# How to Write Fast Numerical Code

*Spring 2017, Lecture 1*

**Instructor:** *Markus Püschel*

**TAs:** *Alen Stojanov, Georg Ofenbeck, Gagandeep Singh*

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Picture: www.tapety-na-pulpit.org

## Minds open...

## ... Laptops closed

*slide by Bertrand Meyer*

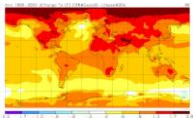
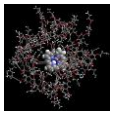
2

# Today

- **Motivation for this course**
- **Organization of this course**



Scientific Computing

*Physics/biology simulations*

Consumer Computing


*Audio/image/video processing*

Embedded Computing

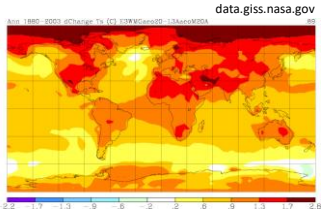

*Signal processing, communication, control*

# Computing

- **Unlimited need for performance**
- **Large set of applications, but …**
- **Relatively small set of critical components (100s to 1000s)**
  - Matrix multiplication
  - Discrete Fourier transform (DFT)
  - Viterbi decoder
  - Shortest path computation
  - Stencils
  - Solving linear system
  - ….

# Scientific Computing (Clusters/Supercomputers)

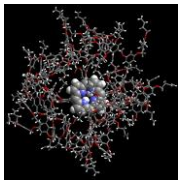
data.giss.nasa.gov



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



# Consumer Computing (Desktop, Phone, …)

Photo/video processing

Audio coding

Security


Image compression

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

# Embedded Computing (Low-Power Processors)

www.dei.unipd.it

**Sensor networks**

www.ece.drexel.edu

**Cars**

www.microway.com.au

**Robotics**

**Computation needed:**
- Signal processing
- Control
- Communication

**Methods:**
- Linear algebra
- Transforms, Filters
- Coding

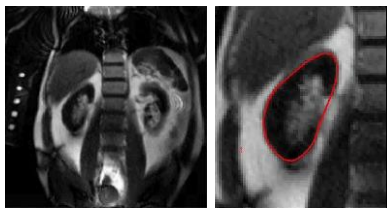


# Research (Examples from Carnegie Mellon)

Bhagavatula/Savvides

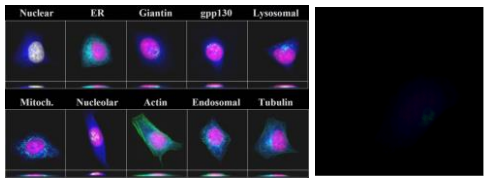
**Biometrics**

Moura



**Medical Imaging**

Kovacevic




**Bioimaging**

Kanade

**Computer vision**

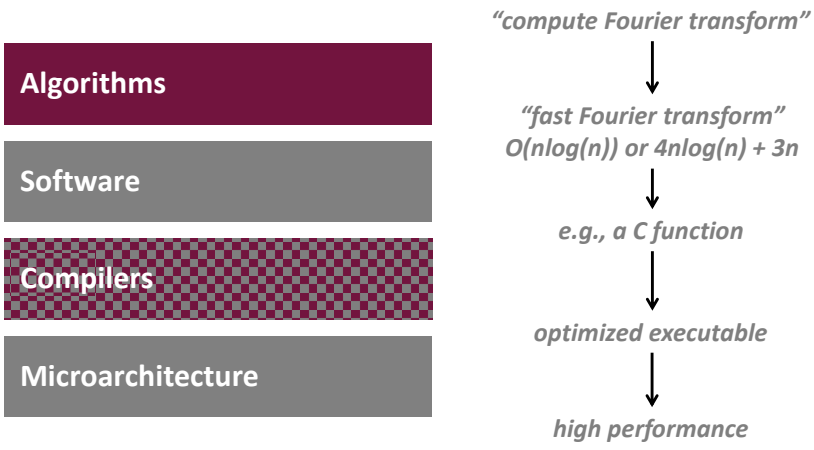
# Classes of Performance-Critical Functions

- Transforms
- Filters/correlation/convolution/stencils/interpolators
- Dense linear algebra functions
- Sparse linear algebra functions
- 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



