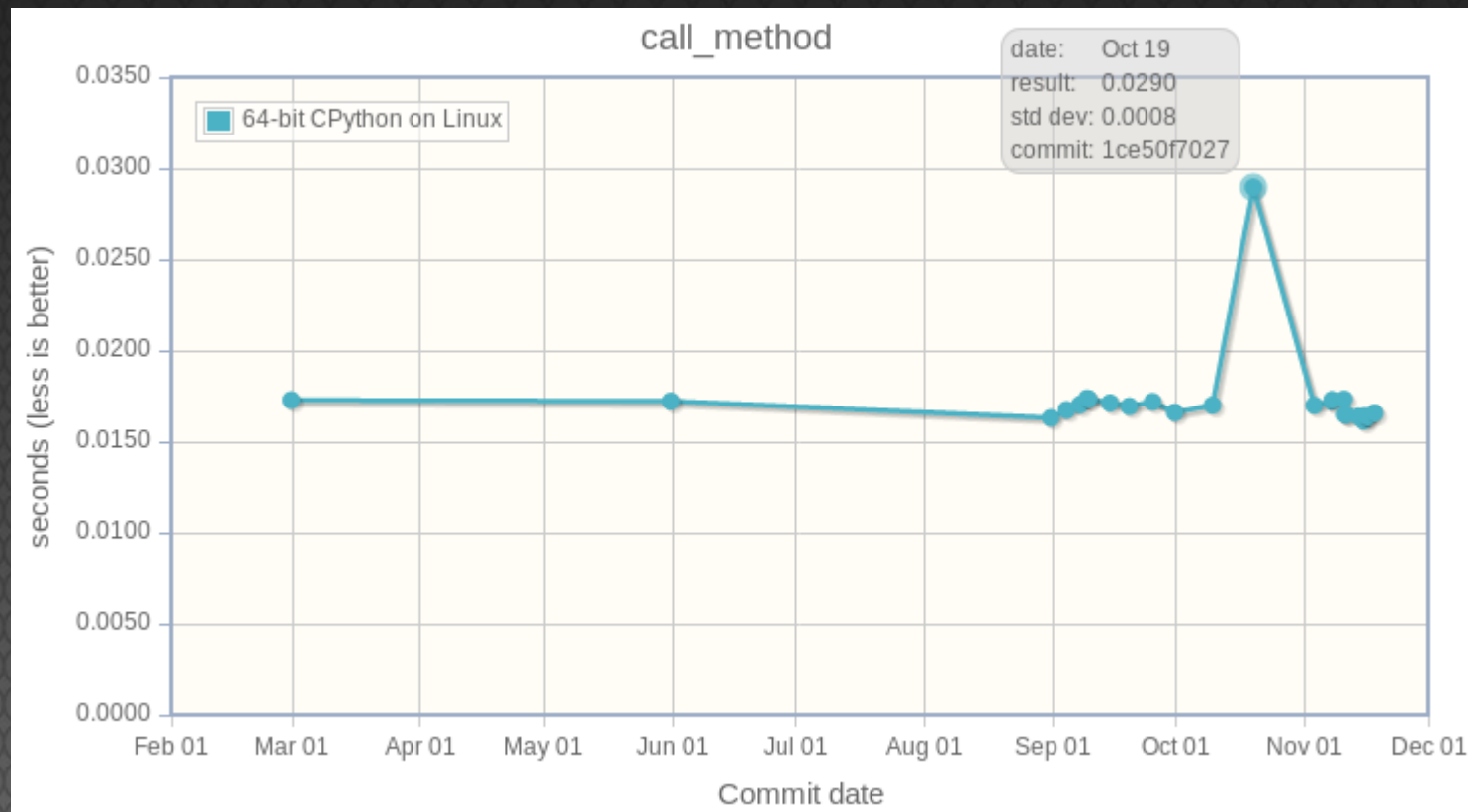


How to run **stable** benchmarks



FOSDEM 2017, Brussels

Victor Stinner
victor.stinner@gmail.com

BINARY_ADD optim



- In 2014, int+int optimization proposed: 14 patches, many authors
- Is it faster? Is it worth it?
- *The Grand Unified Python Benchmark Suite*
- Sometimes slower, sometimes faster
- Unreliable and unstable benchmarks?

