Garbage Collection Fundamentals

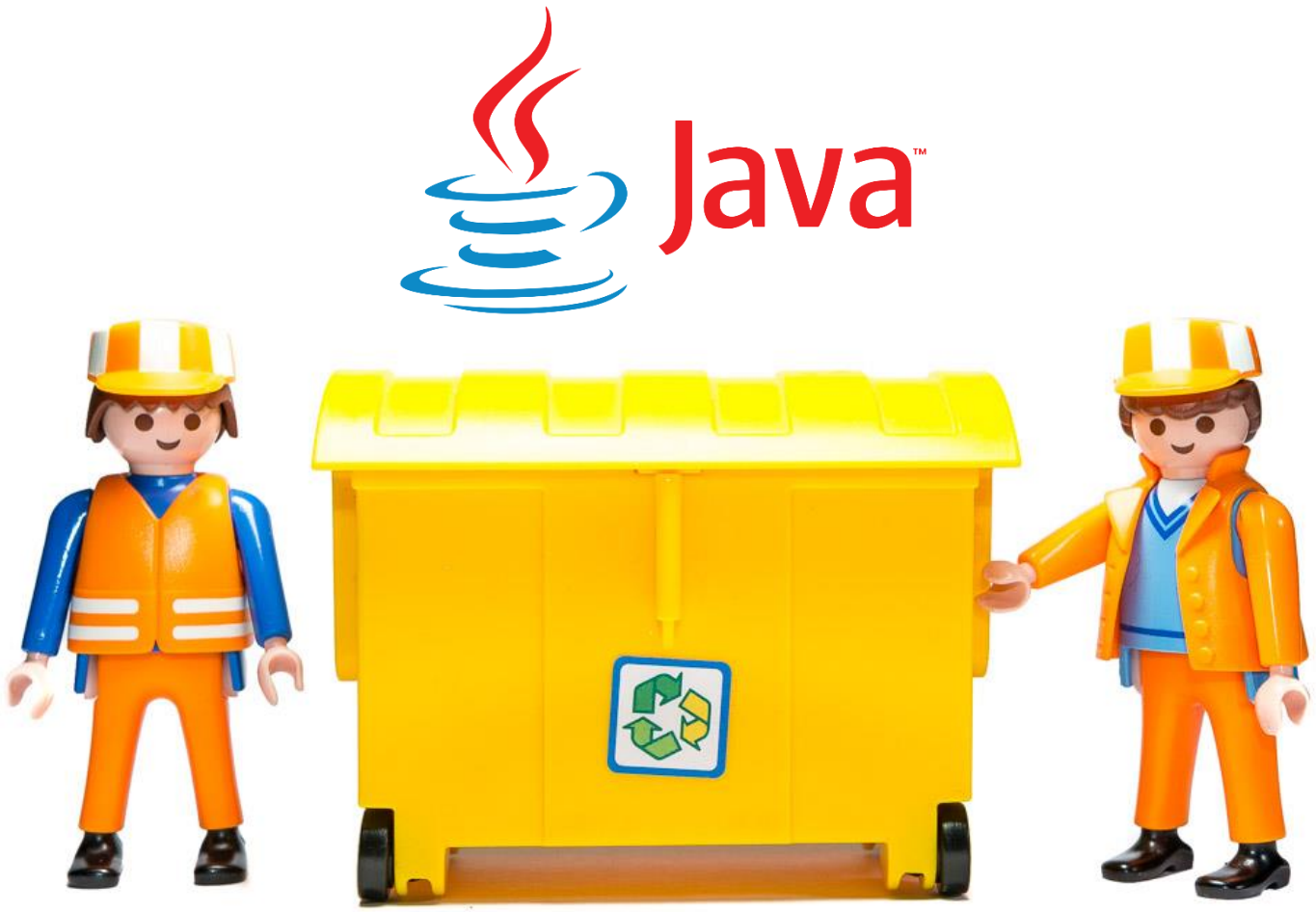
Algorithmen und Applikationen

Michael Aleithe

Agenda

- Einführung
- Algorithmen 
- Messung der GC
- GC-Tuning und Beispiele

so much garbage …

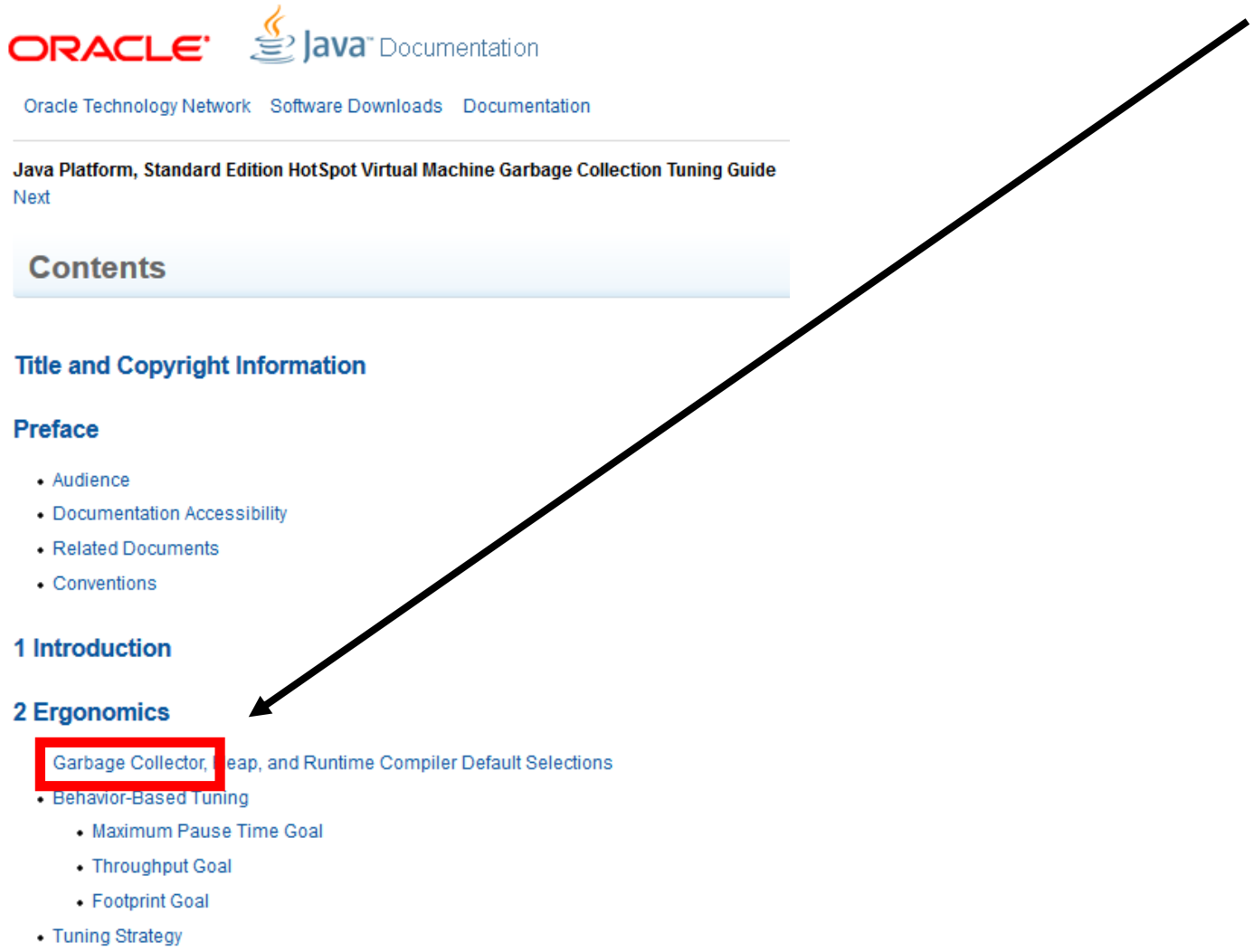


[1, 2]

so much garbage ...

```
6       Object object1 = new Object();
7       Object object2 = new Object();
8       Object object3 = new Object();
```

… Garbage Collection (GC) @ Java

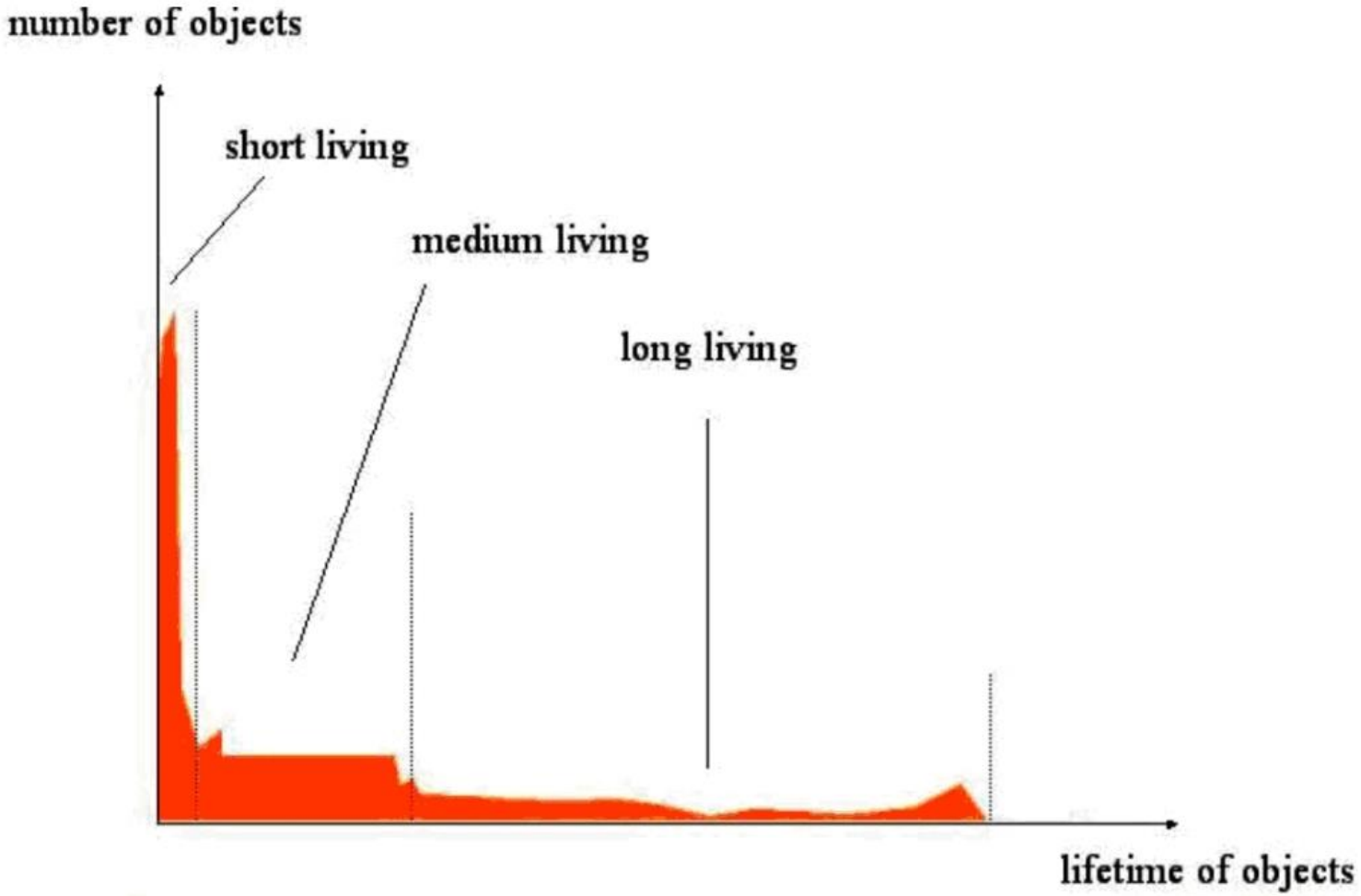


[3]

… but not only in Java!

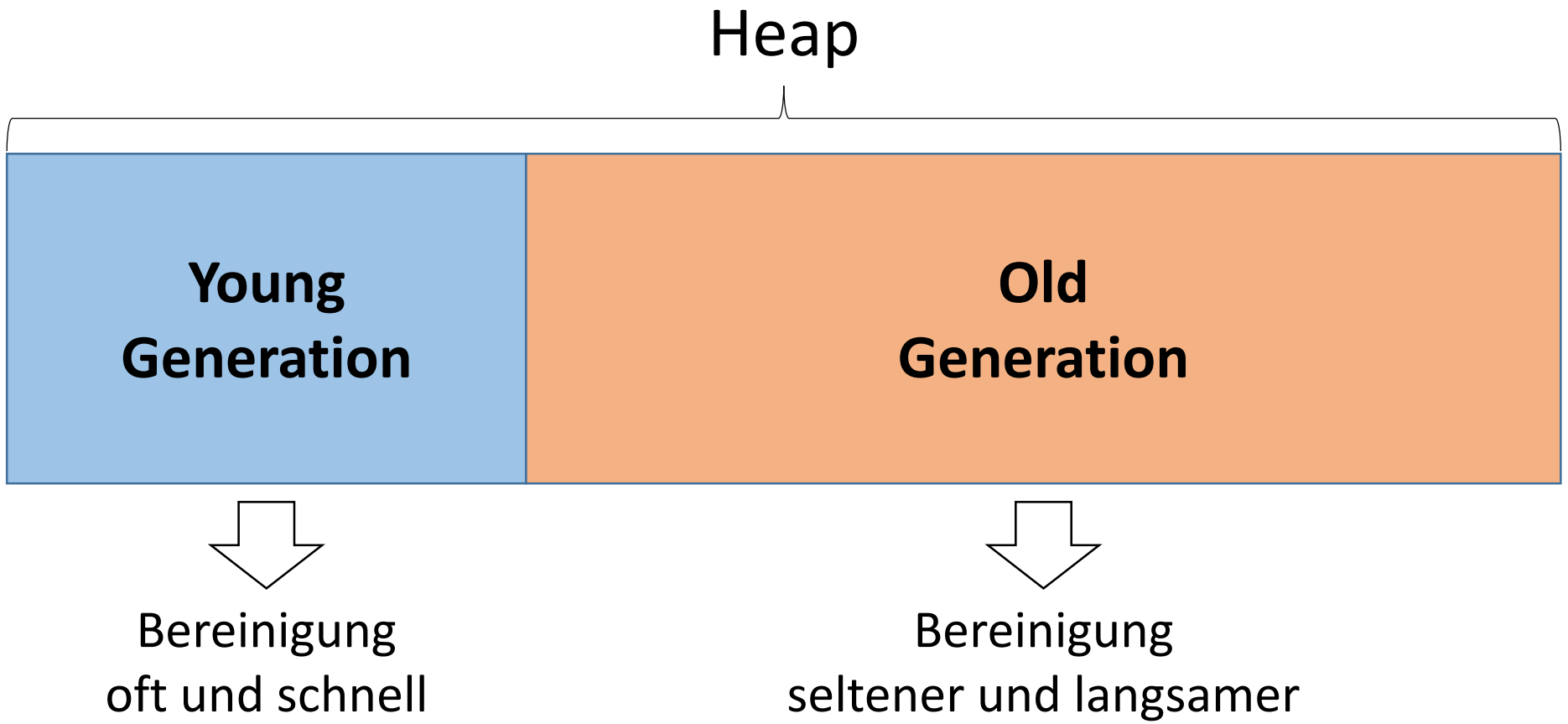
C#

Generational GC

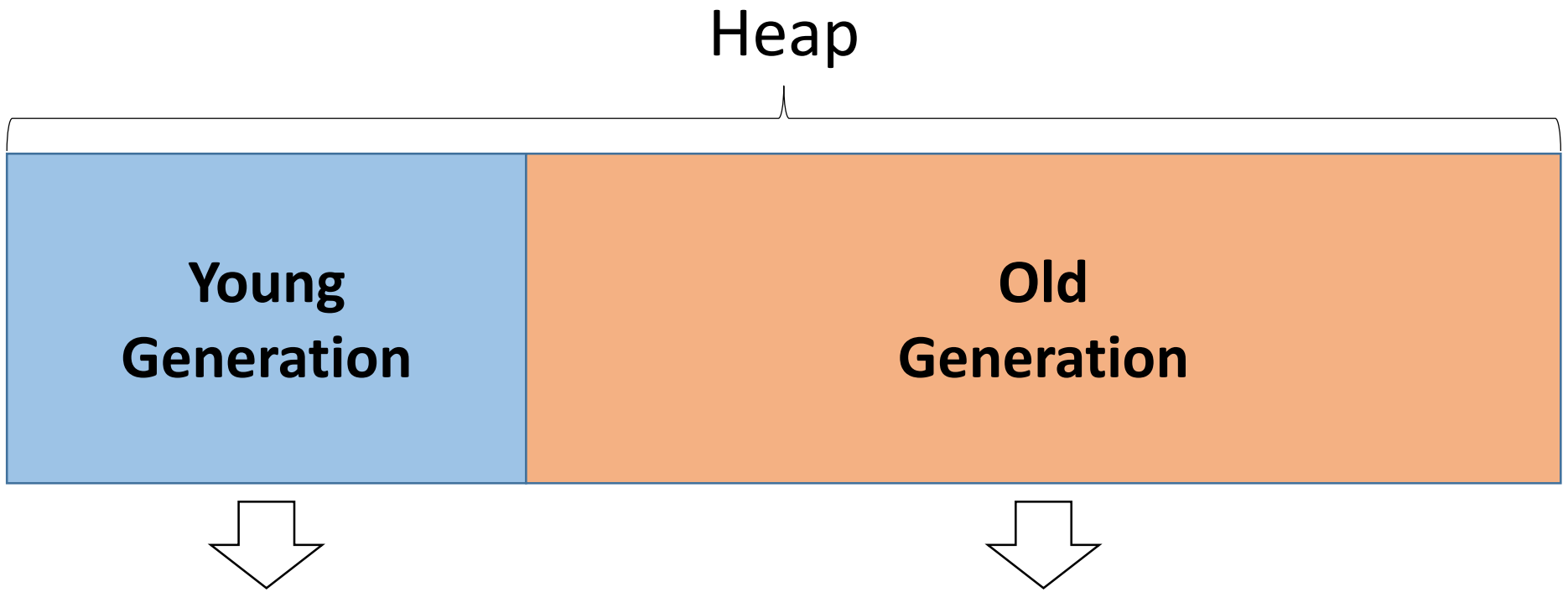


…verschiedene Speicherbereiche

Aufbau des Heap-Speicher

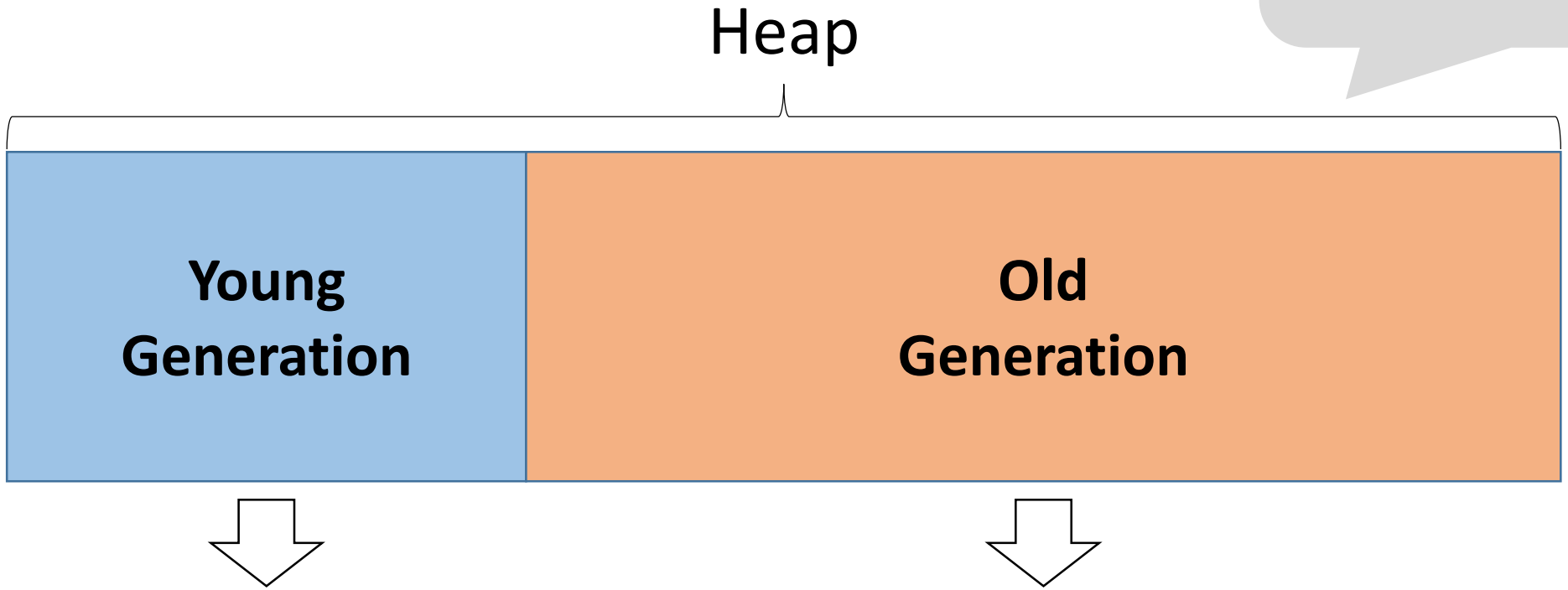


Aufbau des Heap-Speicher

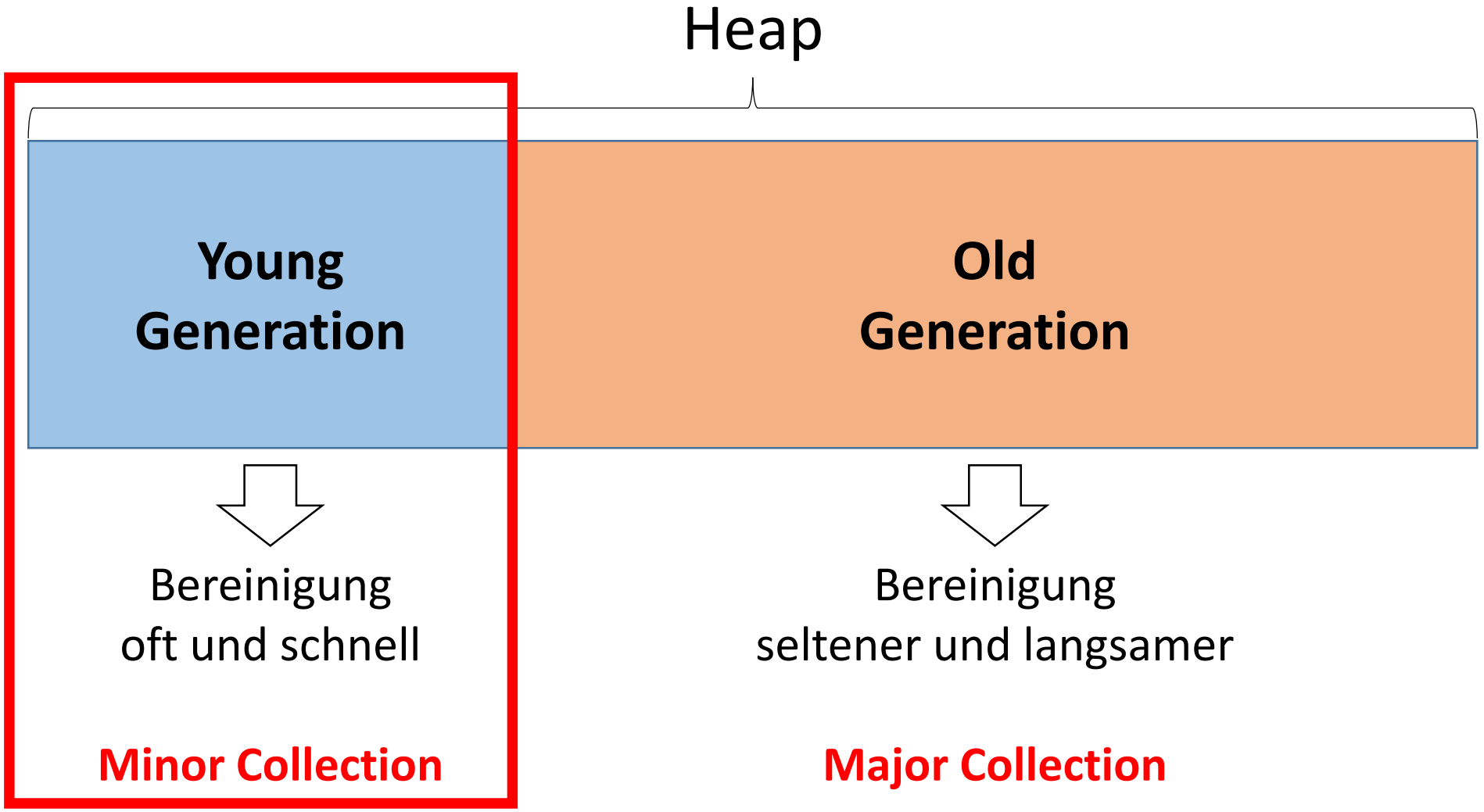


Aufbau des Heap-Speicher

- *deprecated* *
- *verständlicher*
- *analog zu den Erweiterungen*

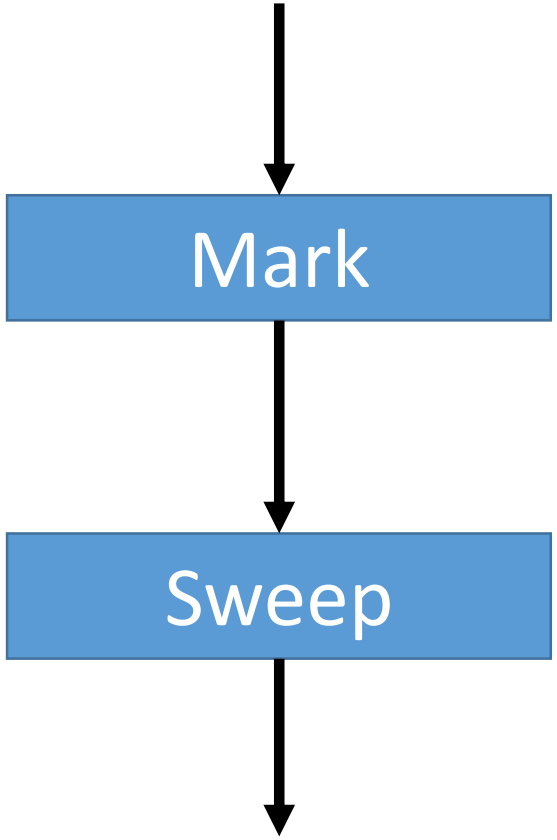


Aufbau des Heap-Speicher



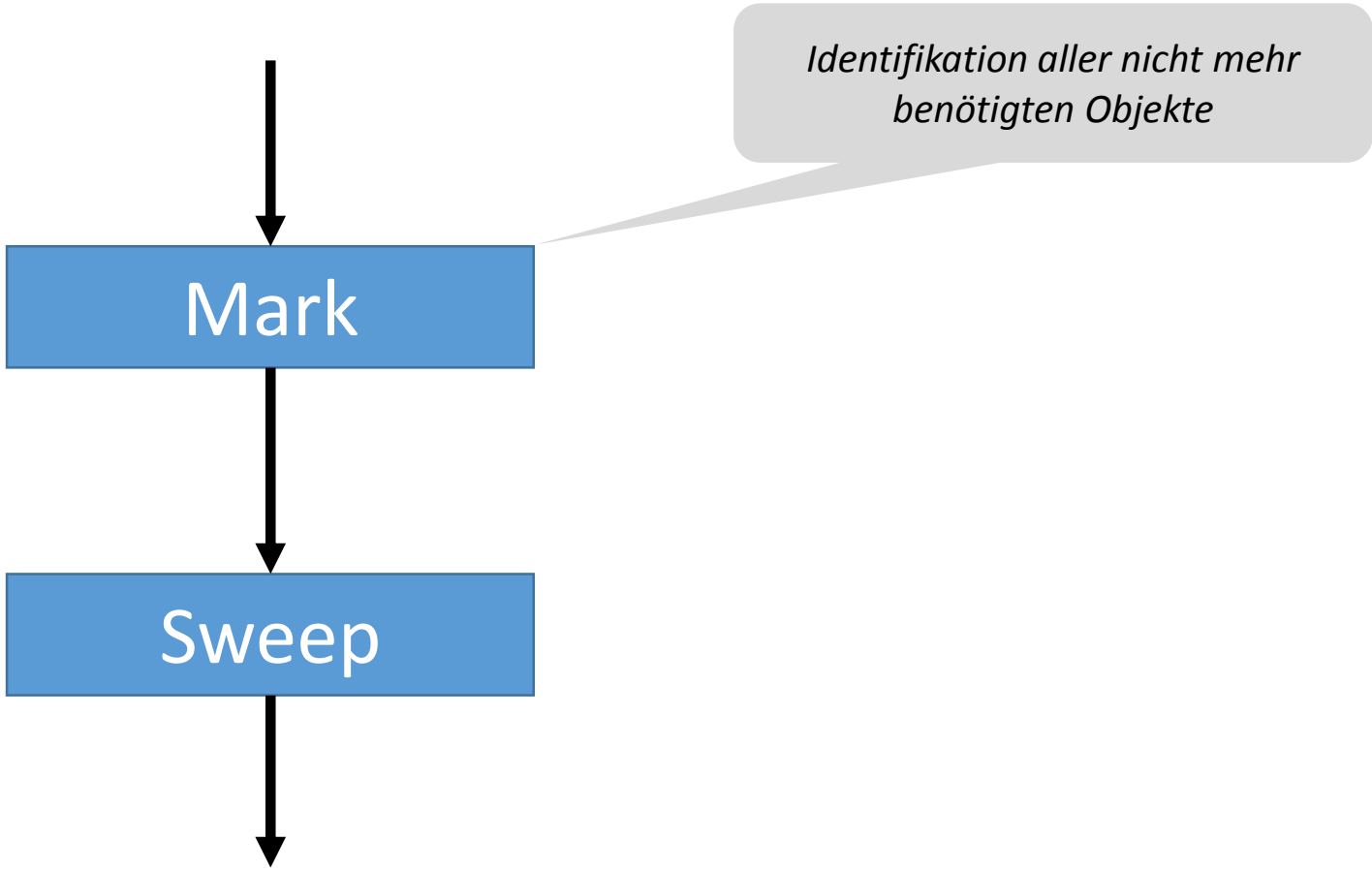
Algorithmen der Young Generation

- **Mark and Sweep**



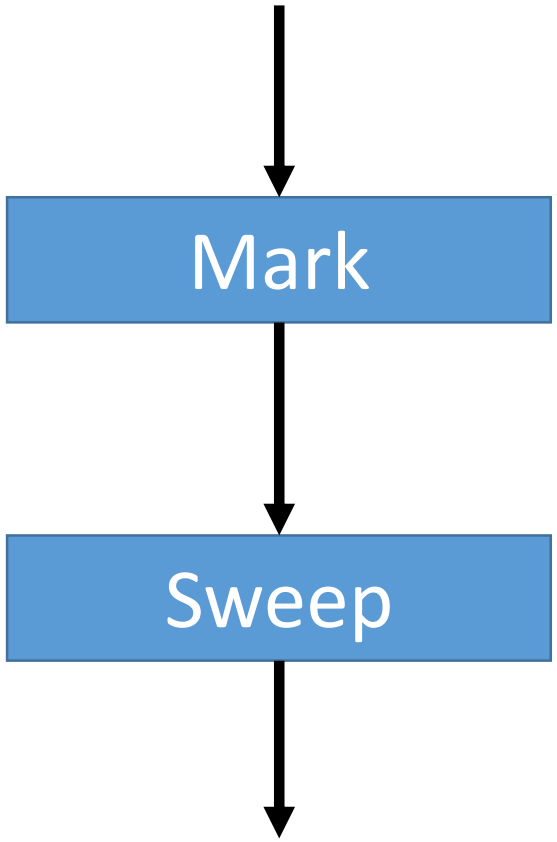
Algorithmen der Young Generation

- **Mark and Sweep**



Algorithmen der Young Generation

- **Mark and Sweep**



Identifikation aller nicht mehr benötigten Objekte

Wie?

