

Evaluating the Pros 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**



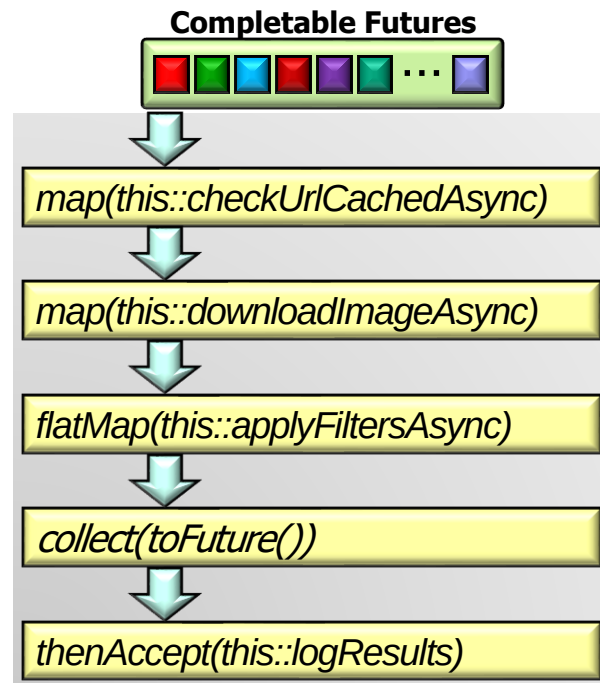
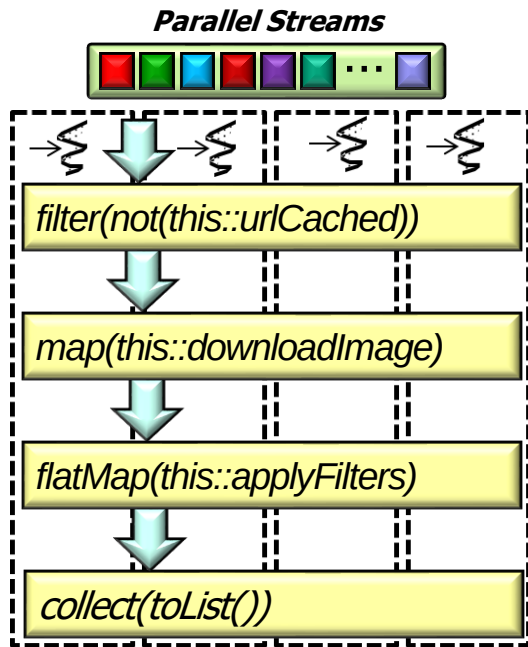
Learning Objectives in this Part of the Lesson

- Evaluate the pros of using the Java completable futures framework



Learning Objectives in this Part of the Lesson

- Evaluate the pros of using the Java completable futures framework
 - We evaluate the Java completable futures framework compared with the parallel streams framework

