**a)** Here are the results by varying **m1** from 2^1 to 2^15. We take n = 2^20.

**M1 Time Taken**

2 57 milliseconds

4 49 milliseconds

8 31 milliseconds

16 40 milliseconds

32 41 milliseconds

64 42 milliseconds

128 33 milliseconds

256 31 milliseconds

512 52 milliseconds

1024 48 milliseconds

2048 66 milliseconds

4096 108 milliseconds

8192 211 milliseconds

16384 383 milliseconds

32768 1058 milliseconds

We see that there are 2 values of **m1** for which we have the minimum time 31 milliseconds, hence we chose the value of **m1** as 256.

Hence by putting **m1=256** we now find N, be the largest power of 2 such that this algorithm can sort N numbers in less than 5 minutes when run on a single processing core. The value of N found is **2^30**. The time take to run this was **258441** milliseconds

**b)** First we optimize for base case of **m2**. We keep **m3** =2 and vary **m2** from 2^1 to 2^15. We keep n = 2^20 for this.

**m2 Time Taken**

2 137 milliseconds

4 85 milliseconds

8 57 milliseconds

16 41 milliseconds

32 35 milliseconds

64 28 milliseconds

128 24 milliseconds

256 22 milliseconds

512 21 milliseconds

1024 21 milliseconds

2048 20 milliseconds

4096 20 milliseconds

8192 20 milliseconds

16384 20 milliseconds

32768 26 milliseconds

We see that there are several values of **m2** for which we have the minimum time 20 milliseconds, we chose the value of **m2** = 2048 for the subsequent calculations. We now optimize for the base case of **m3**  by keeping **m2** = 2048 and varying **m3**  from 2^1 to 2^15.

**m3 Time Taken**

2 21 milliseconds

4 18 milliseconds

8 16 milliseconds

16 14 milliseconds

32 13 milliseconds

64 16 milliseconds

128 14 milliseconds

256 17 milliseconds

512 19 milliseconds

1024 29 milliseconds

2048 49 milliseconds

4096 92 milliseconds

8192 184 milliseconds

16384 421 milliseconds

32768 940 milliseconds

Here we see that the minimum time is coming for multiple values of **m3**  namely **m3** = 16 and **m3** = 128. We chose **m3** = 128. Hence we get **m2** = 2048 and **m3** = 128.

c) We now plot the running time for **Par-Merge-Sort-SM** by taking 10 equispaced points from n = 1000 to n = 2^30. Here we keep **m1 = 256.**

**Value of n Time taken**

1000 0 milliseconds

97613893 3576 milliseconds

195226786 7407 milliseconds

292839679 11182 milliseconds

390452572 15200 milliseconds

488065465 19716 milliseconds

585678358 23336 milliseconds

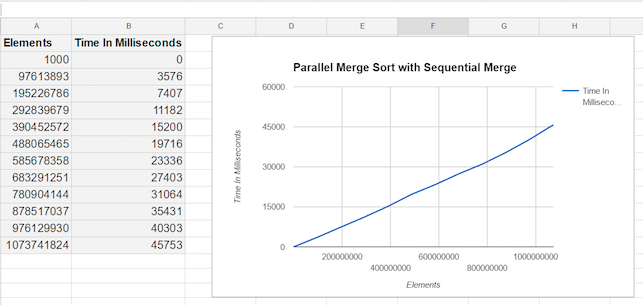
683291251 27403 milliseconds

780904144 31064 milliseconds

878517037 35431 milliseconds

976129930 40303 milliseconds

2^30 45753 milliseconds



We now plot the running time for **Par-Merge-Sort-PM** by taking 10 equispaced points from n = 1000 to n = 2^30. Here we keep **m2** = 256 and **m3** = 128.

