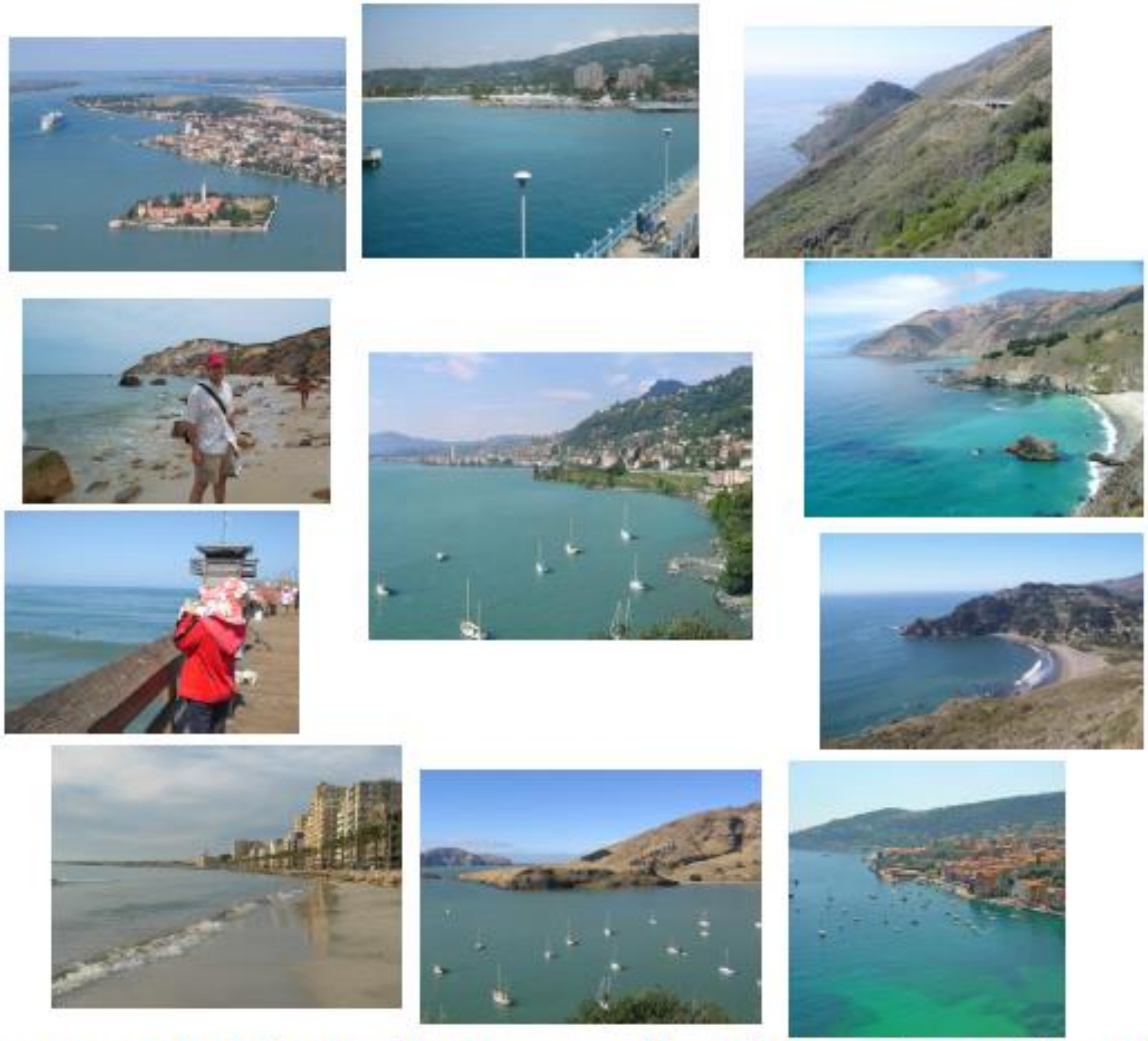
Big Data Analysis: Similarity Search

[Hays and Efros, SIGGRAPH 2007]

