How Parallel Programs are Developed in Java (Part 2)

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

• Recognize the parallelism frameworks supported by Java, e.g.
  • Fork-join pools
  • Parallel streams
    • A synchronous parallel functional programming framework

See docs.oracle.com/javase/tutorial/collections/streams/parallelism.html
Learning Objectives in this Part of the Lesson

- Recognize the parallelism frameworks supported by Java, e.g.
  - Fork-join pools
  - Parallel streams
  - **Completable futures**
    - An asynchronous parallel functional programming framework

---

See [www.callicoder.com/java-8-completablefuture-tutorial](http://www.callicoder.com/java-8-completablefuture-tutorial)
Learning Objectives in this Part of the Lesson

- Recognize the parallelism frameworks supported by Java, e.g.
  - Fork-join pools
  - Parallel streams
  - Completable futures

These parallel functional programming frameworks leverage the object-oriented fork-join pool framework by default

See previous lesson on "How Parallel Programs are Developed in Java (Part 1)"
Overview of Parallel Functional Programming Frameworks
Modern Java provides two parallel functional programming frameworks.
Modern Java provides two parallel functional programming frameworks

1. Parallel streams

```java
filter(not(this::urlCached))
map(this::downloadImage)
map(this::applyFilters)
reduce(Stream::concat) ...
collect(toList())
```

See [docs.oracle.com/javase/tutorial/collections/streams/parallelism.html](docs.oracle.com/javase/tutorial/collections/streams/parallelism.html)
Modern Java provides two parallel functional programming frameworks

1. Parallel streams

Process a list of URLs to images that aren’t already cached by downloading, transforming, & storing these images in parallel

Overview of Parallel Functional Programming Frameworks

- Modern Java provides two parallel functional programming frameworks

1. **Parallel streams**
   - Partitions a stream into multiple "chunks" that run independently & combine into a "reduced" result
• Modern Java provides two parallel functional programming frameworks

1. Parallel streams
   • Partitions a stream into multiple “chunks” that run independently & combine into a “reduced” result
   • These chunks of data are mapped to multiple threads (& cores) & thus processed in parallel
Overview of Parallel Functional Programming Frameworks

1. **Parallel streams**
   - Partitions a stream into multiple “chunks” that run independently & combine into a “reduced” result
   - These chunks of data are mapped to multiple threads (& cores) & thus processed in parallel
   - Leverages the Java common fork-join pool

See [dzone.com/articles/common-fork-join-pool-and-streams](dzone.com/articles/common-fork-join-pool-and-streams)
Overview of Parallel Functional Programming Frameworks

• Modern Java provides two parallel functional programming frameworks

1. Parallel streams
   • Partitions a stream into multiple “chunks” that run independently & combine into a “reduced” result
   • These chunks of data are mapped to multiple threads (& cores) & thus processed in parallel
   • Leverages the Java common fork-join pool

Parallel streams provide fine-grained data parallelism functional programming

```
filter(not(this::urlCached))
map(this::downloadImage)
map(this::applyFilters)
reduce(Stream::concat) ...
collect(toList())
```
Modern Java provides two parallel functional programming frameworks:

1. **Parallel streams**

2. **CompletableFuture**

---

**Task 1**
```
 supplyAsync(getStartPage())
```

**Task 2**
```
getPage(8).thenApplyAsync(countImages(page)).thenApply(List::size)
```

**Task 3**
```
getPage(8).thenComposeAsync(crawlHyperLinks(page))
```

**Task 4**
```
getPage(1).thenCombine(getPage(8), (imgNum1, imgNum2) -> Integer::sum)
```

---

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html)
Overview of Parallel Functional Programming Frameworks

- Modern Java provides two parallel functional programming frameworks

1. Parallel streams

2. Completable futures

Count the # of images in a recursively-defined folder structure using many of the asynchronous features defined in the completable future framework

See github.com/douglas craigschmidt/LiveLessons/tree/master/ImageCounter
Overview of Parallel Functional Programming Frameworks

- Modern Java provides two parallel functional programming frameworks
  1. Parallel streams
  2. Completable futures

  - Supports dependent actions that are triggered upon the completion of async operations

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletionStage.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletionStage.html)
Overview of Parallel Functional Programming Frameworks

- Modern Java provides two parallel functional programming frameworks

1. Parallel streams

2. Completable futures

- Supports dependent actions that are triggered upon the completion of async operations
- Async operations are a model of concurrent programming where the caller doesn’t block waiting for callee to complete

See en.wikipedia.org/wiki/Asynchrony_(computer_programming)
Modern Java provides two parallel functional programming frameworks:

1. Parallel streams

2. Completable futures

- Supports dependent actions that are triggered upon the completion of async operations
- Can also leverage the Java common fork-join pool

Overview of Parallel Functional Programming Frameworks

Task 1:
```
getPage
  .supplyAsync(getStartPage())
```

Task 2:
```
getPage
  .thenApplyAsync(countImages(page))
  .thenApply(List::size)
```

Task 3:
```
getPage
  .thenComposeAsync(crawlHyperLinks(page))
```

Task 4:
```
getPage
  .thenCombine(page, (imgNum1, imgNum2) -> Integer::sum)
```

See [www.baeldung.com/java-completablefuture](http://www.baeldung.com/java-completablefuture)
Overview of Parallel Functional Programming Frameworks

- Modern Java provides two parallel functional programming frameworks
  1. Parallel streams
  2. Completable futures

1. Parallel streams
   - Supports dependent actions that are triggered upon the completion of async operations
   - Can also leverage the Java common fork-join pool

2. Completable futures
   - Supports dependent actions that are triggered upon the completion of async operations
   - Can also leverage the Java common fork-join pool

The completable futures framework supports asynchronous parallel programming.
Evaluating Pros & Cons of Parallel Functional Programming Frameworks
Evaluating Pros & Cons of Parallel Functional Programming Frameworks

- Pros of the parallel functional programming frameworks
Evaluating Pros & Cons of Parallel Functional Programming Frameworks

• Pros of the parallel functional programming frameworks
  • These frameworks perform well on modern multi-core processors, while also enhancing productivity
Pros of the parallel functional programming frameworks

These frameworks perform well on modern multi-core processors, while also enhancing productivity

- e.g., they encapsulate the object-oriented fork-join framework with parallel functional programming façades

See [en.wikipedia.org/wiki/Facade_pattern](en.wikipedia.org/wiki/Facade_pattern)
Evaluating Pros & Cons of Parallel Functional Programming Frameworks

- Pros of the parallel functional programming frameworks
  - These frameworks perform well on modern multi-core processors, while also enhancing productivity
  - Explicit synchronization and/or threading is rarely needed when applying these frameworks

Alleviates many accidental & inherent complexities of concurrency/parallelism
Evaluating Pros & Cons of Parallel Functional Programming Frameworks

- Cons of the parallel functional programming frameworks
Evaluating Pros & Cons of Parallel Functional Programming Frameworks

- Cons of the parallel functional programming frameworks
  - Don’t fully integrate streams with asynchrony to achieve goals of the reactive programming paradigm

See [www.reactivemanifesto.org](http://www.reactivemanifesto.org)
Evaluating Pros & Cons of Parallel Functional Programming Frameworks

- Cons of the parallel functional programming frameworks
  
  - Don’t fully integrate streams with asynchrony to achieve goals of the reactive programming paradigm

Motivates the need for Java reactive streams frameworks, which integrate streams & asynchrony more intentionally

End of How Parallel Programs Are Developed in Java (Part 2)