


# Advanced Systems Lab

Spring 2025, Lecture 1




Instructors: *Markus Püschel*  
TAs: *Tommaso Pegolotti, several more*

**ETH**  
Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich


Picture: [www.tapety-na-pulpit.org](http://www.tapety-na-pulpit.org)

1

**Minds open...**



**... Laptops closed**



*slide by Bertrand Meyer*

2

2

# Today

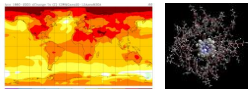
Motivation for this course

Organization of this course

3

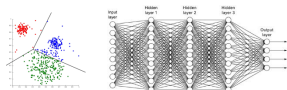
3

## Scientific Computing



Physics/biology simulations, ...

## Cloud Computing



Data analytics, machine learning, ...

## Consumer Computing



Audio/image/video processing, ...

## Embedded Computing



Signal processing, communication, control, ...

# Numerical Computing

Unlimited need for performance

Large set of applications, but ...

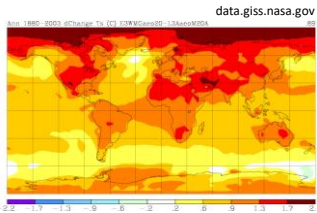
Relatively small set of critical components  
(10s to 100s)

- Matrix multiplication
- Discrete Fourier transform (DFT)
- Viterbi decoder
- Shortest path computation
- Stencils
- Solving linear systems
- ....

4

4

## Scientific Computing (Clusters/Supercomputers)



Climate modelling



Finance simulations



Molecular dynamics

### Other application areas:

- Fluid dynamics
- Chemistry
- Biology
- Medicine
- Geophysics

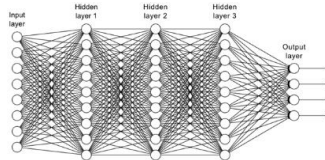
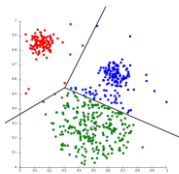
### Methods:

- Mostly linear algebra
- PDE solving
- Linear system solving
- Finite element methods
- Others

5

5

## Cloud Computing (Server Farms)



```

Provider = "DataAccess"
SelectSQL1 = "Select id, name, quantity, price,
QuerySQL1 = " group by id, name
Execute Query; Commit Transaction;
Form Navigation
If KeyAscii = 13 Then Execute Query
    Not Chr(KeyAscii) Like "*" And KeyAscii < 255
    
```

### Application areas:

- Data analytics
- Machine learning
- Database operations
- Others

### Methods:

- Linear algebra
- Convolutions
- Tensor operations
- Others

6

6

## Consumer Computing (Desktop, Phone, ...)



Photo/video processing



Audio decoding



Security



Image compression

### Methods:

- Linear algebra
- Transforms
- Filters
- Others

7

7

## Embedded/Edge Computing (Low-Power Processors)



Sensor networks



Cars



Robotics

### Applications:

- Signal processing
- Control
- Communication
- Inference
- Others

### Methods:

- Linear algebra
- Transforms
- Filters
- Coding
- Others

8

8

## Classes of Performance-Critical Functions

Transforms

Filters/correlation/convolution/stencils/interpolators

Dense linear algebra functions

Sparse linear algebra functions

Tensor operations

Coder/decoders

Graph algorithms

... *several others*

See also the 13 dwarfs/motifs in

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf>

9

9

## How Hard Is It to Get Fast Code?

Algorithms

Software

Compilers

Microarchitecture

"compute Fourier transform"



"fast Fourier transform"  
 $O(n \log(n))$  or  $4n \log(n) + 3n$



e.g., a C function



optimized executable



high runtime performance

*How well does this work?*

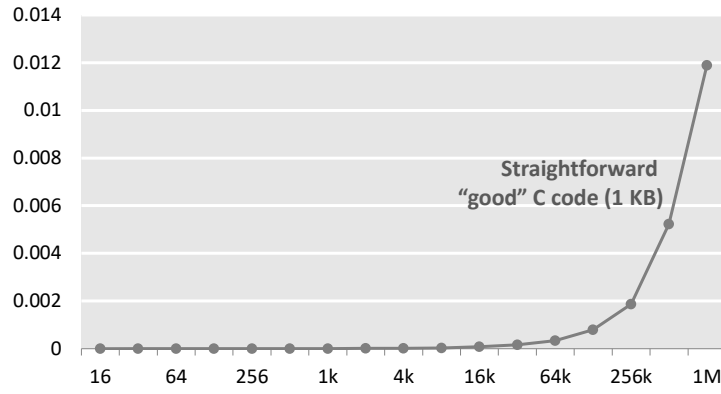
10

10

## The Problem: Example 1

DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)

