Cooperative I/O
A Novel I/O Semantics for Energy-Aware Applications

Andreas Weißel • Björn Beutel • Frank Bellosa

Department of Computer Science 4 (Operating Systems)
University of Erlangen
Martensstr. 1, 91058 Erlangen, Germany
{weissel,bnbeutel,bellosa}@cs.fau.de

Übersicht

- Motivation
- Betriebsmodi & Energieverbrauch von Festplatten
- Cooperative I/O
- Validierung und Tests
- Fazit

Warum Festplatten?

- Speicherkapazität bei mobilen Geräten immer wichtiger (Digitalkameras, PDAs, ...)
- Signifikanter Anteil am Gesamtenergieverbrauch

	Aktiv	Standby
Notebook	15 W	1000 mW
IBM Travelstar	2.1 W	100 mW
PDA (ohne Festplatte)	3.5 W	200 mW
IBM Microdrive	0.8 W	70 mW

- Festplatten bieten bereits Stromsparmodi.
- Bisher:
  - Priorität auf Performance (Latenzzeit, Übertragungsrate)
  - Mechanismen zum Stromsparen vom Betriebssystem nicht ausgeschöpft

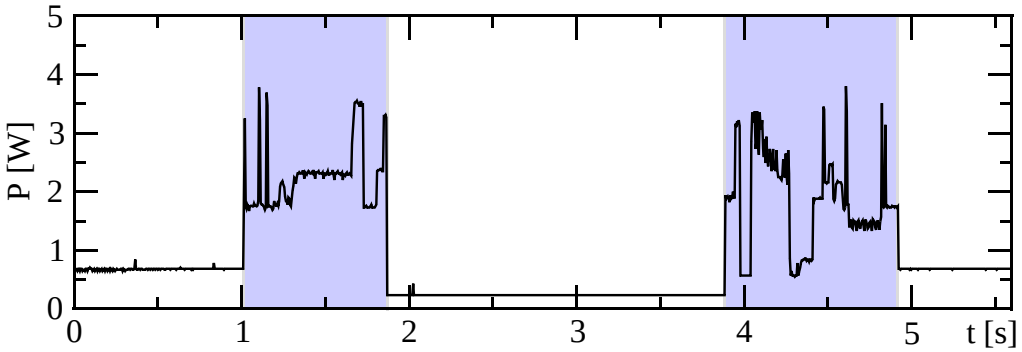
Betriebsmodi von Festplatten

- Die heutige Festplattengeneration unterstützt mehrere Betriebsmodi mit niedriger Energieaufnahme (“Stromsparmodi”).
- Erkauft wird die gesparte Energie mit Zugriffsverzögerungen (je nach Typ und Modus zwischen 20ms und 10s).
- Typische Betriebsmodi einer IBM Travelstar 15GN:

Modus	Eigenschaften	Energiever-brauch	Verzögerung bei Zugriff
Aktiv	Lese-, Schreibvorgang, Moduswechsel	2.1–4.7 W	—
Idle **Low-Power Idle**	 Köpfe geparkt	1.85 W 0.65–0.85 W	— 20–300 ms
Standby	Laufwerksmotor aus	0.25 W	1.0–9.5 s
Sleep	fast gesamte Elektronik aus	0.1 W	3.0–9.5 s

Umschalten zwischen Betriebsmodi

- Moduswechsel verursachen “Kosten” (Energie- und Zeitaufwand):
  - Parken und Positionieren der Köpfe
  - Stoppen bzw. Wiederanfahren des Laufwerksmotors



- Die *break-even* Zeitspanne ist der zeitliche Abstand zwischen zwei Zugriffen, für den gilt:

  Energieverbrauch durch Wechsel in den Standby-Modus und zurück = Energieverbrauch im Idle-Modus

  Travelstar: break-even time = 8.7 s

Umschalten zwischen Betriebsmodi (2)

- Bei kurzen Zeitabschnitten im Idle-Modus (< break-even time) lohnt es sich nicht, einen Stromsparmodus zu aktivieren.
- Im Voraus ist nicht bekannt, wann der nächste Zugriff stattfindet bzw. ob ein Moduswechsel Energie sparen würde.
- Traditioneller Ansatz:
  - Protokollieren von Zugriffen auf die Festplatte
  - Versuch, anhand der Zugriffsmuster den Zeitpunkt zukünftiger I/O-Operationen vorherzusagen
  - Verfahren mit festem Timeout, adaptive Verfahren, stochastische Algorithmen, ...

