

RadixVM

Scalable address spaces for multithreaded applications

Austin T. Clements,
M. Frans Kaashoek,
Nickolai Zeldovich

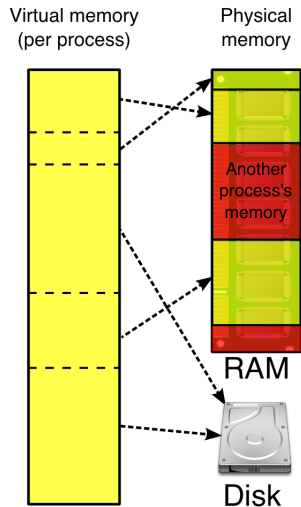
Presented by Simon Pratt

February 12, 2016

RadixVM is a virtual memory (VM) design that attempts to increase multithreaded scalability by:

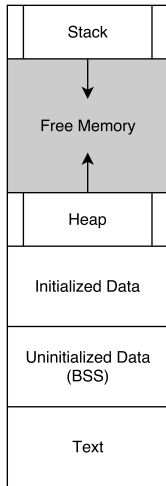
- Storing VM information in a radix tree
- Counting references to memory addresses
- Reducing inter-core virtual address invalidation (remote TLB shutdown)

Background: Virtual Memory



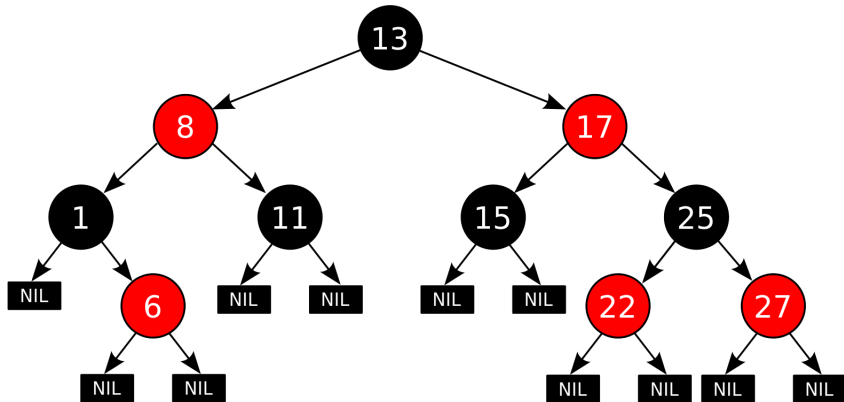
- Maps a contiguous virtual address space to:
 - physical memory (frames)
 - disk (swap)

Background: `malloc` and `mmap`



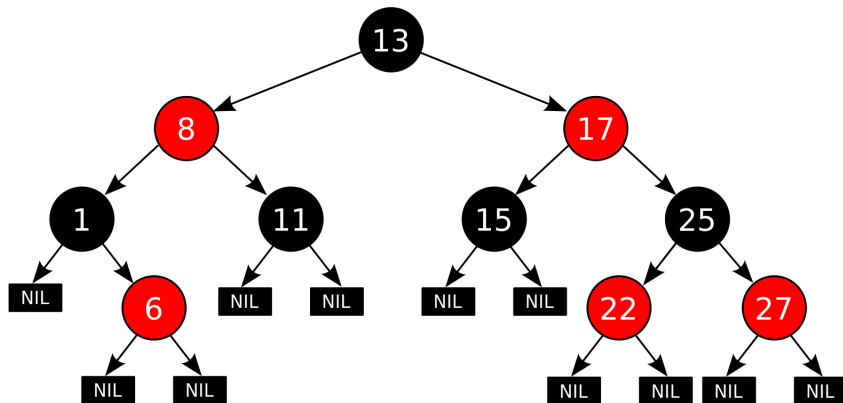
- `malloc` and `free`
 - User-level library function
 - Allocates/frees space in virtual memory
 - Often implemented using `mmap` and `munmap`
- `mmap` and `munmap`
 - System calls
 - Actually allocates/frees space in virtual memory

