MPI- Collective Communication

Tutorial

Lec 4

Agenda

- Collective Communication
- Logistic Regression
- Parallel Logistic Regression with MPI

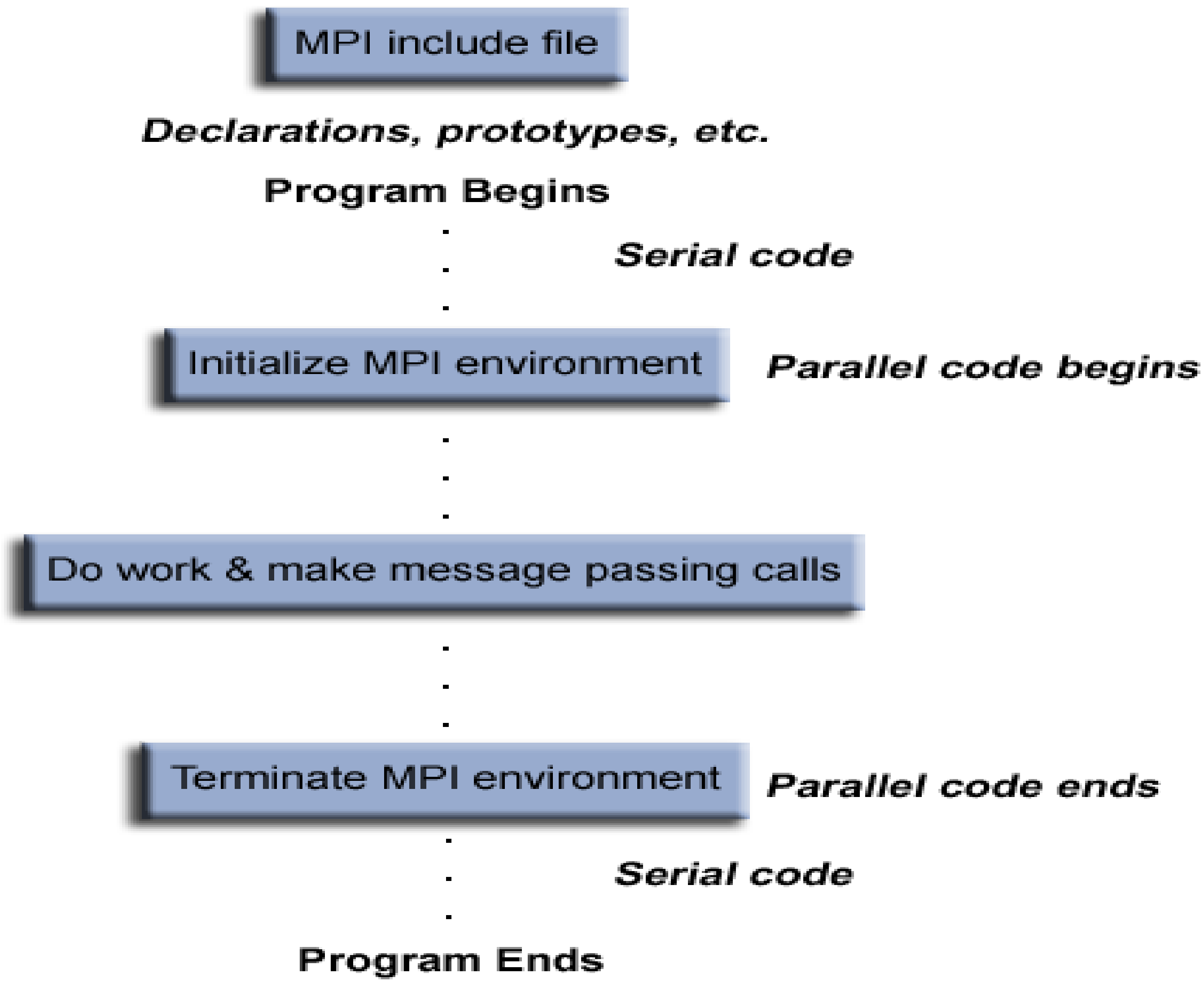
Compute Cluster

