NCHC Open Hackathons 2024

Day Final

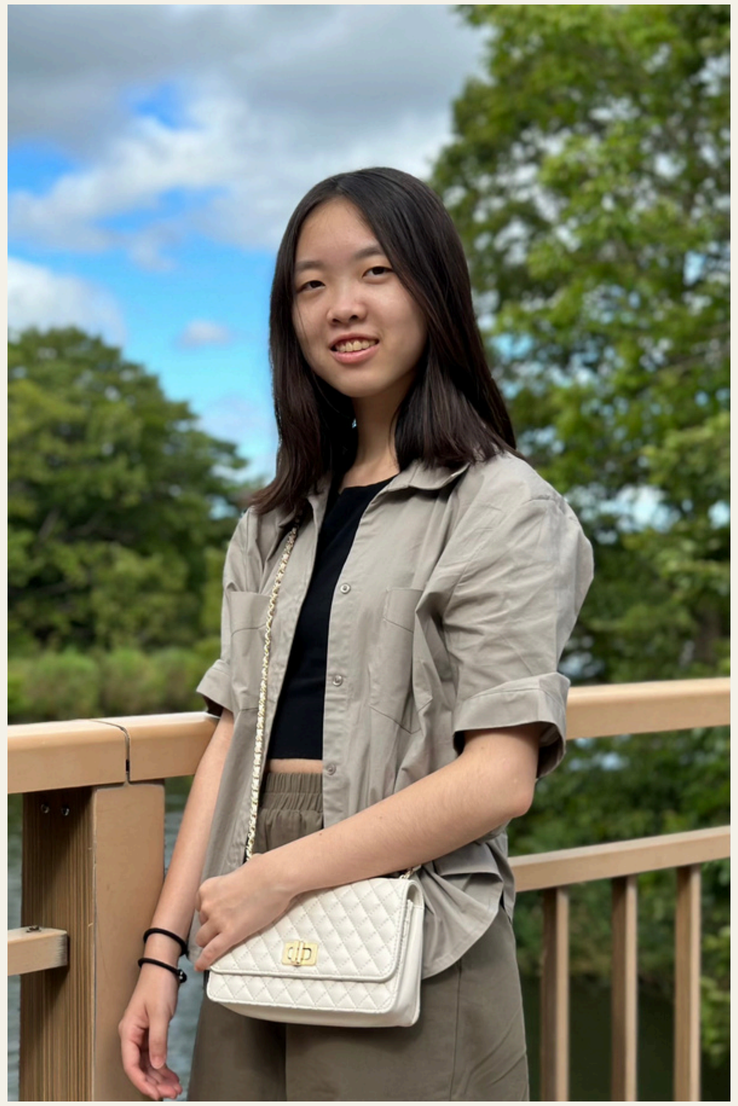
Team5-parallel-minds,NTHU

team-11-parallel-minds

程詩柔

「如果有哆啦A夢在就好了」今天也這麼想著。
metimes wish that Doraemon was with me.

謝之豫



熊恩伶

OpenACC
More Science, Less Programming
OPEN HACKATHONS
NCHC
NARLabs 財團法人國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing
NVIDIA.

Firefly algorithm (visualization)

1. 設定螢火蟲總數(population),空間維度(dimen),訓練次數(max_iter)
2. 將所有螢火蟲隨機放在空間中並隨機給定亮度(fitness)
3. 每個iteration，每隻螢火蟲都要朝附近最亮的螢火蟲前進(位置更新)
4. 更新所有螢火蟲的亮度(fitness更新)

迴圈時間複雜度 max_iter * population * dimen * population

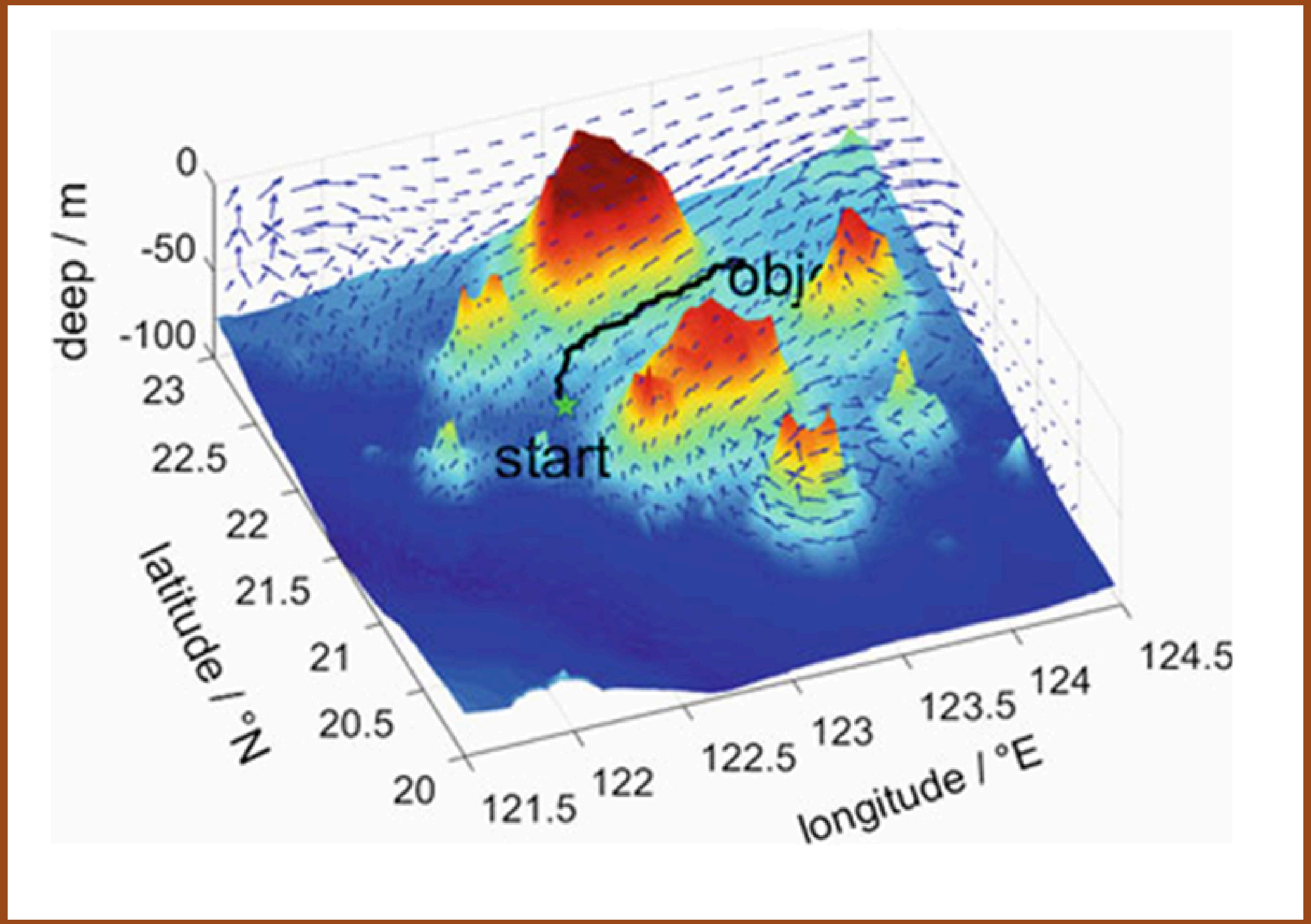
why is it important? Robot path planning

