Caching and Locality

CSE 220: Systems Programming

Ethan Blanton & Carl Alphonce

Department of Computer Science and Engineering

University at Buffalo

Memory Structure

Memory technologies form a hierarchy of storage layers.

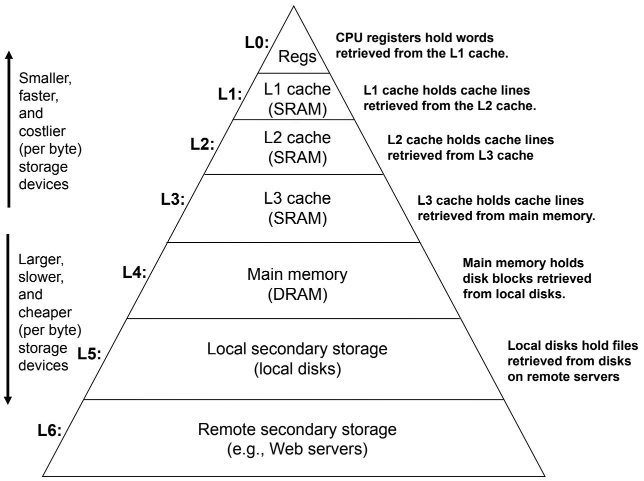
We ordinarily number these layers as L1, L2, …

The lowest-numbered layers are fastest and closest to the CPU.

Memory accesses interact with different layers at different times.

This complexity of structure and notation is about performance.

The Memory Hierarchy


Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
L0: Regs
L1: L1 cache (SRAM)
L2: L2 cache (SRAM)
L3: L3 cache (SRAM)
L4: Main memory (DRAM)
L5: Local secondary storage (local disks)
L6: Remote secondary storage (e.g., Web servers)
CPU registers hold words retrieved from the L1 cache.
L1 cache holds cache lines retrieved from the L2 cache.
L2 cache holds cache lines retrieved from L3 cache
L3 cache holds cache lines retrieved from main memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote servers

Memory Latency

The CPU is up to a thousand times faster than the main memory!

Memory speed is complicated, but:

- A modern processor has a clock cycle of about 0.3 ns.
- Many simple operations can complete in one clock cycle.¶
- Modern RAM can fetch a random location in about 10 ns.

This means a best case memory access is 30+ times longer than a CPU operation.

In reality it's typically several times longer than that.

This speed difference is growing larger.

Caching

Caching is temporarily storing data for faster access.

Typically this means storing it:

- closer to the CPU (electronically speaking)
- in a faster storage technology

A small amount of faster storage can make a big difference!

However, we must use it well.

Why Cache?

The bottom line:

- Fast storage is expensive.
- Large storage is slow.

Caching lets us pretend that our large, slow storage is fast.

Storage Technologies

A typical computer has several types of storage.

Volatile storage is lost when the power is turned off:

- CPU registers
- Static RAM (SRAM)
- Dynamic RAM (DRAM)

Nonvolatile storage retains its data indefinitely:¶

- Flash memory
- Magnetic disks
- Optical disks

Each technology has potentially very different properties.

CPU Registers

CPU Registers are single words of RAM.

They are attached directly to the CPU logic.¶

This means they can be accessed within a single CPU cycle.

They are the fastest storage we will discuss.

CPU registers are typically named or numbered.

Static RAM

Static RAM is the next fastest type of memory we will discuss.

SRAM is constructed from transistor latches.

It is quite fast, but also quite expensive per bit.

The closest caches to the CPU, L1 and L2, are typically SRAM.

Dynamic RAM

Dynamic RAM is the slowest and least expensive RAM.

It is made from transistors and capacitors.¶

This is much cheaper per bit, but requires refresh.

During refresh the RAM cannot be accessed.

Setting up the circuits that read/write the RAM takes time.

Reading a bit is also destructive, requiring it to be re-written.

DRAM Refresh

Capacitors store charge.

DRAM stores charge in a capacitor to encode a 1 bit.

Capacitors also leak their charge.

Therefore, the one bits must be frequently recharged.

A DRAM controller manages this.

Flash Memory

Flash Memory, like SRAM, stores bits in transistors.

Flash Memory used for solid state disks is typically NAND.¶

NAND Flash can be read or written in several kB blocks.

