

EMES: Eigenschaften mobiler und eingebetteter Systeme

Energieverbrauch und Mobile Systeme

Dipl. Inf. Jan Richling
Wintersemester 2005/2006



- Motivation (für diese VL)
 - Mobilität eines mobilen Systems hängt wesentlich davon ab, wie lange es unabhängig von Fremdenergie arbeiten kann
 - Moderne mobile Systeme (PDAs, Mobiltelefone) benötigen für eine Vielzahl neuer Funktionen Energie
 - Direkter Zusammenhang Energieverbrauch — Akkugröße — Gerätegröße — Kosten
 - Niedriger Ruhestrombedarf ist bei vielen Systemen entscheidend
 - Energieverbrauch und Techniken zur Reduzierung sind auch bei stationären Systemen wichtige Parameter
- Quellen:
 - Seminarvortrag von Manuel Klatt und Andre Wiesner, SoSe03
 - Forschungen von A. Weissel, B. Beutel, F. Bellosa, Universität Erlangen (Quelle für fast alle Abbildungen)

Energiequellen mobiler Geräte

- Batterien
 - Einweg (nicht wiederaufladbar)
 - Mehrweg (wieder aufladbar): Akkus, u.a.
 - * NiCD
 - * NiMH
 - * Blei / Bleigel
 - * Li-Ion
- Solarzellen
- Brennstoffzellen
- Energieerzeugung mit Generatoren, angetrieben von
 - Verbrennungsmotoren
 - Mikro-Motoren
 - Menschen (Beispiel: “FreeCharge”)
- ...



Energieverbrauch mobiler Geräte

- CPU
 - Maximalverbrauch
 - Durchschnittlicher Verbrauch
 - Ruheverbrauch (besonders wichtig!)
- Speicher
 - SRAM (kommt mit wenig Strom und ohne Takt aus)
 - DRAM (braucht Refresh und Takt)
 - Flash (hält Informationen auch ohne Strom)
- I/O-Komponenten
 - Festplatten
 - Grafikhardware (insbesondere Display-Beleuchtung)
 - Netzwerkschnittstellen (insbesondere drahtlos)
 - ...

Reduzierung des Energieverbrauchs

Verschiedene Ansatzpunkte:

- Hardware
 - Reduzierung des Verbrauchs bei gleicher (Rechen-)Leistung
 - Unterstützung von Stromsparmodi
 - Unterstützung für partielle Abschaltung
 - Asynchrone Strategien (Takt nur dort, wo benötigt)
- Software
 - Betriebssystem
 - * Verwaltung der Ressource “Energie”
 - * Strategien zum Umgang mit “Leistung vs. Energieverbrauch”
 - Anwendungen
 - * Energiebewusster Umgang mit I/O-Ressourcen

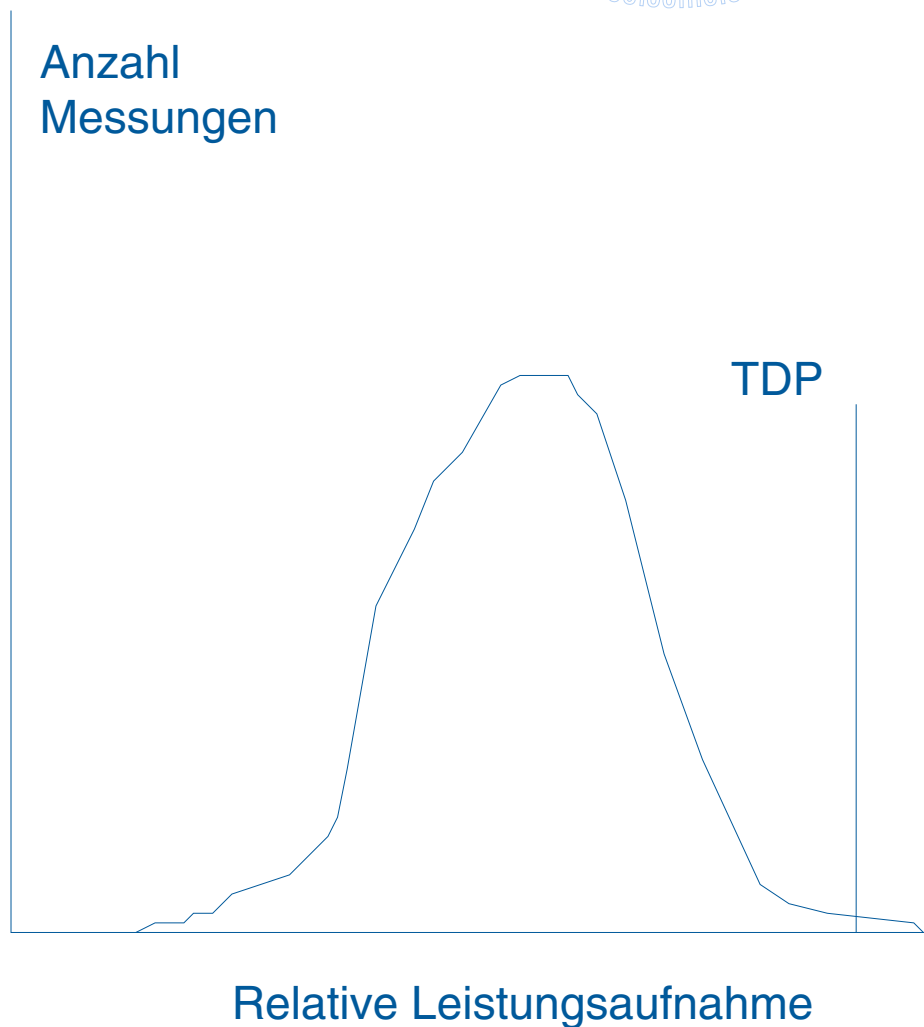
Reduzierung des Energieverbrauchs: Hardware (insb. CPU)

- Erfolgversprechendster Ansatz:
 - Problem wird dort behandelt, wo es entsteht
- Aufwendigster Ansatz
 - Im Nachhinein unmöglich
 - Berücksichtigung beim CPU-Design erforderlich
 - Teilweise unmöglich wegen physikalischer Probleme
- Ansatz:
 - Dynamische Regelung der Leistungsaufnahme
- Aktuelle Hauptprobleme, die hohen Energiebedarf bedingen:
 - Thermal Design Power
 - Kriechströme

Dynamische Regelung der Leistungsaufnahme

- Anpassung der Taktfrequenz und Versorgungsspannung
- Abschaltung von nicht benötigten Komponenten auf Chip- und Systemebene
- Verschiedene Stromsparmodi
- Intelligente Steuerung von I/O-Geräten
 - Anpassung von Sendeleistungen und Sendehäufigkeiten
 - Abschaltung von mechanischen Komponenten (Motoren)
- ...

Thermal Design Power



- Typische Leistungsaufnahme bei Vollast durch “normale” Anwendungen
- Im Normalfall nie erreicht
- Normal etwa ein Drittel
- Grundlage für die Auslegung von Stromversorgung und Kühlung
- Heutige Desktop-CPUs: 70...100 Watt