Background: Linux Virtual Memory



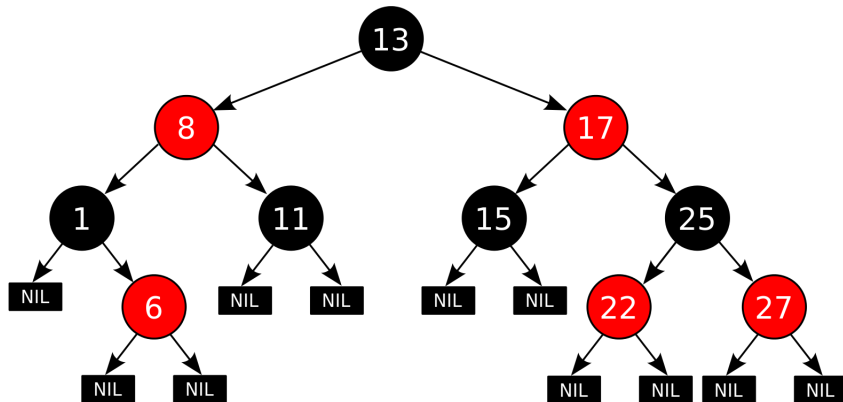
- Red-black tree
- Allows the kernel to search for memory area covering a virtual address

Background: Linux Virtual Memory



- Red-black tree
- Allows the kernel to search for memory area covering a virtual address
- **Problem: A single lock per address space!**

Aside: Psearchy



- A single lock on this structure → mmap within a single process is serialized
- This is probably why the prwlock paper notes that Psearchy is mmap-intensive

RadixVM has 3 parts:

RadixVM has 3 parts:

- Refcache

RadixVM has 3 parts:

- Refcache
- Radix-tree-like data structure

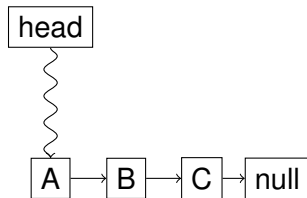
RadixVM has 3 parts:

- Refcache
- Radix-tree-like data structure
- Targeted TLB shootdowns

RadixVM has 3 parts:

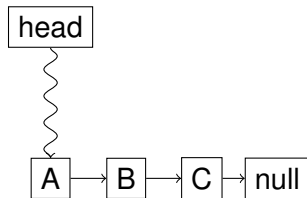
- Refcache
- Radix-tree-like data structure
- Targeted TLB shootdowns

Background: ABA Problem



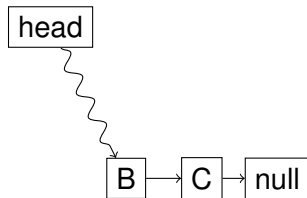
- Process P_1 reads values A, B in order to pop A

Background: ABA Problem



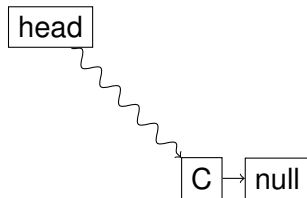
- Process P_1 reads values A, B in order to pop A
- P_1 is preempted

Background: ABA Problem



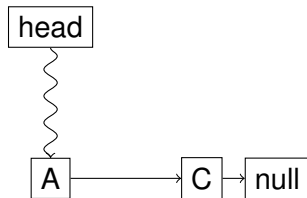
- Process P_1 reads values A, B in order to pop A
- P_1 is preempted
- P_2 pops A , sets head to B

Background: ABA Problem



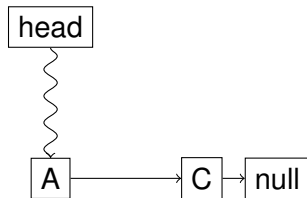
- Process P_1 reads values A, B in order to pop A
- P_1 is preempted
- P_2 pops A , sets head to B
- P_2 pops B , sets head to C

Background: ABA Problem



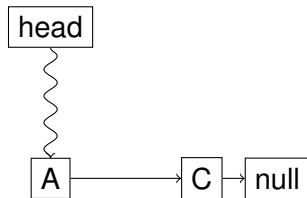
- Process P_1 reads values A, B in order to pop A
- P_1 is preempted
- P_2 pops A , sets head to B
- P_2 pops B , sets head to C
- P_2 pushes A , sets head to A

Background: ABA Problem



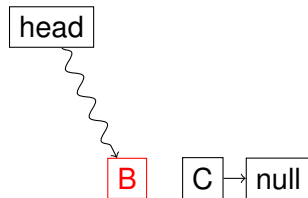
- Process P_1 reads values A, B in order to pop A
- P_1 is preempted
- P_2 pops A , sets head to B
- P_2 pops B , sets head to C
- P_2 pushes A , sets head to A
- P_2 is preempted

