Data Transfer Requirement Analysis with Bandwidth Curves

PROPER 2013, Aachen, Germany
August 27, 2013

Josef Weidendorfer

Chair I-10: Computer Architecture
Technische Universität München

The Complexity of Performance Analysis

Possible reasons for application slowness

- bad instruction selection by compiler (data conflicts, no SIMD)
- bad memory access locality (lots of cache misses)
- synchronization overhead, frequent accesses to shared data
- load imbalance, high communication requirements

The Complexity of Performance Analysis

Possible reasons for application slowness

The Perfect Tool

- observes a program run to find all potential bottlenecks at once
- shows how much better one can get
- provides relation to source/data structures
- uses visualizations hinting at needed optimization strategy

The Complexity of Performance Analysis

Possible reasons for application slowness

The Perfect Tool … does not exist

- measurements show symptoms of overlapping effects
- tradeoff: amount of measurement data vs. measurement overhead

Wouldn‘t it be nice to have a complementing analysis method that

- allows for decoupled analysis of resource restrictions / effects ?
- allows for detailed measurements without overhead?

Both is possible with artifical machine models and simulation

Novel Performance Analysis Approach

Check influence of resource limitations separately

- how fast can a program run given only one limitation?
- what demands does this induce for other resources?

Benefits

- abstract, platform-independent program characteristics
- severity of a resource restriction: compare with detected demand
- upper performance bounds due to single resource restrictions

Outline

- A novel performance analysis approach for
  Analysis of the influence of bandwidth restrictions
- Machine/Execution Model
- Bandwidth Curves
- Tool Prototype
- Example using a 2D Jacobi solver
- Conclusion / Future Work

Analysis of the Influence of Bandwidth Restrictions

Bandwidth restrictions are significant

- lots of applications are memory-bound
- getting worse on multi-core: cores compete for shared connections
- slow connection to accelerators
- slow connections between nodes in larger systems

Analysis of the Influence of Bandwidth Restrictions

Bandwidth restrictions are significant

Analysis Goals

- are bandwidth restrictions a bottleneck for a program?
- if so, how high is the influence?
- how much faster can we get with reduced bandwidth demands?
- useful visualization: Bandwidth Curve

Analysis of the Influence of Bandwidth Restrictions

Bandwidth restrictions are significant

Analysis Goals

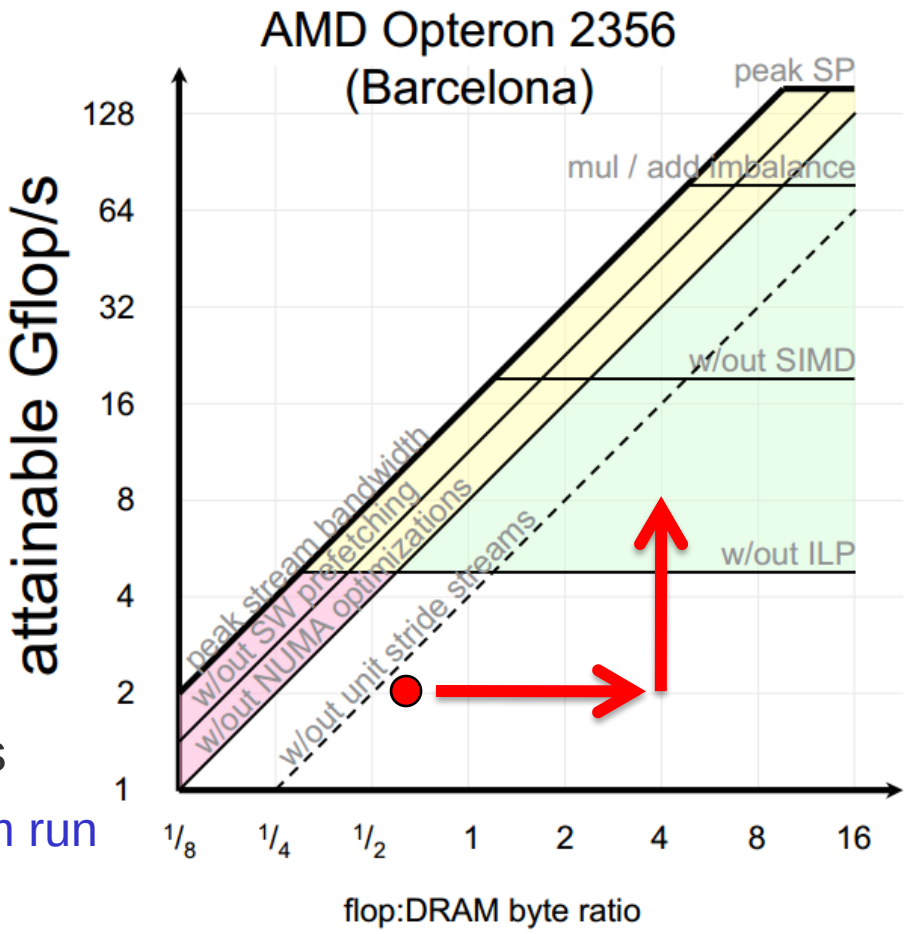
Realization

- feed events from a program execution into a machine model
- assume performance is given by FLOP *or* instruction throughput
- observe bandwidth demands at connections in machine model
- compare with real bandwidth restrictions to get slowdown

