Journal of Communications and Information Networks
Vol.1, No.4, Dec. 2016
DOI: 10.11959/j.issn.2096-1081.2016.051

Research paper

# Smart data deduplication for telehealth systems in heterogeneous cloud computing

GAI Keke[1], QIU Meikang[2], SUN Xiaotong[1], ZHAO Hui[3]

1. Department of Computer Science, Pace University, New York 10176, USA
2. College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
3. Software School, Henan University, Kaifeng 475000, China

**Abstract:** The widespread application of heterogeneous cloud computing has enabled enormous advances in the real-time performance of telehealth systems. A cloud-based telehealth system allows healthcare users to obtain medical data from various data sources supported by heterogeneous cloud providers. Employing data duplications in distributed cloud databases is an alternative approach for achieving data sharing among multiple data users. However, this approach results in additional storage space being used, even though reducing data duplications would lead to a decrease in data acquisitions and real-time performance. To address this issue, this paper focuses on developing a dynamic data deduplication method that uses an intelligent blocker to determine the working mode of data duplications for each data package in heterogeneous cloud-based telehealth systems. The proposed approach is named the SD2M (Smart Data Deduplication Model), in which the main algorithm applies dynamic programming to produce optimal solutions to minimizing the total cost of data usage. We implement experimental evaluations to examine the adaptability of the proposed approach.

**Key words:** data deduplication, telehealth, heterogeneous cloud computing, optimal solution, dynamic programming

**Citation**: GAI K K, QIU M K, SUN X T, et al. Smart data deduplication for telehealth systems in heterogeneous cloud computing[J]. Journal of communications and information networks, 2016, 1(4): 93-104.

## 1 Introduction

Heterogeneous cloud computing has been the driving force behind the dramatic growth of various distributed applications, including telehealth systems[1,2]. Multiple emerging techniques have resulted in the development trends of cloud-based telehealth systems becoming increasingly diverse, depending on the service demands and the technologies that are employed[3,4]. Data duplication is one of the problems presented by distributed data storage on the cloud side. In general, for the purpose of data recovery, data are repeatedly stored on different servers that are geographically distributed[5–11]. However, this approach leads to large volumes of additional data being stored with a concomitant waste of storage capacity, which not only causes overconsumption of computing resources but also has a negative impact on the environment [12, 13].

Manuscript received Sept. 14, 2016; accepted Oct. 02, 2016
This work is supported by the National Natural Science Foundation of China (No. 61672358).

Therefore, the need to reduce storage waste caused by unnecessary repeated data storage is urgently required for the contemporary deployment of cloud-based telehealth systems[14]. This requirement is also related to the data collection features of many telehealth applications, such as the consistent generation of sensor data and continuous data updates[15–17]. These aspects of data generations can lead to increasingly large data storage capacity needs when the number of users in telehealth systems becomes large, even though some of the stored data are rarely used. The main challenge is finding a solution to reduce the storage space used by redundant data when applying distributed data duplication techniques for ensuring data security[18].

To solve this problem, we propose our solution named the SD2M (Smart Data Deduplication Model). We consider data deduplication to be an effective alternative technique for achieving an efficient and functional data storage mechanism. Our approach is based on a method we developed to create optimal solutions by assigning data packages to distributed cloud databases. The data package distribution depends on a few parameters, including the execution time, hash collision probabilities, and storage constraints. Data distribution management is a difficult task, since all these parameters used for creating data deduplication plans are variables such that the objective effectively translates into solving an NP hard problem.

Fig.1 represents the architecture of SD2M for telehealth systems. It shows that our solution involves adding a new cloud-based functional unit named the SA (Smart Assigner). All data collected from telehealth sensors are intended to be sent to the SA before data duplication operations commence. The SA applies an optimal algorithm such that the acquisition of the data duplication plan is based on addressing multiple constraints. After filtering by the SA, data are stored in a distributive manner in the cloud servers, to which multiple telehealth applications involved in a user group would have access. Because of performance concerns, our approach is not intended to remove all repeated data storage; thus, data duplication would only be prevented to a partial extent. The data deduplication plan depends on the available storage resources, which will maximize the storage performance by considering both the usage frequency and the type of content.

The main contributions of our work include:

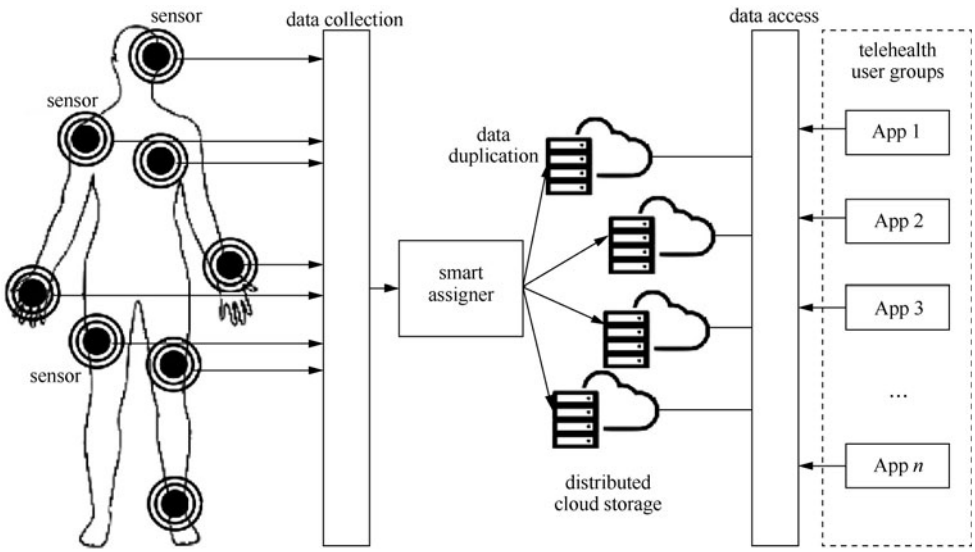


**Figure 1** Architecture of smart data deduplication model for tele-health systems

1) We propose an approach using optimal solutions to maximize the value of the stored data when considering the hash collision probabilities and storage availabilities under the time constraints.

