Paging to Disk

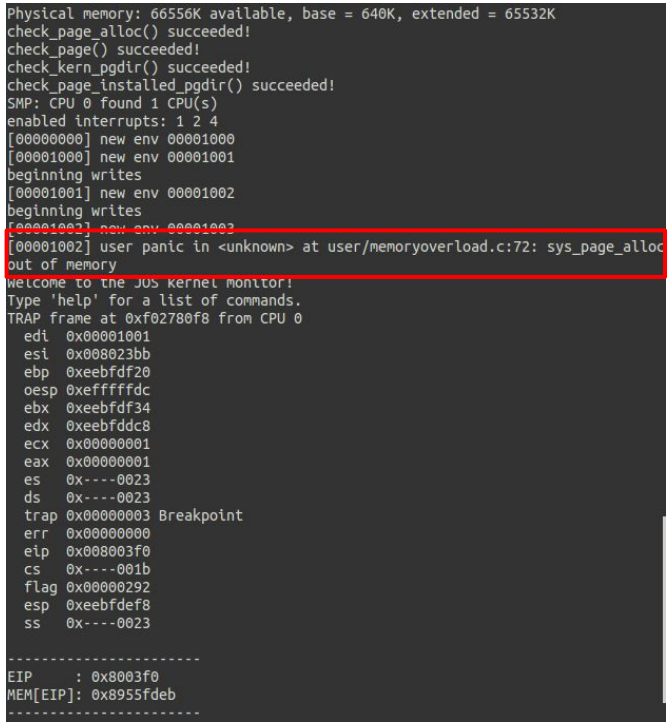
Henry Peteet, Millad Asgharneya, Premkumar Saravanan

Problem statement revisited

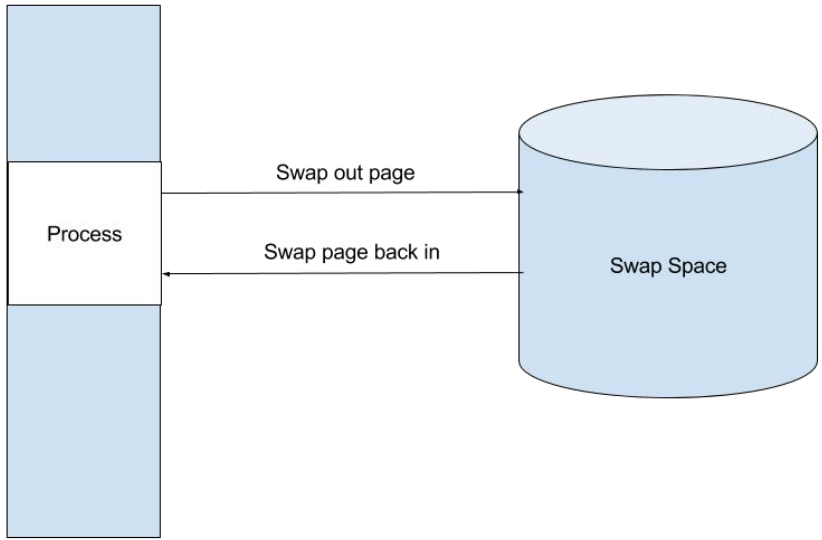
Our configuration of JOS has 64M of physical memory.

