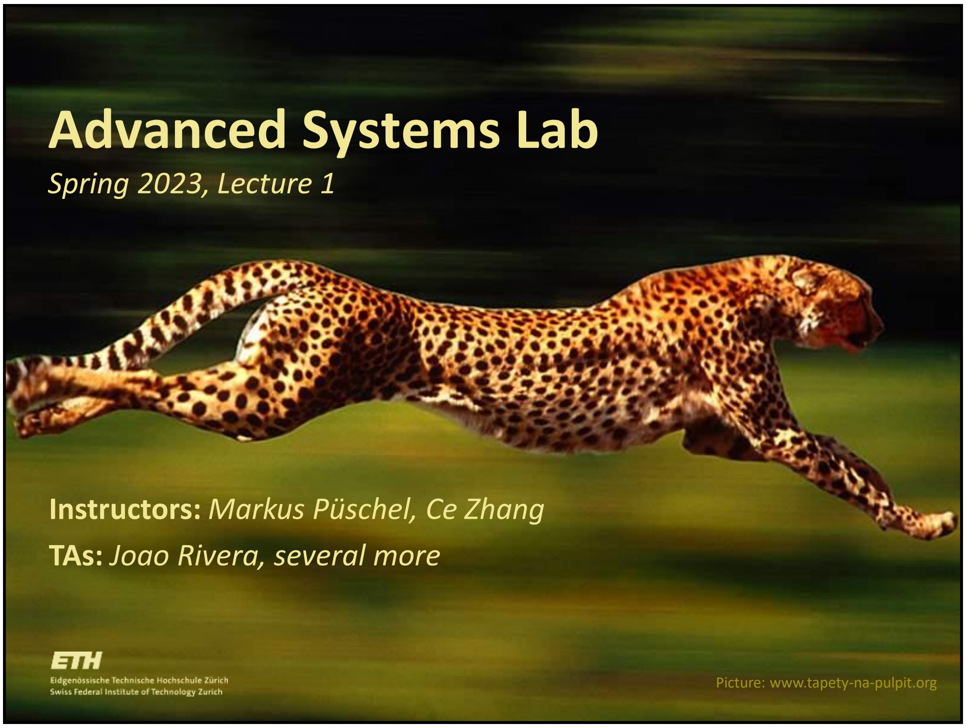


1

Minds open…

… Laptops closed

slide by Bertrand Meyer

2

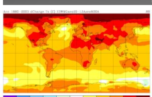
2

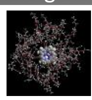
Today

Motivation for this course

Organization of this course

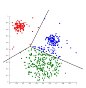
Scientific Computing

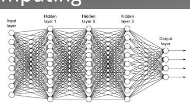




Physics/biology simulations, …

Cloud Computing





Data analytics, machine learning, …

Consumer Computing

Audio/image/video processing, …

Embedded Computing

Signal processing, communication, control, …

Numerical Computing

Unlimited need for performance

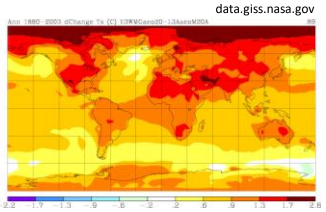
Large set of applications, but …

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

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

Scientific Computing (Clusters/Supercomputers)

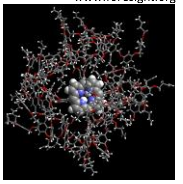
data.giss.nasa.gov



Climate modelling

Finance simulations

www.foresight.org



Molecular dynamics

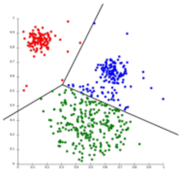
Other application areas:
- Fluid dynamics
- Chemistry
- Biology
- Medicine
- Geophysics

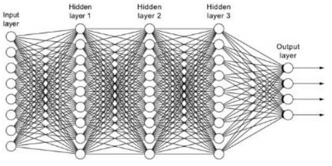
Methods:
- Mostly linear algebra
- PDE solving
- Linear system solving
- Finite element methods
- Others

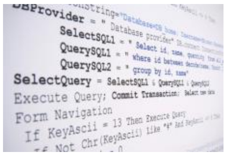
5

5

Cloud Computing (Server Farms)







