Efficient Parallel Set-Similarity Joins Using MapReduce

Rares Vernica  Michael J. Carey  Chen Li

Department of Computer Science
University of California, Irvine

Special Interest Group on Management of Data, 2010
Outline

1. Motivation
2. Problem Statement
3. Algorithms
4. Experimental Evaluation
Example: 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>

Rares Vernica (UC Irvine)  Fuzzy-Join in MapReduce  SIGMOD 2010
Example: 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><strong>John W Smith</strong></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R20</td>
<td><strong>Smith John</strong></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><strong>John W Smith</strong></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Example: Data Cleaning/Master-Data-Management

#### Customer data from two departments

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

#### Master customer data across two departments

<table>
<thead>
<tr>
<th>Customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>C30</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

- Rares Vernica (UC Irvine)
- Fuzzy-Join in MapReduce
- SIGMOD 2010
### Example: Data Cleaning/Master-Data-Management

**Customer data from two departments**

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

**Master customer data across two departments**

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>C30</td>
<td>John W Smith</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rares Vernica (UC Irvine)  Fuzzy-Join in MapReduce  SIGMOD 2010
Example: 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: $\text{jaccard}(x, y) = \frac{|x \cap y|}{|x \cup y|}$
- $\text{jaccard}(S10, R20) = \frac{2}{3}$

Set-Similarity Join
Set-Similarity Join “Sales” and “Returns” on “Name”
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: 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: 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: 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: 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: 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: \( \text{jaccard}(x, y) = \frac{|x \cap y|}{|x \cup y|} \)
- \( \text{jaccard}(S10, R20) = \frac{2}{3} \)

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

<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>

<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>
### Problem Statement: Set-Similarity Join

<table>
<thead>
<tr>
<th>RID</th>
<th>RIID</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Sim</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td><img src="image" alt="Shape" /></td>
<td><img src="image" alt="Shape" /></td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td><img src="image" alt="Shape" /></td>
<td><img src="image" alt="Shape" /></td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td><img src="image" alt="Shape" /></td>
<td><img src="image" alt="Shape" /></td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td><img src="image" alt="Shape" /></td>
<td><img src="image" alt="Shape" /></td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**R** \( a \sim S \) \( c \)

**Sim** = Jaccard

\( \tau = 0.5 \)
Problem Statement: Set-Similarity Join

R. Vernica (UC Irvine)

Fuzzy-Join in MapReduce

SIGMOD 2010

$\text{Problem Statement: Set-Similarity Join}$

$\text{R. a} \sim \text{S. c}$

$\text{Sim} = \text{Jaccard}$

$\tau = .5$

$\begin{array}{c|c|c}
\text{RID} & \text{a} & \text{b} \\
\hline
1 & \text{\includegraphics[width=0.5cm]{purple.png}} & \text{\includegraphics[width=0.5cm]{red-star.png}} & \text{\includegraphics[width=0.5cm]{gray-triangle.png}} & \ldots \\
2 & \text{\includegraphics[width=0.5cm]{green-triangle.png}} & \text{\includegraphics[width=0.5cm]{blue-box.png}} & \text{\includegraphics[width=0.5cm]{purple-circle.png}} & \ldots \\
3 & \text{\includegraphics[width=0.5cm]{red-star.png}} & \text{\includegraphics[width=0.5cm]{yellow-triangle.png}} & \text{\includegraphics[width=0.5cm]{green-triangle.png}} & \ldots \\
\ldots & & & & \\
\end{array}$

$\begin{array}{c|c|c}
\text{RID} & \text{c} & \text{d} \\
\hline
10 & \text{\includegraphics[width=0.5cm]{yellow.png}} & \text{\includegraphics[width=0.5cm]{green-triangle.png}} & \text{\includegraphics[width=0.5cm]{orange-box.png}} & \ldots \\
20 & \text{\includegraphics[width=0.5cm]{red-star.png}} & \text{\includegraphics[width=0.5cm]{gray-triangle.png}} & \text{\includegraphics[width=0.5cm]{blue-box.png}} & \ldots \\
30 & \text{\includegraphics[width=0.5cm]{purple-circle.png}} & \text{\includegraphics[width=0.5cm]{orange-box.png}} & \text{\includegraphics[width=0.5cm]{yellow-triangle.png}} & \ldots \\
\ldots & & & & \\
\end{array}$

