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

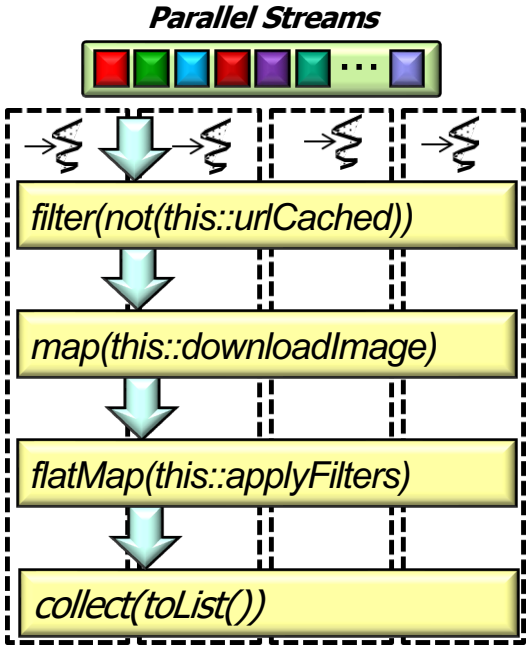
Nashville, Tennessee, USA

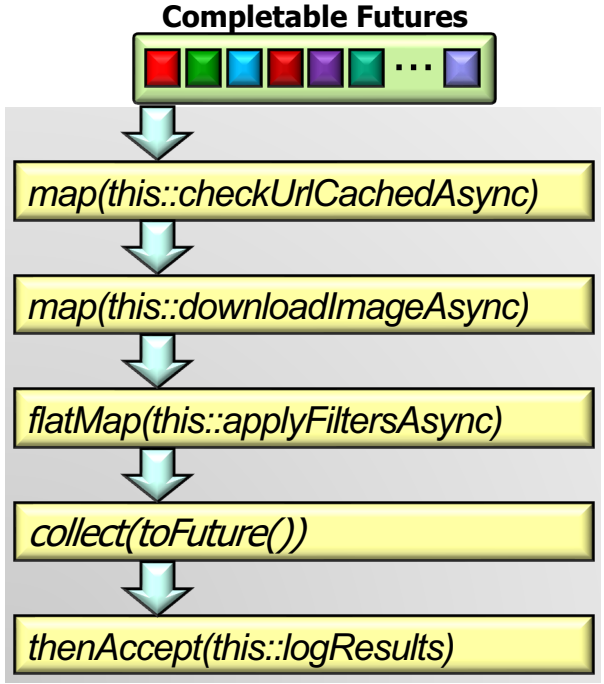
Learning Objectives in this Part of the Lesson

- 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 Java parallel streams framework



