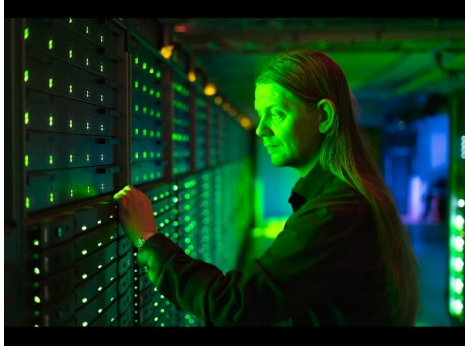
NTNU – Trondheim
Norwegian University of
Science and Technology

TDT4200 Parallel Computing overture

Jan.Christian.Meyer@ntnu.no



Hello, world

- I am Jan.Christian.Meyer@ntnu.no
- Associate professor within parallel computing at IDI/COMP
- I also moonlight for VitDat at NTNU-IT and Uninett Sigma2
- Wearing various roles and hats, I have commuted between these two chairs since 2003

- I have a particular affinity for *performance modeling*
  - We will only do a very little bit of that in this class
  - I can show you how to write an M.Sc. thesis about it, though

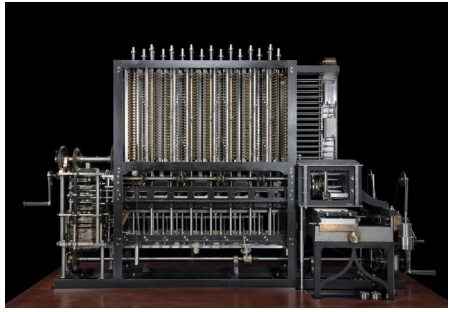
Today's topic

- The lecture is meant to put our subject into context
- I will also make the case that it is important
- I will talk a little bit about
  - How it got to be the way it is
  - What we use it for today
  - Why it will remain important in the future

Tabulating the skies and seas: HPC in the 1700s

- Parallel computing is older than digital computers
  - **1758**: *Alexis-Claude Clairaut, Joseph Jerome Lalande* and *Reine Lepaute* all sit together for 5 months and approximate the orbit of Halley's comet
  - **1765**: *Neville Maskelyne* begins calculating nautical almanacs for 1767 using himself and 5 employees
    - 4 computers work in pairs, alternating noons and midnights
    - 1 compares and corrects
  - **1793**: *Gaspard de Prony* initiates a 6-year effort to produce 19 volumes of trigonometry tables, using 96 untrained workers

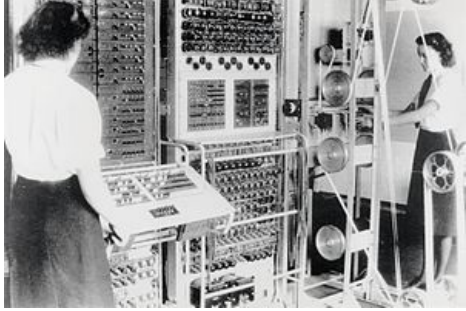
Machinery of change: HPC in the 1800s



- Mechanical devices revolutionized society
  - Steam power for transport, mills, and mining
  - Textile industry develops programmable looms
  - Entertainment industry develops programmable pianos
  - **1822**: *Charles Babbage* announces Difference Engine #2 to the Royal Astronomical Society, its design is completed (but not yet implemented) in 1847
  - **1842**: *Augusta Ada King* writes about the translation of calculations into machine states for its successor (the *Analytic Engine*), and inadvertently invents programming

What we discovered in the era of human computers

- Overclocking
  - When 1766 showed that the 1767 almanac would be late, Neville accelerated his computation by paying the computers £70 extra to finish on time
- Programmability & library functions
  - Gaspard had to prepare idiot-proof instructions and tables for his army of inexpensive non-mathematicians to get the right answer
- Fault tolerance
  - Difference Engine #2 weighs 4 tons and has 25.000 moving parts, some of them inevitably break down quite often
  - The Analytic Engine was never built, it would have been the size of a steam locomotive and broken down all the time