Firefly algorithm

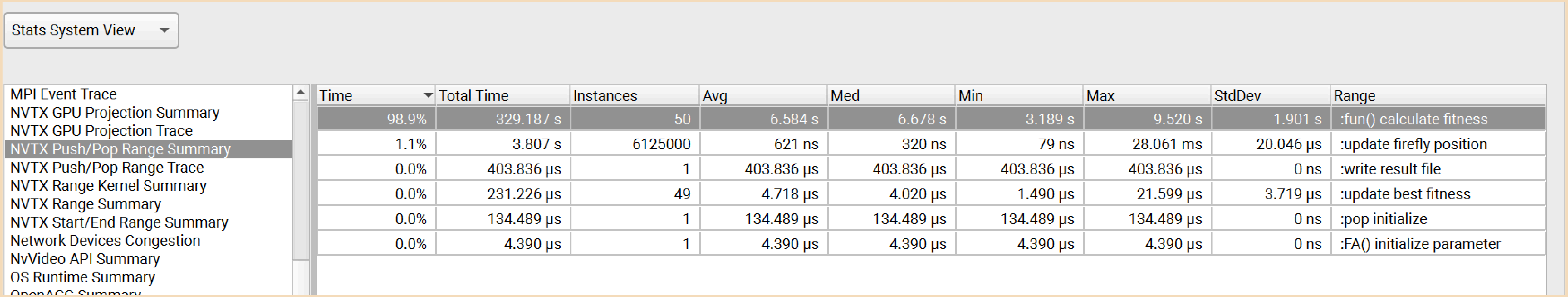
- textbook: Nature-Inspired Computation in Navigation and Routing Problems Algorithms, Methods and Applications Editors: Xin-She Yang, Yu-Xin Zhao
- Vehicle path planning : Particle swarm optimization algorithm vs Firefly algorithm
- 準確率 : Firefly algorithm > Particle swarm optimization algorithm
- 執行時間: Firefly algorithm > Particle swarm optimization algorithm

Goals :

