Marek Olszewski, Jason Ansel, Saman Amarasinghe

Kendo: Efficient Deterministic Multithreading in Software

Computer Science and Articificial Intelligence Laboratory
Massachusetts Institute of Technology

präsentiert von

Robert Harald Ehrenleitner
Paris-Lodron-Universität Salzburg
Naturwissenschaftliche Fakultät
FB Computerwissenschaften

1. Motivation
2. Theorie
   - Deterministische logische Uhr
   - Sperralgorithmen
3. Kendo
   - Implementierung
   - Auswertung

Definitionen

- Ein Prozess (engl. process) ist ein in Ausführung befindliches Programm samt seinen Ressourcen.
- Ein Leichtgewichtprozess (engl. thread) ist ein Arbeitsstrang eines Prozesses. Die Leichtgewichtprozesse teilen sich mit dem Prozess einige Ressourcen.
- Ein Prozess heißt mehrfädig (engl. multithreaded), wenn er mehr als einen gleichzeitigen Arbeitsstrang hat.

Definitionen

- Ein Prozess (engl. process) ist ein in Ausführung befindliches Programm samt seinen Ressourcen.
- Ein Leichtgewichtprozess (engl. thread) ist ein Arbeitsstrang eines Prozesses. Die Leichtgewichtprozesse teilen sich mit dem Prozess einige Ressourcen.
- Ein Prozess heißt mehrfädig (engl. multithreaded), wenn er mehr als einen gleichzeitigen Arbeitsstrang hat.
- Ein Programm heißt deterministisch, wenn es bei jedem Lauf mit denselben Eingaben dieselben Ausgaben liefert.

Wozu deterministische Programmausführung

- Programm muss trotz Überholungen bestimmtes Ergebnis liefern.
- Debuggen von fehlerhaften mehrfädigen Programmen gestatten.
- Mehrfädig erstellte Replikate müssen gleich sein.

Wie deterministische Programmausführung

- Ohne Laufzeitaufzeichnung
- Scheduler soll deterministische Reihenfolge garantieren

Deterministische logische Uhr

Ereignisse in einer parallelen Anwendung mit gemeinsamem Speicher deterministisch ordnen

- Eine Uhr je Leichtgewichtprozess.
- Jeder Leichtgewichtprozess kann die Uhr eines anderen lesen.
- Das Hochzählen einer Uhr hängt niemals von einer anderen ab.
- Dass ein Ereignis früher ist als ein anderes heißt, dass zum Zeitpunkt seines Eintritts die Uhr einen geringeren Wert hat.
- Außerhalb kritischer Abschnitte können die Uhren asynchron laufen.

- Deterministische Reihenfolge der erworbenen Sperren erzwingen.
- Ein Leichtgewichtprozess ist an der Reihe gdw
  - . . . die Uhren aller Leichtgewichtprozesse mit kleinerer ID einen höheren Wert haben;
  - und die Uhren aller Leichtgewichtprozesse mit größerer ID einen höheren oder den gleichen Wert haben.

Einfacher Sperralgorithmus

Sperre setzen

1. Uhr wird angehalten
2. Warten, bis der Leichtgewichtprozess an der Reihe ist
3. Sperre setzen
4. Uhr hochzählen
5. Uhr fortsetzen

Einfacher Sperralgorithmus

Sperre setzen

1. Uhr wird angehalten
2. Warten, bis der Leichtgewichtprozess an der Reihe ist
3. Sperre setzen
4. Uhr hochzählen
5. Uhr fortsetzen

Sperre aufheben

1. Sperre einfach aufheben

Probleme

- Blockiert andere Leichtgewichtprozesse in unabhängigen kritischen Abschnitten
- Verschachtelte Sperren nicht korrekt gehandhabt
- Verklemmungen möglich

Verbesserter Sperralgorithmus

- aktives Warten
  - Höchstens ein Leichtgewichtprozess hat die Sperre
- Erhalten der Sperre kann aus zwei Gründen verfehlen:
  - Ein anderer Leichtgewichtprozess hält die Sperre
  - Die Sperre wurde gerade aufgehoben, aber die Uhr noch nicht hochgezählt

