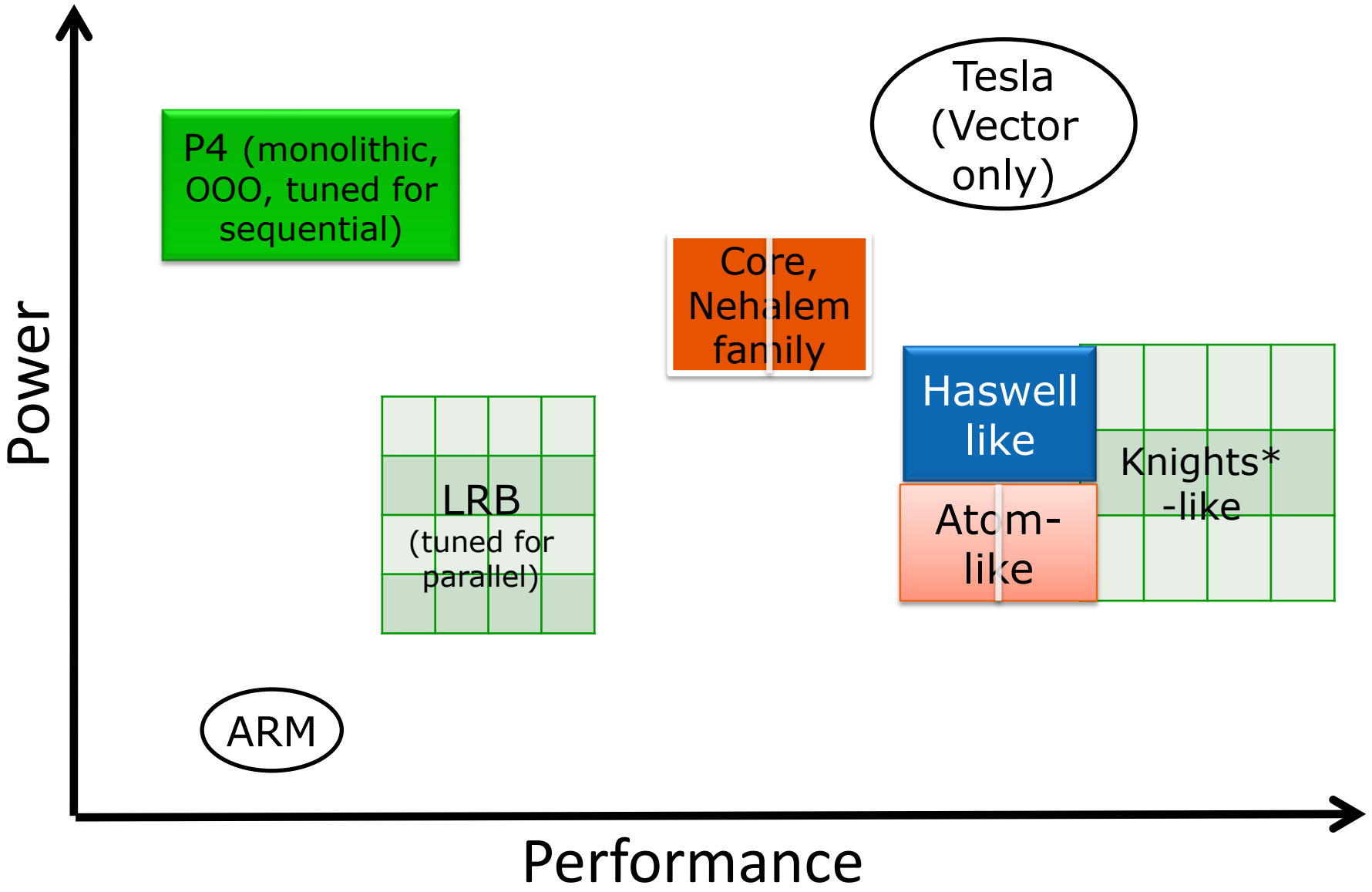
Montage: Scheduling Virtual Machines on Asymmetric Multi-core Platforms

