VM Warmup Blows Hot and Cold





Edd Barrett

Carl Friedrich Bolz

Rebecca Killick (Lancaster)

Sarah Mount

Laurence Tratt

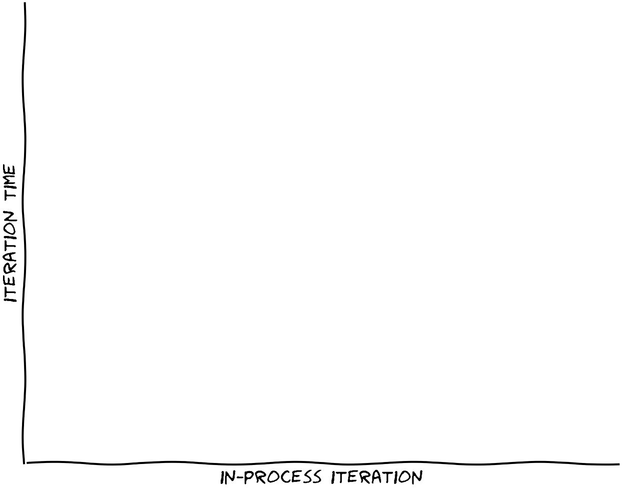
KING'S College LONDON

Software Development Team

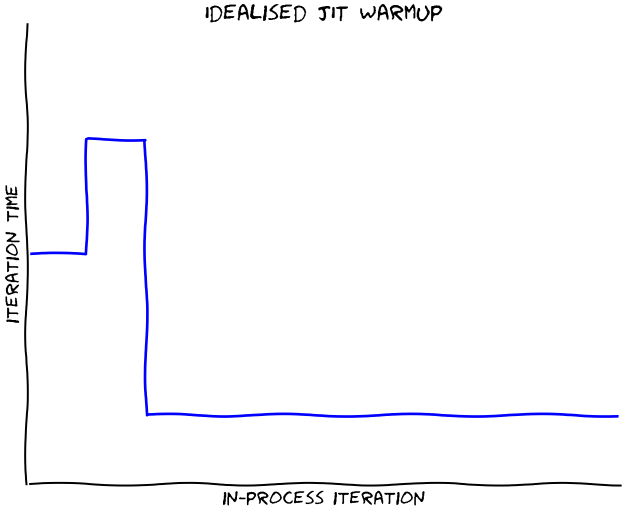
January 25, 2017

What’s VM Warmup?

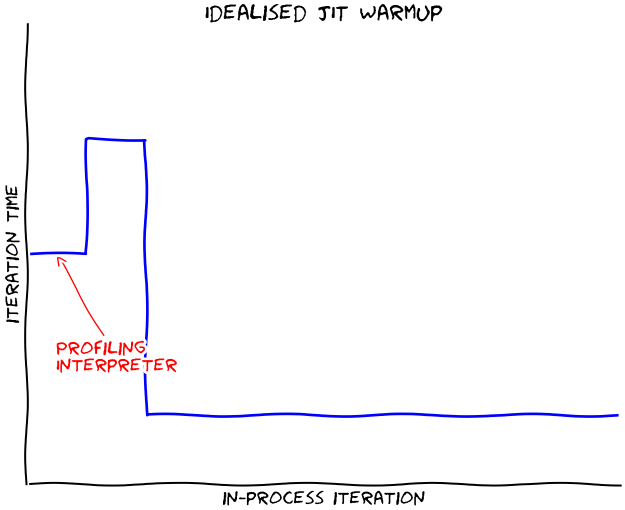
What's VM Warmup?



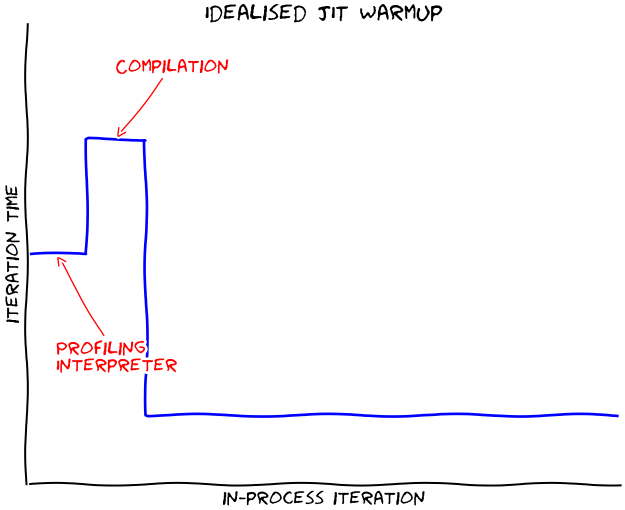
What's VM Warmup?



