

گزارش پروژه:

پیاده سازی کد به زبان سی

(توجه شود که تعداد thread ها و ورودی thread ها ثابت در نظر گرفته شده است.)

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <openssl/sha.h>
#include <math.h>

#define ITERATION_NUMBER (int)pow(2,20)
#define THREAD_NUMBER 10

typedef struct {
    int thread_id;
    char ***last_shals;
    const char *input;
} thread_data_t;

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
// pthread_cond_t condition = PTHREAD_COND_INITIALIZER;

void *thread_function(void *arg) {
    thread_data_t *data = (thread_data_t *)arg;
    int group_mate = data->thread_id ^ 1; // XOR with 1 for finding group
    mate
    for (int counter = 0; counter < ITERATION_NUMBER; counter++) {
        char data_str[(sizeof(data->thread_id)/sizeof(int)) +
        (SHA_DIGEST_LENGTH * 2 + 1) + (sizeof(counter)/sizeof(int)) + strlen(data->input) + 1];

        if (counter == 0) {
            snprintf(data_str, sizeof(data_str), "%d%d%s", data->thread_id,
            counter, data->input);
        }
        else if (strlen(data->last_shals[group_mate][counter - 1]) != 0) {
            pthread_mutex_lock(&mutex);
            snprintf(data_str, sizeof(data_str), "%d%s%d%s", data->thread_id,
            data->last_shals[group_mate][counter - 1], counter, data->input);
            pthread_mutex_unlock(&mutex);
        }
        else{
            //thread_cond_wait(&condition, &mutex);
            do {
                //printf(".");
            } while(strlen(data->last_shals[group_mate][counter - 1]) == 0);
            pthread_mutex_lock(&mutex);
            snprintf(data_str, sizeof(data_str), "%d%s%d%s", data->thread_id,
            data->last_shals[group_mate][counter - 1], counter, data->input);
            pthread_mutex_unlock(&mutex);
        }

        unsigned char sha1[SHA_DIGEST_LENGTH + 1];
        SHA1((unsigned char *)data_str, strlen(data_str), sha1);

        char hex_sha1[SHA_DIGEST_LENGTH * 2 + 1];

```

```

        for (int i = 0; i < SHA_DIGEST_LENGTH; i++) {
            snprintf(hex_shal + i * 2, 3, "%02x", shal[i]);
        }

        strcpy(data->last_shals[data->thread_id][counter], hex_shal);
    }
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
}

char *new_hash_function(char *input, int k) {
    pthread_t threads[k];
    char*** last_shals = (char***)calloc(k, sizeof(char**));

    for (int i = 0; i < k; i++) {
        last_shals[i] = (char**)calloc(ITERATION_NUMBER, sizeof(char*));
        for (int j = 0; j < ITERATION_NUMBER; j++) {
            last_shals[i][j] = (char *)calloc((SHA_DIGEST_LENGTH * 2 +
1), sizeof(char));
        }
    }

    thread_data_t data[k];
    for (int i = 0; i < k; i++) {
        data[i].thread_id = i;
        data[i].last_shals = last_shals;
        data[i].input = input;
        pthread_create(&threads[i], NULL, thread_function, &data[i]);
    }

    for (int i = 0; i < k; i++) {
        pthread_join(threads[i], NULL);
    }

    char final_hash[k * SHA_DIGEST_LENGTH * 2 + 1];
    memset(final_hash, 0, sizeof(final_hash));
    for (int i = 0; i < k; i++) {
        strcat(final_hash, last_shals[i][ITERATION_NUMBER - 1]);
    }

    unsigned char overall_shal[SHA_DIGEST_LENGTH + 1];
    SHA1((unsigned char *)final_hash, strlen(final_hash), overall_shal);

    char hex_overall_shal[SHA_DIGEST_LENGTH * 2 + 1];
    for (int i = 0; i < SHA_DIGEST_LENGTH; i++) {
        snprintf(hex_overall_shal + i * 2, 3, "%02x", overall_shal[i]);
    }

