Comparing Java Sequential Streams with Java Parallel Streams

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

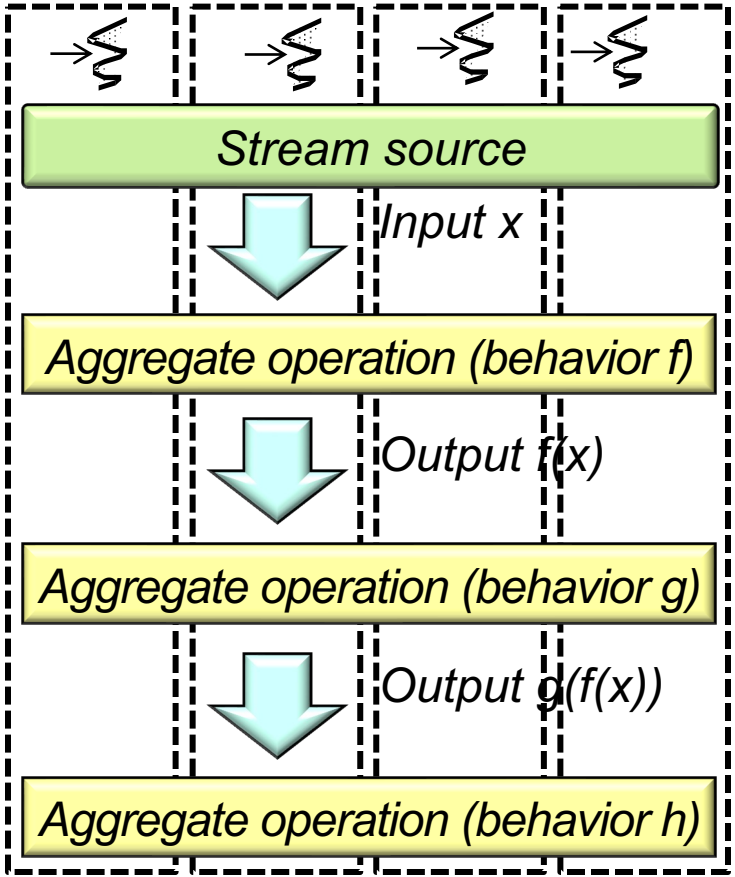
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of Java streams, e.g.,
  - Fundamentals of streams
  - Benefits of streams
  - Creating a stream
  - Aggregate operations in a stream
  - Applying streams in practice
  - Sequential vs. parallel streams

