LAB 3

Introduction

In this laboratory we are going to study the implementation of "divide and conquer" strategy using OpneMP paralization. **(ha de continuar)**

Task decomposition analysis for Mergesort

Divide and conquer

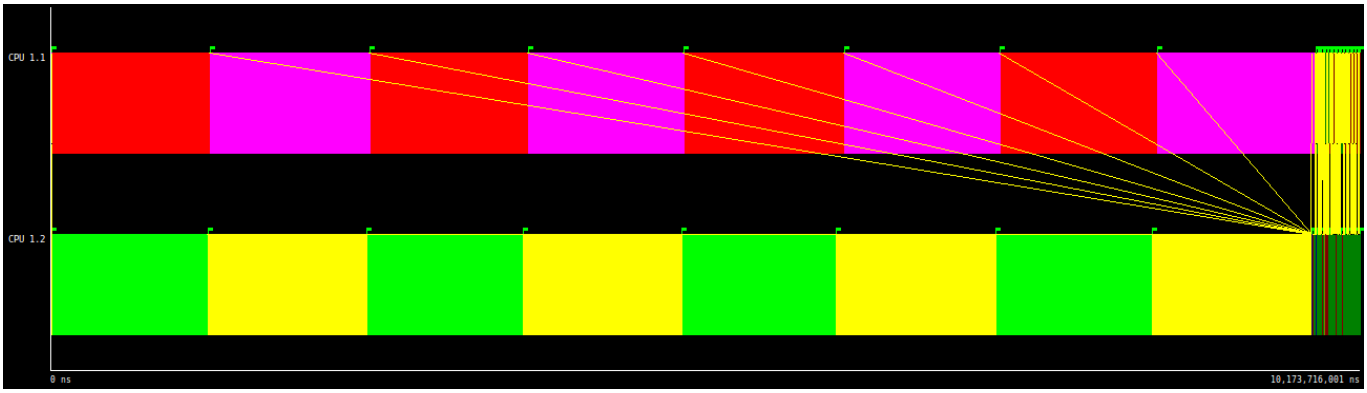
Task decomposition analysis with Tareador

```
ADD CODE
```

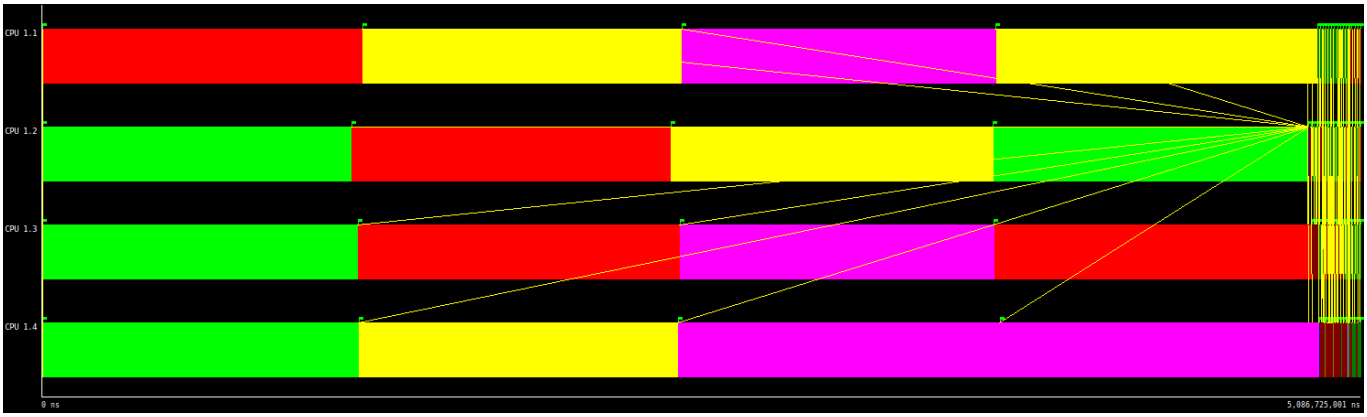
link code

We have added a tareador task in each of the recursive tasks, to fully visualize the possible parallelizations of the code.