Übersicht

- Motivation
- Betriebsmodi & Energieverbrauch von Festplatten
- Cooperative I/O
- Validierung und Tests
- Fazit

Cooperative I/O

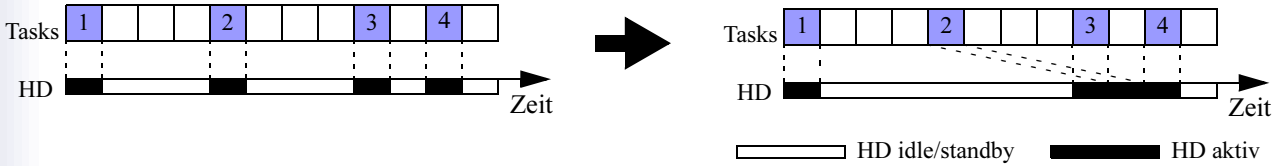
- Grundanahme bisheriger Ansätze:
  - Zeitpunkte zukünftiger Festplattenzugriffe sind unbekannt und können vom Betriebssystem nicht beeinflusst werden.
- Cooperative I/O: Das Timing von Festplattenzugriffen wird nicht mehr als gegeben betrachtet.
  - ➜Dem Benutzer wird die Möglichkeit gegeben, Lese- und Schreiboperationen als *verzögerbar* und/oder *abbrechbar* zu definieren.
  - ➜Betriebssystem kann Zeitpunkt der Gerätezugriffe selbst festlegen.
- Neue Ein-/Ausgabebefehle (zusätzlich zum alten Interface)
  - `read_coop(int fd, void *buf, size_t count, int time-out, int abort);`
  - `write_coop(int fd, void *buf, size_t count, int time-out, int abort);`
  - `open_coop(const char *pathname, int flags, int time-out, int abort);`

Cooperative I/O (2)

- Festplatte im aktiven oder Idle-Modus:
  - Verzögerbare Operationen werden sofort bedient.
- Festplatte im Standby-Modus:
  - Operationen werden verzögert, bis Festplatte durch Zugriff eines anderen Prozesses aktiviert wird *oder*
  - bis benutzerdefinierter Timeout erreicht ist.
  - Dann: Aktivieren der Festplatte erzwingen oder Abbrechen der Operation.
- Beispiele:
  - Audio-/Videoplayer
  - Web Browser
  - Hintergrundprozesse
  - Auto-Save

Cooperative I/O (3)

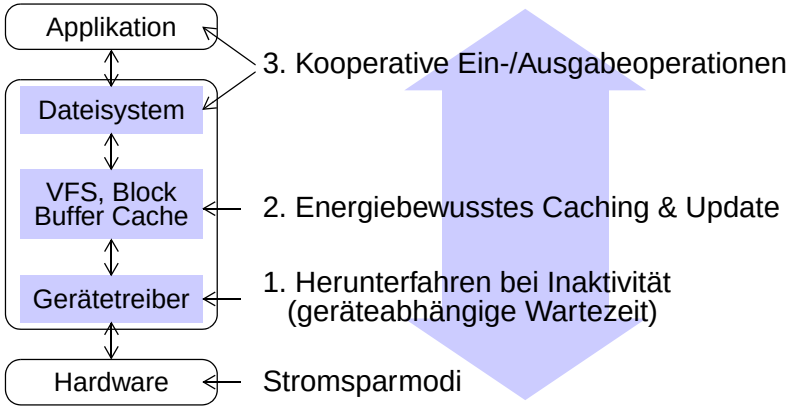
- Zusammenfassen (zeitliches Gruppieren) von Zugriffen auf das Laufwerk



