Analyzing Memory Errors in Production

Markus Weninger

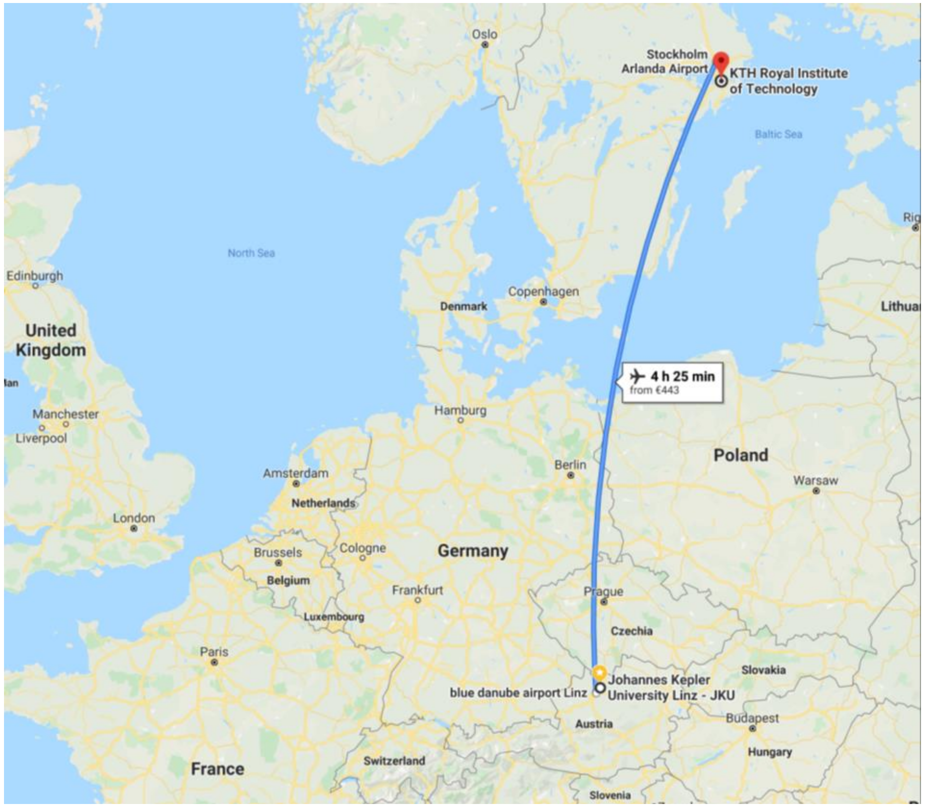
Johannes Kepler University Linz, Austria
Institute for System Software

JOHANNES KEPLER
UNIVERSITY LINZ

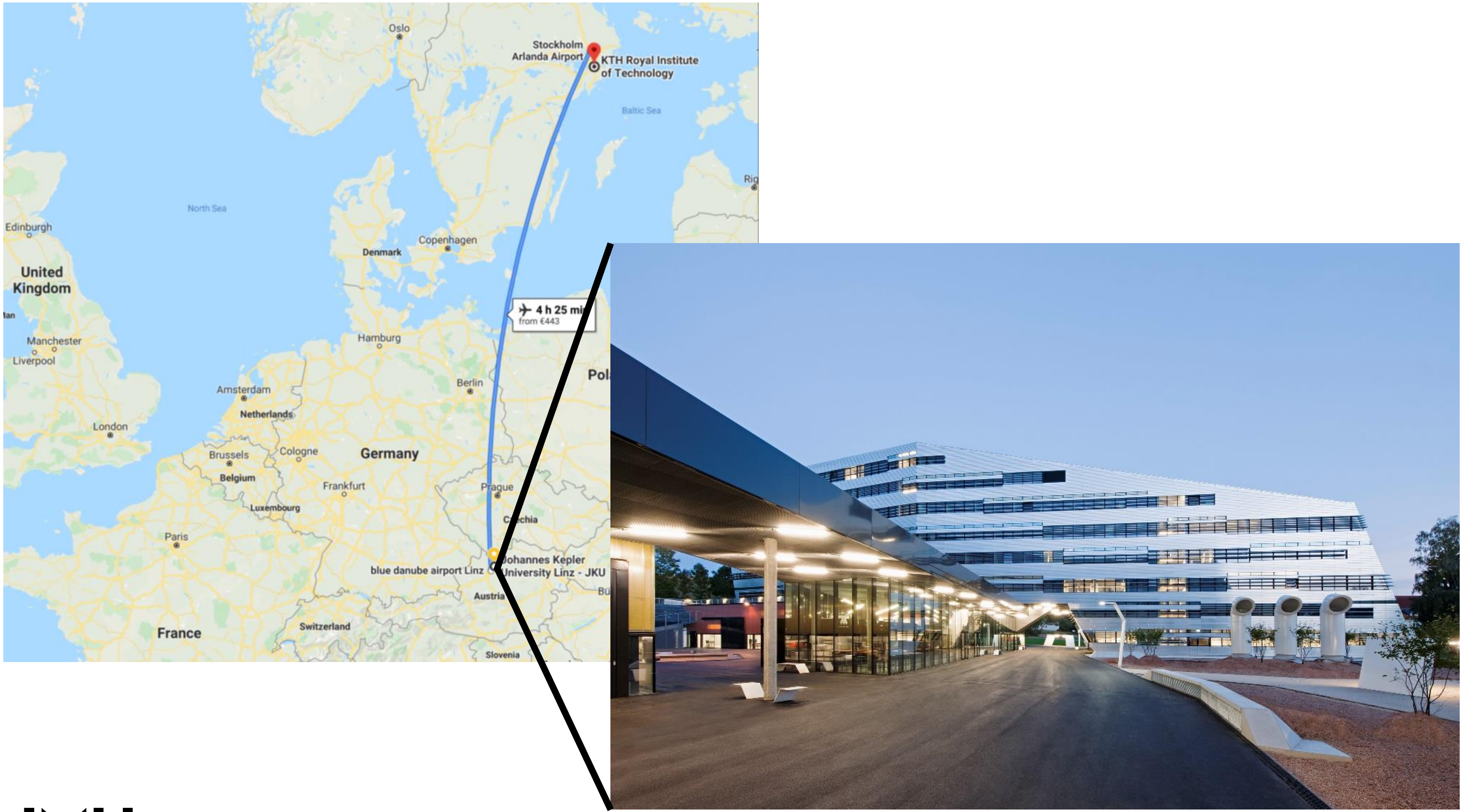
dynatrace

WHO ARE WE?

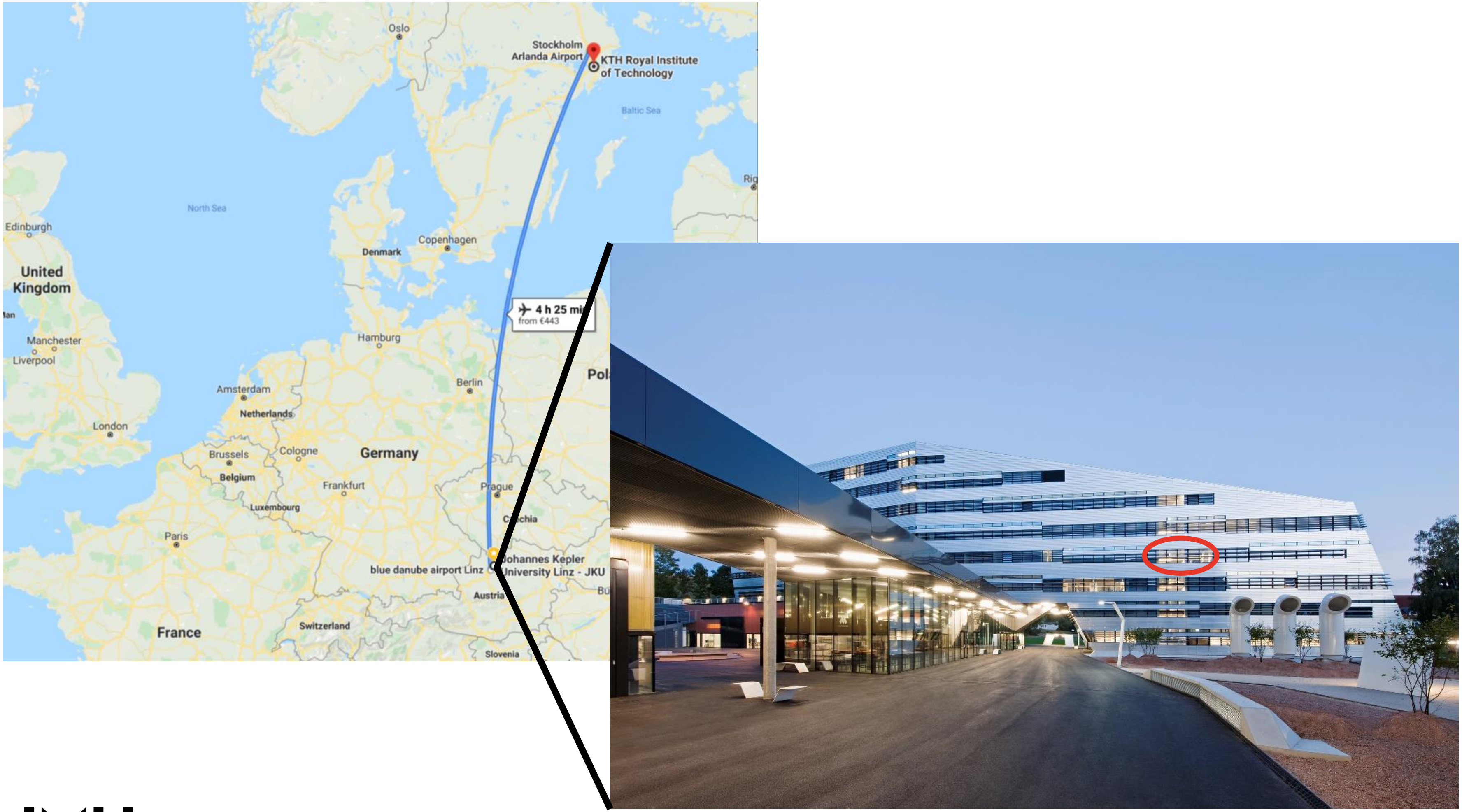
WHO ARE WE?


Stockholm Arlanda Airport
KTH Royal Institute of Technology
Baltic Sea
Oslo
North Sea
Edinburgh
United Kingdom
Manchester
Liverpool
London
Copenhagen
Denmark
4 h 25 min
from €443
Hamburg
Berlin
Poland
Warsaw
Amsterdam
Netherlands
Brussels
Belgium
Cologne
Germany
Frankfurt
Luxembourg
Paris
France
Prague
Czechia
Slovakia
Johannes Kepler University Linz - JKU
blue danube airport Linz
Austria
Budapest
Hungary
Switzerland
Slovenia
Lithua
Rig

WHO ARE WE?

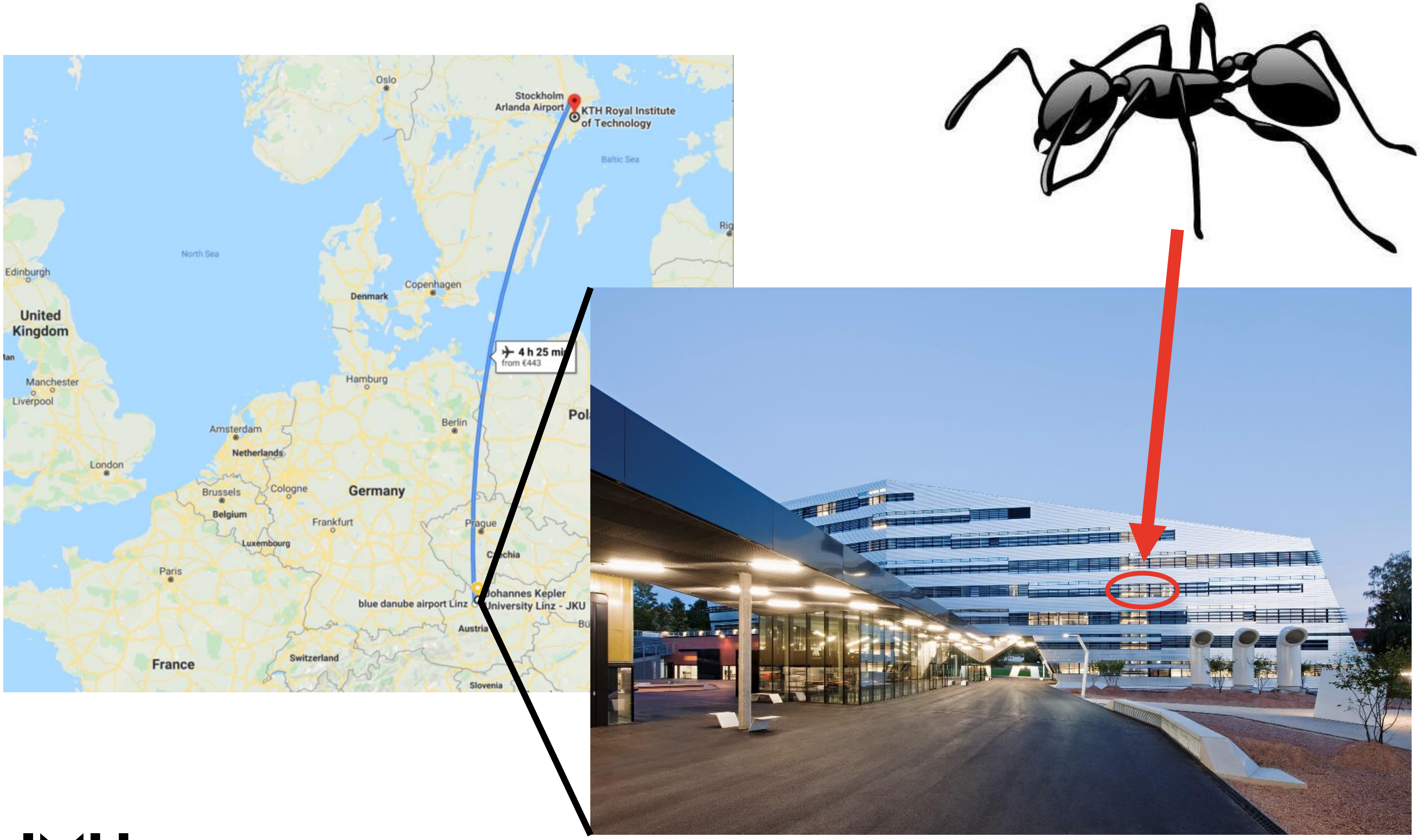

Oslo
Stockholm Arlanda Airport
KTH Royal Institute of Technology
Baltic Sea
Rig
North Sea
Edinburgh
United Kingdom
Denmark
Copenhagen
4 h 25 mi
from €443
Manchester
Liverpool
Hamburg
Berlin
Pol
Amsterdam
Netherlands
London
Brussels
Cologne
Germany
Belgium
Frankfurt
Prague
Luxembourg
Czchia
Paris
Johannes Kepler University Linz - JKU
blue danube airport Linz
Austria
France
Switzerland
Slovenia

WHO ARE WE?



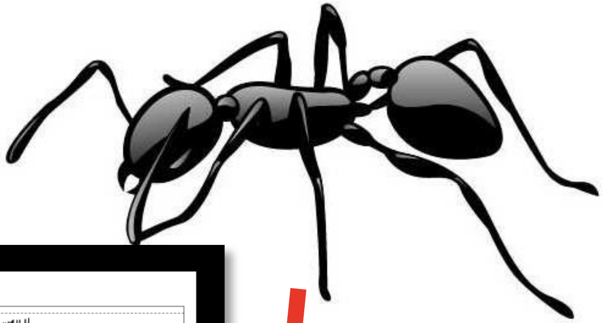
Oslo
Stockholm Arlanda Airport
KTH Royal Institute of Technology
Baltic Sea
Rig
North Sea
Edinburgh
United Kingdom
Copenhagen
Denmark
4 h 25 mi
from €443
Manchester
Liverpool
Hamburg
Pol
Berlin
Amsterdam
Netherlands
London
Brussels
Cologne
Germany
Belgium
Frankfurt
Prague
Luxembourg
Czechia
Paris
Johannes Kepler University Linz - JKU
blue danube airport Linz
Austria
France
Switzerland
Slovenia

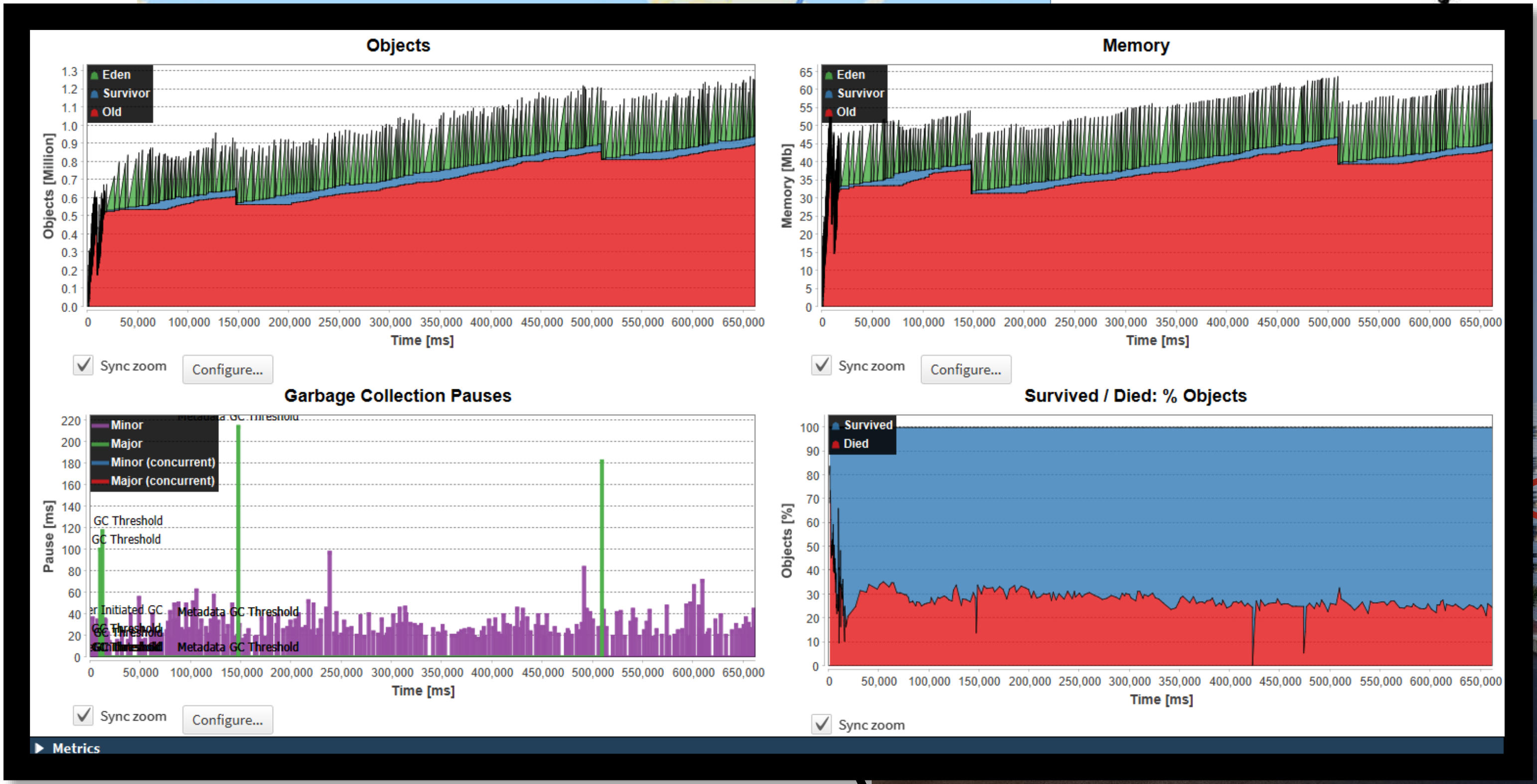
WHO ARE WE?


Oslo
Stockholm Arlanda Airport
KTH Royal Institute of Technology
Baltic Sea
North Sea
Edinburgh
United Kingdom
Copenhagen
Denmark
4 h 25 mi
from €443
Manchester
Liverpool
Hamburg
Berlin
Pol
Amsterdam
Netherlands
London
Germany
Brussels
Cologne
Belgium
Frankfurt
Prague
Luxembourg
Czechia
Paris
Johannes Kepler University Linz - JKU
blue danube airport Linz
Austria
France
Switzerland
Slovenia

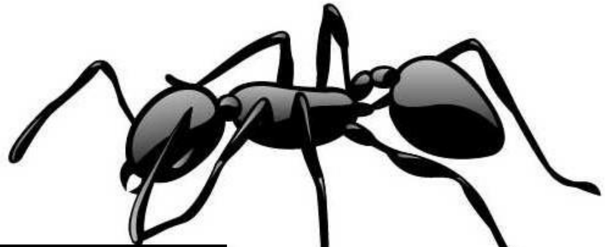
WHO ARE WE?

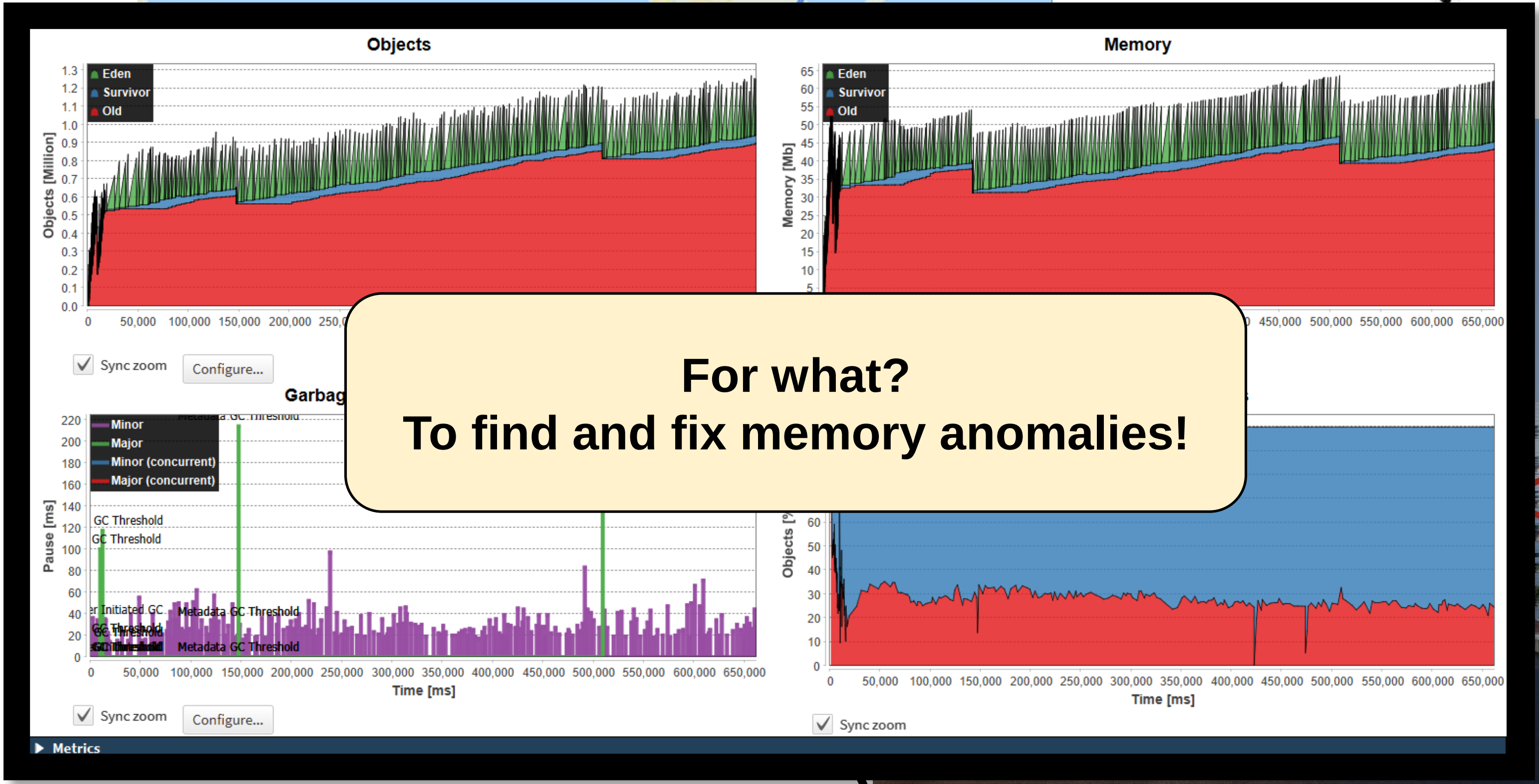
Oslo
Stockholm Arlanda Airport
KTH Royal Institute of Technology
Baltic Sea




Objects
Eden
Survivor
Old
Objects [Million]
Time [ms]
Memory
Memory [Mb]
☑ Sync zoom
Configure...
Garbage Collection Pauses
Minor
Major
Minor (concurrent)
Major (concurrent)
GC Threshold
Metadata GC Threshold
Pause [ms]
Survived / Died: % Objects
Survived
Died
Objects [%]
▸ Metrics

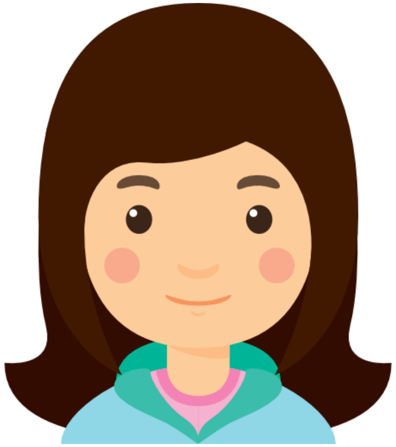
WHO ARE WE?





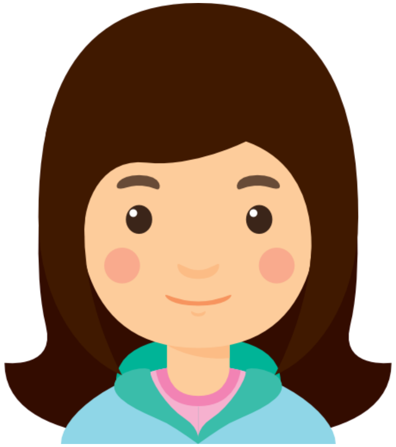
For what?
To find and fix memory anomalies!

