IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 25, 2019, accepted July 19, 2019, date of publication July 31, 2019, date of current version August 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932259

Active Data Replica Recovery for Quality-Assurance Big Data Analysis in IC-IoT

SONGYUN WANG[1], JIABIN YUAN[1], XIN LI[1,2], (Member, IEEE), ZHUZHONG QIAN[2], FABIO ARENA[3], (Member, IEEE), AND ILSUN YOU[4]

[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[3]Faculty of Engineering and Architecture, Kore University of Enna, 94100 Enna, Italy
[4]Department of Information Security Engineering, Soonchunhyang University, Asan 31538, South Korea

Corresponding author: Ilsun You (ilsunu@gmail.com)

This work was supported by the Soonchunhyang University Research Fund as well as the National Natural Science Foundation of China under Grant 61802182.

ABSTRACT QoS-aware big data analysis is critical in Information-Centric Internet of Things (IC-IoT) system to support various applications like smart city, smart grid, smart health, intelligent transportation systems, and so on. The employment of non-volatile memory (NVM) in cloud or edge system provides good opportunity to improve quality of data analysis tasks. However, we have to face the data recovery problem led by NVM failure due to the limited write endurance. In this paper, we investigate the data recovery problem for QoS guarantee and system robustness, followed by proposing a rarity-aware data recovery algorithm. The core idea is to establish the rarity indicator to evaluate the replica distribution and service requirement comprehensively. With this idea, we give the lost replicas with distinguishing priority and eliminate the unnecessary replicas. Then, the data replicas are recovered stage by stage to guarantee QoS and provide system robustness. From our extensive experiments and simulations, it is shown that the proposed algorithm has significant performance improvement on QoS and robustness than the traditional direct data recovery method. Besides, the algorithm gives an acceptable data recovery time.

INDEX TERMS Big data analysis, data recovery, IC-IoT, NVM, QoS improvement.

I. INTRODUCTION

Big data analysis has meaningful importance for IC-IoT system. On the one hand, it is easy to collect big volume and multisource data in IC-IoT with the pervasive utilization of smart equipments, such as smartphones, cameras, and sensors. On the other hand, it is highly valuable to explore the big data to support various applications such as information-center networking [1], [2], smart city and transportation system [3], [4], smart health [5], online social network [6]. Hence, it is essential to conduct QoS (Quality of Service) assurance big data analysis in cloud data center to support IC-IoT services continuously [7], [8]. Generally, the service response time is critical for supporting better prediction and timely decision-making. Due to the limited I/O performance, HDD (Hard Disk Drive) is not an idea device for fast data analysis. Hence, there are two essential factors for IoT service provision, data, and bandwidth for data communication. They are also a challenging problem for data center resource allocation. Recent literature works have proposed lots of resource management approaches for data centers to achieve efficient and fast data analysis [9]–[12]. However, DRAM is becoming part of the bottleneck for fast data analysis due to the limited capacity, and it is approaching scalability limits [13]. It is expected that NVM (Non-Volatile Memory) will be equipped in future data centers to provide in-memory data processing [14], [15], since NVM can achieve storage-class memory and it owns non-volatile feature. Hence, NVM provides an opportunity to conduct faster data analysis for IoT data, and provide real-time services. Nevertheless, NVM also has a noticeable disadvantage caused by its limited write endurance. As a consequence, there may be a NVM failure suddenly, which leads to the risk of losing data during data analysis in the data center.

The associate editor coordinating the review of this manuscript and approving it for publication was Macarena Espinilla.

To provide high reliability and performance improvement, it is usual to place multiple replicas or copies of data in a data center. Once some NVM failure occurs, it is necessary to recover the lost data replicas to maintain the system stability and performance. Traditionally, related works about data recovery pay more attention to the storage system with erasure code and take recovery time as the primary concern [16]. For the data center with NVM, it will be a hybrid storage system [17] because of the different features of NVM, DRAM, and disk. Therefore, the disks will constitute a typical persistent storage system. At the same time, NVM is mainly adopted for computing, e.g., data analysis in this work.

