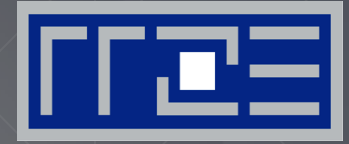


ERLANGEN REGIONAL COMPUTING CENTER



Performance Engineering for Stencil Updates on Modern Processors

Holger Stengel, Jan Treibig, Georg Hager, and Gerhard Wellein

Erlangen Regional Computing Center & Dept. of Computer Science
Friedrich-Alexander-University Erlangen-Nuremberg, Germany



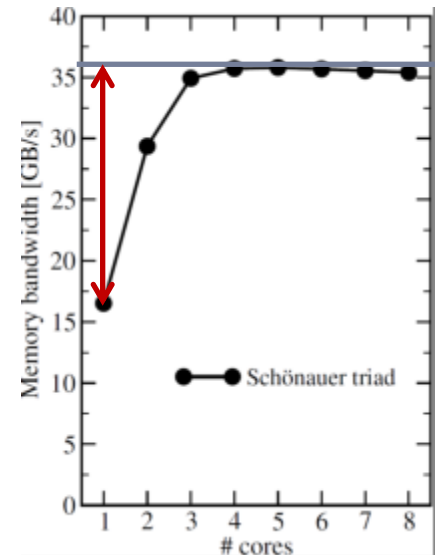
Partially funded by DFG Priority Programme 1648

Motivation

- There are so many
 - potential stencil structures (2D/3D, long-/short-range,...),
 - text book optimizations (register / spatial / temporal blocking,...),
 - parameters for optimization (blocking / unrolling factor,...),
 - parameters for execution (OpenMP schedule, clock speed, #cores).
- Basic questions addressed by ECM model
 - **What is the bottleneck** of my stencil implementation?
→ Choose appropriate **optimization** technique
 - **What is the next bottleneck** I will hit and “how far” is it from the first?
→ This yields the **performance potential** of the optimization
 - Impact of processor frequency and #cores used on performance

Agenda

- ECM model – basics
 - STREAM like kernel
 - Why does a single core not get full BW?
 - Case study 1: 5-pt 2D stencil
- Case study 2: Long range 3D stencil
- Case study 3: 3D stencil with DIVide
- Summary



Execution-Cache-Memory (ECM) model

Assumptions

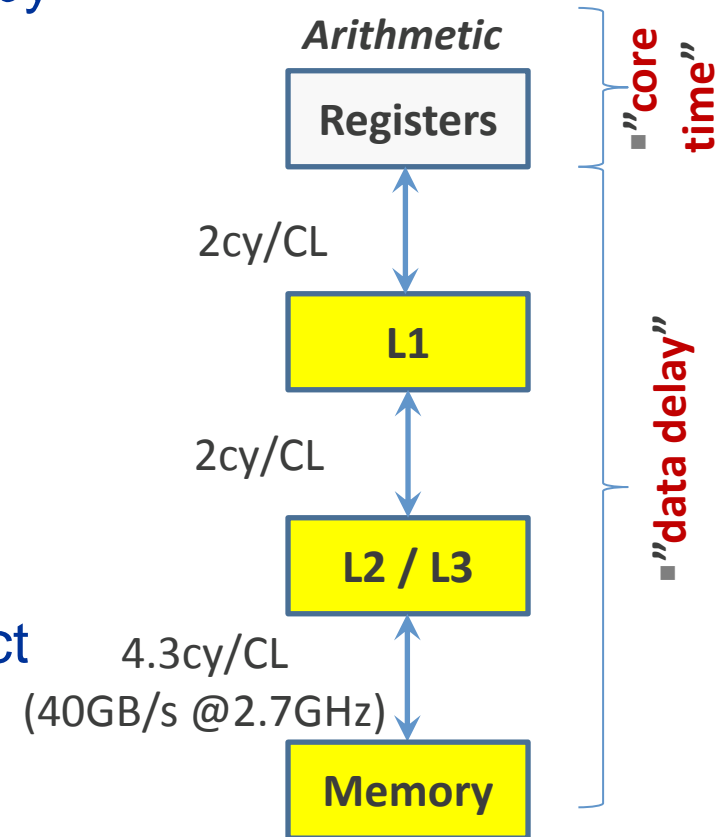
- **Single core** execution time is determined by
 - **In-core** execution **time** &
 - **Data delay** in memory hierarchy
- **Socket scaling** in socket is linear until relevant shared bottleneck is hit

Insights

- Single-core performance & socket scaling
- Relevant bottlenecks & performance impact

Input

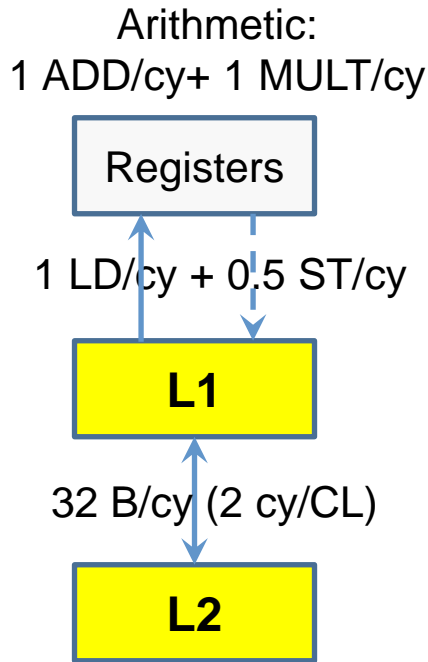
- Similar as for roofline (e.g. STREAM)
- Data transfer times of all memory levels