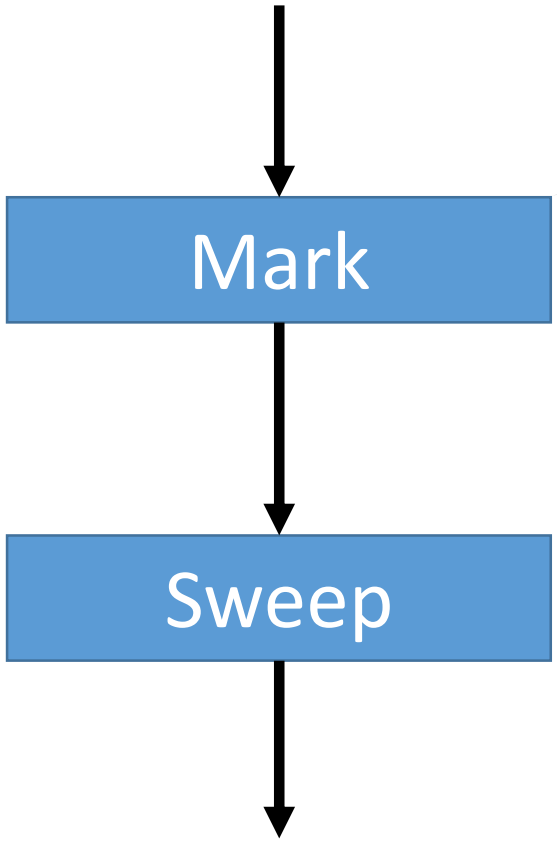
Identifikation aller direkt oder indirekt referenzierbaren Objekte ausgehend von **`rootset`**

`--------`

Rest ist Ergebnis

Algorithmen der Young Generation

- **Mark and Sweep**



Identifikation aller nicht mehr benötigten Objekte

Wie?

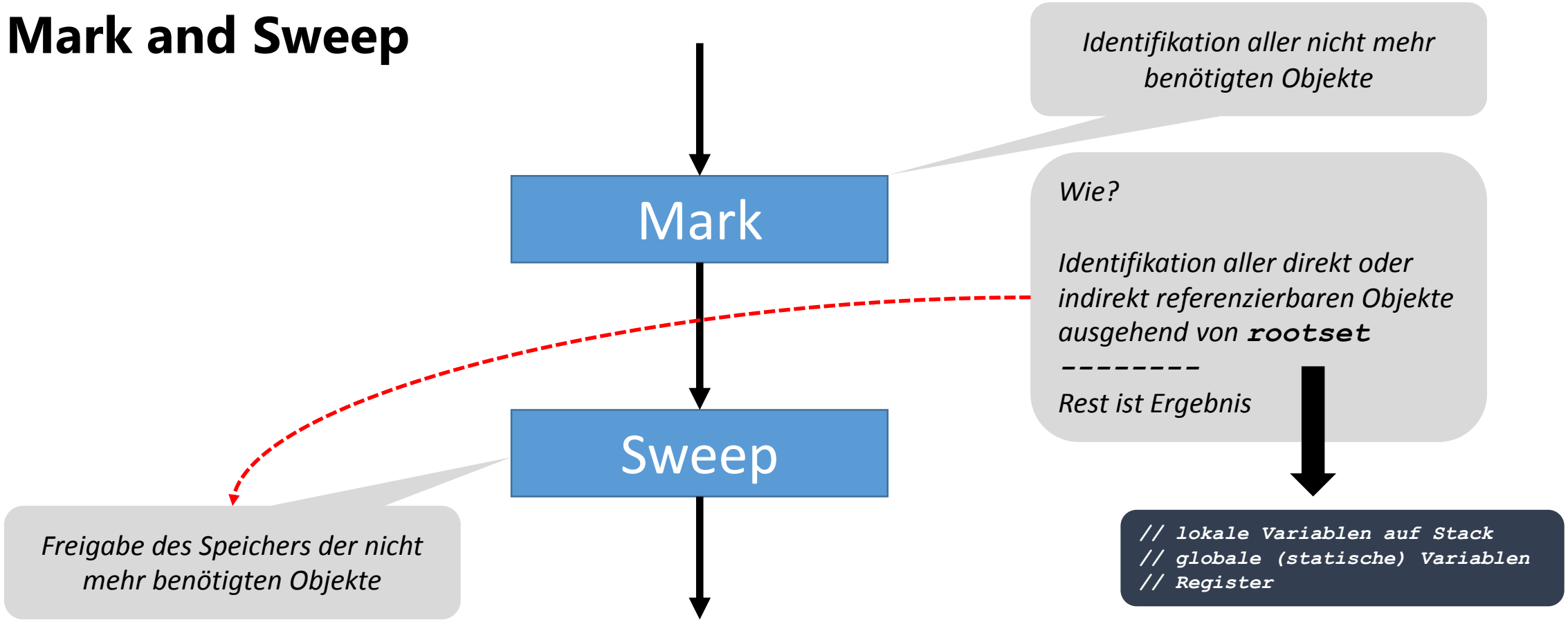
Identifikation aller direkt oder indirekt referenzierbaren Objekte ausgehend von ***rootset***

Rest ist Ergebnis

```
// lokale Variablen auf Stack
// globale (statische) Variablen
// Register
```

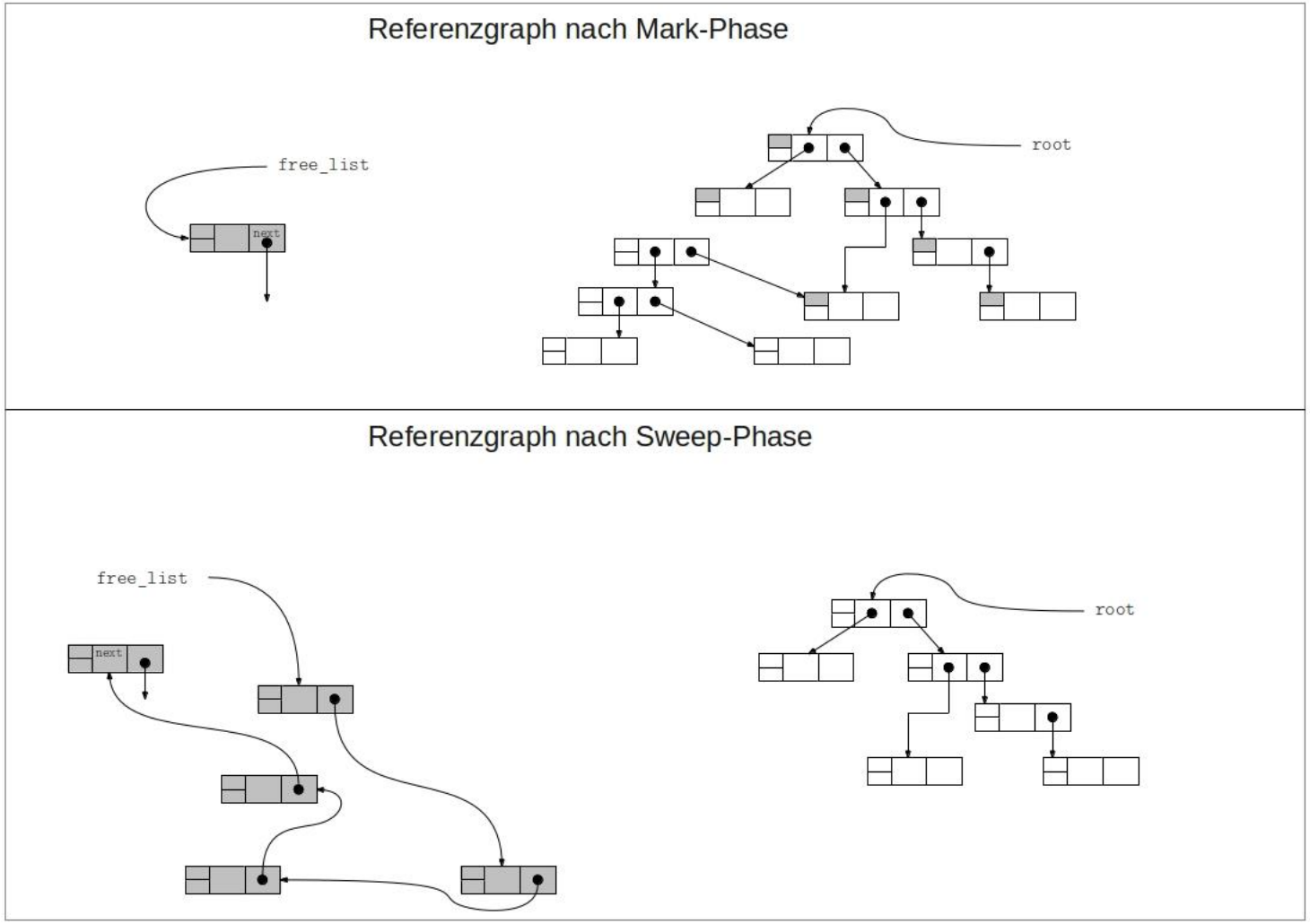
Algorithmen der Young Generation

- **Mark and Sweep**



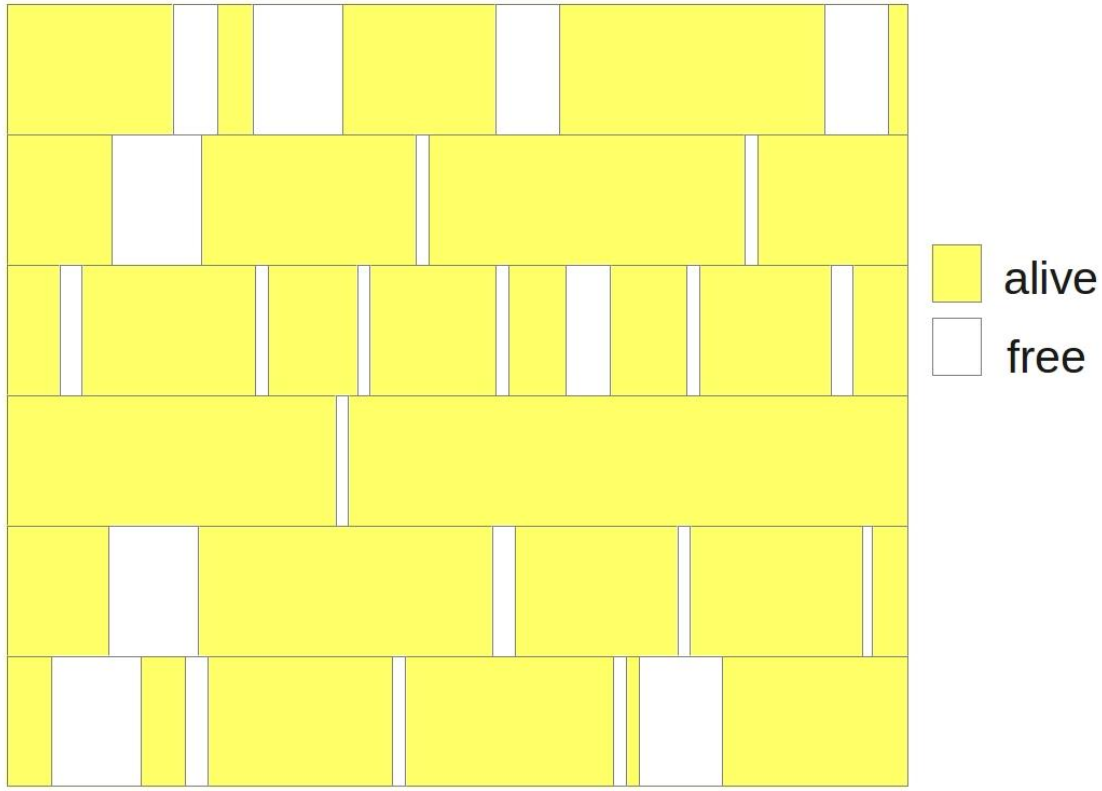
Algorithmen der Young Generation

- **Mark and Sweep**



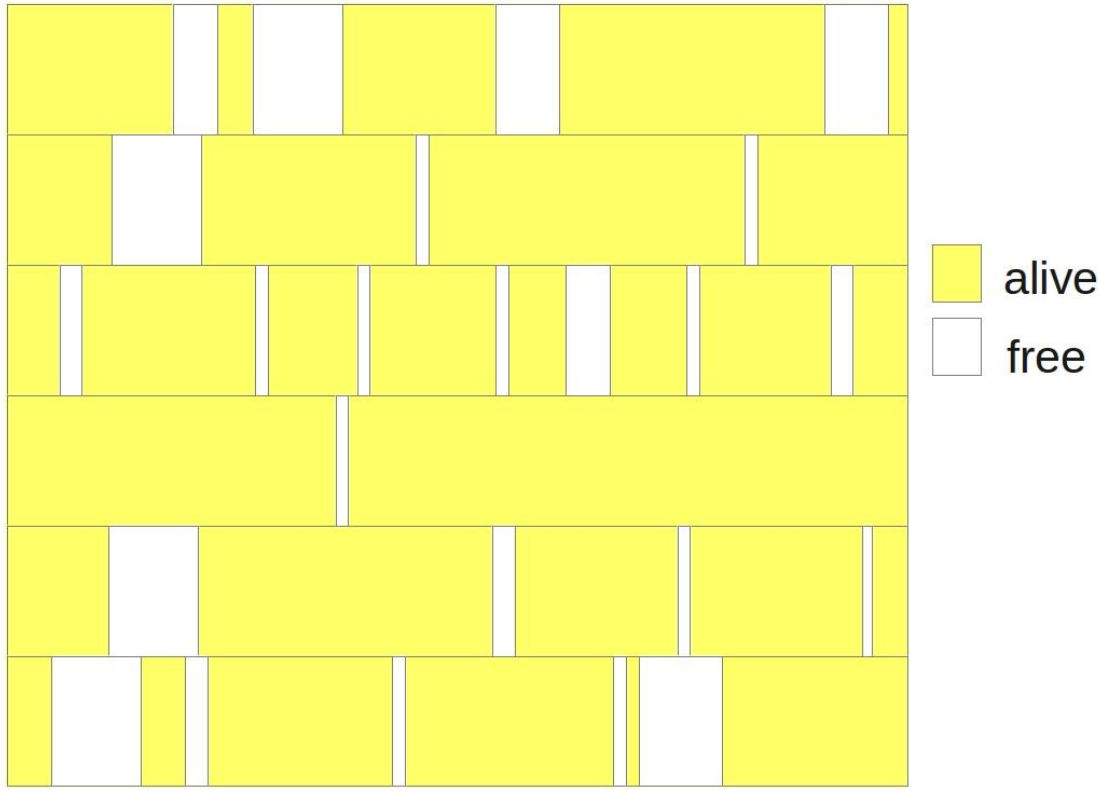
Algorithmen der Young Generation

- **Mark and Sweep**



Algorithmen der Young Generation

- **Mark and Sweep**

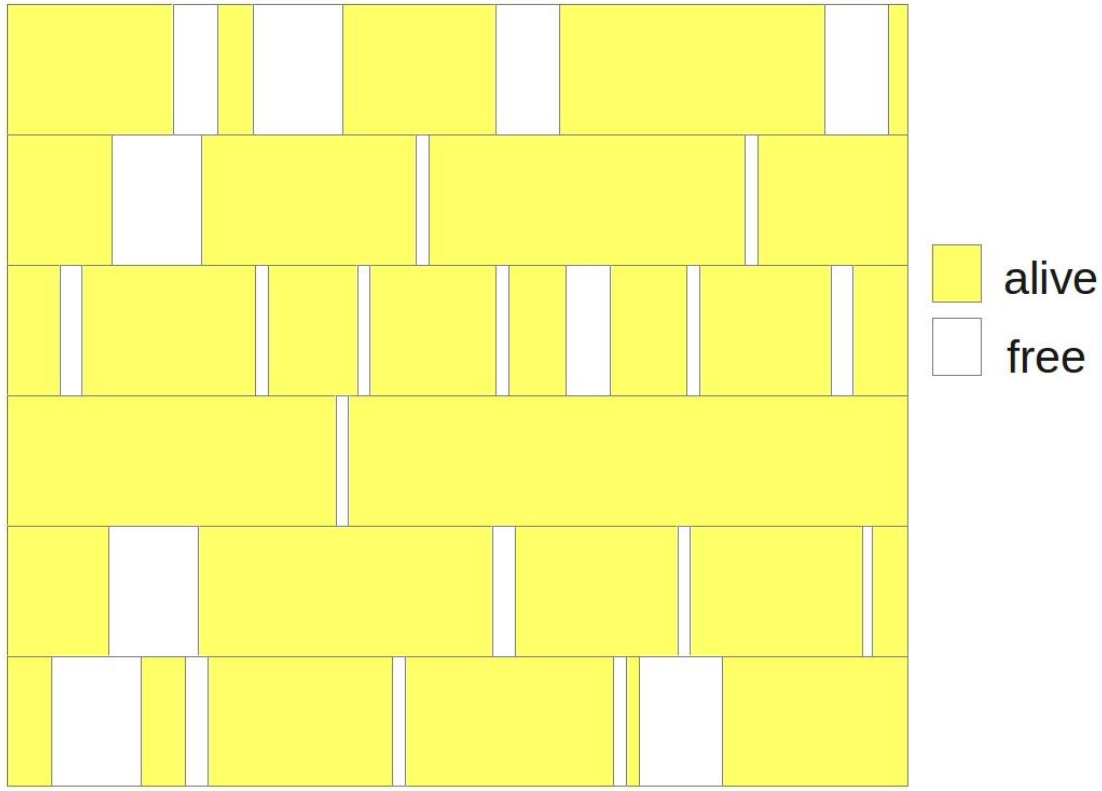


PROBLEM

fragmentierter Heap

Algorithmen der Young Generation

- **Mark and Sweep**



PROBLEM

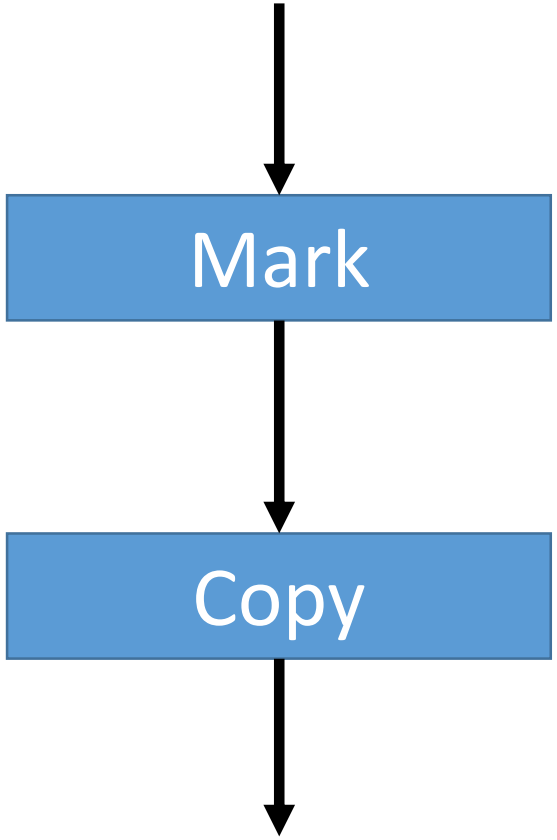
fragmentierter Heap

immer höherer Aufwand zur Bereitstellung von genügend zusammenhängenden Speicher für neue Objekte

Algorithmen der Young Generation

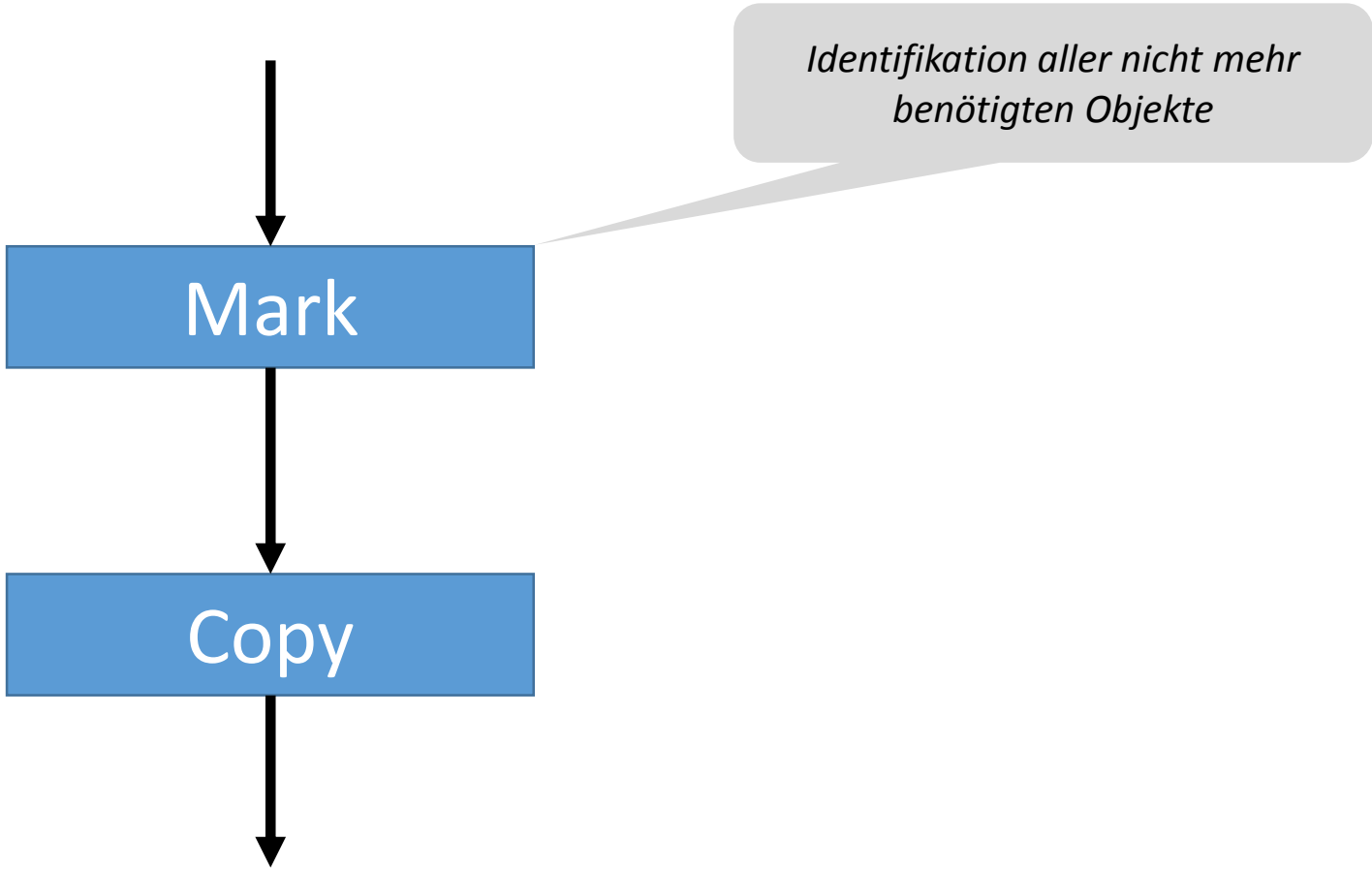
- **Mark and Copy**

  (als Abhilfe)



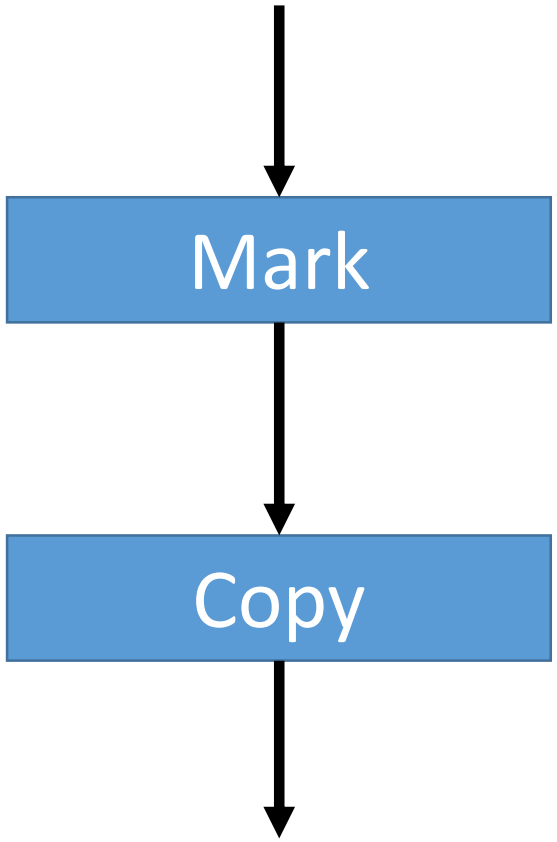
Algorithmen der Young Generation

- **Mark and Copy**



Algorithmen der Young Generation

- **Mark and Copy**



Identifikation aller nicht mehr benötigten Objekte

Wie?

