ERLANGEN REGIONAL COMPUTING CENTER

Systematic Node-Level Performance Engineering

Jan Treibig, Georg Hager

SPEC DevOps Meeting
Würzburg, 2015/02/20

The team in Erlangen

HPC@RRZE core staff

Prof. Dr. Gerhard Wellein
Team lead
Professor for
High Performance Computing

Dr. habil. Georg Hager
User support
Teaching
Research

Rasa Mabande
Team assistant

Dr. Thomas Zeiser
User support
Project management
System administration

Dipl.-Inf. Michael Meier
System administration
Procurements

HPC@RRZE project staff

Dr.-Ing. Jan Treibig
HPCadd (BMBF)
FEPA (BMBF)
CAS (IBM)

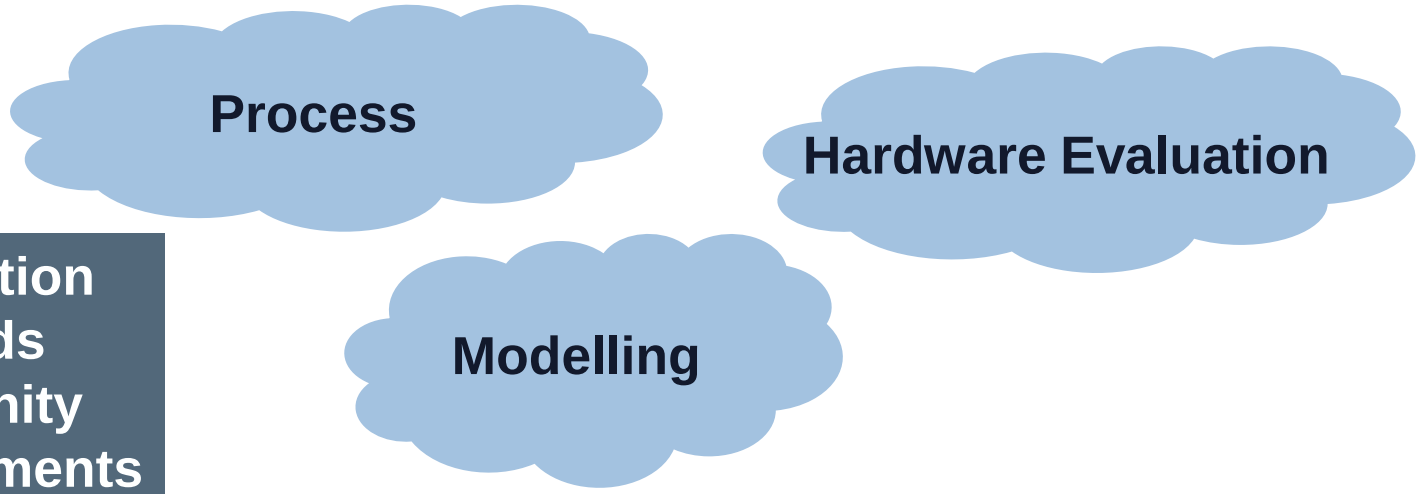
Markus Wittmann
FETOL (BMBF)
SKALB (BMBF)

Holger Stengel
TerraNeo (DFG)
ExaSteel (DFG)

Moritz Kreutzer
ESSEX (DFG SPPEXA)

Faisal Shahzad
ESSEX (DFG SPPEXA)

Overview of activities



- CLI tool collection
- 4000 downloads
- Node info/ affinity
- HPM measurements

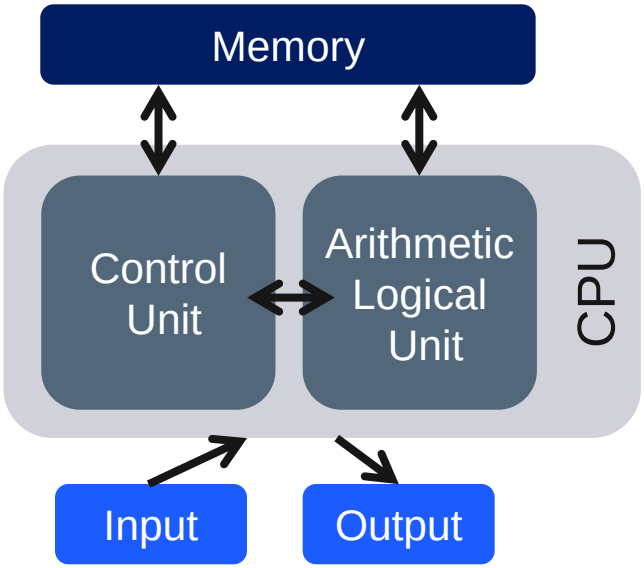
Performance Engineering

- 25 events in last 2 years
- Tutorials (SC, ISC, PPOPP)
- Workshops (Prace PATC)

- Stencil optimization
- Medical imaging
- Sparse matrix alg.
- Binary search trees

- Computational Chemistry
- Fluid Dynamics
- Financial Risk Analysis
- Physics…

Where it all started: Stored Program Computer



Architect's view:
Make the common case fast !

Hardware-Software Co-Design? From algorithm to execution

Application work (user view)

- Flops
- LUPs
- VUPs

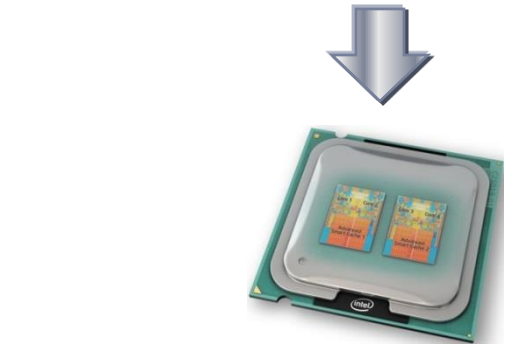
Processor work (architect‘s view)

- Instructions
- Data volume

Algorithm

Programming language

Instruction Set Architecture



Focus on resource utilization

1. **Instruction execution**

   Primary resource of the processor.

