

Akka Stream

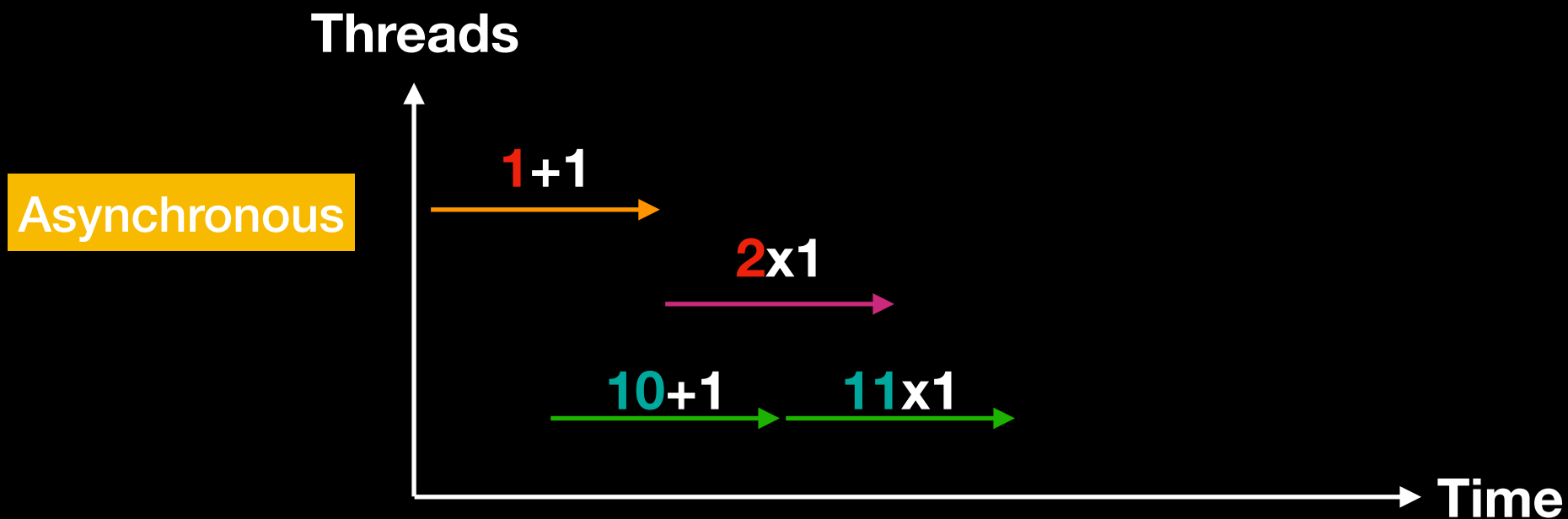
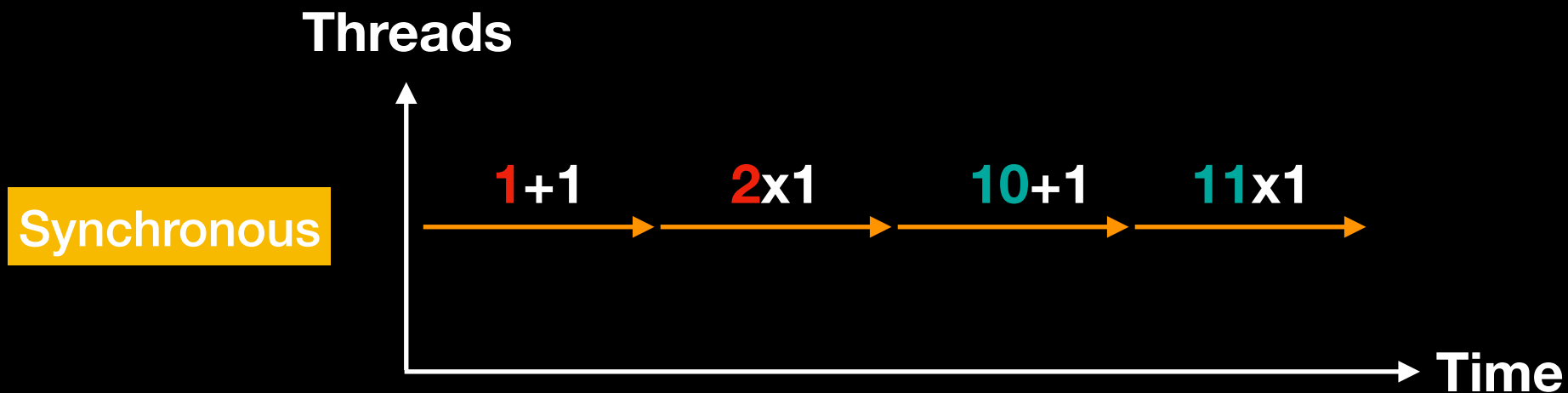
Andy Huang

