From numbers to insight via performance models

Georg Hager

Erlangen National High Performance Computing Center (NHR@FAU)

IACS Seminar, University of Stony Brook

2021-10-14

Overview

- White-/gray-box performance modeling
- Introduction to resource-based modeling
- The Execution-Cache-Memory (and Roofline) model
- Going beyond the node: The pitfalls of highly parallel modeling

Motivation

- Analytic 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?
  - What is the next bottleneck after optimization?
  - Impact of hardware features → co-design, architectural exploration

- What if the model fails?
  - We learn something
  - We may still be able to use the model in a less predictive way

Getting a little more specific

What data/knowledge can a model be based on?

- Only documented hardware properties + hypotheses
  - Purely analytic model

white box

- Hardware properties + measurements + hypotheses
  - (Partly) phenomenological model

gray box

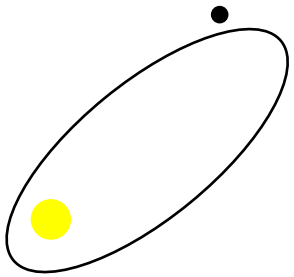
- Measured performance/speedup data + hypotheses
  - Curve-fitting analytic model

black box

An example from physics

$$\vec{F}_{12} = \frac{\gamma m_1 m_2}{r_{12}^3}\vec{r}_{12}$$

$$h(t) = h_0 - \frac{1}{2} g\, t^2$$



Examples for white-/gray-box models in computing

program speedup

serial fraction

$$S(N) = \frac{1}{s + \frac{1-s}{N} + c(N)}$$

Amdahl's Law with communication

latency

msg. length

bandwidth

$$T_{PtP} = T_l + \frac{L}{B}$$

Hockney model for message transmission time

time for computation

time for data transfer

$$T_{\text{exec}} = \max(T_{\text{calc}}, T_{\text{data}})$$

Roofline model for loop code execution time

non-overlapping execution

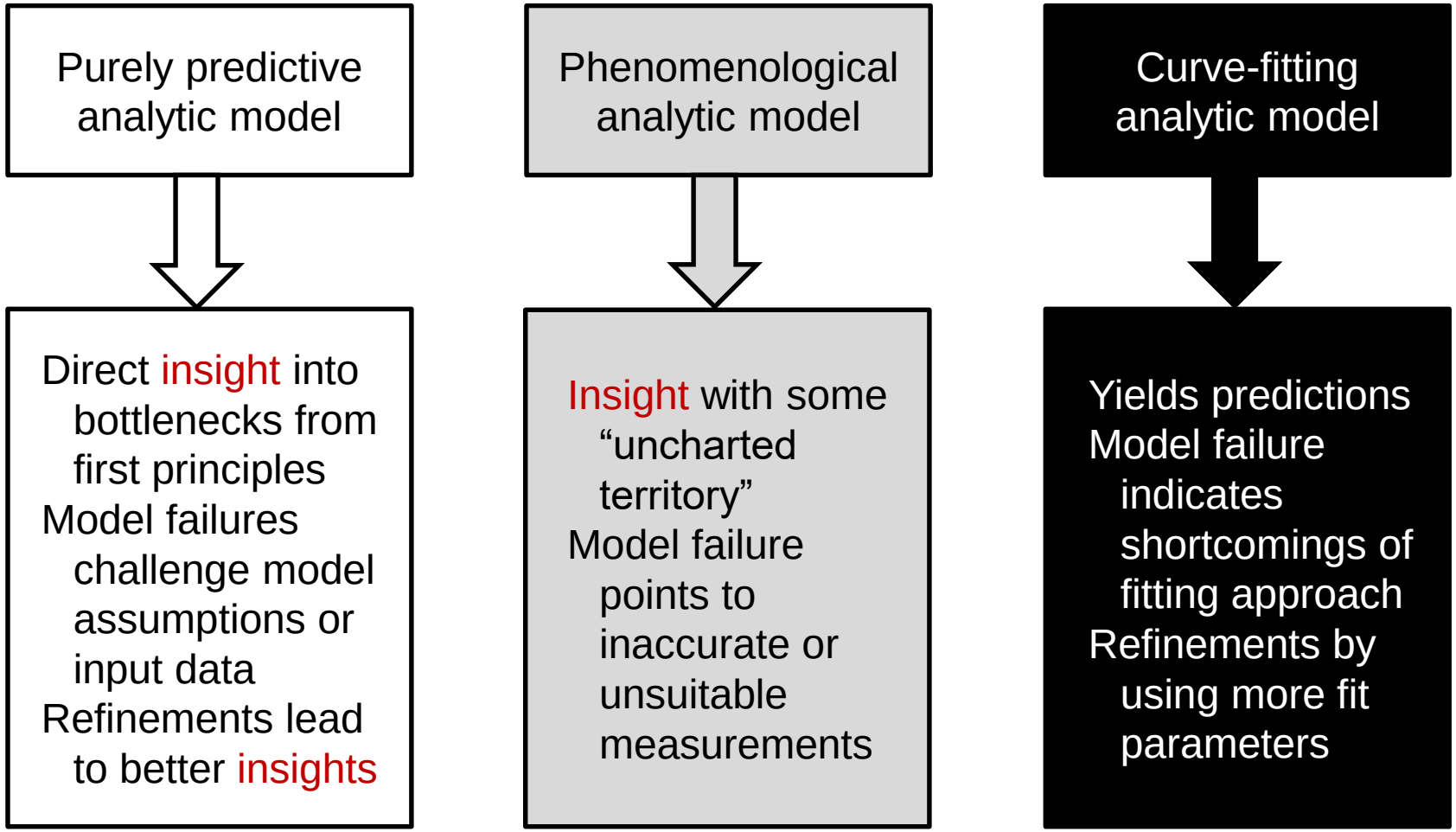
time for data transfer

$$T_{\text{exec}} = f(T_{nOL}, T_{\text{data}}, T_{OL})$$

overlapping execution

ECM model for loop code execution time

Models and insights



NHR FAU

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Resource-based performance models

The questions we ask

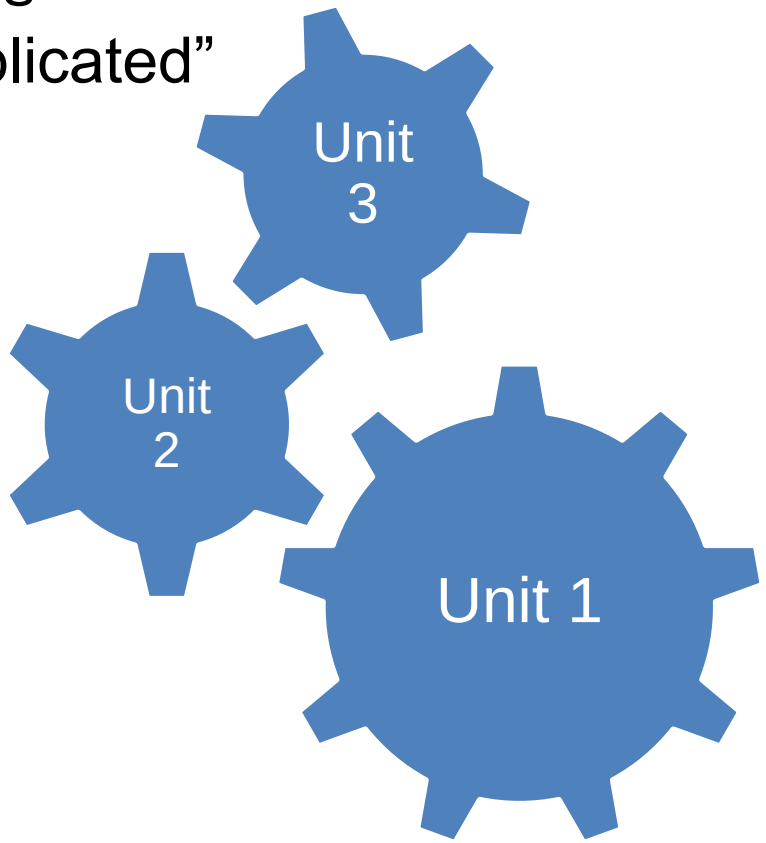
How much of `$RESOURCE` does `$STUFF` need on `$HARDWARE`, and why?