Comparison with the Roofline model

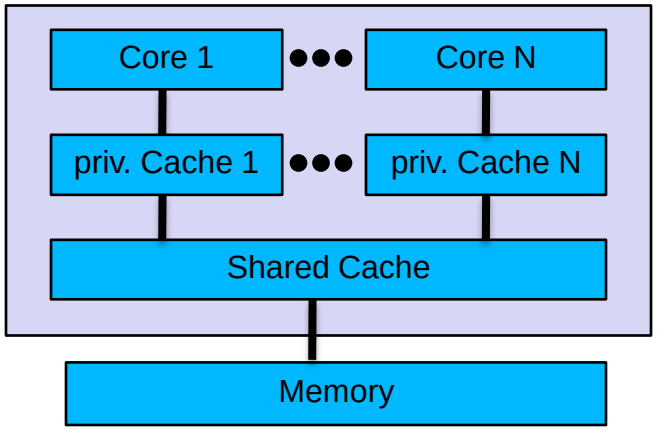
- arithmetic intensity:
  FLOPs executed per
  byte moved from/to memory
- Roofline model:
  diagram which allows to see
  the influence of arithmetic
  intensity of a kernel on
  achievable performance

- our approach: fixed limit is FLOP/s
  - bandwidth demand on full program run
  - measured at multiple connections



[Williams, Patterson: The Roofline Model]

Model



Machine model

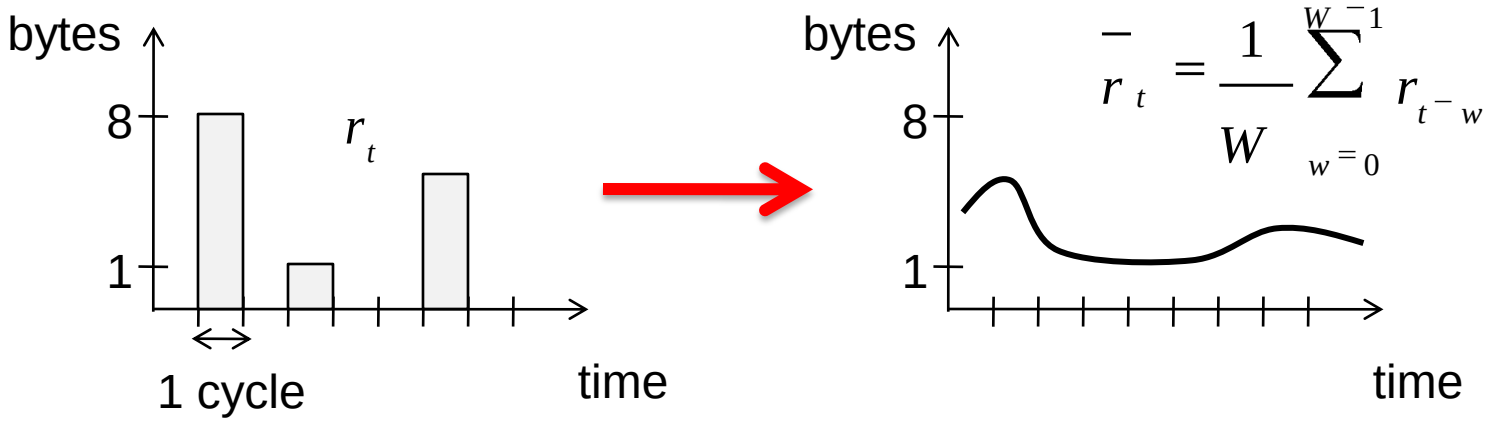
- Multi-core architecture
  - private L1 caches (LRU, write-back, MSI)
  - shared L2 cache (LRU, write-back)
- Performance limitation: e.g. 4 instructions / cycle
- Memory access demands get smoothed

Execution Model

- each thread runs on own core (independence from OS scheduler)
- threads are simulated asynchronously by simulation threads, provide their bandwidth demands of shared connections to others

Smoothing of Bandwidth Demands

- averaging by sliding window of length W
  (assuming out-of-order exec., large buffers, prefetching)