2) Our solution is a performance-oriented implementation for current cloud-based telehealth systems to achieve real-time services when constrained cloud storage is applied.

3) The problem solved in our work is an NP hard problem. The core algorithm used in our proposed approach is based on dynamic programming, which is designed to create optimal solutions for the proposed problem.

The remainder of this paper follows the structure below. Section 2 synthesizes updated findings from recent relevant research work. Section 3 describes a motivational example that explains the mechanism of the proposed approach. Section 4 defines the major problem and illustrates our proposed algorithms. We present our experimental evaluations in Section 5. Finally, Section 6 concludes our work.

## 2 Related work

In this section we review data duplication and deduplication to acquire a solid theoretical foundation and to distinguish our work from prior research. Data deduplication, an optimization technique for data storage, is often considered a form of intelligent compression that entails only storing unique data rather than retaining large volumes of redundant data carrying the same content. The main trade-off between data duplication and deduplication involves a few aspects, including data recovery and security, efficiency, and storage capacity. We address these four main aspects in our review and briefly describe the ability of our approach to successfully address these matters.

Contemporarily, cloud service providers intend using distribution-based data duplication methods to guarantee data recovery, which is also one of the core values of cloud services[19, 20]. One of the approaches is to use a routing algorithm to reduce the amount of redundant data. For example, Boafft[21] is a method that uses data similarities to remove unnecessary data from cloud storage systems. Our method follows a different approach in that it intends to ensure data retrieval performance. Any data removals may lead to a reduction in data acquisition, even though the data can be obtained by the correct pointers. Thus, our aim is to partially remove abundant data to achieve maximum performance using the available resources.

Other researchers concentrated on performance-oriented solutions[22,23] rather than reducing storage costs[24]. Mao, et al[25] proposed an approach that used two-pronged methods to reduce the overhead associated with deduplication. Data selection for duplication purposes depends on a few parameters, such as read/write traffic and the memory management scheme. Xia, et al[26] developed a method that used a near-exact approach by comparing data similarities to eliminate a high degree of duplication. However, this approach could not achieve optimal solutions and would require further improvement. Contrary to this approach, our solution could provide optimal solutions with the additional advantage that the considered parameters were distinct.

Another research group showed that data deduplications can be achieved by estimating data usage. Fu, et al[27] proposed an application-aware data deduplication method that depended on the local application usage for determining the data storage requirements. This work proved that data can be classified in terms of the usage determined by application configurations. Our approach uses a similar mechanism and produces a data deduplication plan based on data utilization.

Recent studies were devoted to exploring security issues[28–30] resulting from dynamic updates of data deduplications[31,32]. Other scholars attempted to

address these security concerns by developing a convergent key scheme. Wen, et al[33] presented a session-key-based security mechanism to prevent data from being exposed to adversarial activities when data deduplications required frequent updates. This scheme used a convergent key-sharing method to achieve group combinations, together with avoiding the use of gateways, to ensure data users can verify the correctness of the data. Li, et al [34]proposed another method of secure convergent key management. This approach outsourced key management to the cloud side, thereby reducing the risk by obviating the need for data users to manage any keys. These previous explorations provided theoretical support for securing our deduplication approach.

In summary, our approach is distinct from all prior related work, and constitutes an improvement based on other researchers' findings. Our work focuses on fully applying available storage resources and maximizing the performance by partially reducing the amount of redundant data. The proposed approach uses an optimal algorithm that is described in the following sections.

## 3 Motivational example

This section provides a motivational example that describes the working mechanism of the proposed model. There are two options for data packages, namely, Non-Dedu (Non-Deduplication) and Dedu (Deduplication) options. Each option has two working modes. For Non-Dedu, the first working model is to store the data without using distributed storage; the other working mode is to store the data by using distributed storage. For Dedu options, the first working model is to use a weak hash table for achieving deduplication; the other working mode is to use a strong hash table. A strong hash table is expected to lower the risk of hash collisions, even though the approach is more time consuming. Tab.1 presents a parameter-mapping table that displays all parameters in the four working modes for each input data package.

Our objective is to output all optimal solutions to minimize the hash collision probabilities that match the requirements of the telehealth applications within the potential execution time range. We achieve this goal by transforming Tab.1 into Tab.2, which is a machine-friendly input table. We name the input table the B-Table. In the table, we pair up the hash collision probabilities with their corresponding required cloud storage space. As seen in Tab.2, data package $D_1$ has two working modes when the available time length is 5, including (0.1, 30) and (0.4, 2). It means that the data package can be stored with either a 0.1 hash collision probability attaching to 30-unit cloud storage or a 0.4 hash collision probability attaching to 2-unit cloud storage.

