# High-Performance Computing Lab for CSE

## Project 4 - Parallel Programming using the Message Passing Interface MPI

Due date: 26 April 2021, midnight.

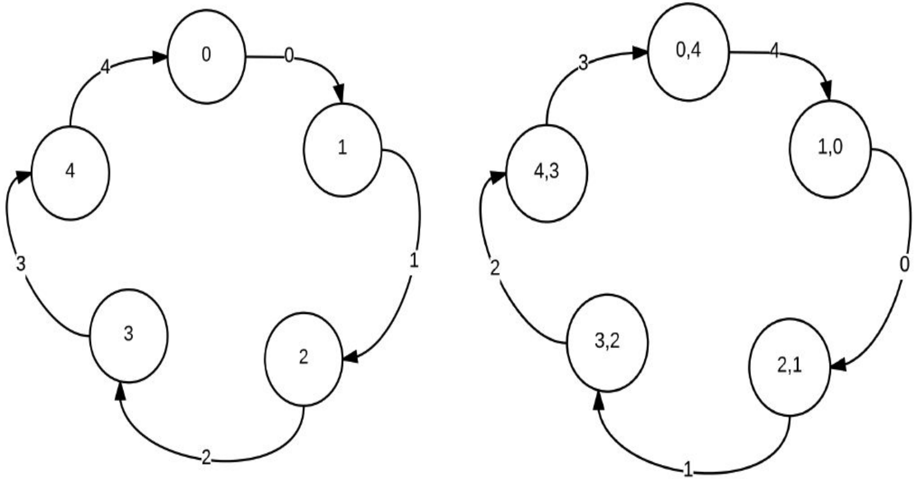
Ignacio Labarca

# Project 4: Parallel Programming using MPI

1. Ring addition using MPI (10 Points).
2. Ghost cells exchange (15 Points).
3. Mandelbrot set (20 Points).
4. Option A: Parallel matrix-vector multiplication and the power method (40 Points).
5. Option B: Parallel PageRank Algorithm and the power method (40 Points).

# Ring addition using MPI

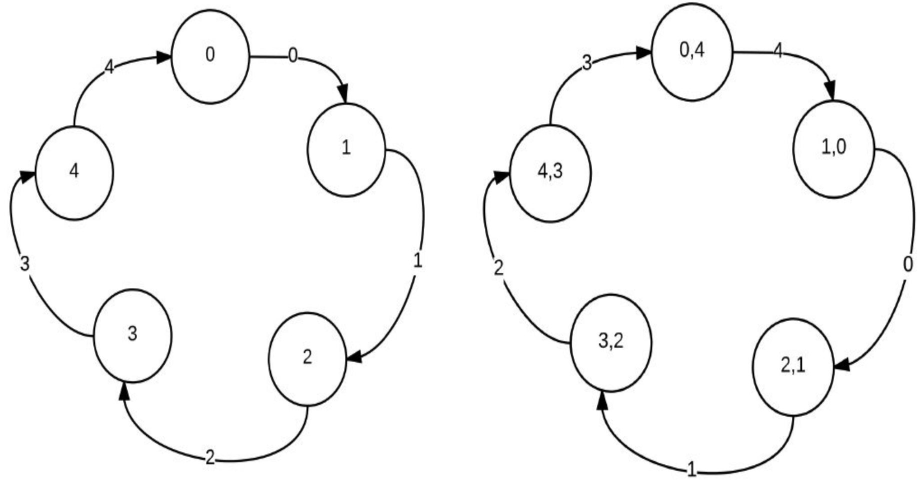
- Each process:
  Receives from the left, sends to the right.




First two iterations of the ring sum algorithm.

# Ring addition using MPI

- After n iterations, each process has the sum of all ranks.




First two iterations of the ring sum algorithm.

# Ring addition using MPI

Solve the following tasks:

- Determine left/right neighbors of each process.
- Implement a ring addition code.

