The History of Parallelism Support in Java

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

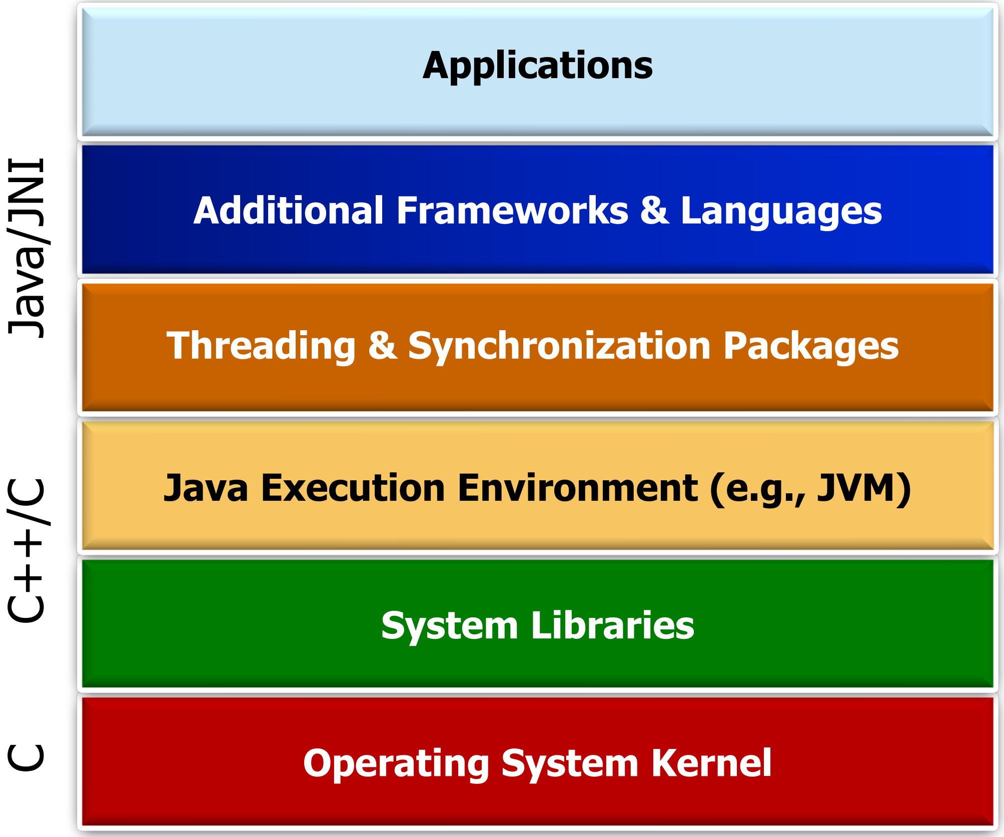
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Learn the history of Java parallelism from 2010 to 2022
  - i.e., fork-join, parallel streams, completable futures, & reactive streams frameworks



Applications

Additional Frameworks & Languages

Threading & Synchronization Packages

Java Execution Environment (e.g., JVM)