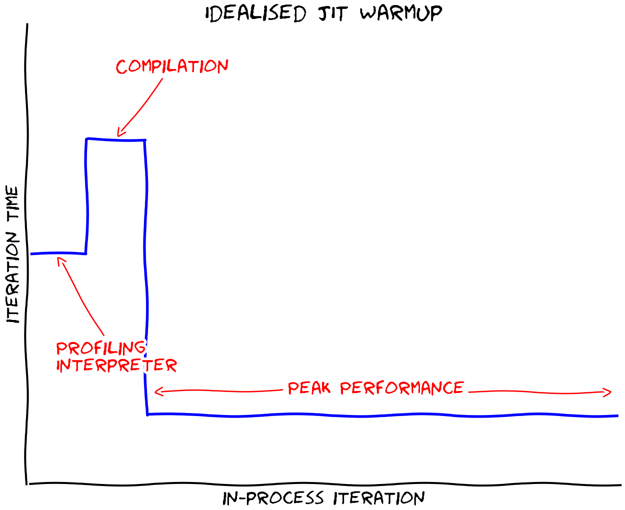
What's VM Warmup?



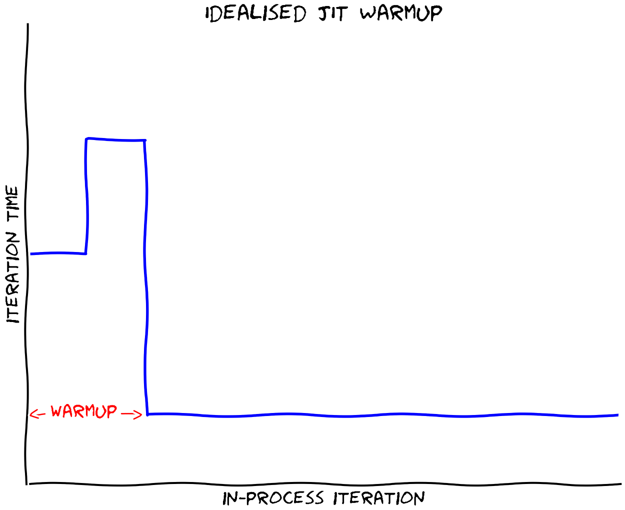
What's VM Warmup?



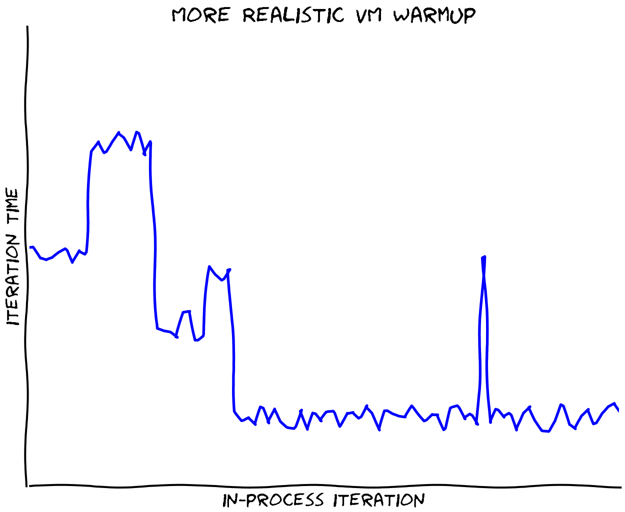
What's VM Warmup?



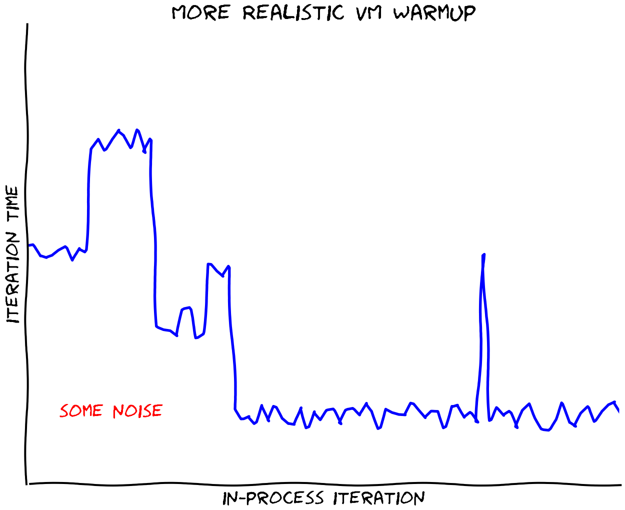
What's VM Warmup?



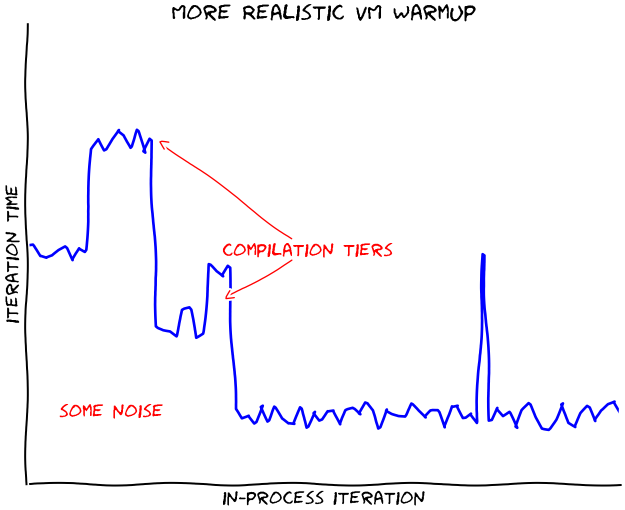
What's VM Warmup?



