Performance Optimization of the Lattice Boltzmann Method for Computational Fluid Dynamics

Kseniia Ovchinnikova, Anna Grebennikova, Anna Remizova
UGA

April 25, 2025

Exercise 1: 1D Blocking Communication

- `lbm_comm_init_ex1()` is based on exercise 0, with some changes:
  - Communicator size is > 0.
  - Total width is divisible by number of processes.
  - X: communicator size; Y remains 1.
- `lbm_comm_ghost_exchange_ex1()`:
  - Computed ranks of left and right neighbors.
  - Neighbors set to `MPI_PROC_NULL` at domain boundaries.
  - Performed column exchange using blocking communication (we first do the yellow passage and wait for it to finish, and then we do the green one).

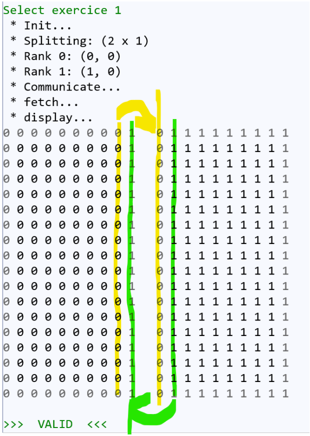

Figure: Illustration of ghost cell exchange for 2 processes.

Exercise 2 and 3: Communication Implementations

Blocking communication can lead to deadlocks when working with large datasets and many processes. To address this, we implement two alternative strategies:

- **Exercise 2: 1D odd/even communication**
  - `lbm_comm_init_ex2()` is based on `ex1`.
  - `lbm_comm_ghost_exchange_ex2()`:
    - Even ranks: receive first, then send.
    - Odd ranks: send first, then receive.
- **Exercise 3: 1D non-blocking communication**
  - `lbm_comm_init_ex3()` is based on `ex1`.
  - `lbm_comm_ghost_exchange_ex3()`:
    - Initialized MPI request array for non-blocking communication.
    - Used `MPI_Isend` and `MPI_Irecv`.
    - Finalized with `MPI_Waitall`.

Exercise 4: 2D blocking communication, manual copy

- `lbm_comm_init_ex4()` is based on `lbm_comm_init_ex1()` with several changes:
  - Initialized 2D grid dimensions with `MPI_Dims_create`.
  - Created a Cartesian communicator with `MPI_Cart_create`.
  - Retrieved process coordinates using `MPI_Cart_coords`.
  - The matrix is stored in column-major order; for up/down exchanges, buffers must be built manually as data is not stored linearly in memory.
  - Allocated `buffer_recv_up`, `buffer_recv_down`, `buffer_send_up`, and `buffer_send_down`.
- `lbm_comm_release_ex4()` was implemented to free the buffers and the communicator.

Exercise 4: Blocking communication and manual copy

- `lbm_comm_ghost_exchange_ex4()`:
  - Defined variables for neighbor ranks.
  - NB: we didn't need to implement a helper function to get the left/right neighbours because we implemented cartesian topology - we simply use `MPI_Cart_shift` for it.
- *Left-right* communication remains the same as before.
- For *up-down* communication we built send buffers manually using `for` loops.
- After communication, we unpacked the received buffers into the mesh.

Exercise 4: Results

```
Select exercice 4
 * Init...
 * Splitting: (2 x 2)
 * Rank 0: (0, 0)
 * Rank 1: (0, 1)
 * Rank 2: (1, 0)
 * Rank 3: (1, 1)
 * Communicate...
 * fetch...
 * display...
000 010 020 030 040 050 060 070 080 090   080 090 100 110 120 130 140 150 160 170
001 011 021 031 041 051 061 071 081 091   081 091 101 111 121 131 141 151 161 171
002 012 022 032 042 052 062 072 082 092   082 092 102 112 122 132 142 152 162 172
003 013 023 033 043 053 063 073 083 093   083 093 103 113 123 133 143 153 163 173
004 014 024 034 044 054 064 074 084 094   084 094 104 114 124 134 144 154 164 174
005 015 025 035 045 055 065 075 085 095   085 095 105 115 125 135 145 155 165 175
006 016 026 036 046 056 066 076 086 096   086 096 106 116 126 136 146 156 166 176
007 017 027 037 047 057 067 077 087 097   087 097 107 117 127 137 147 157 167 177
008 018 028 038 048 058 068 078 088 098   088 098 108 118 128 138 148 158 168 178
009 019 029 039 049 059 069 079 089 099   089 099 109 119 129 139 149 159 169 179

008 018 028 038 048 058 068 078 088 098   088 098 108 118 128 138 148 158 168 178
009 019 029 039 049 059 069 079 089 099   089 099 109 119 129 139 149 159 169 179
010 020 030 040 050 060 070 080 090 100   090 100 110 120 130 140 150 160 170 180
011 021 031 041 051 061 071 081 091 101   091 101 111 121 131 141 151 161 171 181
012 022 032 042 052 062 072 082 092 102   092 102 112 122 132 142 152 162 172 182
013 023 033 043 053 063 073 083 093 103   093 103 113 123 133 143 153 163 173 183
014 024 034 044 054 064 074 084 094 104   094 104 114 124 134 144 154 164 174 184
015 025 035 045 055 065 075 085 095 105   095 105 115 125 135 145 155 165 175 185
016 026 036 046 056 066 076 086 096 106   096 106 116 126 136 146 156 166 176 186
017 027 037 047 057 067 077 087 097 107   097 107 117 127 137 147 157 167 177 187

>>>  VALID  <<<
```

Nota Bene:

Diagonal exchanges are not required. It is sufficient to first exchange *left-right* and then *up-down*. As shown in the figure, corner behavior is handled correctly.

Exercise 5: 2D Blocking Communication with Derived MPI Datatype

- `lbm_comm_init_ex5()`:
  - Reuses `lbm_comm_ghost_exchange_ex4()` and, on top of it, creates an MPI derived datatype (`MPI_Type_vector`) for vertical (non-contiguous) communication.
- `lbm_comm_ghost_exchange_ex5()`:
  - *Left-right* communication remains the same as in Exercise 4.
  - *Up-down* communication simplifies: no need to pack and unpack buffers, we just use the new derived MPI datatype.
- `lbm_comm_release_ex5()`: Reuses `ex4` release logic and on top of it frees the derived MPI datatype.

Exercise 6: 2D Non-Blocking Communication (also with Derived MPI Datatype)

- `lbm_comm_init_ex6()` and `lbm_comm_release_ex6()` reuse `lbm_comm_init_ex5()` and `lbm_comm_release_ex5()` respectively.
- `lbm_comm_ghost_exchange_ex6()` looks very similar to `lbm_comm_ghost_exchange_ex5()` but implements non-blocking communication. We use: `MPI_Isend`, `MPI_Irecv` and `MPI_Waitall`. NB: we use `MPI_Waitall` two times to correctly work with diagonal exchanges. For that we run communication in batches: firstly *left-right* and after it's finished we use*up-down*.

Results

For all the above-described exercises we have done checksums and received desired **-:OK** in comparison with sequational implementation of Exercise 0. We also admired pretty gifs and debugging results.