TDT4200 - PS2b

2.1 Number crunching

1)

- The master rank takes care of the parameter phrasing and the loading of the input image.
- Then some variables (iterations, imgXSize, imgYSize, kernelIndex) that are needed by all the ranks are distributed using **MPI_Bcast.**
- The data is then distributed to the processors by the master rank using **MPI_Scatterv**. If the height of the image does not divide evenly between the ranks, the reminder rows are distributed to the first rank, then the second and so on.
- The size of the borders to send and receive are calculated using the kernel dimension
- Firstly, the odd ranks send and receive their north border (except the last rank), and the even rank send and receive their south border. Then the odd ranks send and receive their south border, and the even ranks send and receive their north border (except the first rank). This is done using **MPI_Sendrecv**, and the pattern prevent a deadlock.
- The send and receive are done every iteration, and the southRecv and northRecv are used in the modified applyKernelFunction for calculating the top and bottom values of the local image chunk.
- The image is then gathered back to the master rank using **MPI_Gatherv** and is written back to disk.

2)

num_of_processors = 4
i = 2
image_height = 2334
image_width = 4000
Laplacian 1 kernel

Before the loop

Call	**Description**	**Amount**	**Type**	**Bytes**
MPI_Bcast	iterations, imgXSize,	4*2 = 8 4*2 = 8	unsigned int int	8*4 bytes = 32 bytes 8*4 bytes = 32 bytes

	imgYSize, kernelIndex			
MPI_Scatterv	The image rawdata is scattered to all the processes by master rank	9336000*3 = 28 008 000	unsigned char	28 008 000 bytes

The loop

Call	Description	Amount	Type	Bytes
MPI_SendRecv (odd, north)	rank 1, rank 3	2 iterations* 4000*2 rows* 3 = 48 000	unsigned char	48 000 bytes
MPI_Sendrecv (even, south)	rank 0, rank 2	2 iterations* 4000*2 rows* 3 = 48 000	unsigned char	48 000 bytes
MPI_Sendrecv(odd, south)	rank 1	2 iterations* 4000*1 row* 3 = 24 000	unsigned char	24 000 bytes
MPI_Sendrecv(even, north)	rank 3	2 iterations* 4000*1 row* 3 = 24 000	unsigned char	24 000 bytes

After the loop

Call	Description	Amount	Type	Bytes
MPI_Gatherv	The image rawdata is gathered back to all the processes by master rank	9336000*3 = 28 008 000	unsigned char	28 008 000 bytes

The total data transferred between processes: 64 + 28 008 000 + 48 000 + 48 000 + 24 000 + 28 008 000 = **56 160 064 bytes**.

3)

num_of_processors = 8
i = 2
image_height = 2334
image_width = 4000
Laplacian 1 kernel

Before the loop

Call	Description	Amount	Type	Bytes
MPI_Bcast	iterations, imgXSize, imgYSize, kernelIndex is broadcasted to all ranks	8*2 = 16 8*2 = 16	unsigned int int	16*4 bytes = 64 bytes 16*4 bytes = 64 bytes
MPI_Scatterv	The image rawdata is scattered to all the processes by master rank	9336000*3 = 28 008 000	unsigned char	28 008 000 bytes

The loop

Call	Description	Amount	Type	Bytes
MPI_SendRecv (odd, north)	rank 1, rank 3, rank 5, rank 7	2 iterations* 4000*4 rows* 3 = 96 000	unsigned char	96 000 bytes
MPI_Sendrecv (even, south)	rank 0, rank 2, rank 4, rank 6	2 iterations* 4000*4 rows* 3 = 96 000	unsigned char	96 000 bytes
MPI_Sendrecv(odd, south)	rank 1, rank 3, rank 5	2 iterations* 4000*3 rows* 3 = 72 000	unsigned char	72 000 bytes
MPI_Sendrecv(even, north)	rank 2, rank 4, rank 6	2 iterations* 4000*3 rows* 3 = 72 000	unsigned char	72 000 bytes

After the loop

Call	Description	Amount	Type	Bytes
MPI_Gatherv	The image rawdata is gathered back to all the processes by master rank	9336000*3 = 28 008 000	unsigned char	28 008 000 bytes

The total data transferred between processes: 128 + 28 008 000 + 96 000 + 96 000 + 72 000 + 72 000 + 28 008 000 = **56 352 128 bytes**.

Overhead: 56 352 128 - 56 160 064 = **192 064 bytes**

2.2 Speedup analysis

- num_of_processors = 8
- Laplacian 1 kernel

Iterations	Baseline	MPI
1	0.175s	0.061s
100	17.261s	5.097s
400	70.401s	21.993s
800	141.548s	41.201s
1024	181.581s	52.692s