CS 3502
Operating Systems

Project 2 Lab

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Project 2

- Read and write parallel program using Pthread
- Learn to use Pthread functions
- User level projects, not kernel code

Assignment 1

- Given two character strings s1 and s2. Write a Pthread program to find out the number of substrings, in string s1, that is exactly the same as s2.

- https://github.com/kevinsuo/CS3502/blob/master/project-2-1.c

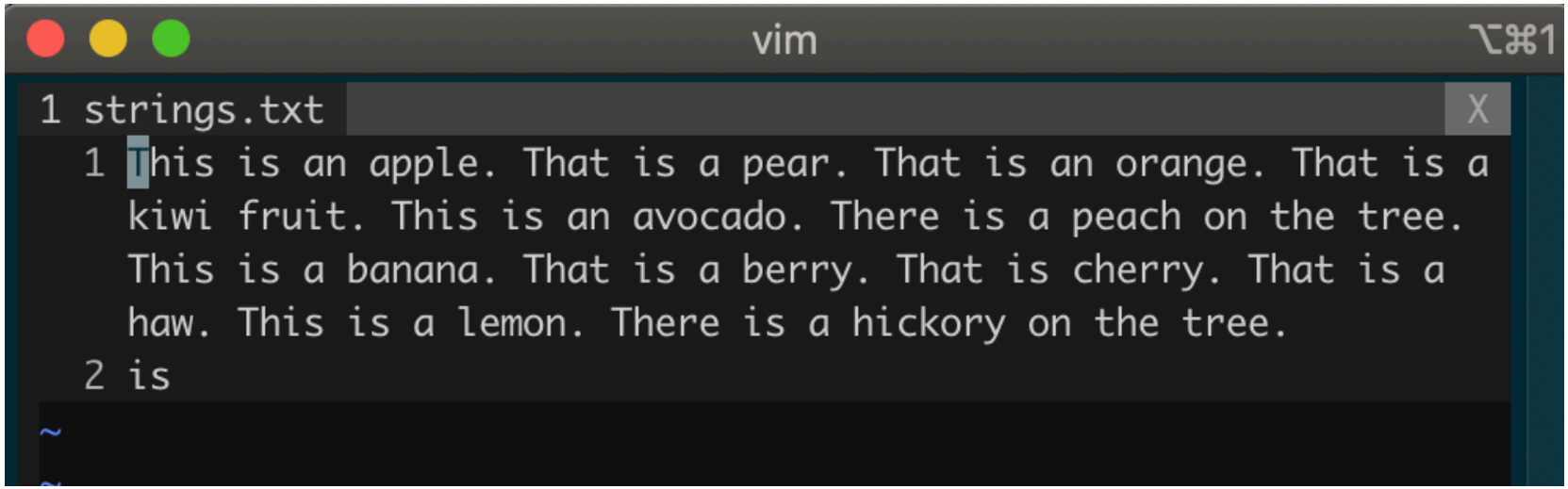
Assignment 1 Examples

- number_substring(“abcdab”, “ab”) = 2,
- number_substring(“aaa”, “a”) = 3,
- number_substring(“abac”, “bc”) = 0.

Subsequence not substring

Assignment 1

- Input file: https://github.com/kevinsuo/CS3502/blob/master/strings.txt



```c
int main(int argc, char *argv[])
{
    int count;

    readf(fp);
    count = num_substring();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```

```c
int total = 0;
int n1,n2;
char *s1,*s2;
FILE *fp;

int readf(FILE *fp)
{
    if((fp=fopen("strings.txt", "r"))==NULL){
        printf("ERROR: can't open string.txt!\n");
        return 0;
    }
    s1=(char *)malloc(sizeof(char)*MAX);
    if(s1==NULL){
        printf("ERROR: Out of memory!\n");
        return -1;
    }
    s2=(char *)malloc(sizeof(char)*MAX);
    if(s2==NULL){
        printf("ERROR: Out of memory\n");
        return -1;
    }
    /*read s1 s2 from the file*/
    s1=fgets(s1, MAX, fp);
    s2=fgets(s2, MAX, fp);
    n1=strlen(s1);  /*length of s1*/
    n2=strlen(s2)-1; /*length of s2*/

    if(s1==NULL || s2==NULL || n1<n2)  /*when error exit*/
        return -1;
    return 0;
}
```

Use strlen(s2)-1
Because there exist a
'\n' at the end of s2

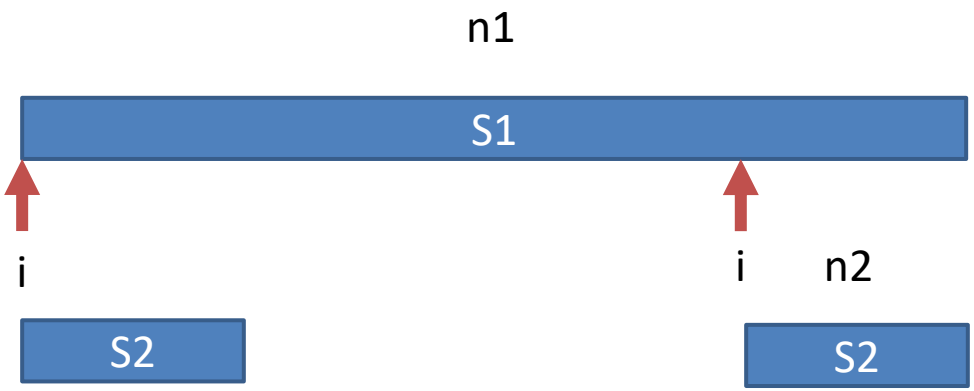
‘\n’ at the end of s2

```
vim                                                    ⌥⌘1
1 strings.txt                                          X
  1 This is an apple. That is a pear. That is an orange. That is a
    kiwi fruit. This is an avocado. There is a peach on the tree.
    This is a banana. That is a berry. That is cherry. That is a
    haw. This is a lemon. There is a hickory on the tree.
  2 is
~
```

Use strlen(s2)-1
Because there exist a
‘\n’ at the end of s2

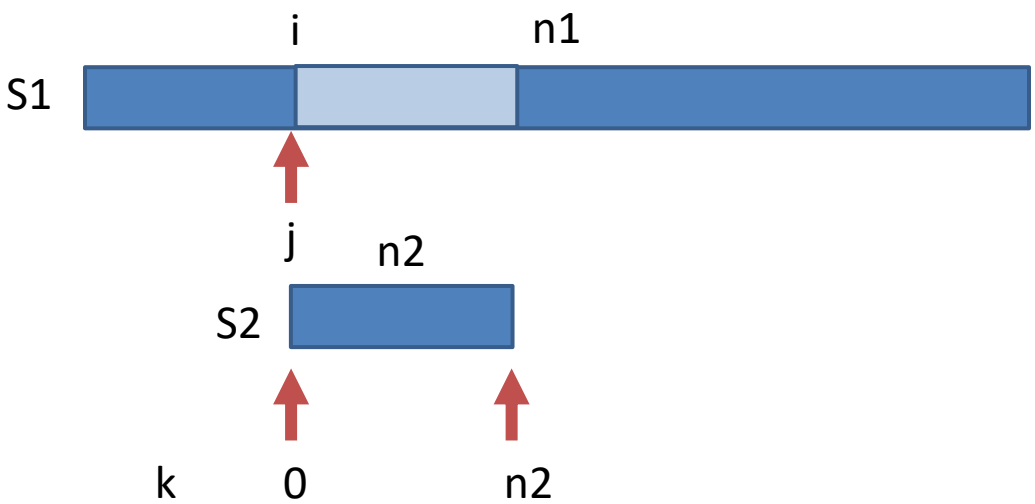
```c
int main(int argc, char *argv[])
{
    int count;

    readf(fp);
    count = num_substring();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```

```c
int num_substring(void)
{
    int i,j,k;
    int count;

    for (i = 0; i <= (n1-n2); i++){
        count=0;
        for(j = i,k = 0; k < n2; j++,k++){  /*search for the next
string of size of n2*/
            if (*(s1+j)!=*(s2+k)){
                break;
            }else{
                count++;
            }

            if(count==n2){
                total++;        /*find a substring in this
step*/
            }
        }
    }
    return total;
}
```