Sperre setzen

1. Logische Uhr anhalten
2. Warten, bis Leichtgewichtprozess an der Reihe ist
3. Sperre anfordern und ggf. setzen
4. Falls 3 gelingt:
    1. Wenn die Uhr des Leichtgewichtprozessses, der die Sperre hatte, einen höheren Wert hat, dann Sperre aufheben.
5. Falls Sperre nicht gesetzt werden konnte, gehe zu 2
6. Uhr hochzählen
7. Uhr fortsetzen

Sperre aufheben

1. Uhr anhalten
2. Wert der Uhr als denjenigen Wert setzen, wann die Sperre aufgehoben wurde
3. Sperre aufheben
4. Uhr hochzählen
5. Uhr fortsetzen

Zusätzliche Optimierungen

- Fairness durch Warteschlange nach verfehlter Zuteilung
- Beim Warten nach verfehlter Zuteilung Uhr „schneller“ hochzählen
- Beim Warten auf dass der Leichtgewichtprozess an die Reihe kommt Uhr hinunterzählen.

Deterministische logische Uhr

- Verwendung des `retired_stores`-Ereignisses von x86-Architekturen.
- Gemeinsame Zugänglichkeit über gemeinsamen Speicher.
- Zwei Paramater der logischen Uhr:
  - `chunk size`, welche die Anzahl der Schreibzugriffe bestimmt, bevor die Uhr hochgezählt wird.
  - `increment amount` gibt an, um wieviel die Uhr hochgezählt wird.

`det_create` – Erzeugung eines Leichtgewichtprozesses

1. Erzeuger wartet darauf, dass er an die Reihe kommt.

`det_create` – Erzeugung eines Leichtgewichtprozesses

1. Erzeuger wartet darauf, dass er an die Reihe kommt.
2. Erzeuger richtet die Strukturen des Erzeugten ein.

`det_create` – Erzeugung eines Leichtgewichtprozesses

1. Erzeuger wartet darauf, dass er an die Reihe kommt.
2. Erzeuger richtet die Strukturen des Erzeugten ein.
3. Kontrolle wird dem Erzeuger entzogen.

`det_lazy_read` – Lesen einer Variable

- Variable wird mit Zeitstempel der logischen Uhr in ein Feld geschrieben.

`det_lazy_read` – Lesen einer Variable

- Variable wird mit Zeitstempel der logischen Uhr in ein Feld geschrieben.
- Beim Lesen wird der Wert mit dem jetzigen Zeitstempel ausgelesen.

Verschaulichendes Beispiel für das Lesen einer Variable

	5	4	3	2	1	0
1						a_0
2					a_0	a_0
3				b_2	a_0	a_0
4			b_2	b_2	a_0	
5		c_4	b_2	b_2		
6	c_4	c_4	b_2			

Welcher Wert wird wann mit welchem Toleranzfenster gelesen?

API

Kendo's Funktionen

`det_create`	...	erzeugt einen Leichtgewichtprozess.
`det_enable`	...	startet logische Uhr.
`det_disable`	...	pausiert logische Uhr.
`det_lazy_init`	...	initialisiert eine Variable.
`det_lazy_read`	...	liest den Wert einer Variable.
`det_lazy_write`	...	schreibt den Wert einer Variable.

