White-box modeling for performance and energy: Useful patterns for resource optimization

Georg Hager, Ayesha Afzal
Erlangen Regional Computing Center (RRZE)
University of Erlangen-Nuremberg
Erlangen, Germany

PACO 2015
Max Planck Institute for Dynamics of Complex Technical Systems
Magdeburg, Germany
July 6, 2015

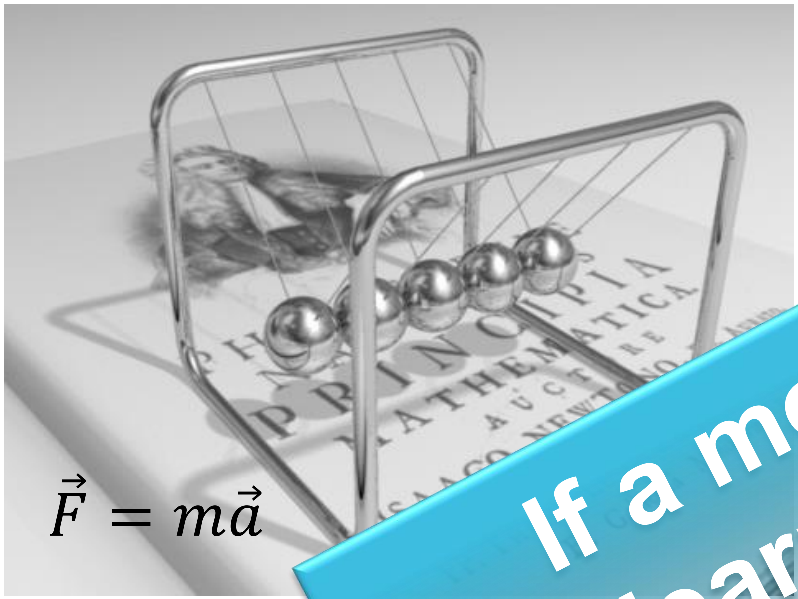
Outline

- Performance Modeling and Engineering
  - Motivation
  - Simple “White Box” modeling: Roofline

- A simple power model for multicore CPUs
  - Observations
  - Model construction
  - Validation

An example from physics

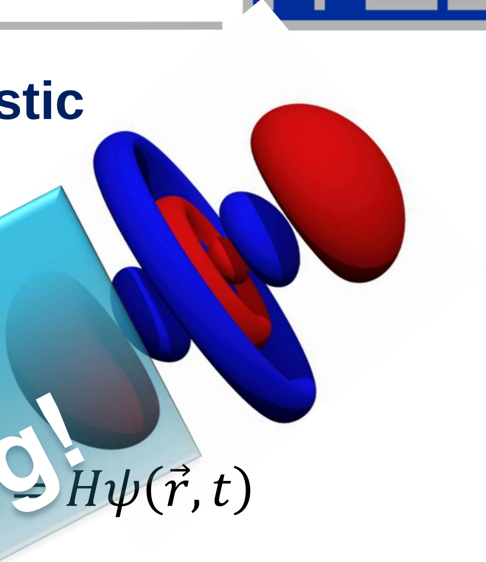
Newtonian mechanics



$\vec{F} = m\vec{a}$

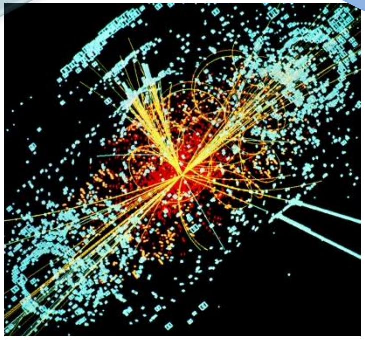
Fails @ small scales!

Nonrelativistic quantum mechanics



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

Fails @ even smaller scales!

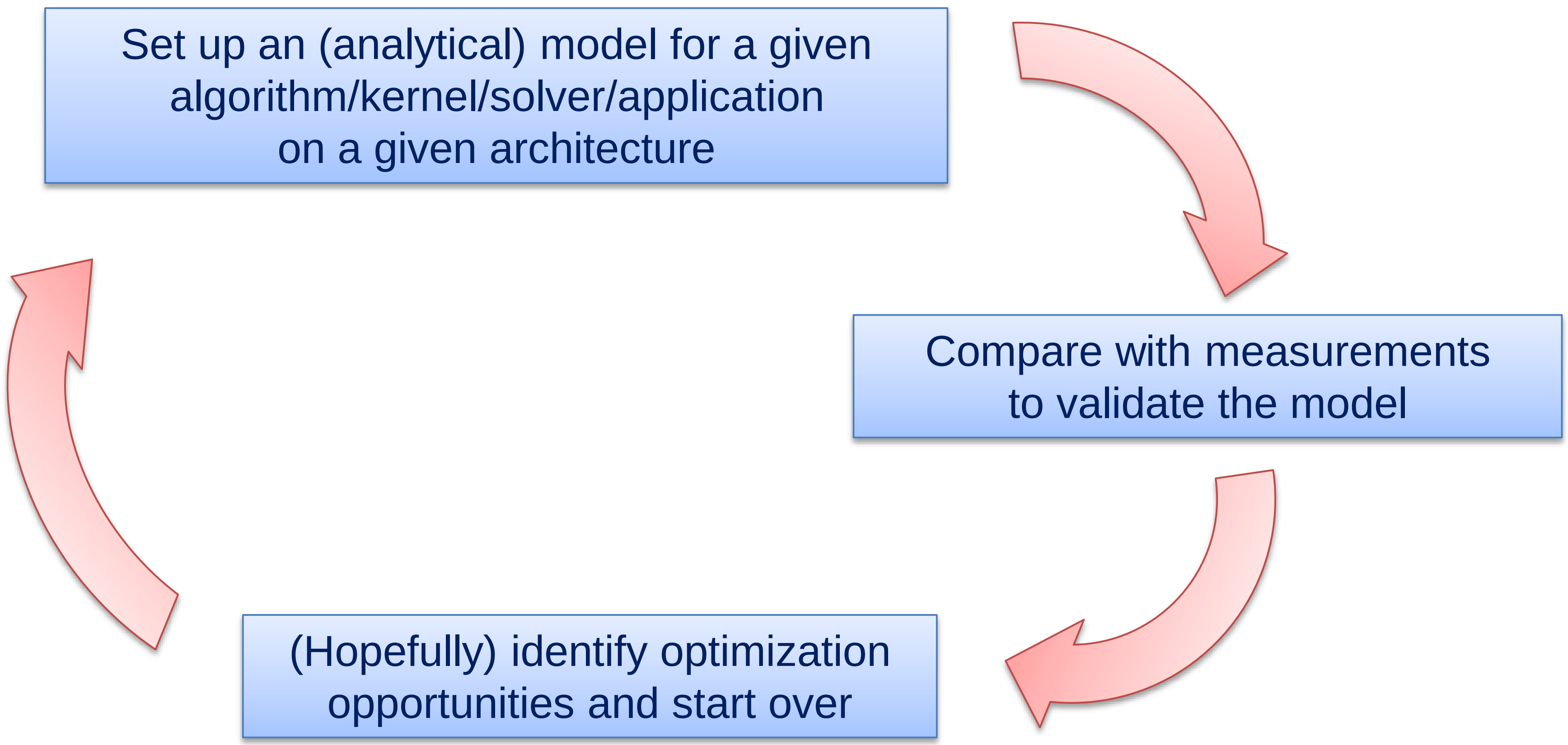


Relativistic quantum field theory

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

If a model fails, we learn something!

White box performance engineering


Set up an (analytical) model for a given algorithm/kernel/solver/application on a given architecture
Compare with measurements to validate the model
(Hopefully) identify optimization opportunities and start over

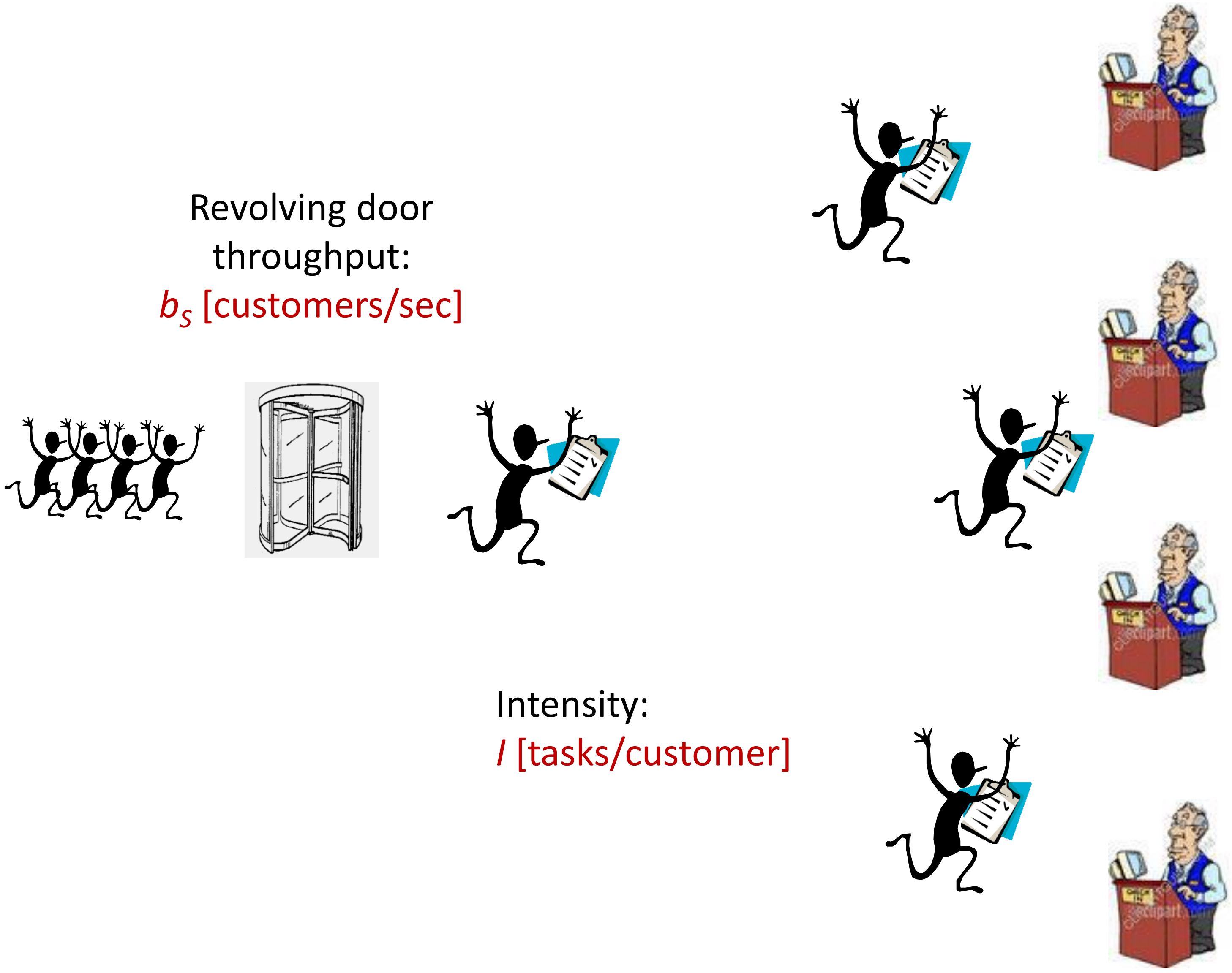
“White Box” Performance Modeling on the Chip Level: Roofline

