Efficient Parallel Set-Similarity Joins Using MapReduce

Rares Vernica

Department of Computer Science
University of California, Irvine

SIGMOD 2010
Joint work with Michael J. Carey and Chen Li (UC Irvine)
Outline

1 Motivation
2 Problem Statement
3 Single Machine Algorithms
   - Inverted List Index
   - Set-Similarity Filters
4 Parallel Algorithms
5 Experimental Evaluation
Example 1: Data Cleaning/Master-Data-Management

Customer data from two departments

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>S10</td>
<td>John W Smith</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R20</td>
<td>Smith John</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Master customer data across two departments

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>C30</td>
<td>John W Smith</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 1: Data Cleaning/Master-Data-Management

Customer data from two departments

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>S10</td>
<td>John W Smith</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R20</td>
<td>Smith John</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Master customer data across two departments

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>C30</td>
<td>John W Smith</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Example 1: Data Cleaning/Master-Data-Management

### Customer data from two departments

#### Sales

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>S10</td>
<td>John W Smith</td>
<td>…</td>
</tr>
<tr>
<td>:</td>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>R20</td>
<td>Smith John</td>
<td>…</td>
</tr>
<tr>
<td>:</td>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>

### Master customer data across two departments

#### Customers

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>C30</td>
<td>John W Smith</td>
<td>…</td>
</tr>
<tr>
<td>:</td>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>
Example 1: Data Cleaning/Master-Data-Management

String → Set
- S10: “John W Smith” → {John, Smith, W}
- R20: “Smith John” → {John, Smith}

Set-Similarity Metric
- Jaccard similarity/Tanimoto coefficient: \( jaccard(x, y) = \frac{|x \cap y|}{|x \cup y|} \)
- \( jaccard(S10, R20) = \frac{2}{3} \)

Set-Similarity Join
Set-Similarity Join “Sales” and “Returns” on “Name”
Example 1: Data Cleaning/Master-Data-Management

**String → Set**
- S10: “John W Smith” → {John, Smith, W}
- R20: “Smith John” → {John, Smith}

**Set-Similarity Metric**
- Jaccard similarity/Tanimoto coefficient: \( jaccard(x, y) = \frac{|x \cap y|}{|x \cup y|} \)

\[ jaccard(S10, R20) = \frac{2}{3} \]

**Set-Similarity Join**
Set-Similarity Join “Sales” and “Returns” on “Name”
Example 1: Data Cleaning/Master-Data-Management

String → Set

- S10: “John W Smith” → {John, Smith, W}
- R20: “Smith John” → {John, Smith}

Set-Similarity Metric

- Jaccard similarity/Tanimoto coefficient: $jaccard(x, y) = \frac{|x \cap y|}{|x \cup y|}$
- $jaccard(S10, R20) = \frac{2}{3}$

Set-Similarity Join

Set-Similarity Join “Sales” and “Returns” on “Name”
Problem Statement: Set-Similarity Join

**R**

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**S**

<table>
<thead>
<tr>
<th>RID</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rares Vernica (UC Irvine)  Fuzzy-Joins in MapReduce
Problem Statement: Set-Similarity Join

R.a ~ S.c
Sim = Jaccard
τ = .5

RIF

RID | a | b |
---|---|---|
1  |紫色圆形|星形|
2  |三角形|方形|紫色圆形|
3  |星形|圆形|三角形|
...|

S

RID | c | d |
---|---|---|
10 |黄色五边形|橙色矩形|
20 |红色星形|紫色梯形|蓝色方形|
30 |紫色圆形|橙色矩形|黄色五边形|
...|

RIDₚ | a | b | RIDₛ | c | d | Sim |
---|---|---|---|---|---|---|
|   |   |   |   |   |   |   |
Problem Statement: Set-Similarity Join

R.1 ∼ S.2

Sim = Jaccard
τ = .5

RID
R
a × b

RID
S
c × d

RIDR
R · a × b

RIDS
c × d

Sim
.