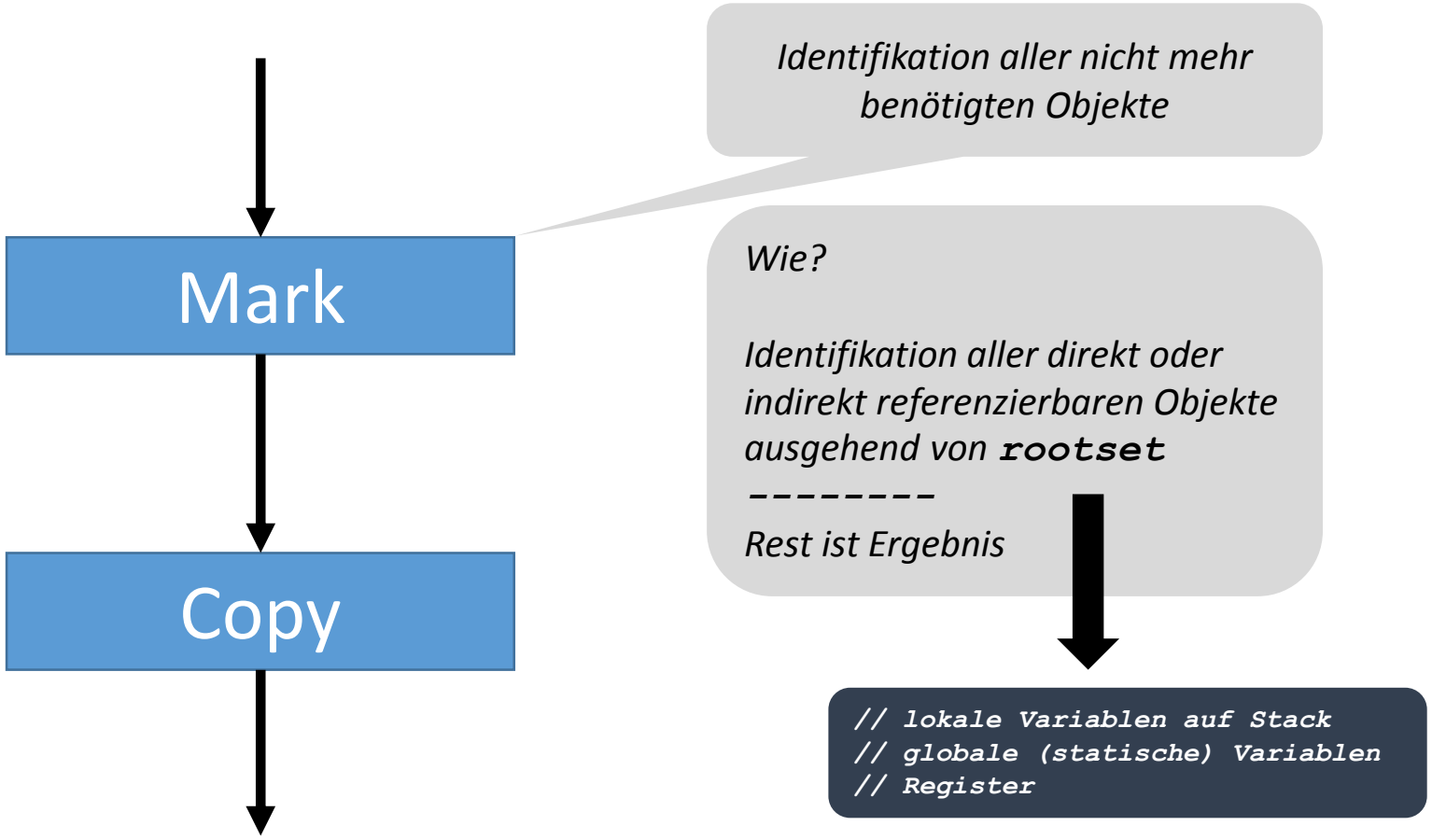
Identifikation aller direkt oder indirekt referenzierbaren Objekte ausgehend von **`rootset`**

`--------`

Rest ist Ergebnis

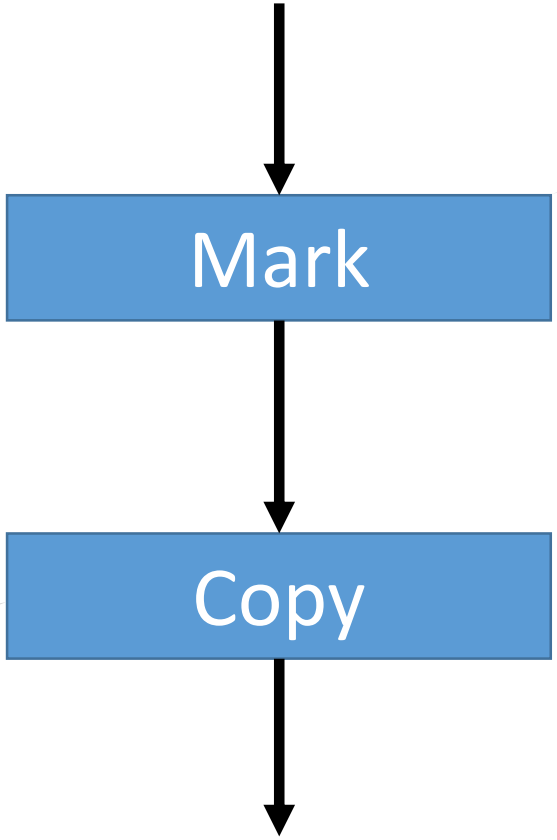
Algorithmen der Young Generation

- **Mark and Copy**



Algorithmen der Young Generation

- **Mark and Copy**



Identifikation aller nicht mehr benötigten Objekte

Wie?

Identifikation aller direkt oder indirekt referenzierbaren Objekte ausgehend von ***`rootset`***

`--------`

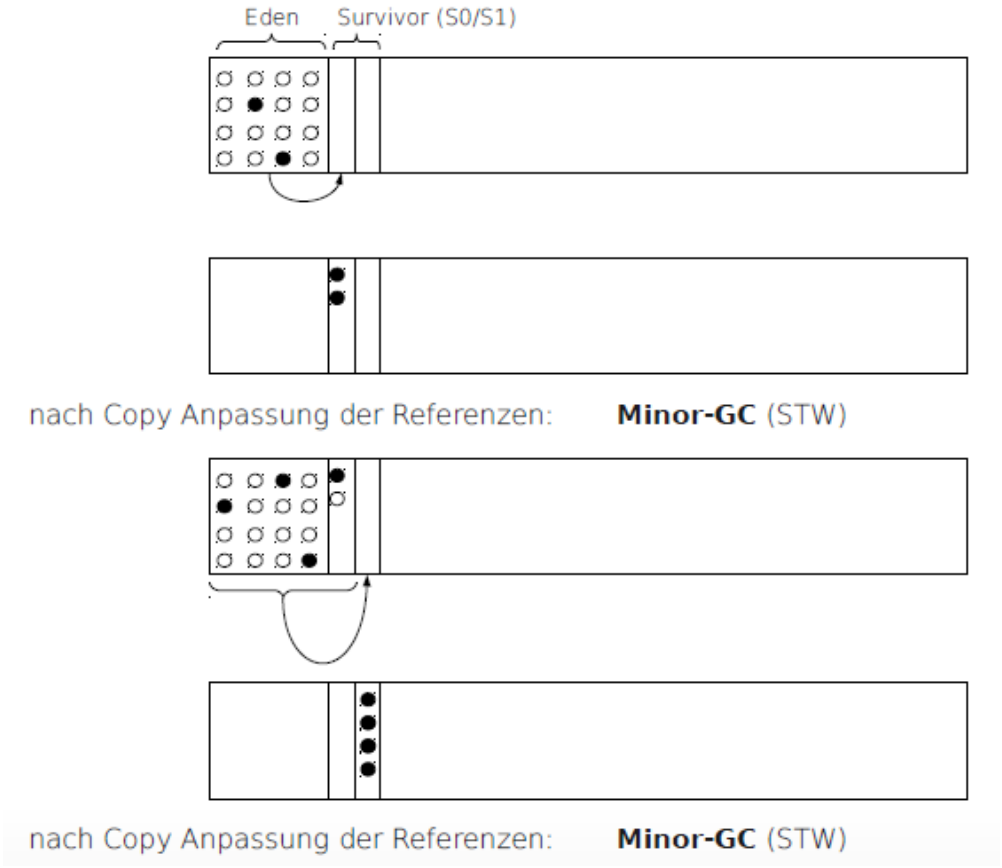
Rest ist Ergebnis

Speicherung in separaten Speicherbereich

```
// lokale Variablen auf Stack
// globale (statische) Variablen
// Register
```

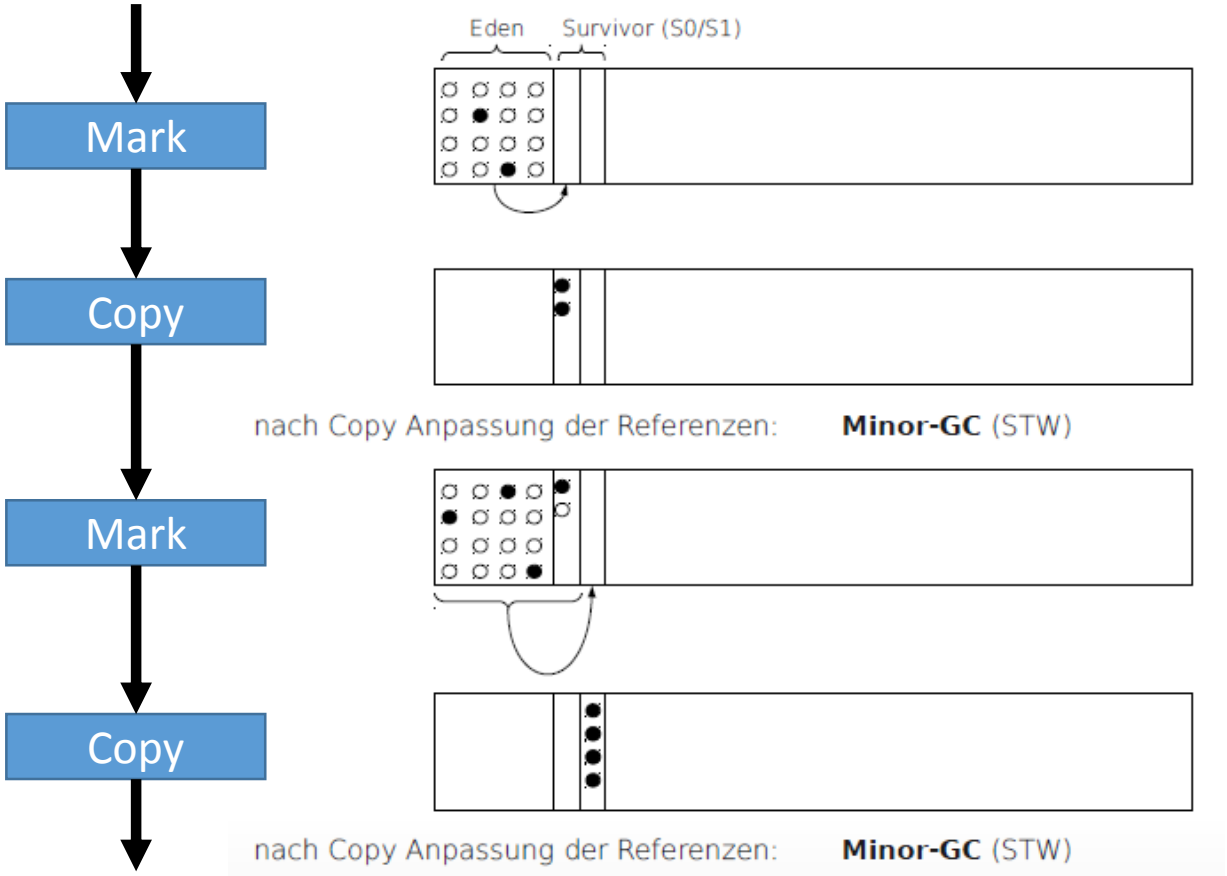
Algorithmen der Young Generation

- **Mark and Copy**



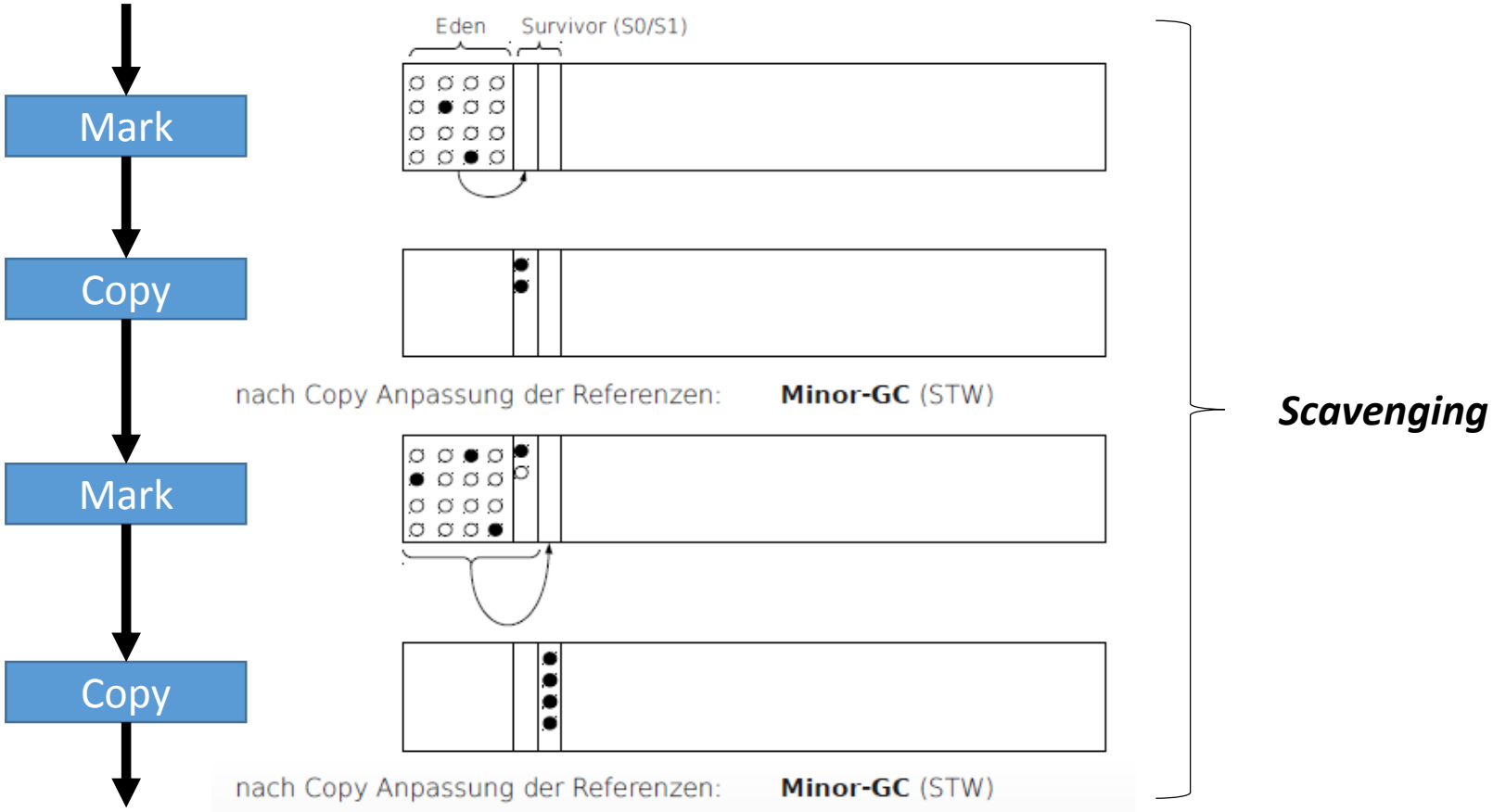
Algorithmen der Young Generation

- **Mark and Copy**



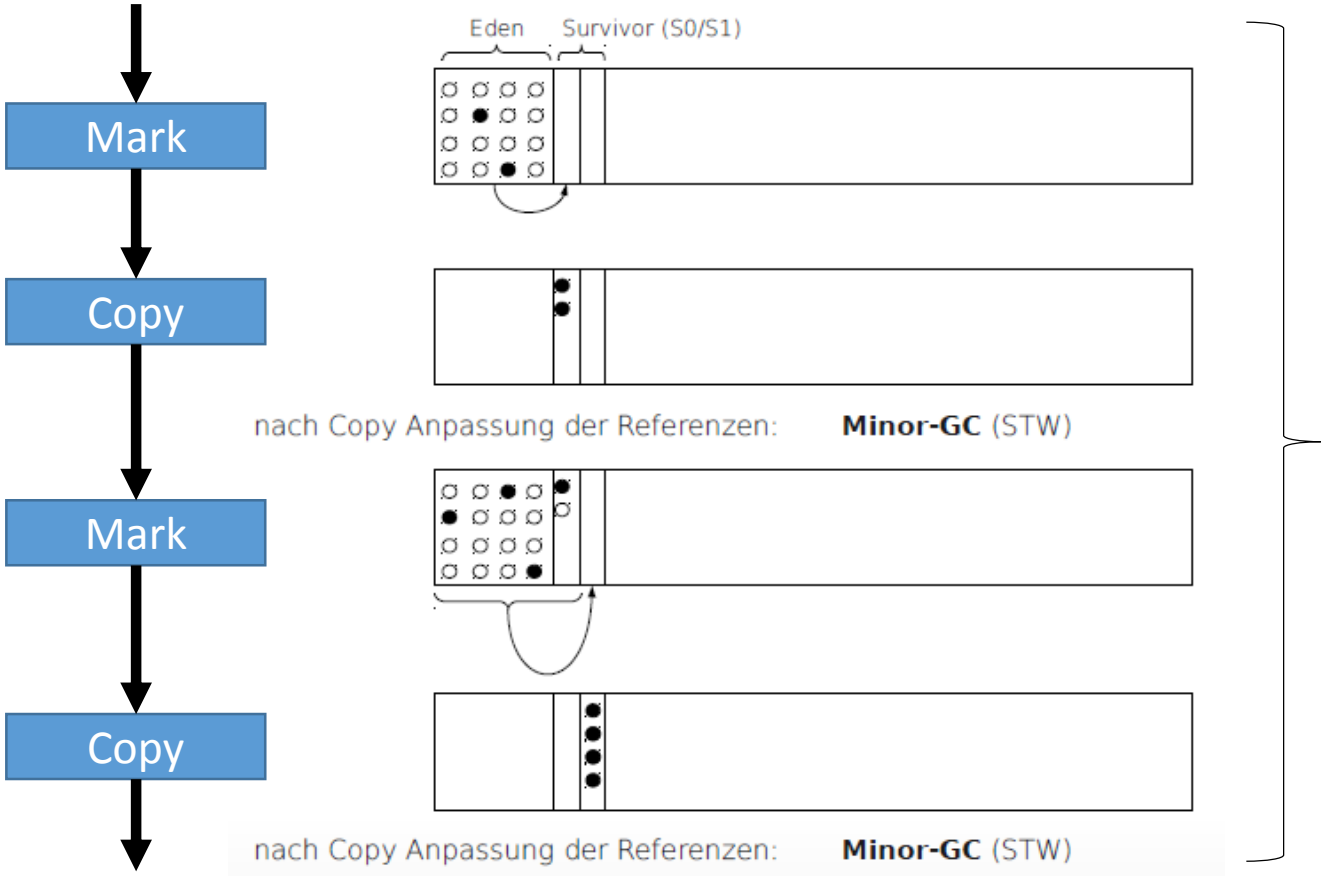
Algorithmen der Young Generation

- **Mark and Copy**



Algorithmen der Young Generation

- **Mark and Copy**



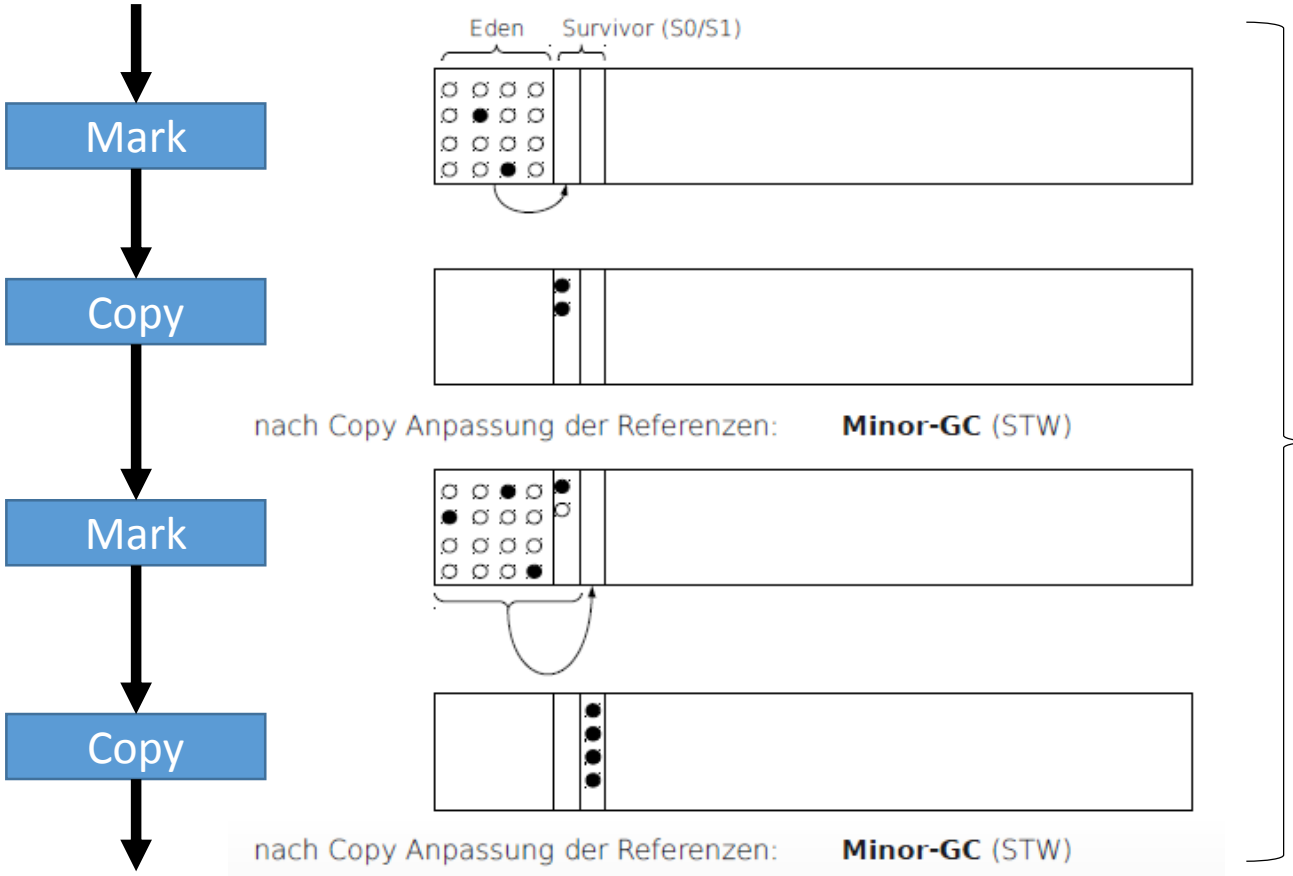
Scavenging

Fragmentierung behoben

ABER:

Algorithmen der Young Generation

- **Mark and Copy**



Scavenging

Fragmentierung behoben

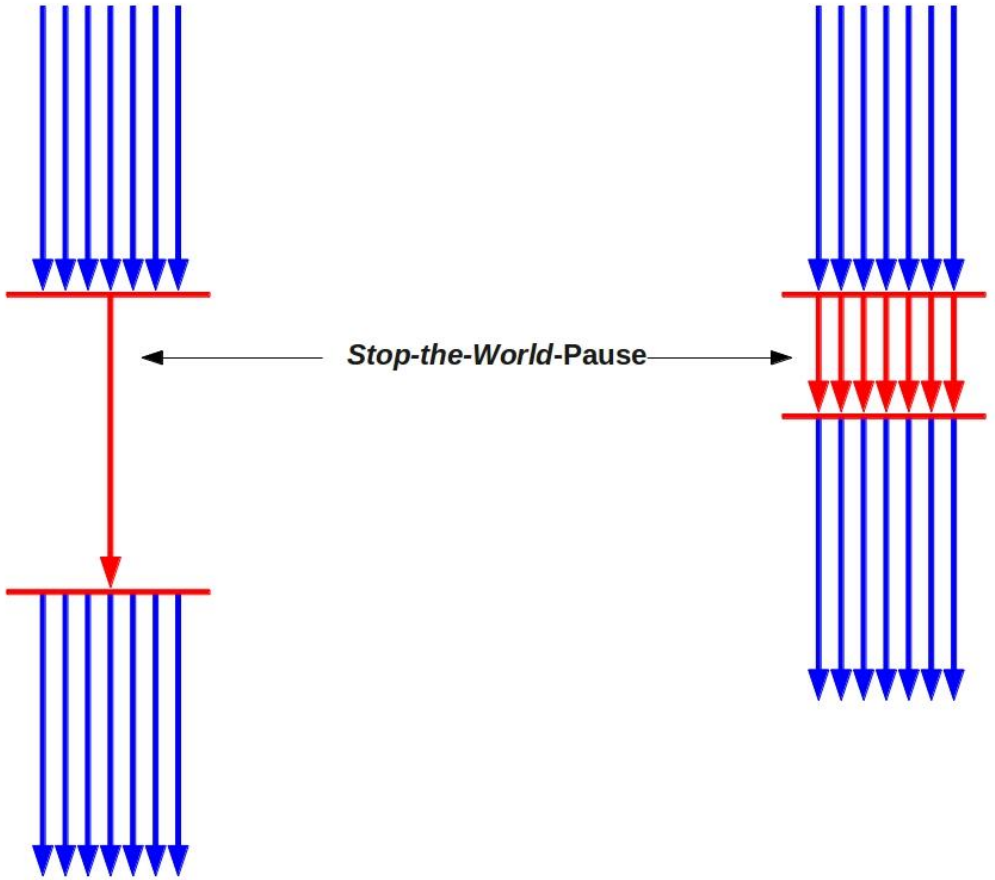
ABER:

Stop-the-World (STW)

```
// Stop der Applikations-Threads

// exklusiver Zugriff des GC
// beim Copy
```

Algorithmen der Young Generation

- **Mark and Copy (Serial vs. Parallel)**



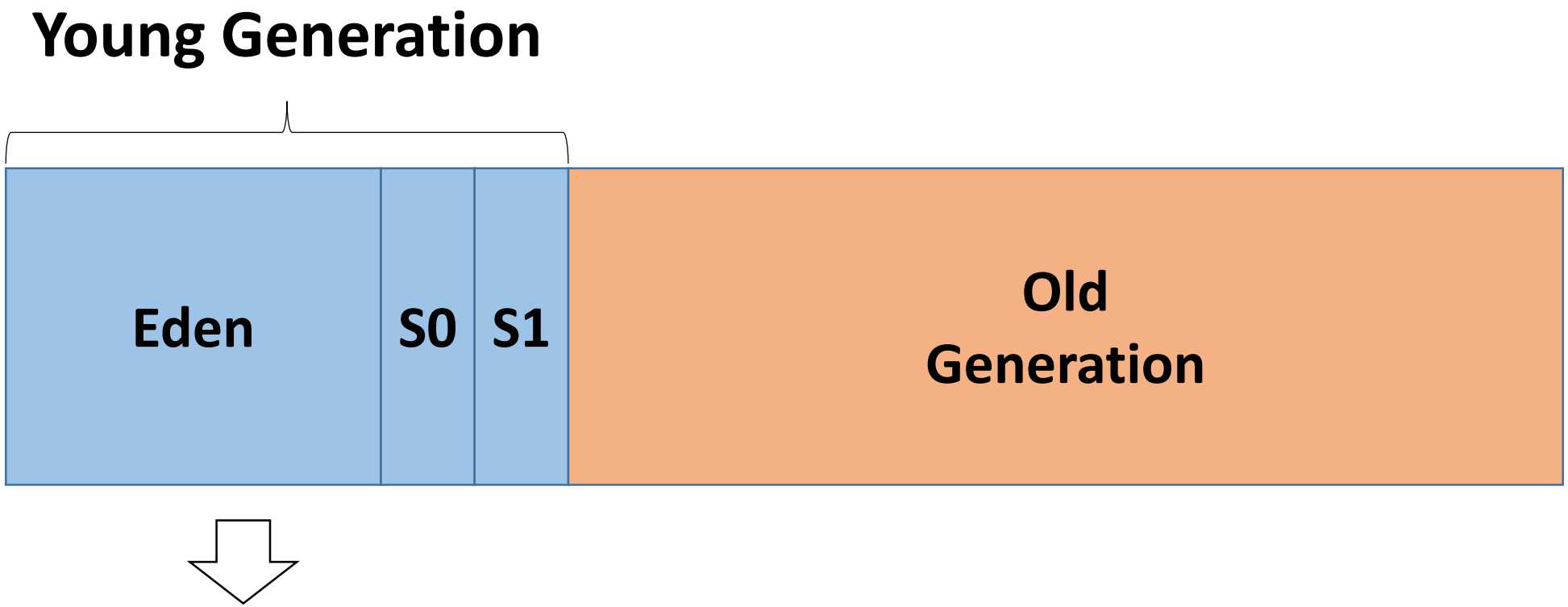
Fragmentierung behoben

ABER:

Stop-the-World (STW)

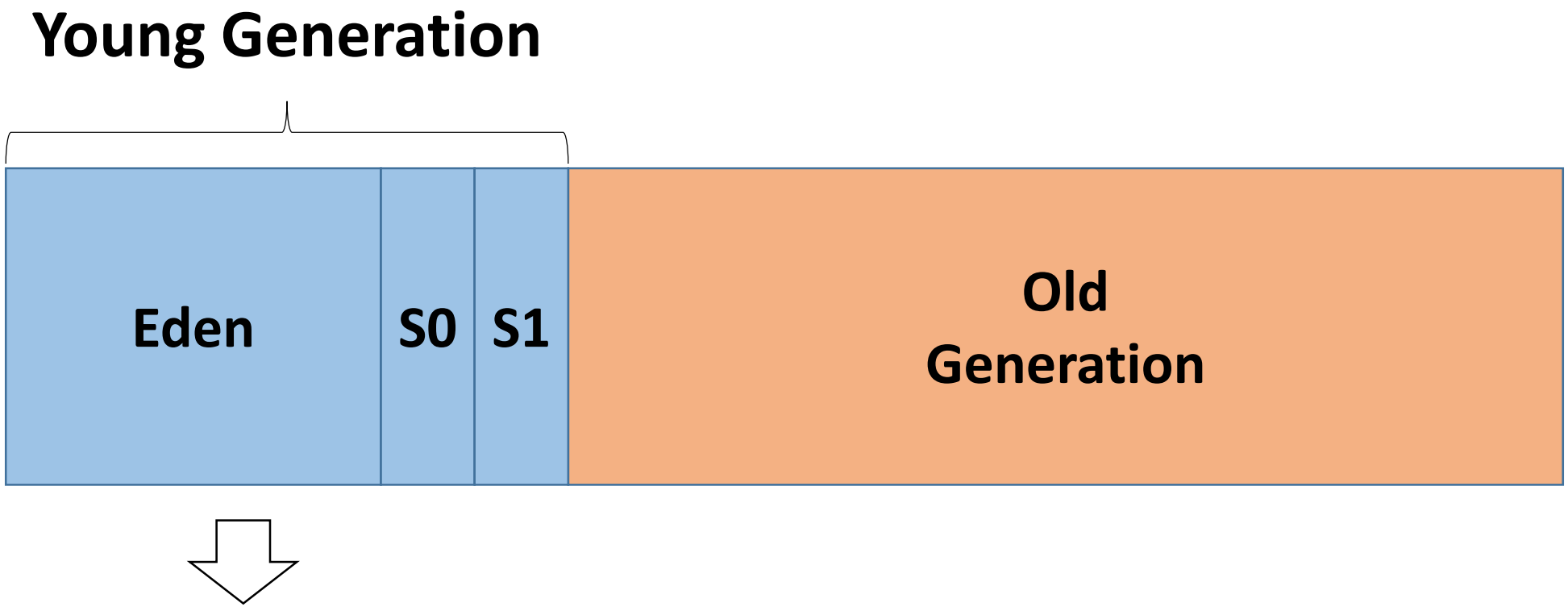
```
// Stop der Applikations-Threads

// exklusiver Zugriff des GC
// beim Copy
```

kurze Gedankenpause



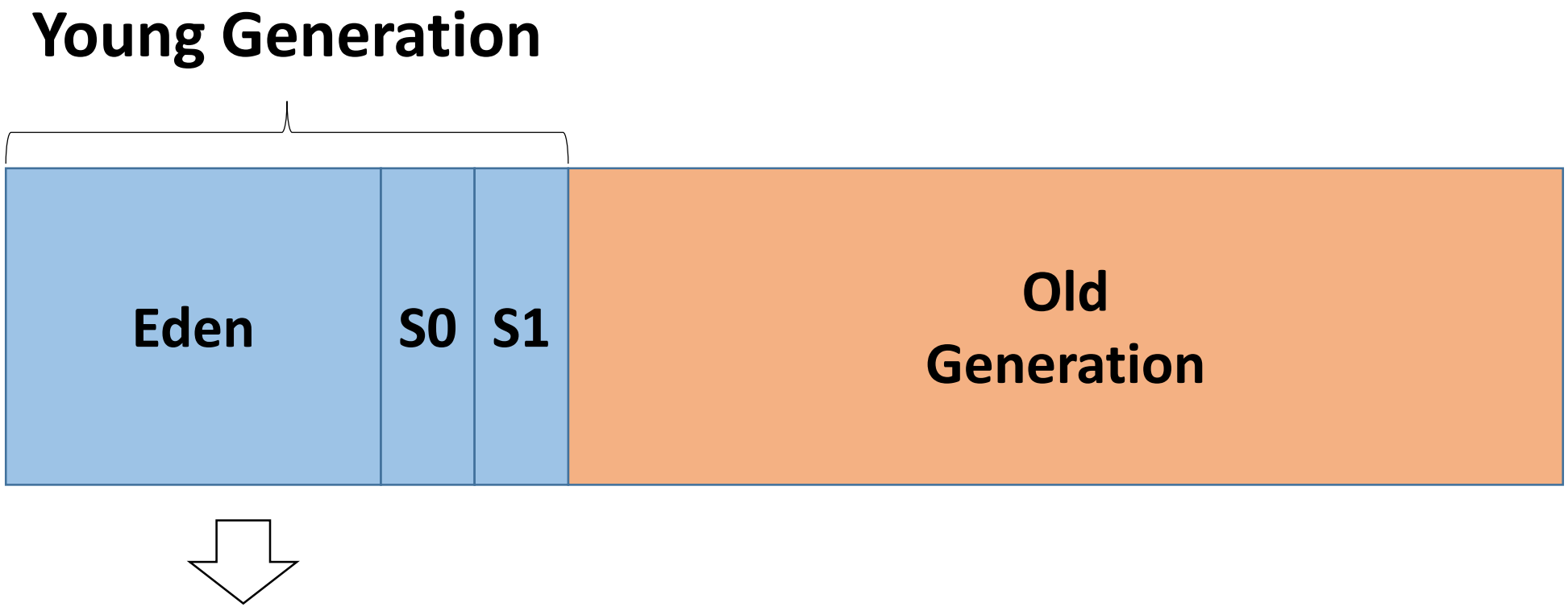
- ***Mark and Sweep***
- ***Serial/Parallel Mark and Copy***

kurze Gedankenpause



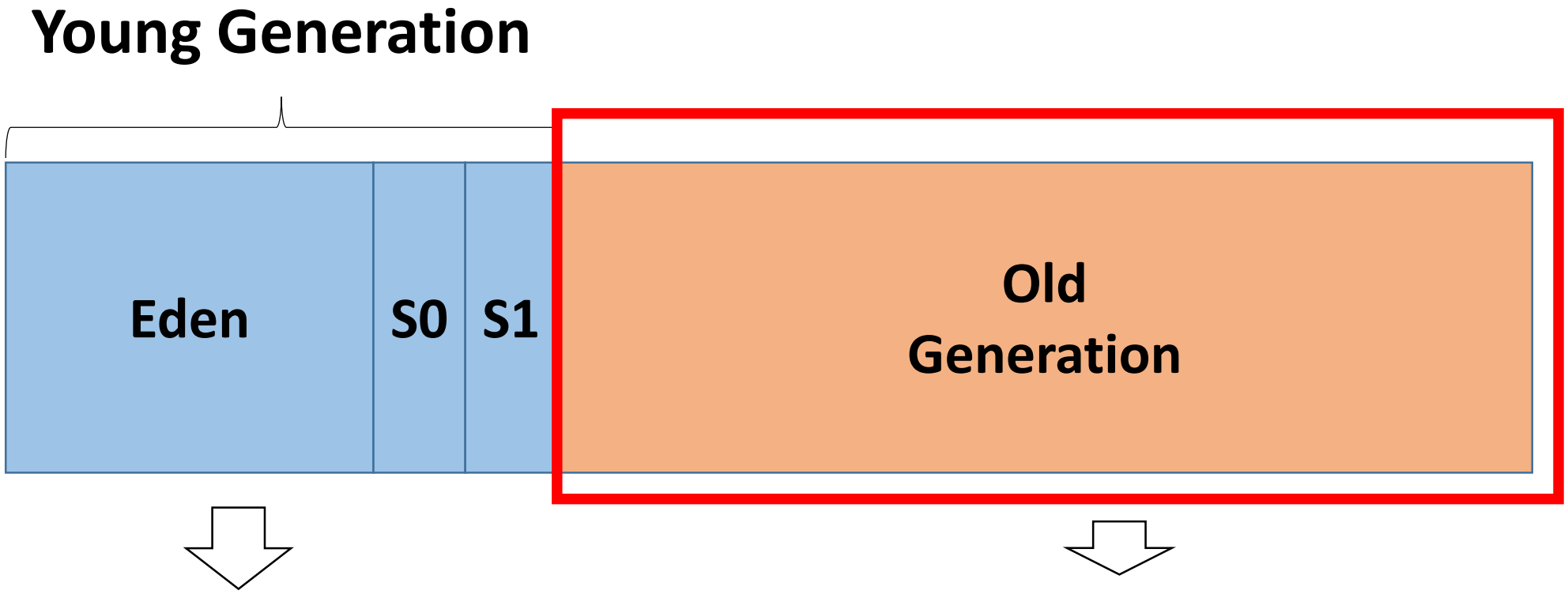
- ***Mark and Sweep*** Fragmentierung
- ***Serial/Parallel Mark and Copy***

kurze Gedankenpause

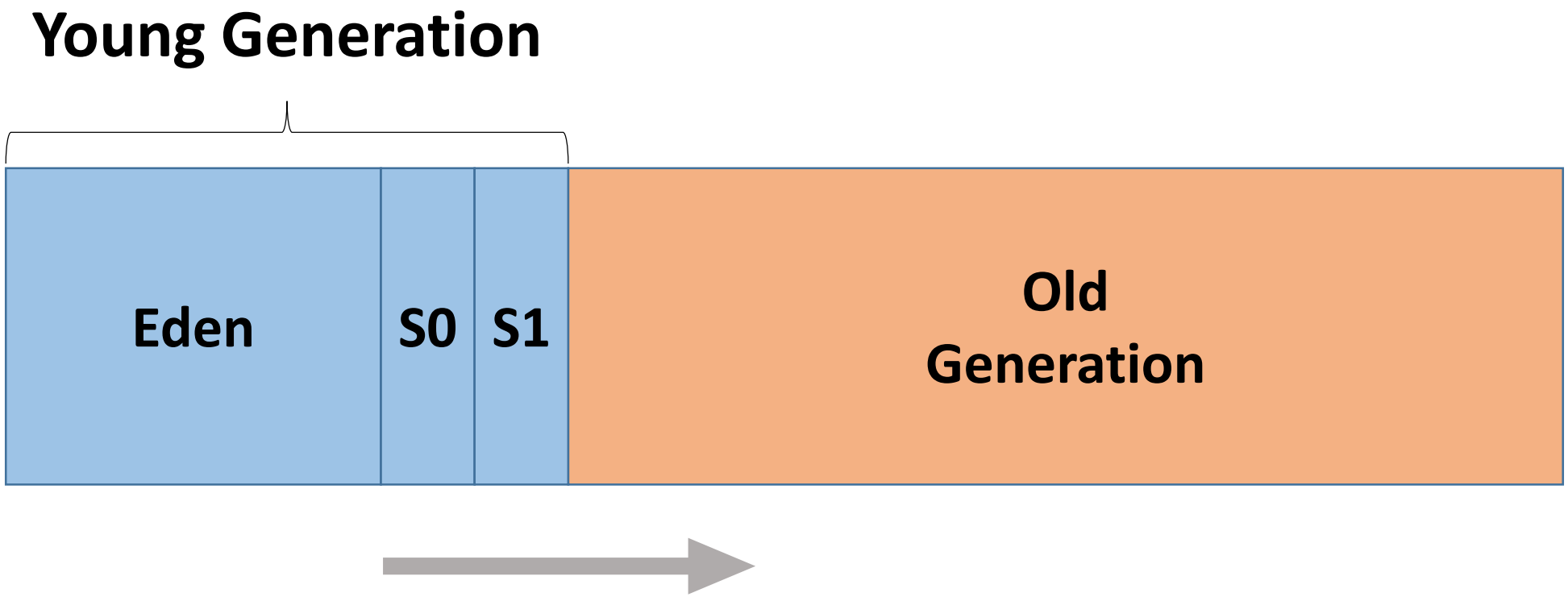


- ***Mark and Sweep*** — Fragmentierung
- ***Serial/Parallel Mark and Copy*** — Stop-the-World (STW)

