Exam Assignments 9

Bandwidth-Bound and Compute-Bound Computations

Some computations are more intensive in their usage of the CPU than their usage of memory; these are called compute-bound, as their performance is bottlenecked by the CPU.

Other computations utilize memory more intensively than the CPU; these are called bandwidth-bound or memory-bound, as their performance is bottlenecked by how quickly data can be transferred to and from memory.

When optimizing, one must focus the respective bottleneck (memory or cpu) to achieve the greatest performance gains.

Temporal and Spatial Locality

Temporal and spatial locality are aspects under which one can examine a programs memory accesses.

Memory accesses are temporally local if the same address is accessed often in a short time window. This allows caching to be very effective, as the content doesn't have to be retrieved from main memory over and over again.

Memory accesses are spatially local if addresses that are close to each other are accessed shortly after one another. This can improve performance because addresses that are on the same cache line are loaded into the cache together. If multiple are accessed before the cache line gets removed, then we can save memory accesses on higher levels in the hierarchy.

Data-Oriented and Object-Oriented Design

Object-oriented design (or programming, thus OOP) focuses on making the life of the developer easier by using easily understandable data abstractions that mirror the way humans categorize the world.
Data-oriented design (DOD) instead focuses on making the life of the computer easier by using data abstractions that allow operations to run as efficiently as possible.

One example of this is the difference between structure-of-arrays and array-of-structures. The array-of-structures approach to organizing data in a program is the one usually employed in OOP, as we humans view the world as a set of objects, each of which have some attributes.
The structure-of-arrays approach is often used with DOD, when computations over one attribute of many objects are done, as this allows for spatial locality.

In OOP, different objects will often have very similar functions (or methods) that only differ slightly, each method often only acting upon one object instance.
In DOD, the functions are general purpose and act upon large chunks of data.

Streaming Stores

Streaming stores are special operations that allow data to be written to main memory without first going through caches, thus bypassing part of the memory hierarchy.

They can improve performance of bandwidth-bound code if the data isn't already in the cache and won't be needed shortly after, because the streaming store will keep the cache free to use for other data.

Intel Cache Hierarchy

A typical intel cache hierarchy would have:

- The main memory on the bottom
- Above that a unified L3 Cache that's shared among all the CPU cores
- Then, for each core:
    - A unified L2 cache
    - Above that an L1 cache for instructions and an L1 cache for data
    - And lastly, registers

Cache Conflicts

Since modern caches aren't fully, but N-way associative, so each location in main memory only has a few places in the cache which it can get mapped to. Because of this, a main memory location may get mapped onto a bucket where all N ways are already occupied, causing another cache line to get thrown out even though there were still empty cache lines in other buckets.

Because the bucket that a cache line gets mapped to is at least partly determined by the memory address, cache conflicts can become a problem if memory is accessed in a regular way, such that only a small part of the cache is ever actually used, causing thrashing.