- Account at ISMLL cluster.
  - I need your name and email id

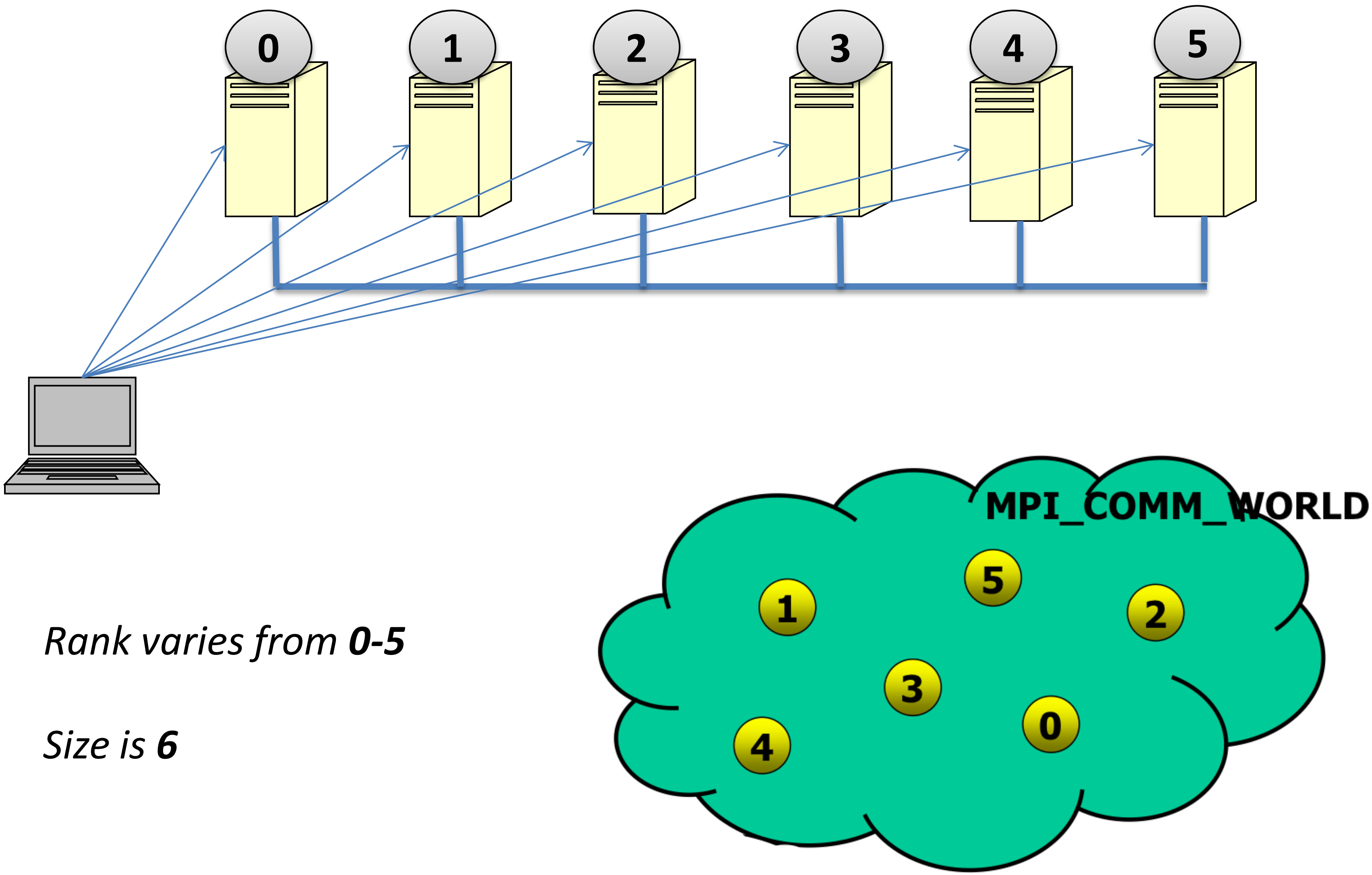
Agenda

- Collective Communication
- Logistic Regression
- Parallel Logistic Regression with MPI

