Report Week 9

Each group member contributed equally to completing the task.

The most important work that had to be done was the communication part in the start_halo_exchange method. To send data to the right neighbor you first had to find the correct position in the data array. You had to differentiate between sending the first row / column of the temperature field and receiving the correct ghost cell row / column.
To communicate we went to the correct spot in the data array and for the upper and lower neighbor proceeded to send / receive a whole row using the rowtype MPI_Datatype. Equally for the left and right neighbor we used the columntype to send / receive a whole column at once without needing to change a lot besides the correct position in the data array.
After completing start_halo_exchange we had to complete the complete_halo_exchange, as we were using non-blocking communication, which was simply done by waiting for all the requests to align with the MPI_Waitall method.
The update_interior_temperature and update_boundary_temperature methods than were pretty similar. We first had to set the correct indices and proceed using the formula given on the exercise-website. It would have been possible to optimize some calculations by extracting formulas like ((a * dt) / (dx_2)) into a separate variable but as we wanted to stay as close as possible to the formula to mix up something we decided to let the formulas be, as they still should work that way (just a little slower over all).
After completing all four methods we went into the main fill to rearrange these methods. We decided to first update the boundary temperature so that every process has the correct ghost values for communication. Then we started the halo exchange to exchange the ghost cells, updated the interior temperature while the communication goes on and finally set the complete_halo_exchange method to signalize that one cycle has been completed.
Additionally we set the alpha value and calculated the timeStep in the main method.

After fulfilling these tasks we went into the ara-cluster to start some experiments. As the normal execution was working with a time about „ 19.7181s“ using 4 processes we proceeded to try the exact same execution with 16 processes resulting in a time of „ 45.0481s“. That meant that using more processes with our implemented code lead to a time that was overall twice as slow, which can only be explained due to the fact that more communication was involved in the process.
Afterwards we changed the resolution from 4000x2000 to 8000x4000 (and 16 processes) which resulted in a time of „180.259s“ which can be explained by saying that it had to do a lot more calculations than before as every other setting was still the same.
An last but not least we changed the temperature of the inside disc temperature to „130“ and the outside temperature to „10“ which of course changed the color in the output.gif file to a red in the inside of the disc but the overall time didn’t change at all: „46.6067s“ (comparing to the 16 process result).

Out of curiosity we wanted to find out if the number of processes increased while using more processes, so we tried the standard execution for 8 and 32 processes. The results were:

1. for 8 processes it took an estimated time of „ 28.1594s“ which was only slightly slower than the run with 4 processes.
2. for 32 processes it took an estimated time of „80.0054s“ which was 4 times as much as the run with 4 processes.

To conclude our results we can say that our bottleneck in the calculation definitely is our communication which could be changed by using one-sided communication as it wouldn’t be necessary to wait every time for all Sends / Receives to complete.

The last thing we did was to optimize the calculations by pre calculating parts of the formula. This time we compared a 4 process run with the 16 process run and the results were:

1. for the 4 process run the time was about „18.6945s“ which is slightly faster than the unoptimized version.
2. for the 16 process run the time was about „46.5834s“ which again is slightly faster than the unoptimized version.

In conclusion one could say that pre calculation didn't really make a big change but we think if you would have more calculations that you could extract and pre calculate it could make a big difference.

P.S.: For clarification we renamed the folders from 1 to 8 so that we have some kind of order in our file systems. Further we didn't include the heat_mpi and HEAT.dat as they would take unnecessary disk space. If you still need these you can tell us and we would calculate them for you.