CleanCloud: Cleaning Big Data on Cloud

Conference Paper · November 2017

DOI: 10.1145/3132847.3133187

5 authors, including:

Hongzhi Wang
Harbin Institute of Technology
166 PUBLICATIONS **536** CITATIONS
SEE PROFILE

Ding Xiaoou
Harbin Institute of Technology
4 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Hong Gao
Harbin Institute of Technology
182 PUBLICATIONS **1,170** CITATIONS
SEE PROFILE

CleanCloud: Cleaning Big Data on Cloud

Hongzhi Wang, Xiaoou Ding, Xiangying Chen, Jianzhong Li, Hong Gao
Harbin Institute of Technology
wangzh@hit.edu.cn,dingxiaoou_hit@163.com,1030330622@qq.com,lijzh@hit.edu.cn,honggao@hit.edu.cn

ABSTRACT

We describe CleanCloud, a system for cleaning big data based on Map-Reduce paradigm in cloud. Using Map-Reduce paradigm, the system detects and repairs various data quality problems in big data. We demonstrate the following features of CleanCloud: (a) the support for cleaning multiple data quality problems in big data; (b) a visual tool for watching the status of big data cleaning process and tuning the parameters for data cleaning; (c) the friendly interface for data input and setting as well as cleaned data collection for big data. CleanCloud is a promising system that provides scalable and effect data cleaning mechanism for big data in either files or databases.

KEYWORDS

data cleaning, parallel computing, entity resolution

1 INTRODUCTION

Big data exist widely in many applications, and may contain various quality problems. Two possible reasons lead to these data quality problems. One is that big data are from various data sources and contain inconsistency, conflicts and incompleteness in high possibility. The other is that big data change frequently and are difficult to maintain manually. Data quality problems prevent effective applications of big data since the analysis and query on low-quality data will lead to wrong results or misleading decisions. Cleaning is an intuitive way to increase the usability of big data. Even though many data cleaning techniques have been proposed, they are not suitable for big data cleaning for following reasons.

- Even though data cleaning could be performed offline, Scalability is still an essential topic for big data cleaning. Parallel computation is a common solution to scale algorithms to big data. Even though some parallel data cleaning approaches [11], each of them focuses on one important point of data quality instead of a systemic scalable solution.
- Current data cleaning systems such as [10] focus on one aspect of data quality problem. For big data, it is inefficient to perform each data quality problem with one individual system. The inefficiency is led by two aspects. One is that involving multiple systems for data cleaning require loading and saving big data multiple times. The other is that the cleaning of various data quality problems may share same operations. For example, both of deduplication [4] and currency data discovery [6] require entity resolution. If different quality problems are processed in separated systems, shared operations will be executed multiple times and global optimization is prevented.
- Existing data cleaning methods often require parameters such as rules for inconsistency detection [5] and threshold for entity resolution [13]. For data in small size, these parameters could be determined by manual observation on the data. Due to the variety and large volume of big data, correct parameters could not be determined by only observing a small share of data and it is impossible to explore the whole data set manually.

A natural way for cleaning big data is to adopt cloud computation techniques, which distribute the computation task to multiple machines to reduce the execution time. Thus we develop CleanCloud, a big data cleaning system on cloud.

Since Map-Reduce [2] is a popular programming paradigm that could scale to hundreds and even thousands of machines, it is suitable for the computation on big data. Therefore, we adopt Map-Reduce paradigm in our system and package data cleaning operation into Map-Reduce jobs. Such that, the scalability of data cleaning tasks is ensured by the high-scalability platform such as Hadoop [1] and Spark [2], and data cleaning is provided as a service on the cloud. Thus, the name of our system is CleanCloud.

To achieve high performance, we extract operators in data cleaning and design Map-Reduce algorithms for them. Such that the order of cleaning operations could be arranged according to requirements, and these operations could be combined with other Map-Reduce jobs by sharing the same data flow. The Map-Reduce mechanism and efficient algorithms allows CleanCloud to obtain remarkable scalability for various data cleaning tasks and make data cleaning flexible.

To meet the requirements of supporting various data quality aspects, CleanCloud contains multiple kinds of operations, including entity resolution, rule-based error detection and repair, imputation for missing attributes and truth discovery. The latter three operations cover three major kinds of errors in the data of rule violation, data conflicts and attribute missing. As an instance, the operations of entity resolution, truth discovery and rule-based error detection could be applied to detect outdated data and find current values [6]. To combine these operations to achieve high-quality data cleaning, we design a proper default cleaning order according to the relationship between these operations. Besides, a system also permits users give their own data cleaning operation execution order.

