An Efficient MapReduce-Based Apriori-Like Algorithm for Mining Frequent Itemsets from Big Data

Ching-Ming Chao[1], Po-Zung Chen[2], Shih-Yang Yang[3(✉)], and Cheng-Hung Yen[1]

[1] Department of Computer Science and Information Management, Soochow University, Taipei 100, Taiwan
[2] Department of Computer Science and Information Engineering, Tamkang University, Taipei, Taiwan
[3] Department of Media Art and Management of Information System, University of Kang Ning, Taipei, Taiwan
shihyang@ukn.edu.tw

Abstract. Data mining can discover valuable information from large amounts of data so as to utilize this information to enhance personal or organizational competitiveness. Apriori is a classic algorithm for mining frequent itemsets. Recently, with rapid growth of the Internet as well as fast development of information and communications technology, the amount of data is augmented in an explosive fashion at a speed of tens of petabytes per day. These rapidly expensive data are characterized by huge amount, high speed, continuous arrival, real-time, and unpredictability. Traditional data mining algorithms are not applicable. Therefore, big data mining has become an important research issue.

Clouding computing is a key technique for big data. In this paper, we study the issue of applying cloud computing to mining frequent itemsets from big data. We propose a MapReduce-based Apriori-like frequent itemset mining algorithm called Apriori-MapReduce (abbreviated as AMR). The salient feature of AMR is that it deletes the items of itemsets lower than the minimum support from the transaction database. In such a way, it can greatly reduce the generation of candidate itemsets to avoid a memory shortage and an overload to I/O and CPU, so that a better mining efficiency can be achieved. Empirical studies show that the processing efficiency of the AMR algorithm is superior to that of another efficient MapReduce-based Apriori algorithm under various minimum supports and numbers of transactions.

Keywords: Data mining · Frequent itemsets · Big data · MapReduce Apriori

1 Introduction

Currently in the age of information explosion, data are used in various aspects of one's daily life. However, the speed of data generation and storage has far exceeded that of analysis and digestion people could achieve. Therefore, it is an important issue of how

J.-L. Chen et al. (Eds.): WiCON 2018, LNICST 264, pp. 76–85, 2019.
https://doi.org/10.1007/978-3-030-06158-6_8

to use information technology to analyze large amounts of data to discover and apply valuable information in order to enhance personal or organizational competitiveness. Data mining comes into the play. One of the important problems in data mining is discovering frequent itemsets from a set of transactions where each transaction contains a set of items [1]. A frequent itemset is a set of items that frequently occur together in a set of transactions. Apriori is a classic algorithm for mining frequent itemsets [2].

With rapid growth of the Internet as well as fast development of information and communications technology, such as the emergence of Web 3.0, the amount of data is augmented in an explosive fashion at a speed of tens of petabytes (1 PB = 1024 TB) per day. The global big data research report published by McKinsey Global Institute in May 2011 indicated that global enterprise data had increased 7 exabytes (1 EB = 1024 PB) while the new data consumers stored in their personal computers (including notebooks) and mobile devises had exceeded 6 exabytes in 2010 alone [3]. According to statistics by International Data Center, the information capacity of digital world will grow from 0.8 zettabytes (1 ZB = 1024 EB) in 2009 to 44 zettabytes in 2020, i.e. a growth of 1 PB in every 15 s and an annual compound growth rate up to 40% [4]. These rapidly expansive data are characterized by huge amount, high speed, continuous arrival, real-time, and unpredictability, which suggest that manual analysis methods and conventional data mining algorithms are no longer applicable [2, 5–7]. Therefore, big data mining has become an important research issue.

The emergence of big data, the pervasion of networks, and the rapid development of personal computers, servers, and mobile devices all together have called for the needs of applying cloud computing to big data mining. It is a great challenge to properly distribute and manage resources, effectively analyze data to discover interesting patterns, and promptly respond to the user. Applying cloud computing to big data mining can have many practical applications. For example, it can be used in medical treatment to analyze examined items (blood, urine, X-ray, magnetic resonance imaging, etc.) and medicine prescriptions, as instructional materials of examined items and medicine prescriptions to a specific disease for interns [8]. The network management personnel may rapidly analyze to reveal user behavior and abnormal flow through monitoring the cloud network flow to insure the quality of networks as well as to find out the primary reasons for internet instability from analyzed results to insure the stable and smooth operation of networks [9].

