Bring Sanity to the GHC Performance Test-suite

Jared Weakly

June 13, 2017

Motivation for Project

The motivation for this project originally stems from ticket #12758 on the trac. To summarize, GHC has a test-suite whose purpose is, among other tasks, to point out performance regressions and prevent errors; the performance regression testing is handled specifically by the performance test-suite.

The performance test-suite's tests in place today currently requires far more manual busy-work on behalf of the programmer than is maintainable and acceptable. More pressingly, the test-suite is also very poor at noticing performance regressions. This is due, in part, to a lack of platform-specific target numbers, which leads to having to set fairly wide acceptance targets in order to accommodate variations in operating systems; there is also an inability to provide useful metrics to programmers running the tests on their home machines, due to spurious environmental differences and a lack of fine-grained control over performance target numbers.

Objectives

My goal for this project is to "Bring Sanity to the GHC Performance Test-suite." The overarching refrain for how this will be accomplished is through two main aims: first, to reduce the maintenance burden of performance tests; secondly, to increase the effectiveness of regression detection.

This will be accomplished through several tasks. The tasks will be broken up into phases to ensure that the highest priority issues are dealt with first. Phase one will be getting the essential necessities of the project done; at that point, the project can be considered to have achieved its most basic goal. The following phases will deal with quality of life enhancements, UX improvements, etc., and will be completed sequentially as time allows. I have structured this project in the following way because certain phases may take far more or less work than anticipated; this way, I will ensure at least 3 months of work without overreaching the project itself.

1. Performance metrics will be refactored out of the current testsuite.
2. Along the way, extensive documentation will be created as necessary, fleshed out where needed, and revamped with the goal of providing equal use to both fresh beginners and experienced GHC hackers.
3. The performance metrics will then be added into git-notes which will provide a metadata from which the test-suite can measure performance consistently.

(a) For ease of use, the eventual goal will be to have tooling completely automate the entire git-note process.

(b) The format of the git-note will be similar to:
`TEST_ENV,TEST,METRIC,VALUE`.

(c) `TEST_ENV` will be a string of a form similar to `ghc-ci-osx-10` (where this string denotes GHC's automated Builder for OSX.)

4. Once this setup is complete, the revamped test-suite will be considered fully functional; the remaining part of the project will be dedicated to UX improvement.

   Specifically:

   (a) Ensuring that the comparison utility is robust and fully functional.

   (b) Automate the comparison utility to give easy access to useful metrics.

   (c) Enable sane defaults for people hacking on personal computers to facilitate quick feedback loops.

5. As time permits, I will then begin on setting up and restructuring the test-suite so that it may be eventually migrated or extended relatively painlessly.

   Possible first improvements might be:

   (a) Auto generation of the global `all.T` file.

   (b) Setting up migration of the entire testing platform outside of the GHC repo and outside of GHC itself. This will allow for test plans that can build multiple version of GHC and test them to look for performance regressions or enhancements over a longer period of time.

   (c) Gradual migration towards a declarative approach similar to how Puppet or Ansible is configured.

   Currently, the test-suite is built fully in python, running homebrewed code and bespoke solutions; testing in general has come a long way since GHC's test-suite started and it shows its age in places. An appealing migration possibility would be to write tests in a declarative format similar to how Puppet, Ansible, and similar programs function. A restricted language that is not Turing-complete is often very desirable in situations like this. This would ensure very easy and consistent test-writing; tooling would be able to take advantage of a very low barrier to adding new metadata to tests. Such metadata may likely allow for deep measurements and powerful analysis in the future that are not even close to possible now.

   If such a migration is undesirable then, at the least, a test-suite that is easily extendable would be able to take advantage of a turing complete language more fully than GHC's test-suite currently does.

Midterm Deliverable

The goal is to have Phase 1 entirely complete by Midterm, with clear progress being made towards Phase 2. Additionally, Phase 3 will fleshed out further. If it seems necessary, a Phase 4 will be added, fleshed out, and pursued.

Final Deliverable

The final deliverable will be to have the test-suite performing to everyone's satisfaction with regards to ease of use, automation, extendability, and accuracy/reliability of metrics. The stretch goals will be the completion of Phase 3 and on, if others are added.

State of the System

Below follows a general overview of the state of affairs as far as locations of information and resources..

1. This is the closest thing to documentation of the testing platform that we have.
2. Most of the performance test descriptions are contained here.
3. Details of Phabricator setup; see the Build Status section for more info.
4. The previously mentioned ticket #12758 is the closest thing to an official specification of this project.
5. Inside `ghc/testsuite/driver` is the code for the test-suite executor; the majority of the code lies in the file `testlib.py`.
6. The `stats_num_field` function will be of particular importance in refactoring the testing suite.

Conclusion

Thank you for reading. As I said in the introduction, the goals of a performance test-suite are, in short, to ease and speed up testing while providing useful guiding metrics to those who wish to improve GHC's performance; the current test-suite arguably does not perform acceptably at either task. In providing a dependable regression testing platform for GHC, I hope to increase the speed and confidence with which developers of GHC can improve the compiler's performance and introduce more extensive designs, refactoring, and features. While such a project may not be a very shiny or flashy project that will be immediately noticeable to the community, I feel that it will bring about a strong improvement to Haskell nevertheless.

Please do not hesitate to contact me if you have any questions, concerns, or comments; I will be more than happy to elucidate and to make changes where necessary. I believe I can make a strong contribution to GHC and look forward to having the opportunity to give back to a wonderful community that has offered so much enlightenment to me.