For the convenience of users to input requirements and watch data cleaning process, CleanCloud provides friendly interfaces for non-expert users. The input consists of files or settings of the databases while the clean data could be downloaded directly as

[1] http://hadoop.apache.org/
[2] https://spark.apache.org/

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3133187

files or exported to required databases. The cleaning is performed on the machines required by users. The system is responsible for cleaning the big data according to the user's settings. Besides, the system provides tools for effective and efficient data cleaning including cleaning process's watching and tuning.

The major goal of this demonstration is to show the architecture and functions of CleanCloud. This is achieved through data cleaning algorithms including inconsistency repairing, data imputation, entity resolution and truth discovery. As a secondary goal, we demonstrate the tools for big data cleaning which enhance the cleaning. We show how each step of data cleaning are performed as well the cleaning quality, scalability and running time. It also emphasizes the benefits of CleanCloud including simplicity of dirty data loading and task setting, high performance for big data, effectiveness of complex data cleaning task, and the tools for the convenience of watching and tuning big data cleaning process.

The remaining parts of this paper are organized as follows. Section 2 discusses the functions and architecture of the system. The parallel data cleaning algorithms are introduced in Section 3. The demonstration scenarios are described in Section 4. Section 5 draws the conclusions and discusses the future research directions.

2 SYSTEM OVERVIEW

CleanCloud is built on Hadoop which is suitable for a cluster with hundreds or even thousands machines and provides Map-Reduce mechanism that is easy for parallel programming on the cluster. The data are managed in a distributed file system. The input data could be in form of flat files or databases. The former is uploaded by users and the latter are accessed according to user settings. The nodes in the cluster access dirty data from the distributed file system. During the cleaning, the data are transferred among mappers and reducers. After the data are cleaned, they are sent back to the HDFS. Users could obtain the cleaned data by downloading the cleaned data as a flat file or immigrating to a database according to the setting of users. Additionally, since the input and cleaned data are located in HDFS, our system could be easily embedded to other systems on Hadoop such as data analysis system.

The system has a master and multiple slaves. The master receives the input data and parameters (such as rules, similarity threshold for entity resolution and the number of machines involving in the cleaning) for data cleaning, as well as the plan for data cleaning. The plan is formed with 4 data cleaning operators, rule-based repair, imputation, entity resolution and truth discovery. The definitions are defined as follows.

- **Imputation** fills the missing attributes in tuples according to existing values.
- **Entity Resolution** identifies the tuples referring to the same real-world entity.
- **Truth Discovery** chooses the true value for conflicting values describing the same entity.
- **Rule-based Repair** detects the tuples that violate given rules and revises the values to make the tuples obey the rules.

As discussed in Section 1, these operators cover most of data quality problems. With the flexible architecture in our system, new operators can be added on demand for new data cleaning requirements.

If parameters are not input by the user, the system generates the plan according to the default settings. The plan is executed to generate the cleaned data in the slaves which contain the implementations of these operators. The system also provides a default execution order for data cleaning operations.

For big data, data cleaning may take a long time. During the cleaning, the system provides a interface for users to watch the machines for cleaning with status watch component and tune the parameters by exploring intermediate cleaning results in the data reviewer. For the exploration of intermediate results, the reporter component in each slave samples revised data from the results of operators and submit them to the data reviewer with corresponding setting of the operator.

3 IMPLEMENTATIONS

In this section, we introduce the operators and implementation algorithms used in the system briefly. As discussed in Section 2, all the four operators provided in the system will run on slaves in cloud. They are implemented in Map-Reduce paradigm. In this section, we introduce the implementation algorithms for these operators briefly.

In our system, the operations could be organized flexibly and default execution is provided. The data flow of the default execution order is shown in Figure 1, where each Map-Reduce component corresponds to a pair of map and reduce operations. In this figure, the arrows in and out HDFS represents loading and storing data on HDFS while other a arrows represents the data movement in single machine. The data repartitions are performed within Map-Reduce components.

