J Inf Process Syst, Vol.14, No.1, pp.140~149, February 2018
https://doi.org/10.3745/JIPS.04.0057

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Development of LDA Topic Association Systems Based on Spark-Hadoop Framework

Kiejin Park* and Limei Peng**

Abstract

Social data such as users' comments are unstructured in nature and up-to-date technologies for analyzing such data are constrained by the available storage space and processing time when fast storing and processing is required. On the other hand, it is even difficult in using a huge amount of dynamically generated social data to analyze the user features in a high speed. To solve this problem, we design and implement a topic association analysis system based on the latent Dirichlet allocation (LDA) model. The LDA does not require the training process and thus can analyze the social users' hourly interests on different topics in an easy way. The proposed system is constructed based on the Spark framework that is located on top of Hadoop cluster. It is advantageous of high-speed processing owing to that minimized access to hard disk is required and all the intermediately generated data are processed in the main memory. In the performance evaluation, it requires about 5 hours to analyze the topics for about 1 TB test social data (SNS comments). Moreover, through analyzing the association among topics, we can track the hourly change of social users' interests on different topics.

Keywords

Association Analysis, Hadoop, LDA (Latent Dirichlet Allocation), Spark, Topic Model

1. Introduction

The recently emerging informal text data that are handled online shows two important features, i.e., huge amount and unstructured. For such huge amount of unstructured data, methods that are used for processing the existing relational data model show their limitations in aspects of storing and processing speed. Especially, it becomes even more difficult to analyze and extract the semantics of the online text data that are continuously generated by a SNS (Social Network Service). To solve this problem with consideration on machine learning, the topic modeling named latent Dirichlet allocation (LDA), which can extract the topics without training, is receiving tremendous attention [1]. By using LDA, we can extract the topic features by analyzing the inherent patterns of all documents, i.e., topic ratios, rather than analyzing the types or the frequencies of special words as in the traditional methods.

In this paper, in order for understanding the topics of social users' hourly interest in an easy and fast way, we propose and implement a topic-association analysis system based on LDA modeling. The feature vectors are summarized and obtained in the step of pre-processing on input datasets of social

Manuscript received May 8, 2017; first revision June 20, 2017; accepted July 30, 2017.
Corresponding Author: Limei Peng (auroraplm@ajou.ac.kr)
* Dept. of Integrative Systems Engineering, Ajou University, Suwon, Korea (kiejin@ajou.ac.kr)
** Dept. of Industrial Engineering, Ajou University, Suwon, Korea (auroraplm@ajou.ac.kr)

big data. Then, we can analyze all the documents after LDA processing and the association among topics hourly. Moreover, to analyze social big data with high speed, we adopt the distributed in-memory based Spark framework. Spark is established on top of Hadoop cluster which can store the data and process distributed-parallel commands under the cluster computing environment [2]. When computing data in Spark, the requirement of access to hard disk is minimized and all the intermediate results are processed in the main memory [3,4]. Note that Spark DataFrame (Spark SQL) which uses column as a basic processing unit is adopted to process queries in Spark [5,6]. The rest of this paper is constructed as follows. Section 2 introduces the topic modeling and association based on probabilities. In Section 3, we explain the approach of data reduction by means of extracting data features and the design of topic association analyses system. Section 4 shows the system implementation and the experiment results to verify the analyses system. Conclusions and future works are introduced in Section 5.

2. Related Works

2.1 LDA Topic Modeling

LDA topic modeling is based on the probability theory [7]. It can find the topics hidden in the documents that are composed of texts using the technology of statistical estimation. We assume k topics exist inside all the document sets and the topic association can be presented in each document. Equation (1) is a generic probabilistic model used to reinterpret all the document sets and find the particular patterns underlying in social big data.

$$p(\beta, \theta, \mathbf{z}, \mathbf{w}) = \left(\prod_{k=1}^{K} p(\beta_k|\eta)\right)\left(\prod_{d=1}^{D} p(\theta_d|\alpha)\prod_{n=1}^{N} p(z_{d,n}|\theta_d)p(w_{d,n}|\beta_{1:K}, z_{d,n})\right) \quad (1)$$

In Eq. (1), the topic k follows a Dirichlet distribution of $\beta_k \sim Dir(\eta)$ within a corpus. The corpus is a set of all words and uses the method called "the bag of words" for the whole documents. The value d of each document follows the $\theta_d \sim Dir(\alpha)$ distribution, and every word of the corresponding document can be presented by words following the multinomial distribution with $z_{d,n} \sim \text{Mult}(\theta_d)$ and $w_{d,n} \sim \text{Mult}\left(\beta_{z_{d,n}}\right)$. According to the Dirichlet distribution, the posterior and prior parts follow the same distribution and thus the inference process can be executed faster and easier.