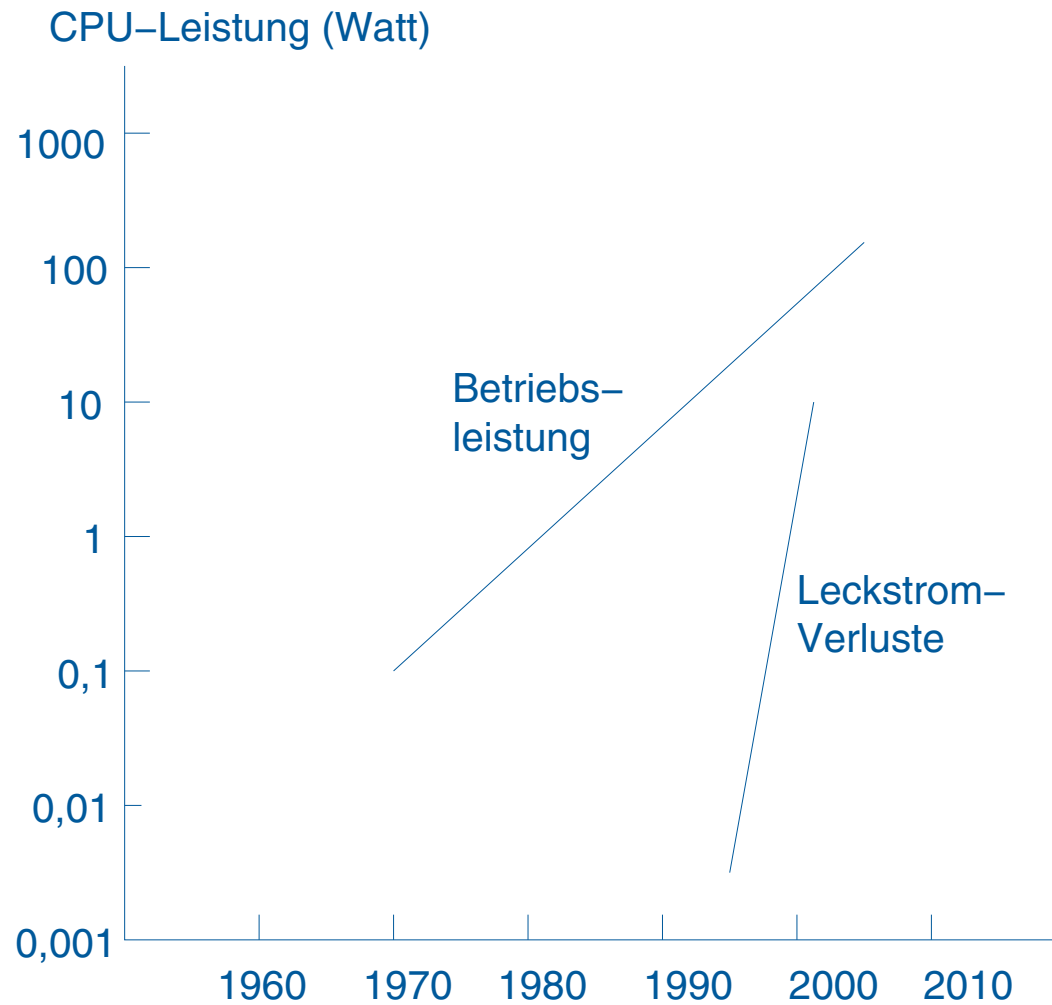


Kriechströme I

- Kriechströme sind die “statische” Leistungsaufnahme eines Chip
- Energieverbrauch durch Kriechströme findet immer statt, sobald der Chip mit Strom versorgt wird
- Ursachen:
 - Ströme trotz Sperrschaltung der Transistoren
 - Effekt verstärkt sich mit
 - * Sinkender Strukturgröße
 - * Steigender Temperatur
- Gegenmaßnahme:
 - Reduzierung der Betriebsspannung, um Stromfluß und damit den Energieverbrauch zu senken
Nebenwirkung: Unterscheidung 0 / 1 wird immer schwerer
 - Effektive Kühlung
 - Andere Herstellungstechniken (z.B. Silicon-on-Insulator)



Kriechströme II



Kriechströme und Prozessor-Leistungsaufnahme (Tendenzen)

Reduzierung des Energieverbrauchs: Software

- Energieverbrauch ist eine nicht-funktionale Eigenschaft
 - Muß auf allen Ebenen/Schichten des Systems behandelt werden
 - Energiesparmöglichkeiten der Hardware machen nur Sinn, wenn sie von der Software genutzt oder berücksichtigt werden
 - Ungünstige Nutzung kann Energieverbrauch sogar erhöhen!
- Ansätze auf verschiedenen Ebenen der Software:
 - Betriebssystem
 - * Verwaltung der Ressource “Energie”
 - * Beispiel: “Process Cruise Control” — Uni Erlangen
 - Anwendungen
 - * Koordination von energieintensiven Vorgängen wie Plattenzugriffe
 - * Beispiel: “Kooperatives I/O” — Uni Erlangen

Steuerung auf Ebene des Betriebssystems

- Beispiel: “Process Cruise Control”
- Möglichkeit der Einflußnahme:
 - Änderung der Taktfrequenz
 - In späteren Versionen: Änderung der Versorgungsspannung
- Idee:
 - Analyse des Verbrauchs in Abhängigkeit von Befehlsmustern
 - Berechnung von “Rechenleistung pro Energieeinheit” für verschiedene Szenarien und Taktfrequenzen
 - Auswahl der effektivsten Taktfrequenz mit Randbedingungen (z.B. minimal zulässige Rechenleistung)

Verbrauchsscharakteristika: Statisch vs. Dynamisch

- Prozessor
 - Statisch: Grundlast
 - Dynamisch: Abhängig von Befehlen und aktiven Einheiten
- Speicherverwaltungseinheit (MMU)
 - Dynamisch: Adressumrechnung
- Speicher:
 - SRAM:
 - * Statisch: Ruhestrom zur Erhaltung des Speicherinhalts
 - * Dynamisch: Schalten von Transistoren
 - DRAM:
 - * Statisch: Ruhestrom und Bedarf für Refreshs
 - * Dynamisch: Wie SRAM, dazu: Multiplexer für Adressierung
 - RDRAM: Mehrere Low-Power-States senken statischen Verbrauch

Verbrauchscharakteristika: Tasks und Auswirkungen

- Für die Steuerung des Verbrauchs durch das Betriebssystem ist zu untersuchen:
 - Welche Komponenten sind von Taktänderungen betroffen
 - Welche Komponenten werden von einer Task in welcher Art und Häufigkeit benutzt
 - Welche Auswirkungen auf die Ausführungsgeschwindigkeit einer Task haben verschiedene Taktraten
- Wichtig: Skalierungen der Taktfrequenz sind bei...
 - ... mobilen Systemen ohne harte Echtzeitanforderungen meist gefahrlos möglich
 - ... harten Echtzeitsystemen von fraglichem Nutzen bzw. gefährlich

Untersuchungen der Uni Erlangen

- System:
 - Intel Xscale 80200 CPU auf Testboard IQ 80310 mit 32 MB SDRAM
 - Takt in 66 MHz-Schritten von 333 MHz bis 733 MHz einstellbar zur Laufzeit
 - Linux als Betriebssystem
- Messungen:
 - Performance mit Hilfe von Performancecountern
 - * Schrittzähler
 - * Cache Hits/Misses
 - * Speicherzugriffe
 - Stromaufnahme extern gemessen

