

Algorithmen II

Peter Sanders, Timo Bingmann

Übungen:

Tobias Heuer, Sebastian Lamm

Institut für Theoretische Informatik

Web:

http://algo2.itl.kit.edu/AlgorithmenII_WS19.php

1 Algorithm Engineering

A detailed definition

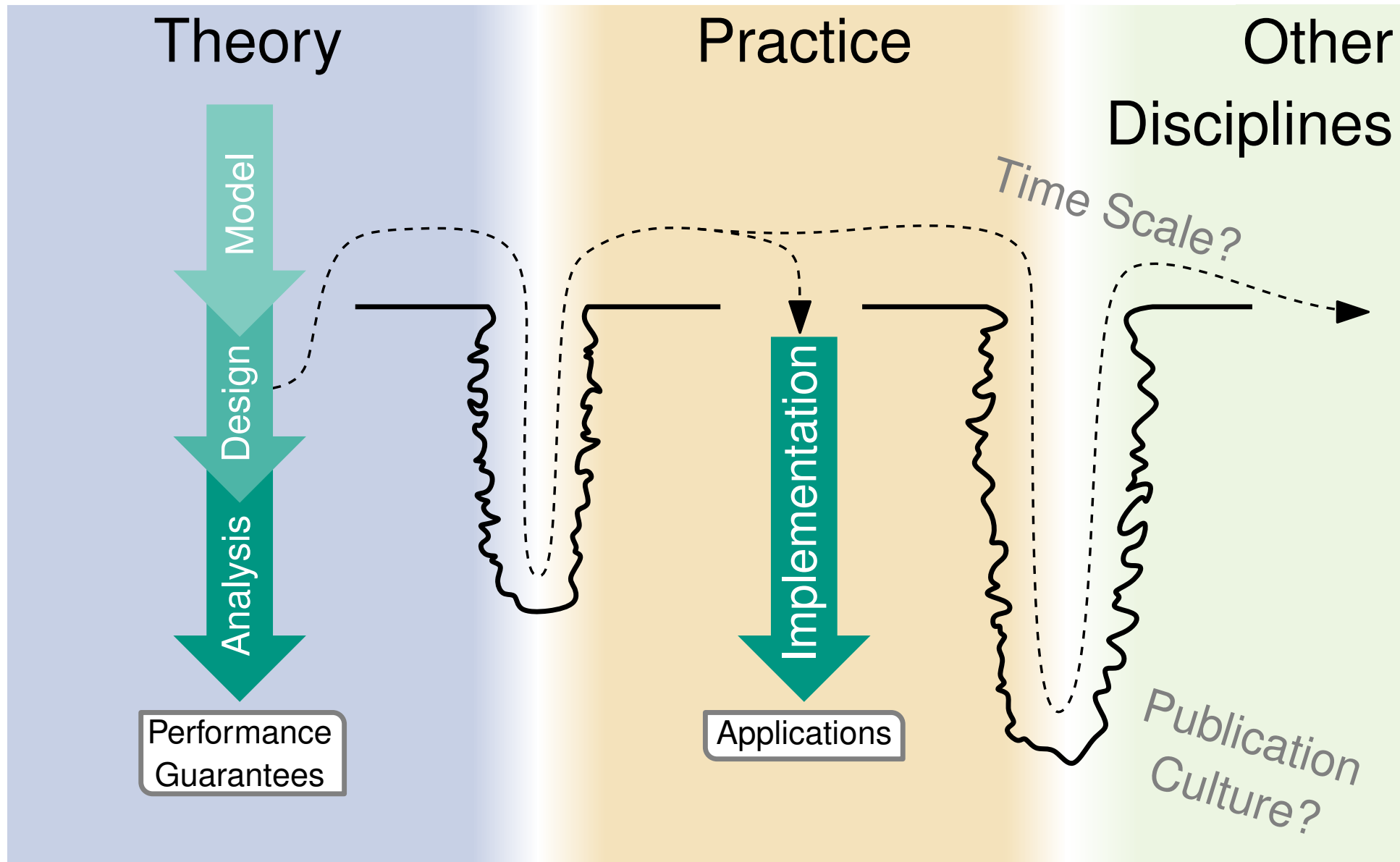
- in general

 - [with Kurt Mehlhorn, Rolf Möhring, Petra Mutzel, Dorothea Wagner]



