The Java Fork-Join Pool Framework

(Part 2)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

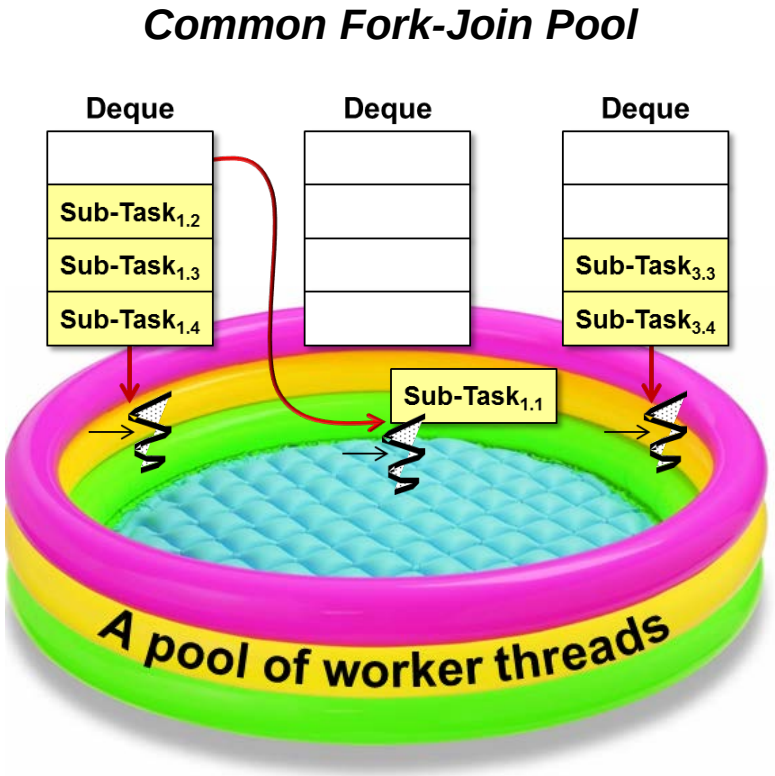
Professor of Computer Science

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
- Be aware of the common fork-join pool



Overview of Java Fork-Join Common Pool

Overview of Java Fork-Join Common Pool

- A static common pool is available & appropriate for most applications

See dzone.com/articles/common-fork-join-pool-and-streams

Overview of Java Fork-Join Common Pool

- A static common pool is available & appropriate for most applications
  - The common pool is used by any ForkJoinTask that is not explicitly submitted to a specified pool

Overview of Java Fork-Join Common Pool

- A static common pool is available & appropriate for most applications
  - The common pool is used by any ForkJoinTask that is not explicitly submitted to a specified pool
  - The common pool may optimize resource utilization since it's aware what cores are being used globally

Overview of Java Fork-Join Common Pool

- A static common pool is available & appropriate for most applications
  - The common pool is used by any ForkJoinTask that is not explicitly submitted to a specified pool
  - The common pool may optimize resource utilization since it's aware what cores are being used globally
  - This "global" vs "local" resource management tradeoff is common in computing & other domains

See blog.tsia.com/blog/local-or-global-resource-management-which-model-is-better

Overview of Java Fork-Join Common Pool

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

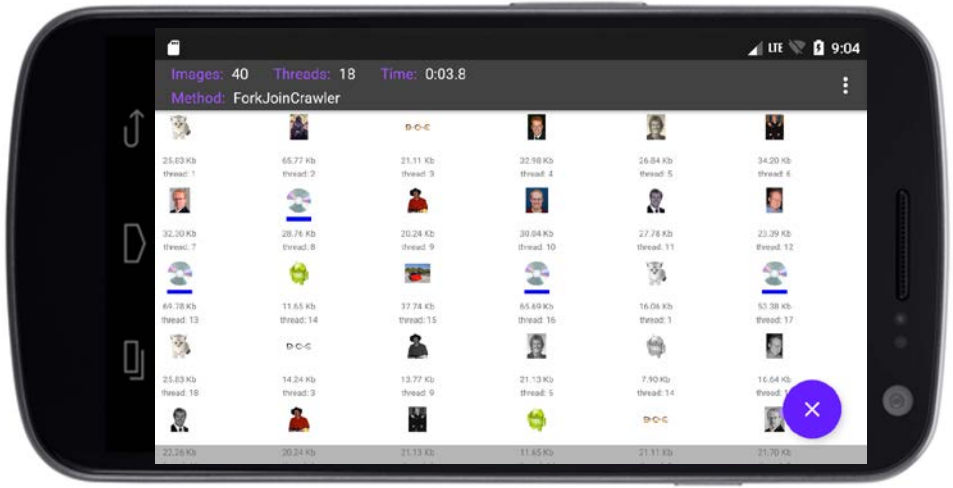
```
System.out.println
  ("The parallelism in the"
   + "common fork-join pool is "
   + ForkJoinPool
       .getCommonPoolParallelism());
```

