Overview of Java Parallel Streams: Phases

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Know how aggregate operations & functional programming features are applied in the parallel streams framework
- Be aware of how parallel stream phases work “under the hood”

Overview of How a Parallel Stream Works
A Java parallel stream implements a “map/reduce” variant optimized for multi-core processors.

See en.wikipedia.org/wiki/MapReduce
A Java parallel stream implements a “map/reduce” variant optimized for multi-core processors.

It’s actually a three phase “split-apply-combine” data processing strategy.

See www.jstatsoft.org/article/view/v040i01
Overview of How a Parallel Stream Works

- The split-apply-combine phases are:

1. **Split** – Recursively partition a data source into independent “chunks”

   - **trySplit()**

   - **trySplit()**

   - **trySplit()**

   - **trySplit()**

Overview of How a Parallel Stream Works

- The split-apply-combine phases are:
  
  **1. Split** – Recursively partition a data source into independent “chunks”

- Spliterators are defined to partition collections in Java

```java
public interface Spliterator<T> {
    boolean tryAdvance(Consumer<? Super T> action);

    Spliterator<T> trySplit();

    long estimateSize();

    int characteristics();
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html](http://docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html)
The split-apply-combine phases are:

1. **Split** – Recursively partition a data source into independent “chunks”

- Spliterators are defined to partition collections in Java

```java
public interface Spliterator<T> {
    boolean tryAdvance(Consumer<? Super T> action);
    Spliterator<T> trySplit();
    long estimateSize();
    int characteristics();
}
```
The split-apply-combine phases are:

1. **Split** – Recursively partition a data source into independent “chunks”

   - Spliterators are defined to partition collections in Java

```
public interface Spliterator<T> {
    boolean tryAdvance(Consumer<? Super T> action);
    Spliterator<T> trySplit();
    long estimateSize();
    int characteristics();
}
```

*Used only for parallel streams*
Overview of How a Parallel Stream Works

- The split-apply-combine phases are:

  1. **Split** – Recursively partition a data source into independent “chunks”
    - Spliterators are defined to partition collections in Java
    - You can also define custom spliterators

See [github.com/douglascraigschmidt/LiveLessons/tree/master/SearchStreamSpliterator](https://github.com/douglascraigschmidt/LiveLessons/tree/master/SearchStreamSpliterator)
Overview of How a Parallel Stream Works

- The split-apply-combine phases are:

1. **Split** – Recursively partition a data source into independent “chunks”
   - Spliterators are defined to partition collections in Java
   - You can also define custom spliterators
   - Parallel streams perform better on data sources that can be split efficiently & evenly

Overview of How a Parallel Stream Works

- The split-apply-combine phases are:

  1. **Split** – Recursively partition a data source into independent “chunks”

  2. **Apply** – Process chunks independently in one (common) thread pool

Splitting & applying run simultaneously (after certain limit met), not sequentially.
Overview of How a Parallel Stream Works

- The split-apply-combine phases are:

1. **Split** – Recursively partition a data source into independent “chunks”

2. **Apply** – Process chunks independently in one (common) thread pool

- Programmers have some control over how many threads are in the pool
Overview of How a Parallel Stream Works

- The split-apply-combine phases are:

  1. **Split** – Recursively partition a data source into independent “chunks”

  2. **Apply** – Process chunks independently in one (common) thread pool

  3. **Combine** – Join partial results into a single result
Overview of How a Parallel Stream Works

- The split-apply-combine phases are:

  1. **Split** – Recursively partition a data source into independent “chunks”
  
  2. **Apply** – Process chunks independently in one (common) thread pool
  
  3. **Combine** – Join partial results into a single result

- Performed by terminal operations like `collect()` & `reduce()`

See [www.codejava.net/java-core/collections/java-8-stream-terminal-operations-examples](http://www.codejava.net/java-core/collections/java-8-stream-terminal-operations-examples)
End of Overview of Java
Parallel Streams: Phases