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

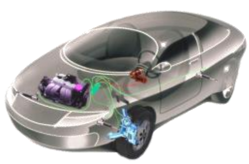
7

Embedded Computing (Low-Power Processors)

www.dei.unipd.it

Sensor networks

www.ece.drexel.edu



Cars

www.microway.com.au

Robotics

Applications:
- Signal processing
- Control
- Communication
- Inference
- Others

Methods:
- Linear algebra
- Transforms
- Filters
- Coding
- Others

8

8

Classes of Performance-Critical Functions

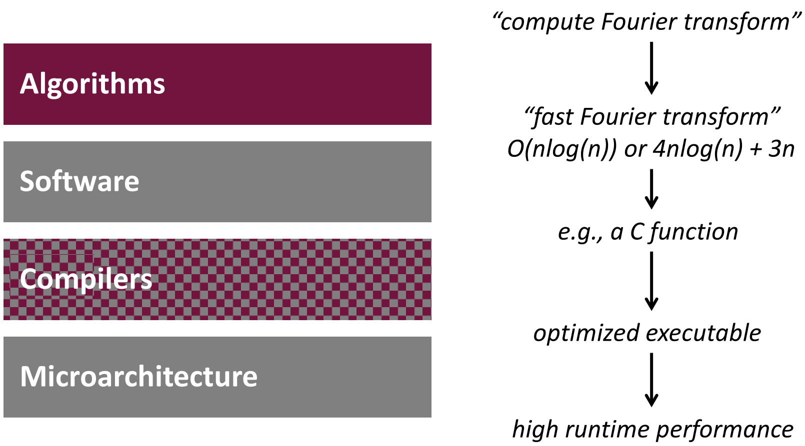
- Transforms
- Filters/correlation/convolution/stencils/interpolators
- Dense linear algebra functions
- Sparse linear algebra functions
- Tensor operations
- Coder/decoders
- Graph algorithms
- *… several others*

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?



How well does this work?

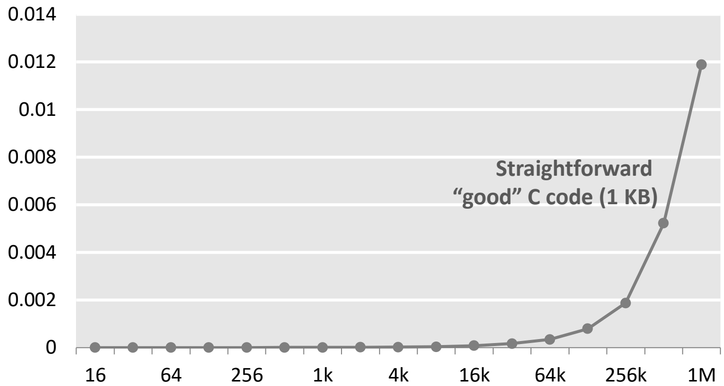
10

10

The Problem: Example 1

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

Runtime [s]



 **or**  **?**

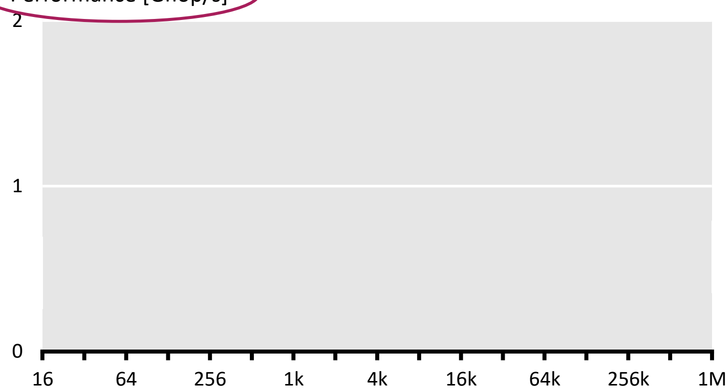
11

11

The Problem: Example 1

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

Performance [Gflop/s]