- Besser wenige lange “inaktive” Zeitabschnitte als viele kurze
  - ➜Festplatte für längere Zeitabschnitte nicht aktiv
  - ➜Kann öfters in den Standby-Modus versetzt werden
  - ➜Weniger Moduswechsel
  - ➜Reduzierung des Energieverbrauchs

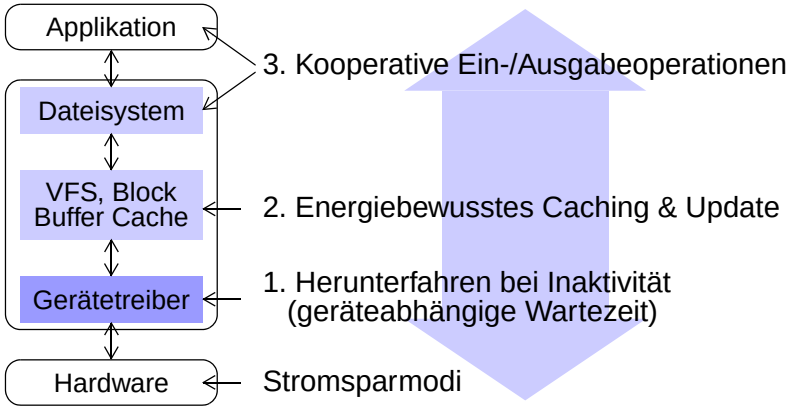
Konzept & Implementierung

- Einbindung aller Schichten - von der Hardware zur Anwendung



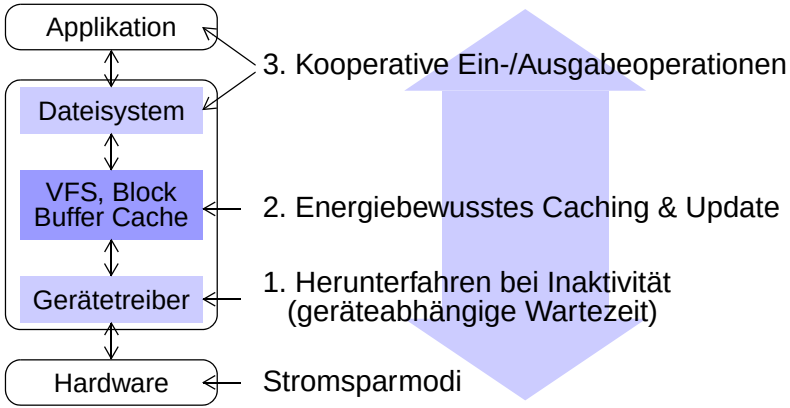
- Implementierung
  - Linux Kernel 2.4.19
  - Modifikationen am IDE-Treiber, VFS (Buffercache und Update-Mechanismus) und Ext2-Dateisystem

Konzept & Implementierung (2)



1. Wechsel in den Standby-Modus bei Inaktivität
   - einfacher, effektiver und bewährter Algorithmus: Device-Dependent Time-out [Lu, Micheli 2001]
   - Laufwerk herunterfahren, falls Zeitpunkt des letzten Zugriffs länger zurückliegt als (geräteabhängige) break-even-Zeitspanne

Konzept & Implementierung (3)



2. Energiebewusstes Caching & Update-Verfahren

➜Ziel: Zusammenfassung von Zugriffen auf das Laufwerk

- Bei einem Update werden alle Blöcke auf Platte geschrieben, nicht nur die ältesten.
- Updates werden an andere Festplattenzugriffe “angehängt”.
- Soll das Laufwerk in den Standby-Modus versetzt werden, wird vorher ein Update erzwungen.

