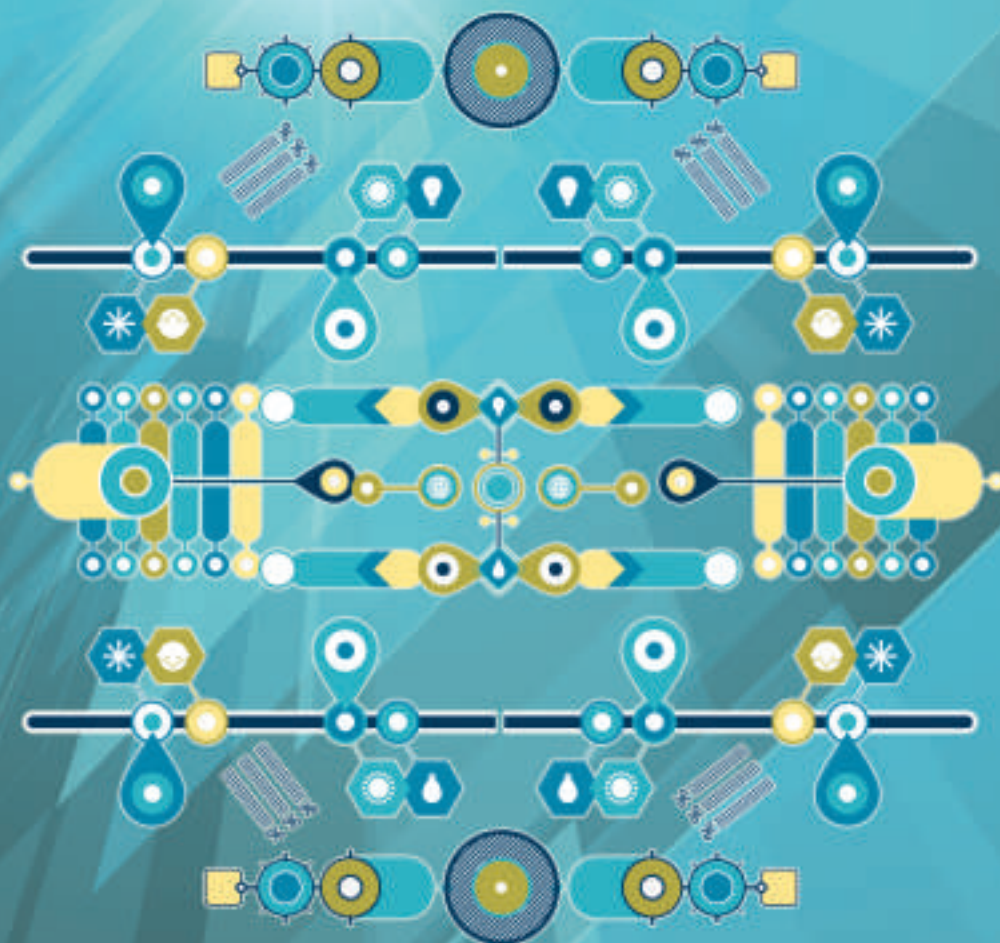


ZTE COMMUNICATIONS

June 2013, Vol.11 No.2

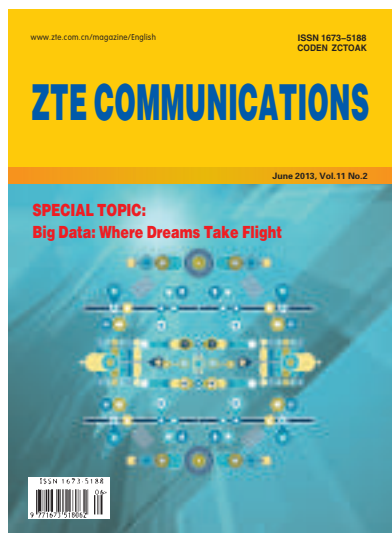
SPECIAL TOPIC: **Big Data: Where Dreams Take Flight**



ISSN 1673-5188



► CONTENTS



.....

Authors are responsible for obtaining permission to reproduce any material for which they do not hold copyright. Permission to reproduce any part of this publication for commercial use must be obtained in advance from the editorial office of *ZTE Communications*. Authors agree that a) the manuscript is a product of research conducted by themselves and the stated co-authors, b) the manuscript has not been published elsewhere in its submitted form, c) the manuscript is not currently being considered for publication elsewhere. If the paper is an adaptation of a speech or presentation, acknowledgement of this is required within the paper.

Responsibility for content rests on authors of signed articles and not on the editorial board of *ZTE Communications* or its sponsors.

All rights reserved.

.....

Special Topic

Big Data: Where Dreams Take Flight

- | | |
|----|---|
| 01 | Guest Editorial
Chengzhong Xu and Zhibin Yu |
| 03 | Content Centric Networking: A New Approach to Big Data Distribution
Yi Zhu and Zhengkun Mi |
| 11 | Big-Data Analytics: Challenges, Key Technologies and Prospects
Shengmei Luo, Zhikun Wang, and Zhiping Wang |
| 18 | Data Security and Privacy in Cloud Storage
Xinhua Dong, Ruixuan Li, Wanwan Zhou, Dongjie Liao, and Shuoyi Zhao |
| 24 | An Efficient Dynamic Proof of Retrievability Scheme
Zhen Mo, Yian Zhou, and Shigang Chen |
| 30 | SPBD: Streamlining Big-Data Processing in Cloud Environments
Tung Nguyen, Jingwen Zhang, and Weisong Shi |
| 38 | A Hadoop Performance Prediction Model Based On Random Forest
Zhendong Bei, Zhibin Yu, Huiling Zhang, Chengzhong Xu, Shenzhong Feng, Zhenjiang Dong, and Hengsheng Zhang |

► CONTENTS

ZTE COMMUNICATIONS

Vol. 11 No.2 (Issue 38)

Quarterly

First English Issue Published in 2003

Supervised by:

Anhui Science and Technology Department

Sponsored by:

ZTE Corporation and Anhui Science
and Technology Information
Research Institute

Staff Members:

Editor-in-Chief: Sun Zheng

Associate Editor-in-Chief: Zhao Jinming

Executive Associate

Editor-in-Chief: Huang Xinming

Editor-in-Charge: Zhu Li

Editors: Paul Sleswick, Xu Ye, Yang Qinyi,
Lu Dan

Producer: Yu Gang

Circulation Executive: Wang Pingping

Assistant: Wang Kun

Editorial Correspondence:

Add: 12/F Kaixuan Building,

329 Jinzhai Road,

HeFei 230061, P. R. China

Tel: +86-551-65533356

Fax: +86-551-65850139

Email: magazine@zte.com.cn

Published and Circulated

(Home and Abroad) by:

Editorial Office of

ZTE Communications

Printed by:

Hefei Zhongjian Color Printing Company

Publication Date:

June 25, 2013

Publication Licenses:

ISSN 1673-5188

CN 34-1294/TN

Advertising License:

皖合工商广字0058号

Annual Subscription Rate:

RMB 80

Research Papers

45

Parallel Spectral Clustering Based on MapReduce

Qiwei Zhong, Yunlong Lin, Junyang Zou, Kuangyan Zhu, Qiao Wang,
and Lei Hu

51

Spam Filtering: Online Naive Bayes Based on TONE

Guanglu Sun, Hongyue Sun, Yingcai Ma, and Yuewu Shen

55

A System for Detecting Refueling Behavior along Freight Trajectories and Recommending Refueling Alternatives

Ye Li, Fan Zhang, Bo Gan, and Chengzhong Xu

Roundup

02

New Members of *ZTE Communications* Editorial Board

23

ZTE Makes Industry Breakthrough with 1 Gbps LTE-Advanced

37

ZTE Cloud Radio Solution to Usher in New Era of High-Performance LTE Networks

54

ZTE Communications Call for Papers -- Special Issue on "Cloud Computing"

Big Data: Where Dreams Take Flight

► Chengzhong Xu



Chengzhong Xu received his BSc degree and MSc degree in computer science and engineering from Nanjing University in 1986 and 1989. He received his PhD degree in computer engineering from the University of Hong Kong in 1993. His research interests include computer architecture, distributed systems, virtualization, and cloud computing.

Dr. Xu is a professor of electrical and computer engineering at Wayne State University, Detroit, USA. He is also the director of the Cloud and Internet Computer Laboratory at Wayne State University. He is a senior member of the IEEE and member of the ACM.

► Zhibin Yu



Zhibin Yu received his PhD degree in computer science from Huazhong University of Science and Technology (HUST) in 2008. He spent one year as a visiting scholar at the Laboratory of Computer Architecture, Department of Electrical and Computer Engineering, University of Texas at Austin. He is currently an associate professor at the Shenzhen Institutes of Advanced

Technology, China. His research interests include micro-architecture simulation, computer architecture, workload characterization and generation, performance evaluation, multicore architecture, and virtualization technologies. In 2005, he won first prize in the HUST Young Lecturers Teaching Contest. In 2003, he won second prize in the HUST Teaching Quality Assessment. He is a member of the IEEE and ACM.

From academia to industry, big data has become a buzzword in information technology. The US Federal Government is paying much attention to the big-data revolution. In 2012, fourteen US government departments allocated funds to 87 big-data projects [1]. Europe has the second largest amount of data [2], and most universities and research institutes have already established big-data research programs. In Asia, especially in China, central and local governments have been setting aside funds for their own big-data programs. The big-data related 973 Projects in China are good examples of this. Industry players have been following in the footsteps of big-data pioneers such as Google, Facebook, Twitter, and Baidu, and more and more companies are rushing into the big-data business. Companies have been analyzing the purchasing behavior of huge numbers of customers and have been devising more attractive plans and policies. Big data is already an important part of the \$64 billion database and data analytics market [3]. Indeed, big data will open up commercial opportunities comparable in scale to those created by enterprise software of the late 1980s, the internet of the 1990s, and the social media explosion today.

However, what is big data? It has been defined in many different ways. We prefer to define big data as data sets that are too big for current information technologies to capture, transmit, store, process, or visualize. Although this definition is simple, it encompasses computing complexity theory, computer architecture, operating system, programming model, database technologies, algorithms, and applications. People from different fields have dramatically different understandings of big data, which is why there is so much excitement and conjecture surrounding it.

In this special issue, we present papers that discuss big-data technology from different perspectives. These are not only high-level surveys but also reports on initial results from big-data projects. Communication infrastructure is one of the most important aspects of big data. Yi Zhu and Zhengkun Mi from Nanjing University of Posts and Telecommunications discuss content-centric networking, which is seen as a promising approach to big-data distribution. They propose a networking architecture for processing big data, and this architecture is fundamentally different from TCP/IP. Shengmei Luo et al. from the Cloud Computing & IT Institute of ZTE Corporation present a survey of big-data analytics. They analyze challenges related to storage, data-mining algorithms, and programming models for big data. They also predict opportunities in the big-data era. Although there are many potential business opportunities in big data, security is of the utmost importance for users and cannot be overlooked. Ruixuan Li et al. from Huazhong University of Science and Technology provide an overview of data security and privacy-preservation for cloud storage. They carefully investigate confidentiality, data integrity, and data availability. They also propose a feasible solution to current security problems. Shigang Chen et al. from the University of Florida delve more deeply into data integrity. They propose a novel authenticated data structure called Cloud Merkle B+ tree that supports dynamic operations such as insertion, deletion and modification. CMBT lowers overhead from $O(n)$ to $O(\log n)$.

Moving to big data applications, algorithms oriented towards a single machine

Big Data: Where Dreams Take Flight

Chengzhong Xu and Zhibin Yu

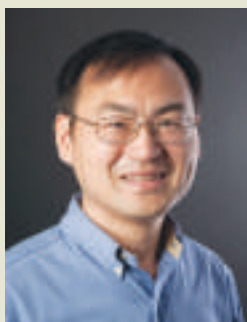
are not necessarily efficient in big-data platforms because many machines need to run concurrently for the same task. Weisong Shi et al. from Wayne State University design a mechanism called SPBD that reduces the response time of big-data systems. This mechanism is very feasible in practice. Zhen-dong Bei et al. report their experiences with big-data applications that use MapReduce/Hadoop. They confirm that manually tuning up to 190 Hadoop configuration parameters is extremely time consuming, if at all possible. They then propose an automatic performance prediction scheme based on random forest to determine the best configuration parameter combinations. Their experimental results show that their scheme can predict the performance of Hadoop systems very accurately.

Challenges and opportunities exist together in the big-data era. We believe most of these challenges will be overcome and opportunities will be realized. Big data is a field where dreams will take flight.

References

- [1] Whitehouse Fact Sheet (2012, Mar. 29). *Big Data Across the Federal Government*. [Online]. Available: http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_fact_sheet_final.pdf
- [2] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. H. Byers (2011, May) McKinsey Global Institute Report. *Big data: The next frontier for innovation, competition, and productivity*. [Online]. Available: http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation
- [3] MASS Technology Leadership Council Report (2012). *Big Data and Analytics: A major market opportunity for Massachusetts*. [Online]. Available: http://76.12.40.240/research/MassTLC_BDR.pdf

New Members of ZTE Communications Editorial Board



Dr. Shigang Chen is an associate professor in the Department of Computer and Information Science and Engineering, University of Florida. He received his BS degree in computer science from the University of Science and Technology of China in 1993. He received his MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign in 1996 and 1999. Prior to joining the University of Florida in 2002, he worked with Cisco Systems for three years. From 2002 to 2003, he served on the technical advisory board of Protego Networks. His research interests include computer networks, internet security, wireless communications, and distributed computing. He has published more than 100 peer-reviewed journal and conference papers. In 1999, he received IEEE Communications Society Best Tutorial Paper Award, and in 2007 he received the NSF CAREER Award. He holds 11 US patents and is an associate editor for IEEE/ACM Transactions on Networking, Elsevier Journal of Computer Networks, and IEEE Transactions on Vehicular Technology. He has been a member of the steering committee of IEEE IWQoS since 2010. He is also a senior IEEE member.



Honggang Zhang is an international chair professor, Université européenne de Bretagne and Supélec, France. He is a full professor in the Department of Information Science and Electronic Engineering, Zhejiang University, China, and co-director of the York-Zhejiang Lab for Cognitive Radio and Green Communications at Zhejiang University. Dr. Zhang is also an honorary visiting professor at the University of York, UK. He received his PhD degree in electrical engineering from Kagoshima University, Japan, in 1999. He was the principle author and contributor of the DS-UWB proposal of the IEEE 802.15 WPAN Standardization Task Group. From September 2004 to February 2008, he led the wireless development teams at CREATE-NET (Italy) and also participated in the European FP6/FP7 projects. From 2011 to 2012, Dr. Zhang chaired the Technical Committee on Cognitive Networks of the IEEE Communications Society. He co-chaired the 2008 IEEE Globecom Symposium and 2013 IEEE ICC Symposium. He was the founding TPC co-chair of CrownCom 2006 and was a member of the CrownCom steering committee from 2006 to 2009. He was the general chair of IEEE/ACM GreenCom 2010. Dr. Zhang was the lead guest editor of IEEE Communications Magazine special issues on green communications. He has co-authored and edited two books: *Cognitive Communications: Distributed Artificial Intelligence, Regulatory Policy & Economics, and Implementation* (John Wiley) and *Green Communications: Theoretical Fundamentals, Algorithms and Applications* (CRC Press).

Content Centric Networking: A New Approach to Big Data Distribution

Yi Zhu^{1,2} and Zhengkun Mi¹

(1.Key Lab of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;
2.School of Computer Science and Telecommunication Engineering, University of Jiangsu, Zhenjiang 212013, China)

Abstract

In this paper, we explore network architecture and key technologies for content-centric networking (CCN), an emerging networking technology in the big-data era. We describe the structure and operation mechanism of a CCN node. Then we discuss mobility management, routing strategy, and caching policy in CCN. For better network performance, we propose a probability cache replacement policy that is based on content popularity. We also propose and evaluate a probability cache with evicted copy-up decision policy.

Keywords

big data; content-centric networking; caching policy; mobility management; routing strategy

1 Introduction

With the development of new network technologies and information services, big data has become the focus of attention in IT [1]. The features of big data are volume, velocity and variety [2]. Volume refers to the massive amounts of data that have to be stored and processed. Velocity refers to the constant updating, caching, and delivery of data. Variety refers to the wide range of data and abundant forms of data representation.

In current IP networks, big data can cause congestion and server overload because IP architecture works in host-to-host mode. However, the problems caused by big data tend to affect the data itself, rather than the host or server. These problems may limit data availability, reduce delivery speed and quality, or compromise data security. Therefore, more efficient data-centric architectures need to be designed to solve the problems created by big data.

Since 2006, several new network architectures have been proposed. These architectures stem from next-generation research projects and include data-oriented network architecture (DONA) [3], proposed by the UC Berkeley RAD Lab; 4WARD [4], proposed as part of the EU's Seventh Framework Programme; publish-subscribe internet routing paradigm (PSIRP)[5] and content-centric networking (CCN)[6], [7], proposed by the Palo Alto Research Center; and named data networking (NDN), proposed as part of the National Science Foundation's Future Internet Architecture (FIA) project.

Of these architectures, CCN represents a sophisticated technical advancement and also comes under the umbrella of NDN.

In CCN, each piece of content is uniquely named, and the content is separated from its location. If we replace traditional routing (based on host address) with new content-based routing, the requested content can be obtained in the nearby CCN node. This node caches the content, and there is no need to forward the request to the far-away content source. The caching mechanism is the key technology of CCN. It can reduce the response time for accessing content, and it can alleviate network congestion and server overload in a big-data environment.

2 CCN Architecture and Operating Mechanism

In current IP networks, CCN uses the content name instead of IP address for routing. An hierarchical naming mechanism similar to a URL is used. An example of this mechanism is `njupt.edu.cn/Video/Computer_Networks/Lecture_1.mpeg`, where `/njupt.edu.cn/Video/Computer_Networks` is the prefix for retrieving and forwarding the content, `/njupt.edu.cn` represents the content provider, `Video/Computer_Networks` represents the content type, and `/Lecture_1.mpeg` represents the content itself.

There are two kinds of packets in CCN: interest and data. Interest packets contain content identification, selector, and nonce. The selector comprises order preference, publisher filter, and scope. Data packets contain signature, signed info, key locator and stale time, and content. The signature comprises di-

This work is supported by National Natural Science Foundation of China under Grant No.60872018 and No. 60902015, and Major National Science and Technology Project No. 2011ZX03005-004-03.

Content Centric Networking: A New Approach to Big Data Distribution

Yi Zhu and Zhengkun Mi

gest algorithm and witness, and the signed info comprises a publisher ID.

An interest packet with ID is sent by the requester, and CCN nodes forward the packet until it reaches a node that can provide the requested content according to the maximum matching principle. Then, the data packet is used to send the content back to the requester via the reverse interest packet forwarding path. This completes the communication.

The key structure of a CCN node comprises a content store (CS), a pending interest table (PIT), and a forwarding information base (FIB). The CS stores content within the node cache. The PIT records the received interest packets for a pending response to the arriving face and accompanying content. Here, “face” is the CCN terminology for interface. The FIB indicates the next hop for forwarding the interest packets. The requested content is cached as much as possible in the CCN nodes during backward delivery so that the content can be quickly provided to subsequent users. This is a completely different to the way a traditional IP router works. Usually, a traditional IP router clears the cache on forwarding.

A maximum matching query is executed in CS, PIT, and FIB in turn when an interest packet arrives at the node. If the requested content is found in the CS, the content is sent to the requester through the arrival face of the interest packet. If the requested content is not found in the CS, the PIT is queried. If the PIT contains the related content entry, the PIT indicates that the content request has been received and waits for response. In doing so, it adds the arrival face the content’s entry; otherwise, the FIB is queried further on. If the FIB has the related content entry, the interest packet is forwarded through the face indicated by the FIB. If no match is found in the FIB, the interest packet is dropped.

The structure of a CCN node is shown in Fig. 1. Suppose an interest packet with content name /njupt.edu.cn/Video/Computer_Networks/Lecture_1.mpeg arrives. The content can be fetched from the CS and sent back to the requester. If an interest packet with content name /njupt.edu.cn/Video/Signal_System/Lecture_1.mpeg arrives from face 2, the PIT has to be queried because the CS does not have this content. The PIT already contains a request entry for this content, and the face number is 3. Therefore, face 2 is added to the face field of that entry. If an interest packet with content name /njupt.edu.cn/Video/Stochastic_Process/Lecture_1.mpeg arrives, and the content name is not contained in the CS nor PIT, then the FIB is queried. This indicates that face 5 is the correct forwarding face. Then, the interest packet is forwarded to the next node through face 5, and the requested content name and interest-packet arrival face is added to the PIT.

For the sake of network scalability, the FIB has a mechanism to aggregate multiple content prefixes into one entry. This reduces the size of the FIB. However, this mechanism cannot be used in the PIT. Reducing the size of the PIT is an important area of research in CCN because the PIT be-

comes excessively large for big-data applications. To scale the CS storage, an appropriate cache replacement policy should be used to free the cache so that it can hold newly obtained content that is of higher importance. Content will be divided into chunks in the delivery process. CS replaces and stores content in the chunk.

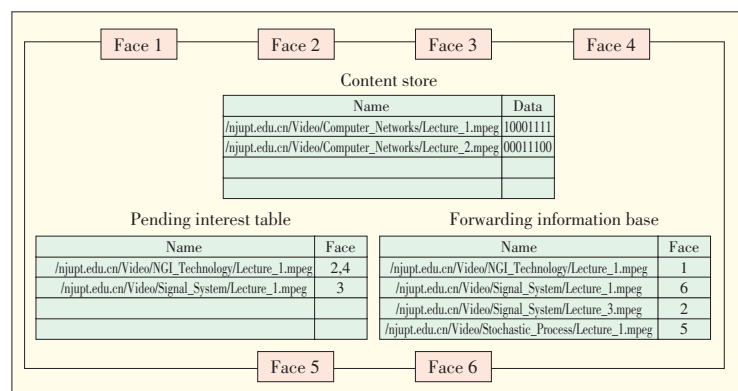
For comparison, the content delivery process for an IP network and CCN network is shown in Fig. 2. With the IP client/server infrastructure, each piece of content delivered has a round trip from the request user to the source server. A request that involves a large amount of content involves a huge amount of network traffic that is likely to cause network congestion or server overload. With the CCN infrastructure, the user may obtain the content from the cache of a nearby node. This eliminates traffic further along the line to and from the source sever. In Fig. 2, the request from user 1 goes to the source server (as in a conventional IP network). However, the content can be cached in routers R2, R4, R5 and R7 on its way back to user 1. If user 2 subsequently requests the same content, R2 can deliver it because there is a content copy in its cache. Similarly, the content can be cached in R1 on the way to user 2. When user 3 requests the same content, they can simply get it from the neighboring router R1. This only involves one hop. It can be seen that, as a distributional resource-caching and managing infrastructure, CCN is a good fit for big data. Through the caching mechanism and content identification, terminal users can obtain content from the network node that is as near to the user as possible. This limits delay, congestion, and performance fluctuations caused by big data.

3 Key Aspects of CCN

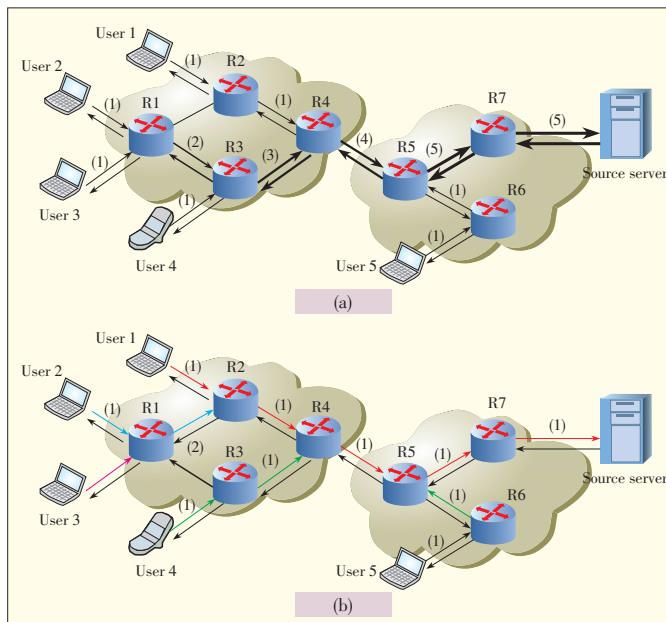
Mobility management, routing strategy, and caching policy all affect the performance of a CCN.

3.1 Mobility Management

For non-real-time services, such as web pages, email, and file sharing, the location of source server is fixed so that the content name is unchanged. When moving to a new site, the us-



▲ Figure 1. Structure of the CCN node.



▲ Figure 2. a) IP-based network infrastructure and b) CCN-based network infrastructure.

er can request content as before. Although some retransmission delay may be incurred, it practically affects the services because they are tolerant to delay.

For real-time services, such as internet telephony, instant messaging, and gaming, the situation is more complicated. Usually, both the source and the user are mobile, which means the prefix of content name may change. CCN routers need to update their routing tables to guarantee correct forwarding. This causes additional time overhead, an invalid forwarding entry in the FIB, and a huge FIB. Additional time overhead cannot be tolerated by real-time services, which are interrupted when delay is more than 150 ms. If many of the forwarding entries in the FIB are invalid, incorrect forwarding will result, and network resources will be wasted. If the FIB is huge, the content prefixes (which change as a result of the mobility of the content source) will be difficult to handle. The worst case scenario occurs when the two parties in the communication move at the same time. As with the mechanism in session initiation protocol (SIP), a fixed registry server can be set up to exchange content names between sides. Of course, this involves additional registration time.

Several mobility management schemes have been proposed for CCN. Problems with managing the mobility of both user and content are outlined in [8], but no solution is given. In [9], [10], proxy-based mobility (PBM) management scheme is proposed. In this scheme, the content requested by the user is cached before moving through the proxy server. Efficient use is made of CCN shared content resources to reduce delay during handover and acquisition. The drawback of this scheme is that content request and acquisition still relies on the traditional IP network. In [11], selective neighbor caching (SNC) is proposed

for mobility support. A group of optimal neighboring proxy servers is selected to proactively request and store content that the user fails to receive as the content moves through the proxy server. When selecting a neighbor proxy, a tradeoff is made between the cost of acquiring the content and the cost of caching the content in proxies. SNC can reduce delay to a large extent but does not use CCN shared content resources. In [12], a partial route update scheme is proposed to reduce negative effects caused by content provider mobility. After the movement path has been determined, routers are chosen, and their content prefixes are updated. The cost of updating routers is reduced. However, there has been no in-depth analysis of the tradeoff between the number of routers updated and the routing miss probability. In [13], a tunnel is set up between the CCN router in the home domain and the CNN router in the foreign domain in order to redirect the interest packet. This provides real-time support when the content source is mobile. The evaluation in [13] shows that the tunnel reduces delay in a network with many nodes.

3.2 Routing Strategy

The semantics and basic processing mechanisms of IP and CCN routing protocols are similar. Hierarchical identifier naming, longest matching lookup, and forwarding mechanism for an IP network can all be used in CCN [7].

Interior router protocols, such as open shortest path first (OSPF) and intermediate system to intermediate system (IS-IS), provide a type-length-value (TLV) option that can be easily used by CCN to publish the content prefix (even though the prefix is different for CCN and IP). Interior router protocols also ignore an unknown message so that the CCN node can connect directly to the IP network running IS-IS or OSPF. This does not adversely affect the network.

For an existing external routing protocol, which is similar to internal gateway protocol (IGP), border gateway protocol (BGP) can also use TLV for interdomain announcement of address information. Different CCNs can be interconnected by announcing content information to each other. This can be done by integrating the content prefixes of the domain into the BGP.

Although existing IP routing strategies can be used in CCN, a specially designed strategy inevitably improves the performance of CCN. Until now, there have only been a few studies on CCN routing strategy. The four strategies reported are: all forwarding, random forwarding, ant colony forwarding, and improved ant colony forwarding.

All forwarding is a basic strategy in which interest packets are forwarded to all the faces matching the prefix in the FIB. The advantage of this strategy is there is less delay during data packet return. The drawback of this strategy is the large amount of redundant traffic that results from the dispatching of multiple interest packet copies. This problem worsens as the network increases in size.

With random forwarding, one face is randomly chosen

Content Centric Networking: A New Approach to Big Data Distribution

Yi Zhu and Zhengkun Mi

among multiple matching faces indicated by the FIB. The chosen face forwards the interest packet. Random forwarding does not lead to any redundant traffic, but it cannot guarantee fast and stable network performance.

Ant colony forwarding is a distributed routing selection strategy in which the ant colony optimal algorithm sends out an exploratory packet to source an optimal forwarding path [14]. An optimal path has the least number of hops to the source server or the lightest-loaded nodes. There may be a tradeoff between hops and load. The path is optimal in the traditional sense and is formed by request node and all the optimal faces of the intermediate nodes. Traffic redundancy can be reduced to some extent, but the path may not be optimal for CCN because content caching in routing nodes is not taken into account with ant colony forwarding.

In [15], a neighbor cache explore (NCE) routing strategy is proposed. This strategy is an improvement of that in [14]. The shortest path is found using ant colony algorithm under the condition of a non-cache network. Then, exploratory packets are sent to the nodes within a particular range (neighbor nodes) to determine their caching status. Finally, a decision is made on whether the requested content can be acquired in the nodes along the shortest path. With NCE, the caching capability of CCN is taken into account, but the shortest path found using the ant colony algorithm may not contain nodes that cache the requested content.

A reasonable CCN routing strategy helps find the node that caches the requested content and is also as near as possible to the requester. Using a traditional routing strategy to find the least-cost path first is not reasonable. Future research is needed into a probabilistic routing strategy for an opportunistic network. In such a routing strategy, the first step involves exploring the path that has the nearest possible node that can provide the content.

3.3 Caching Policy

There are two kinds of CNN caching policy: cache replacement and cache decision. The former involves selectively replacing cached content with newly arrived content. The latter involves making a decision about caching newly arrived content.

3.3.1 Cache Replacement Policy

There are four classes of cache replacement mechanism that can be found in existing caching policies: recency-based, frequency-based, utility-based, and probability-based. Existing CCN caching policies all originate from basic web caching policies.

A recency-based mechanism selects the content to be replaced when the cache is full. Content is selected according to how much it has been used over a period of time. Least recently used (LRU) [16], [17] is the most common policy in this class, and other policies can be regarded as a variation of this.

LRU stems from the web. Whenever there is a hit on a piece of content, the content is moved to the head of the cache so that less frequently used content is replaced when the cache is full. The rationale for this is that recently used content will probably be requested again. LRU is easy to implement.

CCN is a variation of LRU and uses two replacement policies: MRU and MFU. Most recently used (MRU) and most frequently used (MFU) [18] target the multicache architecture of information-centric networking. Assuming that the cache decision policy is to cache everywhere, the requested content is stored in each node on the content delivery path. A hit in one node implies a high probability that a copy of the same content is being stored in neighbor nodes. The most recently used and most frequently used content should be removed when the cache is full.

The frequency-based replacement mechanism is similar to the recency-based replacement mechanism except that the former takes usage, specifically, the number of visits to a piece of content, to determine which content is to be replaced. A side effect of this is called “cache pollution.” If a piece of content was popular in the past, it will stay in cache for a long time, even if it is not used any more. This will remain the case until new content becomes more popular and more often visited. The most frequently used content prevents newer content from entering the cache. Several mechanisms, including an aging mechanism, have been proposed to solve this problem, but they all increase complexity.

LFU is used in the web [19], [20]. When new content arrives, the least-visited content from the past should be replaced. Most frequency-based replacement approaches have their foundations in the web are not designed for the dynamic interests of CCN users. In [21], a new policy called recent usage frequency (RUF) is proposed for CCN. Unlike traditional web-based policies that count content visits in the output face, RUF counts visiting frequency using interest packets arriving at the node. The benefit of changing the checkpoint is that changes in user interests can be instantly detected, and caching policy can be promptly adapted. The content with low instant popularity will be removed when the cache is full.

The utility-based replacement mechanism uses a utility value, for example, content size or content lifetime, as an index to decide which content should be replaced. The choice of utility parameter and calculation of utility value affects the performance and complexity of the mechanism. An example of a utility-based replacement mechanism is size-based policy used in the web [22]. Document size is taken as the utility parameter on the basis that a user tends to request small-sized content. With this rationale, larger content should be removed first. For the content of similar size, an LRU policy is used.

An age-based cooperative (ABC) policy has been proposed for CCN [23]. The age of content is taken as the utility parameter. The distance of the content from the requester and the popularity of the content allows a node to calculate a unique age

for each piece of content in its cache. This indicates the life-time of the content. Only when a timeout event occurs is the corresponding content removed; otherwise, it should be retained in the cache.

A probability-based replacement mechanism reduces the complexity of a policy but does not sacrifice performance too much. Evaluating performance, however, is a little complex because performance differs in different network environments. Uniform random replacement (UNIF) is a probability-based policy that is used in the web to randomly select content to be replaced with uniform distribution.

Randomized replacement (RR) policy is one attempt at a probability-based replacement policy for CCN [24]. N pieces of content are selected randomly from the cache, and the least useful piece of content is removed. The usefulness of content can be determined by any utility function. The remaining M ($M < N$) contents are retained in the cache. The next time round, $N-M$ new samples are drawn from the cache, and the replacement mechanism is executed again.

3.3.2 Cache Decision Policy

Cache decision policy is used to determine whether the arriving content should actually be stored in the cache. Less attention has been given to this type of policy than has been given to cache replacement policy. Caching everywhere, also called leaving copy everywhere (LCE), is the default policy for CCN. However, LCE is not a good policy because it increases redundancy and increases the probability of misses. As with replacement policies, current CCN decision policies derive from existing web policies [25]. These web decision policies include LCE, leave copy down (LCD), move copy down (MCD), and leave copy probability (LCP).

LCE is a commonly used decision policy in multilevel cache architectures. All the middle nodes on the content-delivery path cache the content copy, which is hit in the level i node.

With LCD, the content copy that is hit in the level i node is cached only in the downstream node (i.e. level $i-1$ node). The content copy eventually goes down from level L to level 1 and is cached thereafter at least every $i-1$ requests.

MCD is an improvement on LCD. The copy hit in the level i node is moved to the downstream node (level $i-1$ node), and the copy in the level i node is deleted.

With LCP, the content copy hit in the level i node is cached in the nodes on the content-delivery path with probability p .

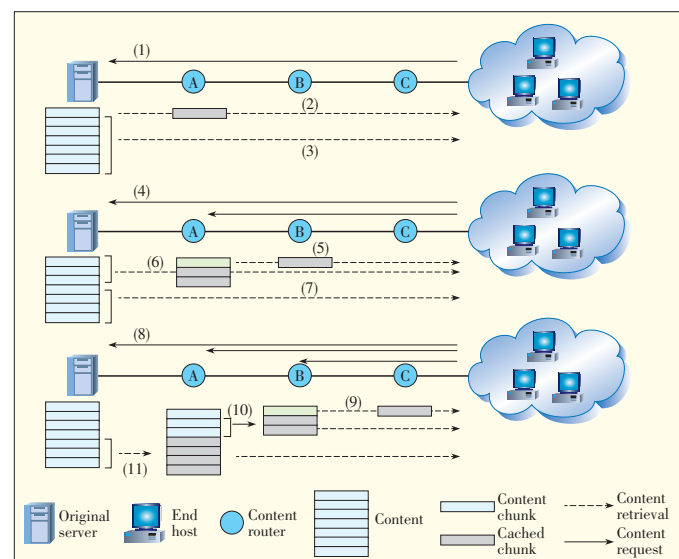
Existing CCN decision policies include WAVE, less-for-more, and ProbCache. WAVE is an example of popularity-based and collaborative in-network caching for CCN [26]. With WAVE, content is divided into chunks (Fig. 3). If a user requests chunk x and gets a hit in node i , chunk x is sent back to the requester and, at the same time, is stored in the next node (node $i-1$). If a request for the same content gets another hit in node $i-1$, then chunk x is stored in node $i-2$, and chunks $x+1$ and $x+2$ are stored in node $i-1$. The number of

stored chunks increases exponentially with an increase in the number of request for chunk x . With this policy, the relevance of requests between chunks is taken into consideration. When a user requests a chunk of content, there is a high probability they will request the rest of the chunks. Therefore, pushing the rest of the chunks to a node nearest user reduces.

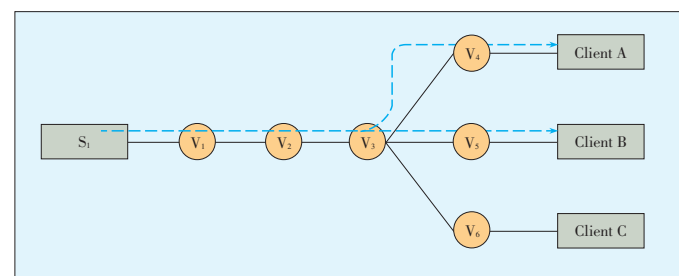
Less-for-more policy is an improvement on LCE proposed in [27]. By storing the content in specific nodes on the backward path can achieve the goal of gaining maximum benefit with minimum copy storage.

Assuming there are M paths from node i to j , and node x is on the m th path, m/M is the importance of node x . When there is a backward delivery of content, the content copy is stored in the selected node according to node's importance. v_3 is the key node on the delivery path (Fig. 4). When v_3 stores the copy of the content, clients A to approximately C can acquire the content via the path with the least hops. Thus, nodes v_1 and v_2 need not store a copy and can remain free to store other content. Both hit probability and network utilization are high for CCN, and there are many different types of content copies provided by the limited nodes.

In [28], a probabilistic algorithm for distributed content caching is proposed. This algorithm, called ProbCache, counts



▲ Figure 3. WAVE CCN decision policy topology.



▲ Figure 4. An example less-for-more topology.

Content Centric Networking: A New Approach to Big Data Distribution

Yi Zhu and Zhengkun Mi

the number of nodes through which an interest packet and data packet have passed. It saves this number in the head of the packet in order to evaluate the capability of the path to cache content. The capability indicator, based on path length and multiplexed content flow, is a probabilistic value that can be used to decide whether content needs to be stored or not.

4 Improved CCN Caching Policies

Here, we describe two caching policies specifically for CCN. The first policy is a replacement policy that reduces the probability of missed requests for low-popularity content. The policy balances the stored proportion of different classes of content in cache. The second policy is a decision policy that performs better than traditional cache decision policies because it extends content survival time in the network.

4.1 Replacement Policy Based on Content Popularity

Requests for content always follow a certain popularity distribution. To balance the load in a CCN, a good replacement policy needs to balance the proportions of content (with different popularities) that is cached in network nodes. Unfortunately, all existing CCN replacement policies previously mentioned in this paper ignore the issue of content popularity, and this leads to relatively poor performance.

In [24], a replacement policy based on popularity preference is proposed. Two chunks are selected randomly, and the more popular chunk is replaced. The goal of this policy is to cache less-popular content longer and guarantee the uniform distribution of content with different popularities. A drawback of this policy is that less popular chunks may not be replaced for a long period of time.

We propose a replacement policy based on content popularity probability (PP) [29]. PP policy is suitable for highly concentrated content requests. It can improve performance for most content by sacrificing a little performance for the most popular content. Assuming that each replacement removes the chunk at the tail of the cache queue, the chunk position indicates the chunk's replacement priority. If a new chunk is to be cached, the PP decides where to insert it according to its popularity. A less popular chunk is inserted the nearer to the top of the queue. In this way, PP policy can extend the time in cache of less popular content and thus reduce the probability a request for this content will be missed. It solves the problems faced by the policy in [24].

4.1.1 Description of Proposed Policy

Assuming the cache comprises C chunks, when a chunk with class k popularity is hit, it is inserted at the i th position with probability $p_k(i)$. This probability is given by

$$p_k(i) = \frac{1}{a} e^{\beta(k-1)\left(1-\frac{i}{C-1}\right)} \text{ for } i \in [0, C-1], k \in [1, K] \quad (1)$$

where, K is the sum of content popularity classes, and a and β are probability adjustment factors. The probability adjustment factor a is given by

$$\sum_{i=0}^{C-1} p_k(i) = 1 \Rightarrow a \approx \frac{C-1}{\beta(k-1)} [e^{\beta(k-1)} - 1] \quad (2)$$

Existing chunks in the i th (or later) positions in the cache shift one place backward in the order, and the chunk at the queue tail is removed if the cache is full. A chunk in a physical router is always shifted by simply modifying the pointer that indicates the position of the corresponding chunk. The chunk itself is not moved, so time overhead is reduced. In (1) and (2), the recommended value of β is [1], [2]. As β increases, low-popularity content at the front of cache queue is more likely to be replaced.

