

Virtual Memory

A Project for CS854

Nick Chen
Simon Pratt
Krishna Vaidyanathan

University of Waterloo

February 24, 2016

Our proposal has 3 parts:

Our proposal has 3 parts:

- 1 Literature Review

Our proposal has 3 parts:

- ① Literature Review
- ② Experimental Design

Our proposal has 3 parts:

- ① Literature Review
- ② Experimental Design
- ③ Implementation

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 OpenBSD

Proposal: Literature Review

We wish to investigate the following operating systems:

- 1 Linux
- 2 OpenBSD
- 3 OpenIndiana
(Previously Solaris)

Proposal: Literature Review

We wish to investigate the following operating systems:

- 1 Linux
- 2 OpenBSD
- 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 OpenBSD
- 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 OpenBSD
- 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 OpenBSD
- 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 OpenBSD
- 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 OpenBSD
- 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 OpenBSD
- 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?

Our proposal has 3 parts:

- ① Literature Review
- ② Experimental Design
- ③ Implementation

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

Our proposal has 3 parts:

- ① Literature Review
- ② Experimental Design
- ③ Implementation

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

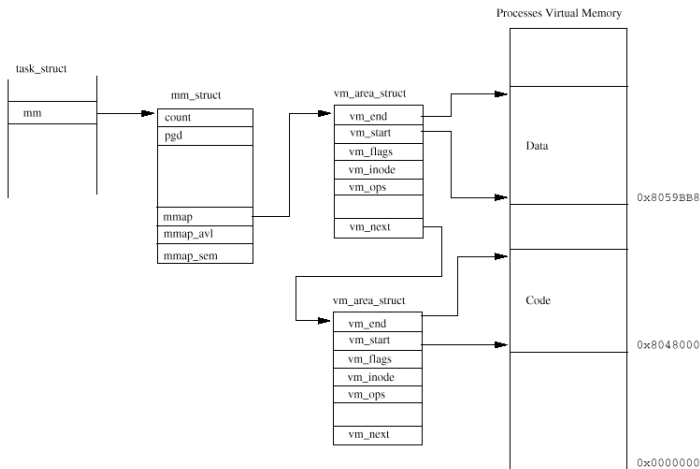
We have made some progress:

- Linux
- OpenBSD
- 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     /* linked list of VM areas per task, sorted by address */
50     struct vm_area_struct *vm_next;
51
52     pgprot_t vm_page_prot;
53     unsigned long vm_flags;
54
55     rb_node_t vm_rb;
56
57     struct vm_area_struct *vm_next_share;
58     struct vm_area_struct **vm_pprev_share;
59
60     /* Function pointers to deal with this struct. */
61     struct vm_operations_struct * vm_ops;
62
63     /* Information about our backing store: */
64     unsigned long vm_pgoff;
65     struct file * vm_file;
66     unsigned long vm_raend;
67     void * vm_private_data;
68 };
```



Progress: OpenBSD

- 386BSD



NetBSD



OpenBSD

Progress: OpenBSD

- 386BSD



NetBSD



OpenBSD

- VM based on NetBSD

Progress: OpenBSD

- 386BSD



NetBSD



OpenBSD

- VM based on NetBSD
 - Rewritten in 1998

Progress: OpenBSD

- 386BSD



NetBSD

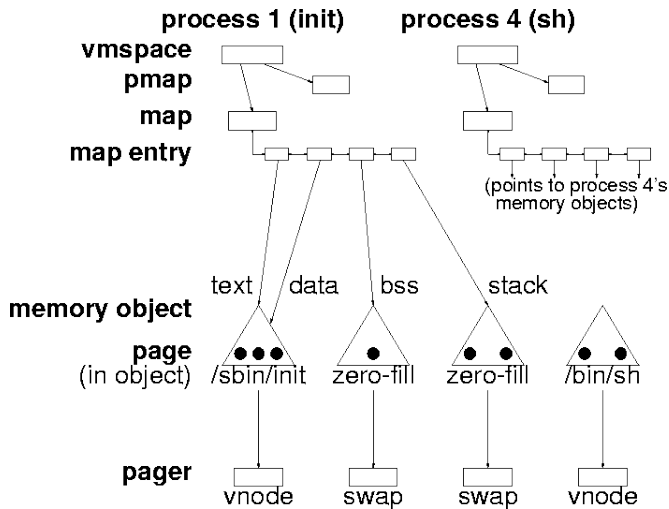


OpenBSD

- VM based on NetBSD
 - Rewritten in 1998
 - 270 page PhD dissertation

Progress: OpenBSD

- 386BSD
 - ↓
- NetBSD
 - ↓
- OpenBSD
- VM based on NetBSD
 - Rewritten in 1998
 - 270 page PhD dissertation



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

We have found some significant differences so far:

- What happens when the kernel runs out of memory
- How the kernel accesses user memory
- Copy-on-write mechanisms

What happens when the kernel runs out of memory

Linux:

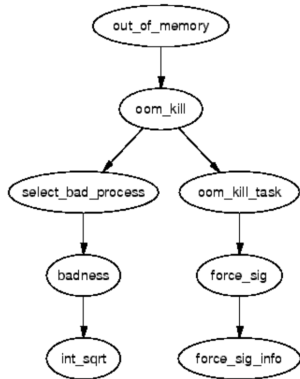
- Start killing processes

BSD:

- Panic!

OpenIndiana:

- ???



How the kernel accesses user memory

Linux:

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

BSD:

- ???

OpenIndiana:

- ???

Copy-on-write mechanisms

Linux:

- ???

BSD:

- Anonymous maps

OpenIndiana:

- ???

Summary

- ① Literature Review
 - Progress: High-level
 - Progress: Differences
- ② Experimental Design
- ③ Implementation

References

- UVM dissertation:
<http://vorpai.math.drexel.edu/course/opsys2/uvm-project/uvm.pdf>
- *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>

- OpenBSD data structure diagram from:
http://usenix.org/legacy/publications/library/proceedings/usenix99/full_papers/cranor/cranor_html/index.html

- These slides are distributed under the creative commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0).
- See <http://creativecommons.org/licenses/by-sa/4.0/> for details.