MOTIVATION



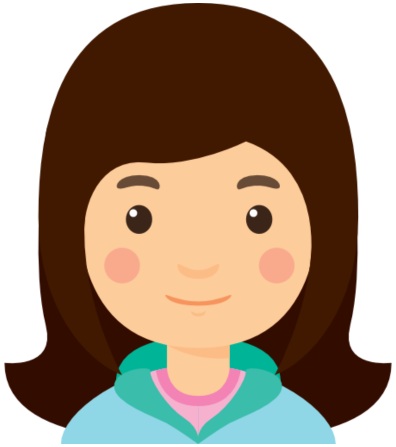
MOTIVATION

Jake



MOTIVATION

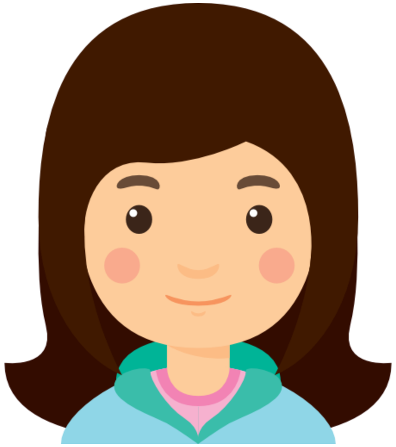
Jake



July

MOTIVATION

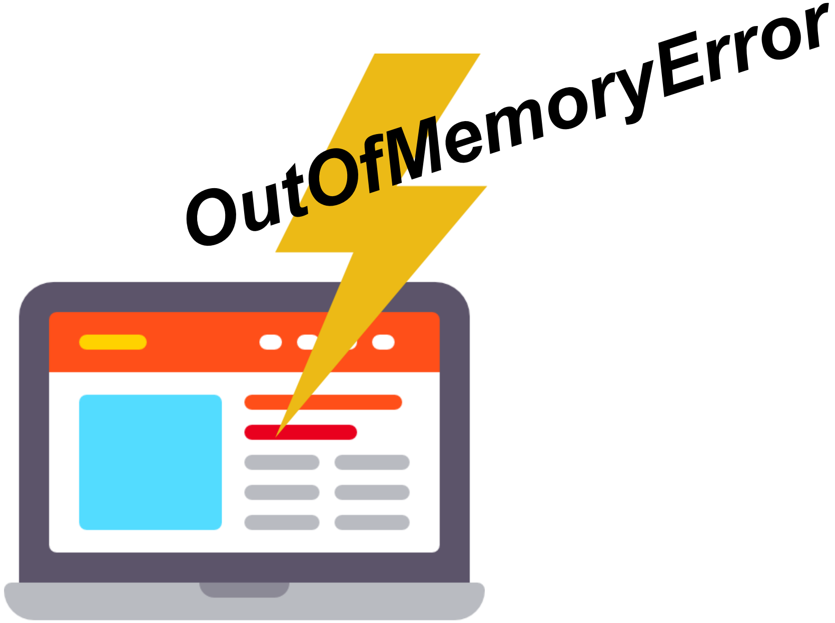
Jake

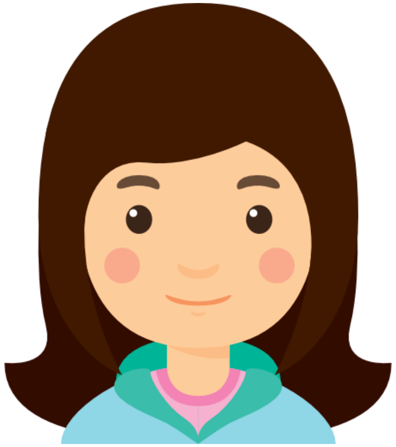


July

MOTIVATION

Jake

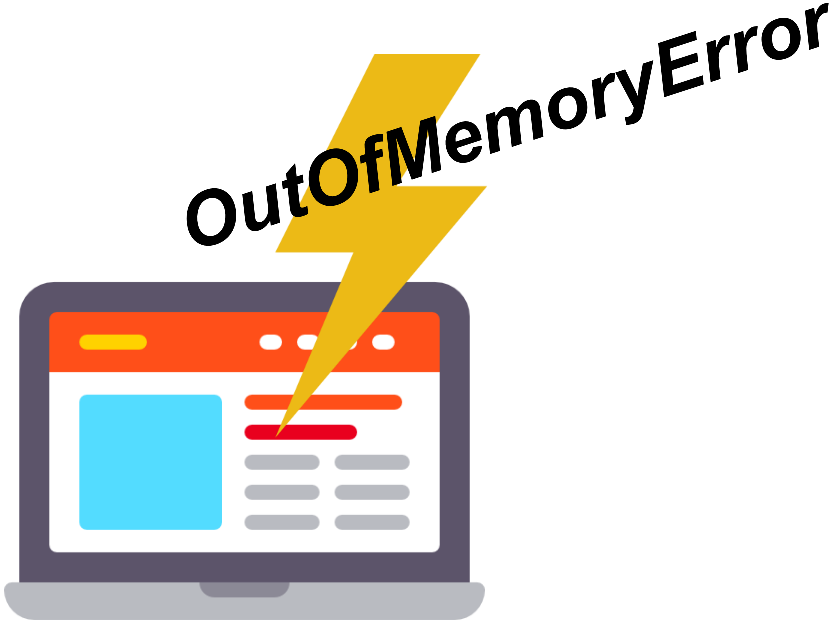


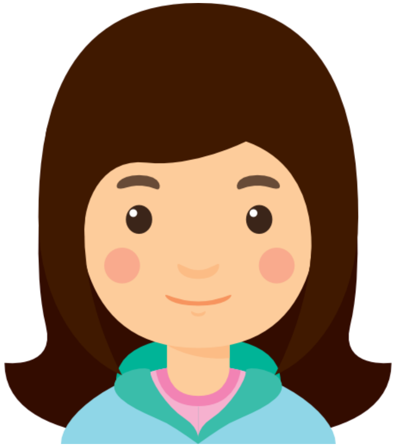


July

MOTIVATION

Jake

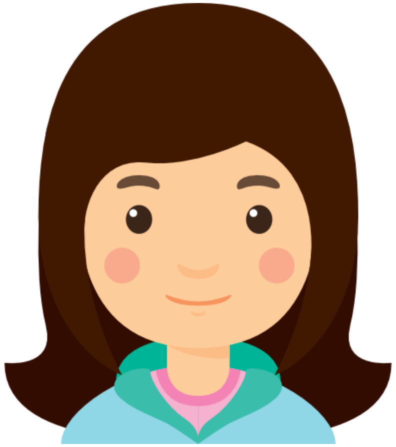




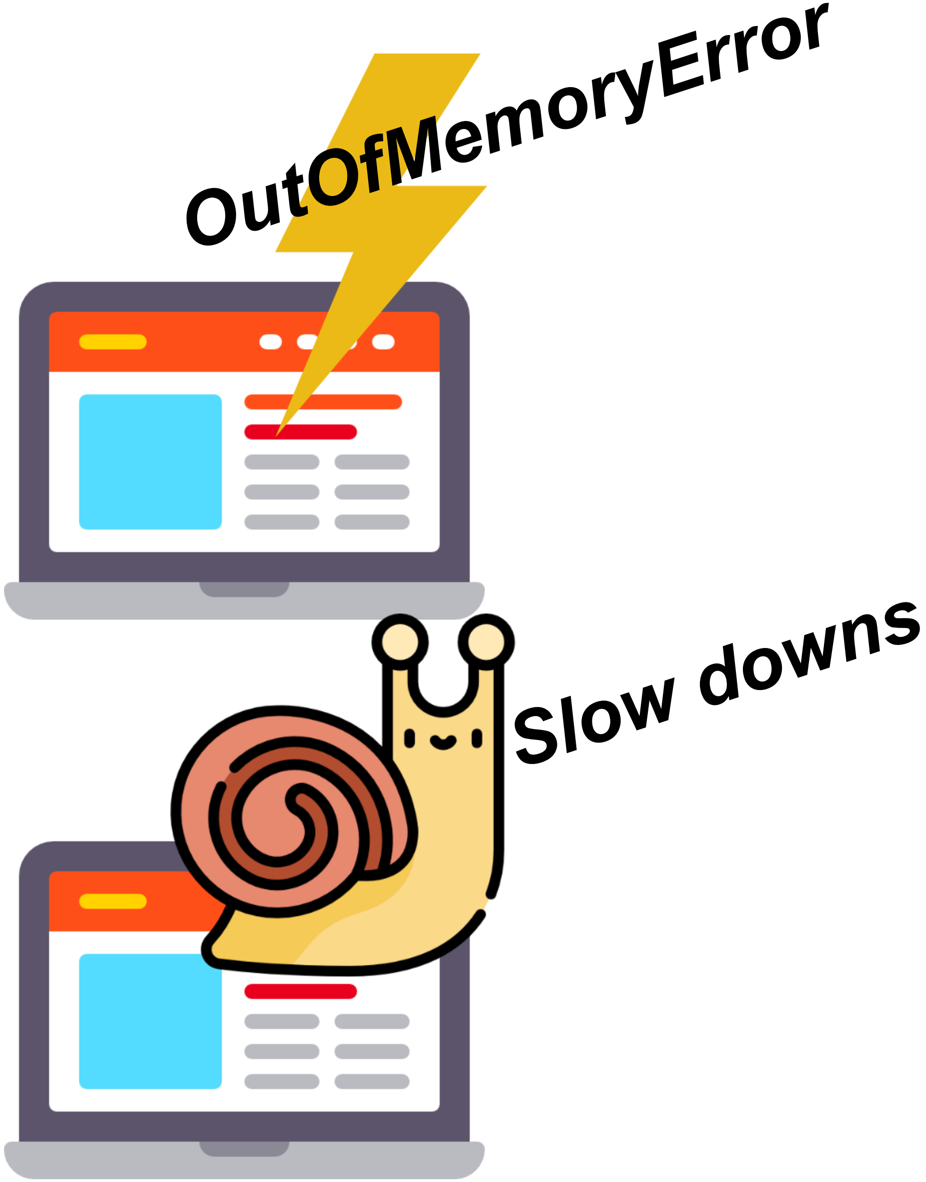
July

MOTIVATION

Jake



July



MOTIVATION

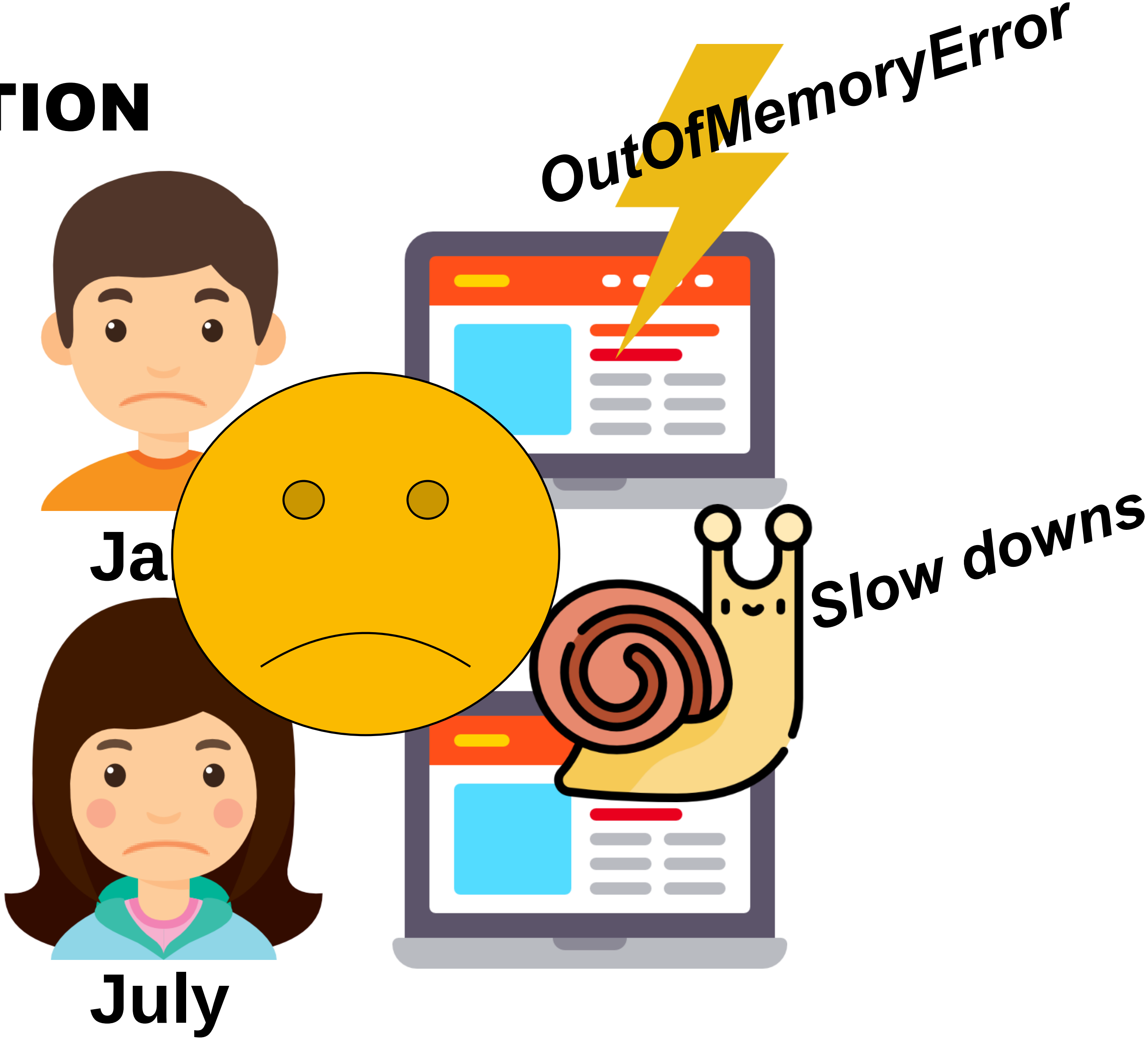
Jake

July

OutOfMemoryError

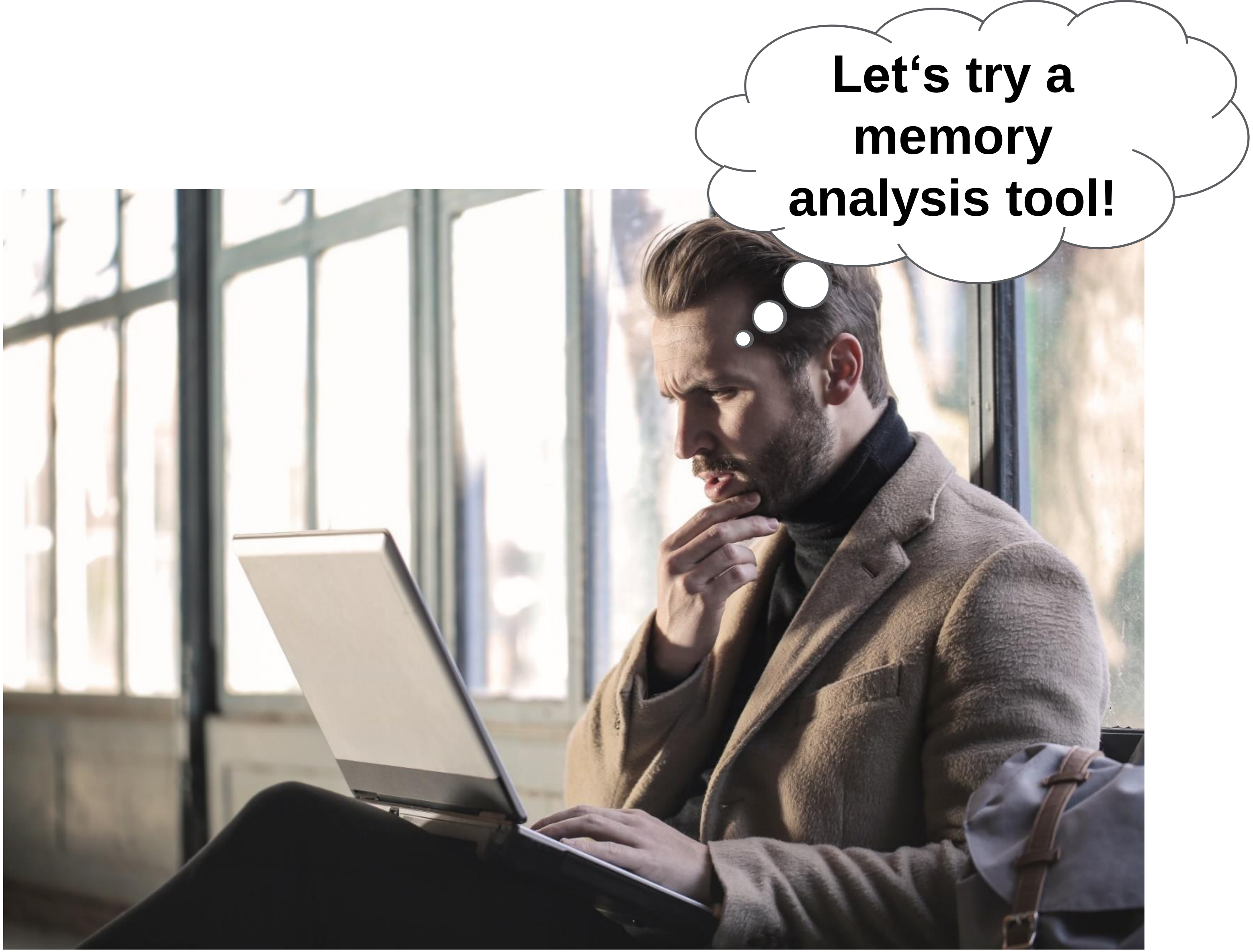
Slow downs

MOTIVATION



*** not actually Jake**

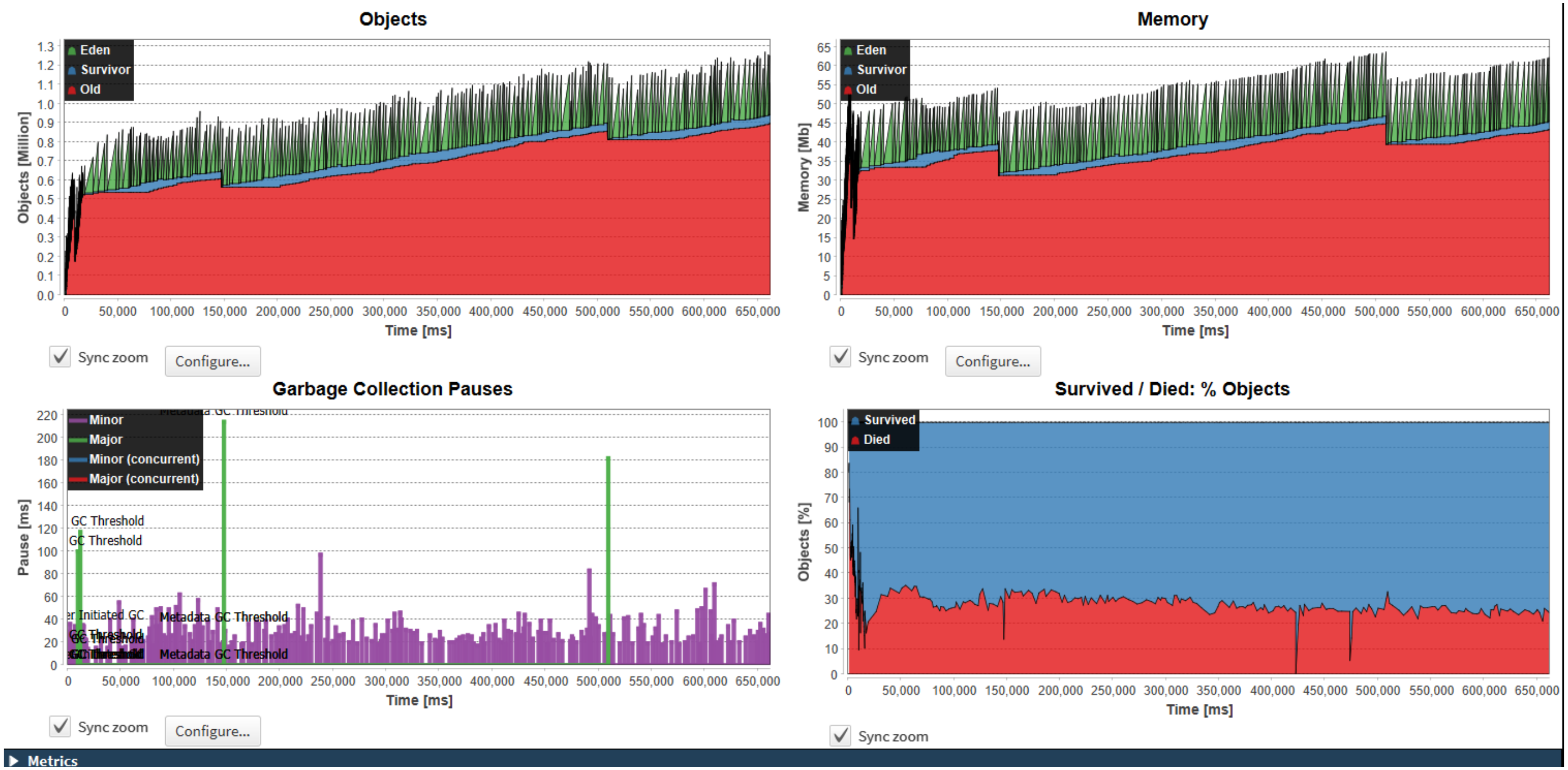
*** not actually Jake**



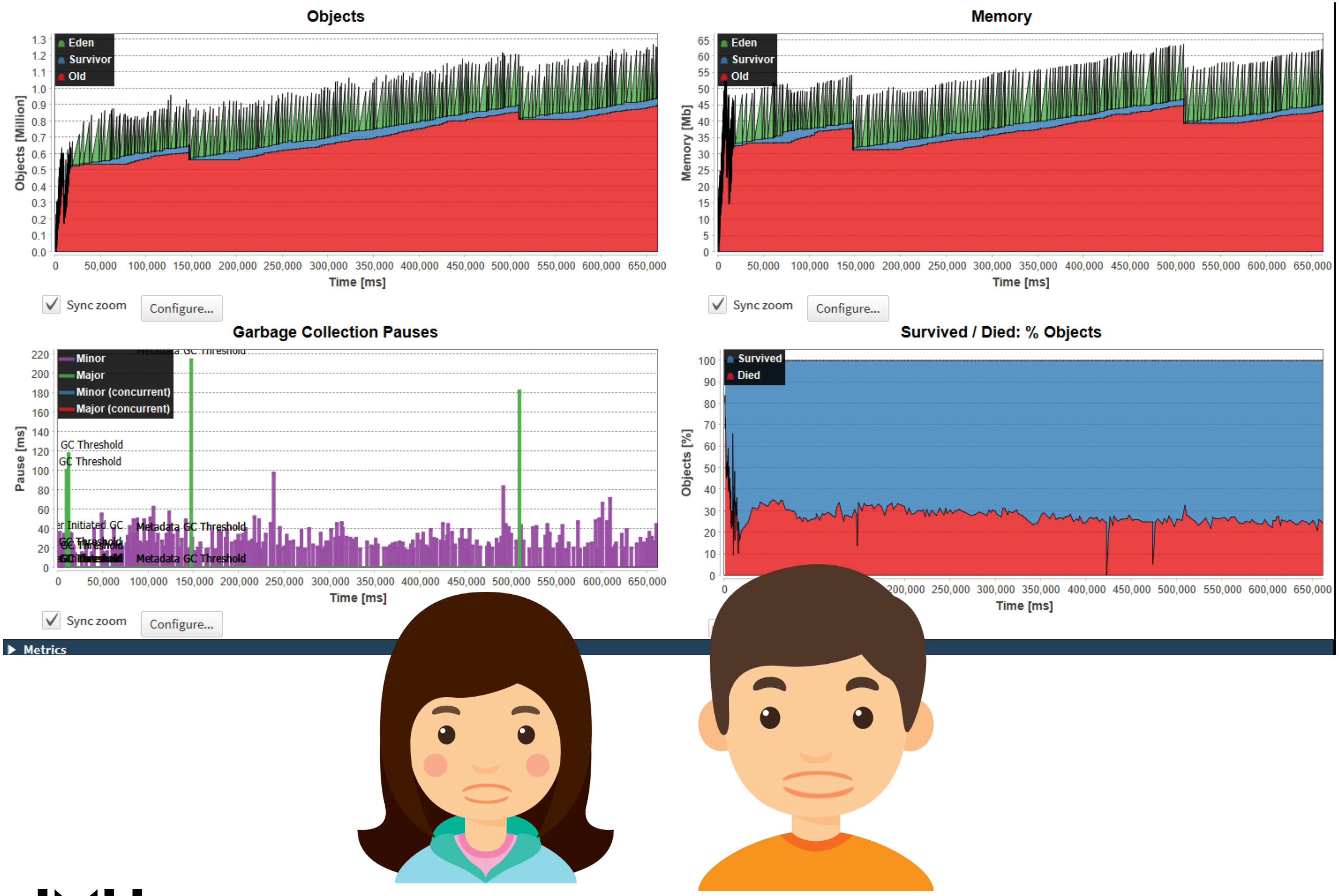
*** not actually Jake**

PROBLEM

PROBLEM

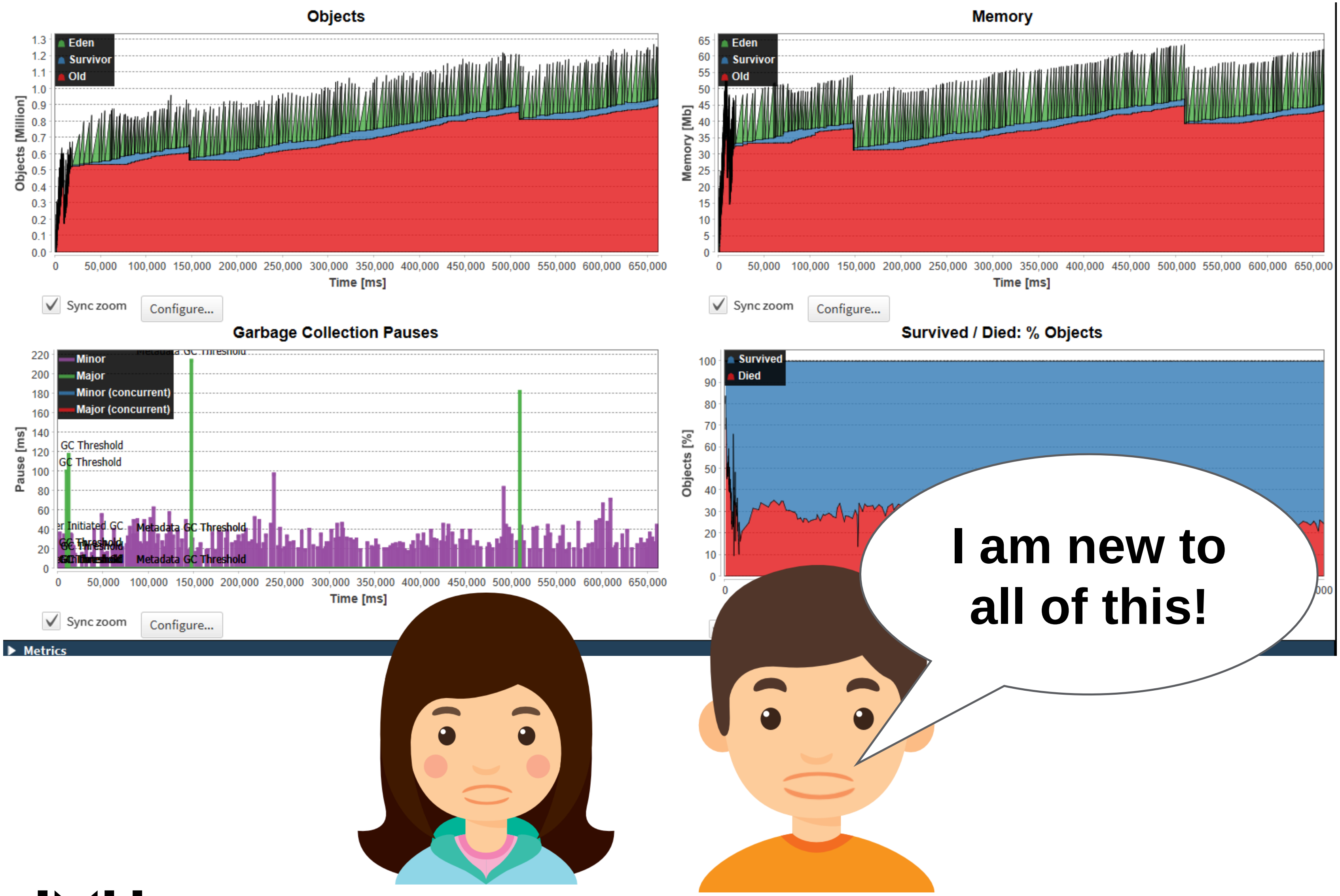

Objects
Eden
Survivor
Old
Objects [Million]
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
50,000
100,000
150,000
200,000
250,000
300,000
350,000
400,000
450,000
500,000
550,000
600,000
650,000
Time [ms]
Sync zoom
Configure...
Memory
Eden
Survivor
Old
Memory [Mb]
65
60
55
50
45
40
35
30
25
20
15
10
5
0
Time [ms]
Sync zoom
Configure...
Garbage Collection Pauses
Minor
Major
Minor (concurrent)
Major (concurrent)
Pause [ms]
220
200
180
160
140
120
100
80
60
40
20
0
GC Threshold
GC Threshold
Metadata GC Threshold
Metadata GC Threshold
Metadata GC Threshold
Time [ms]
Sync zoom
Configure...
Survived / Died: % Objects
Survived
Died
Objects [%]
100
90
80
70
60
50
40
30
20
10
0
Time [ms]
Sync zoom
Metrics

PROBLEM



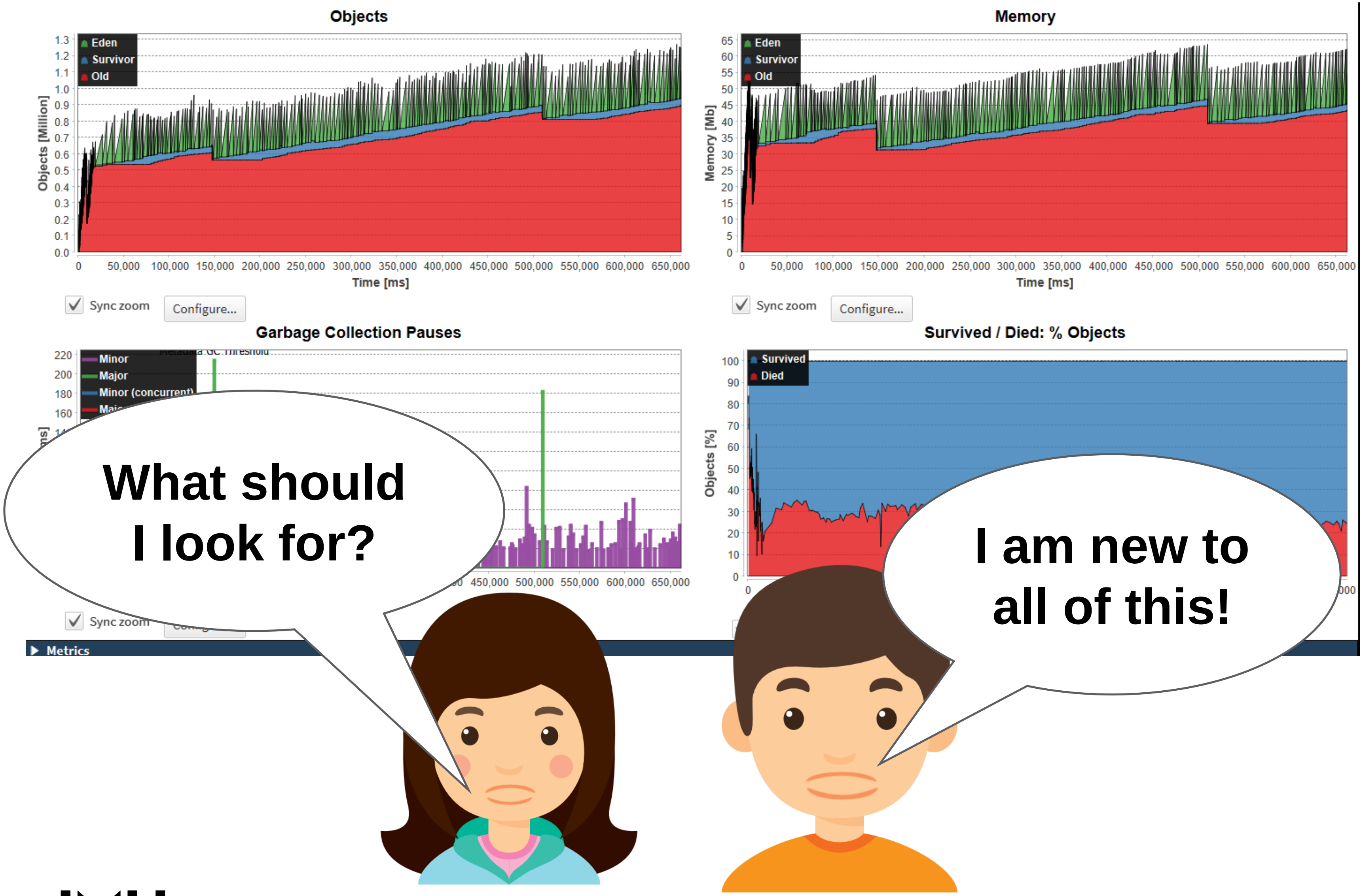
Objects
Eden
Survivor
Old
Objects [Million]
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
Time [ms]
0
50,000
100,000
150,000
200,000
250,000
300,000
350,000
400,000
450,000
500,000
550,000
600,000
650,000
☑ Sync zoom
Configure...
Memory
Memory [Mb]
0
5
10
15
20
25
30
35
40
45
50
55
60
65
Garbage Collection Pauses
Minor
Major
Minor (concurrent)
Major (concurrent)
Pause [ms]
0
20
40
60
80
100
120
140
160
180
200
220
GC Threshold
Metadata GC Threshold
Initiated GC
Survived / Died: % Objects
Survived
Died
Objects [%]
0
10
20
30
40
50
60
70
80
90
100
Metrics

PROBLEM

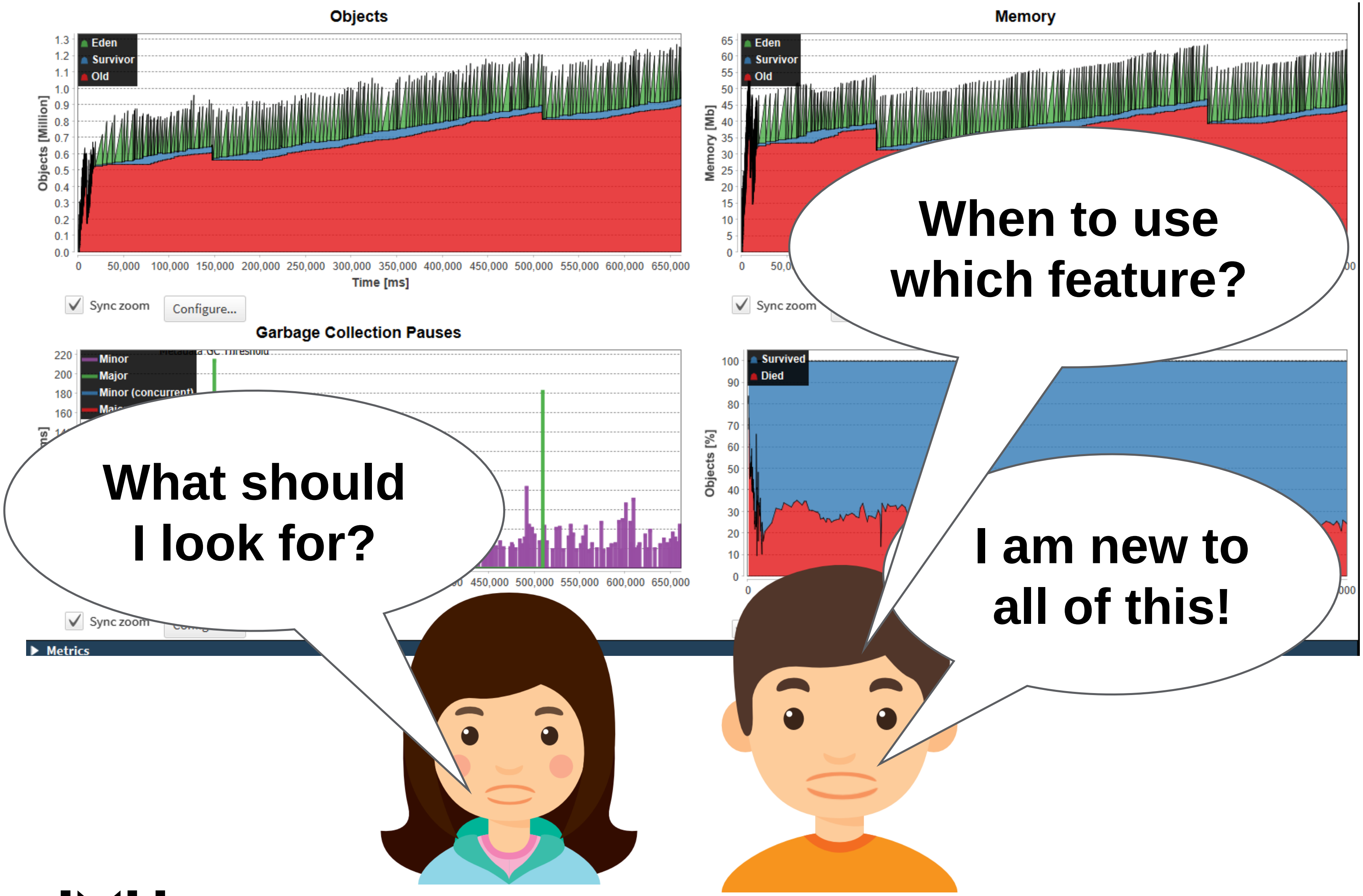


Objects
Eden
Survivor
Old
Objects [Million]
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0 50,000 100,000 150,000 200,000 250,000 300,000 350,000 400,000 450,000 500,000 550,000 600,000 650,000
Time [ms]
☑ Sync zoom
Configure...
Memory
Eden
Survivor
Old
Memory [Mb]
65
60
55
50
45
40
35
30
25
20
15
10
5
0
0 50,000 100,000 150,000 200,000 250,000 300,000 350,000 400,000 450,000 500,000 550,000 600,000 650,000
Time [ms]
☑ Sync zoom
Configure...
Garbage Collection Pauses
Minor
Major
Minor (concurrent)
Major (concurrent)
Pause [ms]
220
200
180
160
140
120
100
80
60
40
20
0
Metadata GC Threshold
GC Threshold
GC Threshold
Initiated GC
Metadata GC Threshold
GC Threshold
Metadata GC Threshold
0 50,000 100,000 150,000 200,000 250,000 300,000 350,000 400,000 450,000 500,000 550,000 600,000 650,000
Time [ms]
☑ Sync zoom
Configure...
Survived / Died: % Objects
Survived
Died
Objects [%]
100
90
80
70
60
50
40
30
20
10
0
0
Metrics
I am new to all of this!

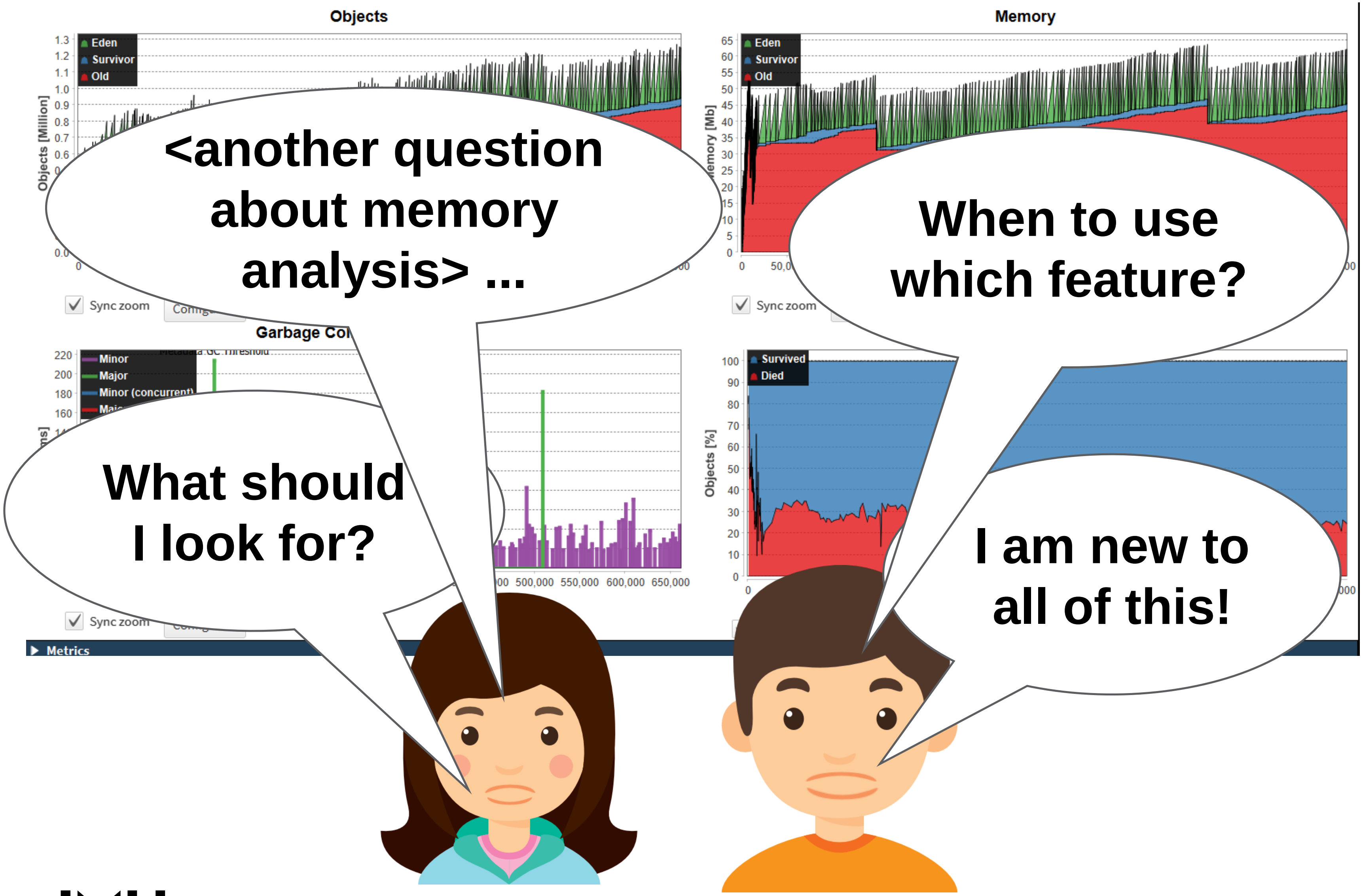
PROBLEM


Objects
Eden
Survivor
Old
Objects [Million]
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
Time [ms]
0
50,000
100,000
150,000
200,000
250,000
300,000
350,000
400,000
450,000
500,000
550,000
600,000
650,000
☑ Sync zoom
Configure...
Memory
Eden
Survivor
Old
Memory [Mb]
0
5
10
15
20
25
30
35
40
45
50
55
60
65
Time [ms]
☑ Sync zoom
Configure...
Garbage Collection Pauses
Minor
Major
Minor (concurrent)
Metadata GC Threshold
220
200
180
160
450,000
500,000
550,000
600,000
650,000
☑ Sync zoom
Survived / Died: % Objects
Survived
Died
Objects [%]
0
10
20
30
40
50
60
70
80
90
100
Metrics
What should I look for?
I am new to all of this!

