CONTEXT

Problem Description

1. We have to compute the inclusive prefix sum (scan) from an input array and convert the same code into executable scan.

2. We have been given an input array, we have to convert it to an output array, which contains elements from the input array greater than a filter element.

Complexity of Algorithm
For serial algo
Problem 1. O(n), n = number of elements in the input array.
Problem 2. O(n), n = number of elements in the input array.

Possible speedup(theoretical) :

Speedup S =1/ (P/n + s)
n – number of cores
P – percentage of code that can be parallelized
s – percentage of serial code(which is not parallelized)
For our code, P ~ 1 and s ~0
n=4, So theoretical speedup = 4

Hardware Details

Memory 3.7 GiB
Processor Intel® Core™ i7-4702MQ CPU @ 2.20GHz × 8
OS type 64-bit
Disk 103.5GB

Input And Output Information

1. Input is a long array and output is also a long array which has prefix sums of input.

2. Input and output are both long arrays.

Optimization Strategy

The efficient parallel method is divided into parts :

1. reduce phase
In the reduce phase we traverse the tree from leaves to root computing partial sums at internal nodes of the tree. This is also known as a parallel reduction, because after this phase, the root node (the last node in the array) holds the sum of all nodes in the array.

2. down-sweep phase
we traverse back up the tree from the root, using the partial sums to build the scan in place on the array using the partial sums computed by the reduce phase.

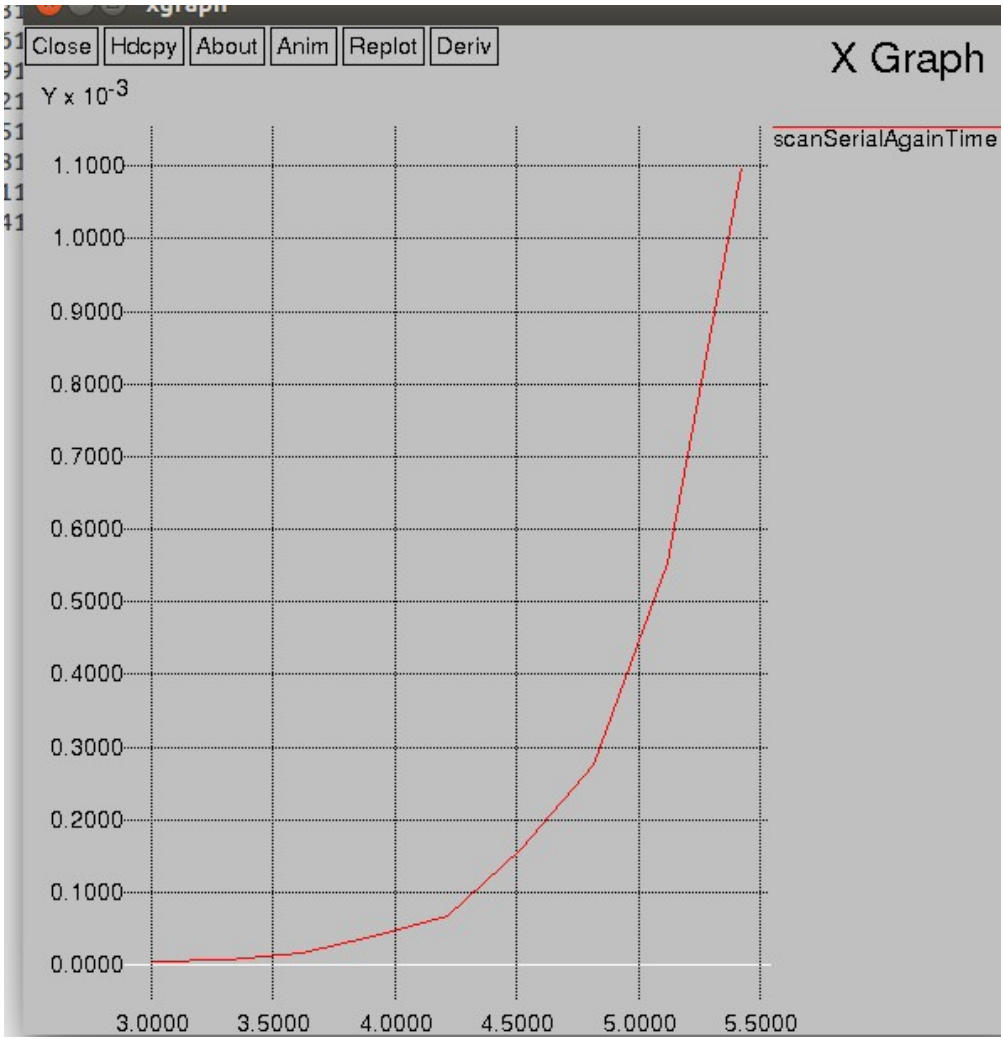
For the filter question, we created an auxilary array, and the index where input array element was lesser than filter, we inserted 0. Then we scanned the auxiliary array. We then created the output array, the indexes