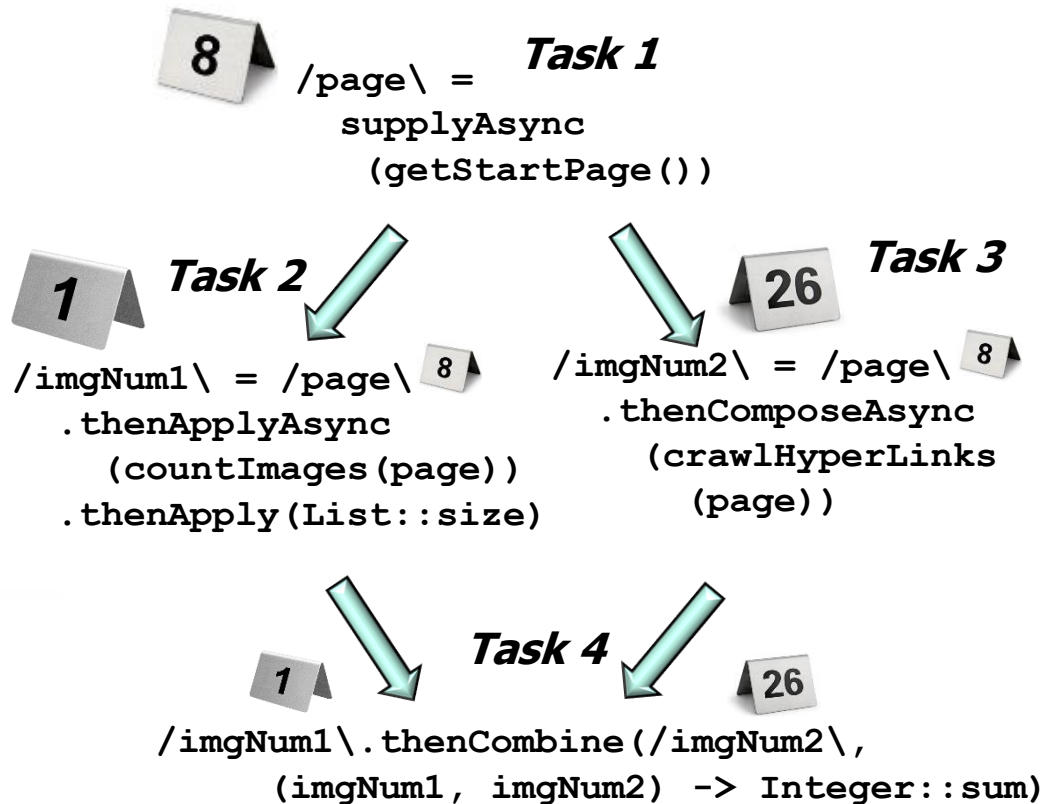


See github.com/douglasraigschmidt/LiveLessons/tree/master/ImageStreamGang

Pros of the Java Completable Futures Framework

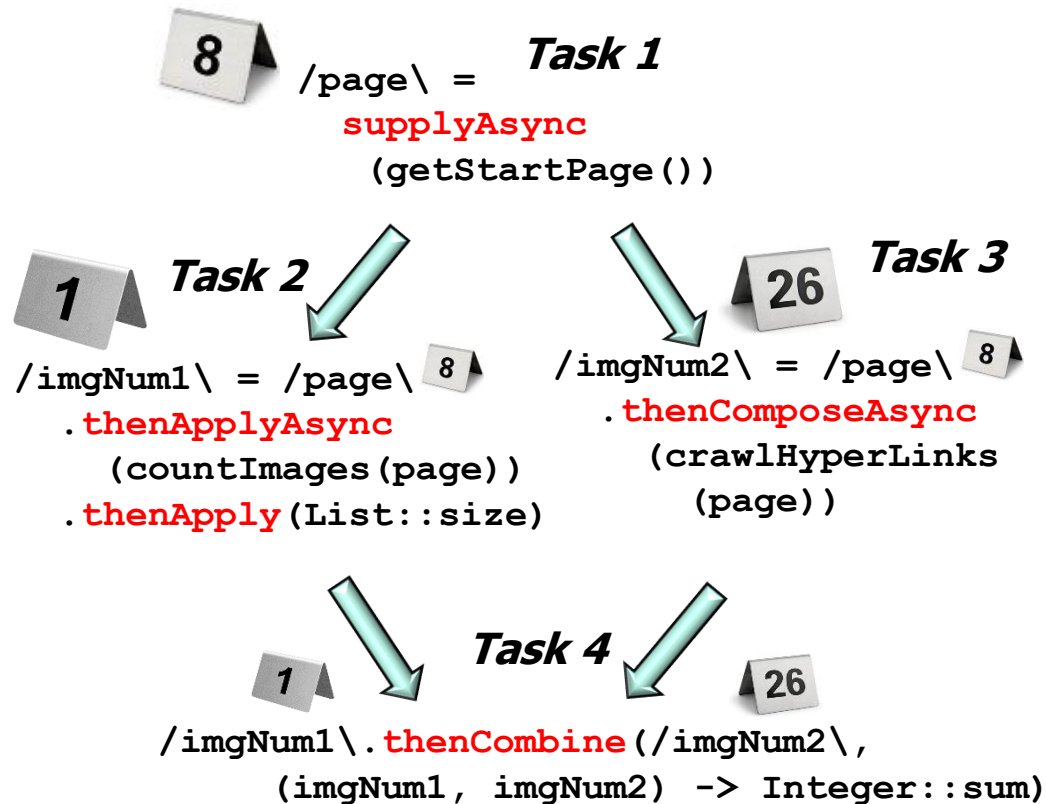
Pros of the Java Completable Futures Framework

- Greatly simplifies programming of asynchronous operations



Pros of the Java Completable Futures Framework

- Greatly simplifies programming of asynchronous operations
- Supports dependent actions that trigger upon completion of async operations

