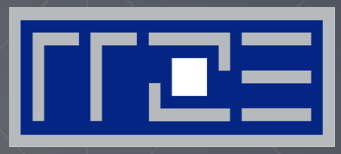
ERLANGEN REGIONAL COMPUTING CENTER



Holistic node-level performance engineering for maximum resource efficiency on modern multi-core CPUs

Georg Hager
Erlangen Regional Computing Center (RRZE)

ParisTech TELECOM
September 7, 2015

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. Lecture Notes in Computer Science Volume 6067, 2010, pp 615-624. DOI: 10.1007/978-3-642-14390-8_64 (2010).
- 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, DOI: 10.1002/cpe.3180 (2013).
- 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 Computat.: Pract. Exper. (2015), DOI: 10.1002/cpe.3489
- H. Stengel, J. Treibig, G. Hager, and G. Wellein: *Quantifying performance bottlenecks of stencil computations using the Execution-Cache-Memory model*. Proc. ICS’15, the 29th International Conference on Supercomputing, Newport Beach, CA, June 8-11, 2015. DOI: 10.1145/2751205.2751240

Further references

- M. Wittmann, G. Hager, J. Treibig and G. Wellein: *Leveraging shared caches for parallel temporal blocking of stencil codes on multicore processors and clusters.* Parallel Processing Letters **20** (4), 359-376 (2010). DOI: 10.1142/S0129626410000296

- J. Treibig, G. Hager, H. G. 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 **27**(2), 162-177 (2013). DOI: 10.1177/1094342012442424

- S. Kronawitter, H. Stengel, G. Hager, and C. Lengauer: *Domain-Specific Optimization of Two Jacobi Smoother Kernels and their Evaluation in the ECM Performance Model*. Parallel Processing Letters **24**, 1441004 (2014). DOI: 10.1142/S0129626414410047

- J. Hofmann, D. Fey, J. Eitzinger, G. Hager, G. Wellein: *Performance analysis of the Kahan-enhanced scalar product on current multicore processors.* Accepted for PPAM2015. Preprint: arXiv:1505.02586

Motivation

Analytical performance modeling:

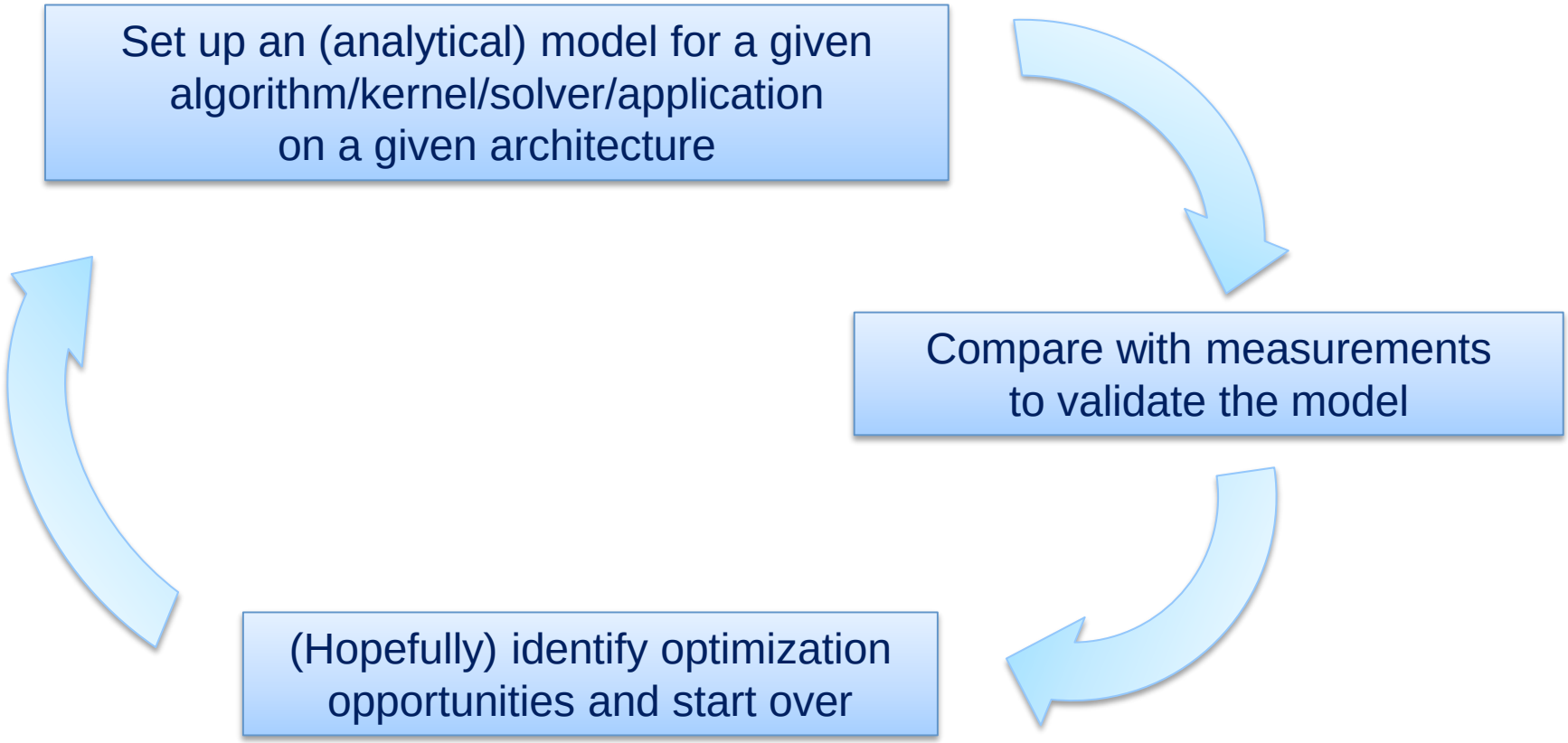
"Constructing a simplified model for the interaction between software and hardware in order to understand lowest-order performance behavior"

Basic questions addressed by analytic performance models

- What is the bottleneck? → optimization technique
- What is the next bottleneck? → performance potential of the optimization
- Impact of processor frequency and socket scalability
  → Appropriate execution parameters, energy-optimized operating point

If the model fails, we learn something!

Performance Engineering



J. Treibig, G. Hager, and G. Wellein: *Performance patterns and hardware metrics on modern multicore processors: Best practices for performance engineering*. Proc. 5th Workshop on Productivity and Performance (PROPER 2012) at Euro-Par 2012, August 28, 2012, Rhodes Island, Greece. Euro-Par 2012: Parallel Processing Workshops, Lecture Notes in Computer Science 7640, 451-460 (2013), Springer, ISBN 978-3-642-36948-3. DOI: 10.1007/978-3-642-36949-0_50.