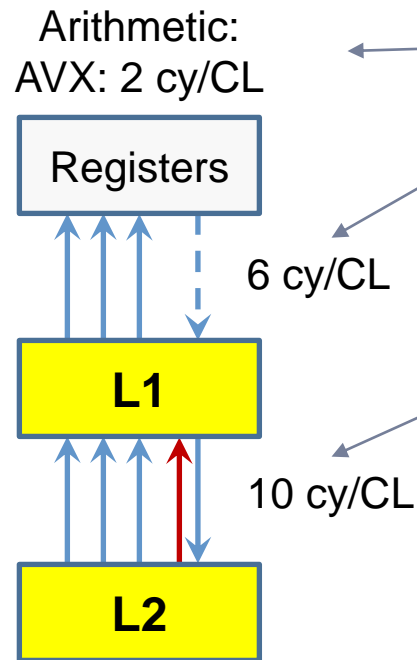
Example: Schönauer Vector Triad in L2 cache

REPEAT[$\mathbf{A}(:,) = \mathbf{B}(:,) + \mathbf{C}(:,) * \mathbf{D}(:,)$] @ double precision
 Analysis: Sandy Bridge core w/ AVX (unit of work: 1 Cache Line)

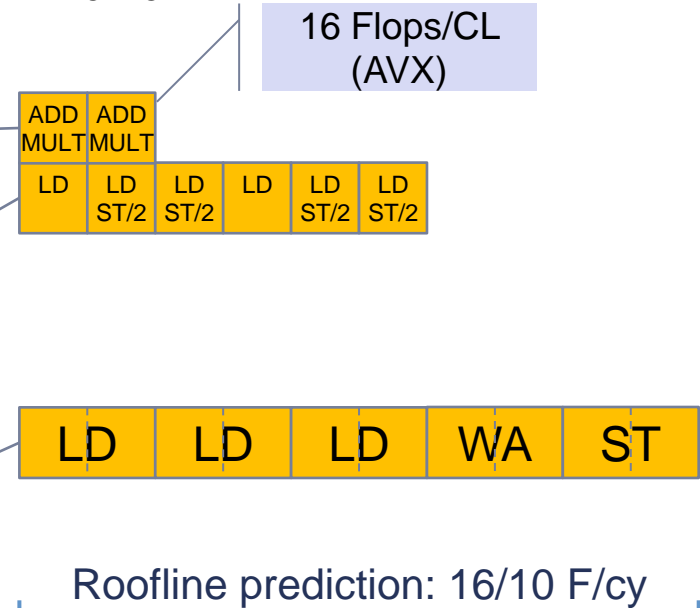
Machine characteristics:



Triad analysis (per CL):



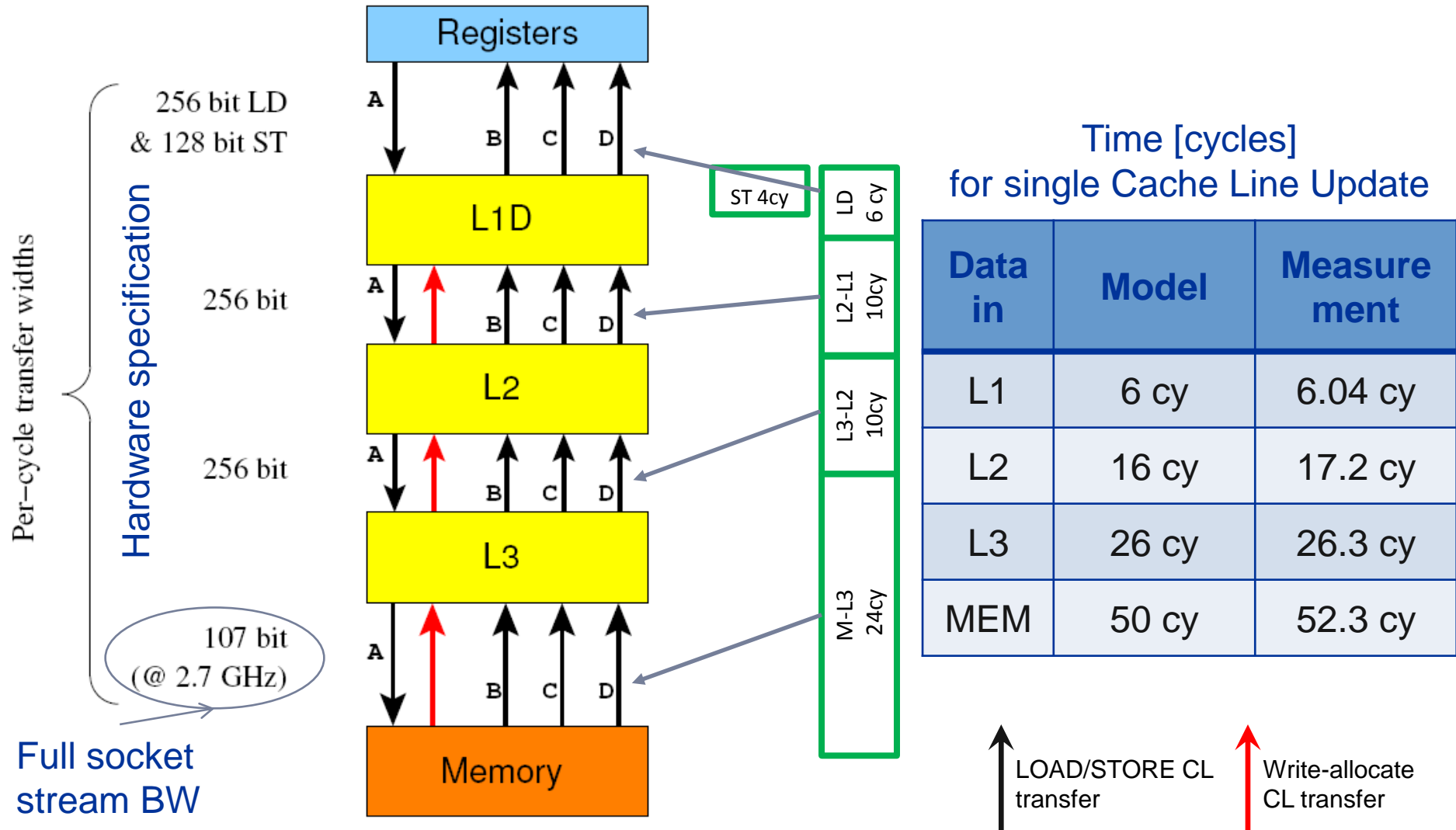
Timeline:



