Joonho Kim
902 905 284
CS 4392

Project 2 Report

Imperative

Concurrency in Java is supported in that it gives you some proper tools/abstractions to implement concurrency, but it is still a little difficult to implement. Unlike Golang, I had to write the threads myself, which does give me some customization with my threads, but I still had to handle things like race conditions. Aside from creating the actual thread and executable task, handling the concurrency wasn't that bad since I could use the synchronized block to handle mutex access to a data structure.

Functional

Concurrency in Haskell was very hard to do because there was very few references online and I was very unfamiliar with the language. Parallelism was very easy to accomplish because I only had to use the `par` function in order to run things in parallel. This was an interesting idea because I had not thought concurrency in functional programming before; it's essentially running multiple instances of recursive code. That being said, I didn't have much idea of whether I was really accomplishing parallelism or concurrency correctly which I would say is the hard aspect of Haskell.

Modern Concurrency

Concurrency in Golang was really simple because it abstracted concurrency in a way that I didn't have to handle things like threads. The go-routine was basically a separate thread of execution that handled itself so it was very easy to implement whereas in Java I had to manually handle threads. The channels in Golang were also very helpful in funnelling multiple return values of parallel functions into a blocking queue; adding these values to a list essentially resembled a controlled mutex access to the list.
I didn't find many hard things in regards to concurrency. Everything was very simplified. Even though it was easier to concurrently program, it didn't run as fast as my Java implementation. I think one problem might have been the unlimited number of goroutines I could run which could've taken up a lot of memory.

Overall, Golang was the easiest to perform concurrency with because everything was build into the language. Java however ran the quickest. This could be because I'm more familiar with the Java language so that I could optimize my decisions better.