

Profiling cz. 1 - kręgi piekła profilingu (JAVA JProfiler G1GC)

Krzysztof Ślusarski

Krótko o mnie

- Programuję od 1992

Krótko o mnie

- Programuję od 1992
- Zawodowo:
 - Od 10.2006 – 08.2007 – RODO
 - Od 09.2007 – Britenet sp z. o. o
 - Java Programmer /
 - Team Leader /
 - System Architect /
 - Solution Architect

Krótko o mnie

- Poza 8h 5/7:

Krótko o mnie

- Poza 8h 5/7:
 - Szkolenia
 - Bebechy JVM
 - Tuning JVM
 - Wycieki pamięci
 - Profiling

Krótko o mnie

- Poza 8h 5/7:
 - Szkolenia
 - Bebechy JVM
 - Tuning JVM
 - Wycieki pamięci
 - Profiling
 - Diagnoza awarii produkcyjnych

Krótko o mnie

- Poza 8h 5/7:
 - Szkolenia
 - Bebechy JVM
 - Tuning JVM
 - Wycieki pamięci
 - Profiling
 - Diagnoza awarii produkcyjnych
 - Profiling

Krótko o mnie

- Poza 8h 5/7:
 - Szkolenia
 - Bebechy JVM
 - Tuning JVM
 - Wycieki pamięci
 - Profiling
 - Diagnoza awarii produkcyjnych
 - Profiling
 - Stolarstwo meblowe

Krótko o mnie

- Prywatnie:



Czym jest profiling?

In software engineering, profiling ("program profiling", "software profiling") is a form of dynamic program analysis that measures, for example, the space (memory) or time complexity of a program, the usage of particular instructions, or the frequency and duration of function calls. Most commonly, profiling information serves to aid program optimization.

Wikipedia

Czym jest profiling?



Zrozumienie

Który case omawiamy?

Który case omawiamy?

- Aplikacja działa stabilnie

Który case omawiamy?

- Aplikacja działa stabilnie
- Aplikacja działa poprawnie

Który case omawiamy?

- Aplikacja działa stabilnie
- Aplikacja działa poprawnie
- Chcemy, żeby działała szybciej

Który case omawiamy?

- Aplikacja działa stabilnie
- Aplikacja działa poprawnie
- Chcemy, żeby działała szybciej
- Nie da się zrzucić winy gdzie indziej:
 - DB
 - Usługi

Kiedy?

Kiedy?

- Zgłoszenia, że „działa wolno”

Kiedy?

- Zgłoszenia, że „działa wolno”
 - Konkretne „coś”, np. :
 - Wolno ładuje się strona
 - Wolno działa request http
 - Wolno działa wystawiona usługa REST/SOAP

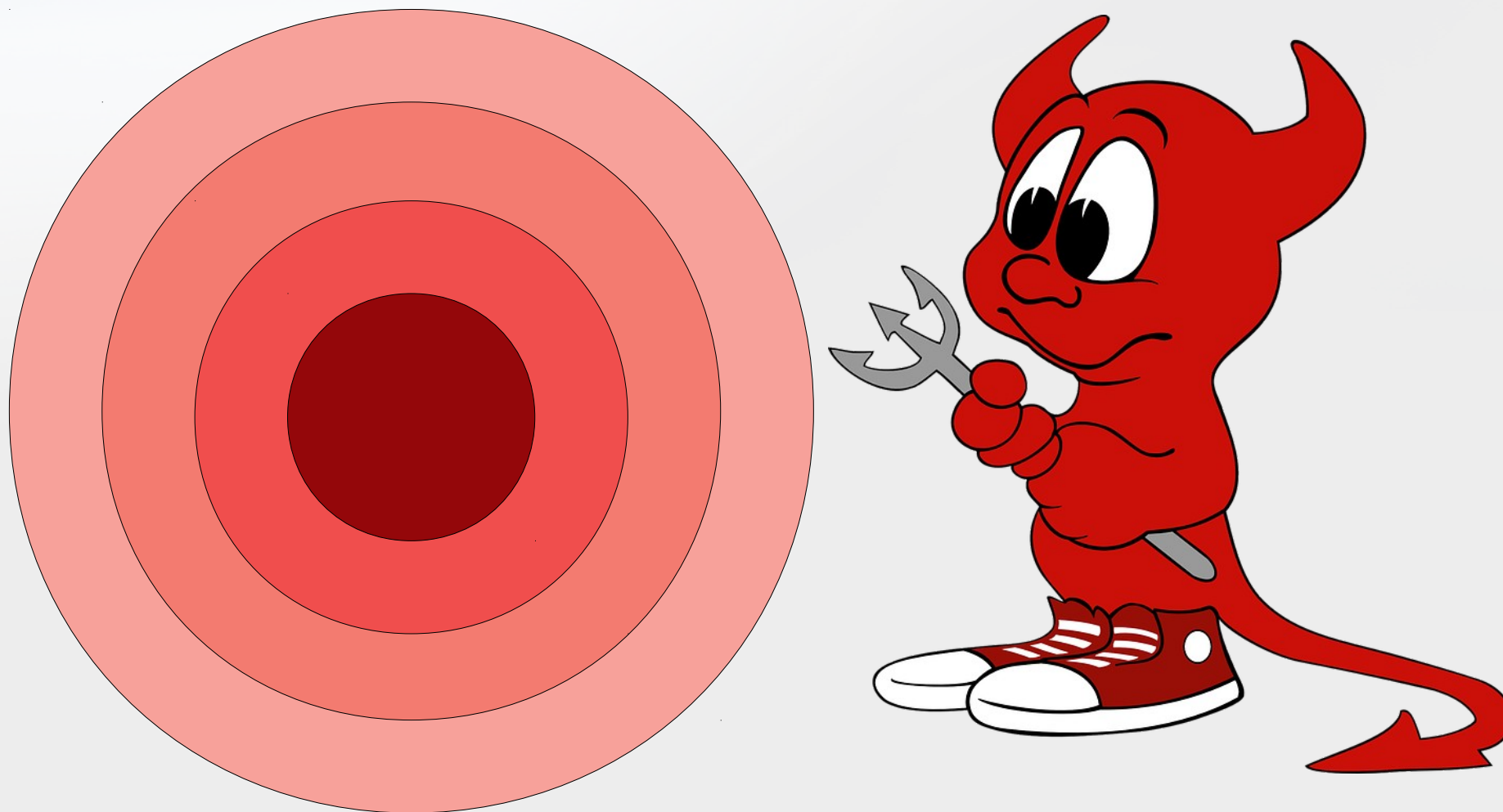
Kiedy?

- Zgłoszenia, że „działa wolno”
 - Konkretnie „coś”, np. :
 - Wolno ładuje się strona
 - Wolno działa request http
 - Wolno działa wystawiona usługa REST/SOAP
 - Cała aplikacja

Kiedy?

- Zgłoszenia, że „działa wolno”
 - Konkretnie „coś”, np. :
 - Wolno ładuje się strona
 - Wolno działa request http
 - Wolno działa wystawiona usługa REST/SOAP
 - Cała aplikacja
- Mamy na to czas

Kręgi piekła



Krąg 1

Profilowana
aplikacja



Co mogliśmy zrobić źle?

Co mogliśmy zrobić źle?

- Nieodpowiednie struktury danych do problemu

Co mogliśmy zrobić źle?

- Nieodpowiednie struktury danych do problemu
- Nieefektywne algorytmy

Co mogliśmy zrobić źle?

- Nieodpowiednie struktury danych do problemu
- Nieefektywne algorytmy
- Powtarzanie tych samych operacji – brak cache-a

Co mogliśmy zrobić źle?

- Nieodpowiednie struktury danych do problemu
- Nieefektywne algorytmy
- Powtarzanie tych samych operacji – brak cache-a
- Niepotrzebny kod

Co mogliśmy zrobić źle?

- Nieodpowiednie struktury danych do problemu
- Nieefektywne algorytmy
- Powtarzanie tych samych operacji – brak cache-a
- Niepotrzebny kod
- Za dużo locków

Czym?

Czym?

- JProfiler – 420 euro + VAT
- YourKit Java Profiler – 89/459 euro + VAT

Czym?

- JProfiler – 420 euro + VAT
- YourKit Java Profiler – 89/459 euro + VAT
- VisualVM – darmowy
- Mission Control – darmowy od JDK 11
- Async Profiler – darmowy
- Perf + Perf map agent – darmowy
- Custom
- ...

Jakie mamy opcje?

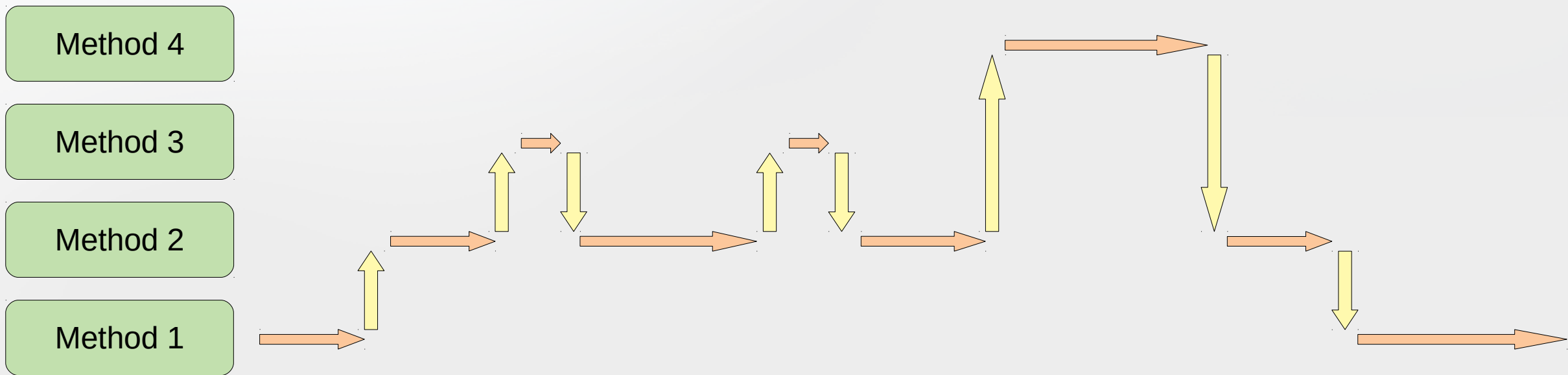
Sampling

Jakie mamy opcje?

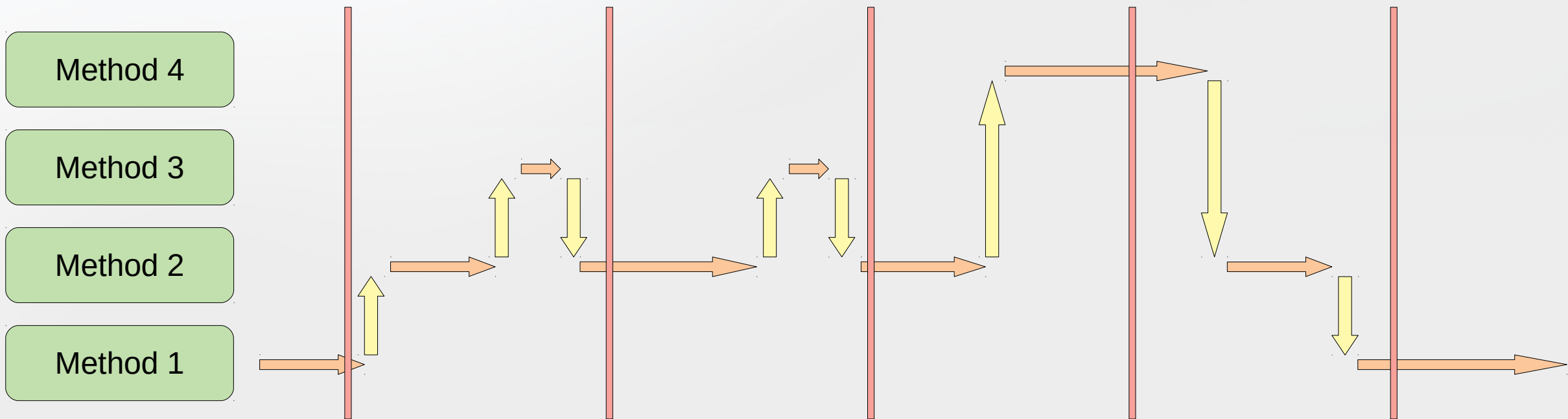
Sampling

Instrumentation

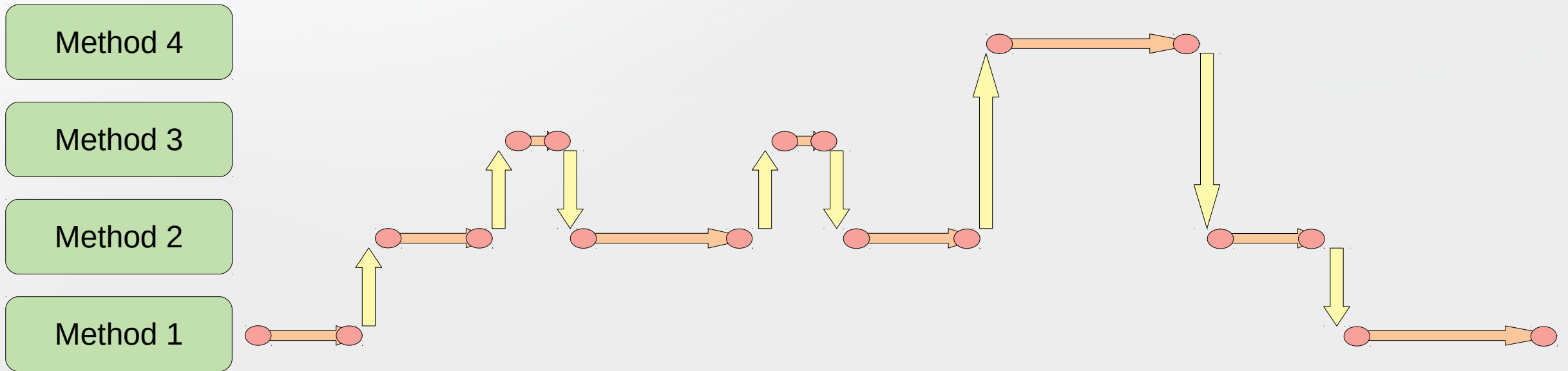
Sampling



Sampling



Instrumentation



Cała aplikacja wolno działa - przykład

Krąg 2

Używane
biblioteki,
frameworki
i serwery



Jakie mamy opcje?

- Wymiana

Jakie mamy opcje?

- Wymiana
- Custom – napisanie lepiej

Jakie mamy opcje?

- Wymiana
- Custom – napisanie lepiej
- Proxy / dekorator

Jakie mamy opcje?

- Wymiana
- Custom – napisanie lepiej
- Proxy / dekorator
- Używać zgodnie z przeznaczeniem/dokumentacją