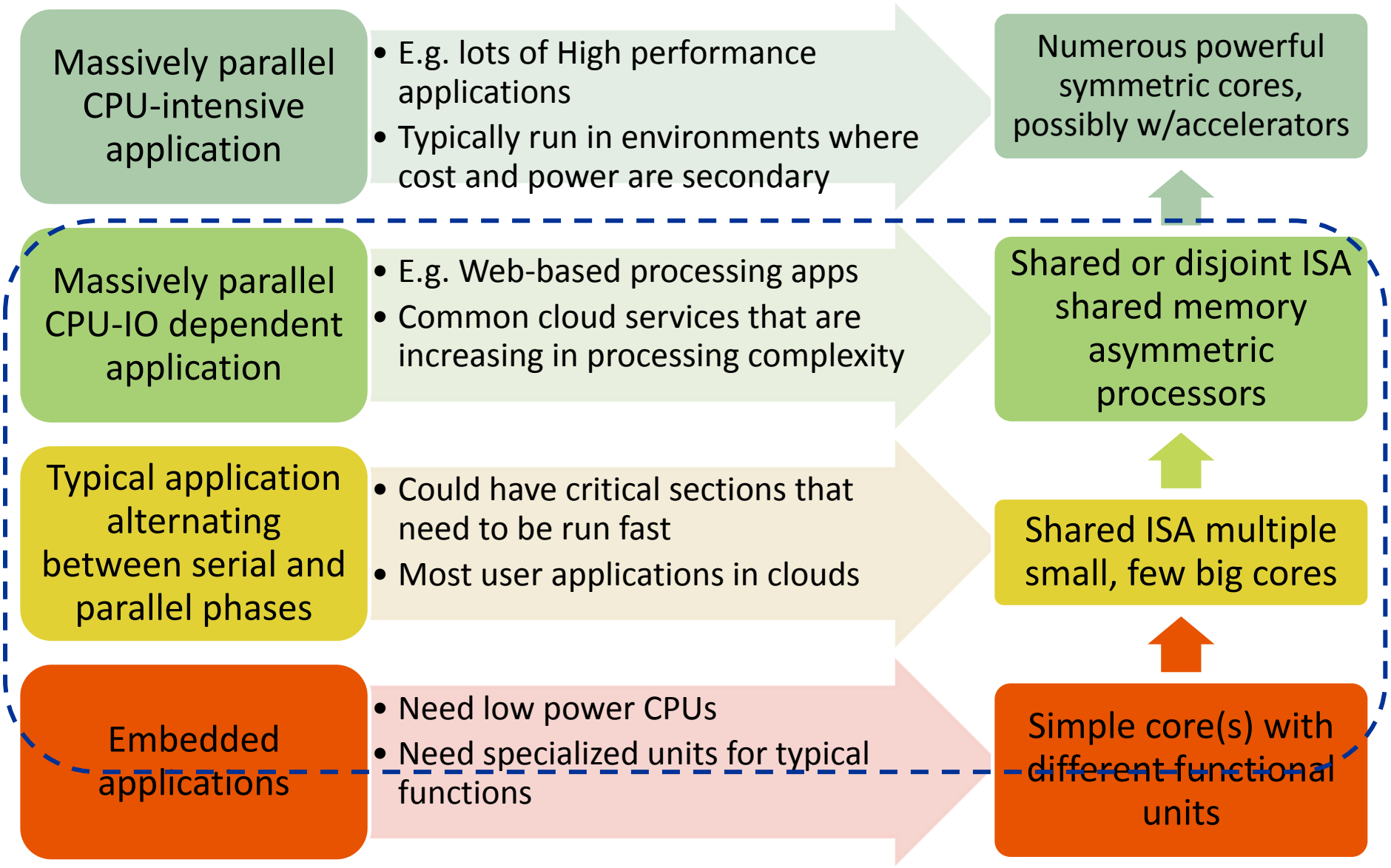
Vishakha Gupta, Karsten Schwan @ GT
Rob Knauerhase, Paul Brett @ Intel

Disclaimer: Contents in this document have been modified to exclude confidential and unpublished material