- *accelerate firefly algorithm with cuda*



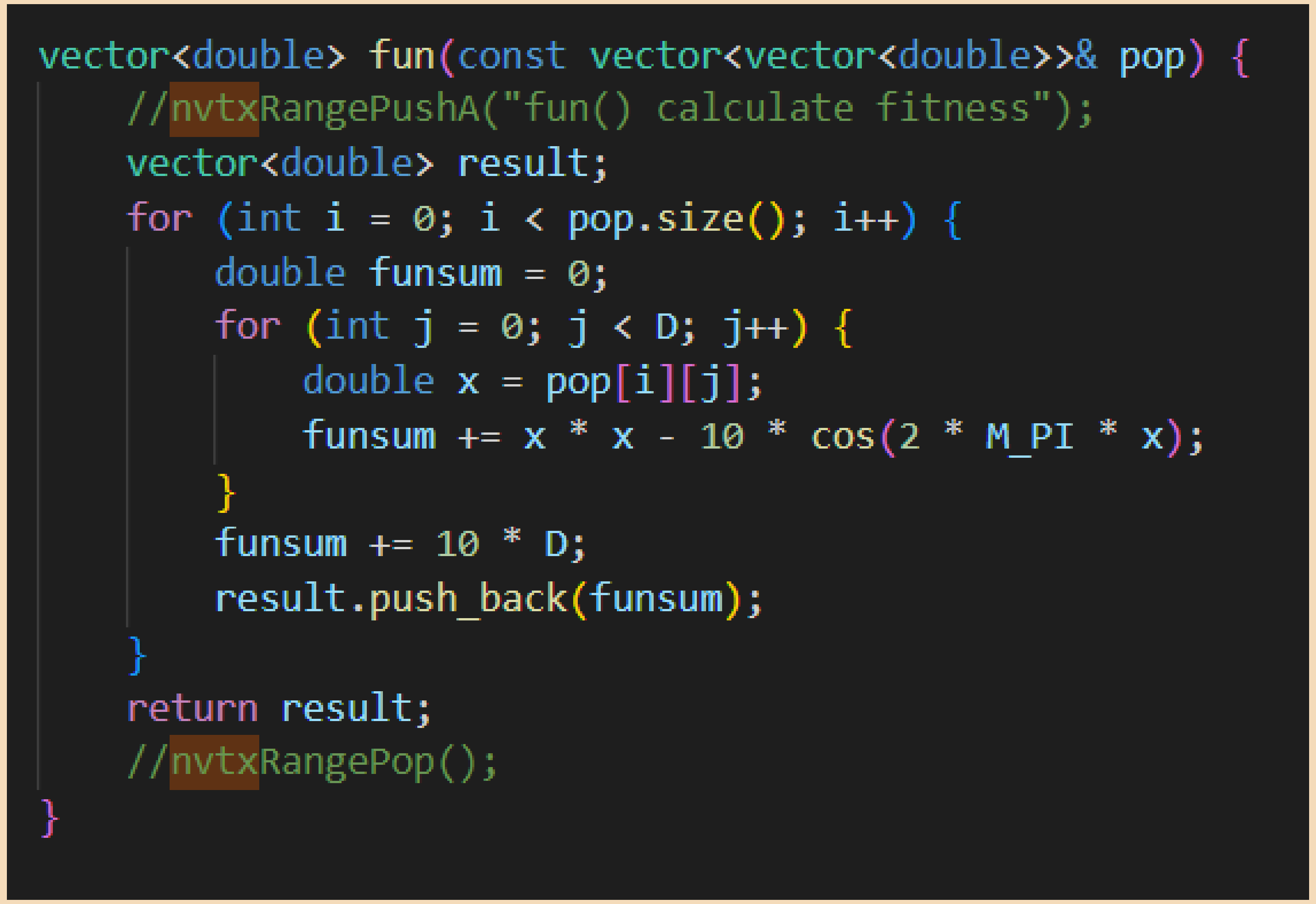
Progress -Week1



Stats System View

- MPI Event Trace
- NVTX GPU Projection Summary
- NVTX GPU Projection Trace
- NVTX Push/Pop Range Summary
- NVTX Push/Pop Range Trace
- NVTX Range Kernel Summary
- NVTX Range Summary
- NVTX Start/End Range Summary
- Network Devices Congestion
- NvVideo API Summary
- OS Runtime Summary
- OpenACC Summary

Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Range
98.9%	329.187 s	50	6.584 s	6.678 s	3.189 s	9.520 s	1.901 s	:fun() calculate fitness
1.1%	3.807 s	6125000	621 ns	320 ns	79 ns	28.061 ms	20.046 μs	:update firefly position
0.0%	403.836 μs	1	403.836 μs	403.836 μs	403.836 μs	403.836 μs	0 ns	:write result file
0.0%	231.226 μs	49	4.718 μs	4.020 μs	1.490 μs	21.599 μs	3.719 μs	:update best fitness
0.0%	134.489 μs	1	134.489 μs	134.489 μs	134.489 μs	134.489 μs	0 ns	:pop initialize
0.0%	4.390 μs	1	4.390 μs	4.390 μs	4.390 μs	4.390 μs	0 ns	:FA() initialize parameter



```
vector<double> fun(const vector<vector<double>>& pop) {
    //nvtxRangePushA("fun() calculate fitness");
    vector<double> result;
    for (int i = 0; i < pop.size(); i++) {
        double funsum = 0;
        for (int j = 0; j < D; j++) {
            double x = pop[i][j];
            funsum += x * x - 10 * cos(2 * M_PI * x);
        }
        funsum += 10 * D;
        result.push_back(funsum);
    }
    return result;
    //nvtxRangePop();
}
```

bottleneck:

fitness亮度更新計算更新耗時最久

strategy:

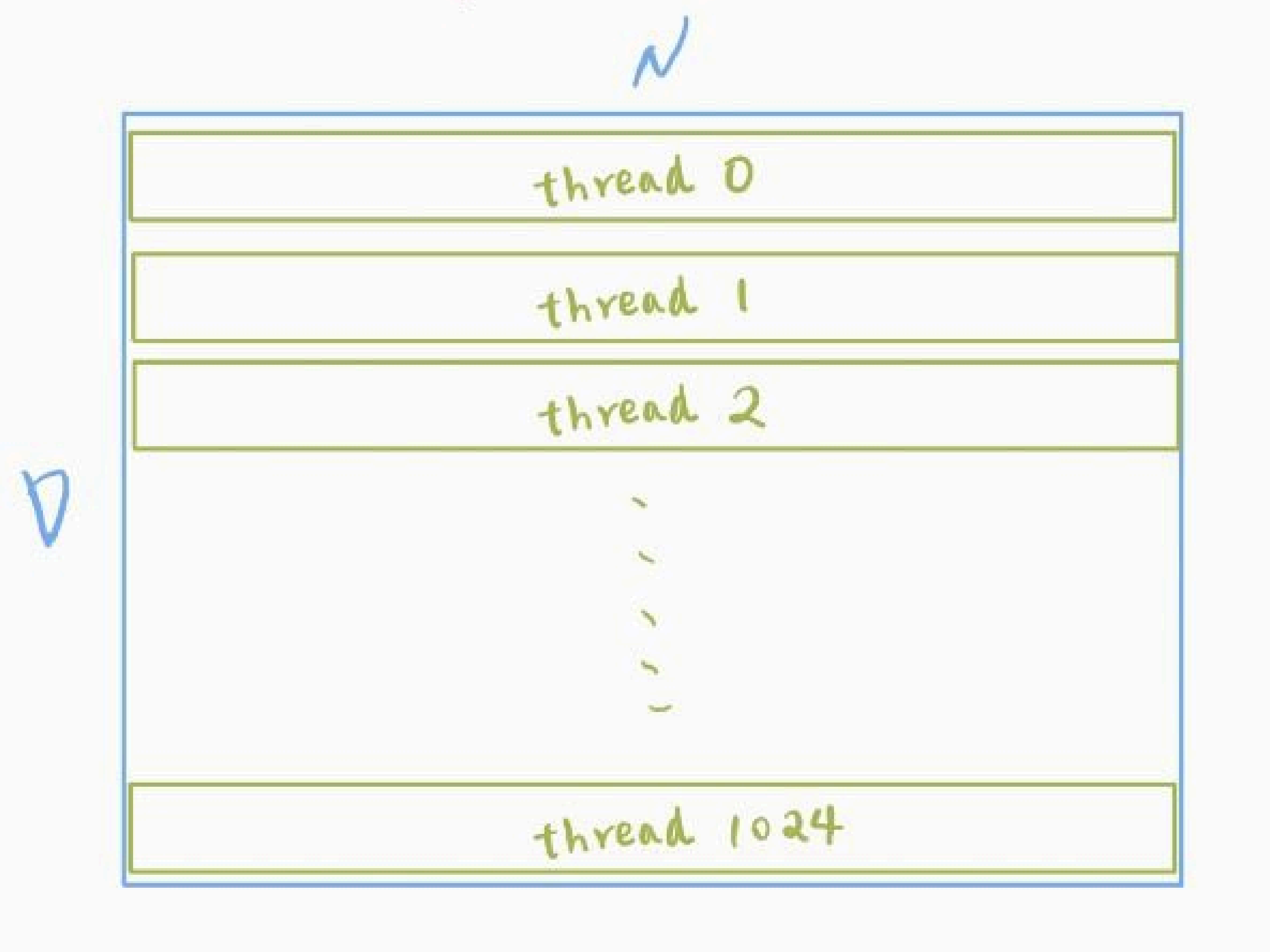
將fun()寫成kernel放上gpu做運算

Progress -Week1

將fun_kernel用1024 thread 平行運算

Problem : 反而比cpu code 還慢!!

