ARIZONA STATE UNIVERSITY

An Experimental Survey of MapReduce-based Similarity Joins

Yasin Silva, Jason Reed, Kyle Brown, Adelbert Wadsworth, Chuitian Rong

Arizona State University

sisap

Motivation

The Problem

- Big-Data systems have been introduced to efficiently process and analyze massive amounts of data.
- One of the key data processing and analysis operations is the Similarity Join (SJ), which finds similar pairs of objects of two datasets.
- Several SJ techniques for Big-Data (MapReduce) have been proposed [1-7] but many of these techniques were not compared against alternative approaches.
  - Some techniques were developed in parallel.
  - Others were not implemented as part of their original publications.
- Consequently, there is not a clear understanding of how these techniques compare to each other and which technique to use in specific scenarios.

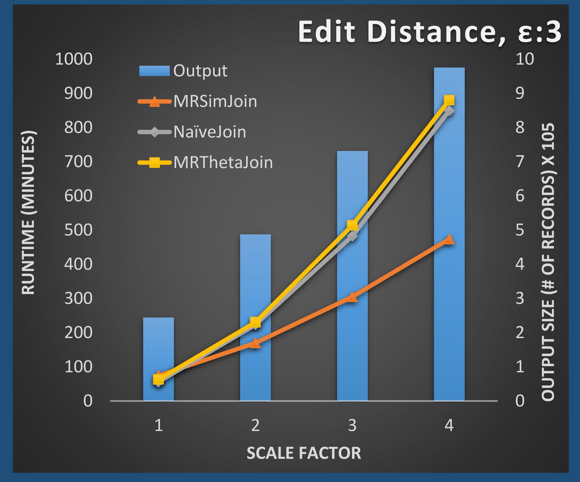
Our Contribution

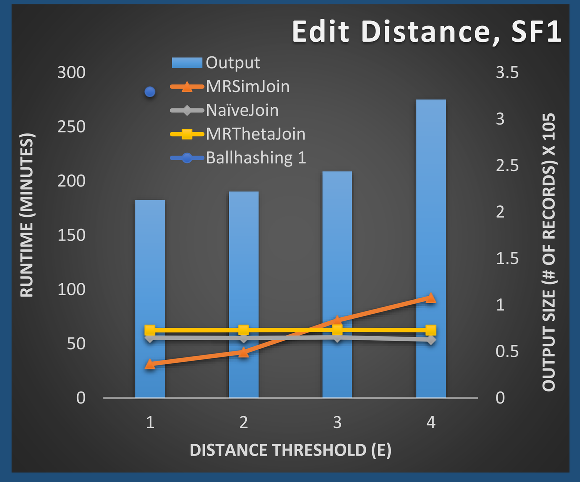
- The classification of Similarity Join techniques based on the supported data types and distance functions
- An extensive set of experimental results:
  - Compare performance based on supported data type and distance function
  - Evaluate performance under various dataset sizes and distance thresholds
- The availability of the authors' open-source implementation of various Similarity Join algorithms [9].

Hadoop Cluster Configuration

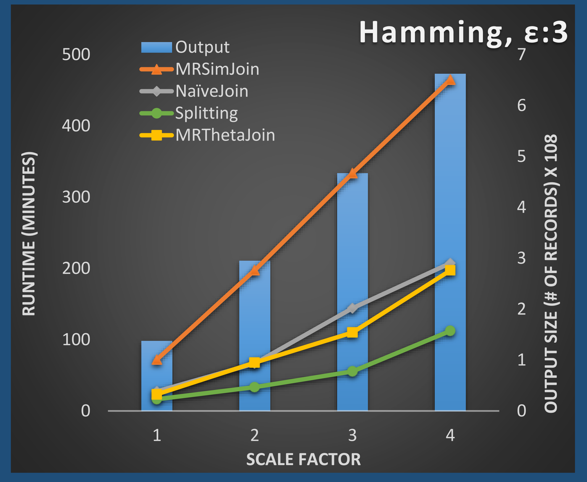
- The experiments were performed using a Hadoop cluster running on the Amazon EC2.
- We used a cluster of 10 nodes.
  - 15 GB of memory.
  - 4 virtual cores with 2 EC2 Compute Units each.
  - 1,690 GB of local in-stance storage.
  - 64-bit platform.
- The number of reducers was computed as: 0.95×⟨no. worker nodes⟩×⟨max reduce tasks per node⟩ = 25.