Scene Completion Problem



10 nearest neighbors from a collection of 2 million images

Similarity Search

- Many problems can be expressed as finding “similar” sets:
  - Find near-neighbors in high-dimensional space
- Examples:
  - Pages with similar words For duplicate detection, classification by topic
  - Customers who purchased similar products Products with similar customer sets
  - Images with similar features Users who visited similar websites

Problem Definition

- Given:
  - High dimensional data points $\boldsymbol{x_1}, \boldsymbol{x_2}, \dots$
    - For example: Image is a long vector of pixel colors
  - A distance function $\boldsymbol{d}(\boldsymbol{x_1}, \boldsymbol{x_2})$ Which quantifies the "distance" between $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$
- Goal:
  - Find all pairs of data points $(\boldsymbol{x_i}, \boldsymbol{x_j})$ that are within some distance threshold $\boldsymbol{d}(\boldsymbol{x_i}, \boldsymbol{x_j}) \leq \boldsymbol{s}$
- Note:
  - Naïve solution would take $\boldsymbol{O}(\boldsymbol{N}^2)$
  - where $\boldsymbol{N}$ is the number of data points

Theta Joins

- Use primitive comparison operators (<,>,≤,≥,≠,=) in the join-predicates

```
SELECT *
FROM R, S
WHERE R.a > S.a;
```

R

	r_{id}	a
r_1	1	1
r_2	2	1
r_3	3	2
r_4	4	3

S

	s_{id}	a
s_1	1	1
s_2	2	1
s_3	3	2
s_4	4	2
s_5	5	3
s_6	6	4

Similarity Joins

- Given
  - A distance measure d
  - A threshold σ

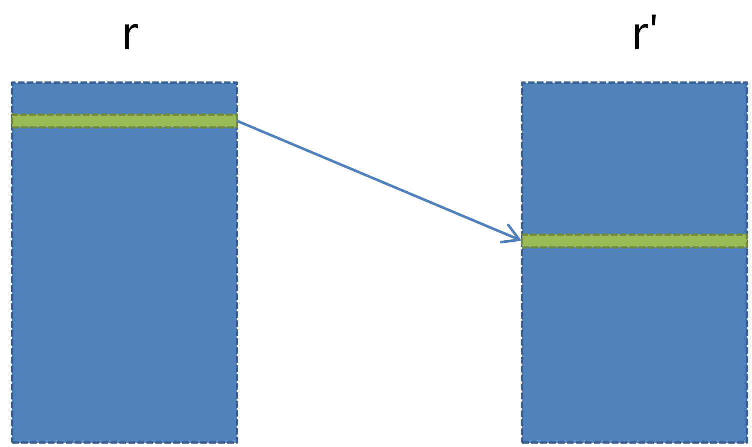
```
SELECT *
FROM R as r1, R as r2
WHERE d(r1,r2) ≤ σ
```

	$p_i(1)$	$p_i(2)$	$p_i(3)$
p_1	0.78	0.4	0.01
p_2	0.07	0.21	0.57
p_3	0.51	0.11	0.32
p_4	0.31	0.79	0.9
p_5	0.77	0.42	0.02
p_6	0.8	0.39	0.04

Nested Loop Join

- **for each** tuple t_r **in** r **do begin**
  - **for each tuple** t_s **located next to** $\underline{t_r}$ **in** r **do begin**
    - test pair (t_r, t_s) to see if they satisfy the join condition θ
    - if they do, add $t_r \bullet t_s$ to the result.
  - **end**
- **end**