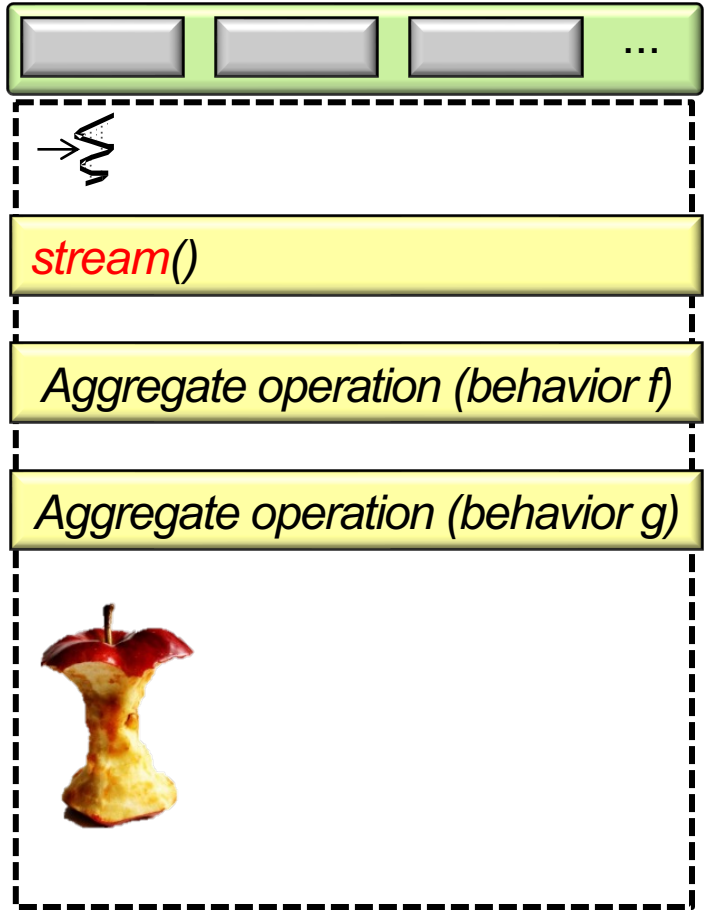


See radar.oreilly.com/2015/02/java-8-streams-api-and-parallelism.html

Comparing Sequential vs. Parallel Streams

Comparing Sequential vs. Parallel Streams

- Stream operations run sequentially

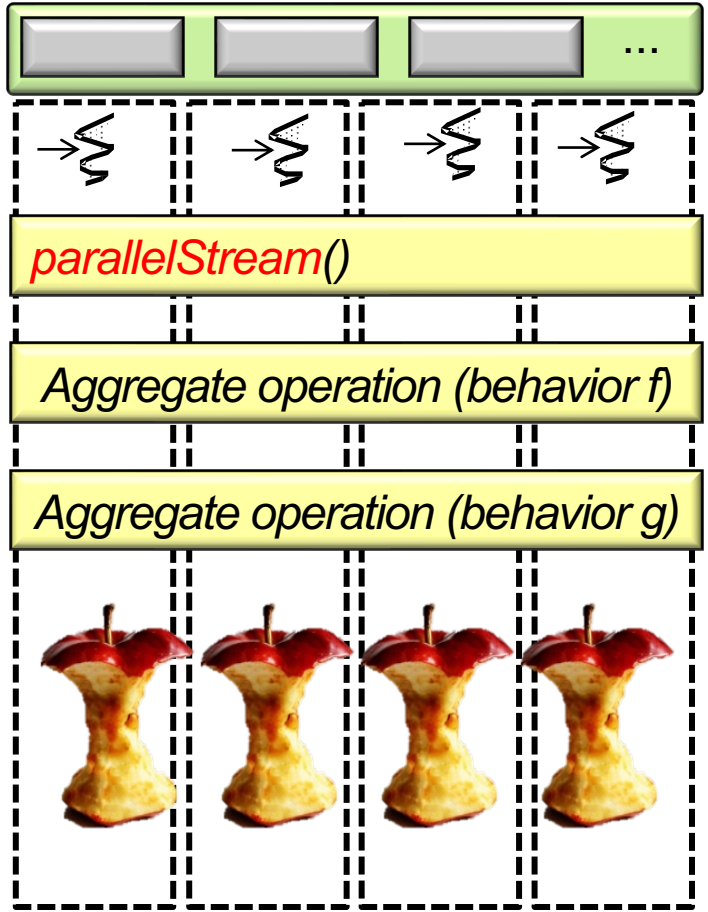


We'll cover sequential streams first

See docs.oracle.com/javase/tutorial/collections/streams

Comparing Sequential vs. Parallel Streams

- Stream operations run sequentially or in parallel

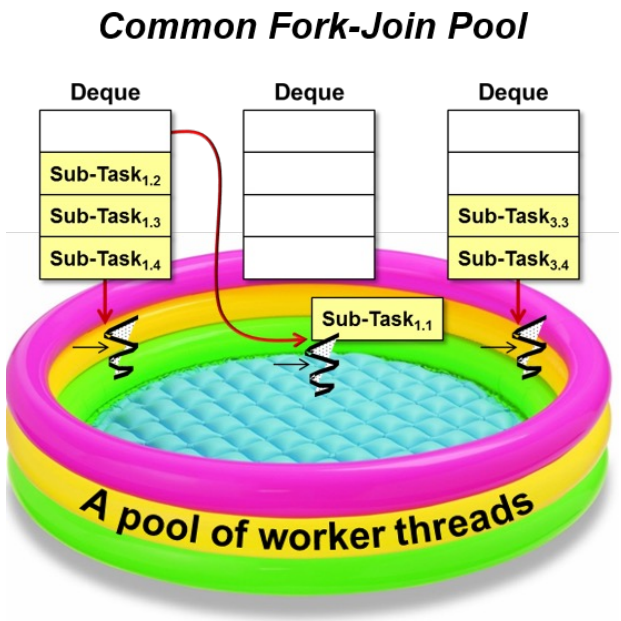


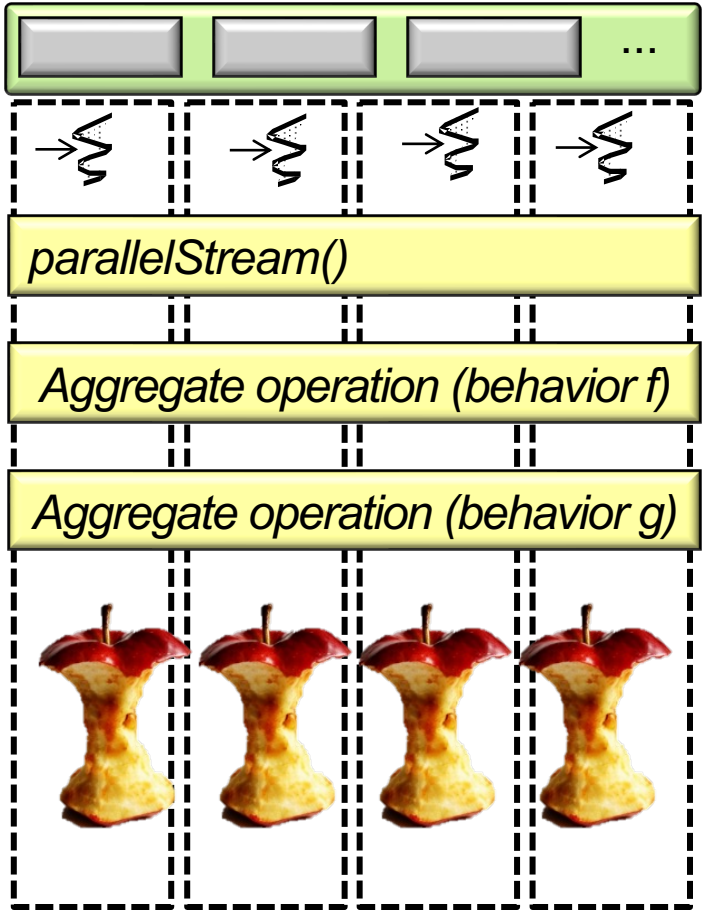
