Мифы и легенды

~~Древней Греции~~

Java Stream API

Зураб Белый
Рексофт 2016

Дисклеймер

Не верьте мне.

Проверяйте всё сами.

Миф №1: “Stream API только для коллекций”

Stream из коллекций

```
Collection objects = ...;

objects.stream();

objects.parallelStream();
```

Stream из массивов

```
int[] array = new int[0];

Arrays.stream(array);
```

Stream из строк

```
"Тут могла быть ваша реклама".chars();


Pattern
   .compile("^[0-9]+[А-Яа-я]{2}")
   .splitAsStream("Тут тоже могла быть
   ваша реклама");
```

Stream из vararg

```
Stream.of(object);

Stream.of(object1, object2, object3, object4);

Stream.builder().add(object1).add(object2).build()
;
```

Генераторы стримов

```
Stream.iterate(0, i -> i++);

Stream.generate(() -> Math.random() % 123);
```

Stream из потоков и файлов

```
Files.lines("/usr/local/java/tutorial.txt")
;
```

```
Reader reader = ...;
new BufferedReader(reader).lines();
```

```
new JarFile("pokmeonOG.jar").stream();
```

Stream повсюду...

```
new Random().ints();
```

```
new BitSet().stream();
```

Миф №1 “Contains” только для коллекций

Тестовый стенд

```
@State(Scope.Benchmark)
@BenchmarkMode(Mode.AverageTime)
@OutputTimeUnit(TimeUnit.MICROSECONDS)
@Warmup(iterations = 20)
@Measurement(iterations = 30)
@OperationsPerInvocation(1000)
@Fork(10)
```

Миф №2: “Stream быстрее цикла”

```java
@Benchmark
public List<String> loop(List<String> strings) {
    List<String> newStrings = new ArrayList<>();
    for (String str : strings) {
        if (str.indexOf("1") > 3) {
          newStrings.add(
            str.toUpperCase().substring(0, 3)
          );
        }
    }
    return newStrings;
}
```

```java
@Benchmark
public List<String> stream(List<String> strings) {
    return strings
            .stream()
            .filter(str -> str.indexOf("1") > 3)
            .map(str ->
                 str.toUpperCase().substring(0, 3))
            .collect(Collectors.toList());
}
```

Benchmark	Items	Samples	Score	Score error	Units
loop	100 000	300	**10,376**	0,063	µs/op
stream	100 000	300	**10,582**	0,074	µs/op

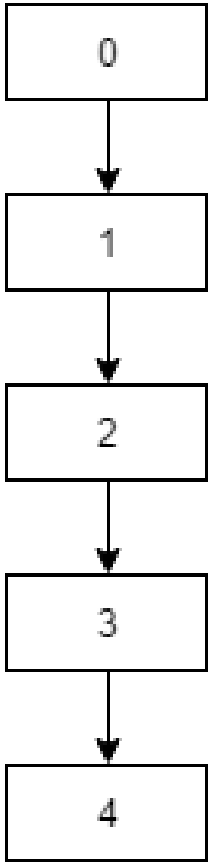
```
@Benchmark
public List<String> stream(List<String> strings)
{
    return strings
            .stream()
            .parallel()
            .filter(str -> str.indexOf("1") > 3)
            .map(str ->
                 str.toUpperCase().substring(0,
3))
            .collect(Collectors.toList());
}
```

Benchmark	Items	Samples	Score	Score error	Units
loop	100 000	300	10,376	0,063	µs/op
stream	100 000	300	10,582	0,074	µs/op
parallel	**100 000**	**300**	3,423	0,097	**µs/op**

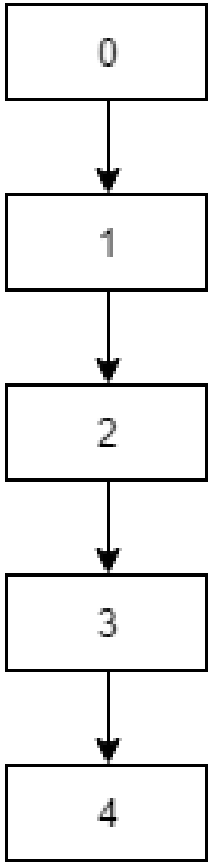
Миф №2: “[illegible]ты [illegible]е цикла”