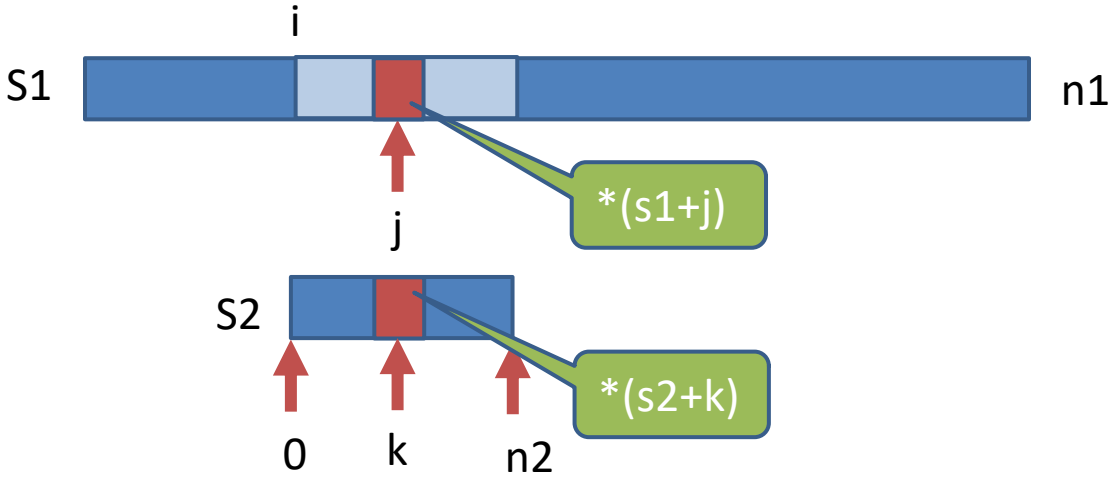
```c
int main(int argc, char *argv[])
{
    int count;

    readf(fp);
    count = num_substring();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```

```c
int num_substring(void)
{
    int i,j,k;
    int count;

    for (i = 0; i <= (n1-n2); i++){
        count=0;
        for(j = i,k = 0; k < n2; j++,k++){  /*search for the next
string of size of n2*/
            if (*(s1+j)!=*(s2+k)){
                break;
            }else{
                count++;
            }

            if(count==n2){
                total++;        /*find a substring in this
step*/
            }
        }
    }
    return total;
}
```



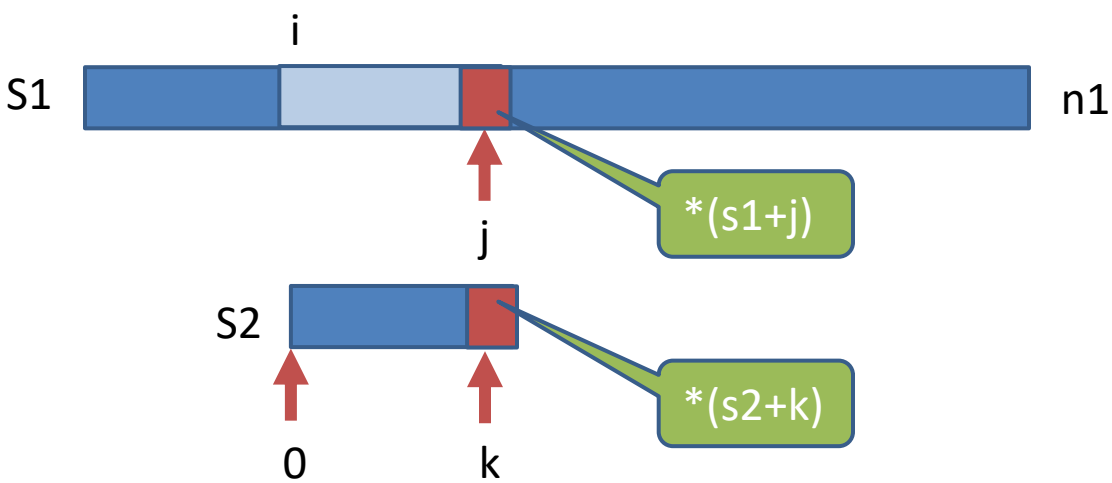
```c
int main(int argc, char *argv[])
{
    int count;

    readf(fp);
    count = num_substring();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```

```c
int num_substring(void)
{
    int i,j,k;
    int count;

    for (i = 0; i <= (n1-n2); i++){
        count=0;
        for(j = i,k = 0; k < n2; j++,k++){  /*search for the next
string of size of n2*/
            if (*(s1+j)!=*(s2+k)){
                break;
            }else{
                count++;
            }

            if(count==n2){
                total++;        /*find a substring in this
step*/
            }
        }
    }
    return total;
}
```



```c
int main(int argc, char *argv[])
{
    int count;

    readf(fp);
    count = num_substring();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```

```c
int num_substring(void)
{
    int i,j,k;
    int count;

    for (i = 0; i <= (n1-n2); i++){
        count=0;
        for(j = i,k = 0; k < n2; j++,k++){  /*search for the next
string of size of n2*/
            if (*(s1+j)!=*(s2+k)){
                break;
            }else{
                count++;
            }

            if(count==n2){
                total++;        /*find a substring in this
step*/
            }
        }
    }
    return total;
}
```



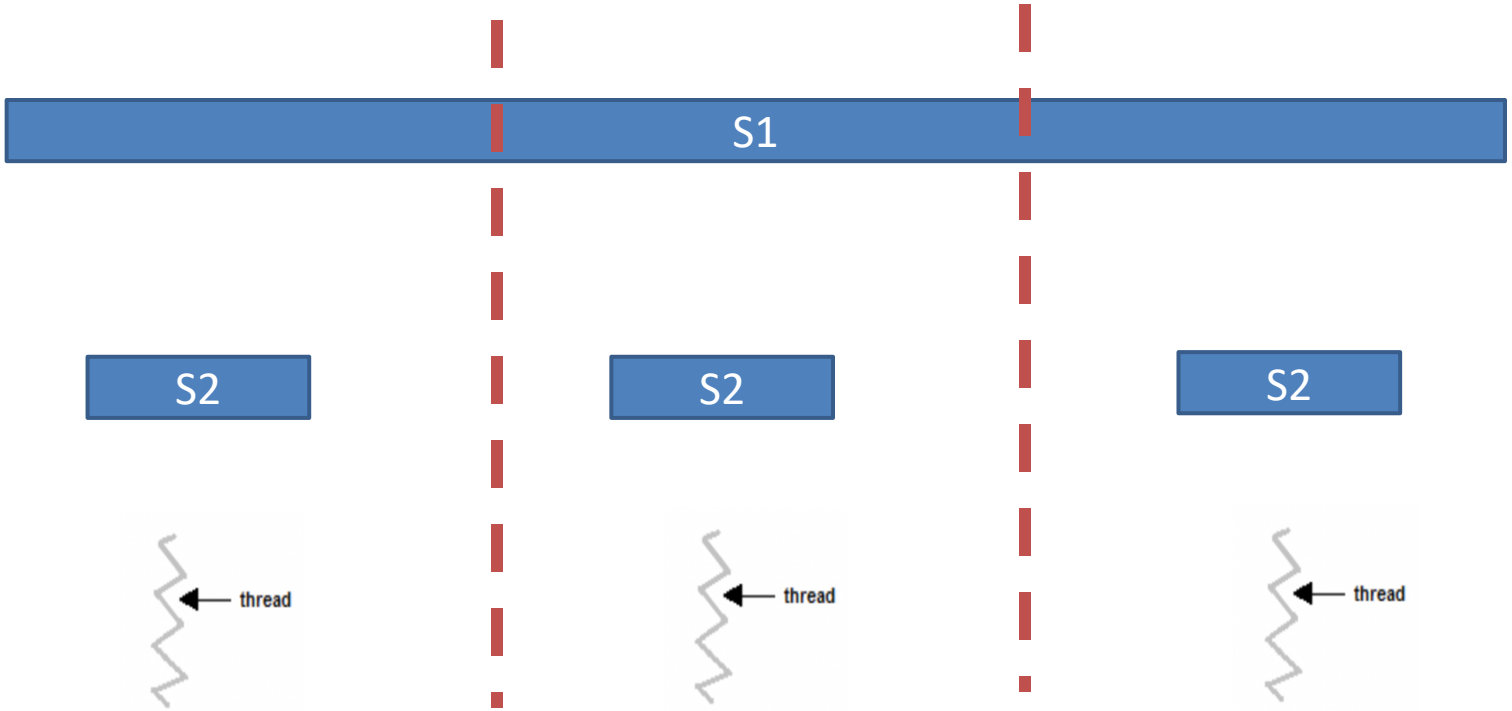
Assignment 1

```c
int main(int argc, char *argv[])
{
    int count;

    readf(fp);
    count = num_substring();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```

```
fish
ksuo@ltksup50143mac ~/O/C/P/p/c/assignment 1> ./test.o
The number of substrings is: 20
```

Assignment 1

- Write a parallel program using Pthread based on this sequential solution.