System Libraries

Operating System Kernel

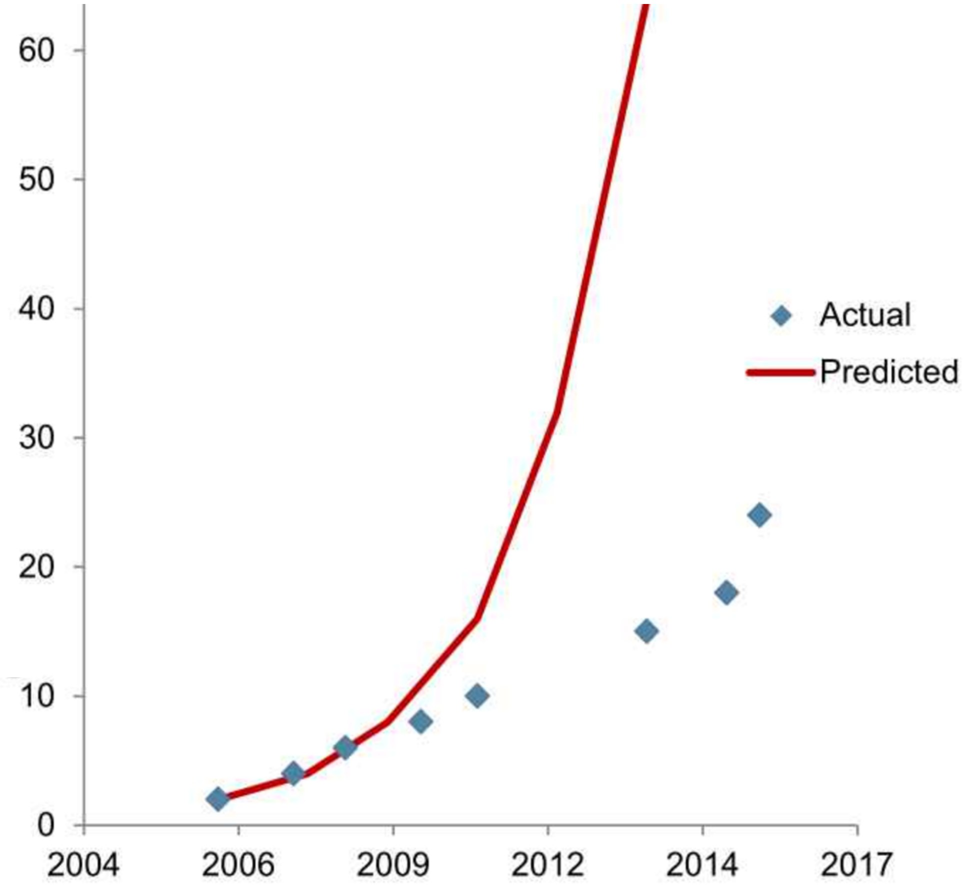
Java/JNI

C++/C

C

Learning Objectives in this Part of the Lesson

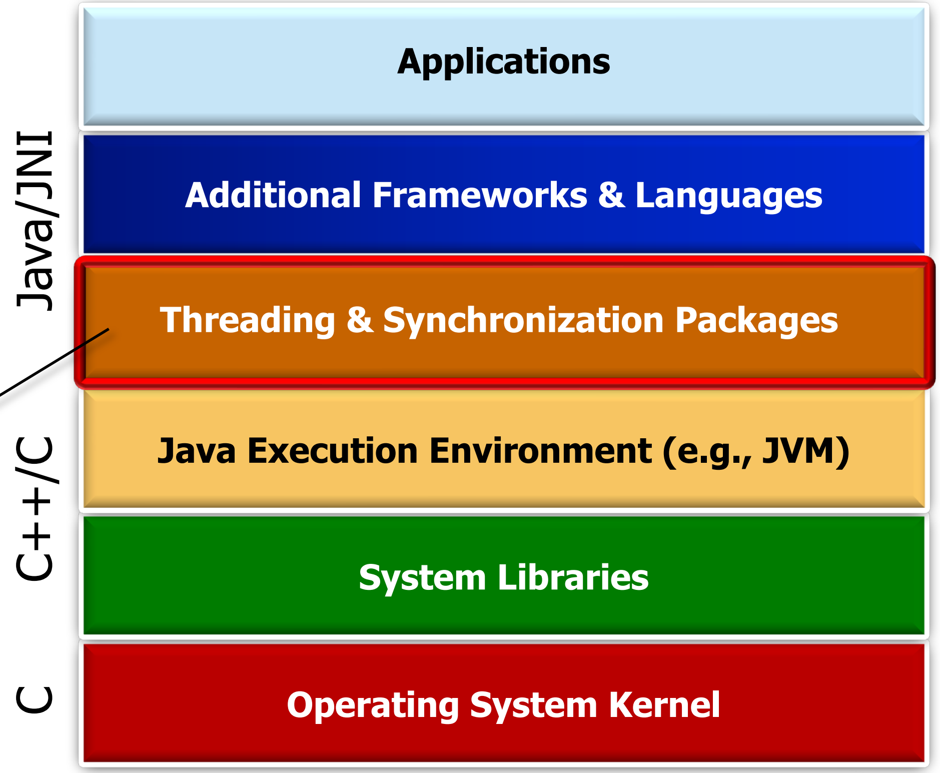
- Learn the history of Java parallelism from 2010 to 2022
- Understand the evolution of Java from concurrency to parallelism



A Brief History of Parallelism in Java

A Brief History of Parallelism in Java

- Foundational parallelism support



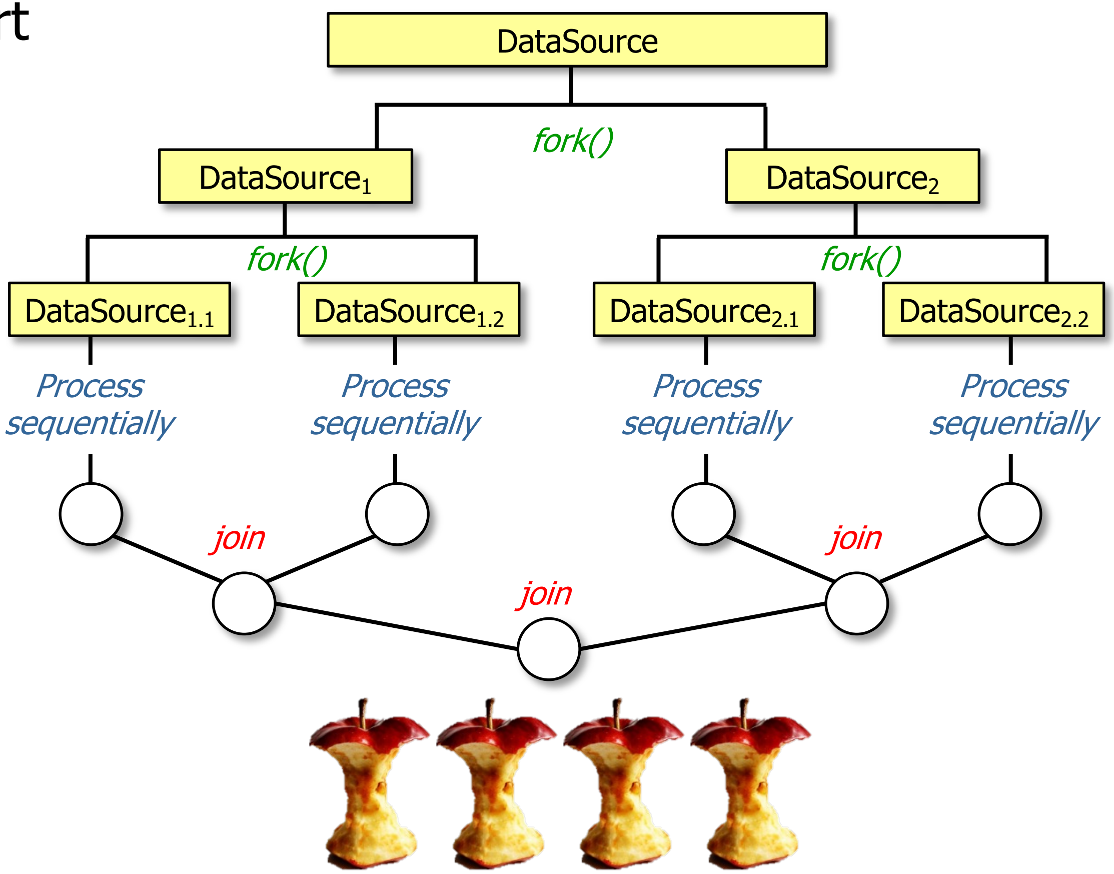
Applications
Additional Frameworks & Languages
Threading & Synchronization Packages
Java Execution Environment (e.g., JVM)
System Libraries
Operating System Kernel
Java/JNI
C++/C
C

