DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

OCI : For all you Continuous Integration and Benchmarking needs

| F.Bobot

16 February

INSTITUT CARNOT CEA LIST

digiteo

List

Continuous Integration Needs

A bug not introduced is a bug that don't need to be fixed!

Continuous Integration Needs

- Repository F on which depend plugins $P_1, \ldots, P_{10}$

Continuous Integration Needs

- Repository F on which depend plugins $P_1, \ldots, P_{10}$
- We want to test each merge-request of F

Continuous Integration Needs

- Repository F on which depend plugins $P_1, \ldots, P_{10}$
- We want to test each merge-request of F
- We want to test they doesn't breaks the master of the plugins P_i

Continuous Integration Needs

- Repository F on which depend plugins $P_1, \ldots, P_{10}$
- We want to test each merge-request of F
- We want to test they doesn't breaks the master of the plugins P_i
- We want to test each merge-request of P_i with the master of F

Continuous Integration Needs

ex: I want to test the commit 2385da5465 of my plugin

Continuous Integration Needs

ex: I want to test the commit 2385da5465 of my plugin

1. For that I need to compile my Frama-C plugin!

Continuous Integration Needs

ex: I want to test the commit 2385da5465 of my plugin

1. For that I need to compile my Frama-C plugin!
2. But for that I need a compiled Frama-C!

Continuous Integration Needs

ex: I want to test the commit 2385da5465 of my plugin

1. For that I need to compile my Frama-C plugin!
2. But for that I need a compiled Frama-C!
3. But for that I need a compiled OCamlgraph!

Continuous Integration Needs

ex: I want to test the commit 2385da5465 of my plugin

1. For that I need to compile my Frama-C plugin!
2. But for that I need a compiled Frama-C!
3. But for that I need a compiled OCamlgraph!
4. But for that I need a compiled ZArith!

Continuous Integration Needs

ex: I want to test the commit 2385da5465 of my plugin

1. For that I need to compile my Frama-C plugin!
2. But for that I need a compiled Frama-C!
3. But for that I need a compiled OCamlgraph!
4. But for that I need a compiled ZArith!
5. But for that I need a compiled OCaml!

Continuous Integration Needs

ex: I want to test the commit 2385da5465 of my plugin

1. For that I need to compile my Frama-C plugin!
2. But for that I need a compiled Frama-C!
3. But for that I need a compiled OCamlgraph!
4. But for that I need a compiled ZArith!
5. But for that I need a compiled OCaml!
6. But for that I need GCC, ...

Benchmarking Needs

ex: I want to compare ten versions of Popop on this benchmark!

1. For that I need to run gnuplot on a data file!

Benchmarking Needs

ex: I want to compare ten versions of Popop on this benchmark!

1. For that I need to run gnuplot on a data file!
2. But for that I need to generate the data file!

Benchmarking Needs

ex: I want to compare ten versions of Popop on this benchmark!

1. For that I need to run gnuplot on a data file!
2. But for that I need to generate the data file!
3. But for that I need to run each versions on each problems!

Benchmarking Needs

ex: I want to compare ten versions of Popop on this benchmark!

1. For that I need to run gnuplot on a data file!
2. But for that I need to generate the data file!
3. But for that I need to run each versions on each problems!
4. But for that I need the compilation of each versions!

Benchmarking Needs

ex: I want to compare ten versions of Popop on this benchmark!

1. For that I need to run gnuplot on a data file!
2. But for that I need to generate the data file!
3. But for that I need to run each versions on each problems!
4. But for that I need the compilation of each versions!
5. But for that I need a compiled OCamlgraph!

Benchmarking Needs

ex: I want to compare ten versions of Popop on this benchmark!

1. For that I need to run gnuplot on a data file!
2. But for that I need to generate the data file!
3. But for that I need to run each versions on each problems!
4. But for that I need the compilation of each versions!
5. But for that I need a compiled OCamlgraph!
6. But for that I need a compiled ZArith!

Benchmarking Needs

ex: I want to compare ten versions of Popop on this benchmark!

1. For that I need to run gnuplot on a data file!
2. But for that I need to generate the data file!
3. But for that I need to run each versions on each problems!
4. But for that I need the compilation of each versions!
5. But for that I need a compiled OCamlgraph!
6. But for that I need a compiled ZArith!
7. But for that I need a compiled OCaml!

Benchmarking Needs

ex: I want to compare ten versions of Popop on this benchmark!