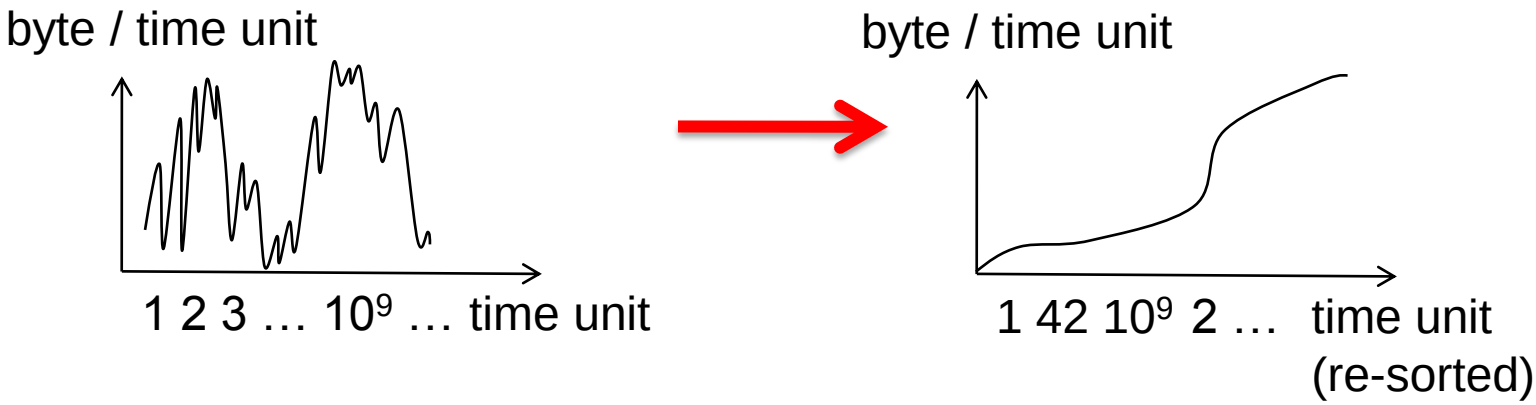


- good choice for W: latency of a memory access (e.g. 200 cycles)

Bandwidth Curves

- storing complete trace of bandwidth demand not feasible
  - too much data (1s real execution relates to billions of cycles)
  - does not provide an overview

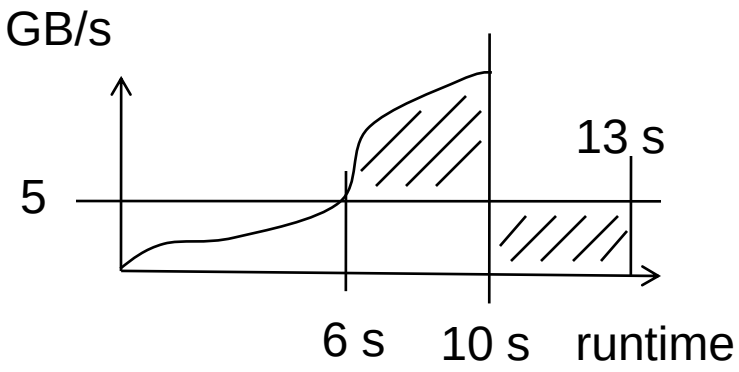
- Sort of time units according to bandwidth demand



Bandwidth Curves

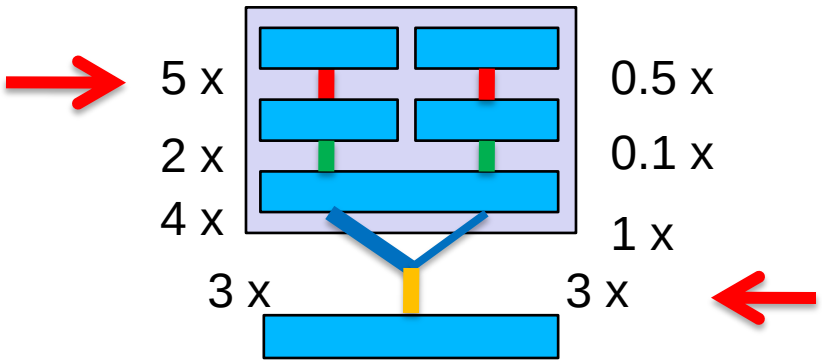
Benefits

- how often does bandwidth demand exceed a given value? (10s : runtime due to instruction throughput limit)
  - example: 4s of runtime bandwidth demand exceeds 5 GB/s
- area under curve represents amount of transfered data
- given a bandwidth restriction, slowdown can be easily derived



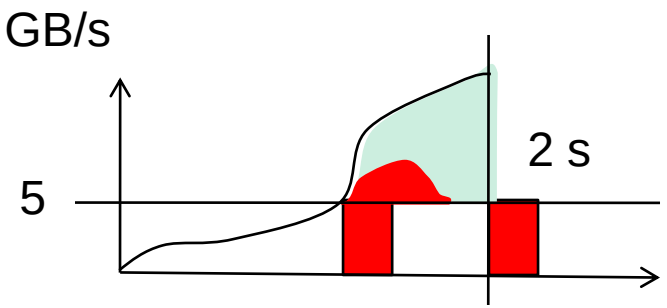
Data Transfers in the Memory Hierarchy

- each thread induces a coupling of bandwidth demands on a sequence of connections between a core and memory
- dominant bandwidth limitation: highest ratio demand vs. real limit
- for shared connections: policy for partitioning of available capacity
- dominance analysis done in each simulated time unit

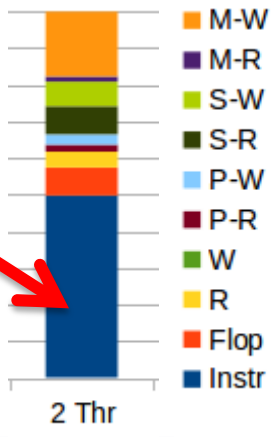


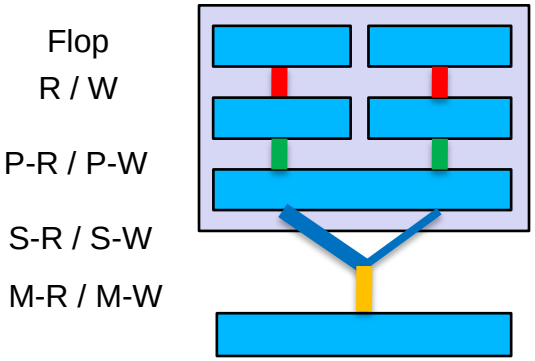
Data Transfers in the Memory Hierarchy

- mark detected dominance in bandwidth curves



- stack bars: slowdown due to dominant connections
  - sum over all threads
  - no dominance