Imputation We design different algorithms for missing value imputation according to the data type of the attribute [9]. Missing numerical values are imputed with regression on other values in the same attribute. Missing category values are imputed according to the classification on the tuples. Regression involves three operations, standardization, sort and regression, each of which is implemented with a round of Map-Reduce. Standardization normalizes all attributes and computes the minimized value of each attribute as the reference vector $minVec$ of distance metric. Sort step computes relative size of $minVec$ of each record and sorts the results. Regression step performs the regression and fills these values. Classification also has three basic operations, *pComp*, *joinMis* and $impute$, each of which is implemented with a round of Map-Reduce. *pComp* computes the conditional probability of each feature attribute and the marginal probability of category attributes. *JoinMis* aggregates required parameters for filling the same missing attributes together. $Impute$ fills the missing attributes according to the results of *JoinMis*. To reduce the number of rounds, we combine standardization and sort operation with the *pComp*, *joinMis* into a round of Map-Reduce. The operations of regression and $impute$ are merged into a round of Map-Reduce.

Entity Resolution Entity resolution parallelizes EIF [13], which can handle both synonym and homonymy. The algorithm has three phases. The first phrase constructs an attribute index table to make the data object with the same attribute value sharing the same entity in the index entry to form preliminary clusters. This phase is implemented by a round of Map-Reduce. Mappers partition the data

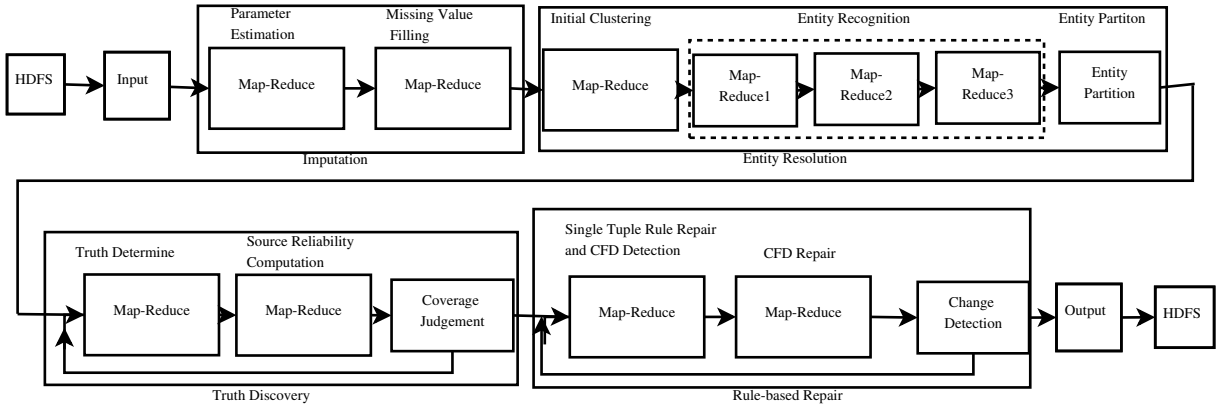

Figure 1: Dataflow of the System

objects according to the attribute value and Reducers group tuples according to the attribute values. The second phase accomplishes the entity resolution by performing the similarity join within each preliminary cluster and outputs the similarity pair set. This phase is implemented with three rounds of Map-Reduces. The first group computes the number of each tuple and pairs of tuples. The second round aggregates the pairs according to the first tuple. Then the similarity of each pair is computed. Then, according to the threshold and the computed similarity, the groups of tuples satisfying the constraint are output in the last phrase. The details of the algorithms are in [7].

Truth Discovery The basic framework of truth discovery algorithm is naive Bayes [8]. The framework computes the true values and the reliability of the data sources iteratively. Each iteration involves two rounds of Map and Reduce. Initially, the reliabilities of all data sources are considered as the same. The first round computes the true value according to the reliabilities of data sources by mapping the values and reliability according to the attributes and id of entity, and reducers votes according to the reliability. The second round maps on data source id and recomputes the reliability of data sources according to the true values. The two rounds are executed iteratively until converge.