n_r records
b_r blocks

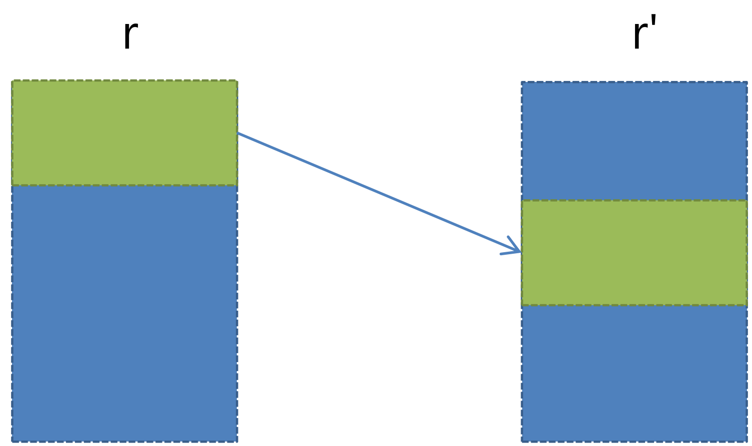


$n_r * b_r + b_r$ block transfers, plus
$n_r + b_r$ seeks

Block Nested Loop Join

- **for each** block B_r **of** r **do begin**
  - **for each next** block B_s **of** r **do begin**
    - **for each** tuple t_r **in** B_r **do begin**
      - **for each** tuple t_s **in** B_s **do begin**
        - Check if (t_r, t_s) satisfy the join condition
        - if they do, add $t_r \bullet t_s$ to the result.
      - **end**
    - **end**
  - **end**
- **end**

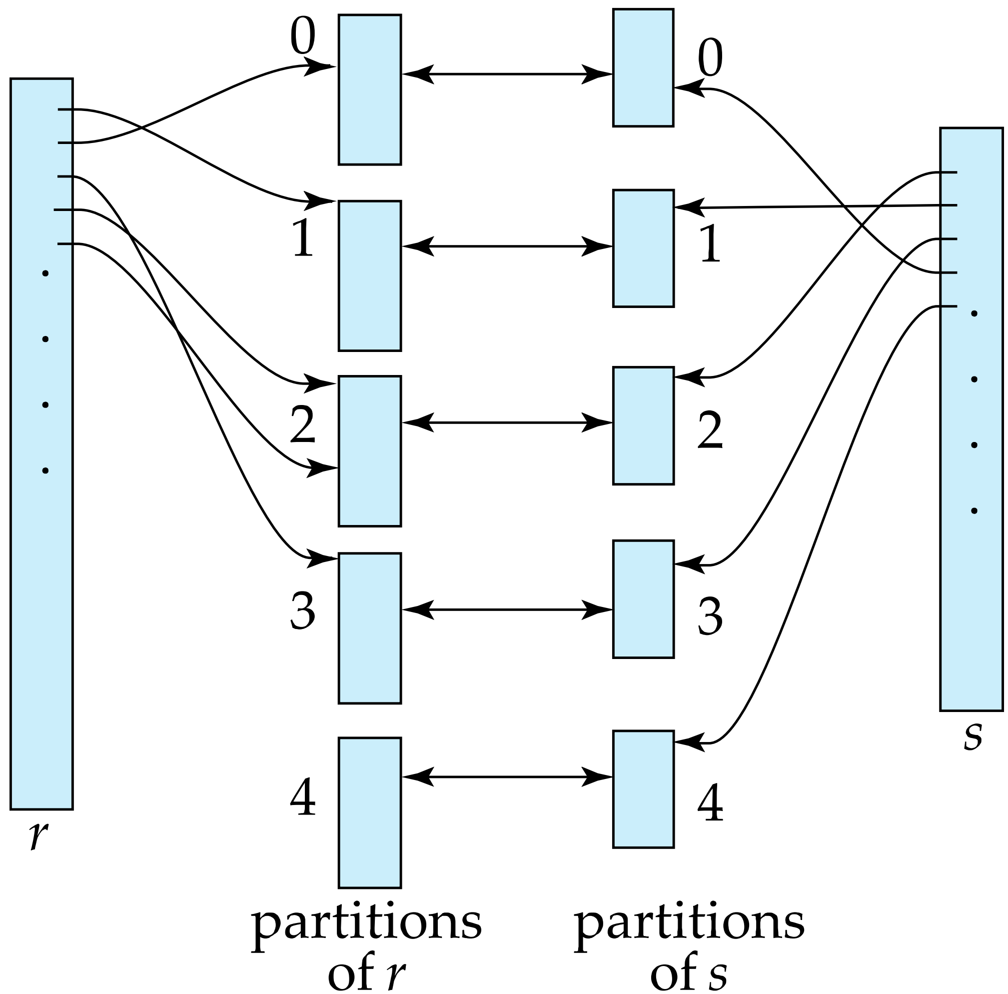
n_r records
b_r blocks



$b_r * b_r + b_r$ block transfers, plus
$n_r + b_r$ seeks

Hash Join

- Best candidate algorithm for MapReduce
- *Can we use a hash join for similarity join?*