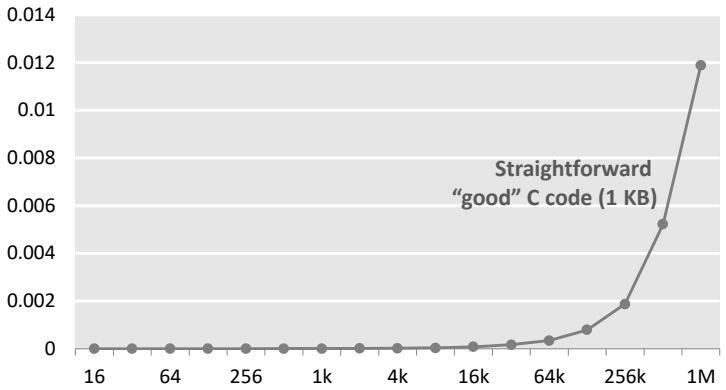
# How Hard Is It to Get Fast Code?




*How well does this work?*

# The Problem: Example 1

**DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)**
Runtime [s]





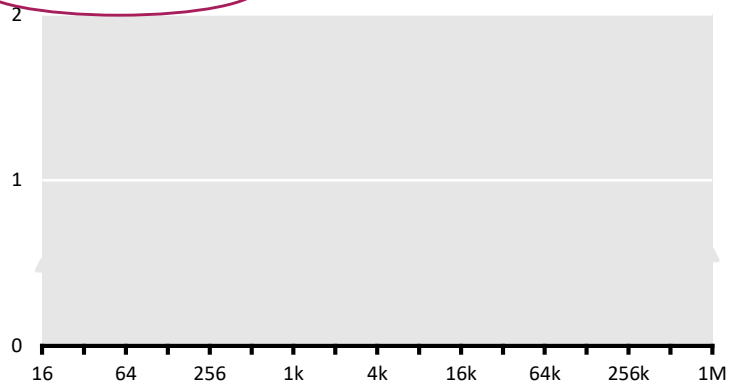
or

?



# The Problem: Example 1

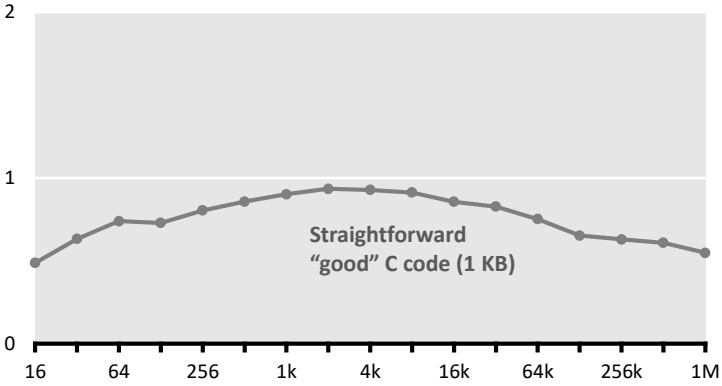
**~~DFT (single precision)~~ on Intel Core i7 (4 cores, 2.66 GHz)**
Performance [Gflop/s]



# The Problem: Example 1

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

Performance [Gflop/s]





or

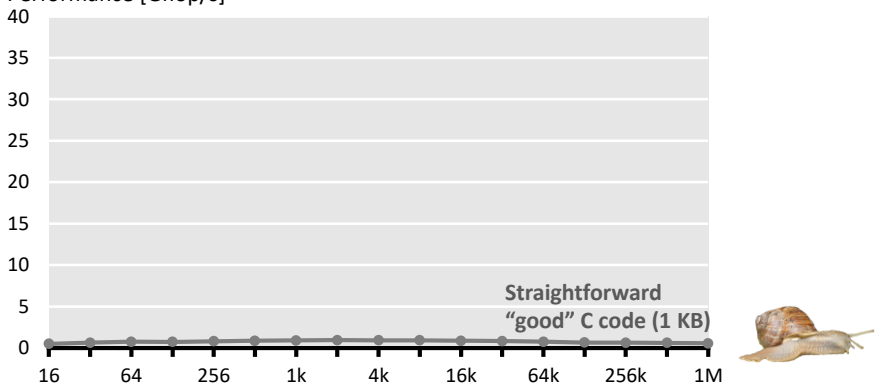
?