D. Callahan et al.: Estimating interlock and improving balance for pipelined architectures.
Journal for Parallel and Distributed Computing 5(4), 334 (1988).
DOI: 10.1016/0743-7315(88)90002-0

W. Schönauer: Scientific Supercomputing: Architecture and Use of Shared and Distributed Memory Parallel Computers. Self-edition (2000)

S. Williams: Auto-tuning Performance on Multicore Computers.
UCB Technical Report No. UCB/EECS-2008-164. PhD thesis (2008)

Prelude: Modeling customer dispatch in a bank



Revolving door throughput:
b_S [customers/sec]

Intensity:
I [tasks/customer]

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

Prelude: Modeling customer dispatch in a bank

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

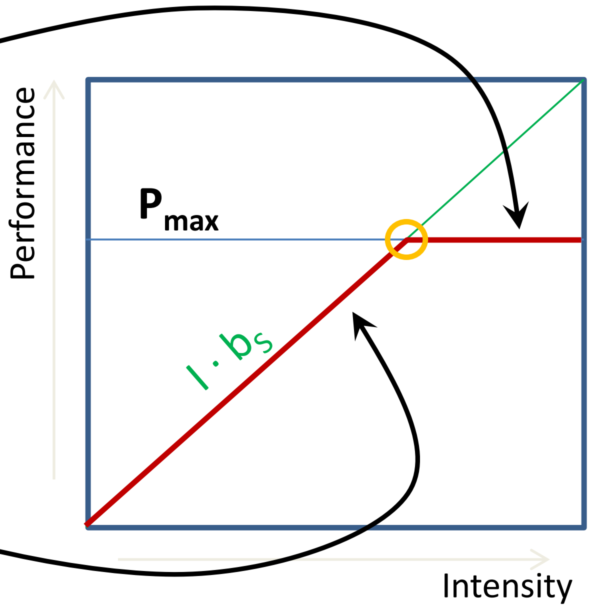
The bottleneck is either

- 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:
  "light speed"

The Roofline Model

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")
   - Code balance $B_C = I^{-1}$

3. **b_S** = Applicable peak bandwidth of the slowest data path utilized

Expected performance:

$$P = \min(P_{\max}, \underbrace{I}_{[F/B]} \cdot \underbrace{b_S}_{[B/s]})$$

Roofline Model assumptions ("machine model")

- There is a clear concept of "work" vs. "traffic"
  - "work" = flops, updates, iterations…
  - "traffic" = required data to do "work"
- No latency effects → perfect streaming mode
- One data transfer bottleneck is modeled only; all others are assumed to be infinitely fast

- **Data transfer and core execution overlap perfectly!**
  - This is the main problem in situations where Roofline does not work!
  - Remedy: Execution-Cache-Memory (ECM) model

H. Stengel, J. Treibig, G. Hager, and G. Wellein: *Quantifying performance bottlenecks of stencil computations using the Execution-Cache-Memory model*.
Proc. ICS'15, DOI: 10.1145/2751205.2751240, Preprint: arXiv:1410.5010

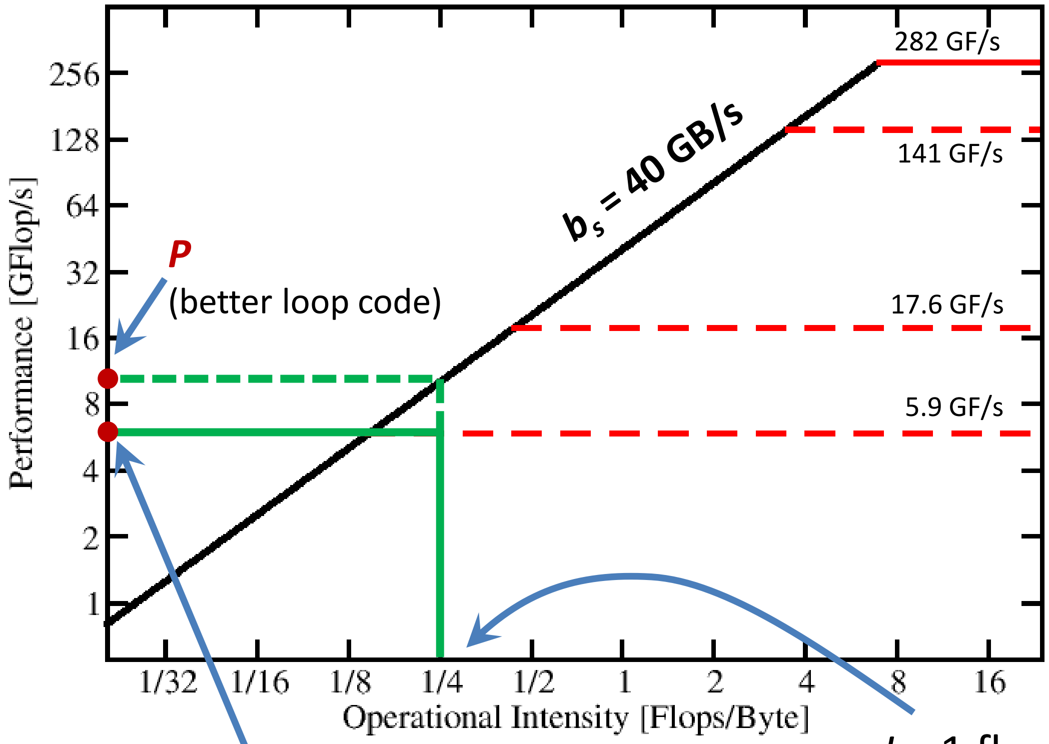
G. Hager, J. Treibig, J. Habich, and G. Wellein: *Exploring performance and power properties of modern multicore chips via simple machine models*. Concurrency and Computation: Practice and Experience (2013), DOI: 10.1002/cpe.3180. Preprint: arXiv:1208.2908

A “simple” Roofline example

Example: `do i=1,N; s=s+a(i); enddo`

in single precision on a 2.2 GHz Sandy Bridge socket @ “large” N

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



P (better loop code)

P (worst loop code)

Machine peak (ADD+MULT) Out of reach for this code

ADD peak (best possible code)

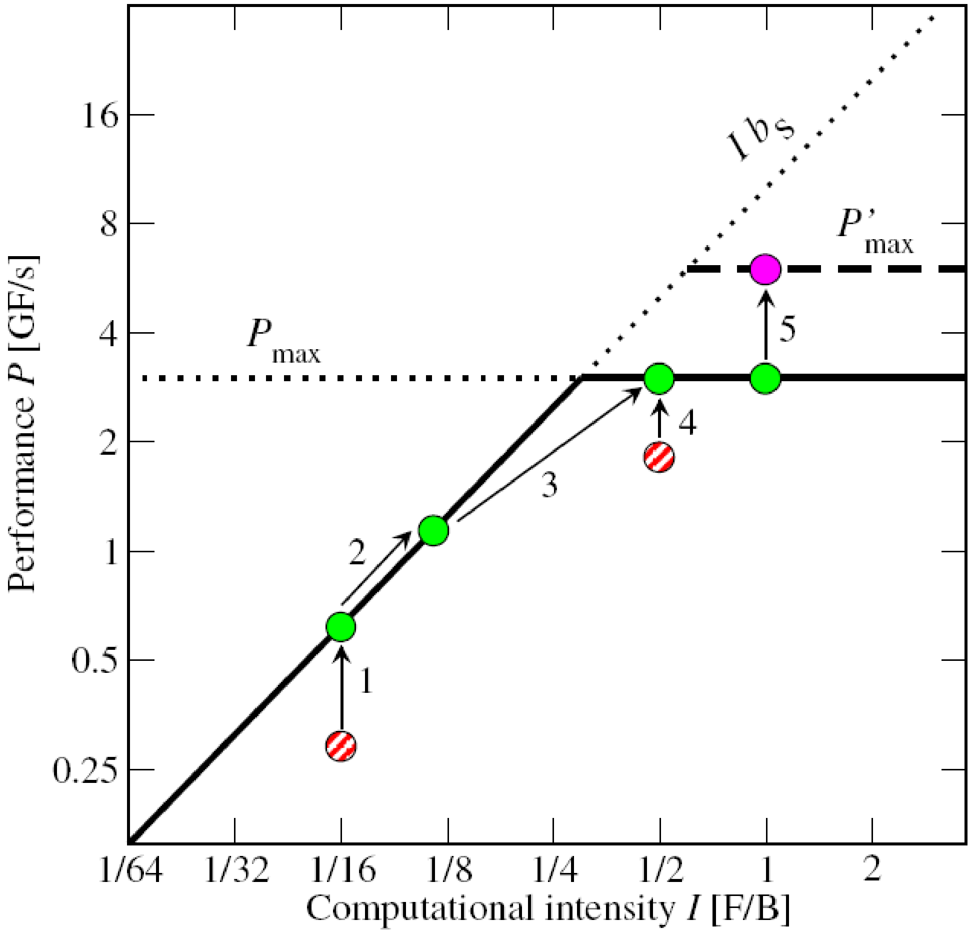
no SIMD

worst code

I = 1 flop / 4 byte (SP!)

