15-110 Hw5 Checkin - Written Portion

For this assignment, use the powerpoint slides provided to make the images demonstrating your understanding of concurrency, pipelining, multitasking, and multiprocessing. Print the powerpoint to pdf and submit it as your hw5checkin written assignment (it should be 5 slides, do not submit this assignment page).

#1 - Concurrency - 10pts

Exponentiation (raising a base to a power like 2^6) can be computed concurrently by first multiplying pairs of bases (e.g., 2*2) together, then multiplying those products together (4*4), and continuing until there is only one answer. Create a concurrency tree for 2^7 that computes the answer in the shortest amount of time (fewest levels). Be sure to fill in the values that are being multiplied.

#2 - Pipelining - 10pts

A factory is looking to speed up its production. Currently, each worker makes one shirt at a time by first measuring the fabric (5 minutes), then setting up cutting supplies (5 minutes), cutting out the pattern (5 minutes), then setting up sewing supplies (5 minutes), sewing it together (5 minutes), and finally removing supplies (5 minutes) and folding it up (5 minutes). Change the background color of the table to show how you would break up this task among workers into a pipeline so that they would each do a subtask, so they do not need to setup and cleanup each supply when switching tasks, which speeds production. Then answer the questions at the bottom of the slide.

#3 - Multitasking - 10pts

You have 4 programs to run on only a single core. Draw a picture of what multitasking may look like in this scenario. If segments of a program will run more than once, be sure to show that in your picture.

#4 - Multiprocessing - 10pts

You have 4 programs to run on 2 cores. Draw a picture of what multiprocessing may look like in this scenario. If segments of a program will run more than once, be sure to show that in your picture.

Programming Problems

Start with the zip file available on the course website. This contains test files and the file you will edit: hw5_checkin.py. You should edit and submit **only** hw5_checkin.py to the Gradescope assignment Hw5 Checkin (Programming) to be autograded.

In this assignment you will write a series of mappers and reducers to solve particular problems. All functions may be checked by running the starter file, which calls the function testAll() to run test cases. Additionally, code has been provided that implements MapReduce and calls your mappers and reducers on large files. You may run test_mapreduce.py to see your functions used on real multiprocessing problems.

Part A - Counting Words

Mapper - file_to_count(filename) - 10pts
Write a function file_to_count(filename) which opens the file of the given filename, and uses a loop to iterate through it, counting all the words in the file. The file contains one word per line. The function should return the count of the words in the file.

Reducer - combine_counts(count_list) - 10pts
Write a function combine_count(count_list) that takes a list of counts returned by the mappers, and returns the sum of those counts.

MapReduce will return a total count of the words in all files.

Part B - Searching for Words

Mapper - file_to_search(filename,word) - 10pts
Write a function file_to_search(filename,word) which opens the file of the given filename and uses a loop to iterate through it, searching for the exact word. Note that the words in the file are in alphabetical order and you may choose to use that information to speed up your search. The function should return a boolean of whether the word is in the file.

Reducer - combine_search(search_results) - 10pts
Write a function combine_search(search_results) which takes a list of booleans and returns the first index of True or -1 if none are True.

MapReduce will return the filename that contains the given word.

Part C - Substring Search

Mapper - file_to_subsearch(filename, substring) - 10pts
Write a function file_to_subsearch(filename,substring) should read a file and check if each word in the file contains the given substring. The function should return a list of all words containing the given substring. You can ensure that your words do not contain the newline character \n at the end by running strip().

Reducer - combine_lists(search_result_lists) - 10pts
Write a function combine_lists(search_result_lists) that takes a list of lists of words and returns a new list of all words in all lists. For example,
combine_lists([["cat","cats"],["catastrophe","certificate"]]) =
["cat","cats","catastrophe","certificate"]

MapReduce will return all words containing the given substring.