THE UNIVERSITY OF TEXAS AT AUSTIN
Department of Statistics and Data Sciences
College of Natural Sciences

SDS 385: Stat Models for Big Data

Lecture 11: Map reduce

Purnamrita Sarkar
Department of Statistics and Data Science
The University of Texas at Austin

https://psarkar.github.io/teaching

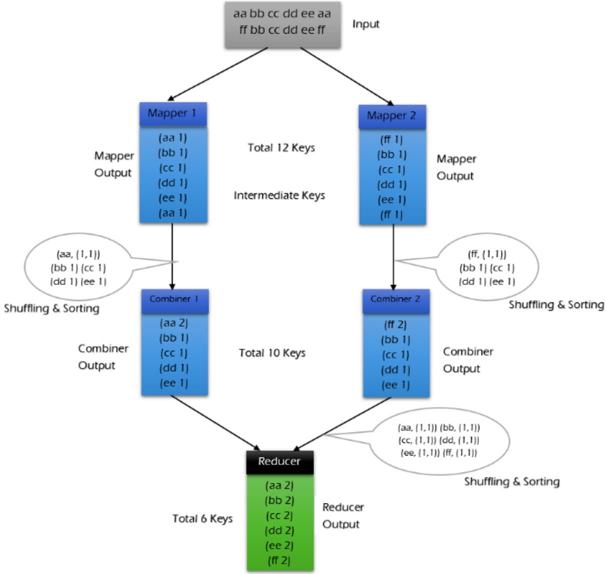
Map reduce framework

- Provides an easy framework for parallel computation
    - distributes data and takes care of synchronization
- But there are restrictions: only certain kinds of parallelisms are allowed

Map reduce basics

- Represent data as $\langle key, value \rangle$ pairs
  - Typically "key" is simple, e.g. the url of an webpage/ ID of a document
  - "value" is complex, can be contents of a webpage/ a document
- distributes data and takes care of synchronization
  - Data from the mapper is fed to the combiner, which is like a local reducer and reduces the amount of data transfer
- But there are restrictions: only certain kinds of parallelisms are allowed

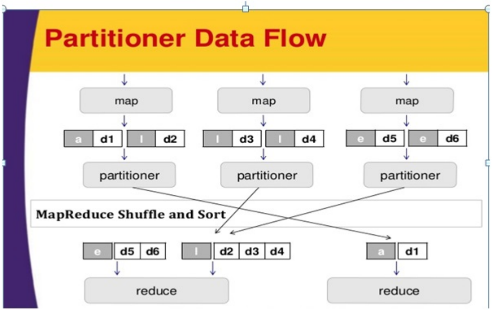
Map reduce basics



Map reduce basics - partitioner

- Runs on the same machine where the mapper had completed its execution by consuming the mapper output.
- Distributes how outputs from the map stage are send to the reducers.
- Controls the keys partition of the intermediate map-outputs.
- The key is used to derive the partition by a hash function.

Map reduce basics



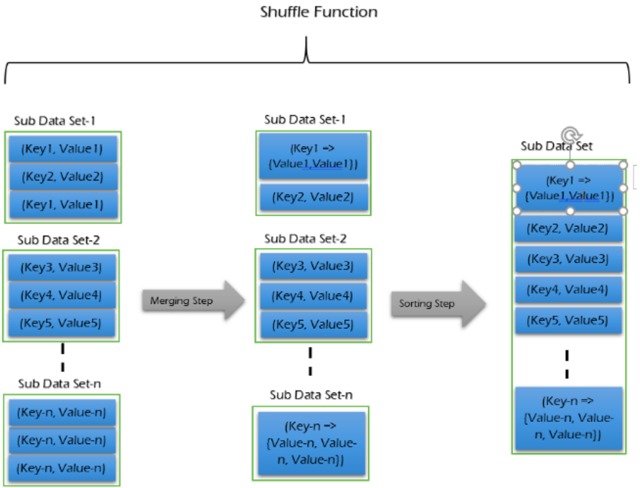
Shuffle and sort

- The shuffling process starts right away as the first mapper has completed its task.
- Once the data is shuffled to the reducer node the intermediate output is sorted based on key before sending it to reduce task.
- The algorithm used for sorting at reducer node is Merge sort.

Shuffle and sort

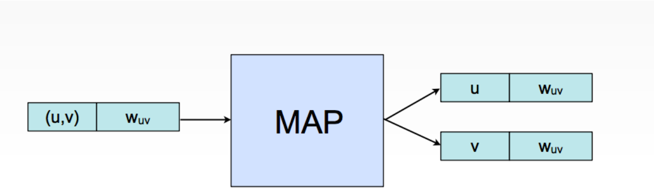
- The shuffling process starts right away as the first mapper has completed its task.
- Once the data is shuffled to the reducer node the intermediate output is sorted based on key before sending it to reduce task.
- The algorithm used for sorting at reducer node is Merge sort.