We'll cover parallel streams later

See docs.oracle.com/javase/tutorial/collections/streams/parallelism.html

Comparing Sequential vs. Parallel Streams

- A parallel stream splits its data into multiple chunks & uses the common fork-join pool to process these chunks independently





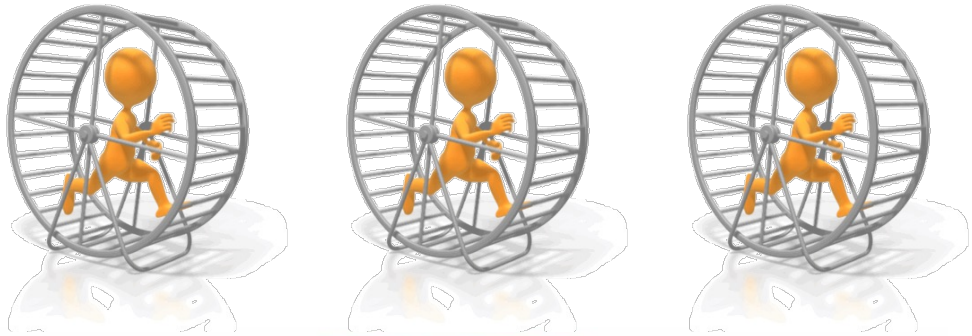
See dzone.com/articles/common-fork-join-pool-and-streams

Comparing Sequential vs. Parallel Streams

- Each worker thread in a fork-join pool runs a loop that scans for (sub-)tasks to execute

