Scientific and Technical Computing

Hardware and Code Optimization

Lars Koesterke

UT Austin, 9/1/22 & …

Discussion

What are the **primary components** of a computer?

Discussion

What are the **primary components** of a computer?

Can you detect some limitations?

Discussion

What are the **primary components** of a computer?

- CPU
- Memory
- (Storage)
- (Motherboard, lots of ‘wires’)
- (Keyboard, screen)

Can you detect some limitations?

- Number of pins connecting CPU and Memory to motherboard

What **sciences and technologies** are

involved in designing and building a

CPU/Memory/Computer?

What sciences and technologies are involved in designing and and building a CPU?

- Electrical engineering, physics, chemistry, math … (all the things you study)
- Cutting edge research
- A lot of institutional knowledge by people in companies and research institutions
  - Not everything is an exact science; many tricks are applied (a bit like cooking)
  - Design decisions are made on incomplete facts (humans weigh the pros and cons)
- Certainly, many computers using current/previous generations of CPUs

This is **not** what we will talk about in this section of the class segment (Hardware and Code Optimization)

Scope of this class

What are we going to explore in the next weeks?

High level overview of the architecture

- High level of abstraction
- Simplified implementation details
- Features that allow for a very high peak performance

How to write code that exploits the hardware features?

Scope of this class

Matching software to hardware: Why?

Much higher performance

- Orders of magnitude!

There is, of course, the idea to match the hardware to the software/purpose

This is done in other areas
In HPC the idea is not feasible (with one notable exception)

Conceptual understanding of hardware features guides software design

Assumption: You are in this class (and other TACC classes) to learn how to

- learn about high-performance computing, HPC (language implications)
- use a supercomputer (or any computer!) in an **efficient** and **effective** way
- write **fast** code
  - This class: exploiting parallelism of the hardware
  - Note: some/many bad code design decision cannot be reversed later
- write **parallel** code with OpenMP and MPI (PCSE in the spring semester)

Getting scientific calculations done
Better than the competition

Terminology

Today's supercomputers are clusters

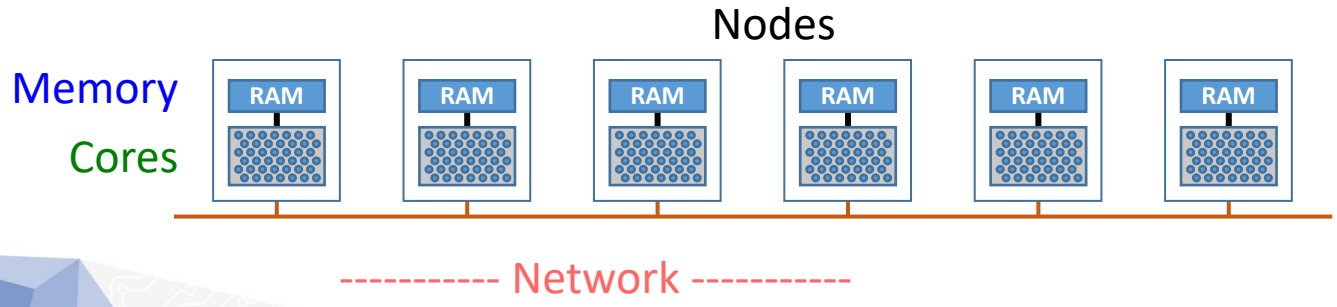
Components of a cluster

- Nodes
  - CPUs and memory
- Network
- (File system)

In this class we strongly focus on a single-core and single-node

A cluster is built from individual computers which are called nodes

The nodes are connected through an interconnect



Discussion

What is a clock tick?

What is a clock cycle?

Discussion

What is a clock tick?

What is a clock cycle?

What does this mean?

Clock frequency = 2.2 GHz

Discussion

What is a clock tick?

What is a clock cycle?

Think of an assembly line

- Smallest unit of time to ‘do’ something
- One or multiple instructions are executed
- An instruction may take several cycles

Examples of instructions are

- Multiply two numbers
- Load data into a register

‘Instruction’ can mean many things. Let’s leave it a bit vague for now

Some answers from the web

Computers use an internal clock to synchronize all of their calculations. The clock ensures that the various circuits inside a computer work together at the same time.

Same as a cycle, the smallest unit of time recognized by a device. For personal computers, clock ticks generally refer to the main system clock, which runs at 66 MHz. This means that there are 66 million clock ticks (or cycles) per second. Since modern CPUs run much faster (up to 3 GHz), the CPU can execute several instructions in a single clock tick.