The pseudo code for deducing the LDA parameters is shown in Fig. 1. The expected values of the potential variables are calculated using the current parameters in E-Step. In M-Step, the expected values of the potential variables that are found in E-Step are used as the reference values to find the maximized log-likelihood parameter. Since the Spark framework-based LDA is applicable to distributed-parallel command processing environment [8], we design and construct a distributed system under the cluster environment which is based on the Hadoop YARN (Yet Another Resource Negotiator) [9]. On the other hand, an online learning machine for LDA topic modeling was developed in [10] on top of the Hadoop-based MapReduce framework. However, its performance is constrained when compared to the in-memory based Spark, because MapReduce-based framework uses the disk-based calculation.

Algorithm: LDA Parameter Inference Pseudo Code

```
for d = 1 to D do
  E-Step:
  repeat
    update document/topic distribution for d
    update topic/word assignments for d
  until convergence
  M-Step:
    update topic/word distribution
end for
```

Fig. 1. The inference process of LDA topic modeling parameter.

2.2 Cosine Similarity

Cosine similarity means to measure the similarity between two vectors by using the cosine values of the two vector angles of inner product. In Eq. (2), when the directions of two vectors are exactly the same, the resultant value is 1; when the angle is 90°, the resultant value is 0. Therefore, the value is used to determine the angle similarity rather than the vector size.

$$\text{Cosine } Similarity = \frac{\mathrm{A} \cdot \mathrm{B}}{|A||B|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} \mathrm{A_i}^2} \times \sqrt{\sum_{i=1}^{n} \mathrm{B_i}^2}} \tag{2}$$

In Spark which is capable of processing big data in a distributed way, the feature values of data frames are exhibited as vectors and then the cosine similarity is calculated for two column vectors to show the association score. This is especially applicable to skinny big data featured by that the data size of rows is much bigger than that of columns.

3. Topic Association Analysis System

3.1 Spark-based Distributed-Parallel Processing System

In this topic-association analyses system, the Spark in-memory distributed analyses framework is used in order for finding the topic association among various informal document sets. Spark framework processes data repeatedly which is similar to machine learning and saves the intermediate results in the main memory to achieve continuous and high-speed processing.

Fig. 2 shows the Hadoop cluster for storing big data and the components of Spark framework. In Hadoop Distributed File System (HDFS) [11], the RM (ResourceManager) of YARN is used to manage resources in the master node, and in the rest slave nodes, the modules of NM (NodeManager) and AM (ApplicationMaster), etc., are used to save and process big data. By using YARN, the Hadoop Cluster node management functions including resource assignment, job scheduling, combination, etc., are provided. In Master node, by using the interactive Spark driver program, the distributed-parallel tasks assigned to each slave are executed on Executor. Resilient Distributed Datasets (RDDs) of Spark are the fundamental processing data units that are sets of data objects obtained after being processed on several

such distributed memories [6,12]. In the proposed Hadoop Cluster system, each node is inherently with the capabilities of load balancing and failover.

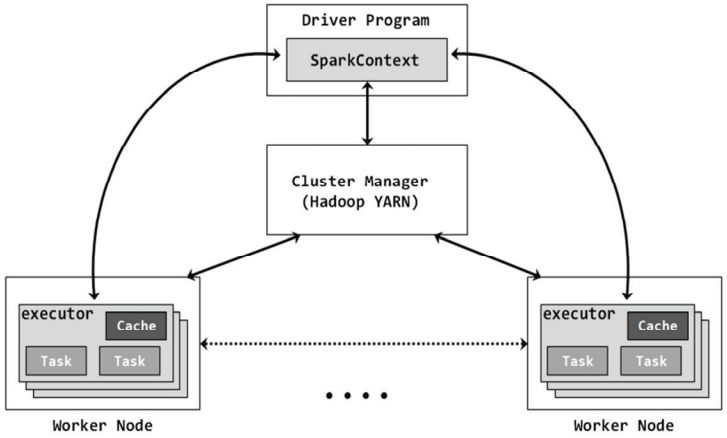

Fig. 2. Association analyses cluster structure based on Spark-Hadoop framework.

