Java Parallel Streams Internals: Implementing a Concurrent Map Collector

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

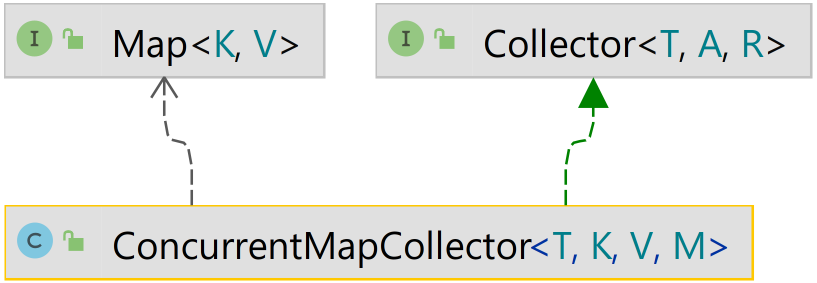
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand parallel stream internals, e.g.
  - Know what can change & what can't
  - Partition a data source into "chunks"
  - Process chunks in parallel via the common fork-join pool
  - Configure the Java parallel stream common fork-join pool
  - Perform a reduction to combine partial results into a single result
  - Recognize key behaviors & differences of non-concurrent & concurrent collectors
  - Be aware of non-concurrent & concurrent collector APIs
  - Grok performance variance in concurrent & non-concurrent collectors
  - Learn how to implement a concurrent Map collector



Rationale for a Concurrent Map Collector

Rationale for a Concurrent Map Collector

- The Java Collectors utility class provides factory methods that make non-concurrent collectors

```
<<Java Class>>
Collectors
Collectors()
toCollection(Supplier<C>):Collector<T,?,C>
toList():Collector<T,?,List<T>>
toSet():Collector<T,?,Set<T>>
toMap(Function<? super T,? extends K>,Function<? super T,? extends U>):Collector<T,?,Map<K,U>>
```

See www.baeldung.com/java-8-collectors

Rationale for a Concurrent Map Collector

- The Java Collectors utility class provides factory methods that make non-concurrent collectors
  - It also contains some factory methods that make collectors based on ConcurrentMap
    - e.g., ConcurrentHashMap & ConcurrentSkipListMap

`static <T,K,U> Collector<T,?,ConcurrentMap<K,U>>`	`toConcurrentMap(Function<? super T,? extends K> keyMapper, Function<? super T,? extends U> valueMapper)` Returns a concurrent `Collector` that accumulates elements into a `ConcurrentMap` whose keys and values are the result of applying the provided mapping functions to the input elements.
`static <T,K,U> Collector<T,?,ConcurrentMap<K,U>>`	`toConcurrentMap(Function<? super T,? extends K> keyMapper, Function<? super T,? extends U> valueMapper, BinaryOperator<U> mergeFunction)` Returns a concurrent `Collector` that accumulates elements into a `ConcurrentMap` whose keys and values are the result of applying the provided mapping functions to the input elements.
`static <T,K,U,M extends ConcurrentMap<K,U>> Collector<T,?,M>`	`toConcurrentMap(Function<? super T,? extends K> keyMapper, Function<? super T,? extends U> valueMapper, BinaryOperator<U> mergeFunction, Supplier<M> mapSupplier)` Returns a concurrent `Collector` that accumulates elements into a `ConcurrentMap` whose keys and values are the result of applying the provided mapping functions to the input elements.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentMap.html

Rationale for a Concurrent Map Collector

- The Java Collectors utility class provides factory methods that make non-concurrent collectors
  - It also contains some factory methods that make collectors based on ConcurrentMap
  - However, there are no pre-defined concurrent collectors provided by Java that return *sorted* maps
    - e.g., TreeMap

Class TreeMap<K,V>

java.lang.Object
    java.util.AbstractMap<K,V>
        java.util.TreeMap<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

All Implemented Interfaces:

Serializable, Cloneable, Map<K,V>, NavigableMap<K,V>, SortedMap<K,V>

```
public class TreeMap<K,V>
extends AbstractMap<K,V>
implements NavigableMap<K,V>, Cloneable, Serializable
```

A Red-Black tree based NavigableMap implementation. The map is sorted according to the natural ordering of its keys, or by a Comparator provided at map creation time, depending on which constructor is used.

