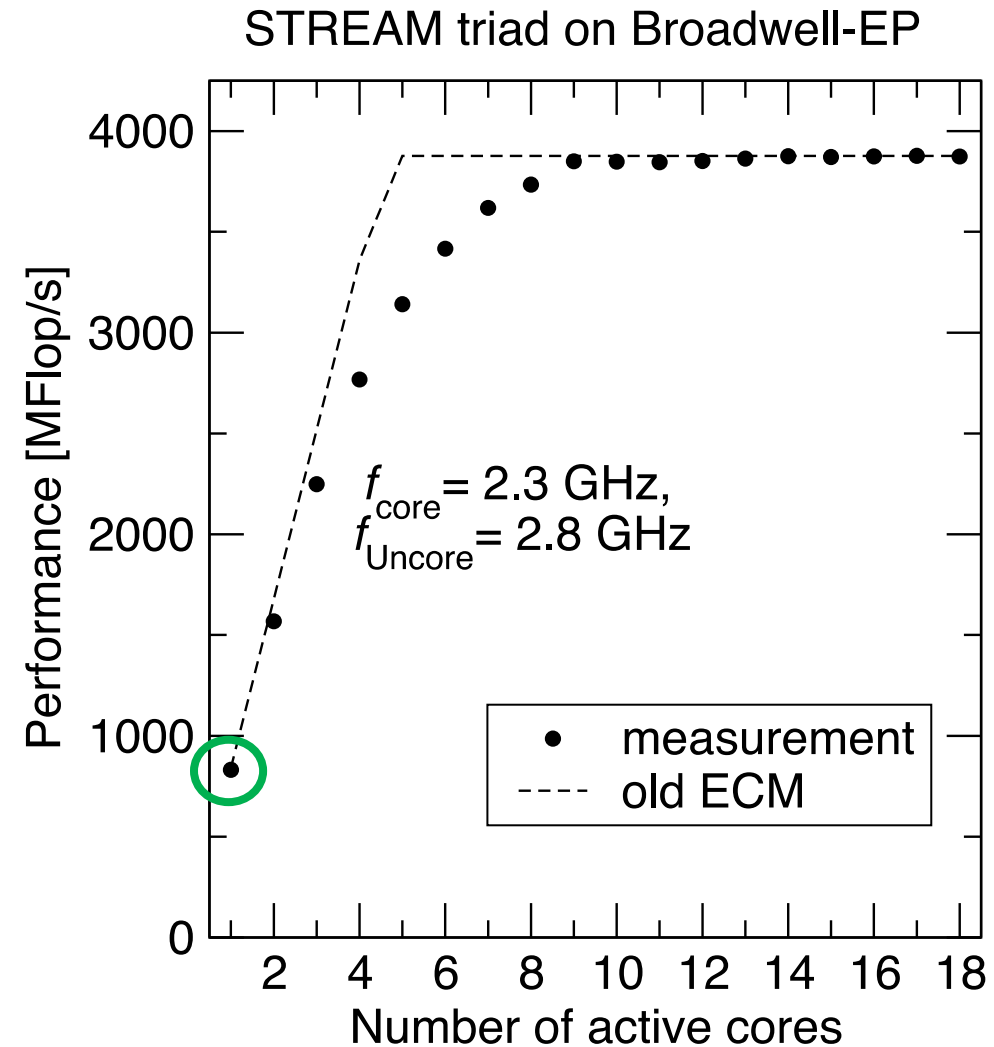


# The Execution-Cache-Memory (ECM) Performance Model



# Motivation

Searching a good model for the single core performance of streaming loop kernels



# The ECM Model

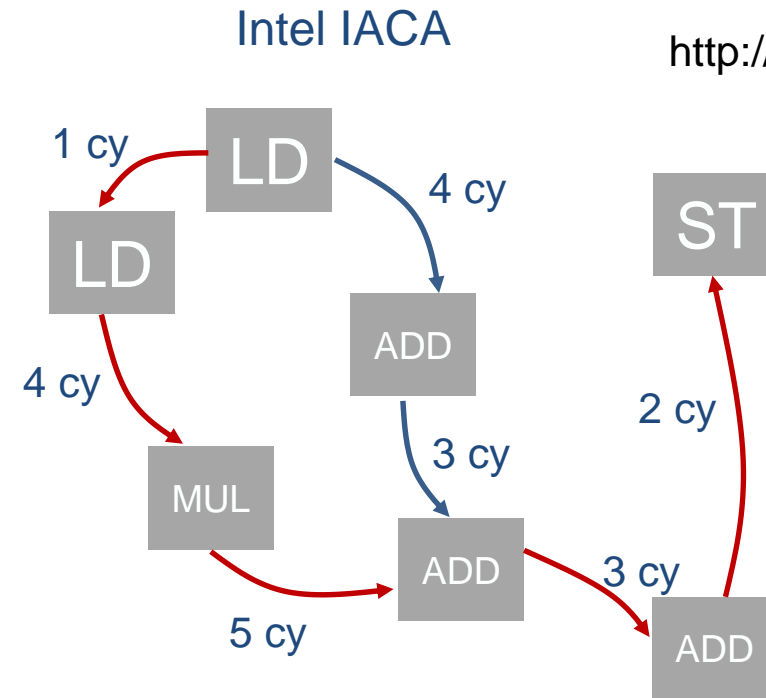
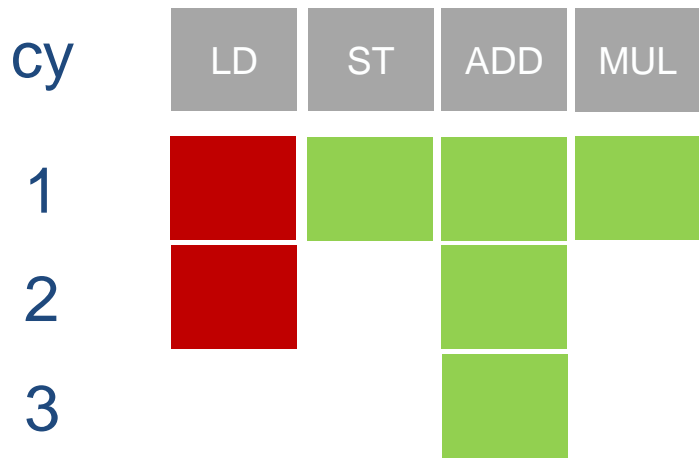
---

