Squawk in Solaris™

Hiroshi Yamauchi
Intern, Mayhem, Sun Labs

Summer 2004

What?

Run Java™ inside the Solaris kernel

Why?

Take advantage of Java for kernel development

- Memory safety
- Type safety
- Automatic memory management
- Extensibility via object- orientation
- Productivity
- Portability
- Rich libraries, etc.

Related Work

Pilot (1980),
SPIN (1995),
Oberon (1992)

- Safe languages in OS

MVM (2000-),
Jkernel (1998),
KaffeOS (2000)

- OS-like isolation in a JVM

JavaOS™ (-1999):

- OS built from scratch in Java
- Commercially failed

JX (1999-):

- OS built from scratch in Java
- Open source

Java in Solaris (2003):

- Attempted to port Hotspot™ into Solaris kernel
- Goal: to run Java apps in the kernel efficiently
    - Better resource control (CPU,etc)
    - Less system call overhead
- Did not complete due to high complexity

This work, Squawk in Solaris

Goal: to use Java as part of Solaris kernel

e.g., device drivers

Challenges: *harsh* kernel environment

No rich libraries

No process abstraction

Little debugging support

Approach: Port Squawk into Solaris 10 kernel

The Squawk Virtual Machine

- Small J2ME JVM
- Developed at Sun Labs
- Mostly written in Java
- Interpreted, compiler in development
- Cheney GC
- Green threaded
- Bootstraps from boot image
- Few external dependencies
- Simple

Porting Work

- Packaged as a kernel module
- Ported to 64 bit
- Added kernel file I/O module (`kfileio`)
- Made re-entrant (all global vars in a struct)

Additional Work

- Added kernel native interface

  Kernel function calls via Squawk's native interface

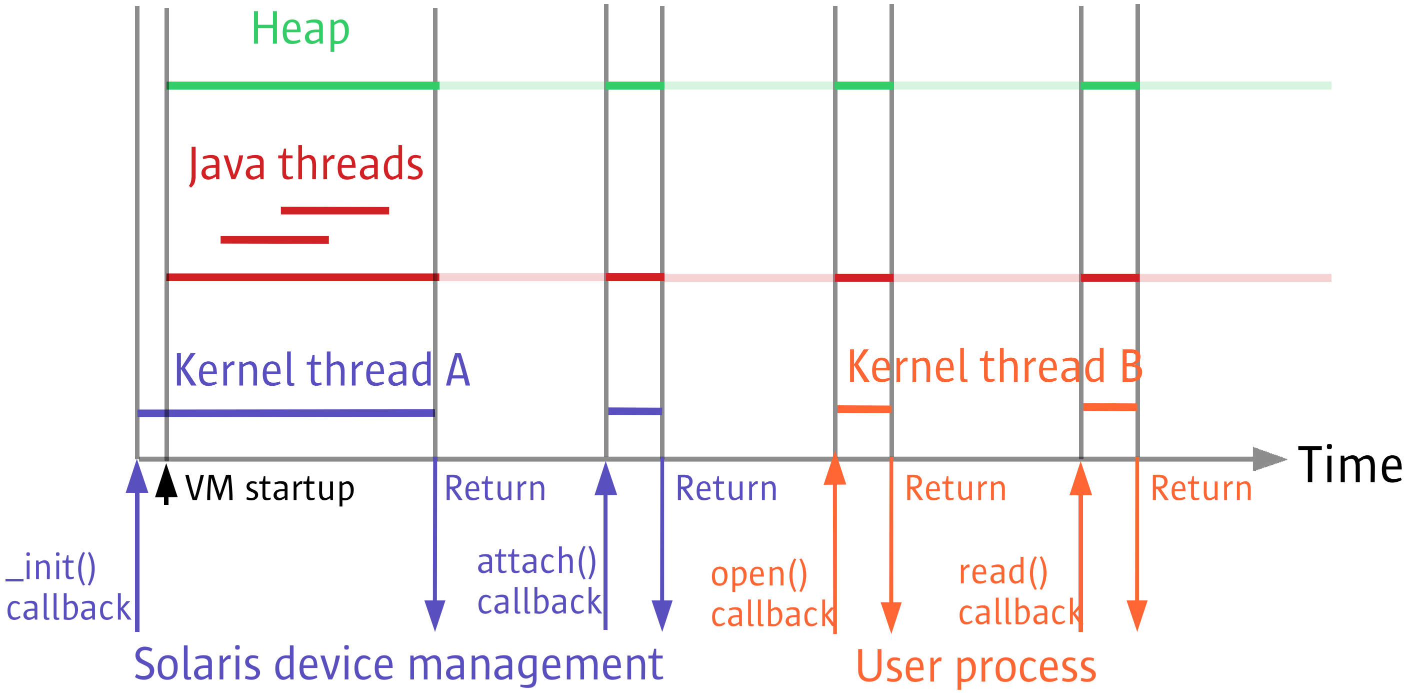
  Java classes wrapping kernel structs

  (Class `Address` to represent native pointers)

Additional Work

- Added device driver execution mode

  VM execution driven by kernel threads calling driver



Application: Device Drivers in Java

- Java device driver interface (DDI) built on top of the Solaris DDI
- Driver code structure

C:

```
struct dev_ops ops = {
    ...
    my_attach,
    my_detach,
    ...
};
int my_attach() { ... }
int my_detach() { ... }
...
```