Rule-based Repair This operator involves multiple iterations under the consideration of the conflicts of the repairs. In each iteration, the repairs are performed according to all the rules. The iteration halts until no further repair is performed. Each iteration contains two tasks. The first one is the detection and revision with rules for single tuple, e.g. value constraints and format constraints. It is accomplished with single round of Map-Reduce denoted by M_1. Mappers distribute the tuples and rules according to the values in the constraints to machines. Reducers detect errors and repair the tuples according to corresponding rules in parallel. The second task, detection and revision with CFD [5], involves two rounds of Map-Reduce. The first round, which is denoted by M_2, groups and maps the tuples according to corresponding to the values in attributes in rules. Each reducer generates the reparation plans for violated tuples according to rules. Note that in this round a tuple could be mapped to multiple groups and attached multiple reparation plans. In the second round denoted by M_3, mappers partitions the data according to the id of tuples and reducers integrate the repairing plans attached to each tuple and revise the tuples. Since the M_1 and M_2 share the data flow, they are merged into one round of Map-Reduce (Single Tuple Rule Repair and CFD Detection in Figure 1) to reduce I/O and data repartition time.

Optimization Since the truth discovery and rule-based repair phrase involve iterations, to accelerate the processing, inspired by [1], we add cache for reducers in the two rounds to avoid data repartition.

Operation Order Selection Even though the system permits user to choose the order of the cleaning operation, we provide a default execution order as imputation, entity resolution, truth discovery and Rule-based repair, as shown in Figure 1. We choose such order to achieve high-quality cleaning with the following concerns.

- Imputation provides more information for other operations. Additionally, the imputation may involve conflicts and inconsistency which could be cleaned in the following steps. Thus, imputation is selected as the first operation.
- Entity resolution is performed before truth discovery and rule-based repair for two reasons. On one hand, since entity resolution is the base of truth discovery, it should be executed in prior. On the other hand, a part of inconsistencies are caused by the conflicting among the tuples referring to the same entity. Thus, the resolution of entities will help the repair of inconsistency, which is the focus of rule-based cleaning.
- As for truth discovery and rule-base repair, the former resolves inconsistency within the tuples referring to the same entity, and the latter helps to correct the wrongly selected true values according to extra semantics provided by rules. As a result, truth discovery is performed before rule-based repair in the default order.

4 DEMONSTRATIONS

The demonstration of CleanCloud will be performed in Hadoop 0.20.2. We will demonstrate the features of our system for the big data cleaning. We will demonstrate the execution of multiple data cleaning tasks using a Hadoop cluster running on a local cluster with 32 machines. CleanCloud provides a web-based interface for users. We will show how the system cleans data and interacts with the user in this section.

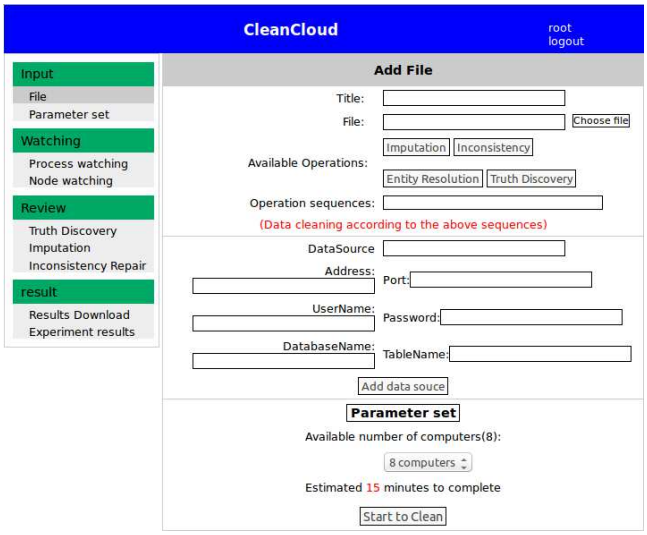

Figure 2: Data Input Form

Input A parallel data cleaning task requires four kinds of input, (1) the data to clean, (2) data cleaning requirements, (3) parameters for data cleaning, and (4) the number of machines to use. A user could input the data to clean by inputting the file name or as remote databases described by connection methods such as ip address and port. The data cleaning requirements could be input as manual cleaning plan for different operators or use default execution order in the system. The parameters for data cleaning operators could be set through an interface or use default settings. After the data are loaded and cleaning requirements are fixed, the computation load is known. For the convenience of user determining the required number of machines, the system provides an estimation of running time with a given number of machines. Thus a user could choose the number according to the estimation. The input interface for task is shown in Figure 2.

Data Cleaning Process Watching The goal of data cleaning process watching is to know the status of data cleaning such that the user could decide whether the cleaning plan and parameters are suitable. The user is permitted to watch the progress at any time during the processing.

