Choosing the Right Sources of Data to Efficiently Go Parallel

José Paumard
PHD, JAVA CHAMPION, JAVA ROCK STAR
@JosePaumard https://github.com/JosePaumard

Agenda

You saw how parallelism is implemented

How synchronization interacts with parallelism

How does the splitting work with sources of data?

Splitting a Source of Data

A stream can be created on many sources

Array, ArrayList

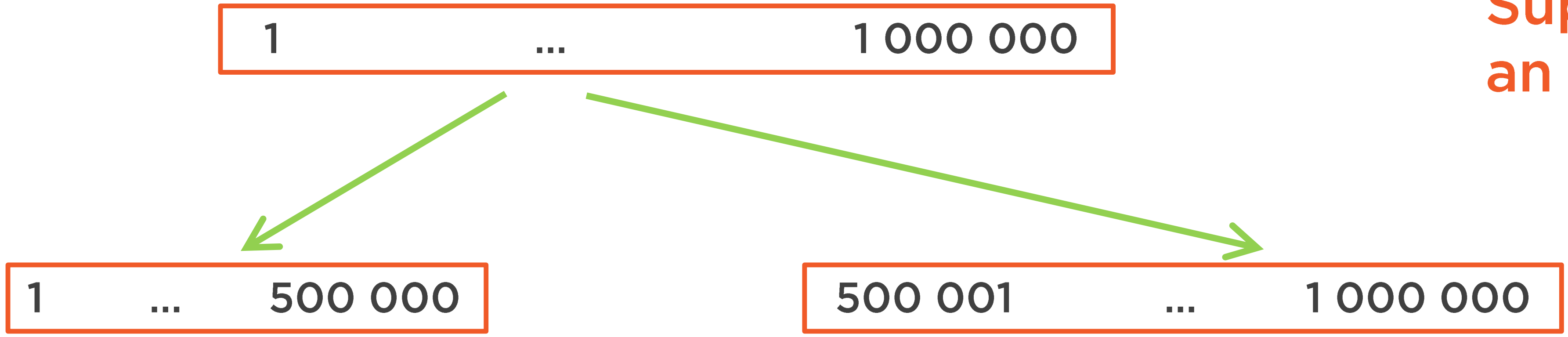
Linked lists

Sets, HashSet

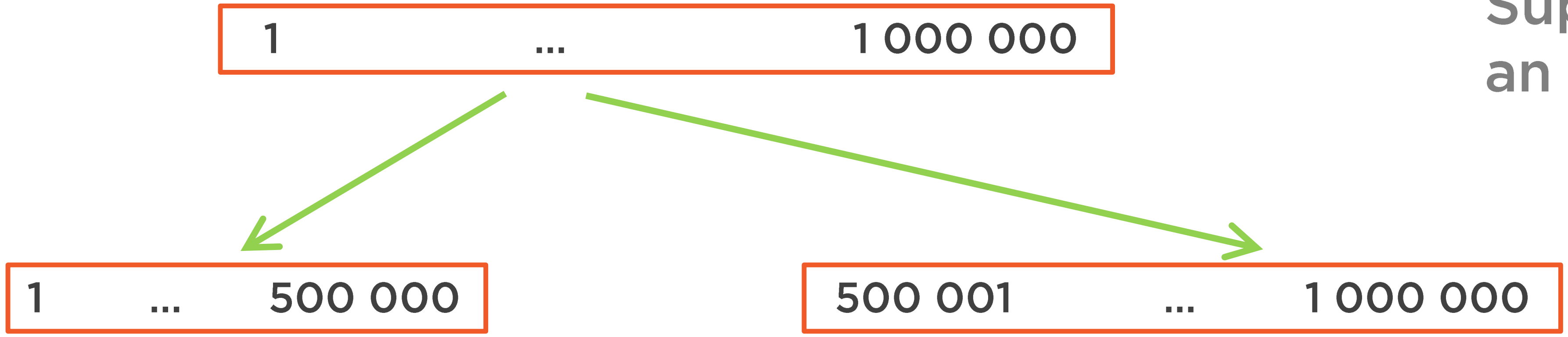
Iterator, lines of a text file

Words of a sentence

And many other



Suppose we have an array of 1M int

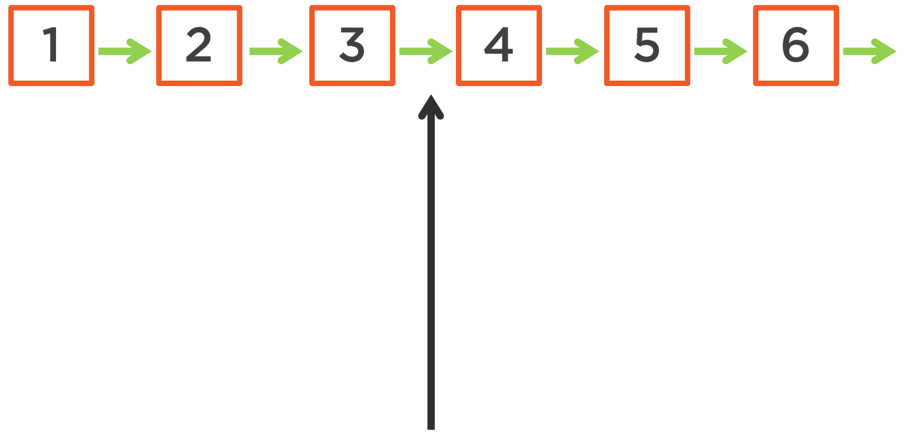


Suppose we have an array of 1M int

It is easy and unexpensive to split an array in two arrays of the same size