Reactive Streams

- **Stream:** An active process that involves moving and transforming data.
- **Reactive Streams:**
 1. Asynchronous stream processing
 2. non-blocking back pressure

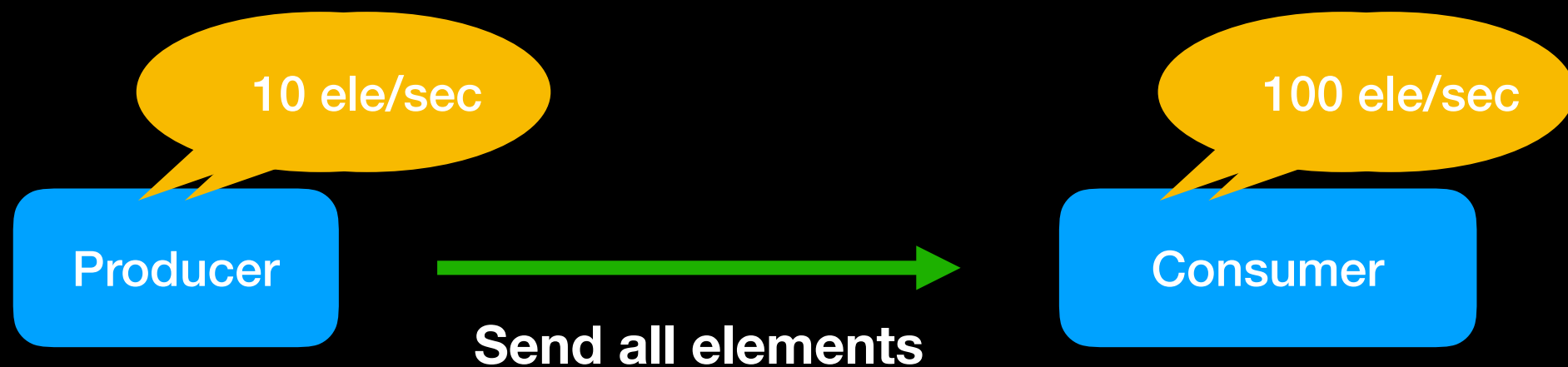
Asynchronous Stream processing

Source : **function: (element + 1) * 1**
1, 10,...



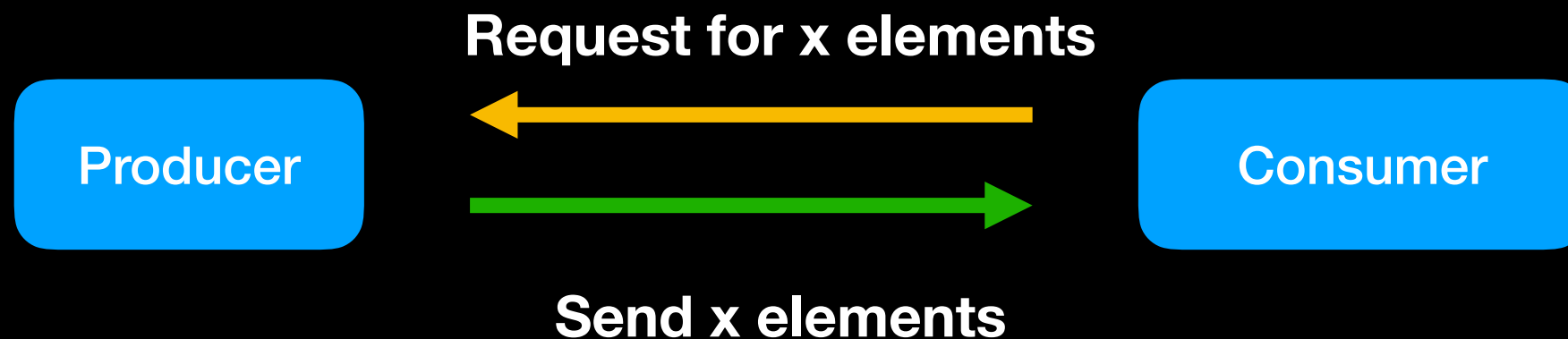
Asynchronous Stream processing

Fast producer, slow Consumer
Slow producer, fast Consumer



Back Pressure

- ★ The benefits of asynchronous processing would be **negated** if the communication of back pressure were synchronous



