

# HEAP EVOLUTION ANALYSIS USING TREE VISUALIZATIONS



**SSP 2020**

**Markus Weninger**, Lukas Makor, Hanspeter Mössenböck

*Johannes Kepler University Linz, Austria*

*Institute for System Software, Christian Doppler Laboratory MEVSS*

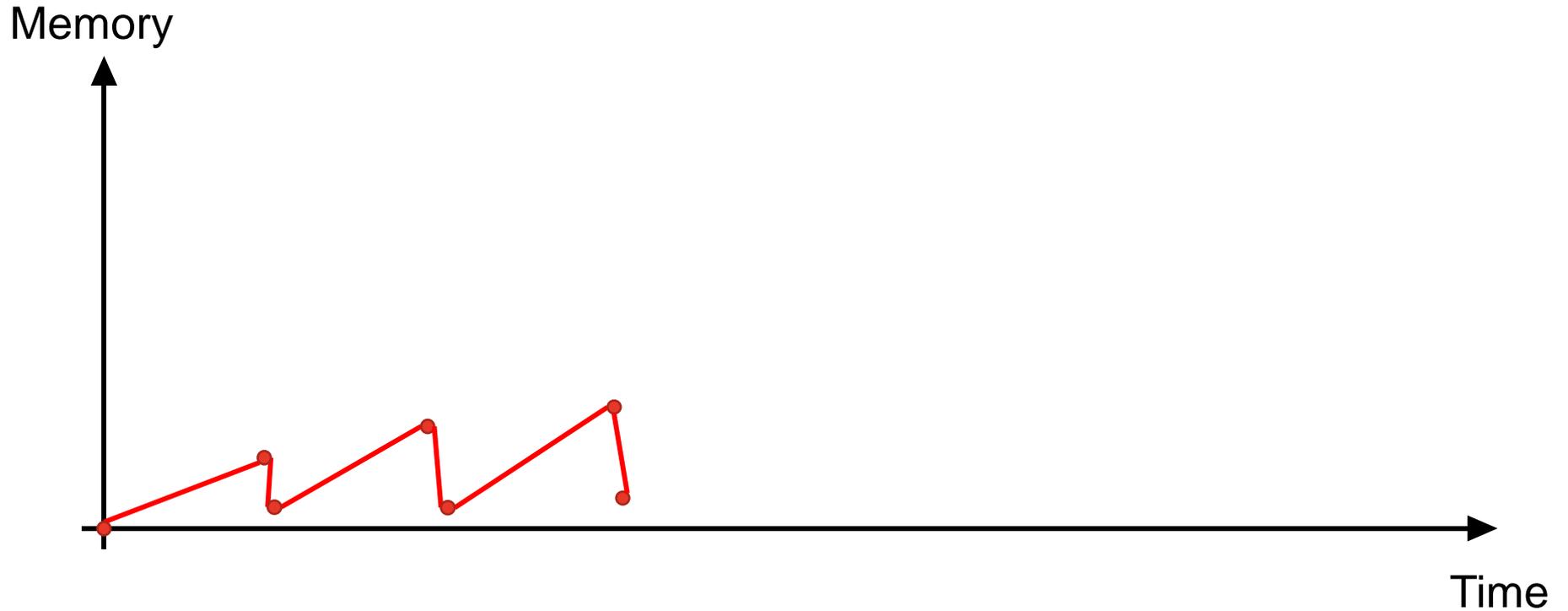


# MOTIVATION: MEMORY ANOMALIES

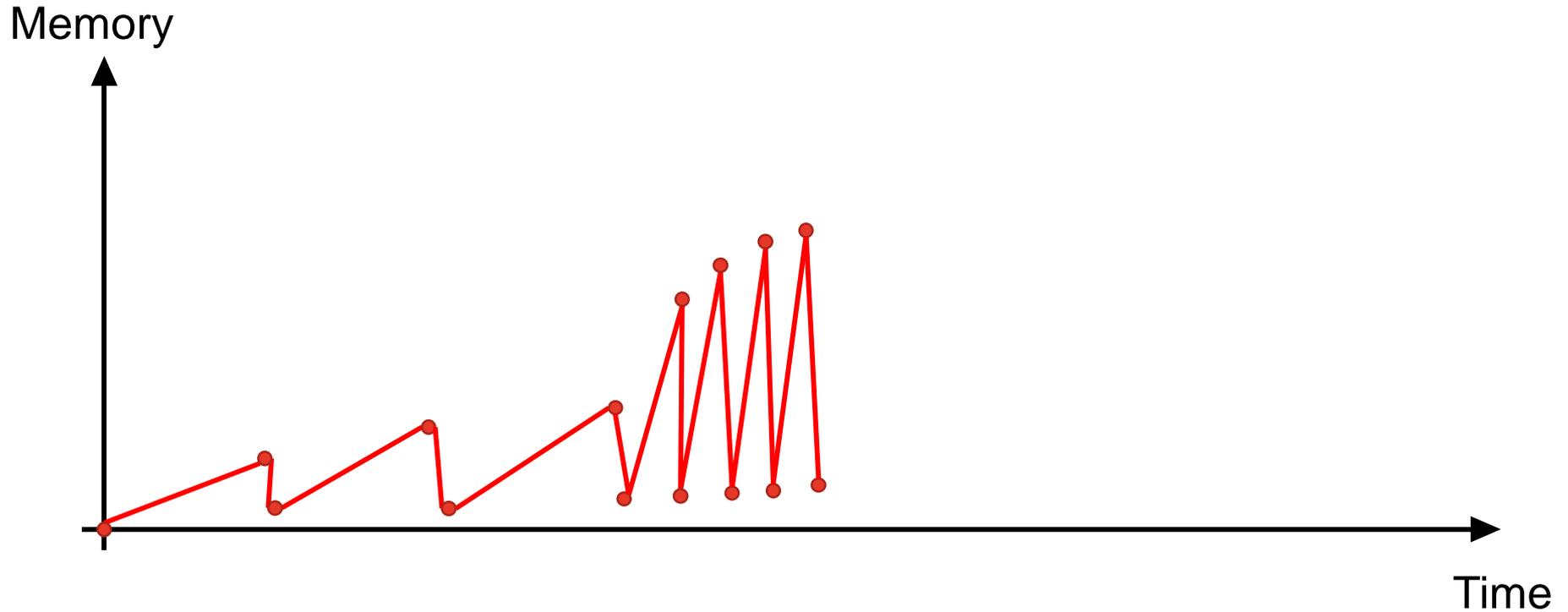
Memory



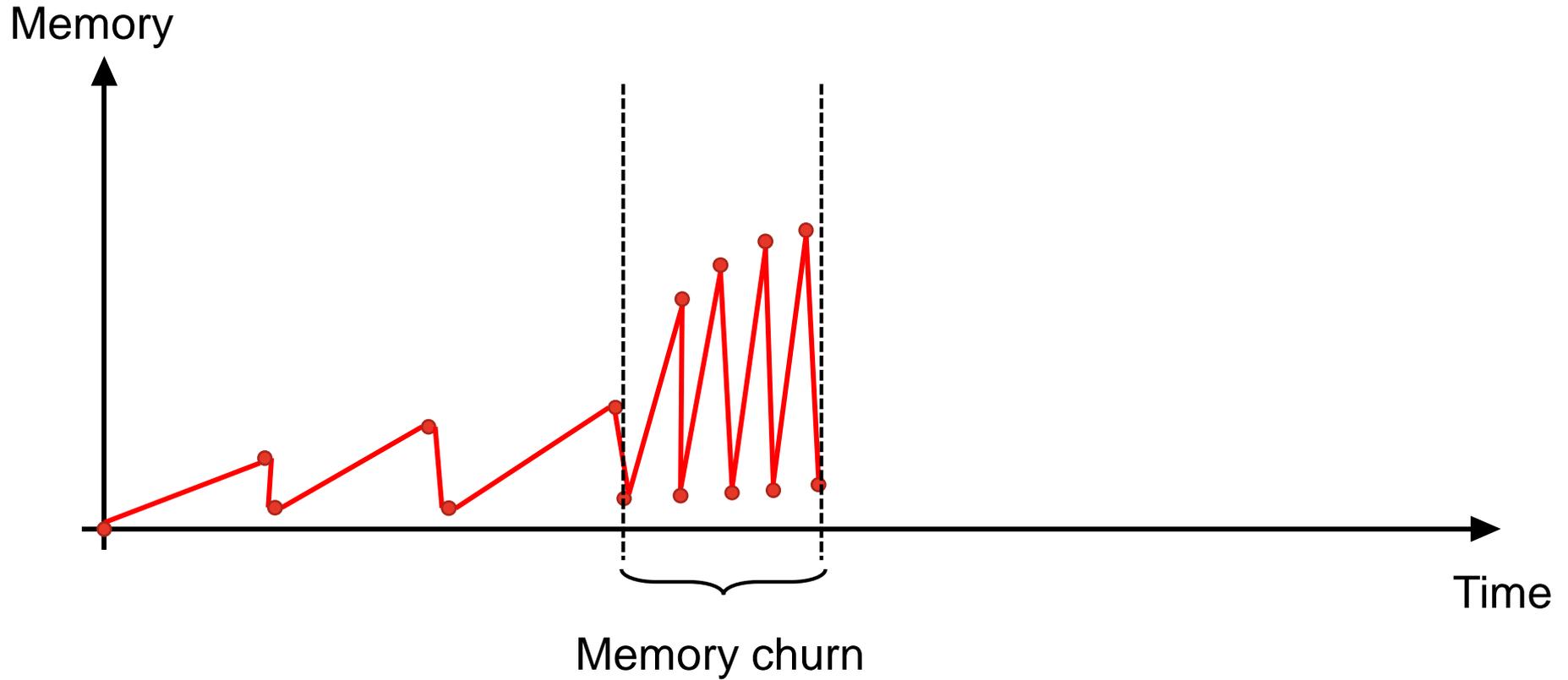
# MOTIVATION: MEMORY ANOMALIES



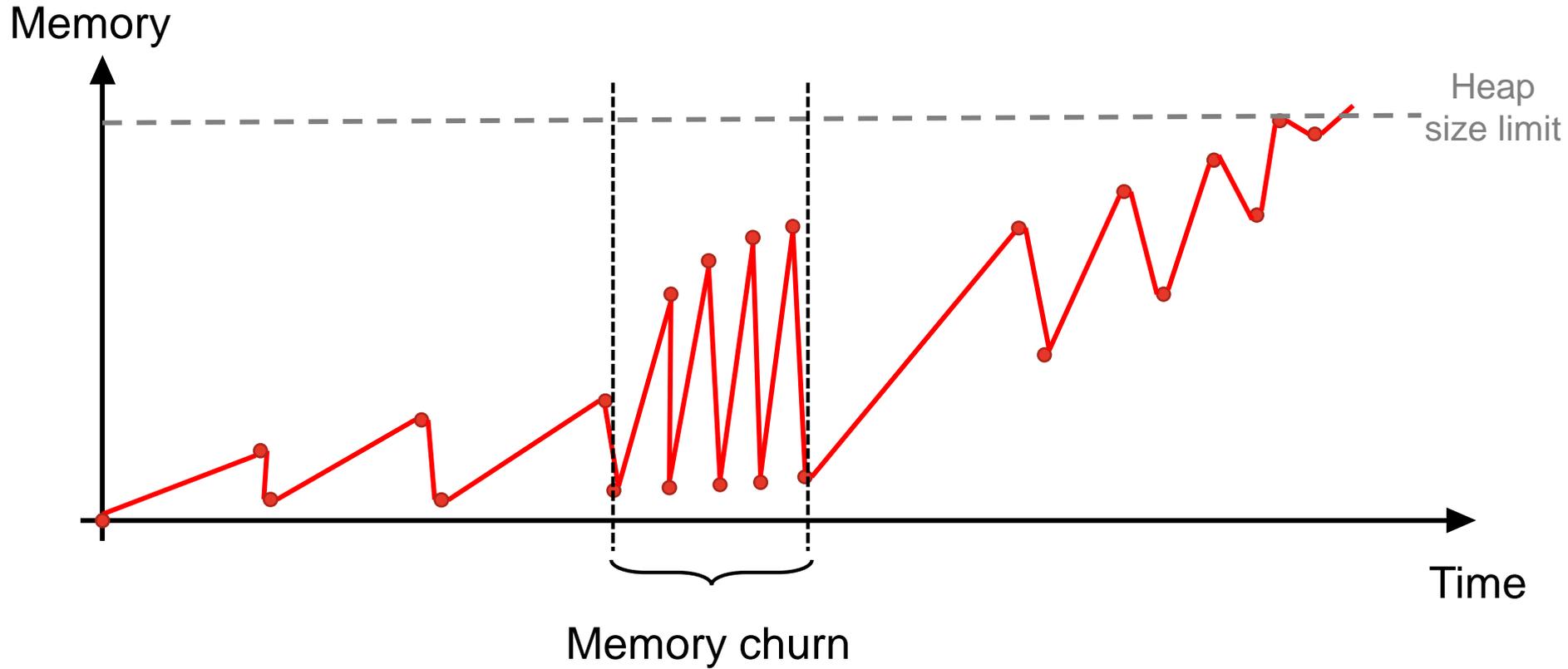
# MOTIVATION: MEMORY ANOMALIES



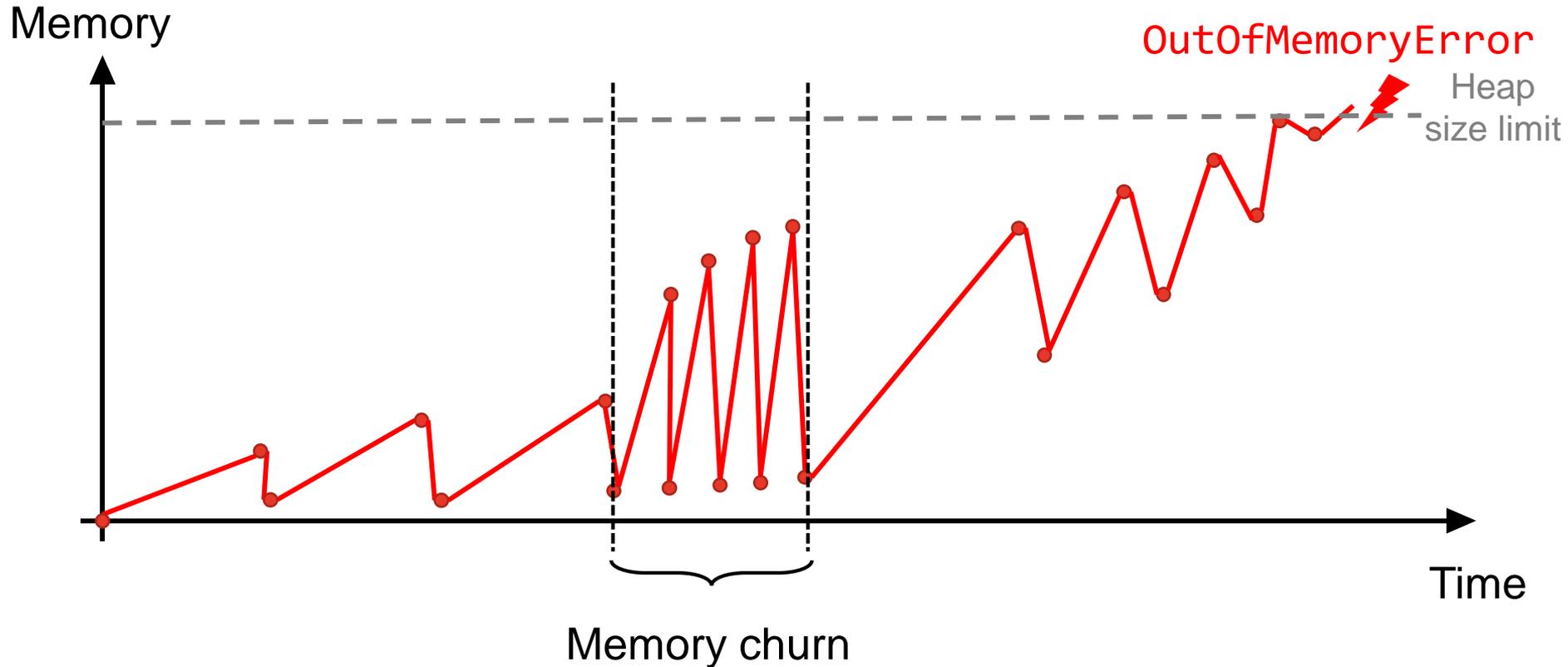
# MOTIVATION: MEMORY ANOMALIES



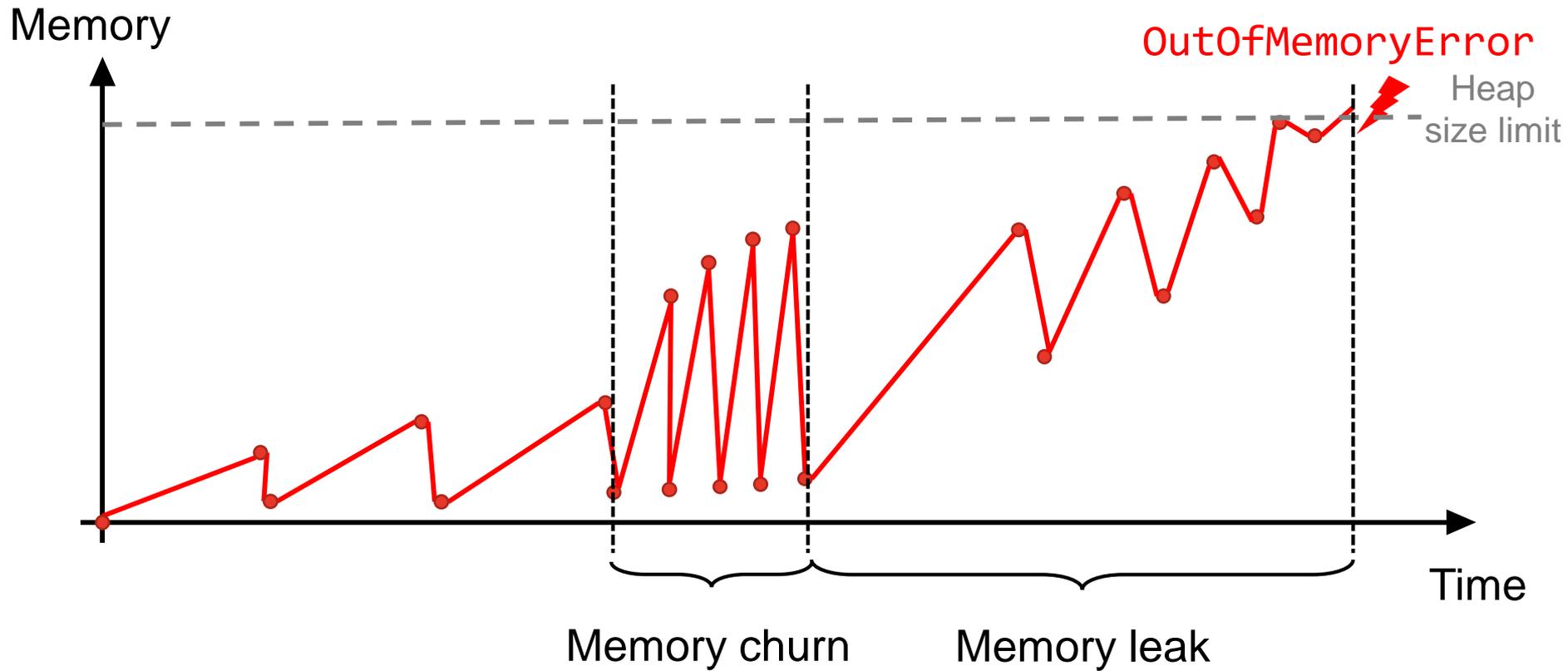
# MOTIVATION: MEMORY ANOMALIES



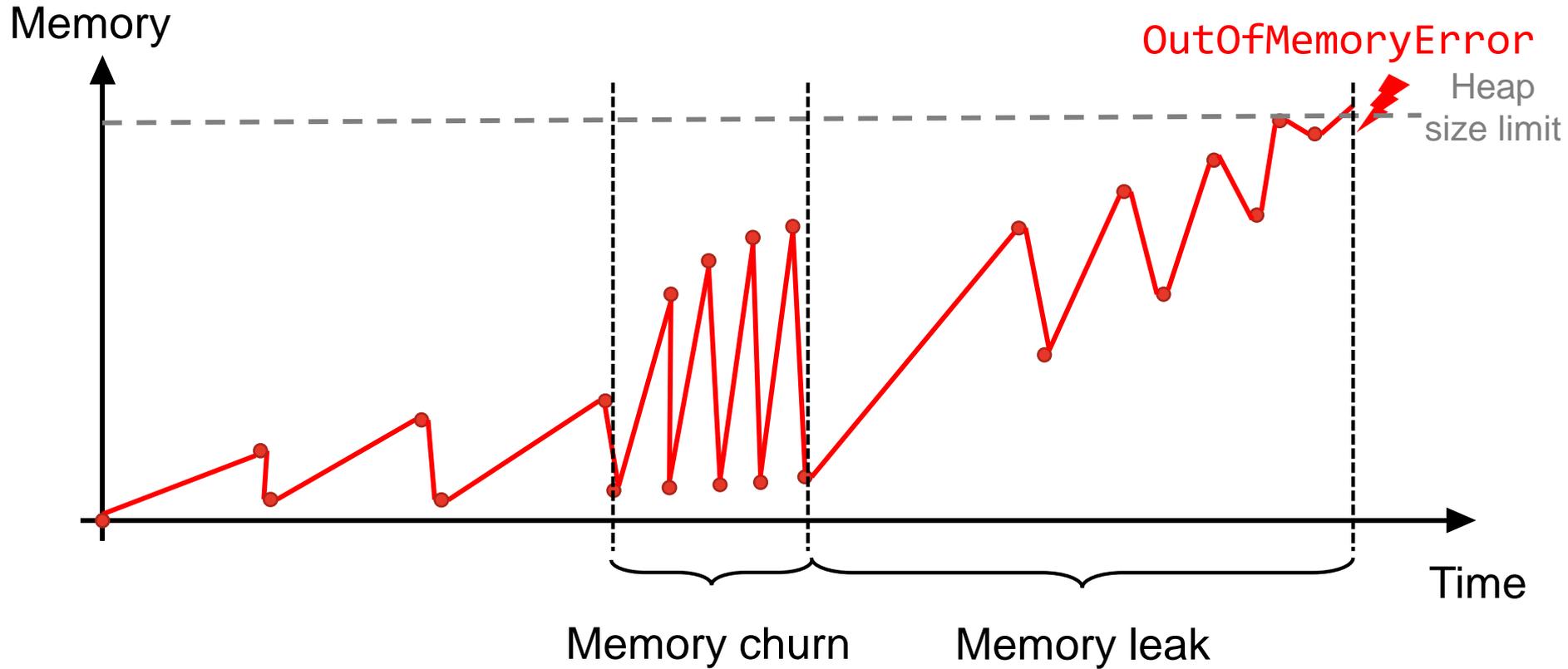
# MOTIVATION: MEMORY ANOMALIES



# MOTIVATION: MEMORY ANOMALIES

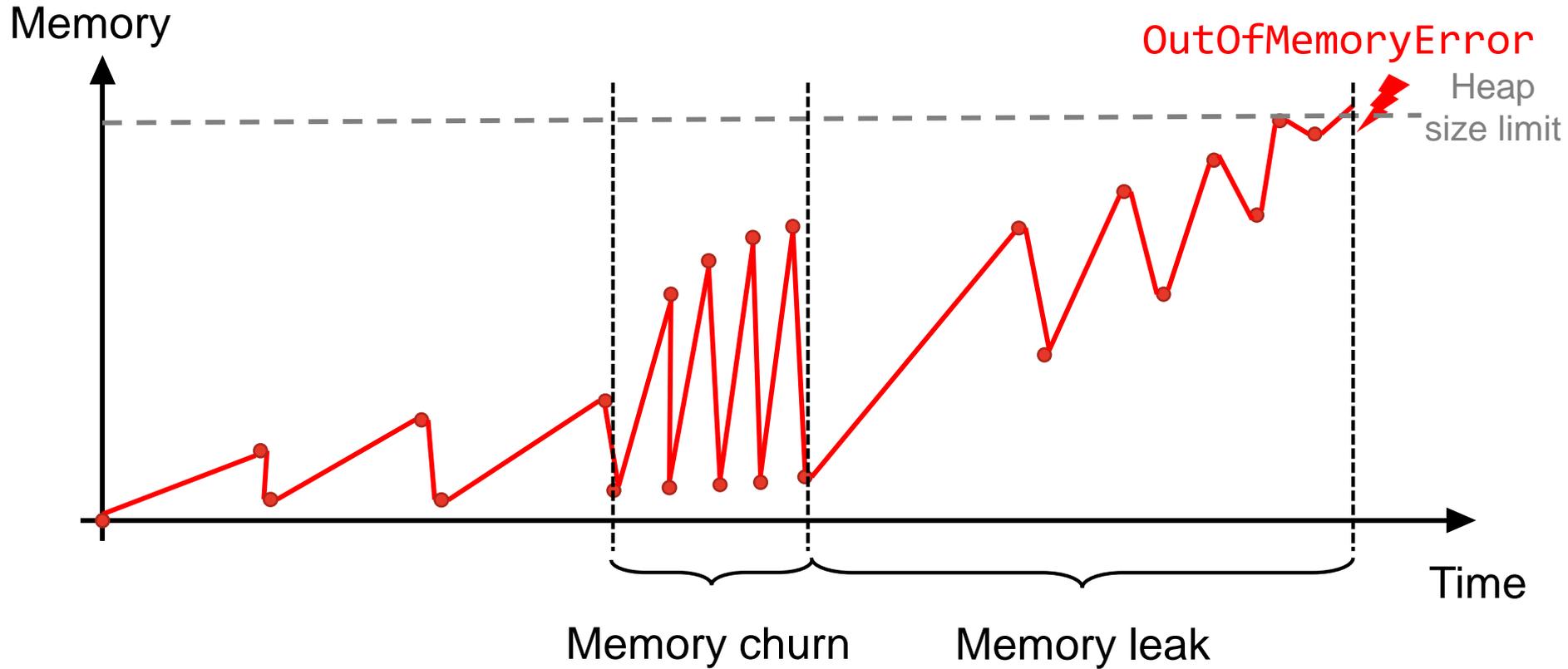


# MOTIVATION: MEMORY ANOMALIES



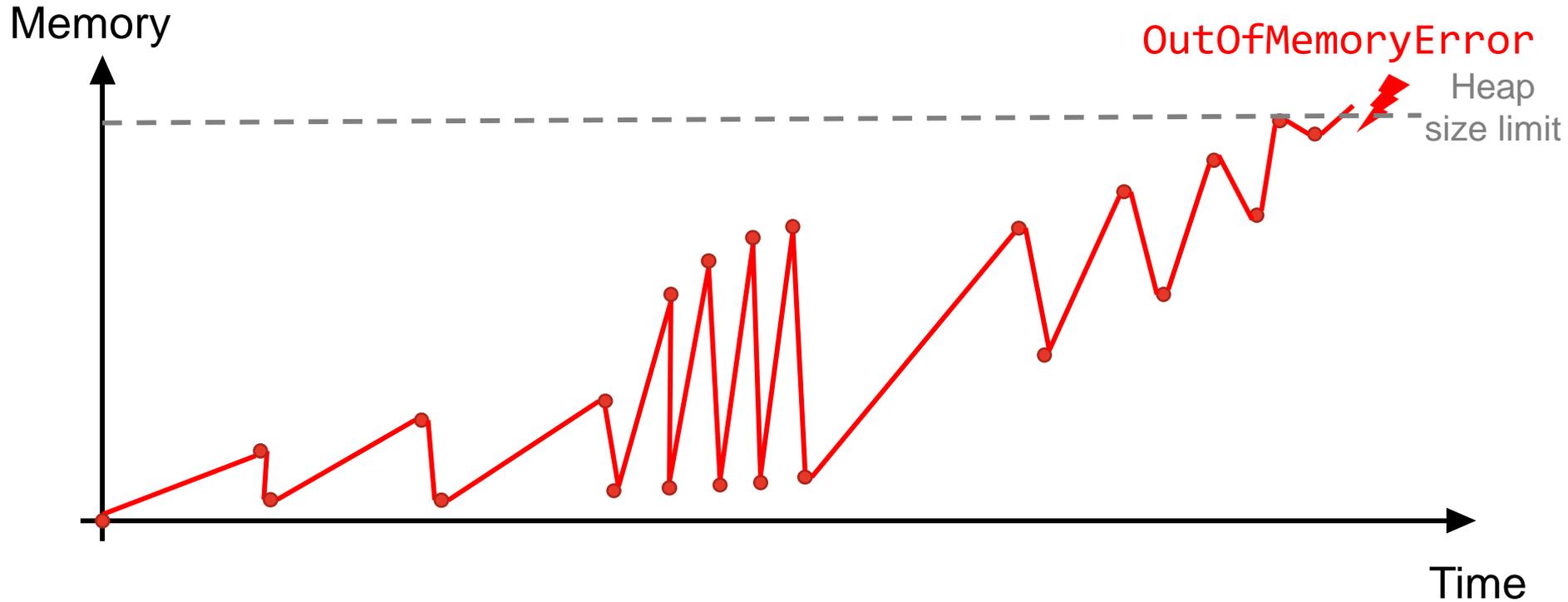