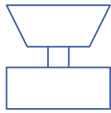

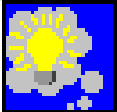



- A few examples, usually sorting

- A little bit on experimental methodology

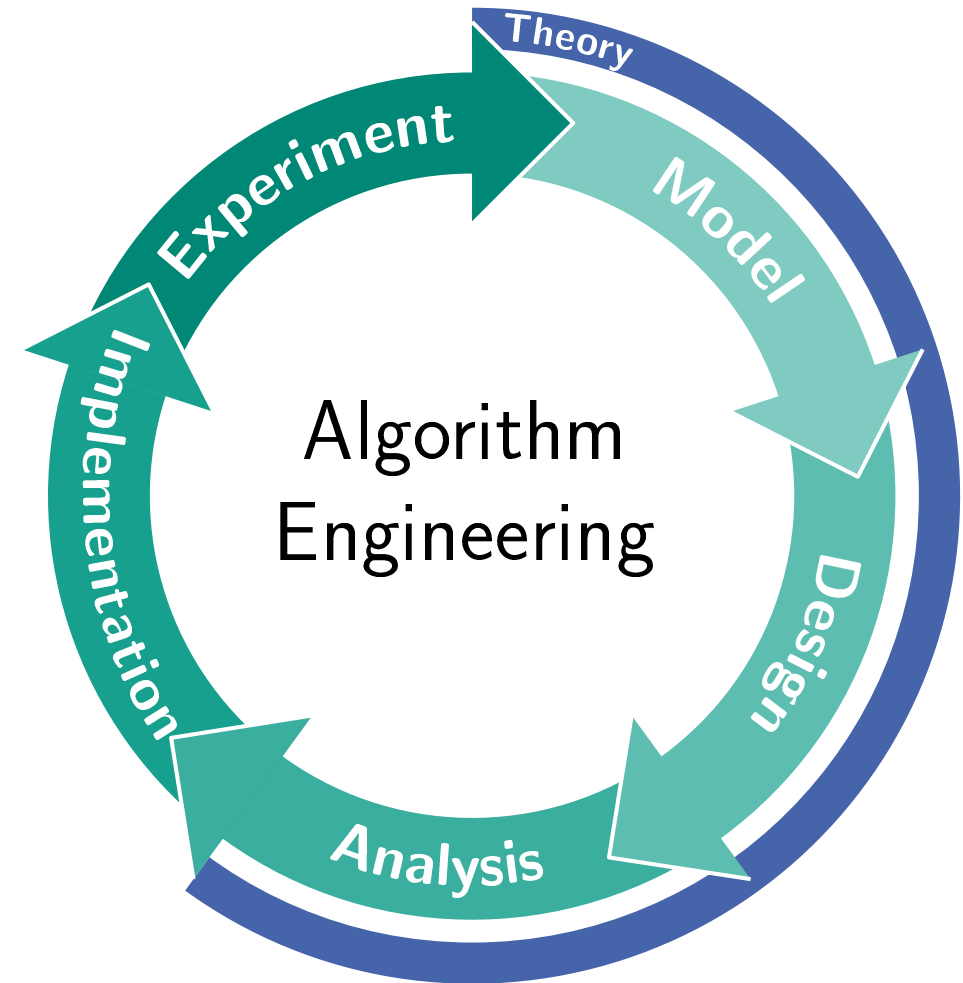
(Caricatured) Traditional View: Algorithm Theory



Gaps Between Theory & Practice

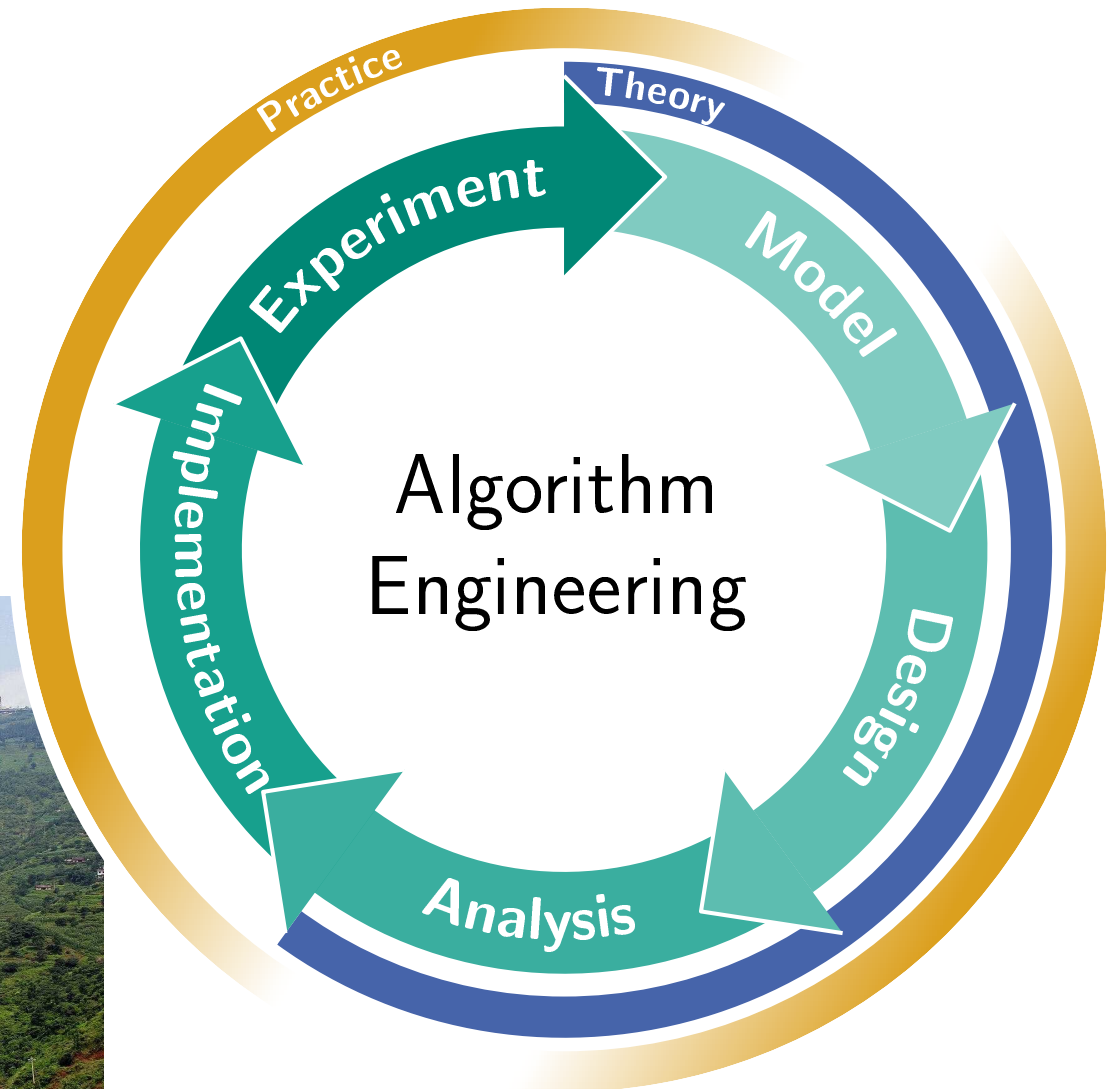
Theory		\longleftrightarrow	Practice	
simple		appl. model		complex
simple		machine model		real
complex		algorithms	<code>FOR</code>	simple
advanced		data structures		arrays,...
worst case	<code>max</code>	complexity measure		inputs
asympt.	<code>O(·)</code>	efficiency	<code>42%</code>	constant factors