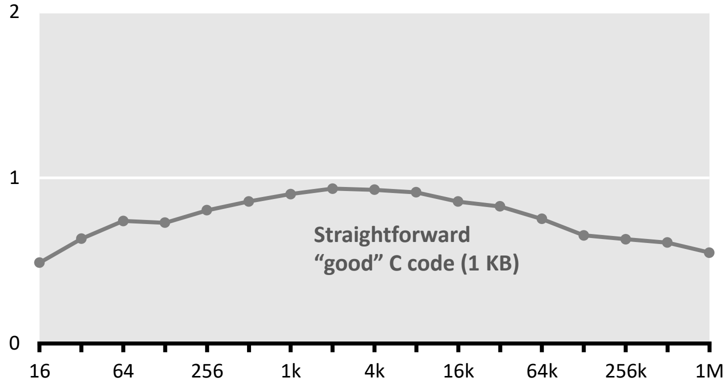
12

12

The Problem: Example 1

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

Performance [Gflop/s]

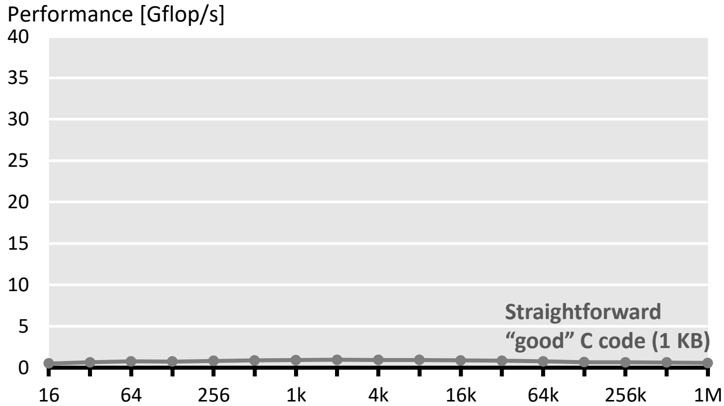


 **or**  **?**

The Problem: Example 1

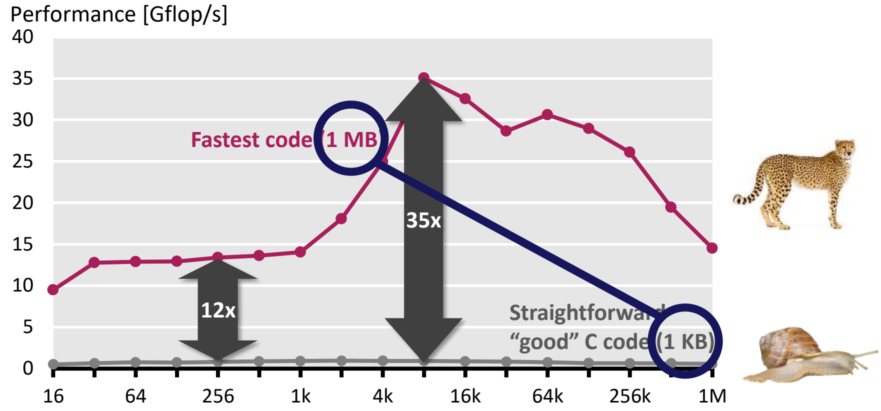
DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)

Performance [Gflop/s]



