Performance and Power Engineering on Multicore Processors

Georg Hager, Jan Treibig, Gerhard Wellein
Erlangen Regional Computing Center (RRZE)
University of Erlangen-Nuremberg
Erlangen, Germany

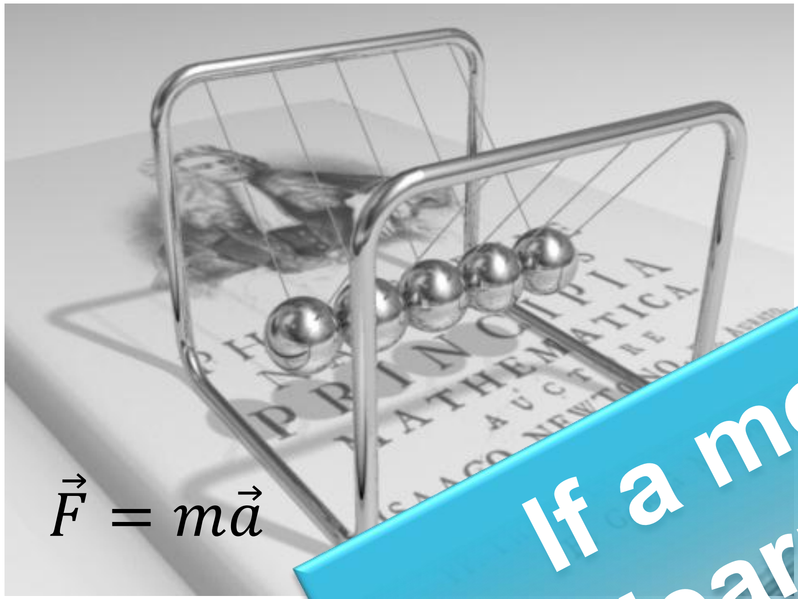
GRS-Sim Aachen
March 11, 2013

Outline

- Performance Modeling and Engineering
  - Motivation
  - “White Box” models: Roofline
- Example: Sparse MVM
- “If the model doesn’t work, we learn something”
  - A starting point for refining Roofline
- The ECM multi-core model
- A simple power model for multicore

An example from physics

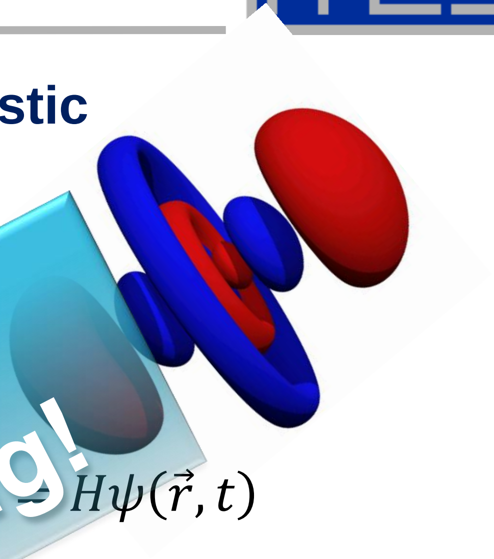
Newtonian mechanics



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

Fails @ small scales!

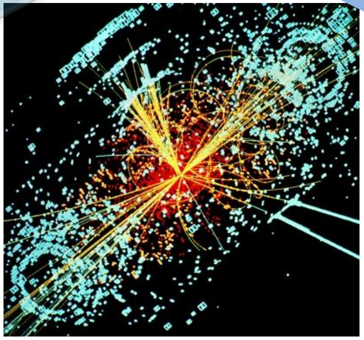
Nonrelativistic quantum mechanics



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

Fails @ even smaller scales!

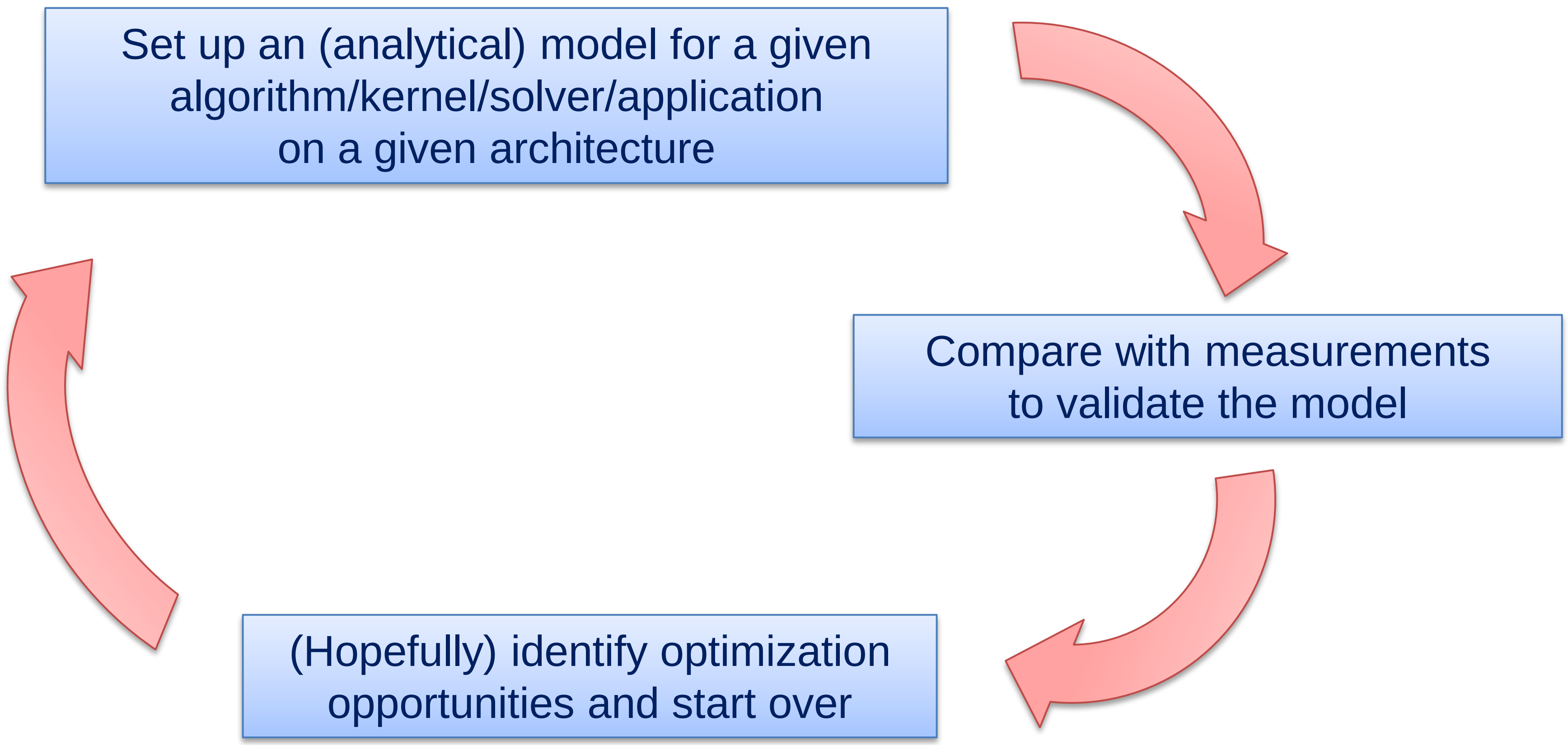
If a model fails, we learn something!



Relativistic quantum field theory

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

White box performance modeling

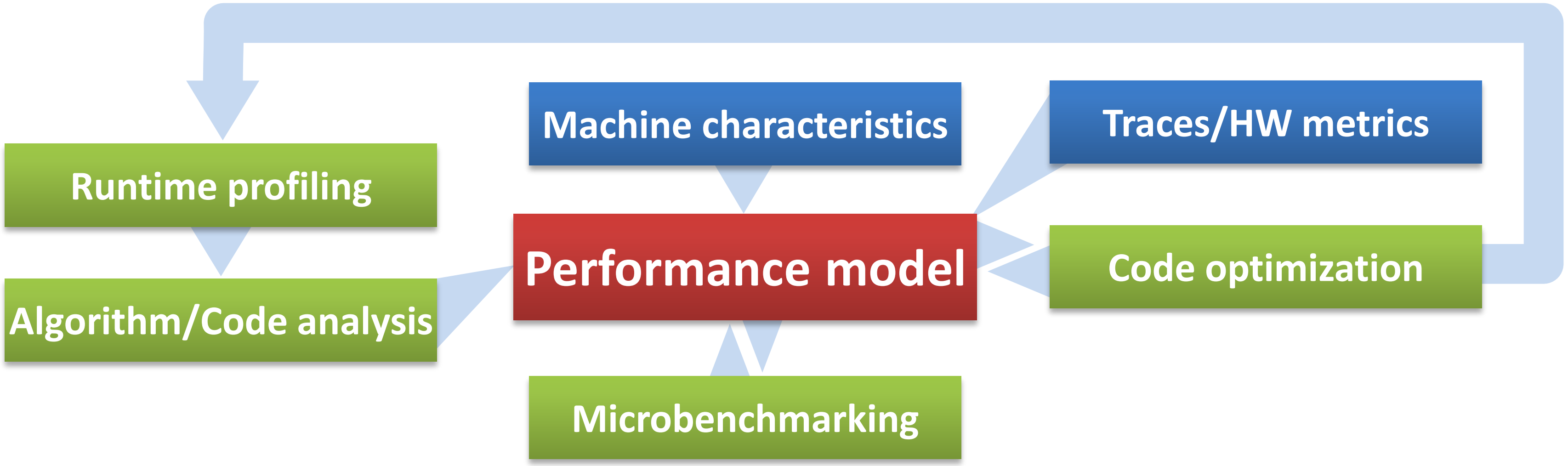


Set up an (analytical) model for a given algorithm/kernel/solver/application on a given architecture

Compare with measurements to validate the model

(Hopefully) identify optimization opportunities and start over

The Performance Engineering (PE) process



“White Box” Performance Models on the Chip Level

Roofline model

ECM model

An example: The Roofline Model[1,2]

1. $P_{\max}$ = Applicable peak performance of a loop, assuming that data comes from L1 cache

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:

