An Evaluation of High-Level Mechanistic Core Models

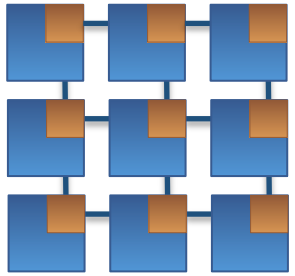
Trevor E. Carlson, Wim Heirman,
Stijn Eyerman, Ibrahim Hur, Lieven Eeckhout

Universiteit Gent
intel
Uppsala Universitet

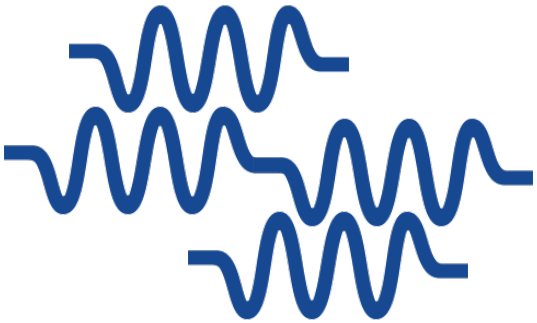
http://www.snipersim.org
Monday, January 19th 2015
HiPEAC Conference 2015, Amsterdam

Design Future Hardware and Software

- Design the processor of tomorrow



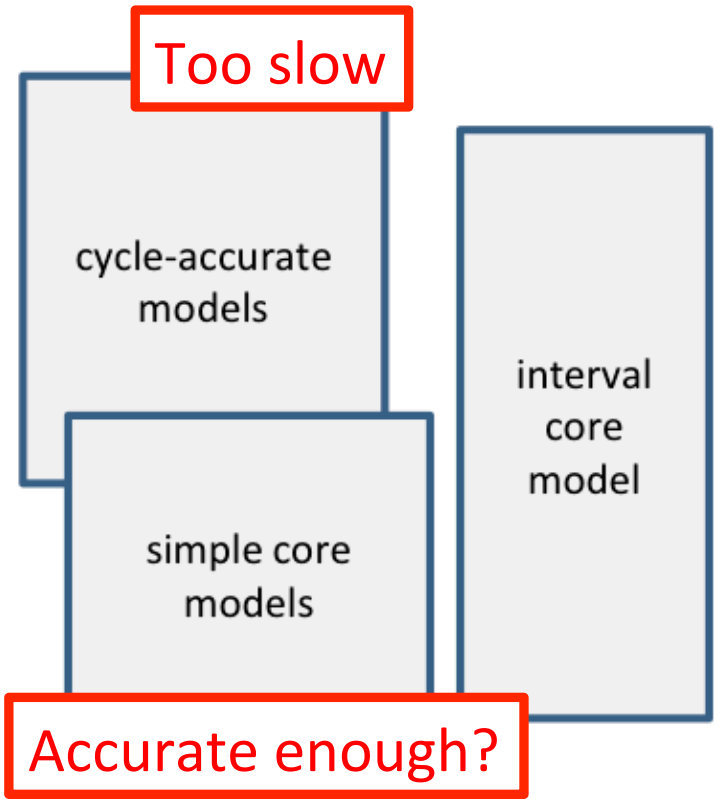
- Optimize next-gen software



- How can we design and evaluate large numbers of design options?
  - Can we do it in a fast way?

Needed Detail Depends on Focus

Component	Single-event time scale	Required sim time
RTL	single clock cycle	millions of cycles
OOO execution		
Core memory ops		
L1 cache access		
LLC access		
Off-socket	microseconds	seconds



One-IPC Modeling

- A simple high-abstraction model, often used in memory hierarchy studies
- Alternative for memory access traces
  - Allows for timing feedback to software (ex. work stealing)

- Drawbacks
  - ILP and MLP is not modeled
  - Memory request rates are not accurate
  - Number of outstanding misses are not correct
    - Underestimation of required queue sizes
  - Does not properly hide latency like an out-of-order core
    - Overestimate runtime improvements

Core Model Case Study

- Do the drawbacks previously listed actually impact accuracy?

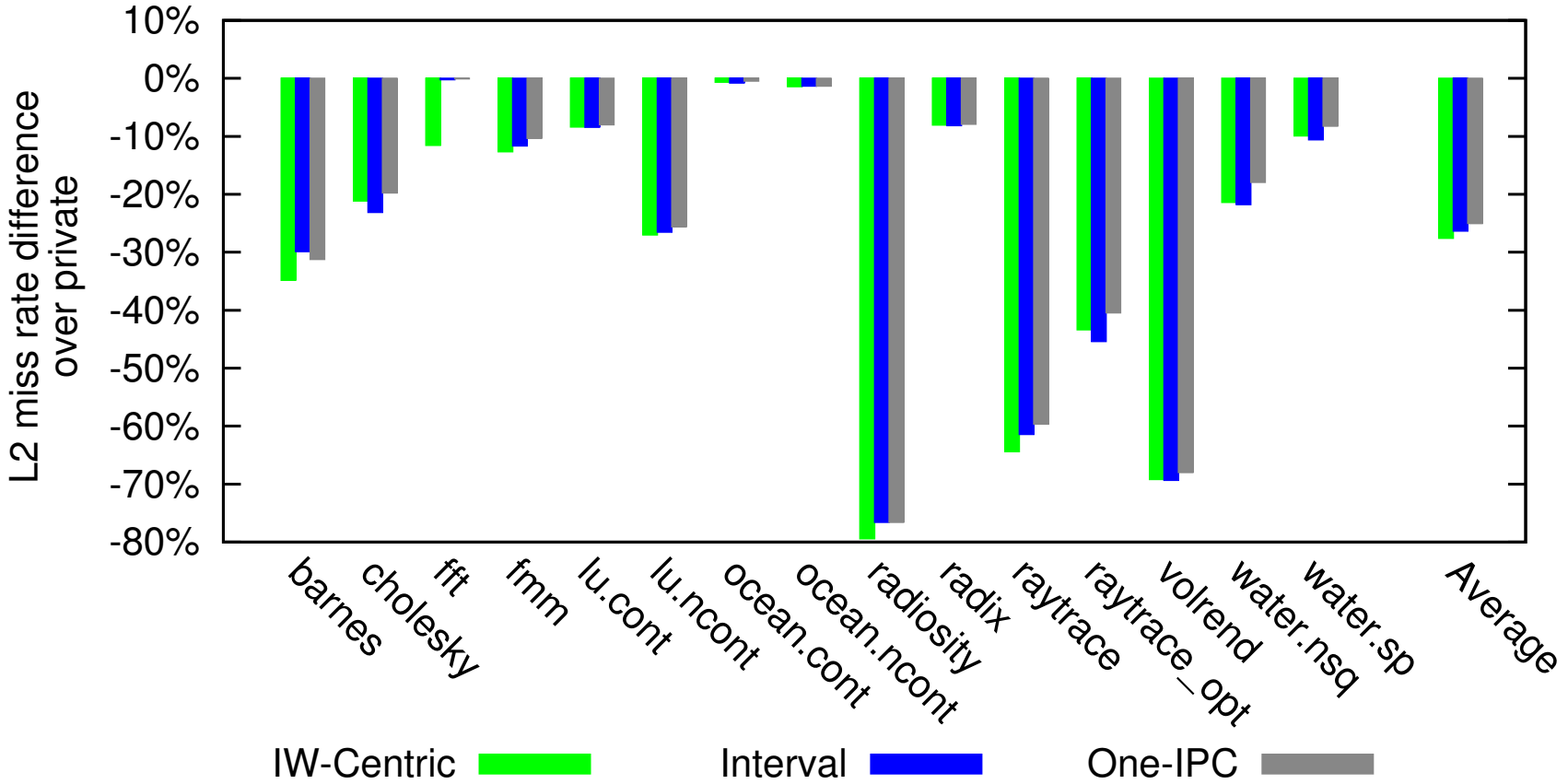
- Evaluate this core with 2 cache configurations
  - Which configuration has better performance?

