The Structure of a Type-Safe Operating System

Der Technischen Fakultät der
Universität Erlangen-Nürnberg
zur Erlangung des Grades

DOKTOR-INGENIEUR

vorgelegt von

Michael Golm

Erlangen - 2002

Als Dissertation genehmigt von
der Technischen Fakultät der
Universität Erlangen-Nürnberg

Tag der Einreichung:	16. 09. 2002
Tag der Promotion:	17. 12. 2002
Dekan:	Prof. Dr. rer. nat. A. Winnacker
Berichterstatter:	Prof. Dr. rer. nat. F. Hofmann

Michael Golm

The Structure of a Type-Safe Operating System

The architecture of traditional operating systems relies on address-based memory protection. To achieve flexibility at a low cost operating system research has recently started to explore alternative protection mechanisms, such as type safety. This dissertation presents an operating system architecture that completely replaces address-based protection with type-based protection. Replacing such an essential part of the system leads to a novel operating system architecture with improved robustness, reusability, configurability, scalability, and security.

The dissertation describes not only the design of such a system but also its prototype implementation and the performance of initial applications, such as a file system, a web server, a data base management system, and a network file server. The prototype, which is called JX, uses Java bytecode as its type-safe instruction set and is able to run existing Java programs without modifications.

The system is based on a modular microkernel, which is the only part of the system that is written in an unsafe language. Light-weight protection domains replace the heavy-weight process concept of traditional systems. These domains are the unit of protection, resource management, and termination. Code is organized in components that are loaded into domains. The portal mechanism—a fast inter-domain communication mechanism—allows mutually distrusting domains to cooperate in a secure way.

This dissertation shows that it is possible to build a complete and efficient general-purpose time-sharing operating system based on type safety.

Part of the material presented in this dissertation has previously been published in the following conference proceedings:

Michael Golm, Meik Felser, Christian Wawersich, and Jürgen Kleinöder. **The JX Operating System**. In *Proc. of the USENIX Annual Technical Conference*, Monterey, CA, USENIX Association, pp. 45-58, June 10-15, 2002.

Michael Golm, Jürgen Kleinöder, and Frank Bellosa: **Beyond Address Spaces - Flexibility, Performance, Protection, and Resource Management in the Type-Safe JX Operating System**. In *Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)* , Schloß Elmau, Germany, IEEE Computer Society Press, pp. 1-6, May 20-23, 2001.

Meik Felser, Christian Wawersich, Michael Golm, and Jürgen Kleinöder. **Execution Time Limitation of Interrupt Handlers in a Java Operating System**. In *Proc. of the 10th ACM SIGOPS European Workshop*, Saint-Emilion, France, Sept. 2002.

Michael Golm and Jürgen Kleinöder. **JX: Eine adaptierbare Java-Betriebssystemarchitektur**. *GI-Herbsttagung: Mobile Computing*, Munich, Germany, November 16-17, 2000.

Contents

List of Figures

Chapter 1: Introduction

Chapter 2: Background

Chapter 3: Architectural Overview

Chapter 4: Domains

Chapter 5: Portals

Chapter 6: Components

Chapter 7: Processor Scheduling

Chapter 8: Memory Management

Chapter 9: Device Drivers

Chapter 10: Security Architecture

Chapter 11: Related Work

Chapter 12: Evaluation

Chapter 13: Conclusions

List of Tables

Chapter 1: Introduction

Chapter 2: Background

Chapter 3: Architectural Overview

Chapter 4: Domains

Chapter 5: Portals

Chapter 6: Components

Chapter 7: Processor Scheduling

Chapter 8: Memory Management

Acknowledgements

Building a complete operating system is nothing one person can accomplish alone. Many people have contributed to the JX project. JX grew out of my master's thesis about a meta architecture for Java, called MetaJava and later metaXa [73]. For this thesis I built a JVM and obtained first experiences with JVM internals. After I finished this thesis in 1997 I was employed as a research and teaching assistant at the Department of Computer Science (Operating Systems) of the University of Erlangen. Giving lectures and seminars allowed me to tell students about my ideas and convince them to do their thesis work in the context of the JX project. The first of those students was Hans Kopp who built a bytecode-to-x86 translator for metaXa as part of his master's thesis in 1998 [106]. Hans' compiler did considerably improve the performance of metaXa and using it as the foundation of an operating system became practical. At the end of 1998 I modified metaXa to run on the bare hardware and called it metaXaOS. At the same time I started to implement a new virtual machine, called JX, that provided multiple protection domains. In 1999 Andreas Weissel ported the IDE driver and the ext2 file system from Linux to metaXaOS [186]. Markus Meyerhöfer ported the driver for the 3Com 3C905B network interface card from Linux to JX and started to implement a TCP protocol stack [127]. In 2000 Christian Wawersich and Andreas Heiduk started to work on their master's thesis. Christian restructured Hans' translator and improved the quality of the generated machine code [185]. Christian's compiler allowed to run JX operating system components nearly as fast as an operating system written in C. Andreas wrote a driver for the Matrox G200 video card and the Brooktree BT878 frame grabber device [88] and a fascinating demo - an interactive puzzle of a live video source. At the end of 2000 Martin Alt and Meik Felser started their master's thesis. Martin wrote a bytecode verifier that performed the standard bytecode verification but could additionally be used to verify worst case execution times [5]. Using Martin's verifier we were able to guarantee an upper bound for response times without manually inspecting each interrupt handler. Meik focused on scheduling and multiprocessor support [63]. He ported JX to an SMP architecture and designed and implemented an interface to use Java schedulers. In 2001 Jürgen Obernolte, Christian Flügel, and Ivan Dedinski started to work on their semester thesis. Jürgen implemented a window manager [138]. Christian implemented a client for the Remote Desktop Protocol (RDP) [65] and Ivan implemented a database system [47]. Jörg Baumann restructured the garbage collector to allow domain-specific garbage collection and improved the performance visualization system. In 2002 Marco Winter implemented the AWT based on Jürgen's window manager [188]. I am grateful for the opportunity to work with all these talented students. Without them I would not have been able to prove that my design leads to a usable operating system.

I am deeply indebted to Jürgen Kleinöder who supported me from the beginning of the project. I thank Franz Hauck and the other members of the Ergoo group for the critical comments after my presentation dry-runs. This lead to considerable improvements of the final presentations. I thank Fridolin Hofmann and Bernd Hindel for referring my thesis and providing valuable comments and Wolfgang Schröder-Preikschat and Heinz Gerhäuser for being members of my committee.

Finally I owe thanks to my parents, my wife Anja, and my children Tobias and Rebecca for their patience and constant support during the often exhausting process of writing this dissertation.