The Java Fork-Join Pool Framework

(Part 4)

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

- Understand how the Java fork-join framework processes tasks in parallel
- Recognize the structure & functionality of the fork-join framework
- Know how the fork-join framework is implemented internally
- Recognize the key methods in the ForkJoinPool class
Learning Objectives in this Part of the Lesson

- Understand how the Java fork-join framework processes tasks in parallel
- Recognize the structure & functionality of the fork-join framework
- Know how the fork-join framework is implemented internally
- Recognize the key methods in the ForkJoinPool class
  - As well as the ForkJoinTask class hierarchy
Key Methods in Java ForkJoinPool
Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService

```java
class ForkJoinPool extends AbstractExecutorService {
    
    void execute(Runnable cmd){...}

    <T> Future<T> submit
      (Callable<T> task){...}

    <T> List<Future<T>> invokeAll
      (Collection<? extends Callable<T>> tasks){...}

    <T> T invokeAny
      (Collection<? extends Callable<T>> tasks){...}
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html)
Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService
- It therefore implements the ExecutorService methods

```java
class ForkJoinPool extends AbstractExecutorService {
    
    ... 
    
    void execute(Runnable cmd){...}

    <T> Future<T> submit 
    (Callable<T> task){...}

    <T> List<Future<T>> invokeAll 
    (Collection<? extends Callable<T>> tasks){...}

    <T> T invokeAny 
    (Collection<? extends Callable<T>> tasks){...}
```

Key Methods in Java ForkJoinPool

- ForkJoinPool extends Abstract ExecutorService
- It therefore implements the ExecutorService methods

```java
class ForkJoinPool extends AbstractExecutorService {
    
    // Key methods
    void execute(Runnable cmd){...}

    <T> Future<T> submit
    (Callable<T> task){...}

    <T> List<Future<T>> invokeAll
    (Collection<? extends Callable<T>> tasks){...}

    <T> T invokeAny
    (Collection<? extends Callable<T>> tasks){...}
}
```

However, these methods don’t leverage the powerful fork-join pool features
ForkJoinPool extends Abstract ExecutorService

It therefore implements the ExecutorService methods

It also implements key methods for non-ForkJoinTask clients

class ForkJoinPool extends AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T> task)
    {
        ...
    }
    
    T invoke(ForkJoinTask<T> task)
    {
        ...
    }
    
    ForkJoinTask<T> submit (ForkJoinTask<T> task)
    {
        ...
    }
}
ForkJoinPool extends AbstractExecutorService

- It therefore implements the ExecutorService methods

- It also implements key methods for non-ForkJoinTask clients

```java
class ForkJoinPool extends AbstractExecutorService {
    ... 
    void execute(ForkJoinTask<? extends T> task) {
        ... 
    }

    T invoke(ForkJoinTask<? extends T> task) {
        ... 
    }

    ForkJoinTask<? extends T> submit(ForkJoinTask<? extends T> task) {
        ... 
    }
}
```

These methods do leverage the powerful properties of the fork-join pool
ForkJoinPool extends AbstractExecutorService

- It therefore implements the ExecutorService methods
- It also implements key methods for non-ForkJoinTask clients
- Arrange async execution

```java
class ForkJoinPool extends AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T> task)
    {
        ...
    }

    T invoke(ForkJoinTask<T> task)
    {
        ...
    }

    ForkJoinTask<T> submit(ForkJoinTask<T> task)
    {
        ...
    }
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#execute
Key Methods in Java ForkJoinPool

- ForkJoinPool extends Abstract ExecutorService
  - It therefore implements the ExecutorService methods
- It also implements key methods for non-ForkJoinTask clients
  - Arrange async execution
  - Perform the task, blocking until it completes

```java
class ForkJoinPool extends AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T> task)
    { ... }

    T invoke(ForkJoinTask<T> task)
    { ... }

    ForkJoinTask<T> submit(ForkJoinTask<T> task)
    { ... }
}
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#invoke
Key Methods in Java ForkJoinPool

- ForkJoinPool extends Abstract ExecutorService
  - It therefore implements the ExecutorService methods
  - It also implements key methods for non-ForkJoinTask clients
    - Arrange async execution
    - Perform the task, blocking until it completes
    - Submit a ForkJoinTask for execution, returns a future