Algorithmen der Old Generation



Algorithmen der Old Generation

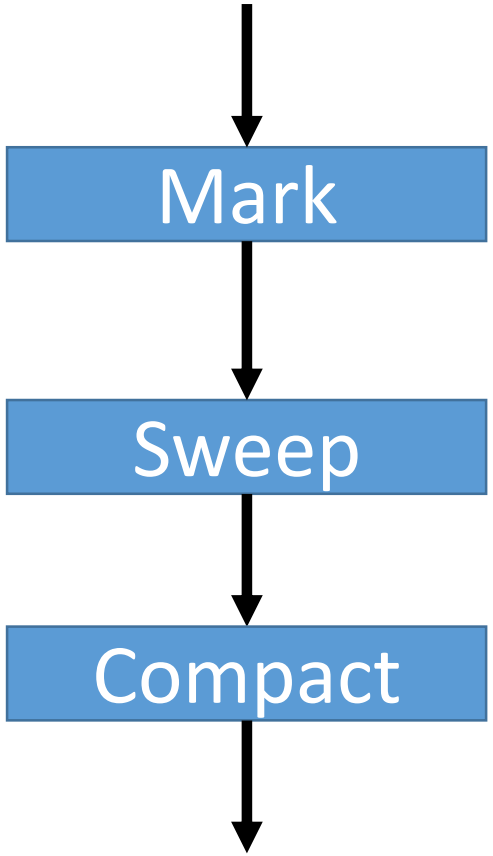


Übergang von Young in Old

Promotion

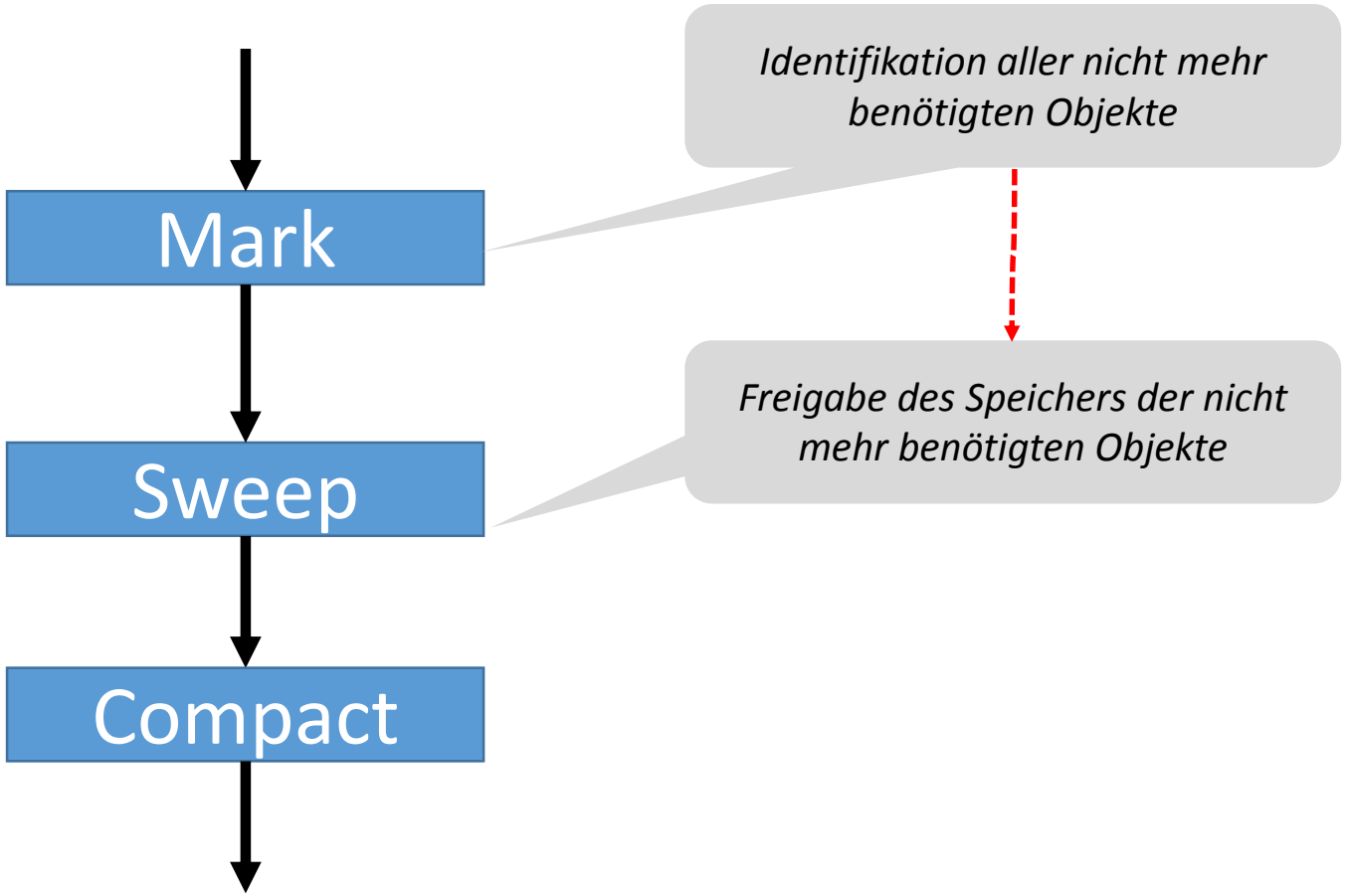
Algorithmen der Old Generation

- **Mark and Compact**



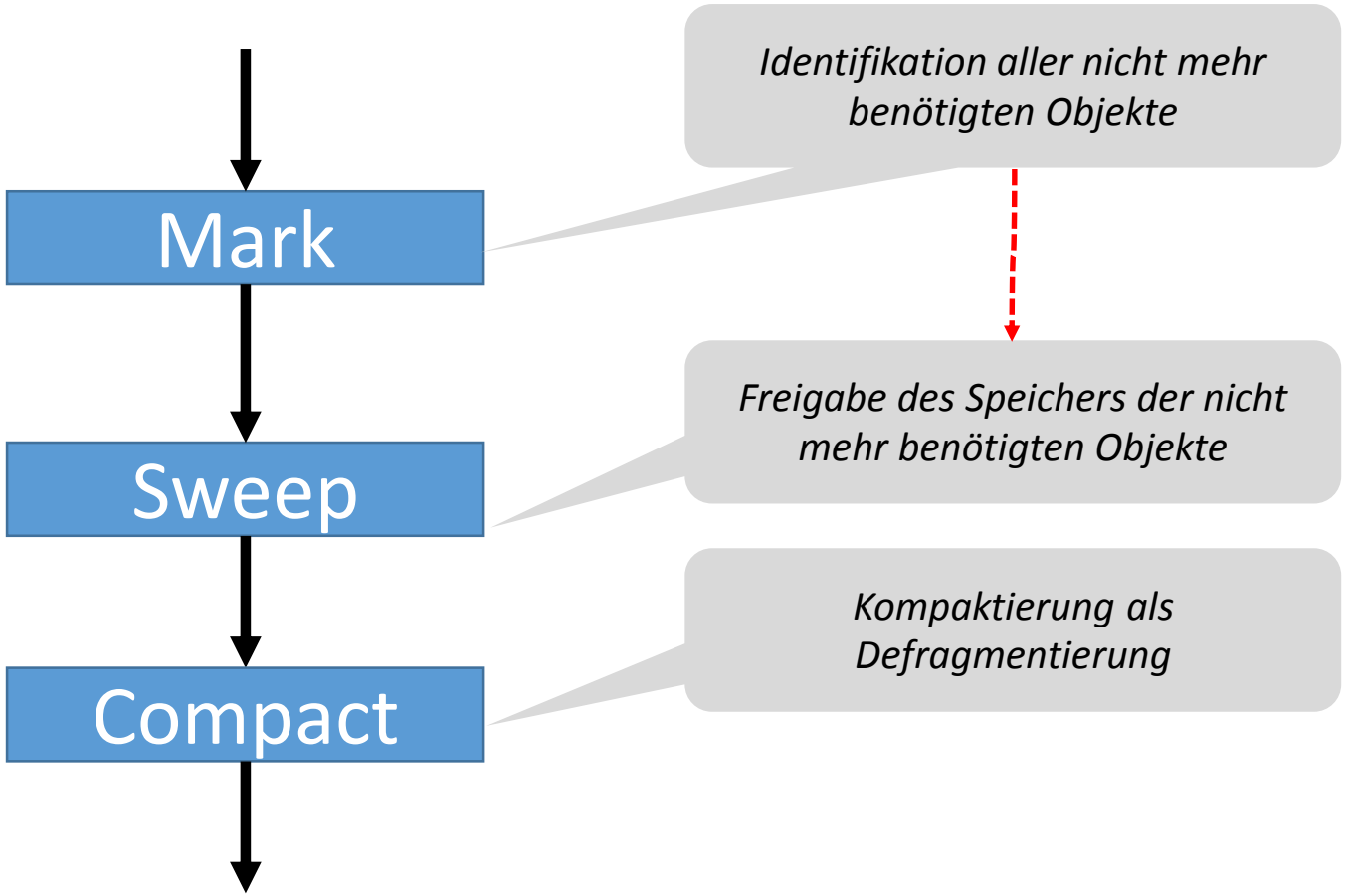
Algorithmen der Old Generation

- **Mark and Compact**



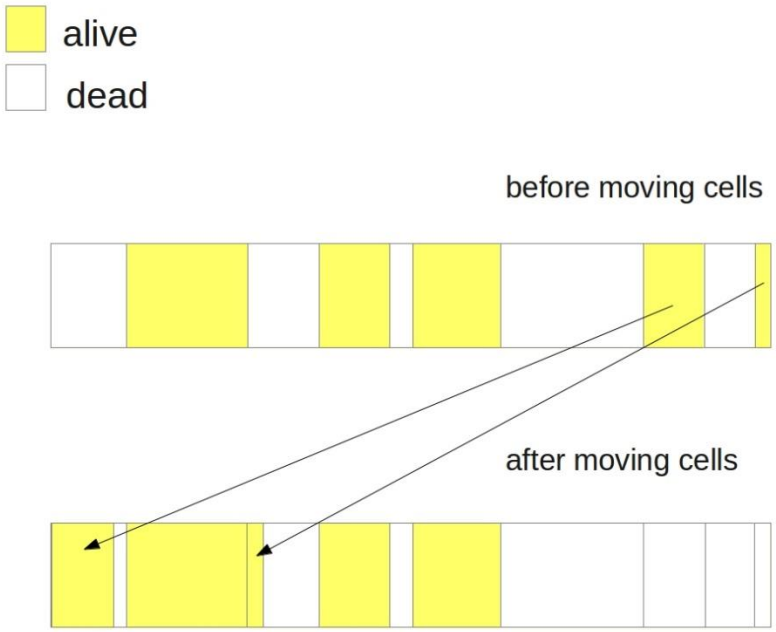
Algorithmen der Old Generation

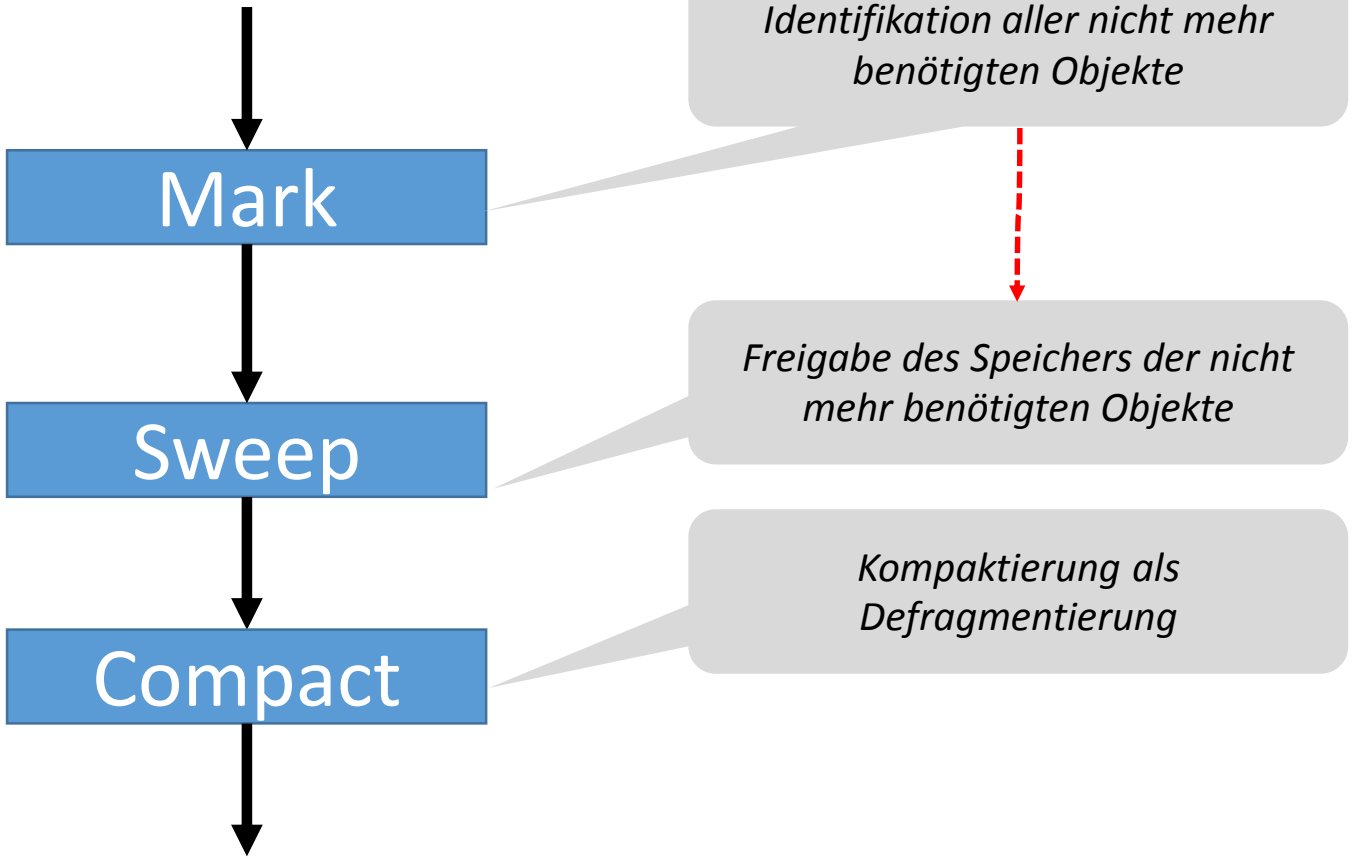
- **Mark and Compact**



Algorithmen der Old Generation

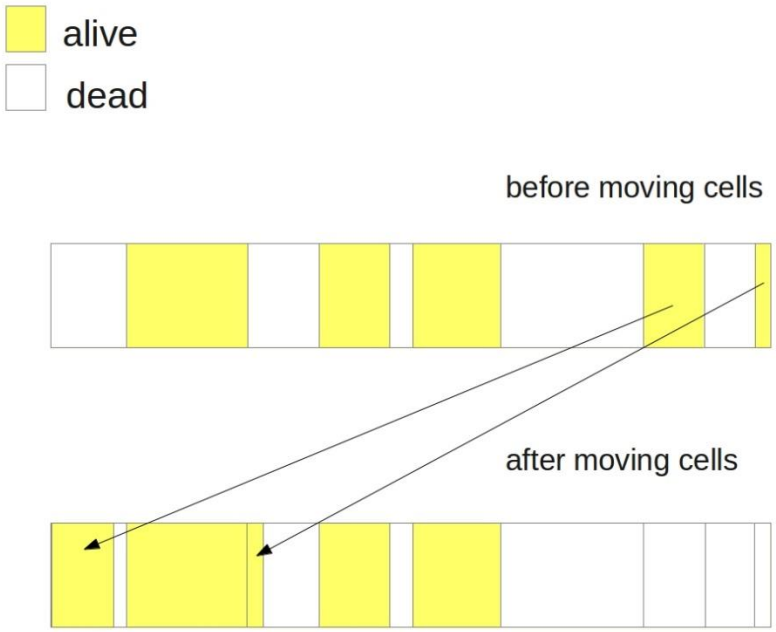
- **Mark and Compact**

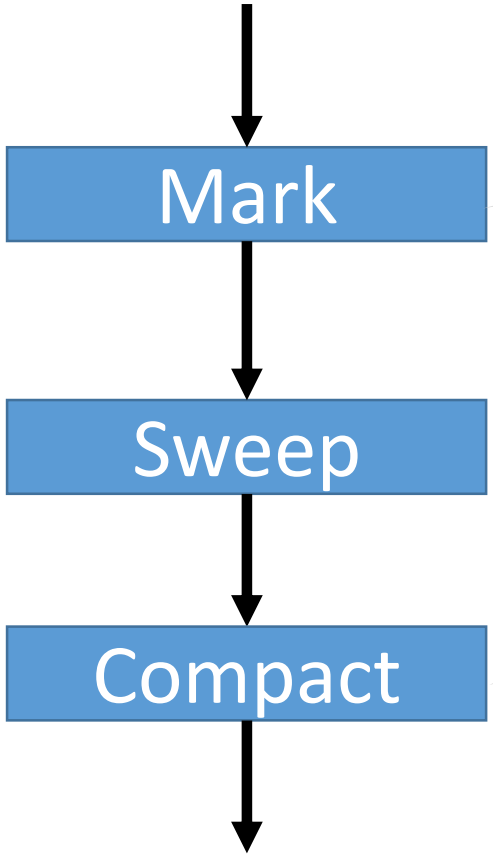




Algorithmen der Old Generation

- **Mark and Compact**





Identifikation aller nicht mehr benötigten Objekte

Freigabe des Speichers der nicht mehr benötigten Objekte

Kompaktierung als Defragmentierung

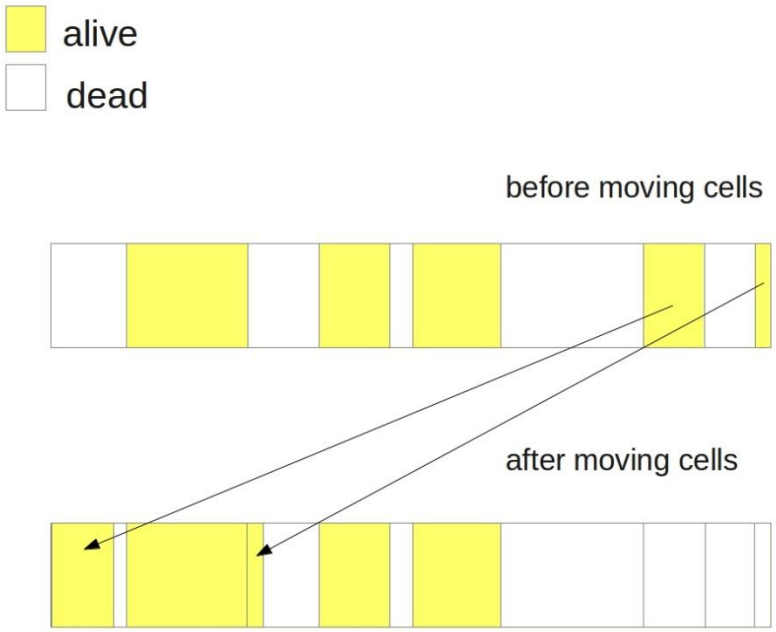
Stop-the-World (STW)

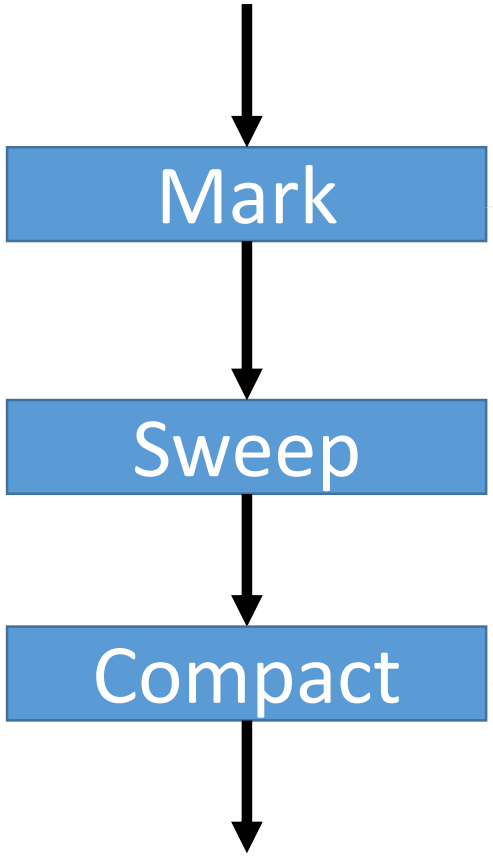
+

Kompaktierung kostet Performance

Algorithmen der Old Generation

- **Mark and Compact (Serial vs. Parallel)**





Identifikation aller nicht mehr benötigten Objekte

Freigabe des Speichers der nicht mehr benötigten Objekte

Kompaktierung als Defragmentierung

Stop-the-World (STW)

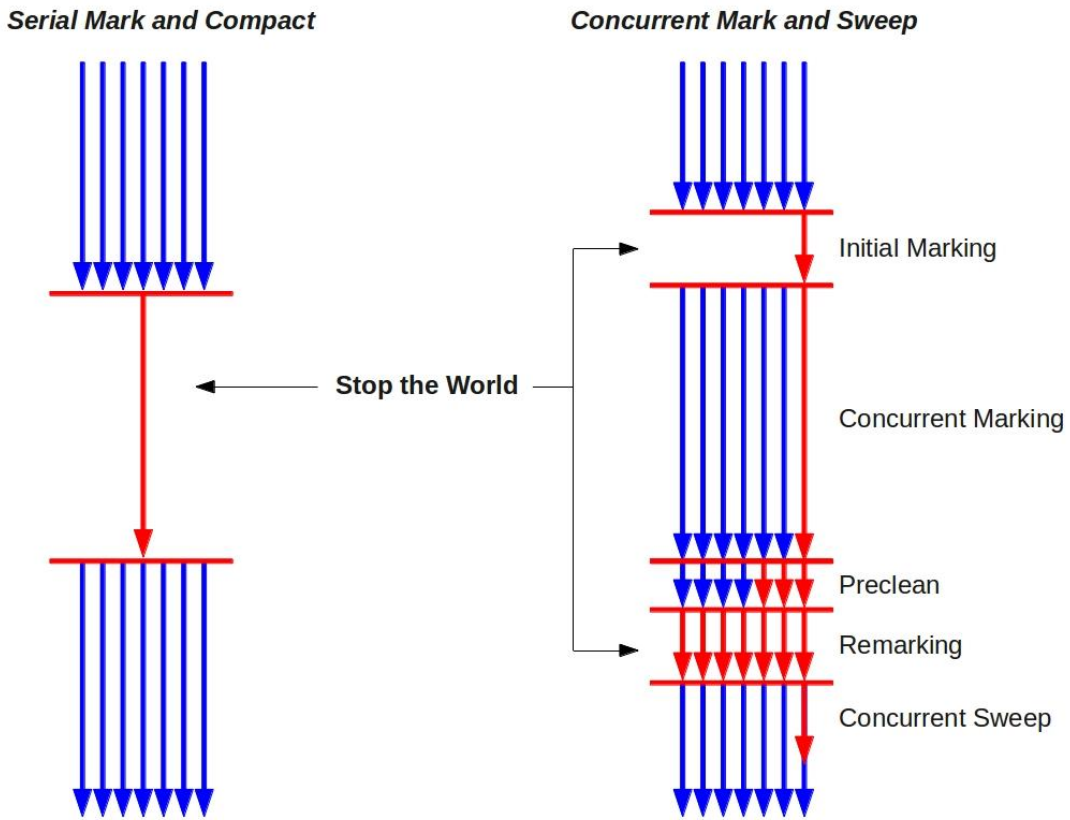
+

Kompaktierung kostet Performance

[12]

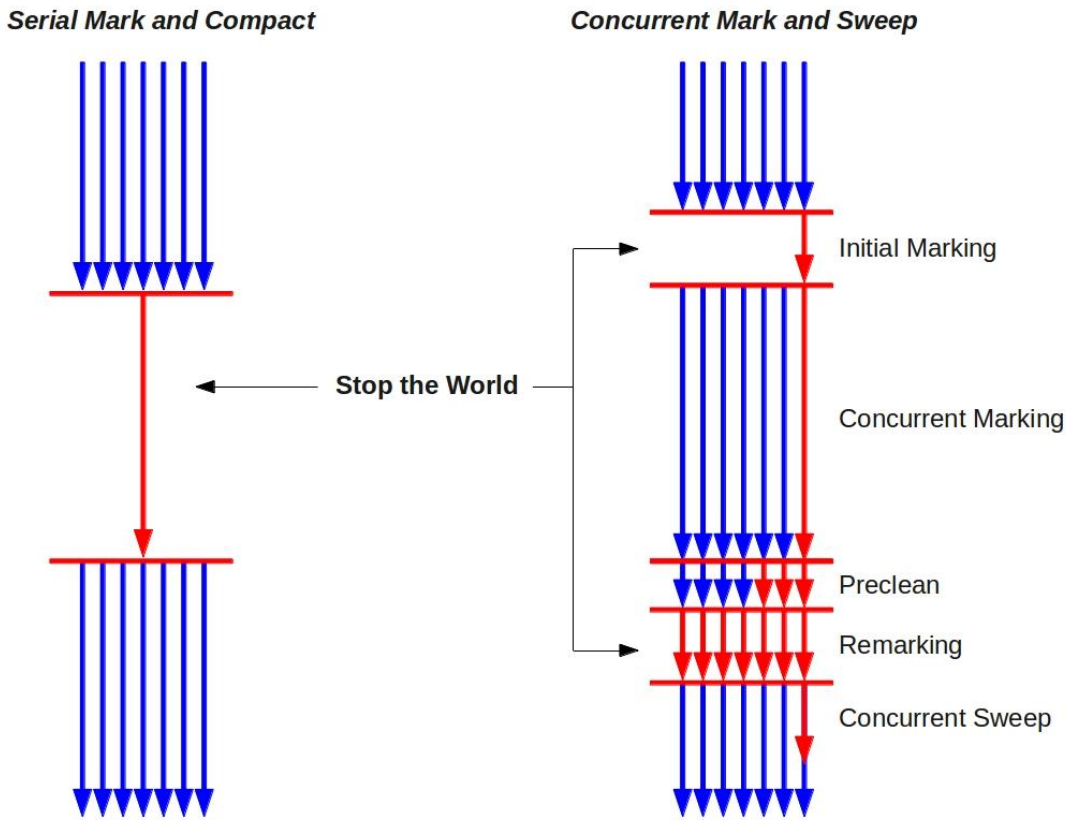
Algorithmen der Old Generation

- **Concurrent Mark and Sweep (CMS)**



Algorithmen der Old Generation

- **Concurrent Mark and Sweep (CMS)**



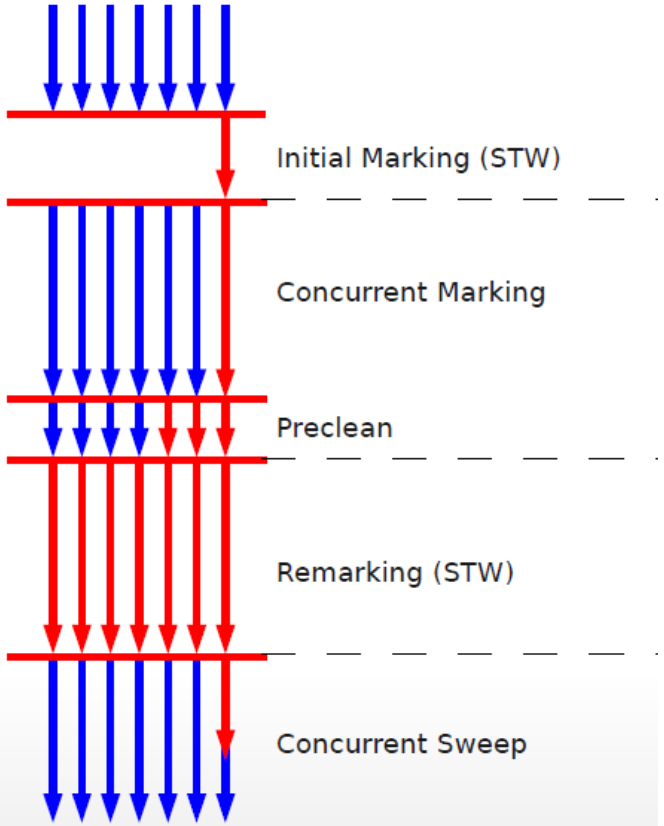
mehrere kleine STW

anstelle

einer großer STW

Algorithmen der Old Generation

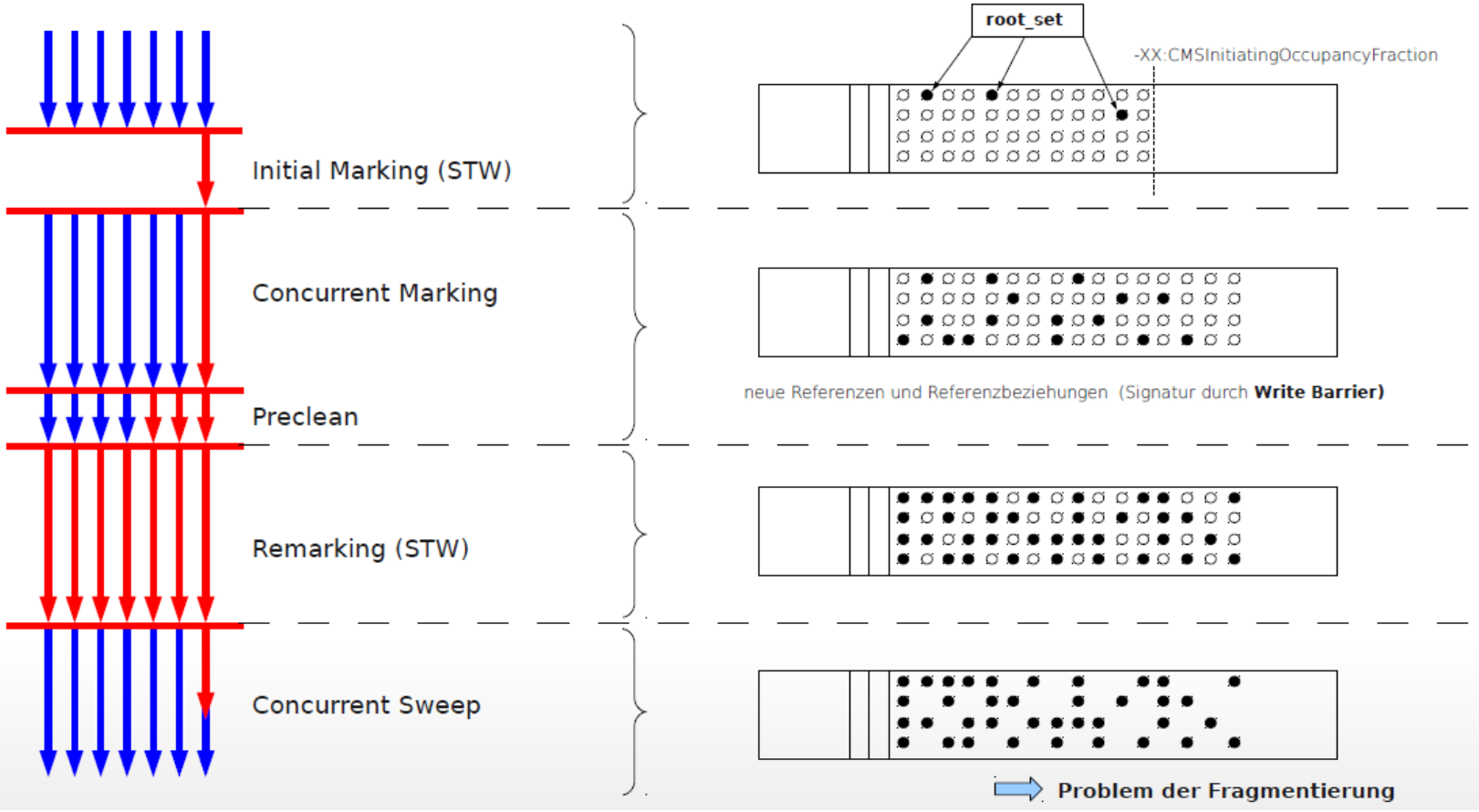
- **Concurrent Mark and Sweep (CMS)**



[12]

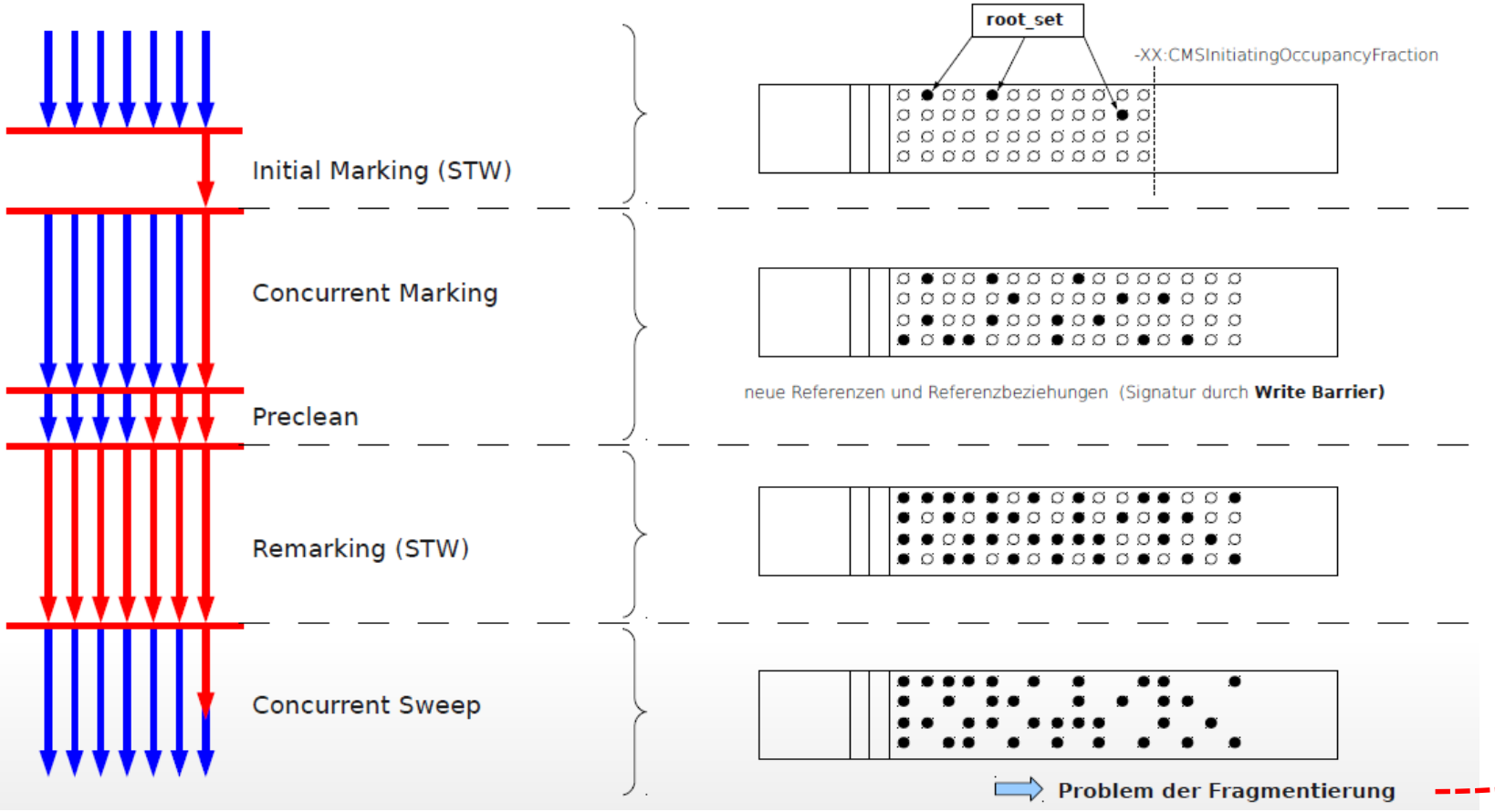
Algorithmen der Old Generation

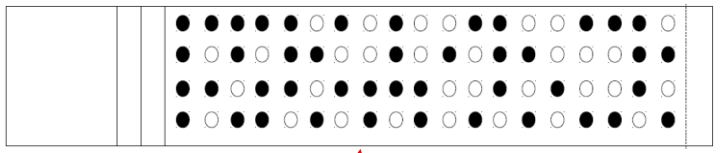
- **Concurrent Mark and Sweep (CMS)**



Algorithmen der Old Generation

- **Concurrent Mark and Sweep (CMS)**

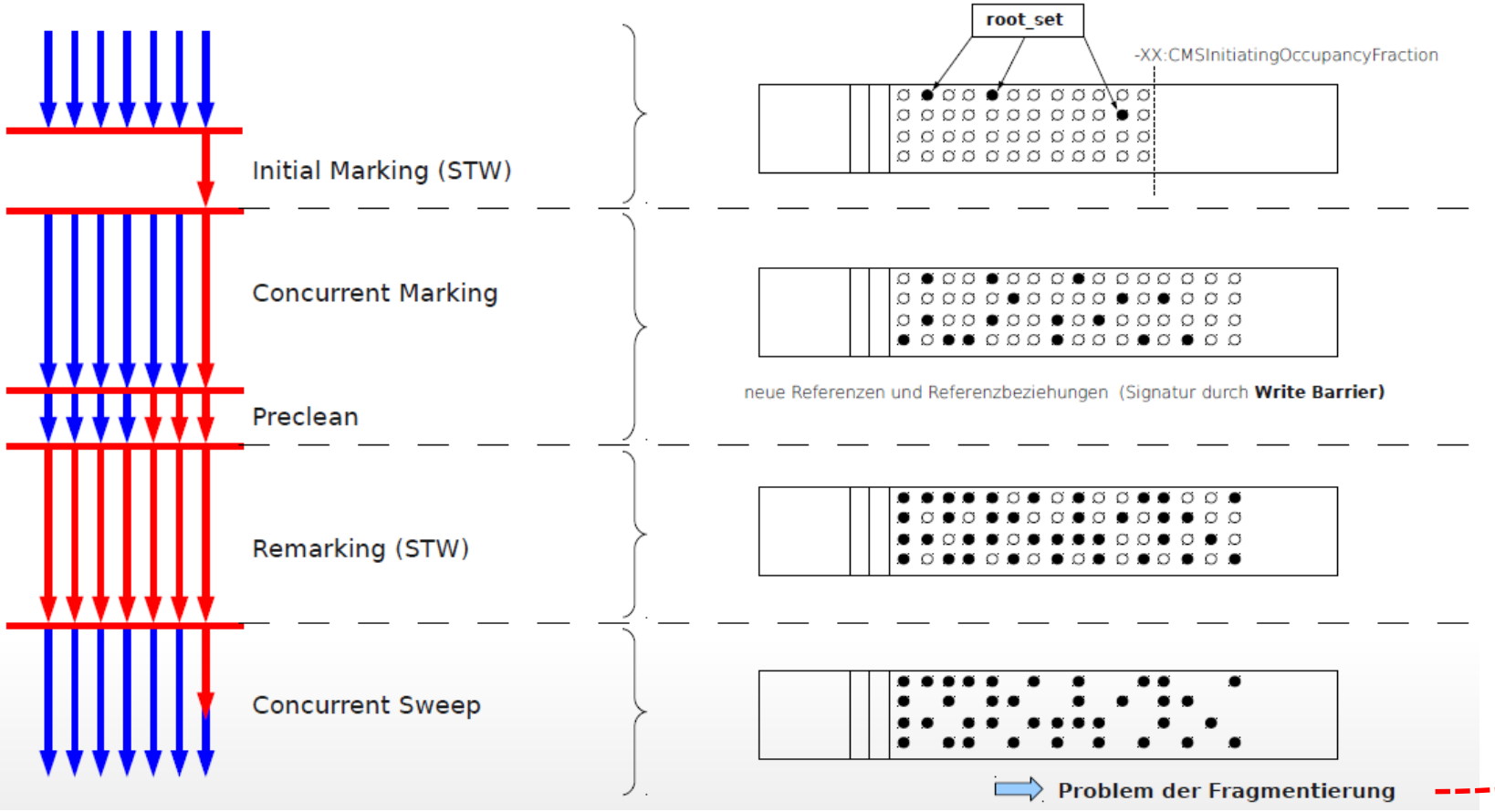


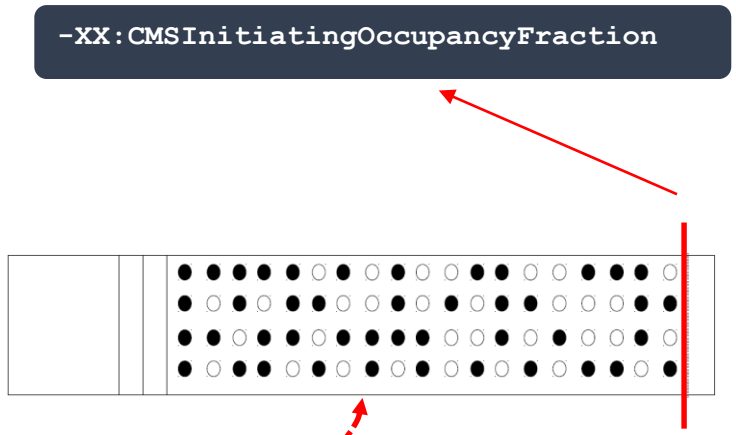


[12]