ECM is a **resource-based model** for the **runtime** of loops on one core of a cache-based multicore CPU

Major model assumptions:

- **Steady-state** loop code execution
  - No startup latencies, “infinitely long loop”
- **No data access latencies**
  - Can be added if need be
- **Out-of-order** scheduler works perfectly
  - But dependencies/critical paths can be taken into account

# ECM model components: In-core execution



Best case: max throughput

Worst case: critical path

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

$$T_{\text{core}}^{\max} = T^{\text{CP}}$$

$T_{\text{nOL}}$  interacts with cache hierarchy,  $T_{\text{OL}}$  does not

# ECM model components:

## Data transfer times

<http://tiny.cc/kerncraft>

**KERNCRAFT**

Automatic Roofline/ECM  
modeling tool

- Optimistic transfer times through mem hierarchy

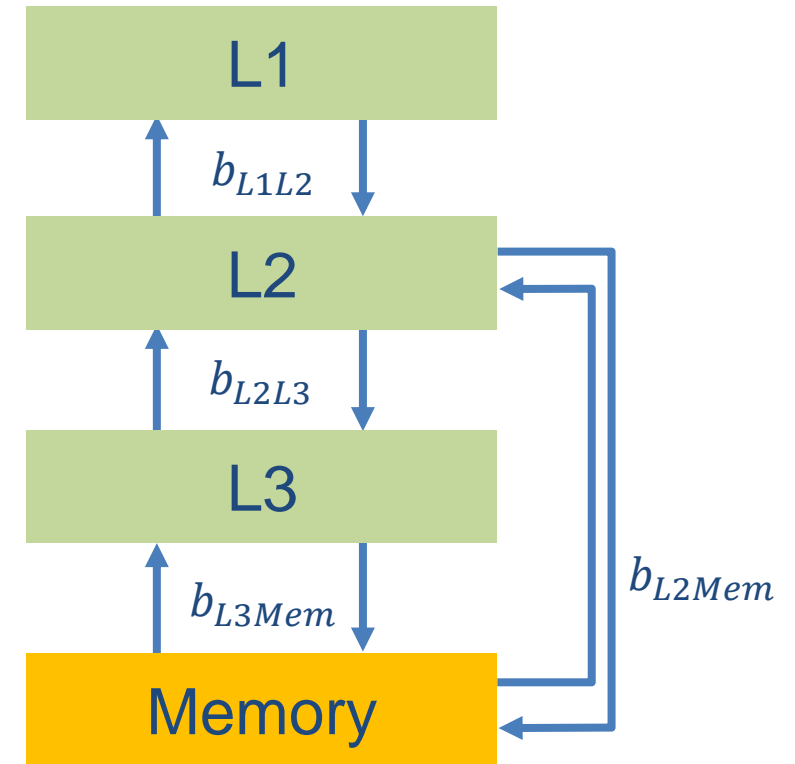
- $T_i = \frac{V_i}{b_i}$

- Transfer time notation for a given loop kernel:

$$\{T_{L1L2} | T_{L2L3} | T_{L3Mem}\} = \{4 | 8 | 18.4\} \text{ cy}/8 \text{ iter}$$

- Input:
  - Cache properties (bandwidths, inclusive/exclusive)
  - Saturated memory bandwidth
  - Application data transfer prediction

Cache architecture & capabilities



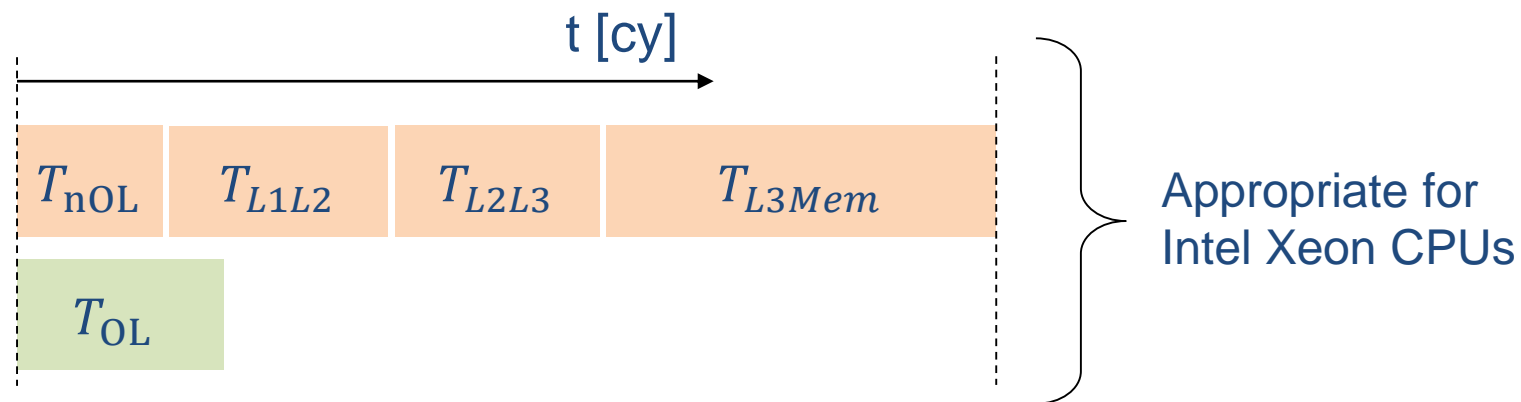
# ECM model components: Overlap assumptions (1)

