

Profiling in Java with JProfiler

Helmes

Before we start

- git clone <https://github.com/raitraidma/profiling.git>
 - mvn clean install
 - mvn spring-boot:run
 - Do not peek into presentation folder!
- <https://www.ej-technologies.com/download/jprofiler/files>

Helmes

Why and when?

Helmes

Why and when?

- You are here
- Something is slow
- Something uses too much memory
- Need to optimize

Helmes

Java optimizations

- Optimizations you get for free!

Helmes

Compilation time optimization (1.1)

```
public class CompilerOptimizationWithFinalExample {  
    private static final boolean IS_PRINTING = false;  
  
    public static void main(String[] args) {  
        for (int i = 0; i < 100_000_000; i++) {  
            if (IS_PRINTING) {  
                System.out.println("i = " + i);  
            }  
        }  
    }  
}
```

Helmes

Compilation time optimization (1.2)

```
public class CompilerOptimizationWithFinalExample {  
    private static final boolean IS_PRINTING = false;  
  
    public CompilerOptimizationWithFinalExample() {  
    }  
  
    public static void main(String[] args) {  
        for(int i = 0; i < 1000000000; ++i) {  
        }  
    }  
}
```

Helmes

Compilation time optimization (2.1)

```
public class CompilerOptimizationWithoutFinalExample {  
    private static boolean IS_PRINTING = false;  
  
    public static void main(String[] args) {  
        for (int i = 0; i < 100_000_000; i++) {  
            if (IS_PRINTING) {  
                System.out.println("i = " + i);  
            }  
        }  
    }  
}
```

Helmes

Compilation time optimization (2.2)

```
public class CompilerOptimizationWithoutFinalExample {  
    private static boolean IS_PRINTING = false;  
  
    public CompilerOptimizationWithoutFinalExample() {  
    }  
  
    public static void main(String[] args) {  
        for(int i = 0; i < 1000000000; ++i) {  
            if (IS_PRINTING) {  
                System.out.println("i = " + i);  
            }  
        }  
    }  
}
```

Compilation time optimization (3.1)

```
public class LiteralConstantExample {  
    public static void main(String[] args) {  
        int age = 3 * 7 + 1;  
  
        final String firstName = "Jon";  
        final String lastName = "Snow";  
        String fullName = firstName + " " + lastName;  
  
        String firstName2 = "Daenerys";  
        String lastName2 = "Targaryen";  
        String fullName2 = firstName2 + " " + lastName2;  
  
        System.out.println(fullName + " is " + age + " years old");  
        System.out.println(fullName2 + " is also " + age + " years old");  
    }  
}
```

Helmes

Compilation time optimization (3.2)

```
public class LiteralConstantExample {  
    public LiteralConstantExample() {  
    }  
  
    public static void main(String[] args) {  
        int age = 22;  
        String firstName = "Jon";  
        String lastName = "Snow";  
        String fullName = "Jon Snow";  
        String firstName2 = "Daenerys";  
        String lastName2 = "Targaryen";  
        String fullName2 = firstName2 + " " + lastName2;  
        System.out.println(fullName + " is " + age + " years old");  
        System.out.println(fullName2 + " is also " + age + " years old");  
    }  
}
```

Helmes

Runtime optimization

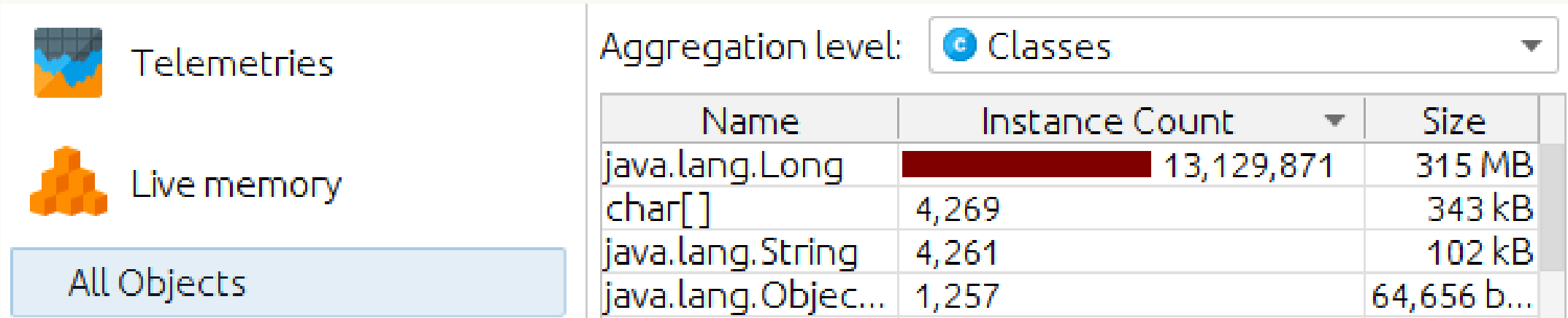
- Hotspot JIT can detect what kind of processor you have and generates code accordingly.

Own intelligence

```
public class GarbageCollectorExample {  
    public static void main(String[] args) {  
        Long count = 0L;  
  
        for (int i = 0; i < 100_000_000; i++) {  
            count++;  
        }  
    }  
}
```

Helmes

Own intelligence



- Took 519 ms

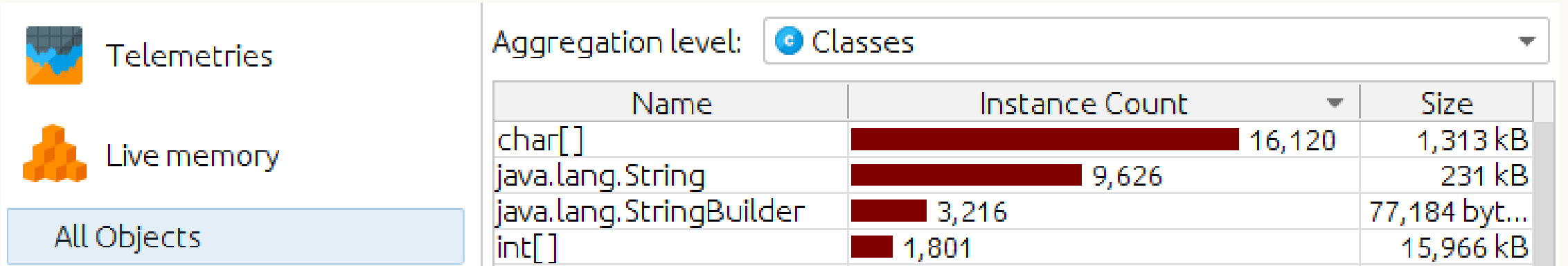
Helmes

Own intelligence

```
public class GarbageCollectorFixExample {  
    public static void main(String[] args) {  
        long count = 0L;  
  
        for (int i = 0; i < 100_000_000; i++) {  
            count++;  
        }  
    }  
}
```

