DSC-204A: Scalable Data Systems, Winter 2024

3: Basics: Computer Organization, Operating systems, Storage

Lecturer: Hao Zhang *Scribe: Trevor Tuttle, Bavanya Kurra*

1 Review Questions

- **Question:** What is the space needed to store GPT-3?
  Ans: GPT-3 can be interpreted as a software (for eg: python file) with some built-in data (trained weights). Hence, we need to figure out the space needed for the code (a python file, which is just a list of strings) and the weights. The space taken by the weights is extremely large compared to the code file (usually only in KBs). GPT-3 has 175 billion parameters and each parameter is 16-bit in size, since they are represented in Bf16 format. Therefore, GPT-3 needs approximately 350 GB (we can ignore the space taken by the code because it will only require space in KBs compared to 350 GB).
- **Question:** What do exponent and fraction control in floating point representation?
  Ans: Bits allotted to the exponent control the range of numbers represented and bits allotted to the fraction controls the precision.
- **Question:** What is the difference between BF16 and FP16?
  Ans: 8 Bits are allotted for exponent in BF16 but only 5 in FP16, similarly 7 bits are allotted for fractional part in BF16 but 10 bits in FP16. So FP16 has smaller range and hence easier to overflow, but FP16 is more precise.
- **Question:** Why is BF16 better in AI/ML?
  Ans: Three reasons:
  Since AI/ML is error tolerant (lower precision in values of weights won't affect the model performance much). Values in deep learning are susceptible to overflows (eg: in the case of exploding gradients). Easier to convert between FP32 and BF16.

2 Rational Number Representations in the Computer

To understand how to calculate the space used by GPT-3, we first need to see how the weights/parameters (which are rational numbers) of the model are represented.

2.1 Fixed Point Representation

Bits to the right of the radix point represent the fractional part of the number, and the remaining bits except the leftmost bit represent the integer part. The expression for the absolute value of a rational number is:

$$\text{Absolute Value} = \sum_{k=-j}^{i} b_k \times 2^k$$

Sign is obtained from the leftmost bit (0 represents positive, 1 represents negative), where $(i + j + 2)$ is the total number of bits used for the representation. Hence FP6 (here $i = 1, j = 3$) can represent numbers from -3.875 (111111) to 3.875 (011111).

2.2 Floating Point Representation

Radix point is moved to the right of the first non-zero digit after converting to fixed-point. Expressed as

$$\text{Floating Point Value} = (-1)^s \times M \times 2^E$$

where s determines whether the number is positive or negative, M is a fractional value in the range [1.0, 2), and E is weighted by the power of 2. Bias is added to the exponent. Standard format: Binary32 (1 sign bit, 8 exponent bits, 23 fraction bits, bias = 127).

Floating-point computers consume more power. Embedded devices designed for low power use fixed-point representation. However, floating-point representation is preferred in data science and machine learning due to a bigger range (owing to the exponent part). The precision is inconsistent with floating-point representation.

3 Disk

Now that we know that GPT-3 takes 350 GB, where do we store this huge amount of data? The answer is in Disk. Now we will discuss a little about the disk, its storage capacity, and how to calculate its access time.

3.1 Disk Components

- Platters: We store the data in the platters, usually multiple platters are present in a disk to increase its storage capacity.
- Spindle: It spins the platters.
- Arm: It reads the data, can only move radially and places the read/write head over the required track.
- Actuator: Used to move the arm.
- SCSI connector.

3.2 Disk Geometry

Disks consist of platters, each with two surfaces. Each surface consists of concentric rings called tracks. Each track consists of sectors separated by gaps. Data is read from the sectors.

3.3 Disk Capacity

Since the data is stored in the tracks, we count the number of tracks to measure the storage capacity of the disk. The factors determining the disk's capacity are:

- Recording density (bits/inch): Number of bits that can be squeezed in a 1-inch segment of a track.
- Track density (tracks/in): Number of tracks that can be squeezed into a 1-inch radial segment.
- Area density (bits/in^2): Product of recording and track density.

3.4 Disk Access

To access data at another sector of another track in the disk, the following steps are performed:

- Seek action: Head of arm is moved to the track of the target sector from the previous sector.
- Rotational latency: Spindle spins the platter in the counterclockwise direction to reach the start of the destination sector.
- Data transfer action: Head of arm starts accessing the data at the destination sector of the destination track.

3.5 Disk Access Time

The average time to access some target sector is approximated by:

$$T_{\text{access}} = T_{\text{avg seek}} + T_{\text{avg rotation}} + T_{\text{avg transfer}}$$

where:

- Seek time ($T_{\text{avg seek}}$): Time to position heads over a cylinder containing the target sector. Typical $T_{\text{avg seek}}$ is 3—9 ms.
- Rotational latency ($T_{\text{avg rotation}}$): Time waiting for the first bit of the target sector to pass under the r/w head. $T_{\text{avg rotation}} = \frac{1}{2} \times \frac{1}{\text{RPMs}} \times 60\,\text{sec}/1\text{ min}$. Typical rotational rate = 7,200 RPMs.
- Transfer time ($T_{\text{avg transfer}}$): Time to read the bits in the target sector. $T_{\text{avg transfer}} = \frac{1}{\text{RPM}} \times \frac{1}{\text{avg \# sectors/track}} \times 60\,\text{sec}/1\text{ min}$.

