Understand 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

God
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

See en.wikipedia.org/wiki/Serenity_Prayer
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());
}
```

vs

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

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

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
Why Knowledge of Parallel Streams Matters

- Recall the 3 phases of a Java parallel stream

See 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 Understand Java
Parallel Stream Internals: Introduction