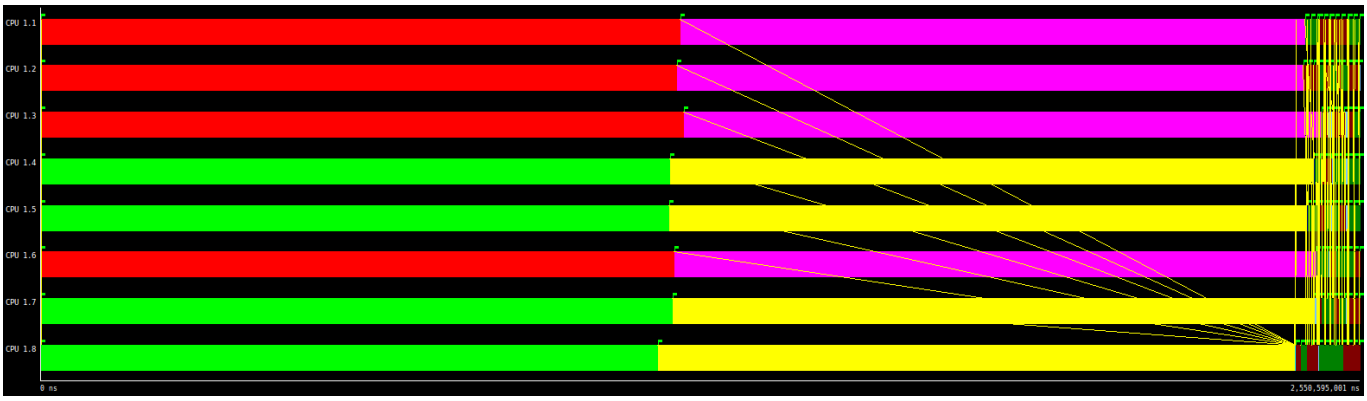
Trace of multisort-tareador using 1 core



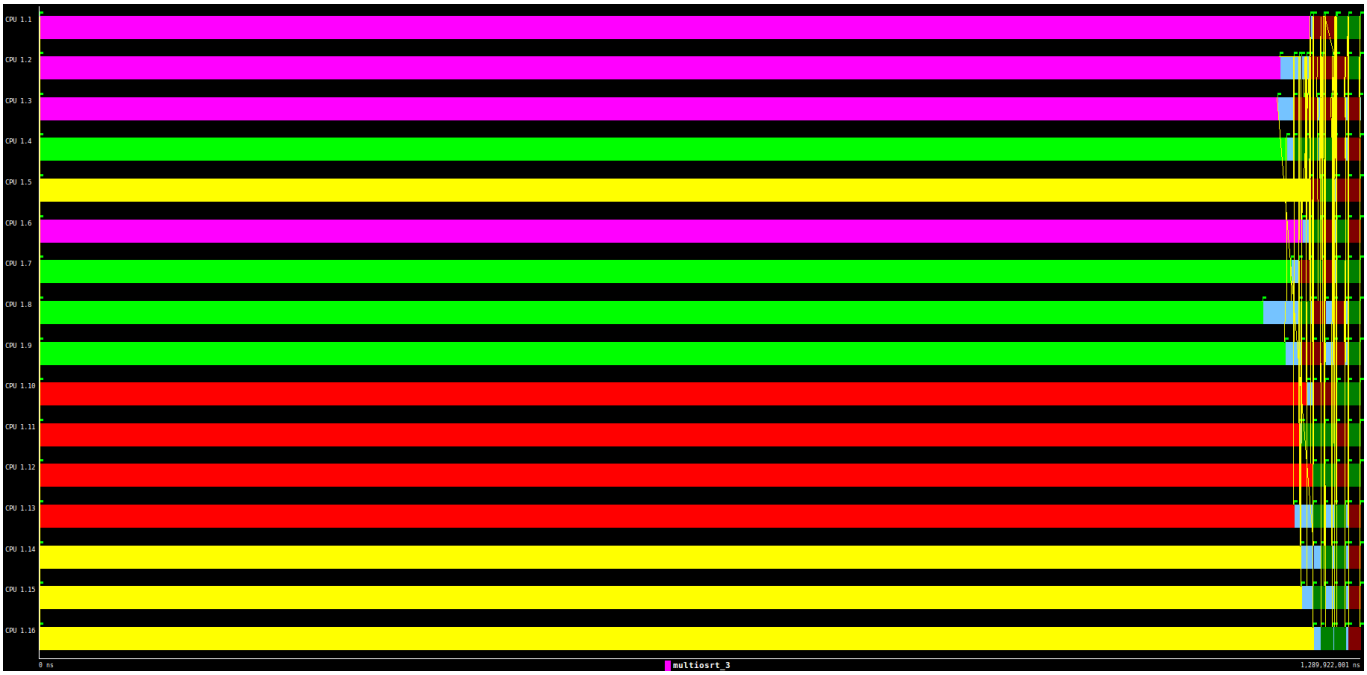
Trace of multisort-tareador using 2 core