In addition, we start creating a D-Table derived from the B-Table. The process of creating the D-Table depends on the sequence of the data package inputs. In this example, we assume that the order in which data packages are input is from $D_1$ to $D_4$. The first row is created when data package $D_1$ inputs into the D-Table. We keep the pairs of optimal solutions to a minimum to minimize the hash collision probabilities and the required cloud storage. The second row is created when the second data package $D_2$ is added and only optimal solution pairs are stored in the table. The same processes continue until all data packages are added to the table.

In addition, we use a backward-forward tracking method to produce optimal solutions, according to the requirements. Fig.2 represents an example of creating a backward-forward track for optimal solutions under timing constraint 10. In this example, we point out that there might be a variety of pairs that can be used as output. The cloud telehealth applications can use any pair listed in the table in terms of the application configurations. We configure three probability ranges

for solution selections in this provided motivational example. The first, second, and third ranges are (−∞, 0.02], (−∞, 0.04], and (−∞, 0.01], respectively, which means that the highest probability tolerance is 0.02, 0.04, and 0.01, respectively.

In order to represent the tracks in the backward direction, we use boxes of different colors to mark the selected pair and lines with arrows to represent the backward-forward track. The boxes and arrow lines in red, green, and blue color

**Table 1** Parameter table for mapping the four working modes. (WM/$M_i$: Working Mode; Non-Dedu: Non Deduplication; Dedu: Deduplication; $T$: Time; Pr: Hash Collision Probability; St: Required Storage Space Level; $D_i$: Data Package)

| data | WM | Non-Dedu | | | Dedu | | |
|---|---|---|---|---|---|---|---|
| | | $T$ | Pr | St | $T$ | Pr | St |
| $D_1$ | $M_1$ | 2 | 0.5 | 10 | 3 | 0.4 | 2 |
| | $M_2$ | 5 | 0.1 | 30 | 6 | 0.1 | 2 |
| $D_2$ | $M_1$ | 1 | 0.6 | 5 | 2 | 0.3 | 1 |
| | $M_2$ | 4 | 0.1 | 15 | 7 | 0.05 | 1 |
| $D_3$ | $M_1$ | 2 | 0.7 | 15 | 3 | 0.5 | 3 |
| | $M_2$ | 7 | 0.2 | 45 | 8 | 0.15 | 3 |
| $D_4$ | $M_1$ | 1 | 0.8 | 12 | 2 | 0.6 | 2 |
| | $M_2$ | 3 | 0.1 | 30 | 6 | 0.1 | 2 |

**Table 2** B-Table: mapping table for inputs

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $D_1$ | — | (0.5, 10) | (0.4, 2) | (0.4, 2) | (0.1, 30)<br>(0.4, 2) | (0.1, 2) | — | — |
| $D_2$ | (0.6, 5) | (0.3, 1) | (0.3, 1) | (0.1, 15)<br>(0.3, 1) | (0.1, 15)<br>(0.3, 1) | (0.1, 15)<br>(0.3, 1) | (0.05, 1) | — |
| $D_3$ | — | (0.7, 15) | (0.5, 3) | (0.5, 3) | (0.5, 3) | (0.5, 3) | (0.2, 45)<br>(0.5, 3) | (0.15, 3) |
| $D_4$ | (0.8, 12) | (0.6, 2) | (0.1, 30)<br>(0.6, 2) | (0.1, 30)<br>(0.6, 2) | (0.1, 30)<br>(0.6, 2) | (0.1, 2) | — | — |

| $T$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | … |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | - | (0.5,10) | (0.4,2) | (0.4,2) | (0.1,30)<br>(0.4,2) | (0.1,2) | … | … | … | … | … |
| $D_2$ | | | (0.3,15) | (0.15,11)<br>(0.24,7) | (0.12,3) | (0.05,25)<br>(0.15,11<br>(0.24,7)) | (0.03,31)<br>(0.04,17)<br>(0.06,7)<br>(0.12,3) | (0.03,3) | (0.01,45)<br>(0.025,11)<br>(0.12,3) | (0.01,17)<br>(0.02,3) | … |
| $D_3$ | | | | | (0.21,30) | (0.105,26)<br>(0.15,18) | (0.075,14)<br>(0.12,10) | (0.035,40)<br>(0.06,6) | (0.021,46)<br>(0.025,28)<br>(0.042,22)<br>(0.075,14)<br>(0.12,10) | (0.015,34)<br>(0.02,20)<br>(0.02,18)<br>(0.03,10)<br>(0.06,6) | … |
| $D_4$ | | | | | | (0.168,42) | (0.084,38)<br>(0.12,30) | (0.021,60)<br>(0.06,26)<br>(0.09,20) | (0.010 5,56)<br>(0.015,48)<br>(0.045,16)<br>(0.072,12) | (0.007 5,44)<br>(0.012,40)<br>(0.033 6,34)<br>(0.036,8) | … |

**Figure 2** Backward-forward tracking to create optimal solutions by using dynamic programming

represent the optimal solution tracks for the first range (−∞, 0.02], the second range (−∞, 0.04], and the third range (−∞, 0.01], respectively. The corresponding probabilities are 0.012, 0.036, and 0.007 5, respectively, with the required cloud storage of 40, 8, and 44, respectively.

Tab.3 represents all optimal solution pairs shown in the output D-Table. The cloud-based telehealth applications can determine the plan of data deduplication based on the hash collision probability tolerance and available cloud storage space. The next section describes the main concepts in our proposed model, as well as the main algorithm.

## 4 Concepts and the proposed algorithm

We illustrate the proposed problem in Definition 1 that defines the inputs, output, and objective of the problem.