Stromverbrauch für bestimmte Aufgaben I

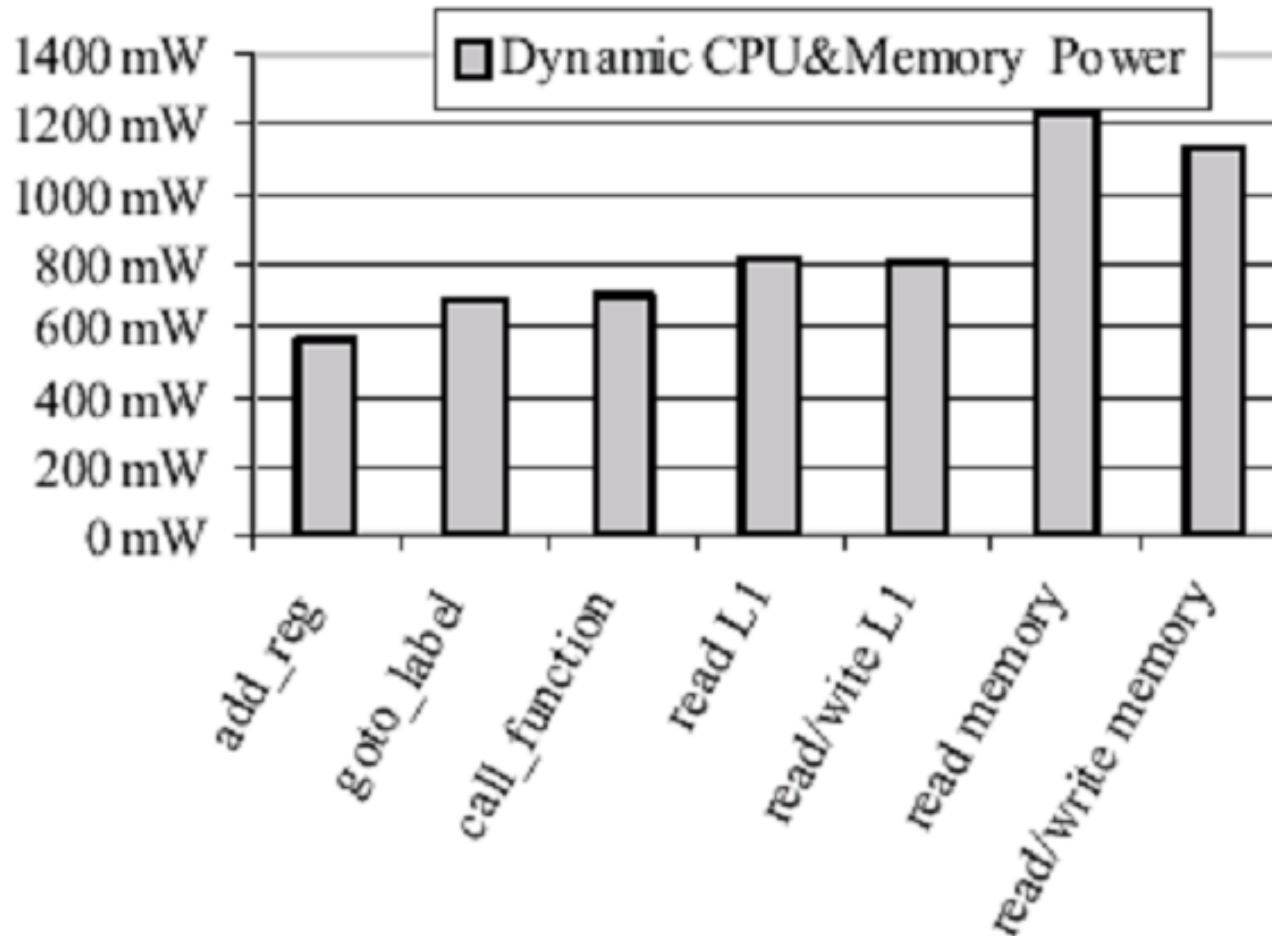


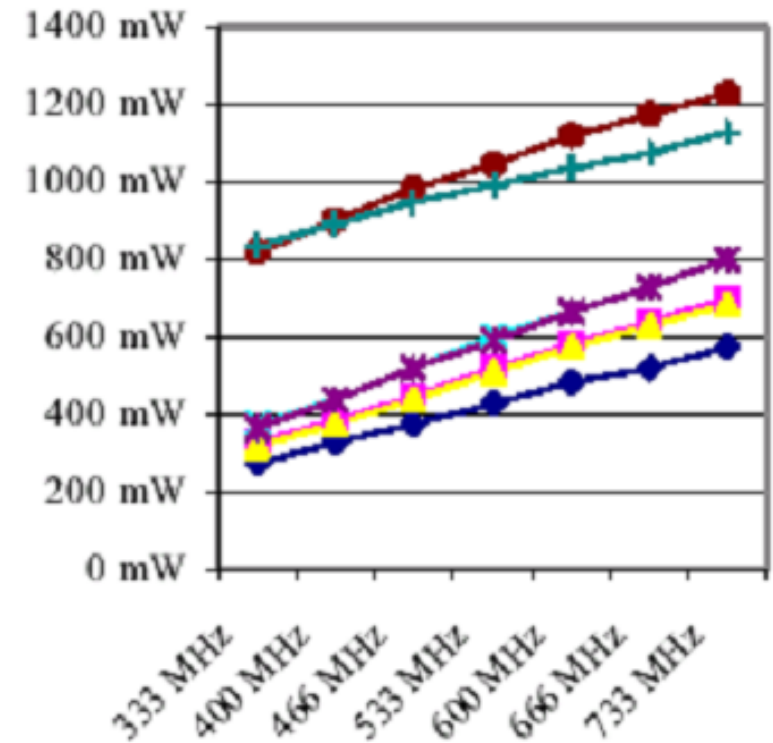
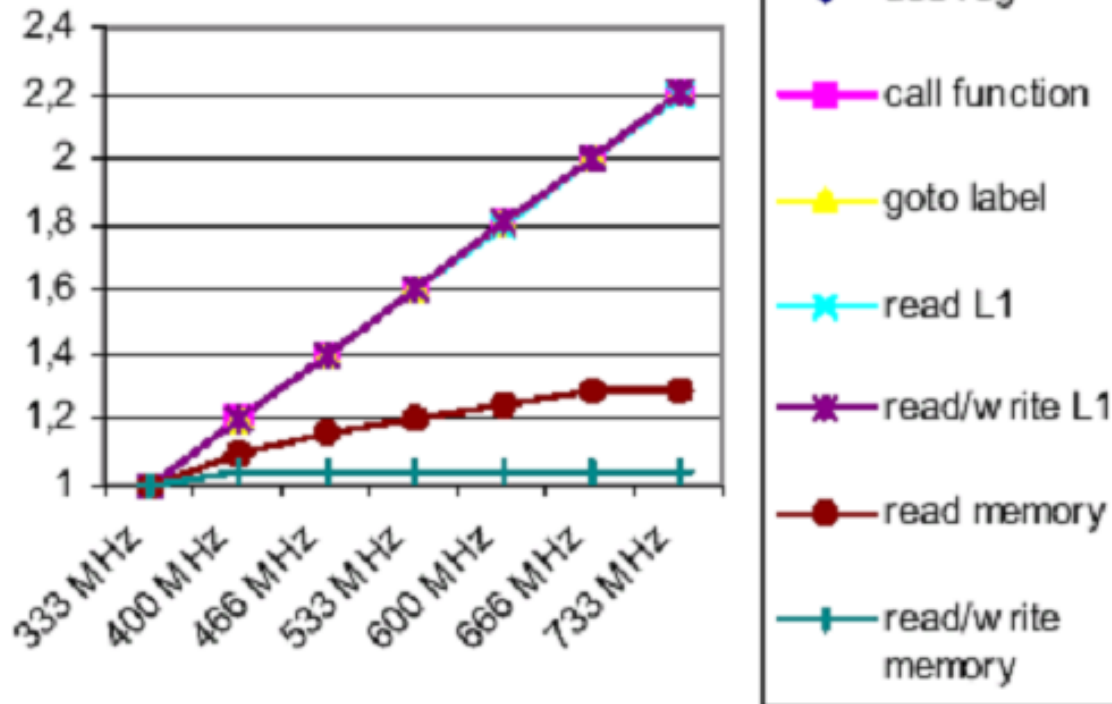
FIG. 1 : IQ80310 Board (XScale@733MHz, 32 MB SDRAM) Power Breakd

Stromverbrauch für bestimmte Aufgaben II

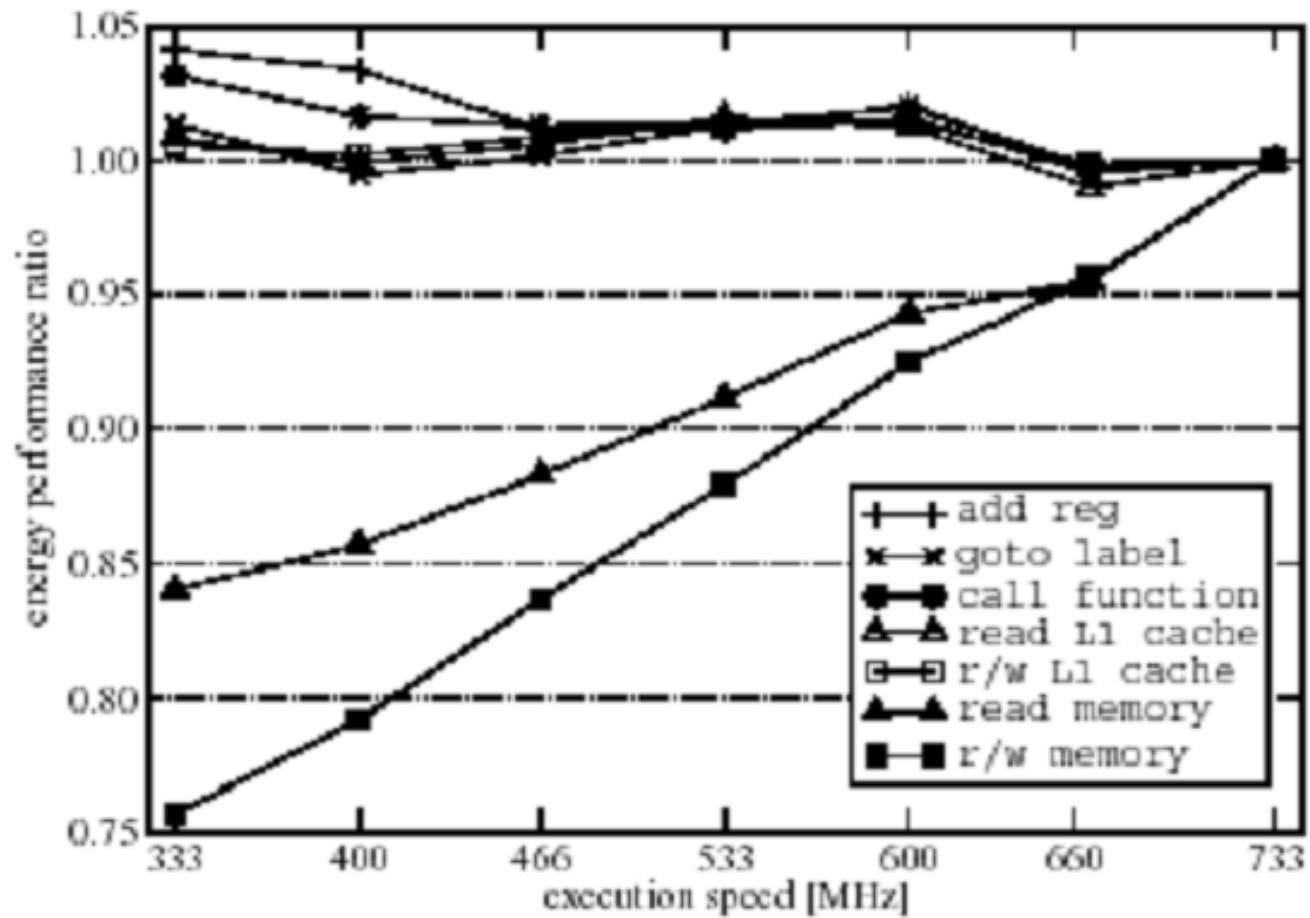
benchmark	instr. per μ s	branches per μ s	L1 ref. per μ s	mem.req. per μ s
add reg	680	12	0	0
goto label	313	104	0	0
call function	379	52	143	0
read L1 cache	548	46	182	0
r/w L1 cache	578	38	307	0
read memory	85	7	28	3.7
r/w memory	43	3	23	4.3

