Java Parallel Streams Internals: Demo'ing Spliterator Performance

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"
    - Via a parallel spliterator
    - Know the impact of different Java collections on the performance of different spliterators

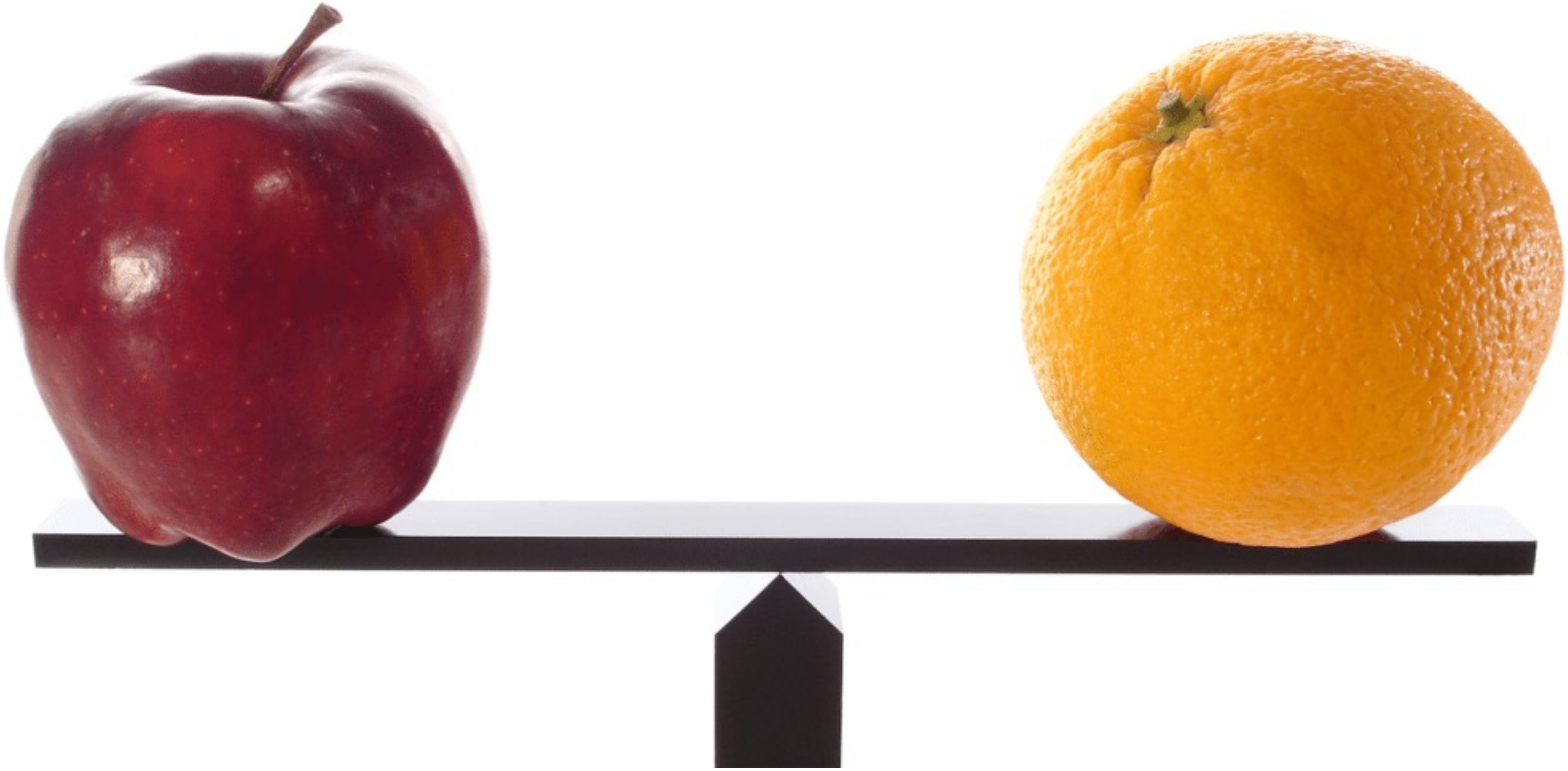
```
Starting spliterator tests for 10000 words..
..printing results
     1 msecs: LinkedList sequential
     1 msecs: ArrayList sequential
     7 msecs: ArrayList parallel
    12 msecs: LinkedList parallel

Starting spliterator tests for 100000 words..
..printing results
     3 msecs: ArrayList parallel
     5 msecs: ArrayList sequential
     6 msecs: LinkedList sequential
    19 msecs: LinkedList parallel

Starting spliterator tests for 910654 words..
..printing results
    12 msecs: ArrayList parallel
    14 msecs: LinkedList parallel
    38 msecs: LinkedList sequential
    43 msecs: ArrayList sequential
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex14

Demonstrating Spliterator Performance

Demonstrating Spliterator Performance

- Spliterators for ArrayList & LinkedList partition data quite differently



See earlier lesson on "*Java Parallel Streams Internals: Partitioning*"

Demonstrating Spliterator Performance

- Spliterators for ArrayList & LinkedList partition data quite differently

