Virtual Memory

A Project for CS854

Nick Chen
Simon Pratt
Krishna Vaidyanathan

University of Waterloo

February 25, 2016

Abstract

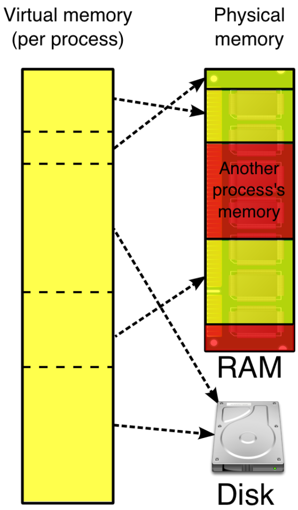
In short:

Abstract

In short:

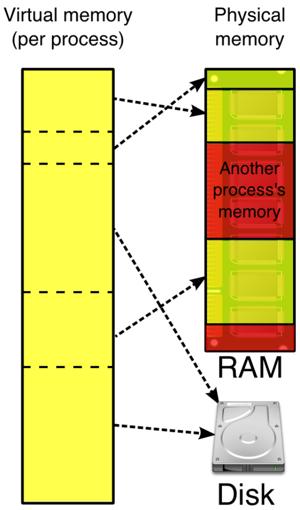
- We propose to study virtual memory!

Background: Virtual Memory



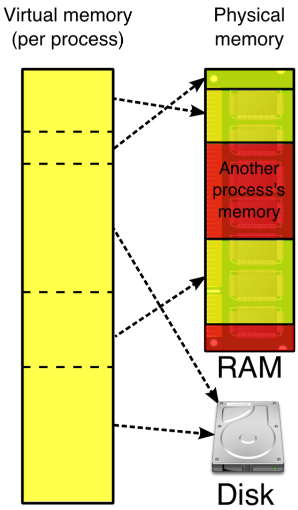
- (on x86) Instructions operate on virtual addresses

Background: Virtual Memory



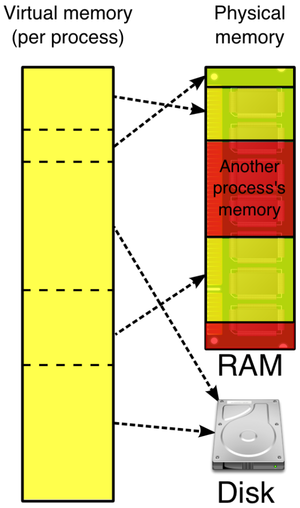
- (on x86) Instructions operate on virtual addresses
- Data may be stored:

Background: Virtual Memory



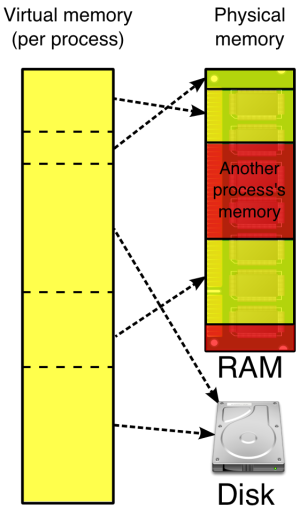
- (on x86) Instructions operate on virtual addresses
- Data may be stored:
  - In physical memory

Background: Virtual Memory



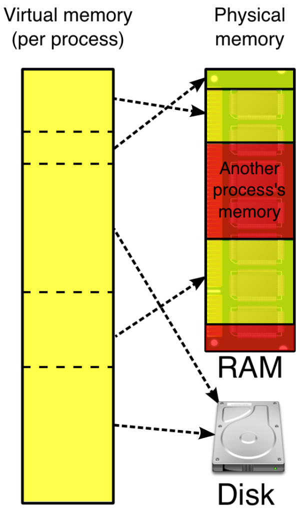
- (on x86) Instructions operate on virtual addresses
- Data may be stored:
  - In physical memory
  - On disk

Background: Virtual Memory



- (on x86) Instructions operate on virtual addresses
- Data may be stored:
  - In physical memory
  - On disk
- Each process has a page table