In this paper, we focus on the data recovery problem to cope with NVM failure during IoT real-time big data analysis. We take the QoS (Quality of Service), recovery time and system robustness into account, rather than only the recovery time. In fact, the primary motivation is that our problem is different from the traditional data recovery mode for a storage system. On the contrary, our work mainly focuses on a computing system. Hence, we use data migration as the principal method to recover lost data replicas. Moreover, we take the allocated bandwidth as the primary indicator for QoS. For the robustness issue, it represents the ability of the system to sustain another NVM failure or rack malfunction during data recovery stage, since there should be at least one replica for each data to satisfy data analysis service. Specifically, it is better to maintain replicas at two different racks in a data center. We will use the percentage of data that can survive from another NVM failure or rack malfunction during the recovery stage to measure the robustness of the recovery approach. Recovery time is intuitively measured by the time to recover the data replicas.

We aim to provide guaranteed services even when NVM failure occurs, which is different from previous works based on wear-leveling NVM endurance enhancement, like the one shown in [18]. To conduct data replica recovery, we can start the recovery process immediately when a failure occurs to achieve shorter recovery time. However, this method will decrease the QoS and robustness significantly. Based on this conclusion, we first analyze the data replica distribution, and further define a replica high reliability rule. The basic idea is that at least one replica should survive for each data in another NVM failure, even rack failure. Then, we propose a staged data recovery method, which considers the replica distribution and can achieve improved QoS performance. Furthermore, we take the data hotness into account, which reflects the bandwidth requirement for the services. We propose a data rarity model to measure the necessity and urgency for recovering replica. Finally, we combine replica distribution and data hotness to give a more reasonable assessment for the data replica and propose a rarity-aware data recovery algorithm.

The contributions can be summarized as follows.

- We present a data recovery problem in a data center with NVM. We aim to guarantee the QoS during data recovery, and we consider the QoS, recovery time and system robustness together to achieve a more comprehensive assessment.
- We propose an algorithm based on data rarity model, which takes data replica distribution and service bandwidth requirement into account. To the best of our knowledge, we are the first to combine the two factors to improve data recovery performance.
- We conduct extensive experiments and simulations, and the results show that our algorithm has a significant improvement on the QoS and robustness with acceptable data recovery time.

The rest of this paper is organized as follows. We review the related works in Section II. We present the scenario and preliminaries in Section III. Then, the staged data recovery and rarity-aware algorithms are proposed in Sections IV and V respectively. These proposed algorithms are evaluated in Section VI. Finally, we conclude our paper in Section VII.

II. RELATED WORKS

In this section, we provide related literature concerning disk failure and NVM feature on write endurance.

A. DISK FAILURE

With the development of Internet cloud technology, more and more data are generated and stored in disks. As a consequence, disk failures lead to data loss. Therefore, it is necessary to investigate the data replica management [19] and disk failure control [20].

Proper data replica management helps with disk failure data recovery. In [21], Myint and Naing deal with the management of replication scheme and their proposed solution can balance the load of a PC cluster. In detail, they focus on the number of replicas and number of data blocks. By adjusting the size of these parameters, the probability of disk failure could be reduced. To decrease access latency, in [22], Kim *et al.* propose a real-time data replica strategy. The priority value (PV) is used to update a file's priority after it is accessed. More in detail, PV represents the importance of the file. The strategy proposed by the authors updates the data replica by PV. In this way, the management of data replica improves the performance of service requests. Another way to reduce access latency is to consider the link bandwidth. In a transfer of many data between physical machines, the link bandwidth will limit performance. So it is crucial for users to provide better service based on the currently limited link bandwidth. Considering the usage of link bandwidth, the distributed storage system proposed in [23] has a good performance. As a result, a full consideration of link bandwidth can effectively shorten access time and load balance. Regarding the link, a formal description of the data link is provided in [24].

Data replica may be lost at any time. Therefore, disk failures require more attention [25]. Besides, there are many algorithms to solve a disk failure. For instance, erasure code protects data security [26]. Most of the algorithms introduced

in the literature are designed for a unique code. In [27], Luo and Shu present two disk recovery strategies composed of two algorithms that can work with any erasure code. These algorithms are designed for load-balanced recovery. The only difference between them is if the condition of minimal read for the recovery must be considered or not. Anyhow, the recovery strategy proposed by the authors obtains better performance than the approach introduced in [28]. Some literature works focus on the data loss probability. Another proposed solution, different from what is proposed in [27], studies the disk failure in the stack-level and not in stripe-level. The authors introduce a BP-scheme to solve the problem. In this case, the obtained performance is better than Khan's scheme [29]. Finally, it is clear that data recovery time must be reduced. A possible method to achieve that is to combine data requested in the same rack and to use the useful link, as shown in [30]. The approach suggested by the authors delivers a right load balance. However, their solution does not pay much attention to the QoS, which is an essential indicator for IoT data analysis.

