How Parallel Programs are Developed in Java (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

- Recognize the parallelism frameworks supported by Java, e.g.
  - **Fork-join pools**
    - An object-oriented data parallelism framework

See [docs.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html](http://docs.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html)
Overview of Java Object-Oriented Parallelism Frameworks
Overview of Java Object-Oriented Parallelism Frameworks

- The fork-join framework defines an object-oriented parallelism model

See [www.infoq.com/interviews/doug-lea-fork-join](http://www.infoq.com/interviews/doug-lea-fork-join)
Overview of Java Object-Oriented Parallelism Frameworks

• The fork-join framework defines an object-oriented parallelism model
• Provides high performance, fine-grained task execution

Designed to scale up to processors with many cores (cf. the executor framework)
Overview of Java Object-Oriented Parallelism Frameworks

- The fork-join framework defines an object-oriented parallelism model
  - Provides high performance, fine-grained task execution
- The focus is on data parallelism
  - i.e., data is partitioned across multiple threads/cores, which operate on the data in parallel

See [en.wikipedia.org/wiki/Data_parallelism](en.wikipedia.org/wiki/Data_parallelism)
The fork-join framework defines an object-oriented parallelism model
- Provides high performance, fine-grained task execution
- The focus is on data parallelism
- The key abstraction is the ForkJoinTask

Overview of Java Object-Oriented Parallelism Frameworks

Class ForkJoinTask<V>
java.lang.Object
   java.util.concurrent.ForkJoinTask<V>

All Implemented Interfaces:
   Serializable, Future<V>

Direct Known Subclasses:
   CountedCompleter, RecursiveAction, RecursiveTask

public abstract class ForkJoinTask<V>
   extends Object
   implements Future<V>, Serializable

Abstract base class for tasks that run within a ForkJoinPool. A ForkJoinTask is a thread-like entity that is much lighter weight than a normal thread. Huge numbers of tasks and subtasks may be hosted by a small number of actual threads in a ForkJoinPool, at the price of some usage limitations.

A "main" ForkJoinTask begins execution when it is explicitly submitted to a ForkJoinPool, or, if not already engaged in a ForkJoin computation, commenced in the ForkJoinPool.commonPool() via fork(), invoke(), or related methods. Once started, it will usually in turn start other subtasks. As indicated by the name of this class, many programs using ForkJoinTask employ only methods fork() and join(), or derivatives such as invokeAll. However, this class also provides a number of other methods that can come into play in advanced usages, as well as extension mechanics that allow support of new forms of fork/join processing.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html
Overview of Java Object-Oriented Parallelism Frameworks

- The fork-join framework defines an object-oriented parallelism model
  - Provides high performance, fine-grained task execution
  - The focus is on data parallelism
  - The key abstraction is the ForkJoinTask
    - A ForkJoinTask is lighter weight than a Java thread

E.g., it doesn’t maintain its own run-time stack, registers, thread-local storage, etc.
Overview of Java Object-Oriented Parallelism Frameworks

- The fork-join framework defines an object-oriented parallelism model
  - Provides high performance, fine-grained task execution
  - The focus is on data parallelism
  - The key abstraction is the ForkJoinTask
    - A ForkJoinTask is lighter weight than a Java thread
    - A large # of ForkJoinTasks can thus run in a small # of worker threads in a fork-join pool
Overview of Java Object-Oriented Parallelism Frameworks

• The fork-join framework defines an object-oriented parallelism model
  • Provides high performance, fine-grained task execution
  • The focus is on data parallelism
  • The key abstraction is the ForkJoinTask
  • Supports parallel programming by solving problems via “divide & conquer”

```java
solve(Problem problem) {
  if (problem is small) {
    directly solve problem
  } else {
    a. split problem into independent parts
    b. fork new sub-tasks to solve each part
    c. join all sub-tasks
    d. compose result from sub-results
  }
}
```

See en.wikipedia.org/wiki/Divide_and_conquer_algorithm
The fork-join framework defines an object-oriented parallelism model.