Algorithmen der Old Generation

- **Concurrent Mark and Sweep (CMS)**

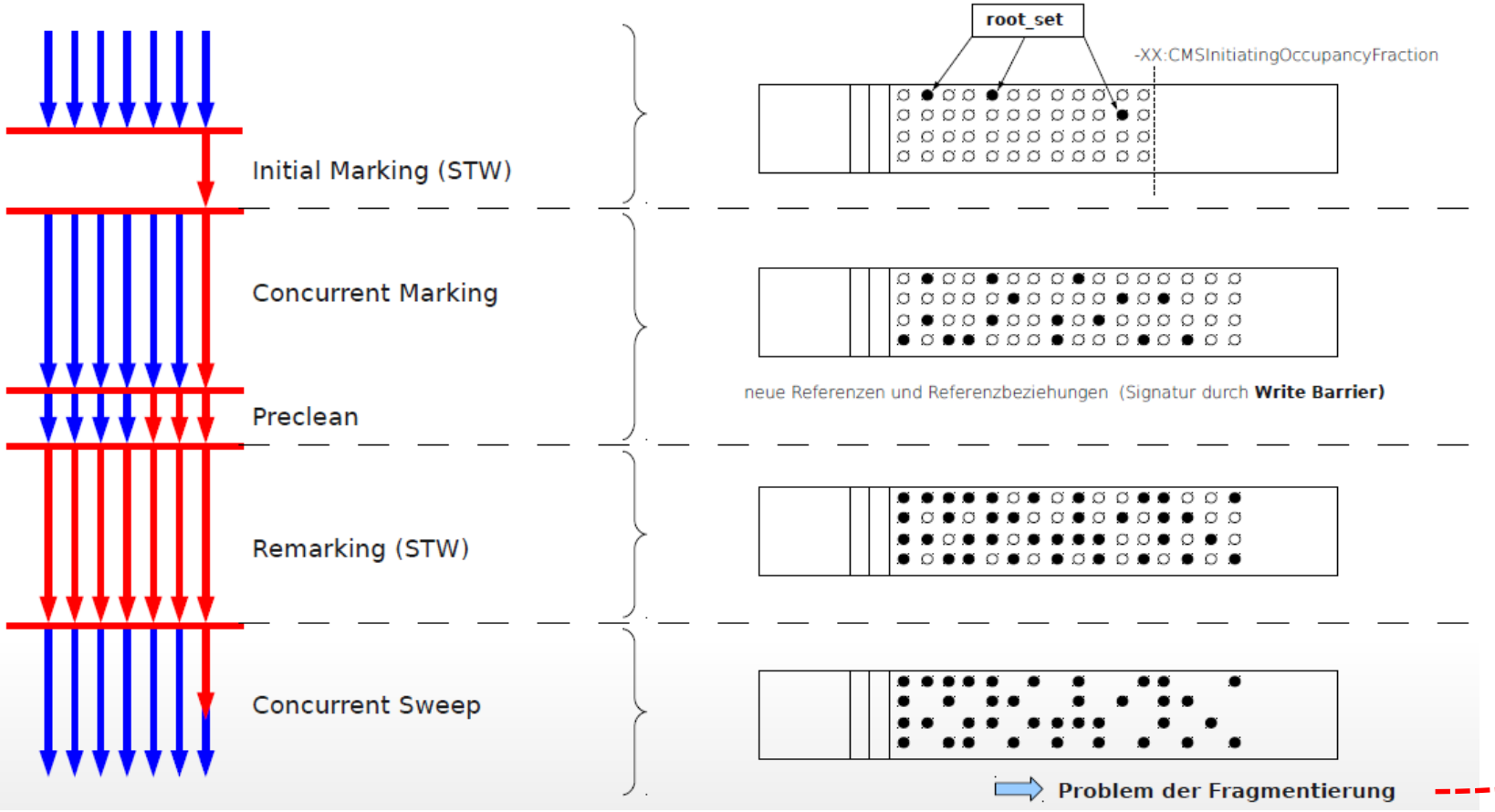




[12]

Algorithmen der Old Generation

- **Concurrent Mark and Sweep (CMS)**

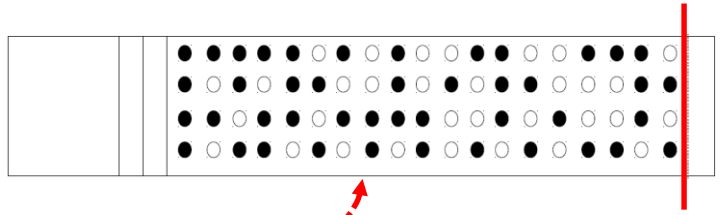


Serial Mark and Compact

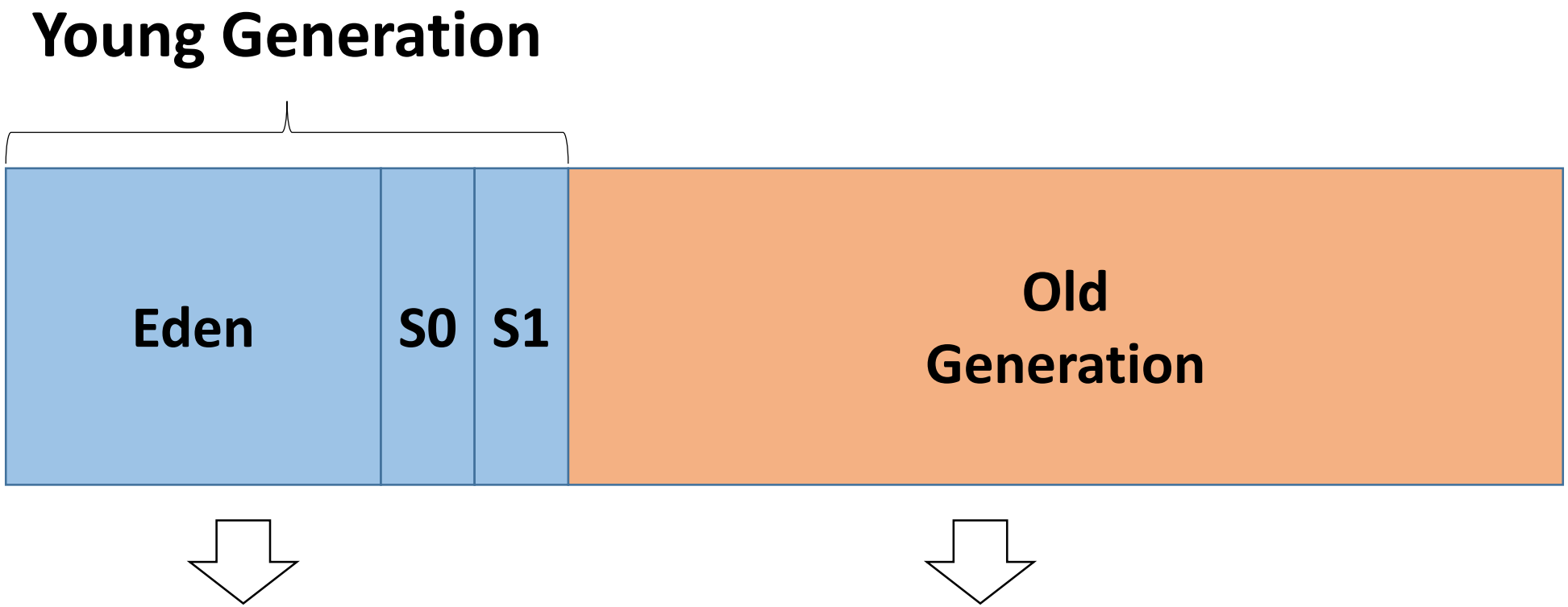
Full GC

sehr lange STW !!!

`-XX:CMSInitiatingOccupancyFraction`



kurze Gedankenpause



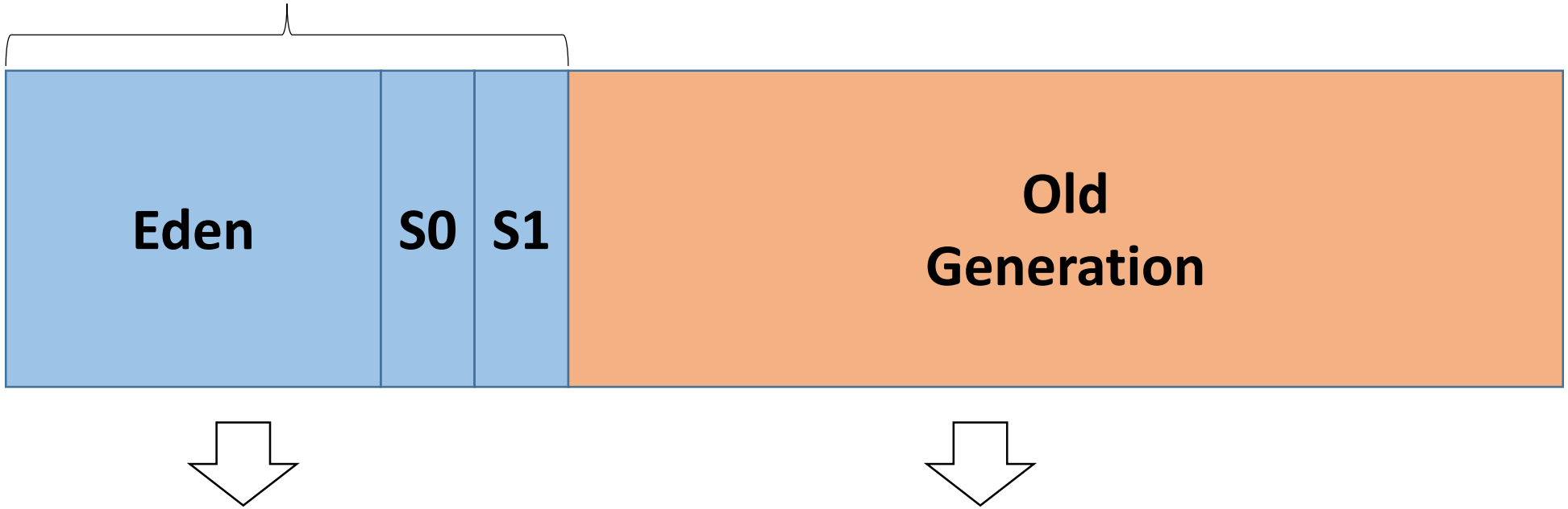
- *Mark and Sweep*
- *Serial/Parallel Mark and Copy*

- *Serial/Parallel Mark and Compact*
- *Concurrent Mark and Sweep (CMS)*

kurze Gedankenpause

Young Generation

`-XX:-UseConcMarkSweepGC`



- *Mark and Sweep*
- *Serial/Parallel Mark and Copy*

- *Serial/Parallel Mark and Compact*
- *Concurrent Mark and Sweep (CMS)*

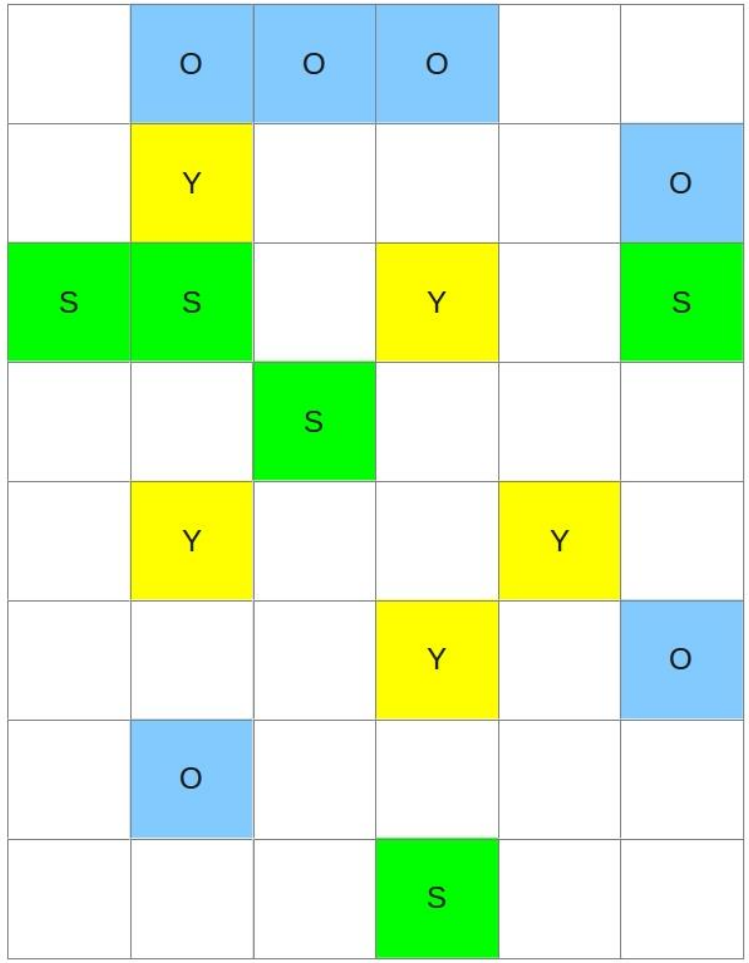
[12, 13]

Garbage First (G1)

Garbage First (G1)

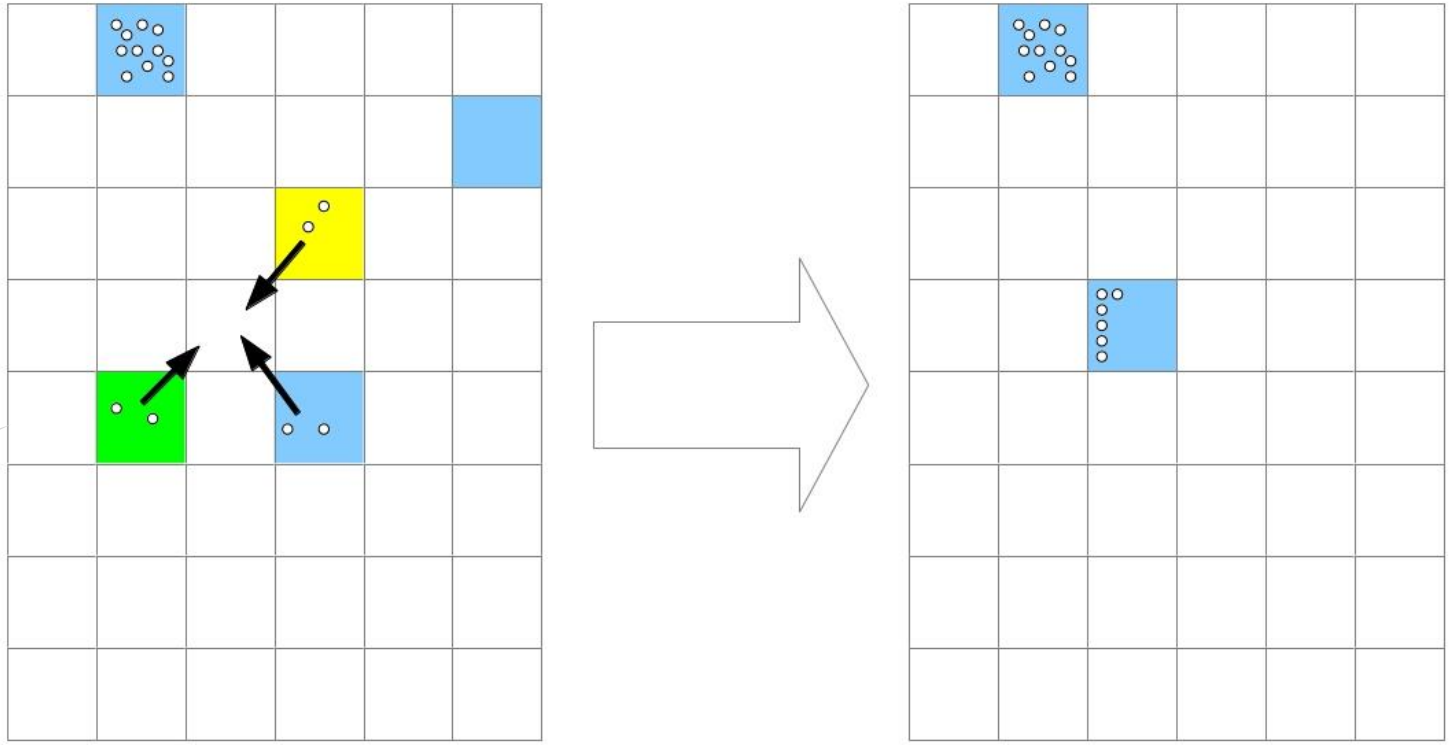
dynamische Heap-Architektur

Rollen der regions werden dynamisch zugeordnet



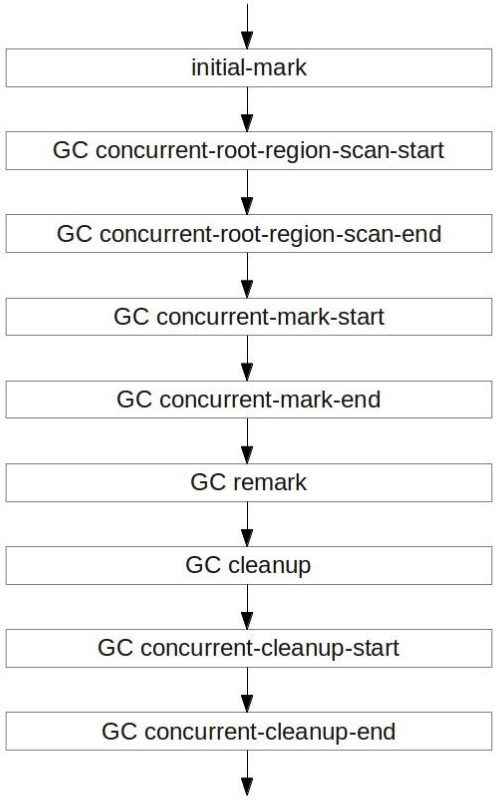
Garbage First (G1)

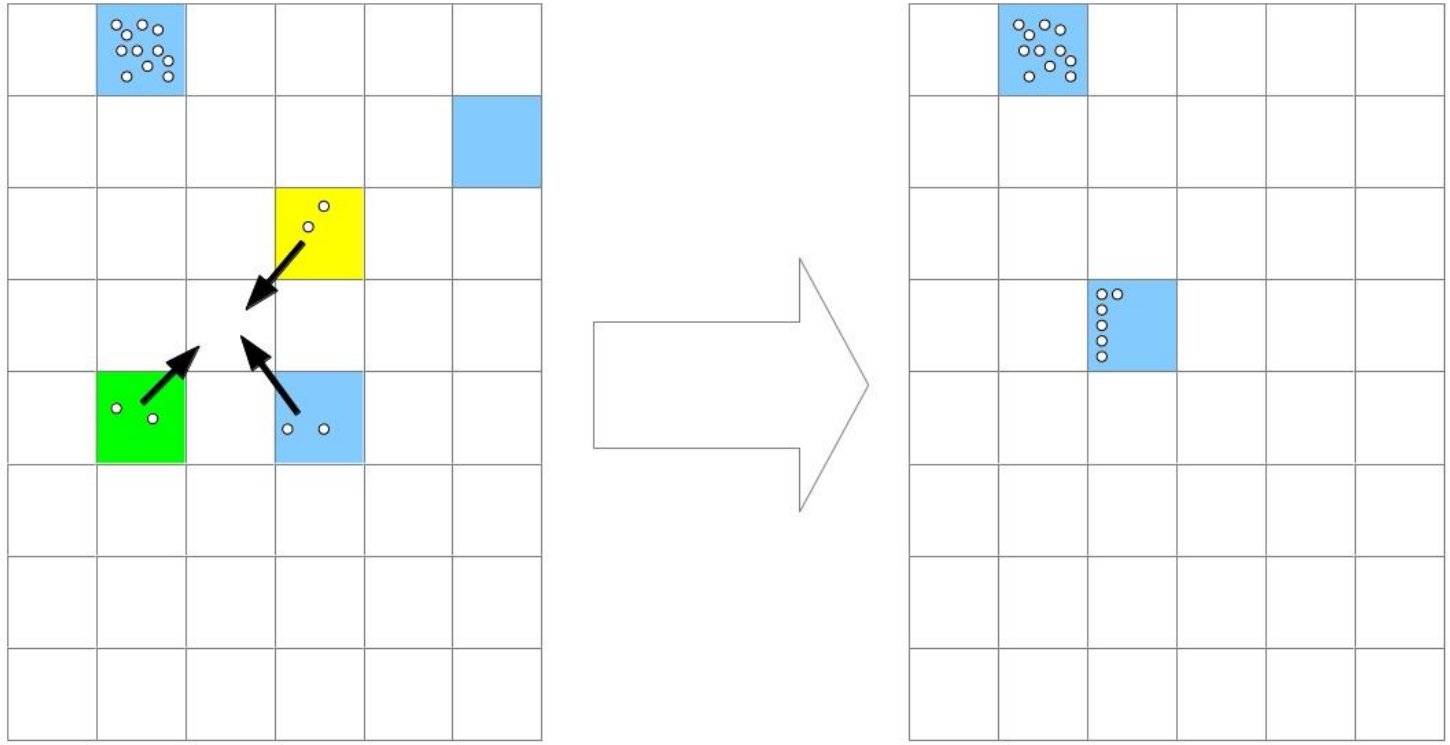
Größe der regions wird dynamisch bestimmt



Free Region
Old Region
Survivor Region
Young Region

Garbage First (G1)

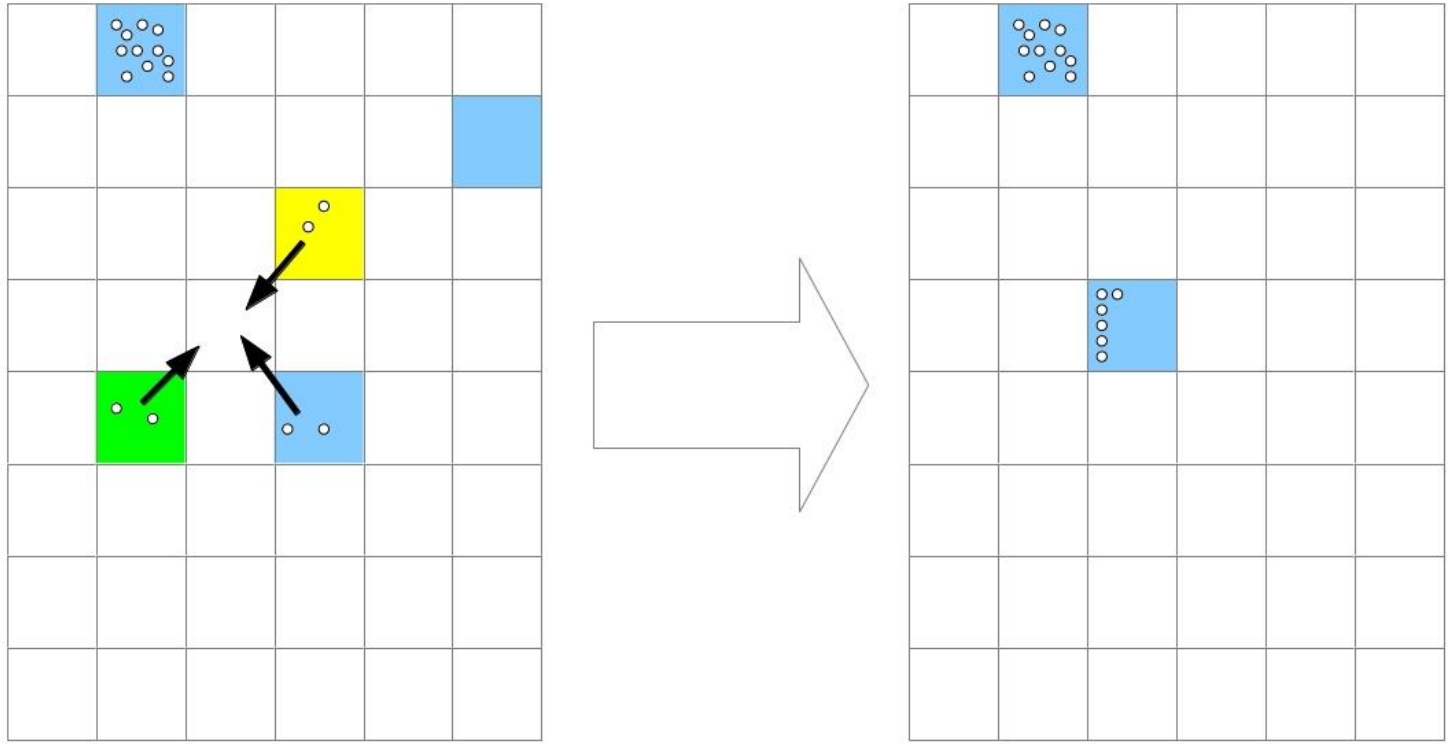




Free Region
Old Region
Survivor Region
Young Region

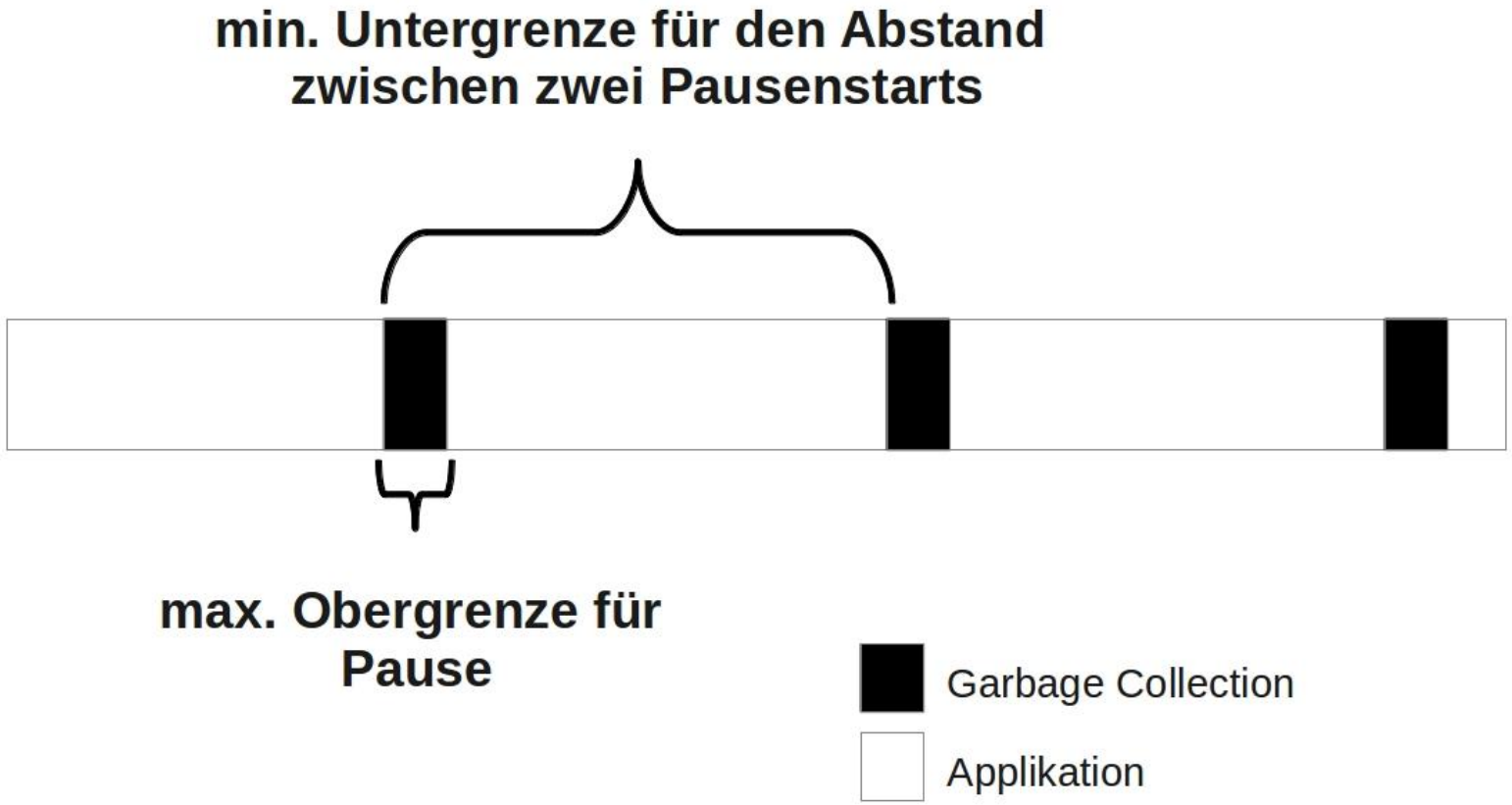
Garbage First (G1)

keine Fragmentierung
ABER:
Stop-the-World (STW)