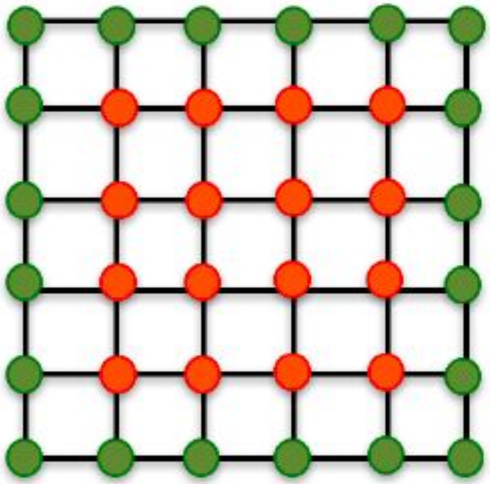
Result of the parallel computation with n = 4 should be:

```
Process 0:        Sum = 6
Process 1:        Sum = 6
Process 2:        Sum = 6
Process 3:        Sum = 6
```

# Ghost cells exchange

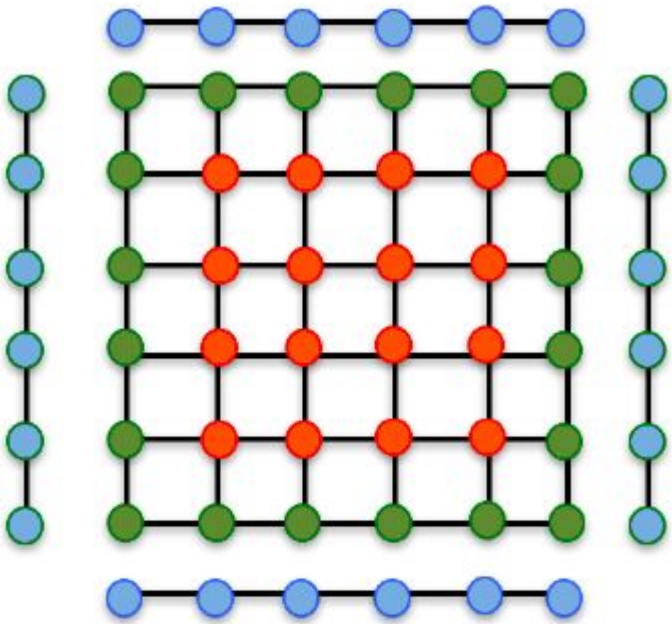
- Each process has a 6x6 local domain.

- The borders must be sent to the neighbors (top, bottom, left, right).



# Ghost cells exchange

- To handle the copies, we create ghost cells.
  Extended domain of size (6 + 2) x (6 + 2)



# Ghost cells exchange

- We assume an n x n grid of processes in a Cartesian topology.

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

# Ghost cells exchange

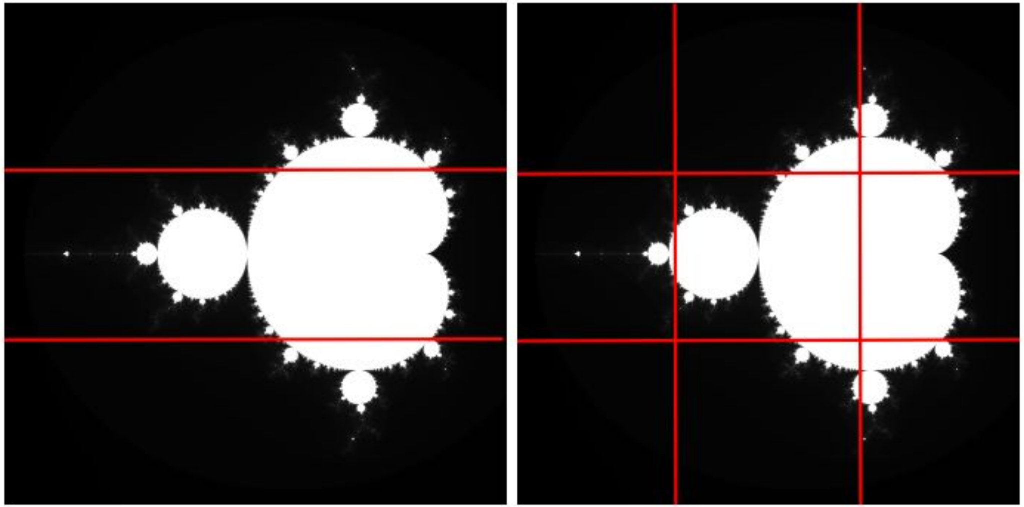
Solve the following tasks:

- Create a Cartesian 2-dimensional communicator with periodic boundaries.
- Create a derived data type for a column border.
- Exchange ghost cells with neighboring cells and verify correctness.
- Work with 16 processes.

# Mandelbrot Set

Create partitioning of the domain.

Each process computes its local portion of Mandelbrot set.



# Mandelbrot Set

Solve the following tasks:

- Implement createPartition and updatePartition.
- Implement createDomain.
- Send the local domain to the master process.
  Compare the output with sequential program.
- Analyze the performance.
  Discuss about benefits and load balancing observed.