Data Cleaning Result Review For the convenience of users to tune the parameters, CleanCloud provides a graphical interface for users to view the intermediate data cleaning results. The interface contains the original data, the cleaning results and the parameters applied for the cleaning, such that the user could change improper values. The parameters are changed according to the changed values, as shown in Figure 3. After the parameter is changed, the user could choose to restart the cleaning with new parameters or apply them on following cleaning process.

Cleaning Result Download When the cleaned results are generated, CleanCloud provides two ways to obtain the clean results. The user can choose to download the result as a text file from the URL provided on the pageor output the cleaned data to some database described by some provided settings of data source similar to the input.

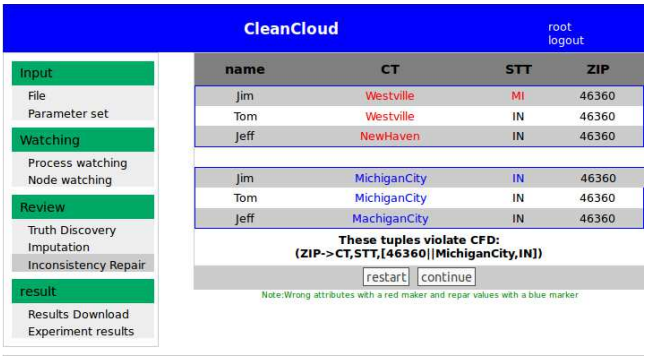

Figure 3: Intermediate Result Review Interface

5 CONCLUSIONS

The demonstration exhibits the benefits for big data cleaning techniques in CleanCloud, and illustrates the suitability of our system for big data. At the same time, the demonstration shows that CleanCloud provides the user with a friendly interface and assists the users to improve the data cleaning process in an interactive way. The future work include discovering the rules and parameters automatically from big data and self-tuning techniques for data cleaning process. Another important future work is to online monitor and repair the errors introduced by updated data.

Acknowledgements. : This paper was partially supported by National Sci-Tech Support Plan 2015BAH10F01, NSFC grant U1509216, 61472099 and the Scientific Research Foundation for the Returned Overseas Chinese Scholars of Heilongjiang Provience LC2016026 and MOE-Microsoft Key Laboratory of Natural Language Processing and Speech, Harbin Institute of Technology.

REFERENCES

[1] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst.. HaLoop: Efficient Iterative Data Processing on Large Clusters. *PVLDB* 3, 1 (2010).
[2] Jeffrey Dean and Sanjay Ghemawat.. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI 2014*.
[3] Amr Ebaid, Ahmed K. Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. NADEEF: A Generalized Data Cleaning System. *PVLDB* 6, 12 (2013).
[4] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.* 19, 1 (2007).
[5] Wenfei Fan. Dependencies revisited for improving data quality. In *PODS 2008*.
[6] Wenfei Fan, Floris Geerts, and Jef Wijsen. Determining the Currency of Data. *ACM Trans. Database Syst.* 37, 4 (2012).
[7] Ran Huo, Hongzhi Wang, Rong Zhu, Jianzhong Li, and Hong Gao. Map-Reduce Based Entity Identification in Big Data. *Journal of Computer Research and Development* 50, z2 (2013).
[8] Li Jia, Hongzhi Wang, Jianzhong Li, and Hong Gao. Incremental Truth Discovery for Information from Multiple Data Sources. In *WAIM Workshops 2013*.
[9] Lian Jin, Hongzhi Wang, Shenbin Huang, and Hong Gao. Missing Value Imputation in Big Data Based on Map-Reduce. *Journal of Computer Research and Development* 50, z1 (2013).
[10] Zuhair Khayyat, Ihab F. Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. BigDansing: A System for Big Data Cleansing. In *SIGMOD 2015*.
[11] Lars Kolb, Andreas Thor, and Erhard Rahm. Dedoop: Efficient Deduplication with Hadoop. *PVLDB* 5, 12 (2012).
[12] Lars Kolb, Andreas Thor, and Erhard Rahm. Load Balancing for Map-Reduce-based Entity Resolution. In *ICDE 2012*.
[13] Lingli Li, Hongzhi Wang, Hong Gao, and Jianzhong Li. EIF: A Framework of Effective Entity Identification. In *WAIM 2010*.