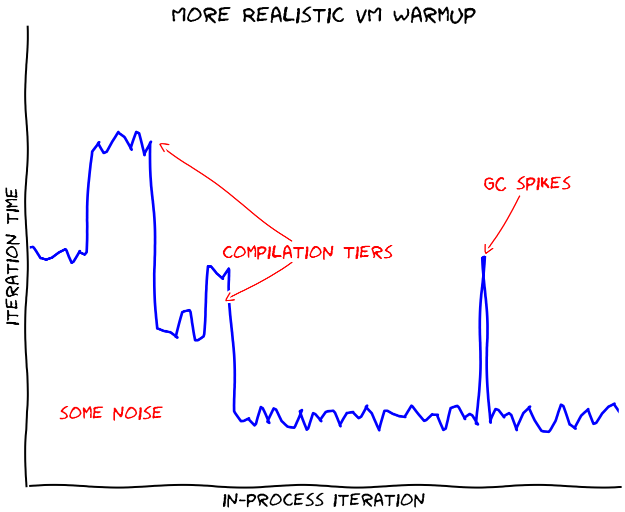
What's VM Warmup?



What's VM Warmup?

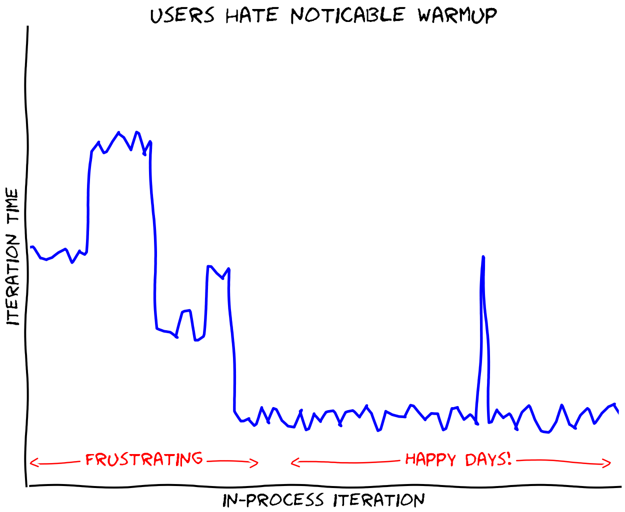


What's VM Warmup?

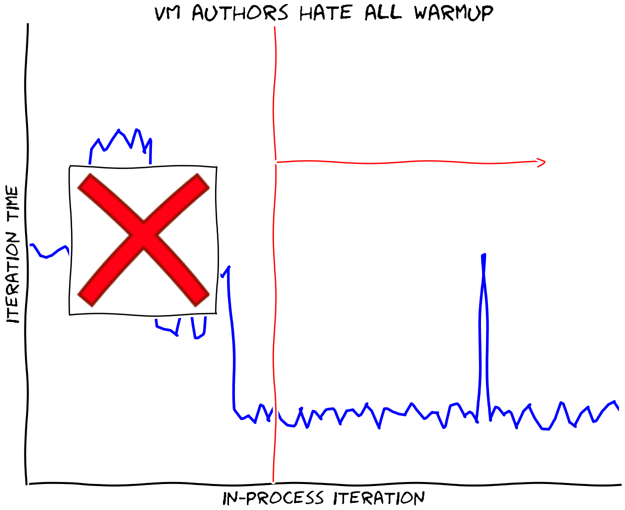


Why care?

What's VM Warmup?



What's VM Warmup?



What’s VM Warmup?

Warmup is bad for everyone.

Our Experiment

Goal:

Measure warmup of modern (JITted) language VMs

The Warmup Experiment

Goal:

Measure warmup of modern (JITted) language VMs

Hypothesis:

Small, deterministic programs exhibit classical warmup behaviour

Which VMs?

VM	Version	Language
Graal	0.18	Java
HHVM	3.15.3	PHP
JRuby/Truffle	graalvm-0.18	Ruby
Hotspot	8u112b15	Java
LuaJit	2.0.4	Lua
PyPy	5.6.0	Python
V8	5.4.500.43	Javascript
GCC	4.9.3	C/C++

Note: same GCC (4.9.3) used for all compilation

Which benchmarks?

The language benchmark games

`https://benchmarksgame.alioth.debian.org/`

(small benchmarks, multi-language, common optimisation target)

Which benchmarks?

The language benchmark games

`https://benchmarksgame.alioth.debian.org/`

(small benchmarks, multi-language, common optimisation target)

- We removed any CFG non-determinism
- We added checksums to all benchmarks

How long to run for?

2000 *in-process iterations* per process execution.

How long to run for?

2000 *in-process iterations* per process execution.

10 *process executions*

Which Machines?

- Linux-Debian8/i4790K, 24GiB RAM
- Linux-Debian8/i4790, 32GiB RAM
- OpenBSD-5.8/i4790, 32GiB RAM

Which Machines?

