Parellel

Christian Winwood
A02074021

February 9, 2021

1 Implementation

My implementation was fairly simple. I had each chef check to see if the cook has sent them a quit message. If they received one then they break out of their while loop and end the program. If they did not receive a quit message then they generate a random time to wait and then send a message to the cook and repeat. The Cook is fairly similar. The first thing he does is check to see if there is a message, if there is he receives the messages and increments the number of messages. When there are no longer any messages left he checks to see how many messages he has, if it over 20 he sends a message to all the chefs and then breaks out of his loop. If it is 0, he waits for one second and then loops again, and if it isn't either of those, he loops through all the orders and sleeps for 1 second for each and prints out that he completed one at the end of each order. That is the implementation.

2 Compile and Run

The compile and run commands are the same they have always been. First you use

mpic++ chefs.cpp

in order to compile the program. This creates an a.out files that you need to run. To run it you use the command

mpirun –oversubscribe -np ¡number-of-chefs-plus-one¿ a.out

and the program will run.

3 Output

Whenever the cook doesn't have an order, he prints that he doesn't have an order and can relax. When he is done checking for orders, he prints out how

many orders he received and then prints out when he finishes each order. Once all orders have been done, the cook states that and then say they are checking for more orders. If he receives more than 20 orders he says he quits. Finally when the chefs receive the message that the cook left they also say they quit. The chefs will also print out whenever