τ = .5
**Problem Statement: Set-Similarity Join**

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
<th></th>
<th></th>
<th></th>
<th>c</th>
<th>d</th>
<th>Sim</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Shape1" /></td>
<td><img src="image2" alt="Shape2" /></td>
<td><img src="image3" alt="Shape3" /></td>
<td><img src="image4" alt="Shape4" /></td>
<td><img src="image5" alt="Shape5" /></td>
<td><img src="image6" alt="Shape6" /></td>
<td><img src="image7" alt="Shape7" /></td>
<td><img src="image8" alt="Shape8" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image1" alt="Shape1" /></td>
<td><img src="image2" alt="Shape2" /></td>
<td><img src="image3" alt="Shape3" /></td>
<td><img src="image4" alt="Shape4" /></td>
<td><img src="image5" alt="Shape5" /></td>
<td><img src="image6" alt="Shape6" /></td>
<td><img src="image7" alt="Shape7" /></td>
<td><img src="image8" alt="Shape8" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image1" alt="Shape1" /></td>
<td><img src="image2" alt="Shape2" /></td>
<td><img src="image3" alt="Shape3" /></td>
<td><img src="image4" alt="Shape4" /></td>
<td><img src="image5" alt="Shape5" /></td>
<td><img src="image6" alt="Shape6" /></td>
<td><img src="image7" alt="Shape7" /></td>
<td><img src="image8" alt="Shape8" /></td>
</tr>
</tbody>
</table>

R.a~S.c

\[ \text{Sim} = \text{Jaccard} \]

\[ \tau = .5 \]
Problem Statement: Set-Similarity Join

Input
- Two files of records e.g., $R(RID, a, b)$ and $S(RID, c, d)$
- A join column on each file e.g., $R.a$ and $S.c$
- A similarity function, $sim$ e.g., Jaccard
- A similarity threshold, $\tau$

Output
All pairs of records from $R$ and $S$ where $sim(R.a, S.c) \geq \tau$
Outline

1 Motivation

2 Problem Statement

3 Single Machine Algorithms
   - Inverted List Index
   - Set-Similarity Filters

4 Parallel Algorithms

5 Experimental Evaluation
Single Machine Set-Similarity Join

1. Nested loops
2. Inverted list index [Sarawagi and Kirpal, 2004]
   1. Indexing phase
   2. Candidate generation phase
   3. Verification phase

10

20

30

40

50
Single Machine Set-Similarity Join

1. Nested loops
2. Inverted list index [Sarawagi and Kirpal, 2004]
   1. Indexing phase
   2. Candidate generation phase
   3. Verification phase
Single Machine Set-Similarity Join

1. Nested loops
2. Inverted list index [Sarawagi and Kirpal, 2004]
   1. Indexing phase
   2. Candidate generation phase
   3. Verification phase
Single Machine Set-Similarity Join

1. Nested loops
2. Inverted list index [Sarawagi and Kirpal, 2004]
   1. Indexing phase
   2. Candidate generation phase
   3. Verification phase

```
10  10  10
30  30  30
40  40  40
50  50  50
```
Set-Similarity Filtering

Prefix Filtering [Chaudhuri et al., 2006]

- Pigeonhole principle
- Global order for set elements:
  
  ![Diagram showing popularity of tokens]

- Sort each record’s token
Set-Similarity Filtering

Prefix Filtering [Chaudhuri et al., 2006]

- Pigeonhole principle
- Global order for set elements:

  ![Diagram](image)

  - Sort each record’s token
  - E.g., sim is overlap size, $\tau = 4$

```plaintext
10

20
```
Prefix Filtering [Chaudhuri et al., 2006]

- Pigeonhole principle
- Global order for set elements:

Sort each record’s token
E.g., sim is overlap size, \( \tau = 4 \)
Prefix length is 2

10

20
Prefix Filtering [Chaudhuri et al., 2006]

- Pigeonhole principle
- Global order for set elements:

Sort each record’s token
E.g., \( \text{sim} \) is overlap size, \( \tau = 4 \)
Prefix length is 2
Set-Similarity Filtering

Prefix Filtering [Chaudhuri et al., 2006]

- Pigeonhole principle
- Global order for set elements:

  ![Diagram](image)