**Value of n Time taken**

1000 0 milliseconds

97613893 1868 milliseconds

195226786 3792 milliseconds

292839679 5722 milliseconds

390452572 7888 milliseconds

488065465 10482 milliseconds

585678358 11861 milliseconds

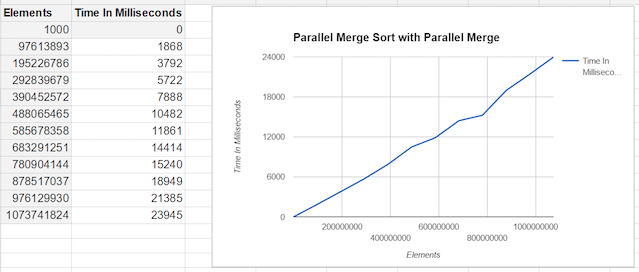
683291251 14414 milliseconds

780904144 15240 milliseconds

878517037 18949 milliseconds

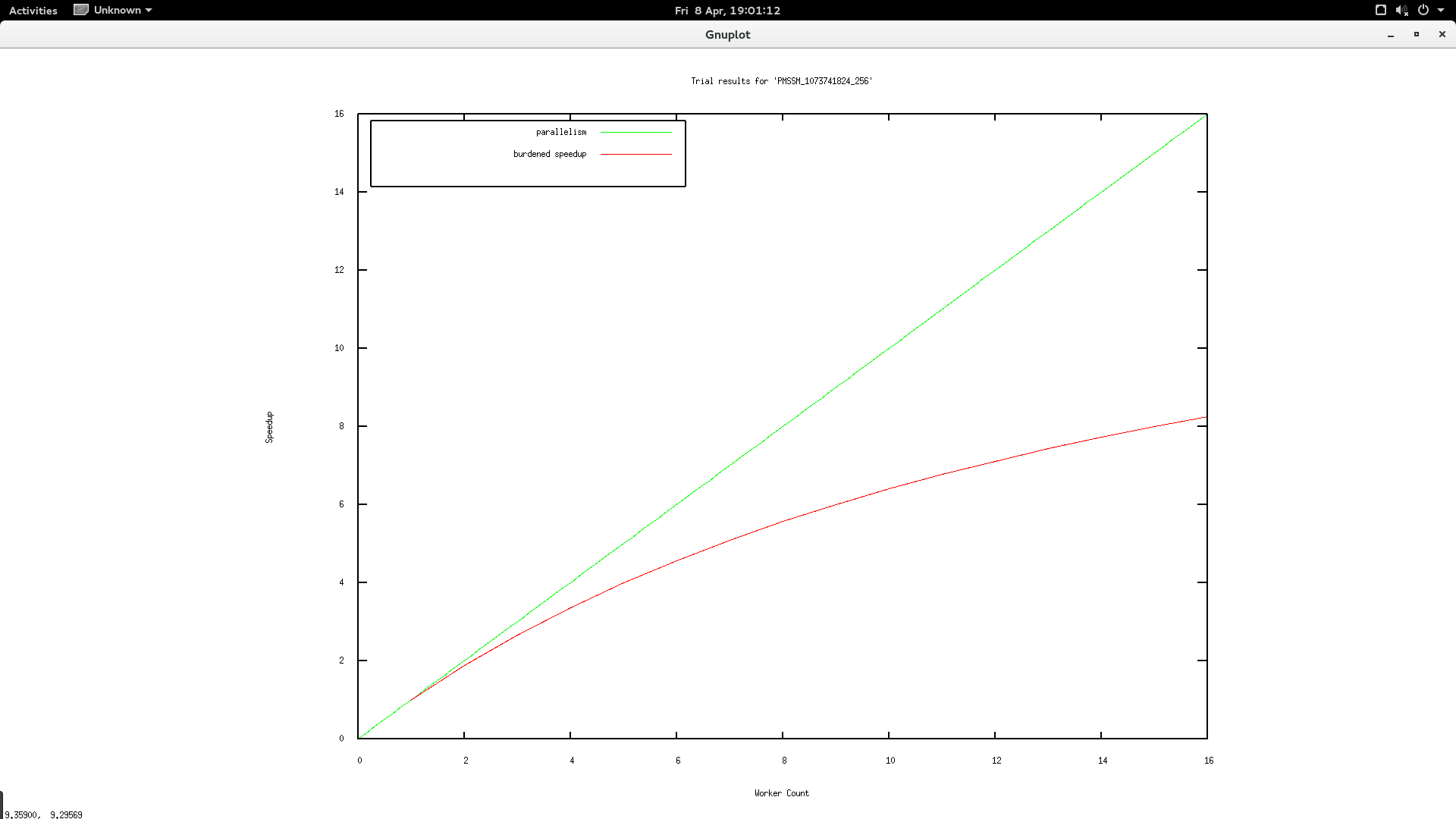
976129930 21385 milliseconds

2^30 23945 milliseconds