PROBLEM

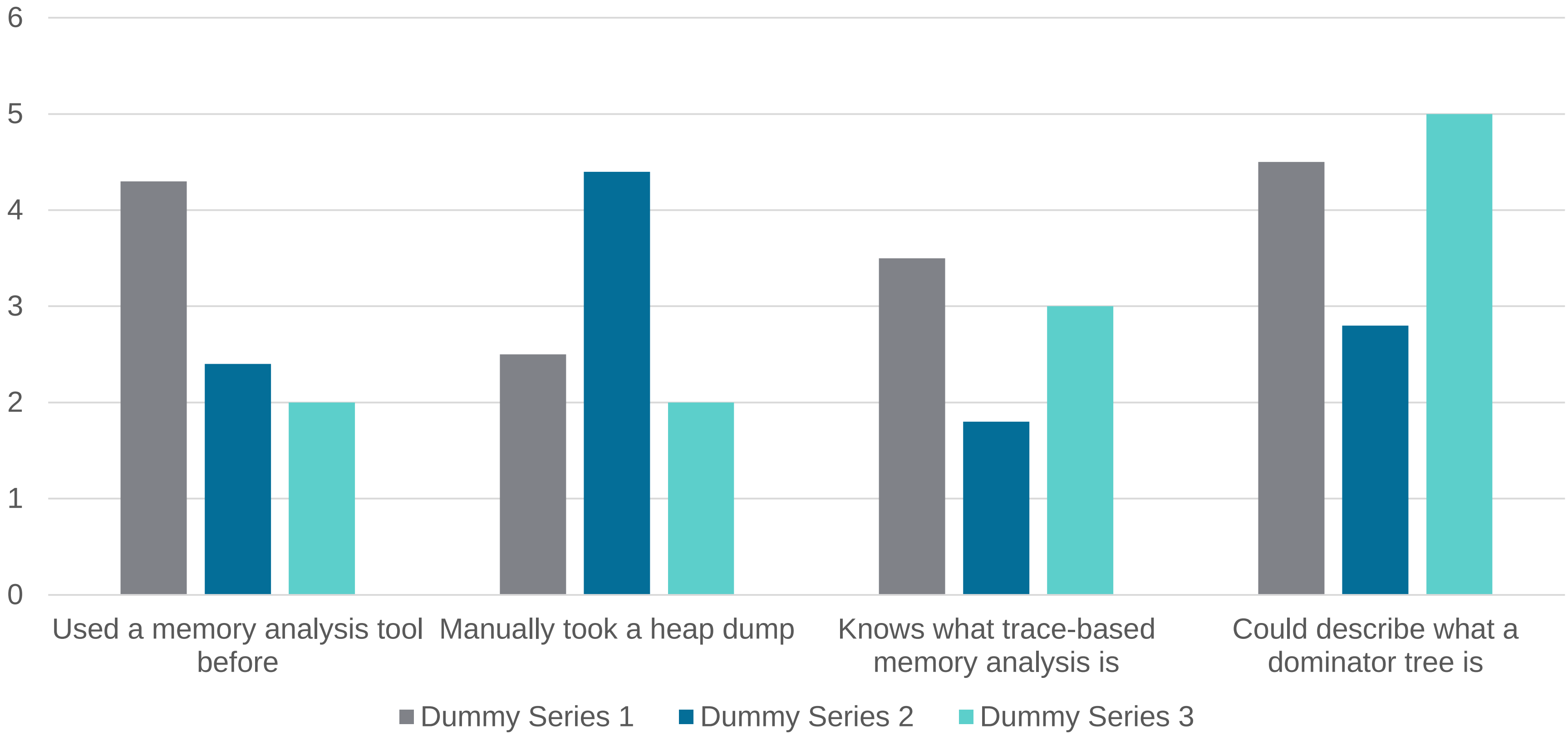

Objects
Eden
Survivor
Old
Objects [Million]
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
50,000
100,000
150,000
200,000
250,000
300,000
350,000
400,000
450,000
500,000
550,000
600,000
650,000
Time [ms]
Sync zoom
Configure...
Memory
Memory [Mb]
65
60
55
50
45
40
35
30
25
20
15
10
5
0
Garbage Collection Pauses
Minor
Major
Minor (concurrent)
Metadata GC Threshold
220
200
180
160
Survived
Died
Objects [%]
100
90
80
70
60
50
40
30
20
10
0
Metrics
When to use which feature?
What should I look for?
I am new to all of this!

PROBLEM


Objects
Eden
Survivor
Old
Objects [Million]
Memory
Memory [Mb]
Garbage Col
Minor
Major
Minor (concurrent)
Metadata GC Threshold
Survived
Died
Objects [%]
Sync zoom
Metrics
<another question about memory analysis> ...
When to use which feature?
What should I look for?
I am new to all of this!

POLL TIME

POLL TIME


6
5
4
3
2
1
0
Used a memory analysis tool before
Manually took a heap dump
Knows what trace-based memory analysis is
Could describe what a dominator tree is
Dummy Series 1
Dummy Series 2
Dummy Series 3

USERS NEED GUIDANCE

MEMORY ANALYSIS 101

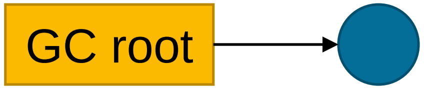
MEMORY ANALYSIS 101

MEMORY ANALYSIS 101

Top-down analysis

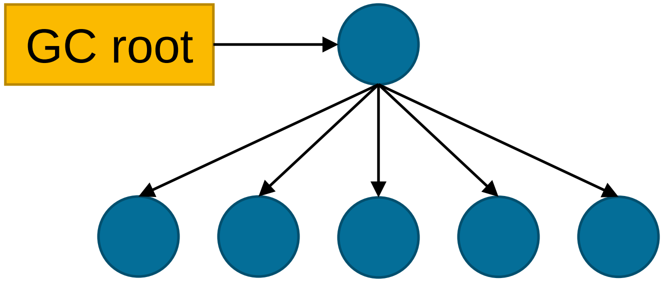
MEMORY ANALYSIS 101

Top-down analysis



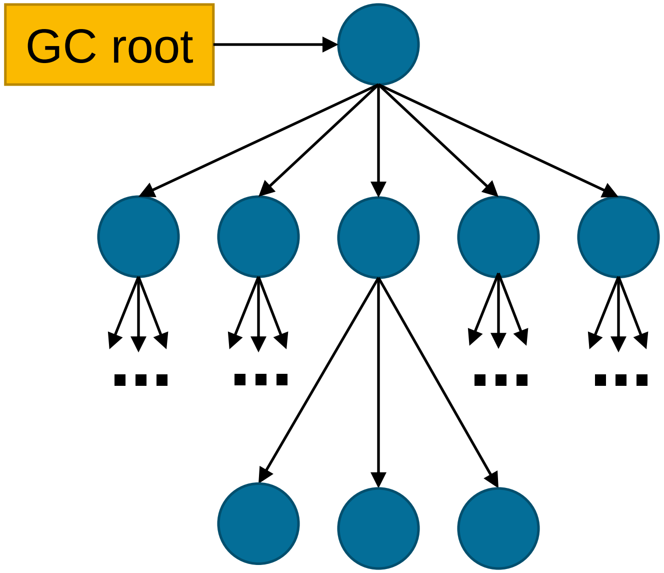
MEMORY ANALYSIS 101

Top-down analysis



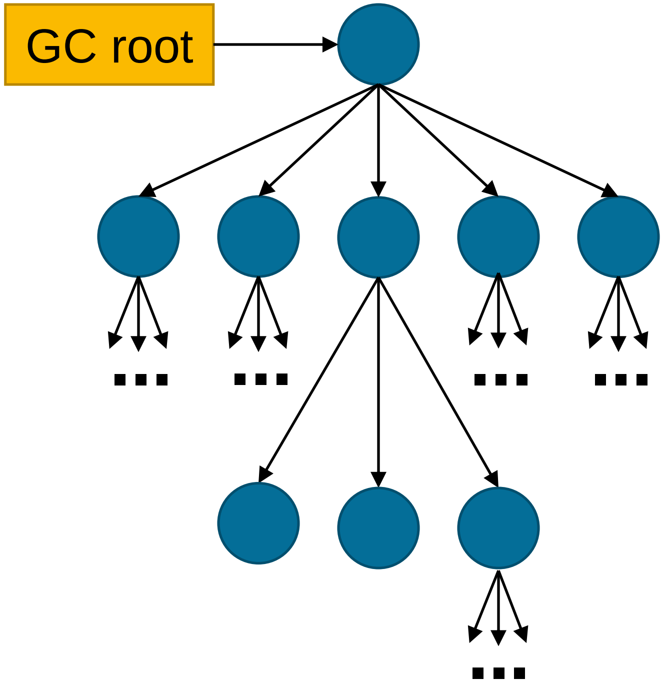
MEMORY ANALYSIS 101

Top-down analysis



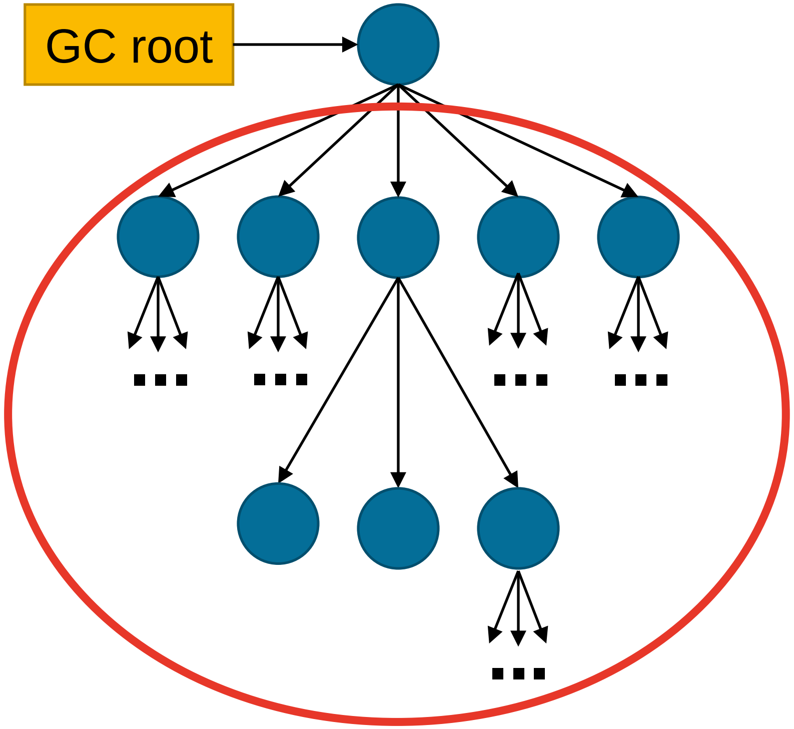
MEMORY ANALYSIS 101

Top-down analysis



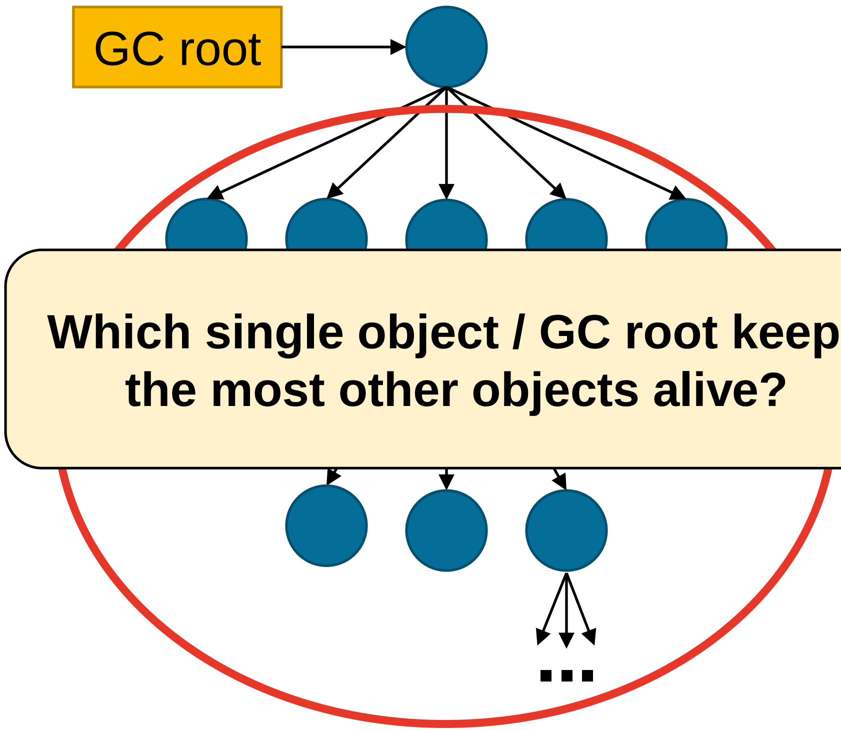
MEMORY ANALYSIS 101

Top-down analysis



MEMORY ANALYSIS 101

Top-down analysis

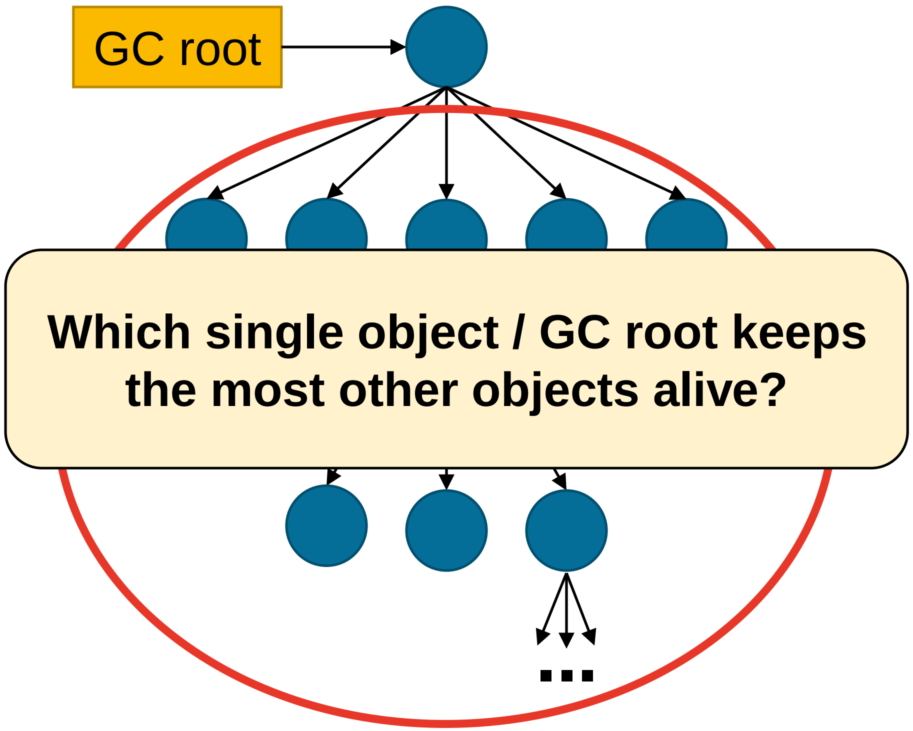


Which single object / GC root keeps the most other objects alive?

MEMORY ANALYSIS 101

Top-down analysis

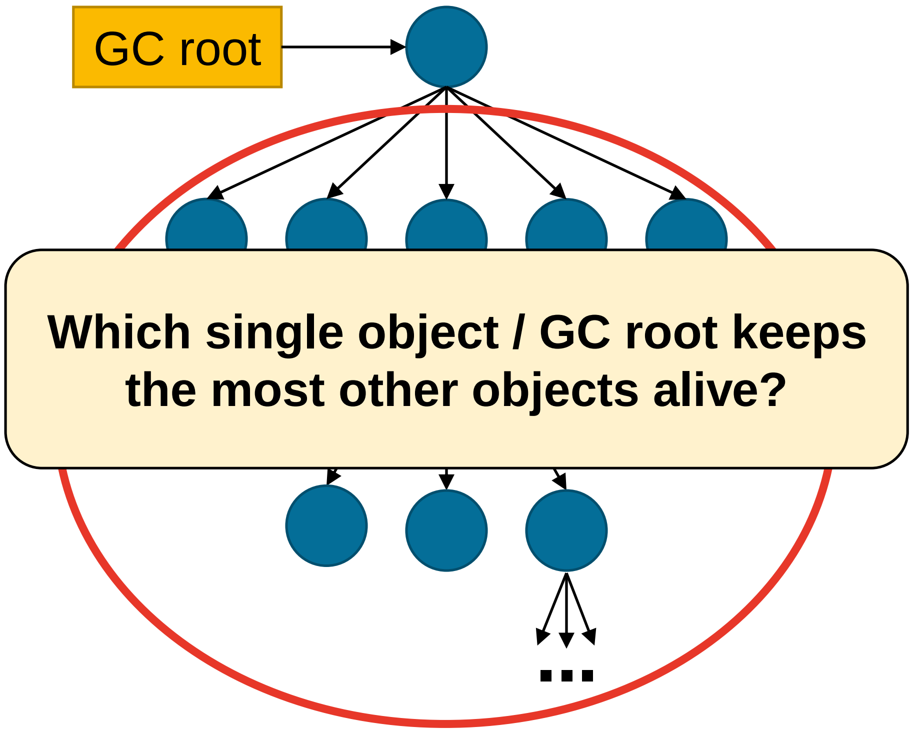
Bottom-up analysis



Which single object / GC root keeps the most other objects alive?

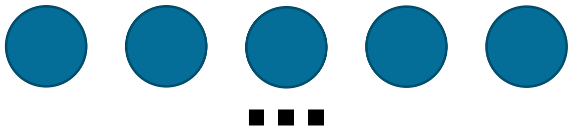
MEMORY ANALYSIS 101

Top-down analysis



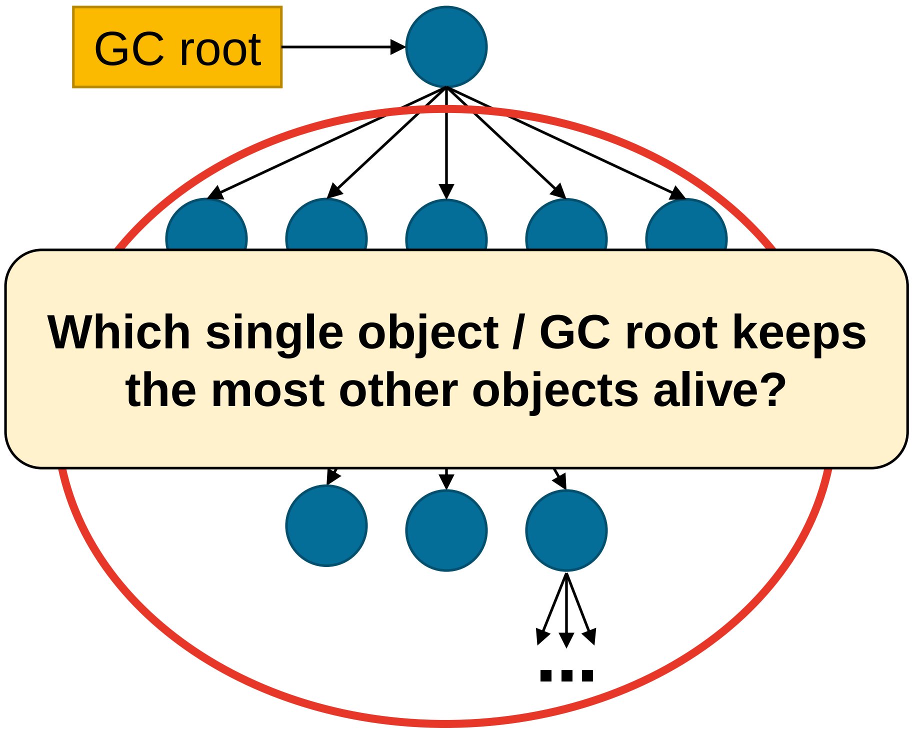
Which single object / GC root keeps the most other objects alive?

Bottom-up analysis



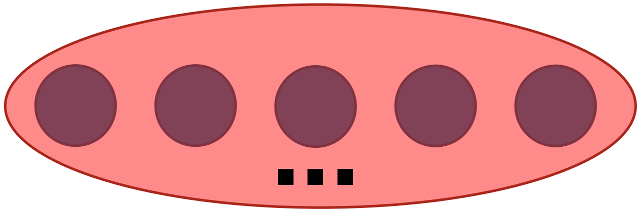
MEMORY ANALYSIS 101

Top-down analysis



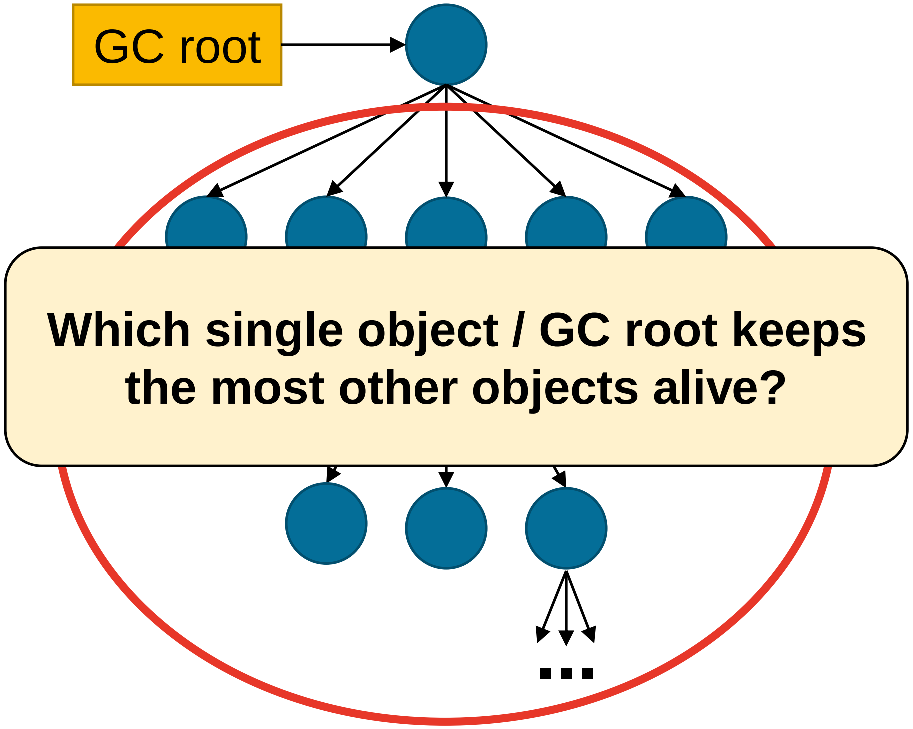
Which single object / GC root keeps the most other objects alive?

Bottom-up analysis



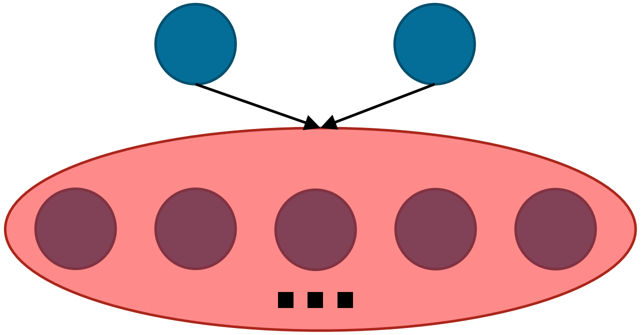
MEMORY ANALYSIS 101

Top-down analysis



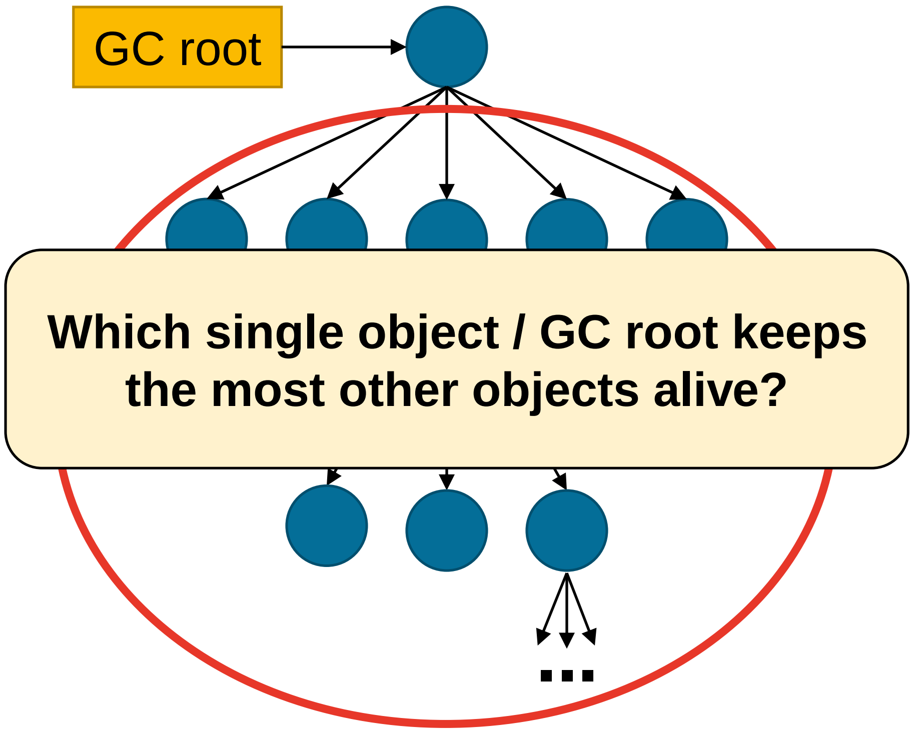
Which single object / GC root keeps the most other objects alive?

Bottom-up analysis



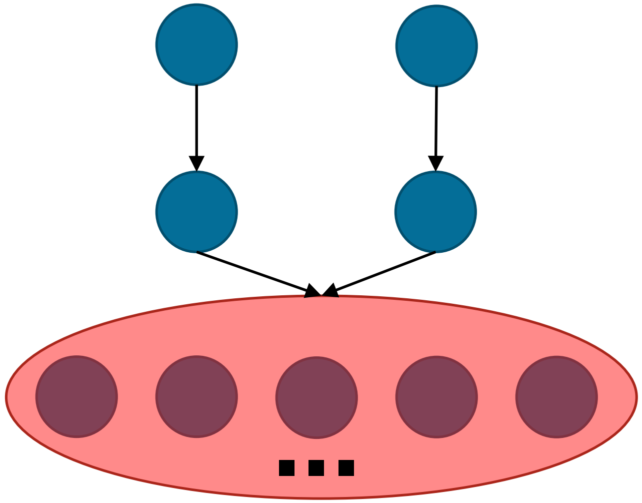
MEMORY ANALYSIS 101

Top-down analysis



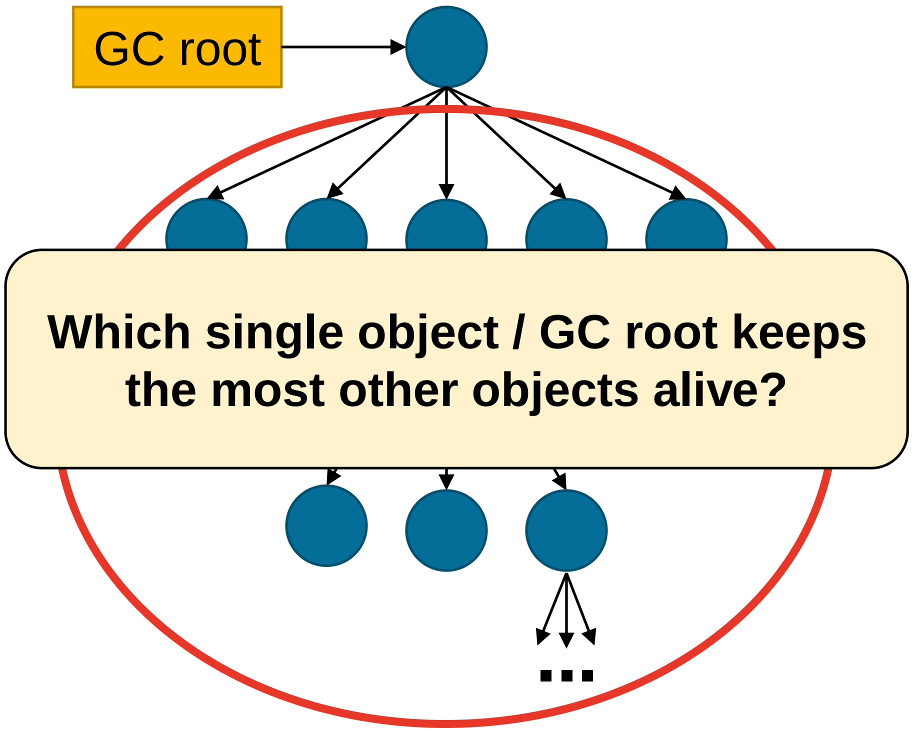
Which single object / GC root keeps the most other objects alive?

Bottom-up analysis



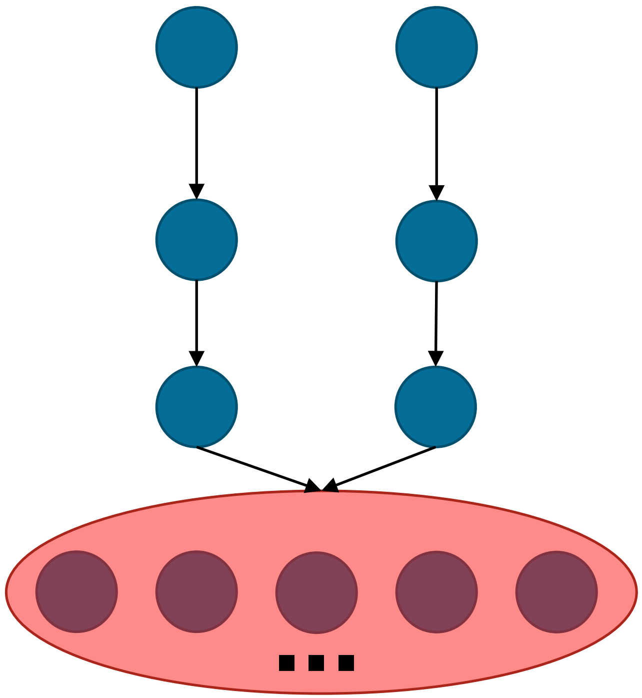
MEMORY ANALYSIS 101

Top-down analysis

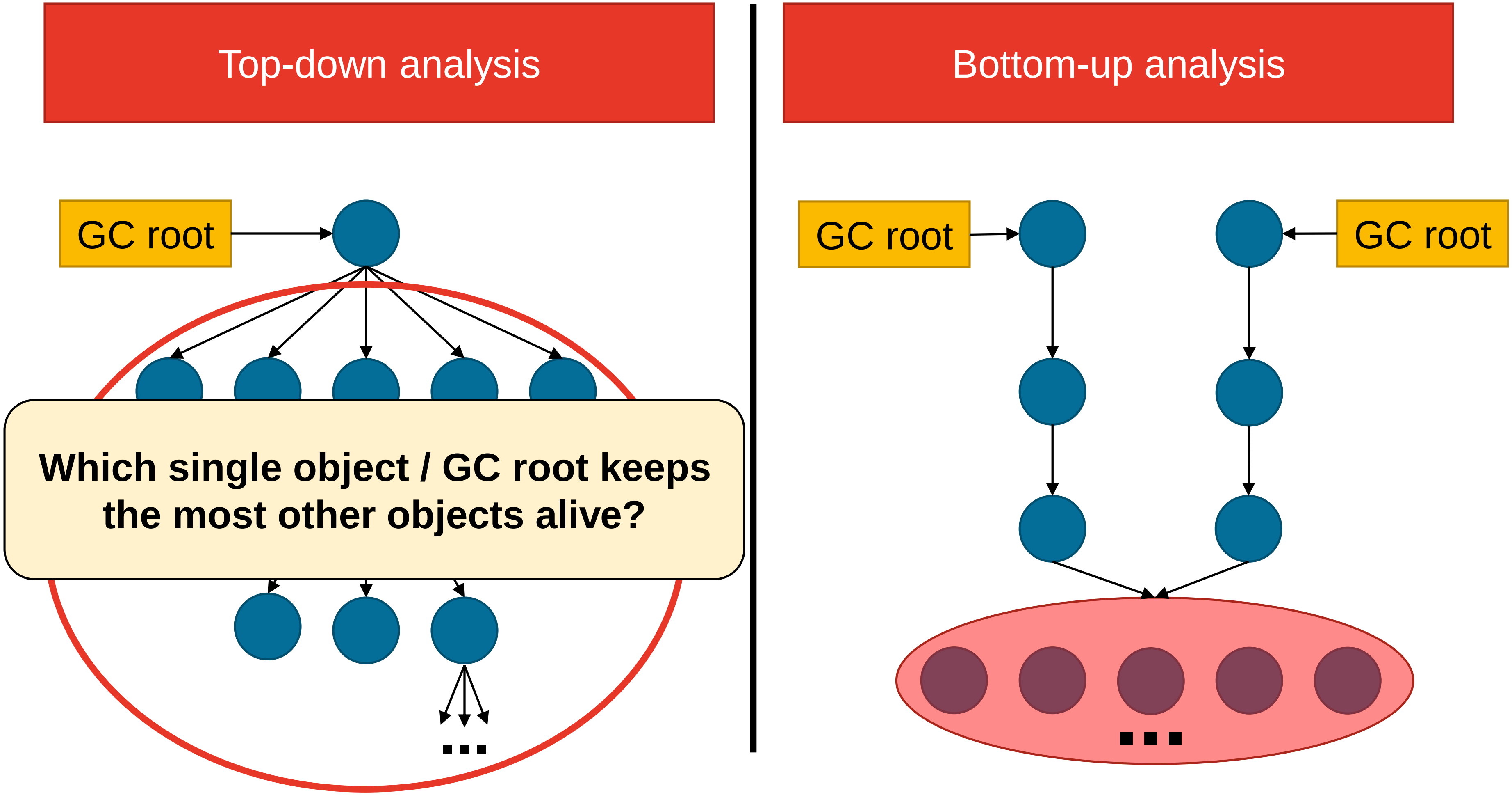


Which single object / GC root keeps the most other objects alive?

