

JVMS.Compare

Benchmark Performance of Different JDK/JVM combos

Chandra Guntur & Donald Raab

About Us



- Java Champions
- JCP Executive Committee Reps. for BNY Mellon
- Ardent bloggers and tweeters

- Director at BNY Mellon
- Working in Financial Tech. since 2003
- Programming in Java since 1998
- JUG Leader @ NYJavaSIG
- Creator of Java-Katas Github repository

- Managing Director at BNY Mellon
- 18+ years of experience in Financial Tech.
- Programming in Java since 1997
- Member of JSR 335 Expert Group
- Creator of Eclipse Collection Java Library

Session Agenda

Compare benchmarks for operations using
Eclipse Collections and JDK's Java Collection Framework:

- on a primitive `IntList & List<Integer>`
- on a `List<Person>`

using several JDK/JVM combinations

Compared JDK/JVMs

JDK/JVM combinations tested:

1. Oracle JDK 8 (v1.8.0_221)
2. GraalVM Enterprise Edition (v19.2.0)
3. GraalVM Community Edition (v19.2.0)
4. AdoptOpenJDK 8 w/Hotspot (v1.8.0_221)
5. AdoptOpenJDK 8 w/OpenJ9 (v1.8.0_221)
6. OpenJDK 11 (v11.0.2)
7. OpenJDK 11 embedded Graal JVMCI (v11.0.2)
8. GraalVM Enterprise Edition C2 Compiler (v19.2.0)

Benchmarks on IntList operations

Three operations were benchmarked on primitive collections:

Benchmarks on IntList operations

Three operations were benchmarked on primitive collections:

- **Filter**: Filter in all even numbers (5 benchmarks)

Benchmarks on IntList operations

Three operations were benchmarked on primitive collections:

- **Filter**: Filter in all even numbers (5 benchmarks)
- **Sum**: Sum of all integers in the collection (5 benchmarks)

Benchmarks on IntList operations

Three operations were benchmarked on primitive collections:

- **Filter**: Filter in all even numbers (5 benchmarks)
- **Sum**: Sum of all integers in the collection (5 benchmarks)
- **Transform**: Multiply each integer by 2 (5 benchmarks)

Benchmarks on Object operations

Four operations were benchmarked on non-primitive collections:

Benchmarks on Object operations

Four operations were benchmarked on non-primitive collections:

- **Filter**: Filter all Persons with height between 80 and 150 inches (7 benchmarks)

Benchmarks on Object operations

Four operations were benchmarked on non-primitive collections:

- **Filter**: Filter all Persons with height between 80 and 150 inches (7 benchmarks)
- **FilterAndGroup**: Filter all Persons less than 150 inches tall, then group by age (7 benchmarks)

Benchmarks on Object operations

Four operations were benchmarked on non-primitive collections:

- `Filter`: Filter all Persons with height between 80 and 150 inches (7 benchmarks)
- `FilterAndGroup`: Filter all Persons less than 150 inches tall, then group by age (7 benchmarks)
- `IntSummaryStatistics`: Summary statistics by age (`integer`) of all Person instances (5 benchmarks)

Benchmarks on Object operations

Four operations were benchmarked on non-primitive collections:

- **Filter**: Filter all Persons with height between 80 and 150 inches (7 benchmarks)
- **FilterAndGroup**: Filter all Persons less than 150 inches tall, then group by age (7 benchmarks)
- **IntSummaryStatistics**: Summary statistics by age (integer) of all Person instances (5 benchmarks)
- **CombinedSummaryStatistics**: Summary statistics on height (double), weight (double) and age (integer) (5 benchmarks)

JDK Flags - Part 1

In order to run the embedded Graal compiler in OpenJDK 11:

- +XX:+UnlockExperimentalVMOptions
- +XX:+EnableJVMCI
- +XX:+UseJVMCICompiler

JDK Flags - Part 2

In order to run the C2 compiler in GraalVM EE v19.2.0:

- +XX:+UnlockExperimentalVMOptions
- +XX:-UseJVMCICompiler

Notice the minus before the UseJVMCICompiler

Hardware - The Beast

Model Name: Mac Pro

Model Identifier: MacPro6,1

Processor Name: 12-Core Intel Xeon E5

Processor Speed: 2.7 GHz

Number of Processors: 1

Total Number of Cores: 12

L2 Cache (per Core): 256 KB

L3 Cache: 30 MB

Memory: 64 GB



Hardware - The Little Monster

Model Name: **MacBook Pro**

Model Identifier: MacBookPro11,1

Processor Name: Intel Core i7

Processor Speed: 2.8 GHz

Number of Processors: 1

Total Number of Cores: 2

L2 Cache (per Core): 256 KB

L3 Cache: 4 MB

Memory: 16 GB



Benchmark Math

Some relevant numbers in terms of what was run:

Benchmark Math

Some relevant numbers in terms of what was run:

- * 7 Benchmark classes

Benchmark Math

Some relevant numbers in terms of what was run:

- * 7 Benchmark classes
- * 8 JDK/JVM combinations

Benchmark Math

Some relevant numbers in terms of what was run:

- * 7 Benchmark classes
- * 8 JDK/JVM combinations
- * 2 runs

Benchmark Math

Some relevant numbers in terms of what was run:

- * 7 Benchmark classes
- * 8 JDK/JVM combinations
- * 2 runs
- * Mode.Throughput on collections, seeded with size = 1_000_000

Benchmark Math

Some relevant numbers in terms of what was run:

- * 7 Benchmark classes
- * 8 JDK/JVM combinations
- * 2 runs
- * Mode.Throughput on collections, seeded with size = 1_000_000
- * Mode.Throughput on collections, seeded with size = 1000

Benchmark Math

Some relevant numbers in terms of what was run:

- * 7 Benchmark classes
- * 8 JDK/JVM combinations
- * 2 runs
- * Mode.Throughput on collections, seeded with size = 1_000_000
- * Mode.Throughput on collections, seeded with size = 1000
- * 39 benchmarks in the 7 classes = 2184 benchmarks.

Benchmark Math

Some relevant numbers in terms of what was run:

- * 7 Benchmark classes
- * 8 JDK/JVM combinations
- * 2 runs
- * Mode.Throughput on collections, seeded with size = 1_000_000
- * Mode.Throughput on collections, seeded with size = 1000
- * 39 benchmarks in the 7 classes = 2184 benchmarks.
- * Executed twice = 4368 total benchmarks.