→ Analytic, resource-based, first-principles models

Mechanistic vs. resource-based modeling

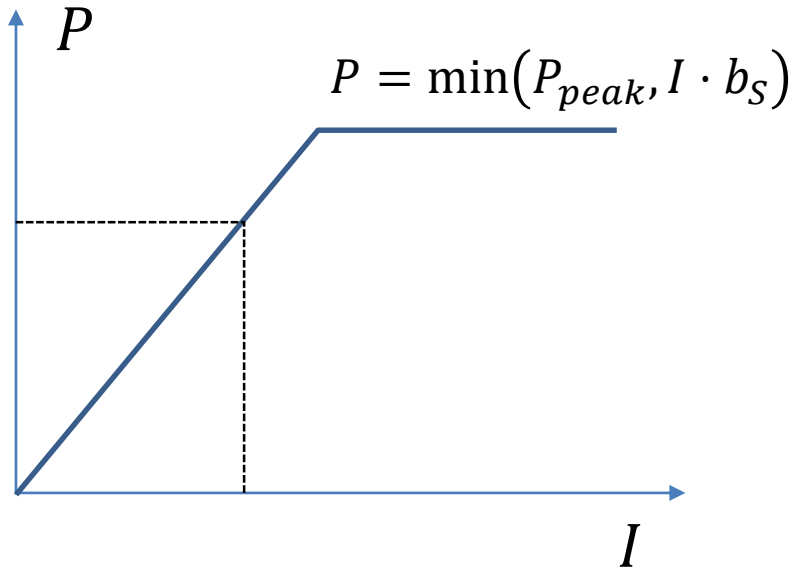
Mechanistic

- Cycle-by-cycle
- Latency
- Simulators
- “It’s complicated”



Resource based

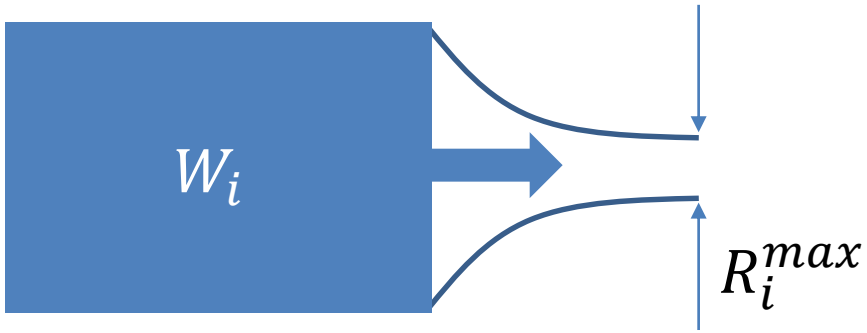
- Resource utilization
- Data flow
- (Non-)overlapping components
- Simplify to make manageable
- If it doesn’t work, refine and iterate



A general view on resource bottlenecks

- What is the maximum performance when limited by a bottleneck?
- Resource bottleneck i delivers resources at maximum rate R_i^{max}
- W_i = needed amount of resources

- Minimum runtime: $T_i = \frac{W_i}{R_i^{max}} + \lambda_i$

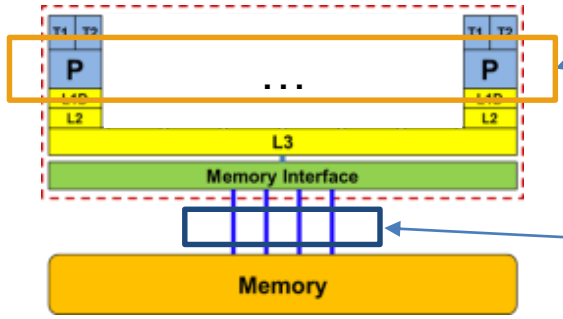


- Multiple bottlenecks → multiple min. runtimes: $T_{\text{expect}} = f(T_1, \ldots T_n)$
- Overall performance: $P_{\text{expect}} = \frac{W}{T_{\text{expect}}}$

A bottleneck model of computing

Example: two bottlenecks

```
#pragma omp parallel for
for(i=0; i<10^7; ++i)
  a[i] = a[i] + s * c[i];
```



$R_{flops}^{max} = 768 \frac{\text{Gflops}}{\text{s}}$

$R_{BW}^{max} = 210 \frac{\text{Gbyte}}{\text{s}}$

n-core CPU
(1 CMG A64FX 2 GHz)

$W_{flops} = 2 \times 10^7 \text{ flops}$

$W_{BW} = 3 \times 8 \times 10^7 \text{ bytes}$

$$T_{flops} = \frac{2 \times 10^7 \text{ flops}}{768 \frac{\text{Gflops}}{\text{s}}} = 26.0\ \mu s$$

$$T_{BW} = \frac{2.4 \times 10^8 \text{ bytes}}{210 \frac{\text{Gbyte}}{\text{s}}} = 1.14 \text{ ms}$$

Bottleneck models

How do we reconcile the multiple bottlenecks?
I.e., what is the functional form of $f(T_1, \ldots T_n)$?

→ pessimistic model (no overlap): $f(T_1, \ldots T_n) = \sum_i T_i$
→ optimistic model (full overlap): $f(T_1, \ldots T_n) = \max(T_1, \ldots T_n)$

Roofline model
(Hockney et al., 1989
Williams et al., 2008)

Roofline for our example: $T_{\min} = \max(T_{flops}, T_{BW}) = 1.14\ \text{ms}$

Maximum performance ("light speed"): $P_{\text{expect}} = \frac{2\times10^7}{1.14\times10^{-3}} \frac{\text{flops}}{\text{s}} = 17.5\ \text{Gflop/s}$

NHR FAU

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

The Execution-Cache-Memory (ECM) model

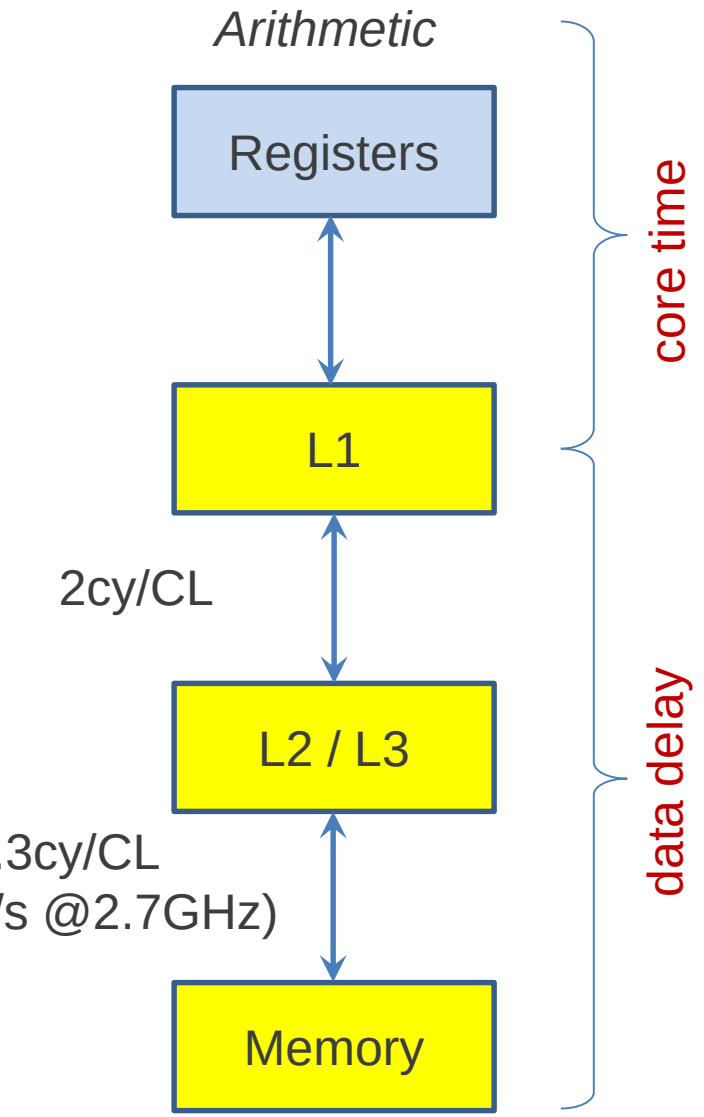
A Refinement: The Execution-Cache-Memory (ECM) model