Background: Virtual Memory



- (on x86) Instructions operate on virtual addresses
- Data may be stored:
  - In physical memory
  - On disk
- Each process has a page table
  - Maps virtual $\rightarrow$ physical addresses

Proposal

Our proposal has 3 parts:

Proposal

Our proposal has 3 parts:

1. Literature Review

Proposal

Our proposal has 3 parts:

1. Literature Review
2. Experimental Design

Proposal

Our proposal has 3 parts:

1. Literature Review
2. Experimental Design
3. Implementation

Proposal

Our proposal has 3 parts:

1. Literature Review
2. Experimental Design
3. Implementation

Proposal: Literature Review

We wish to investigate the following operating systems:

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux
2. NetBSD

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux
2. NetBSD
3. OpenIndiana (Previously Solaris)

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux
2. NetBSD
3. OpenIndiana (Previously Solaris)

For each OS, we wish to answer the following questions:

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux
2. NetBSD
3. OpenIndiana (Previously Solaris)

For each OS, we wish to answer the following questions:

- How is physical memory managed?

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux
2. NetBSD
3. OpenIndiana (Previously Solaris)

For each OS, we wish to answer the following questions:

- How is physical memory managed?
- Are there data structures for physical pages, separate from the page tables?

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux
2. NetBSD
3. OpenIndiana (Previously Solaris)

For each OS, we wish to answer the following questions:

- How is physical memory managed?
- Are there data structures for physical pages, separate from the page tables?
- How are contiguous regions of memory managed?

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux
2. NetBSD
3. OpenIndiana (Previously Solaris)

For each OS, we wish to answer the following questions:

- How is physical memory managed?
- Are there data structures for physical pages, separate from the page tables?
- How are contiguous regions of memory managed?
- How is memory freed?

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux
2. NetBSD
3. OpenIndiana (Previously Solaris)

For each OS, we wish to answer the following questions:

- How is physical memory managed?
- Are there data structures for physical pages, separate from the page tables?
- How are contiguous regions of memory managed?
- How is memory freed?
    - What happens when the kernel runs out of memory?

Proposal: Literature Review

We wish to investigate the following operating systems:

1. Linux
2. NetBSD
3. OpenIndiana (Previously Solaris)

For each OS, we wish to answer the following questions:

- How is physical memory managed?
- Are there data structures for physical pages, separate from the page tables?
- How are contiguous regions of memory managed?
- How is memory freed?
  - What happens when the kernel runs out of memory?
- Do they do anything special on Non-Uniform Memory Access (NUMA) architectures?

Proposal

Our proposal has 3 parts:

1. Literature Review
2. Experimental Design
3. Implementation

Proposal

Our proposal has 3 parts:

1. Literature Review
2. Experimental Design
3. Implementation

Proposal: Experimental Design

Proposal: Experimental Design

- Make a *testable* hypothesis based on lit. review

Proposal: Experimental Design

- Make a *testable* hypothesis based on lit. review
- Design *simple* experiments to test this hypothesis

Proposal: Experimental Design

- Make a *testable* hypothesis based on lit. review
- Design *simple* experiments to test this hypothesis
- Example:

Proposal: Experimental Design

- Make a *testable* hypothesis based on lit. review
- Design *simple* experiments to test this hypothesis
- Example:
    - Implement data structures from different VMs

Proposal: Experimental Design

- Make a *testable* hypothesis based on lit. review
- Design *simple* experiments to test this hypothesis
- Example:
  - Implement data structures from different VMs
  - Test performance

Proposal

Our proposal has 3 parts:

1. Literature Review
2. Experimental Design
3. Implementation

Proposal

Our proposal has 3 parts:

1. Literature Review
2. Experimental Design
3. Implementation

Proposal: Implementation

Proposal: Implementation

- Optional

Proposal: Implementation

- Optional
- Implement a memory management system for KOS

Proposal: Implementation

- Optional
- Implement a memory management system for KOS
- Use findings from:

Proposal: Implementation

- Optional
- Implement a memory management system for KOS
- Use findings from:
    - Lit review

Proposal: Implementation

- Optional
- Implement a memory management system for KOS
- Use findings from:
  - Lit review
  - Experiments