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

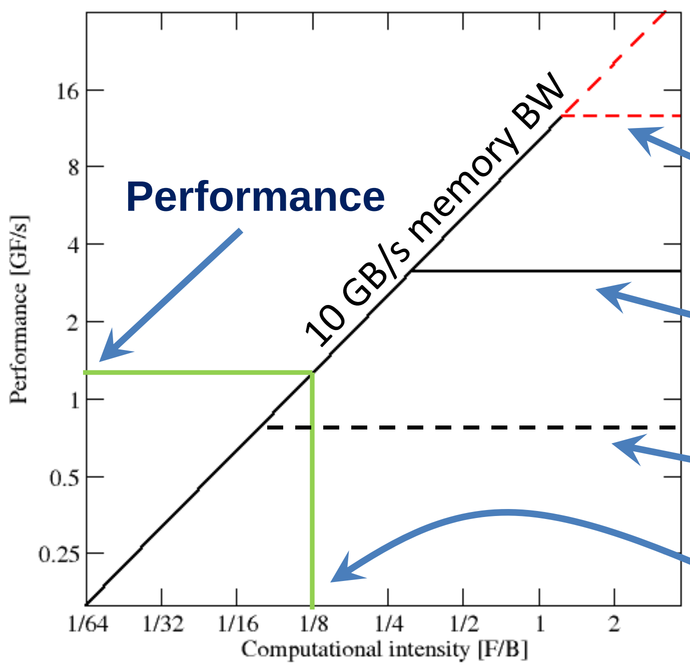
[1] W. Schönauer: Scientific Supercomputing: Architecture and Use of Shared and Distributed Memory Parallel Computers. (2000)
[2] S. Williams: Auto-tuning Performance on Multicore Computers. UCB Technical Report No. UCB/EECS-2008-164. PhD thesis (2008)

A simple Roofline example

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

in double precision on hypothetical 3 GHz CPU, 4-way SIMD, N large

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



ADD peak (half of full peak)

4-cycle latency per ADD if not unrolled

no SIMD

Computational intensity

Roofline Model assumptions

- 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
- Attainable bandwidth of code = input parameter!
  - Microbenchmarking may be required

- Data transfer and core execution overlap perfectly!
- "Applicable peak" can be calculated accurately
- Bottleneck is modeled only; all others are assumed to be infinitely fast

- If data transfer is the limiting factor, the bandwidth of the slowest data path can be utilized to 100% ("saturation")

Using Roofline in a More Complex Setting

Sparse matrix-vector multiply (spMVM)

Example: SpMVM node performance model

- Sparse MVM in double precision w/ CRS data storage:

```
do i = 1,$N_r$
  do j = row_ptr(i), row_ptr(i+1) - 1
    C(i) = C(i) + val(j) * B(col_idx(j))
  enddo
enddo
```

- DP CRS comp. intensity
  - κ quantifies extra traffic for loading RHS more than once

$$
\begin{aligned}
I_{\mathrm{CRS}} &= \frac{2}{\boxed{12}+\boxed{24}/N_{\mathrm{nzr}}+\boxed{\kappa}}\,\frac{\mathrm{Flops}}{\mathrm{Byte}}\\
&= \left(6+\frac{12}{N_{\mathrm{nzr}}}+\frac{\kappa}{2}\right)^{-1}\frac{\mathrm{Flops}}{\mathrm{Byte}}
\end{aligned}
$$

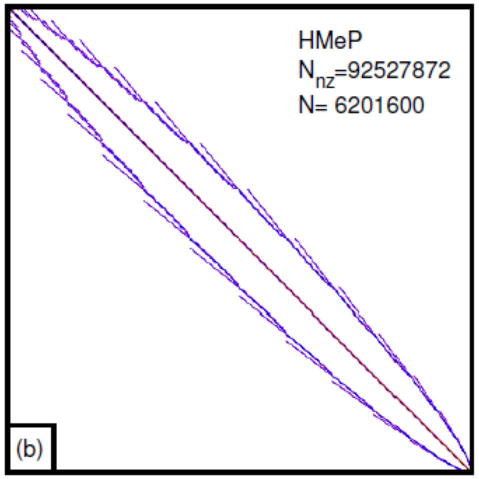
  - Predicted Performance = streamBW·I_{CRS}

  - Determine κ by measuring performance and actual memory bandwidth

Test matrices: Sparsity patterns

- Analysis for HMeP matrix on Nehalem EP socket
  - BW used by spMVM kernel = 18.1 GB/s → should get ≈ 2.66 Gflop/s spMVM performance if $\kappa = 0$
  - Measured spMVM performance = 2.25 Gflop/s
  - Solve 2.25 Gflop/s = BW·I_{CRS} for $\kappa \approx 2.5$

    → 37.5 extra bytes per row
    → RHS is loaded 6 times from memory
    → about 33% of BW goes into RHS



- Conclusion: Even if the roofline model does not work 100%, we can still learn something from the deviations

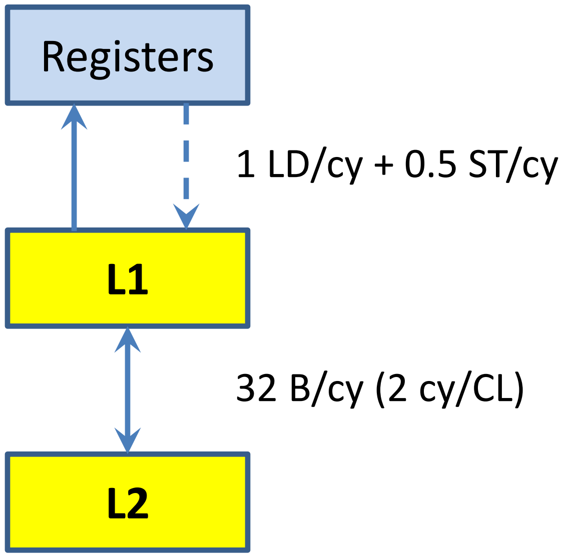
“If the model fails, we learn something”

In-core analysis of the Schönauer triad on Sandy Bridge

Example: Schönauer Vector Triad in L2 cache

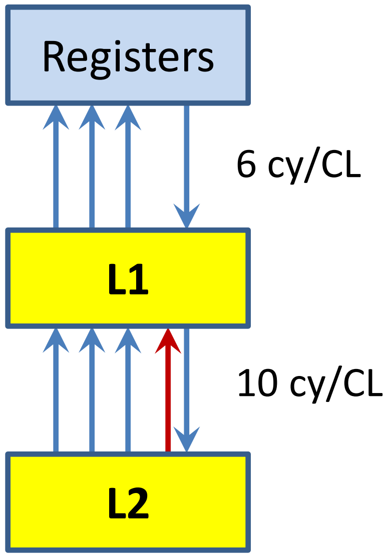
- REPEAT[`A(:) = B(:) + C(:) * D(:)`] @ double precision
- Analysis for Sandy Bridge core w/ AVX (unit of work: 1 cache line)

Machine characteristics:



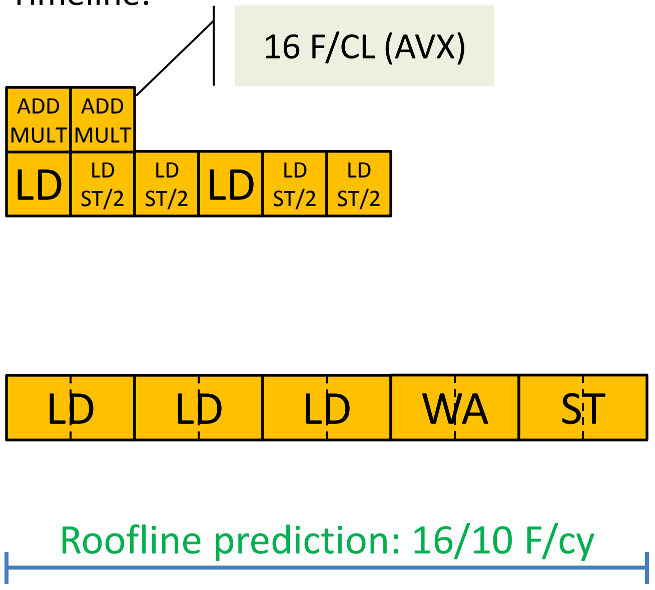
Arithmetic:
1 ADD/cy+ 1 MULT/cy

Triad analysis (per CL):



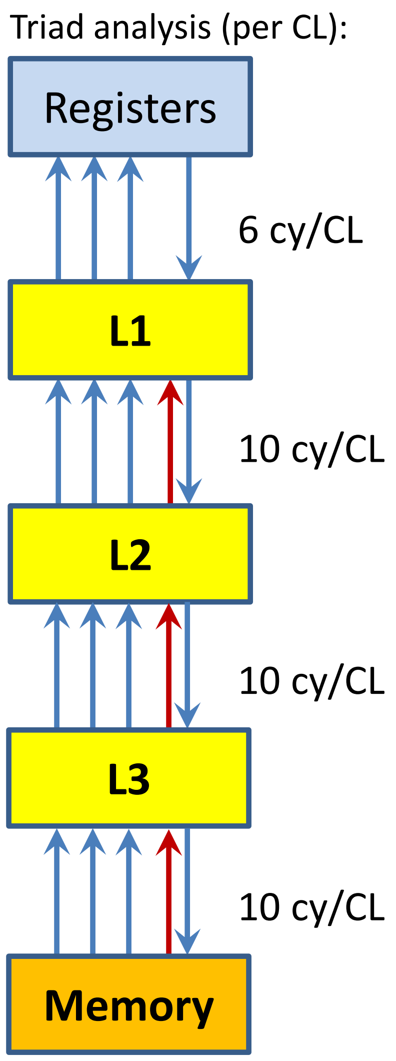
Arithmetic:
AVX: 2 cy/CL
SSE: 4 cy/CL

Timeline:



Measurement: 16F / ≈17cy

Schönauer Vector Triad in L2 cache



- No overlap of evict/refill with LD/ST in L1
  - L1 is “single ported”
- Other cache levels similar?
- How about overlap further down the hierarchy?
  - May be possible to get lower/upper performance bounds

→ Model for single-core execution with data from all levels of the hierarchy!