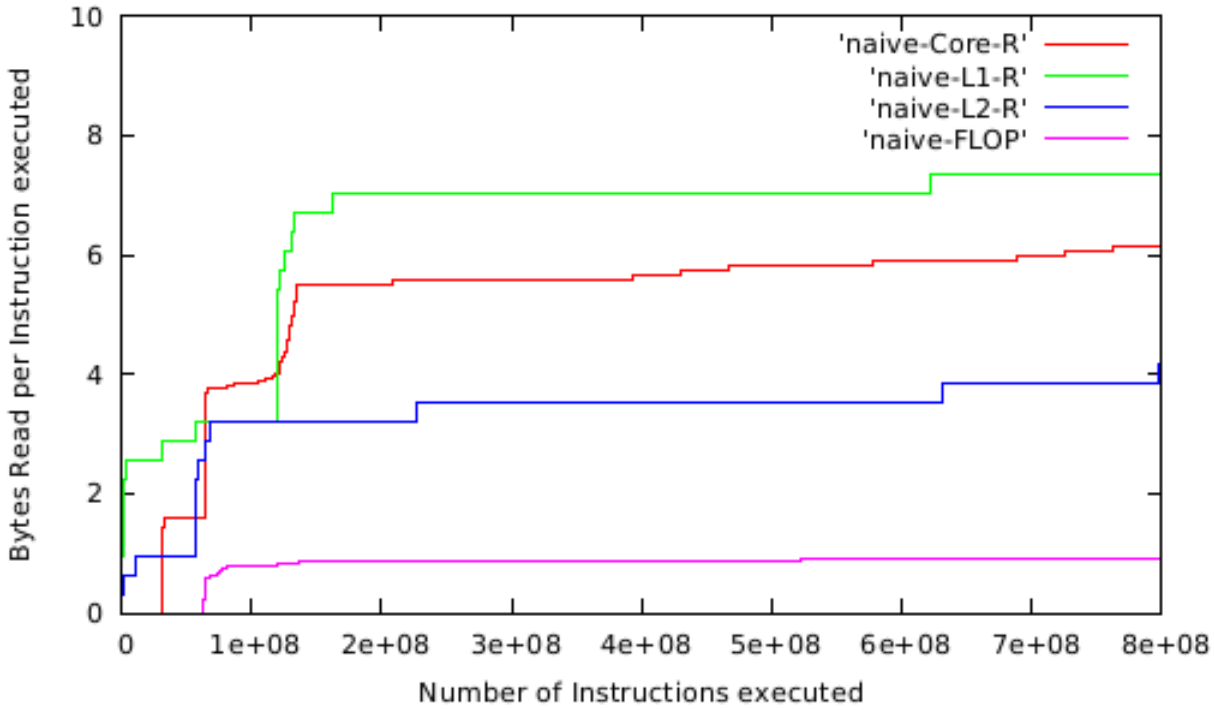


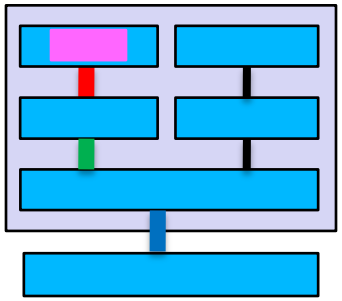


Tool Prototype

- runtime instrumentation for one process in user level
  - enables analysis of compiled binaries (e.g. using OpenMP)
  - execution driven simulation
- uses Intel Pin tool
  - enables parallel simulation of threads (not possible with Valgrind)
- simulates machine model
  - slowdown around 100x
- during system calls assume bandwidth demands of 0
- newly created thread uses new core in the machine model

Example for generated Bandwidth Curves

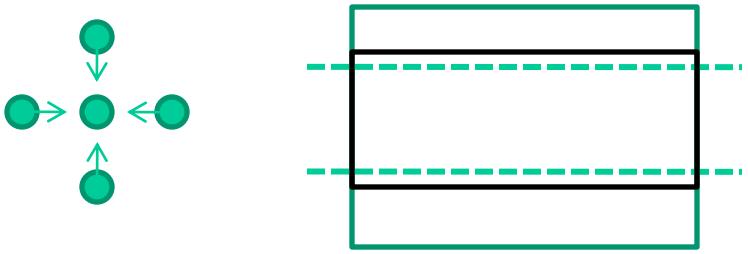




- diagram done with Gnuplot from data generated at program termination (read direction)
- different curves do not show any time relation !