where auxiliary array element was greater than the next element, we inserted the input element to the output.

Prefix Sum (Scan)

Problem size vs time for serial implementation

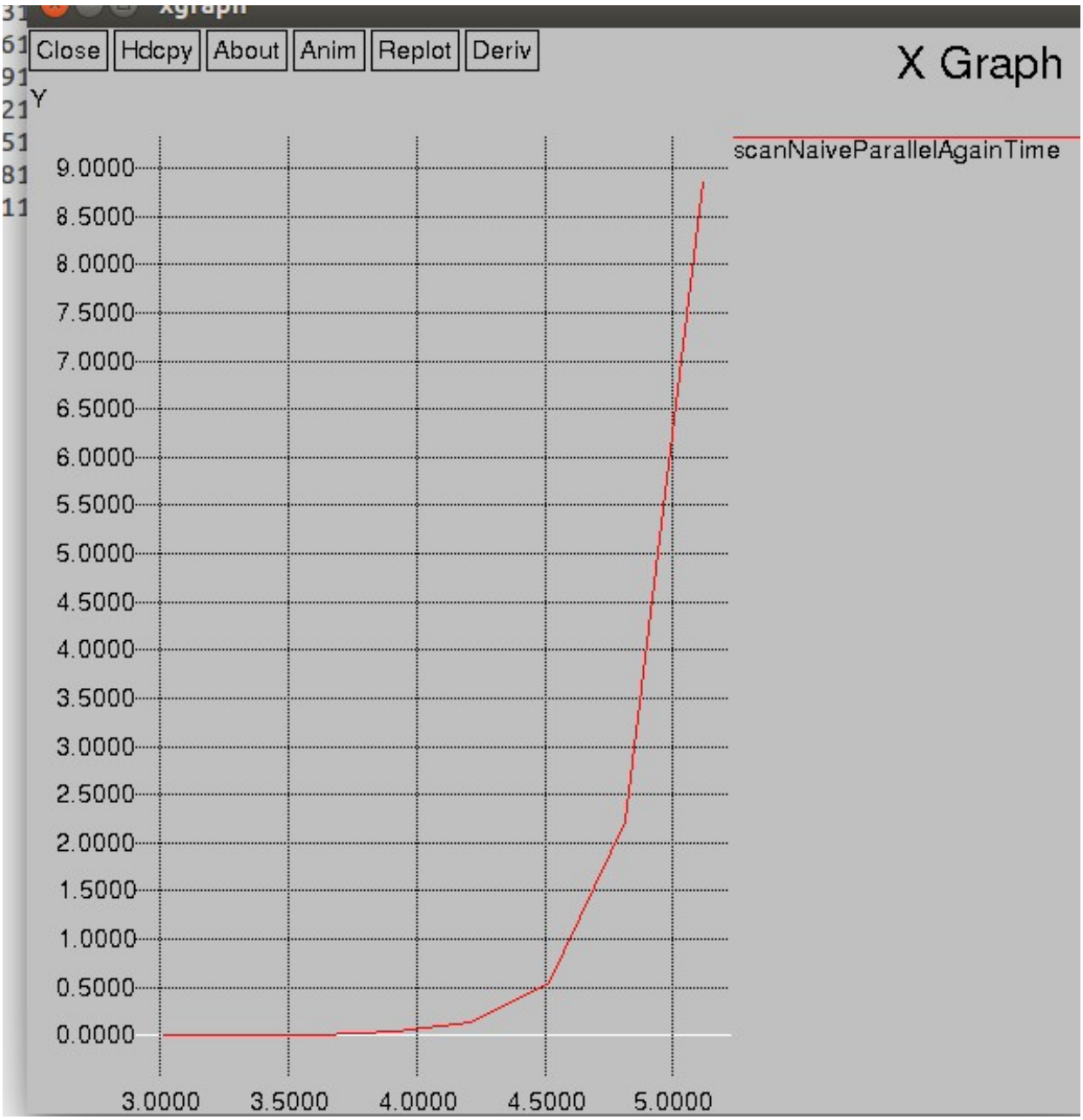
Y axis - Time(in seconds)