e.g., Java fork-join pool was released in Java 7

See en.wikipedia.org/wiki/Java_version_history#Java_SE_7

A Brief History of Parallelism in Java

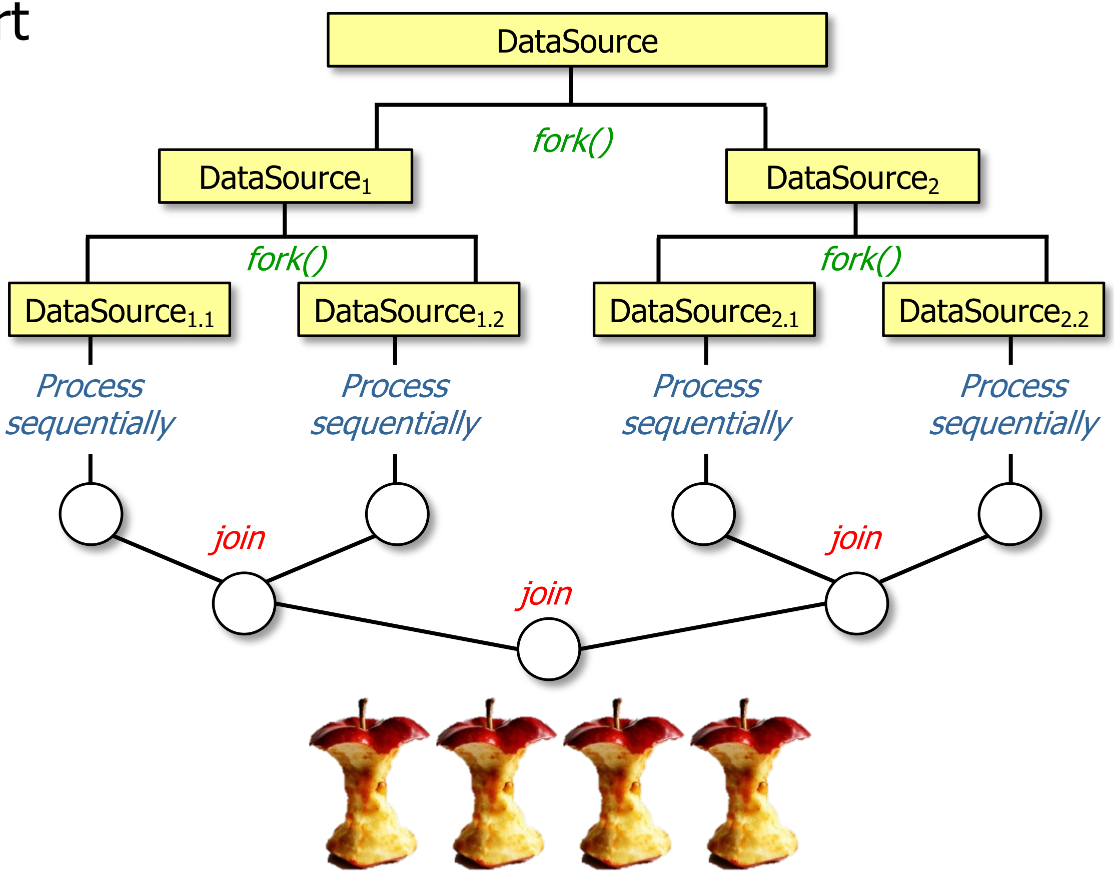
- Foundational parallelism support
  - Focus on fine-grained object-oriented data parallelism



See en.wikipedia.org/wiki/Data_parallelism

A Brief History of Parallelism in Java

- Foundational parallelism support
  - Focus on fine-grained object-oriented data parallelism
    - e.g., runs the same task on different elements of data by using the "split-apply-combine" model



See www.jstatsoft.org/article/view/v040i01/v40i01.pdf

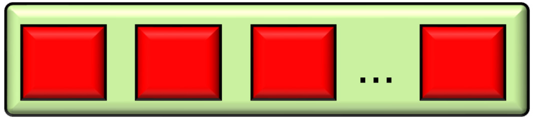
A Brief History of Parallelism in Java

- Foundational parallelism support
  - Focus on fine-grained object-oriented data parallelism
    - e.g., runs the same task on different elements of data by using the "split-apply-combine" model

```
List<List<SearchResults>>
  listOfListOfSearchResults =
    ForkJoinPool
      .commonPool()
      .invoke(new
         SearchWithForkJoinTask
           (inputList,
            mPhrasesToFind, ...));
```

Use a common fork-join pool to search input strings to locate phrases that match famous quotes by the Bard