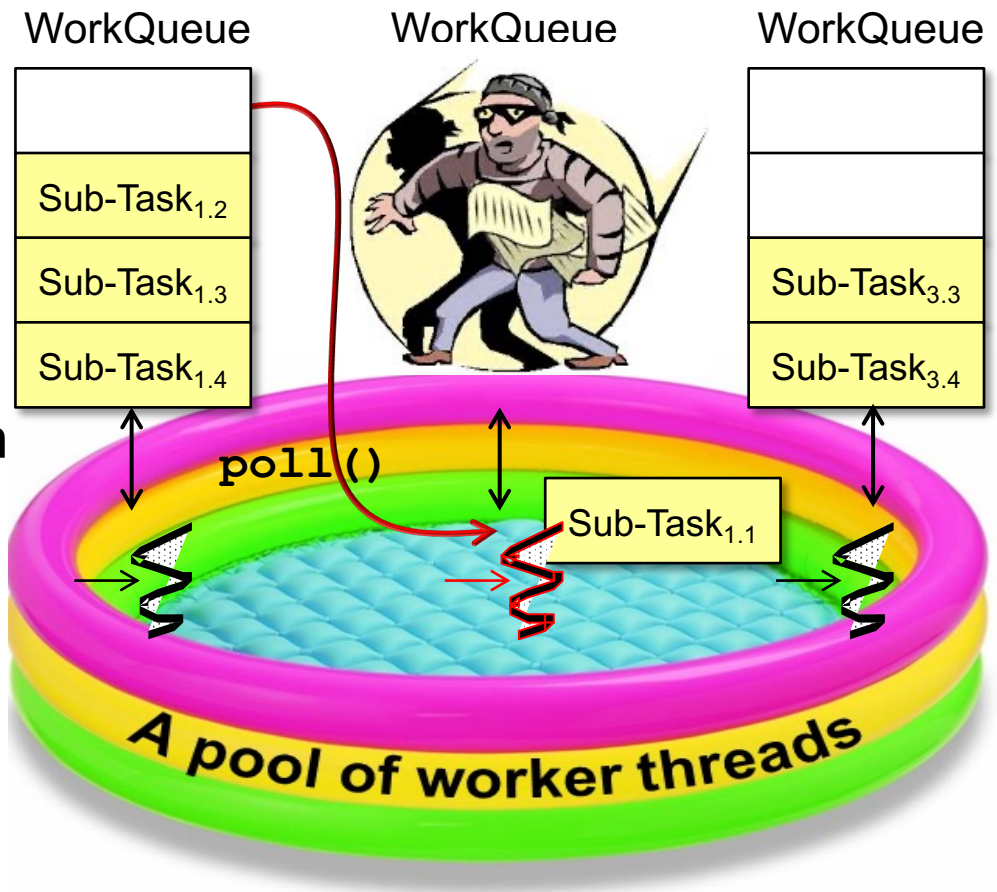
Comparing Sequential vs. Parallel Streams

- Each worker thread in a fork-join pool runs a loop that scans for (sub-)tasks to execute
  - The goal is to keep the worker threads as busy as possible!



Comparing Sequential vs. Parallel Streams

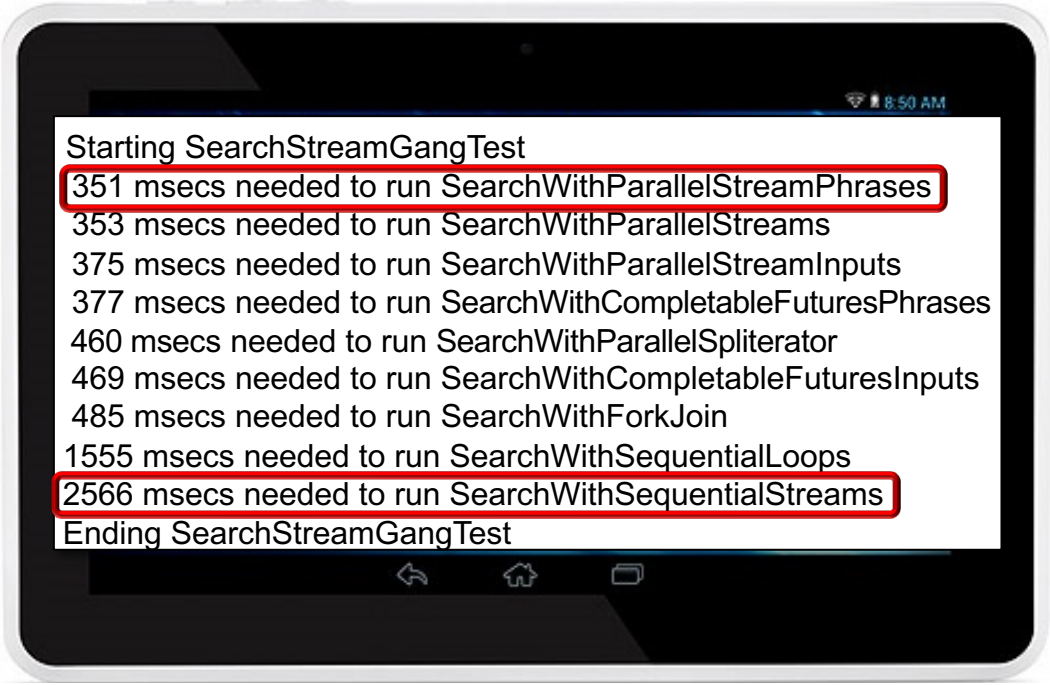
- Each worker thread in a fork-join pool runs a loop that scans for (sub-)tasks to execute
  - The goal is to keep the worker threads as busy as possible!
  - To maximize core utilization, idle worker threads "steal" work from the tail of busy threads' deques