- Linux-Debian8/i4790K, 24GiB RAM
- Linux-Debian8/i4790, 32GiB RAM
- OpenBSD-5.8/i4790, 32GiB RAM

- Turbo boost and hyper-threading disabled
- Daemons disabled (cron, smtpd)
- Tickless kernel (Linux only)
- Disable Intel P-state driver (Linux only)
- Linux machine software identical.

How to run the Benchmarks?

Benchmark runner: KRUN

`https://github.com/softdevteam/krun`

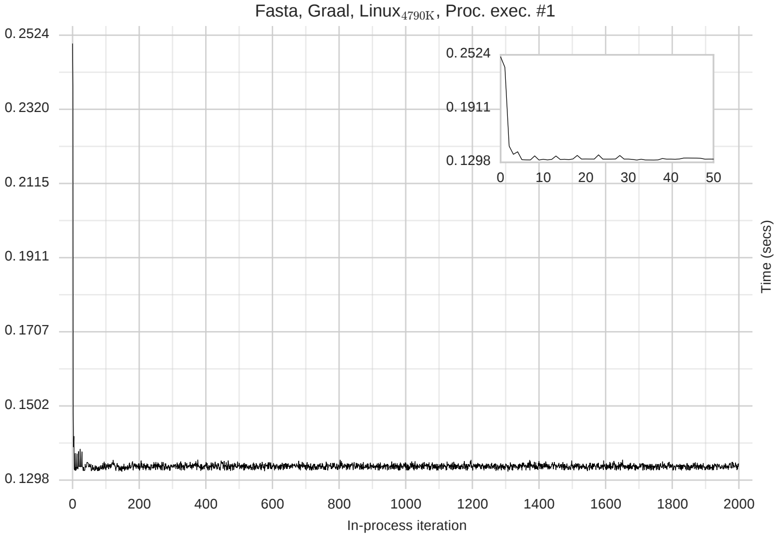
Tries to control confounding variables

How to run the Benchmarks?

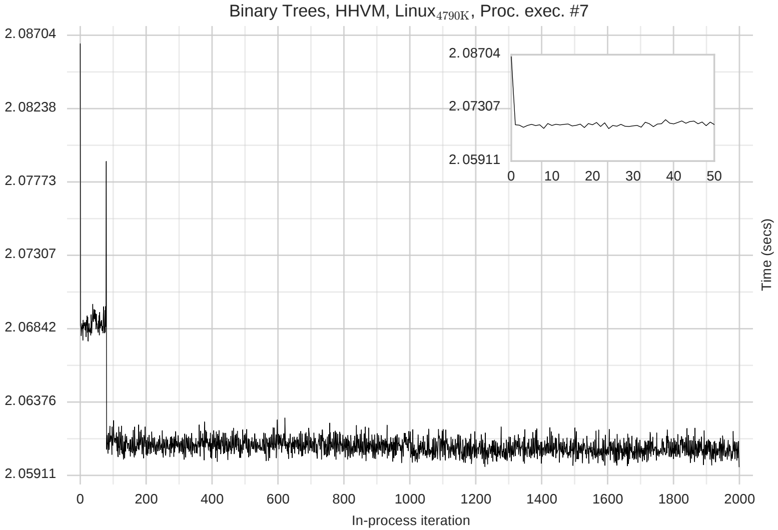
- Network device taken down during benchmarking
- Drops privileges to a fresh user account for each proc. exec
- Automatically reboots the system prior to each proc. exec
- Checks system at (roughly) same temperature for proc. execs
- Minimises I/O
- Sets fixed heap and stack ulimits
- Checks `dmesg` for changes after each proc. exec
- Enforces kernel settings (tickless mode, CPU governors, ...)
- ...

