BIG DATA

MapReduce Algorithms – Matrix Multiplication

K V Subramaniam
Computer Science and Engineering

PES UNIVERSITY ONLINE

Matrices and Vectors - introduction

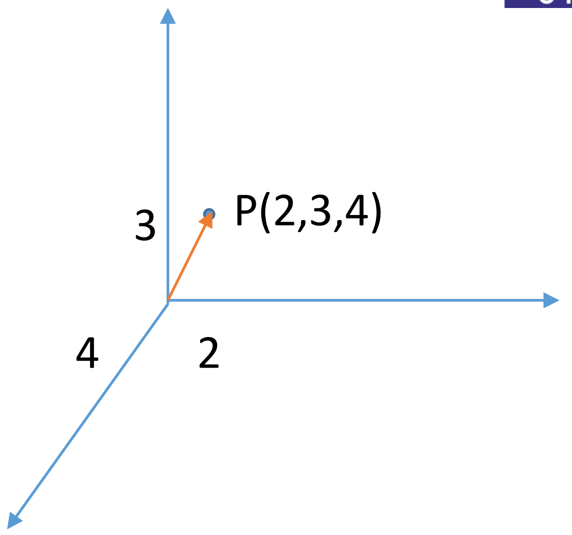
Overview

- Matrix Multiplication algorithms
  - Fundamental to many computations, including Page Rank

- Source
  - Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.
  - http://infolab.stanford.edu/~ullman/mmds/book.pdf
  - 4.3.2 of T1

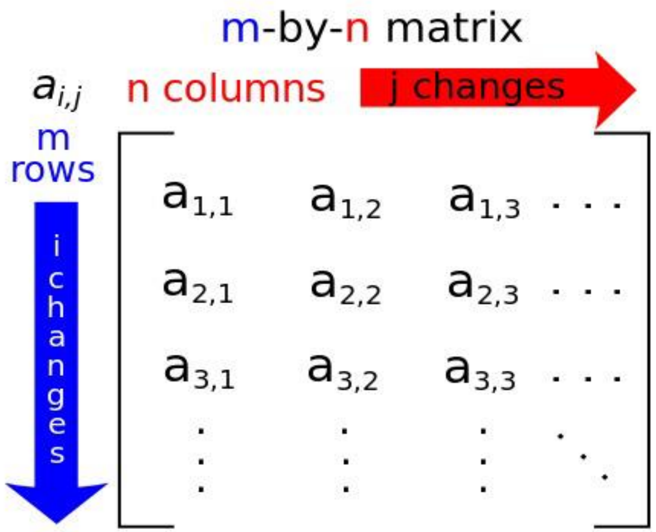
Background: Vectors and Matrices

- Vectors
  - Can be defined as an ordered list of numbers
  - Visualization
    - An arrow where the direction of the vector is given by the relative size of the components
- Some Common Operations
  - Addition: ***v***+*w*
    - Add components
  - Scalar multiplication *a**v***
    - Multiply each component by constant



Matrix

- Rectangular array of numbers.
- The numbers are called the elements of the matrix.
- An *mxn* matrix has *m* rows and *n* columns
  - Can be considered as a collection of
    - *m* row vectors
    - *n* column vectors
  - An *nxn* matrix is called a square matrix.
- Vector can be considered as a
  - *1xn* matrix (row matrix)
  - *nx1* matrix (column matrix)



Each element of a matrix is often denoted by a variable with two subscripts. For example, $a_{2,1}$ represents the element at the second row and first column of a matrix **A**.

Matrix Vector Multiplication – Definition

- Multiply each row vector of ***A*** by the corresponding elements of ***x*** and sum
- Multiplying a *mxn* matrix by an *n* element vector gives an *m* element vector

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}.$$

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 3 & 1 & 2 \end{bmatrix}. \qquad \mathbf{x} = (x, y, z)$$

$$A\mathbf{x} = \begin{bmatrix} 1 & 0 & -1 \\ 3 & 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x - z \\ 3x + y + 2z \end{bmatrix}$$

$$= (x - z, 3x + y + 2z).$$

Traditional Representation of Matrices

- Typically, matrices are stored as multi-dimensional arrays in programs
- int A[10][10]

