Virtual Memory

A Project for CS854

Nick Chen
Simon Pratt
Krishna Vaidyanathan

University of Waterloo

February 24, 2016

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

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

- Implement a memory management system for KOS

Progress: high-level

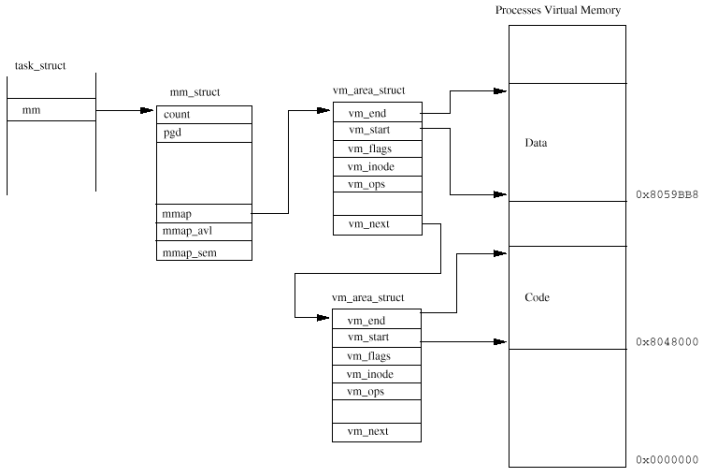
We have made some progress:

- Linux
- NetBSD
- OpenIndiana

Progress: Linux

- vm_area_struct

```
44 struct vm_area_struct {
45     struct mm_struct * vm_mm;
46     unsigned long vm_start;
47     unsigned long vm_end;
48
49
50     /* linked list of VM areas per task, sorted by address */
51     struct vm_area_struct *vm_next;
52
53     pgprot_t vm_page_prot;
54     unsigned long vm_flags;
55
56     rb_node_t vm_rb;
57
63     struct vm_area_struct *vm_next_share;
64     struct vm_area_struct **vm_pprev_share;
65
66     /* Function pointers to deal with this struct. */
67     struct vm_operations_struct * vm_ops;
68
69     /* Information about our backing store: */
70     unsigned long vm_pgoff;
72     struct file * vm_file;
73     unsigned long vm_raend;
74     void * vm_private_data;
75 };
```



Progress: NetBSD

- Based on 386BSD, 4.4BSD-Lite

Progress: NetBSD

- Based on 386BSD, 4.4BSD-Lite
- Rewritten in 1998 by Chuck Cranor

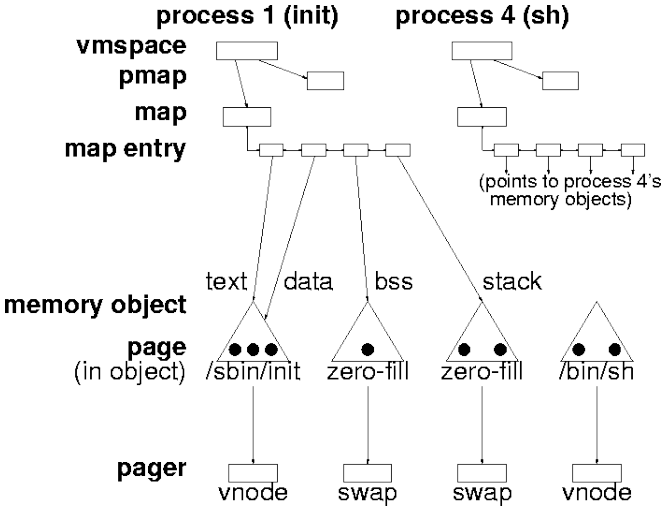
Progress: NetBSD

- Based on 386BSD, 4.4BSD-Lite
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)