Algorithmics as Algorithm Engineering



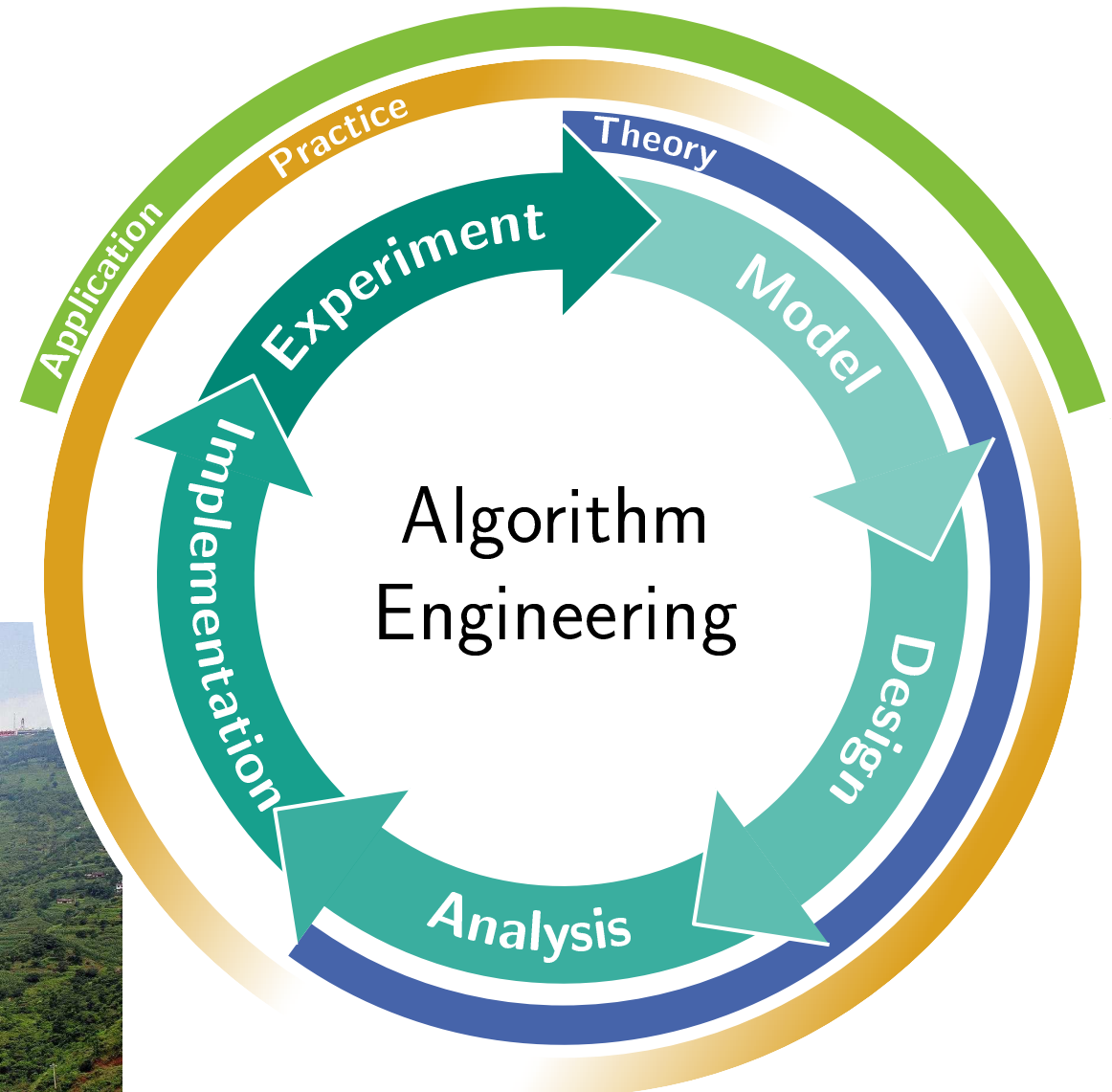
Algorithmics as Algorithm Engineering

- bridge gaps between theory and practice

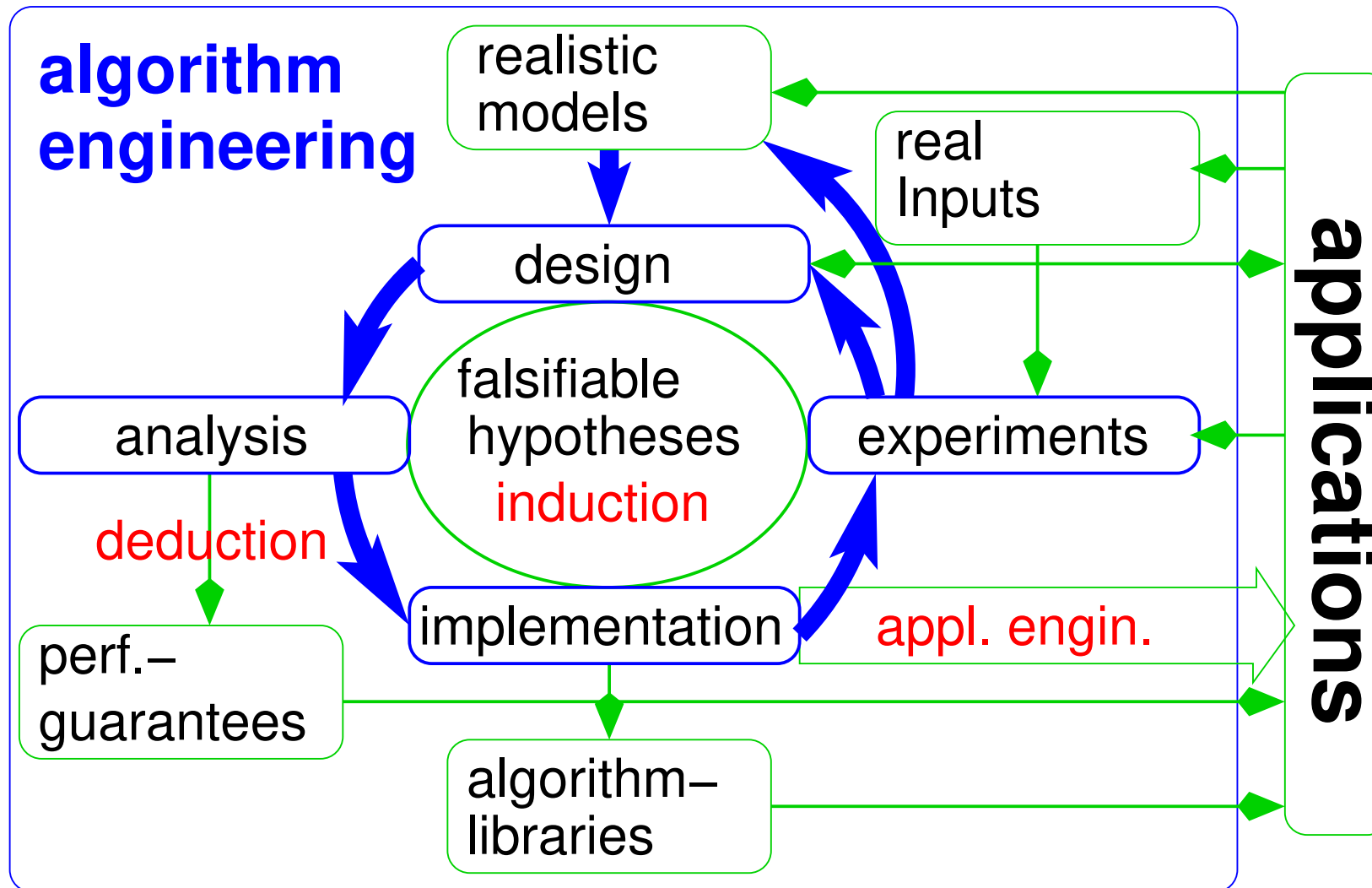


Algorithmics as Algorithm Engineering

- bridge gaps between theory and practice
- integrated interdisciplinary research



Algorithmics as Algorithm Engineering



Bits of History

1843– Algorithms in theory and practice

1950s, 1960s Still infancy

1970s, 1980s Paper and pencil algorithm theory.

Exceptions exist, e.g., [D. Johnson], [J. Bentley]

1986 Term used by [T. Beth],

lecture “**Algorithmentechnik**” in Karlsruhe.

1988– Library of Efficient Data Types and Algorithms
(LEDA) [K. Mehlhorn]

1997– **Workshop on Algorithm Engineering**

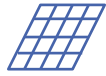