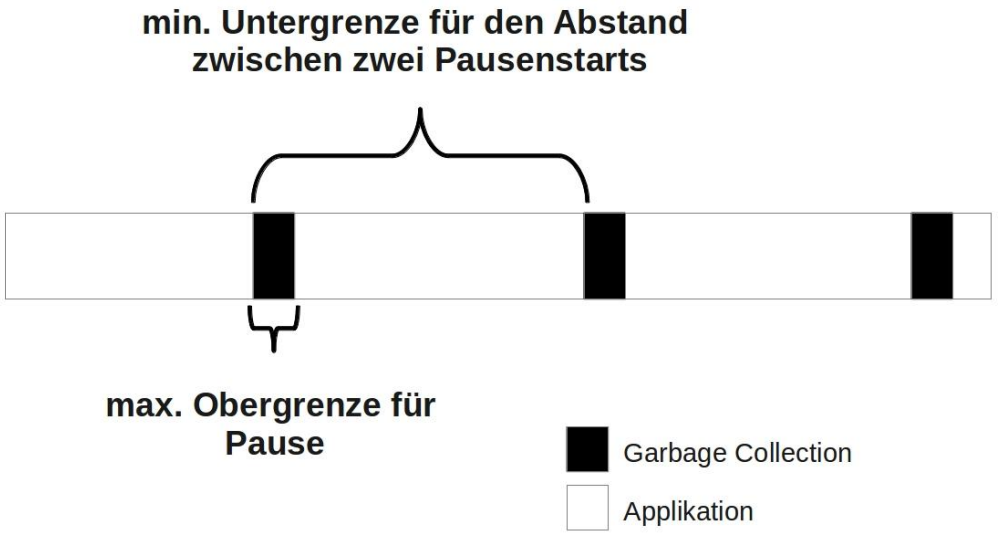


Free Region
Old Region
Survivor Region
Young Region

G1 - Laufzeitvorgaben

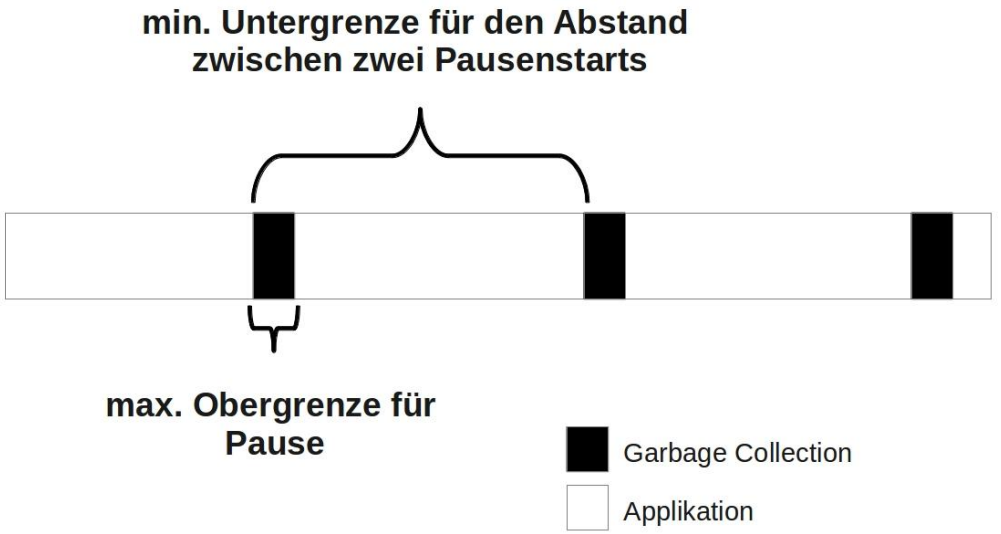


kurze Gedankenpause



kurze Gedankenpause

-XX:GCPauseTimeInterval



-XX:MaxGCPauseMillis

-XX:+UseG1GC

[12, 13]

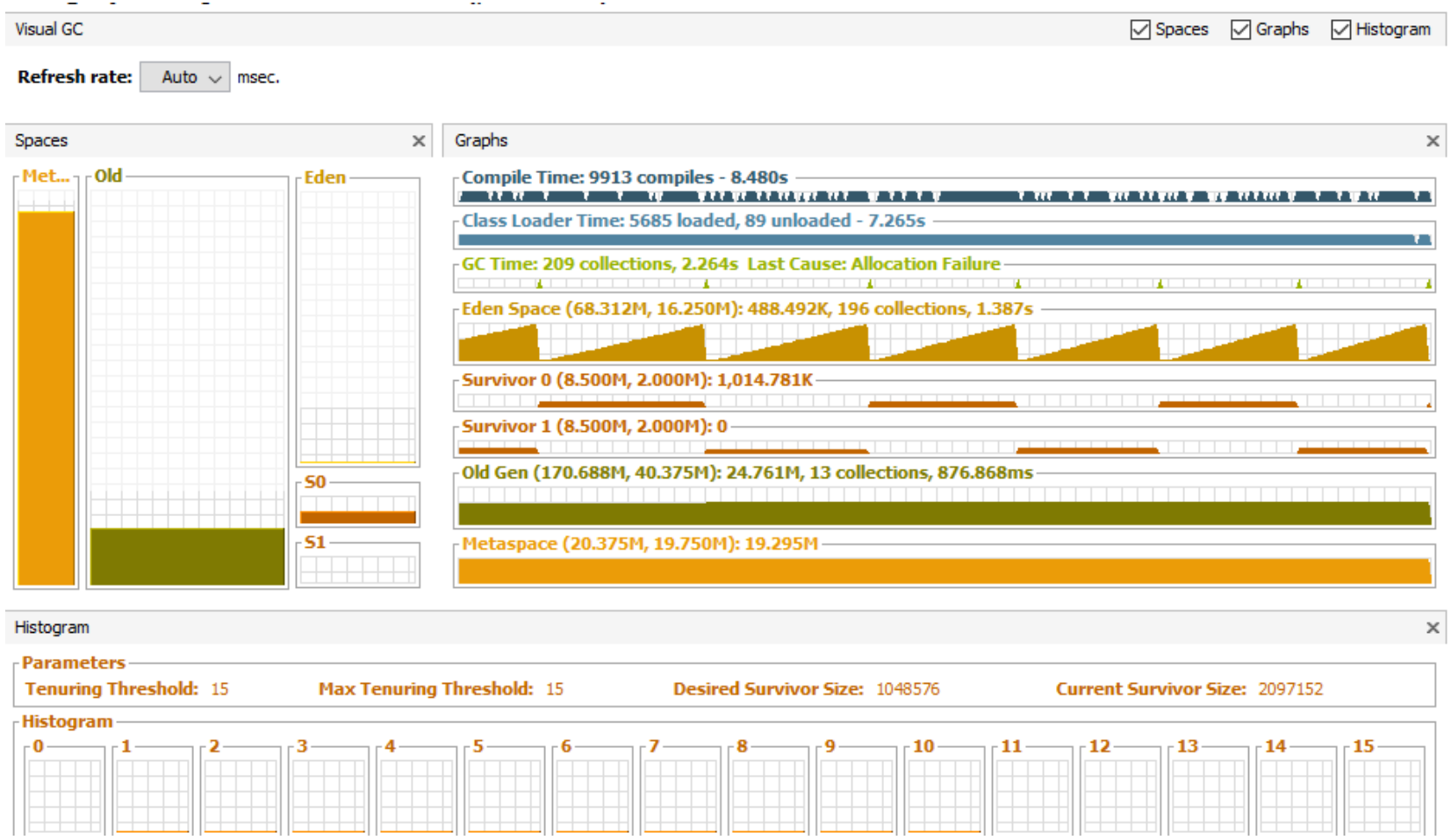
Measuring GC (1)

- jstat

```
C:\Program Files\Java\jdk1.8.0_77\bin>jstat -gc 100728 300ms 30
 S0C    S1C    S0U    S1U      EC       EU        OC         OU       MC     MU    CCSC   CCSU   YGC     YGCT    FGC    FGCT     GCT
 0.0   10240.0  0.0   10240.0 209920.0 164864.0 3974144.0  2997192.4  4992.0 4185.7 640.0  467.8      47   17.669   0      0.000   17.669
 0.0   10240.0  0.0   10240.0 209920.0 178176.0 3974144.0  3009480.4  4992.0 4185.7 640.0  467.8      47   17.669   0      0.000   17.669
 0.0   10240.0  0.0   10240.0 209920.0 194560.0 3974144.0  3009480.4  4992.0 4185.7 640.0  467.8      47   17.669   0      0.000   17.669
 0.0   26624.0  0.0   26624.0 193536.0 53248.0  3974144.0  3240392.4  4992.0 4185.7 640.0  467.8      48   18.187   0      0.000   18.187
 0.0   26624.0  0.0   26624.0 193536.0 182272.0 3974144.0  3250552.9  4992.0 4186.2 640.0  467.8      49   18.676   0      0.000   18.676
 0.0   26624.0  0.0   26624.0 193536.0 182272.0 3974144.0  3482725.7  4992.0 4186.2 640.0  467.8      50   18.676   0      0.000   18.969
 0.0   26624.0  0.0   26624.0 193536.0 182272.0 3974144.0  3746144.2  4992.0 4186.2 640.0  467.8      51   18.969   0      0.000   18.969
 0.0   26624.0  0.0   26624.0 193536.0 56320.0  3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      51   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   26624.0  0.0   26624.0 193536.0 163840.0 3974144.0  3944800.2  4992.0 4186.2 640.0  467.8      52   19.515   0      0.000   19.515
 0.0   1024.0  0.0   1024.0 53248.0     0.0    4140032.0  4139360.2  4992.0 4186.2 640.0  467.8      52   23.695   1      0.000   23.695
 0.0   1024.0  0.0   1024.0 53248.0     0.0    4140032.0  4139360.2  4992.0 4186.2 640.0  467.8      52   23.695   1      0.000   23.695
 0.0   1024.0  0.0   1024.0 53248.0     0.0    4140032.0  4139360.2  4992.0 4186.2 640.0  467.8      52   23.695   1      0.000   23.695
 0.0   1024.0  0.0   1024.0 53248.0     0.0    4140032.0  4139360.2  4992.0 4186.2 640.0  467.8      52   23.695   1      0.000   23.695
 0.0   1024.0  0.0   1024.0 53248.0     0.0    4140032.0  4139360.2  4992.0 4186.2 640.0  467.8      52   23.695   1      0.000   23.695
 0.0   1024.0  0.0   1024.0 53248.0     0.0    4140032.0  4139360.2  4992.0 4186.2 640.0  467.8      52   23.695   1      0.000   23.695
 0.0   1024.0  0.0   1024.0 53248.0     0.0    4140032.0  4139360.2  4992.0 4186.2 640.0  467.8      52   23.695   1      0.000   23.695
 0.0     0.0   0.0     0.0  110592.0    0.0    1986560.0   661080.8  4992.0 4180.1 640.0  465.1      52   23.695   1      2.484   26.179
 0.0   13312.0  0.0   13312.0 97280.0     0.0    1986560.0   789807.9  4992.0 4180.1 640.0  465.1      53   23.819   1      2.484   26.303
```

Measuring GC (1)

- visualgc



Measuring GC (2)

```
-verbose:gc
-XX:+PrintGCDetails
-XX:+PrintGCApplicationStoppedTime
…
```

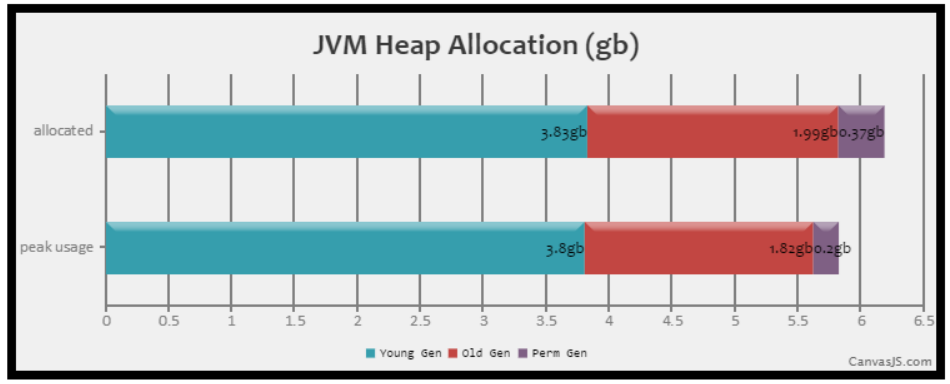
```
Total time for which application threads were stopped: 0.1656340 seconds
2013-05-23T07:02:57.200+0000: 506.744: [GC [1 CMS-initial-mark: 530071K(699072K)]
565479K(1013632K), 0.0589190 secs] [Times: user=0.06 sys=0.00, real=0.06 secs]
Total time for which application threads were stopped: 0.0595530 seconds
2013-05-23T07:02:57.260+0000: 506.803: [CMS-concurrent-mark-start]
2013-05-23T07:02:57.486+0000: 507.029: [CMS-concurrent-mark: 0.226/0.226 secs] [Times:
user=0.51 sys=0.00, real=0.22 secs]
2013-05-23T07:02:57.486+0000: 507.029: [CMS-concurrent-preclean-start]
2013-05-23T07:02:57.489+0000: 507.032: [CMS-concurrent-preclean: 0.003/0.003 secs] [Times:
user=0.01 sys=0.00, real=0.01 secs]
2013-05-23T07:02:57.489+0000: 507.032: [CMS-concurrent-abortable-preclean-start]
Total time for which application threads were stopped: 0.0003090 seconds
 CMS: abort preclean due to time 2013-05-23T07:03:03.032+0000: 512.575: [CMS-concurrent-
abortable-preclean: 4.741/5.543 secs] [Times: user=5.14 sys=0.00, real=5.54 secs]
2013-05-23T07:03:03.032+0000: 512.576: [GC[YG occupancy: 68678 K (314560 K)]512.576: [Rescan
(parallel) , 0.0676490 secs]512.644: [weak refs processing, 0.2948180 secs]512.939: [scrub
ating table, 0.0003660 secs] [1 CMS-remark: 530071K(699072K)] 598749K(1013632K), 0.3630640
secs] [Times: user=0.83 sys=0.00, real=0.37 secs]
Total time for which application threads were stopped: 0.3635990 seconds
2013-05-23T07:03:03.396+0000: 512.939: [CMS-concurrent-sweep-start]
2013-05-23T07:03:03.848+0000: 513.391: [CMS-concurrent-sweep: 0.452/0.452 secs] [Times:
user=0.45 sys=0.00, real=0.45 secs]
```

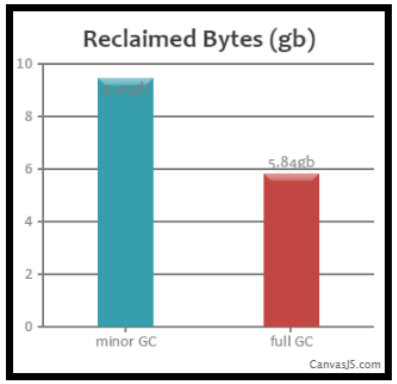
Measuring GC (2)

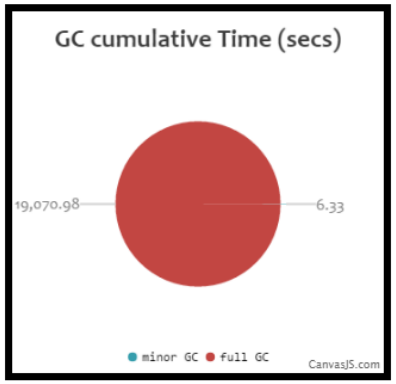
- https://gceasy.io/

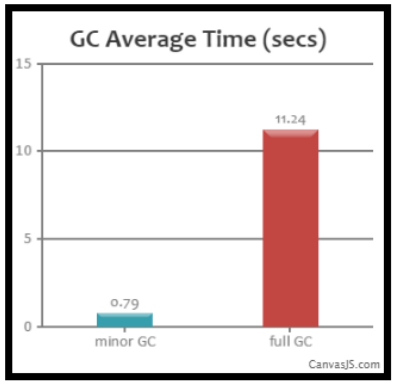
Measuring GC (2)

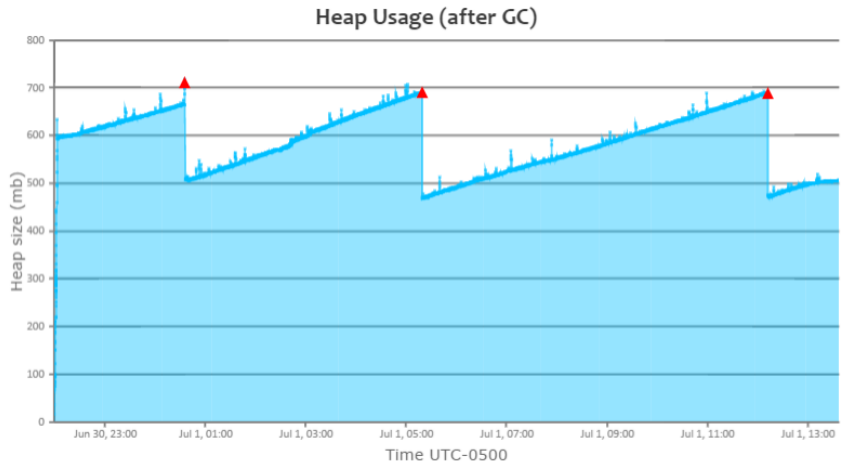
- https://gceasy.io/

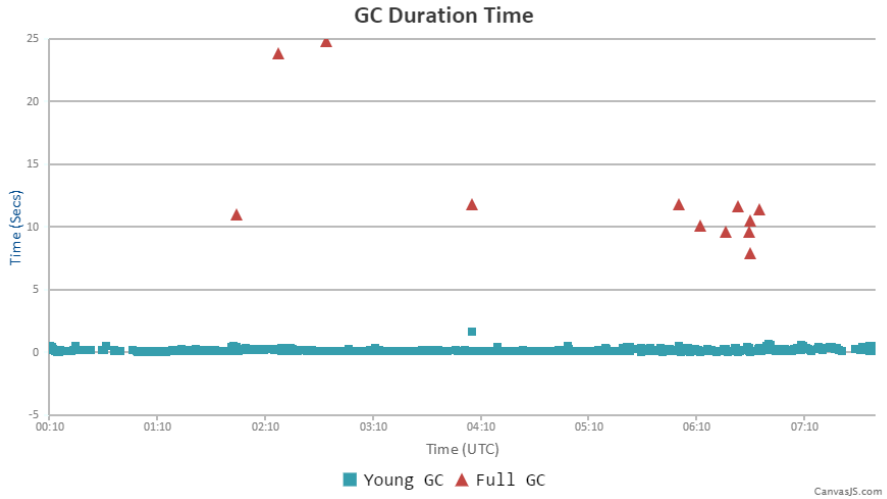




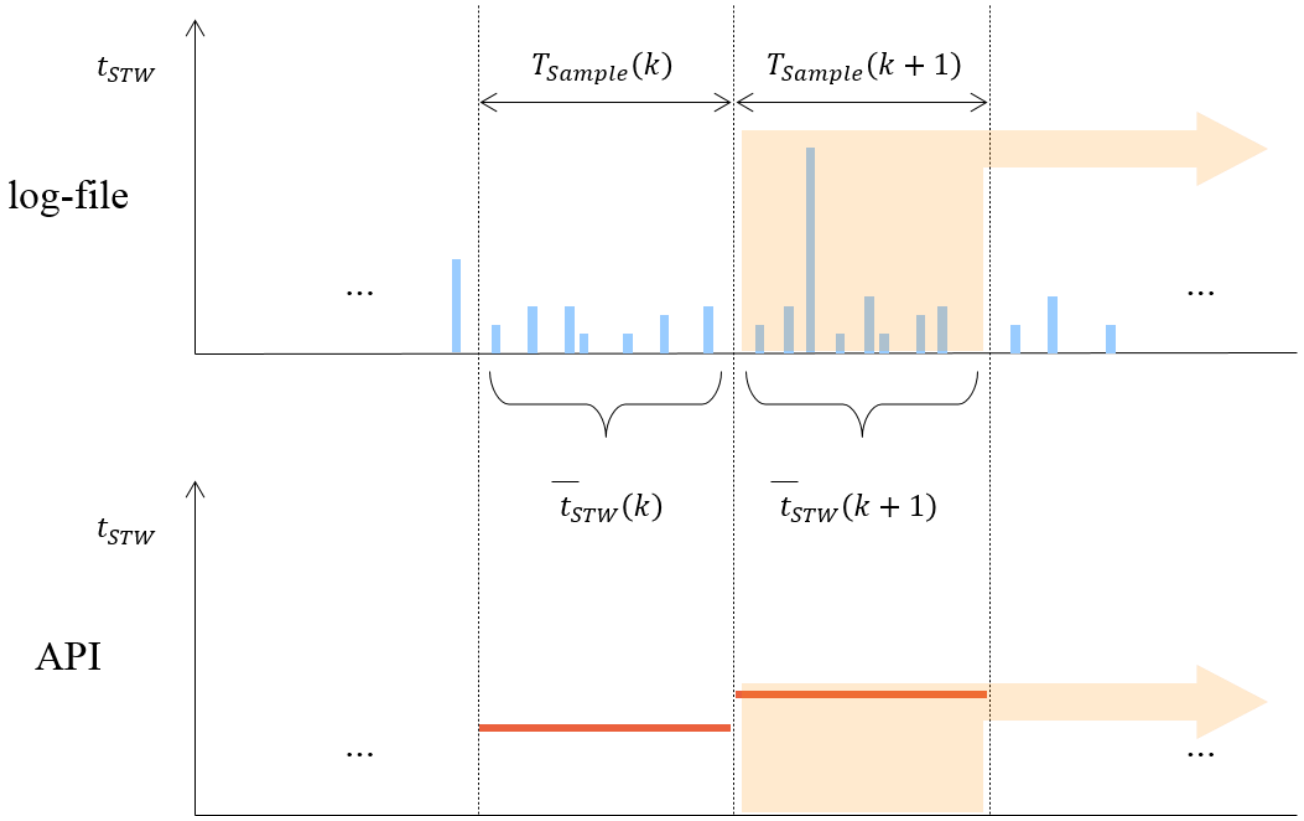








@Medium: Be Precise in Measuring GC



average:

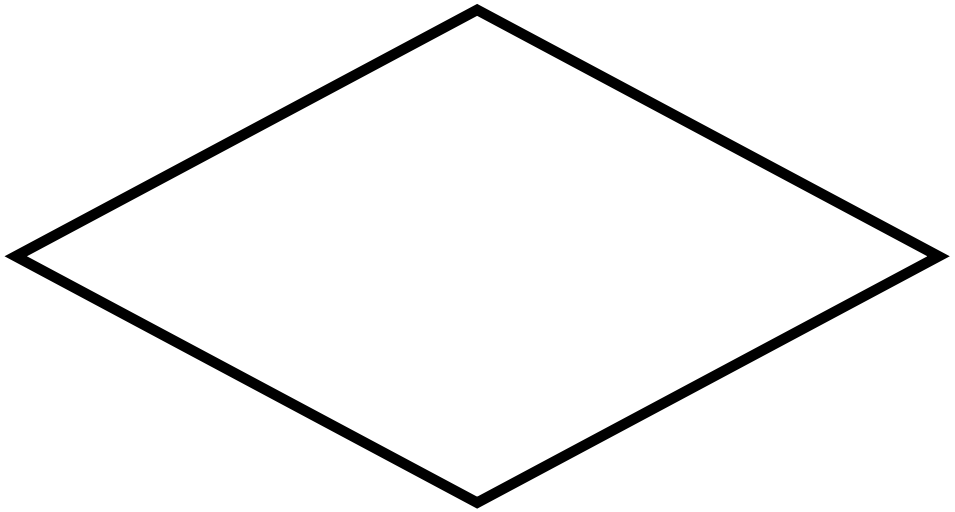
$$\overline{t_{STW}}(k) = \frac{1}{n(k)} \cdot \sum_{i=1}^{n(k)} t_{STW}\,(i_k)$$

$$\frac{getCollectionTime(k+1) - getCollectionTime(k)}{getCollectionCount(k+1) - getCollectionCount(k)}$$

Tuning the GC

high
throughput

short
pauses



less
memory

less
CPU

… and how do I do that in practice?