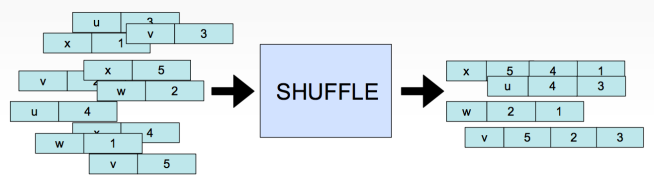
Shuffle and sort



Example - calculate degree in weighted graph

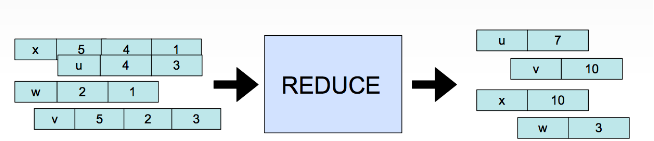
- Data: (src,dst, weight)
- Key (src,dst) value weight

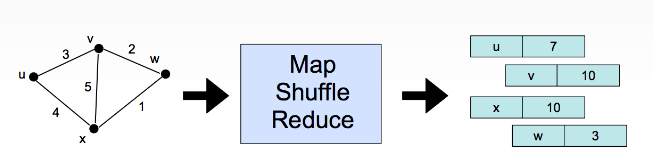




Example - calculate degree in weighted graph

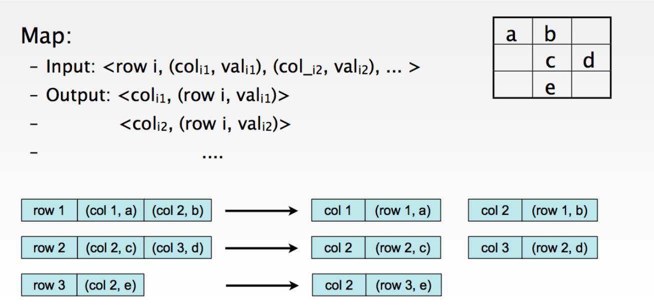
- Data: (src,dst, weight)
- Key (src,dst) value weight





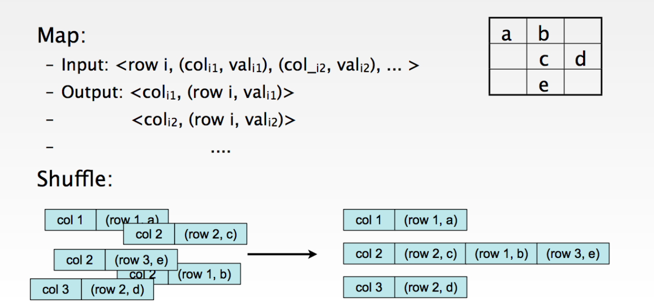
Example - transpose of a giant matrix

- Input: sparse matrix in row major order
- Output: sparse matrix in column major order



Example - transpose of a giant matrix

- Input: sparse matrix in row major order
- Output: sparse matrix in column major order



Example - transpose of a giant matrix

- Input: sparse matrix in row major order
- Output: sparse matrix in column major order

a	b	
	c	d
	e	

Map:

- Input: <row i, (col$_{i1}$, val$_{i1}$), (col_$_{i2}$, val$_{i2}$), ... >
- Output: <col$_{i1}$, (row i, val$_{i1}$)>
- <col$_{i2}$, (row i, val$_{i2}$)>
- ....

Shuffle

Reduce:

- Sort by row number

Example - linear regression

- Data is $x_i = (x_{i1}, \ldots, x_{ip})$
- response variable y_i
- Want to compute $w = (X^T X)^{-1} X^T \boldsymbol{y}$

Example - linear regression

- Data is $x_i = (x_{i1}, \ldots, x_{ip})$
- response variable y_i
- Want to compute $w = (X^T X)^{-1} X^T \boldsymbol{y}$
- $X^T X = \sum_i \boldsymbol{x}_i \boldsymbol{x}_i^T$

Example - linear regression

- Data is $x_i = (x_{i1}, \ldots, x_{ip})$
- response variable y_i
- Want to compute $w = (X^T X)^{-1} X^T \boldsymbol{y}$
- $X^T X = \sum_i \boldsymbol{x}_i \boldsymbol{x}_i^T$
- $X^T Y = \sum_i y_i \boldsymbol{x}_i$

Example - linear regression

- Input data $\langle k, \{\boldsymbol{x}_i, y_i\}_{i \in \text{subgroup} k} \rangle$

Example - linear regression

- Input data $\langle k, \{\boldsymbol{x}_i, y_i\}_{i \in \text{subgroup} k} \rangle$
- Output: two keys
  - K_1 - for $X^T X$ (calculated over subgroup), Value is a $p+1 \times p+1$ matrix
  - K_2 - for $X^Y y$ (calculated over subgroup), Value is a $p+1 \times 1$ vector

