TDT4200 Parallel Computing

Bart van Blokland

Mandatory boring bits:

Announcements

Course Overview

Survey:

33 / 88 can’t come on Mondays 12 - 14

14 / 88 can’t come on Wednesdays 12 - 13

30 / 88 can’t come on Thursdays 12 - 14

Weekly lab sessions:

Wednesdays 12:00 – 14:00 +

ITS-015 “Tulipan”

(“Krokus” on ntnu.no/kart)

(might be starting at 13:00 if we use the wednesday lecture period)

Crash Course C++

Today and Thursday (recitation)

Weekly lecture:

Mondays 12:15 – 14:00 in F2

Doesn’t look like this can be changed :(

Final grade:

5 Assignments (25%)

Final Exam (75%)

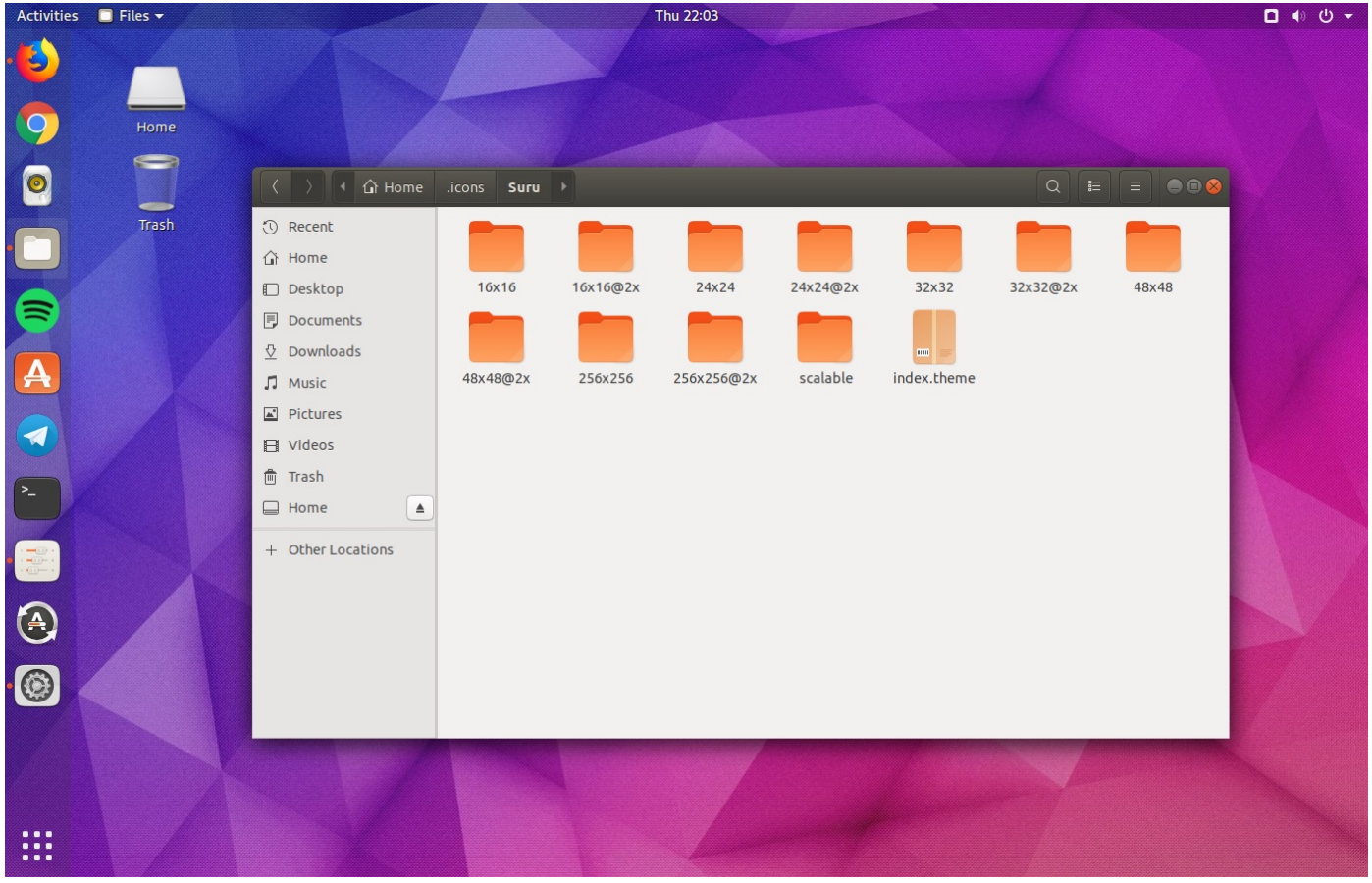
Assignment Schedule (Tentative)

Deadline	**Topic**
N/A	Assignment 0 (Optional): C Intro
14.09	Assignment 1: Optimisation and Profiling (out on Friday)
28.09	Assignment 2: MPI
12.10	Assignment 3: OpenMP & Pthreads
26.10	Assignment 4: CUDA Basics
09.11	Assignment 5: More on CUDA

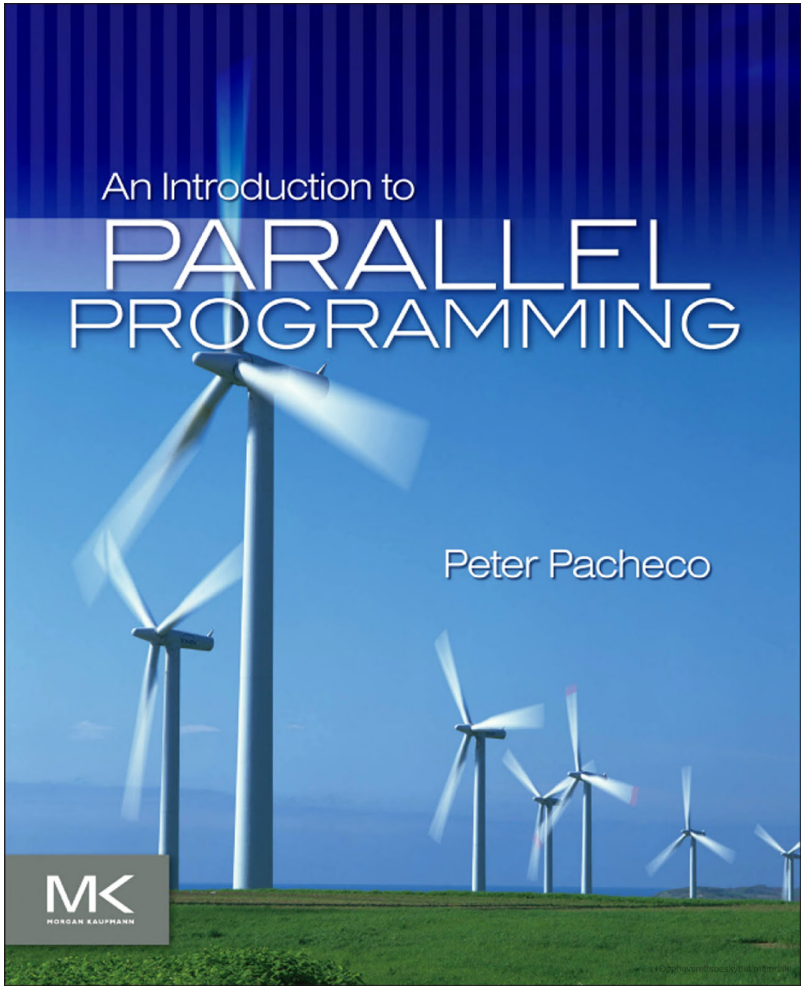
Lecture Schedule (Tentative)

Date	Topic
27.08	Introduction
03.09	Measuring Performance
10.09	Optimising Single Threaded Code
17.09	(Guest Lecture by Jan Christian Meyer) Introduction to MPI
24.09	(Guest Lecture by Anne Elster) More on MPI
01.10	POSIX threads
08.10	OpenMP
15.10	Introduction to CUDA
22.10	CUDA
29.10	CUDA: Warp cooperation
05.11	CUDA: Profiling tools
12.11	OpenCL
19.11	Exam Q&A?