Input Strings to Search

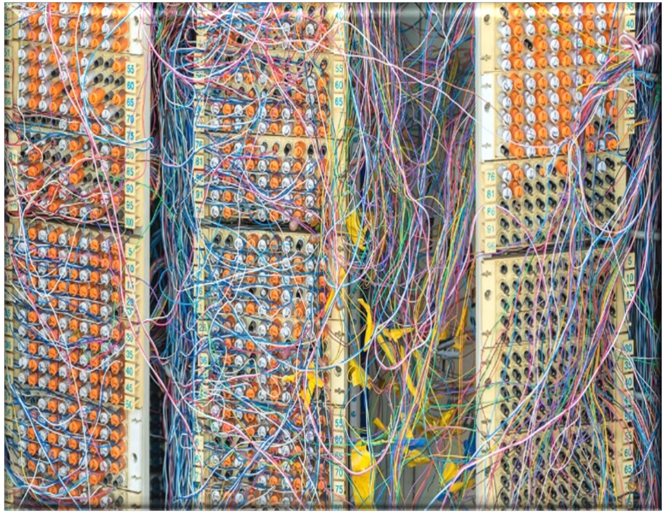


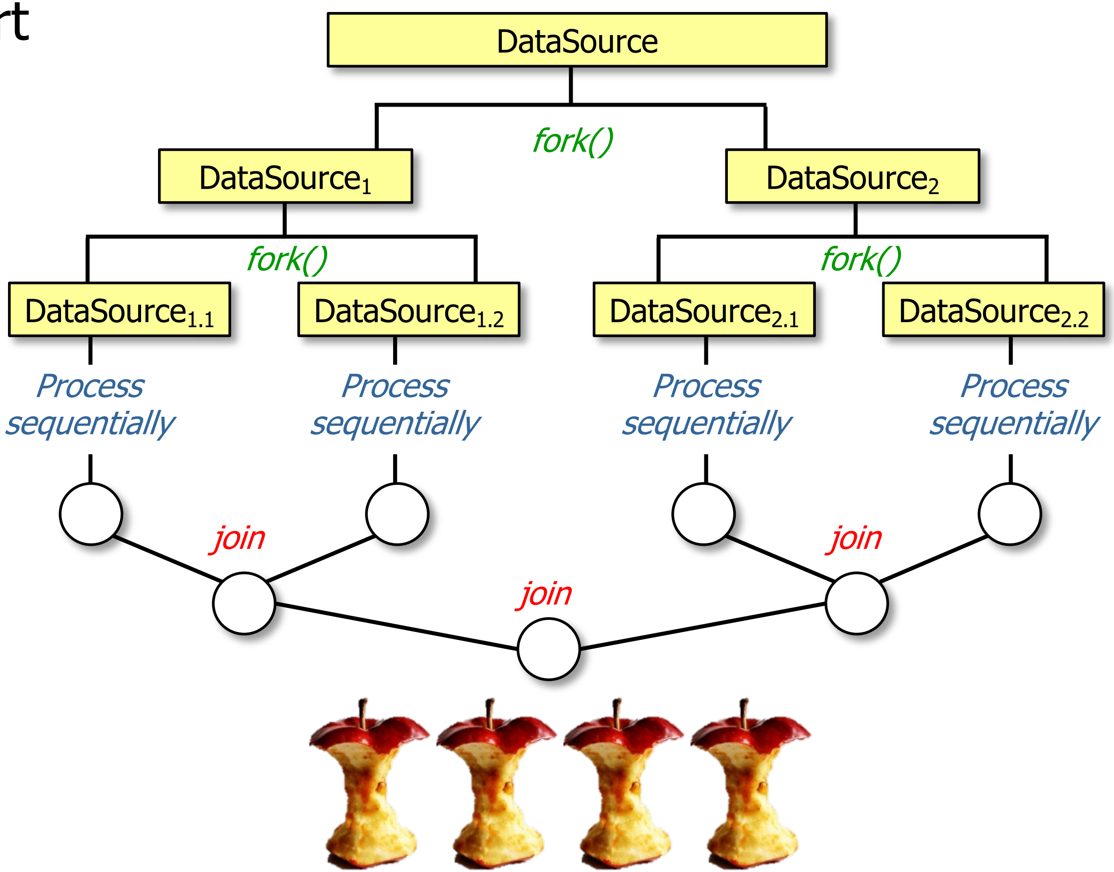
Search Phrases

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

A Brief History of Parallelism in Java

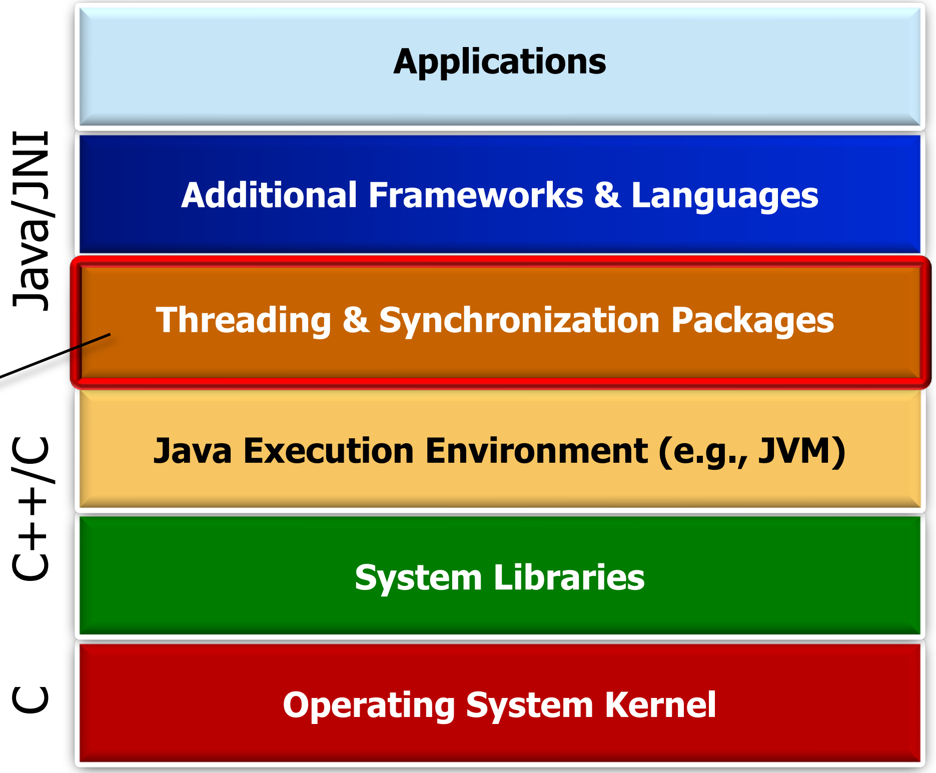
- Foundational parallelism support
  - Focus on fine-grained object-oriented data parallelism
  - Powerful & scalable, but tedious to program directly





A Brief History of Parallelism in Java

- Advanced parallelism support

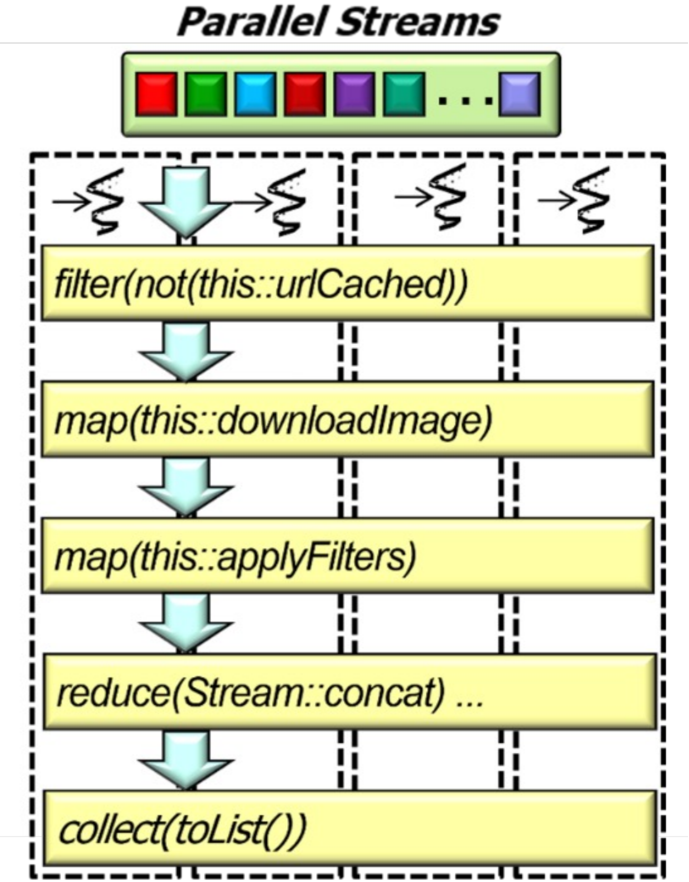

Applications
Additional Frameworks & Languages
Threading & Synchronization Packages
Java Execution Environment (e.g., JVM)
System Libraries
Operating System Kernel
Java/JNI
C++/C
C

e.g., Java parallel streams & completable futures first available in Java 8

See en.wikipedia.org/wiki/Java_version_history#Java_SE_8

A Brief History of Parallelism in Java

- Advanced parallelism support
  - Initial focus on fine-grained functional programming frameworks for data parallelism



See en.wikipedia.org/wiki/Data_parallelism

A Brief History of Parallelism in Java

- Advanced parallelism support
  - Initial focus on fine-grained functional programming frameworks for data parallelism

