Course *Parallel and distributed programming*
Computer Lab no. 4: More on MPI communications and communicators

IT, Uppsala University

The batch system SLURM

The reservation for Lab 3 on UPPMAX is `uppmax2023-2-13_3`, so you can use it when submitting your jobs to `slurm`.

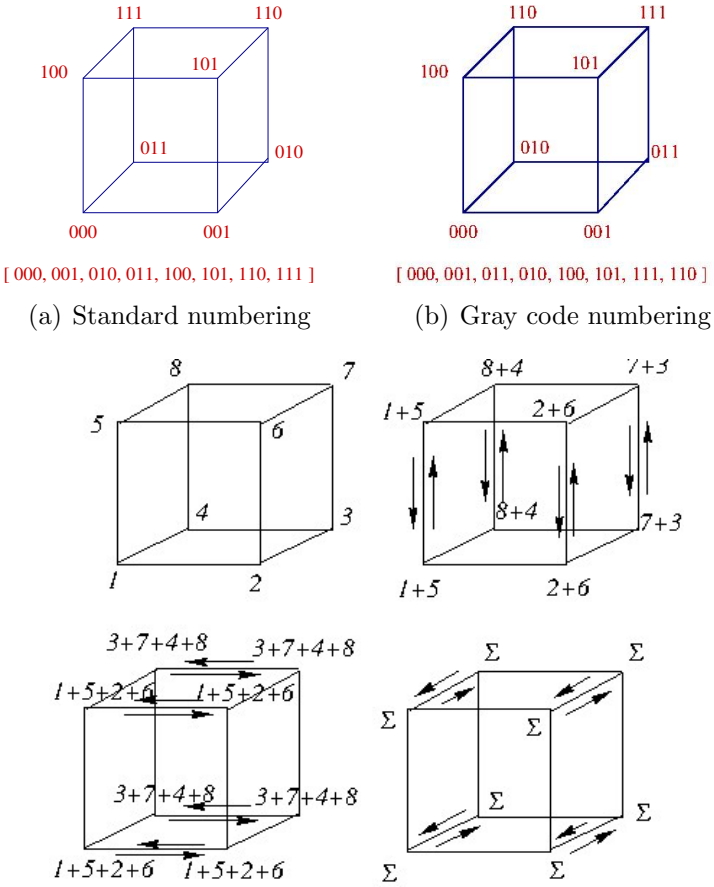
```
#!/bin/bash -l

#SBATCH -M snowy
#SBATCH -A uppmax2023-2-13
#SBATCH --reservation uppmax2023-2-13_3
#SBATCH -p core -n ...
#SBATCH -t xx:00
```

Exercise 1 (Parallel search algorithm (optional, simpler))
Write a code that finds the first zero in a list of N numbers, which are stored as 1D array in a distributed manner.
You might consult with `https://people.sc.fsu.edu/~jburkardt/c_src/search/search.html` and download the serial code `search.c`, compile it and run it. Then you can modify it or write your own code in order to do the task.

Exercise 2 (Recursive communications on a logical hypercube)
Assume you have p processes, where $p = 2^k$, $k = 2, 3, \cdots$, i.e., we assume that the logical topology we work with is a k-dimensional hypercube.
Test the following algorithms to subdivide a k-dimensional hypercube recursively into two $k-1$-dimensional hypercubes.

Task 1 Implement the computation of the scalar product of two distributed vectors of length N on p processes using the so-called Gray code ordering (see the handouts from Lecture 2, Figure 1(a)-1(b) and, for instance, `https://www.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic/part6/faq-doc-1.html`.



(a) Standard numbering (b) Gray code numbering

(c) Scalar product on a 3-D hypercube

Figure 1: Gray code numbering of the processing elements, scalar product

Generate locally two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of length $n = N/p$ and set all the entries to, say, 1, so that the correctness of the code can be easily checked. Perform the local sum $d = \sum_{i=1}^{n} x_i y_i$ on each process. Then exchange-sup up d along each direction of the hypercube as depicted in Figure 1(c). Run a number of experiments enlarging n and k, and observe the time to do the scalar product.

Task 2 Test a recursive splitting of a communicator in the following context. Choose again the number of the PEs as $p = 2^k$. Define locally two variables

```
local_max = (rank+1)*100;
local_min = rank+1;
```

Write a recursive function with the functionality, described in Algorithm 1.

Algorithm 1 Recursive split of Communicators

Input `local_max`, `local_min`, `k`, `communicator`
Determine `rank` and `no_procs`
if "rank=1" **then**
Return
else
Call `MPI_Allreduce` with the functions `MPI_MAX` and `MPI_MIN` to find the max and min of `local_max`, `local_min`.
Print the result together with `rank` and `k`.
Call `MPI_Comm_split`, split the current communicator into two and create a new communicator `halved_communicator`.
Set `k=k-1`
Call the function with `local_max`, `local_min`, `k`, `halved_communicator`.
Free `halved_communicator`.
Return.