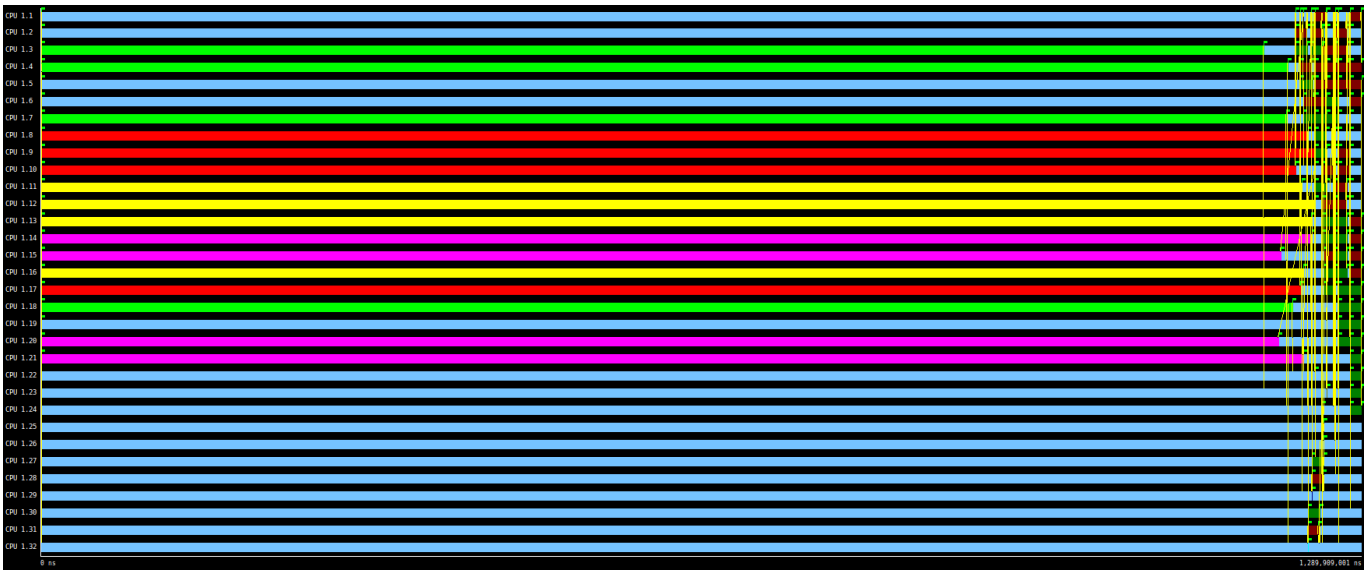
Trace of multisort-tareador using 4 core



