Betriebssystemtechnik

Nachlese

Wolfgang Schröder-Preikschat

Lehrstuhl Informatik 4

17. Juli 2012

Gliederung

Nichtsequentielle Systemprogrammierung

Koordination der Kooperation und Konkurrenz zwischen gleichzeitigen Prozessen

Verfeinerung/Vertiefung von Betriebssysteme [5]

Progammunterbrechungen	• Unterbrechungsprinzipien • Ereigniszustellung $\rightsquigarrow$ AST • Unterbrechungsweitergabe
Fadenimplementierung	• minimale Basis $\rightsquigarrow$ Koroutine • minimale Erweiterungen
Fadenumschaltung	• Prozesseinlastung $\rightsquigarrow$ *Dispatching*
Prozesssynchronisation	• Sperrverfahren $\rightsquigarrow$ „*spin lock*“-Familie • Wettlauftoleranz • Prozesssteuerung $\rightsquigarrow$ Semaphor
Softwaretechnik	• aspektorientierte Programmierung $\mapsto$ KSS • (funktionale) Hierarchie • Programmfamilie

Gliederung

Mikrovergleichstest: Dateideskriptortabelle [2]

Experiment zur **Skalierbarkeit** von (POSIX-artigen) Betriebssystemen auf symmetrischen Multiprozessorsystemen mit gemeinsamem Speicher:

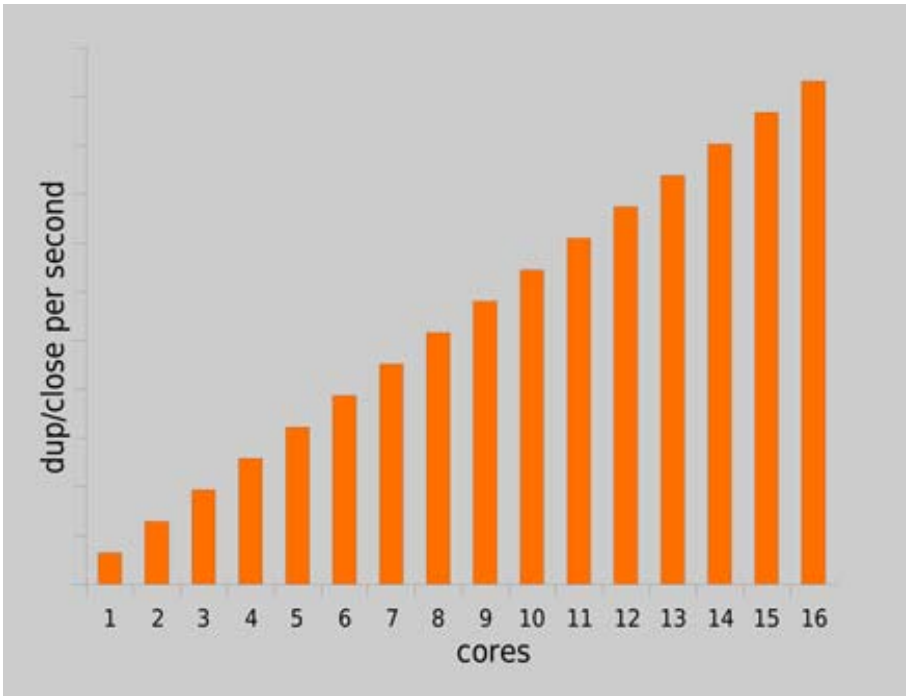
- inkrementelle Erzeugung von Fäden innerhalb desselben Prozesses
  - also schrittweiser Aufbau eines parallelen, mehrfädigen Prozesses
- jeder Faden erzeugt einen Dateideskriptor und ruft wiederholt auf:
  - `dup(2)` auf den eigenen (erzeugten) Dateideskriptor
  - `close(2)` des Dateideskriptorduplikats
- Zugrundeliegende Ausführungsplattform:
  - vier vierkernige Prozessoren (AMD Opteron)
  - Linux 2.6.25

Beachte: POSIX-Semantik

- ein neuer Dateideskriptor ist *allen* Fäden desselben Prozesses sichtbar
- auch wenn nur ein einzelner Faden diesen Dateideskriptor nutzen wird

