Understanding the Pros & Cons of Asynchrony

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 Future & CompletableFuture mechanisms by understanding the pros & cons of synchrony

• Motivate the need for Java Future & CompletableFuture mechanisms by understanding the pros & cons of asynchrony
Overview of Asynchrony & Asynchronous Operations
Overview of Asynchrony & Asynchronous Operations

- Asynchrony is a means of concurrent programming where caller does not block waiting for callee to complete.

See [en.wikipedia.org/wiki/Asynchrony_(computer_programming)](en.wikipedia.org/wiki/Asynchrony_(computer_programming))
Overview of Asynchrony & Asynchronous Operations

- Asynchrony is a means of concurrent programming where caller does not block waiting for callee to complete.
- An async call immediately returns a future & while the computation runs “in the background” concurrently.

See [en.wikipedia.org/wiki/Asynchronous_method_invocation](en.wikipedia.org/wiki/Asynchronous_method_invocation)
Overview of Asynchrony & Asynchronous Operations

- Asynchrony is a means of concurrent programming where caller does not block waiting for callee to complete.
- An async call immediately returns a future & while the computation runs “in the background” concurrently.
- i.e., independent of the calling thread’s flow of control.

See en.wikipedia.org/wiki/Control_flow
Asynchrony is a means of concurrent programming where the caller does not block waiting for the callee to complete.

An async call immediately returns a future and while the computation runs "in the background" concurrently.

The future is triggered when the computation completes.

Overview of Asynchrony & Asynchronous Operations

1. Async call runs
2. Client obtains result after the computation completes

See upcoming lessons on "Overview of Java Futures"
Asynchrony is a means of concurrent programming where caller does not block waiting for callee to complete.

- An async call immediately returns a future & while the computation runs “in the background” concurrently.
- The future is triggered when the computation completes.
- The client may or may not block awaiting the results, depending on various factors.
Overview of Asynchrony & Asynchronous Operations

- e.g., Android’s AsyncTask framework performs background operations & publishes results on the user-interface (UI) thread without having to manipulate threads and/or handlers

See developer.android.com/reference/android/os/AsyncTask
Overview of Asynchrony & Asynchronous Operations

- e.g., Android’s AsyncTask framework performs background operations & publishes results on the user-interface (UI) thread without having to manipulate threads and/or handlers

- AsyncTask executes long-duration operations asynchronously in one or more background threads
Overview of Asynchrony & Asynchronous Operations

- e.g., Android’s AsyncTask framework performs background operations & publishes results on the user-interface (UI) thread without having to manipulate threads and/or handlers
  - AsyncTask executes long-duration operations asynchronously in one or more background threads
  - Blocking operations in background threads don’t block the calling (e.g., UI) thread

See developer.android.com/training/multiple-threads/communicate-ui
Overview of Asynchrony & Asynchronous Operations

- e.g., Android’s AsyncTask framework performs background operations & publishes results on the user-interface (UI) thread without having to manipulate threads and/or handlers

  - AsyncTask executes long-duration operations asynchronously in one or more background threads
  - Blocking operations in background threads don’t block the calling (e.g., UI) thread
  - The calling (UI) thread can be notified upon completion, failure, or progress of the async task

AsyncTask shields client code from details of programming futures
The Pros of Asynchrony
The Pros of Asynchrony

• Pros of asynchronous operations
The Pros of Asynchrony

• Pros of asynchronous operations
  • Responsiveness
    • A calling thread needn’t block waiting for the async request to complete

See en.wikipedia.org/wiki/Asynchronous_method_invocation
The Pros of Asynchrony

- Pros of asynchronous operations
  - Responsiveness
- Elasticity
  - Multiple requests can run scalably & concurrently on multiple cores

See en.wikipedia.org/wiki/Elasticity_(cloud_computing)
The Pros of Asynchrony

- Pros of asynchronous operations
  - Responsiveness
- Elasticity
  - Multiple requests can run scalably & concurrently on multiple cores
  - Able to better leverage parallelism available in multi-core systems

See headcrashing.wordpress.com/2015/07/20/iobound-completablefuture
The Pros of Asynchrony

- Pros of asynchronous operations
  - Responsiveness
  - Elasticity
  - Multiple requests can run scalably & concurrently on multiple cores
    - Able to better leverage parallelism available in multi-core systems
  - Elasticity is particularly useful to auto-scale computations in cloud environments

See en.wikipedia.org/wiki/Elasticity_(cloud_computing) & en.wikipedia.org/wiki/Autoscaling
The Cons of Asynchrony
The Cons of Asynchrony