Helmes

Own intelligence



- Took 5 ms

Helmes

Install JProfiler

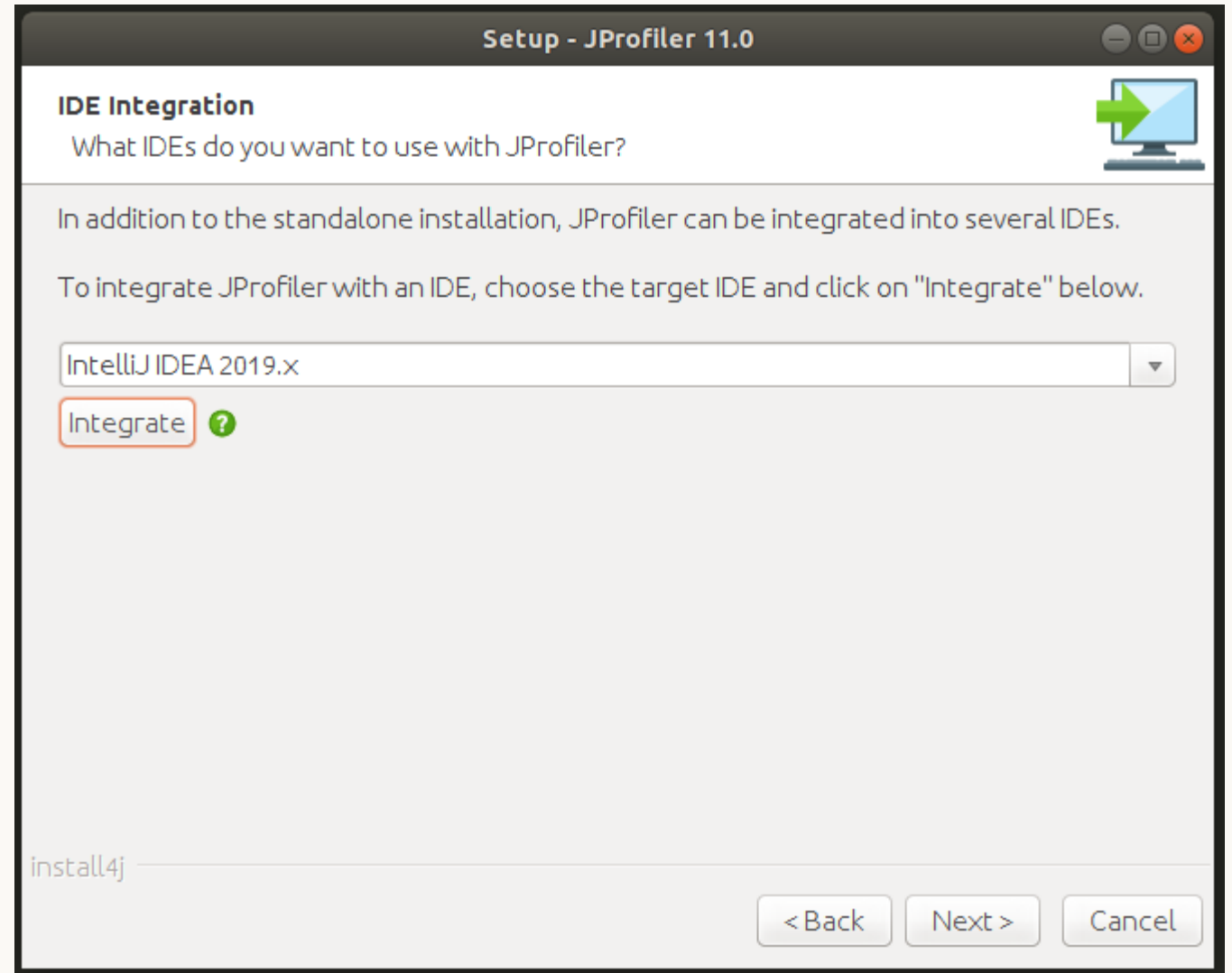
Download

- <https://www.ej-technologies.com/download/jprofiler/files>

IDE integration

- Close IDE
- Select IDE configuration folder

Helmes



Attach JProfiler

- -agentpath VM parameter
 - -agentpath:/opt/jprofiler/bin/linux-x64/libjprofilerti.so=port=8849,nowait

Instrumentation vs Sampling

Initial Profiling Settings

Please choose your initial use case:

➔ Instrumentation

All features, such as invocation counts and method statistics are supported. Good filters are critical for overhead.