B. NVM ENDURANCE MANAGEMENT AND ENHANCEMENT

NVM (Non-Volatile Memory) is becoming the likely choice for its advantages, including non-volatile, byte-by-byte access, high storage density, low power consumption, and good read/write performance. PCM (Phase-Change Memory) is the current choice for NVM. Nevertheless, the limited write endurance still is the apparent shortage and a challenging problem to use PCM.

Wear-Leveling (WL) is the core method to handle the endurance management and to improve the PCM lifetime in recent literature works [31]. In [32], the authors suggest a start-gap wear leveling technique to improve the PCM lifetime. The goal of the authors is to adopt an algebraic mapping between logical and physical addresses, thus bypassing also tracking per-line write counts. Another adaptive wear-leveling algorithm is introduced in [33]. In this case, the approach suggested by the authors is composed of three unique patterns for PRAM main memory with a DRAM buffer. The main benefit is that the buffering solution decreases the write counts and prevents skewed writing by taking into account both the write count and clean data based on a least recently used (LRU) scheme.

In [34], Jiang *et al.* suggest an LLS (Line-Level mapping and Salvaging) scheme in which a portion of the total space is allocated dynamically as backup space in a PCM device while mapping failed lines are employed to backup PCM. As known, the LLS builds a contiguous PCM space and hides lower level failures from the Operating System and applications. Chen *et al.* introduce an efficient age-based wear-leveling scheme in [35]. In this case, the proposed solution is agreeable with existing virtual memory management. Two implementations, i.e., bucket-based and array-based wear leveling, with nearly zero search cost, are proposed to obtain a good tradeoff between time and space complexity. A FGWL (Fine Grained Wear-Leveling) is presented in [36] and is used for producing uniform writes (in the average case) while avoiding per line storage.

For the failure repair issue, Schechter *et al.* propose Error-Correcting Pointers (ECP) in [37]. This approach aims to optimized error correction for memories. In fact, it is clear that the permanent cell failures lead to error but, employing the proposed method, they could be immediately detected. In [38], Ipek *et al.* introduce a dynamically replicated memory (DRM). The solution presented by the authors represents the first hardware and operating system interface, designed for PCM, which enables continued operation through graceful degradation when a hardware failure occurs. On the contrary, in this paper, we pay our attention to the data recovery when a failure occurs.

III. SCENARIO AND PRELIMINARIES

For a given data center with N PMs equipped with NVM to conduct big data analysis tasks for supporting various IoT applications/services, the tasks rely on some data and network bandwidth to provide continuous services. It is necessary to be aware that multiple services may share the same data, such as, vehicles navigation and stations setting are needed by users to obtain the road network information.

For each service, we assume that it is associated with one data, and there is a bandwidth demands for the service to accomplish its task with guaranteed QoS. Hence, we can use a two-tuple to represent the service as $S_i : \langle D_j, B_i \rangle$, where D_j means the j^{th} data, and B_i is the required bandwidth for S_i. We also use $\delta(S_i)$ to represent the associated data for S_i, i.e. $\delta(S_i) = D_j$ for the above example.

Each PM (Physical Machine) or server in the data center is equipped with an NVM (Non-Volatile Memory), which is split into m storage slots to store data. For each data, there are multiple replicas in the system to support redundancy and performance improvement. Actually, the service is associated with some given data replica. Therefore, we can update the service as $S_i : \langle R_{jk}, M_k, B_i \rangle$, where R_{jk} is the replica that hosted on PM M_k for data D_j. Similarity, we define $R(S_i) = R_{jk}$ to represent the replica that actually access by service S_i. To clarify the relationship between service and data replica, we define $R_{jk} = \emptyset$ if there is no replica of D_j placed on PM M_k, and define an indicator as follows.

$$\pi(S_i, R_{jk}) = \begin{cases} 1, & S_i \text{ is associated with replica } R_{jk}; \\ 0, & \text{otherwise.} \end{cases}$$

Based on the bandwidth requirement of the service, we can further define the *load* for the replica. We use ω_{jk} to represent the load for replica R_{jk}, and define it as:

$$\omega_{jk} = \sum_{i=1}^{|\mathcal{S}|} \pi(S_i, R_{jk}) \cdot B_i \tag{1}$$

where $\mathcal{S}$ is the set of services in the system and $|\mathcal{S}|$ is the number of services.