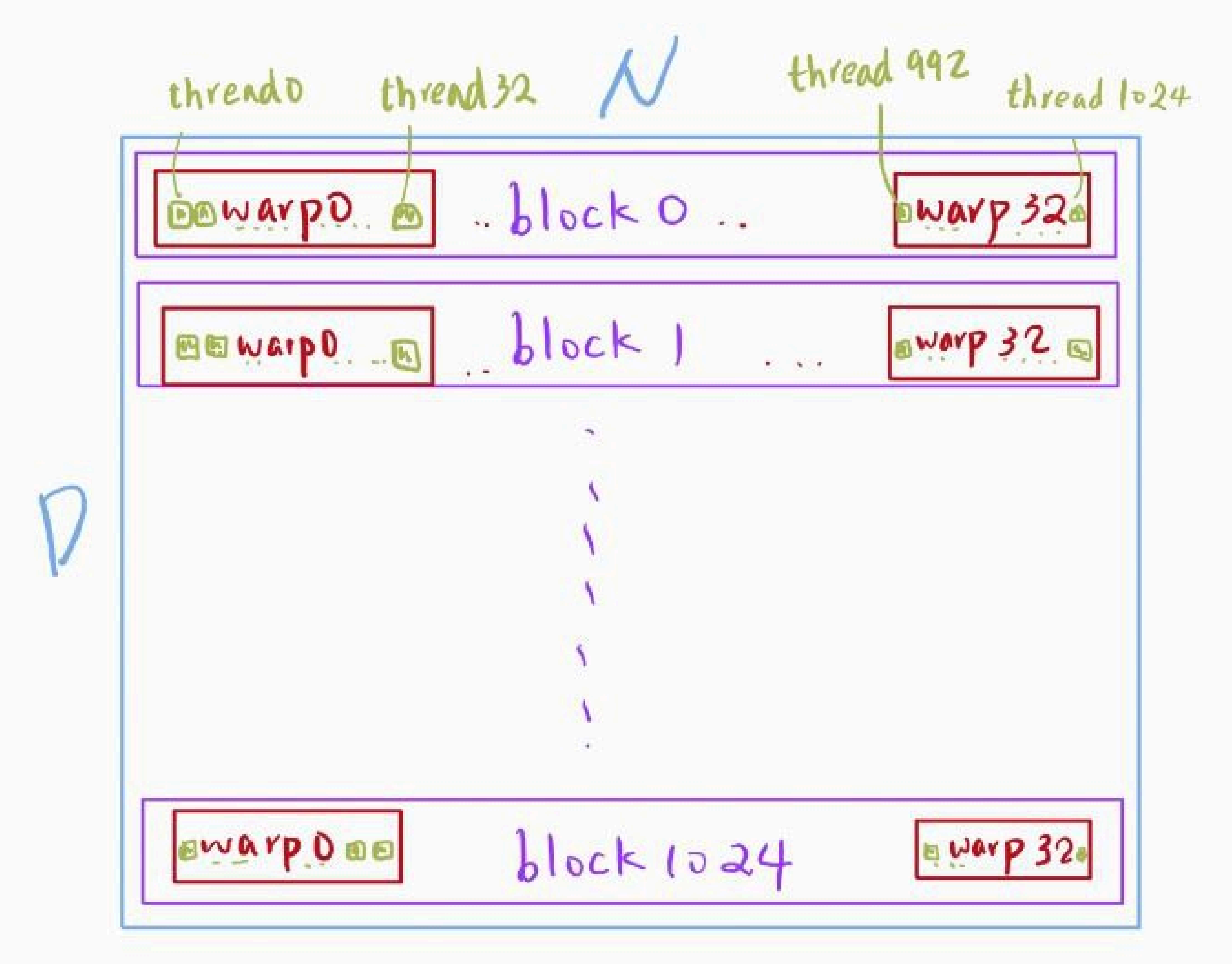
Reason: 平行度太低+頻繁的Memcpy很耗時



low parallelization

Progress -Week2

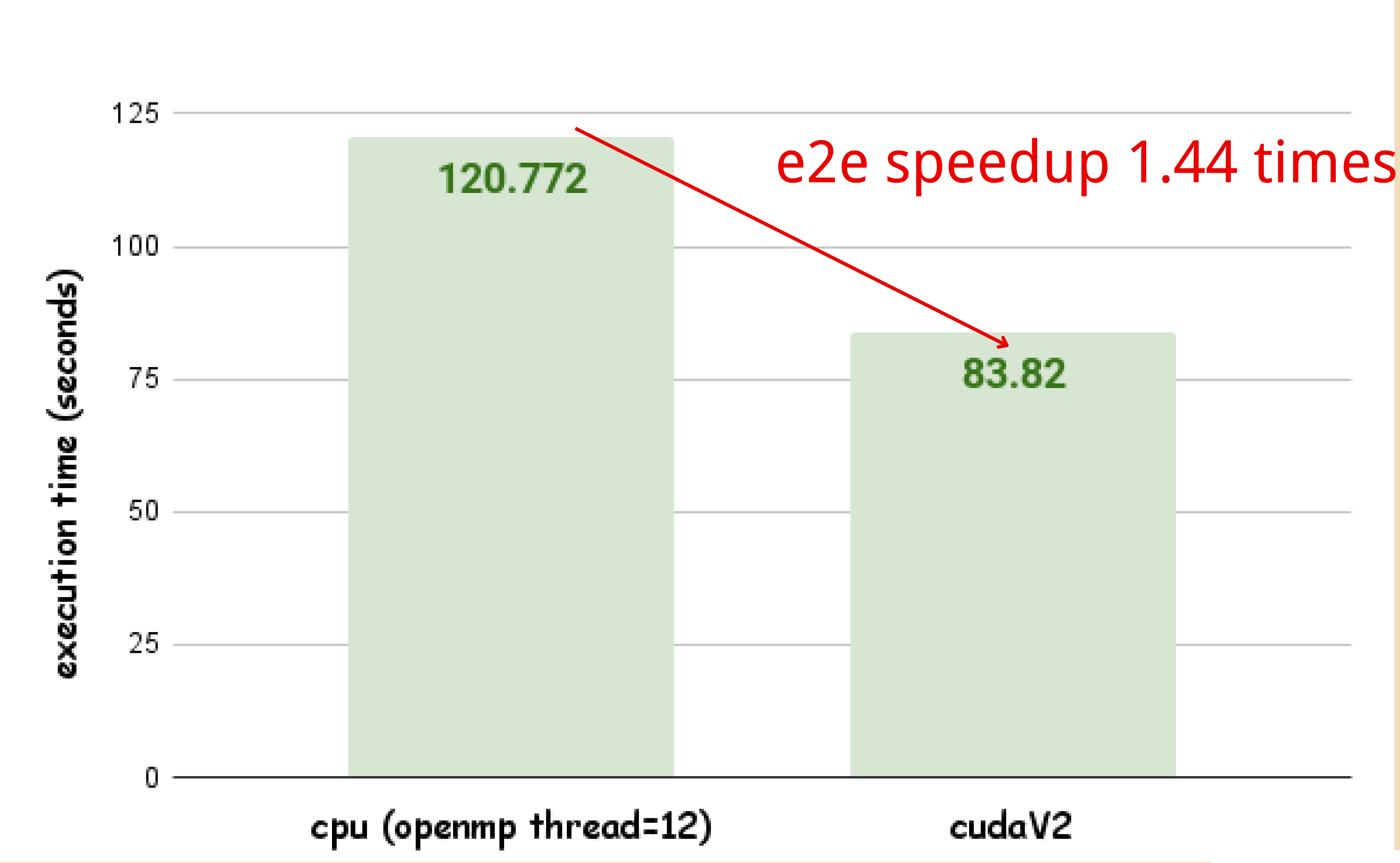
high parallelization



- 使用1024blocks中的32個wraps