3.2 In-memory Processing of Topic Model

Fig. 3 shows the structure for generating new summary documents with reduced size using LDA model. Assume $\{N_1, \ldots, N_D\}$ words exist inside Corpus, then α is the parameter used to show the possibility that a topic would appear inside all the documents; β is the probability of generating the feature word for each topic; θ_d of a particular document d represents the weight of each topic. Take the particular document d for example, firstly we can decide θ_d based on α and then using the probability represented by θ_d we can assign N_d, i.e., $\{Z_{d,1}, \ldots, Z_{d,Nd}\}$, topics. Using β that is determined depending on such assigned topics and η, we can generate N_d, i.e., $\{W_{d,1}, \ldots, W_{d,Nd}\}$, words, where α and η follow the Dirichlet distribution.

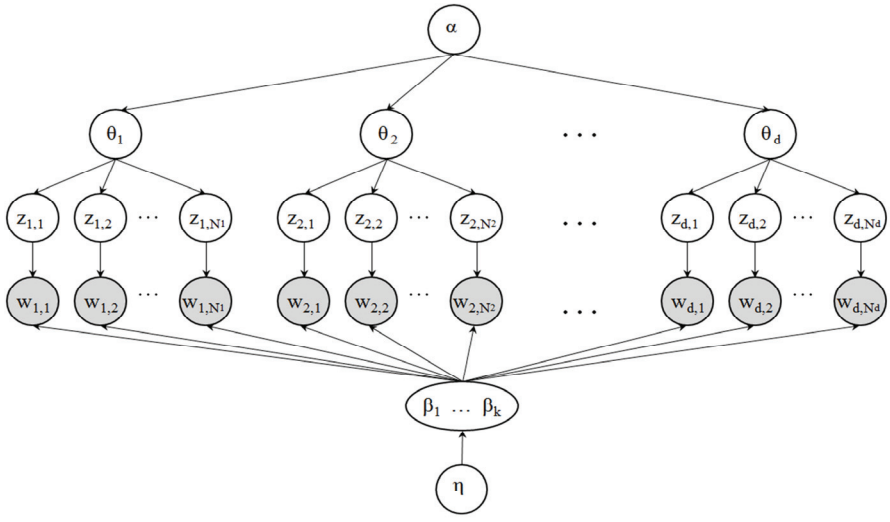

Fig. 3. Structure for generating summarized documents.

To implement the structure shown in Fig. 3, we use the mini-batch method in cluster memory and process them after storing them in terms of RDD. Meanwhile, in order for enabling LDA analyses, we firstly decide the Corpus (i.e., the bag of words) for all the comments. Since the values of parameters are not decided at the beginning of execution, we start from a D-by-W matrix, where D and W represent document and word, respectively. After this, we use the parameter values that are generated through repeated E-steps and M-steps shown in Fig. 1.

3.3 Topic Association Analysis Process

In this system, in order for enabling integrative analyses, we use DataFrame to read all the text document data. DataFrame is provided by Spark SQL and can simultaneously support the procedural processing of functional language and pre-processing of SQL. The features of input comment data are extracted through the process shown in Fig. 4. More specifically, through the preprocessing on data, words per document will be separated. Amongst the separated words, ones that are not used in each comment are removed. Consequently, we use the vector values that are transformed from comments as input for the LDA processing. Meanwhile, since sparsity feature vector value is also used, we can save significant processing time and space for big dataset. After LDA processing, all comment data can be compressively represented by k proportional topics. For every topic column of this data frame, cosine similarity is implemented to analyze the association. Moreover, the hourly change of topics is visually shown by the final graph in Fig. 4.

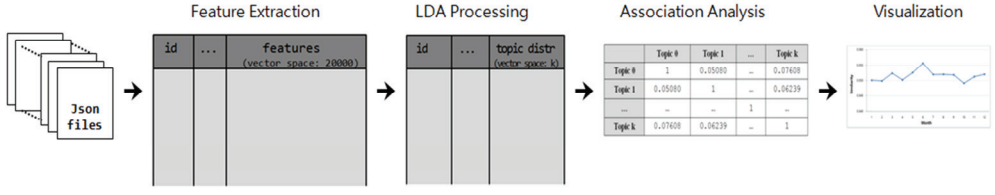

Fig. 4. Topic association analyses process.