DISTANCE MEASURES

Distance Measures

- A distance measure on this space is a function $d(x, y)$ that takes two points in the space as arguments and produces a real number, and satisfies the following axioms:
  - 1. $d(x, y) \geq 0$ (no negative distances).
  - 2. $d(x, y) = 0$ if and only if $x = y$ (distances are positive, except for the distance from a point to itself).
  - 3. $d(x, y) = d(y, x)$ (distance is symmetric).
  - 4. $d(x, y) \leq d(x, z) + d(z, y)$ (***the triangle inequality***).

Jaccard Similarity

- Jaccard coefficient/similarity
  - The Jaccard similarity of two sets is the size of their intersection divided by the size of their union:
    $$\mathbf{sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|}$$
- Jaccard distance: $d(C_1, C_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|$

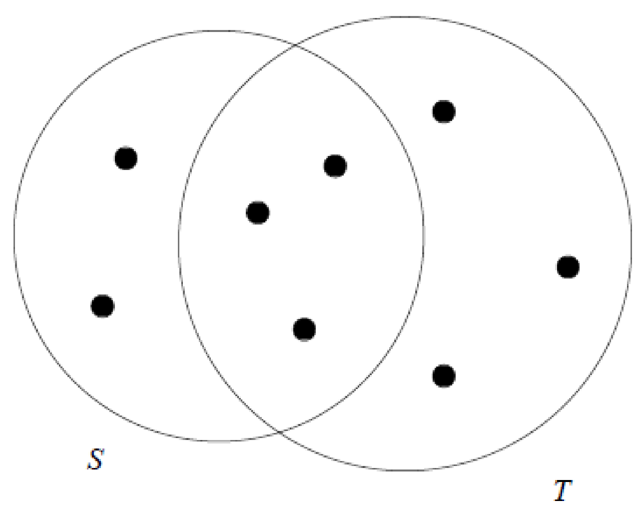


Figure 3.1: Two sets with Jaccard similarity 3/8

Exercise

- Exercise 3.1.1 :
  - Compute the Jaccard similarities of each pair of the following ***three sets***.
  - {1, 2, 3, 4}, {2, 3, 5, 7}, and {2, 4, 6}.

Euclidean Distance

- An n-dimensional Euclidean space
  - points are vectors of n real numbers
- The conventional distance measure in this space,
  - which we shall refer to as the L2-norm, is defined:

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Euclidean Distance

- L_r-norm

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r}$$

- L_1-norm
  - Manhattan distance
- L_∞-norm
  - the maximum of $|x_i - y_i|$ over all dimensions i

Exercise

- Example 3.12 :
  - Consider the two-dimensional Euclidean space (the customary plane) and the points (2, 7) and (6, 4).
  - What is Euclidean distance?
  - What is L_1-norm?
  - What is L_∞-norm?

Cosine Distance

- Given
  - two vectors x and y,
- The cosine of the angle between them is
  - the dot product x·y divided by the L_2-norms of x and y (i.e., their Euclidean distances from the origin).

$$\cos(x, y) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{i=1}^{d} y_i^2}}$$

Exercise

- Let
  - our two vectors be $x = [1, 2, -1]$ and $y = [2, 1, 1]$
- Cosine of the angle between x and y?

Hamming Distance

- Given a space of vectors,
  - we define the Hamming distance between two vectors to be the number of components in which they differ
- Example
  - The Hamming distance between the vectors 10101 and 11110 is 3.