1. For that I need to run gnuplot on a data file!
2. But for that I need to generate the data file!
3. But for that I need to run each versions on each problems!
4. But for that I need the compilation of each versions!
5. But for that I need a compiled OCamlgraph!
6. But for that I need a compiled ZArith!
7. But for that I need a compiled OCaml!
8. But for that I need GCC, ...

Continuous Integration: What's Out There?

- Hudson/Jenkins
- Travis/Gitlab-CI
- $\approx$ Buildbot

Continuous Integration: What's Out There?

- Hudson/Jenkins
- Travis/Gitlab-CI
- $\approx$ Buildbot

- Only triggers: no dependencies

Continuous Integration: What's Out There?

- Hudson/Jenkins
- Travis/Gitlab-CI
- ≈ Buildbot

- Only triggers: no dependencies
- Artifact: fixed ones

Continuous Integration: What's Out There?

- Hudson/Jenkins
- Travis/Gitlab-CI
- ≈ Buildbot

- Only triggers: no dependencies
- Artifact: fixed ones
- Cache: poor-man dependency

Benchmarking

- CVC4: Specific tools

Benchmarking

- CVC4: Specific tools
- StarExec: doesn't compile the prover

Benchmarking

- CVC4: Specific tools
- StarExec: doesn't compile the prover
- ProofManager (Alain Mebsout): doesn't compile the prover

Benchmarking

- CVC4: Specific tools
- StarExec: doesn't compile the prover
- ProofManager (Alain Mebsout): doesn't compile the prover
- Genet (Maxence Guesdon): no environnement management

Benchmarking

- CVC4: Specific tools
- StarExec: doesn't compile the prover
- ProofManager (Alain Mebsout): doesn't compile the prover
- Genet (Maxence Guesdon): no environnement management
- PoC using remake: work well but no environnement management

OCI: three circle of OCI

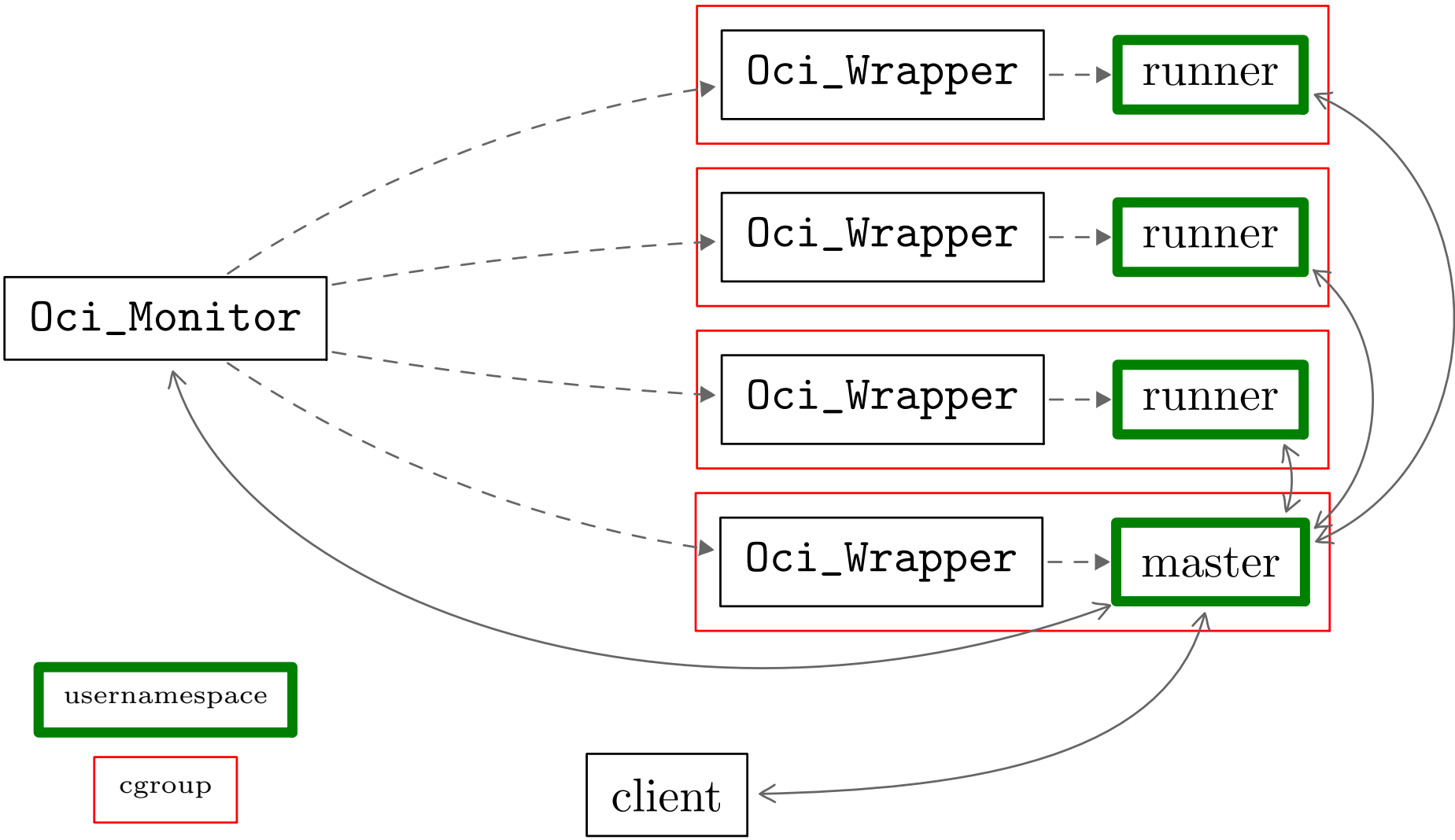
- Inner-Layer: Base techniques
- Middle-Layer: Plumbering
- Client-Layer: As simple and general for the user as possible

