ECE472: Computer Architecture Homework 4: Part1

Rattanai Sawaspanich

I. Memory Optimization (PowerPoint)

IN the past twenty years, the speed of a CPU has increased apporximately 60% per year while the memory speed has only increased by about 10% a year. To close the performance gap, the memory read/write needs to be optimized from a user persepctive.

A. Terminology

Caches is a local memory that can be access at a faster speed than a generic memory. There are two type of caches in general: Instruction cache and Data cache. **Instruction cache** is for instruction set and opcodes to be stored. **Data cahce** is for program data to be store.

Cache Lines are lines of memory where caches are divided into either32 or 64 byte sections. **Direct-Mapping** is a method to map a memory to a corresponding cache line. **N-way set-associative** is a way to map a logical cache line to the physical cache line (a bus).

B. Architecture Overview

Caches are typically divided into different levels e.g. L1, L2, and main memory.

L1 cache is the fastest not-on-CPU memory that a CPU can access. Generally L1 cache has a fairly small storage size in the magnitude of 8KB and takes about 1 clock cycle to access a memory in L1 cache. Note: the smaller size storage is due to the physical hardware limitation.

L2 cache is slower cache than L2. Though, it has a larger storage that can go up the magnitue of 8MB. On the average, it takes a CPU about 5 to 20 clock cycles to access a memory in an L2 cache. Let's assume the system is a 2-level cache heirachy. After all the caches have been depleted, the CPU needs to access the main memory.

Main memory or **system memory** is the biggest and slow source of memory. The storage magnitude of the system memory can be in the order of gigabyt to terabyte. It usually takes a CPU about 40 to 100 clock cycles to access the memory.

C. Optimization

1) Causes of cache misses:

Compulsory Misses are invitable cache misses occur when the data is read for the first time.
Capacity Misses are cache misses occur when the active data requires more memory than a cache storage. This type of miss usually happens when there is too much data accessed.
Conflict Misses are cache misses due to the cache line conflict – a cache mapped to the same cache line. Note: a cache line can only handle a set of data at a time.

2) Key to Optimization:

Reuse the set of data (or variables) to increase the temporal and sparial locality which helps avoiding multiple compulsory cache misses.
Rearrange the order how the data is being stored to change the layout and increase the spatial locality. The rearrangement helps with the capacity cache misses.
Reduce the number of a cache line read to help with cache conflict misses – the lesser accessed data, the lesser cache collision there will be.

3) Optimization Techniques:

Reuse: Prefetch

Software Prefetching – use an arbitrary look-ahead to prefecth the data in cache before needed by CPU
Greedy Prefetching – prefetch all the data that will be used in the next two instructions
Preloading – ask for a data to be used and while it is waiting for the memory I/O has a CPU perform other tasks first.

Rearrange: Structures

Cache-conscious layout – rearrange the data that typically access together next to each other i.e. declare the variables next to each other, or using array declaration.
Use structures – use tree structure to increase spatial locality and reduce the size of pointer used by storing data relative to the parent node e.g. Breadth-first order, Depth-frist order, Van Emde Boas layout, compact static k-d tree.

Linearlized data – store and fetch data in a linear manner using static memory pool. The method yields the best performance. The linearization technique can be used on top of hierarchy data structures to prevent blocking overhead and keep data together with minimal fragmentation. It is highly recommended to allocate memory pool from stack, free the block immediate if not needed, and reuse the memory if possible.

D. Alising

Alising occurs when multiple references point to the same memory location. This can prevent compiler from optimizing the code by prohibiting the reordering (badly impact instruction scheduling), eliminating of loads (hinder looping optimization and subexpression elimination), and storing of the data.

1) Causes of Alising:

Pointers are the major causes of alising specially shared pointers. A compiler does not know how to optimize the program because the value stores in the pointed location can be changed any time at runtime. A compiler does not have access to the runtime values; there is no optimization at compile time.

Global variables/ Class Members can cause aliasing because at the end they can be resolved as shared pointers.

2) Aliasing Prevention:

Program Better: When a CPU fetch information from a cache, the whole cache line is returned. Anti-aliasing can be done using the fact, the data only needs to be fetched once for every values in the same row (assuming a row major language).
Lower Language Abstraction: The more abstract, the more abstraction penalty there is. The code can only breakdown to at most a generic code block which prevents any special customization. Moreover, an abstraction takes away the transparency of data and its operations e.g. temporary objects, implicit aliasing pointers, etc. This can introduce aliasing issues.
Use a Better Compiler some compiler is better at anti-aliasing than the other. Note: use fstrict-aliasing in GCC to turn on anti-aliasing feature.

Potpourri

- Minimize use of pointers and global variables.
- Use local variables as much as possible.
- Avoid operation that requires temporary variables.
- Declare variables close to where they are used.
- Use ’const’

II. What Every Programmer Should Know About Memory

A. The Current Hardware

Personal Computer System
In the past twenty years, personal computers have become more popular. The chipset of such computer has divided into two parts: the Northbride and the Southbride. All the CPU are connected via a common bus to the Northbridge. **The Northbride** takes part in memory control all different type of RAM. **The Southbride** acts as an I/O bridge handling the communication with different devices via peripheral buses e.g. PCI, PCIe, SATA, and USB buses.

