Overview of Parallelism in Java

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

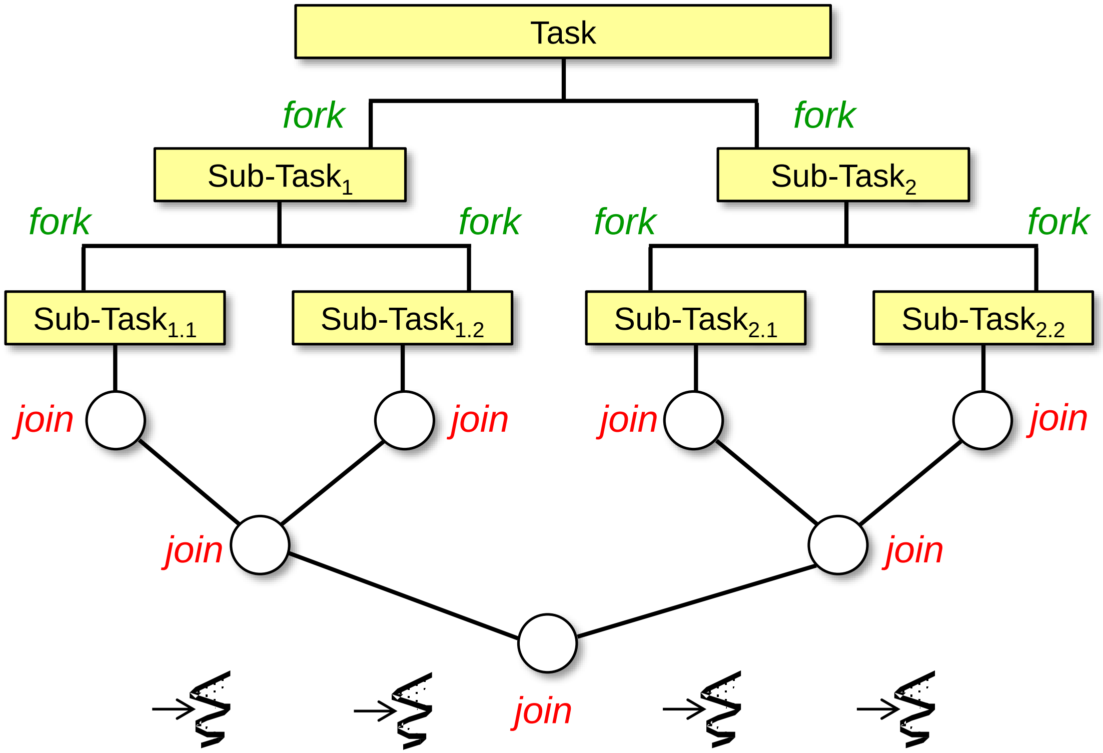
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

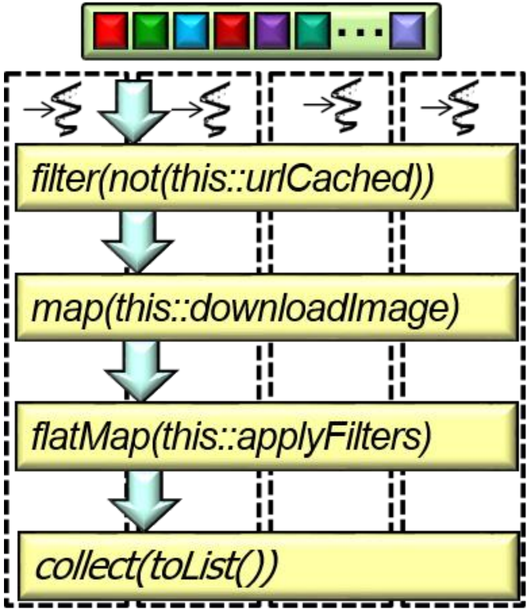
- Understand the meaning of key parallelism concepts



Learning Objectives in this Part of the Lesson

- Understand the meaning of key parallelism concepts
- Recognize how these concepts are supported in Java

Parallel Streams



filter(not(this::urlCached))

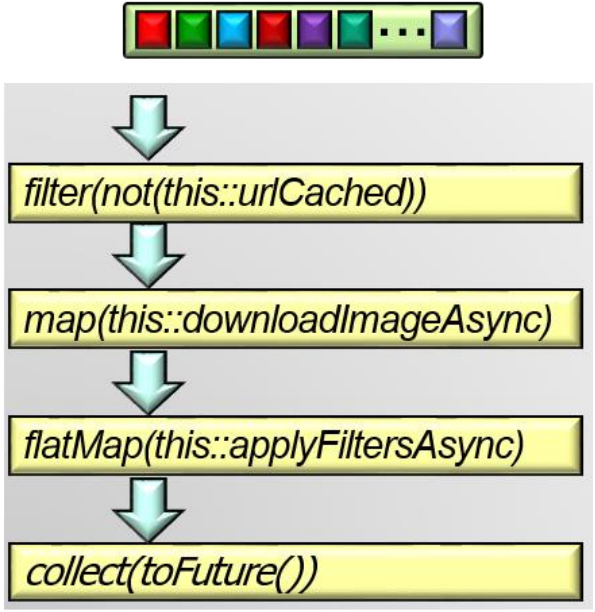
map(this::downloadImage)

flatMap(this::applyFilters)

collect(toList())