log(ProblemSize)

Problem size vs time for Native Parallel implementation

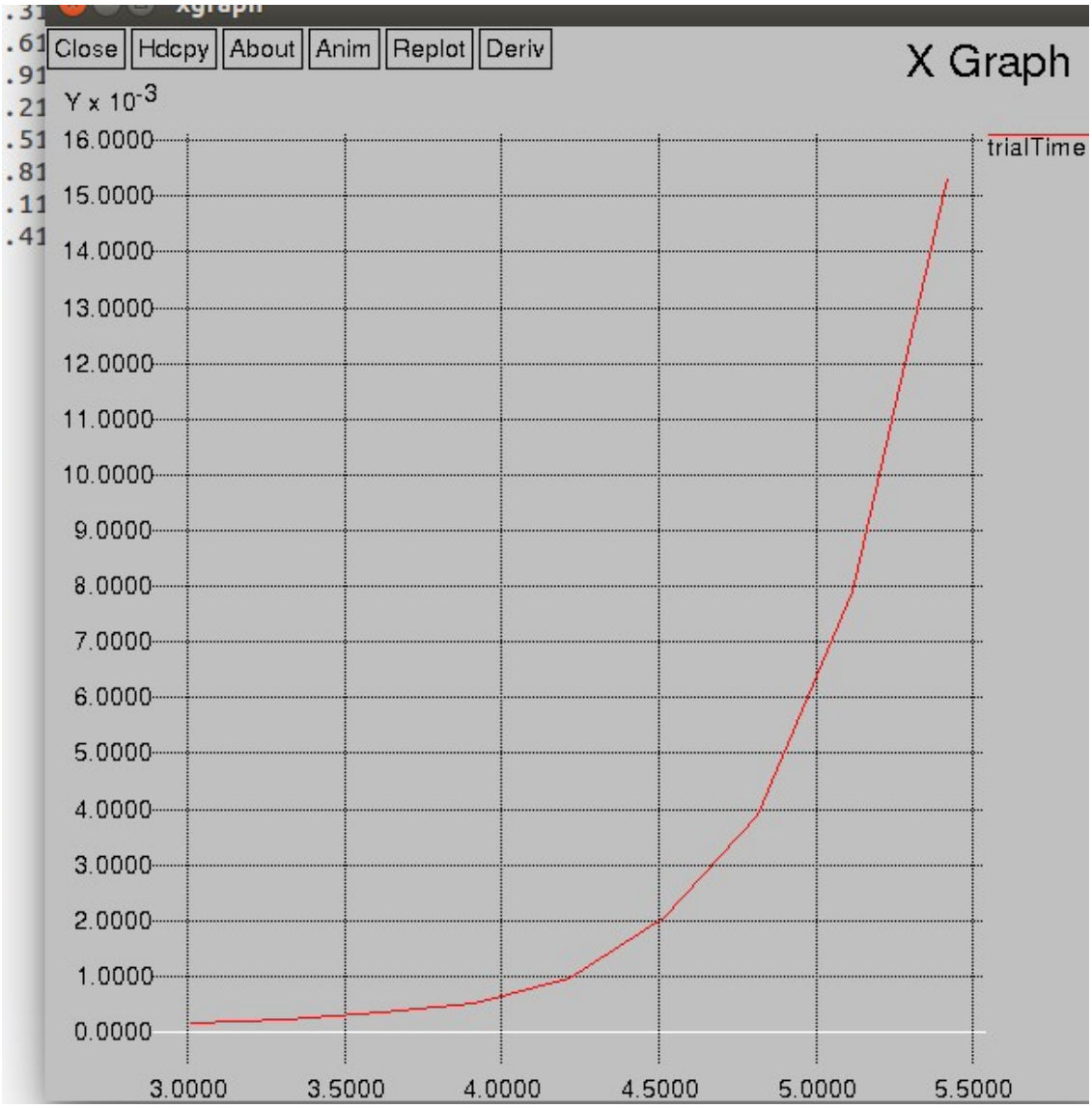
Y-axis = time