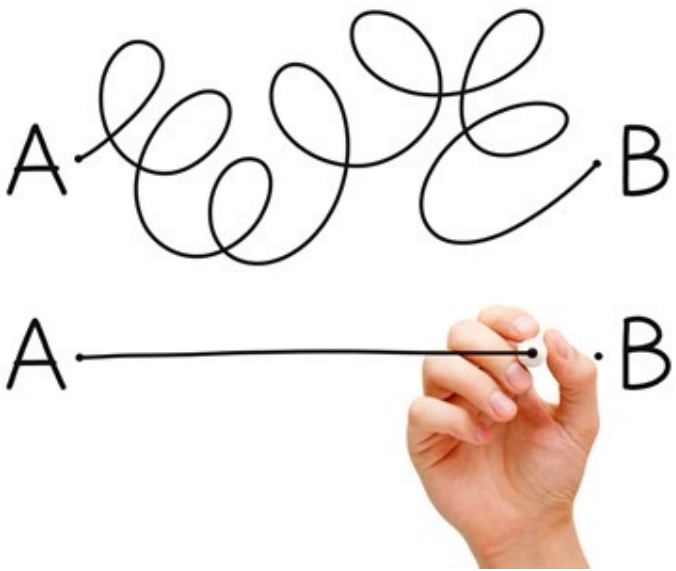


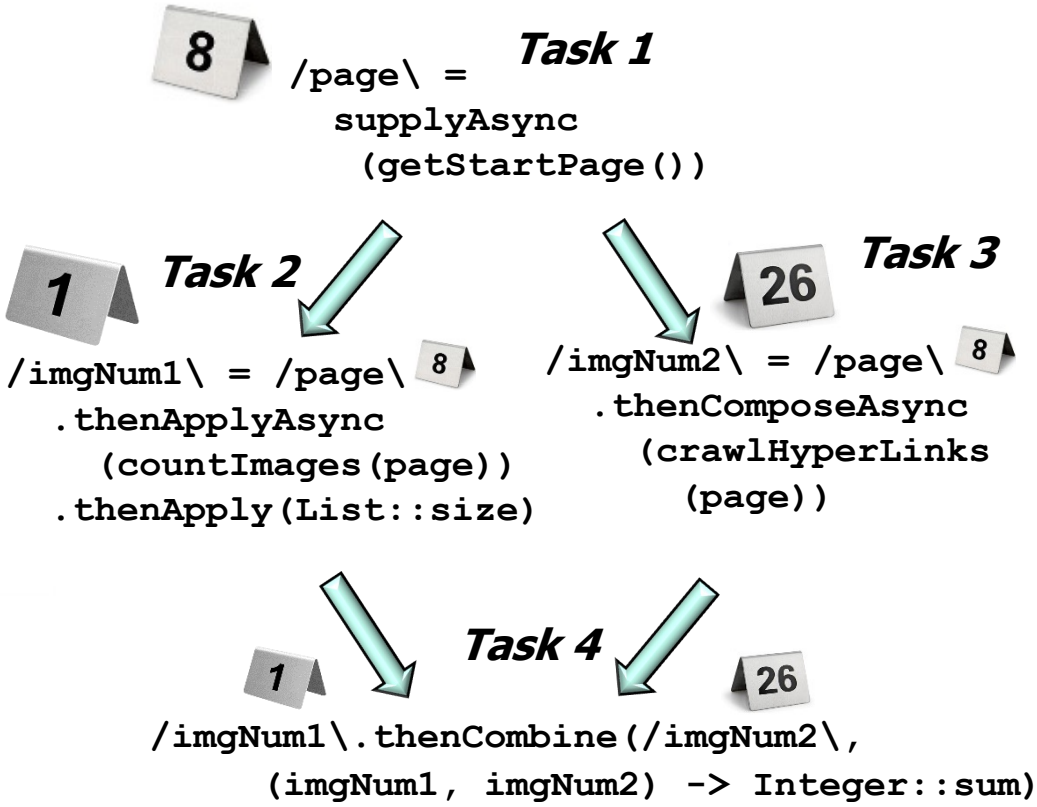
See github.com/douglascraigschmidt/LiveLessons/tree/master/ImageStreamGang

Pros of the Java Completable Futures Framework

Pros of the Java Completable Futures Framework

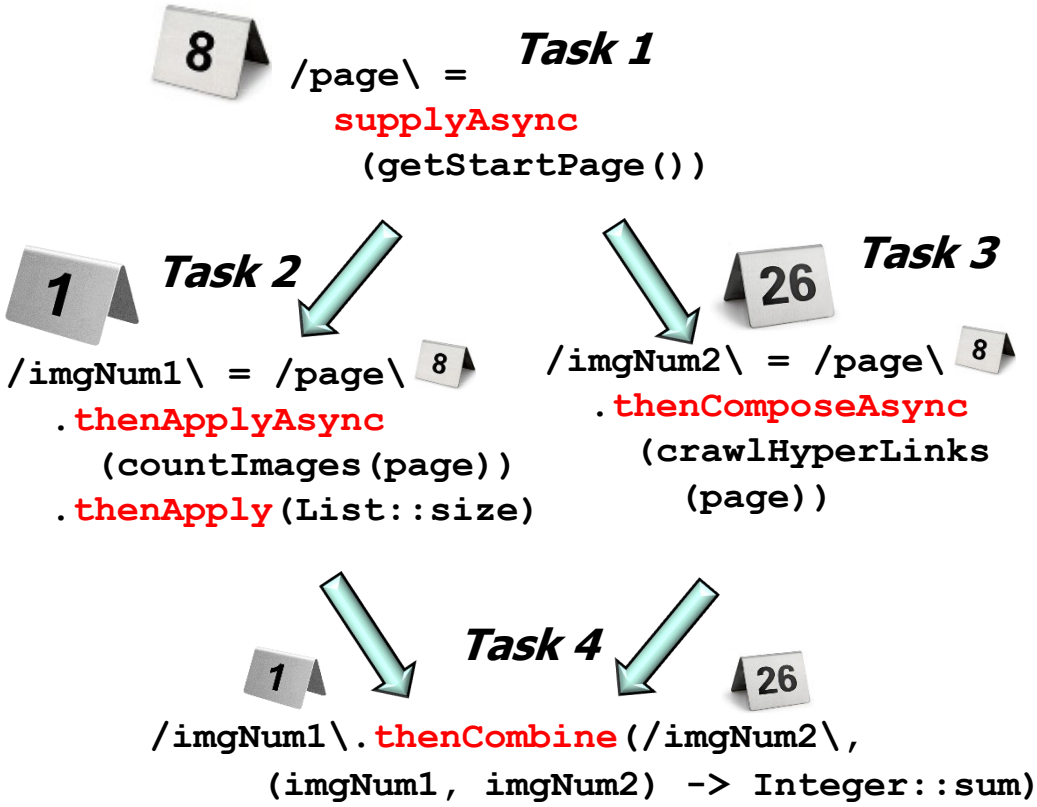
- Greatly simplifies programming of asynchronous operations