Bottom-up analysis



MEMORY ANALYSIS 101



Top-down analysis

Bottom-up analysis

GC root

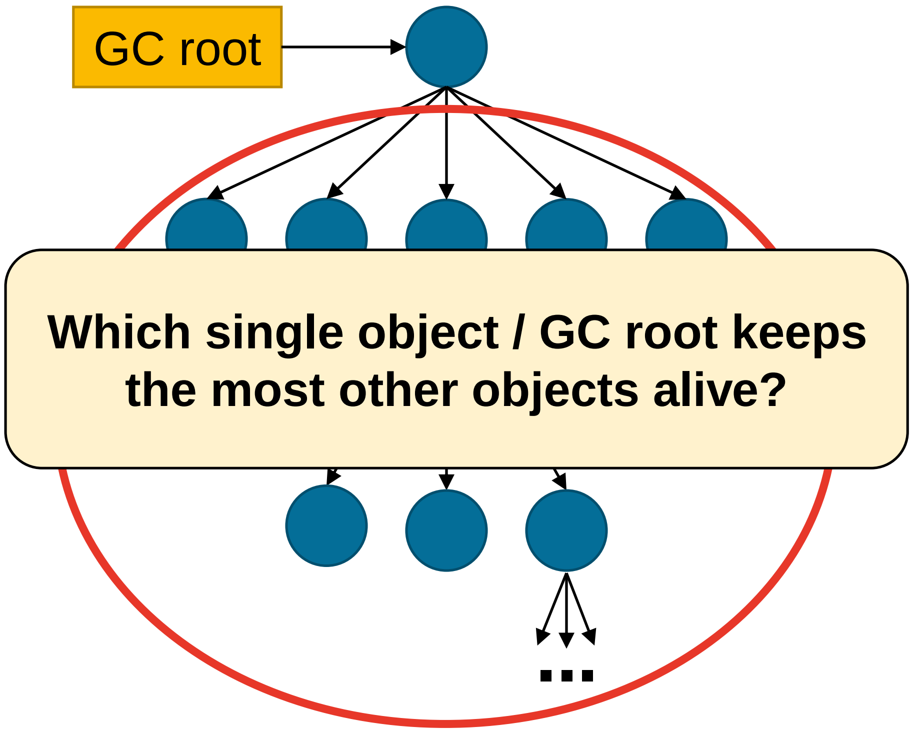
GC root

GC root

Which single object / GC root keeps the most other objects alive?

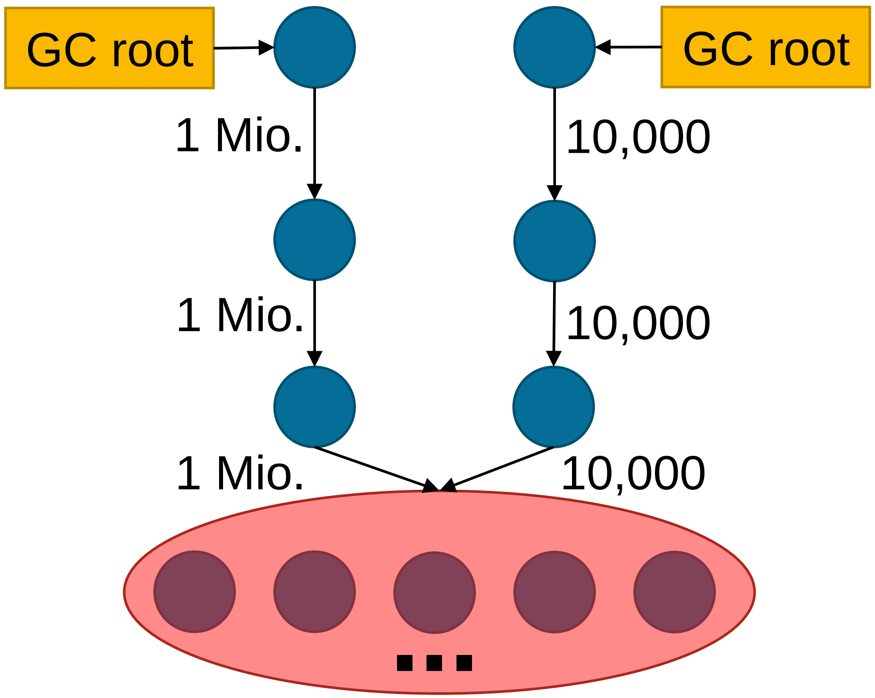
MEMORY ANALYSIS 101

Top-down analysis



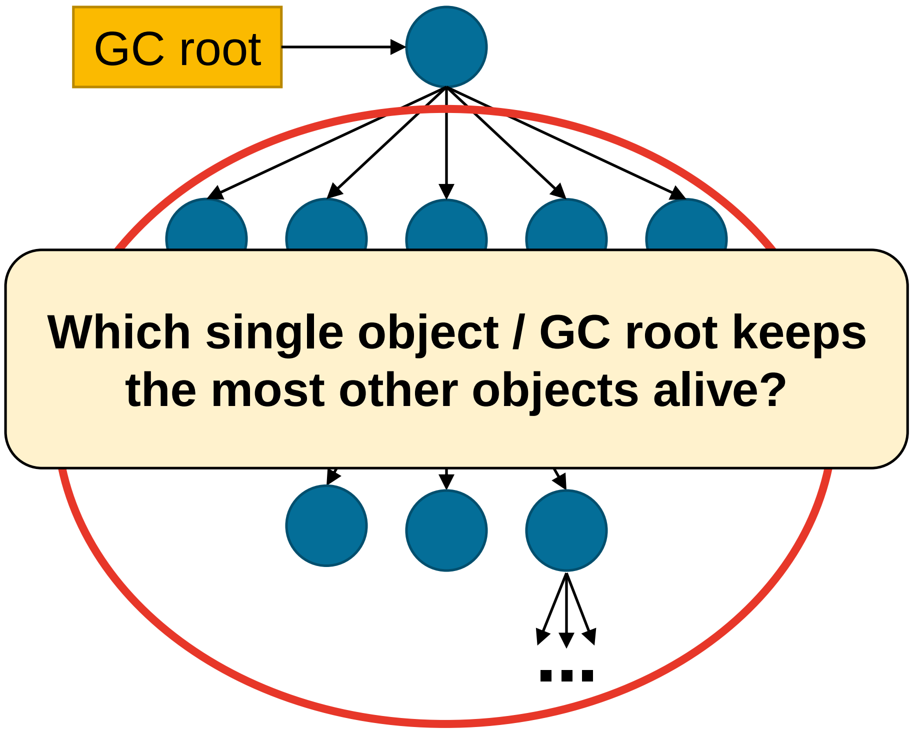
Which single object / GC root keeps the most other objects alive?

Bottom-up analysis



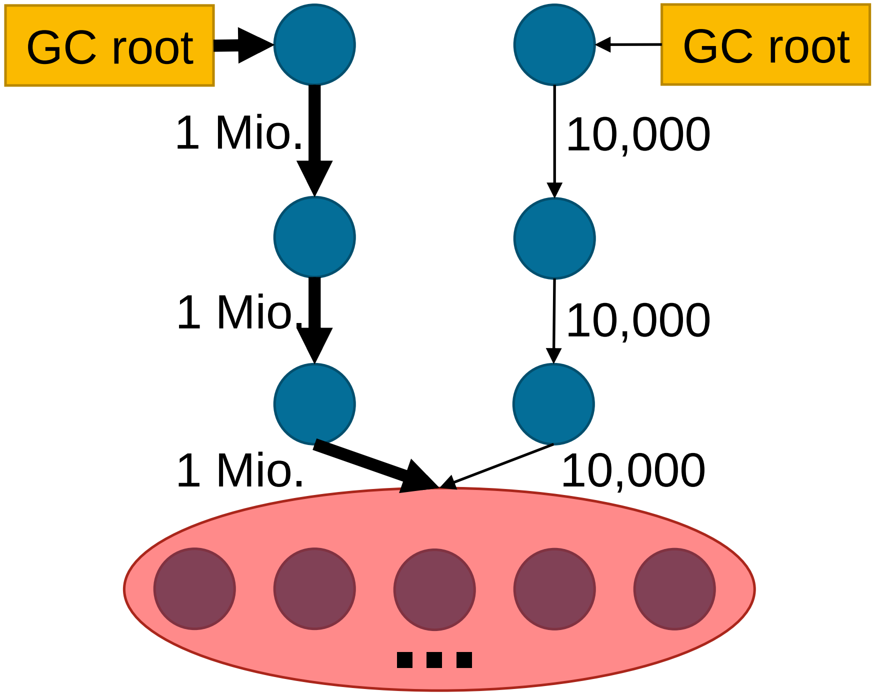
MEMORY ANALYSIS 101

Top-down analysis



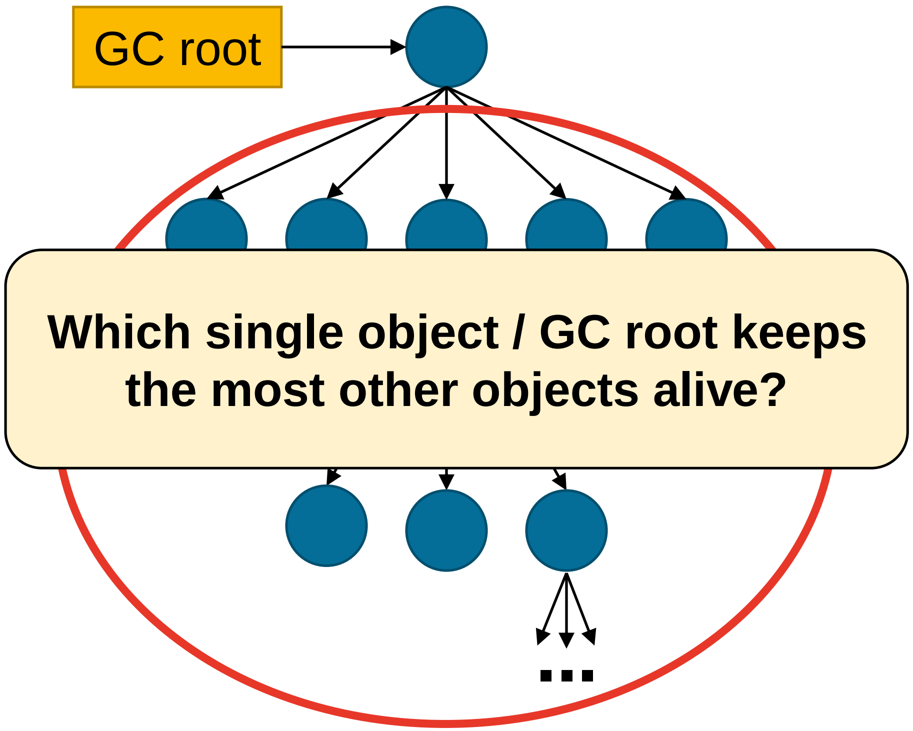
Which single object / GC root keeps the most other objects alive?

Bottom-up analysis



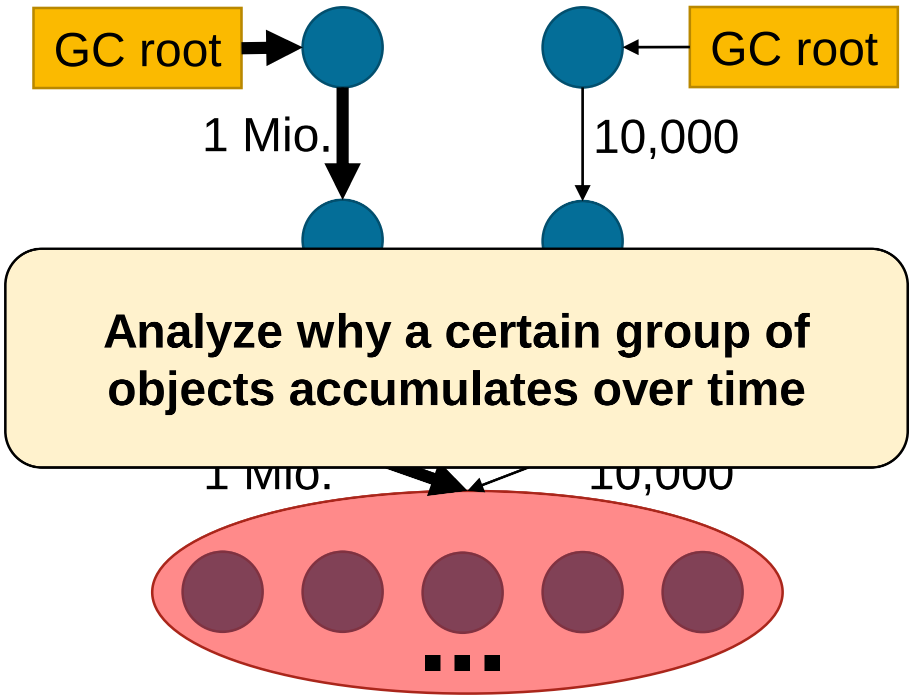
MEMORY ANALYSIS 101

Top-down analysis



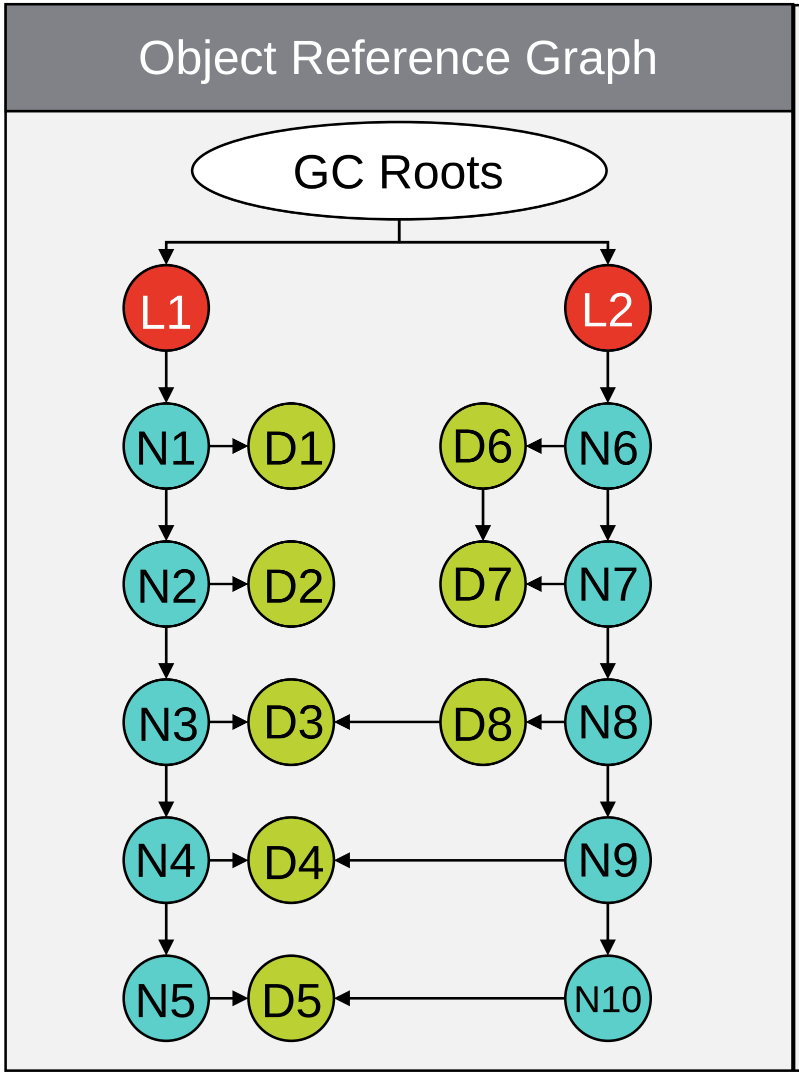
Which single object / GC root keeps the most other objects alive?

Bottom-up analysis

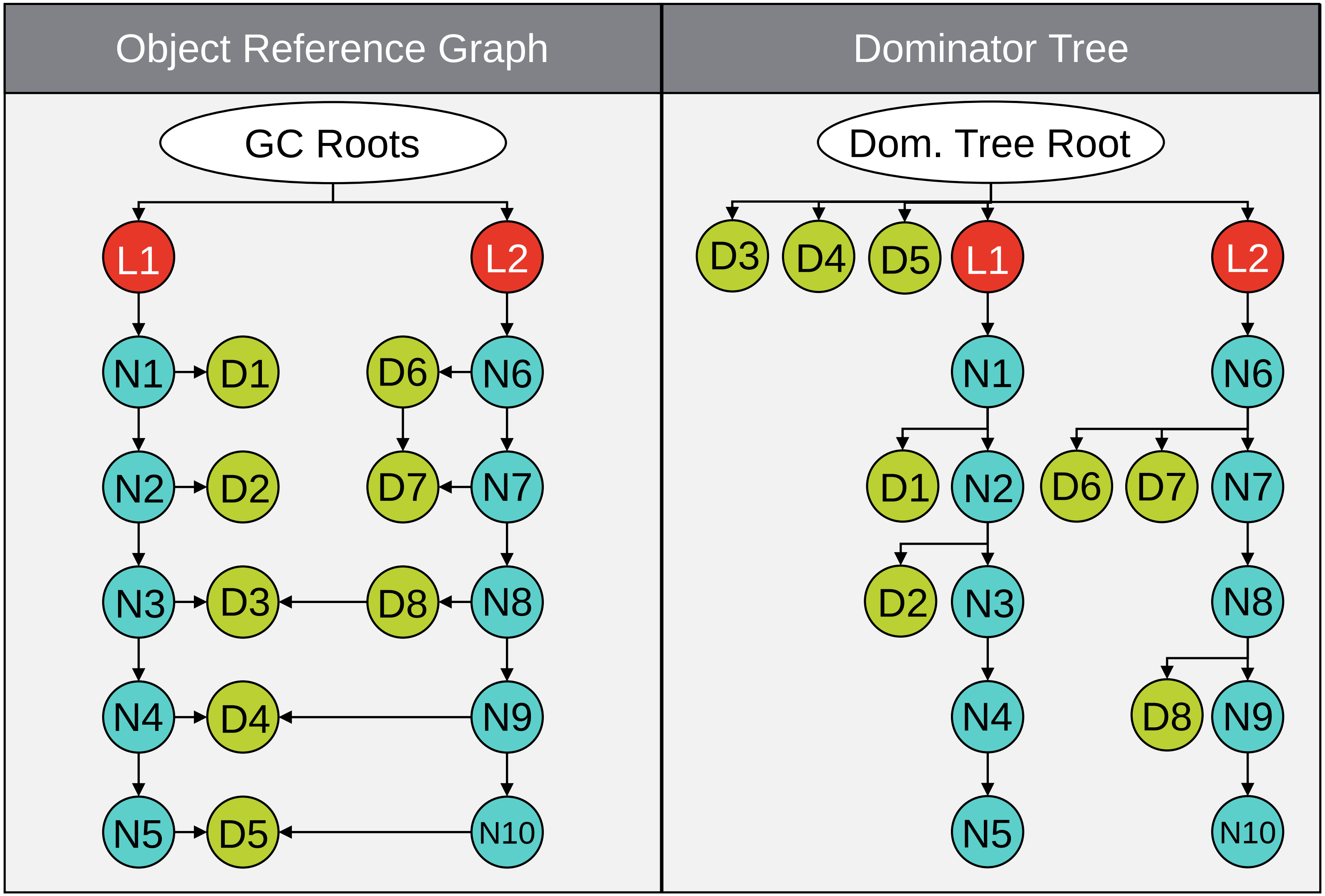


Analyze why a certain group of objects accumulates over time

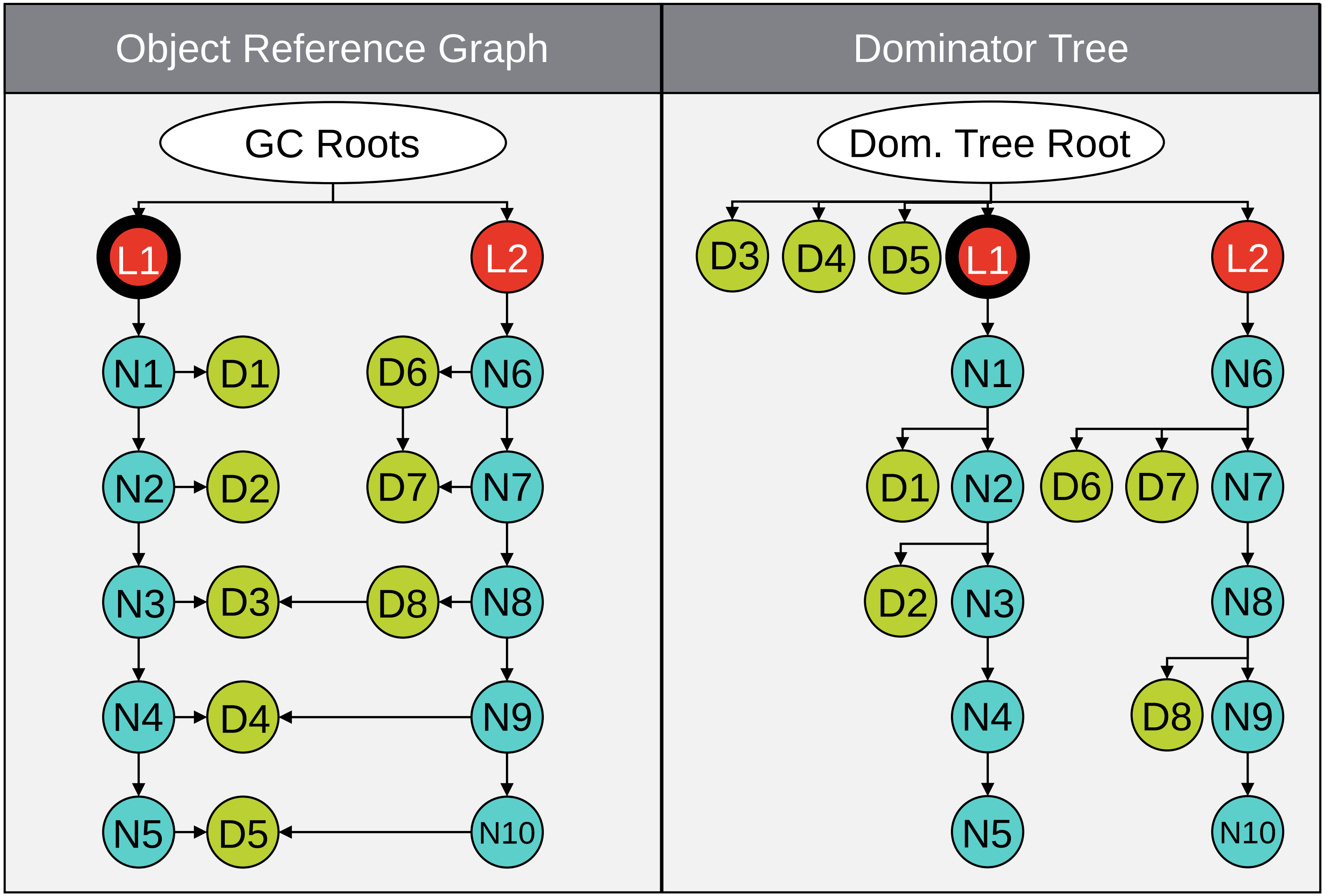
DOMINATOR TREE



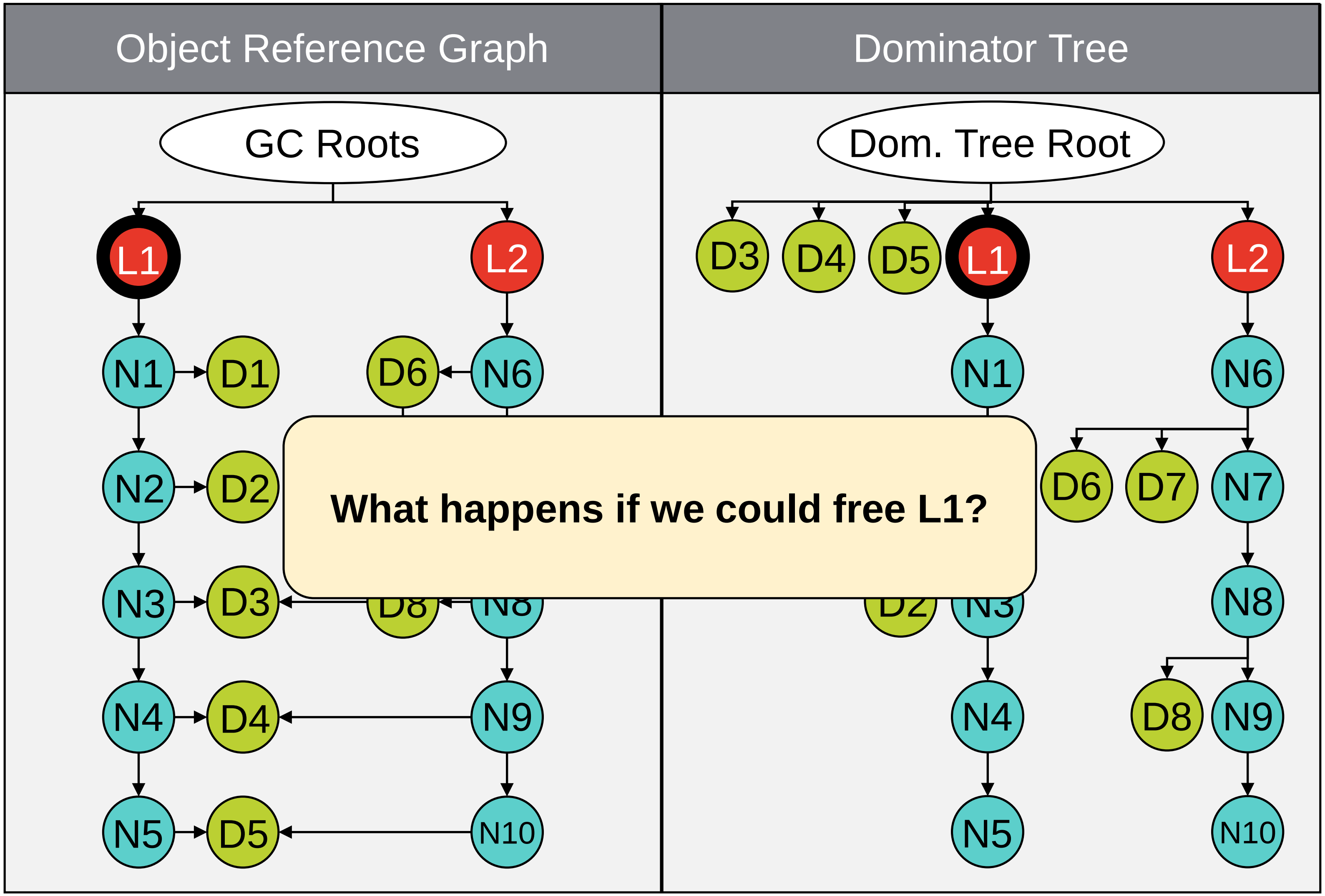
DOMINATOR TREE



DOMINATOR TREE

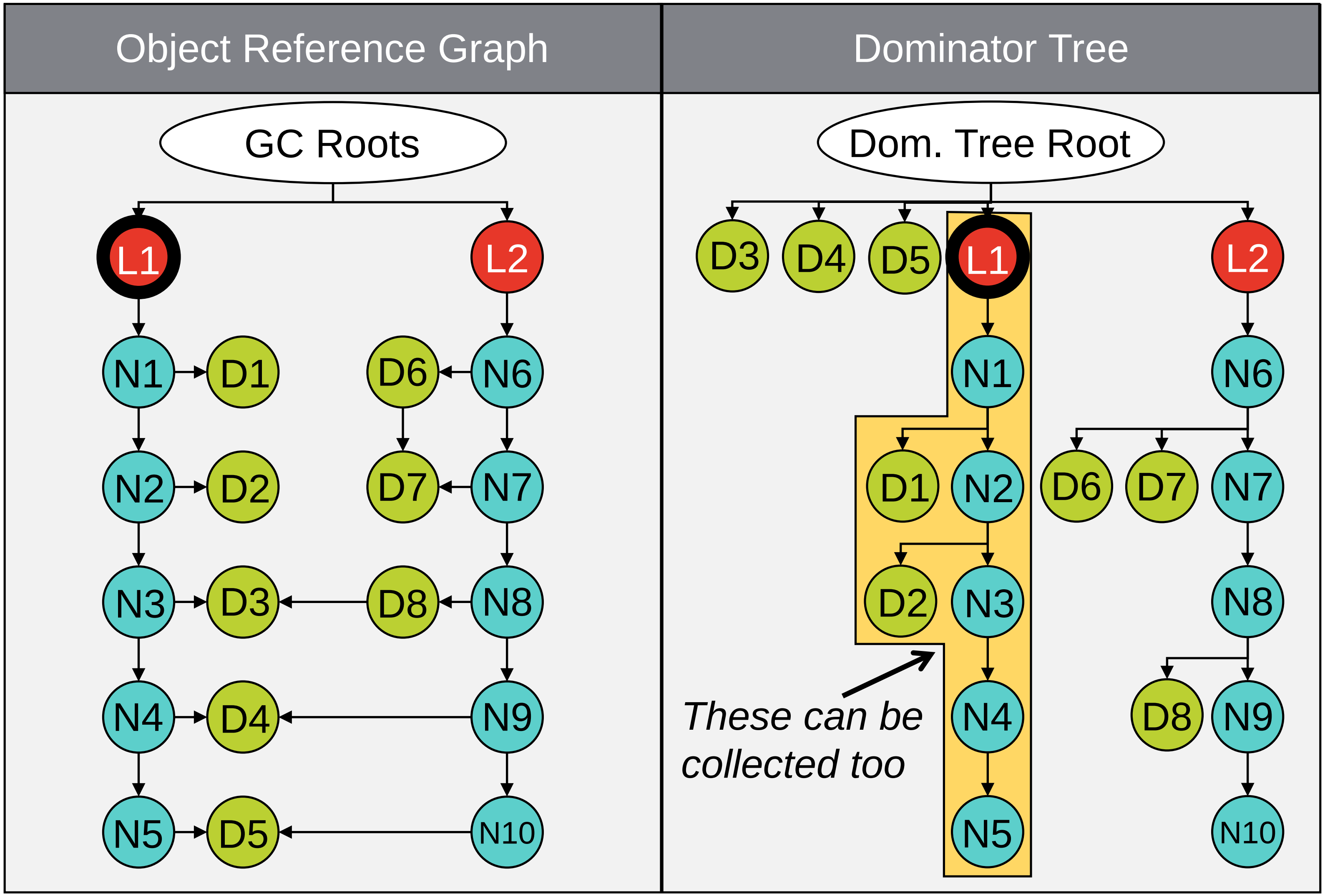


DOMINATOR TREE



What happens if we could free L1?

DOMINATOR TREE



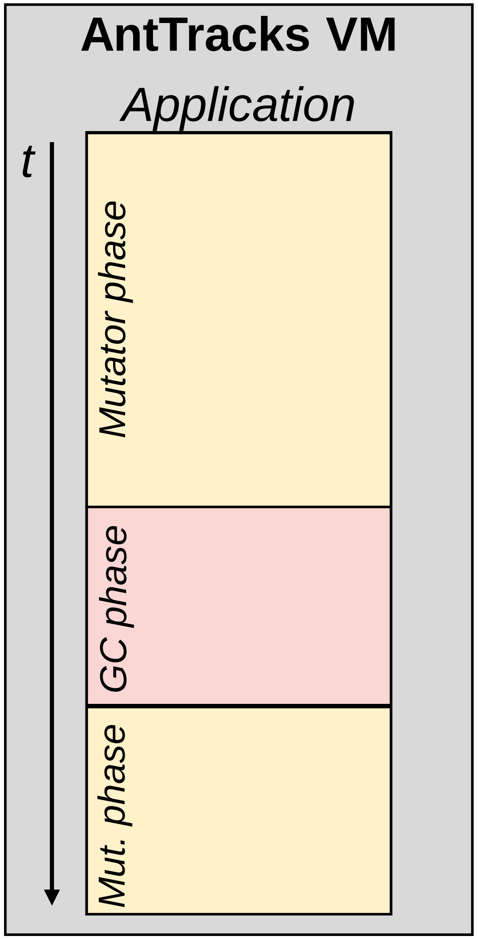
Thanks Markus, but that seems like a lot to learn...

Thanks Markus, but that seems like a lot to learn...

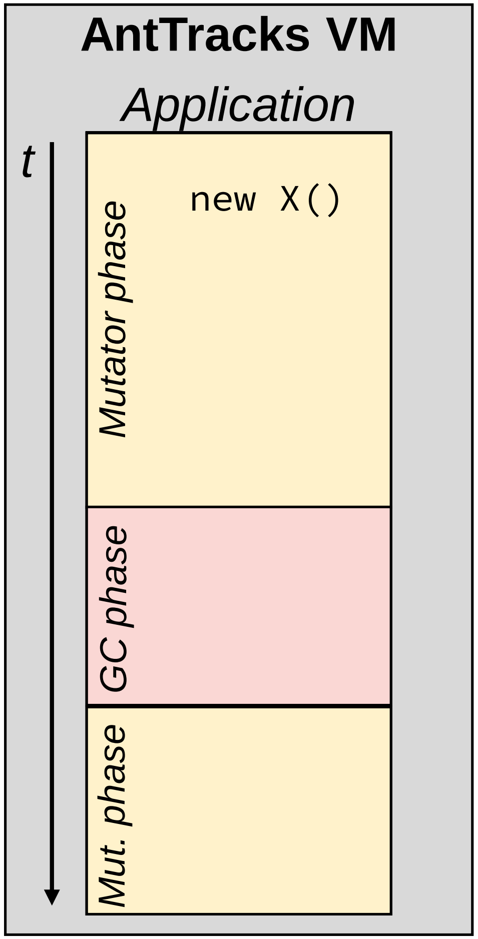
Thanks Markus, but that seems like a lot to learn...

