Minds open...

... Laptops closed

slide by Bertrand Meyer
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)

- **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

---

### Cloud Computing (Server Farms)

**Application areas:**
- Data analytics
- Machine learning
- Database operations
- Others

**Methods:**
- Linear algebra
- Convolutions
- Tensor operations
- Others
Consumer Computing (Desktop, Phone, ...)

Methods:
- Linear algebra
- Transforms
- Filters
- Others

Photo/video processing
Audio decoding
Security

Image compression

Embedded/Edge Computing (Low-Power Processors)

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

Sensor networks
Cars
Robotics

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

How Hard Is It to Get Fast Code?

Algorithms
Software
Compilers
Microarchitecture

“compute Fourier transform”
“fast Fourier transform”
O(nlog(n)) or 4nlog(n) + 3n
e.g., a C function
optimized executable
high runtime performance

How well does this work?
The Problem: Example 1

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

Runtime [s]

Performance [Gflop/s]
The Problem: Example 1

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

Straightforward “good” C code (1 KB)

or

The Problem: Example 1

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

Straightforward “good” C code (1 KB)
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 (2n³)
<table>
<thead>
<tr>
<th>Model predictive control</th>
<th>Singular-value decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenvalues</td>
<td>Mean shift algorithm for segmentation</td>
</tr>
<tr>
<td>LU factorization</td>
<td>Stencil computations</td>
</tr>
<tr>
<td>Optimal binary search organization</td>
<td>Displacement based algorithms</td>
</tr>
<tr>
<td>Image color conversions</td>
<td>Motion estimation</td>
</tr>
<tr>
<td>Image geometry transformations</td>
<td>Multiresolution classifier</td>
</tr>
<tr>
<td>Enclosing ball of points</td>
<td>Kalman filter</td>
</tr>
<tr>
<td>Metropolis algorithm, Monte Carlo</td>
<td>Object detection</td>
</tr>
<tr>
<td>Seam carving</td>
<td>IIR filters</td>
</tr>
<tr>
<td>SURF feature detection</td>
<td>Arithmetic for large numbers</td>
</tr>
<tr>
<td>Submodular function optimization</td>
<td>Optimal binary search organization</td>
</tr>
<tr>
<td>Graph cuts, Edmond-Karps Algorithm</td>
<td>Software defined radio</td>
</tr>
<tr>
<td>Gaussian filter</td>
<td>Shortest path problem</td>
</tr>
<tr>
<td>Black Scholes option pricing</td>
<td>Feature set for biomedical imaging</td>
</tr>
<tr>
<td>Disparity map refinement</td>
<td>Biometrics identification</td>
</tr>
</tbody>
</table>

**“Theorem:”**

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

\[
\frac{\text{Performance of optimal implementation of } f}{\text{Performance of straightforward implementation of } f} \approx 10 \text{–} 100
\]
Evolution of Processors (Intel)

Double Float Peak Performance [Gflop/s]

Multiple cores
Vector units

CPU frequency [GHz]


3380 Gflop/s

48 cores
8-way SIMD

times of free exponential speedup
parallelism: work required

Evolution of Processors (Intel)

Double Float Peak Performance [Gflop/s]

increasing gap

mem bandwidth [normalized starting point]


Performance more and more determined by data movement
And there is Processor Variety ...

Domain-specific (here: Tile)  
FPGA accelerators

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]

Matrix size

Compiler doesn’t do the job
Doing by hand: nightmare

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 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

---

Performance is different than other software quality features
Performance/Productivity Challenge

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

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

Promising area but much more work needed ...

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

Performance [Gflop/s]

Multiple threads: 4x

This course: Single core

Vector instructions: 4x

Memory hierarchy/code style: 20x
This Course: Goals

Obtain a deeper understanding of performance

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

Today

Motivation for this course

Organization of this course
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**

---

Course Website Has all Info

[https://acl.inf.ethz.ch/teaching/fastcode/](https://acl.inf.ethz.ch/teaching/fastcode/)

**Advanced Systems Lab - Spring 2024**

**Basic Information**
- **R2R**: Course description, prerequisites, goals, Integrity
- **FAQs**
- **Module on days of the First Lecture**
- **Course number**: 330-0627-8, 8 credits
- **Lectures**: M 10.15-12.30, HG F3, Tu 9.15-10.30, HG C3; occasional substitute lectures W 14.15-16.00, HG E5
- **The lectures are not streamed but recorded. A login link is different from scito and will be sent by email**
- **Instructor**: Markus Püschel (LAW 499.4, punchel at inf)
  - **Head TA**: Tommasso Poggiotti (TP)
  - **TAs**
    - Mohsen Lashheteran (ML)
    - Theodoros Theodoridis (TT)
    - Mikhail Khaliullin (MH)
    - Dominik Spyropoulos (DS)
    - Shane Zhu (SZ)
- **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 partners: fastcode-forum@lists.inf.ethz.ch (emails go to all students who have no partner yet and to Head TA)
  - **Office Hours**: TTh

**Time Line**

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

- **Fri 06.03**: Project team and project negotiated in the project system; start project anytime now
- **Th 09.03**: HW1 due
- **Th 14.03**: HW1 due
- **Th 26.03**: HW2 due
- **Th 19.04**: HW2 due
- **Wed 26.04**: Midterm
Team and Communication

Head TA: Tommaso Pegolotti

Other TAs:
- Mikhail Khalilov
- Hicham Leghettas
- Dionisios Spiliopoulos
- Theodoros Theodoridis
- Shien Zhu

Course website has ALL information

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

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

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
- Midterm is usually in that slot but uses more rooms
Research Project: Overview

Teams of 4
Yes: 4

*Topic*: Very fast implementation of a numerical problem

*Until March 8*:  
- find a project team  
- suggest to me a problem or pick from list (on course website posted this week)  
  *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 7 page standard conference paper (template on website)

Submit final code

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, and you will automatically be deregistered you from mailing list
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*

Organize Project

Work as a team

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

<table>
<thead>
<tr>
<th>Week of</th>
<th>Milestone Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>29.04.</td>
<td>1st one-on-one project meeting (minimal milestone: base implementation, tested, performance plot, initial optimization plan)</td>
</tr>
<tr>
<td>20.05.</td>
<td>2nd one-on-one project meeting</td>
</tr>
<tr>
<td>03.06.</td>
<td>Project presentations</td>
</tr>
<tr>
<td>Fr 21.06.</td>
<td>Project report due</td>
</tr>
</tbody>
</table>

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
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

Date: Wed, April 24th

No substitute date

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)
**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

---

**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
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*

Feedback 2023

How technically demanding in comparison with other courses

Workload in comparison with other courses

Overall satisfaction with course
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