ForkJoinPool

Completable Futures



filter(not(this::urlCached))

map(this::downloadImageAsync)

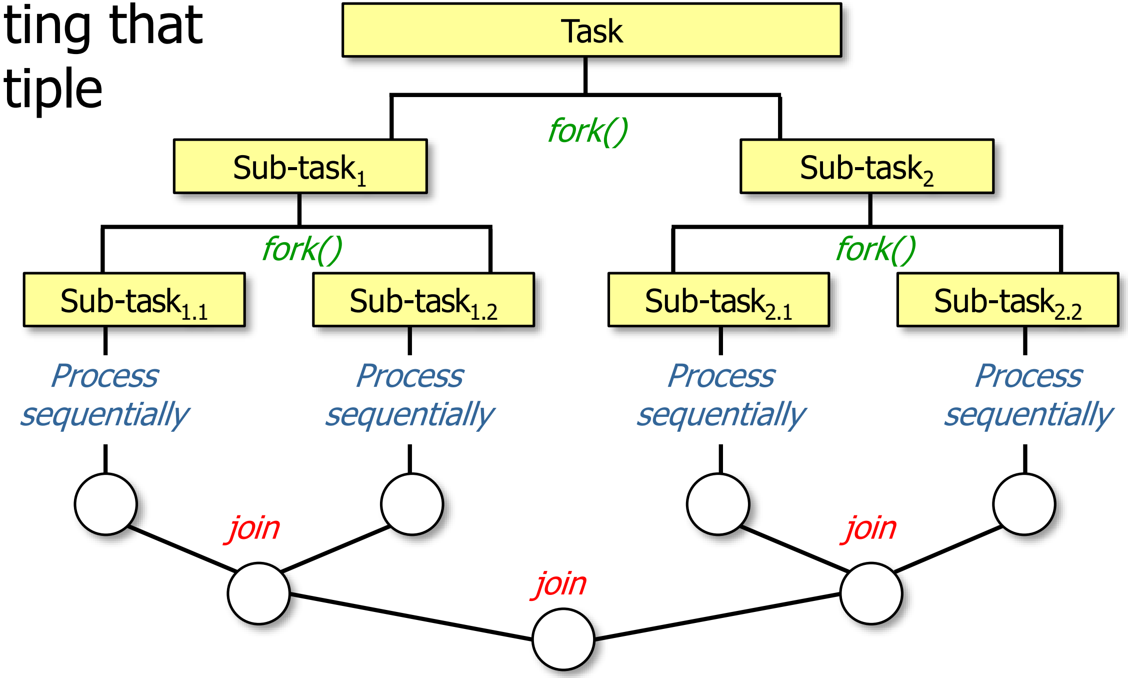
flatMap(this::applyFiltersAsync)

collect(toFuture())

An Overview of Parallelism

An Overview of Parallelism

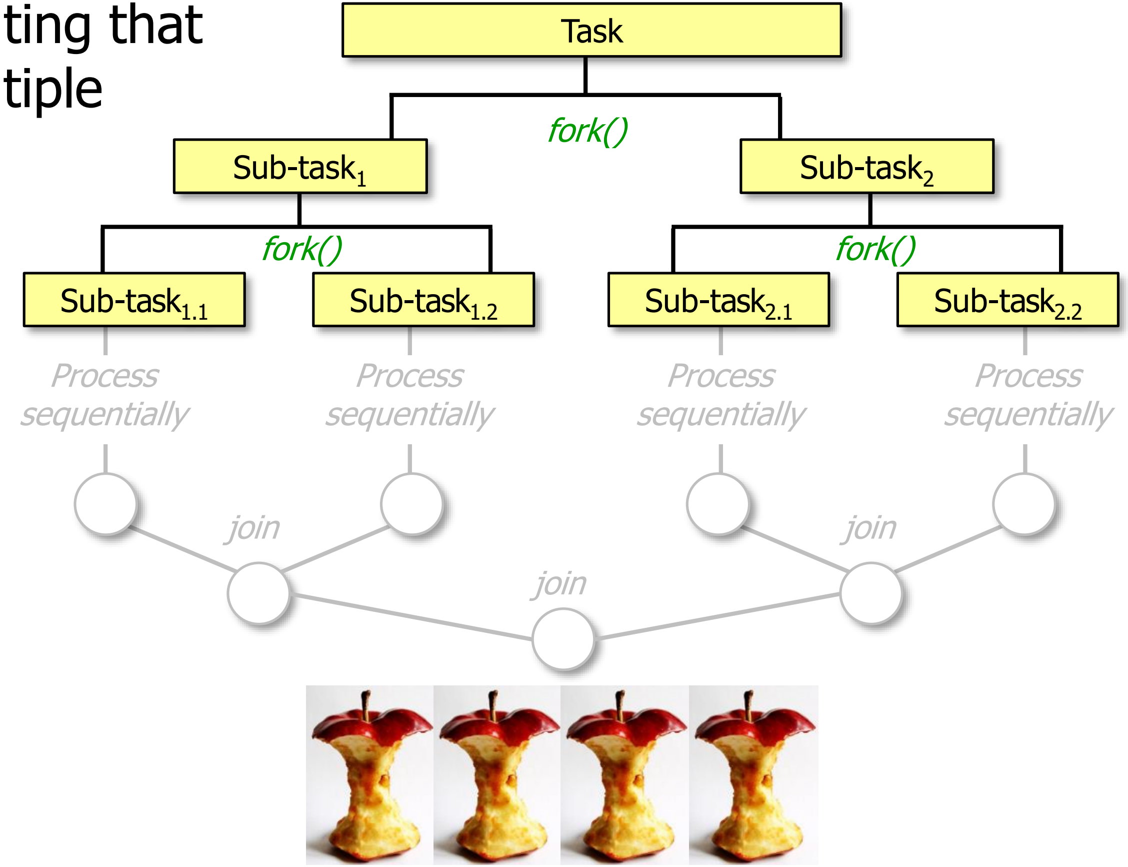
- Parallelism is a form of computing that performs several steps on multiple processor cores