Umgebung

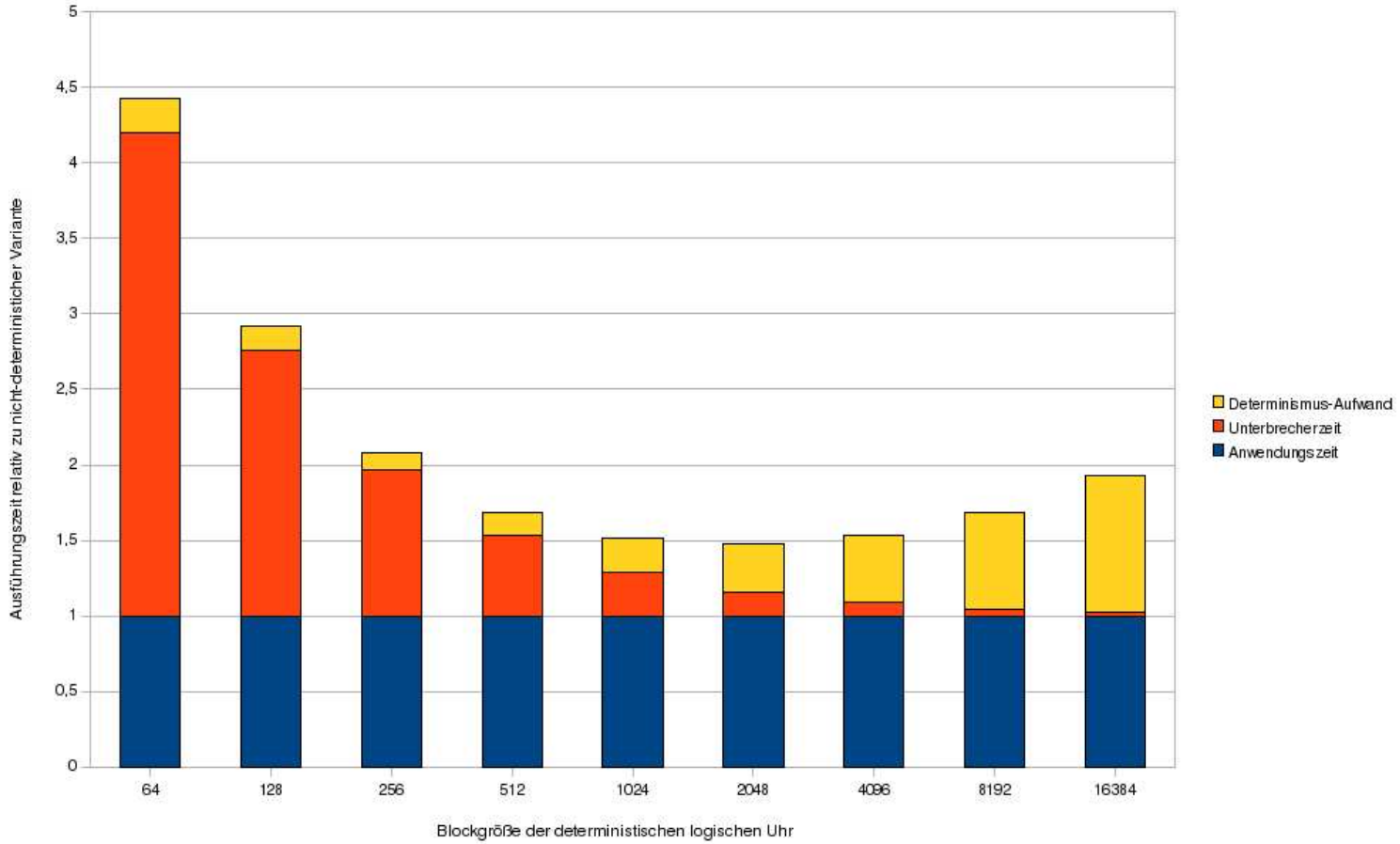
- Intel Core 2, 2,66 GHz, 4-Kern-Prozessor.
- Debian GNU/Linux 2.6.23.
- SPLASH-2-Benchmark-Paket, 2 weitere Tests (insgesamt 9).
- Mittelwert über 10-maliges Laufen.

Umgebung

- Intel Core 2, 2,66 GHz, 4-Kern-Prozessor.
- Debian GNU/Linux 2.6.23.
- SPLASH-2-Benchmark-Paket, 2 weitere Tests (insgesamt 9).
- Mittelwert über 10-maliges Laufen.

Darstellung

- Darstellung als Verhältnis der Laufzeit von Kendo zu jener mit Standard-POSIX.
- Anteile:
  - Reine Anwendungslaufzeit: 100 %.
  - Aufwand der logischen Uhr: 2 %.
  - Synchronisationsverzögerungen: 16 %.


Ausführungszeit relativ zu nicht-deterministischer Variante
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
64
128
256
512
1024
2048
4096
8192
16384
Blockgröße der deterministischen logischen Uhr
Determinismus-Aufwand
Unterbrecherzeit
Anwendungszeit

Danke für die Aufmerksamkeit.