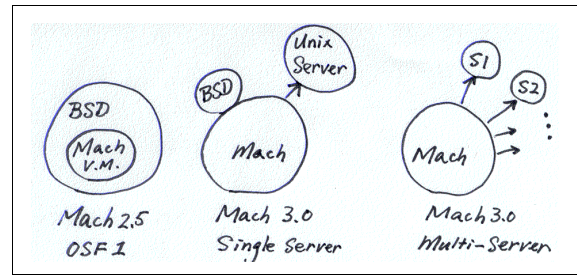


# CS 518

## Mach: External Pagers and User-level OS Servers

Randy Wang  
Fall 2002  
Princeton University

## History of Mach



CS518

1

Randy Wang

## Outline

- Background
- Copy-on-write and external paging
- User-level Unix server
- Micro-kernels: true value or fad?

CS518

2

Randy Wang

## Outline

- ~~Background~~
- **Copy-on-write and external paging**
- User-level Unix server
- Micro-kernels: true value or fad?

CS518

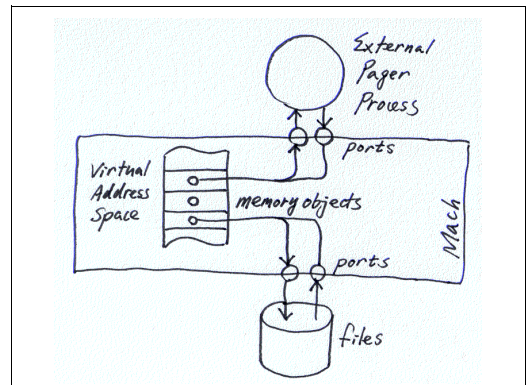
3

Randy Wang

## Copy-on-write (Messages Become Memory Ops)

- Problem: expensive to send big messages
- Solution
  - “Virtual copy”
  - Read-share same data
  - Make copy when written
- More essential for user-level servers

## External Pagers (Memory Become Messages)



- Memory objects backed by various types of files
- More generally, they are backed by processes
- Send/receive messages to perform paging operations

## Uses of External Pagers

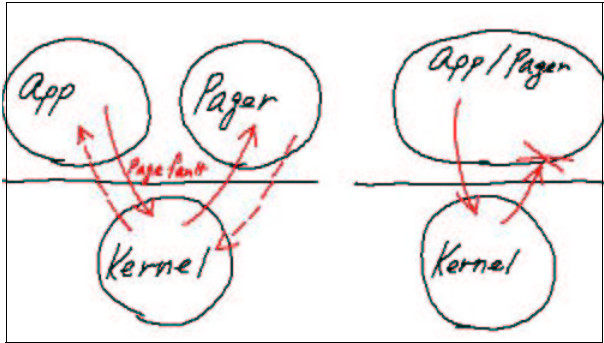
- Distributed shared memory
- Process migration
- Memory-resident object transactions

## Limitations of the Mach External Pagers

Implementation deficiencies, not necessarily fundamental limitations

- External pagers can't control replacement policies
- Shadow objects can't be backed by external pagers

## Interactions



good or bad

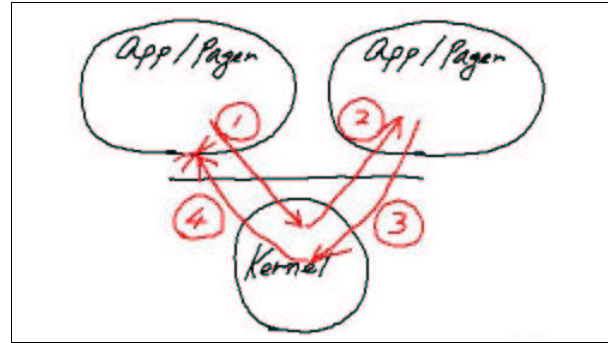
- Copy-on-write needed by external pagers
- Pagers may deadlock themselves

CS518

8

Randy Wang

## Interactions



good or bad

- Copy-on-write needed by external pagers
- Pagers may deadlock themselves

CS518

9

Randy Wang

## Outline

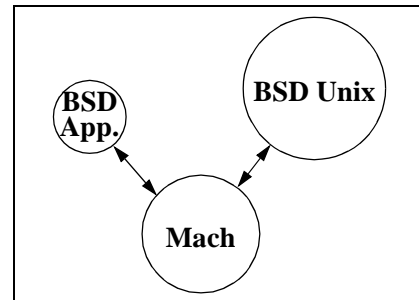
- ~~Background~~
- ~~Copy on write and external paging~~
- **User-level Unix server**
- Micro-kernels: true value or fad?