    for (int i = 0; i < k; i++) {
        for (int j = 0; j < ITERATION_NUMBER; j++) {
            if (NULL != last_shals[i][j]){
                free(last_shals[i][j]);
            }
        }
        free(last_shals[i]);
    }
    free(last_shals);

    return strdup(hex_overall_shal);
}

int main() {
    double start_time = clock();

```

```

char *hash_result = new_hash_function("meysam_khazae", THREAD_NUMBER);

double end_time = clock();
double execution_time = (end_time - start_time) / CLOCKS_PER_SEC;

printf("Hash Value: %s\n", hash_result);
printf("Execution Time: %.18f seconds\n", execution_time);

free(hash_result);
return 0;
}

```

خروجی اجرای کد در زبان سی

The screenshot shows a code editor with two tabs: 'main.c' and 'main.py'. The 'main.c' tab is active, displaying a C program that uses threads to calculate a hash value. The program includes headers for stdio, stdlib, string, pthread, openssl/sha, and math. It defines a macro for the iteration number as 2^{20} and sets the thread number to 10. The thread function calculates a SHA1 hash for each iteration. The terminal output shows the hash value and the execution time.

```

C main.c M X main.py
C main.c > _unnamed_struct_0401_1 > thread_id
You, 16 minutes ago | 2 authors (Meysam Khazae Sabor and others)
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <pthread.h>
5 #include <openssl/sha.h>
6 #include <math.h>
7
8 #define ITERATION_NUMBER (int)pow(2,20)
9 #define THREAD_NUMBER 10
10
11 You, 6 hours ago | 2 authors (Meysam Khazae Sabor and others)
12 typedef struct {
13     int thread_id;
14     char ***last_shas;
15     const char *input;
16 } thread_data_t;
17
18 PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
● khazae@home:~/Documents/sample/multithread/build$ ./home/khazae/Documents/s
Hash Value: 4682cc6ad2006316f2f7d3c62e5910d59b67c3d9
Execution Time: 185.370055000000007794 seconds

```

پیاده سازی کد به زبان پایتون

(توجه شود که تعداد thread ها و ورودی thread ها ثابت در نظر گرفته شده است.)

```

import threading
import hashlib
import time

iteration_number = 2**20
thread_number = 10

def new_hash_function(input, k):
    threads = []
    last_shas = ["" for element in range(iteration_number)]
    for element in range(k):
        mutex = threading.Lock()

```

```

def thread_function(thread_id):
    mutex.acquire()
    nonlocal last_shals # Access last_shals in the enclosing scope
    group_mate = thread_id ^ 1 # XOR with 1 to get the group mate
    for counter in range(itratetion_number):
        if counter == 0 :
            data = f"{thread_id}{counter}{input}"
        else:
            while(last_shals[group_mate][counter-1] == ""):
                pass
            data = f"{thread_id}{last_shals[group_mate][counter-1]}{counter}{input}"
        sha1 = hashlib.shal(data.encode()).hexdigest()
        last_shals[thread_id][counter] = sha1
    mutex.release()

# Create threads and start their execution
for i in range(k):
    thread = threading.Thread(target=thread_function, args=(i,))
    threads.append(thread)
    thread.start()

# Wait for all threads to finish
for thread in threads:
    thread.join()

# Collect final SHA1s from each thread
final_hash = []
for row in last_shals:
    # Append the last element to the row (modifies original matrix)
    final_hash.append(row[-1])
final_shals = "".join(final_hash)

