Algorithm Engineering Lab Assignment 9

Brian Zahoransky (brian.zahoransky@uni-jena.de)

February 15, 2021

1. How do bandwidth-bound computations differ from compute-bound computations?

A **bandwidth-bounded computation** is observed when its speedup is limited by the bandwidth. Typical examples are linear algebra operations with huge matrices which cannot be stored in the cache at once. On the other hand, a **compute-bounded computation** occurs when its maximal performance is restricted by compute power of the processor. Typically, this appears while performing a huge number of operations on a small memory range (every data needed fits in a low-level cache).

2. Explain why temporal locality and spatial locality can improve program performance.

Temporal locality means that data loaded in the cache is used high frequently until its no more needed. An example in the lecture was a block-wise matrix-vector multiplication. Assume, the vector does not fit in the cache. If the algorithm goes row-wise through the matrix, the vector is loaded multiple times into the cache. That is the opposite of temporal locality. Multiple loads of a vector can be avoided by splitting it into parts which can be stored in the cache at once. Now, the matrix block is multiplied with the corresponding vector part and, the interim result is stored. This procedure is executed for each part of the vector.

Spatial locality means that data used by nearby operations is stored as compactly as possible. That should avoid loading data which is never used while it is in the cache. As an example was given the summation over all elements of a two-dimensional array in the lecture. Consider two options, first, going row-wise through the array and, second, going column-wise through the array. It is also assumed that the array is stored row-wise and one column is too long for holding it in the cache. Thus, each cache line has to be loaded multiple times in the column-wise algorithm. On the other hand, using the row-wise algorithm cache lines have only to be loaded once.

3. What are the differences between data-oriented design and object-oriented design?

Object-oriented design means transferring objects of the real world into objects of a programming language. The goal is to find an easily understandable abstractions for humans. This approach usually does not care that much about performance. On the other hand, the main target of **data-oriented design** is a high performance. Since data-oriented design adapts better to how the machine works, the code might be harder to understand for humans. It is more likely applied to large chunks of data and structures data as closely as possible to the output format. It enables efficient usage of the caches and is easier to parallelize.

4. What are streaming stores?

Streaming stores are operations which store values directly in main memory without storing them in the caches. It follows the example from the lecture. If the results of a computation should be stored in a big array and this array is not used for a long time, the program might run faster when the results are stored directly into the main memory. Another requirement to achieve better performance due to streaming stores is that the data which should be written must not be read before. It also only speeds up the program if the algorithm is memory bound.

5. Describe a typical cache hierarchy used in Intel CPUs.

The intel processor form the lecture has three levels of caches. Directly next to the core is the **register**. It allows direct data access and can only hold a small amount of data. The **level one cache** is separated into two parts, the L1-data-cache and the L1-instruction-cache. Each part has a capacity of 32 KB and can be computed by multiplying its associativity (8), block size (64 B), and sets (64). Consider the cache as a hardware hash-table. Sets define the number of buckets, and the associativity specifies how many blocks a single bucket can store. The data and the instructions need four cycles to reach the processor. Behind the level, one cache follows the unified **level two cache**. Unified means that instructions and data are no longer stored separately. The level two cache has a size of 256 KB and an access time of 10 cycles. The last stage before the **main memory** is a unified **level three cache**. Typically, it is shared between the cores. Its size is 8 MB and, the processor needs 40-75 cycles for access data. Cache sizes and access time as well as associativity, block size, and sets may differ between different processors.

6. What are cache conflicts?

Cache conflicts occur if a cache bucket overflows. This happens when a cache bucket already holds as many blocks as defined by its associativity and, a further block is assigned to the bucket. In other words, assume the cache as a hardware hash table; thus, they can suffer from hash table conflicts. Such conflicts can be triggered by accessing the memory in specific address intervals.