Computer Architecture Assignment 4

Neale Ratzlaff

30 September 2015

1 Memory Optimization

The author does a brief review of what the cache is and the important aspects that are needed to understand that cache: cache, lines, and associativity. They go though what the cache heirarchy and what it means to have different levels of cache. The topic of the presentation is how one might avoid cache misses. Cache misses happen because of three main causes. Cache misses due to capacity, i.e. the datd simply cannot be held in one level of cache. There are also misses due to conflict, when the processor tries to map overlapping data to the same line. Finally there are unaboidable misses where data is prefetched and cannot possibly already be in cache. There are ways to avoid most misses in cache. The use of the cache by a program can be optmized by taking advantage of temporal and spatial locality in code. Temporal locality means that if data is going to be processed more than once (array or buffer) the it is better to do all the operations when the program loads the data into cache the first time, instead of doing sparse instructions and making to program load and unload different data sets. Spatial locality refers the speedup gained by placing the code that uses the same piece of memory close together in the program. Spatially local code is usually temporally local. There are various GCC directives that are made for optimizing the program's use of the cache, attribute and restrict are some of them. The author goes through various ways to store and process data efficiently, like trees, unrolling loops, and hot-cold splitting. The main take away from these is the programmers should beware of large structures that can take up more memory than they need to, also contiguous is king. Data structures that allow rapid access by loading the whole thing into cache are very efficient. The author goes on to warn programmers about aliasing. Aliasing occurs when an array is accessed out of bounds. Avoiding this kind of aliasing is trivial. But it can become more difficult when there are structures up against the wall of another. Aliasing also occurs when more than one name points to a spot in memory. When one pointer is changed, all the other variables that point to that spot in memory are also changed, though the programmer may forget. Using type-based alias analysis may eliminate this as it forces the compiler to establish the value as belonging to the correct type, so there is never an issue with data compatibility.

2 What Every Programmer Should Know About Memory

This author also goes through an introduction to the cache, explaining that it is the fastest way to process data in the CPU. The author stresses the disadvantages of adding to cache any data that is smaller or larger than line size, as it causes a slowdown over processing data that is exactly line size. Caches can by sped up by making them fully associative, but it comes with so much complexity that it becomes impossible to do with anything other than a very small cache, TLBs in general are the only caches that are fully associative. Cache size can be calculated with (cache line size * associativity * set number). The caches can be seenby taking a large enough array and iterating through it, accessing each element in turn. This will take a certain amount of time per access for L1, and a larger amount of time per access for L2. If elements in the array are accessed randomly, it becomes far more unstable as the CPU only loads in sequential memory addresses to cache, which may or may not be used by the program. The cache is always in sync with main memory. When the CPU loads data into cache, it then immediately loads the same data into main memory, assuring that there are no discrepencies in memory. If data is changed in cache, the cache line is marked as dirty, and the processor deals with the change in main memory as it can. There is multiprocessor/thread support for cache, but it makes everythin far more complicated. There is one L1 cache per core, and one L2 for the whole CPU. So cache lines on multiple levels will be changed, marked, and merged into main memory. The author ends with discussing cache misses and how they apply as in the previous section.