Measurement: $\approx 16 / 17$ F/cy

Example: ECM model for Schönauer Vector Triad

$$A(:,) = B(:,) + C(:,) * D(:,) \text{ with AVX}$$



Multicore scaling in the ECM model

Identify relevant **bandwidth (shared) bottlenecks**

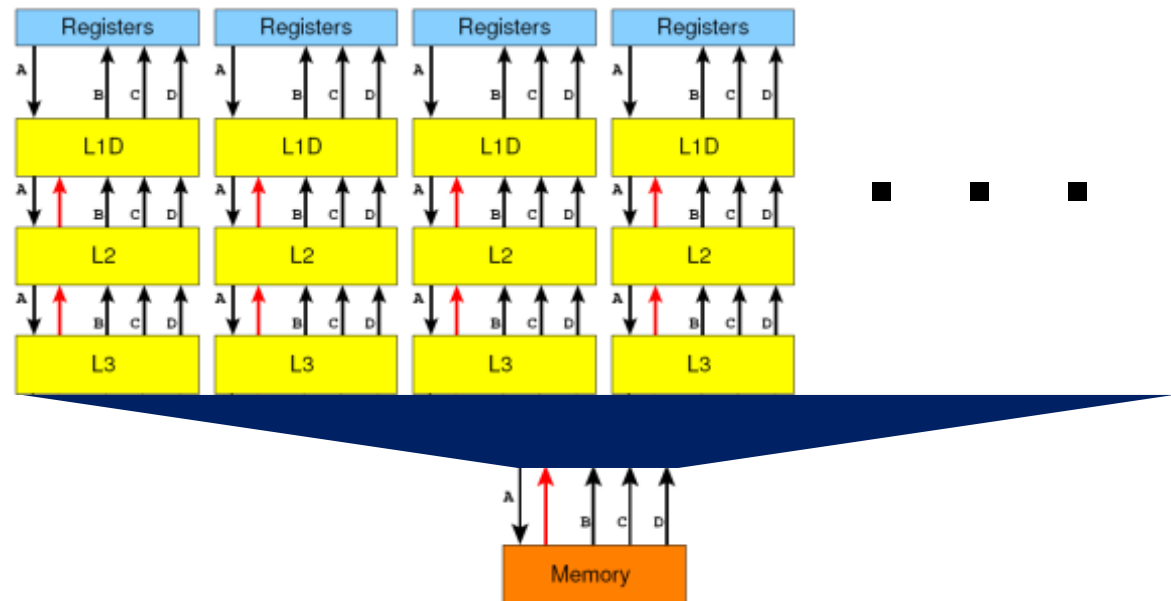
- L3 cache
- Memory interface

Scale single-thread performance until **first bottleneck** is hit:

$$P(t) = \min(t * P_0, P_{\text{roof}}), \text{ with } P_{\text{roof}} = \min(P_{\text{max}}, I * b_s)$$