OCI

OCI

Manager of execution environments

OCI: Schema


Oci_Wrapper
runner
Oci_Wrapper
runner
Oci_Monitor
Oci_Wrapper
runner
Oci_Wrapper
master
usernamespace
cgroup
client

OCI: base features

- Chroot on steroid (similar to nspawn, lxc)

OCI: base features

- Chroot on steroid (similar to nspawn, lxc)
- Users management ($id \in [0; 1000]$ even if oci is a normal user)

OCI: base features

- Chroot on steroid (similar to nspawn, lxc)
- Users management ($id \in [0; 1000]$ even if oci is a normal user)
- Artifacts management

OCI: base features

- Chroot on steroid (similar to nspawn, lxc)
- Users management ($id \in [0; 1000]$ even if oci is a normal user)
- Artifacts management
  - Install previously created one

OCI: base features

- Chroot on steroid (similar to nspawn, lxc)
- Users management ($id \in [0; 1000]$ even if oci is a normal user)
- Artifacts management
  - Install previously created one
  - Create one from the installed files

OCI: base features

- Chroot on steroid (similar to nspawn, lxc)
- Users management ($id \in [0; 1000]$ even if oci is a normal user)
- Artifacts management
  - Install previously created one
  - Create one from the installed files
- CPU management (important for benchmarks)

OCI: base features

- Chroot on steroid (similar to nspawn, lxc)
- Users management ($id \in [0; 1000]$ even if oci is a normal user)
- Artifacts management
  - Install previously created one
  - Create one from the installed files
- CPU management (important for benchmarks)
- Communication management (log, new runners, ...)

OCI: base features

- Chroot on steroid (similar to nspawn, lxc)
- Users management ($id \in [0; 1000]$ even if oci is a normal user)
- Artifacts management
  - Install previously created one
  - Create one from the installed files
- CPU management (important for benchmarks)
- Communication management (log, new runners, ...)

OCI: base features

- Chroot on steroid (similar to nspawn, lxc)
- Users management ($id \in [0; 1000]$ even if oci is a normal user)
- Artifacts management
  - Install previously created one
  - Create one from the installed files
- CPU management (important for benchmarks)
- Communication management (log, new runners, ...)

$\Rightarrow$ Linux $>$ 3.18 only

OCI: Log

```
type kind =
  | Standard | Error | Chapter | Command

type 'a data =
  | Std of kind * string
  | Extra of 'a Or_error.t

type 'a line = {
  data : 'a data;
  time : Core.Time.t;
}
```

Artifacts

- It is a directory hierarchy.
- Can be stacked
- Created by keeping the new files

Best would be overlayfs, but we use hardlinks.