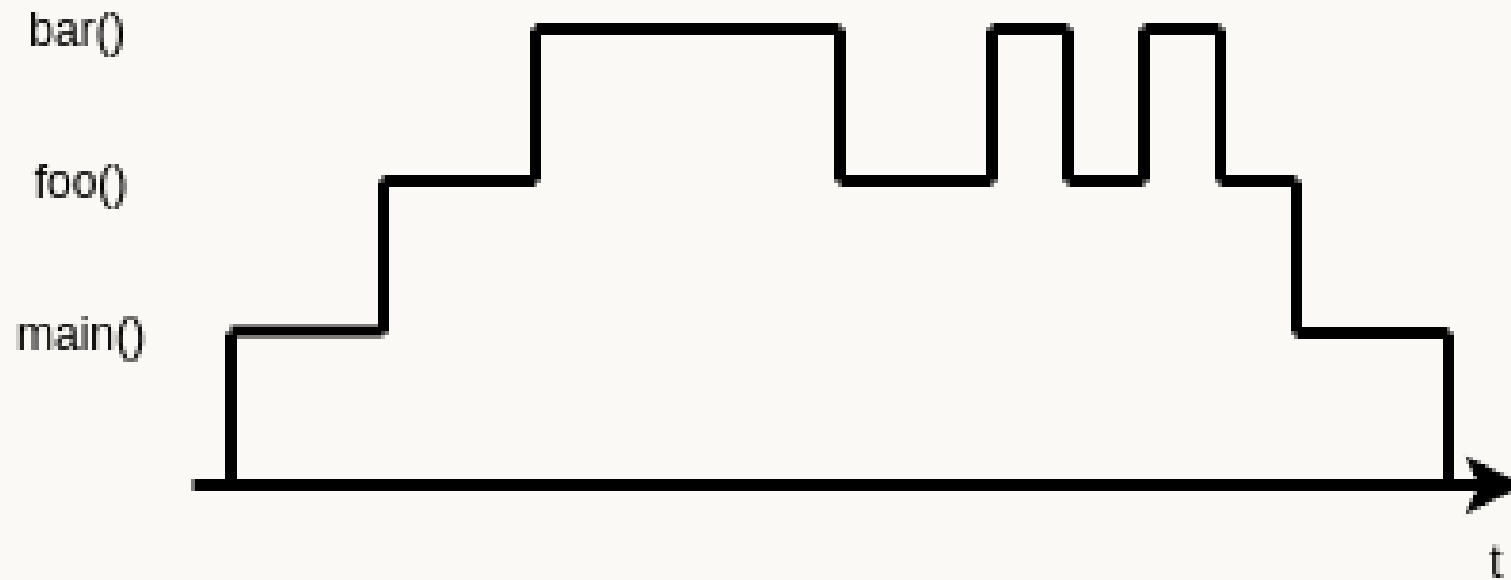
➔ Sampling (Recommended)

CPU profiling is not distorted by short-running methods. Overhead is extremely low. Some features are not supported. This mode is safer when attaching to running JVMs.

Profiling settings can be changed later on at any time.

