Slow and Steady

Measuring and Tuning Multicore Interference

Dan Iorga, Tyler Sorensen, John Wickerson, Alastair F. Donaldson

Imperial College London

Motivation

Image taken from Wikipedia

Real-time:

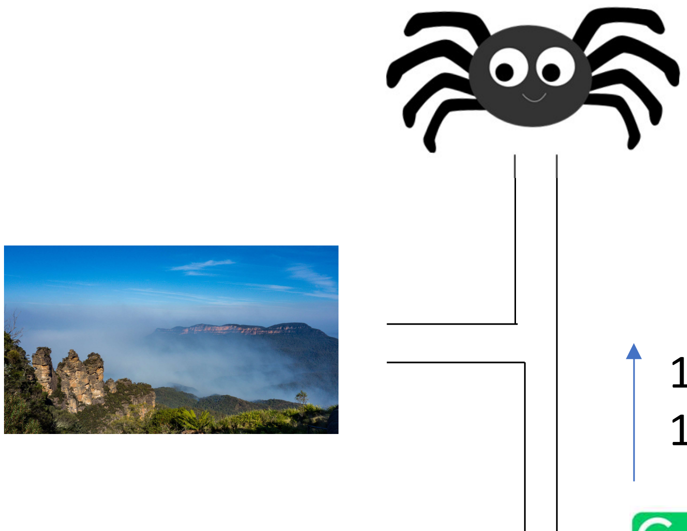
- Needs timing predictability
- Can benefit from multicore processors

Our work:

- Reproducible measurements
- Uncovering aggressive configurations

https://github.com/mc-imperial/multicore-test-harness

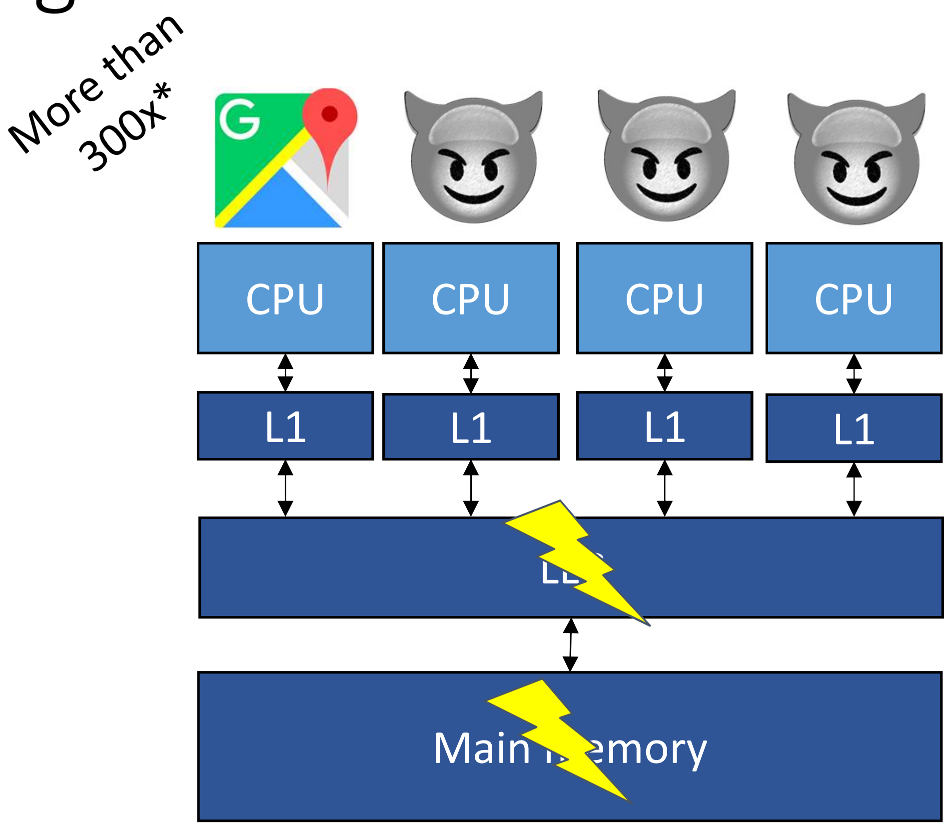
Motivation



10 second deadline
1 second execution time with no interference

https://github.com/mc-imperial/multicore-test-harness

Measuring interference



*Bechtel and Yun. RTAS 2019

Motivation

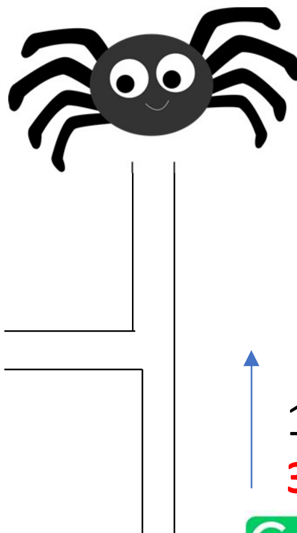


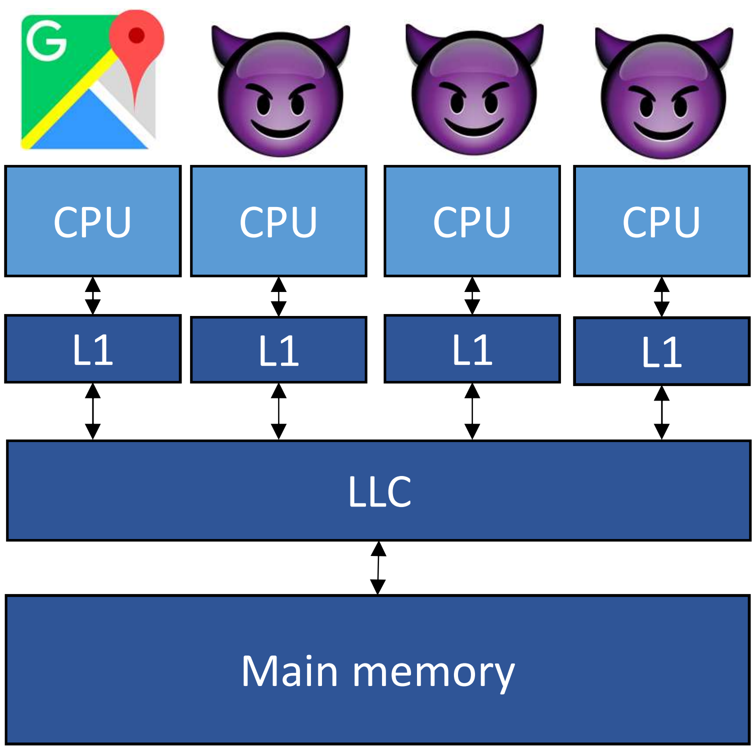
Image taken from Wikipedia

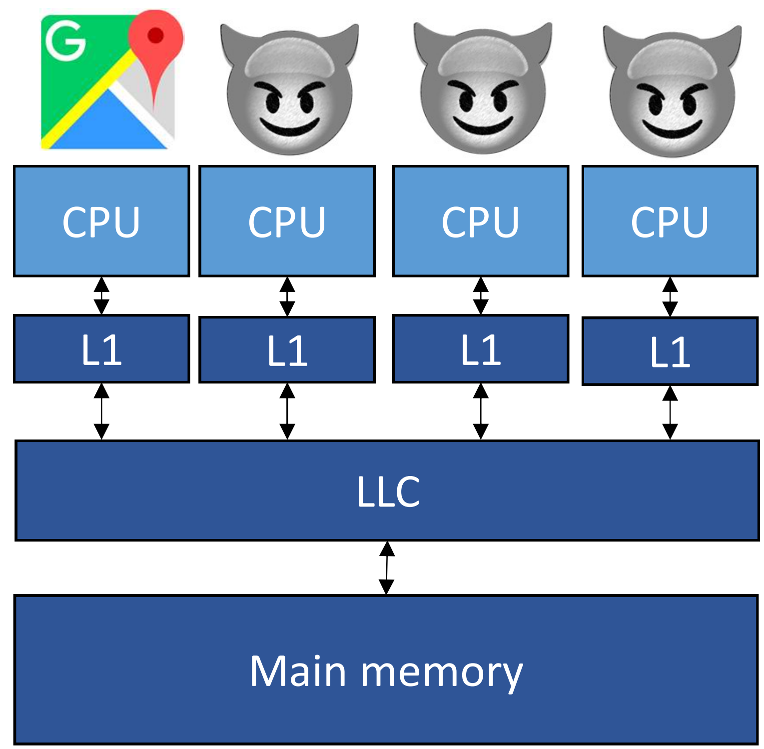
10 second deadline

300 seconds delay with interference

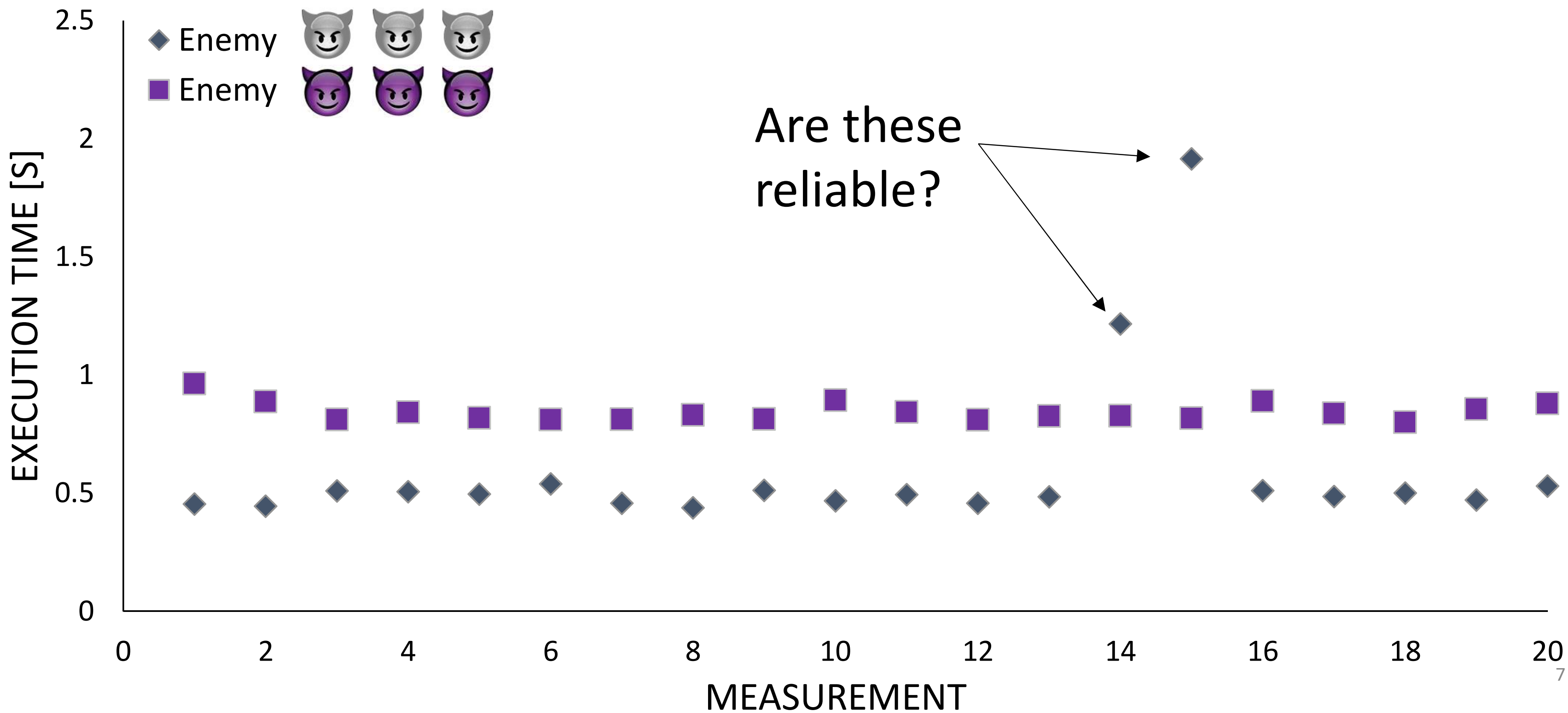
https://github.com/mc-imperial/multicore-test-harness