Insights

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

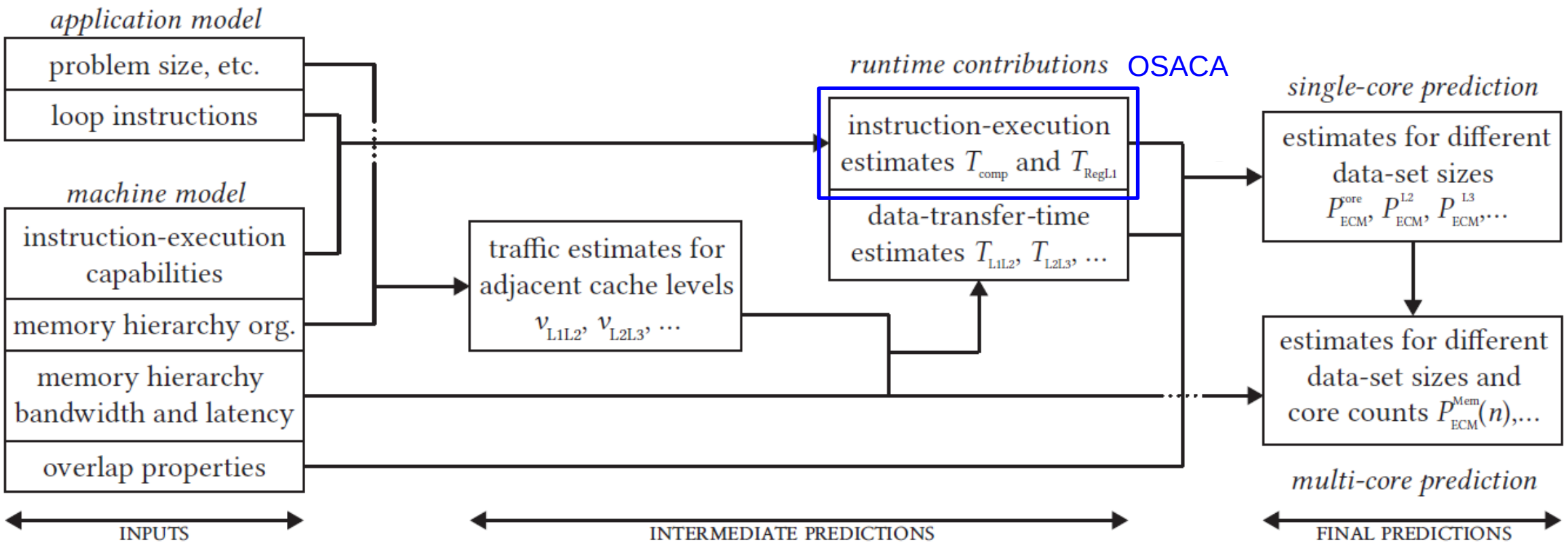
Input

- Machine model
  - [non-]overlap assumptions
  - bandwidths between all hierarchy levels (theoretical or measured)
  - Cache organization (inclusive / exclusive / WB / WT / victim)
- Data transfer model in all memory levels (Machine ←→ Application)
- In-core execution model (Machine ←→ Application)



ECM modeling workflow

J. Hofmann, C. L. Alappat, G. H., D. Fey, G. Wellein, DOI: 10.14529/jsfi200204

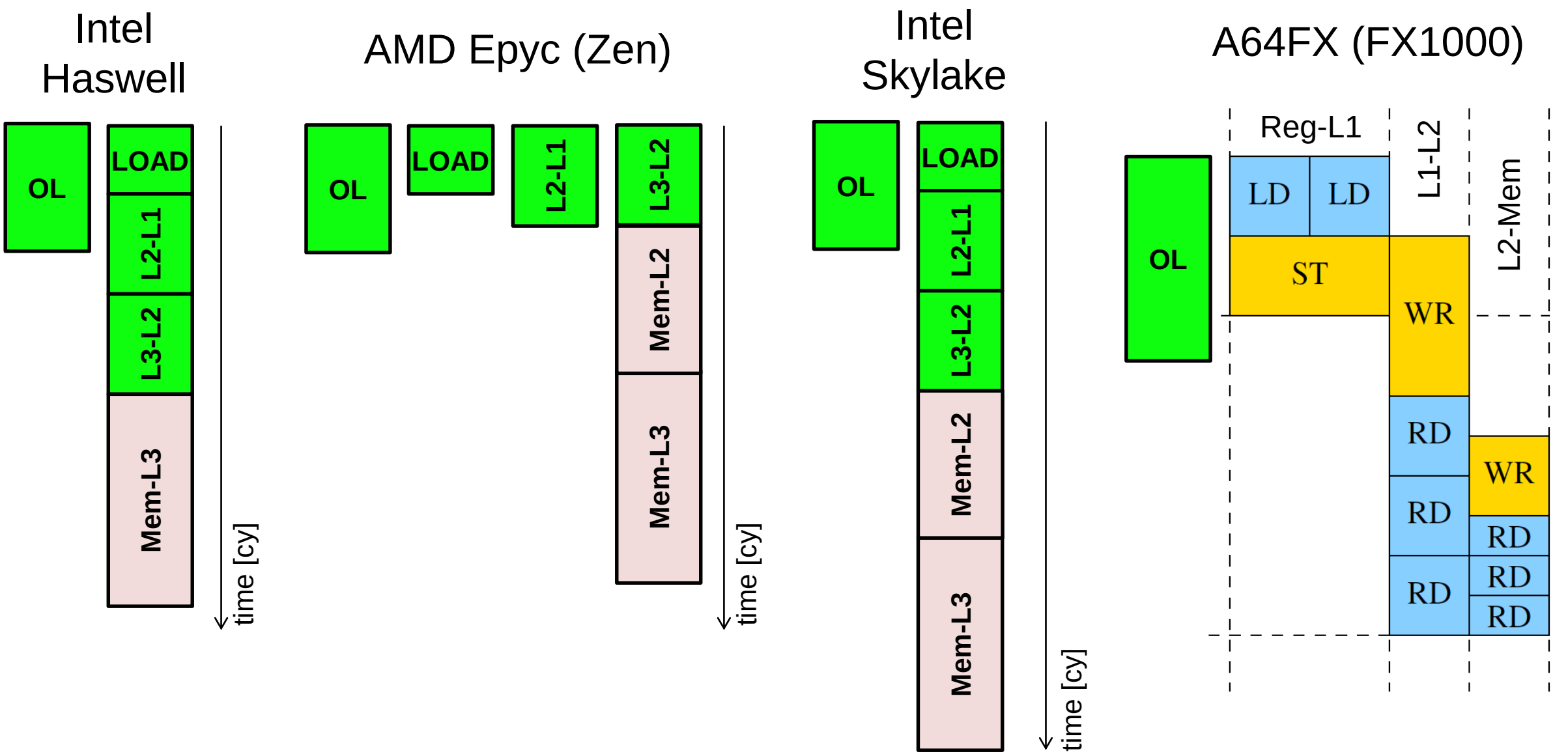


Automating this workflow is possible in some cases:

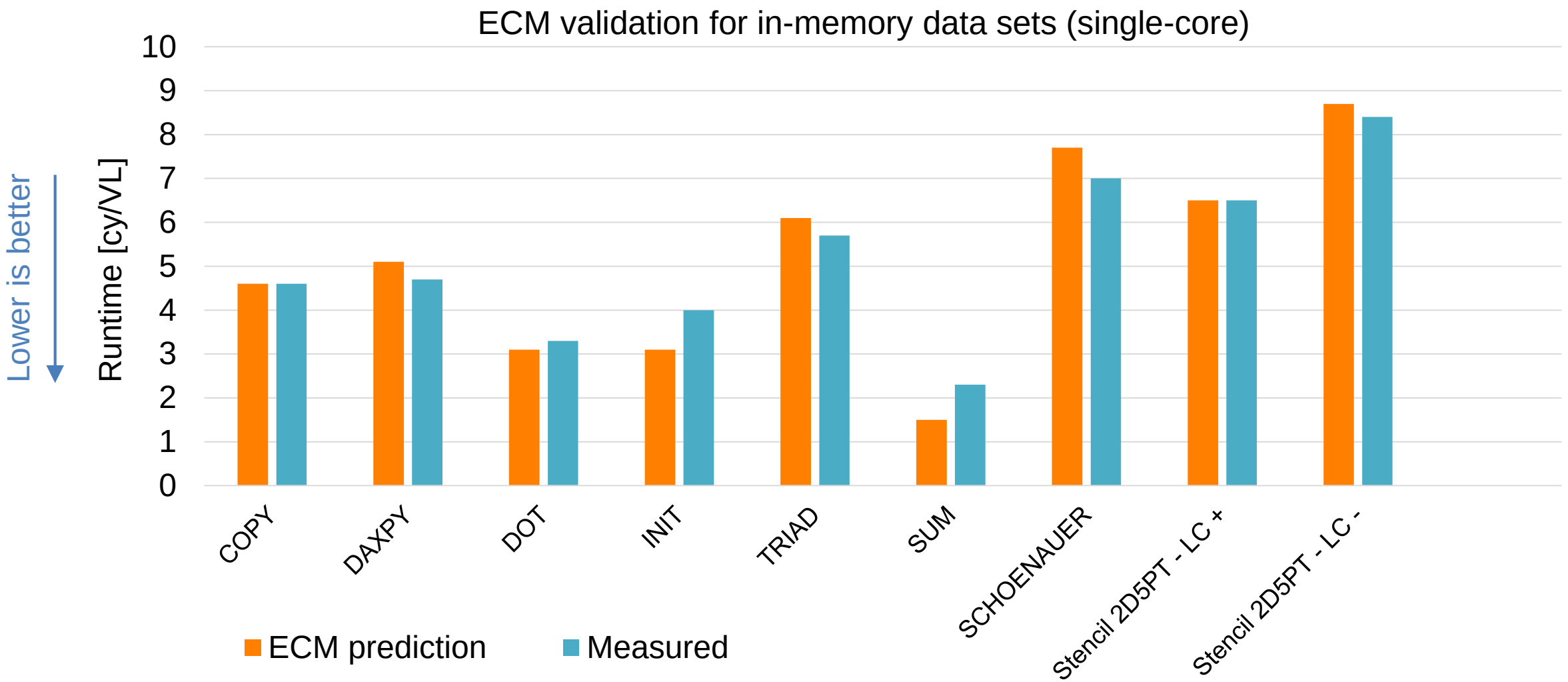
J. Hammer, J. Eitzinger, G. H., G. Wellein, DOI: 10.1007/978-3-319-56702-0_1 (Kerncraft)

J. Laukemann, J. Hammer, G. H., G. Wellein, DOI: 10.1109/PMBS49563.2019.00006 (OSACA)

Overlap assumptions

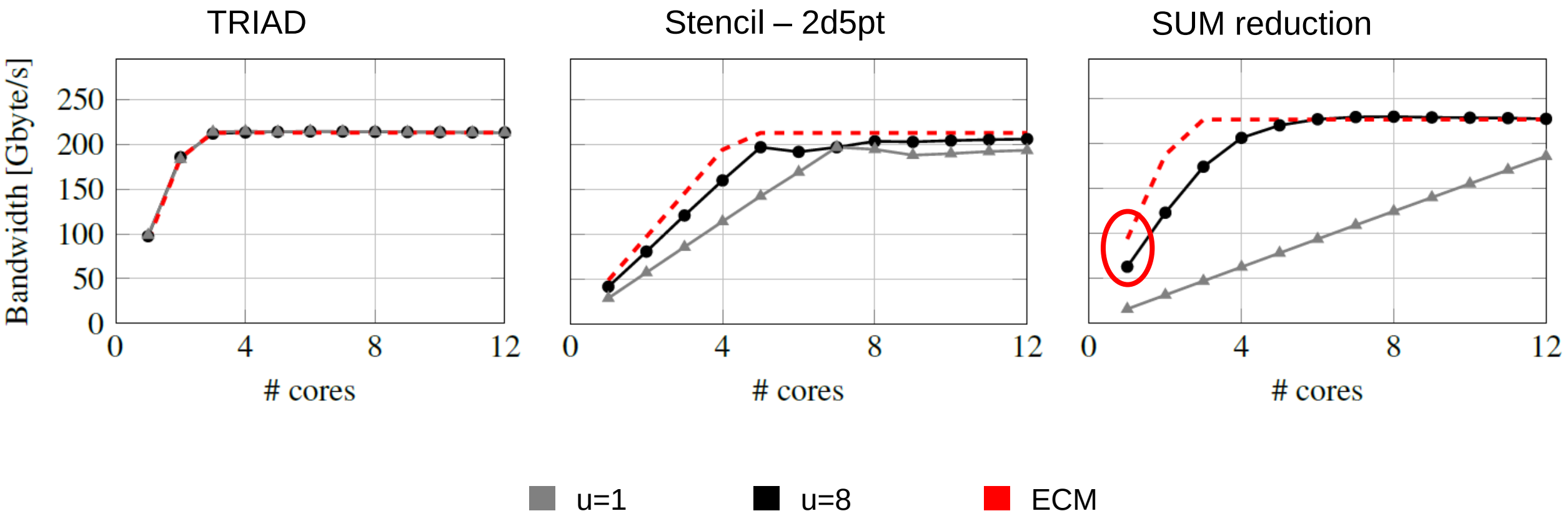


Model validation (FX1000, large pages)



Multicore (in-memory data set) w/ unrolling

C. Alappat et al., DOI: 10.1002/cpe.6512



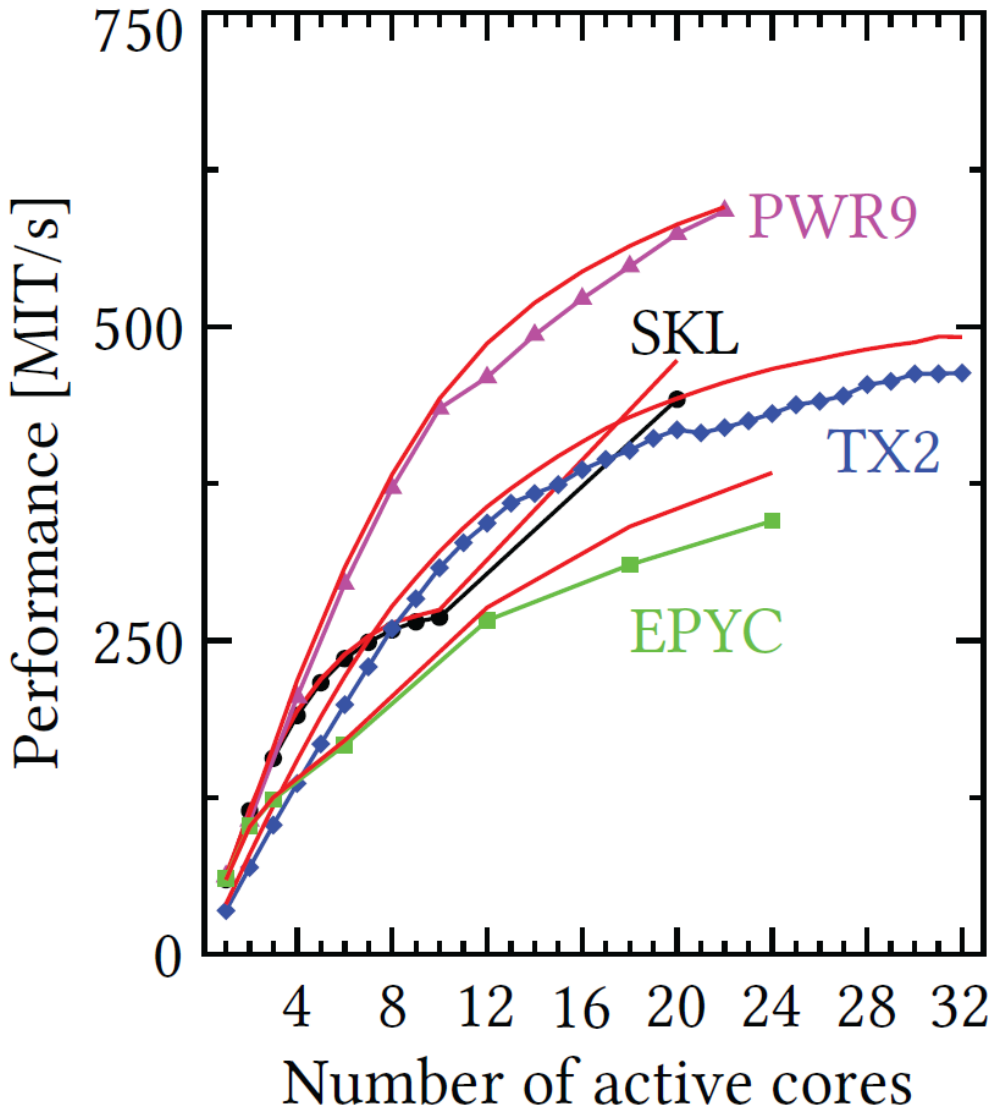
Sufficient unrolling is crucial (but sometimes it's not enough)

Does it work for “real” code, too?