Performance und Stromverbrauch

Performance



Energie-Effizienz I

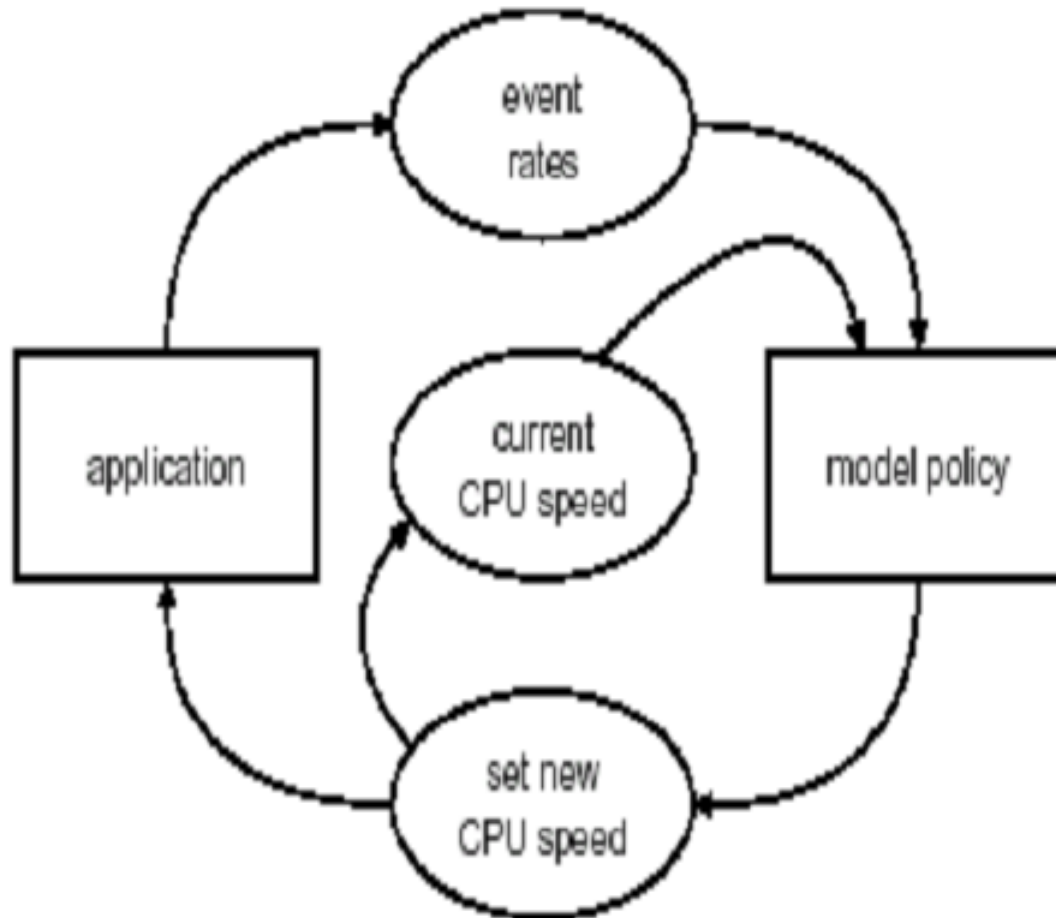


Energie-Effizienz II

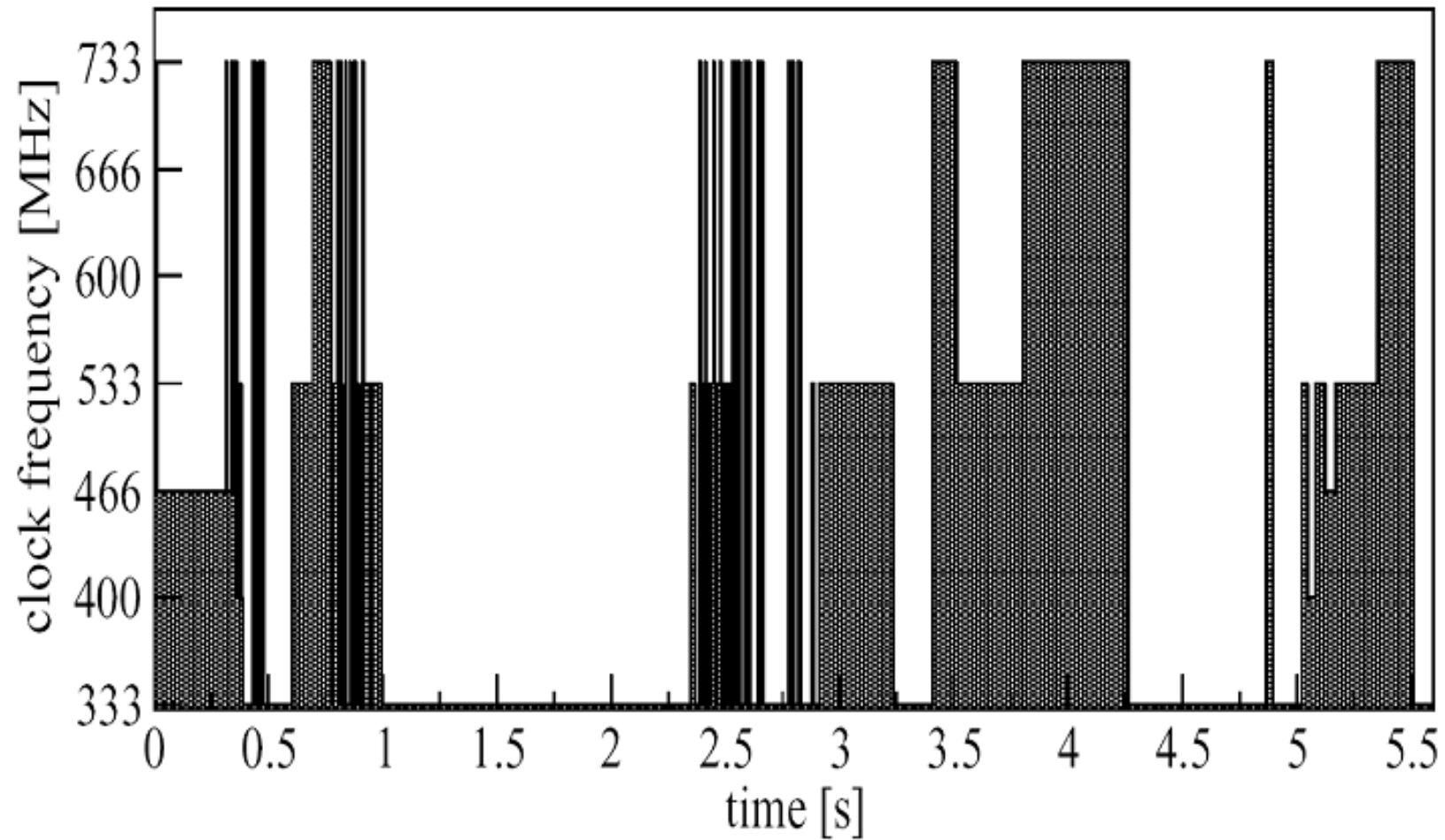
- “Idealfrequenz” ist abhängig von der Aufgabe
- Wahl der Frequenz nach festgestellter Aufgabe
- Problem: Ermitteln der gerade aktiven Aufgabe möglichst feingranular
- Lösung:
 - Datengewinnung durch Beobachtung der Counter und Vergleich mit bekannten Mustern
 - Benutzung von Policies



Policy-Modell



Beispiel: Ghostscript auf Linux

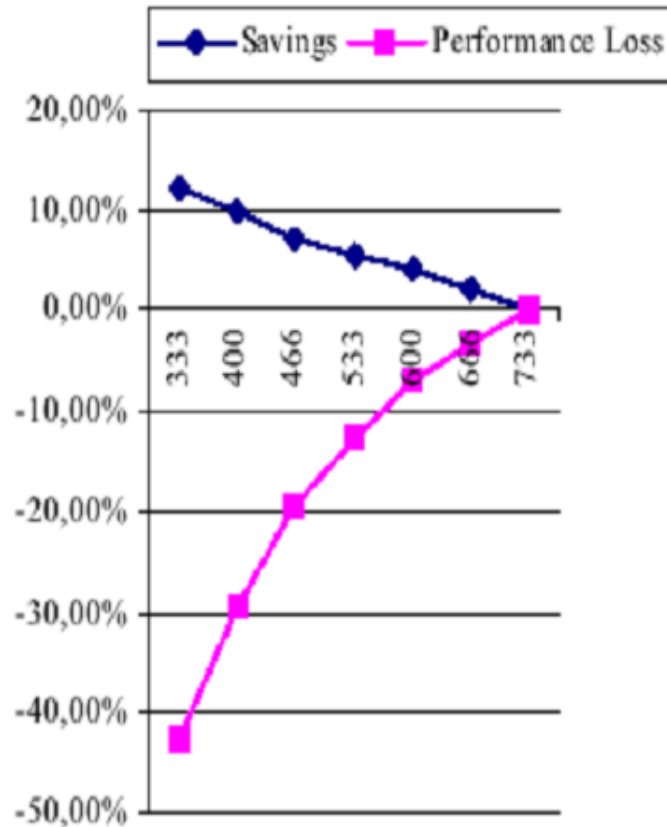
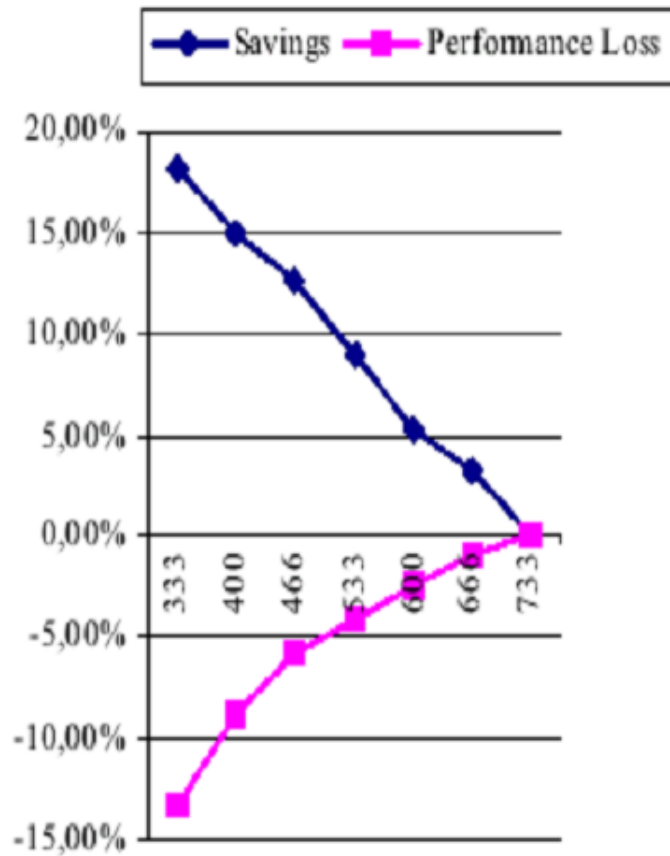