Typical code optimizations in the Roofline Model

1. Hit the BW bottleneck by good serial code
(e.g., Perl → Fortran)
2. Increase intensity to make better use of BW bottleneck
(e.g., loop blocking [see later])
3. Increase intensity and go from memory-bound to core-bound
(e.g., temporal blocking)
4. Hit the core bottleneck by good serial code
(e.g., `-fno-alias` [see later])
5. Shift P_{max} by accessing additional hardware features or using a different algorithm/implementation
(e.g., scalar → SIMD)



A simple power model for multicore processors

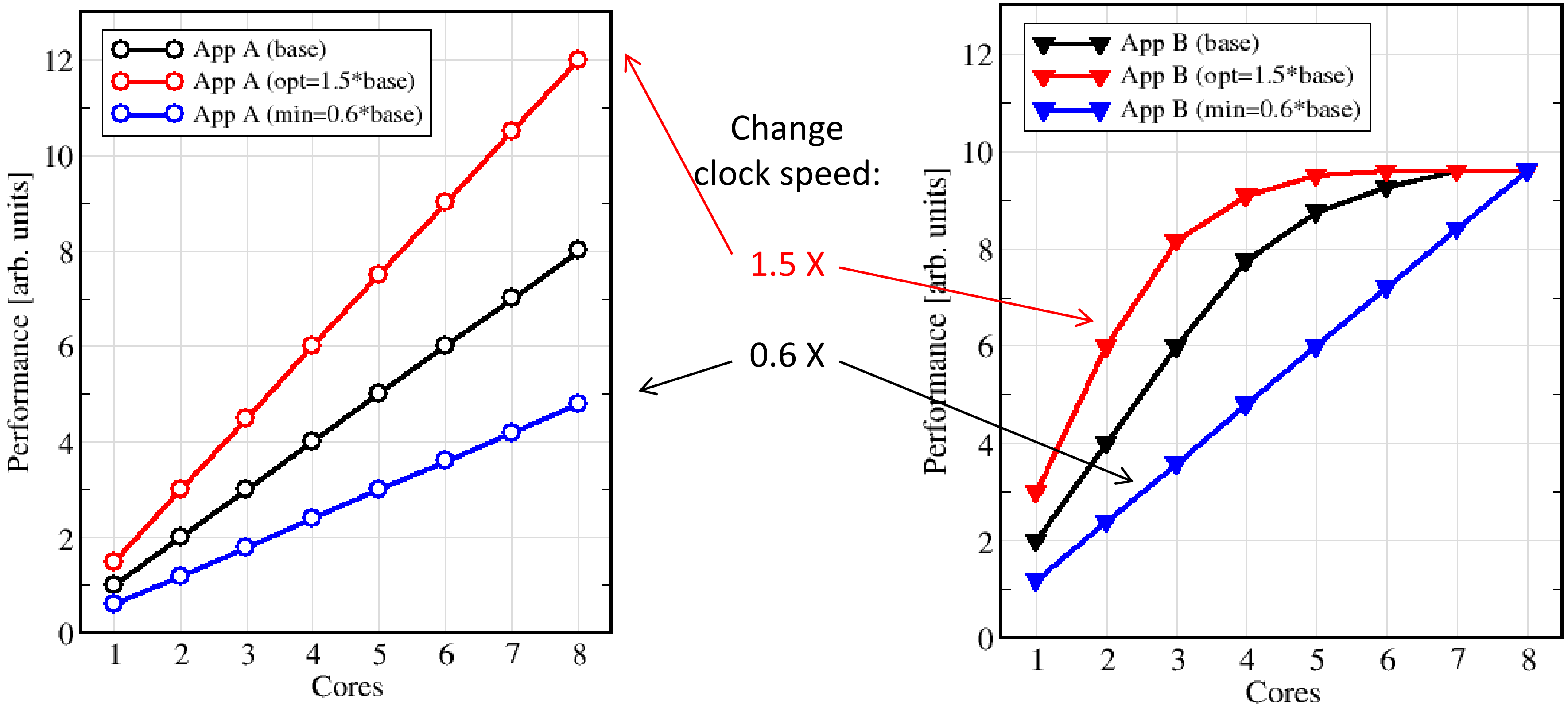
G. Hager, J. Treibig, J. Habich, and G. Wellein: *Exploring performance and power properties of modern multicore chips via simple machine models*. Concurrency and Computation: Practice and Experience (2013), DOI: 10.1002/cpe.3180. Preprint: arXiv:1208.2908

Prelude: There are two kinds of loops …

Typical performance vs. cores behavior on multicore chips:

"LINPACK type"
Limiting factor: core execution
"Flat roof" region

"STREAM type"
Limiting factor: saturation (bandwidth)
"Sloped roof" region



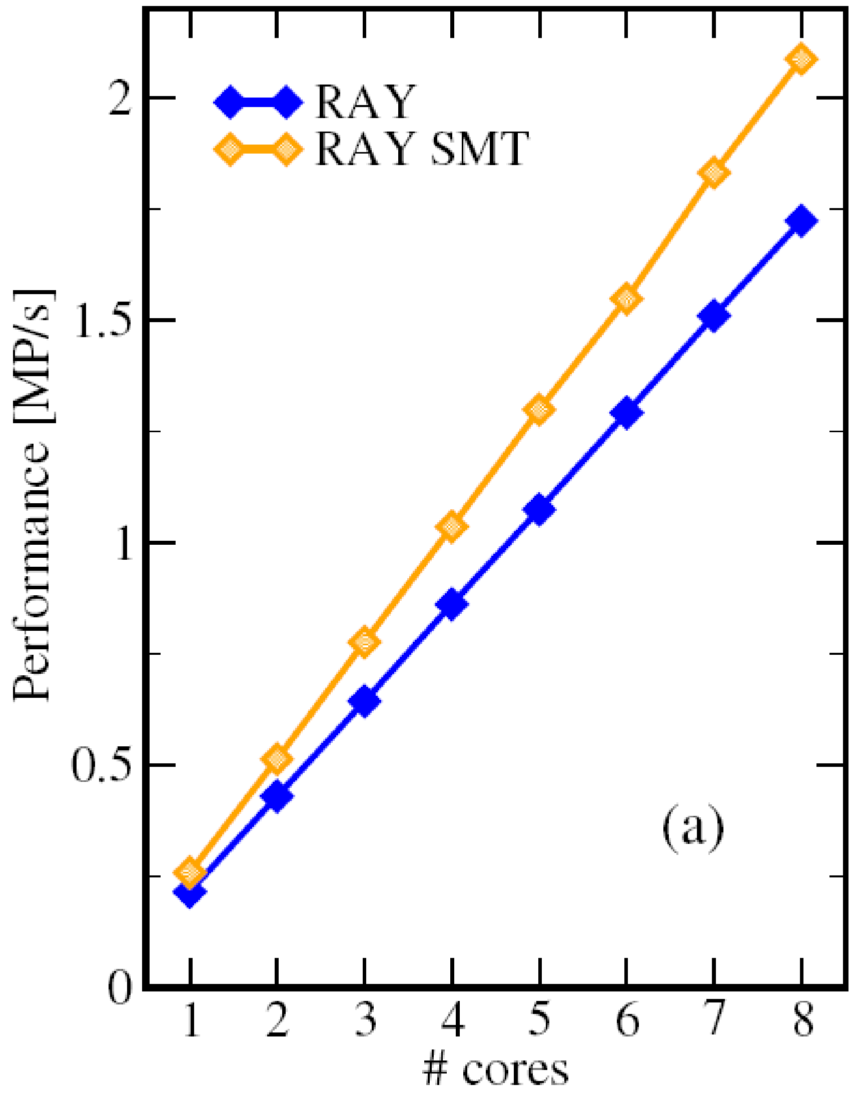
A model for multicore chip power

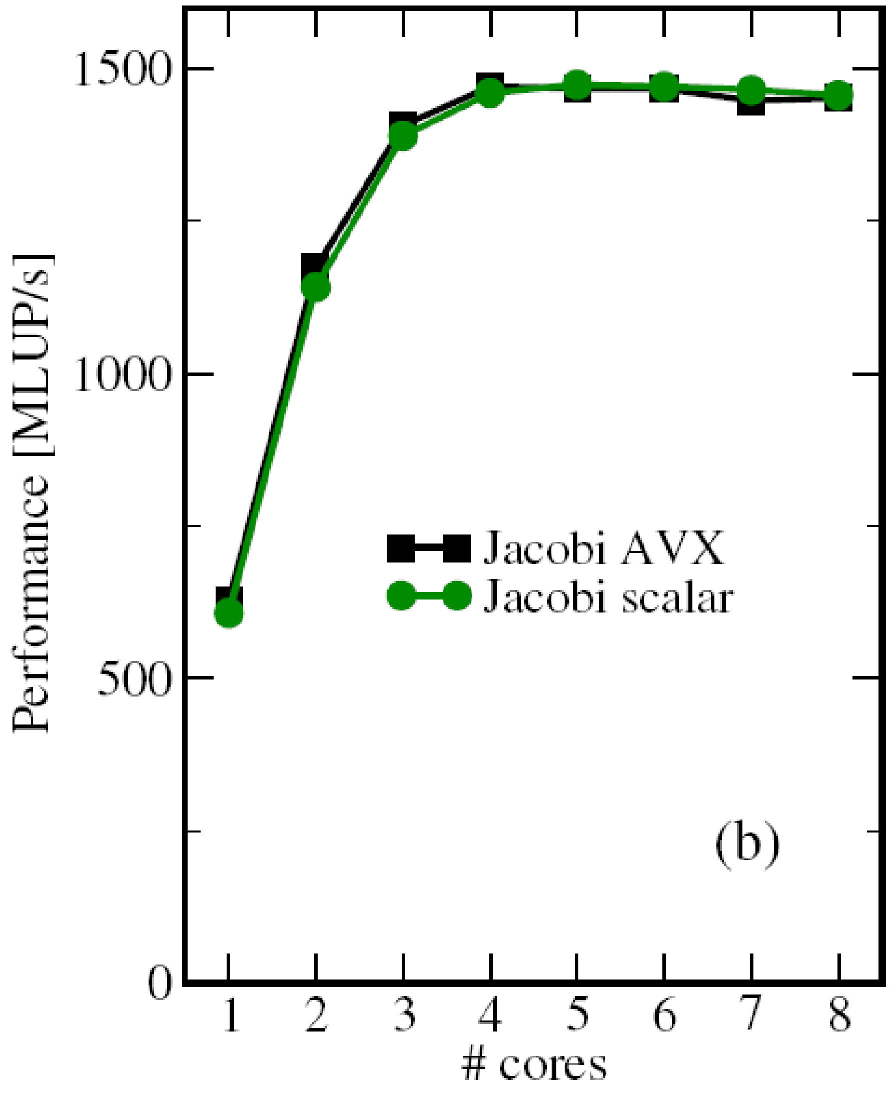
- Goal: Establish model for chip power and program energy consumption with respect to
  - Clock speed
  - Number of cores used
  - Single-thread program performance