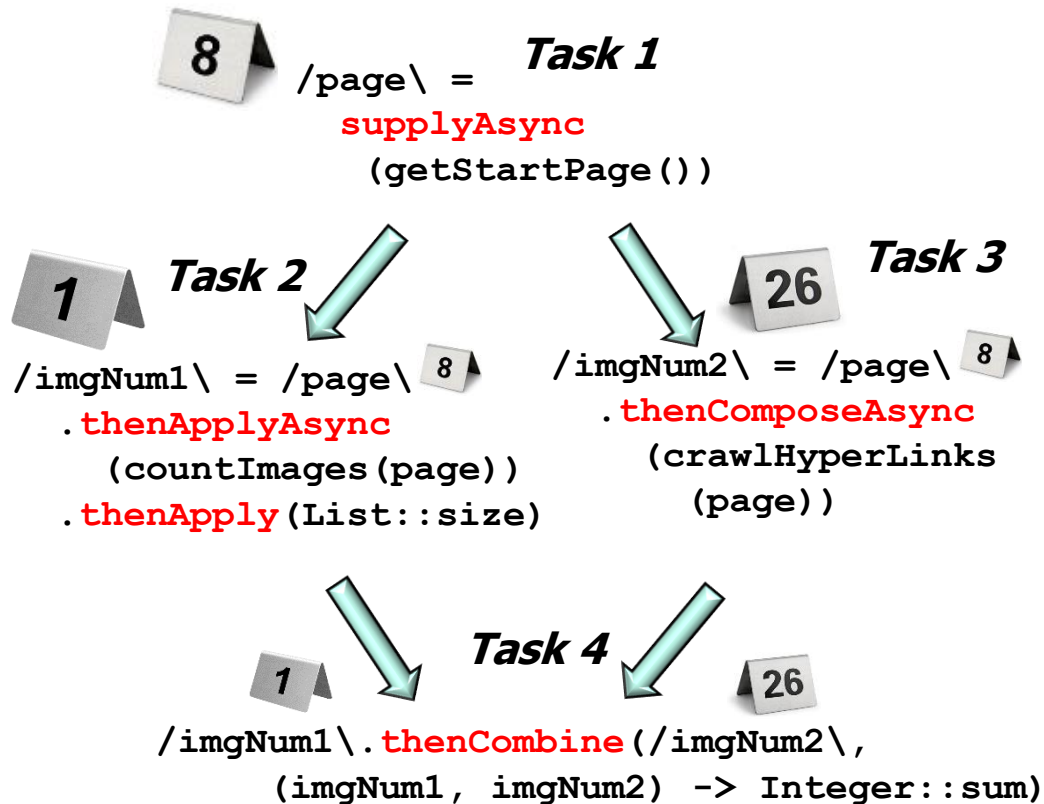


Pros of the Java Completable Futures Framework

- Greatly simplifies programming of asynchronous operations

- Supports dependent actions that trigger upon completion of async operations

- Async operations can be forked, chained, & joined in a relatively intuitive way



Pros of the Java Completable Futures Framework

- Greatly simplifies programming of asynchronous operations

- Supports dependent actions that trigger upon completion of async operations

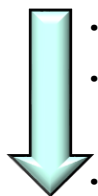
- Async operations can be forked, chained, & joined in a relatively intuitive way

- Enables async programs to appear like sync programs

```
BigFraction unreduced = BigFraction
    .valueOf(new BigInteger
                ("846122553600669882"),
        new BigInteger
                ("188027234133482196"),
        false); // Don't reduce!
```

```
Supplier<BigFraction> reduce = () ->
    BigFraction.reduce(unreduced);
```

```
CompletableFuture
    .supplyAsync(reduce)
    .thenApply(BigFraction
                ::toMixedString)
    .thenAccept(System.out::println);
```