Comparing enemies

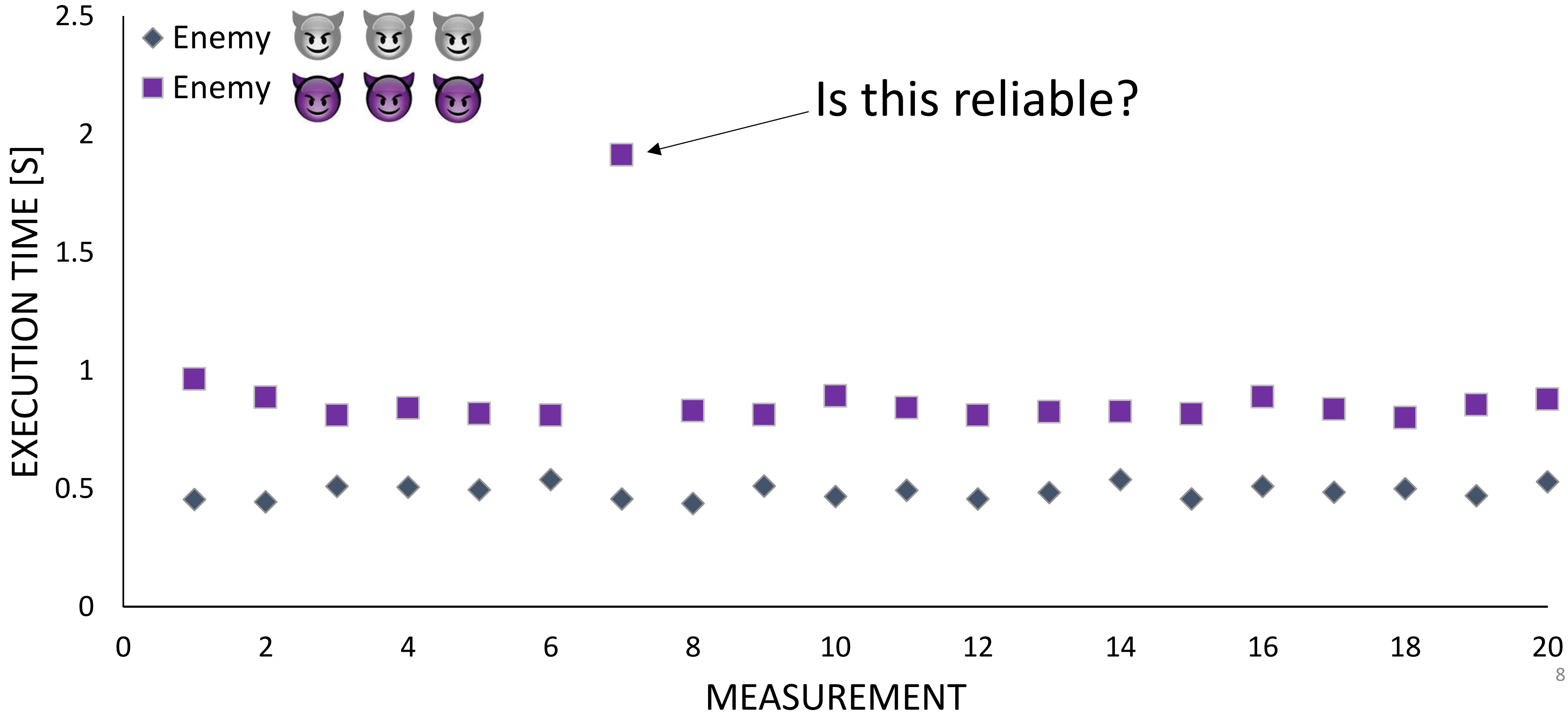




Comparing enemies



Comparing enemies



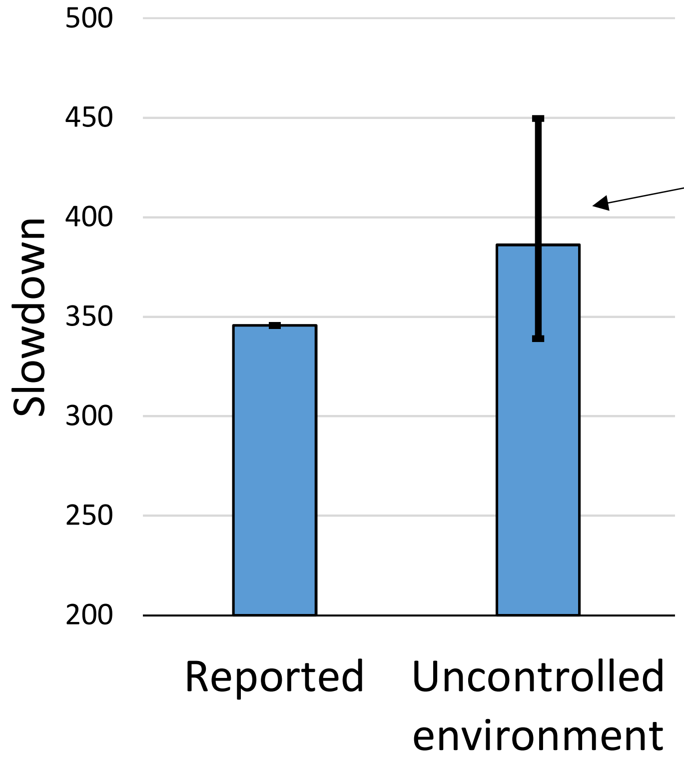
Outline

Reproducible Measurements

Uncovering aggressive configurations

Results

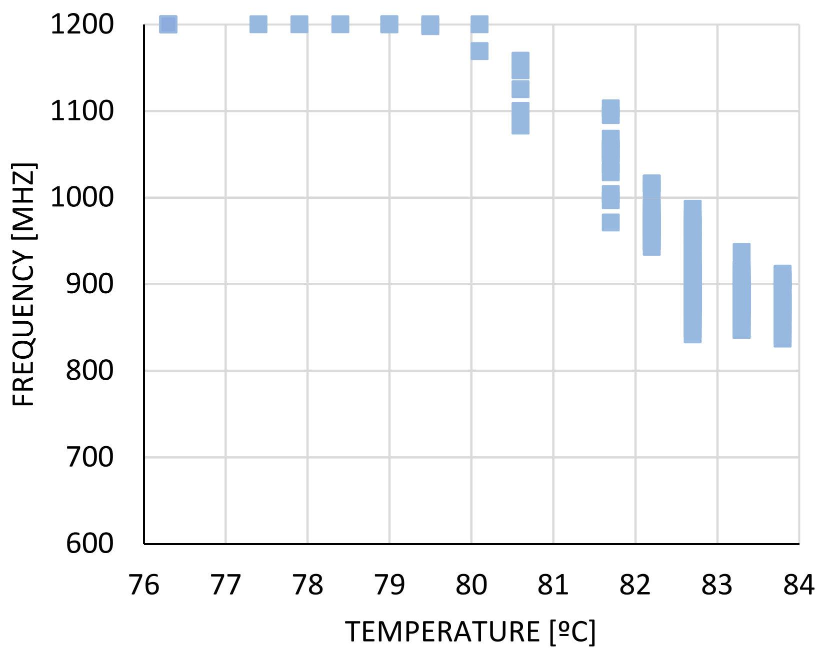
Slowdowns



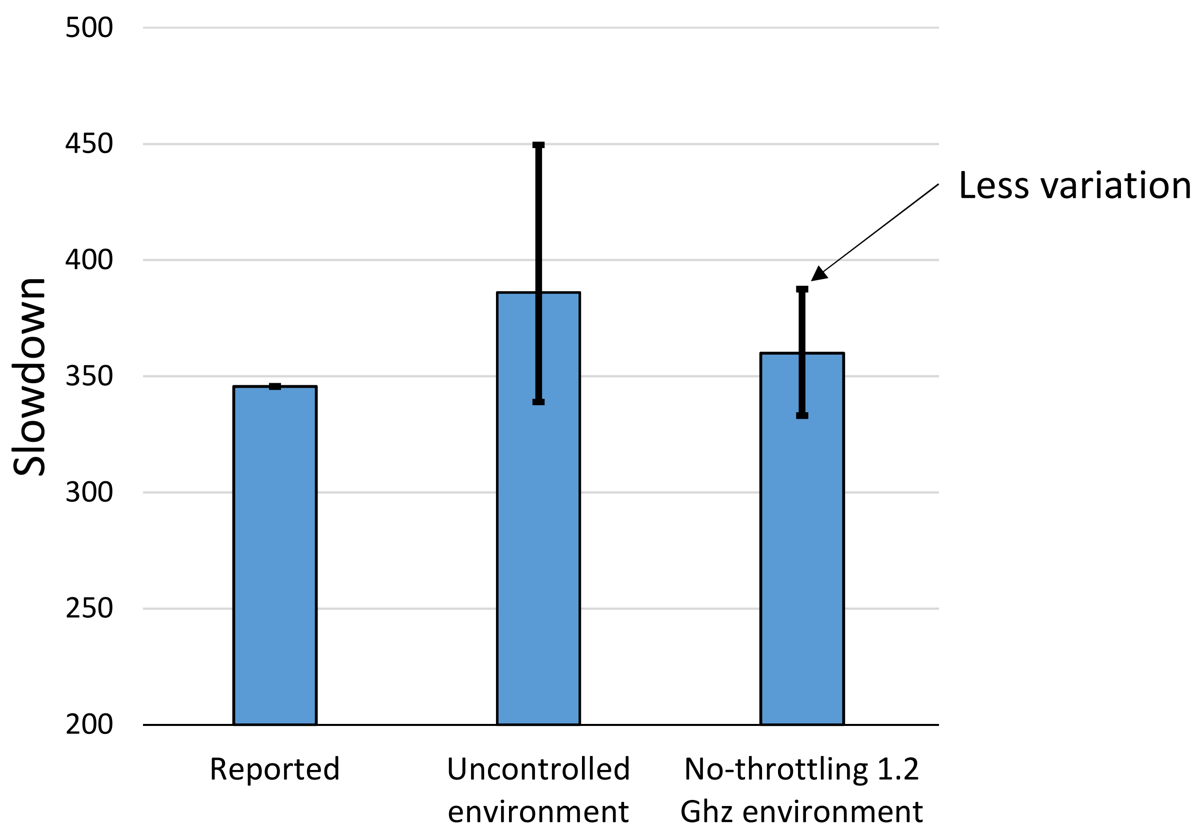
High variation of maximum observed value

