information

MDPI

Article

A Feature Selection Method for Large-Scale Network Traffic Classification Based on Spark

Yong Wang [1,2], Wenlong Ke [3] and Xiaoling Tao [2,3,*]

[1] Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China; ywang@guet.edu.cn

[2] Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, Guilin University of Electronic Technology, Guilin 541004, China

[3] Key Laboratory of Cognitive Radio and Information Processing, Guilin University of Electronic Technology, Guilin 541004, China; wenlongke123@163.com

* Correspondence: txl@guet.edu.cn; Tel.: +86-0773-2291076

Academic Editors: Yong Yu and Yu Wang
Received: 20 December 2015; Accepted: 29 January 2016; Published: 15 February 2016

Abstract: Currently, with the rapid increasing of data scales in network traffic classifications, how to select traffic features efficiently is becoming a big challenge. Although a number of traditional feature selection methods using the Hadoop-MapReduce framework have been proposed, the execution time was still unsatisfactory with numeral iterative computations during the processing. To address this issue, an efficient feature selection method for network traffic based on a new parallel computing framework called Spark is proposed in this paper. In our approach, the complete feature set is firstly preprocessed based on Fisher score, and a sequential forward search strategy is employed for subsets. The optimal feature subset is then selected using the continuous iterations of the Spark computing framework. The implementation demonstrates that, on the precondition of keeping the classification accuracy, our method reduces the time cost of modeling and classification, and improves the execution efficiency of feature selection significantly.

Keywords: feature selection; Fisher score; sequential forward search; MapReduce; Spark

1. Introduction

Network traffic classification [1] refers to the classification of the bidirectional TCP/UDP flows generated by network communications according to the application types such as WWW, P2P, FTP, and ATTACK. As an important preprocessing of traffic classification, the feature selection aims to select a feature subset that has the lowest dimension and also retains classification accuracy so as to optimize the specific evaluation criteria. Due to the significant effect on the performance network traffic selection in the current big data environment, the feature selection has become a key research area in such fields as machine learning, data mining and pattern recognition [2].

Current feature selection methods can be categorized into two groups, according to whether the subset evaluation criteria depend on the subsequent classification algorithm: filter and wrapper. The filter methods take the feature selection process and the subsequent machine learning method into consideration separately, and make a selection on the basis of the characteristics of the dataset itself. They have a fast operation, but lead to poor performance in selection. The wrapper methods take the subsequent classification algorithm as the evaluation index of the feature subset; they achieve higher classification accuracy, but result in low efficiency and a large amount of computation. Thus, how to integrate the two methods to bridge the gap between the accuracy and the executing efficiency is a challenge in the research of feature selection [3].

With the network bandwidth and the number of users growing continuously, the size of network data has been increasing rapidly; thus, improving the efficiency of the feature selection method has become an urgent demand. Some researchers parallelized and improved the traditional feature selection methods using the Hadoop-MapReduce framework, which enabled it to deal with super large-scale data. However, computation complexity of such solutions cannot be ignored. Recently, more and more research shows that the MapReduce computational model is suitable for processing jobs with large amounts of data but uncomplicated computation. In terms of complex computation, like the wrapper methods with many iterations, the MapReduce framework has inferior execution efficiency.

In order to solve the aforementioned problems, in this paper, we put forward a feature selection method based on Spark (FSMS). In our approach, a filter method called Fisher score is used to process the complete feature set, eliminating features with a low distinguishing degree and worse discrimination performance. We also adopt the concept of the wrapper method, in which the sequential forward search is used as the searching strategy, the classification accuracy is used as the evaluation criterion, and the feature combination with the strongest classification ability is continuously selected using the Spark computing framework. Finally, the optimal feature subset for traffic classification is acquired.

2. Related Work

The evolution of Internet applications has made traditional methods for classifying network traffic progressively less effective [4]. Port-based methods can easily misclassify traffic flows, mostly because of new applications randomly selecting port numbers. Current research efforts have formed a new trend of using machine learning methods to classify network traffic based on flow statistical features [5]. The combination algorithms have been proved to allow a further improvement in terms of classification accuracy with multi-classification [6]. Thus, trying to use the combination algorithms to select the most significant and effective features is worthwhile.

