Jacob Chmura

Single Processor Computing

Modern Processors

Unlike the *Von Neumon Architecture Modeel*, modern processors **pipeline** instuctions to get *instruction level parallelism*, and allow for **out-of-order** instruction handling.

There are many **floating point units (FPU** within each **core**. There may also be a **Fused Multiply-Add (FMA)** unit which can execute the instruction $x \leftarrow ax + b$ in a single clock cycle.

Pipelining

A stream of independent operations can be performed at an asymptotic speed of one result per clock cycle.

Each instruction roughly corresponds to the following stages of pipeline:

1. Fetch and decode the instruction
2. Copy the operants into registers
3. Execute instruction
4. Store the result back in memory

As long as there are no data race conditions, and each stage is indpendent, we gain peak performance through this *instruction level parallelism.*

Other techniques include:

- Multiple issue instructino that are independent can be started at the same time
- Branch prediction and specaulatibe execution via the compuler
- Out of order execution on independent instruction
- Prefetching data speculatively

Definition 1. Flops measure the number of floating point operations per second, and can be approximating by clock speed $\times$ number of independent FPUS.

8-bit, 16-bit, 32-bit, 64-bit

These refer to width of path between processor and memory, the extend to which main memory is addressable, the number of bits in aregister, the size of floats.

Memory Hierarchies

The **memory hierarchy** consists of various registers and caches, along with main memory. The goal is to make recently used data available quicker than it would be from main memory.

Buses

Buses move data around from memoyr to cpu or disc. The **Front-Side Bus** connects the processor to memory, also called the **north bridge**. The **south bridge** connects to external devices.

Definition 2. Latency is the delay between the processor issuing a request for a memory item and the item actually arriving.

Bandwidth is the rate at which data arrives t its destination, after the initial latency is overcome.

Registers

Registers are what the processor actually operates on. They have a small amount of memory internal the processor.

Caches

Caches sit between registers and main memory. There are various levels of cache memory. L1, L2 caches are part of the processor chip while L3 is off-chip. L1 has a seperate *instruction cache* and *data cache* whereas the larger caches store data and instructions in the same place.

Cache misses can be either:

1. Compulsory cache miss: first time referencing data
2. Capacity cache miss: cache must evict since it cannot store enough data
3. Conflict cache miss: multiple data itmes being mapped to same cache location while both needed for computation
4. Invalidation cache miss: in multicore, item becomes invalid because other core changed the value

Cache items are typically evicted using **LRU least recently used** policies.

Definition 3. A **cache line** or **cache block** is the smallest unit of data moved between memory and cache, typically 64 or 128 bytes long, i.e. 8 or 116 double precision floats.

Definition 4. A **cache mapping strategy** maps address from main memory to address in cache.

Direct mapped caches simply takes the least significant bits and uses these as teh addres of the data item in cache.

A **fully associative cache** is slower, but has less conflict than direct mapped cache, and allows any data item to go to any cache location.

A **k-way associative cache** is the most common solution, and in this case a data item can go to any of k cache locations.

Prefetch Streams

The memory system can try to detect data patterns by looking at cache miss points and request a *prefetch data stream*.

Lemma 5. Little's Law*:*

$$Concurrency = Bandwitch\ x\ Latency \tag{1}$$

TLB, pages, Virtual memory

All of a prgorams data may not be in memory at once since:

- Computer is running multiple programs that together need more than the physically available memory
- One single program can use more data than available memory

And so we use **virtual memory**: if more memory is needed than available, certain blocks of memory are *swapped* to disc. *Swapping* acts on **memory pages**: contiguous blocks of memory.

Definition 6. The **page table** is a mapping between virtual and physical memory.

Definition 7. A **translation lookaside buffer or TLB** is a *cache* of frequently used page table entries: providing fast address translation for a number of pages.

Multicore Architectures

In recent years we have hit limits on single processor design since clock frequency cannot be increase without degrading due to energy consumption, and we cannot extract more ILP.

Multiple cores are more efficient since two cores at lower frequency can have same througput as single processor at higher frequency. We can also exploit explicit task parallelism in our programming.

Typically each core has its own L1 cache, but they share L2,L3 caches.

Cache coherence

The problem of ensuring that all cached data are an accurate copy of main memory is referred to as *cache coherence*. Each core must keep copy updated.