log(problemSize)

Problem size vs time for Native Parallel implementation

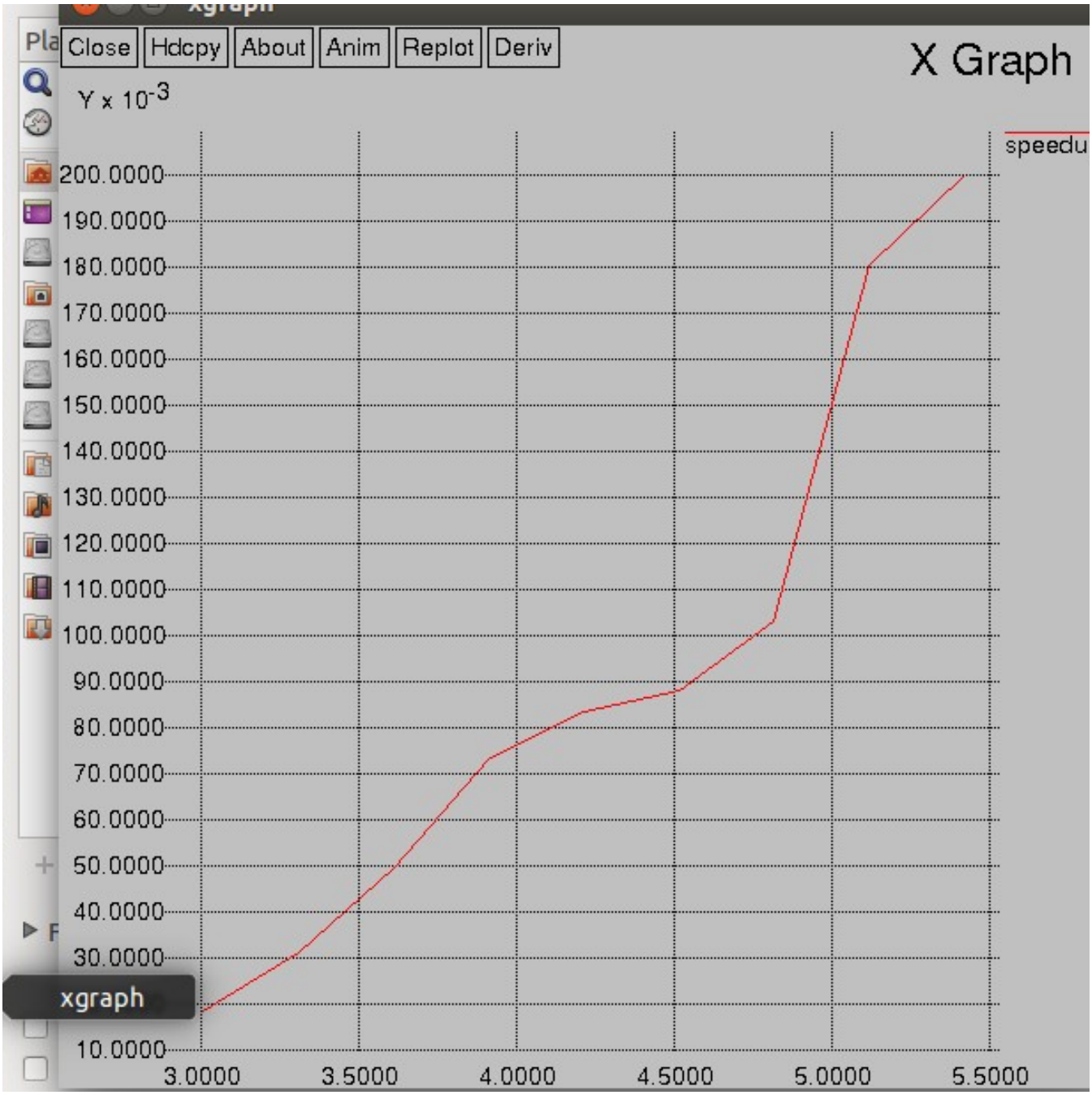
Y-axis = time



log(ProblemSize)