The filter feature selection methods measure features by analyzing their internal characteristics. The classic methods include information gain [7], Fisher score [8], Relief-F [9], *etc.* The wrapper methods adopt different strategies on the feature subset, but one evaluation criterion on the candidate subset. Rodrigues *et al.* [10] took the bat algorithm (BA) as the guidance and the optimum path forest classifier (OPF) as the evaluation function to select the optimal feature subset. Chuang *et al.* [11] proposed a better classification result by combining binary particle swarm optimization (BPSO) with a genetic algorithm (GA) and using *k*-Nearest Neighbors (*k*-NN) as the classifier to select the feature. Both filter and wrapper methods have their own advantages and disadvantages, which are complementary to each other. Peng *et al.* [12] integrated the two approaches into a sequential searching approach to improve the classification performance. The hybrid feature selection method generally filters out some of the unrelated or noisy features using the filter methods first, and subsequently selects the optimal feature subset using the wrapper methods.

With the data scale increasing rapidly, how to carry out the feature selection for big data becomes a critical issue. Appropriate and novel designs for highly parallel low-cost architectures promise significant scalability improvements [13]. The feature selection and traffic classification systems must be redesigned to run on multicore hardware, targeting low-cost but highly parallel architectures [14]. At first, the Hadoop-MapReduce framework had become a research focus. Sun *et al.* [15] introduced the joint mutual information method into the feature selection. The high efficiency and expansibility of the method was tested by experiments. Based on the cloud model, Li *et al.* [16] proposed an improved method of SVM-BPSO feature selection, in which the wrapper evaluation strategy was adopted, and the local optimal solution was achieved with a faster convergence rate, and a better result was thus obtained. Long [17] parallelized and improved the feature selection method using the Hadoop-MapReduce framework to enable it to deal with large-scale data. However, Srirama *et al.* [18,19] pointed out that the MapReduce model was suitable for processing jobs with large amounts of data but uncomplicated computing. In order to deal with the complex iterative

computations, the MapReduce model had lower execution efficiency. Spark [20] is a new-generation parallel-computing framework based on memory, which performs better than MapReduce in the iterative computation. Currently, there are limited research outcomes on Spark in the field of feature selection of network traffic. This paper puts forward a feature selection method for network traffic based on Spark and investigates its performance and effect.

3. The Parallel Computing Framework

3.1. Hadoop-MapReduce

Hadoop is a distributed open-source software framework for the clustered environment and large-scale dataset processing. It provides MapReduce as the core programming interface and Hadoop Distributed File System (HDFS), which can process as many as one million nodes and data with the magnitude of Zettabyte (ZB). Among them, MapReduce is a computing framework used for large-scale data processing, by which the parallel computing is abstracted in two stages, namely Map and Reduce, to deal with data by mapping and protocol. Having many advantages such as simple programming, ease of extension and good fault-tolerance, MapReduce became the most popular parallel processing strategy at present.

However, much research proves that the MapReduce computing framework is suitable for processing the job with large amounts of data but uncomplicated core computing. Faced with many instances of complex iterative computations, the following problems arise: (1) It only provides two expressions for Map and Reduce, which are hard to fully describe the complex algorithm; (2) although the algorithm is the same for each iteration, the implementation of the MapReduce operation must be renewed at each iteration, which will incur unnecessary system overhead; (3) even though the input data is slightly varied at each iteration, MapReduce will read the complete data again from HDFS, which will take up enormous time, CPU resources, and network bandwidth.

3.2. Spark

Spark is a parallel computing framework for big data based on memory computation. Its framework adopts the master-slave model in distributed computing. A master is a node containing the master process and slaves are nodes containing the worker process. Acting as the controller of whole clusters, the master node is responsible for the normal operation of the whole clusters. The worker node, which is equivalent to a computing node, receives commands from the master node and generates a status report. The executor is responsible for the task execution. As the user's client, the client is responsible for submission and application. The driver is responsible for controlling the execution of an application.

