Bachelor's Thesis

Memory Analysis on the Object-level using JVMTI Object Tagging and Heap Dumps

Dipl.-Ing. Markus Weninger
Institute for System Software

P +43-732-2468-4361
F +43-732-2468-4345
markus.weninger@jku.at

Student: Thomas Sulzbacher
Advisor: Dipl.-Ing. Markus Weninger, BSc.
Start date: March 2019

AntTracks comprises a modified Java VM based on the Hotspot VM, i.e., AntTracks VM, and an offline post-processing analysis tool, i.e., AntTracks Analyzer.

The VM's aim is to allow tracking of an application's entire life cycle by writing information about certain events to a trace file. These events include object allocations, object movements by the garbage collector, pointers between the objects and so on.
Such an event trace can then be analyzed in the offline post-processing tool. Based on the information parsed from the trace file, the tool can reconstruct a heap snapshot for any garbage collection point and, in addition to that, it can analyze the application's memory evolution over time on the object-level. This means that the AntTracks Analyzer can determine the lifespan of single objects and can thus perform detailed analyses on single objects over time.

Since a custom VM is needed to create such detailed trace files, most other memory analysis tools offer less detailed analyses by restricting their analyses to a single heap snapshot, reconstructed from a heap dump file. Even though heap dumps (often called HPROF files) are easy to obtain from running applications, they have a major drawback: They are not suitable for memory analysis over time. Even if multiple heap dumps are created at different points in time, they do not identify objects across multiple heap dumps, i.e., they do not provide object identity. Thus, it is not possible to determine which objects have died or have been newly allocated between two heap dumps, making it impossible to perform detailed analyses on single objects over time.

The goal of this thesis is to implement a heap dumper. Instead of needing a custom VM, this dumper should be implemented as a native agent that can be attached to a running Java application, collecting its data using the Java Virtual Machine Tool interface (JVMTI)[1]. The dumper should halt the application and create heap dumps in certain intervals. Yet, compared to normal HPROF heap dumps, these dumps should support memory analysis over time on the object-level, i.e., they should provide object identity. To achieve this, JVMTI's functionality to set *tags* on objects should be exploited. On every dump, the whole tree of live objects should be traversed (e.g., using the JVMTI function *IterateOverReachableObjects*). If an object is encountered the first time, it should be tagged with a unique ID.
During a dump, at least three different event types have to be recorded to be able to reconstruct a heap snapshot: (1) new_obj: when an object is encountered for the first time, its ID and its type should be recorded, e.g., "new_obj 17 Foo[]" encodes that an object (which is is of type Foo[]) was encountered the first time, and it was assigned the tag / ID 17; (2) surv_obj: when an object is encountered that has been tagged in a previous dump, its ID should be recorded, e.g., "surv_obj 25"; (3) ptr: a minimalistic file format dumps one event per pointer between two objects, e.g., "ptr 13 18" encodes that the object with ID 13 references the object with ID 18.

Additionally, a parser for this new file format has to be written in Java. This parser should be able to read a heap dump file, event by event, notifying a list of listeners about the newly read event.
Writing a listener that processes the events and maps them to the AntTracks Analyzer’s heap snapshot data structures is out of the scope of this thesis.

Modalities:
The progress of the project should be discussed at least every two weeks with the advisor. A time schedule and a milestone plan must be set up within the first 3 weeks. It should be continuously refined and monitored to make sure that the thesis will be completed in time. The final version of the thesis must be submitted not later than 18.09.2019.

[1] JVMTI: https://docs.oracle.com/javase/8/docs/platform/jvmti/jvmti.html