Virtual Time Measurement

in Programming Contests

Paweł Dietrich, Bartosz Kostka

Conference at International Olympiad in Informatics

July 30, 2025, Sucre Bolivia

Measurement real (wall) time of program's exectution came with some challanges:

- **Measurement error** On different hardware we have different results.
- **Limits need updates** Computers are gettings faster.
- **Resource sharing** Programs running at the same time fight for resources yelding different results after reruns.
- **Multicore utilization** Parallel judging is very challanging.
- **Platform portability** Problems can't be reused across training/contest environments

Virtual Time Measurement design requirements:

- **Hardware independence** Should be possible to run on any modern hardware.
- **No Source code needed** Ability to benchmark any binary,
- **No kernel modifications** Vanila Debian kernel should work.
- **Fairness in online contests** Deterministic programs should return always the same result on any hardware.
- **Distributable** Participants have ability to run it on their computers.
- **Easier problem setting** Ability to set time limits confidently using model solutions on any machine

Implementation

oitimetool uses Intel's PIN JIT, which instrumented the code to count the number of instructions that were executed, which was then converted to seconds.

- oitimetool was developed in 2008 by Szymon Accedański.
- Released at GitHub[1] under a Creative Commons license
- In 2011 final stage of Polish Olympiad in Informatics was evaluated with both (real time and oitimetool) methods giving participants lower of the two scores.
- In 2011/2012 edition Polish Olympiad in Informatics switched to using oitimetool only.

[1] https://github.com/olimpiada/oitimetool-bin

This implementation was successful, but had it's downsides:

- **Abstracted execution model** Memory access is uniform – ignores real-world cache/memory hierarchy
- **Separate real-time limits required** Needed for handling hangs or long syscalls
- ⋆ **Requires education** Contestants need tools to test within the same environment
- ⋆ **Licensing** Intel's PIN library has a proprietary license
- ⋆ **Common Memory** Instrumentation code and judged program share common memory, meaning it is possible to overwrite the score from the benchmarked program itself.

To address the issues in 2018 we have switched to `sio2jail` developed by Wojciech Dubiel *et alla*.

Implementation

`sio2jail` uses linux's perf tool to get the number of instructions as counted by the linux kernel.

- This fixed 2 issues we had with `oitimetool`.
- Polish Olympiad in Informatics is using `sio2jail` since 2018.

Example perf tool execution

```
$ perf stat -B dd if=/dev/zero of=/dev/null count=1000000

1000000+0 records in
1000000+0 records out
512000000 bytes (512 MB) copied, 0.956217 s, 535 MB/s

 Performance counter stats for 'dd if=/dev/zero of=/dev/null count=1000000':

            5,099 cache-misses             #      0.005 M/sec (scaled from 66.58%)
          235,384 cache-references         #      0.246 M/sec (scaled from 66.56%)
        9,281,660 branch-misses            #      3.858 %     (scaled from 33.50%)
      240,609,766 branches                 #    251.559 M/sec (scaled from 33.66%)
    1,403,561,257 instructions             #      0.679 IPC   (scaled from 50.23%)
    2,066,201,729 cycles                   #   2160.227 M/sec (scaled from 66.67%)
              217 page-faults              #      0.000 M/sec
                3 CPU-migrations           #      0.000 M/sec
               83 context-switches         #      0.000 M/sec
       956.474238 task-clock-msecs         #      0.999 CPUs

       0.957617512  seconds time elapsed
```

Source code

`sio2jail` is available on GitHub (`https://github.com/sio2project/sio2jail`) on MIT license.

Source code

sio2jail is available on GitHub (https://github.com/sio2project/sio2jail) on MIT license.

Easy running

oiejq tool allows to use the sio2jail exactly like a time(1) tool. Download is available at https://oij.edu.pl/zawodnik/srodowisko/oiejq.tar.gz

Benefits

- Full use of multi-core machines
- Cost and time efficiency
- Very high accuracy
- No extra judging overhead

Adaptation issues

- Excluded from collegiate contests and Algorithmic Engagements in Poland
- Experienced contestants found less value in abstraction
- Up until now, no literature was available for non-polish speakers.

Trade–offs

- Simpler CPU cost model
- Uniform memory performance

Virtual Time Measurement

in Programming Contests

We encourage all to explore this approach
in national and international olympiads

Questions?

p.dietrich@fri.edu.pl

kostka@oij.edu.pl