Operf: Benchmarking the OCaml Compiler

Pierre Chambart, Michael Laporte, Vincent Bernardoff, Fabrice Le Fessant

Operf: Benchmarking the OCaml Compiler

Pierre Chambart[1], Michael Laporte[1], Vincent Bernardoff[1], and Fabrice Le Fessant[1,2]

[1]OCamlPro, Paris
[2]INRIA Paris-Rocquencourt

Abstract

We present operf, a set of tools to benchmark the OCaml compiler, with both micro-benchmarks, for fast feedback during development, and macro-benchmarks, for results on a larger set of benchmarks.

1 Introduction

Adding optimisations to the compiler is not risk-free. A compiler is a complicated beast and adding some code can introduce complex bugs and increase the maintenance burden. Before considering an optimisation for inclusion, the risk/reward ratio must be carefully evaluated. Currently, there is no way to simply assert the improvement or regression of a change. This is the problem that operf intend to solve.

2 Use cases

Benchmarking a compiler means benchmarking programs built with a given compiler. There is nothing really specific to compilers benchmarking compared to a library or a program, but it represents a lot more bookeeping and maintainance. In particular there is a need for a common an automatable way to compile and execute lot of different applications, recover the results in a similar fashion and compare them, sometimes over a long period of time. For that, operf is composed of two packages: `operf-micro` and `operf-macro`.

3 The operf-macro Package

The operf-macro package is designed to benchmark real applications and accurately account for their running time. It is the part of operf that is expected to be used to evaluate compiler changes before integration. The package provides tools to measure the running time, CPU counters (like number of intructions executed), and GC statistics. To recover GC information, we propose a simple standard. If a program does not comply with it, it is still possible to extract some GC information with another tool (cf. our submitted article on lldb for OCaml, and the ocp-gcstats tool).

Since the evaluation of the compiler needs a representative set of applications, operf-macro has been designed to allow developers to easily add new benchmarks, by recording benchmarks in a simple database. We expect that this will allow a majority of open-source released OPAM

Authors' addresses: `pierre.chambart at ocamlpro.com`, `fabrice.le_fessant at inria.fr`, `vincent.bernardoff at ocamlpro.com`

packages to declare their own benchmarks, and thus, to be automatically tested for performance and correctness when a new version of OCaml will be released.

operf-macro leverages OPAM to simplify automated bulk testing. It is not strongly tied, but good practices and a few command-line tools makes it as easy as typing one command to test a new compiler version on the world.

4 The operf-micro Package

The operf-micro package is intended to be used in quick iterations of modification/evaluation of the compiler. A developer can typically add the `operf-micro` command to its makefile invocation to evaluate its changes after every build. The operf-micro tool was designed to simply test some small patterns, or algorithms and demonstrate in which cases some optimisation should or should not be triggered. To allow immediate use after the compilation of the compiler, operf-micro has as few dependencies as possible. Benchmarks built using its simple API can report their running time, but also the GC statistics and CPU cycles, in a format usable by operf-macro.

5 Conclusion

Currently, `operf-micro` and `operf-macro` are distributed with a set of benchmarks, which are used to develop the new flambda inliner for OCaml. Those benchmarks were also useful to uncover early bugs during the developement of the 4.03.0 branch. Maintaining a test and benchmark suite for a lot of applications on the developement branch of OCaml represents quite a lot of work. We think that this work is valuable, and can improve the the confidence of compiler developers in their changes and their impact on real world software. Of course, we hope the original authors will take over the work of maintaining benchmarks for their applications, and we designed the tools with this goal in mind. We wish that operf will have a success comparable with OPAM for package developers.