Programming 3

Philip Lassen (vgh804), Nicolas Larsen (vgn209)

December 2018

1 Implementation

- sampleFromSetOfISBNs: This method was simply to implement as we bootstrapped the Collections method shuffle for randomly permuting the list of ISBNs passed into the method. Then we simply chose a sub-list of this newly ordered list from the index 0 to the arguement num.
- nextSetOfStockBooks: In this method we use the Random class to generate random values for arguments that need to be passed to the constructor of StockBook. The author and title we chose to leave the same for all books. This was simply because they had no bearing on any of the other methods as books are identified by their ISBNs. The newly generated stockbook is then added to a set of stockBooks and this proccess is repeated until there are the number of stockBooks passed as an argument.
- runRareStockManagerInteraction: We call the nextSetOfStockBooks method above with the number of books specified in the configuration file. We then create a set of their ISBNs and if the ISBNs are not in the already existing books we add them to a new Set. Then we add this set to StockManager.
- runFrequentBookStoreInteraction: We get the EditorPicks. Then we get their ISBNs in a set. We then call sampleFromSetOfISBNs and use the configuration to decide how many we choose. Then we add them to buy Books.
- runFrequentStockManagerInteraction: We make a sort the books that we get from calling getBooks on StockManager. We do this by sorting them by the number of copies they have in stock. Then we get the number of books that we want to refill as specified by the configuration file and these to the set bookCopies. Then we call addCopies on the storeManger.

2 Discussion

2.1 Experiment setup

2.1.1 Data generation

We decided to initialize the bookstore with 100 books, as it seemed like a realistic number for a bookstore, as well as having more variability within the random choice of books. In the `BookSetGenerator`, we also made sure that the number of copies for each book was at least 100. We hoped that these two factors would alleviate most cases of salemisses.

2.1.2 Hardware

We ran the tests on a computer with a 4-core chip and 16 gigabytes of ram.

2.1.3 Measurement procedures

We decided to stick to the default 100 warmup runs and 500 number of actual runs, since it seemed like a decent test, and it was already taking a long time on our hardware. For calculating the latency, we decided to take the average latency for each run.

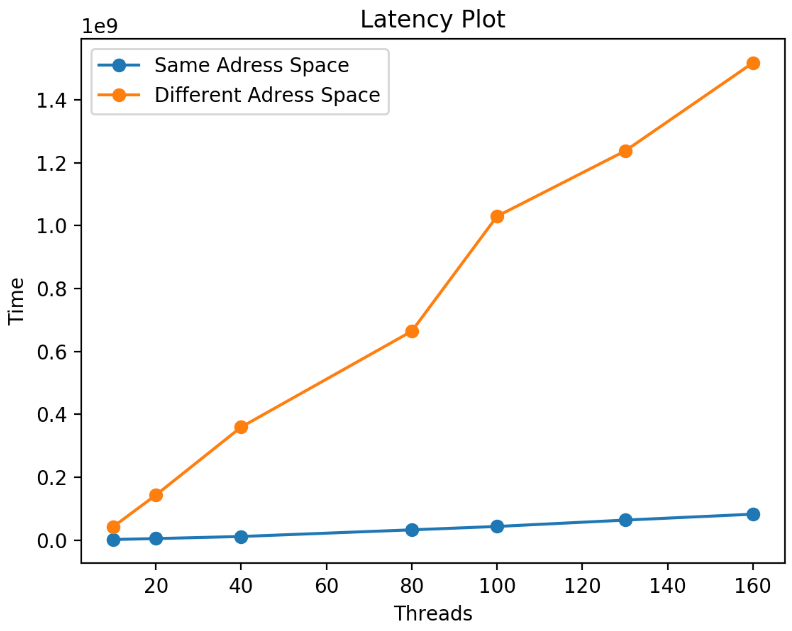

Figure 1: Latency Plot

2.2 Workload reliability

Increasing number of threads seem to strongly correlate with increasing average latency and decreasing throughput. This is expected, and in that sense it is reliable. It is also clear, when looking at the graphs, that it follows the number of threads closely. However, we did not make tests for different numbers of runs, even though that might have been an interesting metric to compare.

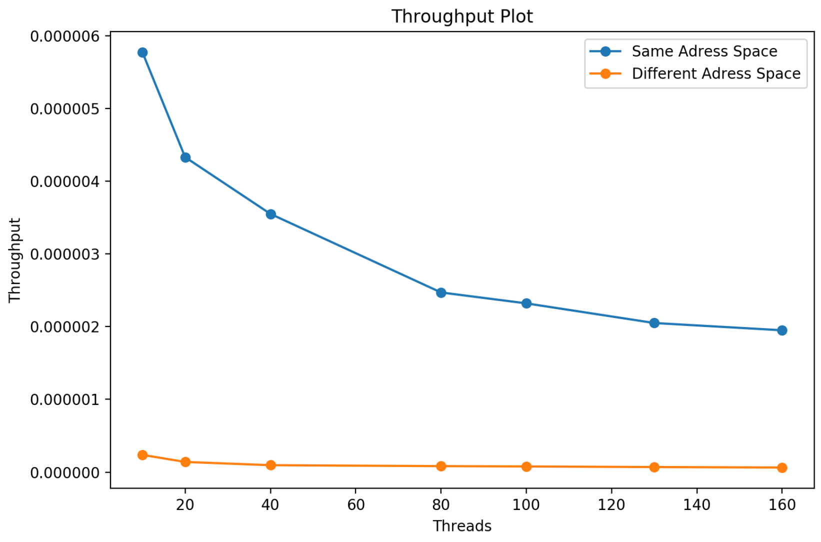

Figure 2: Throughput Plot