2. **Data transfer bandwidth**

   Data transfers are a consequence of instruction execution.

What is the **limiting resource**?
Does the code fully **utilize** the offered **resources**?

Goal: True insight into performance properties of the code

Thinking in Bottlenecks

- A bottleneck is a performance limiting setting
- Microarchitectures expose numerous bottlenecks
- We think about execution in terms of **loops** (steady state)

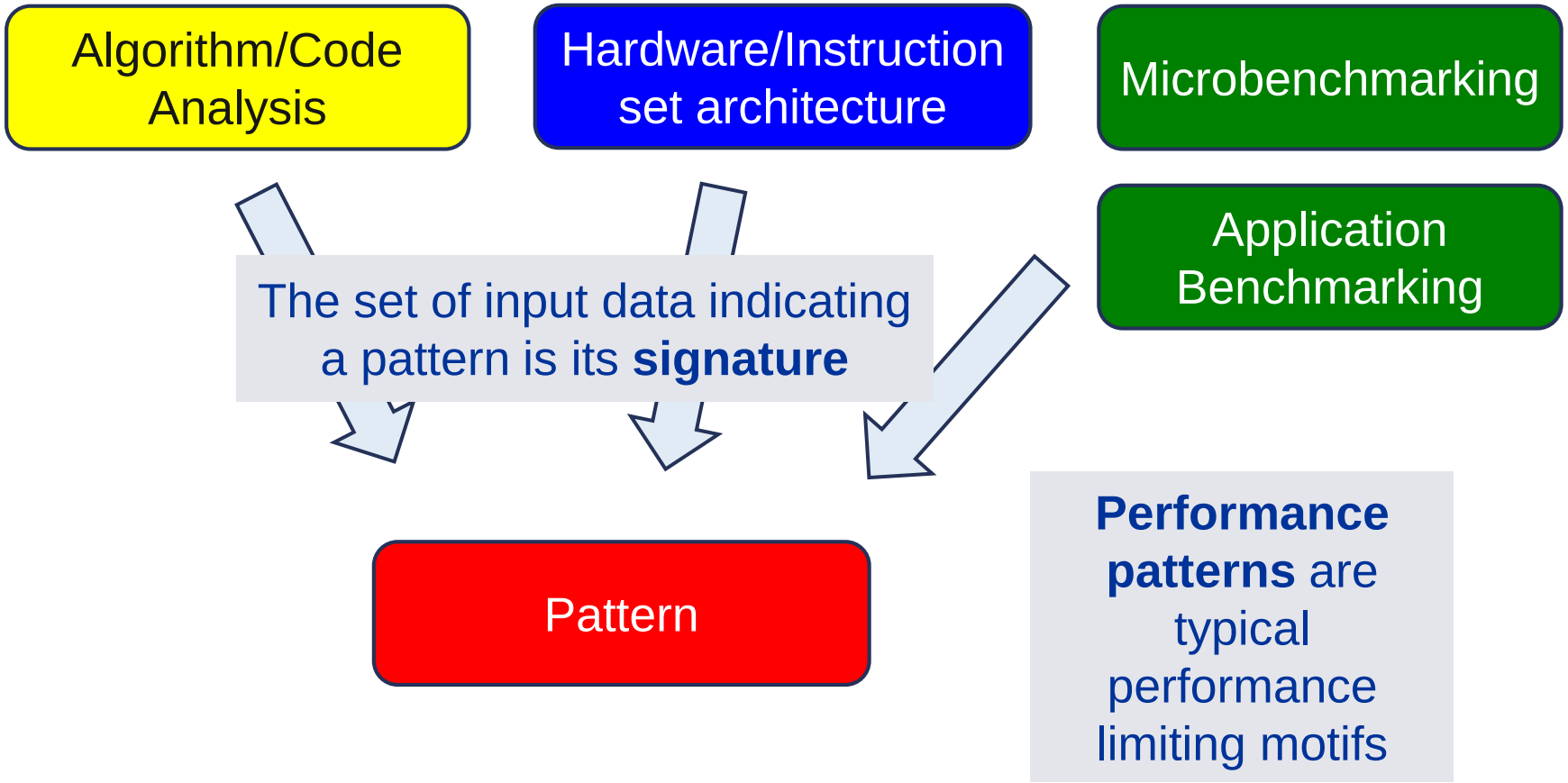
Observation 1:

Most loops face a single (combination of) bottleneck(s) at a time!

Observation 2:

