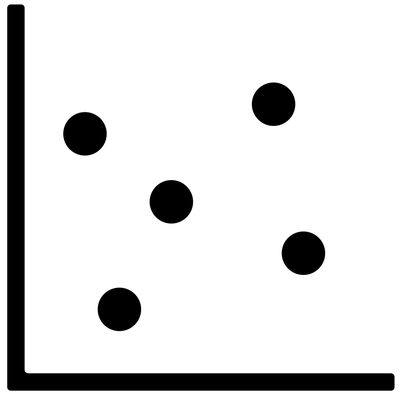
How do applications use OOP?

Seminar

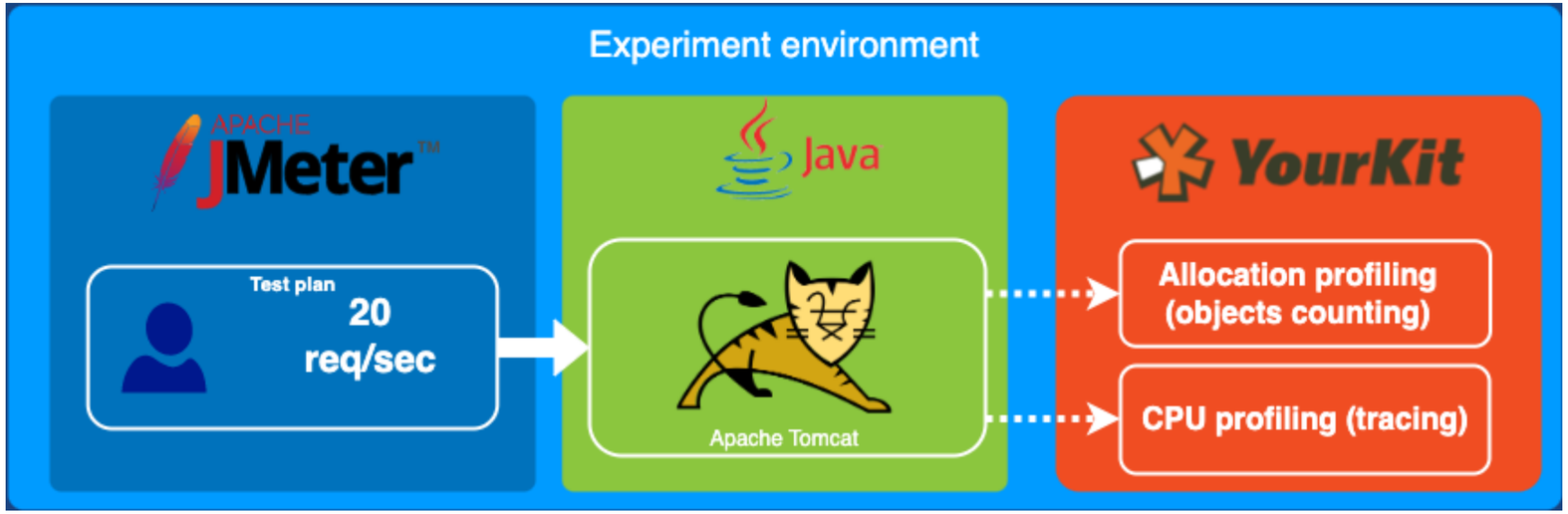
Zakharov Vladimir 27.06.2023

Research Questions

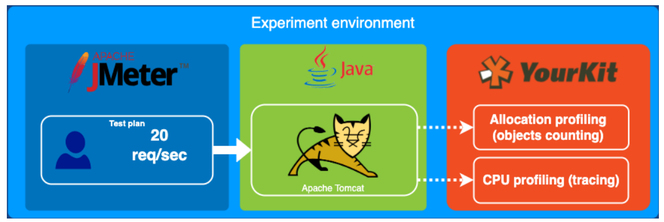
- How many “static” methods are we using?
- How many “instance” methods are we using?
- How many objects do we usually create?
- Do we have some trends in the received data?



How?


Experiment environment
APACHE JMeter™
Test plan
20
req/sec
Java
Apache Tomcat
YourKit
Allocation profiling
(objects counting)
CPU profiling (tracing)

How?