Wymiana biblioteki - przykład

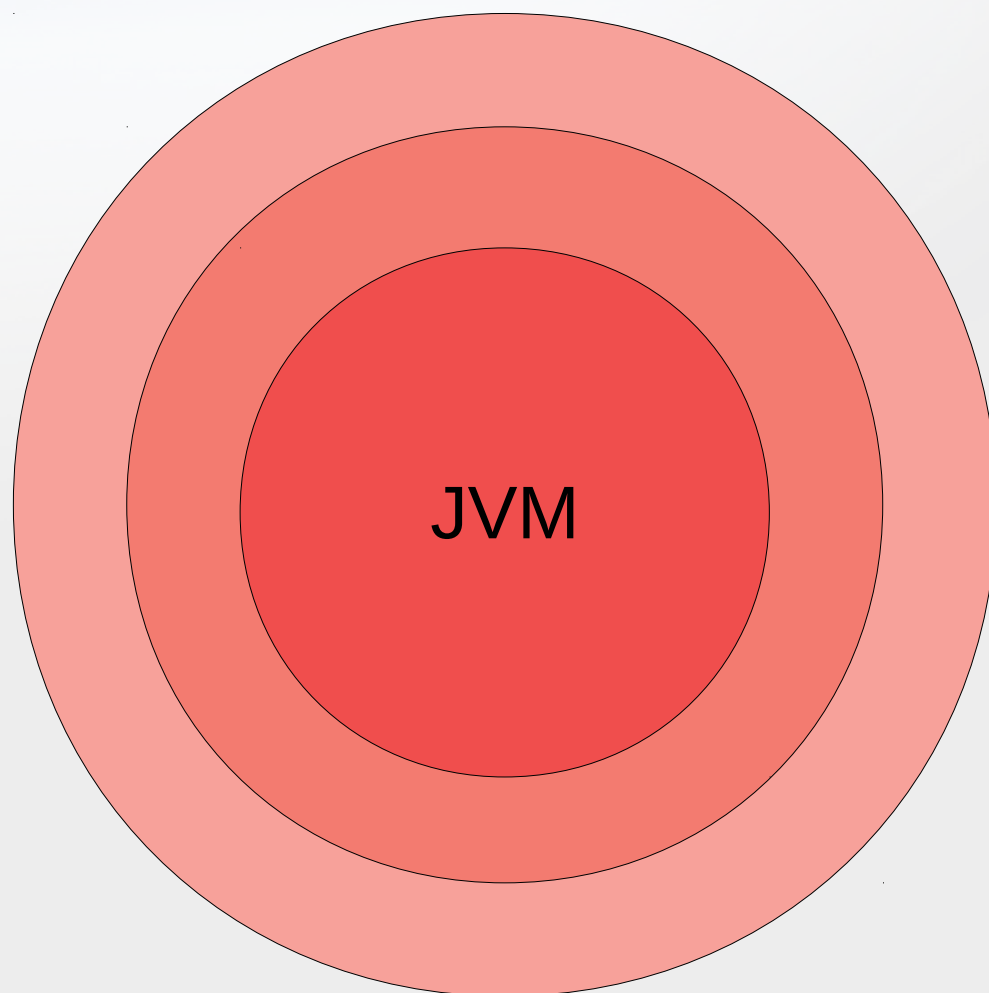
Teoria a życie



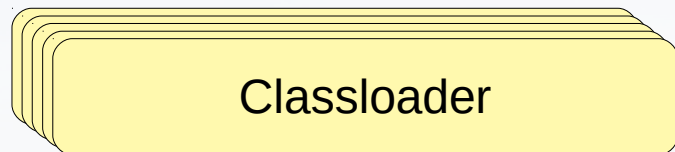
VS



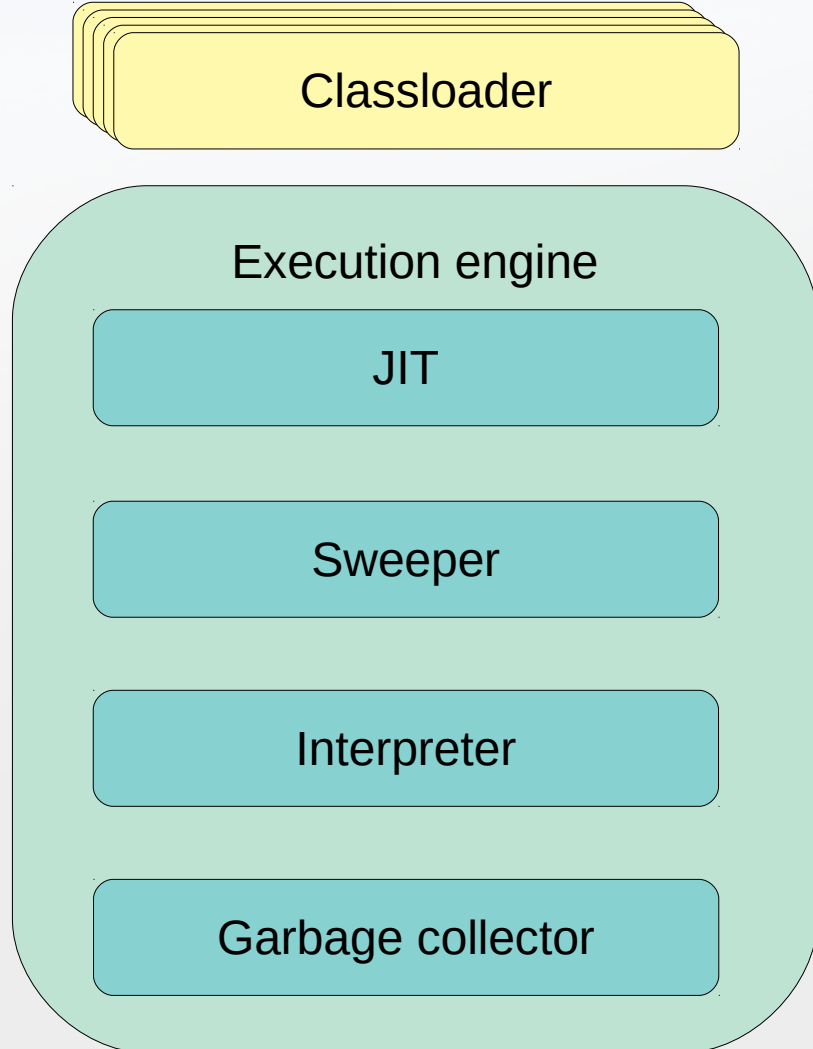
Kręgi piekła



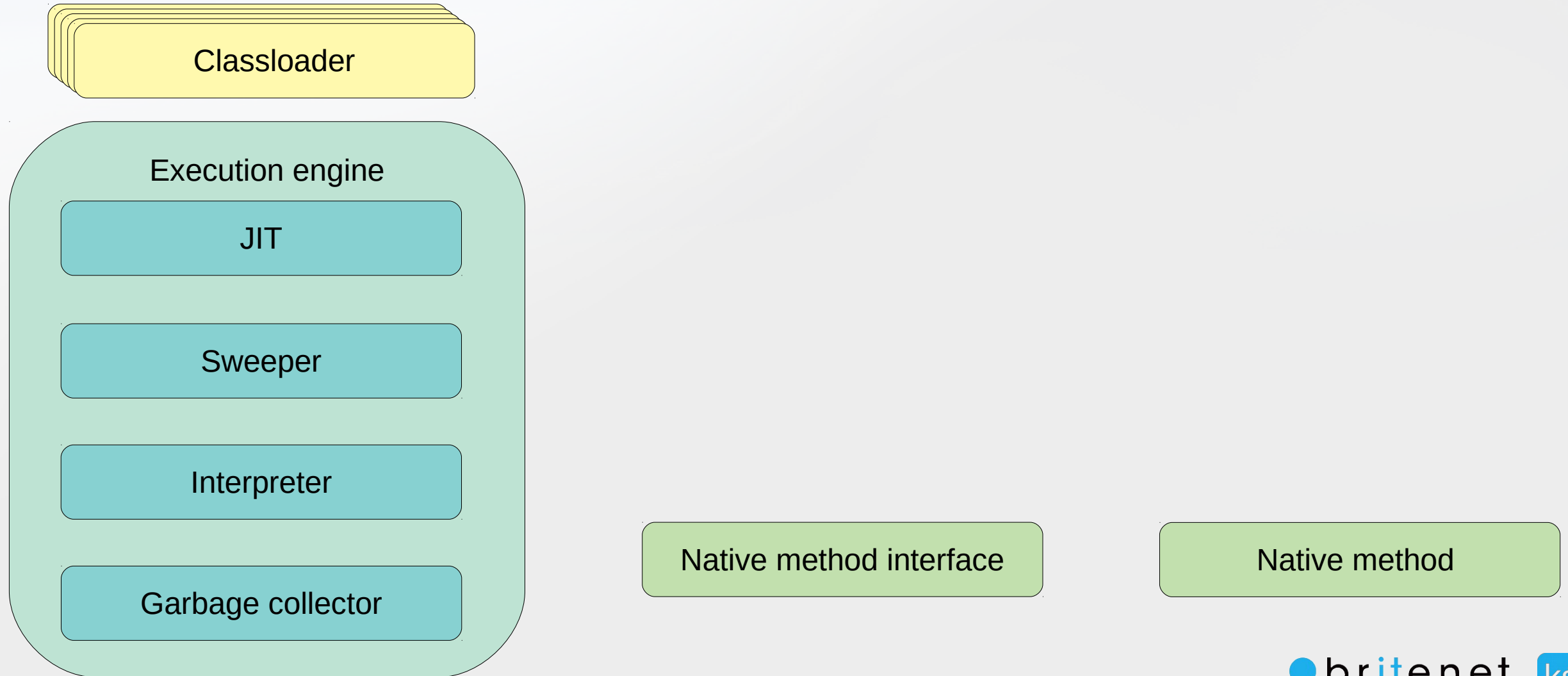
JVM - architektura



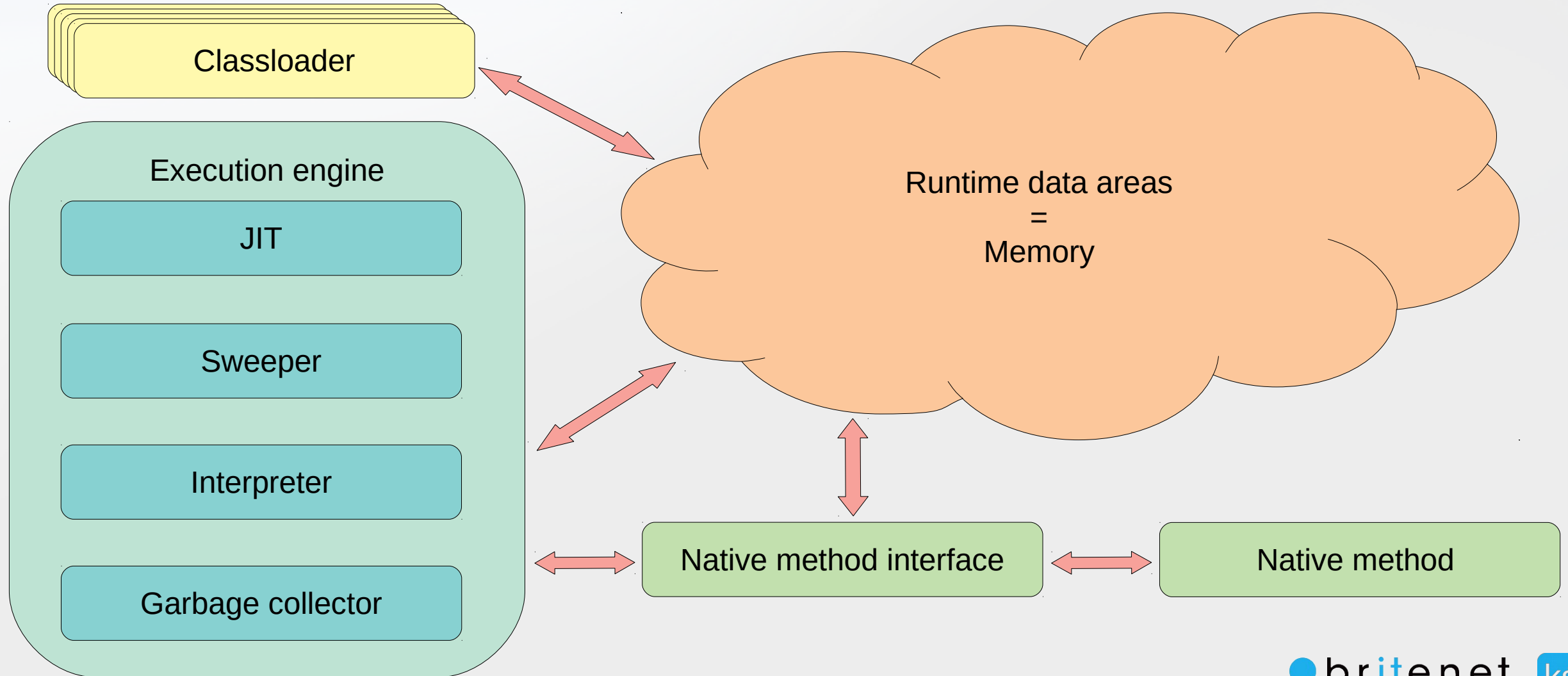
JVM - architektura



JVM - architektura



JVM - architektura

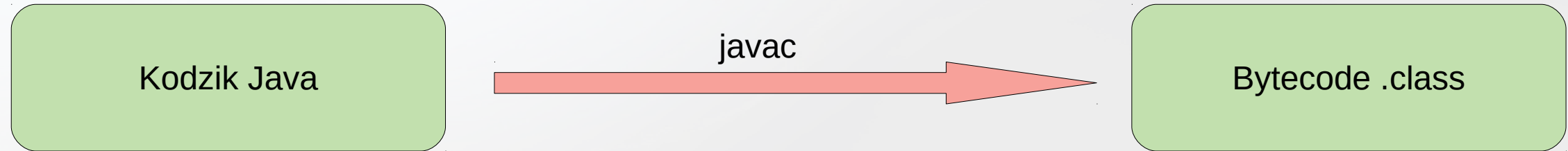


Cykl życia kodziku

Cykl życia kodziku

Kodziki Java

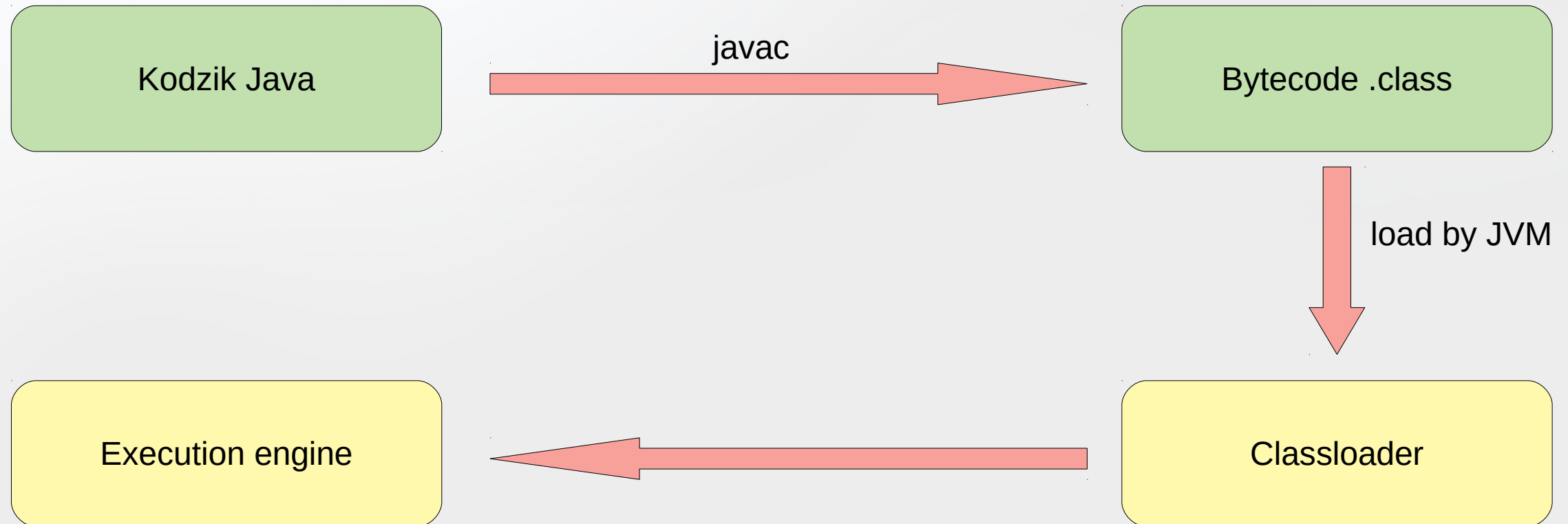
Cykl życia kodziku



Cykl życia kodziku



Cykl życia kodziku

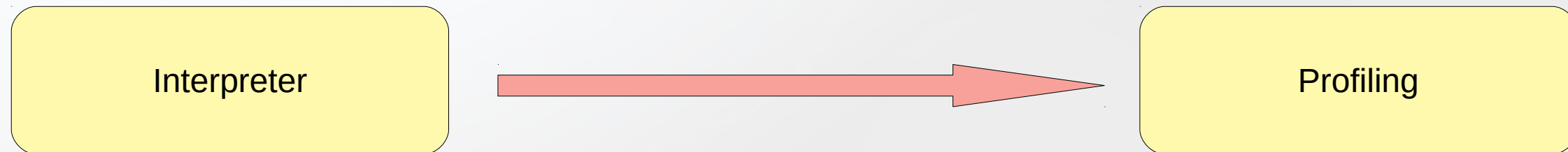


Cykl życia kodziku

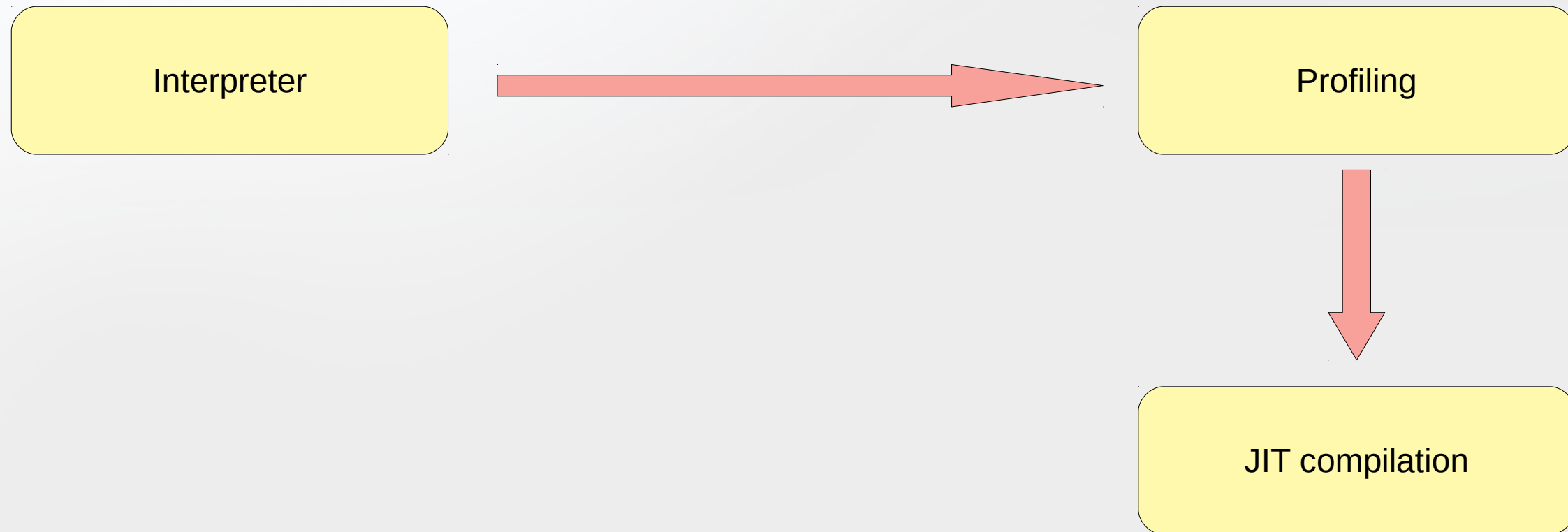
Cykl życia kodziku

Interpreter

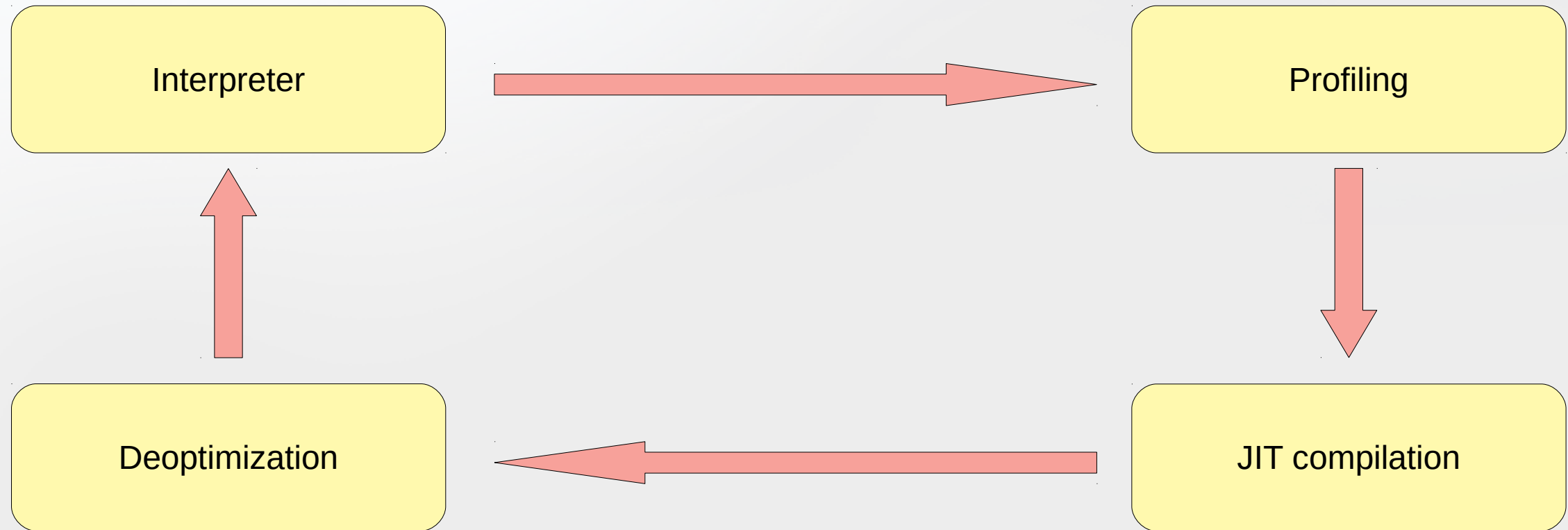
Cykl życia kodziku



Cykl życia kodziku



Cykl życia kodziku



JIT

- Bardzo dużo różnych optymalizacji

JIT

- Bardzo dużo różnych optymalizacji
- Przekroczenie granicy
 - Inlining – 325 byte
 - Compilation – 8000 byte, liczba argumentów