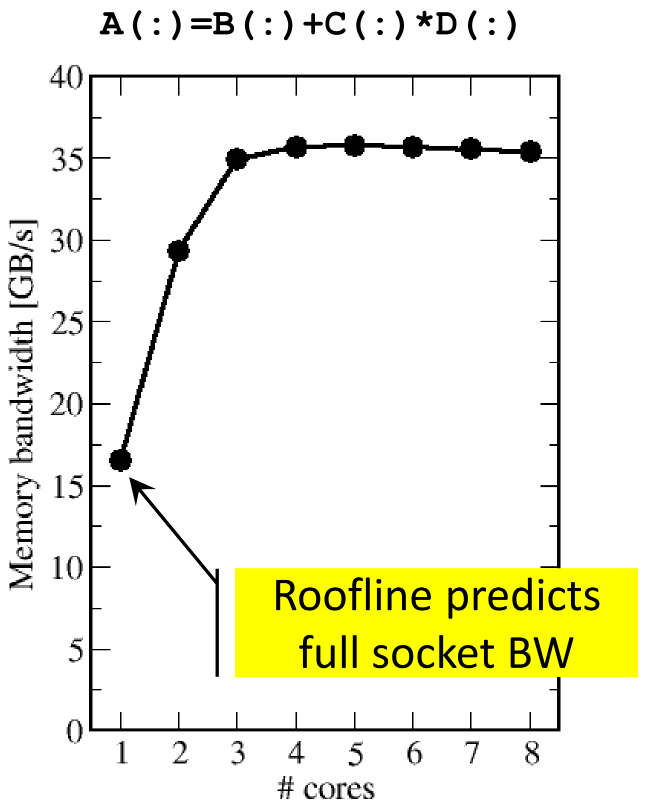
An Improved Performance Model for Multicore

The ECM Model

Roofline sometimes fails for multicore

- Assumes one of two bottlenecks
  1. In-core execution
  2. Bandwidth of a single hierarchy level

- Saturation effects in multicore chips are not explained

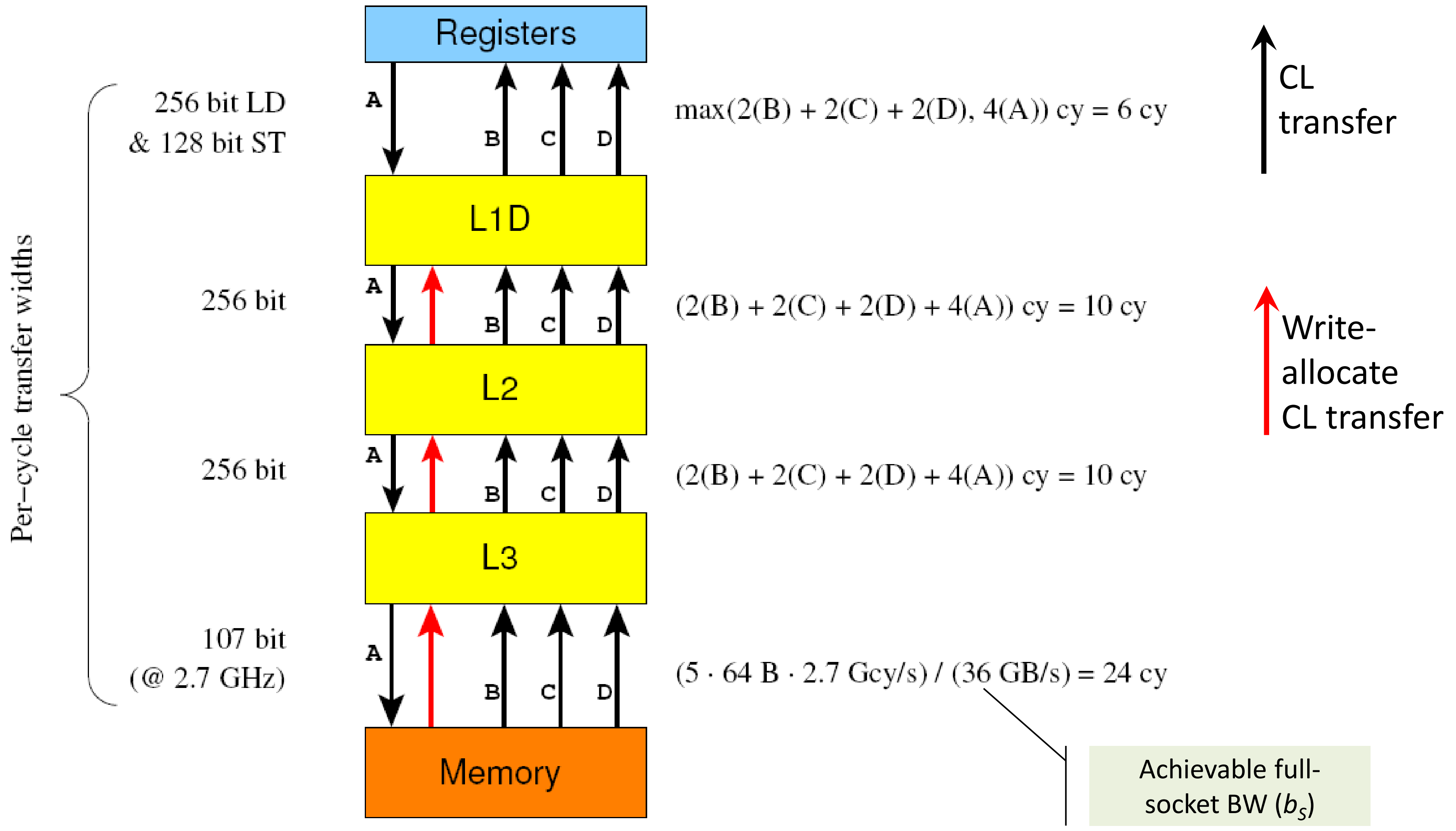


The multicore saturation mystery

- Why is serial performance “too low?”
  - Non-overlapping contributions from data transfers and in-cache execution to overall runtime
- What determines the saturation point?
  - Important for energy efficiency
  - Putting cores to better use
  - Saturation == Bandwidth pressure on relevant bottleneck exhausts the maximum BW capacity
- Requirements for an appropriate multicore performance model
  - Should predict single-core performance
  - Should predict saturation point

→ ECM (Execution – Cache – Memory) model

Example: ECM model for Schönauer Vector Triad `A(:)=B(:)+C(:)*D(:)` on a Sandy Bridge Core with AVX



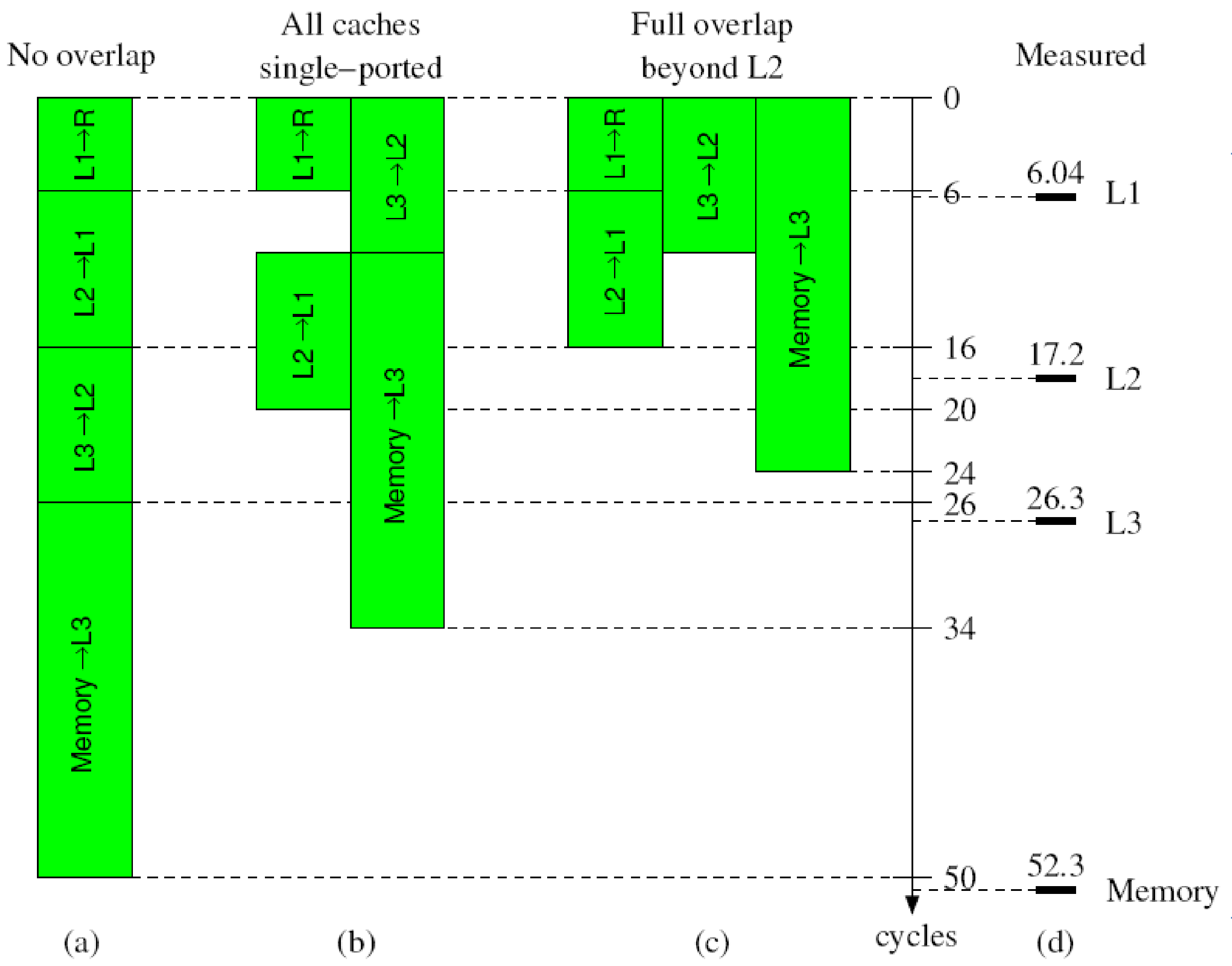
max(2(B) + 2(C) + 2(D), 4(A)) cy = 6 cy

(2(B) + 2(C) + 2(D) + 4(A)) cy = 10 cy

(2(B) + 2(C) + 2(D) + 4(A)) cy = 10 cy

$(5 \cdot 64\ \mathrm{B} \cdot 2.7\ \mathrm{Gcy/s}) / (36\ \mathrm{GB/s}) = 24\ \mathrm{cy}$