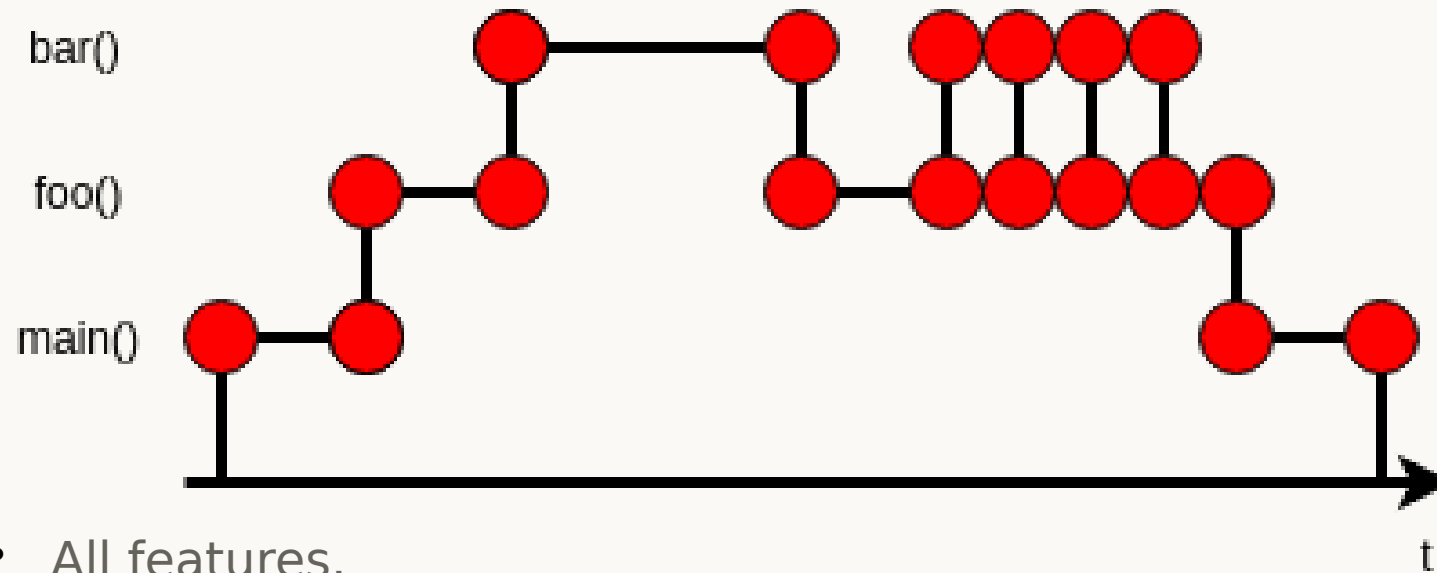
Helmes

Instrumentation vs Sampling



Helmes

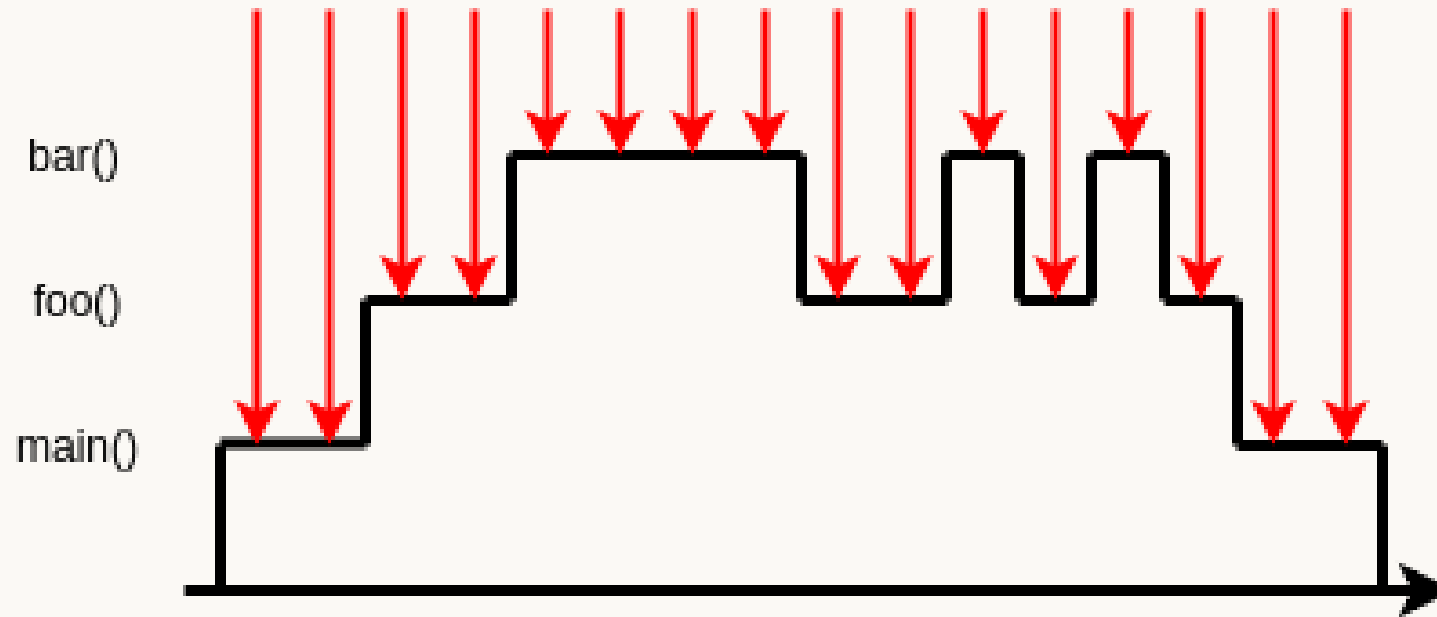
Instrumentation



- All features.
- Filters must be used correctly. Otherwise overhead.

Helmes

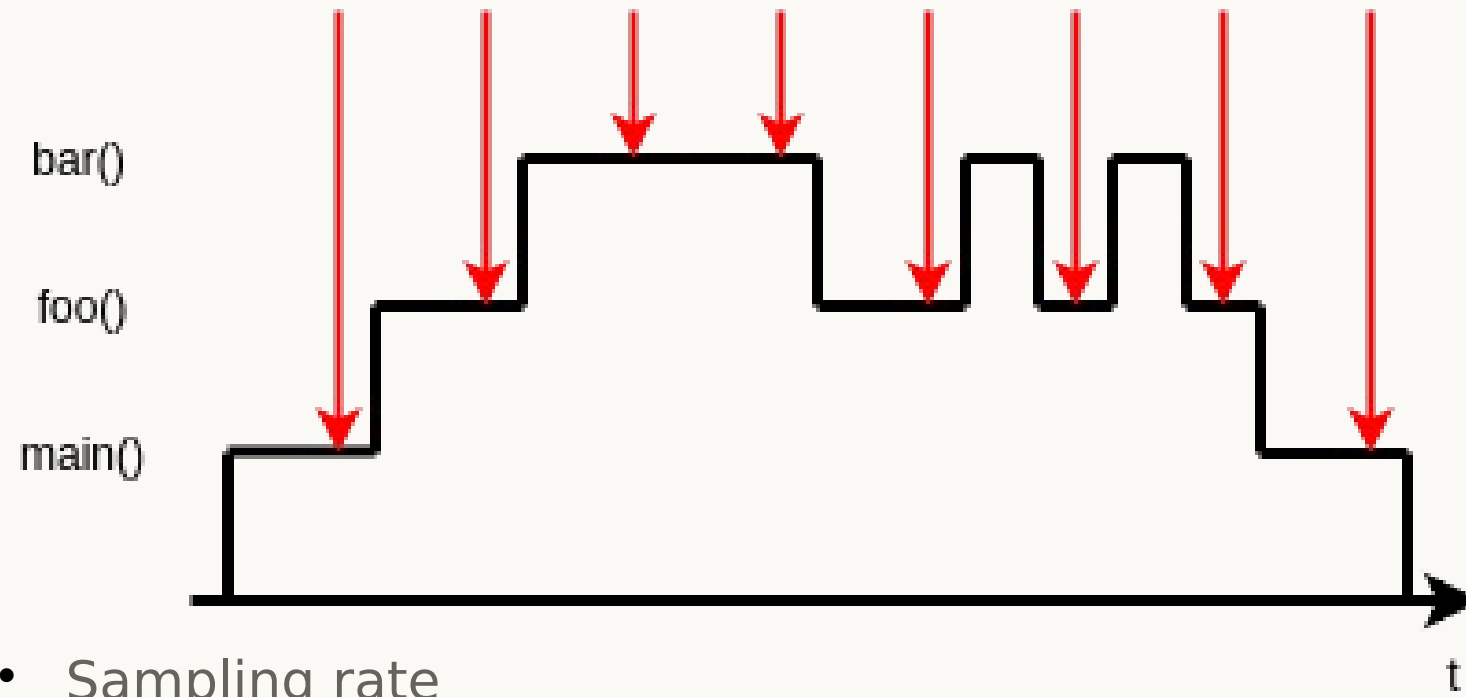
Sampling



- When you do not know where the bottlenecks are.
- Less overhead.