Goal

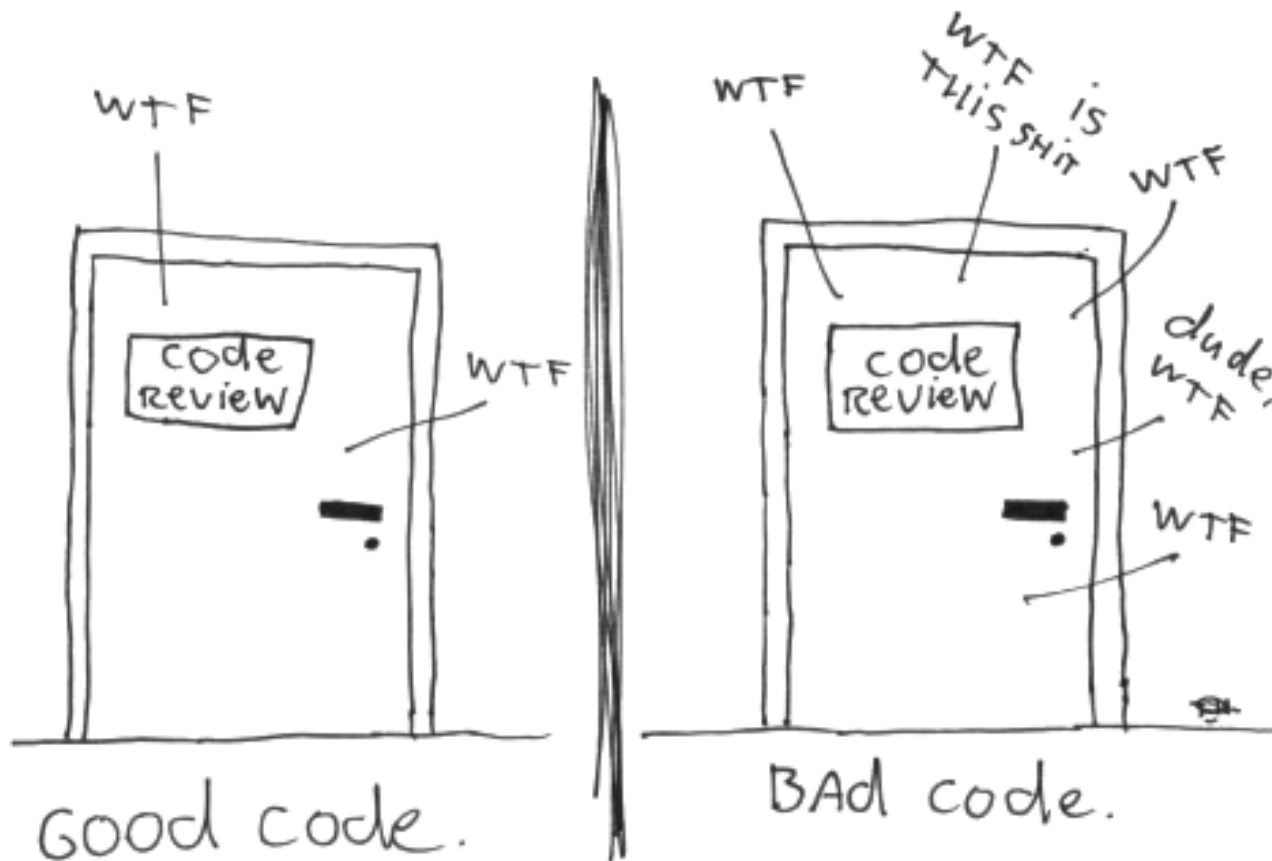


- Unreliable benchmarks leads to bad decisions
- Patch makes Python faster, slower or... is not significant?
- Need **reproducible** benchmark results on the same computer

WTF meter



The ONLY VALID MEASUREMENT
OF CODE QUALITY: WTFs/MINUTE



System & noisy apps



- CPU-bound microbenchmark:
`python3 -m timeit 'sum(range(10**7))'`
- Idle system: **229 ms**
- Busy system: **372 ms (1.6x slower, +62%)**
`python3 -c 'while True: pass'`
- **WTF?**

Isolated CPUs



- Kernel and applications share same CPUs , memory and storage
- Linux kernel **isolcpus=3** don't schedule processes on CPU 3
- Pin a process to a CPU:
taskset -c 3 python3 script.py
- Idle system: **229** ms
- Busy system, isolated CPU: **230** ms!