4.1.2 Evaluation of the Performance of the Proposed Policy

We compared PP and LRU for user-generated video, which is typical big data. We used Matlab to run a simulation on request miss probability. Fig. 5 shows the network topology.

The simulation parameters were taken from [30]. The network is a triple-level tandem architecture. The CCN provided a total of M content files, where $M = 40,000$. These files were divided into K classes, where $K = 400$. Each class had m content files, where $m = 100$, and each file was divided into 10 kB chunks. Requests for content in class k are generated according to a Poisson process with intensity

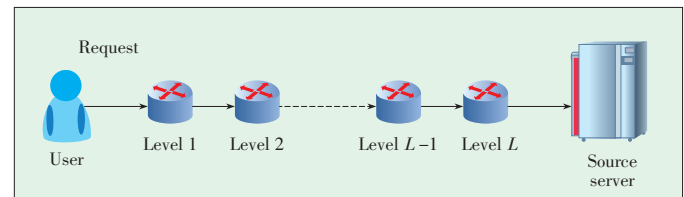
$$\lambda_k = \lambda_1 \cdot q_k \quad (3)$$

where λ_1 is the request rate at the first level ($\lambda_1 = 40$ pieces of content per second in simulation), and q_k follows the Zipf distribution, which describes the popularity of arrival requests for content in class k [30]. The Zipf distribution is given by

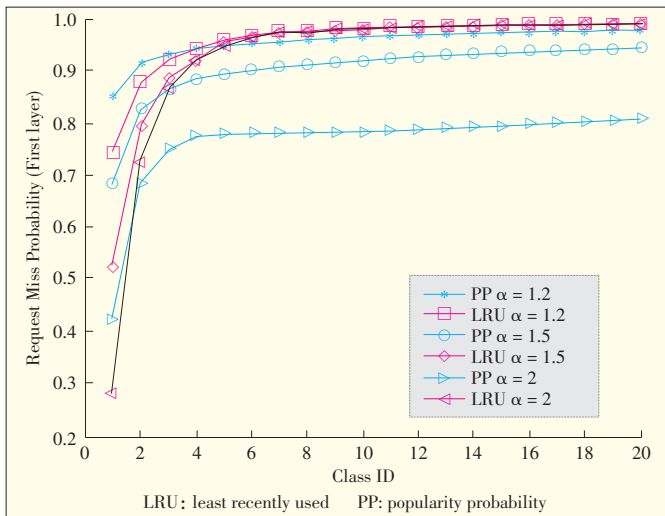
$$q_k = c / k^\alpha \text{ for } c > 0 \quad (4)$$

where α is the concentrations of content popularity. A larger α means the requests are more concentrated in the first several classes of content. Fig. 6 shows the performance of PP and LRU in the first-level CCN node.

The RMP of content increases as k increases. PP and LRU perform well only for first several classes (i.e. those comprising content with a small k , for example, $k < 2$). As α increases, LRU only performs well for the most popular classes but performs poorly for classes with class id $k > 2$. An increase in α means there are more concentrated requests for the first sever-



▲ Figure 5. L -level CCN network topology.



▲ Figure 6. Node performance of PP and LRU.

al classes, but LRU cannot adapt to variations in content popularity. PP increases cache hits because it has an adaptive content popularity distribution policy. When $\alpha = 1.2$, this improvement is slight, but as α increases, the improvement is greater. This means that PP is more suited to a network with highly concentrated content requests. When $\alpha = 2$, the RMP for $k = 1$ class is worse than the LRU for that class. The RMP for $k \geq 2$ classes shows definite improvement. This improvement is caused by the decrease in hits on the $k = 1$ class). PP sacrifices request hits on popular classes but slightly increases the hit distance of $k = 1$. It does this in order to improve performance for other classes and shorten their hit distances [29].

4.2 Probability Cache with Evicted Copy Up Decision Policy

LCD and MCD reduce the number of copies in the cache and extend the time needed to cache them. However, in an L -level tandem network, only after at least $L-1$ requests can a copy be moved to the level 1 node. It is no good for users to access the content from a nearby node. LCP is potentially a better policy because it increases the amount of popular content cached in nearby nodes and reduces the hit distance. However, storage efficiency is reduced, and the time needed to cache the requested content decreased because LCP leads to greater redundancy.

To tackle the above problems, we propose a policy called probability cache with evicted copy up (PCECU). This policy is designed to keep the copies as long as possible in order to increase the hit probability. In the meantime, it also increases the amount of popular content cached near the user in order to reduce hop count and delay during acquisition.

4.2.1 Description of Proposed Policy

If a request for a chunk receives a hit at level i , then the chunk is moved to the level 1 node with probability p and is de-

leted from the level i node. (The chunk is not deleted if node i is the original server). The chunk remains cached in level i with probability $1-p$. Except for the level 1 node, no node on the backward delivery path caches the chunk while delivering it to level 1. If the content chunk cached in level i node is eliminated from the cache to make way for new chunks, this chunk is moved to its upstream node (i.e. $i+1$ node). The chunk is then cached in the head of the $i+1$ node.

4.2.2 Evaluation of the Performance of the Proposed Policy

In our Matlab simulation, we delivered UGC in a 3-level tandem CCN (Fig. 5). The performance parameters were the source server hit probability (SSHP) for the k th content chunk, and the average hit distance (AHD). The CCN provided a total of M content files, where $M = 40,000$. These files were divided into K classes, where $K = 400$. Each class comprised m content files, where $m = 100$, and each file was divided into 10 kB chunks. Requests for content in class k were generated according to a Poisson process with intensity given by (3). The definition of λ_1 and q_k were the same as in section 4.1. In this simulation, $\lambda_1 = 40$ pieces of content per second, and 5×10^7 requests for first-level chunks are randomly generated.

Fig. 7 shows that LCE always performs the worst, and LCP performs the second worst but improves as p decreases. PCECU definitely improves CCN performance it allows probability caching and extends caching time.

5 Conclusion

CCN is a promising network infrastructure for the big-data era. It is content-centric, not host centric in the traditional IP network. A request for content can get a hit in a nearby node and does not need to travel far away to the source server. This alleviates network congestion and significantly reduces delay.

At present, key aspects of CCN being studied include caching policy, naming mechanism, content retrieval, routing strategy, mobility management, and security. Much attention has been paid to CCN at home and abroad. CCN could potentially revolutionize the internet by providing full-scale network support for big data.

References

- [1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. Hung-Byers. (2011, May). "Big data: The next frontier for innovation, competition, and productivity." McKinsey & Company Global Institute. [Online]. Available: http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation,2012
- [2] R. E. Bryant, R. H. Katz, E. D. Lazowska. (2008, Dec.). "Big data computing: Creating revolutionary breakthroughs in commerce, science, and society." Computing Community Consortium White Paper. [Online]. Available: <http://www.cra.org/ccc/resources/ccc-led-white-papers>
- [3] T. Koponen, M. Chawla, C. B. Gon and et al., "A data-oriented (and beyond) network architecture," in *Proc. SIGCOMM '07*, Kyoto, Japan, pp. 181–192.
- [4] "Project FP7 4WARD," [Online]. Available: <http://www.4ward-project.eu>, 2010.
- [5] "Project PSIRP," [Online]. Available: <http://www.psirp.org>, 2010.
- [6] V. Jacobson, D. K. Smetters, J. D. Thornton et al., "Networking named content," in *Proc. 5th ACM Int. Conf. on Emerging Netw. Experiments and Tech. (CoN-*

Content Centric Networking: A New Approach to Big Data Distribution

Yi Zhu and Zhengkun Mi

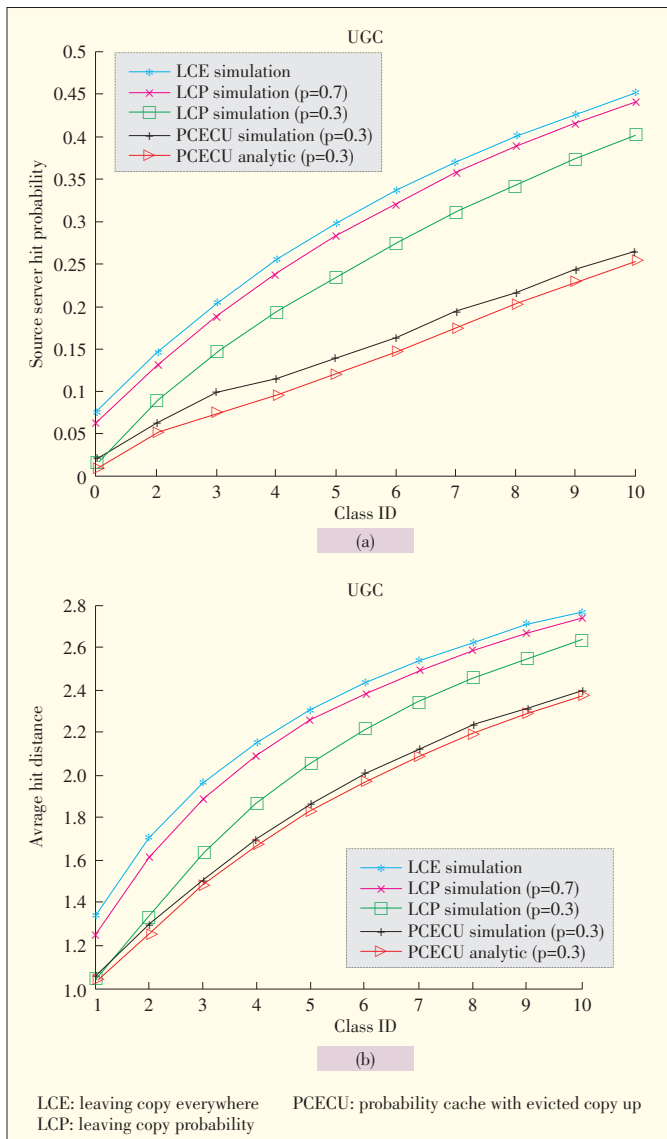


Figure 7. a) Source serve hit probability for LCE, LCP and PCECU under UGC. b) Average hit distance for LCE, LCP and PCECU under UGC.

EXT'09), Rome, Italy, pp.1–12.

- [7] V. Jacobson, D.K. Smetters, J.D. Thornton and et al., “networking named content,” in *Commun. of the ACM*, vol. 55, no. 1, pp. 117–124, 2012.
- [8] D. Kim, J. Kim, Y. Kim et al., “mobility support in content-centric networks,” in *Proc. ACM SIGCOMM ICN Workshop*, Helsinki, Finland, pp.13–18, Aug. 2012.
- [9] J. Lee and D. Kim, “Proxy-assisted content sharing using content-centric networking (CCN) for resource-limited mobile consumer devices,” in *IEEE Trans. Consumer Electronics*, vol. 57, no. 2, pp. 477–483, 2011.
- [10] J. Lee, D. Kim et al., “Proxy-based mobility management scheme in mobile content-centric networking (CCN) environments,” in *Proc 19th IEEE Int. Conf. Consumer Electronics (ICCE'11)*, Las Vegas, NV, pp. 595–596.
- [11] X. Vasilakos, V. A. Siris, G. C. Polyzos et al., “Proactive selective neighbor caching for enhancing mobility support in information-centric networks,” in *Proc. ACM SIGCOMM ICN Workshop*, Helsinki, Finland, pp.61–66, Aug. 2012.
- [12] J. Lee, D. Kim, M. Jang and et al., “Mobility management for mobile consumer devices in content centric networking,” in *Proc. 20th IEEE International Conference on Consumer Electronics (ICCE'12)*, Singapore, pp. 502–503, 2012.

- [13] J. Lee, S. Cho and D.Y. Kim, “Device mobility management in content-centric networking,” *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 28–34, 2012.
- [14] S. Shanbhag, N. Schwan, I. Rimaq and et al., “SoCCeR: services over content-centric routing,” in *Proc. ACM SIGCOMM Workshop on Information-Centric Networking (ICN'11)*, Toronto, Canada, pp.62–67, Aug. 2011.
- [15] R. S. Ye, M. W. Xu, “Neighbor cache explore routing strategy in NDN,” in *J. Frontiers of Comput. Science and Tech.*, vol. 6, no. 7, pp. 593–601, 2012.
- [16] J. Wang, “A survey of web caching schemes for the internet,” in *ACM Comp. Commun. Review*, vol. 29, no. 5, pp. 36–46, Oct. 1999.
- [17] A. Vakali, “Proxy cache replacement algorithms: A history-based approach,” *World Wide Web Journal*, vol. 4, no. 4, pp. 277–297, Dec. 2001.
- [18] K. Katsaros and V. Konstantinos et al., “MultiCache: An incrementally deployable overlay architecture for information-centric networking,” in *Proc. IEEE Conf. Comp. Commun. Workshops (INFOCOM'10)*, San Diego, CA, pp. 1–5.
- [19] S. Podlipnig and L. Boszormenyi, “A survey of web cache replacement strategies,” in *ACM Comput. Surveys*, vol. 35, no. 4, pp. 374–398, 2003.
- [20] M. Arlitt, L. Cherkasova, J. Dille, R. Friedrich, and T. JIN, “Evaluating content management techniques for Web proxy caches,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 4, pp. 3–11, 2000.
- [21] S. J. Kang, “A recent popularity based dynamic cache management for content centric networking,” in *Proc. 4th Int. Conf. Ubiquitous and Future Netw. (ICUFN'12)*, Phuket, Thailand, pp. 219–224.
- [22] S. Williams, M. Abrams, C. R. Standridge and et al., “Removal policies in network caches for World-WideWeb documents,” in *Proc. ACM SIGCOMM'96*, Stanford, CA, pp.293–305.
- [23] Z. X. Ming, M. W. Xu, and D. Wang, “Age-based Cooperative Caching in Information-Centric Networks,” in *Proc. 21st IEEE Conf. Comp. Commun. Workshops (INFOCOM WKSHPS'12)*, Orlando, FL, pp. 268–273.
- [24] D. Rossi and G. Rossini, “Caching performance of content centric networks under multi-path routing (and more),” Telecom ParisTech technical report, 2011. [Online]. Available: <http://netlab.pkusz.edu.cn/wordpress/wp-content/uploads/2011/10/Caching-performance-of-content-centric-networks-under-multi-path-routingand-more.pdf>
- [25] N. Laoutaris, S. Syntila, and I. Stavrakakis, “Meta algorithms for hierarchical web caches,” in *Proc. IEEE Int. Performance Comput. and Commun. Conf. (IPCCC'04)*, Phoenix, AZ, pp. 445–452.
- [26] K. Cho, M. Y. Lee, and et al., “WAVE—Popularity-based and collaborative in-network caching for content-oriented networks,” in *Proc. IEEE Conf. Comp. Commun. Workshops (INFOCOM WKSHPS'12)*, Orlando, FL, pp. 316–321.
- [27] W. K. Chai, D. L. He, I. Psaras et al., “Cache less for more in information-centric networks,” in *NETWORKING 2012*, Part I, LNCS 7289, pp. 27–40, 2012.
- [28] I. Psaras, W. K. Chai and G. Pavlou, “Probabilistic in-network caching for information-centric networks,” in *Proc. 2nd ICN Workshop on Information-Centric Networking*, Helsinki, Finland, pp. 55–60, Aug. 2012.
- [29] Yi Zhu, Zheng-kun Mi, Wen-nai Wang, “A Cache Probability Replacement Policy based on Content Popularity in Content Centric Networks,” in *J. Elec. and Info. Tech. (in Chinese)*, vol. 35, no. 6, Jun. 2013.
- [30] G. Carofoglio, M. Gallo et al., “Modeling data transfer in content-centric networking (extended version),” in *Proc. 23rd Int. Teletraffic Congress*, San Francisco, CA, pp. 111–118, Sep. 2011.

Manuscript received: April 5, 2013

Biographies

Yi Zhu (zhuyi@ujs.edu.cn) received his MS degree in computer application technology from Jiangsu University, China, in 2006. He is an associate professor in the School of Computer Science and Telecommunication Engineering, Jiangsu University. He is currently pursuing his PhD degree at Nanjing University of Posts and Telecommunications. His research interests include information-centric networking, content-centric networking, and green networking.

Zhengkun Mi (mizk@njupt.edu.cn) is a professor in the College of Communication and Information Engineering, Nanjing University of Posts and Telecommunications. He is also a fellow of the Chinese Institute of Communications and a member of the Chinese Expert Delegation to ITU-T SG16. For more than 30 years, he has been researching and teaching switching and network technologies. His research interests include next-generation network theory and technologies, heterogeneous network and service convergence, and network virtualization.

Big-Data Analytics: Challenges, Key Technologies and Prospects

Shengmei Luo, Zhikun Wang, and Zhiping Wang

(Cloud Computing and IT Institute of ZTE Corporation, Nanjing 210012, China)

Abstract

With the rapid development of the internet, internet of things, mobile internet, and cloud computing, the amount of data in circulation has grown rapidly. More social information has contributed to the growth of big data, and data has become a core asset. Big data is challenging in terms of effective storage, efficient computation and analysis, and deep data mining. In this paper, we discuss the significance of big data and discuss key technologies and problems in big-data analytics. We also discuss the future prospects of big-data analytics.

Keywords

big data; mass storage; data mining; business intelligence

1 Introduction

Following on the heels of the PC and internet, cloud computing will be the third wave of IT and will fundamentally change production and business models.

Cloud computing improves data convergence, storage, and processing and makes it easier to extract value from data. More and more intelligent terminals and sensing devices are being connected to networks, and data is rapidly becoming more varied and extensive (Fig. 1).

Data-processing technologies have developed steadily for a long time, but big data has suddenly caused two significant changes. Now, all data can be saved. This means that applications that require data to be accumulated prior to implementation can be more easily implemented. Also, there has been a

shift from data shortage to data flood. Such a flood has created new challenges for data applications. Simply acquiring data from search engines is no longer sufficient for today's applications. It is increasingly difficult to efficiently obtain and process useful data from a mass of data.

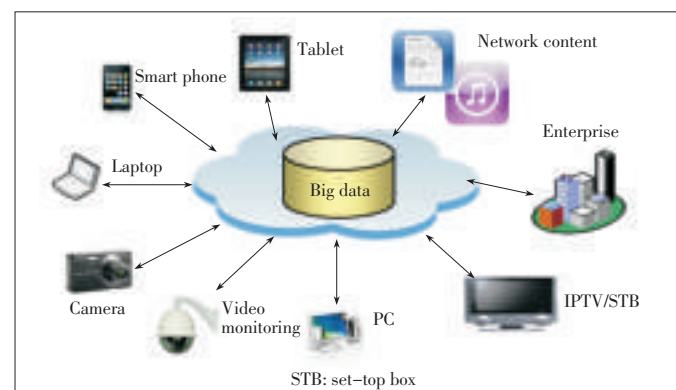
In 2012 Hype Cycle for Emerging Technologies, Gartner stated that cloud computing was falling into a "trough of disillusionment," meaning that technology was set to mature after years of being hyped (Fig. 2) [1]. Big data is rapidly moving from the phase of "technology trigger" to the phase of "peak of inflated expectations." Technological breakthroughs in big data will continue to be made so that big data will be widely used in the next two to five years.

2 Definition and Features of Big Data

Big data implies a collection of data sets that are so large and complex that they are difficult to manage using traditional database management tools or data-processing applications [2]. Compared with traditional data, big data has greater volume; it is more varied; and it derives from a greater range of sources. Big data is not a simple mass of data nor is it only a cloud computing application. The key to big data is deriving valuable information from a mass of data. The features of big data can be summarized as the "four Vs": volume, variety, velocity, and value (Fig. 3).

2.1 Volume

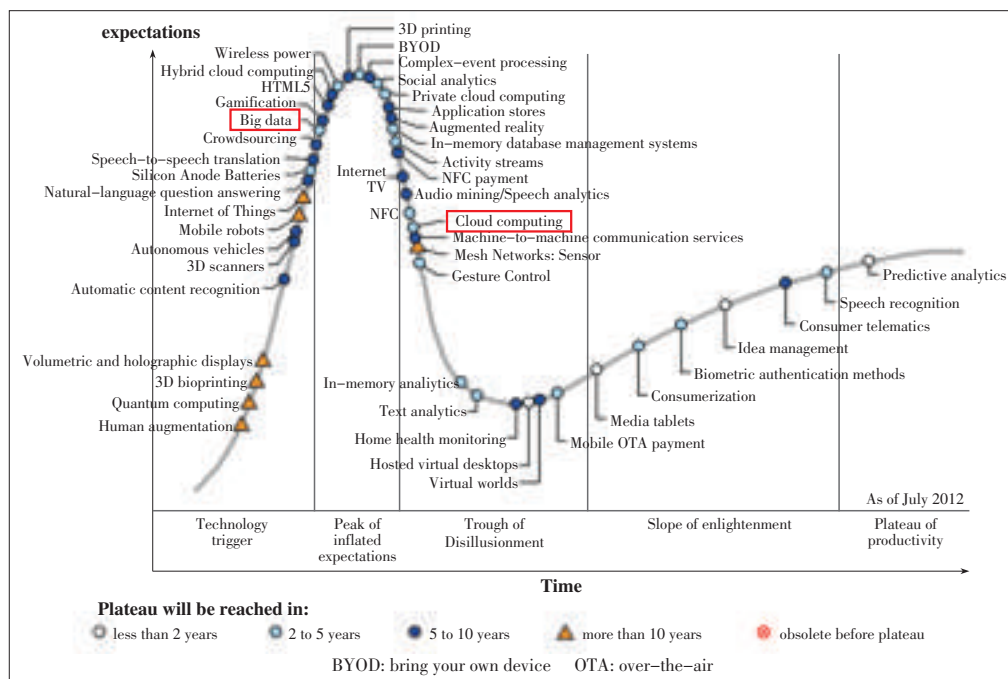
Because of the decreasing cost of generating data, the world-



▲ Figure 1. A big-data scenario.

Big-Data Analytics: Challenges, Key Technologies and Prospects

Shengmei Luo, Zhikun Wang, and Zhiping Wang

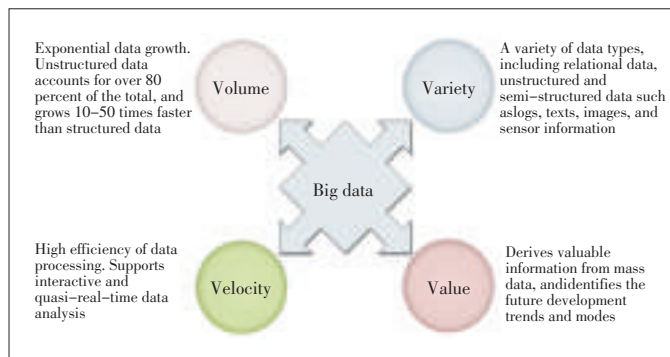


▲ Figure 2. Gartner 2012 hype cycle.

wide volume of data has grown sharply. Ubiquitous mobile devices and wireless sensors are generating data every minute, and bulk data exchanges are occurring every second between billions of the internet services. Scientific applications, video surveillance, medical records, enterprise operational data, discrete manufacturing, and e-commerce are all sources of big data. In 2011, the International Data Corporation claimed that “The world’s information is doubling every two years” [4]. In that year, the world generated a staggering 1.8 ZB (10^{21} B) of data, an increase of 0.6 ZB year-on-year. By 2020, the world will generate up to 35 ZB of data, and this poses significant challenges for storage (Fig. 4 [4]).

2.2 Variety

Data may be structured, unstructured or semi-structured, and all three types of data are frequently and extensively interchanged. Structured data only accounts for 20% of the big data



▲ Figure 3. The “four Vs” of big data.

stored in databases. Data from the internet—including data created by users, data exchanged in social networks, and data from physical sensing devices and the internet of things—is dynamic and unstructured. Unstructured data accounts for 80% of big data stored in databases.

2.3 Velocity

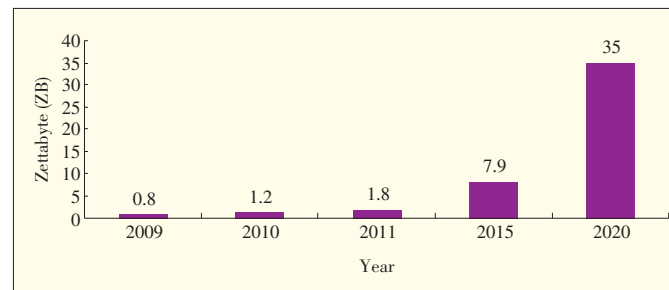
Velocity refers to the high requirements on real-time processing and quasi-real-time analysis of big data. With traditional data warehousing and business intelligence, real-time requirements are lower. In the big-data era, the value of data decreases rapidly over time, so data needs to be exploited as soon as possible.

2.4 Value

Structured data of an enterprise has traditionally been used for statistics and historical analysis. However, big data involves extracting valuable information from mass data to predict future trends and make decisions. In addition, big data has low value density. For example, continuous video surveillance may produce a great deal of data, but only a few seconds of the video may actually be useful.

3 Challenges with Big-Data Analytics

Traditionally, data originates from a single source and has relatively low volume. Storing, managing, and analyzing such data does not present great challenges, and most processing is done through relational databases and data warehouses. In a big-data environment, the volume of data is so great that traditional information processing systems cannot cope with storage, mining, and analysis. Traditional business intelligence software lacks effective tools and methods for processing and analyzing unstructured data. Here, we discuss some problems



▲ Figure 4. Forecast of the volume of global data.

in big-data analytics.

3.1 Extensive Data Sources and Poor Data Quality

Big-data scenarios are characterized by heterogeneous data sources, such as transaction records, text, images, and videos. Such data is described in different ways, and the data input rate may reach up to hundreds of megabits or even gigabits per second. Traditional methods for describing structured data are not suitable for describing big data. In addition, big data is easily affected by noise and may be lost or inconsistent. Filtering and integrating incomplete, noisy, and inconsistent data is a prerequisite for efficiently storing and processing big data.

3.2 Highly Efficient Storage of Big Data

The way that big data is stored affects not only cost but analysis and processing efficiency. Big data is often measured in petabytes or even exabytes and cannot be handled by enterprise storage area networks or network-attached storage. To meet service and analysis requirements in the big-data era, reliable, high-performance, high-availability, low-cost storage solutions need to be developed. Because big data comes from various sources, the same data may exist in the system and cause absolute redundancy. Detecting and eliminating redundancy increases storage space and is a fundamental requirement for big-data storage platforms.

3.3 Efficiently Processing Unstructured and Semi-Structured Data

Enterprise data is mainly processed in relational databases and data warehouses; however, such databases and warehouses are unsatisfactory for processing unstructured or semi-structured big data. With big data, read/write operations need to be highly concurrent for a mass number of users; storage of and access to big data needs to be highly efficient; and the system must be highly scalable. As the size of datasets increases, algorithms may become inefficient, and the atomicity, consistency, isolation, durability (ACID) features of relational databases are resource intensive. The CAP theorem states that it is impossible for a distributed computer system to simultaneously guarantee consistency, availability, and partition tolerance [5]. Because consistency is required in parallel relational databases, these databases are not highly scalable or available. High system scalability is the most important requirement for big-data analytics, and highly extensible data analysis techniques must be developed.

3.4 Mass Data Mining

With an increase in the size of data sets, more and more machine learning algorithms and data-mining algorithms for big data have emerged. Research has shown that for larger data sets, machine learning is more accurate and there is less difference between machine-learning algorithms. However, large data sets are problematic for traditional machine-learning as

well as data-mining algorithms. In fact, most traditional data-mining algorithms are rendered invalid by big data sets. Machine-learning and data-mining algorithms with difficulty levels of $O(n)$, $O(n \log n)$, $O(n^2)$, and $O(n^3)$ can be used for small data sets. However, when the size of a data set reaches petabytes, serial algorithms may fail to compute within an acceptable timeframe. Effective machine-learning and data-mining algorithms need to be developed for big data sets.

4 Big-Data Analytics

4.1 Architecture

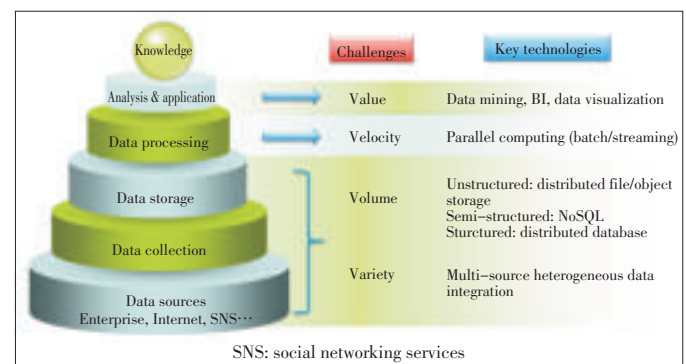
A typical big-data processing system includes collection and preprocessing, storage, analysis, mining, and value application. Fig. 5 shows the architecture of a big-data system. In the data-source layer, data comes from enterprises, industry, the internet, and the internet of things. In the data-collection layer, the collected data is preprocessed. This preprocessing includes data cleanup and heterogeneous data processing. In the data-storage layer, structured, unstructured, and semi-structured data is stored and managed. In the data-processing layer, data is analyzed and mined so that users can analyze services, such as common telecommunications and internet services, on the platform.

4.2 Big-Data Technology

4.2.1 Collection and Preprocessing

Converting the format of mass amounts of data is expensive, and adds to the difficulty of data collection. Traditional data-collection tools have become obsolete, and most internet companies have their own big-data collection systems. Examples of such systems are Apache Chukwa [6], Facebook Scribe [7], Cloudera Flume [8], and LinkedIn Kafka [9].

Cleaning and extracting technologies are used to clean out damaged, redundant, and useless mass data in networks and extract quality data for analysis. Hadoop is used to expedite data cleaning, conversion, and loading and to improve preprocessing of parallel data [10]. Big-data collection and preprocessing



▲ Figure 5. Big-data architecture.

Big-Data Analytics: Challenges, Key Technologies and Prospects

Shengmei Luo, Zhikun Wang, and Zhiping Wang

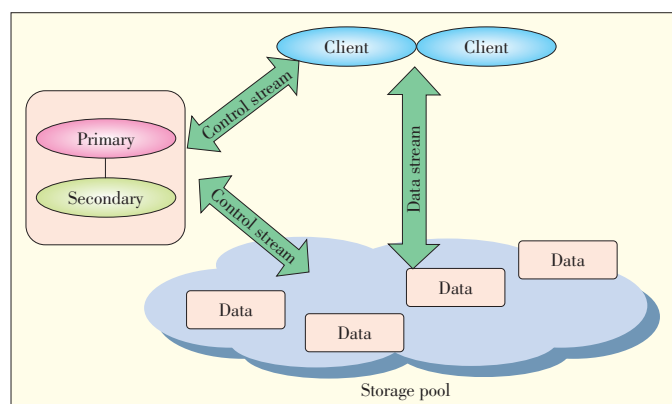
technologies are also designed to optimize the quality of multi-source and multimode data. These technologies perform the computations necessary for the integration of multimode data. They transform high-quality data into information, control the quality of information derived from different sources, and lay the foundation for data analysis.

4.2.2 Storage

Unstructured and semi-structured data account for 80% of stored big data. At present, big-data storage is based on inexpensive X86 server cluster systems that are highly scalable using certain methods.

Unstructured data is stored in distributed file systems. Unlike the traditional Network File System (NFS), a distributed file system separates control streams from data streams in order to improve scalability [11]. In a distributed file system, the metadata servers manage all the metadata and cluster information of data-storage servers. For example, the capacities of Google File System (GFS) [12], Hadoop Distributed File System (HDFS) [10], Lustre [13], and Ceph [14] can be up to 10 PB or even 100 PB. These storage systems provide file access semantics through POSIX interfaces. If a metadata server fails in a storage system of this kind, the whole file system fails to provide services to users. The processing and storage capacity of a single-node metadata server is also quite limited. As the amount of data traffic in the system increases, the processing capability of metadata servers becomes a bottleneck for system scalability.

Metadata servers can therefore be used in active/standby mode (Fig. 6). In normal conditions, the active metadata server processes all requests, manages the whole distributed file system, and regularly sends data to the standby metadata server for synchronization. If the active server fails, the standby server takes over all activities without interrupting services. Data that has not been sent to the standby server may be lost. Using this mode, single-point failures can be solved, but the system cannot be scaled. Therefore, a highly scalable metadata server cluster is critical in a distributed file system. The design of



▲ Figure 6. Architecture of a metadata server cluster in active/standby mode.

such a cluster includes static subtree partitioning, static hashing, dynamic subtree partitioning, and dynamic hashing [15].

Big data includes semi-structured data, which is more structured than plain-text data but has more flexible models than data in relational databases. Semi-structured data does not require strict database transactions. It mainly involves simple single-table query, and in some cases, it has low consistency requirements. Therefore, database transaction management is a burden in heavily loaded databases. The NoSQL database is, broadly speaking, a non-relational database in which the link between relational database and ACID theory is broken. NoSQL data storage does not require a fixed-table structure or connection operations, and this provides significant advantages in terms of access to big data.

In NoSQL databases, relational data storage models are discarded in favor of a schema-free principle that supports distributed horizontal expansion and big data. There are many NoSQL database products and open-source projects, including Dynamo [16], BigTable [17], Cassandra [18], and MongoDB [19]. Fig. 7 shows four types of NoSQL databases based on the data model. From left to right, key-value, column-oriented, graph-oriented, and document-oriented databases are in ascending order of complexity and in descending order of scalability (Fig. 7). A database can be selected according to the application scenario.

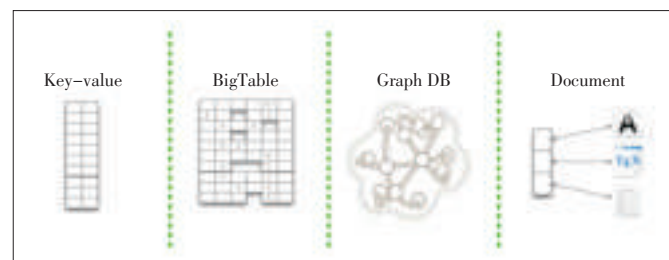
NoSQL systems are not yet mature: Most are open-source projects that have little commercial support. NoSQL systems lack unified application program interfaces (APIs) and do not support SQL, which is costly in terms of learning and application migration.

4.2.3 Processing

Processing large data sets with mixed loads is complex, and certain processing requirements should be met (Table 1).

The data warehouse is the primary means of processing the traditional structured enterprise data. In the big-data era, the data warehouse has changed in terms of

- distributed architecture. A typical data warehouse includes complicated data processing and comprehensive analysis; therefore, the system should be capable of high I/O processing, and the storage system should provide sufficient I/O bandwidth. Most data warehouses use massive parallel processing (MPP) architecture for scalability and to improve ac-



▲ Figure 7. NoSQL database models.

Big-Data Analytics: Challenges, Key Technologies and Prospects

Shengmei Luo, Zhikun Wang, and Zhiping Wang

▼ Table 1. Big data processing requirements

Feature	Description
High scalability	Scale-out scalability, support for large-scale parallel data processing
High performance	Rapid response to mass-data query and analysis requirements
Low cost	Based on universal hardware servers, high price/performance ratio
Fault tolerance	Independent of hardware reliability, some (rather than all) operations are performed in the case of a fault
Easy to use and open interfaces	Compatible with traditional SQL interfaces, support for query and multidimensional data analysis
Downward compatibility	Support for traditional BI tools

cess efficiency.

- storage model. There are two methods for storing physical data in databases: row and column. Row storage is used in traditional databases where OLTP applications read and write in rows and have low data traffic. Column storage is used in data warehouses where most OLAP applications do not need to select all columns. This method of storage can reduce unnecessary I/O consumption and makes data compression easier. A high data compression ratio can be achieved because the data in a column is the same type.
- hardware platform. Traditional databases are mainly found on midrange computers. If data traffic increases sharply, the cost of upgrading hardware can increase significantly. In the big-data era, parallel data warehouses are based on universal X86 servers.

Big unstructured data is mainly processed using a distributed computing architecture (Fig. 8). This architecture, located at the distributed storage layer, encompasses parallel computing, task scheduling, fault tolerance, data distribution, and load balancing. It provides computation services for the upper layer. The language layer encapsulates service interfaces and provides SQL-like programming interfaces. The SQL-like languages vary with the computing architectures.

There are three types of distributed computing based on different computation models:

- 1) MapReduce. This model can be expressed as (input key, input value) \rightarrow (output key, output value). The output key/value pairs are processed, and the input key/value pairs are generated using the MAP or Reduce functions. This model has a simple logic and is widely used [20].
- 2) Bulk Synchronous Parallel (BSP) model [21]. This is an iterative computation model that is similar to a simple computation model; however, the difference lies in communication. In this model, all nodes are synchronized after each round of computation. This is suitable for iterative scenarios. Google Pregel, for example, is based on BSP architecture [22].
- 3) Directed Acyclic Graph (DAG) model. This model uses DAG to describe complicated computing processes and relationships between them. Microsoft uses this computing model in its Dryad project [23].

Real-time stream processing platforms such as Yahoo S4

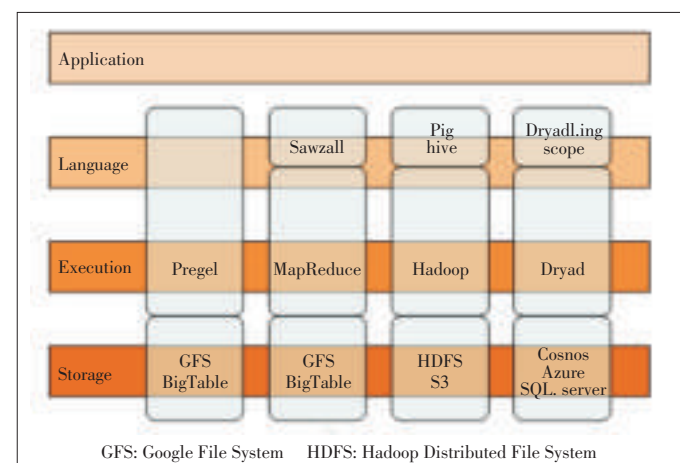
[24] and Twitter Storm [25] have been created to meet big-data processing requirements. These platforms process a data stream in real time in the memory and do not retain much data.

4.2.4 Mining

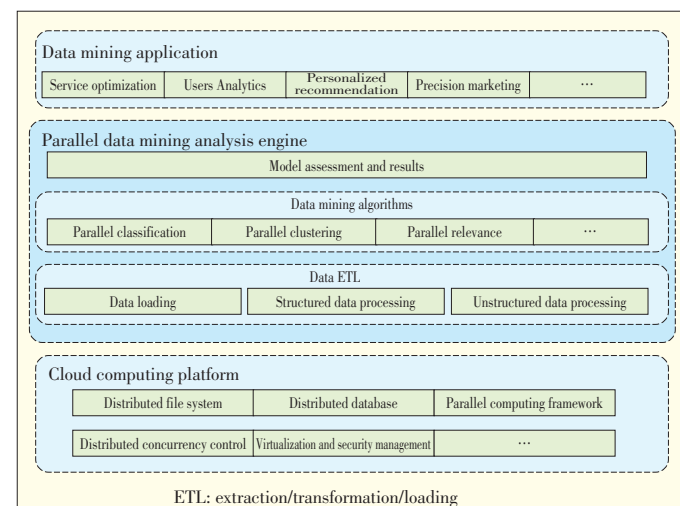
Big-data mining technology is used to effectively extract value from big data. Such technology includes parallel data mining, search engine, recommendation engine, and social network analysis.

Parallel data mining greatly increases the speed of big-data mining by implementing data-mining algorithms in parallel. Hadoop [10] and HDFS aid in this implementation. The data-mining algorithms include parallel classification algorithm and parallel clustering algorithm. Fig. 9 shows a parallel data-mining system architecture based on cloud computing platforms.

A search engine is an information retrieval system that collects data from various services or application systems. It



▲ Figure 8. Service system of a distributed computing architecture.



▲ Figure 9. Parallel data-mining system architecture based on cloud platform.