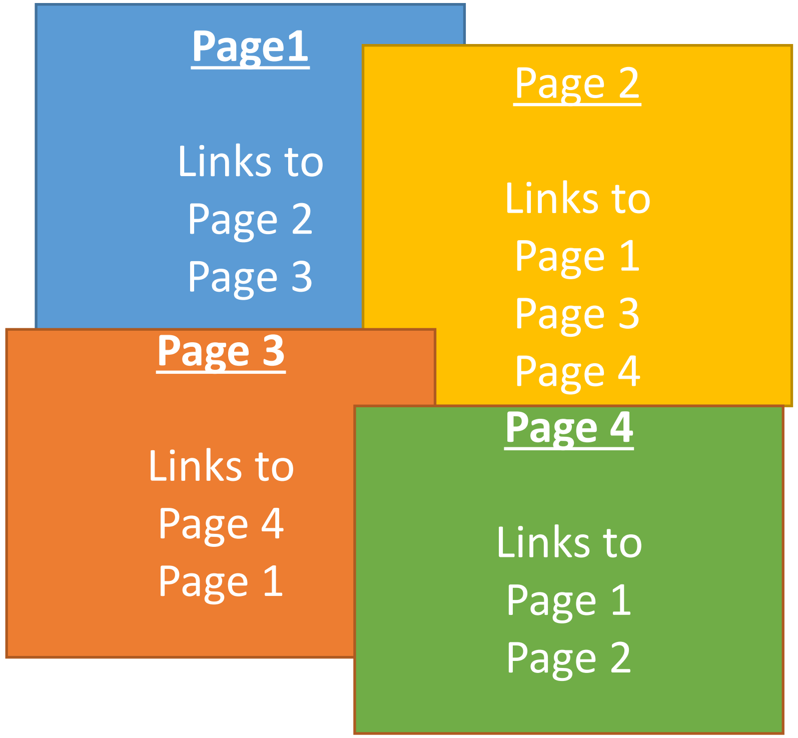
  allocates 100 integers and is accessed as a 10x10 matrix

Space required to store the matrix – =
10x10*sizeof(int)= 100*sizeof(int)=100*4 = 400 bytes.

In general, we need n^2 integers to store an ***nxn*** matrix

Matrix Representation of WWW

How do you represent the pages in WWW?

