VM Warmup Blows Hot and Cold

Edd Barrett, Carl Friedrich Bolz, Rebecca Killick (Lancaster), Vincent Knight (Cardiff), Sarah Mount, Laurence Tratt

Software Development Team
2016-09-27
What’s ‘Warmup’?
With a JIT

IDEALISED JIT WARMUP

ITERATION TIME

IN-PROCESS ITERATION

http://soft-dev.org/
With a JIT

**Idealised JIT Warmup**

- **Iteration Time**
- **In-Process Iteration**

- **Profiling Interpreter**
With a JIT

IDEALISED JIT WARMUP

ITERATION TIME

IN-PROCESS ITERATION

 compilation

 profiling interpreter
With a JIT
With a JIT

IDEALISED JIT WARMUP

ITERATION TIME

← WARMUP →

IN-PROCESS ITERATION
MORE REALISTIC VM WARMUP

ITERATION TIME

IN-PROCESS ITERATION
MORE REALISTIC VM WARMUP

ITERATION TIME

IN-PROCESS ITERATION

SOME NOISE
MORE REALISTIC VM WARMUP

ITERATION TIME

IN-PROCESS ITERATION

COMPILATION TIERS

SOME NOISE
MORE REALISTIC VM WARMUP

ITERATION TIME

IN-PROCESS ITERATION

SOME NOISE

COMPILATION TIERS

GC SPIKES
Why care?
 USERS HATE NOTICABLE WARMUP

ITERATION TIME

FRUSTRATING  HAPPY DAYS!

IN-PROCESS ITERATION
VM AUTHORS HATE ALL WARMUP
Warmup is bad for everyone.
Measure warmup of modern language implementations
Measure warmup of modern language implementations

**Hypothesis:** Small, deterministic programs exhibit classical warmup behaviour
Method 1: Which benchmarks?

The language benchmark games are perfect for us (unusually)
Method 1: Which benchmarks?

The language benchmark games are perfect for us (unusually)

We removed any CFG non-determinism
The language benchmark games are perfect for us (unusually)

We removed any CFG non-determinism

We added checksums to all benchmarks
Method 2: How long to run?

2000 *in-process iterations*
Method 2: How long to run?

2000 \textit{in-process iterations}

10 \textit{process executions}
Method 3: VMs

- Graal-0.13
- HHVM-3.12.0
- JRuby/Truffle (git #f82ac771)
- Hotspot-8u72b15
- LuaJit-2.0.4
- PyPy-4.0.1
- V8-4.9.385.21
- GCC-4.9.3

Note: same GCC (4.9.3) used for all compilation
Method 4: Machines

- Linux-Debian8/i4790K, 24GiB RAM
- Linux-Debian8/i4790, 32GiB RAM
- OpenBSD-5.8/i4790, 32GiB RAM
Method 4: Machines

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

- Turbo boost and hyper-threading disabled
- SSH blocked from non-local machines
- Daemons disabled (cron, smtpd)
Method 5: Krun

Benchmark runner: tries to control as many confounding variables as possible
Method 5: Krun

Benchmark runner: tries to control as many confounding variables as possible e.g.:

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

13/32 HTTP://SOFT-DEV.ORG/
Preliminary results
Classical Warmup

Richards, Graal, Linux1/i7-4790K, Process execution #3

In-process iteration

Time(s)

0.232
0.341
0.449
0.558
0.666
0.775
0.884

0 200 400 600 800 1000 1200 1400 1600 1800 2000

0.232
0.558
0.884

15/32 HTTP://SOFT-DEV.ORG/
Classical Warmup

Spectral Norm, PyPy, Linux1/i7-4790K, Process execution #7

In-process iteration vs. Time (s)
(Different machines)
Richards, Hotspot, Linux2/i7-4790, Process execution #2
Cycles

Fannkuch Redux, Hotspot, OpenBSD/i7-4790, Process execution #4

Time(s)

In-process iteration

0 200 400 600 800 1000 1200 1400 1600 1800 2000

0.358 0.366 0.374 0.382 0.389 0.397 0.405

250 300 350 400 450 500 550 600

0.359 0.369 0.372 0.375 0.377 0.378 0.379 0.380 0.381 0.382 0.383 0.384 0.385 0.386

http://soft-dev.org/
Never-ending Phase Changes

Fasta, LuaJIT, OpenBSD/i7-4790, Process execution #5

In-process iteration

Time(s)

0 200 400 600 800 1000 1200 1400 1600 1800 2000
Inconsistent Process-executions

(Note: same machine)
Inconsistent Process-executions

(Note: different machines. Bouncing ball pattern Linux-specific)
Classical warmup occurs for only:
Classical warmup occurs for only:

50% of process executions
Summary

Classical warmup occurs for only:

50% of process executions

25% of (VM, benchmark) pairs
Classical warmup occurs for only:

- 50% of process executions
- 25% of (VM, benchmark) pairs
- 0% of benchmarks for all VMs, machines & proc execs.
**Hypothesis**

Small, deterministic programs exhibit classical warmup behaviour.
Open Questions

CAN WE MEASURE ANYTHING ANYMORE?

OR HOW LONG HAS THIS BEEN GOING ON?

IS THIS REALLY THE FAULT OF THE 6-S?

HTTP://SOFT-DEV.ORG/
How can we measure anything any more?
Open Questions

How can we measure anything any more?

For how long has this been going on?
Open Questions

How can we measure anything any more?

For how long has this been going on?

Is this really the fault of the VMs?
Ongoing/Future Work

(a.k.a. “making sense of our results”)
Performance Counters

- **CPU_CLK_UNHALTED.CORE**
  - Counts the number of “core cycles” executed per-core.

- **IA32_APERF / IA32_MPERF ratio.**
  - **IA32_MPERF** increments at a fixed reference frequency.
  - **IA32_APERF** increments proportional to “actual performance”.
  - The ratio of two delta’s indicates if clock speed changed.
Plotting performance counters

Measurement vs. Core cycles vs. aperf/mperf ratio

Fannkuch Redux, PyPy, Linux i7-4790K
Process execution #1

Time(s) vs. In-process iteration

# Core Cycles vs. aperf/mperf ratio

Core cycles
For PyPy and Hotspot, record:

- Time spent in GC.
- Time spent in Compilation.
Outlier Detection

Spectral Norm, PyPy, Linux1/i7-4790K, Process execution #1

Spectral Norm, PyPy, Linux1/i7-4790K, Process execution #2

Measurement
Outliers

28 / 32
http://soft-dev.org/
Outliers outside $5\sigma$ of rolling average
Outlier Detection

Spectral Norm, PyPy, Linux1/i7-4790K, Process execution #1

Spectral Norm, PyPy, Linux1/i7-4790K, Process execution #2

Recurring outliers
Change-point Analysis

fasta:V8:default–javascript, run: 5
Change-point Analysis

fannkuch_redux:Hotspot:default-java , run: 1
Change-point Analysis

binarytrees:PyPy:default--python , run: 1
Full (Preliminary) Results

https://archive.org/download/softdev_warmup_experiment_artefacts/v0.2/

- all_graphs.pdf All plots in one huge PDF.
- warmup_results*.json.bz2 Raw results.

(Note: newer results available)
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