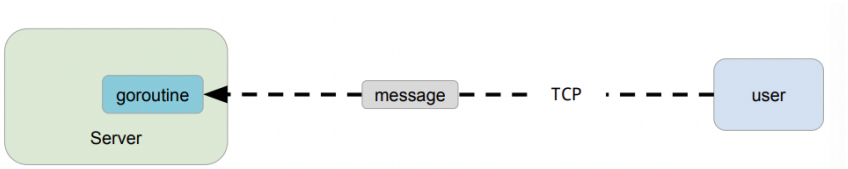
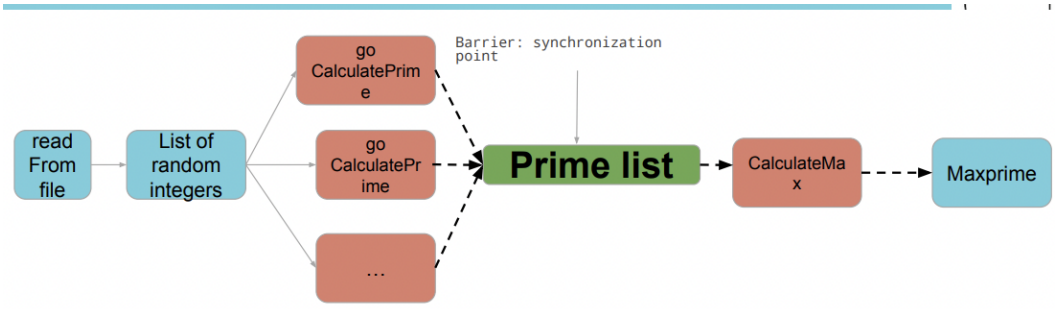
CAS CS 350
Lab 5

1. SimpleChat: Communication across processes
    a. TCP (SimpleChatTCP)
        i. Starting a TCP server that serves clients on certain port
    b. RPC (SimpleChatRPC)
        i. Remote Procedure Call over TCP
        ii. Using go's built-in RPC package
    c. Others working on many layers in the network stack
        i. HTTP, message queues
2. Stage 1 Synopsis (SimpleChatTCP)
    a. 

    b. Messages are sent to the server using just TCP requests. The server spawns goroutines for each request that handle the messages by printing them out
    c. The server uses threads to listen for RPC requests from the client, and when one is found, the message carried with it is handled by the server and printed out.
3. Stage 2 Synopsis
    a. 

    b. The "handleMessage" function requires a mutex to limit the action of displaying goroutine messages to users one at a time, rather than simultaneously
4. Maxprime
    a. Using A Channel
    b. Channels are made of buffer and lock
    c. Channels are themselves thread safe:
        i. read/write from many goroutines will not cause data race

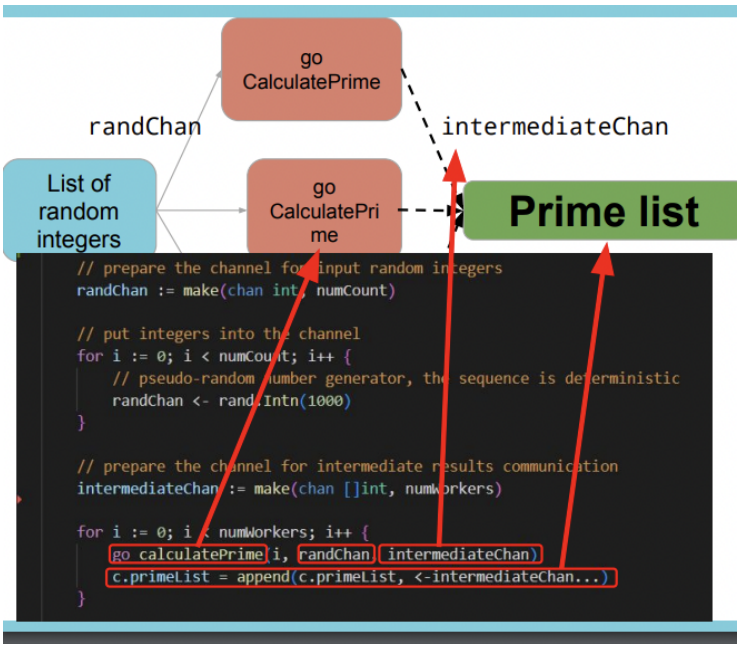
d. Overview



e.