CSV

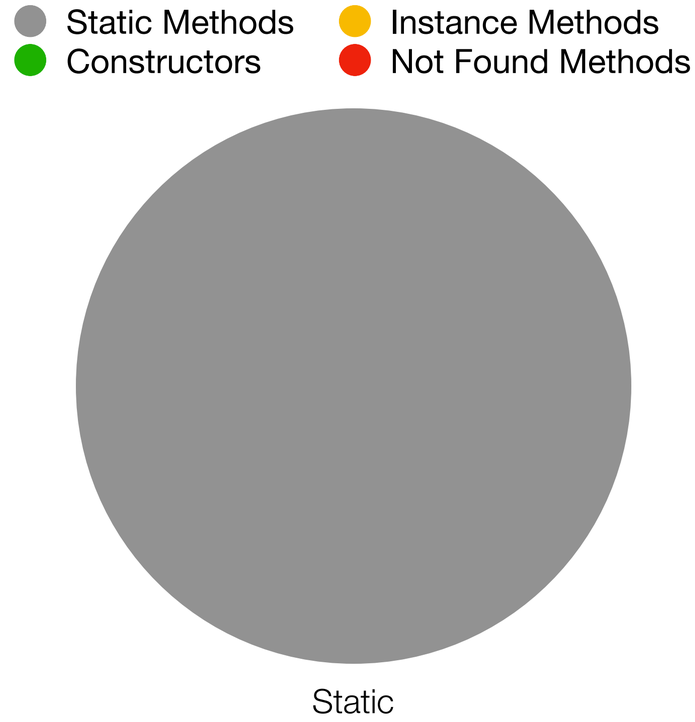
Sources

Verification

Statics only

```java
public class StaticsOnly {

  public static void main(String[] args) {
    int sum = 0;
    for (int i = 0; i < Integer.MAX_VALUE; ++i) {
      sum += discounted(discounted(prime(i)));
      Thread.sleep(10);
    }
    System.out.printf("Total: %d\n", sum);
  }

  private static int prime(final int u) {
    return u;
  }

  private static int discounted(final int u) {
    return u / 2;
  }
}
```

Type	Number
Static	10671
Instance	0
Constructors	0
Not found	0

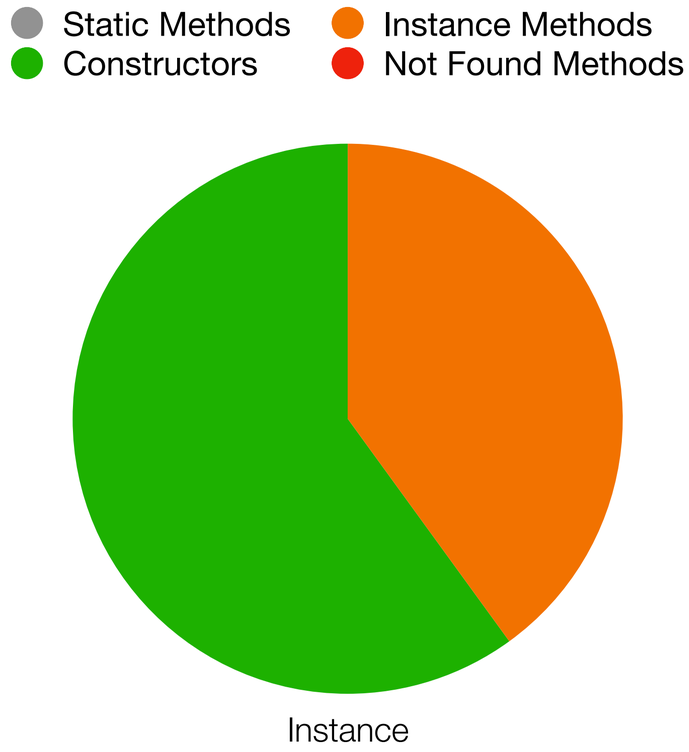


Verification

Instance only

```java
public class InstanceOnly {

  public static void main(final String[] args) {
    int sum = 0;
    for (int i = 0; i < Integer.MAX_VALUE; ++i) {
      Book b = new Discounted(new Discounted(new Prime(i)));
      sum += b.price();
      Thread.sleep(10);
    }
    System.out.printf("Total: %d\n", sum);
  }

  interface Book {
    int price();
  }

  static class Discounted implements Book {
    private final Book book;
    Discounted(final Book b) {
      this.book = b;
    }
    @Override
    public int price() {
      return this.book.price() / 2;
    }
  }

  static class Prime implements Book {
    private final int usd;
    Prime(final int u) {
      this.usd = u;
    }
    @Override
    public int price() {
      return this.usd;
    }
  }
}
```