Linux installation is highly recommended!



Book


An Introduction to
PARALLEL
PROGRAMMING
Peter Pacheco
MK
MORGAN KAUFMANN

</mandatory>

Parallel Programming:

Serial Killer?

What is it (about)?

Why do we care?

15 44 5

21	28	12
4	15	28
27	9	2
28	1	4
8	12	19
15	20	8
14		24
21		

More data → More parallelism

Parallel Computing:

+ Faster

Parallel Computing:

+ Faster

Parallel Computing:

+ Faster

- Communication

Parallel Computing:

+ Faster

- Communication

- Synchronisation

Parallel Computing:

+ Faster

- Communication

- Synchronisation

- Load balancing

MPI

OpenMP

Pthreads

CUDA

OpenCL

Performance?

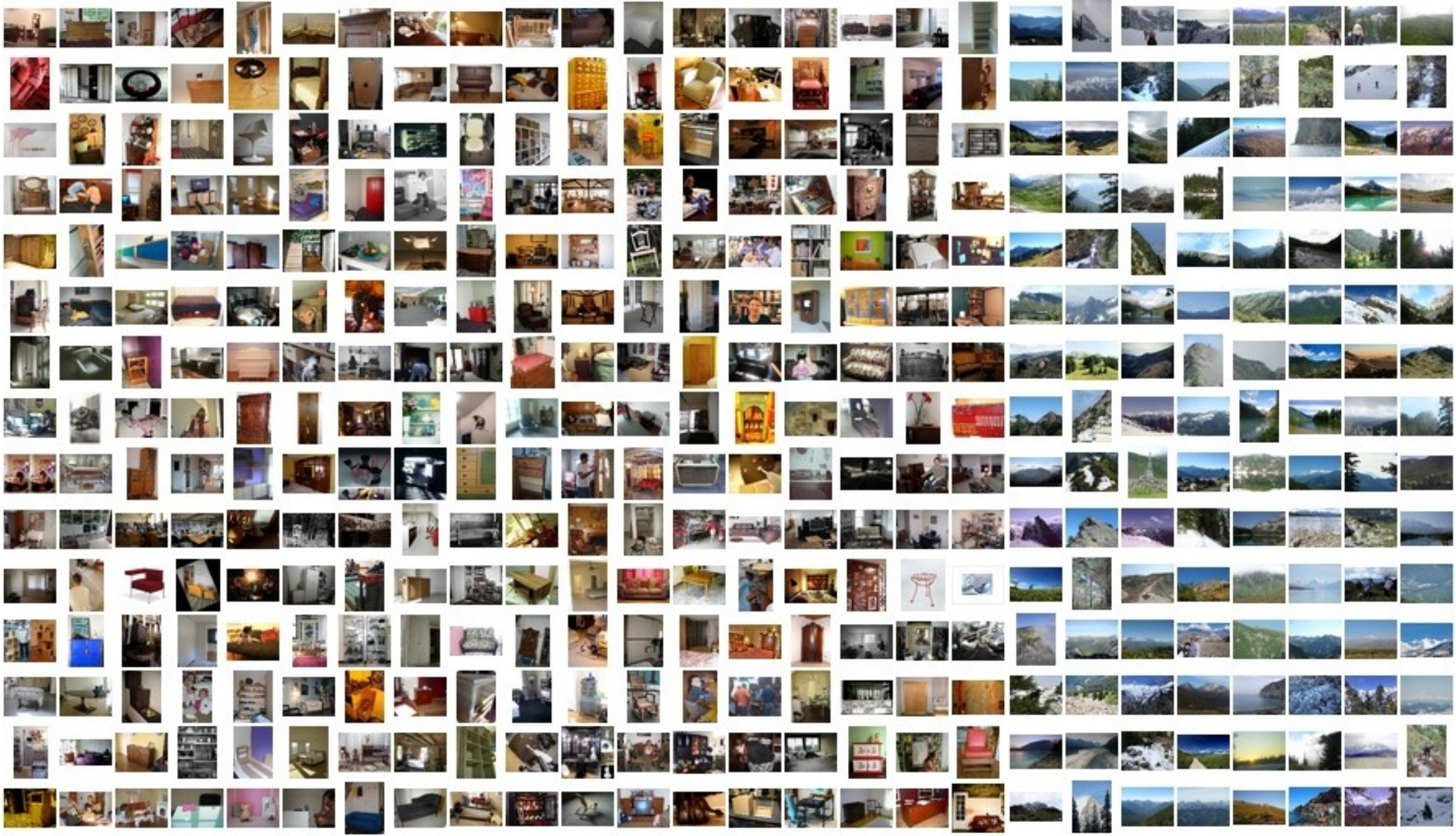
Is that not the compiler’s job?