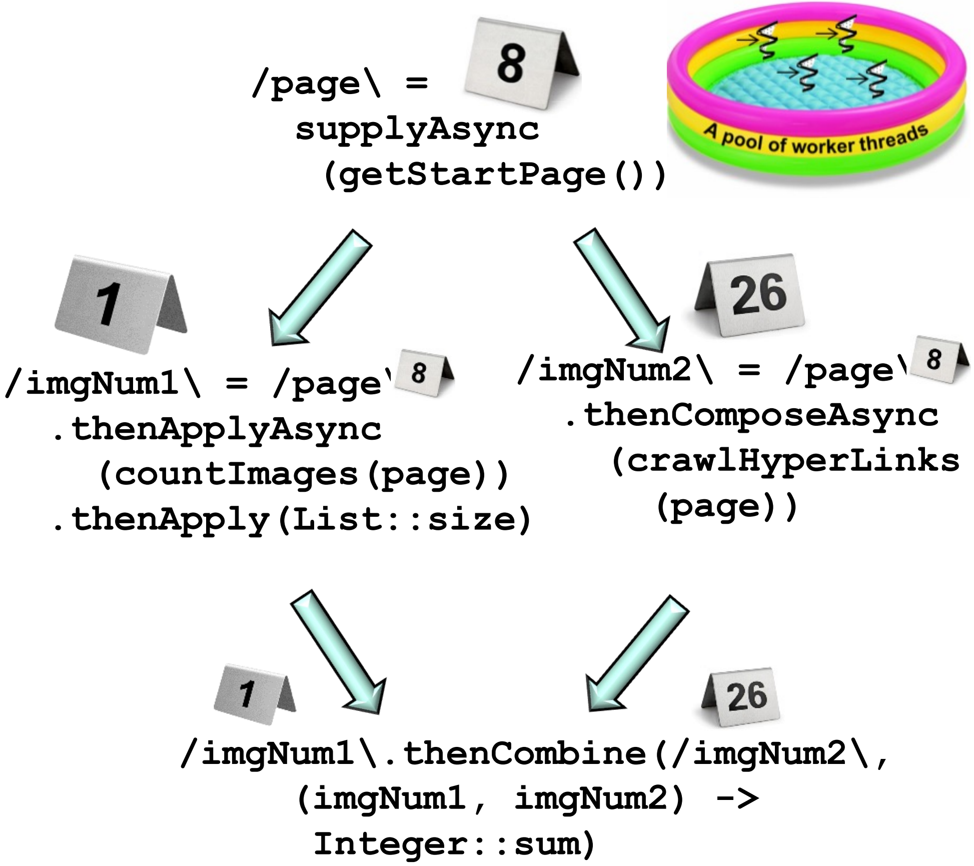
```
List<Image> images =
  urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::downloadImage)
    .map(this::applyFilters)
    .reduce(Stream::concat)
    .orElse(Stream.empty())
    .collect(toList());
```

Synchronously download images that aren't already cached from a list of URLs & process/store the images in parallel

See ImageStreamGang/CommandLine/src/main/java/livelessons/streams/ImageStreamParallel.java

A Brief History of Parallelism in Java

- Advanced parallelism support
  - Initial focus on fine-grained functional programming frameworks for data parallelism & asynchrony



```
/page\ =
  supplyAsync
   (getStartPage())
```

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

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

```
/imgNum1\.thenCombine(/imgNum2\,
  (imgNum1, imgNum2) ->
    Integer::sum)
```

See gist.github.com/staltz/868e7e9bc2a7b8c1f754

A Brief History of Parallelism in Java

- Advanced parallelism support
  - Initial focus on fine-grained functional programming frameworks for data parallelism & asynchrony

```
CompletableFuture
              <Stream<Image>>
  resultsFuture = urls
    .stream()
    .map(this::checkUrlCachedAsync)
    .map(this::downloadImageAsync)
    .flatMap(this::applyFiltersAsync)
    .collect(toFuture())
    .thenApply(stream ->
                log(stream.flatMap
                   (Optional::stream),
                urls.size()))
    .join();
```

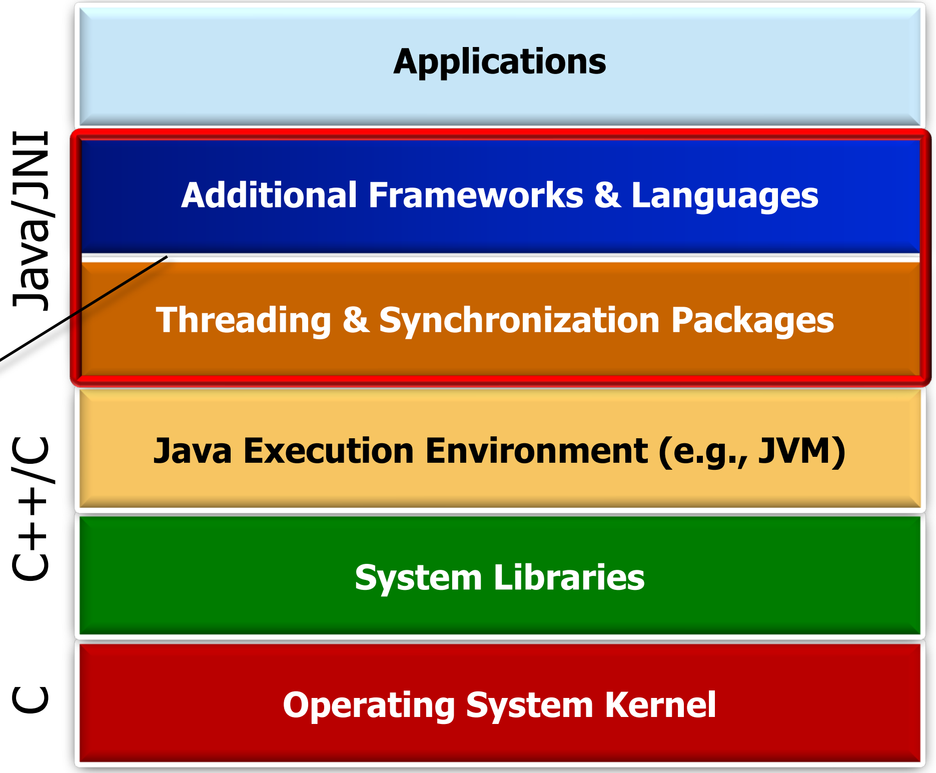
Combines streams & completable futures to asynchronously download images that aren't already cached from a list of URLs & process/store the images in parallel

See ImageStreamGang/CommandLine/src/main/java/livelessons/streams/ImageStreamCompletableFuture1.java

A Brief History of Parallelism in Java

- Advanced parallelism support
  - Initial focus on fine-grained functional programming frameworks for data parallelism & asynchrony
  - Later focus on pub/sub reactive streams frameworks

e.g., Java reactive streams made available in Java 9 have enabled the RxJava & Project Reactor frameworks


Applications
Additional Frameworks & Languages
Threading & Synchronization Packages
Java Execution Environment (e.g., JVM)
System Libraries
Operating System Kernel
Java/JNI
C++/C
C