See en.wikipedia.org/wiki/Parallel_computing

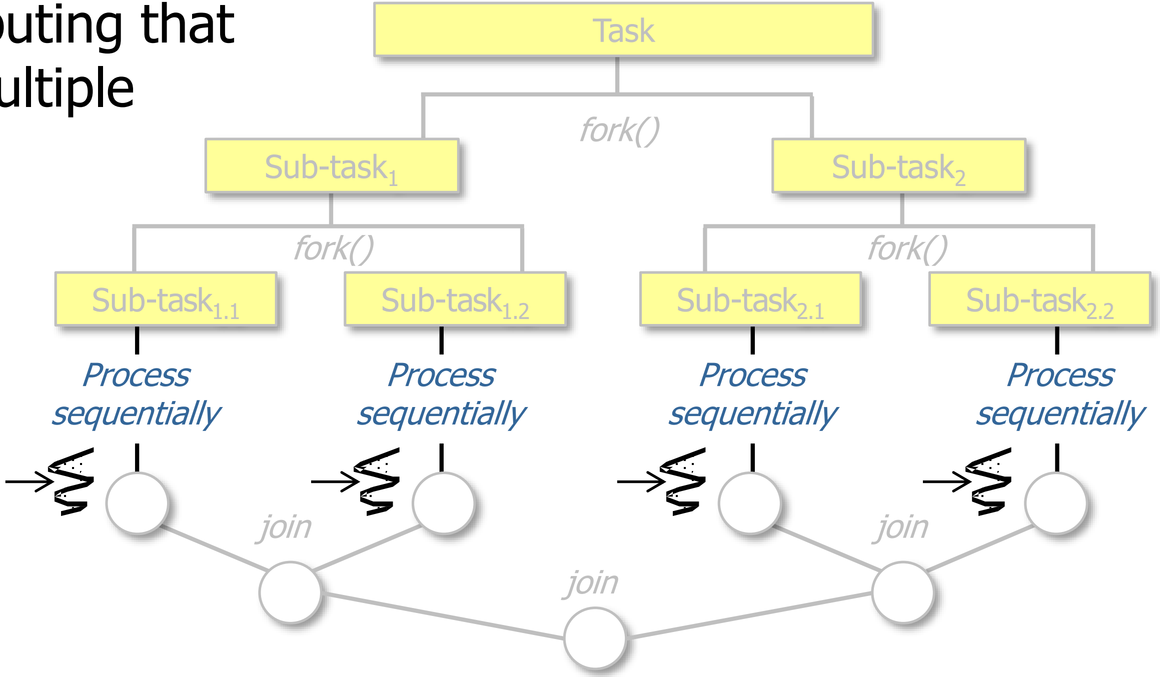
An Overview of Parallelism

- Parallelism is a form of computing that performs several steps on multiple processor cores, i.e.
  - Split – partition a task into sub-tasks



