Rob Dryke

453159

(Serial is without any MPI functions at all, while 1 is the MPI program but with one processor.)
These times below are most likely not an accurate reflection of the algorithm I implemented because I think there is a stack corruption (or something, I really have no clue) somewhere. The program compiles and runs, but some random values of the output vector are wrong. For some parameters the output vector is 100% correct, others 2 or 4 or 6 elements wrong. My program also has a blank printf statement, i.e. printf(“”) as this prevents a seg fault. No clue why that is.

SERIAL:

total time taken for serial of size 50000A = 0.1280

total time taken for serial of size 50000B = 0.2654

total time taken for serial of size 100000A = 0.2551

total time taken for serial of size 100000B = 0.9219

total time taken for serial of size 200000A = 0.5093

total time taken for serial of size 200000B= 2.8858

total time taken for serial of size 400000A = 1.0377

total time taken for serial of size 400000B = 7.3111

1 PROCESSOR

Size: 50000A, Total Time Taken: 0.4483 sec; Time Taken (Steps 5&6) : 0.3262 sec

Size: 50000B, Total Time Taken: 0.5978 sec; Time Taken (Steps 5&6) : 0.3483 sec

Size: 100000A, Total Time Taken: 0.9147 sec; Time Taken (Steps 5&6) : 0.6813 sec

Size: 100000B, Total Time Taken: 1.5921 sec; Time Taken (Steps 5&6) : 0.7047 sec

Size: 200000A, Total Time Taken: 1.7804 sec; Time Taken (Steps 5&6) : 1.3119 sec

Size: 200000B, Total Time Taken: 4.3040 sec; Time Taken (Steps 5&6) : 1.4612 sec

Size: 400000A, Total Time Taken: 3.5892 sec; Time Taken (Steps 5&6) : 2.6327 sec

Size: 400000B, Total Time Taken: 10.0818 sec; Time Taken (Steps 5&6) : 2.9176 sec

2 PROCESSORS

Size: 50000A, Total Time Taken: 0.2342 sec; Time Taken (Steps 5&6) : 0.1640 sec

Size: 50000B, Total Time Taken: 0.3029 sec; Time Taken (Steps 5&6) : 0.1753 sec

Size: 100000A, Total Time Taken: 0.4577 sec; Time Taken (Steps 5&6) : 0.3344 sec

Size: 100000B, Total Time Taken: 0.8319 sec; Time Taken (Steps 5&6) : 0.3530 sec

Size: 200000A, Total Time Taken: 0.9651 sec; Time Taken (Steps 5&6) : 0.6859 sec

Size: 200000B, Total Time Taken: 2.2557 sec; Time Taken (Steps 5&6) : 0.7104 sec

Size: 400000A, Total Time Taken: 1.8684 sec; Time Taken (Steps 5&6) : 1.3107 sec

Size: 400000B, Total Time Taken: 5.3119 sec; Time Taken (Steps 5&6) : 1.4379 sec

4 PROCESSORS

Size: 50000A, Total Time Taken: 0.1207 sec; Time Taken (Steps 5&6) : 0.0816 sec

Size: 50000B, Total Time Taken: 0.2074 sec; Time Taken (Steps 5&6) : 0.0914 sec

Size: 100000A, Total Time Taken: 0.2346 sec; Time Taken (Steps 5&6) : 0.1649 sec

Size: 100000B, Total Time Taken: 0.5617 sec; Time Taken (Steps 5&6) : 0.1773 sec

Size: 200000A, Total Time Taken: 0.4945 sec; Time Taken (Steps 5&6) : 0.3291 sec

Size: 200000B, Total Time Taken: 1.3945 sec; Time Taken (Steps 5&6) : 0.3652 sec

Size: 400000A, Total Time Taken: 1.0234 sec; Time Taken (Steps 5&6) : 0.6543 sec

Size: 400000B, Total Time Taken: 3.2341 sec; Time Taken (Steps 5&6) : 0.7149 sec

8 PROCESSORS

Size: 50000A, Total Time Taken: 0.0590 sec; Time Taken (Steps 5&6) : 0.0407 sec

Size: 50000B, Total Time Taken: 0.0824 sec; Time Taken (Steps 5&6) : 0.0443 sec

Size: 100000A, Total Time Taken: 0.1251 sec; Time Taken (Steps 5&6) : 0.0820 sec

Size: 100000B, Total Time Taken: 0.2540 sec; Time Taken (Steps 5&6) : 0.0889 sec

Size: 200000A, Total Time Taken: 0.2542 sec; Time Taken (Steps 5&6) : 0.1714 sec

Size: 200000B, Total Time Taken: 0.7342 sec; Time Taken (Steps 5&6) : 0.1907 sec

Size: 400000A, Total Time Taken: 0.5311 sec; Time Taken (Steps 5&6) : 0.3293 sec

Size: 400000B, Total Time Taken: 1.9192 sec; Time Taken (Steps 5&6) : 0.3625 sec

16 PROCESSORS

Size: 50000A, Total Time Taken: 0.0455 sec; Time Taken (Steps 5&6) : 0.0403 sec

Size: 50000B, Total Time Taken: 0.0606 sec; Time Taken (Steps 5&6) : 0.0376 sec

Size: 100000A, Total Time Taken: 0.1239 sec; Time Taken (Steps 5&6) : 0.0909 sec

Size: 100000B, Total Time Taken: 0.2020 sec; Time Taken (Steps 5&6) : 0.0898 sec

Size: 200000A, Total Time Taken: 0.2323 sec; Time Taken (Steps 5&6) : 0.1616 sec

Size: 200000B, Total Time Taken: 0.7326 sec; Time Taken (Steps 5&6) : 0.1833 sec

Size: 400000A, Total Time Taken: 0.4398 sec; Time Taken (Steps 5&6) : 0.3244 sec

Size: 400000B, Total Time Taken: 2.0809 sec; Time Taken (Steps 5&6) : 0.3870 sec

Implementation:

Step2: Each processor first calls gk_csr_CreateIndex(mat, GK_CSR_COL) to get the matrix in column form. Then it loops through the number of columns until the mat->colptr says all of the elements have been passed (the rest of the columns are not needed). It then makes an array of that size and puts the actual columns needed in said array. Auxiliary arrays are used to keep track of the number of elements going to each processor and the offsets for each processor. I think I could have done this part in one pass through resizing the array as it grows through realloc, but this was simpler so I kept it.

Step3: This step is implicitly done as a MPI_Scatterv is used, which sends the data request array to the right processor.

Step4: First an MPI_ALLtoall broadcast is used to tell each processor how many elements the other processors need from it. On each processor another auxiliary array is computed with the cumulative number of elements from each processor. Then each processor MPI_Scatterv's its b data request array. Each processor then makes an array of the total number of requested elements and fills that with its b in the same order as within the d request array.

Step5: Each processor does another Scatterv to transfer the elements of b back to the processor that needs it.

Step6: A loop is done over the array that was filled by the p Scattervs in step 6, and each row index and float value from the matrix are found using the mat->colptr, mat->colind, and mat->colval arrays. Thus no more time is needed to access the gathered elements of b versus the ones already on the processor.

RunTime:

This is O(# of distinct elements assigned to each processor) as that is the largest loop in the program.

n is the number of data elements

Tp = (n/p) * Tc + (logp * Ts + logp * Tw)(Broadcast) + 8 * (logp * Ts + p * Tw)(7 scatters, 1 gather) + (p * Ts + p * Tw)(Alltoall)

thus:

n = p^2

p = n/logn