Helmes

Sampling



Helmes

Instrumentation vs Sampling - Example

- `GarbageCollectorExample`
- CPU recording enabled

Memory profiling

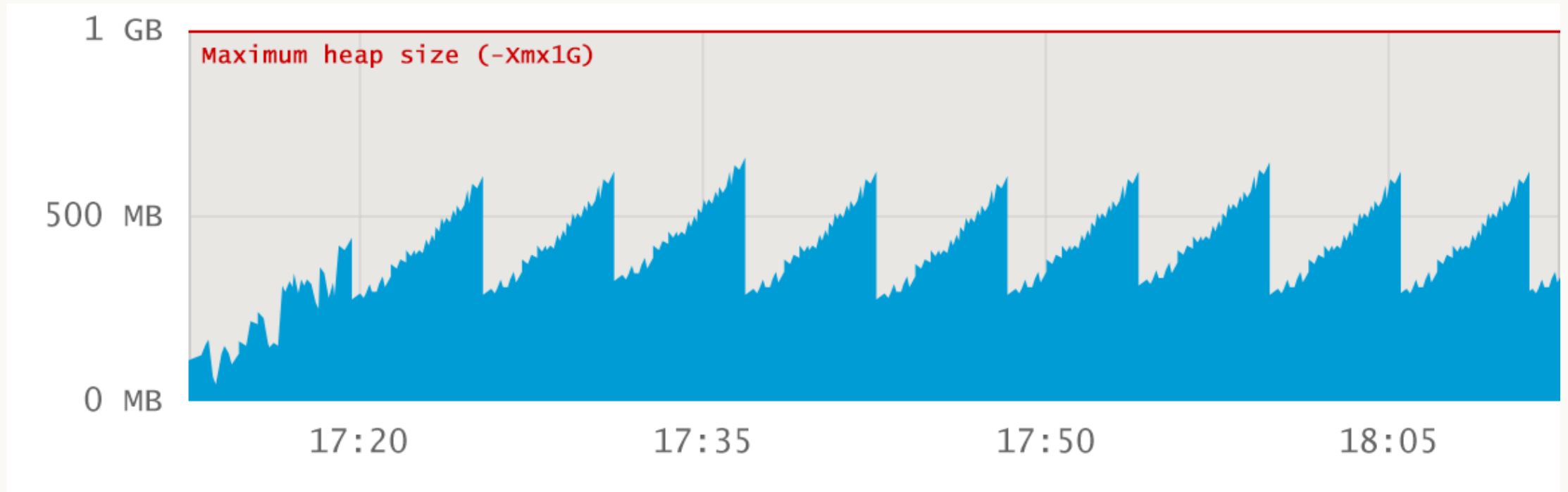
- Memory leak

Helmes

Memory profiling

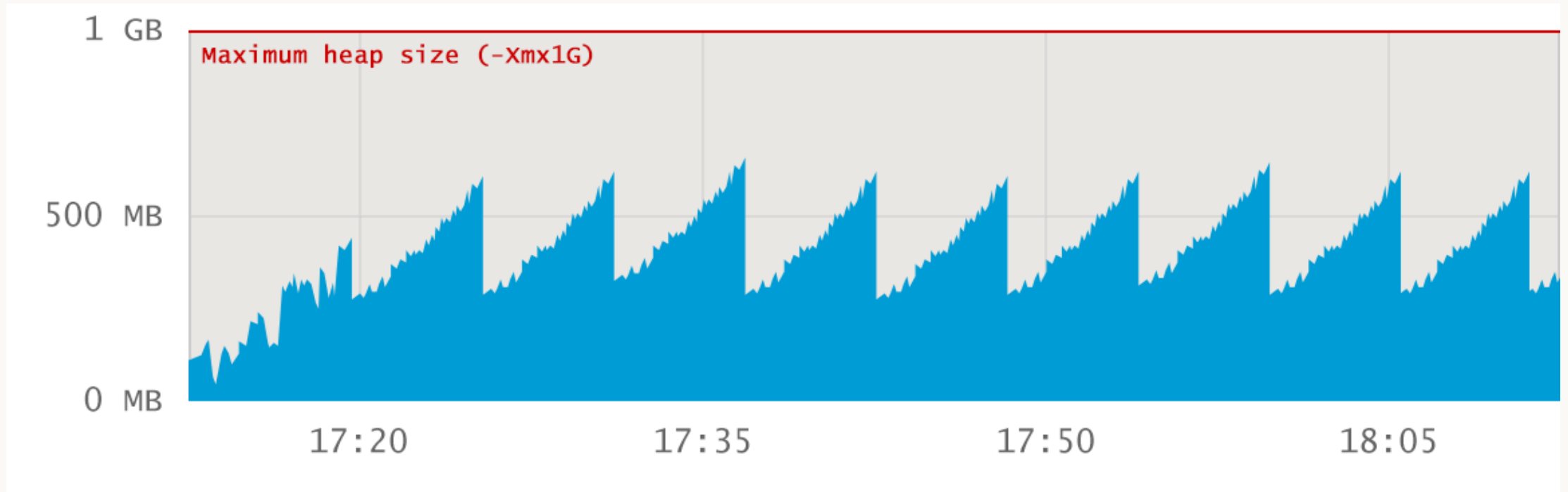
- Memory leak: unused objects cannot be garbage collected.

Memory profiling



Helmes

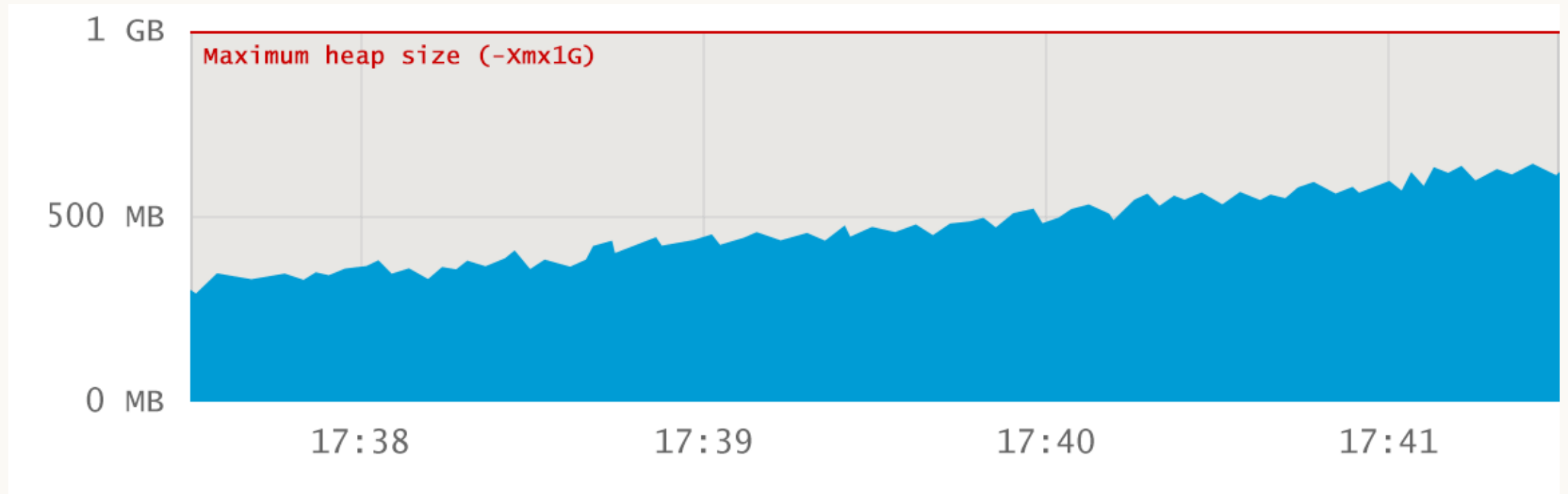
Memory profiling – Healthy JVM



- No memory leak. Flat baseline trend.

