Analyse von Programmspuren: Entwurf und Implementierung eines effizienten Algorithmus zur Datenreduktion

Thesen zur Diplomarbeit von Andreas Knüpfer

Programmspurgrößen wachsen aufgrund des Fortschrittes in der Rechentechnik, der Parallelverarbeitung und dem Performancemonitoring selbst auf Größen an, die für ihre Verarbeitung und Analyse problematisch sind. Eine an die Analyseanforderungen angepasste Datenreduktion kann zur Lösung dieses Problems beitragen. Die vorliegende Arbeit konzentriert sich nur auf Aufruf-Ereignisse innerhalb einer Programmspur. Außerdem bestehen strenge Aufwandsrestriktionen an die Reduktionsalgorithmen, um eine Ausführung zur Trace-Zeit zu ermöglichen.

Um hohe Datenkompressionsraten möglich zu machen, ist sowohl Redundanz (in Form von Wiederholungen) auszunutzen, die verlustfreie Kompression erlaubt, als auch eine Idealisierung der Daten vorzunehmen, die (minimale) Abweichungen zwischen Original und Rekonstruktion verursacht und somit eine verlustbehaftete Kompressionsstrategie verkörpert.

Die Datenreduktion basiert auf einer mehrstufigen Transformation der Ereignisfolge: Eine Sequenz von Ereignissen wird zu einem Meta-Ereignis f zusammengefasst, dem ein Zeitdauervektor zugeordnet ist. Im zweiten Schritt entsteht aus f das Ereignis-Muster g, dem ein (typischer) Zeitdauervektor implizit zugeordnet ist.

Ereignisfolge:	...	e_{i-1}	e_i	e_{i+1}	...	e_{i+n-1}	e_{i+n}	...
		$\downarrow$	$\downarrow$	$\downarrow$		$\downarrow$	$\downarrow$	
Zeitpunktfolge:	...	t_{i-1}	t_i	t_{i+1}	...	t_{i+n-1}	t_{i+n}	...

Ereignisfolge:	...	e_{i-1}	f	e_{i+n}	...
		$\downarrow$	$\downarrow$	$\downarrow$	
Zeitfolge:	...	t_{i-1}	$(t_i; d_1, d_2, ...d_{n-1})^T$	t_{i+n}	...

Ereignisfolge:	...	e_{i-1}	g	e_{i+n}	...
		$\downarrow$	$\downarrow$	$\downarrow$	
Zeitfolge:	...	t_{i-1}	$(t_i, t_{i+n-1} - t_i)$	t_{i+n}	...

Für eine erfolgversprechende Umsetzung sind Strukturmerkmale der Daten auszunutzen:

- **Definition 1 (Stack-Eigenschaft)** *Zu jedem Enter-Ereignis e_E existiert genau ein zugeordnetes Leave-Ereignis $e_L \sim e_E$ und umgekehrt. Dabei gilt $e_E < e_L$ wobei die Relation $<$ die Reihenfolge der Ereignisse beschreibt. Für Ereignisse $e_E \sim e_L$ und $f_E \sim f_L$ gilt darüber hinaus: $e_E < f_E < e_L \Leftrightarrow e_E < f_L < e_L$.*

- **Folgerung 2** *Zwischen zwei einander zugeordneten Ereignissen befinden sich stets ebensoviele Enter-Ereignisse wie Leave-Ereignisse.*

- **Folgerung 3** *Für jedes Ereignis e wird eine **Stufe** $s(e)$ definiert:*

$$s(e) := |\{Ereignis\, f : \exists\, Ereignis\, g \sim f\, mit\, f < e < g\}|\,.$$

  Damit gilt:

$$e \sim d \Rightarrow s(e) = s(d).$$

- **Definition 4** *Ein Meta-Ereignis heißt **Meta-Aufruf**, falls es zu jedem enthaltenen Ereignis e auch das ihm zugeordnete Ereignis $f \sim e$ enthält.*

Für das Kompressionsverhalten kann ein Modell für den Einfluss der Maximalordnung und der Programmspurlänge aufgestellt werden:

$$\mathcal{K} := \frac{L_{Original}}{L_{komprimiert}} = \frac{l_D + l_E \cdot e}{l_D + r \cdot l_{ME} + l_{EM}(N) + s \cdot l_E + k \cdot l_A}$$

- l_D feste Größe für allgemeine Definitions-Records
- $l_E = 18$ Bytes zur Speicherung eines Einzelereignisses
- $l_{ME} = 23$ Bytes für die Definition eines Meta-Ereignisses
- $l_{EM}(n) = 26 + n \cdot 8$ Bytes für die Definition eines Ereignis-Musters
- $l_A = 30$ Bytes für jede Instanz eines Meta-Aufrufes
- e Ereigniszahl, N Maximalordnung für Meta-Ereignisse
- r Anzahl zu definierender Meta-Ereignisse $r_{avg} \approx N, r_{min} \approx \log_2 N$
- s nicht transformierte Ereignisse $s_{avg} = N \cdot \log_2 N, s_{min} \approx 2 \cdot \log_2 N$
- $k = \frac{e-s}{N}$ Wiederholungszahl des maximalen Meta-Ereignisses

Das Modellverhalten wird von einer Reihe von konstruierten Testfällen untermauert. Daran und an 4 realen Beispielen werden auftretende Effekte studiert und erzielbare Kompressionsraten abgelesen:

- die erzielbare Kompressionsrate $\mathcal{K}$ steigt mit der Ereigniszahl e
- Regelmäßigkeiten in Aufrufstruktur und Zeitverhalten führen zu sehr guten Ergebnissen
- hohe Anteile an Nicht-Aufrufereignissen erlauben nur marginale Kompressionsraten

Die Laufzeitkomplexität des Verfahrens verhält sich insgesamt wie $O(e \cdot \log_2 N)$.

Für parallele Programmspuren arbeitet das Verfahren in jedem Prozess separat, d.h. ohne jede Kommunikation - im Anschluss ist eine einfache Anpassung der Meta-Aufrufe und Ereignis-Muster erforderlich.

Neben der reinen Datenreduktion eröffnet die transformierte Darstellung der Programmspur zusätzlich zu den etablierten Ansätzen neue Wege für die Analyse und die Visualisierung:

- Der Test auf gleiches Zeitverhalten kann nun als komponentenweiser Vergleich von Zeitdauervektoren durchgeführt werden, statt als Vergleich einzelner Zeitpunktdifferenzen
  $\Rightarrow$ automatische Berücksichtigung eines Aufruf-Kontextes.
- Korrelation mehrerer gleichzeitig abweichender Komponenten zwischen zwei Zeitdauervektoren untersuchbar
  $\Rightarrow$ Erkennung von Regelmäßigkeiten.
- Gewisse Meta-Aufrufe können in einer Zeitleistendarstellung als „künstliche“ Zustände gezeigt werden, die interaktiv oder adaptiv verscheiden detailliert auflösbar sind
  $\Rightarrow$ Visualisierung auf höheren Abstraktionsniveaus.