Assignment 1

pthread_create (thread, attr, start_routine, arg)

- **creates a thread and makes it executable;**
- **1st parameter: pointer to the thread**
- **2rd parameter: set attributes to threads**
- **3rd parameter: the function for the thread to run**
- **4th parameter: parameter for thread function**

pthread_exit (status)

• **If** main() **finishes before the threads it has created, and exists with the** pthread_exit(), **the other threads will continue to execute. Otherwise, they will be automatically terminated when** main() **finishes**

Start from this template using pthread

```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <stdlib.h>

int num[3];

void *myThread(void *threadid)
{
    int i = (int)threadid;
    num[i] = i;
    pthread_exit(NULL);
}


int main()
{
    int rc, t=0;
    pthread_t threads[3];

    for(t=0; t<3; t++){
         rc = pthread_create(&threads[t], NULL, myThread, (void *) (size_t)t);
         if (rc){
                   printf("ERROR; return code from pthread_create() is %d\n", rc);
                   exit(1);
         }
    }

    for(t=0; t<3; t++){
          pthread_join(threads[t], NULL);
    }

    for(t=0; t<3; t++) {
        printf("num:%d\n", num[t]);
    }
    return 0;
}
```

Pass the parameter into pthread

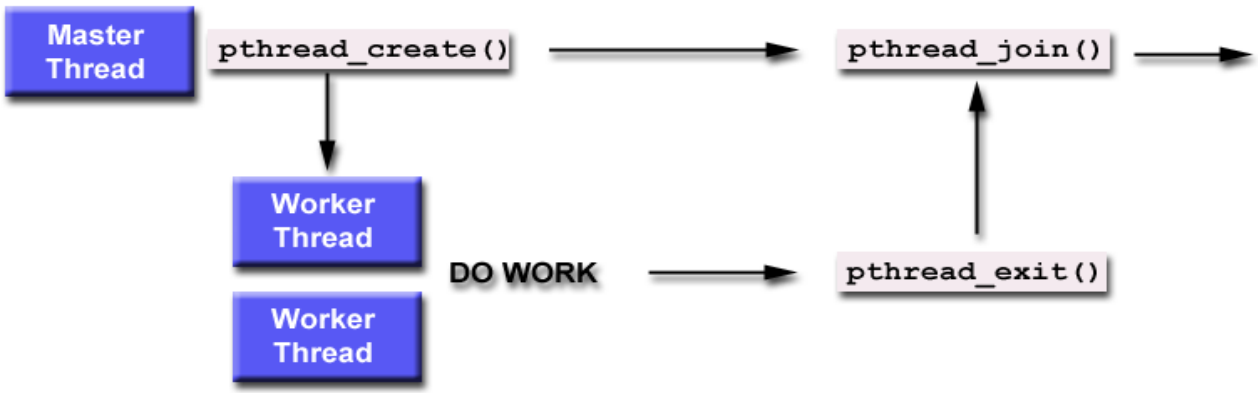
vim

```
ksuo@ltksup50143mac ~/Desktop> ./pthread_join.o
num:0
num:1
num:2
```

Create 3 threads:
The nth thread print number n

https://github.com/kevinsuo/CS3502/blob/master/project2-template-code.c

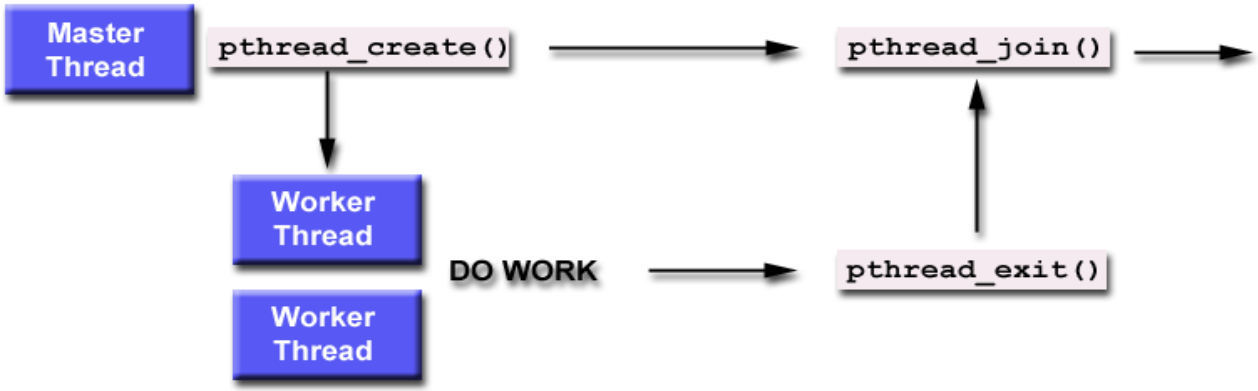
Assignment 1

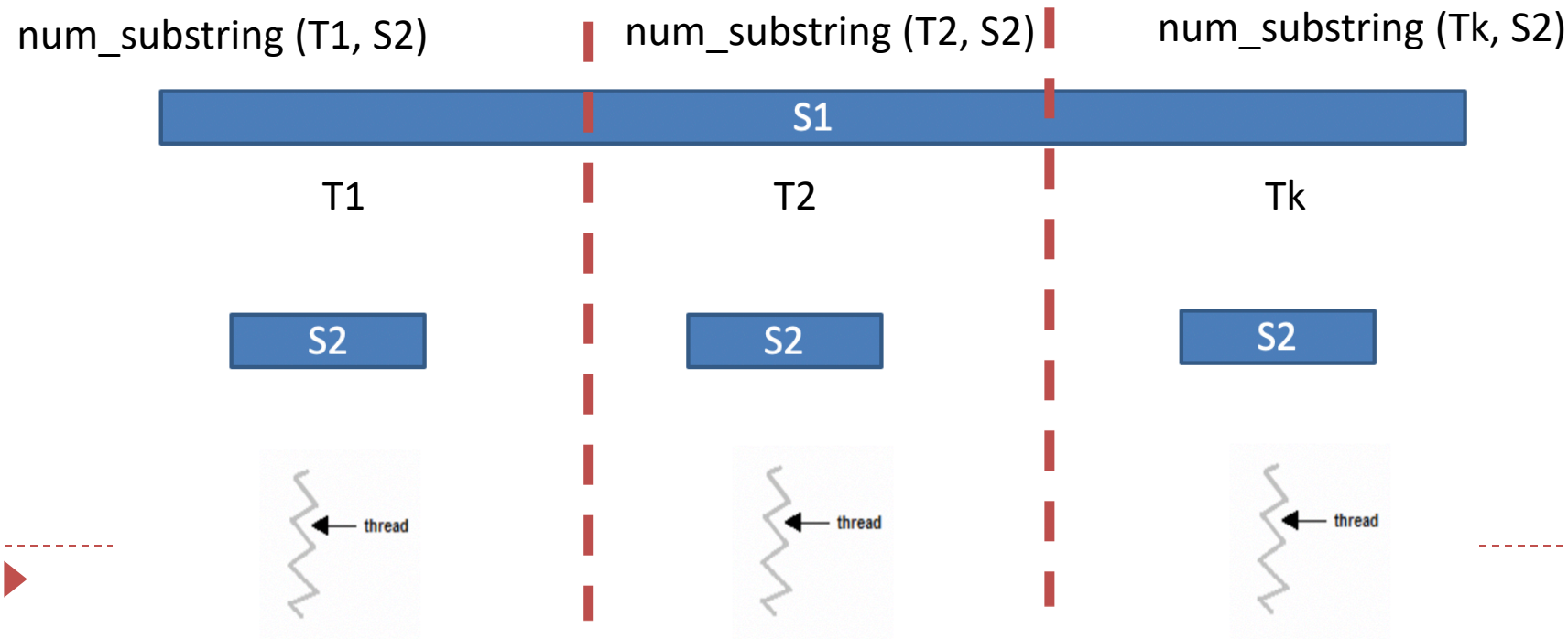



```
Main()
{
        //create multiple threads
        pthread_create

        //wait thread to finish in main
         pthread_join

        //sum up the number from each thread
        sum = num[0]+ num[1]+ … num[k]
}
```