```
ArrayListSpliterator<E> trySplit() {
  int hi = getFence(), lo = index, mid = (lo + hi) >>> 1;
  // divide range in half unless too small
  return lo >= mid ? null : new ArrayListSpliterator<E>
                                (list, lo, index = mid, ...);
}
```

ArrayList's spliterator splits evenly & efficiently (e.g., doesn't copy data)

See openjdk/8u40-b25/java/util/ArrayList.java

Demonstrating Spliterator Performance

- Spliterators for ArrayList & LinkedList partition data quite differently

```
Spliterator<E> trySplit() { ...
  int n = batch + BATCH_UNIT, j = 0; Object[] a = new Object[n];
  do { a[j++] = p.item; }
  while ((p = p.next) != null && j < n); ...
  return Spliterators.spliterator(a, 0, j, Spliterator.ORDERED);
}
```

LinkedList's spliterator does not split evenly & efficiently (e.g., it copies data)

See [openjdk/8u40-b25/java/util/LinkedList.java]()

Demonstrating Spliterator Performance

- This demo program shows the performance difference of parallel spliterators for ArrayList & LinkedList when processing the complete works of Shakepeare

```
void timeStreamModifications
   (String testName, List<String> words, boolean parallel) {
    ...
    IntStream.range(0, sMAX_ITERATIONS)
             .boxed()
             .mapMulti((i, consumer) -> consumer.accept
                ((parallel ? words.parallelStream()
                           : words.stream())
                  .map(s -> rot13(s.toUpperCase())
                                 .toLowerCase())
                  .toList()))
             .count(); ...
```

Demonstrating Spliterator Performance

- This demo program shows the performance difference of parallel spliterators for ArrayList & LinkedList when processing the complete works of Shakepeare

```
void timeStreamModifications
   (String testName, List<String> words, boolean parallel) {
    ...
    IntStream.range(0, sMAX_ITERATIONS)
             .boxed()
             .mapMulti((i, consumer) -> consumer.accept
                ((parallel ? words.parallelStream()
                           : words.stream())
                 .map(s -> rot13(s.toUpperCase())
                                  .toLowerCase())
                 .toList()))
             .count(); ...
```

The words param is passed an ArrayList & a LinkedList

Demonstrating Spliterator Performance

- This demo program shows the performance difference of parallel spliterators for ArrayList & LinkedList when processing the complete works of Shakepeare

```
void timeStreamModifications
   (String testName, List<String> words, boolean parallel) {
    ...
    IntStream.range(0, sMAX_ITERATIONS)
             .boxed()
             .mapMulti((i, consumer) -> consumer.accept
                ((parallel ? words.parallelStream()
                           : words.stream())
                 .map(s -> rot13(s.toUpperCase())
                                 .toLowerCase())
                 .toList()))
             .count(); ...
```

Split & modify words list via a spliterator using mapMulti/()

See docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/stream/Stream.html#mapMulti

Demonstrating Spliterator Performance

- This demo program shows the performance difference of parallel spliterators for ArrayList & LinkedList when processing the complete works of Shakepeare

