BIG DATA

Big Data Algorithm Complexity

K V Subramaniam
Computer Science and Engineering

Overview of lecture

- Motivation – Algorithm complexity
- Communication Cost Complexity Model
- 3 Way joins with the communication cost complexity
- Key parameters
- Similarity join - analysis

Motivation – Algorithm Complexity

Why Study complexity?

- So far, we have looked at MapReduce algorithms
- However, for a particular problem, there could be many algorithms
- Which algorithm should we choose?
- This is why we study complexity of MapReduce
- We will actually study complexity of workflow systems
    - Generalization of MapReduce
    - Many important Big Data systems are workflow systems

Exercise

- Consider the following two problems
  - Matrix multiplication
  - Database query
- What would be the complexity of these algorithms when executing on a single node?
- What does complexity depend on?

Solution: Class Exercise

- Matrix multiplication
  - Expressed in terms of the bound on total #computations performed
- Database query
  - Complexity depends on disk read

Communication Cost Complexity

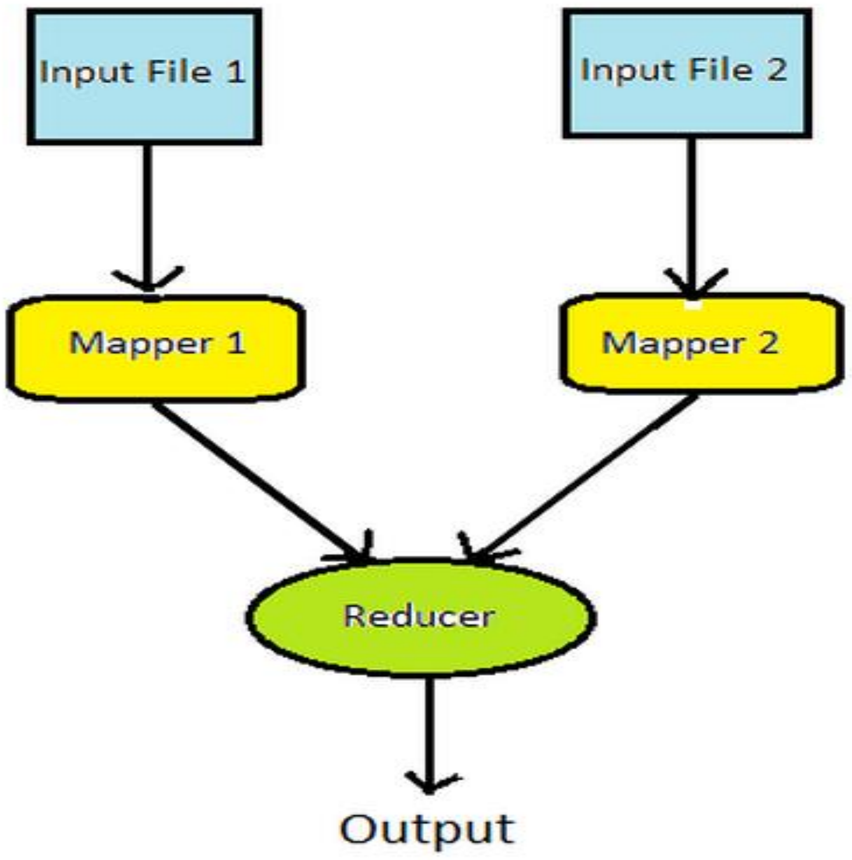
- Communication cost: size of input
- Why communication cost?
  - Algorithm tends to be linear in data
  - Network speed << CPU speed
  - Disk speed << CPU speed
  - Major time could be communication time

Communication Cost Complexity

- Why only input size?
  - Output is input to some other task
  - Final output is generally small by aggregation
    - Otherwise not human-readable

