Identifying a 2 by 2 submatrix with the largest sum

-Mircea Dumitrache

Programming language + libraries

- C++ programming language (mostly C code)
- MPI library for Xubuntu

Brute Force - Simple idea

- Start off with first square matrix at (0,0)

Brute Force - Simple idea

- Start off with first square matrix at (0,0)
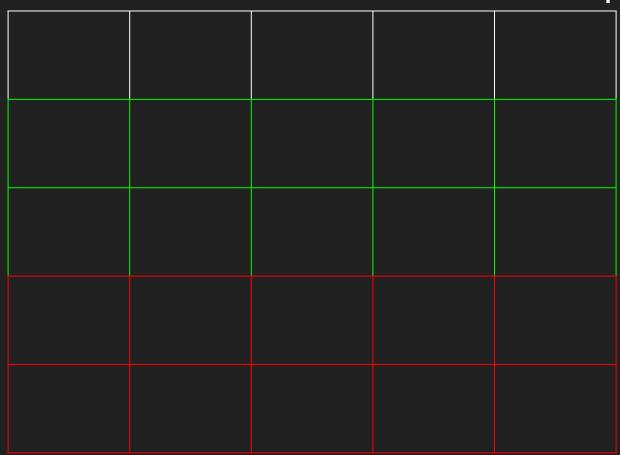- Move one column to right (+1) and computes value of square

Brute Force - Simple idea

- Start off with first square matrix at (0,0)
- Move one column to right (+1) and computes value of square
- Moves one more column to the right etc… until the matrix size - 1

Brute Force - Simple idea

- Start off with first square matrix at (0,0)
- Move one column to right (+1) and computes value of square
- Moves one more column to the right etc… until the matrix size - 1

Then do the same for the rows, repeating the above for each row (except last one).

Decision - Row distribution to nodes first part

Decision - Row distribution to nodes second part
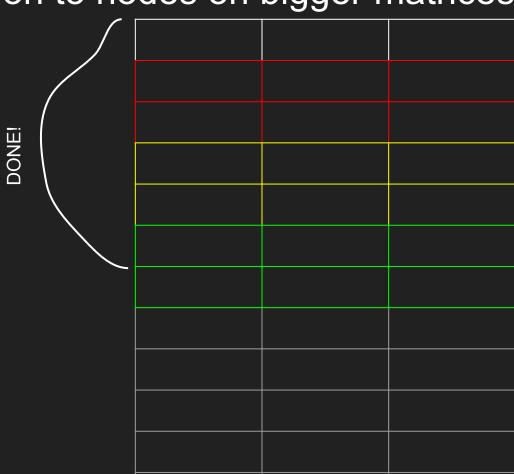
Decision - Row distribution to nodes on bigger matrices

- Assuming 3 nodes
- Note: Matrix not showing all columns.

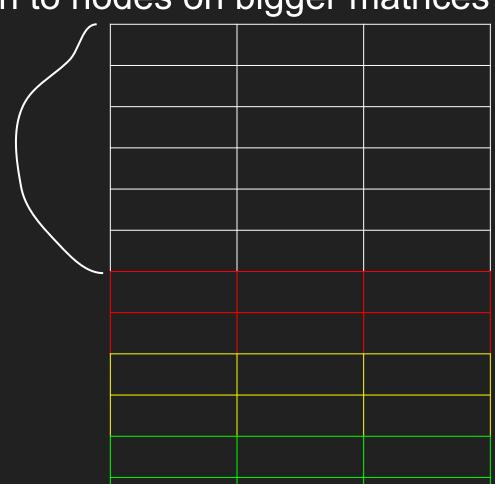# Decision - Row distribution to nodes on bigger matrices

- Assuming 3 nodes
- Note: Matrix not showing all columns.

DONE!

Decision - Row distribution to nodes on bigger matrices

- Assuming 3 nodes
- Note: Matrix not showing all columns.

Difficulties

- Dealing with the matrix as a 2D array.
  - Made 1D array of size^2 instead, such that each row was appended in order to one another.
  - Worked but needed major code restructure.
- Dealing with the end of the iteration, if size is even or odd or when the size is not a multiple of the number of processes.
  - Some nodes attempted to to compute rows that do not exist.

Unsolved difficulties

- When buffer is sent all rows except for first and last are buffered twice.

Solution:

- Get the smallest multiple of the number of nodes that is higher than the size. Divide up the entire data set among all nodes, rather than section of 2 rows at a time.
- Will still need a clean up section to compute last + first row of every node (except root and last node).

Shortcoming

- Sometimes the row value is off by one.

The end.

Thank you for listening!