Full vs. partial vs. no overlap



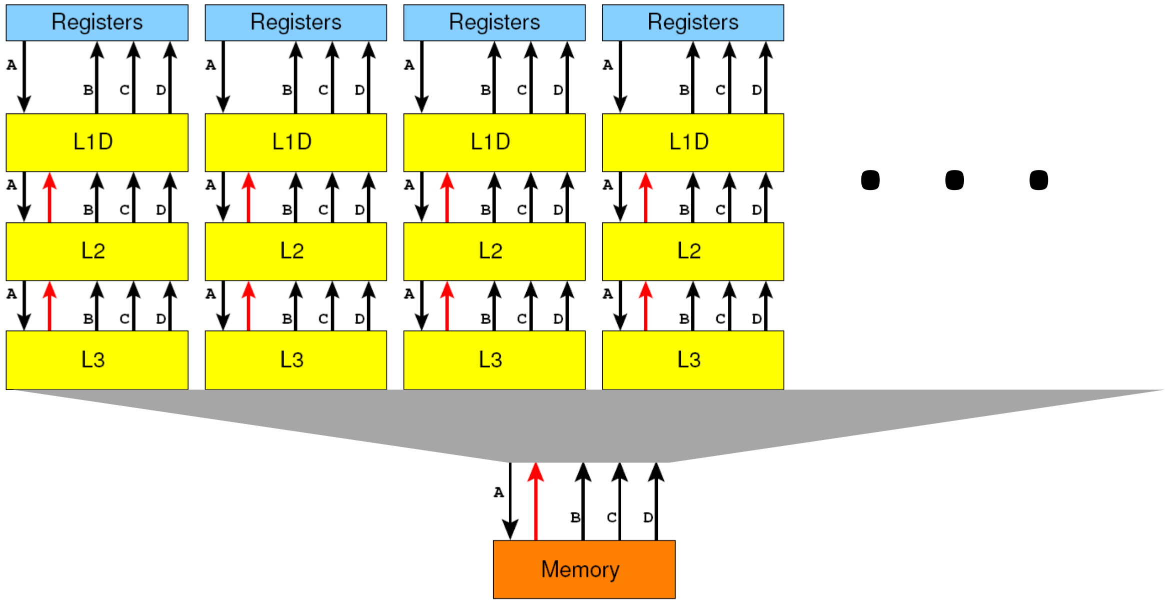
Results suggest no overlap!

Multicore scaling in the ECM model

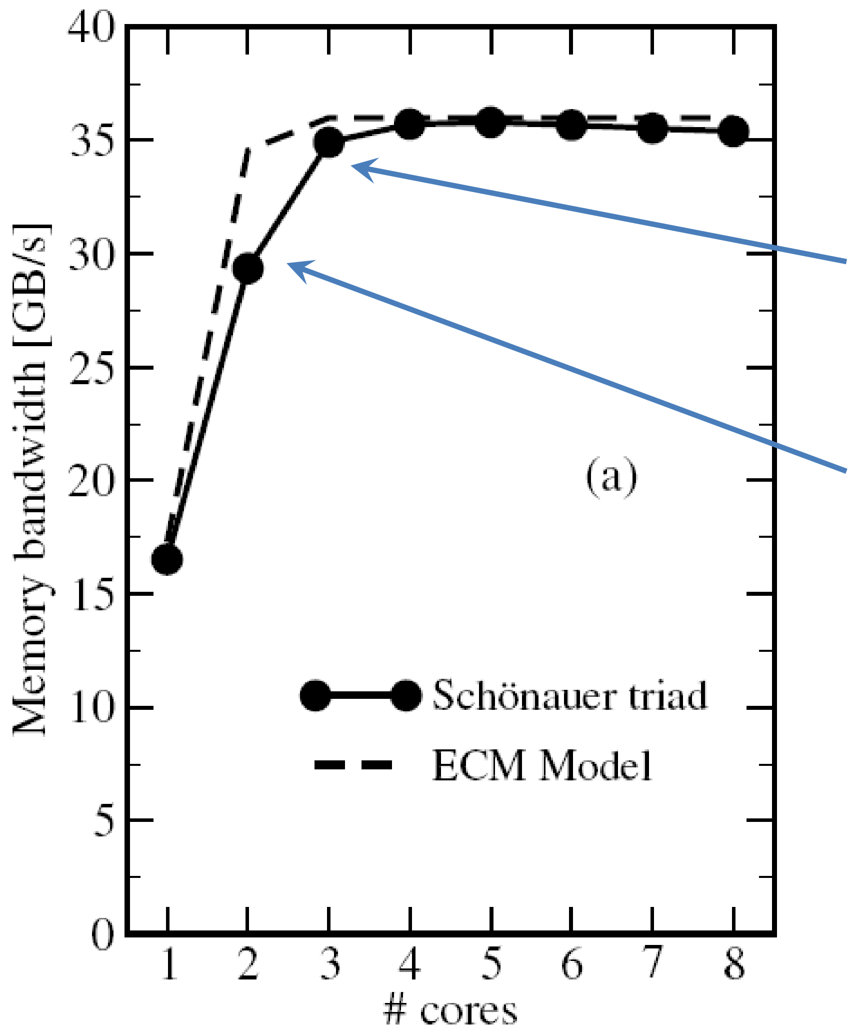
- Identify relevant bandwidth bottlenecks
  - L3 cache
  - Memory interface
- Scale single-thread performance (P_0) until first bottleneck is hit:

$$P(n_t) = \min(n_t P_0, P_{\mathrm{roof}}), \text{ with } P_{\mathrm{roof}} = \min(P_{\max}, I \cdot b_S)$$

Example: Scalable L3 on Sandy Bridge



ECM prediction vs. measurements for `A(:)=B(:)+C(:)*D(:)` on a Sandy Bridge socket (no-overlap assumption)



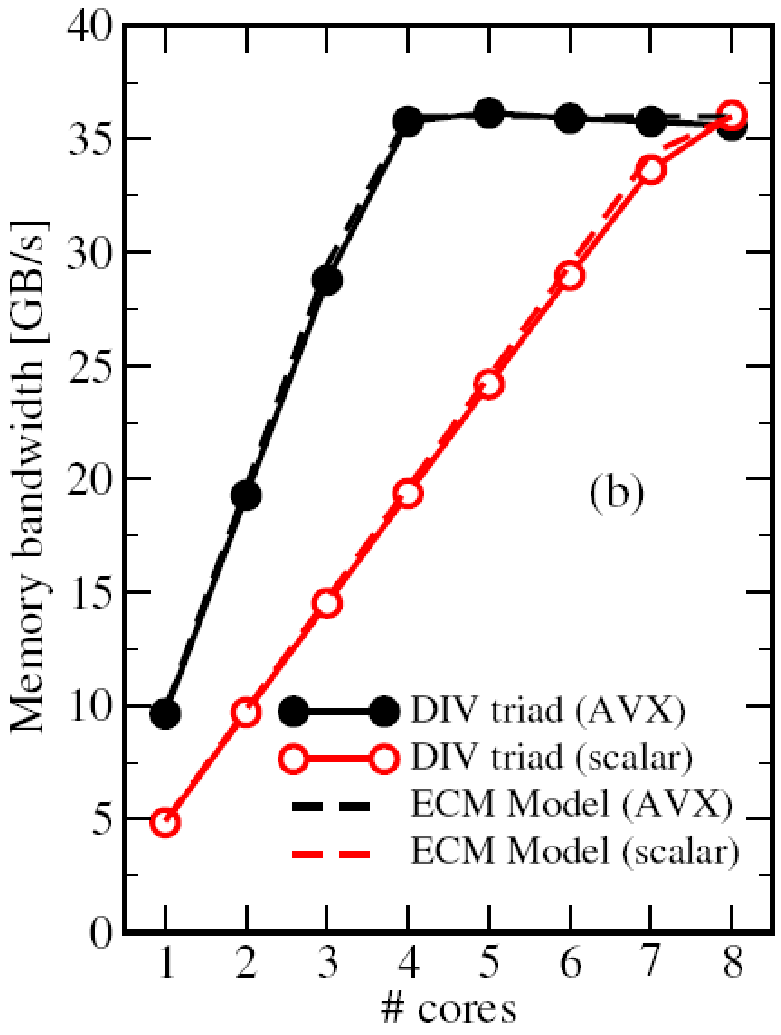
Saturation point (# cores) well predicted

Measurement: scaling not perfect

Caveat: This is specific for this architecture and this benchmark!

Check: Use “overlappable” kernel code

ECM prediction vs. measurements for `A(:)=B(:)+C(:)/D(:)` on a Sandy Bridge socket (full overlap assumption)

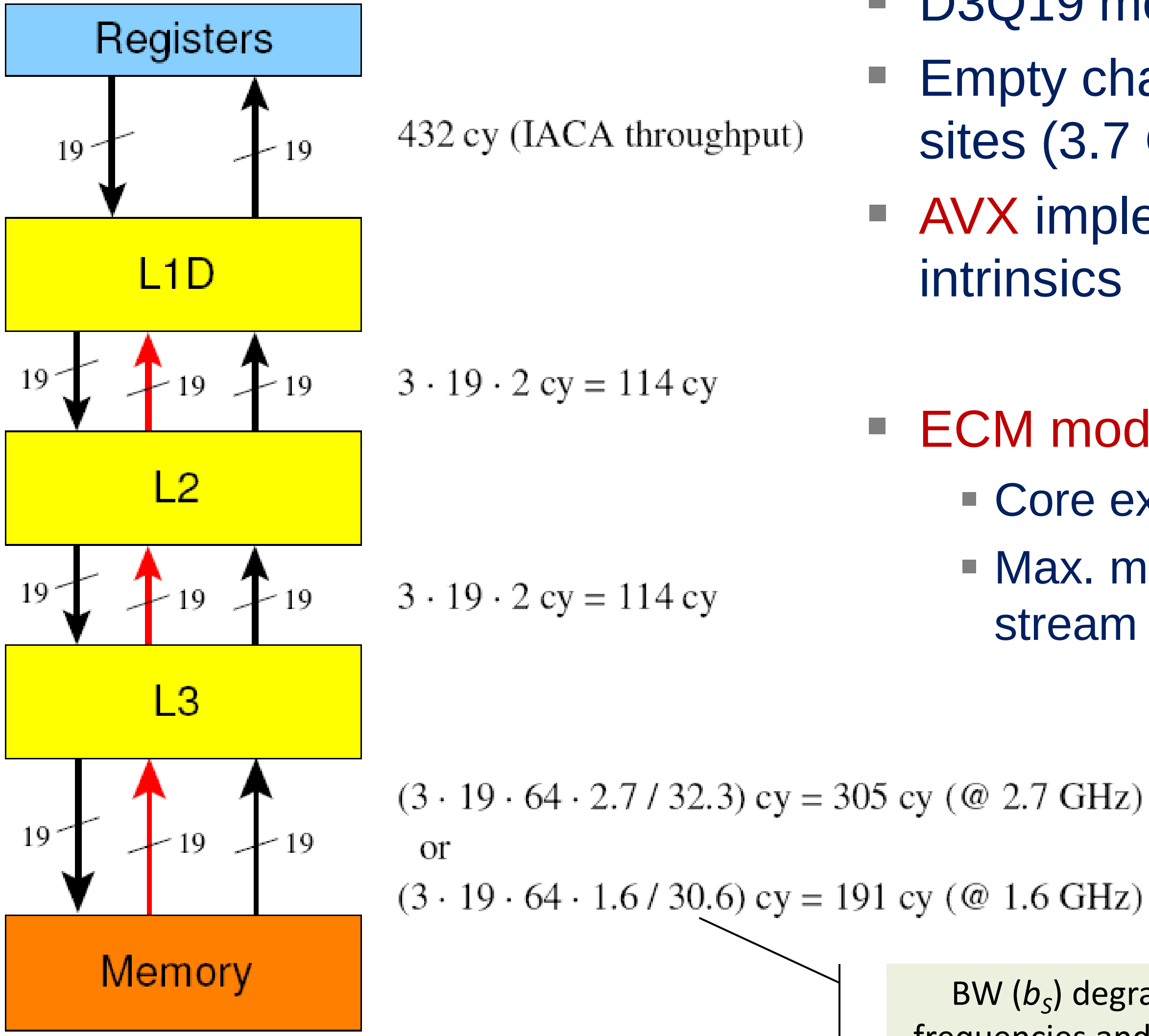


In-core execution is dominated by divide operation

(44 cycles with AVX, 22 scalar)