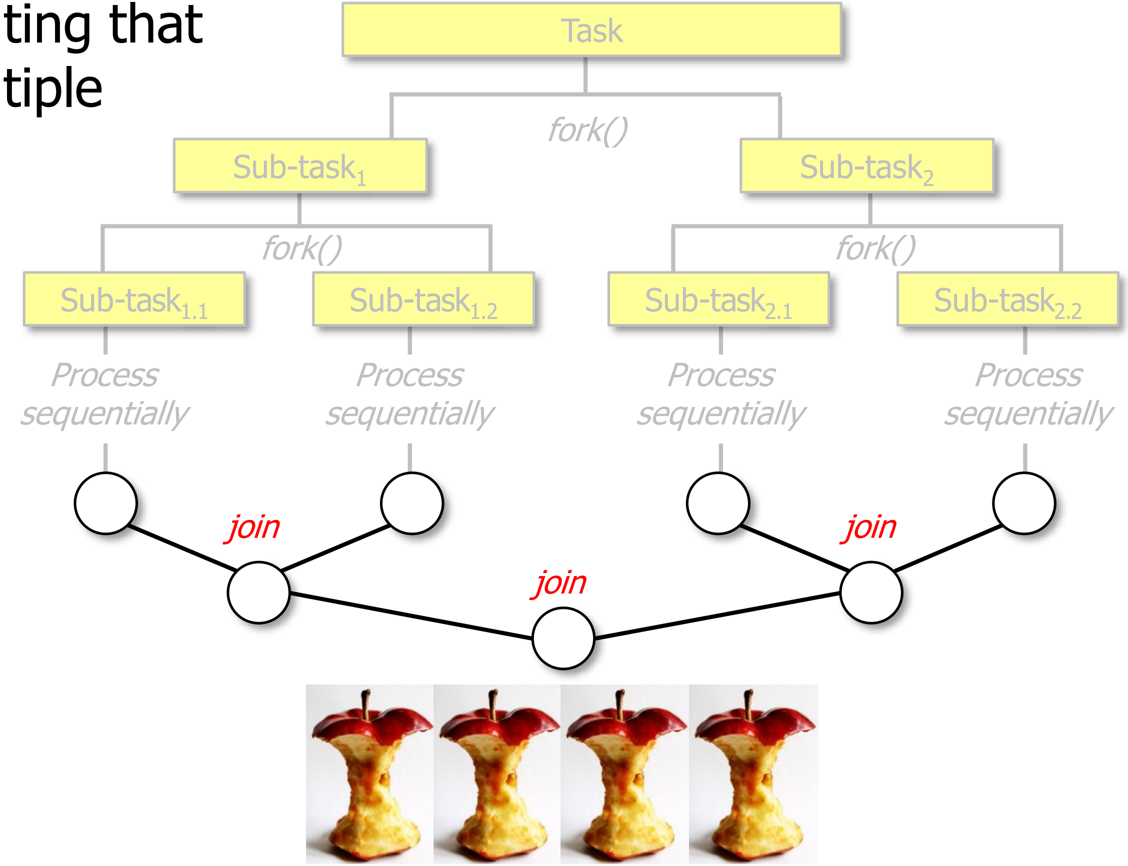
An Overview of Parallelism

- Parallelism is a form of computing that performs several steps on multiple processor cores, i.e.
  - Split – partition a task into sub-tasks
  - Apply – Run independent sub-tasks in parallel



An Overview of Parallelism

- Parallelism is a form of computing that performs several steps on multiple processor cores, i.e.
  - Split – partition a task into sub-tasks
  - Apply – Run independent sub-tasks in parallel
  - Combine – Merge the sub-results from sub-tasks into one final result



An Overview of Parallelism

- A key goal of parallelism is to *efficiently* partition tasks into sub-tasks & combine results

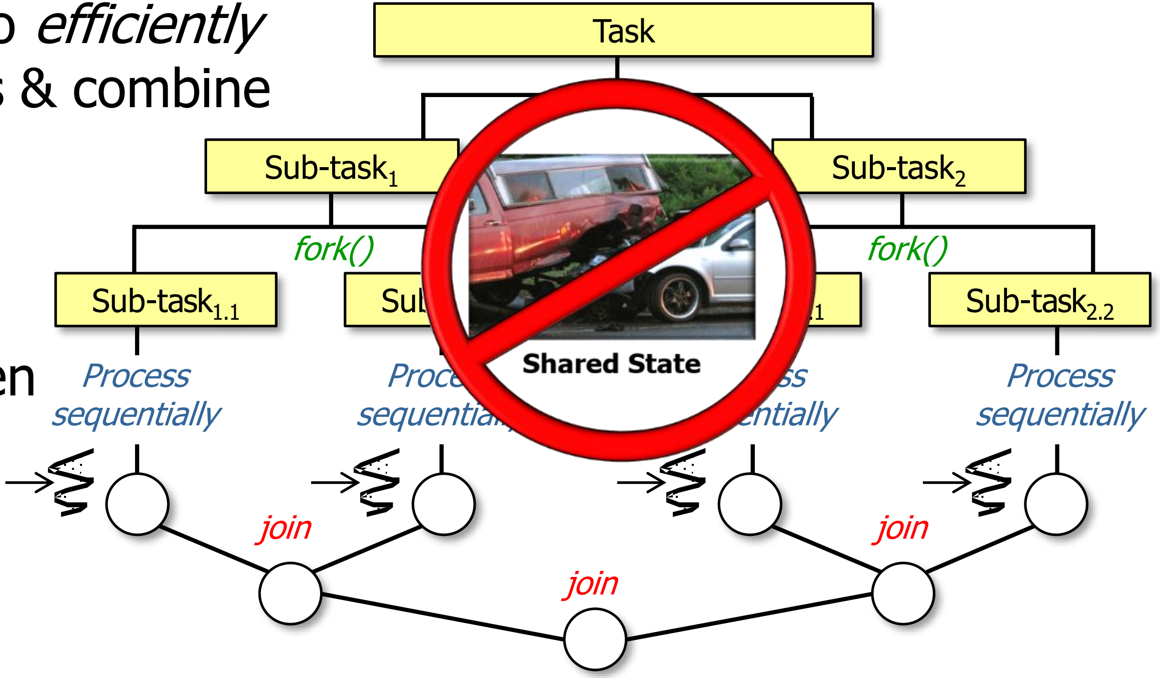
An Overview of Parallelism

- A key goal of parallelism is to *efficiently* partition tasks into sub-tasks & combine results
  - Parallelism thus focuses on optimizing performance
    - e.g., throughput, scalability, & latency