Heterogeneity Trend



Heterogeneity helps…



Industrial/Academic Response

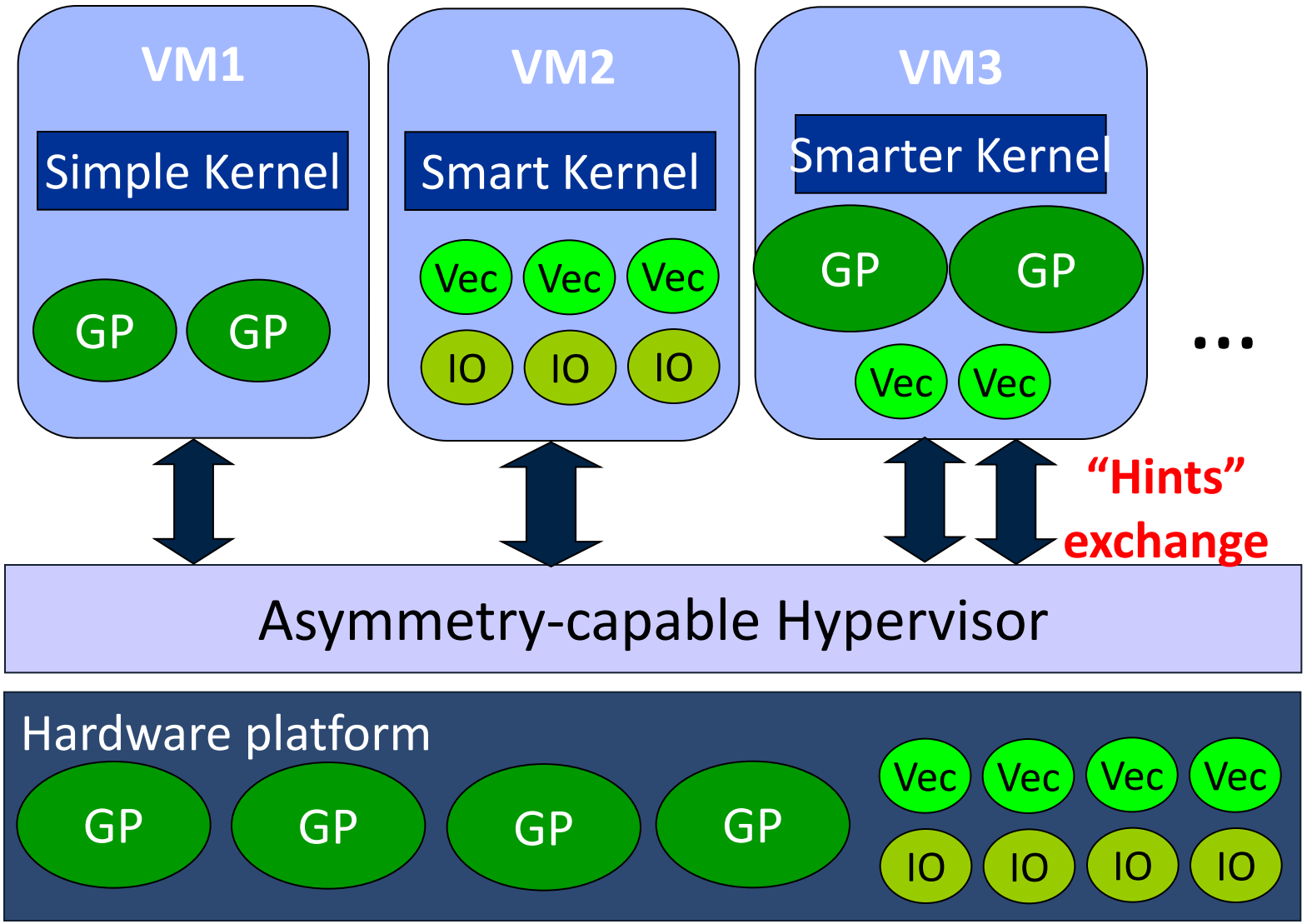
- Hardware response → asymmetric multi-cores
  - Performance asymmetry
  - Functional asymmetry
  - Scalable architectures
- Software response → treat these as peripherals instead of cores
  - Device driver model – hide complexity and maintain status quo
  - Limits what could really be achieved! [GViM]
- Research Challenges → in the right direction
  - Novel execution and programming models, runtimes
  - Asymmetry-aware OS [OBS] [Bias] [CAMP]
  - Power management, auto tuning for performance asymmetry

But efforts at lowest level resource management layer still quite preliminary [AASH]!

Outline

- Software architecture
- Montage VCPU Scheduling
- Current status
- Experimental results
- Future work

Target Software Architecture



Montage VCPU Scheduling

- Assignment of VCPUs to PCPUs
  - Picks a PCPU with highest affinity value
  - Updates the load on picked PCPU
- Observe and modify
  - Updates parameters in the affinity equation wherever required, e.g. on a fault
  - Might result in VCPU migration
- Re-matching VCPU-PCPU Mappings
  - Run periodically for accounting any changes in affinity values over the past period
  - Updates mappings as desired by current situation
  - Might result in VCPU migrations

Defining the "Hints" Interface [1]

- set_vcpu_category(Domain d, Vcpu v, Category c)
  - PCPU_GENERAL
  - PCPU_VECTOR
  - PCPU_CRYPTO
  - PCPU_NETWORK
  - ……
- Default category for all VCPUs is general
- Shared ISA PCPUs are PCPU_GENERAL but can additionally be
  - PCPU_VECTOR if SSE supported
  - PCPU_CRYPTO if AES supported

Would accommodate disjoint ISAs in future

Defining the "Hints" Interface [2]

- set_vcpu_expectation(Domain d, Vcpu v, Expectation e)
  - VCPU_VARYING_OR_UNKNOWN
  - VCPU_MOSTLY_CPU_INTENSIVE
  - VCPU_MOSTLY_CACHE_INTENSIVE
  - VCPU_MOSTLY_IO_INTENSIVE
  - …..
- Default category for VCPUs - VARYING_OR_UNKNOWN
- Parameters are easily determinable by user OR from an asymmetry aware OS

Would be extremely useful in scenarios where the same workload is run for a long time and its behavior is deterministic (e.g. HPC)

Current Status

- Working implementation in hypervisor and automation for experiments
- Hooks for observation based data
- Dom0 level control of processor asymmetry
- Early design and implementation of the “hints channel”