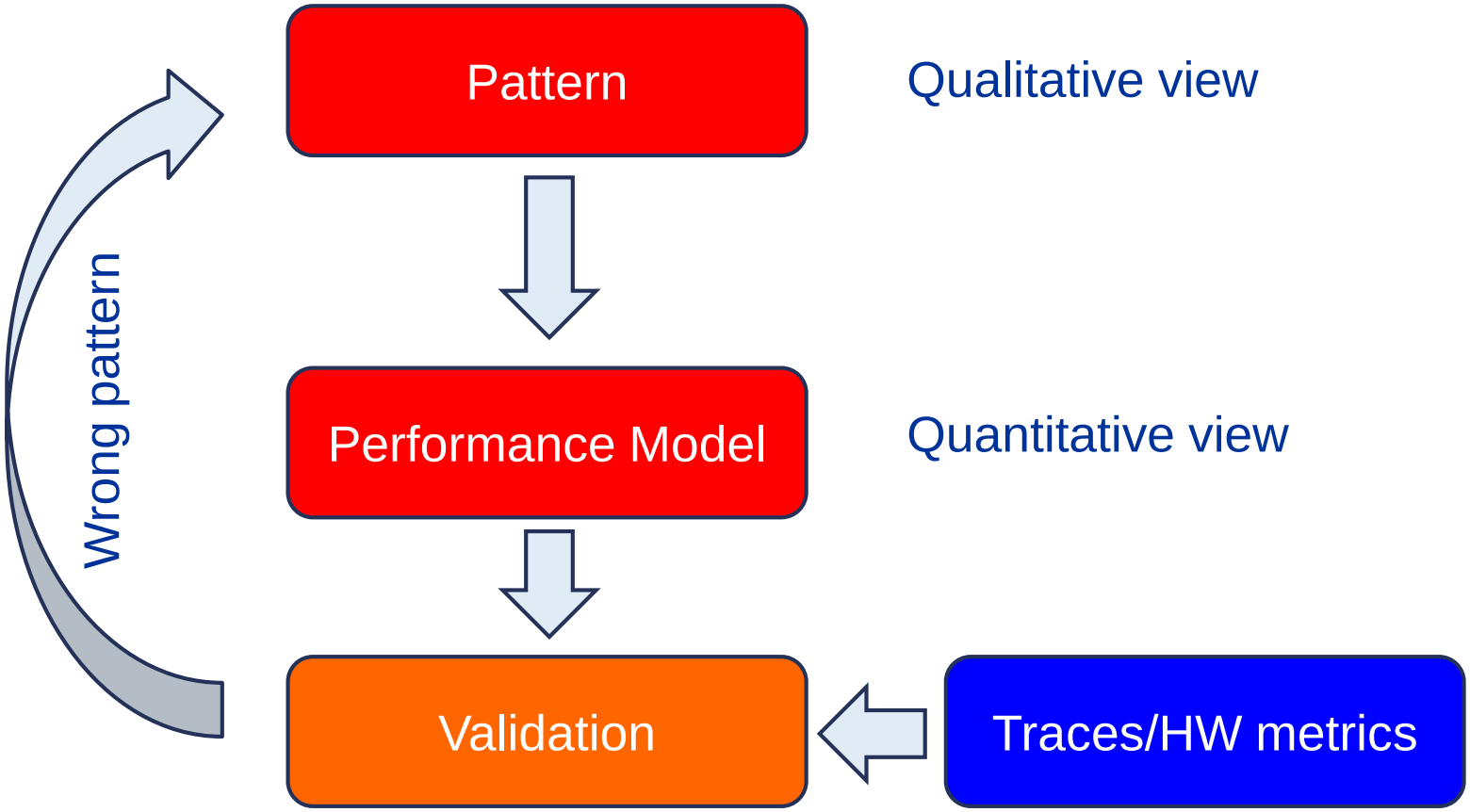
There is a limited number of relevant bottlenecks!

Performance Engineering Process: Analysis



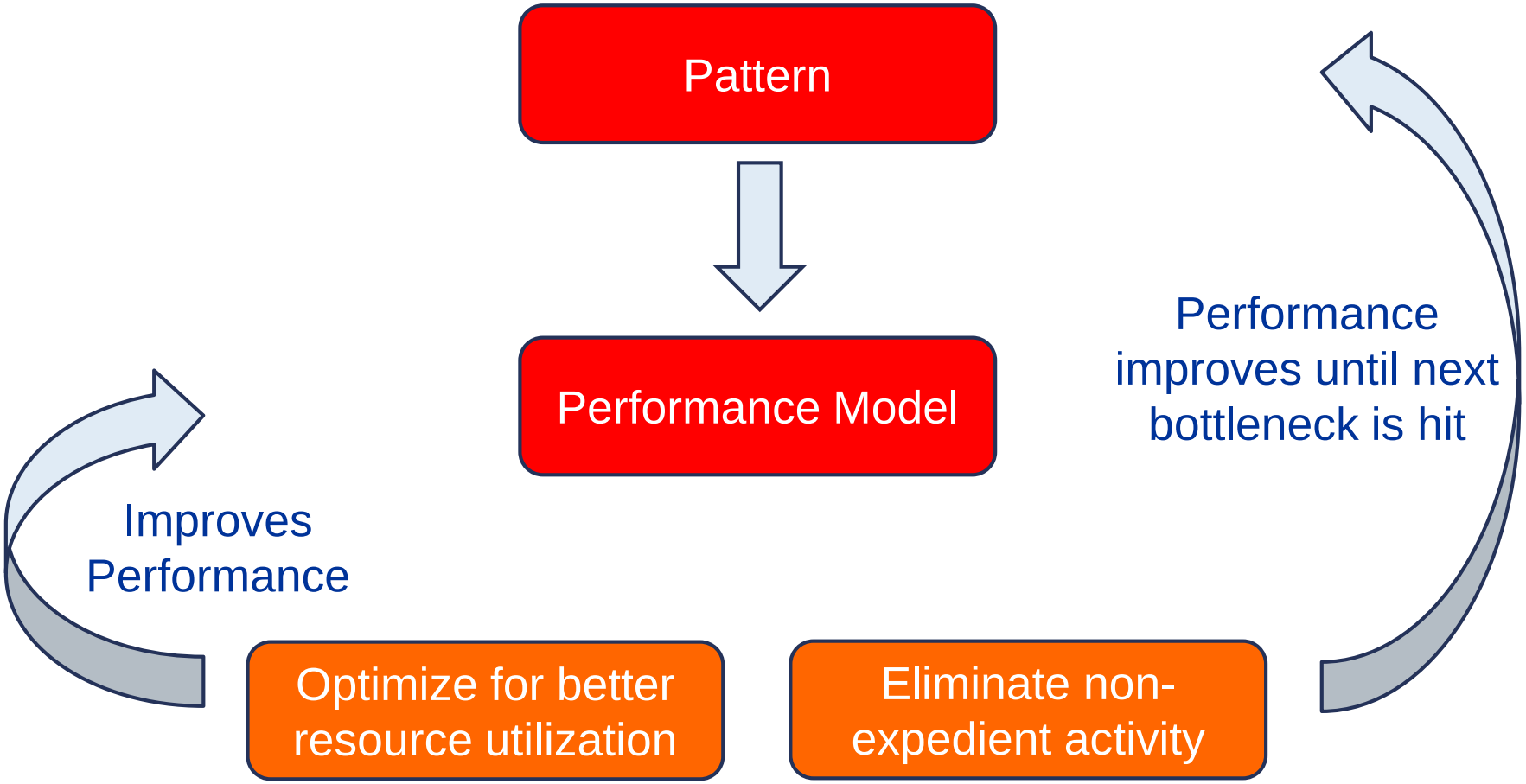
Step 1 **Analysis:** Understanding observed performance