Implementierung in Linux

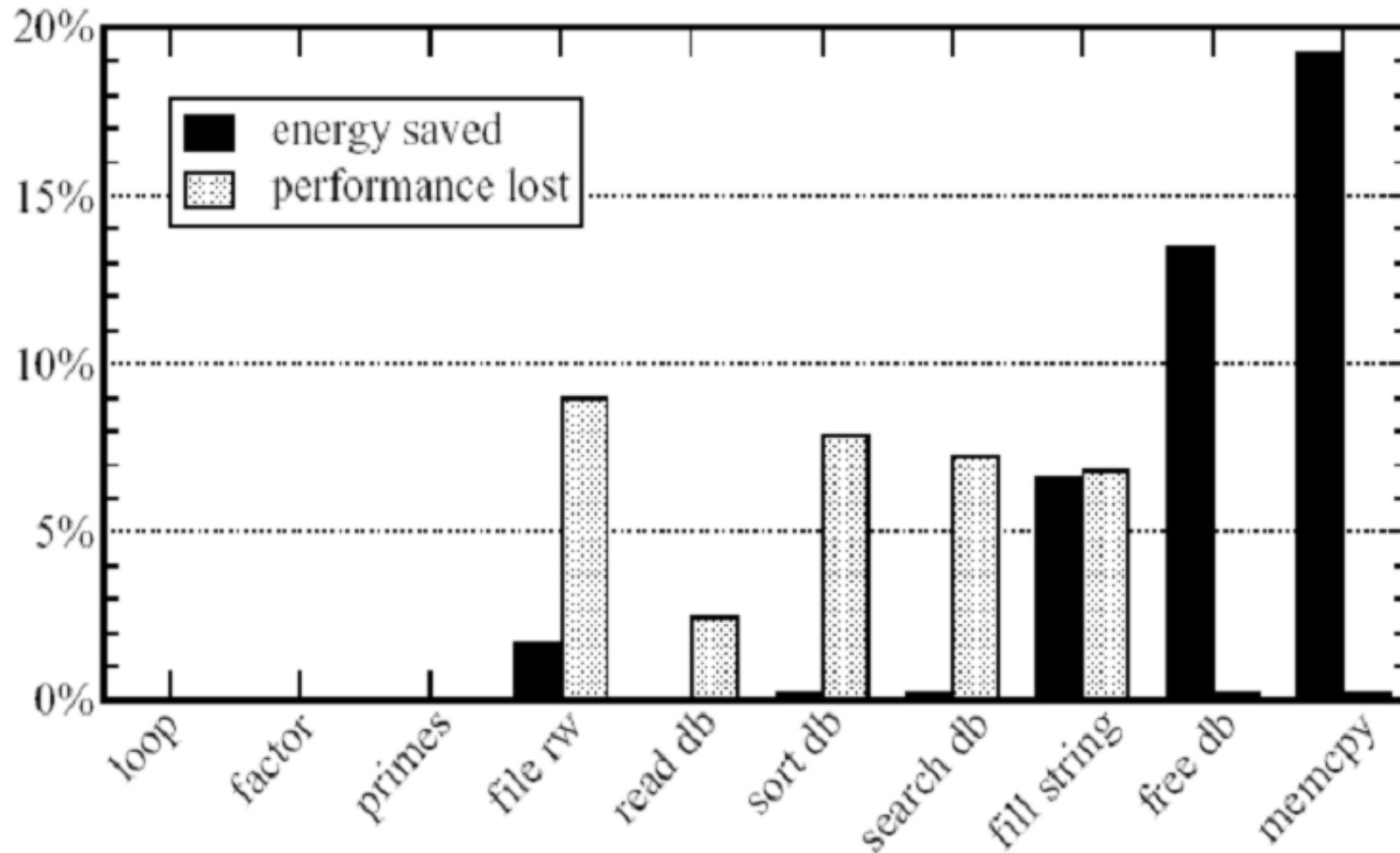
- Anpassung von
 - Routinen für Kontextwechsel
 - Kernel-Daten-Strukturen
- Erfassung und Auswertung der Counterwerte
- Schedule so anpassen, daß optimale Frequenz für Task anliegt
- Problem:
 - Overhead durch Berechnung
 - Benötigt Rechenzeit
 - Benötigt Energie

Performance-Verlust vs. Energieeinsparung I

Grep (links) und Djpeg (rechts)

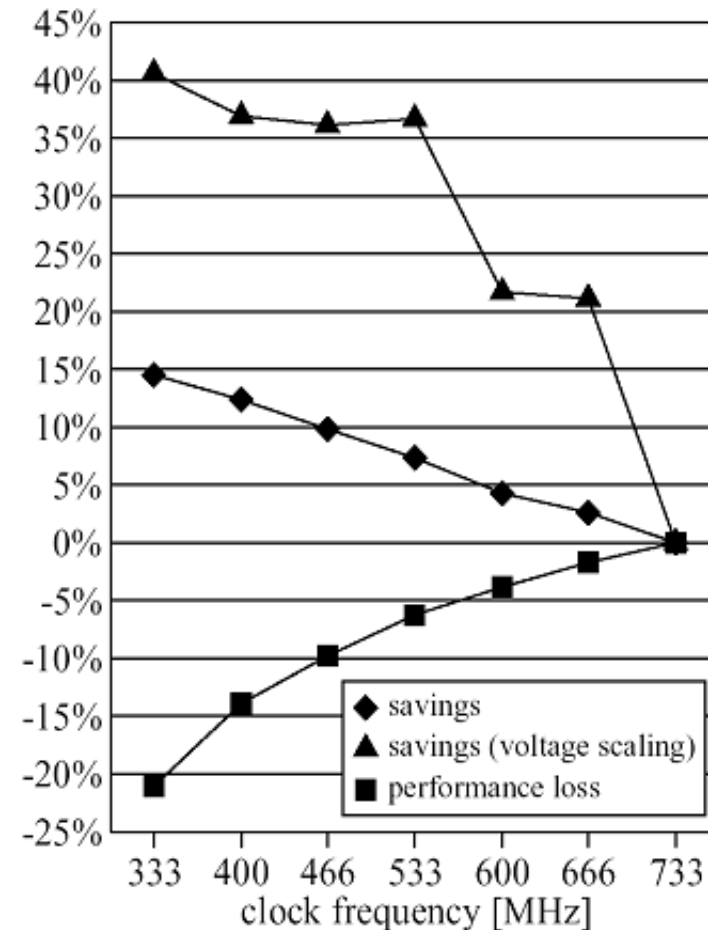


Performance-Verlust vs. Energieeinsparung II



Weitere Möglichkeiten

- Reduzierung der Kernspannung
 - Energieverbrauch ist proportional zu Frequenz
 - Energieverbrauch ist proportional zum Quadrat der Spannung
- Neue Counter für den Energieverbrauch (erfordert Hardware-Anpassung)
- Getrennte Behandlung von Kernel- und Interrupt-Routinen



Steuerung auf Ebene der Anwendungen

- Auf Anwendungsebene sind Möglichkeiten offen, die das Betriebssystem nicht hat:
 - Zeitpunkt des Zugriffs auf energieintensive Peripherie ist oft verschiebbar
 - Dauer der Nutzung von Peripherie ist oft variabel
- Beispiel: Zugriffe auf eine Festplatte (Uni Erlangen: “Kooperatives I/O”)
 - Schreibzugriffe können verzögert werden
 - Lesezugriffe können u.U. vorverlegt werden
 - Dauer eines Zugriffes kann durch Pufferung reduziert werden
- Idee: Zeit in Energiesparmodus maximieren durch
 - Gebündelte Anfragen
 - Reduzierte Anzahl der Moduswechsel