Preliminary Results

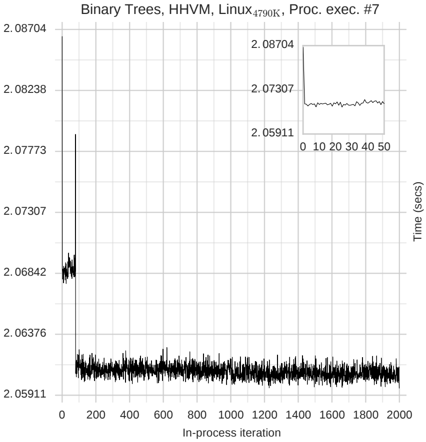
Classical Warmup

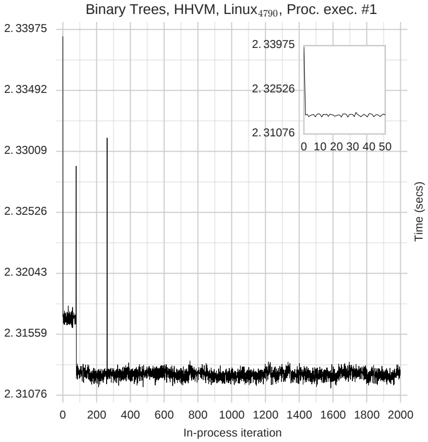


Classical Warmup



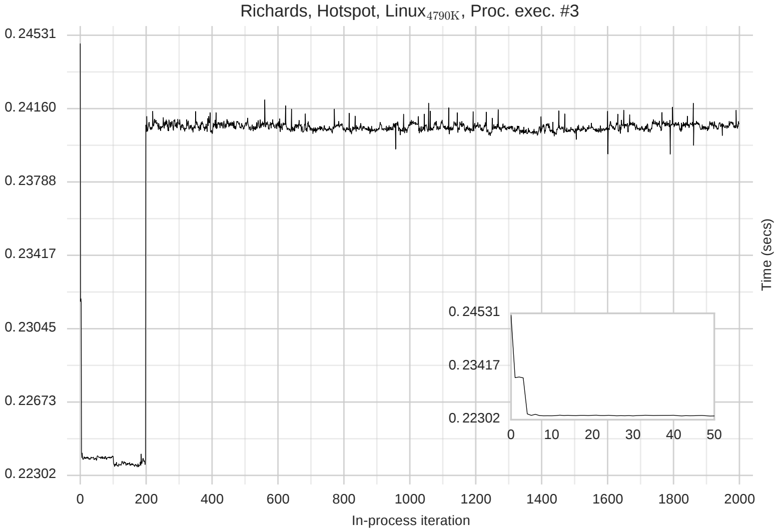
Classical Warmup





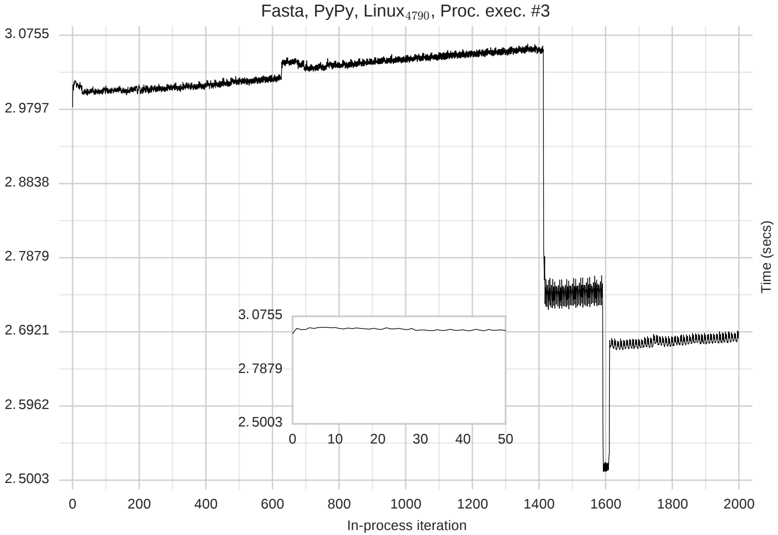
(different machines)

Slowdown

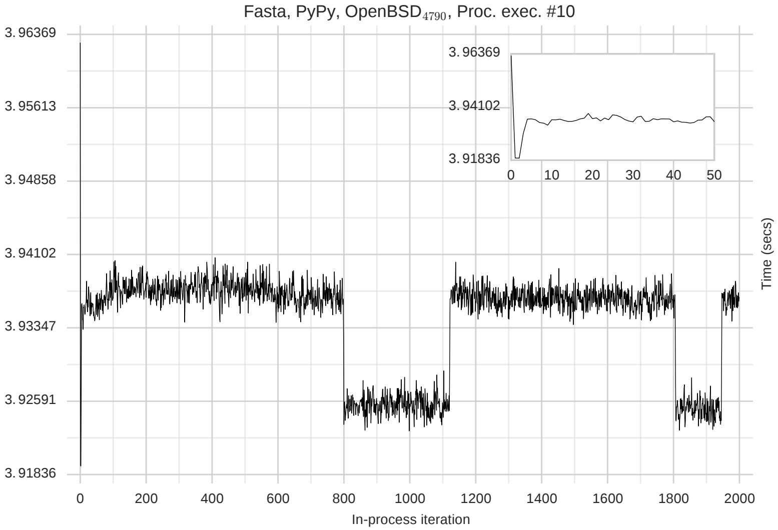


Richards, Hotspot, Linux$_{4790K}$, Proc. exec. #3
0.24531
0.24160
0.23788
0.23417
0.23045
0.22673
0.22302
0
200
400
600
800
1000
1200
1400
1600
1800
2000
In-process iteration
Time (secs)
0.24531
0.23417
0.22302
0
10
20
30
40
50