The advantages of Spark in iterative computations are mainly described as follows: Firstly, a Spark operation based on a resilient distributed dataset (RDD) not only realizes the operator map and reduces function in MapReduce, but also provides ten times as many operators, which can fully describe the iterative algorithm. Secondly, several rounds of iteration can be integrated into one Spark operation. At the beginning of the operation, the raw data will be read into the memory, from which the corresponding data is transferred directly at each iteration to perform multiple calls to the data after reading at one time. Thirdly, the intermediate result of computation is also stored in memory, but is not written to disk in order to avoid an I/O operation on disk with quite low efficiency, which improves the operation efficiency significantly.

4. FSMS

4.1. The Filter Methods for Feature Selection Based on Fisher Score

The filter-based feature selection method takes the feature selection process and the subsequent machine learning method into consideration separately and selects the feature subset based on the

characteristics of data itself. The Fisher score, as a high-efficient filter-based supervised feature selection method, according to the feature selection criteria of the minimum intra-cluster distance and the maximum inter-cluster distance, evaluates and sorts the features by the internal properties of a single feature. The relevant definitions are defined as follows:

$$F(i) = \frac{\sum_{k=1}^{c} n_k \left(\overline{x_i^k} - \overline{x_i}\right)^2}{\sum_{k=1}^{c} n_k \left(\sigma_i^k\right)^2} \tag{1}$$

where i is the number of features, k is the class, c is the total number of classes, n_k is the number of samples in class k, $\overline{x_i^k}$ is the mean value of the i-th feature in class k, $\overline{x_i}$ is the mean value of the i-th feature in all samples, σ_i^k is the variance of the i-th feature in class k. The bigger $F(i)$ is, the better the discernibility of the feature is.

Traditionally, there are at most hundreds of characteristics of network traffic data. The calculation is too large when the iterative discrimination is directly used for whole features. With the Fisher score, the features with low discernibility and poor identification performance are eliminated from the complete feature set, and the initial feature subset is then obtained. Thus, the amount of computation of subsequent wrapper methods is significantly reduced. Due to its simple and fast calculations, it can be applied to the dimension reduction of large-scale and high dimensional data.

4.2. The Wrapper Feature Selection Methods Base on the SFS Strategy

The wrapper feature selection method consists of two parts: the selection strategy and the evaluation strategy, which are considered as the feature selection process and the follow up machine learning method respectively. In this paper, the sequential forward search (SFS) of heuristic searching is used as the selection strategy, and the classification accuracy is used as the evaluation index.

The selection process is described as follows:

The initial feature subset can be defined as $X = \{f_1, f_2, \cdots, f_m\}$, the optimal subset of features Y is null, and the classification model is M.

1. The classification accuracy of each feature in X is obtained separately by the classification model M.
2. The feature achieving the highest accuracy is added into Y and eliminated from X.
3. Each feature in X is combined with all the features in Y separately, and then obtains their own classification accuracy by the classification model M.
4. The corresponding feature of the highest accuracy in X is added into Y and eliminated from X.
5. Steps 3 and 4 are repeated until the stopping criterion is reached.

Note the threshold α for the stopping criterion adopted in this paper. If the obtained accuracy begins to decrease compared with the previous round, and the decrease amplitude is larger than α, then the feature selection process stops. In this case, the selected features are the optimal subset of features available for later classification operations.

4.3. The Realization of FSMS

FSMS is a feature selection method based on the Spark, a parallel programming framework. This method preprocesses the complete feature set by the Fisher score to obtain the initial feature subset. After that, the sequential forward search, which is used as the searching strategy and the classification accuracy, is used as the evaluation criterion to continuously select the feature combination with the strongest classification ability, which is implemented by Spark. The process of FSMS is shown in Figure 1.