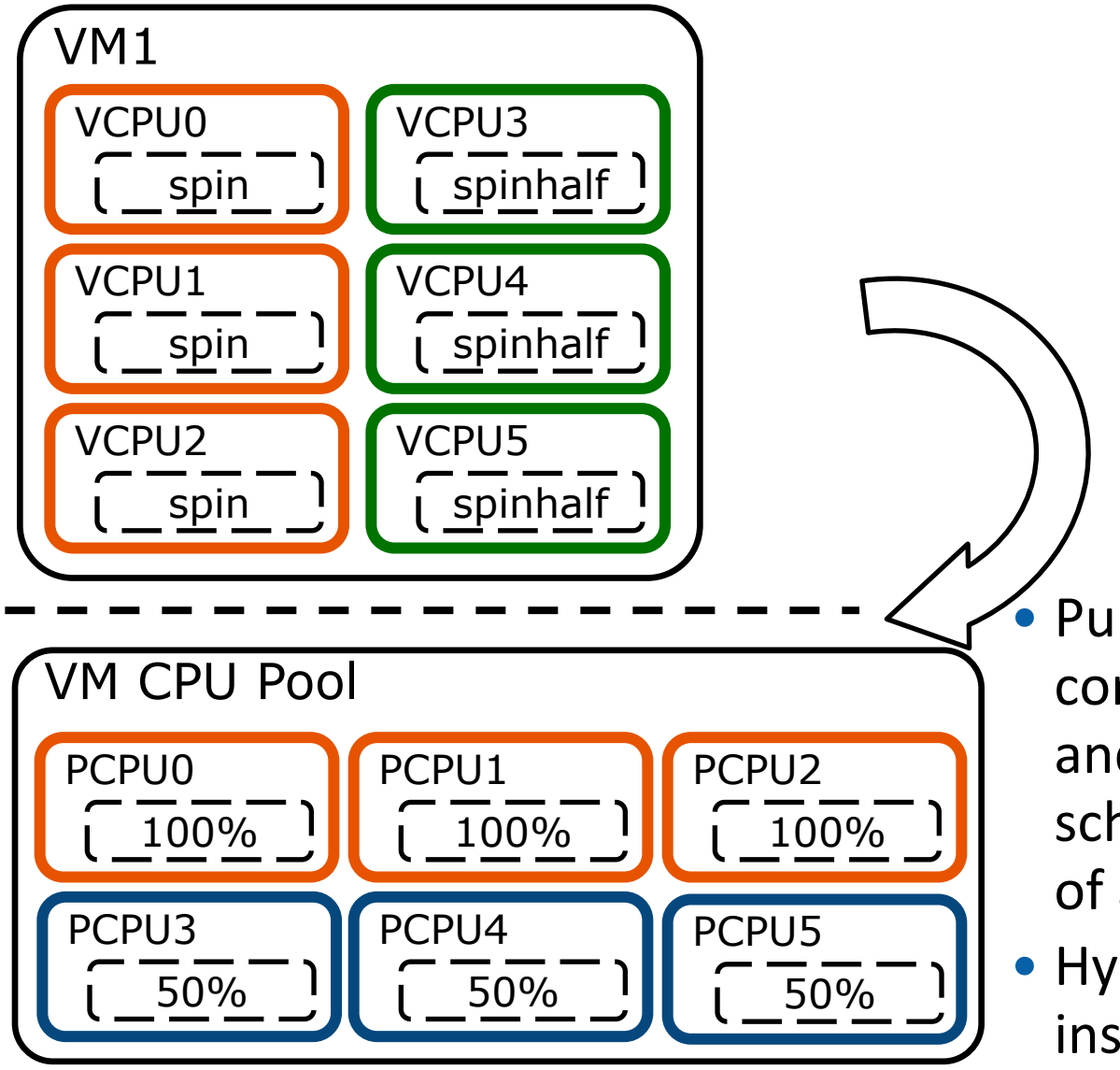
”Attaining System Performance Points: Revisiting the End-to-End Argument in System Design for Heterogeneous Many-core Systems” – SigOps Operating Systems Review 2011

EXPERIMENTAL EVALUATION

Evaluation Methodology

- 12 core Westmere dual socket machine with 6GB RAM and 12MB L2 cache
- Software – Xen 4.0 testing, Linux 2.6.31.13 pvops Dom0 and FC12 guest domains
  - Montage scheduler vs. Credit scheduler
  - Floor and ceiling numbers wherever possible to get an idea of what is the best and worst possible
- Hypercall interfaces to change
  - VCPU – cpupool information
  - VCPU – category and expectations
  - PCPU – MSR values for duty cycle changes

