Project 6 : Mandelbrot Set with OpenMP & MPI

Duane Johnson

Mar 30, 2009

Introduction

The Mandelbrot Set is a set of values on the complex plane that can be approximated to any precision. This lab uses both the OpenMP library for parallelizing a Mandelbrot algorithm across several processors on a single node, as well as the MPI library for message passing between nodes. By combining the two libraries, the calculations can be distributed across a cluster of almost any size.

Procedure

Because the Mandelbrot Set has no data dependencies between its cells ("pixels") it is very easy to parallelize. This lab uses a root node as a central organizing (and reduction) hub so that local and remote worker threads can combine their work and organize it into the final rendered image.

The parallelization of the algorithm was achieved as follows:

1. Vertical "stripes" equal to the number of nodes available were split up from the problem space and sent to each node.
2. Each node was then responsible for the further horizontal partitioning of its problem space for assignment to processors on the node (4 in this case).
3. When all nodes finished their work, they reported to the root node, where a "gather" operation completed the process and stitched the results together.

Experimental Setup

The open-source libraries OpenMPI and OpenMP were used for the implementation of the Mandelbrot algorithm in C.

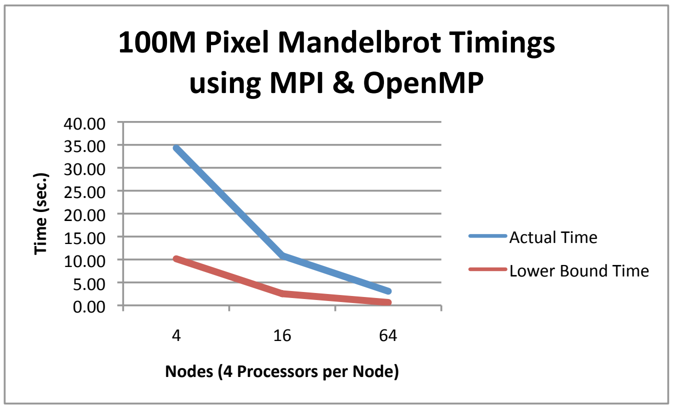
Timing of the solutions were performed on a supercomputer at BYU configured with 2.6GHz Xeon EM64T dual-core processors and 8 GB of fully-buffered RAM. Processes were distributed across a network of nodes using ethernet for communication.

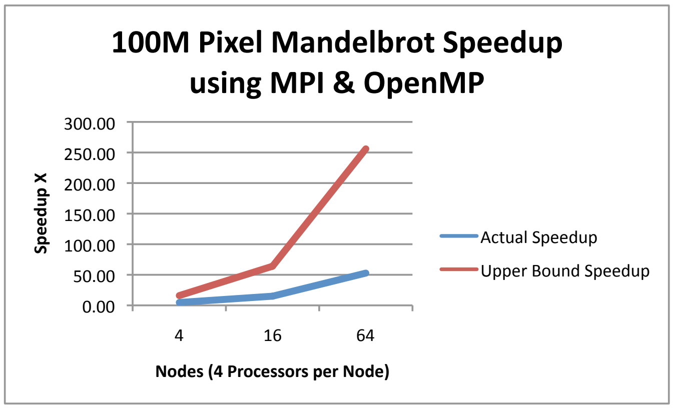
Results

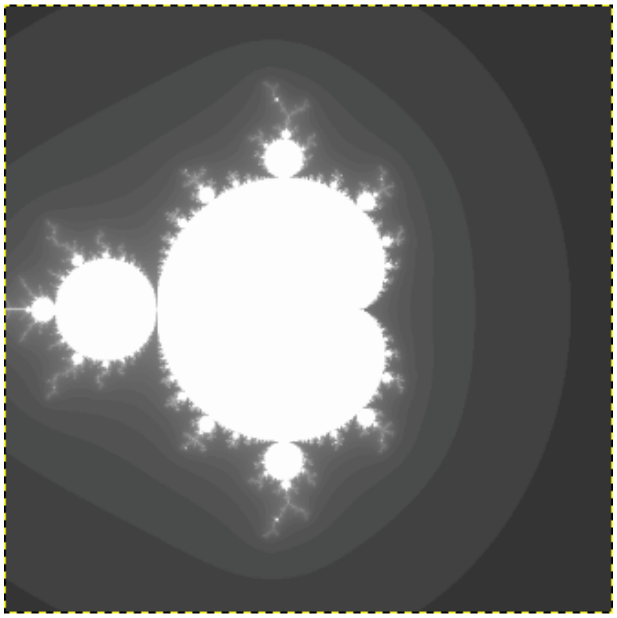
As expected, significant absolute speedups were recorded for multi-node and multi-processor runs. In particular, a speedup of 50x was achieved when the algorithm was spread across 64 nodes with 4 processors each. Although this did not approach the theoretical upper bound of 256 X speedup, it was sufficient for this task.

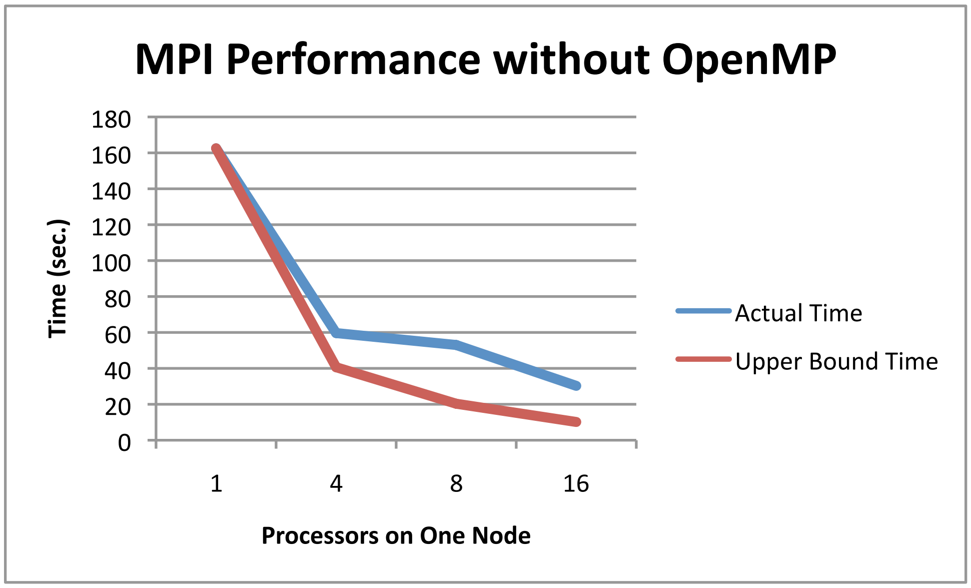
Surprisingly, using MPI without OpenMP achieved nearly the same speedup for up to 16 processors. This suggests that communication is not a bottleneck for my algorithm--rather, it is the raw processing speed that is limiting faster computation.

See page 2 for graphs and data.


100M Pixel Mandelbrot Timings using MPI & OpenMP
Time (sec.)
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
4
16
64
Nodes (4 Processors per Node)
Actual Time
Lower Bound Time


100M Pixel Mandelbrot Speedup using MPI & OpenMP
Speedup X
300.00
250.00
200.00
150.00
100.00
50.00
0.00
4
16
64
Nodes (4 Processors per Node)
Actual Speedup
Upper Bound Speedup




MPI Performance without OpenMP
Time (sec.)
180
160
140
120
100
80
60
40
20
0
1
4
8
16
Processors on One Node
Actual Time
Upper Bound Time