Implications of Parallel Implementations of List Filters

York College Of Pennsylvania

CS 365

Daniel Mashuda

**Introduction:**

Performing operations on a list (mapping) and getting a subset of a list (filtering) are something that developers do quiet often when building software. Till Java 8, there was no native, language feature, way for simple maps and filters to be written without iterating over a list. These features are mainstream in other languages, and offer a concise way of retrieving a subset or mutating a collection. In Java, a loop would be necessary for Collection operations.

In Java 8, Java gained the Streams Api, which allows for filtering and mapping of collections. Along with Streams, Java 8 brought Parallel Streams, which would execute whatever a stream was doing, either mapping or filtering, would execute in parallel to attempt to reduce the runtime of the filtering or mapping. Parallel stream incurs the runtime penalty of starting threads and shutting down threads every time Collection.parallelStream is called.

The objective is to implement filtering of a collection in Java in parallel without incurring the cost of creating and destroying threads for each filter operation. In order to circumvent this, the object to do parallel filtering will have to have an explicit shutdown function, which will be called when no more filtering is needed. As a secondary objective, the created parallel filter will be able to accept lambda expressions in the same manner that the Streams API accepts lambda expressions.

**Implementation:**

**Results:**

**Conclusions:**

**Appendix:**