See en.wikipedia.org/wiki/Java_version_history#Java_SE_9

A Brief History of Parallelism in Java

- Advanced parallelism support
  - Initial focus on fine-grained functional programming frameworks for data parallelism & asynchrony
  - Later focus on pub/sub reactive streams frameworks

```
List<Image> filteredImages
  = Flux
     .fromIterable(urls)
     .parallel()
     .runOn(Schedulers
              .boundedElastic())
     .filter(url -> !urlCached(url))
     .map(this::blockingDownload)
     .flatMap(this::applyFilters)
     .sequential()
     .collectList()
     .block();
```

Applies Project Reactor reactive streams to asynchronously download images that aren't already cached from a list of URLs & process/store the images in parallel

Project Reactor

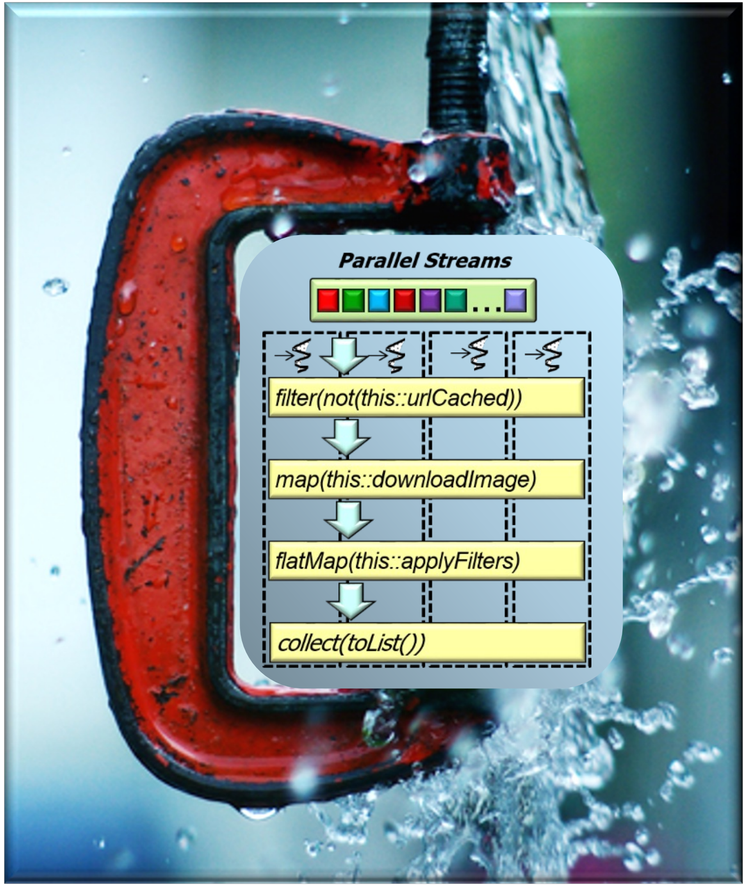
See ImageStreamGang/CommandLine/src/main/java/livelessons/streams/ImageStreamReactor2.java

A Brief History of Parallelism in Java

- Java's advanced parallelism frameworks are designed to strike a balance between productivity & performance

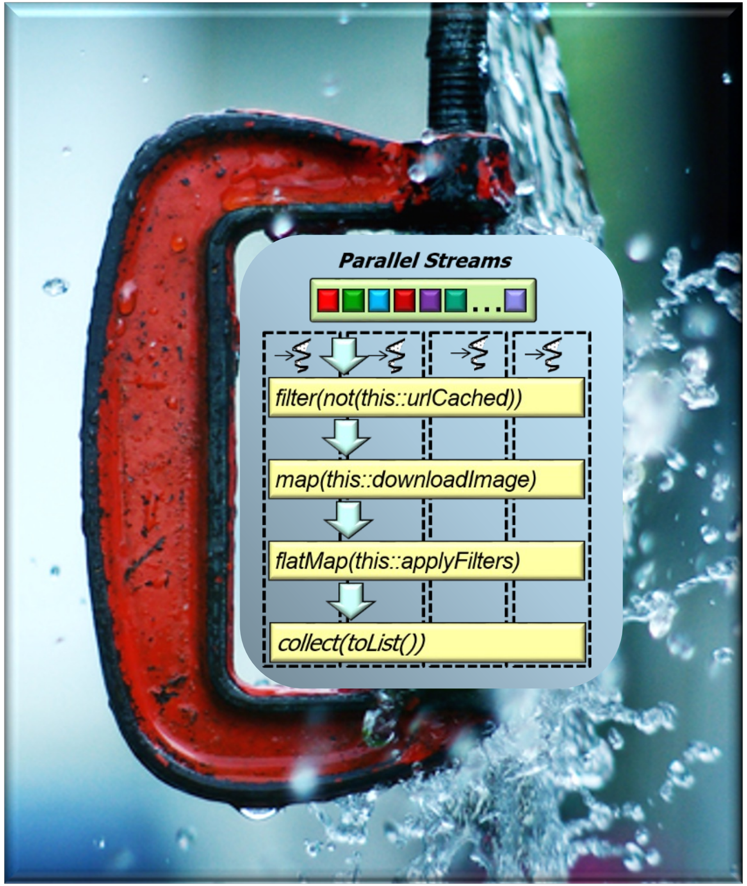
A Brief History of Parallelism in Java

- Java's advanced parallelism frameworks are designed to strike a balance between productivity & performance
  - However, these frameworks can be overly prescriptive