**Investigate!**

# MOTIVATION: MEMORY ANOMALIES

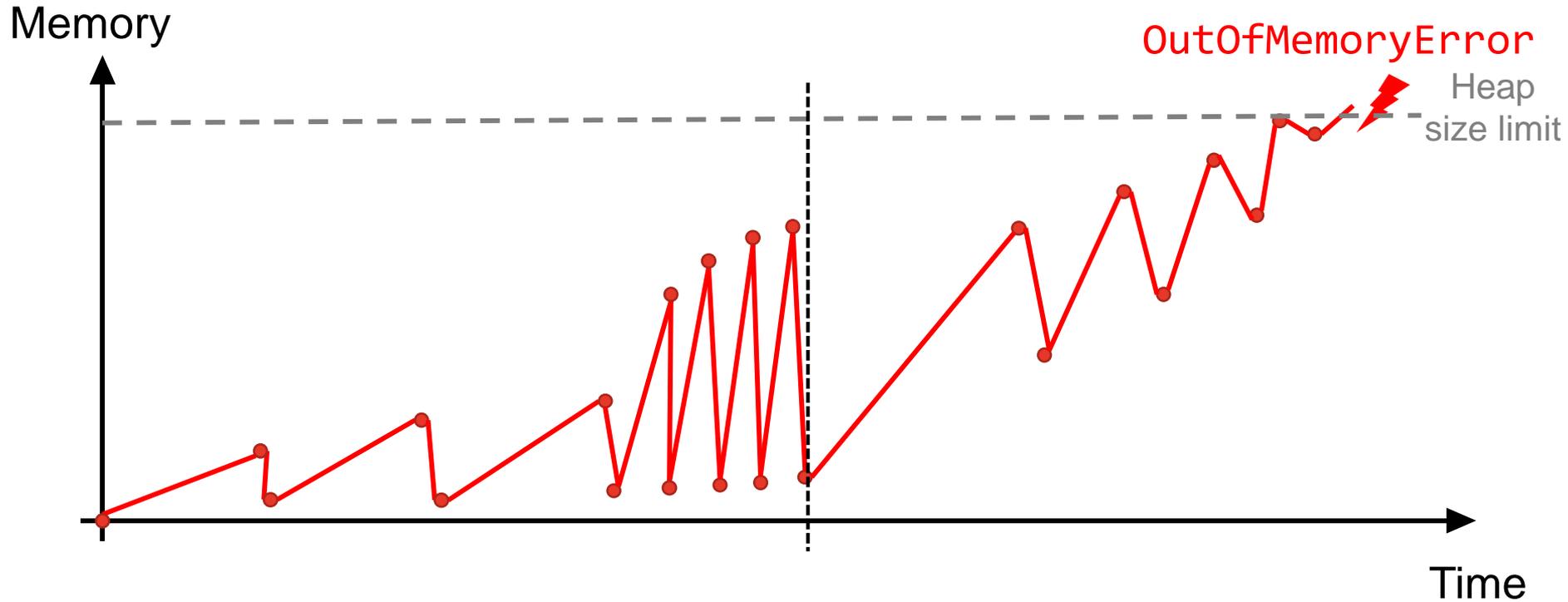


**How?**

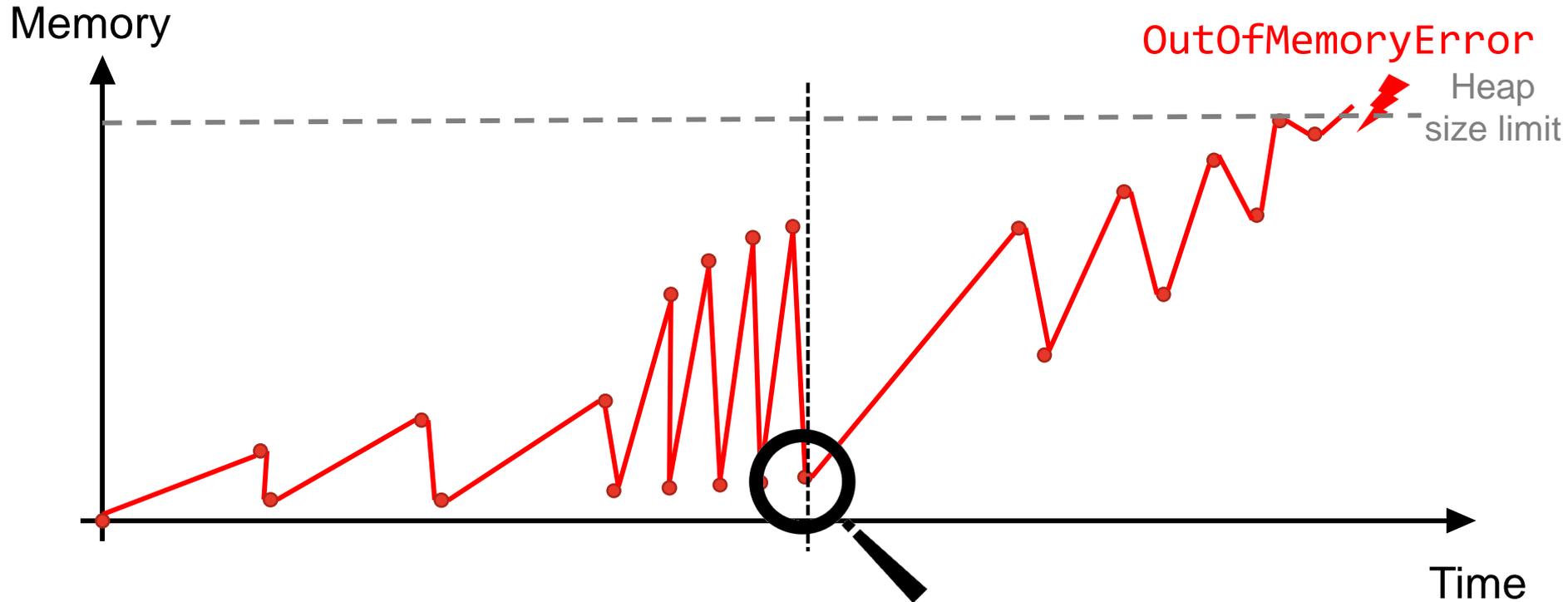
# MOTIVATION: MEMORY ANOMALIES



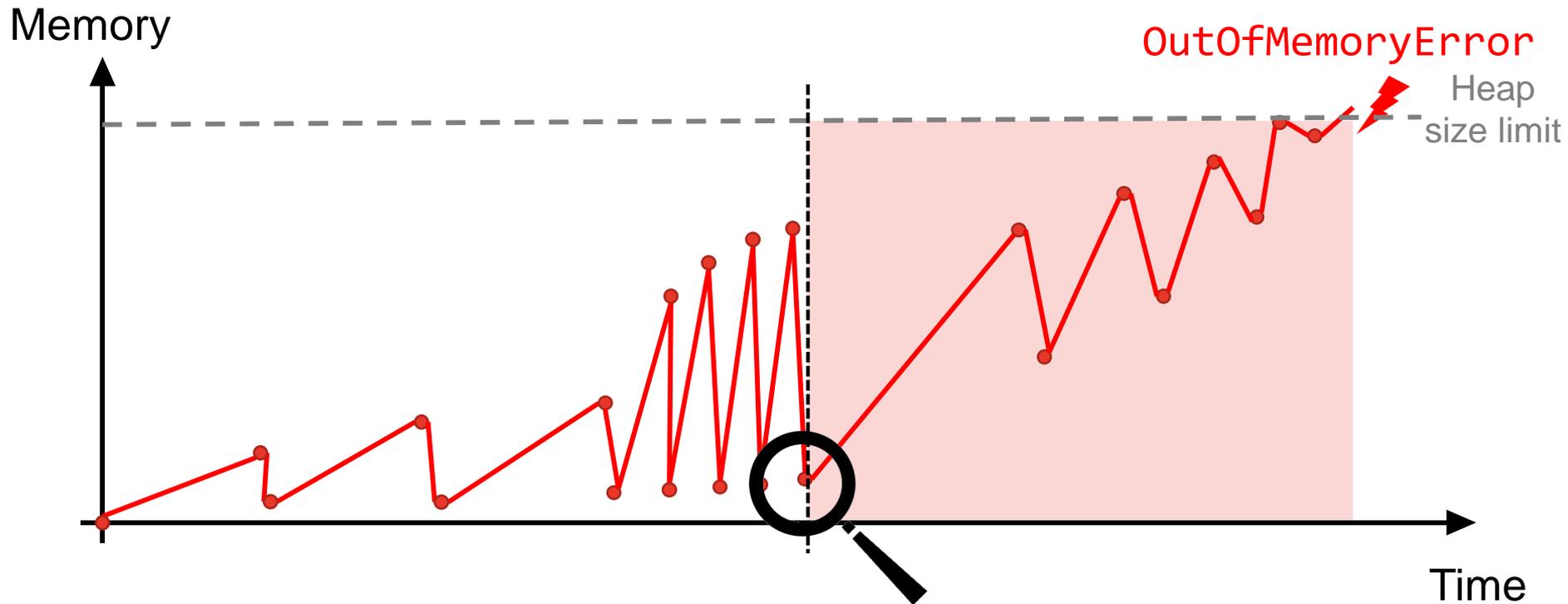
# MOTIVATION: MEMORY ANOMALIES



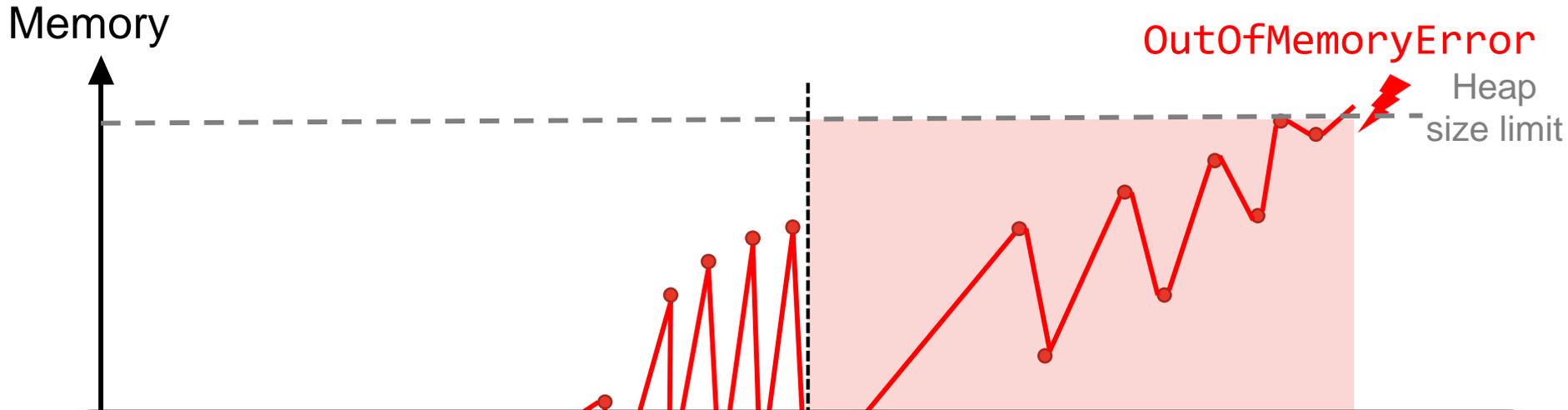
# MOTIVATION: MEMORY ANOMALIES



# MOTIVATION: MEMORY ANOMALIES

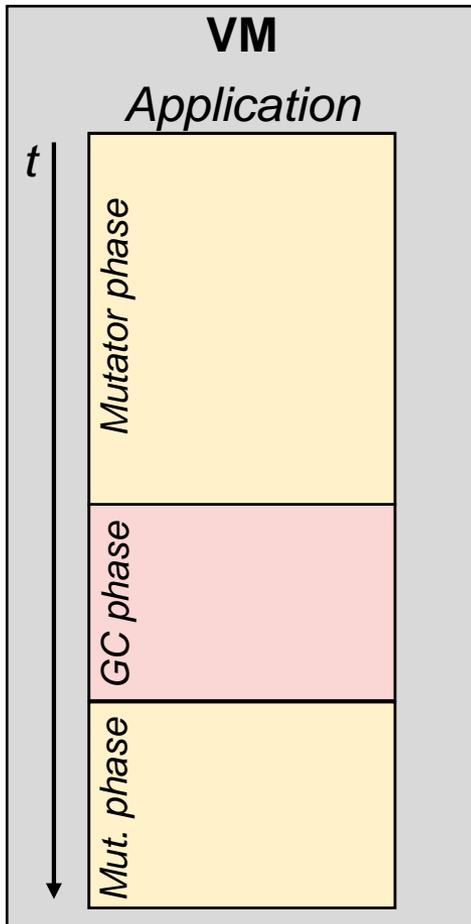


# MOTIVATION: MEMORY ANOMALIES



**Idea:** Help developers in *understanding* memory *growth* over time through the means of *visualization*

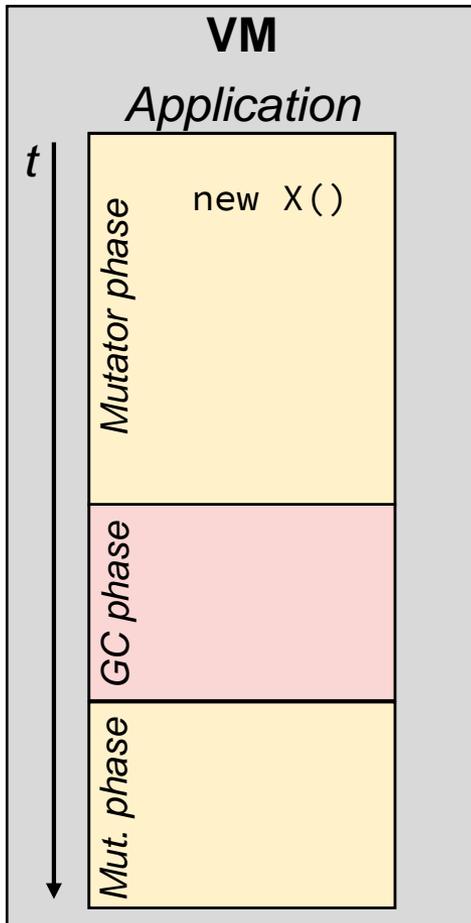
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

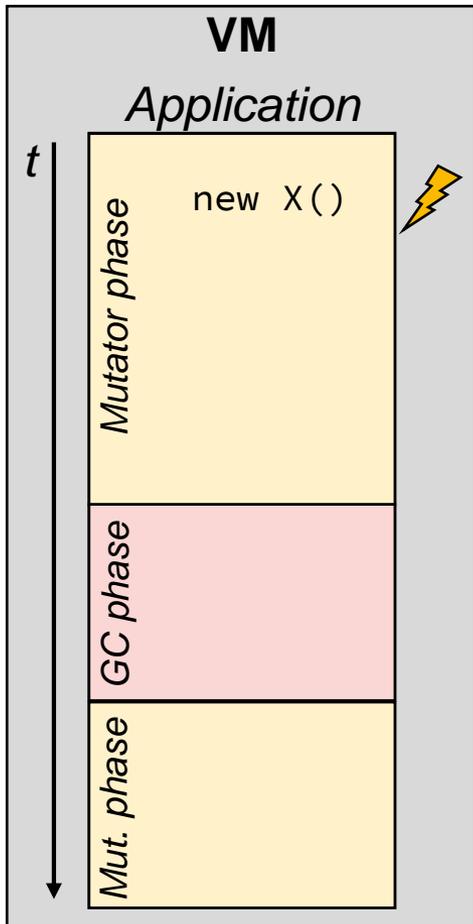
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

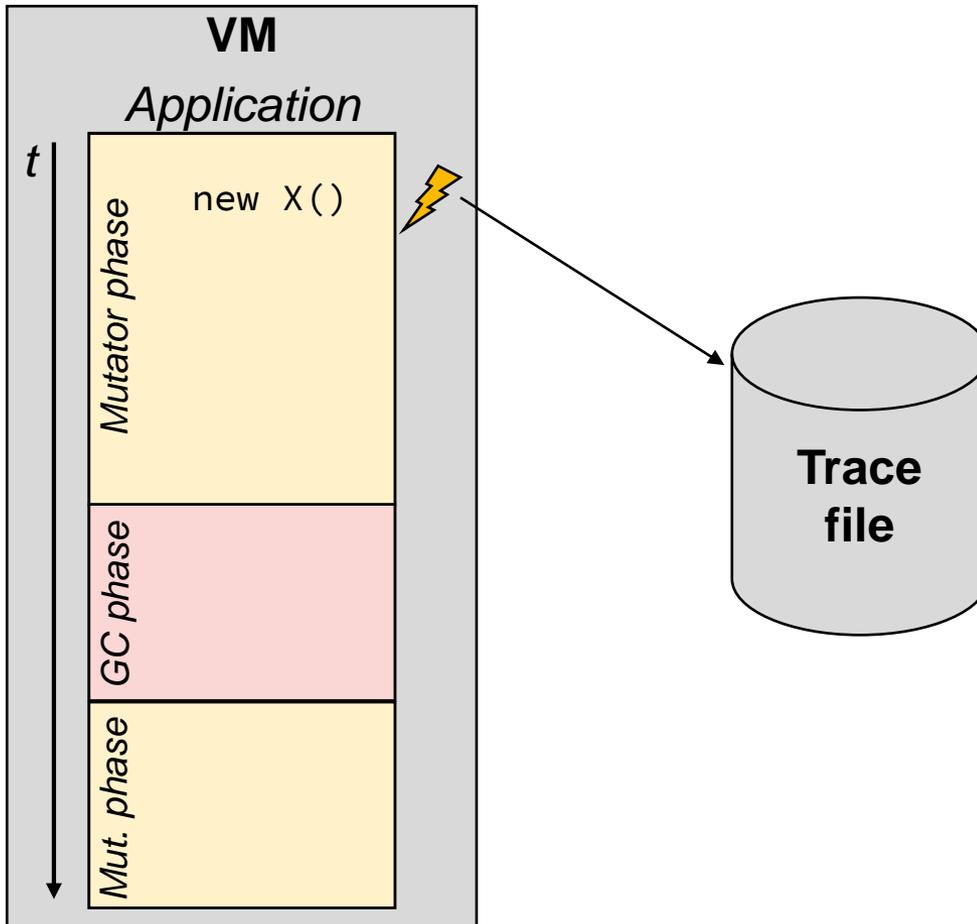
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

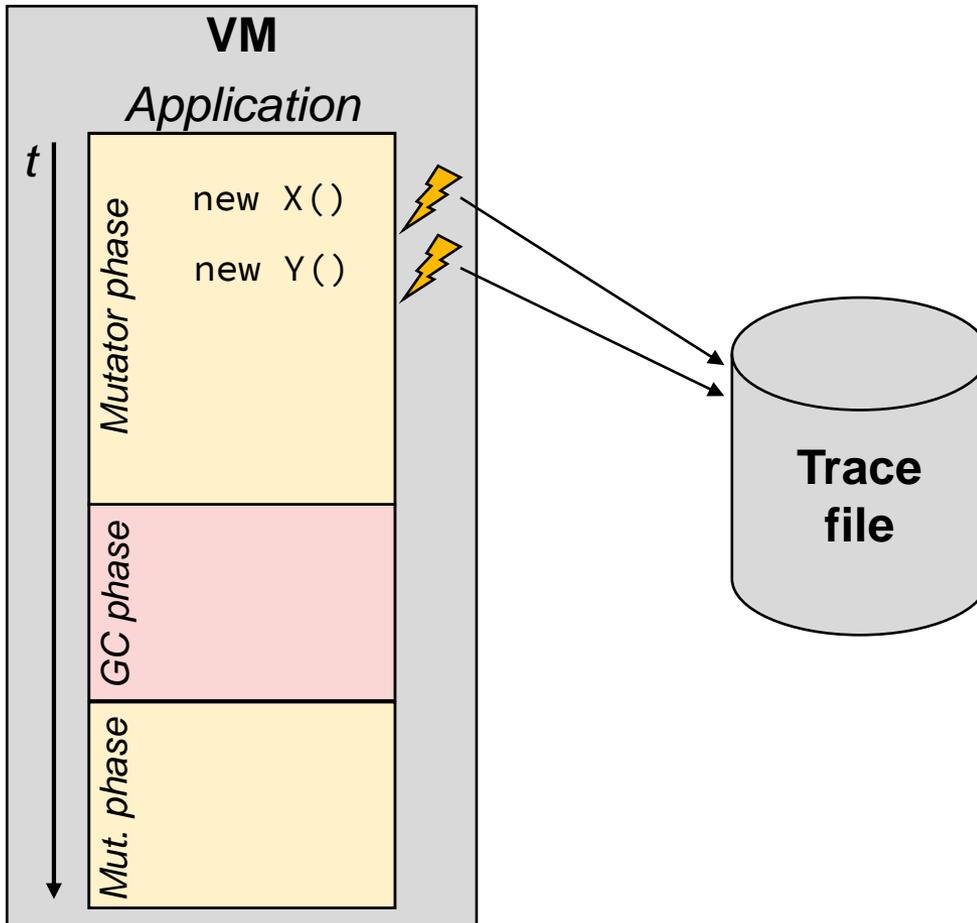
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