However, it is erased in several MB blocks.

This means that writes are periodically quite slow.

Flash access times are approaching DRAM speeds.

Magnetic Media

The modern magnetic media is hard disks.

Hard disks use spinning disks with magnetic coatings.

Reading/writing data requires:

- Moving a physical head on an arm over the data
- Waiting for the data to spin under the head

Hard disks are by far the slowest of the media we are discussing.

Maintaining Speed

How can a CPU maintain speed with such slow RAM?
The key is locality.

There are two important types of locality:

Temporal locality:
Recently-used locations are likely to be used again soon.

Spatial locality:
Newly-used locations are likely to be nearby recently-used locations.

These properties mean that small amounts of fast storage can make a big difference.

Locality Example

```
int sum = 0;
for (int i = 0; i < N; i++) {
    sum += array[i];
}
```

Temporal locality:

- `sum` is accessed frequently
- `i` is accessed frequently

Spatial locality:

- Sequential locations in `array` are accessed
- The instructions for the code are sequential

Caching and Locality

This locality principle is what makes caching effective.

Recently used and nearby data can be stored in fast storage.

Other data can remain in slower storage.

This allows the fast, expensive storage to be small …
Yet many accesses are to that small storage!

Cache Levels

Memories are often referred to by levels L0-L4:

Level	*Type*	*Size*	*Access Time*
L0	CPU registers	O(100 B)	~0 clock cycles
L1	Level 1 cache	O(10 KB)	~1-5 clock cycles
L2	Level 2 cache	O(100 KB)	~10+ clock cycles
L3	Level 3 cache	O(1 MB)	~30+ clock cycles
L4	Main memory	O(10 GB)	~100+ clock cycles

Cache Hits and Misses

Cache is organized in blocks of fixed size.

When reading data, a cache can hit or miss.

A cache hit is when a block of data is in the cache.

In this case, the read is fast!

A cache miss is when a block of data isn’t cached.

In this case, the next slower cache must be checked.

Cache Misses

There are three types of cache miss:

- Cold miss: The cache is empty or has free space but the block is not in the cache.
- Capacity miss: The block was removed from the cache to make room for another block.
- Conflict miss: The block must be stored in the same location as another block in the cache.

Caching the Caches

L1 stores blocks from L2, L2 stores blocks from L3, *etc.*

Software can also cache:

- Operating systems store disk blocks in RAM.
- Web browsers store network files on disk.

Some caches are transparent: you don't know they're there.

Some caches are explicitly managed.

Designing for Caching

Remember this from week two? This is caching!

```
void copyij(int src[2048][2048],
            int dst[2048][2048]) {
    for (int i = 0; i < 2048; i++) {
        for (int j = 0; j < 2048; j++)
           {
            dst[i][j] = src[i][j];
        }
    }
}
```

3.8 ms

```
void copyji(int src[2048][2048],
            int dst[2048][2048]) {
    for (int j = 0; j < 2048; j++) {
        for (int i = 0; i < 2048; i++)
           {
            dst[i][j] = src[i][j];
        }
    }
}
```

72.2 ms

In `copyji()`, spatial locality is much poorer!

Writing Cache-Friendly Code

Your code will be faster if:

- You keep your working set (the items used for a particular task) small.
- You pay attention to locality.

This doesn't mean you can't use large data!
(But it does mean you should try to use it sequentially.)

This doesn't mean you can't use random access!
(But it does mean you should try to do it over small data.)

Compartmentalize the code that isn't cache-friendly.

Honing Locality Sense

You will want to improve your sense of locality.

Does this code have good locality? Could it be better?

Algorithms matter most, but constant factors make a difference.

Always write code with the priorities:

1. Correctness
2. Understandability
3. Performance

Summary

- The CPU is much faster than memory or disks.
- The difference in speeds is growing.
- Programs exhibit locality:
  - Spatial
  - Temporal
- Caching depends on locality to improve performance.
- Writing good programs requires understanding locality.

References I

Required Readings

[1] Randal E. Bryant and David R. O'Hallaron. *Computer Science: A Programmer's Perspective*. Third Edition. Chapter 6: Intro, 6.1-6.3, 6.5-6.7. Pearson, 2016.

License