

Extensibility, Safety and Performance in the *SPIN* Operating System

Brian N. Bershad, Stefan Savage, Przemyslaw Pardyś, Emin Gun Sirer, Marc
Fiuczynski, David Becker, Susan Eggers, Craig Chambers
Department of Computer Science and Engineering
University of Washington, Seattle, WA
1995

Presented by: Nguyet Minh Nguyen
October 18, 2006

Content

- Introduction (SPIN, Goals, Ideas)
- Architecture (Protection model, Extension model, Core extensible services, Building services with SPIN)
- Performance (Microbenchmarks, Networked VS)
- Conclusions (Critiques, Related works)
- Discussion

2

SPIN

- An Extensible Operating System
- Motivation
 - Support high performance applications
 - High performance and functionality demands
 - Poorly matched by traditional operating systems

3

SPIN Goals

- Extensibility
 - Extensible infrastructure
 - Fine-grained access to system resources and functions
- Safety
 - Access is controlled at the same granularity
- Efficiency
 - Overhead of both protection and access is low

4

SPIN Ideas

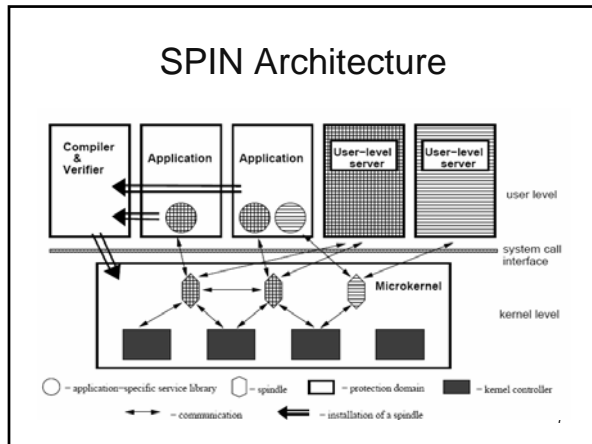
- Using language and runtime services to provide low-cost, fine-grained, protected access to system resources
- Four design techniques
 - Co-location
 - Type safety
 - Logical protection domains
 - Dynamic binding
- System service interfaces

5

System Overview

- Written in **Modula-3**
 - Extensions to be easily integrated
- Using language services to provide *safe extensibility* within the kernel
- Only code that requires *low-latency access* to system services *must* be written in the system's safe extension language

6



- ### SPIN Protection Model
- **Goal:** control the set of operations that can be applied to resources
 - Capability-based protection system
 - Protection domains

- ### Capabilities
- **Definition:** A secure reference to a resource
 - All kernel resources are referenced by capabilities
 - **Implementation:** a type safe pointer to an abstract data type
 - Can be passed from the kernel to user-level applications as *externalized references*
 - **Key point:** Capabilities refer directly to the underlying resources which they name

- ### Protection Domains
- Names of symbols and types are resources
 - Domains: separate entities of symbols
 - Operations
 - Create: create a new domain
 - Resolve: dynamic linking
 - Domain capabilities
 - Modify: domain as the target of a Resolve operation
 - Import: domain as the source of a Resolve operation

- ### SPIN Extension Model
- **Goal:** enable extensions to be specialized with the granularity of a procedure call
- Event-based invocation mechanism*
- **Events:** “hooks” on which applications can attach extensions
 - **Event handler:** executed in response to a specific event
 - **SPIN dispatcher:** event delivery, provides for indirect invocation and multicast

- ### SPIN Extension Model (cont.)
- Extensions can install a handler on an event
 - Call the dispatcher with the name of the event and the handler
 - Event names are protected by the domain machinery
 - Events are exported through interfaces as procedure signatures or procedures
 - Raising and handling of an event can be implemented with a direct procedure call

SPIN Extension Model (cont.)