An example discussed in the class is as follows:

Given: Rotational rate = 7,200 RPM, Average seek time = 9 ms ($T_{\text{avg seek}}$), Average # sectors/track = 400.

We obtain $T_{\text{avg rotation}}$, $T_{\text{avg transfer}}$:

$$T_{\text{avg rotation}} = \frac{1}{2} \times \frac{60\,\text{secs}}{7200\,\text{RPM}} \times 1000\,\text{ms/sec} = 4\,\text{ms}$$
$$T_{\text{avg transfer}} = \frac{60}{7200} \times \frac{1}{400} \times 1000\,\text{ms/sec} = 0.02\,\text{ms}$$

Therefore:

$$T_{\text{access}} = T_{\text{avg seek}} + T_{\text{avg rotation}} + T_{\text{avg transfer}} = 9\,\text{ms} + 4\,\text{ms} + 0.02\,\text{ms}$$

From the example, we can see that access time is dominated by seek time and rotational latency. This is the reason why copying multiple small files from the disk is slow, due to multiple seeks for the beginning of the small files. But when copying a big file which is stored continuously on the track, we perform a seek operation only once, hence access time is smaller. Also accessing the first bit of the sector is expensive due to the seek time involved.

3.6 GPT Again

Why do we care? We want to understand how GPT works. We know that the python file itself takes a few KBs of data, and the parameters take another 350GB of data. The input is read as a list of integers, which is then fitted and multiplied by the weights of the model in a way instructed by the python file. For us to receive an output, the processor needs to compute these operations according to its instructions.

4 Processors

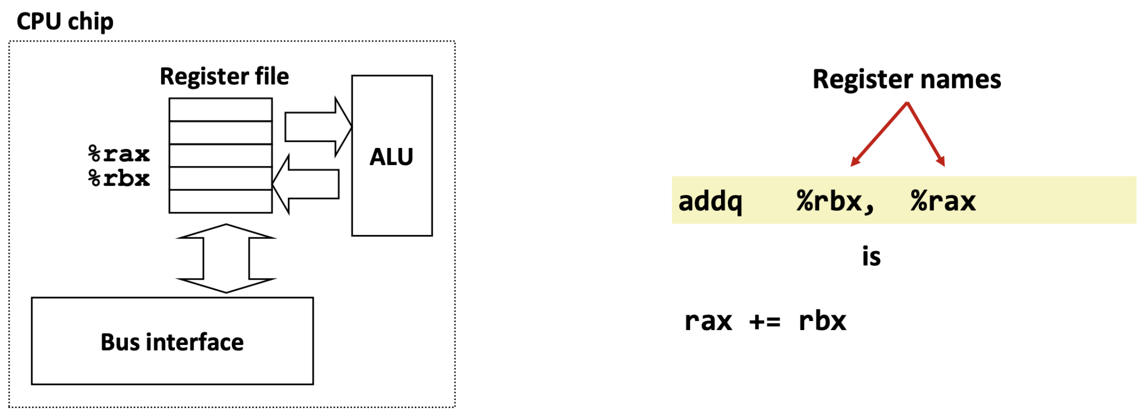
4.1 Basics

The processor is hardware that orchestrates and executes instructions to manipulate data as specified by a program: CPUs, GPUs, and FPGAs, are all examples of processors. The Instruction Set Architecture (ISA) is the vocabulary of commands of a processor. The following is a typical ISA set up:

$$\text{Program in PL} \xRightarrow[\text{compile}]{\text{interpret}} \text{Program in Assembly Lang.} \xRightarrow[\text{assemble}]{} \text{MC tied to ISA} \Rightarrow \text{Run on processor}$$

The program is first written in a programming language (PL) it is then compiled/interpreted into Assembly, after which it is assembled into machine code (MC), and finally it is run on the processor.
How does the processor execute the machine code you ask? The most common approach is called load-store architecture, which has registers. Registers are tiny local memory ("scratch space") on the processor into which instructions and data are copied. Instructions on the registers take almost no time for the processor to run. The ISA specifies both the bit length and the format of machine code commands, and they also have several commands to manipulate register contents.

4.2 Instruction



```
addq   %rbx,  %rax
```

```
rax += rbx
```

Figure 1: A CPU Chip Processor and Registers

In the figure above on the left we have a CPU chip, notably it has an arithmetic logic unit (ALU) which is where the instructions are executed. It is not human readable, but note that in the above case we have two registers named rax and rbx. The instruction is to add the value stored at rax, which is the third register in the register file, with the value stored at rbx which is the fourth register in the register file, and then set this added value to the new value at rax. This is how processors work at a very low level.

