【OS】Day27

Class	Operating System: Three Easy Pieces
Date	@January 26, 2022

【Ch22】Beyong Physical Memory: Policies

When little memory is free when a page fault occurs, the memory pressure forces the OS to start paging out pages to make room for actively-used pages.

Deciding which page to evict is encapsulated within the replacement policy of the OS.

And thus our crux: *How can the OS decide which page to evict from memory?* This decision is made by the replacement policy of the system, which usually follows some general principles but also includes certain tweaks to avoid corner-case behaviours.

22.1 Cache Management

Given that main memory holds some subset of all the pages in the system, it can rightly be viewed as a cache for virtual memory pages in the system.

Thus, our goal in picking a replacement policy for this cache is to minimize the number of cache misses, i.e., to minimize the number of times that we have to fetch a page from disk.

Alternately, one can view our goal as maximizing the number of cache hits.

Knowing the number of cache hits and misses let us calculate the average memory access time(AMAT) for a program. Specifically, given these values, we can compute the AMAT of a program as follows:

$$AMAT = T_M + (P_{Miss} \cdot T_D)$$

where T_M represents the cost of accessing memory, T_D the cost of accessing disk, and P_{Miss} the probability of not finding the data in the cache(a miss).

Note that we always pay the cost of accessing the data in memory; when we miss, however, we must additionally pay the cost of fetching the data from disk.

For example, let us imagine a machine with a (tiny) address space: 4KB, with 256-byte pages. Thus, a virtual address has two components: a 4-bit VPN and an 8-bit offset.

Thus, a process in this example can access 2^4 or 16 total virtual pages.

In this example, the process generates the following memory references: 0x000, 0x100, 0x200, 0x300, 0x400, 0x500, 0x600, 0x700, 0x800, 0x900. These virtual addresses refer to the first byte of each of the first ten pages of the address space.

Let us further assume that every page except page 3 is already is memory. Thus, our sequence of memory references will encounter the following behaviour: hit, hit, hit, miss, hit, hit, hit, hit, hit, hit.

We can compute the hit rate: 90%. The miss rate is thus 10%(P_{miss}=0.1).

To calculate AMAT, we need to know the cost of accessing memory and the cost of accessing disk. Assuming the cost of accessing memory(T_M) is around 100 nanoseconds, and the cost of accessing disk(T_D) is about 10 milliseconds, we have the following AMAT: $100ns + 0.1 \cdot 10ms$, which is 1.0001ms.

Unfortunately, as we can see in this example, the cost of disk access is so high in modern system that even a tiny miss rate will quickly dominate the overall AMAT or running programs. Clearly, we need to avoid as many misses as possible or run slowly, at the rate of the disk.

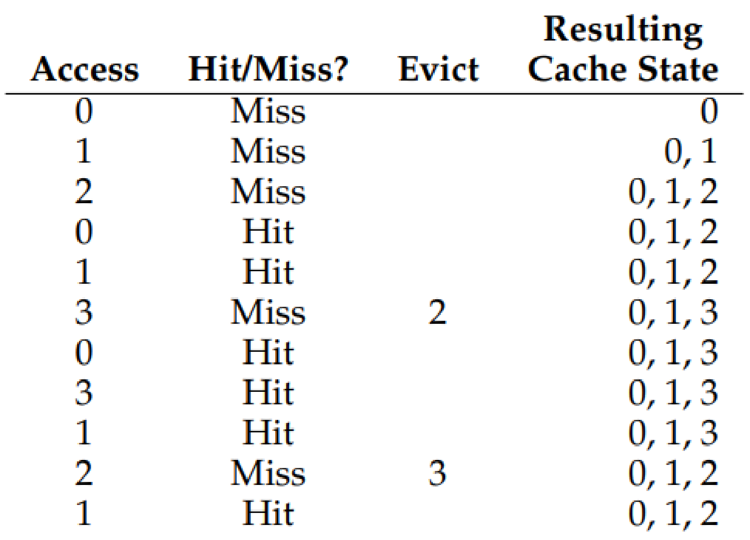
22.2 The Optimal Replacement Policy

The optimal replacement policy leads to the fewest number of misses overall.