The Problem: Example 1

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

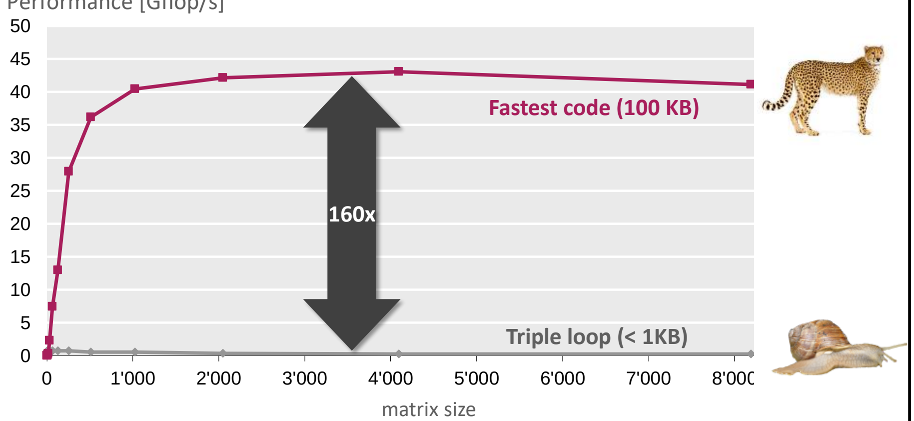


Vendor compiler, best flags

Roughly same operations count

The Problem: Example 2

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



Vendor compiler, best flags

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

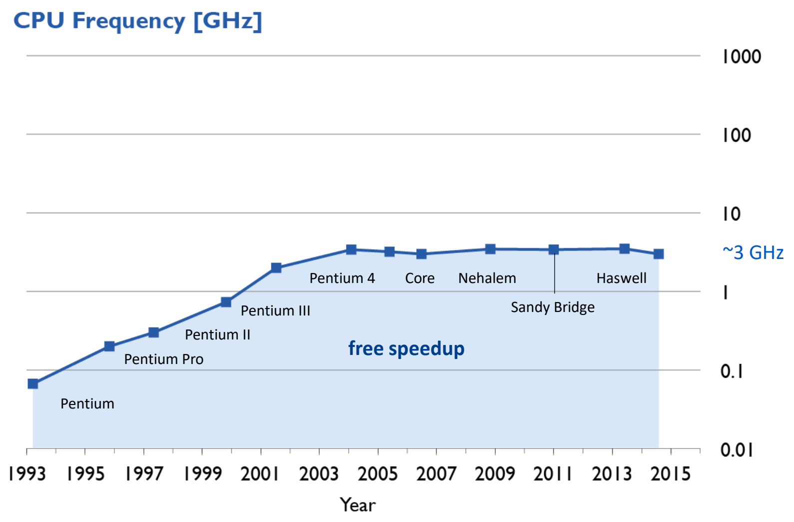
Model predictive control	Singular-value decomposition
Eigenvalues	Mean shift algorithm for segmentation
LU factorization	Stencil computations
Optimal binary search organization	Displacement based algorithms
Image color conversions	Motion estimation
Image geometry transformations	Multiresolution classifier
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

"Theorem:"

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

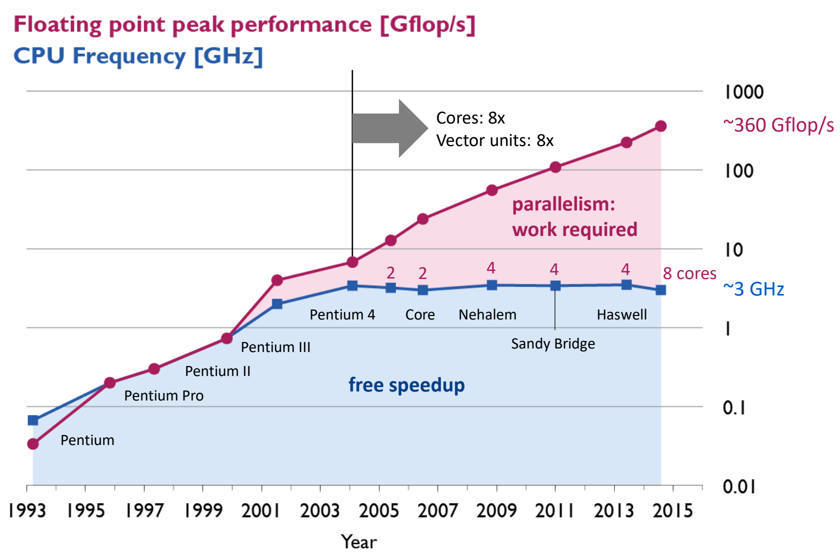
$$\frac{\textit{Performance of optimal implementation of } f}{\textit{Performance of straightforward implementation of } f} \approx \textit{10–100}$$

Evolutions of Processors (Intel)