- Choose different characteristic benchmark applications to measure a chip's power behavior
  - Matrix-matrix-multiply ("DGEMM"): "Hot" code, well scalable
  - Ray tracer: Sensitive to SMT execution (15% speedup), well scalable
  - 2D Jacobi solver: 4000x4000 grid, strong saturation on the chip
    - AVX variant
    - Scalar variant

- Measure characteristics of those apps and establish a power model

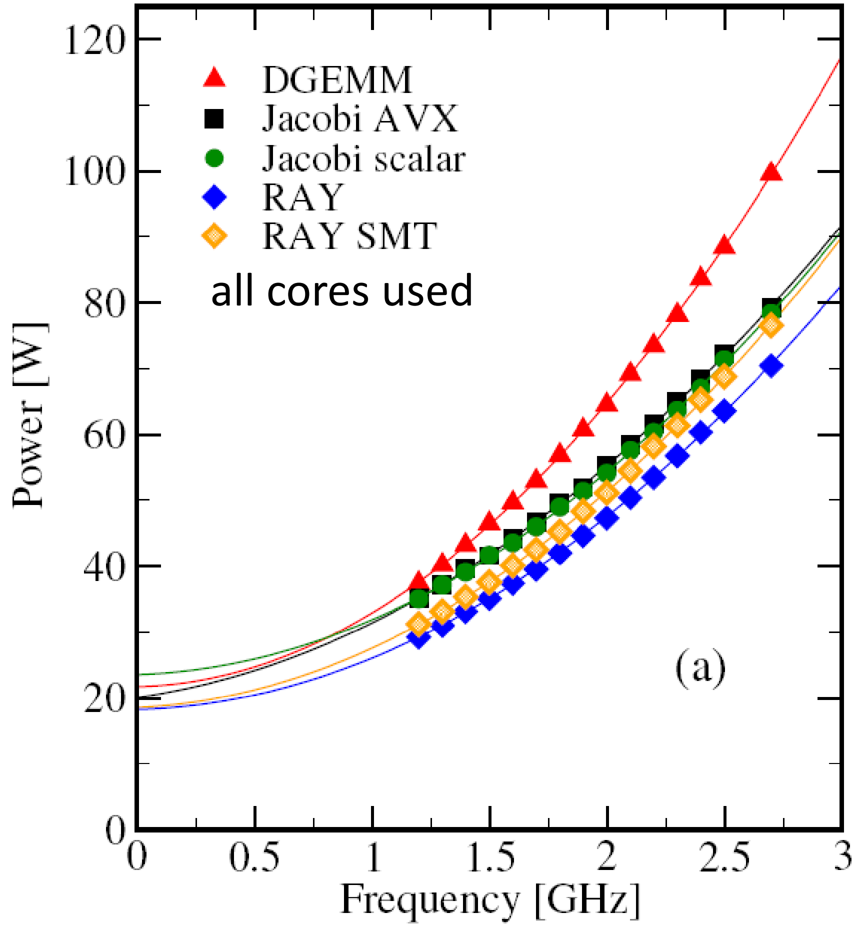
App scaling behavior (DGEMM omitted)

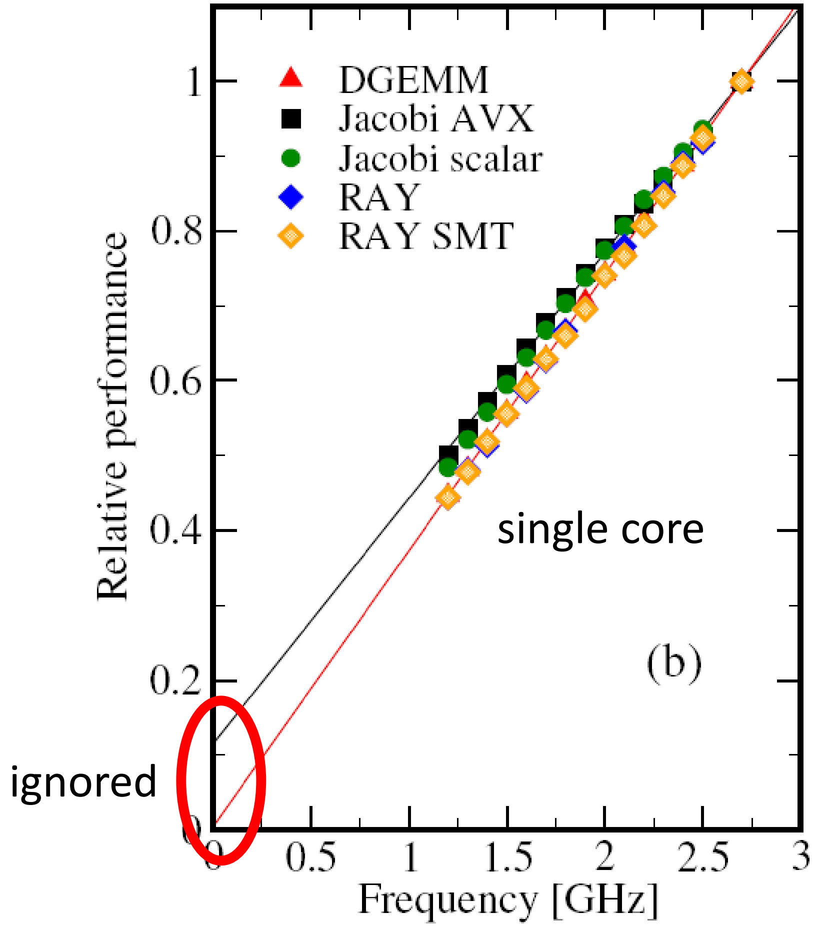




Chip power and performance vs. clock speed on full socket & single core

Sandy Bridge EP (8-core) processor:

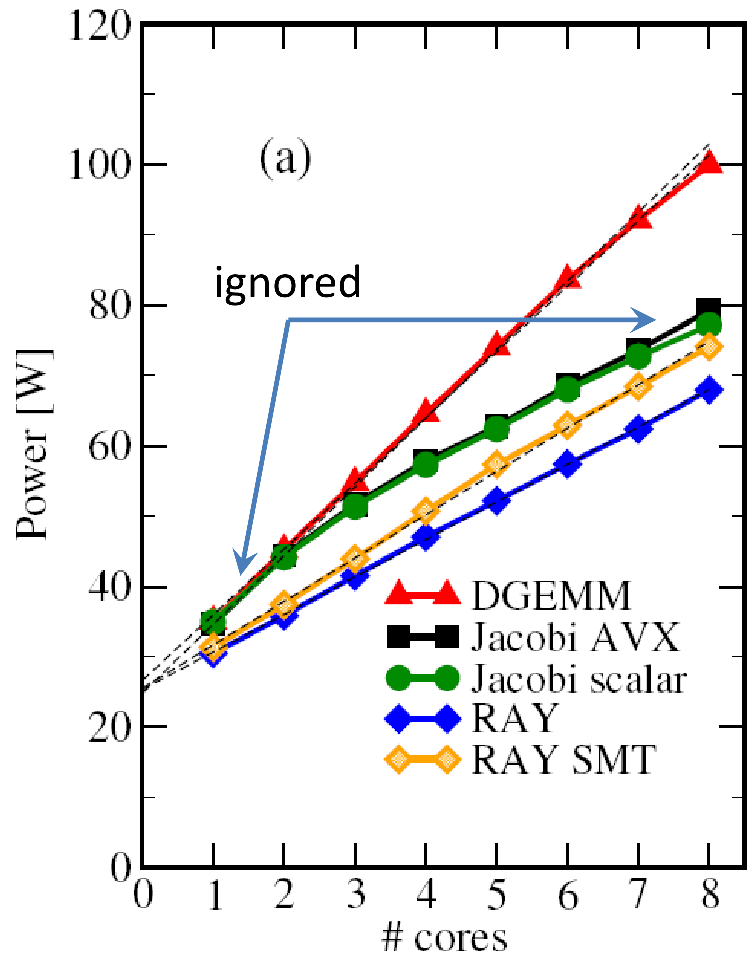


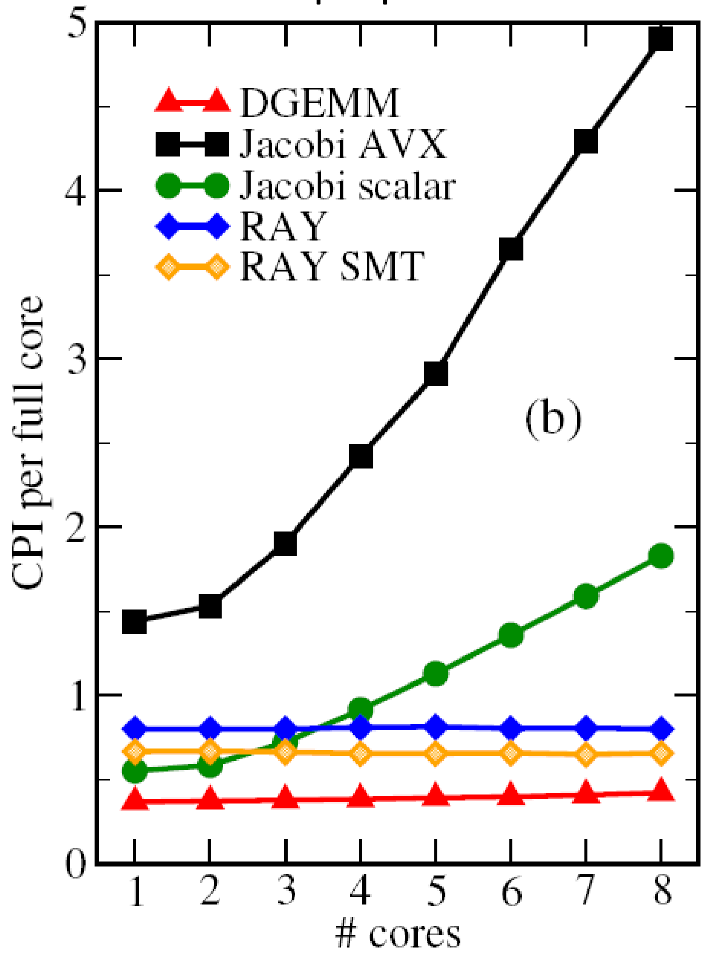


Chip power and cycles per instruction (CPI) vs. # of cores

Sandy Bridge EP (8-core) processor:

