Analysis Report for PA4

By: Omer Waseem (#000470449) and Erica Cheyne (#001201341)

Run Times for PA4:

n\image size	852x480	1920x1079	3840x2160
1	8.117201e-01 seconds	4.412289e+00 seconds	1.758381e+01 seconds
2	4.204490e-01 seconds	2.159110e+00 seconds	8.748059e+00 seconds
3	2.729330e-01 seconds	1.432044e+00 seconds	5.845148e+00 seconds
4	2.023339e-01 seconds	1.066779e+00 seconds	4.393826e+00 seconds
8	9.742689e-02 seconds	5.222611e-01 seconds	2.164460e+00 seconds
16	4.166102e-02 seconds	2.460780e-01 seconds	1.046391e+00 seconds
32	2.076793e-02 seconds	1.118002e-01 seconds	4.980402e-01 seconds

Speedup for PA4 (rounded to one decimal place):

n\image size	852x480	1920x1079	3840x2160
1	1	1	1
2	1.9	2.0	2.0
3	2.9	3.1	3.0
4	4.0	4.1	4.0
8	8.3	8.4	8.1
16	19.5	17.9	16.8
32	39.1	39.5	35.3

Efficiency for PA4 (rounded to 2 decimal places):

n\image size	852x480	1920x1079	3840x2160
1	1	1	1
2	0.97	1.02	1.01
3	0.99	1.03	1.00
4	1.00	1.03	1.00
8	1.04	1.06	1.02
16	1.22	1.12	1.05
32	1.22	1.23	1.10

This program is efficient and strongly scalable because the problem size can be increased without decreasing efficiency, even as more processes are used. The efficiency in general is linear, not only for small p and n but for all values of p and n. Our speedup matches the ideal value of p, or very close, in all cases.

Note: our efficiencies values are over 1 which needs some explanation, since we expect them to be 1 or below. We suspected this could be due to the variance in the measured runtimes for the same n and p values. However even with multiple runs and averaged times we still got

efficiencies over 1, even when we used a purely serial blur implementation without MPI. We suspect that as we add additional nodes to MPI, these nodes are faster in computation than mpihost01. Furthermore, process 0 is responsible for handling top and bottom edges, and to compensate for this extra computation we offload the work assigned to process 0 to other nodes. This might also lead to efficiency values over 1.