Jacob Spigle

COP4520

Professor Dechev

Tervel Queue Assignment

**Correctness**

Because of the utilization of single word compare-and-swap functions utilized within the set\_head() and set\_tail() functions in Tervel we can say that the queue will remain correct. The linearization points of point the enqueue() and dequeue() functions are met when we successfully set the tail or head, respectively.

**Experimental Evaluation**

My test data from running the lock-free version of the queue is available in the test\_data/lf folder in my submission.  
  
I found that in my implementation the enqueue and dequeue threads were able to perform a relatively equal amount of operations.

**Post Assignment Survey**

The following survey will help the CSE-S3 lab develop future software. Please be as honest as possible.

1. How often did you visit ucf-cs.github.io/Tervel/ for references on how to use the Tervel framework?

**The Tervel github was my primary source of figuring out how Tervel worked and how to properly use its’ assets, so I visited the site throughout my development.**

1. How often did you reference the /docs directory within the Tervel repository which contains most of the same information on Tervel’s website?

**Reading through the Tervel documentation on the github website was prefereable to reading the files in the repository.**

1. If you visited Tervel’s website at all, which page was most useful to you? Getting Started, Concepts, User Manual, Tutorial, or some other page?

**The Concepts page was the one I found the most useful.**

1. What was most useful to you about Tervel’s documentation?

**The listing of parameters and other definitions that were given inside the documentation.**

1. What was the least useful?

**Resources are slim, and Tervel does a very good job in giving students a starting point, however, for those still struggling, some of the TODO statements in the actual code were cryptic.**

1. Any additional comments about Tervel or it’s documentation?

**Overall the Tervel documentation and skeleton were great tools to utilize for developing the lock-free and wait-free queue.**