JVM - wykonywanie zadań

JVM - wykonywanie zadań

- Local safepoint

JVM - wykonywanie zadań

- Local safepoint
- Global safepoint

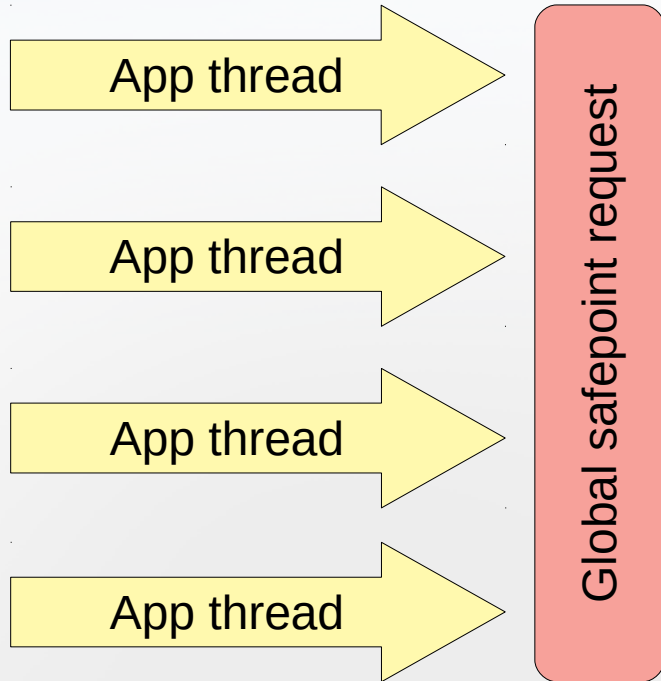
JVM - wykonywanie zadań

- Local safepoint
- Global safepoint
- Safepoint operation

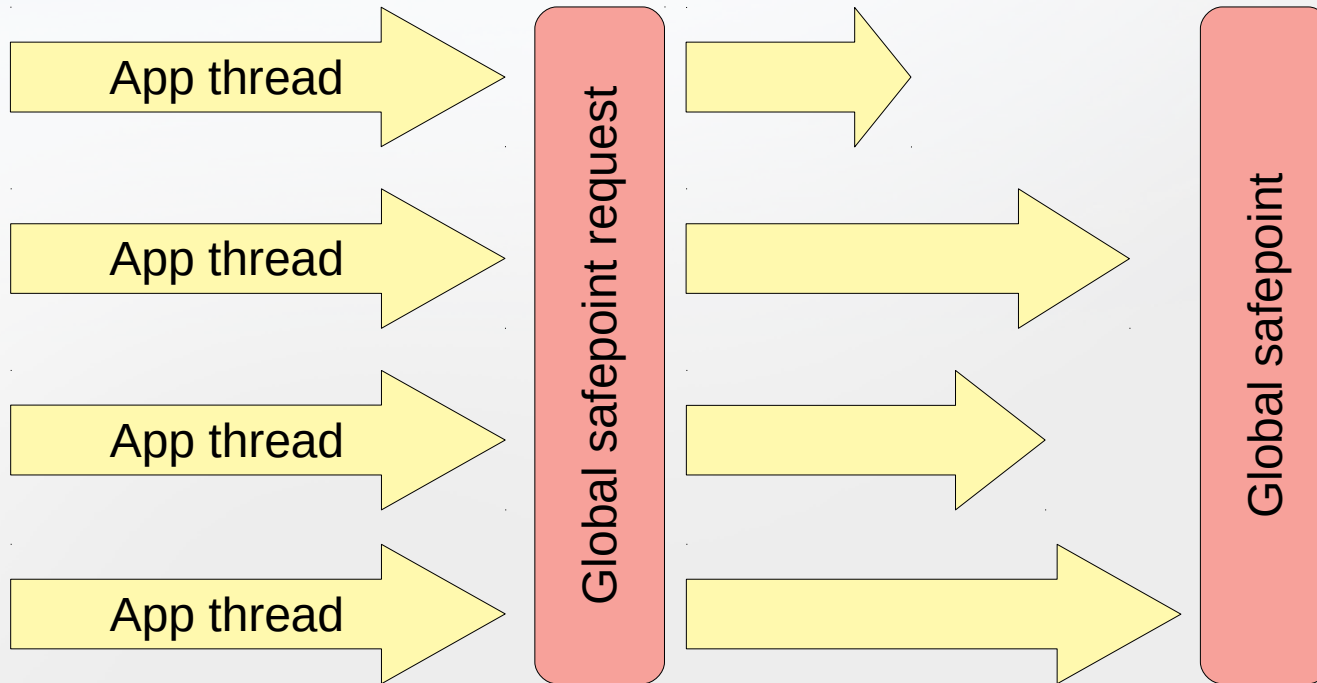
JVM - wykonywanie zadań

- Local safepoint
- Global safepoint
- Safepoint operation
- Stop the world

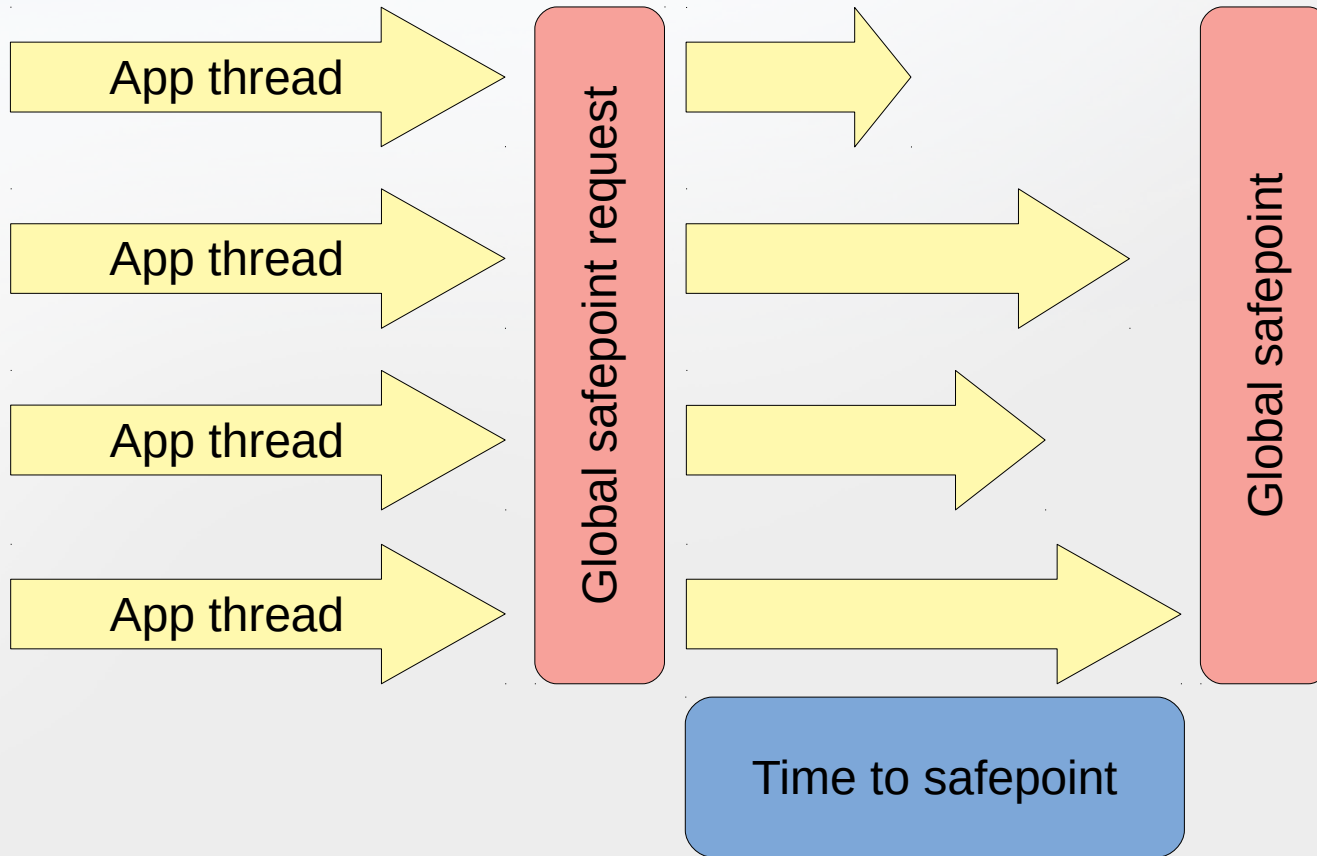
Stop the world



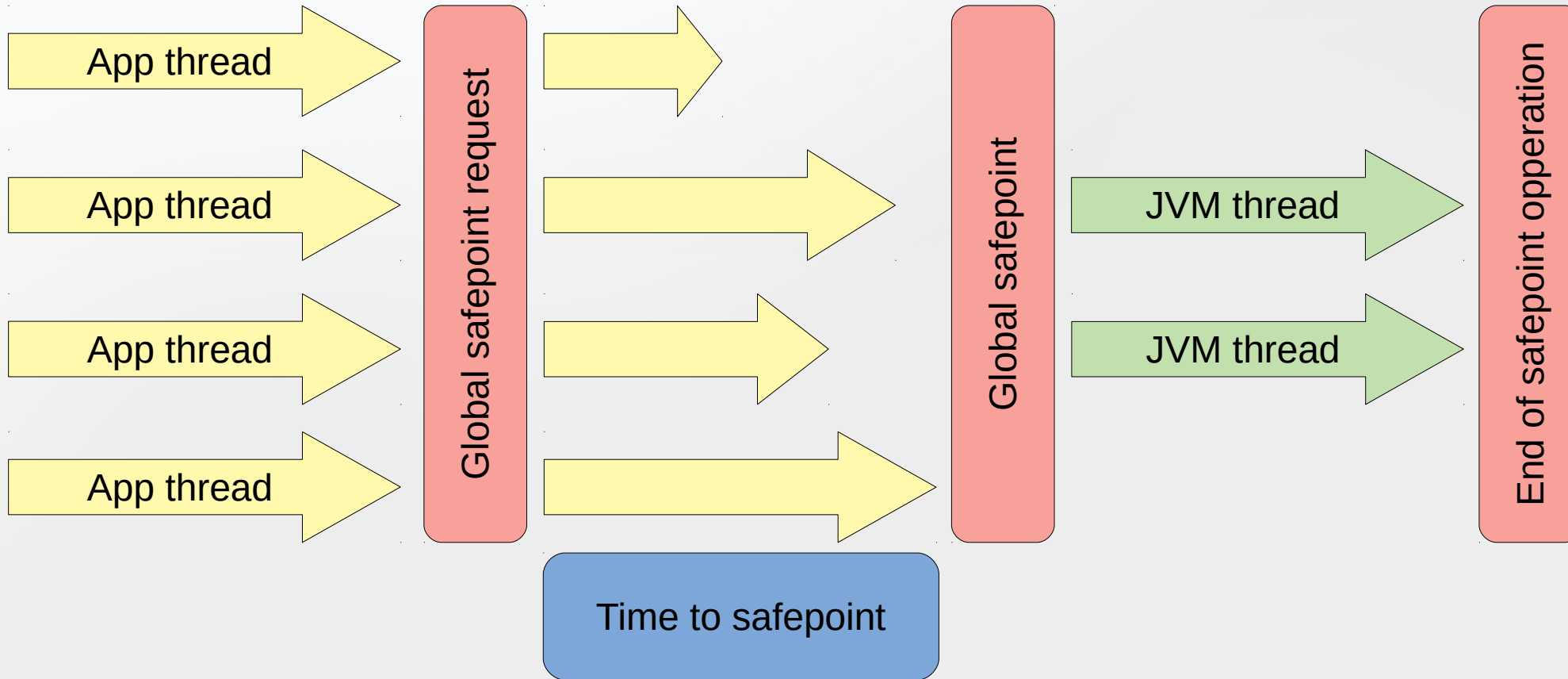
Stop the world



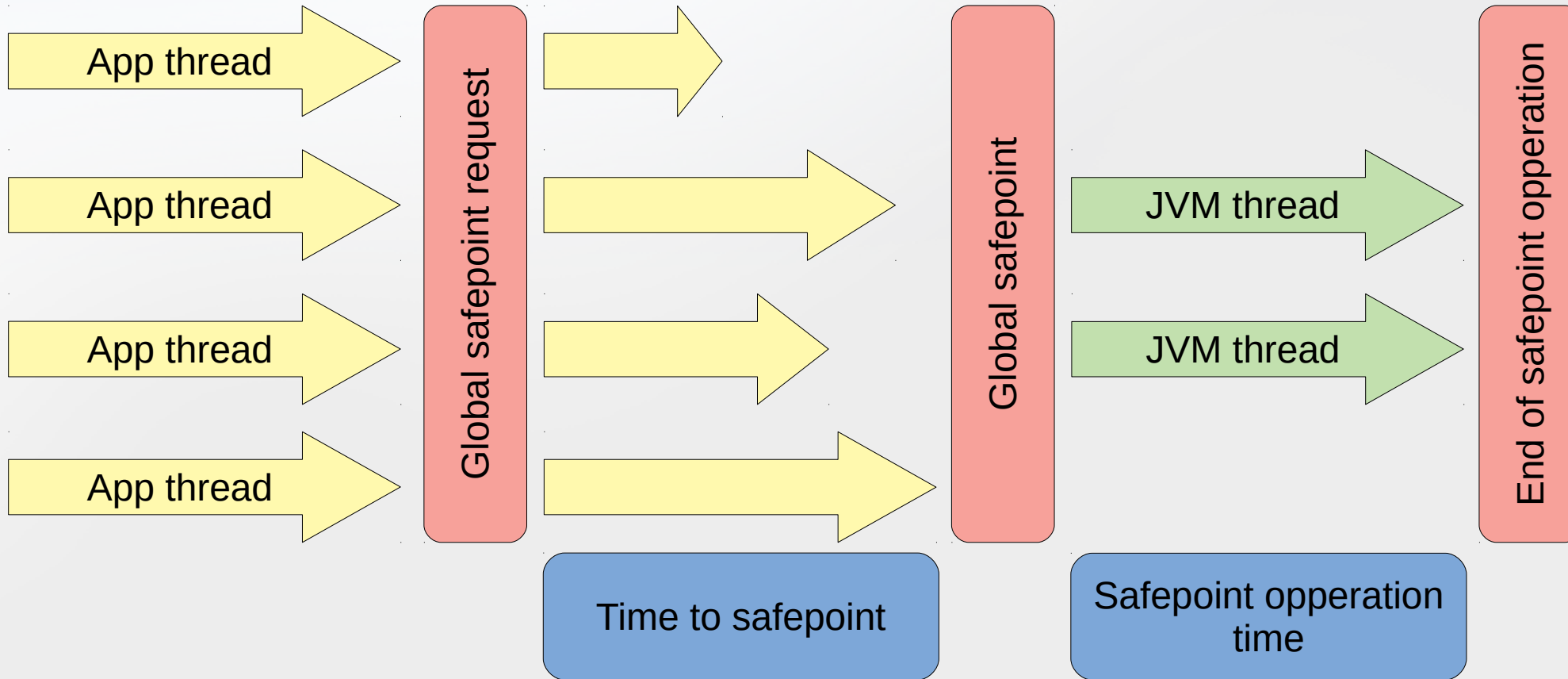
Stop the world



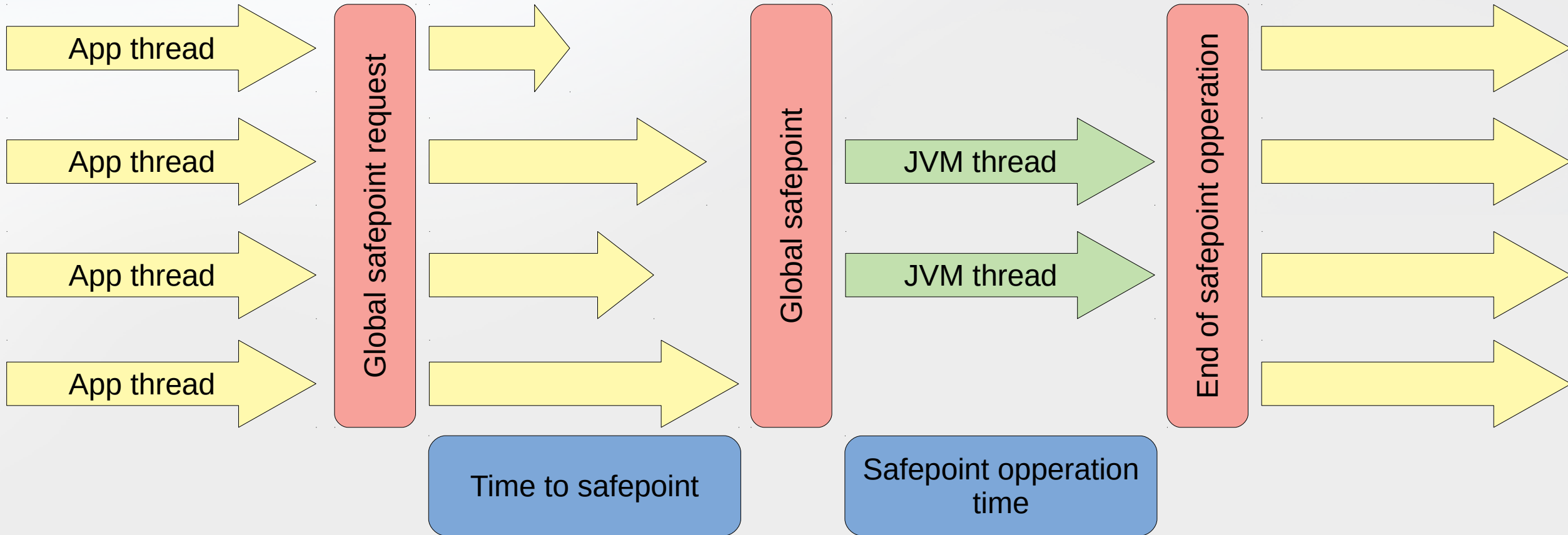
Stop the world



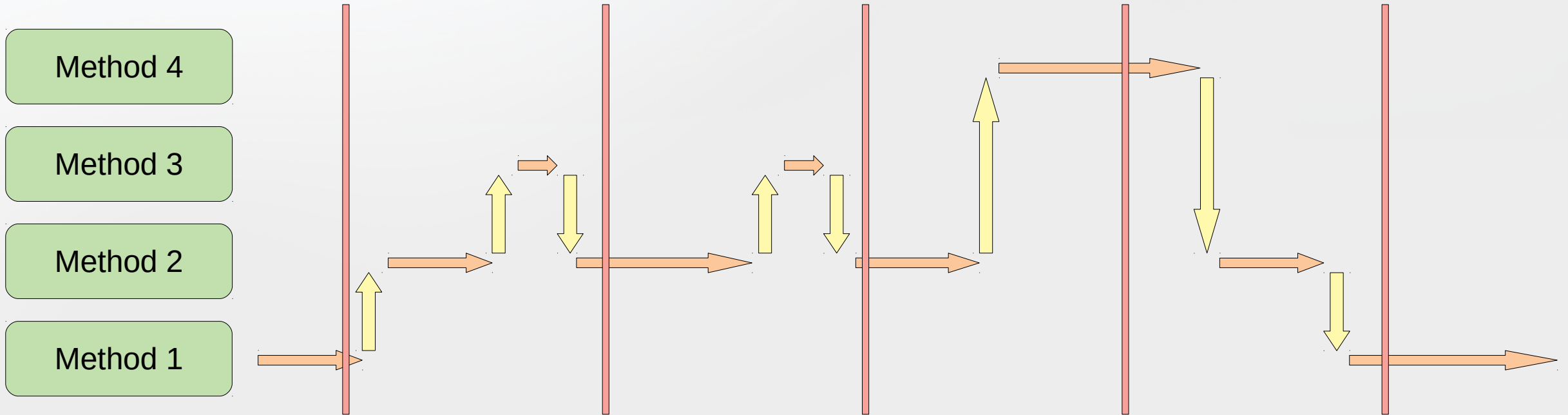
Stop the world



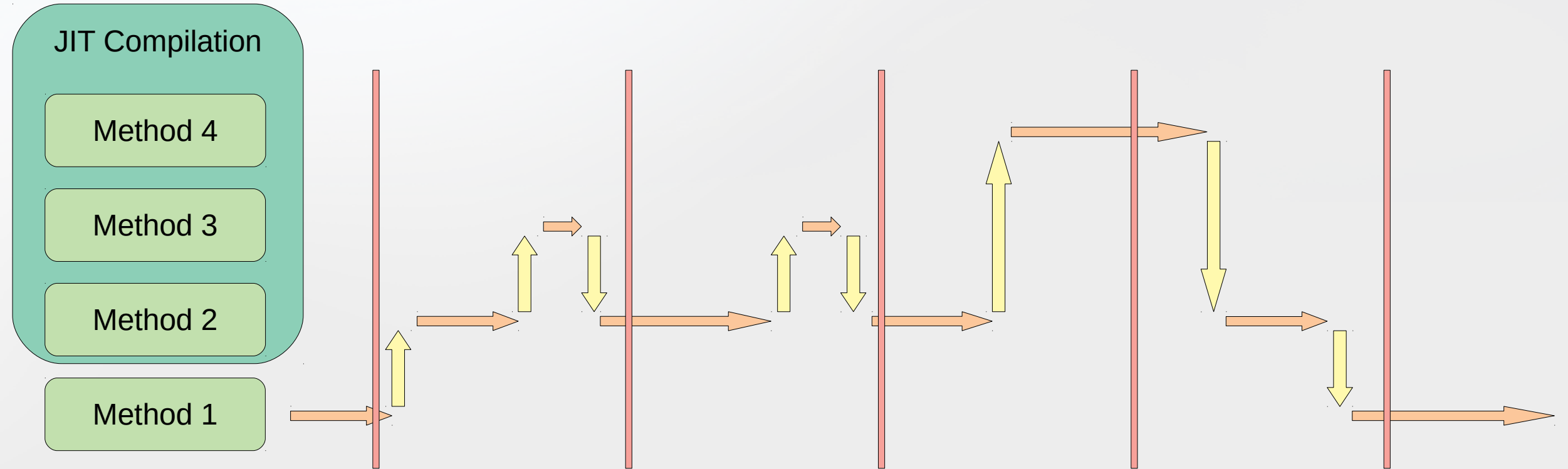
Stop the world



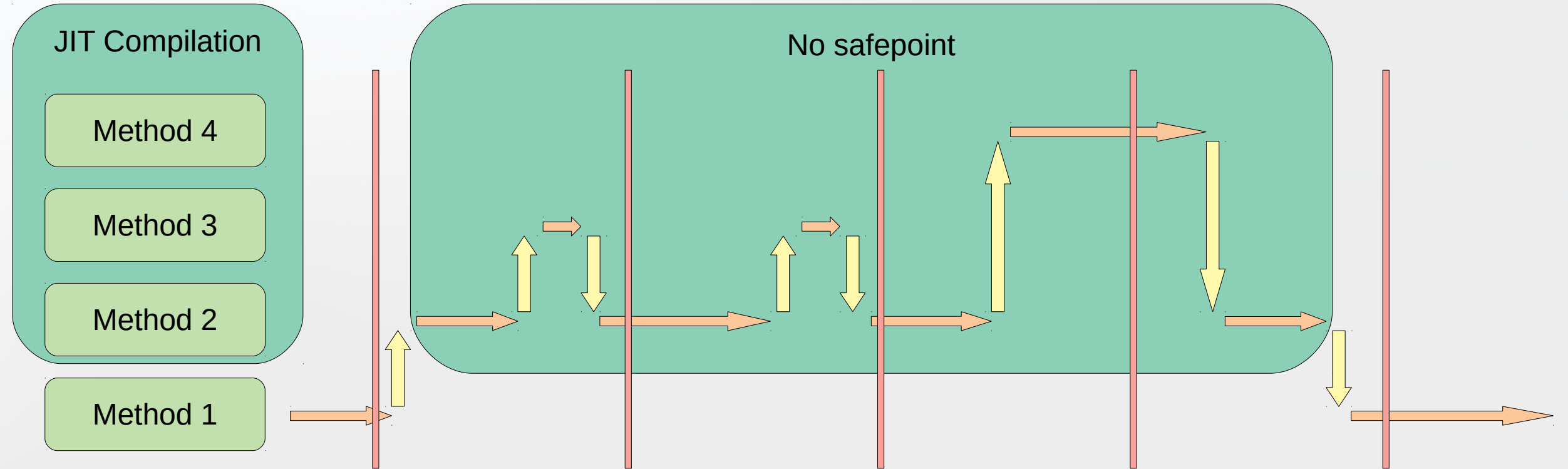
Sampling - the dark side



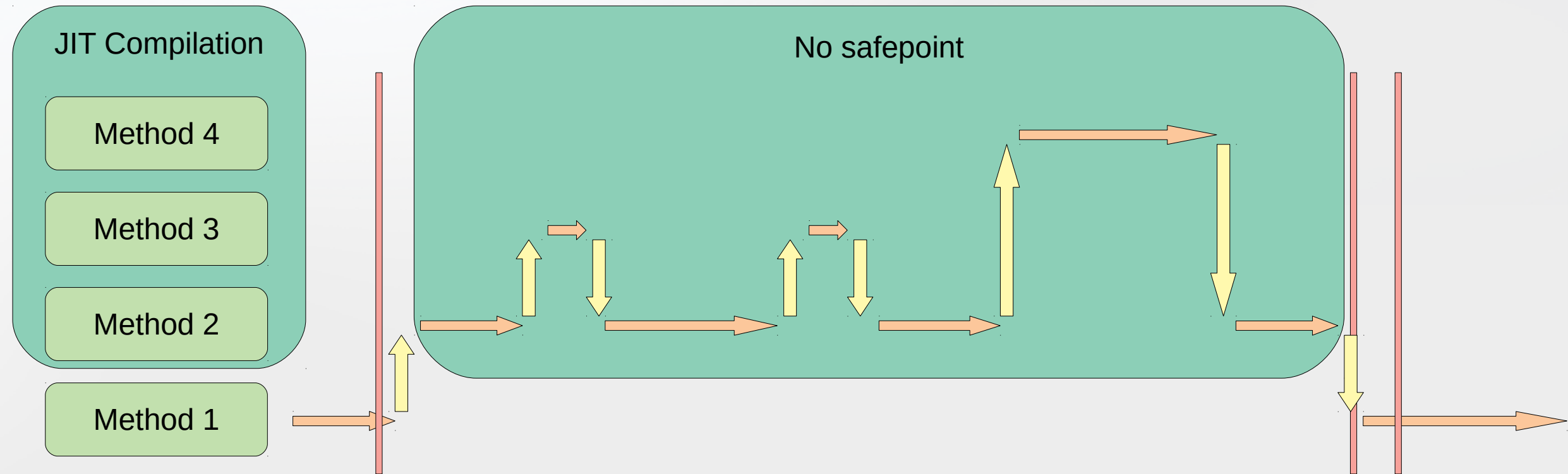
Sampling - the dark side



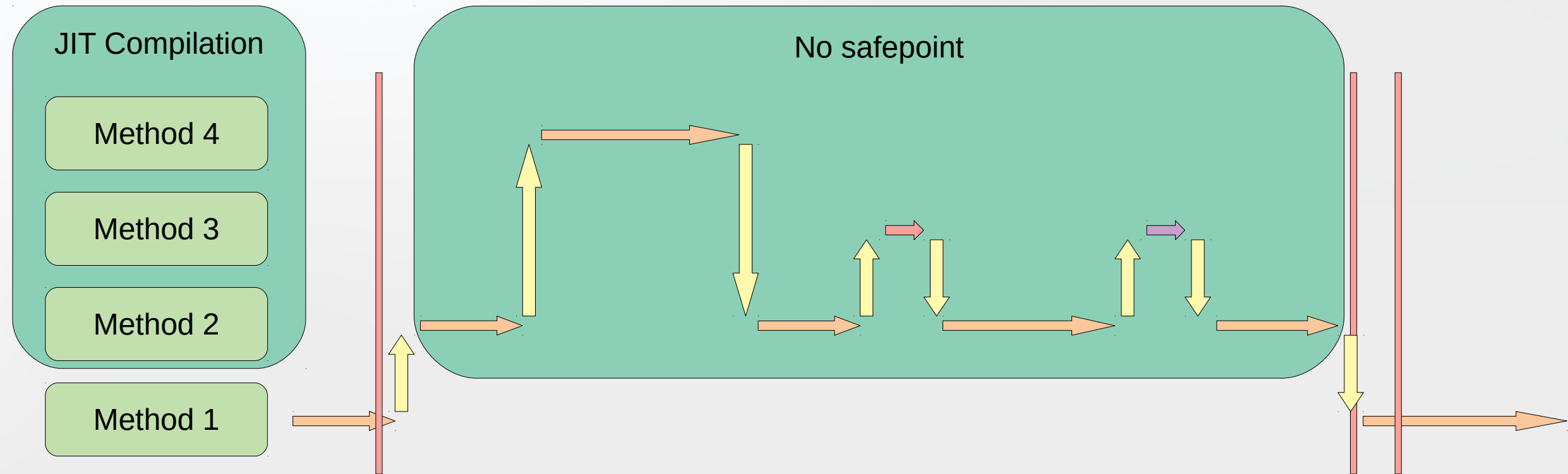
Sampling - the dark side



Sampling - the dark side



Sampling - the dark side



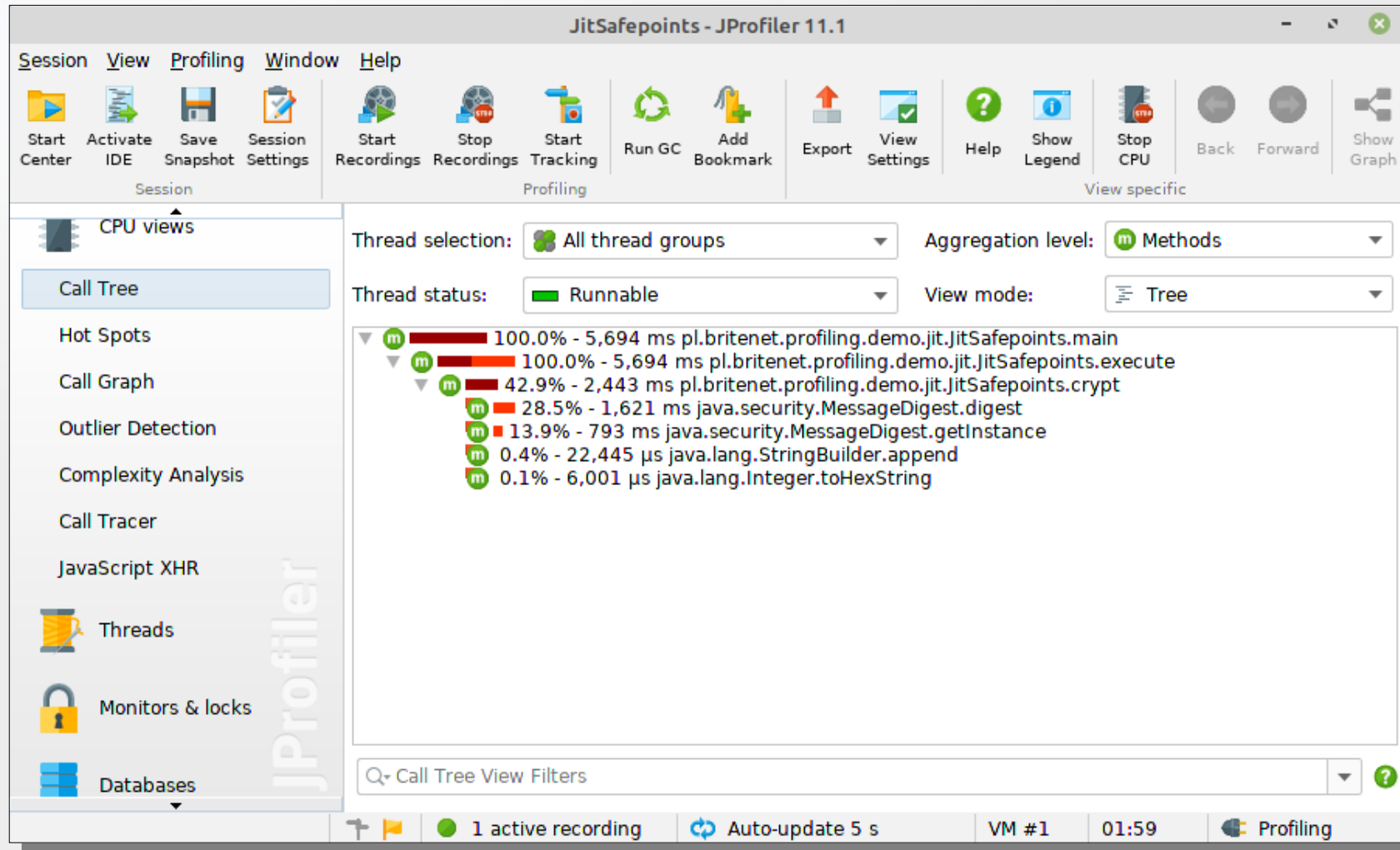
Sampling failure

```
private void execute() {  
    long N = 5 * 1000 * 1000;  
  
    RandomStringUtils randomStringUtils =  
        new RandomStringUtils();  
  
    for (long i = 0; i < N; i++) {  
        String text = randomStringUtils.generate();  
        crypt(text);  
    }  
}
```

Sampling failure

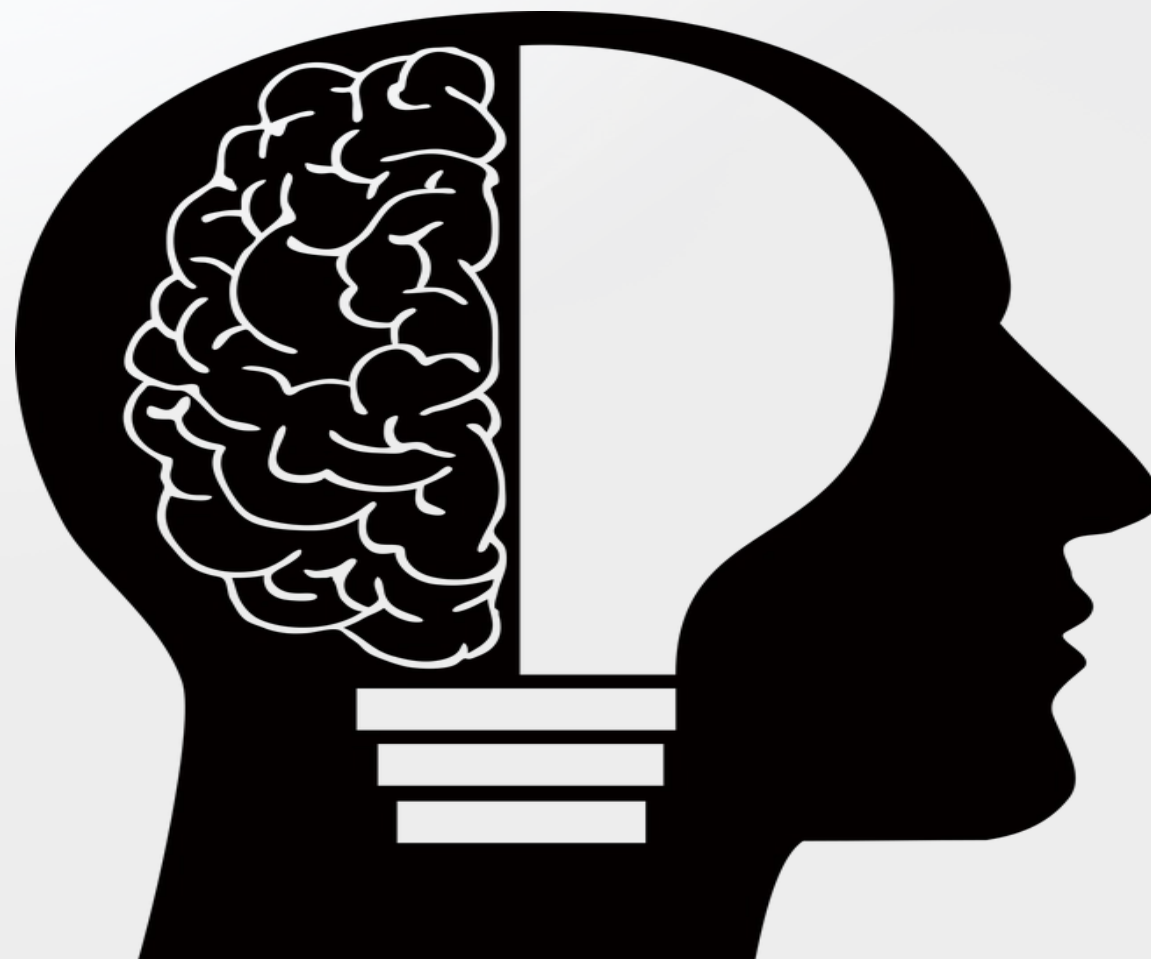
```
private void execute() {  
    long N = 5 * 1000 * 1000;  
  
    RandomStringUtils randomStringUtils =  
        new RandomStringUtils();  
  
    for (long i = 0; i < N; i++) {  
        String text = randomStringUtils.generate();  
        crypt(text);  
    }  
}
```


Sampling failure



Sampling - jak żyć?

Sampling - jak żyć?



Sampling - jak żyć?

Sampling - jak żyć?

- Instrumentation / wyłączenie inliningu

Sampling - jak żyć?

- Instrumentation / wyłączenie inliningu
- Logi inliningu i kompilacji

Sampling - jak żyć?

- Instrumentation / wyłączenie inliningu
- Logi inliningu i kompilacji
- Honest Profilers
 - Honest Profiler (Linux)
 - Async Profiler (Linux, MacOS)
 - Mission Control
 - YourKit Java Profiler 2019.8 Async Sampling (Linux, MacOS)
 - JProfiler 11.1 Async Sampling (Linux, MacOS)

Sampling - jak żyć?

- Instrumentation / wyłączenie inliningu
- Logi inliningu i kompilacji
- Honest Profilers
 - Honest Profiler (Linux)
 - Async Profiler (Linux, MacOS)
 - Mission Control
 - YourKit Java Profiler 2019.8 Async Sampling (Linux, MacOS)
 - JProfiler 11.1 Async Sampling (Linux, MacOS)
- Przydatna flaga:
 - XX:+DebugNonSafepoints

Sampling - jak żyć?

The screenshot shows the JProfiler 11.1 interface. The title bar reads "JitSafepoints - JProfiler 11.1". The menu bar includes "Session", "View", "Profiling", "Window", and "Help". The toolbar is organized into three sections: "Session" (Start Center, Activate IDE, Save Snapshot, Session Settings), "Profiling" (Start Recordings, Stop Recordings, Start Tracking, Run GC, Add Bookmark, Export, View Settings), and "View specific" (Help, Show Legend, Stop CPU, Back, Forward). The left sidebar contains icons for Telemetries, Live memory, Heap Walker, CPU views, and a list of views including Call Tree (selected), Hot Spots, Call Graph, and Outlier Detection. The main panel displays the "Call Tree View Filters" section with the following data:

Thread selection:	Aggregation level:
All thread groups	Metho...

Thread status:	View mode:
Running	Tree

Method	Percentage	Time	Class
pl.britenet.profiling.demo.jit.JitSafepoints.main	100.0%	29,120 ms	pl.britenet.profiling.demo.jit.JitSafepoints.main
pl.britenet.profiling.demo.jit.JitSafepoints.execute	100.0%	29,120 ms	pl.britenet.profiling.demo.jit.JitSafepoints.execute
pl.britenet.profiling.demo.jit.JitSafepoints.crypt	82.3%	23,975 ms	pl.britenet.profiling.demo.jit.JitSafepoints.crypt
pl.britenet.profiling.demo.jit.RandomStringUtils.generate	17.4%	5,055 ms	pl.britenet.profiling.demo.jit.RandomStringUtils.generate
765 ms Sampling misses			

The status bar at the bottom indicates "1 active recording", "Auto-update 5 s", "VM #1", "00:36", and "Profiling".