Runtime [s]



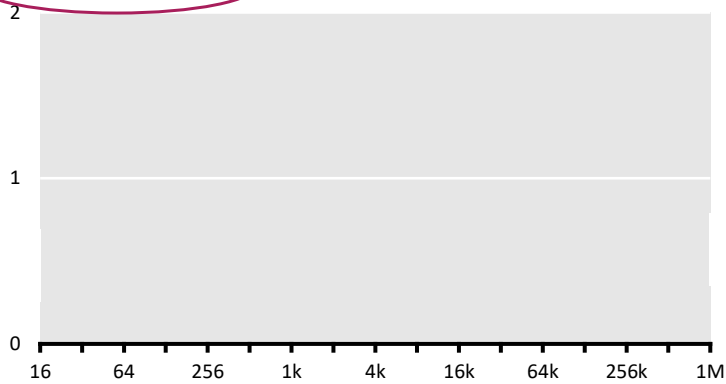
11

11

## The Problem: Example 1

DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)

Performance [Gflop/s]



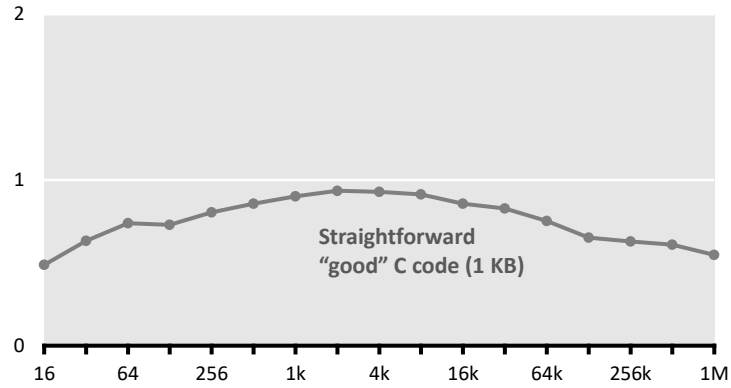
12

12

# The Problem: Example 1

DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)

Performance [Gflop/s]



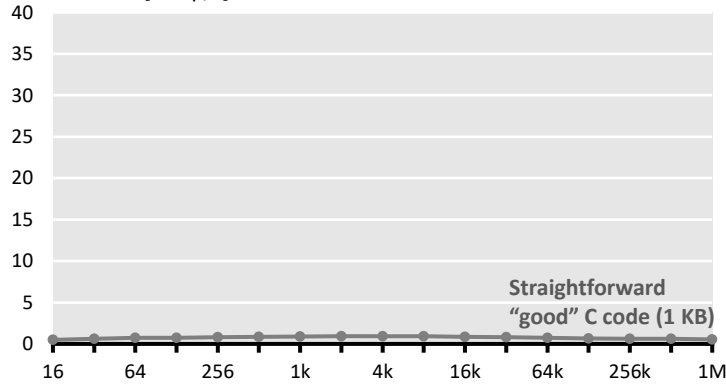
13

13

# The Problem: Example 1

DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)

Performance [Gflop/s]



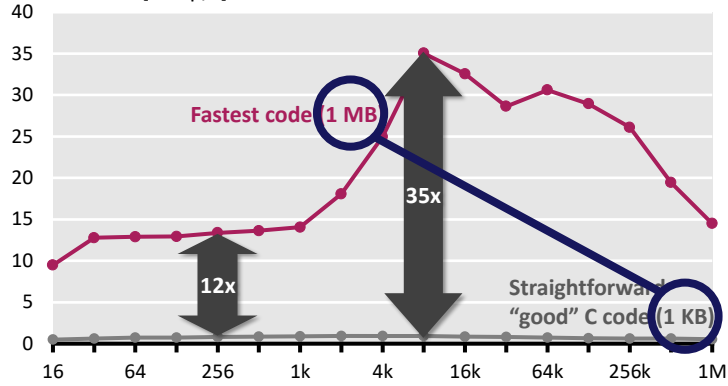
14

14

## The Problem: Example 1

DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)

Performance [Gflop/s]