The process of updating or invalidating cachelines is called *maintaining cache coherence*.

- *Snooping*: any request for data is sent to all caches
- *Tag directory*: distributed directory contains information on what data is present in what cache

OpenMP is a shared memory and shared cache based threaded system for CPU execution. It can dynamically change based on architecture.

Definition 8. NUMA Non-Uniform Memory Access, for a process running on some core, memory attached to its core is slightly faster to access than others.

Locality and Data Reuse

Definition 9. If an algorithm operates on n data items and takes $f(n)$ operations, the **arithmetic intensity** is $\frac{f(n)}{n}$.

For example, basic matrix multiplication involves $3n^2$ data items and $2n^3$ operations hence arithemtic intensity is $O(n)$: each item will be used $O(n)$ times.

Temporal locality is that data items will be re-used once used within a short period of time

Spatial locality is that data items closed to already fetched data will typically be used.

Programmming Strategies for High Performance

- Loop unrolling for pipelining and using more registers
- Loop tiling / Cache blocking to keep data in cache
- Strided access to take advantage of cache lines
- OpenMP parallelism short inner loops, GPU large inner loops with no branches

Parallel Computing

Definition 10. Data parallelism is when the same instruction is applied to many data items at once.

Definition 11. Let T_1 be execution time on single processor and T_p be execution time with p processors. Then **speedup** is defined as $\frac{T_1}{T_p}$ and the **efficiency** is given by $\frac{S_p}{p}$.

We rarely get full efficiency due to communication overhead, load unbalance, and possibly requiring changes to realize parallel ops.

The **critical path** is the longest data dependence path in an algorithm that therefore determines a lower bound on how much can be run in parallel due to serial ops.

Theorem 0.12. *Let m be the total number of tasks, p the number of processors and t the length of the critical path. Then the computation can be done in:*

$$T_p = t + \frac{m-t}{p} \tag{2}$$

Theorem 0.13. *Let F_p and F_s be the parallel and sequential fraction respectively. Then*

$$T_p = T_1(F_s + \frac{F_p}{P}) + T_c \tag{3}$$

where T_c is a communication overhead

Parallel Computer Architecture

Definition 14. There are roughly 3 main types:

SISD: Single Instruction Single Data: Traditional CPU architecture, a single data item is executing a single operation

SIMD : Single Instruction Multiple Data: multiple processors, each operating on their own data item, but performing the same operation, for example GPU, or ARX vector instructions.

MIMD: Multiple Instruction Single Data: multiple CPU cores execute their own instructions on their own data items.

Different types of memory access

In a *distributed memory setup* each process has its own physical memory and its own address space whereas in shared memory there is a single address space.

Symmetric Multi-Processors or UMA: Uniform memory access is such that any memory location is accessible by any processor.

NUMA: Non-Uniform Memory Access is more scalable wherein each process has fast access to its own memory via physically distributed memory but maintaining logically shared address spaceso that any process can still access any memory location.

In the extreme case of NUMA there is a *network layer* between each process giving rise to distributed shared memory. A *hypervisor* is a frontend that ensures cache coherence.

Parallel Programming

A unix *process* corresponds to the execution of a single program, which has in memory the program code, heap, stack, program counter file handle.

A *thread* is an independent strand of computation within a process, and therefore can share data easily.

When we create more threads than CPU cores, we will need to do a **context switch**, which is not free. This can be bypassed with *hyperthreading*

- Shared memory between makes it easy and fast to program independent threads of execution
- But synchronization and ensuring no race conditions needs to be implemented by the programmer

Definition 15. Code that executes the same whether its executed sequentially or threaded is called **thread safe**

OpenMP is an extension to C consistencing of compiler level derivatives that offer thread based dynamic parallelism.

MPI is the standard way to program distributed memory, as opposed to the shared memory offered in OpenMP.

- Flow Dependency: read after write
- Anti Dependency: write after read
- Output Dependency: write after write

GPU

GPU performs identical operations on many data elements in the form of data parallelism.

Threads are ordered in *thread blocks* where all threads in the block execute the same intstruciton making it SIMD.

It is also possible to schedule the same instruction stream (a *kernel*) on more than one thread block.

The collections of cores executing a GPU kernel is known as *grid* and it is structured as a set of *thread blocks* and *threads*. Each thread bock can have up to 512 threads.