Sample Source (in Scala)

```
:
import org.apache.spark.ml.feature.{RegexTokenizer, StopWordsRemover, CountVectorizer, ... }
import org.apache.spark.ml.clustering.LDA
import org.apache.spark.mllib.linalg.Vectors
:
val dataDF = rawdata.select("id", "body", "created_utc")
:
1) val cv = new CountVectorizer().setInputCol("words").setOutputCol("features")
   val df = cv.transform(dataDF)
:
2) val lda= new LDA().setK(14).setMaxIter(10)
   val model = lda.fit(df)
3) val topics = model.describeTopics(5)
   val trans = model.transform(df)
:
4) val rows1 = trans1.select("topicDistribution").rdd.map{case Row(topicDistribution: MLVector) =>
          Vectors.fromML(topicDistribution)}
   val mat1 = new RowMatrix(rows1)
   val sim1 = mat1.columnSimilarities()
:
```

Fig. 5. Sample of source code for topic association analysis.

Fig. 5 shows a sample of source code for analyzing topic associations. It consists of the following steps: 1) use CountVectorizer library to find the feature values by extracting features, 2) for every comment that is represented by word vector, we can know the ratio of word used by each document by means of executing LDA modeling; 3) the ratio that a specific topic is used by a given comment can be shown, and 4) the associations among topics are analyzed hourly.

4. Experimental Environment and Analysis Results for Topic Association

The prototype cluster is composed of one master node and six slave nodes. Hadoop 2.7 and Spark 2.0 are installed on top of Ubuntu 14.04 LTS to establish the processing framework. More descriptions on our experiment setup are as follows:

1) The total available memory of the cluster is 384 GB (64 GB × 6 nodes);

2) The capacity of HDD is 64 TB (8 TB × 6 nodes);

3) Every Slave node uses CPU of either i7 or i5 (Skylake, 3.2 GHz);

4) The memory of Driver Program of Master node is 12 GB and the executor memory of each Slave node is 56 GB; and

5) A total of six executors are used and seven cores per executor are established as the Spark LDA execution parameters.

The total number of input records for social documents used in the experiments is about 16 million. All records consist of 22 attribute items and are saved in format of JSON file which is constructed by Keys and Values. Even though every comment has corresponding sub category, in this experiment, we can see the trend of hourly change of social users' interests based on just comments. Before LDA topic analyses, preprocessing is executed through Tokenizer and StopWordsRemover for all comments.

4.1 Decision on *K* Value of Topic Numbers

To perform LDA clustering, we should define the total number of topics, say *K*, in advance. Since it is difficult to know how many clusters are needed in reality, we use an average value for *K*, which is obtained by converging the values used in several practical methods until they are not reduced anymore, so as to obtain a stable and reliable *K* value. Fig. 6 shows the change of logLikelihood values according to the topic numbers. The logLikelihood values converge when the number of topics is 14. Therefore, we start by determining the number of topics of users' interest as 14.

4.2 LDA Topic Results

To analyze LDA topics, we should extract the features for comment data after preprocessing. In these experiments, we use CountVectorizer and fix the setVocabSize to 20000. For features extracted in this way, we set the value of *K* to 14 and then start the LDA analyses. From the LDA processing results obtained after five hours, we can determine the topic content represented by words with different weight ratios. Table 1 shows the results represented by five words with different weight values for all the 14 topics, where only part of topics are shown due to space limit. From the results, we can observe that

Topic7 and Topic12 indicate interests on game and money, respectively. Moreover, every comment can be represented in a novel way using 14 topic ratios through topic Distribution.

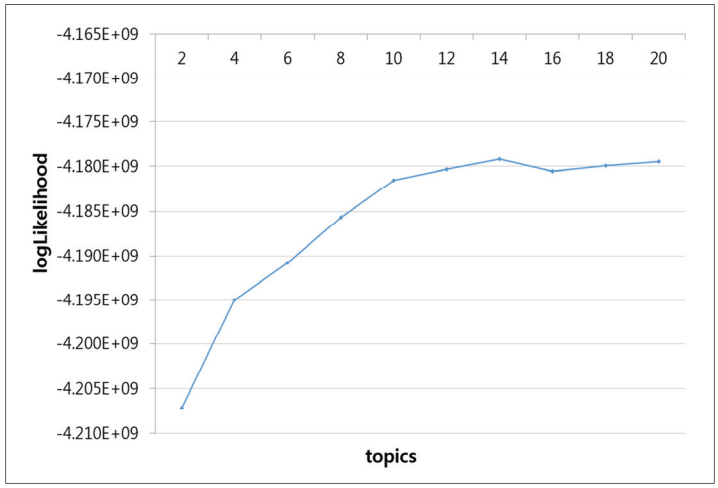

Fig. 6. The number of topics versus the change of logLikeliHood.

