Java 8 Stream Vs ParallelStream – 2023

A sequential stream is executed in a single thread running on one CPU core. **The elements in the stream are processed sequentially in a single pass** **by the stream operations that are executed in the same thread**.

A parallel stream is executed by different threads, running on multiple CPU cores in a computer.

**Parallel streams utilize the fork/join framework for executing parallel tasks**. This framework provides support for the thread management necessary to execute the substreams in parallel. The number of threads employed during parallel stream execution is dependent on the CPU cores in the computer.

**Using Stream for sequential execution**

public class StreamParallelStreamTest {  
  
 public static String doIt(String param) {  
 try {  
 TimeUnit.*SECONDS*.sleep(3);  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 System.*out*.println(param+" completed execution ...");  
 return param.toUpperCase();  
 }  
  
 public static void main(String[] args) {  
 List<String> datas = Arrays.*asList*("a","b","c","d","e","f","g");  
 Instant start = Instant.*now*();  
 *//Using sequentially* List<String> modifiedDatas = **datas.stream().**map( value -> *doIt*(value))  
 .collect(Collectors.*toList*());  
 Instant end = Instant.*now*();  
 long totalTime = Duration.*between*(start,end).toMillis();  
 System.*out*.println(modifiedDatas);  
 System.*out*.println("Total Time Taken: "+totalTime);  
 }  
}

**OUTPUT**

[A, B, C, D, E, F, G]

Total Time Taken: **21066** milliseconds

**Using ParallelStream for parallel execution**

public class StreamParallelStreamTest {  
  
 public static String doIt(String param) {  
 try {  
 TimeUnit.*SECONDS*.sleep(3);  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 System.*out*.println(param+" completed execution ...");  
 return param.toUpperCase();  
 }  
  
 public static void main(String[] args) {  
 List<String> datas = Arrays.*asList*("a","b","c","d","e","f","g");  
 Instant start = Instant.*now*();  
 *//Using sequentially* List<String> modifiedDatas = **datas.parallelStream()**.map( value -> {  
 return *doIt*(value);  
 }).collect(Collectors.*toList*());  
 Instant end = Instant.*now*();  
 long totalTime = Duration.*between*(start,end).toMillis();  
 System.*out*.println(modifiedDatas);  
 System.*out*.println("Total Time Taken: "+totalTime+" milliseconds");  
 }  
}

**OUTPUT**

[A, B, C, D, E, F, G]

Total Time Taken: **3016** milliseconds