- Notation for model contributions

$$\{T_{OL} \parallel T_{nOL} | T_{L1L2} | T_{L2L3} | T_{L3Mem}\} = \{7 \parallel 2 | 4 | 8 | 18.4\} \text{ cy/8 iter}$$

- Most pessimistic overlap model: no overlap

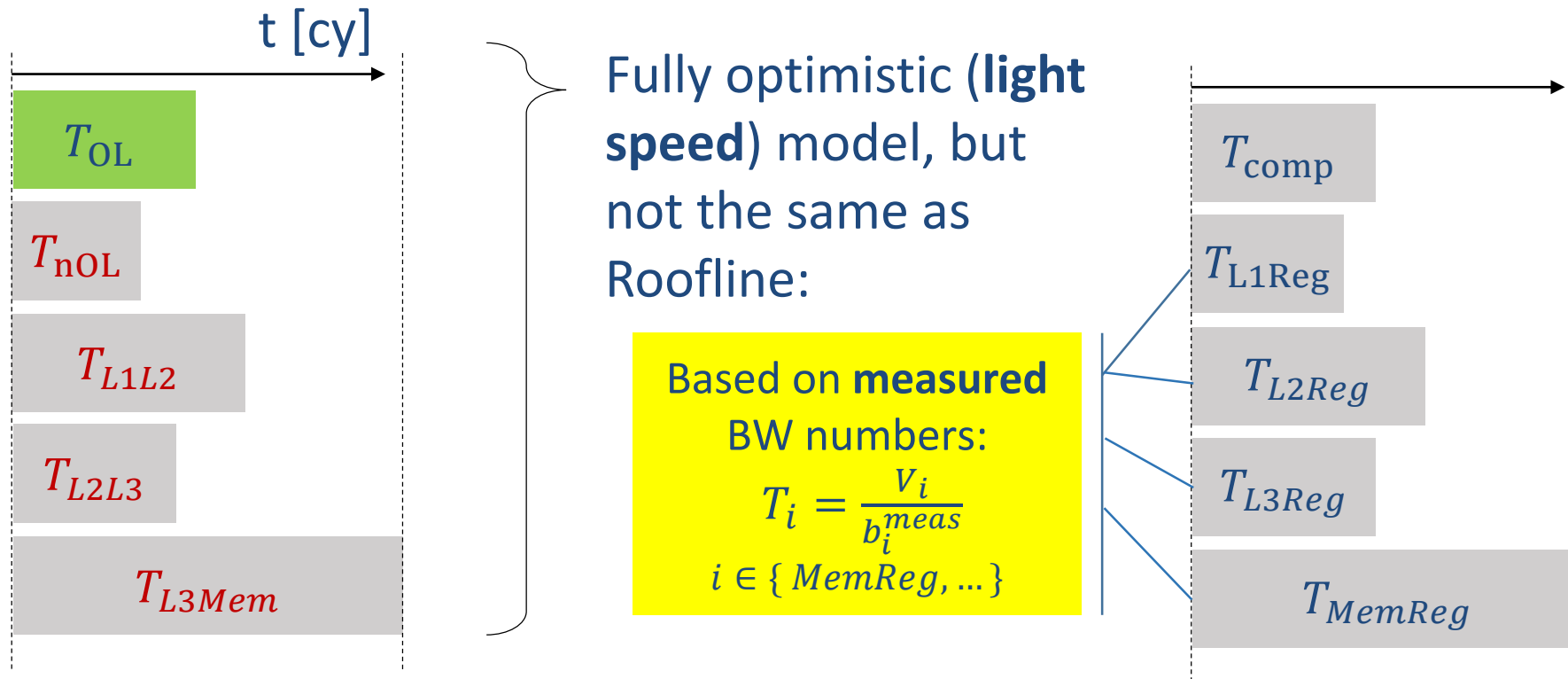
$$T_{ECM}^{Mem} = \max(T_{OL}, T_{nOL} + T_{L1L2} + T_{L2L3} + T_{L3Mem}) \text{ for in-mem data}$$