  Popularity

- Sort each record’s token
- E.g., $sim$ is overlap size, $\tau = 4$
- Prefix length is 2

| 10 | △ | ▽ | □ | □ | □ |
| 20 | □ | □ | □ | □ | □ |
Set-Similarity Filtering

**Length Filtering [Arasu et al., 2006]**

- Similar records have similar lengths
- E.g.
  - \(\text{sim}\) is Jaccard
  - \(\tau = .8\)
  - Record length is 5
- Similar records have length \(\in [4, 6]\)
Set-Similarity Filtering

Length Filtering [Arasu et al., 2006]

- Similar records have similar lengths
- E.g.
  - $sim$ is Jaccard
  - $\tau = .8$
  - Record length is 5
- Similar records have length $\in [4, 6]$

<table>
<thead>
<tr>
<th>Length</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>60</td>
<td>5</td>
</tr>
<tr>
<td>70</td>
<td>6</td>
</tr>
<tr>
<td>80</td>
<td>6</td>
</tr>
<tr>
<td>90</td>
<td>7</td>
</tr>
</tbody>
</table>

Rares Vernica (UC Irvine)  Fuzzy-Joins in MapReduce
Set-Similarity Filtering

Length Filtering [Arasu et al., 2006]

- Similar records have similar lengths
- E.g.
  - $sim$ is Jaccard
  - $\tau = .8$
  - Record length is 5
- Similar records have length $\in [4, 6]$
Set-Similarity Filtering

**Length Filtering [Arasu et al., 2006]**

- Similar records have similar lengths
- E.g.
  - $\text{sim}$ is Jaccard
  - $\tau = .8$
  - Record length is 5
- Similar records have length $\in [4, 6]$

<table>
<thead>
<tr>
<th>Length</th>
<th>Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>
Outline

1. Motivation
2. Problem Statement
   - Inverted List Index
   - Set-Similarity Filters
4. Parallel Algorithms
5. Experimental Evaluation
Parallelizing Set-Similarity Joins

Large amounts of data
- E.g., GeneBank: 100M, Google N-gram: 1T
- Data or processing does not fit in one machine
- Use a cluster of machines and a parallel algorithm
- **MapReduce**: shared-nothing data-processing platform

Challenges
- Partition problem for parallelism
- Solve using **Map, Sort, and Reduce**
- Compute *end-to-end* set-similarity joins
- Deal with out-of-memory situations
Parallelizing Set-Similarity Joins

Large amounts of data
- E.g., GeneBank: 100M, Google N-gram: 1T
- Data or processing does not fit in one machine
- Use a cluster of machines and a parallel algorithm
- **MapReduce**: shared-nothing data-processing platform

Challenges
- Partition problem for parallelism
- Solve using Map, Sort, and Reduce
- Compute *end-to-end* set-similarity joins
- Deal with out-of-memory situations
Parallel Set-Similarity Joins in MapReduce

Main idea
- Hash-partition data across the network based on keys
- Join values **cannot** be directly used as keys
- Use set tokens as keys
  - e.g., “John W Smith” → \{John, Smith, W\}

Partition using prefix filter
- Use tokens in the prefix as keys
  - Minimize replication
- Global order: increasing frequency
  - Reduce skew
Main idea

- Hash-partition data across the network based on keys
- Join values **cannot** be directly used as keys
- Use set tokens as keys
  - e.g., “John W Smith” → {John, Smith, W}

Partition using prefix filter

- Use tokens in the prefix as keys
  - **Minimize replication**
- Global order: increasing frequency
  - **Reduce skew**
Processing Stages and Alternatives

Stage 1: Token Ordering
- Compute the token frequencies and sort
  \[\text{Records} \rightarrow \text{Token\_Order}\]

Stage 2: Kernel (RID-Pair Generation)
- Generate similar record-ID ("RID") pairs
  \[\{\text{Records, Token\_Order}\} \rightarrow \text{RID-pairs}\]

Stage 3: Record Join
- Generate pairs of joined records
  \[\{\text{Records, RID-pairs}\} \rightarrow \text{Record-pairs}\]
Processing Stages and Alternatives

Stage 1: Token Ordering
- Compute the token frequencies and sort
  \[\text{Records} \rightarrow \text{Token\_Order}\]