Component	L2 Private Config	L2 Shared Config
Size	256 KiB / core	1 MiB / 4 cores
Associativity	8-way	16-way
Access latency	8 cycles	30 cycles

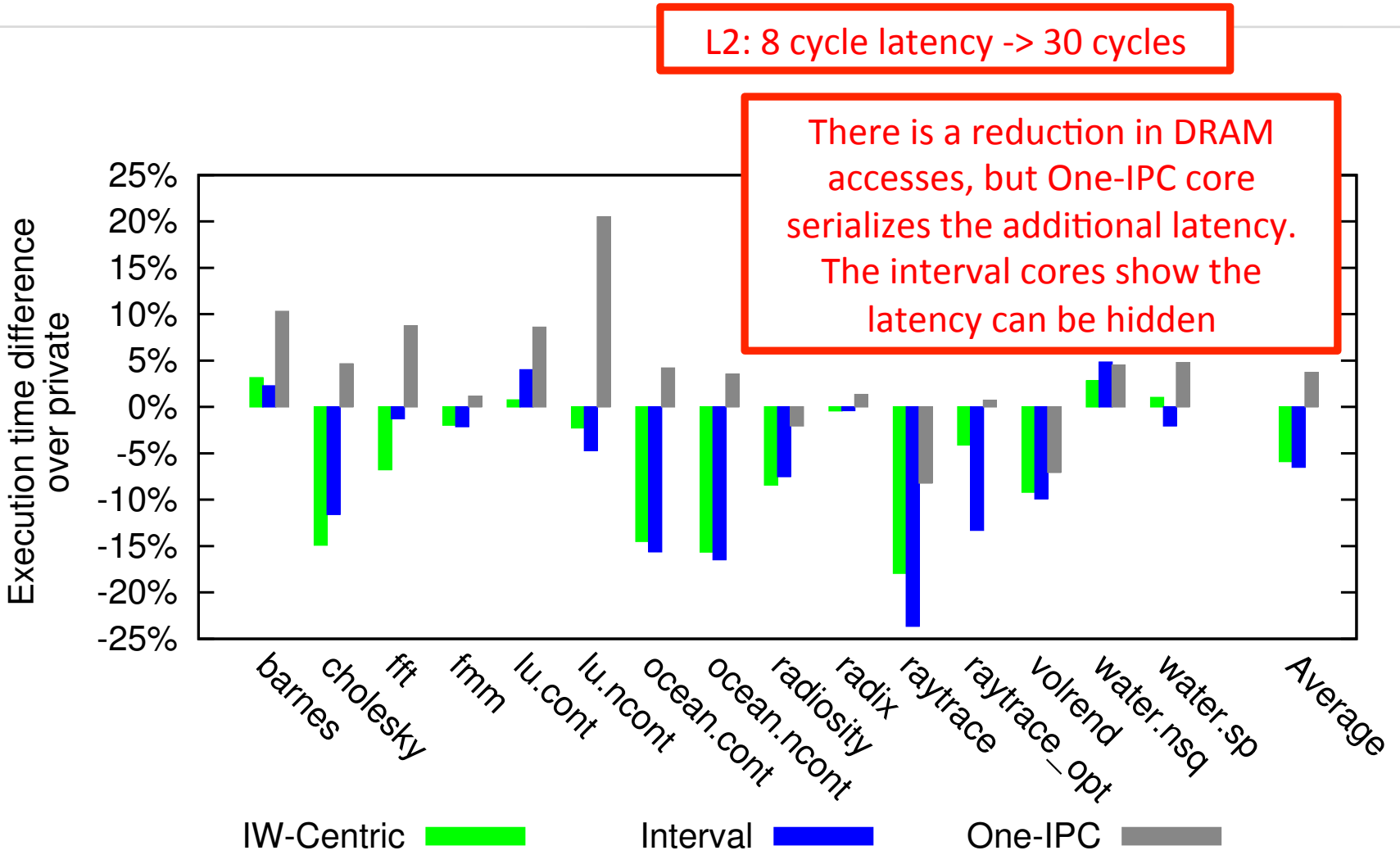
Core Model Case Study

The shared L2 reduces misses in almost all cases

One-IPC and interval core models agree

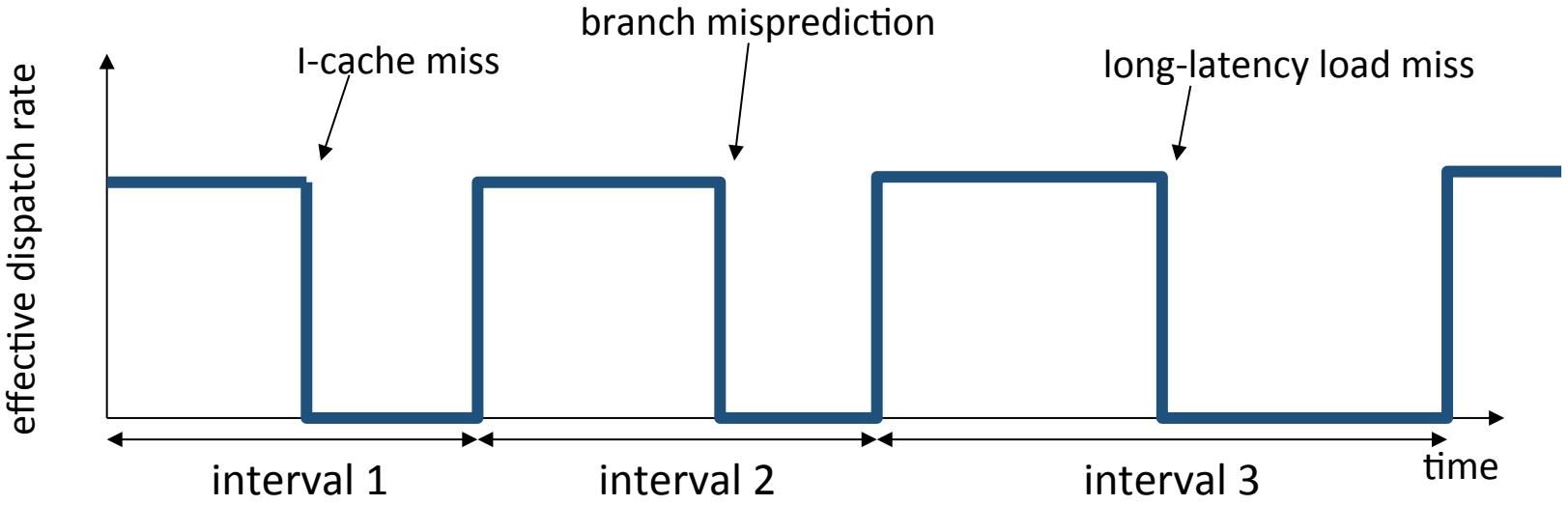


Core Model Case Study



ANALYTICAL MODELING