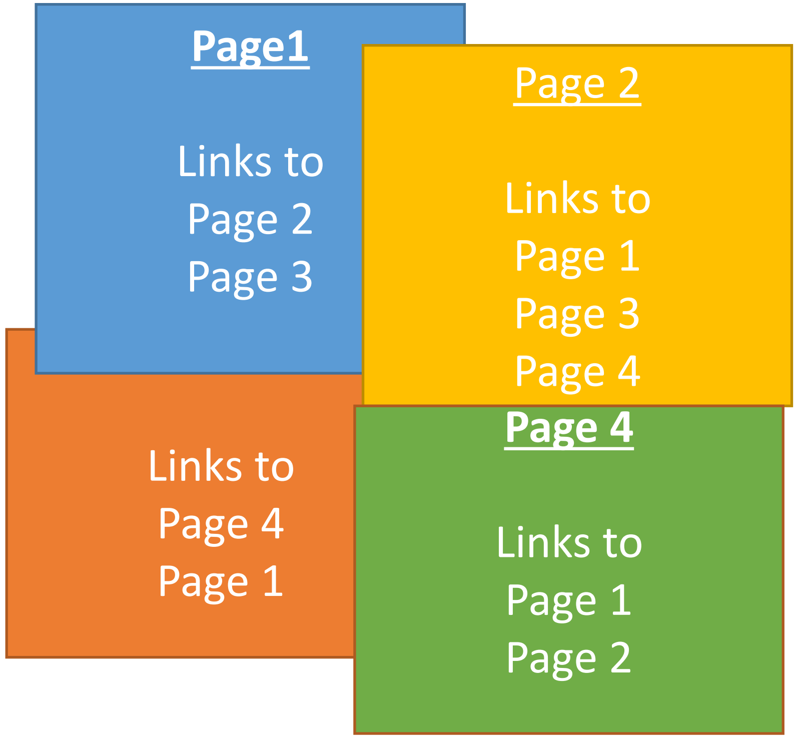


Pages in the WWW

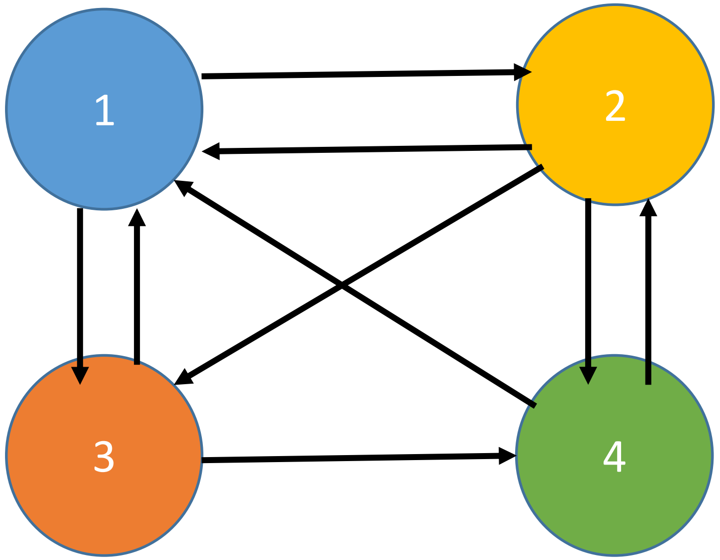
Consider a sample of the internet that contains 4 pages

- How should we represent this?

Modelling the WWW as a directed graph

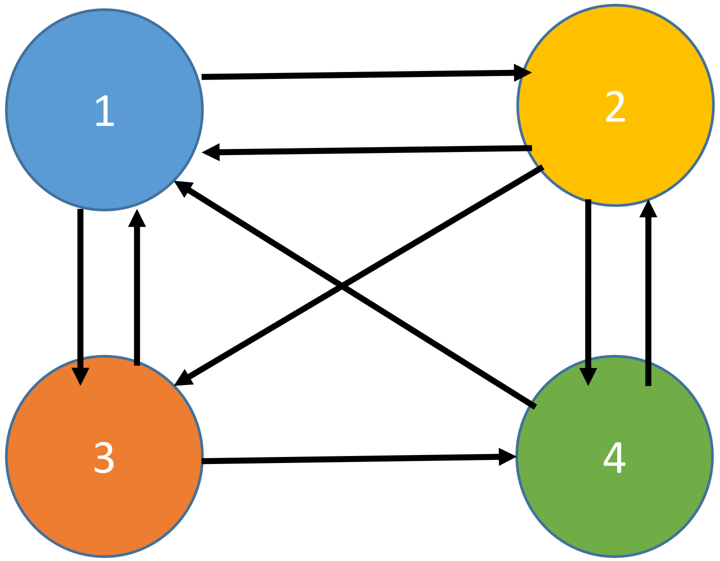


Pages in the WWW



Represented as a directed graph

Representing the graph as an Adjacency Matrix



Directed Graph

Source →

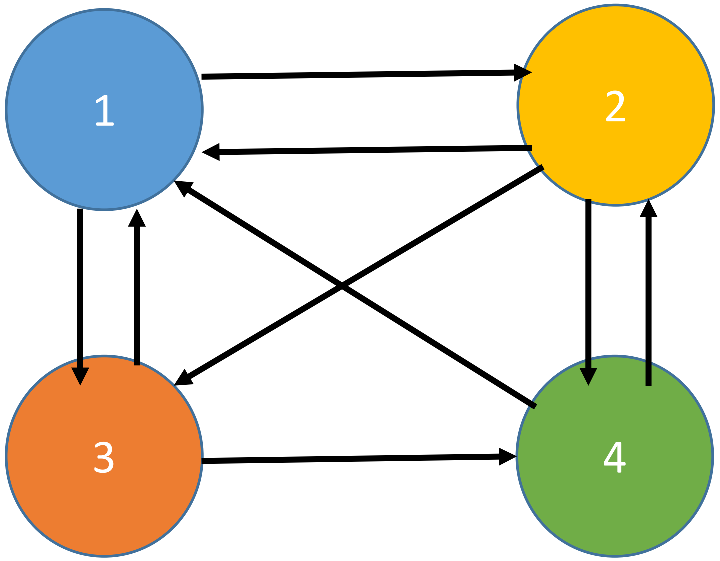
Dest ↓

0	1	1	1
1	0	0	1
1	1	0	0
0	1	1	0

- This is fine for a small graph – 4 pages.
- But internet is large – billions of pages.
- How much storage will we require?

Large scale matrix representations

Representing the graph as an Adjacency Matrix



Directed Graph

Source →

Dest ↓

0	1	1	1
1	0	0	1
1	1	0	0
0	1	1	0

- Internet is large – billions of pages.
- Note – most of the entries will be 0
- Store as a <u>***sparse***</u> matrix..