- Cons of asynchronous operations
The Cons of Asynchrony

- Cons of asynchronous operations
  - Unpredictability
    - Response times may not unpredictable due to non-determinism of async operations

Non-determinism is a general problem with concurrency & not just asynchrony

See [en.wikipedia.org/wiki/Nondeterministic_algorithm](en.wikipedia.org/wiki/Nondeterministic_algorithm)
The Cons of Asynchrony

- Cons of asynchronous operations
  - Unpredictability
    - Response times may not unpredictable due to non-determinism of async operations
  - Results can occur in a different order than the original calls were made

Additional time & effort may be required if results must be ordered somehow
The Cons of Asynchrony

- Cons of asynchronous operations
  - Unpredictability
- Complicated programming & debugging
The Cons of Asynchrony

- Cons of asynchronous operations
  - Unpredictability
  - Complicated programming & debugging
  - The patterns & best-practices of asynchronous programming are not well understood

Parallel and Asynchronous Programming in Java 8

Java 8 offered a boon to parallel and asynchronous programming. Let’s check out the lessons Java learned from JavaScript and how JDK 8 changed the game.

by Lisa Steendam · May. 11, 18 · Java Zone · Tutorial

Parallel code, which is code that runs on more than one thread, was once the nightmare of many an experienced developer, but Java 8 brought a lot of changes that should make this performance-boosting trick a lot more manageable.

CompletableFuture

CompletableFuture implements both the Future and the CompletionStage interface. Future already existed pre-Java 8, but it wasn’t very developer-friendly by itself. You could only get the result of the asynchronous computation by using the get() method, which blocked the rest (making the async part pretty pointless most of the time) and you needed to implement each possible scenario manually. Adding the CompletionStage interface was the breakthrough that made asynchronous programming in Java workable.

CompletionStage is a promise, namely the promise that the computation will eventually be done. It contains a bunch of methods that let you attach callbacks that will be executed on that completion. Now we can handle the result without blocking.

There are two main methods that let you start the asynchronous part of your code: CompletableFuture.supplyAsync if you want to do something with the result of the method, and CompletableFuture.runAsync if you don’t.

See dzone.com/articles/parallel-and-asynchronous-programming-in-java-8
The Cons of Asynchrony

- Cons of asynchronous operations
- Unpredictability
- Complicated programming & debugging
  - The patterns & best-practices of asynchronous programming are not well understood
- Async programming is tricky without proper abstractions

See dzone.com/articles/callback-hell
The Cons of Asynchrony

- Cons of asynchronous operations
  - Unpredictability
  - Complicated programming & debugging
    - The patterns & best-practices of asynchronous programming are not well understood
    - Errors can be hard to track due to unpredictability

The Cons of Asynchrony

- Cons of asynchronous operations
  - Unpredictability
- Complicated programming & debugging
  - The patterns & best-practices of asynchronous programming are not well understood
- Errors can be hard to track due to unpredictability

*Again, non-determinism is a general problem with concurrency & not just with asynchrony*
Weighing the Pros & Cons of Asynchrony
Weighing the Pros & Cons of Asynchrony

- Two things are necessary for the pros of asynchrony to outweigh the cons
Weighing the Pros & Cons of Asynchrony

- Two things are necessary for the pros of asynchrony to outweigh the cons
- Performance should improve to offset the increased complexity of programming & debugging

See upcoming lesson on “Java Completable Futures ImageStreamGang Example”
Two things are necessary for the pros of asynchrony to outweigh the cons:

- Performance should improve to offset the increased complexity of programming & debugging.
- An asynchronous programming model should reflect the key principles of the reactive paradigm.

See earlier lesson on “Overview of Reactive Programming”.
Weighing the Pros & Cons of Asynchrony

- Java’s completable futures framework provides an asynchronous concurrent programming model that performs well & supports the reactive paradigm.

```java
public class CompletableFuture<T>
    extends Object
    implements Future<T>, CompletionStage<T>

A Future that may be explicitly completed (setting its value and status), and may be used as a CompletionStage, supporting dependent functions and actions that trigger upon its completion.

When two or more threads attempt to complete, completeExceptionally, or cancel a CompletableFuture, only one of them succeeds.

In addition to these and related methods for directly manipulating status and results, CompletableFuture implements interface CompletionStage with the following policies:
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html)
Weighing the Pros & Cons of Asynchrony

• Java’s completable futures framework provides an asynchronous concurrent programming model that performs well & supports the reactive paradigm
• However, reactive streams frameworks are even better suited to supporting the reactive programming paradigm

See www.baeldung.com/rx-java & projectreactor.io
End of Understanding the Pros & Cons of Asynchrony