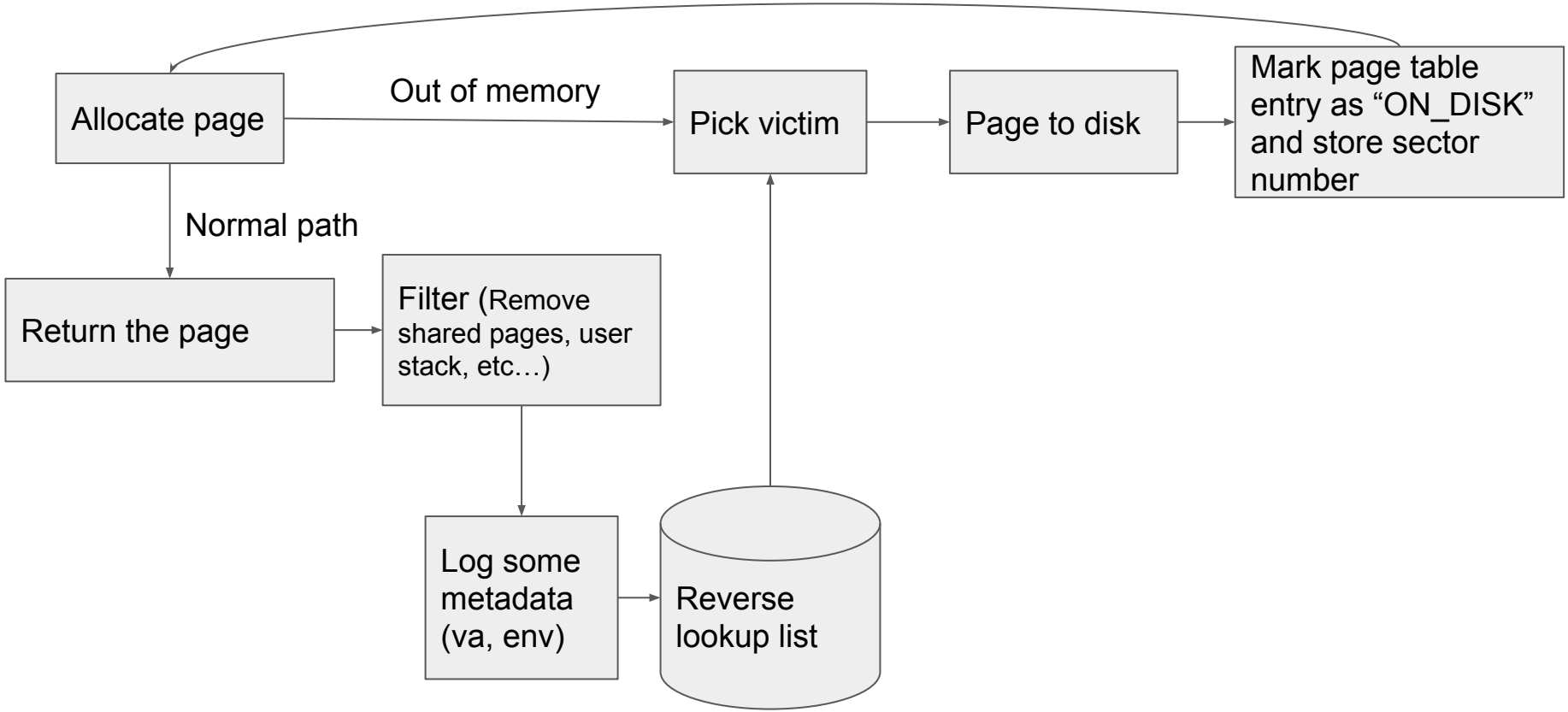
If you use more than 64M the OS will kill the environment.

We added a panic just to highlight the issue.

