Adaptive Experiment output

**1.intruduction**

The way to implement the adaptive:

RRS(default):It could assigns tasks to workers in a round-robin fashion but when there is a great computational difference among tasks the round-robin scheduling policy could lead to load imbalance in worker's workload (especially with short stream length) and after testing. The length of the input queue could be infinite.

AS: that are farms where the workers “ask” for something to be computed rather

than accepting tasks sent by the emitter (explicit or implicit) according to some

scheduling policy.

The length of the input queue could be 1.

Another way to define our own scheduling method: This is a virtual function named schedule\_task() at ff\_loadbalancer class and can be redefined to implement a custom scheduling policy.

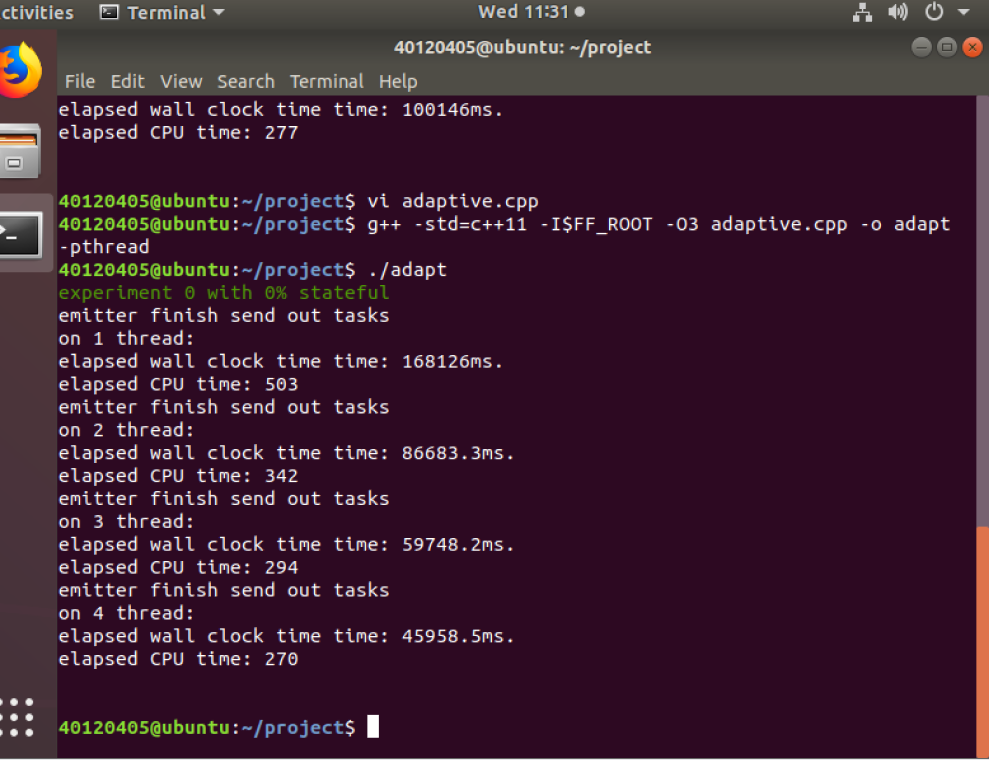
Throughout this Experiment report, I will discuss the overhead and benefit of Auto Scheduler (AS) over Round Robin Scheduler (RRS) in terms of the relative performance (improvement), S, for a given benchmark. The experimental **relative performance** is calculated as the ratio of average execution times using RRS, tR, and ARRS, tA, respectively,

**S = tR/tA**

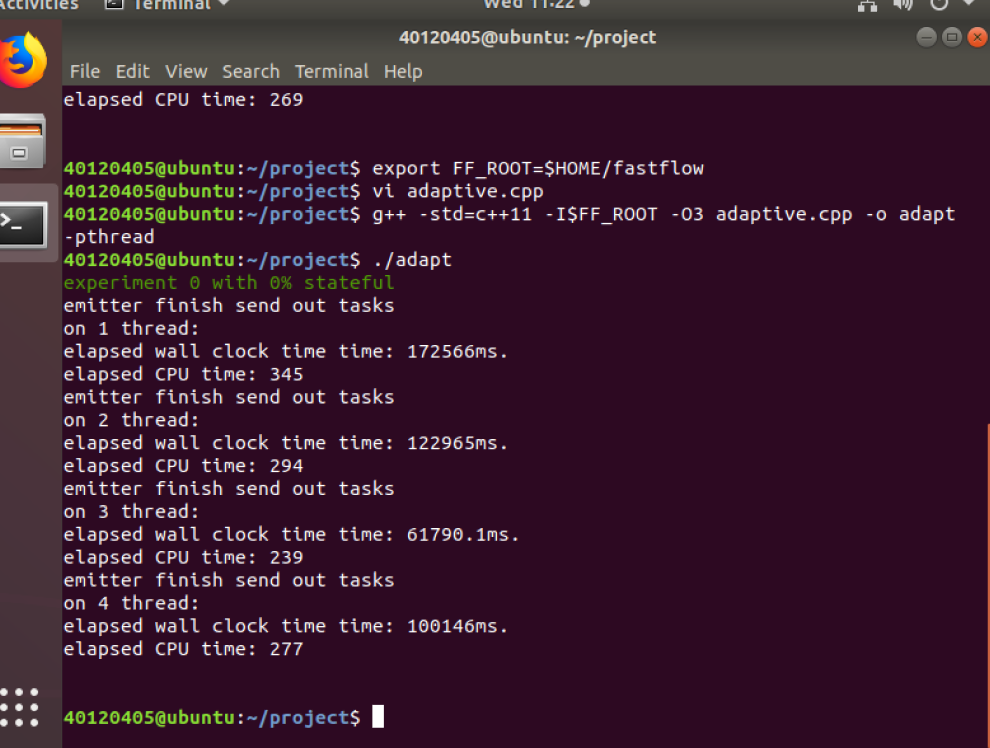
The purpose of this section is to measure the scheduling overhead by AS. To measure the overhead, a FastFlow application with uniform task is devised. The uniform task can be described as a task that requires, in principle, the same amount of time every time it is executed by a worker. The FastFlow application consist of a farm, i.e., emitter, collector and workers. The experiment was exercised with 1,2,3 and 4 workers separately. The emitter of the farm distributes tasks to the workers through a scheduler (RRS or AS). The uniform task consists in 75% short task, which takes around 1080 ms time to execute on this machine, and 25% long task, which takes around 4100 ms time to execute on this machine. This time I put 100 tasks for the test.

**2.output**

2.1 Auto Scheduler



2.2 Round Robin Scheduler



The processing time with 100 tasks:

|  |  |  |
| --- | --- | --- |
| Workers | Adaptive (AS) | Non-Adaptive (RRS) |
| 1 | 168.13s | 172.57s |
| 2 | 86.68s | 122.97s |
| 3 | 59.75s | 61.79s |
| 4 | 45.96s | 100.15s |

Relative performance with 100 tasks:

1 worker: s=tR/tA = 1.03

2 workers: s=tR/tA = 1.42

3 workers: s=tR/tA = 1.03

4 workers: s=tR/tA = 2.18