→ Almost perfect agreement with ECM model

Example: Lattice-Boltzmann flow solver



- D3Q19 model
- Empty channel, 228^3 fluid lattice sites (3.7 GB of memory)
- AVX implementation with compiler intrinsics

- ECM model input
  - Core execution from Intel IACA tool
  - Max. memory bandwidth from multi-stream measurements

BW (b_S) degradation @ lower frequencies and large # of streams

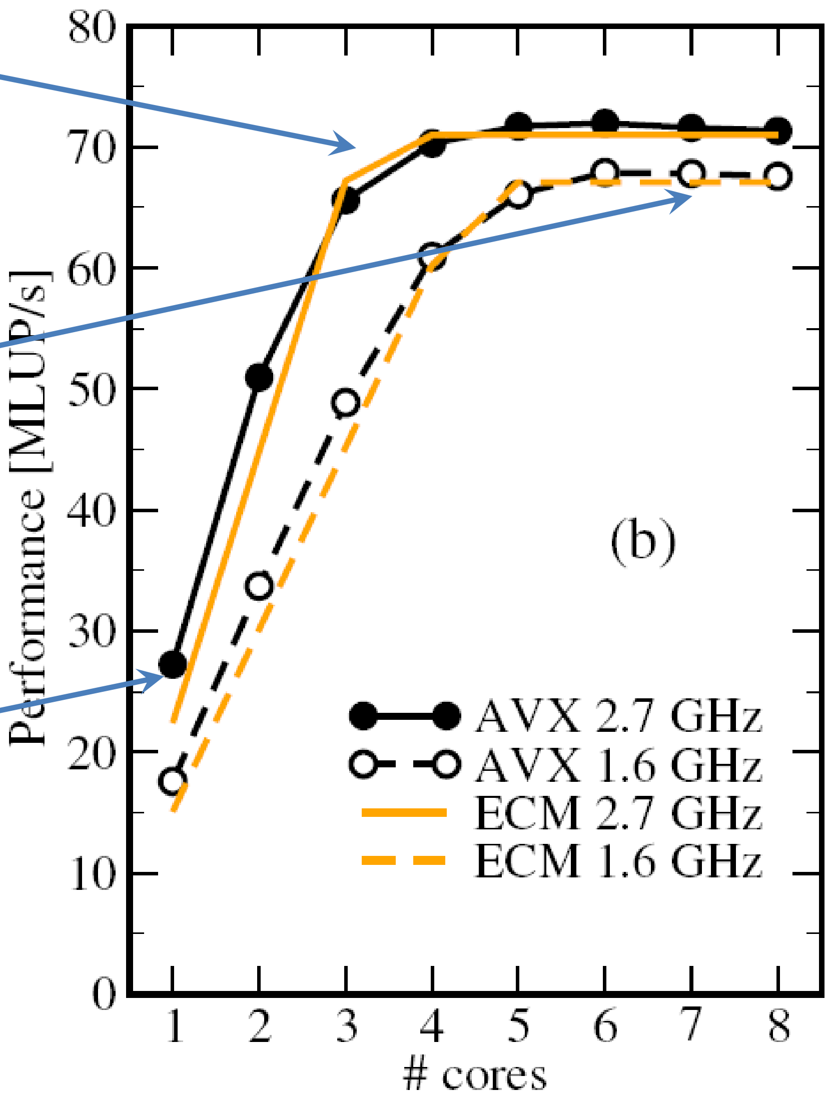
Lattice-Boltzmann solver: ECM (no-overlap) vs. measurements

Saturation point again predicted accurately

Saturation performance matches multi-stream benchmarks (by construction)

No-overlap assumption seems a little pessimistic

- Not all execution is LD and ST (IACA predicts ADD bottleneck)



Why the fuss about the saturation point?

(1) Putting cores to good use

(2) Energy consumption

A simple power model for the Sandy Bridge processor

G. Hager, J. Treibig, J. Habich, and G. Wellein: *Exploring performance and power properties of modern multicore chips via simple machine models*. Submitted. Preprint: arXiv:1208.2908

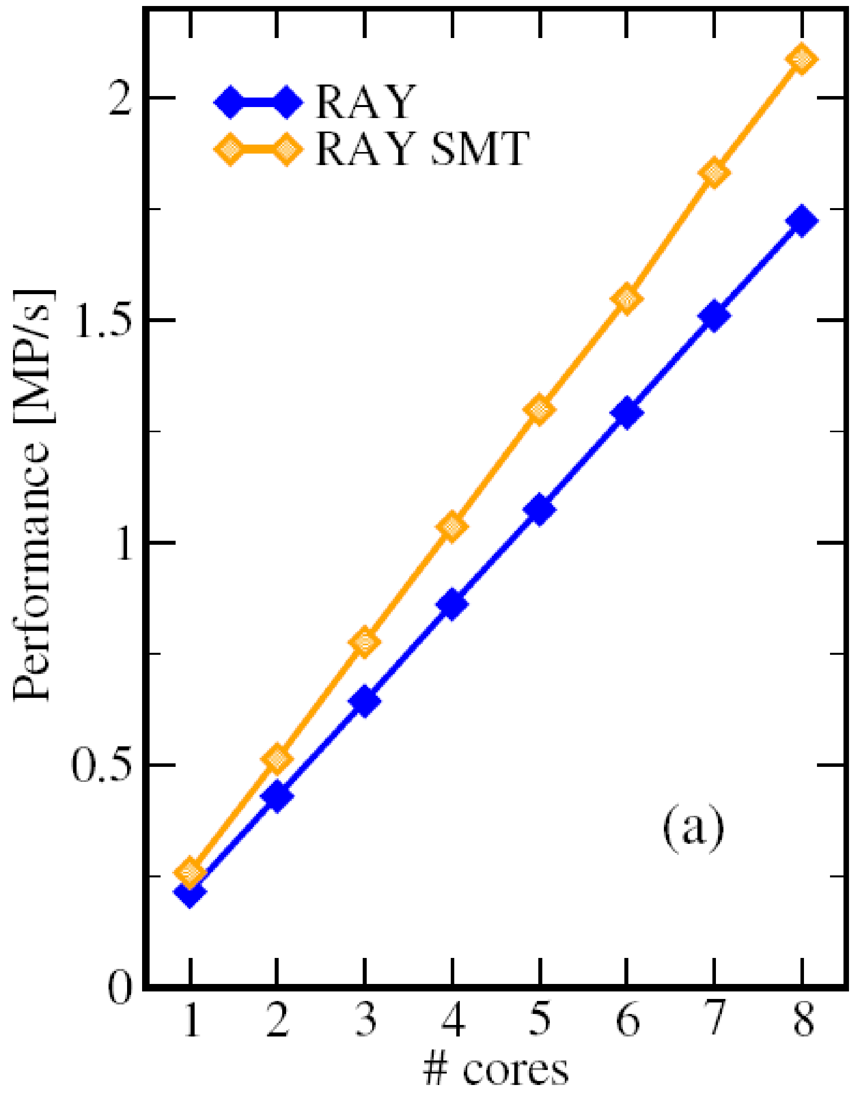
A model for multicore chip power

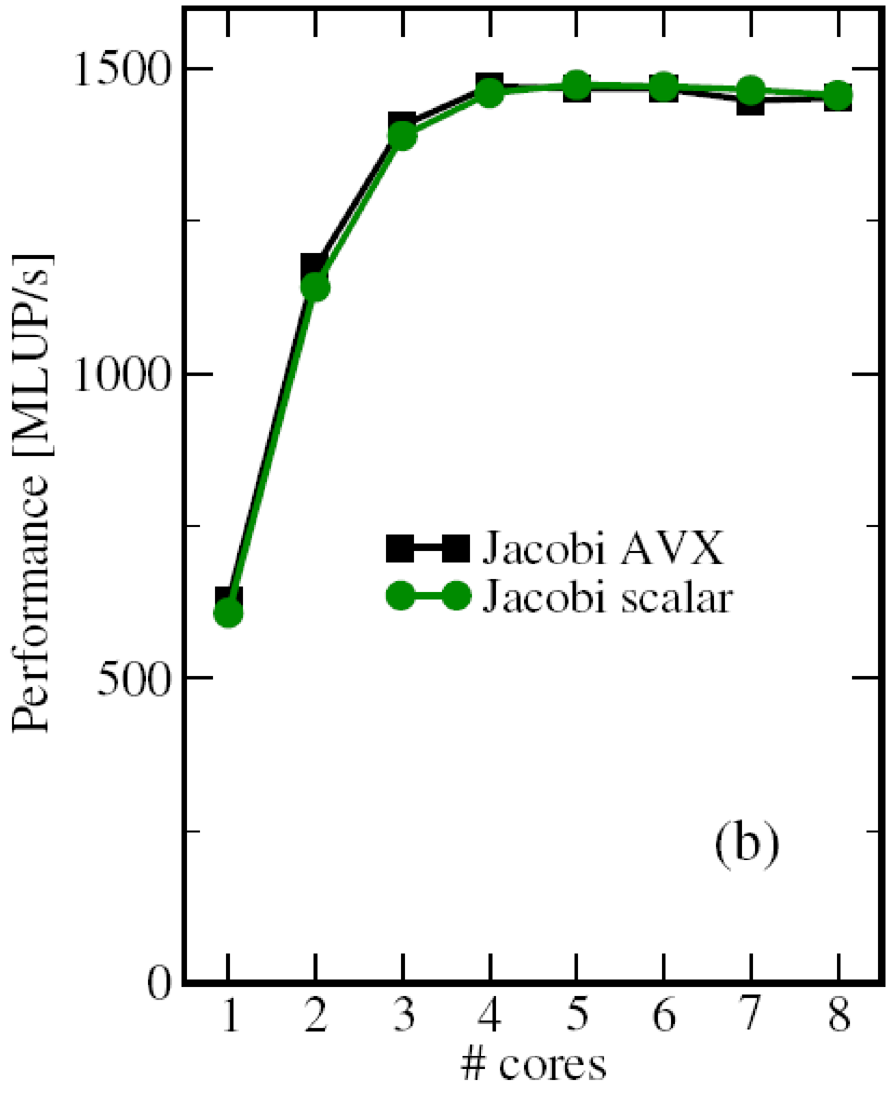
- Goal: Establish model for chip power and program energy consumption with respect to
  - Clock speed
  - Number of cores used
  - Single-thread program performance

