DISS. ETH NO. 20930

TACKLING OS COMPLEXITY WITH DECLARATIVE TECHNIQUES

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

ADRIAN LAURENT SCHÜPBACH

Master of Science ETH in Computer Science, ETH Zurich

19. April 1981

citizen of Landiswil, BE

accepted on the recommendation of

Prof. Dr. Timothy Roscoe
Prof. Dr. Gustavo Alonso
Prof. Dr. Hermann Härtig

2012

Kurzfassung

Diese Dissertation zeigt, dass die erhöhte Betriebssystemkomplexität, die durch die Notwendigkeit entsteht, sich an eine grosse Anzahl unterschiedlicher Rechnersysteme anzupassen, mittels deklarativer Techniken signifikant reduziert werden kann.

Moderne Hardware ist zunehmend unterschiedlich und komplex. Es ist wahrscheinlich, dass sich diese Entwicklung in Zukunft fortsetzt. Diese Entwicklung erschwert den Betriebssystembau. Betriebssysteme müssen sich der Rechnerarchitektur optimal anpassen. Sie müssen den gesamten Funktionsumfang des Rechners ausschöpfen, um die volle Leistung des kompletten Systems zu gewährleisten. Vom Betriebssystem ungenützte, suboptimal genützte oder gar falsch genützte Rechnerfunktionalität führt zu geringerer Leistung des Gesamtsystems. Traditionelle Betriebssysteme passen sich durch vorgefertigte Regeln, die im ganzen Betriebssystem verteilt und mit der eigentlichen Betriebssystemfunktionalität vermischt sind, der Rechnerarchitektur an.

In dieser Arbeit argumentiere ich, dass es aus zwei Gründen nicht mehr möglich ist, vorgefertigte Regeln für eine Anzahl bekannter Rechnerarchitekturen mit der Betriebssystemfunktionalität zu vermischen. Erstens garantieren vorgefertigte Regeln nicht, dass alle Rechnerfunktionen vollständig ausgeschöpft werden. Zweitens bedeutet das, dass die Regeln für jede neue Rechnerarchitektur angepasst werden müssen, wobei die Regeln für bisherige Rechnerarchitekturen beibehalten werden müssen. Dies führt zu erheblicher Betriebssystemkomplexität und schliesslich zu einem enormen Anpassungsaufwand. Um dies zu vermeiden, muss das Betriebs-

system während der Laufzeit Wissen über die Rechnerarchitektur aufbauen und daraus, durch logische Schlussfolgerungen, die bestmöglichen Anpassungsregeln ableiten. Dies führt zu einfacheren, verständlicheren, pflegeleichteren und leichter anpassbaren Betriebssystemfunktionen und stellt sicher, dass die Rechnerfunktionalität vollständig ausgeschöpft wird.

Der Wissensaufbau und das Ableiten von Anpassungsregeln durch logische Schlussfolgerungen sind mit hoher Programmierkomplexität verbunden. Dies gilt insbesondere, wenn dafür maschinennahe Programmiersprachen, wie zum Beispiel C, verwendet werden. Deklarative Techniken erlauben hingegen, angestrebte Regeln durch eine einfache und verständliche Beschreibung der gewünschten Art der Anpassung, basierend auf Wissen über die Rechnerarchitektur, abzuleiten. Durch die natürliche Beschreibung in höheren Programmiersprachen wird die Programmierkomplexität stark verringert.

Um den Vorteil deklarativer Techniken im Zusammenhang mit Komplexität und Anpassungsfähigkeit in Betriebssystemen zu beweisen, stelle ich in dieser Dissertation verschiedene Fallstudien vor, die, basierend auf deklarativen Techniken, Regeln für die Anpassung an die Rechnerarchitektur, mittels logischer Schlussfolgerungen, ableiten. Die Fallstudien setzen kein Wissen über die Rechnerarchitektur voraus, sondern eignen sich dies während der Laufzeit an.

Das Wissen wird in einem zentralen Wissensdienst des Betriebssystems aufgebaut. Regeln werden in diesem Wissensdienst durch logische Schlussfolgerung abgeleitet. Dadurch, dass die Fallstudien, und somit die verschiedenen Betriebssystemkomponenten, diesen Wissensdienst benützen können, wird ihre Komplexität nochmals deutlich verringert. Es ist somit nicht nötig, dass sich jede einzelne Betriebssystemkomponente mit der Wissensgewinnung und der Ableitung von Regeln beschäftigt. Mit dieser Implementation beweise ich die praktische Anwendbarkeit deklarativer Techniken in Betriebssystemen.

Abstract

This thesis argues that tackling the increased operating systems complexity with declarative techniques significantly reduces code complexity involved in adapting to a wide range of modern hardware.

Modern hardware is increasingly diverse and complex. It is likely that this trend continues further. This trend complicates the operating system's construction. Operating systems have to adapt to the hardware architecture and exploit all features to guarantee the best possible overall system performance. Not exploiting all hardware features, or using them in a suboptimal or even wrong way, results in lower overall system performance. Traditionally, operating systems adapt to the underlying architecture by predefined policies, which are intermingled with the core operating system's functionality.

In this thesis I argue that for two reasons it is no longer possible to encode predefined policies for a set of known hardware architectures into the operating system. First, predefined policies do not automatically guarantee that hardware features are fully exploited on all hardware platforms. Second, for this reason, predefined policies would need to be ported to many different hardware platforms, while, at the same time, it would be necessary to keep the policies suitable for older platforms. This leads to a significant complexity of operating systems and finally to a high engineering effort, when porting the operating system to new hardware platforms. To avoid this problem, the operating systems must gain hardware knowledge at runtime and derive policies suitable for the current architecture through online reasoning about the hardware. This leads to operating sys-

tems code that is simpler, better understandable, more maintainable and easier to port, while ensuring that the operating system exploits the hardware features as best as possible.

Reasoning about hardware and deriving policies is a complex task. This is especially the case, if low-level languages like C are used. Instead, declarative techniques allow deriving policies through a simple description of how to adapt to the hardware based on hardware knowledge gathered at runtime. The natural description in a high-level declarative language reduces code complexity significantly.

To prove the usefulness of declarative techniques in the context of adaptability of operating systems and handling of complexity, I present several case studies in this thesis. The case studies are based on declarative techniques. They reason about hardware and derive policies based on hardware knowledge. The case studies do not assume any a priori knowledge about the current hardware platform. Instead, they gain knowledge at runtime by online reasoning about the hardware.

A central knowledge service stores hardware knowledge and allows the operating system and applications to derive policies according to declarative rules. Because the case studies, and therefore the operating system components, can use the central service, their complexity is again reduced significantly. It is not necessary, that every single component deals with knowledge gathering and deriving policies by itself. It pushes this part to the knowledge service. With this implementation I prove the practical feasibility of applying declarative techniques in real operating systems.