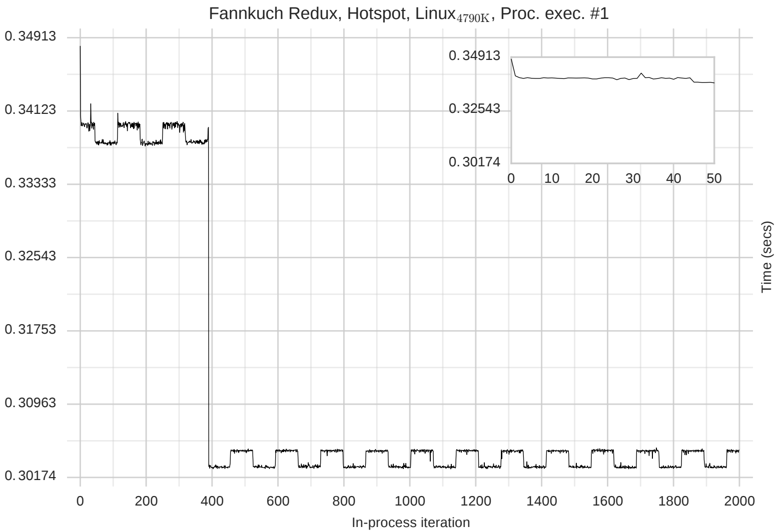
Slowdown


Fasta, PyPy, Linux$_{4790}$, Proc. exec. #3
3.0755
2.9797
2.8838
2.7879
2.6921
2.5962
2.5003
0
200
400
600
800
1000
1200
1400
1600
1800
2000
In-process iteration
Time (secs)
3.0755
2.7879
2.5003
0
10
20
30
40
50

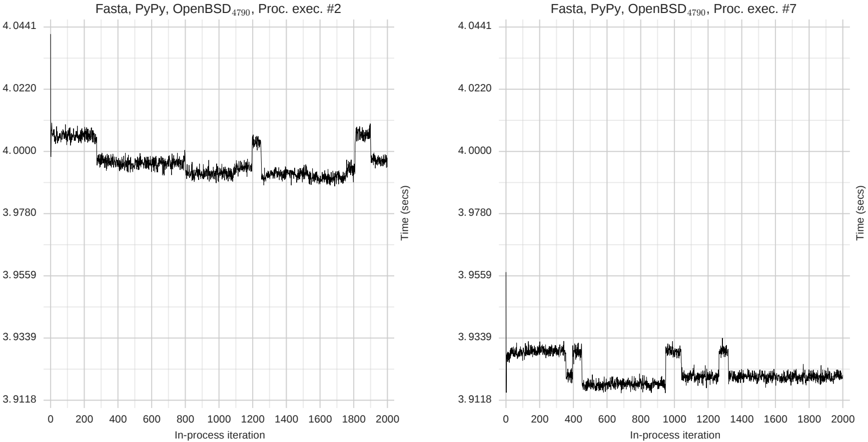
No Steady State



No Steady State

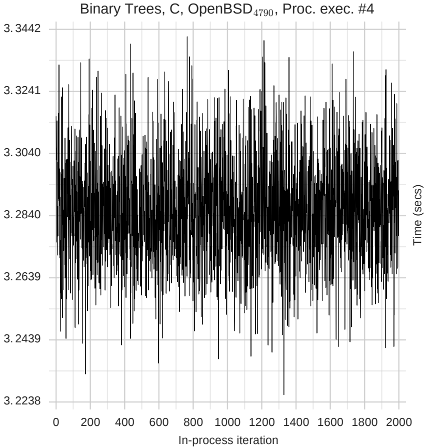


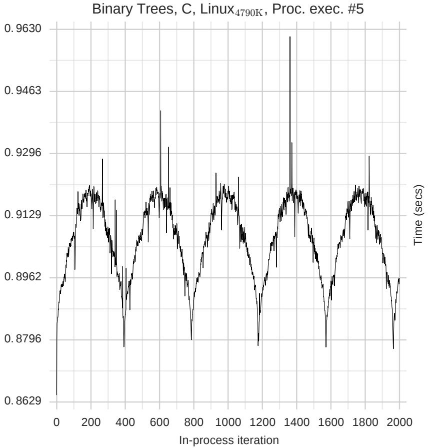
Inconsistent Process-executions



(same machine)