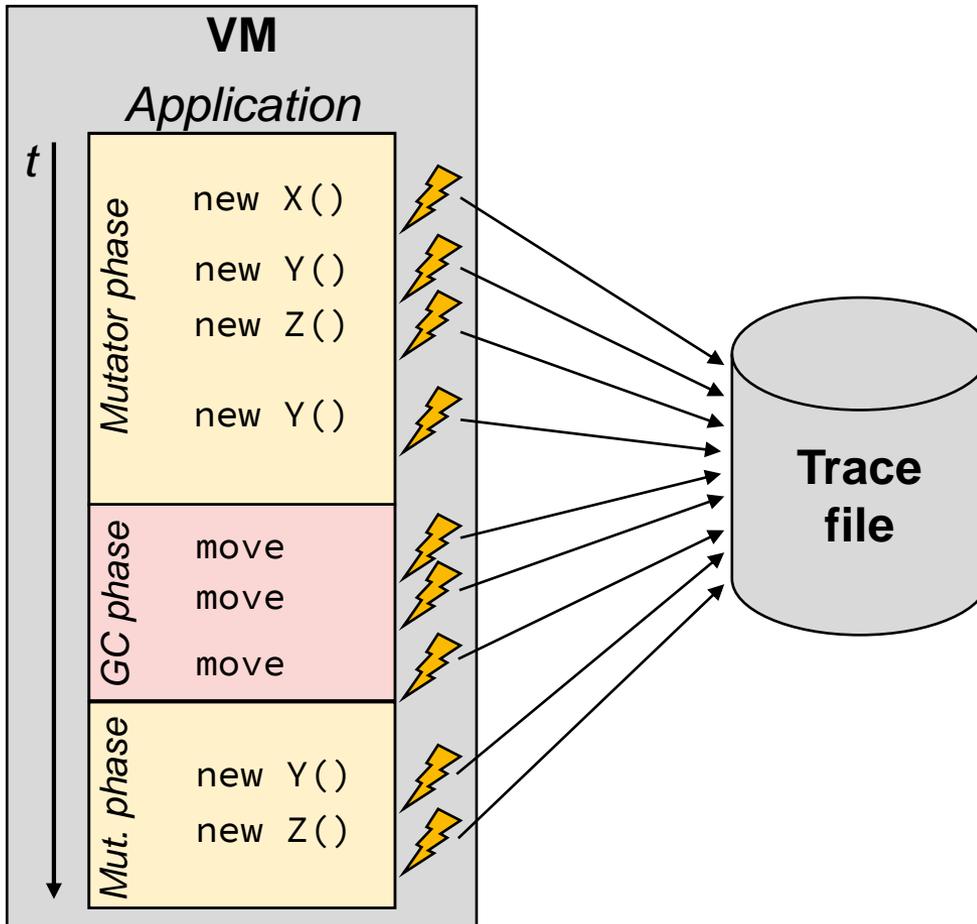
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

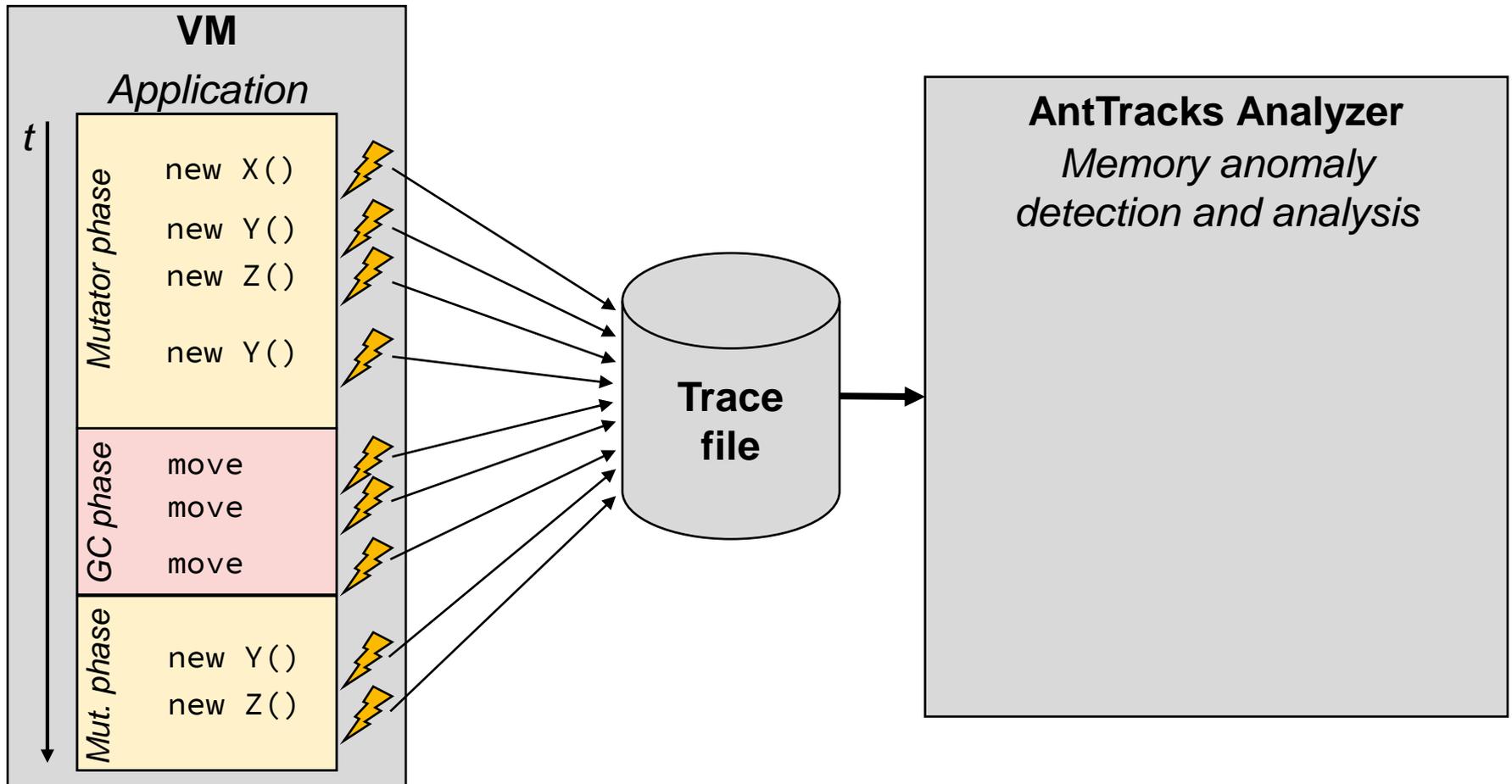
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

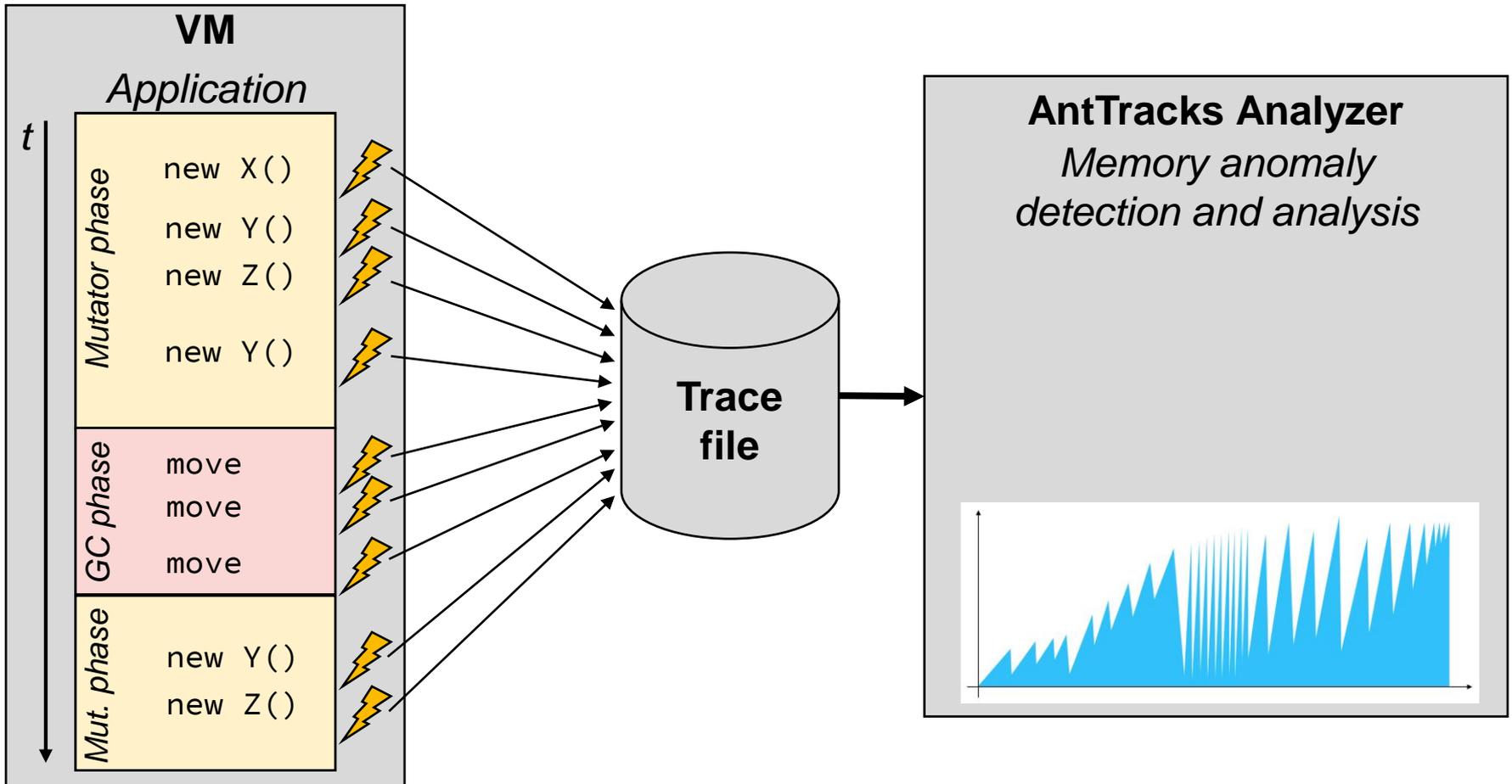
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

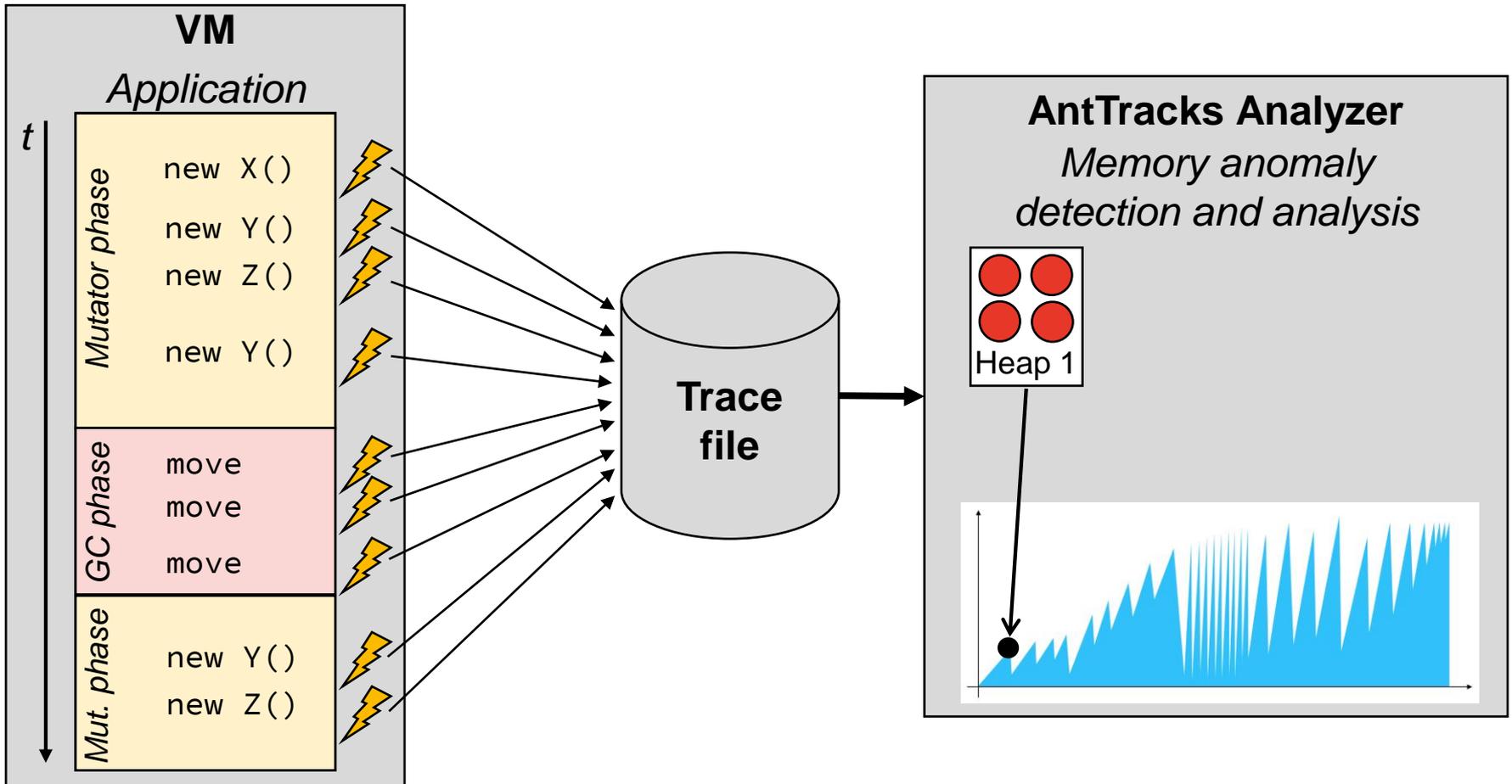
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

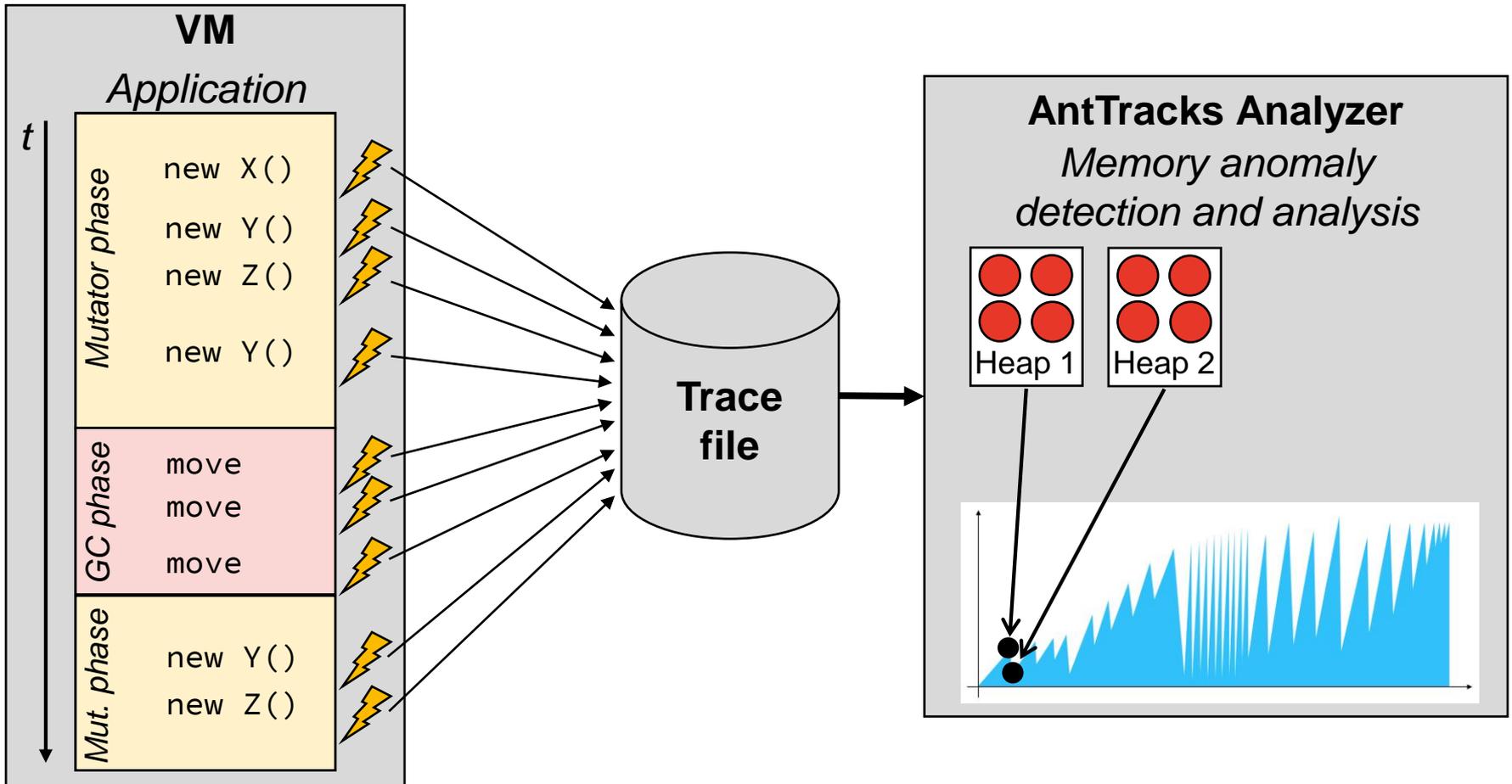
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

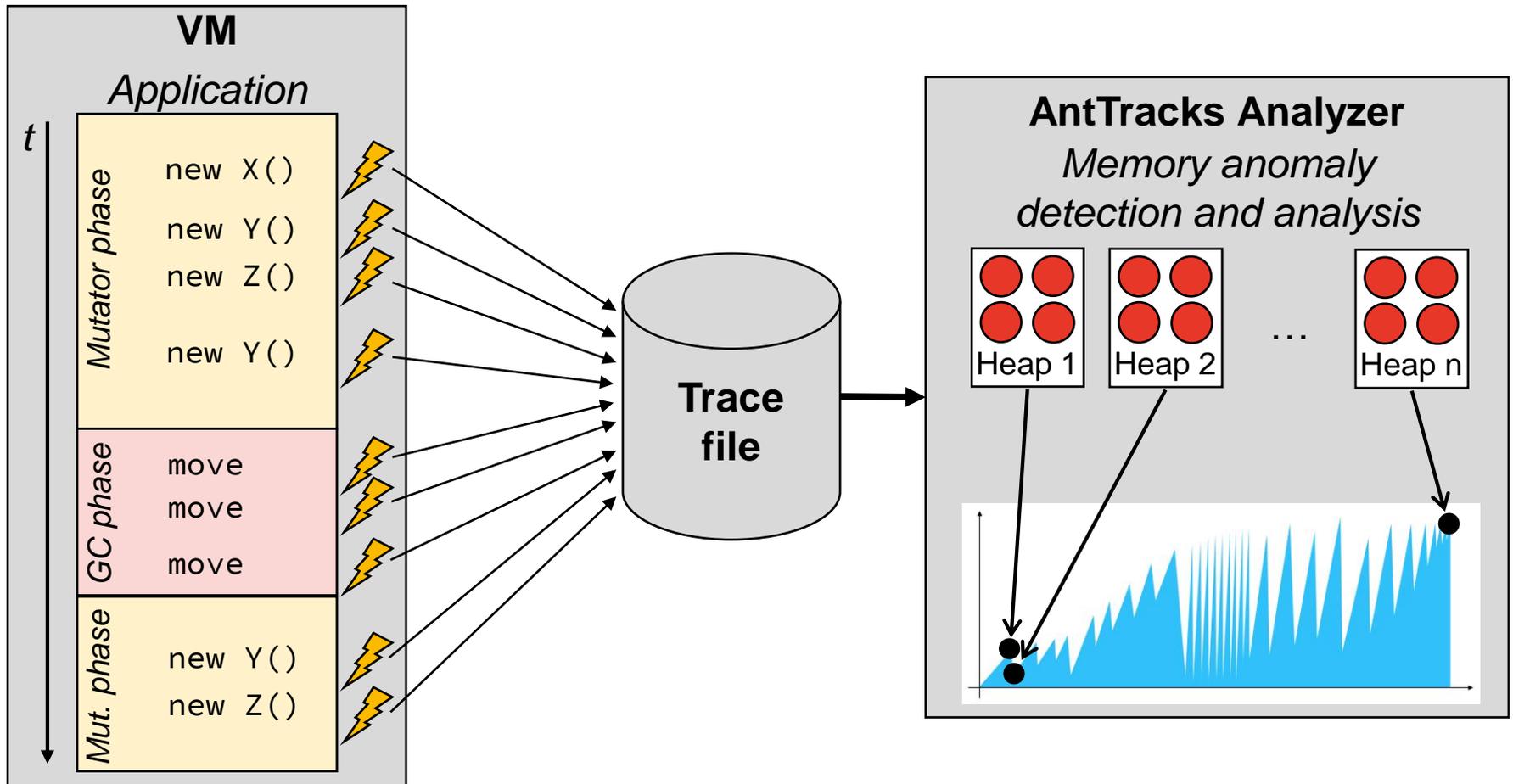
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

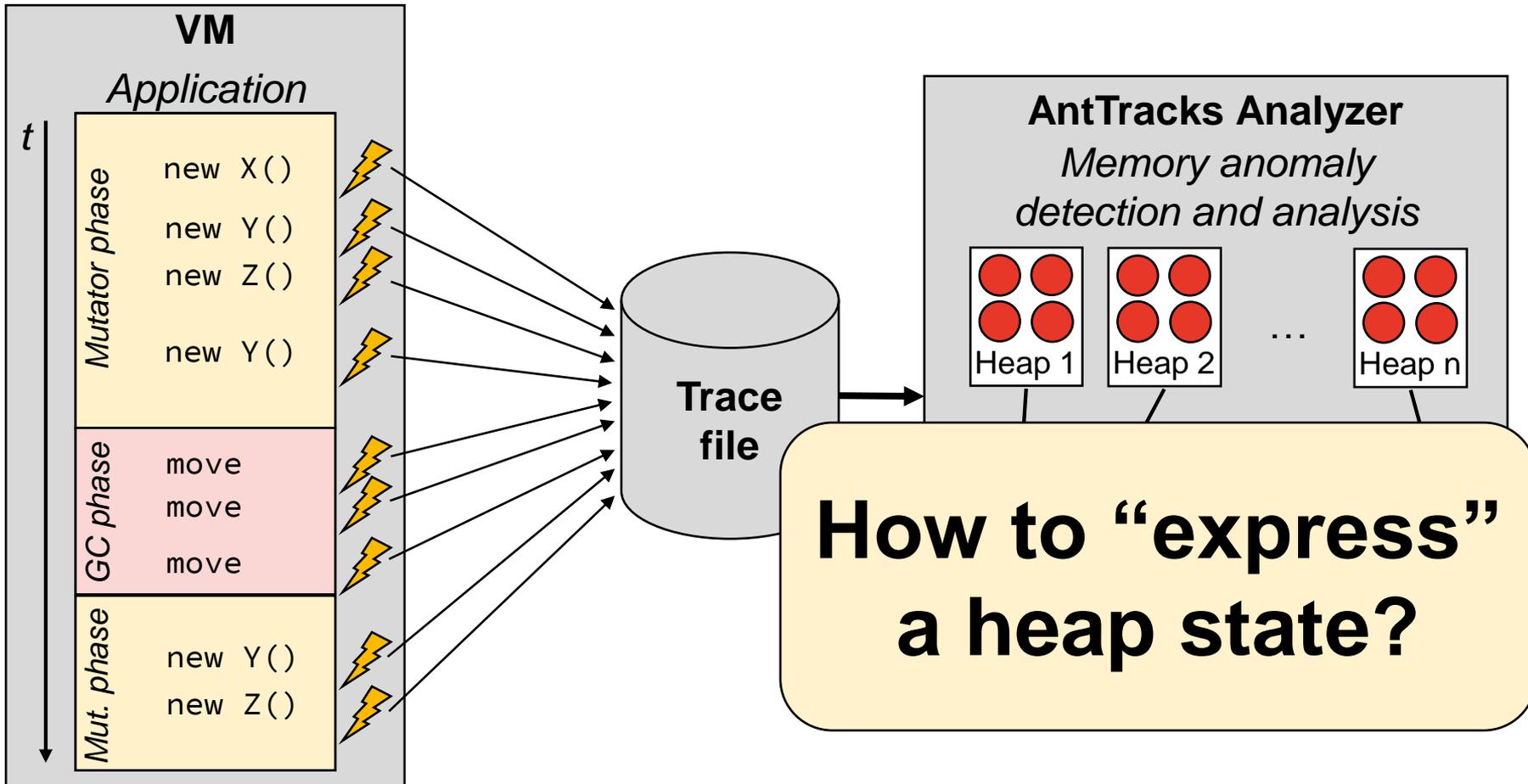
# DATA: MEMORY TRACES



Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

# DATA: MEMORY TRACES

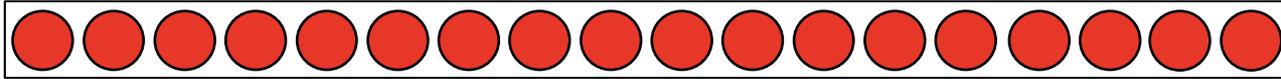


Lengauer, Bitto, Mössenböck: *Accurate and Efficient Object Tracing for Java Applications*, ICPE 2015