- Preconditioned matrix- free conjugate-gradient solver
- Four systems
  - IBM Power9
  - Cavium/Marvell TX2
  - AMD Naples
  - Intel Skylake

- Yes it does.

J. Hofmann et al., DOI: 10.14529/jsfi200204



NHR FAU

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

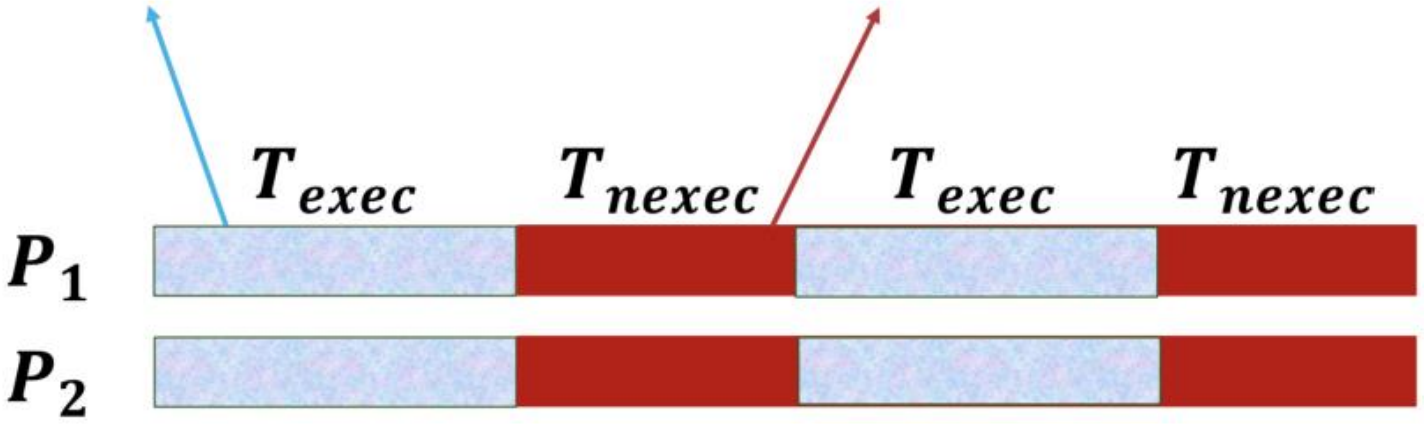
The pitfalls of composite models in the highly parallel case

Composite analytic models

Plausible assumption: $T = T_{exec} + T_{nexec}$

e.g.,
$\max(T_{BW}, T_{flops})$

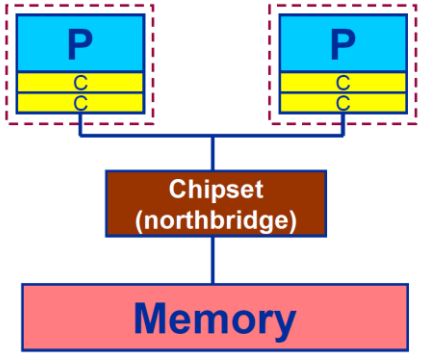
e.g.,
$\lambda + \frac{V}{B}$



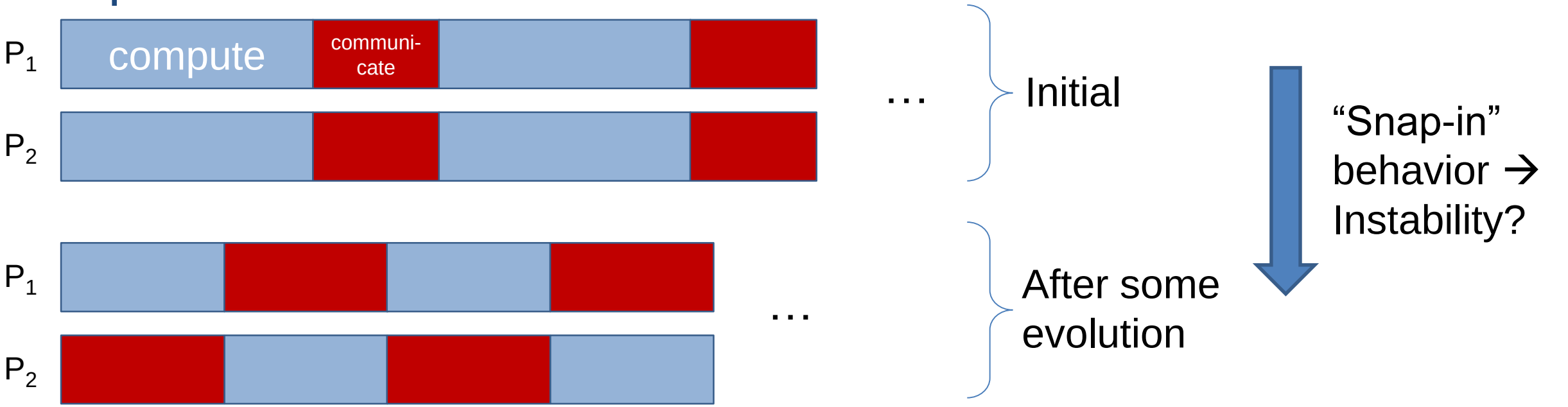
In practice, $T \neq T_{exec} + T_{nexec}$ and it can go in either direction

Initial observation

Two-socket single-core Pentium IV “Prescott” node (2004-ish)



MPI-parallel Lattice-Boltzmann solver timeline view:

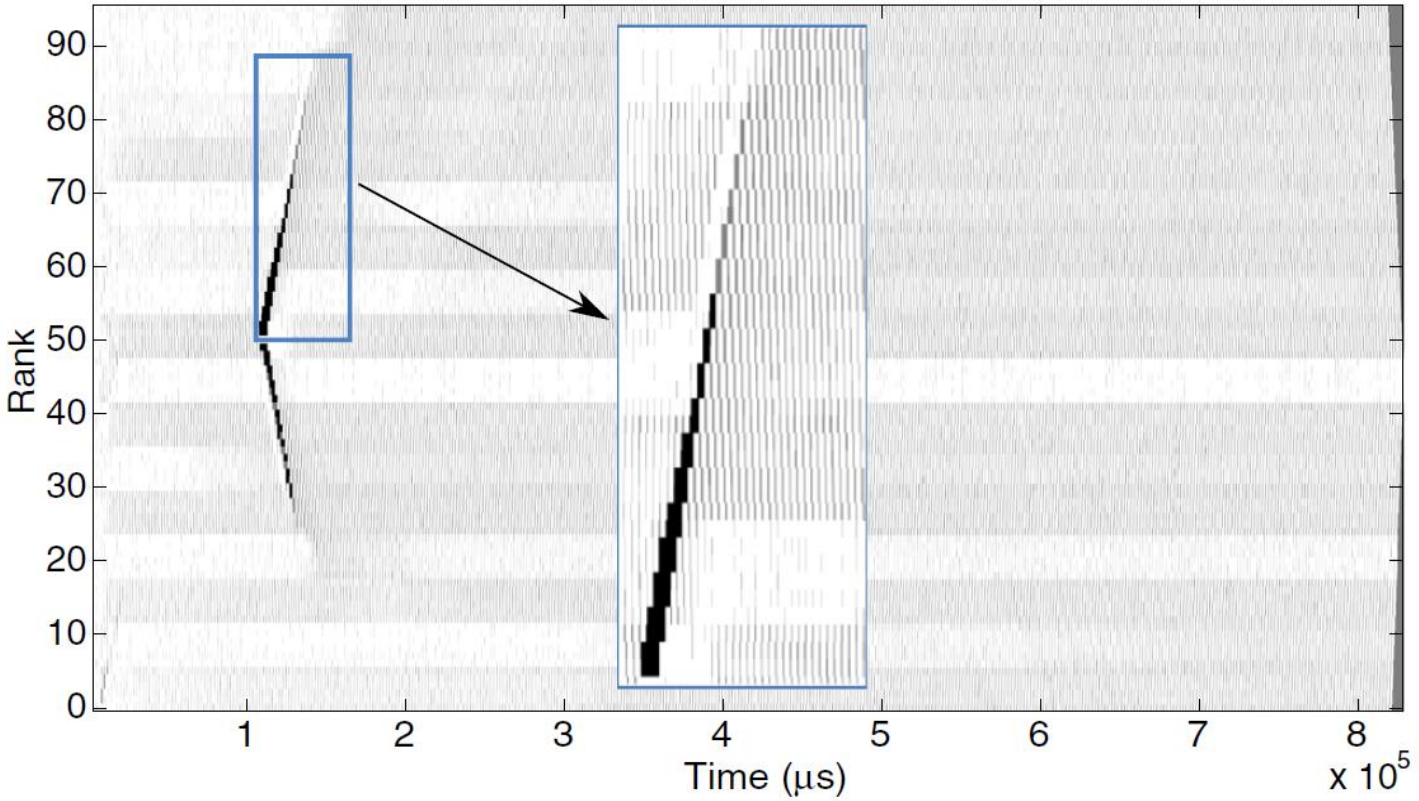


Markidis et al. (2015)