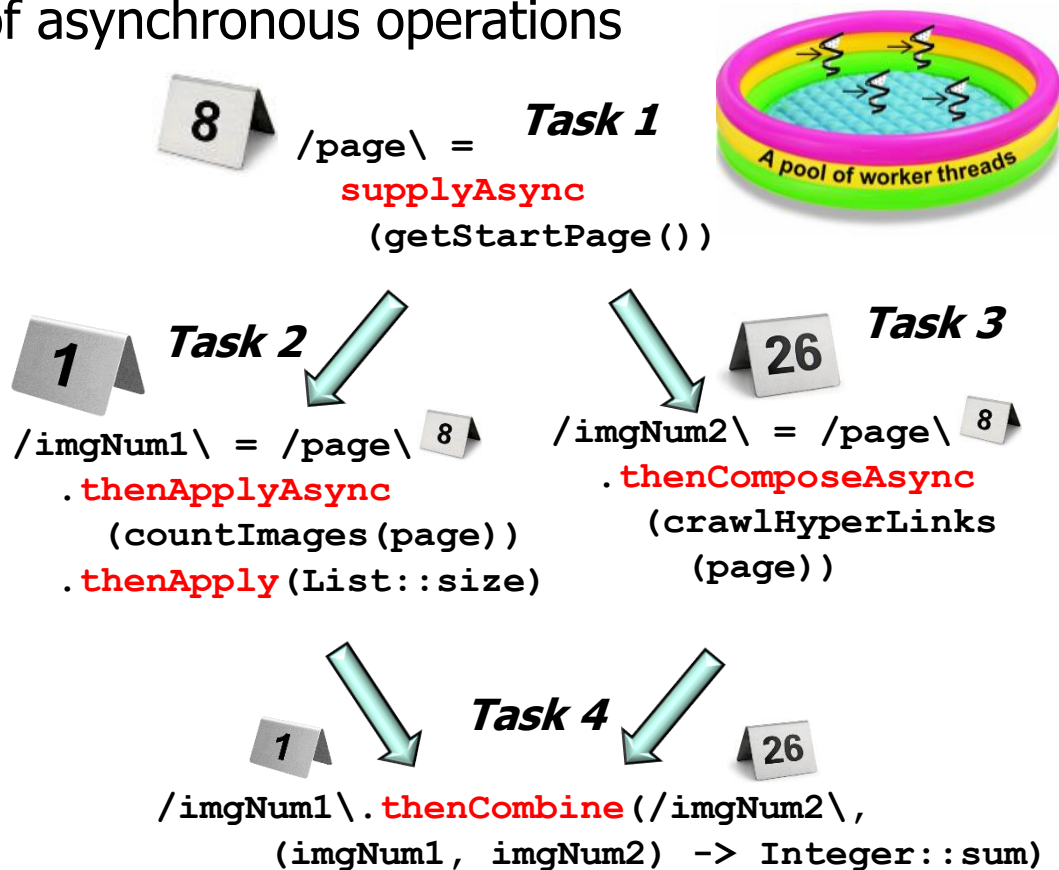


Pros of the Java Completable Futures Framework

- Greatly simplifies programming of asynchronous operations

- Supports dependent actions that trigger upon completion of async operations

- Async operations run in parallel in a thread pool



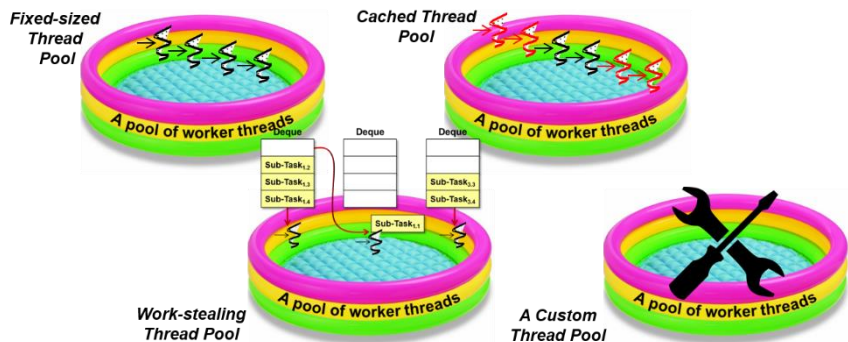
Pros of the Java Completable Futures Framework

- Greatly simplifies programming of asynchronous operations

- Supports dependent actions that trigger upon completion of async operations

- Async operations run in parallel in a thread pool

- Either a (common) fork-join pool or various types of pre- or user-defined thread pools



8 /page\ = **Task 1**
supplyAsync
(getStartPage())



1 **Task 2**
/imgNum1\ = /page\ 8
. **thenApplyAsync**
(countImages(page))
. **thenApply**(List::size)

26 **Task 3**
/imgNum2\ = /page\ 8
. **thenComposeAsync**
(crawlHyperLinks
(page))

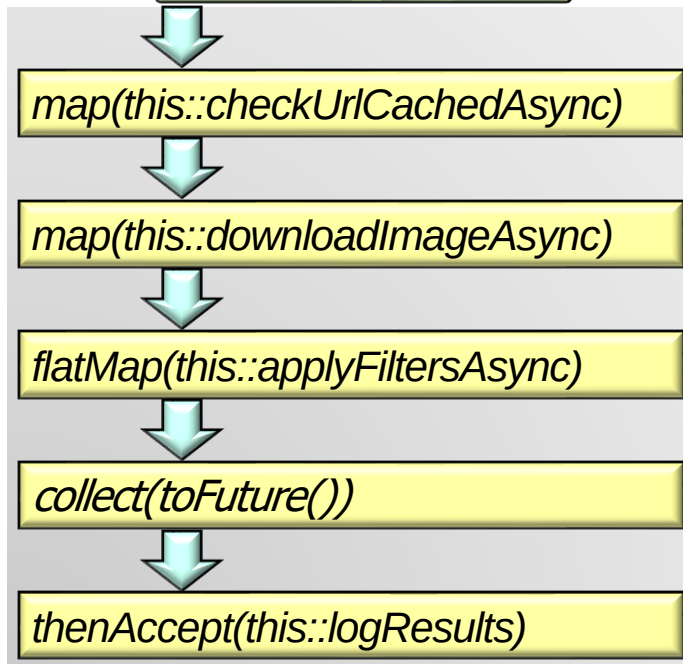
1 **Task 4** 26
/imgNum1\ . **thenCombine** (/imgNum2\
(imgNum1, imgNum2) -> Integer::sum)