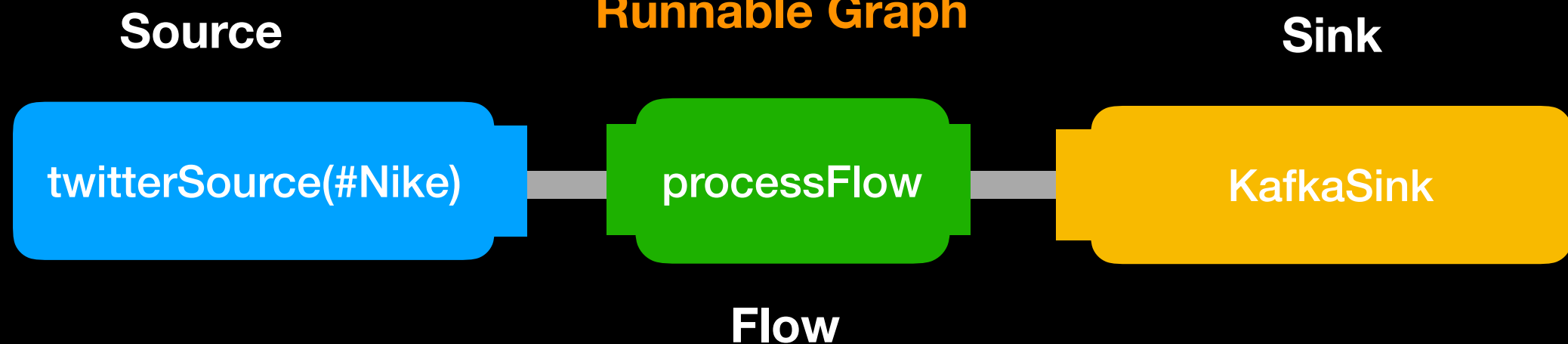
Why Akka Streams

- Without involving blocking
- Avoiding `OutOfMemoryException` e.g. chunked file transfer
- Supporting error recovering
- Doing pipeline processing on a high level

Akka Streams

A Flow that has both ends “attached” to a Source and Sink respectively
A processing stage with *exactly one output*