Stage 2: Kernel (RID-Pair Generation)
- Generate similar record-ID ("RID") pairs
  \[\{\text{Records, Token\_Order}\} \rightarrow \text{RID-pairs}\]

Stage 3: Record Join
- Generate pairs of joined records
  \[\{\text{Records, RID-pairs}\} \rightarrow \text{Record-pairs}\]
Processing Stages and Alternatives

Stage 1: Token Ordering
- Compute the token frequencies and sort
  \[ \text{Records} \rightarrow \text{Token\_Order} \]

Stage 2: Kernel (RID-Pair Generation)
- Generate similar record-ID ("RID") pairs
  \[ \{\text{Records, Token\_Order}\} \rightarrow \text{RID-pairs} \]

Stage 3: Record Join
- Generate pairs of joined records
  \[ \{\text{Records, RID-pairs}\} \rightarrow \text{Record-pairs} \]
Stage 1: Token Ordering

Overview

- **Input**: original records
- **Output**: token order
- Compute the token frequencies and sort

Alternatives

- Basic Token Ordering (BTO)
  - Two MapReduce phases: sort in MapReduce
- One Phase Token Ordering (OPTO)
  - One MapReduce phase: sort in memory
Stage 1: Basic Token Ordering (BTO)

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td>...</td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Stage 1: Basic Token Ordering (BTO)

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
<th>Key Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
<td>A 1</td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
<td>B 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>...</td>
<td>C 2</td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td>...</td>
<td>F 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td>...</td>
<td>F 2</td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td>...</td>
<td>G 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Stage 1: Basic Token Ordering (BTO)

```
<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td>...</td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Key Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 1</td>
</tr>
<tr>
<td>B 1</td>
</tr>
<tr>
<td>C 2</td>
</tr>
<tr>
<td>F 1</td>
</tr>
<tr>
<td>G 1</td>
</tr>
<tr>
<td>A 1</td>
</tr>
<tr>
<td>C 1</td>
</tr>
</tbody>
</table>
```

Rares Vernica (UC Irvine)
Stage 1: Basic Token Ordering (BTO)

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>E C D</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td></td>
</tr>
</tbody>
</table>

**Map**

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>E C D</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td></td>
</tr>
</tbody>
</table>

**Key Value**

<table>
<thead>
<tr>
<th>Key Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 1</td>
</tr>
<tr>
<td>B 1</td>
</tr>
<tr>
<td>C 2</td>
</tr>
<tr>
<td>D 2</td>
</tr>
<tr>
<td>E 3</td>
</tr>
</tbody>
</table>

**Reduce**

<table>
<thead>
<tr>
<th>Key Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B 2</td>
</tr>
<tr>
<td>A 2</td>
</tr>
<tr>
<td>C 2</td>
</tr>
</tbody>
</table>

**Group by key**

<table>
<thead>
<tr>
<th>Key Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 1</td>
</tr>
<tr>
<td>B 1</td>
</tr>
<tr>
<td>C 2</td>
</tr>
</tbody>
</table>

**Compute token frequencies**
Stage 1: Basic Token Ordering (BTO)

Key Value
Reduce
Reduce
Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key

Map

Reduce

Key Value
Group by key
Stage 1: Basic Token Ordering (BTO)

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>E F</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>E C D</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td>B A F</td>
</tr>
<tr>
<td>21</td>
<td>B A</td>
<td>F</td>
</tr>
</tbody>
</table>

**Phase 1: Compute token frequencies**

Map → Group by key → Reduce → Map → Group by key → Reduce → Map → Group by key → Reduce

**Key Value**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

**Group by key**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

**Reduce**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 G</td>
<td></td>
</tr>
<tr>
<td>2 A</td>
<td></td>
</tr>
<tr>
<td>2 B</td>
<td></td>
</tr>
<tr>
<td>2 D</td>
<td></td>
</tr>
<tr>
<td>3 C</td>
<td></td>
</tr>
<tr>
<td>4 F</td>
<td></td>
</tr>
</tbody>
</table>