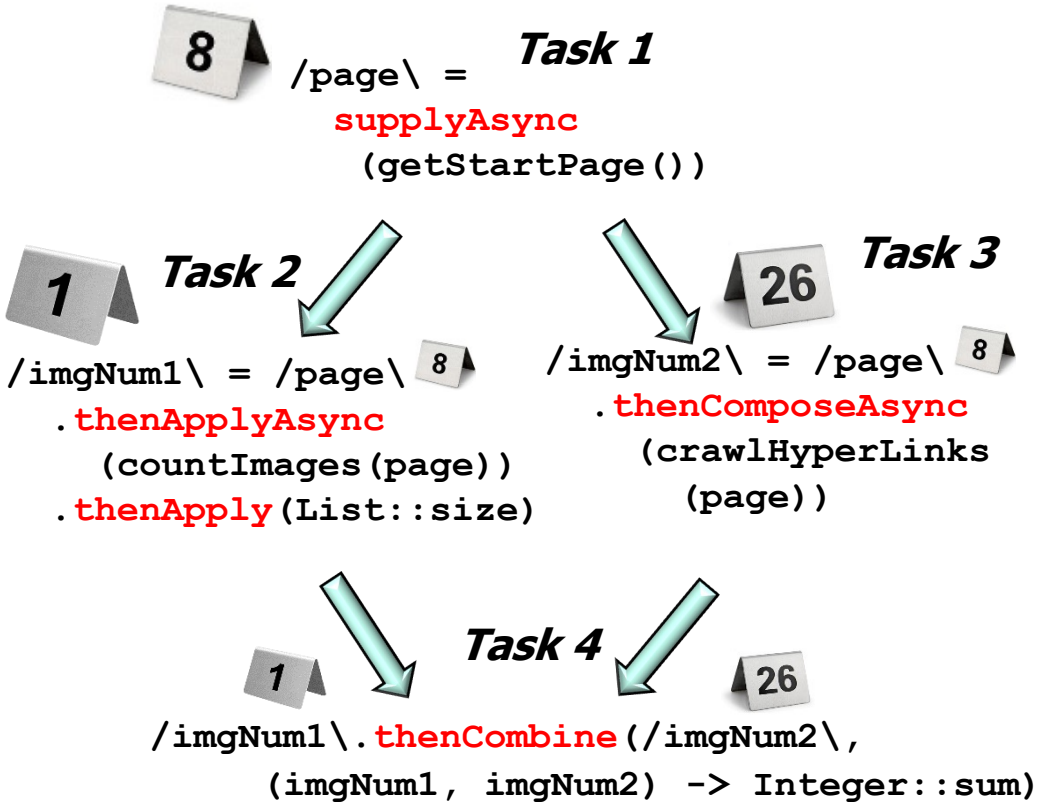
Pros of the Java Completable Futures Framework

- 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
                ("84612255360669882"),
           new BigInteger
                ("18802723413348219 6"),
           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

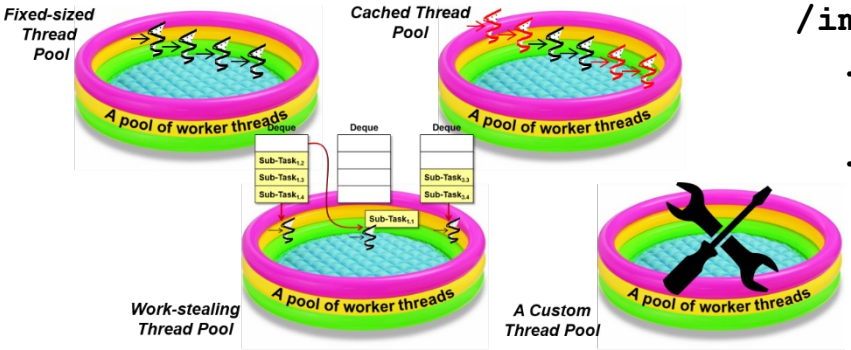
- Also optimizes program performance & scalability

