ASSIGNMENT 2
INTRO TO PARALLEL SCIENTIFIC COMPUTING

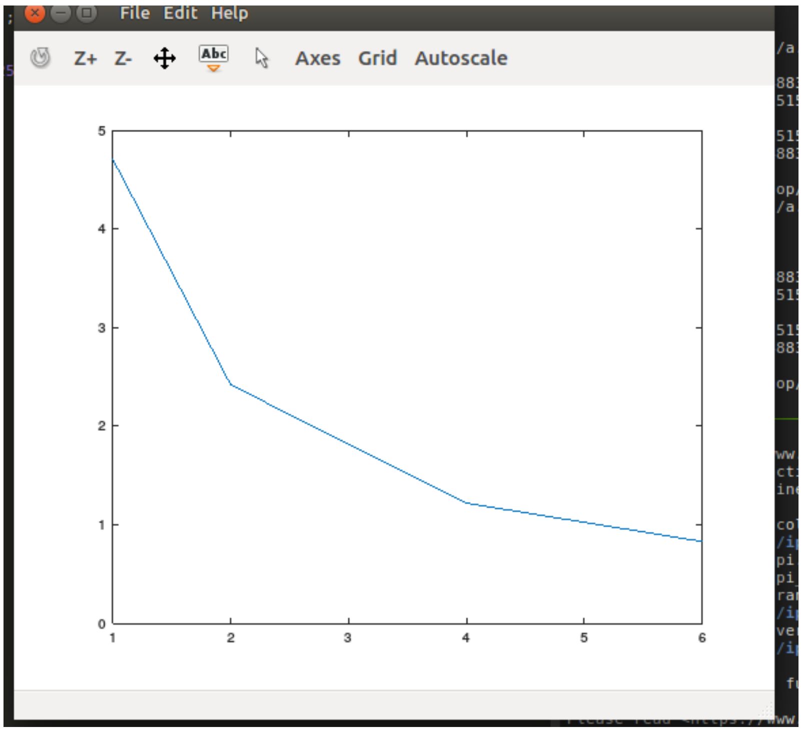
UDIT SINGH PARIHAR
2018701024

1. Scheduling:
    1. Fixed size iteration(Code : schedule.c):
        1. Sequential: Time (seconds): (0.371304), (0.374828), (0.373913)
        Average time(second): 0.3733483
        2. Static: Time (seconds): (0.489133), (0.489093), (0.489055)
        Average time(second): 0.48909366
        3. Dynamic: Time (seconds): (19.2617), (18.6293), (18.2359)
        Average time(second): 18.70896
        4. Guided: Time (seconds): (0.481399), (0.479862), (0.48592)
        Average time(second): 0.4823936
        5. Dynamic with chunk=100: Time (seconds): (0.582136), (0.56903), (0.575079)
        Average time(second): 0.575415

    2. Variable size iteration(Code : variable.c):
        1. Static: Time (seconds): (54), (54)
        Average time(seconds): 54
        2. Guided: Time (seconds): (36), (36)
        Average time(seconds): 36
        3. Dynamic: Time (seconds): (36), (36)
        Average time(seconds): 36

2. PI(Code : pi_mc.c, random.h, random_par.c):
    1. Cores 1: (4.801050), (4.665588), (4.666764) and PI = 3.141494
    Average time(seconds): 4.711134
    2. Cores 2: (2.446955), (2.420618), (2.409973) and PI = 3.141854
    Average time(seconds): 2.42584866
    3. Cores 4: (1.243695), (1.218494), (1.209359) and PI = 3.141760
    Average time(seconds): 1.22384933
    4. Cores 6: (0.839439), (0.842850), (0.830953) and PI = 3.141749
    Average time(seconds): 0.83774733

    a. Speed up 6core / 1core: 5.623
    b. Speed up 4core / 1core: 3.849
    c. Speed up in 4core is not excatly 4 times that of 1core because there is overhead involve in synchronization as a barrier time while summing the Ncirc variable.

