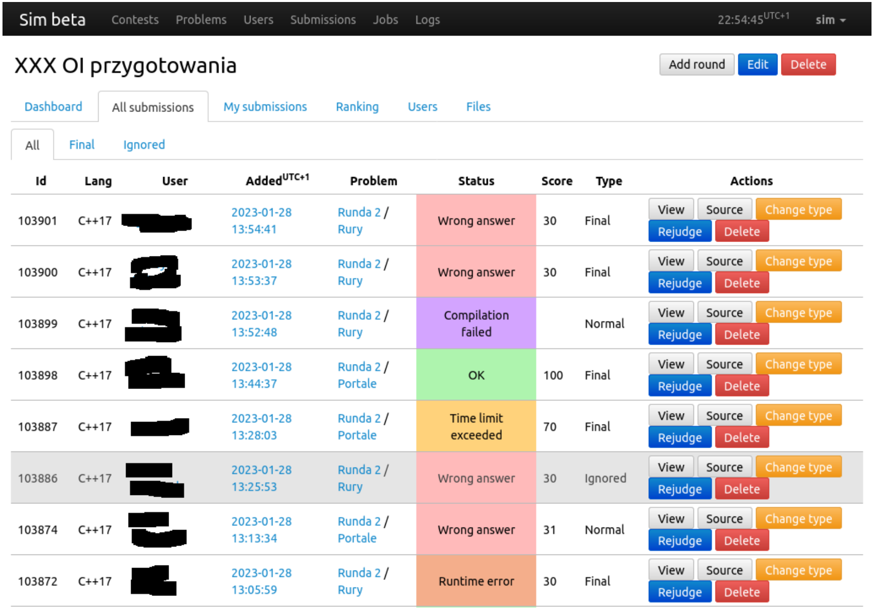
Sandbox for multi-process applications for unprivileged users on Linux

Krzysztof Małysa

University of Warsaw

December 2023

The Sim platform



Sim beta | Contests | Problems | Users | Submissions | Jobs | Logs | 22:54:45 UTC+1 | sim

XXX OI przygotowania

Add round | Edit | Delete

Dashboard | All submissions | My submissions | Ranking | Users | Files

All | Final | Ignored

Id	Lang	User	Added UTC+1	Problem	Status	Score	Type	Actions
103901	C++17	[redacted]	2023-01-28 13:54:41	Runda 2 / Rury	Wrong answer	30	Final	View Source Change type Rejudge Delete
103900	C++17	[redacted]	2023-01-28 13:53:37	Runda 2 / Rury	Wrong answer	30	Final	View Source Change type Rejudge Delete
103899	C++17	[redacted]	2023-01-28 13:52:48	Runda 2 / Rury	Compilation failed		Normal	View Source Change type Rejudge Delete
103898	C++17	[redacted]	2023-01-28 13:44:37	Runda 2 / Portale	OK	100	Final	View Source Change type Rejudge Delete
103887	C++17	[redacted]	2023-01-28 13:28:03	Runda 2 / Portale	Time limit exceeded	70	Final	View Source Change type Rejudge Delete
103886	C++17	[redacted]	2023-01-28 13:25:53	Runda 2 / Rury	Wrong answer	30	Ignored	View Source Change type Rejudge Delete
103874	C++17	[redacted]	2023-01-28 13:13:34	Runda 2 / Portale	Wrong answer	31	Normal	View Source Change type Rejudge Delete
103872	C++17	[redacted]	2023-01-28 13:05:59	Runda 2 / Rury	Runtime error	30	Final	View Source Change type Rejudge Delete

Before the new sandbox

Old sandbox

- Single-threaded, statically-linked executables
- C, C++, Pascal
- Overhead — `ptrace`

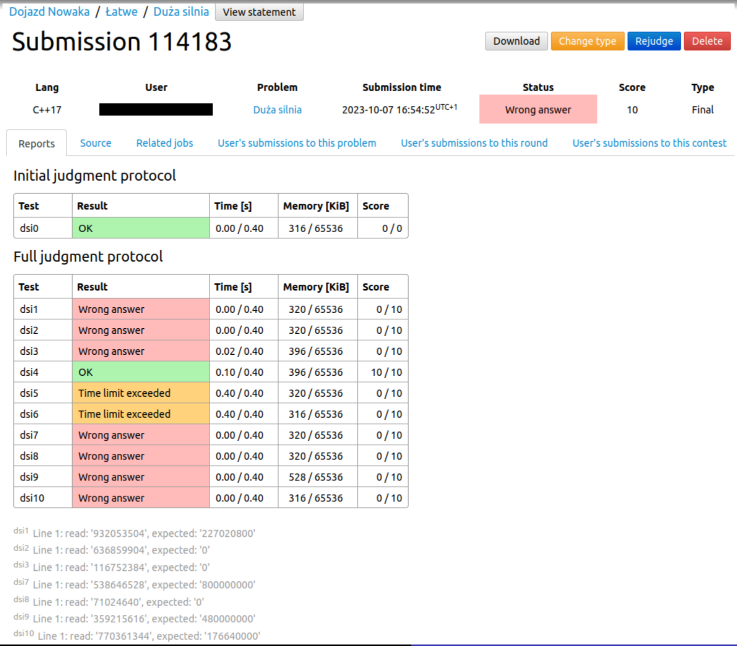
Problems

- No support for other languages e.g. Python
- Compilation

Solution

A new sandbox.