It is worth note taking that all of the data communcation from CPU to CPU ,and all RAM communication can only be done through Northbridge. This can cause a bottleneck in the I/O data stream. This is not to mention that RAM in the system is a single full-duplex port. Instantly, there are two bottleneck spots in the system. The first bottleneck occurs when RAM tries to communicate with the different devices via the bridges. The data has to route through the CPU which can slow down the entire sytem. The second bottleneck occurs when a bus from the Northport tries to communicate with RAM.

Advanced Computer System
On the more expensive and advanced computer system, the Northbride does not contain only a single memory control; rather, a number of external memory controllers. The more memory controllers allow a higher bandwidth to be used in the system. Though, multiple external memory controller is not the only solution to increase the memory I/O bandwidth. Another way to increase the system bandwidth is by integrating the memory controller in the CPU itself along with an integrated on-chip memory.

There is no free lunch in a more advanced system. The multi-memory controllers system needs make all of its memory accessible to all the processors. Moreover, not all of the memory can be accessed with the same constant time. The scheme name for this multi-memory controller is called **NUMA** or **Non-Uniform Memory Architecture**. The reason behind its non-uniform accessing time is because the on-chip local memory can be accessed with the usual speed while the memory that is attached to a different processor needs to be transfer via the interconnect buses.

B. Types of RAM

There are basically two types of RAM: Static RAM (SRAM) and Dynamic RAM (DRAM). SRAM is a lot faster at accessing the memory than DRAM. The only reason why DRAM exists is because of the cost to produce SRAM.

Static RAM (SRAM)
A unit of SRAM is built off of six transistors that forms two cross-coupled inverters. Though, in some designs, only four transistors are needed. These cell state is ready to be

read at almost instant given the cell state does not need a refresh cycle. There are some cons to this type of RAM is that it requires a constant supply of power to maintain its state.

Dynamic RAM (DRAM)
A unit of DRAM is comprised of one transistor and one capacitor. The transistor is used to guard the access state. The reading process of DRAM is to discharge the memory cell. This task cannot be repeated indefinitely, the capacitor wears out and can have electrons leakage. Moreover DRAM cannot be instantly read/write as if SRAM, it requires sensing amplifier to translate the voltage into either 1 or 0.

C. CPU Caches

To obtain a high performance system, the memory I/O should hypothetically run at the same speed as CPU to yield the maximum performance. Though, it is not possible due to the law of physics – a capacitor in DRAM takes time to charge up. Fortunately, SRAM can operate at a much faster speed than DRAM. In this case, SRAM can be use as a medium ground for a DRAM to communicate with the CPU. Though, it is not a viable option to have a section of memory as SRAM and the rest DRAM due to the complication in logical address mapping. So instead of hanving a giant pool of mixed SRAM-DRAM memory under the control of an operating system or a user, SRAM can be used an internal memory for a processor to use. It is used to make a temporary copies of data in main meory which is likely to be used soon by the processor. The on-processor SRAM is called a cache.

Typically the size of SRAM caches is many times smaller than the system memory to keep the cost down and prevent crowded effects. In the cache configuration module, CPU is no longer directly connected to main memory. All the data needs to be loaded and go through caches first before entering the CPU. More recently, an instruction cache is introduced to maximize the instruction decoding processes. At this point there is a huge speed gap between an internal processor SRAM and system memory DRAM. To close the speed gap, more level of caches were introduces. The higher the bigger storage size , but the slower it is.

In the more recent design, each processor can have multiple cores and each core can have multiple threads. Note: the only different between thread and core is that a separate cores have separate copies of all the hardware resources and can run completely independent from each other. On the other hand, threads shared almost all of the processor's resources. The cache storage needs to be able to facilitate the needs of memory resources for all the threads and cores.

D. Caches Operation

If the CPU needs a data word that cahes are searched first, the cache cannot have the content of the entire memory, but since all the memory is cacheable, each cahe entry is tagged using the address of the data word in the main memory. Given the concept of spatial locality is one of the principles on cahces, the neighbor memory is likely to be used together; it should also be cached. When the data in cache is need by the processor, the entire cache line is loaded into the L1 cache which forwards the whole thing to the processor.

In some cases, there is a chance that some information needed by the processor is not stored in a cache (cache misses). The processor needs to fetch the entire new data from system memory.

E. Associativity Cache

It is possible to have a cache where each cache line holds a copy of any memory location (also known as **fully associative cache**) by tagging the entire part of the address which is not offset into the cache line. The only cache to this method is the storage needs to store the tags lookup would be gigantic given the current standard size of L1 and L2.

F. Measurement of Cache Effects

The simplest case is a simple walk over all the entires in the list. The list element are laid out sequentially. What needs to be measured here is how long does it take to access a single list element. The number of workload where there is a spike in the element accessed time is the point where the workload exceeds that cache level capacity, so the processor needs to fetch the information from the next cache level which causes the increases in accessing time.