Effect of temperature

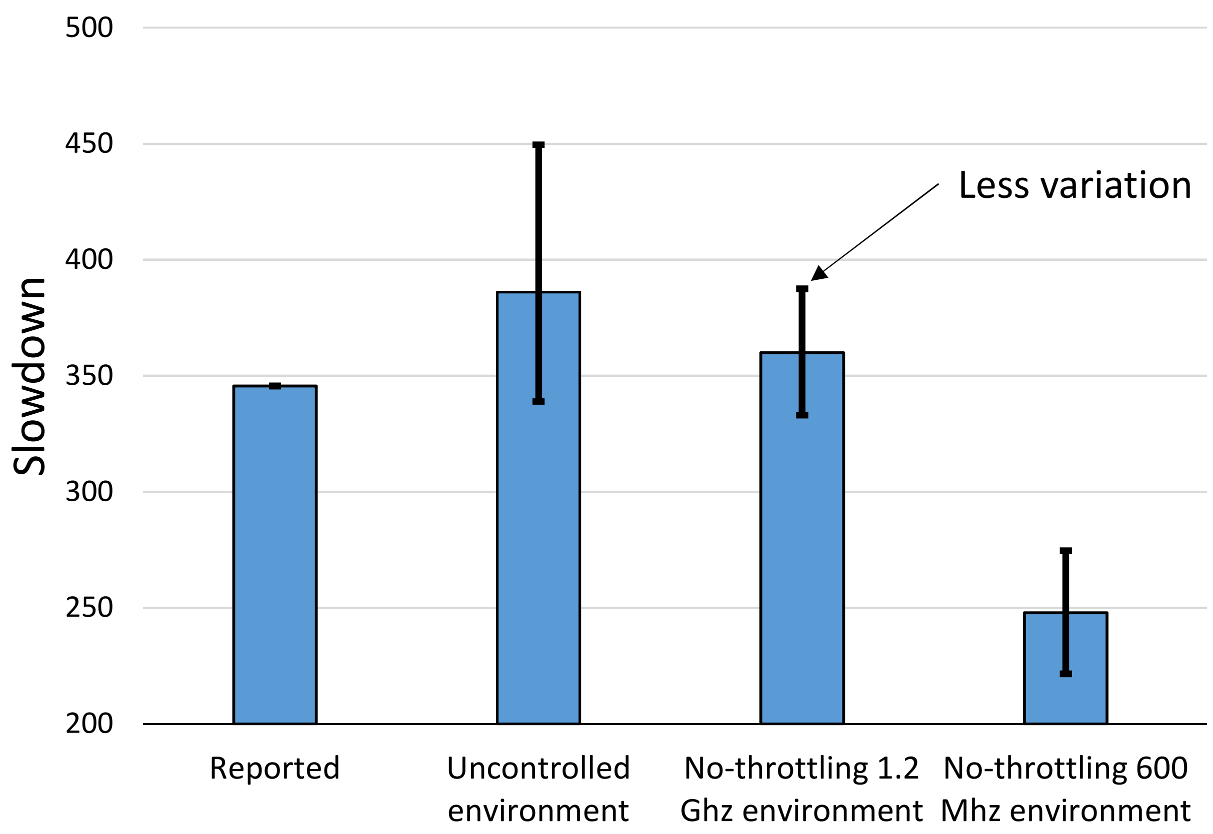
- Frequency throttling significantly impacts measurements
- Discard all measurements taken at more than 80 ºC



Slowdowns


500
450
400
350
300
250
200
Slowdown
Less variation
Reported
Uncontrolled environment
No-throttling 1.2 Ghz environment

Slowdowns



Less variation
Slowdown
500
450
400
350
300
250
200
Reported
Uncontrolled environment
No-throttling 1.2 Ghz environment
No-throttling 600 Mhz environment

Other mitigations

Operating system:

- Disallow thread migration
- Run PUT at max priority
- Ensure parallel execution
- Remove unnecessary software

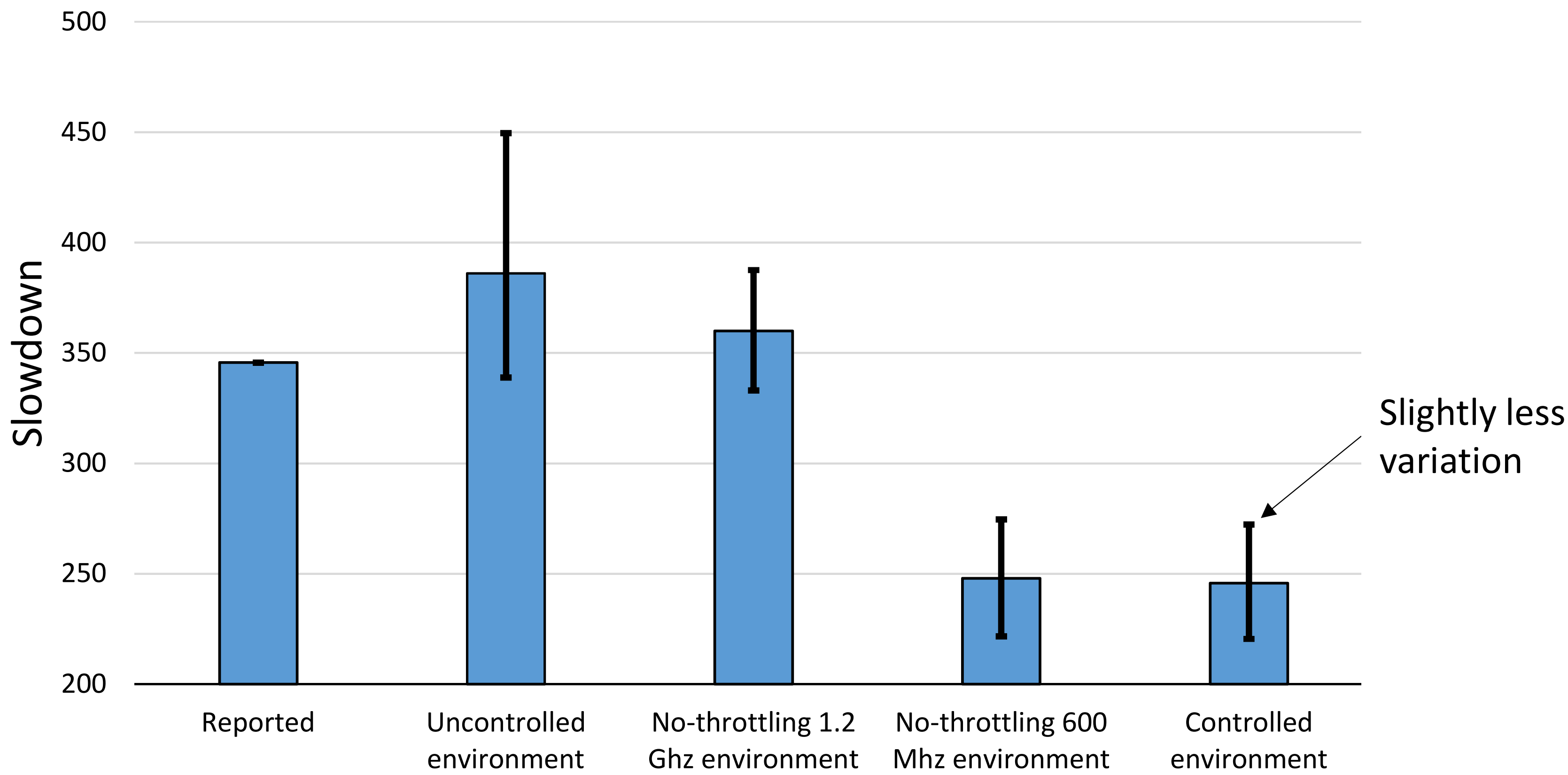
Compiler:

- Disable compiler optimisation for enemy processes

Hardware:

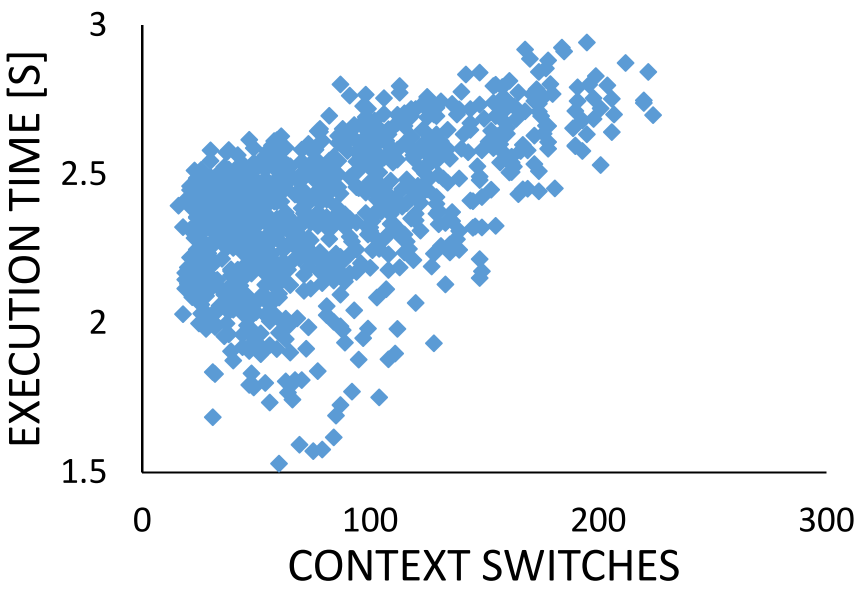
- Flush caches between runs

Slowdowns



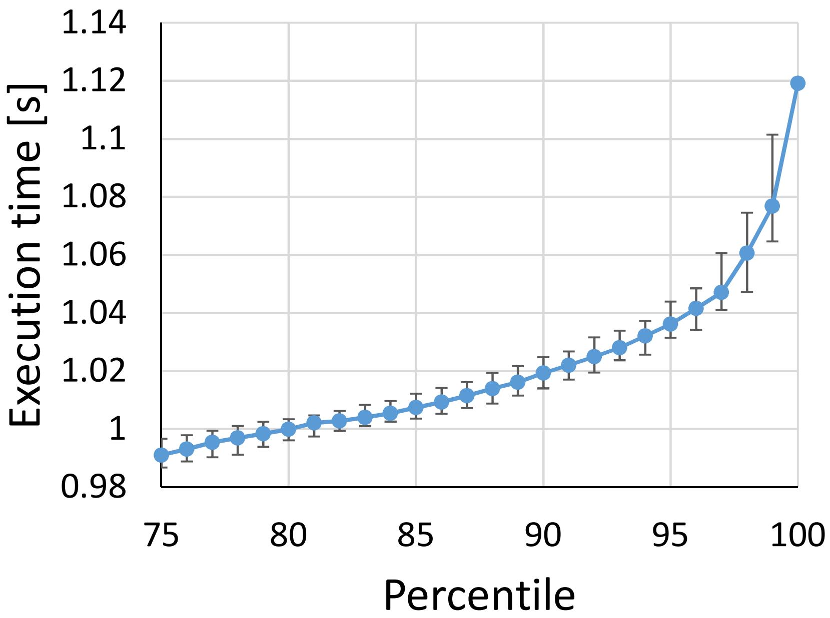
500
450
400
350
300
250
200
Slowdown
Slightly less variation
Reported
Uncontrolled environment
No-throttling 1.2 Ghz environment
No-throttling 600 Mhz environment
Controlled environment

Impact of context switches



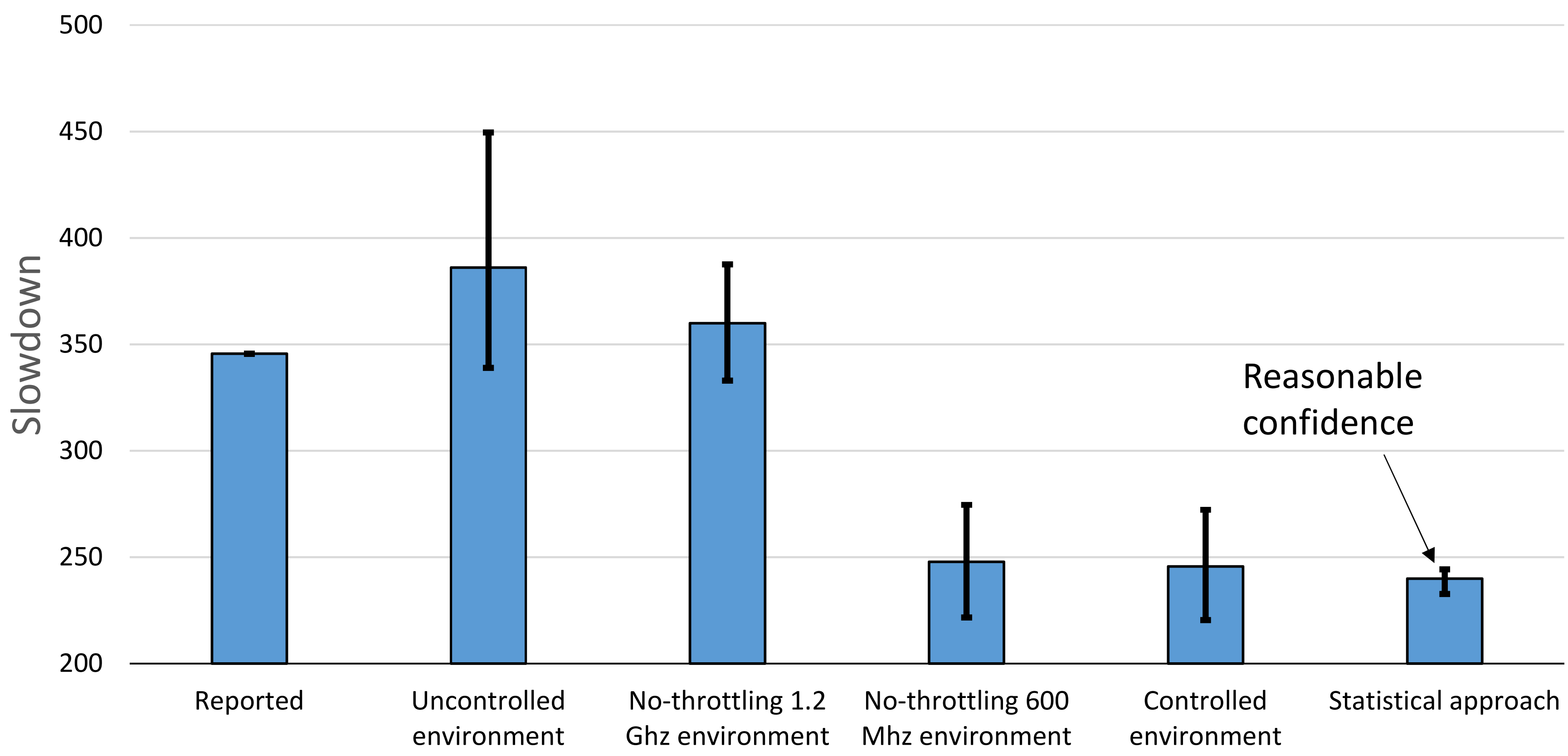
- Context switches still affect the execution time
- There is a linear correlation between execution time and the number of context switches