A processing stage with *exactly one input*
Runnable Graph

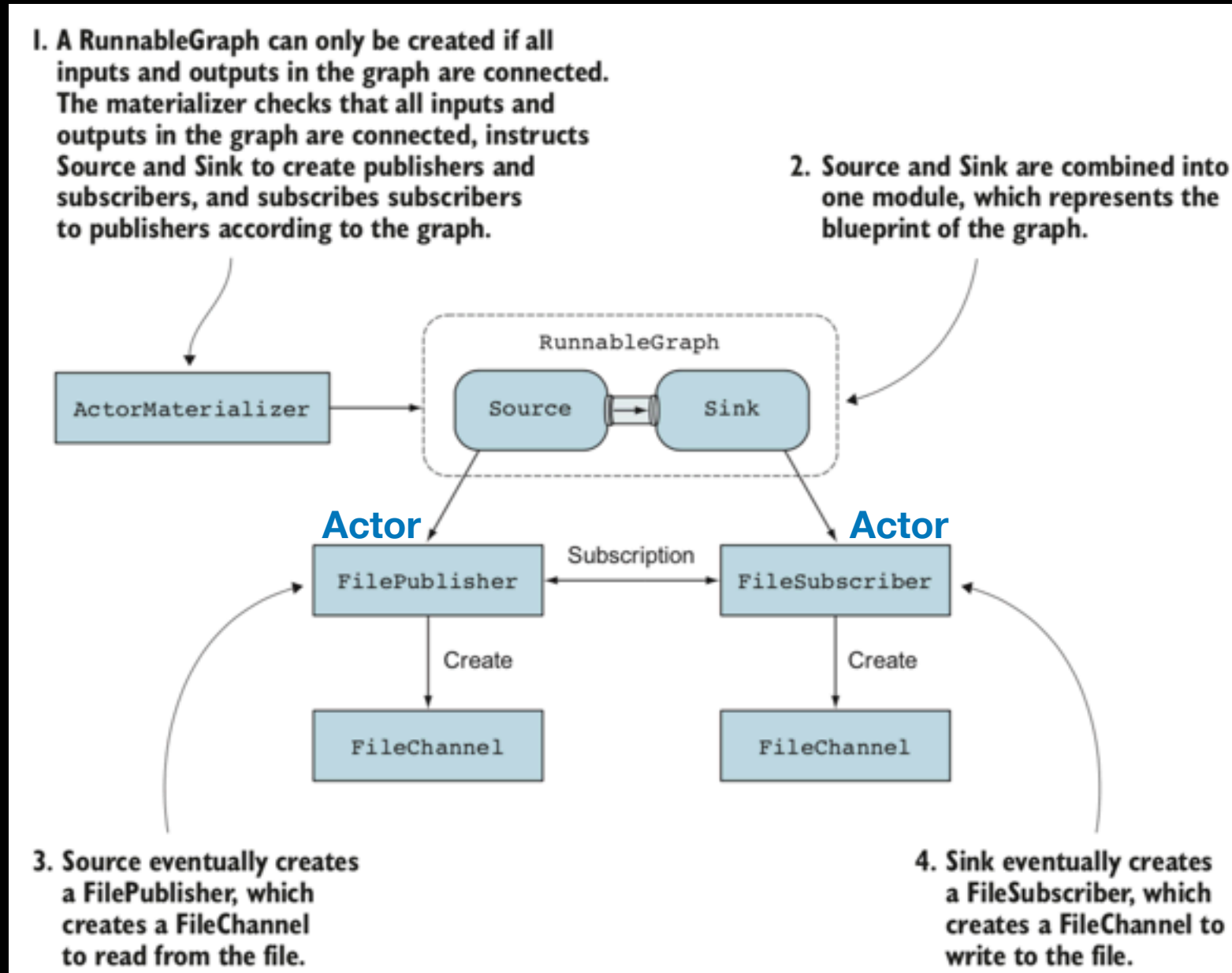


A processing stage which has *exactly one input and output*

Akka Streams

- An **ActorMaterializer** converts the **RunnableGraph** into actors, which execute the graph.
- **Stream Materialization** is the process of taking the stream description and allocating all the necessary resources it need in order to run.