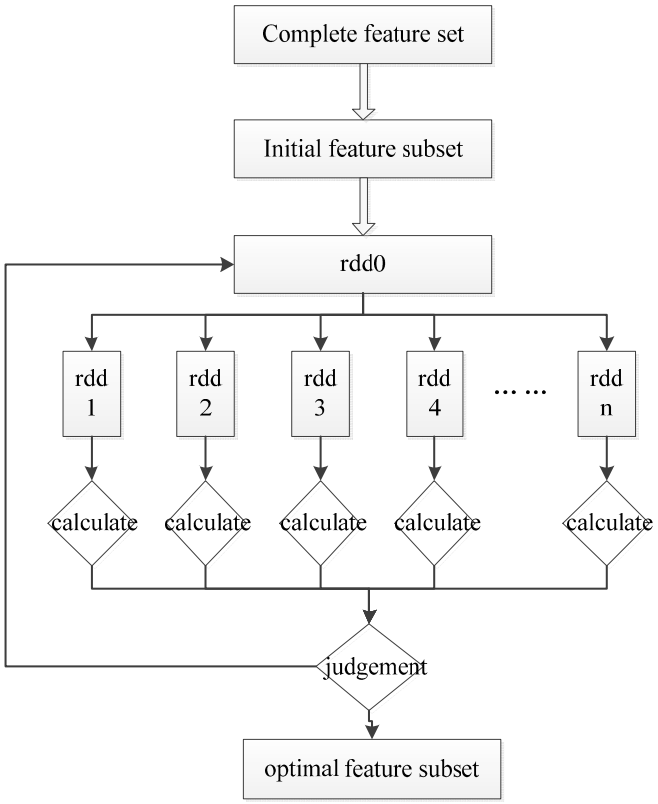

Figure 1. FSMS flowchart.

The Algorithm of FSMS:

input: complete feature set $X = \{f_1, f_2, \cdots, f_l\}$;
output: optimal feature set Z;

//————————————————*Phase* 1: Filter Process————————————————

1: Sort all features $f_i \in X$ by $F(f_i)$ in formula (1) from maximum to minimum;
2: Find the first m features in X to be added to initial data set Y;

//————————————————*Phase* 2: Wrapper Process————————————————

1: Calculate the classification accuracy by per feature and find out the optimal feature f_i, $Y = Y - f_i$,$Z = Z + f_i$;
2: To find the next optimal feature to be added to Z, calculate for all candidate features $Z \cup (f_j \in Y)$, $Y = Y - f_j$,$Z = Z + f_j$;
3: Repeat step 2 until a termination criterion is met.

A threshold $\alpha = 0.002$ is set. When the obtained accuracy begins to decrease compared with the previous round of iteration, and the decrease amplitude is larger than α, the feature selection process stops. In this case, the remaining features are the optimal feature subset.

5. The Experiment Design and the Result Analysis

As an important preprocessing step in traffic classification, the feature selection method aims to select from the complete feature set, and thus to reduce the modeling time and the classification time under the prerequisite condition of having little effect on the accuracy of network traffic classification.

In order to prove the performance of FSMS, the designs of the experiment included two aspects: the effectiveness of feature selection and the efficiency of feature selection.

5.1. The Setup of Experimental Environment and Dataset

The experimental platform cluster consisted of a control node and four computational nodes. All of the nodes had the same configuration, including dual-core CPU (frequency: 3.2 GHz), 4 G Memory and 500 G Hard Disk Drive. The control node and the computational nodes were interconnected by a Gigabit Ethernet switch. Hadoop-2.5.0 and Spark-1.2.1 was installed in the cluster. The data set used in the experiment was the Moore data set [21], which consists of eight kinds of traffic classes, and each flow included 249 feature attributes. The Moore data set is a classic network traffic trace, which has been used widely in the discussions and experiments. It has a structure similar to the newest trace and the most extensive representation.

Table 1. The detail of Moore data set.

Class	Application Type	Count
WWW	http, https	328,091
MAIL	imap, pop2/3, smtp	28,567
BULK	ftp-Control, ftp-Pasv, ftp-Data	11,539
SERVICES	X11, dns, ident, ldap, ntp	2099
DATABASE	postgres, sqlnet oracle, ingres	2648
P2P	KaZaA, BitTorrent, GnuTella	2094
ATTACK	worm, virus	1793
MUITIMEDIA	Windows Media Player, Real	1152

In order to deal with different experimental tests, the data sets with different scales are set up in the experiments, as shown in the tale below.

Table 2. The data set scales used in the experiments.

Dataset	Data1	Data2	Data3	Data4	Data5	Data6
Instances count	24,000	192,000	288,000	384,000	480,000	576,000

Among them, Data1 is the raw data set as shown in Table 1. Table 2 shows the other scaling-up edition.