The submission on the Sim platform



Dojazd Nowaka / Łatwe / Duża silnia View statement

Submission 114183

Download Change type Rejudge Delete

Lang	User	Problem	Submission time	Status	Score	Type
C++17	[redacted]	Duża silnia	2023-10-07 16:54:52 UTC+1	Wrong answer	10	Final

Reports Source Related jobs User's submissions to this problem User's submissions to this round User's submissions to this contest

Initial judgment protocol

Test	Result	Time [s]	Memory [KiB]	Score
dsi0	OK	0.00 / 0.40	316 / 65536	0 / 0

Full judgment protocol

Test	Result	Time [s]	Memory [KiB]	Score
dsi1	Wrong answer	0.00 / 0.40	320 / 65536	0 / 10
dsi2	Wrong answer	0.00 / 0.40	320 / 65536	0 / 10
dsi3	Wrong answer	0.02 / 0.40	396 / 65536	0 / 10
dsi4	OK	0.10 / 0.40	396 / 65536	10 / 10
dsi5	Time limit exceeded	0.40 / 0.40	320 / 65536	0 / 10
dsi6	Time limit exceeded	0.40 / 0.40	316 / 65536	0 / 10
dsi7	Wrong answer	0.00 / 0.40	320 / 65536	0 / 10
dsi8	Wrong answer	0.00 / 0.40	320 / 65536	0 / 10
dsi9	Wrong answer	0.00 / 0.40	528 / 65536	0 / 10
dsi10	Wrong answer	0.00 / 0.40	316 / 65536	0 / 10

```
dsi1 Line 1: read: '932053504', expected: '227020800'
dsi2 Line 1: read: '636859904', expected: '0'
dsi3 Line 1: read: '116752384', expected: '0'
dsi7 Line 1: read: '538646528', expected: '800000000'
dsi8 Line 1: read: '71024640', expected: '0'
dsi9 Line 1: read: '359215616', expected: '480000000'
dsi10 Line 1: read: '770361344', expected: '176640000'
```

Requirements for new Sandbox

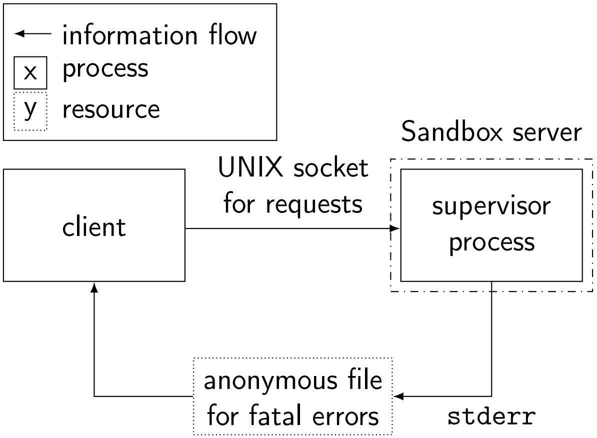
1. Versatile
2. Support for multi-process applications
3. Low overhead
4. Optimized for short-running programs
5. Limiting resources
6. Runtime statistics
7. For unprivileged users

Existing solutions

- Often require privileges e.g. OS modifications.
- Some use `ptrace`. Slow, TOCTOU problem.
- Few provide runtime statistics.
- None is optimized for short-running programs.

Design: client-server

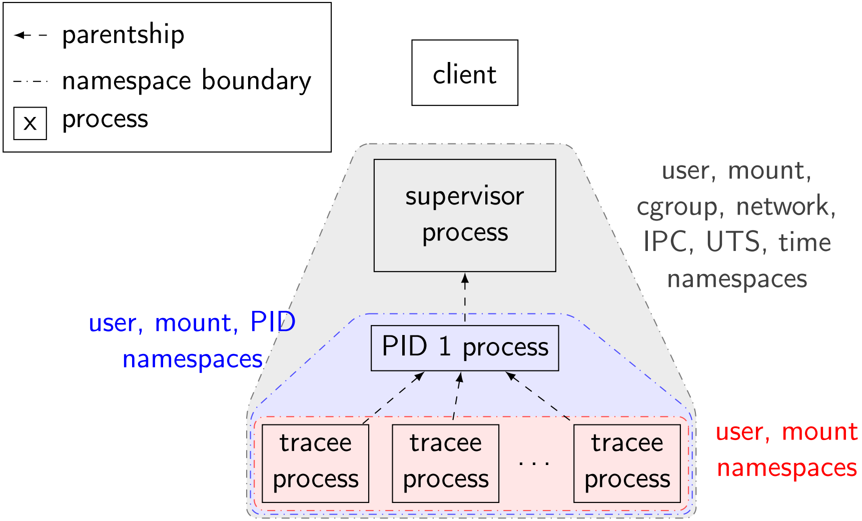
Allows sharing resources between requests.