The “classic” Roofline Model[1,2,3]

1. P_{max} = Applicable peak performance of a loop (this is not necessarily P_{peak})
2. I = Operational 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 (hardware feature)

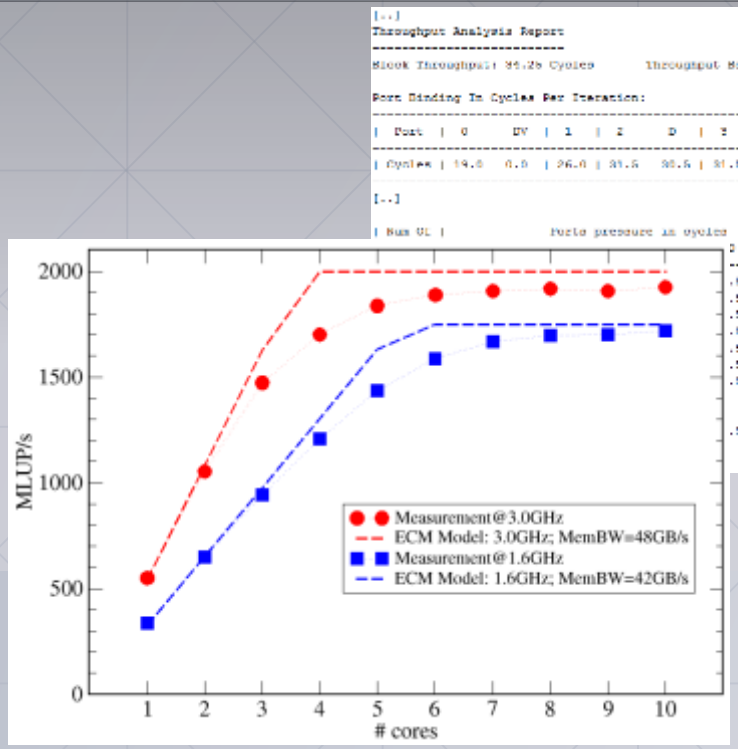
Expected performance: $\overbrace{P = \min(P_{max}, I \cdot b_S)}$ optmistic model (light speed)

[1] D. Callahan et al.: Estimating Interlock and Improving Balance for Pipelined Architectures. Journal of Parallel and Distributed Computing 5, 334-358 (1988)
[2] W. Schönauer: Scientific Supercomputing: Architecture and Use of Shared and Distributed Memory Parallel Computers. (2000)
[3] S. Williams et al.: Roofline: an insightful visual performance model for multicore architectures. Commun. ACM 52(4), 65-76 (2009)

Agenda

- ECM model
  - Basic rules, non-overlap
  - Notation
  - Saturation and comparison with Roofline
  - Case study 1: Kahan-enhanced scalar product
  - Case study 2: 3-D long-range stencil
- A simple power model for multicores
  - Case study: lattice-Boltzmann flow solver
- Summary

THE ECM MODEL

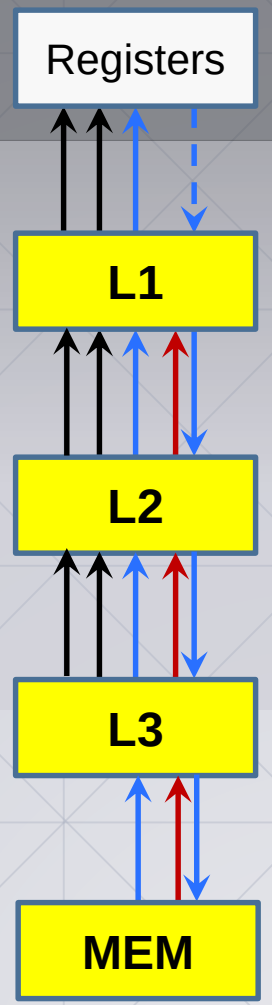



```
[..]
Throughput Analysis Report
--------------------------
Block Throughput: 36.2b Cycles       Throughput Bottleneck: FrontEnd

Port Binding In Cycles Per Iteration:
-------------------------------------------------------------------------
|  Port  |  0     DV  |  1   |  2     D  |  3     D  |  4   |  5   |
-------------------------------------------------------------------------
| Cycles | 14.0   0.0 | 26.0 | 30.5   30.5 | 31.5   30.5 | 2.0  | 24.0 |
-------------------------------------------------------------------------

[..]

| Num Of |              Ports pressure in cycles              |    |
|  Uops  |  0  |  4  |  5  |  6  |  7  |    |
---------------------------------------------------------------------
  .5   |     |     |     |     |     | vmovupd xmm1, xmmword ptr [r12+r14*4+0x10]
  .5   |     |     |     |     |     | vmovupd xmm0, xmmword ptr [rdi+r10*8+0x10]
  .5   |     |     |     |     |     | vmovupd xmm13, xmmword ptr [r12+r14*4+0x14]
  .5   |     |     |     |     |     | vmovupd xmm14, xmmword ptr [r12+r14*4+0xc]
  .5   |     |     |     |     |     | vmovupd xmm12, xmmword ptr [rdi+r11*8+0x10]
  .5   |     |     |     |     |     | mov r15, qword ptr [rsp+0x338]
  .5   |     |     | 1.0 |     |     | vinsertf128 ymm10, ymm1, xmmword ptr [r12+r
       |     |     |     |     |     | vaddpd ymm1, ymm7, ymm10
       |     |     |     |     |     | vmulpd ymm15, ymm8, ymm10
  .5   |     |     | 1.0 |     |     | vinsertf128 ymm11, ymm0, xmmword ptr [rdi+r
       |     |     |     |     |     | vsubpd ymm1, ymm1, ymm11
```



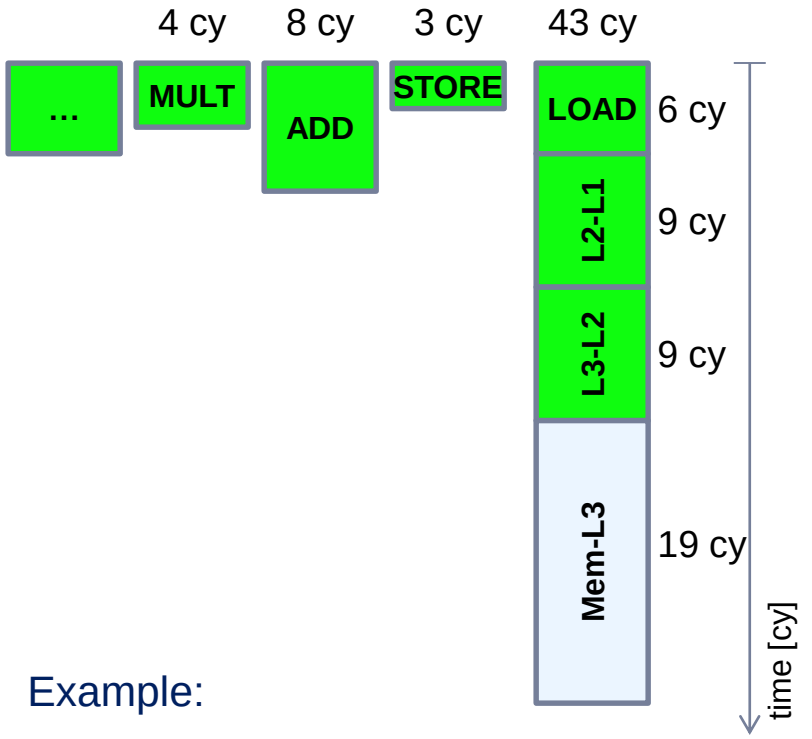
ECM model – predicting execution time for loop kernels

1. LOADs in the L1 cache do not overlap with any other data transfer in the memory hierarchy

2. Everything else in the core overlaps perfectly with data transfers (STOREs may show some non-overlap)

3. The scaling limit is set by the ratio of

$$\frac{\#\text{ cycles per CL overall}}{\#\text{ cycles per CL at the bottleneck}}$$



Example:

Single-core (data in L1): 8 cy (ADD)

