Cache—Efficiency Thinking

Paul S. Wang, Sofpower.com

July 13, 2021

The dictionary defines the word cache as "*a store or collection of items kept in a safe place*". However in computing, **cache** is a technical term referring to *a memory for fast storage and retrieval of data*. In this blog post we are going to talk about the latter.

An understanding of the cache concept can focus our attention on ready access and efficiency. Knowing how cache memory works inspires us to employ the same ideas to better manage businesses, improve our daily living, and even save lives.

Cache in the Modern CPU

A CPU (Central Processing Unit) generally repeats endlessly this cycle: (1) retrieving from main memory the next instruction then (2) executing the instruction. The problem is that a modern CPU performs step 2 much faster than step 1. In other words, main memory (RAM) latency (access delay) becomes the performance bottle neck. The solution? *CPU cache*.

CPU cache is super fast SRAM (static RAM), usually 16 MB (Mega bit) to 64 MB, located on the CPU chip itself, as opposed to on the motherboard. It operates between 10 to 100 times faster than RAM, requiring only a few nanoseconds (1 ns = 10^{-9} second) to respond to a data request. The CPU cache serves as a buffer on the CPU chip for data in the main memory. Sophisticated algorithms have been devised, implemented in the CPU *cache control unit*, to keep required data in the cache as much as possible so that the need for RAM access is minimized. Thus, CPU cache significantly increases overall CPU speed and efficiency.

This sounds complicated but the idea is simple. For example, in our daily lives we carry a wallet or purse (this is the cache). We keep often needed

items in it for instant retrieval, so much faster than looking for them in the house (the main memory). Depending on our current tasks, we can fill our wallet/purse with the right items. The tool belt for workers is another example. These daily cache devices are so efficient we don't mind managing them. In fact we won't do without them.

Hit Or Miss

As the CPU executes, it continuously needs to access the next instruction/data by referring to its RAM address (**A**), usually a 32 or 64 bit quantity. Given any **A**, it can be quickly determined whether it is available in the CPU cache. If it is, we have a cache *hit*, otherwise we have a cache *miss*.

Here is a simplified description of what happens when a read access to **A** takes place.

Hit: Immediately, the requested data is returned from the cache and the CPU continues execution without delay.

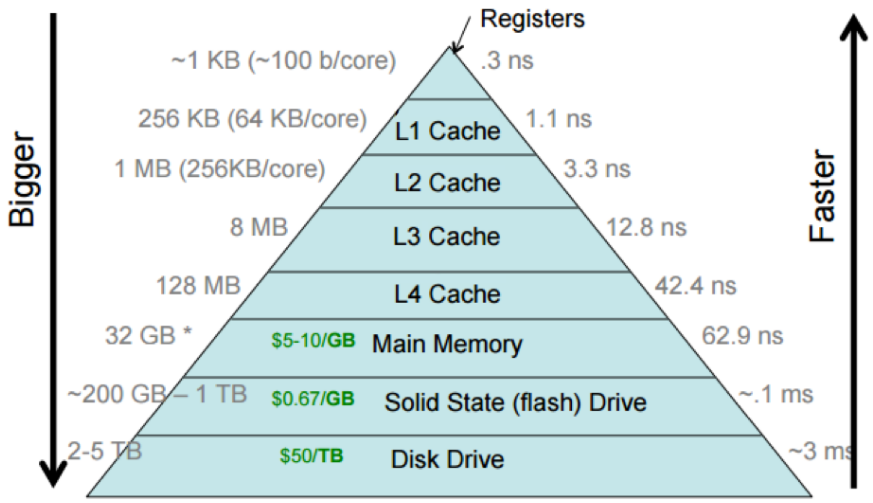
Miss: The requested data, together with a number of close-by items, are copied from RAM to the cache. Space must also be found in the cache to store newly retrieved data. This usually involves the *eviction* of some well-selected in-cache data. Then the data is returned for read access. Meantime, CPU execution has been blocked until the data is returned.

A write access miss may cause less delay. But let's not get into details and simply say that a cache miss is expensive and can delay the CPU's progress significantly. Therefore, arrangements and algorithms are designed to maximize hits. Typical cache hit rates range from 95% to 98%.