This implementation provides guaranteed log(n) time cost for the containsKey, get, put and remove operations. Algorithms are adaptations of those in Cormen, Leiserson, and Rivest's *Introduction to Algorithms*.

See docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html

Rationale for a Concurrent Map Collector

- The ConcurrentMapCollector is designed to overcome this omission with the Java class library

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

See Java8/ex37/src/main/java/utils/ConcurrentMapCollector.java

Rationale for a Concurrent Map Collector

- The ConcurrentMapCollector is designed to overcome this omission with the Java class library

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

The Supplier param can be used to customize the type of Map returned from this collector

See docs.oracle.com/javase/8/docs/api/java/util/function/Supplier.html

Implementing a Generic Concurrent Map Collector

Implementing a Generic Concurrent Map Collector

- ConcurrentMapCollector defines four generic types

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing a Generic Concurrent Map Collector

- ConcurrentMapCollector defines four generic types
  - **T** – The type of objects available from the Stream
    - e.g., String, GCDParam, SimpleImmutableEntry, etc.

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

See docs.oracle.com/javase/8/docs/api/java/util/AbstractMap.SimpleImmutableEntry.html

Implementing a Generic Concurrent Map Collector

- ConcurrentMapCollector defines four generic types
  - T
  - **K** – The type of the key used in the map
    - e.g., Double, String, etc.

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing a Generic Concurrent Map Collector

- ConcurrentMapCollector defines four generic types
  - T
  - A
  - **V** – The type of the value used in the map
    - e.g., SearchResults, Integer, GCDResult, etc.

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing a Generic Concurrent Map Collector

- ConcurrentMapCollector defines four generic types
  - T
  - A
  - V
  - **M** – The type of Map returned from the collector
    - e.g., ConcurrentHashMap, TreeMap, LinkedHashMap, etc.

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

ConcurrentMapCollector uses ConcurrentHashMap internally

Implementing a Generic Concurrent Map Collector

- The toMap() factory method creates a new instance of ConcurrentMapCollector that is parameterized by Java functional interface objects

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing a Generic Concurrent Map Collector

- The toMap() factory method creates a new instance of ConcurrentMapCollector that is parameterized by Java functional interface objects
  - e.g., `return new ConcurrentMapCollector<>(keyMapper, valueMapper, mergeFunction, mapSupplier);`

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing Concurrent Map Collector Methods

Implementing Concurrent Map Collector Methods

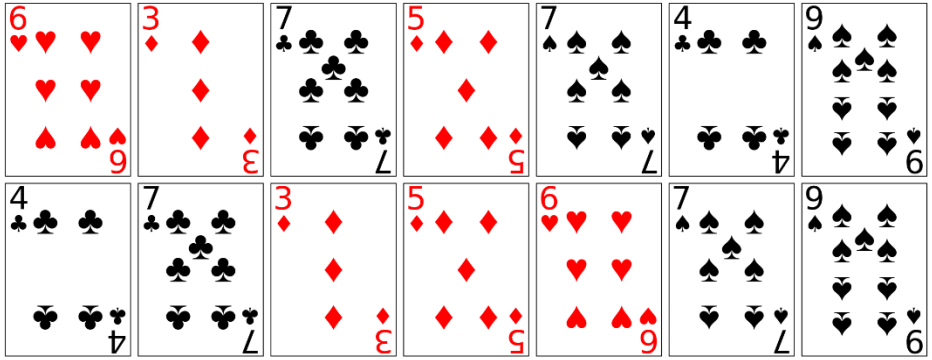
- Five key methods are defined in the ConcurrentMapCollector

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - **characteristics()** – provides additional info to optimize the collector, e.g.
    - UNORDERED
      - The collector need not preserve encounter order

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```



Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - **characteristics()** – provides additional info to optimize the collector, e.g.
    - UNORDERED
    - CONCURRENT
      - accumulator() is called concurrently on the ConcurrentHashMap mutable result container

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

ConcurrentHashMap methods are all synchronized!!

