Lecture Notes, 9/16/98

Machine-Independent Virtual Memory in Mach

I. Machine-Independent VM in Mach

Goals:

- Machine-independent VM management
- Flexible address space manipulation
- Multiprocessor support

Functionality requirements:

- Large, sparse address spaces.
- Copy-on-write and read/write memory sharing between tasks.
- Memory-mapped files.
- User-provided backing store objects and pagers.

Details of Mach abstractions are mostly irrelevant for this paper: need a way for kernel and pagers to communicate.

VM programming interface: (see paper for details)

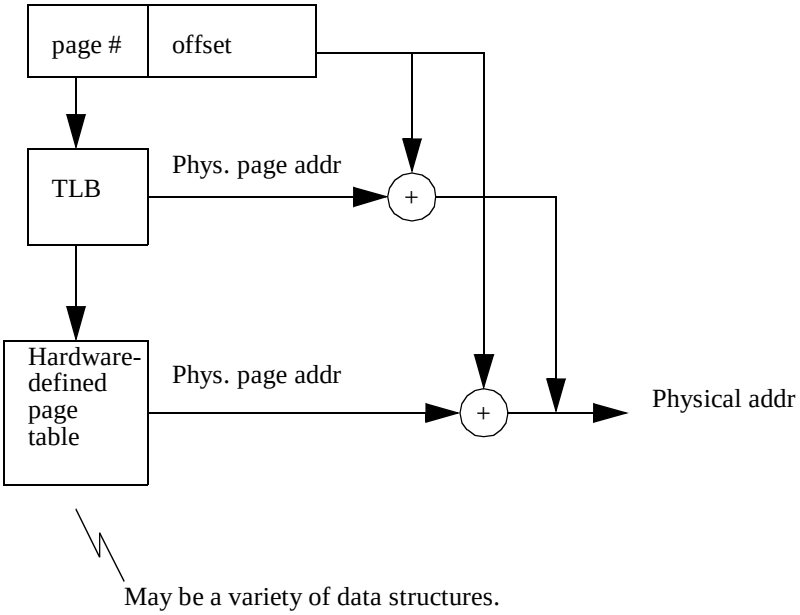
- address space manipulation
- memory protection
- memory inheritance
- misc.

Types of VM data structures:

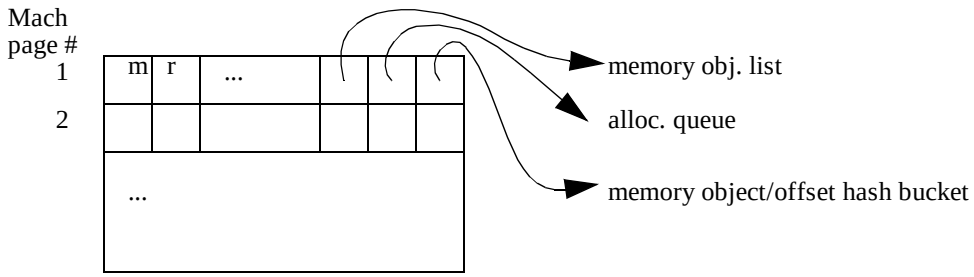
- resident page table: machine-independent info. about physical memory pages.
- memory object: unit of backing storage managed by kernel or user-level pager task.
- address map: describes mapping from a virtual address space to regions of memory objects.
- pmap: machine-dependent memory mapping data structure (i.e. hardware-defined physical addr. map).

Lecture 6

Review of VM paging:

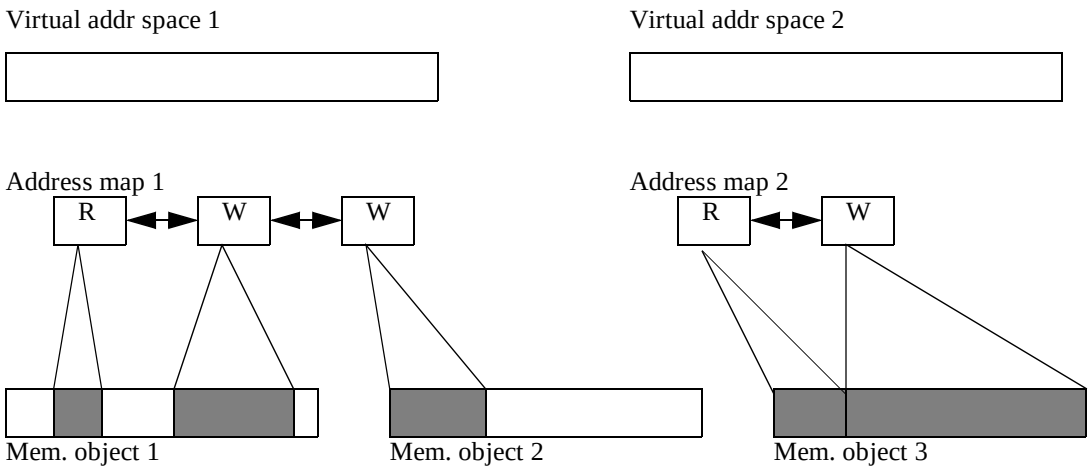


Resident page table: describes state of machine's physical memory

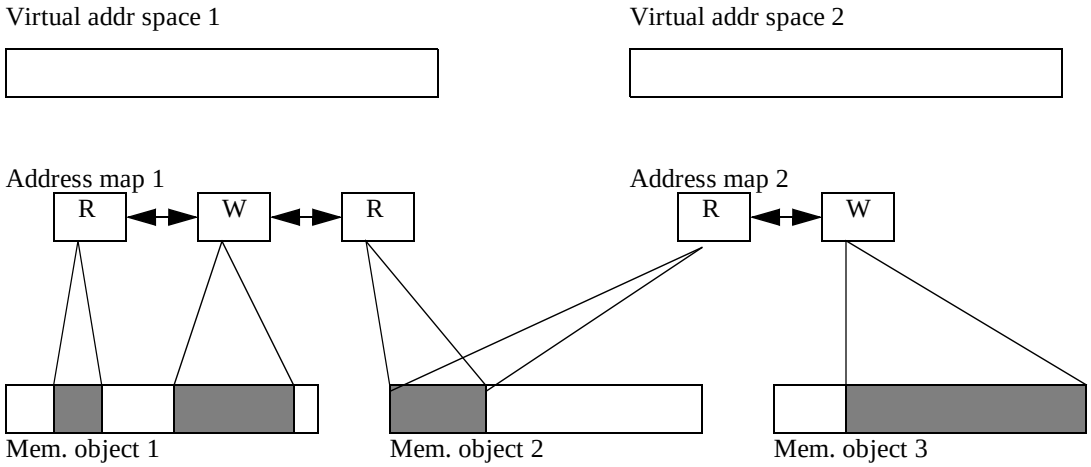


Lecture 6

Address maps: inheritance and protection of virtual memory regions

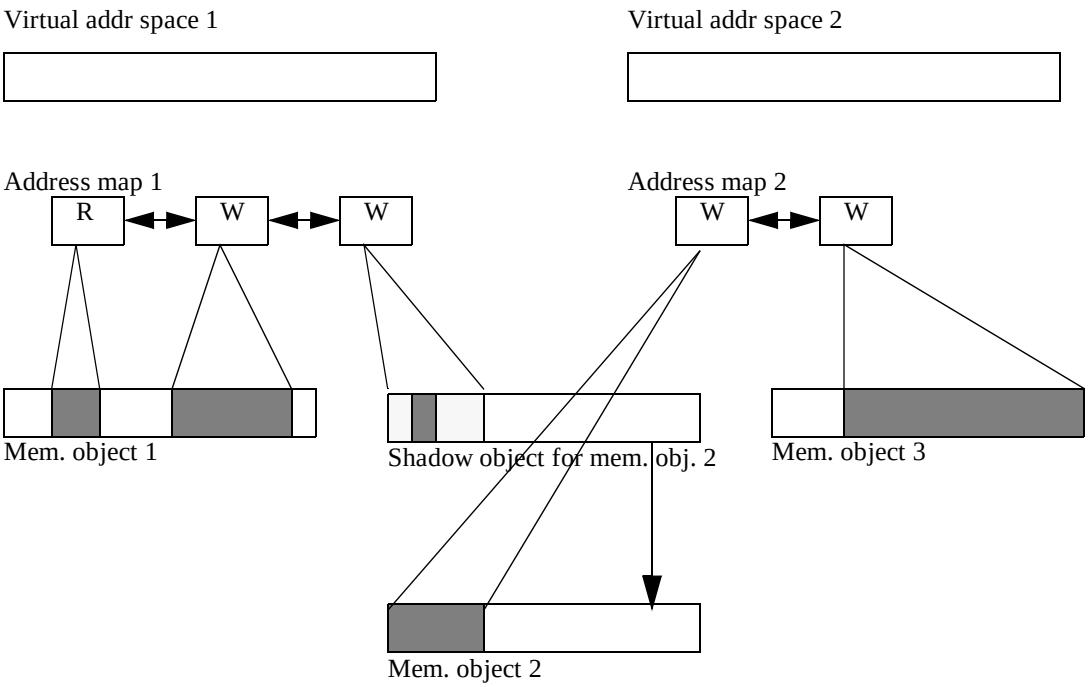


Copy-on-write: read-only case

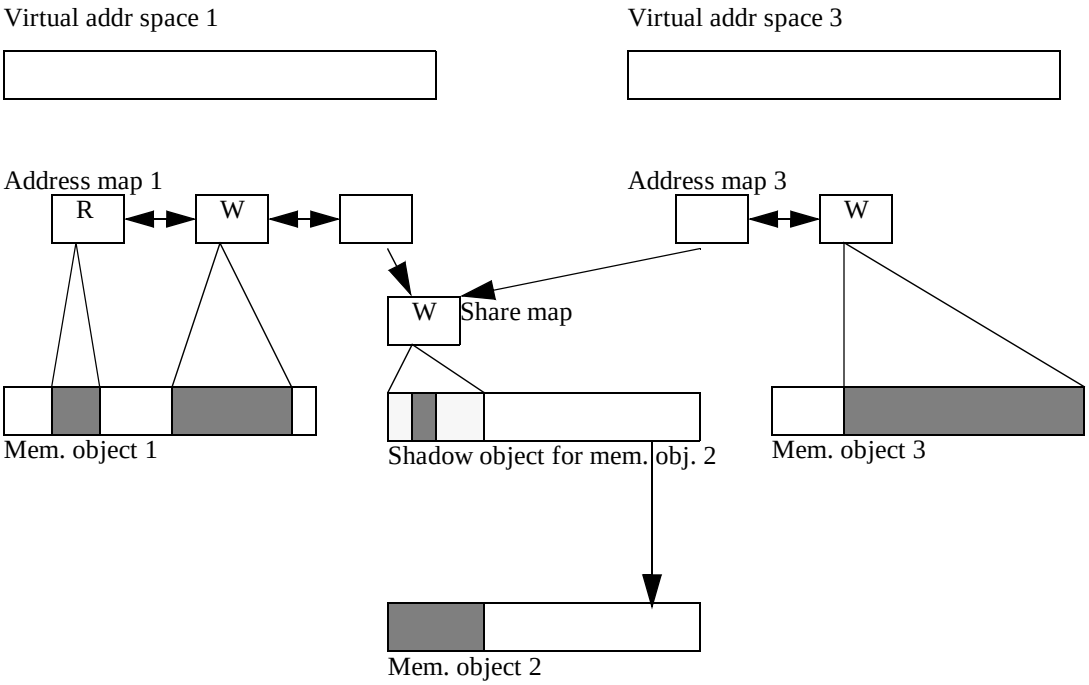


Lecture 6

Shadow objects: supporting copy-on-write with modifications



Share maps: supporting read/write sharing



Lecture 6

An example of remapping an address map entry: sending data copy-on-write to a shared VM region.

Pmap interface hides all machine-dependent aspects of VM (including hardware-defined page table facilities).

Performance:

- Zero fill:
  - Win against uVax because of 4x page size.
  - Win against Sun & IBM because they simulate VAX VM.
- Fork: Mach should compare itself to vfork, not fork! Worse than Unix vfork on uVax.
- Parallel performance:
  - Demonstrates that Mach VM works in an MP environment.
  - Fault-copy case is faster because there is less cache thrashing during a fault.
- Overall performance: paper doesn't say how much of the performance gain in Mach is due to the object cache.

MP issues:

- TLB consistency and "shoot-down".
- No mention of how much trouble they had with making the kernel concurrent.

3 key features of paper:

- First machine-independent, easily portable VM design.
- Supports multi-address space sharing and user-definable pagers through address maps, share maps, memory objects, and shadow objects.
- Machine-dependent part of VM described by a single abstraction: the pmap.

Some flaws:

- Not much discussion of MP issues (none about restructuring Unix kernel to be concurrent).
- In general, lots of things left unsaid or only alluded to.
- Weak performance evaluation section.

A lesson: Easily portable, machine-independent designs are possible for low-level OS components.