Single-core (data in memory):
6+9+9+19 cy = 43 cy

Scaling limit: 43 / 19 = 2.3 cores

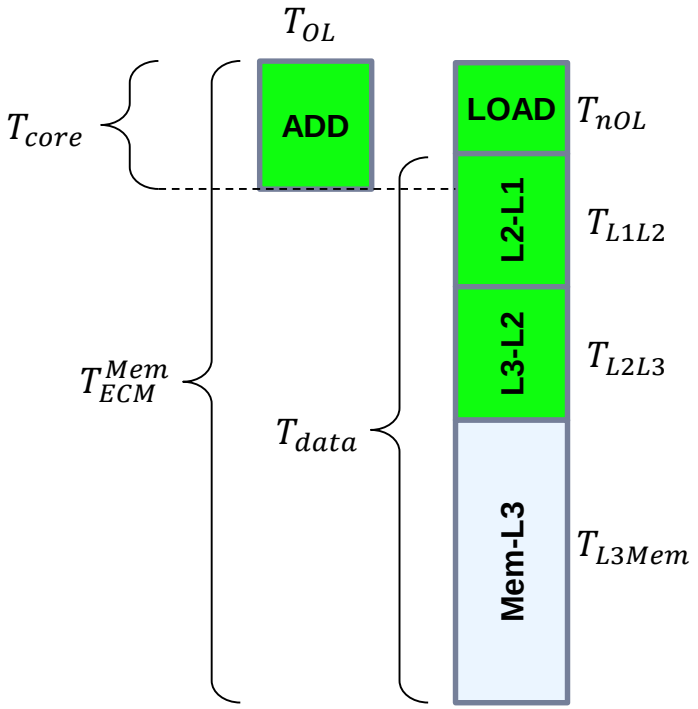
ECM model – composition

ECM predicted time

T_{ECM} = maximum of overlapping time and sum of all other contributions

$$T_{core} = \max(T_{nOL}, T_{OL})$$

$$\boxed{T_{ECM} = \max(T_{nOL} + T_{data}, T_{OL})}$$



Shorthand notation for time contributions:

$$\{\, T_{OL} \parallel T_{nOL} \mid T_{L1L2} \mid T_{L2L3} \mid T_{L3Mem} \,\}$$

cy invariant to clock speed

cy changes w/ clock speed

Example from previous slide:

$$\{\, 8 \parallel 6 \mid 9 \mid 9 \mid 19 \,\}\ \text{cy}$$

ECM model – prediction

Notation for cycle predictions in different memory hierarchy levels:

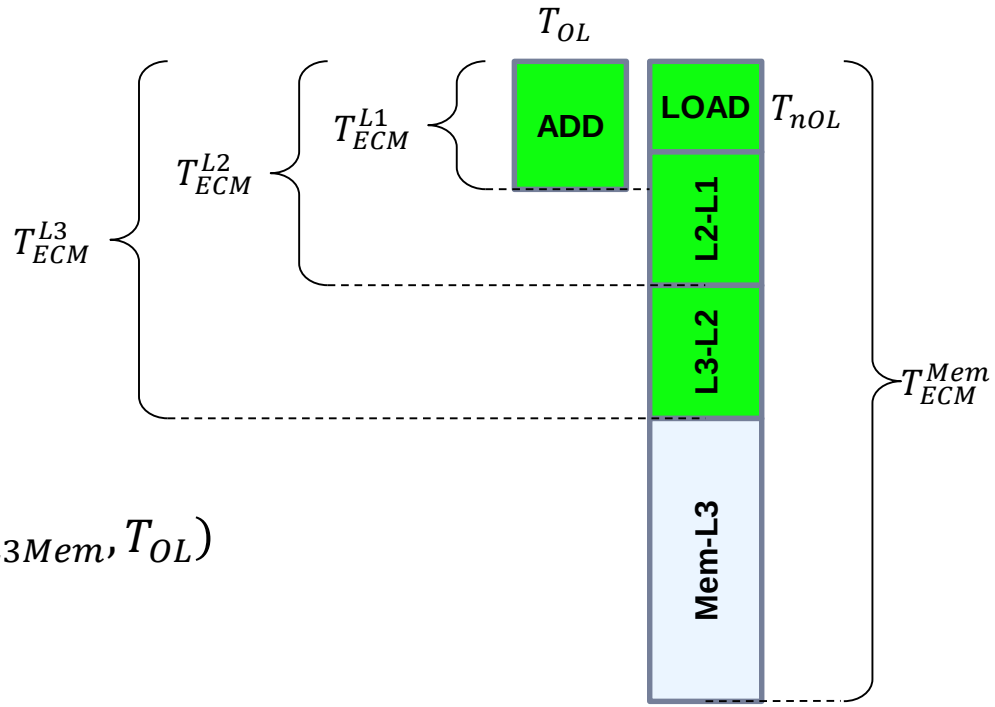
$$\{ T_{ECM}^{L1} \rceil T_{ECM}^{L2} \rceil T_{ECM}^{L3} \rceil T_{ECM}^{Mem} \}$$

$$T_{ECM}^{L1} = T_{core} = \max(T_{nOL}, T_{OL})$$

$$T_{ECM}^{L2} = \max(T_{nOL} + T_{L1L2}, T_{OL})$$

$$T_{ECM}^{L3} = \max(T_{nOL} + T_{L1L2} + T_{L2L3}, T_{OL})$$

$$T_{ECM}^{Mem} = \max(T_{nOL} + T_{L1L2} + T_{L2L3} + T_{L3Mem}, T_{OL})$$



Example: $\{ 8 \rceil 15 \rceil 24 \rceil 43 \}$ cy

Experimental data (measured) notation: $8.6 \rceil 16.2 \rceil 26 \rceil 47$ cy

ECM model – from time to performance with varying clock speed

f_0: base clock speed [cy/s]

f: actual cock speed [cy/s]

Performance is work (W) over time:

$$P_{ECM} = \frac{W \cdot f}{\{ \underbrace{T_{ECM}^{L1} \rceil T_{ECM}^{L2} \rceil T_{ECM}^{L3}}_{\text{in-cache performance is proportional to } f} \rceil \max(T_{ECM}^{L3} + T_{L3Mem} \cdot f/f_0 , T_{OL})\}}$$

ratio T_{ECM}^{L3}/T_{L3Mem} quantifies f-sensitivity of serial in-memory performance

Example:

$$P = \frac{32 \text{ flops} \cdot f}{\{ 8 \rceil 15 \rceil 24 \rceil 24 + 19 \cdot f/f_0 \} \text{ cy}}$$

ECM model – saturation

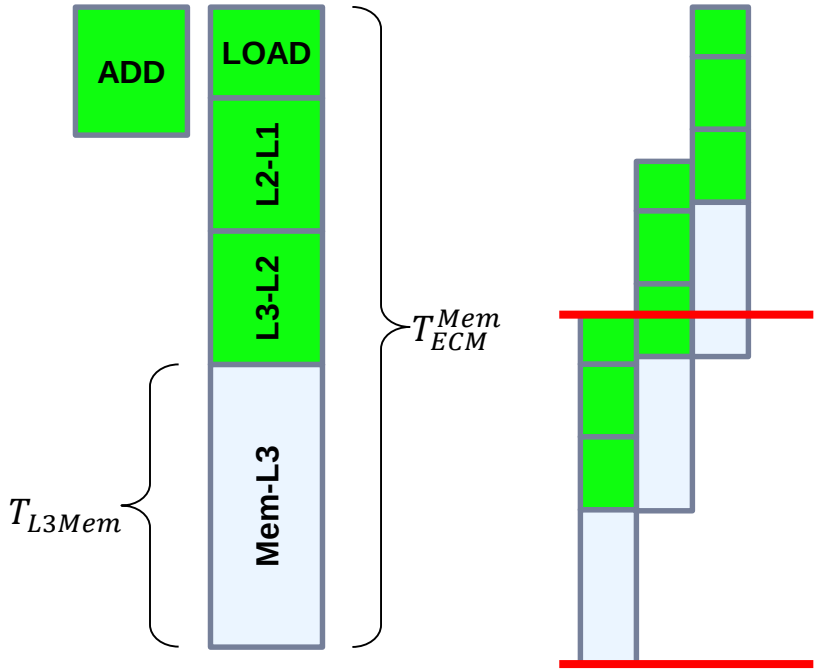
Main assumption: Performance scaling is linear until a bandwidth bottleneck (b_S) is hit

Performance vs. cores (Memory BN):

$$P_{ECM}(n) = \min\left(nP_{ECM}^{Mem}, \frac{b_S^{Mem}}{B_C^{Mem}}\right)$$

Number of cores at saturation:

$$n_S = \left\lceil \frac{b_S/B_C}{P_{ECM}^{Mem}} \right\rceil = \left\lceil \frac{T_{ECM}^{Mem}}{T_{L3Mem}} \right\rceil$$



