Intersection Microbenchmarking using Kotlin

Ali Khaleqi Yekta

Me@Yekta.Dev

2021

Test Environment:

- CPU: Intel i7-6700HQ 3.5GHz
- Total RAM: 16GB DDR4
- OS: Arch Linux x86_64
- Kernel: 5.14.3-arch1-1
- JDK Version: 16

Some Q&A:

- What have you done?!

Comparison of the synchronous & asynchronous approaches.

- Why such effort for a simple task?

Mainly Curiosity!

(+showing off some Kotlin)

- What are the results?

They're pretty satisfying...

How it was done?

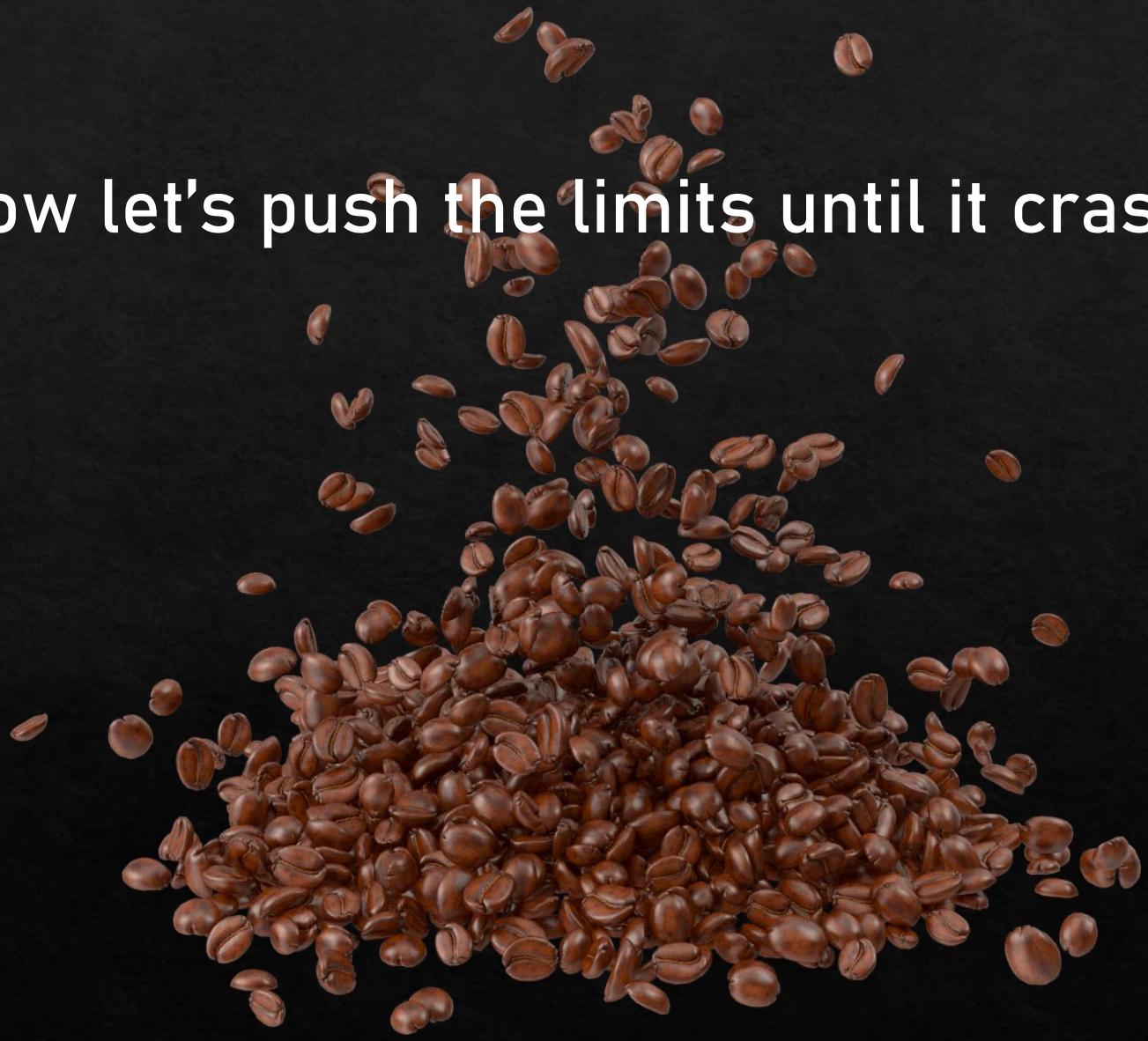
~~List~~

Accessing each & every item
in a HUGE group: HashSet

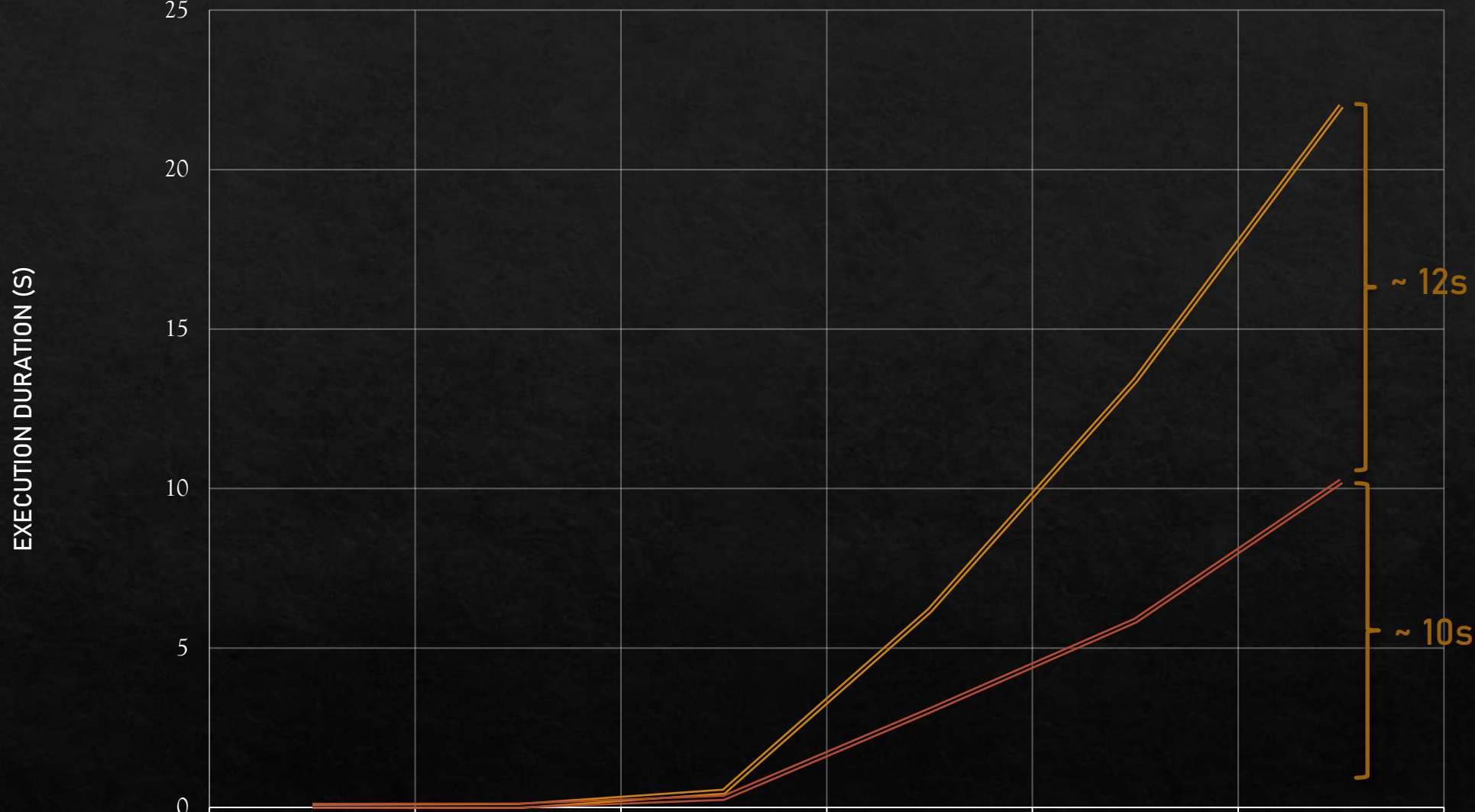
Test results with the specified conditions:

```
Calculating The intersection of two arrays of size 2000000,
filled with random integers in: [1,20000000]...
Using the "nice" algorithm (the desired situation):
Execution time: 0.958 sec
Used memory: ~173873 MB
----------------------------------------
Kotlin version: 1.5.30
JRE version: 16.0.2+7
OS name: Linux
OS version: 5.14.3-arch1-1
OS architecture: amd64
```

Now let's push the limits until it crashes!



INTERSECTION BENCHMARK



Size of array	10,000	100,000	1,000,000	10,000,000	20,000,000	30,000,000
Synchronous	0.016	0.044	0.491	6.173	13.392	21.993
Asynchronous	0.045	0.06	0.305	3.037	5.858	10.228

How was the asynchronous approach implemented?

Using Coroutines

Coroutines are THE well-known approach for async programming in the Kotlin world.

The official Kotlin documentation describes coroutines as:

> “Coroutines can be thought of as light-weight threads, but there is a number of important differences that make their real-life usage very different from threads.”

Since Coroutines are not the topic of our presentation, we don't (and can't) dive into them

The End!

I'd be happy to hear your feedback at:

- Me@Yekta.Dev
- https://yekta.dev/#contact