CPI and power correlated, but not proportional

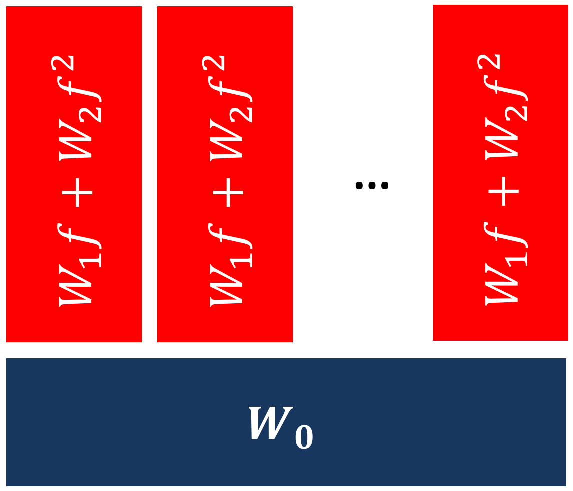




A simple power model for multicore chips

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
4. Performance is linear in the clock frequency unless it hits a bottleneck (simplification from performance models!)
5. Energy to solution is power dissipation divided by performance



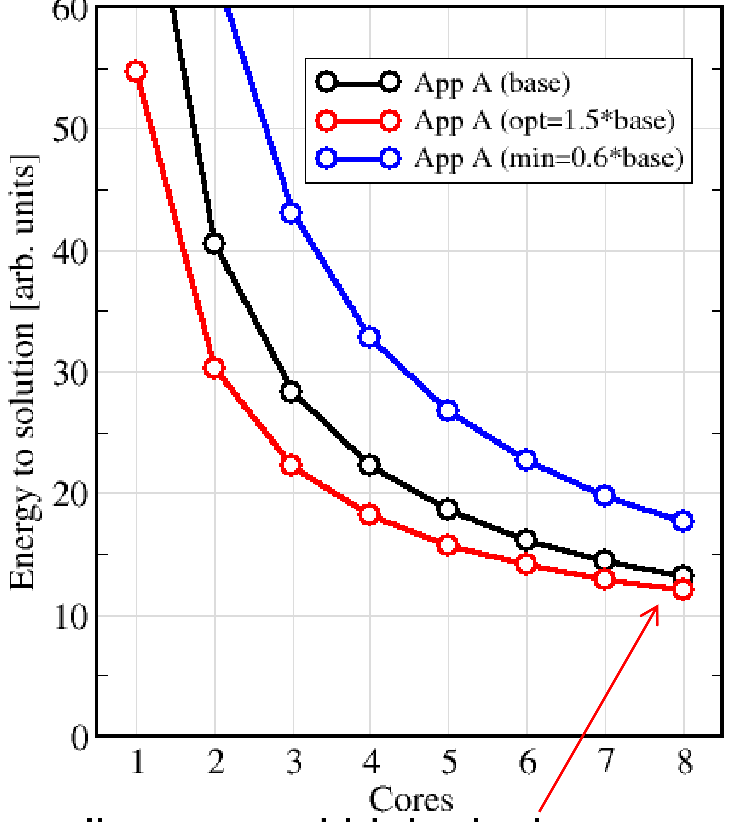
Model:

$$\boxed{E = \frac{\text{Power}}{\text{Performance}} = \frac{W_0 + (W_1 f + W_2 f^2)n}{\min(nP_0\, f/f_0, P_{max})}}$$

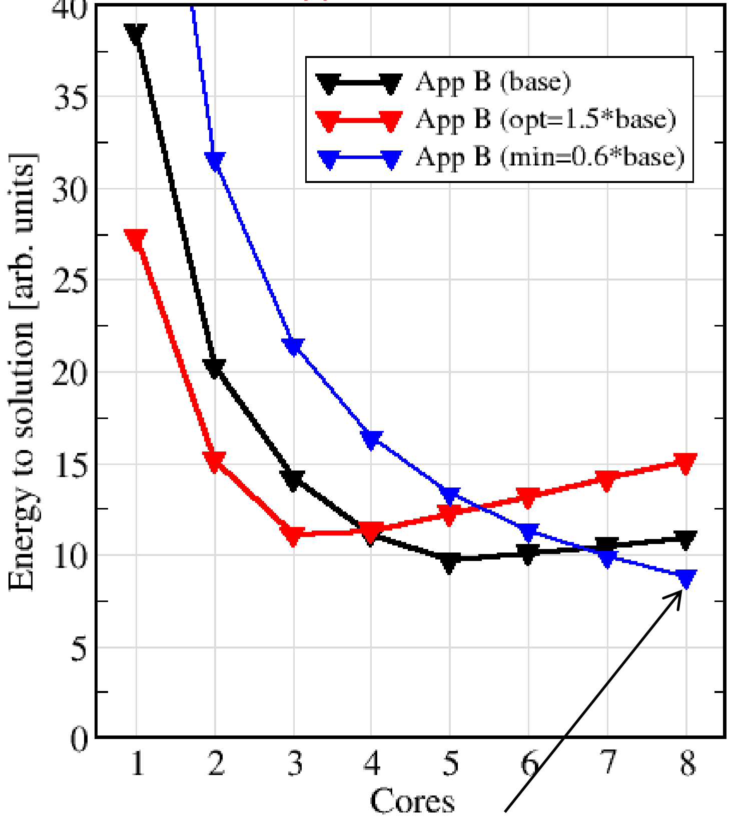
Energy to solution model: Observations

W_0	$= 73$ W	base	= 2 GHz
W_2	$= 1$ W / GHz^2	Turbo	= 3 GHz

LINPACK type



STREAM type



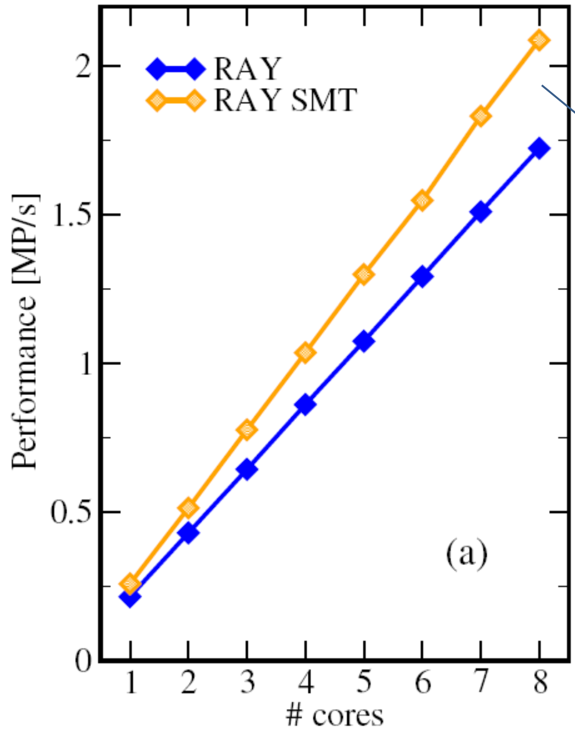
Use all cores and high clock speed!

Run all cores at clock speed that still saturates performance

Model predictions

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

1. If performance is linear in *n*, use all available cores to minimize *E*



Minimum E here

$$\frac{\partial E}{\partial n} = -\frac{W_0}{(1+\Delta\nu) n^2 P_0} < 0$$

Model predictions

$$E = \frac{W_0 + (W_1 f + W_2 f^2)n}{\min(nP_0\, f/f_0\,, P_{max})}$$

2. If performance is linear in n, there is an optimal frequency f_{opt} at which E is minimal:

$$f_{\text{opt}} = \sqrt{\frac{W_0}{W_2 n}}$$

Energy-frequency convexity rule

K. DeVogeleer, G. Memmi, P. Jouvelot, and F. Coelho: The Energy/Frequency Convexity Rule: modeling and experimental validation on mobile devices. In Proc. PPAM 2013, Springer, 2013.

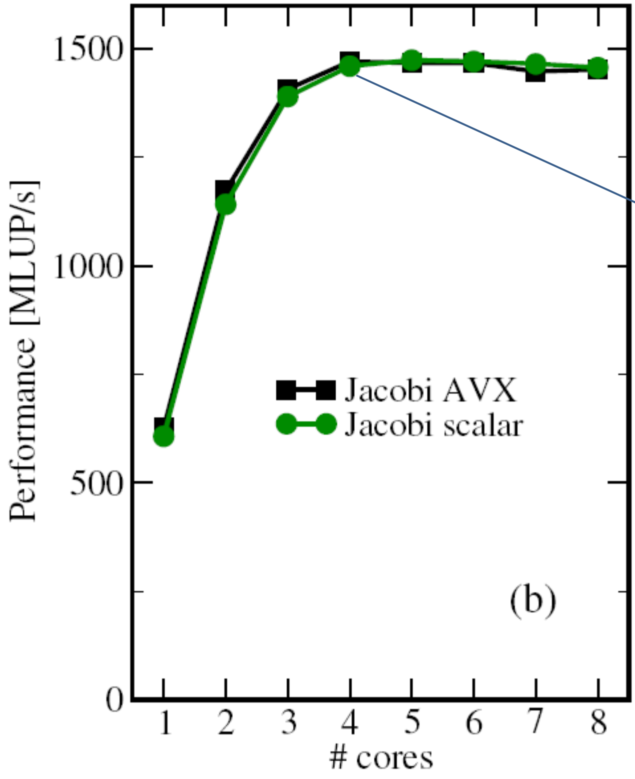
→ “Clock race to idle” if baseline power is large!
→ If $f_{\text{opt}} < f_0$, other target metrics may be suitable, e.g.,
$\boldsymbol{C = E/P}$: “Energy-Delay Product”

$$\frac{\partial C}{\partial \Delta\nu} = -\frac{2W_0 + W_1 f n}{\left(\frac{f}{f_0}\right)^3 P_0^2} < 0$$