# ECM model components: Overlap assumptions (2)

Most optimistic assumption: full overlap of data-related contributions

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

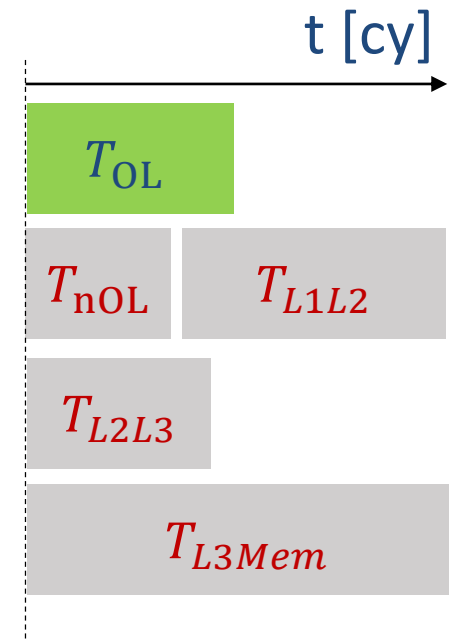


# ECM model components: Overlap assumptions (3)

Mixed model: partial overlap of data-related contributions

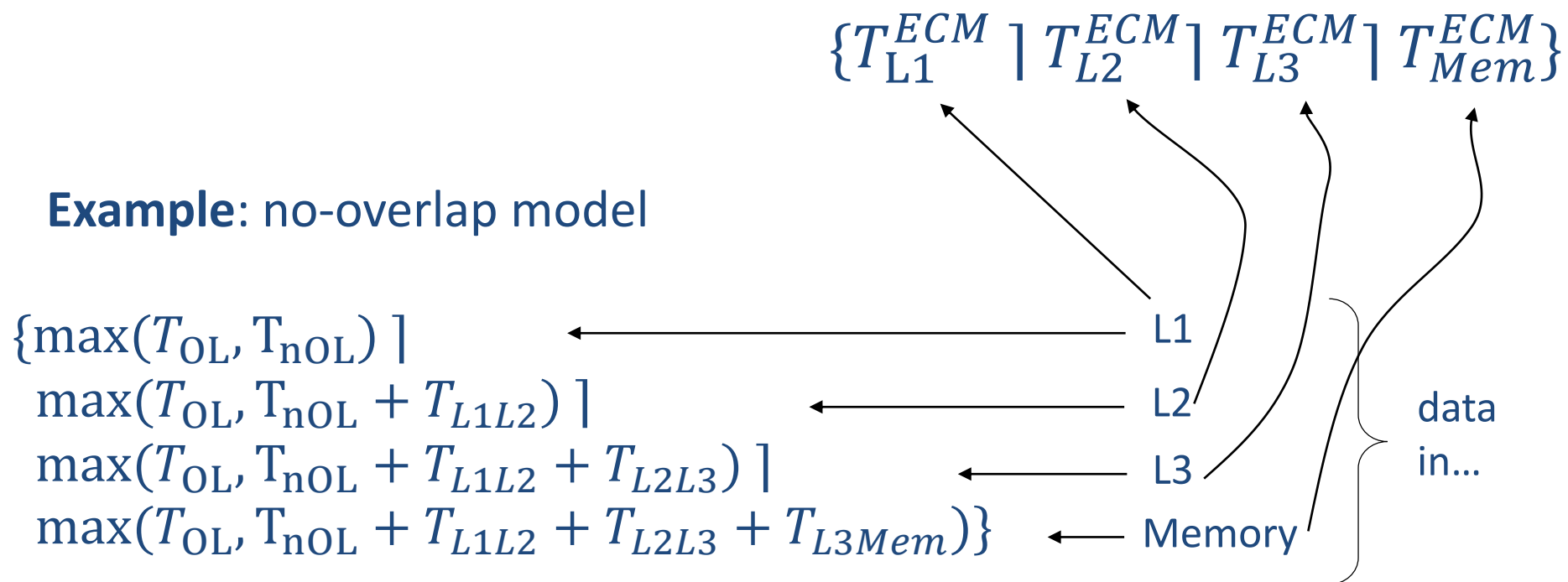
Example: no overlap at L1, full overlap of all other contributions

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





# ECM model: Notation for runtime predictions

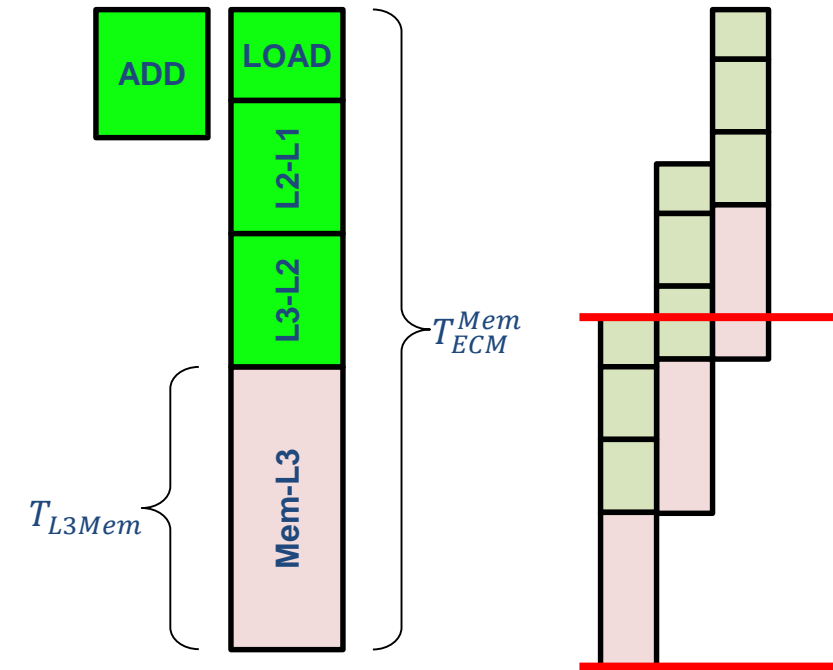


# ECM model: (Naive) saturation assumption

- Performance is assumed to scale across cores until a shared bandwidth bottleneck is hit

$$T_{ECM}(n) = \max\left(\frac{T_{Mem}^{ECM}}{n}, T_{L3Mem}\right) \Rightarrow n_S = \left\lfloor \frac{T_{ECM}^{Mem}}{T_{L3Mem}} \right\rfloor$$

