Next-Gen Programming Interfaces and Compilers

Seminar Kick-off

Alexis Engelke Michael Petter Josef Weidendorfer

Chair of Data Science and Engineering (I25)
School of Computation, Information, and Technology
Technical University of Munich

2025-10-14

Organization

Kick-off meeting	2025-10-14
Literature research + derive structure	
Discuss paper outline	at latest 2025-11-05
Write and discuss paper draft	at latest 2025-11-19
Complete paper submission	2025-12-03
Peer-review two other papers	2025-12-17
Incorporate feedback from peers and advisor Shepherding: propose changes and discuss with advisor	
Final submission of paper/slides	≈2026-01-18
Presentations	≈2026-01-19/20

Topics Today

- Literature and sources
  - Finding literature and citable sources/references
- Writing a (seminar) paper
  - Structure, style, citing
- Presentation techniques
  - Structure, slide design, presentation style

Citable Literature

Good to use

- Books, book chapters
- Papers (conf./journal)
- Published articles
- Manuals
- Websites with identifiable author
  (cite with URL+access date)

Try to avoid

- Secondary Literature
- Wikipedia
- Facebook, etc.
- Advertisements
- Lecture slides
- Source code

Finding literature

- Starting points: IEEExplore, ACM DL, Google Scholar, . . .
  - Select appropriate keywords
  - Many papers/books accessible freely via the university library
- Other starting point: your advisor
- Graph algorithms
  - Publications of the same author(s)
  - Publications at the same venue
  - Cites . . . (listed references)
  - Cited by . . .

Reading Literature

1. Read title — still relevant?
2. Read abstract — still relevant?
3. Skim introduction/contributions — still relevant?
   - Introduction sets framing
4. Skim through text, **examples and figures** — still interesting?
5. Read interesting sections

Managing Citations: BibTeX

- Keep your references in BibTeX files
- Also exportable from DBLP, Google Scholar, ACM, ...
  - Caution: might be wrong (esp. G.Sc.) or contain irrelevant data

```
@inproceedings{lattner2004llvm,
  title={{LLVM}: A compilation framework for
    lifelong program analysis \& transformation},
  author={Lattner, Chris and Adve, Vikram},
  booktitle={Proceedings of the International
    Symposium on Code Generation and Optimization},
  series={CGO '04}
  pages={75--86},
  year={2004},
}
```

Writing: Goals

- What should the reader learn from your paper? $\Rightarrow$ provide value!
  - #annually published papers grows – attention span of readers doesn't

$\Rightarrow$ Clearly define contributions/key information of paper

$\Rightarrow$ Optimize text for reader

  - Make it easy to decide whether the paper is worth reading
  - Make relevant information easy to find and correctly extract
  - Make text concise and easy to understand
  - Provide helpful and easy-to-understand figures/examples

- Bad writing: reader doesn't understand, re-reads, gets angry, stops reading, (reviewer: rejects paper; examiner: gives bad grade)

Paper Structure

- Abstract: Brief summary of area, problem, approach, key result
- Introduction: introduce area, problem, approach, key results, contributions, outline
- Background: if needed, describe prerequisites
- Main part (approach, evaluation, discussion, etc.)
- (*In a paper:* Related Work – *might come before main part*)
- Summary & outlook

Writing

- Text writing $\approx$ linearizing knowledge graph
  - Keep closely related topics together
  - Avoid forward references or at least keep distance small
- Outline (story) for introduction helps focusing on important parts
- Write for readers with different backgrounds
  - E.g., skippable background sections for non-experts, more context in introduction
  - Readers don’t know your idea, approach, etc.
- Repeat important points in: abstract, introduction, main part, summary
  - Readers tend to not read the complete paper

Sections, Figures, Tables

- Avoid walls of text
- Use (Sub-)Sections, paragraphs, bullet points to structure text
  - Allows reader to skip unimportant parts
  - No two headings without text in between

- Figures/tables/listings: self-explaining with caption (possibly multiple lines)
- Keep figures focused, simple, easy to understand, ...
- All figures/tables/listings must be referenced in text
  - Allows reader to put figure in context
- Caption goes below figures/listings, but above tables

Writing Style

- Factual, precise, concise, focused, clear, simple
  - Avoid wordy phrases, unnecessary information/words
  - Use simple words, avoid introducing too much terminology/abbreviations
  - But: some clearly defined terminology can be helpful
- Get to the point!
- Stay on topic, no story telling, . . .
- But: don't omit necessary prerequisites
- Make it easy for *the reader*

- Avoid *I*, prefer *we* (or passive voice)
- *We* only described the authors, not the reader
- Use formal English (e.g., can't $\rightarrow$ cannot)

Revising, Editing, Formatting

- Text won't be perfect on first attempt
- What can be misunderstood?
- Cut out unnecessary words

- Fix grammar, spelling, punctuation, typography
  - Difference between -/–/—; hyphenation, quotes, ...
- Keep format standard and consistent
  - Fonts, colors, emphasis, ...
- Use *italics* (`\emph`), rarely **bold**, never <u>underline</u>

Three LaTeX mistakes that people should stop making?