Logowanie

-Xlog:safepoint

Safepoints - przykład

Teoria a życie



VS



Podstawy G1

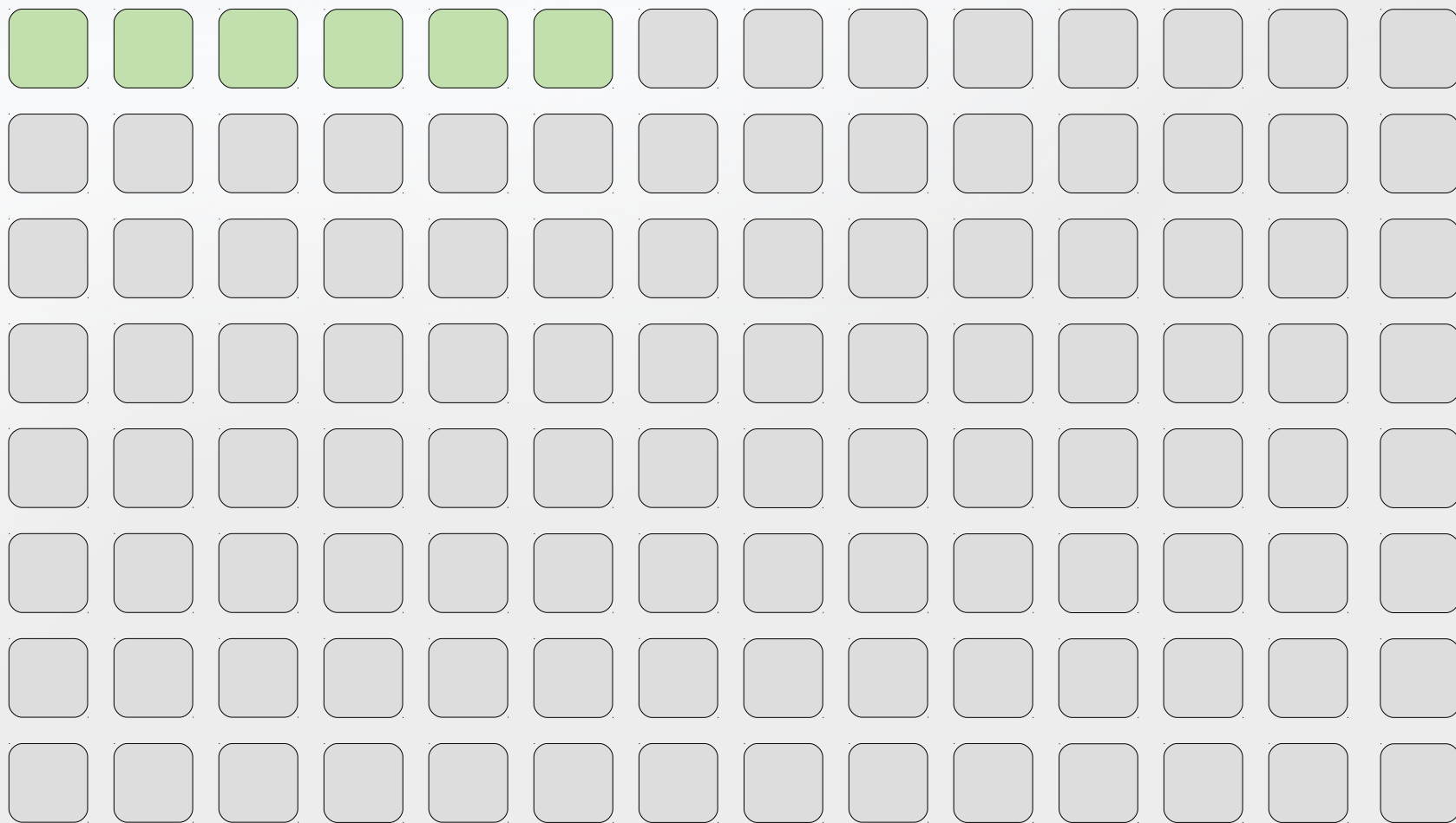
Eden

Survivor

Old

Humongous

Podstawy G1



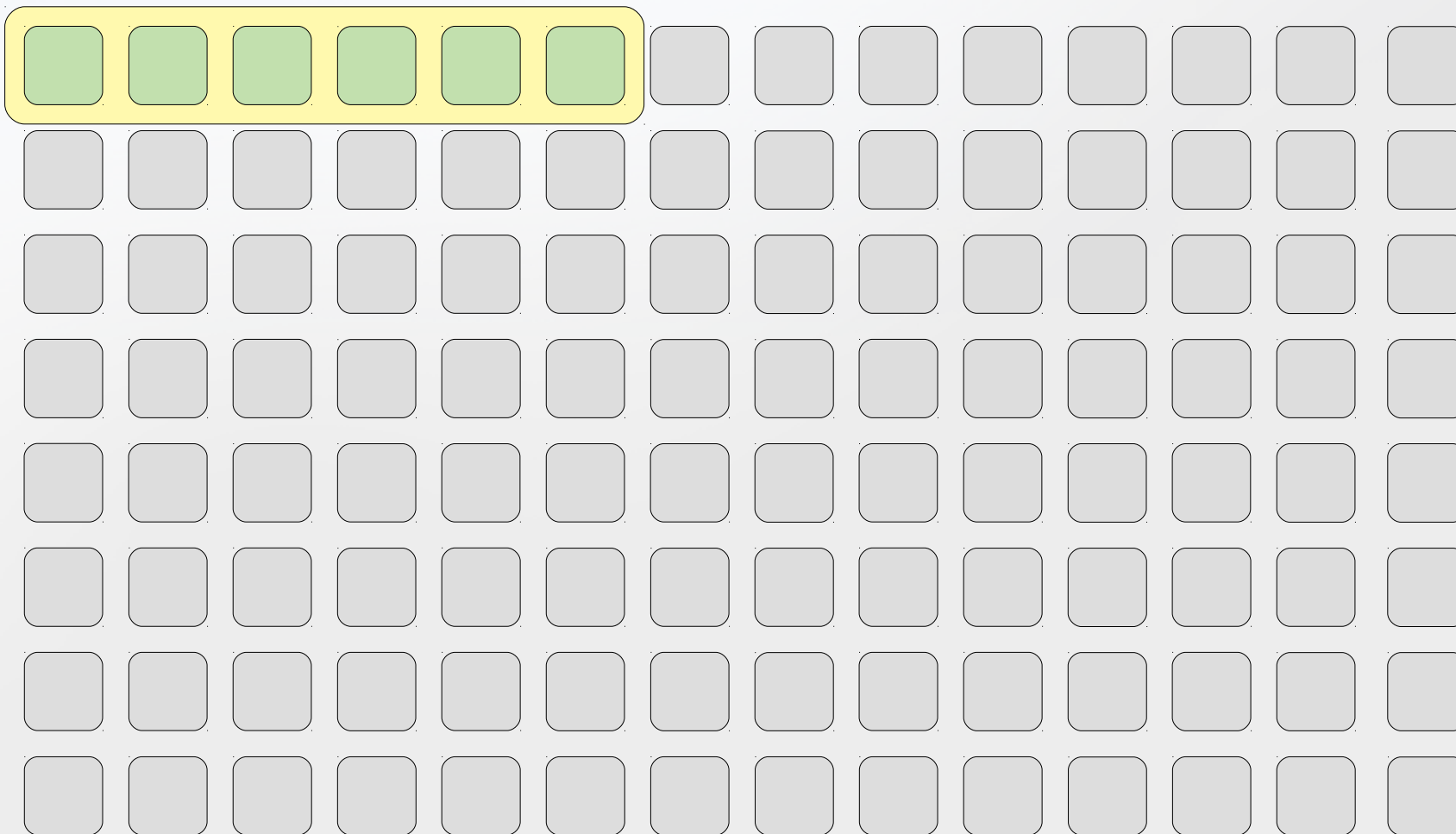
Eden

Survivor

Old

Humongous

Podstawy G1



Eden

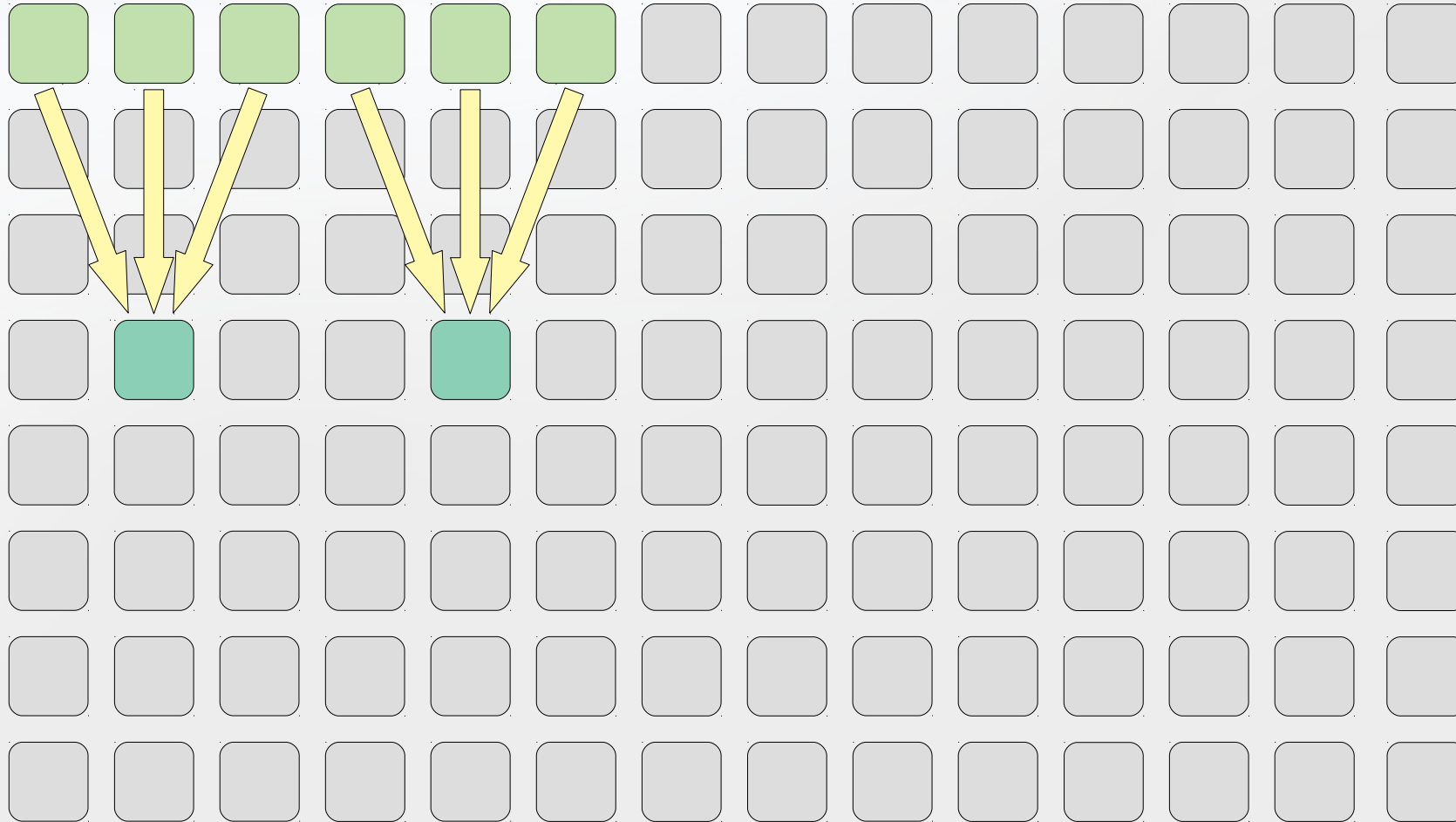
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

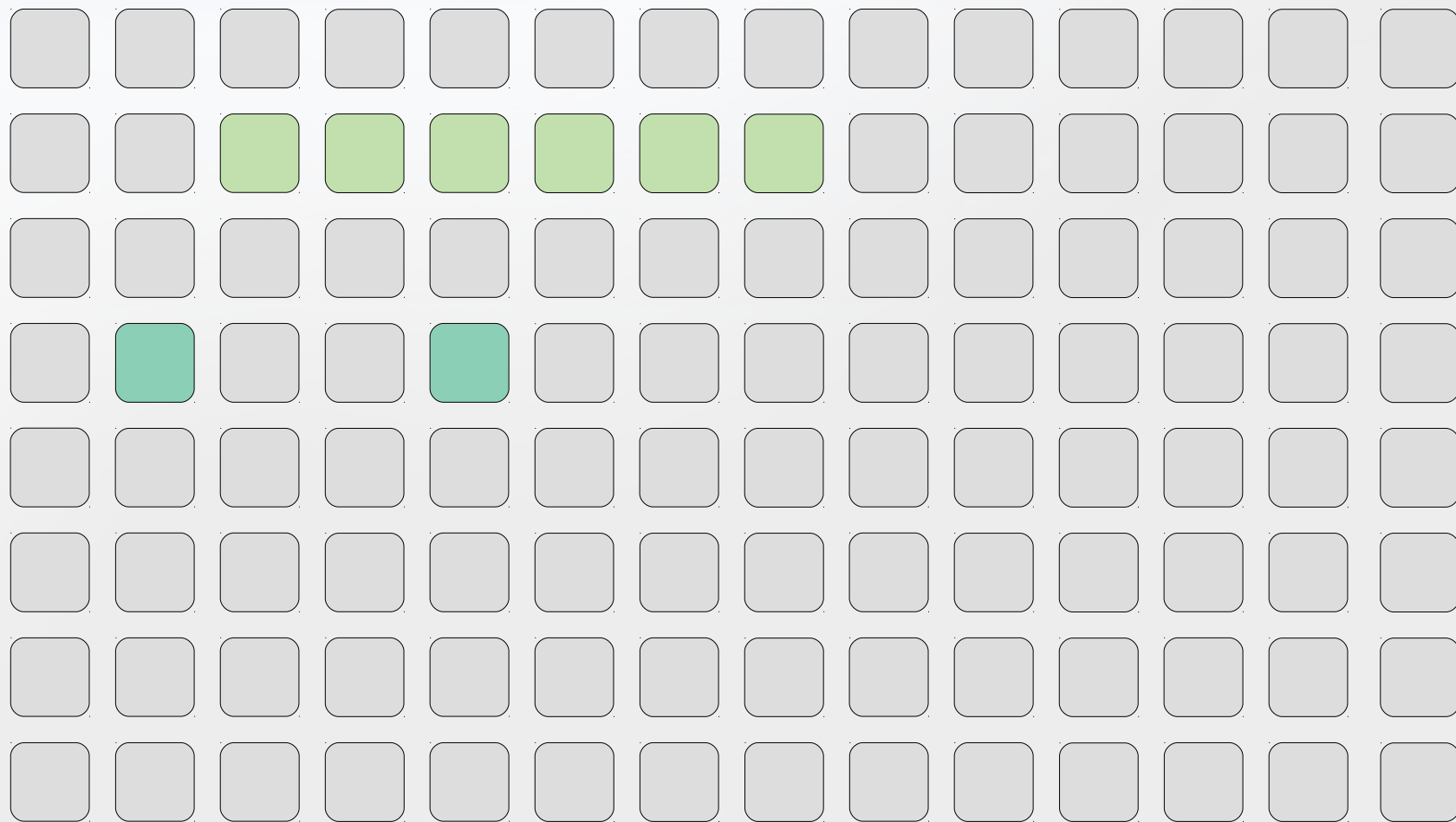
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

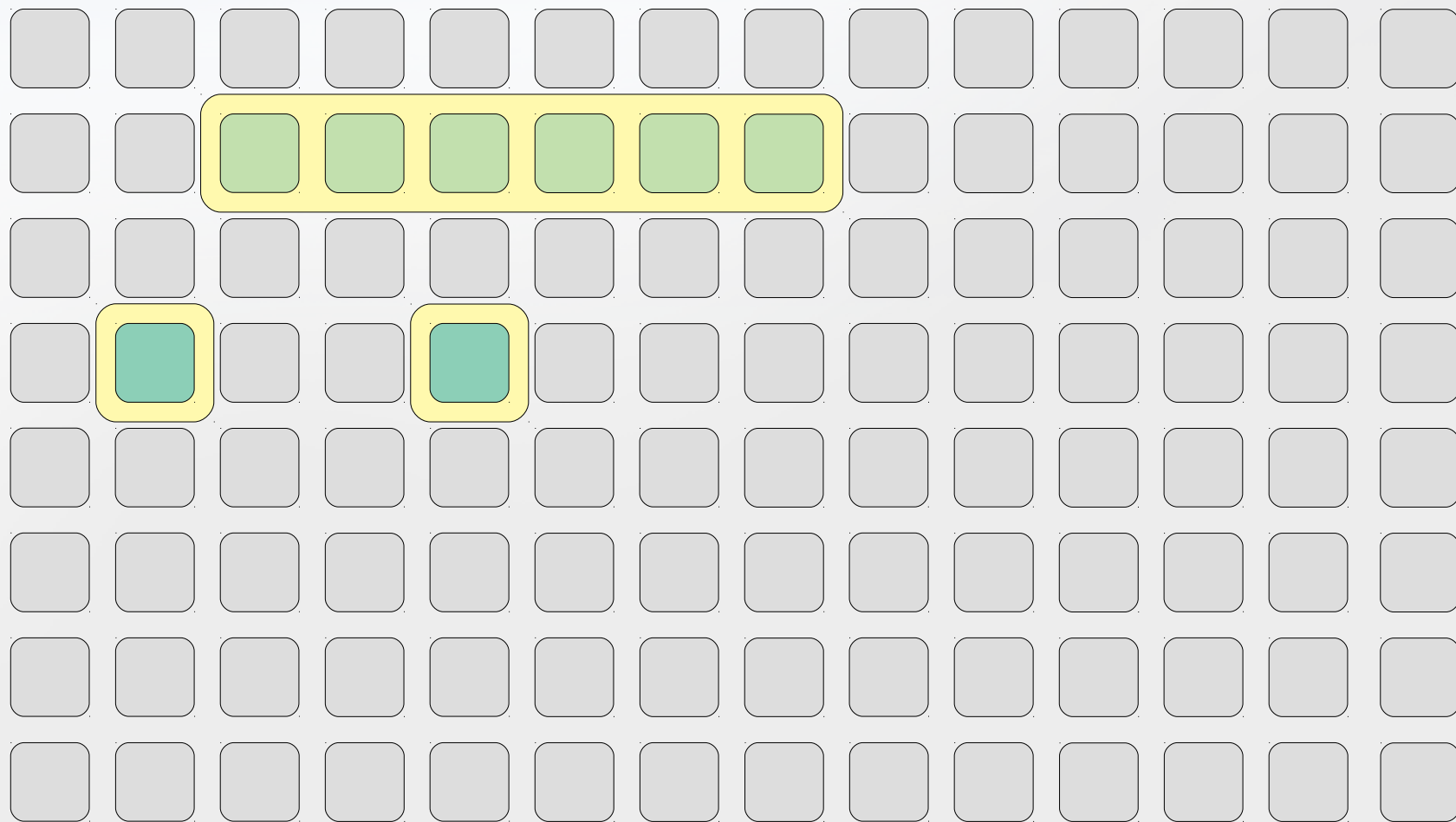
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

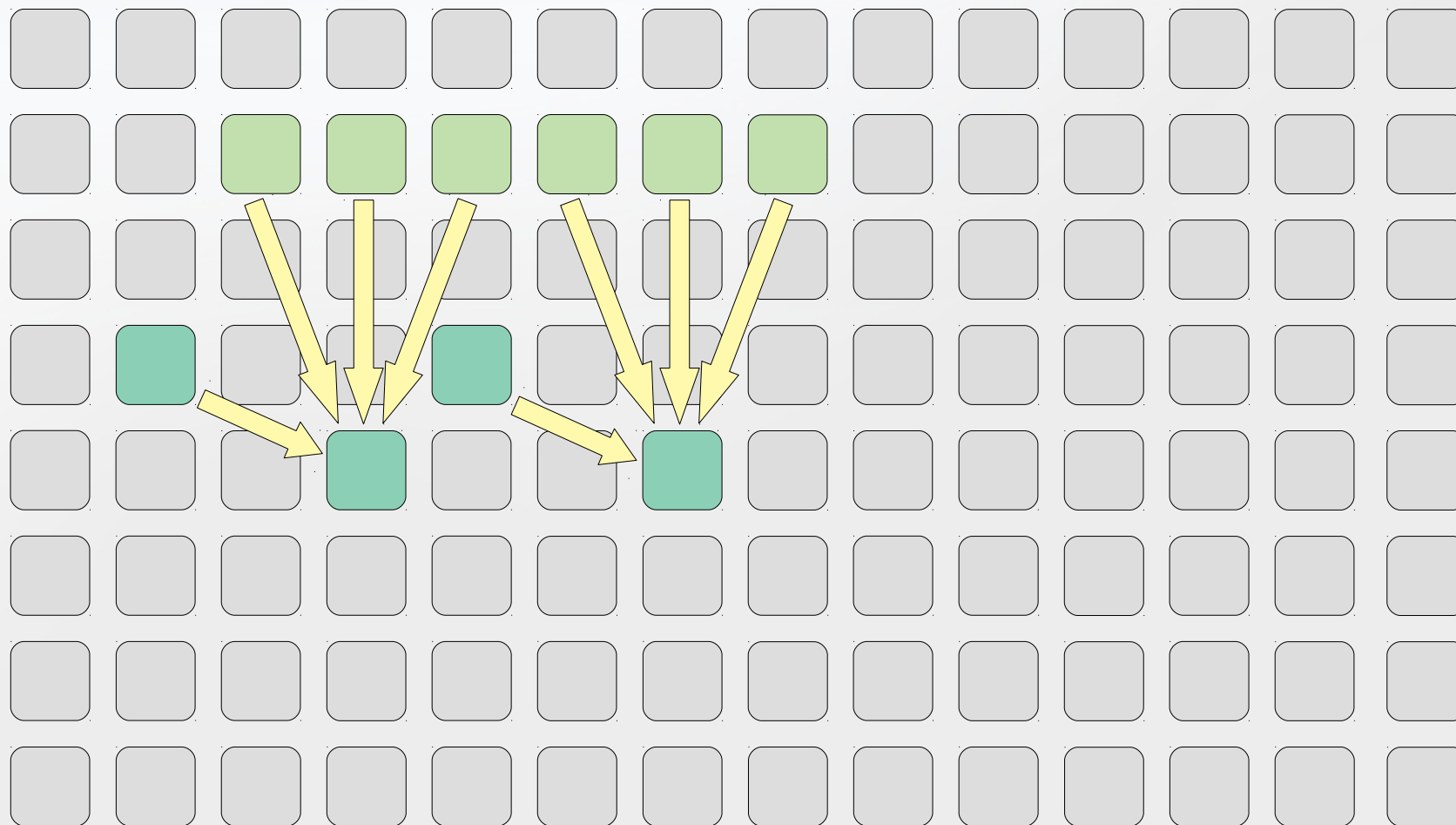
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

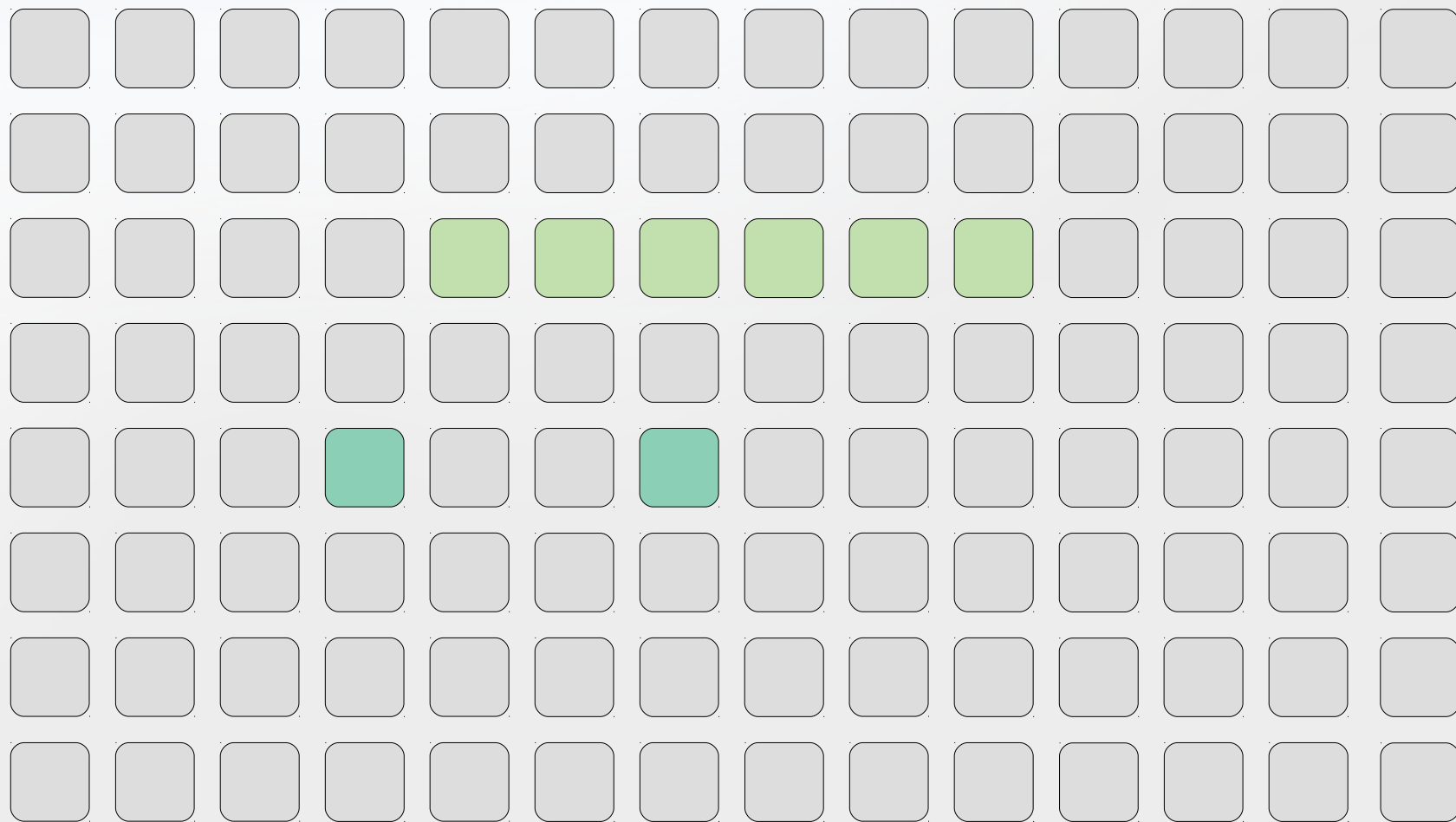
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

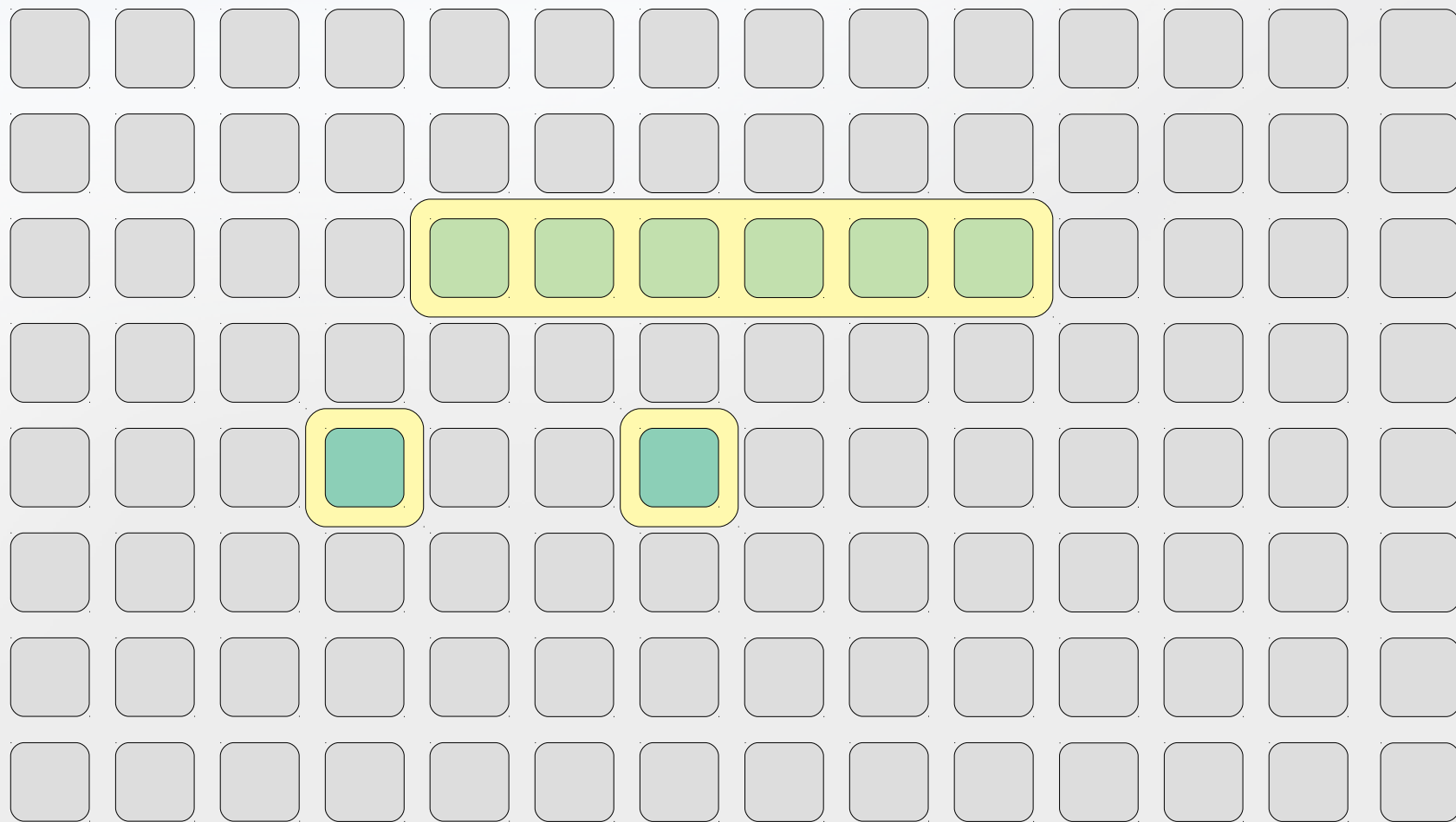
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