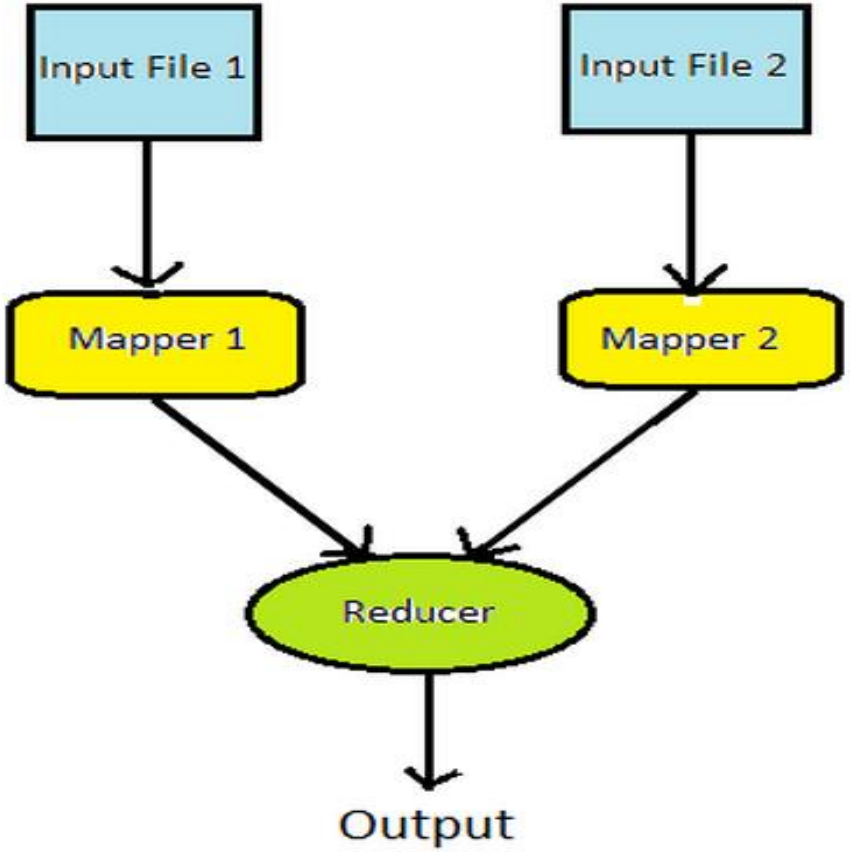
Natural Join: join R, S

- Mapper input complexity = *r+s*
  - Read data from disk
- Reducer input complexity = *r+s*
  - Network reads
- Total Complexity: *O(r+s)*



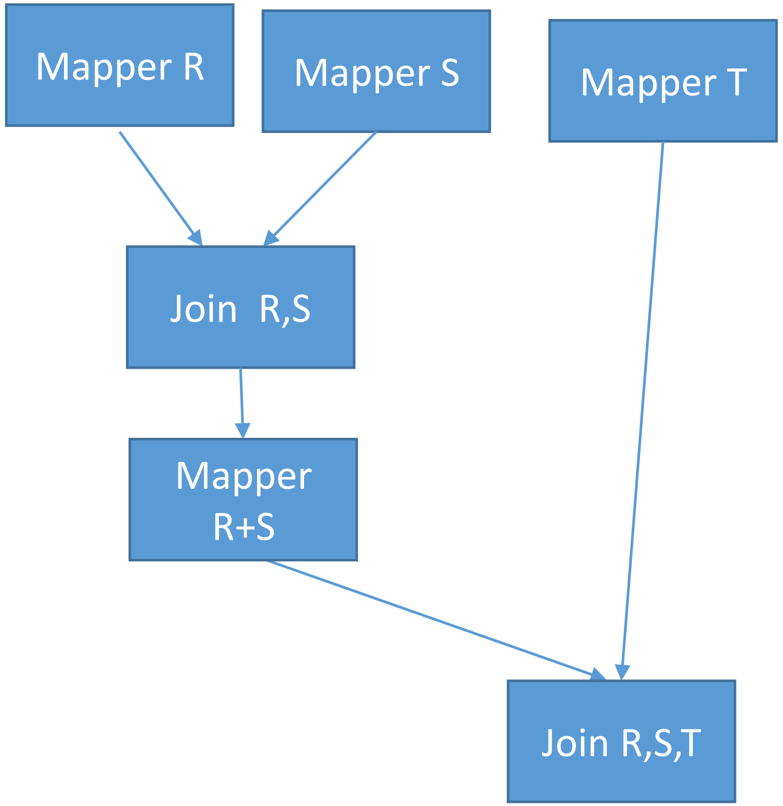
Exercise – Join complexity

- Consider three relations R, S and T
- How will you perform a join using map reduce across the three?
- Estimate the complexity



