The Duality of Memory and Communication in the Implementation of a Multiprocessor Operating System

Michael Young, Avadis Tevanian, Richard Rashid, David Golub, Jeffrey Eppinger, Jonathan Chew, William Bolosky, David Black, and Robert Baron

ACM Symposium on Operating System Principles, 1987

Presented By Rajesh Sudarsan
October 21, 2005

Agenda

- Introduction
- Key Ideas
- Monolithic vs Microkernel
- Primitive abstractions
- Implementation Details
- Issues with External Memory Management
- Benefits of Duality
- Conclusion

Introduction

- Mach OS project started in 1985. Continued till 1994.
- Successor to Accent OS developed at CMU
- MACH ⟹ NeXTSTEP ⟹ OPENSTEP ⟹ Mac OS X
- Mach OS kernel – mainly designed to support multiprocessors.
- Microkernel - a small, efficient kernel providing basic services such as process control and communication

Design goals

- Object oriented interface with small number of basic system objects
- Support for distributed and multiprocessing
- Portability to different multiprocessor and uniprocessor architectures
- Compatibility with BSD UNIX
- Performance comparable to commercial UNIX distributions

Key ideas

- Communication and virtual memory can play complementary roles in OS kernel
  - Increased flexibility in memory management
  - Support for multiprocessors
  - Improved performance
  - Easier task migration
- Memory represented as abstract objects called memory objects
- Single level store implementation

Key ideas (contd.)

- Virtual memory implementation using memory objects
- External memory management – Structure for secondary storage management

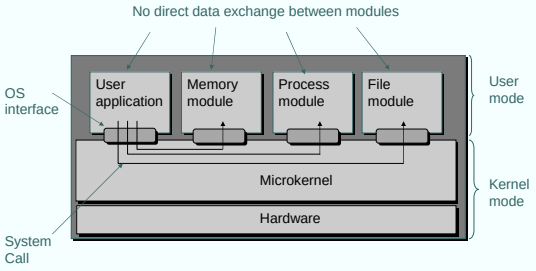
Microkernel vs Monolithic

- Monolithic kernel
  - Kernel interacts directly with the hardware
  - Kernel can be optimized for a particular hardware architecture
  - Kernel is not very portable
- Microkernel
  - Kernel is very small
  - Most OS services are not part of the kernel and run at a layer above it
  - Very easily portable to other systems

Examples

- Microkernel
  - Amoeba, Minix, Chorus, Mach, GNU Hurd, NeXTSTEP, Mac OS X, Windows NT
- Monolithic kernel
  - Traditional UNIX kernels, such as BSD, Linux, Solaris, Agnix

Architecture



*source *Distributed systems – Principles and Paradigms*, Andrew Tannembaum, Maarten van Steen

Primitive abstractions in Mach OS

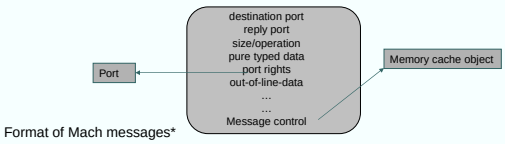
- Four basic abstractions from Accent
  - Task, Threads, Ports, Messages
  - Port set
- Fifth abstraction introduced in Mach
  - Memory Objects
- Tasks and Threads – Execution control primitives
- Ports and Messages - Interprocess communication

Interprocess communication

- Two components of Mach IPC – ports and messages
- Ports
  - Communication channel
  - Provides finite length queue
  - Protected bounded queue within the kernel
- Messages
  - Fixed length header and variable size collection of typed data objects

IPC (contd.)

- One receiver, multiple sender
- Tasks allocate ports to perform communication
- Task can deallocate rights to a port



Format of Mach messages*

*source *Operating System Concepts*, Sixth Edition by Avi Silberschatz, Peter Baer, Galvin Greg Gagne

Memory Management

- Virtual memory – level of abstraction between process memory requests and physical memory
- Continuous address space
- Demand Paging
- Transparent relocation of running programs in memory
- Page and Page frame
- VM -> RAM – page global directory, page table, offset

Virtual Memory Management in microkernel

- Each task has its own virtual address space
- Restriction – virtual address space must be aligned with the system page boundaries
- Supports read/write sharing of memory among tasks of common ancestry through inheritance
- API provided for operation on VM

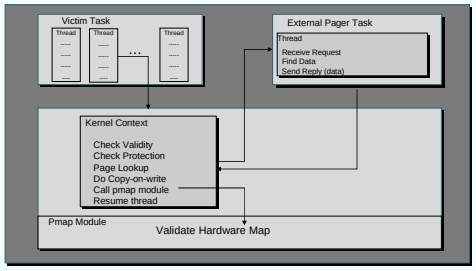
External Memory Management (EMM)

- External memory management interface based on memory object
- Memory object – abstract collection of data bytes with operations defined on them
- Memory object represented by a port
- Secondary storage objects available using message passing (data managers)
- Mach kernel as cache manager for contents of memory object

External Memory Management (contd.)