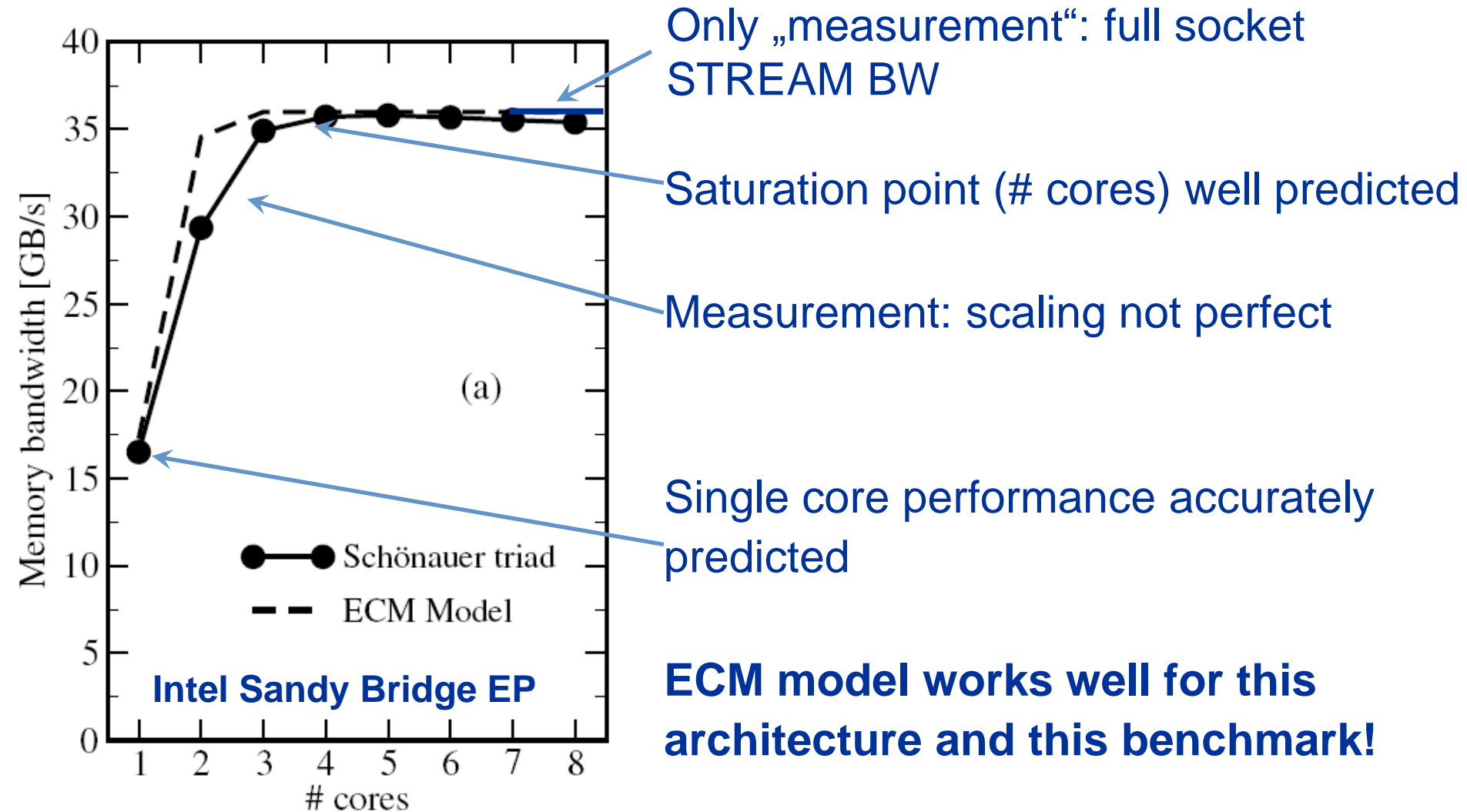
Sandy Bridge: Scalable L3

→ Bottleneck (P_{roof}): Memory



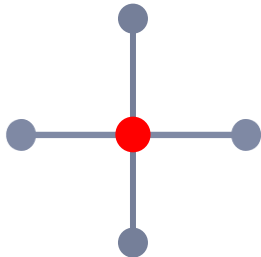
ECM prediction vs. measurements for

$$A(:) = B(:) + C(:) * D(:)$$



Case study 1

ECM modelling for stencil structures:
2D Jacobi (double precision) with SSE2 on SNB



```
1 // Jacobi 2D line update
2 for(int j=start; j<end; j++){
3     t1[i][j]= ( t0[i-1][j]   +
4                 t0[i+1][j]   +
5                 t0[i][j-1]   +
6                 t0[i][j+1] ) * 0.25;
7 }
```

Case study 1: 2D Jacobi in DP with SSE2 on SNB

```
1 // Jacobi 2D line update
2 for(int j=start; j<end; j++){
3     t1[i][j]= ( t0[i-1][j] +
4                 t0[i+1][j] +
5                 t0[i][j-1] +
6                 t0[i][j+1] ) * 0.25;
7 }
```



4-way unrolling
→ 8 LUP / iteration



Instruction count

- 13 LOAD
- 4 STORE
- 12 ADD
- 4 MUL

```
1 movups  (rbp, r15, 8), xmm1
2 movups  16(rbp, r15, 8), xmm3
3 movups  32(rbp, r15, 8), xmm5
4 movups  48(rbp, r15, 8), xmm7
5 addpd   (r9, r15, 8), xmm1
6 addpd   16(r9, r15, 8), xmm3
7 addpd   32(r9, r15, 8), xmm5
8 addpd   48(r9, r15, 8), xmm7
9 addpd   -8(r10, r15, 8), xmm1
10 movups  8(r10, r15, 8), xmm2
11 movups  24(r10, r15, 8), xmm4
12 movups  40(r10, r15, 8), xmm6
13 addpd   xmm2, xmm3
14 addpd   xmm4, xmm5
15 addpd   xmm6, xmm7
16 addpd   xmm2, xmm1
17 addpd   xmm4, xmm3
18 addpd   xmm6, xmm5
19 addpd   56(r10, r15, 8), xmm7
20 mulpd   xmm0, xmm1
21 mulpd   xmm0, xmm3
22 mulpd   xmm0, xmm5
23 mulpd   xmm0, xmm7
24 movups  xmm1, (r11, r15, 8)
25 movups  xmm3, 16(r11, r15, 8)
26 movups  xmm5, 32(r11, r15, 8)
27 movups  xmm7, 48(r11, r15, 8)
```

