Extensibility, Safety and Performance in the SPIN Operating System

Brian Bershad, Stefan Savage, Przemyslaw Pardyak, Emin Gun Sirer, Marc E. Fiuczynski, David Becker, Craig Chambers, Susan Eggers

Department of Computer Science and Engineering
University of Washington
Goals

- **Extensibility**
  - Applications can dynamically extend system to provide specialized services

- **Safety**
  - Kernel is protected from actions of extensions

- **Performance**
  - Extensibility and safety have low cost
Why is this hard?

Can we have all three in a single operating system?
Approach

- Put extension code in the kernel
  - Cheap communication
- Use language protection features
  - Static safety
- Dynamically interpose on any service
  - Fine-grained extensibility
A SPIN extension

Application

User
Kernel

VM Fault

Application Extension

ProtectFault()  UnprotectPage()

SPIN MMU Services
SPIN structure

User

Kernel

Application Extensions

Shared Extensions

SPIN Core Services

OSF/1 Unix server

Unix Apps

Video Server

Web Server

Applications

Mach API

Threads

Unix API

Net Video

HTTP

Syscall

Process

Network

File Sys

Execution State

Memory

Devices

Extension Services

Applications

Extension
Services
Safety
Language-based protection

Modula-3

- Memory safe
- Interfaces for hiding resources
- Cheap capabilities
Restricted dynamic linking

Goal: control access to interfaces cheaply
Strategy: restrict access at dynamic link-time

Extension A

Domain Z

Extension B

Service Code
Extensibility
Event-based communication model

Dispatcher
Using Events

INTERFACE Network;
PROCEDURE PacketArrived(p: Pkt);
END Network.

EVENT definition

MODULE EthernetDriver;
PROCEDURE Interrupt(p: Pkt) =
BEGIN
    Network.PacketArrived(p);
END Interrupt;

Event raise
Other services

- Almost all “system” services are extensions
  - Network protocols
  - File systems
  - System call interface

- SPIN only implements services which cannot be safely implemented as extensions
  - Processor execution state
  - Basic interface to MMU and physical memory
  - Device IO/DMA
  - Dynamic linker and Dispatcher
A protocol graph in SPIN

- UDPrv
- Active messages
- RPC
- Video
- TCP recv
- ICMP count
- Ping
- HTTP
- TCP port 80 packet arrived

- UDP packet arrived
- TCP packet arrived
- ICMP packet arrived

- IP count
- Ether count
- Ether packet arrived
- IP packet arrived
- IP

- Ethernet driver
Design summary

● Safety
  – Memory safe language for extensions
  – Link-time enforcement for access control

● Extensibility
  – Fast and safe centralized control transfer switch

● Result
  – Allows fast and safe fine-grained service extension
Performance
Platform

- **SPIN runs on DEC Alpha platforms**
- **Measurements**
  - DEC AXP 3000/400 @ 133Mhz
- **Comparison systems**
  - DEC OSF/1 V2.1
  - Mach 3.0
SPIN performance advantages

- Extensions provide specialized service
  - Don’t execute unnecessary code
- Extensions close to kernel services
  - Low latency response to faults/interrupts
  - Invoking services is cheap
Per-port TCP packet forwarding

TCP packets in → HTTP Server → TCP packets out

Time in microseconds

<table>
<thead>
<tr>
<th>DEC OSF/1</th>
<th>SPIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>ATM</td>
</tr>
</tbody>
</table>

---

0 500 1000 1500 2000 2500 3000

Ethernet

ATM
Video service

- **DEC OSF/1**
- **SPIN**

![Graph showing the relationship between number of video streams and percent CPU utilized.](image)
Other basic system services

![Bar chart showing time in microseconds for Fork/Join and Protection Fault across three systems: DEC OSF/1, Mach, and SPIN.](chart)

- **Fork/Join**
  - DEC OSF/1: Approx. 1500 microseconds
  - Mach: Approx. 300 microseconds
  - SPIN: Approx. 100 microseconds

- **Protection Fault**
  - DEC OSF/1: Approx. 400 microseconds
  - Mach: Approx. 400 microseconds
  - SPIN: Approx. 100 microseconds
Conclusions

- It is possible to combine extensibility, safety and performance in a single system.
- Static mechanisms, implemented through the compiler, make this possible.
Language-based capabilities

INTERFACE PageTable;
TYPE T <: REFANY;

PROCEDURE New(): T;
END PageTable.

INTERFACE PageTableInternal;
REVEAL PageTable.T =
  BRANDED REF RECORD
  PTBase: ADDRESS;
  ...
  END;
END PageTableInternal.

t := PageTable.New();
Event implementation

Use procedure call to define and invoke events

– Convenient syntax
– High performance implementation for common case
– Can protect events using domains
– Most procedures in the system can be extended
Protected communication

Time in microseconds

- Protected Call
- System Call
- IPC

- DEC OSF/1
- Mach
- SPIN

Time in microseconds:
- Protected Call: 0.13
- System Call: 845
Memory management services

Time in microseconds

DEC OSF/1
Mach
SPIN

Fault
Trap
Prot-1
Prot-100
Unprot-100
Appel1
Appel2
Modifications to Modula-3

- Memory safe cast
  - VIEW operator
- Procedures which may be terminated
  - EPHEMERAL procedure type
- Naming code
  - INTERFACE UNIT, MODULE UNIT
- Universal procedure type
  - PROCANY reference type
Performance of M3 vs C

- Most operations are compiled equivalently whether written in M3 or C
- M3 can sometimes introduce runtime checks to guarantee type safety

MD5 checksum benchmark

<table>
<thead>
<tr>
<th></th>
<th>Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRC M3</td>
<td>15</td>
</tr>
<tr>
<td>Vortex M3</td>
<td>10</td>
</tr>
<tr>
<td>GCC</td>
<td>5</td>
</tr>
</tbody>
</table>