Benchmark Math

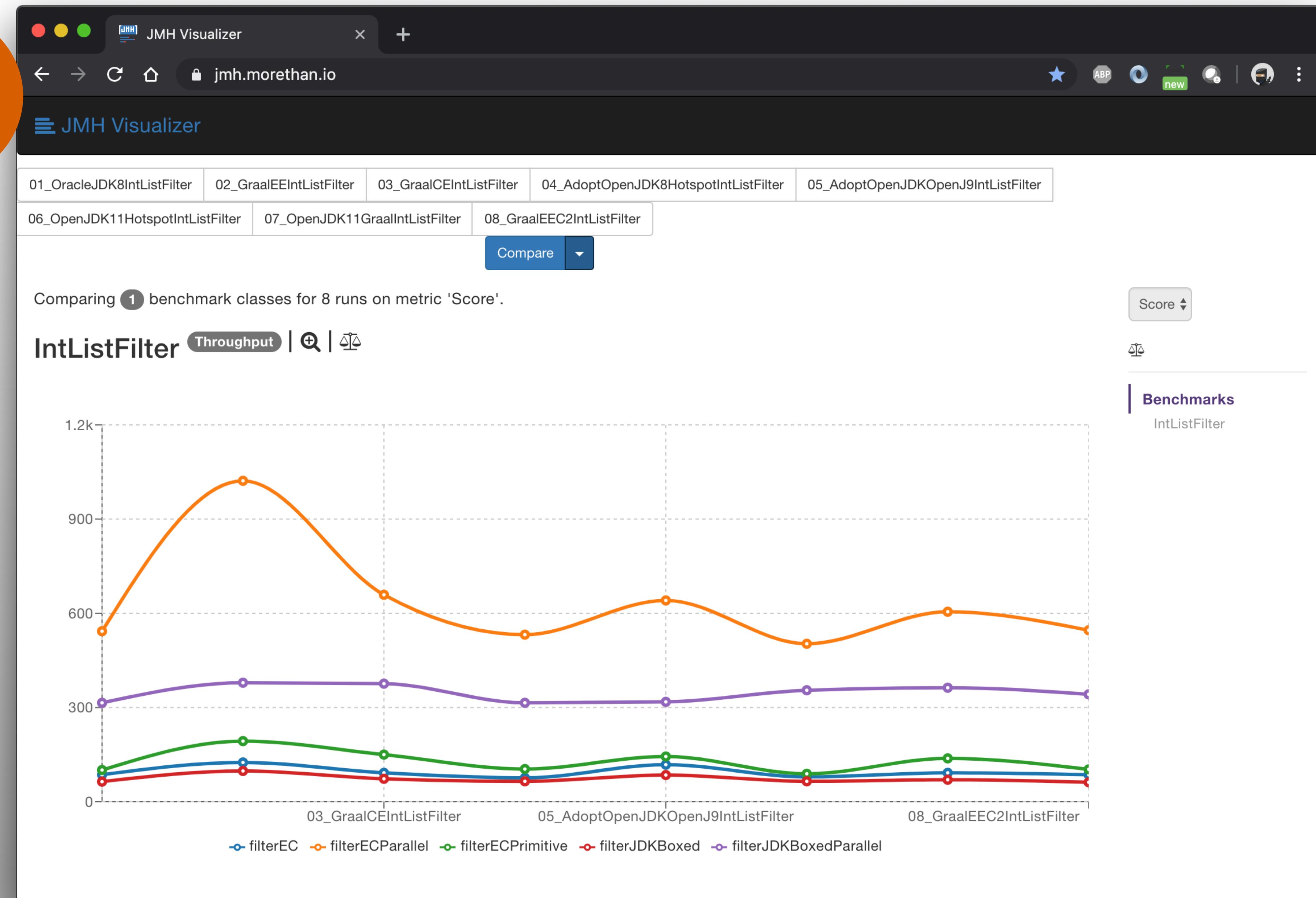
Some relevant numbers in terms of what was run:

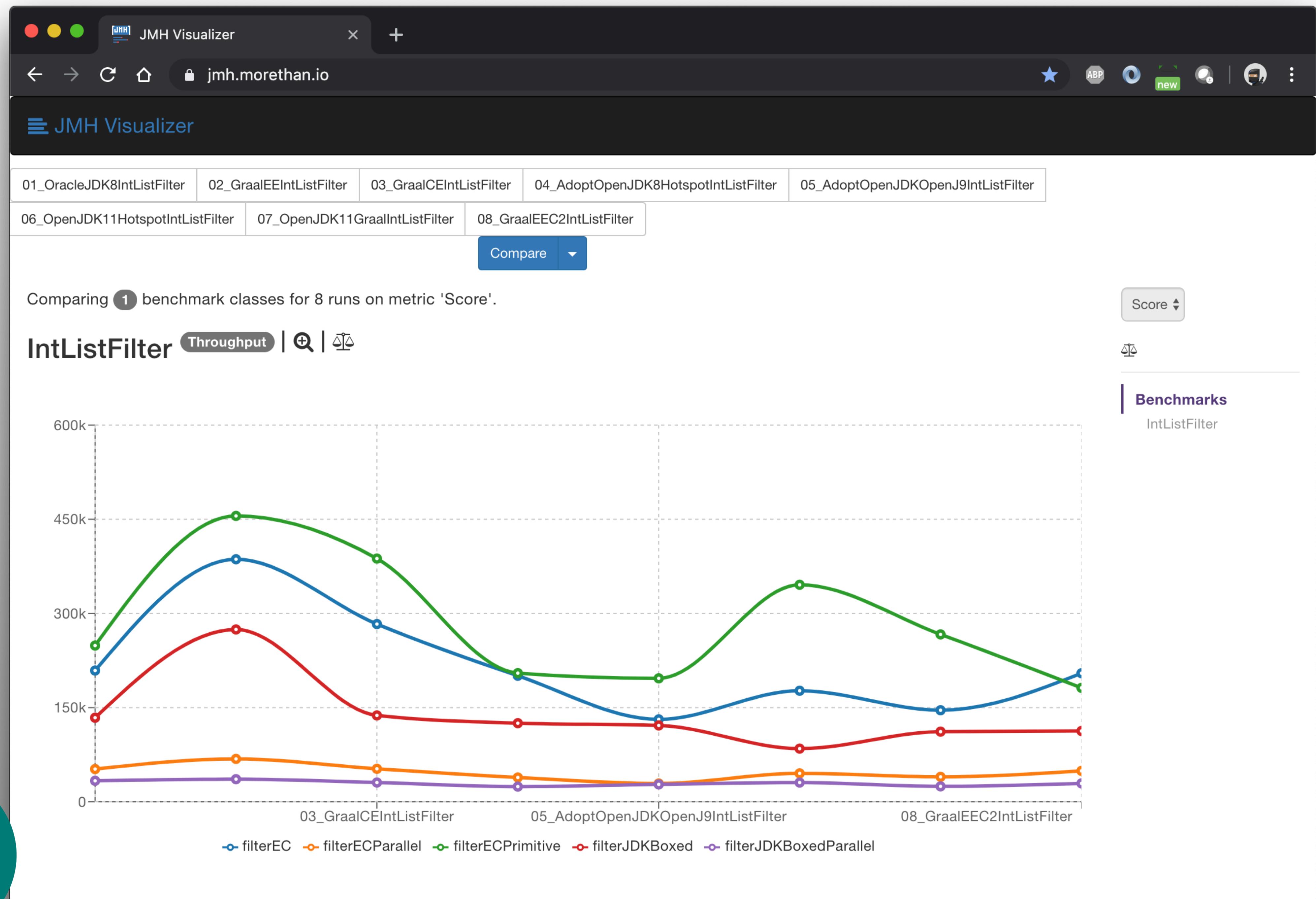
- * 7 Benchmark classes
- * 8 JDK/JVM combinations
- * 2 runs
- * Mode.Throughput on collections, seeded with size = 1_000_000
- * Mode.Throughput on collections, seeded with size = 1000
- * 39 benchmarks in the 7 classes = 2184 benchmarks.
- * Executed twice = 4368 total benchmarks.

