About Aprof

Aprof is a Java memory allocation profiler with very low performance impact on the profiled application that can be used (and is used) on highly-loaded server-side applications in production environment. It acts as an agent which transforms class bytecode by inserting counter increments wherever memory allocation is done and tracks a precise size of allocated objects in bytes. It also keeps limited information about allocation context to aid in finding the memory allocation bottlenecks.

GitHub Repository	https://github.com/Devexperts/aprof
Public Maven repo	https://bintray.com/devexperts/Maven/aprof (check agent/<version>/agent-<version>-bin.zip for distribution)
License	GPLv3
Contact	dxlab@devexperts.com

Using aprof

Download the latest version of aprof distribution, unzip, copy "**aprof.jar**" to your application's directory and run the application with an additional JVM argument:

```
java -javaagent:aprof.jar <your-application-options>
```

Run it and watch for "**aprof.txt**" file that will be updated every minute with memory allocation details reports for your application for the last minute (with totals in the separate section at the end) in a human-readable form. There will be somewhat slower start-up time due to transformation and a lot of extra memory allocations by aprof itself, so wait a couple of minutes until your application and aprof warm up (aprof allocates all the memory it needs to operate during start-up and warm-up), start your application's workload, wait some more time until your application warms up under load, and use reports in "**aprof.txt**" to find memory allocation bottlenecks in your application under load.

To get help on configuration parameters, run

```
java -jar aprof.jar
```

Do not rename agent file "**aprof.jar**", because it is configured by its name in manifest and will not work when renamed.

Features overview

- Fast (can be used and *is designed to be used in production*)
- Provides detailed information (can be used to identify problematic pieces of code)
- Configurable level of detail (as a compromise between speed and level of detail)
- Open Source (see, learn, improve)

Known issues

In the current version (build 31):

- **Does not work on JVM 1.8.0_20 and on JVM 1.7.0_65 (VerifyError on transformed classes) due to a bug in JVM:** http://bugs.java.com/view_bug.do?bug_id=8051012
  For Java 8, use 1.8.0_25 or later. For Java 1.7.0 use 1.6.0_60 or turn off verifier with "-noverify" option.
- Does not work on some releases of 1.6 (1.6.0_21 crashes with FATAL ERROR), but otherwise works in Java 5 to Java 8.
- Does not fully analyze inheritance hierarchy when tracking configured method invocations. It does not intercept locations of invocations of tracked methods that are performed via a subtype that inherits the tracked method, It does misleadingly report invocation locations of a non-tracked method that happen to go via super-type that declares a tracked method for some other implementation of this type where this method is tracked.
- Reports memory allocations that are performed in Java 8 during lambda capture of variables from the scope as belonging to a location in some internal "xxx$Lambda$xxx" class, not in the actual source code method that creates this lambda.
- Does not include Java 8 run-time library method (collections, streams, etc) in the default list of tracked methods.

Origin and Goals

Java VM has an option *-Xaprof* printing how many instances of classes were allocated during the lifetime of the application and how much memory they occupied in total and per instance. The option has no performance impact due to the fact that counting takes place during garbage collection. The only drawbacks are occasional overflows of counters and the absence of any information on locations where the object allocations take place.

Aprof aims to overcome drawbacks of Java VM option *-Xaprof* by:

- collecting information on locations where the object allocations take place.
- providing accurate profiling results.
- having a very low performance impact (to be safely used in production environments).

When to use Aprof

Aprof should be used when the application spends a lot of time in garbage collection. Due to the specifics of Java VM, objects are allocated so fast that CPU profilers are unable to pinpoint the allocation hotspot. The only way is to use memory allocation profilers.

There are many good memory allocation profilers. Unfortunately, in order to find locations where object allocations take place, they do the following:

- on object allocation, a stack-trace is taken. This takes significant time and generates some garbage.
- in order to lessen performance impact of taking stack-traces, sampling is used (i.e. not every object allocation is recorded). This leads to inaccurate profiling results.

As a result, memory allocation profilers leave us with both inaccurate profiling results and an impact on the profiled application.

Aprof gets accurate profiling results and finds locations of object allocations at the same time being garbage-free and having a very low performance impact.

How it works

See presentation on Joker Conference 2014: http://www.slideshare.net/elizarov/aprof-jocker-2014