- Guard
 - Arbitrary predicate evaluated by the dispatcher to invoke the handlers
 - Finely restrict access to events
- Event properties
 - Bounded by a time quantum
 - Asynchronous
- Locating an interface
 - Publish and subscribe interface

13

SPIN Core Services

- *Trusted*
- Statically linked into the kernel
- Interfaces are extensible

Memory Services
Processor Services

14

SPIN Memory Services

- Physical storage (physical addresses)
 - Control the use and allocation of physical pages
- Naming (virtual addresses)
 - Allocate capabilities for virtual addresses
- Protection (translation service)
 - Construct a mapping between physical and virtual addresses

15

SPIN Processor Services

- **No**: not defining interfaces and implementation for scheduling and thread management
- **Yes**: defining set of events coordinating processor allocation between schedulers and thread packages
- **Yes**: Application extensions can handle these events in the kernel
- **No**: Applications can't control how the code will be run

16

Building Services with SPIN

- Kernel and core services can be used to implement more conventional operating system abstractions
 - System calls
 - Address space
 - Networking

17

Performance

Microbenchmarks
Networked video application

Platform

- Alpha 133MHz DEC AXP 3000/400 workstations
 - 64 MB memory, HP C2247-300 1 GB disk drive
- Networking
 - 10Mb/s Ethernet
 - ATM (FORE TCA-100 155Mb/s)
- DEC SRC Modula-3 compiler v3.3
- **Comparison systems**
 - SPIN
 - DEC OSF/1 v2.1
 - Mach 3.0+DEC OSF/1

19

Microbenchmarks

- Page faults
- Thread management (*)
- System call overhead
- Cross-address space procedure call
- Address space management
- Networking (*)

20

Microbenchmarks Thread management

Kernel thread management overhead

Kernel Thread Operation	Mach 3.0 kernel	DEC OSF/1 kernel	SPIN extension
Create	41	332	5
Ping-Pong	71	21	29
Terminate	18	260	7

SPIN's extensible threads does not incur a performance penalty

Overhead to use an implementation of the C-Threads interface for Mach 3.0, DEC OSF/1 2.1 and SPIN from a user-level application

User Thread Operation	Mach 3.0	DEC OSF/1	SPIN layered	SPIN native
Fork	90	1131	103	20
Fork,Run	233	1164	157	64
Ping-Pong	115	233	85	85
Fork,Join	338	1026	223	110

SPIN's thread management is much faster than the others

21

Microbenchmarks Networking

Round trip network RPC time

	DEC OSF/1 kernel	SPIN extension	SPIN bounded
Ethernet	840	579	510
ATM	631	332 (raw 162)	241 (raw 100)

Substantially lower latency when sending directly from the kernel

The user-to-user networking bandwidth and sender CPU utilization for Ethernet and ATM

	DEC OSF/1 kernel	SPIN extension
	Bwidth CPU %	Bwidth CPU %
Ethernet	8.9 Mb/s 35	8.9 Mb/s 20
ATM	25 Mb/s 82	41 Mb/s 55

Less CPU time using SPIN

22

Networked Video Application

- Server (3 extensions)
 - Read video frames from disk
 - Send the video out over the network
 - Transform a single send into multicast to a list of clients
- Client (1 extension)
 - Decompose the image and display it directly to the screen buffer

Server utilization as a function of the number of client video streams

# streams	DEC OSF/1 kernel	SPIN
	CPU %	CPU %
1	28	5
5	64	19
10	75	37
15	78	55
20	NA	72

23

Conclusions

- Possible to achieve *good performance* in an *extensible* operating system without compromising *safety* ☺
- Efficient mechanisms for extending services + core extensible services
- Rely on the language, compiler and runtime mechanisms

24

Critique

- Weaknesses
 - Modula-3
 - The problem of garbage collection
- More about SPIN
 - <http://www-spin.cs.washington.edu/>
 - Developed at UW for approximately two years
 - SPIN Web server
 - SPIN → SPINE

25

Related Works

- Other Extensible Operating Systems
 - Exokernels (MIT) Mach (CMU)
 - NOW (Berkeley) Spring (Sun)
 - Scout (Arizona) VINO (Harvard)
 - Synthetix (OGI)
- **The Singularity Project – Microsoft**

26

Discussion

(Topics from the submitted questions)

1. Language
 - Modula-3 or other type safe languages, C,...?
2. SPIN <> L4, Sandboxing
3. SPIN <> Open-source OS (Linux)
4. Safety (language, core services)
5. A large number of extensions installed into the OS?
6. Security

27