Big-Data Analytics: Challenges, Key Technologies and Prospects

Shengmei Luo, Zhikun Wang, and Zhiping Wang

stores, processes, and reorganizes the data and provides users with query functions and results. The search engine is an important tool for data management after the big data has been obtained by a storage system. The search engine allows users to input simple queries and obtain useful collections of information. Fig. 10 shows the architecture of a universal search engine comprising many technology modules. The key technologies include the Web crawler, document understanding, document indexing, relevance computing, and user understanding.

The recommendation engine helps users to obtain personalized services or content from a mass of information. It is the driver of the transition from a search era to a discovery era. The main challenges for recommendation systems are cold start, scarcity, and scalability. The quality of recommendations depends not only on the models and algorithms but also on non-technical factors, such as product form and service mode.

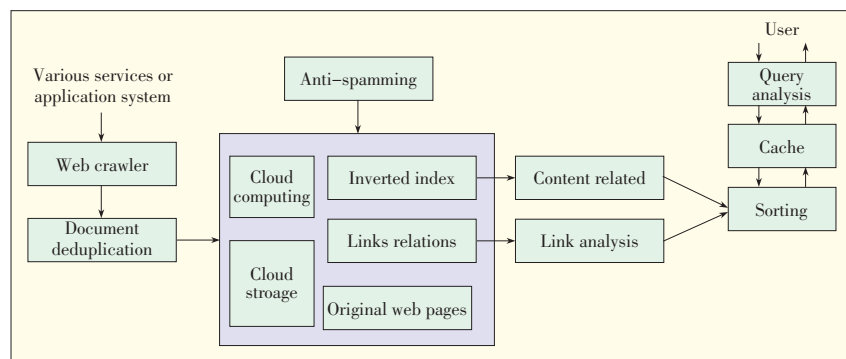
Fig. 11 shows the typical architecture of a recommendation system, including the user operation database, data-mining engine, data warehouse, recommendation model library, recommendation engine, and user interaction agent. Together, these subsystems, databases, and basic operations provide users with personalized services. The main algorithms in a recommendation engine are content-based filtering algorithm, collaborative-filtering algorithm, and relevance-analysis algorithm.

Social network analysis is a new concept for analyzing new problems. It is based on the relationships between members and provides the methods and tools for interactive data mining. It displays crowd-sourced intelligence and ideas and is important for social filtering, marketing, recommendations, and searching. Social network analysis involves user relationship, topic, interest, identification, influence, and emotion analysis. It also involves community discovery.

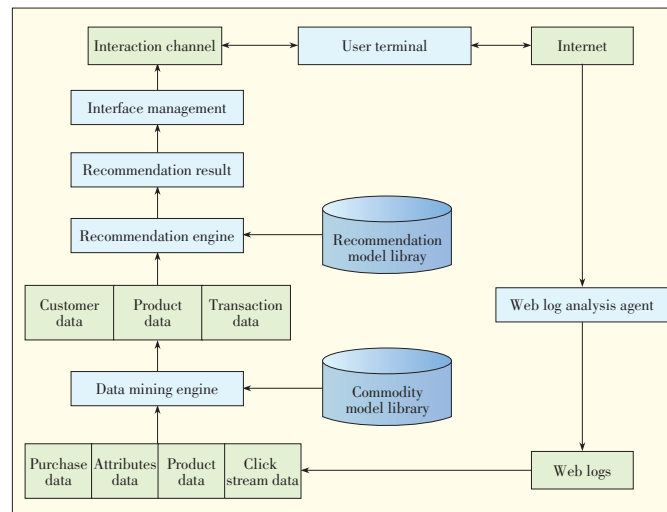
5 Prospects of Big-Data Analytics

5.1 Representing and Understanding Multisource and Multimode Data

Big data comes from different sources and comes in different formats. During preprocessing, different attributes of the data



▲ Figure 10. Search engine architecture.



▲ Figure 11. Intelligent recommendation system architecture.

need to be determined for multidimensional description and also to improve explainability and flexibility so that analysis requirements are met. In the future, research needs to be done on semantic analysis, network data context resolution, and visual media analysis and learning. Theories and methods of intrinsic data representation and mapping from high-dimensional data space to low-dimensional manifold need to be developed to resolve and integrate heterogeneous network data.

5.2 Low-Cost, Highly Efficient Data Storage Technology

The storage of big data affects not only the efficiency of data analysis and processing but also cost. With the emergence of big data, research into parallel data compression, distributed data de-duplication, thin provisioning, automated tiered storage (ATS), energy-saving, and freeing up of storage space needs to be done. This will reduce the storage pressure on servers, increase transmission efficiency, improve data analysis, and reduce costs.

5.3 New Parallel Computing Model and Big-Data Architecture

Big data seriously affects computation and storage. Big data is so large that the existing computation methods and algorithms cannot compute within an acceptable timeframe. Therefore, new processing methods are required.

With unstructured data processing and large-scale parallel processing, non-relational data analysis models have greatly improved search and analysis of mass internet data. Such models include MapReduce and have become commonplace for big-data analysis [20]. However, MapReduce still has many performance problems. To make parallel computation more efficient for big data, more effective and useful pro-

Big-Data Analytics: Challenges, Key Technologies and Prospects

Shengmei Luo, Zhikun Wang, and Zhiping Wang

gramming models and architectures need to be developed. Hybrid programming models and architecture have already been developed. These include MapReduce [20], BSP [21], Message Passing Interface (MPI) [26], Compute Unified Device Architecture (CUDA) [27], and shared-memory parallel programming and computation models [28].

5.4 Multidimensional and Multimode Data Mining and Knowledge Discovery

Big data includes multidimensional and multimode data. Attributes and dimensions can be reduced in big-data mining to classify data nodes and measure data relevance and integration mechanisms. Data-mining algorithms can meet the special requirements of big data, expand the scope of data-mining applications, and satisfy the requirements of users at different data-mining terminals. Application-oriented data-mining algorithms should be developed according to the application environment and an understanding of semantics. With natural language processing and machine learning, knowledge derived from big data can be more practically used in commerce and science.

6 Summary

With the rise of social networks and the widespread application of cloud computing, mobile internet, and the internet of things, big data has become the focus of attention. There are various types of big data, and processing methods are becoming more complex. This has created many challenges. Traditional data-processing architectures such as relational databases and data warehouses struggle to process big data. Systems tailored to big data have distributed storage, MapReduce parallel computing, and data mining. However, these are in a fledgling state. To improve big-data analysis, research on representation, measurement, and semantics for big data must be undertaken. The cost of storing big data should also be reduced, and flexible, highly efficient big-data computing architectures and data-mining algorithms should be developed.

References

- [1] Hung LeHong and J. Fenn. Gartner Report. "Hype Cycle for Emerging Technologies", July 2012. [Online]. Available: <http://www.gartner.com/newsroom/id/2124315>
- [2] Big Data. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Big_data
- [3] D. Vesset and B. Woo et al., "Worldwide Big Data Technology and Services 2012–2015 Forecast," IDC Report 233485, March 2012. [Online]. Available: <http://www.idc.com/getdoc.jsp?containerId=233485>
- [4] J. Gantz and D. Reinsel, "Extracting Value from Chaos," IDC iView Report sponsored by EMC Corporation, Jun. 2011. [Online]. Available: <http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf>
- [5] E. A. Brewer. "Towards robust distributed systems," in *Proc. 19th Annual ACM Symp. on Principles of Dist. Comput.*, Portland, OR, 2000, pp. 7–10.
- [6] Chukwa. [Online]. Available: <http://incubator.apache.org/chukwa/>
- [7] Scribe. [Online]. Available: <http://github.com/facebook/scribe>
- [8] Apache Flume. [Online]. Available: <http://flume.apache.org/>
- [9] Apache Kafka. [Online]. Available: <http://kafka.apache.org/>
- [10] Apache Hadoop. [Online]. Available: <http://hadoop.apache.org/>
- [11] R. Sandberg, D. Goldberg, S. Kleiman, et al. "Design and implementation of the Sun network file system," in *Proc. of Summer USENIX Conference*, 1985, pp. 119–130.
- [12] S. Ghemawat, H. Gobioff, S.-T. Leung, "The Google file system," in *19th Symp. on Operating Syst. Principles*, Lake George, NY, 2003, pp. 29–43.
- [13] P. J. Braam, "The Lustre storage architecture," white paper, Cluster File Systems Inc., Oct. 2003, p. 23.
- [14] S. A. Weil, S. A. Brandt S A, E. L. Miller et al., "Ceph: A scalable, high-performance distributed file system," in *Proc. 7th Symp. on Operating Syst. Design and Implementation*, USENIX Association, Seattle, WA, Nov. 2006, pp. 307–320.
- [15] Jin Xiong, Yiming Hu, Guojie Li, Rongfeng Tang, Zhihua Fan, "Metadata Distribution and Consistency Techniques for Large-scale Cluster File Systems," in *IEEE Trans. Parallel and Dist. Syst.*, vol. 22, pp. 803–816, May 2011.
- [16] G. DeCandia, D. Hastorun, M. Jampani et al., "Dynamo: Amazon's highly available key-value store," in *Proc. 21st ACM SIGOPS Symp. on Operating Syst. Principles*, Stevenson, WA, Oct. 2007, pp. 205–220.
- [17] F. Chang, J. Dean, S. Ghemawat et al. "Bigtable: A distributed structured data storage system," *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*, Seattle, WA, pp. 305–314.
- [18] Apache Cassandra. [Online]. Available: <http://cassandra.apache.org/>
- [19] MongoDB. [Online]. Available: <http://www.mongodb.org/>
- [20] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Proc. 6th Symp. on Operating Syst. Design and Implementation (OSDI '04)*, San Francisco, CA, Dec. 2004, pp. 137–149.
- [21] L. G. Valiant, "Bulk-synchronous parallel computers," In M. Reeve, ed., *Parallel Processing and Artificial Intelligence*, New York: Wiley & Sons, 1989, pp. 15–22.
- [22] G. Malewicz, M. H. Austern, A. J. Bik et al., "Pregel: A system for large-scale graph processing," In *Proc. 2010 ACM SIGMOD Int. Conf. on Management of Data*, Indianapolis, IA, Jun. 2010, pp. 135–146.
- [23] M. Isard, M. Budiu, Y. Yu et al. "Dryad: Distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 59–72, Jun. 2007.
- [24] L. Neumeyer, B. Robbins, A. Nair et al. "S4: Distributed stream computing platform," in *Proc. 2010 IEEE Int. Conf. on Data Mining Workshops*, Sydney, Australia, Dec. 2010, pp. 170–177.
- [25] Storm. [Online]. Available: <https://github.com/nathanmarz/storm>
- [26] L. Dagum and R. Menon, "OpenMP: An industry standard API for shared-memory programming," in *IEEE Trans. Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998.
- [27] Nvidia, "Compute unified device architecture programming guide," Reference Manual, Version 2, Jun. 2008. [Online]. Available: http://www.cs.ucla.edu/~palsberg/course/cs239/papers/CudaReferenceManual_2.0.pdf
- [28] C. Amza, A. L. Cox, S. Dwarkadas et al. "Treadmarks: Shared Memory Computing on Networks of Workstations," in *IEEE Computer*, vol. 29, no. 2, 1996, pp. 18–28.

Manuscript received: April 18, 2013

Biographies

Shengmei Luo (luo.shengmei@zte.com.cn) is the chief software architect at ZTE Corporation and a member of the China Cloud Computing Committee. His research interests include big data and product planning for cloud computing.

Zhikun Wang (wang.zhikun@zte.com.cn) is a senior pre-research engineer at the Cloud Computing and IT Institute of ZTE Corporation. His research interests include cloud computing and big data technologies.

Zhiping Wang (wang.zhiping@zte.com.cn) received his doctoral degree in information and signal processing from Southeast University, China. He is responsible for cloud computing pre-research and big data at ZTE Corporation. He has led major projects from the National Development and Reform Commission of China, the Ministry of Science and Technology of China, and the Ministry of Industry and Information Technology of China. He has applied for and released 11 patents and is a member of China Computer Industry Association's CCF task force on big data.

Data Security and Privacy in Cloud Storage

Xinhua Dong, Ruixuan Li, Wanwan Zhou, Dongjie Liao, and Shuoyi Zhao

DOI: 10.3969/j.issn.1673-5188.2013.02.003

<http://www.cnki.net/kcms/detail/34.1294.TN.20130627.1101.001.html>, published online June 27, 2013

Data Security and Privacy in Cloud Storage

Xinhua Dong, Ruixuan Li, Wanwan Zhou, Dongjie Liao, and Shuoyi Zhao

(School of Computer Science and Technology, HuaZhong University of Science and Technology, Wuhan 430074, China)

Abstract

In this paper, we survey data security and privacy problems created by cloud storage applications and propose a cloud storage security architecture. We discuss state-of-the-art techniques for ensuring the privacy and security of data stored in the cloud. We discuss policies for access control and data integrity, availability, and privacy. We also discuss several key solutions proposed in current literature and point out future research directions.

Keywords

cloud storage; cloud computing; data security; privacy-preserving

1 Introduction

With the recent development of information technology, various types of cloud storage platforms have appeared. These platforms are often convenient, scalable, and cost-effective, so cloud computing services are widely used. Amazon's EC2/S3, Google's MapReduce/AppEngine, Microsoft's Azure, IBM's Blue Cloud and Salesforce's CRM are well-known cloud service platforms. In China, the cloud platforms of Sinochem Group and Wuxi and Dongying municipal governments have appeared. At the same time, the national strategy for cloud storage and utility computing has been developing. Utility computing service models based on cloud storage have many new characteristics. New models based on cloud storage use a variety of techniques designed to tightly manage resources and provide users with flexible services. These new service models are likely to have a knock-on effect and cause significant changes within the industry. They will give rise to many new security issues that need to be addressed. At present, cloud storage is only really applied in scenarios where a high level of security is not required. Privacy and security are significant obstacles in the development of utility computing based on cloud storage. Security vulnerabilities of cloud service providers such as Amazon, Google, and Microsoft are widely publicized, and much attention has been drawn to these vulnerabilities. In [1], cloud security and privacy issues are detailed. The Cloud

Security Alliance (CSA) has made recommendations for solving problems in cloud computing applications. In [2], the European Network and Information Security Agency (ENISA) detailed the risks to data and benefits of data security in cloud computing applications. In [3], EMC Corporation's RSA Security Division analyzed the most basic security issues in the cloud infrastructure. Domestic Chinese counterparts have also widely discussed the issue of cloud security [4]. Satisfactory solutions to cloud security and privacy problems will be a strong driving force for the overall development of cloud storage services. Solving these problems is also theoretically and practically important to vigorously promote the national digital infrastructure and national information security.

2 Cloud Storage: Concepts, Applications and Security

2.1 Concepts

Cloud computing involves the development of distributed processing, parallel processing, and grid computing. With cloud computing, huge computing programs are automatically split into smaller subroutines via the network. Processing and analysis is referred to multiple servers in a large system, and results are returned to the user. The network service provider can process massive amounts of information in seconds—a capability that is equal to that of powerful supercomputer networks.

Cloud storage is an extension of cloud computing and is one of a large variety of storage devices found in networks. Through the use of application software and clustering, grid technology, distributed file system, or other functions, cloud

This work is supported by National Natural Science Foundation of China under grants 61173170 and 60873225, National High Technology Research and Development Program of China under grant 2007AA01Z403, and Innovation Fund of Huazhong University of Science and Technology under grants 2013QN120, 2012TS052 and 2012TS053.

storage works with other storage devices to provide a service access function and a whole data-storage system. When the cloud processing core requires large-scale data storage and management, a large number of storage devices need to be configured, and a cloud storage system needs to be set up. Thus, cloud storage is a cloud computing system in which data storage and management is at the core.

Cloud storage systems differ from traditional storage systems in that they are designed for multiple types of online storage services. Traditional storage systems are designed for specific applications, such as high-performance computing or transaction processing. In terms of performance indexing, cloud storage services are primarily indexed according to data security, reliability, and efficiency. Cloud storage systems are suitable for large-scale applications, a wide range of services, and complex network environments. In terms of data management, cloud storage systems provide traditional file access similar to POSIX, but cloud storage also supports mass data management and public services.

2.2 Applications

Cloud storage is effective, flexible, low-cost, and easy to manage. Cloud storage can be used in either enterprise applications or personal applications, depending on the type of service and user orientation.

In enterprises, cloud storage is used for storage space leasing, remote data backup, disaster recovery, and video surveillance. With an IDC data center, operators can lease storage space to enterprises and institutions that do not wish to purchase mass-storage devices. A high-performance, high-capacity cloud storage system and remote data backup software also allows an operator to help enterprises and institutions build their own remote-backup and disaster-recovery systems as well as remote real-time video surveillance and playback systems.

Cloud storage applications for individuals include network disks, online document editing, and online games. Network disks are used to upload and download files when a user is storing and backing up personal data over the internet. With online document editing, a user can simply access a web page, such as Google Docs, to edit, manage, and transmit documentation. Cloud computing and storage can also be used to build a huge game server cluster so that all players are managed as a game server group, and gaming becomes more exciting.

2.3 Security

Because cloud storage is large-scale, complex, and dynamic, it creates many new security and privacy problems. These problems include unauthorized access to data, threats to confidentiality, threats to data integrity, unavailability of data, and lack of privacy.

To prevent unauthorized access to the storage system, a provider needs to confirm the user's identity and verify whether

that user has the permission to access resources or perform certain operations. The user submits an access request to the cloud storage provider through storage access interfaces.

When using a cloud storage service, a user uploads their local data to the storage server and downloads the data when they need it. In this process, data passes through a public cloud, private cloud, and internet transmission line. Data may be stolen or altered when stored in the server. In this case, the confidentiality of the user's data is compromised.

In a both traditional and cloud storage environments, data integrity is remotely verified to prevent data tampering and counterfeiting. However, in a cloud environment, users do not have absolute control over their data, and there is a greater need to verify the integrity of the data. When the user updates their data in the cloud, the server must promptly update the data. Therefore, real-time verification of integrity is essential.

Preventing data from being lost and ensuring the sustained, effective use of data are important responsibilities of a cloud storage provider. This requires the provision of strategies that ensure data availability, backup, and recovery.

Users often disclose personal information, such as credit card numbers or user name, when purchasing storage services. These need to be protected. A user's digital identity, certificates, access, and operation records also need to be protected. A storage service provider needs mechanisms to guarantee the privacy of user information.

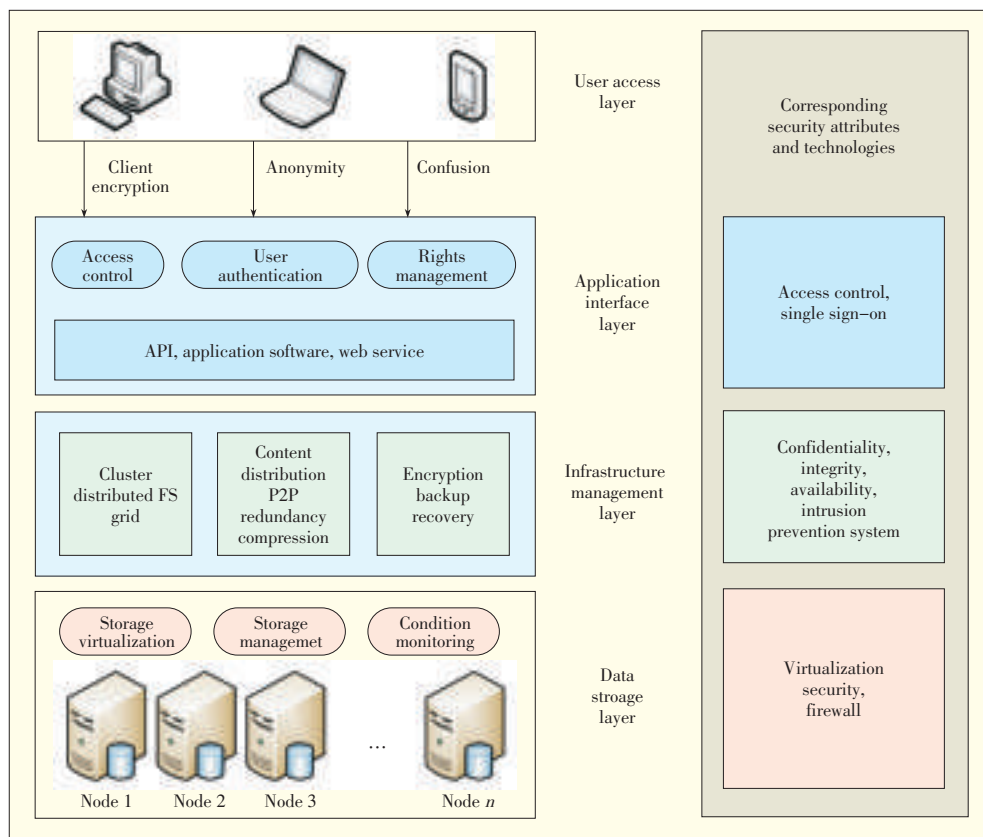
3 Architecture for Ensuring Security and Privacy in a Cloud Storage System

In [5], a secure cloud storage architecture is proposed. This architecture comprises business application layer, application interface layer, platform software layer, and infrastructure layer (Fig. 1). These layers provide information service management, statistical analysis, and a variety of safety measures.

In the access layer, an authorized user can log on to the cloud storage system via a standard public application interface to use cloud storage services. Different cloud storage providers have different types of access methods. The application interface layer is the SaaS layer of cloud computing services. Different cloud storage providers can develop different application interfaces and provide different application services based on the type of business. However, security problems may arise in access control and single sign on (SSO). The infrastructure management layer is the core of the cloud. It includes clusters, distributed file systems, and grid computing and allows cooperation between storage devices in the cloud. Content distribution systems and encryption of stored data ensures that data in the cloud is not accessed by unauthorized users. Various data-backup and disaster-recovery measures ensure that data stored in the cloud is not lost. In this layer, there may be problems with data confidentiality, integrity, availability, and intrusion. The data-storage layer is the fundamental part of cloud

Data Security and Privacy in Cloud Storage

Xinhua Dong, Ruixuan Li, Wanwan Zhou, Dongjie Liao, and Shuoyi Zhao



▲ Figure 1. Architecture for ensuring security and privacy in a cloud storage system.

storage. In the data-storage layer, huge amounts of data are managed in a unified way. The data-storage is also used for virtual management of storage, monitoring hardware, and fixing faults. In this layer, security problems arise in virtualization and in the firewall.

For a user uploading their data to a cloud storage system, privacy mainly depends on whether the cloud storage provider supports encryption, differential privacy protection, compulsory destruction of agreement, or data privacy management. However, cloud storage providers may also snoop on a user's data or analyze the privacy preferences of users. To fully protect user privacy, data needs to be encrypted, anonymized, or scrambled before being uploaded to the cloud.

4 Data Security in Cloud Storage

Existing technologies for controlling access to data, encrypting data, and ensuring data integrity and availability have been transplanted to cloud storage. Experts have proposed different models for ensuring data security in different cloud systems. The key technologies in these models relate to access control, data confidentiality, data integrity authentication, and data availability.

To control access to encrypted data, the owner of the data maintains encryption keys and manually sends them to users

who want access to the data. This has become a bottleneck in the cloud storage environment. In [6], a new cryptographic access control scheme, called attribute-based access control for cloud storage (AB-ACCS), was proposed. Each user's private key is labeled with a set of attributes, and data is encrypted with an attribute condition so that the user can only decrypt the data if their attributes satisfy the data's condition. In [7], the authors consider the complexity of fine-grained access control for a large number of users in the cloud and propose a secure and efficient revocation scheme based on a modified ciphertext-policy attribute-based encryption (CP-ABE) algorithm. This algorithm is used to establish fine-grained access control in which users are revoked according to Shamir's theory of secret sharing. With a single sign-on (SSO), any authorized user can log in to the cloud storage system through a standard common appli-

cation interface.

In a cloud storage environment, internal administrator's privileged mode is potentially a serious threat to user privacy data. To guarantee data privacy when administrator privileged mode is used, a variety of protection methods have been proposed. Attribute-based encryption (ABE) includes key-policy attribute-based encryption (KP-ABE) [8] as well as CP-ABE [9]. In ABE, decryption rules are contained in the encryption algorithm, and frequent distribution of keys in the access-control-based ciphertext is unnecessary. When the access control policy is changed, the data owner encrypts the data again. In [10], a method based on proxy re-encryption is proposed. A semi-trusted agent with proxy key can re-encrypt a ciphertext; however, the agent cannot gain the corresponding plaintext or compute the decryption key of either party in the authorization process [11]. In [12], a fully homomorphic encryption (FHE) mechanism is proposed. FHE permits a specific algebraic operation based on ciphertext, and the result is still encrypted. That is to say, retrieval and comparison of the encrypted data ends with the correct results, but the data is not decrypted throughout the whole process. The FHE scheme requires a huge amount of computation and is not always easy to implement with existing technology.

In [13], proofs of retrievability (POR) are proposed. A POR scheme allows an archive or backup service (prover) to pro-

duce a concise proof that a user (verifier) can retrieve a target file. The POR model verifies data integrity without the user having to download files themselves. A drawback of our proposed POR scheme is that the target file requires processing prior to being stored with the prover. This step creates computational overhead and increases the storage requirements on the prover. In addition, the POR scheme is based on static files, so the scope of its application is smaller. In [14], a flexible distributed scheme based on POR is proposed. This scheme assumes that operations are dynamic and data integrity is publicly verified. With Merkle Hash Tree (MHT), the new scheme supports secure, efficient update, deleting, and appending of data blocks. The improved scheme supports public data-integrity verification and authorized third-party integrity verification.

A variety of data backup and disaster recovery measures guarantee that data stored in cloud is not be lost. These measures ensure that cloud storage is secure and stable. To counter theft of legacy data, the United States Department of Defense proposes reset and special processing [15]. In [16], a new scheme called Safe Vanish is proposed. This scheme prevents hopping attacks by extending the length of key shares and significantly increasing the cost of mounting an attack. The authors of [16] also propose using the public key cryptosystem to protect against sniffing.

Access control technology is more suited to a fine-grained cloud storage environment, and this creates large overhead. Data confidentiality is ensured by a variety of encryption methods, and confidentiality is considered in relation to access control. A data integrity verification scheme eliminates user concern throughout the data storage process. Researchers have paid attention to availability but are now also paying attention to security throughout the storage process. Security mechanisms create efficiency problems, and the trade-off between security and efficiency need to be further researched.

5 Ensuring Privacy in Cloud Storage

In a cloud storage system, privacy can be lost because of data outsourcing and service leasing. User data is stored in the cloud environment and is managed by the cloud storage provider. The security of this data depends on the level of technology used by the service providers.

In [17], a system called Arivat is proposed. This system is based on MapReduce and is designed to provide strong security and privacy for distributed computations on sensitive data. Mandatory access control and differential privacy are integrated in a novel way. In [18], the author proposes privacy management across the whole data lifecycle and uses a mandatory data destruction protocol to control user data. Dissolver is a prototype system based on Xen virtual machine monitor and CHAOS system [18]. It ensures that the user's text data only exists in a private operating space and the user's key only exists in

the memory space of the virtual machine monitor. Data in the memory and the user's key are destroyed at a time specified by the user.

The system ensures the server-side privacy of user data throughout the data's lifecycle. In [19], a cloud storage framework is proposed to ensure data privacy and security. This framework has a multitree structure for indexing. An extirpation-based key derivation algorithm (EKDA) is used for key management, and discrete algorithm-based search on encrypted keyword (DLSEK) is used for data sharing and ciphertext retrieval. Lazy revocation is incorporated into the framework to deal with changes in user access rights and dynamic data operations. In [20], a mechanism with differential privacy is incorporated in the Map-Reduce computation model to analyze service efficiency and security of the mass data. A decision-tree generation algorithm is also incorporated into the computational model. Together, these measures satisfy ϵ -differential privacy.

Generally speaking, users distrust or only partly trust the cloud storage environment because as storage "tenants," they lack complete control over their data. A service provider has the potential to violate the privacy of user data, so the data needs to be processed before being uploaded to the cloud. Typically, data is encrypted, obfuscated, or anonymized before being uploaded.

Encrypting data negatively affects the processing of the data. Improving the speed and efficiency of ciphertext processing and retrieval is the focus of current research. In [21]–[23], the authors have done extended research on privacy preservation in the cloud and propose ciphertext retrieval solutions. In [24], a computable encryption scheme based on vector and matrix calculations (CESVMC) is proposed. In this scheme, cloud data is divided into two main categories: string and numeric. Encrypted strings can be retrieved using fuzzy retrieval, and the four basic arithmetic operations can be performed on numeric data.

Anonymous technology includes k -anonymity, L -diversity anonymous, and T -closeness anonymous. K -anonymity guarantees that each sensitive attribute is hidden in the scale of k groups [25]. This means that the probability of recognizing the individual does not exceed $1/k$. The level of privacy depends on the size of k . The statistical characteristics of the data are retained as much as possible; however, k -anonymity is not only applicable to sensitive data. An attacker could mount a consistency attack or background-knowledge attack to confirm a link between sensitive data and personal data. This would constitute a breach of privacy. L -diversity anonymous ensures that each group's sensitive attributes have at least L different values [26]. This means that an attack has a maximum probability of $1/L$ of recognizing a user's sensitive information. T -closeness anonymous is based on L -diversity anonymous [27]. In T -closeness anonymous, the distribution of the sensitive attribute is taken into account, and the distribution differ-

Data Security and Privacy in Cloud Storage

Xinhua Dong, Ruixuan Li, Wanwan Zhou, Dongjie Liao, and Shuoyi Zhao

ences between sensitive properties and values in groups does not exceed T . Anonymous technology is mainly used for database privacy, location privacy, and trajectory privacy, but we propose applying it cloud storage privacy.

In [28], a privacy manager that scrambles user data in the client is proposed. The privacy manager protects and monitors privacy according to the user's preferences. The privacy-preserving method in [24] supports data dyeing based on the normal cloud model in [10]. This method can be used to protect documents, images, videos, software, and other types of data. It also involves much less computation than traditional encryption or decryption. In [29], a novel privacy-preserving data-perturbation algorithm NETPA is proposed for clustering. The primitive data set can be perturbed by changing the value of neighboring main attributes, which is found in each data object, with the average attribute value of data objects in the data set's k -nearest neighborhood. This perturbation strategy is used to maintain stable k -nearest neighbor relations in primitive data. NETPA effectively stops privacy breaches. In [30], a novel data privacy protection mechanism based on partitioning and classification was proposed. The mechanism partitions the original data into a small, locally deployed block and a large, remotely deployed block. Then, data dyeing and data encryption are used according to the different security requirements of the data. This safeguards the privacy of data in the cloud, increases flexibility, and reduces overhead.

Much attention has been focused on safeguarding data at the storage provider's side; however, dynamic privacy needs at the user side have largely been ignored. Encryption, access control strategies, and other security mechanisms generally safeguard the privacy of data. There are many factors that allow privacy breaches, and user privacy requirements vary widely. Traditional authentication and security management strategies are insufficient for data stored in the cloud.

6 Conclusion

In this paper, we have described an architecture that ensures data privacy and security in cloud storage. We have also discussed access control and data integrity, confidentiality, availability, and privacy technologies. Cloud storage systems are moving towards unlimited bandwidth, capacity, and processing power, and data must be securely accessible anytime and anywhere. Because of changing demands, existing technology cannot ensure the privacy and security of data stored in the cloud. Further research needs to be done on scalability of secure storage and secure storage management in a large, complex cloud.

Storage devices are provided by a number of different service providers and shared by a large number of users. Frequent equipment deployment, data operations, and data access make the cloud a dynamic environment. Cloud data storage and management therefore needs to be safe but also highly scalable.

Intentional breaches of privacy call for dynamic countermeasures in a real-time cloud storage system. With frequent changes in computer technology, the means of breaching data privacy are constantly changing; consequently, security requirements are also changing. Privacy-preservation strategies need to be constantly devised for cloud storage systems.

An optimal balance must also be struck between security and availability in a cloud storage system. Security and availability exist in a contradictory relationship, and increasing safety often decreases availability. The foremost requirement of the data owner is security, and access limitations need to be imposed on applications. Further research is needed into the effects of security on data availability.

References

- [1] T. Mather, S. Kumaraswamy, S. Latif, "Cloud Security and Privacy," USA: O'Reilly Media, 2009.
- [2] D. Catteddu, G. Hogben, "Cloud Computing: Benefits, Risks and Recommendations for Information Security," Europe: European Network and Information Security Agency (ENISA), 2009.
- [3] S. Curry, J. Darbyshire, D. W. Fisher, et al, "Infrastructure Security: Getting to the Bottom of Compliance in the Cloud," *The Security Division of EMC*, March 2010.
- [4] D. G. Feng, M. Zhang, Y. Zhang, et al, "Study on Cloud Computing Security," *Journal of Software*, vol.22 no.1, pp. 71–83, 2011.
- [5] J. Y. Liu, C. Wang, X. D. Xue, "Cloud Storage Security," *ZTE Technology Journal*, vol.18 no.6, pp. 30–33, 2012.
- [6] C. Hong, M. Zhang, D. G. Feng, "AB-ACCS: A Cryptographic Access Control Scheme for Cloud Storage," *Journal of Computer Research and Development*, vol.47, pp. 259–265, 2010.
- [7] Z. Q. Lv, H. Cheng, M. Zhang, et al, "A secure and efficient revocation scheme for fine-grained access control in cloud storage," in *Proc. of 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Taipei, 2012, pp. 545–550.
- [8] S. C. Yu, C. Wang, K. Ren, et al, "Attribute based data sharing with attribute revocation," in *proc. of the 5th ACM Symp. on Information, Comput. and Commun. Security*, Beijing, 2010, pp. 261–270.
- [9] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *IEEE Symp. on Security and Privacy*, California, 2007, pp. 321–334.
- [10] J. Li, G. S. Zhao, X. F. Chen, et al, "Fine-Grained Data Access Control Systems with User Accountability in Cloud Computing," in *proc. of the 2th Int. Conference on Cloud Comput.*, Indiana, 2010, pp. 89–96.
- [11] L. Wang, L. Wang, M. Mambo, et al, "New identity-based proxy re-encryption schemes to prevent collusion attacks," in *proc. of Pairing-Based Cryptography-Pairing*, Ishikawa, 2010, pp. 327–346.
- [12] C. Centry, "A fully homomorphic encryption scheme," Stanford University, California, September 2009.
- [13] A. Juels, Burton S. Kaliski Jr, "Pors: proofs of retrievability for large files," in *proc. of the 2007 ACM Conf. on Comput. and Commun. Security*, Virginia, USA, 2007, pp. 584–597.
- [14] C. Wang, Q. Wang, K. Ren, et al, "Ensuring Data Storage Security in Cloud Computing," *IACR Cryptology ePrint Archive (IACR)*, 2009, 81.
- [15] Y. M. Huo, H. Y. Wang, L. Hu, et al, "A Cloud Storage Architecture Model for Data-Intensive Applications," *Computer and Management (CAMAN)*, Wuhan, 2011, pp. 1–4.
- [16] L. F. Zeng, Z. Shi, S. J. Xu, et al, "SafeVanish: An Improved Data Self-Destruction for Protecting Data Privacy," in *proc. of second IEEE int. conf. on cloud compu. technology and science (CloudCom)*, Indiana, 2010, pp. 521–528.
- [17] I. Roy, H. Ramadan, S. Setty, et al. Airavat: Security and privacy for map reduce, in *proc. of the 7th USENIX conf. on networked syst. design and implementation*, SanJose, 2010, pp. 297–312.
- [18] F. Z. Zhang, J. Chen, H. B. Chen, et al, "Lifetime Privacy and Self-Destruction of Data in the Cloud," *Journal of Computer Research and Development*, vol.48

- no.7,2011, pp. 1155–1167.
- [19] R. W. Huang, X. L. Gui, S. Yu, et al, "Design of Cloud Storage Framework with Privacy-Preserving," *Journal of XI' AN Jiaotong University*, vol.45 no.10, 2011, pp. 1–6.
- [20] S. Y. Yang, S. Q. Wang, "Research of Data Privacy & Security in Map-Reduce Model," *Computer Science*, vol.39 no.12, 2012, pp. 153–157.
- [21] S. Ananthi, M. S. Sendil, S. Karthik, "Privacy Preserving Keyword Search over Encrypted Cloud Data," *Advances in Computing and Communications*, 2011, pp. 480–487.
- [22] H. Hu, J. Xu, C. Ren, et al, "Proc. Private Queries over Untrusted Data Cloud through Privacy Homomorphism," in *Proc. the 27th IEEE Int. Conf. on Data Engineering (ICDE)*, Hannover, Germany, 2011.
- [23] N. Cao, C. Wang, M. Li, et al, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *2011 Proc. IEEE INFOCOM*, Shanghai, 2011, pp. 829–837.
- [24] R. W. Huang, X. L. Gui, S. Yu, et al, "Privacy-Preserving Computable Encryption Scheme of Cloud Computing," *Chinese Journal of Computers*, vol.34 no.12, 2011, pp. 2391–2402.
- [25] P. Samarati, L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10 no. 5, 2002, pp. 557–570.
- [26] A. Machanavajjhala, J. Gehrke, D. Kifer, et al, "L-diversity: Privacy beyond k-anonymity," *ACM Trans on Knowledge Discovery from Data (TKDD)*, vol. 1 no. 1, 2007, pp. 24–33.
- [27] L. Ninghui, L. Tiancheng, S. Venkatasubramanian, "t-Closeness: Privacy beyond k-anonymity and l-diversity," in *Proc. of the 23rd Int. Conf. on Data Engineering (ICDE)*, Istanbul, Turkey, 2007, pp. 106–115.
- [28] M. Mowbray, S. Pearson, "A client-based privacy manager for cloud computing," in *Proc. of the 4th Int. ICST Conf. on Commu*, New York, USA, 2009.
- [29] W. W. Ni, L. Z. Xu, Z.H. Chong, et al, "A Privacy-Preserving Data Perturbation Algorithm Based on Neighborhood Entropy," *Journal of Computer Research and Development*, vol.46 no.3, 2009, pp. 498–504.
- [30] X. L. Xu, J. L. Zhou, G. Yang, "Data Privacy Protection Mechanism for Cloud Storage Based on Data Partition and Classification," *Computer Science*, vol.40 no.2, 2013, pp. 98–102.

Manuscript received: May 15,2013

Biographies

Xinhua Dong (xhDong@hust.edu.cn) is a PhD student in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. He received his MS degree in computer science from HUST in 2008. His research interests include information retrieval, cloud security, and big data management. He is a student member of the CCF.

Ruixuan Li (rxli@hust.edu.cn) received his PhD in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2004. He is currently a professor in the School of Computer Science and Technology, HUST, and an adjunct associate professor at Concordia University, Canada. From 2009–2010, he was a visiting researcher at the University of Toronto. His research interests include cloud computing, big data management, social networking, and distributed system security. He has published more than 100 papers in refereed journals and conference proceedings. He has also co-authored two books and hold 14 China patents of invention. He is a member of IEEE and ACM and a senior member of the CCF.

Wanwan Zhou (zhoumila_wanwan@163.com) is an MS student in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. She received her BS degree in computer science from Wuhan University of Science and Technology, China, in 2012. Her research interests include cloud computing and big data security.

Dongjie Liao (sxxj0301@163.com) is an MS student in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. He received his BS degree from HUST in 2007. His research interests include cloud computing, Hadoop, and information retrieval. He has participated in many projects, and in 2010, was awarded second prize for outstanding software for engineering design (Hubei). In 2010, he also won first prize of for software for engineering design (Wuhan).

Shuoyi Zhao (zhaoshuoyi0508@vip.qq.com) is an MS student in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. He received his BS degree in computer science from Hunan Normal University, China, in 2011. His research interests include cloud computing, big data security, and information retrieval.

ZTE Makes Industry Breakthrough with 1 Gbps LTE-Advanced

ZTE Corporation achieved a wireless data transmission rate of 1 Gbps in a live demonstration at Mobile Asia Expo in Shanghai. This represents a global breakthrough in the development of next-generation LTE-A networks.

In the demonstration, ZTE used carrier aggregation technology with four carrier frequencies on the F band and D band. With carrier aggregation, two or more carrier frequencies sharing the same or different bands are aggregated into one channel. This increases peak transmission speed of TD-LTE cells. ZTE used three carrier frequencies in the 2.6 G band and one carrier in the 1.9 G band to complete the 1 Gbps demonstration.

