OpenMP Case Study: Bubble Sort

Mike Bailey
mjb@cs.orangestate.edu
Oregon State University

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

#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;
}

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

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.


#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]);
        }
    }
    if (stop)
        break;
}

A Comparison

<table>
<thead>
<tr>
<th>N = 6</th>
<th>Threaded</th>
<th>Non-threaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>0 1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>6 5 4 3 2 1</td>
<td>6 5 4 3 2 1</td>
<td></td>
</tr>
<tr>
<td>5 4 3 2 1 6</td>
<td>5 4 3 2 1 6</td>
<td></td>
</tr>
<tr>
<td>4 3 2 1 3 2</td>
<td>4 3 2 1 3 2</td>
<td></td>
</tr>
<tr>
<td>3 2 1 4 5 4</td>
<td>3 2 1 4 5 4</td>
<td></td>
</tr>
<tr>
<td>2 1 5 6 2 1</td>
<td>2 1 5 6 2 1</td>
<td></td>
</tr>
<tr>
<td>1 6 6 6 6 6</td>
<td>1 6 6 6 6 6</td>
<td></td>
</tr>
</tbody>
</table>

For N = 6, the threaded version outperforms the non-threaded version.
OpenMP Performance as a Function of Array Size

OpenMP Performance as a Function of # of Threads