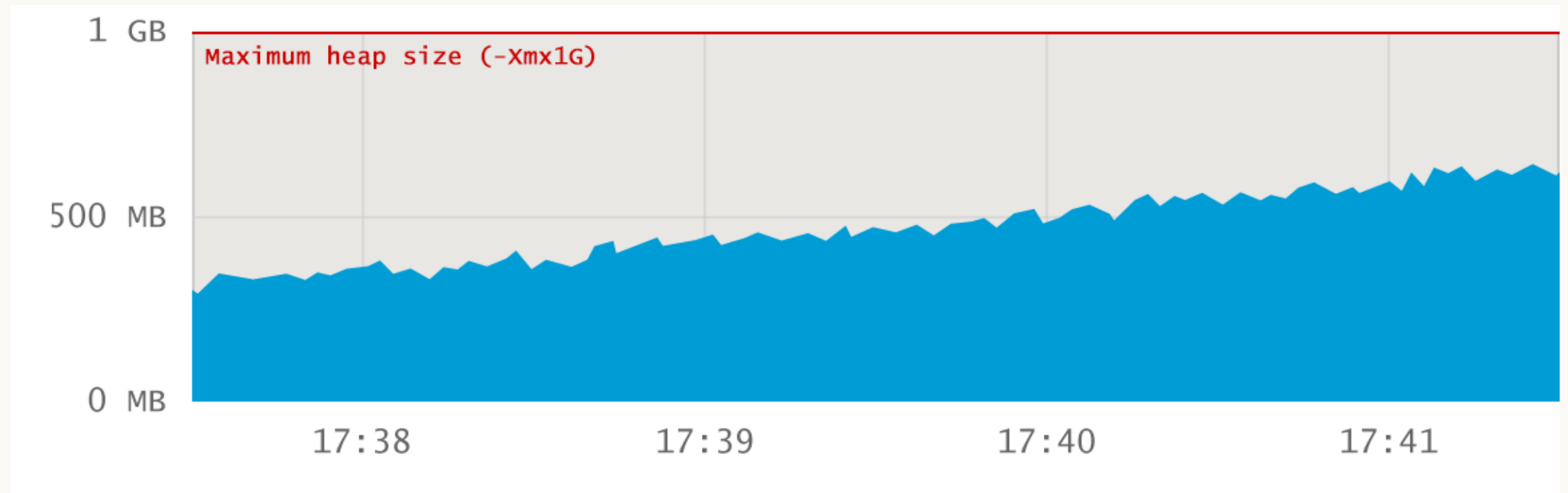
Helmes

Memory profiling



Helmes

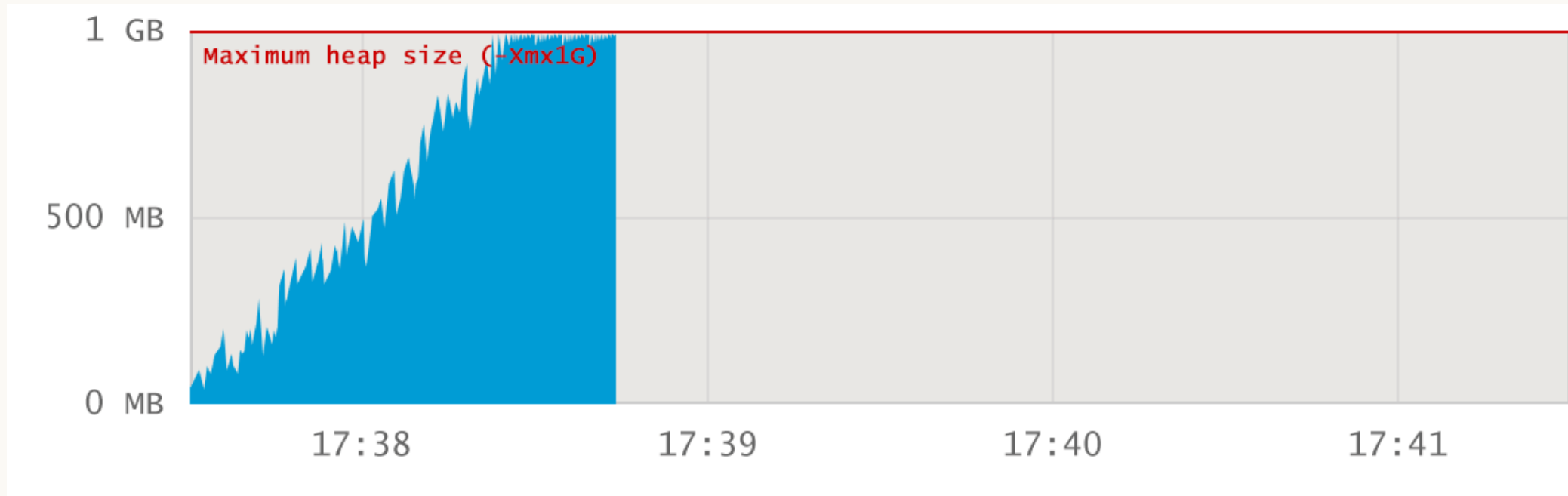
Memory profiling – Healthy JVM



- No memory leak. Period does not contain major GC events.