NOHZ_FULL & RCU



- Enter GRUB, modify Linux command line to add: **isolcpus=3**
- **nohz_full=3**: if only 0 or 1 process running on CPU 3, disable all interruptions on this CPU (WARNING: see later!)
- **rcu_nocbs=3**: don't run kernel code on CPU 3

FASTCALL optim



- April 2016, experimental change to avoid temporary tuple to call functions
- Builtin functions **20-50% faster!**
- **But** some **slower** benchmarks
- 20,000 lines patch reduced to adding two unused functions... still slower.
WTF??

Deadcode



- Reference:
1201.0 ms +/- 0.2 ms
- Add 2 unused functions:
1273.0 ms +/- 1.8 ms (**slower!**)
- Add 1 empty unused function:
1169.6 ms +/- 0.2 ms (**faster!**)

Deadcode

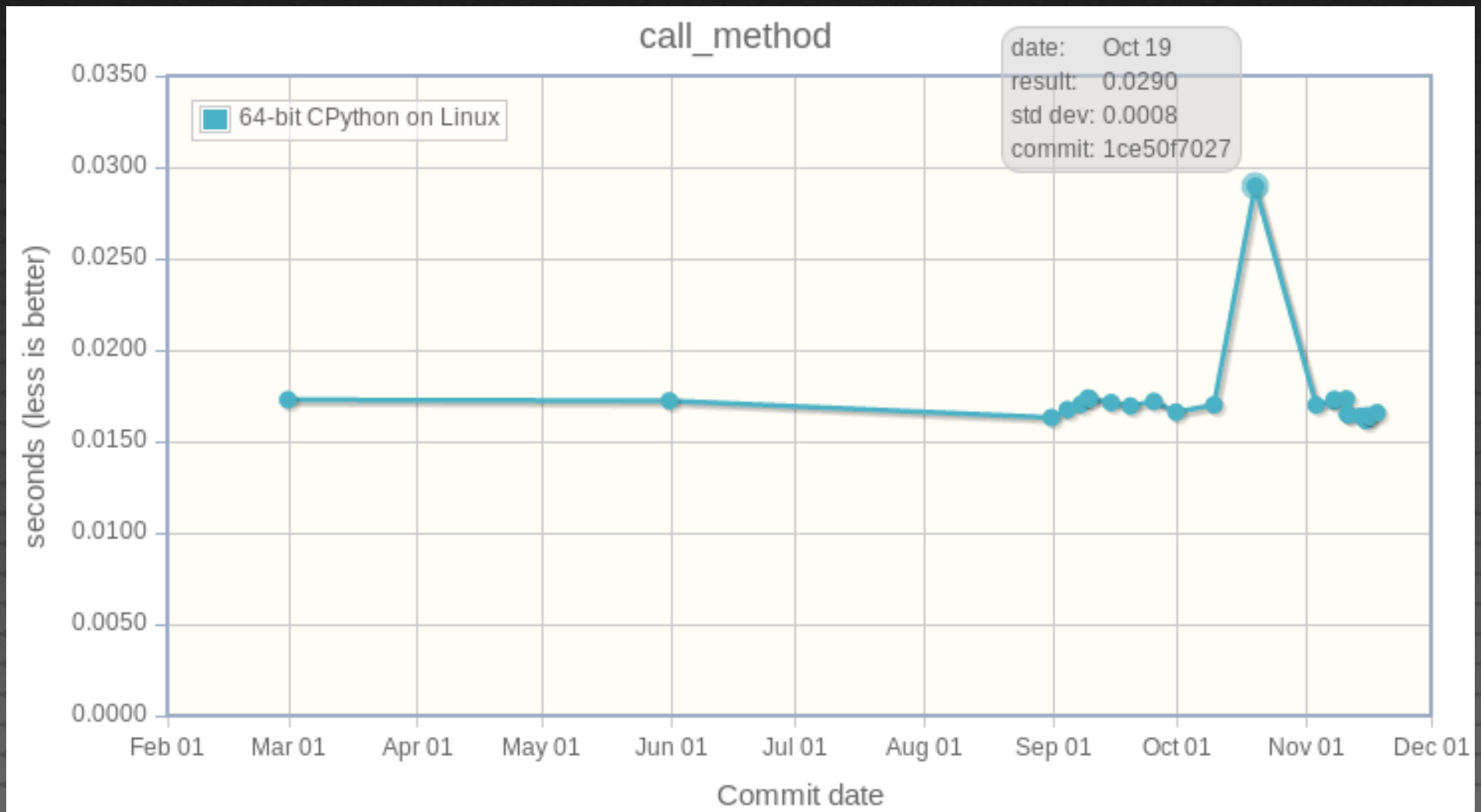


Code placement



- Root cause: *code placement*
- Memory layout and function addresses impact CPU cache usage
- It's very hard to get the best placement and so reproducible benchmarks

70% slower!



PGO fix deadcode



Profiled **G**uided **O**ptimizations (PGO):

```
./configure --with-optimizations
```

- (1) Compile with instrumentation
- (2) Run the test suite to collect statistics on **branches** and **code paths** (hot code)
- (3) Use statistics to recompile Python

Python hash function



- Hash function randomized by default
- PYTHONHASHSEED=1: 198 ms
- PYTHONHASHSEED=3: 207 ms (slower!)
- PYTHONHASHSEED=4: 187 ms (faster!)
- WTF???
- Different number of hash collisions