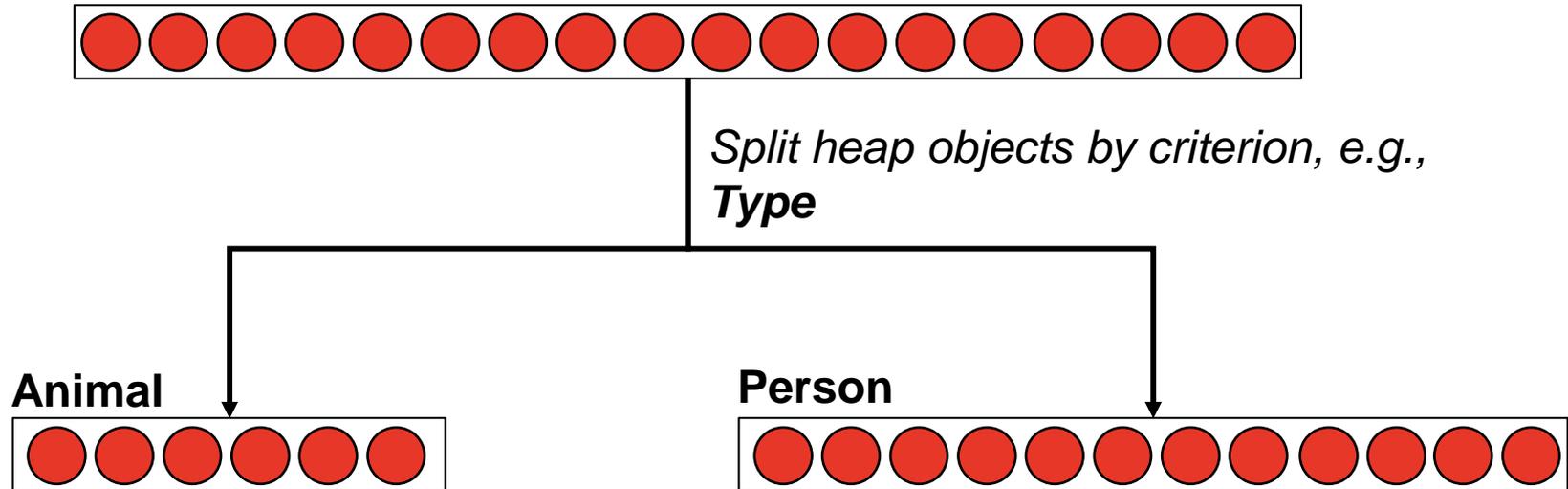
Lengauer et al.: *Efficient Memory Traces with Full Pointer Information*, PPPJ 2016

# OBJECT CLASSIFICATION

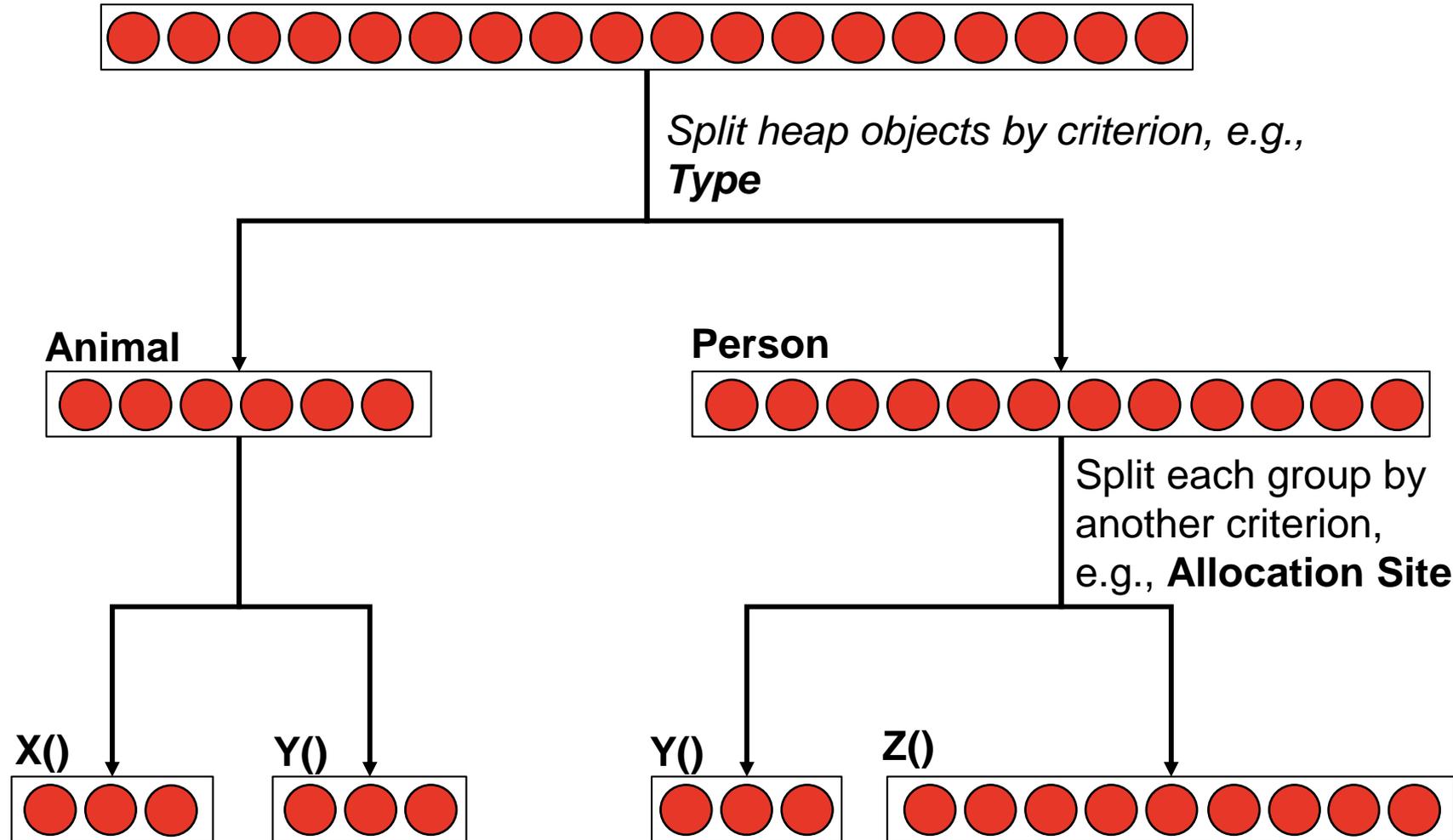
# OBJECT CLASSIFICATION



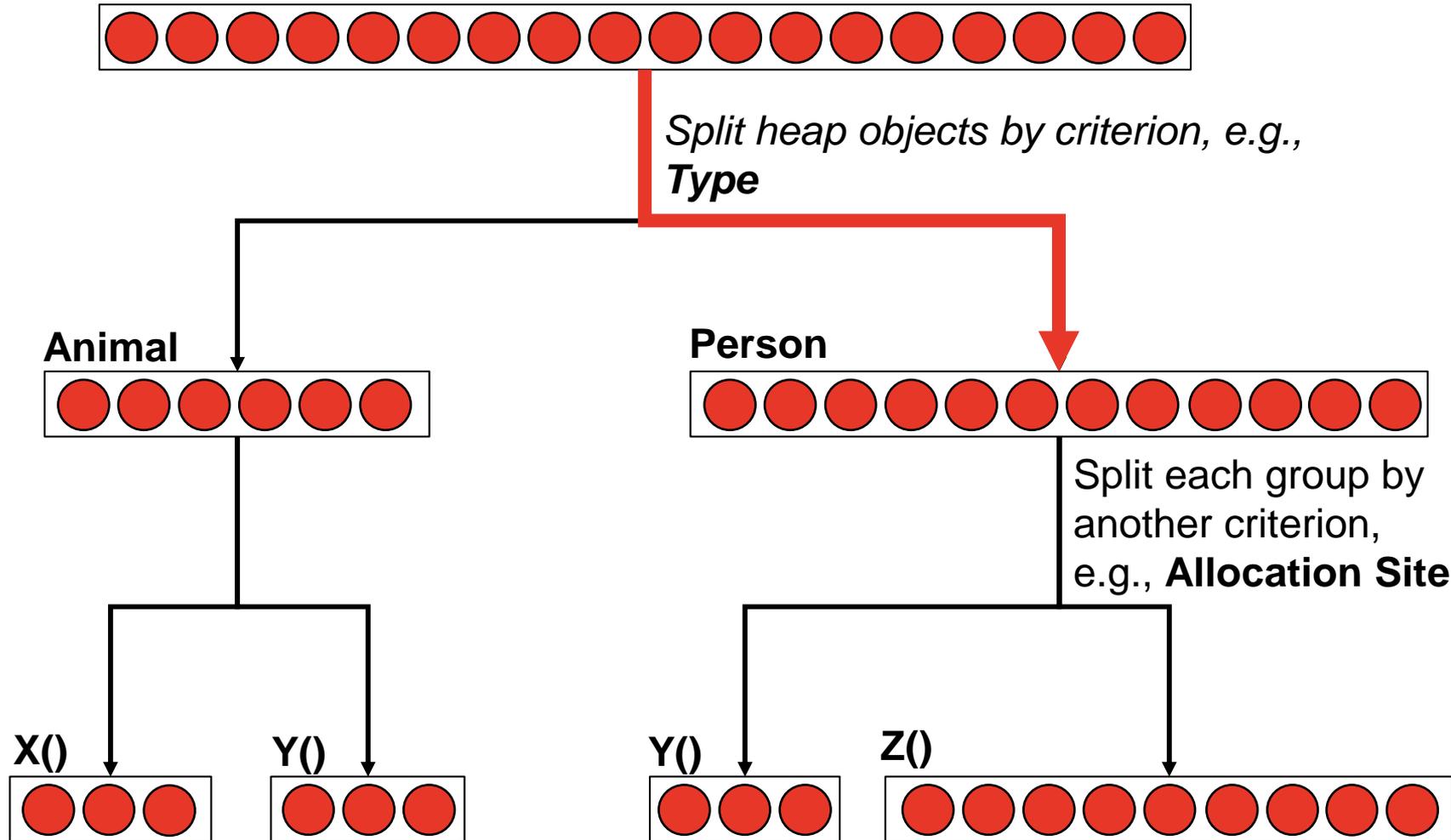
# OBJECT CLASSIFICATION



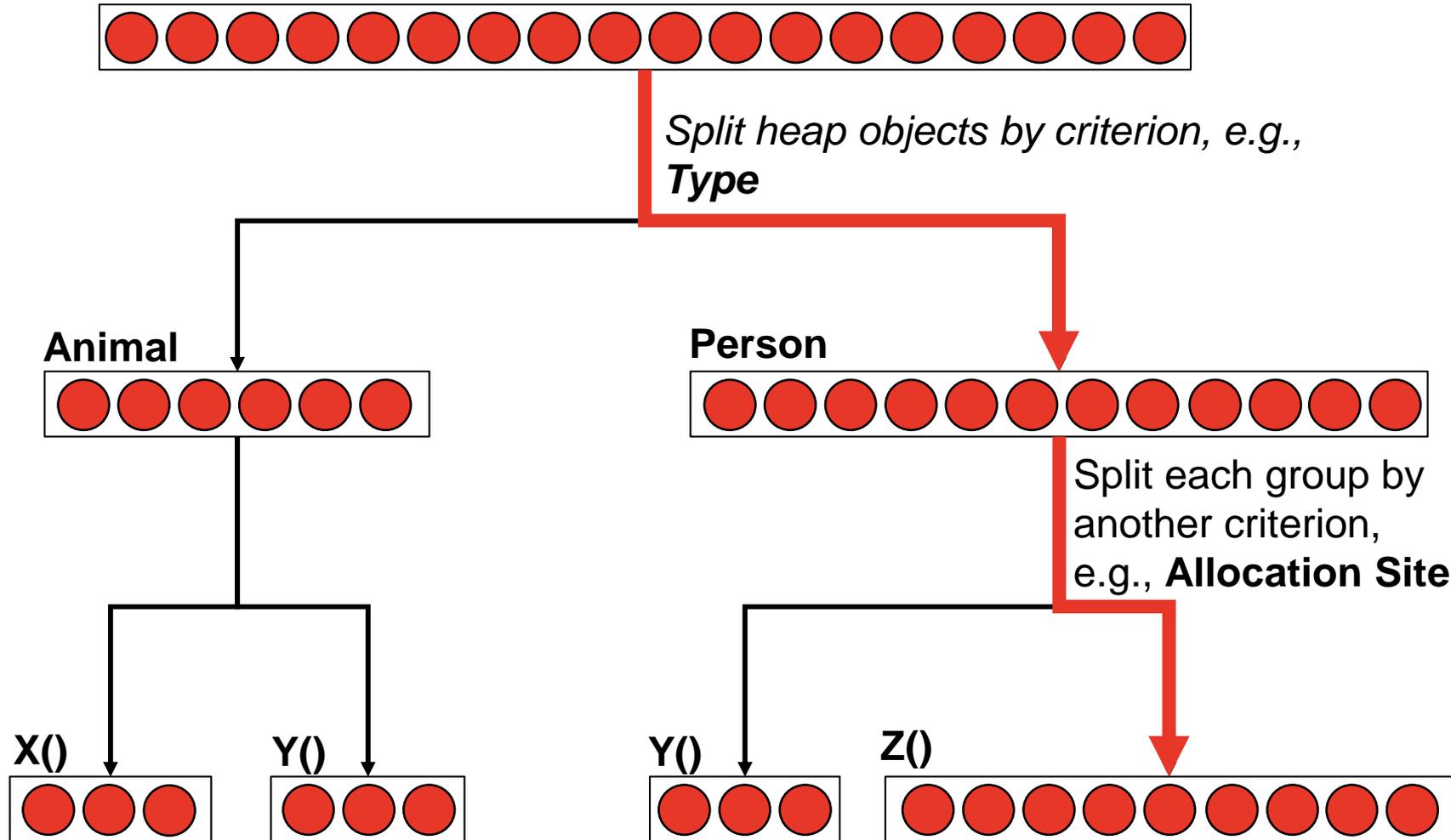
# OBJECT CLASSIFICATION



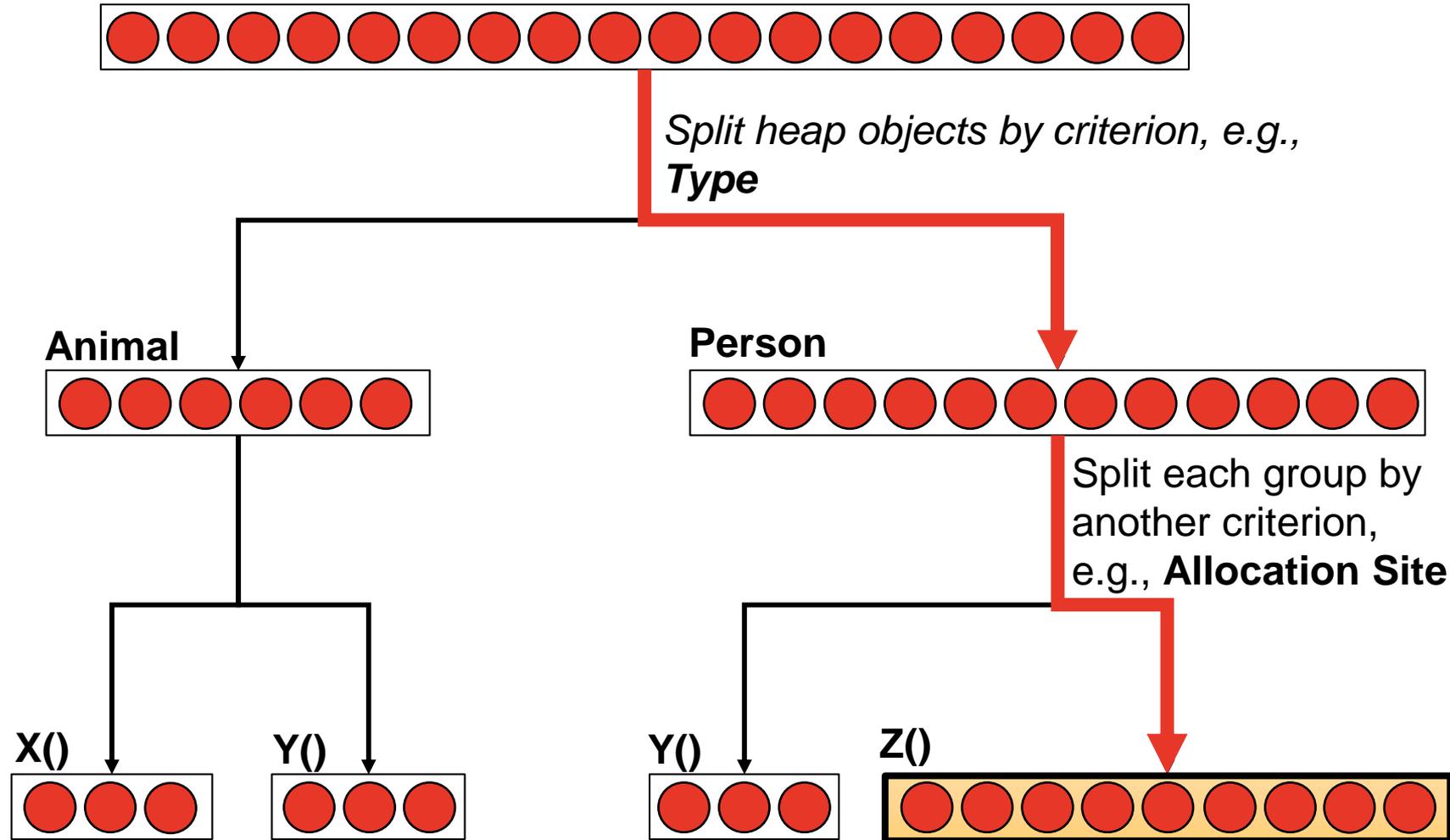
# OBJECT CLASSIFICATION



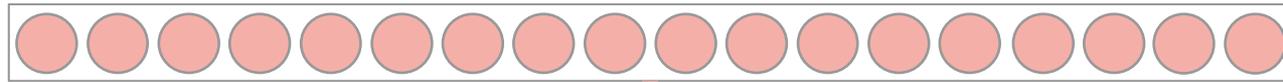
# OBJECT CLASSIFICATION



# OBJECT CLASSIFICATION

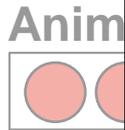


# OBJECT CLASSIFICATION

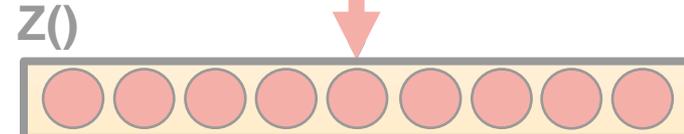
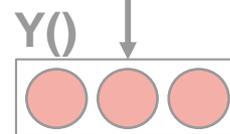
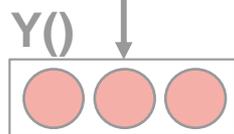
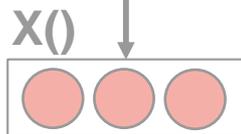


*Split heap objects by criterion, e.g.,*

Various grouping criteria can be used:  
Type,  
Package,  
Allocation site,  
Call sites,  
Allocating thread,  
Data structures,  
etc.



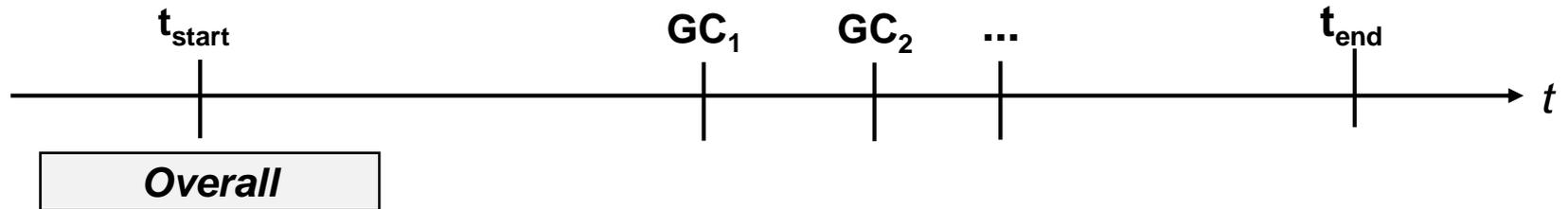
each group by  
her criterion,  
**Allocation Site**