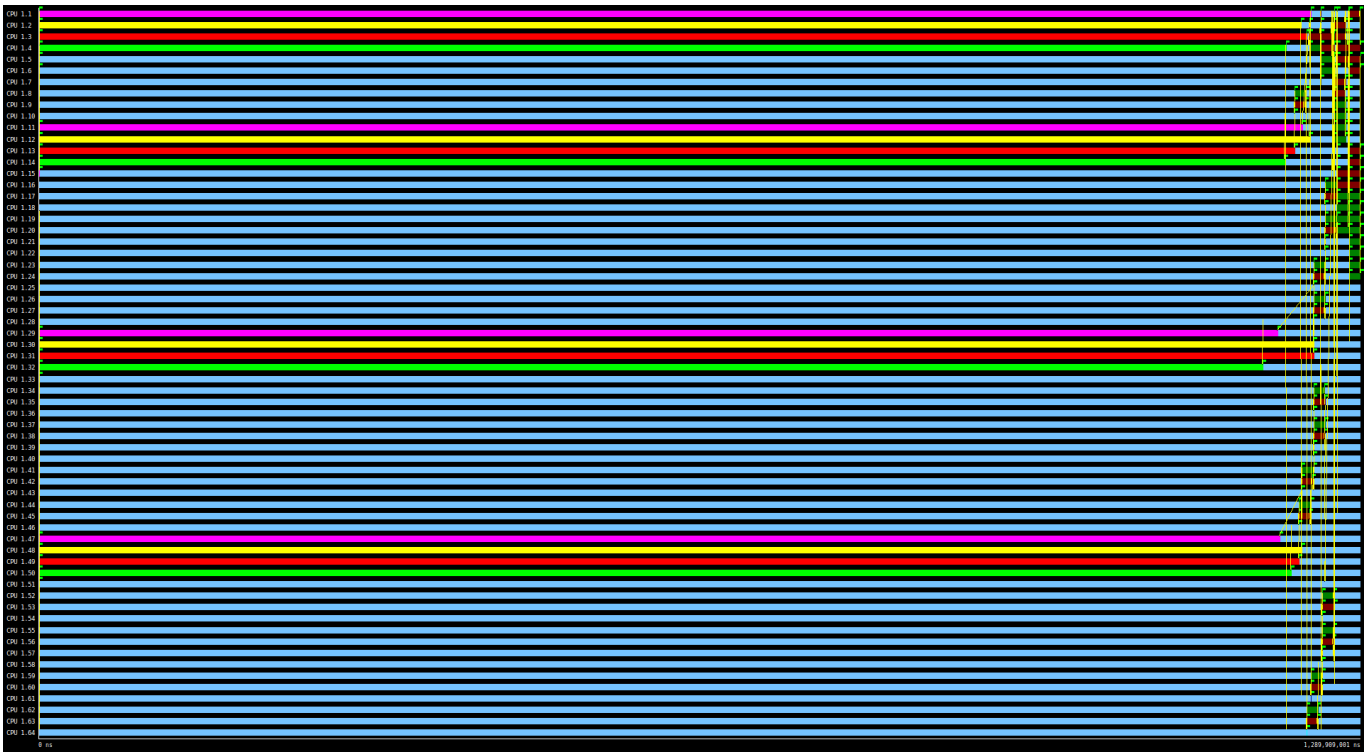
Trace of multisort-tareador using 8 core



Trace of multisort-tareador using 16 core



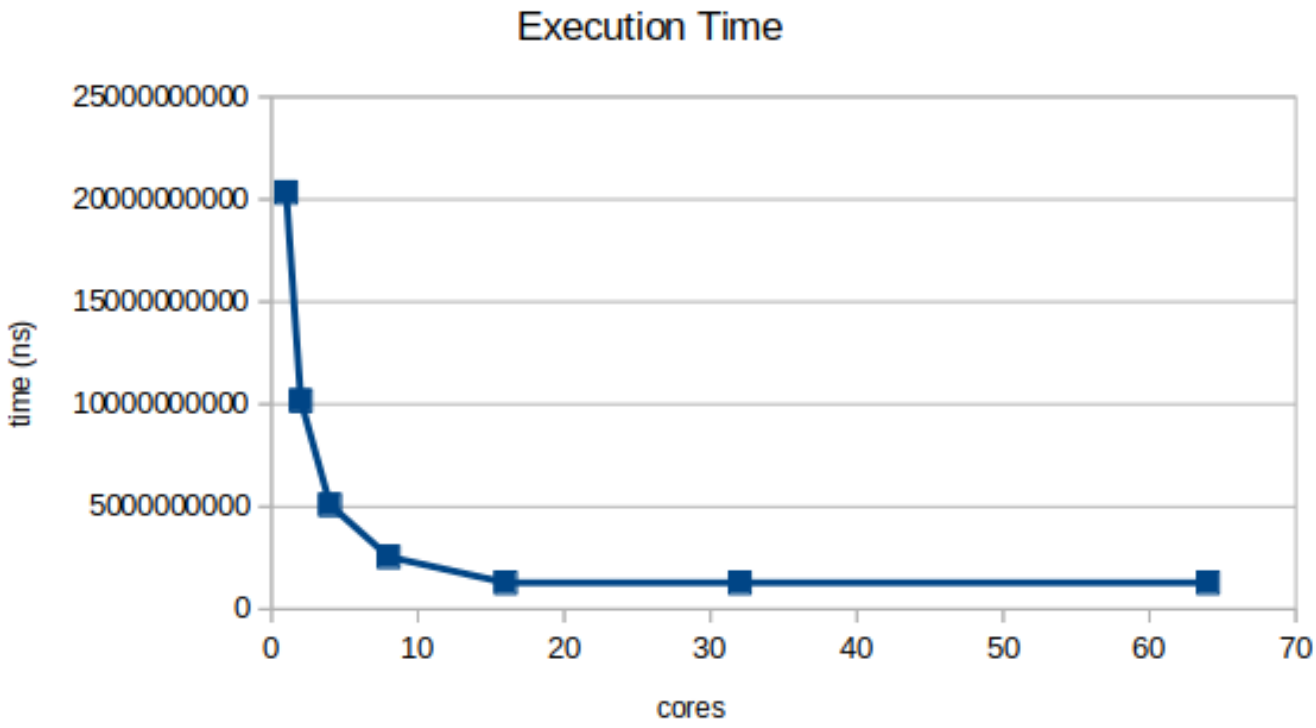
Trace of multisort-tareador using 32 core