High-level design

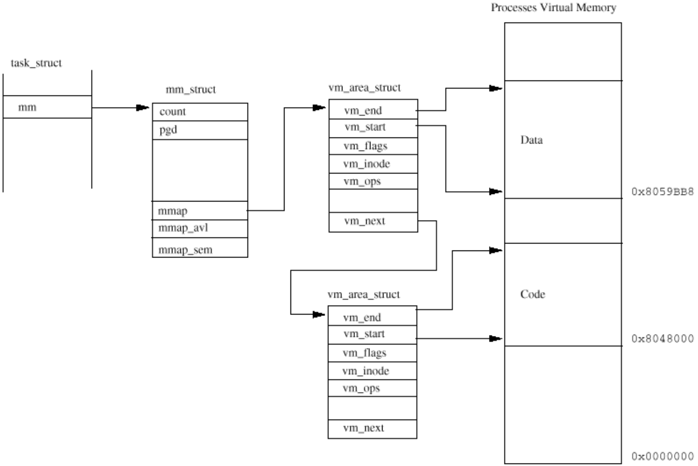
Now we'll summarize the VM design of:

- Linux
- NetBSD
- OpenIndiana

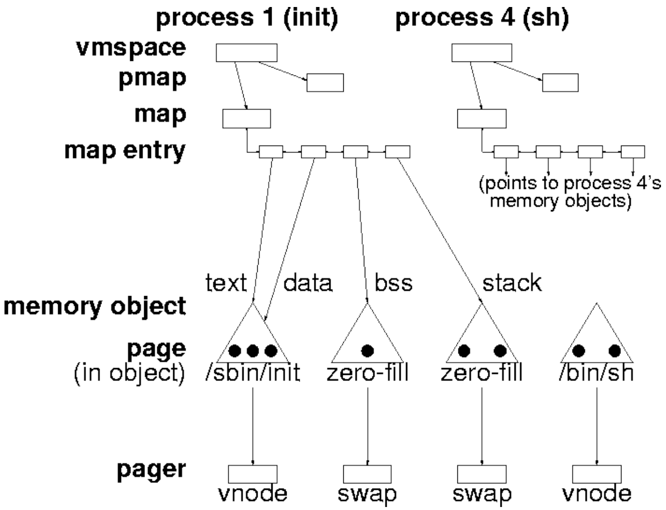
High-level: Linux

- vm_area_struct

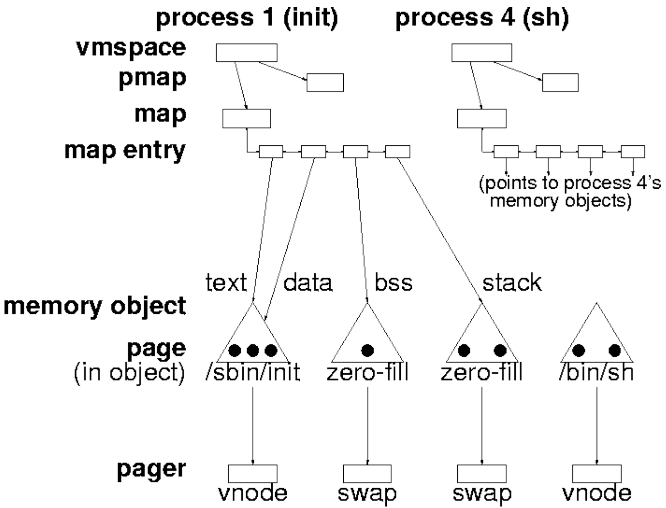
```
struct vm_area_struct {
    struct mm_struct * vm_mm;
    unsigned long vm_start;
    unsigned long vm_end;

    /* linked list of VM areas per task, sorted by address */
    struct vm_area_struct *vm_next;

    pgprot_t vm_page_prot;
    unsigned long vm_flags;

    rb_node_t vm_rb;

    struct vm_area_struct *vm_next_share;
    struct vm_area_struct **vm_pprev_share;

    /* Function pointers to deal with this struct. */
    struct vm_operations_struct * vm_ops;

    /* Information about our backing store: */
    unsigned long vm_pgoff;
    struct file * vm_file;
    unsigned long vm_raend;
    void * vm_private_data;
};
```