- Moduswechsel kosten Zeit und Energie
 - Motoren starten/stoppen
 - Zeit für Beschleunigung bzw. Abbremsen
 - Positionieren der Köpfe
 - eventuell An/Abmelden von Komponenten
- Interessant: Break-even-Zeitspanne
 - Zeit in einem Stromsparmodes, ab dem mehr Energie eingespart wird als der Moduswechsel kostet
 - Stark abhängig von der Hardware
- Nicht berücksichtigt: Zusätzliche Kosten durch Verschleiß der Hardware infolge häufiger Start/Stop-Zyklen

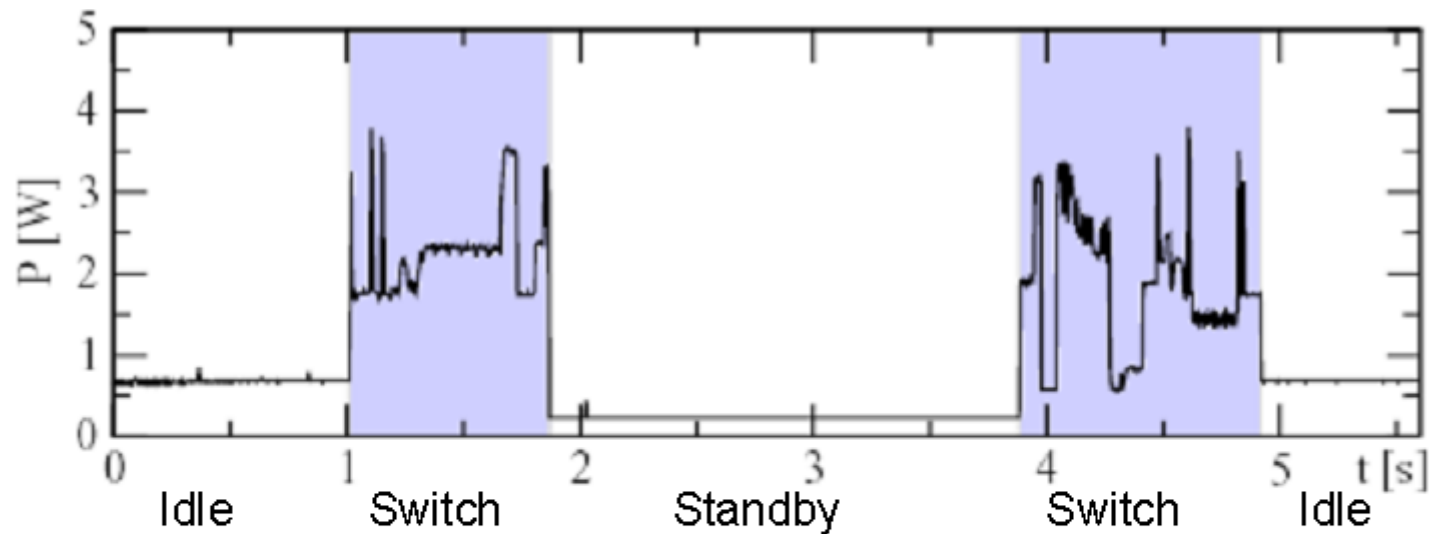
Energiespar-Modi II

IBM Travelstar 15GN: Break-even-Zeitspanne etwa 8,7 Sekunden

Modus	Eigenschaften	Energieverbrauch	Verzögerung bei Zugriff
Aktiv	Lese-, Schreibvorgang, Moduswechsel	2.1–4.7 W	—
Idle		1.85 W	—
Low-Power Idle	Köpfe geparkt	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

Energiespar-Modi III

- Beispiel für Moduswechsel:
 - Parken/Positionieren der Plattenköpfe
 - Stoppen/Anfahren des Motors

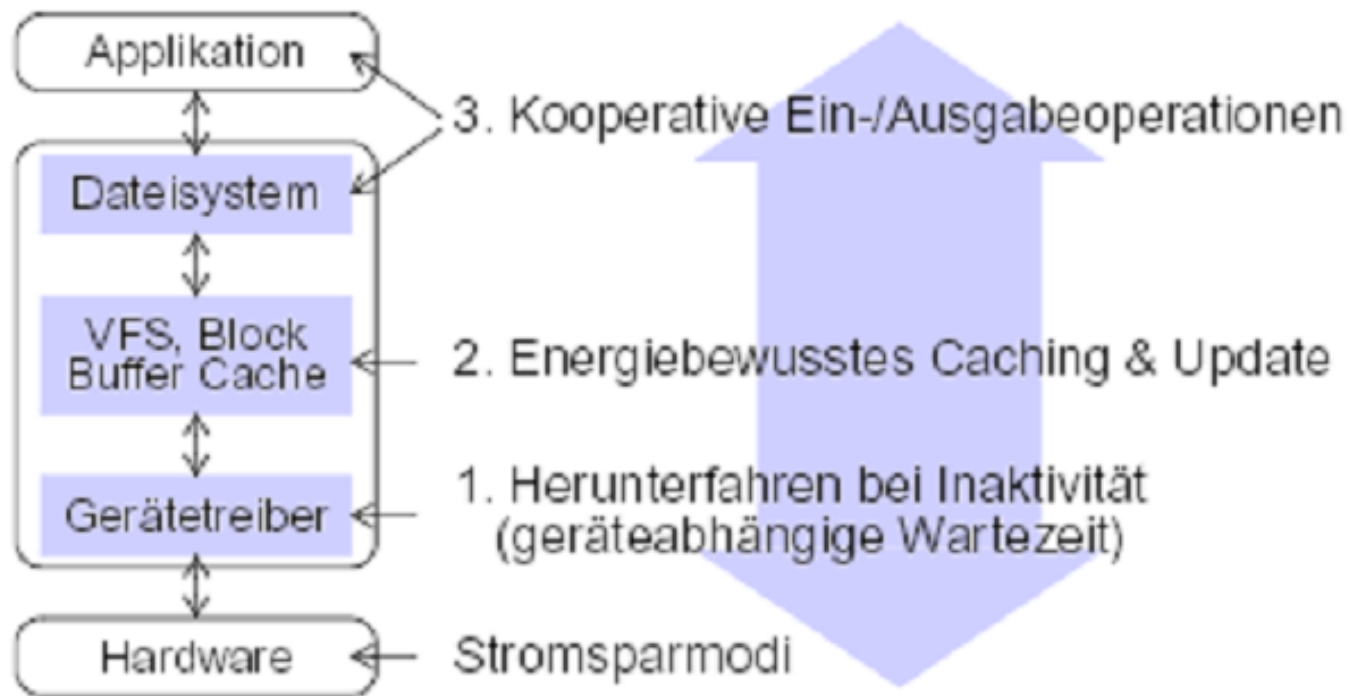


Energiespar-Modi IV

- Umschalten in Energiesparmodus nur effektiv, wenn $\text{idle-Zeit} > \text{break-even-Zeitspanne}$
- Problem: Vorhersage künftiger Zugriffe ist schwierig
- Bekannte Ansätze:
 - Bestimmung der Zugriffsmuster
 - Feste Time-Outs
 - Adaptive stochastische Verfahren
- Kooperatives I/O der Uni Erlangen:
 - Gezielte Steuerung der Zugriffe, soweit möglich
 - Details: Nächste Folien

Kooperatives I/O I

Energieverbrauch ist nichtfunktionale Eigenschaft, also Behandlung auf allen Ebenen



Kooperatives I/O II

- Zukünftige Zugriffe sind weitgehend bekannt, da auf Anwendungsebene durch Time-outs selbst definiert
- I/O-Aktionen werden eingeteilt in
 - verzögerbar
 - abbrechbar
- Dadurch: Vorhersagbarkeit deutlich verbessert
- Implementation:
 - Überladen der Standard-I/O-Funktionen:
 - * Time-out
 - * Flag für Abbruch (cancel-flag)
 - Wrapper für “legacy” Anwendungen:
 - * Time-out = 0
 - * cancel-flag nicht gesetzt

Kooperatives I/O III

- Modus “Active”:
 - Sofortiges Bedienen aller Anfragen
- Modus “Standby”:
 - Verzögern der Operation bis zum Erreichen des nutzerdefinierten Time-outs
 - oder: Verzögern der Operation bis Gerät durch anderen Prozeß aktiviert wird
- Aktivieren des Gerätes wird entweder erzwungen (cancel-flag nicht gesetzt) oder Operation wird abgebrochen (cancel-flag gesetzt)
- Vor jedem Moduswechsel nach “Standby”: Herausschreiben aller Puffer
- Bei jeder Schreib/Lese-Aktion: Herausschreiben aller Puffer