Levels of Cache

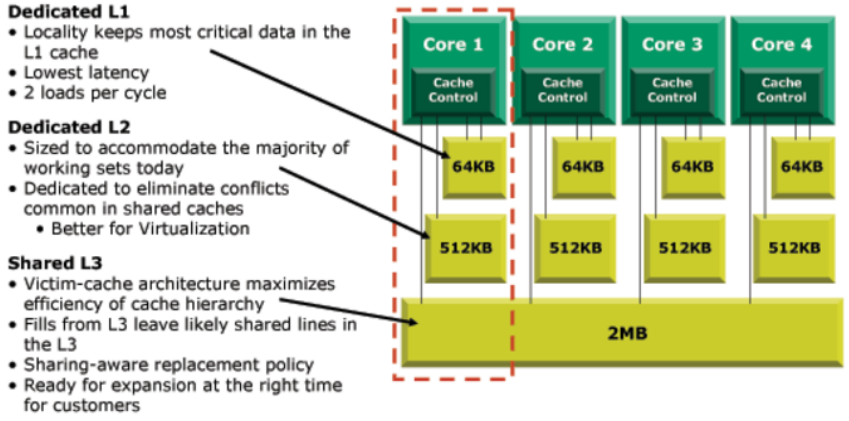
Generally, the cache idea is using ultra-fast but expensive memory to hold select items from much slower but cheaper RAM memory in order to achieve speed without undue increase in cost. Clever algorithms are used to maximize the hit rate so that fast memory access happens almost all the time.

The following diagram shows the memory hierarchy of modern computers.



We see from the diagram that the RAM is about 57 times slower than L1 cache (62.9/1.1). To better illustrate the relative speeds think about it this way. If L1 access were to take one second then RAM access would have taken nearly 1 minute. Better yet, if L1 were to take one day then RAM would have taken almost two months, eek!

As you can see CPU cache has levels L1 through L3 or even L4. Each level decreases in speed and cost, but increases in size. Sophisticated logic circuits are used to manage these levels to use them in optimal ways. For multi-core CPUs, each core has its own L1 and L2 while L3 and L4 are shared among all cores. A cache miss happens when the needed data can't be found through all cache levels.

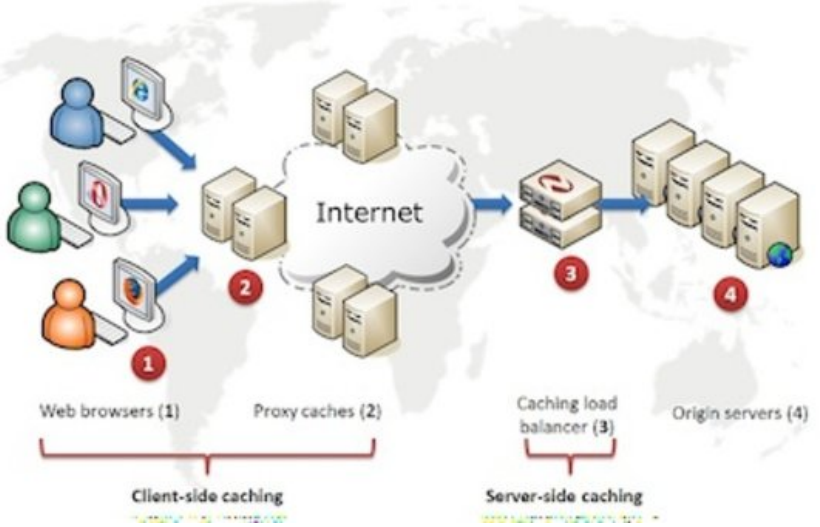


Similar buffering happens between RAM and secondary memories such as flash drives and hard disks. What is the lesson here? "*If an idea works why not use it everywhere it can be applied?*"

For example, the Web protocol HTTP has added its own way of caching to make the Web faster and more efficient.

HTTP Caching

An important improvement of HTTP 1.1 over HTTP 1.0 is the introduction of caching. On the Web, a great deal of contents are not changing often with time. These include static webpages, images, graphics, styling code, scripts, and so on. Saving a copy of such data can avoid a lot of unnecessary work of requesting and retrieving the same data over and over again from *origin servers*. Browsers (user agents) and caching proxy servers are able to serve data from their cache when they know or can verify that the data are still current and unchanged on the servers where they originated.



