

Multimedia-Datenbanken

Kapitel 9: Multimedia-Basissysteme

Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Institut für Informatik
Lehrstuhl für Informatik 6 (Datenbanksysteme)

Prof. Dr. Klaus Meyer-Wegener

Wintersemester 2002 / 2003

Technische Universität Kaiserslautern
Fachbereich Informatik
AG Datenbanken und Informationssysteme

Dr. Ulrich Marder

Wintersemester 2003 / 2004

9. Multimedia-Basissysteme

□ Speicher-Server (Media-Server):

- dateibasiert!
zunächst sogar: gerätebasiert ...
- Video on Demand (VoD)

□ Kommunikation

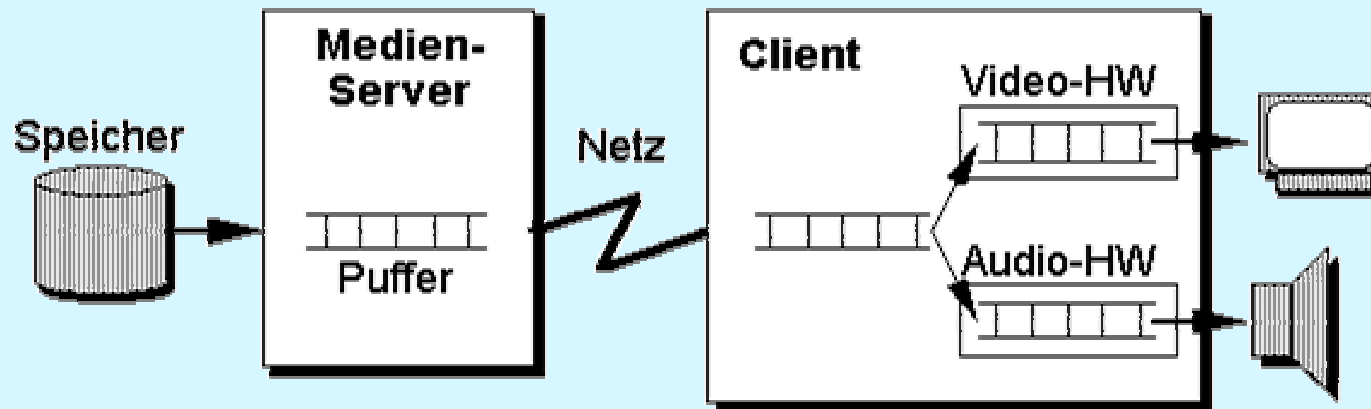
- Austauschformate

9.1 Multimedia-Speicher-Server

[Gemm95a]

zugänglich für die Clients über Hochgeschwindigkeitsnetz
(s. unten Abschnitt 9.2)

□ Architektur:



Multimedia-Speicher-Server (2)

□ primär:

- Wiedergabe (Playback) von Multimedia-Objekten in Echtzeit

□ interaktiver Zugriff:

- Stop, Pause, Resume, evtl. sogar Vor- und Zurückspulen

□ Herausforderung:

- **kontinuierliche** Medien
- Folge von "Medien-Quanten"
 - Audio-Messwerten, Video-Bildern
- zeitbehaftet

Multimedia-Speicher-Server (3)

❑ **Charakteristiken:**

- Speicherung und Wiedergabe in Echtzeit, ggf. sogar synchronisiert
- hohe Datenrate und großer Speicherplatzbedarf

❑ **kritische Komponenten daher Server und Netz**

❑ **hier: Server**

❑ **Aufnahme- und Abspielvorgänge**

❑ **Speicherorganisation (Platzierung)**

❑ **Multimedia-Dateisystem**

9.1.1 Aufnahme- und Abspielvorgänge

□ Aufnahme und Wiedergabe

- Pufferüberlauf bei *Aufnahme* bedeutet Verlust, bei *Wiedergabe* Störung oder ebenfalls Verlust
- beide Operationen äquivalent in Bezug auf Anforderungen an das System
- hier nur Wiedergabe