Performance Engineering Process: Modelling



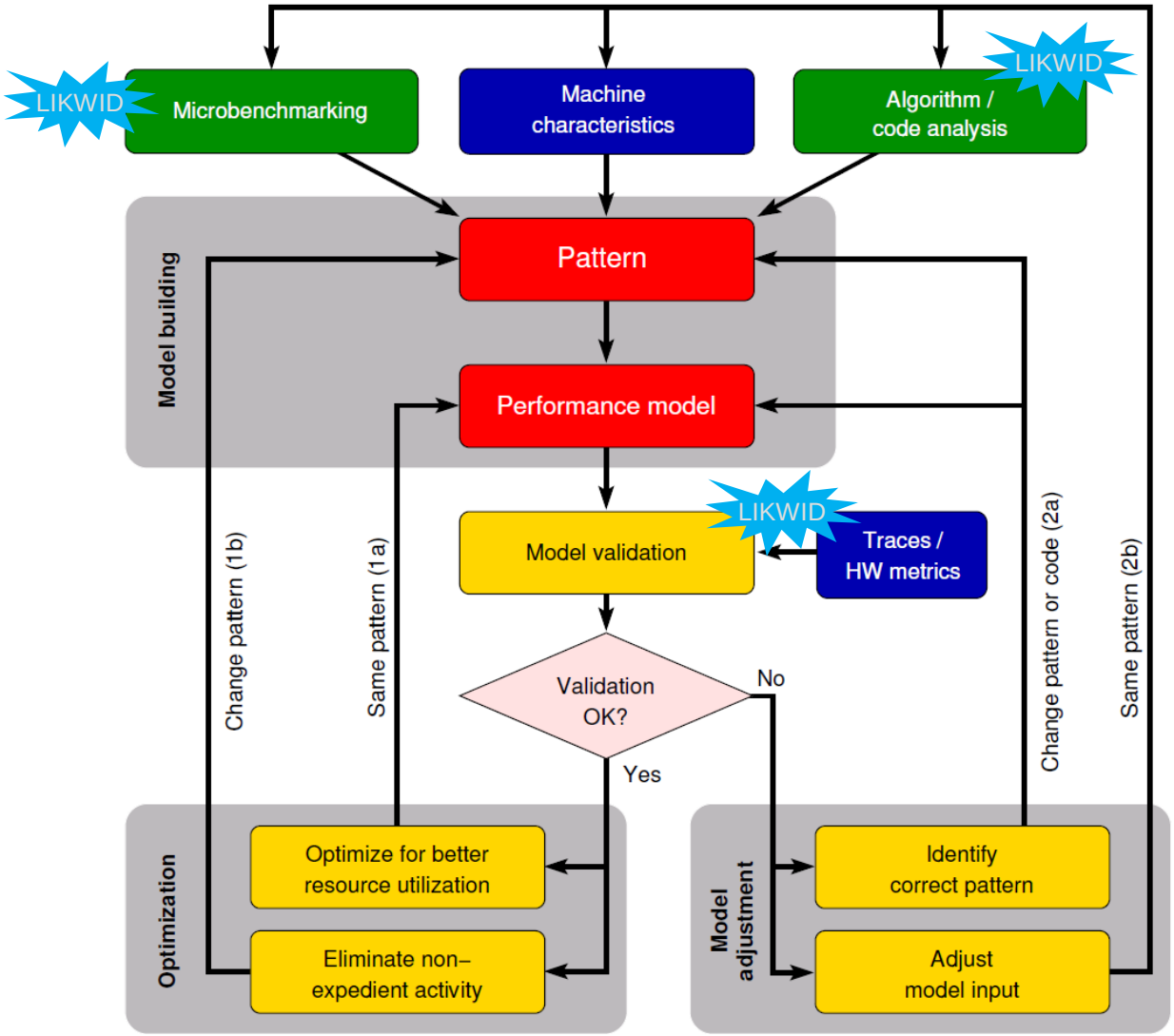
Step 2 **Formulate Model:** Validate pattern and get quantitative insight.

Performance Engineering Process: Optimization



Step 3 **Optimization:** Improve utilization of offered resources.