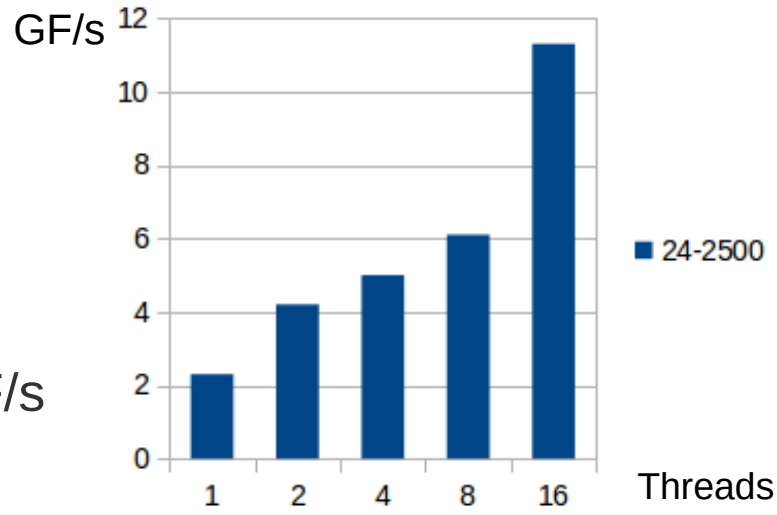
Example: 2D Jacobi Solver

Naive version (OpenMP): 1 sweep/iteration over own partition

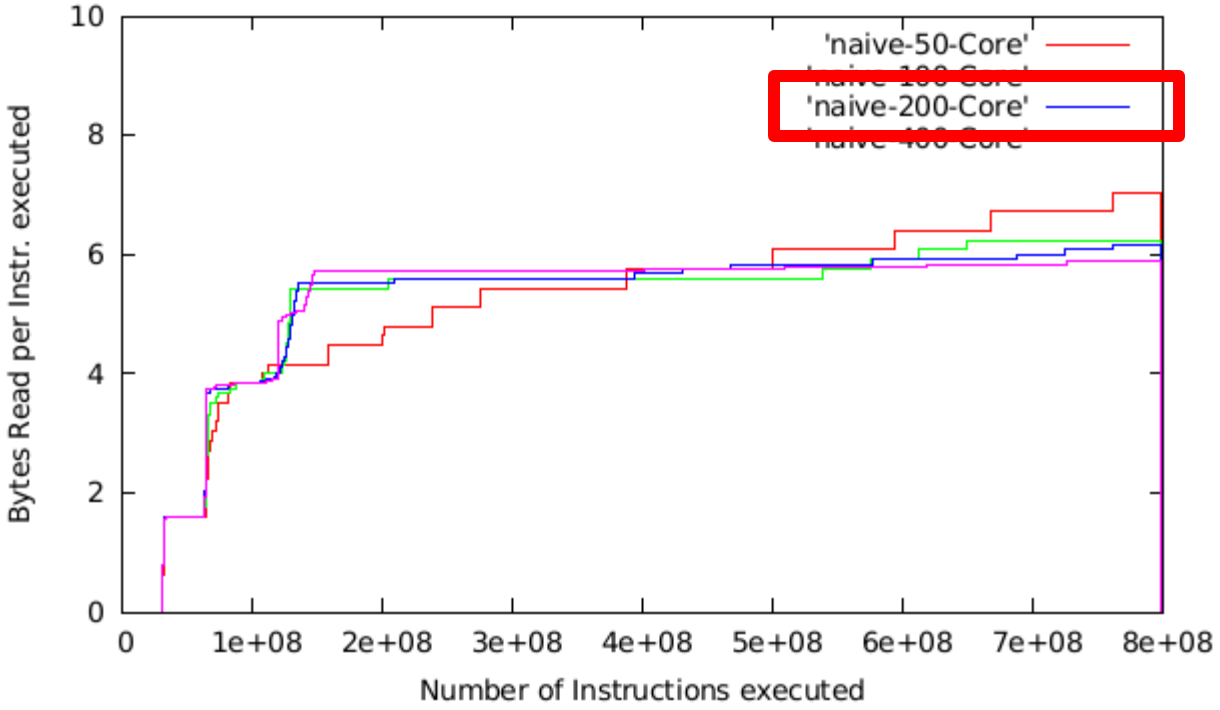


2-socket 8-core Sandy Bridge (2.7GHz)

- matrix size 2500 x 2500, 24 iterations
- peak: 2 Flop (add/mul)
  x 4 (AVX)
  x 16 (cores)
  x 2.7 G/s (frequency) = 345 GF/s