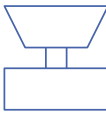

↪ ESA applied track [G. Italiano]

1997 Term used in US policy paper [Aho, Johnson, Karp, et. al]

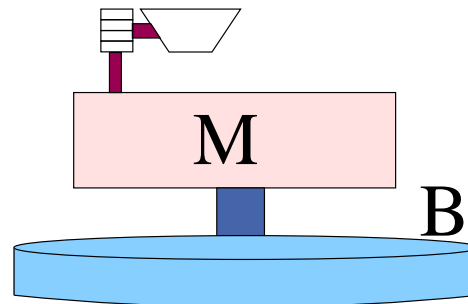
1998 **Alex** workshop in Italy ↪ **ALENEX**



Realistic Models

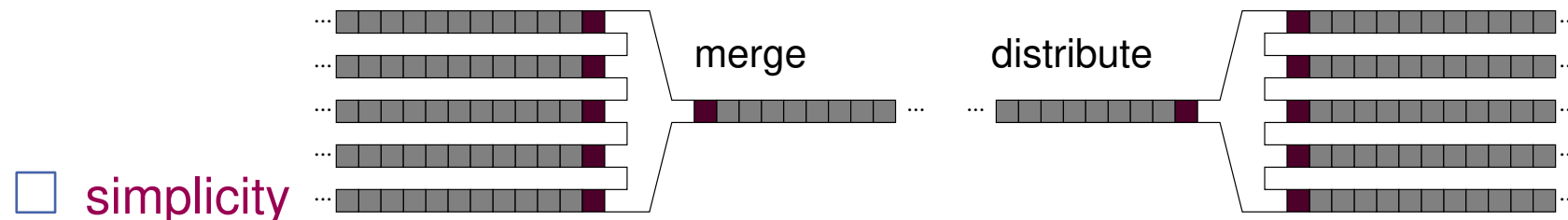
Theory	\longleftrightarrow	Practice
simple 	appl. model	 complex
simple 	machine model	 real

- Careful refinements
- Try to preserve (partial) analyzability / simple results



Design

of algorithms that work well in **practice**



- simplicity
- reuse
- constant factors
- exploit **easy** instances

Analysis

- Constant factors** matter
Beispiel: quicksort
- Beyond worst case** analysis
- Practical algorithms** might be difficult to analyze
(randomization, meta heuristics, . . .)

