Homework 2 Report

Yige Hu and Zhiting Zhu

1 P1

```
1: function REC_PSUM(a, x_0, b, n)

2:     if (n == 1) then
3:         s(0) = x_0; return; end;
4:     end if

5:     x = zeros(n/2, 1);
6:     a_new = zeros(n/2 − 1, 1);
7:     x(0) = x_0;
8:     parfor i = 1 : n do
9:         x(i) = b(i);
10:    end parfor

11:    parfor i = 0 : n/2 − 1 do
12:        y(i) = x(2 ∗ i) ∗ a(2 ∗ i + 1) + x(2 ∗ i + 1);
13:        if (i != 0) then
14:            a_new(i) = a(2 ∗ i) ∗ a(2 ∗ i + 1);
15:        end if
16:    end parfor

17:    c = REC_PSUM(a_new, y(0), y[1 : n/2 − 1], n/2) ;

18:    s(0) = x_0;
19:    parfor i = 1 : n − 1 do
20:        if isOdd(i) then
21:            s(i) = c(i/2);
22:        else
23:            s(i) = c((i − 1)/2) ∗ a(i) + x(i);
24:        end if
25:    end parfor

26:    return s;
27: end function
```

2 P2

2.1 Algorithm

```
1: function SCAN(X, n, l)

2:     step = ceil(log_2(n))
3:     temp = n >> 1
```

```
    offset = 1

    parfor i = 0 : n/2 − 1 do
        for j = i; j < temp; j+ = nthreads do
            indx2 = offset ∗ (2 ∗ i + 2) − 1
            indx1 = offset ∗ (2 ∗ i + 1) − 1
            x(indx2) = x(indx1) + x(indx2)
        end for
        offset∗ = 2
        temp = temp >> 1
    end parfor

    temp = 2
    offset >>= 1

    parfor i = 1 : n/2 − 1 do
        offset >>= 1
        for j = i; j < temp; j+ = nthreads do
            indx2 = offset ∗ (2 ∗ i + 1) − 1
            indx1 = offset ∗ 2 ∗ i − 1
            x(indx2) = x(indx1) + x(indx2)
        end for
        temp∗ = 2
    end parfor
end function
```

2.2 Result

Wall Clock Time(us)	Number of threads		
Length of Arrary	sequential	6 threads	12 threads
1M	15679	15500	38192
10M	156797.9	212012.5	160871
100M	730794.8	1513714	1262623.5
1B	7305516.5	14843186	12431315.5

Table 1: Wall clock execution time for different array size with different number of threads for 1D vectors

Wall Clock Time(us)	Number of threads		
Length of Arrary	sequential	6 threads	12 threads
1M	20525.5	79923.5	146187
10M	247284.5	539063	375131.5
100M	2046770	4615023.5	3381959

Table 2: Wall clock execution time for different array size with different number of threads for 4D vectors