- 每個wrap中的32thread
- WarpReduce

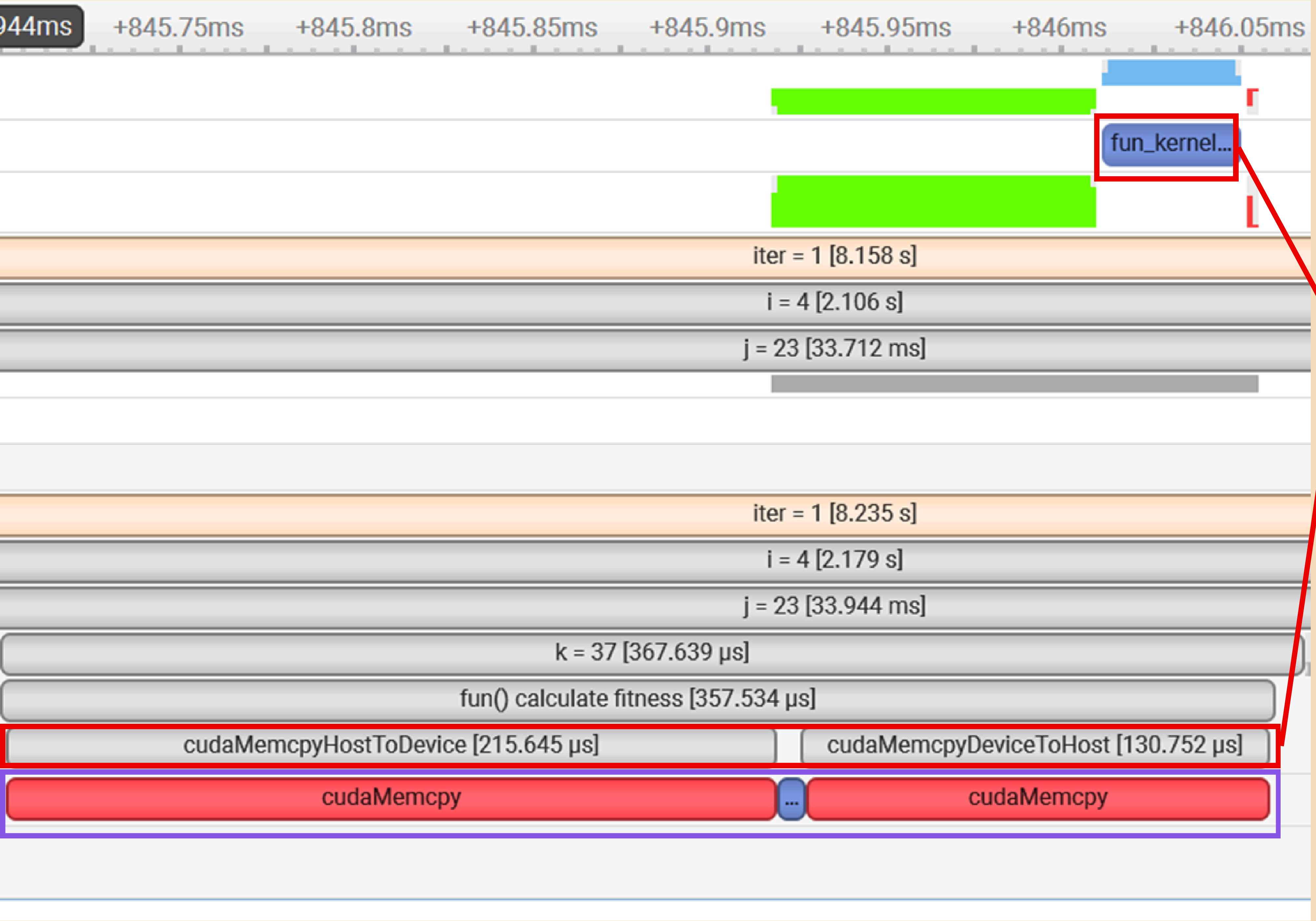
Progress -Week2

problem size population:1024 dimen:512 max_iter 3



e2e speedup 1.44 times