See www.ibm.com/developerworks/library/j-java-streams-4-brian-goetz

An Overview of Parallelism

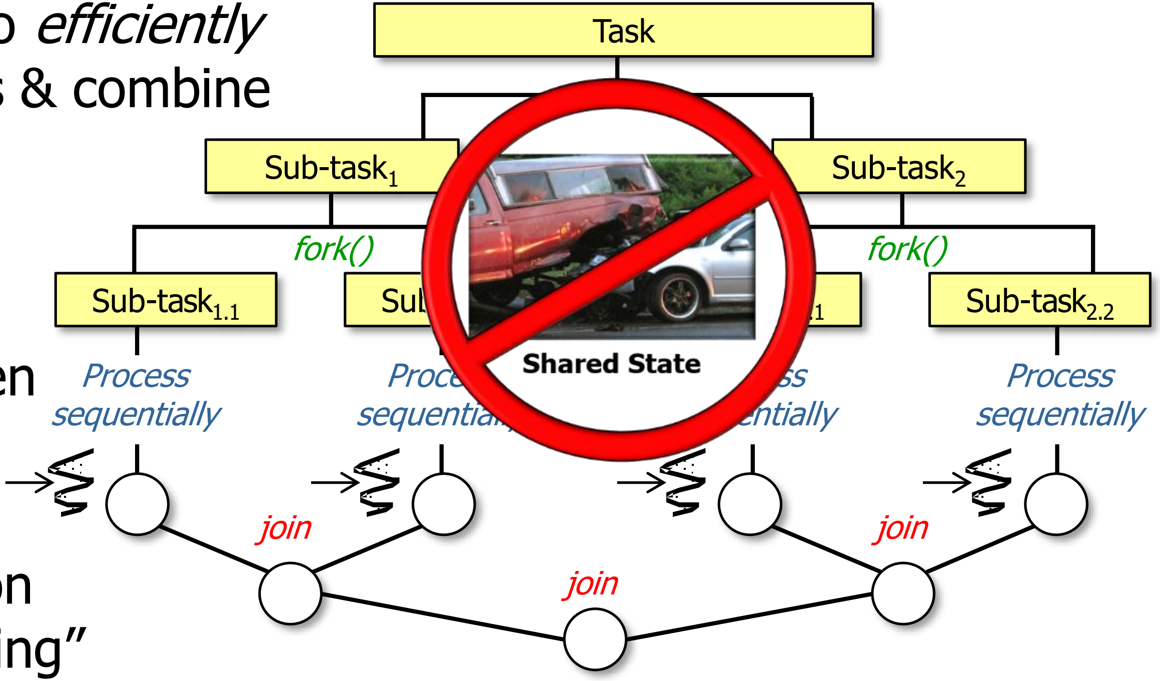
- A key goal of parallelism is to *efficiently* partition tasks into sub-tasks & combine results
  - Parallelism thus focuses on optimizing performance
  - Parallelism works best when threads share no mutable state & don't block



See henrikeichenhardt.blogspot.com/2013/06/why-shared-mutable-state-is-root-of-all.html

An Overview of Parallelism

- A key goal of parallelism is to *efficiently* partition tasks into sub-tasks & combine results
  - Parallelism thus focuses on optimizing performance
  - Parallelism works best when threads share no mutable state & don't block
    - Hence Java's emphasis on "fork-join" & "work-stealing"