3 P3

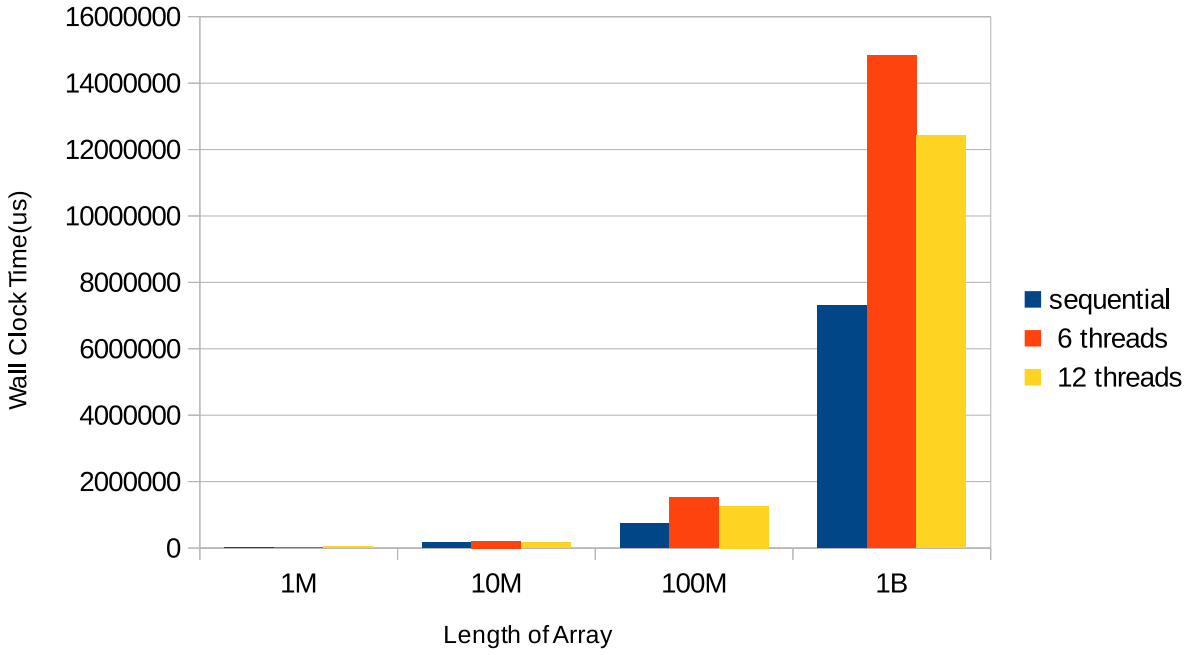

Figure 1: Wall clock execution time for different array size with different number of threads for 1D vectors

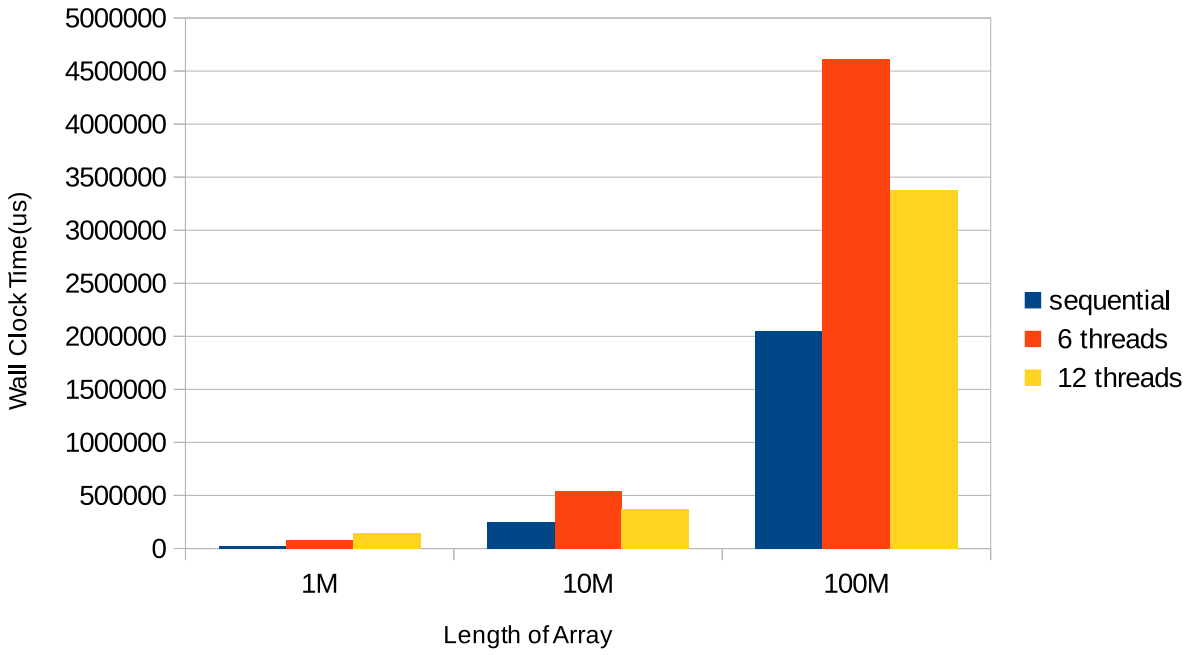

Figure 2: Timing measurements for different array size with different number of threads for 4D vectors