Progress: NetBSD

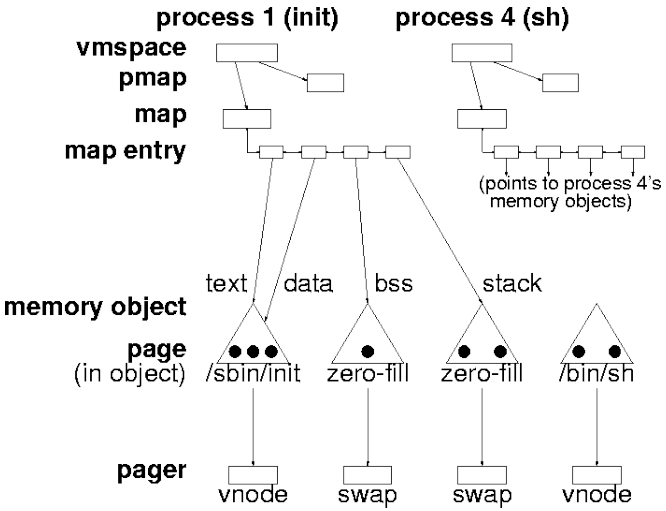
- Based on 386BSD, 4.4BSD-Lite
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper

Progress: NetBSD



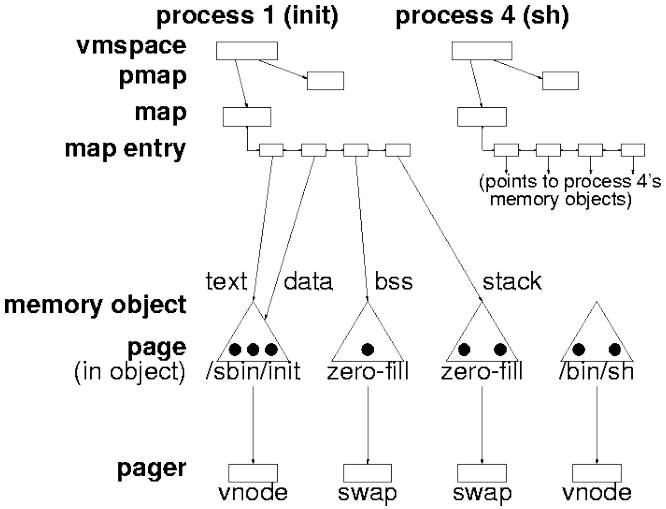
- Based on 386BSD, 4.4BSD-Lite
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper
  - 270 page PhD dissertation

Progress: NetBSD



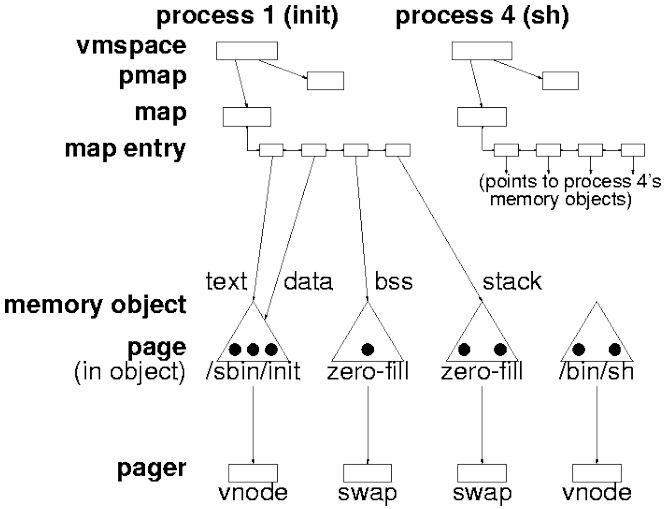
- Based on 386BSD, 4.4BSD-Lite
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper
  - 270 page PhD dissertation
- Slightly modified in 2001 by Chuck Silvers

