

Implementing the Immix Garbage Collection Algorithm on GHC

Marco Túlio Gontijo e Silva
marcot@marcot.eti.br

Supervisor: Carlos Camarão de
Figueiredo
Co-Supervisor: Fernando M. Q.
Pereira
UFMG

November 19, 2010



Introduction — About Haskell



- Haskell is one of the most popular functional programming languages
- Haskell provides developers with managed memory

- In order to deliver this abstraction to developers, Haskell uses a garbage collector (GC)
- Immix is a new GC algorithm, presented in the Conference on Programming Languages, Design and Implementation (PLDI'08)
- The Glasgow Haskell Compiler (GHC) is the *de facto* standard Haskell compiler



Introduction — Motivation

- The garbage collection has a great impact on the performance of the executables produced by GHC
- GHC uses two GC algorithms — copy collection and mark (sweep) compact collection — depending on the amount of memory used by the mutator (application)
- Immix GC outperforms both approaches

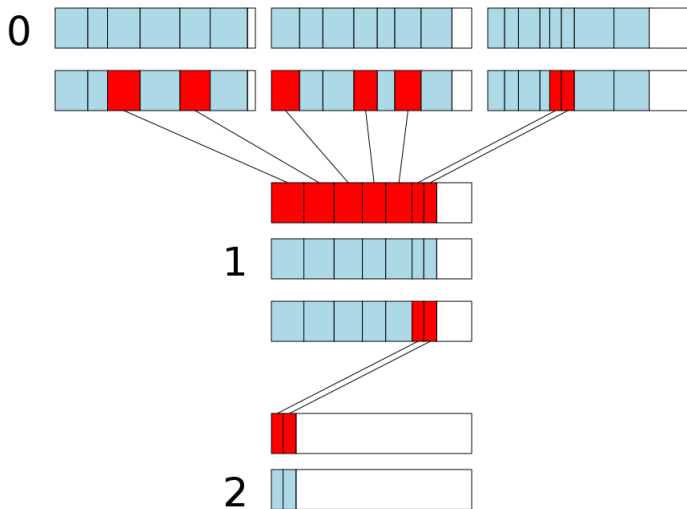


- Improve performance of GC in GHC
 - Implement Immix GC in GHC
 - Improve other aspects of the GC
- Improve the documentation about the GHC GC in the GHC wiki

- Tracing GC: objects pointed by root objects are considered alive
- Memory is allocated in blocks, which allows flexibility
- Memory access inside blocks is sequential

- Objects are created in a nursery (generation 0)
- If they are still alive in the next garbage collection, they are moved (evacuated) to the next generation
- Assumes the death probability of younger objects is higher
- Measurements show that the best performance can be achieved with 2 generations

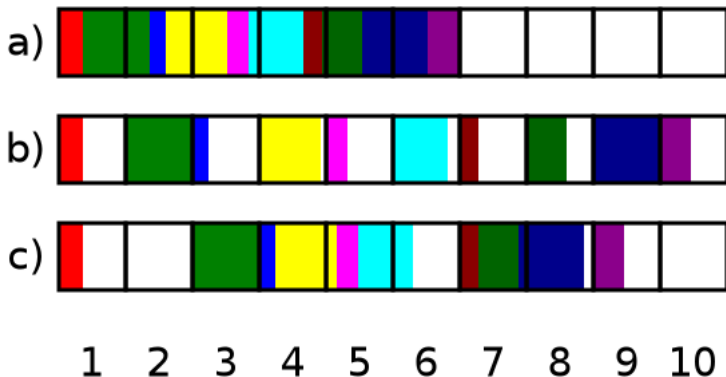
Theoretical reference — Generational GC



- Copy Collection
 - The live objects are copied to another block in the same generation
 - Default in GHC, until 30% of the maximum heap size is reached
 - Similar to the generational GC
- Mark-sweep
 - Objects are marked and when a block contains no object, it's freed
 - Needs manual enablement in GHC
- Mark-compact
 - A combination of copy collection and mark-sweep
 - Objects are marked, then copied to another area of the same block

- Blocks are divided in lines
- Objects are allocated across lines: a and c
- Free lines are used to allocate new objects: b
- Conservative marking: there's no need to check for the object size
- Currently, only objects smaller than one line are considered

Theoretical reference — Immix



Methodology

- I wrote a project to the Google Summer of Code
- I studied the Immix GC and GHC, via documetation on the wiki and the source code
- I've submitted several patches of comments to the GHC source code, which are now in the repository
- I started the implementation of Immix in GHC:
 - Memory is freed in lines
 - Free lines are represented by a linked list
 - Allocation of new objects smaller than the size of a line use these lines in major GCs
- I described my work in a page of the GHC wiki



- The Immix implementation is functional and bug-free, but it could be improved
 - Allocation should use lines even in minor GCs
 - The code should be faster than the original GHC GC