-XX:+UseG1GC

-XX:-UseConcMarkSweepGC

-XX:-UseParallelGC

-XX:-UseParallelOldGC

-XX:NewRatio=2

-XX:MaxGCPauseMillis=n

-XX:SurvivorRatio=8

-XX:NewSize=2m

-XX:NewRatio=n

-XX:TargetSurvivorRatio=50

-XX:SurvivorRatio=n

-XX:+UseBiasedLocking

-XX:MaxTenuringThreshold=n

-XX:+UseLargePages

-XX:AllocatePrefetchStyle=1

-XX:ParallelGCThreads=n

-XX:+UseCompressedStrings

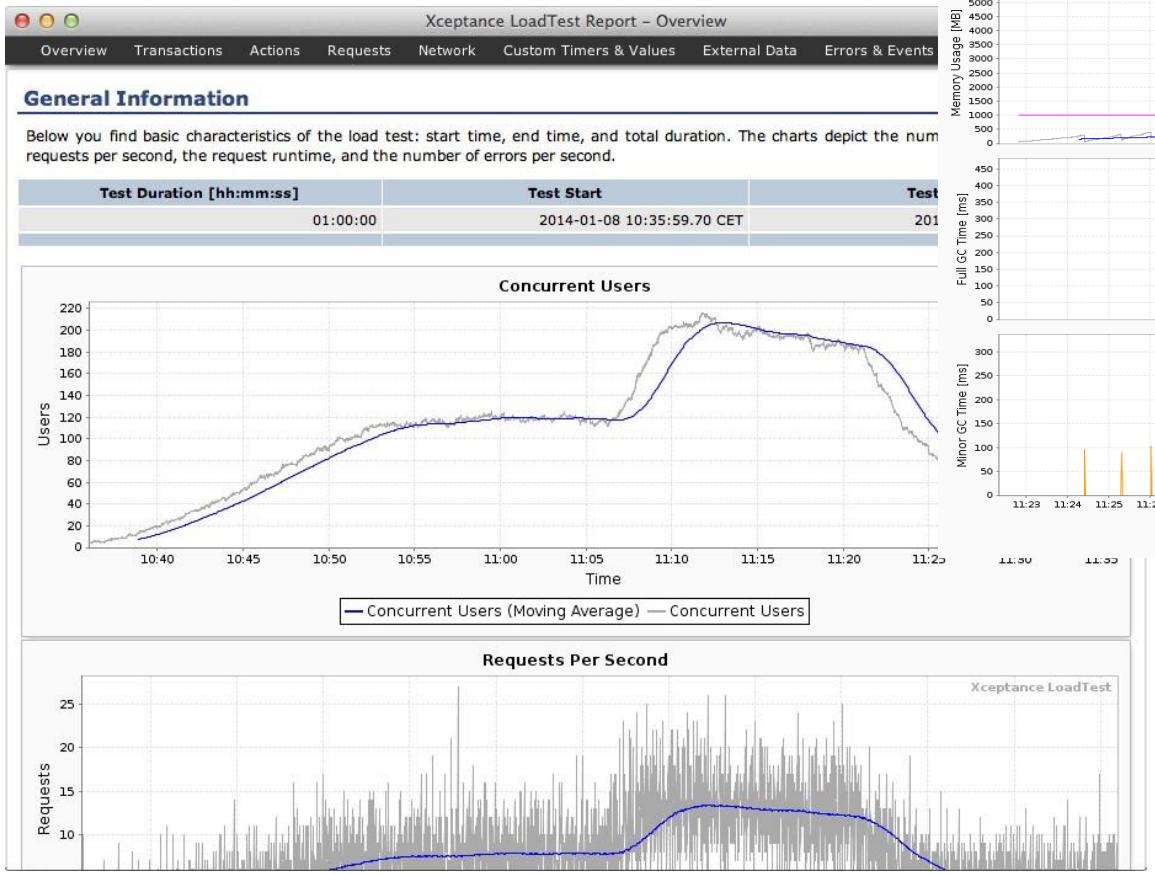
-XX:+OptimizeStringConcat

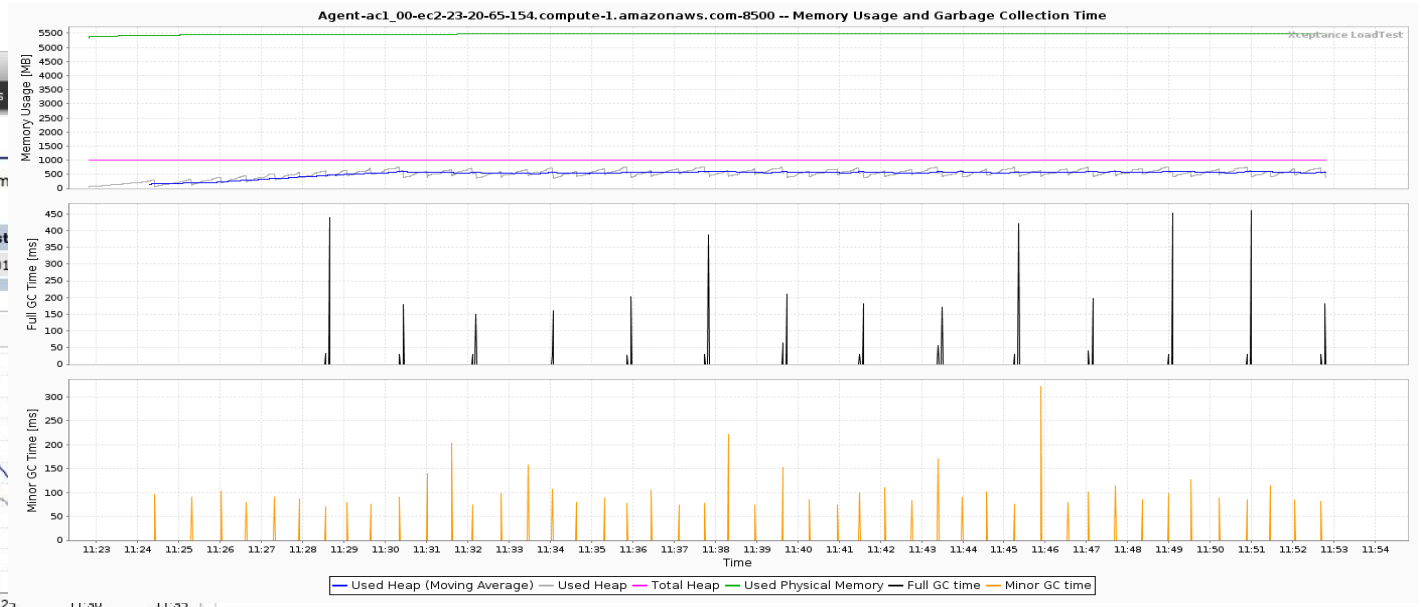
-XX:ConcGCThreads=n

-XX:+UseFastAccessorMethods

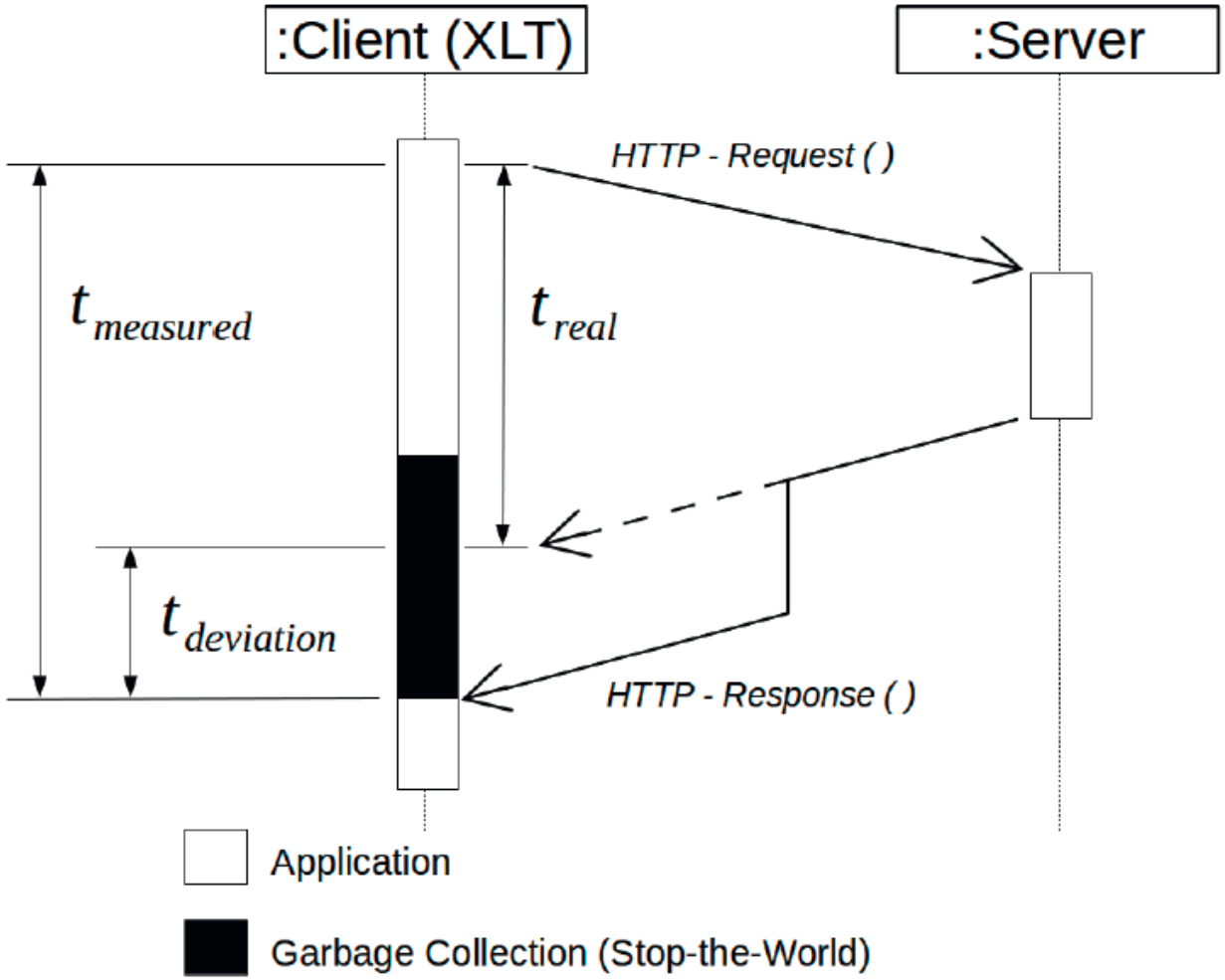
GC in Practice (1)

- Load- and Performancetesting with XLT





Load- and Performance-Testing with XLT

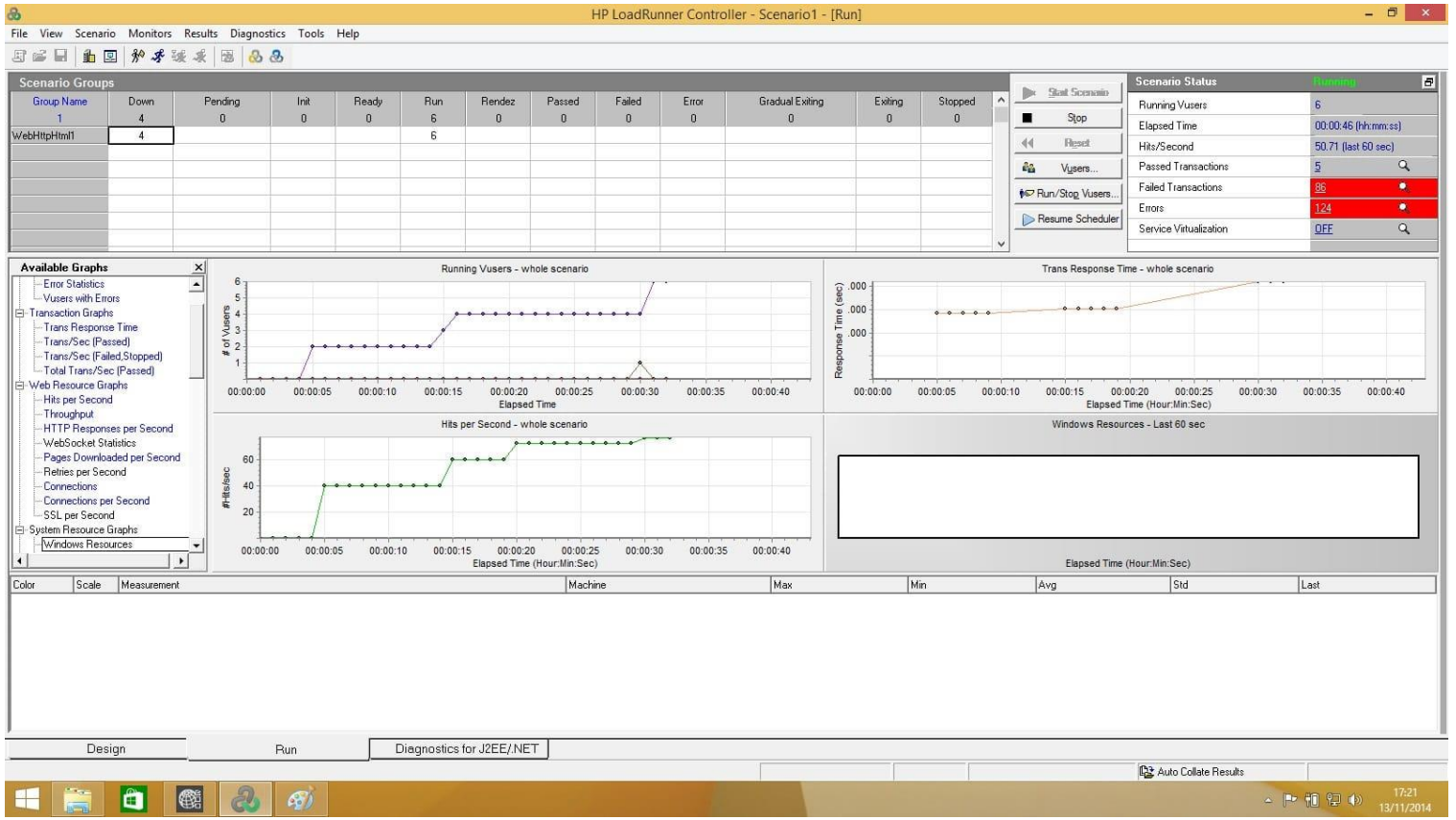


Minimizing GC-Latencies on XLT

Test	Maximum Latencies in Seconds		Reduction in Percentage
	Default	Tuned	
Test 1	0.68	0.55	19.1%
Test 2	0.71	0.45	36.6%
Test 3	0.60	0.49	18.4%
Test 4	0.56	0.49	12.5%
Test 5	0.40	0.26	35.0%

GC in Practice (2)

- Performance-Analysis in telematics (SOAP-Interfaces)

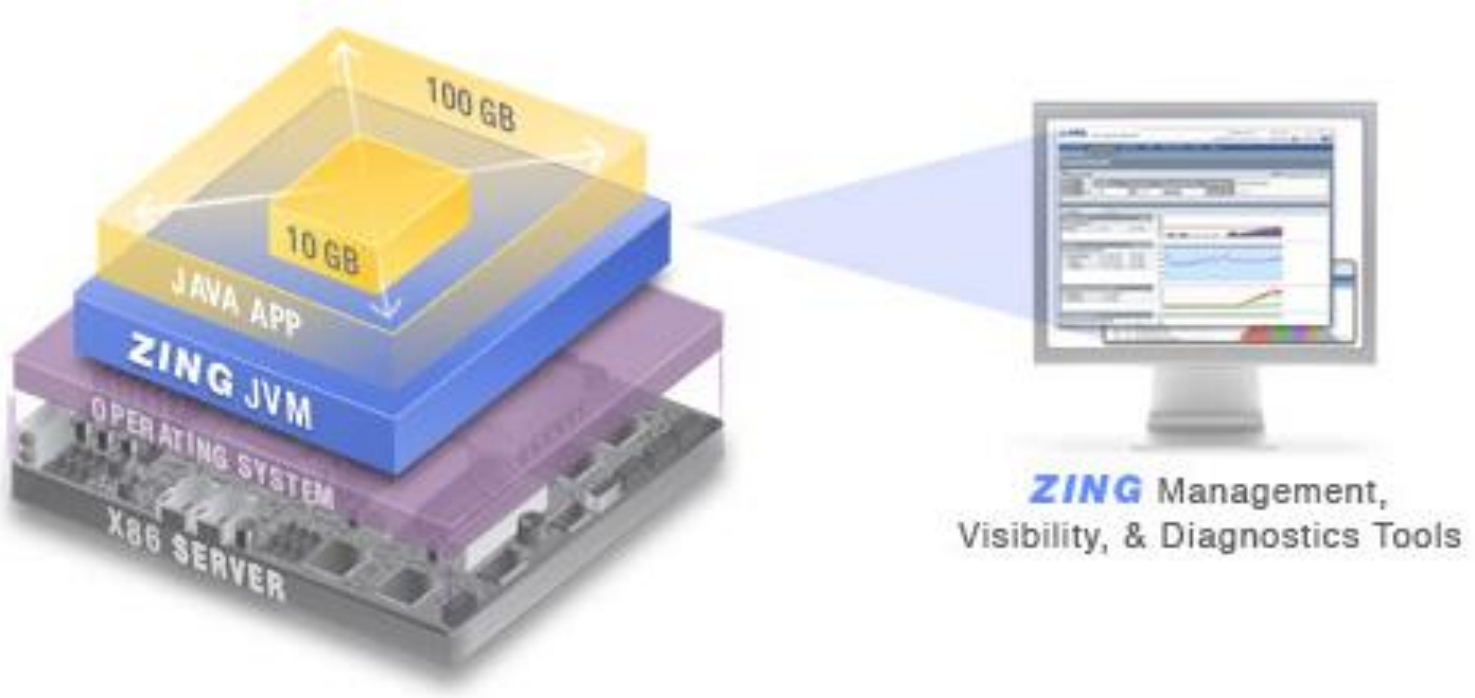


Optimizing GC-throughput

(HP-Loadrunner)

GC in Practice (3)

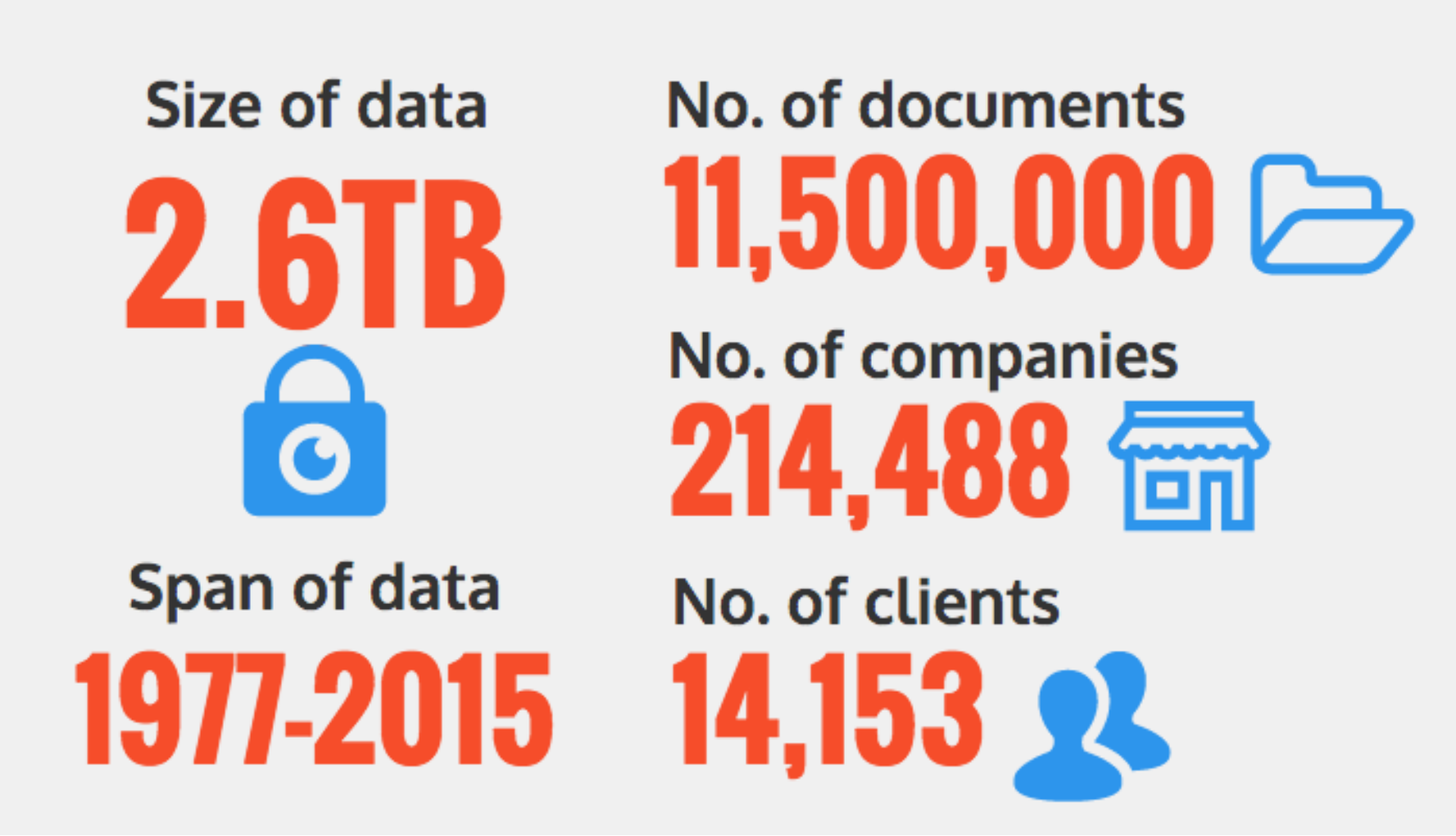
- Software Systems in Finance (Azul Zing JVM)
- C4 – Garbage Collector



GC in Practice (4) – Panama Papers

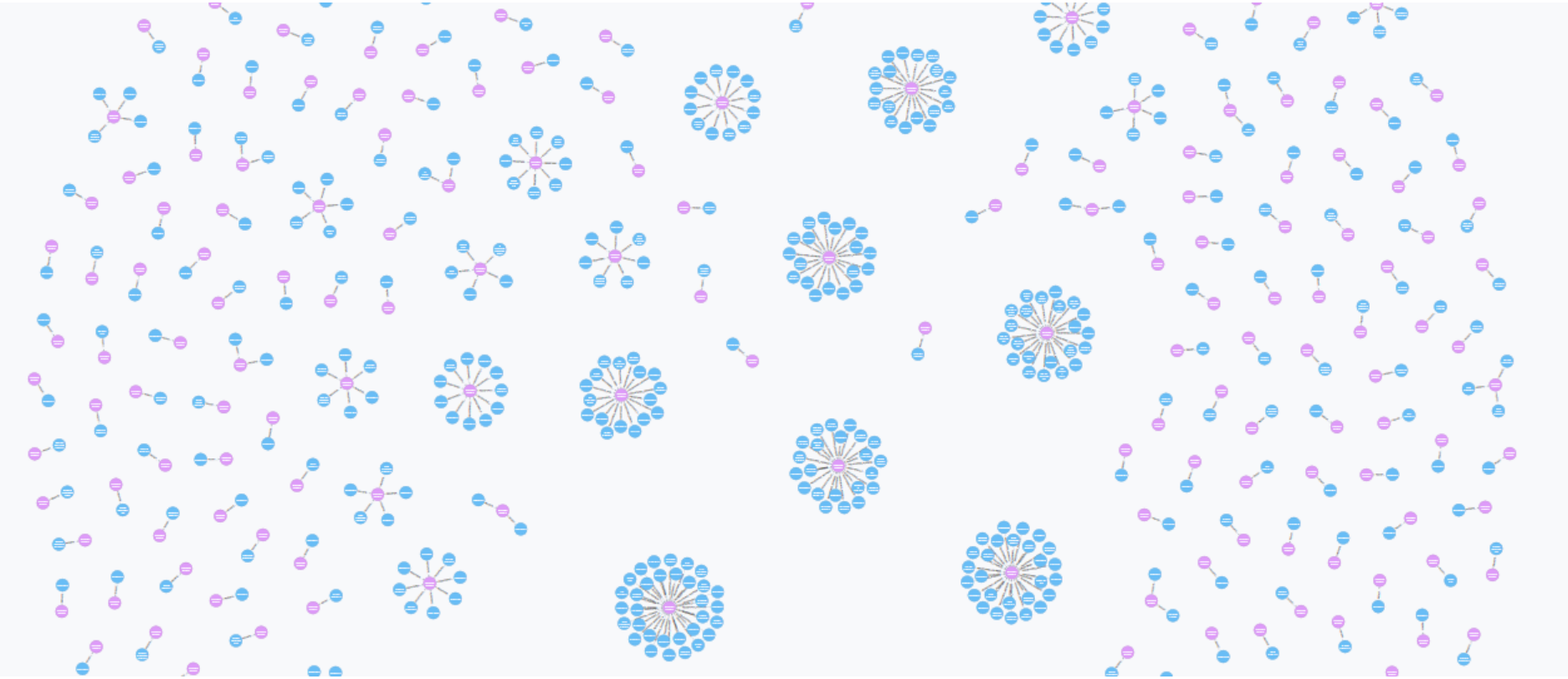
[18]

GC in Practice (4) – Panama Papers

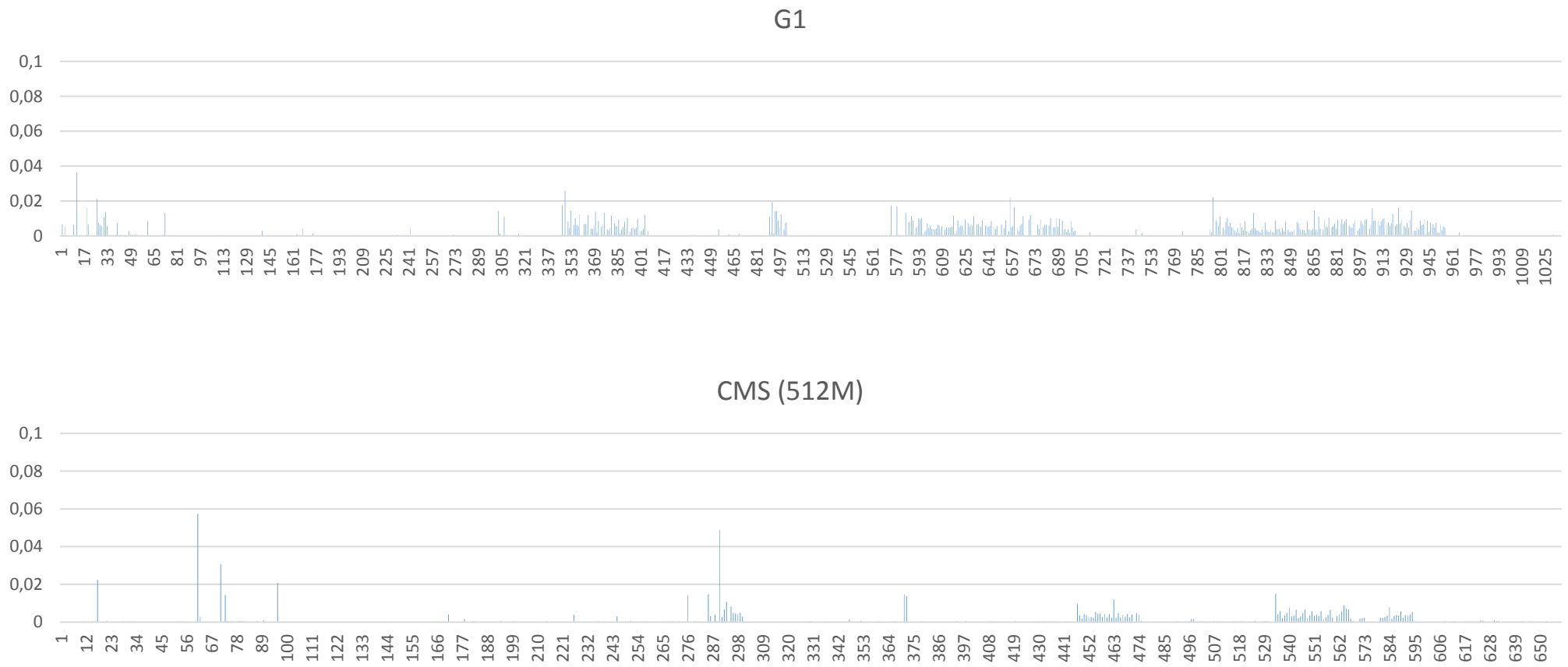


… analyzing on Neo4J

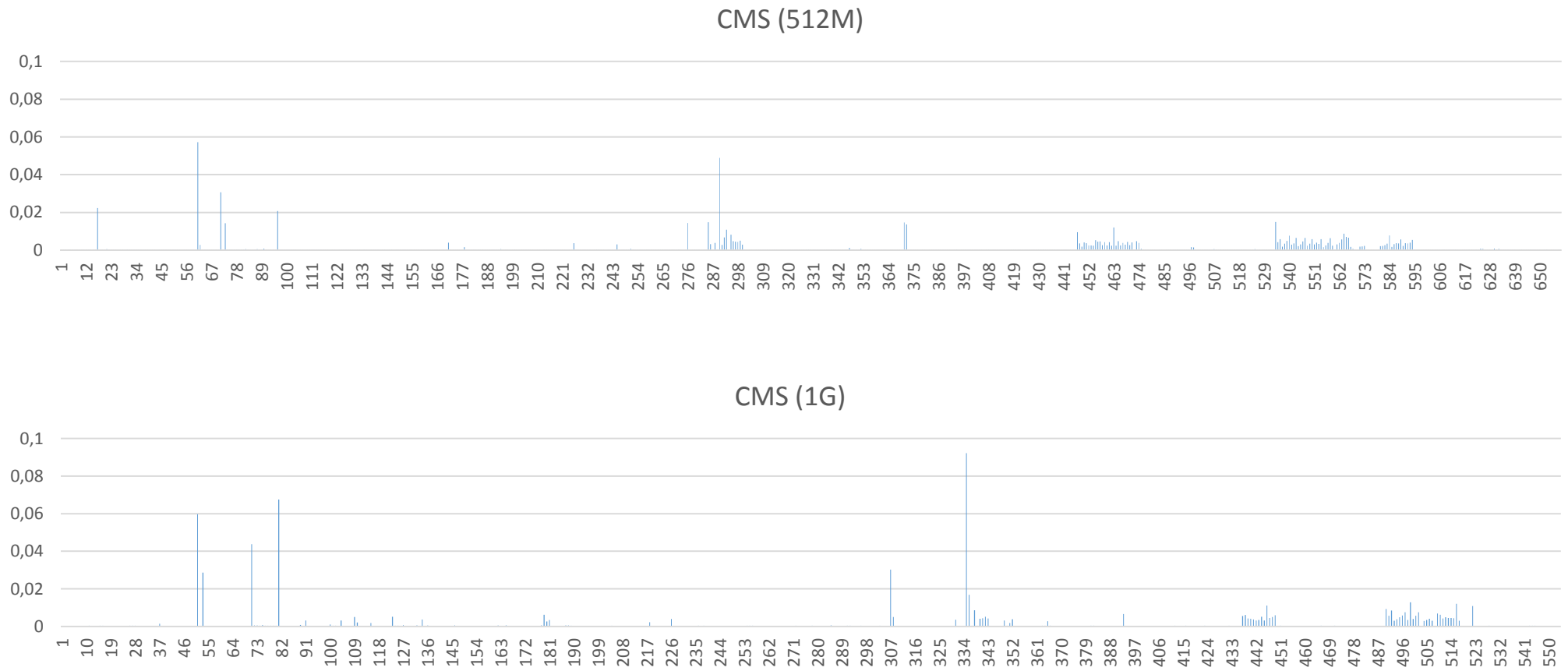
```
MATCH (a:Entity)-[r:REGISTERED_ADDRESS]-(b:Address) RETURN a,r,b LIMIT 1000
```



… analyzing on Neo4J (G1 vs. CMS)



… analyzing on Neo4J (CMS)



Lessons Learned

- Macro-Tuning vs. Micro-Tuning

effort	success
20 %	80 %
80 %	20 %

Pareto-Principle

Questions?

Sources

[1] https://www.kuriose-feiertage.de/internationaler-tag-der-muellabfuhr/

[2] https://cdn-images-1.medium.com/max/2000/1*iIXOmGDzrtTJmdwbn7cGMw.png

[3] https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/index.html

[4] https://en.wikipedia.org/wiki/Go_(programming_language)

[5] https://en.wikipedia.org/wiki/Python_(programming_language)

[6] https://en.wikipedia.org/wiki/JRuby

[7] https://en.wikipedia.org/wiki/Apache_Groovy

[8] https://en.wikipedia.org/wiki/Kotlin_(programming_language)

[9] https://de.wikipedia.org/wiki/Scala_(Programmiersprache)

[10] https://en.wikipedia.org/wiki/.net

[11] https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

[12] Java Core Programmierung: Memory Model und Garbage Collection, 2011, entwickler.press, Angelika Langer und Klaus Kreft

[13] https://www.oracle.com/technetwork/articles/java/vmoptions-jsp-140102.html

[14] https://medium.com/@michael.aleithe/be-precise-in-measuring-garbage-collection-latencies-f53e46d39132

[15] https://www.xceptance.com/de/

[16] Minimizing Garbage Collection Latencies on a Load Testing Software, Juni 2014, Testing Experience, Ausgabe 26, S.39; Michael Aleithe

[17] https://www.azul.com/

[18] https://www.irishtimes.com/business/panama-papers

[19] https://www.pri.org/stories/2016-04-04/9-statistics-explain-magnitude-panama-papers