HANDS ON !

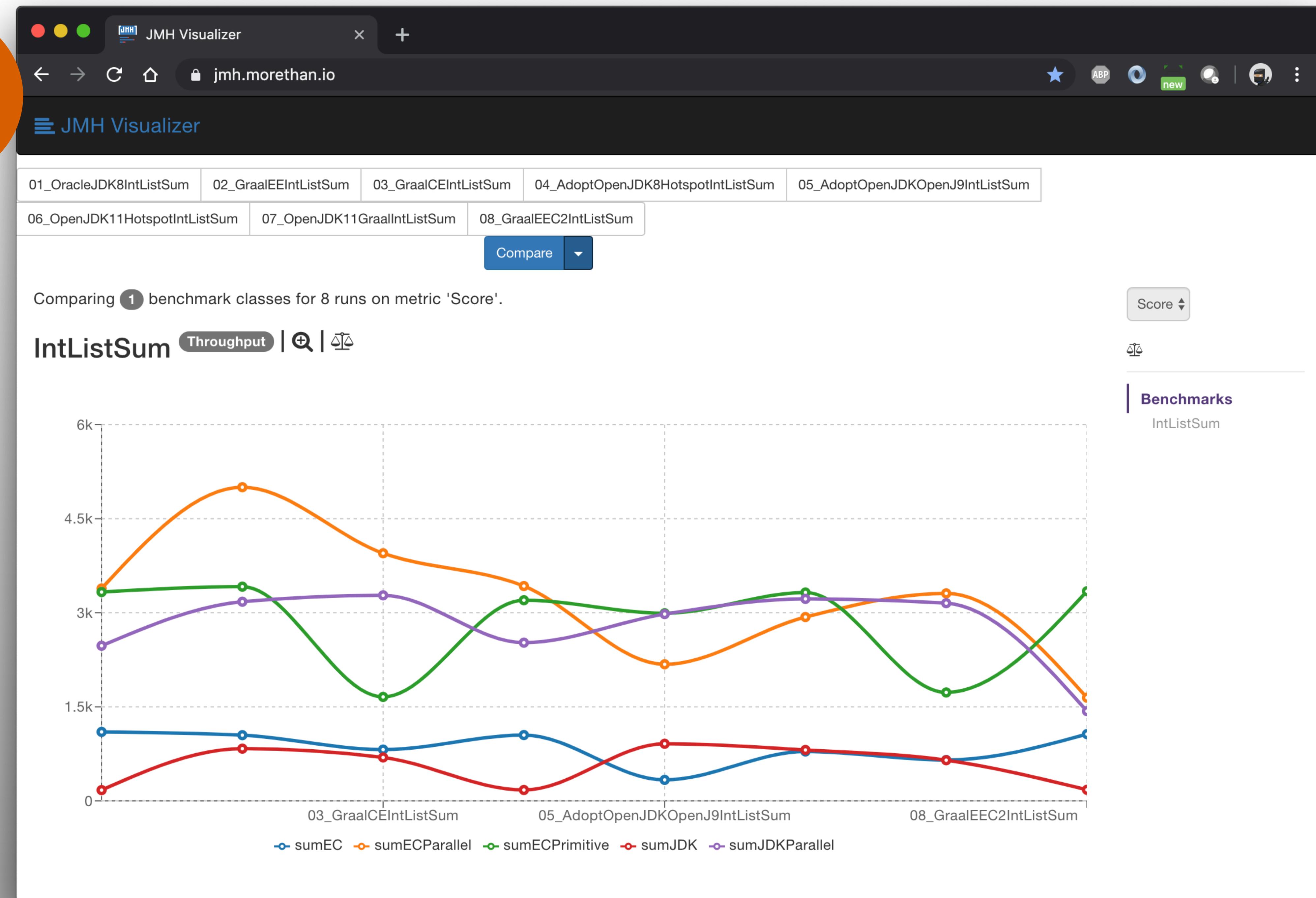


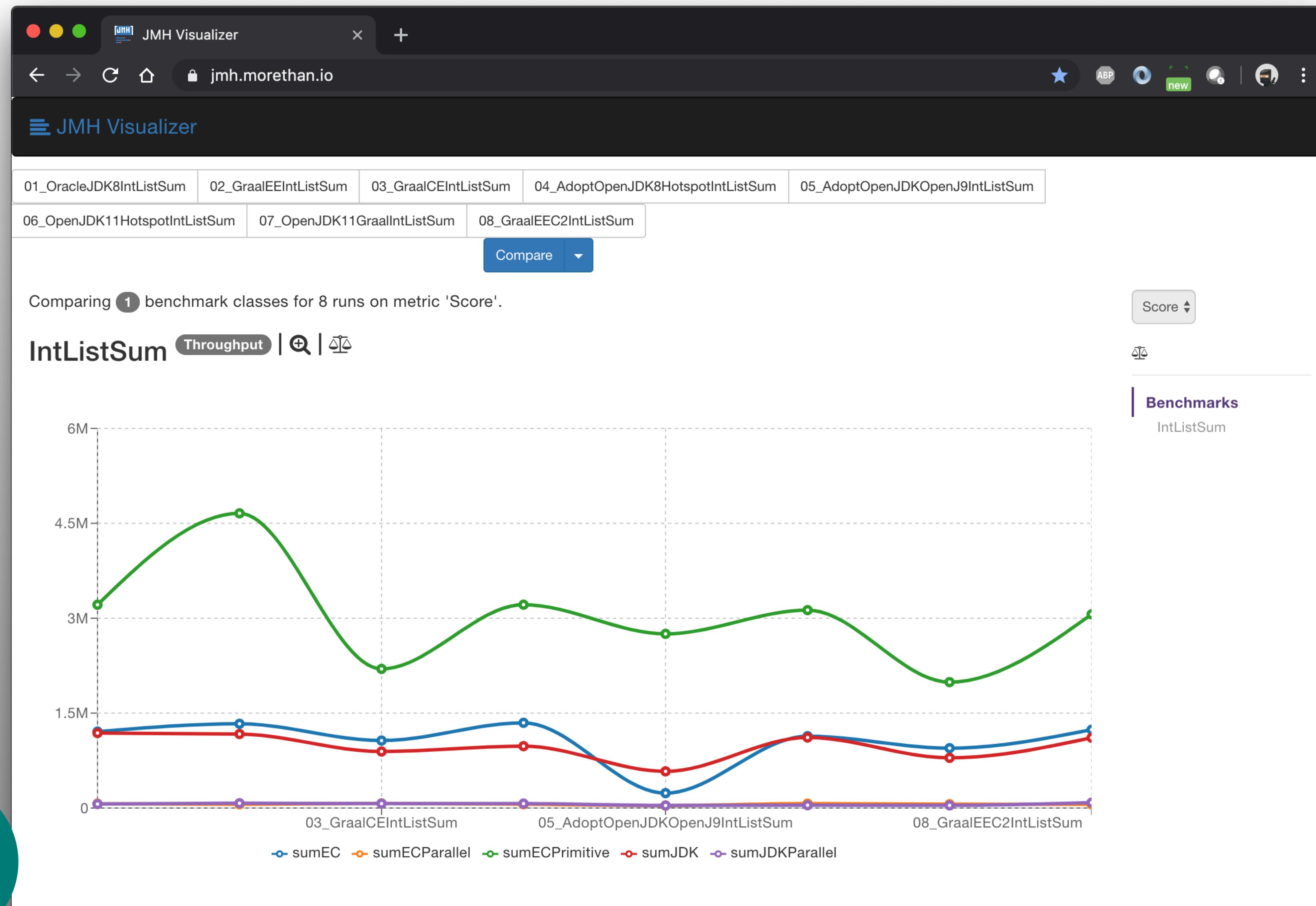
Collection