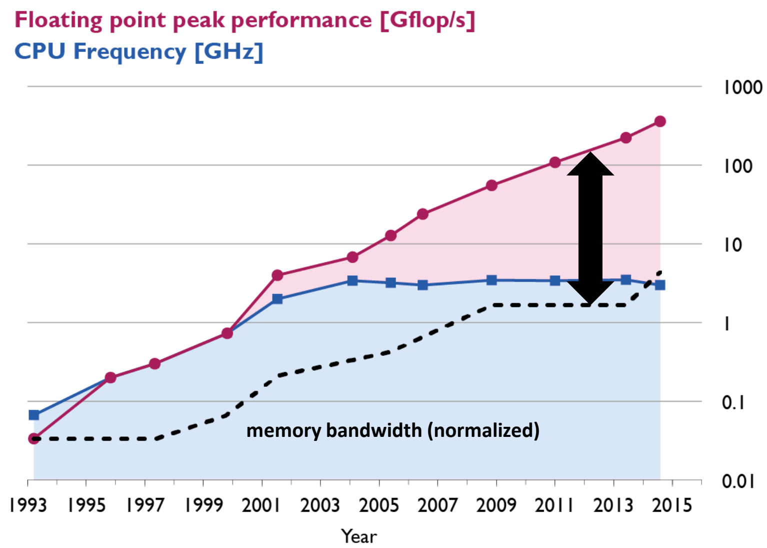
Source: Wikipedia/Intel/PCGuide

Evolutions of Processors (Intel)



Source: Wikipedia/Intel/PCGuide

Evolutions of Processors (Intel)

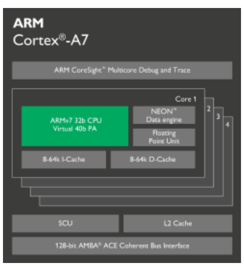


Source: Wikipedia/Intel/PCGuide

21

21

And there is Processor Variety …

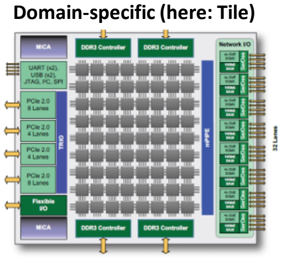


arm.com

GPUs

nvidia.com

Domain-specific (here: Tile)