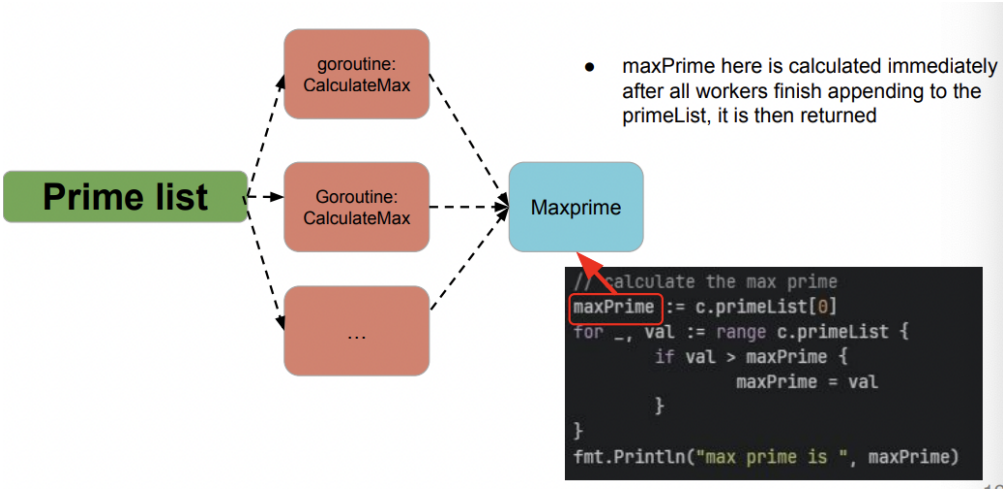
i. Prepare a list of random integers
ii. Spawn several goroutines to take prime numbers
iii. Put those primes into an intermediate slice
iv. Spawn several goroutines to calculate max of slice parts
v. Aggregate

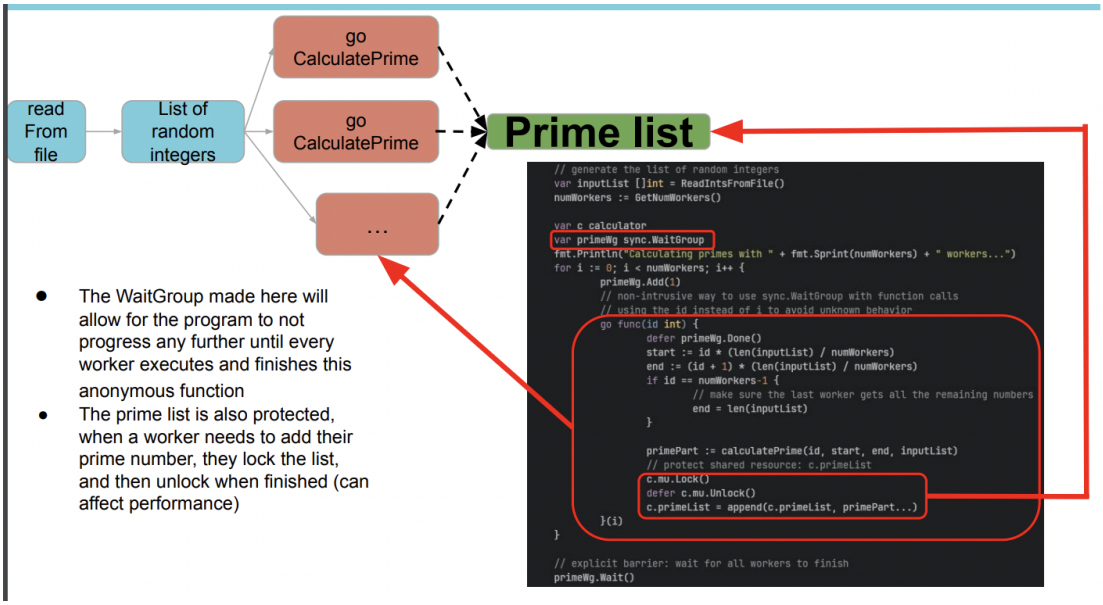
5. Phase 1: main

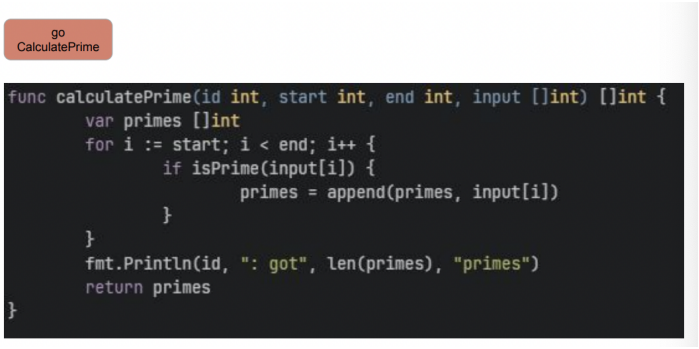


a.

b. Appending to c.primeList is safe because appends only run in 1 goroutine (main)
c. Reading from intermediateChan acts as the implicit barrier: loop only finishes after all workers write to the chan - input exhausted
d. Select: read from inputChan until it's empty

6. Phase 2: calculate max
    a. 

7. Using WaitGroup (and mutexes)
    a. A waitgroup waits for a collection of goroutines to finish. The main goroutine calls Add to set the number of goroutines to wait for. Then, each of the goroutines runs and calls Done when finished. At the same time, Wait can be used to block until all goroutines have finished
    b. 

    c. 

d.

```
fmt.Println("Total primes:", len(c.primeList))

// calculate the max of the primes
var maxPrime = c.primeList[0]
for _, prime := range c.primeList {
        if prime > maxPrime {
                maxPrime = prime
        }
}
fmt.Println("Max of all input is", maxPrime)
```

- Once the intermediate list is made, the max of the primes is then searched for and returned

8.