Clouding computing is a key technique for big data. In this paper, therefore, we study the issue of applying cloud computing to mining frequent itemsets from big data. We proposed MapReduce-based Apriori-like frequent itemset mining algorithm called Apriori-MapReduce (AMR). The salient feature of AMR is that it deletes the items of itemsets lower than the minimum support from the transaction database. In such a way, it can greatly reduce the generation of candidate itemsets to avoid a memory shortage and an overload to I/O and CPU, so that a better mining efficiency can be achieved. Empirical studies show that the processing efficiency of the AMR algorithm is superior to that of another efficient MapReduce-based Apriori algorithm under various minimum supports and numbers of transactions.

2 AMR Algorithm

The AMR algorithm is an improvement of the Apriori algorithm [1]. It overcomes the drawback of generating too many candidate itemsets of the traditional Apriori algorithm by deleting the items of itemsets lower than the minimum support from the transaction database each time after generating frequent itemsets of a certain length. It is safe to do so because any itemsets generated by adding more items to itemsets lower than the minimum support cannot be frequent.

The basic approach of the AMR algorithm is generally the same as that of the traditional Apriori algorithm. The difference is that AMR uses MapReduce to distribute transaction data to multiple Map nodes that separately find candidate itemsets and combine the candidate itemsets returned from these nodes to generate frequent itemsets. Then, the items of itemsets lower than the minimum support are deleted from the transaction database. If the transaction database is empty or no frequent itemsets can be generated, the algorithm stops execution.

2.1 Details of Algorithm

Figure 1 shows the AMR Algorithm proposed in this paper. Input a transaction database and a prespecified minimum support, the AMR Algorithm will output all of the frequent itemsets in the transaction database.

Algorithm: Apriori-MapReduce (AMR)

Input:

- D, a transaction database;
- min_sup, the prespecified minimum support.

Output: L, frequent itemsets in D.

Method:

```
k=0;
repeat {
(1)k++;
   //Map-function:
(2) uniformly distribute transactions in D to each Map node m;
(3) C_mk = {candidate itemsets of length k at node m};
   // Reduce-function:
   // compute C_k with all C_mk
(4) C_k = {candidate itemsets of length k};
   // remove itemsets with support < min_sup from C_k
(5) L_k = {frequent itemsets of length k};
(6) remove items of itemsets with support < min_sup from D;
(7)} until (D == ϕ) or (L_k == ϕ)
   return L = ∪_k L_k;
```

Fig. 1. AMR algorithm

Definitions of parameters in the algorithm are as follows:
k: length of an itemset
m: Map node
Cmk: candidate itemsets of length k generated by the Map node m
Ck: candidate itemsets of length k
Lk: frequent itemsets of length k
Steps of the algorithm are described as follows:

(1) Start with itemsets of length 1.
(2) Uniformly distribute transactions in the transaction database to each Map node.
(3) Separately compute candidate itemsets of length k in the transactions of each node Cmk.
(4) Combine candidate itemsets generated by each node to generate candidate itemsets of length k Ck.
(5) Eliminate candidate itemsets lower than the minimum support from Ck to generate frequent itemsets of length k Lk.
(6) Delete the items of itemsets lower than the minimum support from the transaction data-base.
(7) If there are no transactions in the transactions database or no frequent itemsets in Lk, terminate the execution the algorithm; otherwise, increase the length of itemsets by 1 and repeat steps 2 to 6.

2.2 Example

Table 1 shows a transaction database consisting of 10 transactions. Suppose the minimum support is 3 and there are 3 Map nodes

Table 1. Transaction database

TID	List of items
T001	cookies, cola, pizza
T002	milk, coffee, bread
T003	cookies, bread
T004	bread, milk, coffee
T005	cookies, bread, milk
T006	coffee, cookies, cola
T007	pizza, cola, beer
T008	cookies, coffee, cola
T009	beer, milk,
T010	cookies, milk, bread

First, start with itemsets of length 1. Uniformly distribute the 10 transactions in the transaction database to 3 Map nodes, so that Map1 has transactions T001, T002, T003, and T004, Map2 has T005, T006, and T007, and Map3 has T008, T009, and T010. Each node separately computes candidate itemsets of length 1 Cm1.