"The processor clock coordinates all CPU and memory operations by periodically generating a time reference signal called a *clock cycle or tick*. Clock frequency is specified in gigahertz (GHz), which specifies billions of ticks per second. Clock speed determines how fast instructions execute. Some instructions require one tick, others multiple ticks, and some processors execute multiple instructions during one tick."

Intermission

Let’s talk about how to proceed

I’d like to organize this class having this in mind:

- What and how to learn?
- What will be on the slides?
- How to participate?
- Give me feedback!
- How will participation affect your grade?
- Teamwork
- Let me know (at a later point) what you are interested in

Experiment: Prepare something at home

Your tasks

- Look up what the ‘Horner scheme’ is
  - Wikipedia entry (English Wikipedia site) is very good
  - To be specific, Horner’s notation: ((((z+…)×z+…)×z …
- Describe the ‘Horner notation’
  - Three to four sentences
  - General context (Note: you don’t have to explain what a polynomial is)
  - What is the ‘trick’?
  - Why would you use it when writing code?
- ‘Present’ in class next week
  - Nothing dramatic, I’ll explain
  - Write the 3 sentences down, if that helps you

Let’s make our computer do something

```
a = b + c
```

Let’s make our computer do something

$$a = b + c$$

What needs to happen so that the computer can perform the calulation?

Where is the data coming from?

Where is the data going?

What part of the hardware performs the operation?

Let’s make our computer do something

$$a = b + c$$

What needs to happen so that the CPU can calculate?

Where is the data coming from?	Memory
Where is the data going?	Memory
What part performs the operation?	Floating Point Unit (FPU) in the CPU (Central Processing Unit)

Let’s make our computer do something

$$a = b + c$$

What needs to happen so that the CPU can calculate?

How long does it take?

(What units to use?)

Where is the data coming from?	Memory
Where is the data going?	Memory
What part performs the operation?	FPU in the CPU

Let’s make our computer do something

$$a = b + c$$

What needs to happen so that the CPU can calculate?		How long does it take?
Where is the data coming from?	Memory	A very long time
Where is the data going?	Memory	A very long time
What part performs the operation?	FPU in the CPU	A very short time

Let's make our computer do something

$$a = b + c$$

What needs to happen so that the CPU can calculate?		How long does it take?
Where is the data coming from?	Memory	300 cycles
Where is the data going?	Memory	300 cycles
What part performs the operation?	FPU in the CPU	3 cycles

What a bummer! Actual work takes 0.5% of the total time
We may have built a very ineffective computer

Does this help us?

$$a(i) = b(i) + c(i)$$

Does this help us?

$$a(i) = b(i) + c(i)$$

Hint: where would you likely find such a statement?

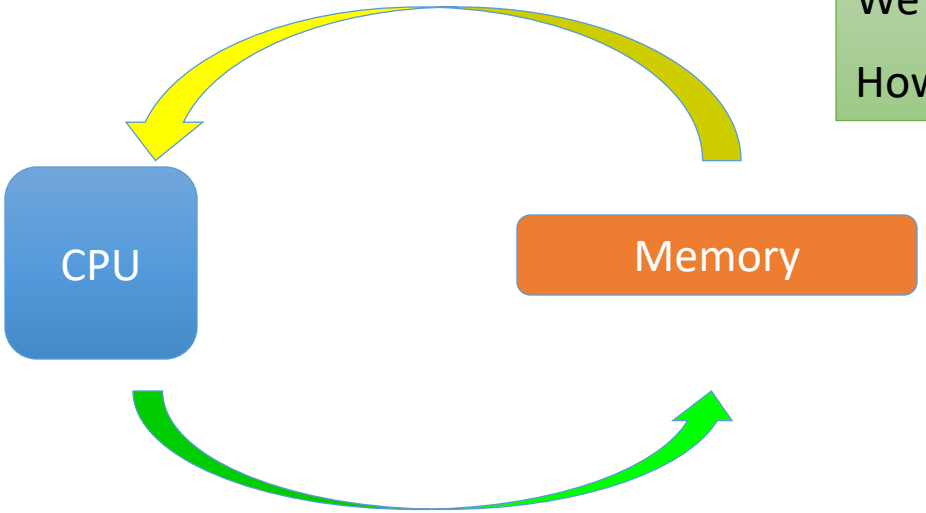
Does this help us?

$$a(i) = b(i) + c(i)$$

```
loop with index i
  a(i) = b(i) + c(i)
end loop
```

Data

a(i) = b(i) + c(i)