Mikrovergleichstest: Theorie vs. Praxis

Theorie

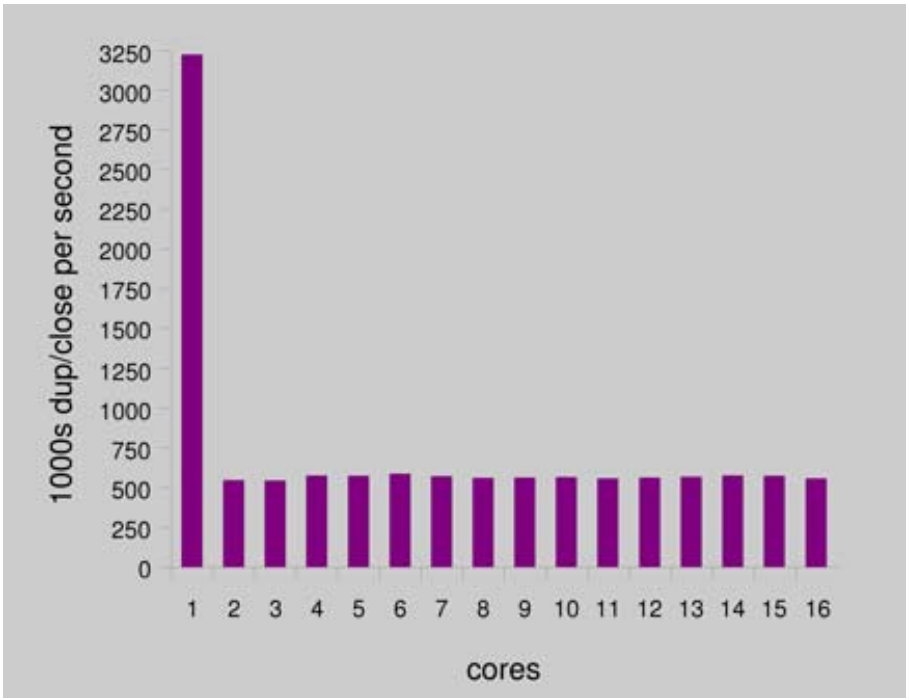


Quelle: [1, S. 11]

Erwartung

- Durchsatz skaliert
- linearer Anstieg

Praxis



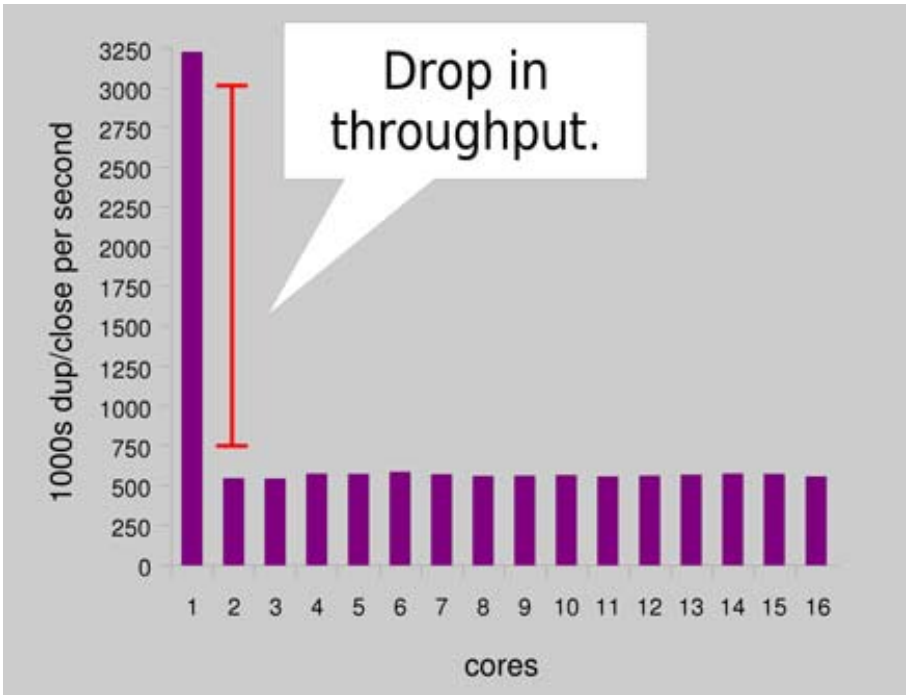
Quelle: [1, S. 12]