**Group by key**
Stage 1: Basic Token Ordering (BTO)

- **Stage 1**: Basic Token Ordering (BTO)
  - **Phase 1**: Compute token frequencies
  - **Phase 2**: Sort tokens by frequency

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td>...</td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- **Map**
  - Group by key
  - Reduce

- **Reduce**
  - Group by key

- **Key Value**
  - A 1
  - B 2
  - ...
Stage 1: One Phase Token Ordering (OPTO)
Stage 2: Kernel (RID-Pair Generation)

Overview

- **Input:** original records and token order
- **Output:** list of similar-RID pairs
- Partition using prefix filter

Steps

- Load token order in memory
- Extract RIDs and join value of records
- Distribute records on prefix tokens
- Group RIDs and join values
- Cross-pair and verify candidates
### Overview

- **Input:** original records and token order
- **Output:** list of similar-RID pairs
- Partition using prefix filter

### Steps

- Load token order in memory
- Extract RIDs and join value of records
- Distribute records on prefix tokens
- Group RIDs and join values
- Cross-pair and verify candidates

```plaintext
Map
```
Stage 2: Kernel (RID-Pair Generation)

**Overview**
- **Input:** original records and token order
- **Output:** list of similar-RID pairs
- Partition using prefix filter

**Steps**
- Load token order in memory
- Extract RIDs and join value of records
- Distribute records on prefix tokens
- Group RIDs and join values
- Cross-pair and verify candidates

\[ \text{Map} ]
\[ \text{Shuffle} \]
Stage 2: Kernel (RID-Pair Generation)

### Overview
- **Input:** original records and token order
- **Output:** list of similar-RID pairs
- Partition using prefix filter

### Steps
- Load token order in memory
- Extract RIDs and join value of records
- Distribute records on prefix tokens
- Group RIDs and join values
- Cross-pair and verify candidates

Map

Shuffle

Reduce
Stage 2: Partition Using Individual Tokens

Token

Map

Key      Value

Group by key

Reduce

RID       a        b

A B C

D E F

C F

E C D

F G

B A F

...
### Stage 2: Partition Using Individual Tokens

#### Map

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td>...</td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

#### Token

- G

#### Reduce

<table>
<thead>
<tr>
<th>RID1 RID2 Sim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>21</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Stage 2: Partition Using Individual Tokens

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td>...</td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Key      Value

Group by key

Reduce

RID       a        b

A B C
D E F
...
C F
E C D
...
F G
B A F
...
...
Stage 2: Partition Using Individual Tokens

```
RID   a     b
1     A B C ...
2     D E F ...
...   ...   ...
10    C F   ...
11    E C D ...
...   ...   ...
20    F G   ...
21    B A F ...
...   ...   ...
```

```
Key | Value
--- | ---
A   | 1, A B C
B   | 1, A B C
... | ...
C   | 10, C F
D   | 11, E C D
... | ...
G   | 20, F G
A   | 21, B A F
... | ...
```
### Stage 2: Partition Using Individual Tokens

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
<td>A</td>
<td>1, A B C</td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
<td>B</td>
<td>1, A B C</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>...</td>
<td>C</td>
<td>10, C F</td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td>...</td>
<td>D</td>
<td>11, E C D</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td>...</td>
<td>G</td>
<td>20, F G</td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td>...</td>
<td>A</td>
<td>21, B A F</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Group by key**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1, A B C</td>
</tr>
<tr>
<td></td>
<td>21, B A F</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>A</td>
<td>1, A B C</td>
</tr>
<tr>
<td></td>
<td>21, B A F</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>C</td>
<td>10, C F</td>
</tr>
<tr>
<td></td>
<td>2, D E F</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Map**

- Token G
- Map
- Reduce
- Map
- Map
- Map

Fuzzy-Joins in MapReduce

Rares Vernica (UC Irvine)
Stage 2: Partition Using Individual Tokens

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
<th>Token</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
<td>G</td>
<td>A</td>
<td>1, A B C</td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
<td></td>
<td>B</td>
<td>1, A B C</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>...</td>
<td></td>
<td>C</td>
<td>10, C F</td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td>...</td>
<td></td>
<td>D</td>
<td>11, E C D</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td>...</td>
<td></td>
<td>G</td>
<td>20, F G</td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td>...</td>
<td></td>
<td>A</td>
<td>21, B A F</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Key | Value
--- | -------
B   | 1, A B C
B   | 21, B A F
A   | 1, A B C
A   | 21, B A F
C   | 10, C F
E   | 2, D E F

RID | RID2 | Sim.
--- | ---  | ----
1   | 21   | 0.5
... | ...  | ...
1   | 21   | 0.5
... | ...  | ...
2   | 11   | 0.5
... | ...  | ...
2   | 11   | 0.5
... | ...  | ...