Model predictions

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

3. If there is saturation, *E* is minimal at the saturation point



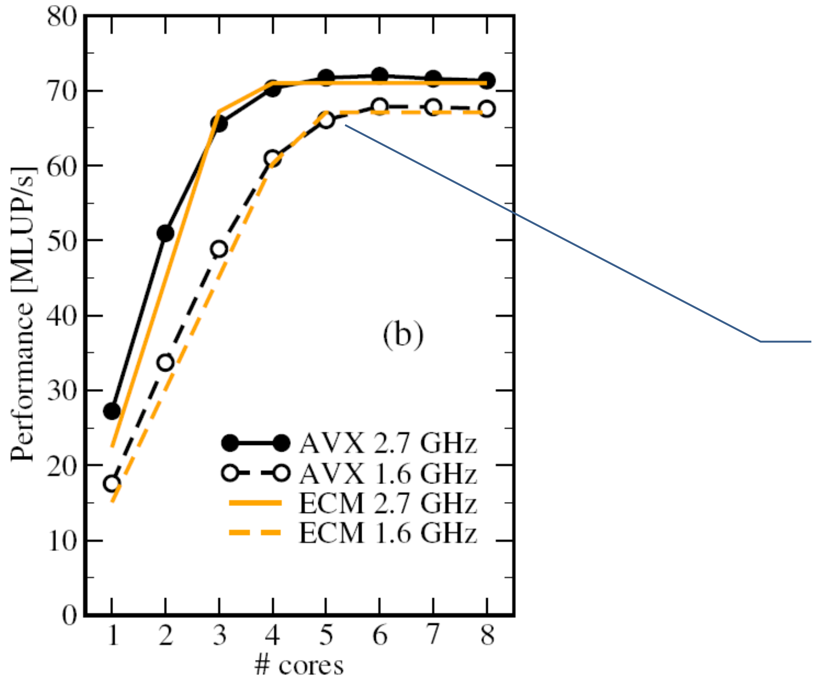
Minimum E here

$$n_s = \left\lfloor \frac{P_{max}}{P_0 f / f_0} \right\rfloor$$

Model predictions

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

4. If there is saturation, *E* shrinks if f is reduced for later saturation (larger n). E is minimal if f is reduced so that the saturation point is at the number of available cores.

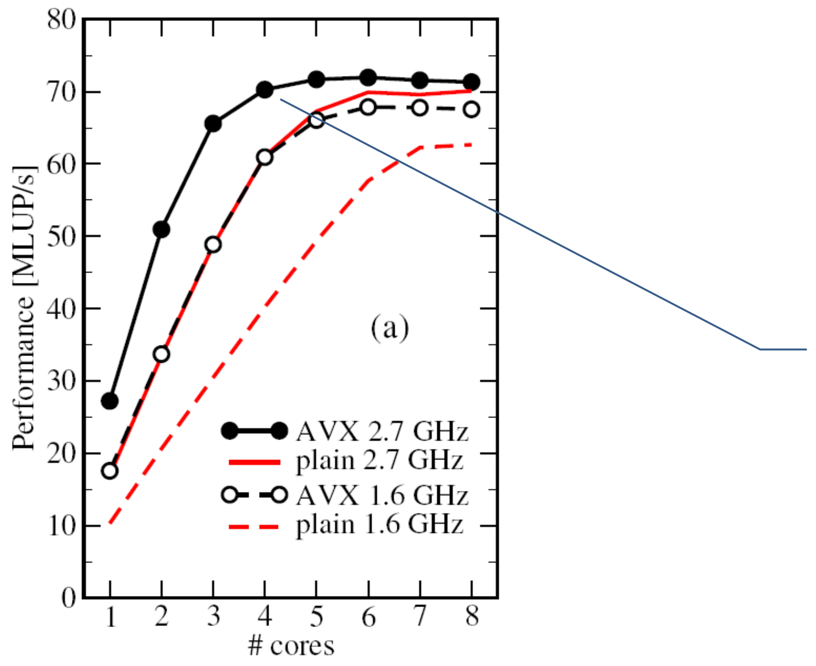


Slower clock
→ more cores to saturation
→ smaller E

Model predictions

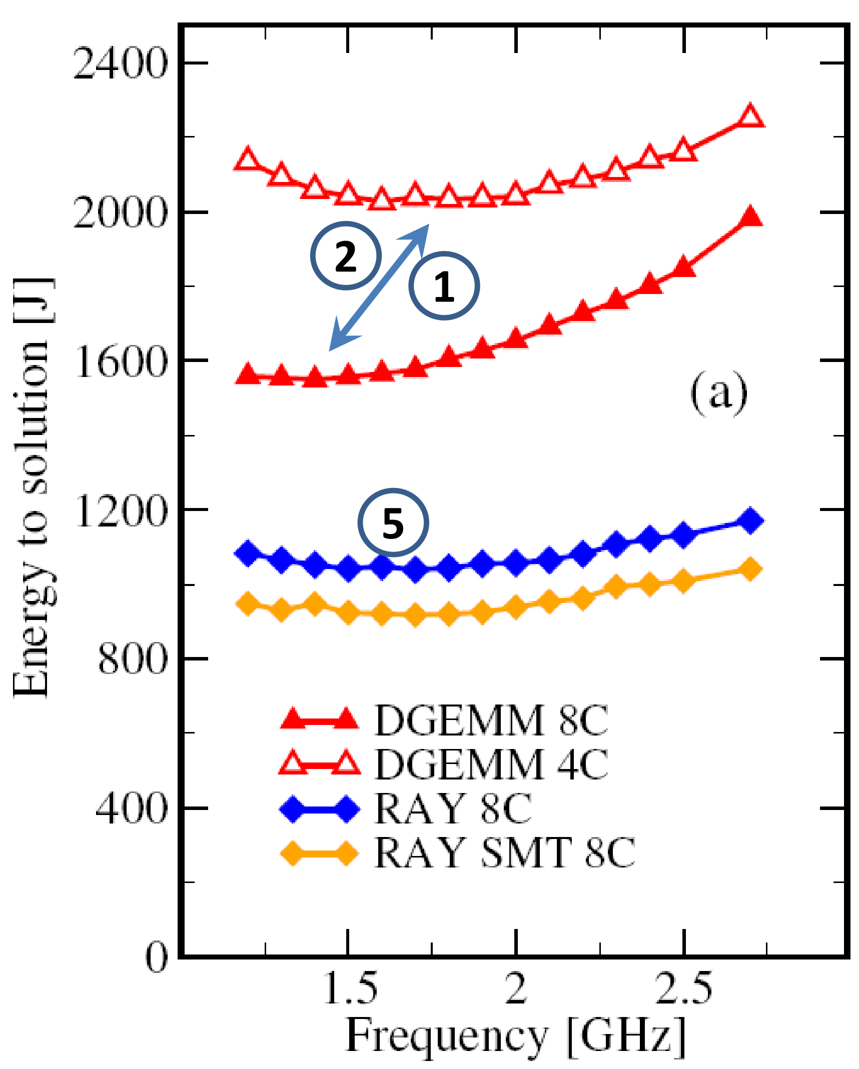
$$E = \frac{W_0 + (W_1 f + W_2 f^2)n}{\min(nP_0\, f/f_0\,, P_{max})}$$

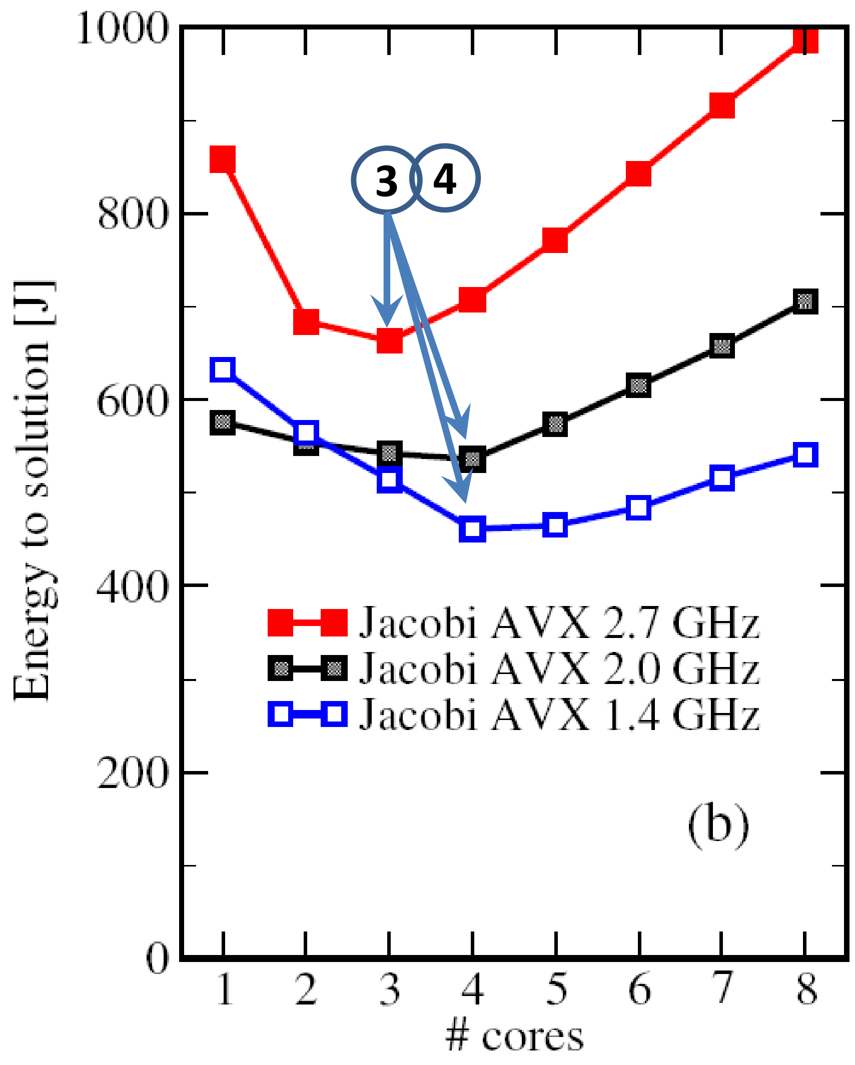
5. Making code execute faster on the core saves energy since
   - The time to solution is smaller if the code scales ("Code race to idle")
   - We can use fewer cores to reach saturation if there is a bottleneck