□ zunächst ein Strom (single-stream playback)

- Folge von periodischen Tasks mit Terminen (deadlines)
 - **Task**: Block von Platte lesen
 - **Termin**: geplante Wiedergabe-Zeit
- "just in time" nicht realistisch → **Puffer**

□ drei Aufgaben:

- "Verhungern" (starvation) des Wiedergabe-Prozesses verhindern
- benötigte Puffergröße minimieren
- Latenzzeit (zum anfänglichen Füllen des Puffers) minimieren

□ **Plattentransferraten (10 bis 50 MB/s)**

- deutlich höher als Datenrate eines einzelnen Stroms
 - 0,42 MB/s für MPEG-2; 0,2 MB/s für unkomprimiertes CD-Audio
- selbst kleiner Puffer ermöglicht Wiedergabe mehrerer Ströme

□ **je einen Lesekopf pro Strom**

- einfachster Ansatz
- beschränkt die Zahl gleichzeitiger Ströme durch die Zahl der Köpfe
- nicht optimal:
 - Transferraten reichen für mehrere Ströme

□ **also mehrere Ströme über einen Kopf**

- dann aber sorgfältige Planung der Zugriffe (scheduling) erforderlich
- und Zahl der Ströme begrenzen, für die das möglich ist

Platten-Zuteilung (disk scheduling)

□ Warteschlange

- Abarbeitung einer Reihe von Aufträgen für ein Gerät
- traditionell:
 - first come, first served (FCFS)
 - shortest seek time first
 - Scan
- reduzieren: *Armbewegung* (seek time) und *Umdrehungswartezeit* (rotational latency)
- Ziele: Durchsatz maximieren, faire Zuteilung
- keine Berücksichtigung von Echtzeit

□ earliest deadline first (EDF)

- bekanntester Algorithmus dafür
- zuerst den Blockzugriff ausführen, dessen Termin am nächsten liegt
- Nachteile:
 - lange Positionierungs- und Umdrehungswartezeiten
 - geringe Ressourcen-Auslastung

Platten-Zuteilung (2)

□ Variante: Scan-EDF

- Scan: Armbewegung abwechselnd nach innen und nach außen, dabei Abarbeitung aller Aufträge, deren Adressen "auf dem Weg liegen" – kann Positionierungszeiten erheblich reduzieren
- jetzt mit EDF kombinieren:
zunächst EDF, bei Aufträgen mit gleichem Termin dann Scan
- wirkt nur, wenn das auch vorkommt!
etwa bei Near-Video-on-Demand: Ströme erst sammeln und dann gemeinsam ausführen

□ Runden (rounds)

- alle anderen Algorithmen fassen dagegen Aufträge zusammen und arbeiten sie in Runden ab
- in jeder Runde für jeden Strom Folge von Blöcken beliebiger Länge (auch null) holen
- kommt der periodischen Natur der kontinuierlichen Ströme entgegen

Platten-Zuteilung (3)

□ in jeder Runde aber immer noch Zuteilung erforderlich

- **Round Robin** – reihum in fester Reihenfolge
 - am einfachsten
 - ignoriert wie EDF relative Lage der Blöcke, erfordert daher eigentlich Platzierung (s. unten)
- **Scan**
 - auch einfach, reduziert Armbewegung
 - neues Problem: Abstand zwischen aufeinanderfolgenden Bedienungen eines Stroms – u. U. einmal am Anfang einer Runde und dann am Ende ...
 - bedeutet auch zu Beginn mehr Wartezeit (Latenz): erst am Ende der zweiten Runde mit Wiedergabe aus dem Puffer beginnen; Puffer muss zudem fast zwei Runden groß sein
 - zugleich sind Runden kürzer ...