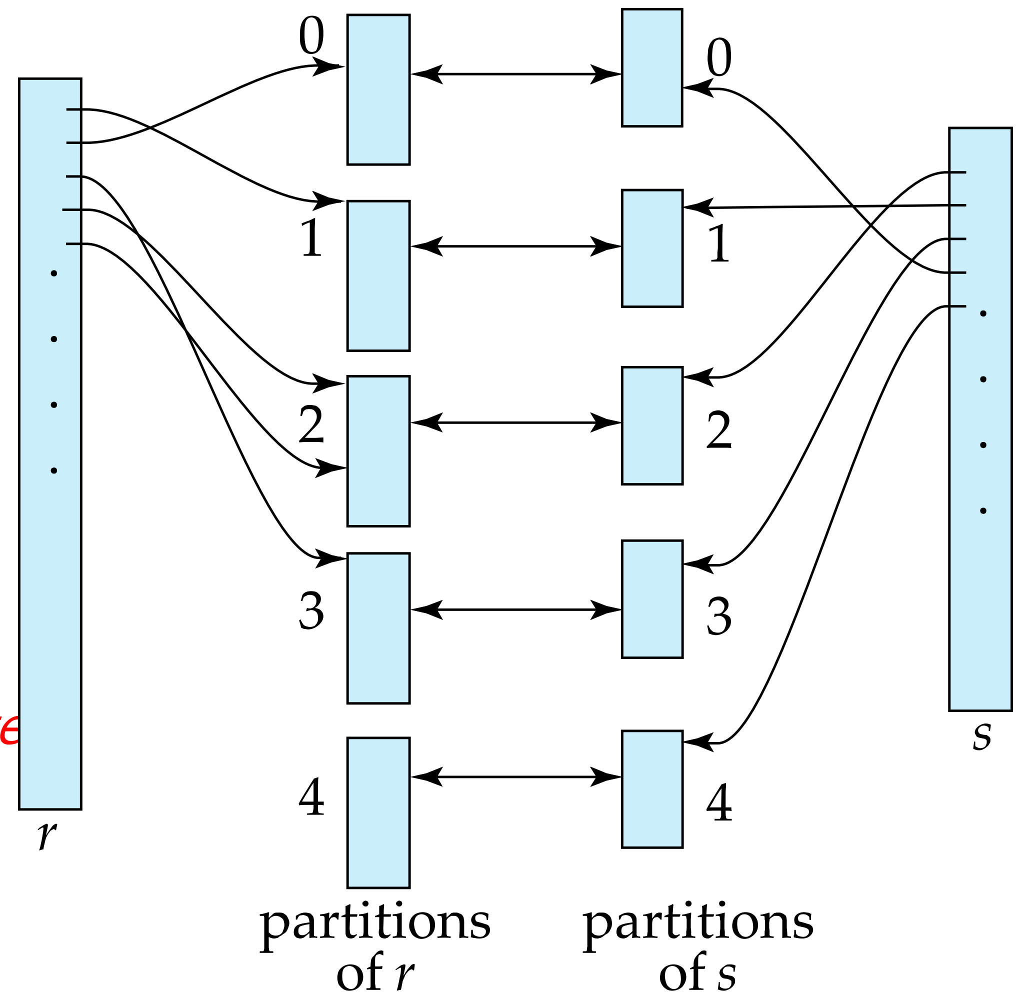
VECTOR SIMILARITY JOIN WITH MAPREDUCE

Vector Similarity Join with MapReduce

- **All pair partition algorithm**
  - **Distribute all pairs of records**
- **A naïve algorithm**
  - **[Elsayed, Lin, Oard: HLT 2008]**
  - **Build inverted lists for <u>all dimensions</u>**
- Prefix-filtering algorithms
  - [Baraglia, Morales and Lucchese, ICDM, 2010]
  - Build inverted lists of <u>a subset of dimensions</u>
- Bucket-filtering algorithm for Euclidean distance
  - [Kim, Shim, ICDE: 2012]
  - Build inverted lists with <u>a set of sub-ranges in a subset of dimensions</u>

