Extensibility, Safety and Performance in the SPIN Operating System

Bershad, Savage, Pardyak, Sirer, Fiuczynski, Becker, Chambers, Eggers.

Department of Computer Science and Engineering
University of Washington
http://www.spin.cs.washington.edu

Presented by Anatoly Akkerman.

SPIN Motivation

✦ Some applications demands are poorly matched by an operatin system’s implementation or interface.
  - Disk buffering and page replacement (database systems)
  - Network stack (high performance distributed applications)

SPIN Solution

✦ Allow application extensions into the OS

Monolithic OS

User application

Kernel

SPIN OS

User application

Kernel

SPIN Goals

✦ Extensibility
  - Export enough interfaces to services and resources (make it FG - fine grain)

✦ Safety
  - Protect the system and other applications from misbehavior
  - Granularity must be the same at which extensions are defined

✦ Performance
  - Need to support low overhead IPC between extension and the system
**SPIN Approach**

- Use language and run-time services to provide low-cost, fine-grained, protected access to OS resources
- Techniques used:
  - Co-location (extensions are in kernel space)
  - Enforced modularity (use of MODULA-3)
  - Logical protection domains (namespaces)
  - Dynamic call binding (extension-to-event binding)
- OS resources and operations are exported through Core System Services Interfaces

**SPIN Approach (cont.)**

Kernel space

- RT
- LIB
- Extensibility Machinery, Domains, Naming, Linking and Dispatching
- SAL

C application
Module-3 application
Fortran application
C application

**SPIN Extensions**

- Granularity of extensions – a procedure call
  - Extension mechanism places no special HW demands.
  - Instead use language-level services such as static type-checking, dynamic linking, etc.
- Extensions are written in MODULA-3
  - Compilation with trusted compiler, execution in the RT+LIB – trusted MODULA-3 run-time an library.

**MODULA-3 Features**

- Support for Interfaces
  - Declares a visible part of an implementation module. Compile-time check for violations of encapsulation.
- Type safety
  - Controlled memory access
- Automatic storage management
  - Ensures that an object reference can serve as a capability, i.e. unforgeable.
- Additional features
  - Generic interfaces, threads, exceptions...
SPIN Protection Model

- **Capabilities**
  - Unforgeable pointer references to system objects and interfaces.
  - Applications written not in MODULA-3 use externalized references to in-kernel structures through a per-application index table.

- **Advantages**
  - No runtime overhead.
  - Very fine-grained protection mechanism.

SPIN Protection Model (cont.)

- **Protection Domains**
  - Defines a set of accessible names available to an execution context.
  - Replaces expensive virtual address protection model (now extensions can be put in kernel-space).
  - SPIN supports explicit creation/coordination/linking of namespaces into protection domains.
  - Kernel nameserver advertises/authorizes protection domains

SPIN Protection Model (cont.)

**Domain Operations**

- **Create**
  - Object File
  - Domain
  - exported
  - imported
  
  Loads a safe object file.
  Symbols exported by interfaces are exported from the domain. Imported symbols are unresolved.

- **Resolve**
  - Target Domain
  - Source Domain
  - exported
  - imported
  
  Resolves any unresolved symbols in the Target Domain against symbols in the Source Domain.
  Resources then can be shared at memory speed

SPIN Protection Model (cont.)

**Domain Operations**

- **Combine**
  - Domain
  
  Creates a union of existing domains.
  Used to bind together collections of related interfaces.
SPIN Extension Model

- Extensions can
  - Passively monitor system activity
  - Offer hints to the system (e.g., for page replacement)
  - Entirely replace system services
- Extension invocation
  - Through events and handlers
  - An extension registers as a handler or as an event
  - Kernel event dispatcher redirects handling of the event to the handler

SPIN Extension Model (cont.)

- Events are procedures exported from an interface
- Handlers are procedures of the same type as events they handle
  - Any number of handlers can be associated with an event
  - Kernel event dispatcher specifies arguments for the handler at run-time
    - Kernel can preempt handler execution
    - When only one handler, event dispatch is a function call, otherwise, a dynamically compiled routine

SPIN Extension Model (cont.)

- Handler restrictions can be enforced by the module that exports the event (primary module)
  - Deny/accept other module handlers
  - Associate guards (predicates) for executing a handler
  - Restrict handler execution to
    - Synchronous/asynchronous
    - Bounded time
    - Order of execution
**SPIN Handler Scalability**

Cost of dispatch as a function of number of handlers when only one handler executes

Optimized – unrolled dispatch processing loop
Inlined – guard and handler inlining in the processing loop
Guard Optimizations – guard evaluation optimization

---

**SPIN Core Services**

- Basic set of primitives for further extensibility
  - Memory management
    - Supports physical storage, naming and translation
  - Thread management
    - Strand interface that provides Blocking/Unblocking/Checkpointing/Resumption

---

**SPIN Extensible Memory Management**

- **Physical Address Service**
  - Controls use and allocation of physical pages
    - PhysAddr.Reclaim event handling allows voluntary of pages
    - Allocate event from clients returns a capability for physical storage
  - **Virtual Address Service**
    - Allocates capabilities for virtual addresses
  - **Translation Service**
    - Used to express relationship between the physical and virtual addresses
    - Raises a set of events that correspond to exceptional MMU conditions
      - Translation.BadAddress – unallocated virtual address access
      - Translation.PageNotPresent – unmapped virtual address access

---

**SPIN Extensible Thread Management**

- Global scheduler implements primary processor allocation between strands
  - Allows thread packages with own schedulers
- **Strand Interface** Events
  - **Block** – hardware signals that a strand is not runnable
  - **Unblock** – interrupt handler signals completion of IO
  - **Checkpoint** – strand is being descheduled, save state
  - **Resume** – strand is being scheduled, restore state
- Global Scheduler sends **Checkpoint/Resume** to the thread package upon receiving **Block/Unblock**
References

- Extensibility, Safety and Performance in the SPIN Operating System
  Benshad, Savage, Radyal, Siler, Becker, Floczinski, Chambers, Beggis
  "Proceedings of the 19th ACM Symposium on Operating System Principles
  (SOSP-15)"
  Paper and SOSP-15 Presentation Slides
- Dynamic Binding for an Extensible System
  Padyal, Benshad
  "Proceedings of the Second USENIX Symposium on Operating System Design
  and Implementation (OSDI-96)"
  Paper and OSDI-96 Presentation Slides
- Honors Operating Systems G22.3250, Lecture 26 Presentation Slides
  Palem, Harrison, Karamcheti
- SPIN web-site
  http://www-spin.cs.washington.edu