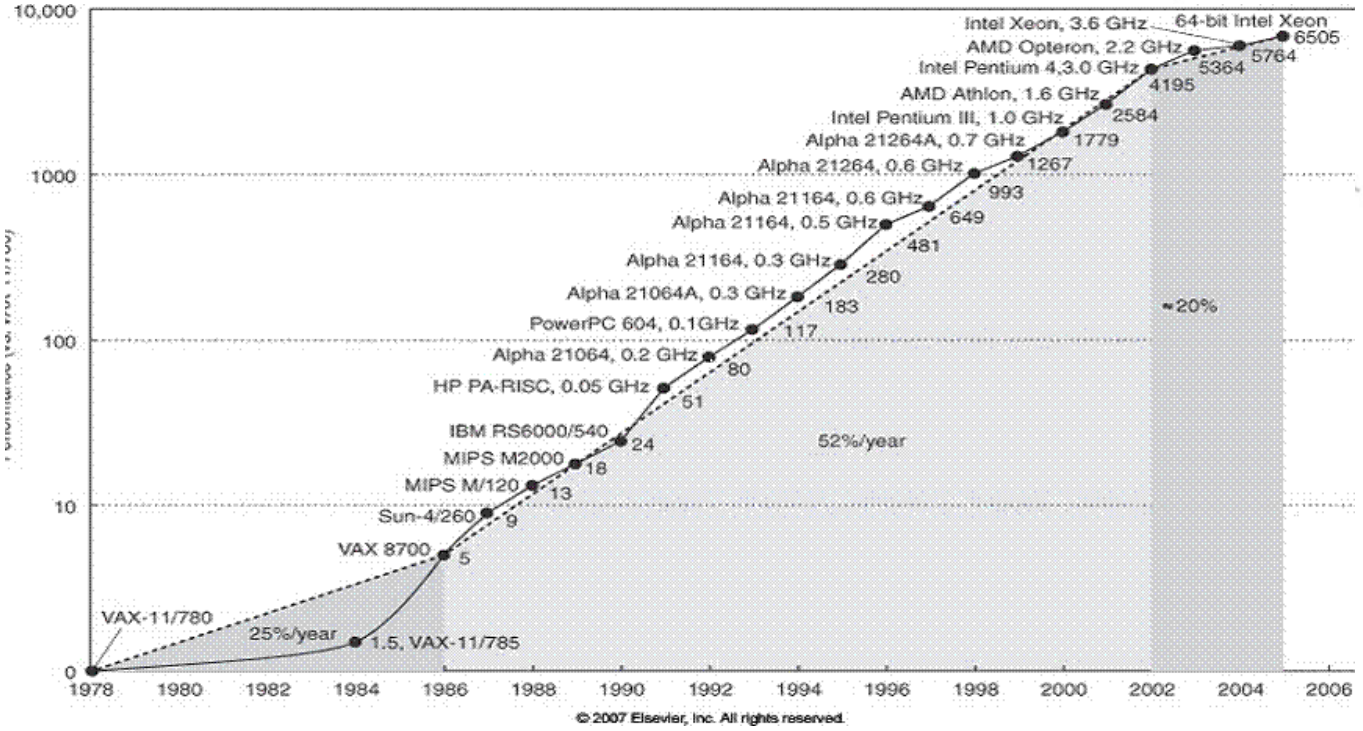
Computers at war: First half of the 1900s

- HPC is a zero-billion dollar business
  - Already in 1814, *Peter Barlow* lamented that the expenses of producing and publishing books of tables *"preclude every idea of adequate remuneration"*
- Wars make governments spend money without question
  - During WW1, map grids, ballistics and navigation tools were worked out by teams of college-educated women who had not been sent to operate the arsenal in the field
  - *Gertrude Blanch* continued afterwards, heading the Mathematical Tables Project
  - During WW2, PM Churchill ordered essentially unlimited resources for the Bletchley Park cryptanalysis group (with *Alan M. Turing* among them), in a memo with "Action This Day!" as its headline
  - *Konrad Zuse* invents the first buildable, programmable computer in 1941
  - *Stanley Phillips Frankel* designs a pipeline of mechanical IBM calculators to calculate the energy release at an atomic bomb explosion (*John von Neumann* was also on the Manhattan Project).

After the wars

- As the dust settled,
  - The late 1940s and 50s sees developments of the electro-mechanical computer which can finally be disclosed in peace time
  - Stability is still an issue, von Neumann proceeds to lecture on *"the synthesis of reliable organisms from unreliable components"*
  - **1959**: *Robert Noyce* first manages to integrate multiple switches on a single silicon substrate, massively improving reliability
  - **1965**: *Gordon Moore* publishes his projection that we can halve their size roughly every 18 months (thereby reducing the required supply voltage, and increasing the switching frequency for the same power budget)

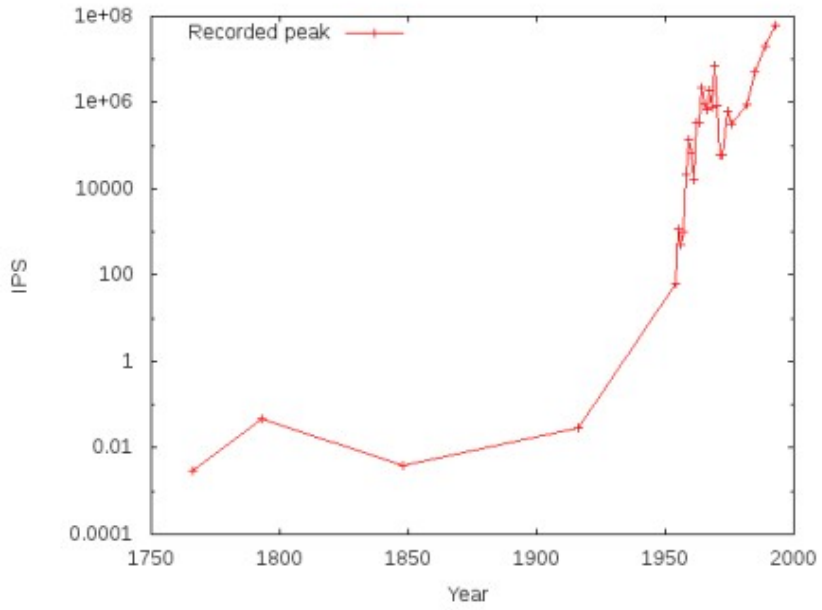
The clock race



You may have seen this figure, unceremoniously borrowed from *Computer Architecture* (4th ed.) by Hennessy & Patterson, 2007.