4.3 Speed of a Processor

The measure of a processor is typically done in instructions per second, which is the number instructions the processor can complete within a single second. In data science we care far more about the computation on floating point numbers, known as FLOPs. FLOPs are the number of floating point operations a processor can do, and are an important metric because they can be used to determine how long a process will take to compute.

4.4 The Problem?

The size of GPT is 350GB, which cannot be put directly into registers as such we put it onto disks. The issue with disks is that the reading speed is very slow, a hard disk drive has a read speed of between 80-160 MB/s versus a CPU which can have 100 GFLOPs/s. If we assume we use 0.5s to perform 50 GFLOPs, we would need to read 50x2 = 100GB in the rest of 0.5s to keep the CPU busy. We would need the CPU to read at the speed of 100GB/0.5s = 200GB/s. This speed is a far cry from the 80-160MB/s that is achieved from reading on disk. To mitigate this difference the computer uses a memory hierarchy.

4.5 Memory Hierarchy

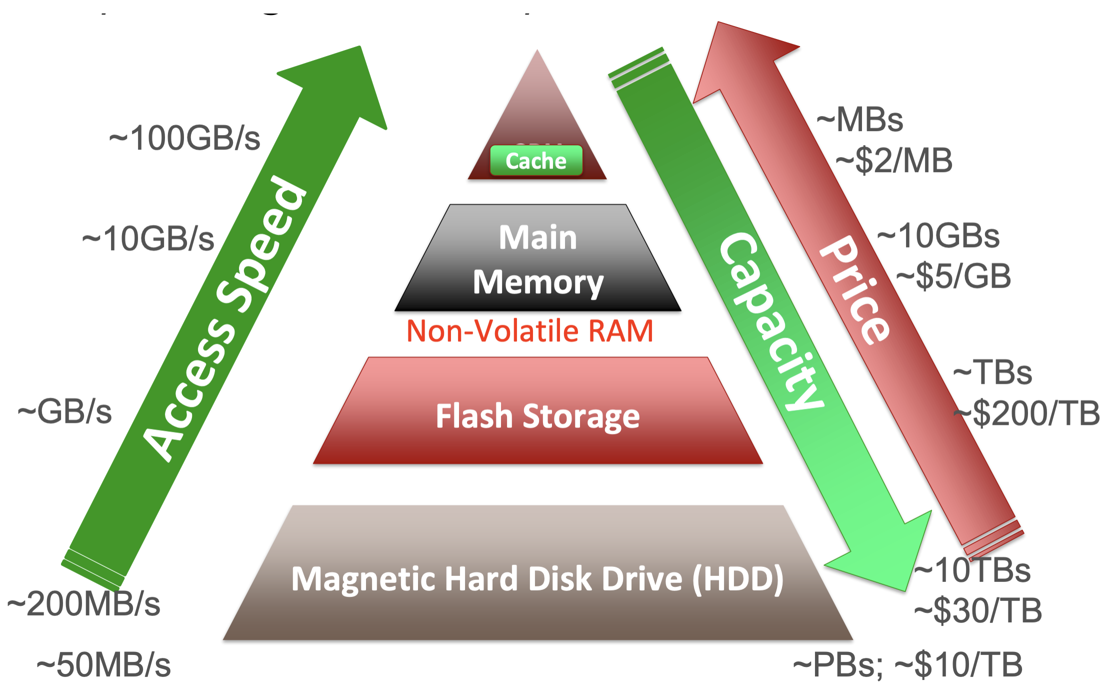


Figure 2: The memory/ storage hierarchy system. The higher on the pyramid the more expensive more storage is, as well as the faster the read speeds.

In Figure 2 we can see the typical memory and storage hierarchy system that is used by computers. At the top there is the Cache (small local memory to buffer instructions/data), which has the fastest read speeds at 100GB/s, but it is also the most expensive to increase storage for. The next down the hierarchy is the main memory, the Non-Volatile RAM, which has slower read speeds than the cache, but it is also a bit cheaper to store data at than the cache. Next is flash storage (SSD), which is around 1 GB/s read speed, but about $200/tb. The cheapest, and slowest is the Hard Disk Drive (HDD), with read speeds of 50-200MB/s but it is also the cheapest at about $30/tb. The faster the read speed, the more money one will spend, and one will have the lower the capacity of storage. It is important for the ISA to have the capability of both reading and writing into memory. To write is to transfer data from memory to the CPU, and to read is to transfer data from the CPU to the memory. The former is known as the ”store” operation, and the latter is the ”load” operation. The Bus is used to connect memory with the processor as such, the way that the processor interacts with the memory is through the Bus.

4.6 Connecting CPU and Memory

A Bus is a collection of parallel wires that carry address, data, and control signals and they are typically shared by multiple devices. In our CPU chip, there is a Bus interface that connects with the System Bus, which is then connected to the Memory Bus. The first step is that CPU places an address, A, on the memory bus. Then the main memory reads A from the memory bus, retrieving the data, and after which it is placed back on the bus. The CPU reads the data from the bus and copies it into a register (such as rax that we spoke of earlier) after which it can perform whatever instruction is necessary.