# Parallel matrix-vector multiplication and the power method

The power method is used to find the largest eigenvalue of a matrix.

```
[n,n] = size(A);
x = rand(n,1);
for i = 1:1000
    x = x / norm(x);
    x = A * x;
end;
lambda = norm(x);
```

# Parallel matrix-vector multiplication and the power method

Your task is to write a parallel C/MPI code that does the same computation.

**What to implement**

- generateMatrix
- powerMethod. Calls norm and matVec.
- main. Calls generateMatrix and powerMethod.

# Parallel matrix-vector multiplication and the power method

Your task is to write a parallel C/MPI code that does the same computation.

**Where's the data**

Assume that n, the number of rows and columns is divisible by p, the number of processors.

Each processor should generate its own rows of the matrix.

# Parallel matrix-vector multiplication and the power method

**What experiments to do**

Strong scaling analysis.

Fix a value for n. Run your code for p = 1, 4, 8, 12, 16, 32, 64.

Report running time and parallel efficiency.

Weak scaling analysis.

Choose n proportional to sqrt(p). In theory, work remains constant as you increase p.

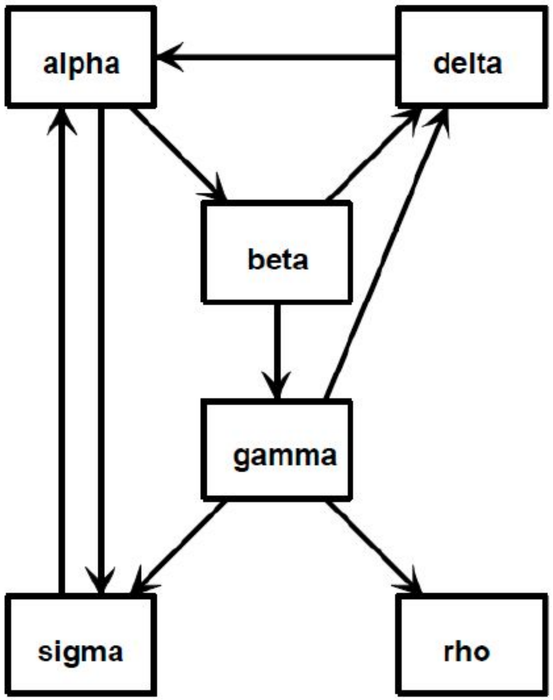
Report running time and parallel efficiency.

# Parallel PageRank Algorithm and the power method

PageRank Algorithm.

A random surfer goes through the websites.

What is the probability of going from one to the other?



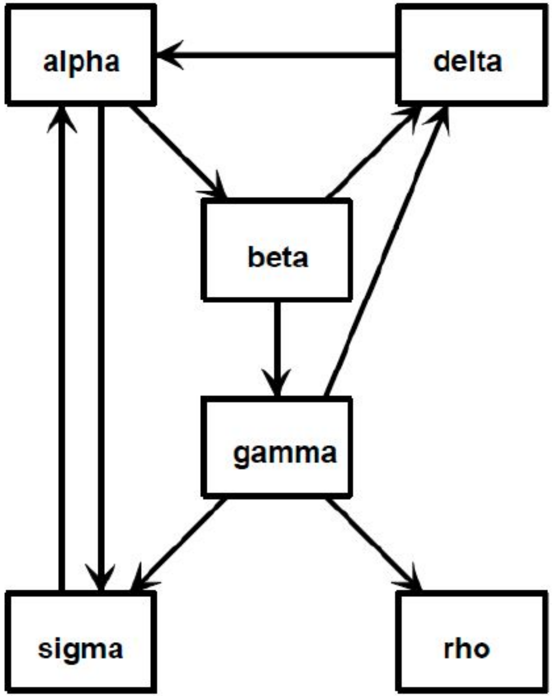
# Parallel PageRank Algorithm and the power method

PageRank Algorithm.

Ranking websites according to the state vector of a Markov chain:

$$x = Ax$$

The largest eigenvalue of A is 1.



# Parallel PageRank Algorithm and the power method

Sparse representation of matrix A

$$A = pGD + ez^T$$

Power method can be written as

```
G = p*G*D
z = ((1-p)*(c~=0) + (c==0))/n;
while termination_test
x = G*x + e*(z*x)
end
```

# Parallel PageRank Algorithm and the power method

A serial implementation is provided.
Your task is to make the necessary MPI changes to obtain a parallel implementation.

# Parallel PageRank Algorithm and the power method

Benchmark your code with the three Stanford Network Dataset Collection (SNAP) examples provided.

Different sparsity patterns. Comment on the results obtained.

Computation, synchronization and communication time.