e.g., returns 7 on my quad-core hyper-threaded processor

See github.com/douglascraigschmidt/LiveLessons/blob/master/SearchForkJoin

Overview of Java Fork-Join Common Pool

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

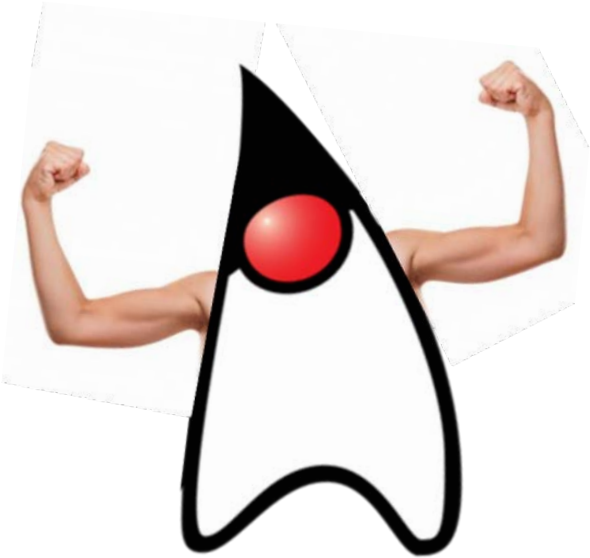


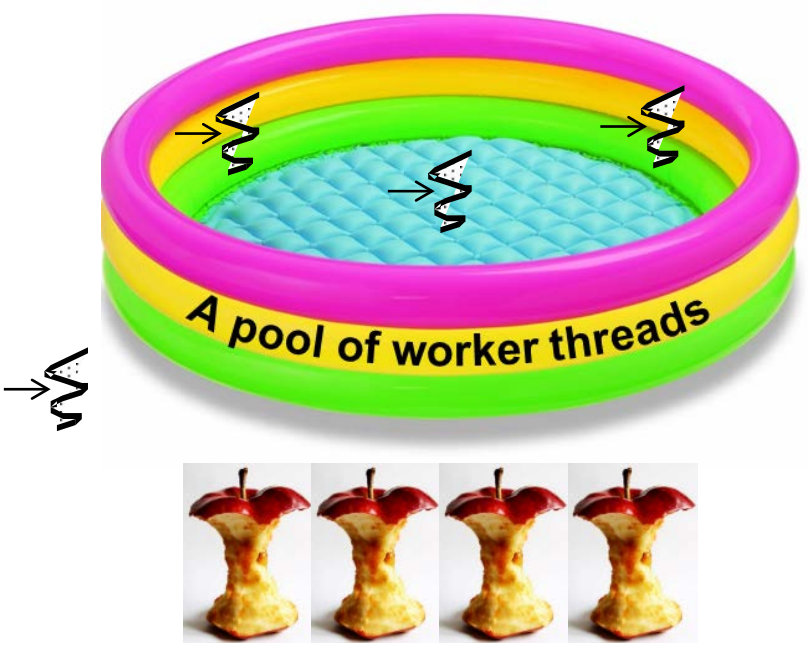
Main thread

A Java program can leverage all cores since it uses the invoking thread, e.g., main thread