Benchmarks in a moment.

Used Linux kernel mechanisms

Linux namespaces, cgroups, `prlimit`, seccomp BPF filters.



Benchmark of some performance optimizations

Benchmark	Mean request time	Std. dev.	Std. err. on the mean	Slowdown
Baseline	2.348ms	0.768ms (32.71%)	0.024ms (1.03%)	0.00%
New network namespace for each request	2.970ms	0.856ms (28.83%)	0.027ms (0.91%)	26.49%
New IPC namespace for each request	2.522ms	0.782ms (31.02%)	0.025ms (0.98%)	7.41%
New UTS namespace for each request	2.478ms	0.771ms (31.14%)	0.024ms (0.98%)	5.54%

Table: Statistics for each row were collected from 1000 runs. Each row contains real time it took to handle request to sandbox the `/bin/true` program.

Comparison with nsjail

Sandbox	Mean time	Std. dev.	Std. err. on the mean	Slowdown
no sandbox	0.893ms	0.409ms (45.80%)	0.013ms (1.45%)	1x
sandbox	2.348ms	0.768ms (32.71%)	0.024ms (1.03%)	2.39x
nsjail	10.393ms	1.327ms (12.77%)	0.042ms (0.40%)	10.57x

Table: Statistics for each row were collected from 1000 runs. Each row contains real time it took to handle request to sandbox the `/bin/true` program. While the slowdown of the sandbox is huge (more than twofold), it still allows for hundreds of runs per second and that was the goal of this thesis, whereas nsjail is more than 4 times slower than our sandbox.

Testing performance and overheads

Tested compilation of all solutions of 4 problems from finals of the XXII Polish Olimpiad in Informatics.

- Compilation.
- Running the model solution on the tests.

Benchmark

Collect statistic from 10 runs.

Compilation performance

Tested compilation of all solutions of 4 problems from finals of the XXII Polish Olimpiad in Informatics. $18 + 18 + 13 + 8 = 57$ solutions.

- Only some compilations present statistically significant difference.
- Half of the time it is statistically no slower that without the sandbox.
- 0% – 24% slowdown. Most of the time below 10%.
- Almost always seccomp BPF filters cause the overhead (up to 15% without it).

Model solutions' performance

For each problem the single model solution was tested. Problems have: 55 + 57 + 69 + 65 = 246 tests.

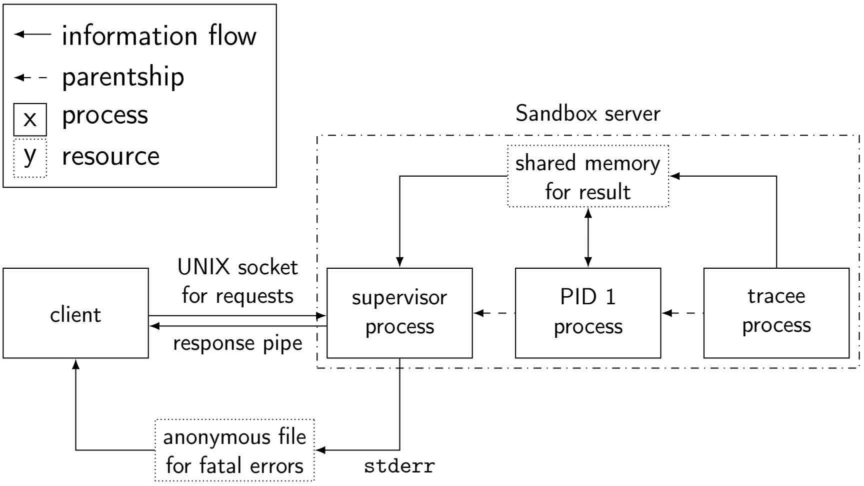
- Vast majority of tests shows no statistical difference.
- Few test show slowdown — up to 19%.
- Medium tests even show speed-up — up to 44%.
- Low number of system calls. Most of the time, computations without IO.

Future work

- Setting CPU affinity
- Support for networking
- Rust frontend / client
- Further experimentation

Thank you.

Bonus: IPC


information flow
parentship
x
process
y
resource
Sandbox server
shared memory for result
client
UNIX socket for requests
response pipe
supervisor process
PID 1 process
tracee process
anonymous file for fatal errors
stderr

Bonus: Implementation

1. Time limits
2. Runtime statistics
3. Error handling (stderr)
4. Request sending and receiving (serialization)
5. File descriptors
6. Cancelling or killing request
7. Sandbox server upon client death
8. PID 1 process upon supervisor death
9. Signals (tracee, SIGPIPE, UBSan)
10. Running as superuser
11. Performance optimizations