CMPSCI 590S Systems for Data Science **Fall 2017**

Lecture September 7

Lecturer: Emery Berger *Scribe: Aishwarya Kamath, Ananya Suraj*

2.1 General Memory Model

Memory can be thought of as a big array. For example, in a 64 bit architecture, the size of a word is 8 bytes. The memory array would have to be of size $2^61 \approx 10^19 \approx 2$ billion gigabytes! Hence, this general memory model which looks like the one in Figure 2.1, is unrealistic.

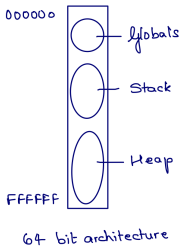

Figure 2.1: General memory model

Limitations of using Physical Memory:

- Portability: If physical memory addresses are used, transferring objects from one machine to another becomes impossible as the physical address that is used to refer to the object will change on the new machine.
- Isolation: If two or more users are running programs at the same time and use physical addresses, a memory error in one program (for example, reading a bad pointer) could destroy memory being used by the other process, taking down multiple programs due to a single crash.
- Limited Memory: Since physical memory addresses would just be from say, 00000000 to FFFFFFFF, this puts a big limitation on the user as one generally needs a lot more memory in practice. Virtual Memory which is a solution to these limitations is discussed in the next section.

2.1.1 Virtual memory

We use an abstraction called "Virtual Memory" which was invented in the 1960s. The operating system still sees the physical memory but the user sees the virtual memory layer. The idea behind virtual memory is the following : "All problems are solved by adding one level of indirection".

Consider a user who looks for an item at address 1,000,000. This is the item's address in virtual memory which is a "view" of the data actually stored in physical memory. The mapping to the physical memory is called "translation" (Fig 2.2. The key idea here is to never give two objects the same memory. So, if another user tries to access the same variable, suppose, it gets mapped to a different chunk of physical memory.

Locations in physical memory are completely separate though the virtual address may appear to be the same.

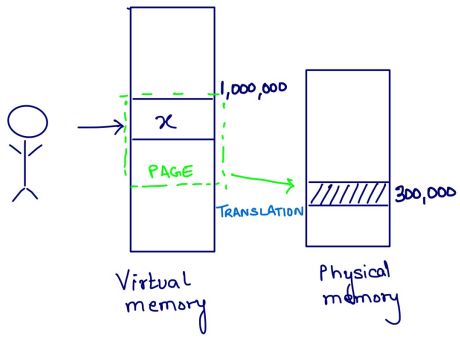

Figure 2.2: Translation

The issue that arises now is, what happens when RAM gets filled?

Swapping is the technique used to combat this problem. Pages that are rarely used are swapped out of memory to bring in new pages from the disk. A separate part of disk called the "swap partition" is assigned the task of holding swapped out data. By default, 4-8 times the size of RAM is saved on the disk for swap on Macs. Linux systems use a separate logical disk as a swap partition Fig 2.3.

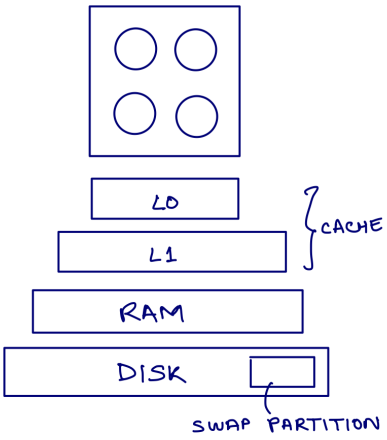

Figure 2.3: Swap partition

A method of virtual memory management called demand paging is used, in which the pages of data are not copied from disk to RAM until they are needed. This is in contrast to anticipatory paging, also called page pre-fetch, in which pages that will be referenced soon as predicted and fetched, so as to reduce the future page faults.

2.1.2 Strategies for Eviction

Eviction is the term used to describe taking pages out from cache and putting it on disk.

- **Least Recently Used (LRU)** : This cache replacement policy replaces the least recently used page from the cache with the latest page that was accessed. In order to implement this kind of caching mechanism, the state of every page will have to be maintained. One way to do this would be to have a counter for the time stamp at which a page was last accessed and increment it every time the page is accessed again. At the time of eviction, the page with the earliest time stamp is thrown out. An example of LRU strategy is the Clock Page Replacement algorithm.
- **Random** : This strategy does not require to keep track of the state. Surprisingly, both these strategies do equally badly. For the random strategy, the odds of getting unlucky - that we would require to fetch data that is on disk and not in memory - is very small. In the case of LRU, it can go from 100% hit to 0% hit in successive fetches as seen in Figure 2.4.

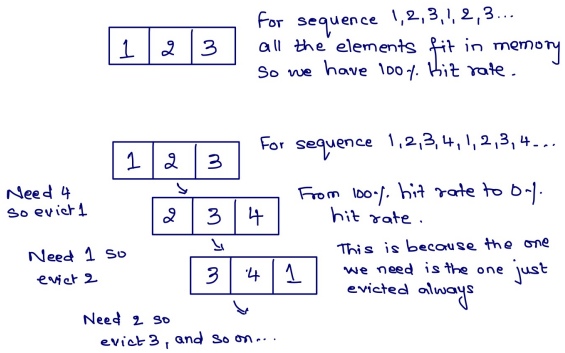

Figure 2.4: Least Recently Used Eviction

2.2 Use of Cache in Web Browsers

In a web browser like Chrome, information is cached extensively so that websites can be accessed faster and data can be loaded quickly (Fig 2.5). When a website gets a lot of hits, cache service companies like Akamai, Cloudflare etc help the websites by providing them with the required memory.

2.3 Translation Lookaside Buffer

A translation lookaside buffer(TLB), also known as an address translation cache, is a memory cache that reduces the time required to access a memory location. The Memory Management Unit (MMU) of a computer has the TLB which contains a few virtual to physical address mappings. The size of the TLB, in terms of number of entries is usually small from about 256 to 512 entries. The entries stored in the TLB are the virtual memory to physical memory translations of recently accessed memory locations (Fig 2.6).

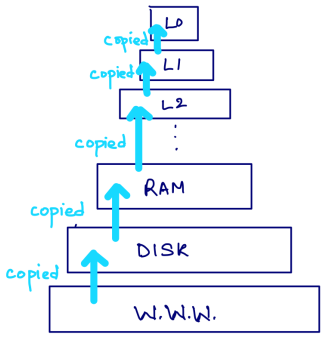

Figure 2.5: Web browser caching

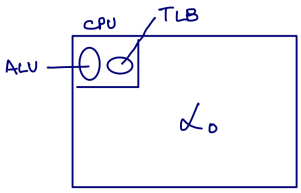

Figure 2.6: Translation Lookaside Buffer

The data structure used to store a TLB is not a hash table because hash functions do not maintain locality of the data. Trees are used instead so that the RAM stores sequential data in contiguous locations.

High TLB Pressure: Suppose the working set T has 512 entries in the TLB and the page size is around 4K, then we can only access 512 x 4K $\approx$ 2MB, which seems very insufficient as today even a small image has a size of about 3MB. Hence, having a very small page size puts a large amount of pressure on the TLB.

Solution: Dynamic Page Sizing: Some architectures support multiple page sizes with pages that can be significantly larger than the standard size (4K, 2MB or even 1GB). Operating systems like Linux support dynamic page sizes called Huge Pages, while Windows offers Large Pages and FreeBSD supports SuperPages. This reduces the pressure on the TLB. For example, if you often use 1GB of space, then it promotes the page size dynamically to 1GB.