Overview of Java Fork-Join Common Pool

- However, the default # of threads in the fork-join pool may be inadequate





Overview of Java Fork-Join Common Pool

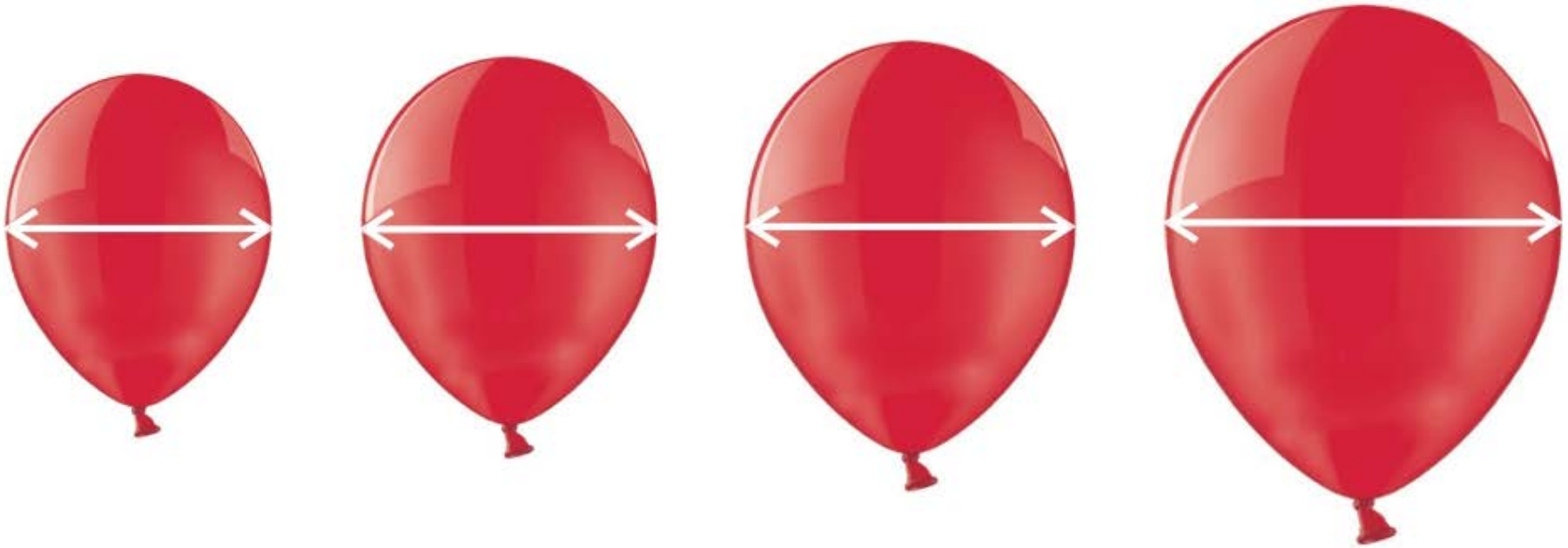
- However, the default # of threads in the fork-join pool may be inadequate
  - e.g., problems occur when blocking operations are used in a parallel stream

e.g., downloading more images than # of cores

These problems may range from underutilization of processor cores to deadlock..

Overview of Java Fork-Join Common Pool

- The common pool size can thus be expanded & contracted programmatically



Overview of Java Fork-Join Common Pool

- The common pool size can thus be expanded & contracted programmatically
  - By modifying a system property

```
System.setProperty
  ("java.util.concurrent." +
   "ForkJoinPool.common." +
   "parallelism",
   numberOfThreads);
```

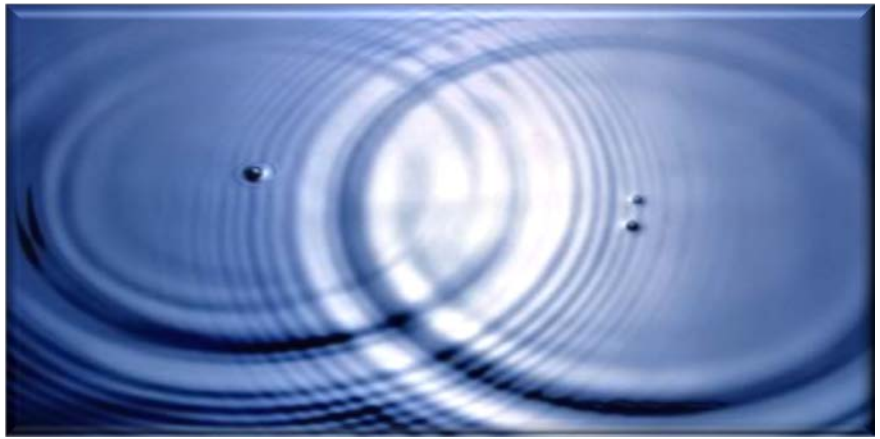
It's hard to estimate the total # of threads to set in the common fork-join pool

