L4/Darwin: Evolving UNIX

Charles Gray
Research Engineer, National ICT Australia
charles.gray@nicta.com.au
Outline

1. Project Overview
2. BSD on the Mach microkernel
3. Porting Darwin to the L4 microkernel
4. Project Status
Darbat Overview

- Para-virtualised Darwin kernel
- L4 μ-kernel hypervisor
- Isolated kernel and device drivers
- Standard virtualisation benefits
Why?

- Virtualisation
- Flexibility
- Robustness
- Code size
Linux Kernel Size (LoC)
Mach

- Once a hot research topic
- Many attempts at BSD on Mach
- Darwin is a monolithic Mach kernel
- Gave microkernels a bad name
What’s in a kernel?

- XML parser
- Decompression algorithms
- Linker
- Network routing
- Device drivers
- Strings
Mac OS X Architecture

Darwin

- launchd
- UNIX App

xnu

Hardware

- OS X App
- Aqua GUI

User

Kernel
Darwin

- Unique kernel design
  - Mach 3.0 μ-kernel
  - 4.4 BSD derivative
  - I/O Kit device drivers
- Mac OS X applications
- Full FreeBSD 5 UNIX environment
BSD Layer

fork() -> Kernel
mmap() -> Kernel
listen() -> User

BSD

Threads

Process

Scheduler

Mach

UBC

VM

HFS+

UFS

TCP

Disk

Network

PCI Bus

I/O Kit

interrupt
The I/O Kit

- BSD
- HFS+
- Partition0
- Partition1
- Disk0
- Disk1
- ATA
- NIC
- PCI Bus
- User
- Client
- Keyboard
- UHCI

mach_msg()

User
Kernel

Interrupt
Paging
BSD and I/O Kit

read() → ioctl() → read()
• 2nd generation u-kernel
• Learn from mistakes of Mach
• Pay attention to cache, TLB, complexity
• Make microkernels work
• 10-20x performance improvement over Mach
L4 μ-kernel

OS Kernel

Application

IPC

Threads

Address Spaces

Scheduler

interrupt

page-fault / exception

Hardware

not to scale
Darbat

- Decompose Darwin kernel
- Tackle complexity problem
- Add flexibility
- Maintain (or improve) performance
- All with binary compatibility
XNU on L4

ISR Thread

Work-loop Thread

Mach

timer

int. mask

exception

page faults

syscalls

I/O Kit
Server Consolidation

- Multiple OS instances
- Kernel support
- Simplified driver model
Crashable Drivers

- Drivers crash
  - Holding locks
  - Hogging resources
- Can isolate drivers
  - and keep performance
- Some devices still critical
VM Aware Scheduling

- Application
- Application
- Application
- Application

- xnu
- xnu
- xnu

- L4
- Hardware

- User
- Kernel
Reliable Servers

- Microkernel is very resilient
- Protect data from 3rd party modules
- Ensure reliable execution of code
- Cheaper software solutions
Heterogeneous Systems

- L4 is OS (everything?) agnostic
  - Linux
  - *BSD
- Share devices
- Share file systems
System Call Performance

Mac OS X 10.4.7

UNIX App

~2500 cycles

xnu

Hardware

Darbat 0.2

UNIX App

~2400 cycles

xnu

L4

Hardware

mach_msg null operation
In-task Synchronisation

Mac OS X 10.4.7

~17,000 cycles

Darbat

~6,500 cycles

Pthreads synchronisation
IPC Optimisation

- L4 IPC is a subset of Mach IPC
- Applications rely on Mach semantics
- Optimisation for some messages
• Sometimes you just need a hack
• In-kernel scripts get messy
• Scheduling and locks are a pain
• Don’t compromise the whole system
Lessons So Far

- Darwin modularity
- Binary compatibility
- Mac OS X has bugs
- Performance
- Debugging
Future Work?

- Further decomposition
- Improved xnu and I/O Kit bindings
- Usable system!
- Experiment with features
- System benchmarks
Summary

- UNIX kernels are feature-packed
- L4 is one option to address this problem
- Extend UNIX kernel to meet modern usage

Questions?