

# Ein- und Ausgabe von Dateien

## Seminar Betriebssysteme

Michael Rennecke

Lehrstuhl für Technische Informatik  
Institut für Informatik  
Martin-Luther-Universität Halle-Wittenberg

19. Februar 2006

# Geschichte

- teure Server-HD's  $\Rightarrow$  waren sicher
- günstige PC-HD's  $\Rightarrow$  fielen oft aus
- Redundanz  $\Rightarrow$  mehrere PC-HD's
- sind günstiger als Server-HD's
- Lösung: RAID, 1987 University of California

RAID ist die Abkürzung von **R**edundant **A**rray of **I**nexpensive **D**isks (*Redundante Anordnung preiswerter Festplatten*).

Heute: **R**edundant **A**rray of **I**ndependent **D**isks (*Redundante Anordnung unabhängiger Festplatten*)

# Einstieg

- Zusammenschluss von Festplatten
- aus Nutzersicht keine Unterschied zwischen RAID-Laufwerk und normaler HD
- Speicherung von Redundanz und Daten
- Rekonstruktion verloren gegangener Daten
- Redundanz
  - Daten selbst  $\Rightarrow$  spiegeln
  - Parity Daten  $\Rightarrow$  aus mehreren Datenblöcken berechnet

# Ziele

- 1 Erhöhung der Datensicherheit (Redundanz)
- 2 Steigerung der Transferraten (Performance)
- 3 Aufbau großer logischer Laufwerke
- 4 Austausch von Festplatten und Erhöhung der Speicherkapazität während des Systembetriebes
- 5 Kostenreduktion durch Einsatz mehrerer preiswerter Festplatten
- 6 schnelle Steigerung der Systemleistungsfähigkeit

## Achtung!

Kein Schutz vor löschen von Daten bzw. Diebstahl/Zerstörung von Festplatten.

# RAID 0 oder Data Striping

## Fakten

- eigentlich keine RAID  $\Rightarrow$  fehlende Redundanz
- Festplatten werden zusammen geschaltet
- Schreib- und Leseoperationen laufen parallel
- relativ neuer RAID-Level
- Teilt Daten in Stücke

# RAID 0 oder Data Striping

Kapazität	½ HD * Kapazität (der kleinste HD)
Geschwindigkeit	sehr hoch
Ausfallwahrscheinlichkeit	sehr hoch
Kosten	sehr gering, da volle Auslastung der HD's
Anwendungen	Video-Schnitt, Temporärspeicher

# RAID 1 oder Drive Mirroring / Drive Duplexing

## Fakten

- gerade Anzahl von HD's
- 2 HD's enthalten die gleichen Daten  $\Rightarrow$  volle Redundanz
- hohe Leseperformance
- Spiegeln  $\Rightarrow$  HD's an einen Controller  $\Rightarrow$  max. 1 Sektor Verlust
- Dupelxing  $\Rightarrow$  HD's an mehreren Controllern  $\Rightarrow$  nie Verlust

# RAID 1 oder Drive Mirroring / Drive Duplexing

Kapazität	$\frac{1}{2} * \# \text{ HD} * \text{Kapazität}$
Geschwindigkeit	hoch
Ausfallwahrscheinlichkeit	gering, eine HD darf ausfallen
Kosten	sehr hoch, da halbe Auslastung der HD's
Anwendungen	Datenbanken, BS, hohe I/O-Last

## Beachte

RAID 1 ist für Echtzeit-Systeme absolut notwendig.

# RAID 2 oder Hamming-System

## Fakten

- RAID-Level für die alten Großrechner
- wird heute nicht mehr eingesetzt
- Daten werden in Bitfolgen fester Größe zerlegt
- Abbildung auf größere Bitfolgen (z.B. 8+3)
- ECC wurde mit Hamming-Algorithmus berechnet
- ECC wurde auf extra HD's gespeichert
- # HD Vielfaches der Hammingcodewortlänge

# RAID 3 und RAID 4

## Fakten

- Parity-Daten werden XOR über die Blöcke (RAID 4) bzw. über die Bytes (RAID 3) berechnet
- fest definierte Paritätsplatte
- Paritätsplatte wird zum Flaschenhals

# RAID 5 oder Block Striping mit verteilter Parity

## Fakten

- wie RAID 4, Parity wird über alle Platten verteilt
- sehr beliebter RAID-Level
- günstige Methode für Redundantes System
- schlechte Schreibeigenschaften
- Rekonstruktion der Daten dauert lange
- hohe Abnahme der Performance, wenn eine HD fehlt