The whole PE process at a glance

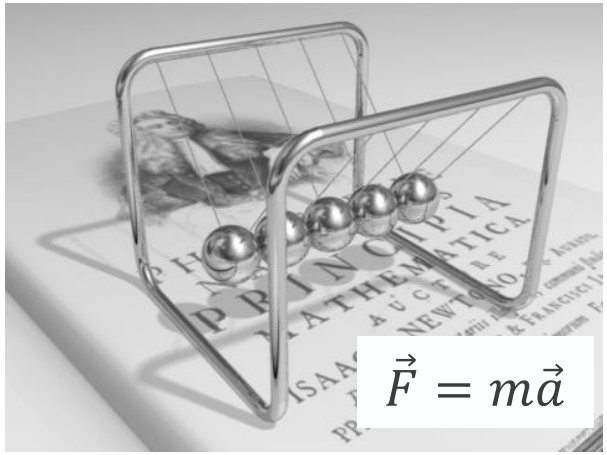


PERFORMANCE MODELS

Analytical Performance Modeling

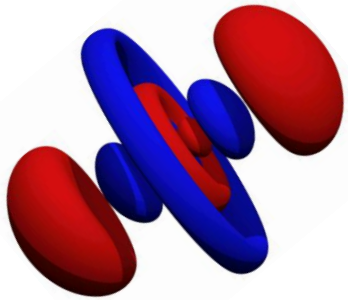
Models in physics

Newtonian mechanics



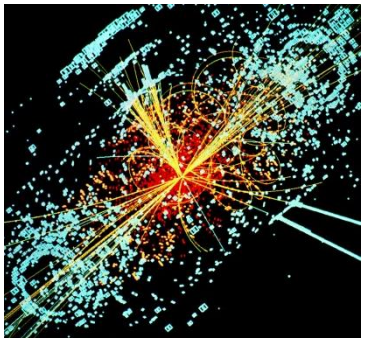
Fails @ small scales!

Nonrelativistic quantum mechanics



$$i\hbar\frac{\partial}{\partial t}\psi(\vec{r},t) = H\psi(\vec{r},t)$$

Fails @ even smaller scales!



Relativistic quantum field theory

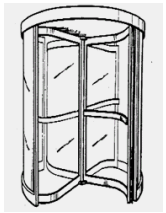
$$U(1)_Y \otimes SU(2)_L \otimes SU(3)_c$$

Consequences

- If models fail, we learn more
- A simple model can get us very far before we need to refine

Example: Modeling customer dispatch in a bank

Revolving door throughput:
b_S [customers/sec]



Processing capability:
P_{max} [tasks/sec]

Intensity:
I [tasks/customer]

Example: Modeling customer dispatch in a bank

How fast can tasks be processed? P [tasks/sec]

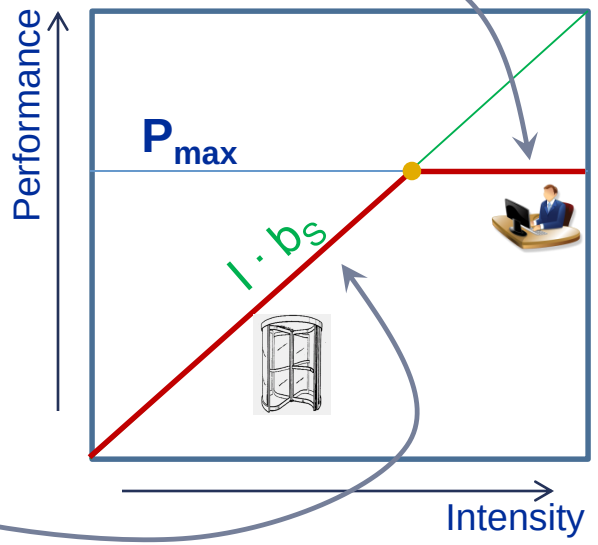
The bottleneck is either

- The service desks (max. tasks/sec): $P_{\max}$
- The revolving door (max. customers/sec): $I \cdot b_S$

$$P = \min(P_{\max}, I \cdot b_S)$$

This is the “Roofline Model”

- High intensity: P limited by “execution”
- Low intensity: P limited by “bottleneck”
- “Knee” at $P_{max} = I \cdot b_S$:
  Best use of resources
- Roofline is an “optimistic” model



The Roofline Model[1,2]

1. P_{max} = Applicable peak performance of a loop, assuming that data comes from L1 cache (this is not necessarily P_{peak})
2. I = Computational intensity ("work" per byte transferred) over the slowest data path utilized ("the bottleneck")
3. b_S = Applicable peak bandwidth of the slowest data path utilized

Expected performance:

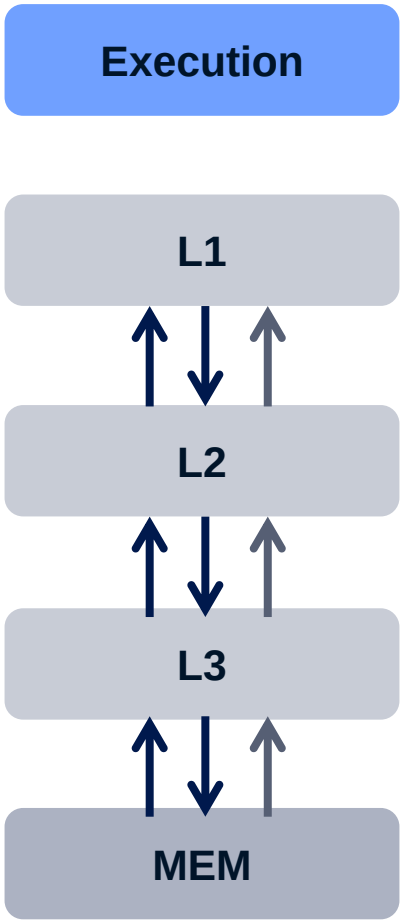
[F/B] [B/s]

$$P = \min(P_{\max}, I \cdot b_S)$$

[1] W. Schönauer: Scientific Supercomputing: Architecture and Use of Shared and Distributed Memory Parallel Computers. (2000)
[2] S. Williams: Auto-tuning Performance on Multicore Computers. UCB Technical Report No. UCB/EECS-2008-164. PhD thesis (2008)

