MapReduce-based Similarity Join Survey

Faculty: Yasin Silva Students: Kyle Gervais, A.J. Wadsworth, Jason Reed (alumnus)
Arizona State University

Motivation

The Problem

- Big-Data systems have been introduced to efficient 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 have been proposed but most of these techniques were developed in parallel and thus did not compare with alternative approaches.
- 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 core goal of the proposed project is to address this problem by focusing on the study, classification and benchmarking of all the SJ techniques proposed for Big-Data systems.
- Open source implementation of all the algorithms using the Hadoop Map-Reduce platform (consider the main framework to process Big Data)

Classification of Algorithms

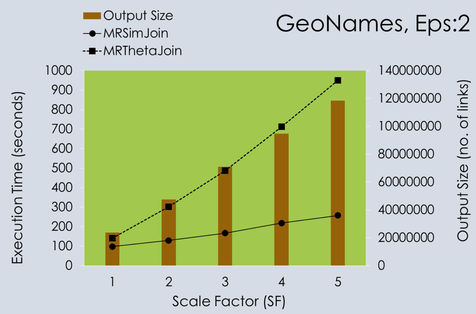
Papers	Algorithm	Compatible Distance Functions	Data Type			
			Text/Strings	Numeric	Vector	Set
Fuzzy Joins	FJ-Naïve	Any	Y	Y	Y	Y
	Ball Hashing 1	HD, ED	Y			
	Ball Hashing 2	HD, ED	Y			
	Subsequence	ED	Y			
	Splitting	HD, ED	Y			
	Hamming Code (specific case)	HD	Y			
	Anchor Points	HD, ED	Y	Y*	Y*	
Theta Join	MRThetaJoin	Any	Y	Y	Y	Y
MRSimJoin	MRSimJoin	Any metric DF	Y	Y	Y	Y
Set Join	MRSetJoin	ED, JD, Tanimoto coefficient, cosine coefficient				Y
V-SMART	Online Aggregation	Any set similarity†				Y
	Lookup	Any set similarity†				Y
	Sharding	Any set similarity†				Y

*could reasonably be extended to support this data type
† Set similarities include: Jaccard, Ruzicka (generalization of Jaccard), Dice, cosine, and vector cosine.

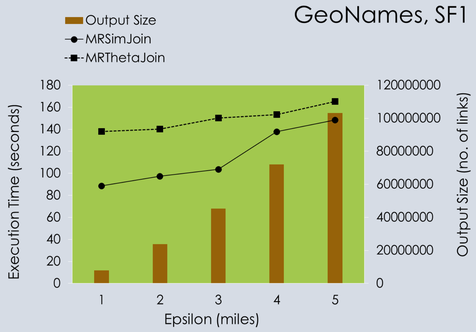
HD: Hamming Distance
ED: Edit Distance
JD: Jaccard Distance

Preliminary Experimental Results (SF1)

Increasing Scale Factor



Increasing Epsilon



Similarity Join Example

Goal: Find pairs of restaurants and movie theaters that are close to each other.
"close" = within ε of one another

Dataset

- We use a real bibliography dataset: Harvard Dataset
- We extracted multiple attributes and generated text attributes for the different data types supported by the algorithms (string, vectors, sets, etc.)
- Number of records in the dataset (SF1): 2,000,000
- The datasets for SF greater than 1 were generated in such a way that the number of links of any SJ operation in SF*N* are *N* times the number of links of the operation in SF1
- For instance, for vector data, the datasets for higher SFs were obtained adding shifted copies of the SF1 dataset such that the separation between the region of new vectors and the region of previous vectors is greater than the maximum distance threshold used in our tests

Types of Experimental Tests

- Increasing distance threshold (ε)
- Increasing data size (scale factor)
- Increasing number of cluster nodes and data size

For String data:

- Increasing length of strings
- Increasing alphabet size
- Variable string length Vs fixed length

For Vector data:

- Increasing dimensionality

Next Steps

- We have classified the algorithms based on the supported distance functions and data types
- We have completed the implementation of all the algorithms using the Hadoop MapReduce framework
- We well focus next on the execution of the tests using Amazon cloud services
- After running all the experimental tests, we will identify cases where specific approaches clearly outperform others
- We plan to write a paper presenting the results of our work

This research project is supported by an Amazon Research grant.