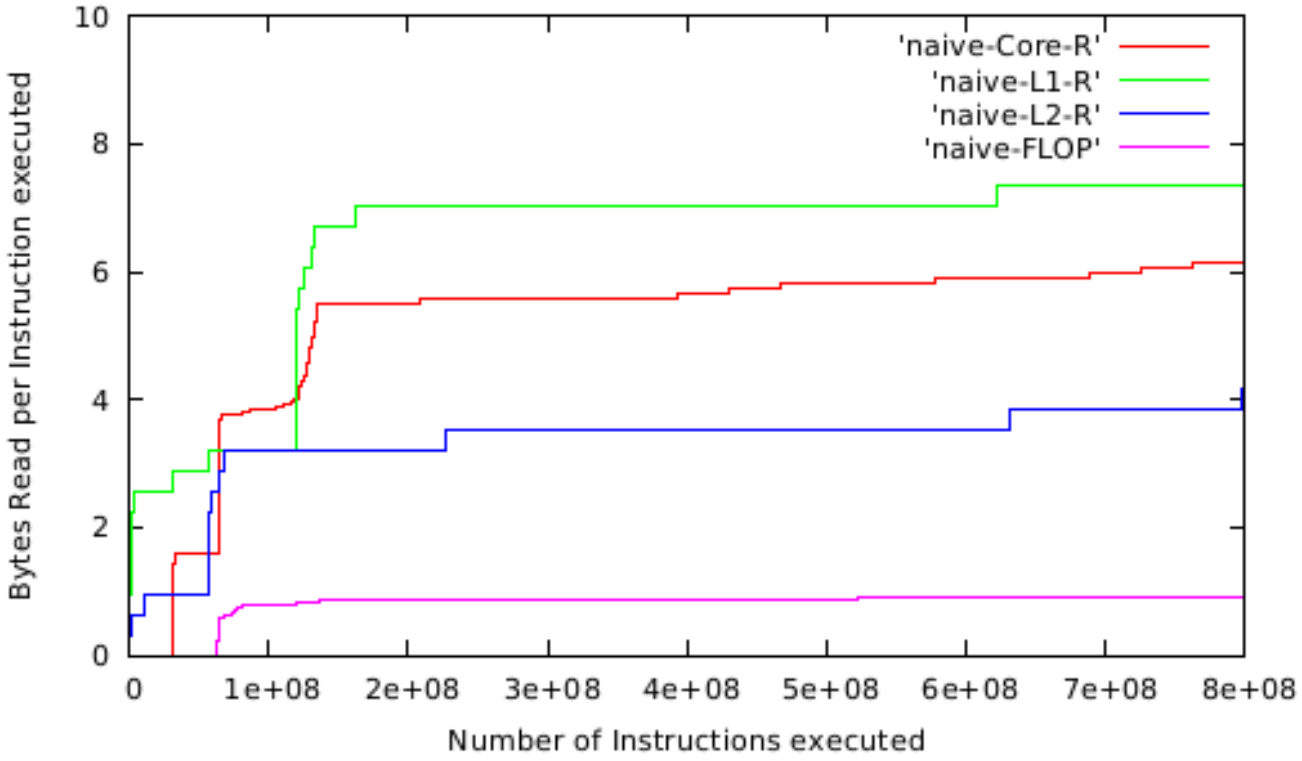
Choice of W for Smoothing

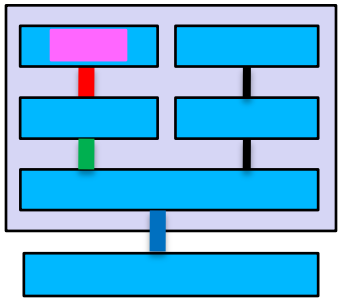


used for following results

Legend: **naive-100-Core**: read bandwidth (L1 ➔ Core), W=100

Bandwidth Curves: 1 thread 2D Jacobi (naive)

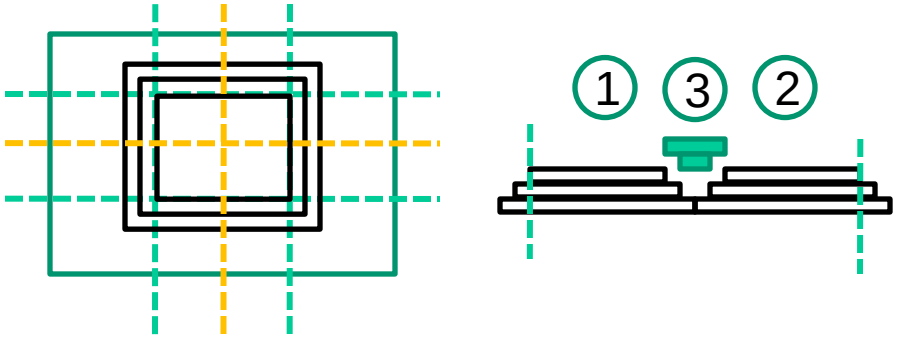




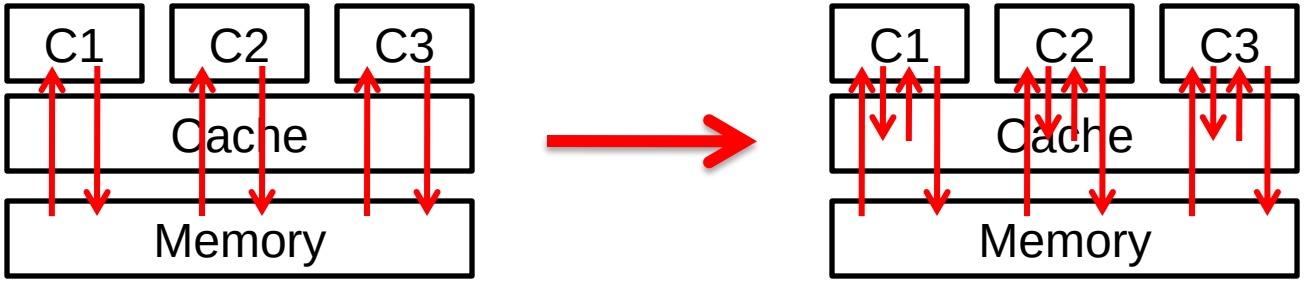
Green line > red line: every cache line must be read before being written

Example: 2D Jacobi Solver (blocked version)

