Konzepte von Betriebssystem-Komponenten

Thema: Research OS - JX

Michael Schmidt
(swmeschm@cip.informatik.uni-erlangen.de)

Inhalt

- Was ist JX, warum wurde es entwickelt?
- Systemaufbau
- Performance-Vergleich
- Zusammenfassung

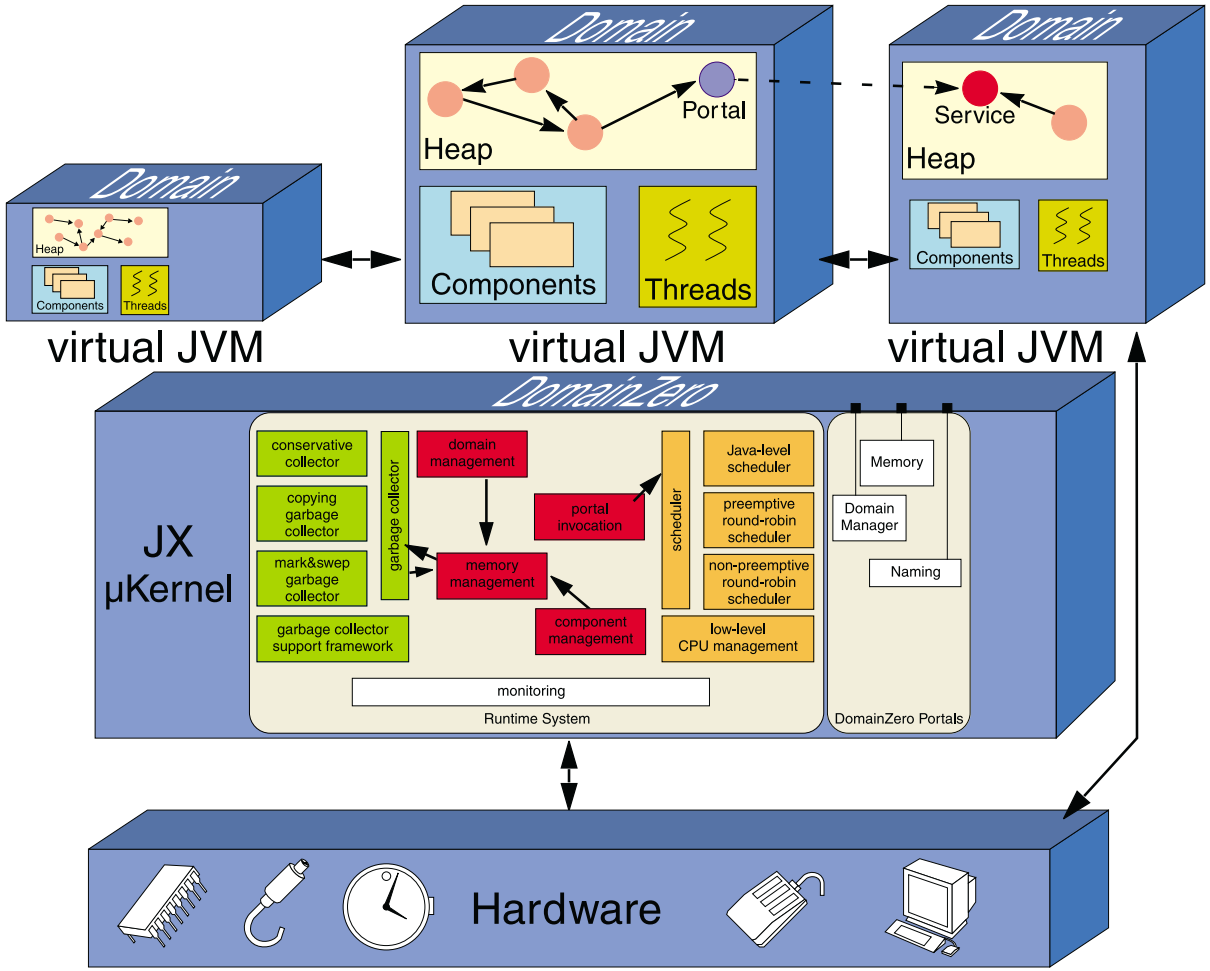
Was ist JX?