**Definition 1** Minimizing storage and probabilities (MSP) problem: Inputs include the number of the input data packages $N_d$, with each data package represented as $D_i$ ), execution time of a data package under working mode $M_i$ for non-deduplication option $T_{M_i}^{\text{Non-Dedu}}$, hash collision probability rate of a data package under working mode $M_i$ for non-

**Table 3** Example of D-Table listing all optimal solutions under all potential timing constraints for the motivational example. Time range: 6~27.

| T | optimal solutions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 6 | (0.168, 42) | | | | | | | | |
| 7 | (0.084, 38) | (0.12, 30) | | | | | | | |
| 8 | (0.021, 60) | (0.06, 26) | (0.09, 20) | | | | | | |
| 9 | (0.010 5, 56) | (0.015, 48) | (0.045, 16) | (0.072, 12) | | | | | |
| 10 | (0.007 5, 44) | (0.012, 40) | (0.033 6, 34) | (0.036, 8) | | | | | |
| 11 | (0.003 4, 70) | (0.006, 36) | (0.015, 30) | (0.024, 22) | (0.036, 8) | | | | |
| 12 | (0.002 1, 76) | (0.002 5, 58) | (0.004 2, 52) | (0.007 5, 44) | (0.009, 36) | (0.010 5, 28) | (0.012, 18) | (0.018, 12) | (0.036, 8) |
| 13 | (0.001 5, 64) | (0.002, 50) | (0.002 1, 48) | (0.003, 40) | (0.006, 36) | (0.007 5, 16) | (0.009, 8) | | |
| 14 | (0.000 7, 90) | (0.001 5, 36) | (0.004, 32) | (0.006, 8) | | | | | |
| 15 | (0.005, 78) | (0.007, 62) | (0.001 25, 44) | (0.002 5, 30) | (0.003, 22) | (0.006, 8) | | | |
| 16 | (0.000 5, 50) | (0.001, 36) | (0.002, 22) | (0.002 1, 20) | (0.003, 12) | (0.006, 8) | | | |
| 17 | (0.000 35, 76) | (0.000 7, 62) | (0.000 75, 58) | (0.001 4, 48) | (0.001 5, 8) | | | | |
| 18 | (0.00025, 64) | (0.000 35, 48) | (0.000 7, 34) | (0.001 25, 16) | (0.001 5, 8) | | | | |
| 19 | (0.000 2, 120) | (0.000 25, 36) | (0.000 5, 22) | (0.001, 8) | | | | | |
| 20 | (0.000 15, 78) | (0.000 35, 48) | (0.000 375, 44) | (0.000 375, 44) | (0.000 9, 22) | (0.001 4, 20) | (0.001 8, 8) | | |
| 21 | (0.000 15, 50) | (0.000 25, 36) | (0.000 35, 20) | (0.000 9, 12) | (0.001, 8) | | | | |
| 22 | (0.000 1, 106) | (0.000 2, 92) | (0.000 25, 8) | | | | | | |
| 23 | (0.000 75, 64) | (0.000 15, 50) | (0.000 25, 36) | (0.000 375, 16) | (0.000 45, 8) | | | | |
| 24 | (0.000 75, 36) | (0.000 15, 22) | (0.000 3, 8) | | | | | | |
| 25 | (0.000 1, 78) | (0.000 25, 36) | (0.001, 8) | | | | | | |
| 26 | (0.000 75, 36) | (0.000 25, 8) | | | | | | | |
| 27 | (0.000 75, 8) | | | | | | | | |

deduplication option $T_{M_i}^{\text{Non-Dedu}}$, required cloud storage level of a data package under working mode $M_i$ for non-deduplication option $T_{M_i}^{\text{Non-Dedu}}$, execution time of a data package under working mode $M_i$ for deduplication option $T_{M_i}^{\text{Dedu}}$, hash collision probability rate of a data package under working mode $M_i$ for deduplication option $T_{M_i}^{\text{Dedu}}$, required cloud storage level of a data package under working mode $M_i$ for deduplication option $T_{M_i}^{\text{Dedu}}$, the timing constraint $TC$.

The output is a data deduplication plan that determines the working mode for each data package.

The proposed problem is to determine the optimal solutions required to obtain the minimum cloud storage and its corresponding lowest probability.

To solve the problem, we develop an optimal algorithm named the OSP (Optimal Storage and Probability) algorithm. The algorithm is designed to minimize the required amount of cloud storage by considering the minimization of the required storage space and its corresponding hash collision probability under a certain timing constraint in a cloud-based telehealth system. The input of the OSP algorithm is a B-Table that maps all required variables in a machine-friendly manner. The OSP algorithm outputs a D-Table that creates the optimal solutions under the configured constraints. The main procedures of the OSP algorithm produce a D-Table in the sequence of the data package inputs.

The pseudo-code of the OSP algorithm is provided in Algorithm 1. The main phases of the OSP algorithm include:

1) Initialize the D-Table and input B-Table in it. We use the timing constraints to create the constraint row and add the pairs consisting of probability and storage space.

2) The first row is created by filling up all pairs attached to the first data package $D_1$ used as input. We compare all possible pairs and only retain the optimal solution pairs. Those non-optimal solutions are removed. The comparisons are based on both probabilities and storage.

3) We repeat the same operations on other inputted data packages until all data packages are added to the table. This process is accomplished by using a few For loops. The final optimal solutions are stored in the last row of the input data package.

4) We output the optimal solutions by considering the configured requirements from D-Table.