Type	Number
Static	1
Instance	10526
Constructors	15789
Not found	0

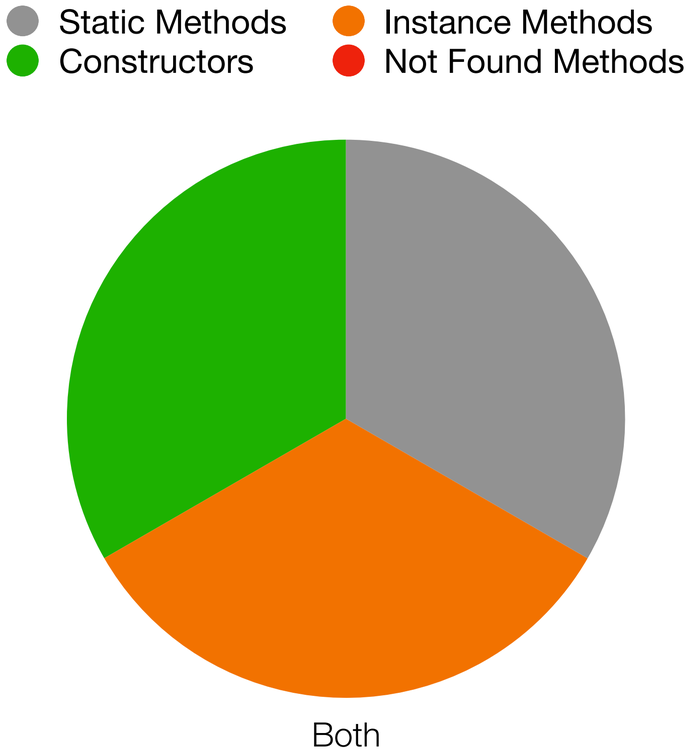


Verification

```java
public class Half {
  public static void main(final String[] args) {
    int sum = 0;
    for (int i = 0; i < Integer.MAX_VALUE; ++i) {
      Book b = new Discounted(i);
      sum += discounted(b.price());
      Thread.sleep(10);
    }
    System.out.printf("Total: %d\n", sum);
  }

  interface Book {
    int price();
  }


  static class Discounted implements Book {
    private final int usd;
    Discounted(final int u) {
      this.usd = u;
    }
    @Override
    public int price() {
      return this.usd / 2;
    }
  }

  private static int discounted(final int u) {
    return u / 2;
  }
}
```