Reaching the center of the data must be easy, reliable and efficient

The number of elements is known before processing them



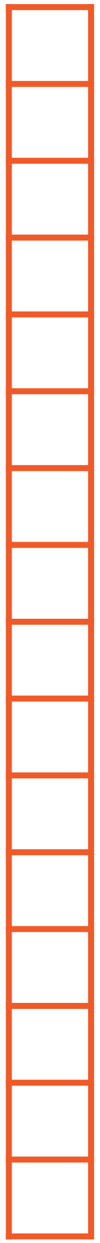
Suppose we have
a linked list of 1M int

It is easy to split a linked list
But costly to reach the element at the center

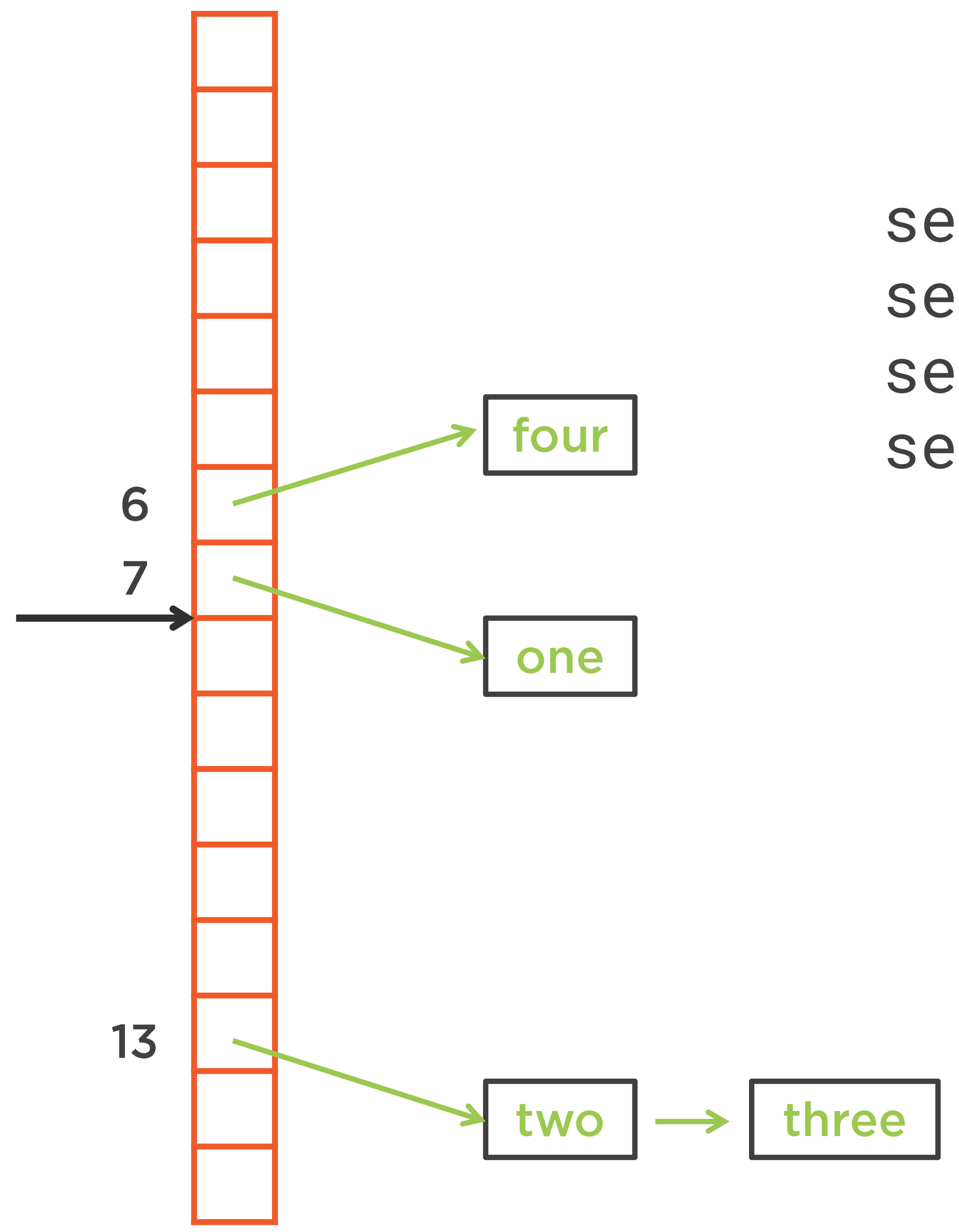
What about Set?