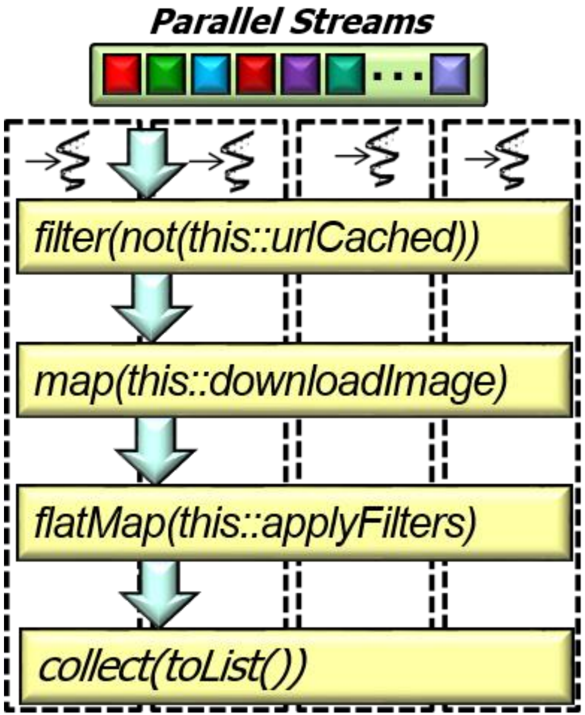


See en.wikipedia.org/wiki/Fork-join_model & en.wikipedia.org/wiki/Work_stealing

An Overview of Parallelism in Java

An Overview of Parallelism in Java

- Java supports parallelism via three primary frameworks



Parallel Streams

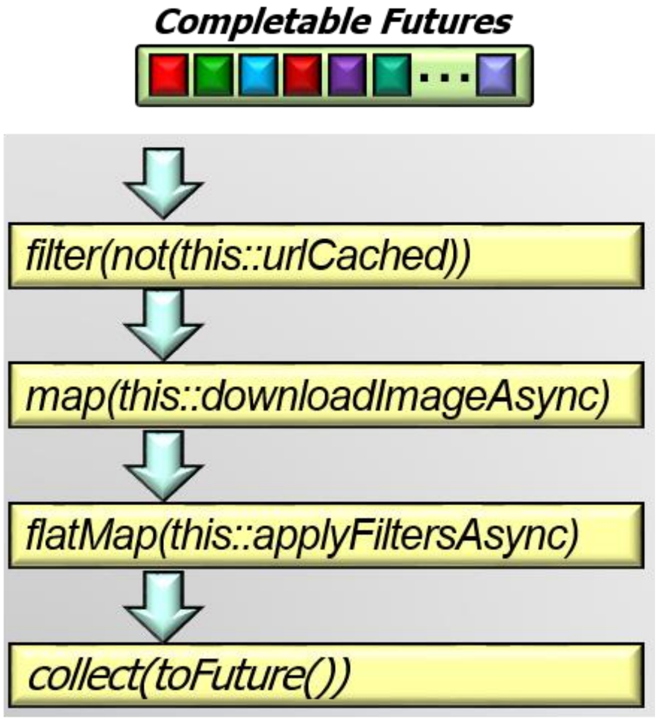
filter(not(this::urlCached))

map(this::downloadImage)

flatMap(this::applyFilters)

collect(toList())

ForkJoinPool



Completable Futures

filter(not(this::urlCached))

map(this::downloadImageAsync)

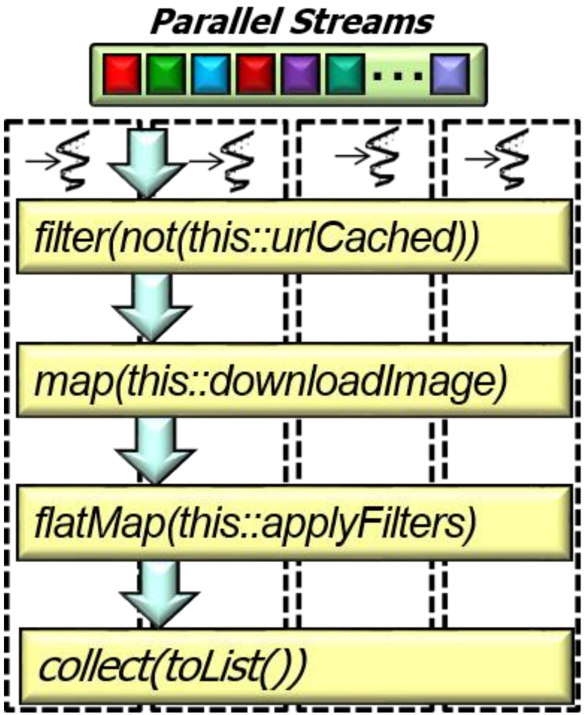
flatMap(this::applyFiltersAsync)

collect(toFuture())

See docs.oracle.com/javase/tutorial/collections/streams/parallelism.html

An Overview of Parallelism in Java

- Java supports parallelism via three primary frameworks



Parallel Streams

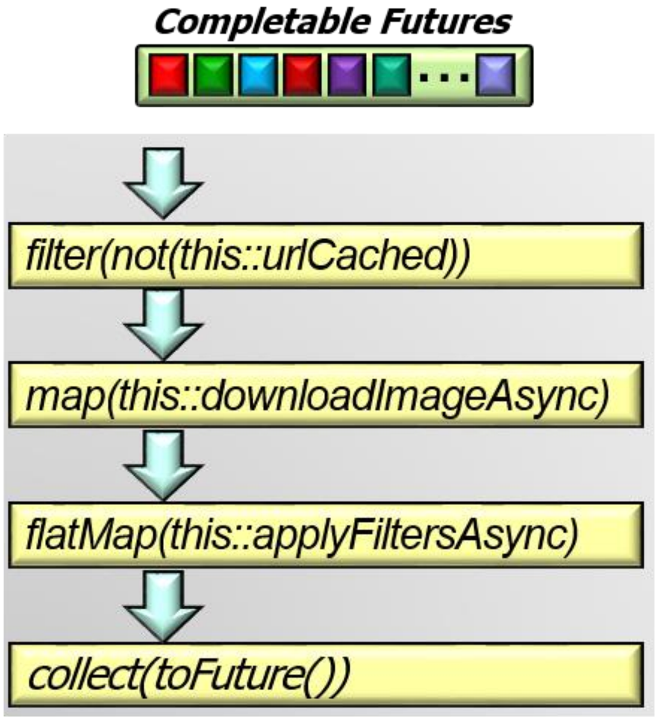
filter(not(this::urlCached))

map(this::downloadImage)

flatMap(this::applyFilters)

collect(toList())

ForkJoinPool



Completable Futures

filter(not(this::urlCached))

map(this::downloadImageAsync)

flatMap(this::applyFiltersAsync)