Example:

$$\{ 8 \parallel 6 \mid 9 \mid 9 \mid 19 \}\ \text{cy}, \qquad \{ 8 \rceil 15 \rceil 24 \rceil 43 \}\ \text{cy} \implies n_S = \left\lceil \frac{43}{19} \right\rceil = 3$$

ECM vs. Roofline

Roofline assumes **full overlap** of all execution and transfer times

Roofline requires **measured baseline bandwidth limits** for all memory levels i (L2…Memory) at all core counts n: $b_S^i(n)$

$$\left.\begin{aligned} T_{Roof}^{L1} &= T_{core} = \max(T_{nOL}, T_{OL}) \\ &\vdots \\ T_{Roof}^{Mem} &= \max(T_{nOL}, T_{L1L2}, T_{L2L3}, T_{L3Mem}, T_{OL}) \end{aligned}\right\} P_{Roof}(n) = \min_i \left(nP_{ECM}^{L1}, \frac{b_S^i(n)}{B_C^i} \right)$$

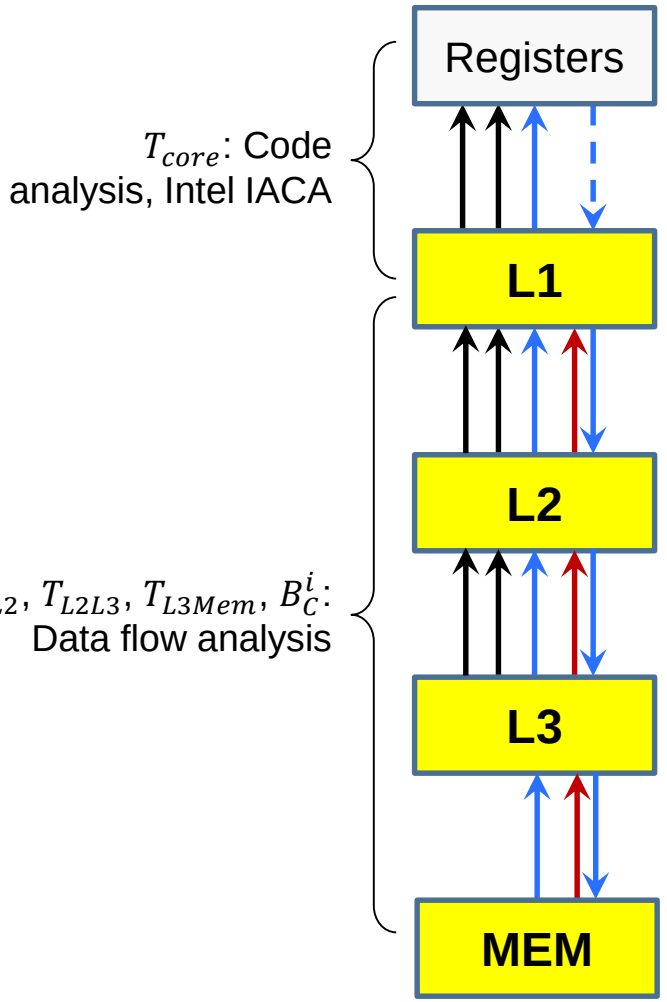
Roofline ≈ ECM if:

- $T_{core} \gg T_{data}$: non-overlapping data transfers are insignificant
  or
- Loop kernel is similar to streaming benchmark used to obtain $b_S^i(n)$

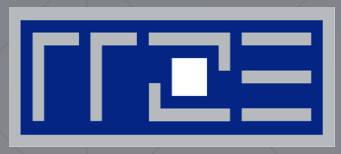
How do we get the numbers?

Basic information about hardware capabilities:

- In-core limitations
  - Throughput limits:µops, LD/ST, ADD/MULT per cycle
  - Pipeline depths
- Cache hierarchy
  - **ECM**: Cycles per CL transfer
  - **RL**: measured max bandwidths for all cache levels, core counts
- Memory interface
  - **ECM**: measured saturated BW
  - **RL**: measured max bandwidths for all core counts



CASE STUDY: KAHAN-ENHANCED SCALAR PRODUCT



Is the Kahan scalar product harmful for performance?

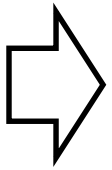
J. Hofmann, D. Fey, J. Eitzinger, G. Hager, G. Wellein: *Performance analysis of the Kahan-enhanced scalar product on current multicore processors.* Accepted for PPAM2015. Preprint: arXiv:1505.02586

Kahan-enhanced scalar product

```
float sum = 0.0;

for (int i=0; i<n; i++) {
   sum = sum + a[i] * b[i]
}
```

1 ADD, 1 MULT



```
float sum = 0.0;
float c = 0.0;
for (int i=0; i<N; ++i) {
    float prod = a[i]*b[i];
    float y = prod-c;
    float t = sum+y;
    c = (t-sum)-y;
    sum = t;
}
```

4 ADD, 1 MULT

- Does it harm performance to augment the dot kernel in this way?
- Is there are difference between single-threaded and multi-threaded?
- SP vs. DP? Influence of architecture?

ECM modeling of sdot on 10-core Ivy Bridge EP 2.2 GHz

Naive sdot (AVX):

$$\{2 \,\|\, 4 \,|\, 4 \,|\, 4 \,|\, 6.1 + 2.9\} \text{ cy}$$

Latency penalty

$$\{4 \rceil 8 \rceil 12 \rceil 18.1 + 2.9\} \text{ cy}$$

saturation at 4 cores

Kahan sdot, scalar mode:

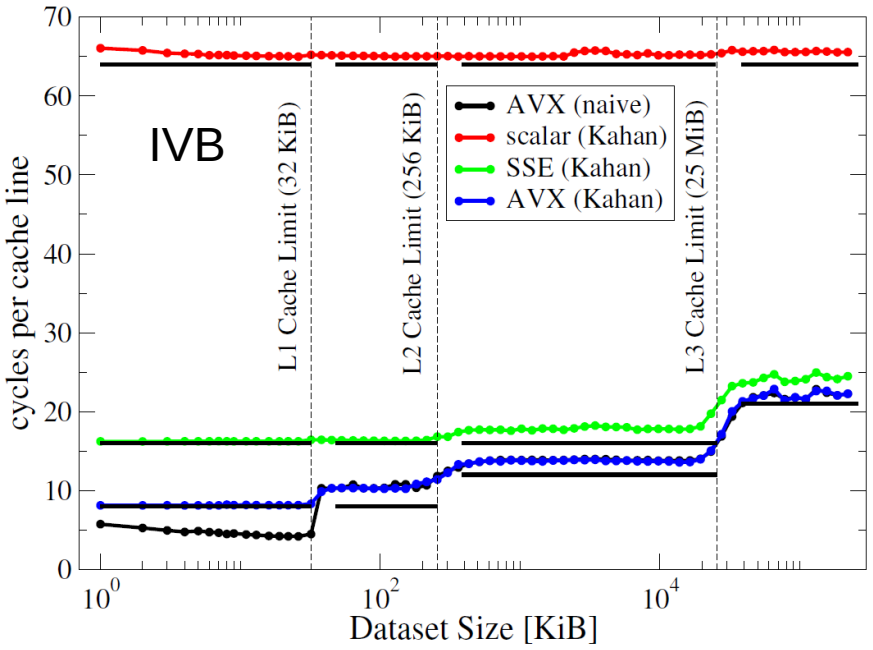
$$\{64 \,\|\, 16 \,|\, 4 \,|\, 4 \,|\, 6.1 + 2.9\} \text{ cy}$$
$$\{64 \rceil 64 \rceil 64 \rceil 64\} \text{ cy}$$

saturation at 11 cores

Comparing optimal AVX implementations on four Intel architectures (SP)