General MPI program structure



MPI Execution Model



*Rank varies from **0-5***

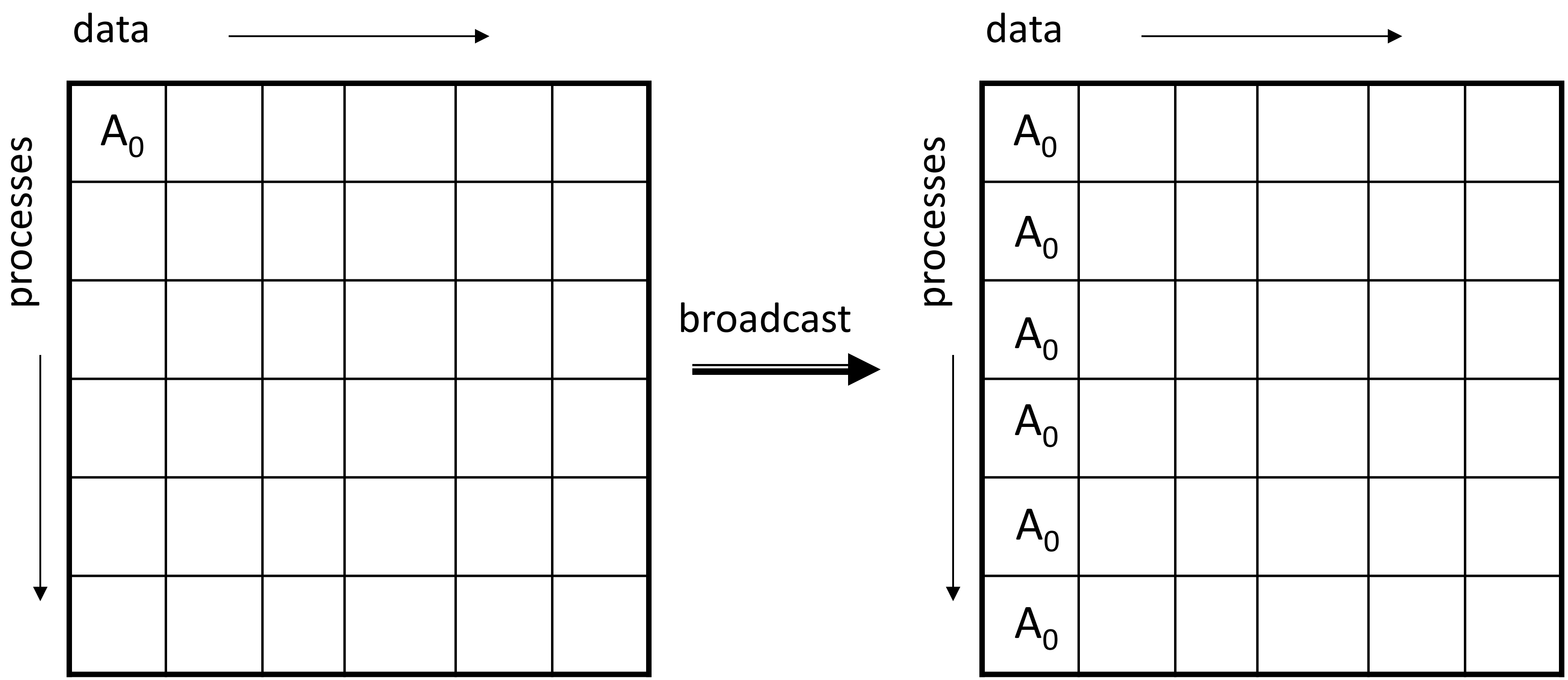
*Size is **6***

Hello World in Java

```
import java.util.*;
import mpi.*;

public class HelloWorld {

   public static void main(String args[]) throws Exception {

   // Initialize MPI
   MPI.Init(args); // start up MPI

      // Get total number of processes and rank
   size = MPI.COMM_WORLD.Size();
   rank = MPI.COMM_WORLD.Rank();

   System.out.println(“Hello World <”+rank+”>”);

   MPI.Finalize();

     }
}
```

Bcast(java.lang.Object buf, int offset, int count, Datatype type, int root)