Background: ABA Problem



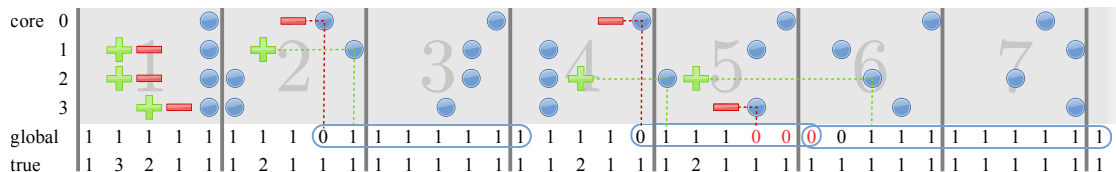
- Process P_1 reads values A, B in order to pop A
- P_1 is preempted
- P_2 pops A , sets head to B
- P_2 pops B , sets head to C
- P_2 pushes A , sets head to A
- P_2 is preempted
- P_1 reads value A , assumes nothing has changed

Background: ABA Problem



- Process P_1 reads values A, B in order to pop A
- P_1 is preempted
- P_2 pops A , sets head to B
- P_2 pops B , sets head to C
- P_2 pushes A , sets head to A
- P_2 is preempted
- P_1 reads value A , assumes nothing has changed
- P_1 pops A , sets head to B

Design: Refcache



RadixVM has 3 parts:

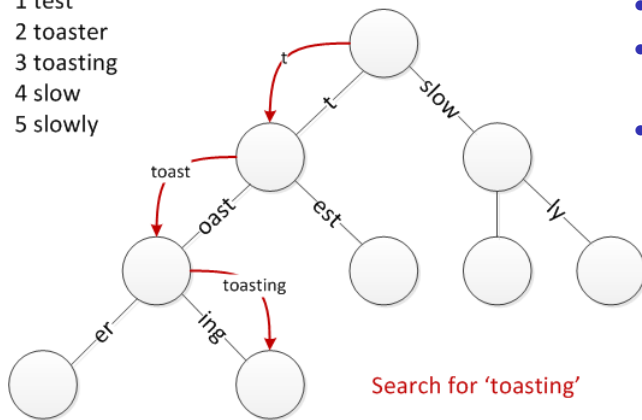
- Refcache
- Radix-tree-like data structure
- Targeted TLB shootdowns

RadixVM has 3 parts:

- Refcache
- Radix-tree-like data structure
- Targeted TLB shootdowns

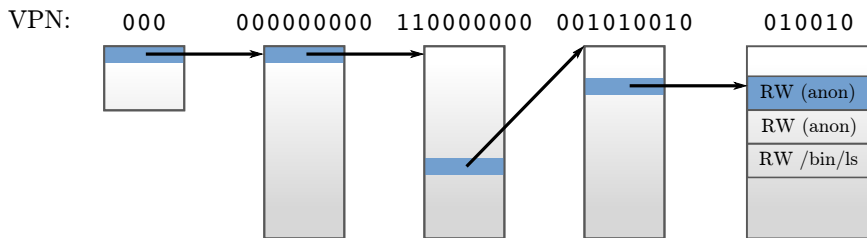
Background: Radix Tree

1 test
2 toaster
3 toasting
4 slow
5 slowly



- A.K.A. prefix tree
- Edges labeled
- Concatenation of edge labels along root→node path gives a string
- In OSes, usually strings of bits

Design: RadixVM Data Structure



- Similar to a radix-tree
- Fixed-height
- Each level indexed by up to 9 bits

RadixVM has 3 parts:

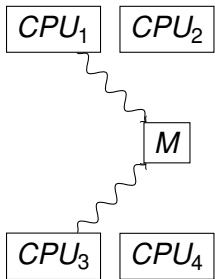
- Refcache
- Radix-tree-like data structure
- Targeted TLB shootdowns

RadixVM has 3 parts:

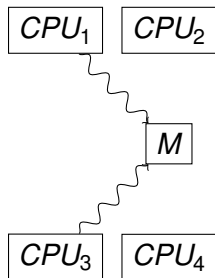
- Refcache
- Radix-tree-like data structure
- Targeted TLB shutdowns

Background: Remote TLB Shootdowns

- Processes on CPU_1 and CPU_3 share memory area M

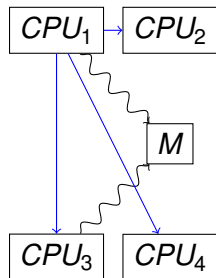


Background: Remote TLB Shootdowns



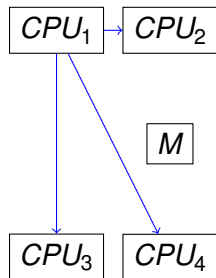
- Processes on CPU_1 and CPU_3 share memory area M
- A process on CPU_1 unmaps M