**Algorithm 1** OSP Algorithm

Require: B-Table

Ensure: D-Table

1: $D_{1,j} \leftarrow BTable_{1,j}$

2: /*Input data from B-Table to D-Table*/

3: for ∀ Working Options: $WO_i$ , $i$>1

4: for ∀ Storage: St $j$

5: for ∀ Storage: St $k$ in $BTable_{i,k}$

6: if $D_{i-1,j-k}$!= *null*

7: Calculate $D_{i,j}$ by pairing $D_{i-1,j-k}$ with $BTable_{i,k}$

8: end if

9: end for

10: Remove all non-optimal solutions from D-Table

11: /* Consider both storage and probability */

12: end for

13: end for

14: return D-Table and output the optimal solution based on the application configuration.

The time complexity of Algorithm 1 is $T(n)$= $O(mpq)$. In $O(mpq)$, $m$ refers to the number of input data packages, $p$ refers to the configured timing constraint that can be considered the number of columns in the D-Table, and $q$ refers to the average number of probability-storage pairs in each cell of the D-Table. The next section presents some of the experimental results obtained from our laboratorial evaluations.

## 5 Experiment and the results

In this section, we present selected evaluation results

from our experiments. The experimental evaluations were designed to examine the performance of SD2M in various dimensions under timing constraints, mainly covering hash collision probabilities and storage levels. We developed a simulator named SD2M-SIM. The hardware employed in our experiments was a processor (Intel(R) Xeon(R) CPU E52687W v4 @ 3.00 GHz), with 64.0 GB memory.

In addition, we designed a variety of experimental configurations in order to assess the performance of SD2M for different application scenarios. The configuration of the probability range used a read-friendly order of magnitude for the purpose of comparisons, which used random probability values in order to simulate a broad range of application situations. We also ran a greedy algorithm in our simulator for comparison with SD2M. We provide the most important experimental settings here:

**Setting 1** We configured the number of input data packages as 4. The parameters of the data packages were varied in order to simulate cloud-based telehealth application scenarios in which different types or sizes of data packages might be required.

**Setting 2** We configured the number of in-put data packages as 8. The parameters of the data packages were varied with the same consideration as Setting 1.

The reason for configuring different input data packages was twofold. First, we intended to test the performance differences when using the SD2M approach with data packages of various sizes used by telehealth systems, as well as comparing its performance with that of the greedy algorithm. Second, we aimed to test the relation between the proposed approach and the diversity of the input data packages.

Fig.3 displays 10 groups of experimental results to compare the hash collision probabilities when the same timing constraints were applied under Setting 1. The average SD2M hash collision probability was 16.98%. According to our collected data, the average probability of SD2M was 23.02% lower than that of the greedy algorithm. Fig.4 shows the comparison results of the required storage levels when using the same timing constraints under Setting 1. The average storage level of our approach was 13.4, which is 29.25% lower than the average storage level obtained by using the greedy algorithm.

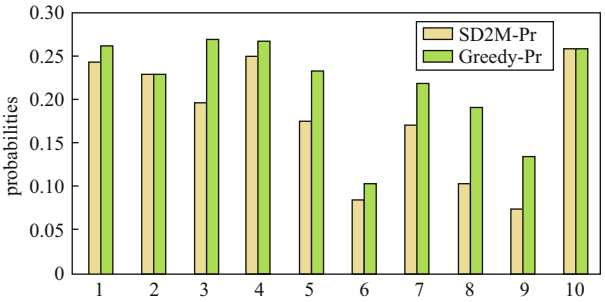


**Figure 3** Comparison of hash collision probabilities using the same timing constraints under Setting 1

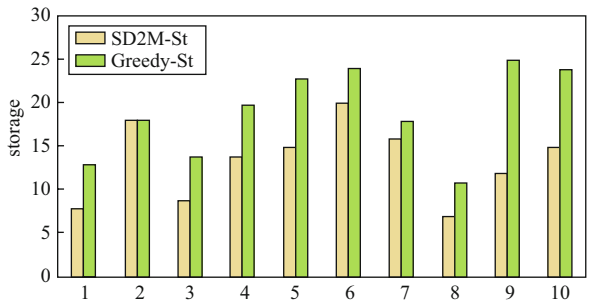


**Figure 4** Comparison of required storage levels using the same timing constraints under Setting 1

Fig.5 compares the results of the hash collision probabilities when the same storage availabilities were applied under Setting 1. The average probability gained from using SD2M was 17.42%, which is 33.88% lower than the probability gained from using the greedy algorithm. Fig.6 further compares the required execution time when the same storage availabilities were applied under Setting 1. We found that the average execution time of our approach was 11.5, which is 27.75% lower than that of the greedy algorithm.