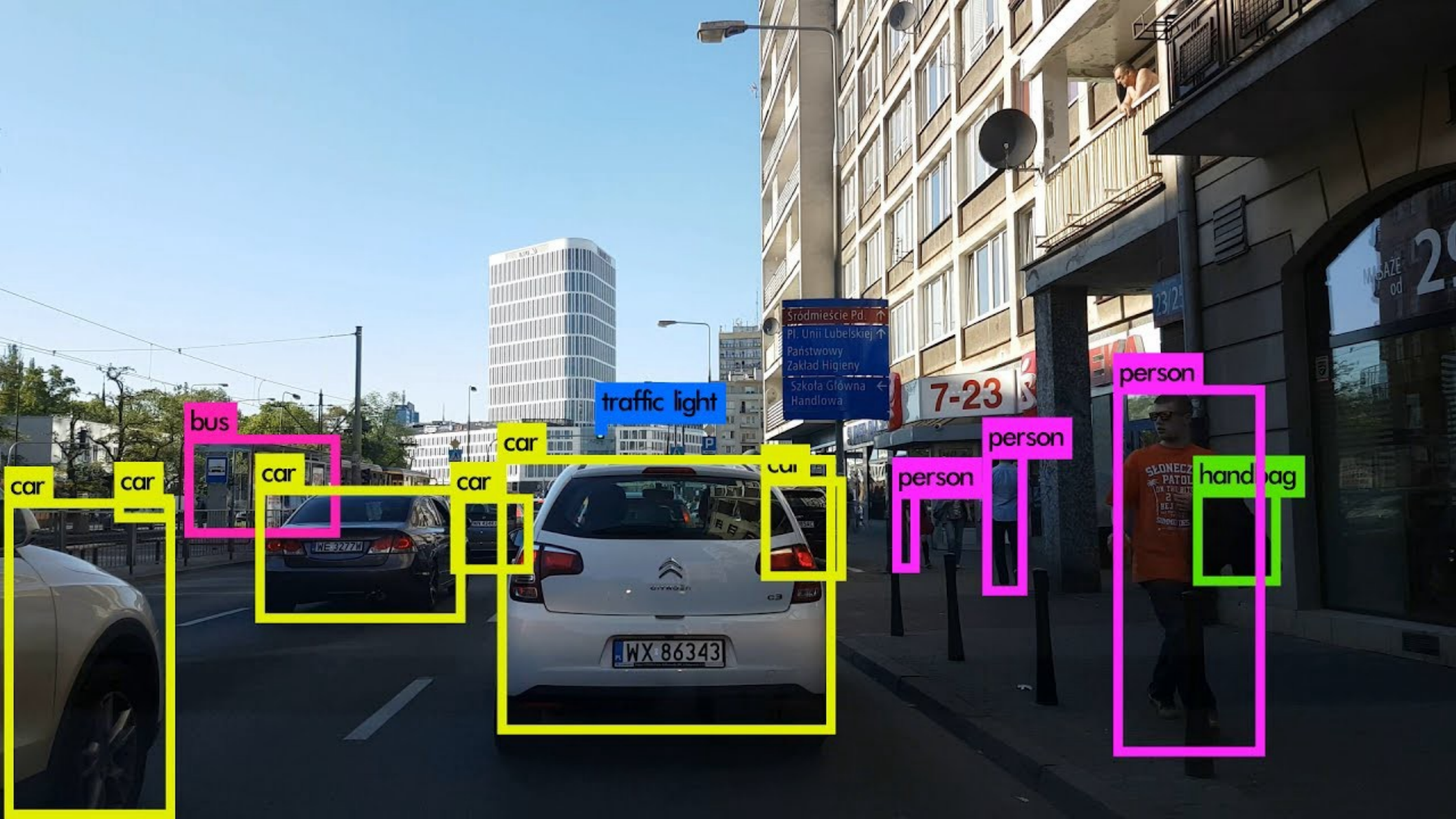
Performance?

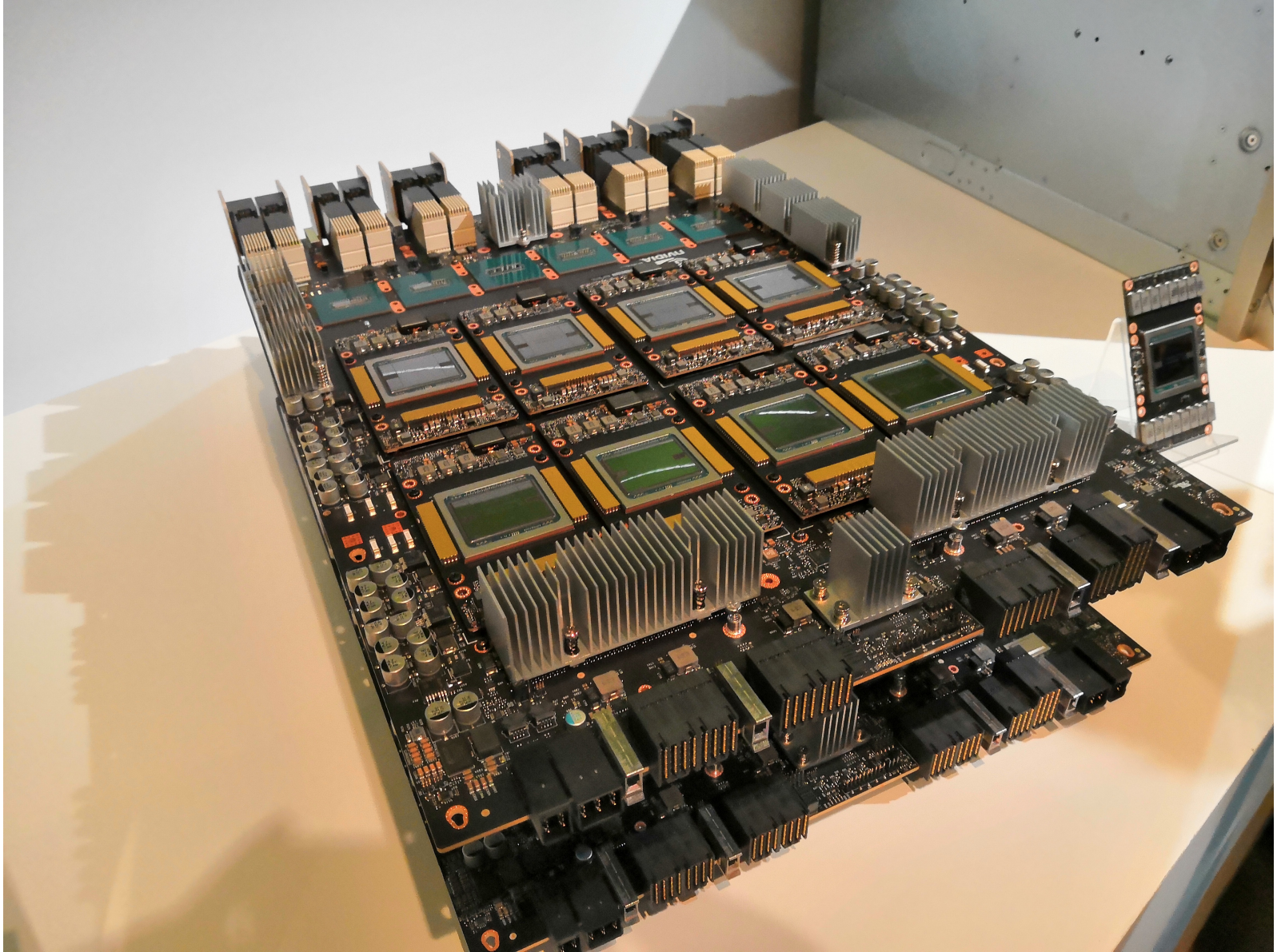
Is that not the compiler’s job?

Course plug: TDT4205

What is it used for?