Example - linear regression

- Input data $\langle k, \{\boldsymbol{x}_i, y_i\}_{i \in \text{subgroup} k} \rangle$
- Output: two keys
    - K_1 - for $X^T X$ (calculated over subgroup), Value is a $p+1 \times p+1$ matrix
    - K_2 - for $X^Y y$ (calculated over subgroup), Value is a $p+1 \times 1$ vector
- Reducer
    - Sum the (K_1, Value) pairs to get $X^T X$
    - Sum the (K_2, Value) pairs to get $X^T \boldsymbol{y}$
    - Calculate $w = (X^T X)^{-1} X^T \boldsymbol{y}$

Example - Naive Bayes

- Data $x = (x_1, \dots, x_p)$, discrete
- Response variable y_i, in $1, \dots, K$
- Prediction for datapoint $(a_1, \dots, a_p)$:
  $y^* = \arg\max_k P(y = k) \prod_j P(x_j = a_j | y = k)$

Example - Naive Bayes

- Map:
  - Input: $\langle k, \{\boldsymbol{x}_i, y_i\}_{i \in \text{subgroup}k} \rangle$
  - Output: three types of keys
    - $K_1 = (j, a_j, k)$, and Value1 $\sum_{subgroup} 1(x_j = a_j, y = k)$
    - $K_2 = (k)$ and Value2 $\sum_{subgroup} 1(y = k)$
    - K_3 and Value3 $\sum_{subgroup} 1$

Example - Naive Bayes

- Reduce
  - Estimate $P(x_j = a_j | y = k)$ by $\frac{\sum_{K_1=(j,a_j,k)} Value1}{\sum_{K_2=k} Value2}$
  - Estimate $P(y = k)$ by $\frac{\sum_{K_2=k} Value2}{\sum_{K_3} Value3}$

Example - Expectation maximization

- Data $x = (x_1, \ldots, x_p)$
- Assumption: data is generated from a mixture of gaussians
- Want to learn parameters of Mixture model $\{\mu_k, \Sigma_k\}_{k=1}^{K}$
- E step: Calculate "expected" soft class memberships

$$\gamma_{k,i}^{(t)} = \frac{P(x_i|y = k, \Theta_t)P(y = k)}{\sum_j P(x_i|y = j, \Theta_t)P(y = j)}$$

- M step: update Θ_{t+1} using these expected weights

$$\mu_k^{(t+1)} = \frac{\sum_j x_j \gamma_{k,j}^{(t)}}{\sum_j \gamma_{k,j}^{(t)}}$$

- Derive the Σ_k update as an exercise.

Example - Expectation maximization

- Map:
  - Input: $\langle k, \{\boldsymbol{x}_i\}_{i \in \text{subgroup}k} \rangle$
  - Compute $\gamma_{k,i}^{(t)}$ for i in subset of data
  - Output: 4 types of keys
    - $K_1 = (k)$, and Value1 $\sum_{subgroup} \gamma_{k,i}^{(t)}$
    - $K_2 = (k)$ and Value2 $\sum_{subgroup} x_i \gamma_{k,i}^{(t)}$
    - $K_3 = (k)$ and Value3 $\sum_{subgroup} (x_i - \mu_k^{(t)})(x_i - \mu_k^{(t)})^T \gamma_{k,i}^{(t)}$
    - K_4 and Value4 $\sum_{subgroup} 1$

Example - EM

- Reduce
  - Estimate $\gamma_{k,i}^{(t)}$ by $\dfrac{\sum_{K_1=k} Value1}{\sum_{K4} Value4}$
  - Estimate $\mu_j^{(t)}$ by $\dfrac{\sum_{K_2=j} Value2}{\sum_{K_1=j} Value1}$
  - Estimate $\Sigma_j^{(t)}$ by $\dfrac{\sum_{K_3=j} Value3}{\sum_{K_1=j} Value1}$

Seems like we can do anything

- Any ML algorithm that has an inner step with sum over matrices or vectors concerning n datapoints can be parallelized.

Seems like we can do anything

- Any ML algorithm that has an inner step with sum over matrices or vectors concerning n datapoints can be parallelized.
- Works well when p is small, since then the communication overhead is relatively small
- Since we are invoking MR framework over each iteration of an algorithm (like GD), for large p things are pretty bad.
- So, how to reduce communication?

Acknowledgment

- https://www.tutorialscampus.com/tutorials/map-reduce/shuffle-and-sort.htm
- Sergei Vassilvitskii's notes
- Modeling with Hadoop - KDD 2011 tutorial
- Map-Reduce for Machine Learning on Multicore - NIPS 2007 paper by Chu et al.