ZTE leads the global telecommunications industry in TD-LTE 4G networks. In February, ZTE demonstrated the first F+D cross-band CA transmission, achieving speeds of 430 Mbps. In January, ZTE and China Mobile completed D-band carrier aggregation testing in a trial network in Guangzhou. An outdoor transmission speed of 223 Mbps was achieved.

Carrier aggregation counters signal interference between neighboring cells that share the same band. By balancing the load between the primary carrier and secondary carrier, network capacity can be increased, and operators can provide customers with higher network speeds and richer user experience. Carrier aggregation is helping the telecommunications industry meet the challenges associated with surging data traffic and is making operator TD-LTE networks more competitive.

(ZTE Corporation)

An Efficient Dynamic Proof of Retrievability Scheme

Zhen Mo, Yian Zhou, and Shigang Chen

DOI: 10.3969/j. issn. 1673-5188. 2013. 02. 004

<http://www.cnki.net/kcms/detail/34.1294.TN.20130608.0953.001.html>, published online June 8, 2013

An Efficient Dynamic Proof of Retrievability Scheme

Zhen Mo, Yian Zhou, and Shigang Chen

(Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL 32611, USA)

Abstract

Data security is a significant issue in cloud storage systems. After outsourcing data to cloud servers, clients lose physical control over the data. To guarantee clients that their data is intact on the server side, some mechanism is needed for clients to periodically check the integrity of their data. Proof of retrievability (PoR) is designed to ensure data integrity. However, most prior PoR schemes focus on static data, and existing dynamic PoR is inefficient. In this paper, we propose a new version of dynamic PoR that is based on a B+ tree and a Merkle hash tree. We propose a novel authenticated data structure, called Cloud Merkle B+ tree (CMBT). By combining CMBT with the BLS signature, dynamic operations such as insertion, deletion, and modification are supported. Compared with existing PoR schemes, our scheme improves worst-case overhead from $O(n)$ to $O(\log n)$.

Keywords

cloud storage; proof of retrievability; data integrity; B+ tree

1 Introduction

Cloud storage is an online storage model. A cloud storage business provides data access and storage services with pay-as-you-go options. Developers and users can easily adapt resources to their needs and do not need to know the physical location or configuration of the system that delivers services. This elasticity, provided without any need for investment, is attracting more and more people to use cloud storage. Although it is as a promising service model, cloud storage also creates security problems. One of the main problems is the integrity of data stored in the cloud. After outsourcing the data to an offsite storage system and deleting the local copies, a client is relieved of the burden of storage. At the same time, it loses its physical control over the data. A cloud storage system is maintained by a third party who rarely has an infallible security system. Therefore, it is extremely important for the clients to have an effective way of periodically checking the integrity of their data. Many schemes have been proposed to address this issue [1]–[7]. These schemes fall into two categories according to design goal: proof of retrievability (PoR) [1], [2], [4], [6] and provable data possession (PDP) [3], [7]. The PoR scheme was proposed in [4]. The design goal was to ensure that clients could retrieve data from the server side. In [3], a similar scheme, called PDP was proposed. In this scheme, clients are used to demonstrate that files are stored correctly at the server side. PDP is weaker than PoR because PDP assurance is weaker than that in PoR. PDP also does not guarantee that clients can retrieve their data in-

tact. With PDP, clients query the server periodically, and the server returns a proof to guarantee that a certain percentage (e. g. 99%) of the file is intact. However, if a very small amount of the file is lost or corrupted, clients may not be able to detect and retrieve the file intact. With PoR, clients may not detect corruption, but they can still recover the file with the help of an erasure code. In this paper, we mainly consider PoR.

Another important concern is support for dynamic updates. In a cloud storage system, clients should not only be able to access data but also dynamically modify, delete, or insert data. Most of previous works only focus on static data files [2], [4], [6], [7]. Although a dynamic PoR model is proposed in [1], unfortunately, the scheme's performance is not tightly bounded.

In this paper, we propose a dynamic new PoR scheme based on a modified Merkle hash tree (MHT) and Boneh-Lynn-Shacham (BLS) signature construction [8]. We design a dynamic PoR model for the cloud storage system and propose a new data structure called Cloud Merkle B+ Tree (CMBT). When CMBT is combined with BLS construction, the worst-case performance scenario is $O(\log n)$.

In section 2, we discuss the system model and security for a typical cloud storage system. In section 3, we introduce background research. In section 4, we present our dynamic PoR scheme. In section 5, we analyze the results of simulations run on our storage system.

2 System Model and Security

A typical cloud storage system includes storage servers and

clients. Clients have limited storage space but have a large amount of data to be stored. Cloud storage servers have a huge amount of storage space that can be made available on a pay-as-you-go basis. Cloud storage servers are maintained by a cloud service provider (CSP) such as Amazon or Google. Clients divide the data files into blocks that are placed into cloud storage servers. Clients delete local copies and only retain a small amount of metadata. In addition, a storage service is not static; clients can delete, insert into, or otherwise modify a block.

A CSP is a third party that does not have infallible security. We propose the following semi-trust model: In normal cases, the CSP operates correctly and does not deliberately delete or modify client data. However, management errors, Byzantine failures, and external intrusions may cause the CSP to inadvertently lose or corrupt hosted data. When these errors occur, the CSP tries to salvage its reputation by hiding the true extent of data loss.

Our new dynamic PoR scheme solves efficiency problems in existing dynamic PoR. With our scheme, file corruptions can be detected with high probability, even if a CSP tries to hide them. Files can also be dynamically updated without affecting the probability of detecting file corruption.

To simplify our discussion, we treat cloud storage servers as one entity (the server) and clients as the other entity (the client).

3 Related Work

PoR was first formalized in [4]. In PoR, “sentinel” blocks are randomly embedded in the outsourced file, and the position of the blocks is hidden by encryption. In this scheme, static data corruption can be effectively detected. However, data cannot be updated, and the number of queries a client can send is fixed.

PDP was first proposed in [3]. This scheme is used to ensure the integrity of outsourced data and makes use of RSA-based homomorphic tags. However, it cannot be used in dynamic scenarios. In [7], a version of dynamic PDP is introduced, but it also cannot be used in fully dynamic scenarios.

In [2], an improved PoR scheme, called Compact PoR, is introduced with rigorous security proofs [2]. Using the BLS signature, proofs are aggregated into a small value, and public verification is supported. However, this scheme is impractical and insecure in dynamic scenarios. There are two reasons for this: 1) block signatures contain the indices of blocks, and 2) replay attacks are not prevented. If a client deletes or inserts a block with index i , then any block with index j , $j > i$ will have to change its index to $j - 1$ or $j + 1$. The client has to re-sign all the blocks whose indices have been changed, and this makes the scheme impractical in terms of dynamic updates.

In [1], another dynamic PoR scheme is defined. In this scheme, the BLS signature and MHT are modified so that integ-

ity can be verified in cloud storage [9]. In order to build an MHT over a large piece of data (e.g. a file), the client first divides the file into data blocks m_i , $1 \leq i \leq n$ and computes the hash value for each block. This hash value is given by $n_i = H(m_i)$. We call n_i the “block tag” of m_i . Then, the client constructs a binary tree whose leaf nodes are the hashes of the block tags. Nodes that are further up in the tree are the hashes of their respective children. Finally, the client generates a root R based on the MHT and takes the signature of the root $sig_{sk}(R)$ as metadata.

Using a classic MHT is inefficient. After inserting or deleting some blocks, the MHT becomes unbalanced. If the client keeps appending blocks to the tail of the file, the height of the tree increases linearly. As a result, the worst-case scenario in an integrity check would be $O(n)$ instead of $O(\log n)$ [1], where n is the total number of blocks.

4 Our Scheme

4.1 Overview

Our scheme comprises three stages: 1) preprocessing, 2) verification, and 3) updating. In the preprocessing stage, the client encodes the file with an erasure code and divides the encoded file into blocks. This is done before the file is outsourced to the server. Then, an authenticated data structure is constructed, and the metadata is generated. The client only keeps the metadata and outsources other data to the server. In the verification stage, the client periodically checks the integrity of its data. This is done after outsourcing to the server. The client queries the server with a subset of the data blocks and requires the server to provide a proof. By verifying the proof against the metadata, the client can accurately detect file corruption. In the update stage, the client sends the server a request to update the file. After each update, the server proves to the client that the update has been successful.

4.2 Model

Our scheme can be described by the following algorithms:

- **KeyGen** (1^k) $\rightarrow (pk, sk)$. This is an algorithm run by the client. The input is a security parameter, and the output is a public key pk and a private key sk . The client stores sk and sends pk to the server.
- **Prepare** (sk, F', F_{tags}) $\rightarrow (\Phi, sig_{sk}(l(R)), \text{CMBT})$. This algorithm is executed by the client. The input is an encoded file F' , which is created by a sequence of blocks m_i , $0 \leq i \leq n$; the block tag set $F_{\text{tags}} = \{H(m_i), 0 \leq i \leq n\}$; and sk . The output is a signature set Φ , which is an ordered collection of signatures $\{\sigma_i\}$ on $\{m_i\}$, $0 \leq i \leq n$. We define the signature set in subsection 4.3. The client also constructs a CMBT by using F_{tags} and signs the label value of root $sig_{sk}(l(R))$ by using sk .
- **GenChallenge** (n) $\rightarrow Q$. This algorithm is executed by the cli-

An Efficient Dynamic Proof of Retrievability Scheme

Zhen Mo, Yian Zhou, and Shigang Chen

ent. The input is the total number of blocks, and the output is a query Q that contains a set of IDs $I = \{i_1, i_2, \dots, i_k\}$. The query is sent to the server as a request to verify the integrity of blocks with index i , for $i \in I$.

- $GenProof(Q, CMBT, F', F_{tags}, \Phi) \rightarrow P$. This algorithm is executed by the server. The input is $Q, CMBT, F', F_{tags}$, and Φ . The output is a proof P that allows the client to check the integrity of the blocks in Q .
- $Verify(pk, Q, P, l(R)) \rightarrow (TRUE, FALSE)$. This algorithm is executed by the client. After receiving the proof P , the client checks the integrity of blocks in Q . The client then outputs $TRUE$ if the integrity of the blocks is confirmed; otherwise, it outputs $FALSE$.
- $UpdateRequest() \rightarrow Request$. This algorithm is executed by the client. Nothing is input. The output is an update request R that contains an $Order \in \{Insert, Delete, Modify\}$ and an index number i . If $Order$ is $Modify$ or $Insert$, then R should also contain a new file block m^* and its signature σ^* .
- $Update(F', F_{tags}, \Phi, R) \rightarrow (P_{old}, P_{new})$. This algorithm is executed by the server. After receiving the R from the client, the algorithm takes F', F_{tags}, Φ , and R as input and outputs two proofs: P_{old} and P_{new} .
- $UpdateVerify(P_{old}, P_{new}) \rightarrow (TRUE, FALSE)$. This algorithm is executed by the client. With P_{old} and P_{new} , the client outputs $TRUE$ if the server's behaviors are honest in the update process; otherwise, it outputs $FALSE$.

4.3 Preprocessing

Before outsourcing the files to the server, the client encodes F to F' using an erasure code. Then, the client runs $KeyGen(1^k)$ to create a pair of keys and uses $Prepare(sk, F, F_{tags})$ to generate Φ , a CMBT, and the metadata $sig_{sk}(l(R))$.

We use the same BLS signature as that defined in [1]. For a bilinear map $e: G \times G \rightarrow GT$, sk and pk are defined as $x \in \mathbb{Z}_p$ and $v = g^x \in G$, respectively, where g is a generator of G . For each $m_i, i \in [1, n]$, the signature on m_i is defined as $\sigma_i = [H(m_i)u^{mi}]^x$, where u is a generator of G . We denote the set of the signature as $\Phi = \{\sigma_i\}, 1 \leq i \leq n$.

MHT [9] has been widely used for checking memory integrity [10], [11] and certificate revocation [12], [13] because it is easy to realize and has $O(\log n)$ complexity in both worst-case and usual scenarios. However, using the classic MHT in cloud storage may cause problems (section 3). Therefore, we develop an authenticated data structure based on a B+ tree and MHT.

We call this new structure a cloud Merkle B+ tree (CMBT). We choose a third-order B+ tree¹ and require each data node to store three elements at most.

We treat the sequence of block tags $H(m_1), H(m_2), \dots, H(m_n)$ as elements and insert them into a B+ tree sequentially. Then, we obtain a B+ tree (Fig. 1) and base the CMBT on it.

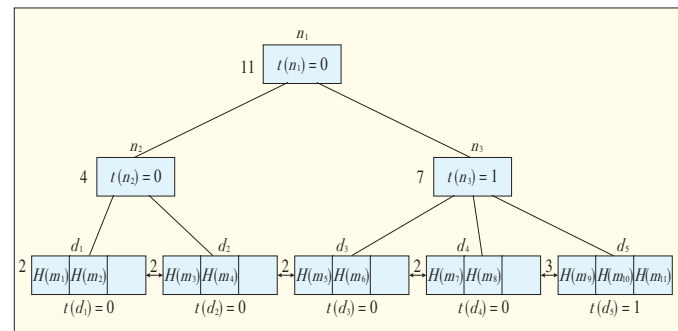
For each node w in a CMBT, the following values are stored:

- $left(w)$, $middle(w)$, and $right(w)$. For an index node, these variables represent the left child, middle child, and right child of the node, respectively. If the node only has two children, then $right(w)$ will be 0. For a data node, these variants represent the elements the node stores (from left to right). If a corresponding position has no element, 0 is set.
 - $r(w)$. This is the rank of node. For an index node w , $r(w)$ stores the number of elements that belong to the subtree whose root is w . For a data node w , $r(w)$ stores the number of elements that belong to w . Fig. 1 shows the rank for each node. The rank for node d_1 is 2 because from d_1 we can visit two elements: $H(m_1)$ and $H(m_2)$.
 - $t(w)$. Keys are not stored in an index node because the CMBT does not need to be searched. Instead, the type of the node is stored as $t(w)$, where
- $$t(w) = \begin{cases} 0 & \text{if } w \text{ has 2 children or contains 2 elements} \\ 1 & \text{if } w \text{ has 3 children or contains 2 elements} \end{cases}$$
- $l(w)$. This is the label of node. To define $l(w)$, first we define a collision-resistant hash function $h(*)$ that has two inputs: $h(a, b) = h(a \parallel b)$, where \parallel means concatenation. Then, we extend the function to more than two inputs: $h(a_1, a_2, \dots, a_{n-1}, a_n) = h(a_1 \parallel a_2 \parallel \dots \parallel a_{n-1} \parallel a_n)$. Then, $l(w) = h(l(left(w)), l(middle(w)), l(right(w)), t(w), r(w))$. For each element e that contains a block m , the value of the element is given by $l(e) = h(H(m))$.

With above definitions, the client can construct a CMBT and obtain the label value of the root R . Then, the client signs the root label $l(R)$ using its private key: $sig_{sk}(l(R)) \leftarrow (l(R))^{sk}$. Next, the client outsources F, Φ , the CMBT, and $sig_{sk}(l(R))$ to the server.

4.4 Query

Suppose F', Φ , CMBT, and $sig_{sk}(l(R))$ have been out-



▲ Figure 1. A cloud Merkle B+ tree.

¹ A B+ tree [14] differs from a B tree in the following three respects:

1. A B+ tree has two types of nodes: index and data. Index nodes store keys, and data nodes store elements. A B tree only has data nodes.
2. Data nodes in a B+ tree are linked by a doubly linked list whereas data nodes in a B tree are not linked.
3. The capacity of data nodes and index nodes can differ in a B+ tree whereas the capacity of nodes in a B tree should be the same. In a B+ tree of order n , index nodes (except for the root node) can hold a maximum of $n-1$ keys and a minimum of $\lceil n/2 - 1 \rceil$ keys. Each data node can contain a maximum of c elements and a minimum of $\lceil n/2 \rceil$ elements (c and n can differ). The root node can hold a maximum of n children and a minimum of two children.

sourced to the server. The client only stores the metadata and number of blocks n , and it generates a query to check the integrity of a series of random blocks whose index numbers belong to the set $I = \{i_1, i_2, \dots, i_k\}$. The client uses the *GenChallenge* (n) $\rightarrow Q$ algorithm to generate a Q . For each index number $i \in I$, the client chooses a random element $v_i \leftarrow \mathbb{Z}_p$. Then, $Q = \{(i, v_i)\}, i_1 \leq i \leq i_k$.

After receiving Q , the server executes *GenProof* (Q , CMBT, F, F_{tags}, Φ) $\rightarrow P$ to generate a proof P by first computing μ and σ :

$$\mu = \sum_{i=i_1}^{i_k} v_i m_i \in \mathbb{Z}_p \quad (1)$$

$$\sigma = \prod_{i=i_1}^{i_k} \sigma_i^{v_i} \quad (2)$$

Then, the server generates a sequence of messages for each block tag $H(m_i)$ in the block tag set $S = \{H(m_i), i \in I\}$. Suppose $\{w_1, w_2, w_3, \dots, w_h, w_{h+1}\}$ is the path from the root node to the element $H(m_i)$, where $i \in [i_1, i_k]$; h is the height of the CMBT; and w_j is the parent node of w_{j+1} . For each node $w_j, j \in [1, h]$, the server provides a message M_j that is a 2-tuple value. We define n_{j+1} and n'_{j+1} as neighbors of w_{j+1} , and $n(j+1)$ is always to the left of n'_{j+1} . We denote T as a set of nodal information that is given by

$$\begin{aligned} T_{n_{j+1}} &= \{l(n_{j+1}), r(n_{j+1}), t(n_{j+1}), p(n_{j+1})\} \\ T_{n'_{j+1}} &= \{l(n'_{j+1}), r(n'_{j+1}), t(n'_{j+1}), p(n'_{j+1})\} \end{aligned} \quad (3)$$

If w_{j+1} has only one sibling, then $T(n'_{j+1}) = \text{NULL}$. The location relationship between n_{j+1} and w_{j+1} is given by

$$p(n_{j+1}) = \begin{cases} 0 & \text{if } n_{j+1} \text{ is to the left of } w_{j+1} \\ 1 & \text{if } n_{j+1} \text{ is to the right of } w_{j+1} \end{cases} \quad (4)$$

Therefore, a message for w_j is given by

$$M_j = \{T_{n_{j+1}}, T_{n'_{j+1}}\} \quad (5)$$

The message sequence for element $H(m_i)$ is given by $\gamma_i = \{M_1, M_2, \dots, M_h\}$, and for all elements in set I , the message set is given by $\Gamma = \{\gamma^1, \dots, \gamma^k\}$. Therefore, $P = \{\mu, \sigma, S, \Gamma\}$.

If the client only wants to check the integrity of one block instead of a group of blocks, the proof of a single block with index i is given by

$$\text{Query}(i) = \{v_i m_i, \sigma_i^{v_i}, H(m_i), \gamma_i\} \quad (6)$$

4.5 Verification

After receiving proof P from the server, the client executes *Verify* ($pk, Q, P, l(R)$) (Algorithm 1) to check the integrity of the blocks whose indices belong to I . In Algorithm 1, $\{w_1, w_2, w_3, \dots, w_h, w_{h+1}\}$ is the node sequence from the root to the element $H(m_i)$. To compute $w_j, 1 \leq j \leq h$, the client first determines how many children w_j has. Then, the client inputs the children's values and their locational relationship $p(1)$ into

the *GetValue* function in order to compute w_j . The computation continues until the root node is reached. During the procedure, the client can verify the index number idx of the block tag $H(m_i)$.

Algorithm 1 *Verify* ($pk, Q, P, l(R)$) $\rightarrow (TRUE, FALSE)$

```

1: Verify  $e(\sigma, g) \stackrel{?}{=} e(\prod_{i=i_1}^{i_k} H(m_i)^{v_i} \cdot u^\mu, v)$ 
2: for  $i$  from  $i_1$  to  $i_k$  do
3:    $\gamma_i = \{M_1, M_2, \dots, M_h\}, M_j = \{T_{n_{j+1}}, T_{n'_{j+1}}\}$ 
4:    $T_{n_{j+1}} = \{l(n'_{j+1}), r(n'_{j+1}), t(n'_{j+1}), p(n'_{j+1})\}$ 
5:    $T_{n'_{j+1}} = \{l(n'_{j+1}), r(n'_{j+1}), t(n'_{j+1}), p(n'_{j+1})\}$ 
6:    $idx = 1$ 
7:   for  $j$  from  $h$  down to 1 do
8:     if  $T_{n'_{j+1}} \neq \text{NULL}$  then
9:        $r(w_j) = r(w_{j+1}) + r(n_{j+1}) + r(n'_{j+1})$ 
10:       $t(w_j) = 1$ 
11:       $l(w_j) = \text{GetValue}(T_{n_{j+1}}, T_{n'_{j+1}})$ 
12:      if  $p(n'_{j+1}) = 0$  then
13:         $idx = idx + r(n'_{j+1})$ 
14:      end if
15:    else
16:       $r(w_j) = r(w_{j+1}) + r(n_{j+1})$ 
17:       $t(w_j) = 0$ 
18:       $l(w_j) = \text{GetValue}(T_{n_{j+1}})$ 
19:    end if
20:    if  $p(n_{j+1}) = 0$  then
21:       $idx = idx + r(n_{j+1})$ 
22:    end if
23:  end for
24:  if  $l(w_1) = l(R)$  AND  $idx = i$  then
25:    if  $i = i_k$  then
26:      return TRUE
27:    end if
28:  else
29:    return FALSE
30:  end if
31: end for
```

4.6 Updates

Here, we show that our scheme can be used to delete, insert into, or otherwise modify blocks. We assume that F', Φ , and CMBT have been generated and stored in the server.

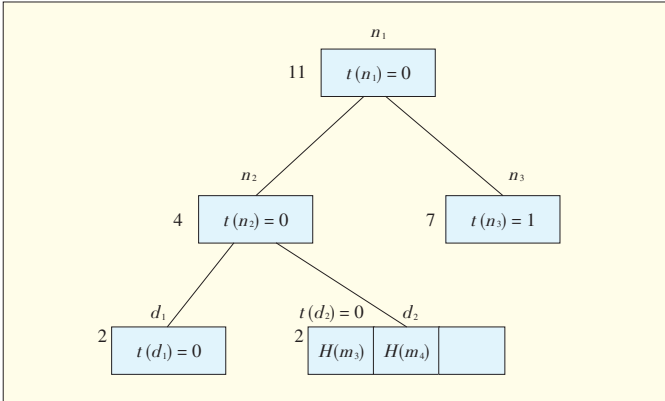
Suppose the client wants to update the j th block, $1 \leq j \leq n$. The client first executes *UpdateRequest* (\cdot) $\rightarrow \text{Request}$ to generate an update request and send it to the server. Upon receiving the modification request, the server updates the block and executes *Update* ($F', F_{\text{tags}}, \Phi, R$) to generate P_{old} and P_{new} . With P_{old} and P_{new} , the client executes *UpdateVerify* ($P_{\text{old}}, P_{\text{new}}$) to ensure the correctness of the update.

Suppose a client wants to modify the j th block, $1 \leq j \leq n$, from m_j to m'_j . The client generates an update request *Re-*

An Efficient Dynamic Proof of Retrievability Scheme

Zhen Mo, Yian Zhou, and Shigang Chen

$quest = \{Modify, j, m'_j, \sigma'_j\}$ and sends it to the server. The server updates the block, reconstructs the CMBT, and generates the proof $P_{old} = Query(i)$ (6). With P_{old} , the client can check the integrity of m_j (Algorithm 1) and construct a partial CMBT (Fig. 2). The partial CMBT is constructed from the query on the CMBT in Fig. 1. The client obtains enough information to update the CMBT from the partial CMBT. In this case, the cli-



▲ Figure 2. Partial CMBT constructed from $P_{old} = Query(4)$.

ent computes the new root R_{new} using P_{old} . The server only needs to send $P_{new} = R'$, which is the new root node to the client. After verifying the correctness of R' , the client signs the new root $sig_{sk}(l(R'))$ and sends it back.

The procedure for insertion is similar to that for modification. The only difference is that when a new element is inserted into a data node that already contains three elements, the data node splits in two. The procedure keeps going until one index node has only two children or we need to generate a new root and increase the height of the tree by one. With the partial CMBT constructed from $P_{old} = Query(i)$, the client has enough information to compute R_{new} . After verifying the correctness of R' , the client signs the new root $sig_{sk}(l(R'))$ and sends it back.

The procedure for deletion differs from that for insertion and modification. Deleting an element from a data node with two elements makes the data node deficient. Therefore, borrow or merge operations need to be performed to keep the tree balanced [14]. Because the partial CMBT is constructed from the P_{old} , the client may not acquire enough information to finish these operations and compute R_{new} . Accordingly, the server needs to send another P_{new} to help the client verify the correctness of R' . Here, we define another algorithms: Algorithm $Query_{new}(i)$ is used to return the proof of the i th element in the updated CMBT.

Using P_{old} , the client can generate a partial CMBT that contains the node sequence $\{w_i\}$ and its siblings $\{n_i, n'_i\}$, $i \in [1, h + 1]$. The element that the client wants to delete is denoted w_{h+1} . There are three cases of deletion:

1) If the leaf node w_h contains three elements, then the client only needs to delete w_{h+1} and generate R' based on P_{old} . Otherwise, the client keeps searching the node sequence from

w_h to w_1 until it finds w_j , $j \in [1, h]$. The right or left sibling nodes of w_j has three children or w_j itself has three children.

- 2) If one sibling of w_j has three children, then the client needs to borrow a child from its sibling to generate a new node. However, P_{old} does not contain the information of this child. The client will therefore use $Query_{new}(i)$ and $Query(k)$ to acquire additional information to delete w_{h+1} and generate a new root. The index number of the element that belongs to the subtree (whose root is the sibling node) is given by k . The client can obtain k easily from P_{old} . The information from $Query_{new}(i)$ can be verified by P_{old} .
- 3) If w_j has three children, the client deletes the element and merges two children. By using $Query_{new}(i)$, the client acquires enough information to generate the new root. If the client cannot find the node until it reaches the root, the client generates a new root. In this case, the CMBT height decreases by one. Here, we do not provide the complete algorithm, but the complexity of the deletion is $O(\log n)$.

5 Simulation Results

In Table 1, we show the performance of our scheme and that of existing PoR schemes on a feature-by-feature basis. Our experiment ran on a system with an Intel Core 2 2.53 GHz pro-

▼ Table 1. Performance of existing PoR schemes

Features	Schemes			
	[4]	[2]	[1]	Our Scheme
Dynamic updates	No	No	Yes	Yes
Public verification	No	Yes	Yes	Yes
Worst comm. complexity	$O(1)$	$O(1)$	$O(n)$	$O(\log n)$
Avg. comm. complexity	$O(1)$	$O(1)$	$O(\log n)$	$O(\log n)$

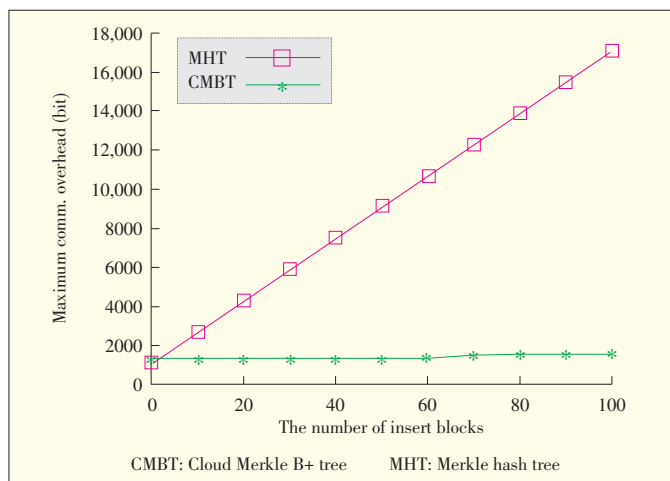
cessor, 4 GB RAM, and a 7200 RPM TOSHIBA 120 GB SATA driver. Algorithms were implemented using C++.

We evaluated the performance of our scheme in terms of overhead. Using previous analysis, we determined that the overhead of PoRs depends on the block size and the number of messages (hashes) sent to the client. As in [3], detecting 1% file corruption with 99% confidence requires querying a constant 460 blocks. Accordingly, if the block size is fixed, performance is determined by the overhead in sending messages to prove the index of a block in the tree. In our experiment, we send messages using SHA1 with an output of 160 bits. The average overhead of our scheme and that in [1] is similar; therefore, in Fig. 3, we show the maximum overhead of proving a block in the CMBT and MHT.

The client divides the encoded file F' into 128 blocks and uses these blocks to construct the MHT and the CMBT. Then, the client outsources F' and the MHT, and CMBT to the server. If the client keeps appending blocks to the tail of F' , the maximum overhead between the MHT and CMBT is as shown in

An Efficient Dynamic Proof of Retrievability Scheme

Zhen Mo, Yian Zhou, and Shigang Chen



▲ Figure 3. Maximum overhead for proving one block in the MHT and CMBT.

Fig. 3. The x axis shows the number of blocks that the client appends to the encoded file after initialization. The y axis shows the overhead for proving a block in the tree. From Fig. 3, the worst-case overhead for the MHT increases linearly with the number of block inserted. The worst-case overhead for MHT is given by $O(n)$.

6 Conclusion

Cloud storage creates security issues, especially in terms of data integrity. In this paper, we extend the static PoR scheme so that it can be used for dynamic scenarios. We propose a new authentication data structure called Cloud Merkle B+ tree. The worst-case overhead for existing dynamic PoR scheme is given by $O(n)$; however, the worst-case overhead for our scheme is given by $O(\log n)$.

Acknowledgment

This work was supported in part by the US National Science Foundation under grant CNS-1115548 and a grant from Cisco Research.

References

- [1] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. 14th European Conf. Research in Computer Security (ESORICS'09)*, St. Malo, France, pp. 355–370, Sep. 2009.
- [2] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. 14th Int. Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'08)*, Melbourne, Australia, pp. 90–107.
- [3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conference on Computer and Communications Security (CCS'07)*, Berlin, Germany, pp. 598–609.
- [4] A. Juels and B.S. Kaliski Jr, "Pors: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comp. and Commun. Security (CCS'07)*, Berlin, Germany, pp. 584–597.
- [5] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 16th ACM Conf. Computer and Commun. Security (CCS'09)*, Chicago, IL, pp. 213–222.
- [6] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proc. ACM Workshop on Cloud Comput. Security (CCSW'09)*, Chicago, IL, pp. 43–54.
- [7] G. Ateniese, R. Di Pietro, L.V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Security and Privacy in Commun. Networks*, Istanbul, Turkey, article 9.
- [8] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *Proc. 14th Int. Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'01)*, Gold Coast, Australia, pp. 514–532.
- [9] R. Merkle, "A digital signature based on a conventional encryption function," in *Proc. CRYPTO'87: Conf. on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, Santa Barbara, CA, pp. 369–378.
- [10] D. Williams and E.G. Sirer, "Optimal parameter selection for efficient memory integrity verification using Merkle hash trees," in *Proc. 3rd IEEE Int. Symp. Netw. Comp. Applications (NCA'04)*, Cambridge, MA, pp. 383–388.
- [11] B. Gassend, G.E. Suh, D. Clarke, M. Van Dijk, and S. Devadas, "Caches and hash trees for efficient memory integrity verification," in *Proc. 9th IEEE Int. Symp. High-Performance Computer Architecture*, Anaheim, CA, pp. 295–306.
- [12] H. Kikuchi, K. Abe, and S. Nakanishi, "Online certification status verification with a red-black hash tree," in *Proc. 1st Int. Workshop for Asian PKI (IWAP'01)*, Korea. [Online]. Available: https://www.jstage.jst.go.jp/article/ipsjdc/2/0/2_0_513/_pdf
- [13] M. Naor and K. Nissim, "Certificate revocation and certificate update," in *IEEE J. Selected Areas in Communications*, vol. 18, no. 4, pp. 561–570, 2000.
- [14] E. Horowitz and S. Sahni, *Fundamentals of data structures*, Michigan: Computer Science Press, 1983.

Manuscript received: April 29, 2013

Biographies

Zhen Mo (zmo@cise.ufl.edu) is a PhD student in the Department of Computer and Information Science Engineering, University of Florida. He received his BE degree in Information Security Engineering from Shanghai Jiao Tong University in 2007. He received his ME degree in theory and new technology of electrical engineering from Shanghai Jiao Tong University in 2010. His research interests include network security and cloud computing security.

Yian Zhou (yian@cise.ufl.edu) is a PhD student in the Department of Computer and Information Science and Engineering, University of Florida. She received her BS degree from Peking University in 2010. Her research interests include cloud computing and privacy-preserving cyber-physical systems.

Shigang Chen (sgchen@cise.ufl.edu) is an associate professor in the Department of Computer and Information Science and Engineering, University of Florida. He received his BS degree in computer science from University of Science and Technology, China, in 1993. He received his MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign in 1996 and 1999. After graduating, he worked at Cisco Systems for three years. He joined the University of Florida in 2002. His research interests include network security and wireless networks. He received the IEEE Communications Society Best Tutorial Paper Award in 1999 and NSF CAREER Award in 2007. He was a guest editor for *ACM/Baltzer Journal of Wireless Networks* and *IEEE Transactions on Vehicle Technologies*. He was TPC co-chair of the *IEEE Computer and Network Security Symposium 2006*, vice TPC chair of *The 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems 2005*, vice general chair of *QShine 2005*, and TPC co-chair of *QShine 2004*.

SPBD: Streamlining Big-Data Processing in Cloud Environments

Tung Nguyen, Jingwen Zhang, and Weisong Shi

DOI: 10.3969/j.issn.1673-5188.2013.02.005

<http://www.cnki.net/kcms/detail/34.1294.TN.20130627.1101.002.html>, published online June 27, 2013

SPBD: Streamlining Big-Data Processing in Cloud Environments

Tung Nguyen, Jingwen Zhang, and Weisong Shi

(Department of Computer Science, Wayne State University, Detroit, MI 48202, USA)

Abstract

Many applications, such as those in genomics, are designed for one machine. This is not problematic if the input data set is small and can fit into the memory of a single powerful machine. However, the application and its algorithms are limited by the capacity and performance of the machine (the application cannot run in parallel). A single machine cannot handle very large data sets. In recent research, cloud computing and MapReduce have been used together to store and process big data. There are three main steps in handling data in the cloud: 1) the user uploads the data, 2) the data is processed, and 3) results are returned. When the size of the data reaches a certain scale, transmission time becomes the dominant factor; however, most research to date has only been focused on reducing the processing time. Also, it is generally assumed that the data is already stored in the cloud. This assumption does not hold because many organizations now store their data locally. In this paper, we propose SPBD (pronounced “speed”) to minimize overall user wait time. We abstract overall processing time as an optimization problem and derive the optimal solution. When evaluated on our private cloud platform, SPBD is shown to reduce user wait time by up to 34% for a traditional WordCount application and up to 31% for a metagenomic application.

Keywords

bigdata; genomics; NGS; MapReduce; cloud

1 Introduction

Recent improvements in genomic technology have allowed researchers to cheaply sequence significantly more data than before. However, storing and processing mass data derived from next-generation sequencing (NGS) is challenging. In our research facility at Wayne State University, the Illumina HiSeq sequencing system can generate about 250 million reads (about 60 GB) per run [1]. It usually has eight lanes, each of which can handle 12 samples at most. In total, we can have 96 samples (i.e. 96 read files), each of which is 60 GB (pair-end read) for one run. It takes several days to generate the data, and with state-of-the-art Noalign alignment software, it takes two days to process the data [2].

If security is not a strict requirement, a cloud can be used to store and process a large data set. A cloud provides highly available, almost unlimited data storage and elastic computing power. There are also no up-front costs. When dealing with a large data set, a parallel programming model is usually the most effective computing model. Google MapReduce [3] and Microsoft Dryad [4] are among the most widely used frameworks for parallelizing genomic applications [5]–[8]. Using a cloud and parallel computing is not always easy for biologists,

who are often not experts in computer science. However, recent attempts have been made to improve the usability of cloud and parallel computing for these scientists [9], [10].

In practice, the place where data is usually generated is different to that where it is processed and analyzed [11]. Therefore, results of the data analysis are usually obtained by moving the data to the data center; running the analysis software inside the data center, cloud or local cluster; and downloading the results to the scientist’s local machine.

Methods such as scp shell command, ftp client, and web (http) are often used by domain scientists to transfer data to remote sites. Even when the data is large, such methods are still used. Most domain scientists are not aware of other methods, such as GridFTP. One of the most significant issues with transferring a large file over the network is failure. Traditional data transfer methods do not handle failure well.

In this paper, we focus on cloud MapReduce, for example, Amazon Elastic MapReduce (EMR). We use the Hadoop implementation of MapReduce as our framework; therefore, the applications are of the MapReduce-compatible variety only. In this context, the data is usually moved to Amazon S3 by the AWS management console, s3cmd, BucketExplorer, or CloudBerry Explorer. Amazon announced its support for objects larger than 5 GB in November 2010; however, at that time, many

S3 clients did not support the new multipart upload feature. This feature was only recently supported in the newest (beta) s3cmd client, which was released in April 2012.

Having narrowed down the context, we now describe how data is moved to HDFS. Traditionally, the most popular method of moving or copying data from a local machine to HDFS is to use Hadoop `dfs-put <filename>` or `distcp` shell command. With Amazon EMR, data is first moved to Amazon S3. In this process, command-line interface (cli) tools, such as `s3cmd` and `S3-Util`, are used. Graphical user interface (GUI) tools, such as CloudBerry Explorer and Bucket Explorer, are also used. Then, either GUIbased AWS Management Console or Amazon EMR Ruby Client (based on cli) is used to create a job flow for running a MapReduce application with the uploaded data.

With mass data, Amazon suggests using their physical import/export service. Users can store up to 4 TB of data in an eS-ATA or USB 2.0 portable hard drive and have it couriered to the nearest Amazon site. In general, there are many other tools, such as Pentaho [1] and Apache Sqoop, that can be used to transfer data to and from HDFS.

Most current work on MapReduce for cloud is focused on either improving the processing time t_{proc} of the software [5], [6], [8] or improving the data transmission time t_{trans} only [13]. In this paper, we aim to minimize user wait time, also called response time t_{res} . This is the time from when the user starts uploading their data to when the results are ready to be returned or downloaded. We do not take into account the time taken to transfer the results from the cloud to the user because in our context, the size of the results is often much smaller than that of the input data.

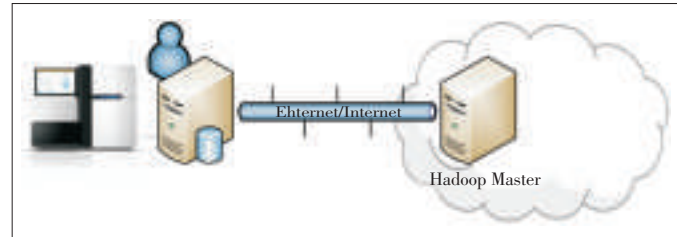
Many people argue that we should only focus on t_{proc} because the data only needs to be transferred once (at most). Nevertheless, in research on large NGS data, alignment or mapping tools—which deal directly with large data in the pipeline mentioned in [14]—are usually run only once or twice because of lengthy runs and high cost. Consequently, it is crucial to minimize t_{res} .

Movement of data from the user to the data center cannot be avoided; therefore, we propose streamlined processing of the data. Our approach involves splitting the user data into smaller parts that are streamlined to the data center. As soon as the first part arrives, the system starts processing it and keeps receiving subsequent parts. After all the parts have been processed, the system merges the results and sends them back to the user.

We implemented a prototype of SPBD and evaluated it with two MapReduce applications: WordCount and metagenomic. The results showed that SPBD improves t_{res} of WordCount by 34% and improves t_{res} of the metagenomic application by 31% for 32 GB of input data.

In this paper, we try to minimize t_{res} when using MapReduce-style software to analyze a large NGS data set in a cloud

with a Hadoop framework. Our proposed system is shown in Fig. 1.



▲ Figure 1. Overview of proposed system.

In section 2, we describe our approach in detail and formulate problems. In section 3, we describe the design and implementation of SPBD. In section 4, we give the results of our experiments on SPBD. In section 5, we discuss future research directions.

