Understand the Java Sequential SearchStreamGang Case Study

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 the design of the SearchStreamGang program

Learning Objectives in this Part of the Lesson

• Understand the design of the SearchStreamGang program

This example is more interesting than the SimpleSearchStream program.
Learning Objectives in this Part of the Lesson

• Understand the design of the SearchStreamGang program
• Later we’ll cover the performance of different implementation strategies

See github.com/douglascraigschmidt/LiveLessons/tree/master/SearchStreamGang

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

Search Phrases

stream()

map(phrase -> searchForPhrase(…))

filter(not(SearchResults::isEmpty))

collect(toList())
Overview of SearchStreamGang
Overview of SearchStreamGang

- SearchStreamGang revises SearchTaskGang to use functional programming & streams instead of OO programming.
Overview of SearchStreamGang

- SearchStreamGang revises SearchTaskGang to use functional programming & streams instead of OO programming.
- SearchTaskGang showcases the Java executor framework for tasks that are "embarrassingly parallel".
Overview of SearchStreamGang

- SearchStreamGang is also a more powerful revision of SimpleSearchStream

Input Strings to Search

Search Phrases

map(phrase -> searchForPhrase(…))

filter(not(SearchResults::isEmpty))

collect(toList())

Search Words

"do", "re", "mi", "fa", "so", "la", "ti", "do"

map(word -> searchForWord(…))

filter(not(SearchResults::isEmpty))

collect(toList())

See [github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream](https://github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream)
Overview of SearchStreamGang

- SearchStreamGang is also a more powerful revision of SimpleSearchStream, e.g.
- It uses regular expressions to find phrases in works of Shakespeare

The Complete Works of William Shakespeare

Welcome to the Web's first edition of the Complete Works of William Shakespeare. This site has offered Shakespeare's plays and poetry to the Internet community since 1993.

For other Shakespeare resources, visit the Mr. William Shakespeare and the Internet Web site.

The original electronic source for this server was the Complete Moby(tm) Shakespeare. The HTML versions of the plays provided here are placed in the public domain.

See shakespeare.mit.edu
Overview of SearchStreamGang

- SearchStreamGang is also a more powerful revision of SimpleSearchStream, e.g.
  - It uses regular expressions to find phrases in works of Shakespeare

  “...
  My liege, and madam, to expostulate
  What majesty should be, what duty is,
  Why day is day, night is night, and time is time.
  Were nothing but to waste night, day, and time.
  Therefore, since **brevity is the soul of wit**,  
  And tediousness the limbs and outward flourishes,
  I will be brief. ...”

  **“Brevity is the soul of wit”**
Overview of SearchStreamGang

- SearchStreamGang is also a more powerful revision of SimpleSearchStream, e.g.
- It uses regular expressions to find phrases in works of Shakespeare

```
"... What's in a name? That which we call a rose
By any other name would smell as sweet.
So Romeo would, were he not Romeo call'd,
Retain that dear perfection which he owes
Without that title. ..."
```

"What’s in a name? That which we call a rose
By any other name would smell as sweet."

The phrases can also match across multiple lines
Overview of SearchStreamGang

- SearchStreamGang is also a more powerful revision of SimpleSearchStream, e.g.,
- It uses regular expressions to find phrases in works of Shakespeare
- It defines a framework for apples-to-apples comparisons of Java’s parallelism strategies.

\[\text{e.g., parallel streams, parallel spliterator, fork-join pool, & completable futures}\]
Overview of SearchStreamGang

- SearchStreamGang is also a more powerful revision of SimpleSearchStream, e.g.
  - It uses regular expressions to find phrases in works of Shakespeare
  - It defines a framework for apples-to-apples comparisons of Java’s parallelism strategies.

The differences in performance is quite informative!
Overview of SearchStreamGang

- SearchStreamGang is also a more powerful revision of SimpleSearchStream, e.g.
  - It uses regular expressions to find phrases in works of Shakespeare
  - It defines a framework for apples-to-apples comparisons of Java’s parallelism strategies.
- We’ll cover the Java parallel strategies after first covering sequential streams.
Overview of SearchStreamGang

- SearchStreamGang is also a more powerful revision of SimpleSearchStream, e.g.
  - It uses regular expressions to find phrases in works of Shakespeare
  - It defines a framework for apples-to-apples comparisons of Java’s parallelism strategies.
- We’ll cover the Java parallel strategies after first covering sequential streams.

Minuscule changes are needed to transition from sequential to parallel streams!
Applying Sequential Streams to SearchStreamGang
Applying Sequential Streams to SearchStreamGang

- We show aggregate operations in the SearchStreamGang’s processStream() & processInput() methods

We show aggregate operations in the SearchStreamGang’s processStream() & processInput() methods

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

See livelessons/streamgangs/SearchWithSequentialStreams.java
We show aggregate operations in the SearchStreamGang’s processStream() & processInput() methods

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

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

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

- We show aggregate operations in the SearchStreamGang’s `processStream()` & `processInput()` methods

- `processStream()`
  - Uses a sequential stream to search a list of input strings in one thread

Each input string contains a work of Shakespeare (e.g., Hamlet, MacBeth, etc.)
Applying Sequential Streams to SearchStreamGang

- We show aggregate operations in the SearchStreamGang’s processStream() & processInput() methods

  - **processStream()**
    - Uses a sequential stream to search a list of input strings in one thread

Returns a list of lists of SearchResults
Applying Sequential Streams to SearchStreamGang

- We show aggregate operations in the SearchStreamGang’s processStream() & processInput() methods
  - processStream()
  - processInput()
    - Uses a sequential stream to search a given input string & locate all the occurrences of phases in one thread

```
stream()
  map(phrase -> searchForPhrase(...))
  filter(not(SearchResults::isEmpty))
  collect(toList())
```
We show aggregate operations in the SearchStreamGang’s `processStream()` & `processInput()` methods

- **processStream()**
- **processInput()**

  Uses a sequential stream to search a given input string & locate all the occurrences of phrases in one thread

*Returns a list of SearchResults*
End of Understand the Java Sequential SearchStreamGang Case Study