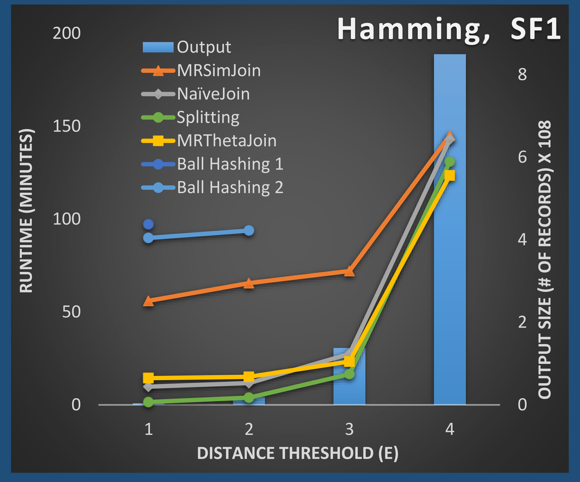
Variable-length Strings – Edit Distance



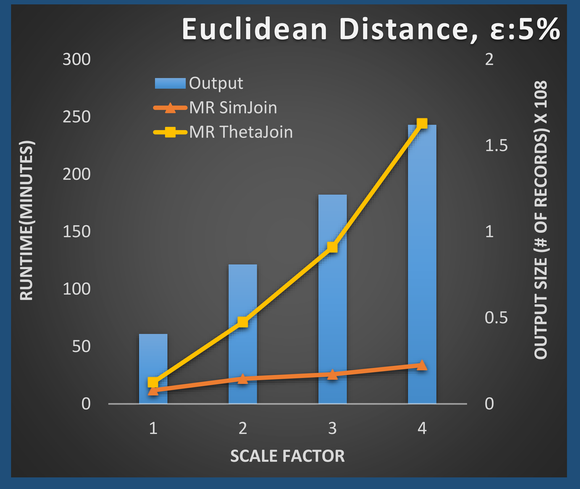


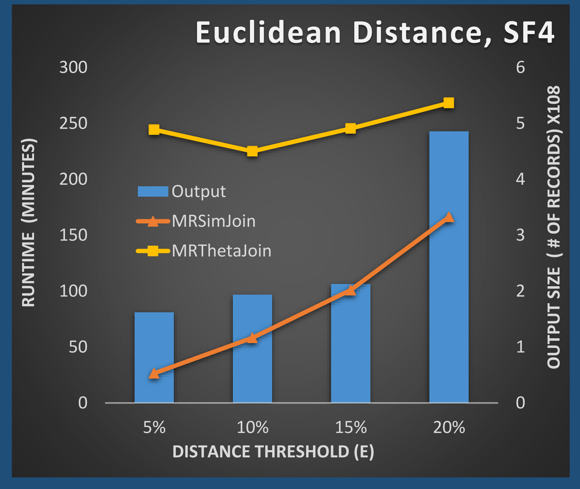
Fixed-length Strings – Hamming Distance



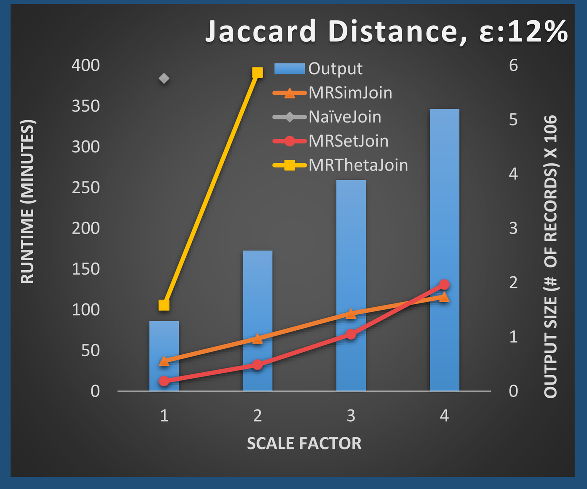


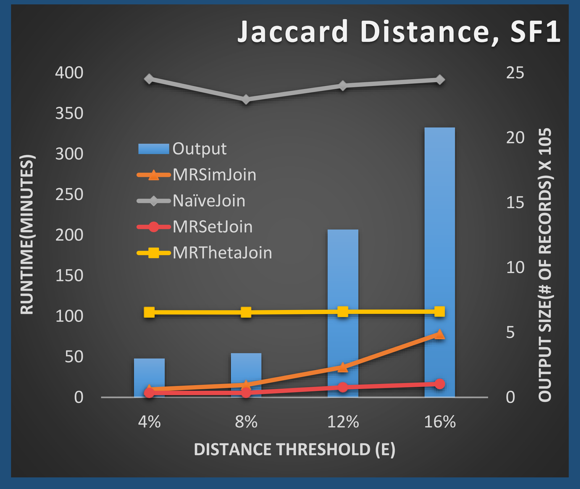
Vector Data (10D) – Euclidean Distance





Set Data – Jaccard Distance





Classification of the Algorithms

Algorithm	Supported Distance/ Similarity Functions	Supported Data Types: Text/String	Numeric	Vector	Set
Naïve Join	Any DF	●	●	*	●
Ball Hashing 1	Hamming Distance Edit Distance	●			
Ball Hashing 2	Hamming Distance Edit Distance	●			
Subsequence	Edit Distance	●			
Splitting	Hamming Distance Edit Distance	●			
Hamming Code	Hamming Distance	●			
Anchor Points	Hamming Distance Edit Distance	●	*	*	
MRThetaJoin	Any DF	●	●	●	●
MRSimJoin	Any metric DF	●	●	●	●
MRSetJoin	JS, TC, CC, Edit Distance*	*			●
Online Aggregation	JS, RS, DS, SC, VC				●
Lookup	JS, RS, DS, SC, VC				●
Sharding	JS, RS, DS, SC, VC				●

● Natively Supported

* Can be extended to support this data type or distance function

JS=Jaccard Similarity, TC=Tanimoto Coefficient, CC=Cosine Coefficient, RS=Ruzicka Similarity, DS=Dice Similarity, SC=Set Cosine Sim., VC=Vector Cosine Sim.

Dataset

- We used a slightly modified version of the Harvard bibliographic dataset [8].
- Attributes: unique ID, title, date issued, record change date, record creation date, Harvard record-ID, first author, all author names, and 10D vector (augmented).
- Minimum and maximum length of attributes:

Unique ID	Title	Date issued	Record change date	Record creation date	Harvard record-ID	First author	All author names
(9, 9)	(6, 996)	(4, 4)	(15, 15)	(6, 6)	(10, 10)	(6, 94)	(6, 2462)

- Scale Factor 1 (SF1) dataset contains 200K records.