Implementation

sanity check for algorithms !

Challenges

Semantic gaps:

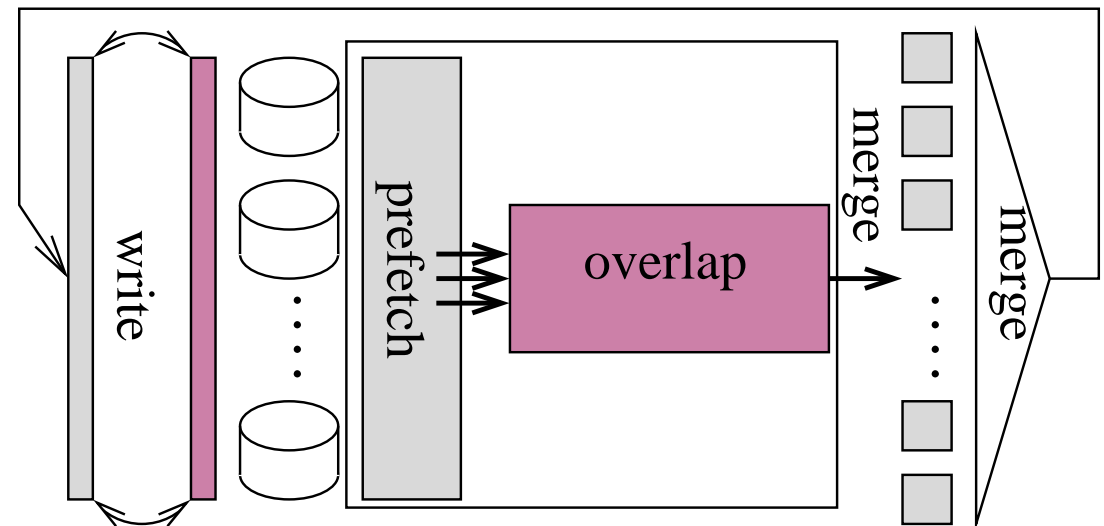
Abstract algorithm

↔

C++...

↔

hardware



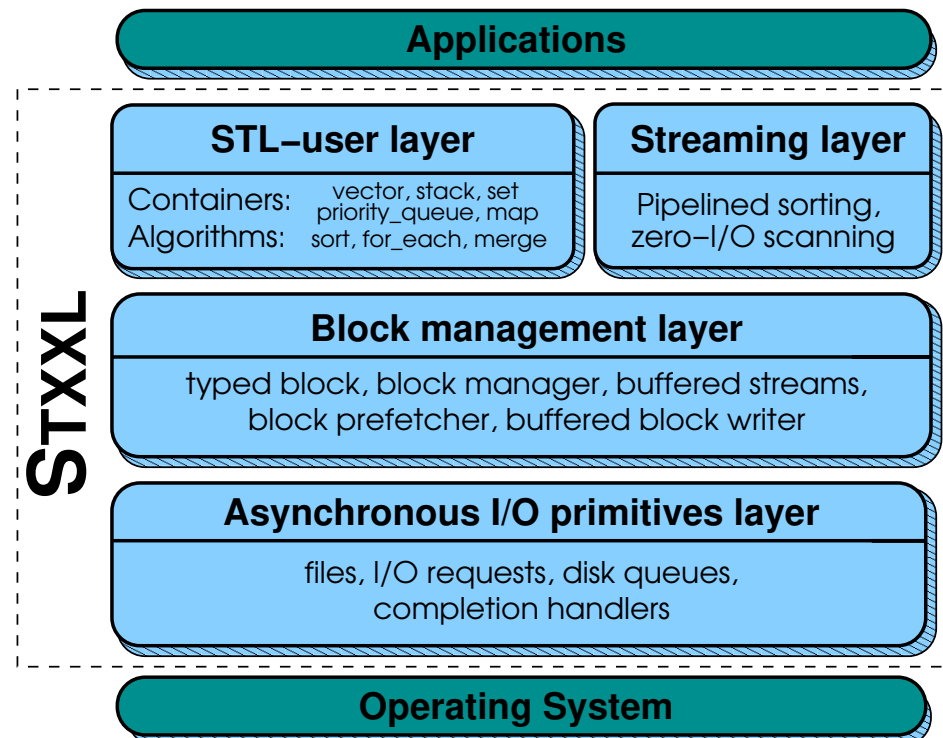
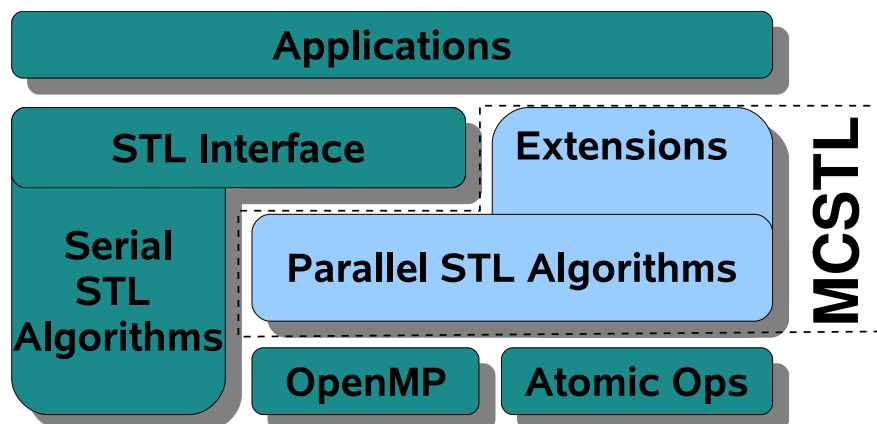
Experiments

- sometimes a good **surrogate** for analysis
- too much** rather than too little **output data**
- reproducibility** (10 years!)
- software engineering**

Stay tuned.