Simulator-based analysis

Idle waves perceived as “damped linear waves”

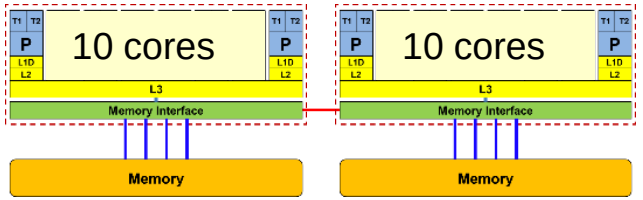
Classical wave equation postulated for continuum description

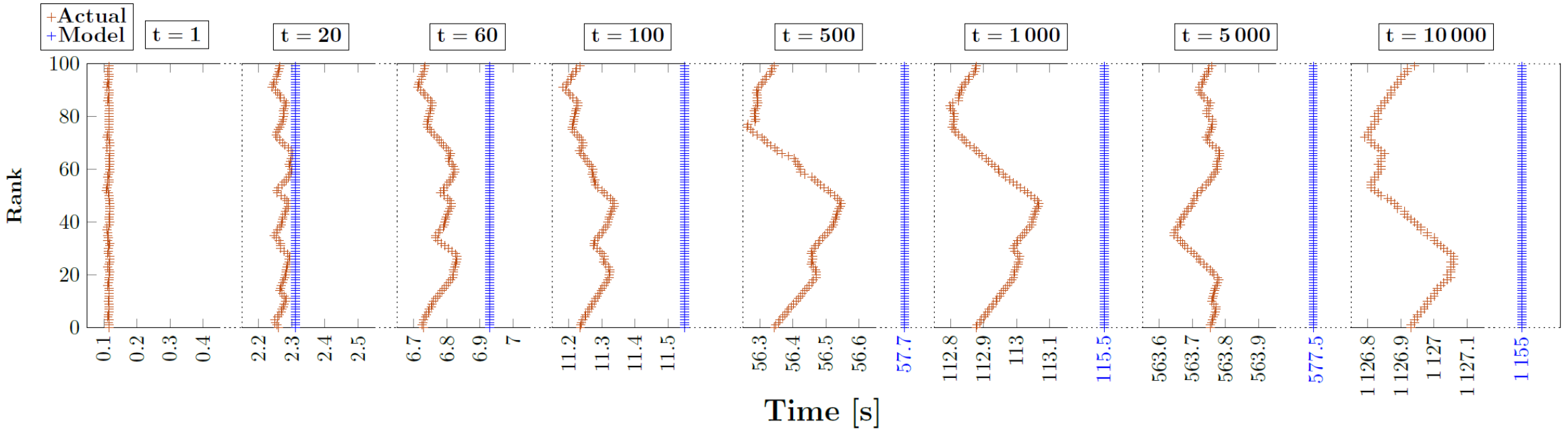


S. Markidis et al.: *Idle waves in high-performance computing*. Phys. Rev. E **91**(1), 013306 (2015).
DOI: 10.1103/PhysRevE.91.013306

A more modern platform

RRZE “Emmy” cluster, 10 cores/socket,
2 sockets/node





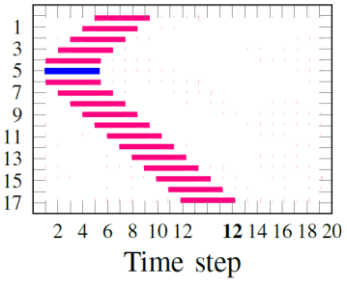
→ Spontaneous symmetry breaking, “computational wave”

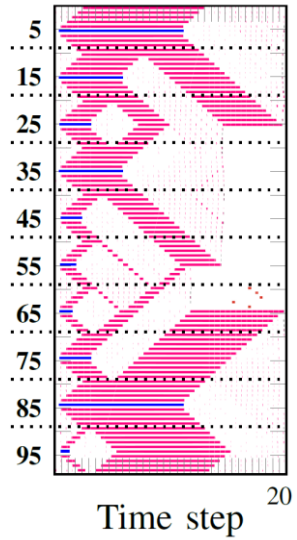
Why? Under which conditions?

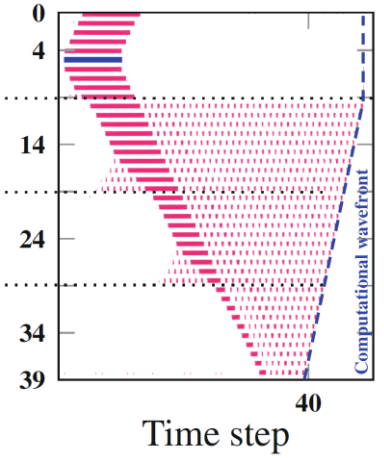
Research questions

Setting: MPI- or hybrid-parallel bulk-synchronous barrier-free programs

- How do “disturbances” propagate?
  - Injected idle periods
  - Dependence on communication characteristics
- How do idle waves interact with each other, with noise, and with the hardware?
  - Idle wave decay
    (noise-induced, bottleneck-induced, topology-induced)
- How do computational waves form? Instabilities?
  - Core-bound vs. memory-bound
  - Amplitude of the computational wave?
- Continuum description?



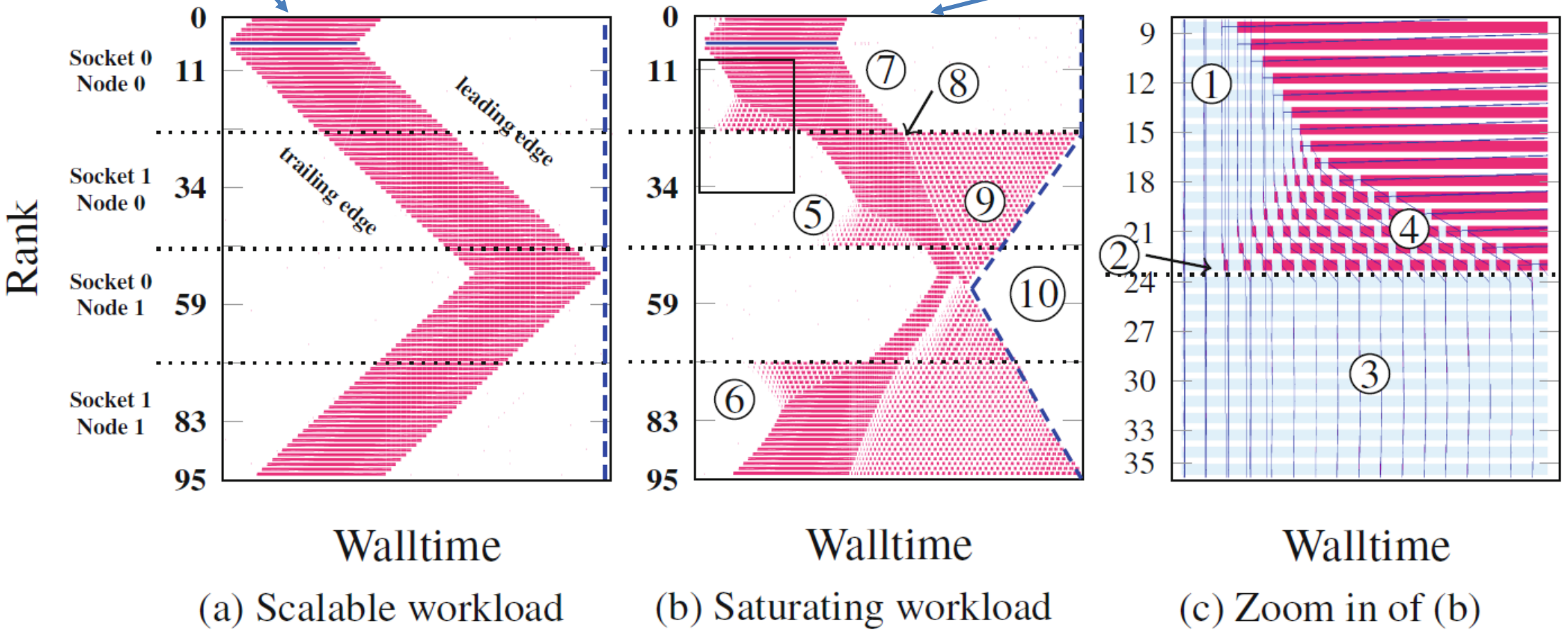




Idle wave propagation and bottleneck-induced decay

Analytical model for idle wave speed with scalalble workload:
DOI: 10.1007/978-3-030-78713-4_19