Table 1. Topic analysis result for five topics out of a total of 14

Topic1		Topic5		Topic7		Topic12		Topic13	
Words	**Weight**	**Words**	**Weight**	**Words**	**Weight**	**Words**	**Weight**	**Words**	**Weight**
people	0.0228	thanks	0.0154	game	0.0187	people	0.0118	one	0.0132
get	0.0106	really	0.0104	get	0.0094	get	0.0077	can	0.0083
can	0.0102	yes	0.0099	play	0.0088	money	0.0065	people	0.0071
one	0.0101	good	0.0079	one	0.0081	can	0.0062	also	0.0042
time	0.0079	thank	0.0074	can	0.0078	want	0.0061	know	0.0042

4.3 Analysis on Association Change among Topics

In the performance analyses, the parameter used is the LDA topic weight, which is obtained based on the number of words for all topics that are extracted from sentences. For each topic, the column similarity is compared. More specifically, a total of 14 topics are compared in the terms of columns in order for obtaining the statistic similarity.

Table 2 shows the column similarities for all the comments that are represented by 14 topics. Among the 14 topics, the highest and lowest association values are 0.07608 between Topic0 and Topic5 and 0.04393 between Topic1 and Topic7, respectively.

Using the above topics in this analysis system, we can see the hourly difference between social users' interests on game and money for example

Fig. 7 shows the monthly change on topic associations. More specifically, among a total of 14 topics, the monthly topic association between Topic7 regarding game and Topic12 regarding money is shown. By doing this, we can conclude that social users who are interested in game pay the most attention on money in June and the least attention in October.

Table 2. Topic similarity using column similarities

	Topic0	Topic1	…	Topic5	…	Topic7	…	Topic12	Topic13
Topic0	1	0.05080	…	0.07608	…	0.06145	…	0.06289	0.06572
Topic1	0.05080	1	…	0.06239	…	0.04393	…	0.05090	0.06690
…	…	…	1	…	…	…	…	…	…
Topic5	0.07608	0.06239	…	1	…	0.06551	…	0.05765	0.06817
…	…	…	…	…	1	…	…	…	…
Topic7	0.06145	0.04393	…	0.06551	…	1	…	0.05151	0.05031
…	…	…	…	…	…	…	1	…	…
Topic12	0.06289	0.05090	…	0.05765	…	0.05151	…	1	0.06684
Topic13	0.06572	0.06690	…	0.06817	…	0.05031	…	0.06684	1

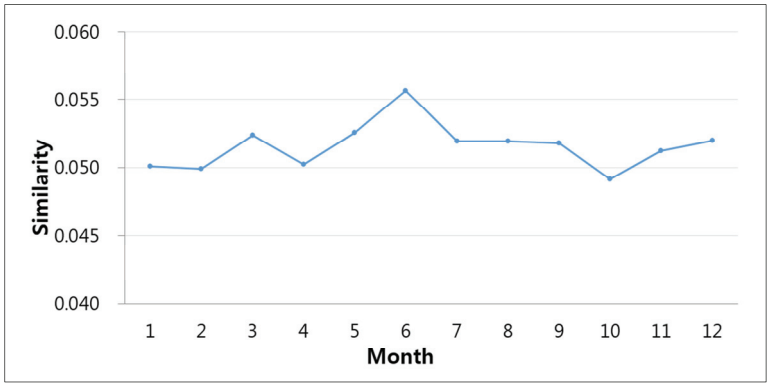

Fig. 7. Change of association between Topic7 and Topic12.

5. Conclusion and Future Research

In this paper, in order to understand users' features from a huge amount of informal text data such as social users' comments, we proposed the topic association analyses system based on LDA model which requires no training process. In the proposed system, we extract the feature vector by preprocessing the input datasets of social big data, then after LDA processing, we executed topic association analyses for all documents and thus can understand the change of social users' hourly interests on different topics. The proposed system was constructed based on Spark framework which is installed on top of Hadoop cluster, and thus the requirement of access to disk was minimized and the intermediately generated results were processed in the main memory which guaranteed high-speed processing.