Experiment 1

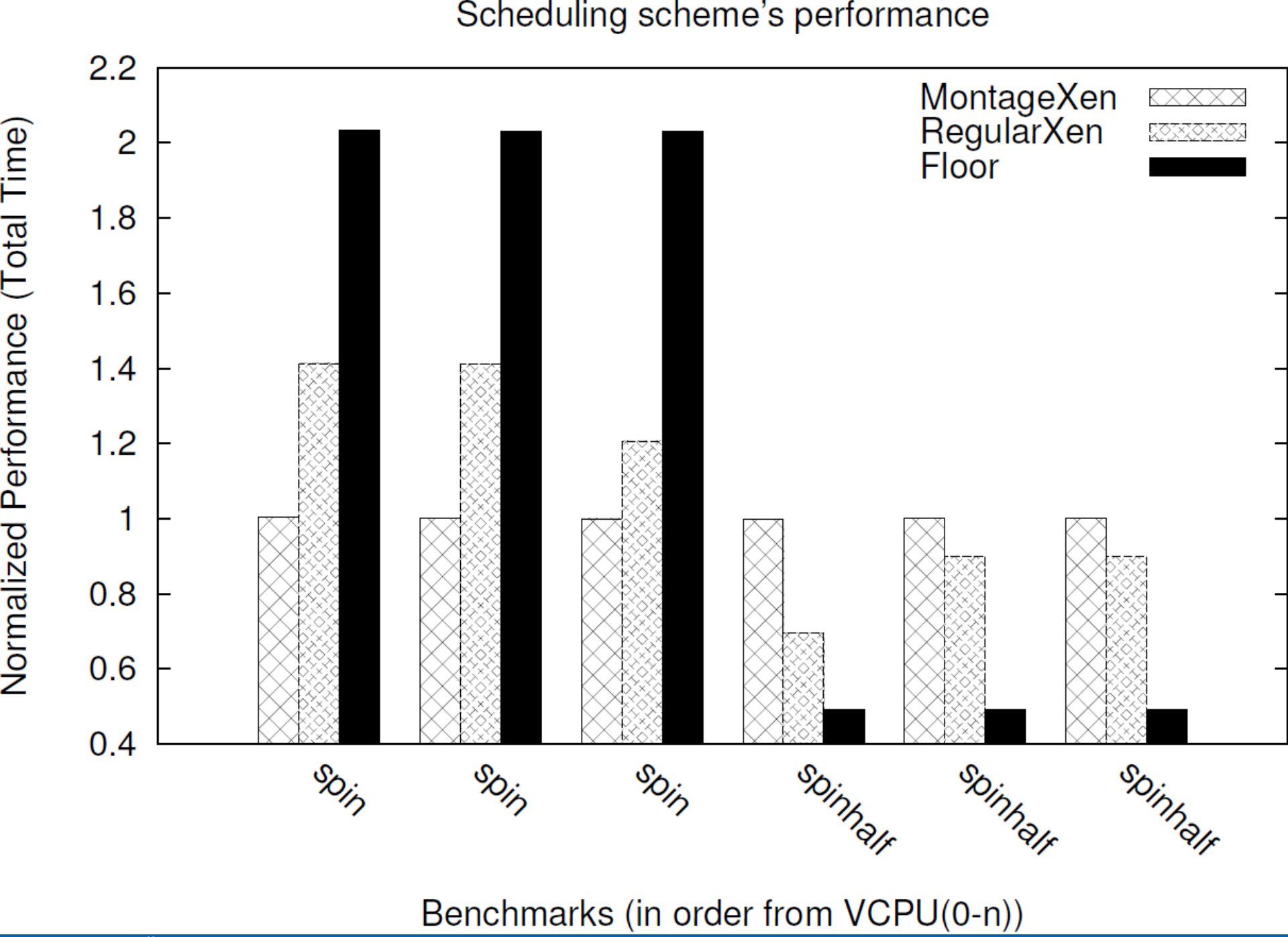


CPU_INTENSIVE VCPU – full performance PCPU

GENERAL VCPU – 75% performance PCPU

MEM_INTENSIVE VCPU– 50% performance PCPU

- Purpose: To evaluate correctness of equations and need for better scheduling in the presence of asymmetry
- Hypothesis: All spin instances on PCPUs 0-2, all spin_halfs on PCPUs 3-5


Scheduling scheme's performance
MontageXen
RegularXen
Floor
Normalized Performance (Total Time)
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
spin
spin
spin
spinhalf
spinhalf
spinhalf
Benchmarks (in order from VCPU(0-n))