Pros of the Java Completable Futures Framework

- No explicit synchronization or threading is required for completable futures

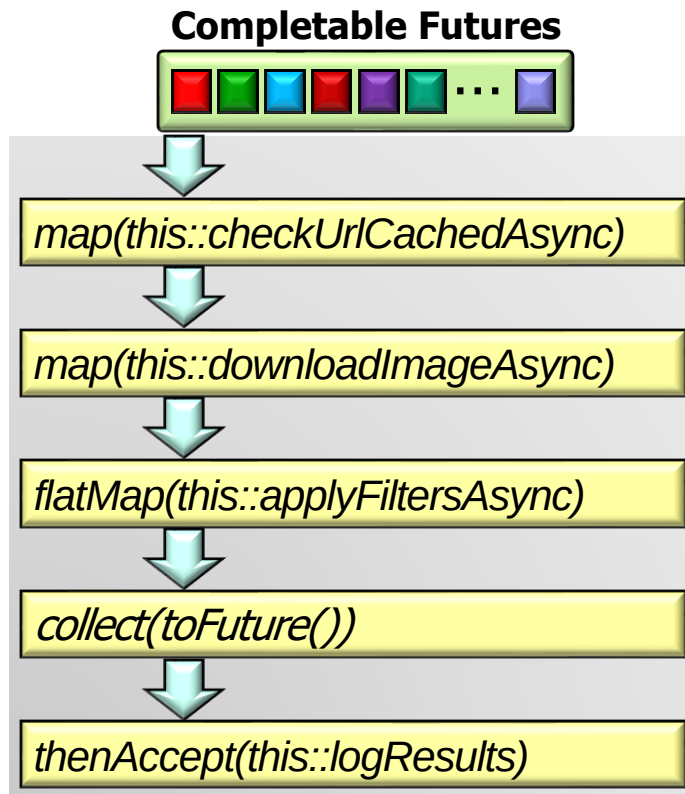
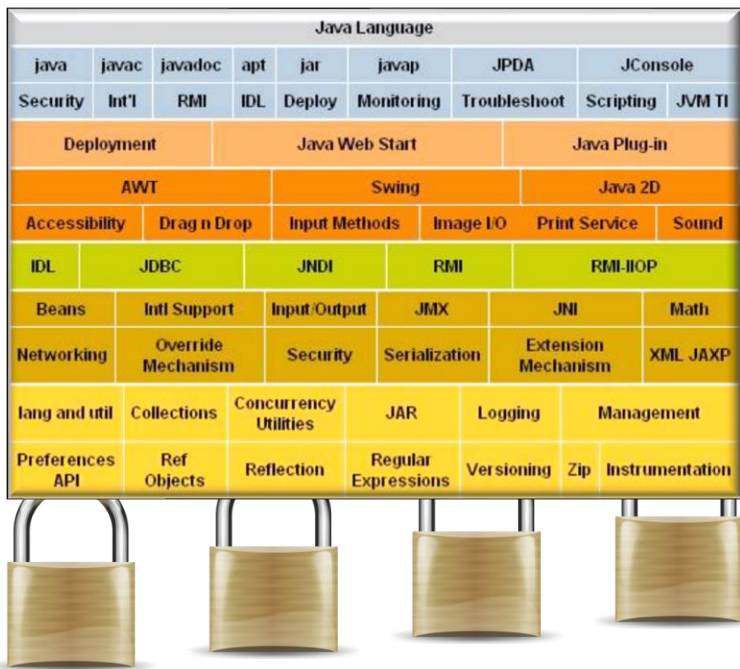