	ECM model [cy]	Prediction [cy/CL]	Pred. performance [GUP/s]
SNB	$\{8 \parallel 4 \mid 4 \mid 4 \mid 7.9+5.1\}$	$\{8 \rceil 8 \rceil 12 \rceil 19.9+5.1\}$	$\{5.40 \rceil 5.40 \rceil 3.60 \rceil 1.73\}$
IVB	$\{8 \parallel 4 \mid 4 \mid 4 \mid 6.1+2.9\}$	$\{8 \rceil 8 \rceil 12 \rceil 18.1+2.9\}$	$\{4.40 \rceil 4.40 \rceil 2.93 \rceil 1.68\}$
HSW	$\{8 \parallel \mathbf{2} \mid \mathbf{2} \mid 4 \mid 4.9+4.5\}$	$\{8 \rceil 8 \rceil 8 \rceil 12.9+4.5\}$	$\{4.60 \rceil 4.60 \rceil 4.60 \rceil 2.11\}$
BDW	$\{8 \parallel 2 \mid 2 \mid 4 \mid 7+\mathbf{1}\}$	$\{8 \rceil 8 \rceil 8 \rceil 15+1\}$	$\{3.60 \rceil 3.60 \rceil 3.60 \rceil 1.8\}$

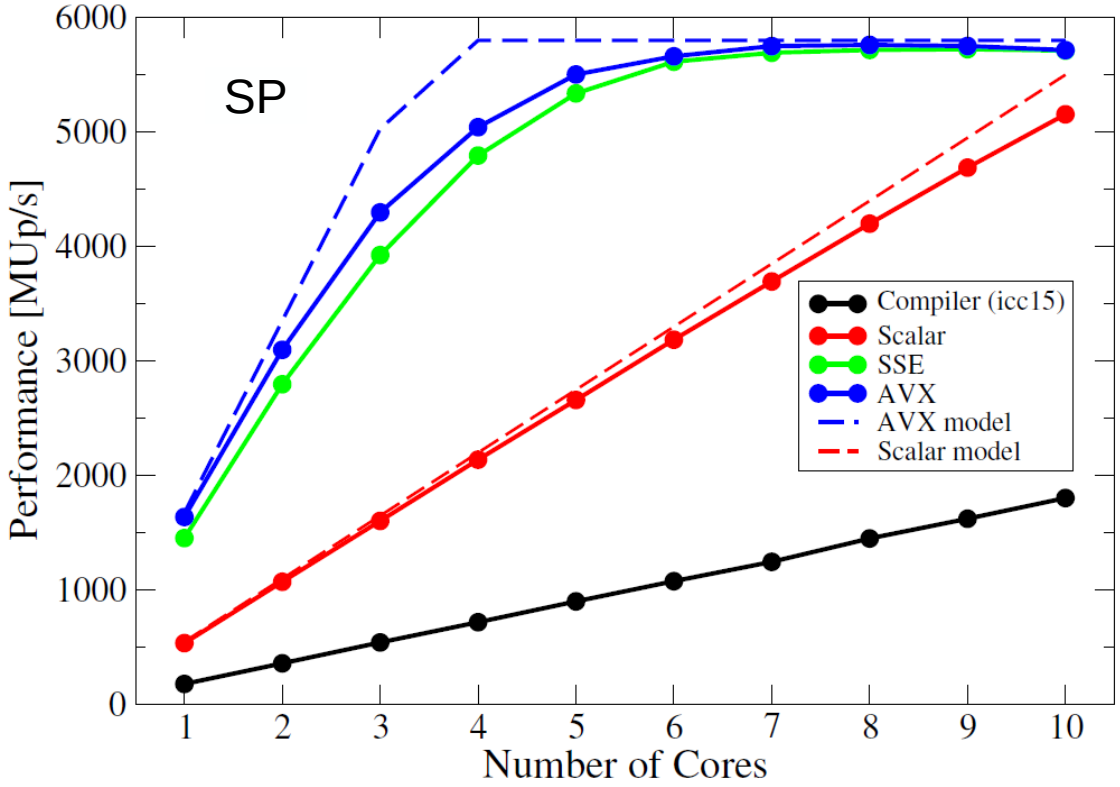
- Kahan without consequence if AVX is applied starting from L2 cache
- BDW latency panelty is rather low (== pure ECM model works well)



Saturation

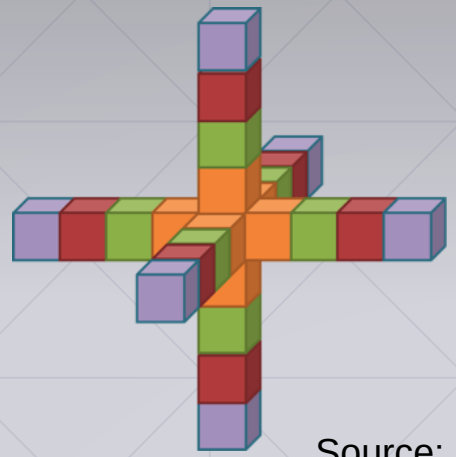
- SP: Saturation possible if any kind of SIMD is applied
- DP: Saturates always

- Compiler is not able to generate decent code



3D LONG-RANGE STENCIL (SINGLE PRECISION)

```
#pragma omp parallel for
for(int k=4; k < N-4; k++) {
 for(int j=4; j < N-4; j++) {
  for(int i=4; i < N-4; i++) {
   float lap = c0 * %V%[k][j][i]
  + c1 * ( V[ k ][ j ][i+1]+ V[ k ][ j ][i-1])
  + c1 * ( V[ k ][j+1][ i ]+ V[ k ][j-1][ i ])
  + c1 * ( V[k+1][ j ][ i ]+ V[k-1][ j ][ i ])
     ...
  + c4 * ( V[ k ][ j ][i+4]+ V[ k ][ j ][i-4])
  + c4 * ( V[ k ][j+4][ i ]+ V[ k ][j-4][ i ])
  + c4 * ( V[k+4][ j ][ i ]+ V[k-4][ j ][ i ]);
   U[k][j][i] = 2.f * V[k][j][i] - U[k][j][i]
              + ROC[k][j][i] * lap;
}}}
```



Source: http://goo.gl/dqOlnI

3D long-range SP stencil ECM model

Layer condition in L3 at problem size $N_i \times N_j \times N_k$:

$$9 \cdot N_i \cdot b_j \cdot n_{threads} \cdot 4\ \mathrm{B} < \frac{C_3}{2}$$

ECM Model: $\{ 68 \parallel 62 \mid 24 \mid 24 \mid 17 \}$ cy → $\{ 68 \rceil 86 \rceil 110 \rceil 127 \}$ cy