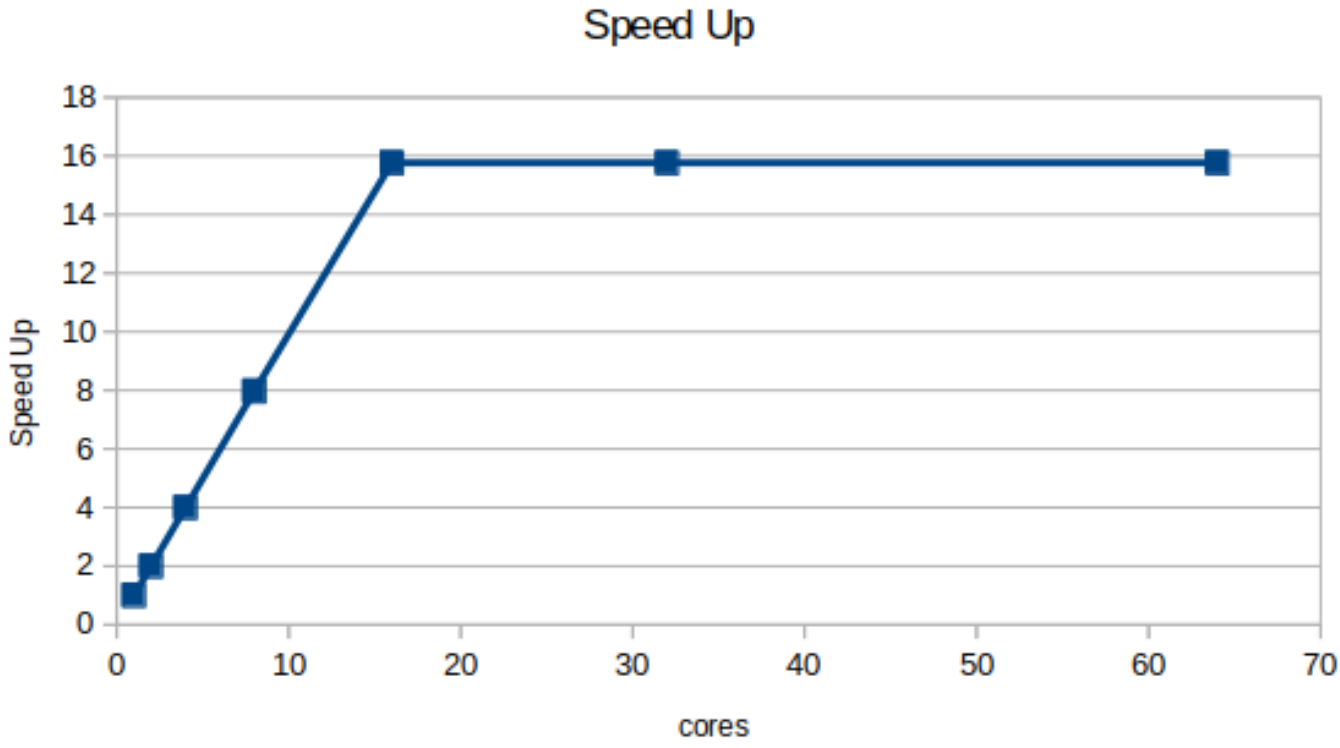
Trace of multisort-tareador using 64 core