Inconsistent Process-executions

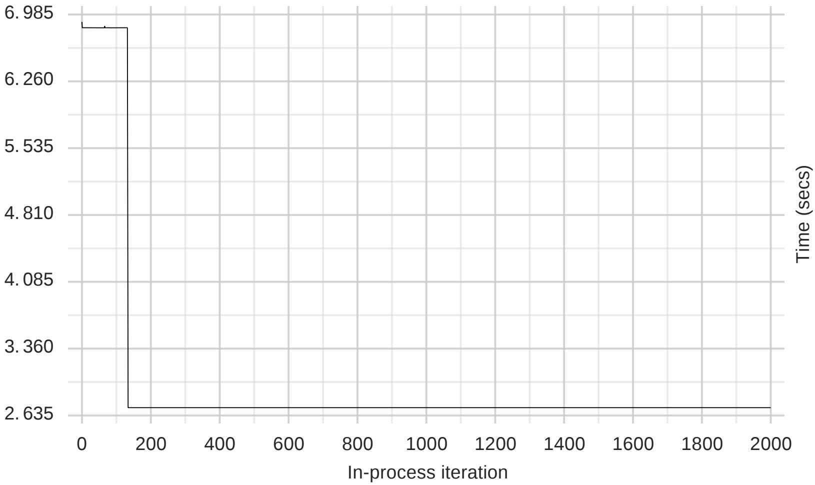




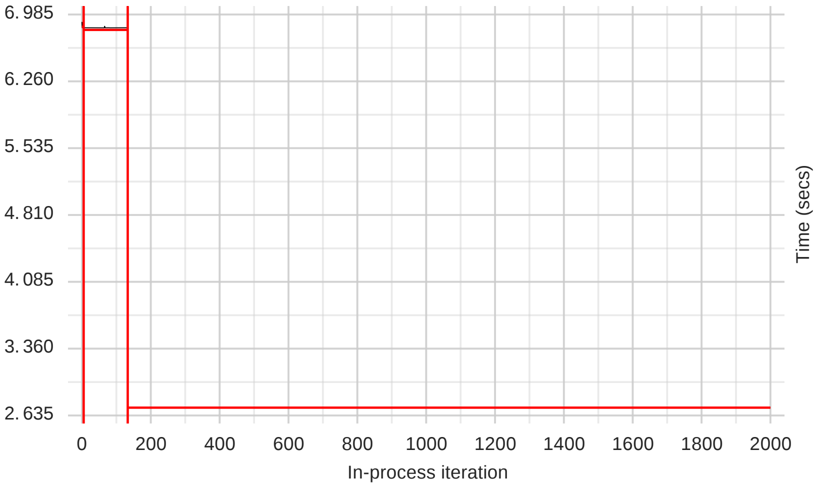
(different machines)

Summarising with Changepoint Analysis

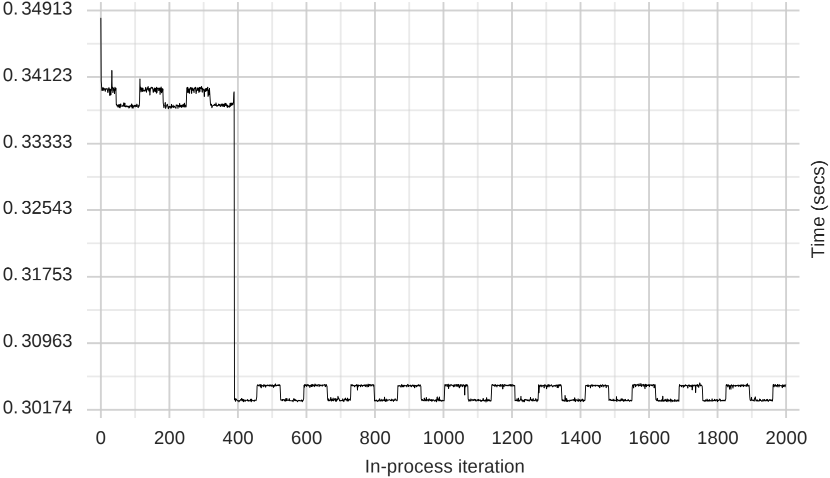
Summarising with Changepoint Analysis



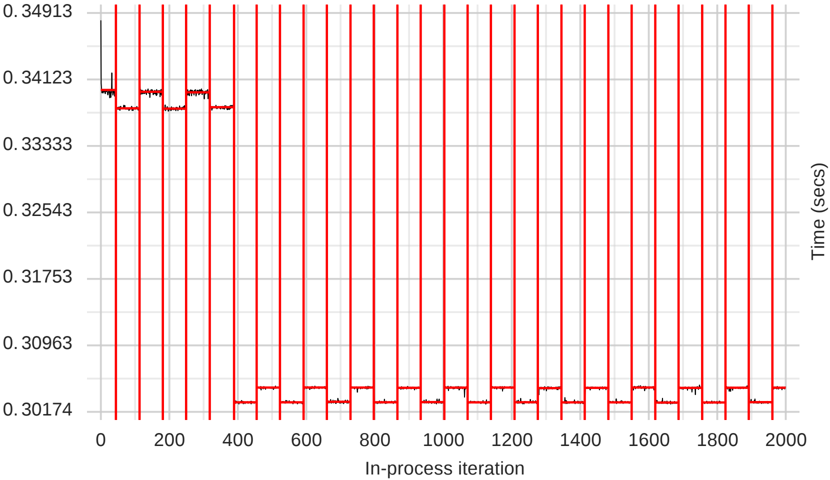
Summarising with Changepoint Analysis



Summarising with Changepoint Analysis



Summarising with Changepoint Analysis



Summarising with Changepoint Analysis

For each ⟨Machine, VM, Benchmark⟩ combo, the process executions can be either:

- — All flat
- ⎦ All warmup
- ⎣ All slowdown
- ʌʌʌ All no steady state
- ≭ Inconsistent ($>$ 1 of the above)

Summarising with Changepoint Analysis

		Class.	Steady iter (#)	Steady iter (s)	Steady perf (s)
C	*binary trees*	⁓			
Graal		⅂	12 ±5	2.78 ±1.047	0.17329 ±0.001097
HHVM		⅂	73 ±11	151.71 ±24.233	2.06641 ±0.002518
HotSpot		⅂	6 ±7	1.12 ±1.296	0.17220 ±0.000601
JRuby+Truffle		≭ (7⅂, 3ʃ)			
LuaJIT		≭ (5−, 4ʃ, 1⅂)			
PyPy		≭ (6⁓, 4ʃ)			
V8		≭ (9⅂, 1ʃ)			
C	*fannkuch redux*	−			0.37885 ±0.000073
Graal		≭ (9⅂, 1ʃ)			
HHVM		⅂	9 ±0	39.64 ±0.003	1.32396 ±0.000415
HotSpot		≭ (9⁓, 1⅂)			
JRuby+Truffle		ʃ	998 ±0	928.46 ±6.095	1.05923 ±0.009511
LuaJIT		−			0.50777 ±0.000048
PyPy		≭ (8−, 2⅂)			
V8		−			0.27360 ±0.000030

Summarising with Changepoint Analysis

How many benchmarks were consistently — (flat) or ⅂ (warmup)?