SpeedUP(for 8 threads)

Y-axis = speedUp

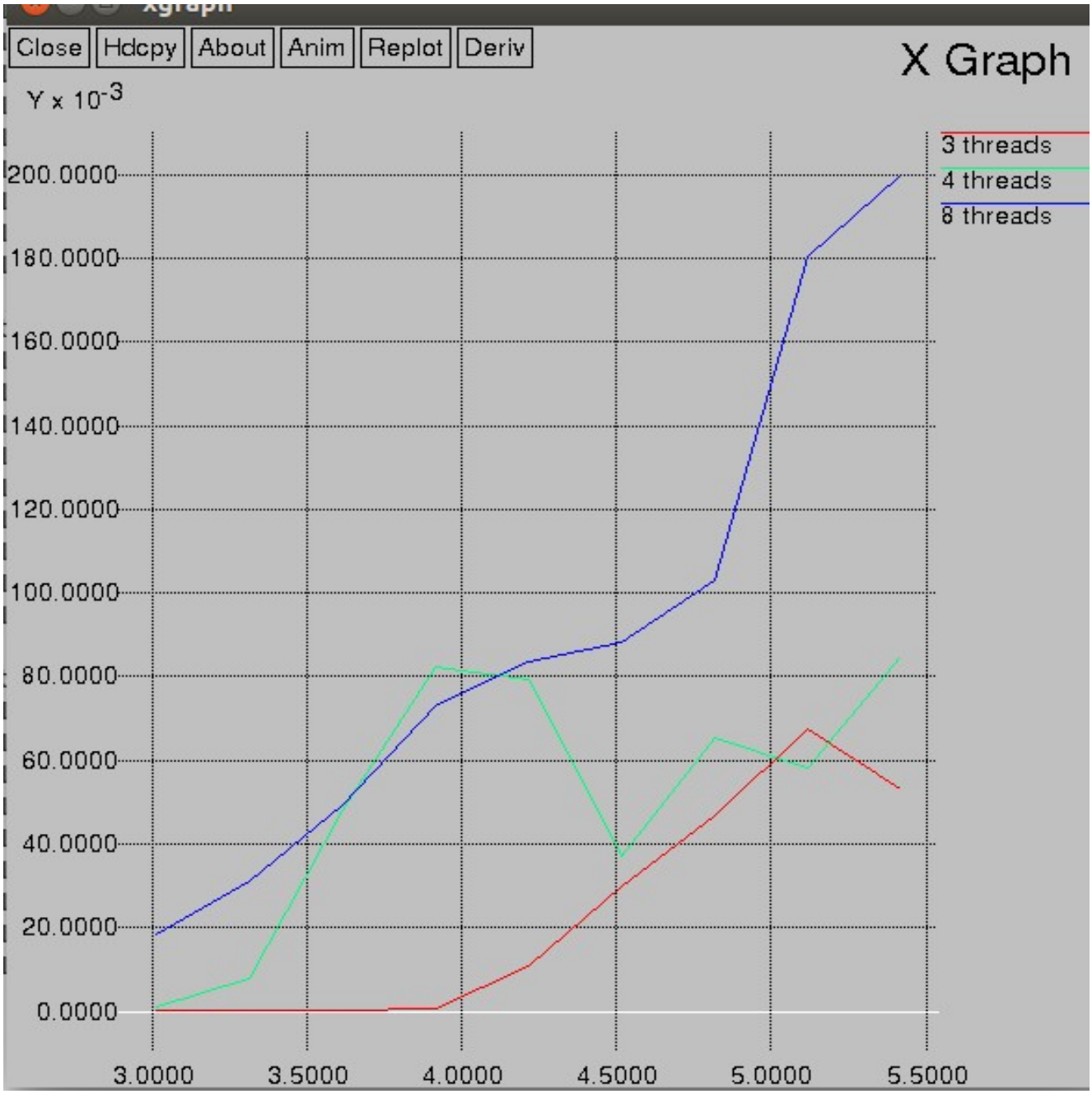


log(ProblemSize)