Realität

- Durchsatz fällt ab
- skaliert nicht

Mikrovergleichstest: Wettstreit

Leistungsabfall

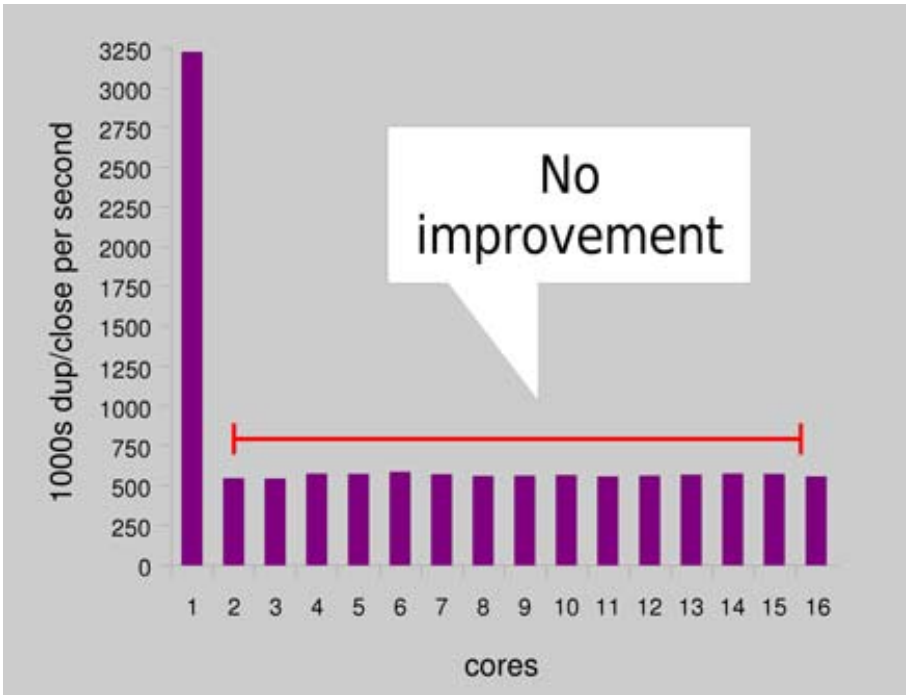


Quelle: [1, S. 13]

Erwartung

- (leichter) Anstieg
- keinen Einbruch

Verstetigung



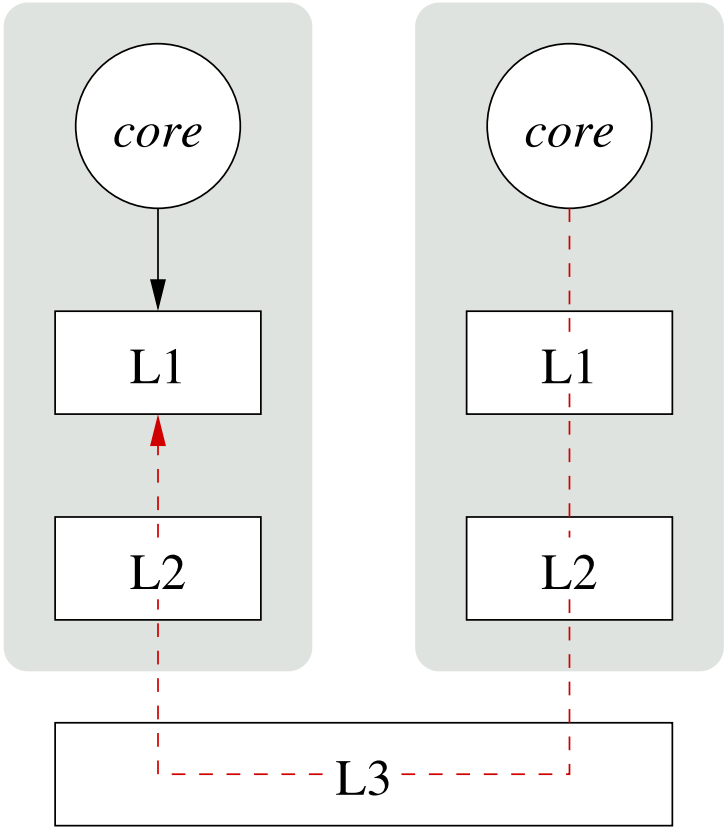
Quelle: [1, S. 14]

Erwartung

- gewisse Erholung
- keinen Bestand

Mikrovergleichstest: Datenlokalitäten [1, S. 15–16]

```
fd_alloc(void) {
    lock(fd_table);
    fd = get_free_fd();
    set_fd_used(fd);
    fix_smallest_fd(fd);
    unlock(fd_table);
}
```



- lokaler Zugriff in 3 Zyklen
- Fernzugriff in 121 Zyklen
  - Zugriffsfehler (*cache miss*)
  - Kohärenzprotokoll