1. Worrying too much about formatting and not enough about content.
2. Worrying too much about formatting and not enough about content.
3. Worrying too much about formatting and not enough about content.

– Leslie Lamport, 2000[1]

[1] LL Lamport. "How (La)TeX changed the face of Mathematics". In: *DMV-Mitteilungen* 1 (2000), pp. 49–51. .

Citing

- All work that is not yours **must** be cited
  - Clearly describe source
  - But: no wrong/inaccurate attributions
- Citing styles:
  - Literal (direct) quote
  - indirect quote (rephrase) ←strongly preferred
- Exception: foundations can be assumed (generally first few Bachelor semesters)

Citing: Examples

The x86 architecture defines the register CR2 [1].

```
The x86 architecture defines the register
CR2~\cite{intel2019man}.
```

The x86 architecture defines the register CR2. It can be used with the instruction MOV. [1]

```
The x86 architecture defines the register
CR2.  It can be used with the instruction
MOV.~\cite{intel2019man}
```

(paragraph)

Valgrind [1] is a tool for run-time instrumentation.

```
Valgrind~\cite{nethercote2007} is a tool
for run-time instrumentation.
```

Other approaches [1,2,3] . . .

```
Other approaches~\cite{foo,bar,baz} \dots
```

Presentation: Content Selection

Presentation for the **audience**!

- What do you want the audience to take away?
  (Not: what can I talk about!)
- What are the key points?
- How much content fits into the time slot?

Structure

- Motivation
  - Why is the topic relevant?
- Background
  - Consider referencing information from previous talks
- Concept
- Evaluation
  - How good is the described concept?
- Conclusions and outlook

- Important: avoid forward references
- Restrict to important details
- Use good/helpful examples

Media

- Slides (Beamer)
    - For use during the talk
    - Good to prepare
    - *Backup slides* as preparation for questions
- Whiteboard, blackboard
    - Permanently needed information
    - Answering questions
- Hardware, demonstrators, etc.

- Check possibilities in advance

Before the Talk

- Prepare slides, etc.
- Do a dry-run
  - Always recommended
  - Helps with uncertainity and time estimation
- Prepare on-site
  - Laptop, Beamer, laser pointer, clock, etc.

Talking Style

- Speak freely
- Don't go too fast/slow
- Stay in contact with the audience
  - Eye contact, position, etc.
- Usually at least 1 minute per slide
- Stay in time limit
  - Optional slides can fill time
  - Regularly consult a watch
- **Stay calm**

Slides: Content

- One topic per slide
- Avoid text
  - ≤ 8 lines
- Prefer graphics/illustrations
- No unused points
  - Cover everything on the slides in your talk

Slides: Content

- Title page
  - Title, name, institution, date, location
- On every other slide: number and title
- Conclusion
  - Summarize important points people should remember
  - No “Thank you” / “Questions” slide

Slides: Colors

- **Black on white**
- **Black on white**

- Sufficient contrast
- Use colors sparingly, but systematically
- Be careful with gradients
- No annoying backgrounds (wave textures, etc.)

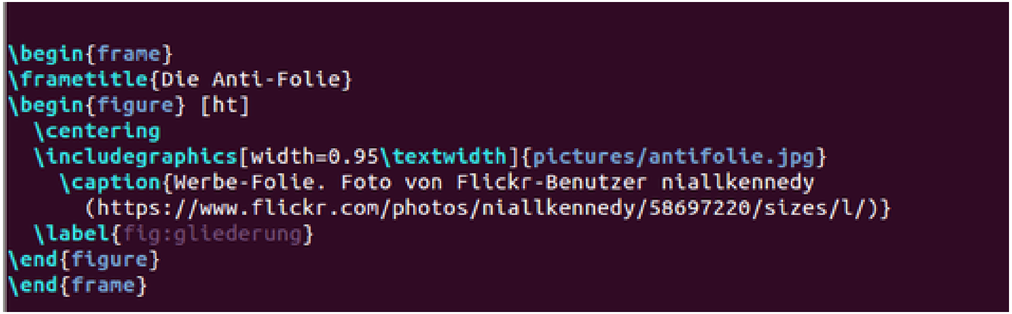
- **Anomations only with sufficiently added value**

