

```
/BACKUP
+--/work
|    history.daily
|    history.weekly
|    +--/daily.0
|    +--/daily.1
|    +--/daily.2
|    +--/daily.3
|    +--/daily.4
|    +--/daily.5
...
|    +--/weekly.0
|    +--/weekly.1
|    +--/weekly.2
|    +--/weekly.3
|    +--/weekly.4
...
```

History des Tasks "daily"

History des Tasks "weekly"

Letzter Archivbaum der täglichen Sicherung

frühere Tagessicherungen

Letzter Archivbaum der wöchentlichen Sicherung

frühere wöchentliche Sicherungen

Ein wöchentliches Backups kann man aus einer Tagessicherung mit `cp -al daily.0 weekly.0` erstellen

```
[hjb@cebulon BACKUP]$ cat history.weekly
# rsback-0.5.0rc3 (hjb -- 2003-01-27)
0       2003-11-17 06:45:03
1       2003-11-10 06:11:36
2       2003-11-03 05:36:57
3       2003-10-27 05:14:45
4       2003-10-20 06:22:01
5       2003-10-13 06:37:26
6       2003-10-06 05:08:30
7       2003-09-29 06:11:06
8       2003-09-22 06:21:53
9       2003-09-15 05:08:37
10      2003-09-08 06:13:58
11      2003-09-01 06:06:29
12      2003-08-25 06:32:46
```

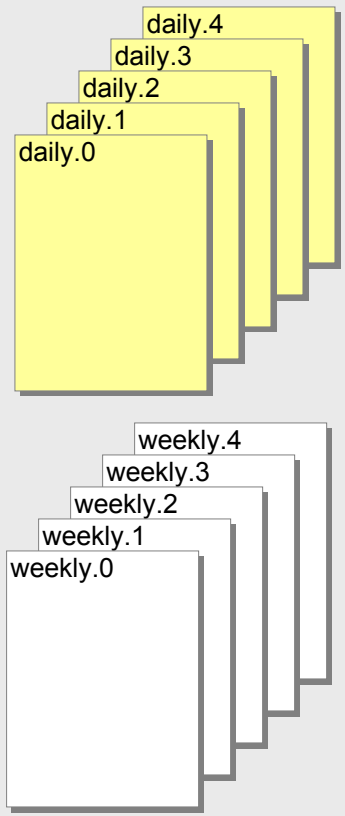
Daten weg! ... Was nun?

- Restore einfach durch Zurückkopienen der Dateien aus dem Repository

```
cp /BACKUP/daily.3/besonders/wichitg.doc ./
```

- Die Arbeit dem User überlassen :-)
  - Backup Repositories über NFS read-only mounten und mit einem symbolic link in das ursprüngliche Quellverzeichnis hängen.
  - Bei Windows-Clients: Samba Share für das Backup Repository einrichten (read-only!)
  - Oder beides
- Es ist keine spezielle Restore-Software nötig. Die Daten liegen genau so in den Archiven, wie die ursprünglichen Originale auch.
- Für Restaurationen größeren Umfangs empfiehlt sich:

```
cp -a BACKUP/daily.3/* da/wo/es/frueher/war/
```



Zusammenfassung

Automatische Datensicherung mit rsync und hard links

- ist schnell: nur beim ersten Lauf werden tatsächlich alle Daten kopiert
- spart Speicherplatz: ein großer Teil der Daten ist in den Backup-Archiven i.d.R. nur einmal physikalisch vorhanden
- bietet einfache Restore-Möglichkeit: Benutzer können ihre Daten meistens ohne Hilfe eines Systemadmins selbst restaurieren, wenn die Archive für sie "transparent" gehalten werden
- kann in Netzwerken leicht zentral verwaltet werden

Nachteile / Einschränkungen

- Die Besitz- und Zugriffsrechte in den älteren Archiven entsprechen immer dem Zustand der jüngsten Sicherung (`cp -al` legt nur einen neuen Link auf eine Datei und damit auch auf die Rechte)
- Eine Backup-Platte stirbt statistisch genau so schnell wie eine Platte mit Produktivdaten. Also sollte man für wichtige Daten am besten mindestens zwei Backup-Repositories auf verschiedenen Platten/Rechnern haben.

Referenzen

- "Easy Automated Snapshot-Style Backups with Linux and Rsync"
  Mike Rubel
  http://www.mikerubel.org/computers/rsync_snapshots/

- rsync Hompage
  http://rsync.samba.org/

- "GNU File Utilities"
  http://www.gnu.org/software/fileutils/

- "Dirvish is a fast, disk based, rotating network backup system..."
  J.W. Schultz
  http://www.pegasys.ws/dirvish/

- "rsback - backup file trees in rotating archives"
  Hans-Jürgen Beie
  http://www.pollux.franken.de/hjb/rsback/