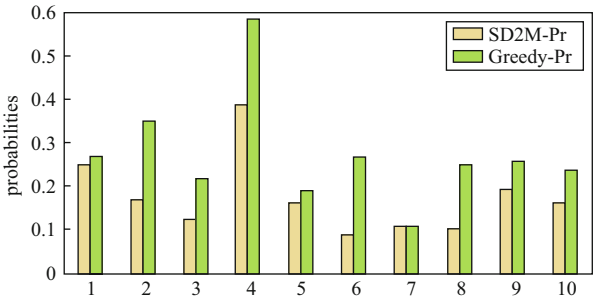


**Figure 5** Comparison of hash collision probabilities by applying the same storage availabilities under Setting 1

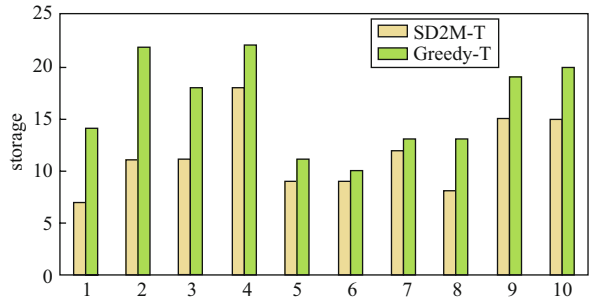


**Figure 6** Comparison of the required storage levels by applying the same storage availabilities under Setting 1

Furthermore, we present a few experimental results for Setting 2 from Fig.7 to Fig.10. Fig.7 shows the results of comparing hash collision probabilities when the same timing constraints were applied. Our observations indicated that the advantage of using our approach was that it is simplified and most probabilities gained from using SD2M were lower than the probabilities gained from the greedy algorithm. The average probability of using SD2M under this setting was 3.24%, which is 54.52% lower than that of the greedy algorithm.

Additionally, Fig.8 shows the experimental results obtained by comparing the required storage levels when the same timing constraints were applied. As shown in the figure, the performance of our approach improved compared to the results obtained with the settings used previously. The average storage level was 29.9, i.e., 29.42% lower than that achieved with the greedy algorithm.

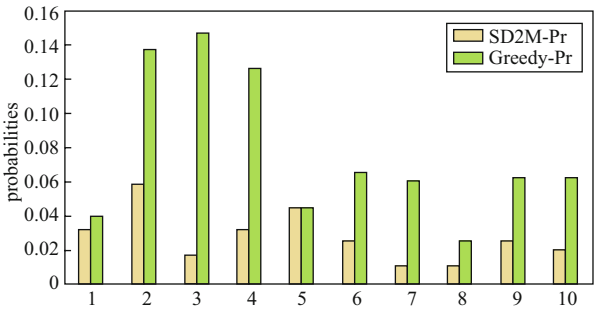


**Figure 8** Comparison of required storage levels using the same timing constraints under Setting 2

Finally, Fig.9 presents a comparison of the hash collision probabilities when the same storage availabilities are employed. We found that the average of the probability by using our approach was 2.76%, which is 56.81% lower than that obtained with the greedy algorithm. Fig.10 presents a comparison of the required execution time when the same storage availabilities are configured. The average execution

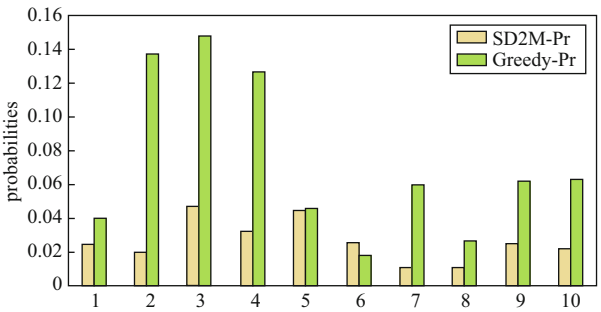


**Figure 7** Comparison of hash collision probabilities using the same timing constraints under Setting 2

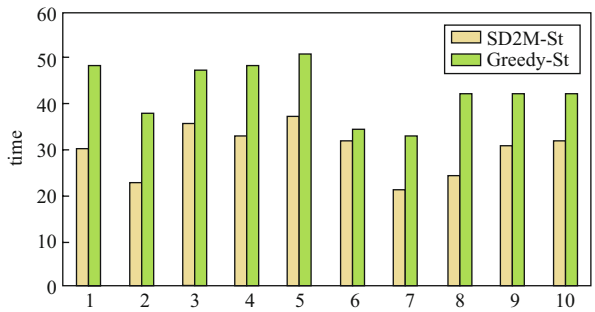


**Figure 9** Comparison of hash collision probabilities by applying the same storage availabilities under Setting 2

time when using our approach was 21.9, i.e., 24.64% lower than that of the greedy algorithm.

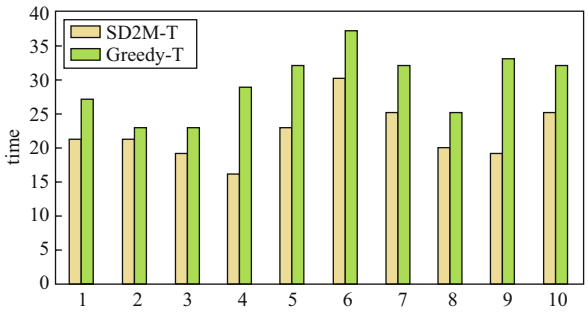


**Figure 10** Comparison of required execution time applying the same storage availabilities under Setting 2

In summary, based on our data collections and evaluations, we noticed that the hash collision probabilities could be influenced by the number of input data packages. The probabilities had a negative relationship with the number of input data packages when both the SD2M and greedy algorithms were applied. Overall, the performance of our proposed algorithm was superior to that of the greedy algorithm because of the creation of an optimal solution.

## 6 Conclusions

This paper proposed a novel approach for data deduplication designed for cloud-based telehealth systems. The proposed approach considered three important performance parameters of data deduplication, including execution time, hash collision probabilities, and data storage space. The proposed approach, named SD2M, was supported by the algorithm we developed using dynamic programming. Our approach produces optimal solutions for obtaining the lowest hash collision probability and the required cloud storage under the specified timing constraints. Our experimental evaluations proved the correctness and effectiveness of our approach.