MPI_Bcast

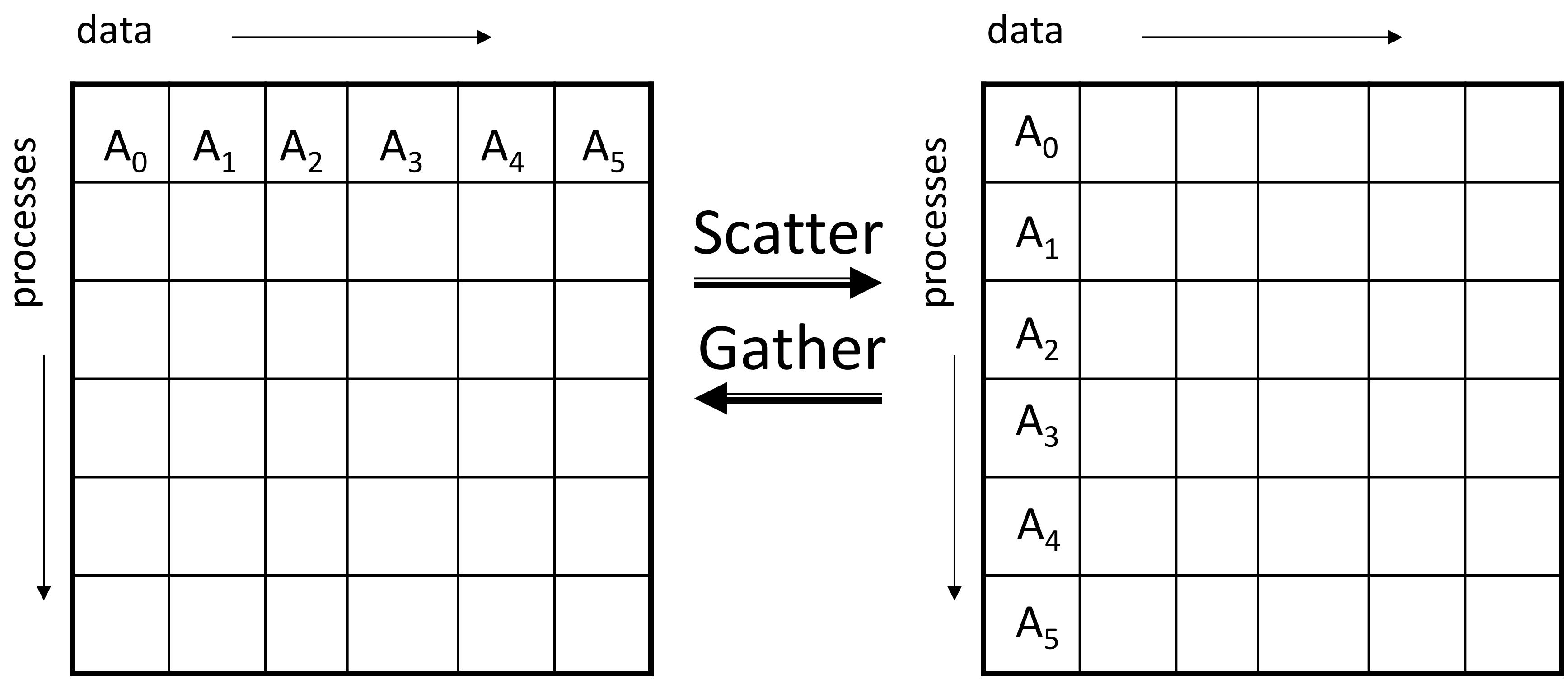


A_0 : any chunk of contiguous data described with MPI_Type and count

Gather(java.lang.Object sendbuf, int sendoffset, int sendcount, **Datatype** sendtype,
java.lang.Object recvbuf, int recvoffset, int recvcount, **Datatype** recvtype, int root)