# The Problem: Example 1

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

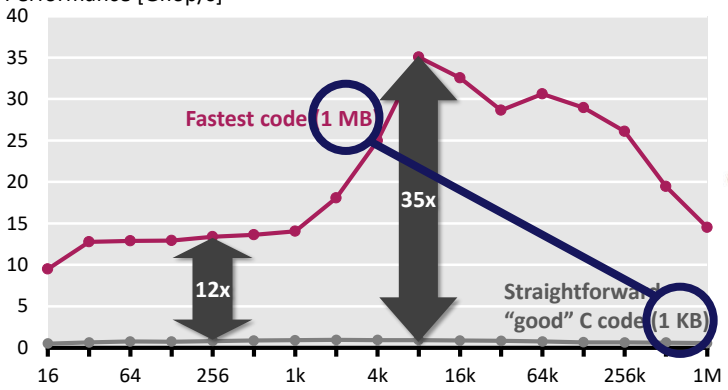
Performance [Gflop/s]



# The Problem: Example 1

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

Performance [Gflop/s]



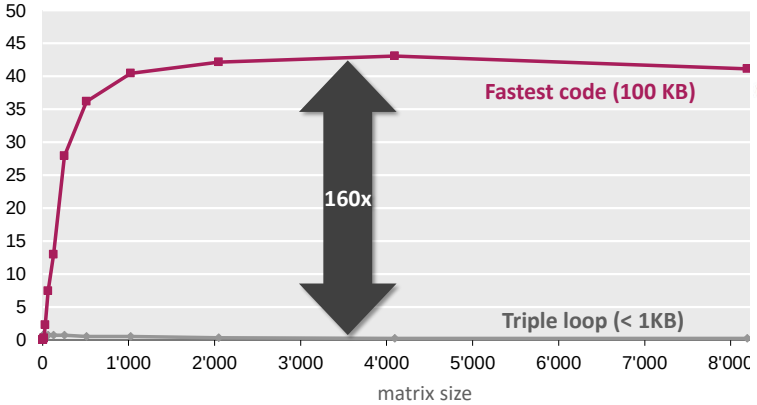

- Vendor compiler, best flags
- Roughly same operations count



# 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 ($2n^3$)

| | |
|---|---|
| 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 |

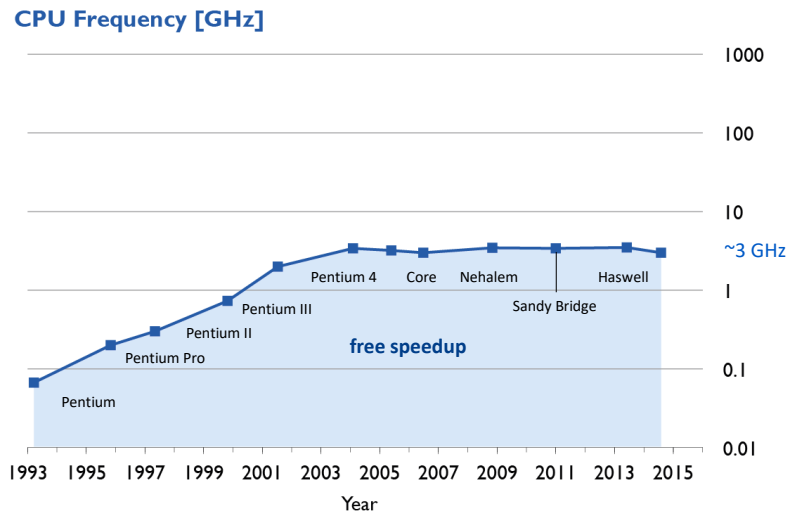


# "Theorem:"

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

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

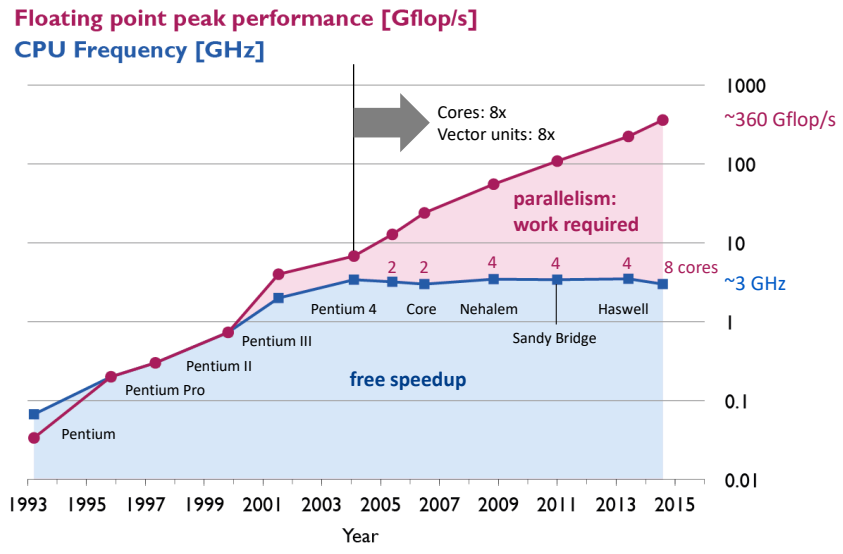