ECM ("Execution-Cache-Memory") Model

```
A(:)=C(:)
```



2 cy

6 cy

6 cy

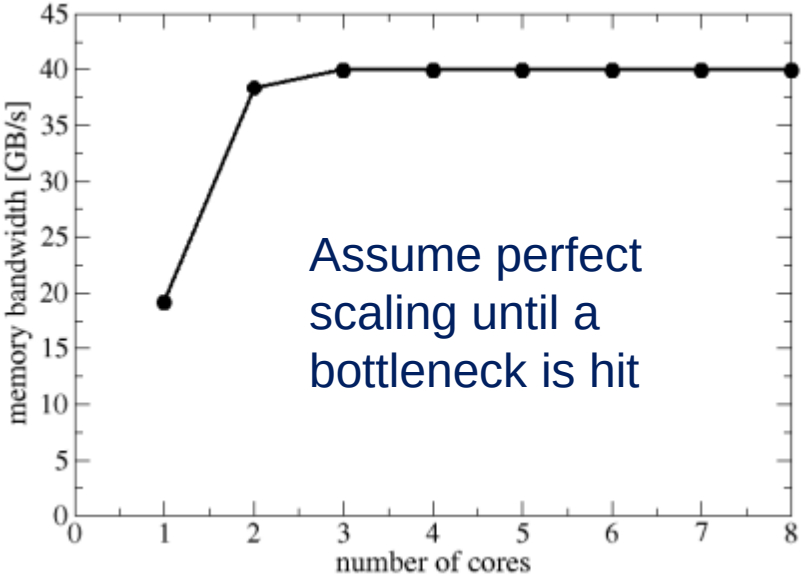
13 cy

Intel SandyBridge

2.7 GHz, 8 cores

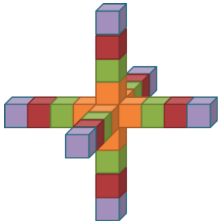
1LD + ½ST per cycle
AVX 32B wide

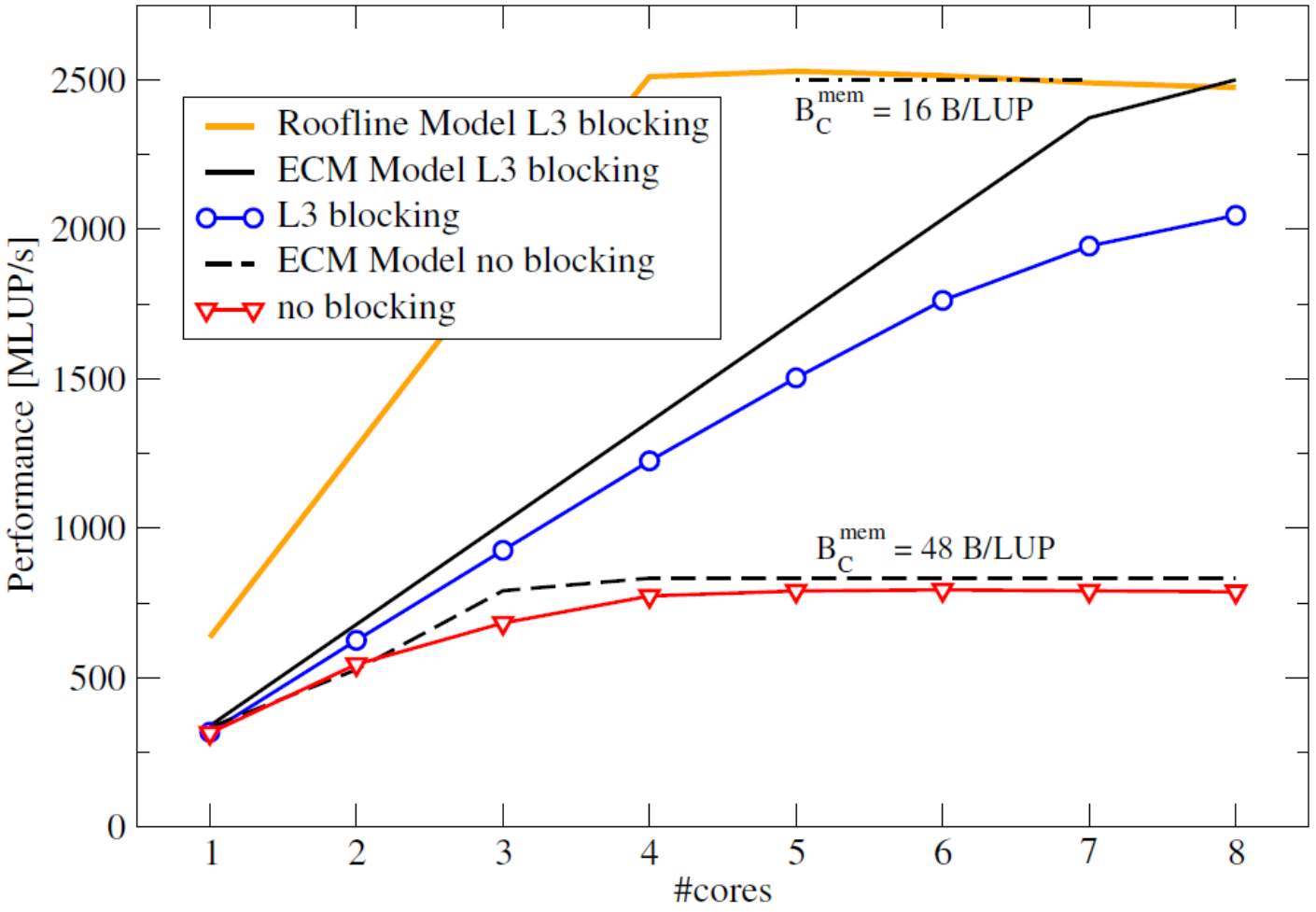
Cache BW: 32 B/cy
Memory BW: 15 B/cy
Cacheline size: 64B