Slides: Text and Graphics

- Double-check text for typos, etc.
- Use a readable, sans-serif font
- Prefer vector graphics (or images with a high resolution)
- Avoid screenshots/scans
- Citations: if critical, use footnote
  - No end notes and [12]-style references

- Listings only with a sufficiently large value

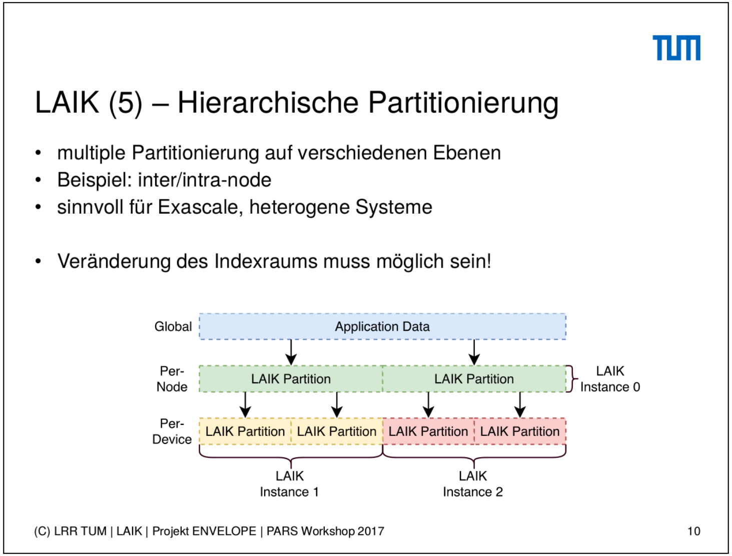
Negative Example



```
\begin{frame}
\frametitle{Die Anti-Folie}
\begin{figure} [ht]
  \centering
  \includegraphics[width=0.95\textwidth]{pictures/antifolie.jpg}
    \caption{Werbe-Folie. Foto von Flickr-Benutzer niallkennedy
      (https://www.flickr.com/photos/niallkennedy/58697220/sizes/l/)}
  \label{fig:gliederung}
\end{figure}
\end{frame}
```

Figure: Screenshot of code with insufficient resultion

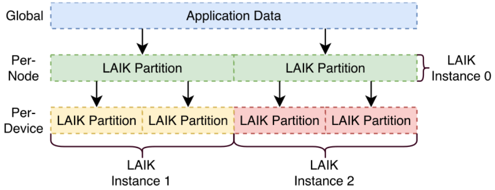
Positive Example (?)



LAIK (5) – Hierarchische Partitionierung

- multiple Partitionierung auf verschiedenen Ebenen
- Beispiel: inter/intra-node
- sinnvoll für Exascale, heterogene Systeme

- Veränderung des Indexraums muss möglich sein!



(C) LRR TUM | LAIK | Projekt ENVELOPE | PARS Workshop 2017 10

Positive Example (?)

```
#include "laik-backend-mpi.h"
int main(int argc, char* argv[])
{
Laik_Instance* inst = laik_init_mpi(&argc,&argv);
Laik_Group* world = laik_world(inst);

// allocate global 1d double (8 bytes) array: 1 mio entries
Laik_Data* a = laik_alloc_1d(world, 8, 1000000);

// initialize at master (others do nothing)
laik_set_new_partitioning(a, LAIK_PT_Master, LAIK_AP_WriteOnly);
double* base; uint64_t count;
laik_map(a, LAIK_DL_CANONICAL, (void**) &base, &count);
for(uint64_t i = 0; i < count; i++) base[i] = (double) i;
}
```

Figure: Example for showing source code

Positive Example (?)

```
#include "laik-backend-mpi.h"
int main(int argc, char* argv[]) {
  Laik_Instance* inst = laik_init_mpi(&argc, &argv);
  Laik_Group* world = laik_world(inst);

  // allocate global 1d double array: 1 mio entries
  Laik_Data* a = laik_alloc_1d(world, 8, 1000000);
  // initialize at master (others do nothing)
  laik_set_new_partitioning(a, LAIK_PT_Master, LAIK_AP_WriteOnly);
  double* base;
  uint64_t count;
  laik_map(a, LAIK_DL_CANONICAL, (void**)&base, &count);
  for (uint64_t i = 0; i < count; i++)
    base[i] = (double) i;
}
```

Figure: Example for showing source code

Summary

- Bring your point to the audience – written or spoken
- Good literature as starting point
- Logical structure for paper and presentation
- Make it easy for audience to get information
- Presentation: good preparation is important

- Chance to learn ☺