Okay, let us try to improve that in AntTracks

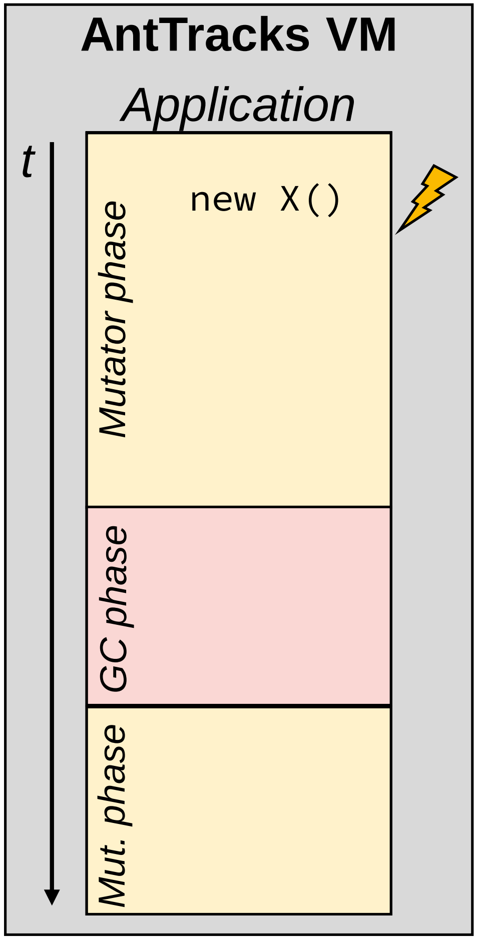
ANTTRACKS SYSTEM OVERVIEW



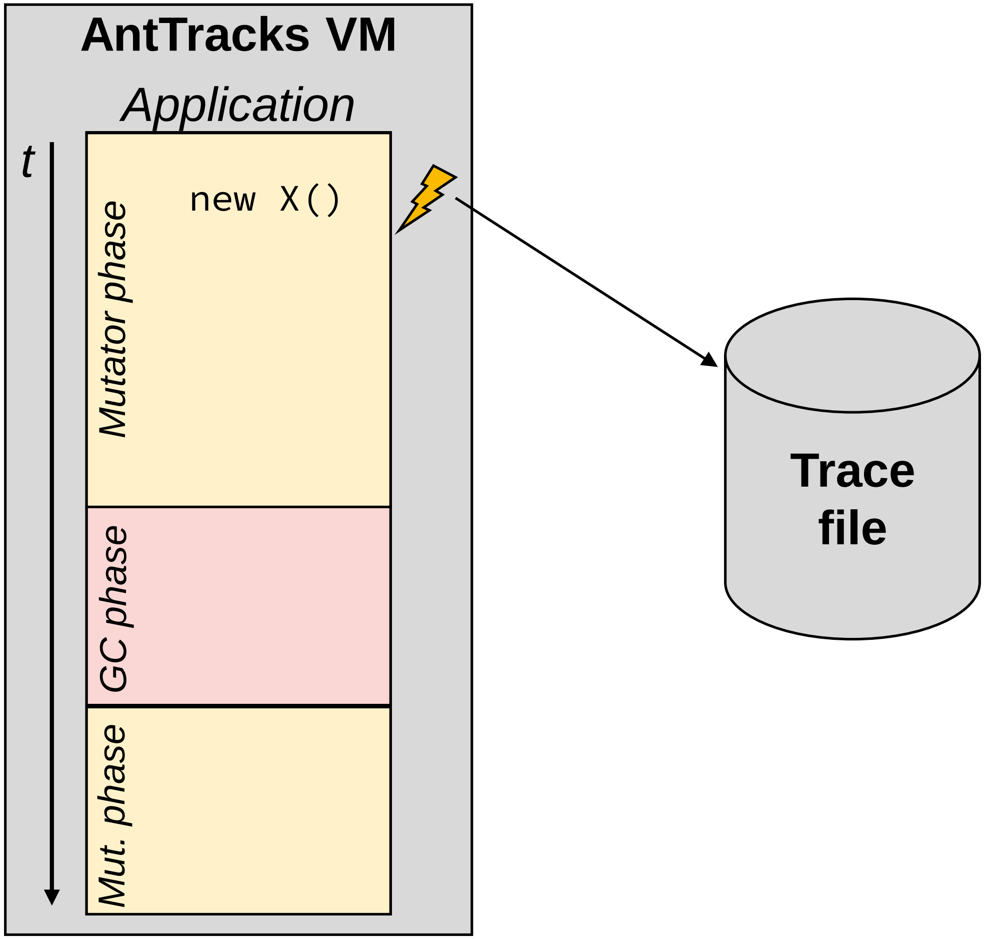
ANTTRACKS SYSTEM OVERVIEW



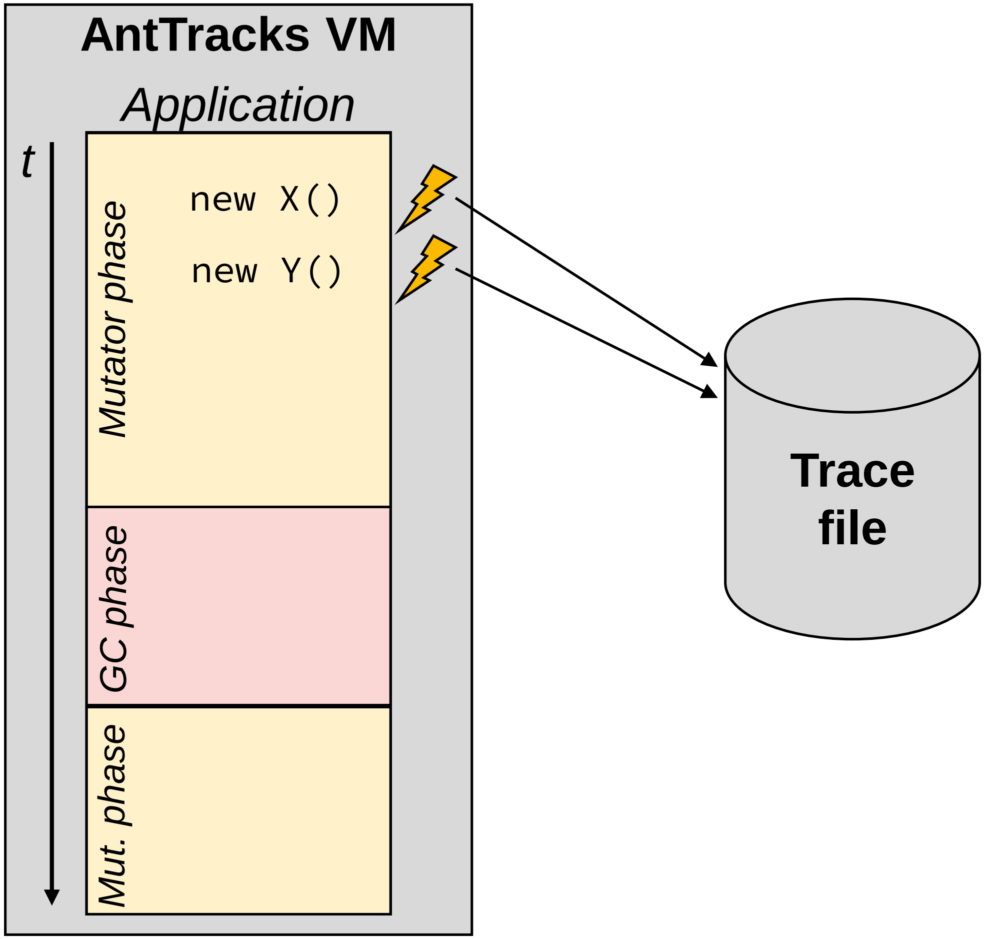
ANTTRACKS SYSTEM OVERVIEW



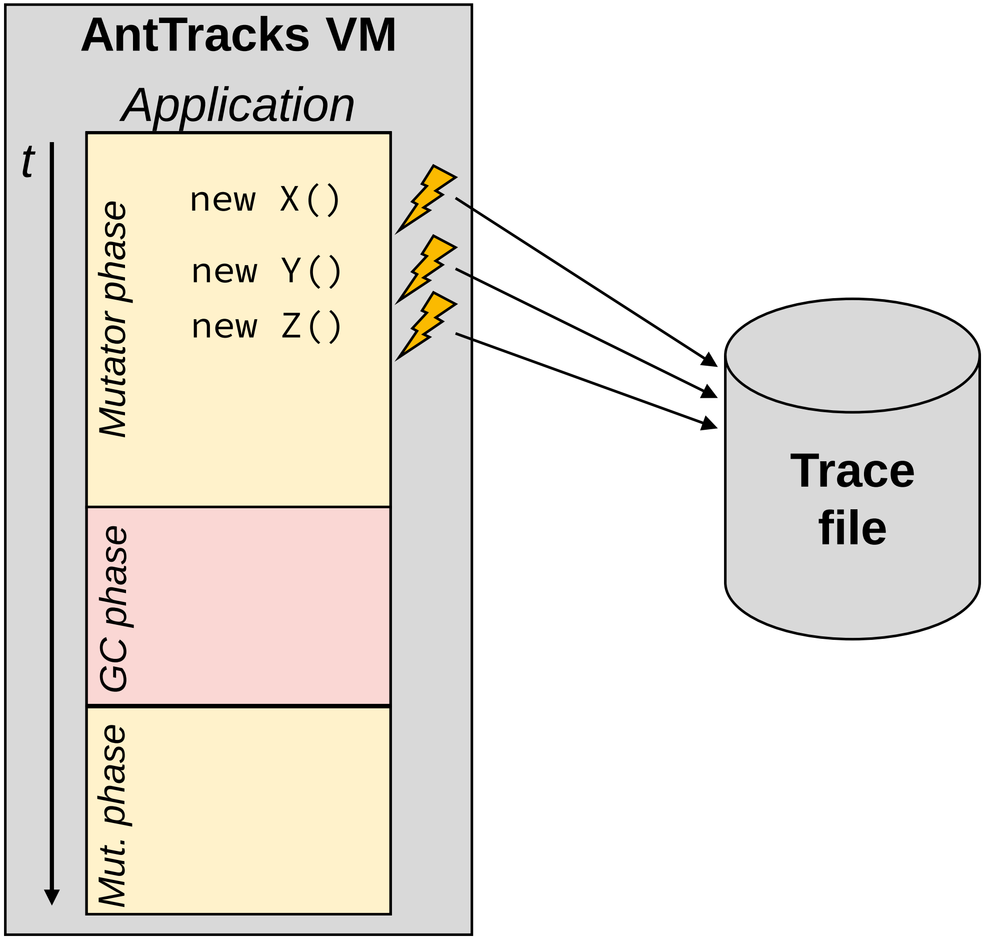
ANTTRACKS SYSTEM OVERVIEW



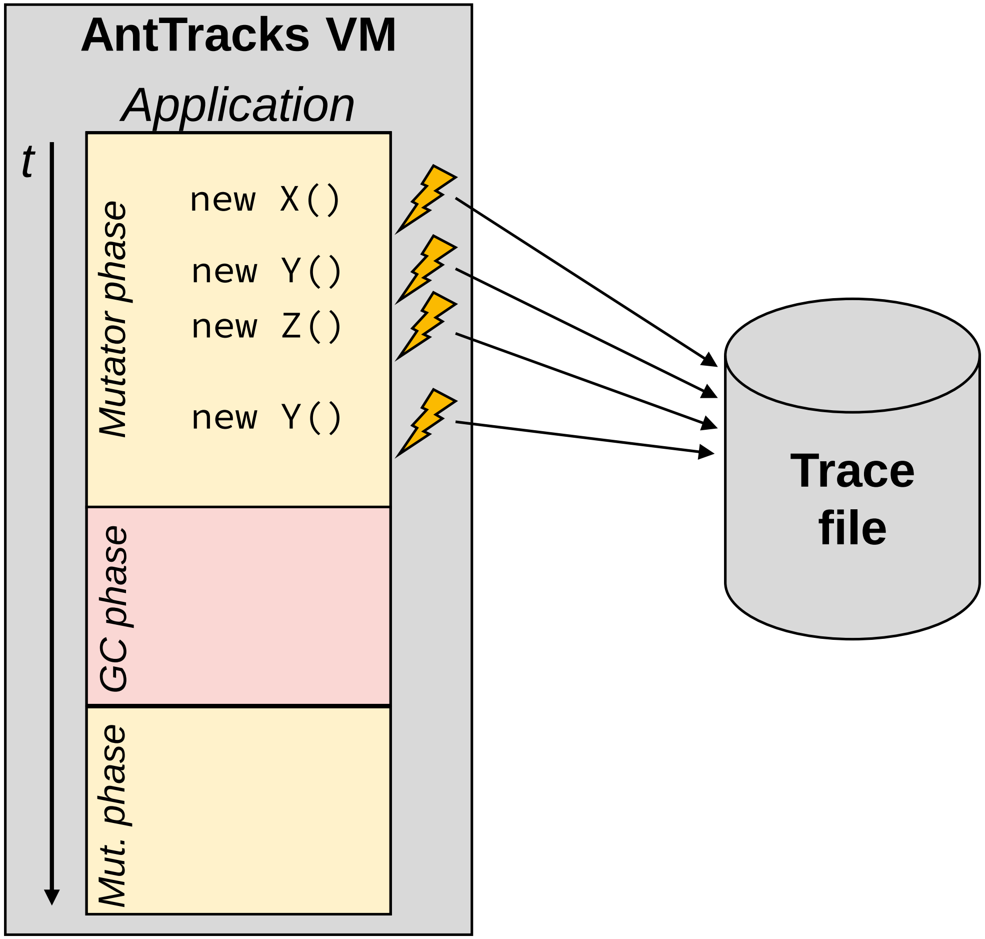
ANTTRACKS SYSTEM OVERVIEW



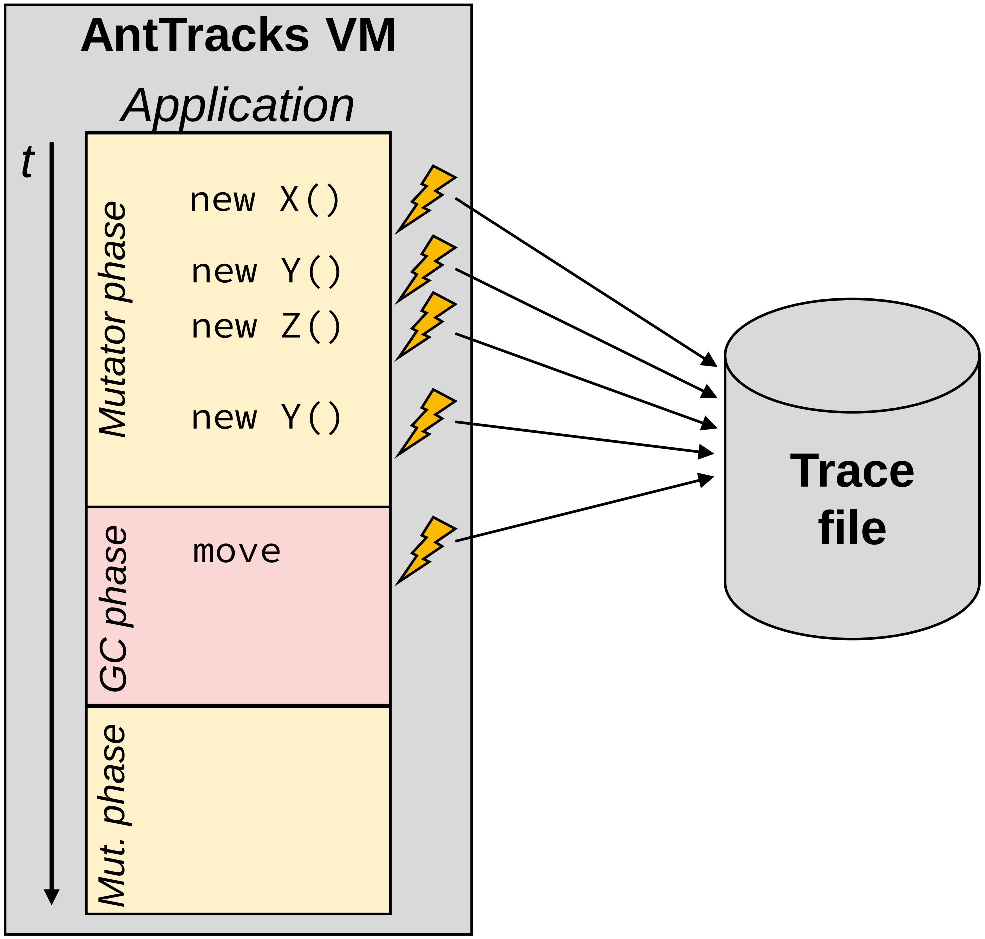
ANTTRACKS SYSTEM OVERVIEW



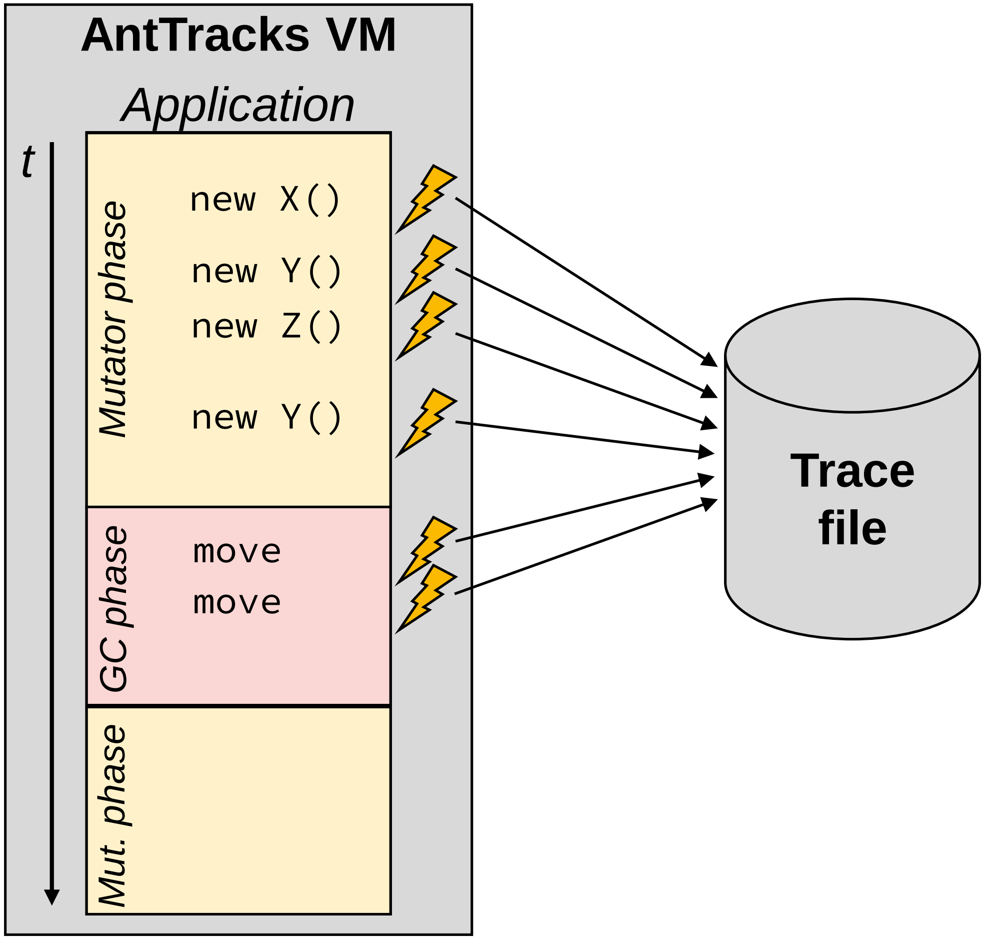
ANTTRACKS SYSTEM OVERVIEW



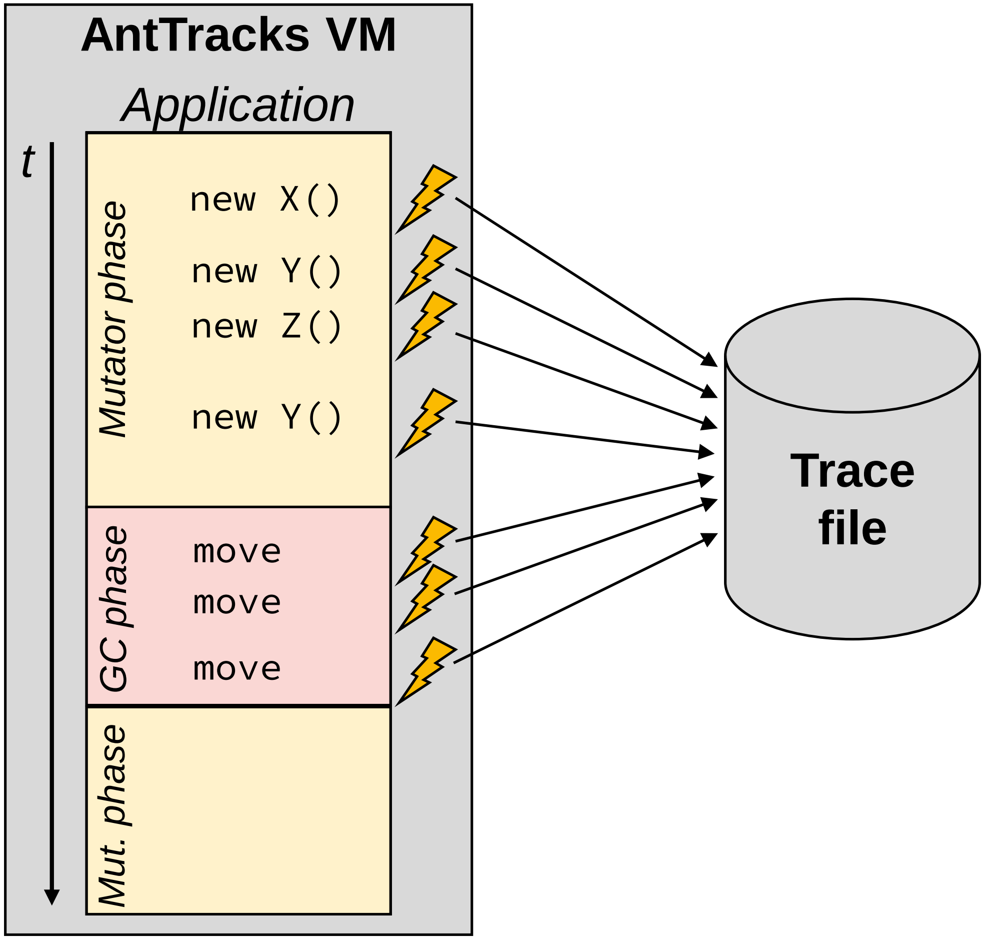
ANTTRACKS SYSTEM OVERVIEW



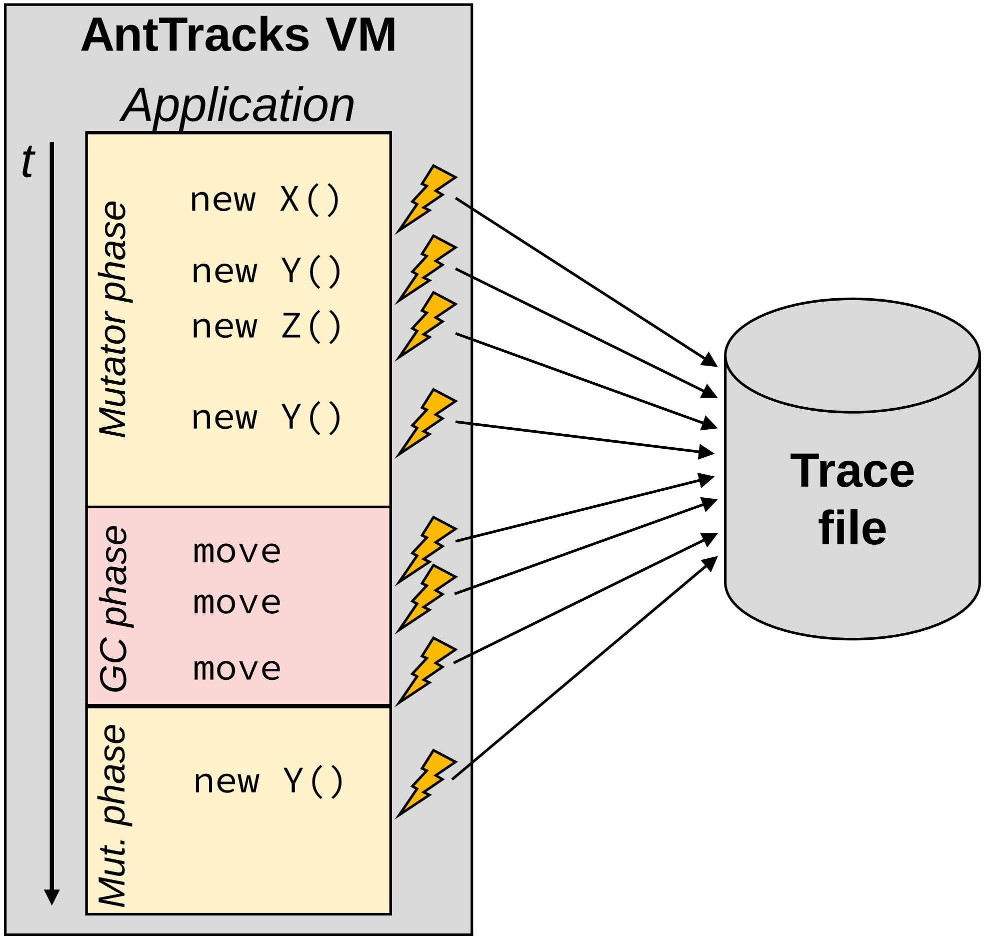
ANTTRACKS SYSTEM OVERVIEW



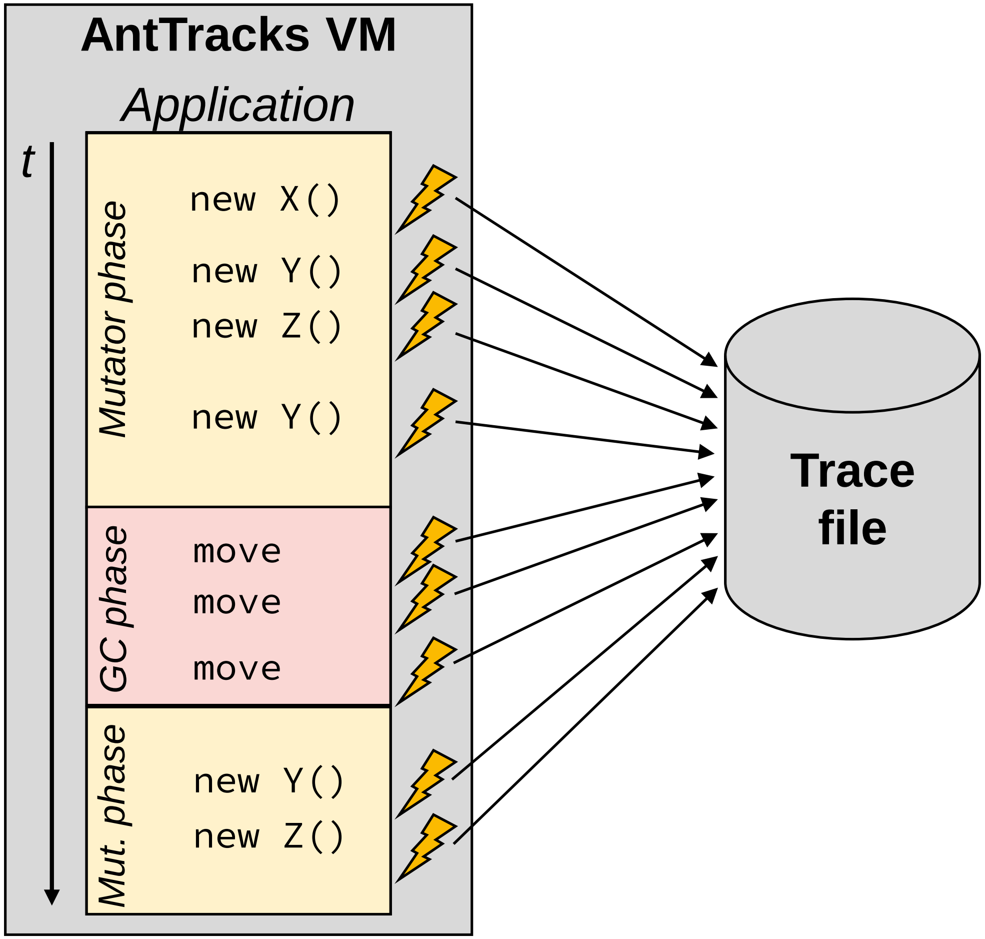
ANTTRACKS SYSTEM OVERVIEW



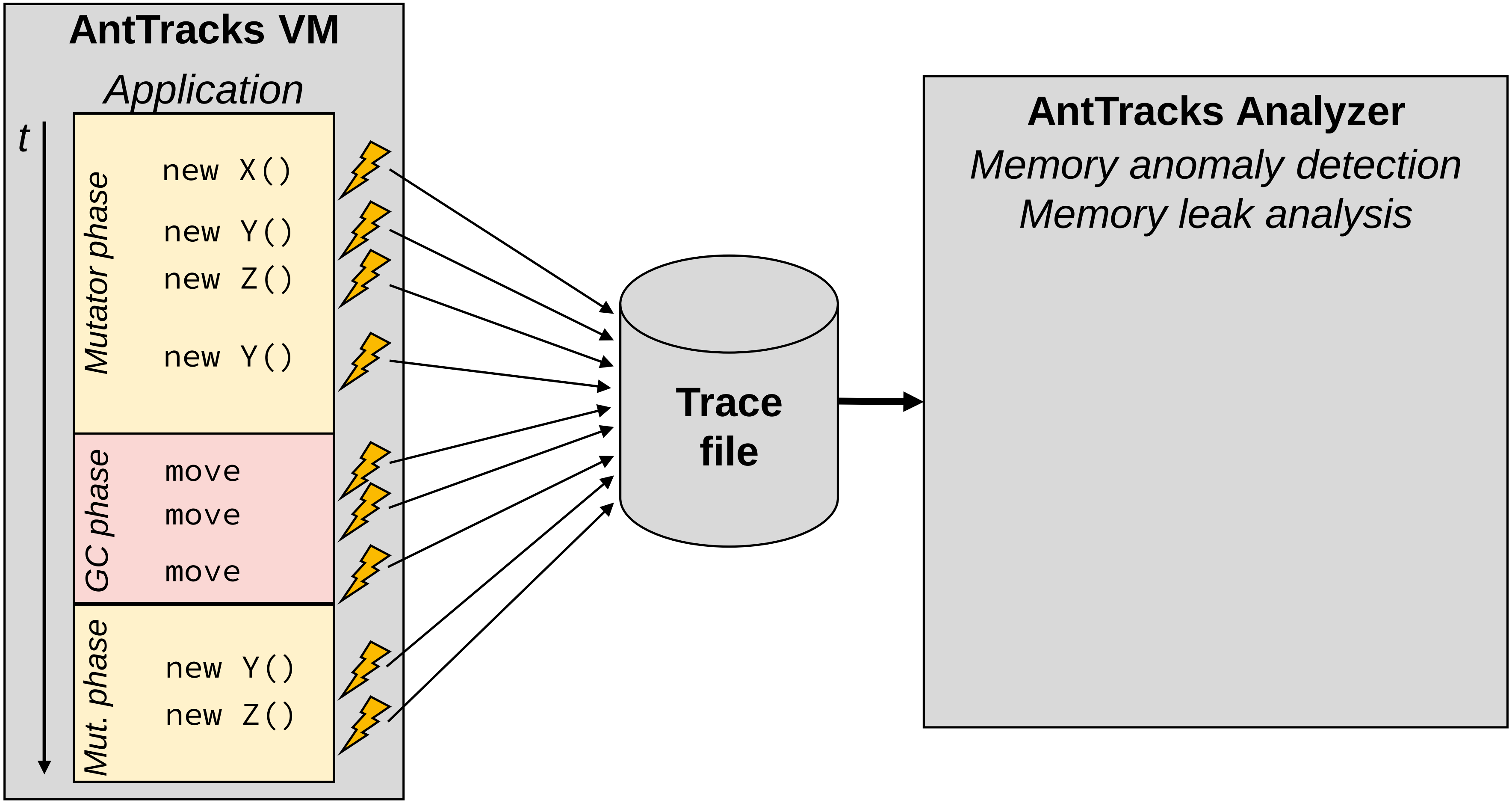
ANTTRACKS SYSTEM OVERVIEW



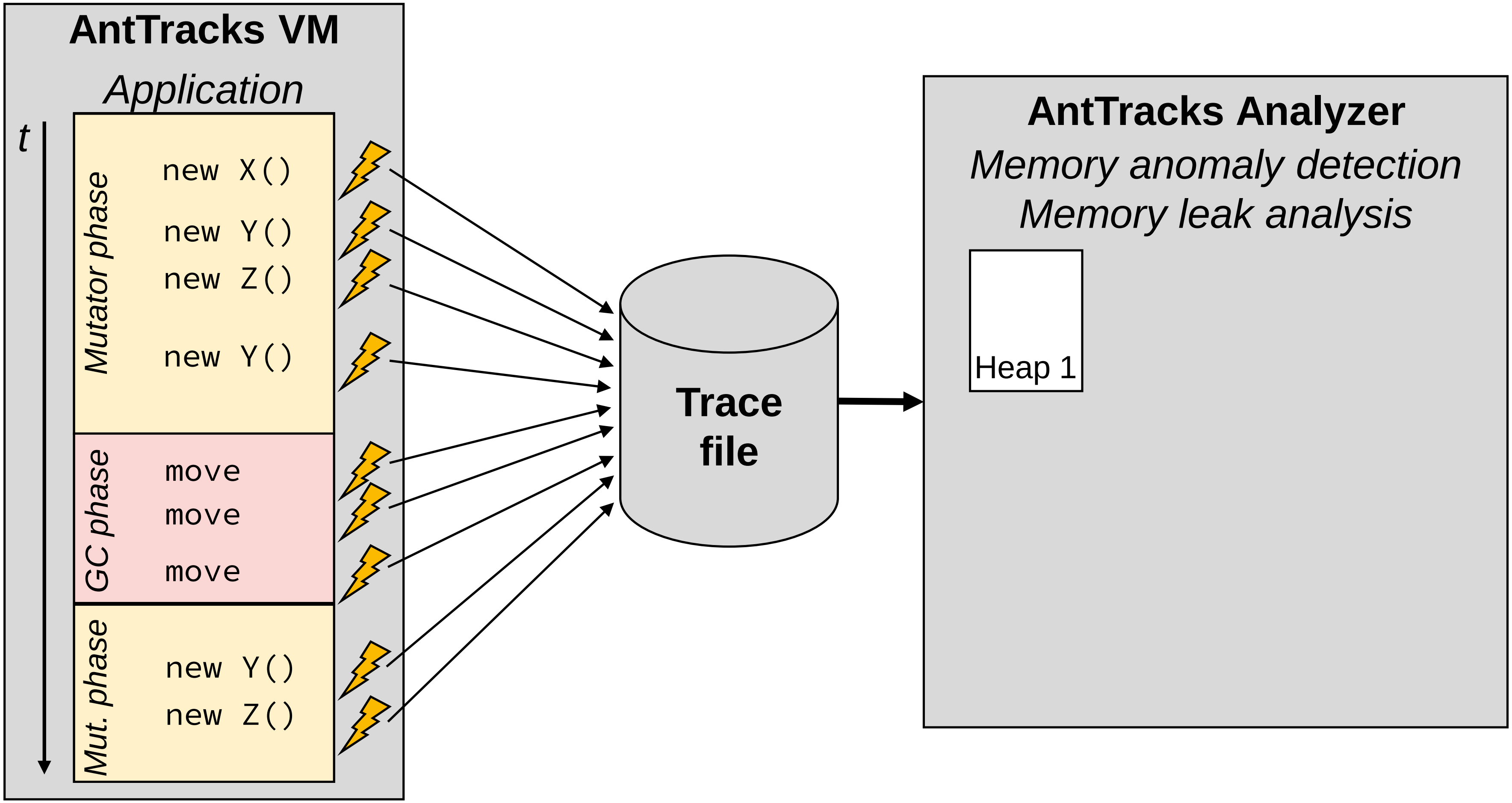
ANTTRACKS SYSTEM OVERVIEW



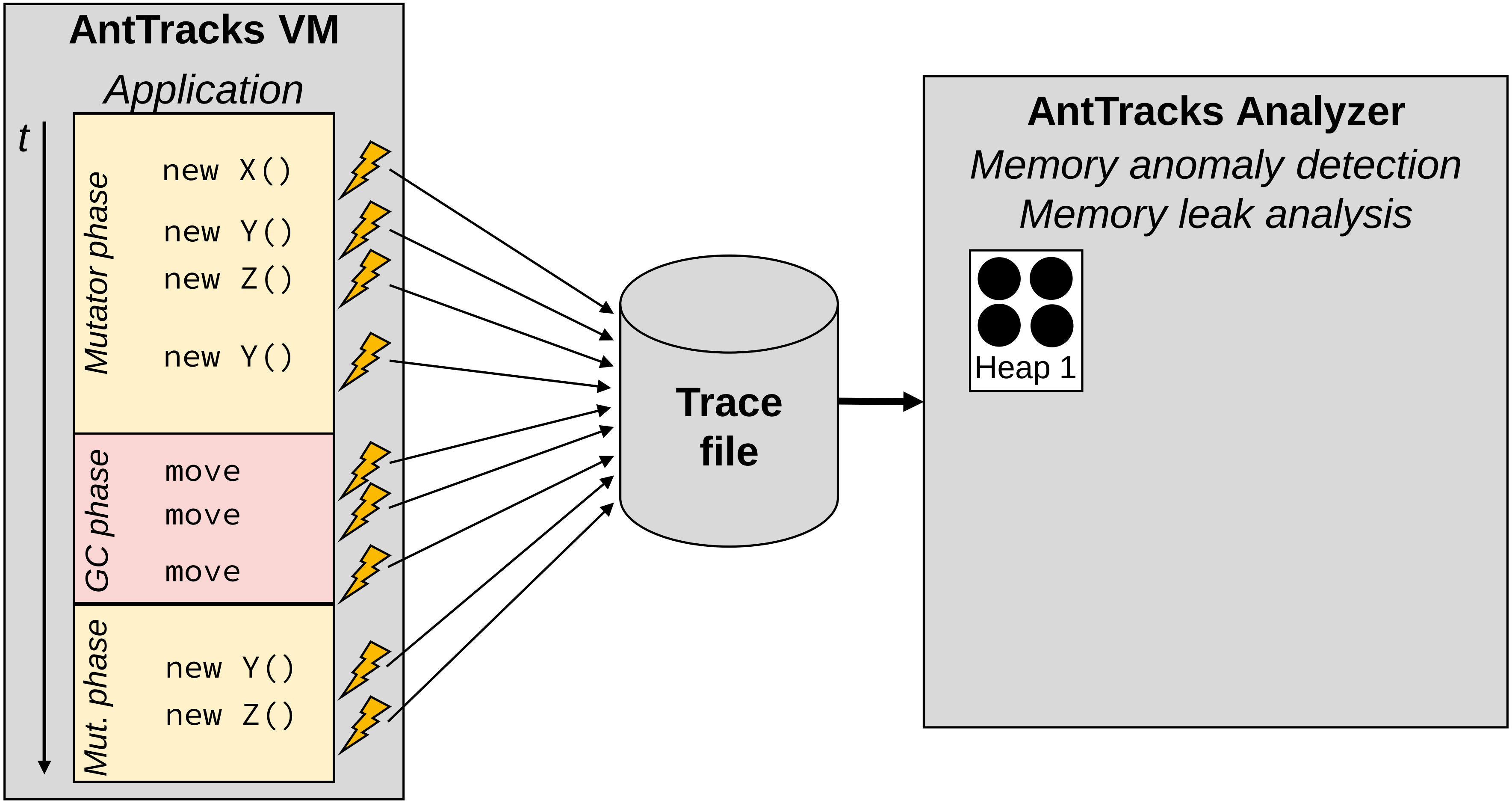