Roofline bandwidth ceiling



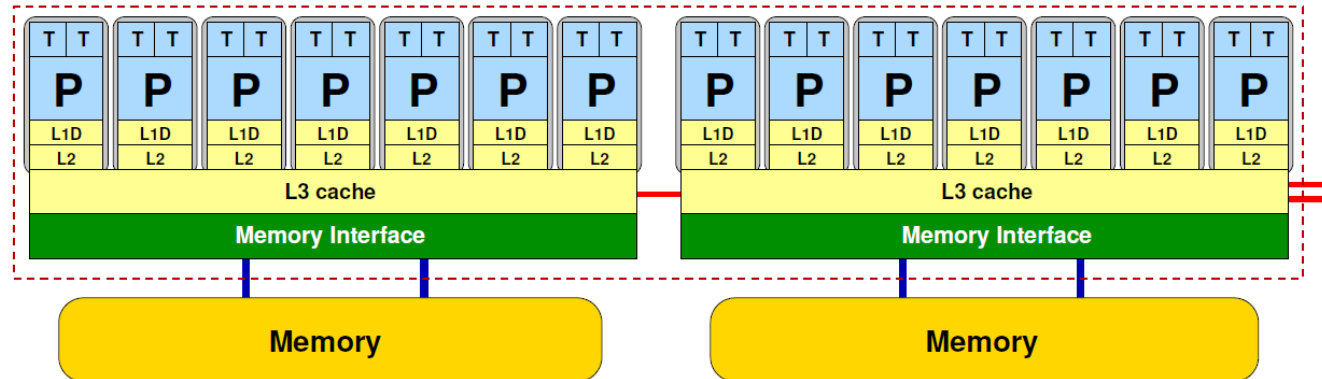
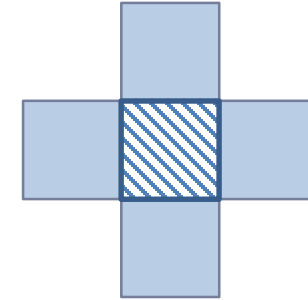
# Modeling a Conjugate-Gradient Solver

Building a model from components



# A matrix-free CG solver

- 2D 5-pt FD Poisson problem
- Dirichlet BCs, matrix-free
- $N_x \times N_y = 40000 \times 1000$  grid
- CPU: Haswell E5-2695v3 CoD mode



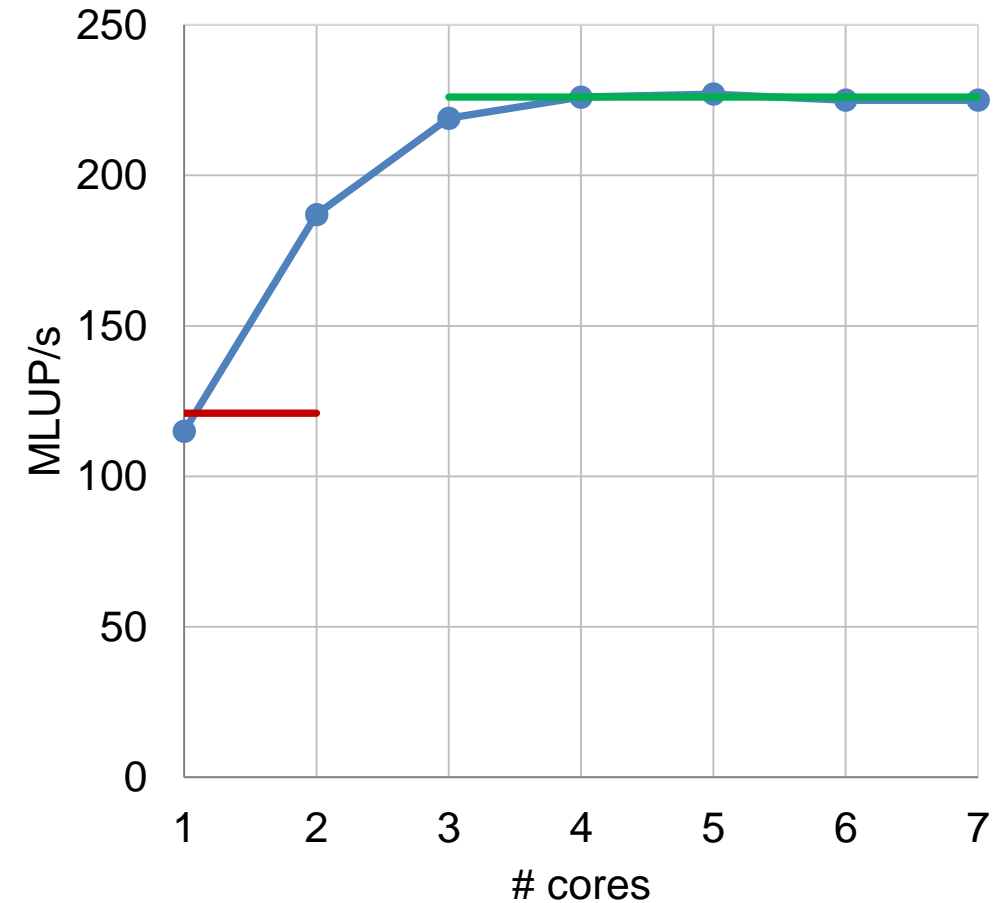
# ECM model composition

Naive implementation of all kernels (omp parallel for)

