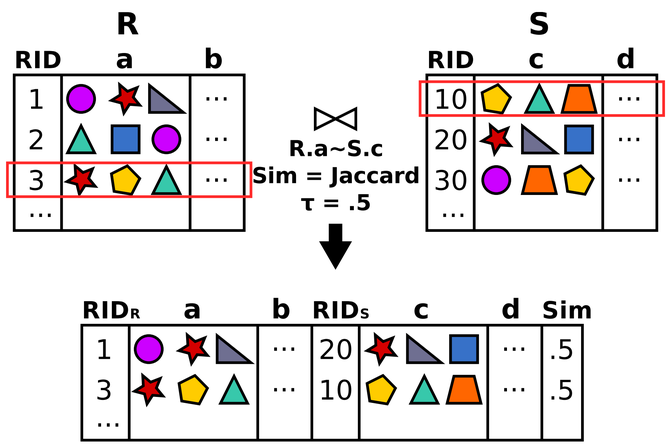
Efficient Parallel Set-Similarity Joins Using MapReduce

Rares Vernica Michael J. Carey Chen Li

Department of Computer Science, University of California, Irvine

http://asterix.ics.uci.edu/fuzzyjoin-mapreduce/

Problem Statement



Example: Data Cleaning/Master-Data-Management

Customer data from two departments

Sales

ID	Name	...
S10	John W Smith	...
⋮		

Returns

ID	Name	...
R20	~~Smith John~~ John W Smith	...
⋮		

Master customer data across two departments

Customers

ID	Name	...
C30	John W Smith	...
⋮		

Parallelizing Set-Similarity Joins

Large amounts of data

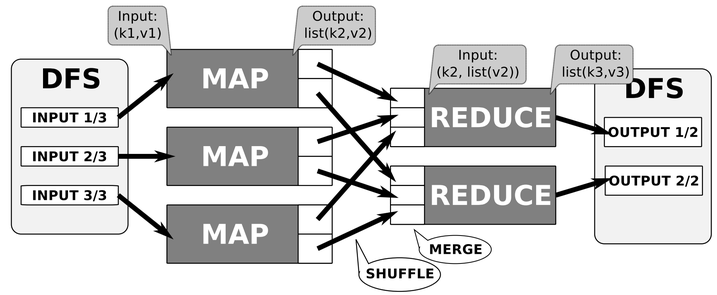
- E.g., GeneBank: 100M, Google N-gram: 1T
- Data or processing does not fit in one machine
- Use a cluster of machines and a parallel algorithm
- MapReduce: shared-nothing data-processing platform

Challenges

- Partition problem for parallelism
- Solve the problem using `Map`, `Sort`, and `Reduce`
- Compute *end-to-end* set-similarity joins
- Deal with out-of-memory situations

MapReduce Review

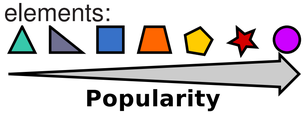
```
map     (k1,v1)        → list(k2,v2);
reduce  (k2,list(v2))  → list(k3,v3).
```



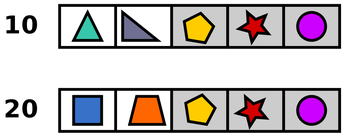

```
combine (k2,list(v2)) → list(k2,v2).
```

Prefix Filtering for Data Partitioning

- Pigeonhole principle
- Global order for set elements:



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



Processing Stages and Alternatives

Stage 1: Token Ordering

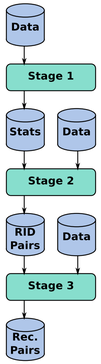
- Compute the token frequencies and sort
  - Two MapReduce phases: sort in MapReduce (BTO)
  - One MapReduce phase: sort in memory (OPTO)

Stage 2: Kernel (RID-Pair Generation)

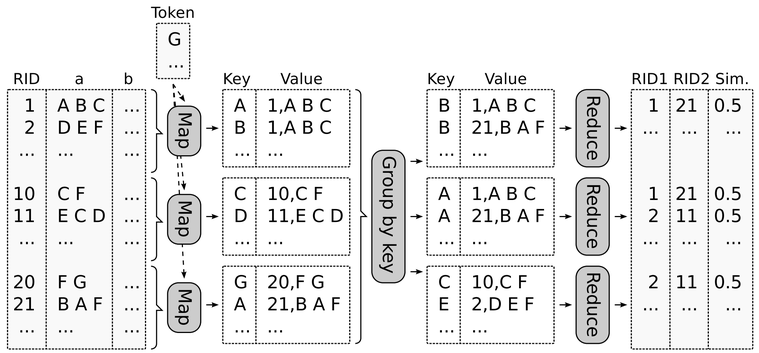
- Use prefix-filter to *divide*, *conquer* using:
  - Nested loops (BK)
  - Single-machine set-similarity join algorithm (PK)

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

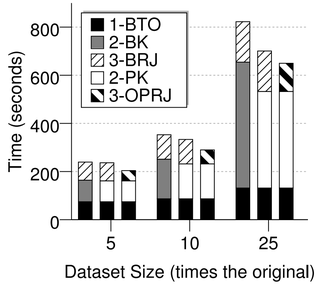
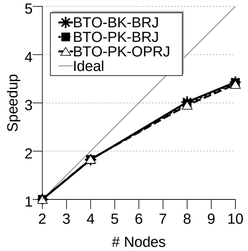
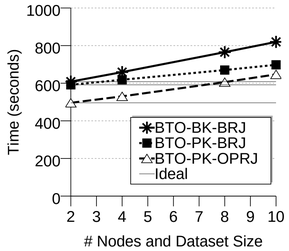


Experimental Setting

Hardware

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

Datasets

- DBLP: average length: 259 bytes; 1.2M records; 300MB
- CITESEERX: average length: 1374 bytes; 1.3M records; 1.8GB
- Increased each up to ×25, preserving join properties

Experimental Results