Vendor compiler, best flags

Roughly same operations count

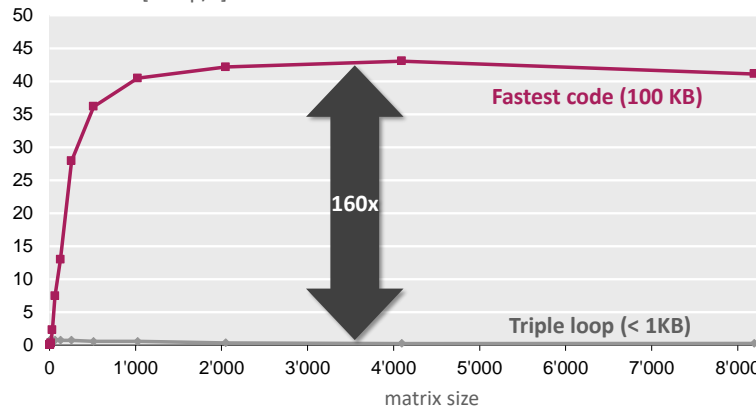
15

15

## The Problem: Example 2

Matrix-Matrix Multiplication (MMM) on 2 x Core 2 Duo 3 GHz

Performance [Gflop/s]



Vendor compiler, best flags

Exact same operations count ( $2n^3$ )

16

16



Model predictive control	Singular-value decomposition
Eigenvalues	Mean shift algorithm for segmentation
LU factorization	Stencil computations
Optimal binary search organization	Displacement based algorithms
Image color conversions	Motion estimation
Image geometry transformations	Multiresolution classifier
Enclosing ball of points	Kalman filter
Metropolis algorithm, Monte Carlo	Object detection
Seam carving	IIR filters
SURF feature detection	Arithmetic for large numbers
Submodular function optimization	Optimal binary search organization
Graph cuts, Edmond-Karps Algorithm	Software defined radio
Gaussian filter	Shortest path problem
Black Scholes option pricing	Feature set for biomedical imaging
Disparity map refinement	Biometrics identification

17

17

## “Theorem:”

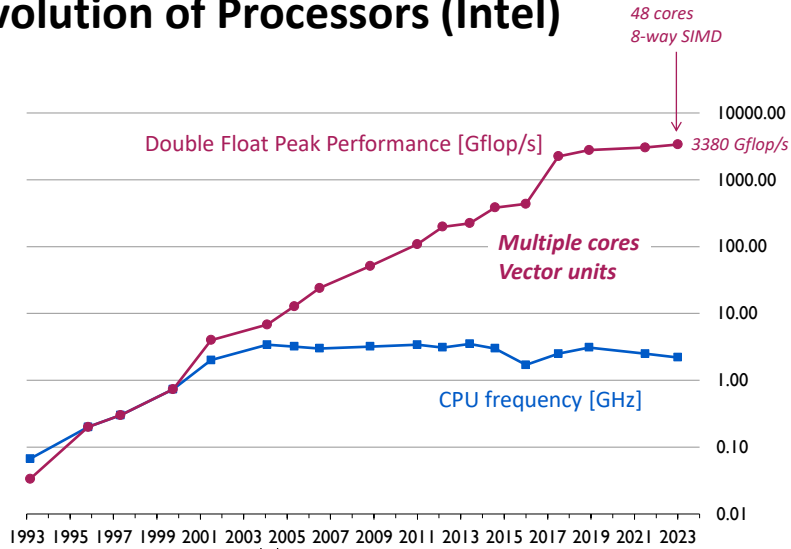
Let  $f$  be a mathematical function to be implemented on a state-of-the-art processor. Then

$$\frac{\text{Performance of optimal implementation of } f}{\text{Performance of straightforward implementation of } f} \approx 10-100$$

18

18

# Evolution of Processors (Intel)



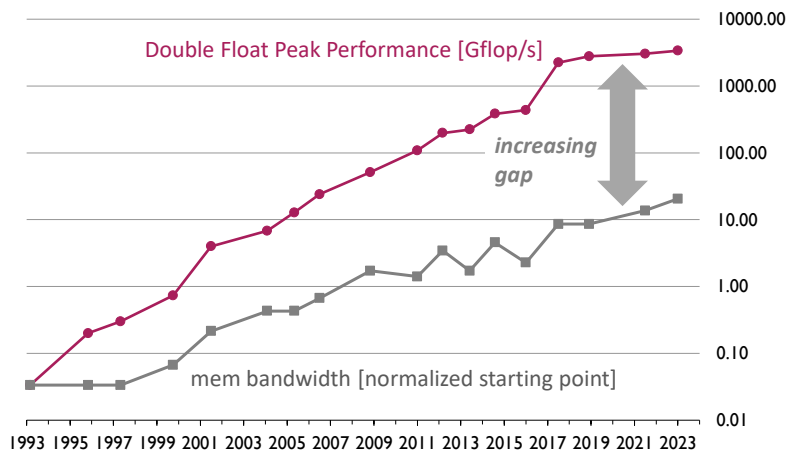
times of free exponential speedup                      parallelism: work required