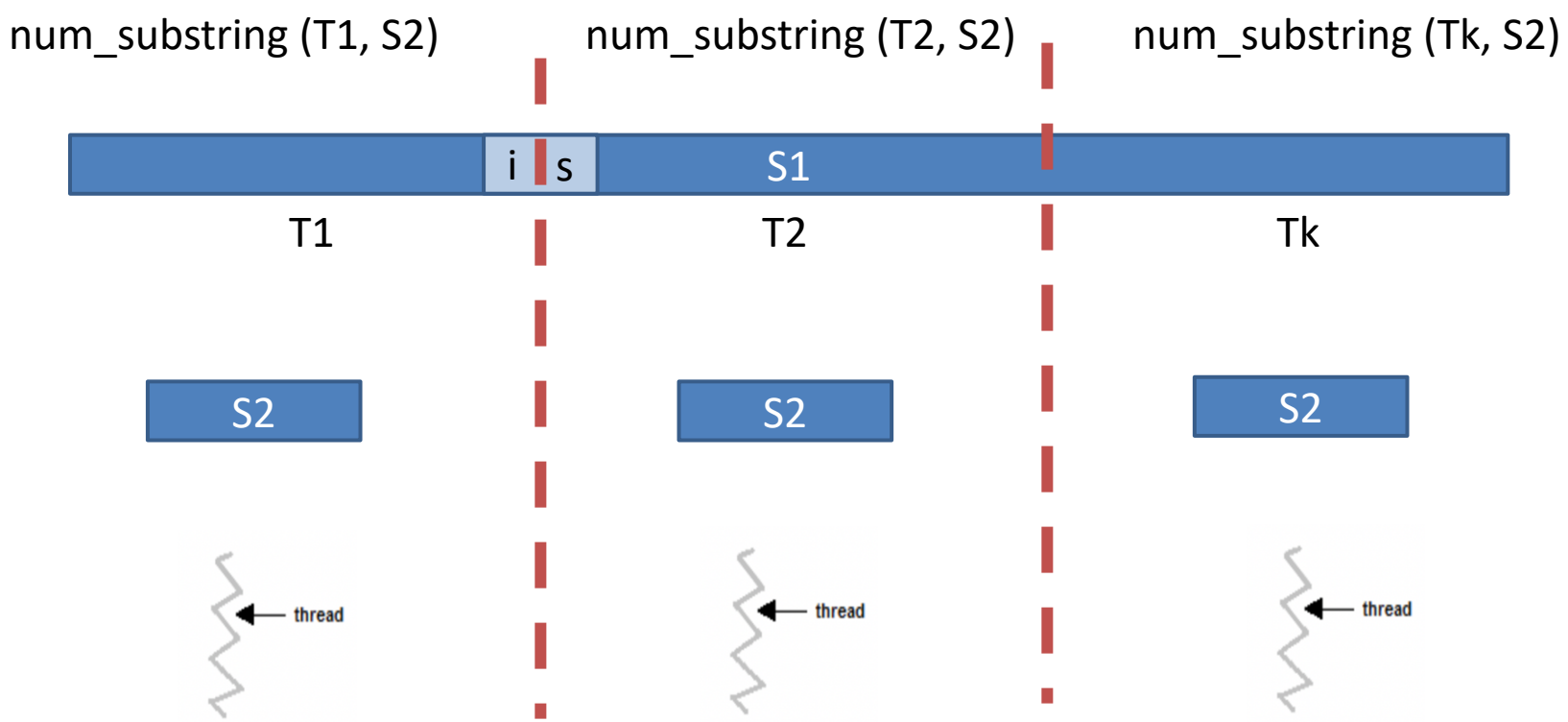
```
thread()
{
        //do something
}
```

Assignment 1





Corner Case in Assignment 1



Verify whether your parallel thread is correct

- Modify the strings.txt by yourself
- Compare the sequential and parallel program results that whether they are the same

```
ksuo@ltksup50143mac ~/O/C/P/p/c/assignment 1> ./test-p.o
This is thread 0
This is thread 3
This is thread 4
This is thread 2
This is thread 1
The number of substrings is : 20
ksuo@ltksup50143mac ~/O/C/P/p/c/assignment 1> ./test.o
The number of substrings is: 20
```

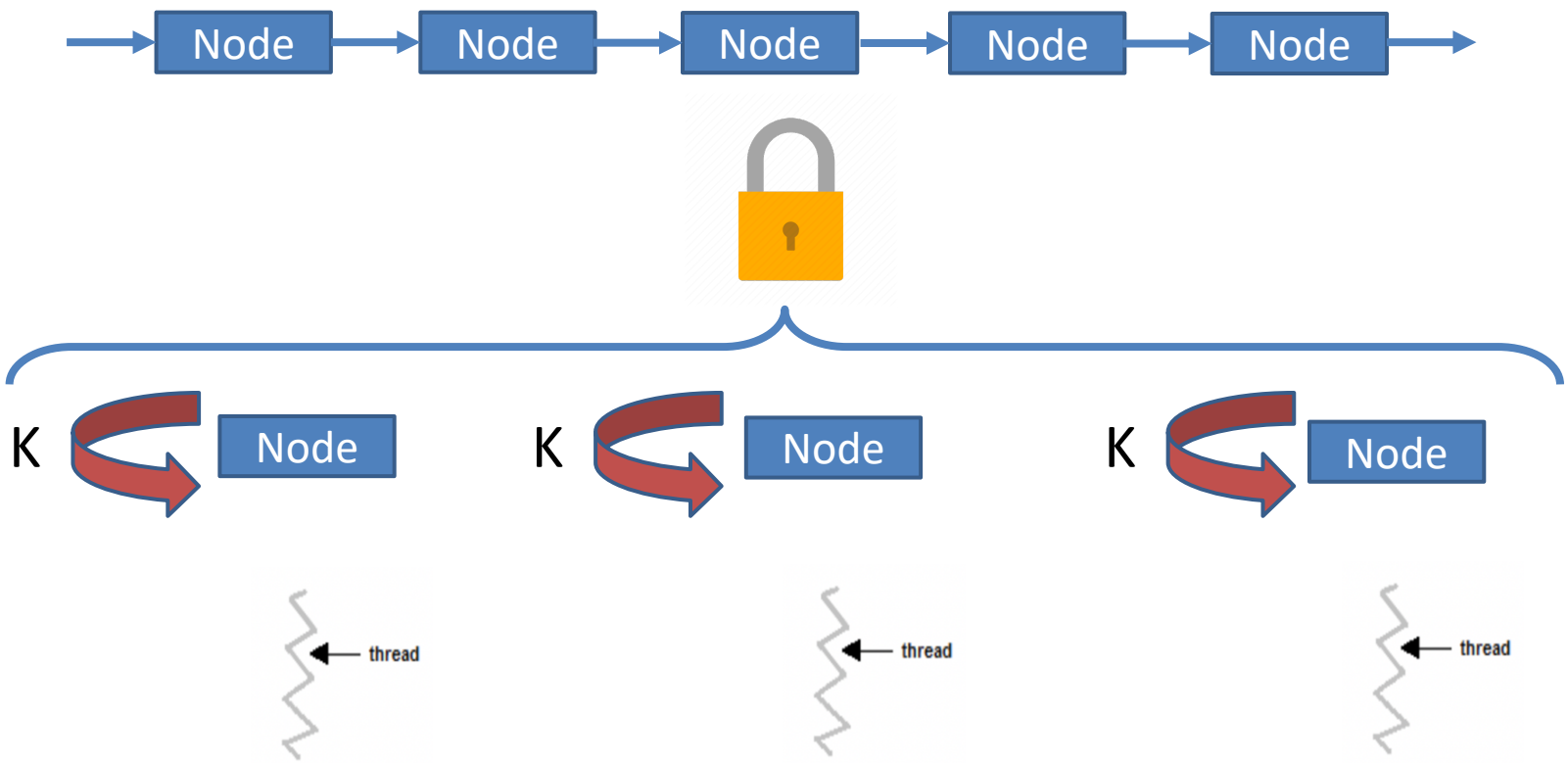
Assignment 2

- Read the following program and modify the program to improve its performance

- https://github.com/kevinsuo/CS3502/blob/master/project-2-2.c

Assignment 2

- Each thread creates a data node and attaches it to a global list. This operation is repeated for K times by each thread.