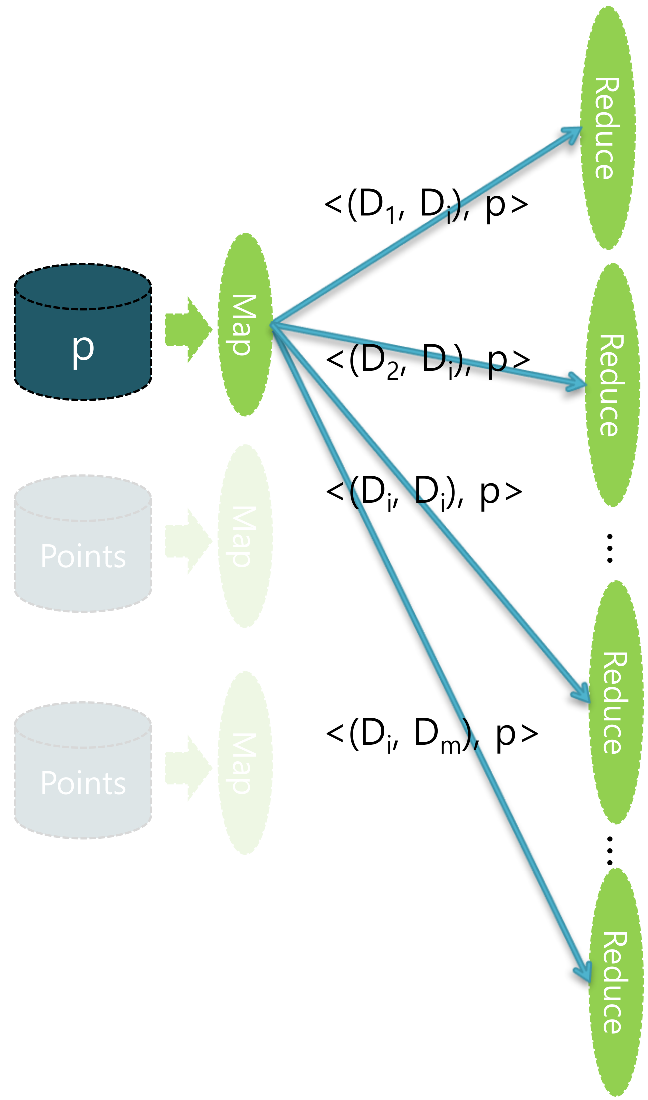
Hash Join

- Best candidate algorithm for MapReduce
- *Can we use a hash join for similarity join?*
  - *➔ Send each record to the bucket (reduce) where we need it to compute every pair of distance*



All Pair Partitioning Algorithm (Self-join)

- Simply divide and distribute the computations to find similar pairs into several reducers
- Record groups
  - $D_1, D_2, \ldots, D_m$: m distinct groups of records
- Map function
  - For each record p in the group Di, emit key-value pairs
    - $<(1,D_i), p>, \ldots, <(D_i, D_i), p>, \ldots, <(D_i, D_m), p>$
- Reduce function
  - (Dx, Dy) : a partition to compute the similarities of all pairs of records from Dx and Dy ($x \le y$)
    - Dx = Dy : self join in Dx(=Dy)
    - Dx ≠ Dy : Cartesian join between Dx and Dy



All Pair Partitioning Algorithm

$$h(p_i) = \lceil i / 2 \rceil$$

	$p_i(1)$	$p_i(2)$	$p_i(3)$
p_1	0.78	0.4	0.01
p_2	0.07	0.21	0.57
p_3	0.51	0.11	0.32
p_4	0.31	0.79	0.9
p_5	0.77	0.42	0.02
p_6	0.8	0.39	0.04

Map

Key	Value	Key	Value	Key	Value
(1,1)	p_1 = <0.78, 0.4, 0.01>	(1,2)	p_3 = <0.51, 0.11, 0.32>	(1,3)	p_5 = …
(1,2)	p_1 = <0.78, 0.4, 0.01>	(2,2)	p_3 = <0.51, 0.11, 0.32>	(2,3)	p_5 = …
(1,3)	p_1 = <0.78, 0.4, 0.01>	(2,3)	p_3 = <0.51, 0.11, 0.32>	(3,3)	p_5 = …
(1,1)	p_2 = <0.07, 0.21, 0.57>	(1,2)	p_4 = …	(1,3)	p_6 = …
(1,2)	p_2 = <0.07, 0.21, 0.57>	(2,2)	p_4 = …	(2,3)	p_6 = …
(1,3)	p_2 = <0.07, 0.21, 0.57>	(2,3)	p_4 = …	(3,3)	p_6 = …