Statistical approach



- The maximum observed value is often unreliable
- We choose the 90th percentile instead

Slowdowns



500
450
400
350
300
250
200
Slowdown
Reasonable confidence
Reported
Uncontrolled environment
No-throttling 1.2 Ghz environment
No-throttling 600 Mhz environment
Controlled environment
Statistical approach

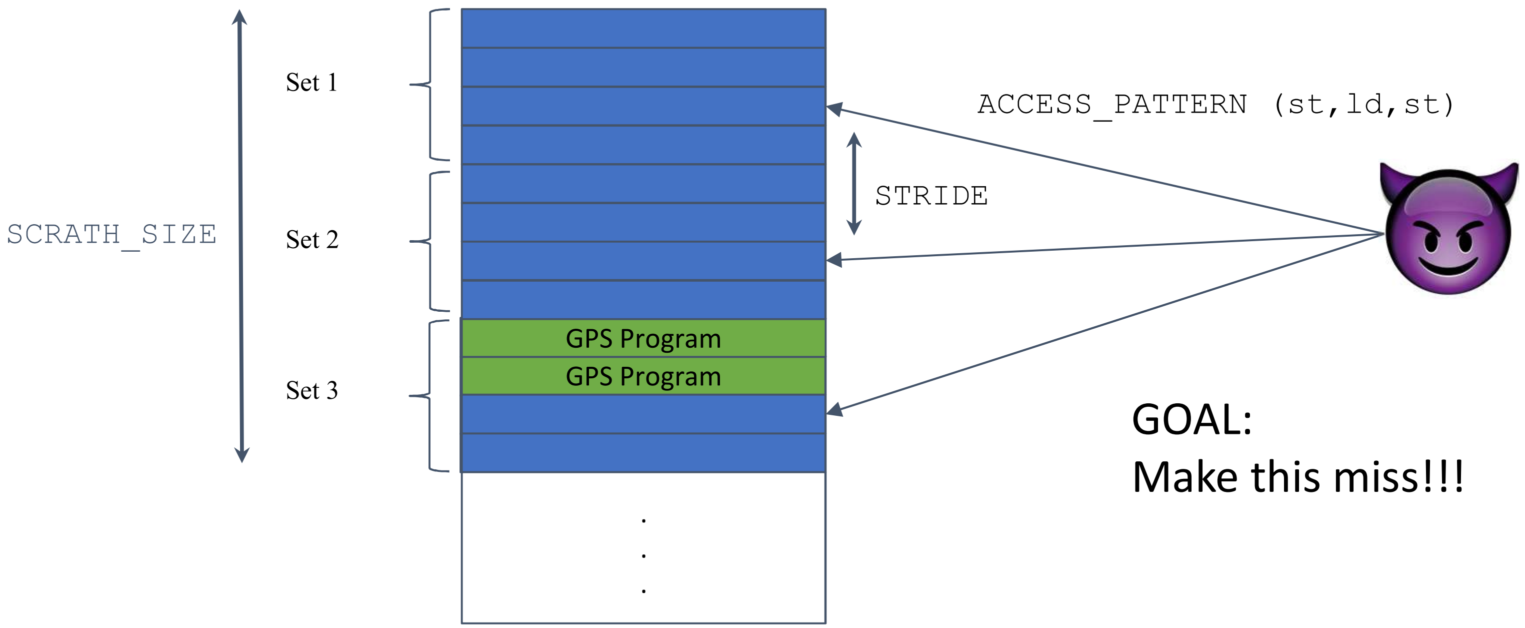
Outline

Reproducible Measurements

Uncovering aggressive configurations

Results

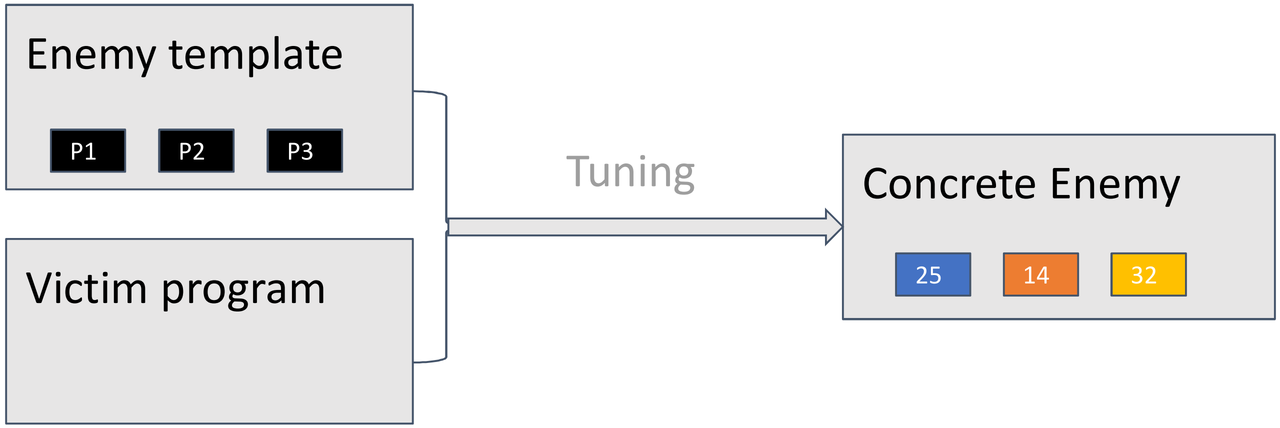
Interference



4-way set associative cache

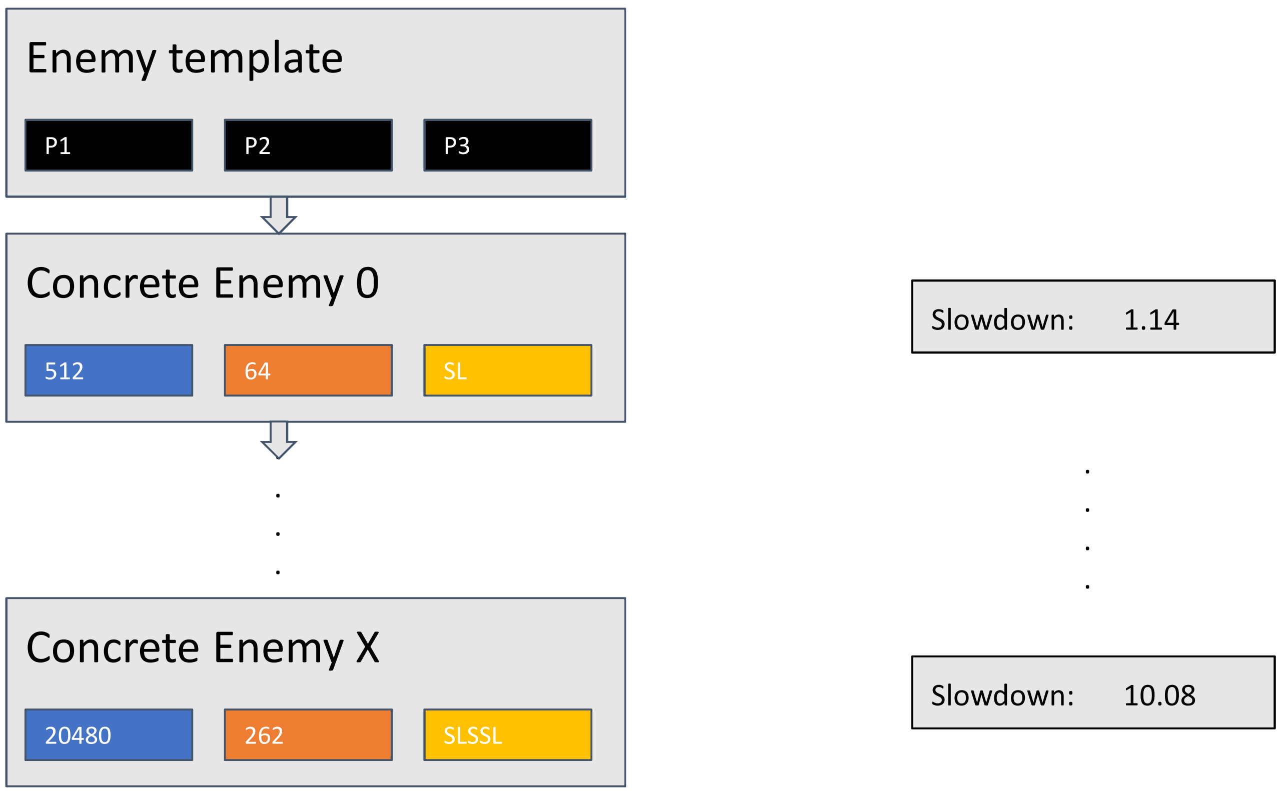
GOAL:
Make this miss!!!

Enemy tuning

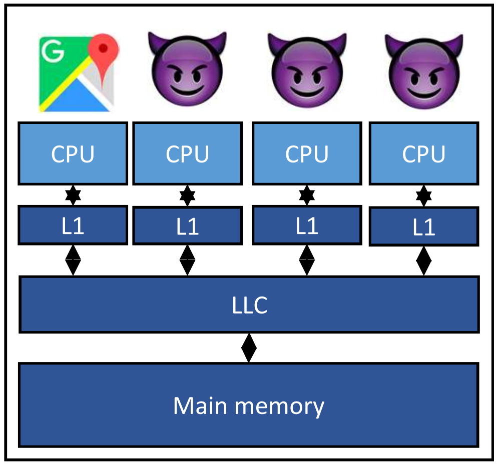