<b>while(<math>\alpha_0 &lt; \text{tol}</math>):</b>	<b><math>T_x</math> [cy/8 iter]</b>	<b><math>T_{Mem}^{ECM}</math> [cy/8 iter]</b>	<b><math>n_s</math> [cores]</b>	<b>Full domain limit [cy/8 iter]</b>
$\vec{v} = A\vec{p}$	{ 8    4   6.7   10   16.9 }	37.6	3	16.9
$\lambda = \alpha_0 / \langle \vec{v}, \vec{p} \rangle$	{ 2    2   2.7   4   9.1 }	17.8	2	9.11
$\vec{x} = \vec{x} + \lambda\vec{p}$	{ 2    4   6   16.9 }	29.0	2	16.9
$\vec{r} = \vec{r} - \lambda\vec{v}$	{ 2    4   6   16.9 }	29.0	2	16.9
$\alpha_1 = \langle \vec{r}, \vec{r} \rangle$	{ 2    2   1.3   2   4.6 }	9.90	3	4.56
$\vec{p} = \vec{r} + \frac{\alpha_1}{\alpha_0}\vec{p}, \alpha_0 = \alpha_1$	{ 2    4   6   16.9 }	29.0	2	16.9
	Sum	<b>152</b>		<b>81.3</b>

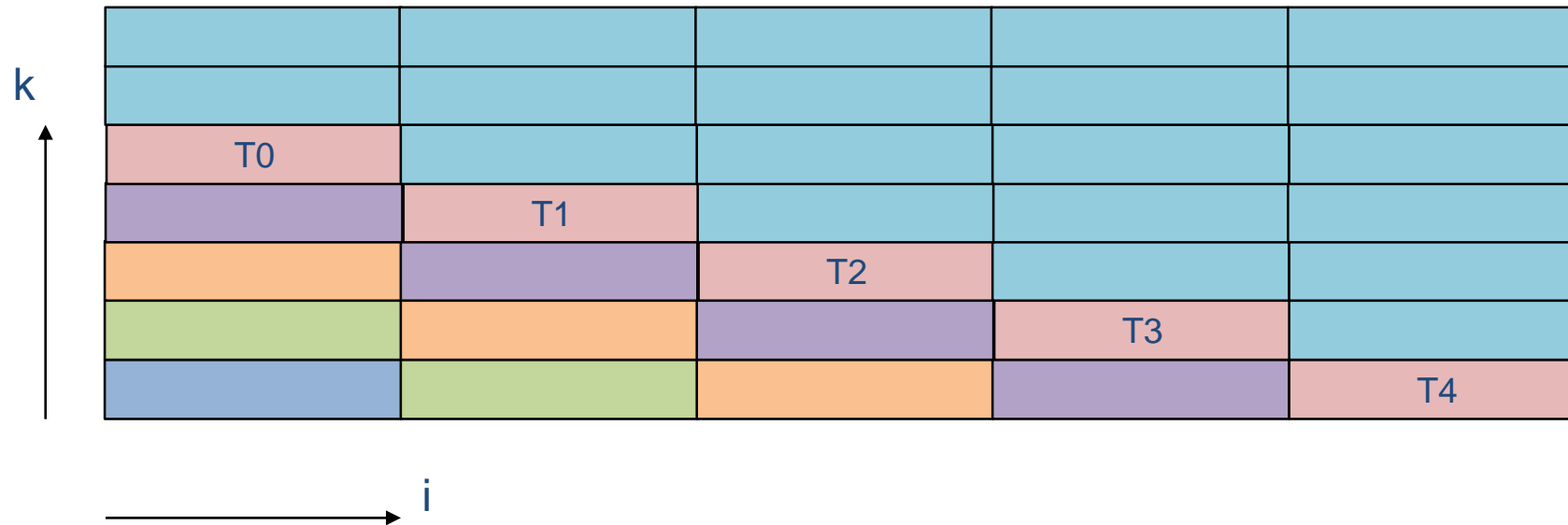
# CG performance – 1 core to full socket

- Multi-loop code well represented
- **Single core** performance predicted with 5% error
- **Saturated performance** predicted with < 0.5% error
- Saturation point predicted approximately
  - Can be fixed by improved ECM model



# CG with GS preconditioner: Naïve parallelization

Pipeline parallel processing: OpenMP barrier after each wavefront update (ugh!)