ANTTRACKS SYSTEM OVERVIEW



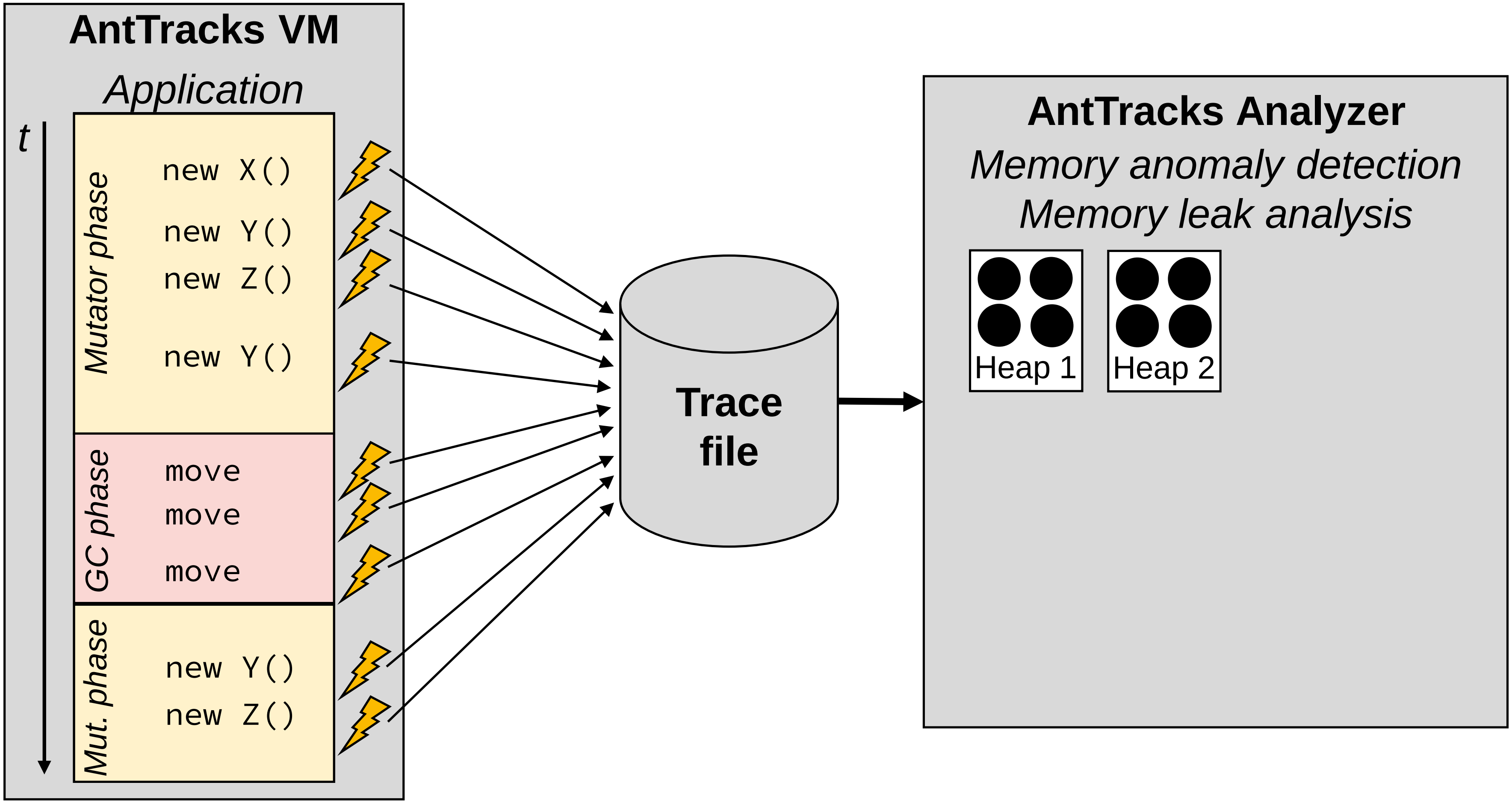
ANTTRACKS SYSTEM OVERVIEW



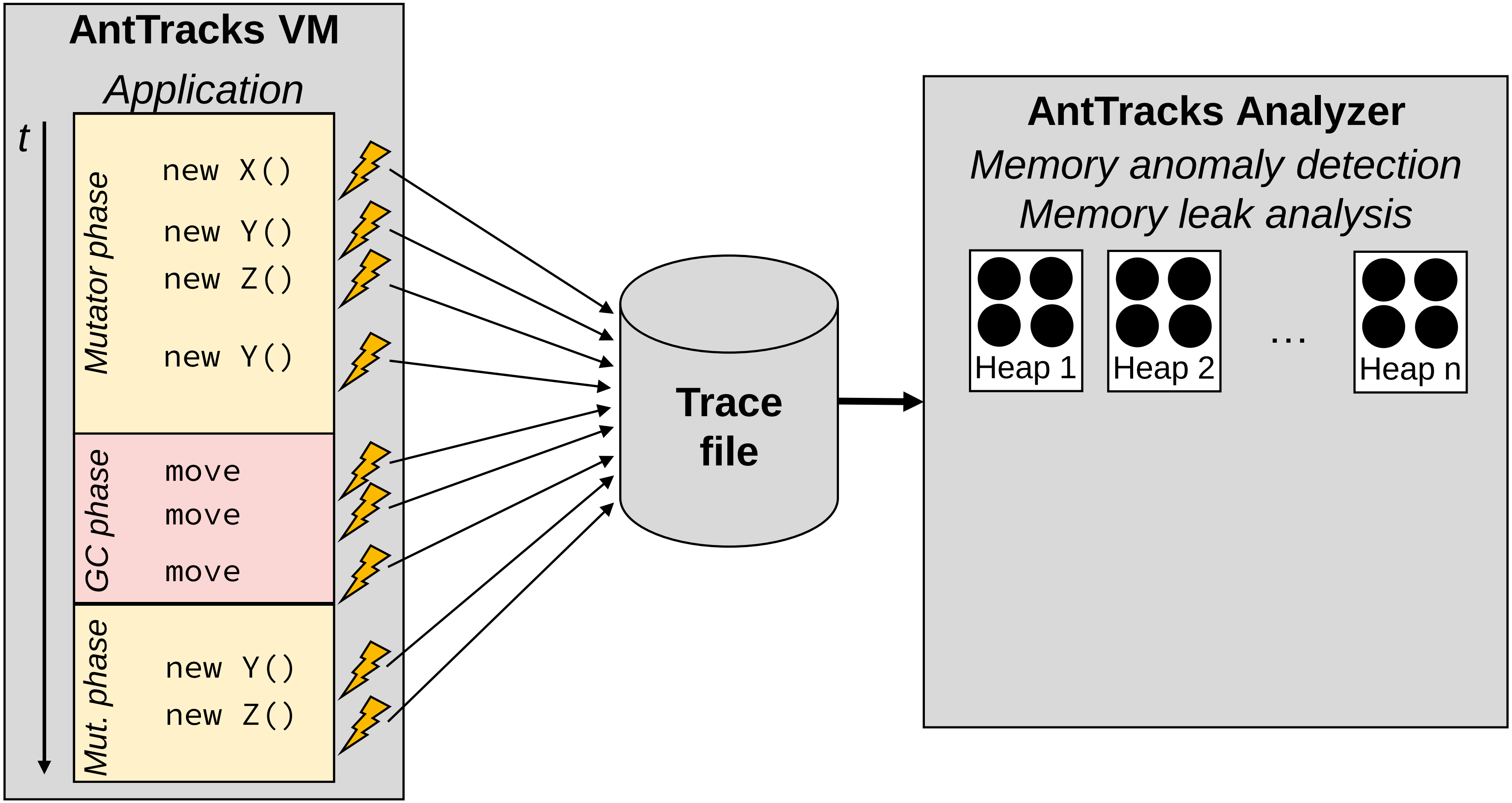
ANTTRACKS SYSTEM OVERVIEW



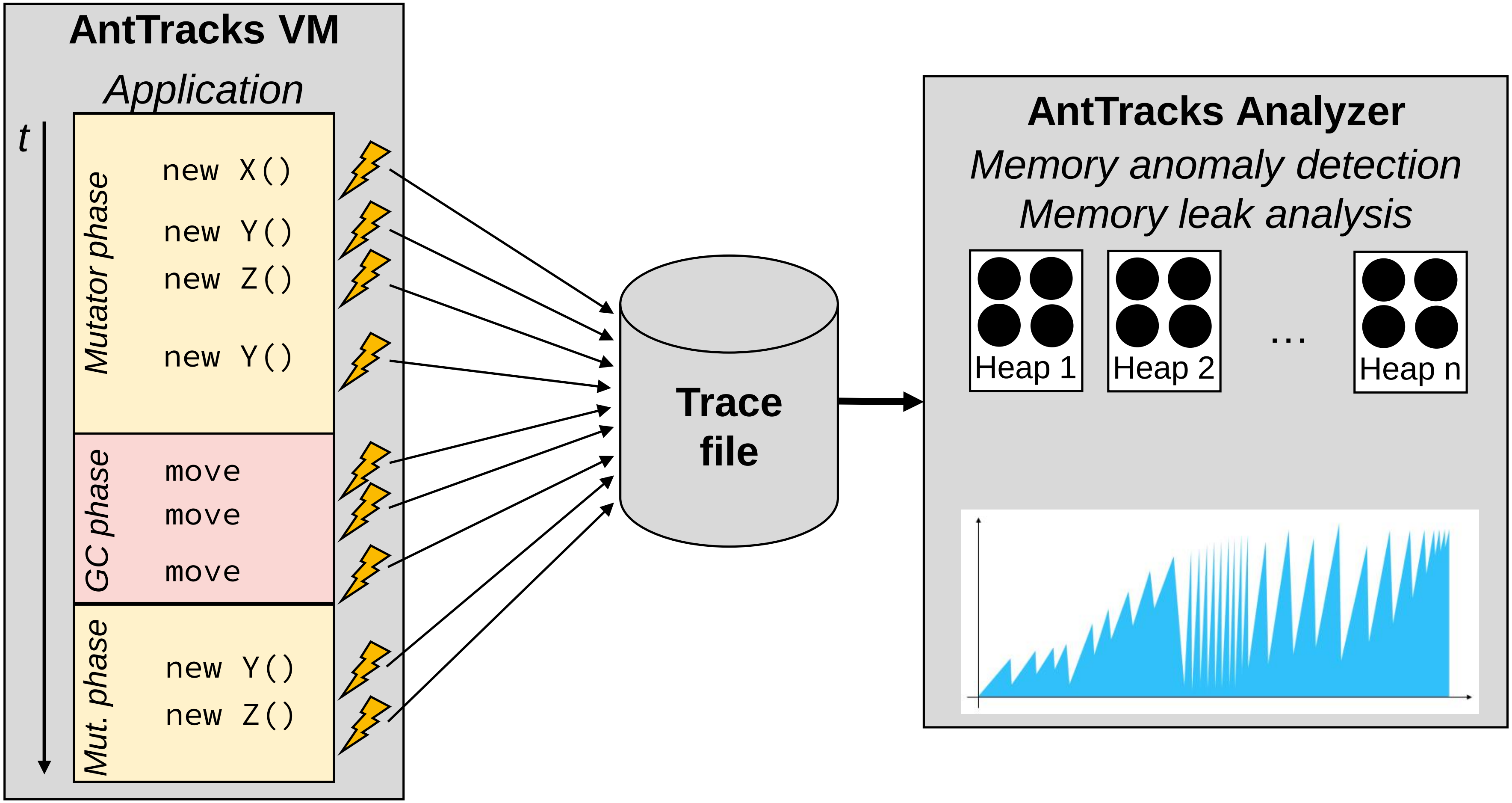
ANTTRACKS SYSTEM OVERVIEW



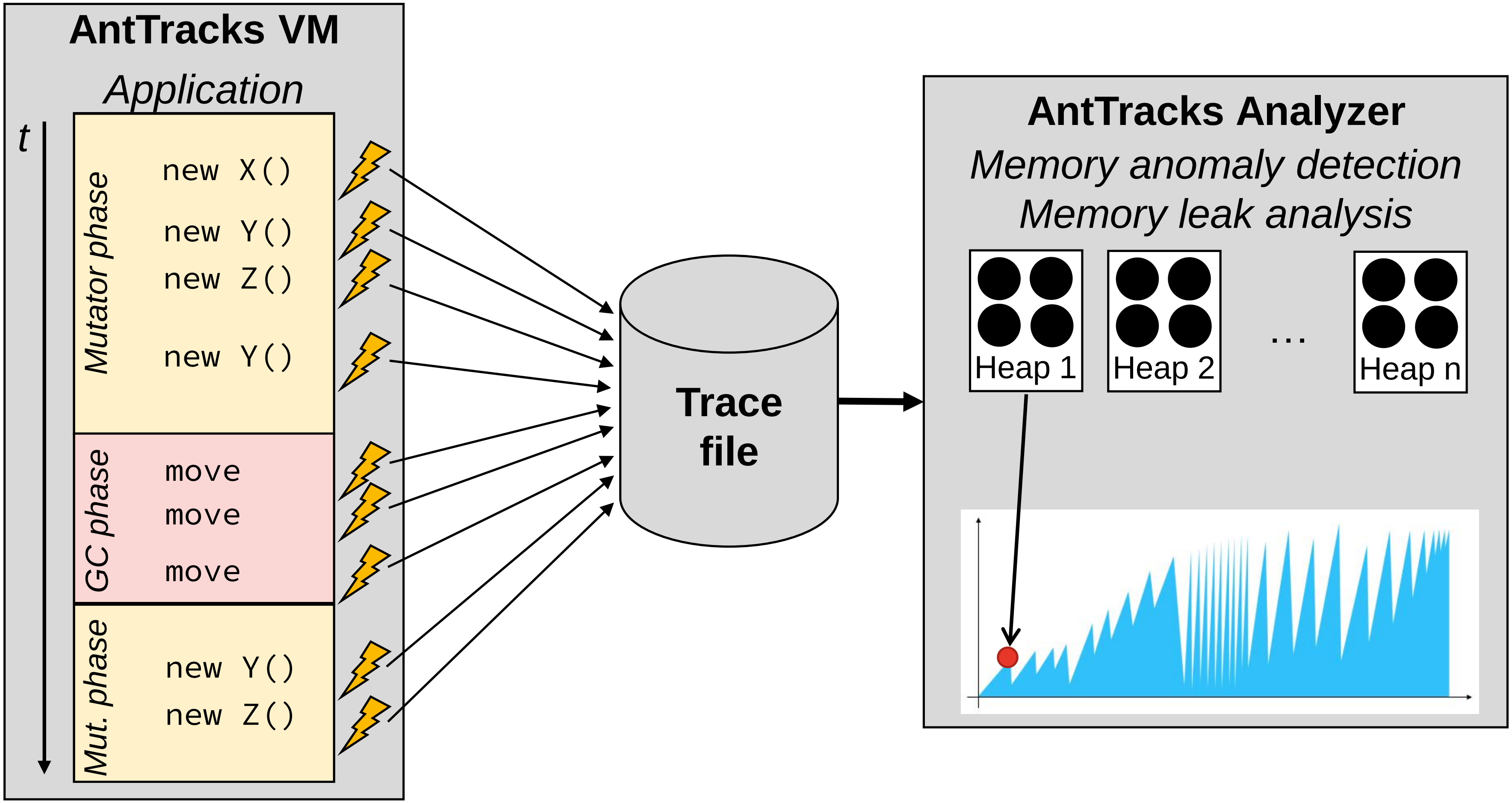
ANTTRACKS SYSTEM OVERVIEW



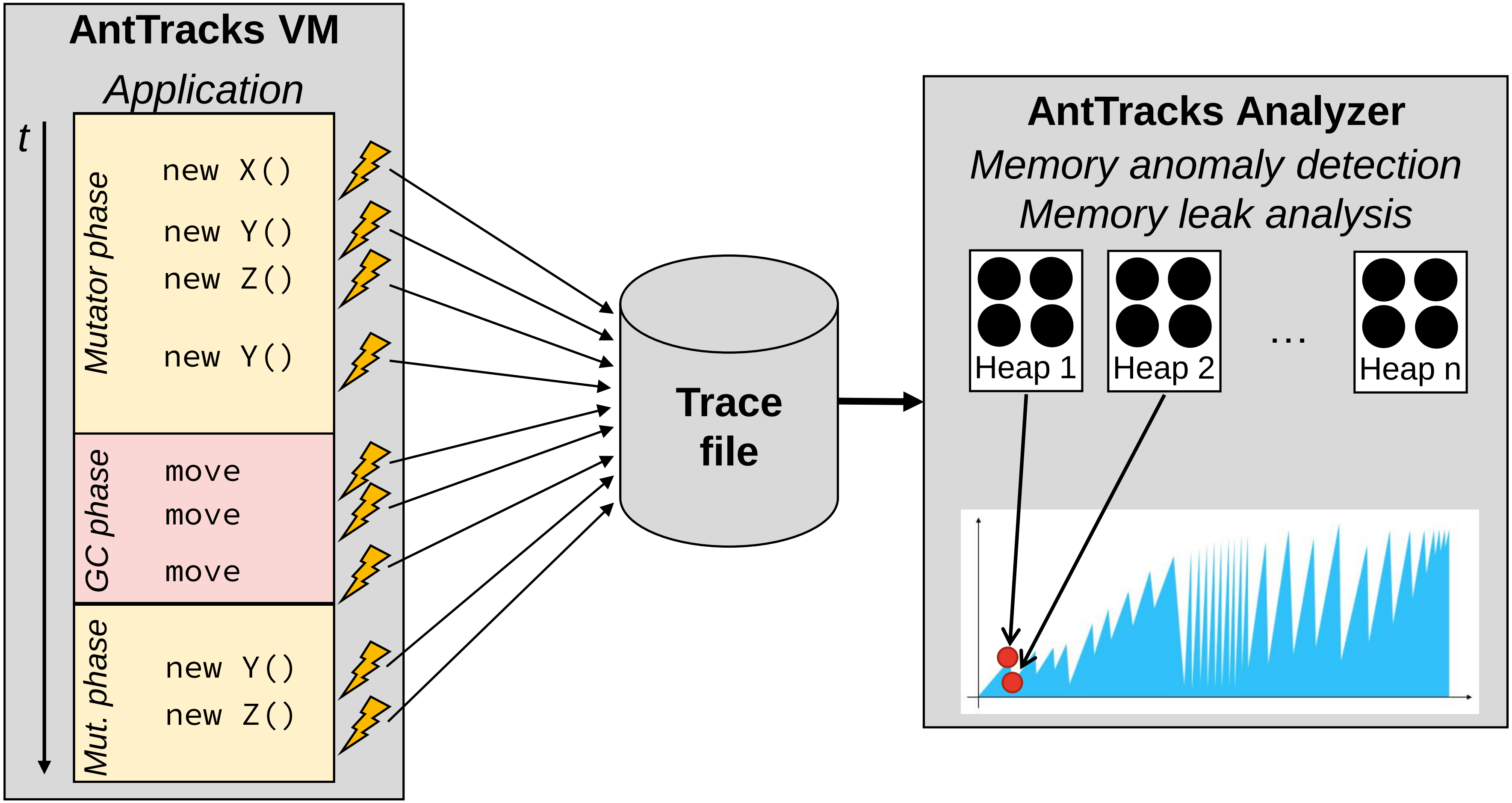
ANTTRACKS SYSTEM OVERVIEW



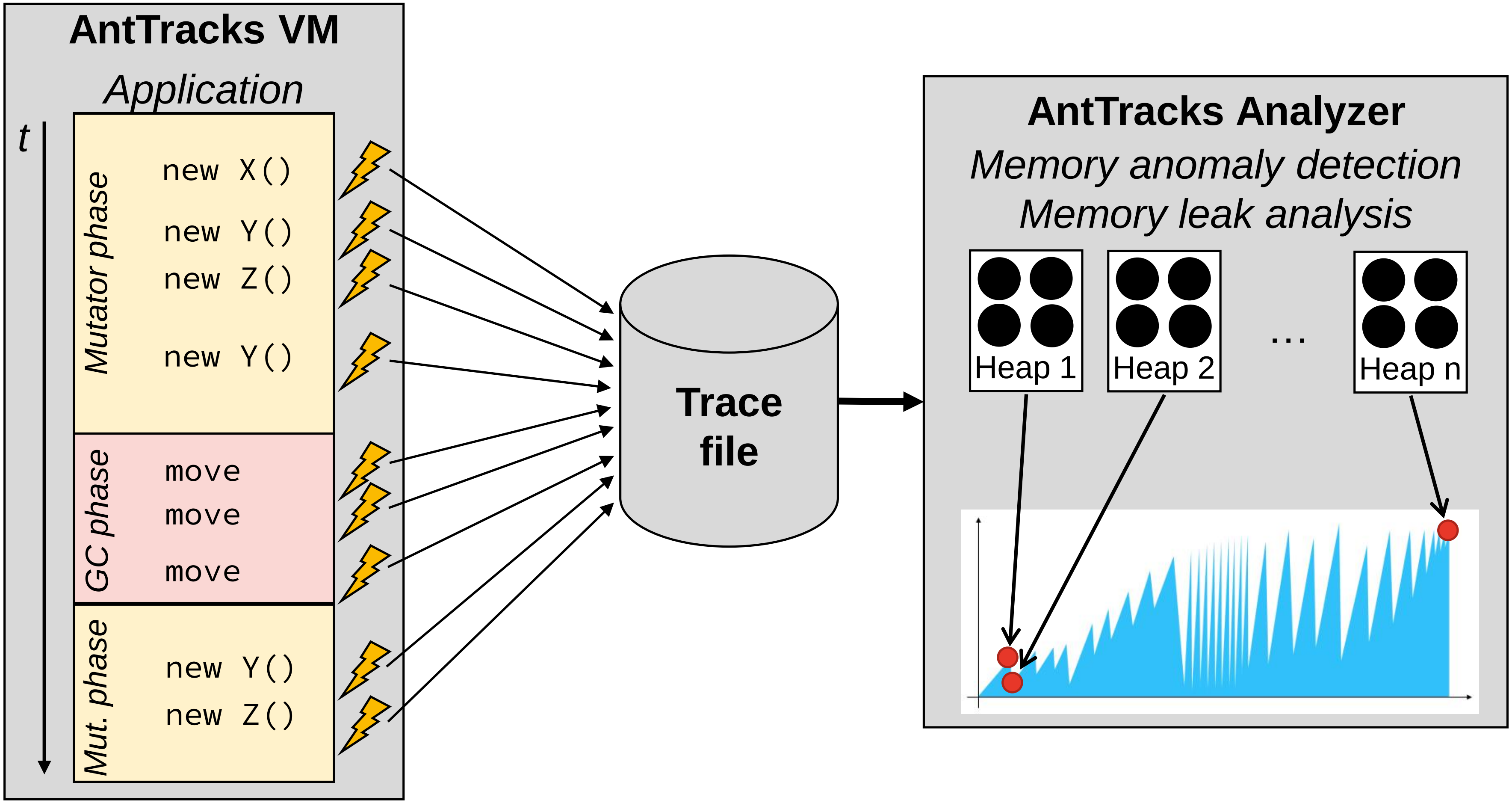
ANTTRACKS SYSTEM OVERVIEW



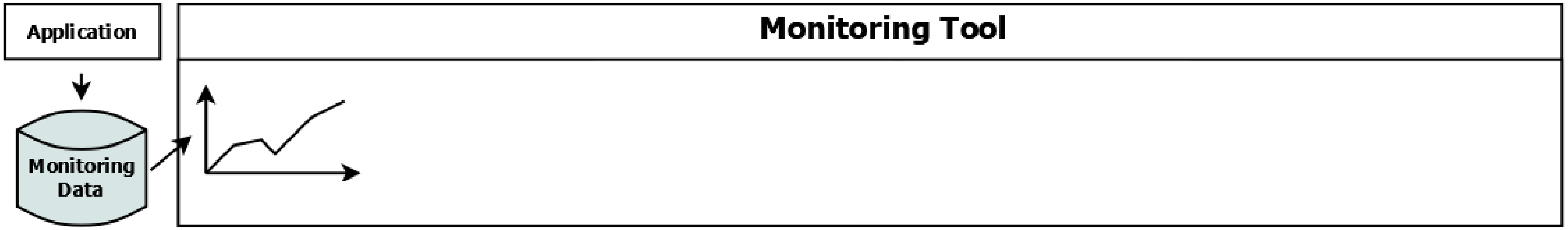
ANTTRACKS SYSTEM OVERVIEW



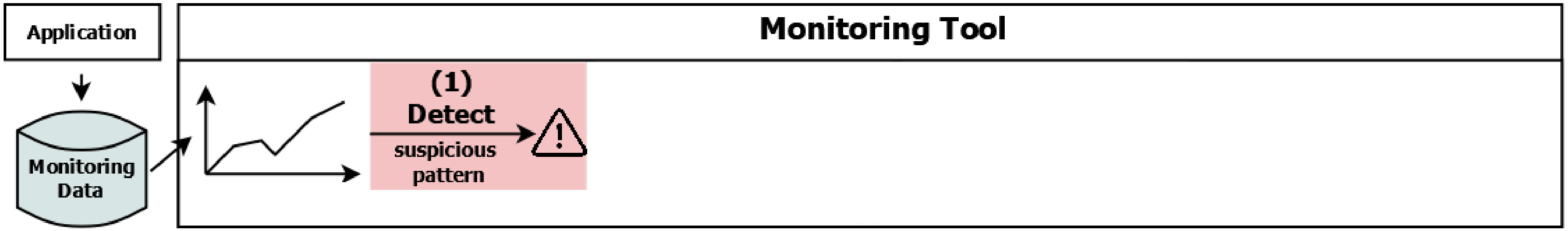
ANTTRACKS SYSTEM OVERVIEW



GUIDED EXPLORATION

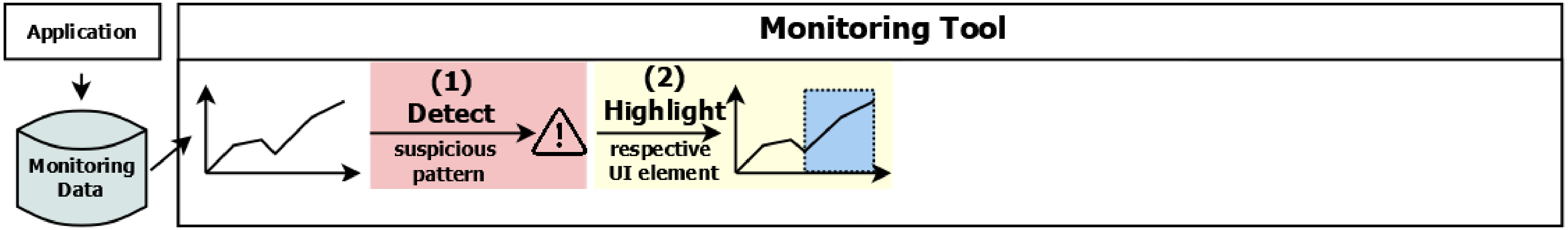


GUIDED EXPLORATION



Detection describes the task of automatically detecting a potential problem, i.e., a suspicious pattern.

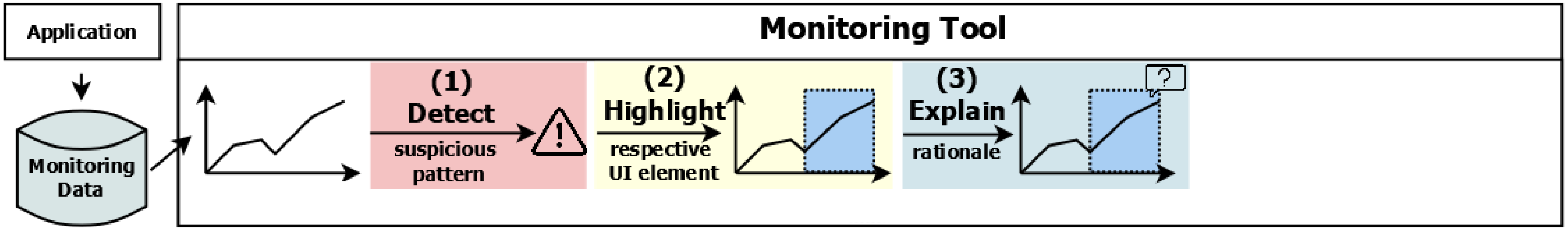
GUIDED EXPLORATION



Detection describes the task of automatically detecting a potential problem, i.e., a suspicious pattern.