OCI: If you want your own middle-layer

Two libraries:

- Oci_Runner
- Oci_Master

Rootfs

- Create artifact from an lxc image
- Create artifact by installing (debian) packages

OCI: Outer-layer: The client

- The only specific needed part.
- Can use the `Oci_Client` library

Steps:

1. Cmdline Parsing
2. Query Generation from *Repository Definition*
   - Contains the description of all the dependencies
   - Only SHA1 used for commits
3. RPC Request
4. Result Analysis

OCI: Query Definition

```
type exec = {
  cmd: string;
  args: [ `S of string |
          `Proc of formatted_proc (* "-j%i" *)
        ] list;
  env : [ `Replace of (string * string) list
        | `Extend of (string * string) list];
  proc_requested : int;
  working_dir: Oci_Filename.t (** Relative path *)
}

val exec: ... -> cmd
val run: ?env:env -> string -> string list -> cmd
val make: ?j:int -> ?vars:(string * string) list ->
  ?env:env -> string list -> cmd
```

OCI: Query Definition

```
val mk_repo:
    ?revspec:string ->
    url:string ->
    deps:repo list ->
    cmds:Git.cmd list ->
    ?tests:Git.cmd list ->
    string (** name *) ->
    repo
```

OCI: Query Definition

```
val git_clone:
  url:string ->
  ?dir:Oci_Filename.t ->
  Commit.t ->
  cmd

val git_copy_file:
  url:string ->
  src:Oci_Filename.t ->
  dst:Oci_Filename.t ->
  Commit.t ->
  cmd
```

Xpra integration

Xpra is screen for X.

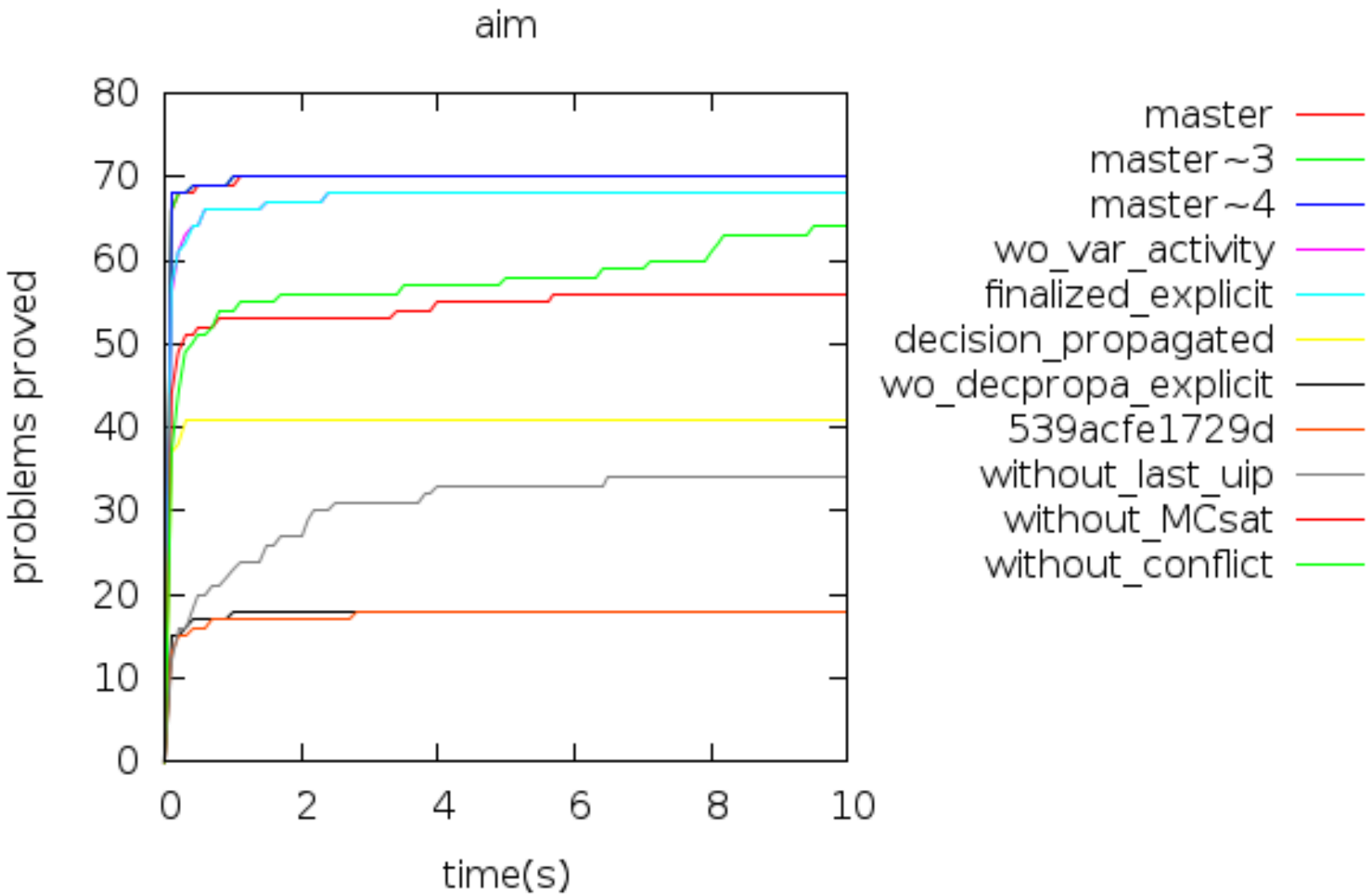
Benchmark

```
--x-input popop_commit
master
master~3
...
without_MCsat
...
--y-input popop_aim
tests/dimacs/aim/aim-50-3_4-yes1-4.cnf
tests/dimacs/aim/aim-100-2_0-yes1-4.cnf
tests/dimacs/aim/aim-200-3_4-yes1-4.cnf
tests/dimacs/aim/aim-50-3_4-yes1-3.cnf
tests/dimacs/aim/aim-50-6_0-yes1-1.cnf
tests/dimacs/aim/aim-50-2_0-yes1-2.cnf
tests/dimacs/aim/aim-50-1_6-yes1-2.cnf
tests/dimacs/aim/aim-200-1_6-no-1.cnf
tests/dimacs/aim/aim-200-1_6-yes1-1.cnf
tests/dimacs/aim/aim-50-1_6-no-2.cnf
...
```