Assignment 2

- Make sure your VM to have more cores

(1) Shutdown your VM and change to VM setting to use 4 vCPUs.

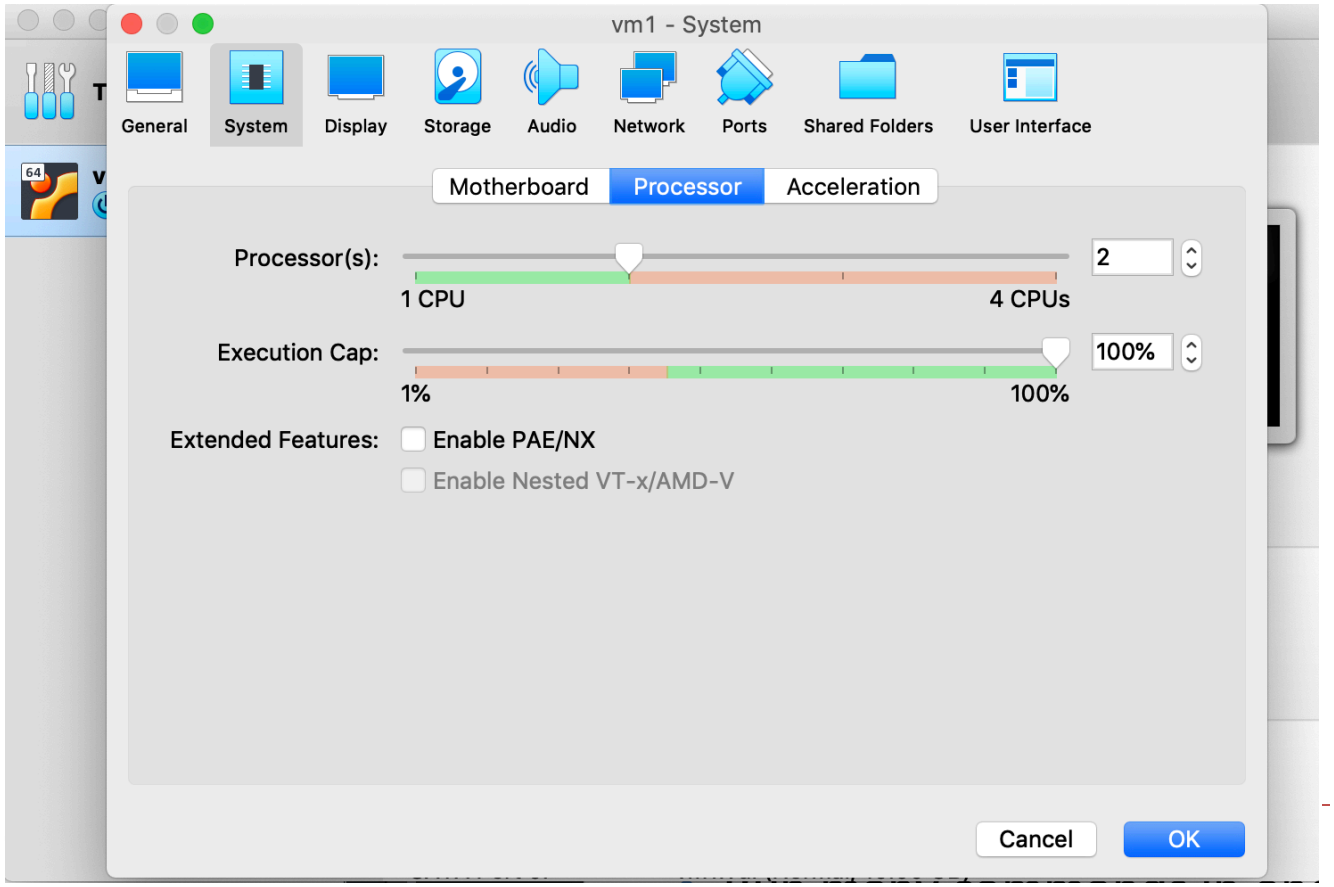
(2) Verify that you VM has 4 vCPUs:

```
$ cat /proc/cpuinfo
```

You should have 4 CPUs (processor: 0-3).