```java
class ForkJoinPool extends AbstractExecutorService {
    ... 
    void execute(ForkJoinTask<T> task) {
        ... 
    }

    T invoke(ForkJoinTask<T> task) {
        ... 
    }

    ForkJoinTask<T> submit(ForkJoinTask<T> task) {
        ... 
    }
}
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#submit
Key Methods in Java ForkJoinPool

- The ForkJoinPool size defaults to the # of cores available to the JVM

```java
class ForkJoinPool extends AbstractExecutorService {
    public ForkJoinPool() {
        this(Math.min(MAX_CAP,
                    Runtime.getRuntime().availableProcessors()),
             ...);
    }

    public ForkJoinPool
        (int parallelism) {
        this(parallelism, ...);
    }

    ...
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#ForkJoinPool](docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#ForkJoinPool)
Key Methods in Java ForkJoinPool

- The ForkJoinPool size defaults to the # of cores available to the JVM
- This size can also be controlled programmatically

```java
class ForkJoinPool extends AbstractExecutorService {
    public ForkJoinPool() {
        this(Math.min(MAX_CAP,
                        Runtime.getRuntime().
                        availableProcessors()),
             ...);
    }

    public ForkJoinPool(int parallelism) {
        this(parallelism, ...);
    }

    ...
}
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#ForkJoinPool
Key Methods in Java ForkJoinPool

- The common fork-join pool can be accessed via a static method

```java
class ForkJoinPool extends AbstractExecutorService {
    ...
    static final ForkJoinPool common;

    public static ForkJoinPool commonPool() {
        return common;
    }
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#commonPool](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#commonPool)
The common fork-join pool can be accessed via a static method.

The common pool is used by any ForkJoinTask that is not explicitly submitted to a specified pool.

```java
class ForkJoinPool extends AbstractExecutorService {
    ...  
    static final ForkJoinPool common;

    public static ForkJoinPool commonPool() {
        return common;
    }
}  
```
### Key Methods in Java ForkJoinPool

- **ForkJoinPool** also provides various management & monitoring operations

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int getParallelism()</code></td>
<td>Returns the targeted parallelism level of this pool</td>
</tr>
<tr>
<td><code>int getPoolSize()</code></td>
<td>Returns the number of worker threads that have started but not yet terminated</td>
</tr>
<tr>
<td><code>int getQueuedSubmissionCount()</code></td>
<td>Returns an estimate of the number of tasks submitted to this pool that have not yet begun executing</td>
</tr>
<tr>
<td><code>long getStealCount()</code></td>
<td>Returns an estimate of the total number of tasks stolen from one thread's work queue by another</td>
</tr>
</tbody>
</table>
Key Methods in Java ForkJoinTask
Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface

abstract class ForkJoinTask<V> implements Future<V>, Serializable {

...
Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
- fork() enables a task to create sub-tasks that run in parallel

abstract class ForkJoinTask<V>
    implements Future<V>, Serializable {
        ...
        final ForkJoinTask<V> fork() {
            ...
        }
        final V join() { ... }
        final V invoke() { ... }

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html#fork
Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
- fork() enables a task to create sub-tasks that run in parallel
- Asynchronously execute this task in the current task’s pool or ForkJoinPool.commonPool()

```java
abstract class ForkJoinTask<V> implements Future<V>, Serializable {
    ...
    final ForkJoinTask<V> fork() {
        ...
    }

    final V join() {
        ...
    }

    final V invoke() {
        ...
    }
```
ForkJoinTask implements the Future interface

- fork() enables a task to create sub-tasks that run in parallel
  - Asynchronously execute this task in the current task’s pool or ForkJoinPool.commonPool()
  - Pushes the task on the head of the deque owned by the current worker thread

Key Methods in Java ForkJoinTask:

1. fork()
2. push()
Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
  - fork() enables a task to create sub-tasks that run in parallel
  - join() returns the result of a computation when it’s done

abstract class ForkJoinTask<V>
  implements Future<V>, Serializable {
    ... 
    final ForkJoinTask<V> fork() 
    { ... } 

    final V join() { ... } 

    final V invoke() { ... }

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html#join
Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
  - fork() enables a task to create sub-tasks that run in parallel
  - join() returns the result of a computation when it’s done
    - It “blocks” the calling task until forked sub-task is done

abstract class ForkJoinTask<V>
  implements Future<V>,
  Serializable {
    ...  
    final ForkJoinTask<V> fork()  
    { ... }  
    final V join() { ... }  
    final V invoke() { ... }
Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface

  - fork() enables a task to create sub-tasks that run in parallel
  - join() returns the result of a computation when it’s done
    - It “blocks” the calling task until forked sub-task is done
  - Defines a synchronization point
    - Ensures all writes in a worker thread that “happen-before” join() are made visible to other threads after the join()

