EECE 7352: Computer Architecture

Assignment 5

Jiayun Xin

NUID: 001563582

College of Engineering

Northeastern University Boston, Massachusetts

Spring, 2022

Q1

Why do the authors propose to use hashing for inverted page tables in this article "Virtual memory: issues of implementation"? What is the purpose of Hash Anchor Table?

Normally, operation system would look for a page frame number through page table depending on the given virtual page number. Using hashing for inverted page tables is to make page frame number available and avoid page number collision by using a collision-chain mechanism.

Hash Anchor Table is used to keep the table small and increase memory accesses per lookup as a trade-off. At the same time, it reduces the average collision-chain length.

Q2

Why do the authors of this paper "Effective Mimicry of Belady's MIN Policy, HPCA'22 (published in February'22)" state that their MockingJay policy is more effective than prior approaches? What is the difference between inferring at priority at the time of insertion vs. at the time eviction?

On both single-core and multi-core platforms and both with and without a prefetcher, the new MockingJay policy shows improvement with different extent comparing with LRU, SHiP and Hawkeye. In addition, it can provide accurate reuse distance predictions. When reuse distance prediction in unavailable, the age information will be considered.

Inferring at priority at the time of eviction takes subsequent lines as a factor to compute line's priority but insertion not.

Q3

What is the key idea behind entangled instruction prefetcher (first appeared in the Instruction Prefetching Championship)?

The entangled instruction prefetcher (EPI) is designed to meet three properties of timeliness, coverage and accuracy. Firstly, it searches for the instruction that triggers the prefetch for the consecutive instruction and computes the latency of each prefetch. As the process of tracking each pair of entangled cache lines, EPI merges those following basic blocks and entangles head of basic blocks to reduce entangled lines.

Q4

Describe the two TLB prefetchers proposed in the following paper: (1) Leader-Follower, and (2) Distance-based Cross-Core prefetchers. Clearly articulate the motivation and benefits of inter-core TLB prefetching.

As growing of chip multiprocessors (CMPs), it is necessary to examine TLB performance on parallel workloads. In fact, it is similar between uniprocessor TLB miss pattern and multi processors. To reduce cache access time and miss rates, two new TLB prefetchers were designed.

1) Leader-Follower pushes the common TLB misses from the “leader” to other cores to avoid missing on the same virtual page entry. But identifying miss patterns and prevent bad prefetch are the challenges that leader-follower should focus on.

2) Distance-based Cross-Core prefetchers is designed to detect and adapt to the stride patterns. It focuses on the successive missing virtual pages on one core and pass the same distance patterns to other cores to eliminate miss rates in total.

Q5

(a)
Page offset length = $\log_2(16k) = 14$
Virtual page length = virtual address length – page offset length = 40 – 14 = 26
Physical page length = physical address length – page offset length = 32 -14 = 18
TLB entry size = virtual page length + physical page length + other purposes = 26 + 18 + 4 = 48bits
Total size of TLB = #TLB entries * TLB entry size = 8 * 48 = 384bits

(b)
PTE size = PTE entry for other purposes + physical page length = 4 + 18 = 22bits
PTEs = 2^26
The total size of the page table = 22 * 2^26 bits

Q6

P = 512bytes
VPO = PPO = log2(512) = 9bits
VPN = virtual address space – VPO = 30 – 9 = 21bits
PPN = physical address space – PPO = 22 – 9 = 13bits

P = 1KB
VPO = PPO = log2(1KB) = 10bits
VPN = virtual address space – VPO = 30 – 10 = 20bits
PPN = physical address space – PPO = 22 – 10 = 12bits

P = 2KB
VPO = PPO = log2(1KB) = 11bits
VPN = virtual address space – VPO = 30 – 11 = 19bits
PPN = physical address space – PPO = 22 – 11 = 11bits

Q7

Virtual address = operation system length
Physical address = log2(RAM size)
Offset = $\log_2$(page size)
Virtual page number = virtual address – offset
Physical page number = physical address – offset

Config	Vir. Addr.	Phy. Addr.	VPN	PPN	Offset
a	32	30	20	18	12
b	32	31	18	17	14
c	64	34	50	20	14

Q8

a. How will you get the pagesize on your system?

```
bash-4.2$ getconf PAGESIZE
4096
```

Use “getconf” command.

b. What is “vmstat” command? What does it report?

```
bash-4.2$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 44465656   3132 3960324    0    0     0     0    3    4  0  0 100  0  0
```

The “vmstat” command is a built-in monitoring utility. It reports real-time information about memory, system processes, paging, block I/O and CPU scheduling.

c. What is the output of “cat /proc/meminfo”?

```
bash-4.2$ cat /proc/meminfo
MemTotal:       49279180 kB
MemFree:        44462912 kB
MemAvailable:   47849336 kB
Buffers:            3132 kB
Cached:          3599616 kB
SwapCached:            0 kB
Active:          2304324 kB
Inactive:        1628248 kB
Active(anon):     331552 kB
Inactive(anon):    24256 kB
Active(file):    1972772 kB
Inactive(file):  1603992 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:      20479996 kB
SwapFree:       20479996 kB
Dirty:                 8 kB
Writeback:             0 kB
AnonPages:        329568 kB
Mapped:           148180 kB
Shmem:             25984 kB
Slab:             360676 kB
SReclaimable:     269360 kB
SUnreclaim:        91316 kB
KernelStack:        9824 kB
PageTables:        19012 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    45119584 kB
Committed_AS:    2150596 kB
VmallocTotal:   34359738367 kB
VmallocUsed:       392676 kB
VmallocChunk:   34333390844 kB
HardwareCorrupted:     0 kB
AnonHugePages:     75776 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:      191972 kB
DirectMap2M:     3989504 kB
DirectMap1G:    46137344 kB
```

It outputs a list of memory usedness. Through “meminfo”, we can determine how much available memory the machine has.

d. What does the output of “sar –B” tell you?

```
bash-4.2$ sar -B
Linux 3.10.0-693.17.1.el7.x86_64 (Zeta.coe.neu.edu)     05/02/2022      _x86_64_        (24 CPU)

12:00:01 AM  pgpgin/s pgpgout/s   fault/s  majflt/s  pgfree/s pgscank/s pgscand/s pgsteal/s    %vmeff
12:10:01 AM      0.00      1.60    267.56      0.00    108.57      0.00      0.00      0.00      0.00
12:20:01 AM      0.00      1.29    235.16      0.00     93.66      0.00      0.00      0.00      0.00
12:30:01 AM      0.00      1.44    272.06      0.00    107.50      0.00      0.00      0.00      0.00
12:40:01 AM      0.55      2.17    255.17      0.00    105.56      0.00      0.00      0.00      0.00
12:50:01 AM      0.00      1.66    308.42      0.00    127.25      0.00      0.00      0.00      0.00
01:00:01 AM      0.00      1.08    235.56      0.00     95.26      0.00      0.00      0.00      0.00
01:10:01 AM      0.00      1.33    253.66      0.00    100.93      0.00      0.00      0.00      0.00
01:20:01 AM      0.00     68.42    280.62      0.00    144.99      0.00      0.00      0.00      0.00
01:30:01 AM      0.00      1.64    237.25      0.00     96.08      0.00      0.00      0.00      0.00
01:40:01 AM      0.01      2.04    629.03      0.11    365.04      0.00      0.00      0.00      0.00
01:50:01 AM      0.00      1.84    692.86      0.00    393.80      0.00      0.00      0.00      0.00
Average:         0.05      7.68    333.40      0.01    158.06      0.00      0.00      0.00      0.00
```

“sar -B” command outputs the paging statistics of coe linux server.