Decay even on silent system:
DOI: 10.1007/978-3-030-50743-5_20

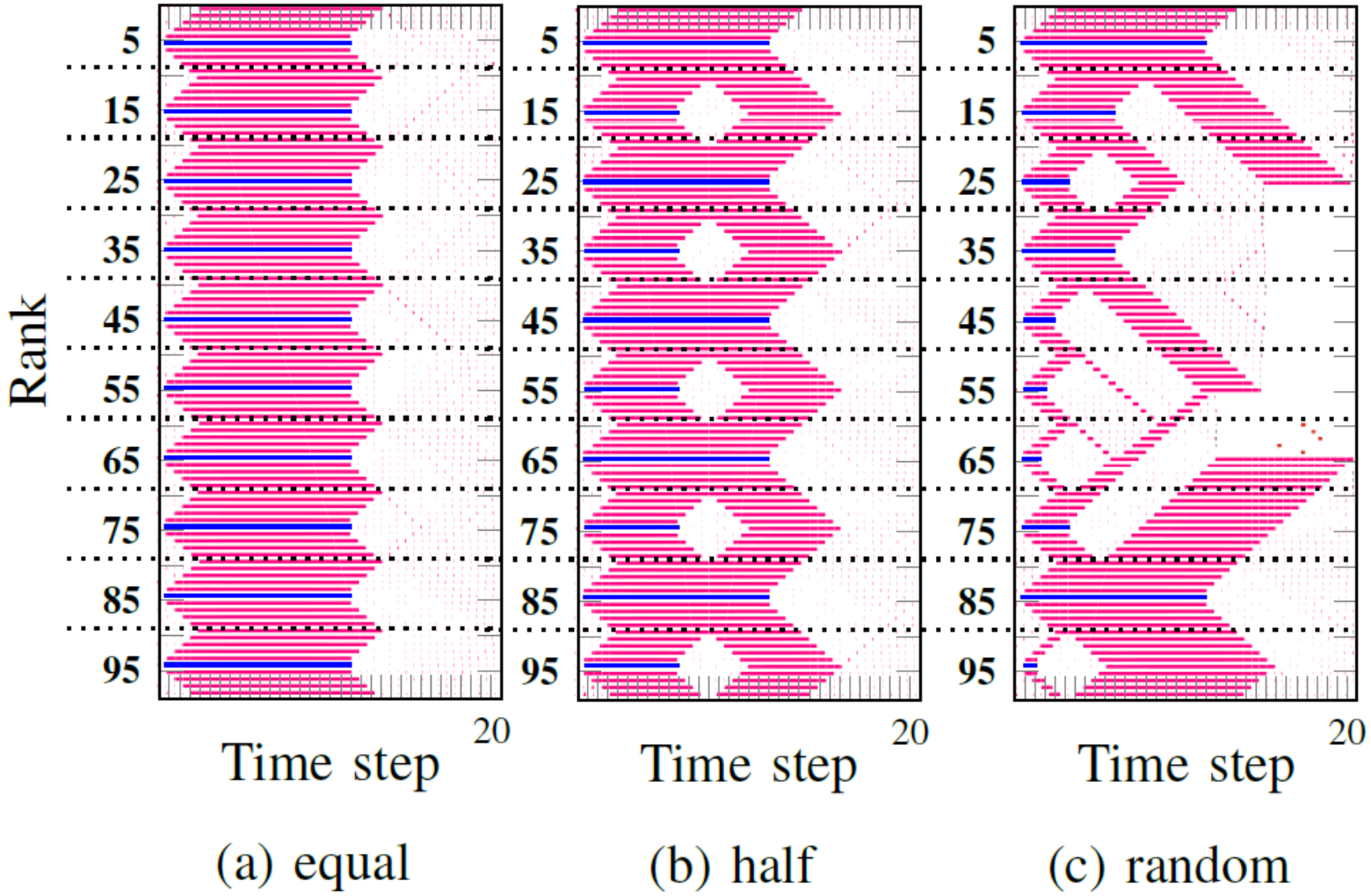


(a) Scalable workload (b) Saturating workload (c) Zoom in of (b)

Idle waves interact nonlinearly

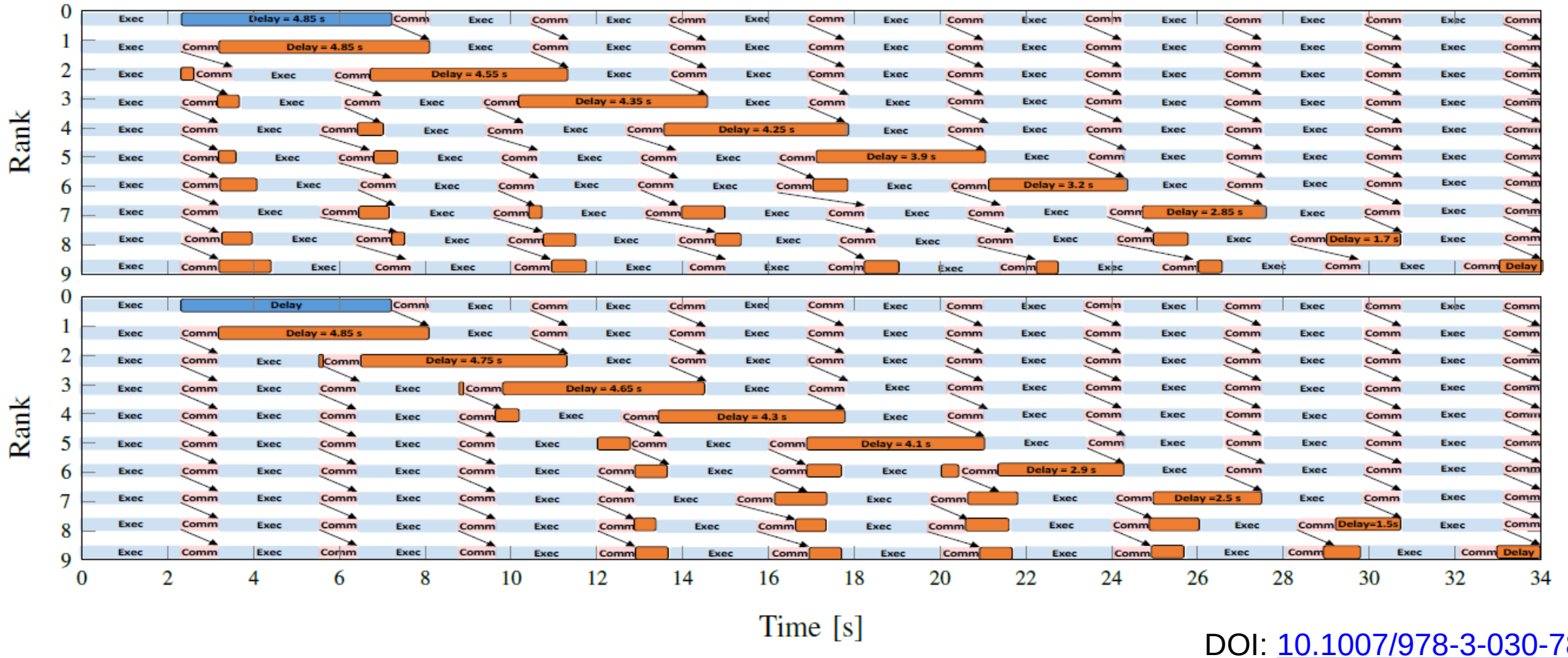
DOI: 10.1109/CLUSTER.2019.8890995

- A wave-like description cannot be based on a linear model
- Basis for noise-induced decay of idle waves