CS518

10

Randy Wang

## User-Level Unix Server



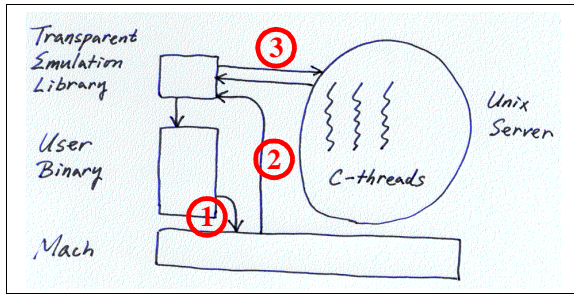
- Goal is binary compatibility between BSD and Mach

CS518

11

Randy Wang

## User-level Unix Server



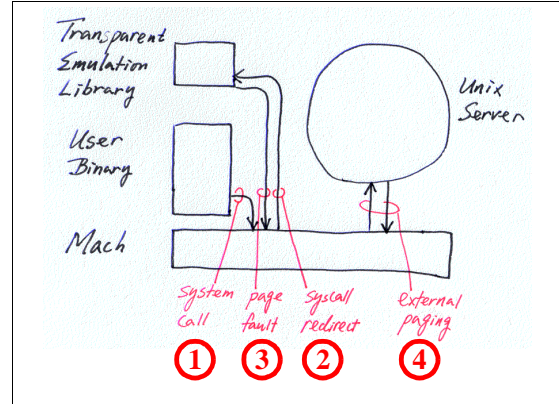
- System call redirect into user space
- Transparent Emulation Library (TEL) part of user address space
  - Handles system calls
  - Translate system calls to RPC to Unix server
  - Vulnerable to tampering

CS518

12

Randy Wang

## Unix I/O



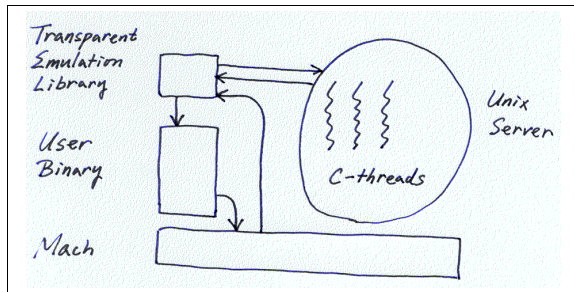
- Choice between message passing and using Mach virtual memory
- Map file data into TEL
- Unix server becomes the external pager

CS518

13

Randy Wang

## Optimizations



- Cache in TEL
- Use shared memory (between TEL and Unix server)
  - tricky for protection and sharing
- Use Mach kernel primitives directly
- Server acts as external pager

CS518

14

Randy Wang

## The Mach Kernel

- Interprocess communication
  - ports, capabilities, and messages
- Memory objects
  - physical memory as a cache of memory objects
- Scheduling
  - Unix processes created and manipulated by the Unix server
- System call redirection
- Device support
  - via ports
- User level multi-threading
  - C-threads

CS518

15

Randy Wang

## Outline

- Background
- Copy-on-write and external paging
- User-level Unix server
- Micro-kernels: true value or fad?

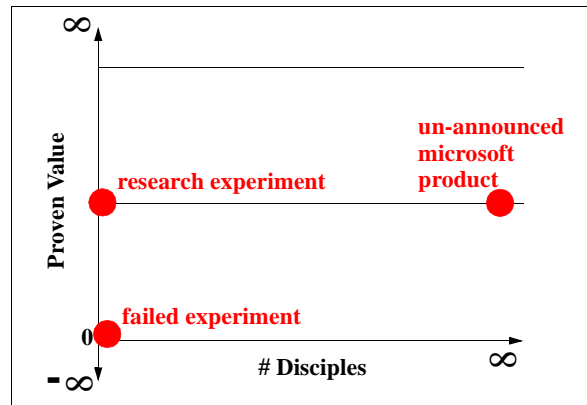
CS518

16

Randy Wang

## Fads or True Value?

(Keynote talk by J. Ousterhout at the 1st Mach Symposium)

