Title: Benchmarking An Object-Oriented Language & Environment

Keywords: testing / continuous integration / benchmarks / performance

Laboratoire, institution et université: INRIA Lille Nord Europe

Location: Lille, France

Team: Equipe RMoD – INRIA Lille Nord Europe

Internship supervisor: Pablo Tesone <pablo.tesone@inria.fr>

Context

Guaranteeing the performance of an application during its maintenance stage is a crucial point to keep the application useful and productive to the users [1].

This problem is also present in the maintenance of a programming language and its set of tools. Programming environments receive new features and bug fixes all the time. Introducing these new features should not affect the performance of the environment.

A programming environment is divided in different elements and tools: compiler, runtime engine, libraries, and a set of interactive tools [2]. A good performance should be guarantee in all the elements. For example, a change in the library might produce an impact in the execution of the developed applications using the programming language; but also a performance degradation in the tools affect the daily operation of the developers.

Pharo is a pure object-oriented programming language and a powerful environment, focused on simplicity and immediate feedback (think IDE and OS rolled into one) [3].

A way of guaranteeing that a code change does not introduce a performance degradation is to execute a series of automatised benchmarks on each of the changes [4]. This operation requires having a stable environment and also it requires times to execute them. A way of solving these caveats is to add the execution of the benchmarks in a continuous integration scheme [5].

Pharo has a framework to implement easily implement automatised benchmarks: SMark [6]. This benchmark building tool allows developers to write benchmarks in the same fashion that test are writing using any unit testing framework.

However, the validation performed by benchmarks only assures the performance regression of the tested elements. Moreover, the existing benchmark sets in the literature cover generic operations of a object-oriented programming language, e.g., arithmetic operations, object allocation. So, a set of valid benchmark is required to evaluate the performance of specific parts of the programming environment is required (for example: compiling methods, navigating the methods, traversing a specific collection).

Finally, to really detect and compare regressions in performance it is required to keep history of the results and identify the results that are a real performance regression (try to identify executions that were affecting by the load of the machine, or if the difference of performance is inside a parametrised threshold).

Objectives

- Implement a series of benchmarks using SMark to validate the performance of the language, the libraries, and tools.
- Extend the execution engine of SMark to collect statistics and keep them in a database.
- Using the collected data show the regression of the benchmarks and decided if a given change has produce a regression.
- Integrate the validation of regressions in the continuous integration process

Personal Skills

- Curiosity
- Willingness to learn
- Good communication skills
- Good programming skills
- English

References

1. Horký, Vojtěch, et al. "Performance regression unit testing: a case study." European Workshop on Performance Engineering. Springer, Berlin, Heidelberg, 2013.
2. Goldberg, Adele. "Smalltalk-80-The Interactive Programming Environment." (1984).
3. Pharo - pharo.org
4. https://en.wikipedia.org/wiki/Benchmark_(computing)
5. https://en.wikipedia.org/wiki/Continuous_integration
6. https://github.com/smarr/SMark