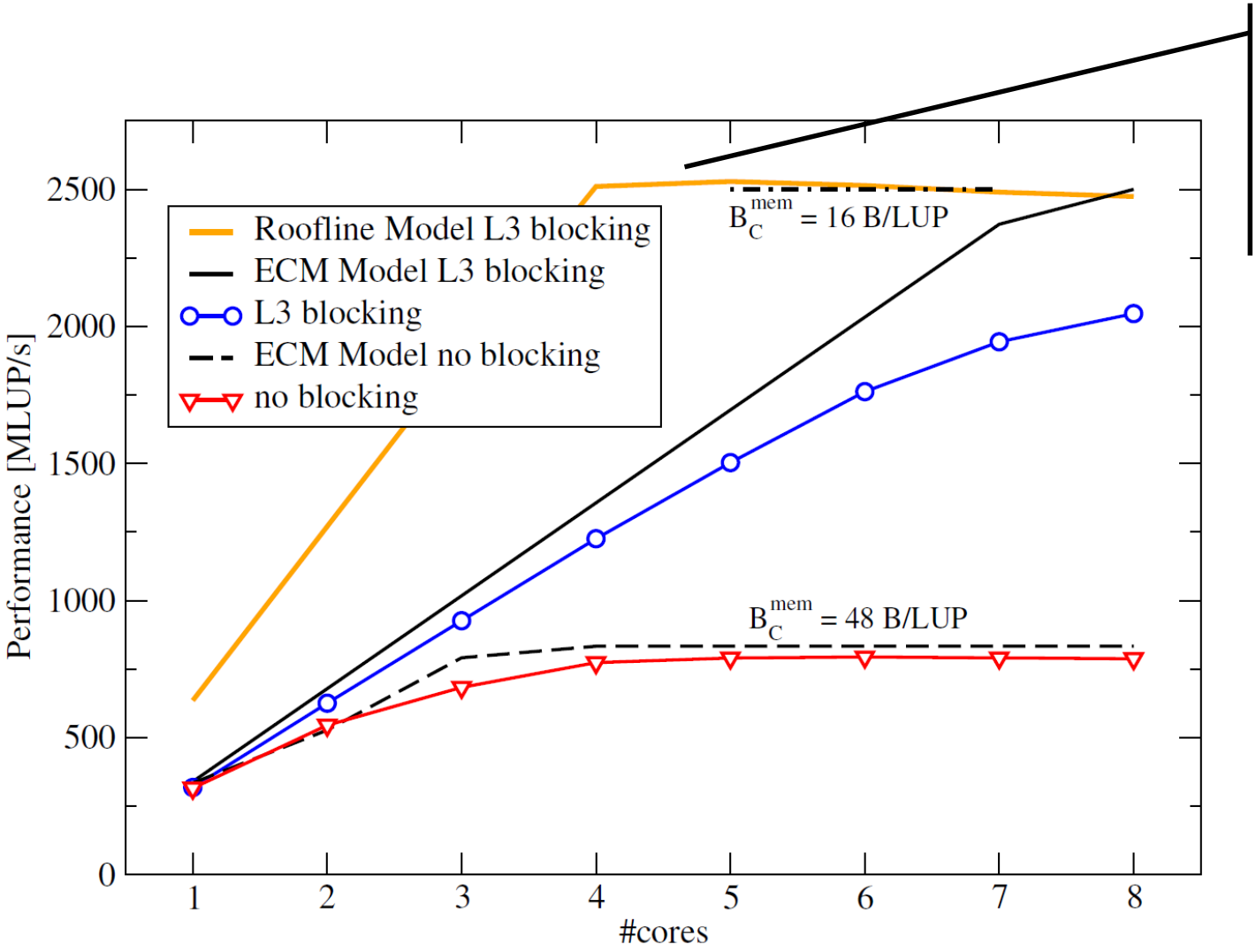
T_{L3Mem} plays minor part

Saturation at $n_s = \left\lceil \frac{127}{17} \right\rceil = 8$ cores.

Consequences:

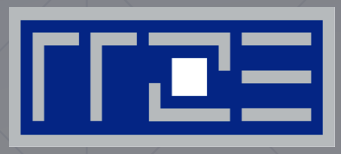
- Temporal blocking will not yield substantial speedup
- Improve low-level code first (semi-stencil…?)

3D long-range SP stencil results (SNB)



Roofline too optimistic due to overlapping assumption

MULTICORE PERFORMANCE ENGINEERING

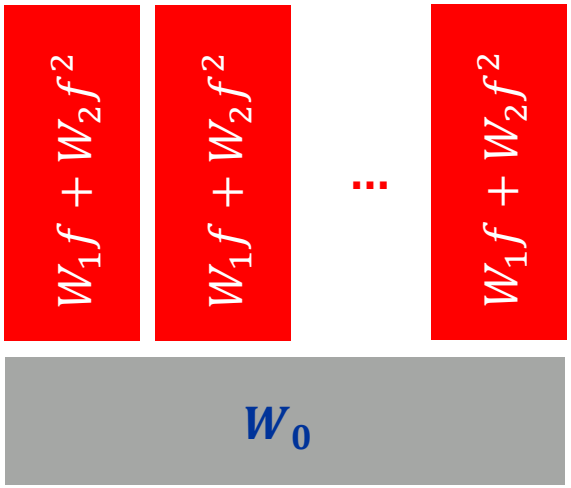


A simple power model for multicore CPUs

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



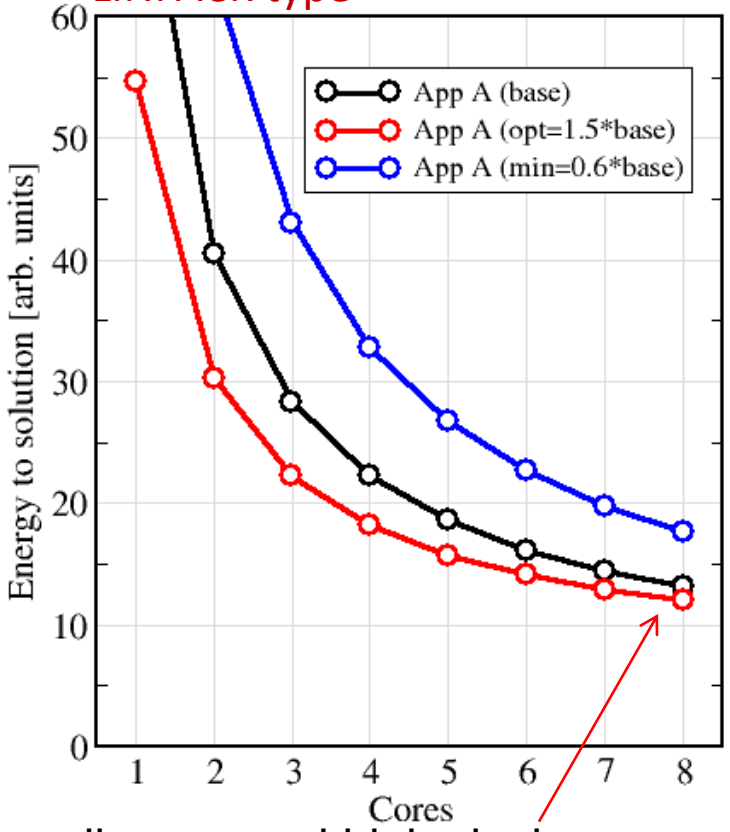
Model:

$$E = \frac{\text{Power}}{\text{Performance}} = \frac{W_0 + (W_1 f + W_2 f^2) n}{P(n, f)}$$

Two simple examples

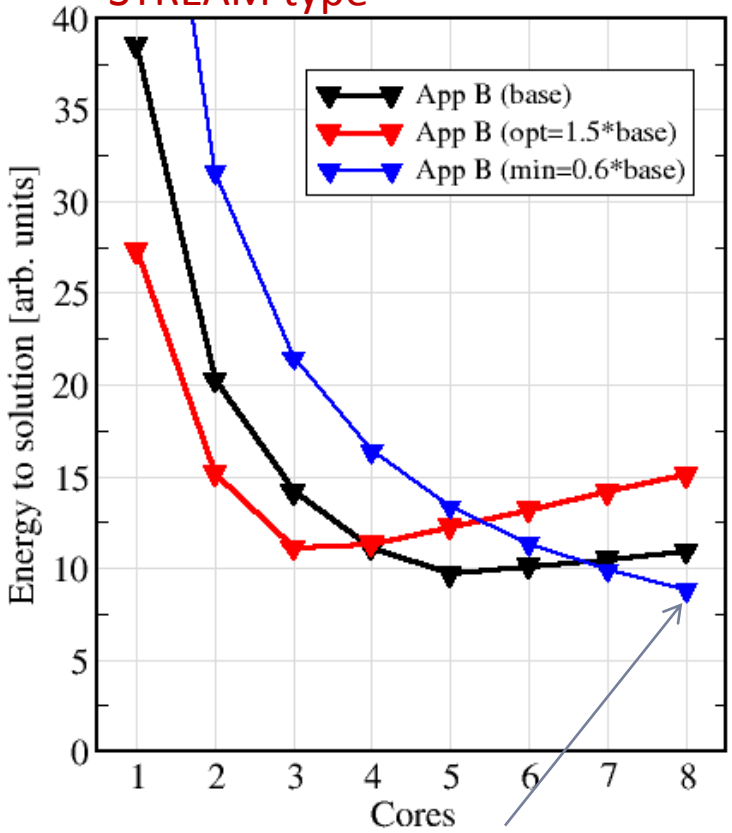
W_0 = 73 W base = 2 GHz
W_2 = 1 W / GHz^2 Turbo = 3 GHz