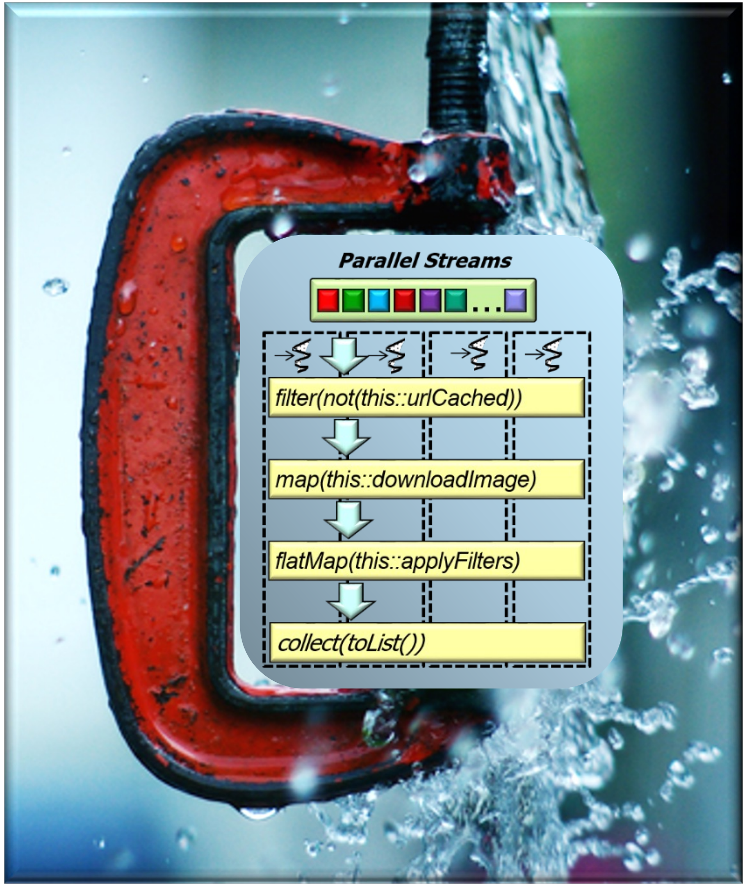
A Brief History of Parallelism in Java

- Java's advanced parallelism frameworks are designed to strike a balance between productivity & performance
  - However, these frameworks can be overly prescriptive, e.g.
    - Abstracting away low-level details limits fine-grained control over the parallelization process



A Brief History of Parallelism in Java

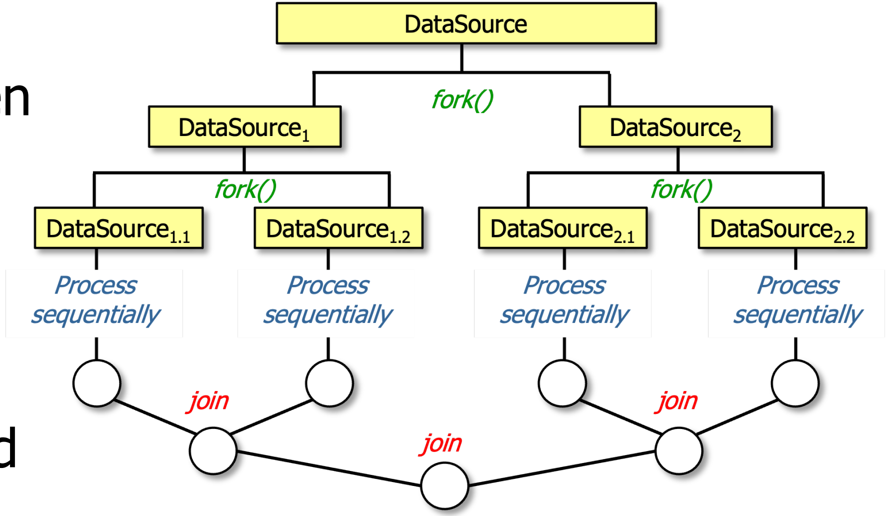
- Java's advanced parallelism frameworks are designed to strike a balance between productivity & performance
  - However, these frameworks can be overly prescriptive, e.g.
    - Abstracting away low-level details limits fine-grained control over the parallelization process
    - If a problem doesn't fit well into the constructs provided it may be hard to implement an efficient solution



```
filter(not(this::urlCached))
map(this::downloadImage)
flatMap(this::applyFilters)
collect(toList())
```

A Brief History of Parallelism in Java

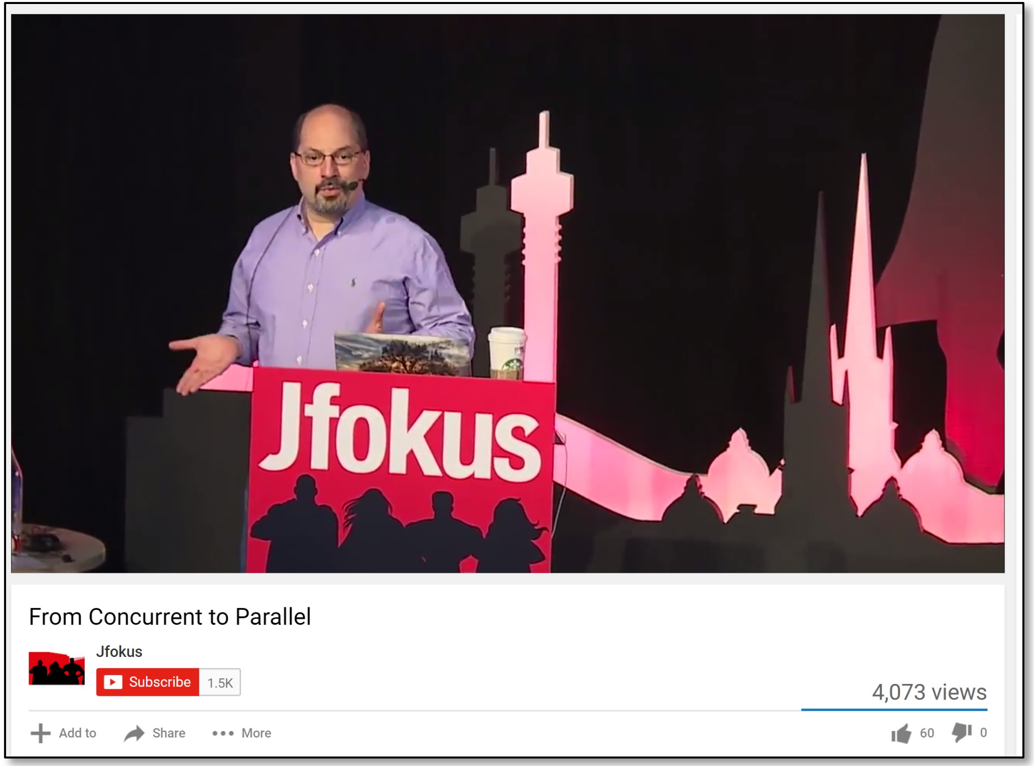
- Java's advanced parallelism frameworks are designed to strike a balance between productivity & performance
    - However, these frameworks can be overly prescriptive
    - There's also some overhead for task scheduling, data partitioning, & thread management