High-level: NetBSD

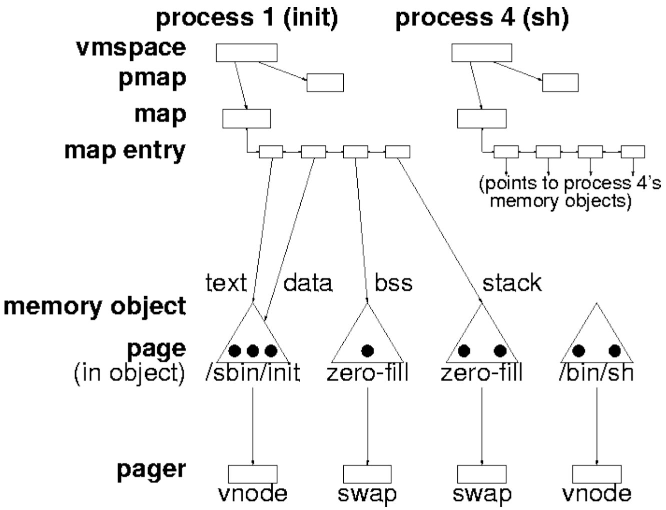


High-level: NetBSD



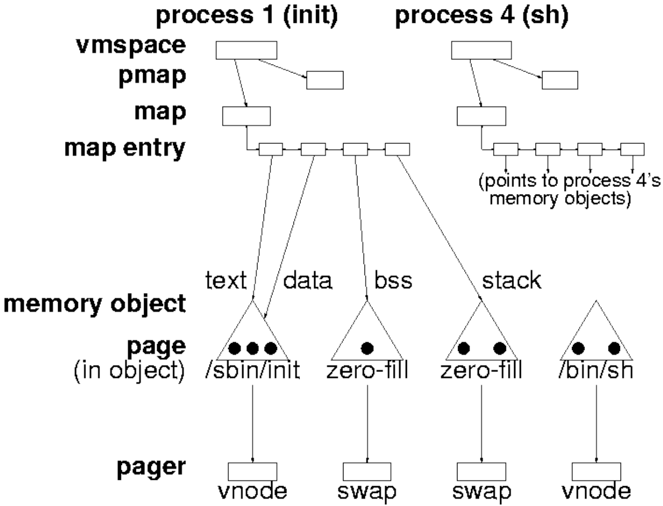
- VM based on:

High-level: NetBSD



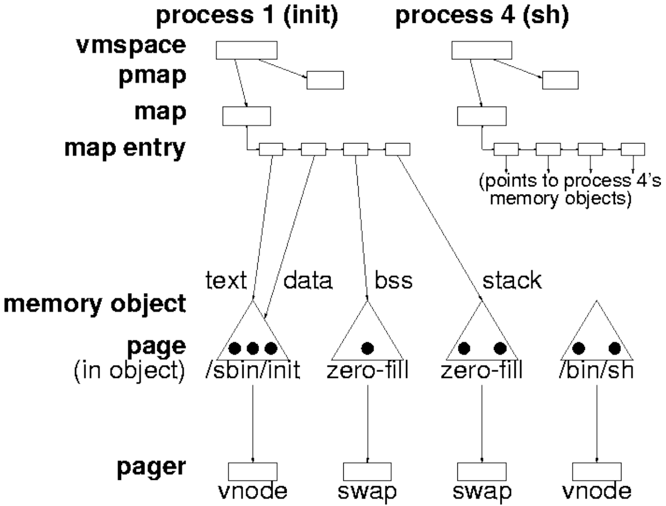
- VM based on:
  - 386BSD

High-level: NetBSD



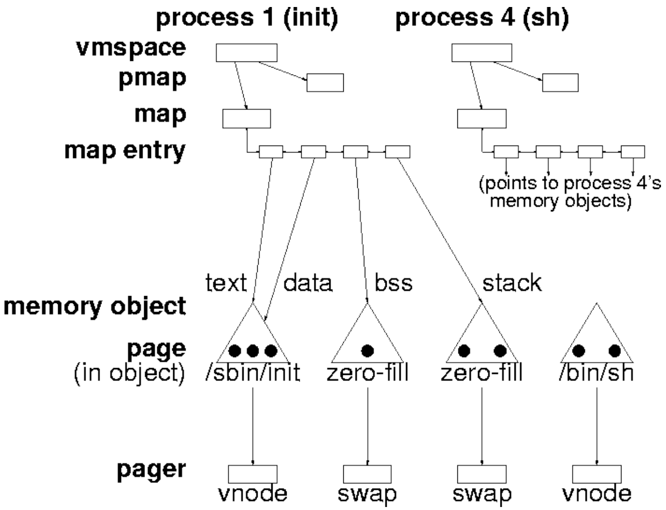
- VM based on:
  - 386BSD
  - 4.4BSD-Lite (Mach)

High-level: NetBSD



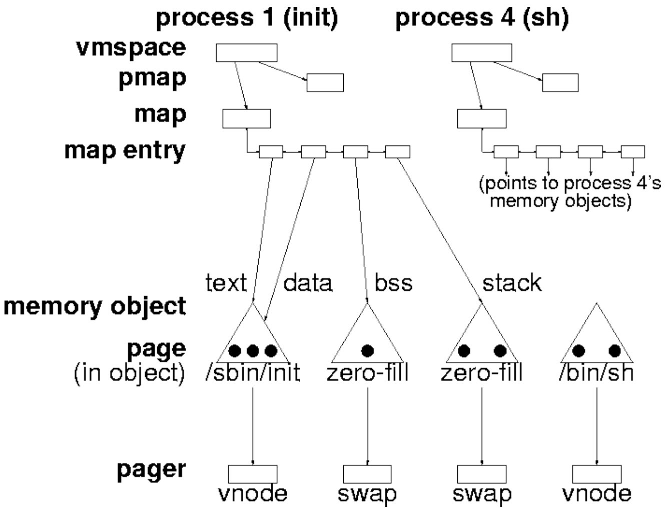
- VM based on:
  - 386BSD
  - 4.4BSD-Lite (Mach)
- Rewritten in 1998 by Chuck Cranor

High-level: NetBSD



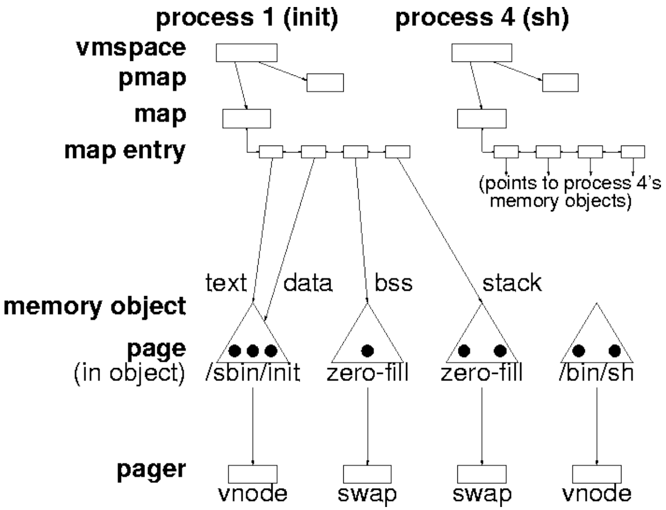
- VM based on:
  - 386BSD
  - 4.4BSD-Lite (Mach)
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)

High-level: NetBSD



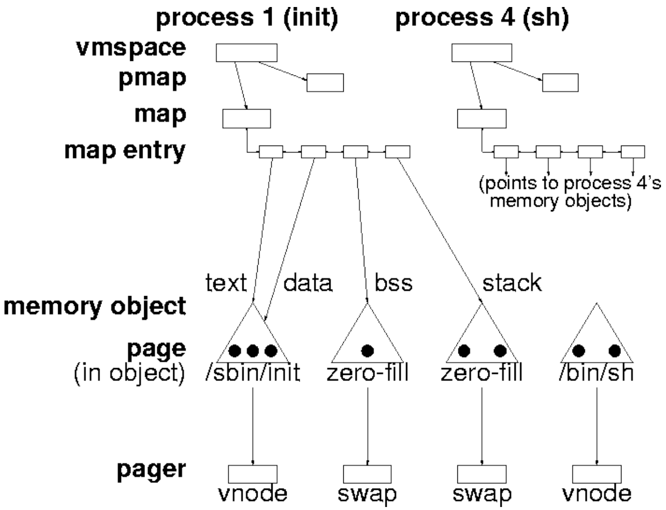
- VM based on:
  - 386BSD
  - 4.4BSD-Lite (Mach)
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper

High-level: NetBSD



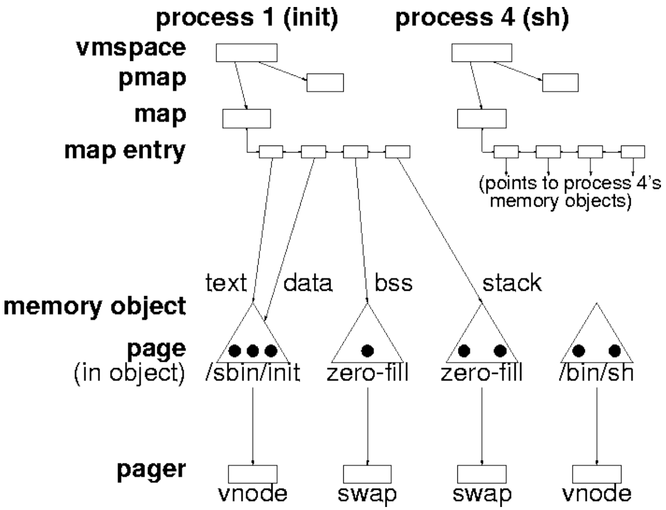
- VM based on:
  - 386BSD
  - 4.4BSD-Lite (Mach)
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper
  - 270 page PhD dissertation

High-level: NetBSD



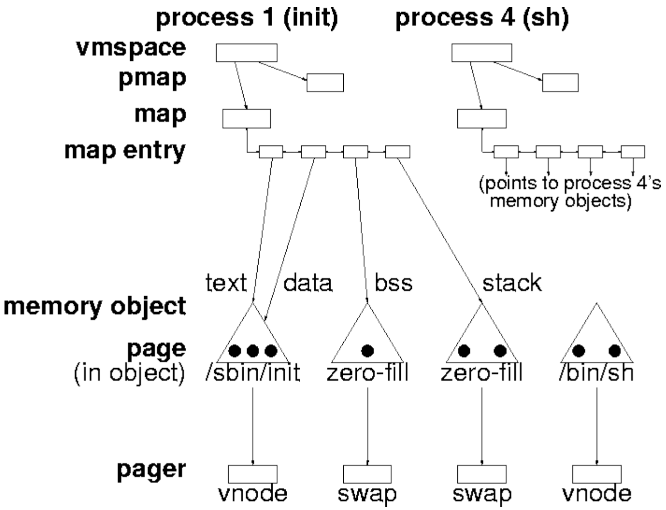
- VM based on:
  - 386BSD
  - 4.4BSD-Lite (Mach)
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper
  - 270 page PhD dissertation
- Slightly modified in 2001 by Chuck Silvers

High-level: NetBSD



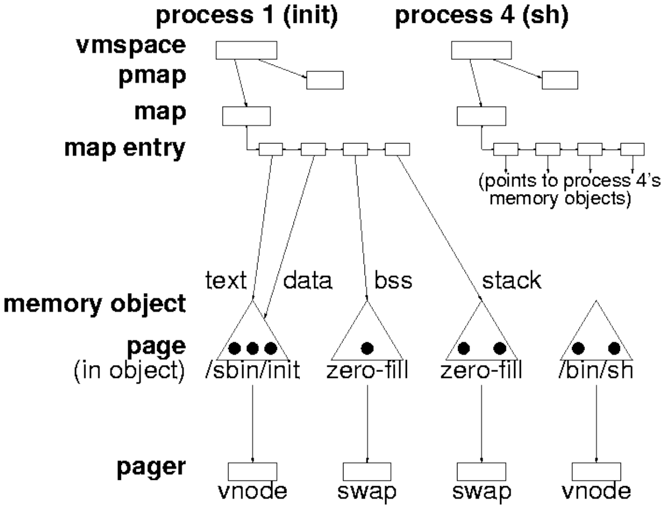
- VM based on:
  - 386BSD
  - 4.4BSD-Lite (Mach)
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper
  - 270 page PhD dissertation
- Slightly modified in 2001 by Chuck Silvers
  - UBC (Unified Buffer Cache)