mellanox.com

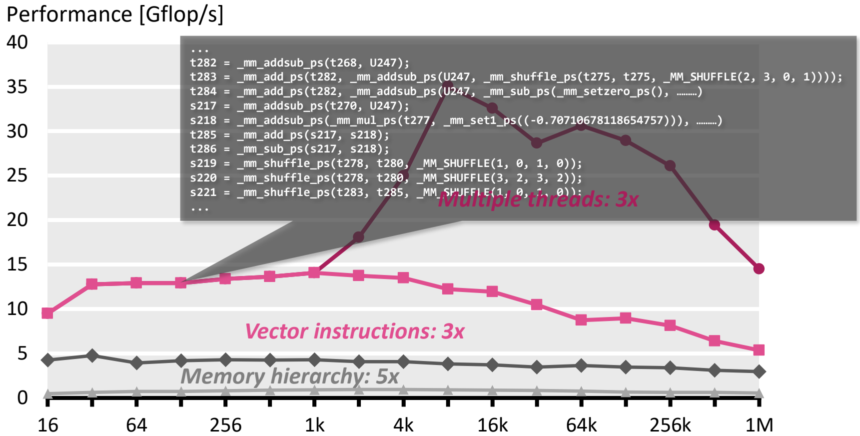
FPGA accelerators

nallatech.com

22

22

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



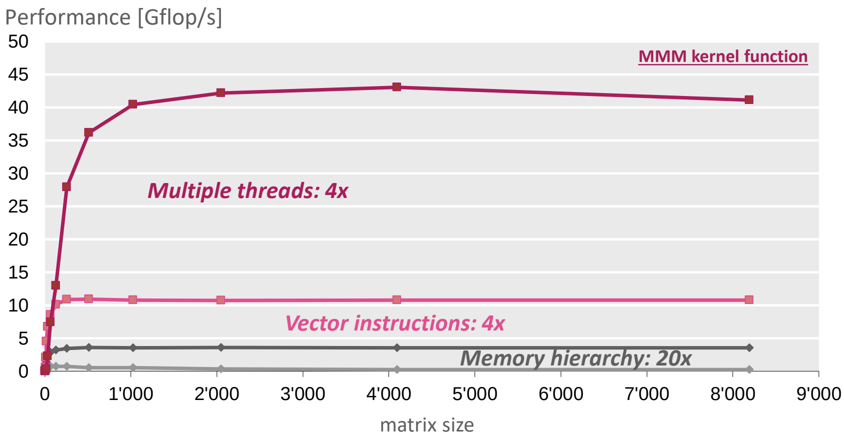
Compiler doesn't do the job

Doing by hand: *nightmare*

23

23

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



Compiler doesn't do the job

Doing by hand: ***nightmare***

24

24

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 ≠ maximizing performance

End of free speed-up for legacy code

- *Future performance gains through increasing parallelism*

Summary and Facts II

It is very difficult to write the fastest code

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

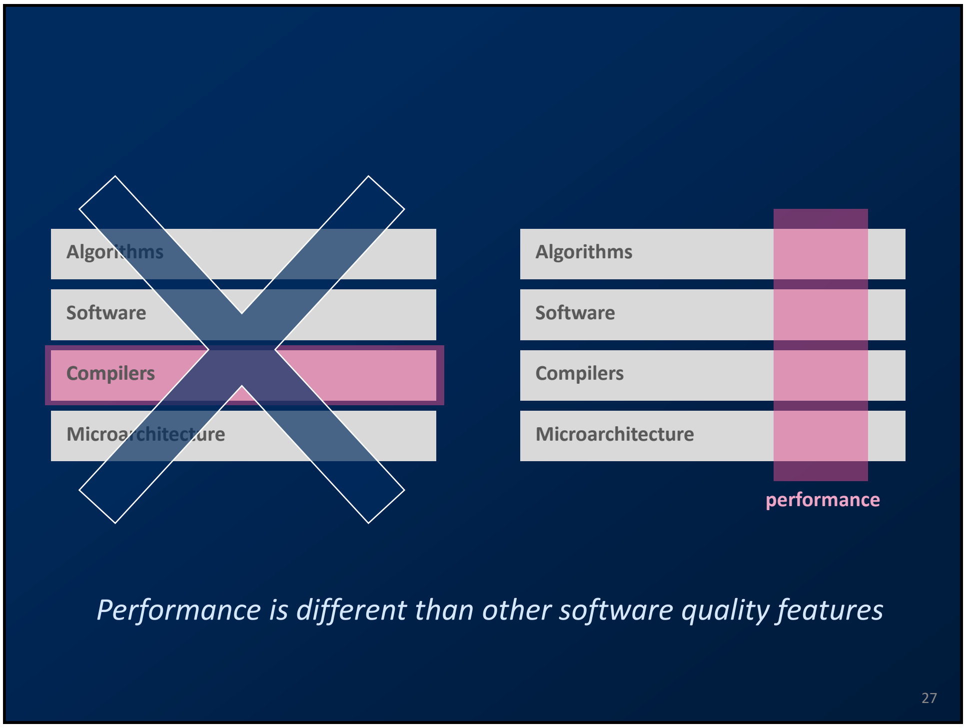
Fast code can be large and hard to maintain

- *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



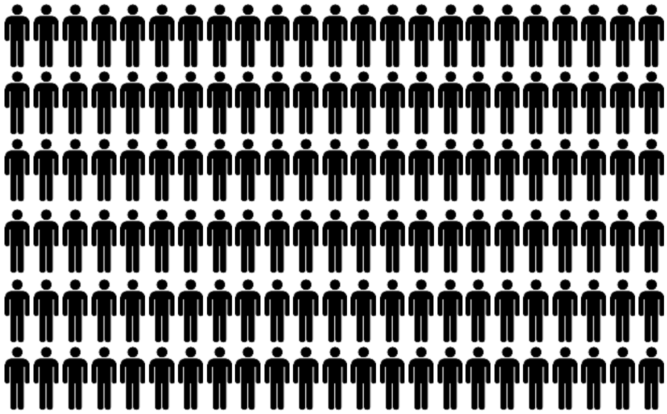
27

Performance/Productivity **Challenge**

28

28

Current Solution



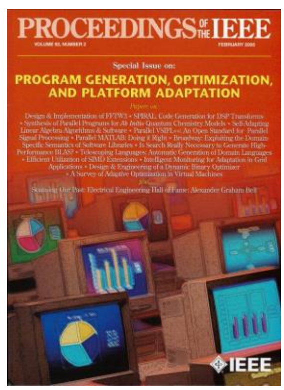
Legions of programmers implement and optimize the *same* functionality for *every* platform and *whenever* a new platform comes out