Rares Vernica (UC Irvine)
Fuzzy-Joins in MapReduce
Stage 2: Partition Using Individual Tokens

Alternatives
- Basic Kernel (BK): nested loops
- PPJoin+ Kernel (PK): inverted list index

Rares Vernica (UC Irvine)
Stage 2: Partition Using Grouped Tokens

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Token Grouping Func.

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>B A F</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Map

Key     Value
X       1,A B C
Y       1,A B C
Z       10,C F
...     ...

Group by key

Key     Value
X       1,A B C
Y       1,A B C
Z       10,C F
X       21,B A F
Y       21,B A F
Z       2,D E F
...     ...

Reduce

Key     Value
Y       1,A B C
...     ...
...     ...
Z       10,C F
...     ...

RID1 RID2 Sim.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Rares Vernica (UC Irvine)
Stage 3: Record Join

Overview

- **Input:** original records and similar-RID pairs
- **Output:** similar-record pairs
- Generate pairs of similar records

Alternatives

- Basic Record Join (BRJ)
  - Two MapReduce phases: reduce-side join
- One Phase Record Join (OPRJ)
  - One MapReduce phase: map-side join
Stage 3: Basic Record Join (BRJ)

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>C F</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>E C D</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

RID1 | RID2 | Sim.
-----|------|-----
2     | 11   | 0.5 |
1     | 21   | 0.5 |
...   | ...  | ... |

Rares Vernica (UC Irvine)  Fuzzy-Joins in MapReduce
Stage 3: Basic Record Join (BRJ)

Map

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, A B C,...</td>
</tr>
<tr>
<td>2</td>
<td>2, D E F,...</td>
</tr>
<tr>
<td>10</td>
<td>10, C F,...</td>
</tr>
<tr>
<td>11</td>
<td>11, E C D,...</td>
</tr>
</tbody>
</table>

Reduce

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, 11</td>
<td>(2, 11), 0.5</td>
</tr>
</tbody>
</table>

Entire Record
Stage 3: Basic Record Join (BRJ)

<table>
<thead>
<tr>
<th>RID1</th>
<th>RID2</th>
<th>Sim.</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>1, A B C,...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2, D E F,...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>10, C F,...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>11, E C D,...</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>0.5</td>
<td>2</td>
<td>2, D E F,...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>(2,11), 0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>11, E C D,...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>(2,11), 0.5</td>
</tr>
</tbody>
</table>

Duplicate the RID pairs and fill half on each.
Stage 3: Basic Record Join (BRJ)

Phase 1
Duplicate the RID pairs and fill half on each

Key                  Value
Entire Record

Map

Key                  Value

Reduce

Phase 1

Map

Reduce

Group by key
Stage 3: Basic Record Join (BRJ)

Identity Map

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>2, D E F, ..., 0.5</td>
</tr>
<tr>
<td>1,21</td>
<td>21, B A F, ..., 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Map ->

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,21</td>
<td>1, A B C, ..., 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Map ->

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>11, E C D, ..., 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Map ->

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,D E F, ..., 0.5</td>
<td></td>
</tr>
<tr>
<td>21,B A F, ..., 0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Group by key

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>2, D E F, ..., 0.5</td>
</tr>
<tr>
<td>1,21</td>
<td>21, B A F, ..., 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,21</td>
<td>1, A B C, ..., 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>11, E C D, ..., 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Identity Map

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>2, D E F, ..., 0.5</td>
</tr>
<tr>
<td>1,21</td>
<td>21, B A F, ..., 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,21</td>
<td>1, A B C, ..., 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>11, E C D, ..., 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Stage 3: Basic Record Join (BRJ)