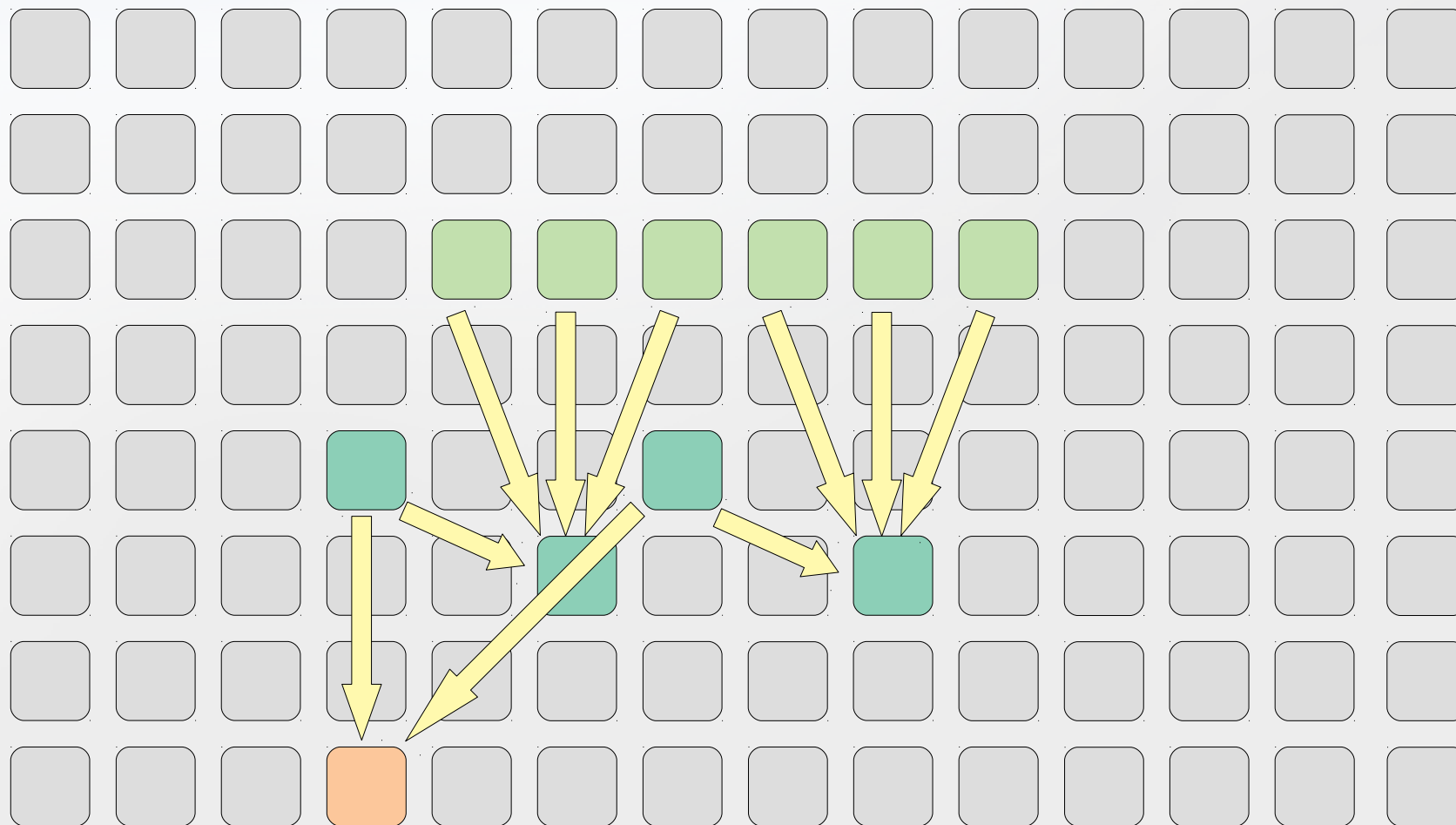
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

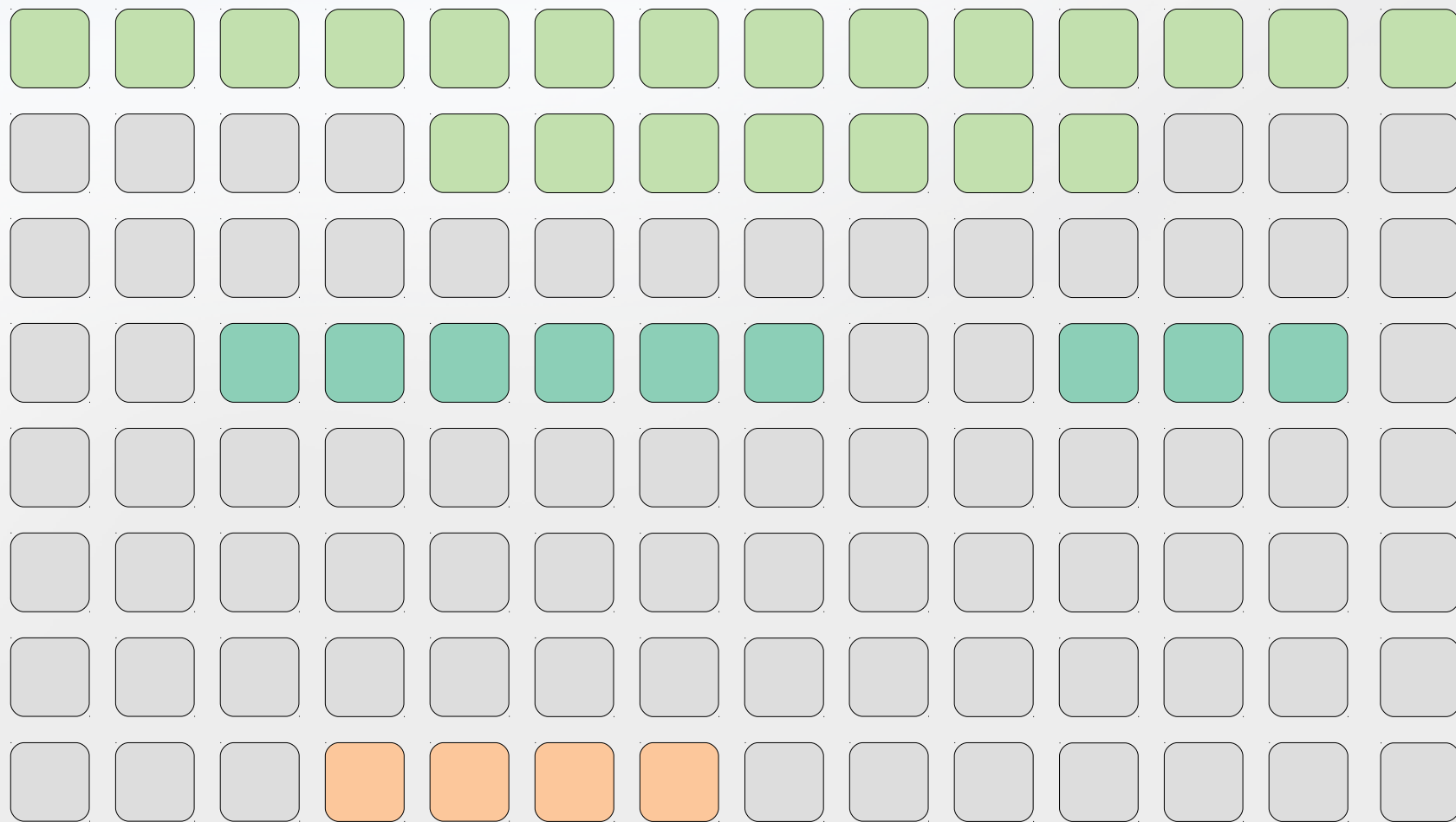
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

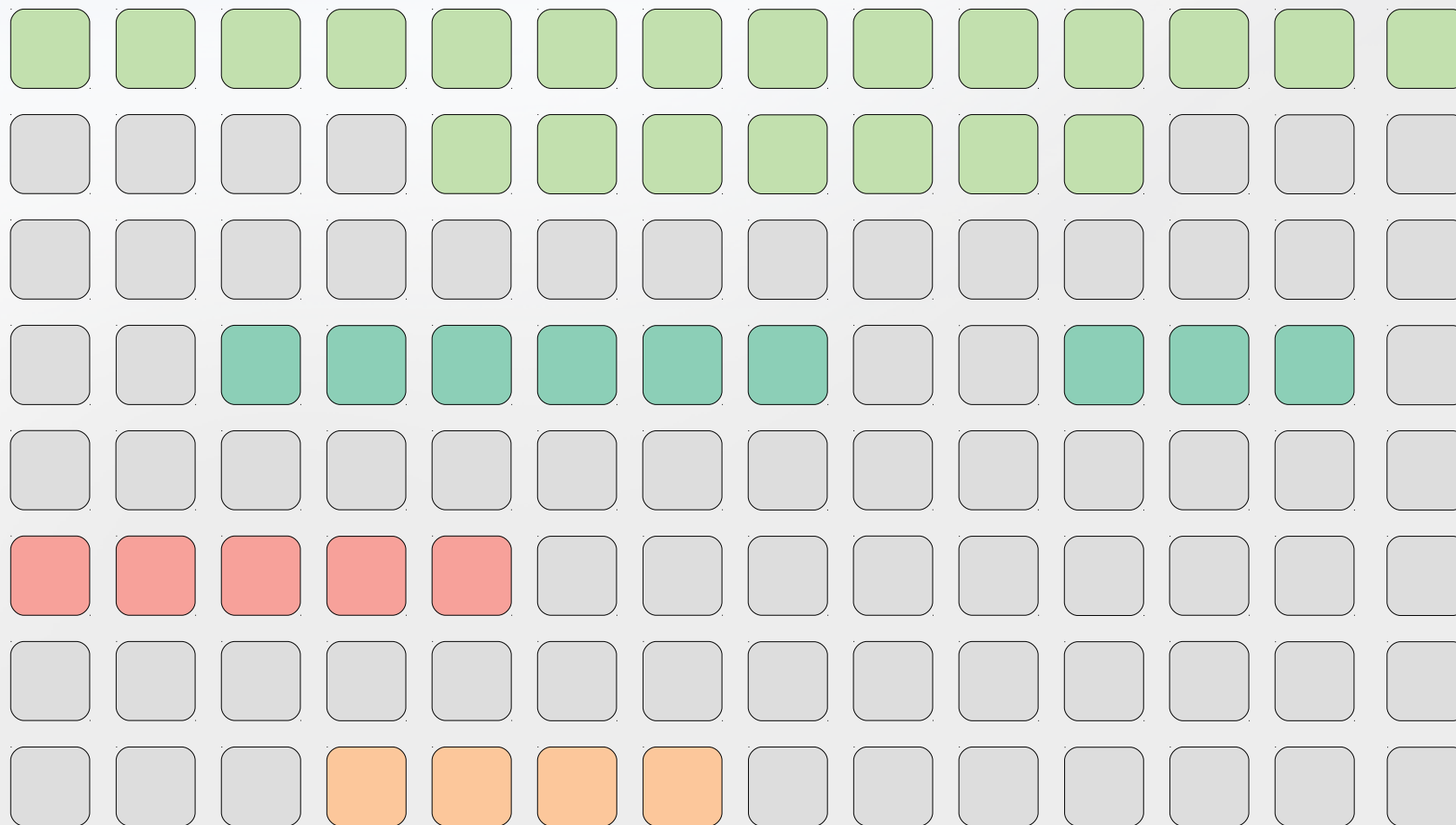
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

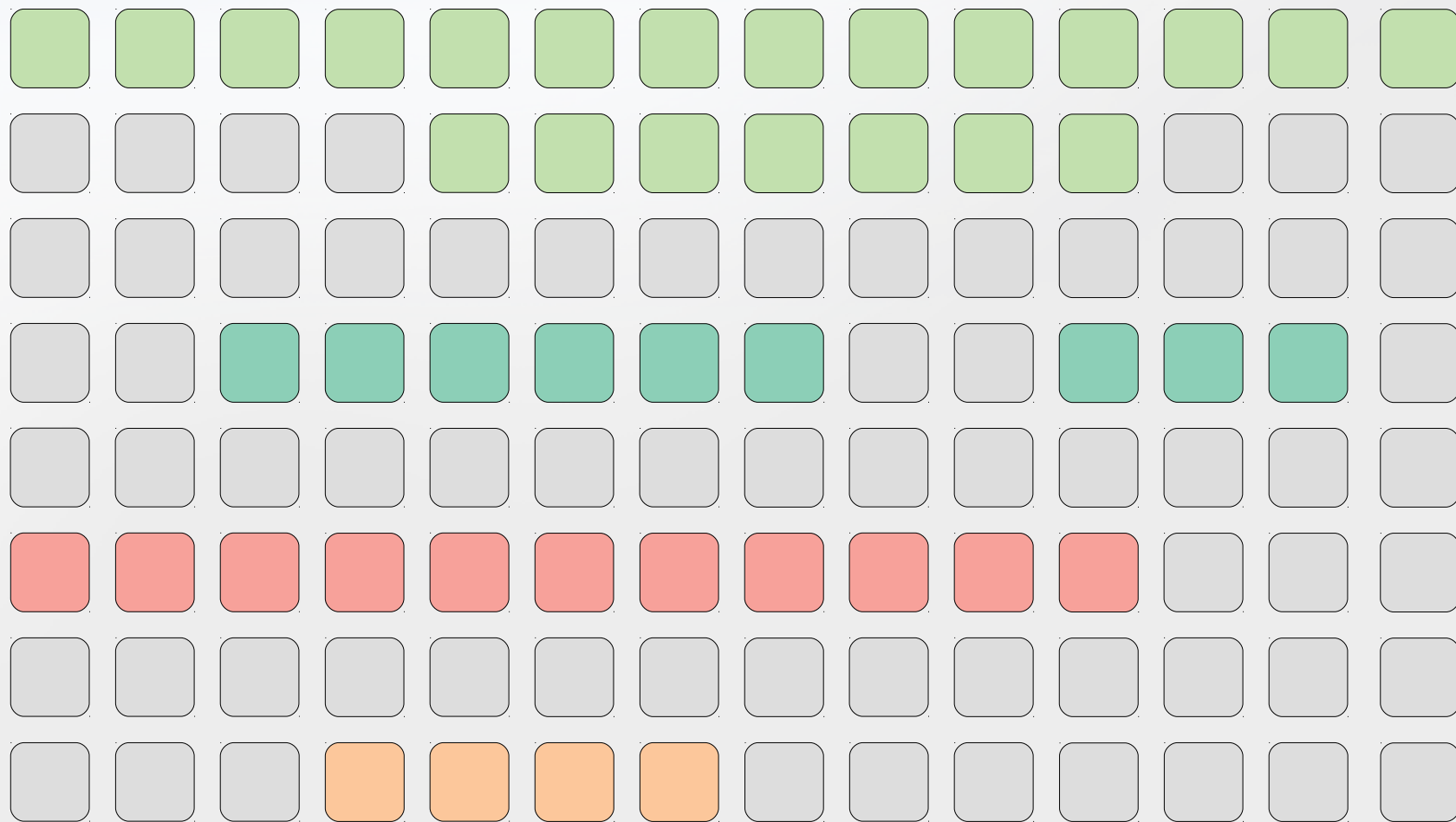
Survivor

Old

Humongous

Collection Set

Podstawy G1



Eden

Survivor

Old

Humongous

Collection Set

Humongous

- Co najmniej 1/2 regionu

Humongous

- Co najmniej $\frac{1}{2}$ regionu
- Ciągły obszar w pamięci

Humongous

- Co najmniej 1/2 regionu
- Ciągły obszar w pamięci
- Traktowane jako stara generacja

Humongous

- Co najmniej ½ regionu
- Ciągły obszar w pamięci
- Traktowane jako stara generacja
- Czyszczone także w nowej generacji (od JDK 8u60)

Humongous

- Co najmniej 1/2 regionu
- Ciągły obszar w pamięci
- Traktowane jako stara generacja
- Czyszczone także w nowej generacji (od JDK 8u60)
- Nigdy nie są relokowane (nawet podczas FullGC)

Stan początkowy

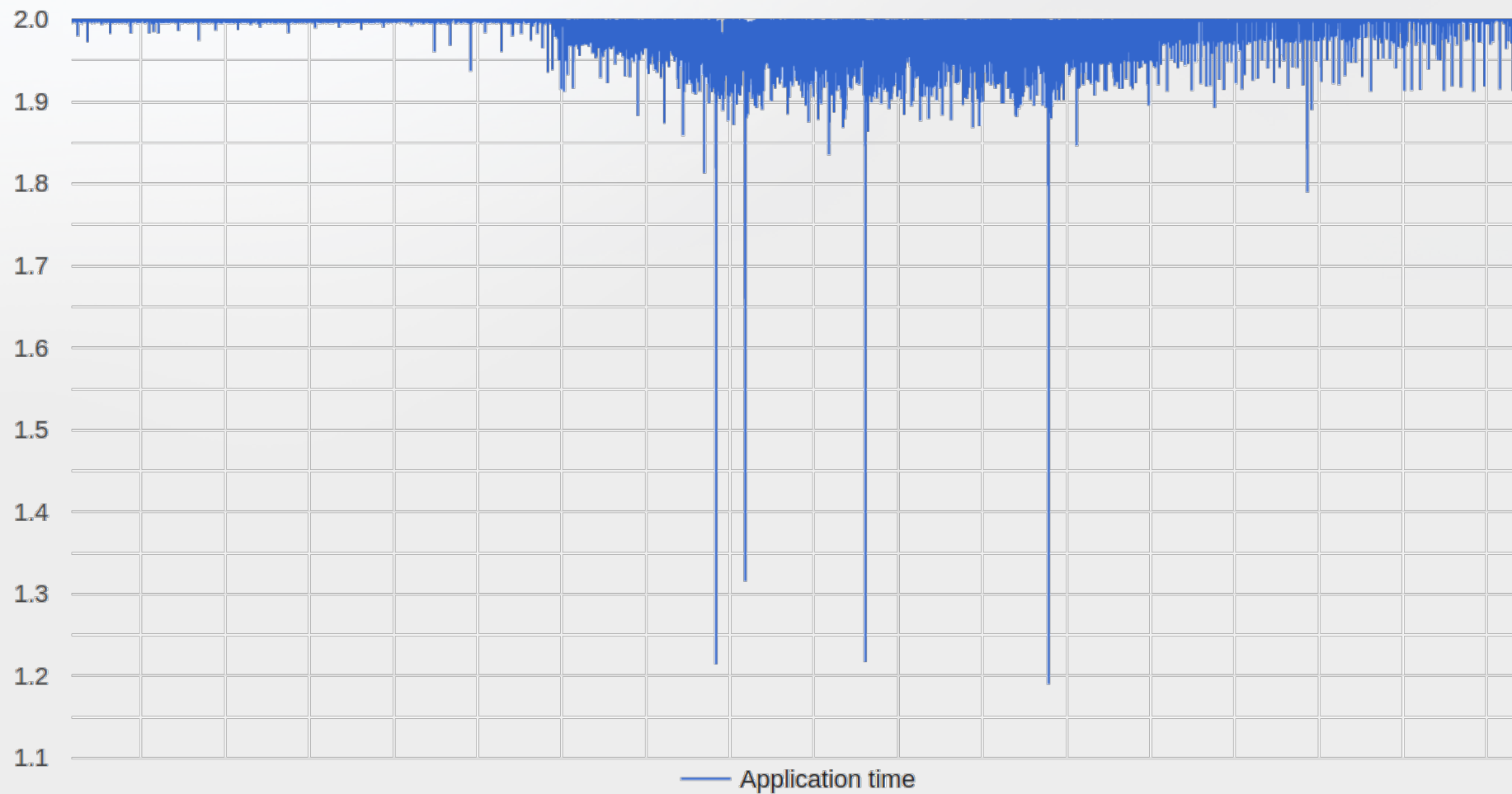
- -Xms5G
- -Xmx5G
- -XX:G1HeapRegionSize=4M



5G / 4M =
1280 regionów

Stan początkowy

2 second window



Stan początkowy

Phase stats (aggregated) - times in ms

Table presents statistics about each Stop The World Garbage Collector phase. Phases are aggregated to major type of collection.

Phase name	Count	Per. 50	Per. 75	Per. 90	Per. 95	Per. 99	Per. 99.9	Per. 100	Average	Total
Full collection	4	731.44	755.49	758.92	758.92	758.92	758.92	758.92	729.96	2 919.82
Mixed collection	884	28.00	32.50	37.65	43.62	67.68	113.99	113.99	29.92	26 445.31
Pause Cleanup	1911	0.65	0.74	0.81	0.88	1.38	4.94	6.44	0.65	1 250.91
Pause Remark	1911	29.52	33.90	43.07	55.97	66.42	115.26	138.26	32.44	61 986.39
Young collection	481	27.04	34.67	42.59	55.11	85.86	154.83	154.83	29.96	14 411.87
Young collection - piggybacks	2774	22.35	25.75	29.70	33.94	50.59	161.66	183.53	22.99	63 768.88

Stan początkowy

```
[gc      ] GC(696) Pause Young (Mixed) (G1 Evacuation Pause) 5117M->5117M(5120M) 9.341ms
[gc,cpu  ] GC(696) User=0.04s Sys=0.00s Real=0.01s
[gc,task ] GC(697) Using 6 workers of 6 for full compaction
[gc,start] GC(697) Pause Full (G1 Evacuation Pause)
```

```
[gc,metaspace] GC(697) Metaspace: 405617K->403519K(1433600K)
[gc      ] GC(697) Pause Full (G1 Evacuation Pause) 5117M->1418M(5120M) 717.679ms
[gc,cpu  ] GC(697) User=3.62s Sys=0.01s Real=0.72s
[safepoint] Leaving safepoint region
```

Stan początkowy

To-space exhausted

List of GC cycles where To-space exhausted occurred

Cycle	Survivor regions	Old regions	Eden regions	Humongous regions
336	35 --> 30	361 --> 760	720 --> 0	133 --> 1
403	34 --> 31	364 --> 644	719 --> 0	130 --> 1
460	34 --> 24	410 --> 846	672 --> 0	139 --> 1
598	22 --> 16	399 --> 856	706 --> 0	137 --> 1
689	22 --> 0	417 --> 917	656 --> 0	185 --> 1
690	0 --> 0	917 --> 1158	297 --> 0	66 --> 1
693	1 --> 0	1119 --> 1230	136 --> 0	24 --> 1
694	0 --> 0	1230 --> 1262	39 --> 0	11 --> 1
695	0 --> 0	1262 --> 1275	15 --> 0	3 --> 1
696	0 --> 0	1275 --> 1279	4 --> 0	1 --> 1

Stan początkowy

[gc,humongous]	GC(689)	Dead	humongous	region	113	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	116	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	117	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	127	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	129	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	131	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	132	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	133	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	159	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	164	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	208	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	214	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	219	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	249	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	293	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	294	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	298	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	306	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	323	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	324	object	size	2250152	start
[gc,humongous]	GC(689)	Dead	humongous	region	328	object	size	2250152	start

Stan początkowy

Humongous statistics

Table presents statistics about humongous regions (sizes in bytes)

Type	Count	Per. 50	Per. 75	Per. 90	Per. 95	Per. 99	Per. 99.9	Per. 100	Average
Live	4 284.00	2 250 120.00	2 250 120.00	2 250 120.00	2 250 120.00	2 250 152.00	2 250 152.00	2 250 152.00	2 250 121.08
Dead	141 999.00	2 250 152.00	2 250 152.00	2 250 152.00	2 250 152.00	2 250 152.00	2 745 472.00	5 784 032.00	2 251 501.86
All (Live + Dead)	146 283.00	2 250 152.00	2 250 152.00	2 250 152.00	2 250 152.00	2 250 152.00	2 686 634.72	5 784 032.00	2 251 461.42

Stan początkowy

Humongous statistics

Table presents statistics about humongous regions (sizes in bytes)

Type	Count	Size	Size / 100	Average
Live	4 284.00	2 250 121.08	2 250 121.08	2 250 121.08
Dead	146 283.00	2 251 501.86	2 251 501.86	2 251 501.86
All (Live + Dead)	146 283.00	2 251 461.42	2 251 461.42	2 251 461.42

-XX:
G1HeapRegionSize=8M ?
???

Szukanie kłopotu

- HeapDump tuż przed kolekcją GC

Szukanie kłopotu

- HeapDump tuż przed kolekcją GC
- Zawierający też martwe obiekty

Szukanie kłopotu

- HeapDump tuż przed kolekcją GC
- Zawierający też martwe obiekty

```
jcmd 17371 GC.heap_dump -all /tmp/dump.hprof
```

Szukanie kłopotu

YourKit Java Profiler 2019.8-b137

File View Memory CPU Settings Tools Help

Memory * Threads * Inspections * Summary *

Welcome

Memory profiling
Allocations were not recorded

Memory & GC telemetry

Object explorer

Biggest objects - Dominators

Inspections

Objects by category

Class

Class and package

Class loader

Web application

Generation

Reachability

Shallow size

All objects (reachable and unreachable) [Reachability scopes](#)
Objects: 66,331,339 / shallow size: 3.4 GB / retained size: 3.4 GB [Strong reachable](#) among them: 26,

Shallow Size Range (by)	Objects	Shallow Size	Retained Size
< 32 bytes	38,982,002 59 %	894,732,960 25 %	1,691,961,264 46 %
32 bytes — 64 bytes (excl.)	21,140,394 32 %	813,322,352 22 %	1,767,494,460 49 %
64 bytes — 128 bytes (excl.)	4,890,296 7 %	414,023,458 11 %	1,501,960,803 41 %
128 bytes — 1 KB (excl.)	1,188,567 2 %	374,952,237 10 %	1,432,763,795 39 %
1 KB — 4 KB (excl.)	103,055 0 %	161,079,680 4 %	484,704,010 13 %
4 KB — 64 KB (excl.)	24,282 0 %	266,413,112 7 %	1,059,972,130 29 %
64 KB — 1 MB (excl.)	2,611 0 %	449,043,424 12 %	497,936,238 14 %
≥ 1 MB	132 0 %	267,559,992 7 %	267,559,992 7 %

Classes Packages Object Explorer Allocations Reachability Ages [Calculate exact retained sizes](#)

Class	Objects	Shallow Size	Retained Size
java.util.ArrayList	1,497,194 4 %	35,932,656 4 %	≈ 1,022,482,384 60 %
java.lang.String	10,105,253 26 %	242,526,072 27 %	≈ 661,667,360 39 %
org.springframework.w	7,399 0 %	177,576 0 %	≈ 466,579,816 28 %
java.util.LinkedList\$Noc	89,071 0 %	2,137,704 0 %	≈ 320,054,224 19 %
java.util.concurrent.Co	2,402 0 %	57,648 0 %	≈ 304,351,408 18 %
org.springframework.w	1,384 0 %	22,144 0 %	≈ 302,673,720 18 %
java.lang.Object[]	3,243,958 8 %	66,252,880 7 %	≈ 263,056,344 16 %

Szukanie kłopotu

YourKit Java Profiler 2019.8-b137

File View Memory CPU Settings Tools Help

Memory * Threads * Inspections * Summary *

Welcome

Memory profiling
Allocations were not recorded

Memory & GC telemetry
Object explorer
Biggest objects - Dominators
Inspections

Objects by category

- Class
- Class and package
- Class loader
- Web application
- Generation
- Reachability
- Shallow size**

All objects (reachable and unreachable) [Reachability scopes](#)
Objects: 66,331,339 / shallow size: 3.4 GB / retained size: 3.4 GB [Strong reachable](#) among them: 26,

Shallow Size Range (by)	Objects	Shallow Size	Retained Size
< 32 bytes	38,982,002 59 %	894,732,960 25 %	1,691,961,264 46 %
32 bytes — 64 bytes (excl.)	21,140,394 32 %	813,322,352 22 %	1,767,494,460 49 %
64 bytes — 128 bytes (excl.)	4,890,296 7 %	414,023,458 11 %	1,501,960,803 41 %
128 bytes — 1 KB (excl.)	1,188,567 2 %	374,952,237 10 %	1,432,763,795 39 %
1 KB — 4 KB (excl.)	103,055 0 %	161,079,680 4 %	484,704,010 13 %
4 KB — 64 KB (excl.)	24,282 0 %	266,413,112 7 %	1,059,972,130 29 %
64 KB — 1 MB (excl.)	2,611 0 %	449,043,424 12 %	497,936,238 14 %
≥ 1 MB	132 0 %	267,559,992 7 %	267,559,992 7 %

Classes Packages Object Explorer Allocations Reachability Ages [Calculate exact retained sizes](#)

Class	Objects	Shallow Size	Retained Size
byte[]	61 46 %	134,595,296 50 %	≈ 134,595,296 50 %
int[]	71 54 %	132,964,696 50 %	≈ 132,964,696 50 %

Szukanie kłopotu

The screenshot shows the YourKit Java Profiler interface. The main window displays the 'Instances of class 'byte[]'' section, which shows 61 objects with a total shallow size of 128 MB and a retained size of 128 MB. A green box highlights a table of these instances, showing their names, retained sizes, and shallow sizes. The table has columns for 'Name', 'Retained Size', and 'Shallow Size'. The 'Shallow Size' column is highlighted with a green box. Below the table, there are tabs for 'Paths from GC Roots', 'Allocations', 'Ages', 'Class Hierarchy', 'Incoming References', and 'Quick Info'. A 'Calculate paths' button is visible at the bottom right of the main window.

