Lernzettel

Nachhaltigkeit, Energieeffizienz und Leistungsoptimierung in Betriebssystemen: Energiebewusstes Scheduling, Profiling und ressourcenschonende Betriebssysteme

Universität: Technische Universität Berlin
Kurs/Modul: Systemprogrammierung
Erstellungsdatum: September 6, 2025

Lernzettel: Nachhaltigkeit, Energieeffizienz und Leistungsoptimierung in Betriebssystemen
(Nachhaltigkeit, Energieeffizienz und Leistungsoptimierung in Betriebssystemen: Energiebewusstes Scheduling, Profiling und ressourcenschonende Betriebssysteme)

(1) Zielsetzung und Einordnung. Die Auswirkungen von Betriebssystemeinstellungen auf Energieverbrauch, Wärmeentwicklung und Ressourcennutzung stehen im Fokus. Ziel ist es, durch intelligentes Scheduling, gezieltes Profiling und robuste, ressourcenschonende Architekturen den Energieverbrauch zu senken, ohne die Systemleistung unverhältnismäßig zu beeinträchtigen. Zur Quantifizierung wird oft herangezogen:

$$E \approx \int_0^T P(t)\,dt \;=\; \sum_i P_i\,\Delta t_i,$$

wobei $P(t)$ die Leistung und T die Ausführungszeit ist. Energy-Driven-Ansätze berücksichtigen neben der reinen Laufzeit auch die Energie-Delay-Produktigkeit (EDP) und ähnliche Metriken.

(2) Energiebewusstes Scheduling. Ziele: reduziertes Energieverhalten bei gleichem oder akzeptiert reduziertem Leistungsniveau, verlängerte Akkulaufzeit (bei mobilen Systemen) und geringere Wärmeentwicklung. Wesentliche Konzepte:

- Dynamische Spannungs- und Frequenzanpassung (DVFS) bzw. CPU-Frequenzskala (z. B. P-States) zur Reduktion von Leistungsaufnahme bei geringer Auslastung.
- Tiefschlaf- und Leerlaufzustände (C-States, Idle States) zur Minimierung unnötiger Energieabgabe.
- Energie-aware Scheduling-Strategien (EAS): Neben Performance treffen Scheduler-Entscheidungen auch Energieziele; CPU-Governors wie „powersave“, „ondemand“, „schedutil“ optimieren adaptiv.
- Leistungs- vs. Energie-Optimierung: Trade-offs zwischen Reichweite und Reaktionszeit; Nutzung von Vorhersagen (Lastprognose, Workload-Charakterisierung).

Typische Metriken:

- Energie je Task, Energie-Delay-Produkt (EDP), Energie-Delay-Squared (ED^2P)$Idle-und Power-States-Verhalten, C-State-Residuum$

(3) Profiling für Energieeffizienz. Ziel ist es, Engpässe und Einsparpotenziale zuverlässig zu erkennen, damit gezielt Optimierungen vorgenommen werden können.

- Tools und Umfeld:
  - Leistungsmessung: PowerTop, turbostat, Powertop-like-Tools
  - CPU- und Scheduler-Insights: perf, ftrace, eBPF-basierte Tracing-Ansätze
  - Speicher- und I/O-Profiling: iostat, vmstat, pidstat, sar

(4) Ressourcenschonende Betriebssysteme. Kernziel ist es, Betriebssystemkomponenten so auszulegen, dass sie mit geringem Ressourcenverbrauch arbeiten, ohne Funktionalität zu opfern.

- Speicherhierarchie und Caching:
  - Optimierte Page-Cache-Strategien, transparente Huge Pages, aggressives Prefetching bei Bedarf
  - Speichermanagement-Strategien unter Druck (Memory Pressure, Ballooning in Virtualisierung)
- I/O und Treiberlogik:
  - asynchrones I/O-Handling, energiesparende Interrupt-Strategien, Wake-Lock-Vermeidung
  - energieeffiziente Treiber-Architekturen und device power management (Runtime PM)
- Scheduling über Kerne und Ressourcenverteilsysteme:
  - CPU-Affinität, NUMA-Bewusstsein, Energiemanagement im Scheduler
  - Multi-Core- und Clover-Strategien zur Verhinderung unnötiger Synchronisations- und Kontextwechsel-Aufrufe
- Sicherheit, Robustheit und Nachhaltigkeit:
  - sichere, resiliente Systeme, die Energieeffizienz nicht auf Kosten von Verfügbarkeit oder Sicherheit opfern

(5) Architektur- und Implementierungsaspekte. Um Energieeffizienz systematisch zu verbessern, sind folgende Punkte zentral:

- Schnittstellen zwischen Scheduler, DVFS/CPU-Governors und Power Management
- Instrumentierung des Kernels zur Erfassung von Energie- und Leistungsdaten (z. B. Messpunkte, Ereignis-Trigger)
- Strategien zur Minimierung von Synchronisations-Overheads (z. B. reduzierte Kontextwechsel, bessere Cache-Nutzung)
- Berücksichtigung von Virtualisierung und Containern (Overheads vs. Isolation)
- Greifbare Implementierungsideen:
  - Anbindung eines Energie-Controllers an Scheduling-Entscheidungen (lastbasierte DVFS-Strategien)
  - Anpassung der Wakeup-Strategien bei I/O-Events, um Leerlaufzeiten sinnvoll zu nutzen
  - Einführen von adaptiven C- und P-State-Governors basierend auf Workload-Profilen

(6) Beispiel- und Übungsaufgaben.

- Analysieren Sie ein Desktop- bzw. Serversystem unter Last: Welche DVFS-Stufen werden genutzt? Welche C-States treten auf? Welche Einsparpotenziale lassen sich identifizieren?
- Entwerfen Sie ein einfaches Profiling-Skript, das CPU-Auslastung, Energieaufnahme und Laufzeit eines Mikrobenchmarks erfasst und grafisch darstellt.

- Diskutieren Sie Trade-offs zwischen Energiesparen und Latenz/Throughput in einem reellen Scheduling-Szenario (z. B. Webserver-Last vs. Hintergrund-Background-Tasks).
- Implementieren Sie eine Mini-Simulation in C, die zwei Tasks mit unterschiedlichen Energieprofilen zeitabhängig priorisiert (als Übungsbeispiel, ohne Kernel-Modifikation).

(7) Kurznotizen und Formeln.

- Energie-Einsparung bei DVFS wird grob durch Verringerung von Leistung P bei gleicher oder verlängerter Ausführungszeit beeinflusst.
- Typische Modelle nutzen $E = \int P(t)\,dt$ oder näherungsweise $E \approx \sum_i P_i \Delta t_i$.
- Wichtige Begriffe: DVFS, P-States, C-States, Idle States, CPU-Governor, Runtime PM, NUMA, Cache-Effizienz.