Think about it like this: if we have to throw out some page, *why not throw out the one that is needed the furthest from now?*

The reason this is true is simple: we will refer to the other pages before we refer to the one furthest out.

Let’s trace through a simple example. Assume a program accesses the following stream of virtual pages: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1. The following figure shows the behaviour of optimal, assuming a cache that fits three pages.



Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0, 1
2	Miss		0, 1, 2
0	Hit		0, 1, 2
1	Hit		0, 1, 2
3	Miss	2	0, 1, 3
0	Hit		0, 1, 3
3	Hit		0, 1, 3
1	Hit		0, 1, 3
2	Miss	3	0, 1, 2
1	Hit		0, 1, 2

Figure 22.1: **Tracing The Optimal Policy**

Not surprisingly, the first three accesses are misses, as the cache begins in an empty state; such a miss is sometimes referred to as a cold-start miss(or compulsory miss).

Then we refer again to pages 0 and 1, which both hit in the cache. Finally, we reach another miss(to page 3), but this time the cache is full; a replacement must take place.

With the optimal policy, we examine the future for each page currently in the cache(0, 1, and 2) and see that 0 is accessed almost immediately, 1 is accessed a little later, and 2 is accessed furthest in the future.

Thus the optimal policy has an easy choice: evict page2, resulting in page 0, 1, and 3 in the cache.

We can also calculate the hit rate for the cache: with 6 hits and 5 misses, the hit rate is $\frac{Hits}{Hits+Misses}$ which is 54.5%.

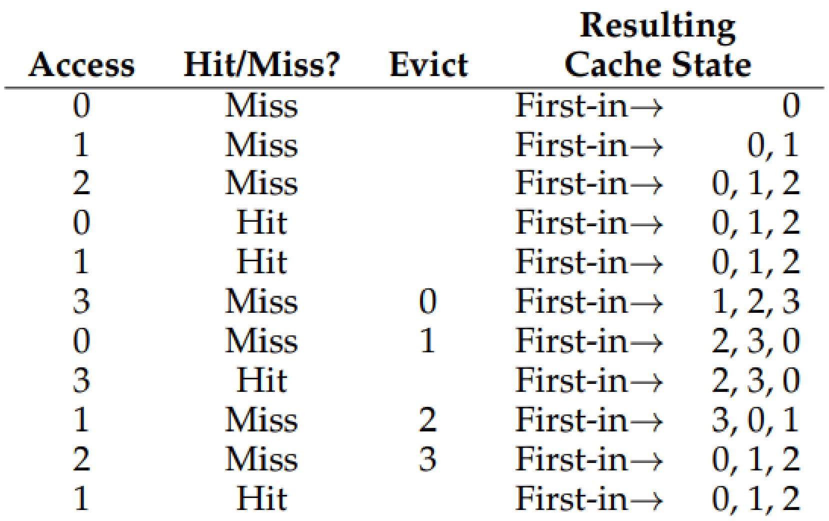
We can also compute the hit rate modulo compulsory misses(i.e. ignore the first miss to a given page), resulting in a 85.7% hit rate.

22.3 A Simple Policy: FIFO

Some system used FIFO(first-in, first-out) replacement, where pages were simply placed in a queue when they enter the system; when a replacement occurs, the page on the tail of the queue is evicted.

FIFO has one great strength; it is quite simple to implement.

Let's examine how FIFO does on our example references stream:



Access	Hit/Miss?	Evict	Resulting Cache State	
0	Miss		First-in→	0
1	Miss		First-in→	0, 1
2	Miss		First-in→	0, 1, 2
0	Hit		First-in→	0, 1, 2
1	Hit		First-in→	0, 1, 2
3	Miss	0	First-in→	1, 2, 3
0	Miss	1	First-in→	2, 3, 0
3	Hit		First-in→	2, 3, 0
1	Miss	2	First-in→	3, 0, 1
2	Miss	3	First-in→	0, 1, 2
1	Hit		First-in→	0, 1, 2

Figure 22.2: **Tracing The FIFO Policy**

Comparing FIFO to optimal, FIFO does notably worse: a 36.4% hit rate(or 57,1% excluding compulsory misses).

FIFO simply can't determine the important of blocks: even though page 0 had been accessed a number of times, FIFO still kicks it out, simply because it was the first one brought into memory.