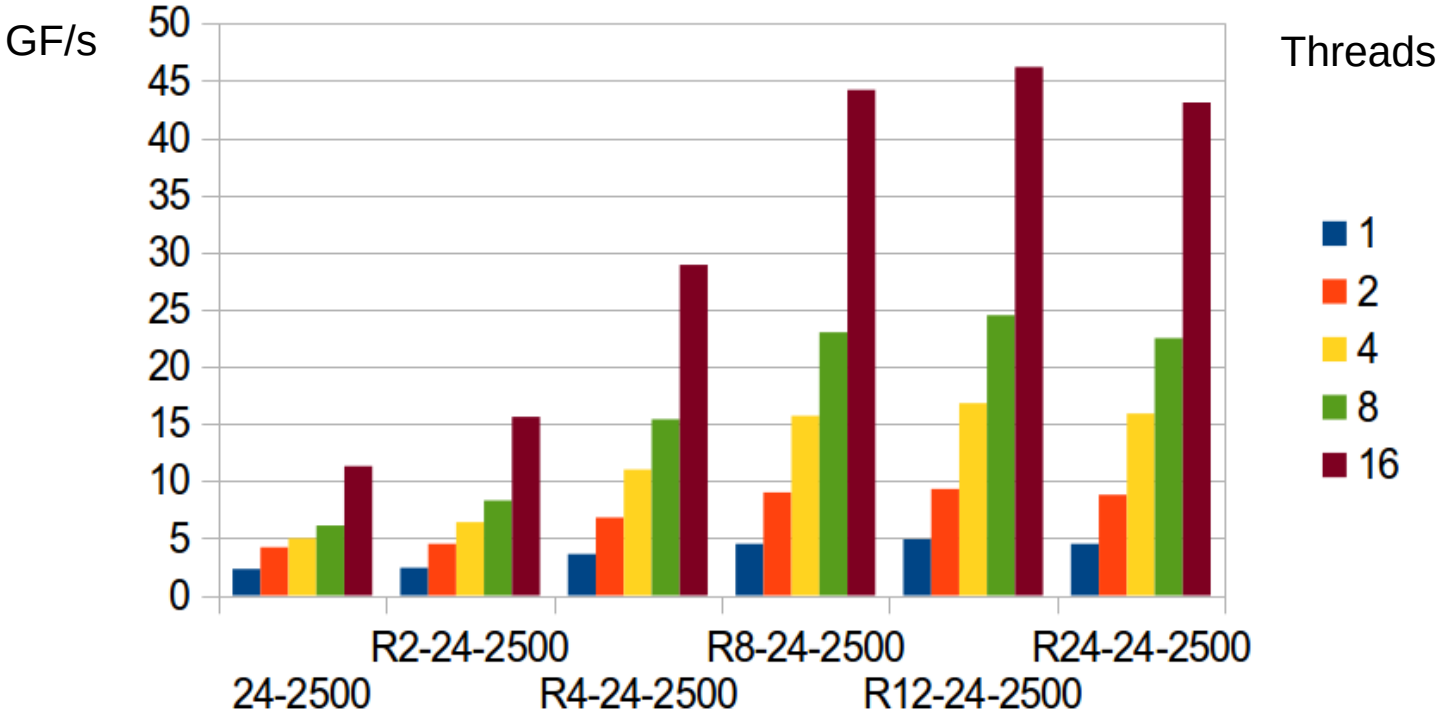
- recursive blocking of multiple iterations (cache oblivious)