Size
1M





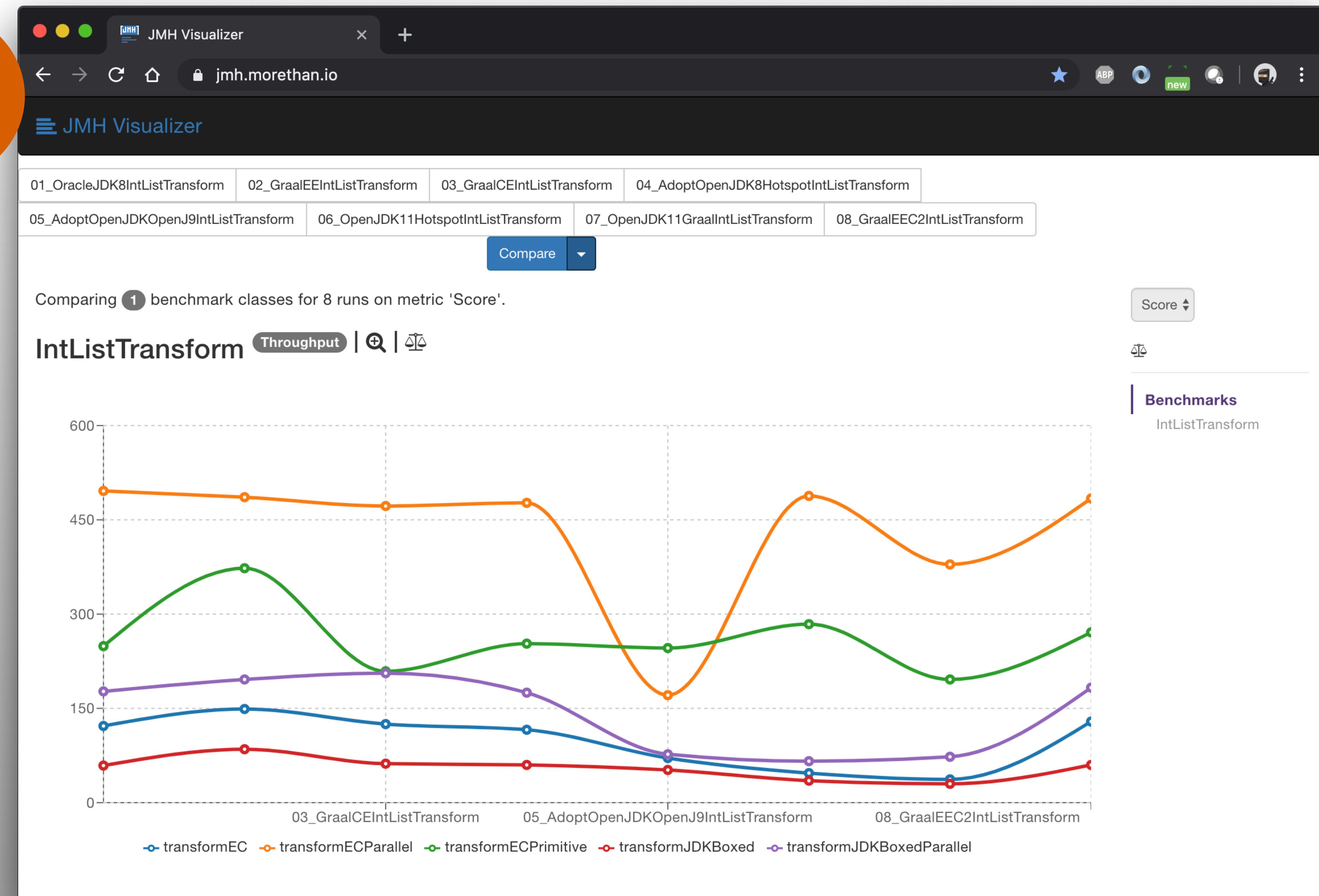
Collection
Size
1M

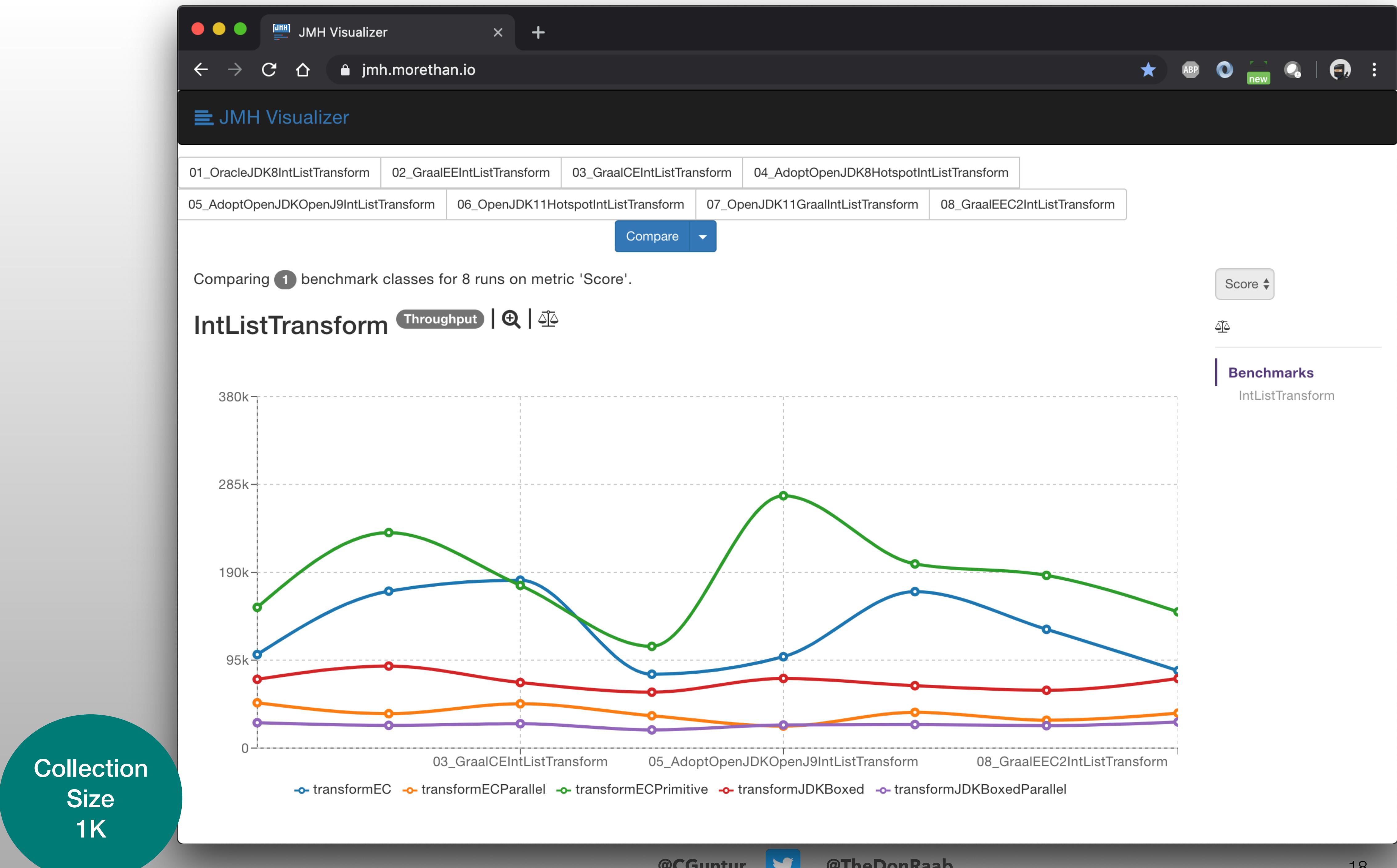