Übersicht

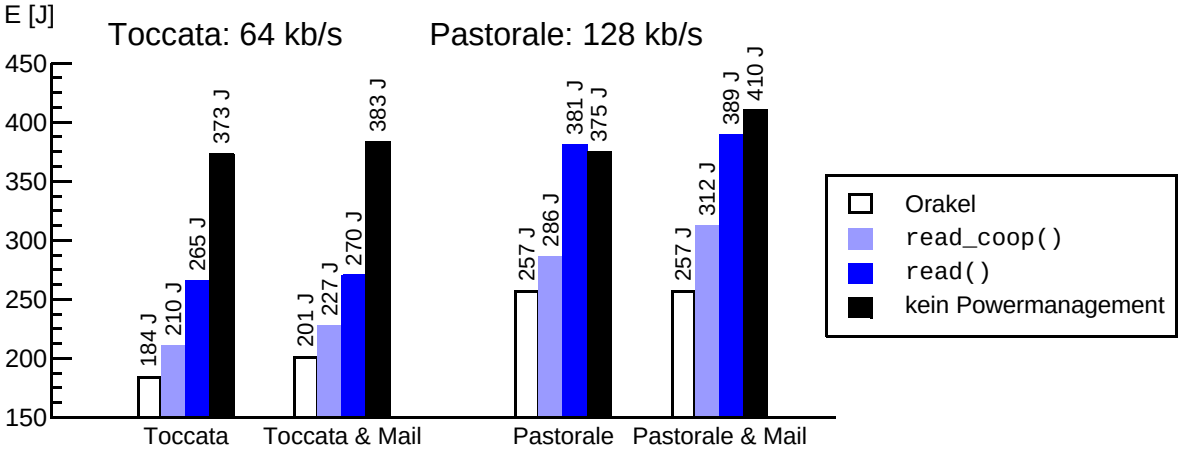
- Motivation
- Betriebsmodi & Energieverbrauch von Festplatten
- Cooperative I/O
- Validierung und Tests
- Fazit

Validierung und Tests

- Messumgebung
  - Messen des Spannungsabfalls an einem Widerstand in der 5V-Versorgung der Festplatte
  - Auflösung: 256 Stufen; 20000 Samples pro Sekunde
- MP3-Player AMP mit verzögerbaren, nicht-abbrechbaren Lese-Aufrufen
- Player besitzt zwei Datenpuffer
  - Musikdaten werden aus einem Puffer gelesen, während ein Thread den anderen Puffer mit kooperativen Leseaufrufen füllt.
  - ca. 150 Zeilen Änderungen am Programmcode
- Parallel dazu Mail-Reader, der alle 60s Mails abfragt und diese in Datei sichert (nicht kooperative Schreibaufrufe)

“Kooperativer” MP3-Player

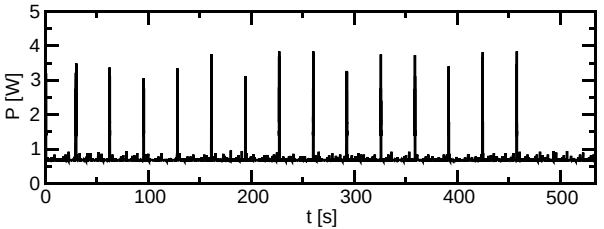
- Zwei MP3-Dateien mit unterschiedlicher Auflösung



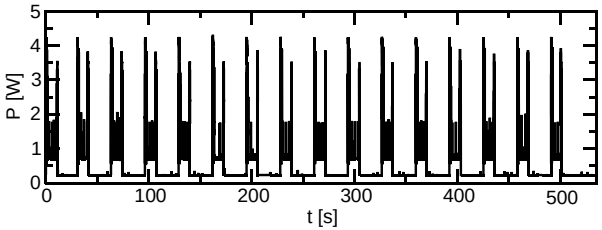
- Vergleich mit “Orakel”-Strategie
  - ◆ kennt Zeitpunkte zukünftiger Aufrufe
  - ◆ schaltet Laufwerk sofort in Standby-Modus, falls dadurch Energie-einsparungen möglich
  - ➜ Bezüglich des Energieverbrauchs optimale Strategie (ohne Timing der Aufrufe zu verändern)

“Kooperativer” MP3-Player (Toccata)