C11 = {<bread, 3>, <coffee, 2>, <cola, 1>, <cookies, 2>, <milk, 2>, <pizza, 1>}

C21 = {<beer, 1>, <bread, 1>, <coffee, 1>, <cola, 2>, <cookies, 2>, <milk, 1>, <pizza, 1>}

C31 = {<beer, 1>, <bread, 1>, <coffee, 1>, <cola, 1>, <cookies, 2>, <milk, 2>}

Combine C11, C21, and C31 to generate candidate itemsets of length 1 C1

C1 = {<beer, 2>, <bread, 5>, <coffee, 4>, <cola, 4>, <cookies, 6>, <milk, 5>, <pizza, 2>}

Eliminate candidate itemsets lower than the minimum support from C1 to generate frequent itemsets of length 1 L1

L1 = {<bread, 5>, <coffee, 4>, <cola, 4>, <cookies, 6>, <milk, 5>}

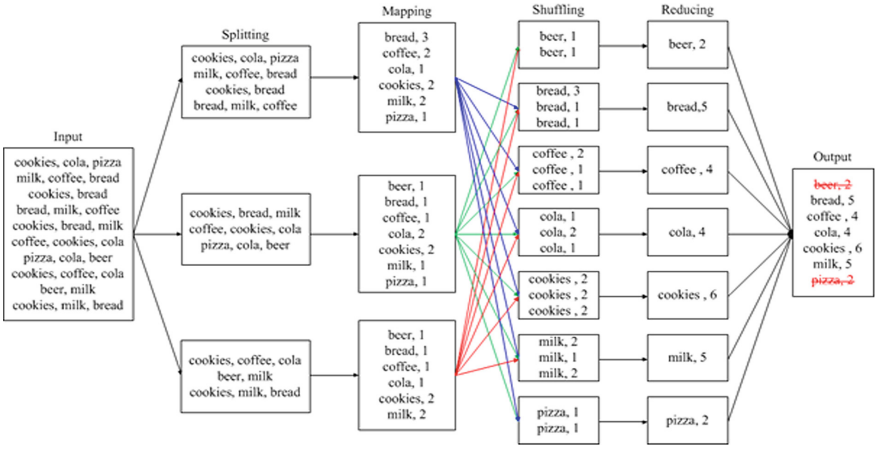

Fig. 2. Generating frequent itemsets of length 1

Figure 2 shows the process of generating frequent itemsets of length 1.

Delete the items in C_1 that are lower than the minimum support from the transaction database. Since there are transactions in the transaction database and frequent itemsets in L_1, increasethe length of itemsets by 1 to become 2. Uniformly distribute the 10 transactions in the transaction database to 3 Map nodes, so that Map1 has transactions T001, T002, T003, and T004, Map2 has T005, T006, and T007, and Map3 has T008, T009, and T010. Each node separately computes candidate itemsets of length 2 C_{m2}.

C_{12} = {<{bread, coffee}, 2>, <{bread, cookies}, 1>, <{bread, milk}, 2>, <{coffee, milk}, 2>, <{cola, cookies}, 1>}

C_{22} = {<{bread, cookies}, 1>, <{bread, milk}, 1>, <{coffee, cola}, 1>, <{coffee, cookies}, 1>, <{cola, cookie}, 1>, <{cookies, milk}, 1>}

C_{32} = {<{bread, cookies}, 1>, <{bread, milk}, 1>, <{coffee, cola}, 1>, <{coffee, cookies}, 1>, <{cola, cookie}, 1>, <{cookies, milk}, 1>}

Combine C_{12}, C_{22}, and C_{32} to generate candidate itemsets of length 2 C_2.

C_2 = {<{bread, coffee}, 2>, <{bread, cookies}, 3>, <{bread, milk}, 4>,<{coffee, cola}, 2>, <{coffee, cookies}, 2>, <{coffee, milk}, 2>, <{cola, cookies}, 3>, <{cookies, milk}, 2>}

Eliminatecandidate itemsets lower than the minimum support from C_2 to generate frequent itemsets of length 2 L_2.

