Assignment MPI: Matrix Vector Multiplication

The purpose of this assignment is for you to learn more about

- data partitioning and complex collective scheme on matrix multiplication

As usual all time measurements are to be performed on the cluster.

To be able to compile and run an MPI program on Centaurus, **you need to add the line `module load openmpi/4.1.0` at the end of the file `.bashrc` located in the home directory of your account on Centaurus**. (log off and back in afterward.)

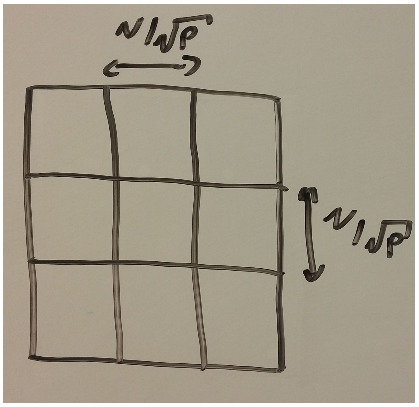
To compile an MPI application, use the `mpicc` compiler in C and the `mpicxx` compiler in C++. They also serve as linker. To run an MPI application using 19 processes, you can run `mpirun -n 19 ./myprogram`. But you will need to have a proper node allocation first. And if you have a proper node allocation then specifying `-n` is not necessary because the cluster scheduler does that for you.

1 Matrix Vector multiplication (50 pts)

The problem is to compute iterated matrix multiplication defined by $x^k = Ax^{k-1}$, where A is a given matrix of size $n \times n$ and x^k is a vector of size n. Pick x^0 as given. (There are functions in the scaffolding generating A and x_0.)

For reference, $x^k = Ax^{k-1}$ is computed using $x^k[i] = \sum_j A[i][j]x^{k-1}[j]$. Or in other words, to compute $x^k[i]$ multiply element wise the ith row of the matrix by x^{k-1} and sum the values.

You should partition the data using blocks:



blocks

Assume that the number of processors is a square number, and that n is divisible by the square root of the number of processors.

Question: Implement iterated matrix multiplication for the block decomposition partitioning scheme. Communicators are a particularly efficient way of implementing this block decomposition. Write the code in `matvec/mpi_matmul.cpp`. This file already contains a sequential implementation of matvec to tell you how to generate the matrix and how to test the multiplication works. You can test your code with `make test`.

For information see

- `http://mpitutorial.com/tutorials/introduction-to-groups-and-communicators/`
- `man MPI_Comm_split`
- `man MPI_Comm_free`

Question: Run the code on Centaurus using `make bench` to queue the jobs. (Note also that some of the matrices are bigger than can be stored on a single node's memory.)
Question: Generate a time table and plots using `make plot`. Do you observe that the code ran on matrices than can not fit in the memory of a single node? Do you observe performance at scale?