Identity Map

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>2,D E F,...,0.5</td>
</tr>
<tr>
<td>1,21</td>
<td>21,B A F,...,0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>2,D E F,...,0.5</td>
</tr>
<tr>
<td>1,21</td>
<td>11,E C D,...,0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Group by key

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>2,D E F,...,0.5</td>
</tr>
<tr>
<td>1,21</td>
<td>21,B A F,...,0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>11,E C D,...,0.5</td>
</tr>
<tr>
<td>1,21</td>
<td>1,A B C,...,0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Stage 3: Basic Record Join (BRJ)

Identity Map

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>2, D E F, ..., 0.5</td>
</tr>
<tr>
<td>1,21</td>
<td>21, B A F, ..., 0.5</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Phase 2
Bring together and fill-in the half filled pairs

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,11</td>
<td>2, D E F, ..., 0.5</td>
</tr>
<tr>
<td>1,21</td>
<td>21, B A F, ..., 0.5</td>
</tr>
<tr>
<td>1</td>
<td>1, A B C, ..., 0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID1</th>
<th>a1</th>
<th>b1</th>
<th>Sim.</th>
<th>RID2</th>
<th>a2</th>
<th>b2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
<td>0.5</td>
<td>11</td>
<td>E C D</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>A B C</td>
<td>...</td>
<td>0.5</td>
<td>21</td>
<td>B A F</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Stage 3: One-Phase Record Join (OPRJ)

Rares Vernica (UC Irvine)  Fuzzy-Joins in MapReduce
Stage 3: One-Phase Record Join (OPRJ)

RID1 RID2 Sim.

2 11 0.5

RID a b

1 A B C ...
2 D E F ...
... ...
10 C F ...
11 E C D ...
... ...
20 F G ...
21 B A F ...
... ...

Group by key

2,11 2,11 ...
1,21 ...
... ...
11,E C D,...,0.5
...
... ...
... ...
... ...

Entire Record

Key Value

Reduce

0.5 ...
0.5 ...
11 ...
21 ...
... ...
... ...
... ...
... ...
... ...
... ...
... ...
... ...

Rares Vernica (UC Irvine)
Stage 3: One-Phase Record Join (OPRJ)

```
RID   a       b
1     A B C   ... 
2     D E F   ... 
10    C F     ... 
11    E C D   ... 
20    F G     ... 
21    B A F   ... 
```

```
2 11 0.5
...
...
```

```
Entire Record
```

```
Group by key
```

```
Reduce
```

```
Key             Value
```

```
RID1     a1       b1   Sim.  RID2     a2       b2
```

```
Rares Vernica (UC Irvine)
```
Stage 3: One-Phase Record Join (OPRJ)

<table>
<thead>
<tr>
<th>RID</th>
<th>a</th>
<th>b</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
<td>1,21</td>
<td>1, A B C, ..., 0.5</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>E</td>
<td>2,11</td>
<td>2, D E F, ..., 0.5</td>
</tr>
<tr>
<td>10</td>
<td>C</td>
<td>F</td>
<td>2,11</td>
<td>2, D E F, ..., 0.5</td>
</tr>
<tr>
<td>11</td>
<td>E</td>
<td>C D</td>
<td>11, E C D</td>
<td>11, E C D, ..., 0.5</td>
</tr>
<tr>
<td>20</td>
<td>F G</td>
<td></td>
<td>1.21</td>
<td>21, B A F, ..., 0.5</td>
</tr>
<tr>
<td>21</td>
<td>B A</td>
<td>F</td>
<td>1.21</td>
<td>21, B A F, ..., 0.5</td>
</tr>
</tbody>
</table>