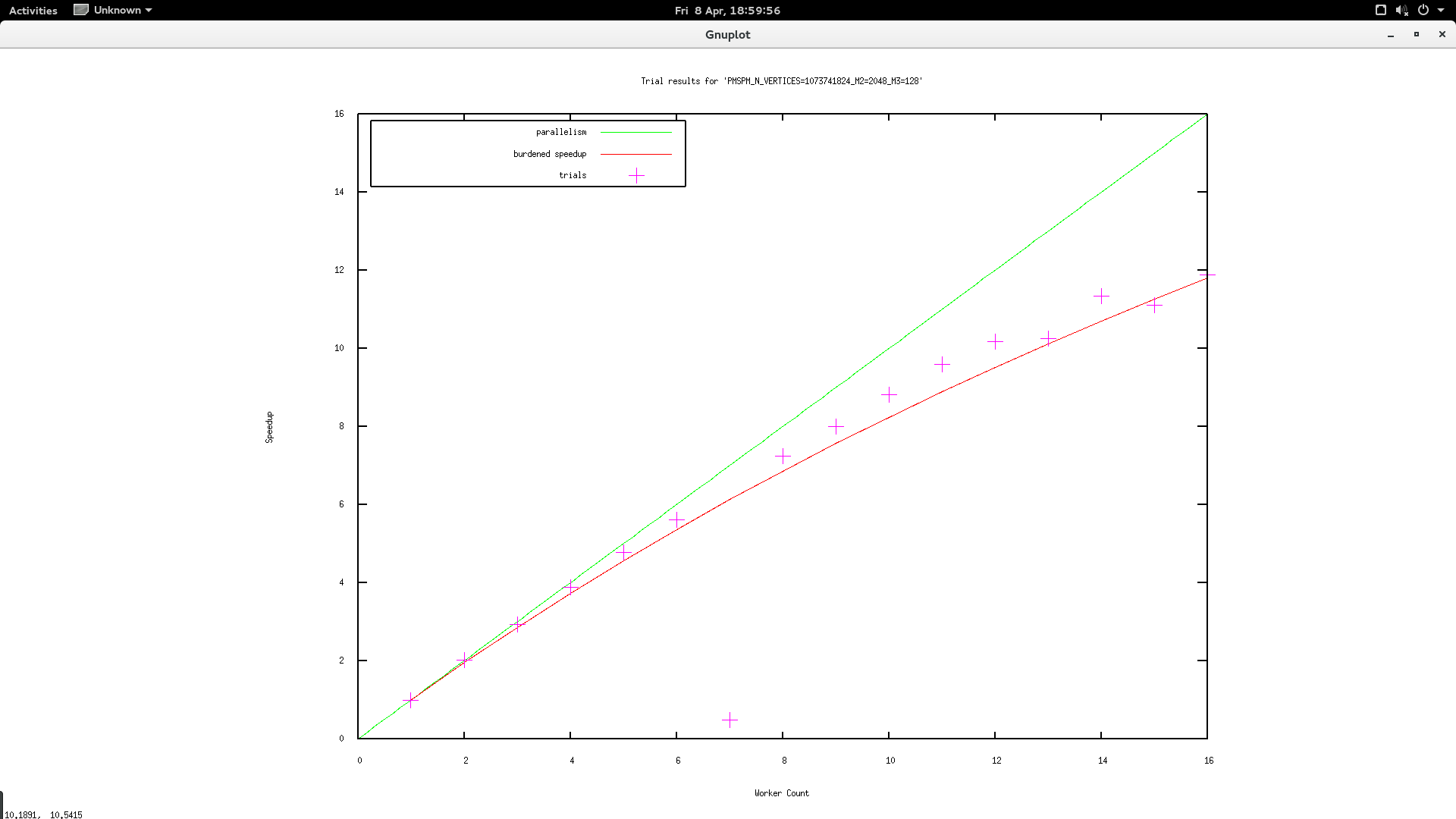


**d)**  The cilkview scalability plots are

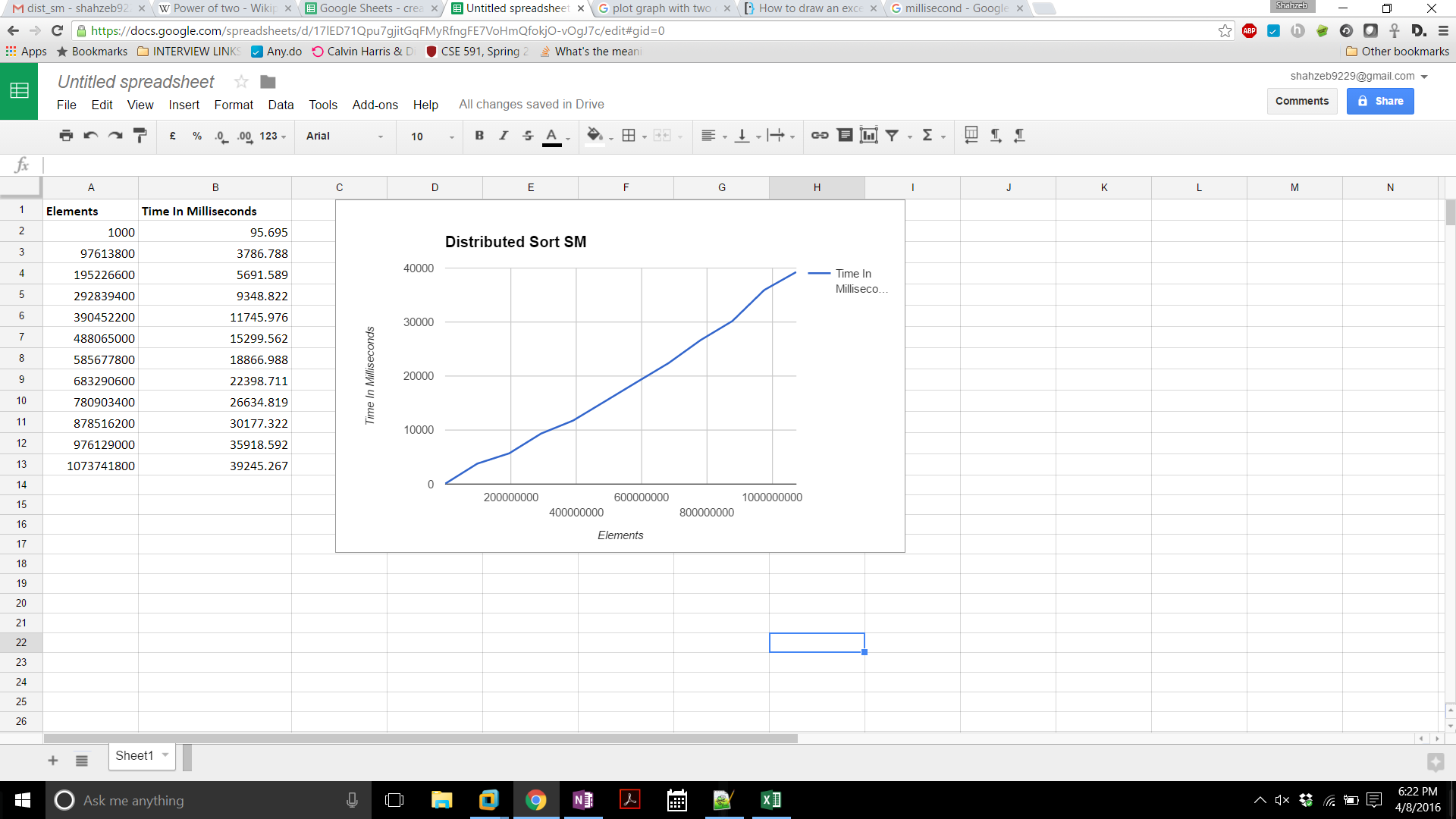
**For Par-Merge-Sort-SM**



**For Par-Merge-Sort-PM**



e) The code for this is in the tarball.



f) 