```java
abstract class ForkJoinTask<V>
    implements Future<V>, Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }
    final V join() { ... }
    final V invoke() { ... }
}
```

See stackoverflow.com/questions/4800503/memory-visibility-in-fork-join
Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
  - `fork()` enables a task to create sub-tasks that run in parallel
  - `join()` returns the result of a computation when it’s done
  - `invoke()` performs this task, awaits its completion if needed, & returns its result

abstract class ForkJoinTask<V>
  implements Future<V>, Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html#invoke
abstract class ForkJoinTask<V> implements Future<V>, Serializable {
  ...
  final ForkJoinTask<V> fork()
  { ... }

  final V join() { ... }

  final V invoke() { ... }

• ForkJoinTask implements the Future interface

• fork() enables a task to create sub-tasks that run in parallel

• join() returns the result of a computation when it’s done

• invoke() performs this task, awaits its completion if needed, & returns its result

• Throws RuntimeException or Error if the underlying computation did so
Key Methods in the Java RecursiveAction
• RecursiveAction extends the ForkJoinTask superclass & does not return a result

abstract class RecursiveAction
  extends ForkJoinTask<Void> {
  ...

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveAction.html
Key Methods in Java RecursiveAction

- RecursiveTask extends ForkJoinTask to return a result
- compute() must be overridden by subclasses to perform the task’s main computation

abstract class RecursiveAction extends ForkJoinTask<Void> {
    protected abstract Void compute();
    ...

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#compute
RecursiveTask extends ForkJoinTask to return a result

- compute() must be overridden by subclasses to perform the task’s main computation
- It may split its work up into smaller sub-tasks that are fork()’d to run in parallel

abstract class RecursiveAction extends ForkJoinTask<Void> {
    protected abstract Void compute();
    ...
}
Key Methods in Java RecursiveAction

- RecursiveTask extends ForkJoinTask to return a result
  - compute() must be overridden by subclasses to perform the task’s main computation
    - It may split its work up into smaller sub-tasks that are fork()'d to run in parallel
    - It join()'s these smaller sub-tasks but does not return a result directly
      - Result’s often stored in array

abstract class RecursiveAction extends ForkJoinTask<Void> {
    protected abstract Void compute();

...
RecursiveTask extends ForkJoinTask to return a result
- compute() must be overridden by subclasses to perform the task’s main computation
- Called internally by the fork-join pool to execute the task

abstract class RecursiveAction extends ForkJoinTask<Void> {
    protected abstract Void compute();

    protected final boolean exec(){
        compute();
        return true;
    }

    ...

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#exec](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#exec)
Key Methods in the Java RecursiveTask
RecursiveTask extends ForkJoinTask to return a result.

abstract class RecursiveTask<V> extends ForkJoinTask<V> {
    ...

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html
Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
- compute() must be overridden by subclasses to perform the task’s main computation

```java
abstract class RecursiveTask<V> extends ForkJoinTask<V> {
    protected abstract V compute();
    ...
}
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#compute
RecursiveTask extends ForkJoinTask to return a result
- compute() must be overridden by subclasses to perform the task’s main computation
- It may split its work up into smaller sub-tasks that are fork()’d to run in parallel
RecursiveTask extends ForkJoinTask to return a result

- compute() must be overridden by subclasses to perform the task’s main computation
  - It may split its work up into smaller sub-tasks that are fork()’d to run in parallel
  - It join()’s the results of these smaller sub-tasks into a collective result

```java
abstract class RecursiveTask<V> extends ForkJoinTask<V> {
    protected abstract V compute();
    ...
}
```
Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
- compute() must be overridden by subclasses to perform the task’s main computation
- Called internally by the fork-join pool to execute the task

```java
abstract class RecursiveTask<V> extends ForkJoinTask<V> {
    protected abstract V compute();

    V result;

    protected final boolean exec() {
        result = compute();
        return true;
    }

    ...
}
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#exec
End of the Java Fork-Join Pool Framework (Part 4)