Dots correspond to a somewhat random selection of Intel processors

19

19

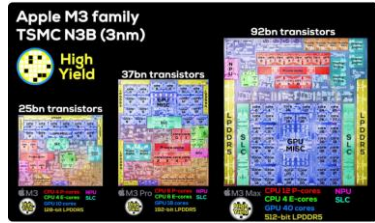
# Evolution of Processors (Intel)



Performance more and more determined by data movement

20

# And there is Processor Variety ...



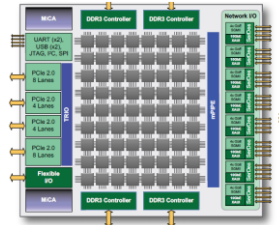
yahoo/tech

## GPUs



nvidia.com

## Domain-specific (here: Tile)



mellanox.com

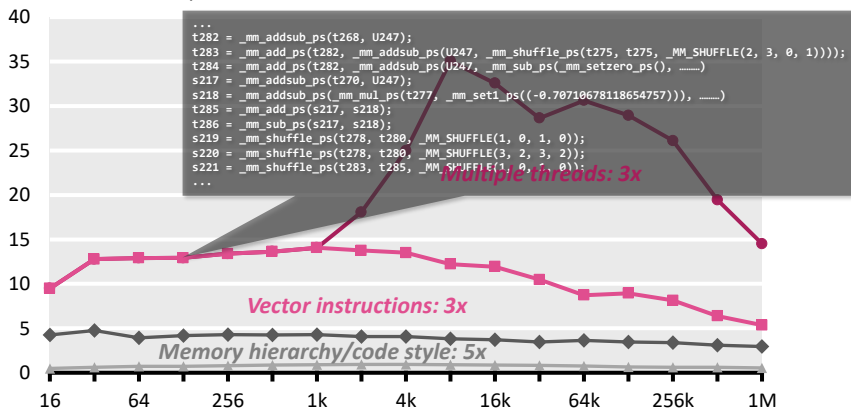
## FPGA accelerators



nallatech.com

## DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)

Performance [Gflop/s]

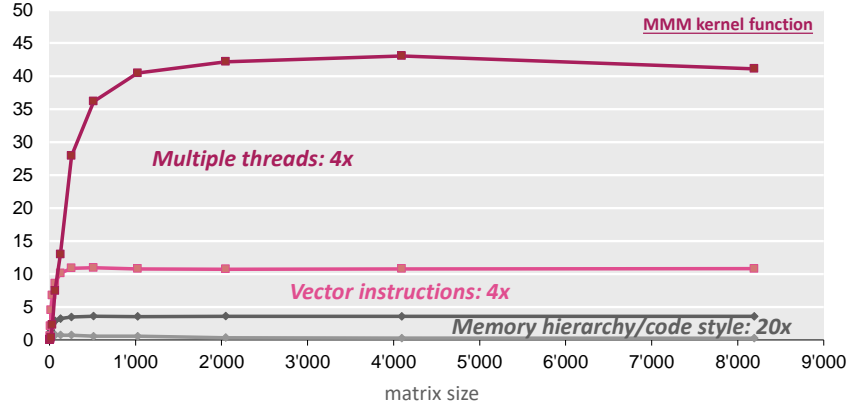


Compiler doesn't do the job

Doing by hand: *nightmare*

### Matrix-Matrix Multiplication (MMM) on 2 x Core 2 Duo 3 GHz

Performance [Gflop/s]



Compiler doesn't do the job

Doing by hand: *nightmare*

23

23

## Summary and Facts I

Implementations with same operations count can have vastly different performance (could be a 100x)

- A cache miss can be 10x more expensive than an operation
- Code style limits compiler
- Vector instructions
- Multiple cores = processors on one die

Minimizing operations count  $\neq$  maximizing performance

End of free speed-up for legacy code

- Future performance gains through increasing parallelism

24

24

## Summary and Facts II

It is very difficult to write the fastest code

- *Tuning for reduced data movement*
- *Vector instructions*
- *Code style (understand compiler limitations)*
- *Efficient parallelization (multiple threads)*
- *Requires expert knowledge in algorithms, coding, and architecture*

Fast code can be large and hard to maintain (and ugly)