# Evolutions of Processors (Intel)




*Source: Wikipedia/Intel/PCGuide*

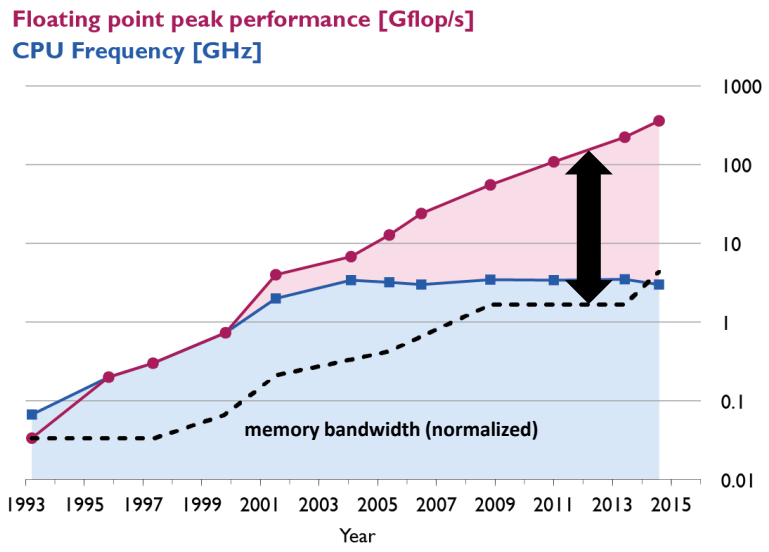


# Evolutions of Processors (Intel)




*Source: Wikipedia/Intel/PCGuide*

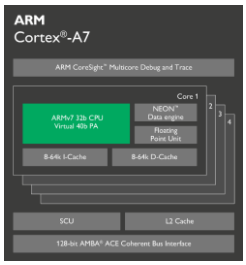
# Evolutions of Processors (Intel)




*Source: Wikipedia/Intel/PCGuide*

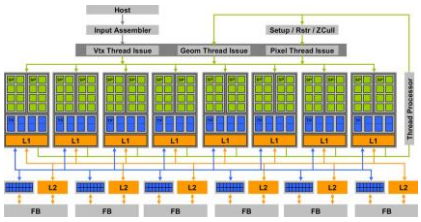


# And there is Processor Variety …



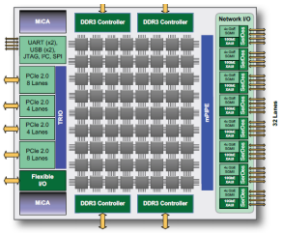

*arm.com*

**Nvidia Tesla**




*beyond3d.com*

**Domain-specific (here: Tile)**



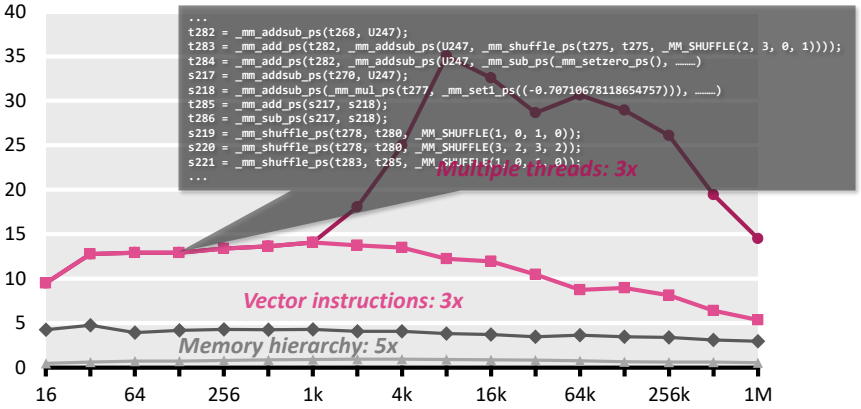

*mellanox.com*

**FPGA accelerators**

*nallatech.com*

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

Performance [Gflop/s]