# RAID 5 oder Block Striping mit verteilter Parity

Kapazität	$(\# \text{ Festplatten} - 1) * \text{Kapazität}$
Geschwindigkeit	normal
Ausfallwahrscheinlichkeit	gering, eine HD darf ausfallen
Kosten	gering, nur eine HD Verlust
Anwendungen	Fileserver, Testsystem, Archivierung, Sever

# RAID 6 oder Block Striping mit verteilter Parity auf 2 HD's

## Fakten

- Weiterentwicklung von RAID 5
- Parity je nach Implementation
  - wie bei RAID 5
  - erst wie bei RAID 5 und dann nochmal über alles
- Schreiben noch schlechter als RAID 5

# RAID 6 oder Block Striping mit verteilter Parity auf 2 HD's

Kapazität	$(\# \text{ Festplatten} - 2) * \text{Kapazität}$
Geschwindigkeit	normal
Ausfallwahrscheinlichkeit	sehr gering, zwei HD's
Kosten	normal, zwei HD's Verlust
Anwendungen	Fileserver, Testsystem, Archivierung, Sever

# NRAID oder Festplattenverbund

## Fakten

- eigentlich kein RAID
- mehrere logische HD's werden zu einen Laufwerk verbunden
- HD's müssen nicht die gleiche Größe haben (250GB + 120GB  
⇒ 370GB)
- keine Redundanz
- kein Performancegewinn

# Kombinierte RAID-Level

## Fakten

- ordnet die RAID-Level in einer Baumstruktur an
- nutzt Vorteile von mehreren RAID-Level
- Schreibweise RAID 5+1 (2 RAID 1 bilden ein RAID 5)

# RAID 01 und RAID 10 oder Mirrored Striped RAID

## Fakten

- wurde erst spät geschaffen
- sehr beliebter kombinierter RAID-Level
- Kombination von aneinander reihen von HD's und spiegeln
- Vorteile: hohe Geschwindigkeit und Sicherheit

# RAID 01 und RAID 10 oder Mirrored Striped RAID

Kapazität	$\frac{1}{2} * \# \text{ Festplatten} * \text{Kapazität}$
Geschwindigkeit	sehr hoch
Ausfallwahrscheinlichkeit	gering, eine HD darf ausfallen
Kosten	hoch nur halbe Auslastung
Anwendungen	Datenbanken, hohe I/O-Last

# Datenrettung

- nur relevant wenn man Redundanz hat
- kann auch Wege eingehen, die nicht RAID spezifisch sind
- RAID 1: einfach intakte Platte weiter nutzen
- theoretisch auch bei anderen RAID's möglich aber mit hohen Leistungsverlust, bei RAID 2 und RAID 6 sehr schlecht
- Redundanz ist nicht mehr gewährleistet
- Lösung: Austauschen der defekten HD, oder Einsatz von Spare-Disk's

# Methoden der Datenrettung

- 1 System in sicheren Zustand bringen und abschalten, dann Austausch der defekten HD
- 2 Hot-Plugging
  - 1 Austausch der defekten HD bei laufendem System (muss von Hardware unterstützt werden (SCSI))
  - 2 einbinden der Spare-Disk (macht das BS) und Austausch der defekten HD  $\Rightarrow$  System sehr schnell wieder redundant

# Software-RAID

- Zusammenwirken der Festplatten komplett softwaremäßig realisiert
- keine besondere Hardware nötig, wie RAID-Controller
- Vorteile
  - **kann** schneller als „Billig-Hardware-RAID“ sein
  - kann RAM als Cache nutzen
  - sehr günstig
- Nachteile
  - Belastung der CPU, (auch bei „Billig-Hardware-RAID“)
  - Cache auf evtl. vorhanden RAID-Controller wird nicht genutzt

# Nutzen

- begrenzender Faktor ist heute die Festplatte
- man kann bestimmte Ordner vor Datenverlust schützen
- verursacht auch keine wesentlichen höheren Kosten
- man nutzt kostenloses Feature

# Benchmark

Modus	#HD	Chunk-Size	Blockgr. ext2	schreiben	lesen
normal	1	-	4kByte	$s := Ref.$	$r := Ref.$
RAID 0	3	4kByte	4kByte	$2,6 * s$	$2,8 * r$
RAID 5	3	32kByte	4kByte	$1,7 * s$	$1,9 * r$
RAID 10	4	4kByte	4kByte	$1,7 * s$	$2,5 * r$