A Set is implemented by HashSet, backed by a HashMap

Built on an array

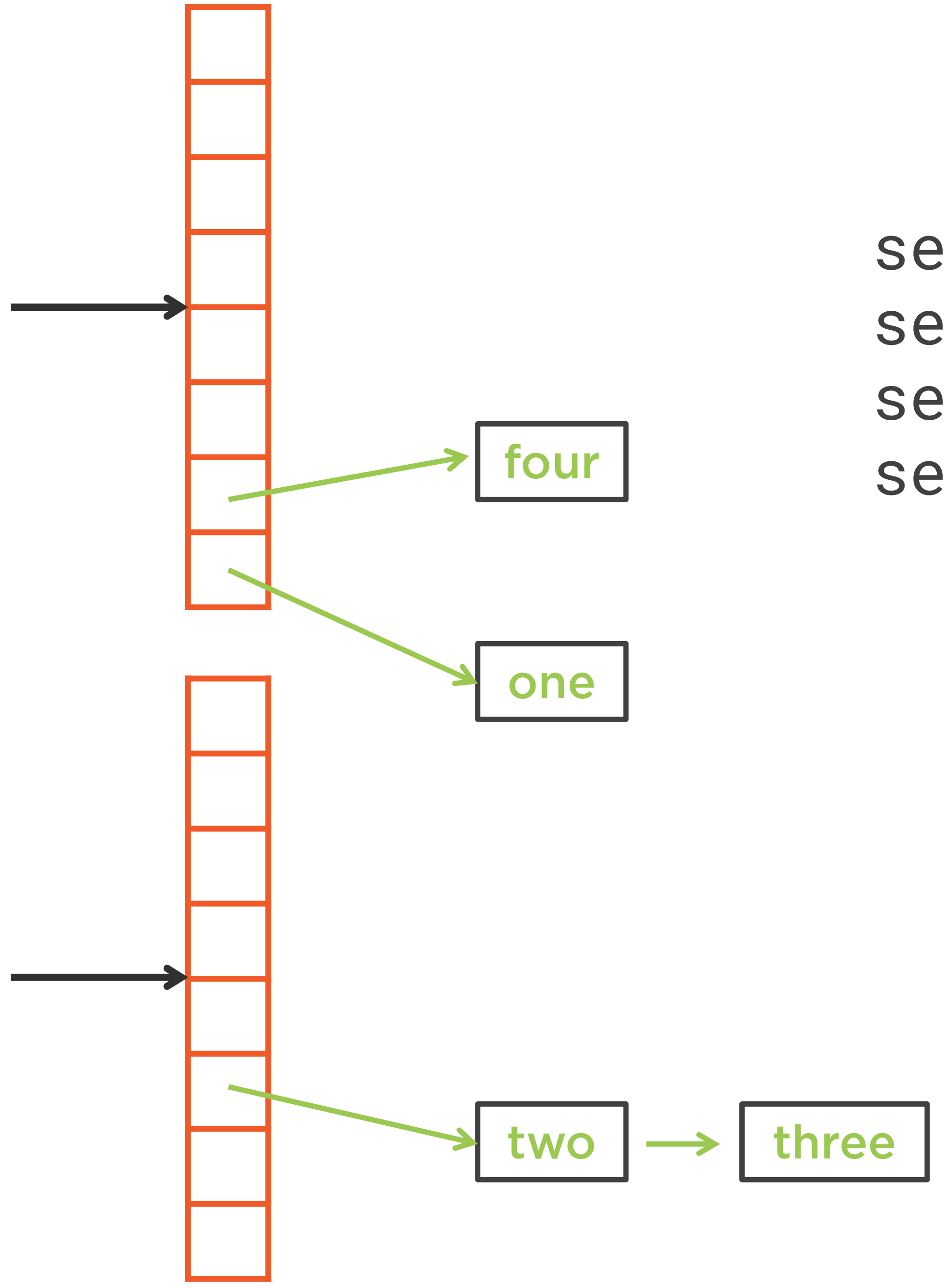


A set is backed by a HashMap
Built on an array



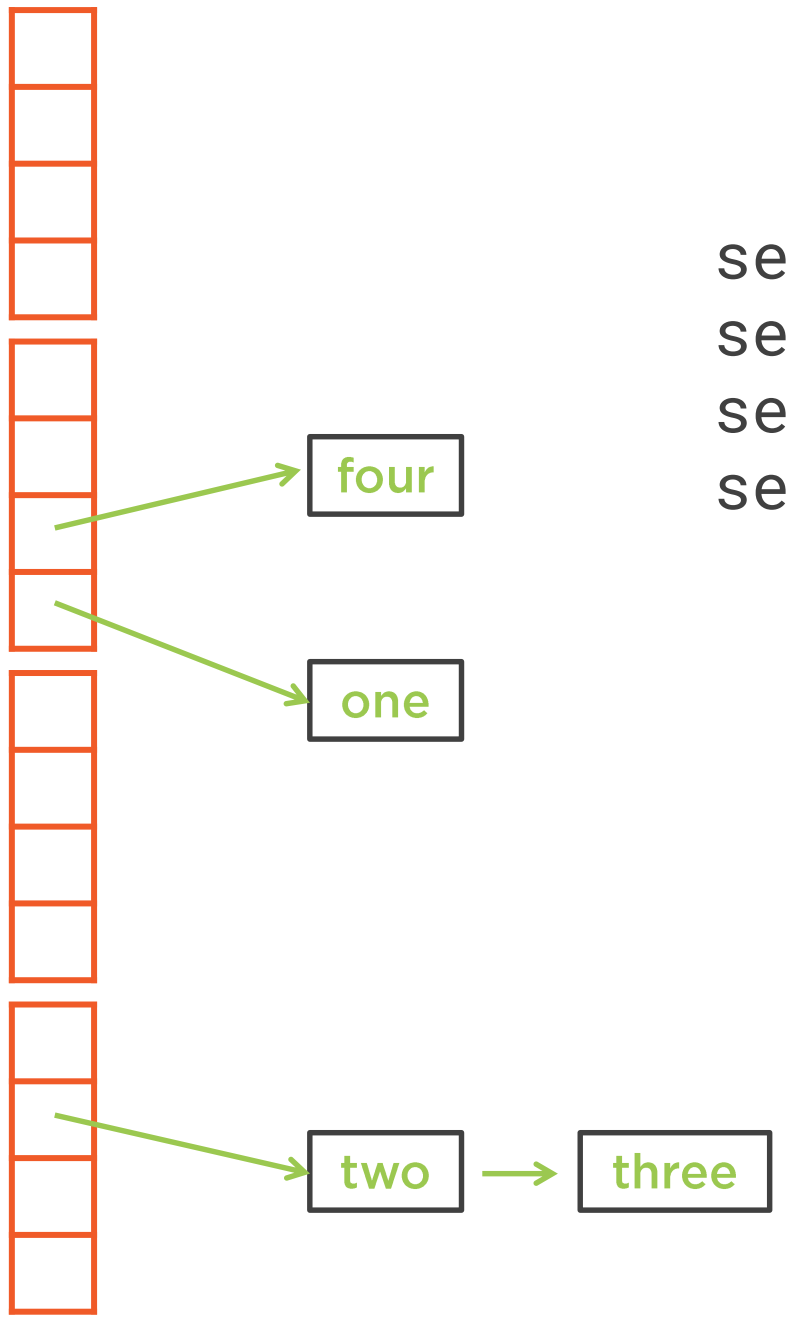
```
set.add("one");
set.add("two");
set.add("three");
set.add("four");
```

Splitting a set is easy
Reaching the center is inexpensive
But impossible to know if both halves are equal



```
set.add("one");
set.add("two");
set.add("three");
set.add("four");
```

Splitting a set is easy
Reaching the center is inexpensive
But impossible to know if both halves are equal



```
set.add("one");
set.add("two");
set.add("three");
set.add("four");
```

Splitting a set is easy
Reaching the center is inexpensive
But impossible to know if both halves are equal

SIZED =
the number of elements
of the source is known

SUBSIZED =
the number of elements
of the two split sources is known

What about Iterator?