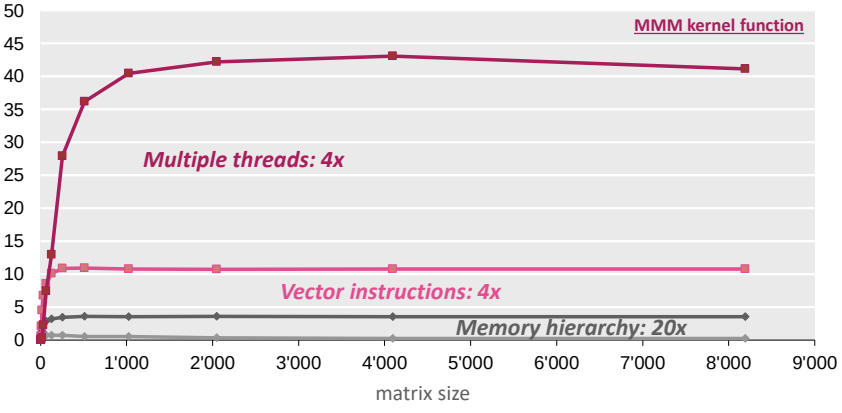



- Compiler doesn't do the job
- Doing by hand: ***nightmare***



**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***

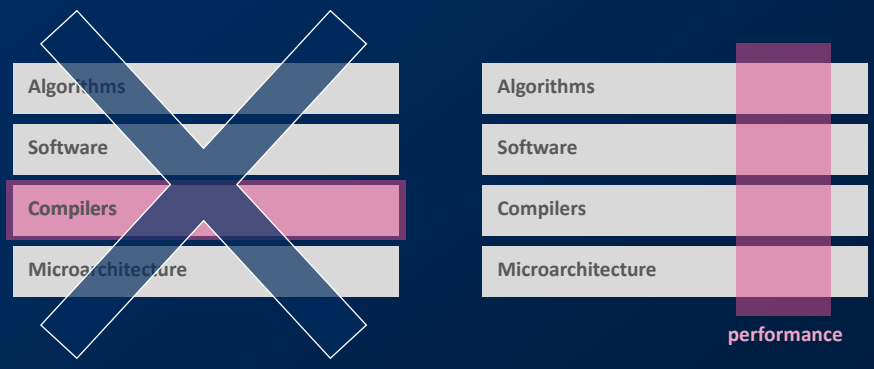
# Summary and Facts I

- **Implementations with same operations count can have vastly different performance (up to 100x and more)**
  - A cache miss can be 100x more expensive than an operation
  - 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
  - Efficient parallelization (multiple threads)
  - Requires expert knowledge in algorithms, coding, and architecture
- **Fast code can be large**
  - Can violate "good" software engineering practices
- **Compilers often can't do the job**
  - Often intricate changes in the algorithm required
  - Parallelization/vectorization still unsolved
- **Highest performance is in general non-portable**