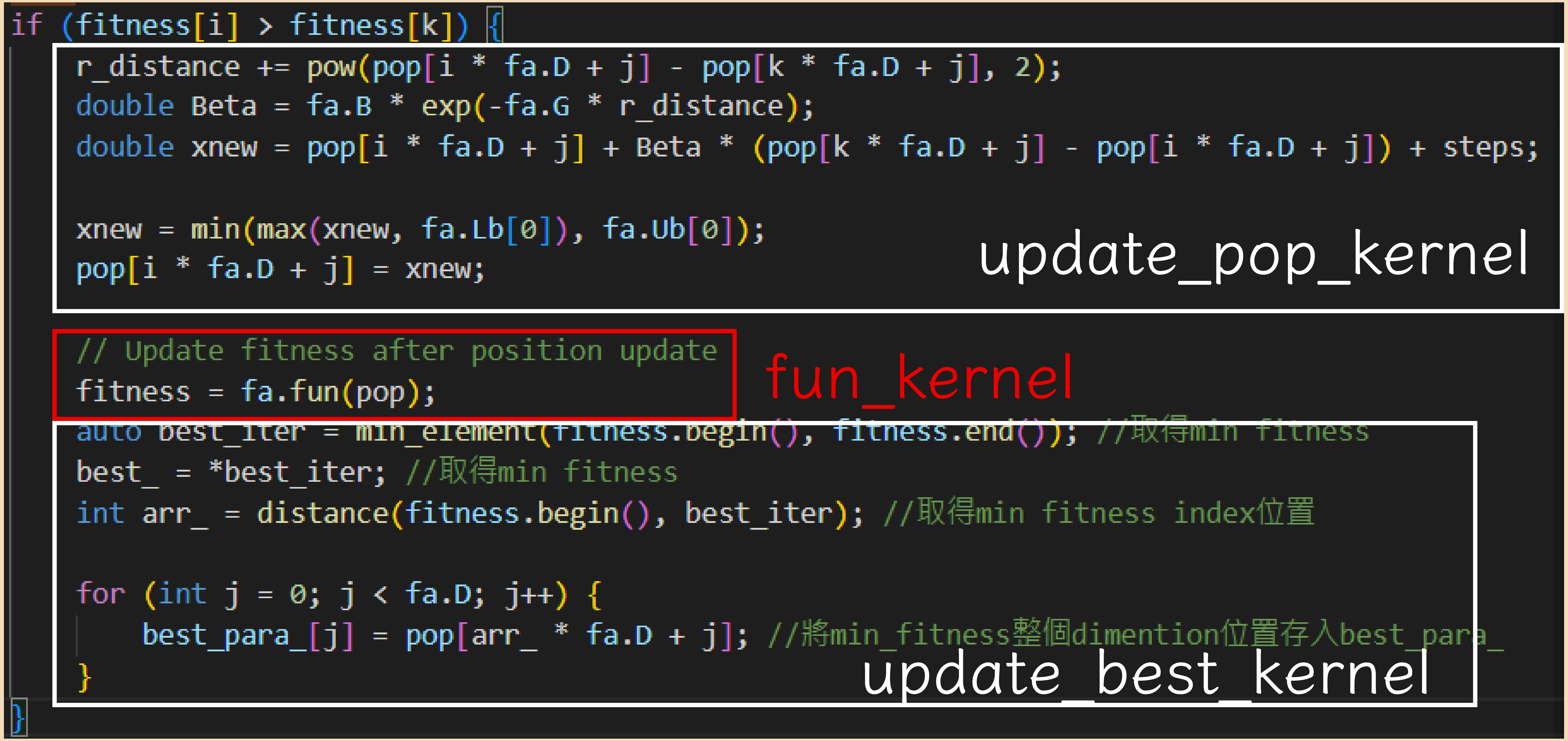
Progress -Week2



bottleneck:
kernel計算時間佔10%
剩餘90%均為Memcpy的時間

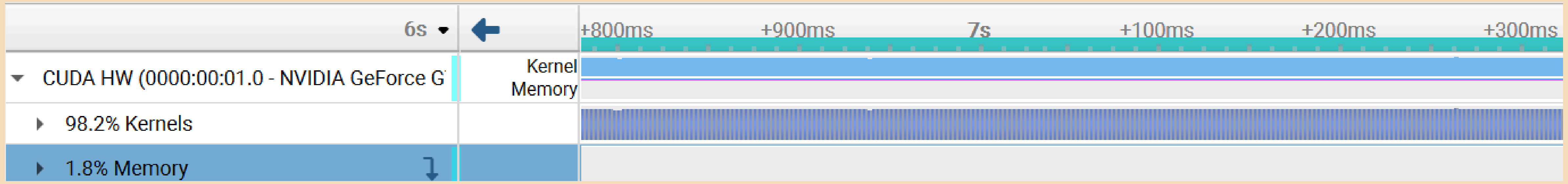
strategy:
透過將fun_kernel計算前後也放上gpu做運算節省Memcpy時間