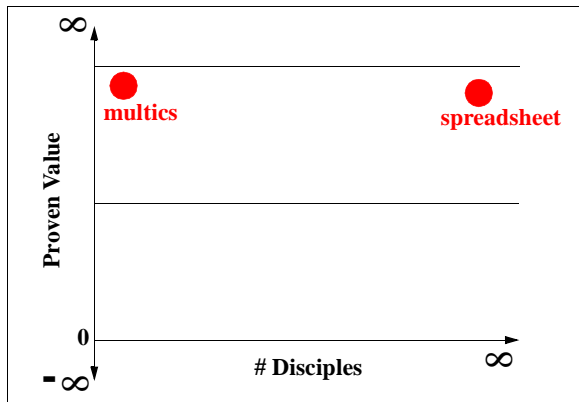


CS518

17

Randy Wang

## Fads or True Value?

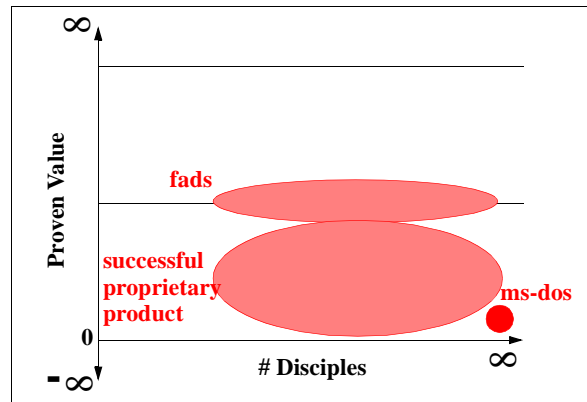


CS518

18

Randy Wang

## Fads or True Value?

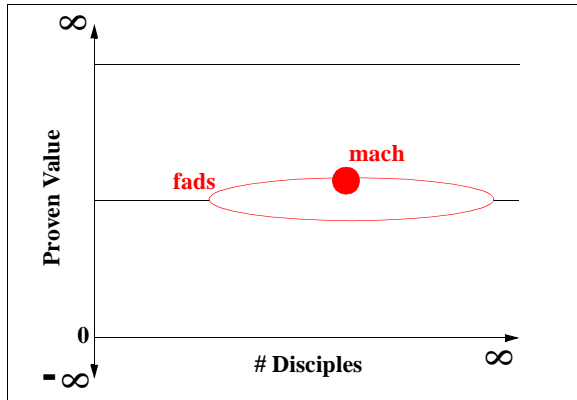


CS518

19

Randy Wang

## Fads or True Value?



- Virtual memory work is good
- But the rest of it: an example of pushing an idea too far

CS518

20

Randy Wang

## Why Micro-kernel and Why Not?

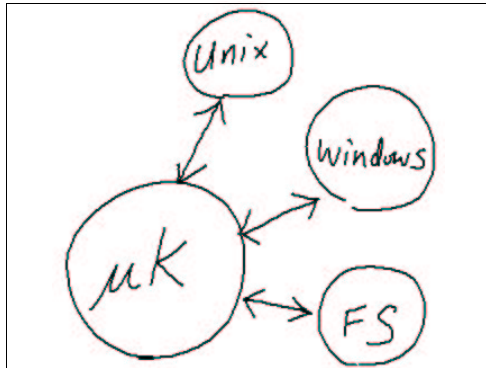
- Extensibility
  - pro: easy to add features
  - pro: easy to debug
  - con: debug OS isn't that hard
  - con: machines are cheap
  - con: not too many subsystems built

CS518

21

Randy Wang

## Why Micro-kernel and Why Not?



- Concurrent personality
  - pro: run multiple OS'es at the same time
  - pro: run apps for different OS at the same time
  - pro: capture legacy system
  - con: interaction among the systems is hard
  - con: no sharing of subsystems

CS518

22

Randy Wang

## Why Micro-kernel and Why Not?

(cont.)

- Modularity
  - pro: easier to engineer and maintain
  - pro: resulting system more reliable
  - con: need to make subsystem crossing cheap without sacrificing modularity
  - con: good engineering style can give modularity anyhow
  - con: major OS components are hard to decompose and become "modularity bottlenecks"
    - 40% file system code?!
  - con: modularity has overhead

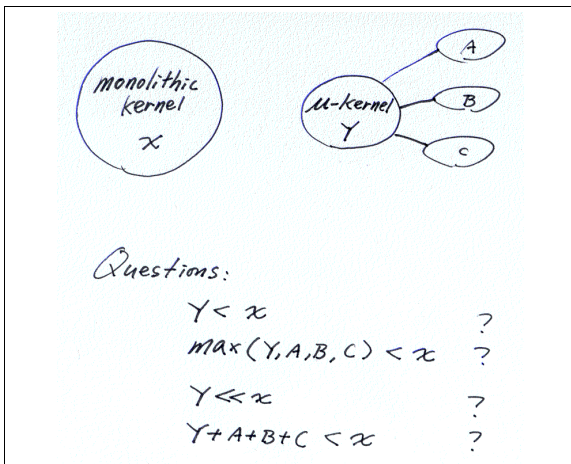
CS518

23

Randy Wang

## Why Micro-kernel and Why Not?

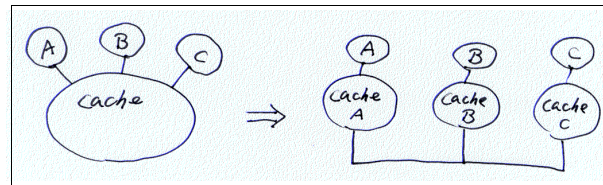
(cont.)



- The complete Plan 9 kernel is smaller than the Mach micro-kernel?

## Why Micro-kernel and Why Not?

(cont.)



- Distributed system support
  - pro: should be easy to distribute the subsystems
  - con: things are never this simple
    - simple example: copy on write
    - more complex example: caching and cache coherence

## Why Micro-kernel and Why Not?

(cont.)

- Security
  - pro: smaller security bottleneck (micro-kernel)
  - pro: firewalls among subsystems
  - con: need to trust the key subsystems anyhow
  - pro: may be easier to prove the correctness of individual systems
- Portability
  - pro: just need to port the micro-kernel
  - con: this is a good argument, although good interface design should be able to achieve this anyhow
    - Consider machine-independent VM