File View Memory CPU Settings Tools Help

Memory * Instances of 'byte[]' * Threads * Inspections * Summary *

Welcome

Memory profiling
Allocations were not recorded

Memory & GC telemetry

Object explorer

Biggest objects - Dominators

Inspections

Objects by category

Class

Class and package

Class loader

Web application

Generation

Reachability

Shallow size

Instances of class 'byte[]'
Objects: 61 / shallow size: 128 MB / retained size: 128 MB Strong reachable among them: 3 (4%) / sh

Class name, string value, thread name or ID (Press "Enter" to apply / ?):

Name	Retained Size	Shallow Size
[Unreachable] byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152

Paths from GC Roots Allocations Ages Class Hierarchy Incoming References Quick Info

Calculate paths

Szukanie kłopotu

YourKit Java Profiler 2019.8-b137

File View Memory CPU Settings Tools Help

Memory * Instances of 'byte[]' * Threads * Inspections * Summary *

Welcome

Memory profiling
Allocations were not recorded

Memory & GC telemetry

Object explorer

Biggest objects - Dominators

Inspections

Objects by category

- Class
- Class and package
- Class loader
- Web application
- Generation
- Reachability
- Shallow size

Instances of class 'byte[]'
Objects: 61 / shallow size: 128 MB / retained size: 128 MB **Strong reachable** among them: 3 (4%) / sh

Class name, string value, thread name or ID (Press "Enter" to apply / ?):

Name	Retained Size	Shallow Size
[Unreachable] A byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] A byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] A byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] A byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] A byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152
[Unreachable] A byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -	2,250,152	2,250,152

Paths from GC Roots | Allocations | Ages | Class Hierarchy | **Incoming References** | Quick Info

Name	Retained Size	Shallow Size
[Unreachable] A byte[2250133] = {0, 0, 0, 0, -1, -1, -1, -2,	2,250,152	2,250,152
payload of [Unreachable] com.hazelcast.nio.Packe	32	32
state of [Unreachable] com.hazelcast.spi.impl.c	32	32
item of [Unreachable] com.hazelcast.internal.u	24	24

Szukanie kłopotu

The screenshot shows the YourKit Java Profiler interface. The main window displays the 'Instances of class 'byte[]'' section. The top bar indicates the application is 'YourKit Java Profiler 2019.8-b137'. The menu bar includes 'File', 'View', 'Memory', 'CPU', 'Settings', 'Tools', and 'Help'. The 'Memory' tab is active, showing 'Instances of 'byte[]''. The summary shows 61 objects with a shallow size of 128 MB and a retained size of 128 MB. A 'Quick Info' button is highlighted with a green box. The 'Quick Info' section provides details about the object class, generation, index, and size. The 'Array text presentation' section shows the first 1000 elements of the array, with the encoding set to UTF-8. A green box highlights the first few elements of the array, which are 'Config=B7 j L configst'.

Instances of class 'byte[]'
Objects: 61 / shallow size: 128 MB / retained size: 128 MB [Strong reachable](#) among them: 3 (4%) / shallow size: 3.9 MB (3%) / retained size: 3.9

Paths from GC Roots | Allocations | Ages | Class Hierarchy | Incoming References | **Quick Info**

Object class: byte[]
Object generation: not available
Object index: #1602614
Web application: None
Distance to nearest GC root: Not reachable from GC roots

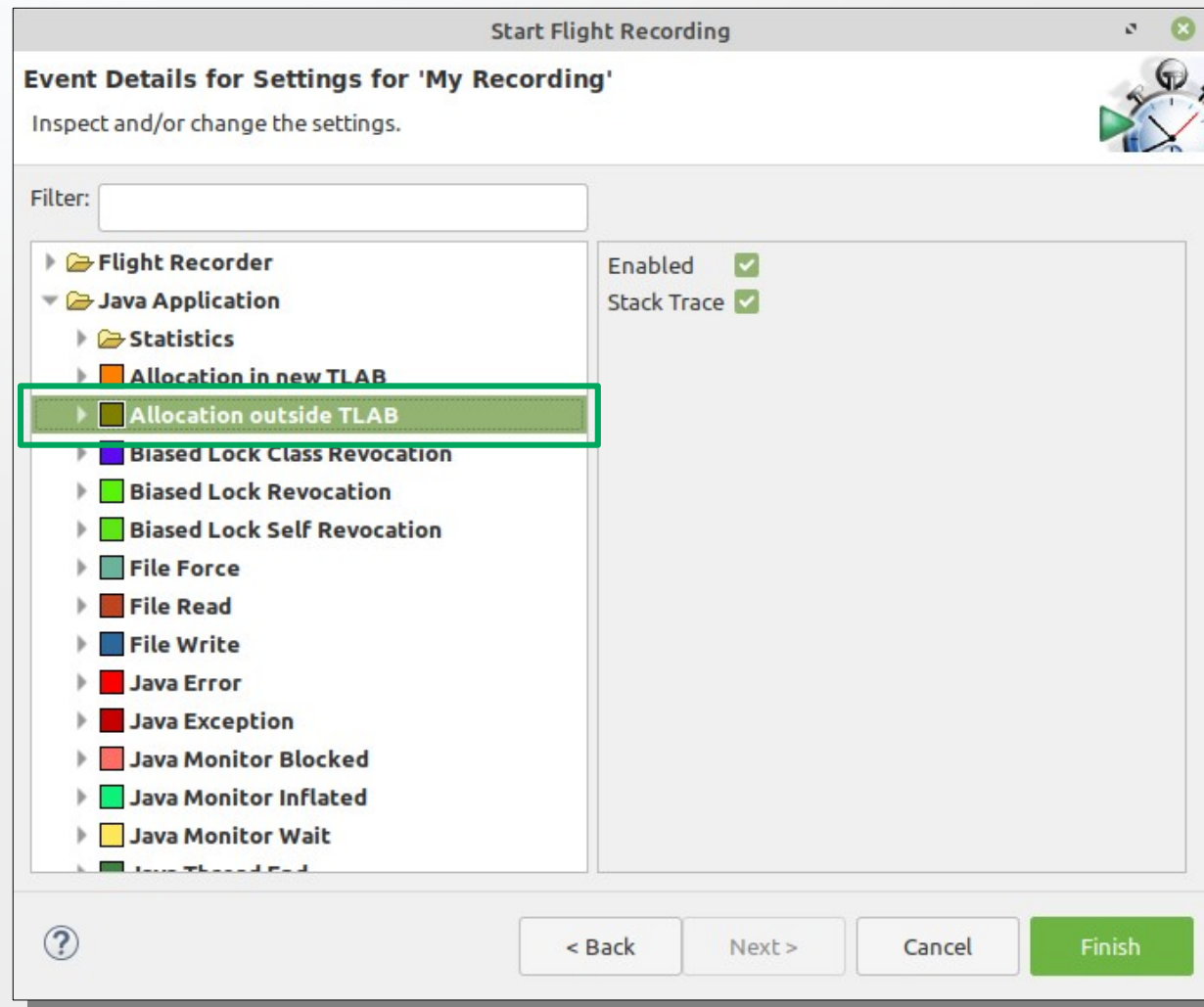
Shallow size: 2,250,152
Retained objects (includes the object itself): 1
Retained size (includes shallow size): 2,250,152

[Export bytes to binary file...](#) [Export array elements to text file \(one element per line\)...](#)

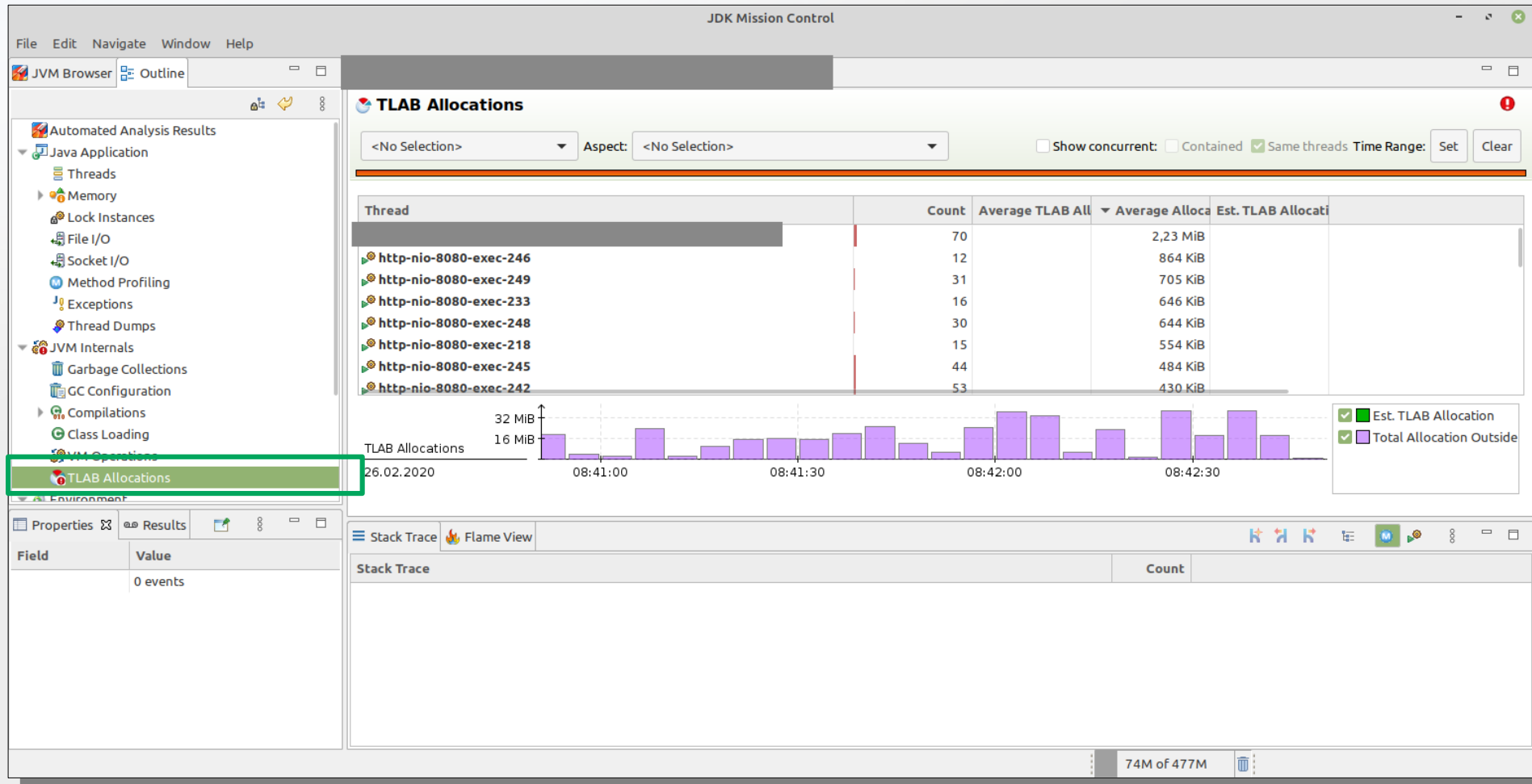
Array text presentation
Show elements starting from index: 0 Show up to 1000 filled elements [Update](#)
Encoding: UTF-8

Config=B7 j L configst

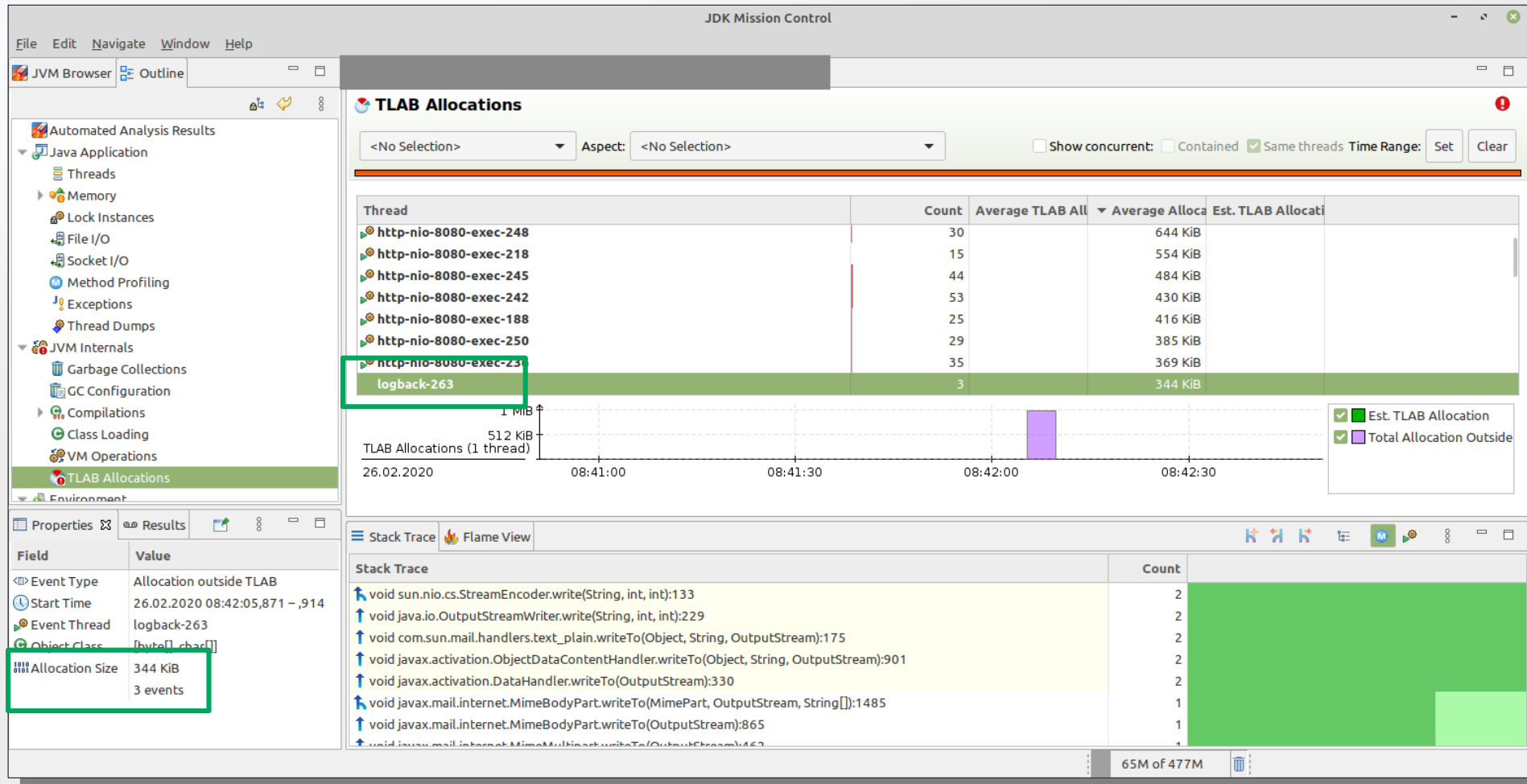
Szukanie kłopotu - JMC



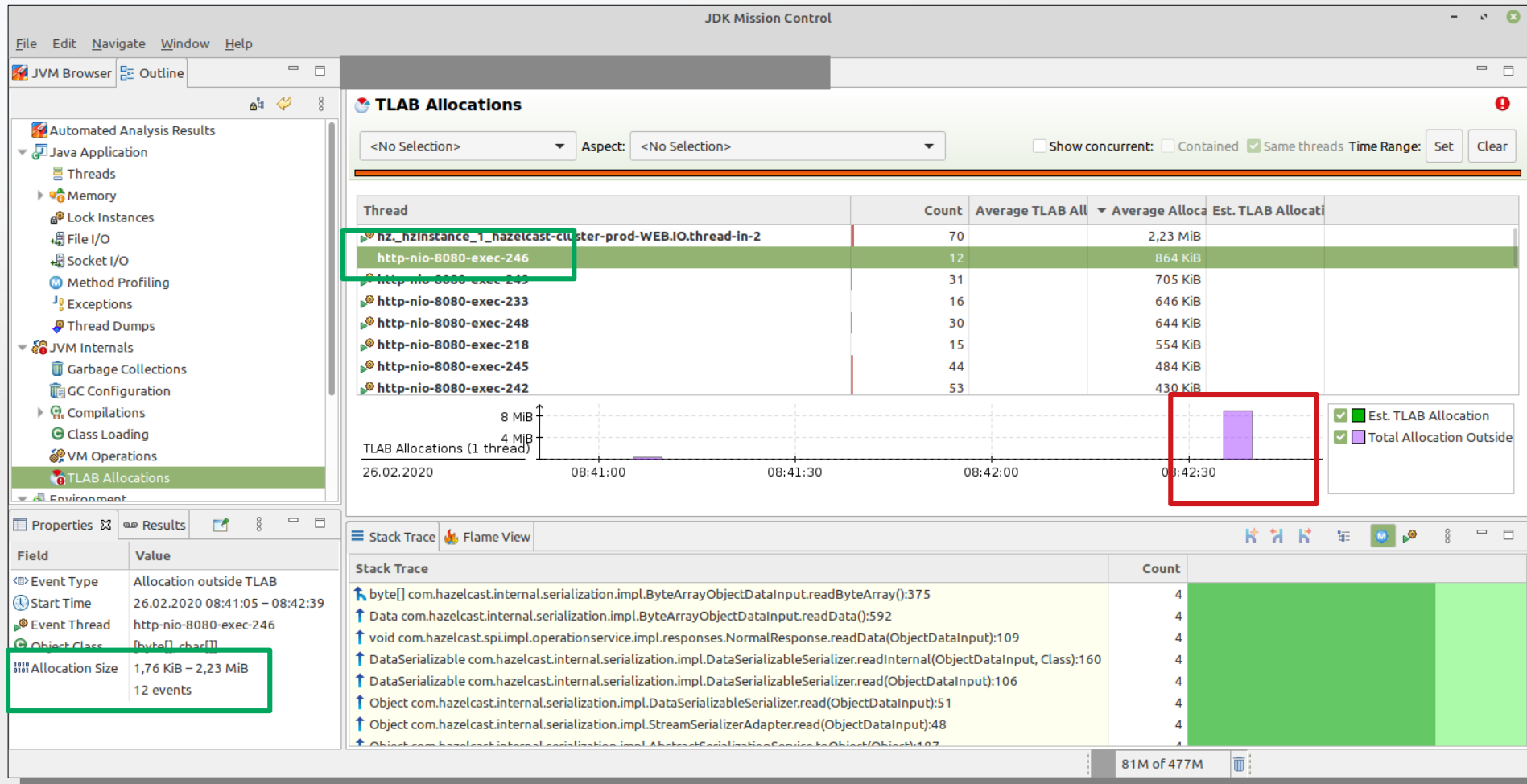
Szukanie kłopotu - JMC



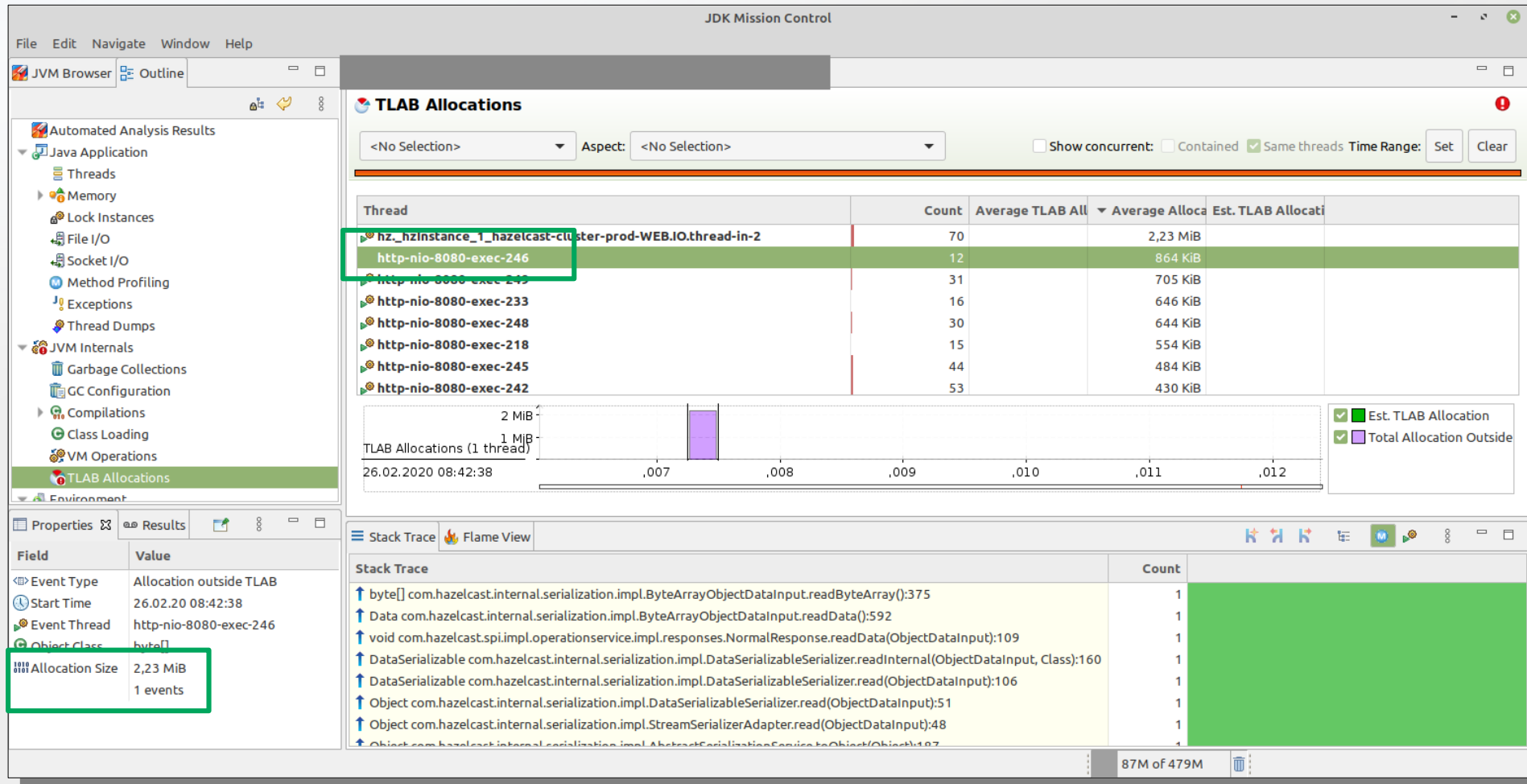
Szukanie kłopotu - JMC



Szukanie kłopotu - JMC

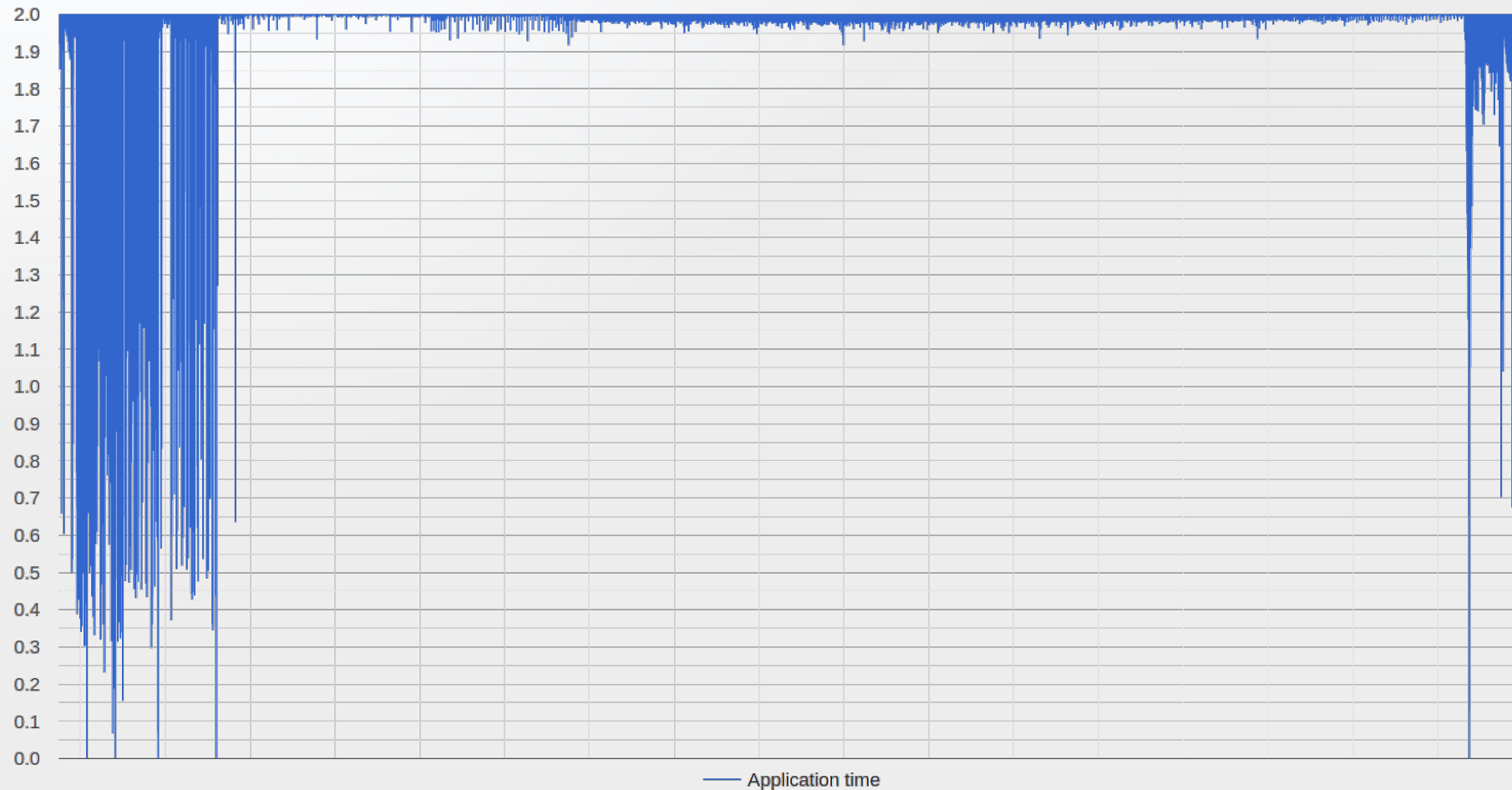


Szukanie kłopotu - JMC



Stan początkowy - case 2

2 second window



Stan początkowy - case 2

Phase stats (aggregated) - times in ms

Table presents statistics about each Stop The World Garbage Collector phase. Phases are aggregated to major type of collection.