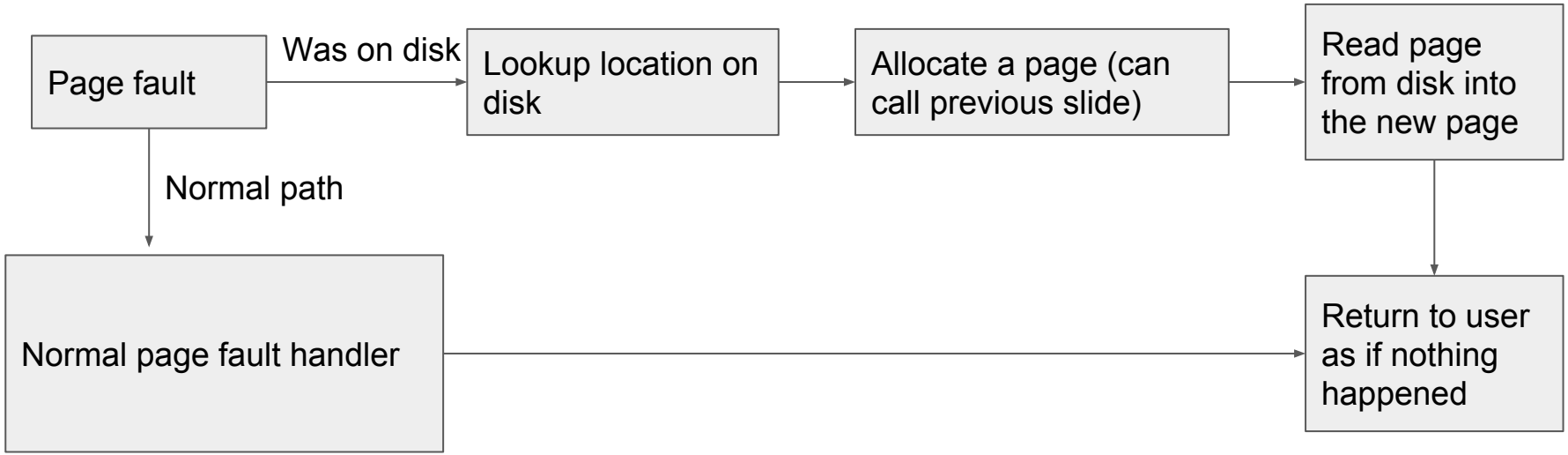


How did we address it?




Allocate page
Out of memory
Pick victim
Page to disk
Mark page table entry as “ON_DISK” and store sector number
Normal path
Return the page
Filter (Remove shared pages, user stack, etc…)
Log some metadata (va, env)
Reverse lookup list

Implementation



Testing

We wrote a test program that mimics dumbfork and writes/reads a bunch of memory guaranteeing that all environments stay active the entire time.

With this we were able to break the old version of JOS, and see a successful run on our modified version when we try to use 64M of memory (since the kernel uses some of it as well)

Testing results

Original

```
Physical memory: 66556K available, base = 640K, extended = 65532K
check_page_alloc() succeeded!
check_page() succeeded!
check_kern_pgdir() succeeded!
check_page_installed_pgdir() succeeded!
SMP: CPU 0 found 1 CPU(s)
enabled interrupts: 1 2 4
[00000000] new env 00001000
[00001000] new env 00001001
beginning writes
[00001001] new env 00001002
beginning writes
[00001002] new env 00001003
[00001002] user panic in <unknown> at user/memoryoverload.c:72: sys_page_alloc
out of memory
Welcome to the JOS kernel monitor!
Type 'help' for a list of commands.
TRAP frame at 0xf02780f8 from CPU 0
  edi  0x00001001
  esi  0x008023bb
  ebp  0xeebfdf20
  oesp 0xefffffdc
  ebx  0xeebfdf34
  edx  0xeebfddc8
  ecx  0x00000001
  eax  0x00000001
  es   0x----0023
  ds   0x----0023
  trap 0x00000003 Breakpoint
  err  0x00000000
  eip  0x008003f0
  cs   0x----001b
  flag 0x00000292
  esp  0xeebfdef8
  ss   0x----0023

----------------------
EIP     : 0x8003f0
MEM[EIP]: 0x8955fdeb
----------------------
```

Ours

```
Physical memory: 64M available(16639 pages), base = 640K, extended = 65532K
SMP: CPU 0 found 1 CPU(s)
enabled interrupts: 1 2 4
Device 1 presence: 1
using disk 1
NBLOCKS=32768
free_block_bitmap size = 32800
[00000000] new env 00001000
[00001000] new env 00001001
beginning writes
[00001001] new env 00001002
beginning writes
[00001002] new env 00001003
beginning writes
beginning writes
base case done in env 1003
[00001003] exiting gracefully
[00001003] free env 00001003
1002 sending to 1001
[00001002] exiting gracefully
[00001002] free env 00001002
1001 sending to 1000
[00001001] exiting gracefully
[00001001] free env 00001001
[00001000] exiting gracefully
[00001000] free env 00001000
No runnable environments in the system!
Welcome to the JOS kernel monitor!
Type 'help' for a list of commands.
K>
```

Results

1. We can run environments that use more than 64M
2. We can evict from other environments (so if we can start new environments even when memory is full)
3. We use shared swap space

Limitations

1. Disk space
2. FIFO limitations
    a. Commonly pages out pages that are used heavily
    b. Can easily lead to more and more page faults
    c. With more processes we can have more and more faults since working sets take up more of memory

Questions?