Background: Remote TLB Shootdowns



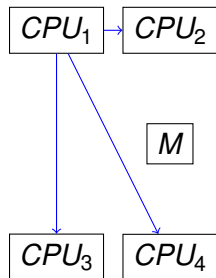
- Processes on CPU_1 and CPU_3 share memory area M
- A process on CPU_1 unmaps M
- The kernel sends a message to all CPU to **flush** their TLBs

Background: Remote TLB Shootdowns



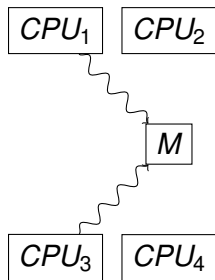
- Processes on CPU_1 and CPU_3 share memory area M
- A process on CPU_1 unmaps M
- The kernel sends a message to all CPU to **flush** their TLBs

Background: Remote TLB Shootdowns



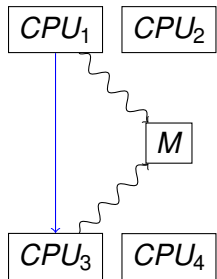
- Processes on CPU_1 and CPU_3 share memory area M
- A process on CPU_1 unmaps M
- The kernel sends a message to all CPU to flush their TLBs
- This is expensive!

Design: Targeted TLB Shootdowns



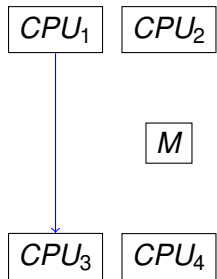
- Store metadata on which cores may have address in TLB

Design: Targeted TLB Shootdowns



- Store metadata on which cores may have address in TLB
- Only **flush** TLBs on cores which may share that memory

Design: Targeted TLB Shootdowns

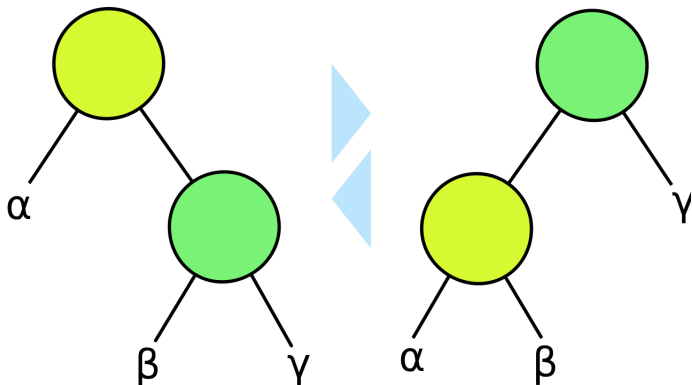


- Store metadata on which cores may have address in TLB
- Only **flush** TLBs on cores which may share that memory

Implementation

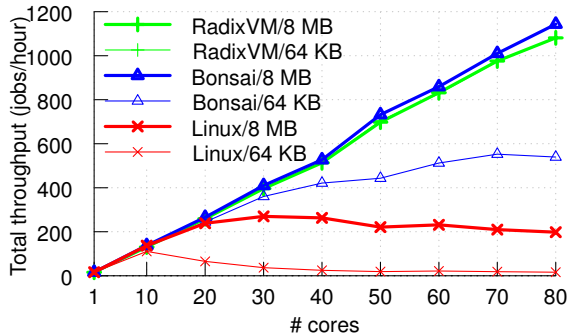
- Implemented on xv6
 - Academic OS
 - Based on v6 Unix
 - Rewritten in ANSI C for x86
 - <https://pdos.csail.mit.edu/6.828/2014/xv6.html>

Background: Bonsai



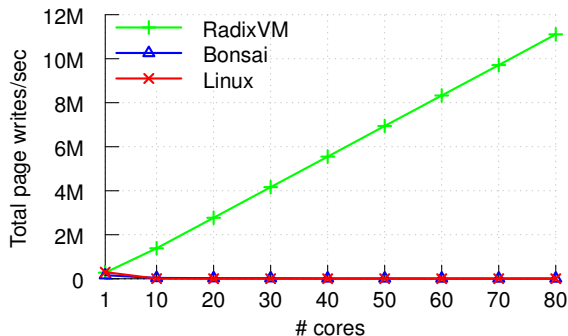
- Designed by the same authors
- “Soft” page faults happen in parallel using RCU
- Uses an RCU-based balanced binary tree
- Maintains bounded balance rather than strict balance (this means fewer rotations)
- Rotations construct a new subtree rather than mutate the old one