Both

Type	Number
Static	5314
Instance	5313
Constructors	5313
Not found	0

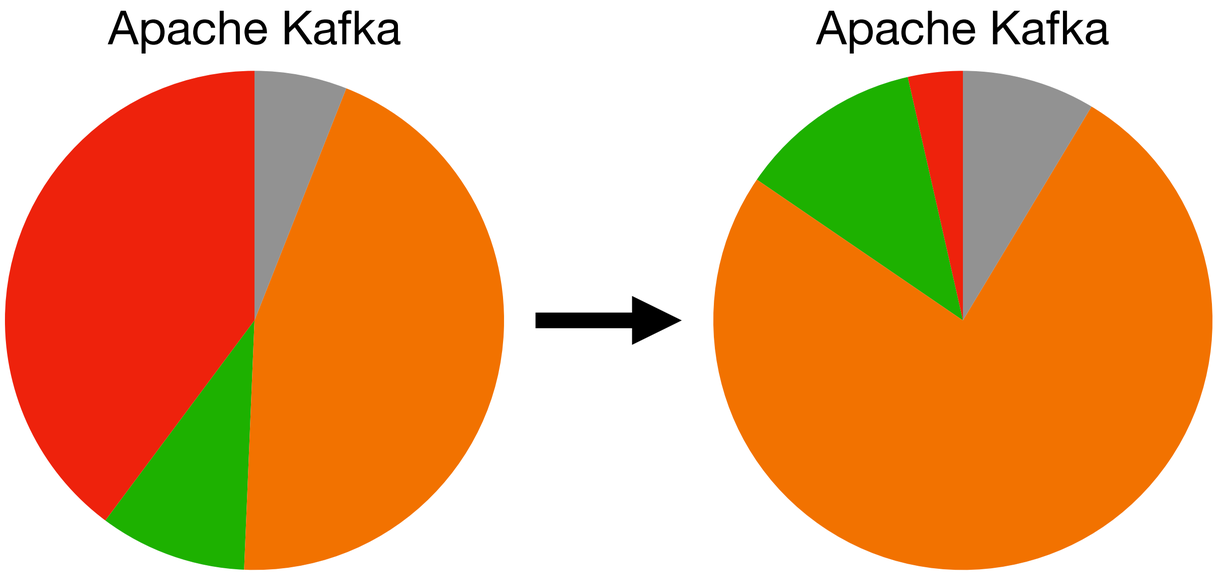


Improvements

Experiment

Not found methods

Application	Was, %	Became, %
Tomcat	14	1,37
Derby	2	1,9
Spring	24	13,87
Kafka	40	3,55
Takes	7	3,33

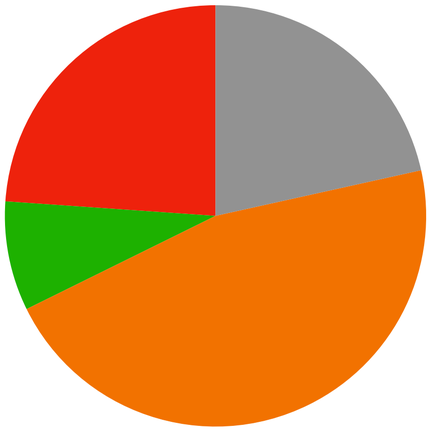


Improvements

Experiment

CSV

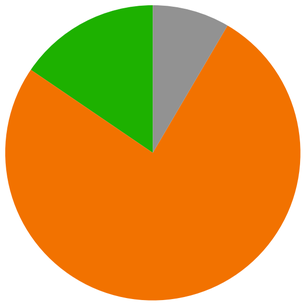
Spring



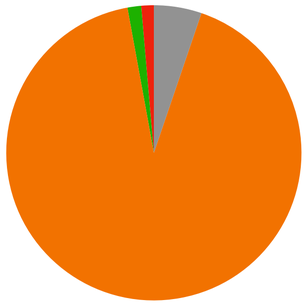
org.springframework
com.fasterxml.jackson.databind
com.fasterxml.jackson.core
org.apache.tomcat
org.apache.catalina
org.apache.coyote