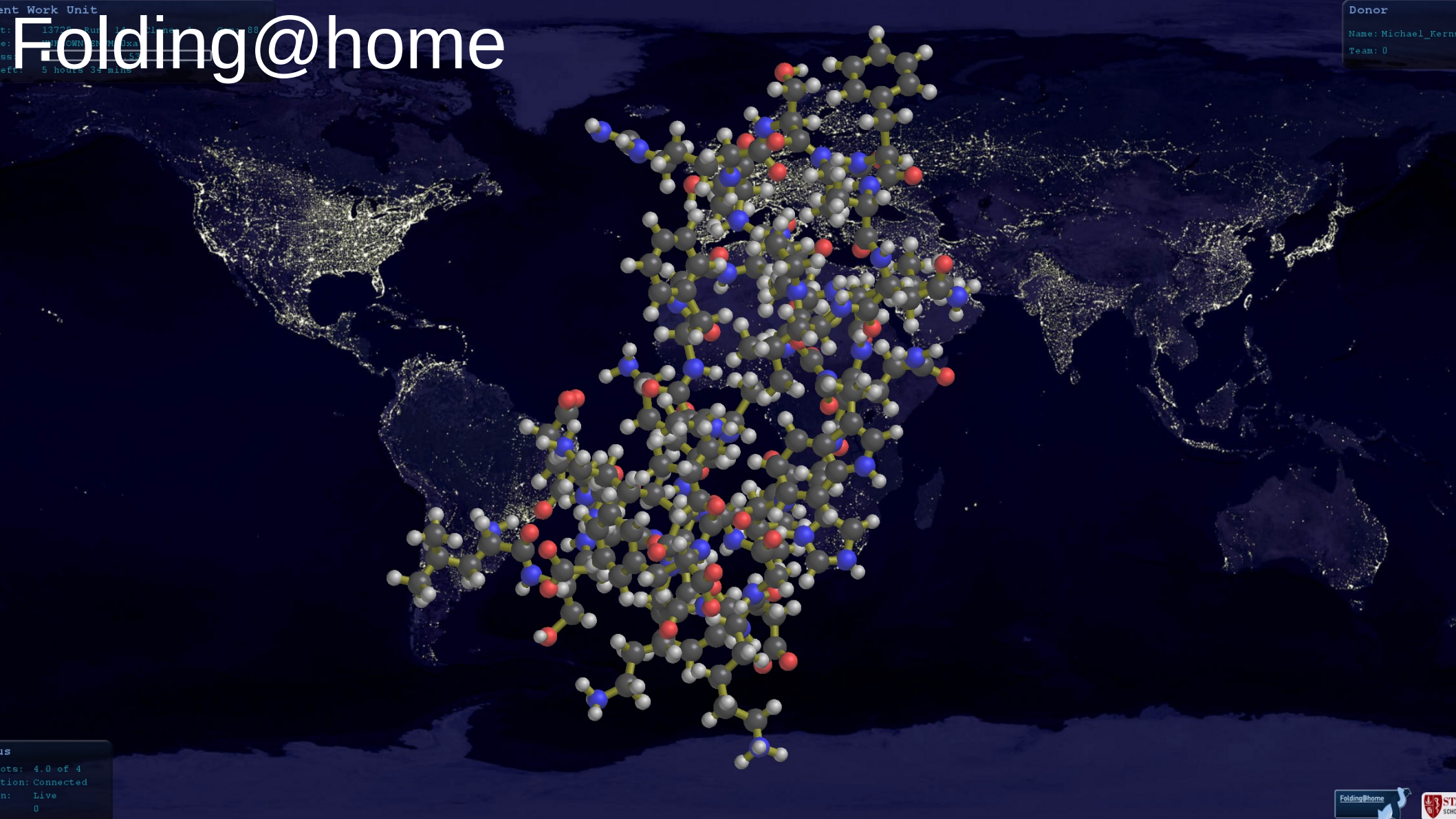



car
car
bus
car
car
car
traffic light
car
person
person
person
handbag
Śródmieście Pd.
Pl. Unii Lubelskiej
Państwowy
Zakład Higieny
Szkoła Główna
Handlowa
7-23
WX 86343




Google
google.com/datacenters

Folding@home




sgi

Another good example of HPC:

Computer Grahics

11 300 000 000 000 FLOPS

DOOM

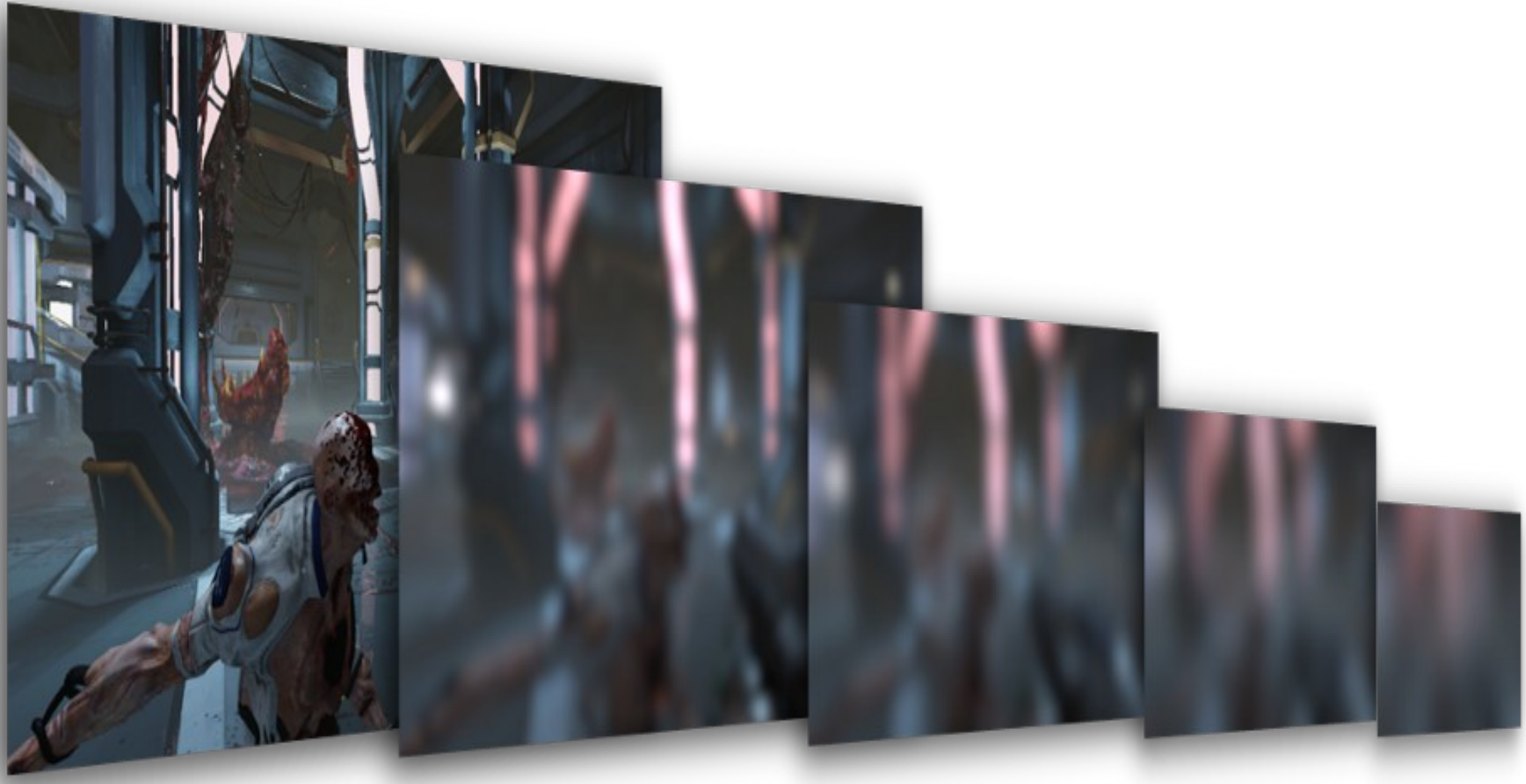

90
9


X4


UNION

[Particle Simulation]


