**2.**

(a) parallelized code

(b) By using block partition, scalability can be nondeterministic. The speedup can be sub-linear, linear or super-linear depends on particular key. Actually, it’s a matter of luck. I will analyze these three scenarios from fundamental origin and convince you with specific data.

There are totally possible keys, we distribute them to p equal parts and give each processor one part.

Then processor i (i from 0 to p-1) will take care of the scope from to . Moreover, each processor’s workload is a factor of 1/p as before.

1. Sub-linear

If my key locates in interval , it will be found in processor 0 by trying as many times as before, which means other processors are wasting of time. Thus, no matter how many processors we have, they cannot shorten our decryption time. I choose the key 101010 in both inp1.txt and inp2.txt. The results are shown in below: speedup decreases a little due to overhead created by communication between processors as number of thread increases.

A screenshot of a cell phone

Description automatically generated

1. Linear

When key , the speedup can be linear for either number of processors. It can be understood as the number of attempts to reach the last one will be a factor of compared with initial case. Note that in tests of linear and super-linear scenarios, I truncate the scope of key to . It saves computation resources and has not much influence on our observation of these scenarios.

In the figure below, key = . We can see linear relations for both files.

A close up of a map

Description automatically generated

1. Super-Linear

If key locates near the starting points of one processor other than 0, it takes less time to be found soon by this processor. Then we achieve super-liner speedup.

I choose key = and the results are shown in the figure below. This key has same distance from a certain processor when p is even, so we can expect similar execution time for any even p. We can see that the figure has great agreement on above analysis.

A close up of a map

Description automatically generated

(c) Such dramatic fluctuation of speedup can be attribute the our partition strategy. In order to improve current situation, I suggest to use cyclic partition. The advantage is as stated below: As , unless key locates at beginning few numbers, we can achieve nearly linear speedup. The reason is that the number of attempts to reach either number will be reduced by a factor of . It’s also a smooth process.

A close up of a map

Description automatically generated