2 Our Approach

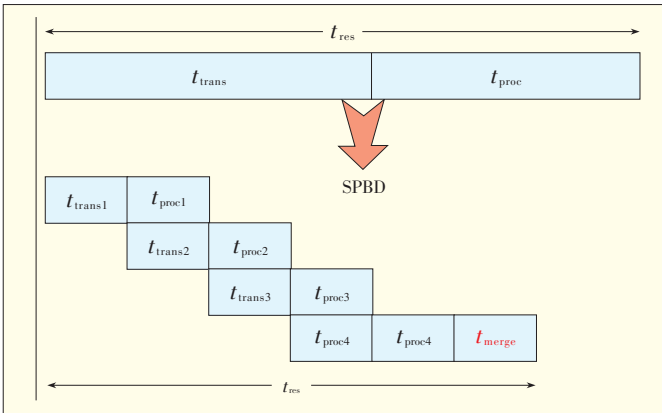
We propose adapting the pipeline processing model for the CPU to our scenario. In the pipeline model, a CPU instruction is equivalent to completely processing a partition of data in our context. This processing is denoted P_i , where i is the number of the partition. In this paper, the terms partition and chunk can be used interchangeably. Each P_i only has transmission and processing stages, not many stages, as in a CPU instruction. All stages of a CPU instruction have the same clock cycle, that is, the time taken to complete a stage. In our proposal, this time is arbitrary.

Instead of waiting until after all the data has been moved to the cloud before analyzing it, we start analyzing the chunks as soon as they arrive. While the first chunk is being analyzed, the second chunk is being received. When the first chunk has been processed, the second chunk is ready to be processed. While processing the second chunk of data, the next chunk is received. This continues until the last chunk has been processed. Depending on the application, an additional merge process can be run after the last chunk has been processed or as soon as the first chunk has been processed. The output results can be merged after all chunks have been processed. Although defined by application developers, such a merge process is usually simple in practice (Fig. 2). The time taken to execute the merge process is denoted t_{merge} .

How large, then, should each chunk be to achieve an optimal t_{res} ? Or how many partitions should the input data be divided into? To answer these questions, we need to compute t_{res} . In Fig. 2, all the stages are the same size; however, in practice, this is not necessarily the case. In the overlapping periods, where receiving and processing occurs at the same time, any stage that requires more time to execute would be used to compute t_{res} . Consider two possible cases: $t_{trans} > t_{proc}$ and $t_{trans} \leq t_{proc}$. For simplicity, we assume that all partitions are the same

SPBD: Streamlining Big-Data Processing in Cloud Environments

Tung Nguyen, Jingwen Zhang, and Weisong Shi



▲ Figure 2. Streamlined processing.

size, and t_{trans} and t_{proc} of a partition are the same as those of other partitions. The merge period contributes to the calculation of t_{res} only after the last chunk has been processed. All executions (if any) in the merge period that take place before the last chunk has been processed are masked by either the data transmission stage or processing stage.

If $t_{trans} > t_{proc}$ (Fig. 3), t_{res} can be given by

$$\begin{aligned} t_{res} &= t_{trans} + (N-1) \times t_{res} + t_{proc} + t_{merg} \\ &= t_{proc} + N \times t_{trans} + t_{merg} \end{aligned} \quad (1)$$

where N is the number of partitions.

In this case, t_{trans} dominates, so the system always finishes processing all the received data and waits for the next chunk.

If $t_{trans} \leq t_{proc}$ (Fig. 4), t_{res} can be given by

$$t_{res} = t_{trans} + N \times t_{proc} + t_{merg} \quad (2)$$

In this case, the data is ready to be processed, but the system has to wait until the previous chunks have been processed because t_{proc} is longer.

In the cases shown in Fig. 3 and Fig. 4, the next chunk of data is transmitted as soon as the previous one has finished. In general, t_{res} is given by

$$t_{res} = \min(t_{proc}, t_{trans}) + N \times \max(t_{proc}, t_{trans}) + t_{merg} \quad (3)$$

In (3), t_{trans} , t_{proc} and t_{merg} are a function of N . As N increases, both t_{trans} and t_{proc} decrease, but t_{merg} increases. In sum, t_{res} is also a function of N . The problem is now one of simple optimization: Given fixed input data, find N so that t_{res} is at its minimum.

In reality, the problem is how to identify all the components in the equation given that the system only receives data of known size and a data-analysis application.

3 System Design and Implementation

3.1 Design

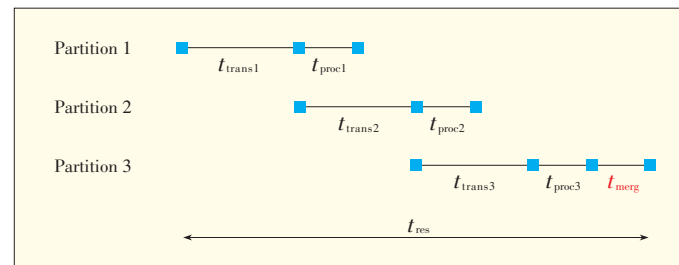
Here, we discuss the proposed SPBD system for realizing

the previously mentioned idea. The inputs for our system are input data, a data-analysis application, and a path to a local folder for storing the output. The data is at the user's site outside the cloud. Like Amazon EMR, there is a pool of data-analysis applications in the cloud for users to choose from. This makes sense because in genomics, researchers often have a collection of data-analysis applications to choose from. These applications include Galaxy and BLAST. Moreover, even if the user wishes to use a special application, it should not be a problem because the time to upload the new application is negligible, and the application can be used many times.

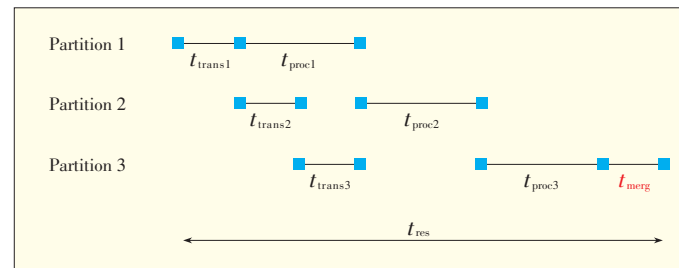
Originally, we assumed that the data-analysis software was a partitionable MapReduce application, that is, a MapReduce job that can process input data in parallel, and chunks of input are independent. In fact, all MapReduce-style jobs are like this because all map tasks are independent. Consequently, our method can be used for any MapReduce job.

The output of our system is also the output of the selected applications. The system takes the input data from the user, transfers it to the cloud, runs the selected applications, and returns the results. The user only needs to specify the path to the local input and output folders. The user also needs to provide the names and other parameters of the applications. The progress of jobs/applications in the cloud can be monitored via a built-in Hadoop website. Once the jobs have finished, the output data is typically written to HDFS or Amazon S3. However, in our system, the output data can be returned to the user if they wish.

There are two approaches to identifying all the terms in (3): static and dynamic. Using the static approach, the functions of the terms in (3) are fixed. If $t_{trans} = 1000/N + 20$, it remains the same all the time. With the dynamic approach, such an as-



▲ Figure 3. t_{res} calculation for $t_{trans} > t_{proc}$.



▲ Figure 4. t_{res} calculation for $t_{trans} \leq t_{proc}$.

sumption is abandoned, and t_{trans} is more realistic. The status of the network is not identical at different time points because of congestion or diverse network traffic. Also, in a virtualized cloud environment, the run time for the same applications with the same-sized input may not be similar at different time points. This dissimilarity occurs because of differences in the statuses of virtual machines, load, and network and also because of differences in the data content itself.

Revisiting (3), t_{proc} , t_{trans} and t_{merg} are not direct functions of N but are, in fact, functions of the size p of the partition. Therefore, we need to change the variables of these terms from p to N (or vice versa). If we assume that all partitions are the same size, and S is the size of all the input data, the relationship between p and N is given by

$$p = S/N \quad (4)$$

If the terms of (3) are substituted into (4), t_{res} becomes a function of S or p .

In practice, t_{merg} is not easily estimated, so we remove it from (3). In our implementation, the input data is partitioned and processed by a specific data-analysis application. After that, the outputs of all partitions go to the merging process to produce the final result. It is therefore difficult to determine the relationship between t_{merg} and S because this relationship depends heavily on the output of the data-analysis applications. In section 5, we show that the t_{merg} forms of different applications are different.

We assume that t_{proc} and t_{trans} are linear with respect to S . Therefore, without loss of generality, we give the following denotations:

$$\begin{cases} t_{\text{trans}} = ap + b \\ t_{\text{proc}} = cp + d \end{cases} \quad (5)$$

In our context, a and c are always greater than 0 because t_{proc} and t_{trans} are monotonically increasing functions of S . The partition size p is also greater than 0. To obtain the minimum t_{res} , we need to know when t_{proc} is greater than, less than, or equal to t_{trans} . This allows us to remove the maximum and minimum functions in (3).

We have the following two cases: $a > c$ and $a < c$. In the former case, (3) becomes

$$t_{\text{res}} = \begin{cases} ap + \frac{dS}{p} + b + Sc & p < \frac{d-b}{a-c} \\ \left(\frac{S}{p} + 1\right)(ap + b) = \left(\frac{S}{p} + 1\right)(cp + d) & p = \frac{d-b}{a-c} \\ cp + \frac{bS}{p} + d + Sa & p > \frac{d-b}{a-c} \end{cases} \quad (6)$$

- When $p = \frac{d-b}{a-c}$, finding the minimum t_{res} is inconsequential because t_{res} is a constant.
- When $p < \frac{d-b}{a-c}$ for $d \geq 0$, $p = \sqrt{\frac{dS}{a}}$, then t_{res} is at a

minimum.

- When $p < \frac{d-b}{a-c}$ for $d < 0$, there is no minimum.

However, this is not likely because we did not consider t_{merg} ,

which is in the form $t_{\text{merg}} = eN + f = \frac{es}{p} + f$ for $e > 0$. In this

case, t_{merg} is a large value. When N increases, t_{merg} increases as well.

- When $p > \frac{d-b}{a-c}$ for $b > 0$, $p = \sqrt{\frac{dS}{c}}$, then t_{res} is at a minimum.

In the latter case, (3) becomes

$$t_{\text{res}} = \begin{cases} cp + \frac{bS}{p} + d + Sa & p < \frac{d-b}{a-c} \\ \left(\frac{S}{p} + 1\right)(ap + b) = \left(\frac{S}{p} + 1\right)(cp + d) & p = \frac{d-b}{a-c} \\ ap + \frac{dS}{p} + b + Sc & p > \frac{d-b}{a-c} \end{cases} \quad (7)$$

Similar to the former case, we can derive p in order to minimize t_{res} .

After obtaining p for each subcase in equation (7), we can calculate t_{res} for each subcase and compare these t_{res} values to obtain the overall minimum t_{res} .

At this stage, we can fine-tune p by monitoring the network status and analysis progress. For example, the network keeps changing over time, so t_{trans} for the incoming partition is most likely different from t_{trans} for the previous partitions.

3.2 Implementation

SPBD is implemented in Java using a client-server model. The server is the portal of the cloud, and the client keeps sending data. The server decides when its buffer has enough data (the size of the partition) and begins processing so that an optimal overall t_{res} is achieved. The problem is deciding how many partitions/chunks the input data needs to be divided into. In other words, when should the system start processing a chunk?

The client provides an interface so that a user can specify their job (application names and parameters). The path to the local input data is included in the parameters. The job specification is similar to that of Amazon EMR Ruby Client. Other information, such as the IP address of the server, credentials and access keys, is specified in a configuration file. Before sending the large input file, the client sends the job specification and data size to the server. After receiving an acknowledgement from the server, the client starts transferring the input data using TCP.

We implement the server using a static or dynamic approach. With the static approach, we run all available applications in the pool offline. The applications are run with different sets of data of differing size, and using linear regression, all the functions in (3) are identified. In section 4, we show that the assumption of linearity holds in practice because the R-squared

SPBD: Streamlining Big-Data Processing in Cloud Environments

Tung Nguyen, Jingwen Zhang, and Weisong Shi

values are very close to 1. This indicates a good-fitting model.

With the dynamic approach, the terms in (3) are identified online when executing the job. At the server side, the input data is divided into two parts that are nearly the same size. The first part is used to learn t_{proc} and t_{trans} . The second part is processed in a streamlined way using the computed optimal size.

The administrator (at the server side) specifies a number of samples when starting SPBD. This number is used to identify different sample sizes in our linear regression process. The size of the first sample partition is denoted $init_size$ and is given by

$$init_size = \frac{S}{2^{SAMPLE_NUM-1}-2} \quad (8)$$

The size of the i th sample partition is double that of the $(i-1)$ th sample partition. Algorithm 1 is used in the receiving function at the server. With this approach, it is difficult to learn t_{merge} because the sizes of partitions are different.

In addition, we fine-tune SPBD by monitoring t_{proc} and t_{trans} after computing the optimal partition size. If the optimal partition size is obtained when $t_{proc} = t_{trans}$ for $|t_{proc} - t_{trans}| > \text{thresh}$, then we update the optimal size accordingly. For example, if $a > c$ and t_{proc} is greater than t_{trans} , we decrease the size of the partition.

In this implementation, SPBD does not send the result back to the user because the result is simple. When using Amazon EMR, the results are stored in S3, and there are many easy-to-use graphical tools that can be used to interact with S3 as if it were a local file system.

4 Evaluation

We evaluate SPBD on WordCount and a metagenomic application. The former is a simple and typical MapReduce application that is useful in gaining insight into SPBD. It also shows that our approach is not necessarily limited to genomics. The latter is chosen from many genomic applications because it deals with large data sets. In addition, its outputs are fixed-sized matrices, which shows that the merge process is very different for different applications.

Our testbed configurations are described in Table 1. The local network speed is 100 Mbps, and the computing cluster runs Hadoop 0.20.203 on the Eucalyptus platform.

To use SPBD, the developer needs to write 76 lines of code for the merge part of WordCount and 152 lines of code for the merge part of the metagenomic application. It is not difficult to

▼ Table 1. Testbed configuration

Type	Number	CPU (GHz)	Memory (GB)	HDD	OS
Client	1	3.6	24	72 TB	Solaris
Master	1	2.4	16	850 GB	CentOS
Slaves	4	2.4	16	> 320 GB	CentOS

write these parts, and most likely, involves only a slight modification of the reduce parts in the original applications.

Algorithm 1. SPBD Algorithm

Input: size of the input data S , $init_size$ and $SAMPLE_NUM$

Output: none

$partition_size = init_size$;

$accumulated_size = 0$;

repeat

bytesRead = Read from socket to buffer;

write buffer to HDFS;

$accumulated_size += bytesRead$;

$partition_num = 0$;

while $accumulated_size \geq partition_size$ **do**

record t_{trans} for the received partition;

create and submit a new job asynchronously to

Hadoop to process the received partition;

if $partition_num < SAMPLE_NUM - 1$

then $partition_size *= 2$;

end

else if $partition_num == SAMPLE_NUM - 1$ **then**

Start a new thread to compute the

optimized size for a partition;

the optimized size will be used to update
the $partition_size$

end

$accumulated_size = 0$

$partition_num++$;

end

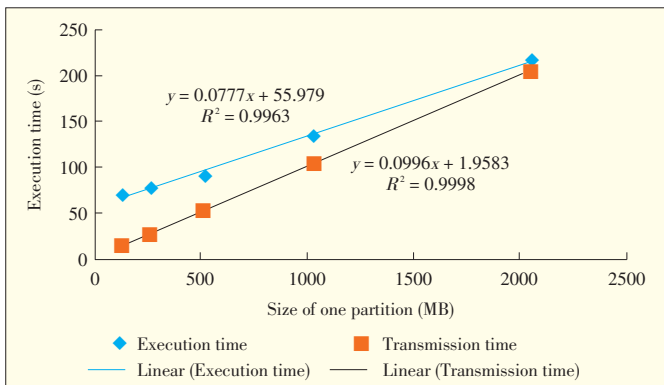
until $BUFFER_SIZE \neq bytesRead$;

4.1 Detailed Experimental Results: WordCount

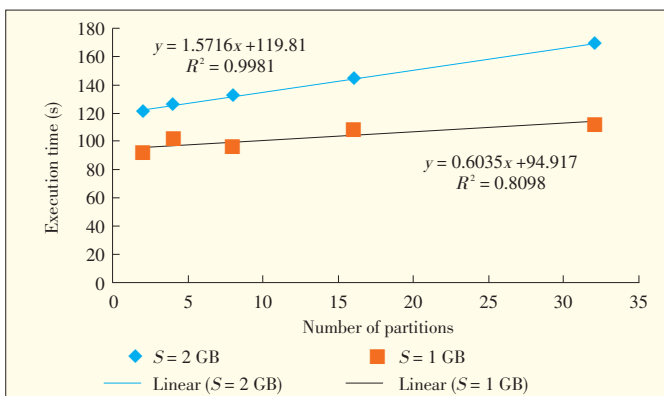
We intentionally did more experiments with WordCount because it is more popular than a metagenomic application. WordCount is a MapReduce application that is included in the example jar file distributed with Hadoop. The application is used to count the number of occurrences of all words in an input file. The input data used in our experiments is the data dump from Wikimedia [2].

Experiments are performed to show the relationship between p and t_{proc} and to show the relationship between partition size and t_{trans} (Fig. 5). The same input data is used. The linear regression functions and R-square values are shown for t_{proc} and t_{trans} . The size of the input data is between 5 GB and 2 GB. Fig. 5 shows that if the same input data (i.e. the same content) is partitioned into different sizes, both t_{proc} and t_{trans} of each partition can be modeled with linear regression. At each data point in Fig. 5, the input data is cut into partitions of the same size. The linear functions obtained with particular input data may differ from those obtained with other input data.

Fig. 6 shows t_{merge} for 1 GB and 2 GB input data and varying



▲ Figure 5. t_{proc} , t_{trans} and their linear regression functions for $S = 2$ GB.



▲ Figure 6. Linear regression functions of the merge process for $S = 1$ GB and $S = 2$ GB.

N . When S remains the same and N increases, t_{merge} generally increases also. Fig. 7 shows the relationship between S , t_{proc} , and t_{trans} . The content of all input data varies.

In Fig. 7, S is measured in gigabytes whereas in Fig. 5, S is measured in megabytes. Therefore, the linear function parameters are significantly different. Also, the input data content used for Fig. 5 is different from that used for Fig. 7. The values obtained in the experiments shown by Fig. 5 and Fig. 7 are the averages of several runs.

Our experiments confirmed that the relationship between S , t_{proc} and t_{trans} is linear regardless of the content of the data. The relationship between p , t_{proc} and t_{trans} is also linear regardless of the content of the data. Also, t_{proc} and t_{trans} , which derive from linear regression, are different for different runs. Consequently, we should not use the same computed optimized p between different runs.

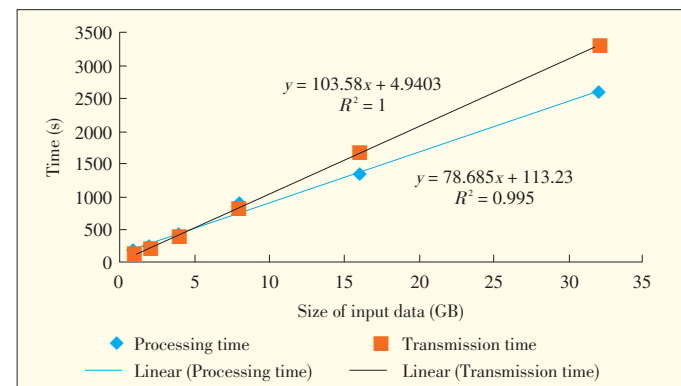
After verifying the linear assumption, we experimented with our SPBD system and mimicked a real scenario. The input data is stored in the client machine, which acts as the storage server at the client side (Table 1). This machine also contains SPBD client code. The master node runs the SPBD server code and accepts job submissions. This master node is also the master node of Hadoop. The remaining nodes are computing nodes. In reality, both the master and slaves are in the cloud,

and the master and slaves are connected to the client through the internet. However, in our experiments, the master, slaves, and client are all on the same local network. This does not affect the applicability of SPBD because SPBD learns the functions through measurement.

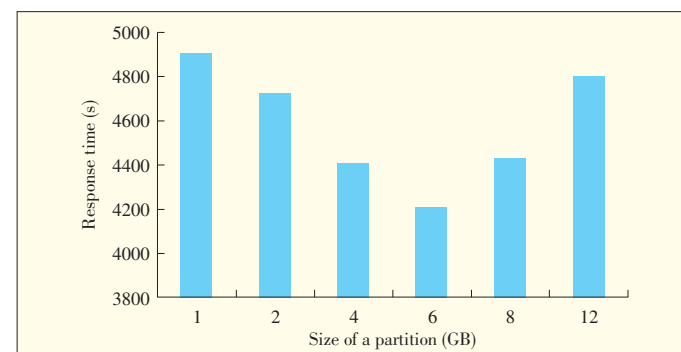
Using the static approach, we evaluate SPBD with varying p . In this experiment, we ran WordCount with 32 GB of data. The results show that varying p for the same input data significantly changes t_{res} (Fig. 8). In addition, as p increases, t_{res} decreases at first and then increases. This suggests that there is an optimal t_{res} with respect to an appropriate p .

To study the bandwidth between the client and the Hadoop master node, we ran WordCount on our local Hadoop testbed, and 16 GB of data was transferred from different machines in the network. The results are shown in Table 2. Bandwidth does affect the performance of the system but not very significantly. SPBD outperforms the traditional approach in all cases.

To determine the benefit of SPBD, we used the dynamic approach with $SAMPLE_NUM = 3$. The NO-SPBD configuration



▲ Figure 7. Linear regression functions for WordCount, when S varies.



▲ Figure 8. t_{res} of WordCount when static approach is used and p is varied.

▼ Table 2. SPBD for different network bandwidths

Bandwidth (Mbit/s)	SPBD (s)	NO-SPBD (s)
89.9	2153	2865
67.3	2247	3021
5.00	2325	3105

SPBD: Streamlining Big-Data Processing in Cloud Environments

Tung Nguyen, Jingwen Zhang, and Weisong Shi

is computed by making p the same as S , and the SPBD values already include t_{merge} . In other words, the outputs for the two configurations are the same (Fig. 9). As the size of input data increases, SPBD provides greater improvement (Fig. 9). For example, when $S = 32$ GB, t_{res} with SPBD is only 66% that of the existing approach.

Finally, we experimented with Amazon EMR. The results show that the assumption of a linear relationship between t_{trans} and S holds. However, we do not discuss it here.

4.2 Detailed Experimental Results: The Metagenomic Application

Metagenomics is the study of genetic material sampled directly from the habitats of microorganisms [16]. In our experiment, we used a metagenomic application developed by a bioinformatics research group [4]. The input data for this application was generated using MetaSim [18]. Availability of input data was also an important reason to select this tool for our experiment.

As with the previous experiments on WordCount, we verify the linear relationship between input data and t_{proc} and the linear relationship between input data and t_{trans} . We study the merging process and determine the benefit of SPBD.

Fig. 10 shows measured t_{proc} and t_{trans} when S varies. Linear regression is also applied to the data. As in the WordCount experiments, the assumption of linearity holds. Table 3 shows t_{merge} for 1 GB of data and varying N . This time, t_{merge} is almost constant, unlike in the WordCount experiments. This is because the output of the metagenomics application is only a small number of small matrices, and these matrices are independent of S . Therefore, combining the matrices in parallel takes only small and almost constant amount of time. This experiment also demonstrates that t_{merge} is difficult to estimate and depends heavily on particular applications.

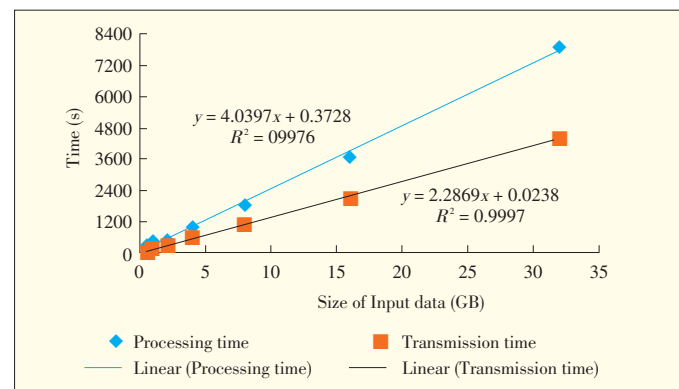
Last, we ran a modified metagenomics application with SPBD and without SPBD and on different data sets. For $S = 32$ GB, t_{res} for SPBD improved 31% compared to t_{res} for NO-SPBD.

When S is less than 8 GB, SPBD performance is a little

worse than that of NO-SPBD (Fig. 11). This indicates that SPBD is only beneficial with big data. If S of an application is less than a certain value, it is good to upload all the input data to the cloud and processing it afterward. We determine that this value is 1 GB.

5 Conclusion

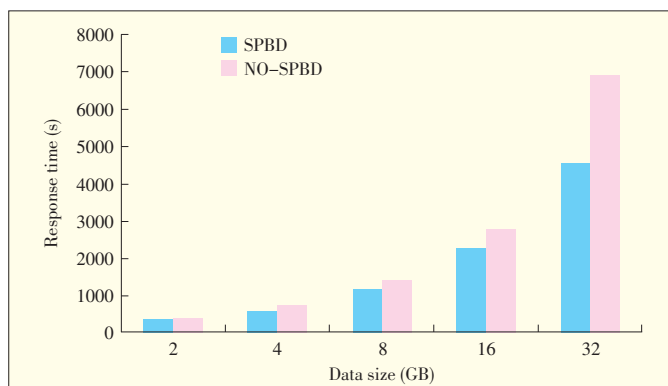
In this paper, we have discussed state-of-the-art techniques for analyzing large data sets in genomics. These techniques involve combining a cloud with a parallel processing framework, such as MapReduce or Dryad. We proposed, implemented, and evaluated SPBD, which is a system that automatically transfers and processes large data in the MapReduce cloud. Our experimental results show that SPBD significantly



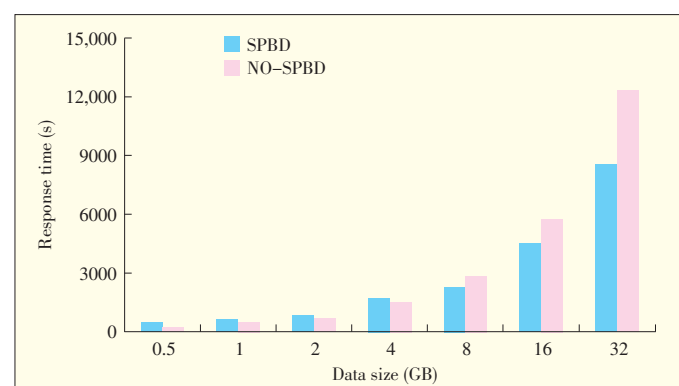
▲ Figure 10. Measured t_{proc} and linear functions of the metagenomic application when S varies.

▼ Table 3. t_{merge} for $S = 1$ GB

Partitions	t_{merge} (s)
2	30
5	31
25	30
50	31
100	30



▲ Figure 9. Benefit of SPBD in WordCount when the dynamic approach is used.



▲ Figure 11. Benefits of SPBD in the modified metagenomic application when the dynamic approach is used.

SPBD: Streamlining Big-Data Processing in Cloud Environments

Tung Nguyen, Jingwen Zhang, and Weisong Shi

improves t_{res} when a large data set is analyzed.

We took an empirical approach to deriving the coefficients needed to compute t_{res} , and we continued refining the parameters after determining the optimal partition size. Our solution is resilient to changes of system parameters, such as load or network traffic. Although we did not experiment with other jobs or interferences in the system, SPBD should also benefit a practical multiuser system. In future work, we plan to extend SPBD to handle more than one input file, and we plan to improve the linear model by using, for example, piecewise regression. We also plan to test SPBD with more applications or, better yet, with a benchmark for large data. Instead of TCP, we might also apply GridFTP or other state-of-the-art data transmission protocols to SPBD.

References

- [1] *System of Illuming* [Online]. Available: <http://www.illumina.com/systems.illum>
- [2] *Novocraft Technologies* [Online]. Available: <http://www.novocraft.com/main/index.php>
- [3] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [4] M. Isard, M. Budi, Y. Yu, A. Birrell and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 3, pp. 59–72, 2007.
- [5] M. Schatz, "CloudBurst: highly sensitive read mapping with MapReduce," *Bioinformatics*, vol. 25, no. 11, p. 1363, 2009.
- [6] B. Langmead, M. Schatz, J. Lin, M. Pop and S. Salzberg, "Searching for SNPs with cloud computing," *Genome Biology*, vol. 10, no. 11, p. R134, 2009.
- [7] X. Qiu, J. Ekanayake, S. Beason, T. Gunarathne, G. Fox, R. Barga and D. Gannon, "Cloud technologies for bioinformatics applications," in *Proc. of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, New York, NY, USA, 2009.
- [8] T. Nguyen, W. Shi and D. Ruden, "CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping," *BMC Research Notes*, vol. 4, no. 1, p. 171, 2011.
- [9] S. Angiuoli, M. Matala, A. Gussman, K. Galens, M. Vangala, D. Riley, C. Arze, J. White, O. White and W. F. Fricke, "CloVR: A virtual machine for automated and portable sequence analysis from the desktop using cloud computing," *BMC Bioinformatics*, vol. 12, no. 1, p. 356, 2011.
- [10] K. Krampis, T. Booth, B. Chapman, B. Tiwari, M. Bica, D. Field and K. Nelson, "Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community," *BMC Bioinformatics*, vol. 13, no. 1, p. 42, 2012.
- [11] B. Allen, J. Bresnahan, L. Childers, I. Foster, G. Kandaswamy, R. Kettimuthu, J. Kordas, M. Link, S. Martin, K. Pickett and S. Tuecke, "Software as a service for data scientists," *Commun. ACM*, vol. 55, no. 2, pp. 81–88, 2012.
- [12] *Pentaho Corporation Pentaho for Big Data* [Online]. Available: <http://wiki.pentaho.com/display/BAD/Pentaho+Big+Data+Community+Home>
- [13] H. Monti, A. Butt and S. Vazhkudai, "CATCH: A Cloud-Based Adaptive Data Transfer Service for HPC," in *Parallel Distributed Processing Symposium (IPDPS)*, 2011 IEEE International, 2011.
- [14] Uwe R hm and J. Blakeley, "Data Management for High-Throughput Genomics," in *Proc. CIDR*, 2009.
- [15] *Wikimedia Dump* [Online]. Available: <http://dumps.wikimedia.org/enwiki/20120502/>
- [16] J. C. Wooley, A. Godzik and I. Friedberg, "A Primer on Metagenomics," *PLoS Comput Biol*, vol. 6, no. 2, p. e1000667, 02 2010.
- [17] "Computational Biology & Data Mining," [Online]. Available: <http://www.cs.wayne.edu/dzhu/home.html>
- [18] D. C. Richter, F. Ott, A. F. Auch, R. Schmid and D. H. Huson, "MetaSim – A Sequencing Simulator for Genomics and Metagenomics," *PLoS ONE*, vol. 3, no. 10, p. e3373, 10 2008.

Manuscript received: April 30, 2013

Biographies

Tung Nguyen (nguyen@i-a-i.com) is a research scientist at Intelligent Automation Inc. He plays a key role in many projects on data-intensive distributed processing, cloud computing, and massive data analysis using topological features. Dr. Nguyen received his PhD degree from Wayne State University in 2012. He received his BS and MS degrees in computer science and engineering from Ho Chi Minh City University of Technology, Vietnam, in 2001 and 2006. His research interests include green computing, cloud computing, data-intensive computing, and application of cloud computing to life sciences. He has published several papers on computer science and bioinformatics and has been published in the proceedings of OSDI and in NPC, SUSCOM, and BMC Frontiers Genetics journals. He has also been a peer reviewer at many conferences, including Euro-Par and CollaborateCom. His homepage is <http://www.cs.wayne.edu/tung/>.

Jingwen Zhang (jingwen.zhang@wayne.edu) received her BS degree in computer science from Xidian University, China. She is currently a PhD student in the Department of Computer Science, Wayne State University. Her research interests include cloud computing and big-data analysis.

Weisong Shi (weisong@wayne.edu) is an associate professor of computer science at Wayne State University. He received his BS degree in computer engineering from Xidian University in 1995. He received his PhD degree in computer engineering from the Chinese Academy of Sciences in 2000. His research interests include computer systems, mobile computing, and cloud computing. Dr. Shi has published 120 peer-reviewed journal and conference papers and has an H-index of 24. He has been the program chair and technical program committee member of numerous international conferences, including WWW and ICDCS. In 2002, he received the NSF CAREER award for outstanding PhD dissertation (China). In 2009, he received the Career Development Chair Award of Wayne State University. He has also won "Best Paper Award" at ICWE'04, IPDPS'05, HPCChina'12, and IISWC'12.

ZTE Cloud Radio Solution to Usher in New Era of High-Performance LTE Networks

ZTE showcased its innovative cloud radio solution for 4G network optimization at the Mobile Asia Expo in Shanghai.

ZTE's cloud radio solution comprises cloud scheduling and cloud coordination modules. These allow an operator to build a high-performance LTE network, deliver improved user-experience, and resolve key engineering challenges in 4G network deployment. ZTE's cloud scheduling module has a central scheduler that manages network resources in real time and unifies network scheduling. The cloud coordination module allows seamless, borderless coordination across the whole network and improves user experience. Cloud scheduling and cloud coordination allow coordination at the cell and user levels respectively. Dual-level coordination helps operators build smooth LTE networks. (ZTE Corporation)

A Hadoop Performance Prediction Model Based on Random Forest

Zhendong Bei, Zhibin Yu, Huiling Zhang, Chengzhong Xu, Shenzhong Feng, Zhenjiang Dong, and Hengsheng Zhang

DOI: 10.3969/j.issn.1673-5188.2013.02.006

<http://www.cnki.net/kcms/detail/34.1294.TN.20130701.1524.001.html>, published online July 1, 2013

A Hadoop Performance Prediction Model Based on Random Forest

Zhendong Bei¹, Zhibin Yu¹, Huiling Zhang¹, Chengzhong Xu^{1,2}, Shenzhong Feng¹, Zhenjiang Dong³, and Hengsheng Zhang³

(1. Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China;

2. Wayne State University, Detroit, Michigan 48202, USA;

3. Cloud Computing and IT Institute of ZTE Corporation, Nanjing 210012, China;)

Abstract

MapReduce is a programming model for processing large data sets, and Hadoop is the most popular open-source implementation of MapReduce. To achieve high performance, up to 190 Hadoop configuration parameters must be manually tuned. This is not only time-consuming but also error-prone. In this paper, we propose a new performance model based on random forest, a recently developed machine-learning algorithm. The model, called RFMS, is used to predict the performance of a Hadoop system according to the system's configuration parameters. RFMS is created from 2000 distinct fine-grained performance observations with different Hadoop configurations. We test RFMS against the measured performance of representative workloads from the Hadoop Micro-benchmark suite. The results show that the prediction accuracy of RFMS achieves 95% on average and up to 99%. This new, highly accurate prediction model can be used to automatically optimize the performance of Hadoop systems.

Keywords

big data; cloud computing; MapReduce; Hadoop; random forest; micro-benchmark

1 Introduction

The MapReduce programming model is widely used in big-data applications because it is simple to program and can handle large data sets. A popular open-source implementation of MapReduce is Apache Hadoop, which has been used for web indexing [1], machine learning [2], log file analysis [3], financial analysis [4], and bioinformatics research [5].

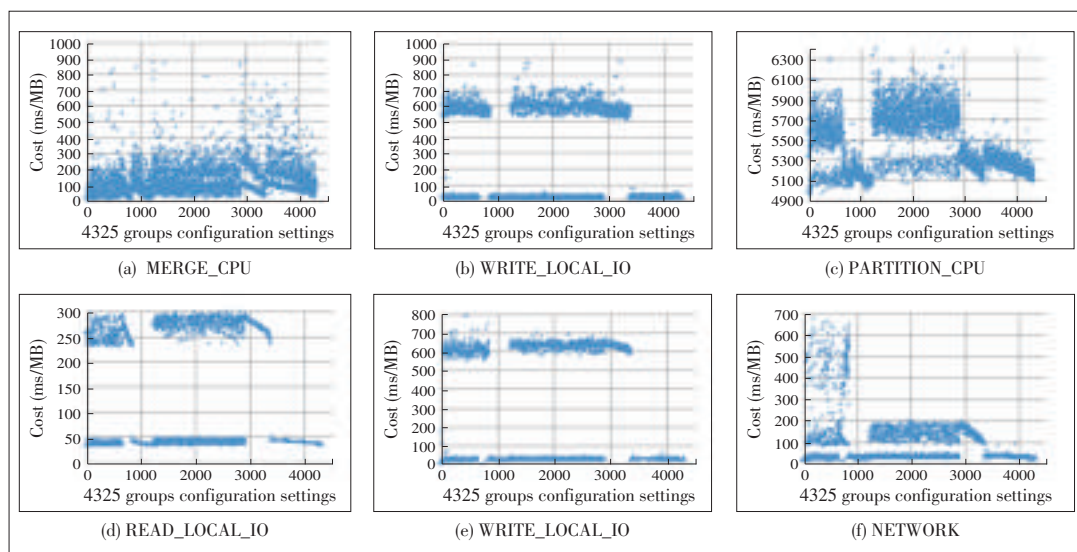
With Hadoop, a programmer needs to manually tune up to 190 parameters to ensure high system performance. However, without in-depth knowledge of the Hadoop system, the programmer may find such a task tedious and may even seriously degrade system performance. This issue has been confirmed by many researchers [6]–[9].

It is therefore desirable to automatically tune the configuration parameters. To this end, a performance prediction model based on historical observation is required. The key to improving performance is to use a highly accurate model with low run-time overhead. Many researchers have tried to construct such a model. In [10], a set of cost-based mathematical functions is used to estimate the fine-grained run time of phases within the map and to reduce tasks when a job is executed. In [6], a coarse-grained SVM regression model is used to estimate the completion time of jobs that belong to a cluster. This estimation

depends on the allocation of resources, parameter settings, and size of input data.

However, these models are not accurate enough because of their assumptions about cluster node homogeneity and over-simplifications. For example, the local and remote CPU and I/O costs during each phase of executing a MapReduce job differs according to the Hadoop parameter settings, even in homogenous nodes in a cluster. If we define operational cost as microseconds per megabyte, variations in parameter settings give rise to variations in operational cost (Fig. 1). Also, Hadoop involves complicated multiphase processing. As well as containing map and reduce phases, a MapReduce workflow also contains fine-grained phases such as read, collect, spill, merge, shuffle, sort, and write. Each phase performs finer grained operations with different costs. For example, a spill phase involves combining, compression, and writing. All the above factors may be intertwined to contest underlying computing resources, such as CPU and I/O. In such a context, if a model is overly simplistic, it is definitely inaccurate.

We propose a model to predict the performance of a Hadoop system. This model is based on fine-grained operations and uses a random forest algorithm. Random forest is a recently developed non-parametric regression model. Unlike traditional regression models, random forest combines tree predictions so that each tree depends on the values of a random vector sam-



▲ Figure 1. Cost of representative operations with 4325 random parameter settings.

pled independently. This random vector has the same distribution for all trees in the forest [11]. We use random forest for two reasons: 1) it does not make linear assumptions between parameters (in our case, the configuration parameters), and 2) it can deal with a large number of configuration parameters, even when there are complex interactions.

In our proposal, a lightweight performance profiler captures the time taken to execute tasks and the size of the data processed in each phase. We construct a fine-grained model for predicting the performance of a Hadoop system. This model is used to accurately estimate phase-level performance under various workloads and without assuming cluster nodes are homogenous. We evaluate RFMS under different workloads generated by Hadoop Micro-benchmark suite. The results show that RFMS provides prediction accuracy of 95% on average and up to 99%.

In section 2, we describe related work. In section 3, we introduce our approach based on random forest. In section 4, we describe our experimental methodology. In section 5, we provide experimental results and describe some applications of our approach. Section 6 concludes the paper.

