Java SearchWithParallelStreams

Example: 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

- Know how Java parallel streams are applied in `SearchWithParallelStreams`

```
<<Java Class>>

SearchWithParallelStreams

* processStream(): List<List<SearchResults>>
* processInput(CharArraySequence): List<SearchResults>
```

Starting SearchStreamGangTest
PARALLEL_SPLITTERATOR executed in 409 msecs
COMPLETABLE_FUTURES_INPUTS executed in 426 msecs
COMPLETABLE_FUTURES_PHASES executed in 427 msecs
PARALLEL_STREAMS executed in 437 msecs
PARALLEL_STREAM_PHASES executed in 440 msecs
RXJAVA_PHASES executed in 485 msecs
PARALLEL_STREAM_INPUTS executed in 802 msecs
RXJAVA_INPUTS executed in 866 msecs
SEQUENTIAL_LOOPS executed in 1638 msecs
SEQUENTIAL_STREAM executed in 1958 msecs
Ending SearchStreamGangTest

Applying Parallel Streams to SearchStreamGang
Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in `processStream()` & `processInput()` from `SearchWithParallelStreams`

<table>
<thead>
<tr>
<th>Java Class</th>
<th>SearchWithParallelStreams</th>
</tr>
</thead>
<tbody>
<tr>
<td>processStream()</td>
<td>List&lt;List&lt;SearchResults&gt;&gt;</td>
</tr>
<tr>
<td>processInput( CharSequence )</td>
<td>List&lt;Search Results&gt;</td>
</tr>
</tbody>
</table>

We focus on parallel streams in `processStream()` & `processInput()` from `SearchWithParallelStreams`. See `SearchStreamGang/src/main/java/livelessons/streamgangs/SearchWithParallelStreams.java`.

```java
getInput()
 .parallelStream()
 .map(this::processInput)
 .collect(toList());
```

```java
return mPhrasesToFind
 .parallelStream()
 .map(phrase -> searchForPhrase(phrase, input, title, false))
 .filter(not(SearchResults::isEmpty)
 .collect(toList());
```

**Applying Parallel Streams to SearchStreamGang**
Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in `processStream()` & `processInput()` from `SearchWithParallelStreams`.

```java
getInput() .parallelStream() .map(this::processInput) .collect(toList());
```

```
return mPhrasesToFind .parallelStream() .map(phrase -> searchForPhrase(phrase, input, title, false)) .filter(not(SearchResults::isEmpty)) .collect(toList());
```

i.e., the map(), filter(), & collect() aggregate operations
Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in processStream() & processInput() from SearchWithParallelStreams
  - **processStream()**
    - Uses a parallel stream to search a list of input strings

  Each input string contains a work of Shakespeare (e.g., Hamlet, MacBeth, etc.)

- This parallel stream uses the common fork-join pool of worker threads
Applying Parallel Streams to SearchStreamGang

• We focus on parallel streams in `processStream()` & `processInput()` from `SearchWithParallelStreams`

• `processStream()`
  • Uses a parallel stream to search a list of input strings

Returns a list of lists of `SearchResults`
We focus on parallel streams in `processStream()` & `processInput()` from `SearchWithParallelStreams`.

- **processStream()**
- **processInput()**
  - Uses a parallel stream to search each input string & locate all occurrences of phases.

This parallel stream also uses the common fork-join pool of worker threads.
Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in `processStream()` & `processInput()` from SearchWithParallelStreams
  - `processStream()`
  - `processInput()`
    - Uses a parallel stream to search each input string & locate all occurrences of phases

\[
\text{parallelStream() -> map(phrase -> searchForPhrase(...) -> filter(not(SearchResults::isEmpty)) -> collect(toList()))}
\]

Returns a list of SearchResults
End of Java SearchWithParallelStreams Example: Introduction