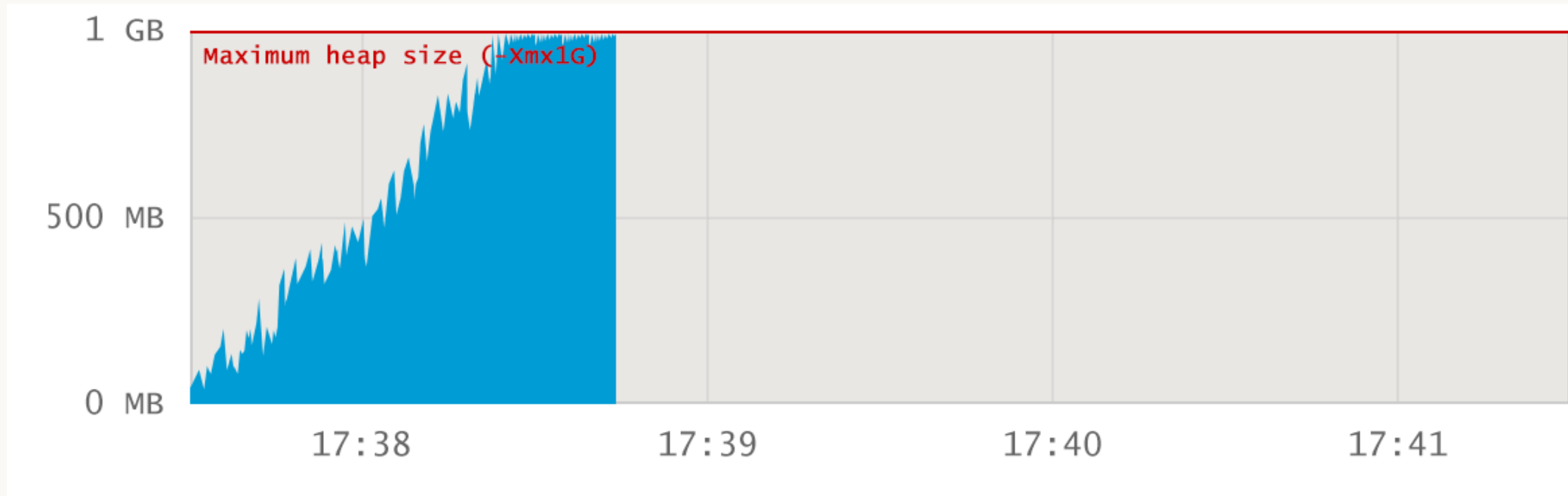
Helmes

Memory profiling



Helmes

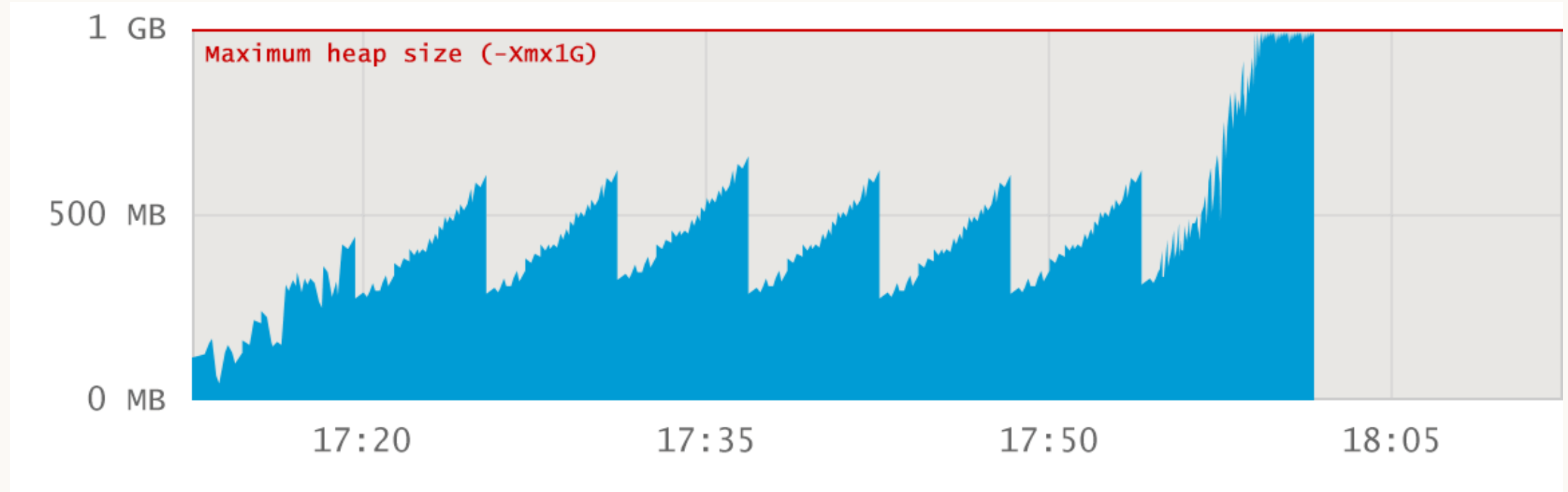
Memory profiling – Memory explosion at startup



- Probably not a memory leak. Big application has not given enough heap space.

