Exploring the potential benefits of the Integrated Hardware Garbage Collector (IHGC) and its applications

Maarten van der Heijden, Supervised by David May

University of Bristol, Bristol

Introduction

Modern high level languages such as Java rely on automatic dynamic memory management. This scheme is implemented in software and has been known to contain bugs (as software often does). The IHGC can solve this problem by implementing garbage collection in hardware. The object-based model the IHGC follows also has some interesting security implications by preventing invalid pointers, buffer overflows and accidental memory leaks. These problems are the source of many bugs in programming. The newly proposed architecture can therefore significantly increase security. There are also energy consumption, reliability and robustness properties that may be benificial for embedded devices.

Garbage Collection

In computer science, garbage collection (GC) is a form of automatic memory management. The garbage collector, or just collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program.

Marking and Sweeping Away Garbage To determine which objects are no longer in use, the Garbage Collector runs what is very aptly called a mark-and-sweep algorithm. It's a straightforward, two-step process:

- The algorithm traverses all object references, starting with those known to be in use, and marks every object found.
- All of the memory that is not occupied by marked objects is reclaimed.
- The memory is then reorganized in such a fashion that there will be 1 block of used memory and one block of unused memory

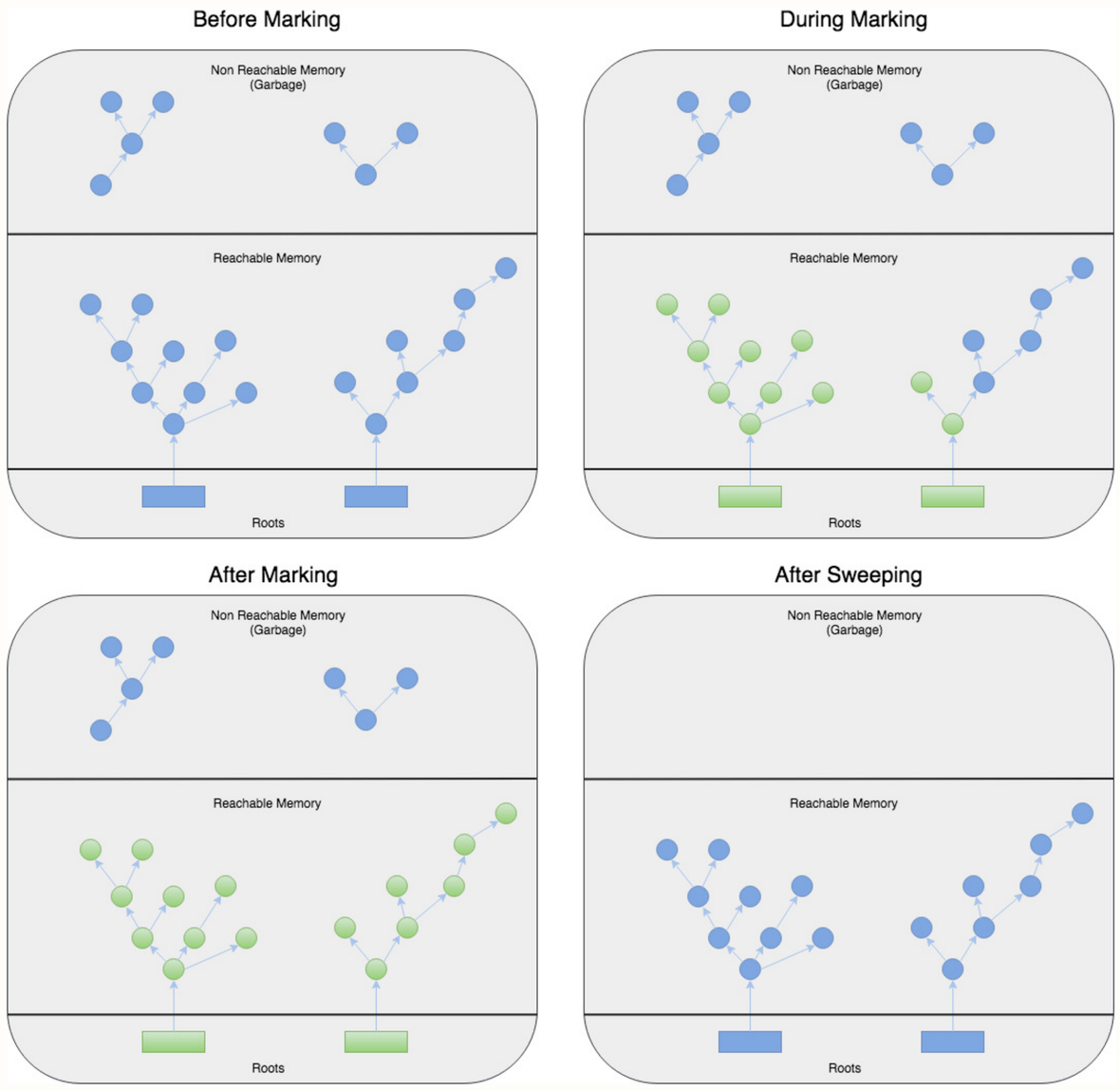

Figure 1: Mark and Sweep Algorithm

Internet of Things (IoT)

The Internet of things (IoT) is the network of physical devices embedded with electronics, software, sensors, actuators, and connectivity which enables these objects to connect and exchange data. The proposed architecture may have significant benefits for Internet of Thing (IoT) devices.

Areas of interest

Error Containment

Detecting and containing faults is very important when there are actuators (robotics, autonomous vehicles) involved. This architecture has the potential to securely contain and handle these errors which will improve the reliability of programs run on the system.

Security

The security of internet of things devices is becoming one of the main challenges manufacturers are facing. The architecture proposed has security features such as checking the validity of memory access and the guarantee that data which is garbage collected will not be recoverable. These are very desired properties for small embedded devices.

Energy

The garbage collector ensures that all data is kept in the lower part of the memory. This means that the unused memory can be powered down which could significantly lower the energy consumption

Continuous Execution

Some garbage collectors have to stop program execution in order to free memory. They are not suitable for devices such as sensors which have to operate continually. The integrated garbage collector is designed such that the execution of programs does not have to be stopped for garbage to be collected.

Reliability

Garbage collectors which are implemented in software can contain errors/bugs in their implementation which can cause programs to fail or for memory leaks to occur.

Programming Languages

The architecture lends itself to the use of object oriented programming languages like java on embedded devices. This is usefull for producing verifiable software. If both the software is verified and the garbage collector is verified (This is currently being investigated) there is a huge potential for creating provable correct programs.

Project

Current state of the project

- A compiler to target the new architecture has been developed
- The simulator for the memory module and the cpu with the Integrated Hardware Garbage Collector is being developed.

Next steps

- Finish and test the compiler and simulator
- Write interesting programs which showcase the benefits this architecture potentially has

Reach goals

- Create a multi-core version of the simulator
- Write parallel programs for the new simulator and explore any potential benefits the scheme gives