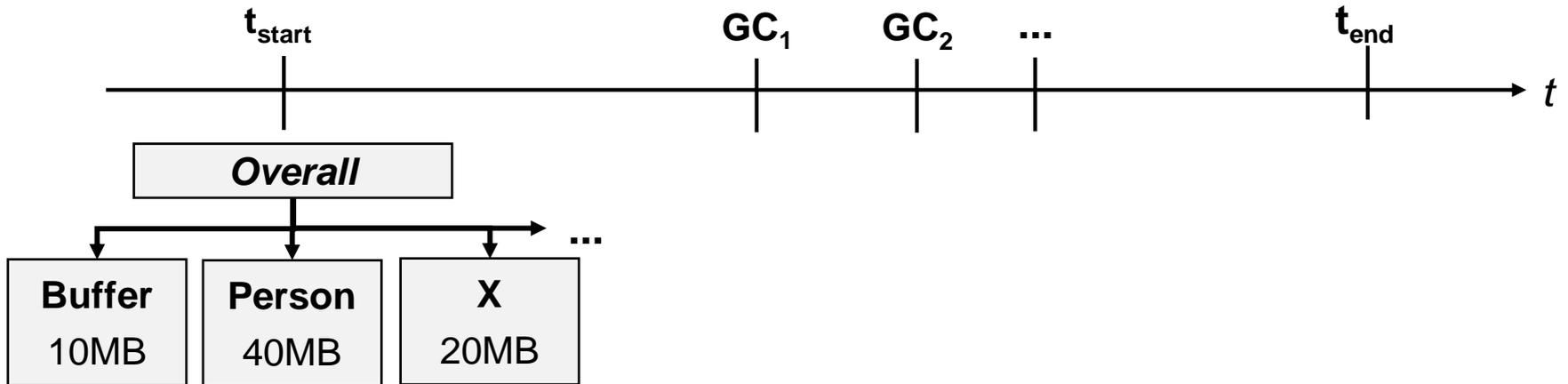
# MEMORY EVOLUTION



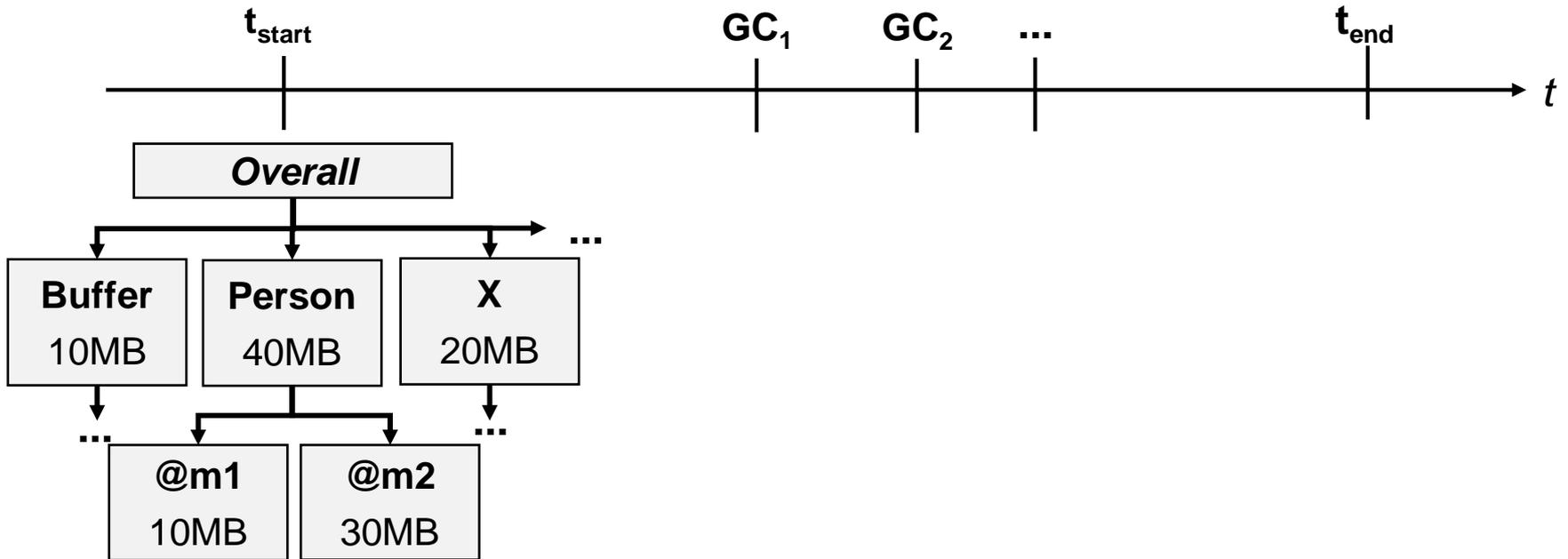
# MEMORY EVOLUTION



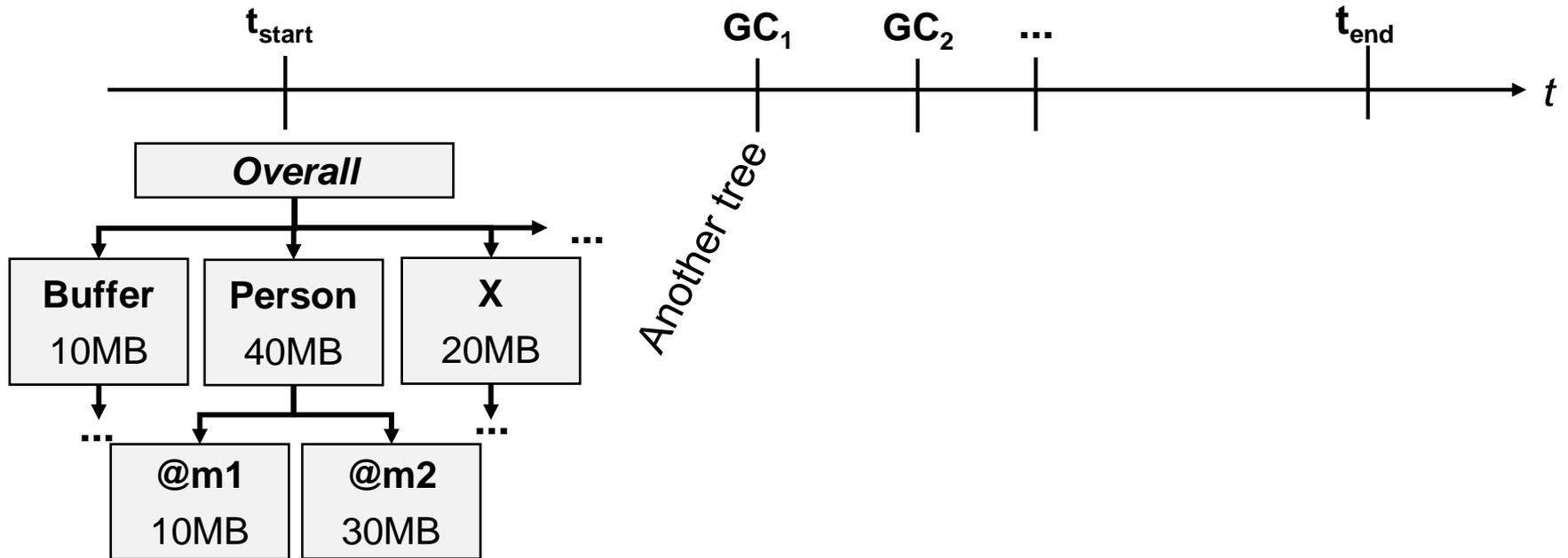
# MEMORY EVOLUTION



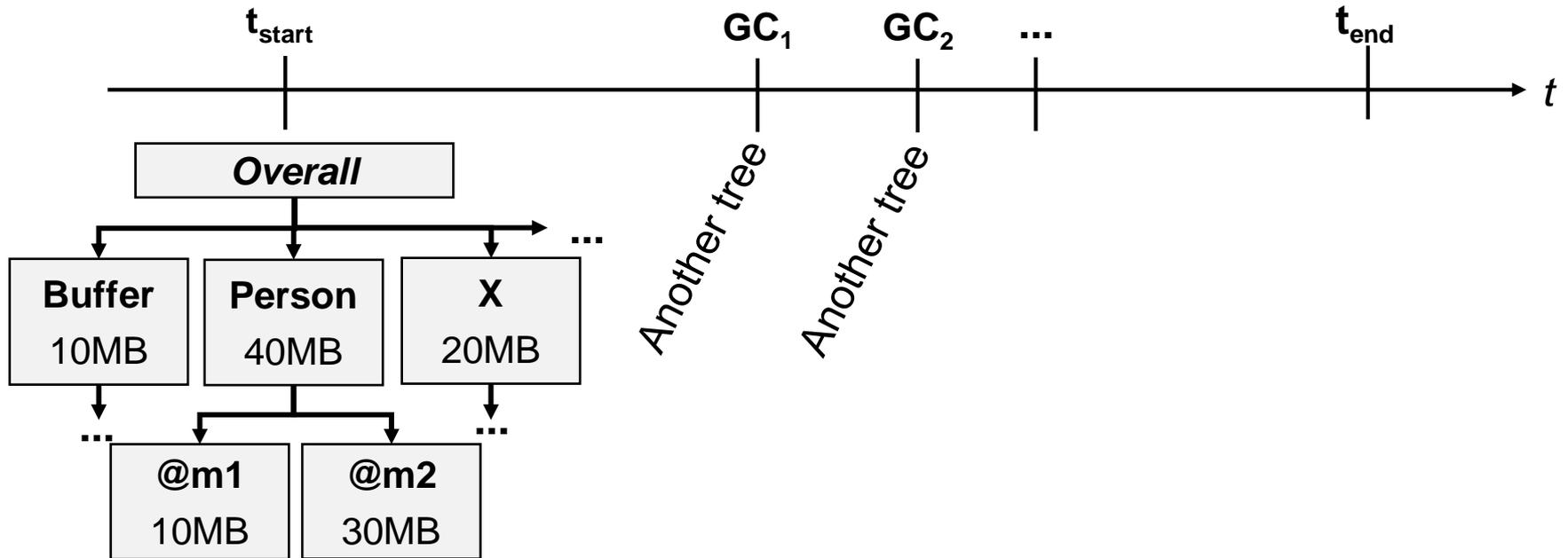
# MEMORY EVOLUTION



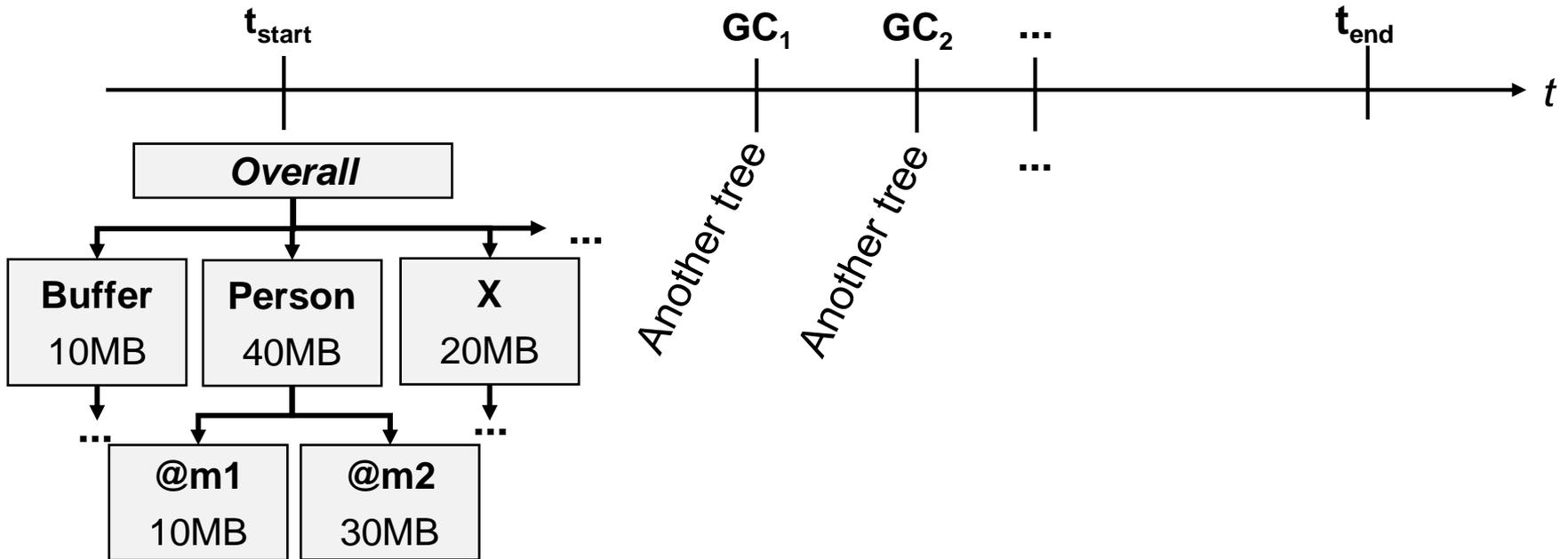
# MEMORY EVOLUTION



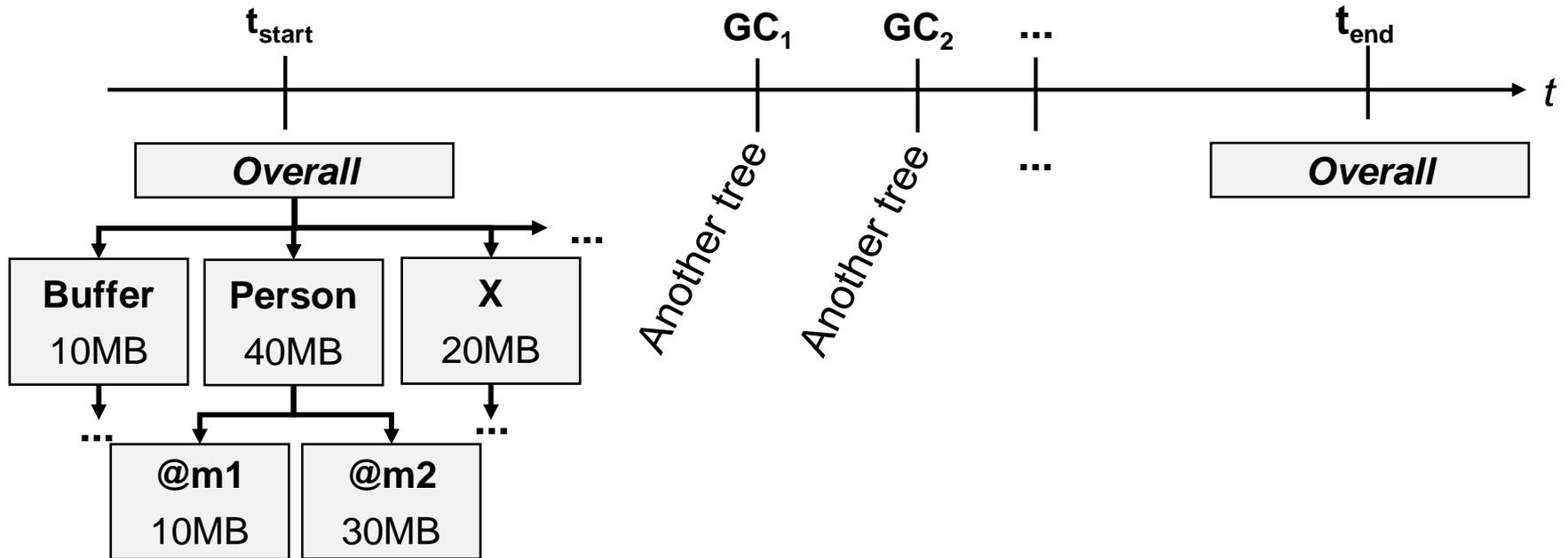
# MEMORY EVOLUTION



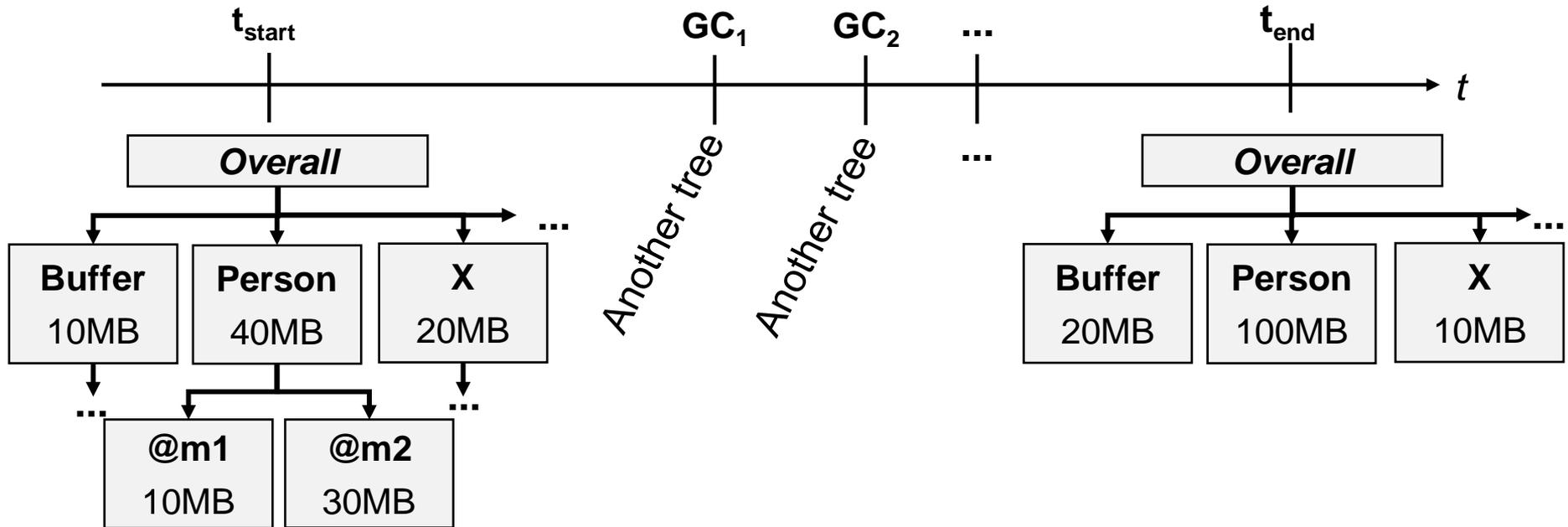
# MEMORY EVOLUTION



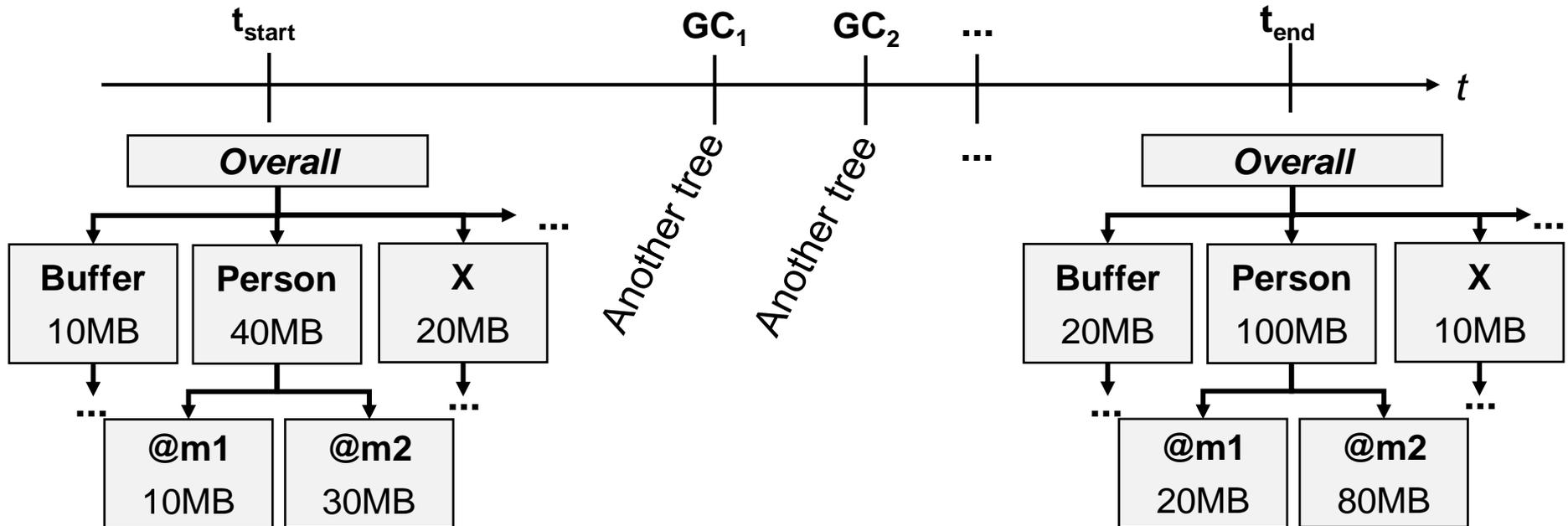
# MEMORY EVOLUTION



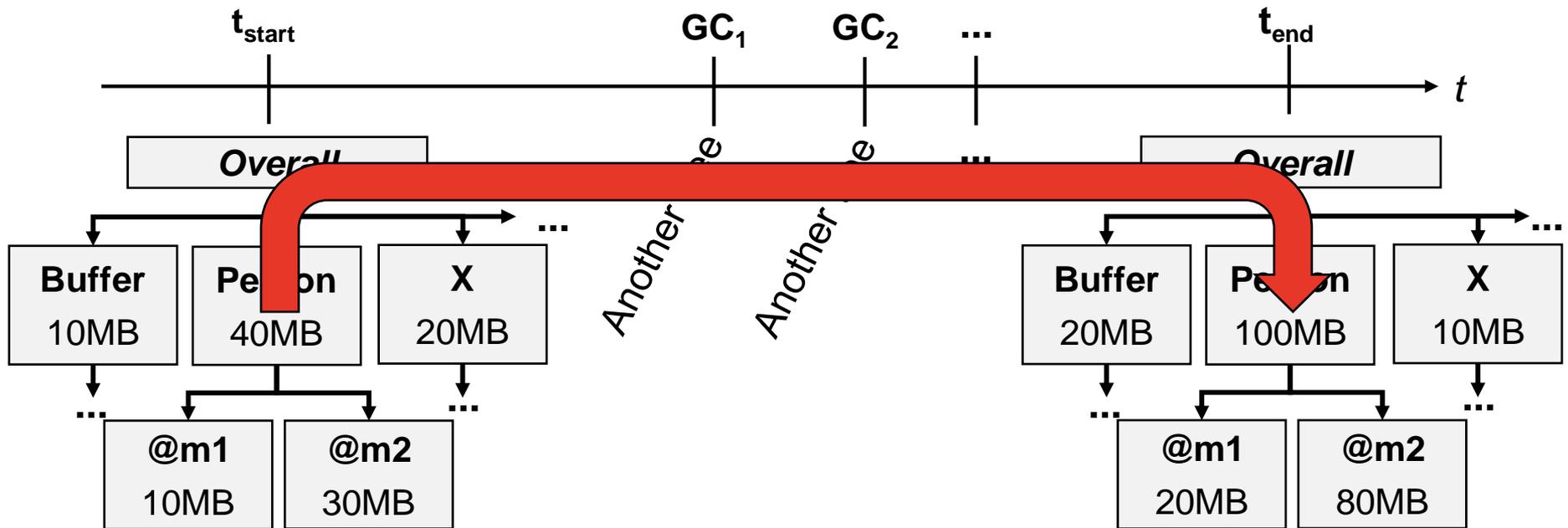
# MEMORY EVOLUTION



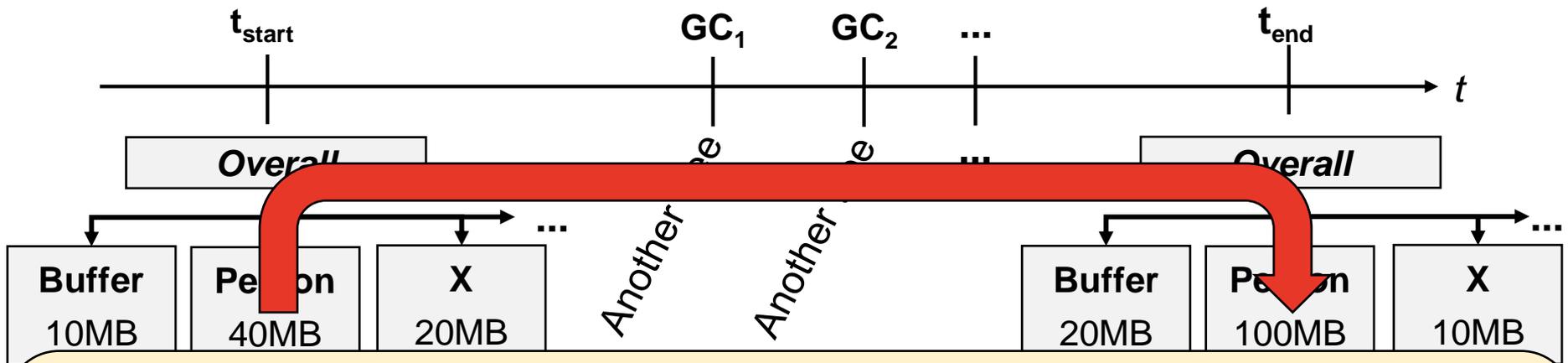
# MEMORY EVOLUTION



# MEMORY EVOLUTION



# MEMORY EVOLUTION



**Question: How to visualize memory trees and their evolution?**

# REQUIREMENTS

# REQUIREMENTS

**“Bigger is more” principle**

# REQUIREMENTS

**“Bigger is more” principle**

**Suitable for interactive visualization**

# REQUIREMENTS

**“Bigger is more” principle**

**Suitable for interactive visualization**

**Proven and easy to understand**

# REQUIREMENTS

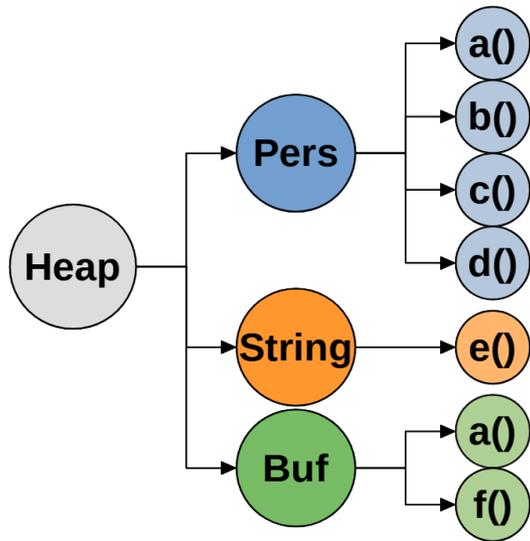
**“Bigger is more” principle**

**Suitable for interactive visualization**

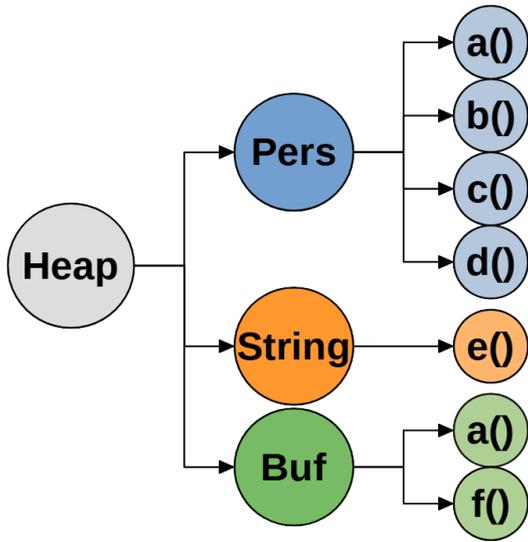
**Proven and easy to understand**

**Stable layout on evolving data**

# VISUALIZATIONS COMPARED

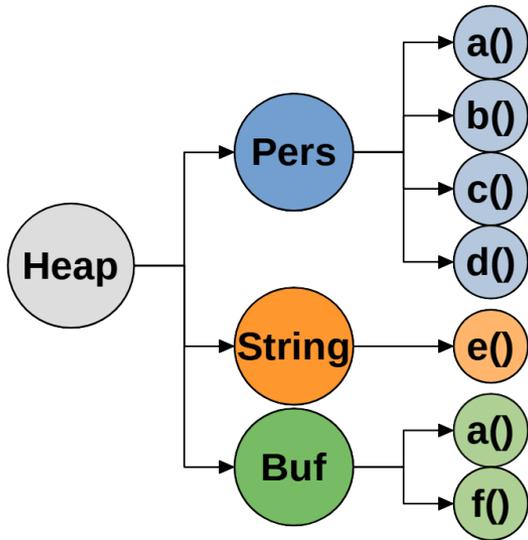


# VISUALIZATIONS COMPARED

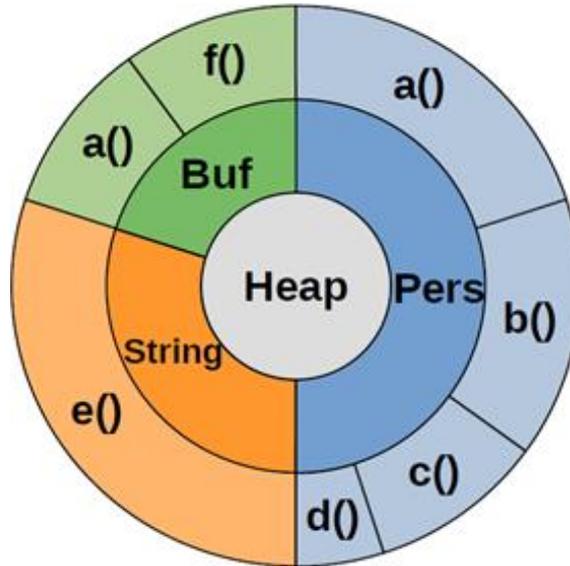


Node-Link Diag.

# VISUALIZATIONS COMPARED

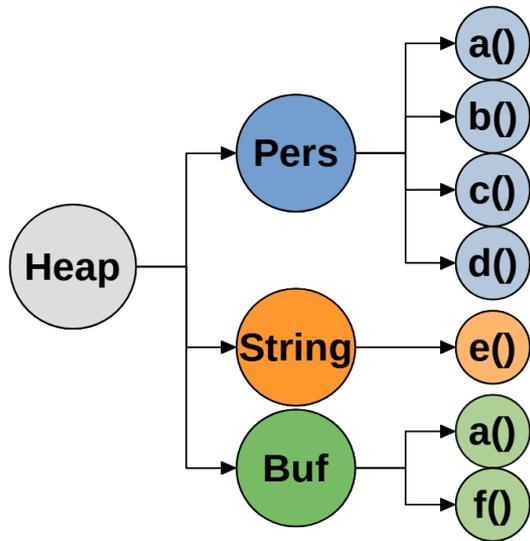


Node-Link Diag.

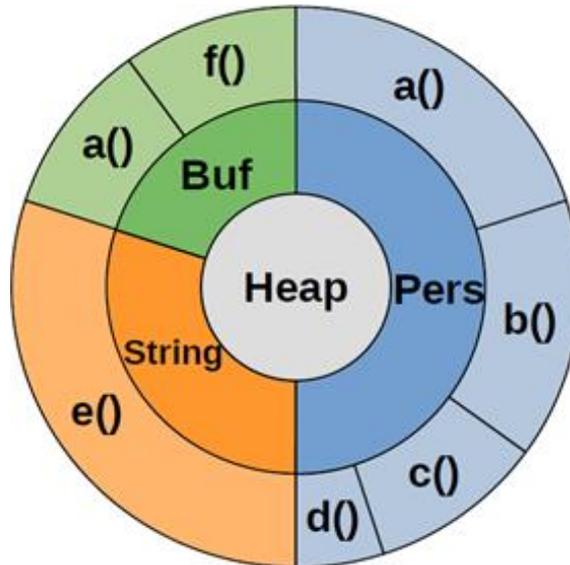


Sunburst

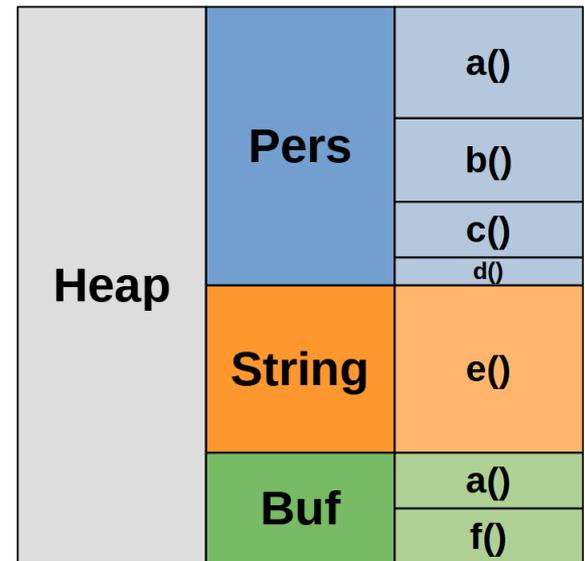
# VISUALIZATIONS COMPARED



Node-Link Diag.