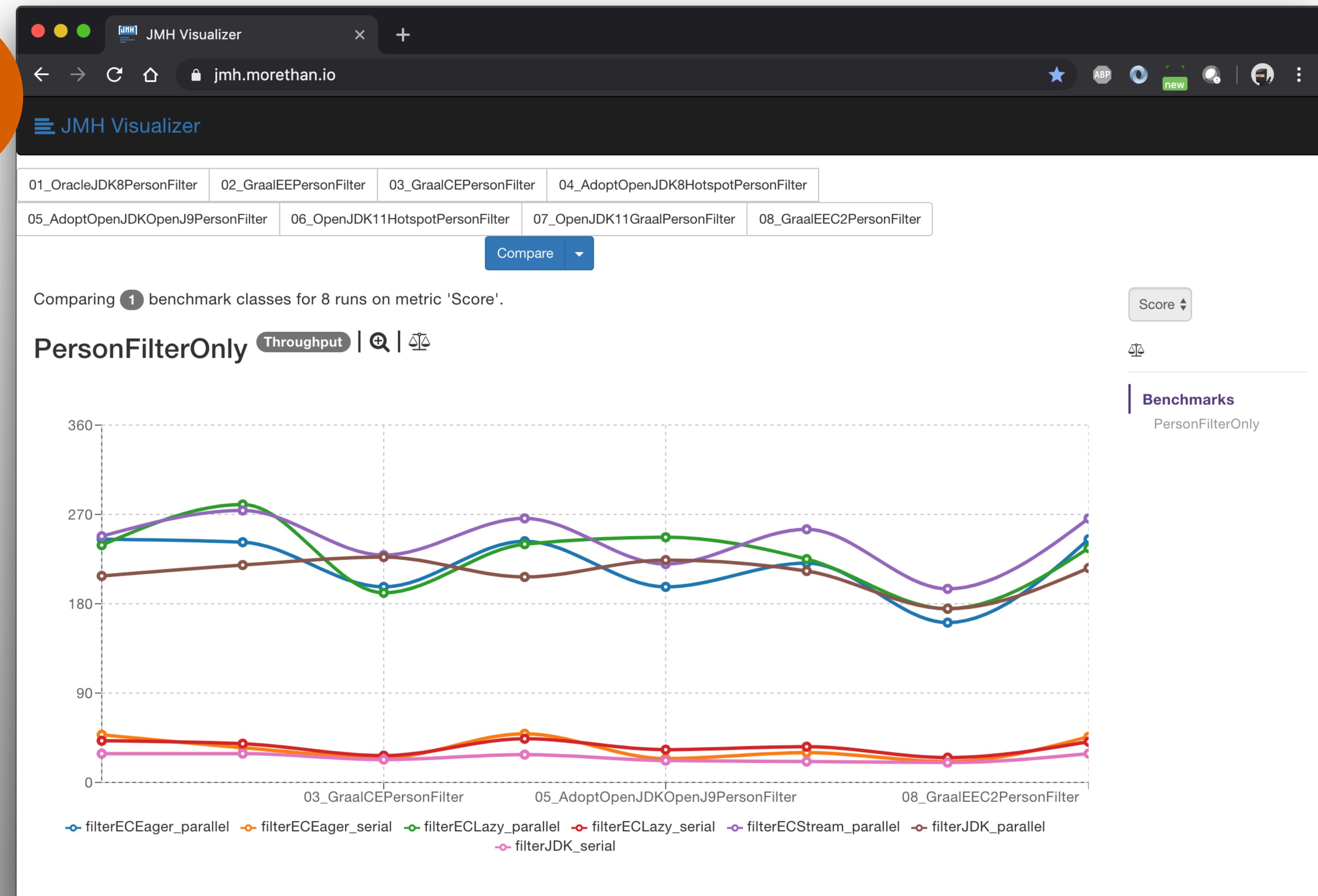
Collection
Size
1K

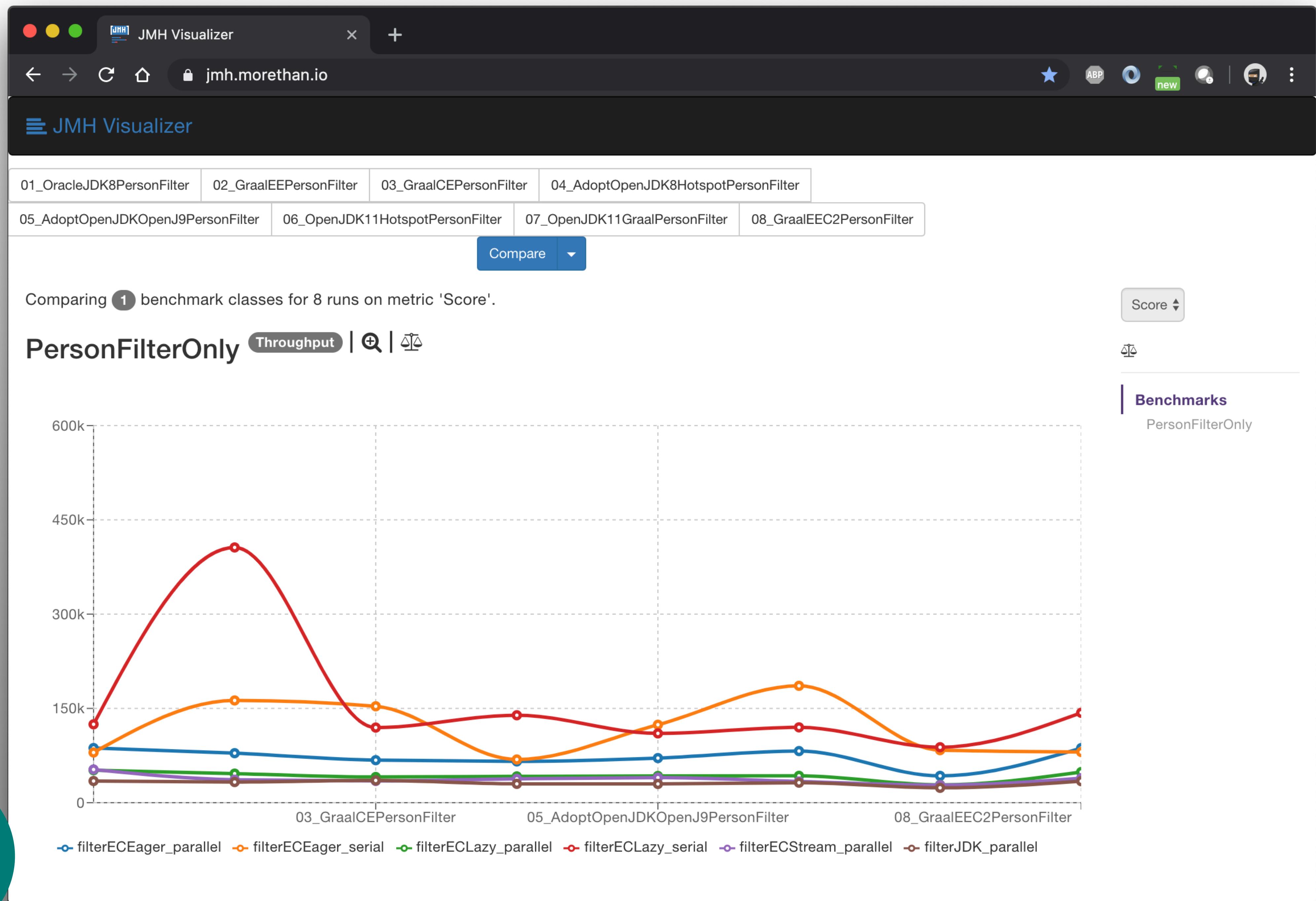