2 Related Work

2.1 Analyzing MapReduce Performance

Analyzing the performance of a MapReduce distributed system is challenging because a system potentially comprises several thousand of programs running on thousands of machines. Low-level performance details are hidden from users by using a high level dataflow model [9]. Several approaches have been taken to understand user-defined workload behavior in a MapReduce system [12]–[14]. With these approaches, information collected from previous job execution logs is used to iden-

tify performance bottlenecks. For example, Hadoop job history files are often used to analyze performance bottlenecks in a Hadoop system.

Log files are often not exposed to developers, and this means that system performance often not optimal. In [9], logs are leveraged through automatic log analysis to improve performance. Such an approach can be used to show the dataflow breakdown of the map/reduce phases for various jobs, but human involvement is still needed

for hotspot detection.

Performance analysis based on dataflow appears to be suitable for Hadoop; however, the focus is on monitoring the dataflow process, and the user needs to identify possible bottlenecks in a Hadoop cluster. Furthermore, a dataflow approach cannot be used to evaluate performance when configuration parameters are randomly set. Our model is based on workflow analysis and uses a random forest to predict the running time of each phase. This is a precise model that can be used to guide the setting of configuration parameters.

2.2 Automatic Configuration

Various approaches have been taken to automatically configure distributed data processing systems [15]–[18]. The pay-per-use utility model of cloud computing creates new opportunities to deploy the MapReduce framework. However, choosing which configuration parameter settings will result in high performance is complex [19]. Automatic approaches to setting configuration parameters in Hadoop systems have therefore been the focus of attention recently.

The first model for predicting the performance of a Hadoop system with automatically set parameters is described in [10]. The model describes the fine-grained dataflow and cost for phases within map and reduces tasks [10]. However, it assumes that the costs for CPU and I/O in each phase of executing a MapReduce job are the same for all nodes in a cluster. In practice, this assumption is not true (Fig. 1).

AROMA [6] uses a performance model based on support vector machine (SVM) to integrate aspects of resource provisioning and auto-configuration for Hadoop jobs. Based on allocated resources, configuration parameters, and the size of input data, AROMA can estimate the completion time of jobs belonging to a cluster. However, unlike fine-grained cost estimation models, it cannot quantitatively analyze the data processes

A Hadoop Performance Prediction Model Based on Random Forest

Zhendong Bei, Zhibin Yu, Huiling Zhang, Chengzhong Xu, Shenzhong Feng, Zhenjiang Dong, and Hengsheng Zhang

of each phase of a distributed system dataflow.

3 Performance Model Based on Random Forest

In this section, we describe how to capture the execution features of a job. We then use these features to construct a Hadoop performance model based on a random forest.

3.1 Job Characterization

RFMS uses dynamic tools to collect run-time monitoring information without modifying MapReduce workloads on Hadoop. One such tool is BTrace—a safe, dynamic tracing tool that runs on the Java platform and captures the execution features of a MapReduce job [20].

The execution of a MapReduce job can be broken into the map and reduce stages. The map stage can be further divided into reading, map processing, buffer data collecting, spilling, and merging phases. Similarly, the reduce stage can be divided into shuffling, sorting, reduce processing, and writing phases. Each phase is part of the overall execution of the job in Hadoop.

When Hadoop runs a MapReduce job, BTrace traces specified Java classes to generate a task feature file. A feature file is a detailed representation of the task execution that captures information at the phase level. The feature file generally logs execution time, input data size, and output data size. However, the shuffle phase of the reduce stage requires special attention because it contains multiple operations, such as network transferring and merging. To simplify the operations in the phase and better analyze the result, BTrace only records the timing of network transferring, and the timing of merging is added to the next phase, called sort. This phase only has merging operations. When a job is finished, RFMS collects the feature files of all tasks and produces a statistical result of the three characteristics of each phase of a job.

3.2 Building a Performance Model

As described in [10], the performance model for a MapReduce job can be given as

$$Pf = F(p, d, r, c) \quad (1)$$

where F is the performance estimation model for a MapReduce job (p, d, r, c) that runs program p on input data d and uses cluster resources r and configuration parameter settings c .

Because a MapReduce workflow comprises nine phases, each of which is denoted $Phase_s$, the performance model for a whole MapReduce job can be given by

$$Pf = \sum_{s=1}^9 FPhase_s(p, d_s, r, c_s) \quad (2)$$

where $FPhase_s$ is the performance model for each phase, and d_s is the size of the data processed in $Phase_s$. The size of the ini-

tial input data in the map and reduce stage determines the size of d_s . The parameter settings related to $Phase_s$ are given by c_s . The performance model for each phase can be estimated by

$$FPhase_s = FPerTaskPhase_s \times numTotalWaves \quad (3)$$

where

$$numTotalWaves = \lceil numTasks / totalTaskSlots \rceil \quad (4)$$

and

$$totalTaskSlots = numNodes \times numTaskSlotPerNode \quad (5)$$

In (3), $FPerTaskPhase_s$ is the performance of $Phase_s$ when executing a single map or reduce task, and $numTotalWaves$ is the total number of task execution waves.

In (4), $totalTaskSlots$ is the sum of the task slots $numTaskSlotPerNode$ allocated for map and reduce tasks in each node of a cluster. The total number of map or reduce tasks is $numTasks$. In the reduce stage, $numTasks$ is a configurable parameter. In the map stage, the configurability of $numTasks$ depends on the size of input data d . The number of map tasks is given by

$$mapNumTasks = totalDataSize / SplitDataSize \quad (6)$$

where $totalDataSize$ is the size of input data d , and $SplitDataSize$ is the size of the input split of each map task. (The default is 64 MB if uncompressed.)

In (5), $numNodes$ is the total number of nodes in a cluster.

Function (3) allows us to estimate $FPerTaskPhase_s$; thus, it is necessary to learn n phase performance models for a workload. These models are used to analyze the workflow so that we can use (2) to evaluate overall performance.

More importantly, by building a performance model for $FPerTaskPhase_s$, we can estimate Pf when the size of the input data is small. In turn, we can predict the workload performance when the size of the input data is larger by calculating $numTotalWaves$ in the map and reduce stage.

In [21], $FPerTaskPhase_s$ is calculated using a set of functions based on a constant cost assumption (a cost-based model). It is assumed that the local and remote CPU and I/O costs per phase in executing a MapReduce job are equal across all the nodes in a cluster with the same hardware resource. However, varying the configuration parameters may vary these costs and prove this assumption false.

RFMS addresses this problem by using a machine learning technique to construct $FPerTaskPhase_s$ models for different phases. RFMS uses the random forest (RF) regression model to estimate $FPerTaskPhase_s$ of a workload with varying configuration parameters. RF methodology is precise when there are regression problems and performs consistently well [11]. Applications that use RFs demonstrate that RF is one of the best of available methodologies for modeling the performance of complex Hadoop workloads in the cloud environment [22]. RFMS

can produce importance measures for each variable [23]. These measures indicate which variables have the strongest effect on the dependent variables being investigated. RFMS can capture the time taken to execute the phases of Hadoop workloads with varying configuration parameters. The ten configuration parameters that are important to overall performance in a Hadoop system are listed in Table 1. These parameters are used as feature candidates to train RFMS. For given input data, the size of the uncompressed split data for each map task, given by $USDFMT$, does not change in the map stage. Thus, the size of the split data is not used to train RFMS. However, the input data (shuffle bytes) for each reduce task changes according to the `mapred.reduce.tasks` parameter. Furthermore, the size of the shuffle bytes significantly affects the execution time of a reduce task. Therefore, the shuffle bytes for each reduce task *ShuffleByteEachTask* should be used as a feature candidate for the reduce stage. *ShuffleByteEachTask* is given by

$$ShuffleByteEachTask = \frac{USDFMT \times Selectivity_{Mstage} \times mapNumTask}{reduceNumTask} \quad (7)$$

where

$$Selectivity_{Mstage} = Selectivity_{Map} \times Selectivity_{Combine} \times Selectivity_{Compress} \quad (8)$$

and

$$Selectivity_{operation} = \left(\sum_{i=1}^n OperationSelectivityOfTask_i \right) / n \quad (9)$$

Selectivity can be defined as the statistical ratio of the output size to input size for a stage or operation in a workload. In (7), $Selectivity_{Mstage}$ is the map stage selectivity. Because map, combine, and compress operations reduce the input data size, $Selectivity_{Mstage}$ can be calculated using (8). In (8), $Selectivity_{Map}$, $Selectivity_{Combine}$, and $Selectivity_{Compress}$ are operation selectivity. These three selectivities are determined by related operation, and they can be calculated using (9). In (9), $OperationSelectivityOfTask$ is the ratio of output size to input size of an operation in a task, and n is the number of map tasks in a given workload. $OperationSelectivityOfTask$ can be calculated using the data captured by our profiler in a feature file. The combine and compress operations are optional, and the value of $Selectivity_{Combine}$ and $Selectivity_{Compress}$ is set at 1 if the user does not specify these operations.

An *RF* can be defined as a collection of tree-structured predictors [11]:

$$RF = \{h_k(X, \theta_k), k = 1, \dots, N_{trees}\} \quad (10)$$

where h_k is the k th individual tree, $h_k(\cdot)$ is the tree's prediction, and N_{trees} is the number of trees. The samples of the total training set are given by $X = \{Fc_k, Pt_k\}$, where $k = 1, \dots, N_{samples}$. The predictor features are given by Fc_k , and the phase time is given by Pt_k . The total training set is divided into two independent subsets: one to train the predictor h_k and the other to test

▼ Table 1. Configuration parameters selected for testing

Configuration Parameters	Default	Test Range
<code>io.sort.factor</code>	10	10–100
<code>mapred.job.shuffle.merge.percent</code>	0.66	0.2–0.9
<code>mapred.output.compress</code>	false	true or false
<code>mapred.inmem.merge.threshold</code>	1000	10–1000
<code>mapred.reduce.tasks</code>	1	1–100
<code>io.sort.spill.percent</code>	0.8	0.5–0.9
<code>mapred.job.shuffle.input.buffer.percent</code>	0.7	0.0–0.8
<code>io.sort.record.percent</code>	0.05	0.01–0.5
<code>io.sort.mb</code>	100	(0.25–0.65)* taskMem (<code>mapred.child.java.opts</code>)
<code>mapred.compress.map.output</code>	false	true or false

the predictor's accuracy. In (10), θ_k are random variables. The nature and dimensionality of θ_k depends on randomness in the construction of N_{trees} trees. This randomness may be caused by the random selection of N_{trees} training records drawn from X with replacement. It may also be caused by m_{try} , the random number of different features tried at each split of a tree.

In RF regression, it is difficult to define the predictor features Fc_k that allow the base predictors to be trained to accurately predict the target on the out-of-bag data. When selecting important features, importance is assessed by replacing each feature with random noise and observing the increase in the mean squared error (MSE) for the out-of-bag validation. The features are then sorted by relative importance, and an important subset of features can be abstracted. The RF is iteratively retrained, and each time, the least important features are removed. In practice, N_{trees} and M_{try} are used to tune the RF model and minimize the MSE. Lower MSE can make the model better fit the training data. N_{trees} and M_{try} are tuned using scripts that iteratively change each parameter one-by-one and regenerate the regression model. The optimum value of M_{try} ranges from 6 to 10, and the optimal value of N_{trees} ranges from 500 to 1000.

We use a stepwise regression model for the data sets collected from our test bed comprising Hadoop nodes. In section 5, we discuss the accuracy of the prediction against the training data and provide out-of-bag accuracy statistics.

4 Experiment Methodology

The experiments are performed on a test bed comprising ten Sugon servers and a gigabit Ethernet network. Each server has a quad-core Intel(R) Xeon(R) CPU E5-2407 0 at 2.20 GHz and 32 GB PC3 memory. The cluster is virtualized by Xen 3.0. We create a pool of virtual machines (VMs) from the virtualized service cluster. Each has eight virtual CPUs and 8 GB memory. We then run the VMs as Hadoop nodes. Each VM uses SUSE Linux Enterprise Server 11 and Hadoop 1.0.4.

We designate one server to host the master VM node and

A Hadoop Performance Prediction Model Based on Random Forest

Zhendong Bei, Zhibin Yu, Huiling Zhang, Chengzhong Xu, Shenzhong Feng, Zhenjiang Dong, and Hengsheng Zhang

use the remaining servers to host the nine slave VM nodes. The master node runs the JobTracker and the NameNode. Each slave node runs both the TaskTracker and the DataNode. Each VM is initially configured with four map slots, four reduce slots, and 300 MB memory per task. The data block is set at 64 MB. The RFMS profile component needs to run on each slave VM node so that BTrace can capture the task execution times. Other components of RFMS can run on a separate VM or standalone machine because RFMS processes the gathered features offline.

We run representative Hadoop workloads, such as TeraSort, to test the precision of RFMS with 10 different configuration parameters (Table 1). We use Hadoop benchmark to produce data of various sizes.

5 Results and Analysis

5.1 The Constant-Cost Assumption

We test the constant-cost assumption by comparing six cost features of TeraSort. Six different configurations were used (Table 2). First, we randomly varied the configuration parameter values up to 4325 times when running TeraSort. The parameters generated within the test range are listed in Table 1. To obtain credible results, we use the profiler in [10] to capture every cost feature. We use the plots of the TeraSort benchmark to show the results. Then, we select six cost features from a total of 12 cost features. These six cost features comprise CPU features (REDUCE_CPU, PARTITION_CPU and MERGE_CPU) and I/O features (READ_LOCAL_IO, WRITE_LOCAL_IO and NETWORK). From 4325 configurations, we select six that have typical cost distributions.

Fig. 2 shows six different values for each cost feature when configuration settings are varied. (Note the log scale on the y-axis.) For example, REDUCE_CPU ranges from 6610 to 9318 ms/MB, and MERGE_CPU ranges from 0.81 to 427 ms/MB. These values have significantly changed, and this indicates that cost features are affected by variations in the configuration settings.

Although we do not show the cost features of other workloads from Hadoop benchmark, from our observations, these features have similar properties to each other. Therefore, we believe the constant-cost assumption is false for the 12 operations in a MapReduce workflow.

Furthermore, we determine whether the inconstant cost value affects the accuracy of performance prediction. Fig. 3 shows the NetworkTransferTime distribution against the cost for NETWORK when transferring 2040 MB data. This distribution is derived from actual measurement. NetworkTransferTime is the time taken to transfer networks in the shuffle phase and is given by

$$\text{Network transfer time} = d\text{ShuffleSize} \times cs\text{NetworkCost} \quad (11)$$

Table 2. The six types of configuration parameter settings

Parameters	Configurations					
	1	2	3	4	5	6
io.sort.factor	31	61	75	14	50	40
mapred.job.shuffle.merge.percent	0.29	0.24	0.61	0.28	0.3	0.89
mapred.output.compress	false	false	true	true	false	true
mapred.inmem.merge.threshold	825	720	717	268	969	469
mapred.reduce.tasks	1	97	79	1	66	2
io.sort.spill.percent	0.5	0.89	0.57	0.72	0.7	0.78
mapred.job.shuffle.input.buffer.percent	0.22	0.26	0.89	0.36	0.23	0.59
io.sort.record.percent	0.01	0.12	0.03	0.27	0.11	0.32
io.sort.mb	96	120	108	131	91	107
mapred.compress.map.output	false	false	true	true	false	true

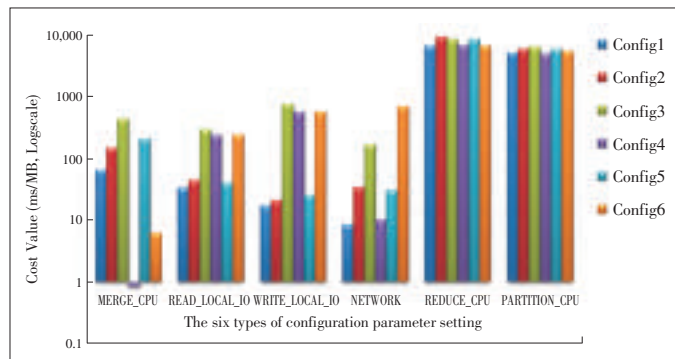


Figure 2. Costs for six types of parameter settings using Terasort benchmark.

where $d\text{ShuffleSize}$ is total shuffle size [21]. In Fig. 3, NetworkTransferTime is significantly affected by inconstant Network-cost. We set $d\text{ShuffleSize}$ at a constant 2040 MB. The total shuffle size is generated by a map phase with 10 GB input data. The cost of the network transfer is $cs\text{NetworkCost}$.

Using (11), we can predict NetworkTransferTime when the configuration parameters are varied. We set $cs\text{NetworkCost}$ at a constant 8.826963634 ms/MB observed with default configuration parameters. In Fig. 4, predicted NetworkTransferTime is plotted against the real measurement of NetworkTransferTime. The predictions shown in Fig. 4 are poor, which suggests that the constant-cost assumption is false.

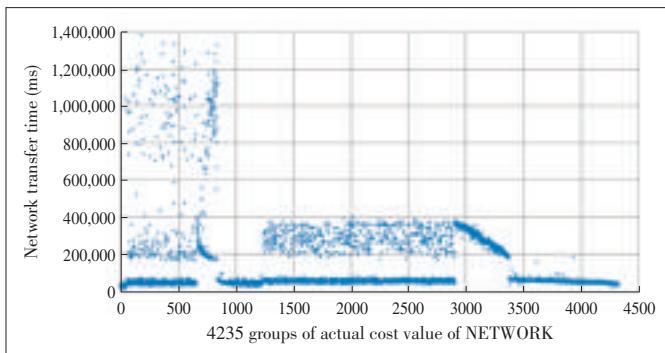
5.2 Accuracy of RFMS

Fig. 5 shows phase timing predicted using RFMS against the measured phase timing for a TeraSort workload with 10G input data. The phase timing is measured in 100 groups of experiments.

In Fig. 5, phase timing predicted using RFMS is similar to the measured phase timing for a MapReduce workflow. Our prediction model is reasonably accurate. Although the predictions for read timing are not particularly accurate, the affected performance range is only between 2000 and 2500, which is

A Hadoop Performance Prediction Model Based on Random Forest

Zhendong Bei, Zhibin Yu, Huiling Zhang, Chengzhong Xu, Shenzhong Feng, Zhenjiang Dong, and Hengsheng Zhang



▲ Figure 3. Network transfer time for 2040 MB of data with inconstant network cost.

low. To quantitatively evaluate the accuracy of our RFMS model, we use the relative error E_r , given by

$$E_r = \left(\sum_{i=1}^n \frac{Pre_i - Real_i}{Real_i} \right) / n \quad (12)$$

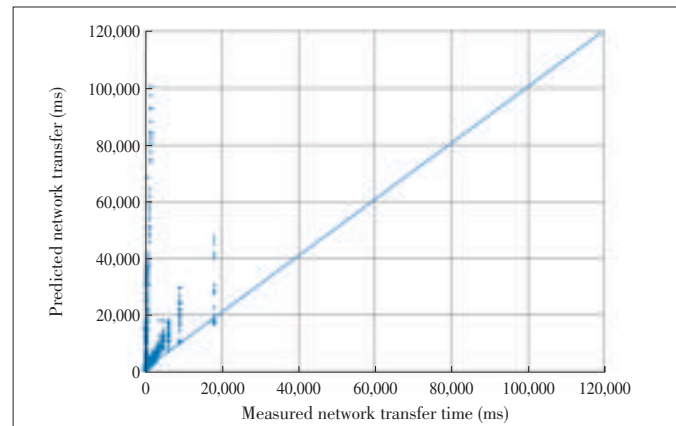
Where Pre_i is the i th value predicted by RFMS, $Real_i$ is the i th value actually measured, and n is the total number of tests. Precisely, we use average relative error of 100 predictions to represent the prediction error rate between each real and corresponding predicted value, selected from different ranges of phase timing distribution.

The error rates for the nine phases that we experimented with were calculated using E_r and are shown in the Table 3. The error rate varies from 0.566% to 7.169%. All are less than 8%, and the average is 5%. This indicates that our RFMS predictions are close to the measured phase timing.

6 Conclusion

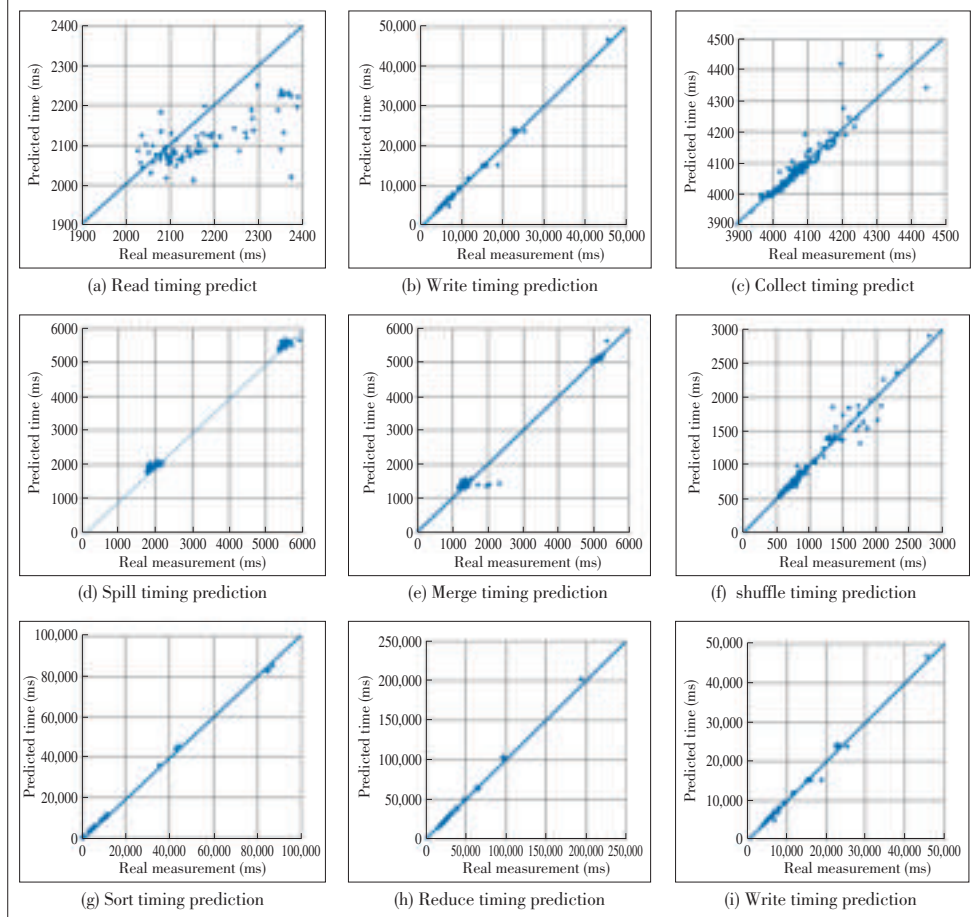
In this paper, we have confirmed that varying the configuration parameter values significantly affects the precision of a cost-based model. Therefore, we have designed RFMS, a phase-based workflow performance analysis model for precisely tuning Hadoop parameters. We focused on non-transparent performance tuning for various Hadoop workflow processes.

RFMS prediction is significantly better than that provided by a



▲ Figure 4. Predicted network transfer time vs. measured network transfer time.

cost-based model. The average accuracy reaches 95%, which is reasonably good. Besides being accurate, our approach has several other advantages in practice. It has an improved, lightweight workload in-depth profiler that collects key task execution information from an unmodified MapReduce/Hadoop workload. It also has a novel dataflow analysis mechanism for Ha-



▲ Figure 5. The best RF model for predicting phase timing.

A Hadoop Performance Prediction Model Based on Random Forest

Zhendong Bei, Zhibin Yu, Huiling Zhang, Chengzhong Xu, Shenzhong Feng, Zhenjiang Dong, and Hengsheng Zhang

▼ Table 3. Relative Error of RFMS

Phase Name	Relative Error (%)
Read	6.053
map	4.865
collect	0.566
spill	1.893
merge	6.583
shuffle	7.169
sort	4.593
reduce	4.353
write	3.458
Avg.	4.393

doop.

However, there are several aspects of RFMS that need to be improved. We would like to further investigate the potential for self-configuration. We also need to reduce the RFMS overhead for online Hadoop self-tuning.

Acknowledgment

This work is partially supported by the cooperation project of Research on Green

Cloud IDC Resource Scheduling with ZTE Corporation.

References

- [1] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello, "Sindice.com: a document oriented lookup index for open linked data," *Int. J. Metadata, Semantics, and Ontologies*, vol. 3, no. 1, pp. 37–52, Nov. 2008.
- [2] Mahout – Apache Software Foundation project home page [Online]. Available: <http://lucene.apache.org/mahout>
- [3] K. S. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Y. Eltabakh, C. C. Kanne, F. Özcan, and E. J. Shekita, "Jaql: a scripting language for large scale semistructured data analysis," *PVLDB*, vol. 4, no. 12, p. 1272–1283, 2011.
- [4] Mao-Ping Wen, Hsio-Yi Lin, An-Pin Chen, and Chyan Yang, "An integrated home financial investment learning environment applying cloud computing in social network analysis," in *Int. Conf. ASONAM*, Kaohsiung, 2011, pp. 751–754.
- [5] B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg, "Searching for SNPs with cloud computing," *Genome Biology*, R134, Oct. 2009.
- [6] P. Lama and Xiaobo Zhou, "AROMA: automated resource allocation and configuration of mapReduce environment in the cloud," in *Proc. 9th Int. Conf. Autonomic Computing (ICAC)*, San Jose, 2012, pp. 63–72.
- [7] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "Starfish: a self-tuning system for big data analytics," in *Proc. Conf. Innovative Data Systems Research*, Asilomar, CA, Jan. 9–12, 2011, pp. 261–272.
- [8] K. Kambhata, A. Pathak, and H. Pucha, "Towards optimizing hadoop provisioning in the cloud," in *Proc. HotCloud'09*, San Diego, CA.
- [9] Jinquan Dai, Jie Huang, Shengsheng Huang, Bo Huang, and Yan Liu, "HiTune: dataflow-based performance analysis for big data cloud," in *Proc. USENIXATC'11*, Shanghai, 2011, pp. 7–7.
- [10] H. Herodotou and S. Babu, "Profiling, what-if analysis, and cost-based optimization of MapReduce programs," in *Proc. VLDB*, Seattle, WA, Aug. 29–Sep. 3, 2011, vol. 4, no. 11, pp. 1111–1122.
- [11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [12] V. Bhat, S. Gogate, and M. Bhandarkar, "Hadoop Vaidya," *Hadoop World*, 2009.
- [13] Jiaqi Tan and Xinghao Pan, "Kahuna: problem diagnosis for MapReduce-based cloud computing environments," in *Proc. 12th IEEE/IFIP NOMS 2010*, Osaka, pp. 112–119.
- [14] Intel VTune Performance Analyzer [Online]. Available: <http://software.intel.com/en-us/intel-vtune/>
- [15] Guofei Jiang and Haifeng Chen, "Autotuning Configurations in Distributed Systems for Performance Improvements using Evolutionary Strategies," in *ICDCS'08*, Beijing, Jun. 2008, pp. 769–776.
- [16] Jia Rao, Xiangping Bu, Cheng-Zhong Xu, and Kun Wang, "A distributed self-learning approach for elastic provisioning of virtualized cloud resources," in *Proc. 2011 IEEE 19th Int. Symp. MASCOTS*, Singapore, pp. 45–54.
- [17] Cheng-Zhong Xu, Jia Rao, and Xiangping Bu, "URL: a unified reinforcement learning approach for autonomic cloud management," *Journal of Parallel and Distributed Computing*, vol. 72, no. 2, pp. 95–105, Feb. 2012.
- [18] Yanfei Guo, P. Lama and Xiaobo Zhou, "Automated and agile server parameter tuning with learning and control," in *2012 IEEE 26th Int. IPDPS*, Shanghai, 2012, pp. 656–667.
- [19] S. Babu, "Towards automatic optimization of MapReduce programs," in *Proc. 1st ACM Symp. Cloud Computing*, Indianapolis, IN, 2010, pp. 137–142.
- [20] A Dynamic Instrumentation Tool for Java [Online]. Available: kenai.com/projects/btrace.
- [21] H. Herodotou, "Hadoop Performance Models," Computer Science Department Duke Univ., Tech. Rep. May 2011.
- [22] T. Luo, K. Kramer, D. B. Goldgof, L. O. Hall, S. Samson, A. Remsen, et al., "Recognizing plankton images from the shadow image particle profiling evaluation recorder," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34 no. 4, pp. 1753–1762, Aug. 2004.
- [23] H. Ishwaran, E. H. Blackstone, C. E. Pothier, and M. S. Lauer, "Relative risk forests for exercise heart rate recovery as a predictor of mortality," *Journal of the American Statistical Association*, vol. 99, no. 467, pp. 591–600, Sep. 2004.

Manuscript received: May 20, 2013

Biographies

Zhendong Bei is a PhD student in computer applications at Shenzhen Institutes of Advanced Technology, China. He received his BS degree from the National University of Defense Technology, China, in 2006, and received his MS degree from Central South University, China, in 2009. His research interests include cloud computing, data mining, machine learning, and image processing.

Zhibin Yu (zb.yu@siat.ac.cn) received his PhD degree in computer science from Huazhong University of Science and Technology in 2008. He spent one year as a visiting scholar in the Laboratory of Computer Architecture, University of Texas at Austin. He is currently an associate professor at the Shenzhen Institutes of Advanced Technology. His research interests include microarchitecture simulation, computer architecture, workload characterization and generation, performance evaluation, multicore architecture, and virtualization technologies. In 2005, he won first prize in the HUST Young Lecturers Teaching Contest, and in 2003, he won second prize in the teaching quality assessment of HUST. He is a member of IEEE and ACM.

Huiling Zhang received her MS degree in signal and information processing from Southwest University, China, in 2011. She joined the Center for High-Performance Computing at Shenzhen Institutes of Advanced Technology and now works as a research assistant there. Her current research interests include high-performance computing, and machine learning and its applications in bioinformatics.

Chengzhong Xu (czxu@wayne.edu) received his BS degree and MSc degree in computer science and engineering from Nanjing University in 1986 and 1989. He received his PhD degree from the University of Hong Kong in 1993. His research interests include computer architecture, distributed systems, virtualization, and cloud computing. Dr. Xu is a professor of electrical and computer engineering at Wayne State University. He is also the director of the Cloud and Internet Computer Laboratory at Wayne State University. He is an IEEE senior member and ACM member.

Shengzhong Feng is a professor, deputy director of the Institute of Advanced Computing and Digital Engineering, Shenzhen Institute of Advanced Technology. His research interests are parallel algorithms, grid computing, and bioinformatics. In particular, he is focused on developing novel, effective methods of modeling digital cities and applications. Before coming to SIAT, he worked in the Institute of Computing Technology, Chinese Academy of Sciences, and participated in research on the Dawning supercomputer. He graduated from the University of Science and Technology of China in 1991 and received his PhD from Beijing Institute of Technology in 1997.

Zhenjiang Dong (dong.zhenjiang@zte.com.cn) is the vice president of the Communication Services R&D Institute for Cloud Computing and IT Operation, ZTE Corporation. He received his MS degree from Harbin Institute of Technology in 1996. His research interests include cloud computing, multimedia networking, and mobile networking.

Hengsheng Zhang (zhang.hengsheng@zte.com.cn) received his bachelor's degree from Anhui University, China. He joined ZTE in 2005, and is a pre-research engineer and senior architect. His research interests include value added services and cloud computing.

Parallel Spectral Clustering Based on MapReduce

Qiwei Zhong¹, Yunlong Lin¹, Junyang Zou¹,
Kuangyan Zhu¹, Qiao Wang¹, and Lei Hu²

(1. School of Information Science and Engineering, Southeast University,
Nanjing 210096, China;
2. ZTE Corporation, Nanjing 210012, China)



Abstract

Clustering is one of the most widely used techniques for exploratory data analysis. Spectral clustering algorithm, a popular modern clustering algorithm, has been shown to be more effective in detecting clusters than many traditional algorithms. It has applications ranging from computer vision and information retrieval to social science and biology. With the size of databases soaring, clustering algorithms have scaling computational time and memory use. In this paper, we propose a parallel spectral clustering implementation based on MapReduce. Both the computation and data storage are distributed, which solves the scalability problems for most existing algorithms. We empirically analyze the proposed implementation on both benchmark networks and a real social network dataset of about two million vertices and two billion edges crawled from Sina Weibo. It is shown that the proposed implementation scales well, speeds up the clustering without sacrificing quality, and processes massive datasets efficiently on commodity machine clusters.



Keywords

spectral clustering; parallel implementation; massive dataset; Hadoop MapReduce; data mining

1 Introduction

Clustering is a method of unsupervised learning that is widely used in exploratory data analysis [1]. Recently, a spectral clustering algorithm was shown to be more effective than many other traditional algorithms in detecting clusters. Extracting valuable information from an ocean of data is a hot research topic, and applications of spectral clustering range from computer vision and information retrieval to social science and biology [1], [2]. Spectral clustering cannot be adequately scaled in terms of computational time and memory use to deal with massive datasets. For example, a quadratic resource bottleneck occurs when computing pairwise similarities and the number of data instances n

(Table 1) is large. A considerable amount of time is needed to compute the first k (Table 1) eigenvectors of a Laplacian matrix; therefore, it is necessary to develop dataset-oriented par-

▼ Table 1. Notation used in this paper

Symbol	Quantity
n	number of data instances
K	number of desired clusters
T	number of nearest neighbors
G	graph
V	vertex set
E	edge set
W	adjacency matrix
D	degree matrix
S	similarity matrix
L	Laplacian matrix
T	real symmetric tridiagonal matrix
Q	Modularity

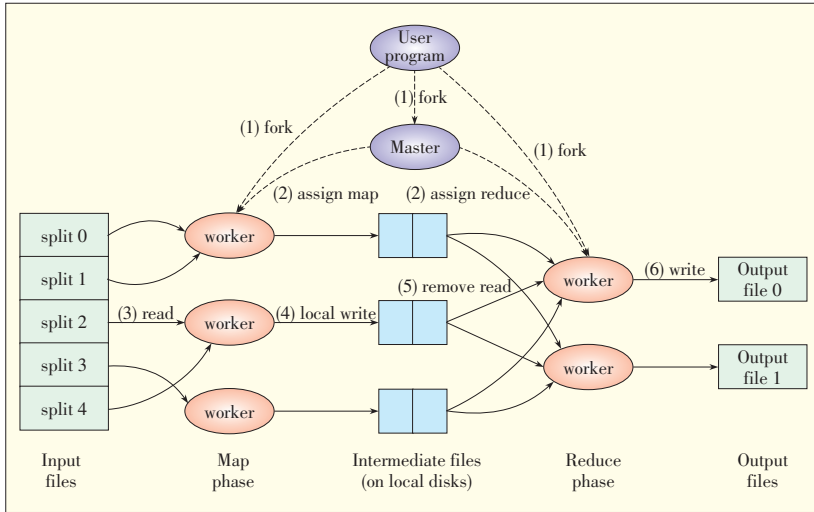
allel implementations [2], [3]. In [4], message passing interface (MPI) is used to parallelize the spectral clustering algorithm. MapReduce is better than MPI and can automatically split mass data with Hadoop Distributed File System (HDFS). Better load balancing and data parallelism improve the scalability and efficiency of the machine clusters when a dataset is large. Therefore, a MapReduce framework is more suitable for handling this problem.

MapReduce is a programming model and associated implementation for processing large datasets [5]. Users specify the computational rules in forms of a *Map* function that processes a $\langle key, value \rangle$ pair to generate a set of intermediate $\langle key, value \rangle$ pairs. A *Reduce* function merges all intermediate values associated with the same *key*. Programs written in this way can be executed in parallel on large commodity machine clusters. The underlying runtime system automatically partitions the input data, schedules the program's execution, handles machine failures, and manages intermachine communication. This allows programmers who are inexperienced with parallel to easily use the resources of a large machine cluster. The overall flow of a common MapReduce operation is shown in Fig. 1. Both Google and Hadoop provide MapReduce runtimes that are flexible and tolerant to faults.

In this paper, we propose a parallel spectral clustering implementation (PSCI) that is based on Hadoop MapReduce and that can be applied to massive datasets. By constructing proper $\langle key, value \rangle$ pairs, the proposed implementation can be efficiently executed in parallel. Tests on benchmark networks show the effectiveness of PSCI in detecting communities. We analyze a real social network dataset of about two million vertices and two billion edges crawled from Sina Weibo (a popular

Parallel Spectral Clustering Based on MapReduce

Qiwei Zhong, Yunlong Lin, Junyang Zou, Kuangyan Zhu, Qiao Wang, and Lei Hu



▲ Figure 1. MapReduce execution overview.

Twitter-like microblog popular in China) to show the efficiency and practicability of PSCI for massive datasets.

In section 2, we give a brief overview of spectral clustering algorithm in order to understand particular bottlenecks and to analyze the parallel parts of the algorithm. In section 3, we propose PSCI based on the Hadoop MapReduce framework. In section 4, we show the results of tests on PSCI and evaluate its clustering quality and scalability (in terms of runtime speedup and scale up). Concluding remarks are made in section 5.

2 Spectral Clustering Algorithm

The most common spectral clustering algorithm is graph Laplacian matrix [1], [6]. We assume that $G = (V, E)$ is a weighted undirected graph with vertex set $V = \{v_1, v_2, \dots, v_n\}$ and edge set $E = \{e_1, e_2, \dots, e_m\}$. Each edge between vertices v_i and v_j has a non-negative weight $w_{ij} \geq 0$ and $w_{ij} = w_{ji}$. The adjacency matrix of the graph is

$$W = (w_{ij})_{i,j=1,\dots,n} \quad (1)$$

where $w_{ij} = 0$ means that v_i and v_j are not connected along an edge. Then, the degree of a v_i can be defined as

$$d_i = \sum_{j=1}^n w_{ij} \quad (2)$$

where the sum in (2) only runs over all vertices that are adjacent v_i , on account that the weight $w_{ij} = 0$ for all other vertices v_j . The degree matrix of the graph is

$$D = \text{diag}(d_1, d_2, \dots, d_n) \quad (3)$$

where the degrees d_1, d_2, \dots, d_n are on the diagonal. Based on the adjacency matrix and degree matrix, an un-normalized graph Laplacian matrix is given by $L = D - W$, and the normalized graph Laplacian matrix is given by

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \quad (4)$$

We now assume that a dataset comprises n instances x_1, x_2, \dots, x_n , which can be arbitrary objects. The pairwise similarity between x_i and x_j is given by $s_{ij} = s(x_i, x_j)$ and can be measured by a similarity function that is non-negative and symmetric. The corresponding similarity matrix is given by

$$S = (s_{ij})_{i,j=1,\dots,n} \quad (5)$$

An example of a similarity function is the Gaussian function

$$s_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|}{2\sigma^2}\right) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where the parameter σ controls the number of the neighbors. Nevertheless, s_{ij} in network is usually defined as

$$s_{ij} = \begin{cases} 1 & \text{if vertices } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

If we replace W in (4) with S , we obtain the new formula in a spectral clustering algorithm:

$$L_{sym} = I - D^{-1/2} S D^{-1/2} \quad (8)$$

Furthermore, one often reduces the matrix S to a sparse one by considering only significant relationship between instances for conserving the computational time. A summary of the spectral clustering algorithm is shown here [1].

Algorithm 1. Spectral Clustering Algorithm

Input: the similarity matrix $S \in R^{n \times n}$, and the number of desired clusters k .

Output: k Clusters.

Procedure:

1. Construct a similarity graph and let W be its weighted adjacency matrix.
2. Compute the normalized Laplacian matrix L_{sym} .
3. Compute the first k eigenvectors u_1, u_2, \dots, u_k of L_{sym} .
4. Let $U \in R^{n \times k}$ be the matrix containing the vectors u_1, u_2, \dots, u_k as columns.
5. Form the matrix $T \in R^{n \times k}$ from U by normalizing the rows to norm 1.
6. For $i = 1, \dots, n$, $y_i \in R^k$ be the vector corresponding to the i th row of T .
7. Cluster the points $(y_i)_{i=1,\dots,n}$ with the k -Means algorithm into clusters C_1, C_2, \dots, C_k .
8. Output clusters A_1, A_2, \dots, A_k with $A_i = \{j | y_j \in C_i\}$.