Better Solution: Autotuning

Automate (parts of) the implementation or optimization

Research efforts

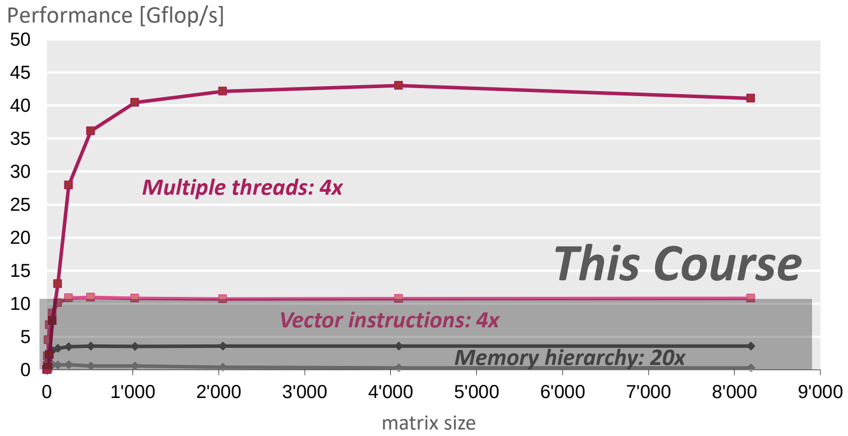
- *Linear algebra:* ***Phipac/ATLAS****, LAPACK,* ***Sparsity/Bebop/OSKI****, Flame*
- *Tensor computations*
- *PDE/finite elements: Fenics*
- *Adaptive sorting*
- ***Fourier transform: FFTW***
- *Linear transforms: Spiral*
- *…many more since then*
- *New compiler techniques*



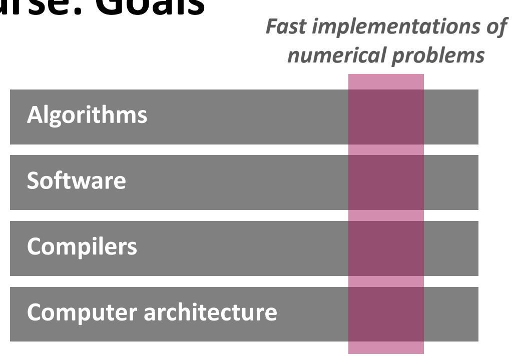
Proceedings of the IEEE special issue, Feb. 2005

Promising area but much more work needed …

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



This Course: Goals



Obtain a deeper understanding of performance

Learn how to write *fast code* for numerical problems

- *Focus: Single core*
- *Principles studied using important examples*
- ***Applied in homeworks and a research project***

Learn about autotuning

Today

Motivation for this course

Organization of this course

Course: Times and Places

Lectures:

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

Extra sessions: Only used when announced on website

- *Wednesday 14-16, HG E5*

Course deregistration rule:

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

Course Website Has all Info

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

Advanced Systems Lab - Spring 2023

Basic Information

- READ: Course description, prerequisites, goals, integrity
- Read the slides of the first lecture
- FAQs
- Course number: 263-0007, 8 credits
- Spring 2023, lectures: M 10:15-12:00, HG F3; Th 9:15-10:00 HG F3; occasional substitute lectures: W 14:15-16:00 HG E5
- **Instructor**: Markus Püschel (CAB H69.3, pueschel at inf)
- **Head TA**:
  - Joao Rivera (JR)
- TAs:
  - Tommaso Pegolotti (TP)
  - Theodoros Theodoridis (TT)
  - Mikhail Khalilov (MK)
  - Yann Girsberger (YG)

Time Line

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

Fr 10.03.	Project team and project registered in the project system; start project anytime now
Th 09.03.	HW1 due
Th 16.03.	HW2 due
Th 30.03	HW3 due
Th 13.04.	HW4 due
Wed 26.04.	Midterm
week of 01.05	1st one-on-one project meeting (minimal milestone: base implementation, tested, performance plot, initial optimization plan)

etc.