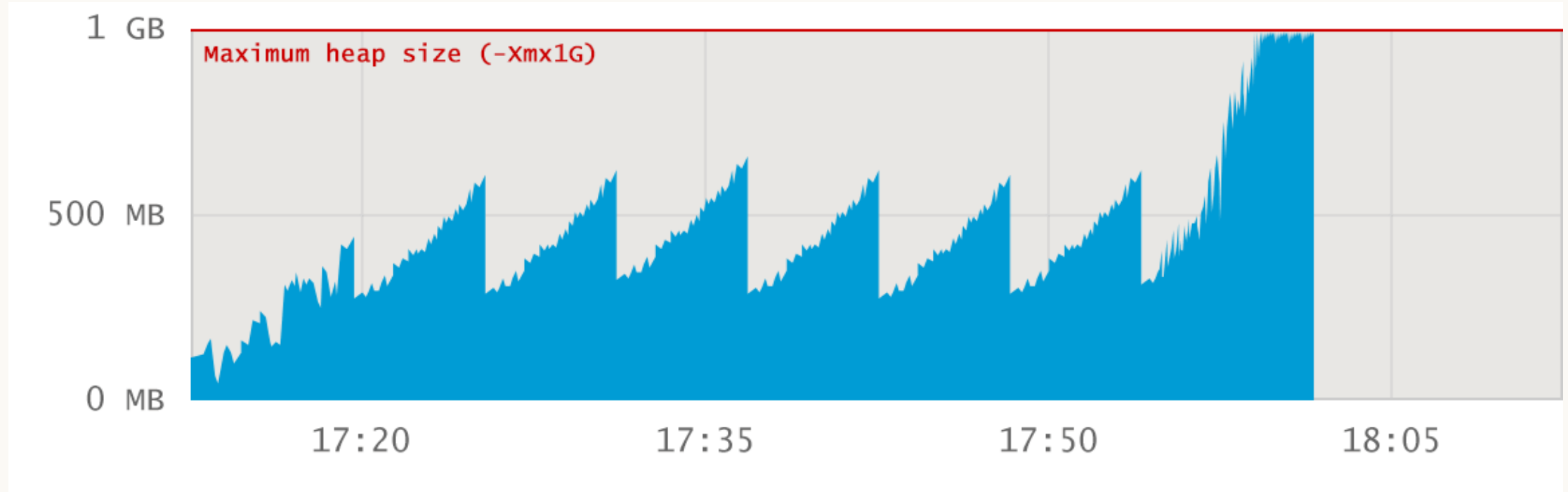
Helmes

Memory profiling



Helmes

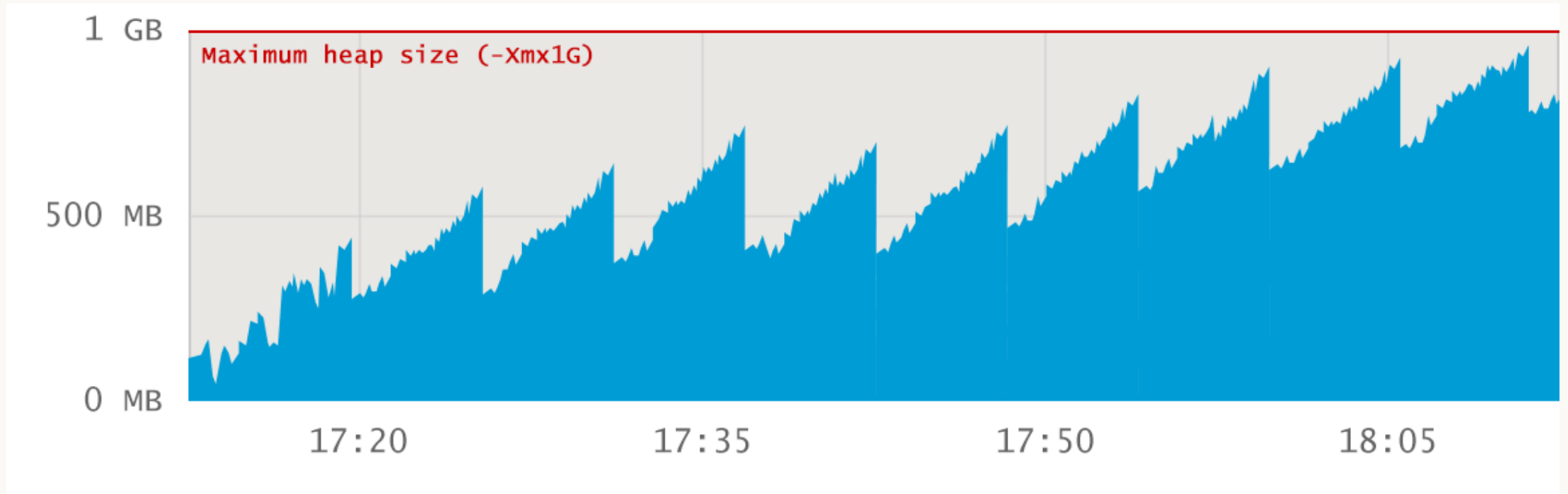
Memory profiling – Surge allocation



- Probably not a memory leak. Too much data is loaded via specific action.

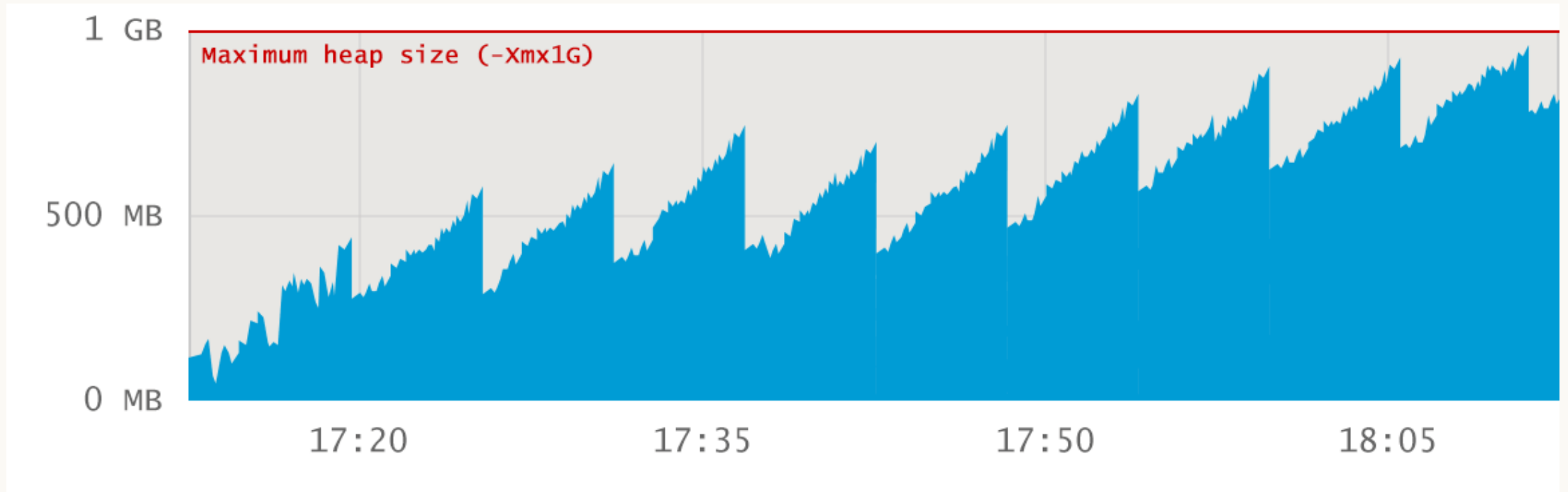
Helmes

Memory profiling



Helmes

Memory profiling – Leaking application



- Memory leak. Baseline growth.

Helmes

Memory profiling – Example

- Call tree Recording:
`Instrumentation`
- Initial recording profile:
`Allocation call stacks`
- `Heap Walker`
- Pin `Mark Heap`
- `Take a Snapshot`
- `Use new` > `Classes` > `Ok`
- Double-click on "interesting" class name
 - `References` > `Incoming references`
- Select "interesting" object
 - `Show Paths To GC Root` > `Single root`
 - Now you see where leaked objects are stored
- Select `Merged incoming references`
- Allocations
 - Show code

CPU profiling

- Issues:
 - Frequently invoked
 - Inefficient code/algorithm

Helmes

CPU profiling - Example

- Call tree Recording: `Instrumentation`
- Initial recording profile: `CPU recording`
- `CPU views`
 - `Hot Spots`
 - Thread status: All states
- Watch `Self Time`, `Average Time` and `Invocations`

Helmes

Hands on!

- Run application: ProfilingApplication
- Create load:
 - `mvn gatling:test`

Helmes

Happy Profiling!

Helmes