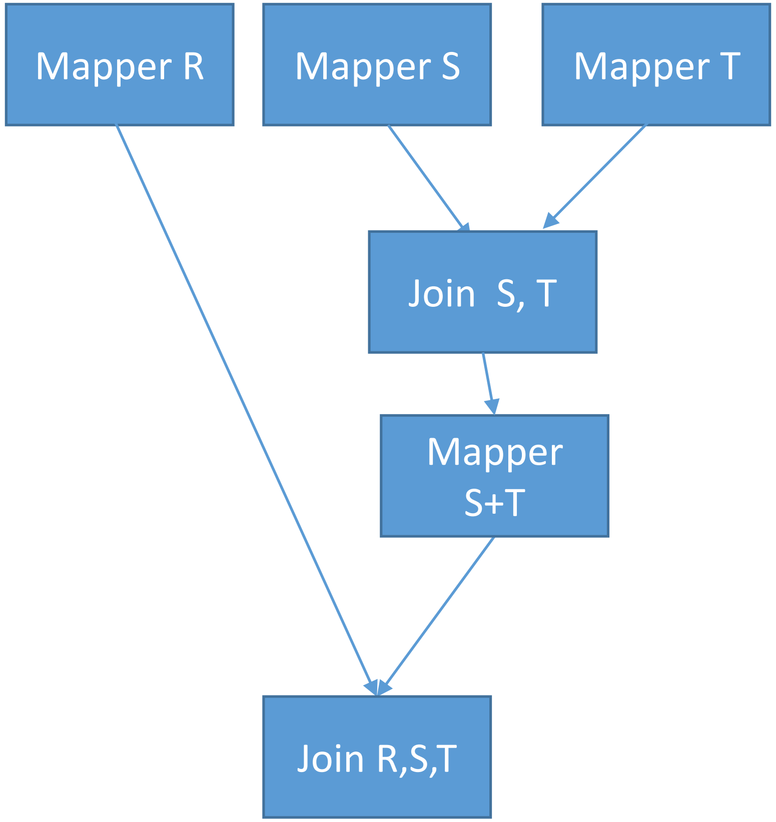
3-way Join: R, S, T

- 2 MapReduce phases
- Case 1: Join *R,S* and then join *T*
  - Input to Mapper 1: *r*
  - Input to Mapper 2: *s*
  - Input to Reducer 1: *r+s*
  - Let *p* be the probability of match between *r,s*
  - Input to Mapper 3: *prs*
  - Input to Mapper 4: *t*
  - Input to Reducer 2: *t+prs*
- Total Complexity: *O(r+s+t+prs)*



...3 Way Join R, S, T

- Case 1: Join *S,T* and then join *R*
  - Input to Mapper 1: *s*
  - Input to Mapper 2: *t*
  - Input to Reducer 1: *s+t*
  - Let *q* be the probability of match between *s,t*
  - Input to Mapper 3: *qst*
  - Input to Mapper 4: *r*
  - Input to Reducer 2: *r+qst*
- Total Complexity: *O(r+s+t+qst)*
  - Depends upon join order
  - If *p~q,* first join could be whichever of *rs, st, rt* is the smallest



Wall Clock Time

- Can make communication cost very low by executing all tasks on single CPU
- However, program may run slowly
- Need to consider *wall clock time*
  - Time taken for entire job to finish
- Dividing jobs could increase communication but reduce wall clock tme
  - Need to trade off communication time, wall clock time

Parameters: Wall Clock Time vs Communication cost

- Reducer size *q*
  - Max # of values that can have the same key
    - Not the number of reducers
  - If *q* is small, there can be more reducers
    - Suppose the number of Map outputs is *T*
    - Max number of reducers = *T/q*
    - This will reduce the wall clock time
    - But increase the communication cost
- Replication rate *r*
  - *r = (#key value pairs output by Mapper) / (# input records to Mapper)*
  - Average communication cost from Map tasks to Reduce tasks