Collection Size 1M



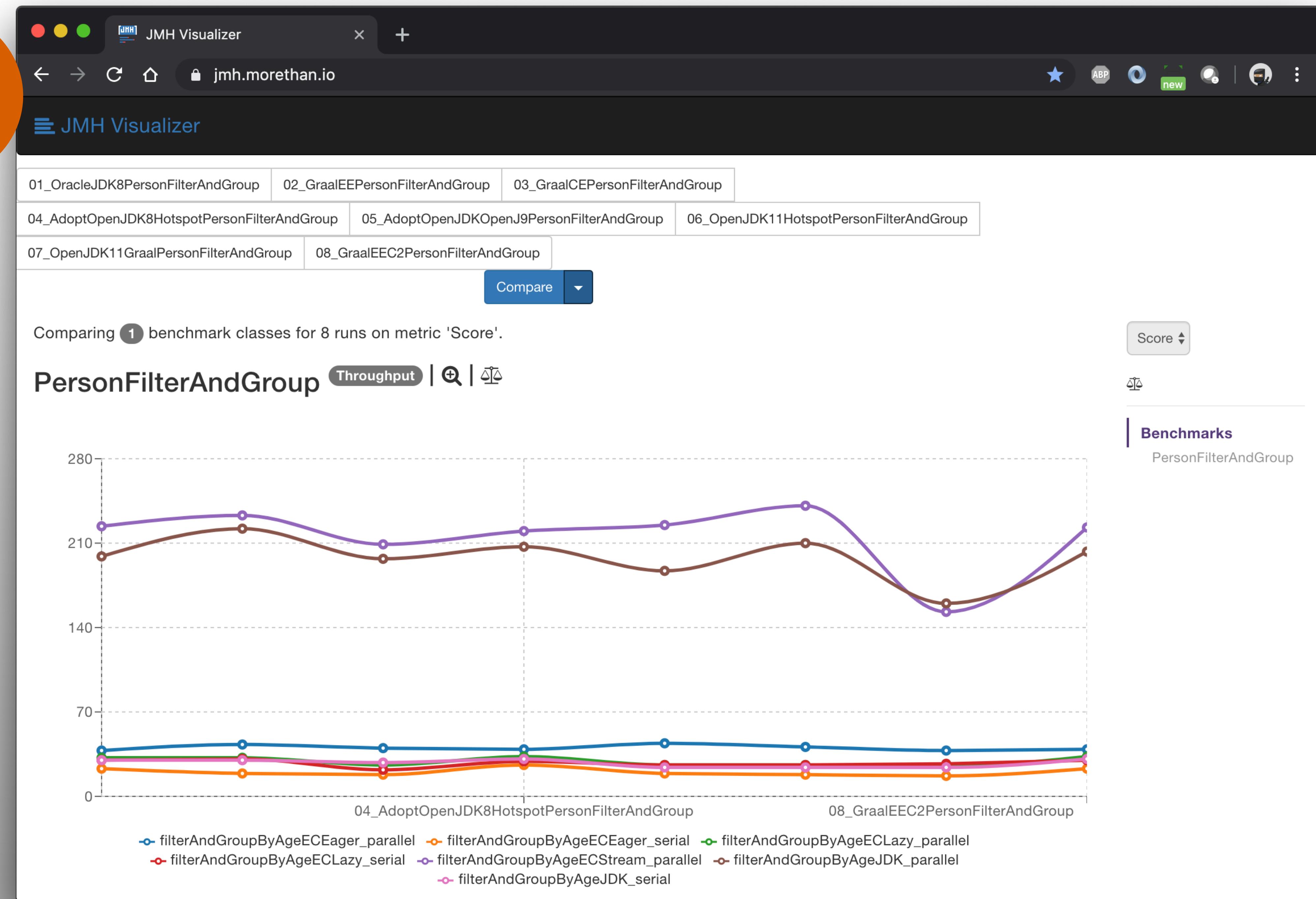


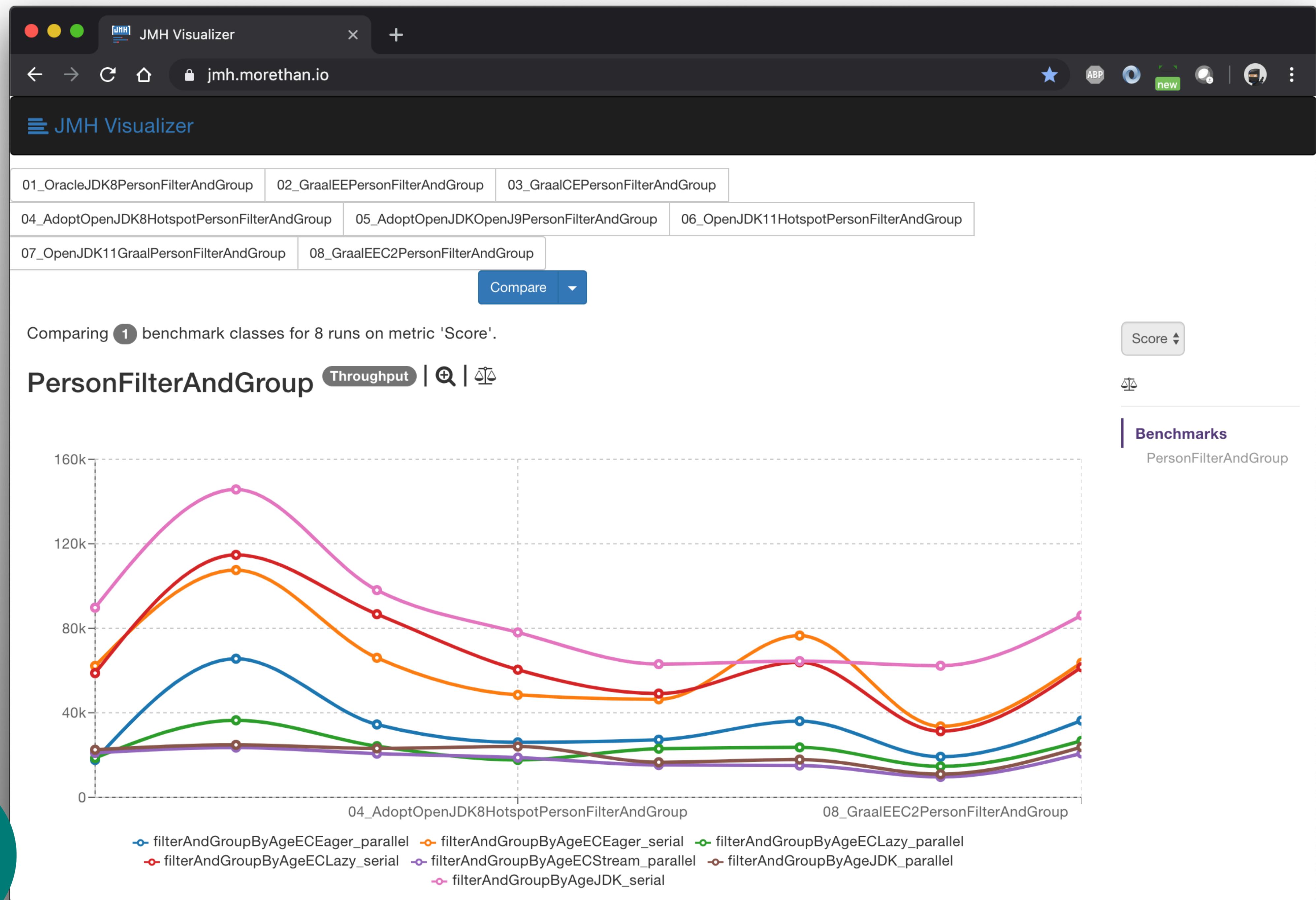