- falsche Mitbenutzung (engl. *false sharing*) von Zwischenspeicherzeilen

Mikrovergleichstest: Mitbenutzung [1, S. 17]

```
fd_alloc(void) {
    lock(fd_table);
    fd = get_free_fd();
    set_fd_used(fd);
    fix_smallest_fd(fd);
    unlock(fd_table);
}
```

Umlaufsperre

- Schutz des kritischen Abschnitts
- serialisiert Aktualisierungen auf `fd_table`
- obwohl die Fäden verschiedene Einträge in der Tabelle benutzen

- falsche Mitbenutzung (engl. *false sharing*) der Dateideskriptortabelle

Beachte

- bezogen auf den kritischen Programmtext erscheint die Umlaufsperre durchaus passend platziert zu sein
- hinsichtlich der hier eigentlich zu schützenden Daten stellt sie jedoch einen Flaschenhals dar

Skalierbarkeit: Linux [8]

Fähigkeit eines Systems, sein **Leistungsvermögen** durch Hinzu- oder auch Wegnahme von Betriebsmitteln (im Idealfall) linear zu beeinflussen

- Linux 2.4 skaliert kaum über vier Prozessorkerne hinaus
- Linux 2.6 verbringt 91% der Zeit in Umlaufsperren
- Vergleichstests im Zusammenhang mit EXT4 zeigten:
  - die meisten skalieren gut bis 6–7 von 8 Prozessorkernen
  - viele skalieren gut bis 12 von 16 Prozessorkernen
  - eine akzeptable Anzahl skalieren gut bis 32 Prozessorkerne

„Büchse der Pandorra“

- bevorstehende Systeme mit 32 Prozessorkernen erfordern schon einige Anstrengungen für Verbesserungen
- zukünftige Systeme mit (weit) mehr Prozessorkernen sind ein Problem

Empfehlungen zum Bau vielkernfähiger Betriebssysteme

Erfahrungen aus Corey [1, 2]

- falsche Mitbenutzung (engl. *false sharing*) ausschließen/beseitigen
- „schluderige“ (engl. *sloppy*; hier: phasenweise inkonsistente) Zähler
- prozessor(kern)lokale Datenstrukturen/-objekte
- sperrfreie Vergleiche
- Vorbeugung/Vermeidung unnötiger Blockierungen

Erfahrungen aus EXT4 [8]

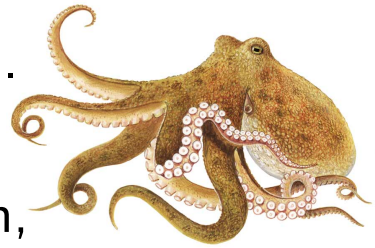
- atomare Variablen
- Lese-/Schreibsperren
- feinkörnige Sperren
- bündelweise Verarbeitung kritischer Befehle/Abschnitte

Gliederung

Laufzeitunterstützungssystem für invasives Rechnen [6]

Octo

- der Bezeichnung eines Wesens entnommen, das:
  - (a) hoch parallel in seinen Aktionen ist und
  - (b) sich sehr gut an seine Umgebung anpassen kann.
- der Krake (Ordnung *Octopoda*)
  - kann kraft seiner (acht) Tentakel parallel agieren,
  - vermag sich durch Farbänderung anzupassen und
  - verfügt über ein hoch entwickeltes Nevensystem,
  - das es ihm ermöglicht, sich auf dynamische Umgebungsbedingungen und -einflüsse einzustellen



POS

- Abk. für (engl.) *Parallel Operating System*
  - ein Betriebssystem, das nicht bloß parallele Prozesse unterstützt
  - sondern dabei selbst **inhärent parallel** arbeitet

⇒ `http://invasic.informatik.uni-erlangen.de/en/tp_c1.php`

Fäden als erstrangige Abstraktionen der Hardware [4]

Latenzverbergung durch Ausnutzung funktioneller und architektonischer Merkmale des zugrundeliegenden (realen, abstrakten) Prozessors

- Unterbrecherhardware übernimmt die Einplanung/Einlastung von Fäden
- Unterbrechungsprioritäten (IPL) geben die Einplanungsprioritäten von Fäden vor
- Bereitstellung eines Fadens kommt einer Unterbrechungsanforderung (IRQ) gleich
- **Prozessoranforderungen**
  - $num(IPL) \geq num(Threads)$
  - IRQ durch Software auslösbar

  $\Rightarrow$ Infineon TriCore, ARM Cortex-M3

$\Rightarrow$ `http://www4.informatik.uni-erlangen.de/Research/Sloth/`

Latency Awareness in Operating Systems [7]