Better code
→ earlier saturation
→ smaller E @ saturation

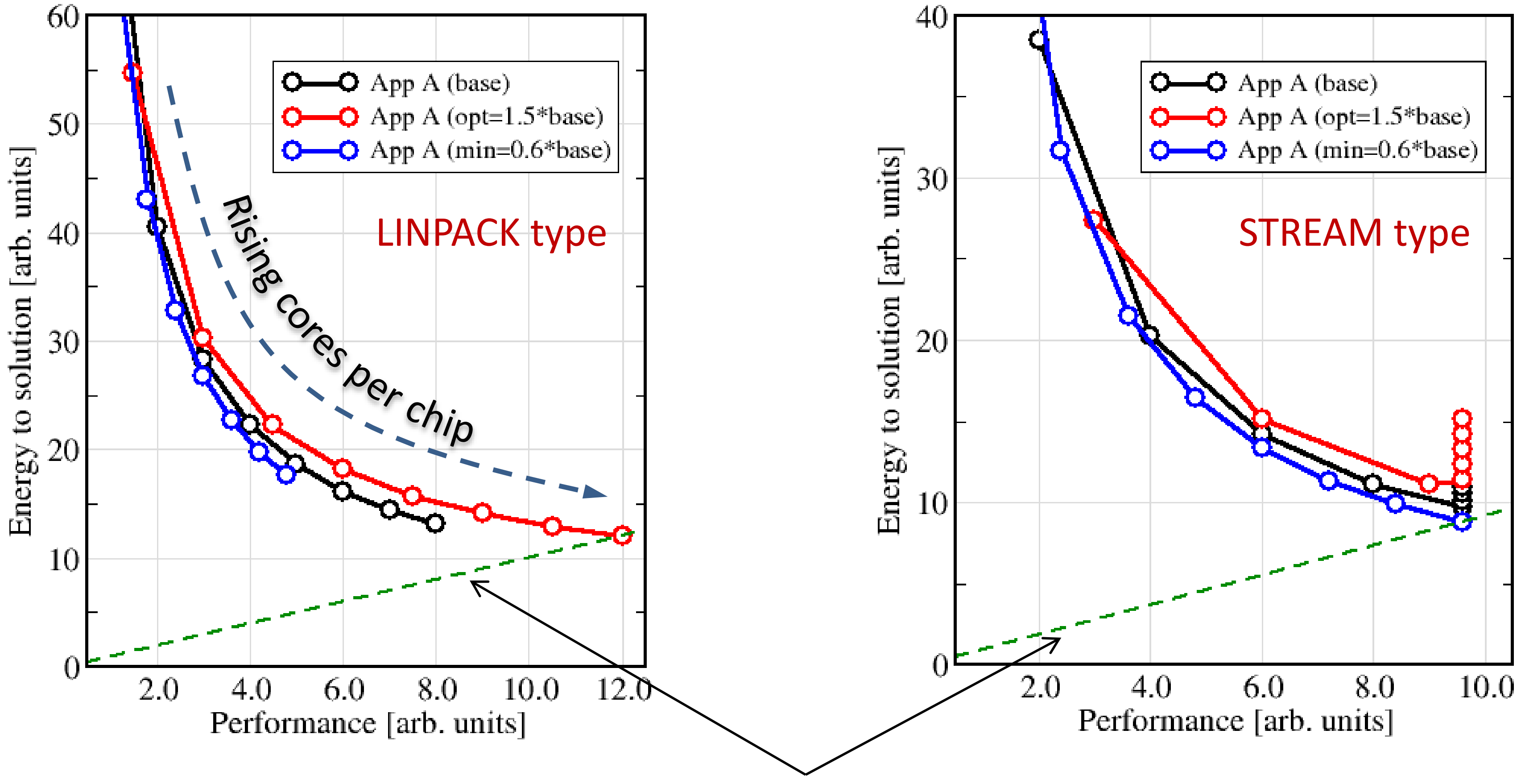
Model validation with the benchmark apps





Energy to Solution: A different way of presentation

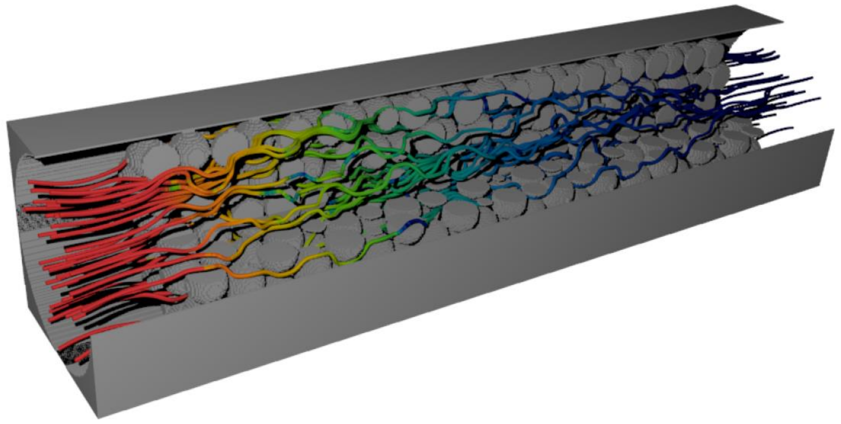
Energy vs. Performance ("Z-plot")

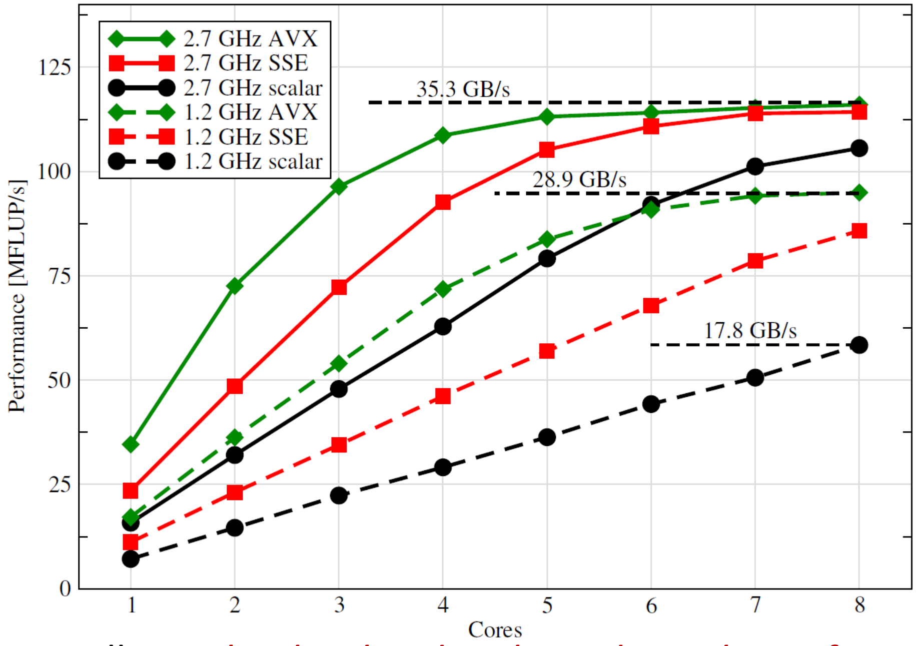


"Isoline" of constant Energy delay product ($E \times \Delta t$)

Case study: ILBDC Code

- Sparse representation lattice-Boltzmann flow solver
- Well suited for highly porous geometries, MPI parallel
- „AA pattern“ propagation → SIMD friendly, 304-376 bytes/LUP
- Saturating performance for vectorized code on modern Intel chips