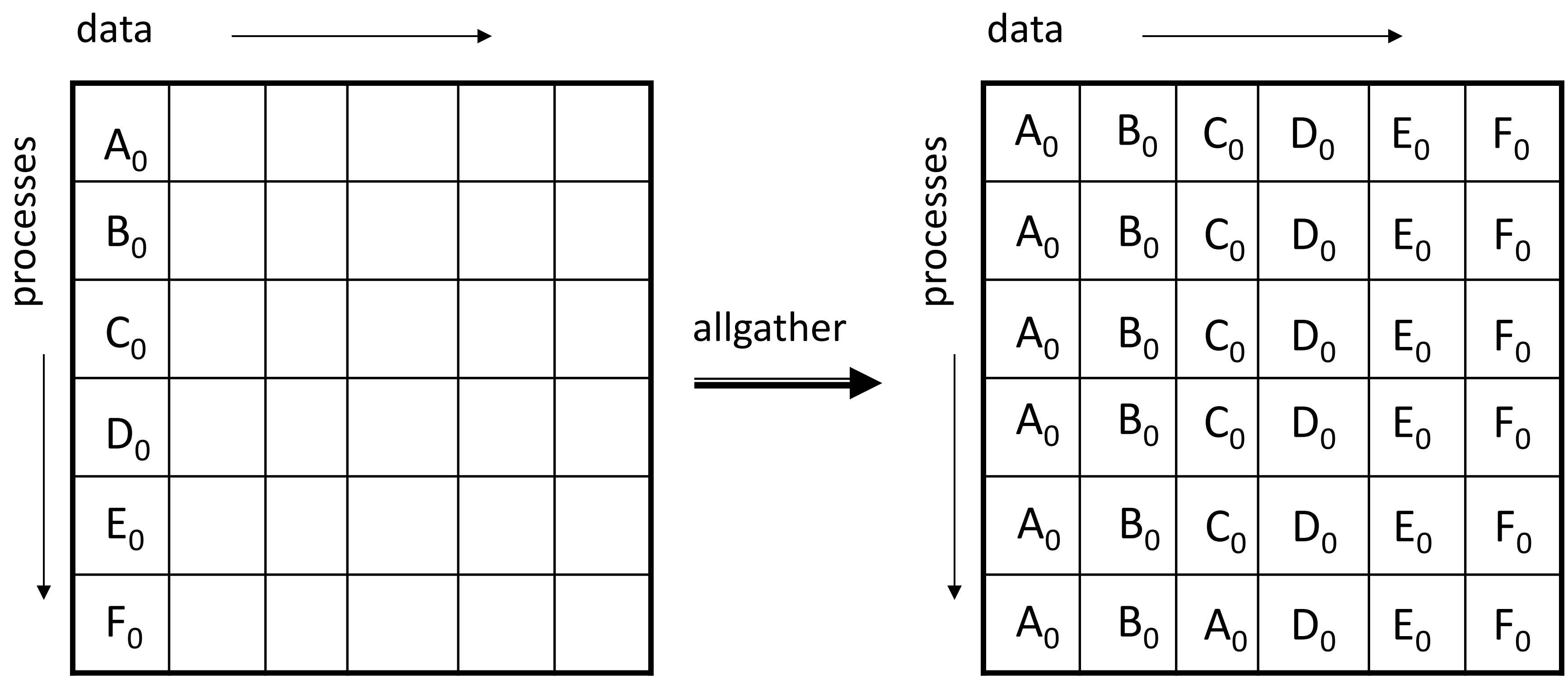
Scatter(java.lang.Object sendbuf, int sendoffset, int sendcount, **Datatype** sendtype,
java.lang.Object recvbuf, int recvoffset, int recvcount, **Datatype** recvtype, int root)