(m=3) groups

All Pair Partitioning Algorithm

	$p_i(1)$	$p_i(2)$	$p_i(3)$
p_1	0.78	0.4	0.01
p_2	0.07	0.21	0.57
p_3	0.51	0.11	0.32
p_4	0.31	0.79	0.9
p_5	0.77	0.42	0.02
p_6	0.8	0.39	0.04

(m=3) groups

Map

Key	Value list
(1,1)	p_1, p_2
(1,2)	$p_1, p_2, \quad p_3, p_4$
(1,3)	$p_1, p_2, \quad p_5, p_6$
(2,2)	p_3, p_4
(2,3)	$p_3, p_4, \quad p_5, p_6$
(3,3)	p_5, p_6

Sort key-value pairs in the order of key & group by key

Reduce

Reduce

Key	Value
(p_1, p_2)	0.92

Key	Value
(p_1, p_3)	0.50
(p_2, p_3)	0.52
…	…

Key	Value
(p_3, p_4)	0.92

⋮

Output pairs satisfying the threshold only

<u>The number of similarity computation</u>

$= (n/m)\ (n/m-1) / 2 * m + (n/m) * (n/m) * m(m-1)/2$

$=1 * 3 + 4 * 3$

$= n(n-1) / 2$

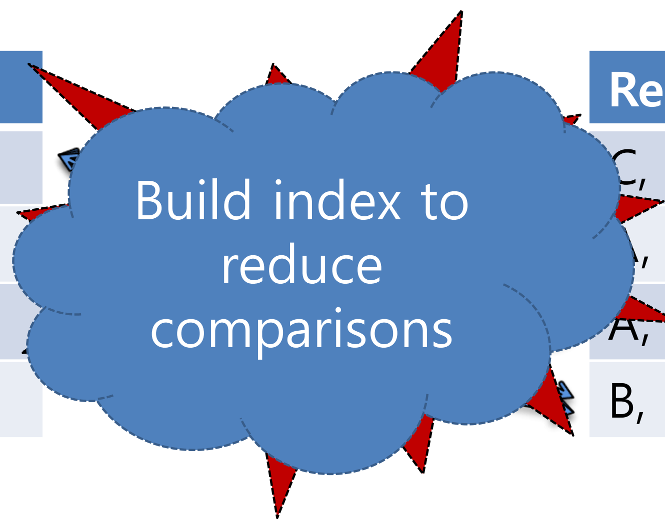
$= 15$

SET SIMILARITY JOIN USING MAPREDUCE

A Traditional Brute-force Algorithm

- Enumerate every pair of records and compute their similarities
- Expensive for large datasets
  - $O(|R|^2)$ similarity computations

Record
C, D, F
A, B, E, F, G
A, B, C, D, E
B, C, D, E, F



Record
C, D, F
A, B, E, F, G
A, B, C, D, E
B, C, D, E, F

Similarity Joins using Inverted Lists

- Make an inverted lists for all items in set data
- Generate candidates by considering every pair of record IDs in the each inverted list
- Find similar pairs by verifying each candidate
  - Relationship between Jaccard and Overlap similarity measures
    - $\text{Jaccard}(x, y) \geq \sigma \Leftrightarrow$
      $\text{Overlap}(x, y) \geq \sigma / (1+\sigma) \cdot (|x| + |y|) = \alpha$
    - We call α the overlap threshold
  - Check $\text{overlap}(x,y) \geq \alpha$ instead of $\text{Jaccard}(x,y) \geq \sigma$

Building Inverted Lists

- While scan each record in the data
  - Insert the identifier of the record (RID) into the inverted list entries of its items