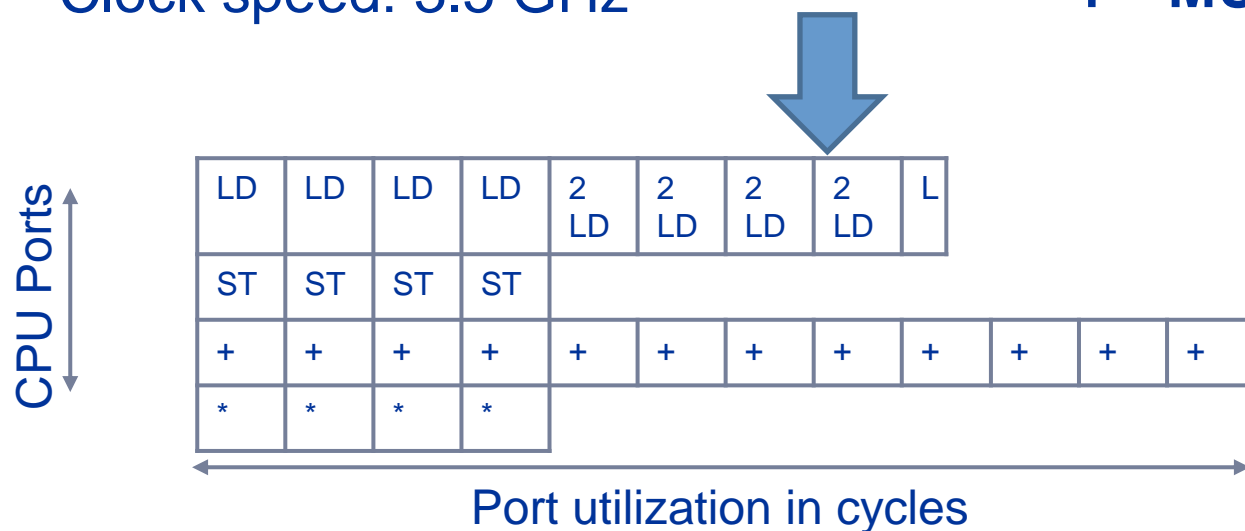
Case study 1: 2D Jacobi in DP with SSE2 on SNB

Machine characteristics (SSE instructions per cycle)

- 2 LOAD || (1 LOAD + 1 STORE)
- 1 ADD
- 1 MUL
- Clock speed: 3.5 GHz

Code characteristics (SSE instructions per CL)

- 13 LOAD
- 4 STORE
- 12 ADD
- 4 MUL



Core execution
12 cycles

Bottleneck
ADD

Case study 1: 2D Jacobi in DP with SSE2 on SNB

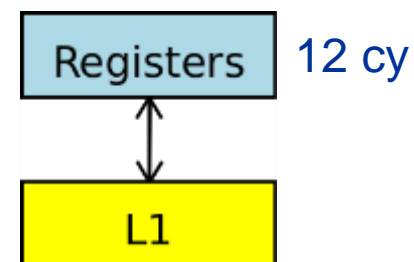
■ Situation 1: Data set fits into L1 cache

■ ECM prediction:

$$(8 \text{ LUP} / 12 \text{ cy}) * 3.5 \text{ GHz} = 2.3 \text{ GLUP/s}$$

■ Measurement:

$$2.2 \text{ GLUP/s}$$



■ Situation 2: Data set fits into L2 cache

- 3 transfer streams from L2 to L1
(LD T_0 + LD/ST T_1)

■ ECM prediction:

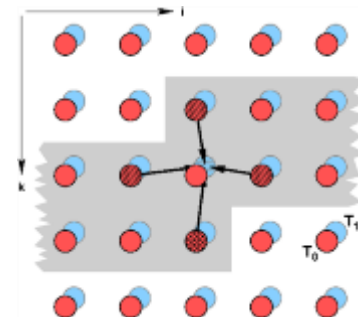
$$(8 \text{ LUP} / (12+6) \text{ cy}) * 3.5 \text{ GHz} = 1.5 \text{ GLUP/s}$$

■ Measurement:

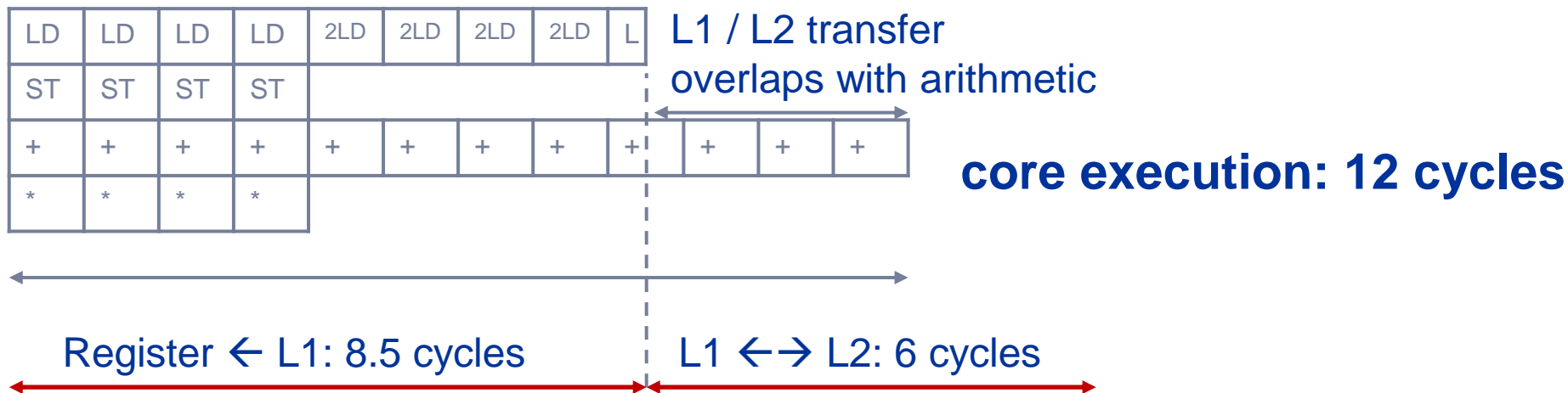
$$1.9 \text{ GLUP/s}$$




What's wrong??

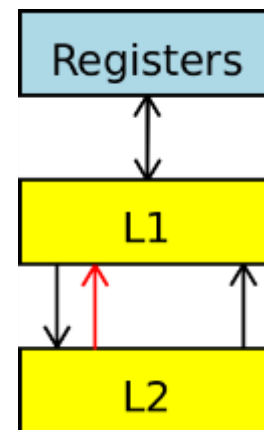


Case study 1: 2D Jacobi in DP with SSE2 on SNB



- ECM prediction:
 $(8 \text{ LUP} / (8.5+6) \text{ cy}) * 3.5 \text{ GHz} = 1.9 \text{ GLUP/s}$
- Measurement:  1.9 GLUP/s

“If the model fails, we learn something”



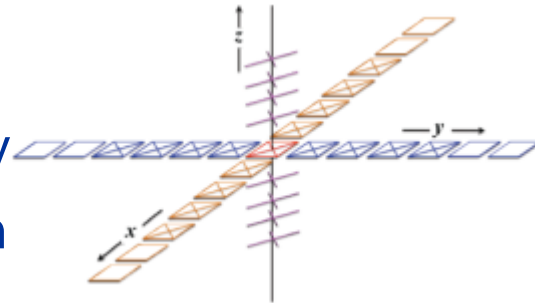
- No concurrent data transfer between memory levels
- Data transfer may overlap with other in-core instructions

Case study 2: Long range 3D stencil

ECM modelling for stencil structures: Long range 3D 25-pt stencil

- Background:

- Three-dimensional Seismic simulations in oil industry
- Finite-Difference-Time-Domain (FDTD) discretization



- Stencil operator is

- Isotropic (symmetric),
- With constant coefficients,
- 2nd order in time
- 8th order in space

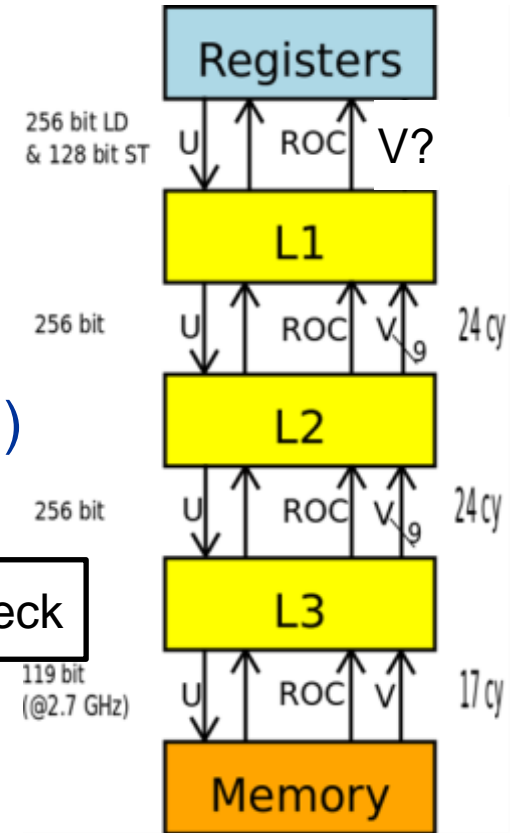
Collaboration with
D. Keyes & T. Malas
(KAUST)

```
1 // 3D long-range line update (single precision)
2 for(i=4; i<nnx-4; i++) {
3     lap = coef0 * V(i,j,k)
4         + coef[1] * ( V(i+1,j ,k ) + V(i-1,j ,k ) )
5         + coef[1] * ( V(i ,j+1,k ) + V(i ,j-1,k ) )
6         + coef[1] * ( V(i ,j ,k+1) + V(i ,j ,k-1) )
7         + coef[2] * ( V(i+2,j ,k ) + V(i-2,j ,k ) )
8         + coef[2] * ( V(i ,j+2,k ) + V(i ,j-2,k ) )
9         + coef[2] * ( V(i ,j ,k+2) + V(i ,j ,k-2) )
10        + coef[3] * ( V(i+3,j ,k ) + V(i-3,j ,k ) )
11        + coef[3] * ( V(i ,j+3,k ) + V(i ,j-3,k ) )
12        + coef[3] * ( V(i ,j ,k+3) + V(i ,j ,k-3) )
13        + coef[4] * ( V(i+4,j ,k ) + V(i-4,j ,k ) )
14        + coef[4] * ( V(i ,j+4,k ) + V(i ,j-4,k ) )
15        + coef[4] * ( V(i ,j ,k+4) + V(i ,j ,k-4) );
16    U(i,j,k) = 2.f * V(i,j,k) - U(i,j,k) + ROC2(i,j,k) * lap;
17 }
```