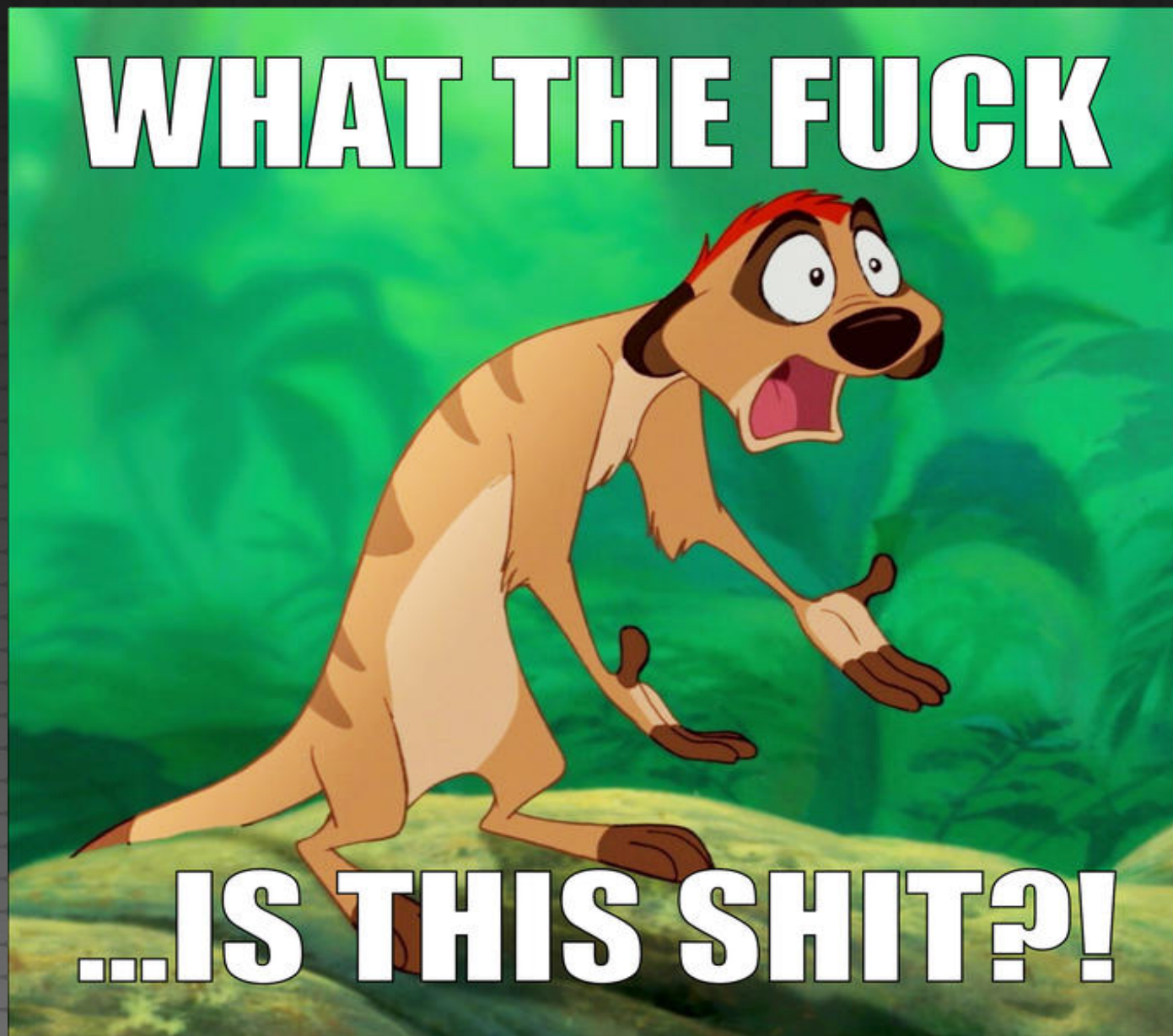
More fun



Performance also impacted by:

- Unused **environment variables**
- Current working **directory**
- Unused **command line arguments**
- etc.

WTF????



Average



- First, I disabled Address Space Layout Randomization (ASLR), randomizing Python hash function, etc.
- Lost cause: too many factors impact **randomly** performances
- Solution to random noise: compute **average of multiple samples**

perf



- Spawn multiple processes
- Compute average and standard deviation
- Store all individual run timings as JSON
- Command line tool to display, compare and analyze data

<http://perf.rtfld.io/>

New drama



- Everything was fine for weeks...
- ... until... the new drama
- **Suddenly**, a benchmark became **20% faster**
- WHAT-THE-FUCK ??????

Modern Intel CPUs



- Since 2005, the frequency of Intel CPUs changes **anytime** for various reasons
- Workload
- CPU temperature
- and... the **number of active cores**

Turbo Button?



Turbo Boost



- My laptop: i7-3520M, 2 physical cores, HyperThreading (4 cores)
- 2-4 active cores: 3.4 GHz
- 1 active core: **3.6 GHz (+5%)**
- "boost state support" of:
sudo cpupower frequency-info