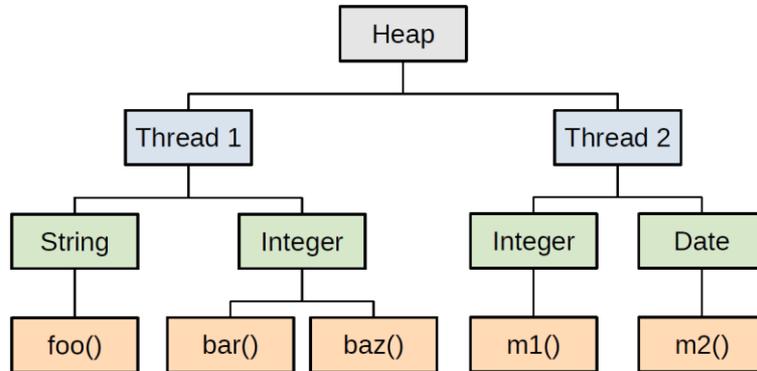
Sunburst



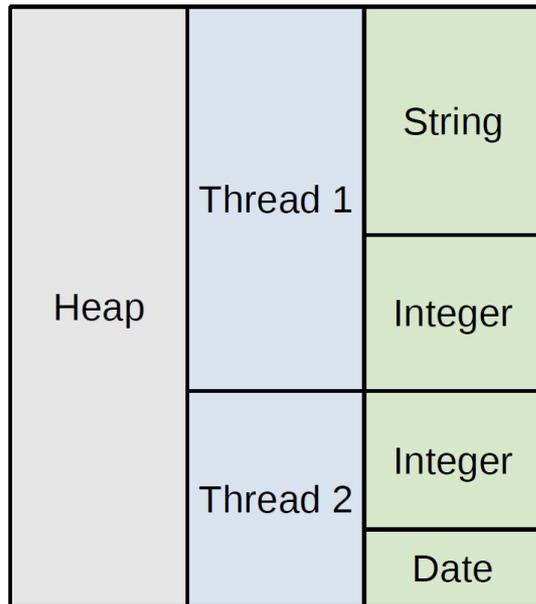
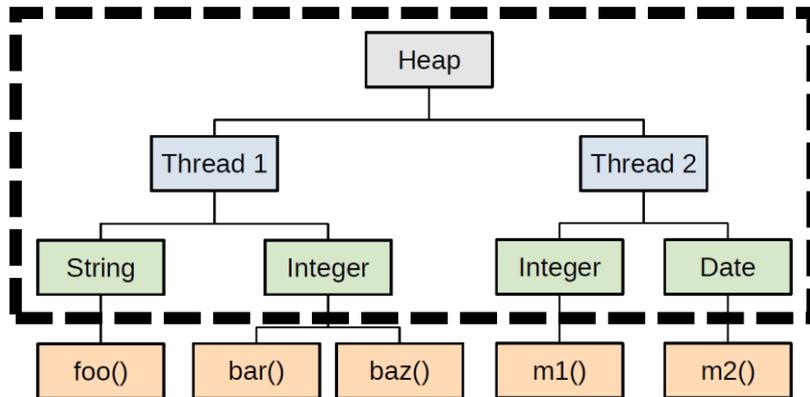
Icicle

# HEAP STATE VISUALIZATION

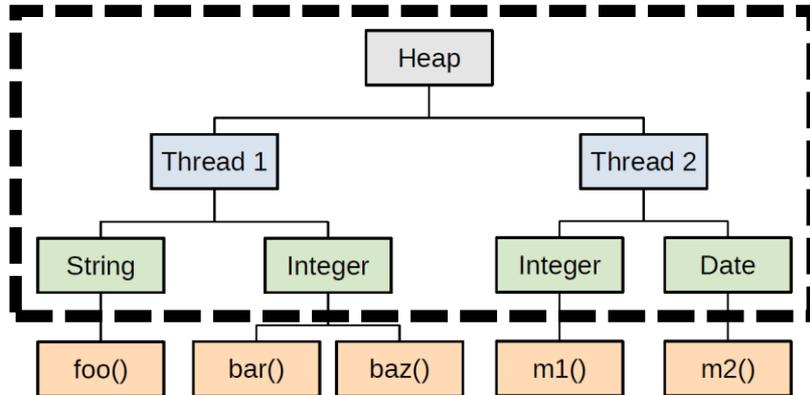
# HEAP STATE VISUALIZATION



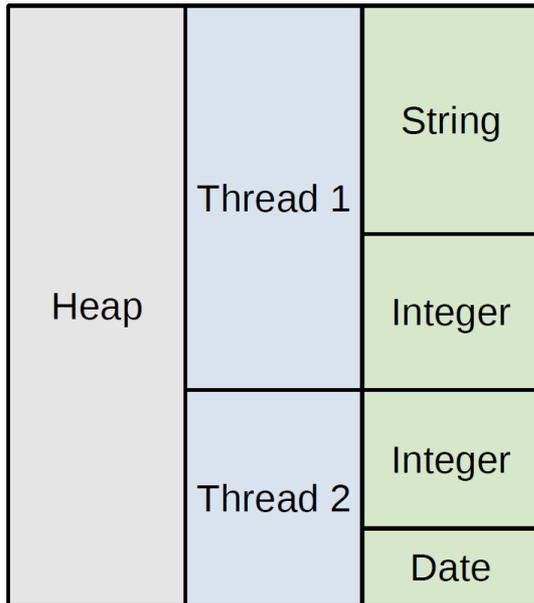
# HEAP STATE VISUALIZATION



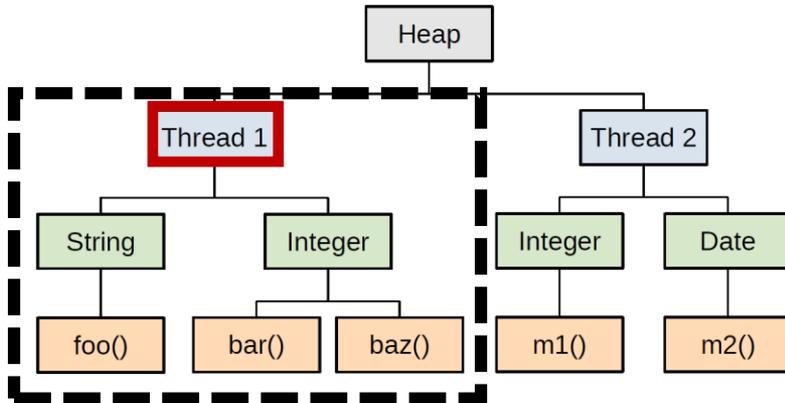
# HEAP STATE VISUALIZATION



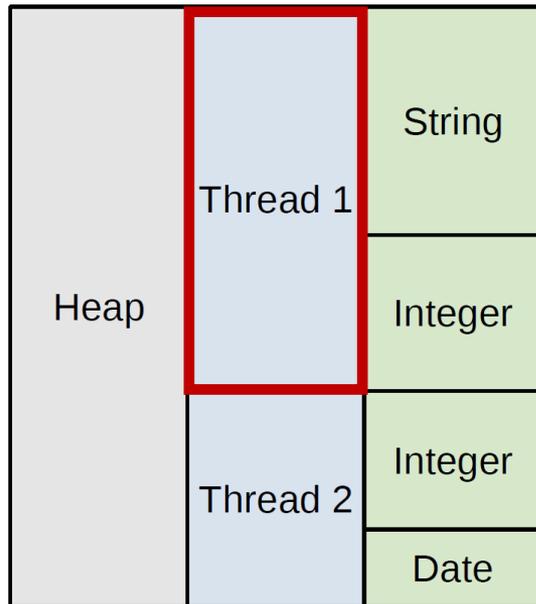
**Drill-down**



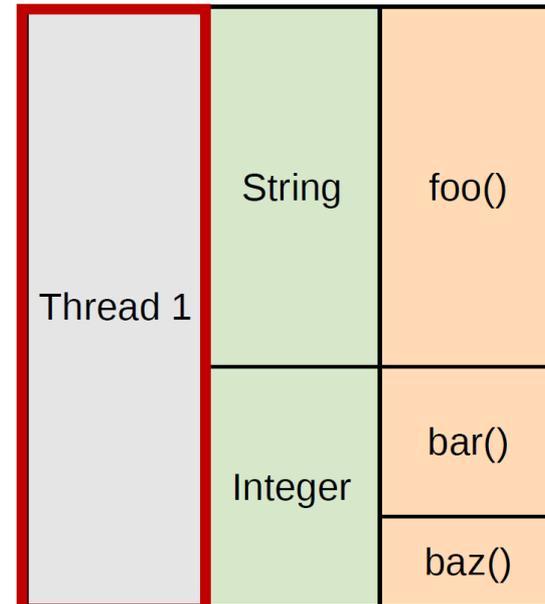
# HEAP STATE VISUALIZATION



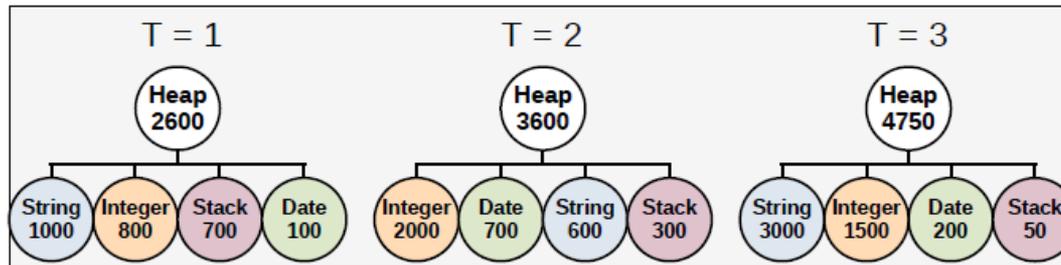
**Drill-down**



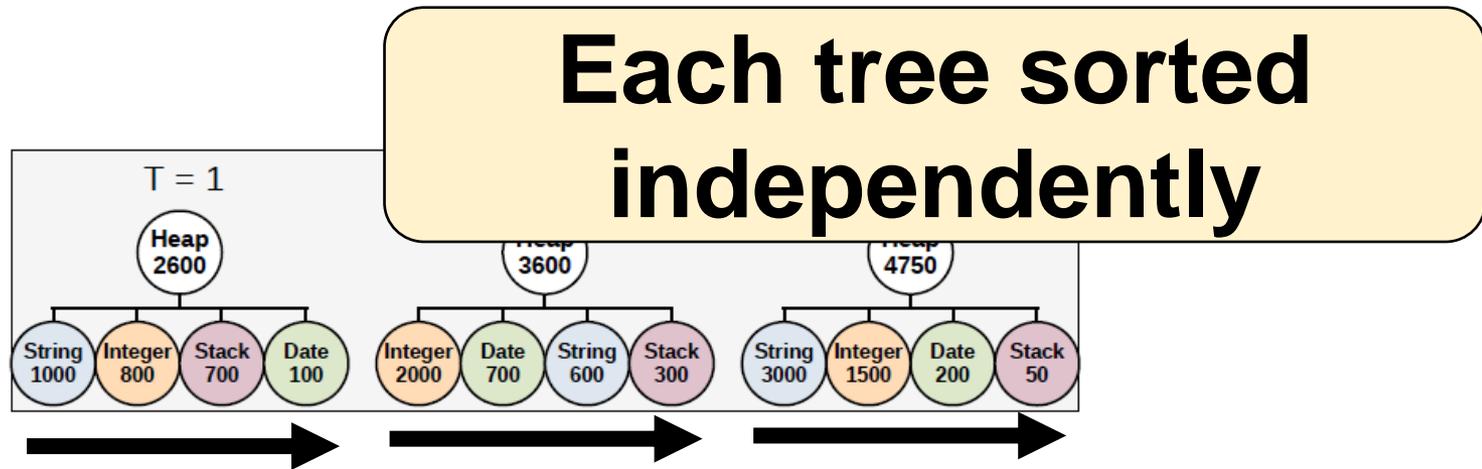
Drill-down into *Thread 1*



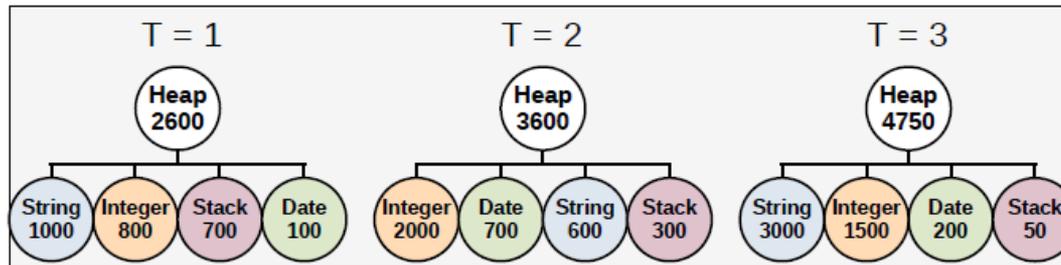
# HEAP EVOLUTION VISUALIZATION



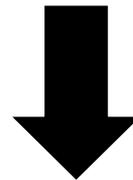
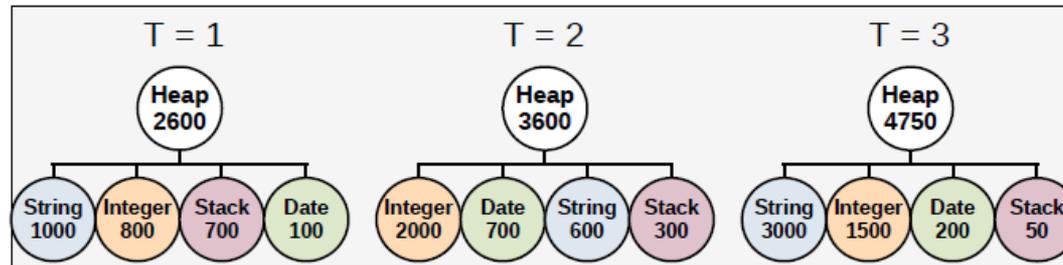
# HEAP EVOLUTION VISUALIZATION



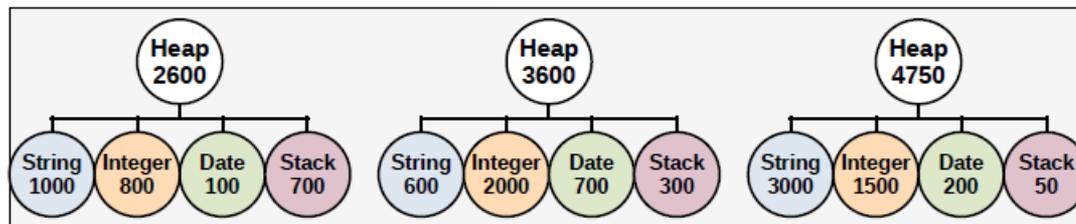
# HEAP EVOLUTION VISUALIZATION



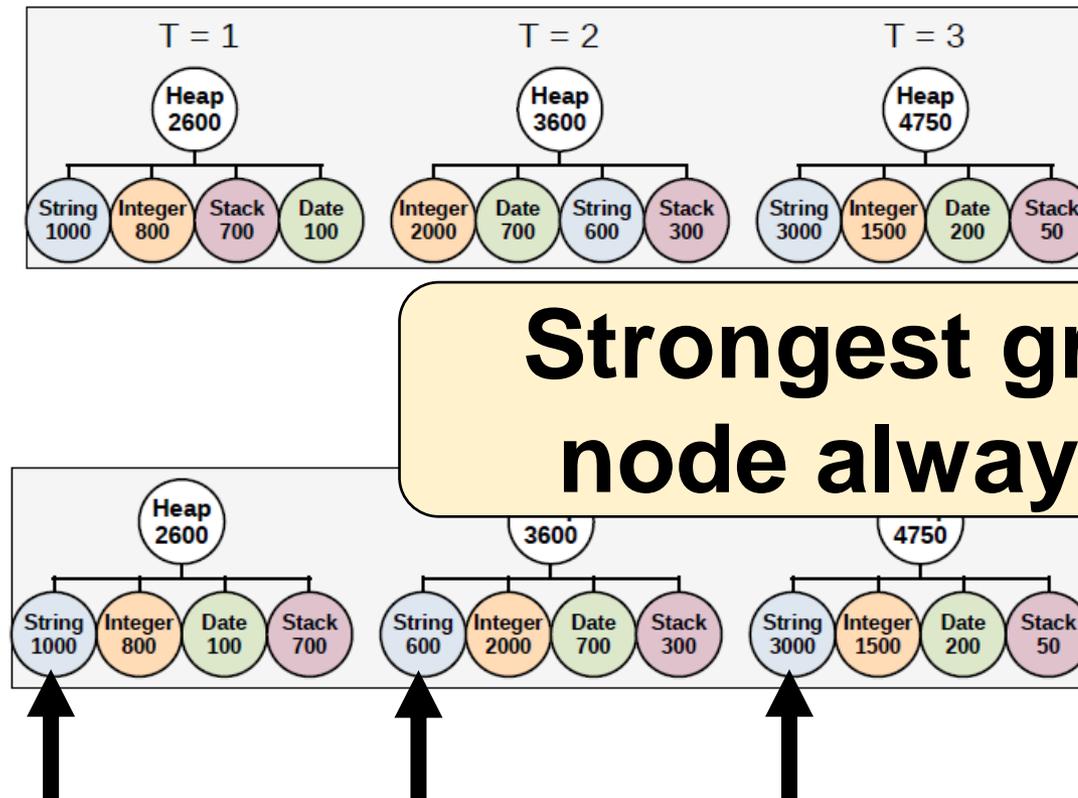
# HEAP EVOLUTION VISUALIZATION



Sort all tree nodes using fixed sorting (e.g., absolute growth)



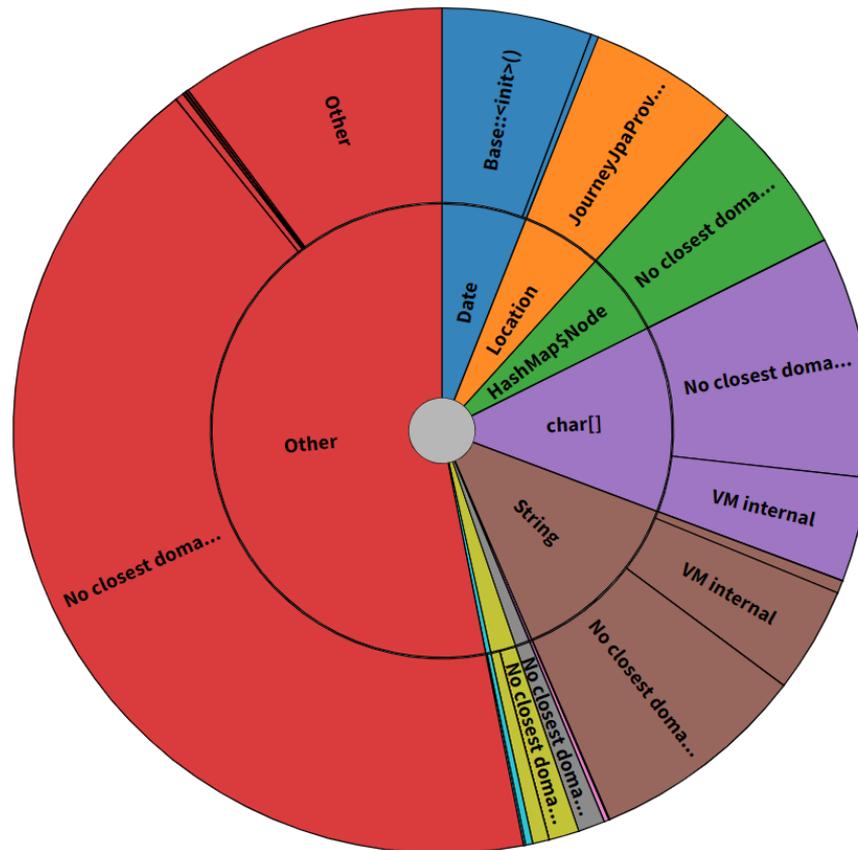
# HEAP EVOLUTION VISUALIZATION



# RELATIVE POSITION ANIMATION

# RELATIVE POSITION ANIMATION

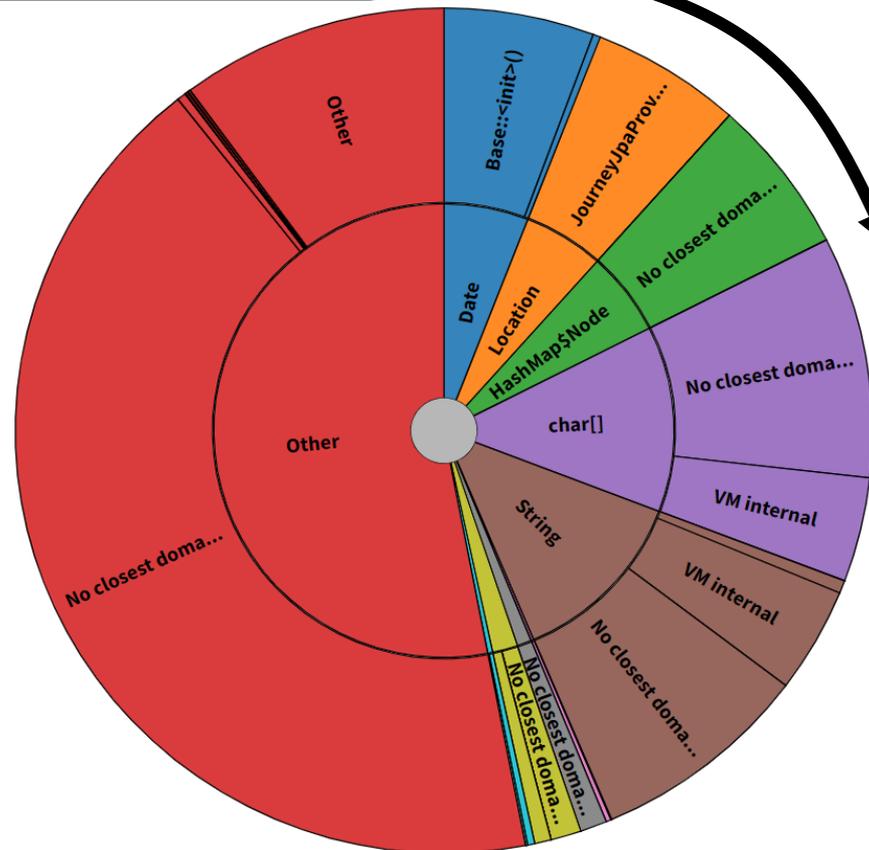
Current Heap: Number: 0 | Time: 147,252ms | Size: 564,129