- Single-Address-OS
- Microkernel
- Domains
  - Einheit der Datenkapselung
  - Java-Code

Ziele

- Betriebssystem auf Typsicherheit: Java
- anpassungsfähig
- flexibel
- hohe Robustheit
- Anwendungsgebiet: Embedded Systems, PC

Aufbau des Betriebssystemes



DomainZero / Kern

- Microkernel
  - Assembler und C Code, steht zur Laufzeit fest
- Schnittstelle für System Calls
- Initialisierung System
- Kontextumschaltung
- Grunddienste

Domains

- “Container” zur Datenkapselung
- 100%-Java-Code
- Objekte in einer Domain:
  - Java-Klassen
  - Heap
  - Threads
  - Portale
  - Interrupt-Handler

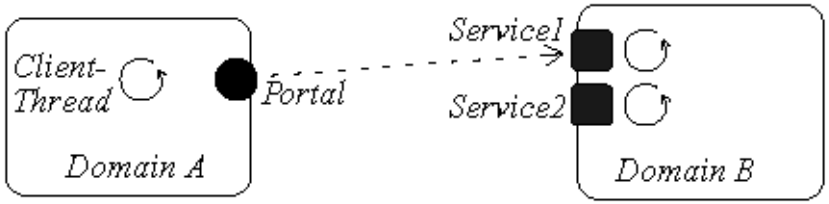
Domains (2)

- In jeder Domain können eigene
  - Scheduler
  - Garbage Collection
  - Speichermanager

  implementiert werden

Portale

- IPC
- Bestandteile: Schnittstelle, Dienstthread, Dienstzähler, Dienstobjekt



- Arten der Kommunikation/Parameterübergabe: Intradomain ↔ Interdomain
- Portale als Parameter

Fast Portal

- Optimierte Portale
- Unterschiedlicher Ausführungskontext
- Anwedungsbeispiele
  - yield()-Methode
  - Erstellung Memory Objekte

JX-Translator

- Einhaltung von Speichergrenzen wichtig
- Java-Bytecode
  - plattformunabhängig
  - Einhaltung Java-Regeln
  - Verifizierung zur Laufzeit
- → Typsicherheit

JX-Translator (2)

- JX-Translator
  - Sicherheitsprüfungen
  - Optimierungen
  - JVM-Speicherschutz zur Laufzeit
- Garbage Collection, DMA
  - DMA: Zugriff auf Speicher anderer Domain
  - Plugins: Maschinenbefehle
  - Plugins: kein Speicherschutzkonzept → trusted

Memory Objekte

- Verwaltung großer Datenmengen: Array in Java
  - kein Zugriffsschutz
  - Ausgliederung Teilbereiche nicht möglich
- Memory Objekte in JX
  - Erstellung: fast portal
  - Parameterübergabe als Portal
  - Zugriff: Methodenaufruf (Plugins)

Memory Objekte (2)

- Vermeidung Speicheroverhead
- Memory sharing leicht möglich
  → Grundlage für wichtige Systemdienste:
  - Netzwerk-Stack
  - Dateisystem

Interrupt-Handling

- zweistufiges Konzept
  - handleInterrupt-Methode eines Interrupthandlers
  - Zeitabschätzung der Laufzeit: lineare Anweisungen in handleInterrupt-Methode ↔ Stillstand des Systems
  - Interrupt-Thread (Interrupts deaktiviert) (1. Stufe)
  - asynschrone Abarbeitung: Geräteinterrupt benachrichtigen, asynchroner Thread zur Abarbeitung (2. Stufe)

Performance

- Domains co-located
- Interprozesskommunikation

IPC-Performance im Vergleich

System	IPC Zyklen
L4KA (PIII 450)	800
Fiasco/L4 (PIII 450 MHz)	2610
J-Kernel (Pentium Pro 200 MHz)	440
JX (hosted, Linux 2.2.14, PIII 500MHz)	7100
JX (native, PIII 500MHz)	650

- Dateisystem-Durchsatz
  - JX: 201 Mbyte/sec, Linux: 400 MByte/sec
  - → Optimierung (Compiler)

Zusammenfassung

- Ziele wurden erreicht
  - flexibel, offen, performant
- höhere Robustheit: Java ↔ C, Assembler
- natives Java, keine Spracherweiterungen