The Big Dot

Coding:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

#define N 1024*1024
#define THREADS_PER_BLOCK 512

float gpuResult, cpuResult;

__global__ void dot_product(float *a, float *b, float *c, int n ){
   int index = threadIdx.x + blockIdx.x * blockDim.x;
   if (index < n)
      c[index] = a[index] * b[index];
}

void random_floats(float *x, int size)
{
   for (int i = 0; i < size; i++) {
      x[i] = (float)(rand()/(float)RAND_MAX);
   }
}

long long start_timer() {
   struct timeval tv;
   gettimeofday(&tv, NULL);
   return tv.tv_sec * 1000000 + tv.tv_usec;
}

long long stop_timer(long long start_time, char *name) {
```

```
    struct timeval tv;
    gettimeofday(&tv, NULL);
    long long end_time = tv.tv_sec * 1000000 + tv.tv_usec;
    printf("%s: %.5f sec\n", name, ((float)(end_time - start_time)) / (1000 * 1000) );
    return end_time - start_time;
}

float GPU_big_dot(float *a, float *b, int size){
    float *c; // host copies of c
    float *d_a, *d_b, *d_c; // device copies of a, b, c

    long long memoryTime;
    memoryTime = start_timer();
    // Allocate space for device copies of a, b, c
    cudaMalloc((void **) &d_a, size);
    cudaMalloc((void **) &d_b, size);
    cudaMalloc((void **) &d_c, size);
    // Allocate space for host copies of c
    c = (float *) malloc(size);
    // Copy inputs to device
    cudaMemcpy(d_a, a, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, size, cudaMemcpyHostToDevice);
    stop_timer(memoryTime, (char *) "Memory allocation and data transfer from
CPU to GPU time");

    long long kernelTime;
    kernelTime = start_timer();
    // Launch dot_product() kernel on GPU with N threads
    dot_product<<<(N + THREADS_PER_BLOCK - 1)/THREADS_PER_BLOCK,
THREADS_PER_BLOCK>>>(d_a, d_b, d_c, N);
    stop_timer(kernelTime, (char *) "Kernel execution time");

    long long TransferTime;
```

```
    TransferTime = start_timer();
    // Copy result back to host
    cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);
    stop_timer(TransferTime, (char *) "Data transfer from GPU to CPU time");

    for(int i = 0; i < N; i++){
        gpuResult += c[i];
    }
    printf("Gpu computing result is: %f\n", gpuResult);

    // Cleanup
    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    return gpuResult;
}

float CPU_big_dot(float *a, float *b, int n){
    for(int i = 0; i < n; i++){
        cpuResult += a[i] * b[i];
    }
    printf("Cpu computing result is: %f\n", cpuResult);

    return cpuResult;
}

int main(void) {
    float *a, *b; // host copies of a, b, c
    int size = N * sizeof(float);

    // Alloc space for host copies of a, b and setup input values
    srand ((unsigned) time (NULL));
```

```
    a = (float *) malloc(size);
    random_floats(a, N);
    b = (float *) malloc(size);
    random_floats(b, N);

    long long startTime,gpuTime,cpuTime;

    startTime = start_timer();
    gpuResult = GPU_big_dot(a,b,size);
    gpuTime = stop_timer(startTime, (char *) "GPU Computing time");

    startTime = start_timer();
    cpuResult = CPU_big_dot(a,b,N);
    cpuTime = stop_timer(startTime, (char *) "CPU Computing time");

    float speedUp = (float) cpuTime / gpuTime;
    printf("The speedup is: %f\n", speedUp);

    if (gpuResult - cpuResult < 1.0e-6) {
        printf("Two results are same!\n");
    }else {
        printf("Two results are not same!\n");
    }

    return 0;
}
```

Result:

```
[biyangf@node1685 ~]$ ./a.out
Memory allocation and data transfer from CPU to GPU time: 0.09515 sec
Kernel execution time: 0.00035 sec
Data transfer from GPU to CPU time: 0.00265 sec
Gpu computing result is: 261818.218750
GPU Computing time: 0.10294 sec
Cpu computing result is: 261818.218750
CPU Computing time: 0.00334 sec
The speedup is: 0.032446
Two results are same!
```

According to the results, we can see that the calculation time of the CPU is shorter than that of the GPU. This is because the GPU consumes too much time in memory allocation and data transfer, resulting in a significant increase in overall GPU computing time. Essentially, the computing time of the GPU is the execution time of its core. We can see that the execution time of the kernel is much shorter than the calculation time of the CPU.