LINPACK type



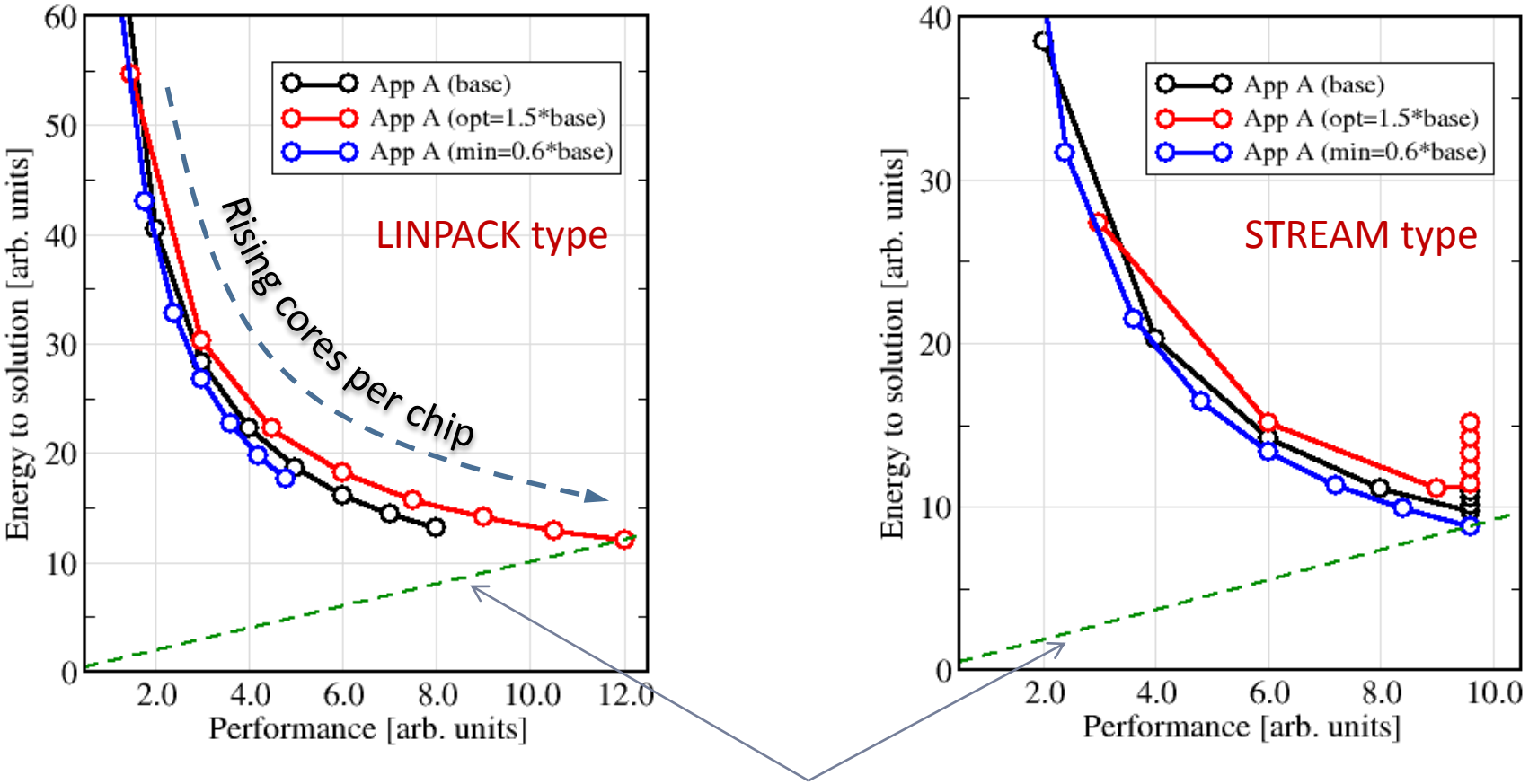
Use all cores and high clock speed!

STREAM type



Run all cores at clock speed that still saturates performance

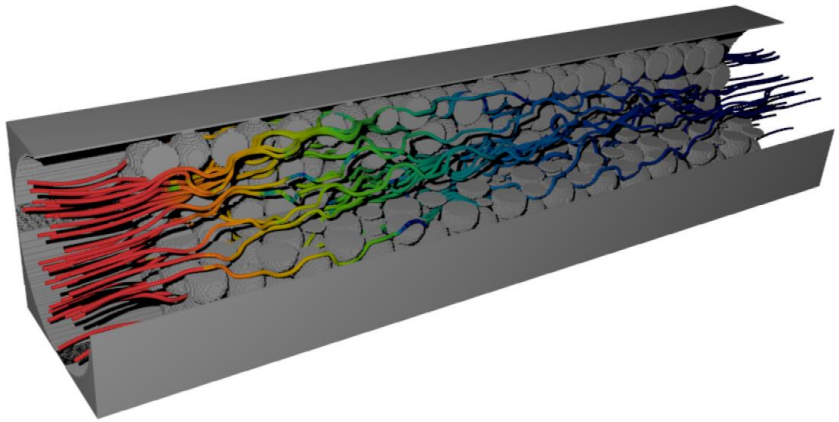
Energy vs. Performance ("Z-plot")

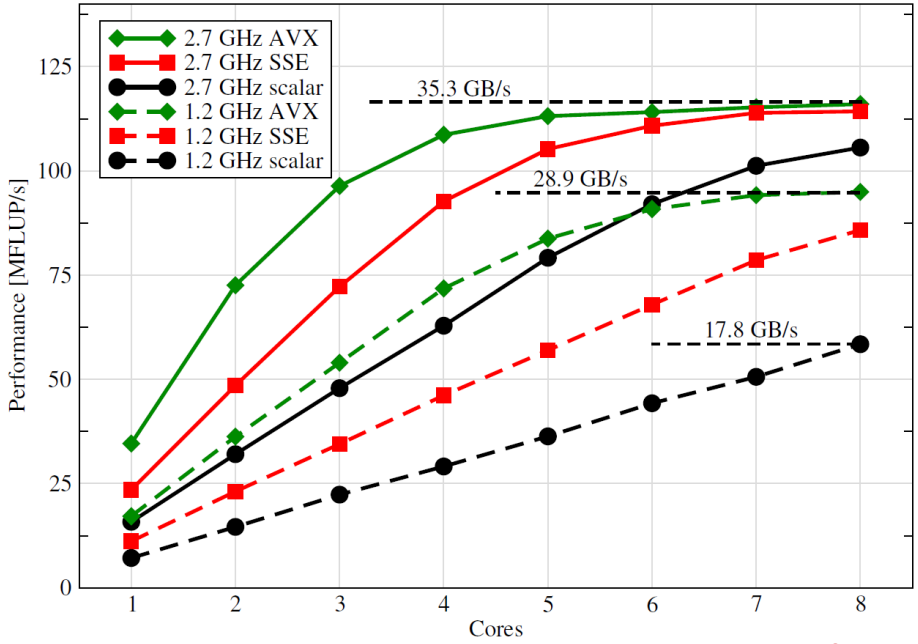


"Isoline" of constant energy delay product ($\boldsymbol{E} \times \Delta \boldsymbol{t}$)