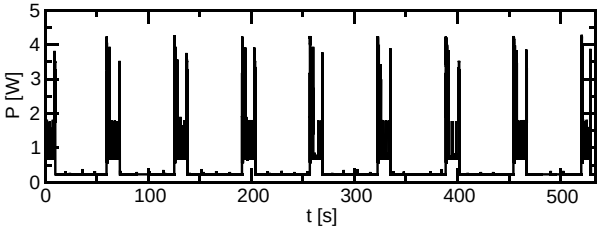
- Falls Laufwerk im Standby-Modus:
  - Kooperative Leseaufrufe werden solange verzögert, bis aktiver MP3-Lesepuffer leer.
  - Mit `read_coop()` wird der Puffer wieder gefüllt.
  - Laufwerk nun im Idle-Modus.
  - ➜ `read_coop()`-Aufruf für den anderen Puffer wird sofort ausgeführt.
  - ➜ Effektiv werden zwei Leseaufrufe zusammengefasst.



a) Kein Powermanagement (373 J)



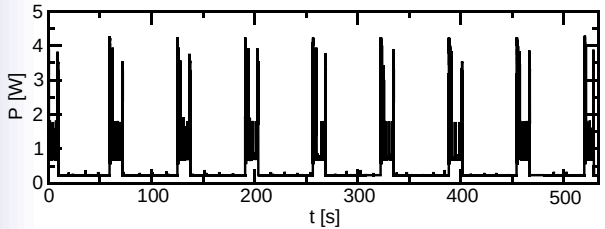
b) Nicht-kooperative Leseaufrufe (265 J)



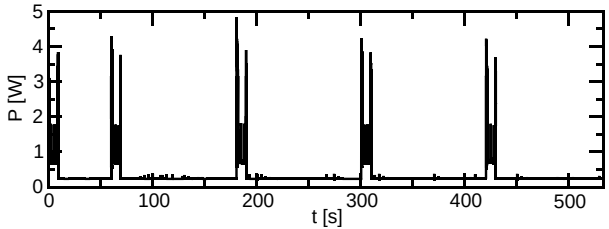
c) Kooperative Leseaufrufe (210 J)

“Kooperativer” MP3-Player + Mail-Reader

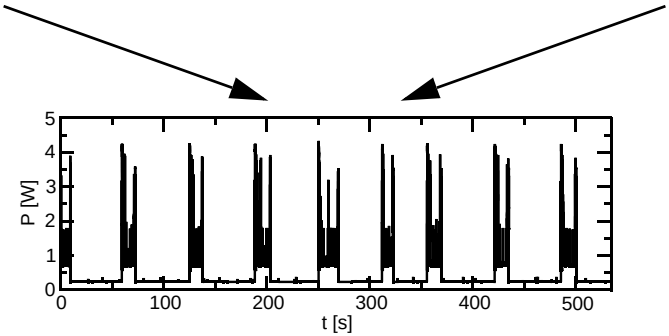
- Schreibzugriffe des Mail-Readers und Lesezugriffe des MP3-Players werden “verschmolzen”.



a) Toccata kooperativ (210 J)



b) Mail kooperativ (164 J)

