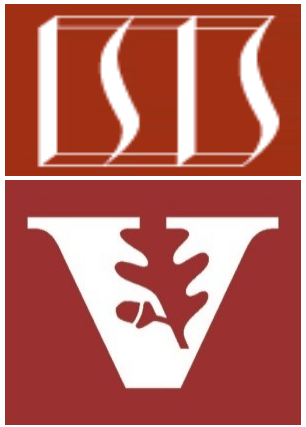
Java Parallel Streams Internals: Parallel Processing w/the Common Fork-Join Pool (Part 2)

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 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
  - Know how to apply the common fork-join pool

```
Long compute() {
  ...
  List<ForkJoinTask<Long>> forks =
    new ArrayList<>();

  for (File file : mFile.listFiles())
     forks.add(new FileCounterTask
       (file, mDocCount, mFileCount)
        .fork());

  long sum = 0;

  for (ForkJoinTask<Long> task : forks)
    sum += task.join();

  return sum; ...
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Folders/ForkJoin

Applying the Common Fork-Join Pool

Applying the Common Fork-Join Pool

- ForkJoinPool is best used for programs that don't match the parallel streams model

```
Long compute() {
  ...
  List<ForkJoinTask<Long>> forks =
    new ArrayList<>();

  for (File file : mFile.listFiles())
     forks.add(new FileCounterTask
       (file, mDocCount, mFileCount)
        .fork());

  long sum = 0;

  for (ForkJoinTask<Long> task : forks)
    sum += task.join();

  return sum; ...
```

See www.oracle.com/technetwork/articles/java/fork-join-422606.html

Applying the Common Fork-Join Pool

- ForkJoinPool is best used for programs that don't match the parallel streams model
  - e.g., this program sums the size of all files reachable from a root folder in a recursive directory hierarchy

```
Long compute() {
  ...
  List<ForkJoinTask<Long>> forks =
    new ArrayList<>();

  for (File file : mFile.listFiles())
     forks.add(new FileCounterTask
       (file, mDocCount, mFileCount)
        .fork());

  long sum = 0;

  for (ForkJoinTask<Long> task : forks)
    sum += task.join();

  return sum; ...
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Folders/ForkJoin

Applying the Common Fork-Join Pool

- ForkJoinPool is best used for programs that don't match the parallel streams model
  - e.g., this program sums the size of all files reachable from a root folder in a recursive directory hierarchy

Create an ArrayList of recursive task objects

```
Long compute() {
  ...
  List<ForkJoinTask<Long>> forks =
    new ArrayList<>();

  for (File file : mFile.listFiles())
     forks.add(new FileCounterTask
       (file, mDocCount, mFileCount)
        .fork());

  long sum = 0;

  for (ForkJoinTask<Long> task : forks)
    sum += task.join();

  return sum; ...
```

Applying the Common Fork-Join Pool

- ForkJoinPool is best used for programs that don't match the parallel streams model
  - e.g., this program sums the size of all files reachable from a root folder in a recursive directory hierarchy

Create & fork tasks to search folders recursively

```
Long compute() {
  ...
  List<ForkJoinTask<Long>> forks =
    new ArrayList<>();

  for (File file : mFile.listFiles())
     forks.add(new FileCounterTask
       (file, mDocCount, mFileCount)
        .fork());

  long sum = 0;

  for (ForkJoinTask<Long> task : forks)
    sum += task.join();

  return sum; ...
```

Applying the Common Fork-Join Pool

- ForkJoinPool is best used for programs that don't match the parallel streams model
  - e.g., this program sums the size of all files reachable from a root folder in a recursive directory hierarchy

Join all the tasks together & sum the # of search matches

```
Long compute() {
  ...
  List<ForkJoinTask<Long>> forks =
    new ArrayList<>();

  for (File file : mFile.listFiles())
     forks.add(new FileCounterTask
       (file, mDocCount, mFileCount)
        .fork());

  long sum = 0;

  for (ForkJoinTask<Long> task : forks)
    sum += task.join();

  return sum; ...
```

Applying the Common Fork-Join Pool

- ForkJoinPool is best used for programs that don't match the parallel streams model
  - e.g., this program sums the size of all files reachable from a root folder in a recursive directory hierarchy

```
Long compute() {
  ...
  List<ForkJoinTask<Long>> forks =
    new ArrayList<>();

  for (File file : mFile.listFiles())
     forks.add(new FileCounterTask
       (file, mDocCount, mFileCount)
        .fork());

  long sum = 0;

  for (ForkJoinTask<Long> task : forks)
    sum += task.join();

  return sum; ...
```

Return the final sum

End of Java Parallel Streams Internals: Parallel Processing w/the Common Fork-Join Pool (Part 2)