AntTracks VM

The AntTracks VM can generate memory traces (i.e., a trace file, a symbols file, and a class-def file). These memory traces can later be analyzed for anomalies in the AntTracks Analyzer.

How to run the AntTracks VM:

`/pathto/anttracks/bin/java <VM flags> -jar <target>`
Just like any other Java application start

One of these two VM flags must be added to generate traces:
`-XX:+TraceObjects`
Generate trace without pointer info
`-XX:+TraceObjectsPointers`
Generate trace with pointer info (= references between objects)

Further helpful flags:

`-XX:TraceObjectsTraceFile="<pathtofile>"`
Where to store the trace file (default: "./trace")
`-XX:TraceObjectsSymbolsFile="<pathtofile>"`
Where to store the symbols file (default: "./symbols")
`-XX:TraceObjectsClassDefinitionsFile="<pathtofile>"`
Where to store the class def. file (default: "./classdefinitons")

Example:

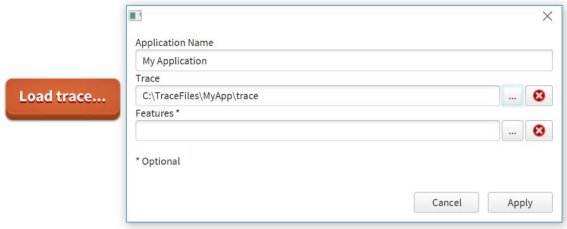
```
~/workspace: /opt/anttracks/bin/java -XX:+TraceObjectsPointers -jar /home/myuser/Application.jar
```

This will create three files in the folder ~/workspace: "trace", "symbols", "class_def". These three files make up all the information collected during the run. Keep them together in a folder and open the "trace" file in the AntTracks analyzer to inspect the application.

AntTracks Analyzer

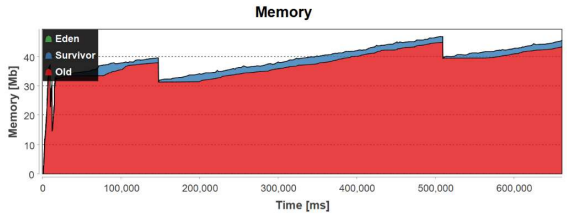
The AntTracks Analyzer can load AntTracks trace files to analyze them.
Use the scripts `runAntTracksTool.sh` / `runAntTracksTool.bat` to start the Analyzer.

Trace loading:

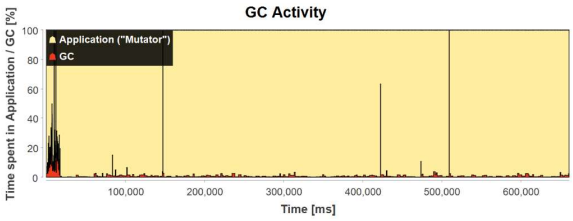


Load a trace file on the start screen.

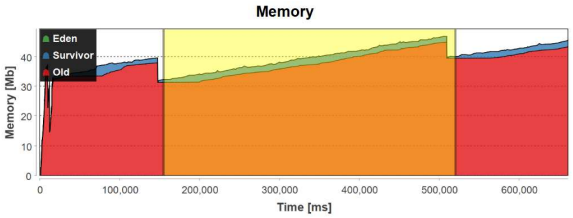
Overview:



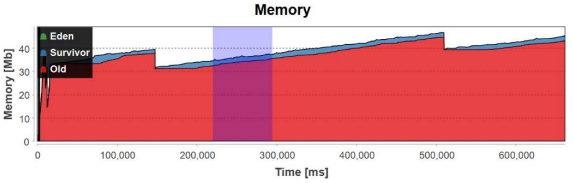
This chart shows the memory consumption (one entry at the end of every GC). The red area is the GC's OLD generation, i.e., objects that have survived for a longer time.



This chart shows the GC activity. More red = more GC activity.



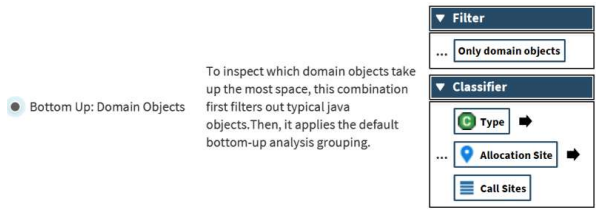
Click on the charts to select a single point in time.
Click a second time to select the time span between the two selected points in time.



Dragging the mouse to the right zooms in.
Dragging the mouse to the left zooms out.

Heap State Analysis:

If you select a single point in time, you can open the Heap State Analysis. This is used to inspect the content of the heap at that given point in time.

