The Pros & Cons of Synchrony

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
Overview of Synchrony & Synchronous Operations
Overview of Synchrony & Synchronous Operations

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

---

e.g., calls on Java collections & behaviors in Java stream aggregate operations
Overview of Synchrony & Synchronous Operations

- Method calls in typical Java programs are largely *synchronous*
- i.e., a callee borrows the thread of its caller until its computation(s) finish
Overview of Synchrony & Synchronous Operations

• Method calls in typical Java programs are largely *synchronous*
  • i.e., a callee borrows the thread of its caller until its computation(s) finish

  ![Diagram of synchronous method calls]

  *Note "request/response" nature of these calls*

  ![TWO WAY sign]
The Pros of Synchrony
The Pros of Synchrony

• Pros of synchronous calls:
• “Intuitive” to program & debug
The Pros 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](http://www.iro.umontreal.ca/~keller/Layla/remote.pdf)
The Pros 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
The Pros 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);
    return return;
}
```

See [wiki.c2.com/?ActivationRecord](http://wiki.c2.com/?ActivationRecord)
The Cons of Synchrony
The Cons of Synchrony

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

See www.ibm.com/developerworks/library/j-jvmc3
The 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

The 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));
}
```

The 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 help to improve performance, but can also waste resources.
The 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 small # of threads may conserve resources at the cost of performance
The 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

Particularly tricky for I/O-bound programs that need more threads to run efficiently
The 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 parallel stream

See dzone.com/articles/think-twice-using-java-8
The 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 parallel stream
  - Or use the Java fork-join pool ManagedBlocker feature to increase common pool size automatically/temporarily

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.ManagedBlocker.html
End of the Pros & Cons of Synchrony