RID	Items
R_1	C, D, F
R_2	A, B, E, F, G
R_3	A, B, C, D, E
R_4	B, C, D, E, F
R_5	A, E, G

Item	RIDs
A	R_2
B	R_2
C	R_1
D	R_1
E	R_2
F	R_1, R_2
G	R_2

Building Inverted Lists

- While scan each record in the data
  - Insert the identifier of the record (RID) into the inverted list entries of its items

RID	Items
R_1	C, D, F
R_2	A, B, E, F, G
R_3	A, B, C, D, E
R_4	B, C, D, E, F
R_5	A, E, G

Item	RIDs
A	R_2, R_3, R_5
B	R_2, R_3, R_4
C	R_1, R_3, R_4
D	R_1, R_3, R_4
E	R_2, R_3, R_4, R_5
F	R_1, R_2, R_4
G	R_2, R_5

Generating Candidates

- Generate candidates by making every RID pair in the each inverted list entry
  - Increase the overlap of the candidate pair

Item	RIDs
A	R_2, R_3, R_5
B	R_2, R_3, R_4
C	R_1, R_3, R_4
D	R_1, R_3, R_4
E	R_2, R_3, R_4, R_5
F	R_1, R_2, R_4
G	R_2, R_5

Candidate pair	Overlap
(R_2, R_3)	1
(R_3, R_5)	1
(R_2, R_5)	1

Generating Candidates

- Generate candidates by making every RID pair in the each inverted list entry
  - Increase the overlap of the candidate pair

Item	RIDs
A	R_2, R_3, R_5
B	R_2, R_3, R_4
C	R_1, R_3, R_4
D	R_1, R_3, R_4
E	R_2, R_3, R_4, R_5
F	R_1, R_2, R_4
G	R_2, R_5

Candidate pair	Overlap
(R_2, R_3)	2
(R_3, R_5)	1
(R_2, R_5)	1
(R_3, R_4)	1
(R_2, R_4)	1

Generating Candidates

- Generate candidates by making every RID pair in the each inverted list entry
  - Increase the overlap of the candidate pair

Item	RIDs
A	R_2, R_3, R_5
B	R_2, R_3, R_4
C	R_1, R_3, R_4
D	R_1, R_3, R_4
E	R_2, R_3, R_4, R_5
F	R_1, R_2, R_4
G	R_2, R_5

Candidate pair	Overlap
(R_2, R_3)	3
(R_3, R_5)	2
(R_2, R_5)	3
(R_3, R_4)	4
(R_2, R_4)	3
(R_1, R_3)	2
(R_1, R_4)	3

Candidate pair	Overlap
(R_4, R_5)	1
(R_1, R_2)	1

Finding Similar Pairs

Jaccard coefficient threshold $\sigma = 0.6$

Recall Jaccard(x, y) $\geq \sigma \Leftrightarrow O(x, y) \geq \alpha = \sigma/(1+\sigma)(|x| + |y|)$

Substitute σ values

We need the size of each record

Candidate pair	Overlap	Overlap threshold α
(R_2, R_3)	3	3.75
(R_3, R_5)	2	
(R_2, R_5)	3	
(R_3, R_4)	4	
(R_2, R_4)	3	
(R_1, R_3)	2	
(R_1, R_4)	3	
(R_4, R_5)	1	
(R_1, R_2)	1	

RID	Size
R_1	3
R_2	5
R_3	5
R_4	5
R_5	3

Calculate each record size

Verifying Candidates

Jaccard coefficient threshold $\sigma = 0.6$

Recall Jaccard(x, y) $\geq \sigma \Leftrightarrow O(x, y) \geq \alpha = \sigma/(1+\sigma)(|x| + |y|)$

Candidate pair	Overlap	Overlap threshold α
(R_2, R_3)	3	3.75
(R_3, R_5)	2	3
(R_2, R_5)	3	3
(R_3, R_4)	4	3.75
(R_2, R_4)	3	3.75
(R_1, R_3)	2	3
(R_1, R_4)	3	3
(R_4, R_5)	1	3
(R_1, R_2)	1	3.75

Overlap is smaller than the overlap threshold α

$\Rightarrow$ Not a similar pair

Similar pair
(R_2, R_5)
(R_3, R_4)
(R_1, R_4)