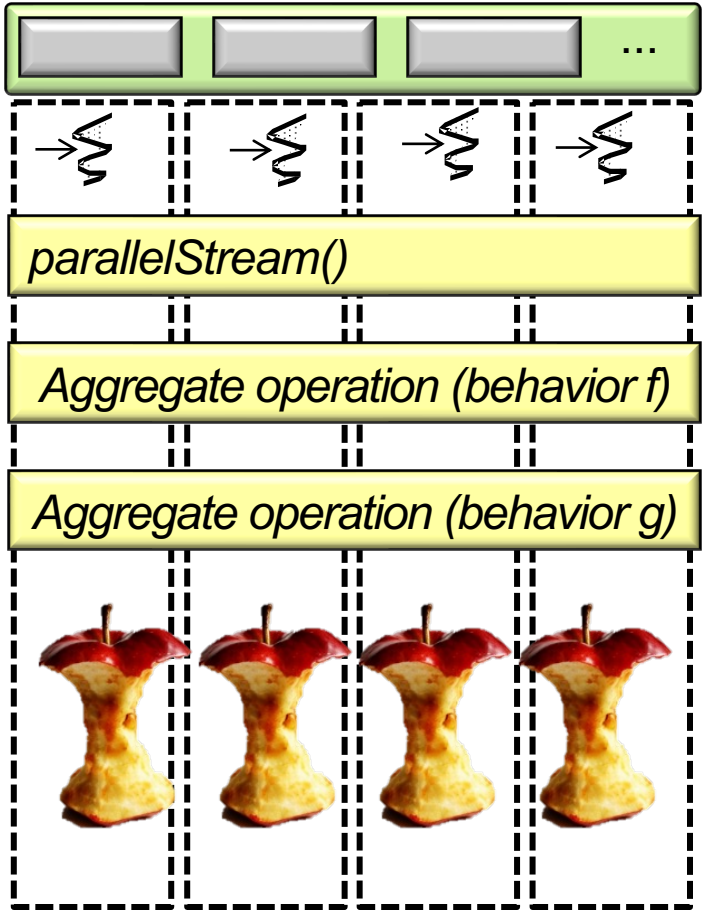


See www.dre.vanderbilt.edu/~schmidt/PDF/work-stealing-dequeue.pdf

Comparing Sequential vs. Parallel Streams

- A parallel stream can often be much more efficient & scalable than a sequential stream

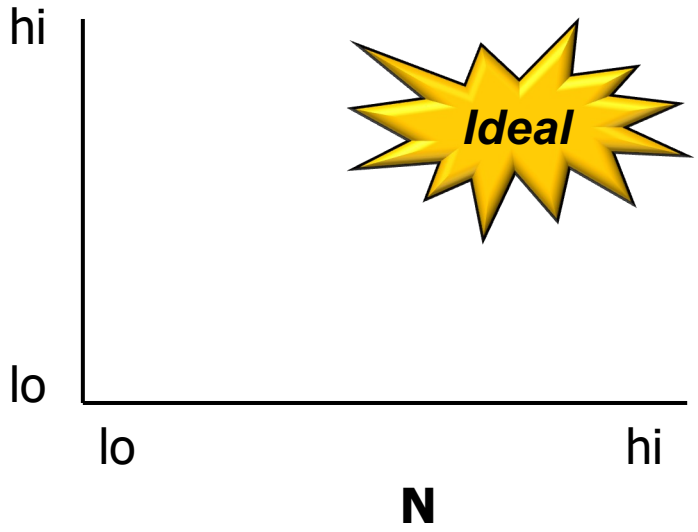




Tests conducted on a 10-core MacBook Pro with 64 Gbytes of RAM

Comparing Sequential vs. Parallel Streams

- A parallel stream can often be much more efficient & scalable than a sequential stream
  - However, certain conditions must apply for a parallel stream to be a "win"!



The "NQ" model:

- *N is the # of data elements to process per thread*
- *Q quantifies how CPU-intensive the processing is*

See on-sw-integration.epischel.de/2016/08/05/parallel-stream-processing-with-java-8-stream-api

End of Comparing Java Sequential Streams with Java Parallel Streams