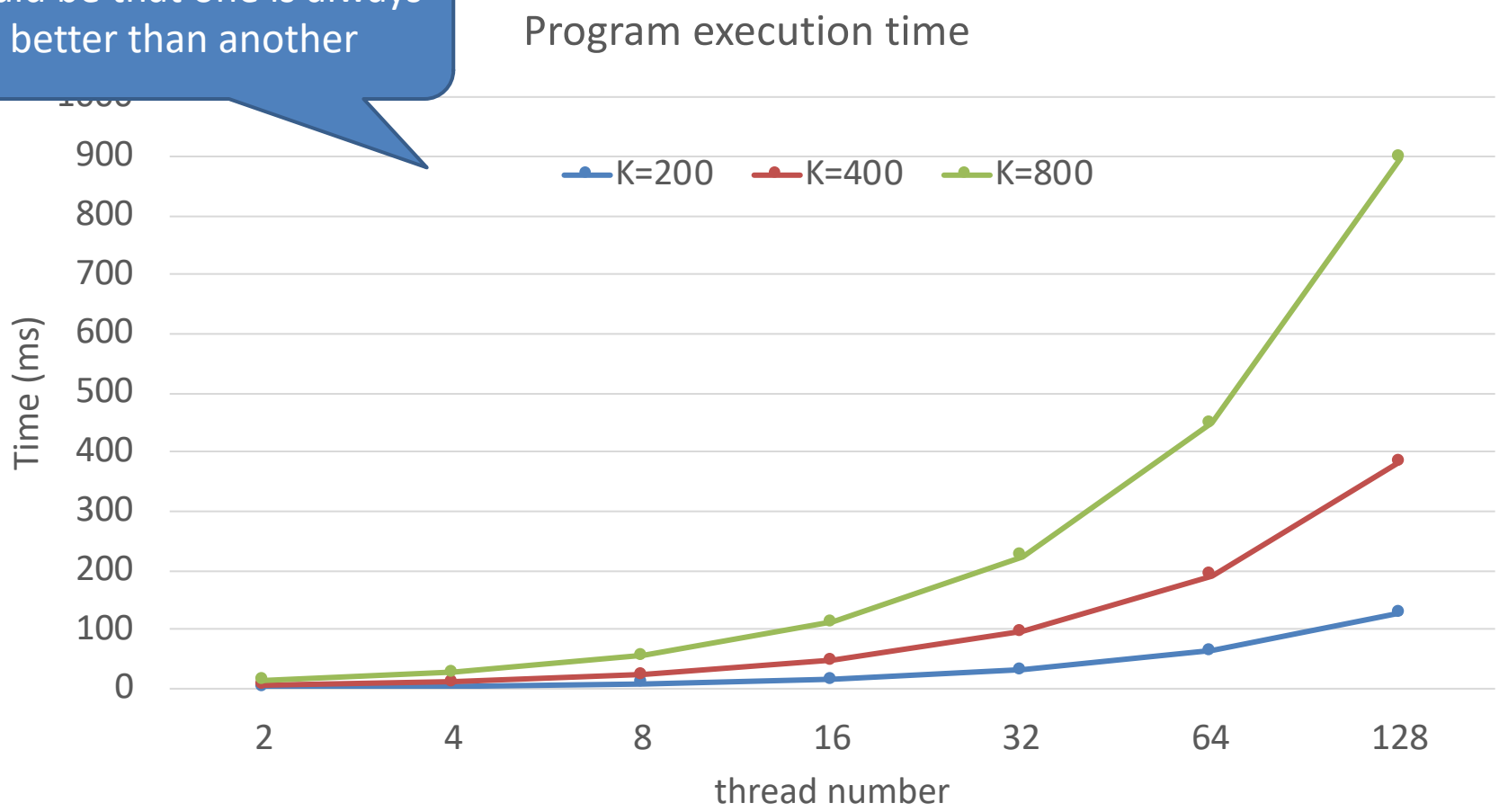
Tips

- Set up more CPUs for VM



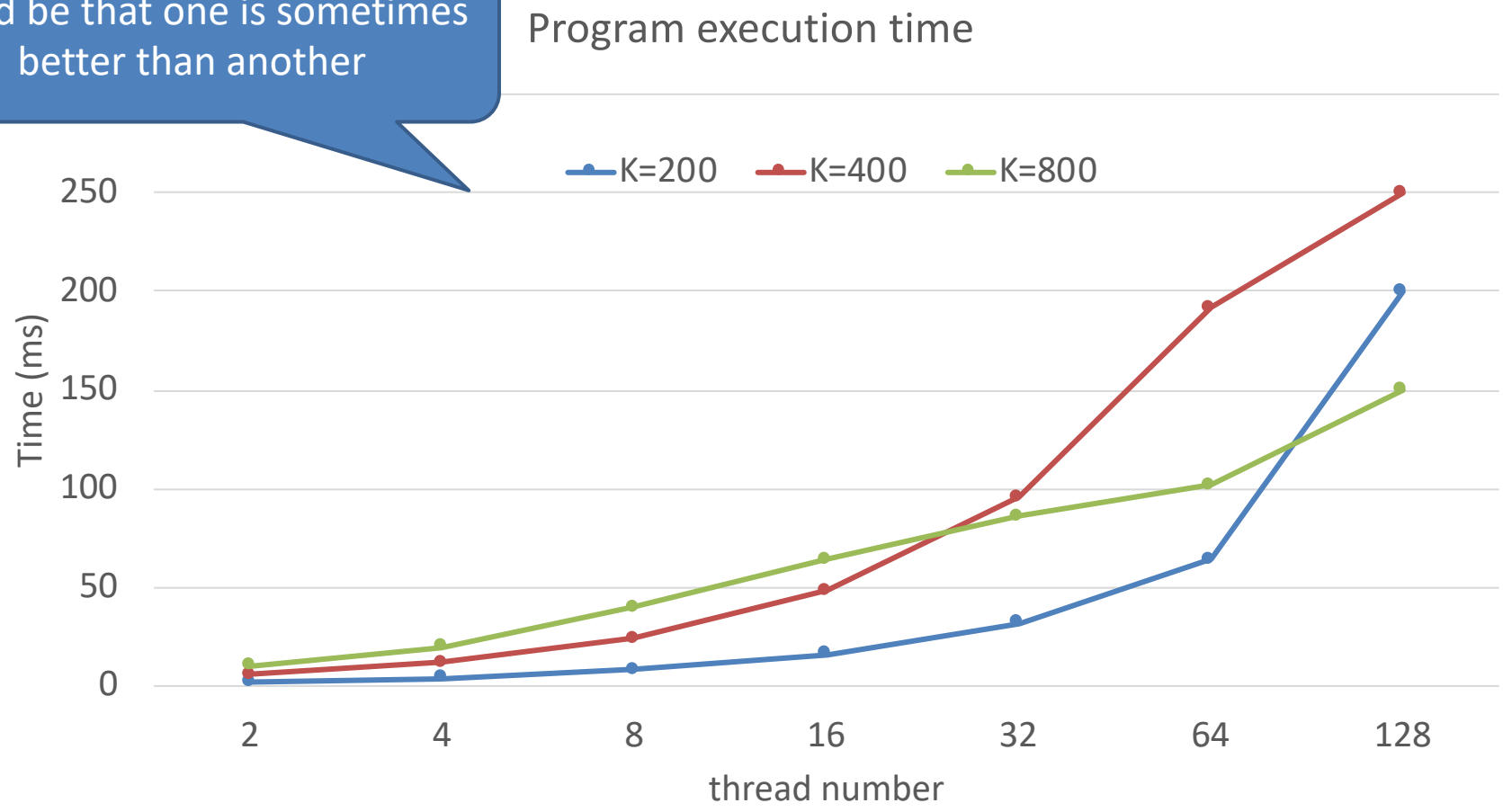
Test different K and thread num

Could be that one is always better than another



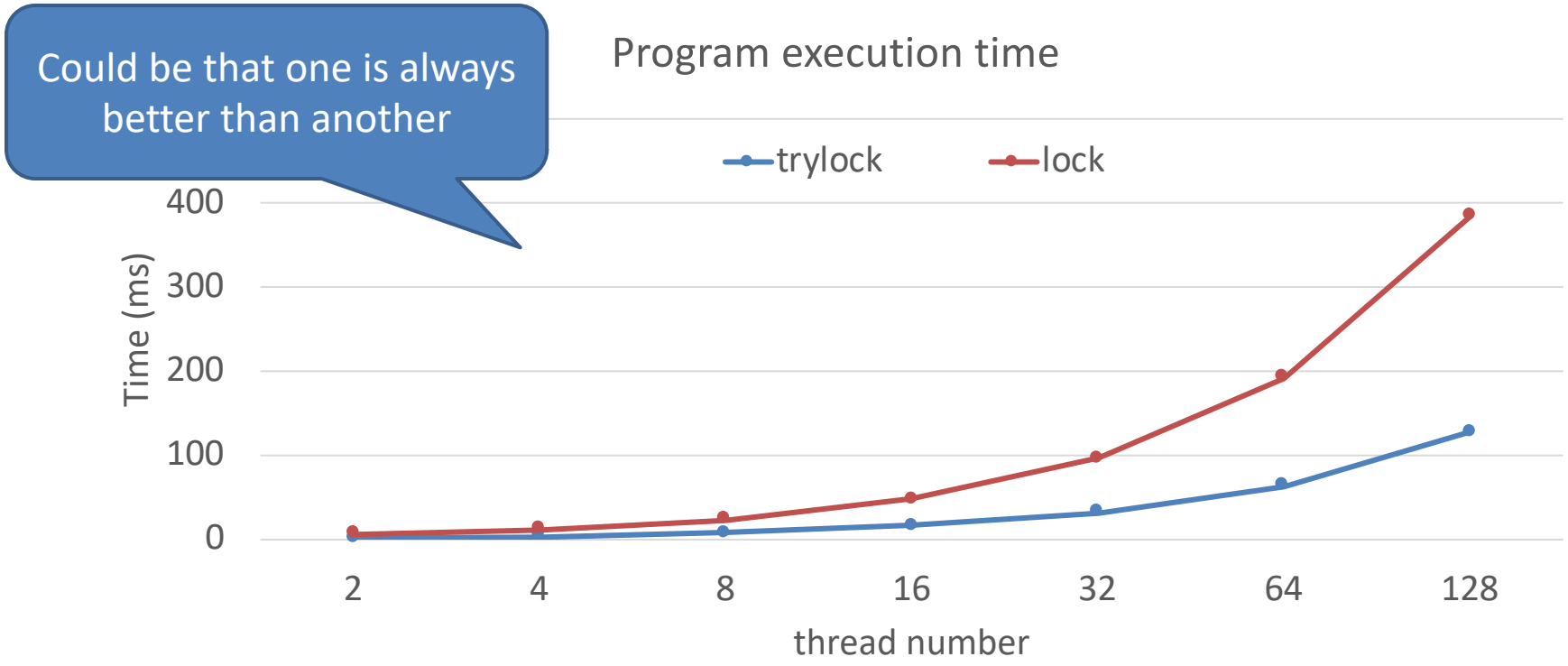
Test different K and thread num

Could be that one is sometimes better than another



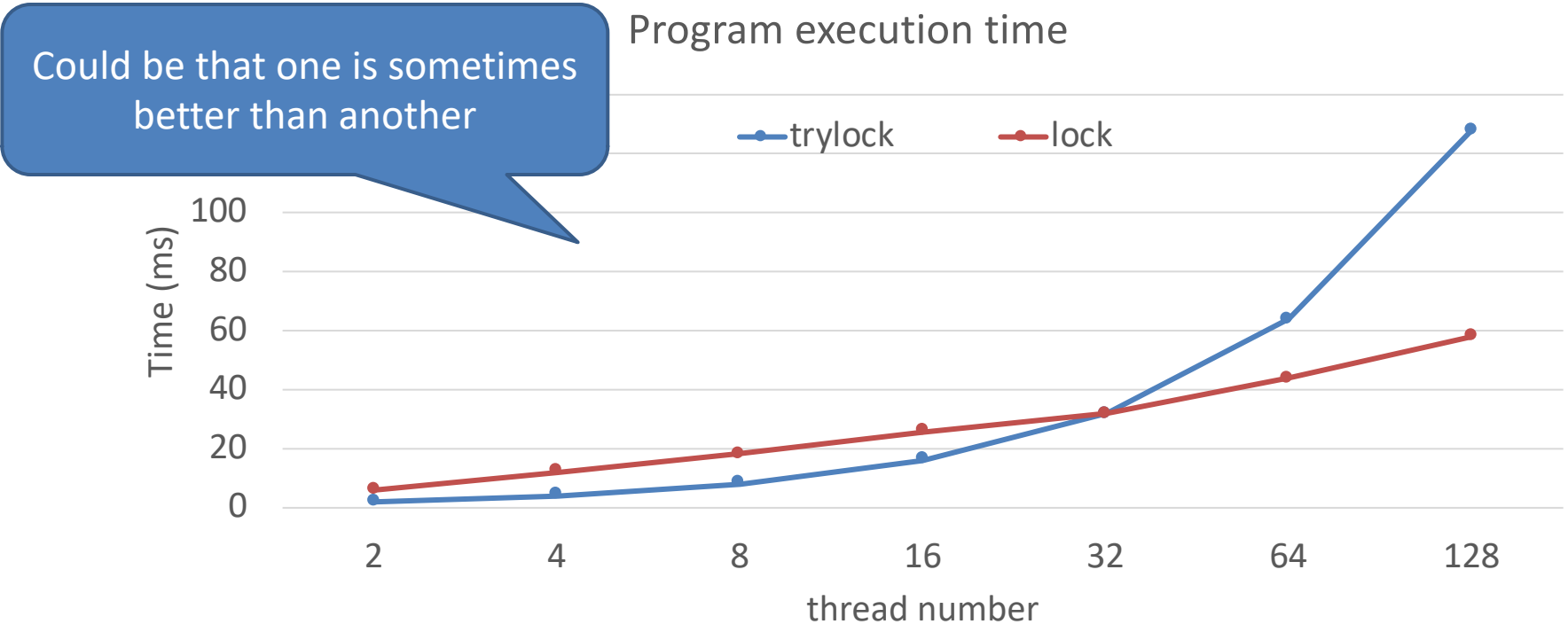