UNION




90
9

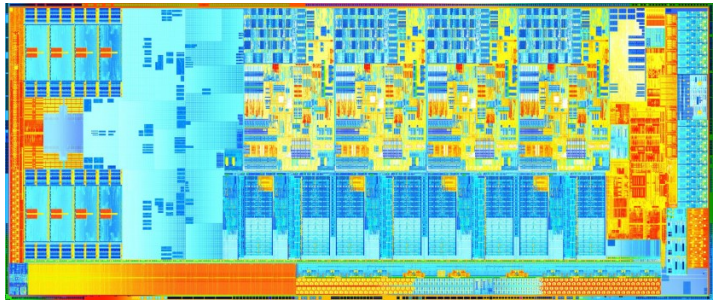
GTX 1080 Ti:

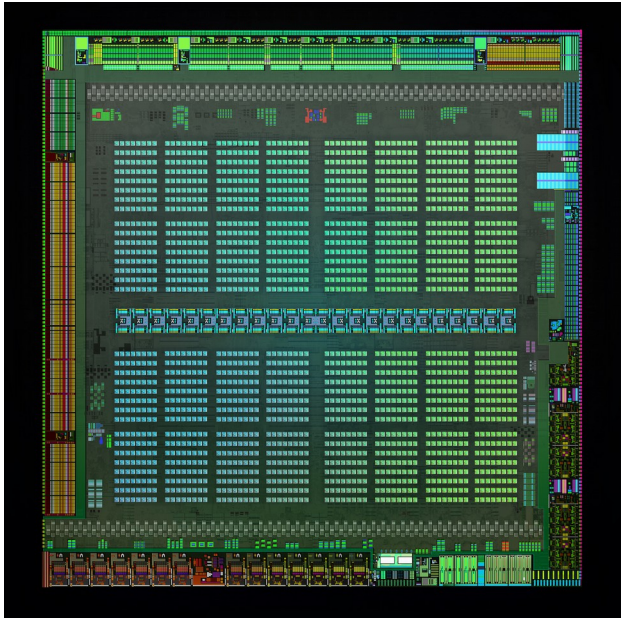
4K Resolution (3840 x 2160)

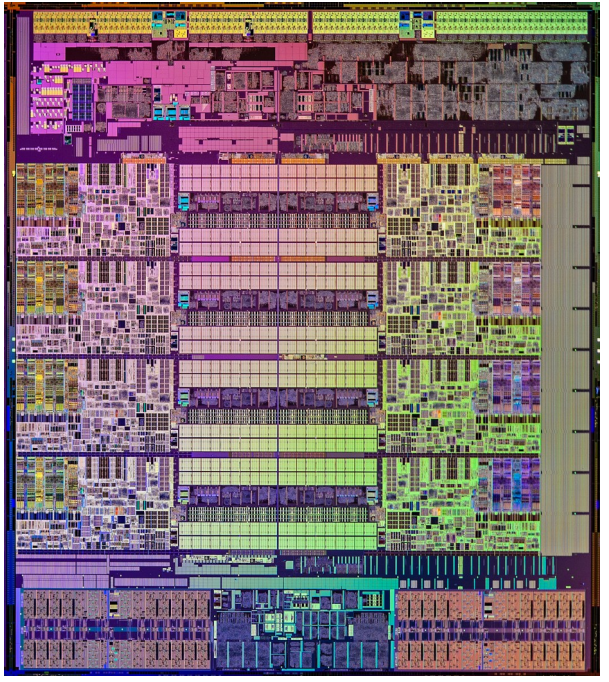
82 frames per second (average)

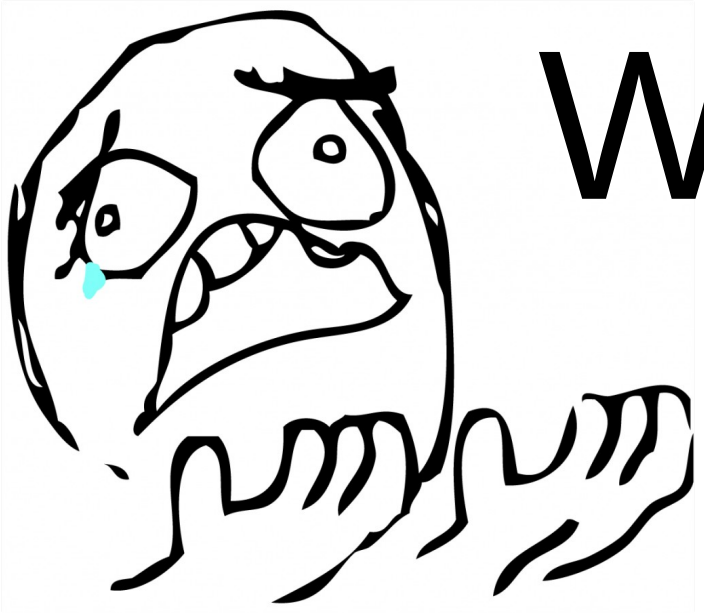

JOIN ME
We have cores
and cookies

Look around you!

