The Process Model (2)

Lecture 4, Part 3: More on VM
Prof. Robert N. M. Watson
2022-2023
Virtual memory (quick, painful)

Lecture 4 - The Process Model (2)
So: back to Virtual Memory (VM)

• The process model’s isolation guarantees incur real expense
• The VM subsystem works quite hard to avoid expense
  • Shared memory, copy-on-write, page flipping
  • Background page zeroing
  • Superpages to improve TLB efficiency
• VM avoids work, but also manages memory footprint
  • Memory as a cache of secondary storage (files, swap)
  • Demand paging vs. I/O clustering
  • LRU / preemptive swapping to maintain free-page pool
  • Recently: memory compression and deduplication
• These ideas were known before Mach, but...
  • Acetta, et al. impose principled design, turn them into an art form
  • Provide a model beyond V→P mappings in page tables
  • And ideas such as the message-passing—shared-memory duality
Kernel programmer view of VM

Machine-independent virtual memory (VM)

Machine-dependant physical map (PMAP)

Stack
- "vm_map_entry"
- "vm_object"
- anonymous swap-backed VM object
- swap pager
- page
- page
- "vm_page"
- Read/write, grows down, anonymous object

Heap
- "vm_object"
- anonymous swap-backed VM object
- swap pager
- page
- page
- "vm_page"
- Read/write, anonymous object

Library
- "vm_object"
- shadow anonymous swap-backed VM object
- swap pager
- page
- page
- "vm_page"
- Read/copy-on-write, named object

Code
- "vm_object"
- vnode VM object
- vnode pager
- page
- page
- "vm_page"
- Read/copy-on-write, named object

"vmspace", "vm_map"

Lecture 4 - The Process Model (2)
Mach VM in other operating systems

- **Mach**: VM mappings, objects, pages, etc., are first-class kernel services exposed via system calls

- In two directly derived systems, quite different stories:

<table>
<thead>
<tr>
<th>Mac OS X</th>
<th>Although not a microkernel, Mach’s VM/IPC Application Programming Interfaces (APIs) are available to user programs, and widely used for IPC, debugging, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeBSD</td>
<td>Mach VM is used as a foundation for UNIX APIs, but is available for use only as a Kernel Programming Interface (KPI)</td>
</tr>
</tbody>
</table>

- In FreeBSD, Mach is used:
  - To efficiently implement UNIX’s `fork()` and `execve()`
  - For memory-management APIs – e.g., `mmap()` and `mprotect()`
  - By VM-optimised IPC – e.g., `pipe()` and `sendfile()`
  - By the filesystem to implement a **merged VM-buffer cache**
  - By **device drivers** that manage memory in interesting ways (e.g., GPU drivers mapping pages into user processes)
  - By a set of VM worker threads, such as the **page daemon, swapper, syncer**, and **page-zeroing thread**