5.2. The Feature Selection Effectiveness

5.2.1. The Classification Accuracy

In this experiment, firstly, all 249 features from the complete feature set were directly used for the traffic classification. Secondly, the optimal feature subset obtained by FSMS was used for the traffic classification. Throughout the comparison between the two sets, the effect of FSMS on the classification accuracy was verified. Note that Data1 and RandomTree Algorithm were used in the experiment.

As shown in Figure 2, if the optimal feature subset including 16 features was selected by FSMS, then the optimal accuracy was basically achieved. With the selected features increasing, the accuracy remained mostly unchanged and inclined to the one obtained by the complete feature set. It can be seen that FSMS can also guarantee its classification accuracy when fewer features are selected. The validity of this method is thus proved.

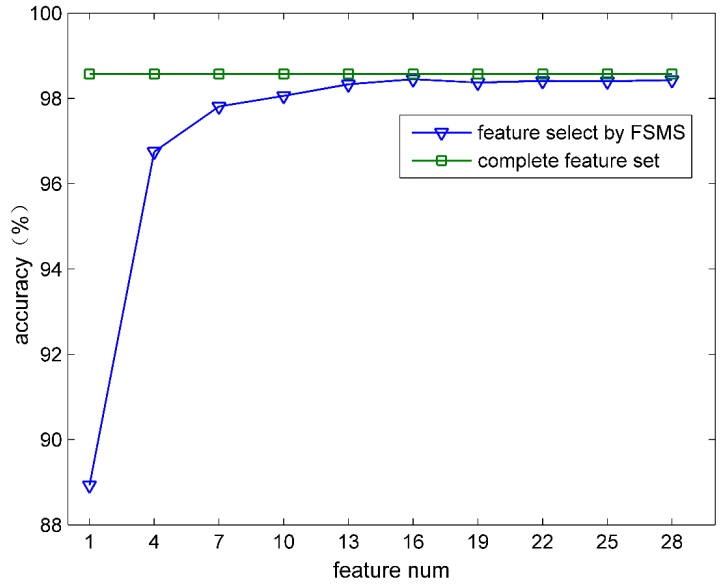

Figure 2. Accuracy comparison.

5.2.2. The Amount of Classification Computation

The comparison of the time consumed by the respective network traffic classifications of complete feature set and optimal feature set was made. The compared items including the time parameters were acquired by the Spark RandomTree algorithm in the modeling and classification process. This experiment used Data 6.

The complete feature set was the raw data that has not yet processed by FSMS, including 249 features in total. The optimal feature subset was a sixteen-dimensional feature set filtered by FSMS. The parameters were expressed this way, total: time for modeling; findSplitsBins: to partition the raw data into buckets according to the feature dimension; findBestSplits: to determine the best split point quickly for each node during the training period in each layer of the decision tree; chooseSplits: time for classification.

As shown in Figure 3, when the feature set processed by FSMS was classified, the consumed time for each indicator was much less than that for adopting the complete feature set directly. The experiment proves that, as an important preprocessing step in traffic classification, FSMS can significantly reduce the amount of computation of network traffic classification.

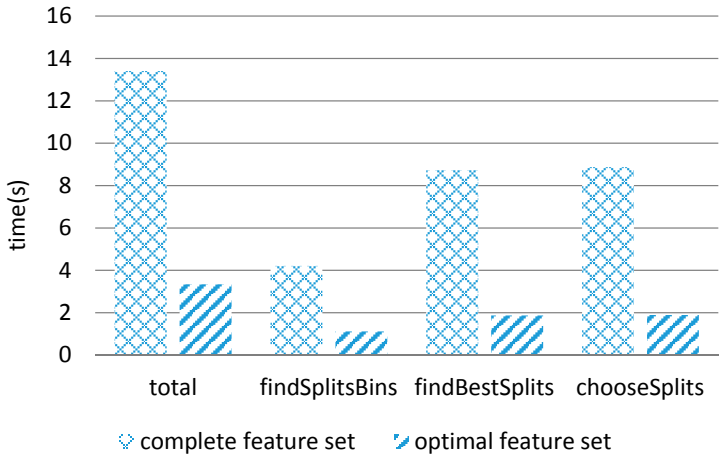

Figure 3. The comparison of the time for modeling and classification.