A Stream can be created on an Iterator

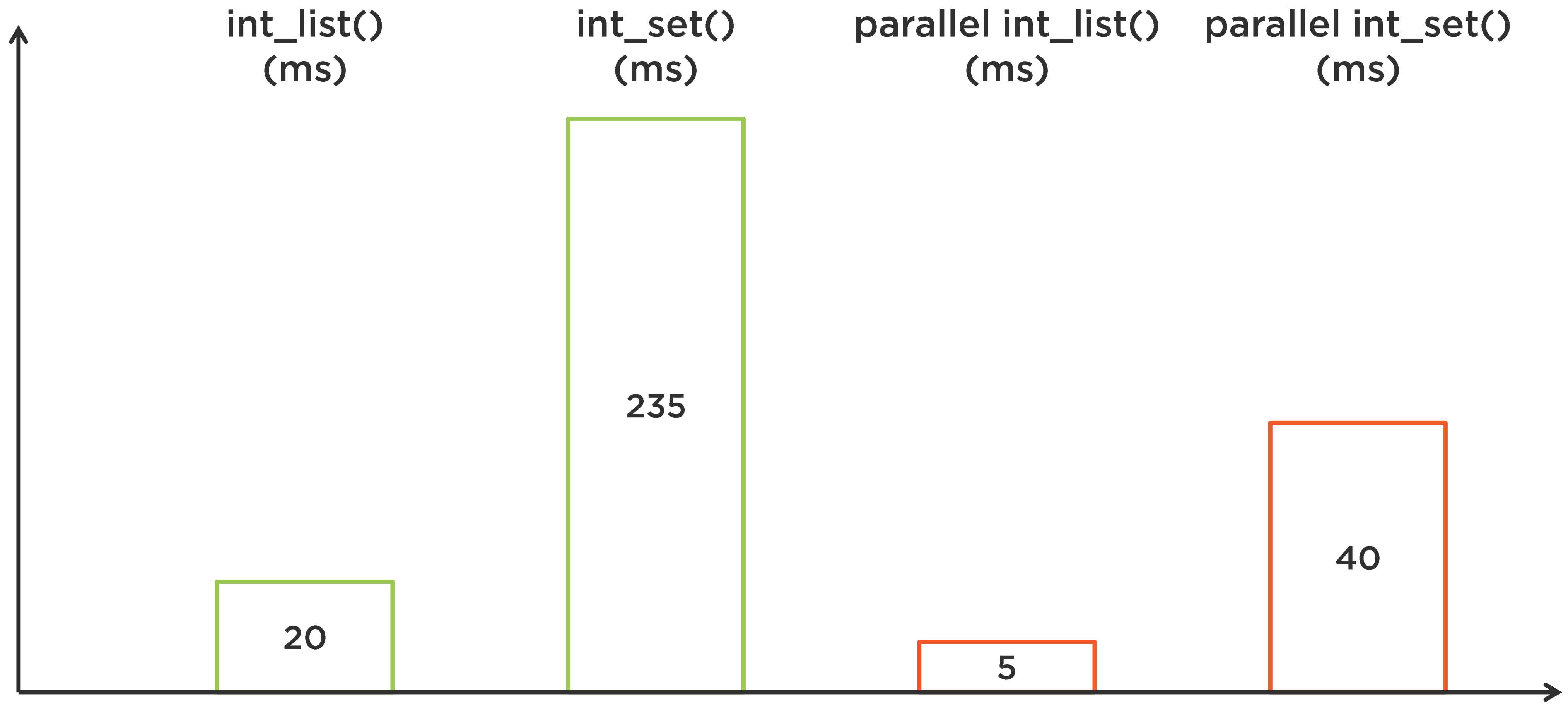
But the number of elements is unknown

Demo

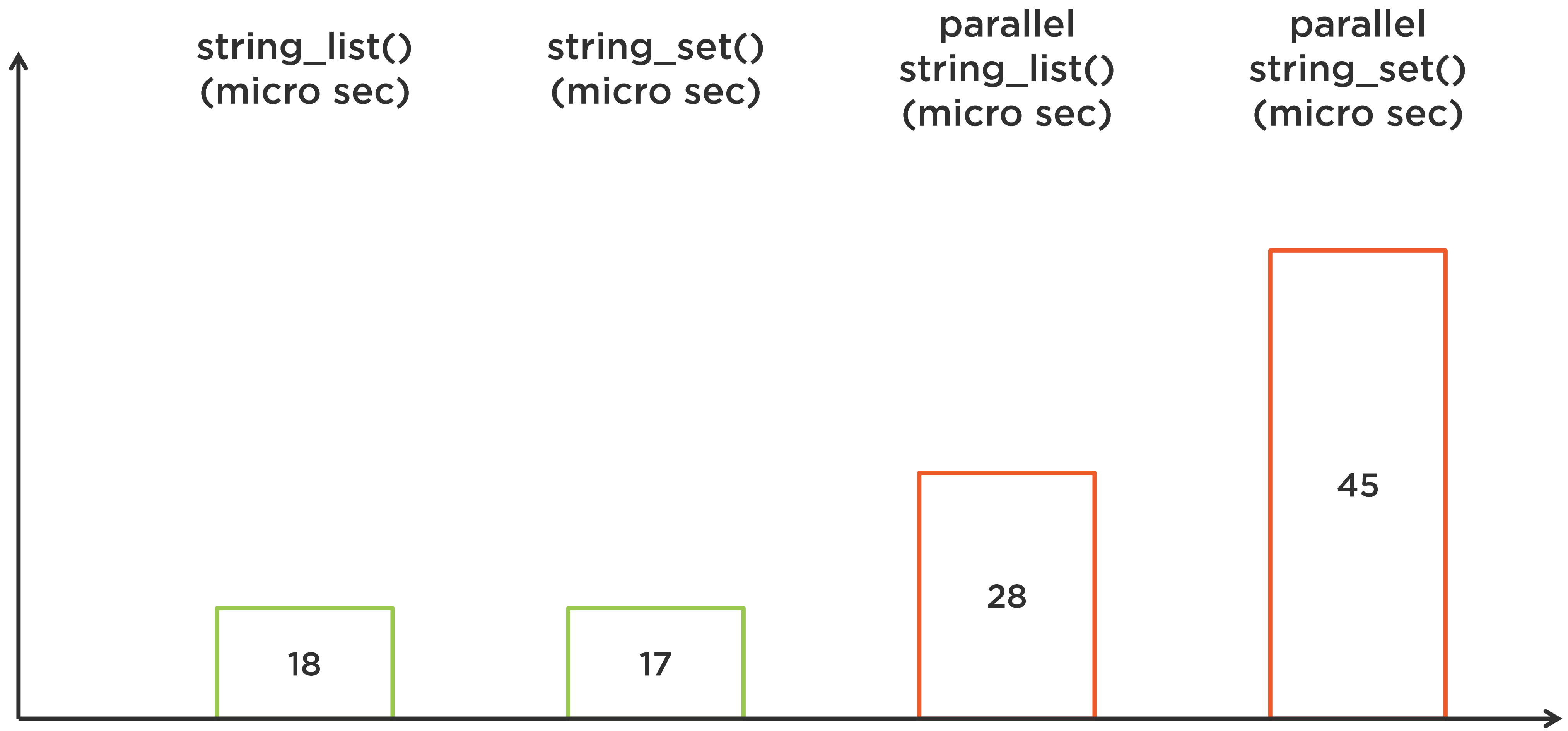
Let us write some code!

And see how splitting works with these sources

And result is…



And result is...



And result is...

```
Benchmark                                        (N)  Mode  Cnt       Score       Error  Units
M05_SourceSplit.process_int_list            10000000  avgt   15   19778,041 ±   261,286  us/op
M05_SourceSplit.process_int_set             10000000  avgt   15  235401,822 ±  1374,660  us/op

M05_SourceSplit.process_string_list         10000000  avgt   15      18,305 ±     1,851  us/op
M05_SourceSplit.process_string_list_parallel 10000000  avgt   15      28,386 ±     0,269  us/op

M05_SourceSplit.process_string_set          10000000  avgt   15      17,933 ±     0,422  us/op
M05_SourceSplit.process_string_set_parallel 10000000  avgt   15      45,966 ±     2,746  us/op


M05_SourceSplit.process_int_list_parallel   10000000  avgt   15    5587,108 ±    54,324  us/op
M05_SourceSplit.process_int_set_parallel    10000000  avgt   15   40335,790 ±   516,616  us/op
```

Do not go parallel on the wrong source

Do not use parallel streams on sources of unknown size

Prefer Lists over Sets

Make sure your source is SIZED and SUBSIZED

Module Wrap Up

What did you learn?

How to choose a source of data:

- it should be cache friendly

- cache friendly is not always parallel friendly

- it should be sized

- it should be easy to split it

- it should be subsized

Course Wrap Up

What did you learn?

When and when not to use parallel streams

Fork / Join framework

Parallel unfriendliness!

- hidden inter-thread communication

- faulty reduction

- hard to split source

Are you sure that your threads
should be used
to compute your streams in parallel?

Course Wrap Up

Thank you!

@JosePaumard

https://github.com/JosePaumard

https://www.youtube.com/user/jpaumard