Collection Size 1M





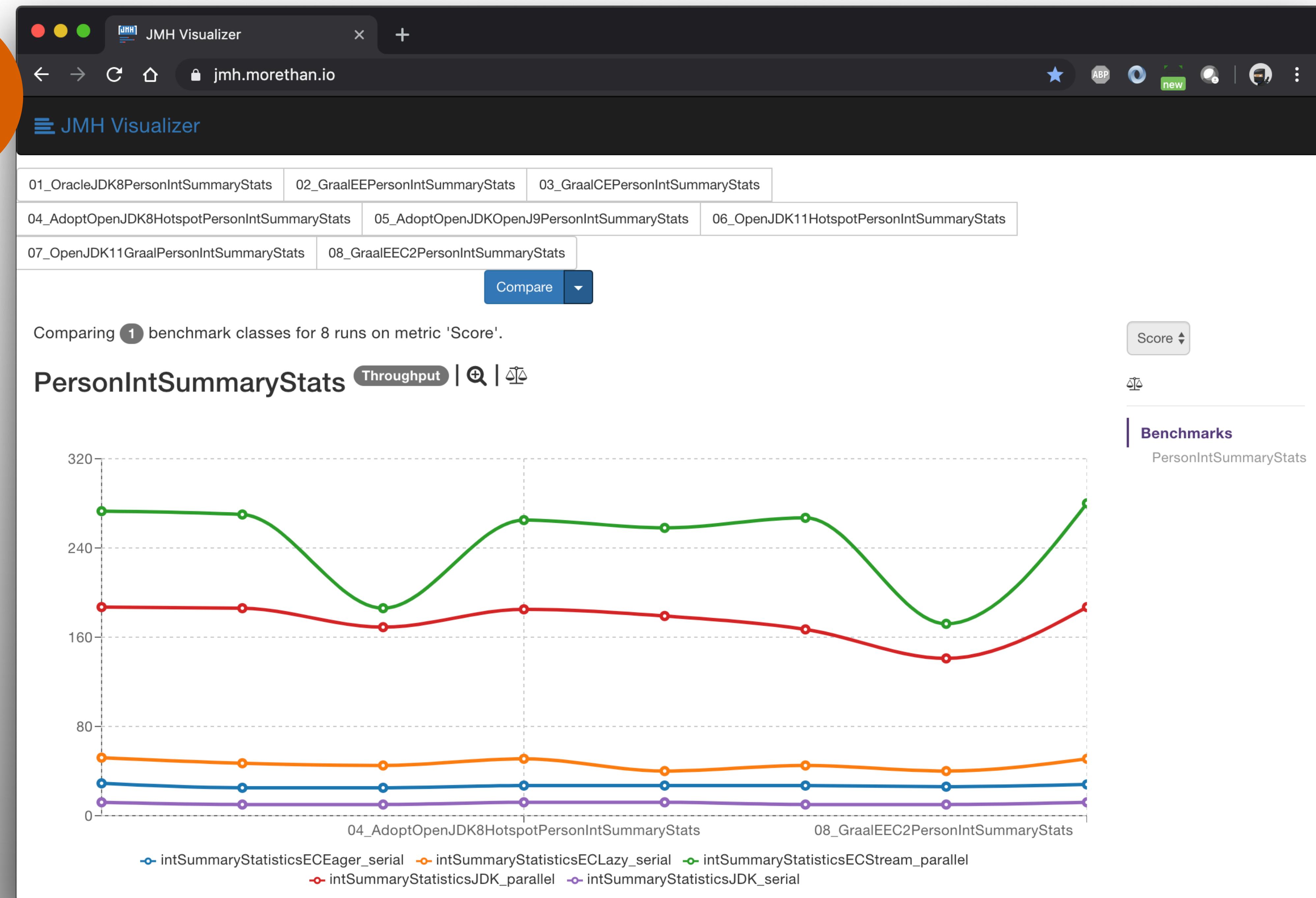
Collection
Size
1M