$\begin{array}{c|c|c|c|c}
\text{RID}_R & \text{a} & \text{b} & \text{RID}_S & \text{c} & \text{d} & \text{Sim} \\
\hline
1 & \text{\includegraphics[width=0.5cm]{purple.png}} & \text{\includegraphics[width=0.5cm]{red-star.png}} & \text{\includegraphics[width=0.5cm]{gray-triangle.png}} & \ldots & 20 & \text{\includegraphics[width=0.5cm]{red-star.png}} & \text{\includegraphics[width=0.5cm]{gray-triangle.png}} & \text{\includegraphics[width=0.5cm]{blue-box.png}} & \ldots & .5 \\
\end{array}$

http://sig.cs.ucr.edu
Problem Statement: Set-Similarity Join

\[
\begin{array}{|c|c|c|}
\hline
\text{RID} & a & b \\
\hline
1 & \text{\ding{51} \star \triangle \square \circ} & \ldots \\
2 & \text{\triangle \square \circ \circ} & \ldots \\
3 & \text{\star \pentagon \triangle \triangle} & \ldots \\
\vdots & \vdots & \vdots \\
\hline
\end{array}
\quad \bowtie \quad
\begin{array}{|c|c|c|}
\hline
\text{RID} & c & d \\
\hline
10 & \text{\pentagon \triangle \square \square} & \ldots \\
20 & \text{\star \triangle \square \square} & \ldots \\
30 & \text{\circ \pentagon \square \triangle} & \ldots \\
\vdots & \vdots & \vdots \\
\hline
\end{array}
\]

Sim = Jaccard
\[\tau = 0.5\]
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 $sim(R.a, S.c) \geq \tau$
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
MapReduce Review

map \ (k1,v1) \rightarrow \text{list}(k2,v2);\ 
reduce \ (k2,\text{list}(v2)) \rightarrow \text{list}(k3,v3).\ 

combine \ (k2,\text{list}(v2)) \rightarrow \text{list}(k2,v2).
map \ (k_1, v_1) \rightarrow \text{list}(k_2, v_2);
reduce \ (k_2, \text{list}(v_2)) \rightarrow \text{list}(k_3, v_3).

combine \ (k_2, \text{list}(v_2)) \rightarrow \text{list}(k_2, v_2).
map \ (k_1, v_1) \rightarrow \text{list} (k_2, v_2);
reduce \ (k_2, \text{list} (v_2)) \rightarrow \text{list} (k_3, v_3).

combine \ (k_2, \text{list} (v_2)) \rightarrow \text{list} (k_2, v_2).
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
- Generate *signatures* from join values and use them as keys
  - e.g., “John W Smith” → {John, Smith, W}

Three Stages
1. Compute set-element statistics
   Records → Statistics
2. Generate similar record-ID (“RID”) pairs
   {Records, Statistics} → RID-pairs
3. Generate pairs of joined records
   {Records, RID-pairs} → Record-pairs
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
- Generate signatures from join values and use them as keys
  - e.g., “John W Smith” → {John, Smith, W}

Three Stages
1. Compute set-element statistics
   \(\text{Records} \rightarrow \text{Statistics}\)

2. Generate similar record-ID ("RID") pairs
   \(\{\text{Records, Statistics}\} \rightarrow \text{RID-pairs}\)

3. Generate pairs of joined records
   \(\{\text{Records, RID-pairs}\} \rightarrow \text{Record-pairs}\)
Set-Similarity Filtering

E.g., Prefix Filtering

- Pigeonhole principle
- Global order for set elements:

  ![Diagram showing shapes ordered by popularity](image)

  Popularity
Set-Similarity Filtering

E.g., Prefix Filtering

- Pigeonhole principle
- Global order for set elements:
  - E.g., $\text{sim}$ is overlap size, $\tau = 4$

Popularity

10

20
Set-Similarity Filtering

E.g., Prefix Filtering

- Pigeonhole principle
- Global order for set elements:

![Diagram](image)

- E.g., $\text{sim}$ is overlap size, $\tau = 4$
- Prefix length is 2

<table>
<thead>
<tr>
<th>10</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Set-Similarity Filtering

E.g., Prefix Filtering

- Pigeonhole principle
- Global order for set elements:

  ![Diagram of popularity](image)

  **Popularity**

- E.g., $\text{sim}$ is overlap size, $\tau = 4$
- Prefix length is 2

<table>
<thead>
<tr>
<th>10</th>
<th>▲</th>
<th>▼</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>■</td>
<td>△</td>
</tr>
</tbody>
</table>
Set-Similarity Filtering

E.g., Prefix Filtering

- Pigeonhole principle
- Global order for set elements:

  ![Diagram of shapes with arrows indicating popularity]

- E.g., \( \text{sim} \) is overlap size, \( \tau = 4 \)
- Prefix length is 2

10

![Shapes with numbers and colors]

20
Processing Stages and Alternatives

Stage 1: Token Ordering
- Compute the token frequencies and sort
  - Two MapReduce phases: sort in MapReduce (BTO)
  - One MapReduce phase: sort in memory (OPTO)