- *Can violate "good" software engineering practices*

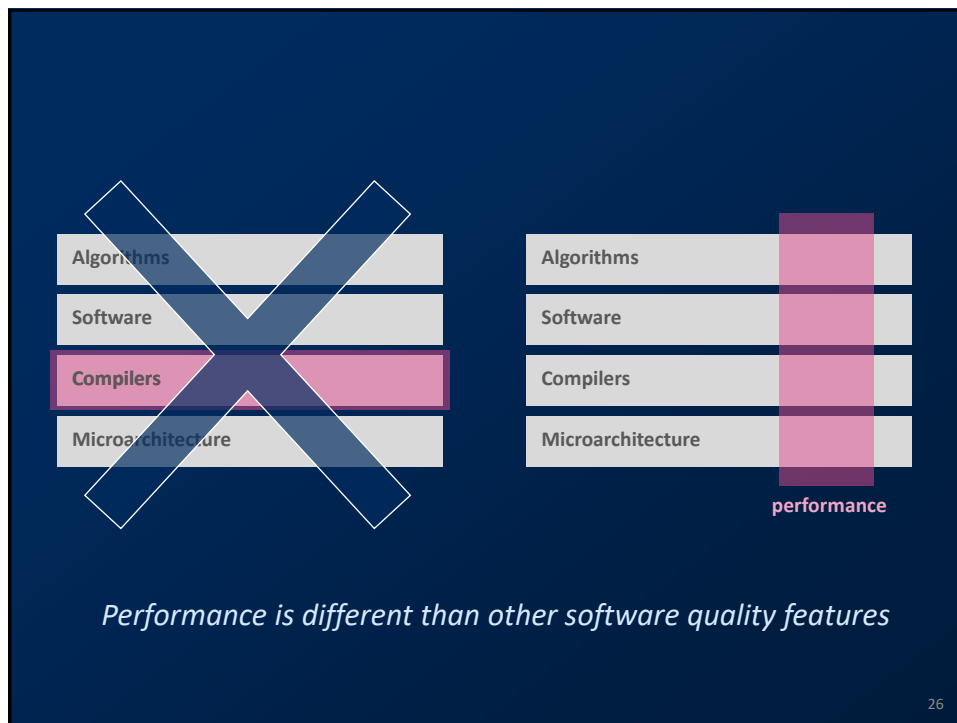
Compilers often can't do the job

- *Often intricate changes in the algorithm required*
- *Optimization blockers*
- *No good way of evaluating choices*

Highest performance is in general non-portable

25

25



26

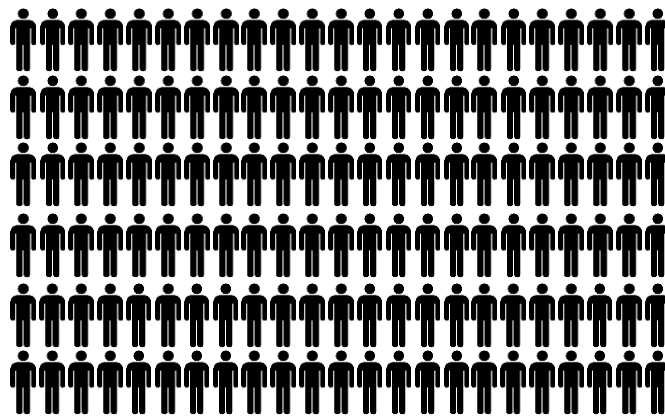
26

# Performance/Productivity Challenge

27

27

## Current Solution



*Legions* of programmers implement and optimize the *same* functionality for *every* platform and *whenever* a new platform comes out

28

28

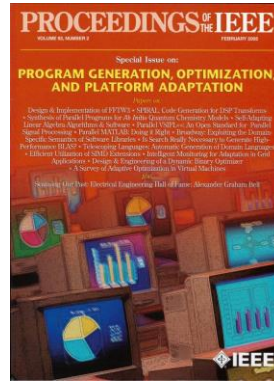
# Better Solution: Autotuning

Automate (parts of) the implementation or optimization



First autotuning research efforts

- **Linear algebra:** *Phipac/ATLAS*, LAPACK, *Sparsity/Bebop/OSKI*, Flame
- **Tensor computations**
- **PDE/finite elements:** *Fenics*
- **Adaptive sorting**
- **Fourier transform:** *FFTW*
- **Linear transforms:** *Spiral*
- ...many more since then
- **New compiler techniques**