Миф №3: “Оптимизировать код можно добавлением .parallel()”

java.util.Iterator<E>

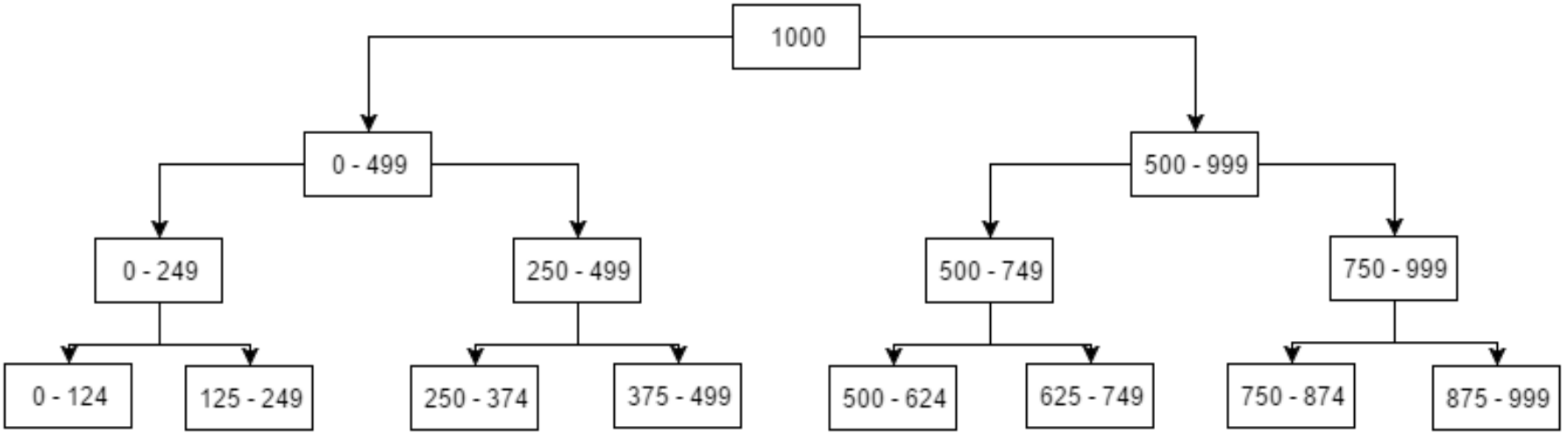


java.util.Iterator<E>



java.util.Spliterator<T>

java.util.Spliterator<T>



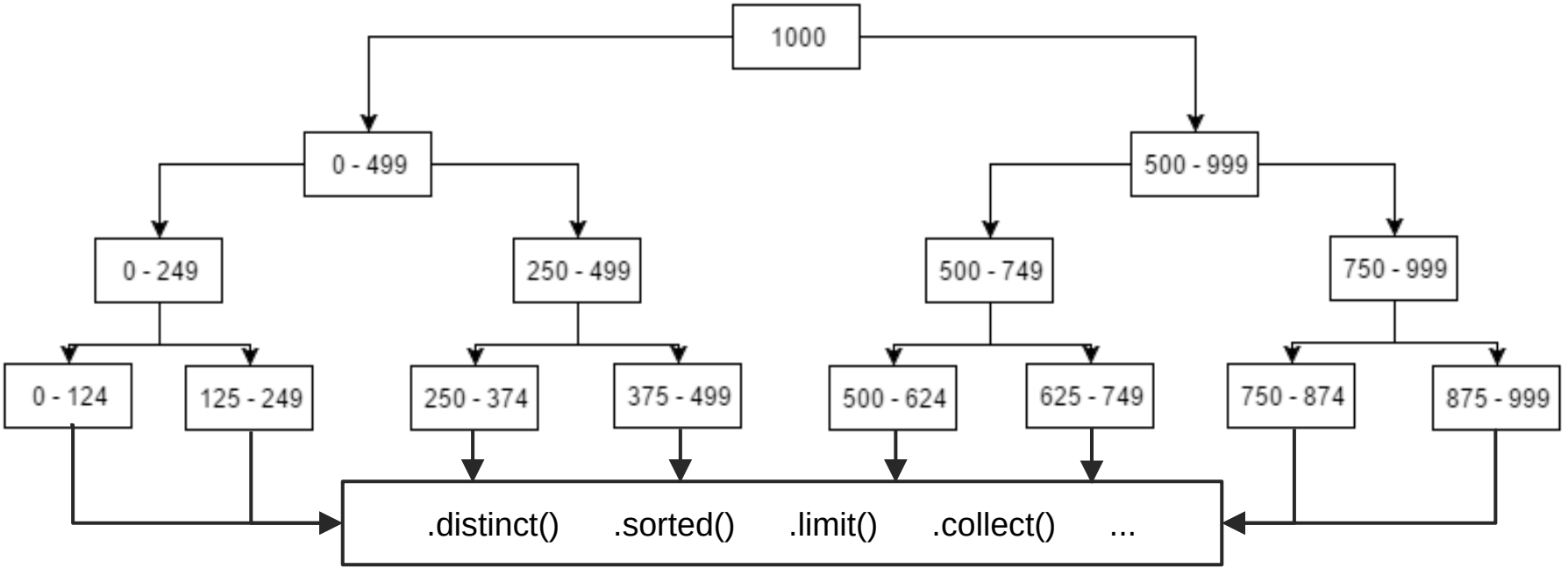
java.util.Spliterator<T>

```java
@Benchmark
public List<String> top10Users(List<User> users) {
    return users
            .stream()
            // .parallel()
            .distinct()
            .sorted()
            .limit(10)
            .map(Objects::toString)
            .collect(Collectors.toList());
}
```

```
public class User implements Comparable {
    private long id, count;
    ...
    public boolean equals(Object o) {
        if (this == o) return true;
      else if (!(o instanceof User)) return false;
      else return id == ((User) o).id;
    }
    public int compareTo(Object o) {
        long count2 = ((User) o).getCount();
      if (count > count2) return 1;
      else if (count == count2) return 0;
      else return -1
    }
}
```

Benchmark	Items	Samples	Score	Score error	Units
simple	100 000	300	**82,199**	0,728	us/op
parallel	100 000	300	53,882	0,294	us/op

java.util.Spliterator<T>



Benchmark	Items	Samples	Score	Score error	Units
simple	100 000	300	82,199	0,728	us/op
parallel	100 000	300	53,882	0,294	us/op
simple	**100**	300	**0,033**	0,000	us/op
parallel	**100**	300	**0,105**	0,002	us/op

Benchmark	Items	Samples	Score	Score error	Units
simple	100 000	300	**82,199**	0,728	us/op
parallel	100 000	300	53,882	0,294	us/op
simple	**100**	300	**0,033**	0,000	us/op
parallel	**100**	300	**0,105**	0,002	us/op
simple	**10m**	300	**15731,075**	337,593	us/op
parallel	**10m**	300	**17065,969**	596,584	us/op

Миф №3: “Оптимизировать код можно добавлением .parallel()”

Миф №4: “Класс источника стрима прямо не влияет на скорость”