- Interval model is an approximation of the ideal, balanced out-of-order processor

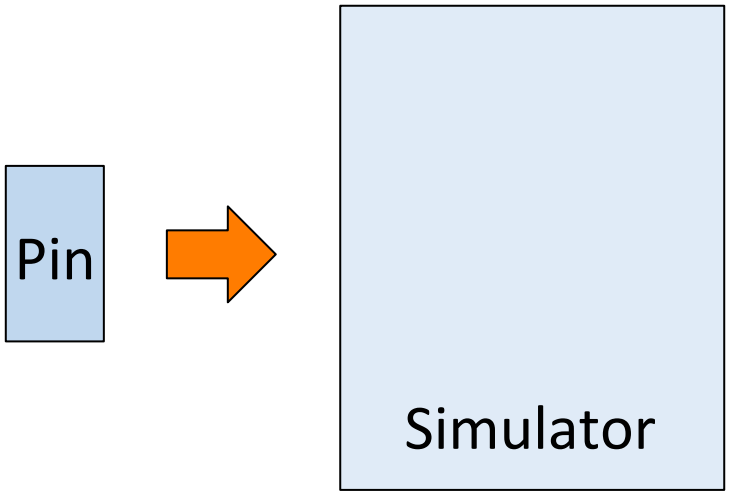


- Does not currently support multi-core processors

D. Genbrugge et al., HPCA'10
S. Eyerman et al., ACM TOCS, May 2009
T. Karkhanis and J. E. Smith, ISCA'04, ISCA'07

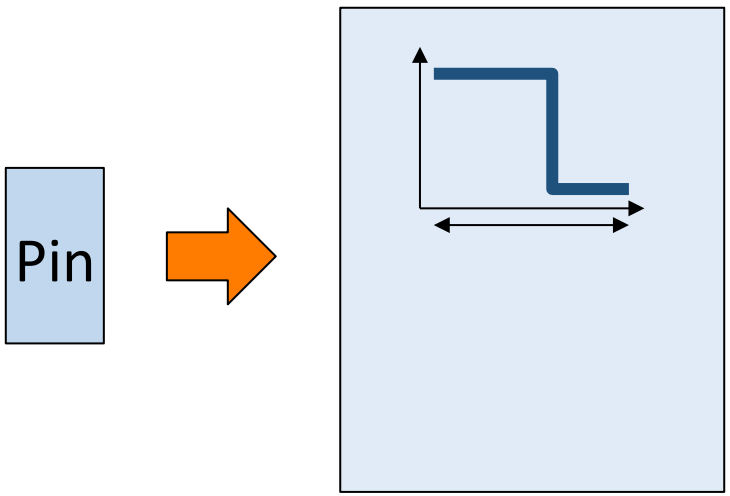
BRIDGE ANALYTICAL MODELING, SIMULATION

- Interval Simulation
  - In-order stream of instructions (from Pin, QEMU)