Phase name	Count	Per. 50	Per. 75	Per. 90	Per. 95	Per. 99	Per. 99.9	Per. 100	Average	Total
Full collection	224	1 615.25	1 688.28	1 760.21	1 807.26	1 886.79	2 133.14	2 133.14	1 605.94	359 730.79
Mixed collection	127	23.86	28.86	54.11	153.03	308.35	326.49	326.49	37.63	4 778.68
Pause Cleanup	214	0.33	0.38	0.47	0.50	0.54	0.69	0.69	0.34	73.68
Pause Remark	229	25.96	43.59	46.27	48.69	61.06	101.80	101.80	31.94	7 313.20
Young collection	1873	18.68	27.72	92.68	127.79	432.53	1 552.39	1 566.09	40.80	76 414.07
Young collection - piggybacks	518	34.61	50.64	117.56	157.59	265.34	305.60	305.60	48.57	25 157.98

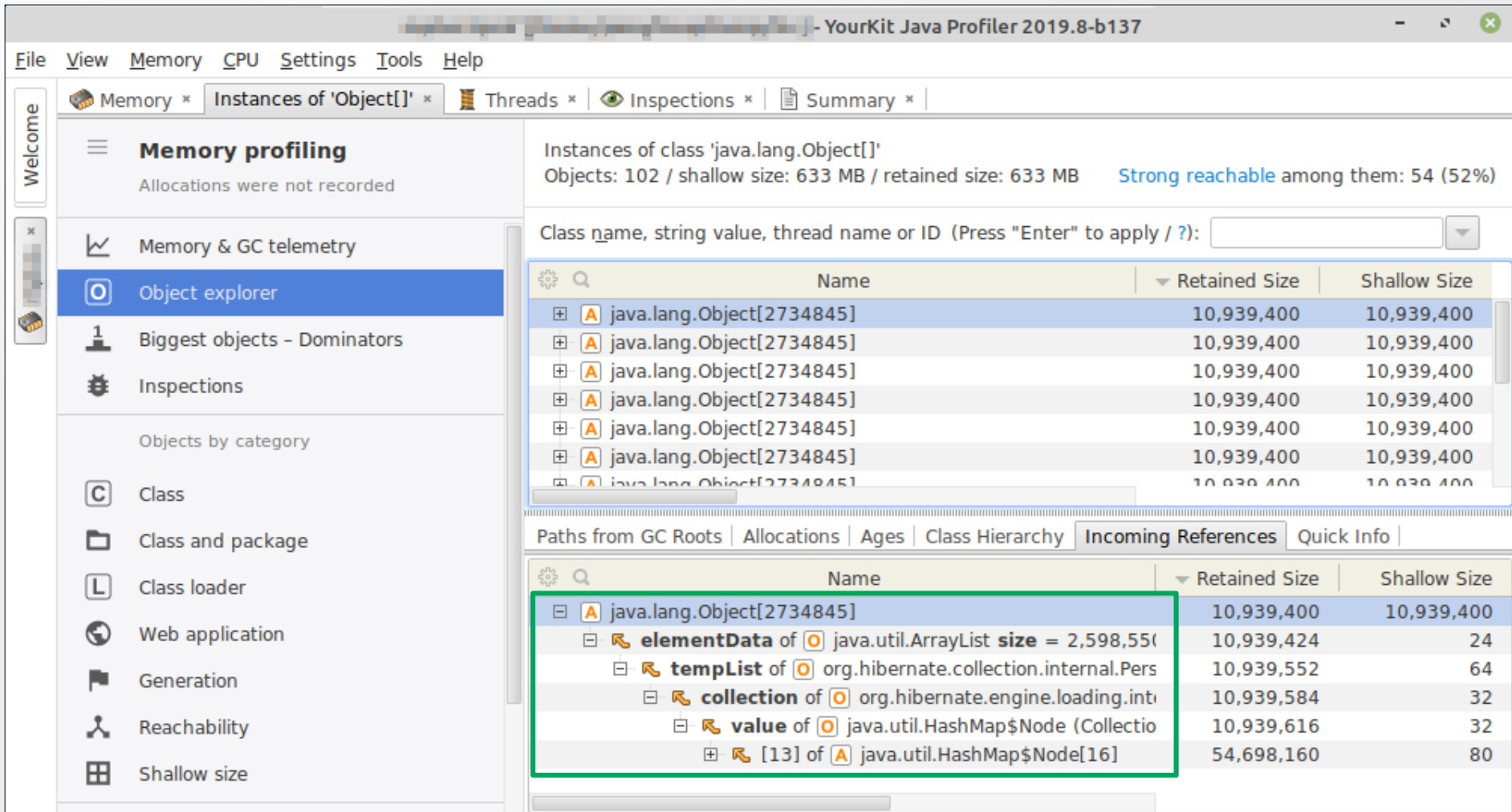
Stan początkowy - case 2

Humongous statistics

Table presents statistics about humongous regions (sizes in bytes)

Type	Count	Per. 50	Per. 75	Per. 90	Per. 95	Per. 99	Per. 99.9	Per. 100	Average
Live	44 105.00	83 070 920.00	124 606 368.00	186 909 536.00	280 364 296.00	280 364 296.00	420 546 440.00	420 546 440.00	91 323 998.28
Dead	45.00	5 116 600.00	7 025 216.00	8 047 940.80	9 706 473.60	14 190 568.00	14 190 568.00	14 190 568.00	5 815 104.36
All (Live + Dead)	44 150.00	83 070 920.00	124 606 368.00	186 909 536.00	280 364 296.00	280 364 296.00	420 546 440.00	420 546 440.00	91 236 843.12

Szukanie kłopotu - case 2



The screenshot shows the YourKit Java Profiler interface. The left sidebar contains a 'Welcome' panel and a list of tools: Memory & GC telemetry, Object explorer (selected), Biggest objects - Dominators, and Inspections. Below these are 'Objects by category' with options like Class, Class and package, Class loader, Web application, Generation, Reachability, and Shallow size.

The main window displays 'Memory profiling' with the status 'Allocations were not recorded'. The top tabs include Memory, Instances of 'Object[]', Threads, Inspections, and Summary. The 'Instances of class 'java.lang.Object[]' section shows a summary: 'Objects: 102 / shallow size: 633 MB / retained size: 633 MB' and 'Strong reachable among them: 54 (52%)'. A search bar is present above a table of instances.

Name	Retained Size	Shallow Size
java.lang.Object[2734845]	10,939,400	10,939,400
java.lang.Object[2734845]	10,939,400	10,939,400
java.lang.Object[2734845]	10,939,400	10,939,400
java.lang.Object[2734845]	10,939,400	10,939,400
java.lang.Object[2734845]	10,939,400	10,939,400
java.lang.Object[2734845]	10,939,400	10,939,400
java.lang.Object[2734845]	10,939,400	10,939,400

Below the table, the 'Incoming References' tab is selected, showing a detailed view of the selected object's references. A green box highlights the following references:

Name	Retained Size	Shallow Size
java.lang.Object[2734845]	10,939,400	10,939,400
elementData of java.util.ArrayList size = 2,598,551	10,939,424	24
tempList of org.hibernate.collection.internal.Pers	10,939,552	64
collection of org.hibernate.engine.loading.inti	10,939,584	32
value of java.util.HashMap\$Node (Collectio	10,939,616	32
[13] of java.util.HashMap\$Node[16]	54,698,160	80

Szukanie kłopotu - case 2

JDK Mission Control

File Edit Navigate Window Help

JVM Browser Outline

Automated Analysis Results

- Java Application
 - Threads
 - Memory
 - Lock Instances
 - File I/O
 - Socket I/O
 - Method Profiling
 - Exceptions

Properties Results

Field	Value
Event Type	Allocation outside TLAB
Start Time	17.03.2020, 23:37:40
Event Thread	http-nio-8080-exec-1955
Object Class	java.lang.Object[]
Allocation Size	52,8 MiB 1 events

TLAB Allocations

<No Selection> Aspect: <No Selection> ☐ Show concurrent: ☐ Contained ☒ Same threads Time Range: Set Clear

Thread	Count	Average TLAB All	Average Allocated	Est. TLAB Allocated
http-nio-8080-exec-1959	6		52,8 MiB	

☒ Est. TLAB Allocation

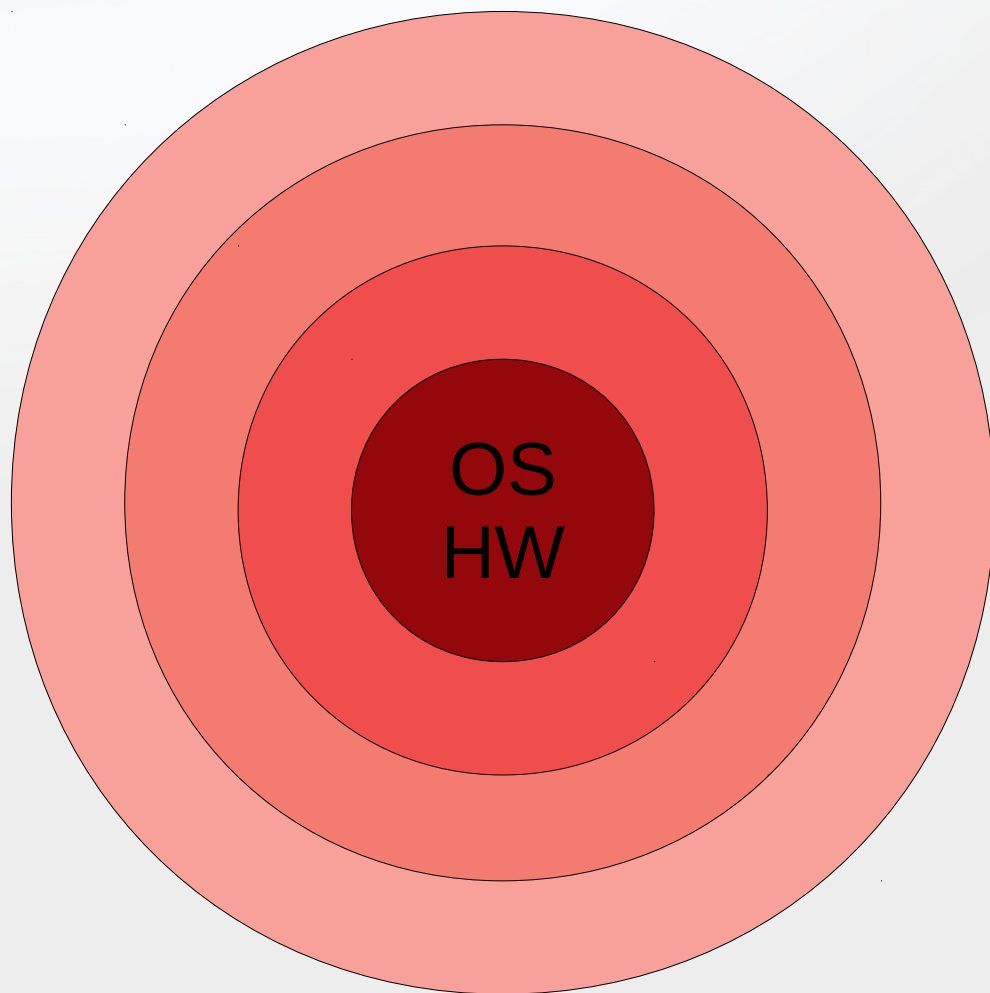
Stack Trace Flame View

Stack Trace

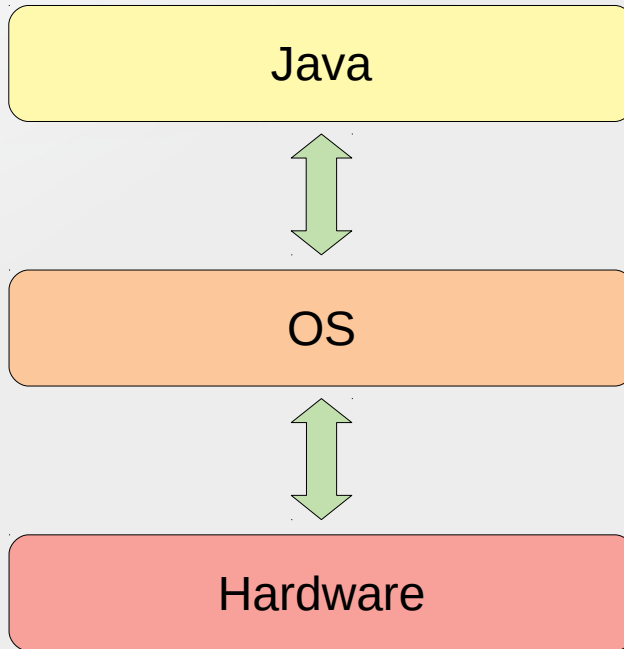
	Count
Object[] java.util.Arrays.copyOf(Object[], int)	1
Object[] java.util.ArrayList.grow(int)	1
Object[] java.util.ArrayList.grow()	1
void java.util.ArrayList.add(Object, Object[], int)	1
boolean java.util.ArrayList.add(Object)	1
List org.hibernate.loader.plan.exec.process.internal.ResultSetProcessorImpl.extractResults(ResultSet, SharedSessionContractImplementor)	1
List org.hibernate.loader.plan.exec.internal.AbstractLoadPlanBasedLoader.executeLoad(SharedSessionContractImplementor)	1
List org.hibernate.loader.plan.exec.internal.AbstractLoadPlanBasedLoader.executeLoad(SharedSessionContractImplementor)	1
Object org.hibernate.loader.entity.plan.AbstractLoadPlanBasedEntityLoader.load(Serializable, Object, SharedSessionContractImplementor)	1
Object org.hibernate.persister.entity.AbstractEntityPersister.load(Serializable, Object, LockOptions, SharedSessionContractImplementor)	1
Object org.hibernate.event.internal.DefaultLoadEventListener.loadFromDatasource(LoadEvent, EntityPersister, EntityKey)	1
Object org.hibernate.event.internal.DefaultLoadEventListener.doLoad(LoadEvent, EntityPersister, EntityKey)	1
Object org.hibernate.event.internal.DefaultLoadEventListener.load(LoadEvent, EntityPersister, EntityKey)	1
Object org.hibernate.event.internal.DefaultLoadEventListener.proxyOrLoad(LoadEvent, EntityPersister, EntityKey)	1
void org.hibernate.event.internal.DefaultLoadEventListener.doOnLoad(EntityPersister, LoadEvent, LoadEventListener)	1
void org.hibernate.event.internal.DefaultLoadEventListener.onLoad(LoadEvent, LoadEventListener)	1

86M of 139M

Kręgi piekła



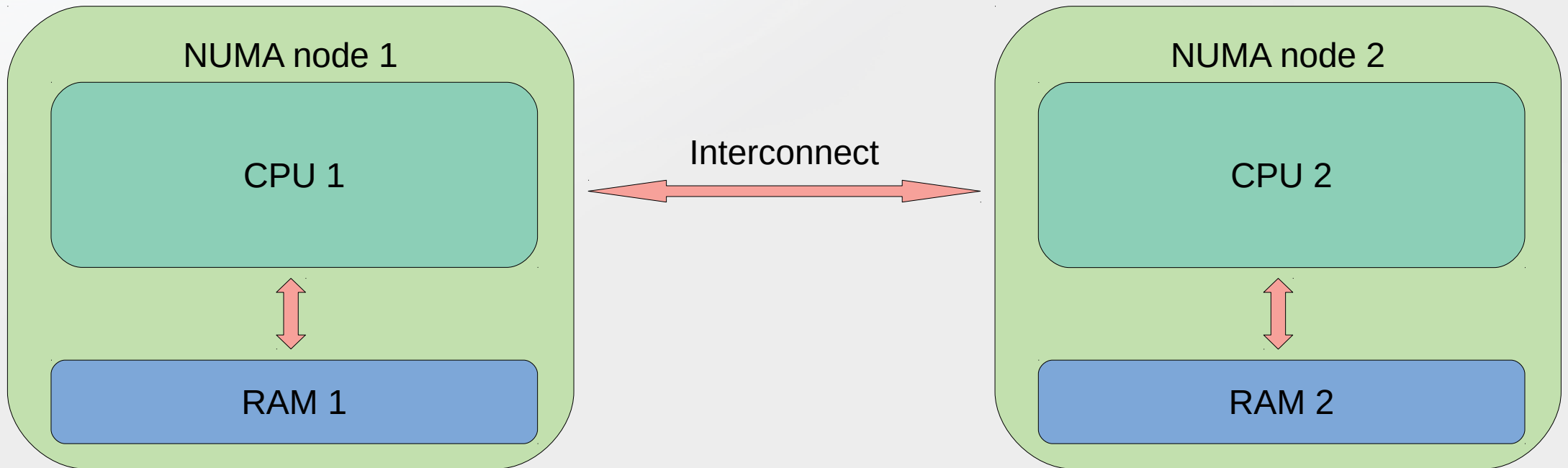
JVM to proces



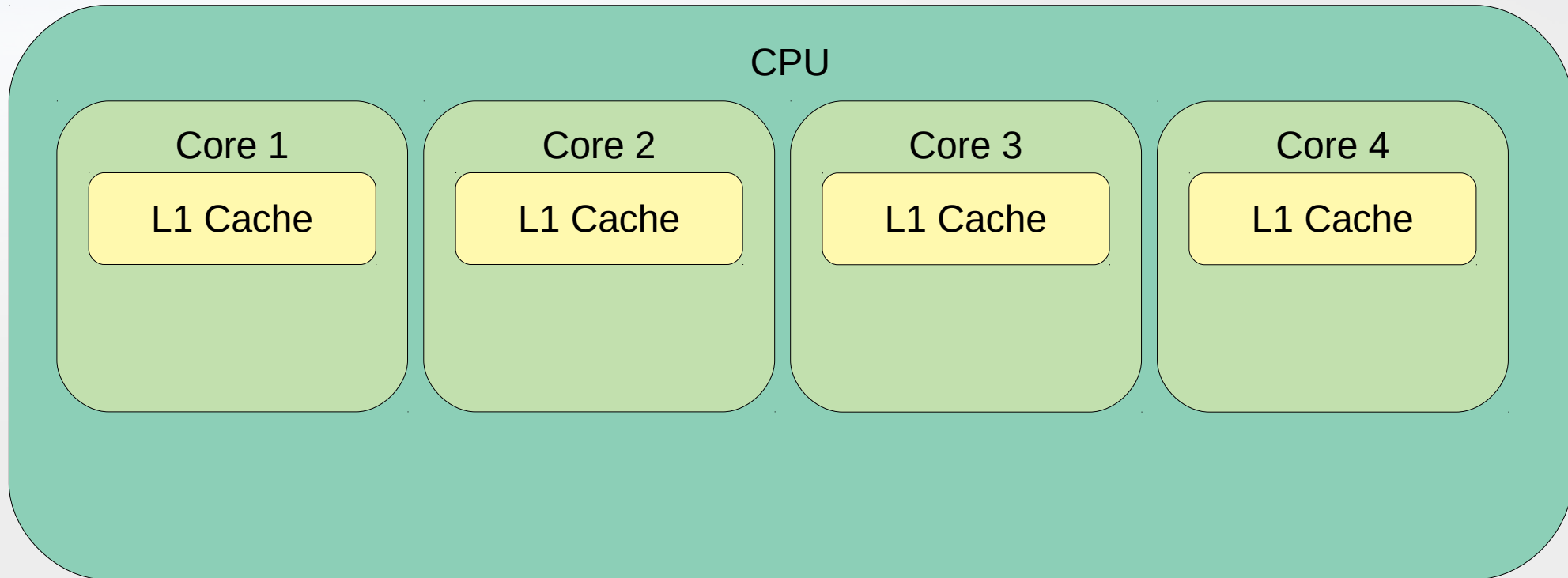
OS

- Co proces chce od OS:
 - strace -f -T -p<pid>
- Jakie zasoby zjada proces:
 - top / htop
 - pidstat -w -t -l -p <pid>
 - iostat / iotop