- Larger datasets were generated in such a way that the number of matches of any SJ in SF*N* is *N* times the number of matches in SF1.
- The records of each dataset are equally divided between tables *R* and *S*.

Key Findings

- The algorithms vary significantly in terms of Supported distance functions.
  - MRSimJoin and MRThetaJoin support multiple metrics.
  - Subsequence and Hamming Code support only one.
- No single algorithm outperforms all the others for all the evaluated data types and distance functions.
  - In some cases, an algorithm performs consistently better than the others for a given data type and metric.
  - In others, the identification of the best algorithm depends on the dataset size and distance threshold.

References

1. R. Vernica, M. J. Carey, and C. Li. 2010. Efficient Parallel Set-similarity Joins using MapReduce. In SIGMOD, 2010.
2. Y. N. Silva, J. M. Reed, L. M. Tsosie. MapReduce-based Similarity Join for Metric Spaces. In VLDB/Cloud-I, 2012.
3. Y. N. Silva, J. M. Reed. Exploiting MapReduce-based Similarity Joins. In SIGMOD, 2012.
4. F. N. Afrati, A. D. Sarma, D. Menestrina, A. Parameswaran, and J. D. Ullman. Fuzzy Joins Using MapReduce. In ICDE, 2012.
5. A. Okcan and M. Riedewald. Processing Theta-joins using Mapreduce. In SIGMOD, 2011.
6. A. Metwally and C. Faloutsos. V-SMART-join: a Scalable MapReduce Framework for All-pair Similarity Joins of Multisets and Vectors. In VLDB, 2012.
7. C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient Similarity Joins for Near Duplicate Detection. In WWW, 2008.
8. Harvard Library. Harvard bibliographic dataset. http://library.harvard.edu/open-metadata.
9. SimCloud Project. MapReduce-based Similarity Join Survey. http://www.public.asu.edu/~ynsilva/SimCloud/SJSurvey.

This work was supported by Arizona State University's SRCA and NCUIRE awards, the NSFC (No. 61402329), and the China Scholarship Council.