Case study: Lattice Boltzmann

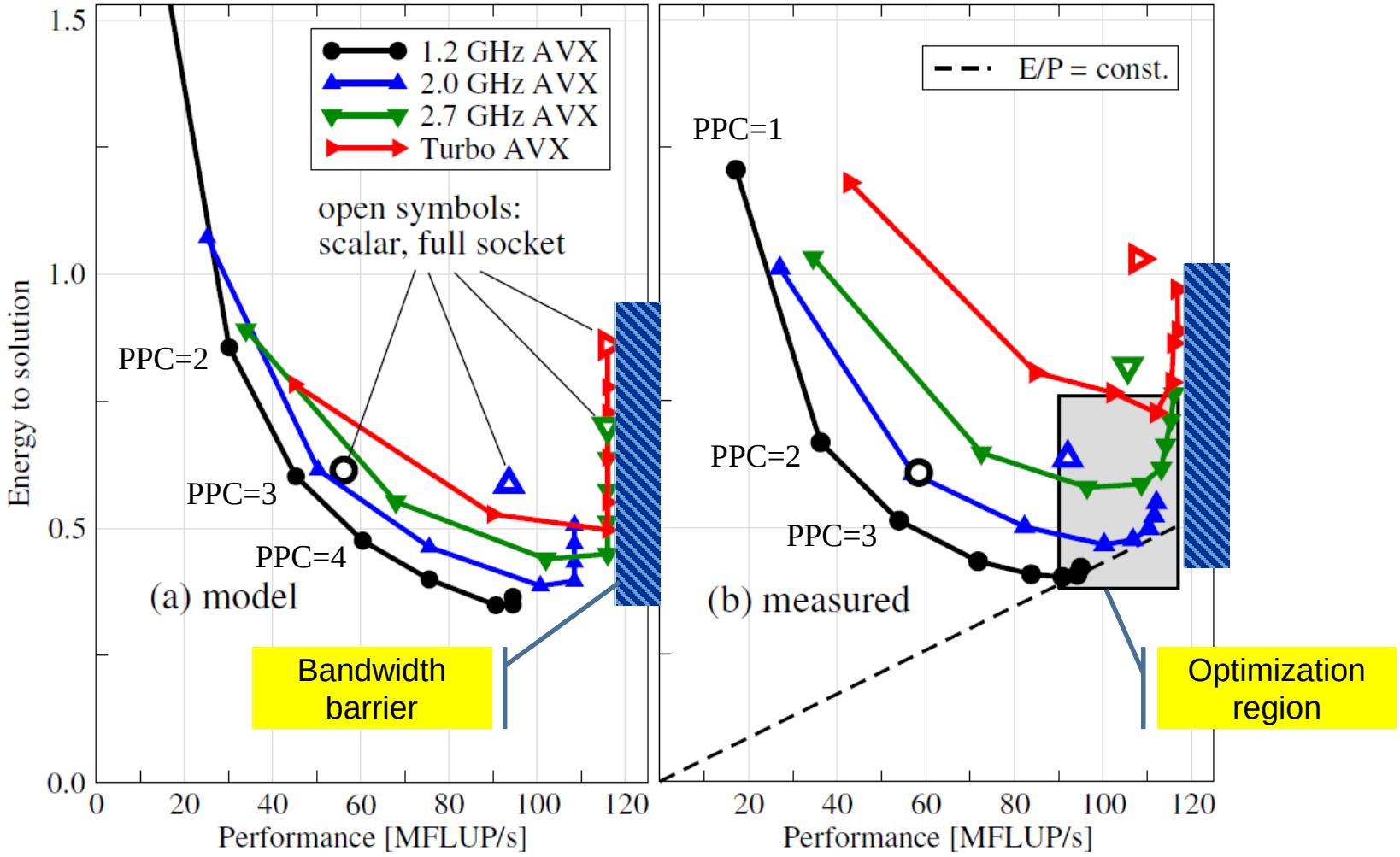
- Sparse representation lattice-Boltzmann flow solver
- Well suited for highly porous geometries, MPI parallel
- „AA pattern“ propagation → SIMD friendly, 304-376 bytes/LUP
- Saturating performance for vectorized code on modern Intel chips





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 Preprint: arXiv:1304.7664

Energy to solution vs. Performance on the socket (SNB) for a lattice-Boltzmann flow solver



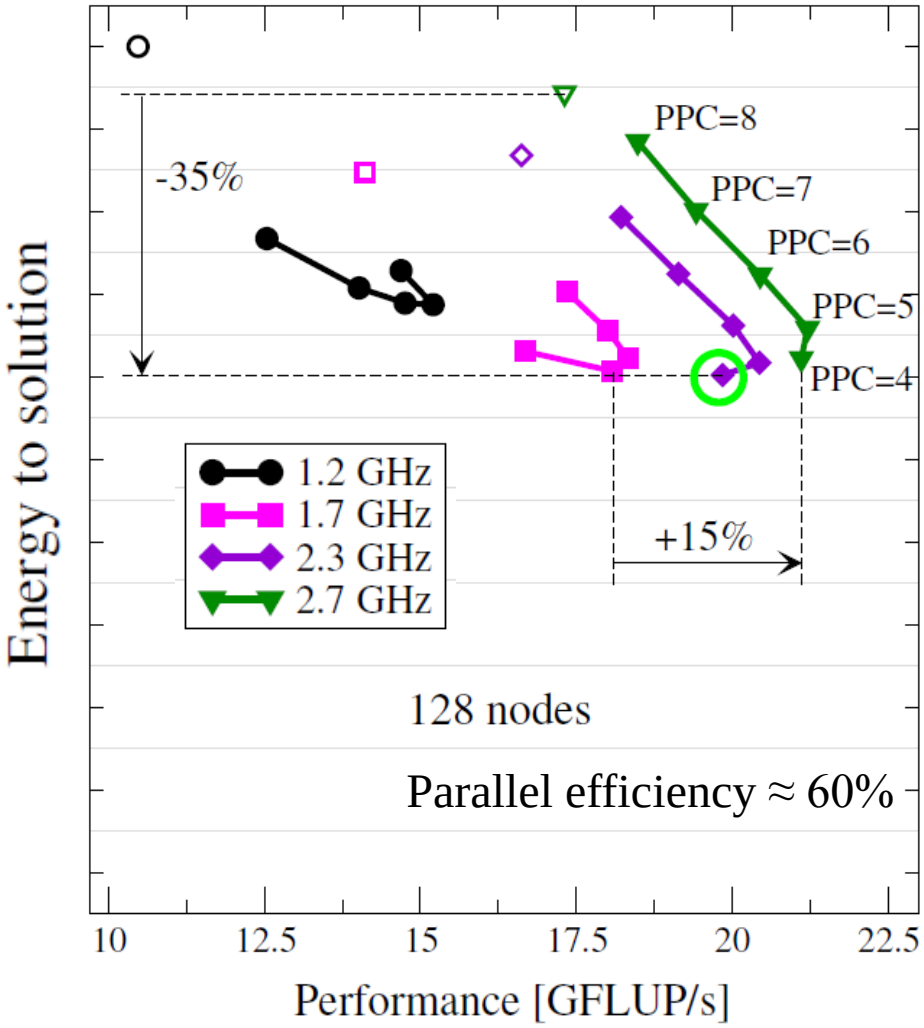
Single node → multi node

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!



Summary

- Analytical models are extremely helpful in understanding performance behavior and guiding optimizations
  - Roofline
  - ECM
- Simple power models can qualitatively describe the power consumption characteristics of code on the chip level
  - Saturating vs. scalable code
  - Z-plot
  - Energy-Delay-Product
  - Coupling with performance model

ERLANGEN REGIONAL COMPUTING CENTER

DFG Priority Programme1648

Bavarian Network for HPC

Thank You.

Holger Stengel
Johannes Hofmann
Julian Hammer
Jan Eitzinger
Gerhard Wellein