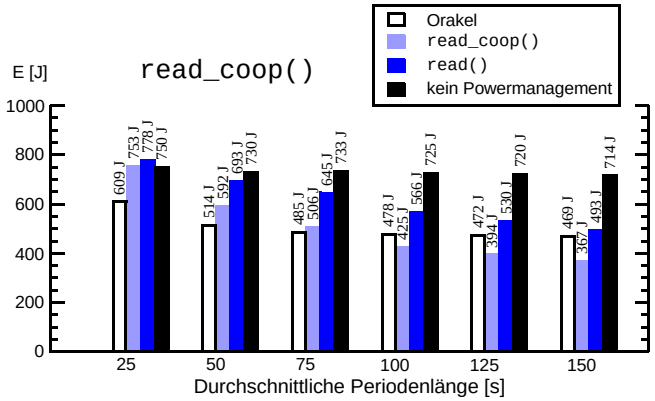


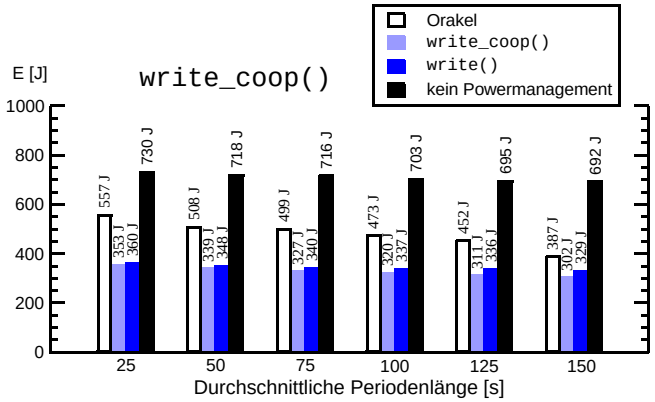
c) Toccata & Mail kooperativ (227 J)

Parametrisierte Tests

- Fünf Prozesse, die periodisch nach zufallsgesteuerten Wartezeiten kooperativ lesen bzw. schreiben.
- Cooperative I/O fasst Aufrufe zusammen.
  - ➜Reduzierung von Moduswechseln
  - ➜Mehr Zeit im Standby-Modus
  - ➜Energieersparnis gegenüber (nicht-kooperativem) Orakel

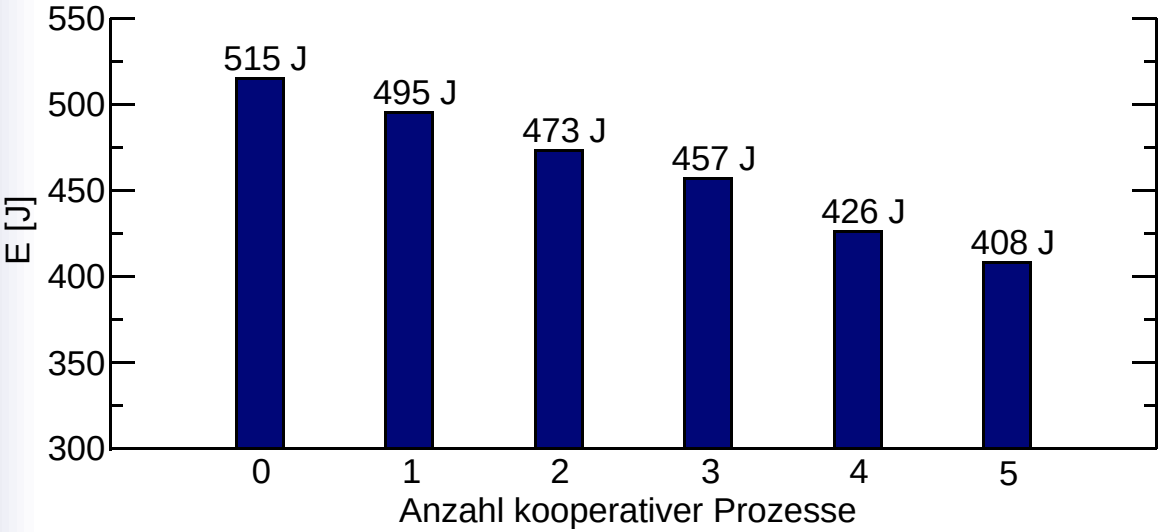
Strategie	Aktiv	Idle	Standby
Kooperativ	29 s	153 s	868 s
Orakel	107 s	132 s	811 s





Mehrere kooperative Prozesse

- Insgesamt 5 Prozesse, davon 0–5 kooperativ



- Energieersparnis steigt mit zunehmender Anzahl kooperativer Prozesse.

Fazit

- Geeignet für alle Gerätetypen mit rotierenden Medien
- Wrapper möglich für Alt-Anwendungen
  - konfigurierbare Abbildung der read/write-Schnittstelle auf die kooperativen Systemcalls
- Bremsen und Wiederanfahren des Laufwerkmotors und Parken der Köpfe eine der häufigsten Ursachen für Laufwerksversagen
  - früher max. 50.000 Moduswechsel
  - heute dank IBMs load/unload-Technologie: 300.000