Application: Metis



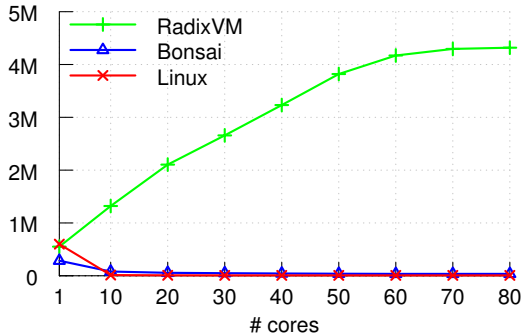
- MapReduce Library
- Single-server
- Multithreaded
- Stresses concurrent `mmaps` and `pagefaults`, but not concurrent `munmaps`
- Compiles on `xv6` and `linux`

Microbenchmark: Local



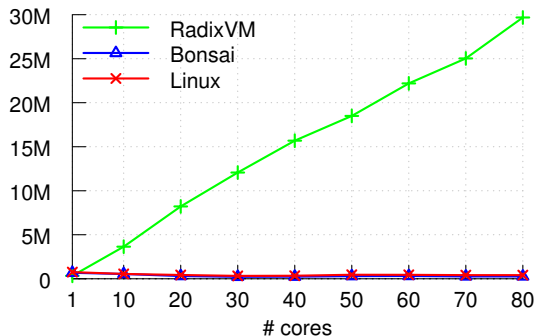
- `mmap` a private 4KB region in shared address space
- Write to every page in region
- `munmap` region

Microbenchmark: Pipeline



- Each thread `mmap` a region
- Write to every page in region
- Pass region to next thread
- Write to every page in passed region
- `munmap` region

Microbenchmark: Global

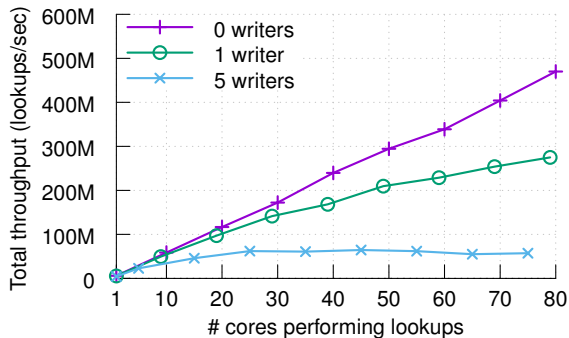


- Each thread `mmap` a 64KB region within a large region of memory
- All threads access all pages in random order

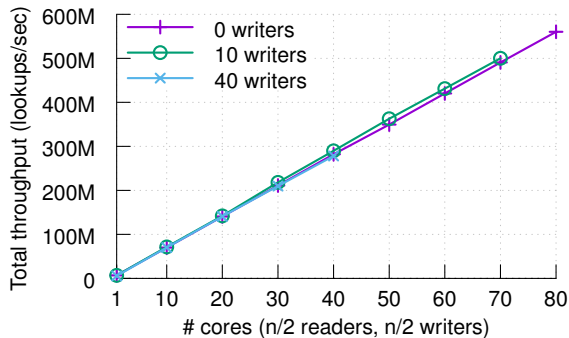
Do we really need all 3 pieces?

- Radix trees
- Refcache
- Targeted TLB shutdown

Question: Do we need radix trees?

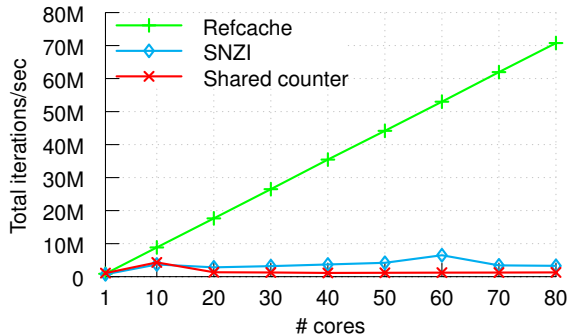


Skip lists

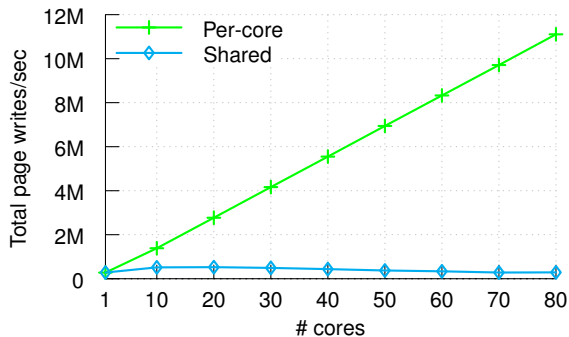


Radix trees

Question: Do we need Refcache?