Sparse Matrix representation

- In Big Data, we deal with large matrices
  - e.g, n will be the order of 10^{10} if n is number of web pages
- And it will be a sparse matrix
- Won't fit in the memory (DRAM)
- Have to store it in HDFS

HDFS Sparse matrix representation

- Store only non-zero elements as a separate record in CSV format

- For each element store
  - *<row_number, column_number, value>*
  - As the format

- As many entries as there are links

- Exercise –Store the graph given on the right into a HDFS CSV file

0	1	1	1
1	0	0	1
1	1	0	0
0	1	1	0

Solution

1, 2, 1
1, 3, 1
1, 4, 1
2, 1, 1
2, 4, 1
3, 1, 1
3, 2, 1
4, 2, 1
4, 3, 1

0	1	1	1
1	0	0	1
1	1	0	0
0	1	1	0

As an exercise, try saving this in a file and loading it onto HDFS that you have installed.

Matrix Vector Multiplication

Matrix Vector multiplication with MapReduce

- To multiply an *nxn* matrix ***M*** with an *n*-element vector ***v***, compute

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \end{bmatrix}$$

Matrix Vector Multiplication with MapReduce

- Let us assume that the vector ***v*** fits into memory
- Vector v is shared by all the mappers
- M_{ij} is stored as a CSV file on HDFS and is distributed across multiple nodes

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \boxed{a_{21}} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} \boxed{x_1} \\ x_2 \\ \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ \boxed{\boxed{a_{21}x_1} + a_{22}x_2 + \ldots + a_{2n}x_n} \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \end{bmatrix}$$

2

Matrix Vector Multiplication with MapReduce

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

- **map**:
  - Computes the partial product
  - Uses the key as i → the index into the target vector
  - output (i, $m_{ij}v_j$)
- **reduce**:
  - Sums all the partial products.

Working of the MR algorithm – Map Stage

Matrix

0	1	1	1
1	0	0	1
1	1	0	0
0	1	1	0

Vector

5
3
4
2

Mapper 1

Key	Value
1	1*3 = 3
2	1*5 = 5
3	1*5= 5
3	1*3=3
4	1*3=3

Mapper 2

Key	Value
1	1*4=4
1	1*2=2
2	1*2=2
4	1*4=4

Note that the key is the index into the vector where this value will contribute

Working of the MR algorithm – Reduce Stage

Matrix

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Vector

$$\begin{bmatrix} 5 \\ 3 \\ 4 \\ 2 \end{bmatrix}$$

Mapper 1 | Mapper 2

Reducer Input

Key	Intermediate Value List
1	2,3,4
2	2, 5
3	3, 5
4	3, 4

Reducer output

Key	Value
1	9
1	7
2	8
4	7

Review Problem

Matrix

$$\begin{bmatrix} 0 & 0 & 3 & 8 \\ 0 & 9 & 5 & 0 \\ 0 & 10 & 0 & 0 \\ 5 & 0 & 0 & 1 \end{bmatrix}$$

Vector

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

- Assuming rows 1 and 3 are in datanode1 and 2 and 4 are in datanode 2, perform a matrix multiplication using Map Reduce
- Show inputs/outputs of mappers/reducers as dicussed in previous slides

Matrix Vector Multiplication - extensions

Matrix-Vector Multiplication using Mapreduce - 2

- Case 2: ***v*** doesn't fit into main memory.
- Partition *M* and ***v*** into stripes.

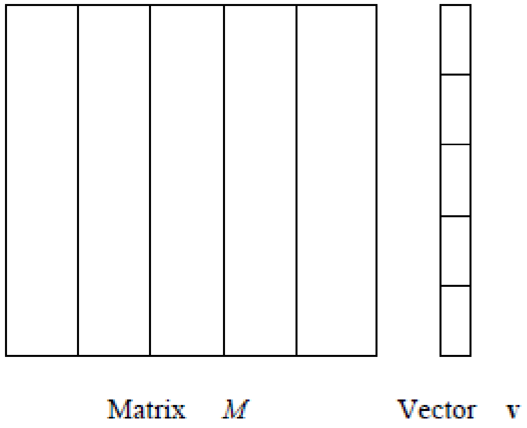


Figure 2.4: Division of a matrix and vector into five stripes

- The same MapReduce algorithm can be used.

THANK YOU

K V Subramaniam
Dept. of Computer Science and Engineering
subramaniamkv@pes.edu,
ushadevibg@pes.edu