Java:

```
abstract class DeviceDriver {
    abstract int attach();
    abstract int detach();
    ...
}
class MyDeviceDriver
    extends DeviceDriver {
    int attach() { ... }
    int detach() { ... }
    ...
}
```

A RAM disk driver in Java

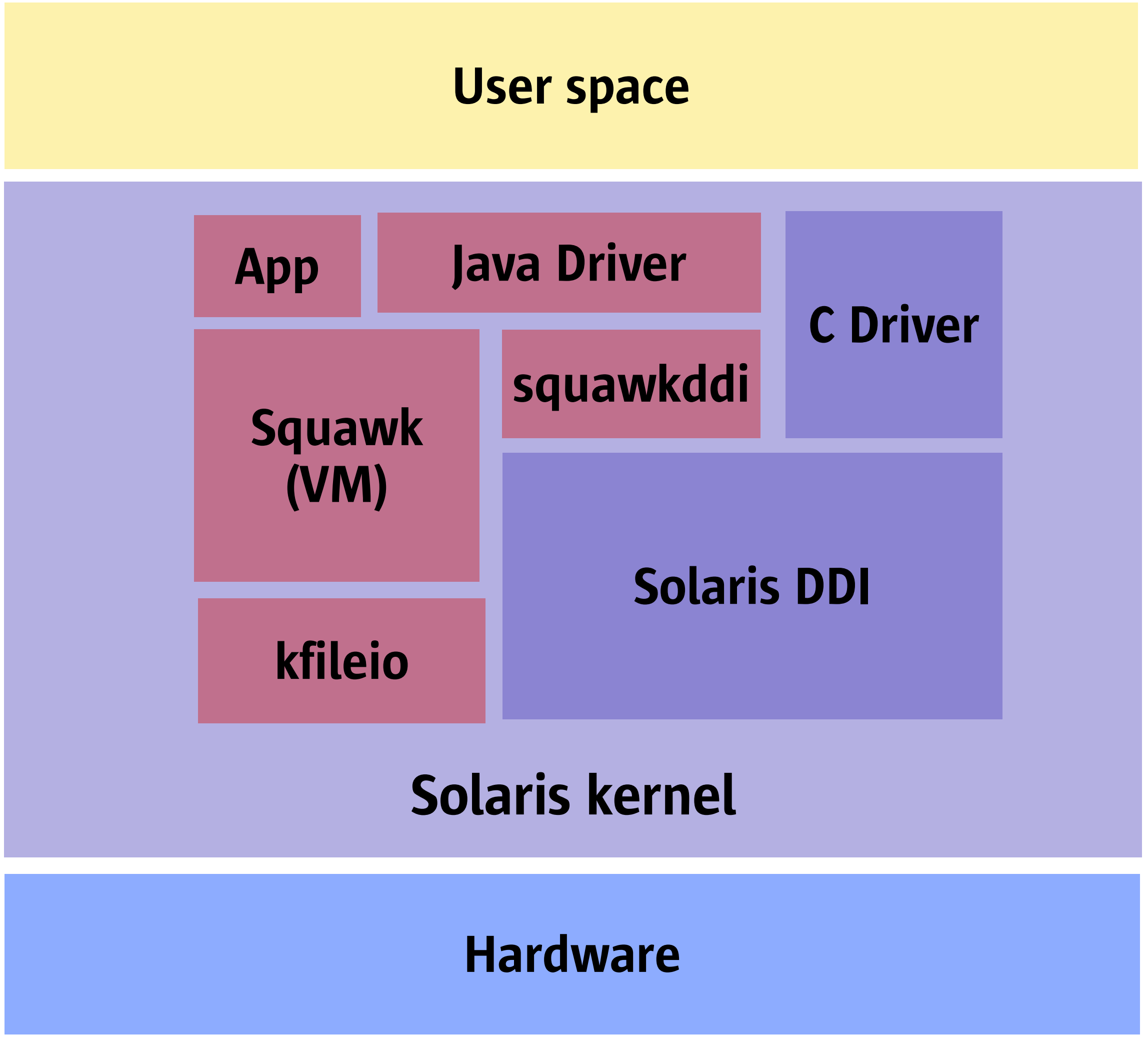
- Translated `ramdisk.c` to `RamDiskDriver.java`
  - Took one intern-week
  - LOC, C: 578 vs Java: 422
- Measurements
  - Raw system call overhead

    C: 3.84 vs Java: 3.85 µsecs
  - Block IO performance : copy a 1MB file

    C: 63 vs Java: 178 (w/o GC) µsecs 2.8x slower

    230 (w/ GC) 3.8x slower

Software Stack



User space

App

Java Driver

C Driver

Squawk (VM)

squawkddi

Solaris DDI

kfileio

Solaris kernel

Hardware

Future Work

- Multiple kernel threads support

  Necessary for simultaneous callbacks

- Complete Java DDI

  Writing more working drivers

- Advanced VM implementation technologies

  Compiler and garbage collector

Summary

- Ran a JVM inside Solaris kernel and applied it to device drivers
- Established a basis for exploring the benefits of Java inside the Solaris kernel

Demo: RAM disk driver in action

1. It's working
2. An intentional bug inserted
   - A null pointer exception will happen if the 14496th disk block is accessed
   - Java vs C

Acknowledgement

Mario Wolczko (my mgr)
Cristina Cifuentes
Grzegorz Czajkowski
Erika Gunadi
Matt Seidl
Doug Simon
Nik Shaylor
Glenn Skinner
Greg Wright