**Samonorom Chum**

**Tuesday, April 15, 2025**

**EECE 4811/5811 Operating System Spring 2025**

**Professor Tseng**

**HW6**

1. **Latency Numbers (3 pt)**

* Check the slide “Numbers Everyone Should Know” in Jeff Dean’s talk back in 2009

<https://www.cs.cornell.edu/projects/ladis2009/talks/dean-keynote-ladis2009.pdf>

* Do some research and see if the following numbers have been improved or not nowadays?
* L1 cache reference
* Branch mispredict
* L2 cache reference
* Mutex lock/unlock
* Main memory reference
* Read 1 MB sequentially from memory
* Round trip within same datacenter
* Disk seek
* Read 1 MB sequentially from disk
* Send packet CA->Netherlands->CA
* To get full credits, list the latency numbers you found and include your source and/or test by your own.
* You may ask any ChatBot, but you need to include your prompt. You should also know that, for this type of questions, ChatBot is pretty easy to make mistakes.
* **Key concepts**: latency, cache, memory, disk

**Answer:**

|  |  |  |
| --- | --- | --- |
| **Operation** | **2009 Value (ns)** | **2020 Value (ns)** |
| L1 cache reference | 0.5 | 1 |
| Branch mispredict | 5 | 3 |
| L2 cache reference | 7 | 4 |
| Mutex lock/unlock | 25 | 17 |
| Main memory reference | 100 | 100 |
| Read 1 MB sequentially from memory | 250,000 | 3000 |
| Round trip within same datacenter | 500,000 | 500,000 |
| Disk seek (HDD) | 10,000,000 | 2,000,000 |
| Read 1 MB sequentially from disk (HDD) | 20,000,000 | 825,000 |
| Send packet CA ↔ Netherlands ↔ CA | 150,000,000 | 150,000,000 |
| https://colin-scott.github.io/personal\_website/research/interactive\_latency.html | | |

**ChatGPT Prompt 1:**   
*"Compare Jeff Dean’s 2009 latency slide (“Numbers Everyone Should Know”) with current 2024 latency values for CPU cache, memory, disk, datacenter latency, and internet round trips. Include credible sources."*

**ChatGPT Prompt 2:**

“for the 2024 table, add the link to each source”

Afterwards I fell down a rabbit hole of trying to find reliable answer, but ultimately Colin Scott’s visual representation seems to be reliable enough to be featured on ARM’s official website (<https://learn.arm.com/learning-paths/cross-platform/memory-latency/what-is-latency/> ).

I’m still confused on why the L1 cache is considered 1 ns instead of 0.5 as originally stated in Jeff Dean’s PDF file. However, Branch Mispredict, L2 cache reference, Mutex lock/unlock have significantly improved. I believe more researchers/developers are focusing on improving them since L1 cache reference is extremely fast already. Reading 1MB from memory and disk sequentially has also drastically improved. While round trips, HDD disk seek, and sending packets remain relatively the same in 2020.

2. **Process and Page (3 pt)**

* When you fork a child process from a parent, you need to replicate the address space of the parent process. Hence, this operation may take a while to complete, if the address space is large. One technique to mitigate the latency is called “copy-on-write” or COW.
* Q1: do some research to explain what COW is and why it improves the latency for forking.
* Q2: present a pseudo-code to implement COW for forking a process. Your pseudo-code should include the part to free physical pages as well.
* Note 1: Your pseudo-code should be self-explanatory. If not, use comments wisely.
* Note 2: You do not need to present a real implementation in any kernel, as it takes a while to complete.
* Note 3: There are a lot of solutions out there, but not all of them are correct.
* **Key concepts**: page, COW, fork

**Q1 Answer:**

“Copy-on-Write (COW) is a memory optimization strategy where a parent and child process initially share the same physical memory pages after a fork(). Pages are only duplicated when either process modifies them. This avoids the costly operation of copying all pages at fork time.”

From what I understand, it’s when the OS creates a fork, it is currently sharing the same pages in memory, and only if the data is going to be modified will it create a copy to reduce latency cost.

**ChatGPT Prompt 1:**   
*"Write pseudocode for implementing process forking using Copy-on-Write (COW) with memory page tracking and refcounting, and handle page faults."*

**Q2 Answer: Pseudo-Code**

// Data structures

PageTable parent\_page\_table, child\_page\_table;

PhysicalPage \*physical\_pages[MAX\_PAGES]; // array of physical page pointers

RefCount ref\_count[MAX\_PAGES];

// Function to fork a process using Copy-on-Write

Process\* fork\_process(Process \*parent) {

Process \*child = create\_process();

for (int i = 0; i < MAX\_PAGES; i++) {

if (parent->page\_table[i].valid) {

// Point to same physical page

child->page\_table[i].frame = parent->page\_table[i].frame;

child->page\_table[i].valid = true;

child->page\_table[i].writable = false;

parent->page\_table[i].writable = false;

// Increment reference count

ref\_count[parent->page\_table[i].frame]++;

}

}

return child;

}

// On page fault (write to read-only shared page)

void handle\_page\_fault(Process \*proc, int page\_num) {

int frame = proc->page\_table[page\_num].frame;

if (ref\_count[frame] > 1) {

// Allocate new page

int new\_frame = alloc\_physical\_page();

copy\_page\_contents(new\_frame, frame);

ref\_count[frame]--;

ref\_count[new\_frame] = 1;

proc->page\_table[page\_num].frame = new\_frame;

proc->page\_table[page\_num].writable = true;

} else {

// Only this process owns the page now

proc->page\_table[page\_num].writable = true;

}

}

// Page deallocation (optional in this HW)

void free\_physical\_pages(Process \*proc) {

for (int i = 0; i < MAX\_PAGES; i++) {

if (proc->page\_table[i].valid) {

int frame = proc->page\_table[i].frame;

ref\_count[frame]--;

if (ref\_count[frame] == 0) {

dealloc\_physical\_page(frame);

}

}

}

}

**ChatGPT Prompt 1:**   
*"Write pseudocode for implementing process forking using Copy-on-Write (COW) with memory page tracking and refcounting, and handle page faults."*

**References:**

1. ChatGPT, “Assistance with HW6 priority scheduling and paging concepts,” OpenAI, personal communication, May. 6, 2025.
2. R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, Operating Systems: Three Easy Pieces, Version 1.10, Chapter 18, 2023. [Online]. Available: <https://pages.cs.wisc.edu/~remzi/OSTEP/>
3. **UC Berkeley – Latency Reference**  
   C. Scott, “Latency Numbers Every Programmer Should Know,” Colin Scott’s Personal Website, [Online]. Available: <https://colin-scott.github.io/personal_website/research/interactive_latency.html> [Accessed: May 6, 2025].
4. **ARM – Memory Latency Explanation**  
   ARM Ltd., “What is Latency?,” ARM Learn, [Online]. Available: <https://learn.arm.com/learning-paths/cross-platform/memory-latency/what-is-latency/> [Accessed: May 6, 2025].
5. **GeeksforGeeks – Copy-on-Write**  
   GeeksforGeeks, “Copy-On-Write (COW) in Operating Systems,” GeeksforGeeks, [Online]. Available: <https://www.geeksforgeeks.org/copy-on-write/> [Accessed: May 6, 2025].