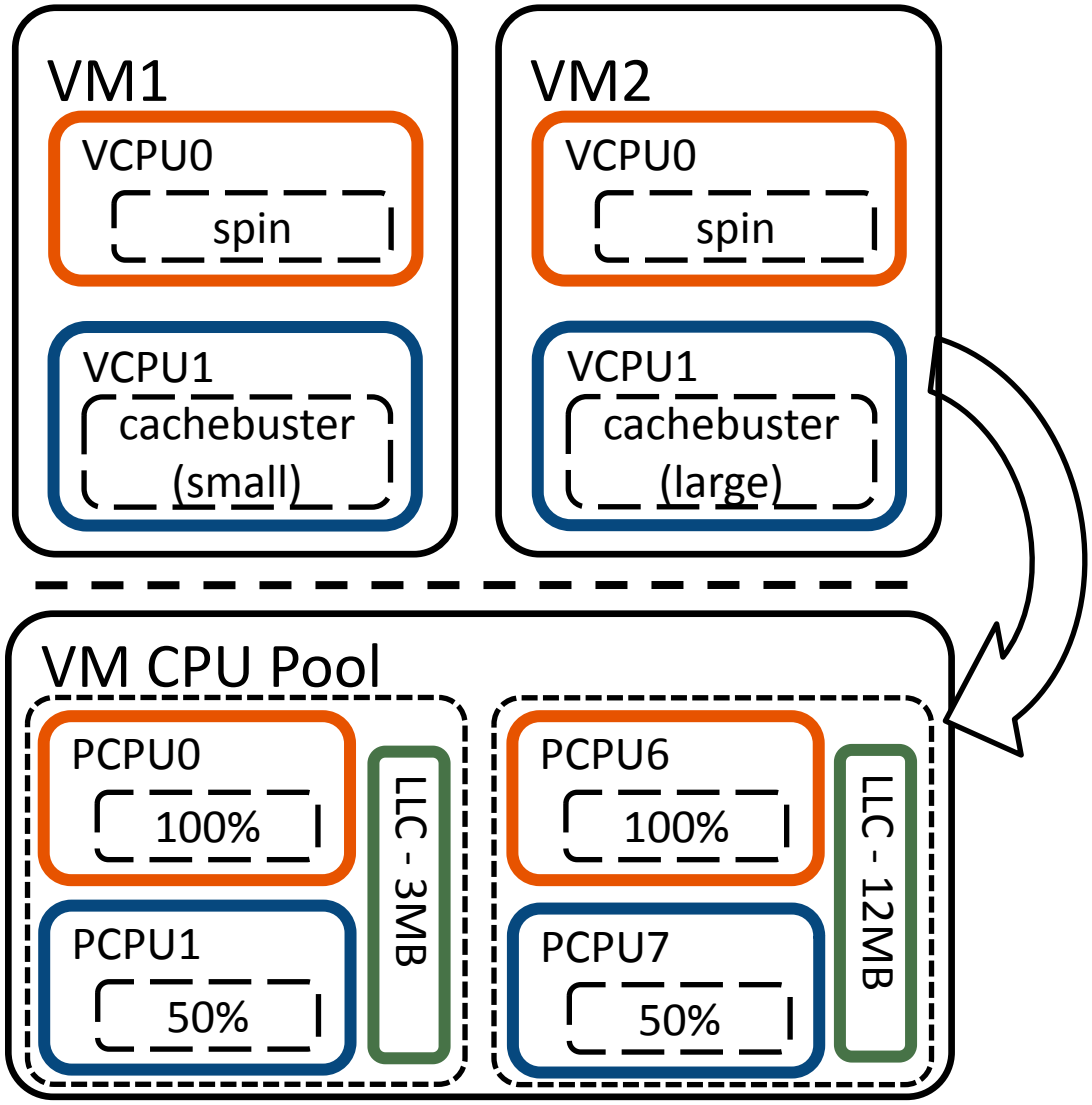
Experiment1 - Discussion

- Floor captures the worst times (by pinning VCPUs 0-2 on CPUs 3-5 and VCPUs3-5 on CPUs0-2) – from the perspective of the more intensive benchmark
- Ceiling does the right thing – spin instances on the faster CPUs vs. spinhalf on the 50% ones

