Java Parallel Streams Internals:
Order of Processing Overview

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, e.g.
• Know what can change & what can’t
  • Splitting, combining, & pooling mechanisms
• Order of processing
Java Parallel Stream Processing Order
Java Parallel Stream Processing Order

• The Java parallel streams framework allows for variability in the order of its processing, while still being deterministic in the processing results.

Non-deterministic processing order

Deterministic processing results
The order in which chunks in a parallel stream are processed is non-deterministic.

See en.wikipedia.org/wiki/Nondeterministic_algorithm
The order in which chunks in a parallel stream are processed is non-deterministic.

The ordering can exhibit different behaviors on different runs, even for the same input.
The order in which chunks in a parallel stream are processed is non-deterministic.

Programmers have little/no control over how chunks are processed.

**Java Parallel Stream Processing Order**

- **Input x**
- **Intermediate operation (behavior f)**: Output f(x)
- **Intermediate operation (behavior g)**: Output g(f(x))
- **Terminal operation (reducer)**

**Diagram Elements**

- **Stream factory operation ()**
- **Input x**
- **Intermediate operation (behavior f)**: f(x)
- **Intermediate operation (behavior g)**: g(f(x))
- **Terminal operation (reducer)**

**Additional Text**

- **KEEP CALM AND LOSE CONTROL**

**Image**

- A diagram illustrating the flow of data through a stream, starting with input x, followed by intermediate operations (behavior f and g), and ending with a terminal operation (reducer).
Java Parallel Stream Processing Order

- The *order* in which chunks in a parallel stream are processed is non-deterministic.
- Programmers have little/no control over how chunks are processed.
- Non-determinism enables optimizations at multiple layers!

Applications

- Additional Frameworks & Languages
- Threading & Synchronization Packages
- Java Execution Environment (e.g., JVM)
- System Libraries
- Operating System Kernel

e.g., scheduling & execution of tasks via fork-join pool, JVM, hardware cores, etc.
Java Parallel Stream Processing Order

- The *order* in which chunks in a parallel stream are processed is non-deterministic
  - Programmers have little/no control over how chunks are processed
  - Non-determinism enables optimizations at multiple layers!

**Fork-Join Pool**

- e.g., fork-join framework's support for work-stealing is a non-deterministic optimization

See upcoming lessons on "The Java Fork-Join Framework"
End of Java Parallel Streams Internals: Order of Processing Overview