Total: 27 cycles
MEM Bandwidth (1 core): 19.2 GB/s
Saturated Bandwidth: 40 GB/s

Example: 3D long-range stencil on Sandy Bridge





ECM model usage

Educational

- Explain cache behavior
- Explain SIMD benefit
- Explain bandwidth scaling

Performance Engineering

- Determine limiting bottleneck
- Get a clear picture about runtime contributions

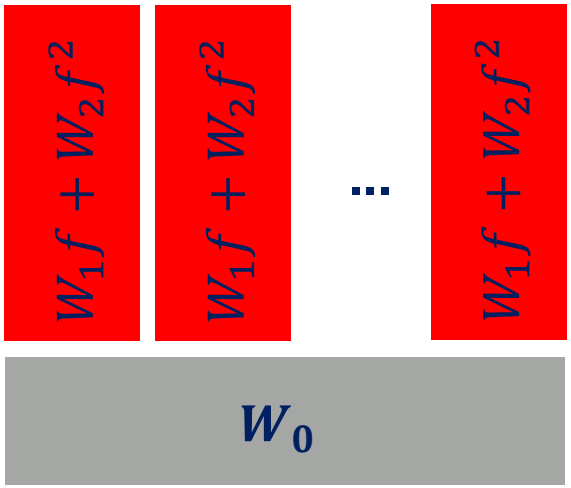
Research

- Couple with power model
- Architectural exploration
- Reveal architectural shortcomings

A simple power model for multicore chips

Model assumptions:

1. Power is a quadratic polynomial in the clock frequency: $W = W_0 + w_1 f + w_2 f^2$
2. Dynamic power is linear in the number of active cores: $W_{dyn} = (W_1 f + W_2 f^2) n$
3. Performance is linear in the number of cores until it hits a bottleneck (← ECM model)
4. Performance is linear in the clock frequency unless it hits a bottleneck (simplification from the ECM model)
5. Energy to solution is power dissipation divided by performance



Model:

$$E = \frac{\text{Power}}{\text{Performance}} = \frac{W_0 + (W_1 f + W_2 f^2) n}{\min(n P_0 \, f / f_0, P_{max})}$$

PERFORMANCE PATTERNS

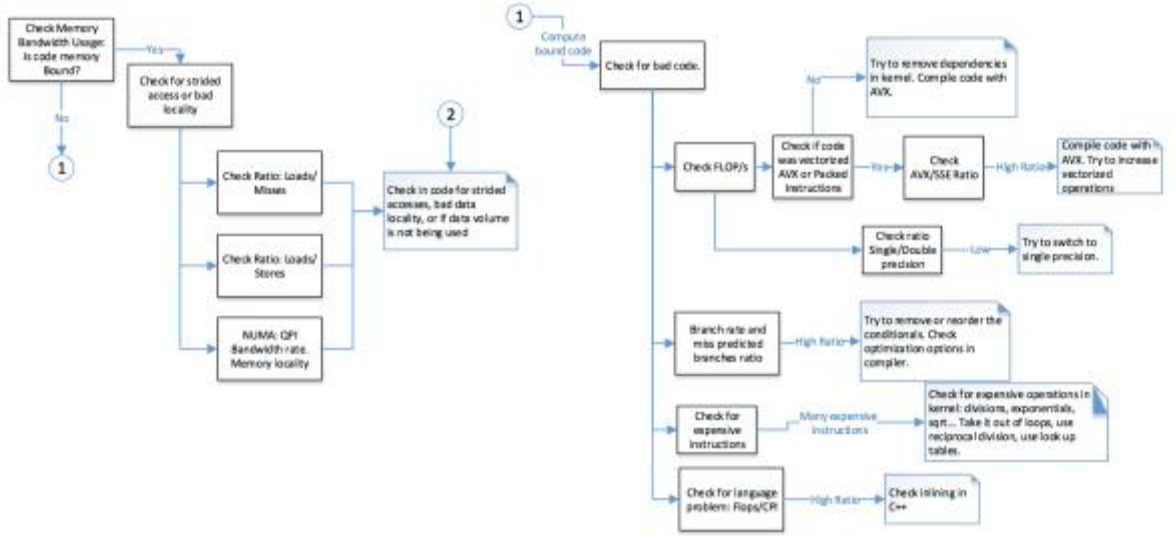
Helpful motifs for performance analysis

Performance pattern classification

1. Maximum resource utilization
2. Hazards
3. Work related (Application or Processor)

J. Treibig, G. Hager, and G. Wellein: *Performance patterns and hardware metrics on modern multicore processors: Best practices for performance engineering*.
DOI: 10.1007/978-3-642-36949-0_50