Completable Futures



Pros of the Java Completable Futures Framework

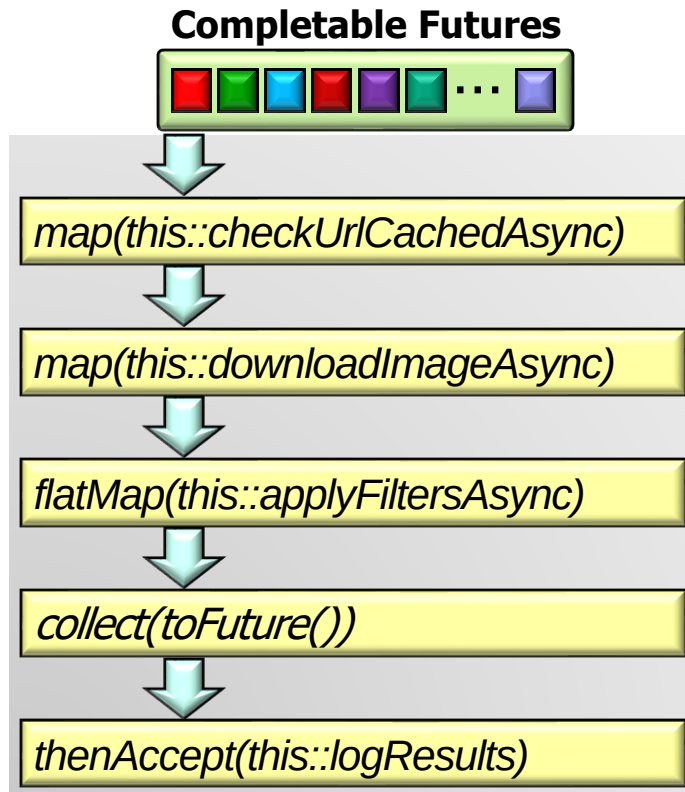
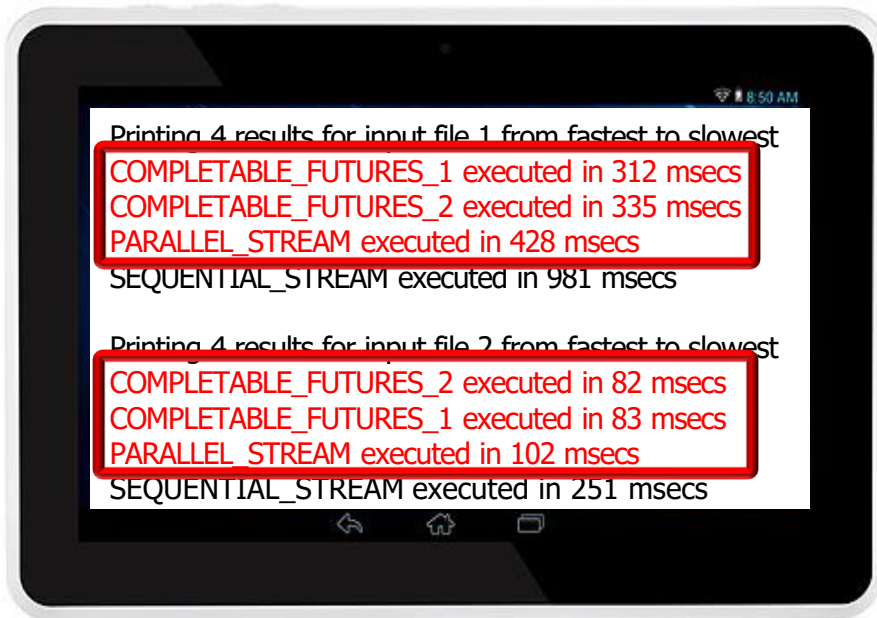
- No explicit synchronization or threading is required for completable futures
- Java libraries handle locking needed to protect shared mutable state



See docs.oracle.com/javase/tutorial/essential/concurrency/collections.html

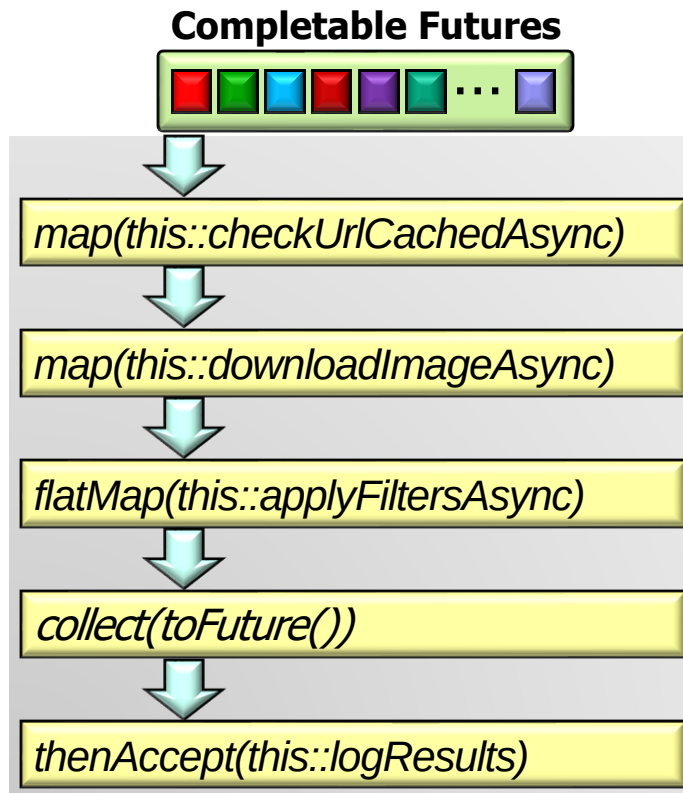
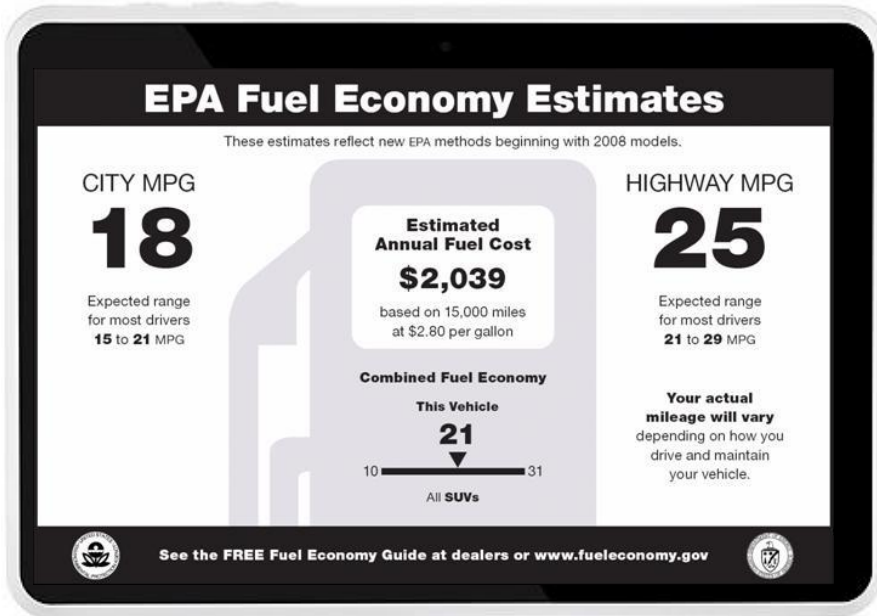
Pros of the Java Completable Futures Framework