## References

[1] GAI K K, QIU M K, JAYARAMAN S, et al. Ontology-based knowledge representation for secure self-diagnosis in patient-centered telehealth with cloud systems[C]//The 2nd IEEE International Conference on Cyber Security and Cloud Computing, New York, 2015: 98-103.

[2] GAI K K, QIU M K, CHEN L Q, et al. Electronic health record error prevention approach using ontology in big data[C]//The 17th IEEE International Conference on High Performance Computing and Communications, New York, 2015: 752-757.

[3] GAI K K, DU Z H, QIU M K, et al. Efficiency-aware workload optimizations of heterogeneous cloud computing for capacity planning in financial industry[C]//The 2nd IEEE International Conference on Cyber Security and Cloud Computing, New York, USA, 2015: 1-6.

[4] GAI K K, Li S E. Towards cloud computing: a literature review on cloud computing and its development trends[C]//The 4th International Conference on Multimedia Information Networking and Security, Nanjing, China, 2012: 142-146.

[5] ATENIESE G, FU K, GREEN M, et al. Improved proxy re-encryption schemes with applications to secure distributed storage[J]. ACM trans. on info. and syst. security, 2006, 9(1): 1-30.

[6] LI Y, XU M, NG C, et al. Efficient hybrid inline and out-of-line deduplication for backup storage[J]. ACM transactions on storage, 2015, 11(1): 2.

[7] CHU C, CHOW S, TZENG W, et al. Key-aggregate cryptosystem for scalable data sharing in cloud storage[J]. IEEE transactions on parallel and distributed systems, 2014, 25(2): 468-477.

[8] LI Y B, GAI K K, QIU L F, et al. Intelligent cryptography approach for secure distributed big data storage in cloud computing[J]. Information sciences, 2016, PP(99): 1.

[9] GAI K K, QIU M K, ZHAO H. Security-aware efficient mass distributed storage approach for cloud systems in big data[C]//The 2nd IEEE International Conference on Big Data Security on Cloud, New York, USA, 2016: 140-145.

[10] GAI K K, QIU M K, ZHAO H, et al. Anti-counterfeit schema using monte car-lo simulation for e-commerce in cloud systems[C]//The 2nd IEEE IEEE International Conference on Cyber Security and Cloud Computing, New York, USA, c2015: 74-79.

[11] LI Y B, DAI W Y, MING Z, et al. Privacy protection for preventing data over-collection in smart city[J]. IEEE transactions on computers, 2015, 65(5): 1339-1350.

[12] QIU M K, MING Z, LI J Y, et al. Phase-change memory optimization for green cloud with genetic algorithm[J]. IEEE transactions on computers, 2015, 64(12): 3528-3540.

[13] JAYARAMAN S, TAO L X, GAI K K, et al. Drug side effects data representation and full spectrum inferencing using knowledge graphs in intelligent telehealth[C]//The IEEE 3rd International Conference on Cyber Security and Cloud Computing, Beijing, China, 2016: 289-294.

[14] GAI K K, QIU M K, TAO L X, et al. Intrusion detection techniques for mobile cloud computing in heterogeneous 5G[J]. Security and

communication networks, 2016, 9(16): 3049-3058.
[15] HE C, FAN X, LI Y. Toward ubiquitous healthcare services with a novel efficient cloud platform[J]. IEEE Transactions on biomedical engineering, 2013, 60(1): 230-234.
[16] HABABEH I, KHALIL I, KHREISHAH A. Designing high performance web-based computing services to promote telemedicine database management system[J]. IEEE transactions on services computing, 2015, 8(1): 47-64.
[17] CHANG F, DEAN J, GHEMAWAT S, et al. A distributed storage system for structured data[J]. ACM transactions on computer systems, 2008, 26(2): 4.
[18] MEYER D, BOLOSKY W. A study of practical deduplication[J]. ACM transactions on storage, 2012, 7(4): 14.
[19] GAI K K, STEENKAMP A L. A feasibility study of platform-as-a-service using cloud computing for a global service organization[J]. Journal of information system applied, 2014, 7: 28-42.
[20] GAI K K, QIU L F, ZHAO H, et al. Cost-aware multimedia data allocation for heterogeneous memory using genetic algorithm in cloud computing[J]. IEEE transactions on cloud computing, 2016, PP(99): 1.
[21] LUO S, ZHANG G, WU C, et al. Boat: distributed deduplication for big data storage in the cloud[J]. IEEE transactions on cloud computing, 2015, PP(99): 1.
[22] GAI K K, QIU M K, ZHAO H, et al. Energy-aware optimal task assignment for mobile heterogeneous embedded systems in cloud computing[C]//IEEE 3rd International Conference on Cyber Security and Cloud Computing, Beijing, China, 2016: 198-203.
[23] GAI K K, QIU M K, ZHAO H, et al. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing[J]. Journal of network and computer applications, 2016, 59: 46-54.
[24] GAI K K, QIU M K, CHEN M, et al. SAEAST: security-aware efficient data transmission for ITS in mobile heterogeneous cloud computing[J]. ACM transactions on embedded computing systems, 2016, PP: 1.
[25] MAO B, JIANG H, WU S, et al. Leveraging data deduplication to improve the performance of primary storage systems in the cloud[J]. IEEE transactions on computers, 2016, 65(6): 1775-1788.
[26] XIA W, JIANG H, FENG D, et al. Similarity and locality based indexing for high performance data deduplication[J]. IEEE transactions on computers, 2015, 64(4): 1162-1176.
[27] FU Y, JIANG H, XIAO N, et al. Application-aware local-global source deduplication for cloud backup services of personal storage[J]. IEEE transactions on parallel and distributed systems, 2014, 25(5): 1155-1165.
[28] YAN Z, DING W, YU X, et al. Deduplication on encrypted big data in cloud [J]. IEEE transactions on big data, 2016, 2(2): 138-150.
[29] GAI K K, QIU M K, THURAISINGHAM B, et al. Proactive attribute-based secure data schema for mobile cloud in financial industry[C]//The IEEE International Symposium on Big Data Security on Cloud; 17th IEEE International Conference on High Performance Computing and Communications, New York, USA, 2015: 1332-1337.
[30] QIU M K, GAI K K, THURAISINGHAM B, et al. Proactive user-centric secure data scheme using attribute-based semantic access controls for mobile clouds in financial industry[J]. Future generation computer systems, 2016, PP: 1.
[31] LI J, LI Y, CHEN X, et al. A hybrid cloud approach for secure authorized deduplication[J]. IEEE transactions on parallel and distributed systems, 2015, 26(5): 1206-1216.
[32] GAI K K, QIU M K, ZHAO H, et al. Privacy-aware adaptive data encryption strategy of big data in cloud computing[C]//IEEE 3rd International Conference 12 Journal of Communications and Information Networks 2016 on Cyber Security and Cloud Computing, The 2nd IEEE International Conference of Scalable and Smart Cloud, Beijing, China, 2016: 273-278.
[33] WEN M, OTA K, LI H, et al. Secure data deduplication with reliable key management for dynamic updates in CPSS[J]. IEEE transactions on computational social systems, 2015, 2(4): 137-147.
[34] LI J, CHEN X, LI M, et al. Secure deduplication with efficient and reliable convergent key management[J]. IEEE transactions on parallel and distributed systems, 2014, 25(6): 1615-1625.

