Organization

About myself

Philipp Schuster

BSc and MSc in Koblenz in Computer Visualistics.

PhD in Tübingen in Programming Languages.

PostDoc in Tübingen in Software Engineering.

Teaching Specialist for Practical Computer Science.

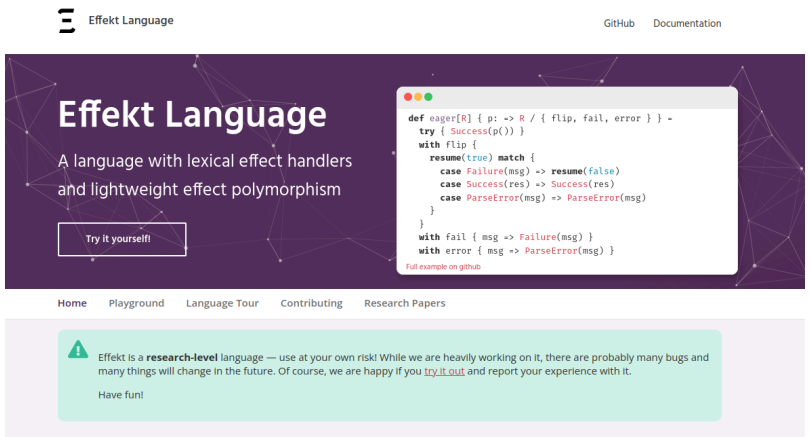
Started in May, this is my first lecture.

Research Interest

All programming languages.

Compilers for languages with non-sequential control flow.

Backend developer on Effekt.



Lecture

Two lectures per week: Tuesday 14:00 and Friday 14:00.

One theoretical topic per week.

One practical technology per week.

Mix of slides and live coding.

No script, do take notes!

Lab

One lab per week: Tuesday 12:00.

Practice last week's topic and technology.

Same exercise sheet for homework and lab.

Homework is not checked.

Use ChatGPT, Claude, etc. to prepare for lab.

Everyone has to have live coded in lab twice to be admitted to the exam.

Forum

https://ps-forum.cs.uni-tuebingen.de/invites/7oBoGCSEBF

Our central communication platform. No Emails!

Distribution of exercise sheets.

Questions about lecture content and administration.

Exam

Code reading and code writing tasks.

Multiple choice questions to test your knowledge.

Main exam: 10.02.25, 14:00

Make-up exam: 24.03.25, 14:00

This course

Parallel, concurrent, and distributed *programming*.

Hands-on course in practical computer science.

Focus on what works, not on what doesn't.

Advanced programming in multiple languages.

Broad overview over many concepts.

Parallel, Concurrent, Distributed

They are different!

Parallelism

Same result but faster!

Examples:

- Assembly lines
- Audio encoding
- Graphics rendering
- Neural network inference

Must involve multiple processors.

Concurrency

Events with unknowable order!

Examples:

- Traffic
- Central database
- Graphical user interface
- Web server

May involve only a single processor.

Distribution

Running in different locations!

Examples for distributed parallelism:

- Search engines
- Neural network training
- Weather simulation

Examples for distributed concurrency:

- Chat software
- Multiplayer games
- Collaborative editing

Unreliable communication.

Throughput versus Latency

Throughput: items per time.

Latency: time per item.

The two might be in conflict.

Parallelism: often we want to optimize througput.

Concurrency: often we want to optimize latency.

Distribution: often we need to balance both.

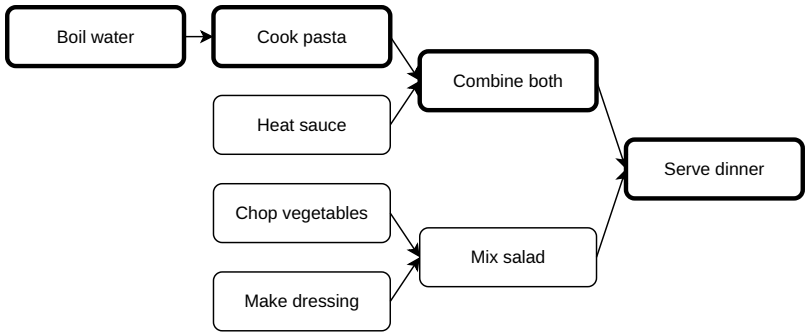
Work versus Span

Work: total duration of operations.

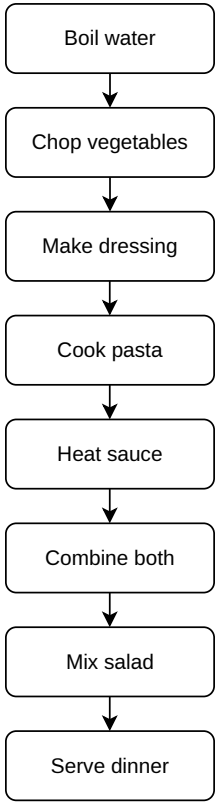
Span: duration of critical path.

Speedup: Work divided by Span.

Parallel:



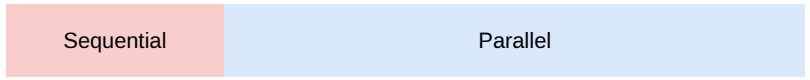
Sequential:



Amdahl's Law

> [...] the effort expended on achieving high parallel processing rates is wasted unless it is accompanied by achievements in sequential processing rates of very nearly the same magnitude.

Gene M. Amdahl. 1967. Validity of the single processor approach to achieving large scale computing capabilities.



Speedup = Work / (Sequential + Parallel / Processors)

Gustafson's Law

> [...] in practice, the problem size scales with the number of processors.

John L. Gustafson. 1988. Reevaluating Amdahl's law.

Examples: image resolution, weights in neural networks, number of websites.

Tools

Nix

Reproducible development environments.

https://nix.dev/install-nix

Enable nix-command and flakes.

In file `~/.config/nix/nix.conf`:

```
experimental-features = nix-command flakes
```

demo/flake.nix

Benchmarking

Statistics: report empirical mean and standard deviation over multiple runs.

Reproducibility: pin down hardware, operating system, compilers, libraries, …

Warmups: beware of power saving, thermal throttling, memory and file caching, just-in-time compilation, …

Baseline: compare against the fastest alternative.

Scaling: compare on different input sizes.

Benchmarking Tools

Chronos

For quick iteration.

```
chronos './main1' './main2' './main3'
```

Hyperfine

For reportable results.

```
hyperfine -w 10 --export-csv results.csv './main1' './main2' './main3'
```

demo/main1.c

Summary and Outlook

Parallelism and Concurrency are different.

https://inside.java/2021/11/30/on-parallelism-and-concurrency/

Distribution can be about both.

Do not guess performance, measure it.

Next week: regular data parallelism.