- Interface between kernel and data manager consists of 3 parts:
  - Calls made by application program to cause object to be mapped into its address space
  - Calls made by the kernel on the data manager
  - Calls made by the data manager on the Mach kernel to control use of its memory object

Fault Handling



Minimal Filesystem

```
Char * file_data;
int i, file_size;
extern float rand();

fs_read_file("filename", &file_data,
file_size);
.....
.....
fs_write_file("filename", file_data,
file_size/2);
vm_deallocate(task_self(), file data,
file_size);
```

Call from the application

```
return_t fs_read_file( name, data, size)
 {

//Allocate memory object (a port) and
accept request
port_allocate(t...);
port_enable(...);
.....
//perform file lookup. Find current file
size
.....
//Map the memory object into client
address space
vm_allocate_with_pager(...);
return (success);
}
```

Minimal Filesystem (contd.)

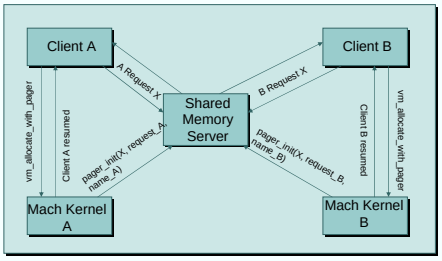
```
void pager_data_request(
memory_object, pager_request, offset,
size, access)
{
//Allocate disk buffer
vm_allocate (…);
//Lookup memory object and read disk
data
disk_read(…);
//Return the data without any lock
pager_data_povided(…);
//Deallocate disk buffer
vm_deallocate(…);
}
```

File retrieval for the application

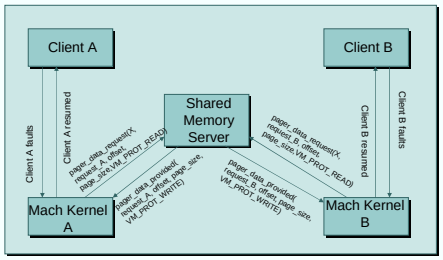
```
void port_death (request_port)
{
//find associated memory object with the
port
lookup_by_request_port(…);
//Release resources
port_deallocate(…);
vm_deallocate(…);
}
```

Release filesystem resources after application deallocates its resources

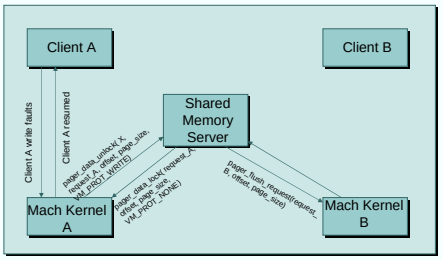
Consistent Network Shared Memory (Initialization)



Consistent Network Shared Memory (Read)



Consistent Network Shared Memory (Write)



Implementation Details

- Four basic data structures used to implement EMM
  - Address Map -Two level map
    - Top level – protection and inheritance information, link to second level
    - Second level – Map to memory object structures
  - Virtual Memory Object structures
  - Resident Memory Structures
  - Page replacement queues

External Memory Management Issues

- Types of Memory failure – Data manager
  - doesn't return data
  - fails to free flushed data
  - floods the cache
  - changes its own data
  - backs up its own data
- Handling Memory failure
  - Timeout, notification, wait, abort
  - Default pager
  - Reserved memory pool

Benefits of duality

- Multiprocessor support for UMA, NUMA, and NORMA architectures
- The programmer has the option to choose between shared memory and message-based communication
- Emulation of operating system environment such as UNIX achieved on Mach
- Generic UNIX system calls can be implemented outside Mach kernel
- Other features supported are transaction and database facilities, task migration, and AI knowledge bases

Conclusion

- Significant contribution to the operating system research
- No experimental to support performance claim
- More information could be presented in the implementation
- Drawbacks
  - Frequent message passing may cause degradation in performance
  - Continuous monitoring of external services

Current Trend

- Pure microkernel architecture not common
- Most OS kernels are hybrid models of monolithic and microkernel, e.g., Windows XP, Windows 2000

Questions?