The Evolution of Java from Concurrency to Parallelism

The Evolution of Java from Concurrency to Parallelism

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



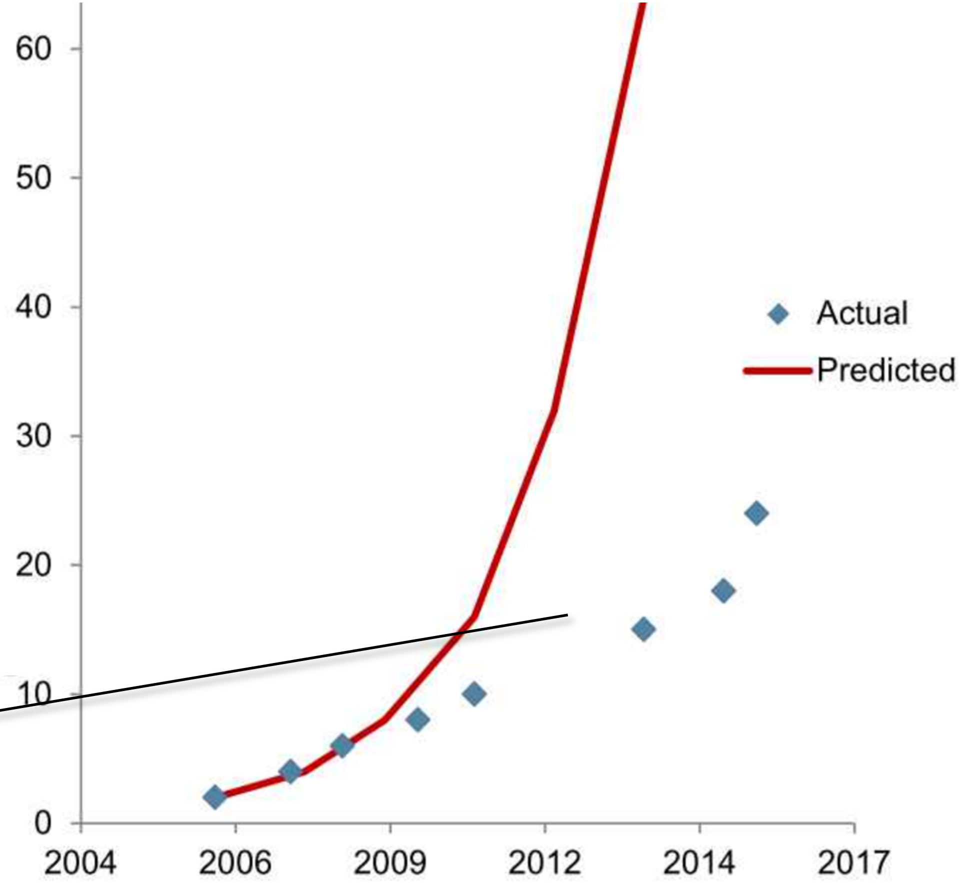
From Concurrent to Parallel

Jfokus

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

The Evolution of Java from Concurrency to Parallelism

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



His talk explains how modern Java combines functional programming with fine-grained data parallelism to leverage many-core processors

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

The Evolution of Java from Concurrency to Parallelism

- Rob Pike also has a good talk that explains the differences between concurrency & parallelism

His talk explains how concurrency is about dealing with lots of things at once, whereas parallelism is about doing lots of things at once

Rob Pike

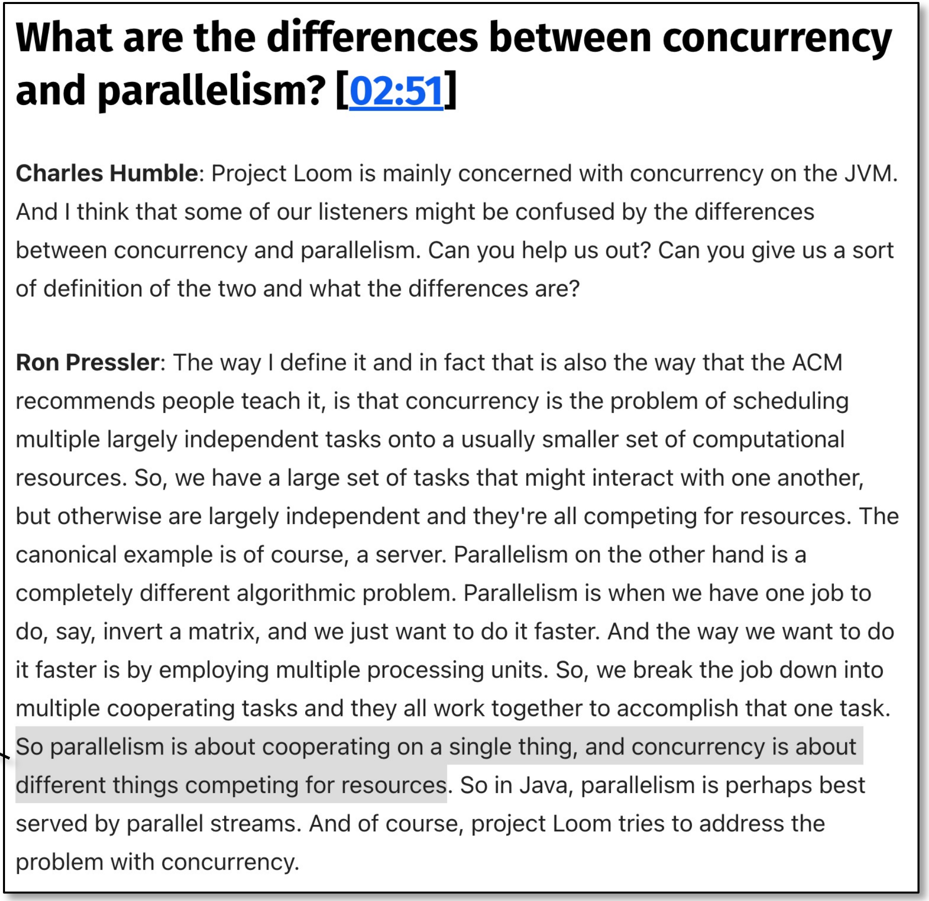
Concurrency is not Parallelism

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

The Evolution of Java from Concurrency to Parallelism

- Likewise, Ron Pressler's podcast differentiates concurrency & parallelism

Parallelism is about cooperating on a single thing, & concurrency is about different things competing for resources



What are the differences between concurrency and parallelism? [02:51]

Charles Humble: Project Loom is mainly concerned with concurrency on the JVM. And I think that some of our listeners might be confused by the differences between concurrency and parallelism. Can you help us out? Can you give us a sort of definition of the two and what the differences are?

Ron Pressler: The way I define it and in fact that is also the way that the ACM recommends people teach it, is that concurrency is the problem of scheduling multiple largely independent tasks onto a usually smaller set of computational resources. So, we have a large set of tasks that might interact with one another, but otherwise are largely independent and they're all competing for resources. The canonical example is of course, a server. Parallelism on the other hand is a completely different algorithmic problem. Parallelism is when we have one job to do, say, invert a matrix, and we just want to do it faster. And the way we want to do it faster is by employing multiple processing units. So, we break the job down into multiple cooperating tasks and they all work together to accomplish that one task. So parallelism is about cooperating on a single thing, and concurrency is about different things competing for resources. So in Java, parallelism is perhaps best served by parallel streams. And of course, project Loom tries to address the problem with concurrency.

See www.infoq.com/podcasts/java-project-loom

End of the History of Parallelism Support in Java