Drake Bridgewater & Ryan Phillips
May 15, 2014
CS 472 LAB 4

Summary of What Every Programmer Should Know About Memory article

As the documented suggested I skipped to section 6 as other assignments need our effort too. The content is actually really interesting and I we will have to read it all at a later time, but it is intended to advise software developers "on how to write code which performs well in the various situations" [1, p. 2]

A CPU without a cache is rare as caches can cover up most of the cost of random access this is due to the advancements we have made in optimization of un-cached read and write. As a programmer performance is essential in this world of "I want now." Therefore as Drepper states "changes affected the level 1 cache... will likely yield the best results" [1, p. 49] This can be achieved by aligning code and data and improving locality. This is easier said then done as optimization of L1i is done at the instruction level meaning that unless the programmer writes assembly then you are relaying on the compiler, but as a programmer you still can "indirectly determine the L1i use by guiding the compiler to create better code." [1, p. 55] Some ways of creating better code as far as speed is to when a function is called a single time to ensure it is executed inline; when using gcc you can add the always_inline function attribute at compilation time to instruct the compiler to move functions in with the code. This can become a problem the function is called multiple times as it will create many more instructions. For the next level of cache L2 and higher you want your code to dynamically adjust itself to the cache line size.

Another way to hide latency, which is implemented on many of today's processors, is the use of pipelining, out-of-order execution, and prefetching. Prefetching can be triggered by certain events (hardware prefetching) or explicitly requested by the program (software prefetching). [1, p. 61] With software prefetching the programs don't have to change, but ensuring the access patterns don't happen across page boundaries is key.

Multi-threading can increase overall speed, but there is concurrency optimization, atomicity optimization, and bandwidth considerations that need to be considered. Concurrency optimization is optimizing the code to prevent other threads from accessing cache lines in 'E' (exclusive) state. This can be done by grouping like access levels such as read-only and read-write commands. While optimization for concurrency

Summary Memory Optimization article

References

[1] Ulrich Drepper. What every programmer should know about memory. November 2007.