

OpenMP Case Study: Bubble Sort

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University



original	0	1	2	3	4	5
6	5	5	3	3	1	1
5	6	3	5	1	3	2
4	3	6	1	5	2	3
3	4	1	6	2	5	4
2	1	4	2	6	4	5
1	2	2	4	4	6	6



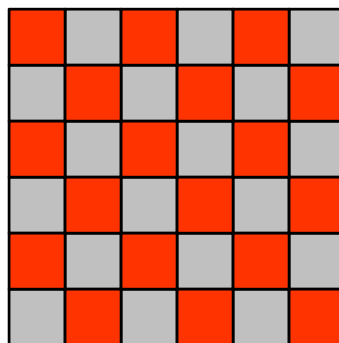
Oregon State University
Computer Graphics

bubblesort.pptx

mjb - March 29, 2017

A Special Parallel Design Pattern

Implementing a Bubble Sort in parallel is an example of a special design pattern called **Even-Odd**, or **Red-Black**



Oregon State University
Computer Graphics

mjb - March 29, 2017

Non-threaded Bubble Sort

3

```
#include <algorithm>
...
for( int i = 0; i < N; i++ )
{
    bool stop = true;
    for( int j = 0; j < N-1; j++ )
    {
        if( B[j] > B[j+1] )
        {
            std::swap( B[j], B[j+1] );
            stop = false;
        }
    }
    if( stop )
        break;
}
```

N = 6

original	Step #				
	0	1	2	3	4
6	5	4	3	2	1
5	4	3	2	1	2
4	3	2	1	3	3
3	2	1	4	4	4
2	1	5	5	5	5
1	6	6	6	6	6



Oregon State University
Computer Graphics

mjb - March 29, 2017

Why Can't This Version of the Bubble Sort Be Run in Parallel?

4

```
#include <algorithm>
...
for( int i = 0; i < N; i++ )
{
    bool stop = true;
    if( B[0] > B[1] )
    {
        std::swap( B[0], B[1] );
        stop = false;
    }
    if( B[1] > B[2] )
    {
        std::swap( B[1], B[2] );
        stop = false;
    }
    if( B[2] > B[3] )
    {
        std::swap( B[2], B[3] );
        stop = false;
    }
    ...
    if( stop )
        break;
}
```

Let's unroll the inner (j) loop so we can see what the for-loop really looks like.

Suppose each of these if-blocks gets assigned to a different thread (remember that OpenMP tries to assign different for-loop passes to different threads).

Remembering that we have no explicit control over thread scheduling, notice that both the first and second if-blocks are both reading from and writing to **B[1]**. There is no synchronization to control in which order this is happening. We have a classic **Race Condition**.

The solution is, in one pass, allow a single thread access to B[0] and B[1], another thread access to B[2] and B[3], another thread access to B[4] and B[5], etc.

Then, in the next pass, allow a single thread access to B[1] and B[2], another thread access to B[3] and B[4], another thread access to B[5] and B[6], etc.



mjb - March 29, 2017

Threaded Bubble Sort

5

```
#include <algorithm>
...
for( int i = 0; i < N; i++)
{
    int first = i % 2; // 0 if i is 0, 2, 4, ...
                    // 1 if i is 1, 3, 5, ...

    #pragma omp parallel for default(none),shared(A,first)

    for( int j = first; j < N-1; j += 2 )
    {
        if( A[j] > A[j+1] )
        {
            std::swap( A[j], A[j+1] );
        }
    }
}

```

N = 6

	original	0	1	2	3	4	5
6	6	5	5	3	3	1	1
5	5	6	3	5	1	3	2
4	4	3	6	1	5	2	3
3	3	4	1	6	2	5	4
2	2	1	4	2	6	4	5
1	1	2	2	4	4	6	6



Oregon State University
Computer Graphics

mjb - March 29, 2017

A Comparison

6

N = 6

	original	0	1	2	3	4
6	6	5	4	3	2	1
5	5	4	3	2	1	2
4	4	3	2	1	3	3
3	3	2	1	4	4	4
2	2	1	5	5	5	5
1	1	6	6	6	6	6

Non-threaded

	original	0	1	2	3	4	5
6	6	5	5	3	3	1	1
5	5	6	3	5	1	3	2
4	4	3	6	1	5	2	3
3	3	4	1	6	2	5	4
2	2	1	4	2	6	4	5
1	1	2	2	4	4	6	6

Threaded



Oregon State University
Computer Graphics

mjb - March 29, 2017