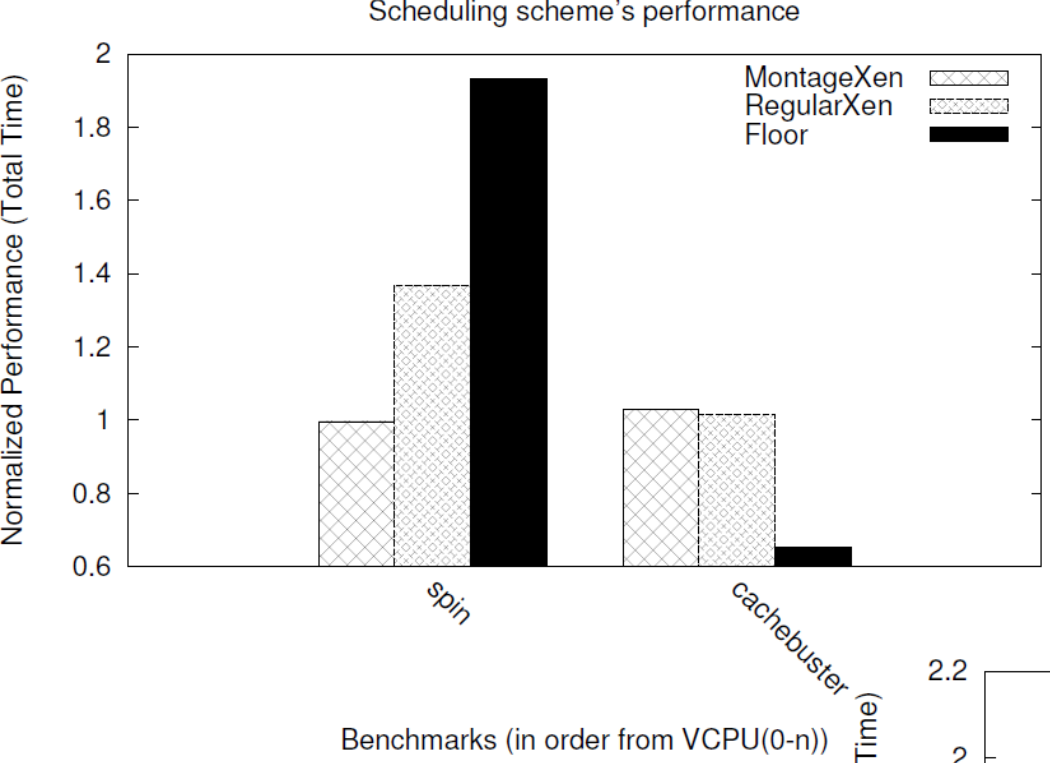
Scenario	**Average total time (sec) for 5 runs**	**Average deviation over the 5runs**
MontageXen	261.678	0.223
RegularXen	283.85	15.05
Floor	328.475	0.23
Ceiling	261.5	0.144

MontageXen takes less execution time overall and shows more predictable and stable behavior