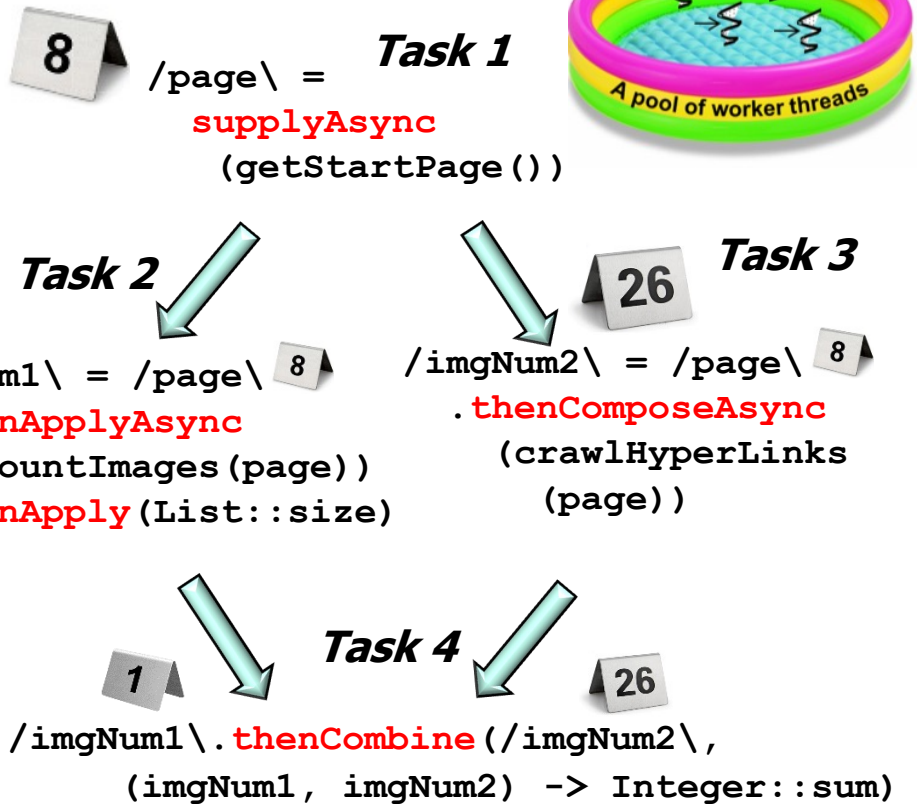
Pros of the Java Completable Futures Framework

- Also optimizes program performance & scalability
  - Async operations run in parallel in a thread pool

Pros of the Java Completable Futures Framework

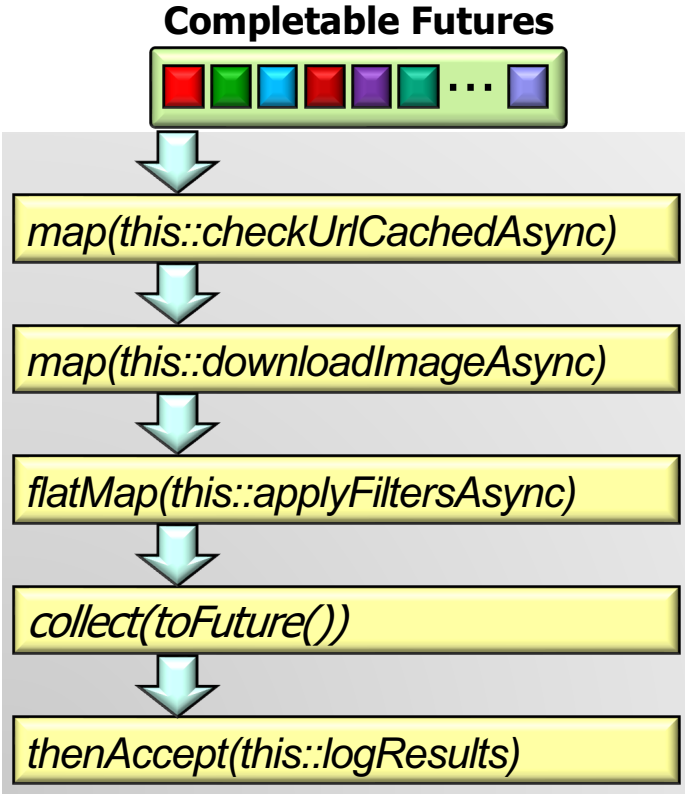
- Also optimizes program performance & scalability
  - Async operations run in parallel in a thread pool
    - Either a (common) fork-join pool or various types of pre- or user-defined thread pools





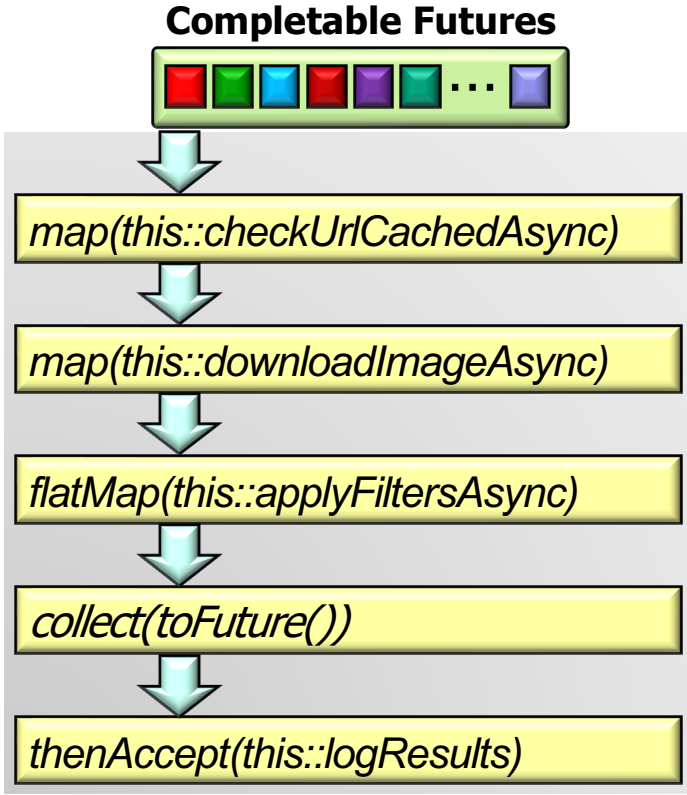
Pros of the Java Completable Futures Framework