From this paper, we can observe that the LDA topic analysis system based on single machine, is restricted by the data processing capability. In contrast, the system architecture proposed in our paper, which is based on the in-memory distributed-parallel method, showed relatively faster processing speed when processing big data as much as terabits.

In the experiment part, the topic association for about 1 TB informal text data (SNS comments) was executed for about five hours. Moreover, when compared to association analysis on such extracted

topics, we could quickly confirm the hourly change of topics that social users were interested. We can envision that in order to assist the processing on informal data, design on computation framework and various topic analysis algorithms are needed in the future.

Acknowledgement

This paper is partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (No. 2015R1C1A1A02036536) and partially supported by Ajou University Research Fund.

References

[1] D. M. Blei, "Probabilistic topic models," *Communication of the ACM*, vol. 55, no. 4, pp. 77-87, 2012.

[2] K. Park, C. Baek, and L. Peng, "A development of streaming big data analysis system using in-memory cluster computing framework: Spark," *Lecture Notes in Electrical Engineering*, vol. 393, pp. 157-163, 2016.

[3] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud'10)*, Boston, MA, 2010, pp. 1-7.

[4] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation (OSDI'04)*, San Francisco, CA, 2004, pp. 137-149.

[5] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, et al., "Spark SQL: relational data processing in Spark," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD'15)*, Melbourne, Australia, 2015, pp. 1383-1394.

[6] S. Kang, K. Park, and L. Peng, "Improving diversity using bandwagon effect for developing recommendation system," *Far East Journal of Electronics and Communications*, vol. 17, no. 3, pp. 539-544, 2017.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993-1022, 2003.

[8] M. D. Hoffman, D. M. Blei, and F. Bach, "Online learning for latent Dirichlet allocation," in *Proceedings of the 23rd International Conference on Neural Information Processing Systems (NIPS'10)*, Vancouver, Canada, 2010, pp. 856-864.

[9] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, et al., "Apache Hadoop YARN: yet another resource negotiator," in *Proceedings of the 4th Annual Symposium on Cloud Computing (SOCC'13)*, Santa Clara, CA, 2014, pp 1-16.

[10] J. Park and H. Oh, "Distributed online machine learning for topic models," *Communications of the Korean Institute of Information Scientists and Engineers*, vol. 32, no. 7, pp. 40-45, 2014.

[11] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proceedings of 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, Incline Village, NV, 2010, pp. 1-10.

[12] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI'12)*, Berkeley, CA, 2012, pp. 1-14.

Kiejin Park https://orcid.org/0000-0002-6872-2806

He received the B.S. degree from Hanyang University in 1989, the M.S. degree in industrial engineering from POSTECH, Korea, in 1991, and the Ph.D. degree from the Department of Computer Engineering, Graduate School of Ajou University, Korea, in 2001. He has been a Professor with the Division of Industrial and Information Systems Engineering and Department of Integrative Systems Engineering, Ajou University, since 2004. From 1991 to 1997, he was with the Software Research and Development Center, Samsung Electronics Co., Korea, as a Senior Researcher. From 2001 to 2002, he was with the Network Equipment Test Center, ETRI, as a Senior Researcher. From 2002 to 2004, he was with the Department of Computer Engineering, Anyang University, Korea, as a Professor. From 2010 to 2011, he was with Rutgers, the State of New Jersey as a Visiting Professor. His research interests include cloud computing, big data processing platform, recommendation system, and fault-tolerant computing.

Limei Peng https://orcid.org/0000-0001-9984-9861

She received her M.S. and Ph.D. degrees in 2006 and 2010, respectively, from the Chonbuk National University, Jeonju, Korea. She is now an assistant professor with the department of Industrial Engineering, Ajou University, Korea. She worked as a research professor in the Grid Middleware Research Center, Korea Advanced Institute of Science and Technology (KAIST), South Korea, from February 2010 to February 2011. After that, she worked as an associated professor in Soochow University, China, from April 2011 to July 2013. She served as a TPC chair of CloudComp'2015 and a workshop co-chair of CIT'2012. Her research interests include optical communication networks and protocols, datacenter networks, optical fiber sensor networks, Cloud computing networks, etc.