Innermost loop shown only

Case study 2: Long range 3D stencil

- Data delay below L1 assuming spatial blocking easy to calculate: 65 cy/CL
- Data transfer L1 – register for V depends on compiler
- How to calculate / estimate core execution time and actual load / stores?
→ Use Intel Architecture Code Analyzer (IACA)



```

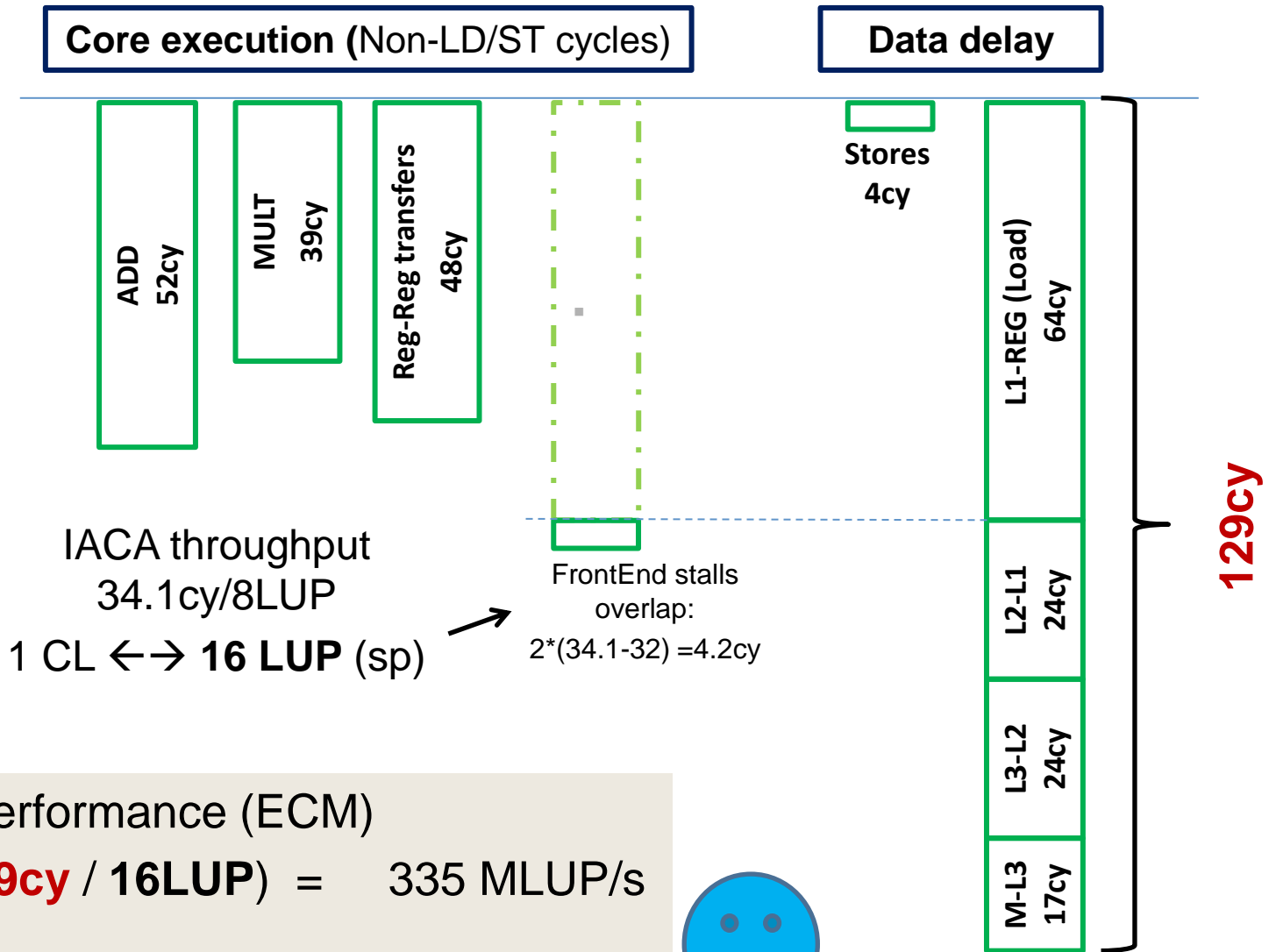
1 Intel(R) Architecture Code Analyzer Version - 2.1
2 Analyzed File - ./kaust-kernel_03_AVX_noinline_noalias_restrict_iaca.o
3 Binary Format - 64Bit
4 Architecture - IVB
5 Analysis Type - Throughput
6
7 Throughput Analysis Report
8 -----
9 Block Throughput: 34.10 Cycles          Throughput Bottleneck: FrontEnd
10
11 Port Binding In Cycles Per Iteration:
12 -----
13 | Port | 0 - DV | 1 | 2 - D | 3 - D | 4 | 5 |
14 -----
15 | Cycles | 19.7  0.0 | 26.0 | 32.0  31.0 | 32.0  31.0 | 2.0 | 24.3 |
16 -----
  