- Provides high performance, fine-grained task execution.
- The focus is on data parallelism.
- The key abstraction is the ForkJoinTask.
- Supports parallel programming by solving problems via “divide & conquer.”

```java
solve(Problem problem) {
    if (problem is small) {
        directly solve problem
    } else {
        a. split problem into independent parts
        b. fork new sub-tasks to solve each part
        c. join all sub-tasks
        d. compose result from sub-results
    }
}
```
solve(Problem problem) {
    if (problem is small)
        directly solve problem
    else {
        a. split problem into independent parts
        b. fork new sub-tasks to solve each part
        c. join all sub-tasks
        d. compose result from sub-results
    }
}

• The fork-join framework defines an object-oriented parallelism model
• Provides high performance, fine-grained task execution
• The focus is on data parallelism
• The key abstraction is the ForkJoinTask
• Supports parallel programming by solving problems via “divide & conquer”
The fork-join framework defines an object-oriented parallelism model:

- Provides high performance, fine-grained task execution
- The focus is on data parallelism
- The key abstraction is the `ForkJoinTask`
- Supports parallel programming by solving problems via “divide & conquer”

```java
solve(Problem problem) {
    if (problem is small)
        directly solve problem
    else {
        a. split problem into independent parts
        b. fork new sub-tasks to solve each part
        c. join all sub-tasks
        d. compose result from sub-results
    }
}
```
The fork-join framework defines an object-oriented parallelism model

- Provides high performance, fine-grained task execution
- The focus is on data parallelism
- The key abstraction is the ForkJoinTask
- Supports parallel programming by solving problems via “divide & conquer”

```
solve(Problem problem) {
    if (problem is small)
        directly solve problem
    else {
        a. split problem into independent parts
        b. fork new sub-tasks to solve each part
        c. join all sub-tasks
        d. compose result from sub-results
    }
}
```
The fork-join framework defines an object-oriented parallelism model.

- Provides high performance, fine-grained task execution.
- The focus is on data parallelism.
- The key abstraction is the ForkJoinTask.
- Supports parallel programming by solving problems via “divide & conquer”.

```java
solve(Problem problem) {
    if (problem is small)
        directly solve problem
    else {
        a. split problem into independent parts
        b. fork new sub-tasks to solve each part
        c. join all sub-tasks
        d. compose result from sub-results
    }
}
```
Evaluating the Pros & Cons of the Fork-Join Framework
Evaluating the Pros & Cons of the Fork-Join Framework

• Pros of the fork-join framework
Evaluating the Pros & Cons of the Fork-Join Framework

- Pros of the fork-join framework
  - Employs *work-stealing* to maximize multi-core processor utilization

See gee.cs.oswego.edu/dl/papers/fj.pdf
Evaluating the Pros & Cons of the Fork-Join Framework

• Pros of the fork-join framework
  • Employs *work-stealing* to maximize multi-core processor utilization
  • The common fork-join pool size can be expanded automatically via the ManagedBlocker mechanism

Evaluating the Pros & Cons of the Fork-Join Framework

- Cons of the fork-join framework
Evaluating the Pros & Cons of the Fork-Join Framework

- Cons of the fork-join framework
  - It can be tedious & error-prone to program
Evaluating the Pros & Cons of the Fork-Join Framework

- Cons of the fork-join framework
  - It can be tedious & error-prone to program, e.g.,
    - It uses a “white-box” object-oriented design based on inheritance

See [www.laputan.org/drc.html](http://www.laputan.org/drc.html)
Evaluating the Pros & Cons of the Fork-Join Framework

- Cons of the fork-join framework
  - It can be tedious & error-prone to program, e.g.,
    - It uses a “white-box” object-oriented design based on inheritance
    - It’s thus not integrated with modern Java’s functional programming features

See docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html
Evaluating the Pros & Cons of the Fork-Join Framework

- Cons of the fork-join framework
  - It can be tedious & error-prone to program, e.g.,
    - It uses a “white-box” object-oriented design based on inheritance
  - It’s thus not integrated with modern Java’s functional programming features

Overcoming these ‘cons’ motivates Java’s parallel functional programming frameworks, both of which encapsulate the Java fork-join framework
End of How Parallel Programs Are Developed in Java (Part 1)