- 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

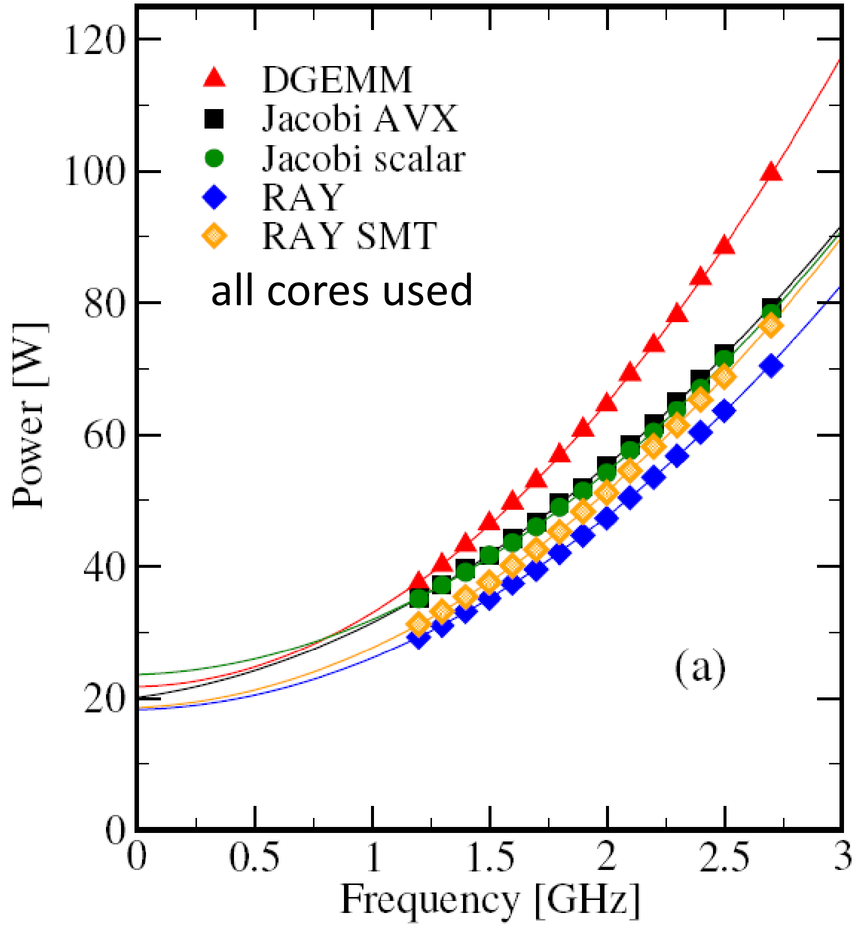
App scaling behavior (DGEMM omitted)

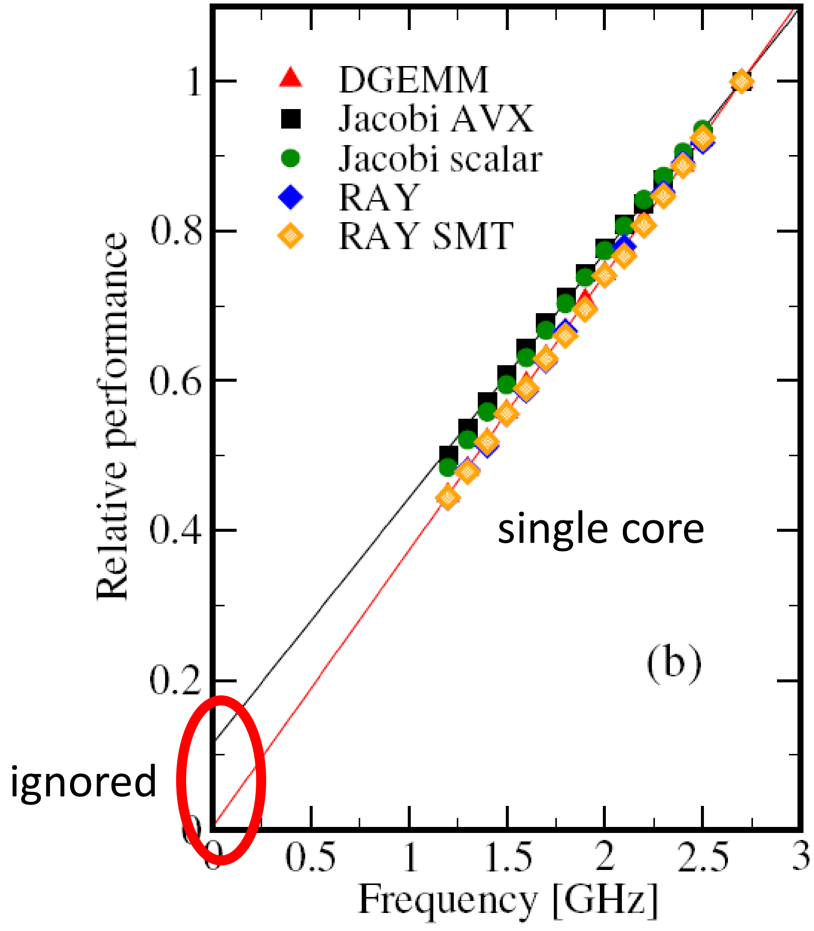




Chip power and performance vs. clock speed on full socket & single core

Sandy Bridge EP (8-core) processor:

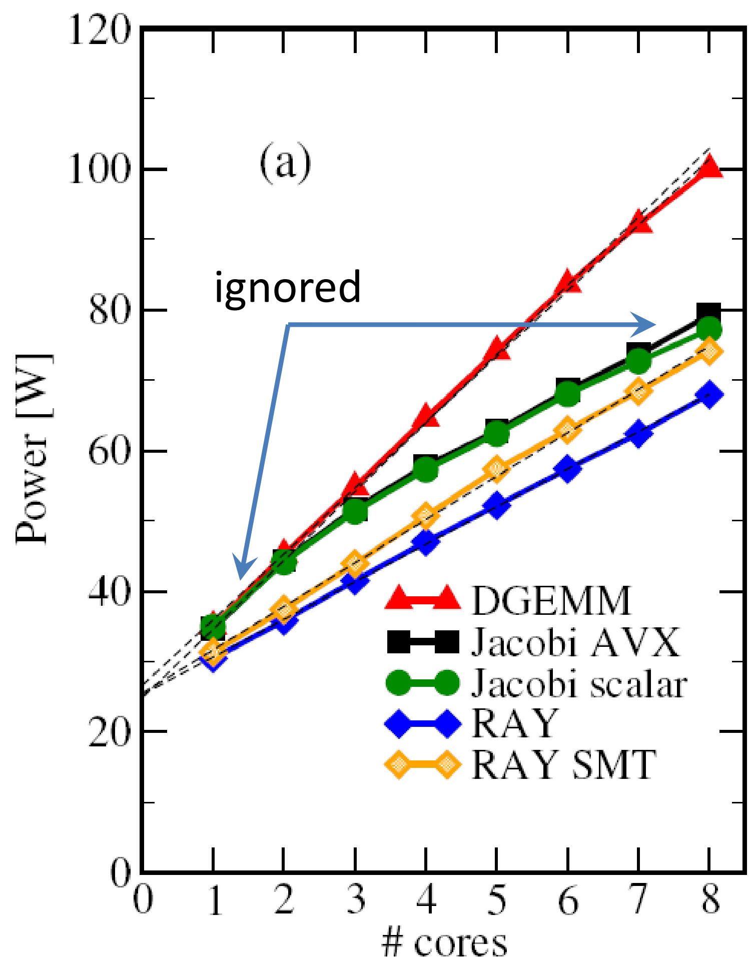


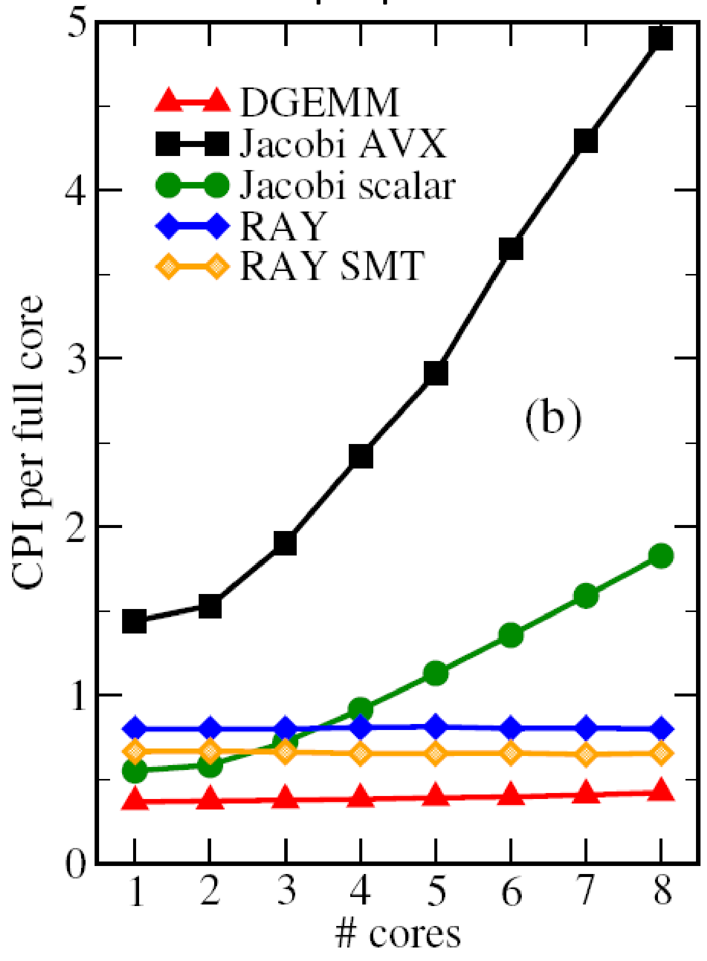


Chip power and cycles per instruction (CPI) vs. # of cores

Sandy Bridge EP (8-core) processor:

CPI and power correlated, but not proportional





A simple power model for multicore chips

Assumptions:

1. Power is a quadratic polynomial in the clock frequency
2. Dynamic power is linear in the number of active cores *t*
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
5. Energy to solution is power dissipation divided by performance

Model:

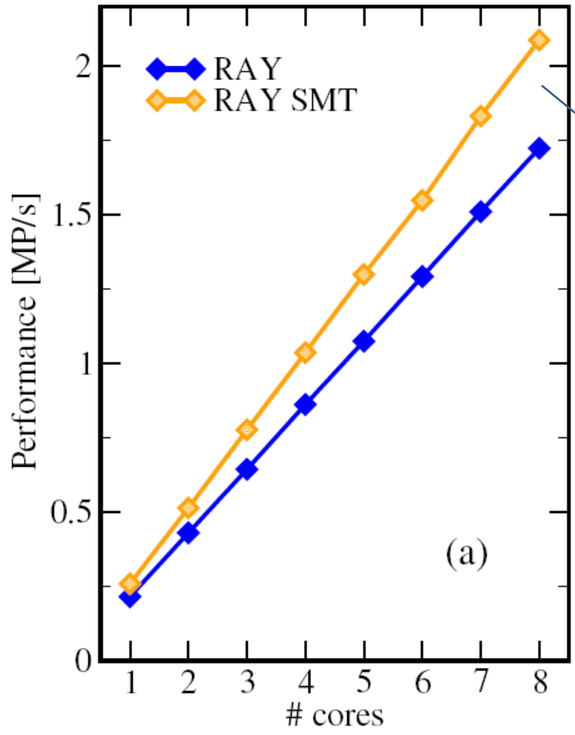
$$\boxed{E = \frac{W_0 + (W_1 f + W_2 f^2)t}{\min\left((1+\Delta\nu)tP_0, P_{\max}\right)}}$$

where $f = (1+\Delta\nu)f_0$

Model predictions

$$E = \frac{W_0 + (W_1 f + W_2 f^2)t}{\min\left((1+\Delta\nu)tP_0, P_{\max}\right)}$$

1. If there is no saturation, use all available cores to minimize *E*



Minimum E here

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

Model predictions

$$E = \frac{W_0 + (W_1 f + W_2 f^2)t}{\min\left((1+\Delta\nu)tP_0, P_{\max}\right)}$$

2. There is an optimal frequency f_{opt} at which E is minimal in the non-saturated case, with

$f_{\mathrm{opt}} = \sqrt{\frac{W_0}{W_2 t}}$, hence it depends on the baseline power

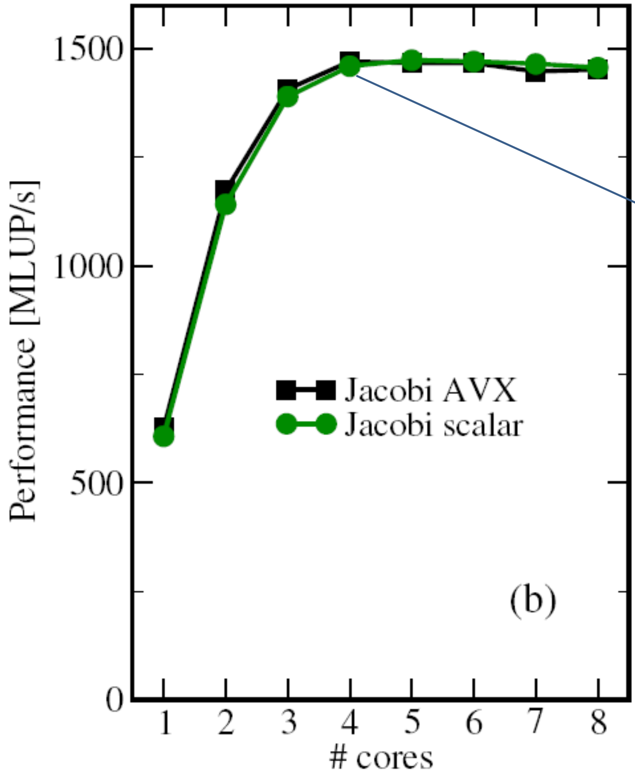
→ "Clock race to idle" if baseline power accommodates whole system!
→ If $f_{\mathrm{opt}} < f_0$, may have to look at other metrics, e.g., $\boldsymbol{C = E/P}$

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

Model predictions

$$E = \frac{W_0 + (W_1 f + W_2 f^2)t}{\min\left((1+\Delta\nu)tP_0, P_{\max}\right)}$$

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



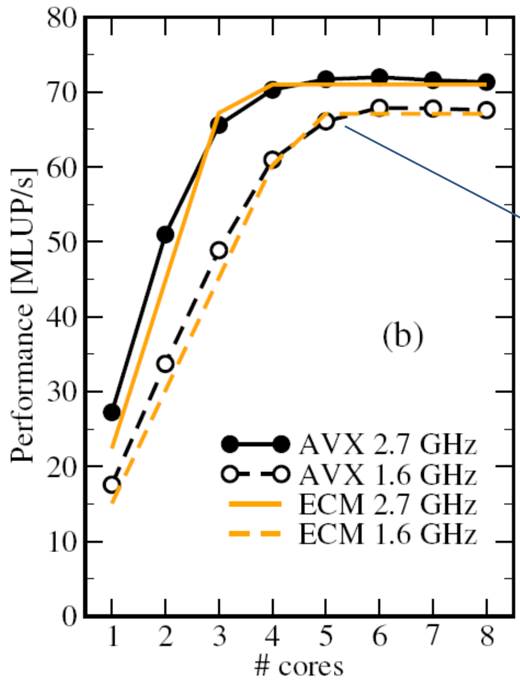
Minimum E here

$$t_s = \frac{P_{\max}}{(1+\Delta\nu)P_0}$$

Model predictions

$$E = \frac{W_0 + (W_1 f + W_2 f^2)t}{\min\left((1+\Delta\nu)tP_0, P_{\max}\right)}$$

4. If there is saturation, absolute minimum *E* is reached if the saturation point is at the number of available cores

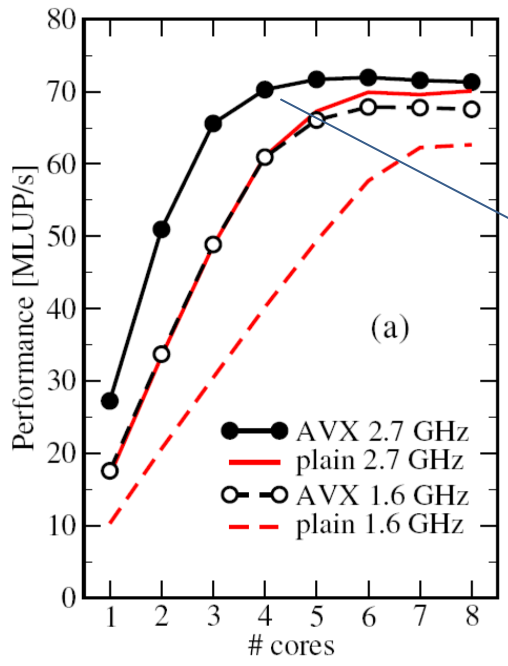


Slower clock
→ more cores to saturation
→ smaller E

Model predictions

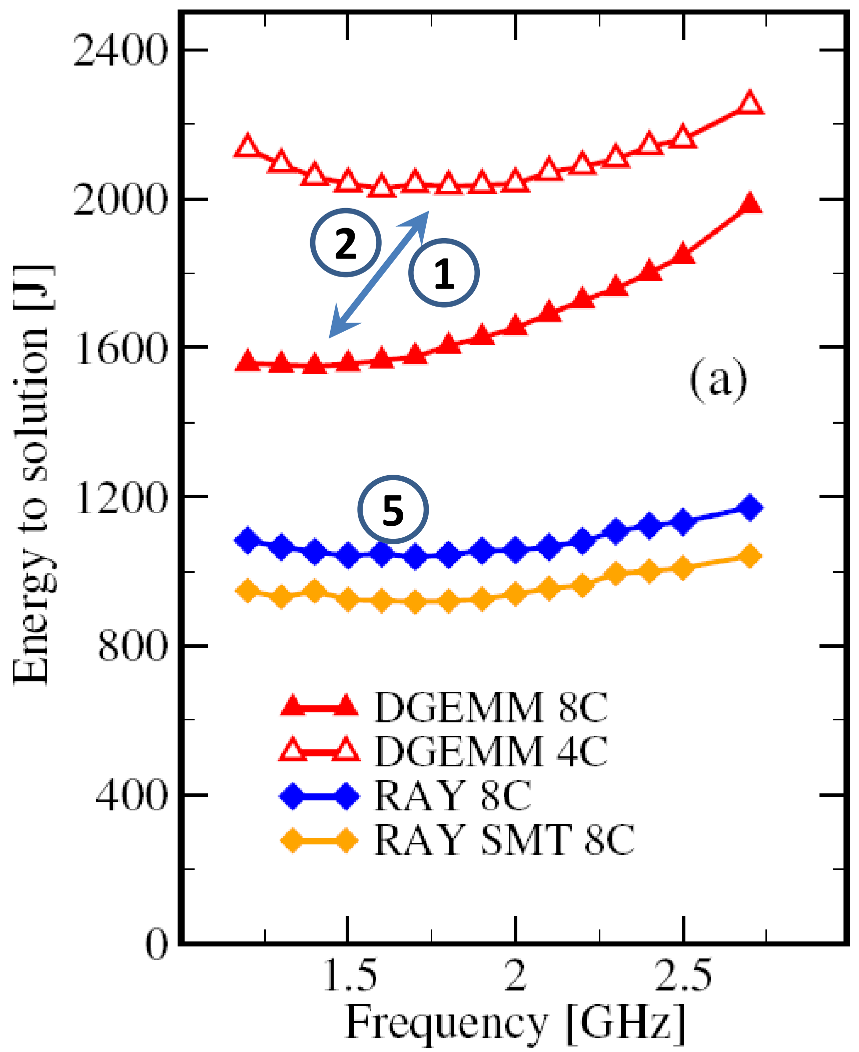
$$E = \frac{W_0 + (W_1 f + W_2 f^2)t}{\min\left((1+\Delta\nu)tP_0, P_{\max}\right)}$$