- Disable in BIOS, or write 1 into
/sys/devices/system/cpu/
intel_pstate/no_turbo

Stable benchmarks!



Nightmare never ends



But...

... one friday afternoon when I **closed**
my GNOME session

... the benchmark becomes **2.0x faster**

WTF?????? (sorry, this one should
really be the last one... right?)

Nightmare never ends



Let me recall



- System and noisy apps: **isolcpus**
- Deadcode, code placement: **PGO**
- ASLR, Python hash function, env vars, cmdline, ...: **average** + std dev
- Turbo Boost: **disable TB**

CPU temperature?



CPU temperature?



NOHZ_FULL and Pstate



- **nohz_full=3** (...) disable all interruptions
- intel_pstate and intel_idle CPU drivers registers a scheduler callback
- No interruption means no scheduler interruption (LOC in /proc/interrupts)
- **Pstate** doesn't depend on isolated CPUs workload, but other CPUs workloda

NOHZ_FULL and Pstate



- intel_pstate and intel_idle drivers maintainer never tried NOHZ_FULL
- Intel RT engineers: « it's not a bug, it's a feature »
 - ⇒ Use a **fixed CPU frequency**
 - ⇒ or: **don't use NOHZ_FULL**

Questions?



<http://perf.rtfld.io/>

<https://github.com/python/performance/>