The vertical axis is performance, the horizontal is time. (Note the logarithmic vertical scale.)

The clock race in perspective



Logarithmic IPS

This is my figure, unceremoniously created by estimating computing rates from past centuries through dividing the size of the output by the number of years required to produce it.

(Note the still logarithmic vertical scale, and that the curve flattens out quite noticeably near 2000AD)

1959-2003 was a burst of computing power

- It was brought about by the aftermath of optimistic war investments, combined with a lucky break
- It has turned computing power into a driving force of the global economy (WTO, 2008), and made it a public commodity
- That was fun, but we can't expect it to happen again and again (and again)

Things we have seen so far

- Increased computing power goes hand-in-hand with radical changes in society
- Increased computing power requires long-term investments that bring no immediate benefit
- Parallel computing was the only way to increase computing power until 1959, and today it is the only way again

Why parallel computing?

- That's the title of chapter 1, it goes on to ask
  - Why do we care?
  - Aren't single-processor systems fast enough?
  - Why can't microprocessor manufacturers continue to develop faster faster single-processor systems?
  - Why can't we write programs that automatically convert serial programs into parallel programs?

My personal answers

- Why do we care?
  - Because increased computing power demonstrably changes everything for the human race
- Aren't single-processor systems fast enough?
  - There is no such thing as "fast enough", when we get more power to do things, we invent new things to do
- Why can't microprocessor manufacturers continue to develop faster faster single-processor systems?
  - Because they literally combust before we reliably reach 10GHz
- Why can't we write programs that automatically convert serial programs into parallel programs?
  - Because the converter can't tell whether its output will run any faster than the original, it only has functional requirements to reason with, and speed is a non-functional requirement

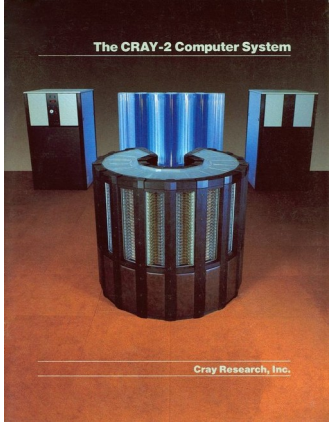
The book's answers

- Why do we care? Aren't single-processor systems fast enough?
  - We have *grand-challenge problems* that can use arbitrary amounts of processing power: climate modeling, medicine, energy tech, ...
- Why can't microprocessor manufacturers continue to develop faster faster single-processor systems?
  - They self-immolate at high clock speeds
- Why can't we write programs that automatically convert serial programs into parallel programs?
  - The best parallel implementations often change the applied algorithm

Where do we go from here?

- Last time we were approaching a barrier to computing power, Moore's law came to the rescue
- In the meantime, computers have gone from being gigantic investments to household items
- Scientific computing is no longer the driving force of the industry
  - Good thing too, there's no money in it

The Wal-Mart effect



- Back in 1985, the Cray-2 was the hot thing to have
  - A grand total of 27 such machines were built
- Back in 2022, more than 225 million people bought iphones
  - If there were only 27 customers, these devices would be as expensive as the Cray-2 was
- Scientific high-performance computing (HPC) gets cheap hardware as long as the hardware has another market as well

Mutual benefits

- Hardware companies invest heavily in HPC installations even if they don't earn a lot of money there
  - To them, it's a kind of laboratory where they can test the performance of their designs
- The techniques and designs they discover come back in the form of consumer electronics 10-20 years later
  - Cray-2 was a vector processor in the mid 1980s, near the end of the 1990s, Intel's Pentium 2 came with built-in vector registers
  - A 512-core shared memory system filled a room in 1998, in 2012 you could get a similar device as a PCI expansion card
  - NTNU installed a computing cluster with 64 processors in 2003, in 2023 you can buy 64-core desktop machines
  - I could go on...

What's in this class

- We're going to write some small simulators for physics problems
- This is just because it gives us a calculation that can always benefit from bigger machinery
- The main focus is on *programming models:*
  - Message Passing Interface (MPI)
  - Posix threads (pthreads)
  - Open Multi-Processing (OpenMP)
  - Compute Unified Device Architecture (CUDA)

Why take it?

- It’s super fun. :)
- It’s also a kind of crystal ball:
  - Our tools are presently in use to push the limits of state-of-the-art supercomputers
  - The performance issues we address there will probably become relevant in consumer equipment soon enough

Welcome to the vanguard

- Hardware went parallel around the turn of the century, and continues to pile on the resources
- Software hasn't really kept up, making code run better with extra processors is an "advanced topic"

  (it's not really that hard, though, you'll see)
- *Figuring out how to present parallel computing for the masses is very much a work-in-progress*
  - HPC proposes different programming models all the time
- If you pick up the ones we've got so far, you can join in!