Highlighting the relevant region on the UI helps users to understand where the automatically gained insight can be found if the view was inspected manually.

GUIDED EXPLORATION

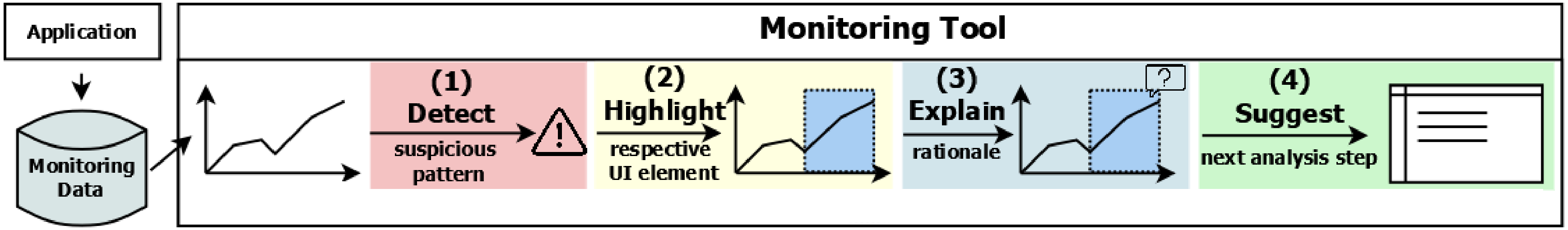


Detection describes the task of automatically detecting a potential problem, i.e., a suspicious pattern.

Highlighting the relevant region on the UI helps users to understand where the automatically gained insight can be found if the view was inspected manually.

Explanation of the highlighted visualization helps users to interpret it and explains concepts that are needed for this interpretation.

GUIDED EXPLORATION



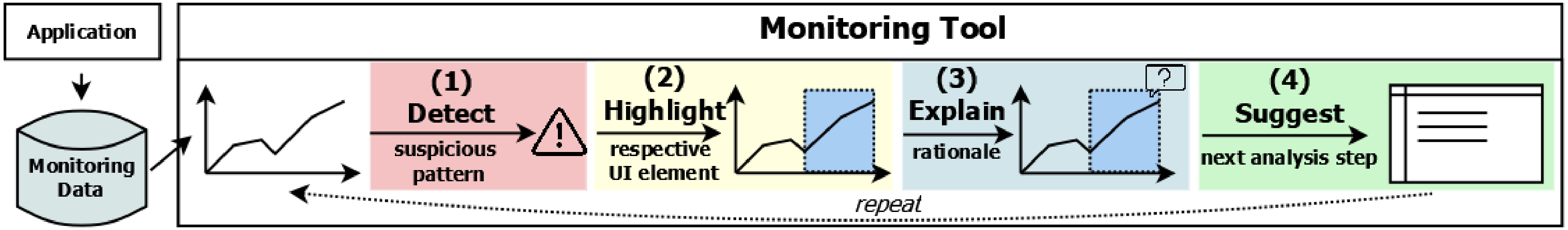
Detection describes the task of automatically detecting a potential problem, i.e., a suspicious pattern.

Highlighting the relevant region on the UI helps users to understand where the automatically gained insight can be found if the view was inspected manually.

Explanation of the highlighted visualization helps users to interpret it and explains concepts that are needed for this interpretation.

Suggestions on which steps could / should be taken next make it easier for the user to understand what operations are possible and why they might be useful.

GUIDED EXPLORATION



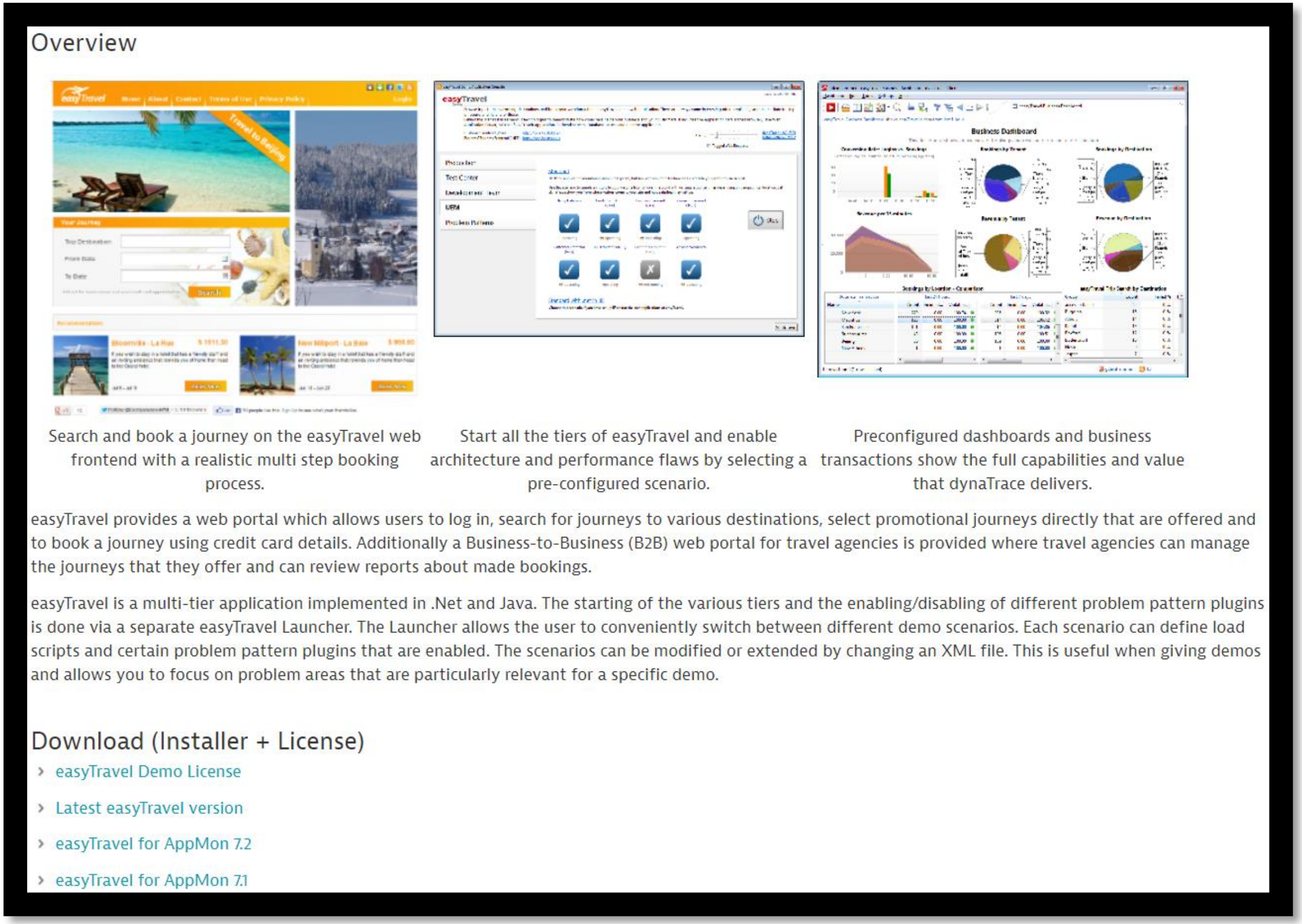
Detection describes the task of automatically detecting a potential problem, i.e., a suspicious pattern.

Highlighting the relevant region on the UI helps users to understand where the automatically gained insight can be found if the view was inspected manually.

Explanation of the highlighted visualization helps users to interpret it and explains concepts that are needed for this interpretation.

Suggestions on which steps could / should be taken next make it easier for the user to understand what operations are possible and why they might be useful.

DEMO: EASYTRAVEL



Overview

Search and book a journey on the easyTravel web frontend with a realistic multi step booking process.

Start all the tiers of easyTravel and enable architecture and performance flaws by selecting a pre-configured scenario.

Preconfigured dashboards and business transactions show the full capabilities and value that dynaTrace delivers.

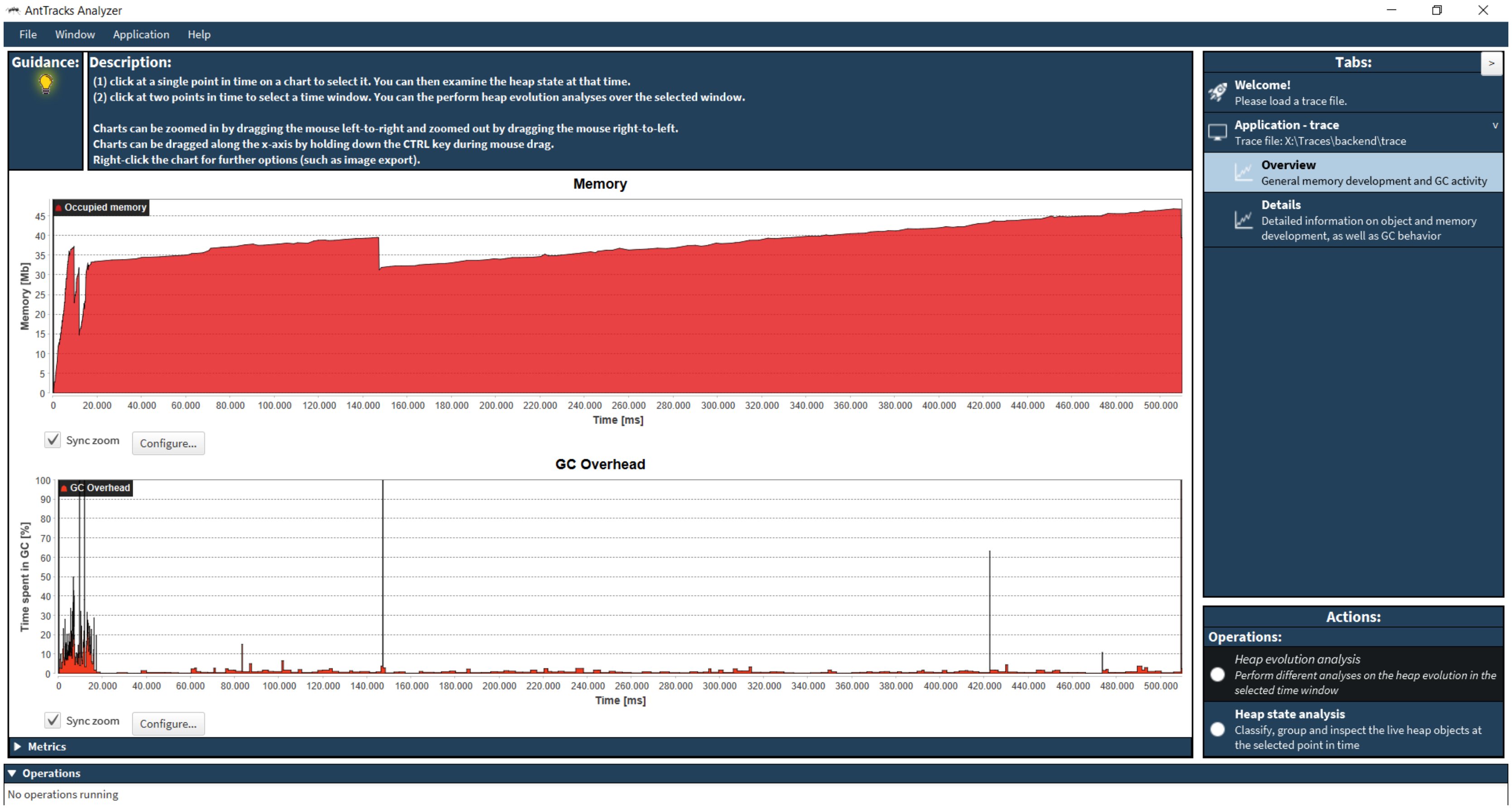
easyTravel provides a web portal which allows users to log in, search for journeys to various destinations, select promotional journeys directly that are offered and to book a journey using credit card details. Additionally a Business-to-Business (B2B) web portal for travel agencies is provided where travel agencies can manage the journeys that they offer and can review reports about made bookings.

easyTravel is a multi-tier application implemented in .Net and Java. The starting of the various tiers and the enabling/disabling of different problem pattern plugins is done via a separate easyTravel Launcher. The Launcher allows the user to conveniently switch between different demo scenarios. Each scenario can define load scripts and certain problem pattern plugins that are enabled. The scenarios can be modified or extended by changing an XML file. This is useful when giving demos and allows you to focus on problem areas that are particularly relevant for a specific demo.

Download (Installer + License)

- easyTravel Demo License
- Latest easyTravel version
- easyTravel for AppMon 7.2
- easyTravel for AppMon 7.1

DEMO: OVERVIEW



Description:

(1) click at a single point in time on a chart to select it. You can then examine the heap state at that time.
(2) click at two points in time to select a time window. You can the perform heap evolution analyses over the selected window.

Charts can be zoomed in by dragging the mouse left-to-right and zoomed out by dragging the mouse right-to-left.
Charts can be dragged along the x-axis by holding down the CTRL key during mouse drag.
Right-click the chart for further options (such as image export).

Welcome!
Please load a trace file.

Application - trace
Trace file: X:\Traces\backend\trace

Overview
General memory development and GC activity

Details
Detailed information on object and memory development, as well as GC behavior

Actions:

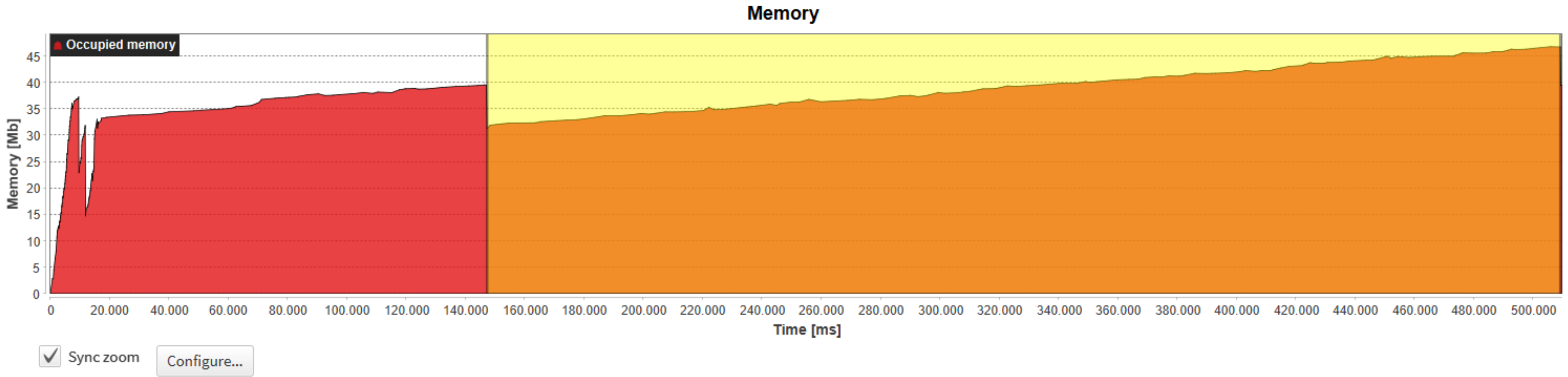
Operations:

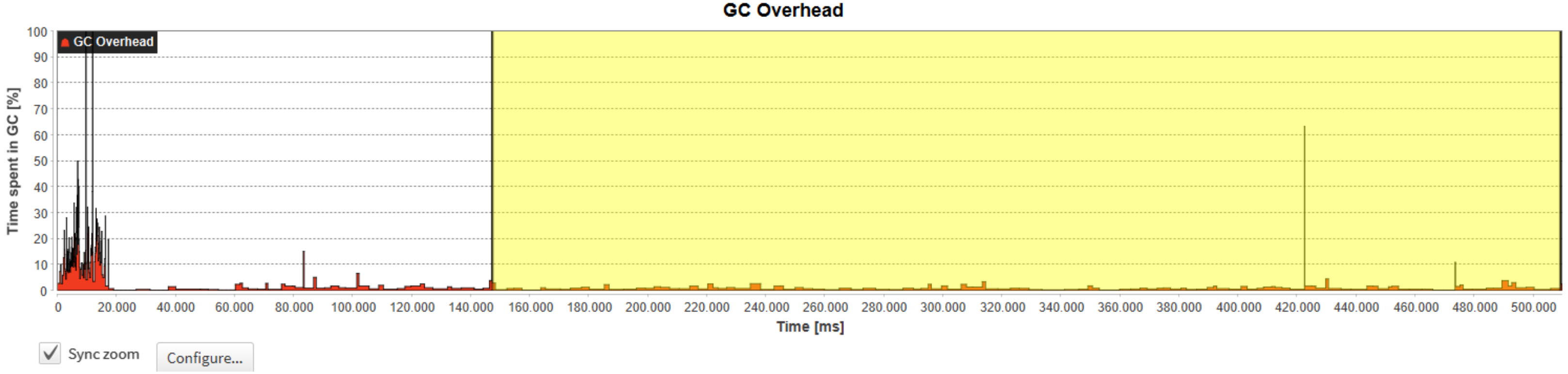
○ *Heap evolution analysis*
Perform different analyses on the heap evolution in the selected time window

○ **Heap state analysis**
Classify, group and inspect the live heap objects at the selected point in time

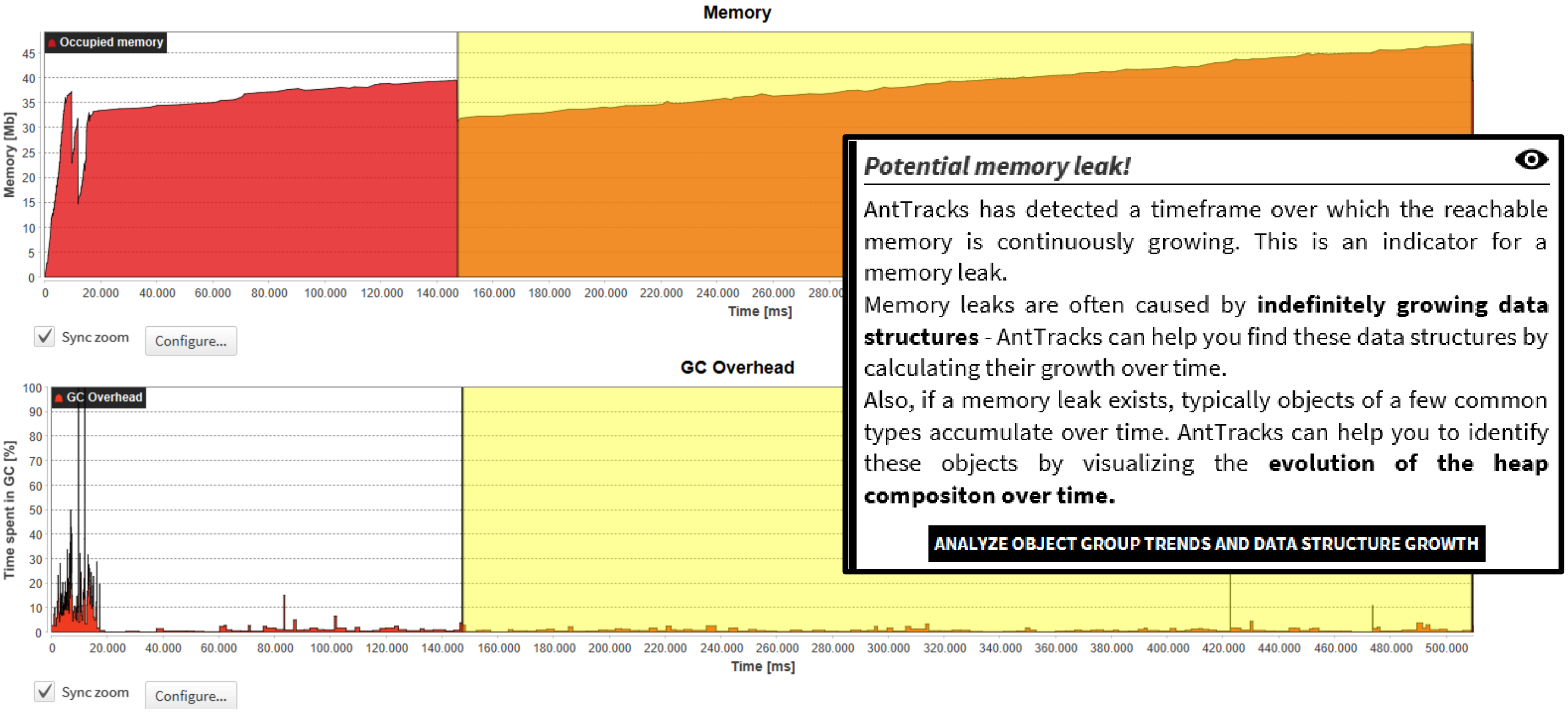
No operations running

DEMO: OVERVIEW
(1) DETECTION + (2) HIGHLIGHTING





DEMO: OVERVIEW
(3) EXPLANATION + (4) SUGGESTION



Potential memory leak!

AntTracks has detected a timeframe over which the reachable memory is continuously growing. This is an indicator for a memory leak.

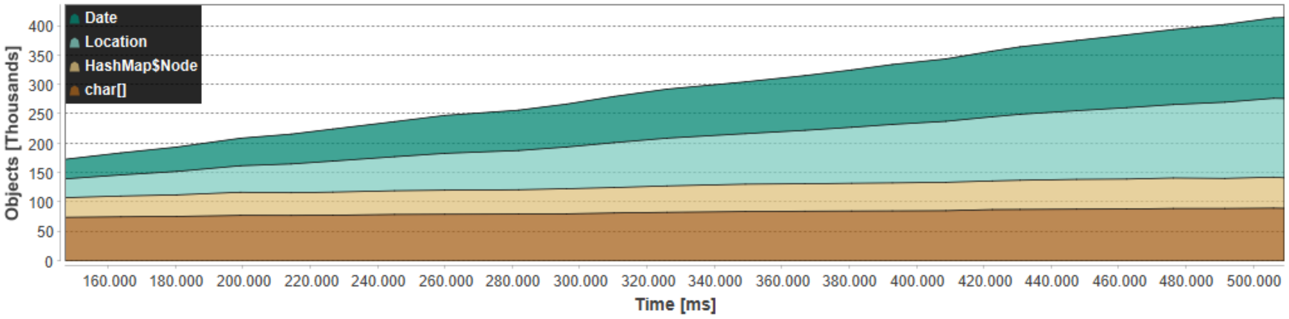
Memory leaks are often caused by **indefinitely growing data structures** - AntTracks can help you find these data structures by calculating their growth over time.

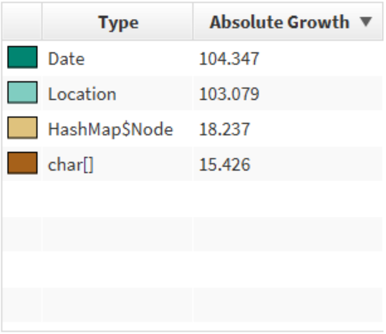
Also, if a memory leak exists, typically objects of a few common types accumulate over time. AntTracks can help you to identify these objects by visualizing the **evolution of the heap compositon over time.**

ANALYZE OBJECT GROUP TRENDS AND DATA STRUCTURE GROWTH

DEMO: EVOLUTION

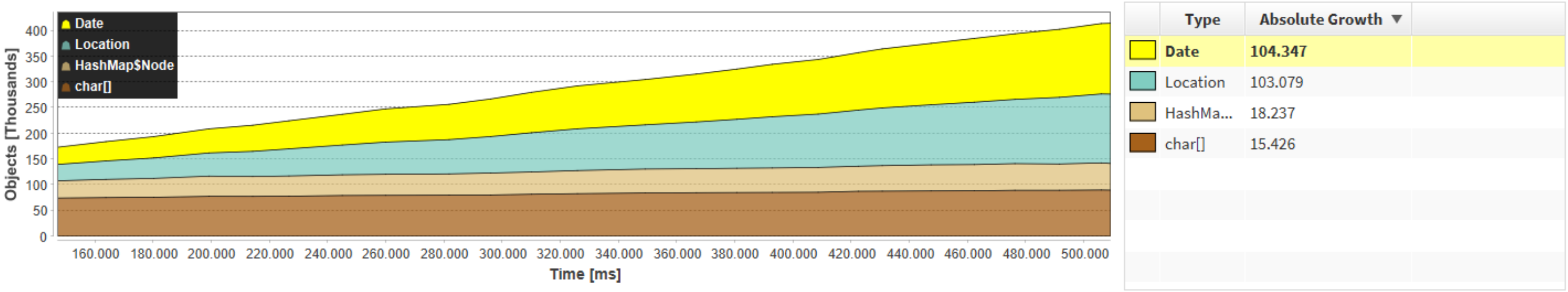
Type:





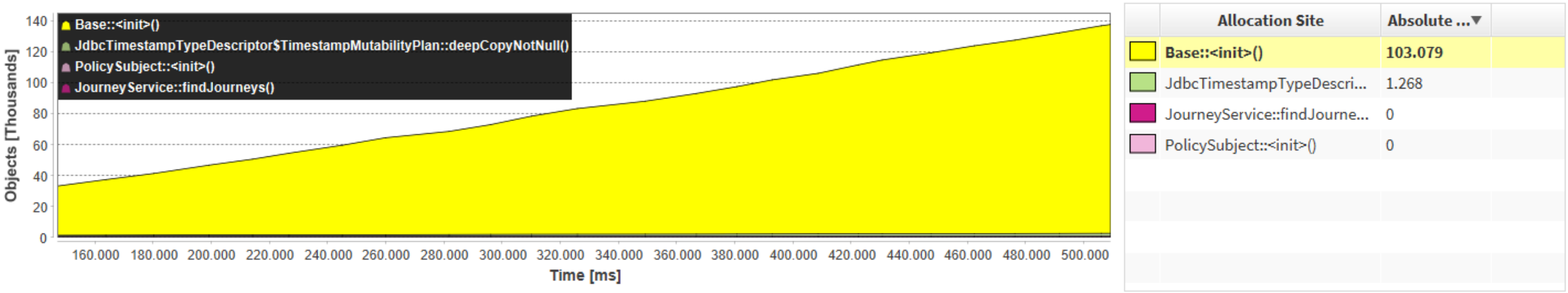
	Type	Absolute Growth ▼
🟩	Date	104.347
🟩	Location	103.079
🟨	HashMap$Node	18.237
🟫	char[]	15.426

DEMO: EVOLUTION



	Type	Absolute Growth ▼
🟨	**Date**	**104.347**
🟦	Location	103.079
🟧	HashMa...	18.237
🟫	char[]	15.426

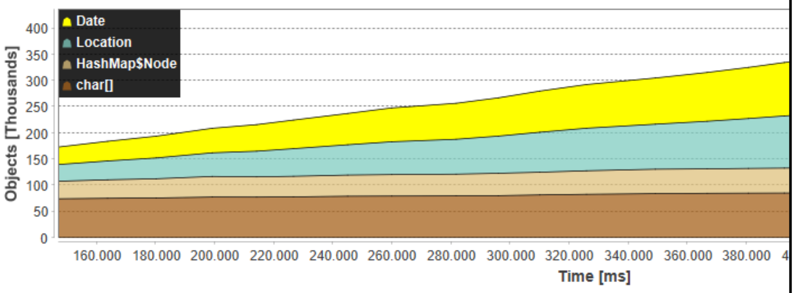
Drill-down selection: (1) Type: Date
Allocation Site:



	Allocation Site	Absolute ...▼
🟨	**Base::<init>()**	**103.079**
🟩	JdbcTimestampTypeDescri...	1.268
🟪	JourneyService::findJourne...	0
▨	PolicySubject::<init>()	0

Drill-down selection: (1) Type: Date >>> (2) Allocation Site: Base::<init>()
Call Sites:

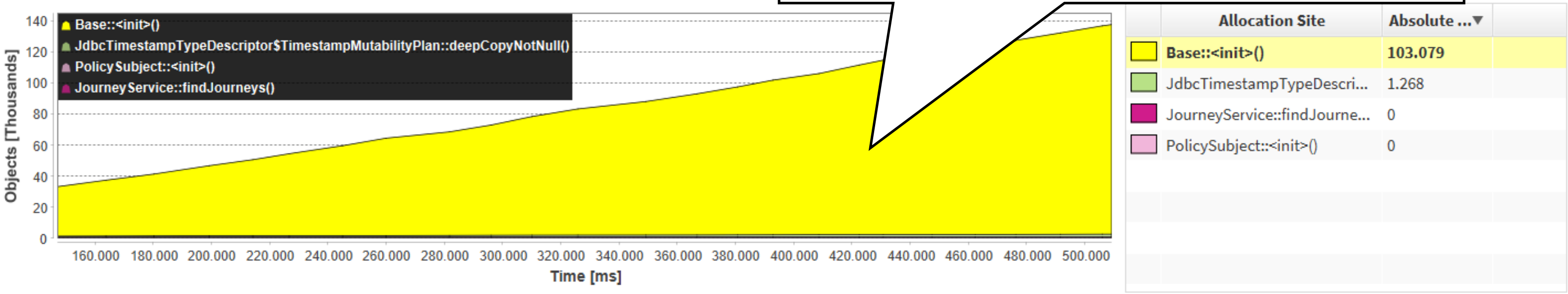
DEMO: EVOLUTION



Date objects allocated in the constructor of **Base** are the major suspects for a memory leak since about 30% of the overall heap growth is accounted to them.

Inspect who keeps them alive.

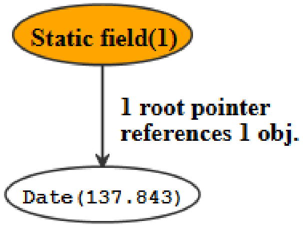
Drill-down selection:
Allocation



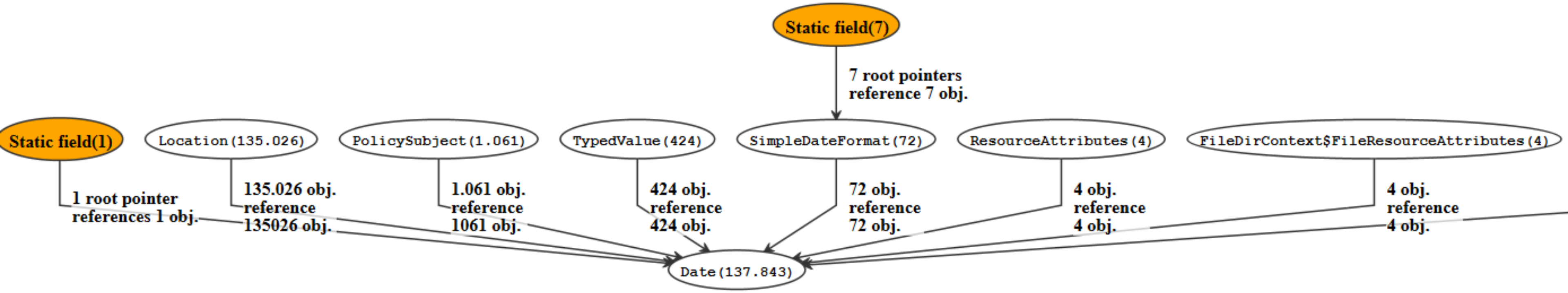
	Allocation Site	Absolute ...▼
🟨	**Base::<init>()**	**103.079**
🟩	JdbcTimestampTypeDescri...	1.268
🟥	JourneyService::findJourne...	0
🟪	PolicySubject::<init>()	0

Drill-down selection: (1) Type: Date >>> (2) Allocation Site: Base::<init>()
Call Sites:

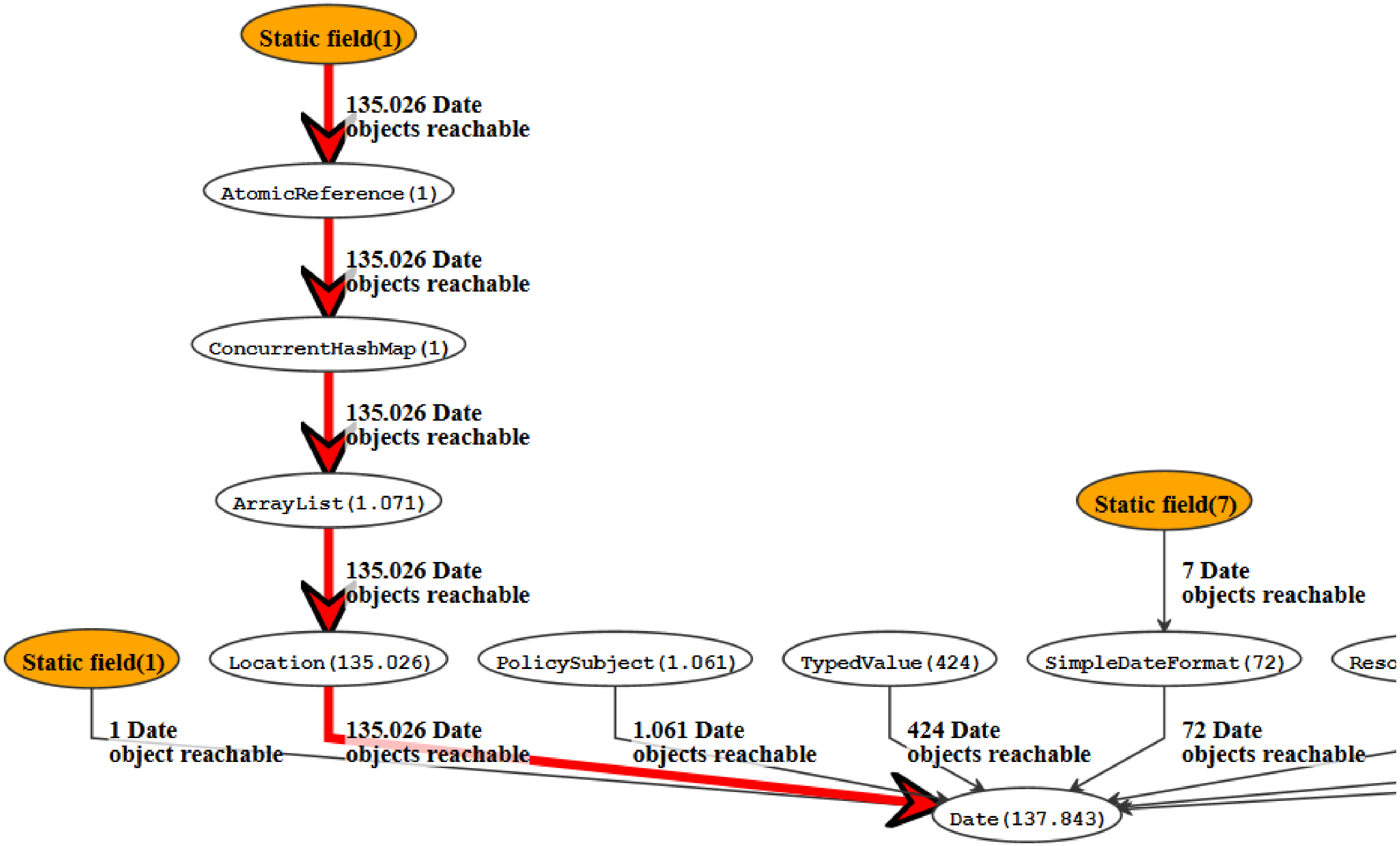
DEMO: GRAPH VIEW



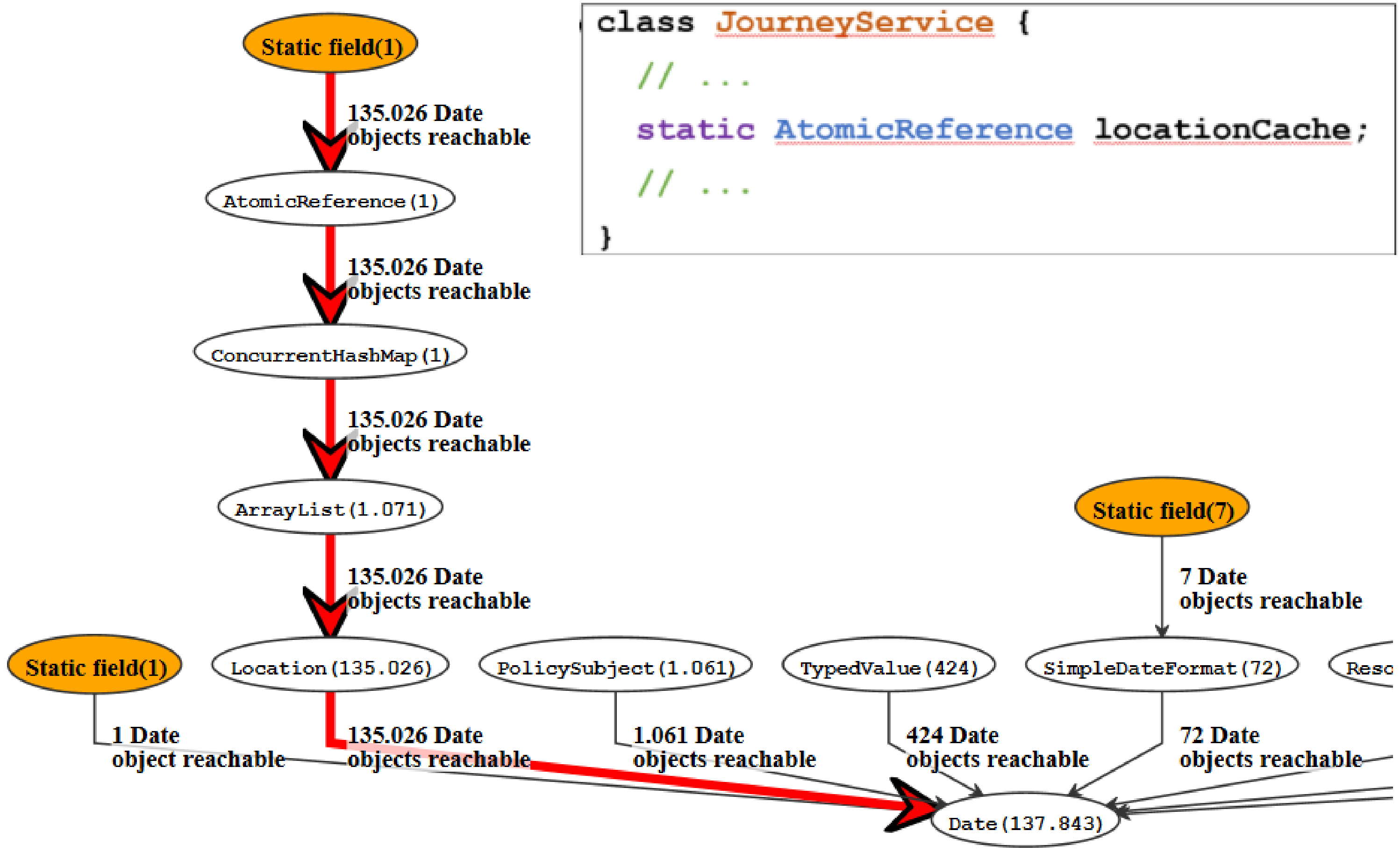
DEMO: GRAPH VIEW


Static field(7)
7 root pointers reference 7 obj.
Static field(1)
Location(135.026)
PolicySubject(1.061)
TypedValue(424)
SimpleDateFormat(72)
ResourceAttributes(4)
FileDirContext$FileResourceAttributes(4)
1 root pointer references 1 obj.
135.026 obj. reference 135026 obj.
1.061 obj. reference 1061 obj.
424 obj. reference 424 obj.
72 obj. reference 72 obj.
4 obj. reference 4 obj.
4 obj. reference 4 obj.
Date(137.843)

DEMO: GRAPH VIEW

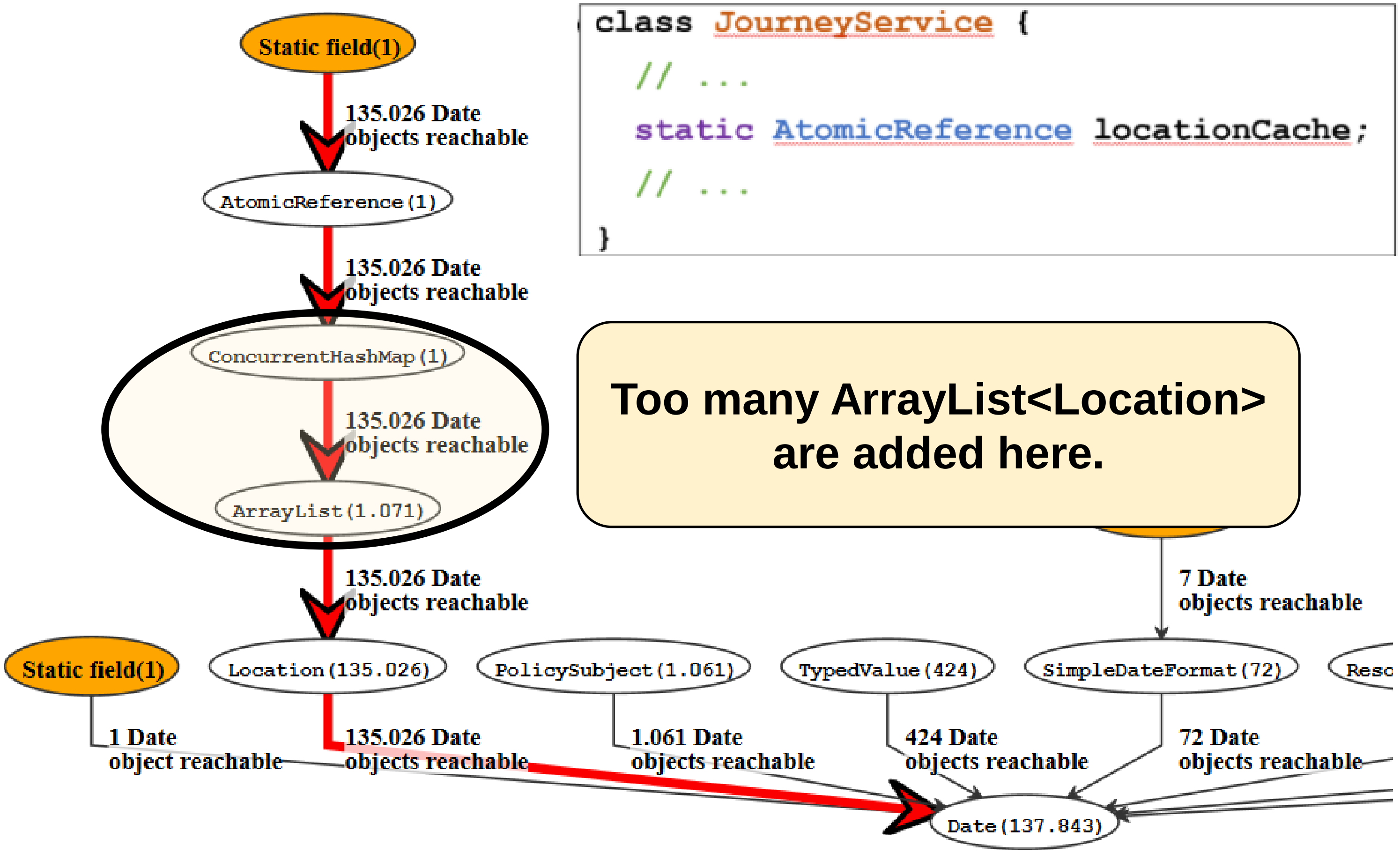


DEMO: GRAPH VIEW



```
class JourneyService {
  // ...
  static AtomicReference locationCache;
  // ...
}
```

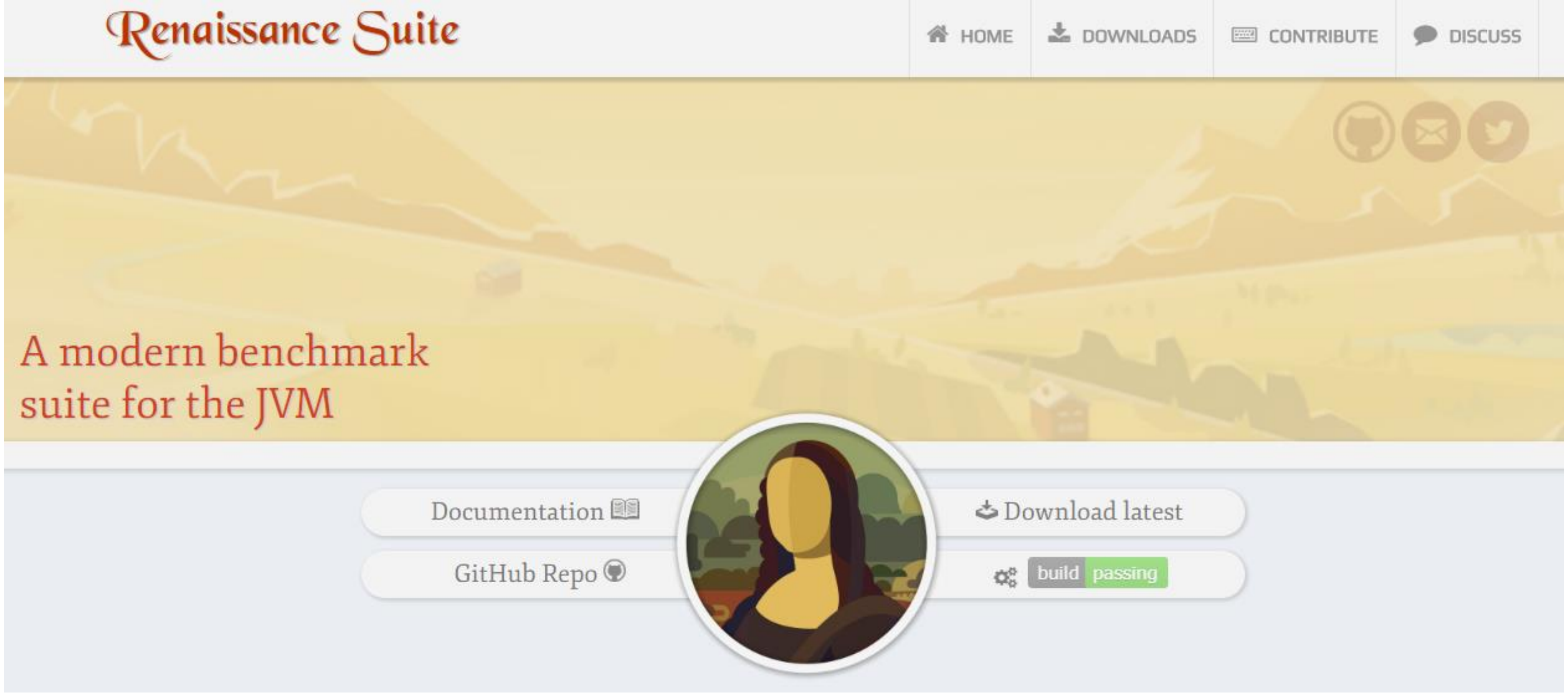
DEMO: GRAPH VIEW


Static field(1)
135.026 Date objects reachable
AtomicReference(1)
135.026 Date objects reachable
ConcurrentHashMap(1)
135.026 Date objects reachable
ArrayList(1.071)
135.026 Date objects reachable
7 Date objects reachable
Static field(1)
Location(135.026)
PolicySubject(1.061)
TypedValue(424)
SimpleDateFormat(72)
Reso
1 Date object reachable
135.026 Date objects reachable
1.061 Date objects reachable
424 Date objects reachable
72 Date objects reachable
Date(137.843)

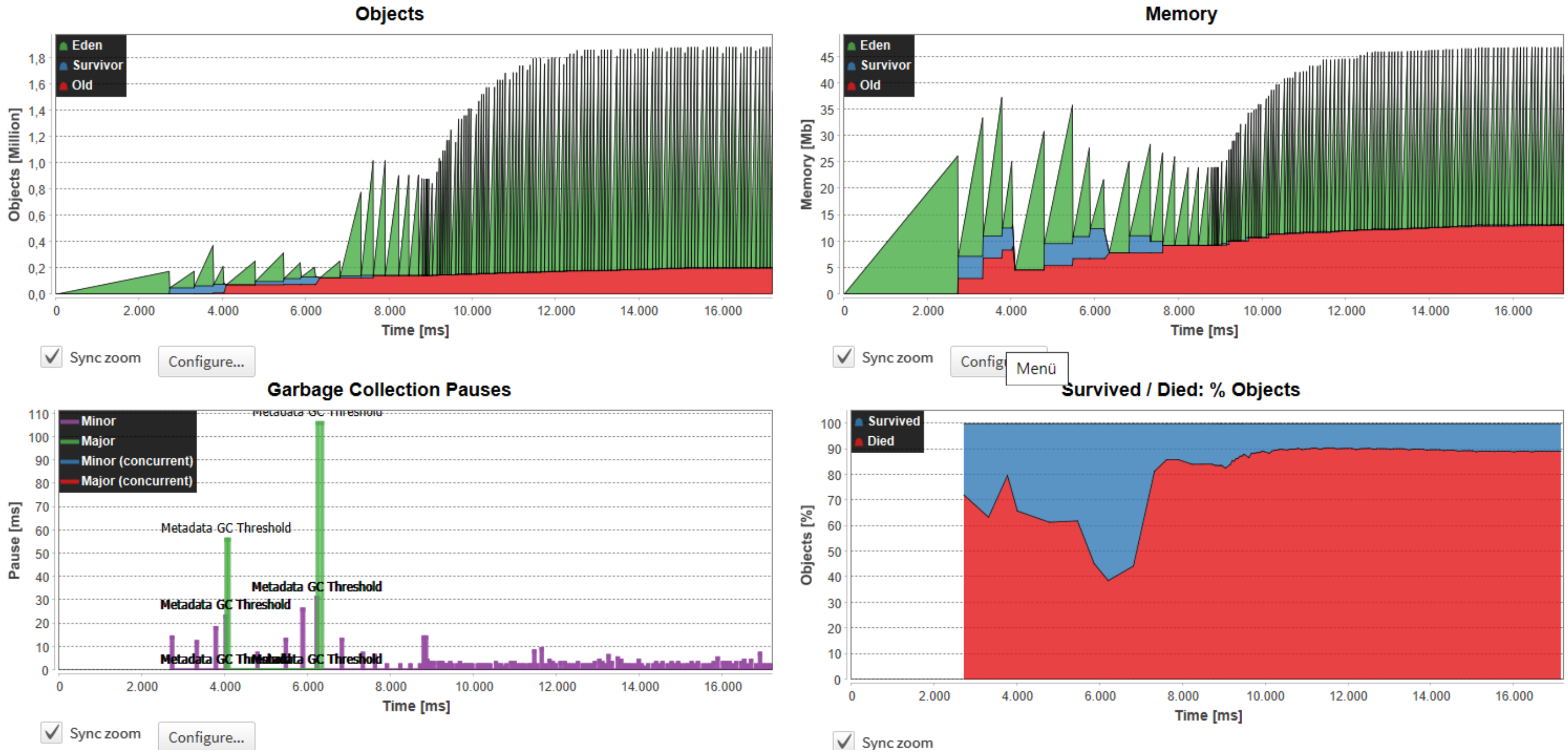
```
class JourneyService {
  // ...
  static AtomicReference locationCache;
  // ...
}
```

Too many ArrayList<Location> are added here.

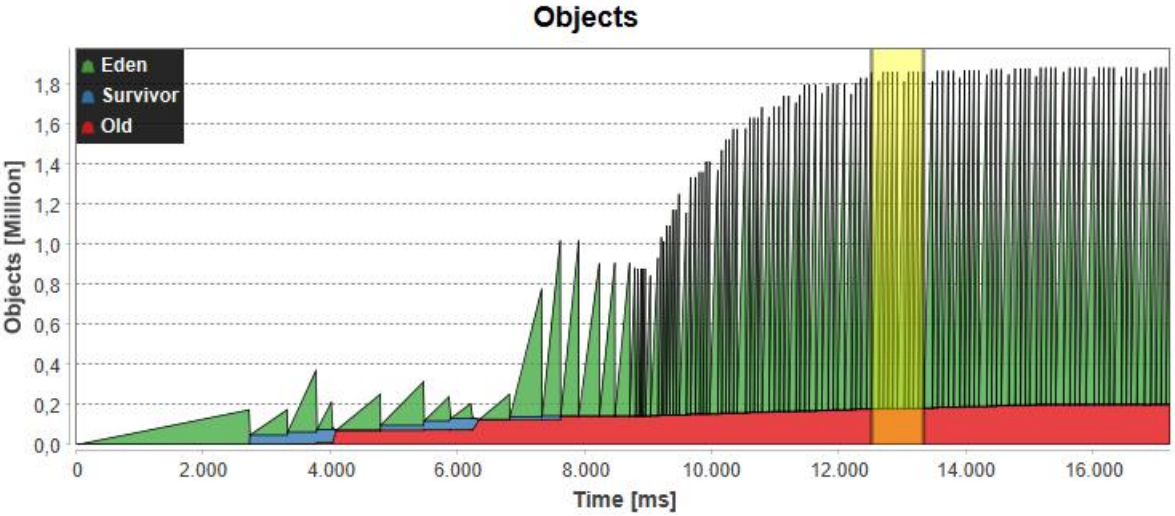
ANOTHER DEMO: FINAGLE-HTTP



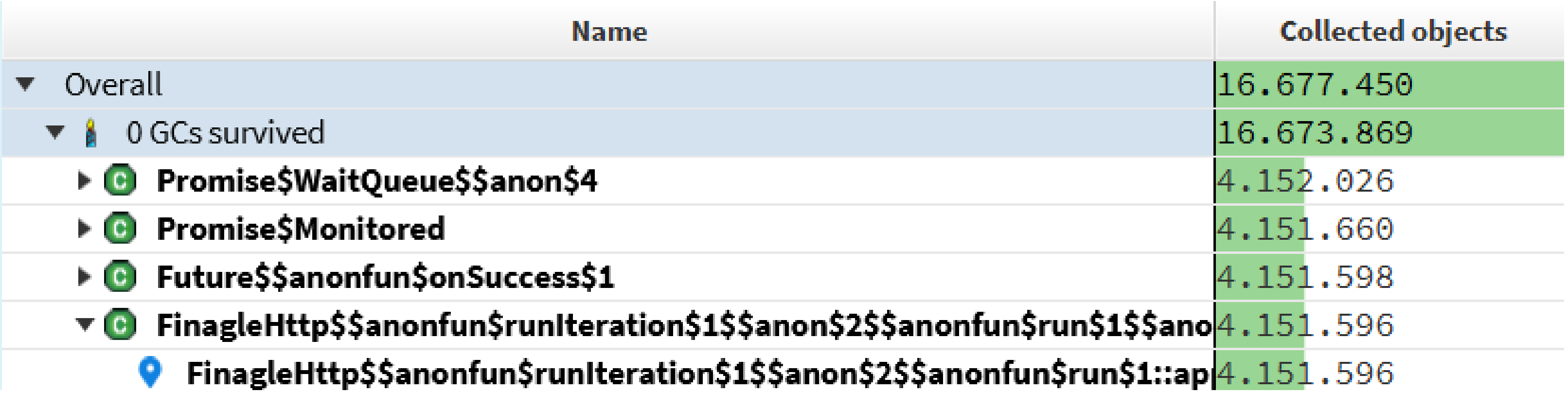
ANOTHER DEMO: FINAGLE-HTTP



ANOTHER DEMO: FINAGLE-HTTP

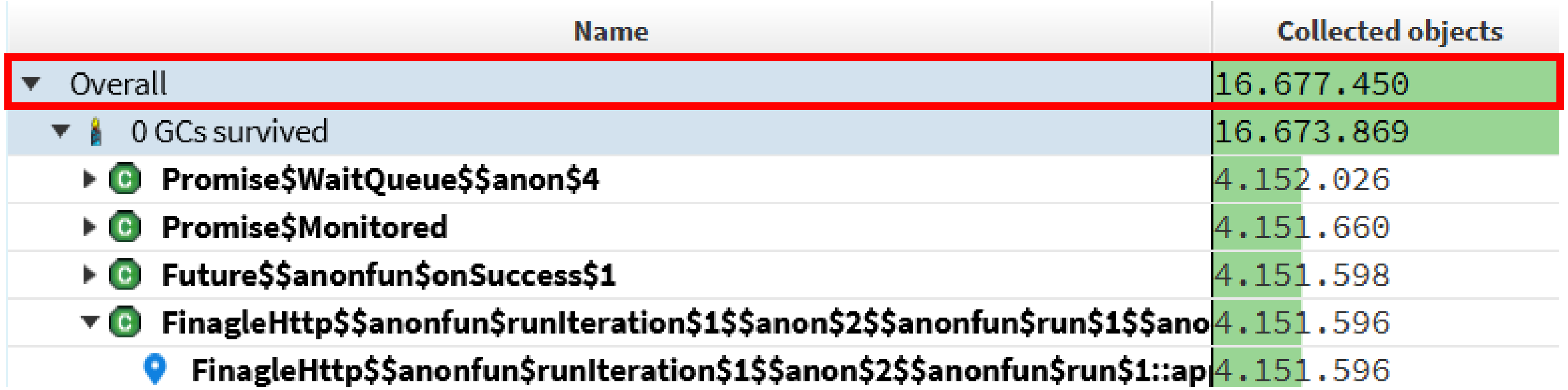


ANOTHER DEMO: FINAGLE-HTTP



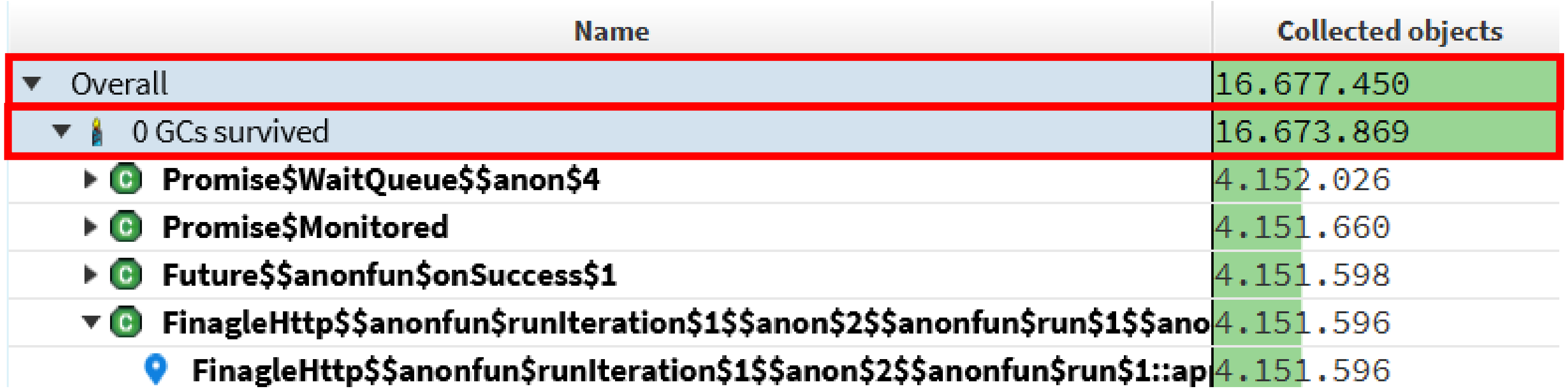
Name	Collected objects
Overall	16.677.450
0 GCs survived	16.673.869
Promise$WaitQueue$$anon$4	4.152.026
Promise$Monitored	4.151.660
Future$$anonfun$onSuccess$1	4.151.598
FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1$$ano	4.151.596
FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1::ap	4.151.596

ANOTHER DEMO: FINAGLE-HTTP



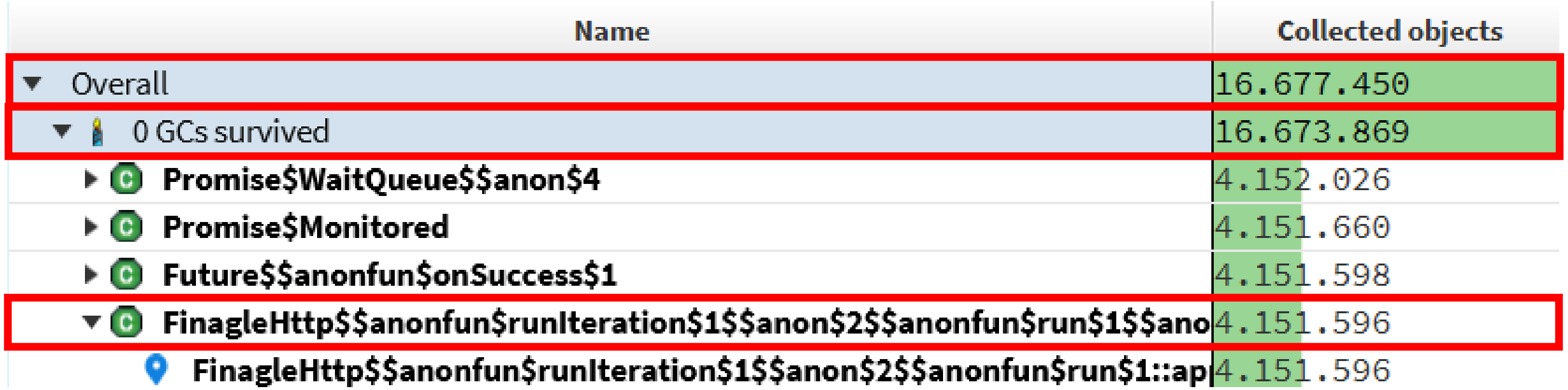
Name	Collected objects
▼ Overall	16.677.450
▼ 0 GCs survived	16.673.869
▶ Ⓒ Promise$WaitQueue$$anon$4	4.152.026
▶ Ⓒ Promise$Monitored	4.151.660
▶ Ⓒ Future$$anonfun$onSuccess$1	4.151.598
▼ Ⓒ FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1$$ano	4.151.596
FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1::ap	4.151.596

ANOTHER DEMO: FINAGLE-HTTP



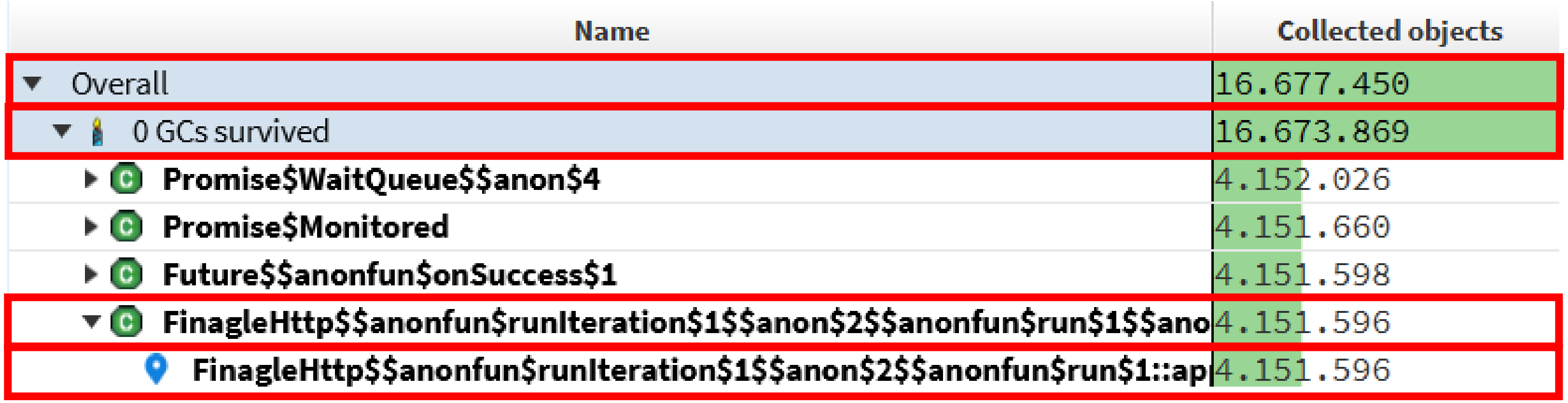
Name	Collected objects
▼ Overall	16.677.450
▼ 0 GCs survived	16.673.869
▸ Promise$WaitQueue$$anon$4	4.152.026
▸ Promise$Monitored	4.151.660
▸ Future$$anonfun$onSuccess$1	4.151.598
▼ FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1$$ano	4.151.596
FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1::ap	4.151.596

ANOTHER DEMO: FINAGLE-HTTP



Name	Collected objects
Overall	16.677.450
0 GCs survived	16.673.869
Promise$WaitQueue$$anon$4	4.152.026
Promise$Monitored	4.151.660
Future$$anonfun$onSuccess$1	4.151.598
FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1$$ano	4.151.596
FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1::ap	4.151.596

ANOTHER DEMO: FINAGLE-HTTP



Name	Collected objects
▼ Overall	16.677.450
▼ 0 GCs survived	16.673.869
▶ Promise$WaitQueue$$anon$4	4.152.026
▶ Promise$Monitored	4.151.660
▶ Future$$anonfun$onSuccess$1	4.151.598
▼ FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1$$ano	4.151.596
FinagleHttp$$anonfun$runIteration$1$$anon$2$$anonfunrun1::ap	4.151.596

DEMO: FINAGLE-HTTP

```
val response: Future[http.Response] = client(request)
```

DEMO: FINAGLE-HTTP

```
val response: Future[http.Response] = client(request)
```

```
for (i <- 0 until NUM_REQUESTS) {
    Await.result(response.onSuccess { rep: http.Response =>
        totalLength += rep.content.length
    })
}
```

DEMO: FINAGLE-HTTP

```
val response: Future[http.Response] = client(request)
```

```
for (i <- 0 until NUM_REQUESTS) {
    Await.result(response.onSuccess { rep: http.Response =>
        totalLength += rep.content.length
    })
}
```

DEMO: FINAGLE-HTTP

```
val response: Future[http.Response] = client(request)
```

```
for (i <- 0 until NUM_REQUESTS) {
    Await.result(response.onSuccess { rep: http.Response =>
        totalLength += rep.content.length
    })
}
```

```
val h = { rep: http.Response =>
    totalLength += rep.content.length
}
for (i <- 0 until NUM_REQUESTS {
    Await.result(response.onSuccess(h))
}
```

DEMO: FINAGLE-HTTP

```
val response: Future[http.Response] = client(request)
```

```
for (i <- 0 until NUM_REQUESTS) {
    Await.result(response.onSuccess { rep: http.Response =>
        totalLength += rep.content.length
    })
}
```

```
val h = { rep: http.Response =>
    totalLength += rep.content.length
}
for (i <- 0 until NUM_REQUESTS {
    Await.result(response.onSuccess(h))
}
```

DEMO: FINAGLE-HTTP

```
val response: Future[http.Response] = client(request)
```

```
for (i <- 0 until NUM_REQUESTS) {
    Await.result(response.onSuccess { rep: http.Response =>
        totalLength += rep.content.length
    })
}
```

```
val h = { rep: http.Response =>
    totalLength += rep.content.length
}
for (i <- 0 until NUM_REQUESTS {
    Await.result(response.onSuccess(h))
}
```

DEMO: FINAGLE-HTTP

```
val response: Future[http.Response] = client(request)
```

```
for (i <- 0 until NUM_REQUESTS) {
    Await.result(response.onSuccess { rep: http.Response =>
        totalLength += rep.content.length
    })
}
```

```
val h = { rep: http.Response =>
    totalLength += rep.content.length
}
for (i <- 0 until NUM_REQUESTS {
    Await.result(response.onSuccess(h))
}
```

DEMO: FINAGLE-HTTP

```
val response: Future[http.Response] = client(request)
```

```
for (i <- 0 until NUM_REQUESTS) {
    Await.result(response.onSuccess { rep: http.Response =>
        totalLength += rep.content.length
    })
}
```

```
val h = { rep: http.Response =>
    totalLength += rep.content.length
}
for (i <- 0 until NUM_REQUESTS {
    Await.result(response.onSuccess(h))
}
```

TAKE-AWAYS

TAKE-AWAYS

Memory Analysis

Top-down
Bottom-up

Heap dumps

Trace-based

... and 100 other things

TAKE-AWAYS

Memory Analysis	Problem
Top-down Bottom-up Heap dumps Trace-based ... and 100 other things	New users need guidance to get started faster It is not enough to „throw tools at them“

TAKE-AWAYS

Memory Analysis	Problem	Guided Exploration
Top-down Bottom-up Heap dumps Trace-based ... and 100 other things	New users need guidance to get started faster It is not enough to „throw tools at them“	Detection Highlighting Explanation Suggestion

TAKE-AWAYS

Memory Analysis	Problem	Guided Exploration
Top-down Bottom-up Heap dumps Trace-based ... and 100 other things	New users need guidance to get started faster It is not enough to „throw tools at them“	Detection Highlighting Explanation Suggestion

Markus Weninger
Johannes Kepler University
Linz, Austria
markus.weninger@jku.at

http://mevss.jku.at/AntTracks
http://bit.ly/weninger_ssw

Additional Notes

- Some icons made by Freepik & Smashicons from https://www.flaticon.com
- Some photos made by bruce mars and Isaque Pereira from Pexels