- No explicit synchronization or threading is required for completable futures



Pros of the Java Completable Futures Framework

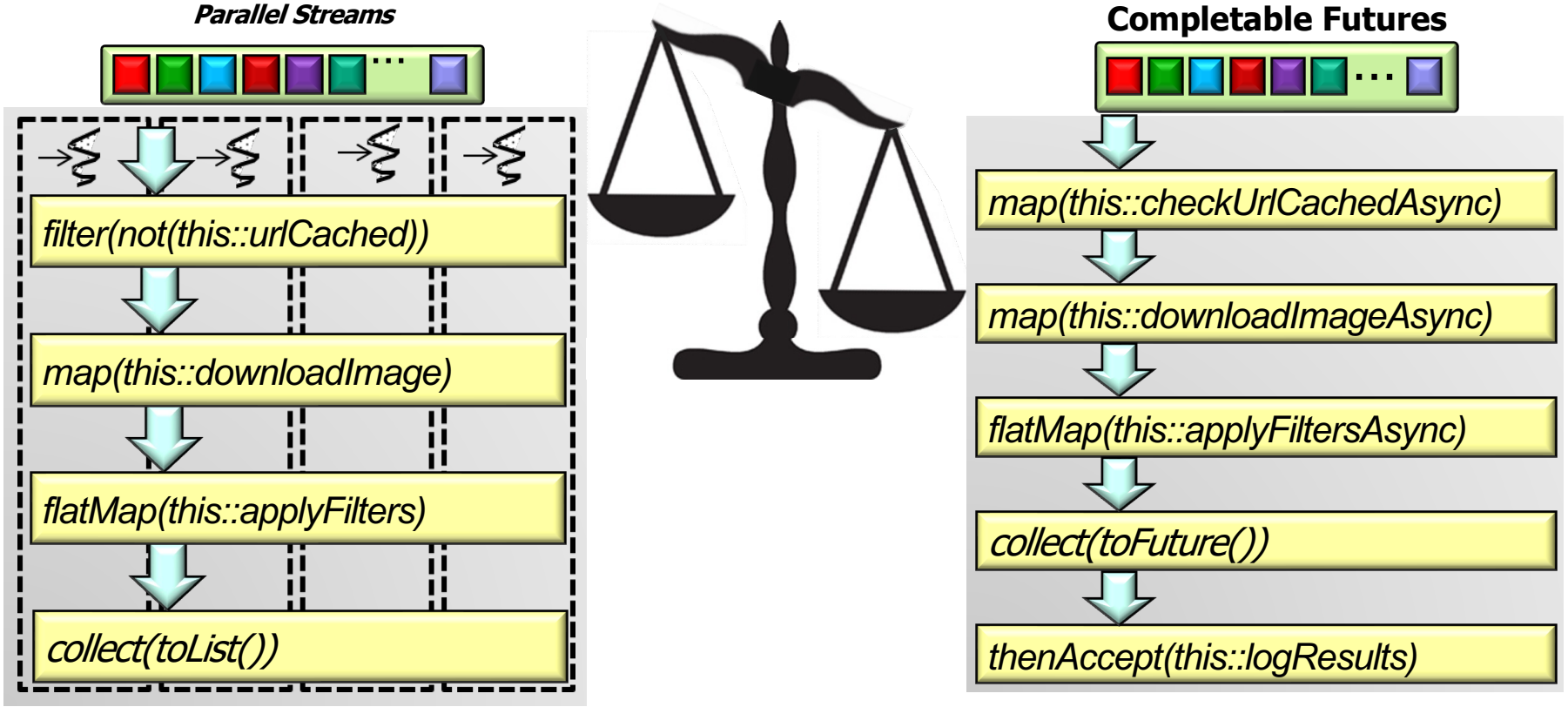
- No explicit synchronization or threading is required for completable futures
  - Java libraries handle locking needed to protect shared mutable state