## About the authors

**GAI Keke** holds degrees from Nanjing University of Science and Technology (B.Eng.), The University of British Columbia (MET) and Lawrence Technological University (MBA and M.S.). He is currently pursuing his Ph.D. degree at the Department of Computer Science at Pace University, New York, USA. He has published more than 60 peer-reviewed papers in journals or conference proceedings, more than 20 journal papers (including ACM/IEEE Transactions), and more than 40 conference papers. He has been granted three IEEE Best Paper Awards (IEEE SSC'16, IEEE CSCloud'15, IEEE BigDataSecurity '15) and one IEEE and two Best Student Paper Awards (HPCC'16, SmartCloud '16) by IEEE conferences in recent years. His paper about cloud computing has been ranked as one of the "Most Downloaded Articles" of JNCA (Journal of Network and Computer Applications). He is involved in a number of professional/academic associations, including ACM and IEEE. Currently, he is serving as a Secretary/Treasurer of the IEEE STC (Special Technical Community) in Smart Computing of the IEEE Computer Society. His research interests include mobile cloud computing, cyber security, combinatorial optimization, business process modeling, enterprise architecture, and Internet computing. He also served as Finance Chair/Operation Chair/Publicity Chair/Web Chair in a number of academic events, such as SmartCom'16, IEEE HPCC/ICESS/CSS'15. (Email: kg71231w@pace.edu)

**QIU Meikang** [corresponding author] received the BE and ME degrees from Shanghai Jiao Tong University and received Ph.D. degree of computer science from University of Texas at Dallas. Currently, he is an adjunct professor at Columbia University and associate professor of computer science at Pace University. He is an IEEE senior member and ACM senior member. He is the chair of IEEE Smart Computing Technical Committee. His research interests include cyber security, cloud computing, big data storage, hybrid memory, heterogeneous systems, embedded systems, operating systems, optimization, intelligent systems, sensor networks, etc. A lot of novel results have been produced and most of them have already been reported to research community through high-quality journal and conference papers. He has published 5 books, 330 peer reviewed journal and conference papers (including more than 150 journal articles, more than 180 conference papers, more than 50 IEEE/ACM Transactions papers), and 3 patents. He has won ACM TODAES (Transactions on Design Automation of Electrical Systems) 2011 best paper award. His paper about cloud computing has been published in JPDC (Journal of Paralleland Distributed Computing, Elsevier) and ranked #1 in top hottest 25 papers of JPDC 2012. He has won another 8 conference best paper awards in recent years. Currently he is an associate editor of more than 10 international journals, including IEEE Transactions on Computer and IEEE Transactions on Cloud Computing. He is the General Chair/ Program Chair of a dozen of IEEE/ACM international conferences, such as IEEE HPCC, IEEE CSCloud, IEEE BigDataSecurity. He has given more than 100 talks all over the world, including Oxford, Princeton, Stanford, and New York University. He has won Navy Summer Faculty Award in 2012 and Air Force Summer Faculty Award in 2009. His research is supported by US government such as NSF, Air Force, Navy and companies such as GE, Nokia, TCL, and Cavium. (Email: mqiu@pace.edu)

**SUN Xiaotong** received the B.E. degree from Qingdao University of Science and Technology, China, in 2014. Currently, she is pursuing her Master's Degree in computer science at the Department of Computer Science at Pace University, New York, USA. Her research interests include big data, data mining, and data analysis. (Email: xs43599n@pace.edu)

**ZHAO Hui** received the B.E. and M.S. degrees from Xi'an Technology University, Shanxi and Henan University, Henan, China, in 2000 and 2008, respectively. He is a Ph.D. student at the Seidenberg School of Computer Science and Information Systems at Pace University. He is currently an associate professor in the Software School of Henan University. (Email: zhh@henu.edu.cn)