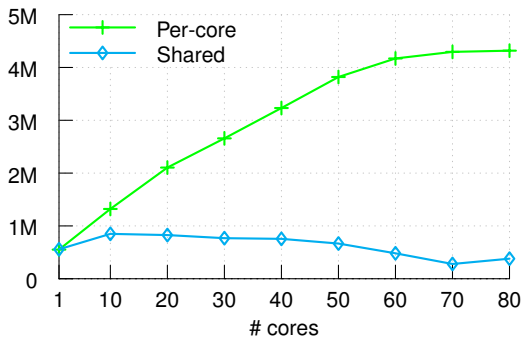


Question: Do we need targeted TLB shutdown?



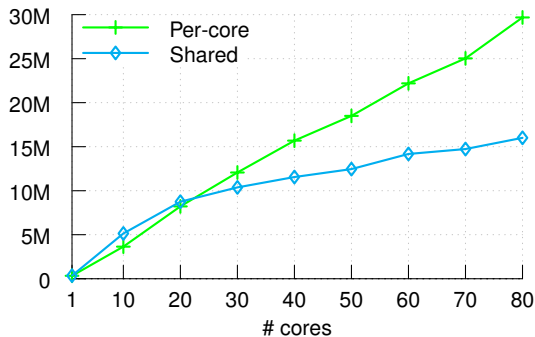
Local microbenchmark, per-core versus shared

Question: Do we need targeted TLB shutdown?



Pipeline microbenchmark, per-core versus shared

Question: Do we need targeted TLB shutdown?



Global microbenchmark, per-core versus shared

Memory Overhead

	RSS	Linux		Radix tree
		VMA tree	Page table	(rel. to Linux)
Firefox	352 MB	117 KB	1.5 MB	3.9 MB (2.4×)
Chrome	152 MB	124 KB	1.1 MB	2.4 MB (2.0×)
Apache	16 MB	44 KB	368 KB	616 KB (1.5×)
MySQL	84 MB	18 KB	348 KB	980 KB (2.7×)

- RSS
 - Resident Set Size
 - physical memory used by a process
- VMA
 - Virtual Memory Areas
 - stored in a red-black tree in Linux

Summary

TODO

- TODO

References

- Clements, Austin T., M. Frans Kaashoek, and Nickolai Zeldovich. "RadixVM: Scalable address spaces for multithreaded applications." In *Proceedings of the 8th ACM European Conference on Computer Systems*, pp. 211-224. ACM, 2013.
 - Revised version: <https://pdos.csail.mit.edu/papers/radixvm:eurosys13-2014-08-05.pdf>
- Clements, Austin T., M. Frans Kaashoek, and Nickolai Zeldovich. "Scalable address spaces using RCU balanced trees." *ACM SIGPLAN Notices* 47, no. 4 (2012): 199-210.
 - Available online: <https://pdos.csail.mit.edu/papers/rcuvm:asplos12.pdf>
- Linux VM info from:
<http://duartes.org/gustavo/blog/post/how-the-kernel-manages-your-memory/>

Attribution

- Refcache diagram, radix data structure diagram, and RadixVM charts used with permission by Austin Clements
- Virtual memory diagram by Ehamberg (Own work) [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0>) or GFDL (<http://www.gnu.org/copyleft/fdl.html>)], via Wikimedia Commons
- Address space diagram by Majenko (Own work) [CC BY-SA 4.0 (<http://creativecommons.org/licenses/by-sa/4.0>)], via Wikimedia Commons
- Patricia trie diagram by Saffles (Microsoft Visio) [GFDL (<http://www.gnu.org/copyleft/fdl.html>) or CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0>)], via Wikimedia Commons
- Binary tree rotation diagram by Josell7 (Own work) [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0>)], via Wikimedia Commons

- 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.