Experiment 2 – Speed + Cache Asymmetry

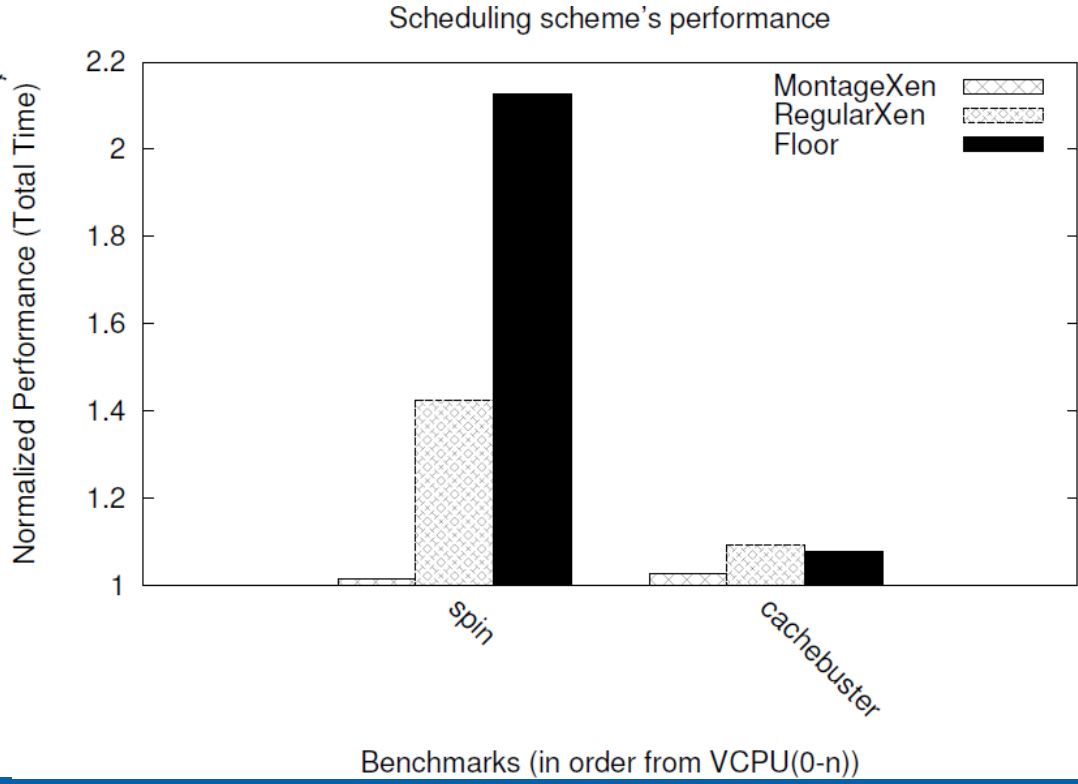


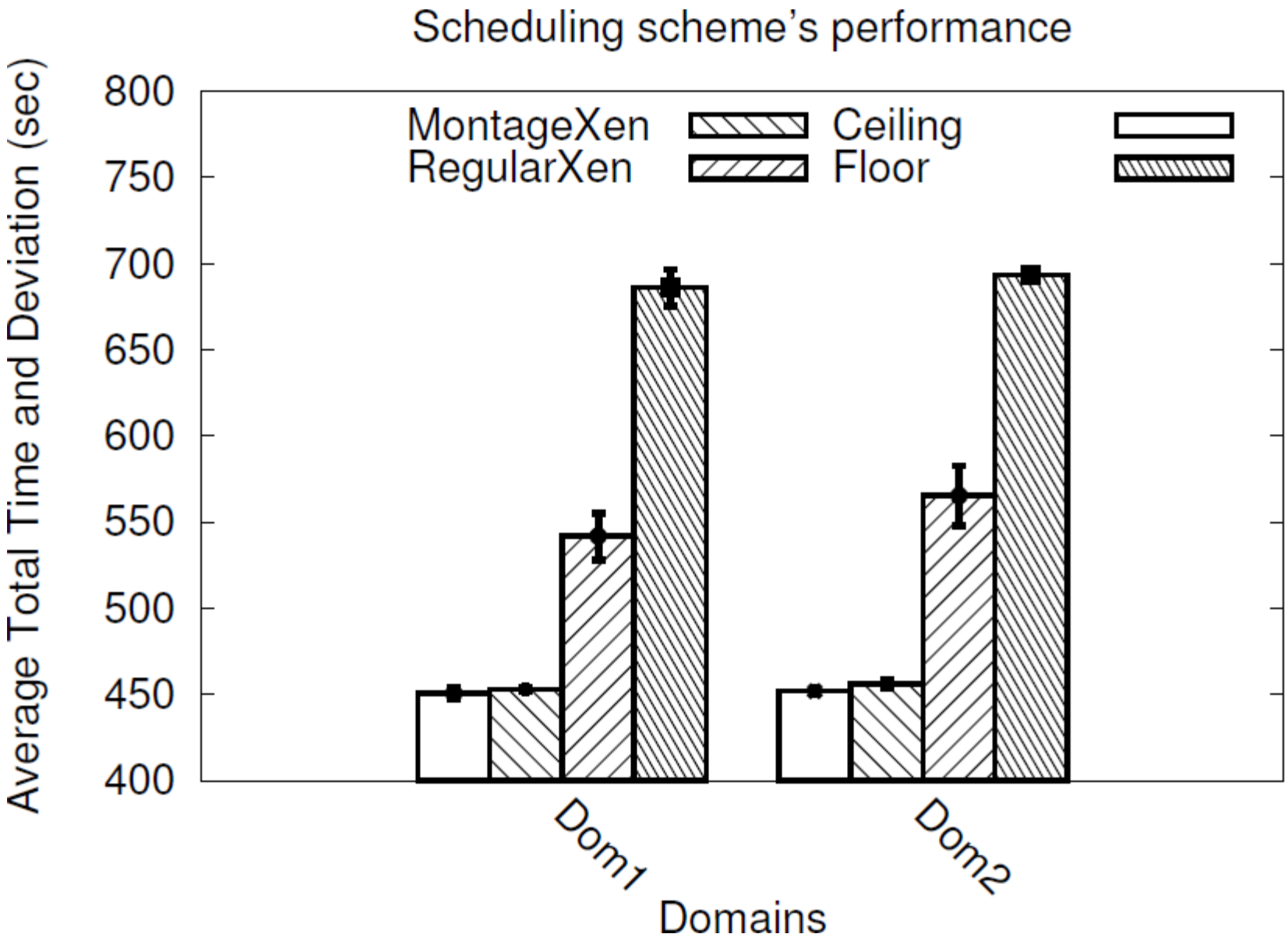
- Purpose: To evaluate performance for cache and speed asymmetry
- Hypothesis: Both spins on PCPU0, PCPU6 respectively. Small cachebuster on PCPU1, large on PCPU7


Scheduling scheme's performance
Normalized Performance (Total Time)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
MontageXen
RegularXen
Floor
spin
cachebuster
Benchmarks (in order from VCPU(0-n))

Dom1

Dom2


Scheduling scheme's performance
Normalized Performance (Total Time)
2.2
2
1.8
1.6
1.4
1.2
1
MontageXen
RegularXen
Floor
spin
cachebuster
Benchmarks (in order from VCPU(0-n))


Scheduling scheme's performance
Average Total Time and Deviation (sec)
800
750
700
650
600
550
500
450
400
MontageXen
RegularXen
Ceiling
Floor
Dom1
Dom2
Domains

Future Work

- Complete observation capability in hypervisor and further evaluation
- Evaluation of hints channel
- Evaluation with smart guest OSes
- Scalability analysis

Thank you!