# CG with GS preconditioner: additional kernels

Intel IACA

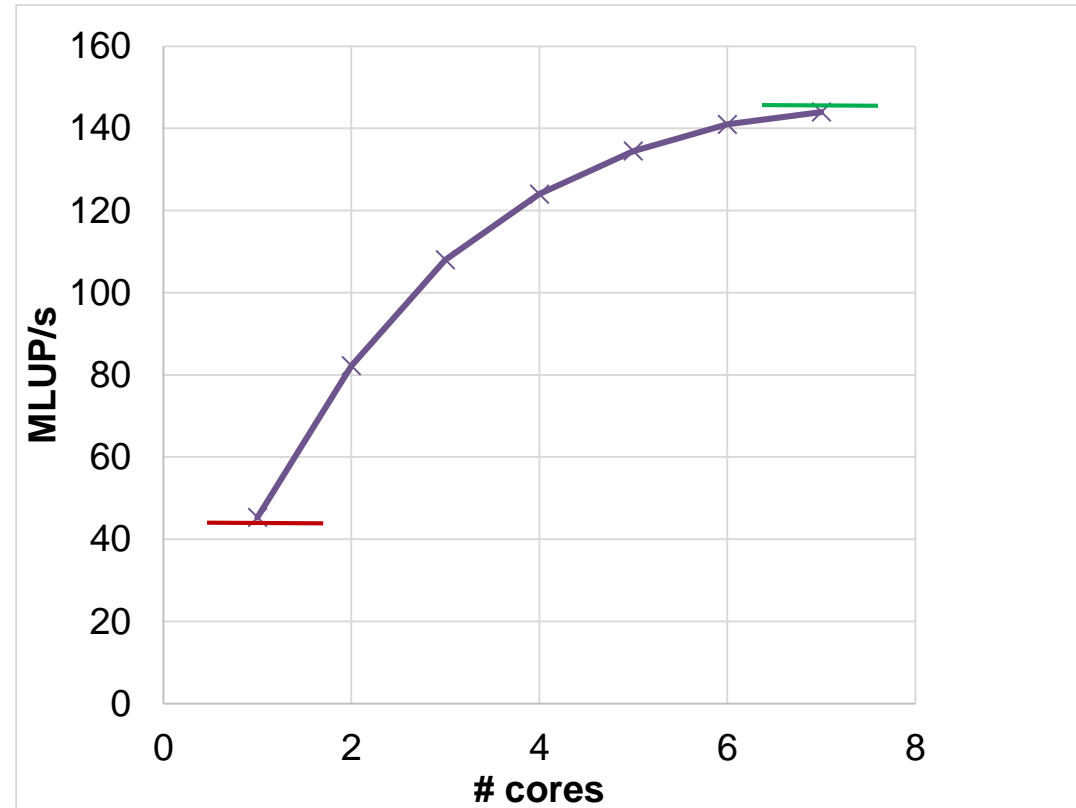
	$T_x$ [cy/8 iter]	$T_{Mem}^{ECM}$ [cy/8 iter]	$n_s$ [cores]	Full domain limit [cy/8 iter]
Non-PC model		<b>152</b>		<b>81.3</b>
$\vec{z} = P\vec{r}$ (fw)	{ 108   16   5.4   8   16.9 }	108	7	16.9
$\vec{z} = P\vec{r}$ (bw)	{ 138   16   4.0   6   11.3 }	138	<b>13</b>	19.7
$\alpha = \langle \vec{r}, \vec{z} \rangle$	{ 2   2   2.7   4   9.1 }	17.8	2	9.1
	Sum	<b>416</b>		<b>127</b>

- Back substitution does not saturate the memory bandwidth!
  - → full algorithm does not fully saturate
- Impact of barrier still negligible overall, but noticeable in the preconditioner



# PCG measurement

- <2% model error for single threaded and saturated performance
- Expected large impact of barrier at smaller problem sizes in x direction



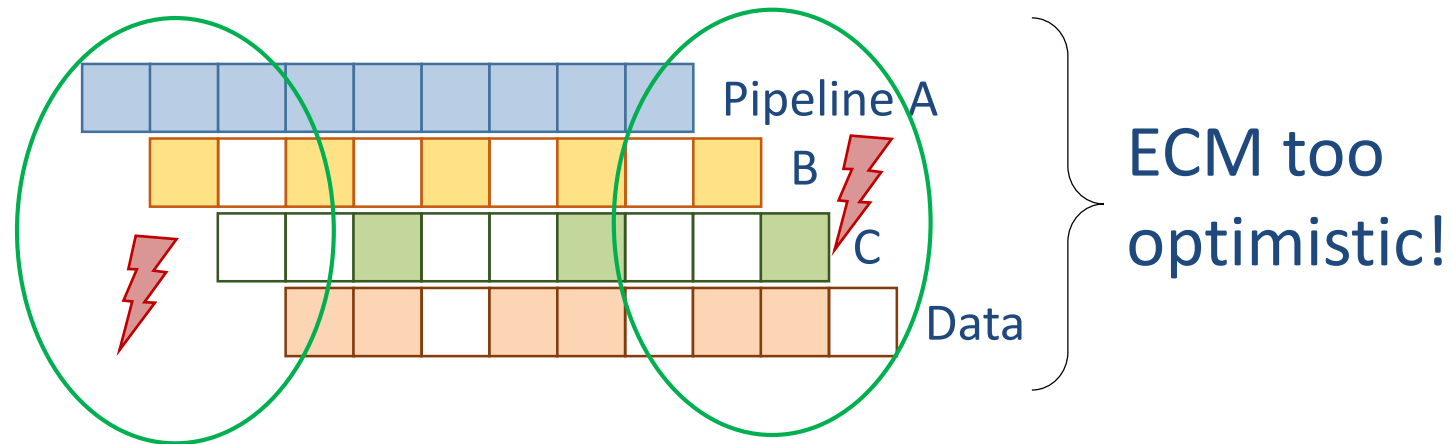
## Problems and Open Questions

What ECM cannot do (well)