Number of cores

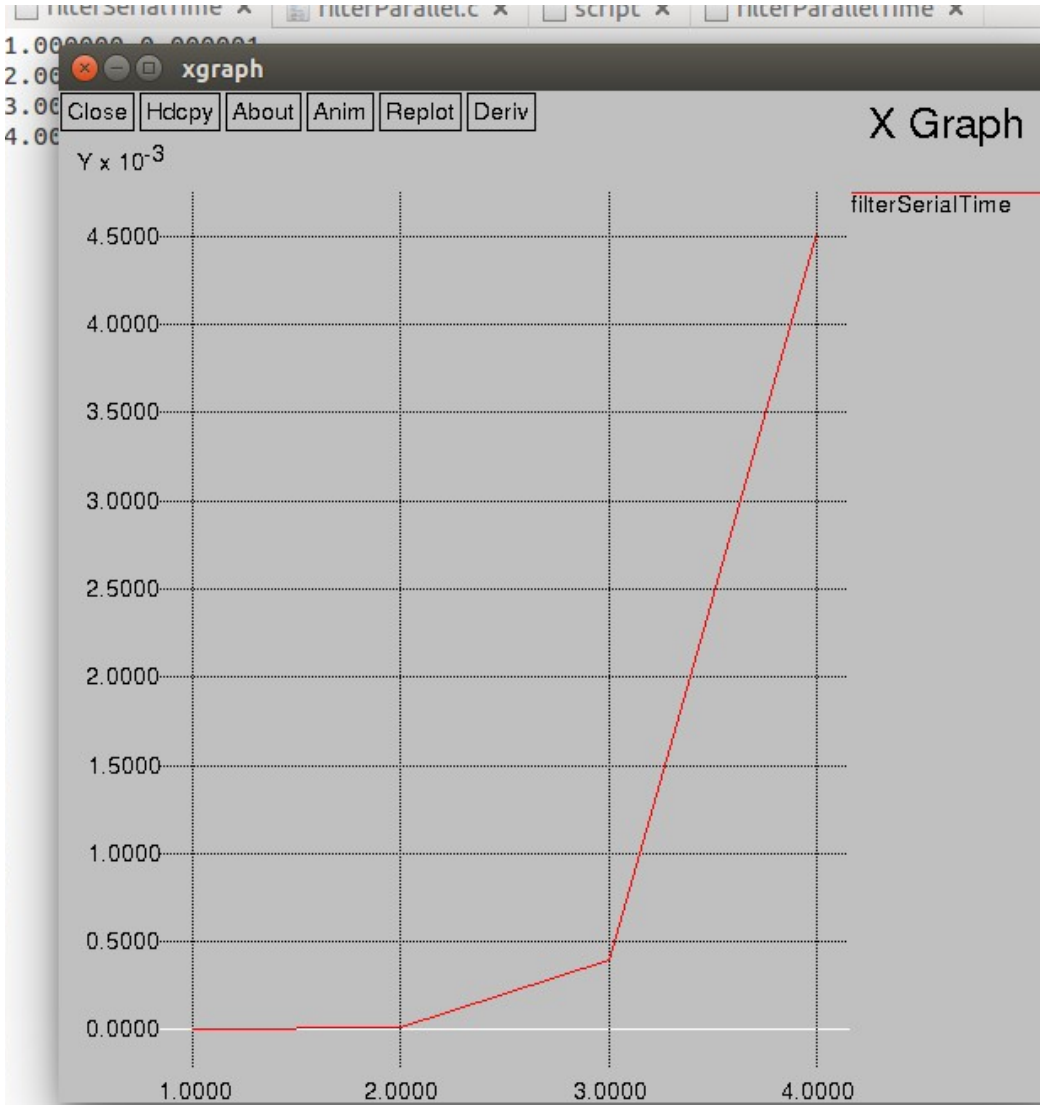
X axis – log(problemSize)