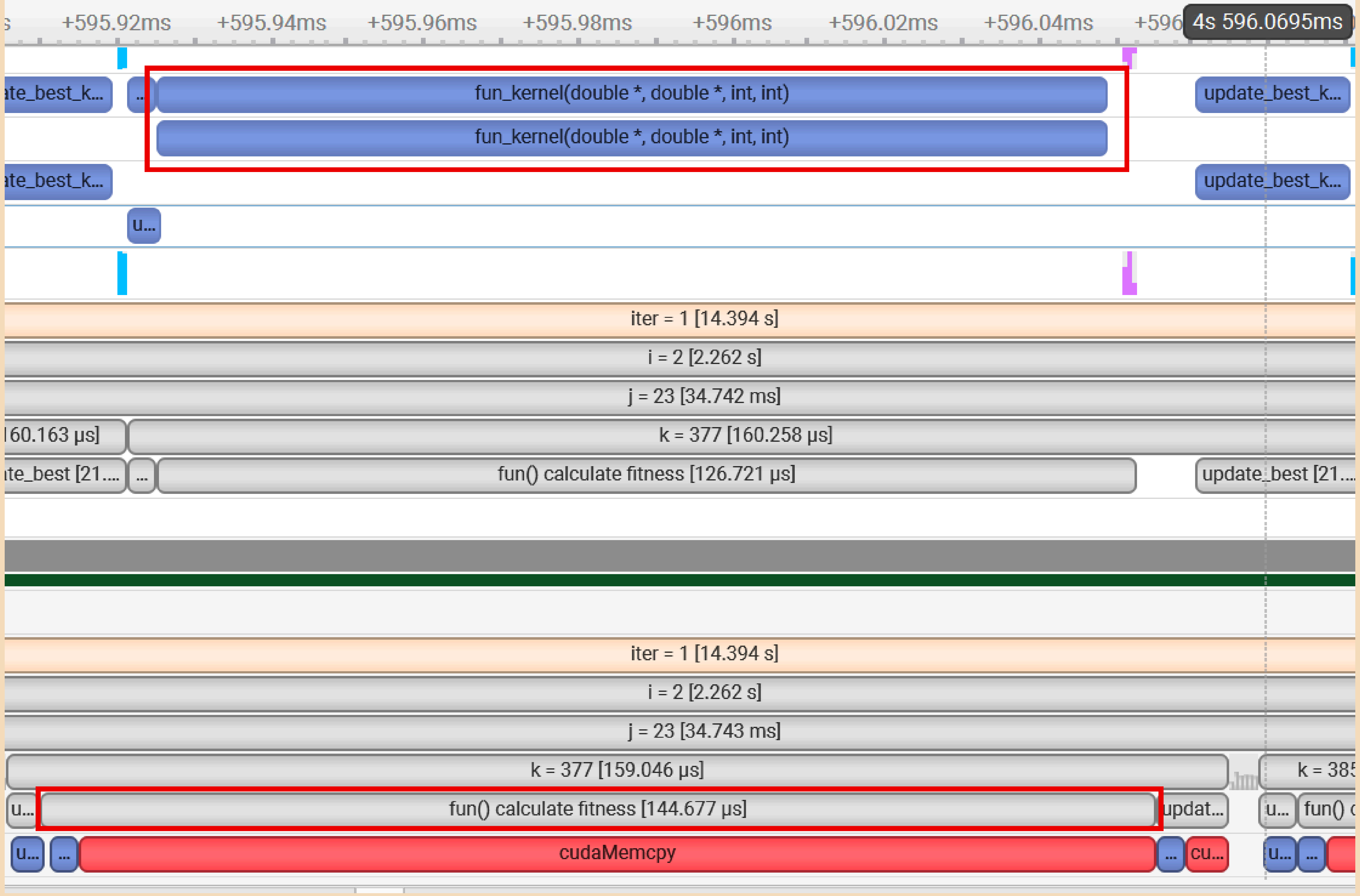
Progress -Week3



```
if (fitness[i] > fitness[k]) {
    r_distance += pow(pop[i * fa.D + j] - pop[k * fa.D + j], 2);
    double Beta = fa.B * exp(-fa.G * r_distance);
    double xnew = pop[i * fa.D + j] + Beta * (pop[k * fa.D + j] - pop[i * fa.D + j]) + steps;

    xnew = min(max(xnew, fa.Lb[0]), fa.Ub[0]);
    pop[i * fa.D + j] = xnew;

    // Update fitness after position update
    fitness = fa.fun(pop);
    auto best_iter = min_element(fitness.begin(), fitness.end()); //取得min fitness
    best_ = *best_iter; //取得min fitness
    int arr_ = distance(fitness.begin(), best_iter); //取得min fitness index位置

    for (int j = 0; j < fa.D; j++) {
        best_para_[j] = pop[arr_ * fa.D + j]; //將min_fitness整個dimention位置存入best_para_
    }
}
```