# Non-steady-state execution

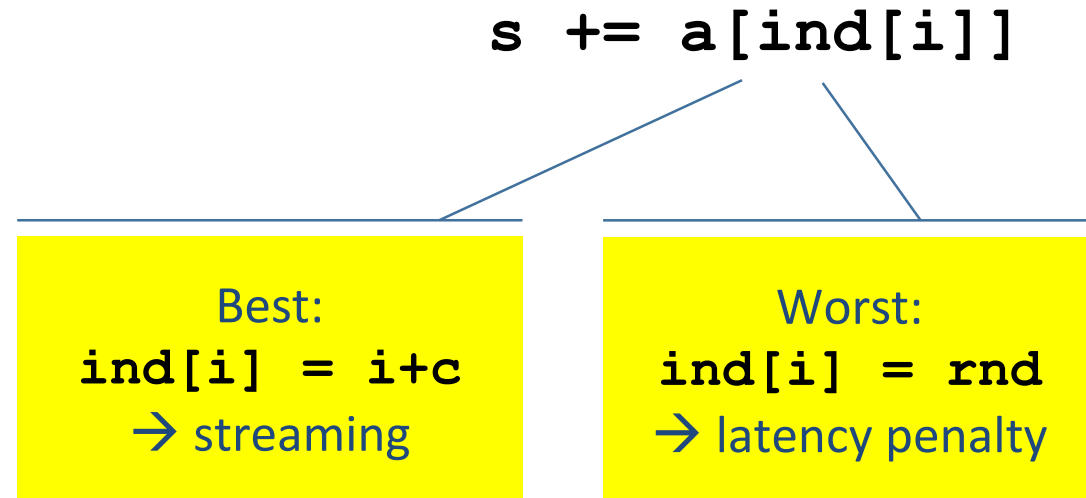
- Wind-up/wind-down effects are not part of the model



- May be added via corrections

# Irregular data access

- Indirect  $\neq$  irregular



- Unknown access order → only best/worst-case analysis possible

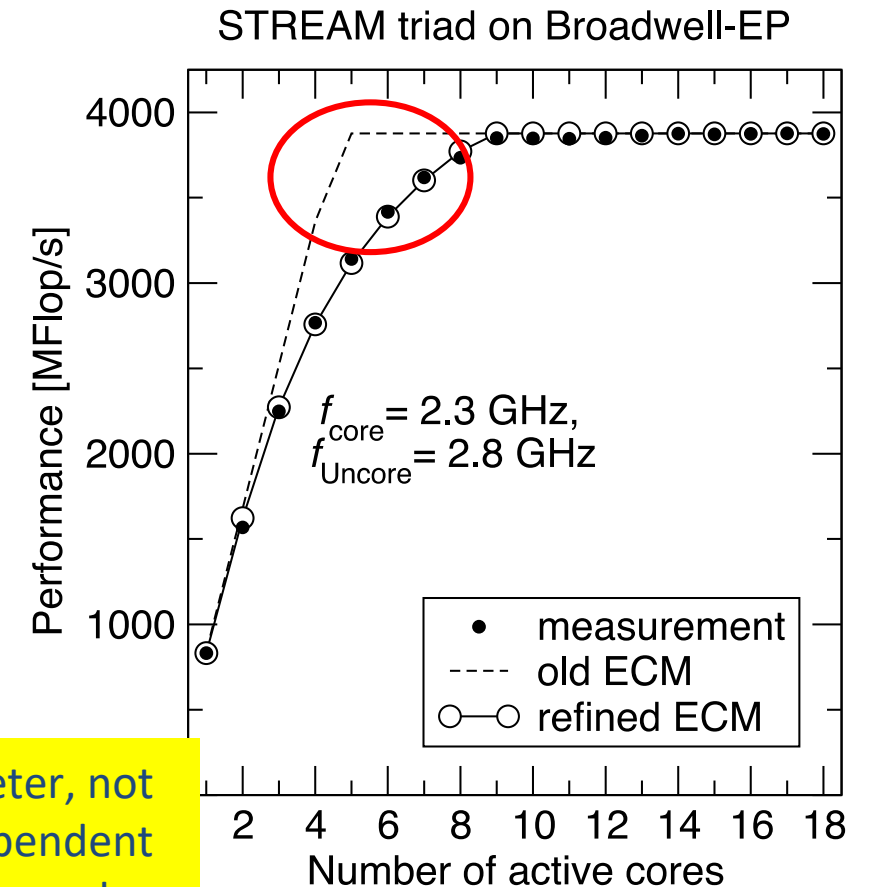
# Saturation

- Original ECM model too optimistic near saturation point
- Refinement: Adaptive latency penalty, depends on bus utilization  $u(n)$ :

$$u(1) = \frac{T_{L3Mem}}{T_{Mem}^{ECM}} \leftarrow \text{single-core model}$$

$$u(n) = \frac{T_{L3Mem}}{T_{Mem}^{ECM} + (n-1)u(n-1)p_0}$$

Fit parameter, not code independent  
→ future work



J. Hofmann, C. L. Alappat, G. Hager, D. Fey, and G. Wellein: *Bridging the Architecture Gap: Abstracting Performance-Relevant Properties of Modern Server Processors*. Supercomputing Frontiers and Innovations **7**(2), 54-78, July 2020. Available with Open Access. DOI: [10.14529/jsfi200204](https://doi.org/10.14529/jsfi200204).

# Tutorial conclusion

---

- Know your system (node) **architecture**
- Enforce **affinity**
- **Back-of-the-envelope models** are extremely useful
- **Modeling is not always predictive**
- **Bottleneck awareness rules**
- Performance is not about tools. **Use your brain!**