Y axis – SpeedUp

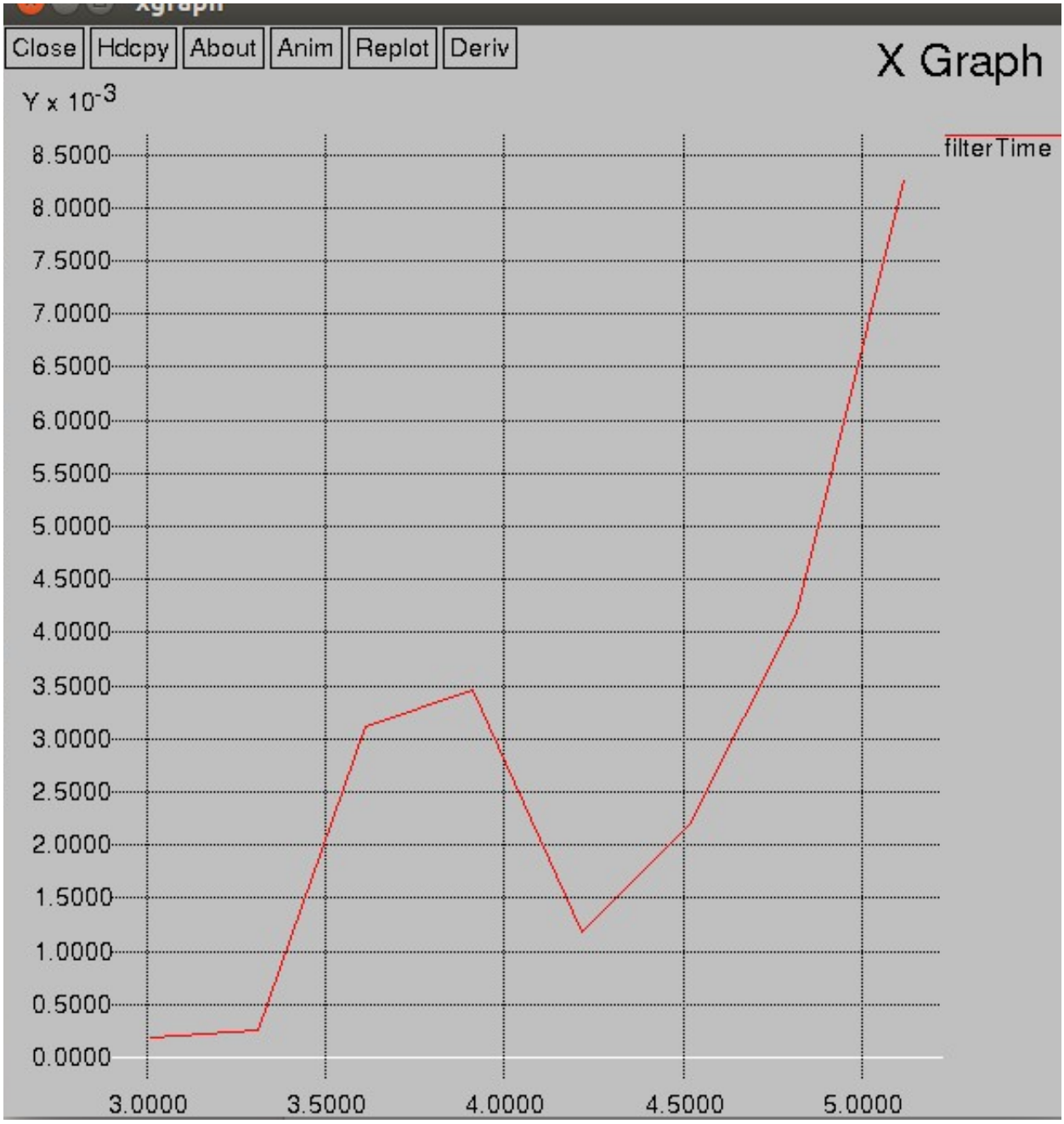


Filtering (without using scan)