# Einstieg und Anforderungen

- 1 man Dateien lesen, schreiben, löschen, ändern und erstellen können
- 2 Dateien vor den Zugriff anderer Benutzer schützen können
- 3 die Zugriffsrechte für seine Dateien selbst festlegen können
- 4 seine Dateien sicher und wiederherstellen können
- 5 seine Dateien selbst organisieren können
- 6 seine Dateien über symbolische Namen identifizieren können
- 7 die Ressource Festplattenplatz für jeden Benutzer reglementieren können
- 8 soll ökonomisch mit den vorhandenen Ressourcen umgehen

# Hierarchisches Dateisystem

- Verzeichnisse und Dateien befinden sich in Stammverzeichnis
- diese Struktur wird als Baum bezeichnet
- man kann auch andere Dateisysteme in den Baum einhängen  
Bei Linux alle Geräte und evtl. andere Partitionen

## Bsp. wie man auf die Datei *Vortrag.tex* zugreift

- C:\Dokumente und Einstellungen\Michael\Eigene Dateien\BS\ Vortrag.tex (Windows 2000/XP)
- /Users/Michael/BS/Vortrag.tex (Mac OS X, (: statt / bei alten Versionen))
- /home/Michael/BS/Vortrag.tex (Unix / Linux)

# Dateiorganisation

- Festplatten haben eine Blockstruktur
- aus Sicht des (modernen) BS lassen sich nur Blöcke lesen und schreiben
- Festplatte zeigt sich als Fläche mit nummerierten Blöcken
- ein Block ist meist 512 ( $2^9$ ) Byte groß

## Frage

Wie ordnet man freie Blöcke eine neuen Datei zu (*Dateiallokation*)?

# Dateiallokation

Es gibt folgende Dinge zu bedenken:

- 1 Wenn man eine Datei erstellt, soll man ihr dann den maximal erforderlichen Platz zuweisen oder soll man Speicherplatz dynamisch zuweisen?
- 2 Wenn eine Datei über mehrere Blöcke geht, wie groß soll ein Block sein oder sollte man besser variable Blöcke nehmen oder soll man Eindatei-Blöcke nutzen?
- 3 Welche Struktur soll man für die Verwaltung der Blöcke nehmen?

# Vergleich: Voraballokation und dynamische Allokation

## Voraballokation

- Größe muss im Voraus bekannt sein (Compiler, Archivieren,.. )
- Lokalitätsprinzip
- man fordert mehr an als nötig ist
- Verschwendung von Speicher

## dynamische Allokation

- fordert Speicher stückweise an
- gute Speicherausnutzung
- Datei über Festplatte verteilt

# Blockgröße 1/2

Man kann die Blöcke so groß wählen, dass die gesamte Datei hinein passt oder es passt nur ein Bit, Byte, Wort, hinein.

- zusammenhängender Speicherplatz ist gut in Bezug auf I/O-Operationen von großen Dateien
- viel kleine Blöcke lässt die FAT schnell wachsen
- Blöcke fester Größe sind bei der Neuallokation einfach zu handhaben
- Blöcke variabler Größe wird kein Speicherplatz verschwendet

## Blockgröße 2/2

Das den Punkten lassen sich folgende Lösungen ableiten:

- 1 **Große Blöcke, variabler Größe** ermöglichen ein gutes Leistungsverhalten und es wird kein Speicherplatz verschwendet, außerdem ist die FAT klein. Es ist aber schwer frei werdenden Speicherplatz wieder zu bewenden.
- 2 Kleine **Blöcke** fester Größe bieten ein hohes Maß an Flexibilität. Jedoch wird die FAT größer, oder man muss sich eine gute Datenstruktur überlegen. Die Blöcke werden nach Bedarf zugeordnet.

# Dateiallokationsverfahren

Im Laufe der Zeit haben sich 3 Allokationsverfahren herausgebildet:

	Zusammenhängend	Verkettet	Indiziert
Vorallokation?	erforderlich	möglich	möglich
Blockgröße	groß	klein	klein
Allokations- häufigkeit	einmal	selten bis häufig	häufig
Größe der FAT	ein Eintrag	ein Eintrag	umfangreich

# zusammenhängende Allokation

Fakten:

- gesamte Speicher wird in Blöcken angefordert
- gut für sequenzielle Dateien und direkten Zugriff
- Externe Fragmentierung
  - freie Blöcke zwischen den Dateien lassen sich nicht mehr belegen
  - Speicherverdichtung

# verketteten Allokation

## Fakten:

- jeder Block enthält einen Zeiger auf dem nachfolgenden Block
- Startblock enthält Länge der Datei in Blöcken und Zeiger auf ersten Datenblock
- Datei über die gesamte Festplatte verstreut  $\Rightarrow$  Datenzusammenlegung
- keine direkten Zugriffe auf Dateien möglich

# indizierten Allokation

Fakten:

- mehrstufige Beschreibung wo die Datei gespeichert
- FAT hat einen Eintrag, wo Index-Block ist
- Index-Block stehen sind alle Zeiger auf Dateiblöcke
- direkter Zugriff immer möglich
- sequenzieller nach Datenzusammenlegung
- Verbesserung durch bilden von Clustern

Die indizierte Allokation ist die verbreitetste Form der Dateiallokation.