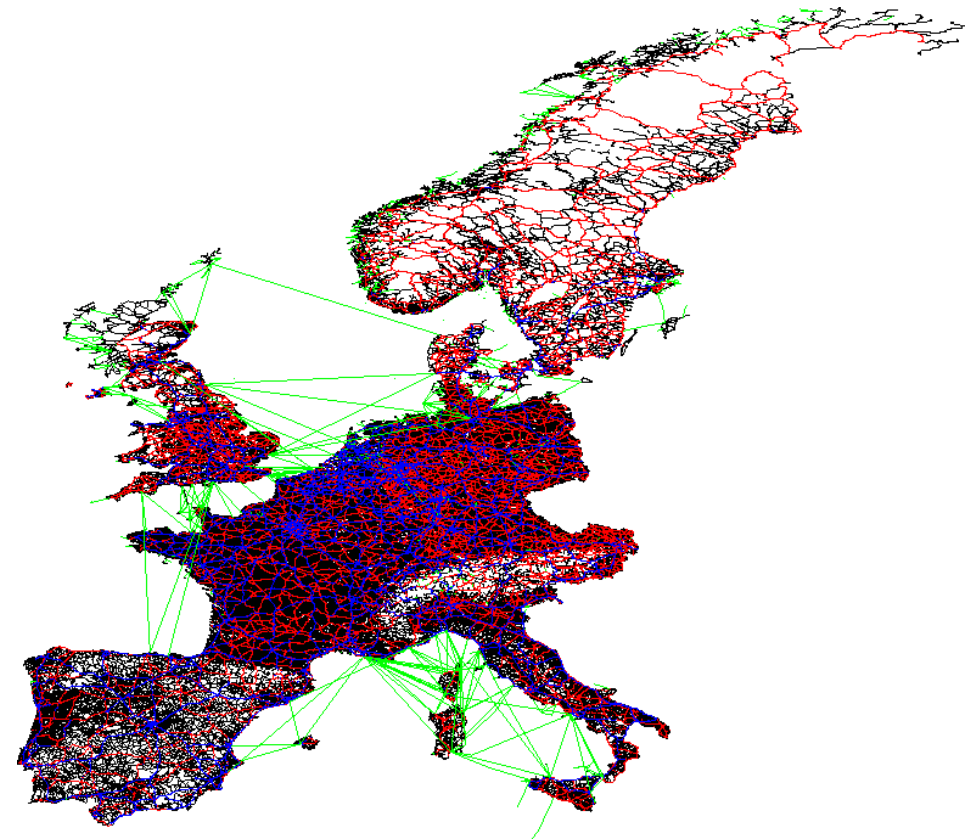
Algorithm Libraries — Challenges

- software engineering
- standardization, e.g. java.util, C++ STL and BOOST
- performance ↔ generality ↔ simplicity
- applications are a priori unknown
- result checking, verification



Problem Instances

Benchmark instances are essential for development of practical algorithms



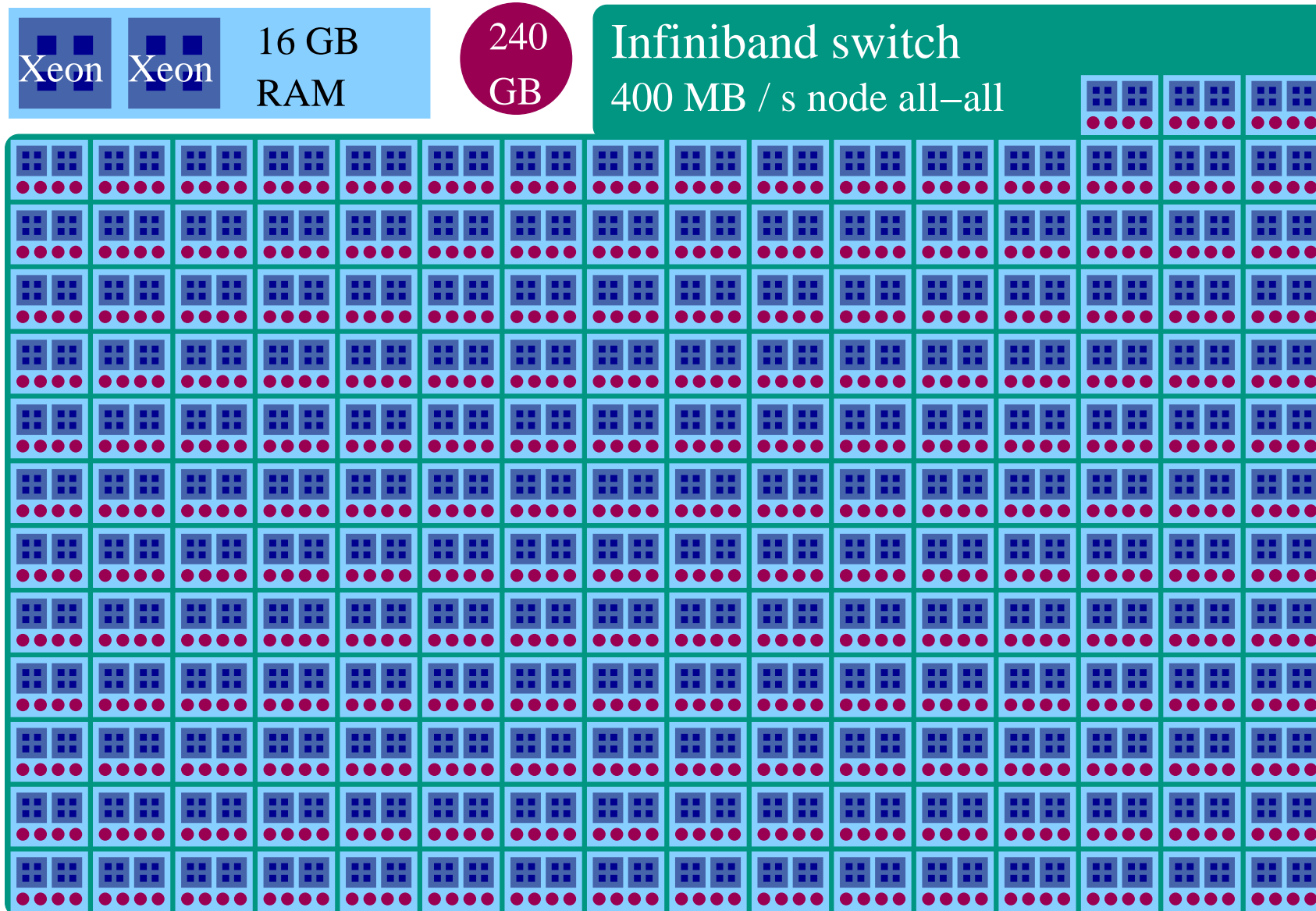
Example: Sorting Benchmark (Indy)