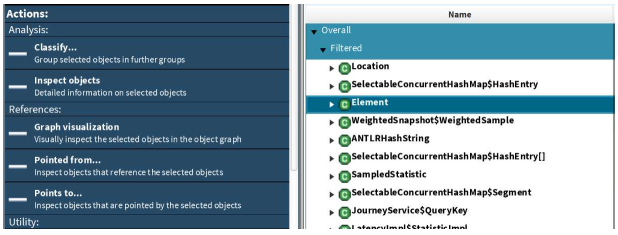


You can select pre-defined classifier and filter combinations, e.g., "Bottom Up: Domain Objects", or you can use the expert mode to select your own classifiers.
Filters specify if certain objects should be excluded from the analysis and classifiers define based on which properties the heap objects should be grouped.

Name	Objects	Shallow size	Retained size
Overall	875,934	45.9 MB	45.9 MB
Filtered	160,333	5.6 MB	16.6 MB
Location	127,697	4.1 MB	7.4 MB
SelectableConcurrentHashMap$HashEntry	7,521	300.8 kB	6.6 MB
Element	6,724	537.9 kB	6 MB
EhcacheTransactionalDataRegion.put	5,789	463.1 kB	5.3 MB
EhcacheGeneralDataRegion.put	935	74.8 kB	643.3 kB
WeightedSnapshot$WeightedSample	3,909	125.1 kB	125.1 kB

The classification result is shown in a tree-table view.
The "Objects" column shows the number of objects a given object group.
The "Shallow size" column shows how many bytes are taken up by the object group itself.
The "Retained size" column shows how many bytes are kept alive / owned by the object group.

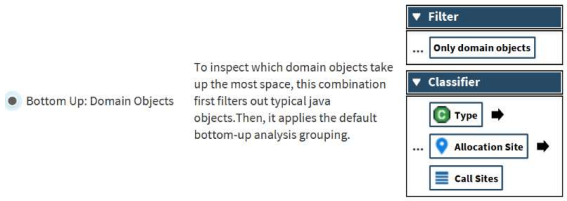
In this example, the objects are first grouped by type. You can expand an object group to see the results of the second classifier, e.g., the allocation site.



If you select an object group, you may also choose different actions.
For example, "Graph visualization" allows you to explore the references between the objects and to explore graphically which GC roots keep the selected objects alive.
Another useful feature is "Classify…". It can be used to apply another classifier on the currently selected group.

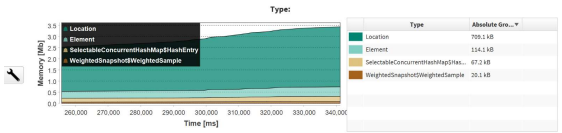
Object Group Trend Analysis

If you select a time window, you can use the Object Group Trend Analysis to inspect which objects accumulated the most in the given time span.

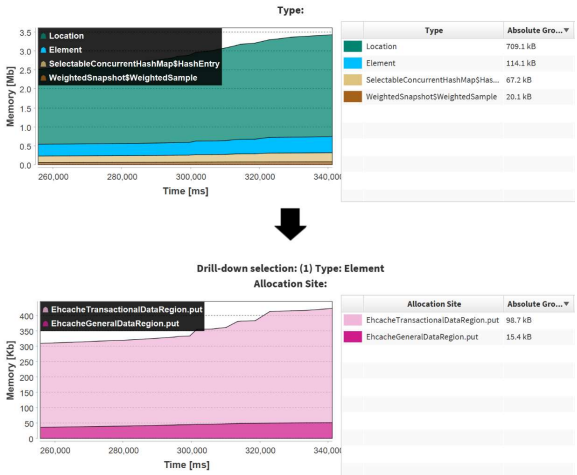


First, like in the heap state analysis, you can select filters and classifiers that should be used to filter and group the heap objects.

The default settings should be sufficient and don't have to be changed for default analysis.



By default, only a single chart is shown. This chart shows the evolution of the object groups created by the first classifier, e.g., type.
The goal is to detect (a) certain group(s) the show suspicious growth.

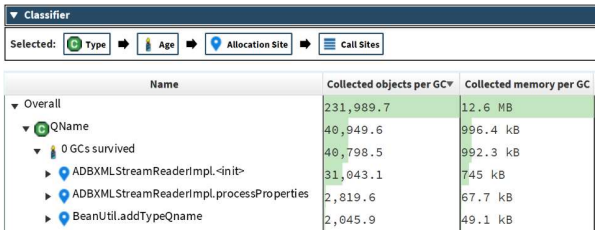


To inspect an object group in more detail, you can click on the chart series or the entry in the table.
This "drills down" into the group, i.e., it shows the results of the second classifier for the given group.
In this example, we clicked on the type "Element" (highlighted blue in the top chart), and the second chart shows the allocation site, i.e., where these Element objects have been created.

Short-lived Objects Analysis

If you select a time window, you can use the Short-lived Objects Analysis to inspect which objects are created and die in high quantity in a short period of time.

The overview page shows different information on the garbage collections themselves and which objects died the most in the selected time window (see the pie charts above).



To get a more detailed view on which objects died, and how long the lived, you can switch to the "Collected objects" tab.
This view tells you how many objects / bytes of a given type have been collected on average during GCs, how long these objects survived, and where these objects have been created.