35

35

Team and Communication

Lecturers: Markus Püschel and Ce Zhang

Head TA: *Joao Rivera*

Other TAs:

Yann Girsberger
Mikhail Khalilov
Tommaso Pegolotti
Theodoros Theodoridis

Course website has *ALL* information

Questions:

- *Office hours (during HW period): see website*
- *fastcode@lists.inf.ethz.ch: goes to TAs and lecturers*

Finding project partner: fastcode-forum@lists.inf.ethz.ch

36

36

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*

37

37

Research Project: Overview

Teams of 4

Yes: 4

Topic: Very fast implementation of a numerical problem

Until March 10th:

- *find a project team*
- *suggest to me a problem or pick from list (on course website)*
  Tip: pick something from your research or that you are interested in
- *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 8 page standard conference paper (template on website)

Submit final code

38

38

Finding Project Team

Teams of 4: no exceptions

Use fastcode-forum@lists.inf.ethz.ch:

- *"I have a project (short description) and am looking for partners"*
- *"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,
tell Joao, and we deregister you from mailing list

39

39

Finding Project

Pick from list on website or select on yourself

Projects from website: number of teams is limited, ***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***

40

40

Organize Project

Work as a team

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

week of 01.05	1st one-on-one project meeting (minimal milestone: base implementation, tested, performance plot, initial optimization plan)
week of 22.05.	2nd one-on-one project meeting
week of 05.06.	Project presentations
Fr 23.06.	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, say so

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

41

41

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" …*

42

42

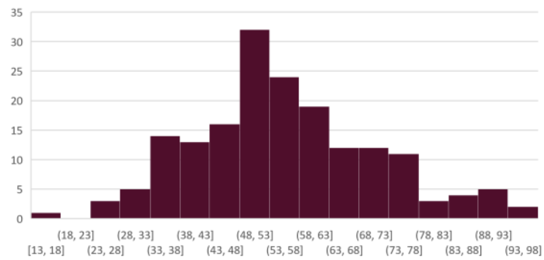
Midterm Exam

Covers first part of course

Date: Wed, April 26th

No substitute date

Point distribution 2022 (max = 100)



There is no final exam

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)*

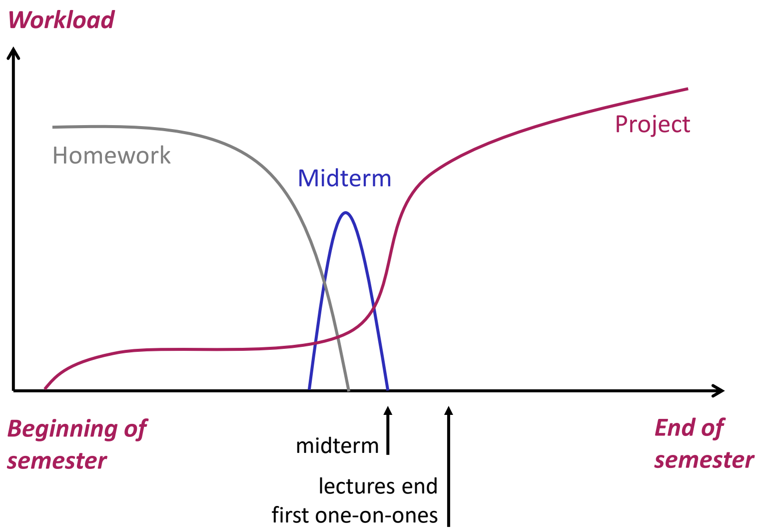
Small part of homework grade for neatness

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

Workload During Semester (Sketch)



45

45

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*

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*

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, feel free to consult extra resources (e.g., Wikipedia) 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

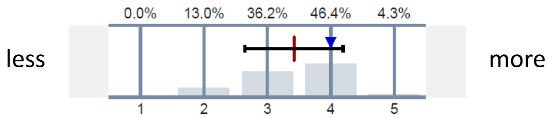
If you drop the course, please unregister in mystudies

48

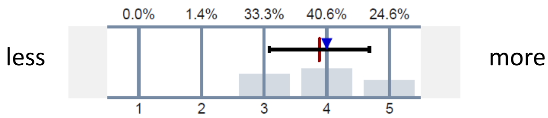
48

Feedback 2021

How technically demanding in comparison with other courses



Workload in comparison with other courses



Overall satisfaction with course