Recognize the Structure & Functionality of the Java Completable Futures Framework

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 the key principles underlying reactive programming
- Recognize the Java completable futures framework’s structure & functionality

```
Task 1

supplyAsync
  (getStartPage())

Task 2

/imgNum1\ = /page
  .thenApplyAsync
    (countImages(page))
  .thenApply(List::size)

Task 3

/imgNum2\ = /page
  .thenComposeAsync
    (crawlHyperLinks(page))

Task 4

/imgNum1\ .thenCombine(/imgNum2, 
  (imgNum1, imgNum2) -> 
  Integer::sum)
```
Overview of the Java Completable Futures Framework
Overview of the Java Completable Futures Framework

- Java's completable futures framework provides an asynchronous & reactive parallel programming model.

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 the Java Completable Futures Framework

• Java's completable futures framework provides an asynchronous & reactive parallel programming model
• As a baseline, consider a web crawler implementation that’s synchronous

See en.wikipedia.org/wiki/Web_crawler

Step 1: Get start page
Step 2: Count images on the page
Step 3: Count images on all hyperlinked pages
Step 4: Combine results to create the total
Java's completable futures framework provides an asynchronous & reactive parallel programming model.

As a baseline, consider a web crawler implementation that's synchronous.

The time needed to perform all these steps is the sum of each step sequentially.

Overview of the Java Completable Futures Framework

- **Step 1**: Get start page
- **Step 2**: Count images on the page
- **Step 3**: Count images on all hyperlinked pages
- **Step 4**: Combine results to create the total

See en.wikipedia.org/wiki/Sequential_algorithm
Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations.

**Task 1:** Get start page asynchronously

**Task 2:** Count images on the page asynchronously

**Task 3:** Count images on all hyperlinked pages asynchronously

**Task 4:** Combine results to create the total asynchronously

These dependencies can be modeled via a data flow diagram

See [en.wikipedia.org/wiki/Data-flow_diagram](en.wikipedia.org/wiki/Data-flow_diagram)
Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations.

Async operations can be forked, chained, & joined
Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations.

Enables async programming to resemble sync programming via “completion stages”

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletionStage.html
Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations.
- These async operations can run in a thread pool.

See [www.nurkiewicz.com/2013/05/java-8-definitive-guide-to.html](www.nurkiewicz.com/2013/05/java-8-definitive-guide-to.html)
Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations.
  - These async operations can run in a thread pool.
  - Either a (common) fork-join pool or various types of pre- or user-defined thread pools.

```
Task 1
supplyAsync(getStartPage())

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

Task 3
crawlHyperLinks(page).thenComposeAsync(crawlHyperLinks(page))

Task 4
getPage().thenCombine(getPage(), (imgNum1, imgNum2) -> Integer::sum)
```
Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations.
- These async operations can run in a thread pool.
- The time needed to perform these tasks depends on how well tasks can be parallelized.

\[
\text{Speedup}(N) = \frac{1}{(1-P) + \frac{P}{N}}
\]

Serial part of job = 1 (100%) - Parallel part
Parallel part is divided up by N workers

See [en.wikipedia.org/wiki/Amdahl’s_law](http://en.wikipedia.org/wiki/Amdahl’s_law)
Overview of the Java Completable Futures Framework

- The entire Java completable futures framework resides in one public class with 60+ methods

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html
Overview of the Java Completable Futures Framework

- Java completable futures, sequential streams, & functional programming features can be combined nicely!!

See github.com/douglascraigschmidt/LiveLessons/tree/master/ImageStreamGang
Overview of the Java Completable Futures Framework

- Java completable futures often need no explicit synchronization or threading when developing parallel programs!

List of URLs to Download

- map(this::checkUrlCachedAsync)
- map(this::downloadImageAsync)
- flatMap(this::applyFiltersAsync)
- collect(toFuture())
- thenAccept(this::log)

Alleviates many accidental & inherent complexities of parallel programming
Overview of the Java Completable Futures Framework

- Java completable futures often need no explicit synchronization or threading when developing parallel programs!

Java class libraries handle locking needed to protect shared mutable state
End of Recognize the Structure & Functionality of the Java Completable Futures Framework