Wall Clock Example: Similarity Join Between Images

- Assume we have a database of 1 million images
- Each image is 1 MB
- Total DB size = 1 TB
- Assume there is a function $s(x,y)$ which determines how similar two images x,y are
  - $s(x,y) = s(y,x)$
- Problem: output all pairs x,y such that $s(x,y) > t$

Naïve Algorithm for example

- Assume each image P_i has an index i
- Mapper
  - Reads in (i,P_i)
  - Generates all pairs $(\{i,j\},\{P_i,P_j\})$
- Reducer
  - Reads $(\{i,j\},\{P_i,P_j\})$
  - Computes $s(P_i,Pj)$

Exercise

- What is the
    - Communication cost of the naïve algorithm?
    - Parallelism of the naïve algorithm?

Exercise - Solution

- What is the
  - Communication cost of the naïve algorithm?
  - Parallelism of the naïve algorithm?
- Algorithm doesn't work
  - Data to be transmitted = 1,000,000 x 999,999 x 1,000,000 bytes = 10^{18}
  - Communication cost is ~n^2 where n is the number of images (extremely high)
  - However, potential parallelism is very high

The other extreme

- Do everything on one node
- Mapper
  - Reads in (i,P_i)
  - Generates all pairs $(\{i,j\},\{P_i,P_j\})$
- Reducer (runs on same node as mapper)
  - Reads $(\{i,j\},\{P_i,P_j\})$
  - Computes $s(P_i,Pj)$
- No communication cost
- Very low parallelism (wall clock time high)

Summary

Send one pair to each reducer

- High communication cost (bad)
- High parallelism (good)

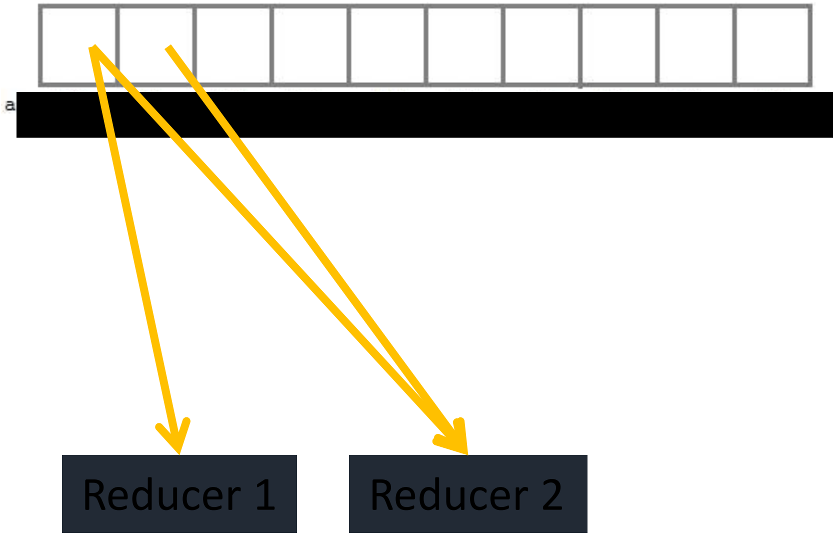
Do everything on one node

- Low parallelism (bad)
- Low communication cost (good)

Can we get something in between?

Group-based algorithm

- Overview
  - Group images
  - Read in two groups of images in a single reducer
  - Store them in memory
  - Compare all pairs from those groups
  - Output results



Example: 100 Groups

- Suppose the groups are G0, … G99
- Group G0 is sent to nodes 0, 1, ..., 98
  - Why?
- Group G0 has to be compared with 99 other groups
- Group G1 is sent to?
  - ~~0, 1, …, 98~~
- Group G1 is sent to 0, 99, 100, ...196 (0+98 other nodes)
- Group G2 is sent to?
- Group G2 is sent to 1, 99, 197, 198, ... 293 (1, 99 +97 other nodes)
- Group G3 is sent to 2, 100, 197, 294, ...389 (2, 100, 197, +96 other nodes)

In General

- If there are *g* groups
- The number of images per group $m=n/g$
- Each group is sent to $g-1$ servers
- Total number of messages = $g(g-1)$
- **Total data = $mg(g-1) = n(g-1) \sim ng$**
- **Parallelism = no of nodes = $(g-1) + (g-2) + (g-3) + \ldots = g(g-1)/2 = O(g^2)$**
- Another way = no of nodes = ${}^{n}C_{2} = g(g-1)/2$

Exercise

- Suppose we have groups of 100
  - How many groups are there?
  - How many nodes is each group sent to?
- What is the
  - Communication cost of the algorithm?
  - Parallelism of the algorithm?