```

Core execution time

Core bottleneck

LD cycles: L1 → Reg.

Case study 2: Long range 3D stencil



Single-core performance (ECM)

$$2.7\text{GHz} / (129\text{cy} / 16\text{LUP}) = 335 \text{ MLUP/s}$$

Measurement: 320 MLUP/s



Case study 3: 3D stencil with DIVide

ECM modelling for stencil structures: Impact of complex arithmetic

Background:

- Three-dimensional Seismic simulations
- Unelastic Wave Propagation Code (<http://hpgeoc.sdsc.edu/personnel.html>)
- Finite-Difference-Time-Domain (FDTD) discretization

```
1 - for(Jb=nyb; Jb<=nye; Jb+=BS){
2     Je = (Jb+BS)<=nye ? (Jb+BS):nye;
3     #pragma omp parallel for private(k,j,i) schedule(runtime)
4 -   for(k=nzb; k<=nze; k++){
5     -   for (j=Jb; j<=Je; j++){
6     -     #pragma simd
7     -     for (i=nxb; i<=nxe; i++){
8     -       float d=0.25*(d1[IDX(i,j,k)] +d1[IDX(i,j-1,k)]
9     -         + d1[IDX(i,j,k-1)]+d1[IDX(i,j-1,k-1)]);
10    -     u1[IDX(i,j,k)]=u1[IDX(i,j,k)] + (dth/d) * (
11    -       c1*(xx[IDX(i,j,k)] - xx[IDX(i-1,j,k)])
12    -       + c2*(xx[IDX(i+1,j,k)] - xx[IDX(i-2,j,k)])
13    -       + c1*(xy[IDX(i,j,k)] - xy[IDX(i,j-1,k)])
14    -       + c2*(xy[IDX(i,j+1,k)] - xy[IDX(i,j-2,k)])
15    -       + c1*(xz[IDX(i,j,k)] - xz[IDX(i,j,k-1)])
16    -       + c2*(xz[IDX(i,j,k+1)] - xz[IDX(i,j,k-2)]));
17    -   }
18  }
```

DIV operation

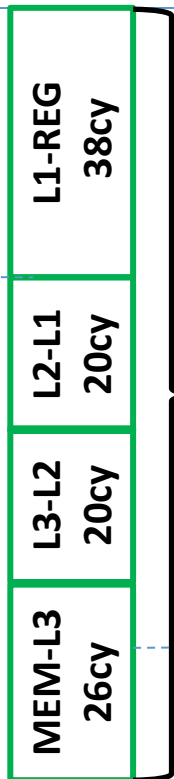
Collaboration with
O. Schenk
(USI Lugano)

Case study 3: 3D stencil with DIV



Compiler avoids DIV instruction

FrontEnd stalls



DIV becomes bottleneck for temporal blocking!

Single-core performance (SP)

ECM (**104cy**): 415 MLUP/s
Measurement: 406 MLUP/s

Single-core performance (DP)

ECM (**104cy**): 208 MLUP/s
Measurement: 179 MLUP/s

Conclusions

- ECM allows good estimate/prediction of single core performance and socket scalability for “streaming kernels”
- ECM reveals computational/hardware bottlenecks and quantifies their impact on performance
- ECM allows to quantify impact of clock speed or vectorization
- ECM could be integrated into autotuning frameworks or DSLs

Work is funded by



DFG Priority Programme 1648



Bavarian Network for HPC

High Performance Computing is Computing at a Bottleneck!*

References – ECM model basics & applications

1. G. Hager, J. Treibig, J. Habich, and G. Wellein: *Exploring performance and power properties of modern multicore chips via simple machine models*. (accepted) *Concurrency and Computation: Practice and Experience*, DOI: [10.1002/cpe.3180](https://doi.org/10.1002/cpe.3180) (2013). Preprint: [arXiv:1208.2908](https://arxiv.org/abs/1208.2908)
2. M. Wittmann, G. Hager, T. Zeiser, J. Treibig, and G. Wellein: *Chip-level and multi-node analysis of energy-optimized lattice-Boltzmann CFD simulations*. Submitted. Preprint: [arXiv:1304.7664](https://arxiv.org/abs/1304.7664)
3. 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](#). [Euro-Par 2012: Parallel Processing Workshops](#), LNCS 7640, 451-460 (2013), Springer, ISBN 978-3-642-36948-3. DOI: [10.1007/978-3-642-36949-0_50](https://doi.org/10.1007/978-3-642-36949-0_50). Preprint: [arXiv:1206.3738](https://arxiv.org/abs/1206.3738)