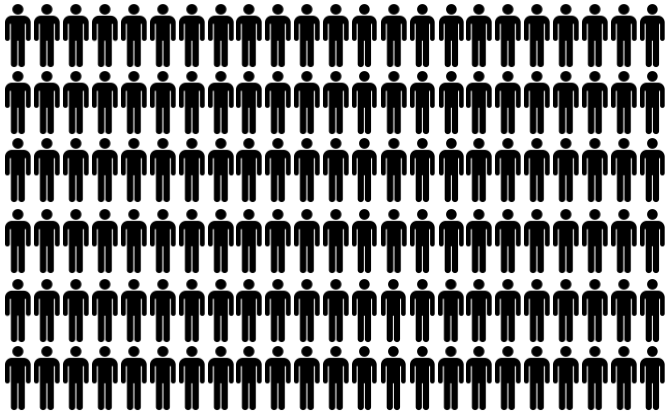

*Performance is different than other software quality features*

27

# Performance/Productivity **Challenge**

28

# Current Solution



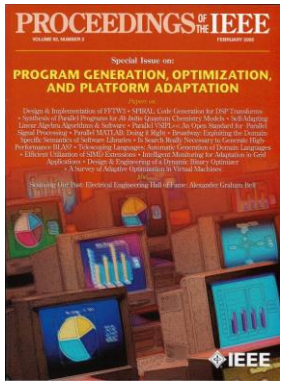
***Legions*** of programmers implement and optimize the ***same*** functionality for ***every*** platform and ***whenever*** a new platform comes out



# Better Solution: Autotuning

- **Automate (parts of) the implementation or optimization**

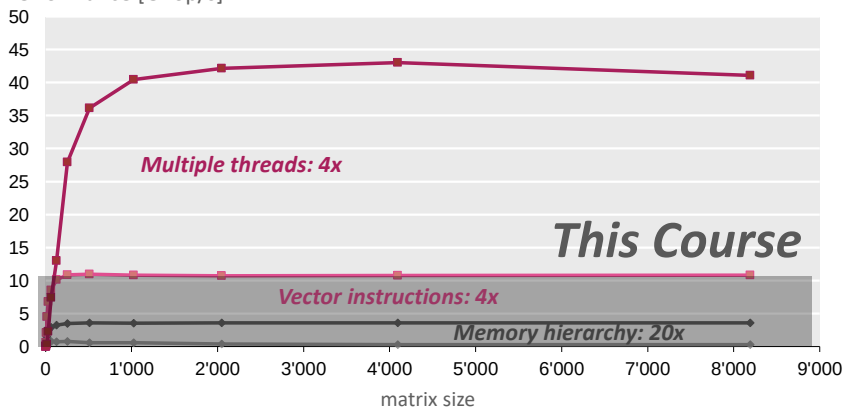
- **Research efforts**
  - Linear algebra: ***Phipac/ATLAS***, LAPACK, ***Sparsity/Bebop/OSKI***, Flame
  - Tensor computations
  - PDE/finite elements: Fenics
  - Adaptive sorting
  - ***Fourier transform: FFTW***
  - Linear transforms: Spiral
  - …others
  - New compiler techniques




***Proceedings of the IEEE special issue, Feb. 2005***

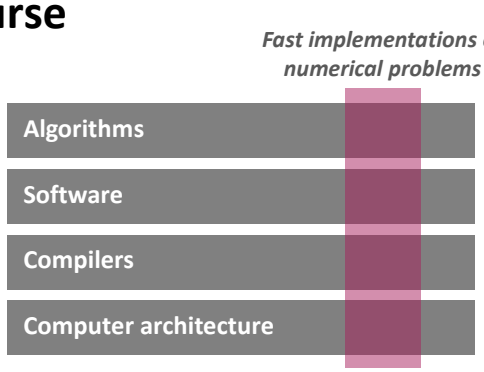
***Promising new area but much more work needed …***

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




31

# This Course




- **Obtain an understanding of performance (runtime)**
- **Learn how to write *fast code* for numerical problems**
  - Focus: Memory hierarchy and vector instructions
  - Principles studied using important examples
  - ***Applied in homeworks and a semester-long research project***
- **Learn about autotuning**

32

# Today

- Motivation for this course
- **Organization of this course**



# Course: Times and Places

- **Lectures:**
  - Monday 10-12, HG D3.2
  - Thursday 9-10, CAB G51
- **Recitations:**
  - Wednesday 13-15, HG D3.2
- **New course drop-out rule:** ***Deadline April 15th***

# About this Course

- **Team**
  - Me
  - TAs: Alen Stojanov Georg Ofenbeck Gagandeep Singh



- **Office hours: to be determined**
- **Course website has *ALL* information**
- **Questions: fastcode@lists.inf.ethz.ch**
- **Finding project partner: fastcode-forum@lists.inf.ethz.ch**



# About this Course (cont'd)

- **Requirements**
  - solid C programming skills
  - matrix algebra
  - Master student or above
- **Grading**
  - 40% research project
  - 25% midterm exam
  - 35% homework
- **Wednesday slot**
  - Gives you scheduled time to work together
  - Occasionally I will move lecture there (check course website)
  - By default will not take place