We have a lot of data to process

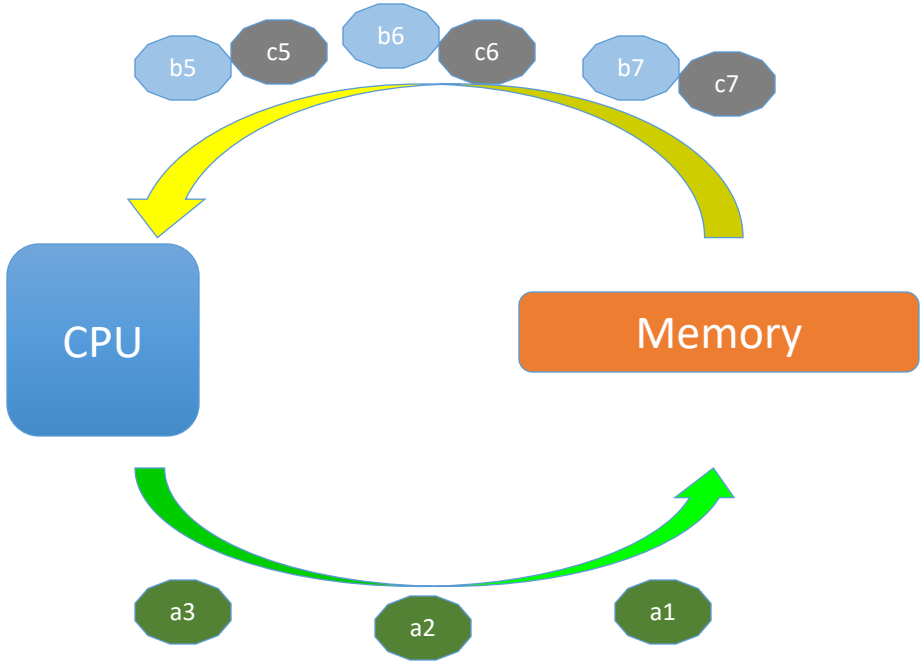
How can that help to **‘getting more done’**?

‘getting more done’

We are mostly interested in floating point operations (flops) that move the calculation closer to the solution

Data Streams

a(i) = b(i) + c(i)



Some data is 'en route'

b and c: from memory to CPU

a: from CPU to memory

Some data is being processed

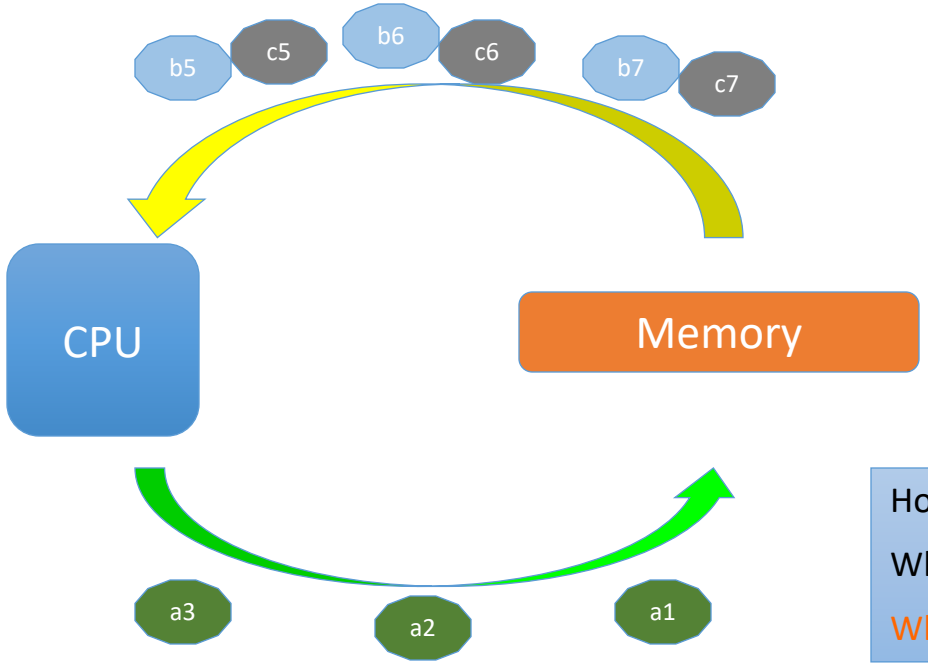
a(i) = b(i) + c(i)

How much data has to be 'en route'?

What are the main factors?

Data Streams

a(i) = b(i) + c(i)



Some data is 'en route'

b and c: from memory to CPU

a: from CPU to memory

Some data is being processed

a(i) = b(i) + c(i)

How much data has to be 'en route'?

What is one of the main factors?

What if I had drawn CPU and Memory further apart?