MPI_Scatter
MPI_Gather



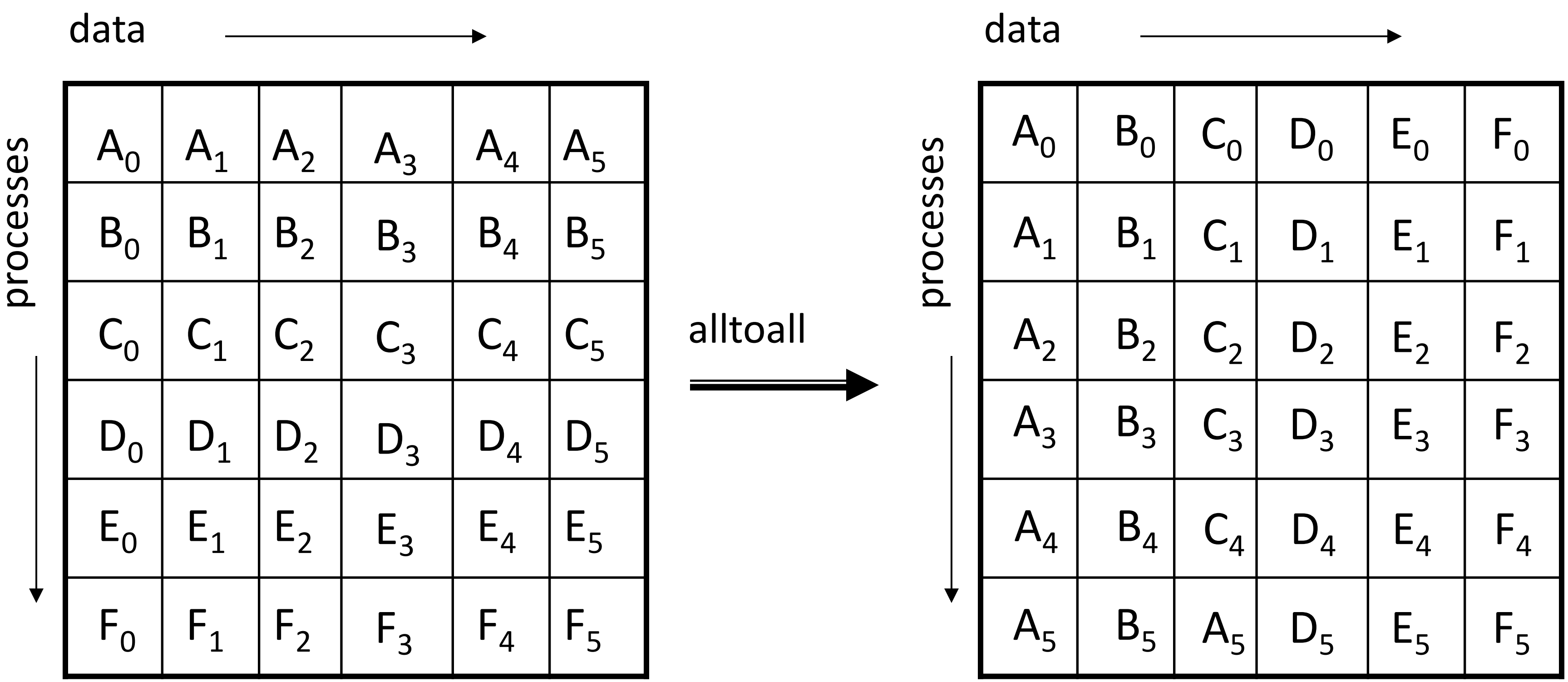
Allgather(java.lang.Object sendbuf, int sendoffset, int sendcount, **Datatype** sendtype, java.lang.Object recvbuf, int recvoffset, int recvcount,**Datatype** recvtype)

MPI_Allgather



Alltoall(java.lang.Object sendbuf, int sendoffset, int sendcount, **Datatype** sendtype, java.lang.Object recvbuf, int recvoffset, int recvcount,**Datatype** recvtype

MPI_Alltoall



Allreduce(java.lang.Object sendbuf, int sendoffset, java.lang.Object recvbuf, int recvoffset, int count, **Datatype** datatype, **Op** op)

Reduce(java.lang.Object sendbuf, int sendoffset, java.lang.Object recvbuf, int recvoffset, int count, **Datatype** datatype, **Op** op, int root)