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

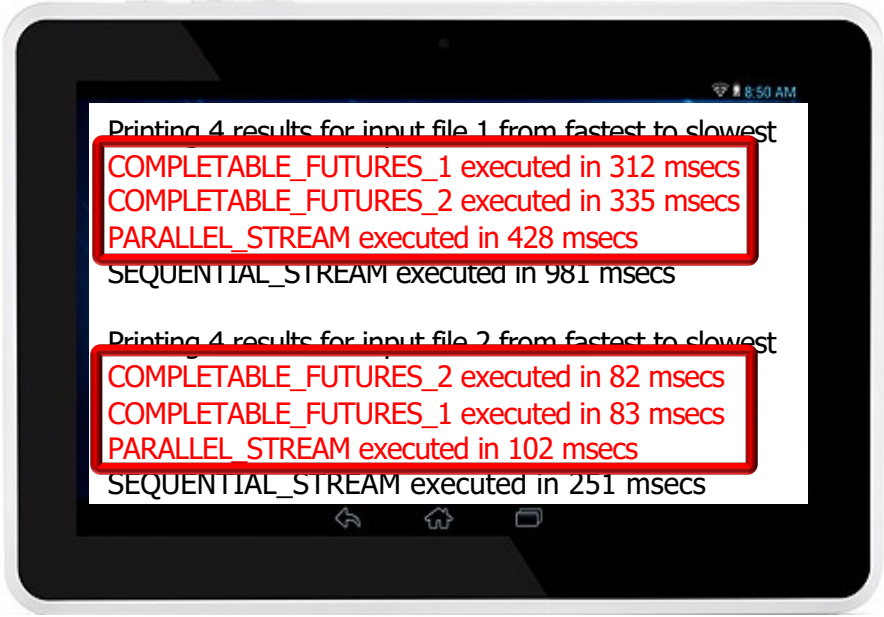
Pros of the Java Completable Futures Framework

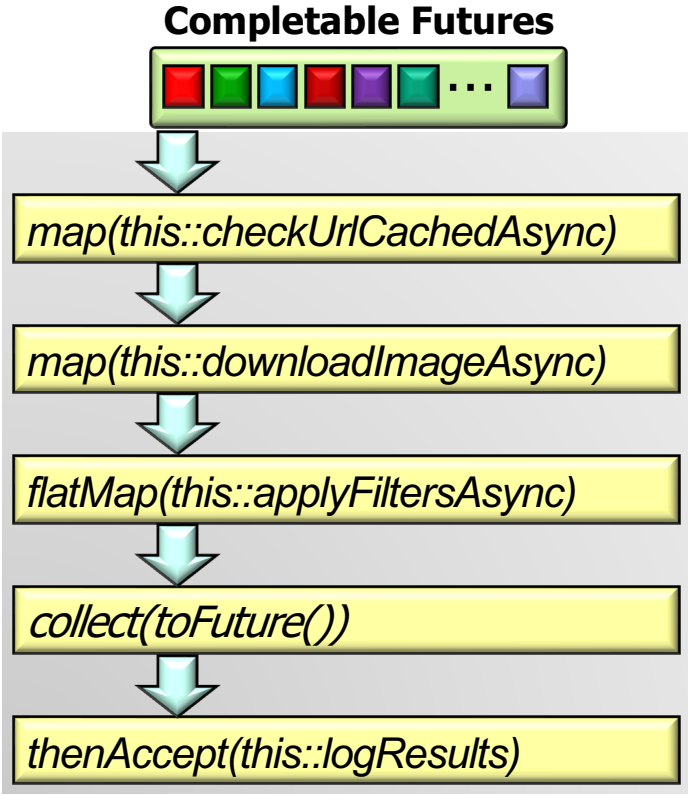
- Completable futures are often more efficient than parallel streams



Pros of the Java Completable Futures Framework

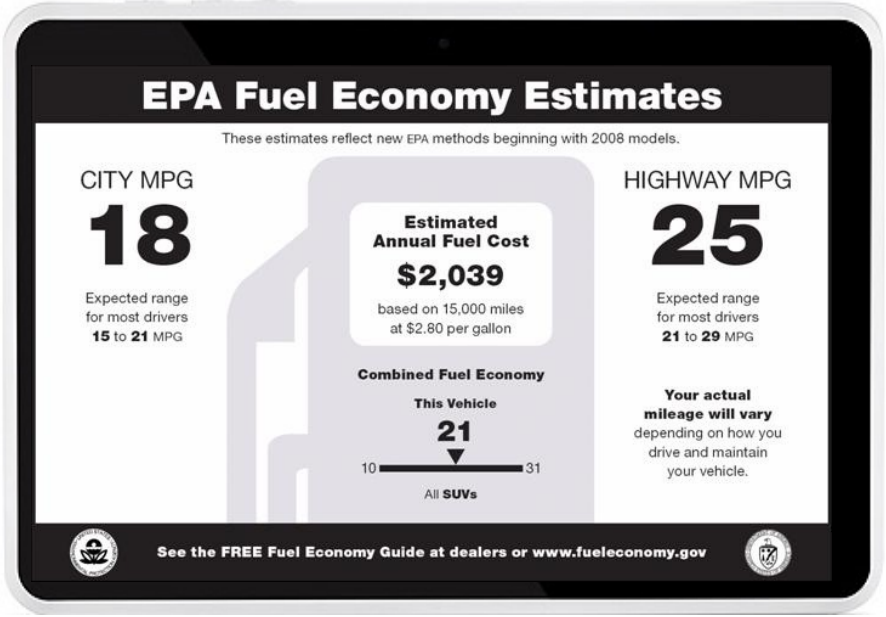
- Completable futures are often more efficient than parallel streams
  - Especially for I/O-bound tasks