pthread_mutex_trylock vs. pthread_mutex_lock

- The original program uses pthread_mutex_trylock. Will the use of pthread_mutex_lock make a difference? Why?



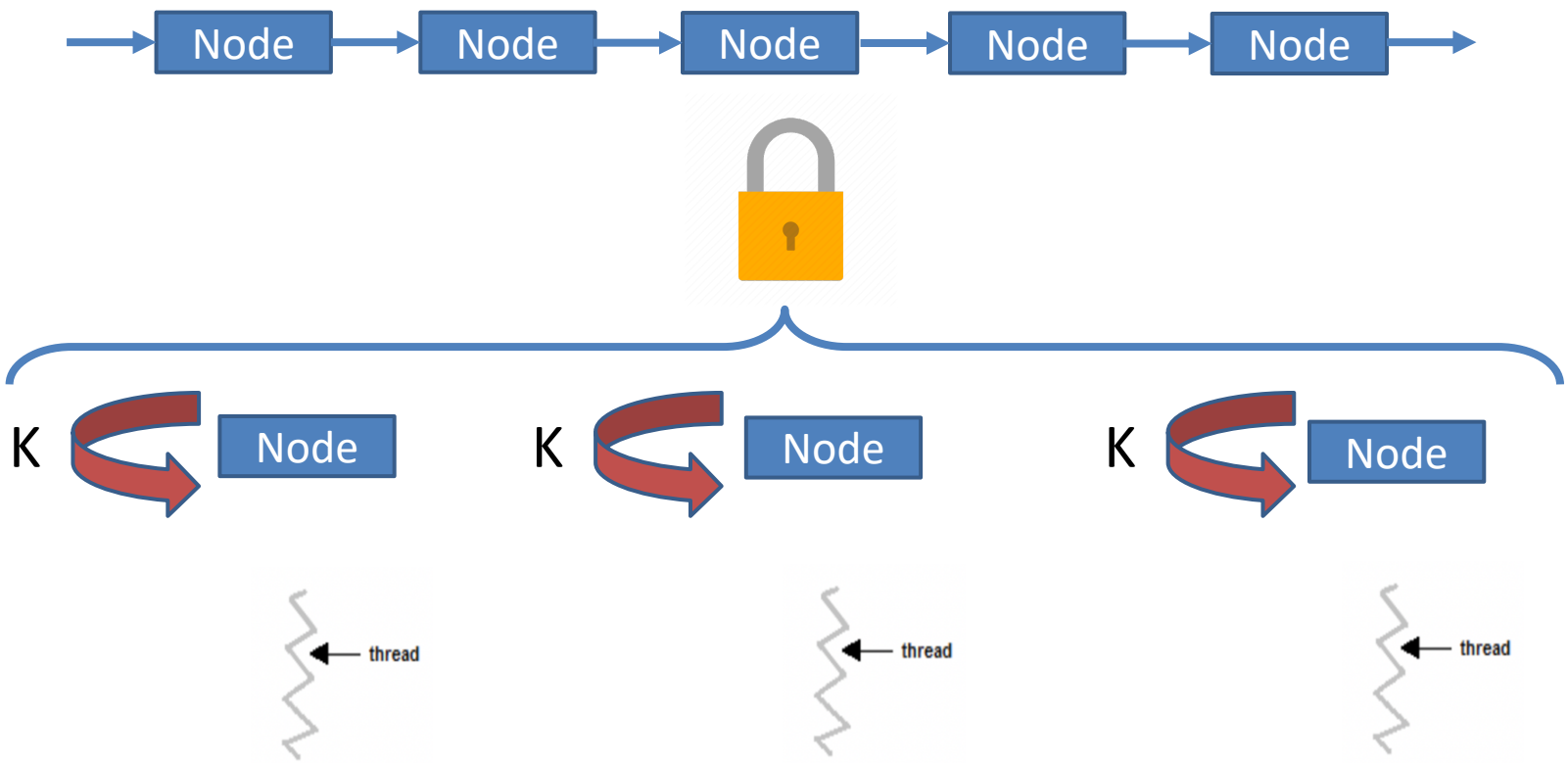
pthread_mutex_trylock vs. pthread_mutex_lock

- The original program uses pthread_mutex_trylock. Will the use of pthread_mutex_lock make a difference? Why?



