Chapter 1: Parrellel Computing Hardware

1.1: Sequential vs. parallel computing

Sequential

- One step at a time, the program is broken up into each individual step, one thing can only happen at a time.
- Only one intstruction can be exectued at any one moment, there is no overlap between the instructions.
- The time that a sequential program takes to run is limited by the hardware the program is ran on,

Parrallel

- Tasks can run at the exact same time. There is overlap between the steps, hence saving a lot of time, making the exection much faster.
- Parrallel programming requires extra effort inorder to communicate with the other threads in order to complete tasks that are dependent on one another
- completexity is added in order to improve the speed.
- Parrallel Extrectuon allows you to increase the throughput (the rate at which something is processed) of a program.

1.2: Parrallel Computing Architectures

- Parrallel Programming requires having hardware that is parraellel; parrallel processors.

Flynn's Taxonmy

- Used to describe multiprocessor architectures based on two factors:
    - The number of concurrent instruction or control streams
    - The number of data streams.

SISD - Single Instruction, Single data stream

- Sequecntial computer with a single processor unit.
- Can operate one insrction at a time, and can only process one piece of data at a time.
- Example to a kitchen: one person cutting one carrot.

SIMD - Single Instruction, Multiple data stream

- Parrallel computer with multiple processing units.
- All processors exectute the same instruction at any given time, but they can each operate on different data element.
- Example, two cooks in a kitchen, but one is cutting carrots, and another is cutting celery, and they are cutting in sync

MISD - Multiple instruction, single data stream

- Each processing unit has its own intrcutions, but all of the processors have the same data to

handle.
- Example in a kitchen, two cooks, one is peeling the carrot, and the other is cutting the carrot at the same time.
- This does not make pratcial sense, hence, not commonly used.

MIMD - Multiple instruction, Multiple data stream

- Every processor is processing a different set of instrcutions, with different pieces of data.
- Example in a kicthen is one cooking is peeling carrots, and other is chopping celery.
- Most commonly used architcture, and can be divided into two different subcategories:
    - SPMD - Single Program Multiple Data
    - MPMD - Multiple Program Multiple Data

SPMD - Single Program Multiple Data

- Multiple processing units are executing the same program, however each of them have different data.
- This seems similar to the SIMD, however, the processors can operate asynchronously. The program usually includes conditional logic that allows different tasks within the program to only execute specific parts of the overall program. ---- A simiplier way to put it is that each processor is made response for a specfic part of the task (I think).
- This is the most common parrallel programming

MPMD - Multiple Program Multiple Data

- Processors will be executing different independent programms at the same time, while handling different pieces of data. One processor is typically has one program running on it, and it will be handing out data to the other processors.
- Not as common as SPMD, however, it is useful for some applications that lend themselves to functional decomposition (not sure what this is but it is covered later on).

1.3: Shared vs. Distributed Memory

- Memory is a key part of the hardware arch of a parrallel computuer. Even if you have many processors, but the computer is not able to access the memory of the computer fast enough than the computer will be extremley slow.
- Memory speed << Processor Speed.
- When a block of memory is being used (read/write) by one processor, than all other processors are blocked from using that piece of memory.
- Two types of memory exist for parrall Programming: shared and distributed memory.

Shared Memory

- All processors have access to the same memory as part of a global address space. If one processor changes a memory location all of the other processors will see that change.
- all of the processors can see what is ahppening in shared memory speace.
- Two different types:
    - UMA - Uniform Memory Access

    - NUMA - Non-Uniform Memory Access

- Types of shared memory are based on how the processors are connected to memory, and how quickly they can access it.

UMA - Uniform Memory Access

- All of the processors have equal access to the memory; they can access the memory equally fast.
- The most common form of UMA is an SMP (symmetric multiprocessing system).
    - two or more identical processors connected to a single shared memory.
    - each core has its own cache so it can access data that it is accessing the most often.
    - Chache Coherncy: challenge with cache is if one processor copies a value from the shared main memory, and then makes a change to it in its local cache, then that change needs to be updated back in the shared memory before another processor reads the old value which is no longer current: handled in hardware.

NUMA - Non-Uniform Memory Access

- Made by Connecting multiple SMP systems together.
- some processors will have quicker access to some parts of memory compared to others.
- Shared Memory cannot be scaled properly, hence not the ideal soltuion:
- Cache Cohenrcy becomes a more apparent issue when it comes to this.
- Puts responsibilty on the programmer to ensure that memory accesses to ensure correct behaviour.

Distributed Memory

- Each processor has its own local memory, with its own address space.
- Concept of global address space does not exist.
- Each processor is connected to a network (ex. ethernet).
- Each processor is independent, hence changes made to memory are not reflected in other processors; it is up to the programmer how data will be distrucbited between the different processors. This is a disadvanttage communication is hard.
- Very scable, adding more processors add more memory to the system.