Java Parallel Streams Internals: Configuring the Common Fork-Join Pool

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 parallel stream internals, e.g.
  - Know what can change & what can’t
  - Partition a data source into “chunks”
  - Process chunks in parallel via the common fork-join pool
- Configure the Java parallel stream common fork-join pool

```java
String desiredThreads = "8";
System.setProperty("java.util.concurrent.ForkJoinPool.common.parallelism", desiredThreads);
```
Configuring the Parallel Stream Common Fork-Join Pool
Configuring the Parallel Stream Common Fork-Join Pool

- By default the common ForkJoinPool has one less thread than the # of cores

System.out.println
("The parallelism in the"
 + "common fork-join pool is "
 + ForkJoinPool
  .getCommonPoolParallelism());

e.g., returns 3 on a quad-core processor

See github.com/douglascairnschmidt/LiveLessons/blob/master/SearchForkJoin
Configuring the Parallel Stream Common Fork-Join Pool

- By default the common ForkJoinPool has one less thread than the # of cores.

A parallel stream can use all cores since it uses the invoking thread, e.g., main thread.
Configuring the Parallel Stream Common Fork-Join Pool

• However, the default # of fork-join pool threads may be inadequate
Configuring the Parallel Stream Common Fork-Join Pool

- However, the default number of fork-join pool threads may be inadequate, e.g.
- Consider a parallel image downloading & processing app

Configuring the Parallel Stream Common Fork-Join Pool

- However, the default # of fork-join pool threads may be inadequate, e.g.
  - Consider a parallel image downloading & processing app

Problems may occur when trying to download more images than # of cores

These problems may range from underutilization of processor cores to deadlock.
Configuring the Parallel Stream Common Fork-Join Pool

- The common fork-join pool size can be controlled programmatically.

```java
String desiredThreads = "8";
System.setProperty("java.util.concurrent.ForkJoinPool.common.parallelism",
+ "ForkJoinPool.common.
+ "parallelism",
+ desiredThreads);
```

It's hard to estimate the total # of threads to set in the common fork-join pool.
Configuring the Parallel Stream Common Fork-Join Pool

- The common fork-join pool size can be controlled programmatically
- Setting this property affects all parallel streams in a process

```java
String desiredThreads = "8";
System.setProperty("java.util.concurrent.ForkJoinPool.common.parallelism",
+ "ForkJoinPool.common.",
+ "parallelism",
desiredThreads);
```

A pool of worker threads
Configuring the Parallel Stream Common Fork-Join Pool

- The common fork-join pool size can be controlled programmatically
  - Setting this property affects all parallel streams in a process
  - This property can be changed only before the common fork-join pool is initialized
    - i.e., it’s initialized “on-demand” the first time it’s used

String desiredThreads = "8";
System.setProperty
("java.util.concurrent.
 + "ForkJoinPool.common.
 + "parallelism",
 desiredThreads);

See en.wikipedia.org/wiki/Lazy_initialization
Configuring the Parallel Stream Common Fork-Join Pool

- The common fork-join pool size can be controlled programmatically
  - Setting this property affects all parallel streams in a process
- The ManagedBlocker interface can also be used to add worker threads to common fork-join pool temporarily

```java
SupplierManagedBlocker<T> mb = new SupplierManagedBlocker<>((supplier);

... ForkJoinPool.managedBlock(mb);
...
return mb.getResult();
```

The common fork-join pool size can be controlled programmatically:

- Setting this property affects all parallel streams in a process.

The ManagedBlocker interface can also be used to add worker threads to common fork-join pool temporarily:

- This is useful for behaviors that block on I/O and/or synchronizers.

```java
SupplierManagedBlocker<T> mb = new SupplierManagedBlocker<> (supplier);
...
ForkJoinPool.managedBlock(mb);
...
return mb.getResult();
```

Configuring the Parallel Stream Common Fork-Join Pool

- The common fork-join pool size can be controlled programmatically
  - Setting this property affects all parallel streams in a process
- The ManagedBlocker interface can also be used to add worker threads to common fork-join pool temporarily
  - This is useful for behaviors that block on I/O and/or synchronizers
  - This interface can only be used with the common fork-join pool...

```
SupplierManagedBlocker<T> mb = new SupplierManagedBlocker<> (supplier);
...
ForkJoinPool.managedBlock (mb);
...
return mb.getResult ();
```

See lessons on “The Java Fork-Join Pool: the ManagedBlocker Interface"
End of Understand Java Parallel Streams Internals: Configuring the Common Fork-Join Pool