# Calculate the overall SHA1
overall_shal = hashlib.shal(final_shals.encode()).hexdigest()

return overall_shal

def main():
    start_time = time.time()
    hash_result = new_hash_function("meysam_khazaei", thread_number)
    end_time = time.time()
    execution_time = end_time - start_time
    print("Hash Value:", hash_result)
    print("Execution Time:", execution_time, "seconds")

if __name__ == "__main__":
    main()

```

خروجی اجرای کد در زبان پایتون

```

C main.c M  main.py M X
main.py > new_hash_function > thread_function
import threading
5
6 itratetion_number = 2**20
7 thread_number = 10
8
9 def new_hash_function(input, k):
10     threads = []
11     last_shais = [""] for element in range(itratetion_number)
12     mutex = threading.Lock()
13
14     def thread_function(thread_id):
15         nonlocal last_shais # Access last_shais in the e
16         group_mate = thread_id ^ 1 # XOR with 1 to get t
17         for counter in range(itratetion_number):
18             print("counter = {}".format(counter))
19             if counter == 0 : # Meysam Khazaei Sabor,
20                 data = f"{thread_id}{counter}{input}"
21             elif len(last_shais[group_mate][counter - 1])
22                 mutex.acquire()

```

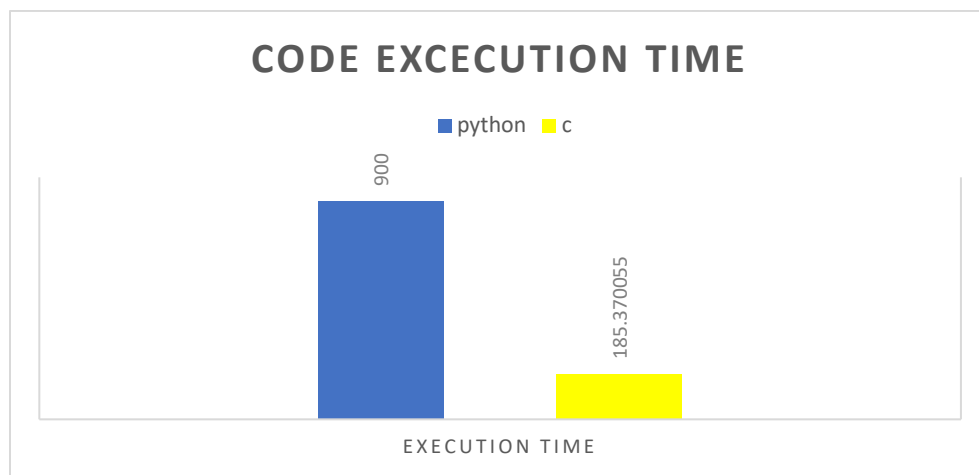
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```

counter = 979
counter = 980
counter = 969
counter = 1004
counter = 1003
counter = 1004
counter = 1005
counter = 981
counter = 968
counter = 969
counter = 980
counter = 981
counter = 982
counter = 979

```

مقایسه زمان اجرا



زمان اجرای کد پایتون خیلی زیاد بود و عدد تقریبی است. (برای زمان دقیق می بایست اسکریپت پایتونی مدت زمان زیادی اجرا شود که بنده به خاطر ضیق وقت صرف نظر کردم)

۳- آیا این برنامه را می توان به صورت **microservice** پیاده سازی کرد؟ چه مزایایی دارد؟ روی زمان اجرا چه

اثری دارد؟ لطفا پاسختان را به صورت تشریحی و به طور مفصل بیان کنید. در صورت تمایل خودتان، این

برنامه را به صورت **microservice** پیاده سازی و زمان اجرای آن را با حالت های قبلی مقایسه کنید. (پیاده سازی، امتیاز مثبت در نظر گرفته می شود)

کد پیاده سازی شده یک هش SHA-1 را محاسبه می کند و مستقیماً چندین ورودی مستقل را پردازش نمی کند، اگر کد را بازسازی کنیم تا به عنوان پایه ای برای یک میکروسرویس با تغییرات احتمالی می بایست کد را به گونه ای بازسازی کنیم که:

- ورودی های چندگانه: برای پردازش چندین ورودی مستقل، کد را طوری تنظیم کرد که آرایه ای از رشته ها را به عنوان ورودی بپذیرد یا به مشتریان اجازه دهید رشته های جداگانه را از طریق یک API ارسال کنند.
- هش(های) محاسبه شده را به عنوان ساختمان داده برگرداند.

>> پایان گزارش <<