High-level: NetBSD



- VM based on:
  - 386BSD
  - 4.4BSD-Lite (Mach)
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper
  - 270 page PhD dissertation
- Slightly modified in 2001 by Chuck Silvers
  - UBC (Unified Buffer Cache)
  - 5 page Usenix paper

High-level: NetBSD



- VM based on:
  - 386BSD
  - 4.4BSD-Lite (Mach)
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper
  - 270 page PhD dissertation
- Slightly modified in 2001 by Chuck Silvers
  - UBC (Unified Buffer Cache)
  - 5 page Usenix paper
- Minor modifications since then

History: OpenIndiana

History: OpenIndiana

1. Open source fork of OpenSolaris after Oracle take over

History: OpenIndiana

1. Open source fork of OpenSolaris after Oracle take over
2. Stewarded by the Illumos Foundation

High-level: OpenIndiana

1. Solaris kernel breaks up virtual address space into mappings for each type of memory (eg., heap, stack)

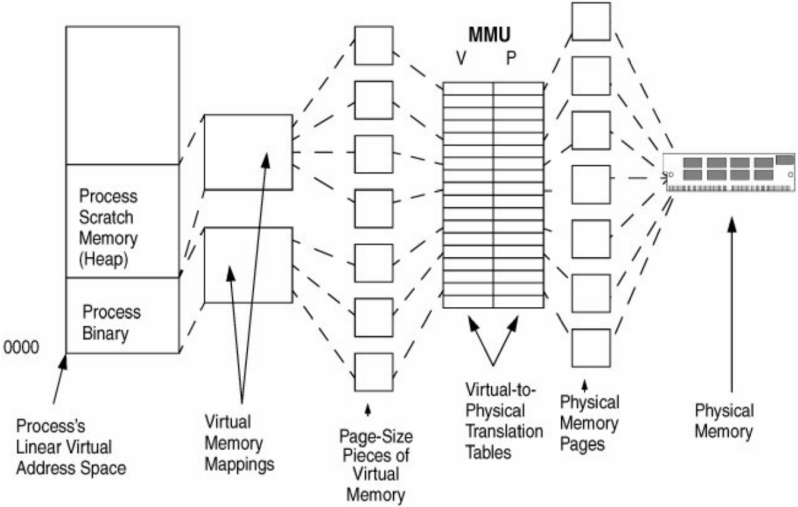
High-level: OpenIndiana

1. Solaris kernel breaks up virtual address space into mappings for each type of memory (eg., heap, stack)
2. Hardware MMU maps pages to physical memory using platform-specific translation tables

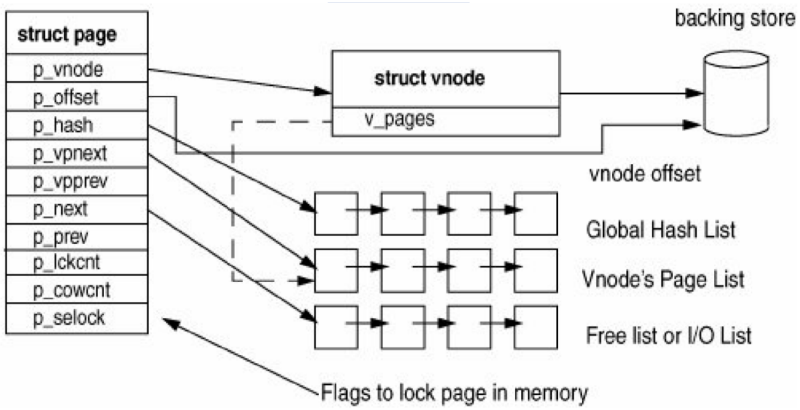
High-level: OpenIndiana

1. Solaris kernel breaks up virtual address space into mappings for each type of memory (eg., heap, stack)
2. Hardware MMU maps pages to physical memory using platform-specific translation tables
3. Memory management to manage pages is basically swapping and demand paging

High-level: OpenIndiana



High-level: OpenIndiana



- Page table structure different from x86 hardware page table structure

Differences

We have found some significant differences so far:

Differences

We have found some significant differences so far:

- What happens when the kernel runs out of memory?