Grouped Sweeping Scheme (GSS)

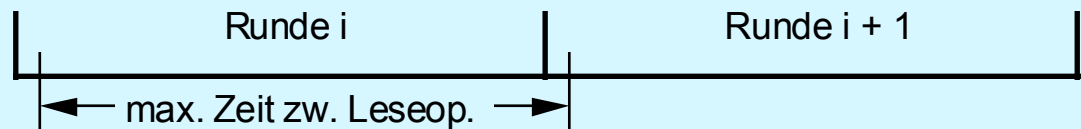
□ Partitionierung der Runden in Gruppen

- in fester Reihenfolge abarbeiten
- innerhalb der Gruppen dann Scan

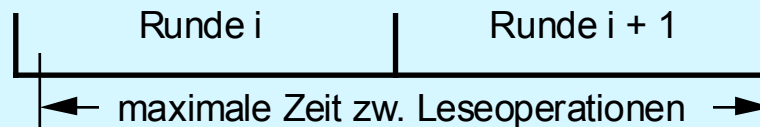
□ optimale Gruppengröße suchen

- Ausgleich zwischen Umfang einer Runde und Abstand der Bedienungen eines Stroms

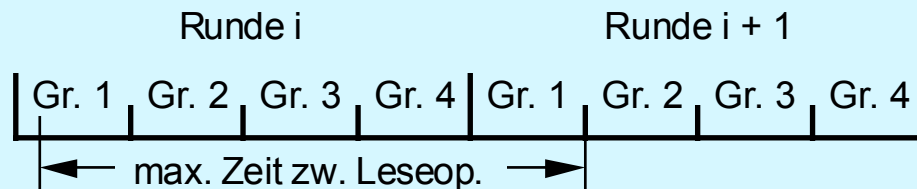
Round Robin



Scan



Grouped Sweeping



Anforderungen an Lesen und Puffern

□ Ziel:

- Anlieferung = Verbrauch in jeder Runde, Lieferung hinkt nie hinterher, und Umfang der gepufferten Daten verringert sich nie

□ Puffererhaltung:

- Algorithmen mit dieser Eigenschaft heißen "work-ahead-augmenting" oder "buffer-conserving" (**puffererhaltend**)
- auch Algorithmen denkbar, die diese Eigenschaft nicht aufweisen: fallen zeitweise zurück und müssen dann wieder aufholen – komplexer!
- Puffererhaltung nicht notwendig, aber hinreichend, um Verhungern auszuschließen
- vorab genug Daten lesen (Prefetch), um längstmögliche Runde abdecken zu können, und dann immer Puffer erhalten
- Anzahl der pro Runde nachzuladenden Blöcke:
maximaler Umfang der Runde muss bekannt sein – hängt wiederum von der Zahl der in jedem Strom zu lesenden Blöcke ab
nicht zu viele lesen, also nicht für alle Ströme die gleiche (maximale) Zahl – proportional zur Verbrauchsrate eines Stroms

Pufferverwaltung für maximalen Verbrauch

□ first in first out (FIFO)

- Puffer muss ausreichend Platz für nächste Leseoperation bieten
- am geeignetsten: FIFO
- bei Round Robin: Puffergröße = maximaler Verbrauch, FIFO füllt immer wieder auf ("topping-up")
- bei Scan: doppelte Größe
- evtl. noch Tricks erforderlich, wenn verbrauchte Einheiten nicht auf Blockgrenze – s. Literatur

Admission Control (Zulassungssteuerung)

- ❑ **kann neuer Strom zugelassen werden, ohne die schon laufenden zu beeinträchtigen?**
 - bisherige Annahme: alle Termine müssen gehalten werden
 - einige Anwendungen können verpasste Termine tolerieren
 - wenige fehlende Bilder bei Video oder Knacken bei Audio
 - vor allem dann, wenn mit geringeren Kosten verbunden!
 - zur Einhaltung aller Termine muss Server **Worst-case-Annahmen** zu Armpositionierungs- und Umdrehungswartezeit machen – die tatsächlichen dann allerdings meist kürzer
 - Raum für zusätzliche Ströme:
 - statistische Schwankung in Blockzugriffszeiten
 - und ggf. Kompressionsraten

