Objective of the JX system

- Make writing an OS as easy as writing applications
  - simple and robust architecture
- Dynamic OS extension with untrusted components
  - exact resource accounting and customizable management
- Code reuse
  - tailored OS configurations; dedicated systems
- Protection
  - flexibility
  - performance
  - robustness

Outline

- Objectives
- Architecture
- Protection
- Flexibility
- Performance

Single Address Space
Domains

- Domain is unit of
  - protection
  - resource management
  - fault containment
  - termination

Objects & Heap

- No shared objects
  - decoupled garbage collection
  - no uncontrolled information flow

Components

- Components contain 100% Java
  - Bytecode is verified by an extended verifier
  - All components are compiled to native code at load time
  - DomainZero contains core written in C and assembler

Threads

- No shared threads
  - decoupled CPU scheduling
  - two-level scheduling
Communication: Portals

- Domain can export a service
- Associated with interface component and thread

Communication: Memory

- Advantages
  - access control possible
  - can create subranges
  - revocation
  - Zero-copy components (e.g., network stack)
**Device Driver**

- JX Architecture

**Protection: Device Driver**

- JX protection is based on type safety and portals
- Some domains can circumvent these mechanisms
  - DomainZero, Translator, Verifier → Trust
  - (some) device drivers

**Protection: Interrupt Handler**

- Interrupts on the interrupted CPU are blocked during execution of the interrupt handler
- Verifier checks interrupt handler for upper limit of execution time
  - can insert runtime checks to ensure timely termination
  - runtime check can terminate interrupt handler and initiate counter measure (e.g., switch off device interrupts)

**Building an OS: A Dedicated System**

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System

- Building an OS: A Dedicated System
Performance

- Hardware:
  500MHz PIII, 128MB RAM,
  IDE: Maxtor 91303D6, 12427MB, 512kB Cache
  NIC: 3C905B 100 MB/s

- IPC:
  - Portal round trip 650 cycles
  - L4 (including RPC stubs): 800 cycles
  - KaffeOS: 27270 cycles

Performance

- Iozone-like benchmark: re-read of a 512 kB file

Performance

- JX as NFS server: `getattr` request rate
Performance:
JX Advantages and Limitations

- Advantages:
  - No expensive border crossings (JNI, OS border)
  - Safe inlining of OS-level code into application code
  - Avoid locking in favour of specialized scheduling

- Limitations:
  - Unavoidable safety checks
  - Semantic gap between stack machine and register machine

Conclusion

- Single Address Space
- Full Protection
  - completely decoupled domains
  - fast communication using portals or memory objects
- Reusable components
- Dynamic extensibility
- Good performance

→ [http://www4.cs.fau.de/Projects/JX/](http://www4.cs.fau.de/Projects/JX/)

Status

- JX runs on off-the-shelf PCs
- Drivers for: IDE, Matrox G200, 3COM 3C905B, BT848
- Ext2-FS
- UDP, TCP, RPC (client), NFS (client)
- SMP support

Component Interaction

<table>
<thead>
<tr>
<th></th>
<th>co-located components</th>
<th>dis-located components</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter passing</td>
<td>by reference possible</td>
<td>parameter objects must be copied</td>
</tr>
<tr>
<td>thread</td>
<td>can execute in same thread</td>
<td>threads must be switched</td>
</tr>
<tr>
<td>resources</td>
<td>caller can be trusted to carefully use resources (e.g., Memory objects)</td>
<td>access rights for Memory objects must be restricted</td>
</tr>
</tbody>
</table>

- Scheduling: scheduling strategy of all co-located components must be compatible
- Execution engine of co-located components must be identical (translated or interpreted)
Memory Mapping

- Performance problem: range check
- (partial) solution: map objects to memory range

Memory

Scheduling

- Performance problem: range check
- (partial) solution: map objects to memory range
Scheduling
Domain Zero
Assembler
Domain A
Components
Classes
Threads
Java Stacks Thread
Control Blocks
CPU
Heap
Domain B
Components
Classes
Threads
Java Stacks Thread
Control Blocks
Heap
Domain C
Components
Classes
Threads
Java Stacks Thread
Control Blocks
CPU
Heap
Low Level Scheduler
Domains
High Level Scheduler
Threads
High Level Scheduler B
Threads
High Level Scheduler C
Threads
Resource Management
Physical resources (CPU, Memory)
- resource management is supported by Domain Zero (but no policy)
Device-specific physical resources
- (e.g. network bandwidth)
Virtual resources
- managed by their respective domains (e.g. TCP port numbers, files)

Zero-copy using Memory
- immediate processing or buffering possible

upstream:

mem1 = receive(mem0)

mem0 = processMemory(mem1)

mem1 = DMA

mem0 = processMemory(mem1)

mem0 = processMemory(mem1)

mem0 = processMemory(mem1)

DMA

NIC

Ether IP UDP

Ether IP UDP

Ether IP UDP

Ether IP UDP

mem0 = processMemory(mem1)