...Group Based Algorithm...

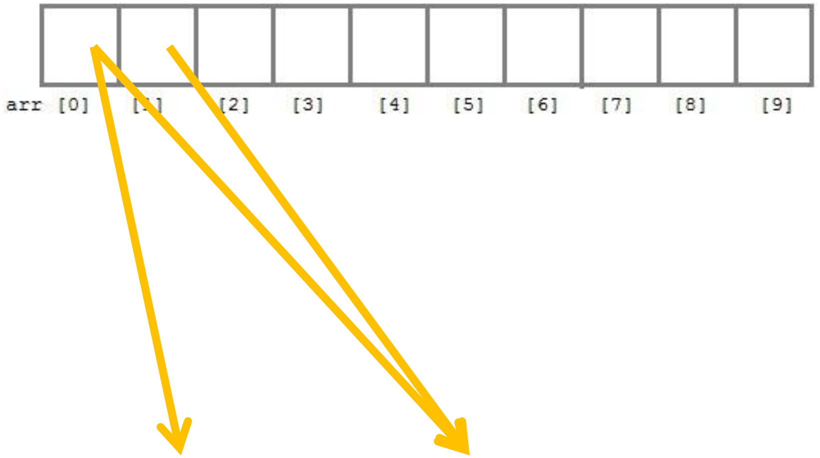
- Naïve: approximately 1,000,000x1,000,000 images = 10^{12} images, parallelism = 10^{12}
- Example: group images into groups of100
  - Communication cost: 1,000,000 images x 100 groups = 10^{8} images
  - Parallelism = 10^{4}
- Example: group images into groups of1000
  - Communication cost = 1,000,000 x 1000 = 10^{9} images
  - Parallelism = 10^{6}
- Trade-off: increasing group size
  - Increases communication complexity (more reads)
  - Reduces wall clock time (increases parallelism)

…Group Based Algorithm

- On next slide
- Note
  - Group-based algorithm is discussed in book as if mapper sends pictures to reducer
  - In previous slides, we have discussed as if reducer reads in pictures from disk
  - From communication cost complexity both are the same
  - Performance: probably better to read from disk

Group Based algorithm

- Overview
  - Group images
  - Read in two groups of images in a single reducer
  - Store them in memory
  - Compare all pairs from those groups
  - Output results
- Complexity
  - Communication cost ~*ng* where *g* is the number of groups
  - Can still get good parallelism



Grouping images..

- Group images into *g* groups
- Mapper
  - Input: (i, P_i)
  - Find u = group to which image i belongs
  - Output $g-1$ key-value pairs $(\{u,v\}, i, P_i)$ for all $v \mathrel{!}= u$
- Reducer:
  - There is one reducer for each unique key $\{u,v\}$
  - Use the input to store images for groups u,v
  - Compare all pairs of images in groups u,v
  - If $v=u+1$, then compare all pairs of images in group u

Actual Values

- If group size is 1000
- Need ~2GB to store 2000 images in memory
- Need ~500,000 reducers
- If we have a 10,000 node cluster
  - Can do computation in 50 passes
  - Speedup of 10,000

Summary

- There could be many MapReduce algorithms to implement a particular functionality
  - Matrix multiplication
  - Multi-way joins
- There are two factors to consider when selecting an algorithm
  - Communication complexity: volume of data that is input to the different phases of the algorithm
  - Wall clock time: Total elapsed time for algorithm
    - Depends upon parallelism
  - Frequently there is a trade-off between these factors
    - Group size

References

- Mining of Massive Datasets
  - Rajaraman et. Al.
  - Chapter 2.4-2.5

THANK YOU

K V Subramaniam, Usha Devi
Dept. of Computer Science and Engineering

subramaniamkv@pes.edu
ushadevibg@pes.edu