Admission Control (2)

□ drei Dienstgüte-Kategorien für Ströme:

- **deterministisch:**

- alle Termine müssen gehalten werden
- Zulassungssteuerung muss Worst-case-Annahmen machen

- **statistisch:**

- Termine werden mit einer vorgegebenen Wahrscheinlichkeit gehalten
- Zulassungssteuerung muss statistisches Verhalten des Systems kennen und berücksichtigen

- **Hintergrund:**

- keine Garantien für Einhaltung der Termine
- Zuteilung nur, wenn noch Zeit nach Bedienung der deterministischen und statistischen Ströme

Admission Control (3)

❑ **neuer deterministischer Strom:**

- Umfang der Runde verlängert sich
 - Puffer der schon laufenden Ströme ggf. vergrößern (dynamisch)
- alternativ
 - alle Puffer von vornherein auf maximale Rundenlänge ausrichten
 - dann bei Zulassung nur auf Erreichung des Maximums prüfen

❑ **neuer statistischer Strom:**

- analog, aber mit Mittelwerten

❑ **in einer Runde dann**

- immer erst deterministische, dann statistische, dann Hintergrund-Ströme bedienen
- verpasste Termine gleichmäßig verteilen

Behandlung von Dienstgarantien und Terminen

- ❑ **variable Kompressionsrate:**
 - ein Block ergibt variable Menge von Medien-Quanten
 - Anzahl der zu lesenden Blöcke variiert mit Kompressionsrate
- ❑ **auch hier:**
 - Worst case für deterministische Ströme, Mittelwerte für statistische
- ❑ **weitere Möglichkeit:**
 - Raten im voraus ermitteln und abspeichern – beides genauer
- ❑ **Umgang mit verpassten Terminen (bei statistischen und Hintergrund-Strömen):**
 - entweder nachholen (verlängert Dauer der Wiedergabe)
 - oder überspringen
- ❑ **grundlegend andere Technik:**
 - Variation der Auflösung eines speziellen Mediums, z. B. nur die höherwertigen Bits bei Audio übertragen, skalierbare Kompression (Teilmenge direkt verwendbar) – entspricht schnellem Vorlauf

9.1.2 Speicherorganisation

- optimale Platzierung von Blöcken auf den Platten
- Verwendung mehrerer Platten
- Hinzunahme von Tertiärspeicher zur Erhöhung der Kapazität
- Aufbau von Speicherhierarchien

□ Platzierung / Allokation von Blöcken

- **zusammenhängend**
 - einfach zu implementieren, aber anfällig für Fragmentierung
 - enormer Aufwand (Kopieren), wenn Zusammenhang auch über Einfügungen und Löschen hinweg erhalten wird
 - nur für read-only server (Video on Demand)
- **verstreut**
 - Hauptproblem: sog. intrafile seeks, Armpositionierung innerhalb der Leseoperationen eines Stroms in einer Runde
 - vermeidbar durch geeignet große Blöcke

Speicherorganisation (2)

alternativ:

wenn mehr als ein Block gelesen werden muss, Armpositionierung auf erträgliches Maß reduzieren

- **beschränkte Platzierung** (constraint placement)
 - Obergrenze für Distanz zwischen aufeinanderfolgenden Blöcken
(nicht nur für Paare, sondern als Mittelwert für endliche Folge von Blöcken)
 - besonders interessant bei kleineren Blockgrößen
(z. B. in einem Dateisystem, das für Text ausgelegt wurde)
 - Algorithmen zur Gewährleistung recht aufwändig
 - Effekt auch nur, wenn wirklich alle Blöcke einer Runde sequentiell gelesen –
Scan kann empfindlich stören!