Noise-induced idle wave decay

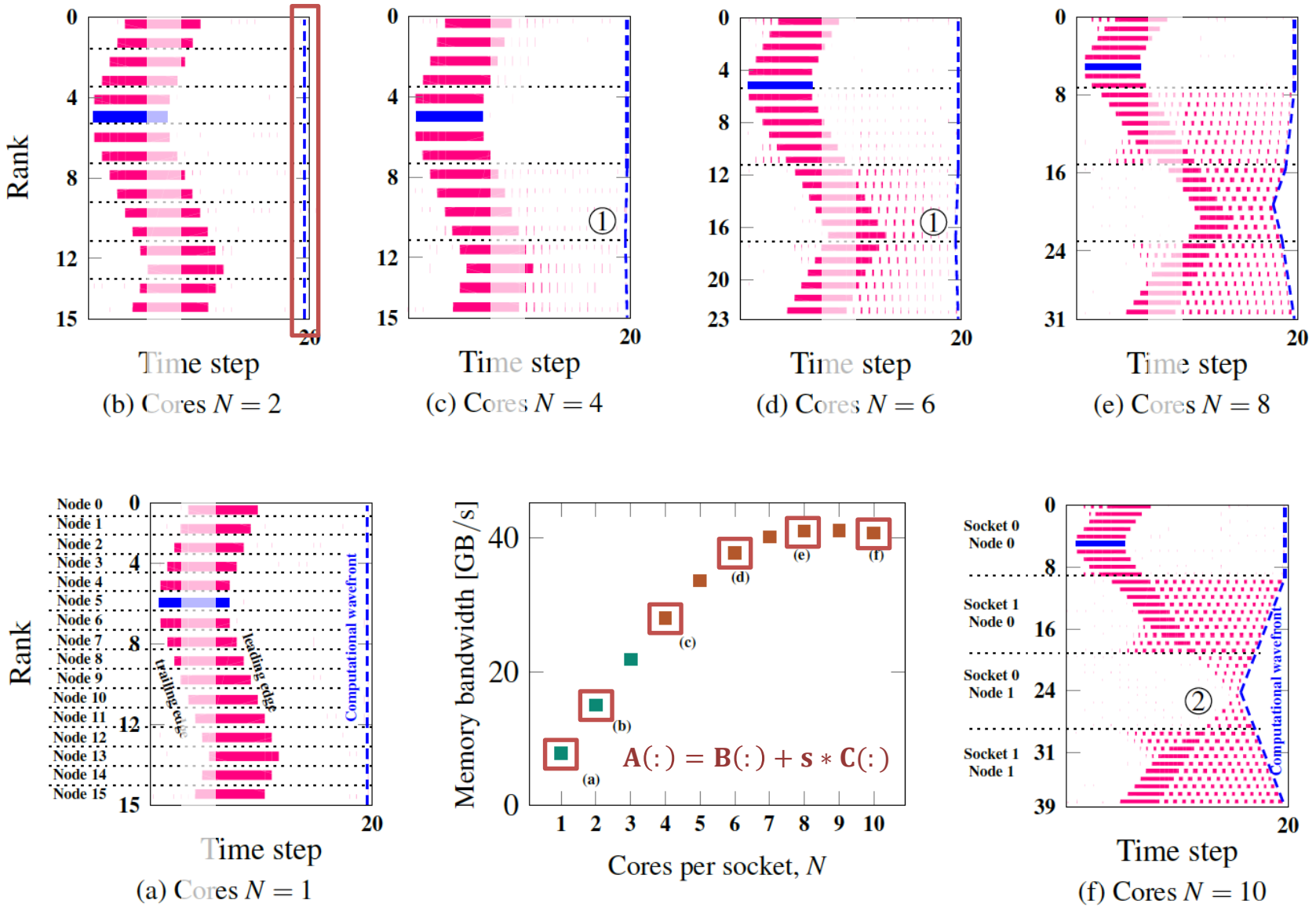
- System or application noise “eats away” on the idle wave
- Statistical details do not matter (only integrated noise power)



DOI: 10.1007/978-3-030-78713-4_19

Formation of computational wavefronts from idle waves

- 2-socket 10-core
- No decay if in non-saturated regime
- Faster decay with stronger saturation



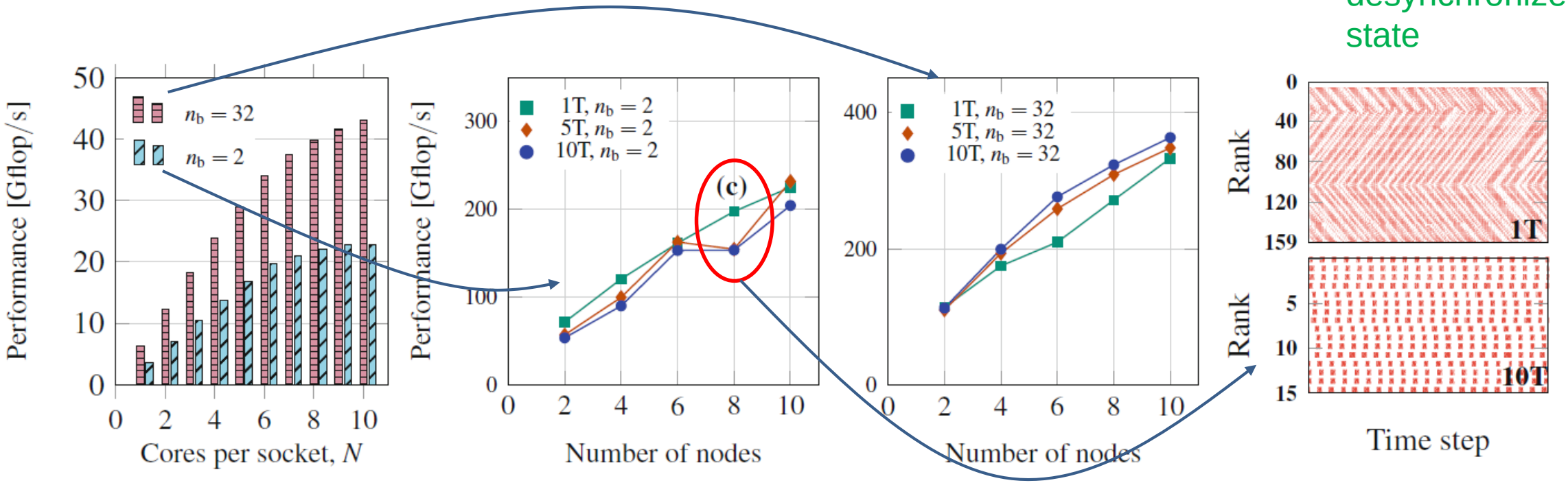
(a) Cores $N = 1$ (b) Cores $N = 2$ (c) Cores $N = 4$ (d) Cores $N = 6$ (e) Cores $N = 8$ (f) Cores $N = 10$

Application: Chebyshev Filter Diagonalization (ChebFD)

DOI: 10.1007/978-3-030-50743-5_20

- Computes inner eigenvalues of a large sparse matrix
- Blocking optimization: M. Kreutzer, G. H., D. Ernst, H. Fehske, A.R. Bishop, G. Wellein, DOI: 10.1007/978-3-319-92040-5_17
- MPI+OpenMP hybrid, topological insulator matrix, Emmy@RRZE

Computes faster in desynchronized state



(a) Single socket performance (b) MPI only vs. hybrid communication, $n_b = 2$, $n_b = 32$ (c) Timeline visulization

Current results

- Instability of bulk-synchronous barrier-free programs is bound to the presence of a resource bottleneck
- Desynchronized bottlenecked programs can exhibit automatic communication/execution overlap via formation of computational waves
- Idle waves can be absorbed by fine-grained system noise, and the mechanism behind this is well understood
- Idle waves can decay via topological noise caused by inhomogeneous communication characteristics
- Proof that noise statistics is largely irrelevant for idle wave decay rate
- Analytic model for idle wave velocity w.r.t. communication topology and characteristics
- Experimental evidence that MPI collectives can be transparent to idle waves

Future directions

- Development of a comprehensive, bottleneck-aware simulator framework for message-passing programs
- Analytic description of decaying wave for bottleneck-triggered decay
- Bottlenecks other than memory bandwidth
- Analytic understanding of computational wave amplitude w.r.t. communication characteristics and bottleneck saturation
- Idle wave phenomena in irregular programs
- Physical model for coupled processes (Kuramoto-like)

$$\dot{\theta}_i = \omega_i + \alpha \sum_j T_{ij} V(\theta_j - \theta_i)$$

- Continuum description of parallel system as a nonlinear (dissipative?) medium

NHR FAU

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Thank you.