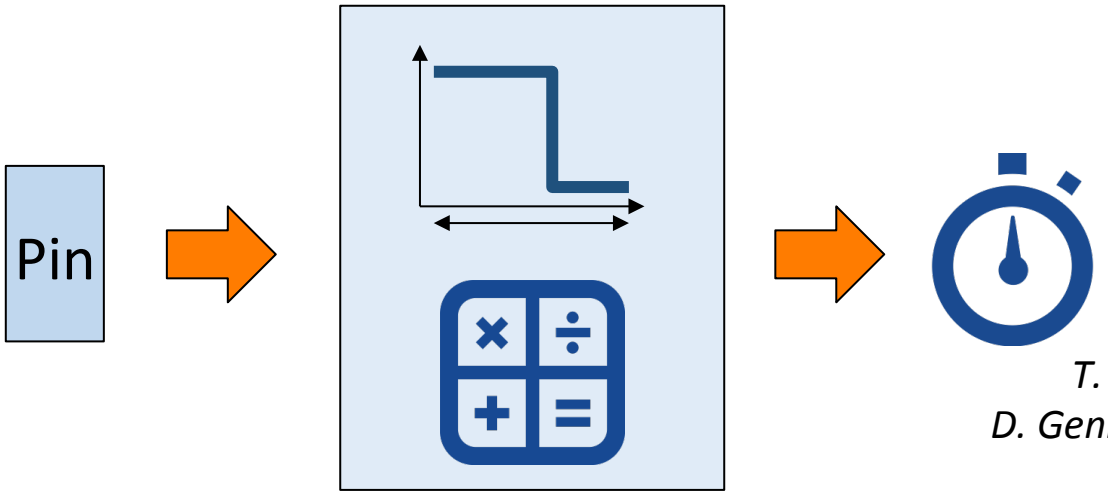
BRIDGE ANALYTICAL MODELING, SIMULATION

- Interval Simulation
  - In-order stream of instructions (from Pin, QEMU)
  - Intervals (and their corresponding delays) are formed from miss events (e.g. branch pred., etc.)



BRIDGE ANALYTICAL MODELING, SIMULATION

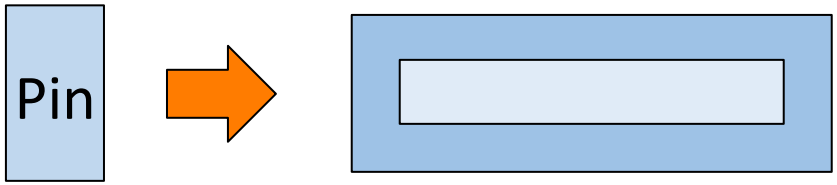
- Interval Simulation
  - In-order stream of instructions (from Pin, QEMU)
  - Intervals (and their corresponding delays) are formed from miss events (e.g. branch pred., etc.)
  - Steady-state performance is determined with queueing theory (Little's Law)



T. E. Carlson et al., SC 2011
D. Genbrugge et al., HPCA 2010

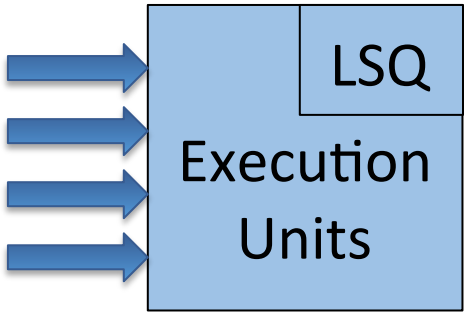
INTERVAL SIMULATION

- Little's Law and interval simulation assume that a processor is balanced with respect to the dispatch width
  - Front end (ex. fetch units) are designed to provide instructions at the dispatch with of the processor
  - The execution units can handle any type of instruction at any time
  - Commit width is sufficiently large to prevent stalling
- Predict the progress of the upcoming instructions by using the most recent instructions



Modern Processors are not Ideal

- Intel Nehalem issue ports
  - One branch port
  - One load and one store port
  - Limited ports for FP operations
  - 3 general-purpose (integer operation) ports
    - Shared with floating point ports



- Nehalem (and other modern cores) are unable to issue the dispatch-width (W=4) of instructions of the same type each cycle
  - Some applications (fp, memory-intensive, etc.) and micro-benchmarks do

Main Interval Enhancements

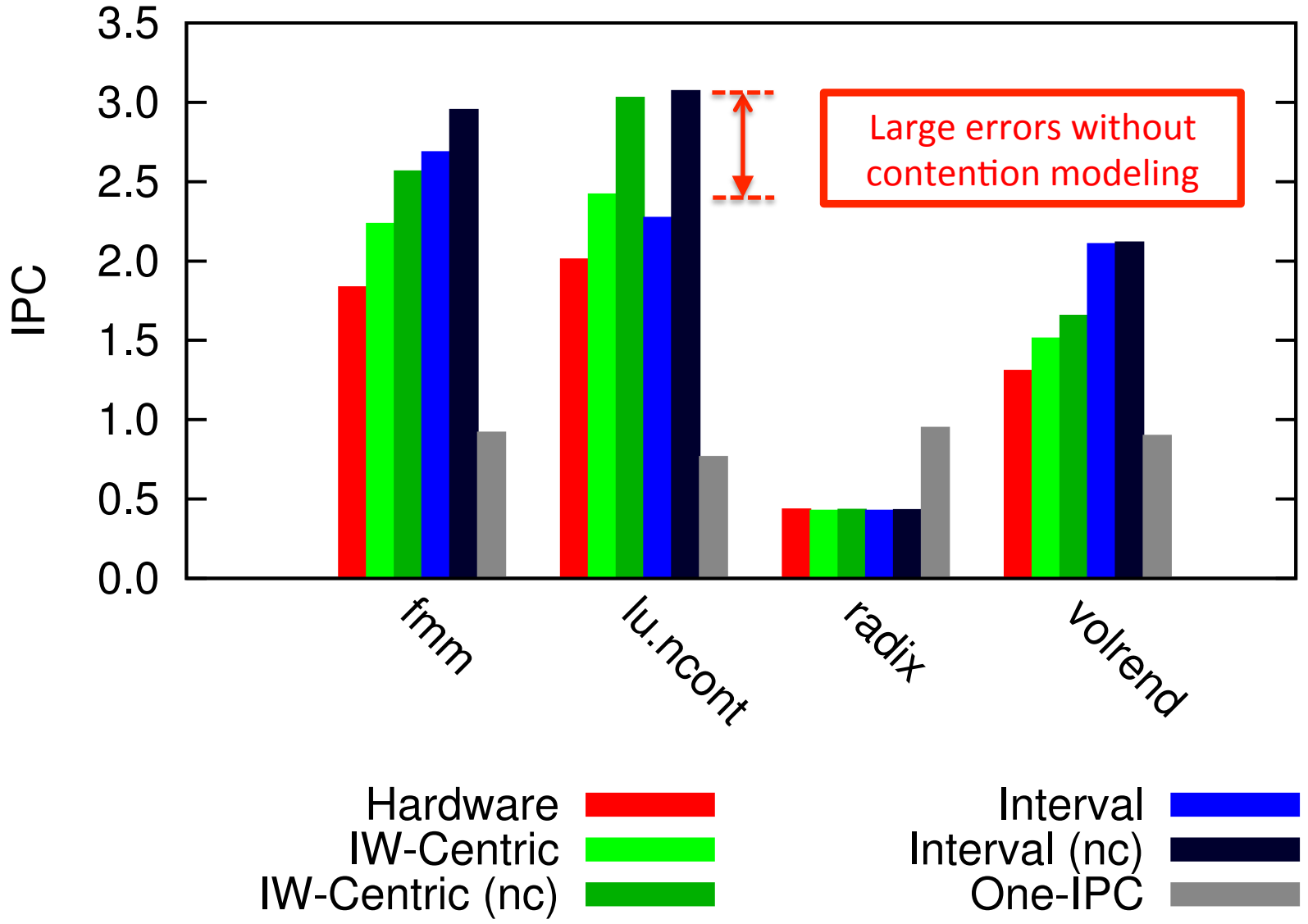
- Interval simulation issue contention
  - Takes into account the maximum execution rate possible for each port
  - For example:
    - One issue port per load
    - An application with 100% of instruction as loads
    - Results in a maximum IPC of 1.0
    - Traditional interval simulation would not take this into account and only use dependencies to derive performance
- Instruction-Window Centric (IW-Centric)
  - Replaces Little's Law calculations with detailed cycle-level wake-up and issue logic
  - Slower code as the expense of higher fidelity

