OpenMP Case Study: Bubble Sort

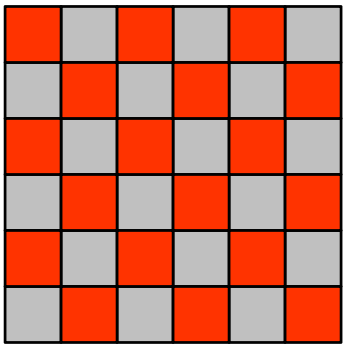
Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University

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***



Non-threaded Bubble Sort

```
#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

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

```
#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.

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 to allow a single thread access to B[0] and B[1] only, another thread access to B[2] and B[3] only, another thread access to B[4] and B[5] only, etc.

Threaded Bubble Sort

```
#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	Step # 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

A Comparison

N = 6

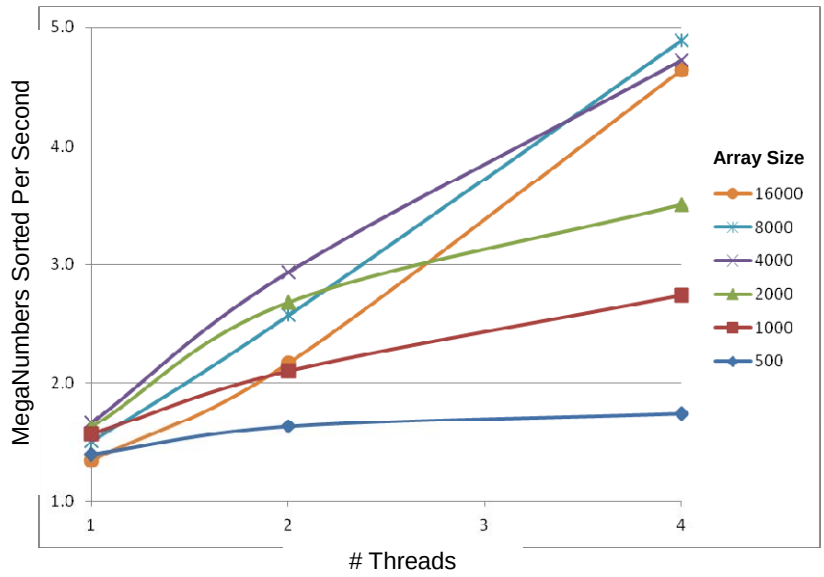
original	Step # 0	1	2	3	4
6	5	4	3	2	1
5	4	3	2	1	2
4	3	2	1	3	3
3	2	1	4	4	4
2	1	5	5	5	5
1	6	6	6	6	6

Non-threaded

original	Step # 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

Threaded

OpenMP Performance as a Function of # of Threads



OpenMP Performance as a Function of Array Size