A caching proxy server accelerates requests by providing contents from its cache. Caching proxies keep local copies of popular resources so large organizations can greatly reduces their Internet usage and costs, while significantly enhance performance. Most ISPs and large businesses employ caching proxies.

The HTTP caching scheme significantly cuts down round-trip Web traffic to origin servers and reduce response time to users. This explains why it is slower the first time you visit a website. Then it is lightning fast when you visit again.

Cache Inspired CT

Thus, we have this computational thinking principle **Cache for Speed**: "*Use cache to increase efficiency and speed. In many situations, significant improvement may result from storing the right items in a cache.*" In Essence, the cache idea from computing is fundamentally *prepositioning needed items in convenient locations for immediate access* to reduce delay and maximize efficiency.

Applications of this idea can be found in many situations. For driving emergencies—bringing spare tire, jump cables, flash lights, work gloves; for fire fighting—prepositioning fire stations, fire hydrants, and extinguishers; for life saving—placing police outposts, ambulance stations, emergency lighting, exit maps, first aid kits, defibrillator, and gas masks.

Let's see some more day-to-day applications. Restaurants use caching to better serve customers. Ready to cook ingredients for popular dishes are arranged, collected and prepositioned near the cook station (the restaurant CPU). This cuts down on customer's waiting time at the tables significantly. Less popular dishes will take longer when materials must be gathered and prepared before cooking.

A busy airport has a 'taxi cache' located close to arriving passenger exits, so clever and efficient.

Cache And Obama Care

Toward the end of 2013, when a team of super coders were helping to rescue and fix the `healthcare.gov` site, one immediate technique they used was introducing a *database cache* so that frequent queries could be separated from other queries to the huge database. The database cache reduced congestion, and they were able to lower the average page access time from 8 seconds to about 2 seconds. Later, with continued improvements, the access time was reduced to below 0.35 seconds. The rescue work may well have saved the Affordable Care Act from disaster.

Don't we all wish, during the early days of the COVID-19 pandemic, that PPE (personal protective equipment) were readily available to front-line workers? That quick test kits (or any test kit) were available for immediate use? Could such preparedness have saved many lives? The question is not if pandemics will happen again but when. We need to apply the lessons learned to get ready for the next one, cache inspired preparedness included.

Cache And Me

The cache concept is useful in a variety of situations including our own lives.

In terms of personal information processing, our brains are like the CPU. Data in the 'brain cache' are information we remember.That data are available immediately without going outside the brain. The next place to keep often-used information may be our smartphones. Data kept there are also easily available, like in the RAM of a computer, where we can go next when there is a 'brain cache miss'. Of course, there are other information sources. But they are much slower to access.

You have your home phone number and address, for example, kept in brain cache, right? But what about other important data? If you have a well-developed plan for where to keep information, could you be better off?

In our home, we also have a hierarchy of storage spaces. Drawers, cabinets, closets, garage, and basement. Of course we need to organize our things into these spaces to optimize ease of access. And most importantly, we need to place items back where they belong if we are going to find them easily next time. How similar is it to managing the computer's memory hierarchy.

Next, let's talk about the car. We all love our cars and we use them to go places. The car is also a place to store things. Why not treat our car like a 'cache to go', placing things we need to bring in the car way ahead of time. Such things could be items to donate, online orders to return, gifts to send, letters to mail, tools, supplies, for example. This way we won't forget to bring needed things when it comes time to drive off.

Oh yes, when driving, a good driver will always keep a bird's eye view of the car relative to other cars. This cached view in the driver's brain can enable evasive moves without delay. It can save lives.

When it comes to money management, we will keep cash and credit cards in our wallet (our money cache), for larger amounts we depend on our checking accounts (our money RAM), then there is our savings account and other

even slower sources of funds. None of us will have any objection to more efficient money management.

Finally, the cache concept is about efficiency. When we are more efficient, we save time and energy, that literally can bring cash if not lives saved! Doesn't reading 'cache brings cash' out loud crash your CPU? See my *Computational Thinking* blog (`computize.org/ctblog`) for other interesting topics.