Experimental Setup

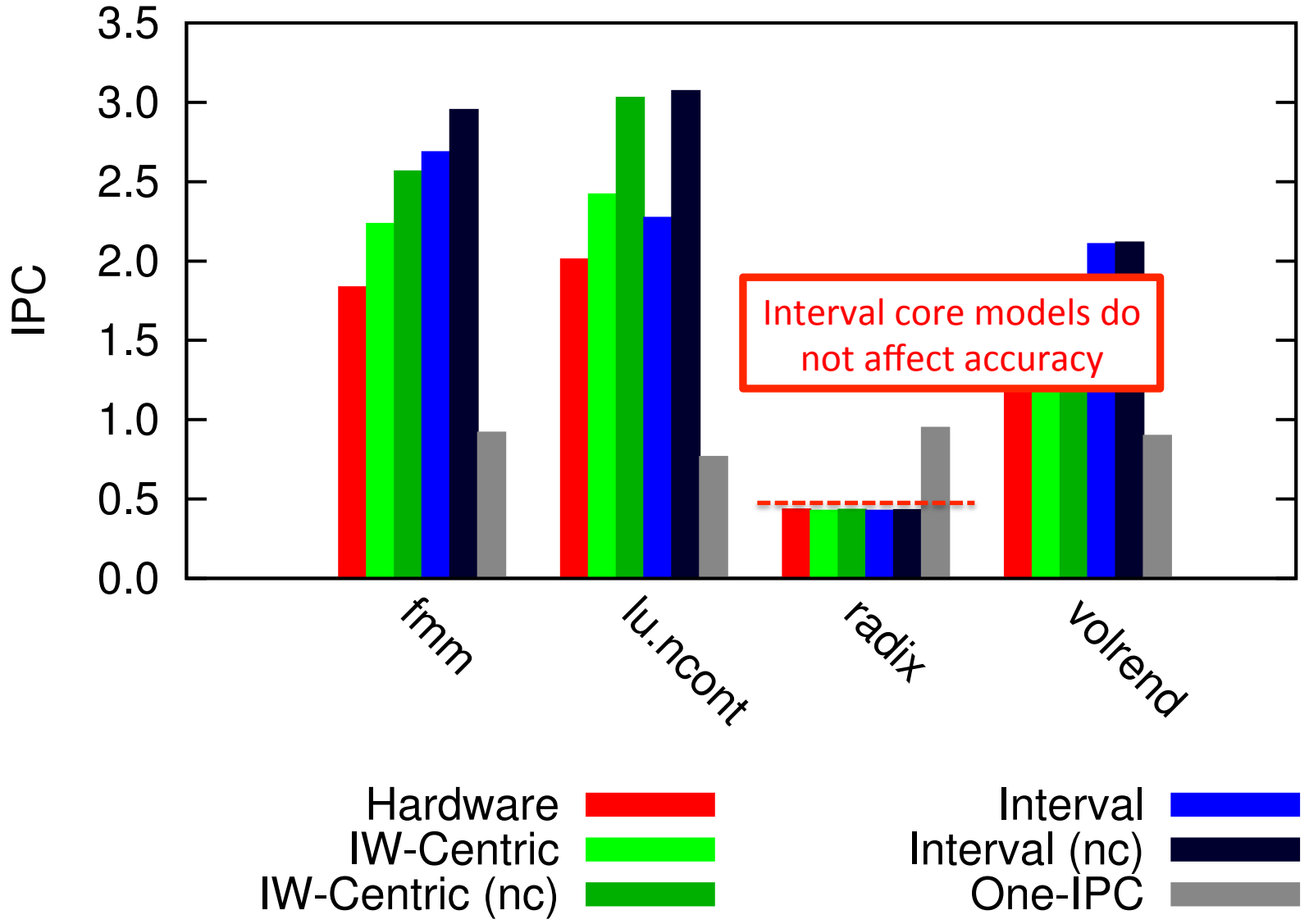
- Sniper Multi-Core Simulator, version 6.0
- SPLASH-2 Benchmark Suite
- Modeled the Intel Xeon X5550 (Nehalem)