Reduce/Allreduce

A0	B0	C0
A1	B1	C1
A2	B2	C2

reduce ⟹

A0+A1+A2	B0+B1+B2	C0+C1+C2

A0	B0	C0
A1	B1	C1
A2	B2	C2

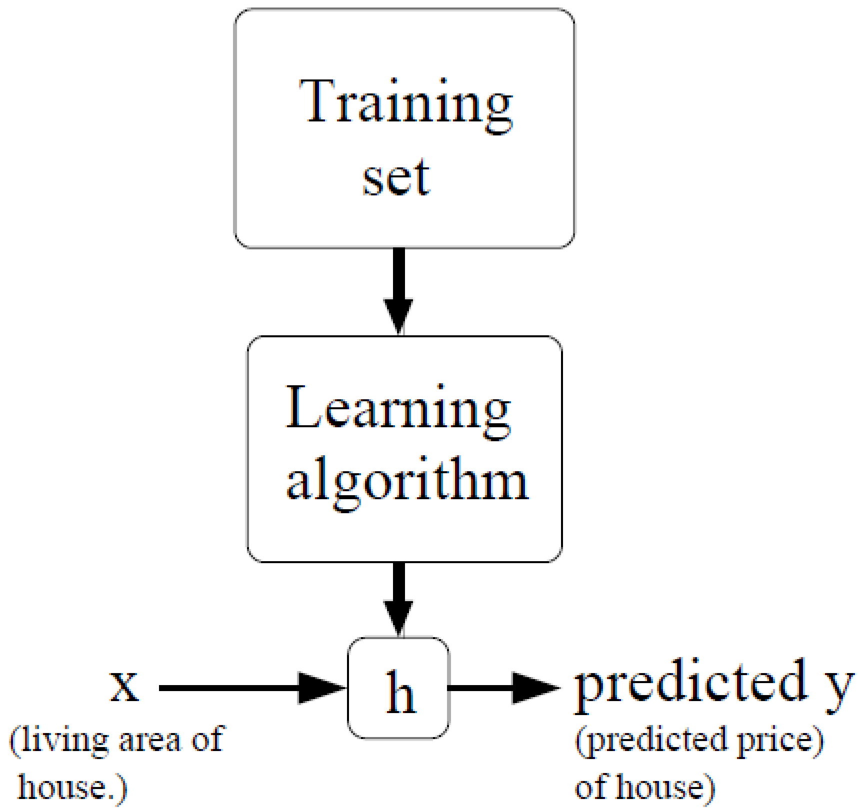
allreduce ⟹

A0+A1+A2	B0+B1+B2	C0+C1+C2
A0+A1+A2	B0+B1+B2	C0+C1+C2
A0+A1+A2	B0+B1+B2	C0+C1+C2

Agenda

- Collective Communication
- Logistic Regression
- Parallel Logistic Regression with MPI

Machine learning



Regression

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Reference:
Andrew NG
http://cs229.stanford.edu/notes/cs229-notes1.pdf

Linear model: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ $\qquad h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$

Loss function: $J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2.$

Learning algorithm

Loop {

for i=1 to m, {

$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$ (for every j).

}

}

Classification

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	setosa
2	4.90	3.00	1.40	0.20	setosa
3	4.70	3.20	1.30	0.20	setosa
⋮	⋮	⋮	⋮	⋮	
51	7.00	3.20	4.70	1.40	versicolor
52	6.40	3.20	4.50	1.50	versicolor
53	6.90	3.10	4.90	1.50	versicolor
⋮	⋮	⋮	⋮	⋮	
101	6.30	3.30	6.00	2.50	virginica
⋮	⋮	⋮	⋮	⋮	
150	5.90	3.00	5.10	1.80	virginica

Logistic regression:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}, \qquad \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

Andrew NG
http://cs229.stanford.edu/notes/cs229-notes1.pdf

Learning algorithm

Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

}

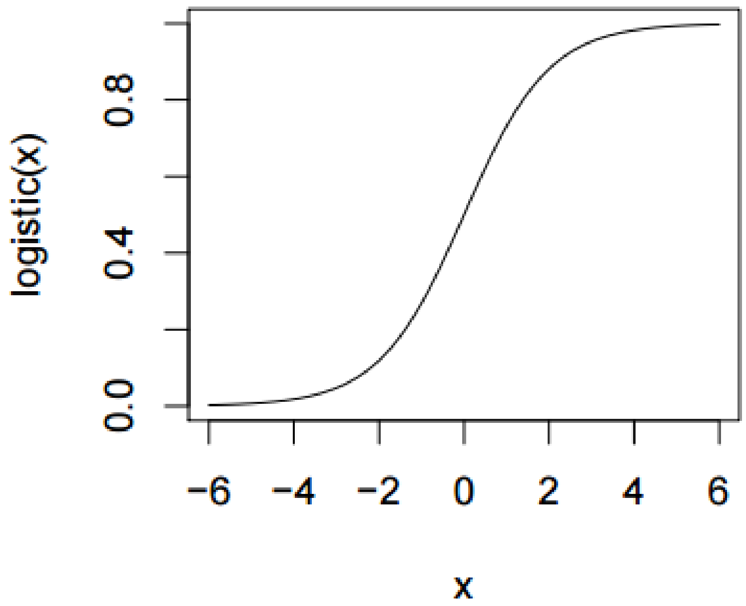
Logistic Function

Logistic function:

$$\text{logistic}(x) := \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

The logistic function is a function that

- has values between 0 and 1,
- converges to 1 when approaching $+\infty$,
- converges to 0 when approaching $-\infty$,
- is smooth and symmetric at $(0, 0.5)$.



Agenda

- Collective Communication
- Logistic Regression
- Parallel Logistic Regression with MPI

Logistic Regression with MPI

- To parallelize Logistic Regression with SGD algorithm following steps are needed:
- Split train and test data among workers
- **train()**
  1. Initialize model parameter (weights in code or theta in algorithms) with zeroes or uniform random.
  2. Repeat until convergence
     a. Iterate over all training examples and update weight
     b. Averaging weights from all the workers
- **evaluate()**
  1. Initialize error to zero
  2. Iterate over all test examples and predict score
     a. Calculate RMSE
  3. Average RMSE score from all the workers