Motivating the Need for
Java 8 Completable Futures (Part 1)

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

• Motivate the need for Java futures by understanding the pros & cons of synchrony & asynchrony
Motivating Need for Futures: Pros & Cons of Synchrony
Motivating Need for Futures: Pros & Cons of Synchrony

- Method calls in typical Java programs are largely *synchronous*

```
e.g., calls on Java collections & behaviors in Java 8 stream aggregate operations
```
Motivating Need for Futures: Pros & Cons of Synchrony

- Method calls in typical Java programs are largely *synchronous*.
  - i.e., a callee borrows the thread of its caller until its computation(s) finish.
Motivating Need for Futures: Pros & Cons of Synchrony

- Method calls in typical Java programs are largely *synchronous*
- i.e., a callee borrows the thread of its caller until its computation(s) finish
Motivating Need for Futures: Pros & Cons of Synchrony

- Synchronous calls have pros & cons
Motivating Need for Futures: Pros & Cons of Synchrony

• Pros of synchronous calls:
  • “Intuitive” to program & debug
Motivating Need for Futures: Pros & Cons of Synchrony

• Pros of synchronous calls:
  • “Intuitive” to program & debug, e.g.
  • Maps onto common two-way method patterns

See www.iro.umontreal.ca/~keller/Layla/remote.pdf
Motivating Need for Futures: Pros & Cons of Synchrony

- Pros of synchronous calls:
  - “Intuitive” to program & debug, e.g.
  - Maps onto common two-way method patterns
  - Local caller state retained when callee returns

See [wiki.c2.com/?ActivationRecord](http://wiki.c2.com/?ActivationRecord)
Motivating Need for Futures: Pros & Cons of Synchrony

- Pros of synchronous calls:
  - “Intuitive” to program & debug, e.g.
  - Maps onto common two-way method patterns
  - Local caller state retained when callee returns

```java
byte[] downloadContent(URL url) {
    byte[] buf = new byte[BUFSIZ];
    ByteArrayOutputStream os =
        new ByteArrayOutputStream();
    InputStream is = url.openStream();
    for (int bytes;
         (bytes = is.read(buf)) > 0;)
        os.write(buf, 0, bytes); ... 
}
```

See [wiki.c2.com/?ActivationRecord](http://wiki.c2.com/?ActivationRecord)
Motivating Need for Futures: Pros & Cons of Synchrony

- Cons of synchronous calls:
  - May not leverage all parallelism available in multi-core systems

CALLER

searchForWord_1

return result_1

searchForWord_2

return result_2

searchForWord_3

return return_3

CALLEE
Motivating Need for Futures: Pros & Cons of Synchrony

- Cons of synchronous calls:
  - May not leverage all parallelism available in multi-core systems
  - Blocking threads incur overhead
    - e.g., synchronization, context switching, data movement, & memory management costs

Motivating Need for Futures: Pros & Cons of Synchrony

- Cons of synchronous calls:
  - May not leverage all parallelism available in multi-core systems
  - Blocking threads incur overhead
  - Selecting right # of threads is hard

```java
List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::downloadImage)
    .flatMap(this::applyFilters)
    .collect(toList());

Image downloadImage(URL url){
    return new Image(url,
                     downloadContent(url));
}
```

Motivating Need for Futures: Pros & Cons of Synchrony

- **Cons of synchronous calls:**
  - May not leverage all parallelism available in multi-core systems
  - Blocking threads incur overhead
  - Selecting right # of threads is hard

```java
List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::downloadImage)
    .flatMap(this::applyFilters)
    .collect(toList());
```

A large # of threads may improve performance at the cost of wasted resources
Motivating Need for Futures: Pros & Cons of Synchrony

• Cons of synchronous calls:
  • May not leverage all parallelism available in multi-core systems
  • Blocking threads incur overhead
  • Selecting right # of threads is hard

List<Image> filteredImages = urls
  .parallelStream()
  .filter(not(this::urlCached))
  .map(this::downloadImage)
  .flatMap(this::applyFilters)
  .collect(toList());
Motivating Need for Futures: Pros & Cons of Synchrony

• Cons of synchronous calls:
  • May not leverage all parallelism available in multi-core systems
    • Blocking threads incur overhead
  • Selecting right # of threads is hard

Efficient Resource Utilization
Efficient Performance

Particularly tricky for I/O-bound programs that need more threads to run efficiently
Motivating Need for Futures: Pros & Cons of Synchrony

• Cons of synchronous calls:
  • May not leverage all parallelism available in multi-core systems
  • May need to change common fork-join pool size in a Java 8 parallel stream

See dzone.com/articles/think-twice-using-java-8
Motivating Need for Futures: Pros & Cons of Asynchrony
Motivating Need for Futures: Pros & Cons of Asynchrony

- Asynchronous (async) calls can alleviate the limitations with synchronous calls.

See en.wikipedia.org/wiki/Asynchrony_(computer_programming)
Motivating Need for Futures: Pros & Cons of Asynchrony

- Asynchronous (async) calls can alleviate the limitations with synchronous calls.
- Asynchrony is a means of concurrent programming where a unit of work has certain properties.
Motivating Need for Futures: Pros & Cons of Asynchrony

- Asynchronous (async) calls can alleviate the limitations with synchronous calls.
- Asynchrony is a means of concurrent programming where a unit of work has certain properties.
- Runs separately from the calling thread “in the background.”
Motivating Need for Futures: Pros & Cons of Asynchrony

- Asynchronous (async) calls can alleviate the limitations with synchronous calls.
- Asynchrony is a means of concurrent programming where a unit of work has certain properties:
  - Runs separately from the calling thread “in the background”
  - Notifies the calling thread of its completion, failure, or progress.
Motivating Need for Futures: Pros & Cons of Asynchrony

- Asynchronous calls have pros & cons
Motivating Need for Futures: Pros & Cons of Asynchrony

- Pros of asynchronous calls
  - Responsiveness
    - A thread does not block waiting for other requests to complete
Motivating Need for Futures: Pros & Cons of Asynchrony

- Pros of asynchronous calls
  - Responsiveness
- Elasticity
  - Multiple requests can run scalably & concurrently in multiple cores
Motivating Need for Futures: Pros & Cons of Asynchrony

- Cons of asynchronous calls
  - Unpredictability
    - Response times may not be predictable
Motivating Need for Futures: Pros & Cons of Asynchrony

- Cons of asynchronous calls
  - Unpredictability
    - Response times may not be predictable
  - Results can occur in a different order than the original calls were made
Motivating Need for Futures: Pros & Cons of Asynchrony

- Cons of asynchronous calls
  - Unpredictability
  - Complicated debugging
    - Errors can be hard to track due to unpredictability

End of Motivating the Need for Java 8 Completable Futures (Part 1)