Benchmark

```
bin/soprano_client.native compare_n ...
  popop_compare_n --x-input popop_commit  --y-input

((master
(0 0.01 0.04 0 0 0 0 0 0.02 0 0 0 0 0.08 0.02 0 0 0.01 0.02 0
 0.02 0 0 0 0.04 0 0.02 0.02 0 0.76 0.01 0 0 0 0 0 0.01 5.29
 0.03 0 0.01 0 0.01 0 0.5 0 0 0.02 0 0 0.04 0.01 0.01 0.04 0.
 0 0 0 0.19 0))
(master~3
(0 0.01 0.04 0 0 0 0 0 0.01 0 0 0 0 0.08 0.03 0 0 0.01 0.02 0
 0.02 0 0 0 0.04 0 0.04 0.02 0 0.76 0.01 0 0.01 0 0 0 0.01 5.
 0.02 0 0 0 0 0 0.5 0 0 0.02 0 0 0.04 0.02 0.01 0.04 0.01 0 0
 0.19 0))
...
(without_MCsat
(0.01 52.89 0.79 0.01 0.01 0 0 0.22 12.25 0 0.06 0.1 0 0.13 0
 8.06 0.06 0.05 0.06 0.02 0.09 11.46 0.02 0 0.14 0.46 102.62
 0.07 41.3 0 0.04 0 7.52 0 0.05 5.95 0 0.12 0.02 0.04 0.09 0.
 0.09))
```


aim
problems proved
time(s)
0
10
20
30
40
50
60
70
80
0
2
4
6
8
10
master
master~3
master~4
wo_var_activity
finalized_explicit
decision_propagated
wo_decpropa_explicit
539acfe1729d
without_last_uip
without_MCsat
without_conflict

Benchmark: compare ocaml version

```
((ocaml 59a4fd6615454362aba2b7c4c5ea788d19edd739)
  (camlp4 2b894c317dde75bcf26f4c71b3f489efe41bc1df)
((ocaml bdf3b0fac7dd2c93f80475c9f7774b62295860c1)
  (camlp4 4.02+6))
```

Benchmark: test another ocaml version

```
bin/soprano_client.native run
   --socket oci-data/oci.socket --rootfs 3
   --ocaml trunk --camlp4 trunk popop
```

Benchmark: test another ocaml version

```
bin/soprano_client.native run
   --socket oci-data/oci.socket --rootfs 3
   --ocaml trunk --camlp4 trunk popop

File "src/variable.ml", line 1:
 Error: The implementation src/variable.ml
        does not match the interface src/variable.c
        Values do not match:
          val add_dec :
            dec:Data.t -> Solver.Delayed.t -> Popop
        is not included in
          val add_dec :
            dec:make_dec -> Solver.Delayed.t -> Pop
        File "src/variable.ml", line 65, characters
 Command exited with code 2.
```

A lot of dependencies

- Extunix: for system calls
- Core: for a good standard library
- Async: for a good cooperative thread library
- Async_shell: Great for calling programs
- Async_rpc: Great for RPC
- Textutils: for terminal colors
- Cmdliner: For parsing command line

Soon available at `http://github.com/bobot/oci` and in opam

Big Todos:

- Improve stability
- Use opam data for building the queries
- Compress storage of artifact