L_2 = {<{bread, cookies}, 3>, <{bread, milk}, 4>, <{cola, cookies}, 3>}

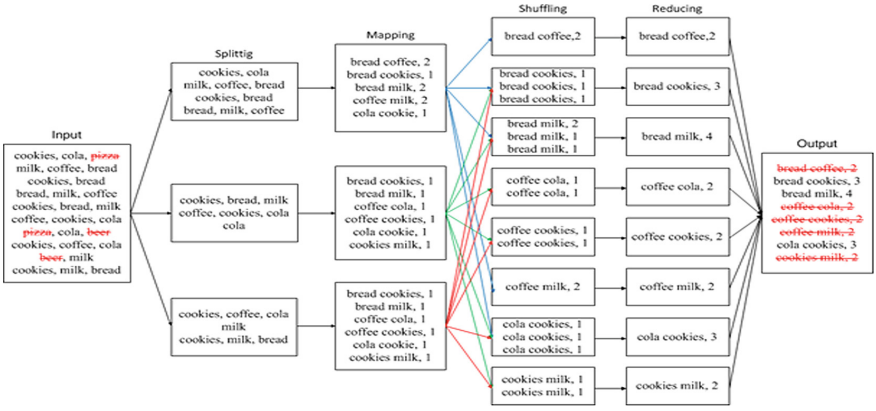

Fig. 3. Generating frequent itemsets of length 2

Figure 3 shows the process of generating frequent itemsets of length 2.

Delete the itemsof itemsets in C_2 that are lower than the minimum support from the transaction database. Since there are transactions in the transaction database and frequent itemsets in L_2, increasethe length of itemsets by 1 to become 3. Uniformly distribute the 10 transactions in the transaction database to 3 Map nodes, so that Map1 has transactions T001, T002, T003, and T004, Map2 has T005, T006, and T007, and Map3 has T008, T009, and T010. Each node separately computes candidate itemsets of length 3 C_{m3}.

C_{13} = { }

C_{23} = { }

C_{33} = { }

Combine C_{13}, C_{23}, and C_{33} to generate candidate itemsets of length 3 C_3.

C_3 = { }

Eliminatecandidate itemsets lower than the minimum support from C_3 to generate frequent itemsets of length 3 L_3.

L_3 = { }

Figure 4 shows the process of generating frequent itemsets of length 3.

Since there is no frequent itemset in L3, terminate the execution the algorithm. Finally, by using the AMR algorithm, the frequent itemsets generated from the transaction database in Table 1 are as follows:

L = L1 ∪ L2 ∪ L3 = {<bread, 5>, <coffee, 4>, <cola, 4>, <cookies, 6>, <milk, 5>, <{bread, cookies}, 3>, <{bread, milk}, 4>, <{cola, cookies}, 3>}

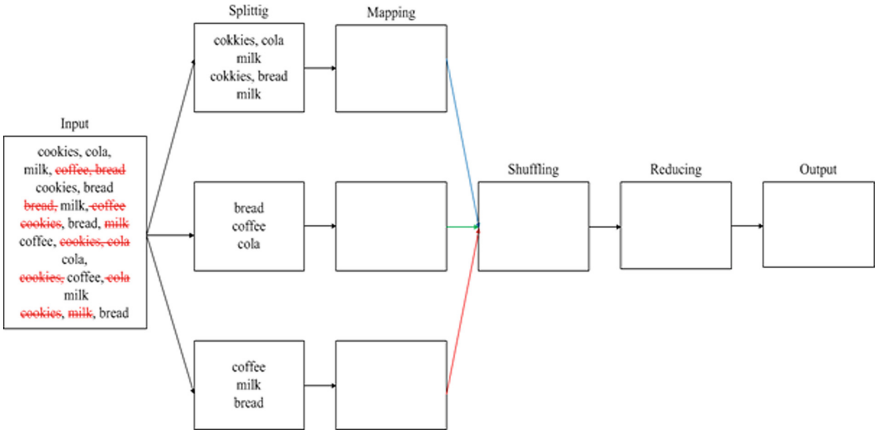

Fig. 4. Generating frequent itemsets of length 3

3 Performance Evaluation

3.1 Experiment Environment and Data

The experiment environment uses two HP servers to generate eight virtual computers for performance test. One of the virtual computers serves as Master and the remaining seven virtual computers as Slaves. Each virtual computer is equipped with one CPU (2 thread) and 2 GB memory. The development tool Eclipse is used to write programs.