5.3. The Efficiency of Feature Selection

5.3.1. The Velocity of Feature Selection

In order to demonstrate the performance of FSMS in the aspect of feature selection velocity, the velocity of FSMS was compared with that of FSMM (feature selection method based on MapReduce). FSMM was implemented by the MapReduce computing framework. Both of them have the same computing procedure and adopt the same classification algorithm. Figures 4 and 5 show the comparison of time that both FSMS and FSMM take to select the features from 1 to 16 when DATA1 and DATA6 are used.

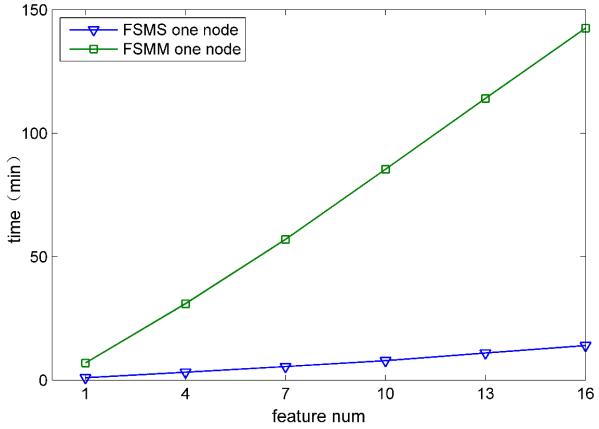

Figure 4. The single node processes Datal.

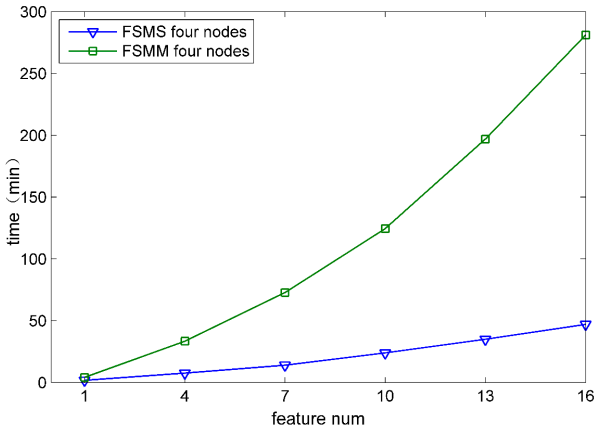

Figure 5. Four nodes process Data6.

According to Figures 4 and 5 under the same computing node, the feature selection time for FSMS is obviously less than that for FSMM. Moreover, with the selected feature increasing, FSMS has more remarkable advantages. The main reason is that the Spark computing framework used by FSMS can cache data into memory. At each iteration, data is directly transferred from memory, which avoids frequent disk I/O operations and greatly improves the iteration efficiency.

5.3.2. The Parallel Computing Effectiveness

Figure 6 shows the time consumed by the selection of 10 features by FSMS in the case of different computing nodes when Data1 to Data6 are used respectively. When Data1 is used, the consumed time is shown as: single node < dual nodes < four nodes. The reason is that, when the amount of data is small, the computing advantages offered by the increasingly number of nodes are not enough to compensate the time consumed by the multiple node task schedules. However, with the constant

increase of data scale, the computing advantages of multiple nodes are shown. The more computing nodes there are, the gentler the increased amplitude of the consumed time is. The running time for single node increases evidently when Data6 is used. This is due to the fact that the single node has limited memory. When the amount of data is too large, the intermediate results of all data and operations cannot be cached in memory by Spark. The data, not being cacheable, will be transferred into local storage. Thus, the time for data storage and data call increases substantially.

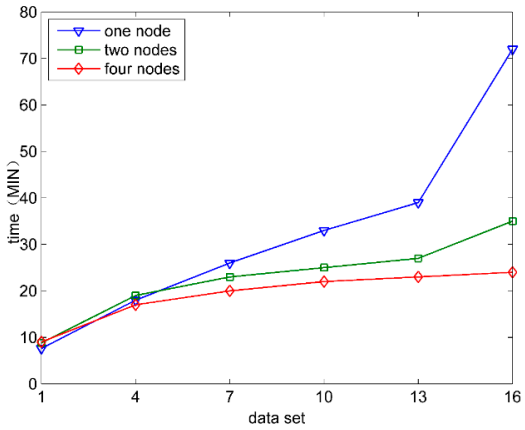

Figure 6. The parallel computing effectiveness.