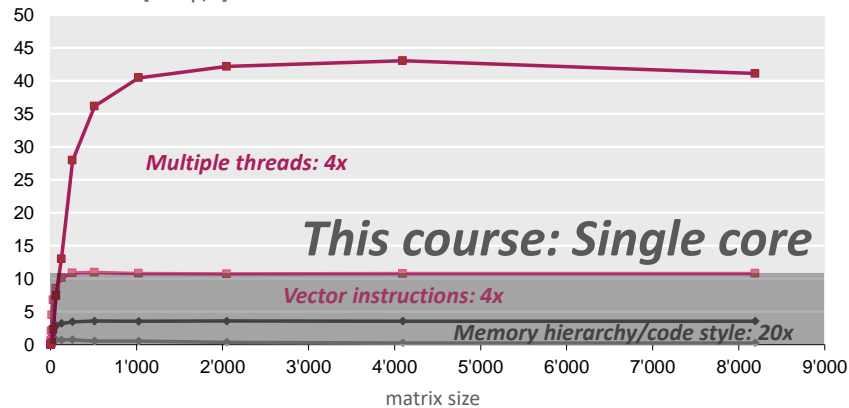


Proceedings of the IEEE special issue, Feb. 2005

*Promising ideas but not generally adopted ...*

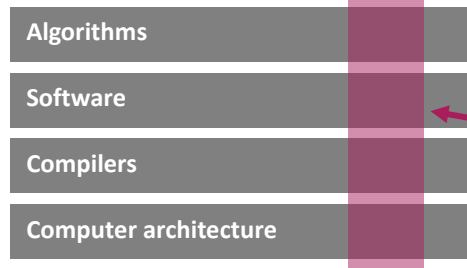
## Matrix-Matrix Multiplication (MMM) on 2 x Core 2 Duo 3 GHz

Performance [Gflop/s]



## This Course: Goals

*Fast implementations of  
numerical problems*



**Obtain a deeper understanding of performance and how to obtain it**

Learn how to write *fast code*

- *Focus: Numerical programs & single core*
- *Principles studied using important examples*
- *Applied in homeworks and a research project*

Learn about autotuning

31

31

## Today

Motivation for this course

Organization of this course

32

32



# Course: Times and Places

## Lectures:

- Monday 10-12, HG F3
- Thursday 9-10, HG G3

Extra sessions: Only used when announced on website (default: does not take place)

- Wednesday 14-16, HG E5

## Course deregistration rule:

- *Deadline: Second Friday in March*
- *After that: drop out = fail*

33

33

# Course Website Has All Info

<https://acl.inf.ethz.ch/teaching/fastcode/>

## Advanced Systems Lab - Spring 2025

### Basic Information

- **READ:** Course description, prerequisites, goals, integrity
- **READ:** FAQs
- Read the slides of the first lecture
- Course number: 263-0007, 8 credits
- Lectures: M 10:15-12:00, HG F3; Th 9:15-10:00, HG G3; occasional substitute lectures: W 14:15-16:00 HG E5
- The lectures are not streamed but recorded. Login info is different from netzh and has been sent by email.
- **Instructor:** Markus Püschel (CAB H69.3, pueschel@inf)
- **Head TA:**
  - Tommaso Pegolotti (TP)
- **TAs:**
  - Mihail Khailov (MK)
  - Hicham Leghettas (ML)
  - Dionisios Spiliopoulos (DS)
  - Shien Zhu (SZ)
  - Emil Schätzle (ES)
  - Tommaso Bonato (TB)
- **Mailing lists:**
  - For technical questions: fastcode@lists.inf.ethz.ch (emails to this address go to the lecturer and all TAs)
  - Forum to find project partner: fastcode-forum@lists.inf.ethz.ch (emails go to all students who have no partner yet and Head TA)
- **Office Hours: (TBD.)**

### Time Line

This list can be subject to minor changes, which would be announced in a timely manner.

Fr 07.03.	Project team and project registered in the <b>project system</b> ; start project anytime now
Th 06.03.	HW1 due
Th 13.03.	HW2 due
Th 27.03.	HW3 due
Th 17.04.	HW4 due
Wed 16.04.	Midterm

34

34

## Team and Communication

Head TA: Tommaso Pegolotti



Other TAs:

*Mikhail Khalilov (MK)*  
*Hicham Leghettas (ML)*  
*Dionisios Spiliopoulos (DS)*  
*Shien Zhu (SZ)*  
*Emil Schätzle (ES)*  
*Tommaso Bonato (TB)*

[Course website](#) has *ALL* information

Questions:

- *Office hours (during period with homeworks): see website*
- [fastcode@lists.inf.ethz.ch](mailto:fastcode@lists.inf.ethz.ch): goes to TAs and lecturers