Akka Streams



Refer to **AKKA IN ACTION**

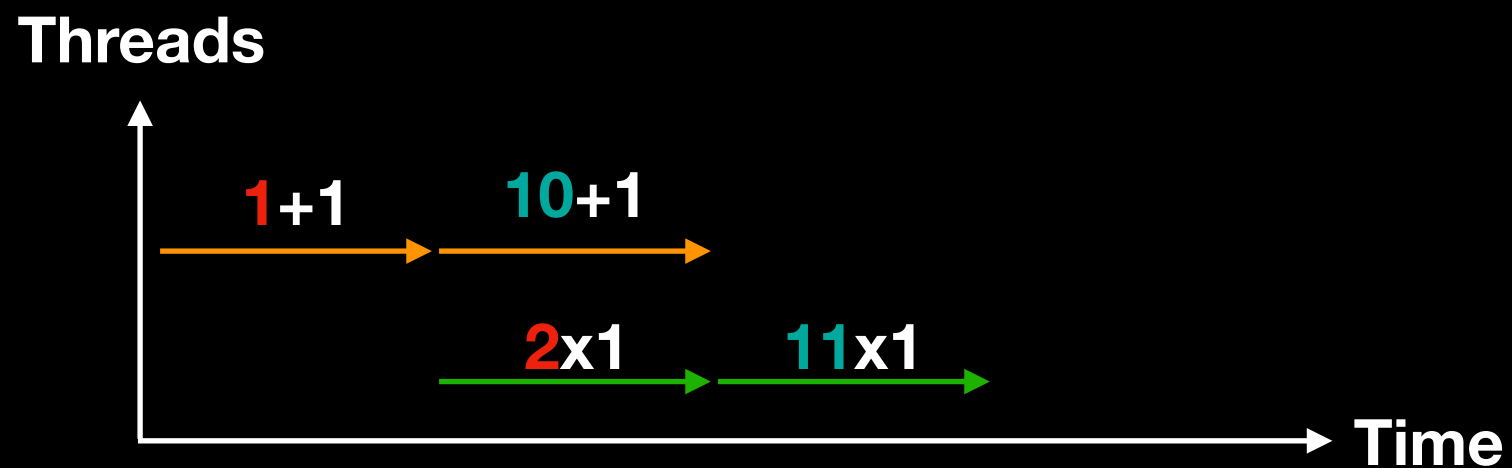
Akka Streams

- Stream stages usually share the same actor(**Operator fusion**) unless they are explicitly demarcated from each other by an asynchronous boundary
- How to process stream asynchronously?
 1. Async Boundary
 2. `mapAsync` or `map(ele => Future { ... })`

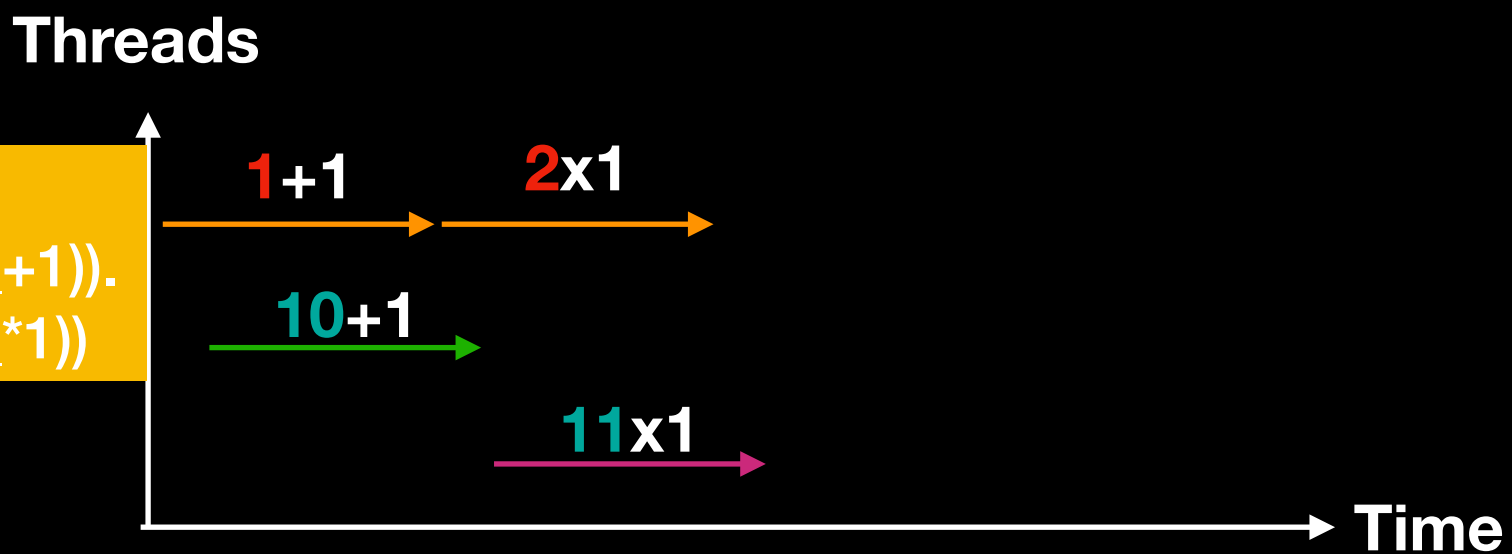
```
Source(List(1,10)).  
map(_+1).map(_*1)
```



```
Source(List(1,10)).  
map(_+1).async.  
map(_*1)
```



```
Source(List(1,10)).  
mapAsync(2)(Future(_+1)).  
mapAsync(2)(Future(_*1))
```



Akka Streams

- **Materialized Value:**
 - a network of running processing entities, inaccessible from the outside
 - a materialized value, optionally providing a controlled interaction capability with the network



Everything is a stream

References

- <https://akka.io/blog/2016/07/06/threading-and-concurrency-in-akka-streams-explained>
- <https://doc.akka.io/docs/akka/2.5.4/scala/stream/stream-flows-and-basics.html>
- <https://www.manning.com/books/akka-in-action>
- <https://info.lightbend.com/ebook-serving-machine-learning-models-register.html>