Object reference mutability analysis

While working with many Java applications we noticed that after startup, long-living objects do not mutate the object graph much. The parts of the object graph they create and they are connected to seem to be mostly immutable in a range of applications.

The task for this bachelor thesis is to experimentally verify this hypothesis if and to what degree this is true on some applications by augmenting a virtual machine with the necessary code to track object graph mutability.

The results of the bachelor thesis should answer the following questions:

- think about how to extract the needed information from the application during runtime as efficiently as possible. Sketch out possibilities to track this information across garbage collection invocations that might move objects. What mechanisms in the virtual machine are involved to do this?
- how can you visualize this information adequately, what information is needed for this, and how to do this? Consider that not only summary data is interesting, also over-time visualization. Consider that the generated data is very high volume.
- statistically evaluate the observed behavior for different applications and slightly formalize the hypothesis. Prove or disprove it for the entire application. Does this hypothesis hold for the entire application(s) or is it possible to discern phases?
- is there a correlation between object age and mutability of its surrounding object graph?
- what is the the impact of the tracing augmentations on the system?
- while in this thesis you will implement only one kind of mechanism, are there alternatives and how do you expect them to behave compared to the implemented one in several aspects, e.g. accuracy, system performance impact, …?
- any other related questions that can be answered using this data set you think that may be interesting

Tasks

- change the host virtual machine to track modifications to the object graph in Java applications to the extent required by the problem description. Take care of any special requirements of your code in the virtual machine.
- use the DaCapo benchmark applications ([1], maybe a subset in coordination with the advisor) as sample applications.
- from the output, evaluate and visualize the data in a meaningful way, that can answer the original questions.

Basic knowledge and interest in memory management is recommended, e.g. previous attendance of the “System Software” lecture of the institute is more than sufficient.

Programming language/tools: Java, using the Maxine VM [2] as a host system. For data post-processing use either Java, or any other programming language/tool in coordination with the advisor. A source control system will be used, in particular Mercurial, to track your changes to the system.

As sample applications we will use the DaCapo benchmark: there are two versions of it, a newer “9.12-

bach” release and an older “2006-10-MR2” one. The latter is known to work with Maxine, while the former not; quickly evaluate whether it is feasible to do the experiments on the bach release.

Procedure, deliverables

- problem analysis including an outline of how you want to solve the problem, which changes to the host VM are required, how you are going to visualize the data to answer, which problems do you expect? (2-4 pages).
- implementation of the tasks
- after a more intensive beginning phase, short bi-weekly (later monthly) meetings about recent issues, the current progress, and planning for the next time are expected to be held.
- at the end of the bachelor's project a written report of at least 30 pages has to be handed in. It should at least contain the following contents: problem description, problem analysis, description of the implementation, evaluation and discussion of the results on the DaCapo benchmark applications, discussion of limitations and alternatives for the implementation, and further work.
- it is recommended to write the thesis in English

Advisor: Dipl.-Ing. Thomas Schatzl (thomas.schatzl@jku.at, room HF305 at the SSW institute)

References:

[1] DaCapo benchmark suite: http://www.dacapobench.org
[2] Maxine Research VM: http://wikis.sun.com/display/MaxineVM/Home