Overview of Java Fork-Join Common Pool

- The common pool size can thus be expanded & contracted programmatically
  - By modifying a system property

```
System.setProperty
  ("java.util.concurrent." +
   "ForkJoinPool.common." +
   "parallelism",
   numberOfThreads);
```

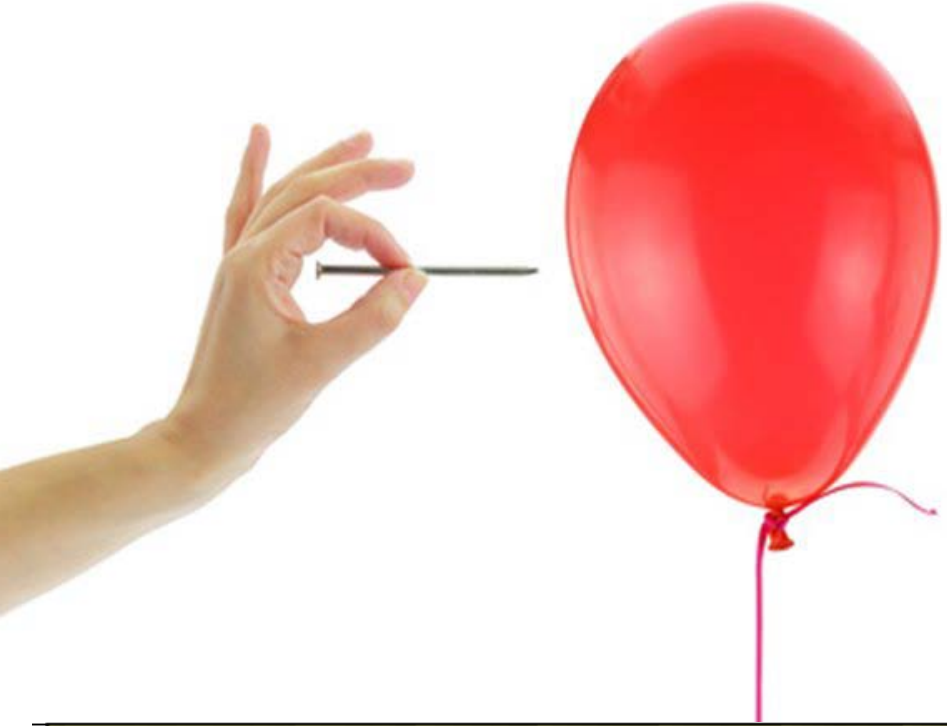
Modifying this property affects all common fork-join usage in a process!



Overview of Java Fork-Join Common Pool

- The common pool size can thus be expanded & contracted programmatically
  - By modifying a system property

```
System.setProperty
  ("java.util.concurrent." +
   "ForkJoinPool.common." +
   "parallelism",
   numberOfThreads);
```



It's thus necessary to be able to automatically increasing fork/join pool size

Overview of Java Fork-Join Common Pool

- The common pool size can thus be expanded & contracted programmatically
  - By modifying a system property
  - By using a ManagedBlocker

Interface ForkJoinPool.ManagedBlocker

Enclosing class:
`ForkJoinPool`

```
public static interface ForkJoinPool.ManagedBlocker
```

Interface for extending managed parallelism for tasks running in `ForkJoinPools`.