We use the 2013 whole year meteorological data at Taipei, Danshui, Keelung, and Branchial from Central Weather Bureau as the realdataset for experiments.It consists of 52600 data records and each data record includes atmospheric pressure, air temperature, dew point temperature, relative moisture, average wind speed, maximum average wind speed, and maximum instantaneous wind speed. In addition, we use IBM Quest Synthetic Data Generator to generatesynthetic datasets. A synthetic dataset has three parameters, of which T denotes the number of transactions, L denotes the average length of transactions, and N denotes the number of different kinds of items. For example, T1000KL5N0.5 K denotes that this synthetic dataset consists of one million transactions, the average length of transactions is 5, and there are 500 different kinds of items.

3.2 Comparison and Analysis

Because both Apriori-Map/Reduce and AMRare MapReduce-based Apriori-like algorithms for mining frequent itemsets, we compare and analyze these two algorithms with respect to their processing efficiency under various minimum supports and numbers of transactions, respectively.

Processing efficiency under various minimum supports. Figures 5 and 6 show the comparison of processing efficiency between AMR and Apriori-Map/Reduce under various minimum supports, where the X-axis is the minimum support (%) and the Y-

axis is the execution time in seconds. The experiment data for Fig. 5 is a synthetic database T1000KL6N0.5 K. The experiment data for Fig. 6 is taken from the real dataset with 52000 data records and of data length 7. As shown in Figs. 5 and 6, the processing efficiency of AMR is superior to that of Apriori-Map/Reduce under various minimum supports. It is because that AMR deletes the items of itemsets lower than the mini-mum support from the transaction database, which can greatly reduce the generation of candidate itemsets so as to reduce database scanning time.

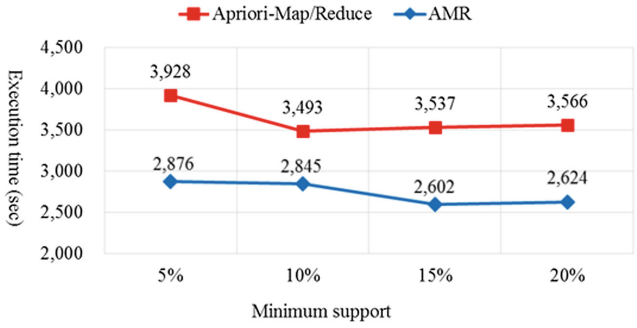

Fig. 5. Processing efficiency under various minimum supports (synthetic dataset)

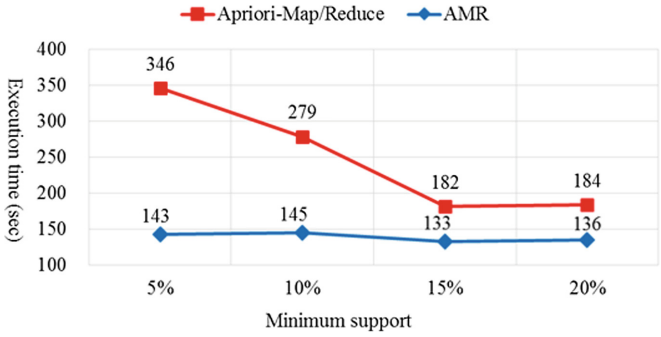

Fig. 6. Processing efficiency under various minimum supports (real dataset)

Processing efficiency under various numbers of transactions. Figures 7 and 8 show the comparison of processing efficiency between AMR and Apriori-Map/Reduce under various numbers of transactions, where the X-axis is the number of transactions in thousands and the Y-axis is the execution time in seconds. The experiment data for Fig. 7 are four synthetic datasets T1000KL8N0.5 K, T5000KL8N0.5 K, T10000 K L8N0.5 K and T20000KL8N0.5 K. The experiment data for Fig. 8 are four datasets taken from the real dataset with 5000, 10000, 25000, and 52000 data records, respectively, and a data length 7. As shown in Figs. 7 and 8, the processing efficiency of AMR is superior to that of Apriori-Map/Reduce under various numbers of transactions, and the difference becomes more prominent as the number of transactions gets

larger. It is because that AMR deletes the items of itemsets lower than the minimum support from the transaction database, which can greatly reduce the generation of candidate itemsets so as to reduce database scanning time.

