Probabilistic resource limits using StatMemprof

Guillaume Munch-Maccagnoni

Probabilistic resource limits using StatMemprof

Guillaume Munch-Maccagnoni[*]
Inria, LS2N CNRS, Nantes, France

June 4th, 2021

We present Memprof-limits[1], a probabilistic implementation of per-thread global memory limits, and per-thread allocation limits, for OCaml 4.12.

- Per-thread global memory limits let you limit the size the major heap can reach in specific parts of your program.
- Per-thread allocation limits let you bound the execution of parts of the program measured in number of allocation, analogous to the same feature in Haskell. Allocation limits count allocations but not deallocations, and is therefore a measure of the work done which can be more suitable than execution time in addition to being more portable.

Memprof-limits is probabilistic: it is based on the statistical memory profiler back-end StatMemprof (`Gc.Memprof`) added in OCaml 4.11. StatMemprof samples words in the OCaml heaps randomly, and runs custom callbacks at life events of these words: allocation, promotion, and deallocation. This can be used to implement memory profilers as libraries.

During integration into OCaml 4.11, the question arose of whether the callbacks should be allowed to raise exceptions. Such exceptions are *asynchronous*: they can arise at almost any location in the program, and for this reason are delicate to reason about. We argued at the time that raising from those callbacks could be useful to implement *resource limits* in addition to profilers, and we made the demonstration with a prototype that grew into a usable library.

The two main conceptual contributions are the following:

1. we provide Memprof-limits with a statistical analysis that the user can rely on to get guarantees about the enforcement of limits.[2]

2. we provide Memprof-limits with a guide on how to recover from asynchronous exceptions and other unexpected exceptions, thereby summarising practical knowledge acquired in OCaml by the Coq proof assistant, and also acquired in other programming languages such as Isabelle/ML—to my knowledge written for the first time.[3]

[*] Guillaume.Munch-Maccagnoni@inria.fr

[1] https://gitlab.com/gadmm/memprof-limits

[2] https://gitlab.com/gadmm/memprof-limits/-/blob/master/doc/statistical.md

[3] https://gitlab.com/gadmm/memprof-limits/-/blob/master/doc/recovering.md

The first part of the talk will focus on use-cases and usage of Memprof-limits, as well as current limitations. The second part of the talk will discuss the reasoning about programs in the presence of asynchronous exceptions; why Memprof-limits improves on the situation; and why the situation, although still imperfect, is likely to remain the same until more ambitious evolutions of the language are made possible.