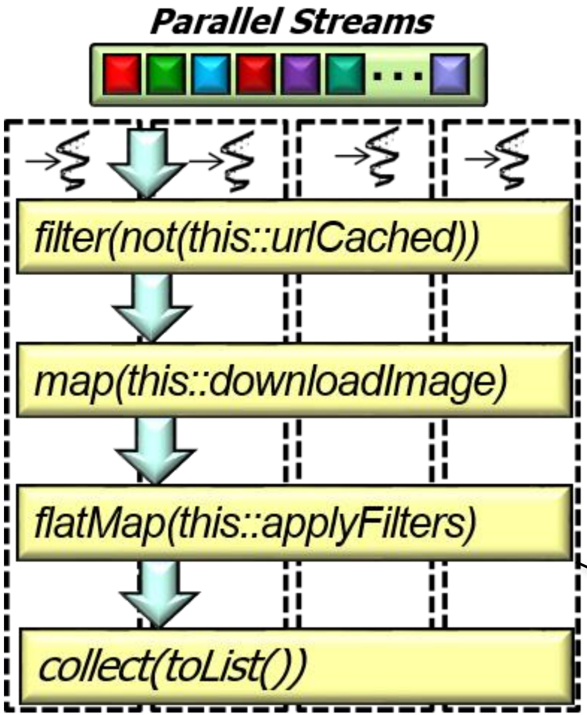
collect(toFuture())

This is an object-oriented framework

See www.dre.vanderbilt.edu/~schmidt/frameworks.html

An Overview of Parallelism in Java

- Java supports parallelism via three primary frameworks



Parallel Streams

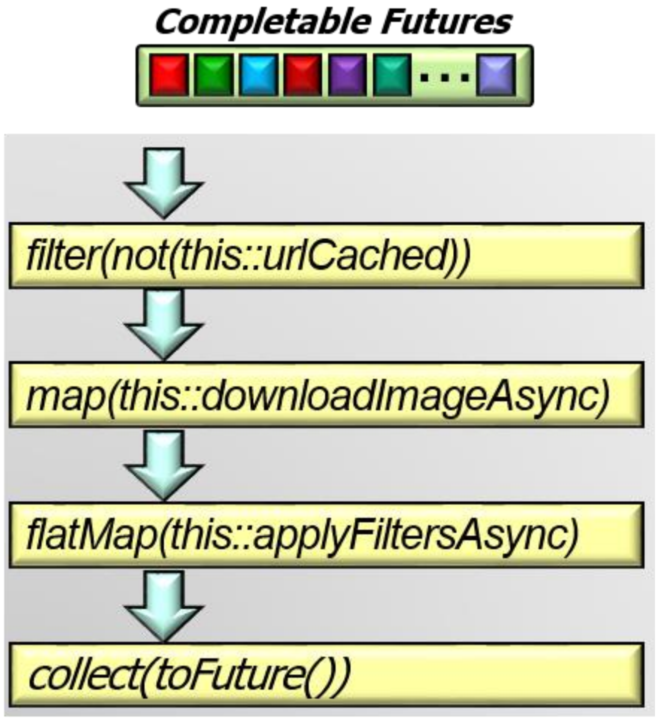
filter(not(this::urlCached))

map(this::downloadImage)

flatMap(this::applyFilters)

collect(toList())

ForkJoinPool



Completable Futures

filter(not(this::urlCached))

map(this::downloadImageAsync)

flatMap(this::applyFiltersAsync)

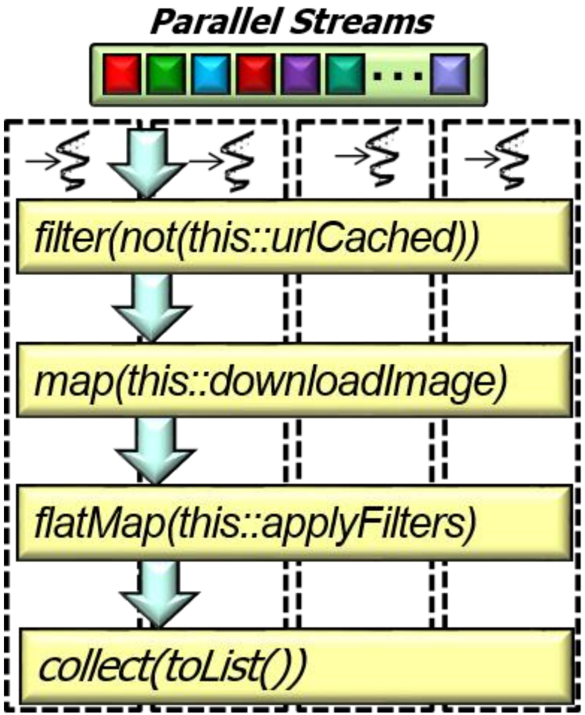
collect(toFuture())