java.util.Spliterator<T>. Характеристики

ORDERED

DISTINCT

SORTED

SIZED

NONNULL

IMMUTABLE

CONCURRENT

SUBSIZED

java.util.Spliterator<T>. Характеристики

ORDERED	NONNULL
DISTINCT	IMMUTABLE
SORTED	CONCURRENT
SIZED	SUBSIZED

java.util.Spliterator<T>. Характеристики

ORDERED	NONNULL
DISTINCT	IMMUTABLE
SORTED	CONCURRENT
SIZED	SUBSIZED

java.util.Spliterator<T>. Характеристики

ORDERED	NONNULL
DISTINCT	IMMUTABLE
SORTED	CONCURRENT
SIZED	SUBSIZED

java.util.Spliterator<T>. Характеристики

ORDERED	NONNULL
DISTINCT	IMMUTABLE
SORTED	CONCURRENT
SIZED	SUBSIZED

java.util.Spliterator<T>. Характеристики

ORDERED

DISTINCT

SORTED

SIZED

NONNULL

IMMUTABLE

CONCURRENT

SUBSIZED

java.util.Spliterator<T>. Характеристики

ORDERED	NONNULL
DISTINCT	**IMMUTABLE**
SORTED	CONCURRENT
SIZED	SUBSIZED

java.util.Spliterator<T>. Характеристики

ORDERED	NONNULL
DISTINCT	IMMUTABLE
SORTED	**CONCURRENT**
SIZED	SUBSIZED

java.util.Spliterator<T>. Характеристики

ORDERED	NONNULL
DISTINCT	IMMUTABLE
SORTED	CONCURRENT
SIZED	**SUBSIZED**


ЗАЧЕМ ВСЕ ЭТО НУЖНО?
КАКИЕ ЕЩЕ ХАРАКТЕРИСТИКИ?
risovach.ru

Benchmark	Items	Samples	Score	Score error	Units
simple	100 000	300	**82,199**	0,728	us/op
parallel	100 000	300	**53,882**	0,294	us/op

Benchmark	Items	Samples	Score	Score error	Units
simple	100 000	300	**0,005**	0,000	us/op
parallel	100 000	300	**0,850**	0,009	us/op

Миф №4: “Класс итератор стрима прямо влияет на скорость”

Миф №5: “Параллельный стрим выполняет лишние операции”

```
Stream
  .of(1, 2, 3, 4, 5, 6)
  .map(i -> {
      System.out.println("Map " + i);
      return i;
  })
  .filter(i -> {
      System.out.println("Filter " + i);
      return true;
  })
  .limit(3)
  .forEach(i -> System.out.println("ForEach " +
i));
```

Обычный стрим

Map 1
Filter 1
ForEach 1
Map 2
Filter 2
ForEach 2
Map 3
Filter 3
ForEach 3

Обычный стрим

Map 1
Filter 1
ForEach 1
Map 2
Filter 2
ForEach 2
Map 3
Filter 3
ForEach 3

Параллельный стрим

Map 1
Map 3
Map 4
Map 2
Filter 4
Filter 3
Filter 1
Filter 2
Map 5
Map 6

Filter 5
Filter 6
ForEach 1
ForEach 3
ForEach 2

Миф №5: “Параллельный стрим выполняет лишние операции”

Итог

Миф №1: “Stream API только для коллекций”

Миф №2: “Stream быстрее цикла”

Миф №3: “Оптимизировать код можно добавлением .parallel()”

Миф №4: “Класс источника стрима прямо не влияет на скорость”

Миф №5: “Параллельный стрим выполняет лишние операции”

Итог

Миф №1: “Stream API только для коллекций”

Миф №2: “Stream быстрее цикла”

Миф №3: “Оптимизировать код можно добавлением .parallel()”

Миф №4: “Класс источника стрима прямо не влияет на скорость”

Миф №5: “Параллельный стрим выполняет лишние операции”

Итог

: “Stream API только для коллекций”

: “Stream быстрее цикла”

Миф №3: “Оптимизировать код можно добавлением .parallel()”

Миф №4: “Класс источника стрима прямо не влияет на скорость”

Миф №5: “Параллельный стрим выполняет лишние операции”

Итог

: “Stream API только для коллекций”

: “Stream быстрее цикла”

: “Оптимизировать код можно добавлением .parallel()”

Миф №4: “Класс источника стрима прямо не влияет на скорость”

Миф №5: “Параллельный стрим выполняет лишние операции”

Итог

: “Stream API только для коллекций”

: “Stream быстрее цикла”

: “Оптимизировать код можно добавлением .parallel()”

: “Класс источника стрима прямо не влияет на скорость”

Миф №5: “Параллельный стрим выполняет лишние операции”

Итог

: “Stream API только для коллекций”

: “Stream быстрее цикла”

: “Оптимизировать код можно добавлением .parallel()”

: “Класс источника стрима прямо не влияет на скорость”

: “Параллельный стрим выполняет лишние операции”

Спасибо за внимание!

Вопросы?

@ZurabBelyi

BelyiZ