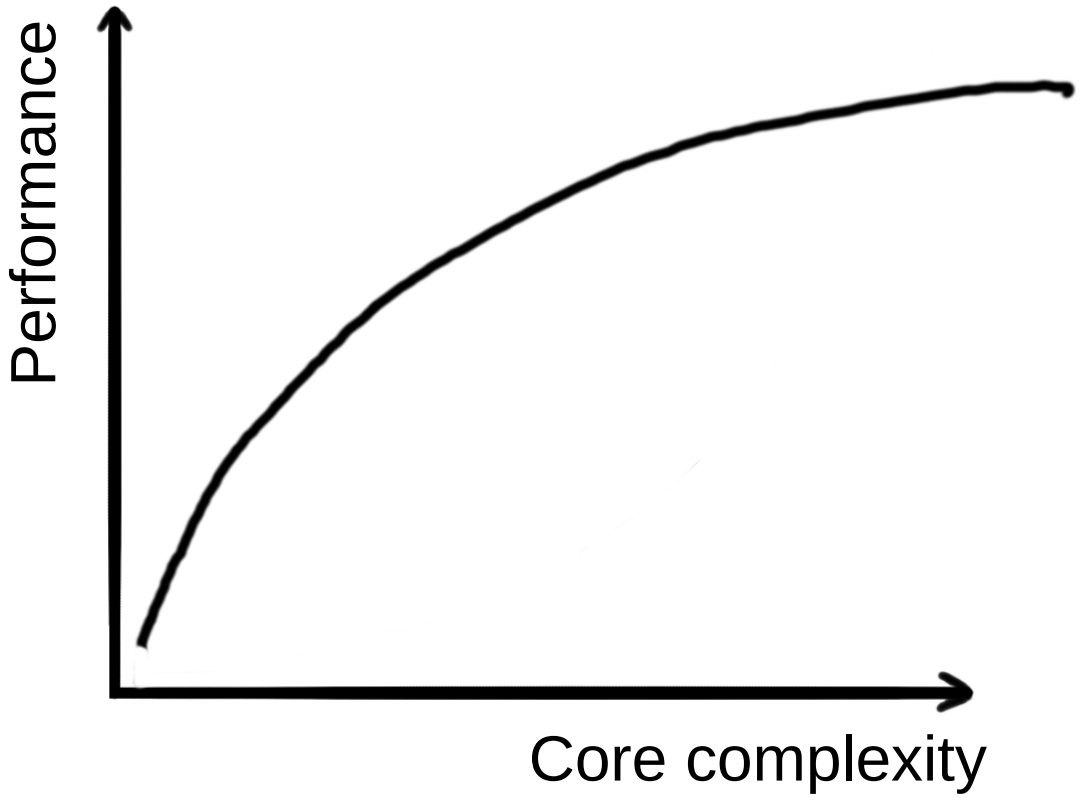


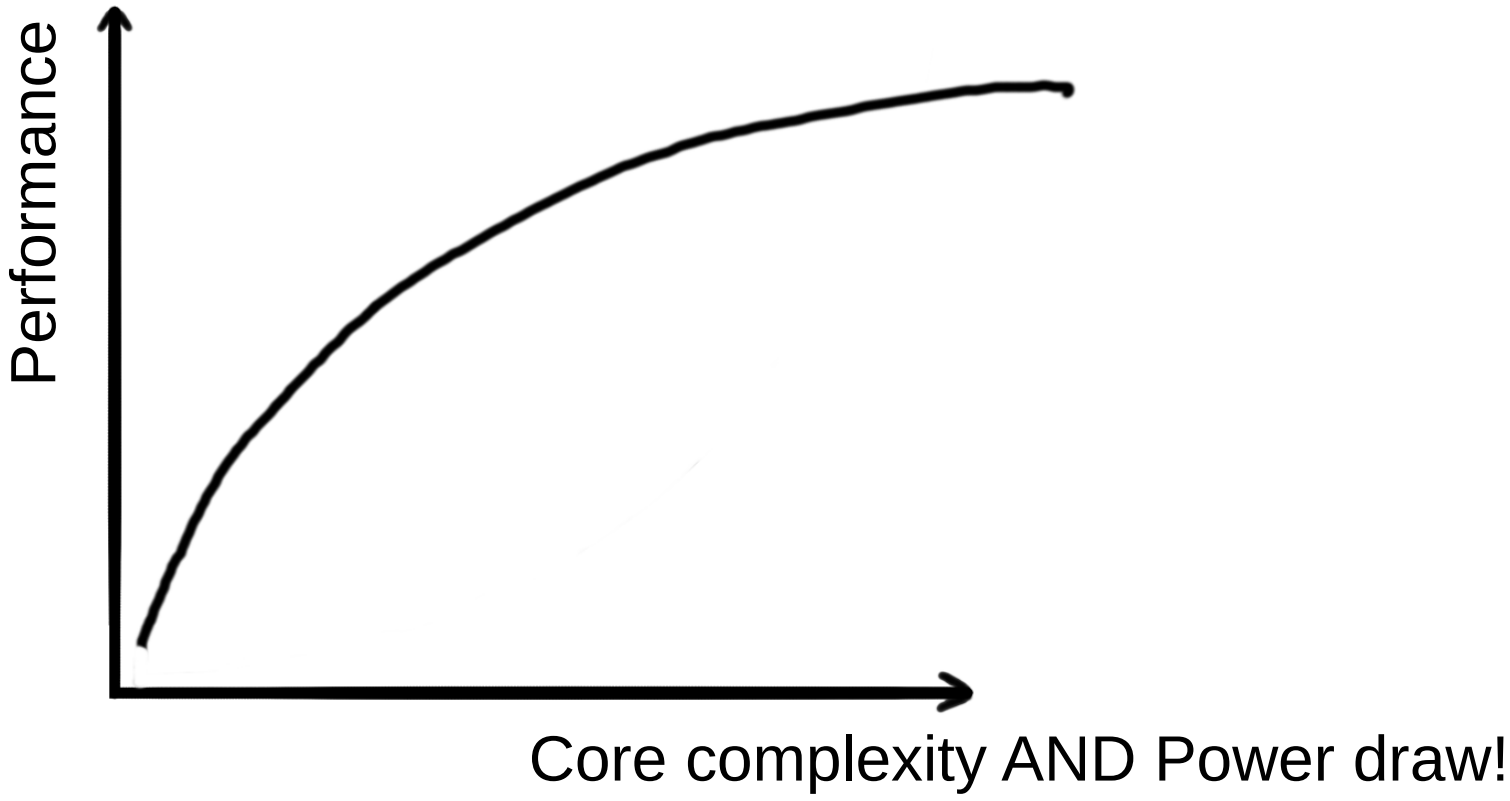




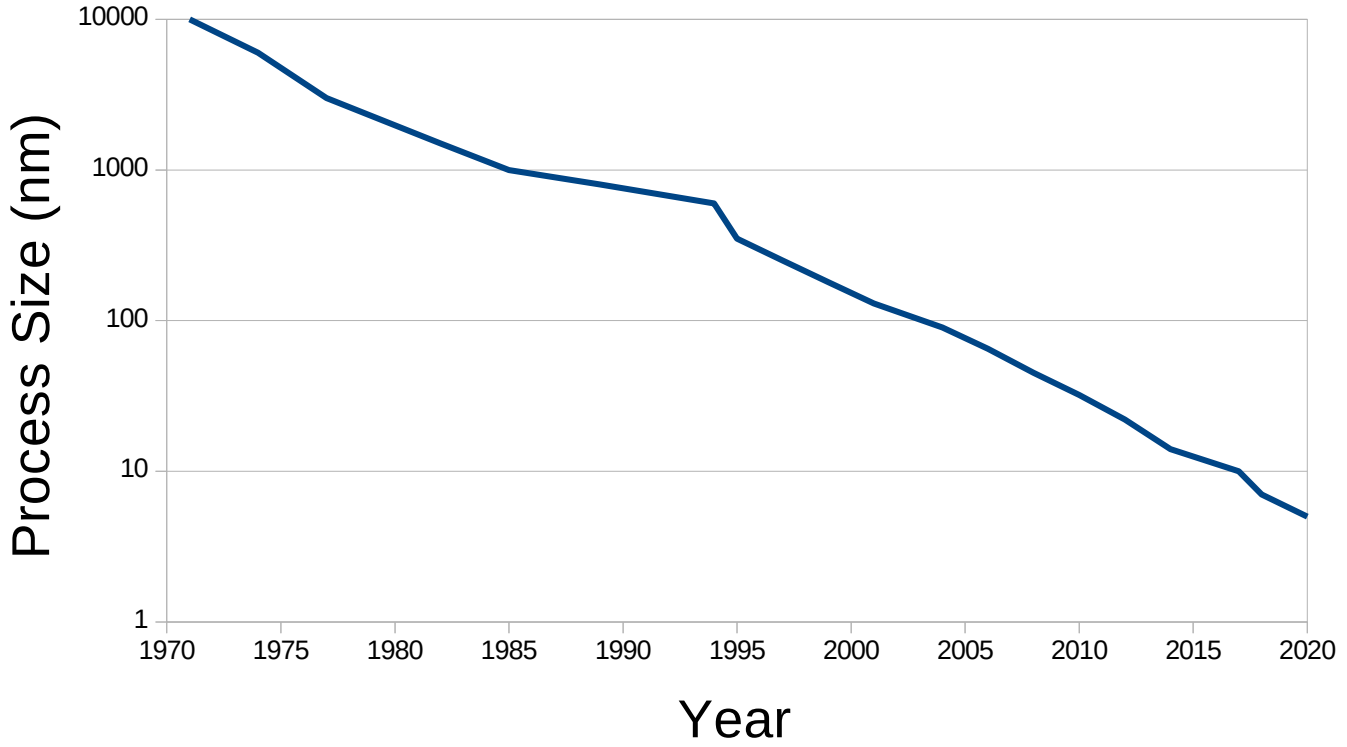


Why?