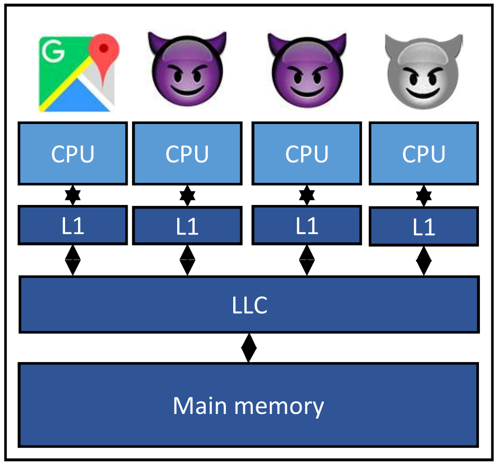
- Template enemies for each shared resource
- Victim programs for each enemy
- We tune the enemy programs to cause maximum interference

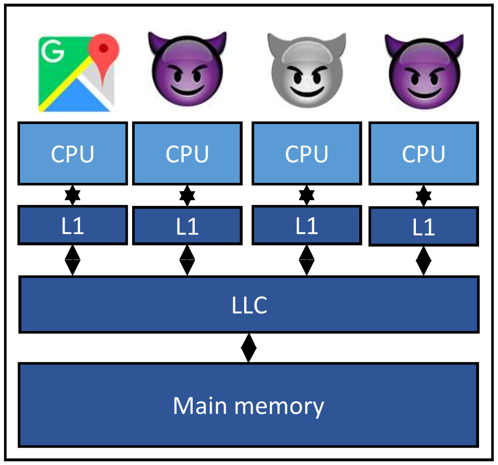
Enemy tuning



Selecting optimal configuration







… (+ 5 more)

Enemy templates

```
#define ACCESS_PATTERN(*scratch_addr) ...
volatile int8_t *scratch = (int8_t*) malloc(SCRATCH_SIZE);
for (HEADER)
  for (int i = 0; i += STRIDE; i < SCRATCH_SIZE)
    ACCESS_PATTERN(&(scratch[i]));
```

Parameter	Enemy range	Cache victim	Memory victim
SCRATCH_SIZE	1-5120KB	LLC	10 x LLC
STRIDE	1-20	cache line size	Cache line size
ACCESS_PATTERN	1-5 read/writes	read, write	read, write