Generally, it is useful to change representation of the abstract data points x_i to points $y_i \in R_k$ due to the properties of graph Laplacian matrix. It enhances the cluster performance, so that clusters can be trivially detected in the new representation. In particular, the simple k -means clustering algorithm detects the clusters without any difficulties.

Algorithm 2. k -Means Algorithm

Input: the dataset comprised of instances, and the number of desired clusters k .

Output: k Clusters.

Procedure:

1. k initial group centroids (so-called "Means") are randomly selected from the dataset.
2. k clusters are created by associating every object with the nearest centroid.
3. The positions of the k centroids are recalculated when all objects have been assigned.
4. Step 2 and 3 are repeated until the centroids no longer move (or convergence is reached).

3 Parallel Implementation Based on MapReduce

There are three intensive computing processes in a spectral clustering algorithm: construction of the Laplacian matrix, computation of the first k eigenvectors, and calculation of distances in k -means. Therefore, after preprocessing the original dataset, we reasonably segment the similarity matrix computation and sparse computation by data point index in order to construct the Laplacian matrix. When calculating the eigenvectors, we put the Laplacian matrix on an HDFS and launch distributed Lanczos operations to get the first k eigenvectors of the Laplacian matrix. Finally, we make parallel k -means clustering algorithms on the eigenvector's transposed matrix to get the final clustering results. The entire process of our parallel implementation is shown in Fig. 2.

3.1 Construction of the Laplacian Matrix

The Laplacian matrix is constructed in two steps: computation of similarity matrix and sparsification of the similarity matrix. Fortunately, both computation of the pairwise similarities and t -nearest neighbors of one object are not related to the computation of those of other objects. Therefore, the computations for different objects can be done in parallel via dataset partitioning (Fig. 3).

3.1.1 Mapper: Calculation of Similarity

Input: $\langle \text{key}, \text{value} \rangle$ pair, where key is the unique index and features of one object and value is the index and corresponding features of each other object whose index is greater.

Output: $\langle \text{key}', \text{value}' \rangle$ pair, where key' is the index of one object, and value' is the index of each other object and its similarity with the object in key' .

3.1.2 Reducer: Construction of the Sparse Matrix

Input: $\langle \text{key}, \text{value} \rangle$ pair, where key is the index of one object and value is the index of each other object and its similarity with the object in key .

Output: $\langle \text{key}', \text{value}' \rangle$ pair, where key' is the index of one object and value' is the index and corresponding similarity of each other object among

t -nearest neighbors of the object in key' .

3.2 Computation of First k Eigenvectors

After we have calculated and stored the Laplacian matrix, we must parallelize the eigensolver. The Lanczos algorithm is an iterative algorithm. It is an adaptation of power methods and is used to find eigenvalues and eigenvectors of a square matrix or the singular value decompositions of a rectangular matrix. It is particularly useful for finding decompositions of very large sparse matrices.

Algorithm 3. Lanczos Algorithm

Input: the Laplacian matrix.

Output: a tridiagonal matrix T and a Q matrix.

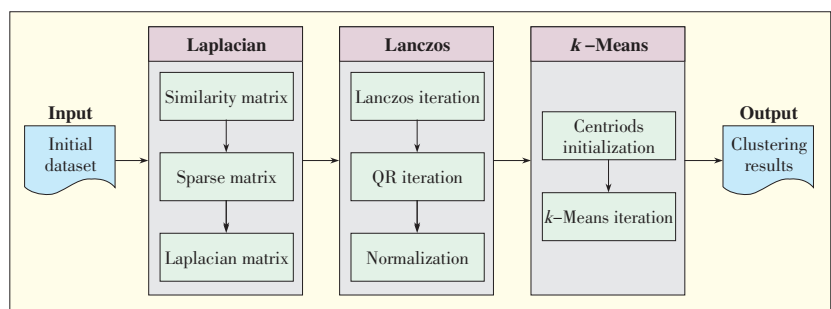
Procedure:

1. Initialization.
 $\vec{q}_0 \leftarrow 0, \beta_0 \leftarrow 0.$
 $\vec{q}_1 \leftarrow$ random vector normalized 1.
2. Iteration.
 For $k = 1, 2, \dots, m$
 $\vec{w}_k = L_{\text{sym}} \vec{q}_k - \beta_k \vec{q}_{k-1}.$
 $\alpha_k = (\vec{w}_k, \vec{q}_k).$
 $\vec{w}_k = \vec{w}_k - \alpha_k \vec{q}_k.$
 $\beta_{k+1} = \|\vec{w}_k\|.$
 $\vec{q}_{k+1} = \vec{w}_k / \beta_{k+1}.$
3. Return.

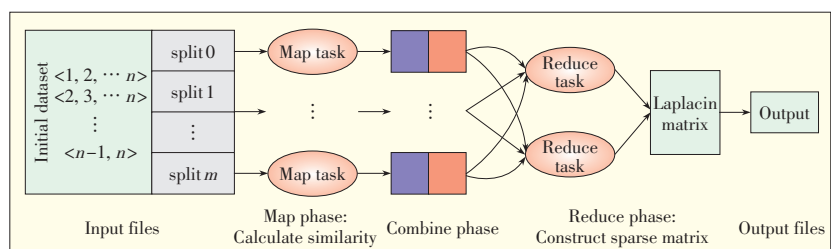
$$Q_{\text{norm}} = (\vec{q}_1, \vec{q}_2, \vec{q}_3, \dots, \vec{q}_{m-1}, \vec{q}_m).$$

$$T_{\text{mm}} = \begin{pmatrix} \alpha_1 & \beta_2 & & & & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \ddots & & \\ & & \ddots & \ddots & \beta_{m-1} & \\ & & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & & & & \beta_m & \alpha_m \end{pmatrix}.$$

The Lanczos iteration converts the Laplacian matrix $L_{m \times n}$ into



▲ Figure 2. Parallel spectral clustering implementation process.



▲ Figure 3. Laplacian matrix construction.

Parallel Spectral Clustering Based on MapReduce

Qiwei Zhong, Yunlong Lin, Junyang Zou, Kuangyan Zhu, Qiao Wang, and Lei Hu

a real symmetric tridiagonal matrix $T_{m \times m}$ ($m < n$), through which eigenvalues and eigenvectors can be easily solved by methods such as *QR Iteration*. A $\langle \text{eigenvalue}, \text{eigenvector} \rangle$ pair of L , which is $(\lambda_k, Q_{n \times m} \vec{v}_k)$, corresponds to the $\langle \text{eigenvalue}, \text{eigenvector} \rangle$ pair of T , which is (λ_k, \vec{v}_k) . In each iteration of the Lanczos algorithm, matrix–vector multiplication is the most intensive calculation, and can be done in parallel via matrix partitioning (Fig. 4).

3.2.1 Mapper: Matrix–Vector Multiplication

Input: Global vector V . $\langle \text{key}, \text{value} \rangle$ pair, where key is the line index of matrix, and value is the content corresponding to the index in key .

Output: $\langle \text{key}', \text{value}' \rangle$ pair, where key' is $NULL$, and value' is the line index and multiplication result between value and vector V .

3.2.2 Reducer: Construction of the Multiplication Result

Input: $\langle \text{key}, \text{value} \rangle$ pair, where key is $NULL$, and value is all the partial results.

Output: $\langle \text{key}', \text{value}' \rangle$ pair, where key' is $NULL$, and value' is the final result vector of matrix–vector multiplication.

3.3 Parallel k –Means Algorithm

Computations of the distances between one object and the centers are not related to computations of the distances between other objects and corresponding centers [7]. Therefore, the computation of distances between different objects and centers can be done in parallel. The new centers, which will be used in the next iteration, should be updated in each iteration. Therefore, the iterative procedures must be executed serially (Fig. 5).

3.3.1 Mapper: Associate Every Instance with the Nearest Center

Input: Global variable centers, $\langle \text{key}, \text{value} \rangle$ pair, where key is the index of one instance and value is the feature (i.e. the dimension values) corresponding to the instance in key .

Output: $\langle \text{key}', \text{value}' \rangle$ pair, where key' is the index of the nearest center of the instance and value' is the index and feature of the instance.

3.3.2 Combiner: Calculation of the Sum and Quadratic Sum of Values of Each Dimension of Instances and its Number Assigned to the Same Center on Local Disk

Input: $\langle \text{key}, \text{value} \rangle$ pair, where key is the index of the center, and value is the index and feature of each instance assigned to the same center

Output: $\langle \text{key}', \text{value}' \rangle$ pair, where key' is the index of the center, and value' is the sum and quadratic sum of values of each dimension of instances and its number with the same center.

dratic sum of values of each dimension of instances and its number with the same center.

3.3.3 Reducer: Calculate the New Centers and Iteration Condition.

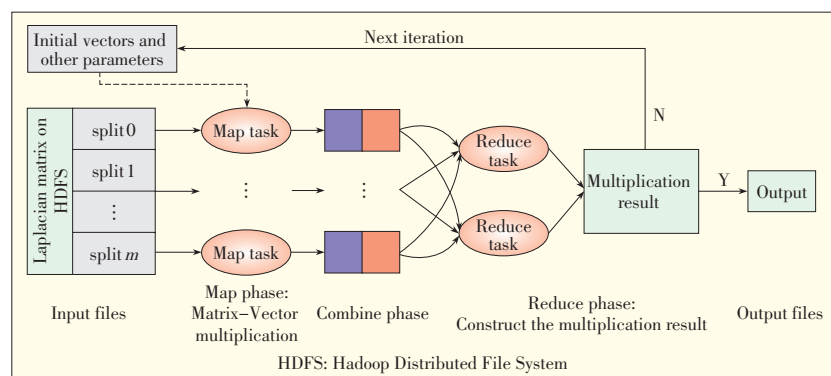
Input: $\langle \text{key}, \text{value} \rangle$ pair, where key is the index of the center and value comprises the sum and quadratic sum of the values in each dimension of instances with the same center from all hosts, and the number of those instances.

Output: $\langle \text{key}', \text{value}' \rangle$ pair, where key' is the index of the new center and value' is the features representing the new center.

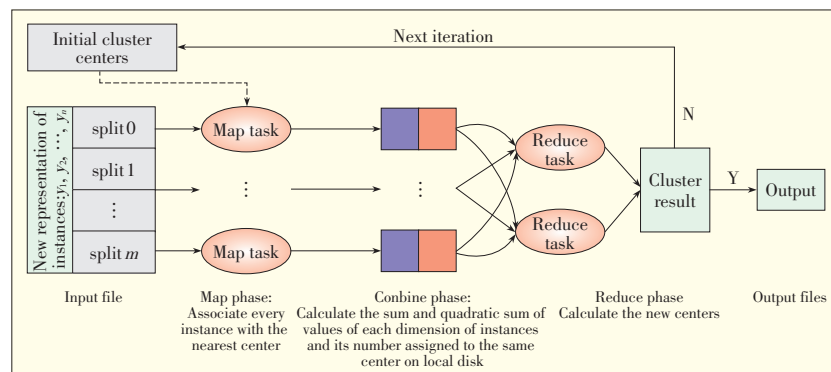
4 Empirical Analysis

We designed our experiments to validate the scalability and quality of parallel implementation in PSCI. Our experiments are based on both computer-generated benchmark networks and a real social network dataset of 1,628,853 vertices and 176,620,423 edges crawled from Sina Weibo beforehand. We ran the experiments on IBM Blade Cluster with 10 compute nodes. All the nodes were identical, and each was configured with a CPU faster than 2 GHz and memory greater than 8 GB. Hadoop version 0.20.0 and Java 1.6.0 were used as the MapReduce system for all experiments.

To test the performance of parallel implementation, we use



▲ Figure 4. Matrix–Vector multiplication.



▲ Figure 5. Parallel k –means algorithm.

Modularity as an evaluation function. Modularity is a property of a network and a specific proposed division of that network into communities [8], [9]. It measures whether the division is good in the sense that there are many more edges within communities but only a few in between. The higher the Modularity score, the better the clustering quality. Generally, this score falls between 0.3 and 0.7 in social networks [9].

If $G = (V, E)$ with the adjacency matrix A is constructed of

$$A_{uv} = \begin{cases} 1 & \text{if vertices } u \text{ and } v \text{ are connected} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

then we assume that the vertices are divided into communities so that vertex u belongs to community c_u . The Modularity Q is

$$Q = \sum_i (e_{ii} - a_i^2) \quad (10)$$

Also,

$$e_{ij} = \frac{1}{2m} \sum_{uv} A_{uv} \delta(c_u, i) \delta(c_v, j) \quad (11)$$

$$d_u = \sum_v A_{uv} \quad (12)$$

$$a_i = \frac{1}{2m} \sum_u d_u \delta(c_u, i) \quad (13)$$

$$m = \frac{1}{2} \sum_{uv} A_{uv} \quad (14)$$

where e_{ij} ($i \neq j$) is the fraction of edges that join vertices in community i to vertices in community j , and e_{ii} is the fraction of edges in the same community i . The δ function $\delta(i, j)$ is 1 if $i = j$ and 0 otherwise; d_u is the degree of vertex u ; a_i is the fraction of ends of edges that are attached to vertices in community i and m is the number of edges in the graph.

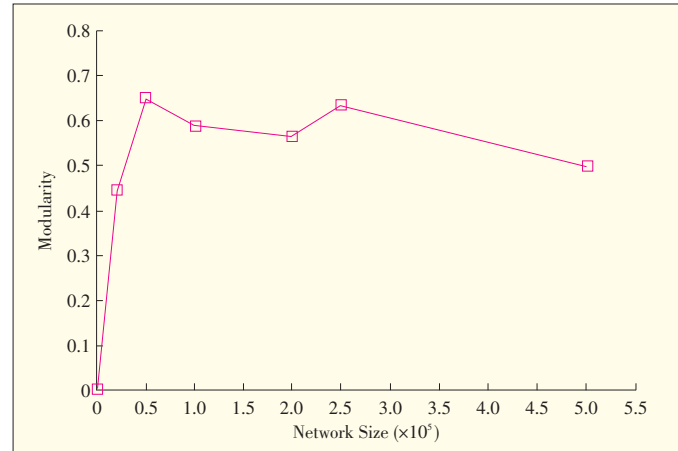
4.1 Experiments on Benchmark Networks

Benchmark networks are standard measures for community-detection algorithms [10]. Here, we generate the benchmark networks with parameters $\langle k \rangle = 120$, $\beta = 1$, $\gamma = 2$, $\mu = 0.2$, $k_{\max} = 1000$, $s_{\min} = 500$, and $s_{\max} = 2000$. First, we test performance of PSCI with network sizes ranging from 20,000 to 500,000. The results are shown in Fig. 6 and Fig. 7. Modularity is high for different network sizes and falls between 0.4 and 0.7. This supports the results in [9]. Fig. 7 shows that PSCI has very good scalability. Computational time almost becomes linear as the network grows above 100,000. This means that PSCI has good scalability and treats massive datasets efficiently.

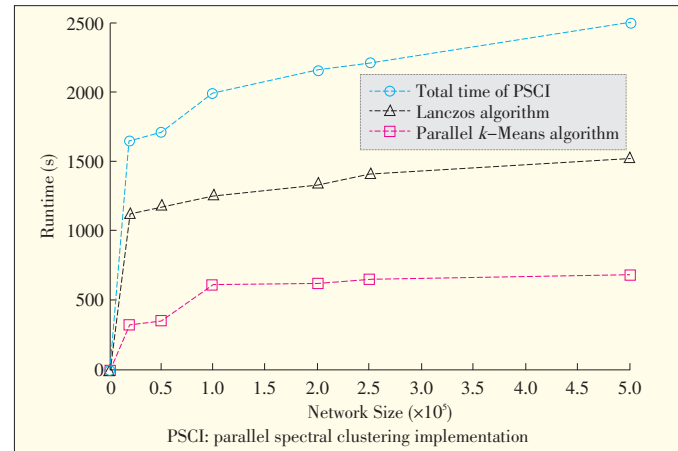
When determining the speedup with different machine numbers on a benchmark network of 500,000 instances (Fig. 8), we see that in the beginning, the implementation has nearly linear substantial speedup as the number of machines increases. Then, there is a slowdown caused by the overhead time of framework startup and required intermachine communication.

4.2 Experiments on Real Networks

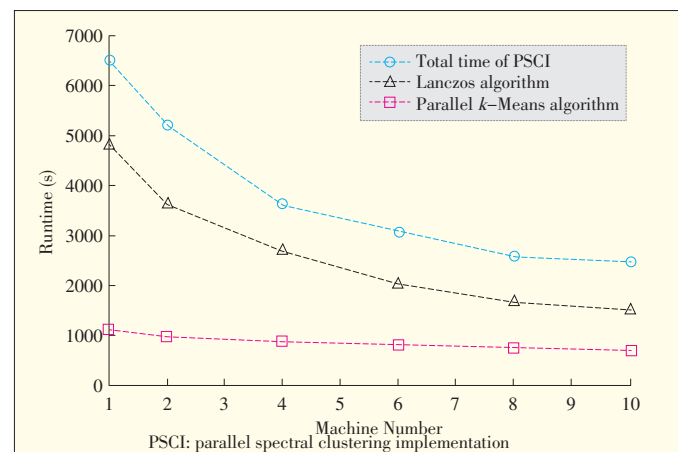
Here, we analyze the quality and speedup on a real social network of 1,628,853 vertices and 176,620,423 edges crawled



▲ Figure 6. Modularity of clustering results with different network sizes.



▲ Figure 7. Runtimes of PSCI for different network sizes.



▲ Figure 8. The runtimes of PSCI with the increasing number of machines.

from Sina Weibo. The edge represents the following relationship between one vertex and another. Table 2 shows Modularity and runtimes of processing on the real social network on 10 machines. The visual clustering using the adjacency matrix plot is shown in Fig. 9. The results indicate that our parallel

Parallel Spectral Clustering Based on MapReduce

Qiwei Zhong, Yunlong Lin, Junyang Zou, Kuangyan Zhu, Qiao Wang, and Lei Hu

▼ Table 2. Modularity and runtimes on the real network

Modularity	Runtime (s)		
	Total time	Lanczos	Parallel k -Means
0.50044	3976.31518	3136.38745	522.09389



▲ Figure 9. A visual clustering result using the adjacency matrix plot of the real network. The horizontal and vertical coordinates represent the vertices, and the diagonal is a clear line that indicates the good quality of the clustering.

implementation speeds up the clustering process without sacrificing quality.

In addition, the runtime for dealing with Picasa (a dataset of 637,137 images) on 16 machines is nearly 15,000 seconds [4]. This parallelizes the spectral clustering algorithm based on MPI. As a consequence, our implementation is faster for parallel processing because both the dataset and spectral clustering algorithm are both distributed.

5 Conclusion

The spectral clustering algorithm is one of the most important modern algorithms and has been shown to be more effective in community detection than many traditional algorithms. However, the growing amount of data in applications makes spectral clustering of massive datasets challenging. In this paper, we propose a parallel spectral clustering implementation based on MapReduce. In our PSCI, both the computation and data storage are distributed, and this solves the problems of most of the existing algorithms mentioned at the outset of this paper. By empirically analyzing both benchmark networks and a real massive social network dataset, we show that the proposed implementation has high Modularity across different network sizes; it reduces the computational time so that it is almost linear; and the substantial speedup is nearly linear as the number of machines increases within a certain range. The implementation scales well, speeds up clustering without sacrific-

ing quality, and processes massive datasets efficiently on commodity machine clusters.

References

- [1] U. von Luxburg, "A Tutorial on Spectral Clustering," *Statistics and Computing*, vol. 17, pp. 395–416, Aug. 2007.
- [2] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y. Chang, "Parallel Spectral Clustering in Distributed Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 568–586, Mar. 2011.
- [3] U. von Luxburg, O. Bousquet, and M. Belkin, "Limits of Spectral Clustering," *Neural Information Processing Systems Foundation*, 2004.
- [4] Yangqiu Song, Wen-Yen Chen, Hongjie Bai, Chih-Jen Lin, and Edward Y. Chang, "Parallel Spectral Clustering," *Machine Learning and Knowledge Discovery in Databases*, vol. 5212, pp. 374–389, 2008.
- [5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM—50th anniversary issue: 1958–2008*, vol. 51, pp. 107–113, Jan. 2008.
- [6] Fan R. K. Chung, *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*, Providence, RI: American Mathematical Society, 2007.
- [7] Weizhong Zhao, Huifang Ma, and Qing He, "Parallel K-Means Clustering Based on MapReduce," *Cloud Computing: First International Conference*, Beijing, China, Dec. 2009, pp. 674–679.
- [8] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, 69, 026113, 2004.
- [9] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, 70, 066111, 2004.
- [10] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, 78, 046110, 2008.

Manuscript received: September 19, 2012

Biographies

Qiwei Zhong (qwzhong1988@seu.edu.cn) received his BS degree in electronic information Engineering from Nanjing University of Science and Technology. He is currently working towards his MS degree in the School of Information Science and Engineering, Southeast University, Nanjing. His research interests include data mining, social network analysis, distributed processing, and data visualization.

Yunlong Lin (linyl@seu.edu.cn) received his BS degree from the School of Information Science and Engineering, Southeast University. He is currently working towards his MS degree in the Laboratory of Netmedia and Datamining, Southeast University, Nanjing. His research interests include data mining, recommendation system, and distributed computing.

Junyang Zou (zoujyjs@seu.edu.cn) received his BS degree in opto-electronic engineering from Nanjing University of Posts and Telecommunications. He is currently working towards his BS degree at the School of Information Science and Engineering, Southeast University, Nanjing. His research interests include data mining and distributed computing.

Kuangyan Zhu (zhukuangyan@seu.edu.cn) received his BS degree from the School of Information Science and Engineering, Southeast University, Nanjing. He is currently working towards his MS degree at Southeast University. His research interests include recommendation system and distributed computing.

Qiao Wang (qiaowang@seu.edu.cn) is a professor and doctoral tutor at Southeast University, Nanjing. He received his PhD degree in mathematics from Wuhan University. He is now the director of the Signal and Information Processing Laboratory, Southeast University. He was a visiting scientist in the Division of Engineering and Applied Science, Harvard University, from 2003 to 2004. His research interests include spectral analysis, statistical signal processing, image processing, and signal processing in applications such as biology and transportation.

Lei Hu (hu.lei2@zte.com.cn) received his MS degree from the Laboratory of Intelligent Recognition and Image Processing, Beihang University, in 2008. He is a senior research engineer for mass data analysis projects of ZTE Corporation. His research interests include data mining, information retrieval, and social network analysis.

Spam Filtering: Online Naive Bayes Based on TONE

Guanglu Sun¹, Hongyue Sun², Yingcai Ma³,
and Yuewu Shen³

(1. Research Institute of Information Technology, Tsinghua University, Beijing 100084, China;

2. ZTE Corporation, Shenzhen 518057, China;

3. School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China)



Abstract

The naive Bayes (NB) model has been successfully used to tackle spam, and is very accurate. However, there is still room for improvement. We use a train on or near error (TONE) method in online NB to enhance the performance of NB and reduce the number of training emails. We conducted an experiment to determine the performance of the improved algorithm by plotting (1-ROCA)% curves. The results show that the proposed method improves the performance of original NB.



Keywords

spam filtering; online naive Bayes; train-on or near error

1 Introduction

Email is an efficient communication technology and one of the most widely used internet applications. However, spam is a drain on network resources and is often detrimental to user experience. Some people use spam for malicious purposes, so spam filtering is a hot-spot in current research.

The body of the email contains essential information and is arguably the most important part of the email. Content-based filtering is a reliable method for combating spam. Machine learning provides more accurate prediction and is an attractive solution for content-based filtering. However, there is no consensus about which learning algorithms are best [1].

Machine learning techniques are usually based on generative models, such as naive Bayes (NB), or discriminative models, such as support vector machines (SVMs). For most large-scale tasks, discriminative models perform better than

generative models [2]. This is especially true when there is sufficient training data. In TREC spam track, the Bogofilter is a fast Bayesian spam filter that is used as the baseline [3]. Many researchers have achieved state-of-the-art spam filtering using SVMs; however, SVMs typically require training time that is quadratic in the number of training examples [4]. SVMs are not suitable for online filtering because they are not updated in real time. With the Bayesian method, filtering is inaccurate, but only linear training time is required, and robustness is less likely to be affected by bad data [5], [6]. The Bayesian filtering system is easy to deploy because it is simple and lightweight [7].

In this paper, we propose an improved online NB classifier. Online NB is often used in spam filtering, but unlike SMV, it can update itself in real time according to spam behavior. In the original NB model, training data is passively accepted. Updating the training data is expensive for most classifiers, and this practice has been strongly discouraged by industry [8]. Train on or near error (TONE) is a sample-selection method that can be used to discard useless examples [9]. Only parts of examples are trained using TONE. When TONE is applied to the online NB model and tested with several large spam data sets, the online model performs better than the original NB model. In particular, the number of examples needed to train an effective classifier decreases.

In section 2, we review the framework of an online learning model for spam filtering. In section 3, we describe online NB models based on TONE. In section 4, we show that the improved algorithm is much more efficient than the original NB algorithm. Section 5 concludes the paper.

2 Online Learning Model for Spam Filtering

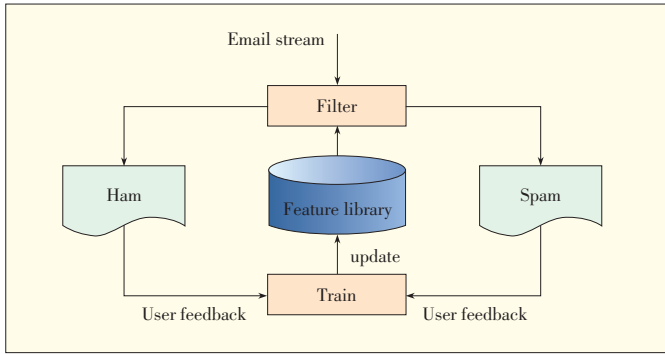
Many models used in traditional machine-learning applications operate in pool-based (offline) mode [10]. The model is trained on a large data set and the examples are reclassified without retraining. The process of an offline learning model tends to be optimal for all the training data, but the online mode has an online learning process that can adapt to a changing environment. Online learning algorithms update the learner with new received examples; that is, they can use an old hypothesis (if one exists) as the starting point for retraining and adapting to changes in data.

Spam filtering is typically done online (Fig. 1). Emails are viewed as a stream, not as a pool, when entering the system one by one. The filter makes a spam or ham prediction for each email. Next, the user reads the message and perhaps gives feedback to the learning-based filter. The filter uses a label to update the feature library and retrain the learner. Ideally, this improves future predictive performance. Large-scale and online classification problems can be solved with a classifier that

This work is supported by National Natural Science Foundation of China under Grant NO. 60903083, Research fund for the doctoral program of higher education of China under Grant NO.20092303120005, and the Research Fund of ZTE Corporation.

Spam Filtering: Online Naive Bayes Based on TONE

Guanglu Sun, Hongyue Sun, Yingcai Ma, and Yuewu Shen



▲ Figure 1. The online spam filtering scenario.

allows online training and classification [11]. In a changing environment, an online NB learner is typically used for spam filtering, which proceeds incrementally [12]. An online NB learner only has linear training time and can be easily deployed in an online setting with incremental updates.

3 Online NB Model Based on TONE

3.1 Bayesian Spam Filtering

Naive Bayes is popular in industry probably because of its simplicity and the ease with which it can be implemented. Its linear computational complexity and high accuracy are comparable to that of more elaborate learning algorithms.

Here, we give notations for the NB model. In an example data set $\{(X^{(1)}, y^{(1)}) \dots (X^{(m)}, y^{(m)}) \dots\}$, $X^{(m)}$ denotes a vector containing features of the m th example. The corresponding label is $y^{(m)}$. The spam likelihood $P(y = \text{spam} | X)$ is calculated using the Bayesian formula:

$$P(y = \text{spam} | X) = \frac{P(\text{spam}) P(X | y = \text{spam})}{P(X)} \quad (1)$$

Similarly, the ham likelihood is calculated using

$$P(y = \text{ham} | X) = \frac{P(\text{ham}) P(X | y = \text{ham})}{P(X)} \quad (2)$$

To model $P(y | X)$, x_i is conditionally independent for a given y . This assumption is called the NB assumption. The resulting algorithm is called the NB classifier and is given by

$$P(X | y) = P(x_1 | y) P(x_2 | y) P(x_3 | y) \dots P(x_n | y) = \prod_{i=1}^n P(x_i | y) \quad (3)$$

In spam filtering, there is no need to estimate $P(X)$. The quotient of (1) and (2) is given by

$$\frac{P(y = \text{spam} | X)}{P(y = \text{ham} | X)} = \frac{P(\text{spam}) \prod_{i=1}^n P(x_i | y = \text{spam})}{P(\text{ham}) \prod_{i=1}^n P(x_i | y = \text{ham})} \quad (4)$$

We can use (4) to classify the email as spam or something

else. In (4), $P(\text{spam})$ is the a priori probability of spam, and $P(x_i | y = \text{spam})$ is expressed as a frequency in the spam category. The a priori probability of spam is given by

$$P(\text{spam}) = \frac{N_{\text{spam}}}{N_{\text{spam}} + N_{\text{ham}}} \quad (5)$$

and $P(x_i | y = \text{spam})$ is given by

$$P(x_i | y = \text{spam}) = \frac{N_{\text{spam email includes } x_i}}{N_{\text{spam}}} \quad (6)$$

where N_{spam} is the number of spams, and N_{ham} is the number of hams. The situation for ham is similar to that for spam.

3.2 The Model

The proposed NB algorithm works fairly well, but there is a simple tweak that makes it work much better, especially for text classification. If a feature only occurs in ham, then $P(x_i | y = \text{spam})$ may be zero. To avoid this, we can use Laplace smoothing, given by

$$P(x_i | y = \text{spam}) = \frac{N_{\text{spam email includes } x_i} + \epsilon}{N_{\text{spam}} + (\epsilon \times 2)} \quad (7)$$

To avoid underflow in the practical calculation, we use a logarithm. Therefore, (4) is transformed into

$$\begin{aligned} P_{\text{prime}} &= \log \frac{P(y = \text{spam} | X)}{P(y = \text{ham} | X)} \\ &= \log \frac{P(\text{spam})}{P(\text{ham})} + \sum_{i=1}^n \log \frac{P(x_i | y = \text{spam})}{P(x_i | y = \text{ham})} \end{aligned} \quad (8)$$

We can now classify the email by P_{prime} . If P_{prime} is greater than 0, the mail is predicted to be spam; otherwise, it is ham.

To apply TONE algorithm, we use the logistic function to convert P_{prime} into a score of 0~1. Equation (9) maps P_{prime} to a score of between 0 and approximately 1. The scale parameter ensures that P_{prime} is not too big:

$$\text{score} = \frac{1}{1 + \exp\left(-\frac{P_{\text{prime}}}{\text{scale parameter}}\right)} \quad (9)$$

To meet the spam filter's requirements, the online filter should update itself at the appropriate time. Spam filtering needs to be highly scalable because it involves large amounts of high-dimensional data. Content-based spam detection often requires training the learner. In original NB, there is no need to update the learner; however, in improved online NB, TONE can be applied to the training process (called thick threshold training) [13]. TONE is developed from train on error (TOE). There are two scenarios in which the learner training mode can be activated using this approach: 1) when samples have been misclassified by the filter and 2) when correctly classified samples fall within a predefined boundary. We improve the predicting ability of NB by introducing an online NB method based on TONE. The improved algorithm, called NB-TONE, is

cheap and does not result in performance loss.

With TONE, examples that have the least classification confidence are chosen. The parameter c is a thick threshold for training. Regardless of how the email is classified, if the score does not exceed the thick threshold, the email is not well classified, and the learner has to be trained and updated. On the other hand, TONE can also make a classifier more robust so that overfitting is averted. If the example is far from the hyperplane, the classifier predicts the example with higher confidence, and such examples do not need training. TONE is a sample-selection method that reduces the number of training examples and cuts down training time.

Algorithm 1. NB-TONE

```

1: for each mail  $\{ \langle X^{(i)}, y^{(i)} \rangle, \dots \} \ i=1, 2, \dots$ 
2:   A new message arrives
3:   Eq.8 // calculate the  $P_{\text{prime}}$ 
5:   Eq.9 //  $P_{\text{prime}}$  mapping to score
7:   if  $(\text{score} > 0.5)$  then
8:      $X^{(i)}$  is spam
9:   else
10:     $X^{(i)}$  is ham
11:   endif
12:   if  $(|\text{score} - 0.5| < c \text{ or } X^{(i)} \text{ is misclassified})$  then
13:     train model by  $\langle X^{(i)}, y^{(i)} \rangle$ 
14:   endif
15: end for

```

4 Evaluations and Results

In section 3, online NB spam filtering based on TONE was proposed. Here, we test the algorithm on large benchmark sets of email data.

4.1 Data Sets

Two benchmark data sets were used, both of which were from TREC spam filtering competitions. These data sets were trec05p, which contained 92,189 messages in English [3], and trec06p, which contained 37,822 messages in English [14]. For a fair comparison, each data set was ordered, and we compared our method with the original model. (1-ROCA)% was used as the standard performance measure.

4.2 Feature Space

Feature extraction is important for machine learning. An appropriate feature extraction method greatly improves the accuracy of the learner. In [15], character-level n -grams are valid and robust for a variety of spam detection methods [16]. Here, an email is represented as a vector that has a unique dimension for each possible substring of n characters. With the 4-gram feature extraction method, only the first 3000 features of each email were extracted, and the same features were re-

moved from each email.

4.3 Classification Performance

In our experiments on NB-TONE, we found that the sampling threshold c ranged from 0.01 to 0.50. We used the parameter $\varepsilon = 10^{-5}$, and the scale parameter was 2500.

We examined the difference between pure NB and NB-TONE. For NB-TONE, $c = 0.15$ and $c = 0.25$. Note that if $c = 0.5$, the algorithm degenerates into pure NB. The train% represents the overall percentage of training data. The results in Table 1 show that NB-TONE comprehensively outperforms

▼ Table 1. NB-TONE beats pure NB on trec05p and trec06p

	c	Corpus	Lam%	(1-ROCA)%	Train%
NB-TONE	0.15	trec05p	0.47	0.0284	22.91
NB-TONE	0.25	trec05p	0.54	0.0263	51.77
Pure NB	0.50	trec05p	0.72	0.0486	100.0
NB-TONE	0.15	trec06p	0.48	0.0396	17.25
NB-TONE	0.25	trec06p	0.55	0.0520	42.39
Pure NB	0.50	trec06p	0.95	0.1138	100.0

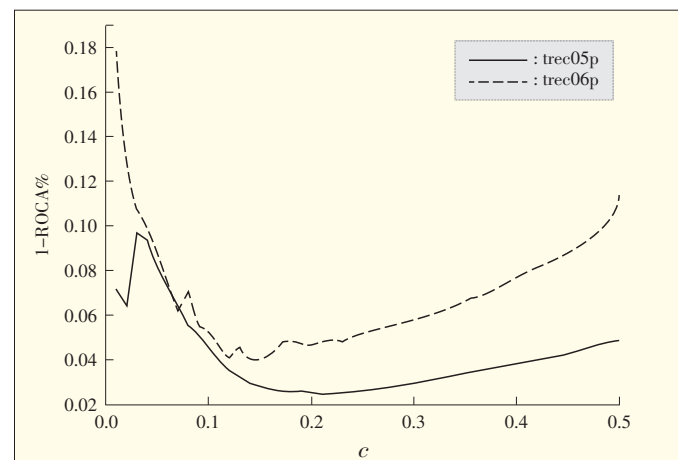
pure NB. Moreover, NB-TONE can cut down the number of training examples and reduce computational cost.

4.4 Parameter Sensitivity

Fig. 2 shows the effect of c on (1-ROCA)% performance. The results indicate that (1-ROCA)% performance varies with respect to c . From $c = 0$ to 0.15, the number of examples increases and performance improves. However, as c approaches 0.5, performance worsens.

5 Conclusion

We improved traditional online naive Bayes by using TONE. In the online process, the classifier updates itself at the appro-



▲ Figure 2. NB-TONE on data set results reported as (1-ROCA)% by threshold c .

Spam Filtering: Online Naive Bayes Based on TONE

Guanglu Sun, Hongyue Sun, Yingcai Ma, and Yuewu Shen

prate time. This method improves classification and low-confidence method and reduces the number of labels needed for high performance. Furthermore, the approach is well suited to this domain because spam filtering is inherently an online task. Our experiment shows that our NB-TONE is reliable.

References

- [1] B. Su and C. Xu, "Not so naive online Bayes spam filter," *21st Innovative Applications of Artificial Intelligence Conf.*, Pasadena, CA, July 14–16, 2009.
- [2] A. Ng and M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes," *Advances in Neural Information Processing Systems*, Canada, Dec. 9–14, 2002.
- [3] G. V. Cormack and T. R. Lynam, "TREC 2005 spam track overview," *14th Text Retrieval Conf.*, Gaithersburg, MD, Nov. 15–18, 2005.
- [4] D. Sculley and G. Wachman, "Relaxed online SVMs for spam filtering," *13th Annual ACM SIGIR Conf.*, Amsterdam, Netherlands, Jul. 23–27, 2007.
- [5] P. Graham. (2002). A plan for spam. [Online] Available: <http://paulgraham.com/spam.html>
- [6] P. Graham. (2003). Better Bayes filtering. [Online] Available: <http://www.paulgraham.com/better.html>
- [7] C. Chen, Y. Tian, C. Zhang, "Spam Filtering with Several Novel Bayesian Classifiers," *Int. Conf. on Pattern Recognition*, Tampa, FL, December 8–11, 2008.
- [8] J. Goodman and W. Yin, "Online Discriminative Spam Filter Training," *3rd Conf. on Email and Anti-Spam*, Mountain View, CA, USA, July 27–28, 2006.
- [9] H. Qi et al., "Online Linear Discriminative Learning for Spam Filter," *5th Int. Conf. Fuzzy Syst. and Knowledge Discovery*, vol. 2, Jinan, China, Oct. 18–20, 2008.
- [10] D. Tax and P. Laskov, "Online SVM learning: From classification to data description and back," *Neural Network and Signal Processing*, Toulouse, France, September 17–19, 2003.
- [11] Y. Guo and D. Schuurmans, "Discriminative Batch Mode Active Learning," *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, B.C., Canada, December 8–13, 2008.
- [12] G. V. Cormack and A. Bratko, "Batch and Online Spam Filter Comparison," *5th Conference on Email and Anti-Spam*, Mountain View, CA, USA, July 27–28, 2006.
- [13] H. Yong et al., "The Improved Logistic Regression Models for Spam Filtering," *Conf. on Asian Language Processing*, Harbin, China, December 28–31, 2009.

- [14] G. V. Cormack, "TREC 2006 Spam Track Overview," *15th Text Retrieval Conference (TREC 2006)*, Gaithersburg, Maryland, USA, November 14–17, 2006.
- [15] H. Drucker, V. Vapnik, and D. Wu, "Support Vector Machines for Spam Categorization," *IEEE Trans. on Neural Networks*, 1999, vol. 10, no. 5, 1048–1054.
- [16] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam Filtering with Naive Bayes—Which Naive Bayes?" presented at *3rd Conf. on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA, July 27–28, 2006.

Manuscript received: January 4, 2013

Biographies

Guanglu Sun (guanglu.sun@gmail.com) received his B.S., MSc, and PhD degrees in computer science from Harbin Institute of Technology. He is a professor and deputy dean of the School of Computer Science and Technology, HUST. He is also the director of the university's Information Security and Intelligent Technology Lab. Before joining HUST, he was a postdoctoral researcher at the computer department of Tsinghua University. Dr. Sun's research interests include computer networks, information security, and machine learning. He has co-edited two books and published more than 30 papers. Dr. Sun is a senior member of the China Computer Federation, a member of the IEEE Computer Society, and a member of the ACM. He is also a member of the IEEE Technical Committee on Globecom and the IEEE Technical Committee on ICC.

Hongyue Sun (sun.hongyue@zte.com.cn) received his MSc degree in electrical engineering from University of Science and Technology of China (USTC). He is the director of the ZTE's deep packet inspect and analysis team. His research interests include 3G/4G, information security, big data and machine learning. He has applied for 5 patents.

Yingcai Ma received his BS degree in computer science from HUST, China. His research interests include spam filtering and machine learning.