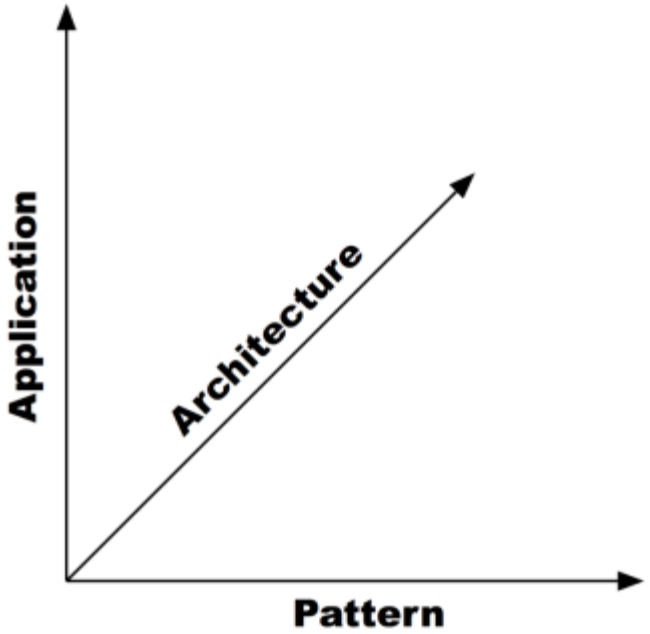
Classification can be semi-automated



Application classification using patterns

- Categorize relevant benchmarks and application classes according to performance patterns

- This application map can be used:
  - To get complete list of relevant patterns and their probability
  - As a knowledge base about relevant performance problems and their cure
  - To suggest architectural improvements



LIKWID TOOLS

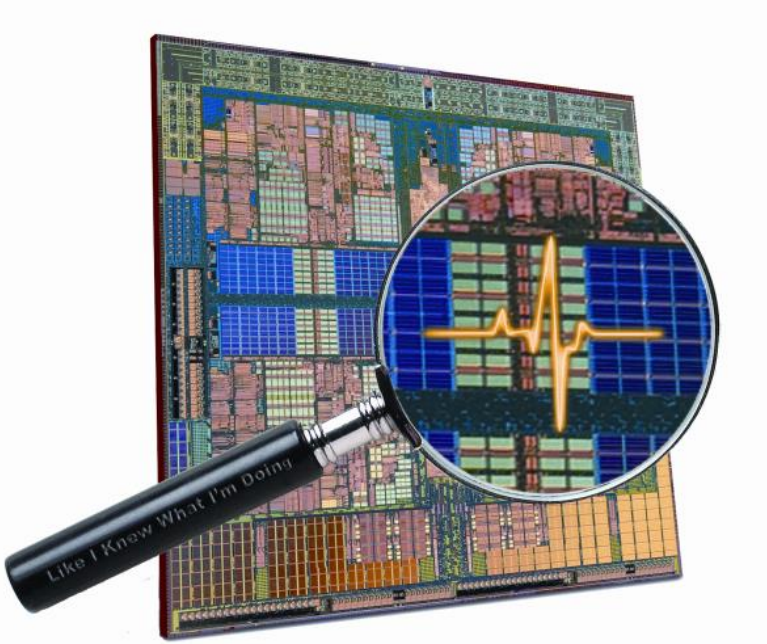
A performance-oriented tool suite for multicore processors

LIKWID

- LIKWID tool suite:

  Like
  I
  Knew
  What
  I'm
  Doing

- Open source tool collection (developed at RRZE): http://code.google.com/p/likwid



J. Treibig, G. Hager, G. Wellein: ***LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments.***
PSTI2010, Sep 13-16, 2010, San Diego, CA
DOI: 10.1109/ICPPW.2010.38

LIKWID Tool Suite

- Command line tools for Linux:
  - easy to install
  - standard linux kernel
  - simple and clear to use
  - supports Intel and AMD

- Current tools:
  - **likwid-topology**: Print thread and cache topology
  - **likwid-pin**: Pin threaded application without touching code
  - **likwid-perfctr:** Measure performance counters
  - **likwid-powermeter:** Measure power, energy, temperature
  - **likwid-mpirun**: mpirun wrapper script for easy LIKWID integration
  - **likwid-bench**: Low-level bandwidth benchmark generator tool
  - … some more

Conclusion

Present work:

- Enable performance engineers to do the job
- Provide knowledge, methods and tools
- Concentrate on scientific computing
- Coupling performance and power models

Mid-term future research:

- Pattern classification map
- Analysis of architectures and software/hardware interfaces

Long-term future research:

- Future architectures (simple, heterogeneous, special purpose)
- Tackle other important areas (big data, pattern recognition)

References (selection)

Book:

G. Hager and G. Wellein: Introduction to High Performance Computing for Scientists and Engineers. CRC Computational Science Series, 2010. ISBN 978-1439811924
http://www.hpc.rrze.uni-erlangen.de/HPC4SE/

Papers:

M. Kreutzer, G. Hager, G. Wellein, A. Pieper, A. Alvermann, and H. Fehske: Performance Engineering of the Kernel Polynomial Method on Large-Scale CPU-GPU Systems. Accepted for IPDPS15.
Preprint: arXiv:1410.5242

G. Hager, J. Treibig, J. Habich and G. Wellein: Exploring performance and power properties of modern multicore chips via simple machine models. Computation and Concurrency: Practice and Experience DOI: 10.1002/cpe.3180 (2014)

J. Treibig, G. Hager and G. Wellein: Performance patterns and hardware metrics on modern multicore processors: Best practices for performance engineering. Workshop on Productivity and Performance (PROPER 2012) at Euro-Par 2012, August 28, 2012, Rhodes Island, Greece. DOI: 10.1007/978-3-642-36949-0_50.

J. Treibig, G. Hager, H. Hofmann, J. Hornegger and G. Wellein: Pushing the limits for medical image reconstruction on recent standard multicore processors. International Journal of High Performance Computing Applications, DOI: 10.1177/1094342012442424.

J. Treibig, G. Hager and G. Wellein: LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments. Proc. PSTI2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures, San Diego CA, September 13, 2010. DOI: 10.1109/ICPPW.2010.38.