CSV

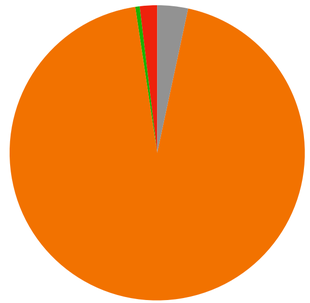
Jackson Core 2.13.5



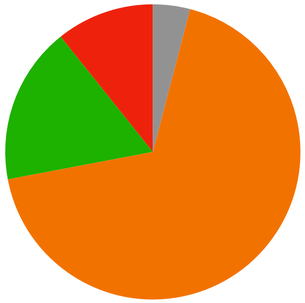
Apache Tomcat 10.1.8



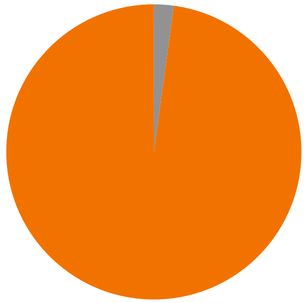
Apache Catalina 10.1.8



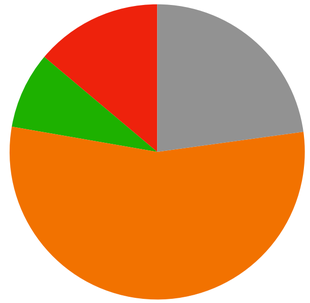
Jackson Databind 2.13.5



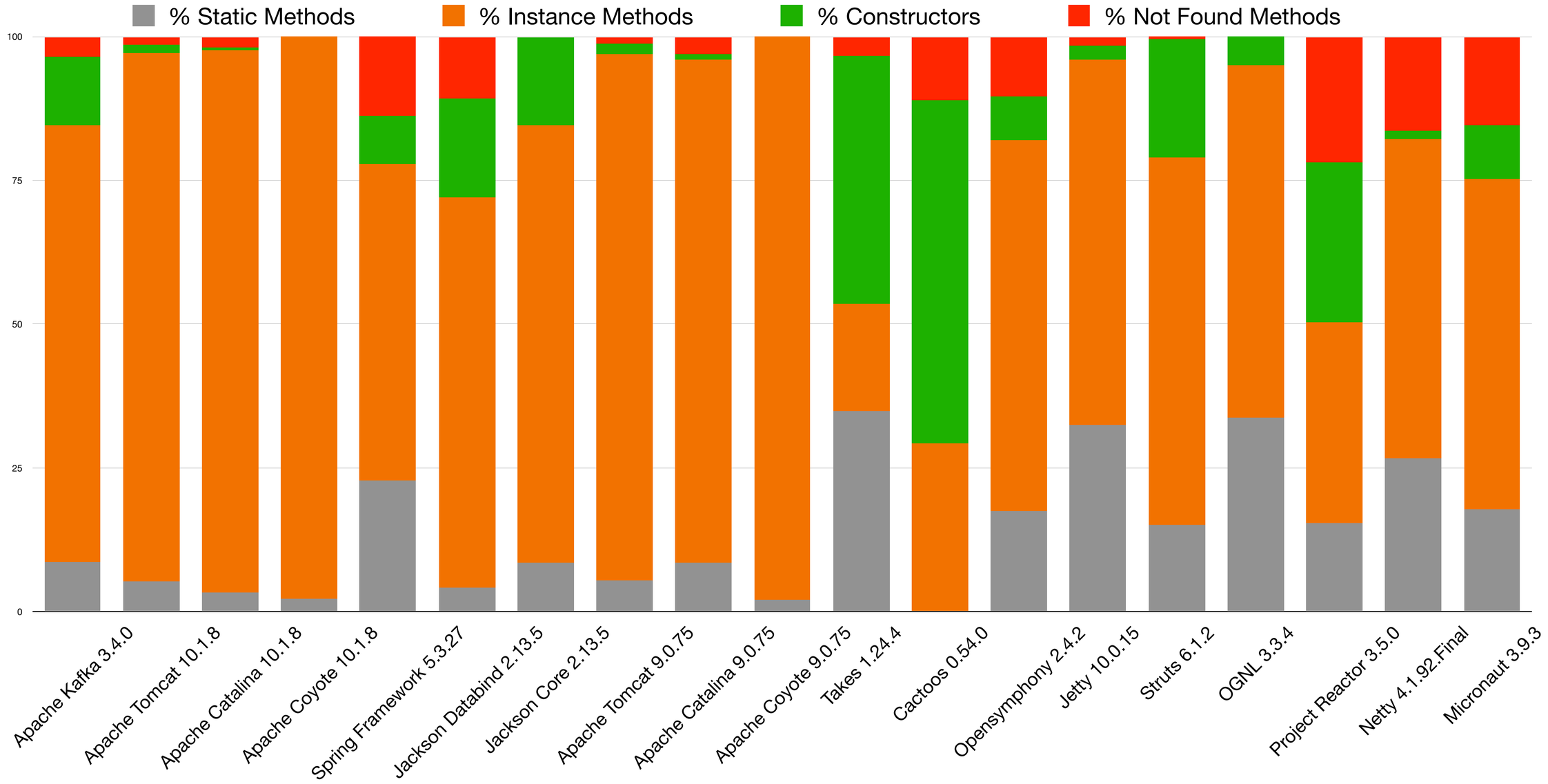
Apache Coyote 10.1.8



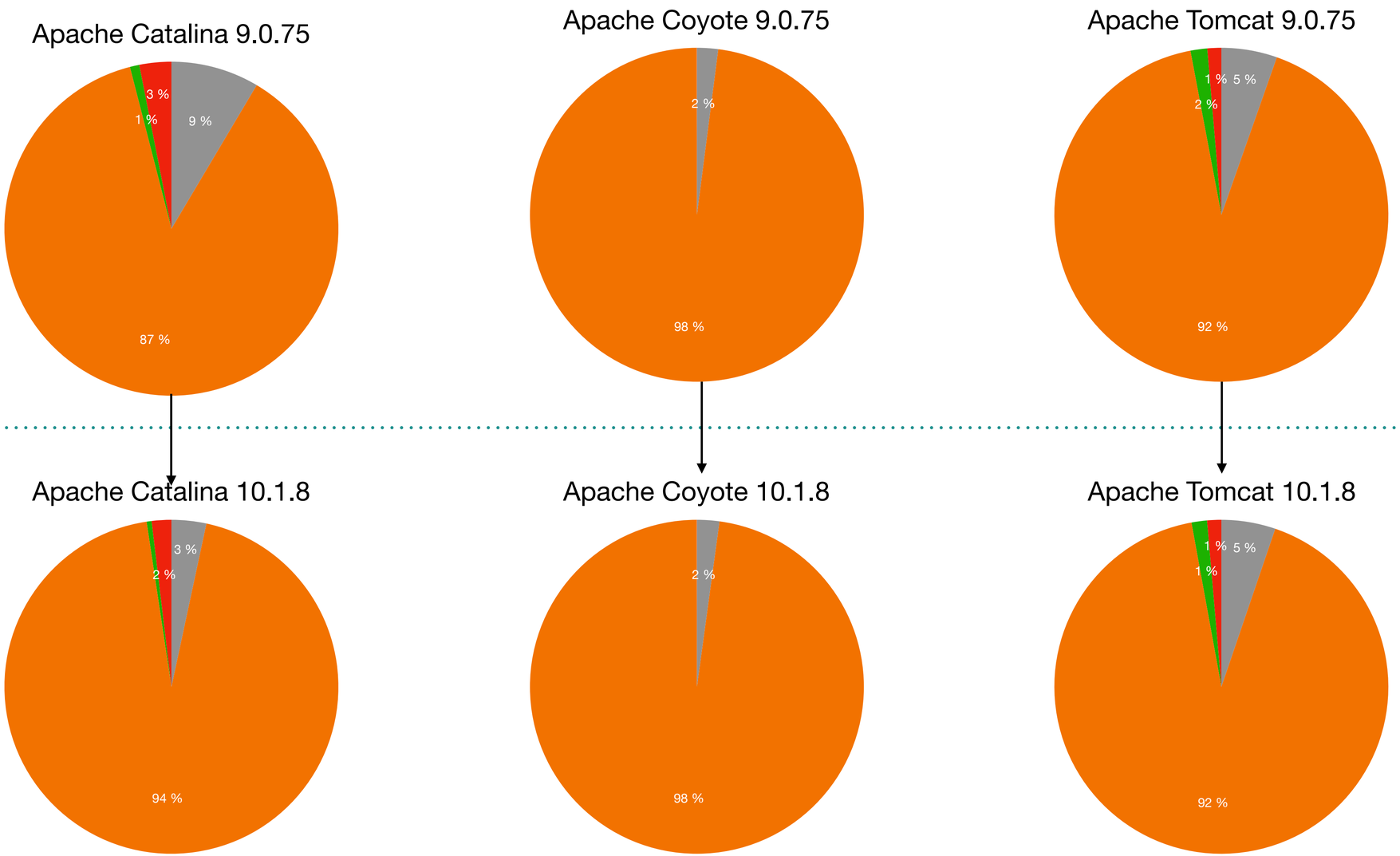
Spring Framework 5.3.27



Results


% Static Methods
% Instance Methods
% Constructors
% Not Found Methods
100
75
50
25
0
Apache Kafka 3.4.0
Apache Tomcat 10.1.8
Apache Catalina 10.1.8
Apache Coyote 10.1.8
Spring Framework 5.3.27
Jackson Databind 2.13.5
Jackson Core 2.13.5
Apache Tomcat 9.0.75
Apache Catalina 9.0.75
Apache Coyote 9.0.75
Takes 1.24.4
Cactoos 0.54.0
Opensymphony 2.4.2
Jetty 10.0.15
Struts 6.1.2
OGNL 3.3.4
Project Reactor 3.5.0
Netty 4.1.92.Final
Micronaut 3.9.3

Application versions



Problems

- Specific - artificial load (It would be better to check applications using real load)
- Small number of profiled applications (Actually, it’s hard to start and profile an application) - unrepresentative statistics
- Profilers don’t provide enough functionality to define the types of methods (constructors, static, instance methods, etc.)

by **Vladimir Kondratyev** » Wed May 31, 2023 11:01 am

Dear Volodya,

I see your point, and I've added corresponded feature requests. But I can't promise any dates when these features will be implemented.

Conclusion

- Instance methods definitely prevail
- Specific trends weren’t found

Future questions

- Do applications have any common dynamics across versions (Tomcat 1, Tomcat 2, Tomcat 9.0.4, Tomcat 10.05, etc.)?
- How to gather more statistics from runtime and how to profile more applications?
- Can we gather statistics from Docker containers or by using plugins?
- Can we use tests to judge about application behaviour in a real runtime?
- We need an open-source tool which would modify byte code and count different statistics more precise without the need to parse sources