100 byte records, 10 byte random keys, with file I/O

Category	data volume	performance	improvement
GraySort	100 000 GB	564 GB / min	17×
MinuteSort	955 GB	955 GB / min	> 10×
JouleSort	100 000 GB	3 400 Recs/Joule	???
JouleSort	1 000 GB	17 500 Recs/Joule	5.1×
JouleSort	100 GB	39 800 Recs/Joule	3.4×
JouleSort	10 GB	43 500 Recs/Joule	5.7×

Also: PennySort

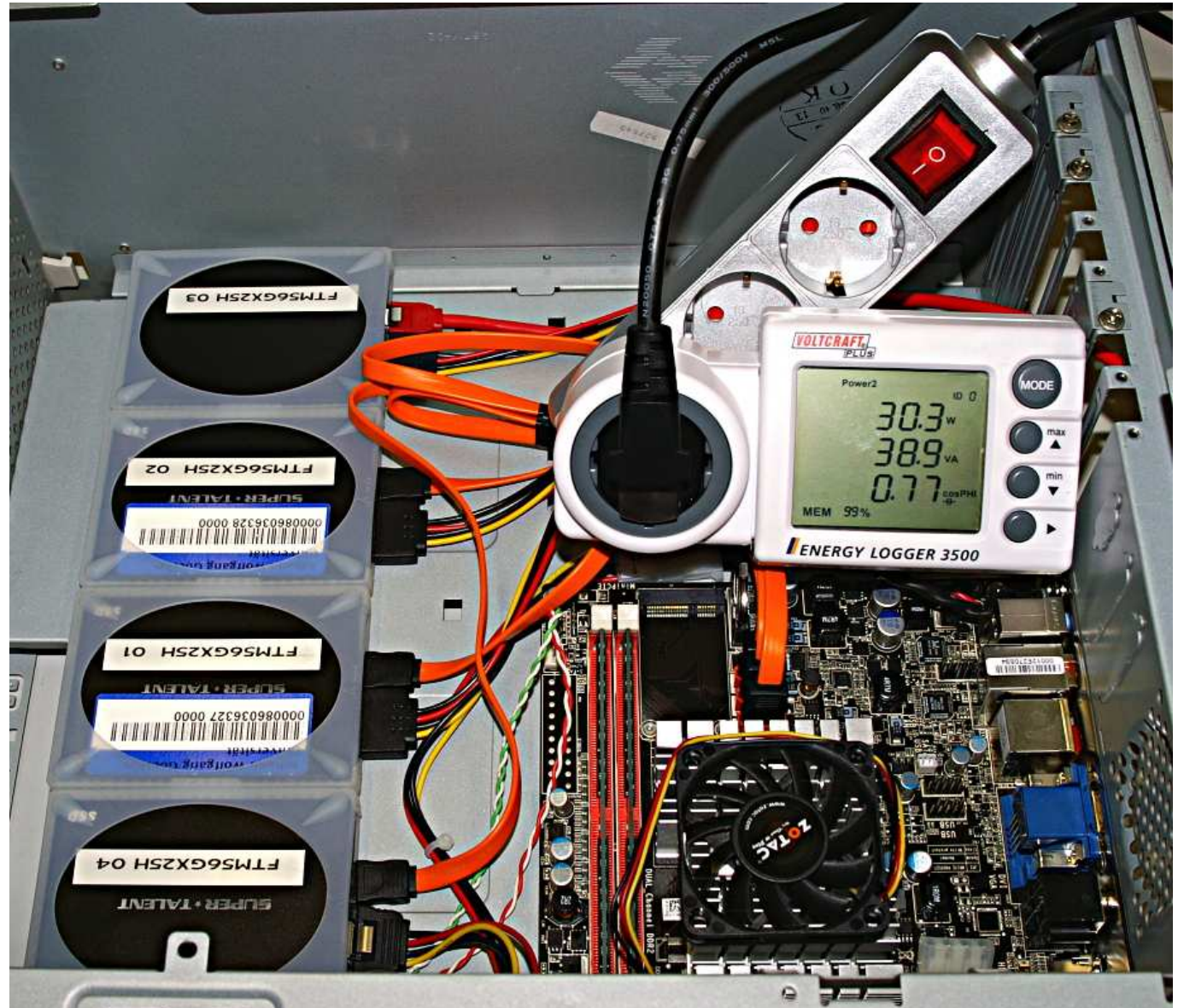
GraySort: *inplace* multiway mergesort, *exact* splitting



