Kanal: Ein deklarativer und ereignisgetriebener Ansatz zur ebenenübergreifenden Erhebung und Verarbeitung von Betriebssystemdaten

Alexander Lochmann
alexander.lochmann@tu-dortmund.de

14. Januar 2015

Die Performanz- und Sicherheitsanalyse eines Systems ist eines der Hauptaugenmerke eines Systemadministrators. Dafür ist es nicht selten erforderlich, auf detaillierte Informationen aus dem Betriebssystemkern zurückzugreifen. Die Standardmechanismen zum Debugging, wie auch das Betriebssystemlog, stellen die Informationen nur unzureichend bereit. Daher wurden ereignisgetriebene Werkzeuge entwickelt, die einen tiefen Einblick ermöglichen – bei gleichzeitigem geringeren Einfluss auf die Performance. Der prominenteste Vertreter ist das Werkzeug dTrace aus dem Solaris-Umfeld, das entsprechende Pendant für Linux heißt SystemTap [1, 2]. Beide zeichnen sich durch eine deklarative Beschreibung der Ereignisse aus, die Erhebung der Kontextinformationen erfolgt indes imperativ. Letzteres erlaubt keine automatisierte Analyse, geschweige eine Optimierung der Verarbeitungsschritte. Ferner sind derartige Werkzeuge auch Gegenstand aktueller Forschung, wie anhand von Fay zu sehen ist [3]. Hier wird die Datenerhebung auf einen Cluster ausgeweitet. Es erlaubt zusätzlich die deklarative Verarbeitung der Daten, lässt jedoch die Verknüpfung von Quellen vermissen. Eine weitere Entwicklung stellt das Werkzeug PicoQL dar [4]. Es exportiert die Datenstrukturen des Linux-Kerns als relationale Datenbank und erlaubt so auch die deklarative Verarbeitung von Daten. Es gestattet jedoch nicht den Zugang zu Ereignissen, wie es z.B. SystemTap ermöglicht.

Im Rahmen des Vortrags wird das Werkzeug *Kanal* vorgestellt. Es führt die Ansätze der ereignisbasierten Werkzeuge mit einer deklarativen Datenverarbeitung wie bei PicoQL zusammen. Die Darstellung der Betriebssystemdaten und -ereignisse durch ein dediziertes Modell erlaubt die Abstraktion von Funktionen als Ereignisquellen sowie von Variablen als Entitäten einer relationalen Datenbank. Über ein Metamodell, das als generische Schnittstelle zwischen Datenquelle und *Kanal* dient, wird eine Beschreibung der Daten möglich. Die Quellen werden in sog. *Providern* gekapselt und erweitern das existierende Modell um ihre bereitgestellten Daten. Im Gegensatz zu den bisher genannten Werkzeugen deklariert das sog. Datenmodell explizit zu jedem Ereignis Kontextinformationen, die bei der Erhebung zur Verfügung stehen. Zur Formulierung von Anfragen sowie zur Verarbeitung dieser wird das datenstromorientierte Rahmenwerk CQL eingesetzt [5]. Dies ist das Ergebnis langjähriger Forschung aus dem Bereich der Datenstromverarbeitung und somit bestens für diesen Zweck geeignet [6]. Außerdem bietet *Kanal* Schnittstellen sowohl für die Nutzer- als auch die

Kernebene. Sie ermöglichen, Anfragen zu stellen sowie weitere Datenquellen bereitzustellen. Darüber hinaus wird eine erste Implementierung demonstriert, die es erlaubt Daten sowohl auf der Nutzer- als auch auf der Kernebene zu erheben, zu verknüpfen und zu verarbeiten. Es bietet so Vorteile gegenüber den rein ereignisbasierten bzw. datenbankbasierten Werkzeugen.

Für weitere Arbeiten wird eine automatisierte Analyse und anschließende Optimierung der Anfragen anvisiert. Dies ist für den zukünftigen Einsatzzweck von *Kanal* besonders interessant, da es in ubiquitären Systemen, wie Smartphones, verwendet werden soll. Hier spielt die Energieeffizienz eine besondere Rolle. Neben dem geringen Einfluss auf das Gesamtsystem ist vor allem eine energiesparende Erhebung der Daten wichtig. Durch eine Analyse der Anfragen soll ermittelt werden, welcher Teil der erhobenen Daten mit welcher Genauigkeit benötigt wird, so dass die Quellen möglichst wenig Mehraufwand generieren.

Literatur

[1] Cantrill, Bryan M. ; Shapiro, Michael W. ; Leventhal, Adam H. ; Microsystems, Sun: Dynamic instrumentation of production systems, 2004, S. 15–28

[2] Egiler, Frank C. ; Prasad, Vara ; Cohen, Will ; Nguyen, Hien ; Hunter, Martin ; Keniston ; Chen, Brad: *Architecture of systemtap: a Linux trace/probe tool*. `https://sourceware.org/systemtap/archpaper.pdf`. Version: 2005

[3] Erlingsson, Úlfar ; Peinado, Marcus ; Peter, Simon ; Budiu, Mihai ; Mainar-Ruiz, Gloria: Fay: Extensible Distributed Tracing from Kernels to Clusters. In: *ACM Trans. Comput. Syst.* 30 (2012), November, Nr. 4, 13:1–13:35. `http://dx.doi.org/10.1145/2382553.2382555`. – DOI 10.1145/2382553.2382555. – ISSN 0734–2071

[4] Fragkoulis, Marios ; Spinellis, Diomidis ; Louridas, Panos ; Bilas, Angelos: Relational Access to Unix Kernel Data Structures. In: *Proceedings of the Ninth European Conference on Computer Systems*. New York, NY, USA : ACM, 2014 (EuroSys '14). – ISBN 978–1–4503–2704–6, 12:1–12:14

[5] Arasu, Arvind ; Babu, Shivnath ; Widom, Jennifer: The CQL Continuous Query Language: Semantic Foundations and Query Execution. In: *VLDB Journal* (2006), June. `http://research.microsoft.com/apps/pubs/default.aspx?id=77607`

[6] *Stanford Stream Data Manager*. `http://infolab.stanford.edu/stream/`