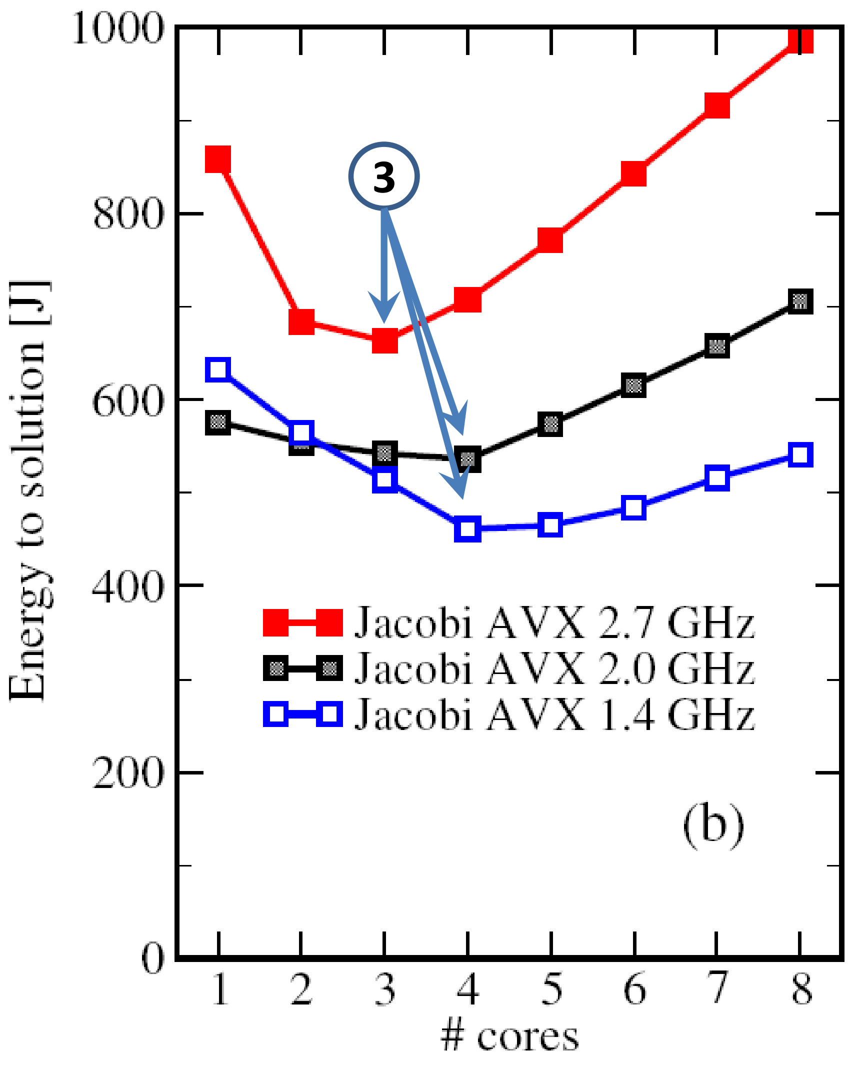
5. Making code execute faster on the core saves energy since
   - The time to solution is smaller if the code scales ("Code race to idle")
   - We can use fewer cores to reach saturation if there is a bottleneck



Better code
→ earlier saturation
→ smaller E @ saturation

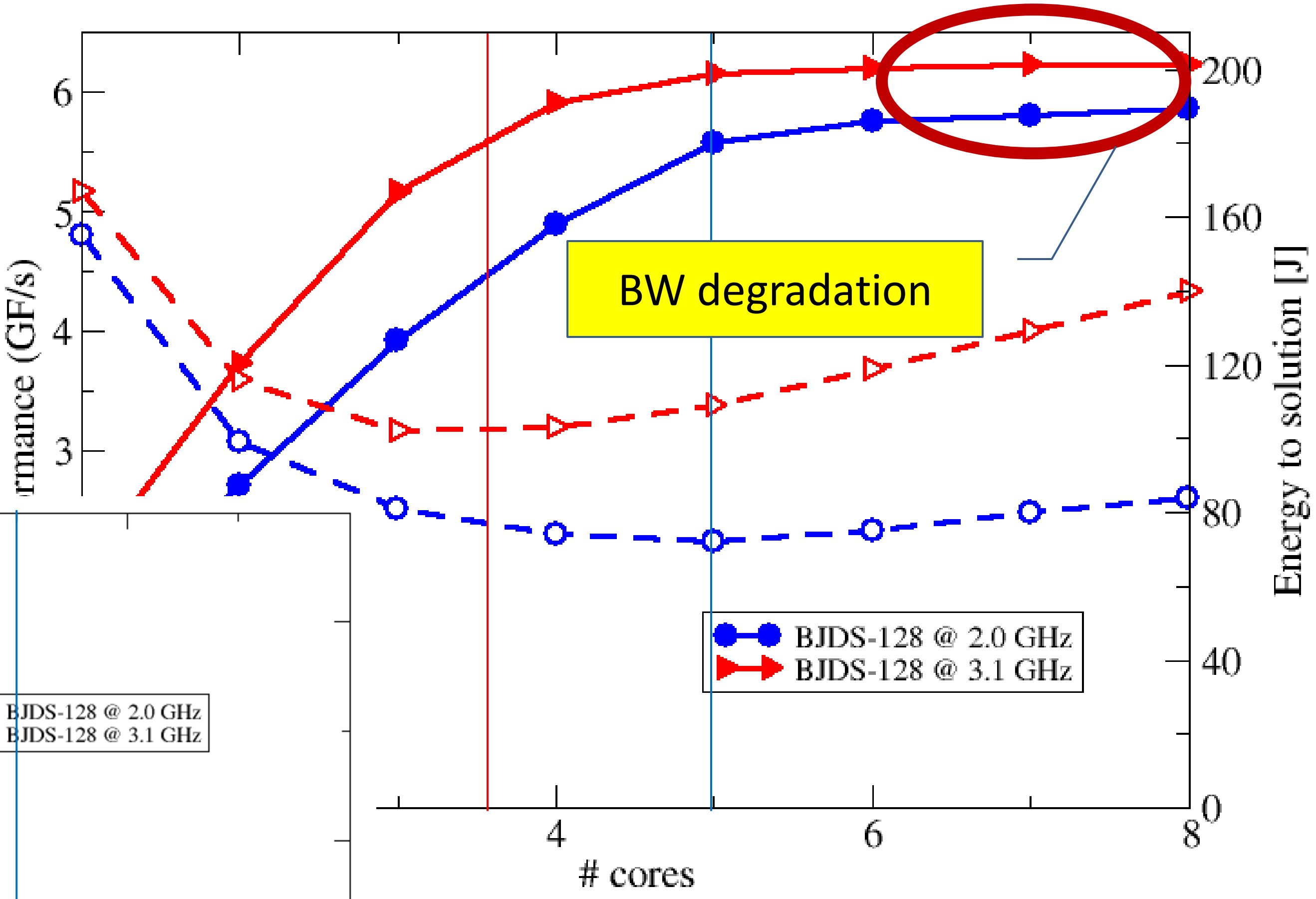
Model validation with the benchmark apps

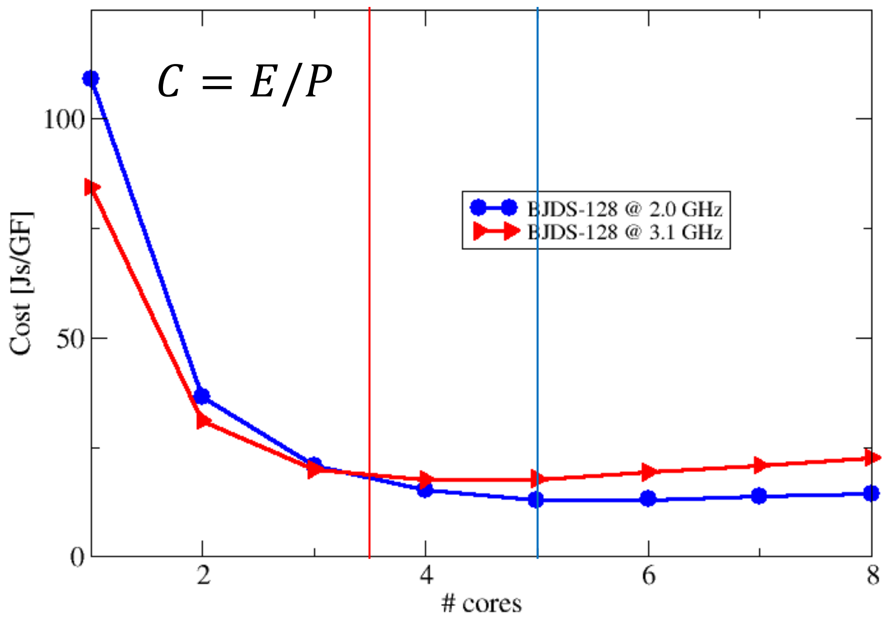




Example: spMVM on Sandy Bridge socket

Need for a cost model!?





→ Lower cost for slower clock speed (in this case)

Conclusions

- Performance Engineering == Performance Modeling with “bells and whistles”
- PE is a structured process which gives insight into the interaction of hardware and software
- Saturation effects are ubiquitous; understanding them gives us opportunity to
  - Find out about optimization opportunities
  - Put cores to good use
  - Save energy
- Possible extensions to the power model
  - Allow for per-core frequency setting (coming with Intel Haswell)
  - Accommodate load imbalance & sync overhead

Thank you.

Bundesministerium
für Bildung
und Forschung

hpcADD

KONWIHR II

OMI4papps

References

- J. Treibig and G. Hager: *Introducing a Performance Model for Bandwidth-Limited Loop Kernels.* Proceedings of the Workshop “Memory issues on Multi- and Manycore Platforms” at PPAM 2009, the 8th International Conference on Parallel Processing and Applied Mathematics, Wroclaw, Poland, September 13-16, 2009. DOI: 10.1007/978-3-642-14390-8_64

- G. Schubert, H. Fehske, G. Hager, and G. Wellein: *Hybrid-parallel sparse matrix-vector multiplication with explicit communication overlap on current multicore-based systems.* Parallel Processing Letters 21(3), 339-358 (2011). DOI: 10.1142/S0129626411000254

- G. Hager, J. Treibig, J. Habich, and G. Wellein: *Exploring performance and power properties of modern multicore chips via simple machine models*. Submitted. Preprint: arXiv:1208.2908