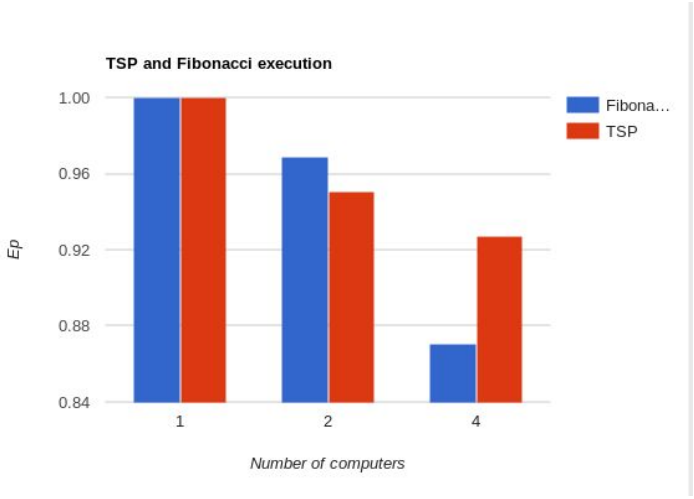
CS190B

HW3 REPORT

1. Performance



2. Key Design Issues

2.1 - Api:

We had to modify the API in some manners. The most important one was the creation of a class called “ComposedTask” that extends “Task”. The class “ComposedTask” is a task whose inputs are the single output from each of the objects of type “Task” and whose output is some function of its inputs.

2.2 - Client:

The clients still almost the same as in homework 1 and 2. The difference from homework 2 is that the client only creates one task whereas in homework 2 the client creates various tasks and then merge all the results.

2.3 - Space:

The space was severely modified, we had to create: A BlockingQueue for the ready tasks; A ConcurrentHashMap that maps Strings (IDs) to BlockingQueue of waiting tasks. Besides the data structures, we had to create some logic to add the tasks to move the tasks to the right data structure.

2.4 - Computer:

We had to make each computer able to perform some important steps for each task: Call the task; Get the task's children; Send the task's children to the Space; Send some information to Space where this information is going to be used by the task's parent successor.

3. How Our Architecture Solve The Key Design Issues

Our architecture is the following steps:

1. The client sent a task, with necessary information, such as CITIES coordinates, and final task ID, to the space, and wait for space to send the final result, which is the result of the task with the final task ID.
2. Once the space receive a task or a list or a task, it will check how many input variables it still need. If the number is not zero, put it into a hashmap "waiting", otherwise put it into ready queue.
3. Computer will try to get a task from the space to execute if it does not have a task. If the task is a decompose task, according to its ID, when the task is executed, the task may do 1. Do the calculation and return the result, or 2. Do nothing and return a null result. If the task is a compose task, It will return the result that it has. The result includes following information: 1. The Task's ID 2. The result, ex. A minpath in TSP 3. The target task in the waiting hashmap that need this result.
4. After executing the task, the computer will ask space to save the children tasks of the executed task, and update its waiting hashmap according to the returned result.

5. If the space get the result and found that the id is final task ID, the return the result to the client, and the job is finished.