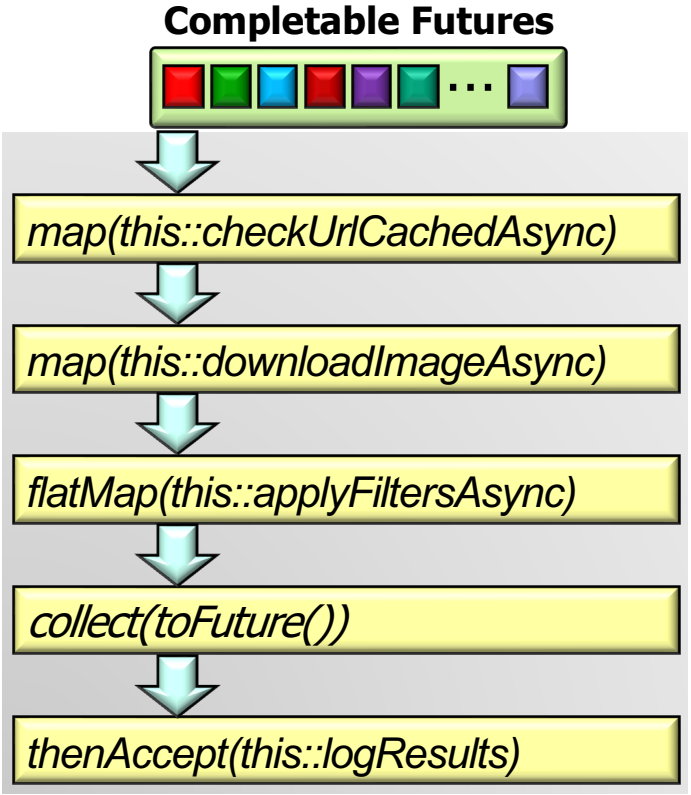




Pros of the Java Completable Futures Framework

- Completable futures are often more efficient than parallel streams
  - Especially for I/O-bound tasks
  - Naturally, your mileage may vary..

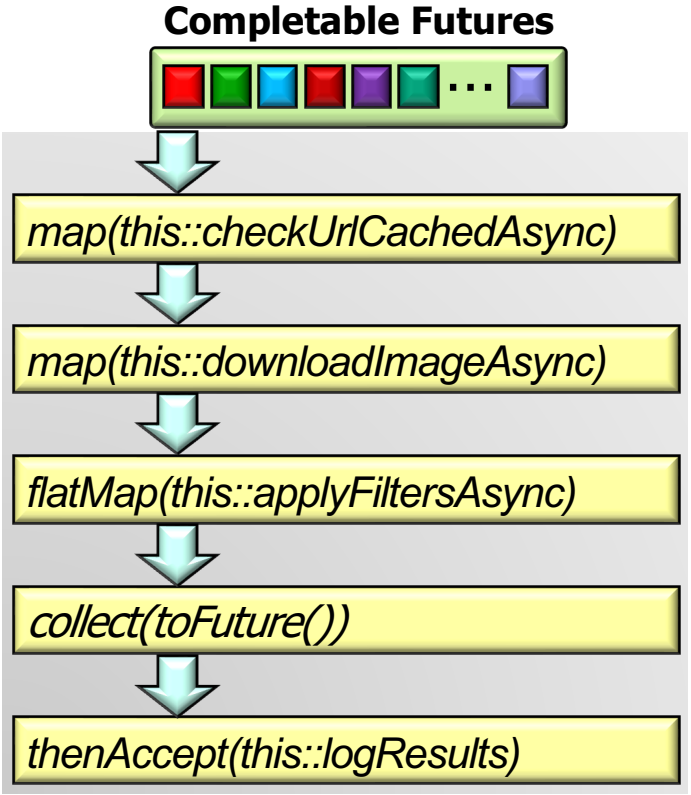




There's no substitute for benchmarking, e.g., java-performance.info/jmh!