Bildung von Streifen und Verzahnungen

"data striping and interleaving"

□ Datei auf einer Platte:

- Durchsatz dieses Geräts beschränkt Anzahl der Ströme
→ über mehrere Platten verteilen

□ RAID (s. Vorl. Betriebssysteme):

- logische Blöcke in "**Streifen**" über alle Platten gezogen, physischen Block 1 aller Stapel gleichzeitig lesen als logischen Block 1 usw.
- Laufwerke streng synchron, Zugriff auf logischen Block dauert so lange wie Zugriff auf physischen eines Stapels
- Durchsatz erhöht um Anzahl der Laufwerke
- Armpositionierung und Umdrehungswartezeit bleiben gleich

Bildung von Streifen und Verzahnungen (2)

□ **Verzahnung:**

- aufeinanderfolgende Blöcke einer Datei auf verschiedenen Platten, oft einfach reihum
- Platten nicht mehr synchron

□ **zwei Zugriffsarten:**

- in jeder Runde je einen Block *von jeder Platte* (wie bei den Streifen)
 - gute Lastbalancierung, aber größere Puffer
- in jeder Runde einen Block *von einer Platte*
 - in N Runden also von N Platten lesen
 - Lastbalancierung verlangt gezielten Versatz

□ **Streifen und Verzahnung auch kombinierbar**

Tertiärspeicher und Speicherhierarchie

❑ Magnetplatten:

- für große (Video-) Server zu teuer
- Tertiärspeicher: Magnetbänder, optische Platten, Wechselautomaten
- allerdings zu langsam für direkte Wiedergabe
→ Magnetplatten als Cache verwenden

❑ Ansatz:

- *Anfangssegmente* der Medienobjekte auf Platte, reduziert Latenz
- alternativ: vor dem Abspielen *ganz* auf Platte laden, hohe Latenz, aber nicht kritisch bei Anwendungen mit wenigen populären Objekten – meist schon im Cache
- Zugriffsmuster oft auch vorhersehbar, z. B. in Lernumgebungen
- auch mehrere Caches im Netz, Konsistenz kein Problem, da read-only

❑ Berkeley Distributed VoD System:

- Archive-Server mit Tertiärspeicher, Metadaten und Anfragesprache für Suche und Planung der Wiedergabe

9.1.3 Multimedia-Dateisysteme

- Client/Server-Interaktion
- Auffinden von Daten in Dateistrukturen
- Erzeugen, Bearbeiten und Wiedergewinnen von Multimedia-Objekten

□ Client-Schnittstelle

- **dateisystem-orientiert (Client-Pull):**
 - MM-Objekt = Datei, Operationen entsprechend: open, close, read
 - Client setzt periodisch read-Operationen ab; "pause" und "resume" einfach durch Aussetzen und Wiederaufnehmen
 - Server macht in Ausführung von open Zulassungssteuerung und Prefetch, auch periodisch weitere Prefetches
- **strom-orientiert (Server-Push):**
 - Client benutzt Operationen wie play, pause, resume
 - Server sendet nach Initialisierung periodisch Daten an den Client – ohne read-Aufrufe

- Zuordnung der Blöcke zu Dateien
- Unterstützung des sequenziellen Zugriffs auf Blöcke

□ **Katalogstrukturen**

- verkettete Liste aller Blöcke
- file allocation table (FAT) wie in DOS: Zeiger auf jeden Block
- Index pro bzw. in Datei (UNIX)
- hybrid:
 - Verkettung *und* Index – unterstützt sequenziellen wie wahlfreien Zugriff
 - hoher Wartungsaufwand in Systemen mit wenigen Änderungen (VoD) akzeptabel
 - evtl. zusätzliche Verkettungen für Vor- und Zurückspulen

□ **denkbar:**

- *multi*-mediale Dateien, mit getrennter Behandlung der medialen Teile und Synchronisationsinformation

Bearbeiten von Multimedia-Objekten

□ Ziel:

- Kopieren ganzer Objekte möglichst vermeiden
- Objekte als *unveränderlich* erklären und Änderungen auf Umsetzen von Zeigern beschränken