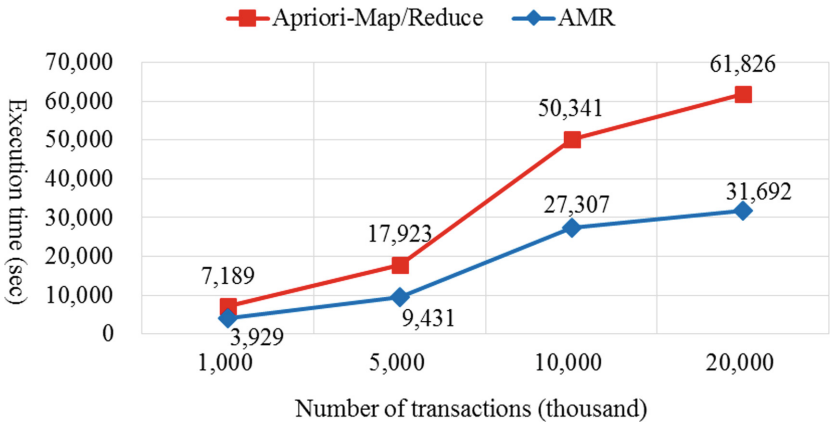

Fig. 7. Processing efficiency under various numbers of transactions (synthetic dataset)

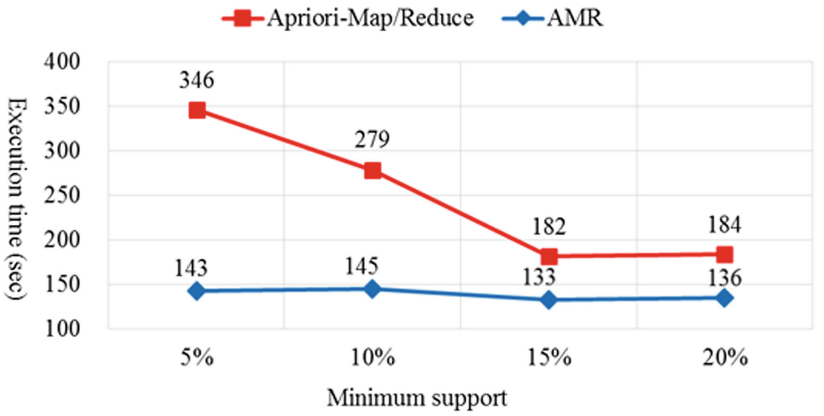

Fig. 8. Processing efficiency under various numbers of transactions (real dataset)

4 Conclusion

In this paper, we have proposed the AMR algorithm that applies cloud computing to big data mining. It can efficiently mine big data under the framework of MapReduce. AMR differs from Apriori-Map/Reduce in that, after the generation of frequent itemsets, it deletes the items of itemsets lower than the minimum support from the transaction database to avoid a memory shortage and an overload to I/O and CPU, so that a

better mining efficiency can be achieved. Empirical studies show that the processing efficiency of the AMR algorithm is superior to that of the Apriori-Map/Reduce algorithm under various minimum supports and numbers of transactions.

References

1. Agarwal R., Srikant, R.: Fast algorithms for mining association rules in large database. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499, Santiago de Chile (1994)
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the 21st ACM SIGMOD-SIGACT-AIGART Symposium on Principles of Database Systems, pp. 1–16, Madison, WI, June 2002
3. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H.: Big data: the next frontier for innovation, competition, and productivity. McKinsey Global Institute, New York (2011)
4. Turner, V., Gantz, J.F., Reinsel, D., Minton, S.: The digital universe of opportunities: rich data and the increasing value of the internet of things. In: International Data Corporation, White Paper, IDC_1672, May 2014
5. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Towards an adaptive approach for mining data streams in resource constrained environments. In: Proceedings of the 2004 International Conference on Data Warehousing and Knowledge Discovery, pp. 189–198, Zaragoza, Spain, September 2004
6. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. ACM Sigmod Record **34**(2), 18–26 (2005)
7. Golab, L., Ozsu, T.M.: Issues in data stream management. ACM Sigmod Record **32**(2), 5–14 (2003)
8. Wang, F., Ercegovac, V., Syeda-Mahmood, T., et al.: Large-scale multimodal mining for healthcare with MapReduce. In: Proceedings of the 1st ACM International Health Informatics Symposium, pp. 479–483, New York, November 2010
9. Lin, R.C.H., Liao, H.J., Tung, K.Y., Lin, Y.C., Wu, S.L.: Network traffic analysis with cloud platform. J. Internet Technol. **13**(6), 953–961 (2012)