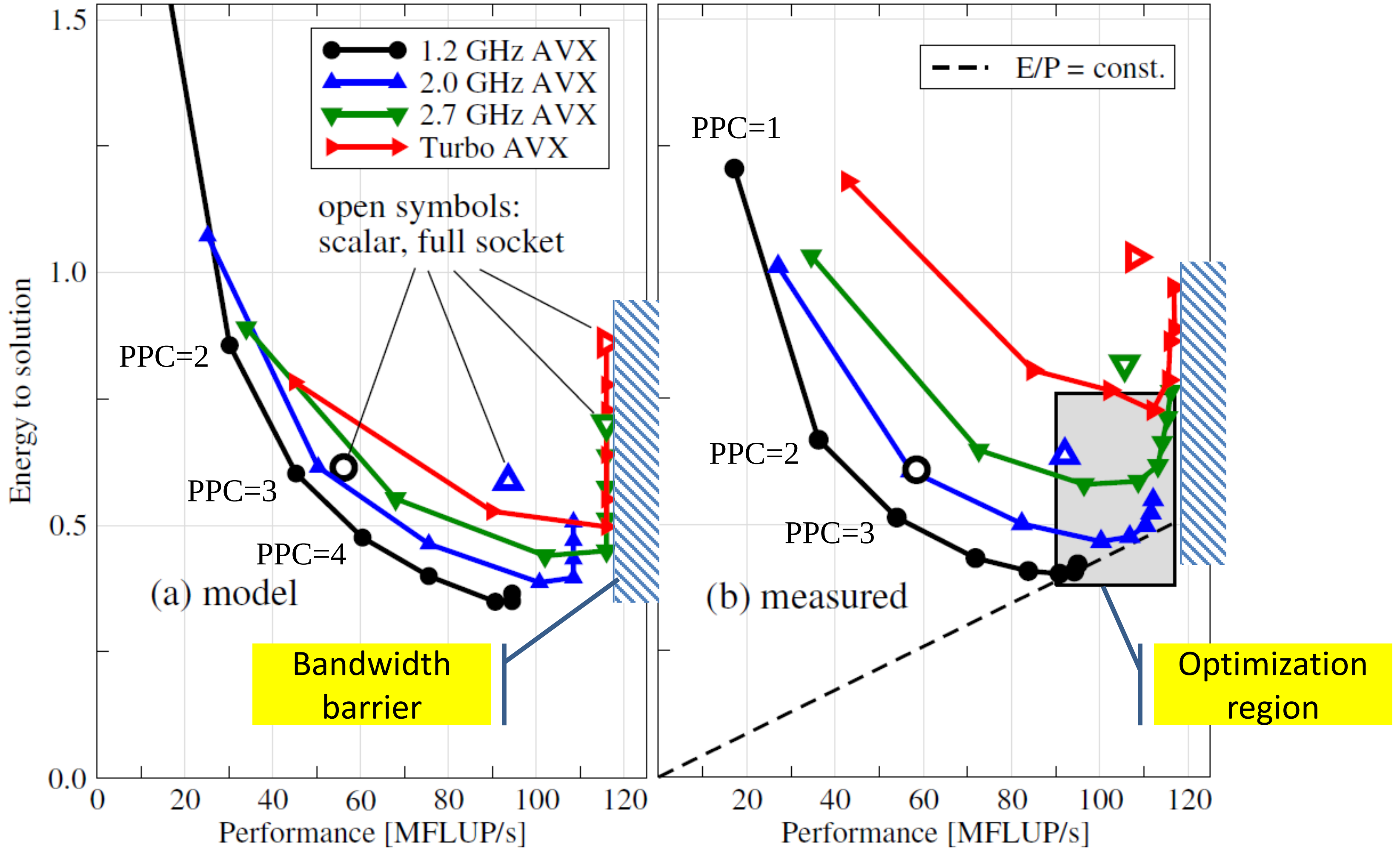




M. Wittmann, G. Hager, T. Zeiser, J. Treibig, and G. Wellein: *Chip-level and multi-node analysis of energy-optimized lattice-Boltzmann CFD simulations*. Concurrency and Computation: Practice and Experience (2015). [DOI: 10.1002/cpe.3489](https://doi.org/10.1002/cpe.3489) Preprint: [arXiv:1304.7664](https://arxiv.org/abs/1304.7664)

Energy to solution vs. performance on the socket (SNB)

Model vs. Measurement at different clock speeds (PPC=proc.s per chip)



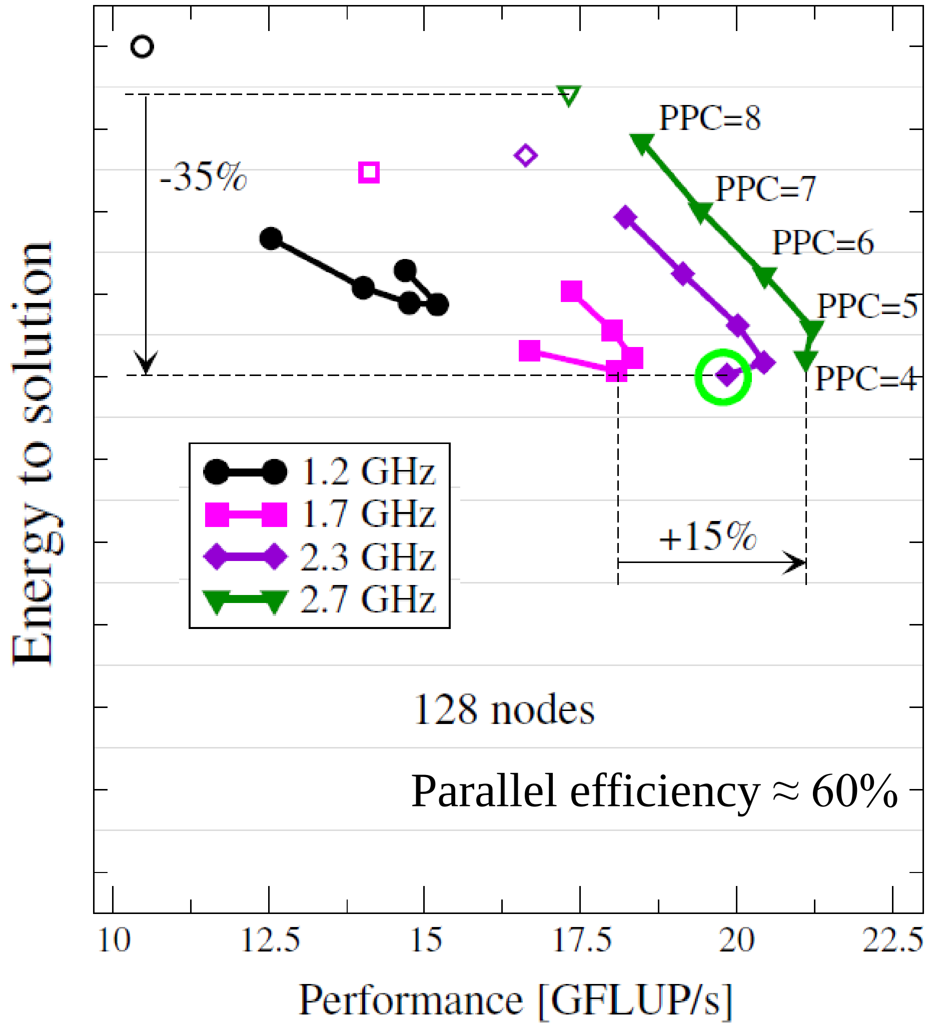
Energy to solution vs. performance on the cluster (SuperMUC)

How does that change when going multi-node with substantial communication overhead?

- Dependence on socket-level concurrency?
- Dependence on clock speed?

Observations:

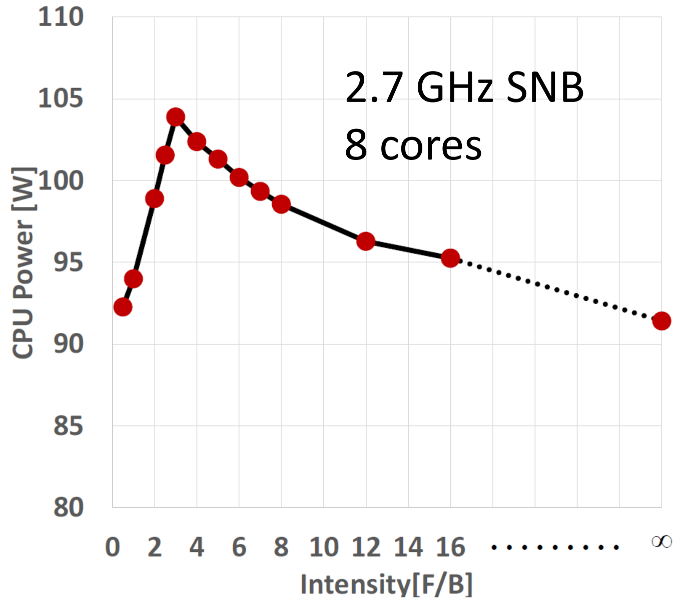
- Optimal PPC is crucial for lowest energy!
- Higher clock speed yields better performance without energy penalty!

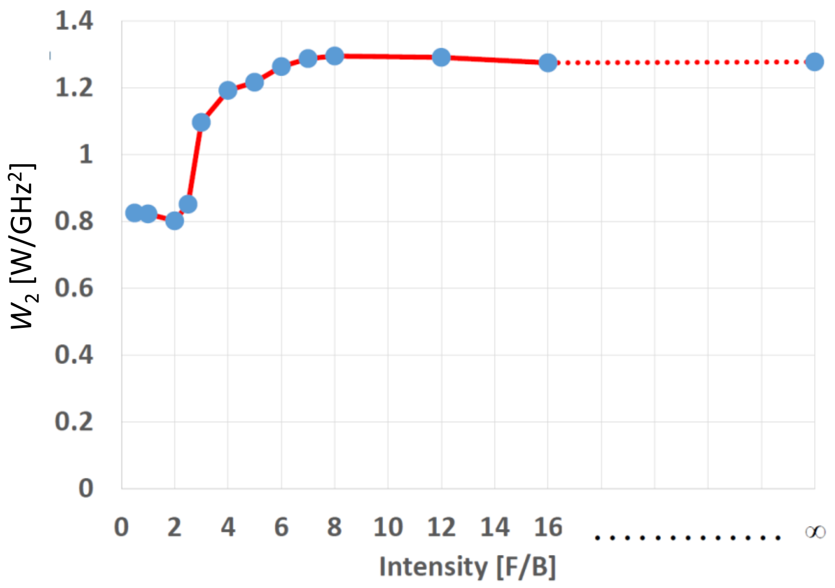


Can we predict/calculate the model parameters?

Where do W_1 and W_2 come from?

→Depend on hardware and code characteristics





Connection to microscopic models is possible, e.g.:

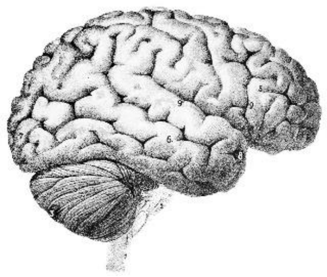
J.W. Choi et al.: A Roofline Model of Energy. Proc. IPDPS 2013
DOI: 10.1109/IPDPS.2013.77

Conclusions & outlook

- White-box performance modeling generates insight into the interaction of hardware and software
- Roofline model is a good start, but more advanced models exist
  - 100% accuracy is not required
- Multicore energy consumption is a function of very few parameters
  - 100% accuracy is not required
- Simple modeling techniques and patterns help us
  - … understand the limits of our code on the given hardware
  - … identify optimization opportunities
  - … learn more, especially when they do not work!

- Problems of white-box analytical modeling
  - Assumes steady state situation (loops)
  - Complex code → lots of tedious work, but there is a reward!

- Simple tools get you 95% of the way!
  - E.g., LIKWID: http://tiny.cc/LIKWID



Thank you.

Bundesministerium
für Bildung
und Forschung

hpcADD

KONWIHR III

OMI4papps

Save the date!

ISC15 Workshop:

Performance Modeling: Methods and Applications

ISC15, Frankfurt, Germany, July 16, 9:00-18:00

Speakers: Bill Gropp (UIUC, keynote), Nathan Tallent (PNNL), Dimitrios Nikolopoulos (Belfast), Martin Schulz (LLNL), Laura Carrington (SDSC), Jeffrey Vetter (ORNL), Felix Wolf (Darmstadt), Alexander Grebhahn (Passau), Robert Numrich (CUNY), Rich Vuduc (GATech), Brian van Straalen (LBNL), Georg Hager (RRZE)