Progress -Week3

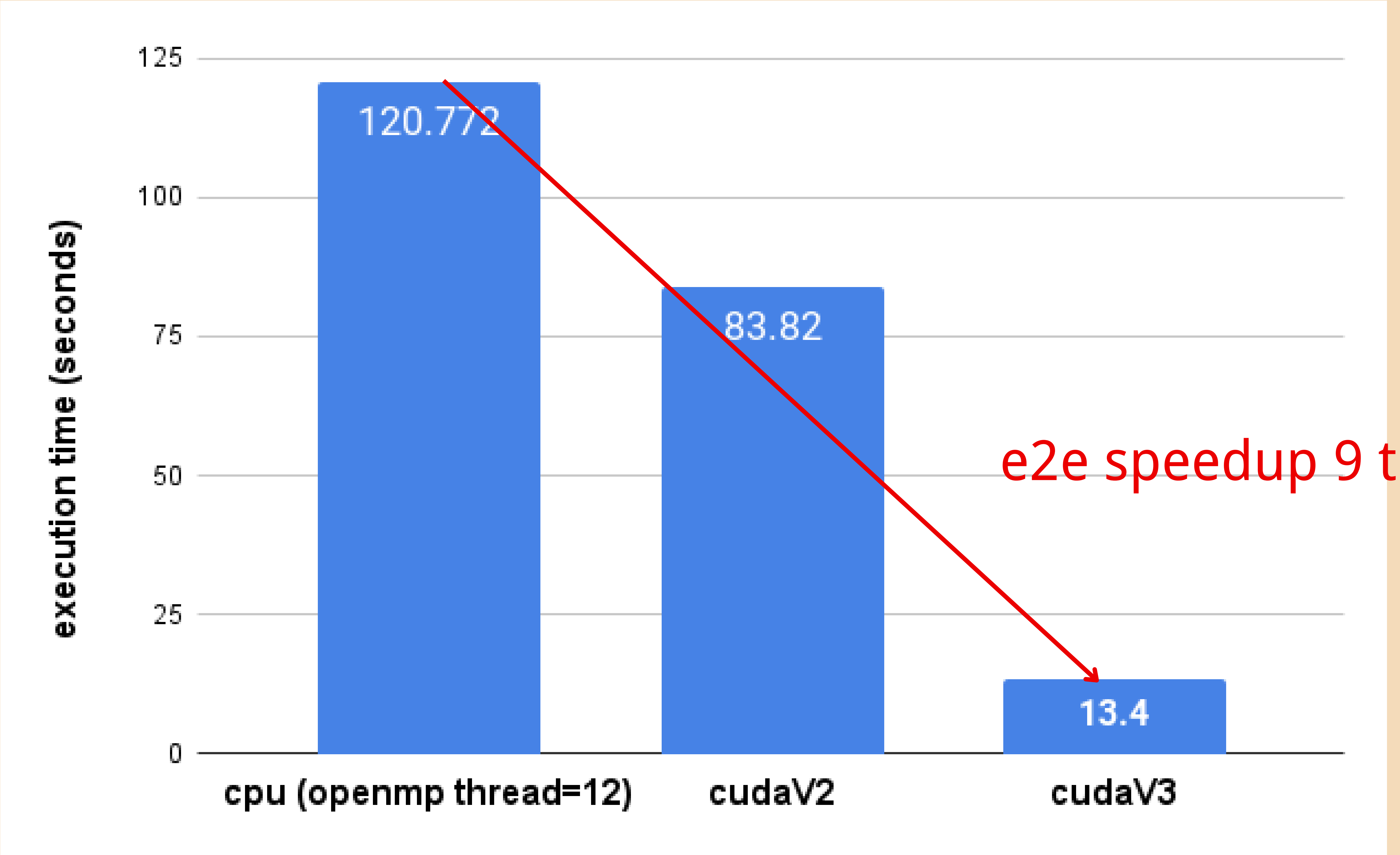




- 解決Week2遇到的memcpy
- gpu utilization > 90%

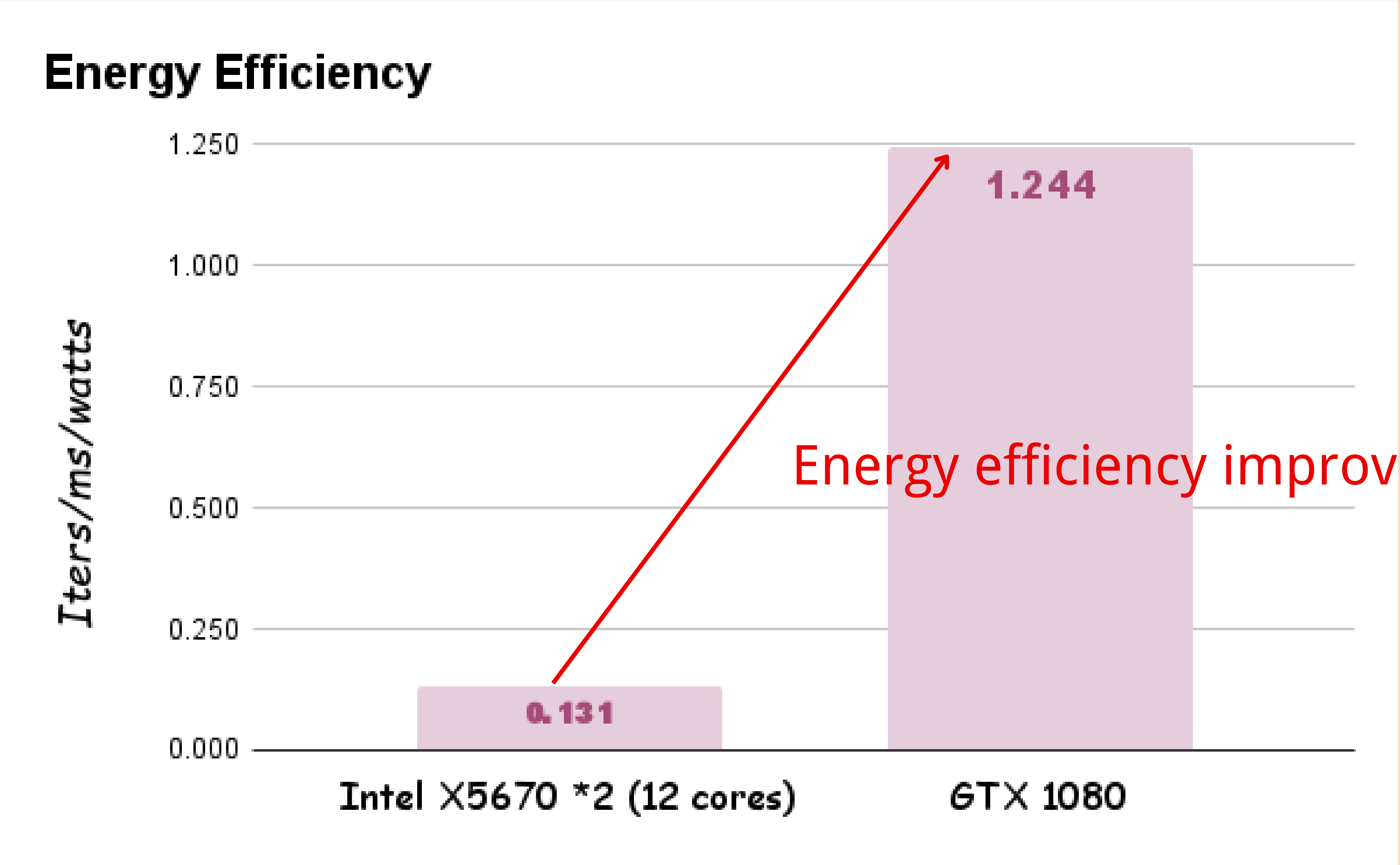
Progress -Week3

problem size : population:1024 dimen:512 max_iter 3



e2e speedup 9 times

Energy Efficiency improvement



Energy efficiency improve 10 times

What problems have you encountered?

- 平行運算時遇到race condition：謹慎處理data dependency避免同步讀寫資料
- 盡可能減少Memcpy時間
- 能夠運用多個blocks中多個wrap運算提高平行度

Wishlist

- 可以贊助H100給清大的超算團隊嗎？
- 免費cuda教學課程

OpenACC More Science, Less Programming OPEN HACKATHONS NCHC NARLabs 財團法人國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing NVIDIA

Final Thoughts 超級值得!!!!

1.讓我更熟悉cuda語法，並成功用cuda加速firefly algorithm

2.知道如何解決Memcpy耗時情況

3.學會cuda加速小技巧 : WarpReduce。

4.更熟悉如何使用nsys profile並解讀report

What’s next?

1.優化fun_kernel : use coalesced memory access

2.優化update_best_kernel : 使用多個blocks平行計算

謝謝大家!!!

OpenACC
More Science. Less Programming

OPEN HACKATHONS

NCHC

NARLabs 財團法人國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing

NVIDIA.