# Cluster

## Fakten:

- aus Performance gründen werden in modernen BS Blöcke zu Clustern zusammengefasst
- Clustergröße ist im allgemeinen eine Zweierpotenz (2048, 4096 usw.)
- Verlust von Speicher
- sehr ungünstig, wenn viele Dateien viel kleiner als Clustergröße

# Dateiverzeichnis 1/2

	<b>Basisinformationen</b>
Dateiname Dateityp Organisation	Dateiname, innerhalb Verzeichnisses eindeutig z.B.: Textdatei, ausführbare Datei usw. Organisationsart
	<b>Adressinformationen</b>
Datenträger Startadresse Genutzte Größe zugeteilte Größe	Gerät auf dem die Datei gespeichert ist physikalische Startadresse auf der HD Größe der Datei in Byte, Worte, Blöcke o. Cluster max. Größe der Datei
	<b>Zugriffskontrollinformationen</b>
Besitzer Zugriffsrechte	Kontrolle über Datei, kann Zugriffsrechte setzen Wer darf mit der Datei was tun

## Dateiverzeichnis 2/2

	<b>Nutzungsinformationen</b>
Erstellungsdatum	Wann wurde die Datei zum ersten mal erstellt
Ersteller	Ist in der Regel der Besitzer aber nicht zwingend
letzter Lesezugriff	Wann wurde die Datei zum letzten mal gelesen
letzter Leser	Wer hat die Datei zuletzt gelesen
letzte Änderung	Wann wurde die Datei zuletzt geändert
Wer änderte	Wer hat die Änderung durchgeführt
Sicherungskopie	Datum von letzte Sicherungskopie
Nutzung	Aktuelle Aktivität der Datei: Prozesse, die sie nutzen; Ob sie im Hauptspeicher aktualisiert worden ist und auf der HD noch nicht

## Zugriffsrechte 1/2

- **Keine** Der Benutzer erfährt nicht von der Existenz einer Datei, das betreffende Verzeichnis darf man in der Regel nicht öffnen
- **Kenntnis** Man erfährt, dass eine bestimmte Datei existiert. Man kann die Attribute der Datei lesen aber man darf sie nicht öffnen oder gar verändern.
- **Ausführen** Man kann ein Programm ausführen oder ein Verzeichnis öffnen. Man kann das Programm aber nicht kopieren.
- **Lesen** Man kann den Inhalt einer Datei lesen. Es gibt aber Implementationen, die unterscheiden zwischen lesen und kopieren. In den meisten Systemen kann man auch eine Datei kopieren, wenn man sie lesen darf.

## Zugriffsrechte 2/2

- **Anhängen** Man darf die Datei durch anfügen von Daten vergrößern. Es ist aber untersagt Daten zu löschen und zu verändern.
- **Aktualisieren** Man kann den Inhalt einer Datei beliebig verändern. Die Datei darf man aber nicht löschen. Es gibt auch Systeme, die zwischen den unterschiedlichen Graden von aktualisieren unterscheiden, so ist es evtl. gut wenn man nur Daten modifizieren darf, d.h. ich ersetze Inhalte der Datei durch neue.
- **Schutz ändern** Dieses Privileg hat in den meisten Systemen der Besitzer der Datei.
- **Löschen** Man kann die Datei aus dem Dateisystem entfernen

## Bsp. zu Zugriffsrechten

### Linux/UNIX

- Jede Datei hat 9 Schutzbits
- read/write/execute für Besitzer, seine Gruppe, Rest der Welt

```
-rw-r--r- 1 michael users 21554 Jan 2 18:30 Vortrag.tex
```

# Quellen

- William Stallings; Betriebssysteme, Prinzipien und Umsetzungen
- Michael Kofler; Linux, Installation, Konfiguration, Anwendung
- Niels Happel; Linux Software-RAID HOWTO; e-book
- [www.storitback.de](http://www.storitback.de)
- [www.icp-vortex.com](http://www.icp-vortex.com)