Progress: NetBSD



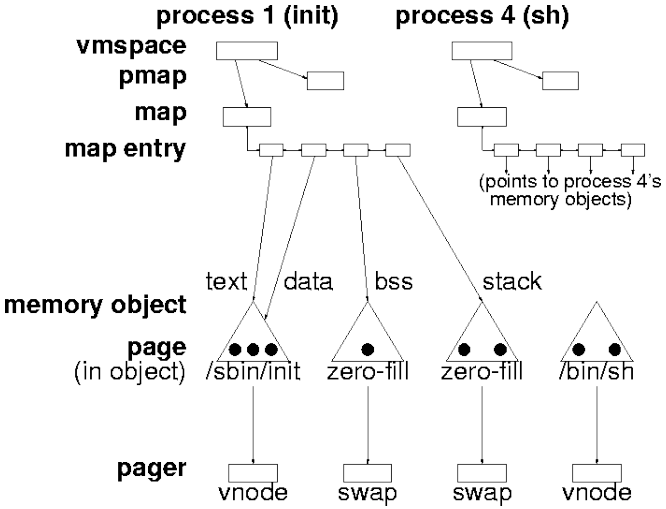
- Based on 386BSD, 4.4BSD-Lite
- Rewritten in 1998 by Chuck Cranor
  - UVM (Unified(?) VM)
  - 13 page Usenix paper
  - 270 page PhD dissertation
- Slightly modified in 2001 by Chuck Silvers
  - UBC (Unified Buffer Cache)

Progress: NetBSD



- Based on 386BSD, 4.4BSD-Lite
- 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

Progress: NetBSD



- Based on 386BSD, 4.4BSD-Lite
- 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

Progress: OpenIndiana

1. Open source fork of OpenSolaris after Oracle take over
2. Stewarded by the Illumos Foundation
3. VM uses the ast package by AT&T, written by Kiem-Phong Vo
4. Based on paper "Vmalloc: A General and Efficient Memory Allocator"

Progress: OpenIndiana

1. Legacy malloc function is old, has shortcomings
2. malloc not designed for modern environments
3. Vmalloc a memory allocation library that is flexible and allows a wide range of memory operations
    1. Regions to organize memory
    2. Obtain memory by application definable disciplines
    3. Customize memory management

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?
- How does the kernel access user memory?

Differences

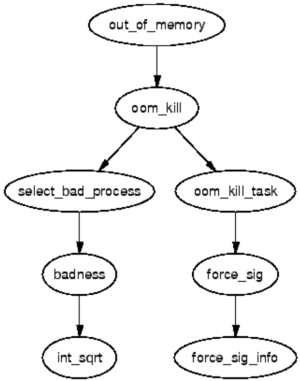
We have found some significant differences so far:

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

What happens when the kernel runs out of memory?

Linux:

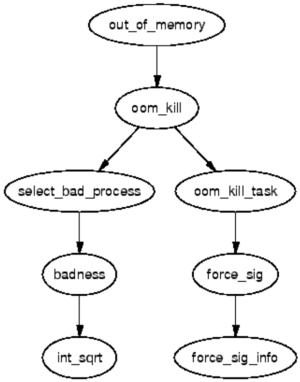
- Start killing processes



What happens when the kernel runs out of memory?

Linux:

- Start killing processes



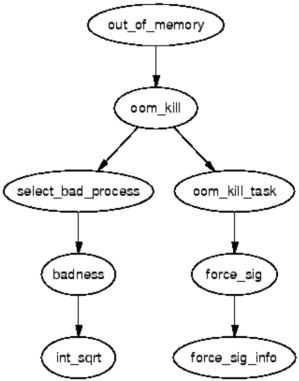
NetBSD:

- Panic!

What happens when the kernel runs out of memory?

Linux:

- Start killing processes



NetBSD:

- Panic!

OpenIndiana:

- ???

How does the kernel access user memory?

Linux:

- Map all of physical memory into kernel's virtual address space

How does the kernel access user memory?

Linux:

- Map all of physical memory into kernel's virtual address space

NetBSD:

- ???

How does the kernel access user memory?

Linux:

- Map all of physical memory into kernel's virtual address space

NetBSD:

- ???

OpenIndiana:

- ???

What are the copy-on-write mechanisms?

Linux:

- ???

What are the copy-on-write mechanisms?

Linux:

- ???

NetBSD:

- Anonymous maps

What are the copy-on-write mechanisms?

Linux:

- ???

NetBSD:

- Anonymous maps

OpenIndiana:

- ???

Summary

1. Literature Review
   - High-level
   - 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
- Vmalloc: A General and Efficient Memory Allocator:
  http://onlinelibrary.wiley.com/doi/10.1002/%28SICI%291097-024X%28199603%2926:3%3C357::AID-SPE15%3E3.0.CO;2-%23/abstract

Attribution

- 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:

  ???

License