Map Reduce

Group by key

Entire Record
Stage 3: One-Phase Record Join (OPRJ)

```
| RID | a   | b | ...
|-----|-----|---|-----
| 1   | ABC |   |     |
| 2   | DEF |   |     |
| ... | ... |   |     |
| 10  | C F | E | D |
| 11  | E C | D |   |
| ... | ... |   | ...|
| 20  | F G | B | A F|
| 21  | B A | F |   |
| ... | ... |   | ...|
| 2   | 1,21|   |     |
| 11  | 2,11|   |     |
| ... |     |   |     |
| 2   | 2,11|   |     |
| 11  | 11,E C D,...,0.5 |
|     |     |   |     |
| 21  | 21,B A F,...,0.5 |
|     |     |   |     |
| ... |     |   |     |
```

Entire Record

```
<table>
<thead>
<tr>
<th>RID1</th>
<th>a1</th>
<th>b1</th>
<th>Sim.</th>
<th>RID2</th>
<th>a2</th>
<th>b2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>D E F</td>
<td>...</td>
<td>0.5</td>
<td>11</td>
<td>E C D</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Rares Vernica (UC Irvine)

Fuzzy-Joins in MapReduce
Additional Contributions

- Solve R-S-join case
- Optimize memory requirements for R-S join
- Handle insufficient memory
  - Introduce additional filters
  - Use external memory
1 Motivation

2 Problem Statement

3 Single Machine Algorithms
   - Inverted List Index
   - Set-Similarity Filters

4 Parallel Algorithms

5 Experimental Evaluation

Rares Vernica (UC Irvine)
### Experimental Setting

#### Hardware
- 10-node IBM x3650 cluster
  - Intel Xeon processor E5520 2.26GHz with four cores
  - Four 300GB hard disks
  - 12GB RAM

#### Software
- Ubuntu 9.06, 64-bit, server edition OS
- Java 1.6, 64-bit, server
- Hadoop 0.20.1
Experimental Setting

Datasets

- **DBLP**
  - Average length: 259 bytes
  - Number of records: 1.2M
  - Total size: 300MB

- **CITESEERX**
  - Average length: 1374 bytes
  - Number of records: 1.3M
  - Total size: 1.8GB

Increased each up to $\times 25$, preserving join properties

- DBLP: 31M records, 8.2GB
- CITESEERX: 32M records, 45GB
Running Time

- Self-join DBLP $\times n$
- $n \in [5, 25]$
- 10-node cluster
- Best time
- Bulk of the time

Dataset Size (times the original)

<table>
<thead>
<tr>
<th></th>
<th>1-BTO</th>
<th>2-BK</th>
<th>3-BRJ</th>
<th>2-PK</th>
<th>3-OPRJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Running Time

Legend
- Stage 1
  - BTO: Basic Token Ordering
  - OPTO: One Phase Token Ordering
- Stage 2
  - BK: Basic Kernel
  - PK: PPJoin+ Kernel
- Stage 3
  - BRJ: Basic Record Join
  - OPRJ: One Phase Record Join
Running Time

Self-join DBLP × n

$n \in [5, 25]$

10-node cluster

Best time

Bulk of the time

Dataset Size (times the original)
Running Time

- Self-join DBLP \( \times n \)
- \( n \in [5, 25] \)
- 10-node cluster
- Best time
- Bulk of the time

Time (seconds)

Dataset Size (times the original)
Speedup

- BTO-BK-BRJ
- BTO-PK-BRJ
- BTO-PK-OPRJ
- Ideal

Relative running time
Self-join DBLP × 10
Different cluster sizes
Speedup Breakdown

Stage 1

- BTO
- OPTO
- Ideal

Stage 2

- BK
- PK
- Ideal

Stage 3

- BRJ
- OPRJ
- Ideal

- Relative running time
- Self-join DBLP $\times 10$
- Different cluster sizes
Scaleup

Running time
Self-joining DBLP $\times n$
$n \in [5, 25]$
Proportional cluster

Rares Vernica (UC Irvine) Fuzzy-Joins in MapReduce 34 / 37
Running time

Self-joining DBLP $\times n$, $n \in [5, 25]$

Proportional cluster
Summary

- Set-similarity joins in MapReduce
  - Three-stage approach
  - Balance workload and minimize replication
- *End-to-end* algorithms
  - Self-join
  - R-S join
- Memory issues
  - Optimize memory requirements
  - Handle insufficient memory
- Experiments
  - Speedup and scaleup
  - 40 cores, 40 disks cluster