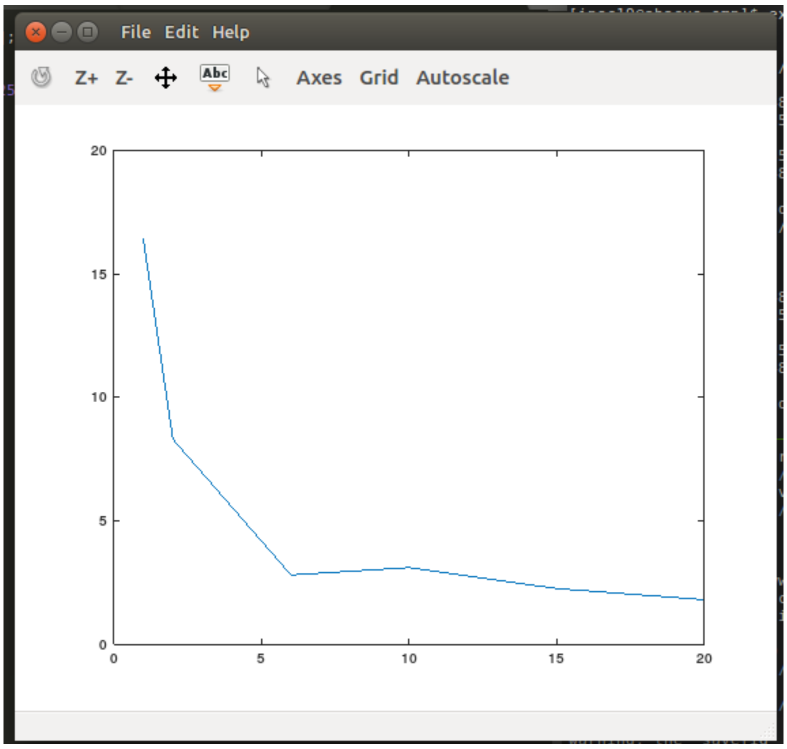


3. Jacobi(Code : solver_jacobi.cpp, heat.cpp):
    a. Different Cores: Size of matrix = 1500
        1. Cores 1: (16.338), (16.542)
        2. Cores 2: (8.382), (8.467)
        3. Cores 6: (2.898), (2.878)
        4. Cores 10: (3.197), (2.469)
        5. Core 15: (2.255), (2.254)
        6. Core 20: (1.827), (1.828)

    b. Scheduling: Size of matrix = 1500, Cores = 10
        1. Static:
            1. Chunk 100: (2.954), (3.133)
            2. Chunk 200: (2.638), (2.396)

        2. Dynamic:
            1. Chunk 100: (3.160), (3.118)
            2. Chunk 200: (2.557), (2.792)

        3. Guided:
            1. Chunk 100: (1.959), (2.001)
            2. Chunk 200: (1.961), (1.917)



4. Race Condition(bank.c):
1. Sequential: Time : (0.015126)
Result: Same every time and no race condition
2. Uncomment A1 + A2: Time (0.028742)
Result: It is different at each different run, due to race condition at updating balance shared variable
3. Recomment A1+A2 and Uncomment B1+B2: Time(0.014149)
Result: Now balance is a local variable for each thread, so its effect is not resulted in the compounded balance, so it is also a wrong result.
4. Recomment B1+B2 and Uncomment A1+A2, C1+C2: Due to atomic operation each thread is writing balance variable at a time, so there is no race condition, but there is overhead that each thread has to wait for its turn to update balance. That is the reason single core implementation is faster than 4 core implementation.
1. 1 Core: Time(0.054806)
Result : Correct result
2. 4 Core: Time(1.2508)
Result: Correct result
5. A1+A2 with Reduction: This is best implementation if we want to use parallel computing in this scinerio. As thread do not need to wait for each other to update balance. And resulting operation is done at the end of parallel region.
Time: (0.029863)
Result: Correct result