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

SPIN MMU Services

ProtectFault() UnprotectPage()
SPIN structure

User

OSF/1 Unix server
Unix Apps
Video Server
Web Server

Kernel

Mach API
Threads
Unix API
Net Video
HTTP
Syscall
Process
Network
File Sys

Applications

Application Extensions

Shared Extensions

SPIN Core Services

Execution State
Memory
Devices
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

Extension B

Domain Z

Service Code
Extensibility
Dispatcher

Event-based communication model
Using Events

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

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

Event definition

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

UDP packet arrived

TCP port 80 packet arrived

IP packet arrived

Ether packet arrived

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

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

0 500 1000 1500 2000 2500 3000
Video service

![Graph showing the relationship between the number of video streams and percent CPU utilized. The graph includes two lines: one for DEC OSF/1 and another for SPIN. The x-axis represents the number of video streams, ranging from 0 to 15, while the y-axis represents percent CPU utilized, ranging from 0 to 45. The DEC OSF/1 line starts at the origin and shows a positive linear trend, while the SPIN line starts at a lower point and also shows a positive linear trend.](image_url)
Other basic system services

- **Fork/Join Protection**
  - DEC OSF/1: 1200 microseconds
  - Mach: 300 microseconds
  - SPIN: 50 microseconds

- **Protection Fault**
  - DEC OSF/1: 400 microseconds
  - Mach: 300 microseconds
  - SPIN: 50 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

![Bar chart showing time in microseconds for different calls and IPCs:]

- **Protected Call**: 0.13 microseconds
- **System Call**: DEC OSF/1
- **IPC**: 845 microseconds

Legend:
- Blue: DEC OSF/1
- Red: Mach
- Gray: SPIN
Memory management services

Time in microseconds

- DEC OSF/1
- Mach
- SPIN

Categories: 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