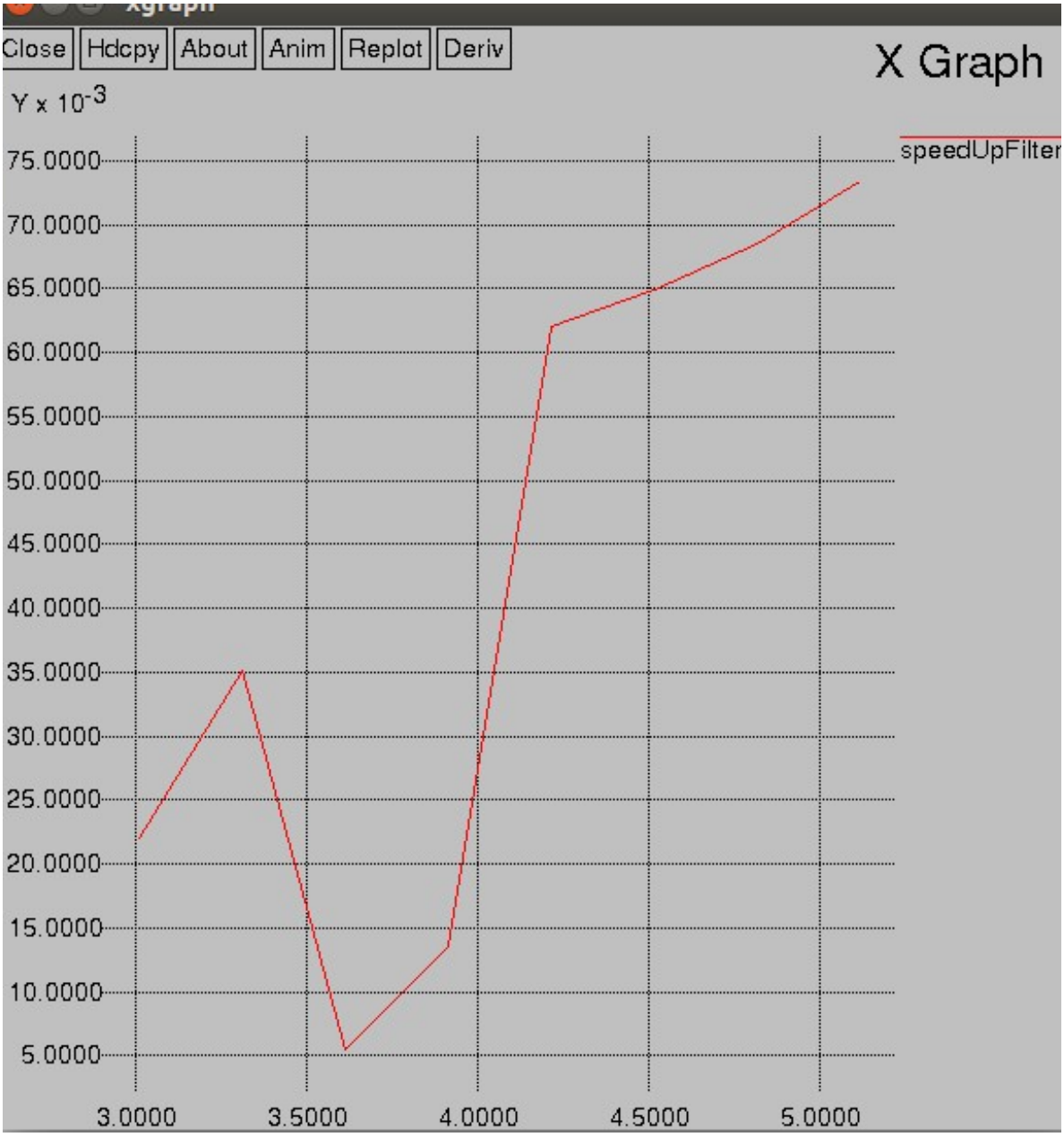
Problem size vs time for Serial implementation



Problem size vs time for Parallel (with scan) implementation



SpeedUp



log(problemSize)