- Microarchitectural Configuration
  - 1 and 2 sockets, 4 cores per socket
  - 2.66 GHz, 4-way dispatch, 128-entry ROB
  - Dothan branch predictor
  - L1-I 32 KB, 4 way, 4 cycle access time
  - L1-D 32 KB, 8 way, 4 cycle access time
  - L2 256 KB, 8 way, 8 cycle access time
  - L3 8MB per 4 cores, 16 way, 30 cycle access time
  - DRAM 65 ns access time, 8GB/s per socket
  - Inter-processor Bus QPI, 12.8 GB/s per direction

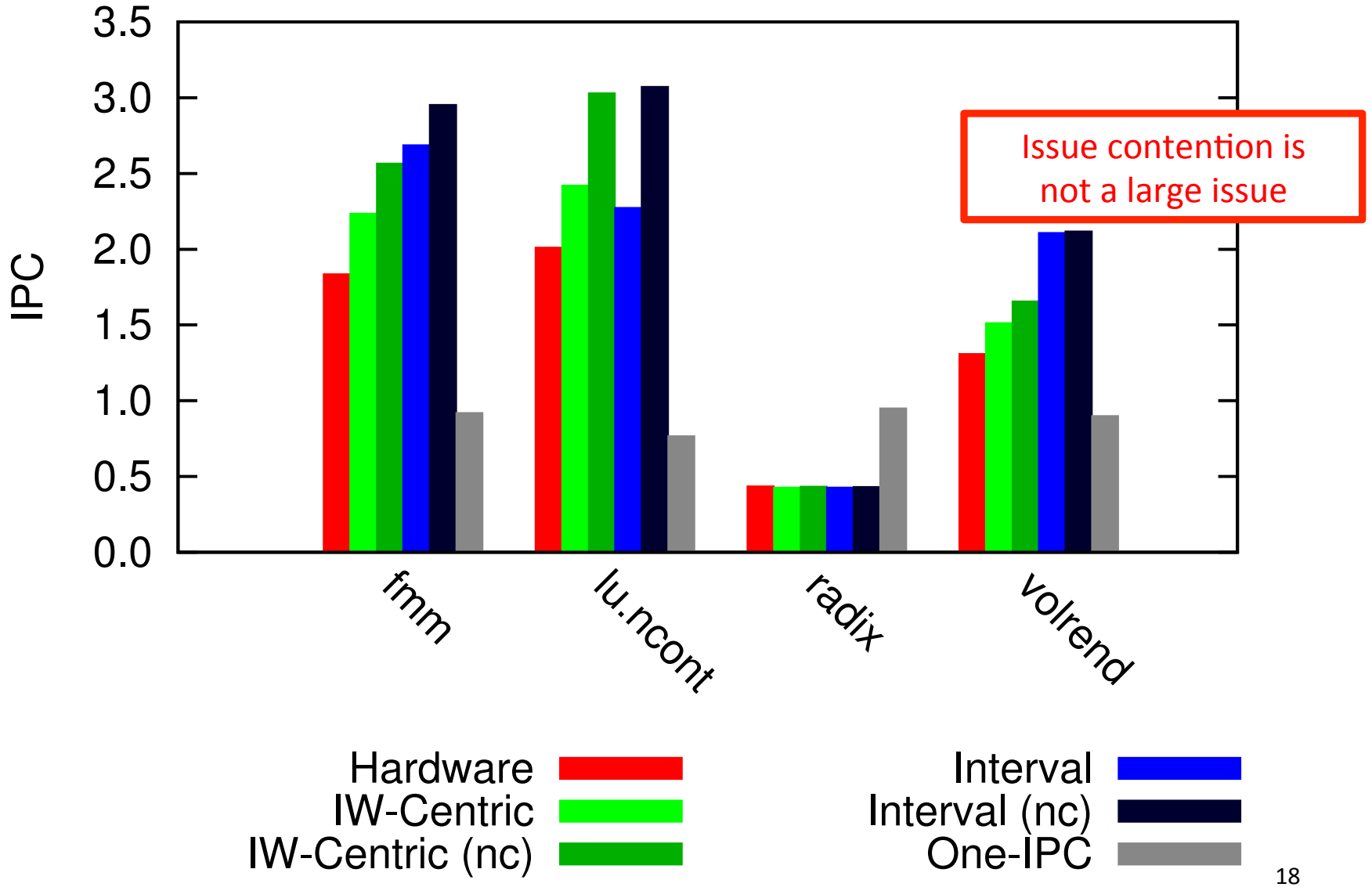
SIMULATION ACCURACY



SIMULATION ACCURACY



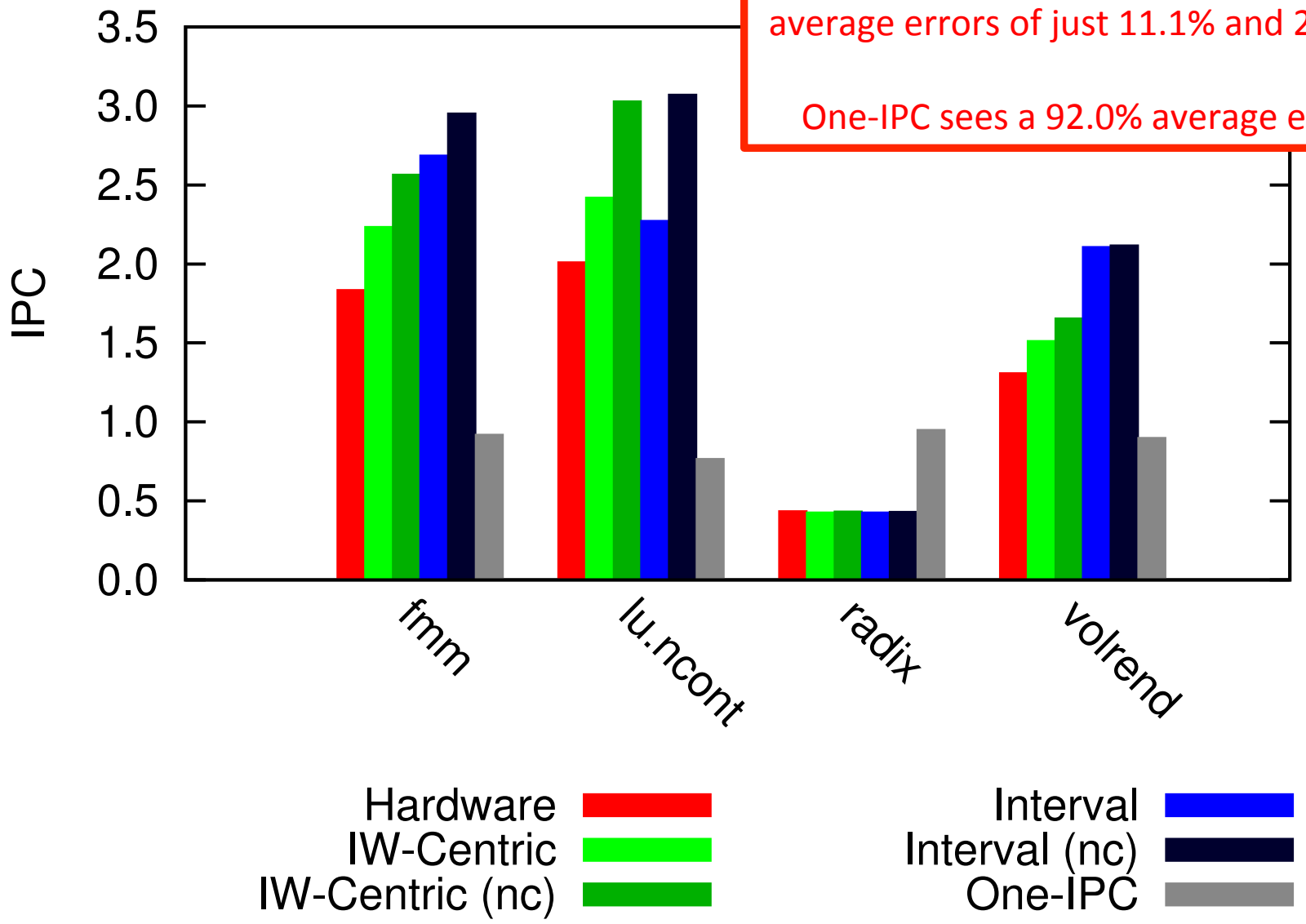
SIMULATION ACCURACY



Simulation Accuracy

IW-Centric and Interval have low average errors of just 11.1% and 24.3%.

One-IPC sees a 92.0% average error

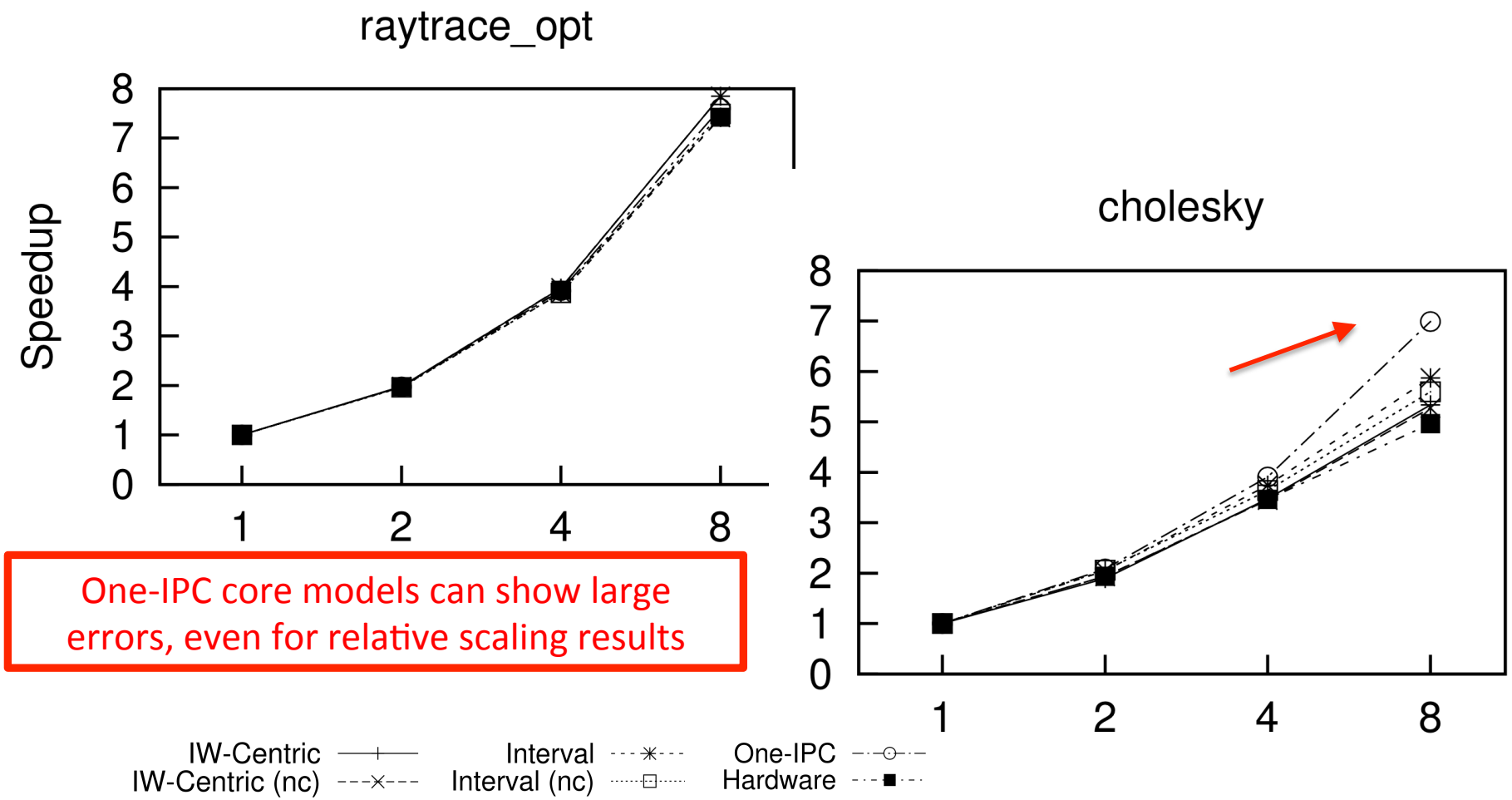


PERFORMANCE CONCLUSIONS

- Interval simulation model has been improved to more accurately reflect limitations of modern processors