Finding project partner: [fastcode-forum@lists.inf.ethz.ch](mailto:fastcode-forum@lists.inf.ethz.ch)

35

35

## Prerequisites and Organization

Requirements

- *solid C programming skills*
- *matrix algebra*
- *Master student or above*

Grading

- *40% research project*
- *30% midterm exam*
- *30% homework*

Wednesday slot

- *Gives you scheduled time to work together*
- *Occasionally we will move lecture there (will communicate if so)*
- *By default will not take place*

36

36

## Research Project: Overview

Teams of 4

Yes: 4

*Topic:* Very fast implementation of a numerical problem

*Until March 7:*

- *find a project team*
- *pick from list (on course website posted this week) or suggest to me*
- *Register in our project system + you get a git repo for project*

Show “milestones” during semester: One-on-one meetings

Give short presentation end of semester

Write 7 page standard conference paper (template on website)

Submit final code

37

37

## Finding Project Team

Teams of 4: no exceptions

Use [fastcode-forum@lists.inf.ethz.ch](mailto:fastcode-forum@lists.inf.ethz.ch):

- *“I am looking for a team, am interested in anything related to visual computing”*
- *“We are a group of three with a project on xxx and are looking for a fourth team member”*

In the beginning all of you are registered to that list

Once team is formed register it in our [project system](#),  
and you will automatically be deregistered you from mailing list

38

38

## Finding Project

Pick from list on website or select on yourself

Projects from website (*posted later this week*): number of teams is limited as shown, *once picked it is final*

Select yourself:

- *Pick something you are interested in*
- *Nothing that is dominated by standard linear algebra (matrix-matrix mult, solving linear systems) or FFT, no stencil computations*
- *Send me a short explanation plus a publication with the algorithm for approval*

Exact scope can be adapted during semester

- *reduced to critical component*
- *specialized*

*You are in charge of your project!*

- *If too big, adapt*
- *If too easy, expand*
- *Don't come after 2 months and say project does not work*

39

39

## Organize Project

Work as a team

**Start *asap* with a team meeting, check milestones in project system**

week of 28.04.	1st one-on-one project meeting (minimal milestone: base implementation done, tested, performance plot, initial optimization plan, explain how you plan to divide the optimization work)
week of 19.05.	2nd one-on-one project meeting
week of 02.06.	Project presentations
Fr 20.06.	Project report due

Keep communicating *regularly* during semester

Be fair to your team members, be a team player

Being able to work as a team is part of the exercise

If you give up on the course and thus the project, say so

If you don't contribute we will fail you for the project

40

40

## Research Project: Possible Failures

Don't do this:

- *never meet*
- *not respond to emails*
- *"I don't have time right to work on this project in the next few months, why don't you start and I catch up later"*
- *"I have a paper deadline in 1 month, cannot do anything else right now"*
- **while** *not desparate(project-partners) do*  
*"I do my part until end of next week"*  
*... nothing happens ...*  
**end**
- *"why don't you take care of the presentation"*
- *"why don't you take care of the report, I'll do the project presentation"*

Single point of failure:

- *One team member is the expert on the project and says: I quickly code up the basic infrastructure, then the three of you can join working on parts*
- *1 month later, the "quickly coding up" ...*

41

41

## Midterm Exam

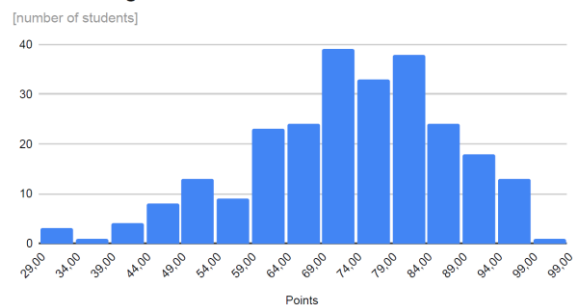
Covers first part of course *Point distribution 2024 (max = 100)*

*Date:* Wed, April 16  
(might be 16:00-18:00)  
or small chance that  
day before or after

*No substitute date*

*There is no final exam*

Points histogram



42

42

# Homework

4 homeworks during first half of course

Done individually, we use Moodle and Code Expert for some autograding

Exercises on algorithm/performance analysis, check out previous years

Implementation exercises

- *Concrete numerical problems*
- *Study the effect of program optimizations, use of compilers, use of special instructions, etc. (Writing C code + creating runtime/performance plots)*

Small part of homework grade for neatness

Late homework policy:

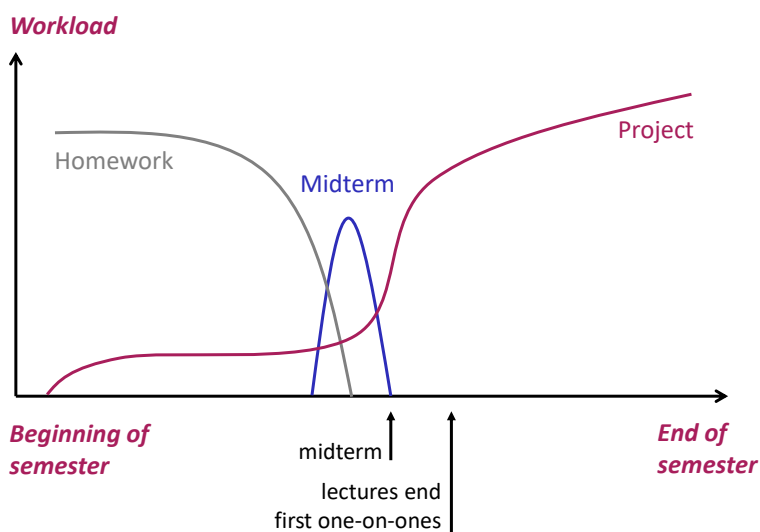
- **No deadline extensions, but**
- **3 late days for the entire semester (at most 2 for one homework)**

Solving homeworks completely analogous to homeworks in prior years is no 100% guarantee for full points – the material gets updated occasionally

43

43

# Workload During Semester (Sketch)



44

44

# Academic Integrity

Zero tolerance cheating policy (cheat = fail + being reported)

## Homeworks

- *All single-student*
- *Don't look at other students code*
- *Don't copy code from anywhere*
- *Don't share your code or solutions*
- *Ok to discuss things – but then you have to do it alone*
- ***Careful with online discussion channels!***

We use Moss to check copying (check out what it can do)

## *Don't do copy-paste*

- *code*
- *ANY text*
- *pictures*
- *especially not from Wikipedia*

45

45

# Academic Integrity

Zero tolerance cheating policy (cheat = fail + being reported)

## Projects

- *No showing code or copying code from other teams*
- ***Careful with online discussion channels!***

## AI Tools

- *Should not be used for anything that is graded core work (e.g., solving a homework problem, writing optimized code for the project)*
- *Other uses are ok but you have to explicitly mention it (e.g., in the project report)*

46

46

## Background Material

See course website and links in slides

Prior versions of this course: see website

I post all slides, notes, etc. on the course website

Training material: midterms and homeworks from prior years

On certain topics, and for more details if desired, feel free to consult extra resources that are easily found by a web search

47

47

## Class Participation

All material I cover goes on the website, but not all my verbal explanations

We record all lectures (login credentials will be communicated by email)

It is a good idea to attend but not obligatory (obviously)

Do ask questions

*If you drop the course, please unregister in mystudies*

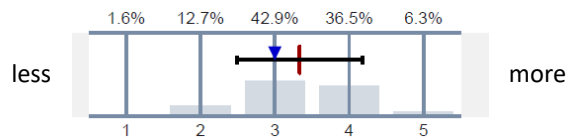
48

48

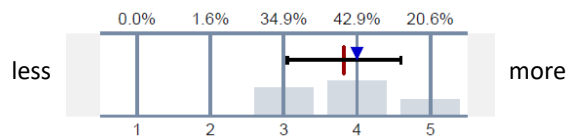


## Feedback 2023 (none in 2024)

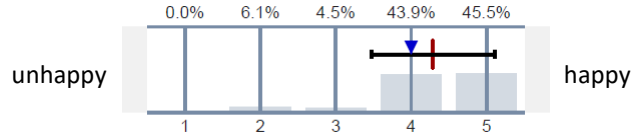
How technically demanding in comparison with other courses



Workload in comparison with other courses



Overall satisfaction with course



49

49

## Some Comments from Feedback

The amount of work required is too large.

Insane amount of work. There is absolutely no chance one can get the required 30 credits per semester with this course because there simply is not enough time to work on the other courses.

The homework takes a very long time.

I wouldn't change much to be honest, this is a pretty intense course that requires a lot of studying and time spent but I understand it is an Inter Focus course so I didn't expect it to be very different.

Amazing class [...] reasonable workload, fair midterm, good homework

The graded assignments are very helpful and nice to solve. Moreover, the project is also very nice

This course might honestly be the most well structured course out of all the ones I've taken at ETH

Everything was amazing, the course has very interesting, up-to-date materials

50

50