Latenzgewahrheit in Betriebssystemen für massiv-parallele Prozessoren

Latenzvorbeugung

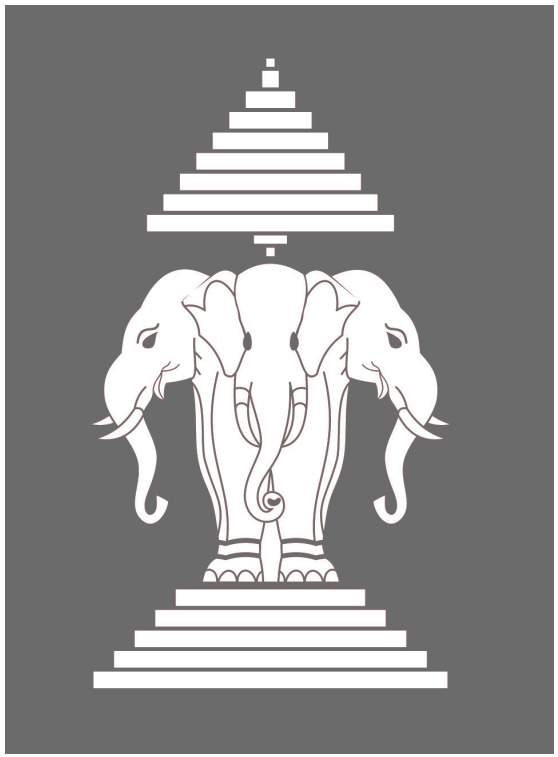
- domänenspezifische Entwurfsmuster
- sperr- und wartefreie Synchronisation

Latenzvermeidung

- Interferenzschutz $\rightsquigarrow$ AOP
- Eindämmung von Wettstreitigkeiten

Latenzverbergung

- asynchrone Prozedurfern-/Systemaufrufe
- Kerne für Betriebssysteme nutzen



$\Rightarrow$ `http://univis.uni-erlangen.de` $\rightarrow$ Forschungsprojekte $\rightarrow$ LAOS

Literaturverzeichnis

[1] Boyd-Wickizier, S. :
Corey: An Operating System for Many Cores.
In: [3], S. 43–57. –
Präsentationsfolien

[2] Boyd-Wickizier, S. ; Chen, H. ; Chen, R. ; Mao, Y. ; Kaashoek, M. F. ; Morris, R. ; Pesterev, A. ; Stein, L. ; Ming Wu nd ; Dai, Y. ; Zhang, Y. ; Zhang, Z. :
Corey: An Operating System for Many Cores.
In: [3], S. 43–57

[3] Draves, R. (Hrsg.) ; Renesse, R. van (Hrsg.):
Proceedings of 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI '08).
Berkeley, CA, USA : USENIX Association, 2008 . –
ISBN 978–1–931971–65–2

[4] Hofer, W. ; Lohmann, D. ; Scheler, F. ; Schröder-Preikschat, W. :
Sloth: Threads as Interrupts.
In: Baker, T. P. (Hrsg.): *Proceedings of The 30th IEEE Real-Time Systems Symposium (RTSS 2009)*.
Washington, DC, USA : IEEE Computer Society, 2009. –
ISBN 978–0–7695–3875–4, S. 204–213

Literaturverzeichnis (Forts.)

[5] Lohmann, D. ; Schröder-Preikschat, W. :
Betriebssysteme.
http://www4.informatik.uni-erlangen.de/Lehre/WS08/V_BS, 2008 *ff.*

[6] Oechslein, B. ; Schedel, J. ; Kleinöder, J. ; Bauer, L. ; Henkel, J. ; Schröder-Preikschat, W. :
OctoPOS: A Parallel Operating System for Invasive Computing.
In: McIlroy, R. (Hrsg.) ; Sventek, J. (Hrsg.) ; Harris, T. (Hrsg.) ; Roscoe, T. (Hrsg.) ; EuroSys (Veranst.): *Proceedings of the International Workshop on Systems for Future Multi-Core Architectures (SFMA'11)* Bd. USB Proceedings EuroSys, 2011 (Sixth International ACM/EuroSys European Conference on Computer Systems (EuroSys'11)), S. 9–14

[7] Schröder-Preikschat, W. :
Latenzgewahrheit in Betriebssystemen für massiv-parallele Prozessoren / DFG.
2011 (SCHR 603/8-1). –
Projektantrag (Normalverfahren)

[8] Ts'o, T. :
Making File Systems Scale: A Case Study Using EXT4.
In: *linux.conf.au 2011*, 2011. –
Präsentation