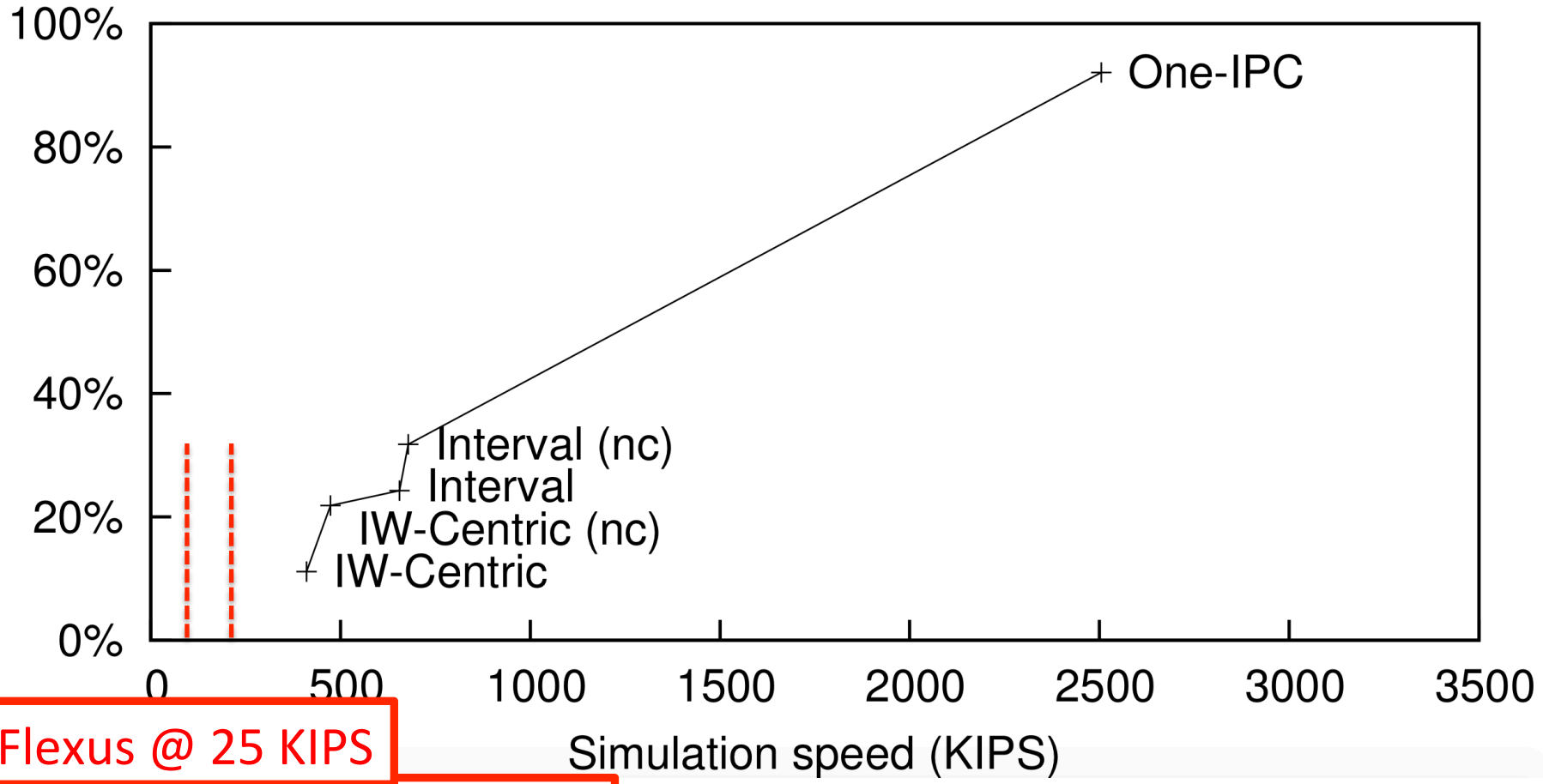
- The instruction-window centric model allows us to better understand interval simulation error
  - Interval simulation works with averages over the length of the ROB
  - More precise handling of the dependencies in the ROB provides added accuracy while maintaining performance

Relative Scaling Results



One-IPC core models can show large errors, even for relative scaling results

Simulation Speed Comparison



Flexus @ 25 KIPS

gem5 @ 200 KIPS

ADDITIONAL RESULTS IN THE PAPER

- Enhancements to interval simulation
  - Improved modeling of overlapping memory accesses
  - Improve modeling of the front-end miss refill rate
- Additional relative scaling results
- Detailed issue contention example
- Simulation speed and scaling comparison

SUMMARY

- Enhance interval simulation to support issue contention to improve accuracy with very little slowdown
  - 24% avg. absolute error vs. hardware
- Developed a new core model to bridge the performance/accuracy gap with cycle-level simulation
  - 11% avg absolute error vs. hardware
  - Allows for future core designs (in-order, SMT) without the need for analytical models
- Show results that imply that caution must be taken when using simple (One-IPC-style) core models for absolute or even relative studies
- Recently released in Sniper 6.0
  - Interval simulation enhancements
  - IW-Centric core model

An Evaluation of High-Level Mechanistic Core Models

Trevor E. Carlson, Wim Heirman,
Stijn Eyerman, Ibrahim Hur, Lieven Eeckhout

Universiteit Gent
intel
Uppsala Universitet

http://www.snipersim.org
Monday, January 19th 2015
HiPEAC Conference 2015, Amsterdam