Machine	% ⟨Bench, VM⟩ pairs
Linux_{4790K}	50.0
Linux_{4790}	56.5
OpenBSD_{4790}	43.3

Hypothesis Invalidated

~~*Hypothesis*: Small, deterministic programs exhibit classical warmup behaviour~~

Problems

How can we measure anything any more?

Standard methods assume:

- Convergence upon steady state
- Normally distributed samples

Problems

Are our experiments repeatable?

Many inconsistent process executions

Understanding Why

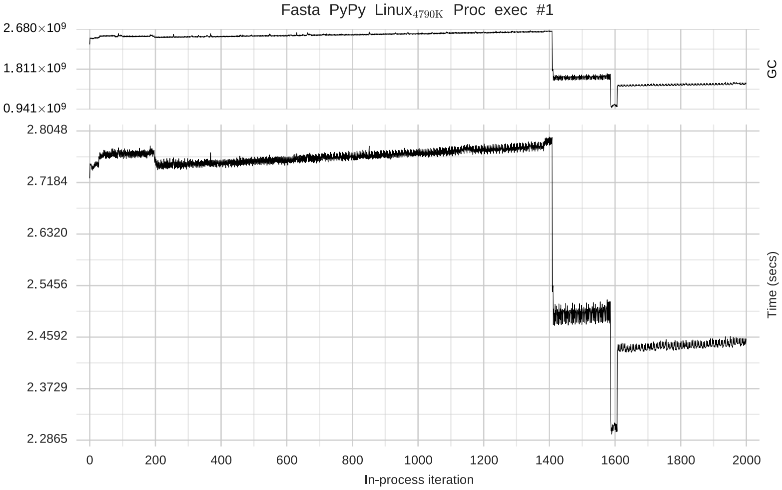
For PyPy and HotSpot we measured (per in-process iteration):

Time spent in the GC

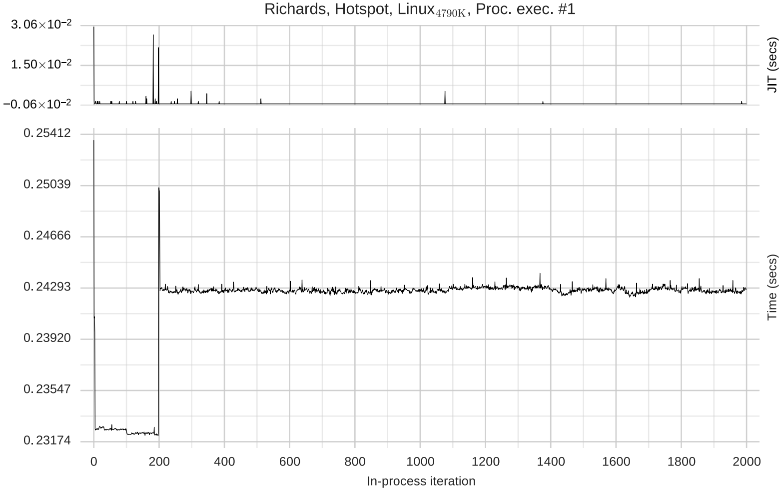
Time spent in the JIT

Can this help to explain why?

GC Instrumentation



JIT Instrumentation



But...

In many cases we can’t explain

(Future work to find out?)

Conclusions

Conclusions

- Only $\simeq$50% of $\langle$Machine, VM, Benchmark$\rangle$ combos consistently warm up OK.
  - (−, ⎤_).
- We can't assume that benchmarks will warmup or stabilise
  - (_⎡, ⩚⩚)
- We can't assume that benchmarking is repeatable
  - Even on the same machine, same installation, ...
  - ($\not\asymp$)

Downloads

Code

`https://github.com/softdevteam/krun/`

`https://github.com/softdevteam/warmup_experiment/`

Data

`https://archive.org/download/softdev_warmup_experiment_artefacts/v0.7/`

References

VM Warmup Blows Hot and Cold
E. Barrett, C. F. Bolz, R. Killick, V. Knight, S. Mount and L. Tratt.

Rigorous Benchmarking in Reasonable Time
T. Kalibera and R. Jones

Specialising Dynamic Techniques for Implementing the Ruby Programming Language
C. Seaton (Chapter 4)

Quantifying performance changes with effect size confidence intervals
T. Kalibera and R. Jones

Thanks for listening