# RELATIVE POSITION ANIMATION

Sorted by growth

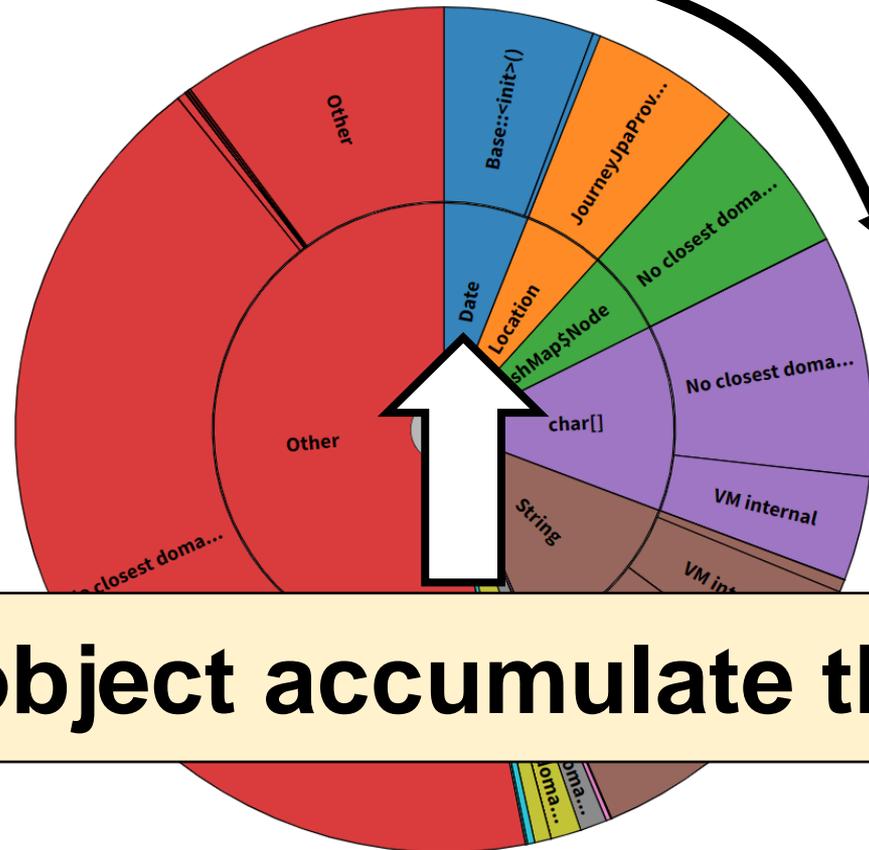
Time: 147,252ms | Size: 564,129



# RELATIVE POSITION ANIMATION

Sorted by growth

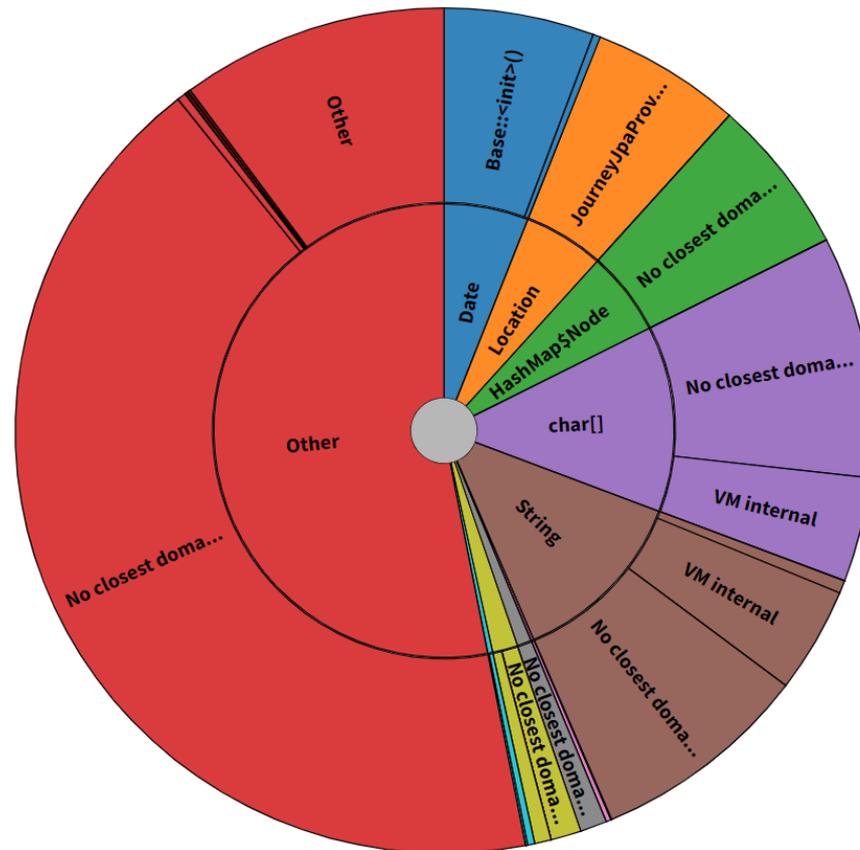
Time: 147,252ms | Size: 564,129



*Date* object accumulate the most

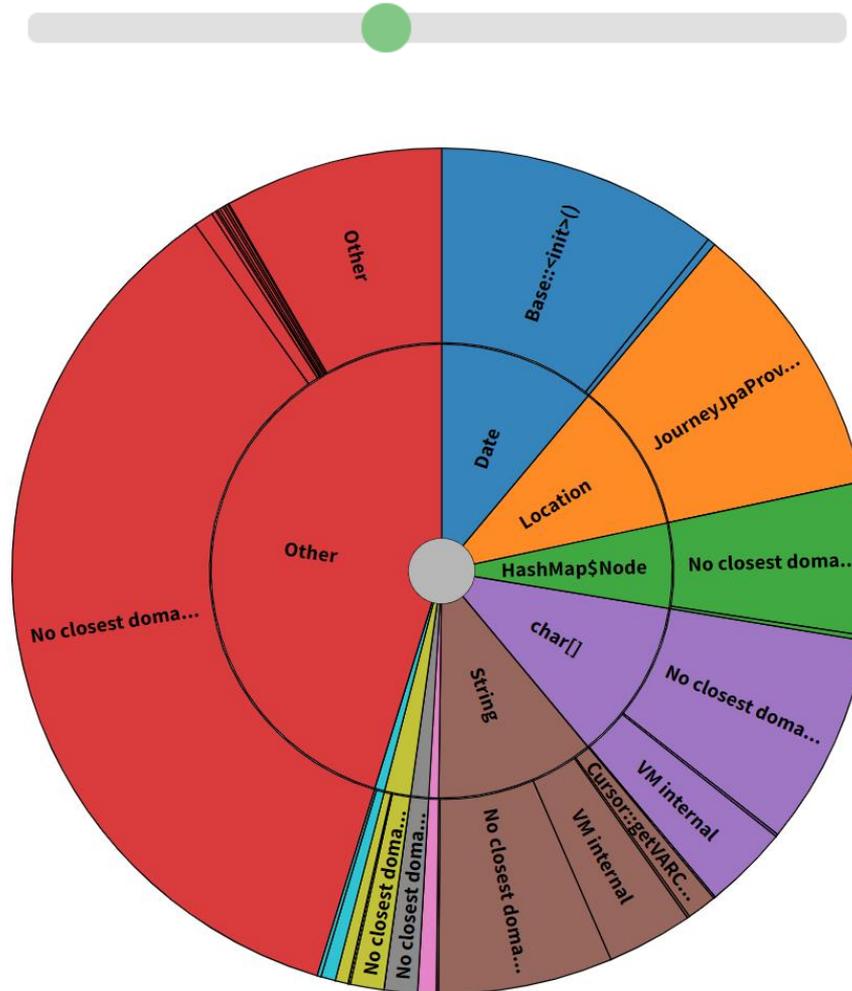
# RELATIVE POSITION ANIMATION

Current Heap: Number: 0 | Time: 147,252ms | Size: 564,129



# RELATIVE POSITION ANIMATION

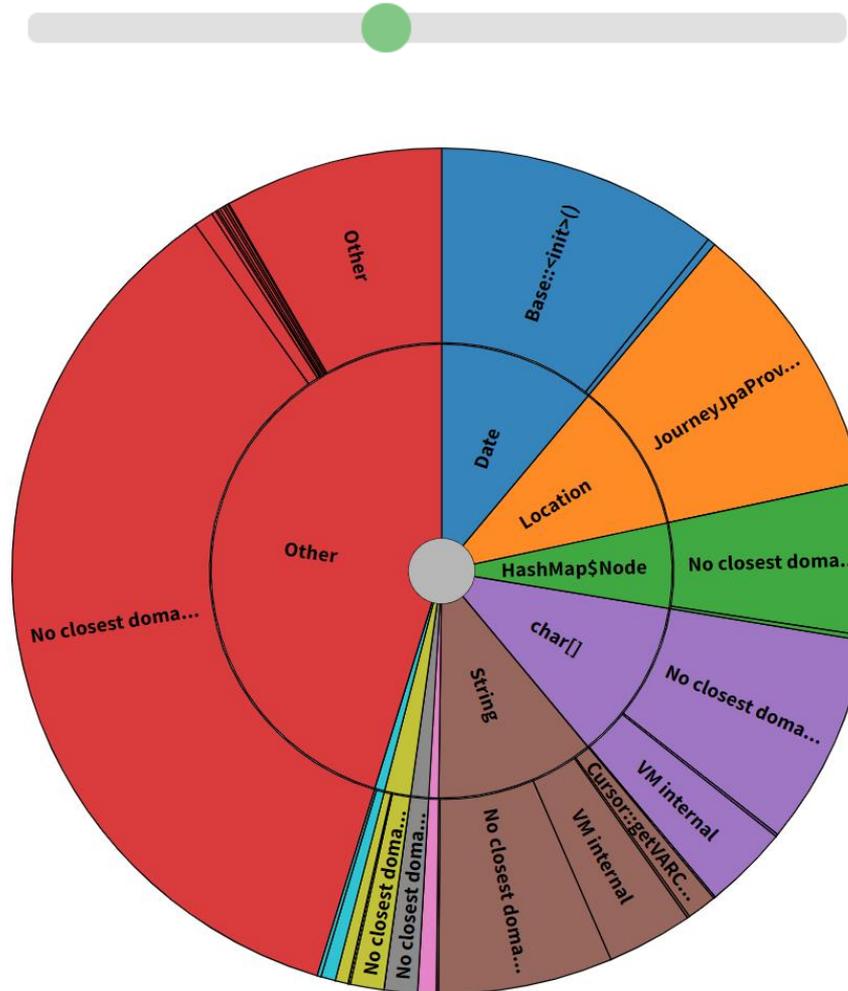
Current Heap: Number: 10 | Time: 309,987ms | Size: 717,439





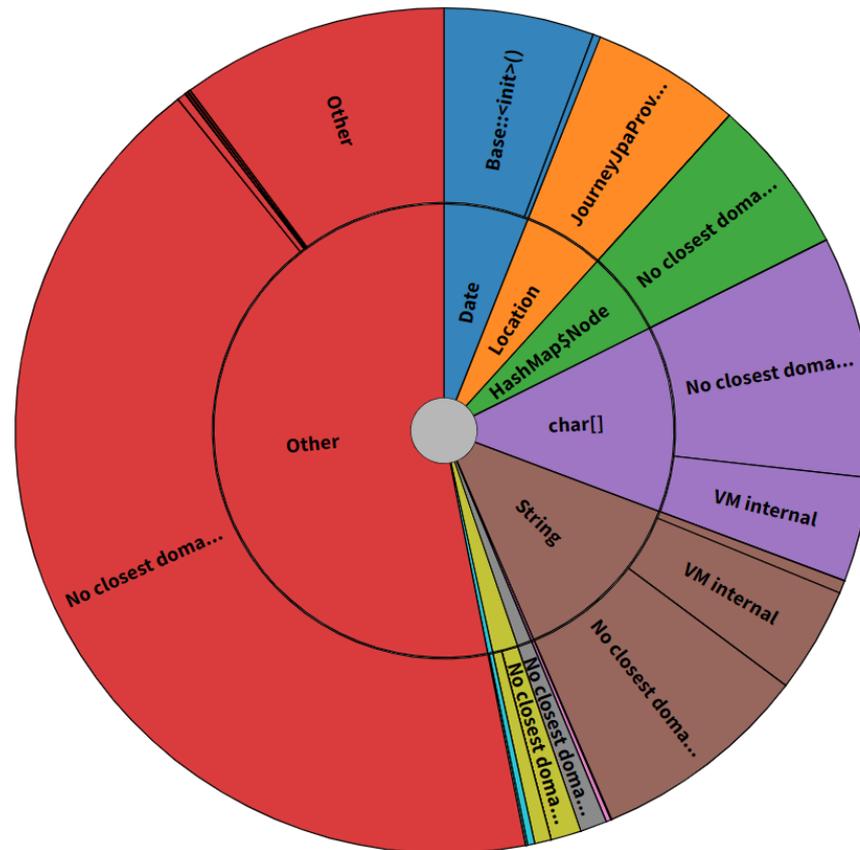
# RELATIVE POSITION ANIMATION

Current Heap: Number: 10 | Time: 309,987ms | Size: 717,439



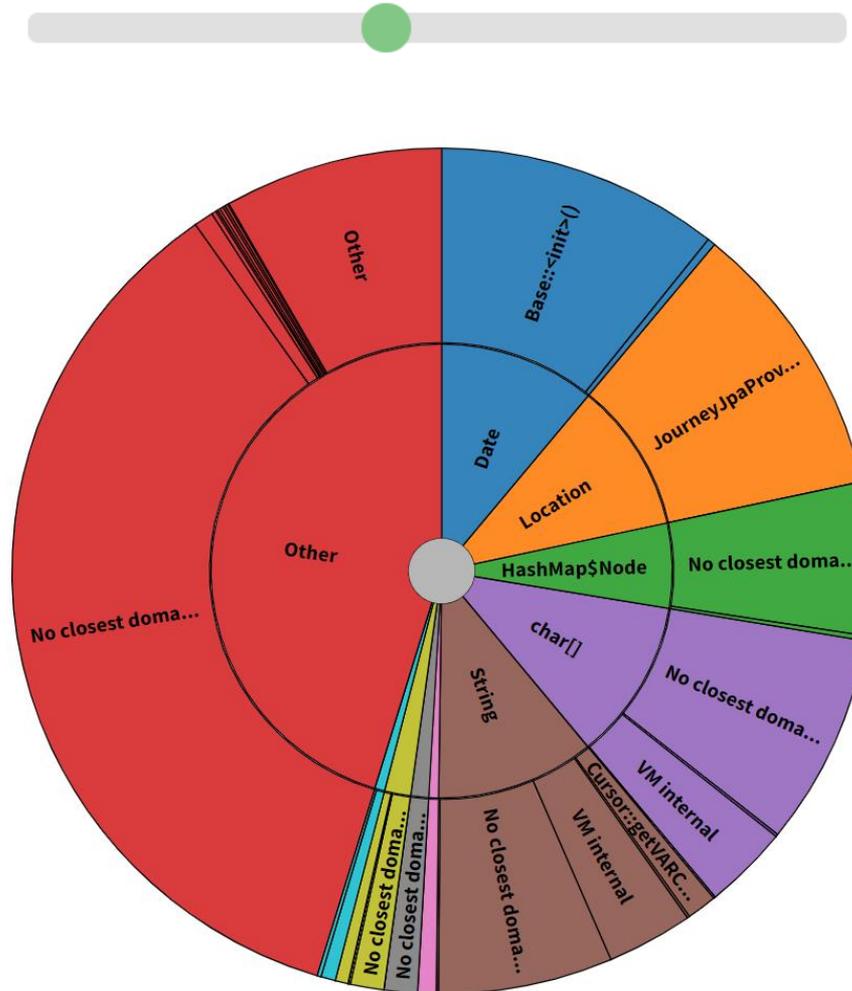
# RELATIVE POSITION ANIMATION

Current Heap: Number: 0 | Time: 147,252ms | Size: 564,129



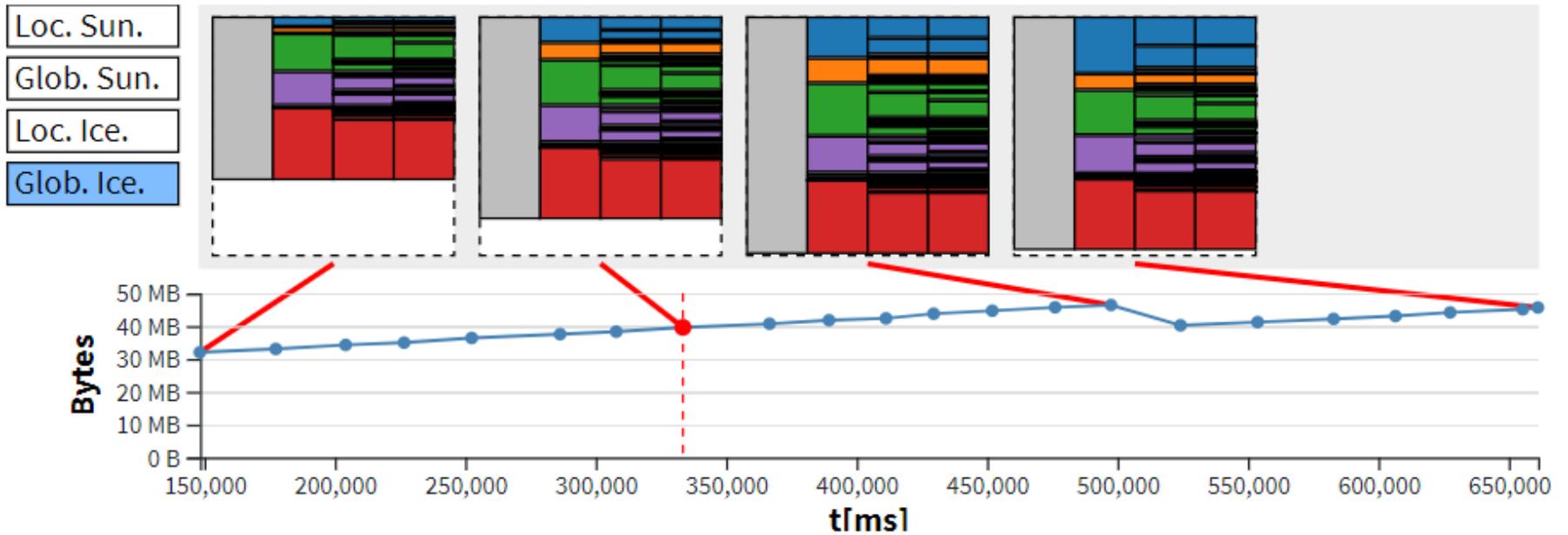
# RELATIVE POSITION ANIMATION

Current Heap: Number: 10 | Time: 309,987ms | Size: 717,439

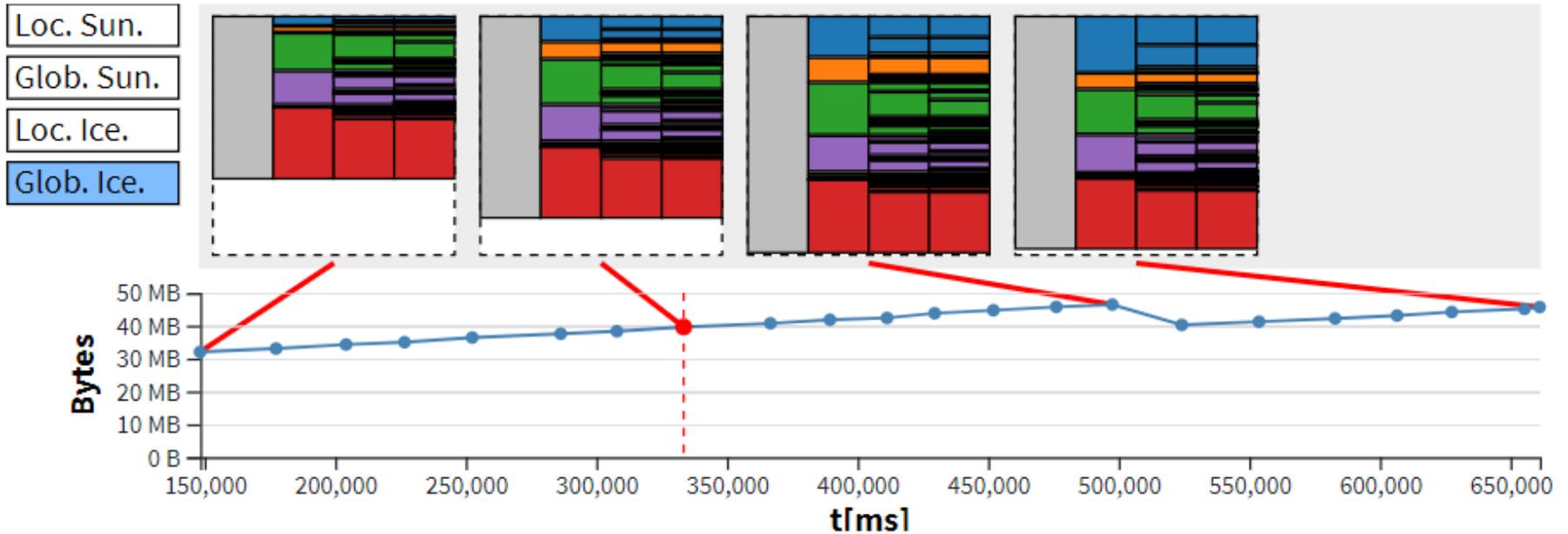




# TIMELINE VIEW

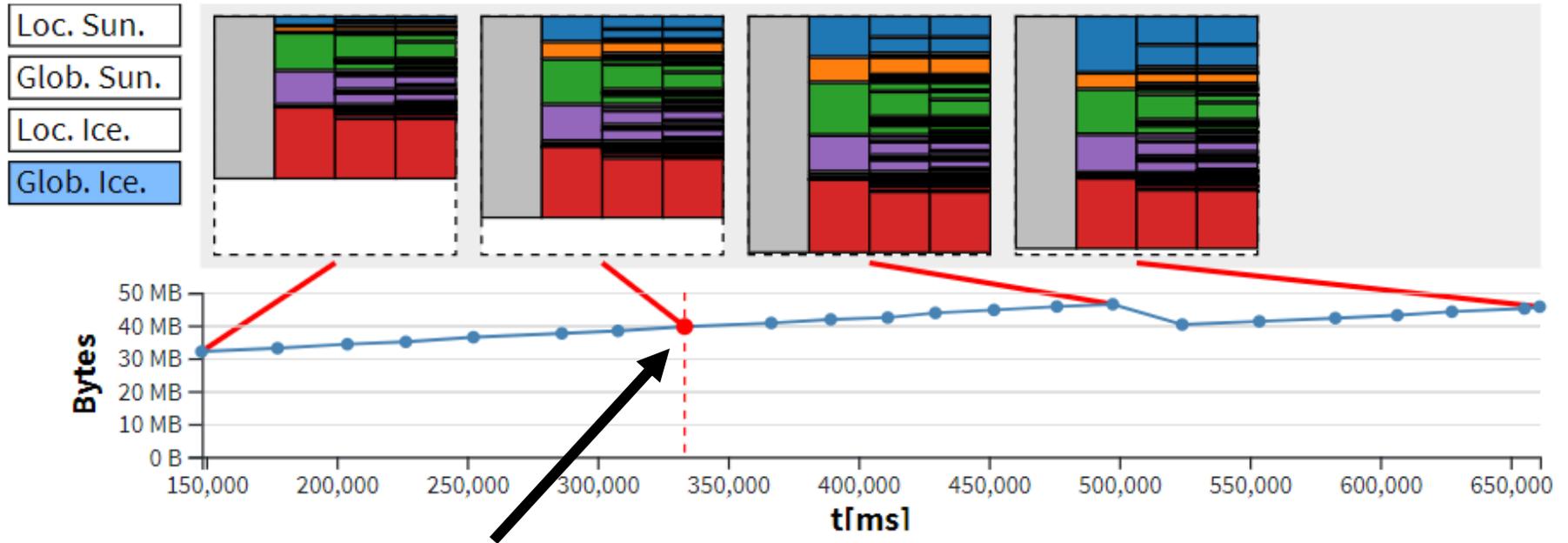


# TIMELINE VIEW



**Small multiples at different points  
in time**

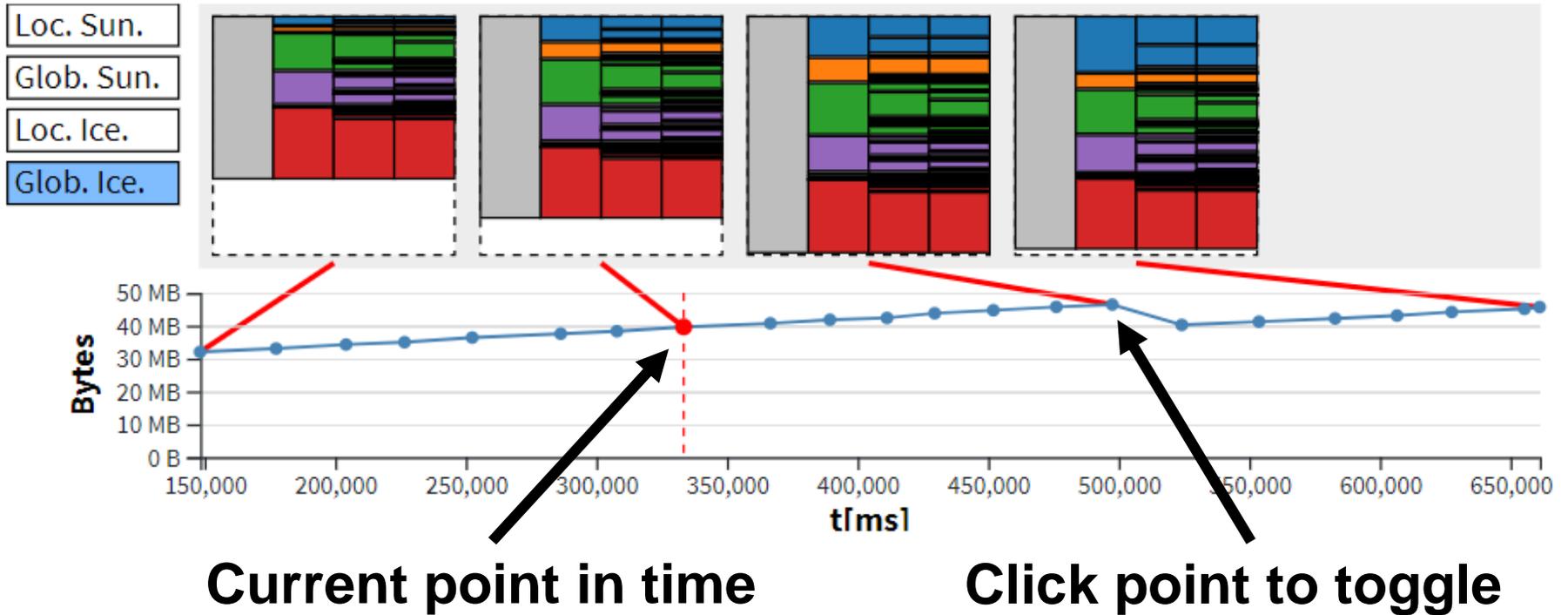
# TIMELINE VIEW



Current point in time

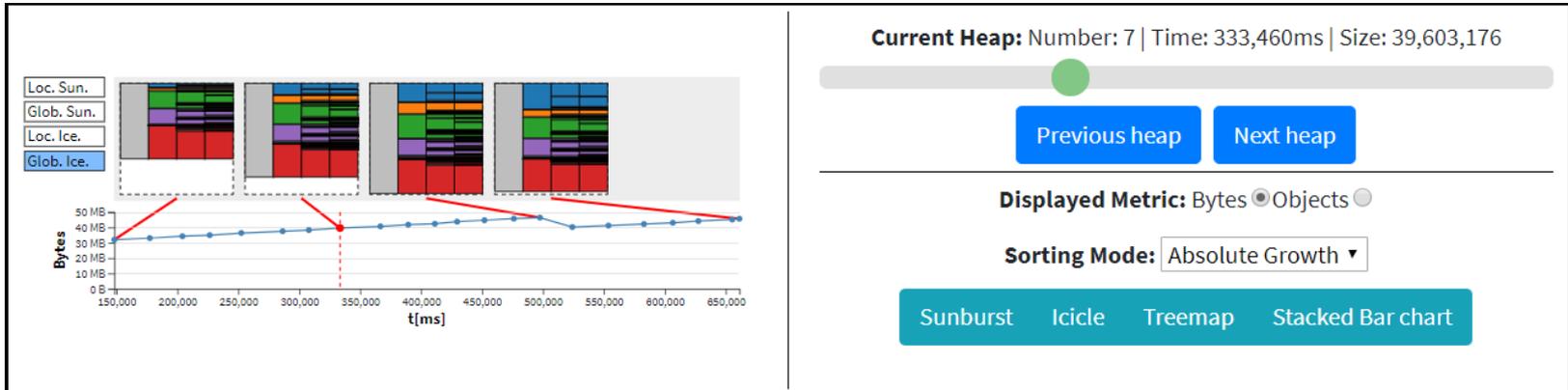
Small multiples at different points in time