NUM CPUs	time (ns)	SPEED UP	EFFICIENCY
1	20334411001	1	0.999360066616823
2	10173716001	1.99872013323365	0.99938619627572
4	5086725001	3.99754478510288	0.99938619627572
8	2550595001	7.97241858979085	0.996552323723856
16	1289922001	15.7640624667507	0.985253904171916
32	1289909001	15.7642213406029	0.492631916893841
64	1289909001	15.7642213406029	0.246315958446921

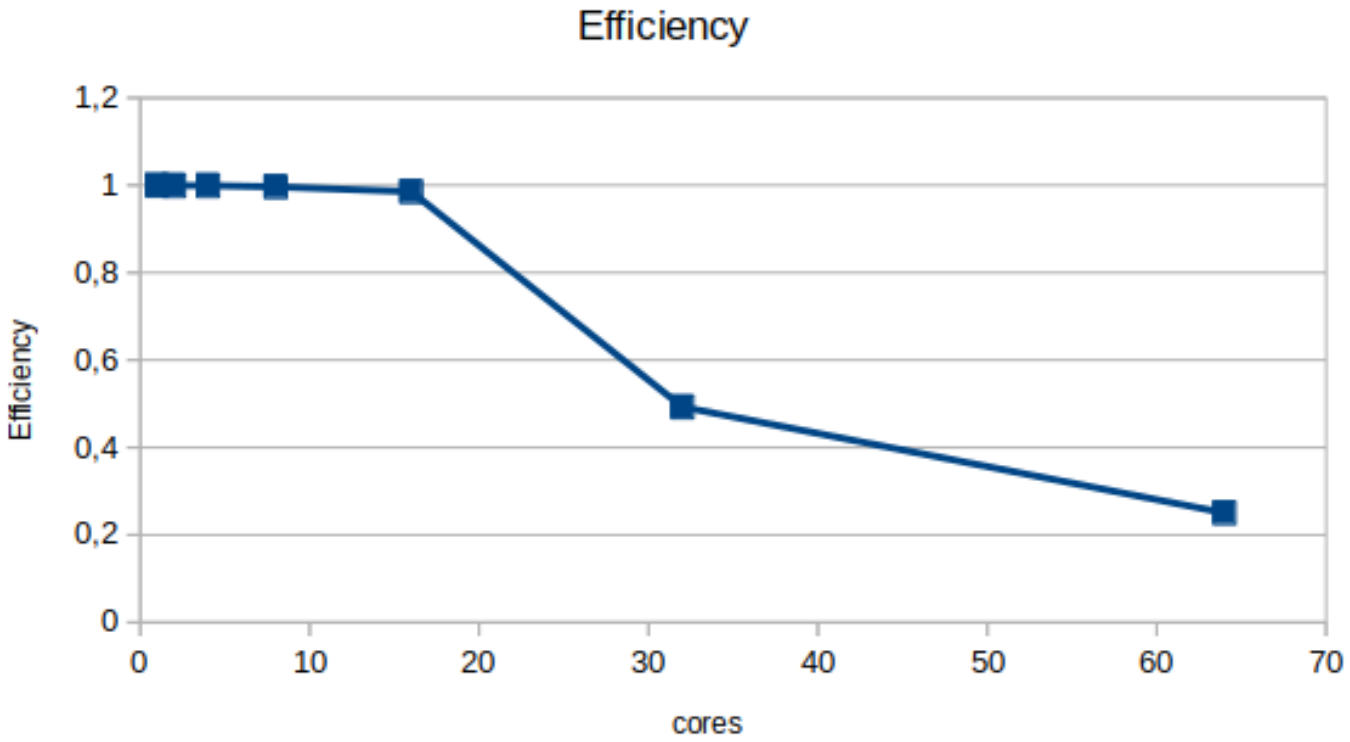
The following pots represents number of cores vs. plot of time, speed up and efficiency, respectively.



Number of cores vs. execution time plot



Number of cores vs. speed up plot



Number of cores vs. efficiency plot

The multisort program parallizes ideally untill 16 cores, when , the dependences between the multisort marge calls, as seen in the tareador capture of the tasks, doesn't allow for more. The efficiency drops going further than 16 threads (i.e fig: efficiency plot and 32,64 captures) Proving that adding more CPUs is pointless.