A `ManagedBlocker` provides two methods. Method `isReleasable()` must return `true` if blocking is not necessary. Method `block()` blocks the current thread if necessary (perhaps internally invoking `isReleasable` before actually blocking). These actions are performed by any thread invoking `ForkJoinPool.managedBlock(ManagedBlocker)`. The unusual methods in this API accommodate synchronizers that may, but don't usually, block for long periods. Similarly, they allow more efficient internal handling of cases in which additional workers may be, but usually are not, needed to ensure sufficient parallelism. Toward this end, implementations of method `isReleasable` must be amenable to repeated invocation.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.ManagedBlocker.html

Overview of Java Fork-Join Common Pool

- The common pool size can thus be expanded & contracted programmatically
  - By modifying a system property
  - By using a ManagedBlocker
    - Temporarily add worker threads to common fork-join pool

Overview of Java Fork-Join Common Pool

- The common pool size can thus be expanded & contracted programmatically
  - By modifying a system property
  - By using a ManagedBlocker
    - Temporarily add worker threads to common fork-join pool

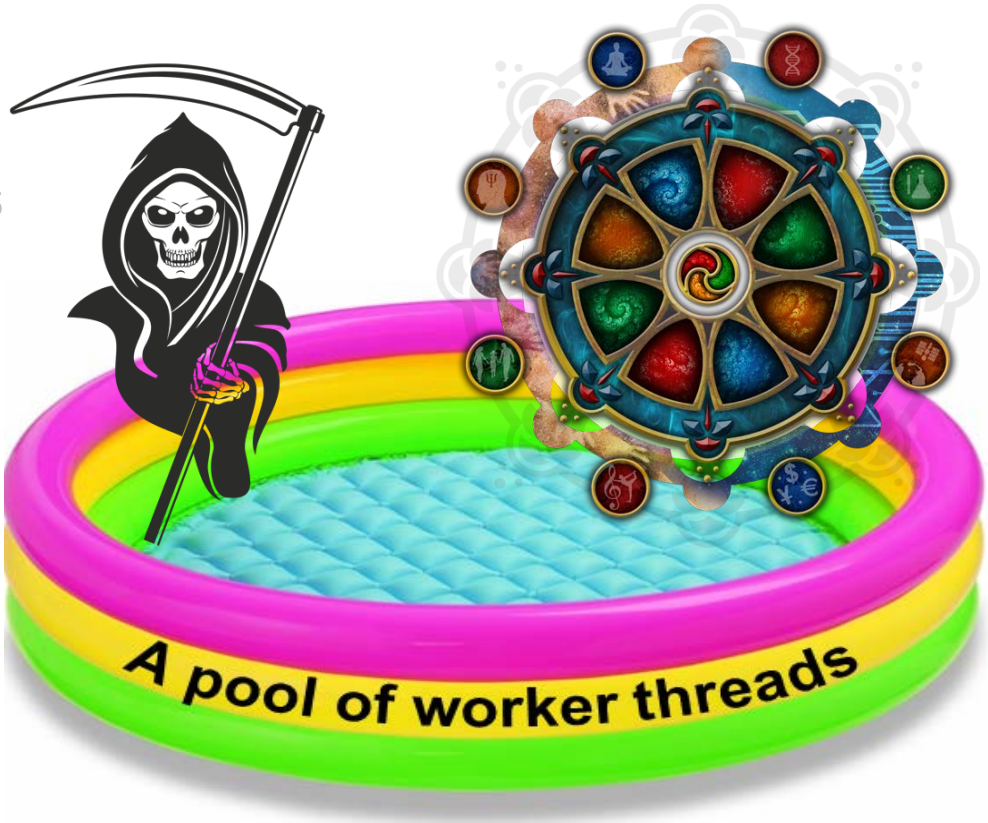
Overview of Java Fork-Join Common Pool

- The common pool size can thus be expanded & contracted programmatically
  - By modifying a system property
  - By using a ManagedBlocker
    - Temporarily add worker threads to common fork-join pool

ManageBlocker is useful for behaviors that block on I/O and/or synchronizers

Overview of Java Fork-Join Common Pool

- The common pool size can thus be expanded & contracted programmatically
  - By modifying a system property
  - By using a ManagedBlocker
    - Temporarily add worker threads to common fork-join pool
    - ForkJoinPool reclaims threads during periods of non-use & reinstates them on later use



End of the Java Fork-Join Pool Framework (Part 2)