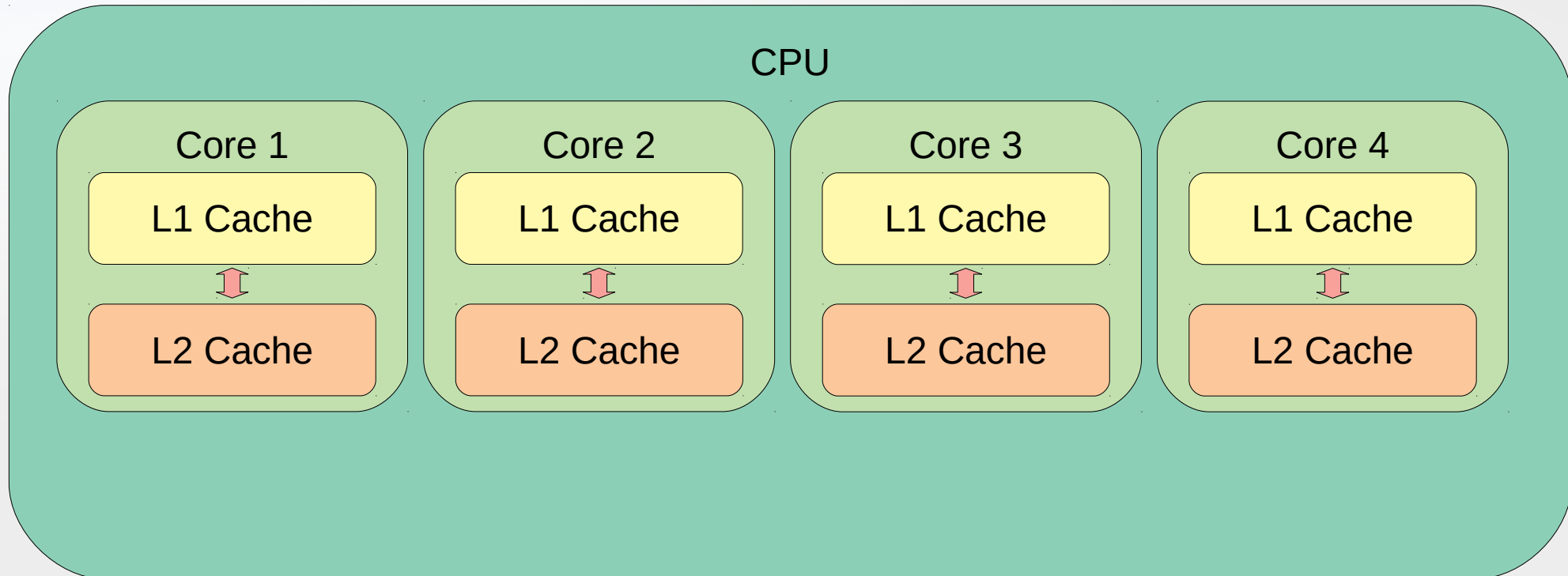
Hardware - NUMA



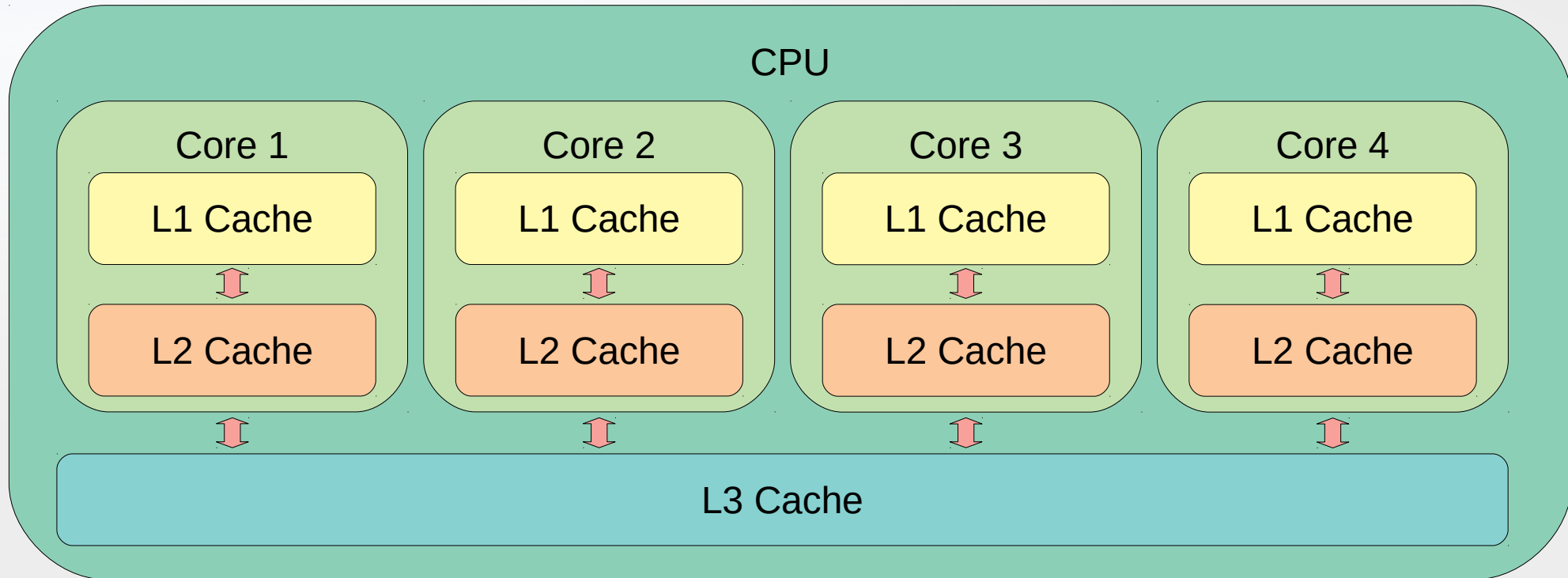
Hardware



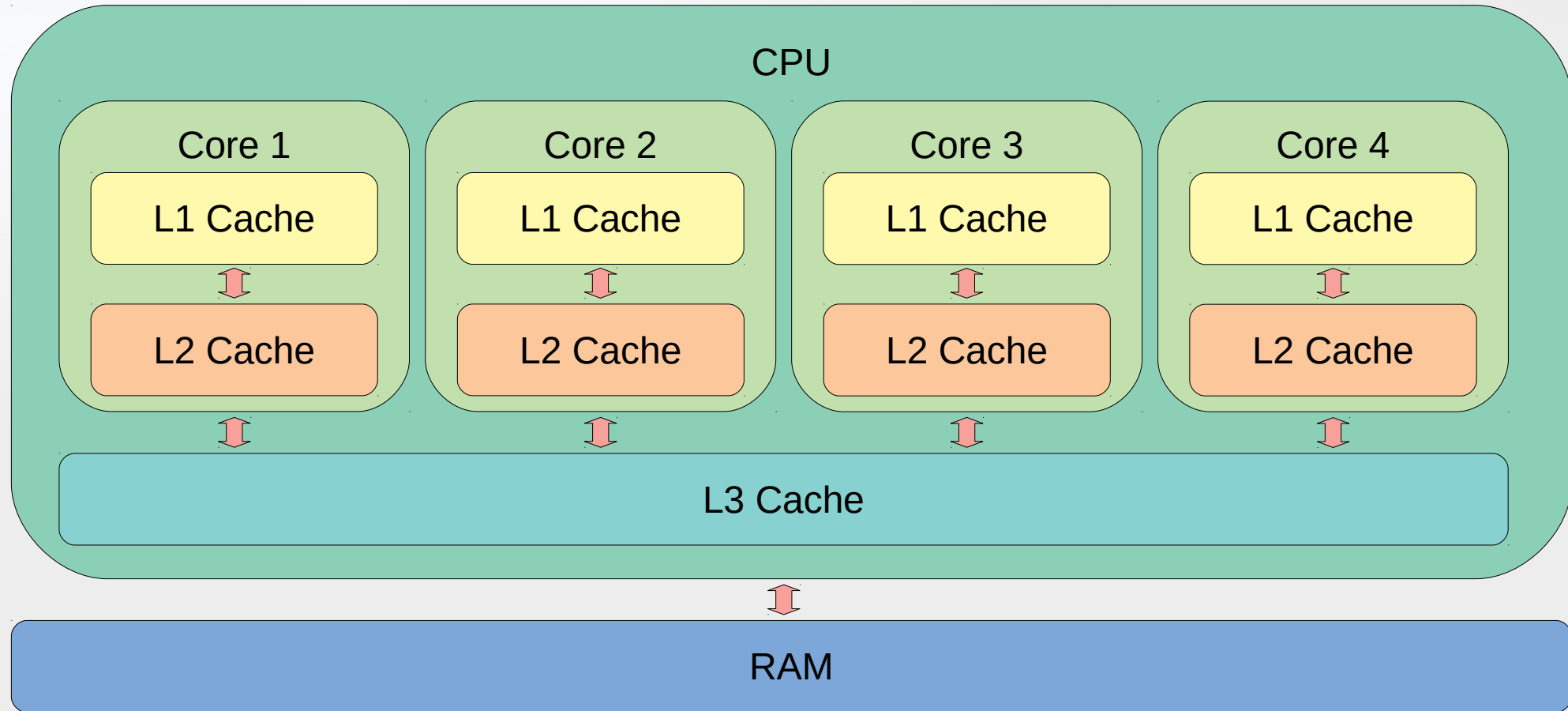
Hardware



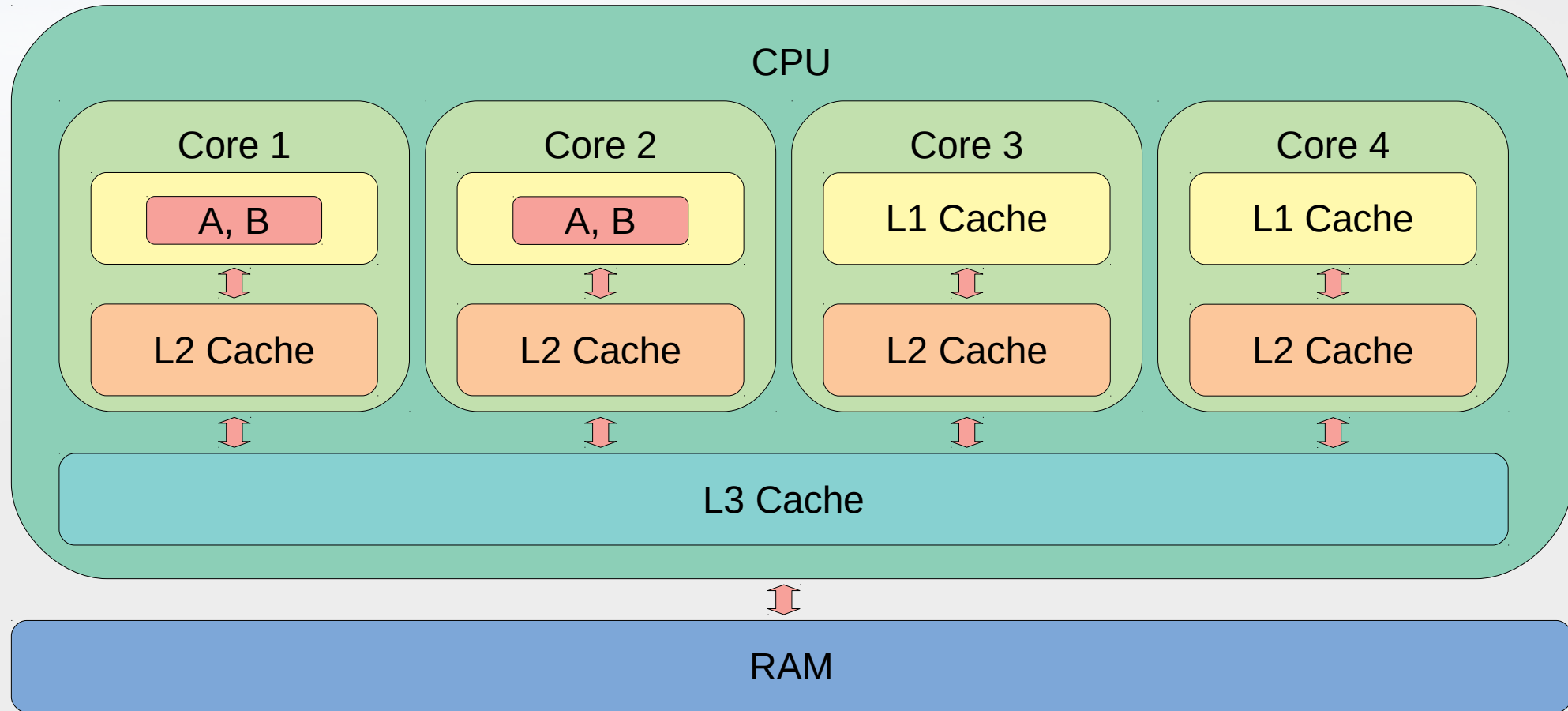
Hardware



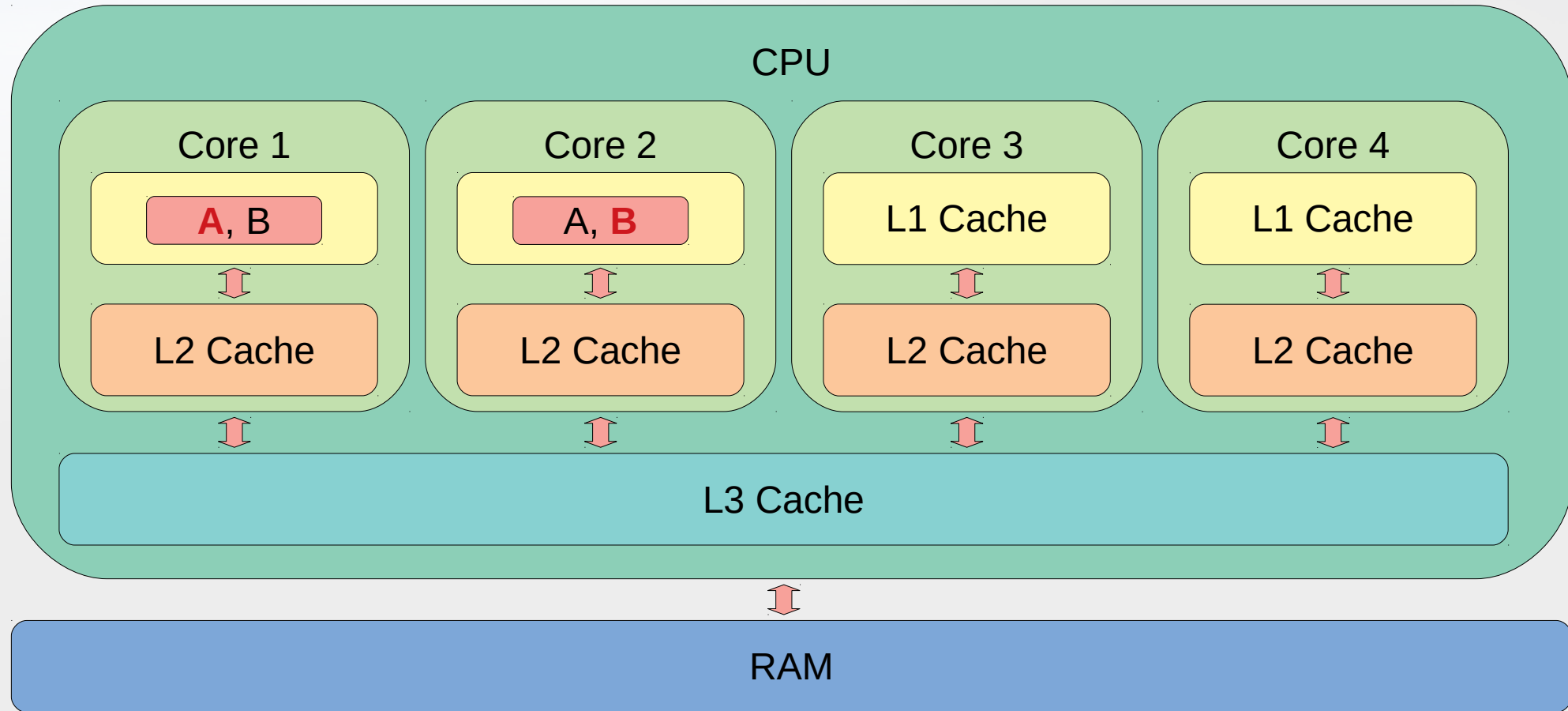
Hardware



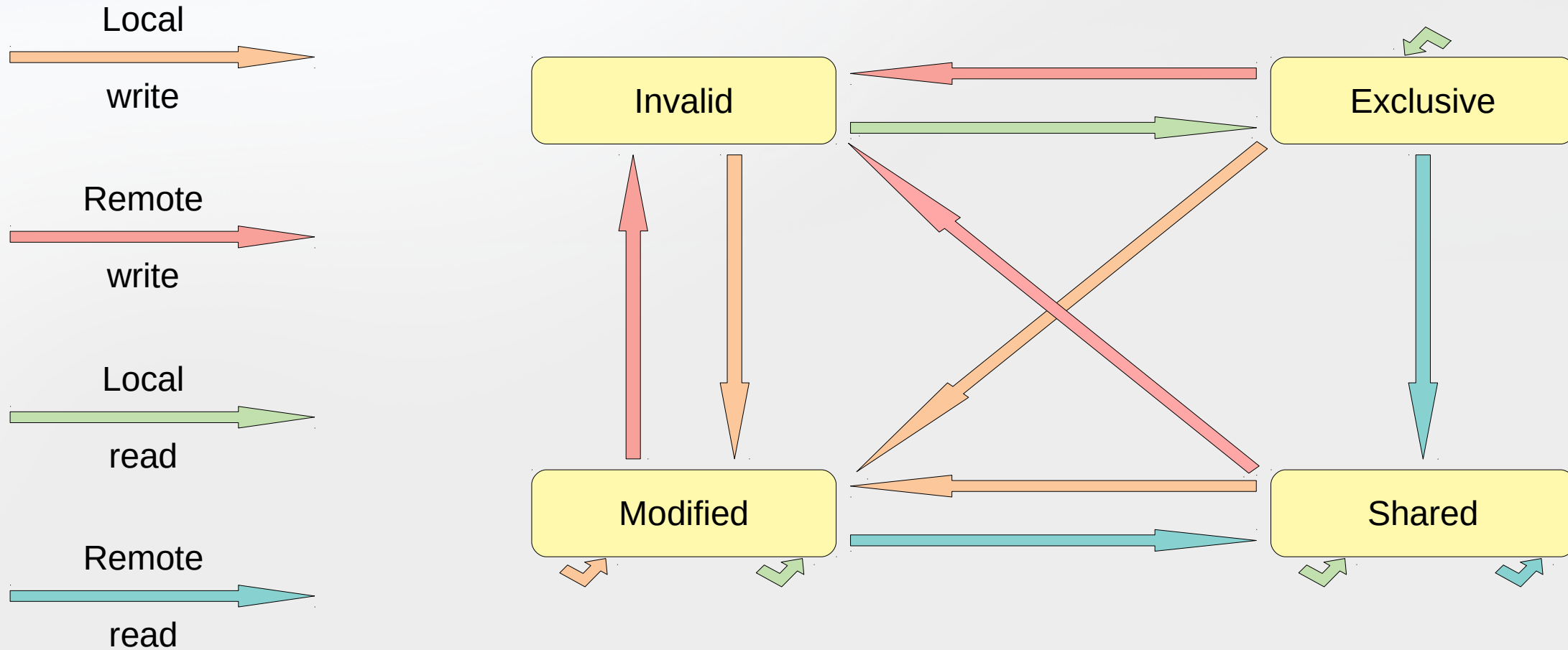
Hardware - false sharing



Hardware - false sharing



MESI - coherence protocol



False sharing - przykład

Złote rady wujka Krzyśka

Złote rady wujka Krzyśka

- Wypróbuj software profilujący darmowy i płatny

Złote rady wujka Krzyśka

- Wypróbuj software profilujący darmowy i płatny
- Pamiętaj, że sampling to tylko „nakierowanie”

Złote rady wujka Krzyśka

- Wypróbuj software profilujący darmowy i płatny
- Pamiętaj, że sampling to tylko „nakierowanie”
- Masz przestój w projekcie --> profiluj

Złote rady wujka Krzyśka

- Wypróbuj software profilujący darmowy i płatny
- Pamiętaj, że sampling to tylko „nakierowanie”
- Masz przestój w projekcie --> profiluj
- Przed profilowaniem na produkcji

Złote rady wujka Krzyśka

- Wypróbuj software profilujący darmowy i płatny
- Pamiętaj, że sampling to tylko „nakierowanie”
- Masz przestój w projekcie --> profiluj
- Przed profilowaniem na produkcji
 - Jeżeli obawiasz się overheadu --> zmniejsz ruch do profilowanej maszyny

Złote rady wujka Krzyśka

- Wypróbuj software profilujący darmowy i płatny
- Pamiętaj, że sampling to tylko „nakierowanie”
- Masz przestój w projekcie --> profiluj
- Przed profilowaniem na produkcji
 - Jeżeli obawiasz się overheadu --> zmniejsz ruch do profilowanej maszyny
 - Uświadom użytkowników/właścicieli biznesowych:

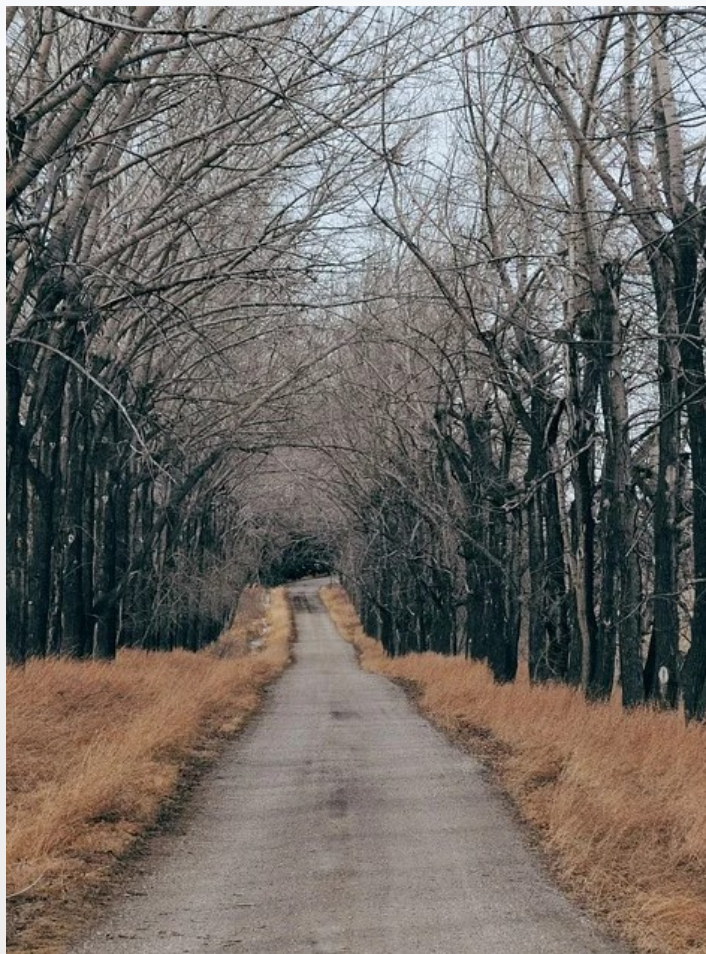
Złote rady wujka Krzyśka

- Wypróbuj software profilujący darmowy i płatny
- Pamiętaj, że sampling to tylko „nakierowanie”
- Masz przestój w projekcie --> profiluj
- Przed profilowaniem na produkcji
 - Jeżeli obawiasz się overheadu --> zmniejsz ruch do profilowanej maszyny
 - Uświadom użytkowników/właścicieli biznesowych:
 - Że to robisz

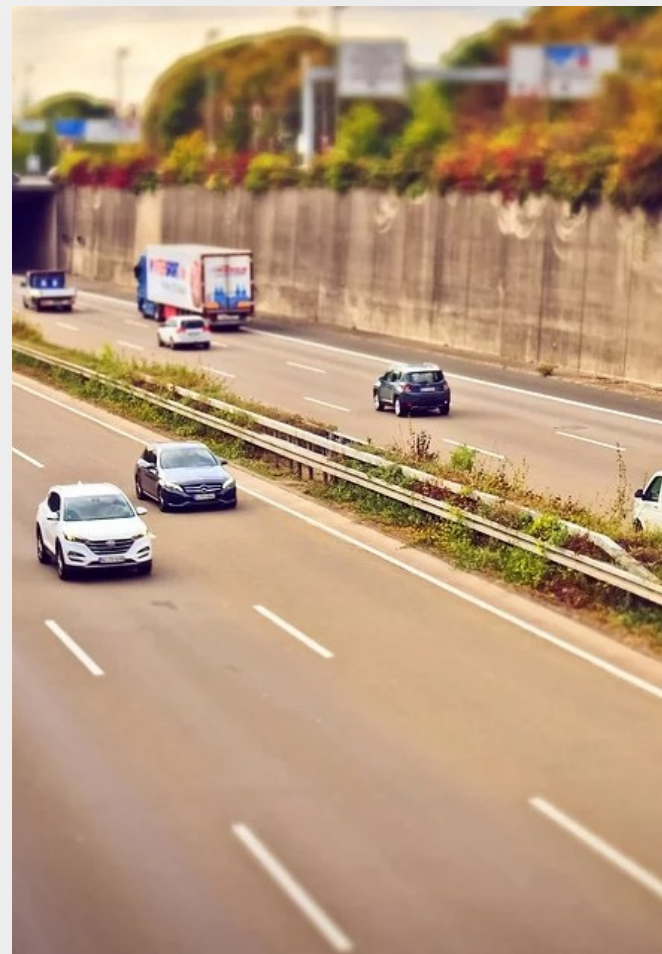
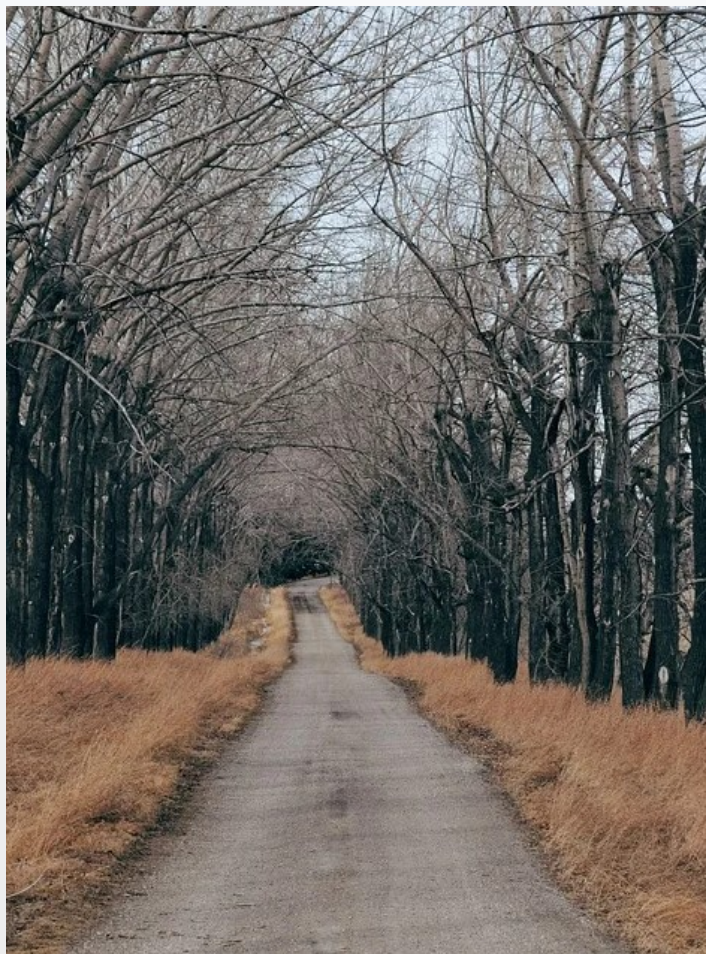
Złote rady wujka Krzyśka

- Wypróbuj software profilujący darmowy i płatny
- Pamiętaj, że sampling to tylko „nakierowanie”
- Masz przestój w projekcie --> profiluj
- Przed profilowaniem na produkcji
 - Jeżeli obawiasz się overheadu --> zmniejsz ruch do profilowanej maszyny
 - Uświadom użytkowników/właścicieli biznesowych:
 - Że to robisz
 - Że to spowolni na chwilę aplikację

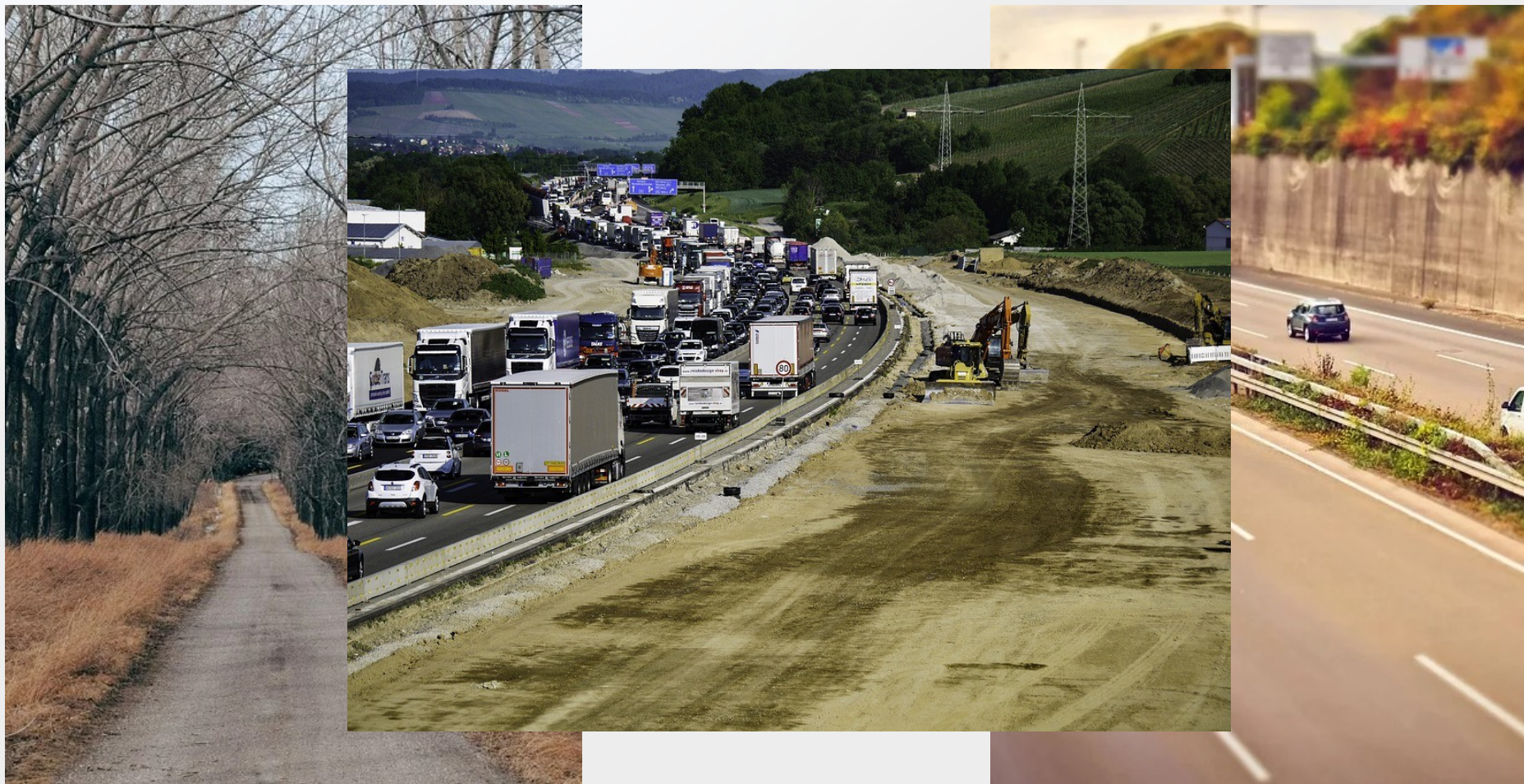
Ma być lepiej? Trzeba cierpieć



Ma być lepiej? Trzeba cierpieć



Ma być lepiej? Trzeba cierpieć



Co dalej?

Co dalej?



Co dalej?



<http://jug2020.kś.pl/>

<http://kś.pl>
<http://gclogs.com>

conf@kś.pl
ks@gclogs.com

Dziękuję



<http://jug2020.kś.pl/>