# TIMELINE VIEW

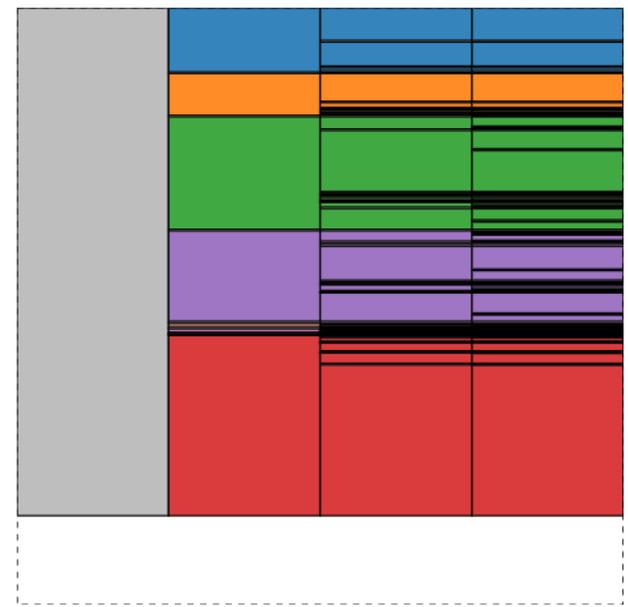
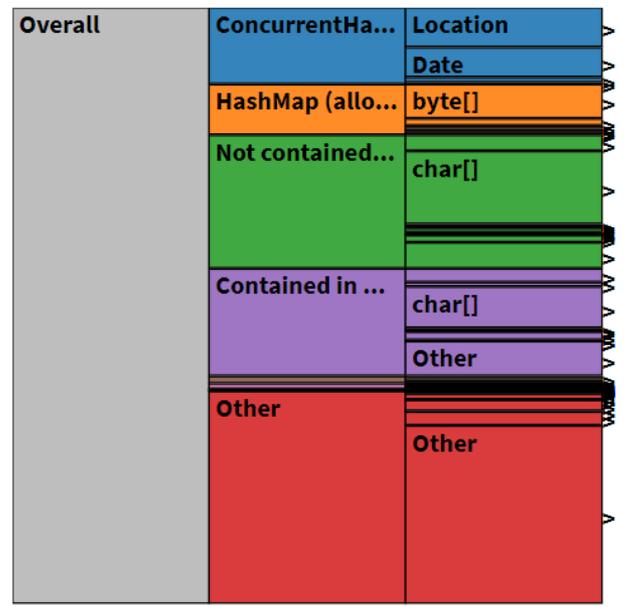


**Small multiples at different points in time**

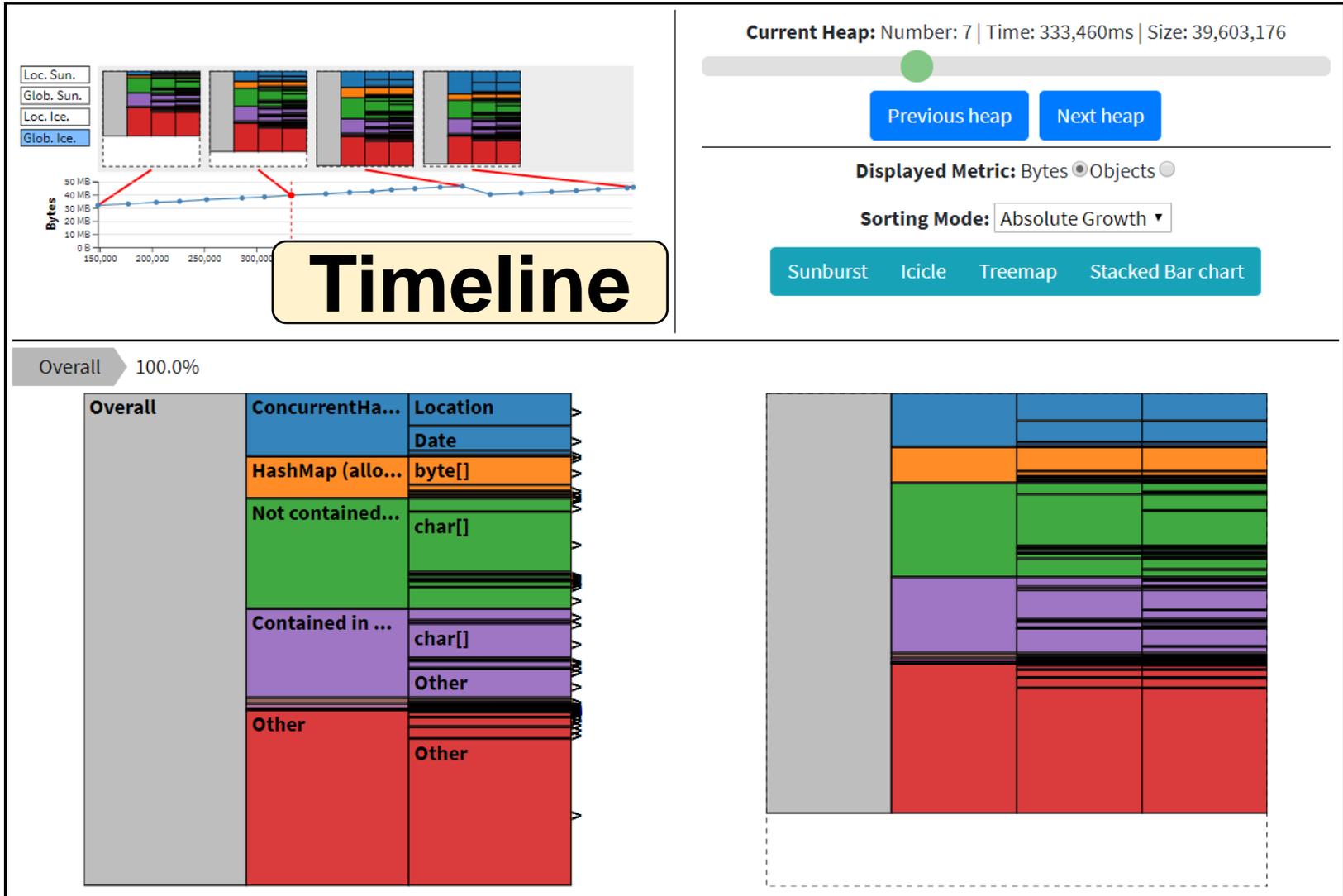
# PROTOTYPE AVAILABLE



Overall 100.0%

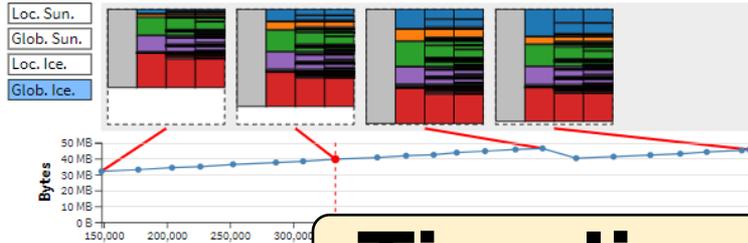


# PROTOTYPE AVAILABLE



# Time navigation

## PROTOTYPE AVAILABLE



Current Heap: Number: 7 | Time: 333,460ms | Size: 39,603,176



Previous heap

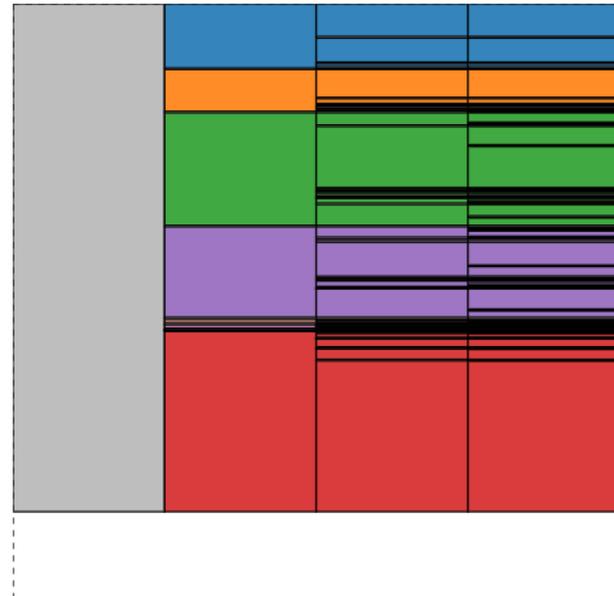
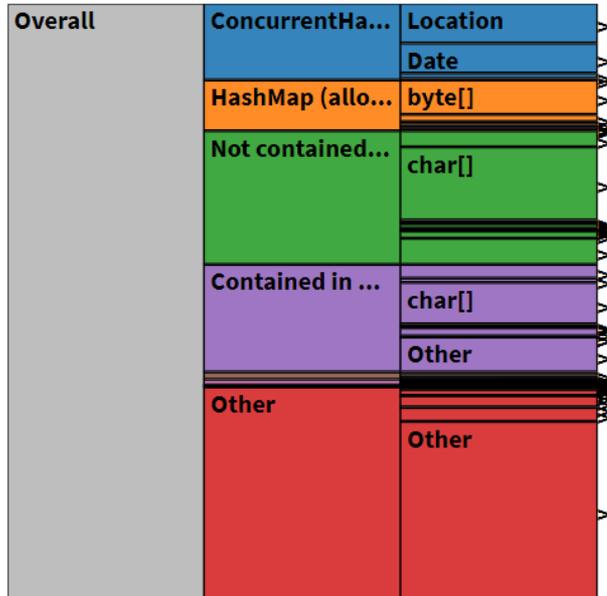
Next heap

Displayed Metric: Bytes  Objects

Sorting Mode: Absolute Growth ▾

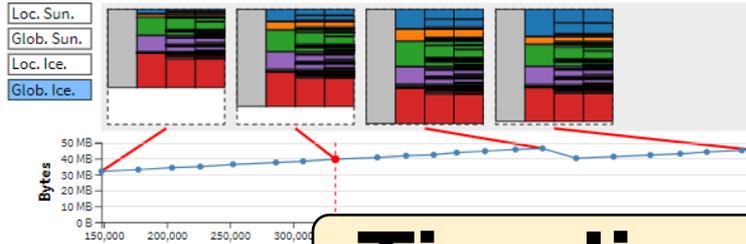
Sunburst Icicle Treemap Stacked Bar chart

Overall 100.0%



# PROTOTYPE AVAILABLE

# Time navigation



## Timeline

Current Heap: Number: 7 | Time: 333,460ms | Size: 39,603,176



Previous heap

Next heap

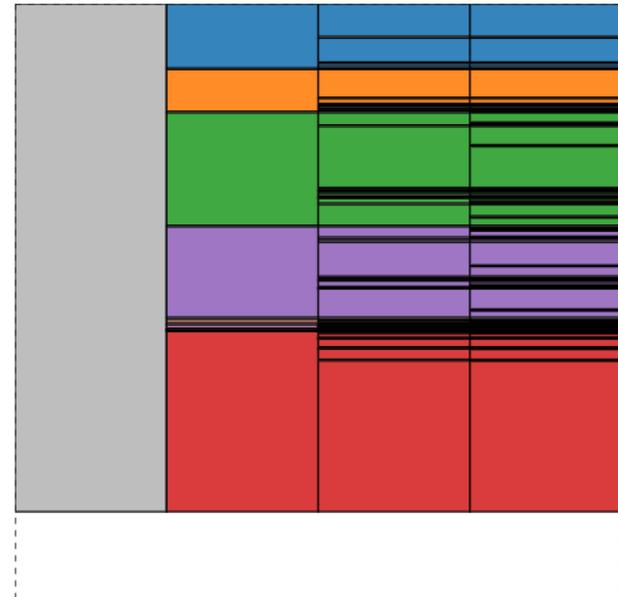
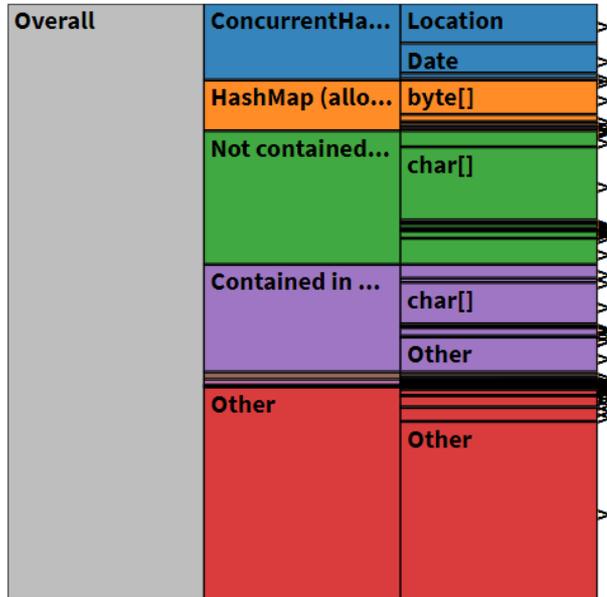
Displayed Metric: Bytes  Objects

Sorting Mode: Absolute Growth ▾

Sunburst Icicle Tree

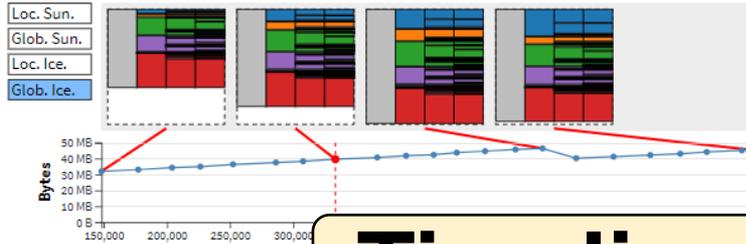
## Settings

Overall 100.0%



# PROTOTYPE AVAILABLE

# Time navigation



# Timeline

Current Heap: Number: 7 | Time: 333,460ms | Size: 39,603,176

Navigation controls: Previous heap, Next heap

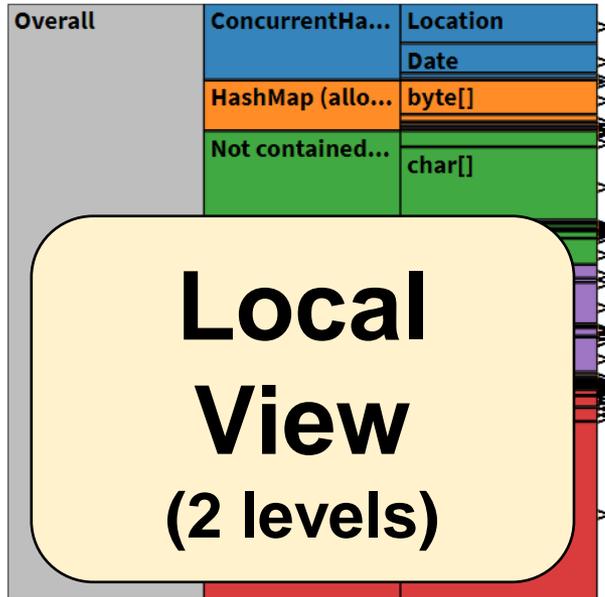
Displayed Metric: Bytes  Objects

Sorting Mode: Absolute Growth ▾

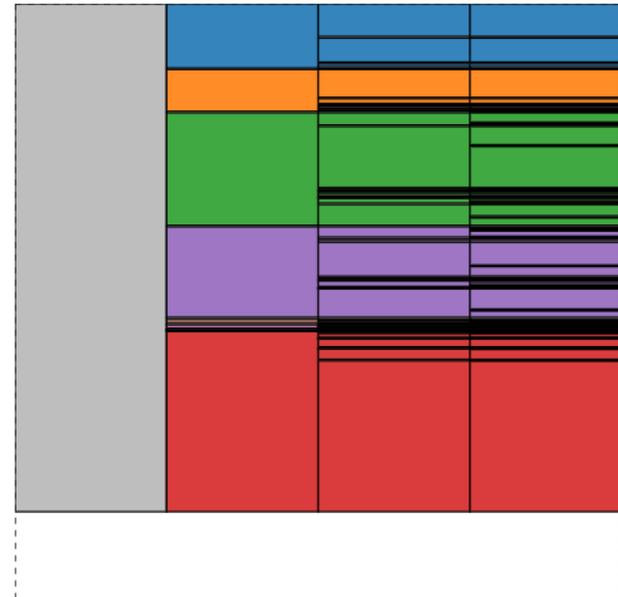
View options: Sunburst, Icicle, Tree

# Settings

Overall 100.0%

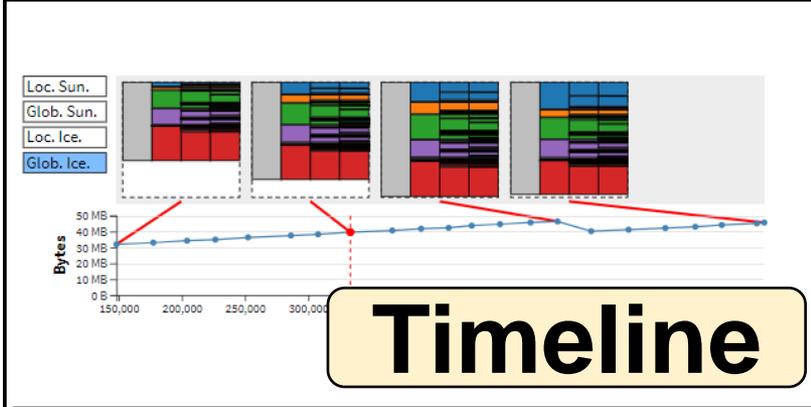


# Local View (2 levels)



# PROTOTYPE AVAILABLE

## Time navigation



Current Heap: Number: 7 | Time: 333,460ms | Size: 39,603,176

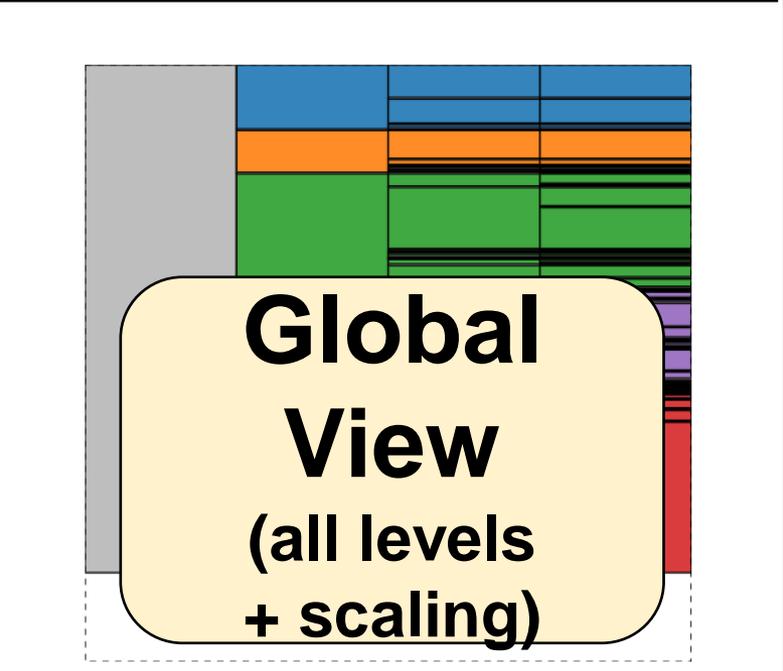
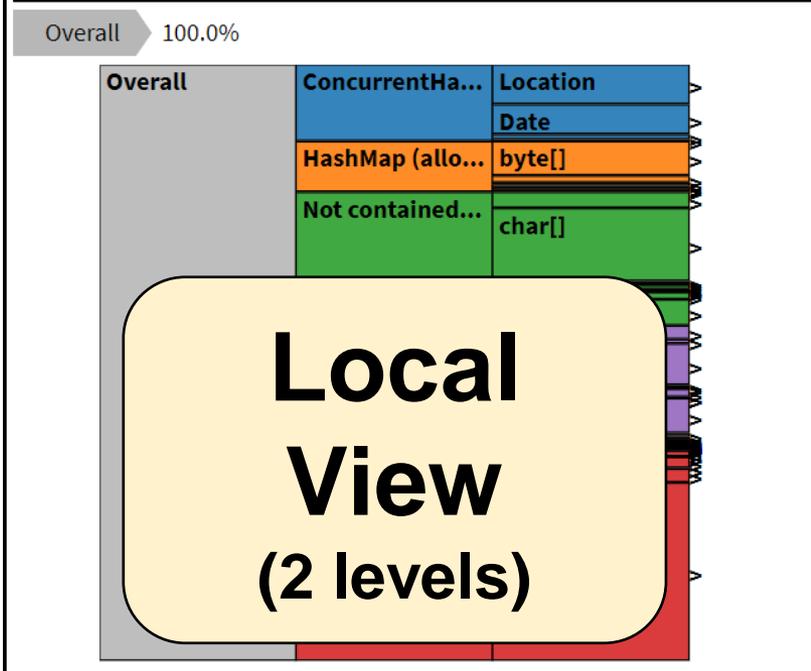
Previous heap Next heap

Displayed Metric: Bytes  Objects

Sorting Mode: Absolute Growth ▾

Sunburst Icicle Tree

**Settings**



# FUTURE WORK

Icons taken from <https://fontawesome.com/> - license: <https://fontawesome.com/license/free>

# FUTURE WORK

- Follow-up work accepted at STAG 2020  
„Memory Leak Analysis using Time-Travel-based and  
Timeline-based Tree Evolution Visualizations “

**STAG 2020**  
SMART TOOLS AND APPLICATIONS FOR GRAPHICS

# FUTURE WORK

- Follow-up work accepted at STAG 2020  
„Memory Leak Analysis using Time-Travel-based and  
Timeline-based Tree Evolution Visualizations “

- Guidance



Icons taken from <https://fontawesome.com/> - license: <https://fontawesome.com/license/free>

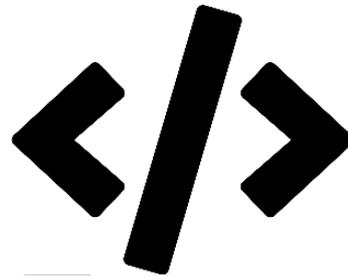
# FUTURE WORK

- Follow-up work accepted at STAG 2020  
„Memory Leak Analysis using Time-Travel-based and  
Timeline-based Tree Evolution Visualizations “

- Guidance



- IDE Integration



Icons taken from <https://fontawesome.com/> - license: <https://fontawesome.com/license/free>

# TAKE-AWAYS

# TAKE-AWAYS

## Problem

**How to visualize  
memory evolution  
(i.e., memory  
trees) over time?**

# TAKE-AWAYS

**Problem**

**How to visualize  
memory evolution  
(i.e., memory  
trees) over time?**

**Tree  
Visualizations**

**Target audience:  
Novice memory  
analysts**

**Sunburst**

**Icicle**

# TAKE-AWAYS

## Problem

How to visualize  
memory evolution  
(i.e., memory  
trees) over time?

## Tree Visualizations

Target audience:  
Novice memory  
analysts

Sunburst

Icicle

## Evolution Over Time

Stable layout

„Relative position  
animation“

Don't destroy  
user's  
cognitive map

# TAKE-AWAYS

**Problem**

How to visualize  
memory evolution  
(i.e., memory  
trees) over time?

**Tree  
Visualizations**

Target audience:  
Novice memory  
analysts

Sunburst

Icicle

**Evolution Over  
Time**

Stable layout

„Relative position  
animation“

Don't destroy  
user's  
cognitive map

**Interaction**

Timeline view

Time navigation

Drill-down

# TAKE-AWAYS

## Problem

How to visualize memory evolution (i.e., memory trees) over time?

## Tree Visualizations

Target audience:  
Novice memory analysts

Sunburst

Icicle

## Evolution Over Time

Stable layout

„Relative position animation“

Don't destroy user's cognitive map

## Interaction

Timeline view

Time navigation

Drill-down



**Prototype available at**

[https://bit.ly/](https://bit.ly/STAG-MemoryTreeVizTool)

STAG-MemoryTreeVizTool



**Video available at**

[https://bit.ly/](https://bit.ly/STAG-MemoryTreeVizVideo)

STAG-MemoryTreeVizVideo

# TAKE-AWAYS

**Problem**

How to visualize memory evolution (i.e., memory trees) over time?

**Tree Visualizations**

Target audience:  
Novice memory analysts

Sunburst

Icicle

**Evolution Over Time**

Stable layout

„Relative position animation“

Don't destroy user's cognitive map

**Interaction**

Timeline view

Time navigation

Drill-down



**Markus Weninger**

Johannes Kepler University  
Linz, Austria

markus.weninger@jku.at



**Prototype available at**

<https://bit.ly/>

STAG-MemoryTreeVizTool



**Video available at**

<https://bit.ly/>

STAG-MemoryTreeVizVideo