Stage 2: Kernel (RID-Pair Generation)
- Use prefix-filter to divide, conquer using:
  - Nested loops (BK)
  - Single-machine set-similarity join algorithm (PK)

Stage 3: Record Join
- Generate pairs of similar records
  - Two MapReduce phases: reduce-side join (BRJ)
  - One MapReduce phase: map-side join (OPRJ)
Processing Stages and Alternatives

Stage 1: Token Ordering
- Compute the token frequencies and sort
  - Two MapReduce phases: sort in MapReduce (BTO)
  - One MapReduce phase: sort in memory (OPTO)

Stage 2: Kernel (RID-Pair Generation)
- Use prefix-filter to divide, conquer using:
  - Nested loops (BK)
  - Single-machine set-similarity join algorithm (PK)

Stage 3: Record Join
- Generate pairs of similar records
  - Two MapReduce phases: reduce-side join (BRJ)
  - One MapReduce phase: map-side join (OPRJ)
Processing Stages and Alternatives

**Stage 1: Token Ordering**
- Compute the token frequencies and sort
  - Two MapReduce phases: sort in MapReduce (BTO)
  - One MapReduce phase: sort in memory (OPTO)

**Stage 2: Kernel (RID-Pair Generation)**
- Use prefix-filter to *divide, conquer* using:
  - Nested loops (BK)
  - Single-machine set-similarity join algorithm (PK)

**Stage 3: Record Join**
- Generate pairs of similar records
  - Two MapReduce phases: reduce-side join (BRJ)
  - One MapReduce phase: map-side join (OPRJ)
Stage 2: RID-Pair Generation

Token
G
...

Map

Map

Map
Stage 2: RID-Pair Generation

<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

Map

G

...
Stage 2: RID-Pair Generation

<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

Map

Map

Map
## Stage 2: RID-Pair Generation

### Table

<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>...</td>
</tr>
<tr>
<td>11</td>
<td>E C D</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>

### Diagram

```
Token
  G

RID    | a    | b    | Key   | Value        |
-------|------|------|-------|--------------|
 1      | A B C| ...  | A     | 1, A B C     |
 2      | D E F| ...  | B     | 1, A B C     |
 10     | C F  | ...  | C     | 10, C F      |
 11     | E C D| ...  | D     | 11, E C D    |
 20     | F G  | ...  | G     | 20, F G      |
 21     | B A F| ...  | A     | 21, B A F    |
```
Stage 2: RID-Pair Generation

Rares Vernica (UC Irvine)

Fuzzy-Join in MapReduce

SIGMOD 2010

12 / 22
Stage 2: RID-Pair Generation

Rares Vernica (UC Irvine)

Fuzzy-Join in MapReduce

SIGMOD 2010
Additional Contributions

- Solve R-S-join case
- Optimize memory requirements for R-S join
- Handle insufficient memory
  - Introduce additional filters
  - Use external memory
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 $\times25$, 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)
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)

- 1-BTO
- 2-BK
- 3-BPJ
- 2-PK
- 3-OPRJ

Time (seconds)
Running Time

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

<table>
<thead>
<tr>
<th>Dataset Size (times the original)</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1-BTO</td>
</tr>
<tr>
<td>10</td>
<td>2-BK</td>
</tr>
<tr>
<td>25</td>
<td>3-BRJ</td>
</tr>
<tr>
<td>2-PK</td>
<td></td>
</tr>
<tr>
<td>3-OPRJ</td>
<td></td>
</tr>
</tbody>
</table>

Rares Vernica (UC Irvine) | Fuzzy-Join in MapReduce | SIGMOD 2010
Speedup

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

Relative running time

Self-join DBLP $\times 10$

Different cluster sizes
Speedup Breakdown

Stage 1

Stage 2

Stage 3

- Relative running time
- Self-join DBLP × 10
- Different cluster sizes
Running time
Self-joining DBLP $\times n$
$n \in [5, 25]$
Proportional cluster

Scaleup

# Nodes and Dataset Size

Time (seconds)

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

Stage 1

- Running time
- Self-joining DBLP × n, n ∈ [5, 25]
- Proportional cluster

Stage 2

Stage 3
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
  - 40 cores, 40 disks cluster
  - Speedup and scaleup
A longer version of the paper, the source-code and the datasets:
http://asterix.ics.uci.edu/fuzzyjoin-mapreduce/

This work is part of the
ASTERIX
http://asterix.ics.uci.edu/
and
Flamingo
http://flamingo.ics.uci.edu/
projects at UC Irvine.
Stage 1: Basic Token Ordering (BTO)

**Phase 1:** Compute token frequencies

- **Map:** Input data is read and split into key-value pairs (A 1, B 1, C 2, D 1, E 1, F 2, G 1).
- **Reduce:** Groups key-value pairs by key (A, B, C, D, E, F, G).
- **Group by key:** Groups keys further by their frequency (A 1, B 1, C 1, D 2, E 1, F 2, G 1).