JouleSort

- Intel Atom N330
- 4 GB RAM
- 4 × 256 GB
SSD (SuperTalent)

Algorithm similar to
GraySort



Applications that “Change the World”

Algorithmics has the potential to SHAPE applications
(not just the other way round)

[G. Myers]

Bioinformatics: sequencing, proteomics, phylogenetic trees,...



Information Retrieval: Searching, ranking,...

Traffic Planning: navigation, flow optimization,
adaptive toll, disruption management

Geographic Information Systems: agriculture, environmental
protection, disaster management, tourism,...

Communication Networks: mobile, P2P, cloud, selfish users,...

Conclusion:

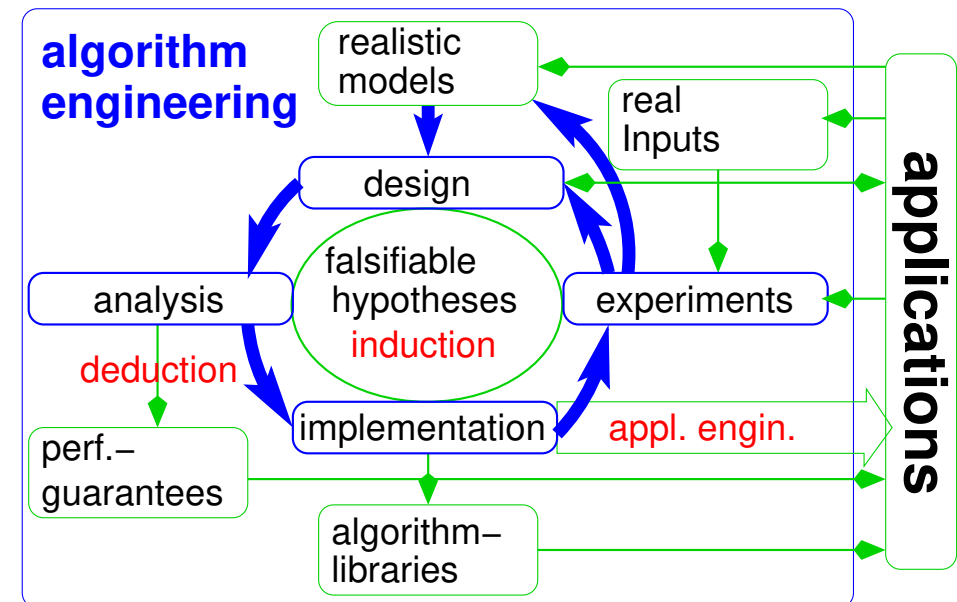
Algorithm Engineering \leftrightarrow Algorithm Theory

- algorithm engineering is a wider view on algorithmics
(but no revolution. None of the ingredients is really new)
- rich methodology
- better coupling to applications
- experimental algorithmics \ll algorithm engineering
- algorithm theory \subset algorithm engineering
- sometimes different theoretical questions
- algorithm theory may still yield the strongest, deepest and most persistent results within algorithm engineering

More On Experimental Methodology

Scientific Method:

- Experiment need a possible outcome that **falsifies** a hypothesis
- Reproducible
 - keep data/code for at least 10 years
 - + documentation (aka laboratory journal (Laborbuch))
- clear and detailed description in papers / TRs
- share instances and code



Quality Criteria

- Beat the state of the art, globally – (not your own toy codes or the toy codes used in your community!)
- Clearly** demonstrate this !
 - both codes use same data ideally from accepted benchmarks (not just your favorite data!)
 - comparable machines or fair (conservative) scaling
 - Avoid uncomparabilities like:
 - “Yeah we have worse quality but are twice as fast”
 - real world data wherever possible
 - as much **different**, **fresh** inputs as possible
 - its fine if you are better just on some (important) inputs

Not Here but Important

- describing the setup (machine, compiler, OS, instances, repetitions, . . .)
- finding sources of measurement errors
- reducing measurement errors (averaging, median, unloaded machine. . .)
- measurements in the **creative** phase of experimental algorithmics.

The Starting Point

- (Several) Algorithm(s)
- A few quantities to be measured: time, space, solution quality, comparisons, cache faults, . . . There may also be **measurement errors**.
- An unlimited number of potential inputs. \rightsquigarrow condense to a few characteristic ones (size, $|V|$, $|E|$, . . . or problem instances from applications)

Usually there is not a lack but an **abundance** of data \neq many other sciences

The Process

Waterfall model?

1. Design
2. Measurement
3. Interpretation

Perhaps the paper should at least look like that.

The Process

- Eventually stop asking questions (Advisors/Referees listen !)
- build measurement tools
- automate (re)measurements
- Choice of Experiments driven by risk and opportunity
- Distinguish mode

explorative: many different parameter settings, interactive, short turnaround times

consolidating: many large instances, standardized measurement conditions, batch mode, many machines

Of Risks and Opportunities

Example: Hypothesis = my algorithm is the best

big risk: untried main competitor

small risk: tuning of a subroutine that takes 20 % of the time.

big opportunity: use algorithm for a new application

~> new input instances