```
void timeStreamModifications
   (String testName, List<String> words, boolean parallel) {
    ...
    IntStream.range(0, sMAX_ITERATIONS)
             .boxed()
             .mapMulti((i, consumer) -> consumer.accept
                ((parallel ? words.parallelStream()
                           : words.stream())
                 .map(s -> rot13(s.toUpperCase()
                                  .toLowerCase())
                 .toList()))
             .count(); ...
```

Conditionally select a parallel or sequential spliterator

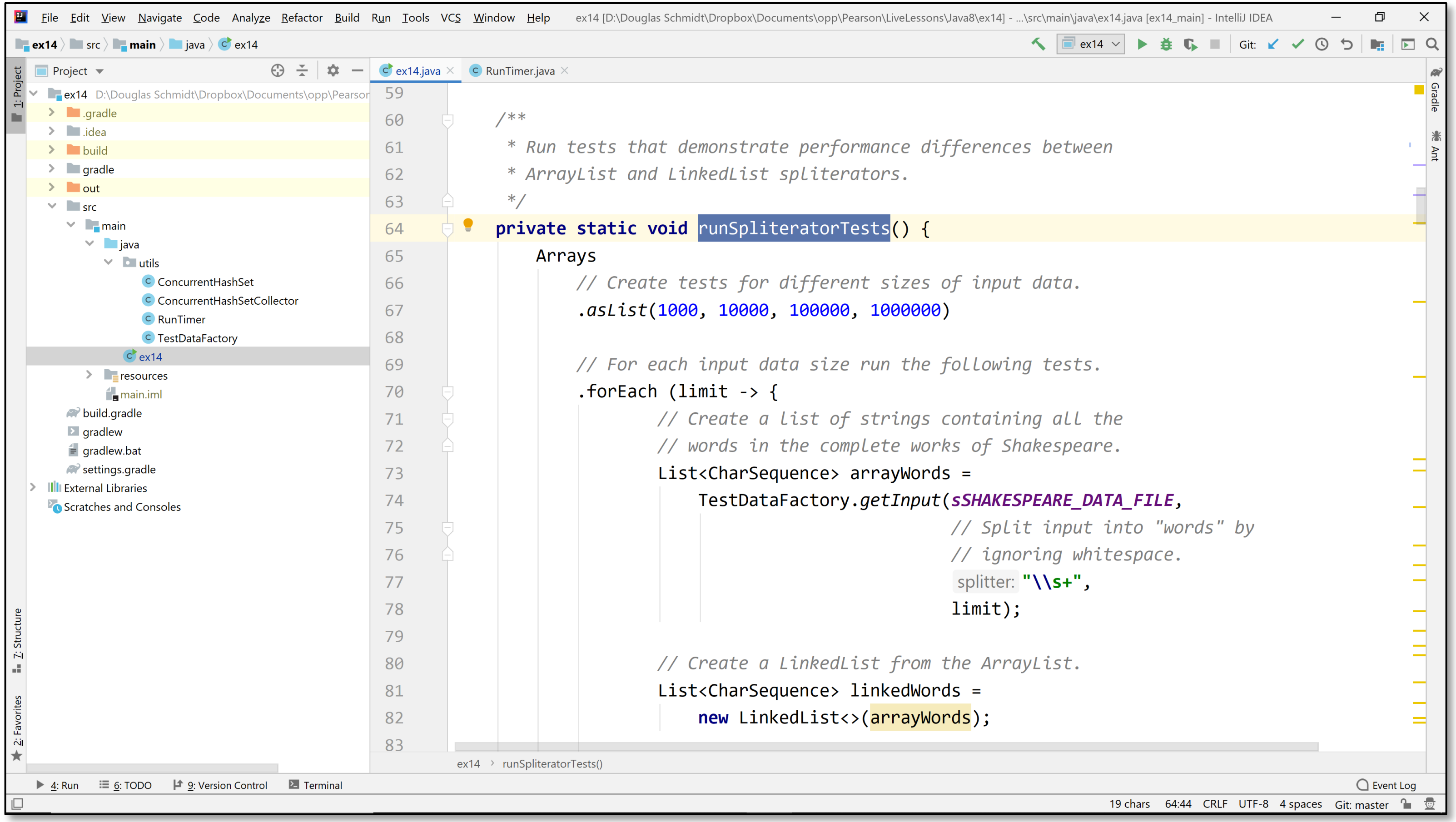
Demonstrating Spliterator Performance

- Results show spliterator differences become more significant as input grows

```
...
Starting spliterator tests for 100000 words..
..printing results
     3 msecs: ArrayList parallel
     5 msecs: ArrayList sequential
     6 msecs: LinkedList sequential
    19 msecs: LinkedList parallel

Starting spliterator tests for 883311 words..
..printing results
    12 msecs: ArrayList parallel
    14 msecs: LinkedList parallel
    38 msecs: LinkedList sequential
    43 msecs: ArrayList sequential
```

See upcoming lessons on "*When [Not] to Use Parallel Streams*"

Demonstrating Spliterator Performance



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ex14 [D:\Douglas Schmidt\Dropbox\Documents\opp\Pearson\LiveLessons\Java8\ex14] - ...\src\main\java\ex14.java [ex14_main] - IntelliJ IDEA
ex14 src main java ex14
Project
ex14 D:\Douglas Schmidt\Dropbox\Documents\opp\Pearson
.gradle
.idea
build
gradle
out
src
main
java
utils
ConcurrentHashSet
ConcurrentHashSetCollector
RunTimer
TestDataFactory
ex14
resources
main.iml
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
ex14.java
RunTimer.java
ex14 › runSpliteratorTests()
4: Run
6: TODO
9: Version Control
Terminal
Event Log
19 chars 64:44 CRLF UTF-8 4 spaces Git: master

```java

    /**
     * Run tests that demonstrate performance differences between
     * ArrayList and LinkedList spliterators.
     */
    private static void runSpliteratorTests() {
        Arrays
            // Create tests for different sizes of input data.
            .asList(1000, 10000, 100000, 1000000)

            // For each input data size run the following tests.
            .forEach (limit -> {
                    // Create a list of strings containing all the
                    // words in the complete works of Shakespeare.
                    List<CharSequence> arrayWords =
                        TestDataFactory.getInput(sSHAKESPEARE_DATA_FILE,
                                                 // Split input into "words" by
                                                 // ignoring whitespace.
                                                 splitter: "\\s+",
                                                 limit);

                    // Create a LinkedList from the ArrayList.
                    List<CharSequence> linkedWords =
                        new LinkedList<>(arrayWords);

```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex14

End of Java Parallel Streams Internals: Demo'ing Spliterator Performance