- improvement

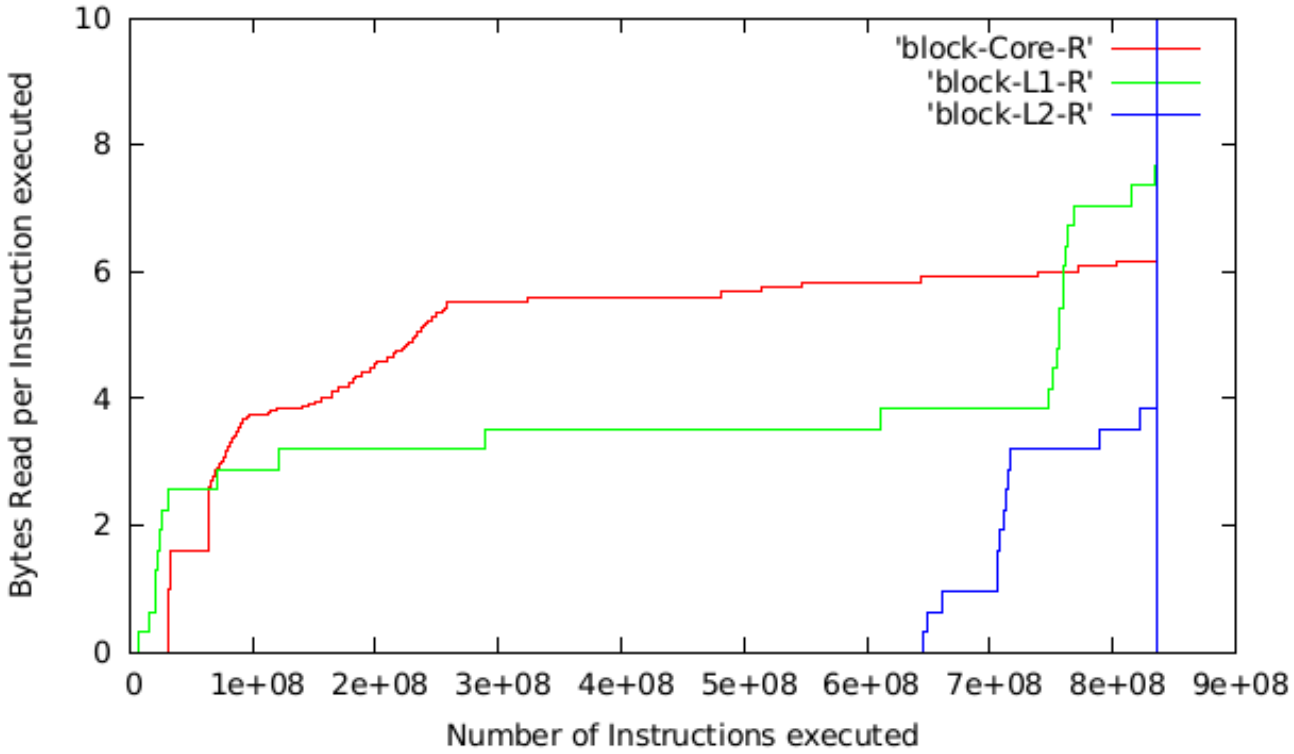


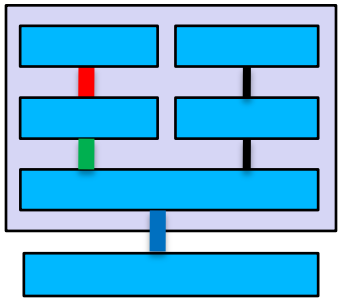
Example: 2D Jacobi Solver (blocked version)



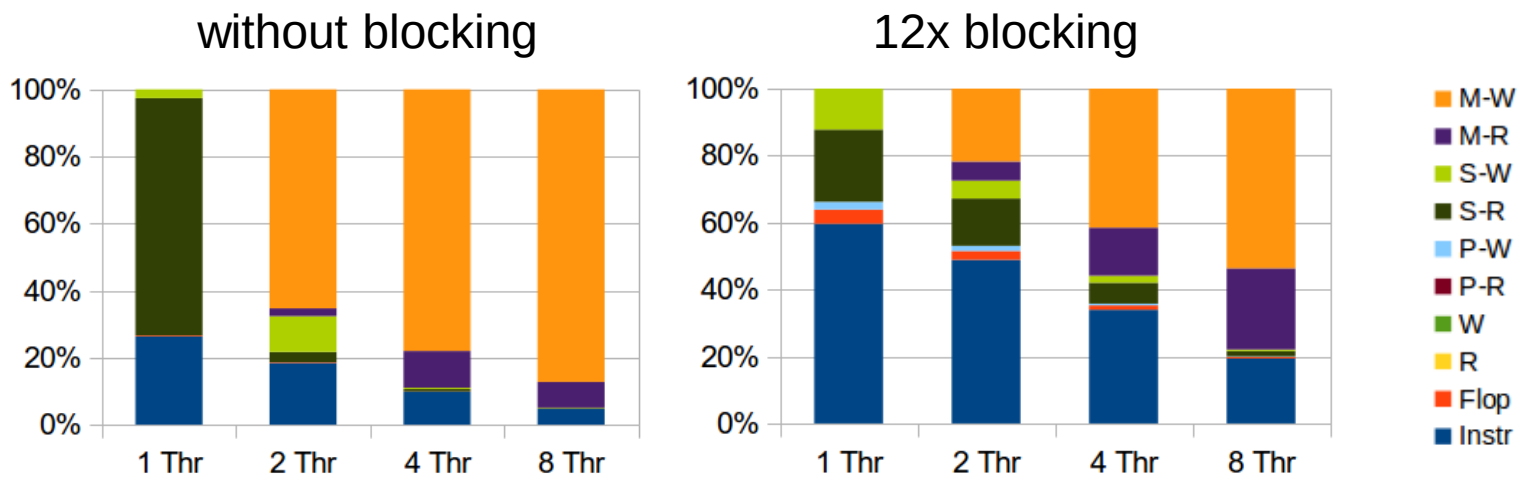
Legend: **R4-24-2500**: 4x blocking, 24 iterations, size 2500x2500

Example: 2D Jacobi Solver (1 thread, 12x blocking)



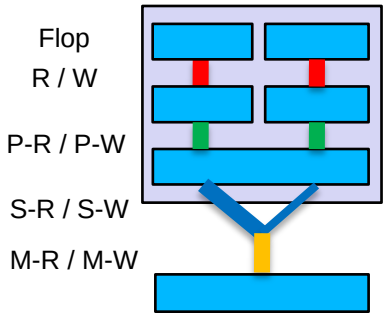


Example: 2D Jacobi Solver (without / with blocking)



Comparison estimated vs. measured runtime

- without blocking: estimation around 30% too low
- 12x blocking: estimation 35% - 40% too low



Conclusion & Outlook

- Artificial machine models with only one resource restriction
  - provide easy to understand architecture-independed characterizations
  - allow to see the severity of a restriction
  - provide lower performance bounds
- future work
  - show relation to source:
    sample memory accesses responsible for a given bandwidth
  - extend machine model for NUMA configurations
  - add GUI

Questions?