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 change wrt splitting, applying, & combining

See en.wikipedia.org/wiki/Serenity_Prayer
Why Knowledge of Parallel Streams Matters
Converting a Java sequential stream to a parallel stream is usually quite straightforward. Why Knowledge of Parallel Streams Matters

List<List<SearchResults>>

```
processStream() {
    return getInput()
        .stream()
        .map(this::processInput)
        .collect(toList());
}
```

vs

```
List<List<SearchResults>>

processStream() {
    return getInput()
        .parallelStream()
        .map(this::processInput)
        .collect(toList());
}
```

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

See prior lesson on "Java SearchWithParallelStreams Example"
Why Knowledge of Parallel Streams Matters

- Converting a Java sequential stream to a parallel stream is usually quite straightforward
- However, just because creating a parallel stream is easy doesn’t mean it’s the right thing to do!

```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());
}
```

See upcoming lesson on "When to Not to Use Java Parallel Streams"
Why Knowledge of Parallel Streams Matters

- Therefore, 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](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

  Stream factory operation ()

  Input x

  Intermediate operation (behavior f)

  Output f(x)

  Intermediate operation (behavior g)

  Output g(f(x))

  Terminal operation (reducer)

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

Programmers have a great degree of control over this phase
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

Programmers have a limited amount of control over this phase
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

Programmers have a great degree of control over this phase.
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

Knowing which phases you can control & which you can’t can be very important!
End of Java Parallel Stream Internals: Introduction