- Completable futures are often more efficient than parallel streams



Pros of the Java Completable Futures Framework

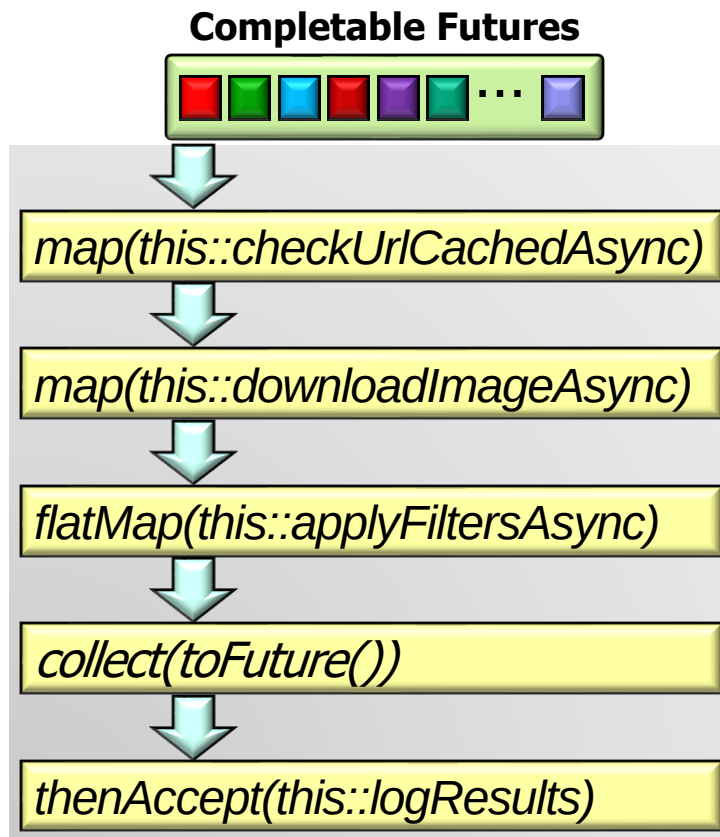
- Completable futures are often more efficient than parallel streams
 - Naturally, your mileage may vary..



There's no substitute for benchmarking, e.g., java-performance.info/jmh!

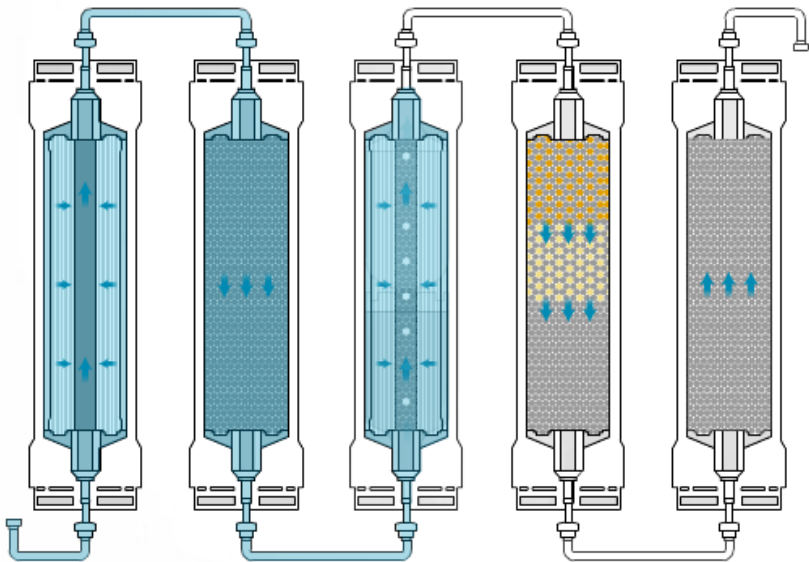
Pros of the Java Completable Futures Framework