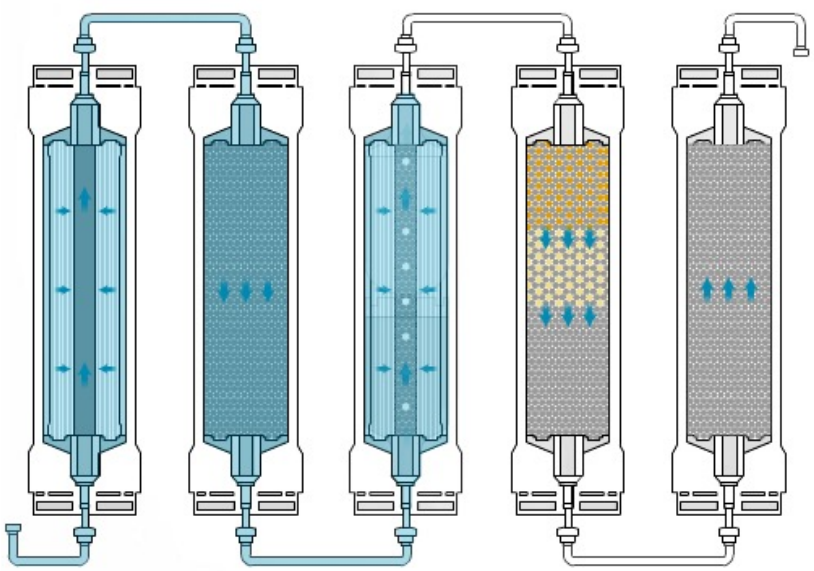
Pros of the Java Completable Futures Framework

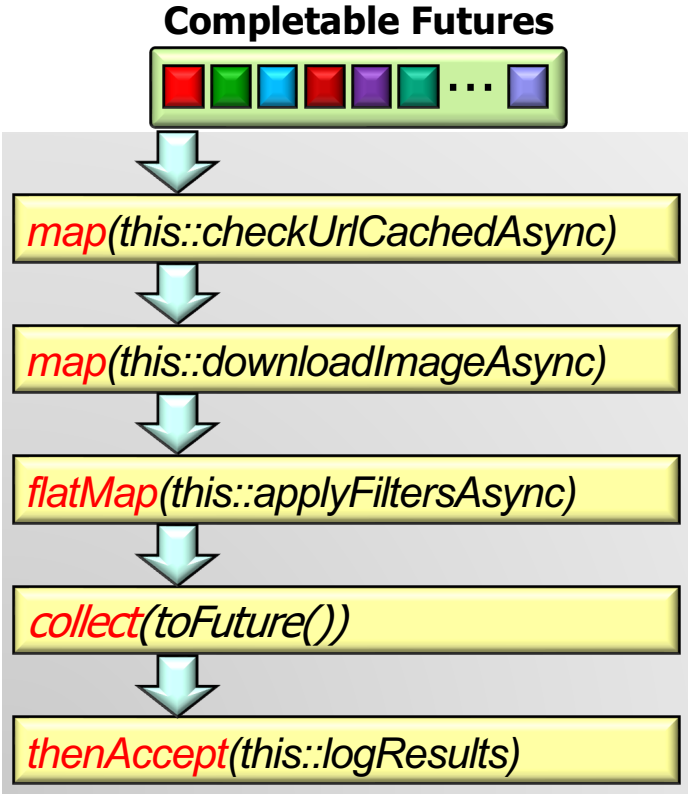
- Combining sequential streams & completable futures is often a win



Pros of the Java Completable Futures Framework

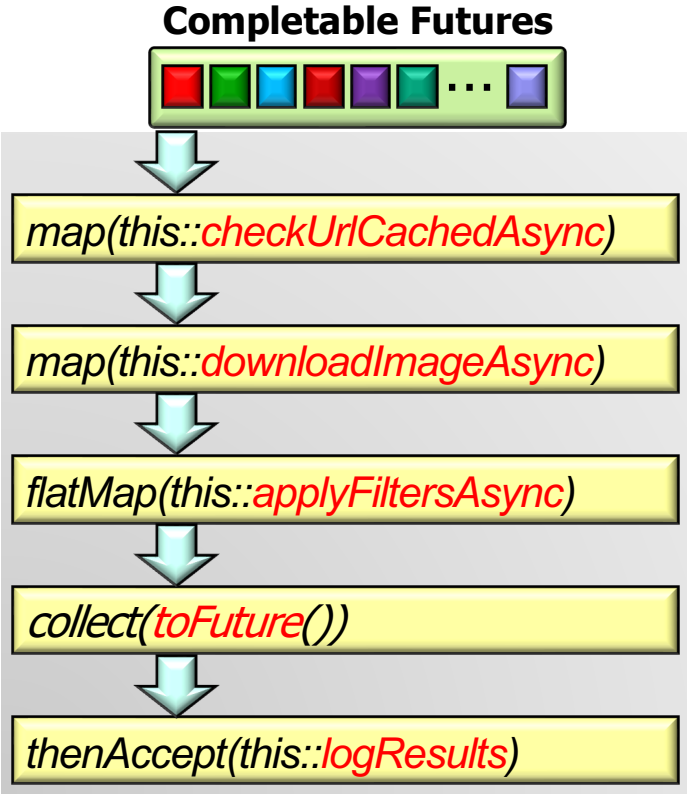
- Combining sequential streams & completable futures is often a win
  - Streams guide the overall flow of control...





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 asynchronous 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 asynchronous operations efficiently in parallel
  - However, combining parallel streams & completable futures is overkill..

End of Evaluating the Pros of the Java Completable Futures Framework