This is a functional framework

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

An Overview of Parallelism in Java

- Java supports parallelism via three primary frameworks



Parallel Streams

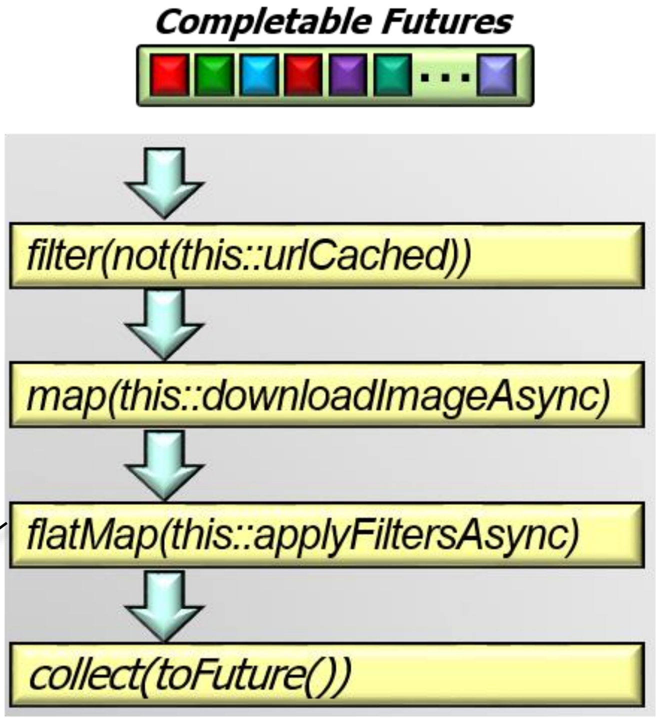
filter(not(this::urlCached))

map(this::downloadImage)

flatMap(this::applyFilters)

collect(toList())

ForkJoinPool



Completable Futures

filter(not(this::urlCached))

map(this::downloadImageAsync)

flatMap(this::applyFiltersAsync)

collect(toFuture())

This is a reactive asynchronous framework

See www.nurkiewicz.com/2013/05/java-8-definitive-guide-to.html

An Overview of Parallelism in Java

- Brian Goetz has an excellent talk about the evolution of Java from concurrent to parallel computing

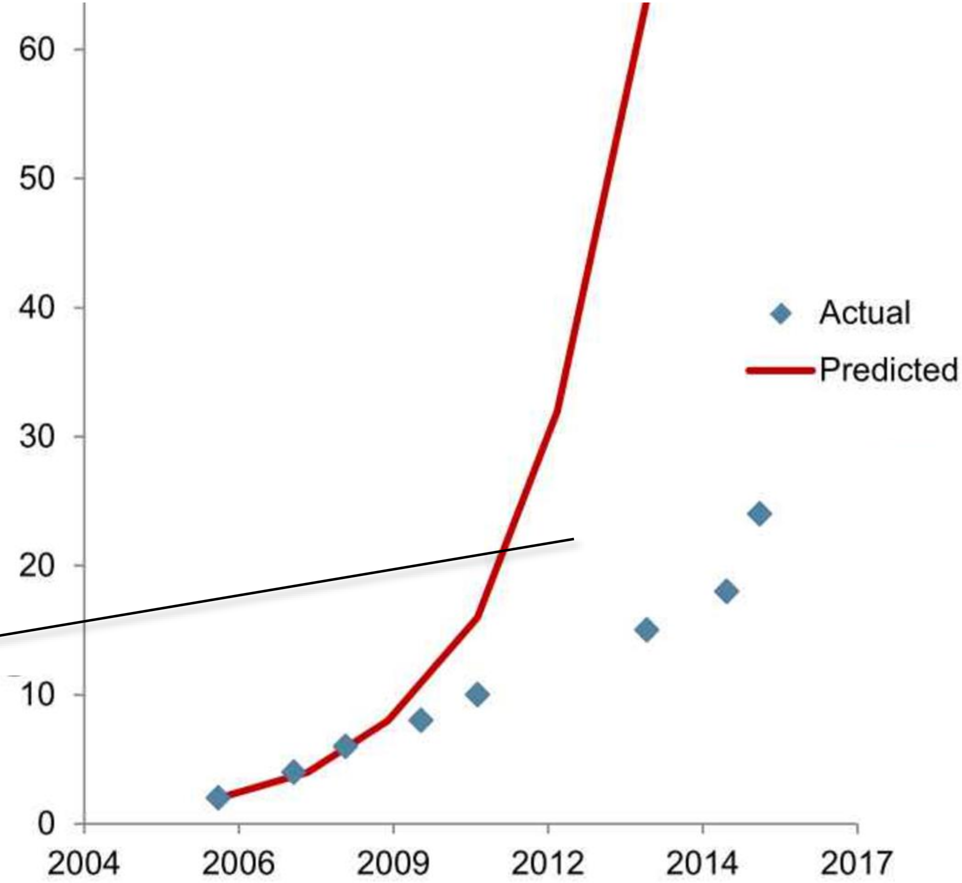
From Concurrent to Parallel

Jfokus

See www.youtube.com/watch?v=NsDE7E8sIdQ

An Overview of Parallelism in Java

- Brian Goetz has an excellent talk about the evolution of Java from concurrent to parallel computing



His talk emphasizes that Java 8 combines functional programming with fine-grained data parallelism to leverage many-core processors

See www.infoq.com/presentations/parallel-java-se-8

End of Overview of Parallelism in Java