Differences

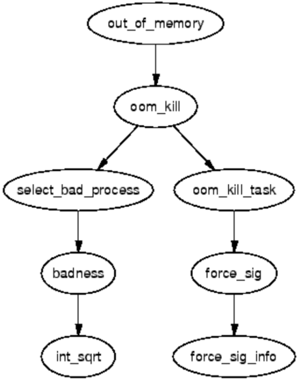
We have found some significant differences so far:

- What happens when the kernel runs out of memory?
- What are the copy-on-write mechanisms?

What happens when the kernel runs out of memory?

Linux:

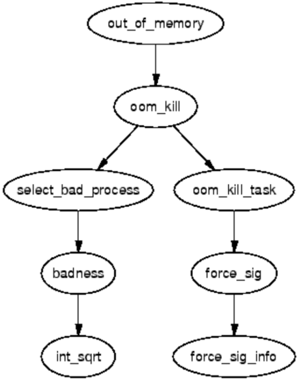
- Start killing processes



What happens when the kernel runs out of memory?

Linux:

- Start killing processes



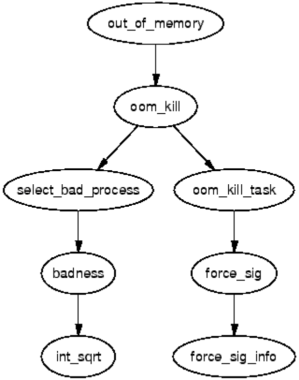
NetBSD:

- Panic!

What happens when the kernel runs out of memory?

Linux:

- Start killing processes



NetBSD:

- Panic!

OpenIndiana:

- Periodically checks kernel space, and "snaps" data to user space if kernel space is low
- If kernel runs out of memory, crashes as far as I can tell

What are the copy-on-write mechanisms?

Linux:

- Page-based copy

What are the copy-on-write mechanisms?

Linux:

- Page-based copy

OpenIndiana:

- Anonymous maps

What are the copy-on-write mechanisms?

Linux:

- Page-based copy

OpenIndiana:

- Anonymous maps

NetBSD:

- Copied SunOS/Solaris

Summary

1. Literature Review
   - High-level design
   - Differences
2. Experimental Design
3. Implementation

References

- UVM dissertation:
  http://vorpal.math.drexel.edu/course/opsys2/uvm-project/uvm.pdf
- UVM paper:
  https://www.usenix.org/legacy/event/usenix99/full_papers/cranor/cranor.pdf
- UBC paper:
  https://www.usenix.org/legacy/publications/library/proceedings/usenix2000/freenix/silvers.html
- *Understanding the Linux Virtual Memory Manager*
  https://www.kernel.org/doc/gorman/html/understand/index.html
- McDougall, Richard, and Jim Mauro. Solaris internals: Solaris 10 and OpenSolaris kernel architecture. Pearson Education, 2006.

Attribution

- Virtual memory diagram by Ehamberg (Own work) [CC BY-SA 3.0 (`http://creativecommons.org/licenses/by-sa/3.0`)], via Wikimedia Commons
- NetBSD data structure diagram from:
`http://usenix.org/legacy/publications/library/proceedings/usenix99/full_papers/cranor/cranor_html/index.html`
- Linux vm_area_struct source from:
`???`
- Linux data structures diagram from:
`???`
- Linux OOM diagram from:
`???`
- Solaris VM diagram:
McDougall, Richard, and Jim Mauro. Solaris internals: Solaris 10 and OpenSolaris kernel architecture. Pearson Education, 2006.

License