Outline

Reproducible Measurements

Uncovering aggressive configurations

Results

Experimental setup

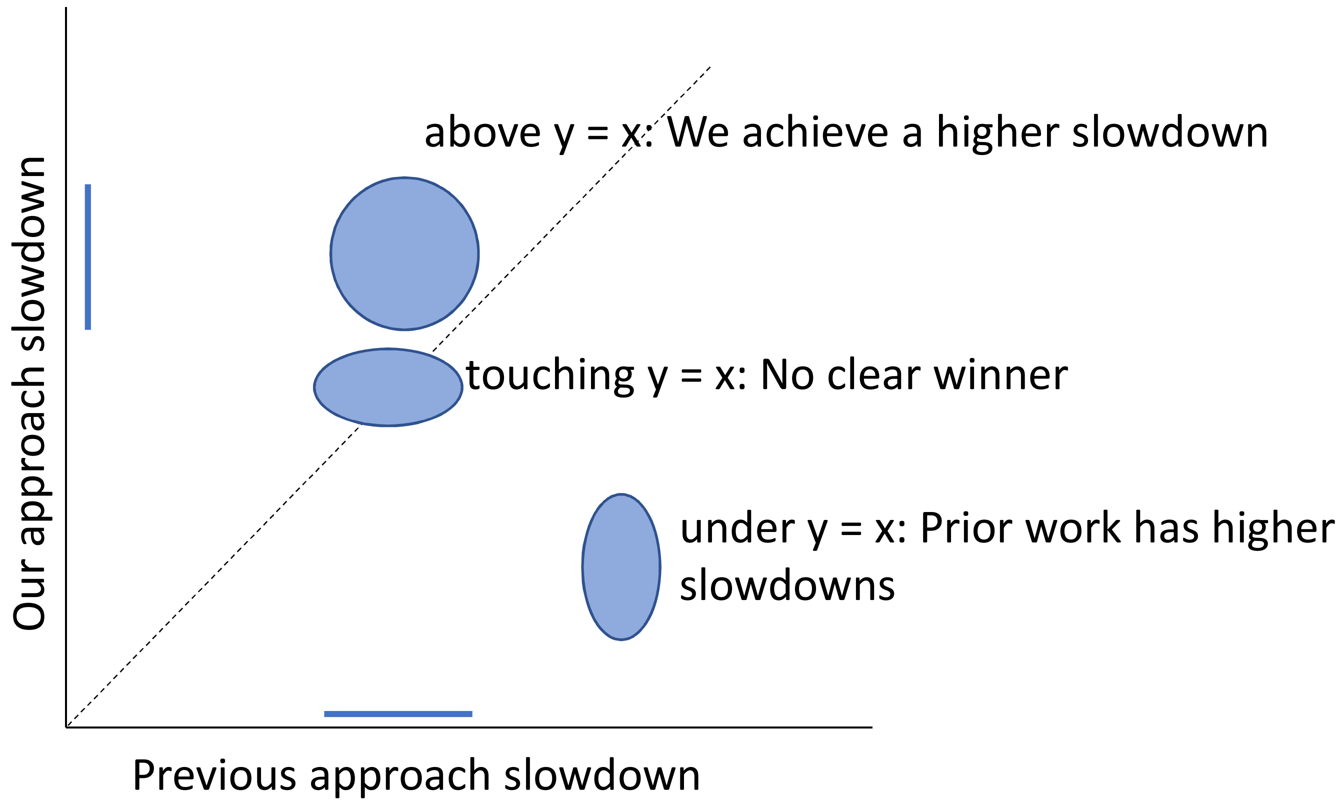
Name	SoC	Arch	Cores
Raspberry Pi 3 B	BCM2837	ARM A53	4
DragonBoard 410c	Adreno306	ARM A53	4
Intel Joule 570x	570x	Atom x86	4
Nano-PC T3	S5P6818	Arm A53	8
BananaPi M3	A837	ARM A7	8

- We experiment on both ARM and Intel architectures
- We use as benchmarks coremark and autobench

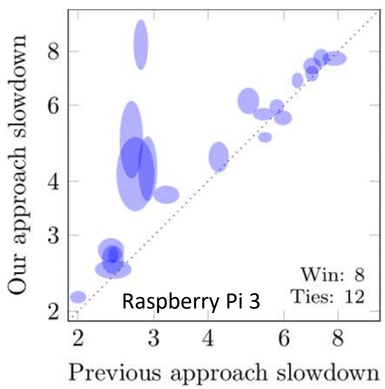
Hostile environments

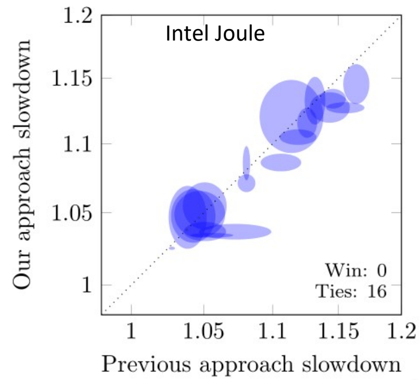
Board	Most aggressive hostile environment, per resource		
	Cache	Main	Overall
Raspberry Pi 3 B	VCCC	VMMM	VMMM
DragonBoard 410c	VMCM	VCMM	VCMM
Intel Joule 570x	VMMC	VMMC	VMMC
Nano-PC T3	VMCM MCMM	VMCC CMCM	CMCM MCMM
BananaPi M3	VCCC CMCC	VCMC MCMC	VCCC CMCC

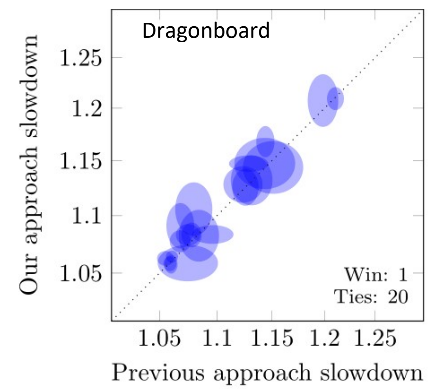
Results Visualization

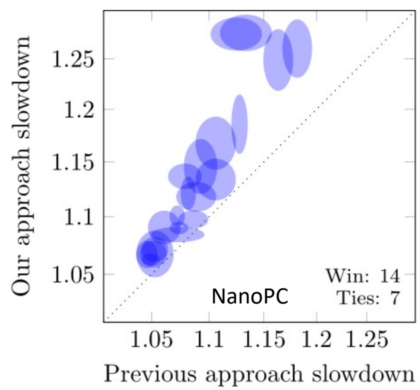


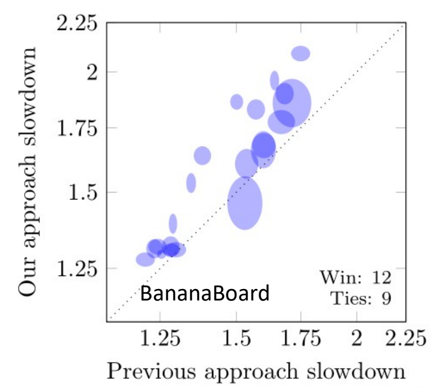
Results


Raspberry Pi 3
Win: 8
Ties: 12
Our approach slowdown
Previous approach slowdown
2 3 4 6 8
2 3 4 6 8


Intel Joule
Win: 0
Ties: 16
Our approach slowdown
Previous approach slowdown
1 1.05 1.1 1.15 1.2
1 1.05 1.1 1.15 1.2


Dragonboard
Win: 1
Ties: 20
Our approach slowdown
Previous approach slowdown
1.05 1.1 1.15 1.2 1.25
1.05 1.1 1.15 1.2 1.25


NanoPC
Win: 14
Ties: 7
Our approach slowdown
Previous approach slowdown
1.05 1.1 1.15 1.2 1.25
1.05 1.1 1.15 1.2 1.25


BananaBoard
Win: 12
Ties: 9
Our approach slowdown
Previous approach slowdown
1.25 1.5 1.75 2 2.25
1.25 1.5 1.75 2 2.25

Graveyard of enemies

Shared resource targeted	Operation performed
Bus	Transfers between the CPU and RAM
Memory thrashing	Random writes to RAM
Pipeline	Arithmetic operations
System	I/O operations

Conclusions

- Reproducible measurements can be obtained using a two-pronged approach: system interference mitigation and a percentile-based metric
- Enemy programs can be precisely compared, and thus tuned using our reproducible metrics
- Tuning can uncover higher slowdowns - achieves a statistically larger slowdown compared to prior work in 35 out of 105 benchmark/chip combinations

Try it out!
https://github.com/mc-imperial/multicore-test-harness

Dan Iorga,
Imperial College London

Imperial College London

Funded in part by DSTL grant R1000115750