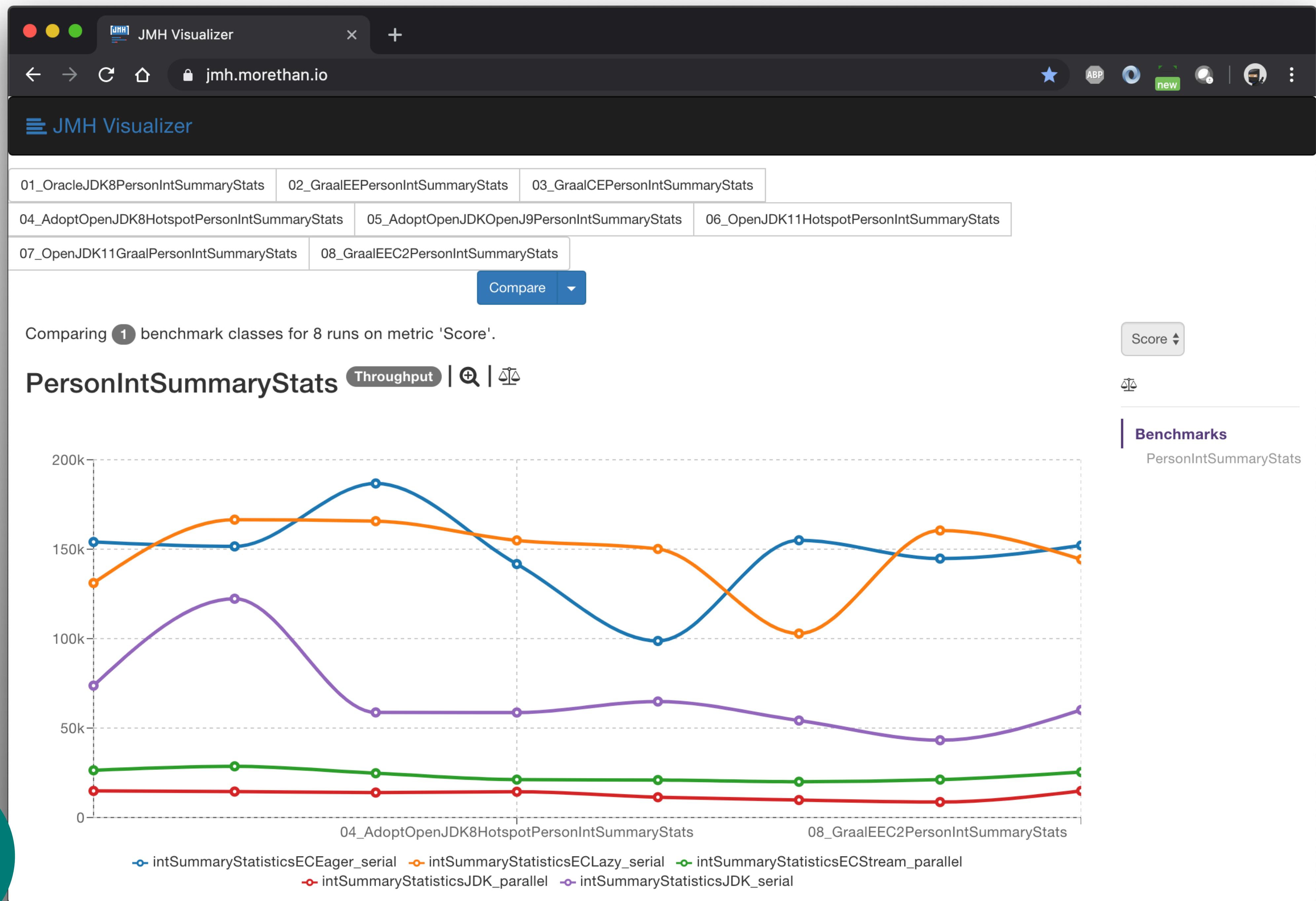




Collection
Size
1K

Collection Size 1M





Collection Size

1M