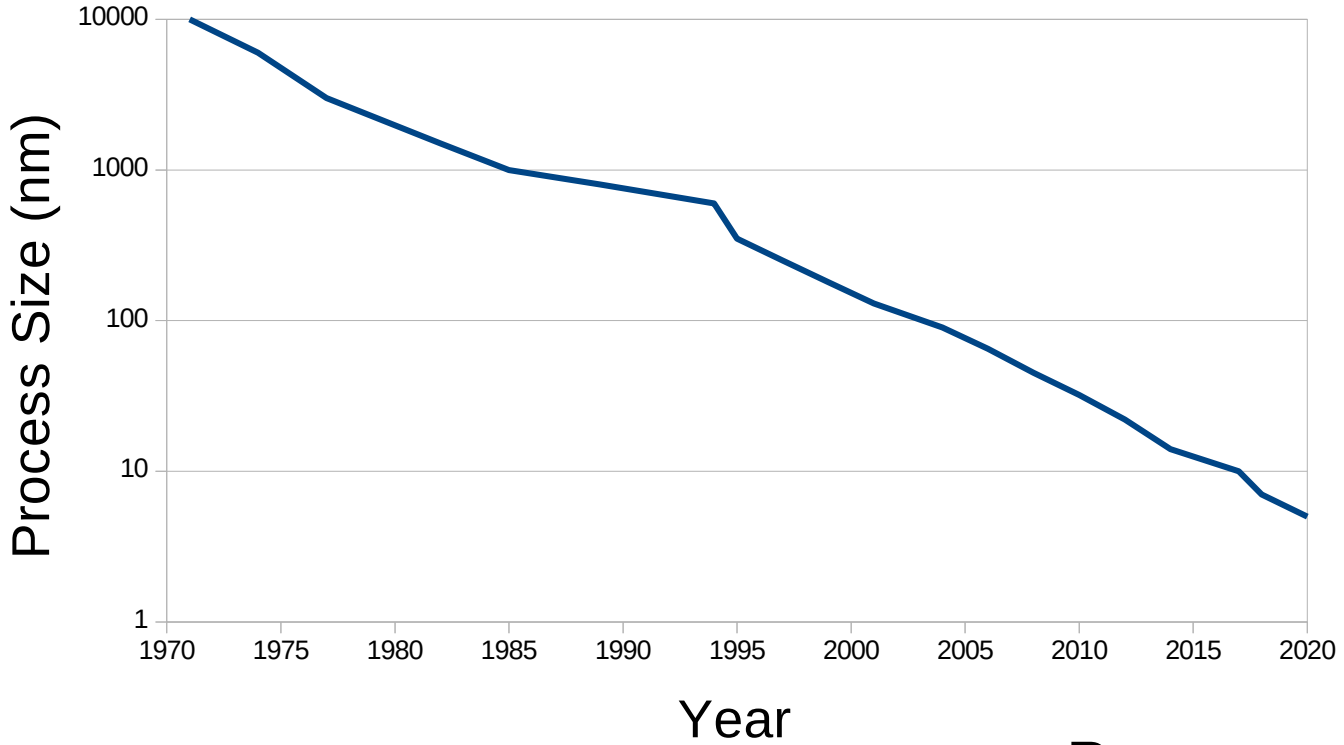

Performance
Core complexity


Performance
Core complexity AND Power draw!

Die Shrink


Process Size (nm)
10000
1000
100
10
1
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
Year