**Phase 2:** Sort tokens by frequency

- **Map:** Input data is read and split into key-value pairs (A 1, B 2, C 3, D 2, E 2, F 4, G 1).
- **Reduce:** Groups key-value pairs by key (A, B, C, D, E, F, G).
- **Group by key:** Groups keys further by their frequency (A 1, B 2, C 3, D 2, E 2, F 4, G 1).

**Token Output:**

- 1: G
- 2: A, B, D, E
- 3: C
- 4: F
## Stage 1: One Phase Token Ordering (OPTO)

<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**
  - A 1
  - B 1

### Group by key
- **Key Value**
  - B 1
  - A 1

### Reduce
- **Token**
  - G
  - A
  - B
  - D
  - E
  - C
  - F

---

One Phase

Compute token ordering
### Stage 2: Basic Kernel (BK)

#### Rares Vernica (UC Irvine)

Fuzzy-Join in MapReduce

SIGMOD 2010

---

<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>A</td>
<td>1, A</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>E</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>C</td>
<td>F</td>
<td>C</td>
<td>10, C</td>
</tr>
<tr>
<td>11</td>
<td>E</td>
<td>C</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>F</td>
<td>G</td>
<td>G</td>
<td>20, F</td>
</tr>
<tr>
<td>21</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>21, B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1, A</td>
</tr>
<tr>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>10, C</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>21, B</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>20, F</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>21, B</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>10, C</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2, D</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>RID1</th>
<th>RID2</th>
<th>Sim.</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>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>
Stage 2: Indexed Kernel (PK)

- **Token G**
- **Token Grouping Func.**

<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>X</td>
<td>1, A, B, C</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>E</td>
<td>Y</td>
<td>1, A, B, C</td>
</tr>
<tr>
<td>10</td>
<td>C</td>
<td>F</td>
<td>Z</td>
<td>10, C, F</td>
</tr>
<tr>
<td>11</td>
<td>E</td>
<td>C</td>
<td>Z</td>
<td>11, E, C, D</td>
</tr>
<tr>
<td>20</td>
<td>F</td>
<td>G</td>
<td>Y</td>
<td>20, F, G</td>
</tr>
<tr>
<td>21</td>
<td>B</td>
<td>A</td>
<td>X</td>
<td>21, B, A, F</td>
</tr>
</tbody>
</table>

**Map**

**Group by key**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Reduce</th>
<th>RID1</th>
<th>RID2</th>
<th>Sim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>1, A, B, C</td>
<td>Reduce</td>
<td>1</td>
<td>21</td>
<td>0.5</td>
</tr>
<tr>
<td>Y</td>
<td>21, B, A, F</td>
<td>Reduce</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Z</td>
<td>10, C, F</td>
<td>Reduce</td>
<td>1</td>
<td>21</td>
<td>0.5</td>
</tr>
<tr>
<td>Z</td>
<td>11, E, C, D</td>
<td>Reduce</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Y</td>
<td>20, F, G</td>
<td>Reduce</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>X</td>
<td>21, B, A, F</td>
<td>Reduce</td>
<td>1</td>
<td>21</td>
<td>0.5</td>
</tr>
<tr>
<td>Z</td>
<td>2, D, E, F</td>
<td>Reduce</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Rares Vernica (UC Irvine)  Fuzzy-Join in MapReduce  SIGMOD 2010
Stage 3: Basic Record Join (BRJ)

**Phase 1**
Duplicate the RID pairs and fill half on each

**Phase 2**
Bring together and join the half filled pairs

Rares Vernica (UC Irvine)
Fuzzy-Join in MapReduce
SIGMOD 2010
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>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a1</td>
<td>b1</td>
<td>1,21</td>
<td>1, A B C,...,0.5</td>
<td>2,11</td>
<td>2, D E F,...,0.5</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>2,11</td>
<td></td>
<td>11, E C D,...,0.5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>2,11</td>
<td></td>
<td>1,21</td>
<td>21, B A F,...,0.5</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1,21</td>
<td>1, A B C,...,0.5</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>1,21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

RID1 RID2 Sim.
2 11 0.5

Entire Record

Group by key

Reduce

```
<table>
<thead>
<tr>
<th>RID1</th>
<th>a2</th>
<th>b2</th>
<th>Sim.</th>
<th>RID2</th>
<th>a1</th>
<th>b1</th>
<th>Sim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>D</td>
<td>E</td>
<td>0.5</td>
<td>11</td>
<td>E</td>
<td>C</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
<td>0.5</td>
<td>21</td>
<td>B</td>
<td>A</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Rares Vernica (UC Irvine)
Fuzzy-Join in MapReduce
SIGMOD 2010