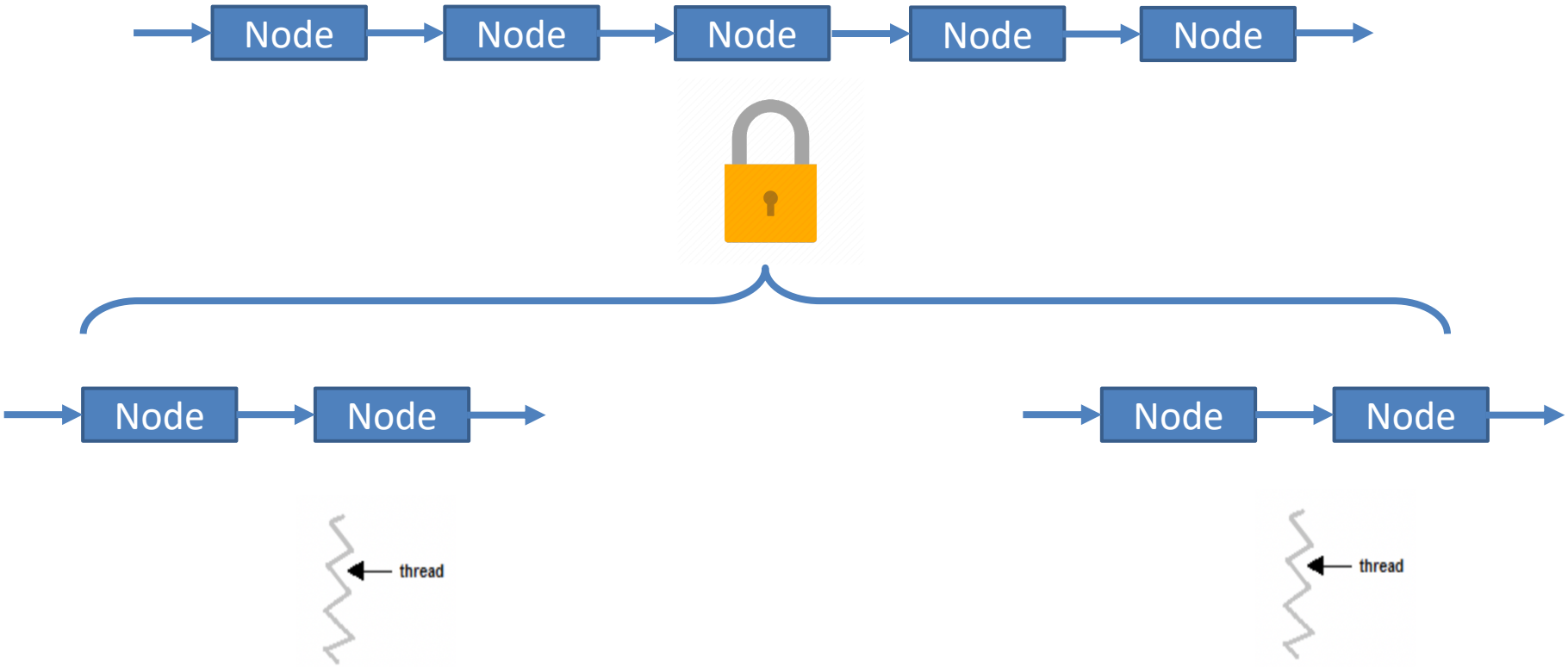
Assignment 2

Old design

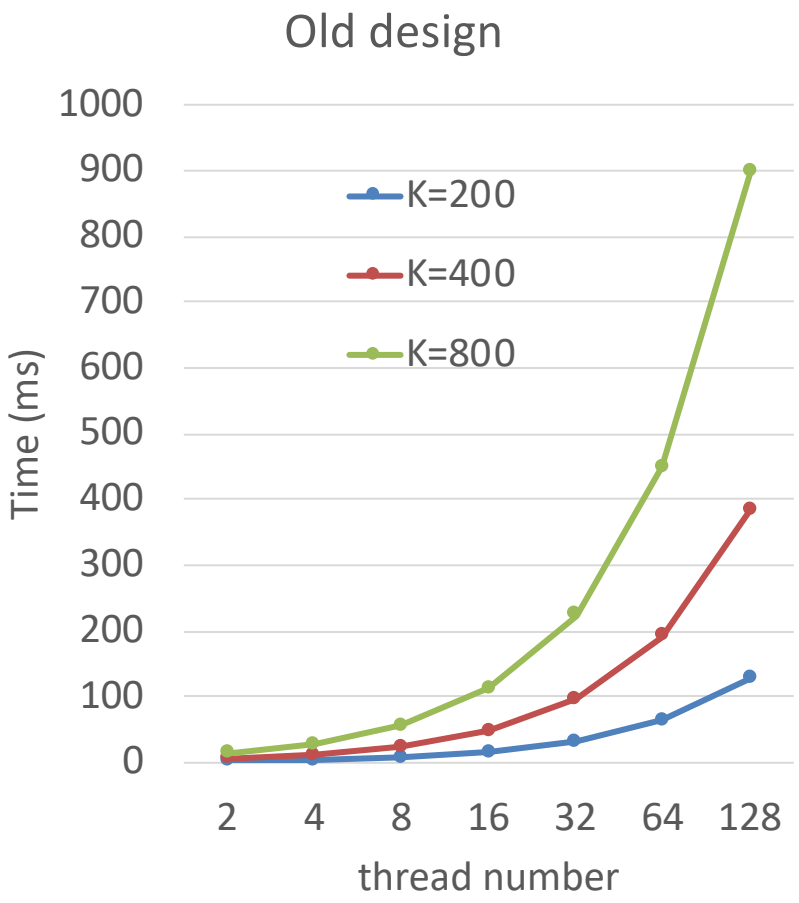


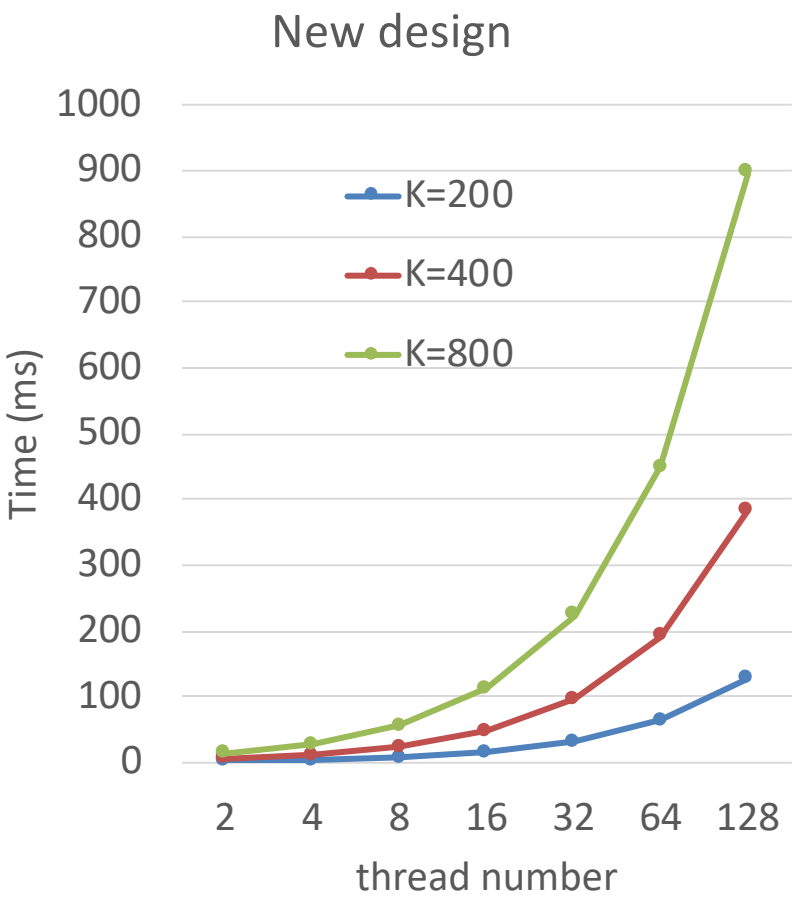
Assignment 2

New design



Performance difference and analysis





Submission

Submit your assignment zip file through D2L using the appropriate assignment link.

For assignment 1, please submit the source code;

For assignment 2, please submit a report with all the figures and analysis included.

CS3502_D2Lname.zip

Questions

- T/Th 2-3pm, J-318
- Skype ID: *suokun.nju*, share home screen, only voice, no video, no remote control and privacy issue. No time or location limitation.