In order to measure the effectiveness of FSMS parallel computation more accurately, the speedup can be an indicator introduced into this paper. The speedup means that the running time is reduced by the parallel computation to improve the performance obtained. The computing formula is: $S = T_s/T_p$, where T_s is the time consumed by the single node computation, T_p is the time consumed by computing, and the number of nodes is p. The higher the speedup is, the less the time consumed by the parallel computation is, and the more highly the parallel efficiency and the performance improve. The given data sets are from Data1 to Data6, and the number of computing nodes includes one node, two nodes and four nodes. Figure 7 describes the performance of the speedup of FSMS based on Spark under different data scales. When the amount of data is small, the speedup of multi-nodes is less than 1, and the cluster computing does not show any advantage. However, as the amount of data increases, the speedup of multi-nodes increases steadily, and the higher the number of nodes is, the faster the amplitude increases. The result shows that, when processing the data with different quantity degree, FSMS can enhance the processing efficiency by increasing or decreasing the size of clusters, which has the capability to be extended.

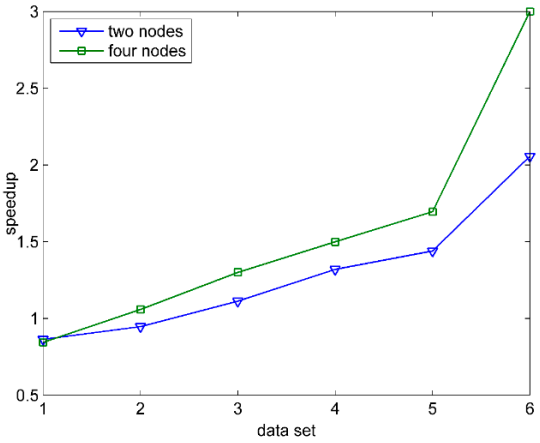

Figure 7. Speedup.

6. Conclusions

In this paper, we proposed a feature selection method for network traffic based on Spark (FSMS). In our proposal, the data is preprocessed firstly by the Fisher score, and the feature set with the best classification ability is then selected by the Spark computing framework along with continuous iterations. The implementation shows that this method serving as an important preprocessing step in traffic classification can greatly reduce the modeling and classification time for the classifier under the prerequisite condition of having no effect on the accuracy of traffic classification. In addition, due to the combination with the advantages of the Spark computing framework in the complicated iterative calculation, compared to the traditional feature selection method base on MapReduce, FSMS achieves a great performance improvement in the velocity of feature selection. Based on the Spark platform, future research will ideally discover a parallel network traffic classification method that can be combined with the feature selection method mentioned in this paper to implement highly effective and fast classification of network traffic.

Acknowledgements: This work was supported by the National Natural Science Foundation of China (61163058 and 61363006), Key Laboratory of Cognitive Radio and Information Processing, Guangxi Key Laboratory of Trusted Software (No. KX201306), and Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems (No. 14104).