# Research Project: Overview

- **Teams of 4**
- **Yes: 4**
- ***Topic:*** **Very fast implementation of a numerical problem**
- ***Until March 4th:***
  - *find a project team*
  - suggest to me a problem or I give you a problem
    *Tip: pick something from your research or that you are interested in*
  - *Register on project website + you get svn access*
- **Show "milestones" during semester**
- **One-on-one meetings**
- **Give short presentation end of semester**
- **Write 6 page standard conference paper (template will be provided)**
- **Submit final code (early semester break)**



# 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"
- **In the beginning all of you are registered to that list**
- **Once team is formed (with or without project fixed) inform head TA, you will get deregistered from the list**

# Finding Project

- **Pick something you are interested in**
- **Ok if prior code exists**
- **Nothing that is dominated by**
  - dense linear algebra computations (matrix-matrix mult, solving linear systems, Cholesky factorization etc.)
  - fast Fourier transform
- **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***



# Organize Project

- **Work as a team**
- **Start *asap* with a team meeting**
- **Keep communicating *regularly* during semester**
- **Be fair to your team members**
- **Being able to work as a team is part of the exercise**
- **Be a team player**

# Research Project: Possible Failures

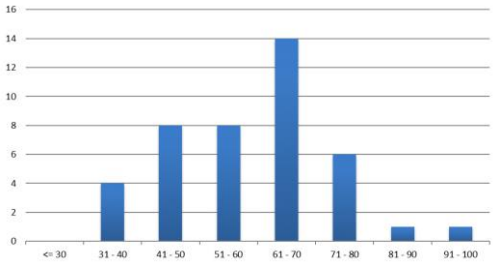
- **Don't do this:**
  - never meet
  - not respond to emails
  - "I don't have time right to work on this project in the next few months, why don't you start and I catch up later"
  - "I have a paper deadline in 1 month, cannot do anything else right now"
  - **while** not desparate(project-partners) **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" …



# Midterm Exam

- **Covers first part of course**
- **Will fix time soon**
- **No substitute date**
- ***There is no final exam***

*Point distribution 2016*

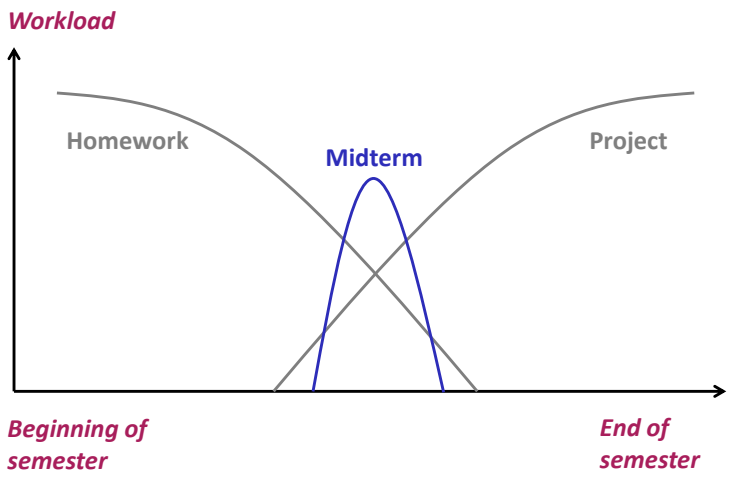


# Homework

- **Done individually**
- ***Solving homeworks analogous to homeworks in prior years is no guarantee for full points***
- **Exercises on algorithm/performance analysis**
- **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)
  - Some templates will be provided
- **Homework is scheduled to leave time for research project**
- **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)



# Workload During Semester



# 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
  - Ok to discuss things – but then you have to do it alone
- **Code may be checked with tools**
- ***Don't do copy-paste***
  - code
  - ANY text
  - pictures
  - especially not from Wikipedia



# Background Material

- **See course website**
- **Prior versions of this course: see website**
- **I post all slides, notes, etc. on the course website**

# Class Participation

- **I'll start on time**
- **It is important to attend**
  - Most things I'll teach are not in books
  - I'll use part slides part blackboard
- **Do ask questions**
- **We like when people come to office hours**

- **I you drop the course, please unregister from edoz**



# Course Evaluation 2015

*Master students CS*