See www.geeksforgeeks.org/concurrenthashmap-in-java

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - **characteristics()** – provides additional info to optimize the collector, e.g.

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Any/all characteristics can be set using EnumSet.of()

```
Set<Characteristics> characteristics() {
  return Collections.unmodifiableSet
    (EnumSet.of(Collector.Characteristics.CONCURRENT,
                Collector.Characteristics.UNORDERED));
}
```

See docs.oracle.com/javase/8/docs/api/java/util/EnumSet.html

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - characteristics()
  - **supplier()** – returns a Supplier that acts as a factory method to generate an empty result container

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - characteristics()
  - **supplier()** – returns a Supplier that acts as a factory method to generate an empty result container, e.g.
    - `return ConcurrentHashMap::new`

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - characteristics()
  - supplier()
  - **accumulator()** – returns a BiConsumer that adds a new element to the existing ConcurrentHashMap

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - characteristics()
  - supplier()
  - **accumulator()** – returns a BiConsumer that adds a new element to the existing ConcurrentHashMap, e.g.
    - `return (Map<K, V> map, T element) -> map`
      `.merge(mKeyMapper.apply(element),`
      `mValueMapper.apply(element),`
      `mMergeFunction);`

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

ConcurrentHashMap's merge() method is efficiently synchronized

See codepumpkin.com/hashtable-vs-synchronizedmap-vs-concurrenthashmap

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - characteristics()
  - supplier()
  - accumulator()
  - **combiner()** – returns a Binary Operator that merges two result containers together

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - characteristics()
  - supplier()
  - accumulator()
  - **combiner()** – returns a Binary Operator that merges two result containers together, e.g.
    - `return (one, another) -> { one.putAll(another); return one; }`

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

This method is only called for non-concurrent collectors..

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - characteristics()
  - supplier()
  - accumulator()
  - combiner()
  - **finisher()** – returns a Function that converts ConcurrentHashMap to the final Map result type

```
ConcurrentMapCollector<T, K, V, M>
ConcurrentMapCollector(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)
finisher()                                                    Function<Map<K, V>, M>
accumulator()                                                 BiConsumer<Map<K, V>, T>
characteristics()                                             Set<Characteristics>
toMap(Function<T, K>, Function<T, V>, BinaryOperator<V>, Supplier<M>)   Collector<T, ?, M>
supplier()                                                    Supplier<Map<K, V>>
combiner()                                                    BinaryOperator<Map<K, V>>
```

Implementing Concurrent Map Collector Methods

- Five key methods are defined in the ConcurrentMapCollector
  - characteristics()
  - supplier()
  - accumulator()
  - combiner()
  - **finisher()** – returns a Function that converts ConcurrentHashMap to the final Map result type, e.g.

```
return map -> {
  M newMap =
    mMapSupplier.get();

  if (newMap instanceof
      ConcurrentHashMap)
    return (M) map;
  else {
    newMap.putAll(map);
    return newMap;
  }
};
```

Only copies data if M isn't a ConcurrentHashMap

Implementing Concurrent Map Collector Methods

```java
/**
 * A concurrent collector that accumulates input elements of type {@code T}
 * into a {@link ConcurrentHashMap} parameterized with {@code K} and {@code V}
 * types and returns a type {@link M} that extends {@link Map}.
 */
public class ConcurrentMapCollector<T, K, V, M extends Map<K, V>>
        implements Collector<T,
                             Map<K, V>,
                             M> {
    /**
     * A {@link Function} that maps the key {@code K}.
     */
    private final Function<? super T, ? extends K> mKeyMapper;

    /**
     * A {@link Function} that maps the value {@code V}.
     */
    private final Function<? super T, ? extends V> mValueMapper;

    /**
     * A {@link Supplier} that returns a new, empty {@code Map} into
     * which the results will be inserted.
     */
    private final Supplier<M> mMapSupplier;
```

See Java8/ex37/src/main/java/utils/ConcurrentMapCollector.java

End of Java Parallel Streams Internals: Implementing a Concurrent Map Collector