Yuewu Shen received his BS degree in computer science from Southeast China University. He received his MSc degree in computer science from Harbin University of Science and Technology and later joined Baidu as an R&D engineer. Mr. Shen's research interests include spam filtering, machine learning, and feature engineering.

ZTE Communications Call for Papers—Special Issue on "Cloud Computing"

Scope

Cloud computing is a consumer/delivery model where IT capabilities are offered as services to be consumed on demand. Cloud services include IaaS, PaaS, and SaaS. The underlying cloud architecture includes a pool of virtualized computing, storage, and networking resources that can be aggregated and launched as platforms to run workloads and satisfy service-level agreements. This special issue of ZTE Communications presents recent advances in cloud computing, software-defined data centers etc.

Topics

The topics include, but not limited to:

- Cloud computing platforms (IaaS, PaaS, SaaS)
- Software-defined data centers
- Cloud oriented IDC (power and cooling efficiency, DCIM, etc.)
- Cloud oriented system architecture and hardware design (server, storage, workload optimized system)
 - Cloud oriented software system (high distributed file systems, storage optimization software, global synchronized database, etc.)
 - Open source technologies (Openstack, etc.)

Paper Submission

Papers should be no more than 10,000 words or 10 pages in length and in Word format. Submissions will be reviewed anonymously by at least 3 reviewers. If possible, please ensure and equations, figures, and tables in your paper are in an editable format. Please avoid submitting equations and mathematical notation in image form.

Papers will be judged on originality, correctness, clarity and relevance. Submitted papers must be original work, and may not be under consideration for another conference or journal. Complete formatting and submission instructions can be found on http://www.zte.com.cn/endata/magazine/ztecommunications/index_5104.html.

Guest Editor

Hong Cai, PhD, CTO of Cloud Computing, ZTE Corporation
Email: hong.cai@zteusa.com

Important dates

Paper submission due: **Jul. 31, 2013**
Notification of acceptance: **Aug. 31, 2013**
Camera-ready deadline: **Sept. 15, 2013**

A System for Detecting Refueling Behavior along Freight Trajectories and Recommending Refueling Alternatives

Ye Li¹, Fan Zhang¹, Bo Gan¹, and Chengzhong Xu^{1,2}

(1. Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China;

2. Department of Electrical and Computer Engineering, Wayne State University, MI 48202, USA)



Abstract

Smart refueling can reduce costs and lower the possibility of an emergency. Refueling intelligence can only be obtained by mining historical refueling behaviors from big data; however, without devices, such as fuel tank cursors, and cooperation from drivers, these behaviors are hard to detect. Thus, detecting refueling behaviors from big data derived from easy-to-approach trajectories is one of the most efficient retrieve evidences for research of refueling behaviors. In this paper, we describe a complete procedure for detecting refueling behavior in big data derived from freight trajectories. This procedure involves the integration of spatial data mining and machine-learning techniques. The key part of the methodology is a pattern detector that extends the naive Bayes classifier. By drawing on the spatial and temporal characteristics of freight trajectories, refueling behaviors can be identified with high accuracy. Further, we present a refueling prediction and recommendation system to show how our refueling detector can be used practically in big data. Our experiments on real trajectories show that our refueling detector is accurate, and the system performs well.



Keywords

spatial data mining; trajectory processing; big data

1 Introduction

Research into smart refueling has become more important as drivers become more sensitive to fluctuating fuel prices. Previous research has been heavily focused on the use of theoretical models to analyze refueling behavior in small sets of experimental data

[1]–[5]. However, intelligence on refueling behaviors can also be directly obtained from massive trajectories. Collective refueling intelligence is useful because refueling behaviors are summarized as a result of smart decisions by experienced freight drivers. Traffic systems using collective intelligence have performed well, both in scientific research and in real-world applications [6]. However, without additional devices and cooperation from drivers, refueling records are hard to collate and are often inaccurate. Traditional methods for monitoring refueling behaviors, such as floating cursors in a fuel tank, are not widely used because of the cost of equipment and maintenance. Thus, there are not huge volumes of refueling records, and we have to turn to trajectories to obtain data. Embedded GPS equipment has become common in the freight and logistics industries.

With elaborate algorithms, patterns can be recognized in trajectories; for example, we can identify whether a car is moving or not. If a car has stopped near a gas station, it is likely refueling. However, the subtle distinction between refueling and casual parking cannot be made.

1.1 Motivations

We have designed algorithms that can accurately identify refueling stops along trajectories. To benefit drivers, we have also built a refueling prediction and recommendation system based on collective intelligence. The main motivations of our research are no additional cost and powerful collective intelligence. We design the refueling-detection algorithms only by mining mass data from trajectories, and no additional devices are required. GPS is already installed in most cars; hence, our algorithms can be widely used without having to install, test, and maintain new devices. This saves costs. We were also inspired to design our data-mining algorithms partly because of successful use of collective intelligence in taxi services. Similarly, in the logistics and freight industries, refueling behaviors derived from collective intelligence can reveal industrial advantages and disadvantages.

1.2 Challenges and Contributions

Given enough spatial, temporal, and other information, a detector can follow simple rules, including rules about spatial constraints and constraints on traveled distance, to identify potential refueling stops. However, such methods rarely work well in practice because of a lack of valid prior knowledge. One crucial problem of all naive methods is the lack of reliable prior knowledge and training data set. To our knowledge, there are no publicly available records on refueling behavior, especially for the freight industry. Some data on refueling has been collected as part of research into fuel consumption and vehicle efficiency rather than refueling behavior [1]. Without customized sensors in the fuel tanks of cars, such information can on-

A System for Detecting Refueling Behavior along Freight Trajectories and Recommending Refueling Alternatives

Ye Li, Fan Zhang, Bo Gan, and Chengzhong Xu

ly be obtained if drivers are willing to cooperate. This contributes to data inaccuracy and deficiency. Also, for rules-based decisions, prior knowledge extends further than simply routes and spatial distribution of gas stations. A solution therefore needs to be found for reliably collecting prior knowledge about refueling behavior. As well as valid prior knowledge, there are no training data sets because nothing can be directly observed from the spatial trajectories. Detectors cannot be optimized without training. Prior knowledge may offer the foundation for constructing a training data set.

The major drawback of naive methods is that they neglect the mutual influence of consecutive stops at nearby gas stations. For example, there may be two gas stations that are spatially and temporally close along a trajectory. The car has traveled such a distance from its last refueling stop that it is likely the driver will refuel at one of these gas stations. The detectors must classify one of these two eligible candidates as the refueling stop. Using a naive method, the first eligible candidate where the car firstly stopped by is classified as a refueling stop, and the result of this classification affects the other candidate. If the first stop near the first gas station is treated as the refueling stop, then the traveled distance from the last refueling stop is diminished for the next stop, and this makes the second candidate ineligible, even though the stop may be a possible alternative refueling stop [2]. The multiple hypothesis mechanism can handle this problem.

To overcome the above problems, we introduce several advanced data-mining and AI techniques, and we elaborate our algorithms. First, we propose a hill-climbing algorithm in local search and optimization. This algorithm is used to generate reliable prior knowledge about refueling behavior and construct the training data set. Then, we use a naive Bayes classifier algorithm to detect refueling stops with regard to multiple hypotheses. Last, we suggest that our detector has good extensibility by suggesting possible applications. Through abundant evaluations and data analysis, we show the robustness of our algorithms. The main contributions of this paper are

- a new method of inferring reliable prior knowledge about refueling behavior. This prior knowledge is derived from historical freight trajectories
- a method for determining freight refueling patterns based on our observations and processing of spatial trajectory data sets
- a solution to accurately detecting refueling stops along a single trajectory. This solution is based on naive Bayes.
- a real-time stream processing and batch-processing framework for processing and mining trajectory data
- a practical application for recommending refueling alternatives to drivers. This application is an extension of our detection algorithms.

The remainder of this paper is organized as follows: In section 2, we define terms used in this paper and outline problems. In section 3, we describe our methods. In section 4, we introduce a prediction and recommendation application based

on our proposed algorithms. In section 5, we describe the design of our system. In section 6, we present our experimental results. In section 7, we discuss related work. Section 8 concludes the paper.

2 Definitions

The data set for freight trajectories is similar to that for non-freight trajectories [7]. However, the interests of freight and non-freight sectors are slightly different, and both look for different patterns. This means that different measurements are used, and there are differences between freight GPS records and non-freight GPS records.

Freight GPS record: A GPS record data reported by a freight car. In our research, a GPS record r_i contains a time stamp t_s , longitude lon , latitude lat , and odometer reading odm and is given by $r_i = \{ts_i, lon_i, lat_i, odm_i\}$.

Single freight trip: An integrated, temporally continuous series of freight GPS records form a single freight trip, which is given by $T = \{r_1, r_2, \dots, r_i\}$. A freight trajectory must have a distinct origin and destination. The time stamp and odometer reading of the first freight GPS record are minimums. Typically, a trip starts after an unusually long period of parking and ends with another unusually long period of parking that is also considered the start of the next trip.

Gas station location: The gas station location is given by $gs_i = \{gsid_i, lon_i, lat_i, rfcoun_i\}$. The last attribute $rfcoun_i$ is the number of refueling behaviors that occur at a gas station within a certain timeframe of the data set. The GPS coordinates of a gas station are an approximate location generated after map digitalization.

Vehicular stop: Vehicular stops emerge along the freight trajectories. A vehicular stop contains a set of continuous GPS records. A vehicular stop has the following attributes: locations, time stamp for when the stop begins, stopping interval, and minimum bound rectangle (MBR). This vehicular stop is given by $VS_i = \{x_i, y_i, ts_i, ti_i, x_{i_{max}}, x_{i_{min}}, y_{i_{max}}, y_{i_{min}}\}$, where (X_i, Y_i) is the expected coordinates of the stop, ts is when the stop begins, and ti is how long the stop lasts. The boundary of the MBR of GPS records for this stop is given by $MBR = (x_{i_{max}}, x_{i_{min}}, y_{i_{max}}, y_{i_{min}})$. We divide the vehicular stops into the following two definitions.

Refueling stop: This is a specific kind of vehicular stop that occurs when the driver refuels during a stop. Refueling stops are distinct from other stops in that they only occur around gas stations. A refueling is given by $rs_i = \{vs_i, gs_k\}$ where the j th vehicular stop is identified as the i th refueling stop, and the driver refueled at gas station k .

Casual stop: a stop that is not a refueling stop. The main difference between a casual stop and a refueling stop is that the former may occur near a gas station or not; however, the latter occur at a gas station. A casual stop is given by $cs_i = \{vs_i\}$.

With enough prior definitions, we can now state our prob-

lem: Sets of vehicular stops extracted from trajectories must be correctly classified as refueling stops or casual stops.

In the following, we base our algorithms and applications on the definitions above.

3 Methodology

3.1 Extracting Prior Knowledge and Constructing the Training Data Set

When a given a vehicular stop is fed to the classifier, the classifier can correctly identify this stop as either casual or refueling. This is the goal of our algorithm. To optimize the classifier, reliable prior knowledge needs to be retrieved. Essential prior knowledge is factors affecting the classifier parameters. For our data set, these factors can be observed from odometer readings and temporal intervals. With prior knowledge, it is possible to predict how far a car has to travel from the refueling stop before it has to refuel. It should also be possible to predict how long the refueling stop will last.

The predicted distance a car travels from the last refueling stop before refueling is v , and the time of a refueling stop is τ . These parameters tend to converge to two constants for the same route. The greatest distance a car can drive on a full tank is mainly determined by the load of the car, the capacity of the tank, the driver's driving habits, highway conditions, and distribution of gas stations. In long-distance transport, refueling behavior is usually unique for the same driver using the same vehicle on the same route. The driver is always optimizing their decisions to gain the maximum benefit. A refueling process comprises pumping the gas and paying. Hence, the time of the whole process only decreases by a relatively small amount.

Because the patterns are known, a reasonable method is required to calculate these variables. Authentic patterns can be mined from data obtained from real refueling stops. We propose a naive method with hill climbing algorithm to approximate these variables. Algorithm 1 is the naive method for a given v .

Algorithm 1. Naive Bayes ($VS; v$)

Require:

$VS = \{vs_1, vs_2, \dots, vs_n\}$, a set of vehicular stops, when $vs_i = \{x, y, odm, t, gsdis, lastdis, r_i = 0\}$.
 v , the spatial parameter.

Ensure:

$VS = \{vs_1, vs_2, \dots, vs_n\}$, where $vs_i.r_i$ is determined.
 $RS = \{rs_1, rs_2, \dots, rs_m\}$, a set of refueling stops.
 $CS = \{cs_1, cs_2, \dots, cs_{m_2}\}$, a set of casual stops.
1: $lastodm = 0, m_1 = 0, m_2 = 0, RS \leftarrow null, CS \leftarrow null$;
2: **for** $i = 1; i \leq n$; **do**
3: $vs_i.lastdis = vs_i.odm - lastodm$;

```

4:  if  $vs_i.lastdis > v$  then
5:     $lastodm = vs_i.odm$ ;
6:     $vs_i.r_i = 1$ ;
7:     $rs_m1 = vs_i$ ;
8:     $RS.add(rs_m1)$ ;
9:     $m_1 = m_1 + 1$ ;
10:  else
11:     $cs_{m_2} = vs_i$ ;
12:     $CS.add(cs_{m_2})$ ;
13:     $m_2 = m_2 + 1$ ;
14:  end if
15: end for
16: return  $\{VS, RS, CS\}$ ;

```

The main drawback of the naive method is that it cannot distinguish between nearby refueling stops and casual stops. However, it can give reliable answers for highly distinguishable events. For example, there is a car which stops consecutively at two gas stations at a distance of more than v apart and without other stops near them. In this case, we can confidently detect two reliable refueling stops and mine the patterns. Nevertheless, we have to eliminate indistinguishable neighbors of the refueling stops.

3.1.1 Eliminating Indistinguishable Neighbors

For each refueling stop detected using the naive Bayes method, there are neighboring stops that can be either refueling stops or casual stops. These neighboring stops are located at a distance d_0 and time t_0 from the refueling stop. These neighboring stops are indistinguishable, so we proceed to the elimination of the pool of this detection and its neighbors without considering these indistinguishable stops in our hill climbing algorithm. Then, we obtain the remaining set of refueling stops RS_{remain} and casual stops CS_{remain} .

After eliminating unreliable detections, we calculate the distance between the patterns from real data v using

$$\iota = |RS_{remain}.lastdis - v| \quad (1)$$

If ι is small enough, we can confidently say that v corresponds to authentic patterns. Initially, v cannot be determined from raw trajectories data set because of the first problem mentioned in section 1. To overcome this problem, we propose a hill-climbing algorithm to approximate the best v with the lowest possible ι . The hill-climbing algorithm is shown in Algorithm 2.

Algorithm 2. Hill-Climbing ($\{VS\}, v_{init}, increment$)

Require:

$\{VS\} = \{VS_1, VS_2, \dots, VS_n\}$, a set of trajectory, when $VS_i = \{vs_1^i, vs_2^i, \dots, vs_n^i\}$
 v_{init} , an randomly initial spatial parameter.
 $increment$, a constant stands for the increment in iterations.

A System for Detecting Refueling Behavior along Freight Trajectories and Recommending Refueling Alternatives

Ye Li, Fan Zhang, Bo Gan, and Chengzhong Xu

Ensure:

v_{best} , the best spatial parameter.

```

1:  $v_{\text{best}} = v_{\text{init}}$ ;
2: while  $v_{\text{current}} \neq v_{\text{best}}$  do
3:  $v_{\text{current}} = v_{\text{best}}$ ;
4:  $\iota_1 \leftarrow \text{DistCal}(\{VS\}, v_{\text{current}})$ ;
5:  $\iota_2 \leftarrow \text{DistCal}(\{VS\}, v_{\text{current}} - \text{increment})$ ;
6:  $\iota_3 \leftarrow \text{DistCal}(\{VS\}, v_{\text{current}} + \text{increment})$ ;
7:  $v_{\text{best}} \leftarrow \text{argmin}\{\iota_1, \iota_2, \iota_3\}$ ;
8: end while
9: return  $v_{\text{best}}$ ;

```

The algorithm extracts v_{best} , which produces the smallest ι from $\{v, v - \text{increment}, v + \text{increment}\}$ in line 7. The sub-algorithm *DistCal* is shown in Algorithm 3. The eliminations process in the algorithm conforms to the definition of eliminating indistinguishable neighbors previously given.

Algorithm 3. DistCal ($\{VS\}, V$)

Require:

$\{VS\} = \{VS_1, VS_2, \dots, VS_n\}$, a set of trajectory, when
 $\{VS_i\} = \{vs_i^1, vs_i^2, \dots, vs_i^n\}$
 v , a given spatial parameter.

Ensure:

ι , the distance between real data and given spatial parameter.

```

1:  $\iota \leftarrow \text{null}$ ;
2: for  $i = 1; i \leq n; i++$  do
3:  $\{VS_i, RS_i, CS_i\} \leftarrow \text{NaiveMethod}(VS_i, v)$ ;
4: if  $RS_i = \text{null}$  then
5:  $\iota_{\text{current}} = |VS_i.allodm - v|$ ;
6: else
7:  $\{RS_{i\text{remain}}, CS_{i\text{remain}}\}$   

 $\quad \text{Eliminations}(VS_i, RS_i, CS_i)$ ;
8:  $\iota_{\text{current}} = |RS_{i\text{remain}}.lastdis - v|$ ;
9: end if
10:  $\iota \leftarrow \text{add}(\iota_{\text{current}})$ ;
11: end for
12: return  $\bar{\iota}$ ;

```

The hill-climbing algorithm starts from a randomly initiated spatial parameter v_{init} from $[v, \bar{v}]$ and then iterates until v reaches a local minimum. This algorithm can generate a reliable spatial parameter because of its convergence. The number of vehicular stops in a single trajectory with VS_i is given by n_{max} . Then, regardless of how v changes, the number of detected refueling stops n_r lies in $[0, n_{\text{max}}]$. As v decreases, n_r increases until it reaches n_{max} . However, v continues to decrease, which

makes $\iota \approx \left| \frac{VS_i.allodm}{n_{\text{max}}} - v \right|$ increase when $(VS_i \times allodm)/n_{\text{max}}$ is a constant. The results are the same when v continually increases. Hence, the best approximated v can be obtained from

the real data set as ι decreases.

3.1.2 Constructing the Training Data Set

When v_{best} has been determined, we can construct the training data set for the naive Bayes classifier. First, we use the naive method on the real data set with $v = v_{\text{best}}$. Then, we eliminate indistinguishable neighbors in the outcome $\{VS\}$ and obtain the training data set, which comprises $\{RS_{\text{remain}}\}$ and $\{CS_{\text{remain}}\}$. The set $\{RS_{\text{remain}}\}$ is the set of positive examples in the naive Bayes classifier, and $\{CS_{\text{remain}}\}$ is the set of negative examples in the naive Bayes classifier.

3.2 A Naive Bayes Classifier with Multiple Hypotheses

Detecting refueling stops can be thought of as a classification problem. An event is unambiguously classified according to evidence derived from observations. A classifier should be able to correctly identify whether a given vsi is a refueling stop or casual stop by observing the time of the stop and distances traveled from last refueling.

3.2.1 A Naive Bayes Classifier Framework

We use a naive Bayes classification framework [8]–[12], which is based on Bayes' formula [13], [14] and is given by

$$P(Y|X) = \frac{P(Y) \prod_{i=1}^d P(X_i|Y)}{P(X)} \quad (2)$$

where Y is the class variable and X is the attribute for each X_i comprising d attributes [9]. From our observations, $X = \{x_1, x_2\}$, where $d = 2$, $x_1 = vs_i \times \text{lastdis}$ and $x_2 = vs_i \times t_i$; and $Y = \{y_1, y_2\}$, where $y_1 = 0$ is a casual stop, and $y_2 = 1$ is a refueling stop. Then, classification problem can be briefly expressed as $vs_i \times t_i = \text{argmax}_{y_i} (P(Y = y_i|X))$. However, before using (2), $P(Y)$,

$P(Y|X)$ has to be calculated first. In all circumstances, $P(X)$ is a constant [8], [9].

Given a raw data set, $P(Y)$ can be directly observed and calculated. However, construction of training data set requires indistinguishable neighbors to be eliminated. Hence, we have to eliminate the effects of data loss. There are n_p positive examples and n_n negative examples, so we know that we have eliminated n_r indistinguishable stops, of which there are n_r refueling stops that have been detected using the naive Bayes method.

Then, we can calculate $P(Y = y_1) = \frac{n_n + n_v - n_r}{n_p + n_n + n_v}$ and

$$P(Y = y_2) = \frac{n_p + n_r}{n_p + n_n + n_v}.$$

If we assume Gaussian distributions of continuous attributes x_1, x_2 (section 3.1), the class-conditional probability for any attribute X_i can be expressed as

$$P(X_i = x_i|Y = y_j) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp \left(-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2} \right) \quad (3)$$

where μ_{ij} can be estimated using the sample mean \bar{x}_i of X_i for

all training data that belong to the class y_j , and σ_{ij}^2 can be estimated using the sample variance of the same subset of training data [8].

The naive Bayes classifier is suitable for detecting refueling stops based on similar observations from our data set and the characteristics of the naive Bayes classifier [9].

The naive Bayes Classifier is not sensitive to isolated noise points because the class-conditional probabilities are estimated over all data. A single observation of location data may be erroneous due to errors in the GPS receivers, odometer, and other sensors. However, these noises are eliminated by averaging out.

The naive Bayes classifier is not sensitive to irrelevant attributes. In our data set, the distance from last refueling and the length of time spent at a stop are only loosely related.

3.2.2 Multiple-Hypotheses Framework

A naive Bayes classifier does not yet solve the problem of indistinguishable neighbors. We propose a framework that combines multiple hypotheses with a naive Bayes classifier to overcome this problem.

For a set of stops and neighboring stops (including both refueling and casual stops) within a distance of d_0 and time t_0 , it needs to be determined whether there is only refueling stop in a set or whether there is no refueling stop in a set.

For a set of neighboring stops VS of sizes m , there are $m+1$ possible results from the following equation:

$$P_k' = \begin{cases} \prod_{a=1}^m P(Y^a = y_1 | X^a), & k = 0 \\ P(Y^k = y_2 | X^k) \prod_{a=1, a \neq k}^m P(Y^a = y_1 | X^a), & k \neq 0 \end{cases} \quad (4)$$

where $k = 0$ indicates that none of m is a refueling stop, and $k \neq 0$ indicates that the k th vehicular stop is a refueling stop. We solve the indistinguishable neighbor problem by calculating all possible situations in (4) and determining the highest probability by using $\arg\max_k \{P'\} 1 \times (m+1)$.

Algorithm 4 the Naive Bayes Classifier and Multiple Hypothesis Combined Framework

Require:

$VS = \{vs_1, vs_2, \dots, vs_n\}$, when $vs_i = \{x, y, odm, ti, gsid, gsdis, lastdis, ri = 0\}$.
 $\{\mu_{ij}\}, \{\sigma_{ij}\}$, the set of parameters from the training data.
 d_0, t_0 , two parameters to determine neighbors.

Ensure:

$S = \{vs_1, vs_2, \dots, vs_n\}$, when $vs_i.ri$ is determined.

```

1: for  $i = 1; i \leq n; i++$  do
2:    $NeighborSet_{i \times m} \leftarrow GetNeighborSet(vs_i);$ 
3:   if  $NeighborSet_{i \times m} = null$  then
4:      $vs_i.ri = \arg\max_{y_j} (P(Y=y_j | X));$ 
5:   else
6:      $k = \arg\max_k \{P'\} 1 \times (m+1);$ 

```

```

7:   for  $j = 0; j < m; j++$  do
8:     if  $j+1 \neq k$  then
9:        $vs_{i+j}.ri = y_1 = 0;$ 
10:    else
11:       $vs_{i+j}.ri = y_2 = 1;$ 
12:    end if
13:  end for
14:   $i = i + m;$ 
15: end if
16: end for
17: return  $VS;$ 

```

4 The Application

In this section, we describe an application that uses the previously mentioned algorithms to output reliable indications of refueling behavior. Combining gas station background information and collective intelligence derived from drivers, the application, called refueling prediction and recommendation, provides optimized refueling choices. The application is straightforward. With well-designed logical models, we show how refueling behavior can be determined by mining driver intelligence from freight trajectory data.

Given real-time information about the car, and given background information from other sources, the application tells a driver when and where they should refuel.

The real-time information of a car changes when the car needs refueling. This information includes the distance from where the car last refueled. This attribute is exactly the same as that used in the above algorithms. However, here it is monitored in real time in order to provide instant evidence for our recommendation. The real-time spatial attribute of a specific time stamp t is d_t . The statistical spatial constant parameters μ_c and σ_c are the successions of the definition in section 3.2.1. These constant parameters can be computed from the set of determined refueling stops by using the above algorithms.

Background information can also dominate the recommendation. Background information has two major parameters which represent the popularity of one gas station and the distance of one gas station from the real-time location of one car. The popularity of one gas station can be obtained by using the Naive Bayes Classifier with Multiple Hypothesis Algorithm on a massive data set and then collecting the results. It emits a possibility show that how likely a refueling will happen on one specific gas station. And the distance of one gas station from the instant location of one car can be used to estimate how likely the car will take a refueling action when it arrives at this gas station, in other words, refueling prediction. The popularity of a gas station i and the distance between this gas station and the car's instant location at time t are denoted by P_{gi} and d_{ti} , respectively.

By integrating real-time and background information, we can model the problem in this prediction and recommendation system. Given d_t of a car at time t , what is the probability P_{gt}

A System for Detecting Refueling Behavior along Freight Trajectories and Recommending Refueling Alternatives

Ye Li, Fan Zhang, Bo Gan, and Chengzhong Xu

that the car will refuel at gas station i with background information P_{gi} and d_{ii} ? This probability can be computed using

$$P_{gi} = P_{gi} \times \frac{1}{\sqrt{2\pi}\sigma_c} \exp\left(-\frac{(d_i+d_{ii}-\mu_c)^2}{2\sigma_c^2}\right) \quad (5)$$

In order to recommend a specific gas station to this car at time t , we can compute (5) for all gas stations and recommend the gas station with the highest probability $\arg\max_i\{P_{gi}\}$. In this real application, we construct the set of candidate gas stations with direction and distance thresholds in order to eliminate invalid candidates for efficiently computing (5).

5 System Framework

Our framework of a refueling detection and recommendation/prediction system comprises data preprocessor, knowledge discovery, and intelligence generator (Fig. 1). This design is general and scalable. We embed our refueling detector and recommendation/prediction into the framework.

The data preprocessor contains a spout and bolts in a Storm cluster that is especially designed to preprocess data from spatial trajectories. The spout continually emits GPS measurements; however, the level of the bolts is determined by logical complexity of the preprocessing tasks. In our system, these bolts clean unqualified GPS records, reorganize the trajectory, and extract vehicular stops. This module constantly receives GPS records and transforms them into trajectories, and the last bolts extract vehicular stops along each trajectory. The output of the last bolts is fed to the next knowledge-discovery module.

The knowledge-discovery module is built on the Hadoop batch-processing platform. This module is mainly responsible for carrying out the most computationally expensive tasks in the system. The batch-processing platform runs these tasks routinely, or only in certain conditions, at low frequency and with latency, unlike the real-time data preprocessor and intelligence generator components. In our case, the naive Bayes classifier with multiple hypotheses algorithm is implemented

in MapReduce and run in the Hadoop clusters. The output of algorithms is stored in database systems as knowledge and experience extracted from big data.

The intelligence generator comprises real-time applications that interact with users. It also comprises a spout and bolts, which are located inside the Storm cluster. The intelligence generator and data preprocessor may even share the same spout. However, the data preprocessor digests all the data from the source while the intelligence generator extracts and purifies real-time information from the data source with some of bolts to handle data selections and transformations. Further, after real-time information is captured, the application in a bolt is connected to database systems where knowledge and experience are stored. By combining real-time information and background knowledge, applications in the intelligence generator can accurately provide recommendations in real time. In our case, the bolts in the intelligence generator monitor real-time information for each car, and our recommendation/prediction algorithm is applied to every bolt. Then, a recommendation is made as to whether the driver should refuel.

These three components are highly integrated and are all built on scalable, fault-tolerant distributed systems on Hadoop and Storm platforms. Comprehensive use of these platforms means our system can provide both real-time services and complex data-mining services. Additional applications can be easily integrated into the existing framework in the form of Hadoop MapReduce jobs or Storm bolts, taking the advantage of reusable resources.

6 Performance Evaluation

6.1 Data Descriptions

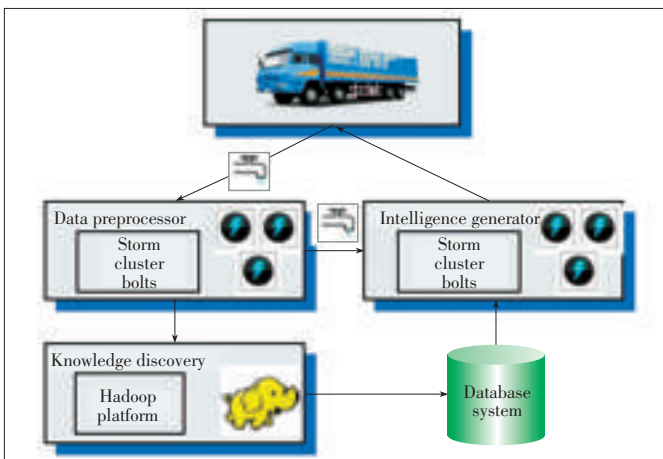
The GPS data set was collected from 14,800 cars involved in the logistics and freight industries in China. Each car constantly reported its location about every 15 s. The data was collected from October 2009 to April 2012, and real-time data is still incrementing by several Gigabytes every day. Tables 1 shows the data information of the dataset, Tables 2 shows the fields in each record.

Because of the variety of freight routes, our trajectory data set covers most of China. Fig. 2 shows an example route from our data set. This route originates in Shenzhen and ends in Shanghai. The driving distance is about 2000 km.

Besides, we have location information of gas stations all around the country. These massive data of gas stations contain location records of 114,000 stations in China. The number of gas stations matches our knowledge of the number of all gas stations in China. Fig. 3 shows the spatial distributions of all these gas stations.

6.2 Experiment Settings

In the initial part of our experiment, we selected a part of



▲ Figure 1. Integrated system framework.

A System for Detecting Refueling Behavior along Freight Trajectories and Recommending Refueling Alternatives

Ye Li, Fan Zhang, Bo Gan, and Chengzhong Xu

▼ Table 1. Dataset descriptions

Data information	
Name	GPS location report
Number of cars	14,800
Time spanning	2009.10–2012.04
Volume	300 Gigabytes

▼ Table 2. Data fields

Data fields	
Longitude	Float
Latitude	Float
Odometer reading	Float
Time stamp	Datetime



◀ Figure 2. A route from Shenzhen to Shanghai.

our data set of 1000 cars. This raw data was 12 GB. First, we input this data into the data preprocessor module and collected the output of several vehicular stops from different routes. Second, we ran our Naive Bayes Classifier with Multiple Hypothesis algorithm with $v_{init} = 700$. Then, we obtained the result of a set of determined refueling stops with spatial statistical parameters μ_c and σ_c . Third, we randomly chose 3000 instant records from different cars and ran our prediction and recommendation application. The input for was instant information from these cars and background information generated in previous procedures. By the end of the three stages, we had obtained 3000 instant records from different cars and routes and recommended a gas station for each car.

6.3 Performance Evaluation

Given the 3000 instant records and 3000 gas stations generated by our application, we designed an indicator called *drift* to measure the differences between ground truth and assigned recommendations. For the i th instant record, $gr(i)$ is the recommended gas station, and $gt(i)$ is the actual gas station where the car refueled. Then, $drift_i$ can be obtained for the i th instant record:

$$drift_i = distance(gr(i), gt(i)). \quad (6)$$

The $distance(gr(i), gt(i))$ function returns driving distances, recommended gas station $gr(i)$, and actual gas station $gt(i)$, which are calculated from historical trajectories instead of straight-line distances.

Fig. 4 shows the cumulative distribution function of $drift_i$ measurements for all 3000 samples. This result shows that our recommendation/prediction system is precise, and nearby alternative gas stations can be accurately recommended for refueling. More than 50% of our recommendations were close to the

actual gas stations with no drifts. More than 60% of the recommendations drifted within 100 kilometers. Almost 85% of recommendations drifted within 300 kilometers. Assuming an average speed of 120 km/h on highway in China, we provided 85% of users with alternative refueling gas stations within a driving time of around 2.5 hours. This is acceptable because the assigned alternative gas stations are always more attractive to drivers because they provide better services.

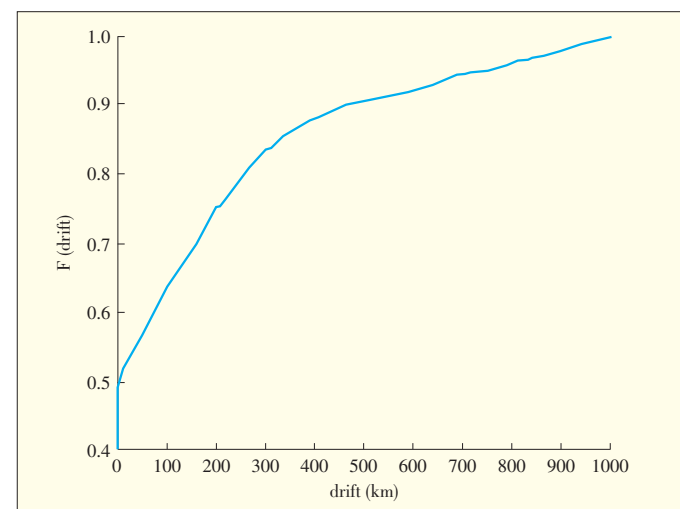
However, 15% of recommendations were far away from the actual refueling spots. These should be considered failed recommendations. Possible causes of failed recommendations are unavoidable random corruption in raw GPS records and insufficient experimental data. Although some unqualified records could be removed in the data-cleaning process, defective odometer readings and abrupt discontinuance of trajectories affect the precision of the refueling detector. This may have a detrimental impact on our recommendation system. A partial data set in the big data may cause bias in when calculating the popularity of gas stations. This leads to a lack of equilibrium in the statistical distribution. Accuracy can be improved by dropping inaccurate raw data and adding more reliable raw data in trade of larger system scale.

7 Related Work

Much previous research has been centered on obtaining



Figure 3. ▶ Spatial distribution of gas stations in China.

▲ Figure 4. Cumulative distribution of *drift*.

A System for Detecting Refueling Behavior along Freight Trajectories and Recommending Refueling Alternatives

Ye Li, Fan Zhang, Bo Gan, and Chengzhong Xu

knowledge from large numbers of trajectories and has resulted in the creation of route recommendation systems and traffic prediction systems [6], [7]. Research on refueling behavior has also been done [1], [3]–[5], [8], [10], [11], [13].

In [1], refueling behavior of gasoline-car drivers was analyzed, and it was found that refueling behavior is strongly correlated to characteristics how a car is driven. This research spurred us to use algorithms that detect refueling stops based on the driving distance. Analysis of refuel availability in [3], [4] model of optimization of refueling stations locating in [5] presented research on spatial distribution of gasoline stations and quantified refuel availability in order to optimizing the locating of gasoline stations. Our refueling recommendation/prediction system evaluates refueling availability with spatial distributions of gasoline stations combined with their popularity among all drivers. However, such previous research work did not use a huge amount of trajectories data, which means that these research results limited in a small range both spatially and temporally. In comparison, our system is based on a big volume of real trajectories data. Moreover, we introduced the classification methodologies for data mining [8]–[11] into our algorithms to achieve high accuracy under the scope of data uncertainty.

A smart route service system for taxicab drivers based on a large amount of historical taxi trajectories was proposed in [6], as well as a graph model to estimate traffic conditions. Our research work shared the same designing ideas with [6] in the data preprocessing model. However, we further integrated batch processing platform Hadoop and real-time stream processing platform to process big data for real-time applications. Our architecture and framework have a systematically advantage in generality and scalability.

8 Conclusion

In this paper, we have integrated machine-learning and data-mining techniques to create a refueling detection algorithm called Naive Bayes Classifier with Multiple Hypotheses. This algorithm detects whether a car has refueled. We have also designed a practical application to recommend refueling alternatives. This application is based on real-time information taken from a vehicular trajectory and background knowledge about the distribution of gas stations and driver experience. Furthermore, we have developed a general framework for integrating Storm real-time processing platform and Hadoop batch-processing platform in order to process and data mine the trajectory under the scope of real-time big data. Our experiment shows that, our recommendation/prediction system recommends acceptable gas station alternatives with low drifts.

Acknowledgment

This work was supported by a grant from the Science Technology and Innovation Committee of Shenzhen Municipality.

References

- [1] R. Kitamura and D. Sperling, "Refueling behavior of automobile drivers," *Transportation Research Part A: General*, vol. 21, no. 3, pp. 235–245, 1987.
- [2] M.A. Nicholas, "Driving demand: What can gasoline refueling patterns tell us about planning an alternative fuel network?" in *J. Transport Geography*, vol. 18, no. 6, pp. 738–749, 2010.
- [3] M. Melaina and J. Bremson, "Refueling availability for alternative fuel vehicle markets: Sufficient urban station coverage," in *Energy Policy*, vol. 36, no. 8, pp. 3233–3241, 2008.
- [4] D. L. Greene, "Survey evidence on the importance of fuel availability to the choice of alternative fuels and vehicles," in *Energy Studies Review*, vol. 8, no. 3, p. 2 1998.
- [5] Y. W. Wang and C. C. Lin, "Locating road-vehicle refueling stations," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, no. 5, pp. 821–829, 2009.
- [6] J. Yuan, Y. Zheng, L. Zhang, X.I. Xie, and G. Sun, "Where to find my next passenger" in *Proc. 13th ACM Int. Conf. on Ubiquitous Comput. (UbiComp '11)*, 2011.
- [7] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.Z. Sun, "An interactive-voting based map matching algorithm," in *Proc. 11th Int. Conf. on Mobile Data Management (MDM '10)*, pp. 43–52.
- [8] J. Han and M. Kamber, *Data mining: concepts and techniques*. Morgan Kaufmann Publishers, 2006.
- [9] P. N. Tan et al. *Introduction to data mining*. Pearson Education India, 2007.
- [10] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *Proc. Nat. Conf. on Artificial Intelligence*, 1992, pp. 223–223.
- [11] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2, pp: 103–130, 1997.
- [12] M. Ramoni and P. Sebastiani, "Robust Bayes classifiers," *Artificial Intelligence*, vol. 125, no. 1, pp: 209–226, 2001.
- [13] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern classification and scene analysis 2nd ed.* New York: Wiley, 1995.
- [14] R. Sheldon et al. *A first course in probability*. Pearson Education India, 2002.

Manuscript received: May 18, 2013

Biographies

Ye Li (li.ye@siat.ac.cn) is a research assistant at Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. His research interests include big-data analysis and spatial data mining. He received his BS degree in geographic information science from South China Normal University in 2012.

Fan Zhang (zhangfan@siat.ac.cn) is an associate professor at the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. Dr. Zhang's research interests include cloud computing, big-data analysis, data security, and privacy protection. He received his BS, MS, and PhD degrees in electronic and information engineering from Huazhong University of Science and Technology in 2002, 2004 and 2007.

Bo Gan (bo.gan@siat.ac.cn) is a visiting research student at Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. His research interests include big-data analysis. He is studying his MS degree in communication and information systems at Wuhan University of Technology.

Chengzhong Xu (cz.xu@siat.ac.cn) is a professor of electrical and computer engineering at Wayne State University. He is director of the Cloud and Internet Computing Laboratory (CIC), Wayne State University, and director of the Cloud Research Computing Center, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. Dr. Xu's research interests include resource management for improving the performance, reliability, availability, power efficiency, and security of networked computing systems. The systems of particular interest to Dr. Xu are distributed systems and the internet, servers and datacenters, scalable parallel computers, and wireless embedded devices. He is an editor of a number of journals, including *IEEE Trans. on Parallel and Distributed Systems (TPDS)* and *Journal of Parallel and Distributed Computing (JPDC)*. Dr. Xu obtained his BSc and MSc degrees in computer science and engineering from Nanjing University in 1986 and 1989. He received his Ph.D degree in computer science and engineering from the University of Hong Kong in 1993.