Author Contributions: The work presented here was a collaboration between all of the authors. All authors contributed to designing the methods and experiments. Wenlong Ke performed the experiments. Yong Wang and Xiaoling Tao analyzed the data and interpreted the results. Yong Wang and Xiaoling Tao wrote the paper. All authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Soysal, M.; Schmidt, E.G. Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison. *Perform. Eval.* **2010**, *67*, 451–467. [CrossRef]
2. Fahad, A.; Tari, Z.; Khalil, I.; Habib, I.; Alnuweiri, H. Toward an efficient and scalable feature selection approach for internet traffic classification. *Comput. Netw.* **2013**, *57*, 2040–2057. [CrossRef]
3. Crone, S.F.; Kourentzes, N. Feature selection for time series prediction—A combined filter and wrapper approach for neural networks. *Neurocomputing* **2010**, *73*, 1923–1936. [CrossRef]
4. De Donato, W.; Pescapé, A.; Dainotti, A. Traffic Identification Engine: An Open Platform for Traffic Classification. *IEEE Netw.* **2014**, *28*, 56–64.
5. Xu, M.; Zhu, W.; Xu, J.; Zheng, N. Towards Selecting Optimal Features for Flow Statistical Based Network Traffic Classification. In Proceedings of the 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), Busan, Korea, 19–21 August 2015; pp. 479–482.
6. Dainotti, A.; Pescapé, A.; Sansone, C. Early Classification of Network Traffic through Multi-classification. In *Traffic Monitoring and Analysis*; Domingo-Pascual, J., Shavitt, Y., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 122–135.
7. Jing, R.; Zhang, C. A Novel Feature Selection Method for the Conditional Information Entropy Model. In Proceedings of the 3rd International Conference on Artificial Intelligence and Computational Intelligence(AICI 2011), Taiyuan, China, 24–25 September 2011; pp. 598–605.
8. Wang, Y. Fisher scoring: An interpolation family and its Monte Carlo implementations. *Computational Stat. Data Anal.* **2010**, *54*, 1744–1755. [CrossRef]
9. Wang, P.; Sanin, C.; Szczerbicki, E. Prediction based on Integration of Decisional DNA and a Feature Selection Algorithm relief-F. *Cybern. Syst.* **2013**, *44*, 173–183. [CrossRef]
10. Rodrigues, D.; Pereira, L.A.M.; Nakamura, R.Y.M.; Costa, K.A.P.; Yang, X.S.; Souza, A.N.; Papa, J.P. A wrapper approach for feature selection based on bat algorithm and optimum-path forest. *Expert Syst. Appl.* **2014**, *41*, 2250–2258. [CrossRef]
11. Chuang, L.-Y.; Yang, C.-H.; Li, J.C.; Yang, C.-H. A hybrid BPSO-CGA approach for gene selection and classification of microarray data. *J. Comput. Biol.* **2012**, *19*, 68–82. [CrossRef] [PubMed]

12. Peng, Y.H.; Wu, Z.Q.; Jiang, J.M. A novel feature selection approach for biomedical data classification. *J. Biomed. Inform.* **2010**, *43*, 15–23. [CrossRef] [PubMed]
13. Dainotti, A.; Pescapé, A.; Claffy, K.C. Issues and Future Directions in Traffic Classification. *IEEE J. Mag.* **2012**, *25*, 35–40. [CrossRef]
14. Szabó, G.; Veres, A.; Malomsoky, S.; Gódor, I.; Molnár, S. Traffic Classification over Gbit Speed with Commodity Hardware. *IEEE J. Commun. Softw. Syst.* **2010**, *5*. Available online: http://hsnlab.tmit.bme.hu/molnar/files/jcss2010.pdf (accessed on 15 February 2016).
15. Sun, Z.Q.; Li, Z. Data intensive parallel feature selection method study. In Proceedings of 2014 International Joint Conference on Neural Networks (IJCNN), Beijing, China, 6–11 July 2014; pp. 2256–2262.
16. Li, J.Z.; Meng, X.R.; Wen, J. An improved method of SVM-BPSO feature selection based on cloud model. *IAES Telkomnika Indones. J. Electr. Eng.* **2014**, *12*, 3979–3986. [CrossRef]
17. Long, Y. Network Traffic Classification Method Research Based on Cloud Computing and Ensemble Learning. Ph.D. Thesis, Guilin University of Electronic Technology, Guilin, China, June 2015.
18. Srirama, S.N.; Batrashev, O.; Jakovits, P.; Vainikko, E. Scalability of parallel scientific applications on the cloud. *Sci. Program.* **2011**, *19*, 91–105. [CrossRef]
19. Srirama, S.N.; Jakovits, P.; Vainikko, E. Adapting scientific computing problems to clouds using MapReduce. *Future Gener. Comput. Syst.* **2012**, *28*, 184–192. [CrossRef]
20. Apache Spark. Available online: http://spark.apache.org (accessed on 14 February 2016).
21. Moore, A.; Zuev, D. *Discriminators for Use in Flow-based Classification*; Queen Mary and Westfield College: London, UK, 2005; pp. 1–13.