- Combining sequential streams & completable futures is often a win

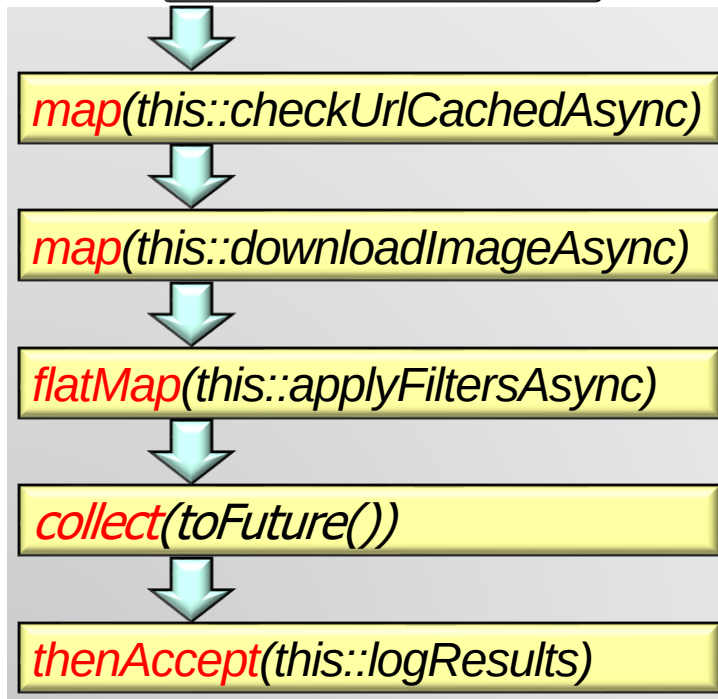


Pros of the Java Completable Futures Framework

- Combining sequential streams & completable futures is often a win
- Streams guide the overall flow of control...

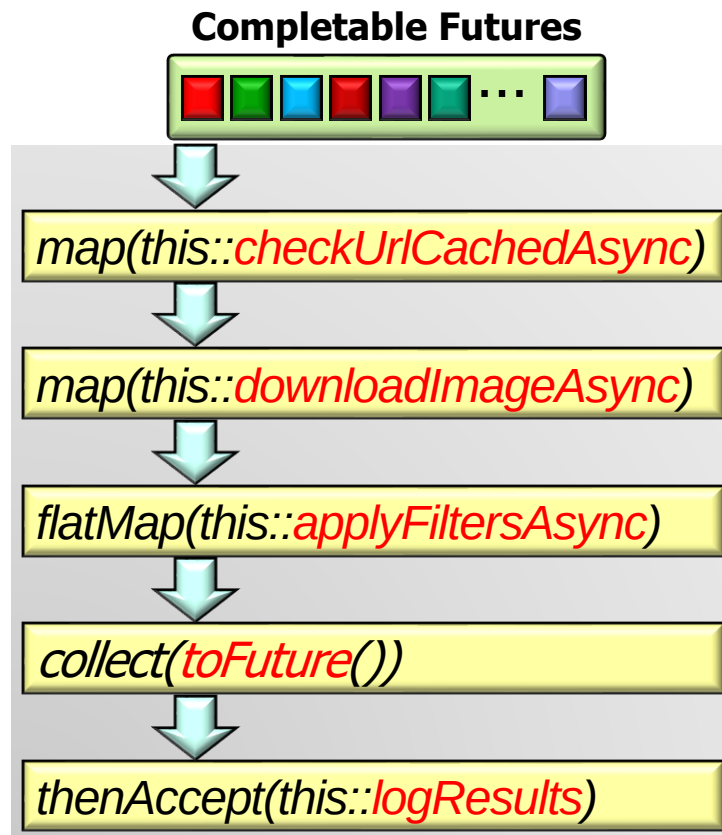


Completable Futures



Pros of the Java Completable Futures Framework

- Combining sequential streams & completable futures is often a win
 - Streams guide the overall flow of control & completable futures perform async operations efficiently in parallel.



Pros of the Java Completable Futures Framework

- Combining sequential streams & completable futures is often a win
 - Streams guide the overall flow of control & completable futures perform async operations efficiently in parallel.
- However, combining parallel streams & completable futures is overkill..



End of Evaluating the Pros of the Java Completable Futures Framework