Kooperatives I/O IV

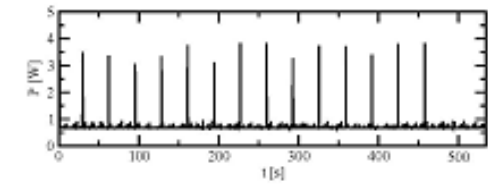
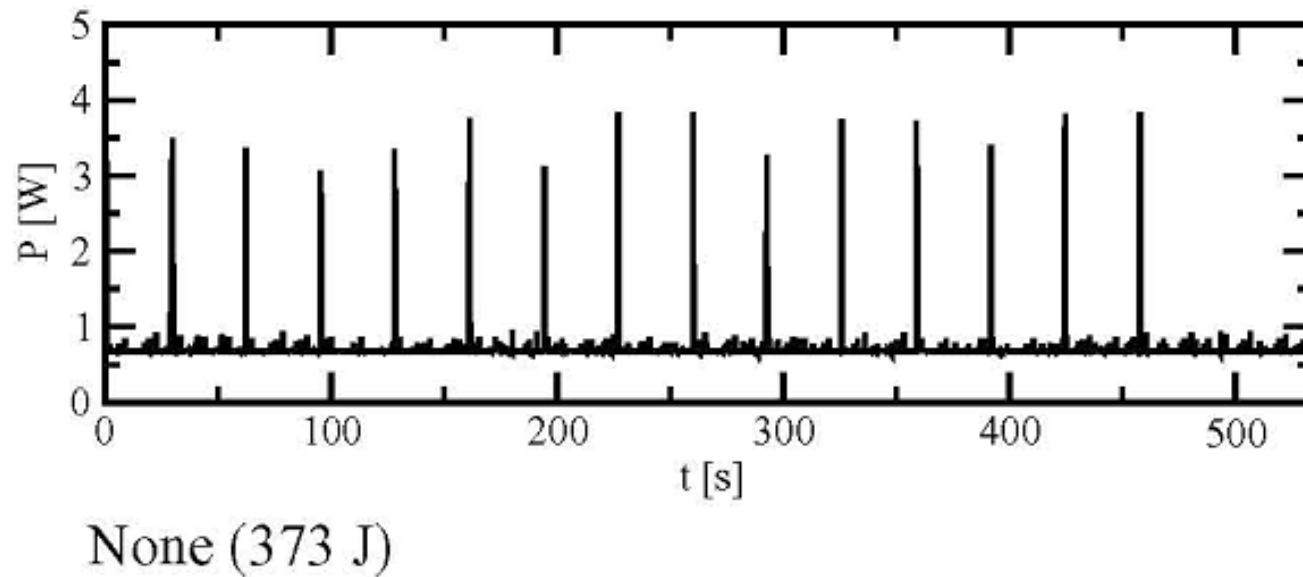
- Wechsel in den “Standby”-Modus ist geräteabhängig
 - Geräteabhängige Vorgaben für Time-out
 - In Abhängigkeit vom letzten Zugriff und der Break-even-Timespanne
- Bei Plattensystemen:
 - Separate Behandlung der Einzelplatten
 - Puffer werden Laufwerken zugeteilt (unterhalb der Ebene des File-Systems)
 - * Sinnvoll bei LVM (logical volume manager)
 - * Problematisch bei RAID-Systemen

Kooperatives I/O: Messungen I

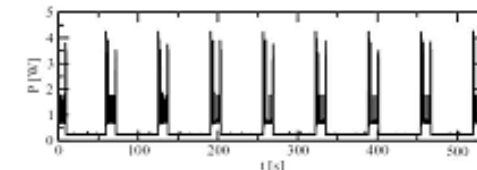
Beispiel der Uni Erlangen für kooperatives I/O:

- Kooperativer Audio-Player
 - Lesen aus Musikpuffer
 - Zweiter Puffer durch kooperatives Lesen gefüllt (verzögerbar, nicht abbrechbar)
- Minütlich aufgerufenes Mailprogramm
 - Schreibt Mails auf die Platte
- Untersuchung verschiedener Kombinationen

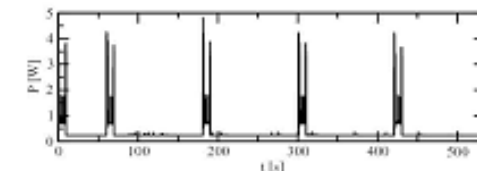
Kooperatives I/O: Messungen II



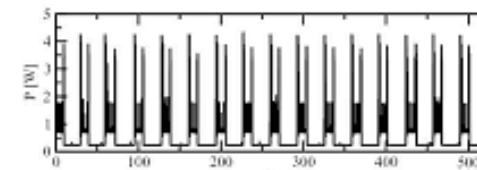
None (373 J)



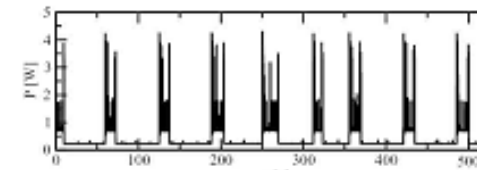
Tocatta cooperative (210 J)



Mail cooperative (164 J)

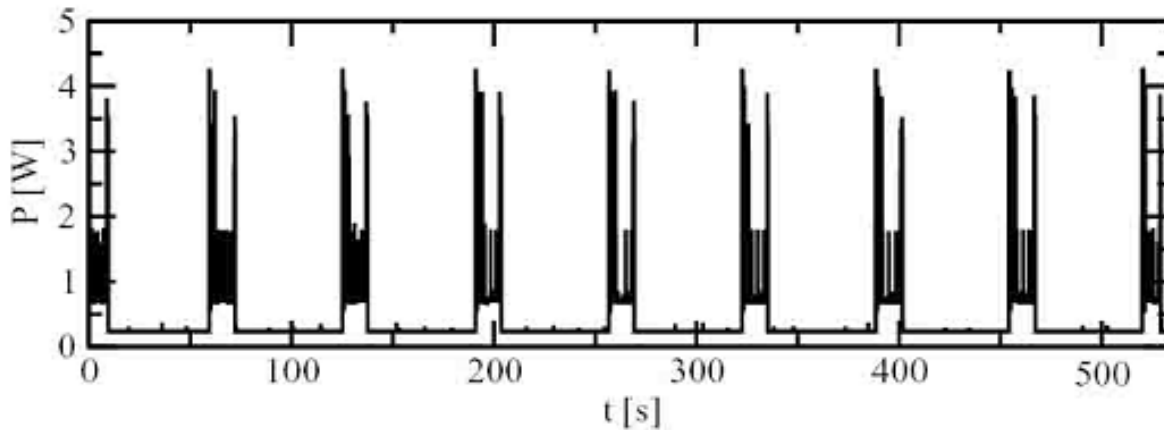


Tocatta & Mail non-cooperative (270 J)

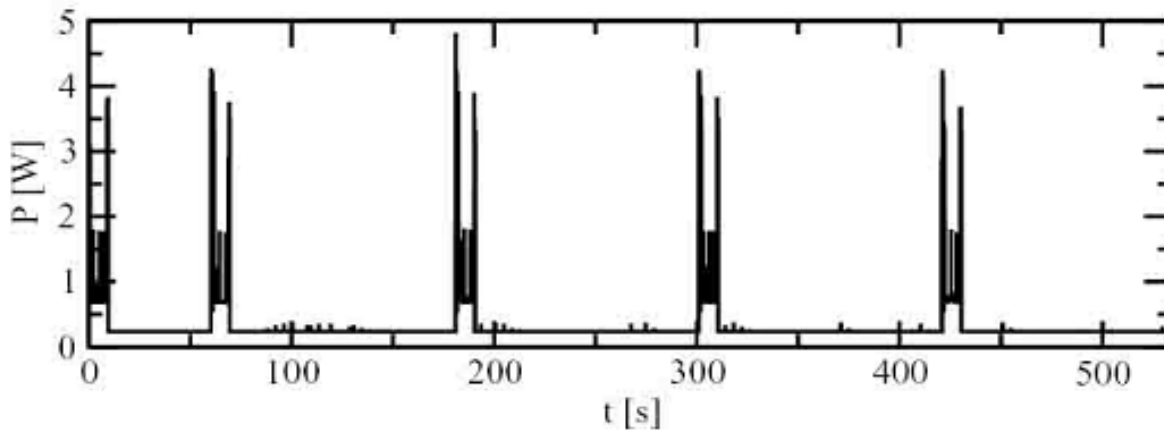


Tocatta & Mail cooperative (227 J)

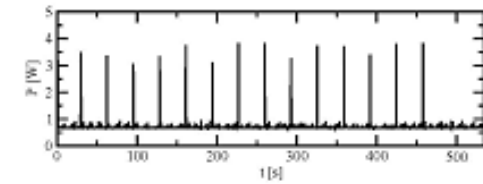
Kooperatives I/O: Messungen III



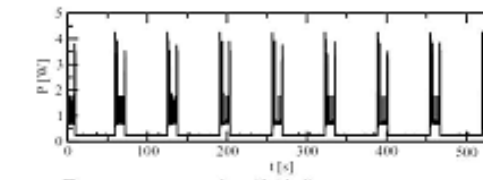
Toccata cooperative (210 J)