□ garbage collection

- Problem: bei kleinen Einfügungen und Löschungen auch Zeiger noch zu aufwändig (für Wiedergabe)
- Aufwand sollte der Größe der Änderung entsprechen

□ Vorschlag:

- minimaler Füllgrad für Blöcke (so dass kontinuierliche Wiedergabe möglich)
- Einfügen/Löschen betrifft dann bestimmte Anzahl voller Blöcke plus einen nur teilweise gefüllten
- werden in FAT eingefügt/aus ihr gelöscht
- dann Daten zwischen benachbarten Blöcken umverteilen, bis Füllgrad erreicht

Interaktive Steuerungsfunktionen

- pause/resume, fast forward, fast backward

❑ **pause/resume:**

- kann Pufferverwaltung stören,
z. B. bei gemeinsamer Nutzung eines Stroms durch mehrere Zuschauer

❑ **Spulen kann erfolgen als**

- Abspielen mit höherer Geschwindigkeit
(Datenrate u. U. zu hoch) oder
- als normale Wiedergabe mit Übergehen von Daten
(muss sich mit Komprimierung vertragen – evtl. nur Differenzen)

❑ **Vorspulen mit Übergehen:**

- eigene, hochkomprimierte Datei (z. B. MPEG D-Rahmen);
 - keine zusätzlichen Speicherungsverfahren, aber Speicherplatzbedarf und schlechte Qualität der Wiedergabe
- Blöcke als relevant oder irrelevant kennzeichnen
- skalierbare Kompressionsverfahren:
 - Aufwand für Aufteilen und Rekombinieren der Blöcke
 - relevante Blöcke sind gerade die größeren – Datenrate höher als normal

❑ **MM-Speicher-Server**

- grundlegend anders als normale Datei-Server
- Echtzeit, deutlich höherer Ressourcen-Bedarf

❑ **verfügbare Produkte z. B.:**

- IBM LANServer Ultimedia:
 - Video und Audio für entsprechend ausgerüstete PCs
- Oracle Media Server:
 - Video on Demand, angeblich 25.000 Videoströme

❑ **Dienste**

- Dienste von Betriebssystemen und Rechnernetzen müssen weiter ausgebaut werden

❑ **Dateisystem**

- muss Dateitypen unterscheiden, teilweise auch Inhalte kennen, und dann geeignete Platzierung, Treiber, Scheduling usw. vornehmen

9.2 Multimedia-Kommunikation

[Krön88a; Khos96a, p. 505ff., ch. 11 Multimedia Networking]

□ **Kommunikationsarten:**

- direkte Kommunikation:
 - Gespräch, Telefon, Konferenzschaltung
 - räumliche Distanz, aber keine zeitliche
- indirekte Kommunikation:
 - Post
 - räumliche und zeitliche Distanz
 - braucht einen dauerhaften Nachrichtenträger: Dokument
 - hier: elektronische Dokumente

□ **Austausch zwischen verschiedenartigen Systemen**

□ **Ziel: Dokument beim Empfänger**

- darstellen (drucken)
- Layout verändern (z. B. A4 → US Letter)
- editieren, Inhalt bearbeiten

→ **Austauschformate**

Multimedia-Kommunikation (2)

□ warum interessant für Multimedia-DBS?

- neutrale, vollständige **Datenstrukturen** (Daten-Schemata), Informationen sehr vieler (aller?) Anwendungen darstellbar
- jedoch ohne Operationen, daher kein Datenmodell

□ These:

- Austausch von Dokumenten kann prinzipiell ersetzt werden durch Zugriff auf gemeinsame Datenbasis
- hinzukommen muss: Aufmerksammachen des Empfängers
 - Definition eines Eingangskorbs
 - Zeitverwaltung im DBS ("alle neuen Daten seit heute morgen, 8 Uhr")
 - Alerter: Signal oder Ausgabe von Daten bei Zutreffen einer Bedingung