Java Parallel Streams Internals:

Introduction

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

See developer.ibm.com/languages/java/articles/j-java-streams-3-brian-goetz
Learning Objectives in this Part of the Lesson

- Understand parallel stream internals, e.g.
- Know what can change & what can’t

See en.wikipedia.org/wiki/Serenity_Prayer

Grant me the Serenity to accept the things I cannot change, the Courage to change the things I can and the Wisdom to know the difference.
Why Knowledge of Parallel Streams Matters
Why Knowledge of Parallel Streams Matters

- Converting a Java sequential stream to a parallel stream is usually quite straightforward.

```java
List<List<SearchResults>> processStream() {
    return getInput().stream()
        .map(this::processInput)
        .collect(toList());
}
```

**Changing stream() calls to parallelStream() calls involves minuscule effort!!**

```java
List<List<SearchResults>> processStream() {
    return getInput().parallelStream()
        .map(this::processInput)
        .collect(toList());
}
```

See "Java SearchWithParallelStreams Example"
Why Knowledge of Parallel Streams Matters

- However, knowledge of parallel streams internals will make you a better Java streams programmer!

*When performance is critical, it's important to understand how streams work internally*

See [developer.ibm.com/languages/java/articles/j-java-streams-3-brian-goetz](http://developer.ibm.com/languages/java/articles/j-java-streams-3-brian-goetz)
### Why Knowledge of Parallel Streams Matters

- Recall the 3 phases of a Java parallel stream

<table>
<thead>
<tr>
<th>Input x</th>
<th>Intermediate operation (behavior f)</th>
<th>Output f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate operation (behavior g)</td>
<td>Output g(f(x))</td>
<td></td>
</tr>
<tr>
<td>Terminal operation (reducer)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See [docs.oracle.com/javase/tutorial/collections STREAMS/parallelism.html](docs.oracle.com/javase/tutorial/collections/streams/parallelism.html)
Why Knowledge of Parallel Streams Matters

- Recall the 3 phases of a Java parallel stream
- *Split* – Uses a spliterator to partition a data source into multiple chunks
Why Knowledge of Parallel Streams Matters

- Recall the 3 phases of a Java parallel stream
  - *Split* – Uses a spliterator to partition a data source into multiple chunks
  - *Apply* – Independently processes these chunks in the common fork-join pool
Why Knowledge of Parallel Streams Matters

- Recall the 3 phases of a Java parallel stream
  - *Split* – Uses a spliterator to partition a data source into multiple chunks
  - *Apply* – Independently processes these chunks in the common fork-join pool
  - *Combine* – Joins partial sub-results into a single result
Why Knowledge of Parallel Streams Matters

- Recall the 3 phases of a Java parallel stream
  - **Split** – Uses a spliterator to partition a data source into multiple chunks
  - **Apply** – Independently processes these chunks in the common fork-join pool
  - **Combine** – Joins partial sub-results into a single result

It’s important to which of these phases you can control & which you can’t!
End of Java Parallel Stream Internals: Introduction