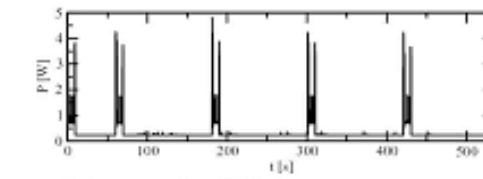
Mail cooperative (164 J)



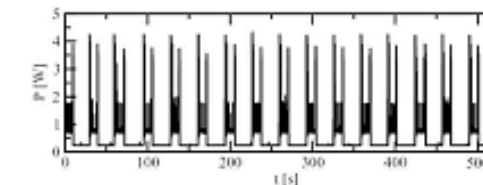
None (373 J)



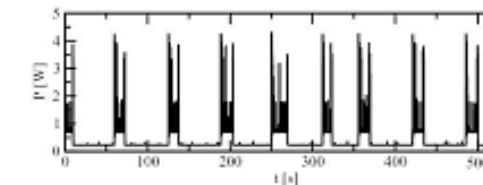
Toccata cooperative (210 J)



Mail cooperative (164 J)

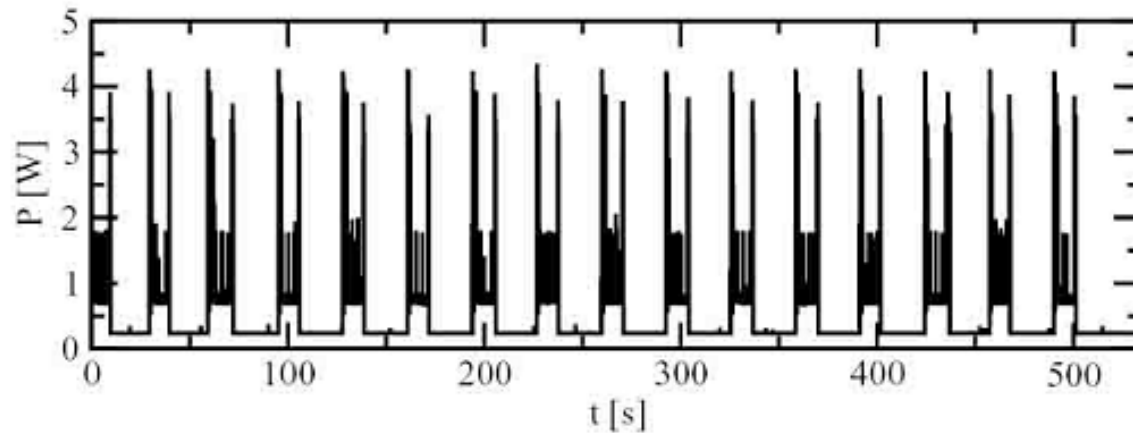


Toccata & Mail non-cooperative (270 J)

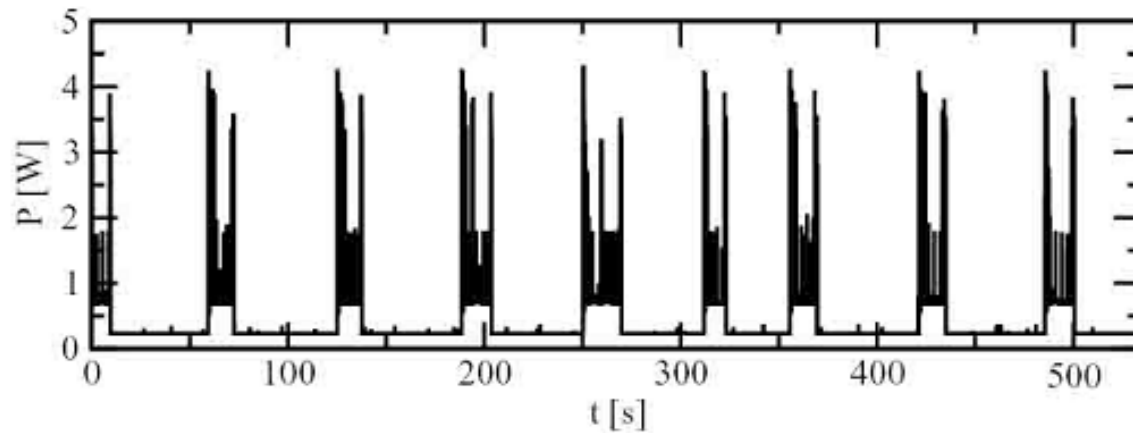


Toccata & Mail cooperative (227 J)

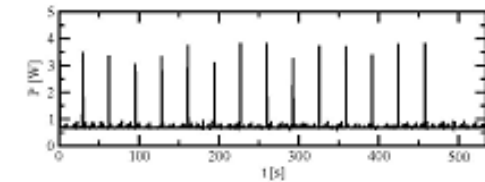
Kooperatives I/O: Messungen IV



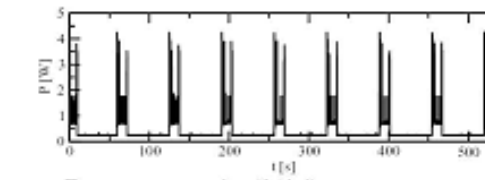
Toccata & Mail non-cooperative (270 J)



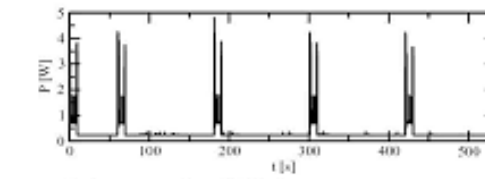
Toccata & Mail cooperative (227 J)



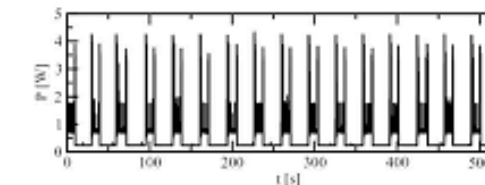
None (373 J)



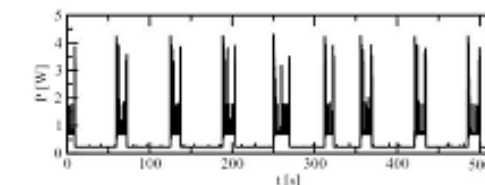
Toccata cooperative (210 J)



Mail cooperative (164 J)

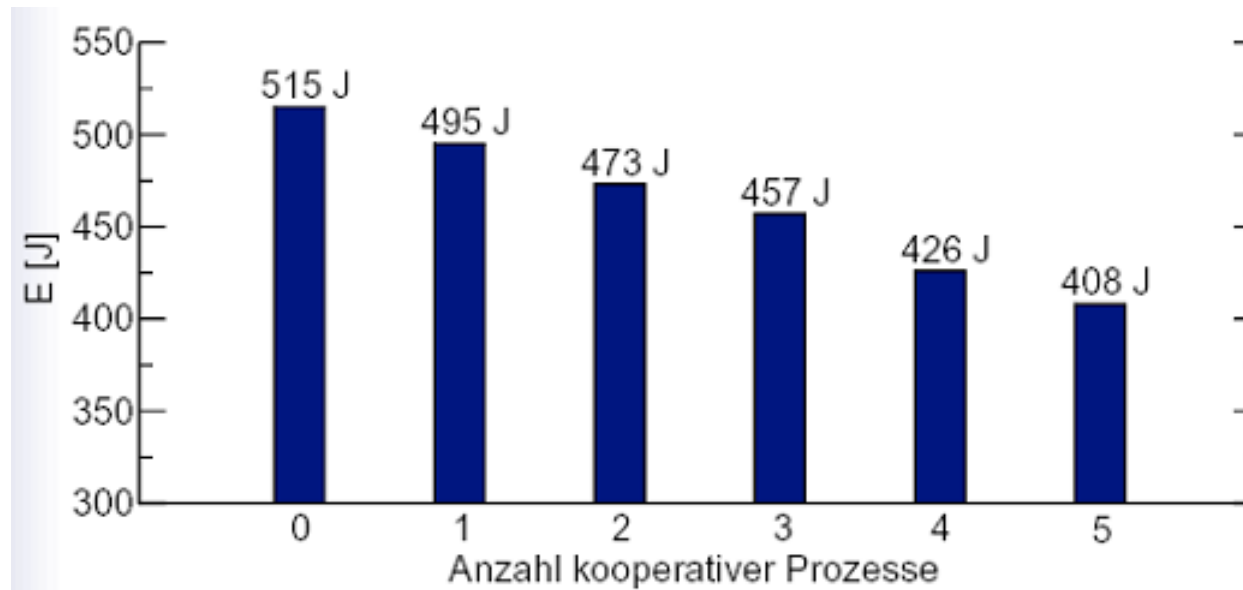


Toccata & Mail non-cooperative (270 J)



Toccata & Mail cooperative (227 J)

Kooperatives I/O: Messungen V



Energieverbrauch hängt von Anzahl der kooperativen Prozesse ab

Entwicklungstendenzen

- Neue Technologien in der Chipherstellung (Verringerung von Leckströmen)
- Asynchrone Schaltungen
- Entwicklung von energieoptimierter Software
 - Benutzung/Weiterentwicklung der vorgestellten Ansätze
 - Entwicklung neuer Ansätze
- Desktop-Bereich:
 - Einsatz von Mobil-Technologien
 - Energiebewußtsein der Nutzer