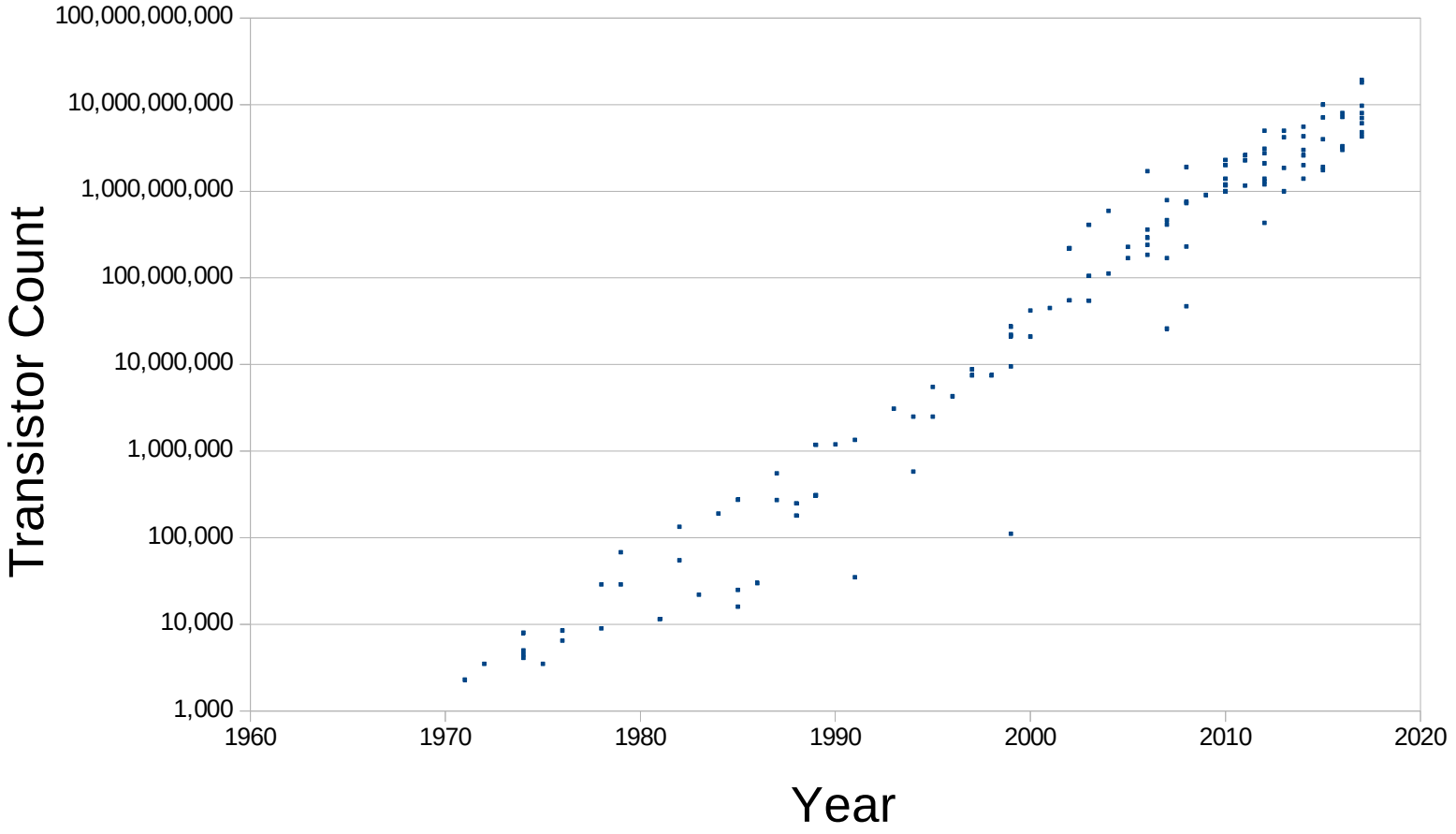
Die Shrink



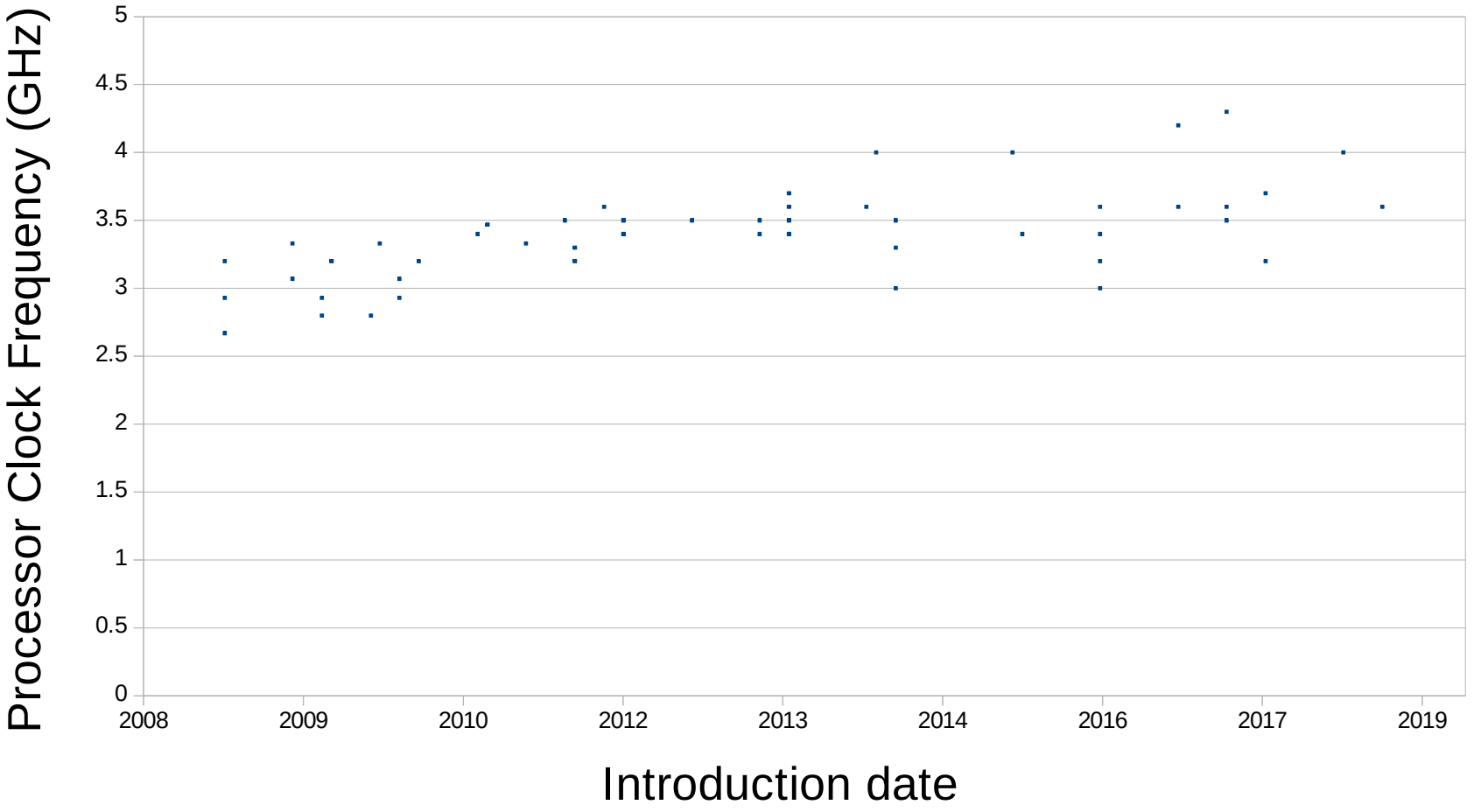
Silicon Atom radius: 0.2 nm

Process size: distance between transistors

Moore’s Law


Transistor Count
100,000,000,000
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
1960
1970
1980
1990
2000
2010
2020
Year

Reaching the limits of single core performance



Possible solutions:

Divide processor into multiple cores

Use multiple processors

Possible solutions:

Divide processor into multiple cores

Use multiple processors

→ Both solutions are parallel computing!

But wait, there is more..

Level	Time
CPU Cycle (4.3 GHz)	0.23 ns
L1 Cache Access	0.93 ns
L2 Cache Access	3.26 ns
L3 Cache Access	18.4 ns
RAM Access	68.4 ns
Intel Optane Access	14.9 µs
NVMe SSD Access	338 µs
SATA SSD Access	2.0 ms
Mechanical HDD Access	7.6 ms
Ping to uio.no	14.4 ms
Ping to ucla.edu	189.8 ms
Windows installs update	40 m

Level	Time	Time (Human)
CPU Cycle (4.3 GHz)	0.23 ns	1 s
L1 Cache Access	0.93 ns	4 s
L2 Cache Access	3.26 ns	14 s
L3 Cache Access	18.4 ns	79 s
RAM Access	68.4 ns	5 m
Intel Optane Access	14.9 µs	18 h
NVMe SSD Access	338 µs	17 days
SATA SSD Access	2.0 ms	3 months
Mechanical HDD Access	7.6 ms	1 year
Ping to uio.no	14.4 ms	2 years
Ping to ucla.edu	189.8 ms	26 years
Windows installs update	40 m	∞

Conclusions:

- Automated parallelisation is complicated

- Processor complexity requires parallel hardware

- Parallel hardware requires parallel software

So how fast *can* we go?

Good news: potentially infinitely fast

Bad news: depending on the problem

```
for(int i = 0; i < 9001; i++) {
}
```

Amdahl's Law

$$S_{latency}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

Gustafson's Law

$$S_{latency}(s) = (1 - p) + N\ p$$

Summary:

Parallel Computing is good

Overhead is bad

Power limits are a hot topic

Demonstration!

Crash Course C++

```c
int main(int argc, char** argv) {
  int wops = 1/0;
  return 0;
}
```

See you next week!

~~See you next week!~~

We’ll be using C++ in this course

.. It’s definitely not an easy language.

We’ll be using C++ in this course

.. It’s a complex language.