The bandwidth requirement of the services usually is guaranteed to ensure the QoS. Once NVM failure occurs, the hosted data replicas will be lost. For the associated services, we assume they will be redirected to other replicas of the same data equally. It is necessary to recover the lost data replicas for system robustness and performance improvement. As mentioned above, we consider the QoS, recovery time and robustness together during data recovery. For each service, we use the allocated bandwidth to measure the QoS, i.e.:

$$Q_i = A_i/B_i$$

where A_i is the allocated bandwidth to service S_i. In fact, the QoS should be a notation over time, here is just the value under a specific time. We will give more details on the QoS of the global system in Section VI.

It is straightforward to understand the recovery time, which concerns the time to recover all lost data. The recovery time is also a window of vulnerability since the system is subject to another NVM failure with the risk of losing data. On the one hand, we should reduce the window of vulnerability, and it should be shorter than the time between two regular NVM failures, which is determined by the NVM feature (limited write endurance). On the other hand, we need to maintain the system robustness during data recovery. Regarding the robustness issue, it aims to provide the ability for each data to survive from another NVM failure or rack failure during data recovery. Therefore, we need to define a high reliability rule to represent the robustness. As a consequence, we give a rule as follows:

HR-Rule (*High Reliability Rule*): There should be at least two racks to store the replicas for each data.

The HR-rule indicates that there are at least two replicas for each data, and the data can survive from one single NVM failure or rack failure. A direct consequence of this feature is that there is at least one replica after one NVM failure or rack failure. Alternatively, we declare that the replica distribution is robust if the data can survive from one single NVM failure or rack failure. Indeed, the replica distribution obeys HR-rule is robust. To simplify the description, we define another indicator as follows:

$$\mathbf{\Gamma}(D_j) = \begin{cases} 1, & \text{replica distribution of } D_j \text{ obeys HR-rule;} \\ 0, & \text{otherwise.} \end{cases}$$

We will use the percentage of data whose replica distribution obey the HR-rule, i.e.:

$$\alpha = \frac{\sum_{j=1}^{|\mathcal{D}|} \Gamma(D_j)}{|\mathcal{D}|} \tag{2}$$

where $\mathcal{D}$ is the set of data in the system, and $|\mathcal{D}|$ is the number of data.

Furthermore, it is essential to be aware that HR-rule must be flexible since could be adjusted according to the real system requirement.

We investigate the data recovery problem when NVM failure occurs. We take the QoS, recovery time and robustness into account, and aim to propose a useful data replica recovery algorithm to achieve improved QoS and robustness for the system within acceptable recovery time.

IV. STAGED DATA REPLICA RECOVERY

It is straightforward to assume that the direct data replica recovery method can achieve shorter recovery time, which can decrease the window of vulnerability for the system. However, the QoS will be degraded significantly. This issue occurs because data transfer will occupy bandwidth, and the allocated bandwidth to services will be decreased. We analyze the problem and find that the replica distribution for each data is different. We can treat the lost replica respectively. Hence, we propose a staged data replica recovery algorithm. The basic idea is to recover the data with only one replica in the system firstly. Then, we recover the data with two replicas, and so on. The benefit of this algorithm is to avoid a burst of data communication. The details of the Staged Data replica Recovery (SDR) algorithm is shown in Alg. 1.

Algorithm 1 Staged Data Replica Recovery: $sdr(\Omega)$

Require: Ω: the set of data having lost replica.

```
max ← 0;
for all D_j ∈ Ω do
    if max < N_r(D_j) then
        max ← N_r(D_j);
    end if
end for
for i ← 1 → max do
    Ω_i ← ∅;
end for
for all D_j ∈ Ω do
    n ← N_r(D_j)
    Ω_n ← Ω_n ∪ {D_j};
end for
for i ← 1 → max do
    recovery(Ω_i);
end for
```

To describe the replica distribution and algorithm clearly, we define the following indicator:

$$\boldsymbol{\pi}(D_j, M_k) = \begin{cases} 1, & \text{if } R_{jk} \neq \emptyset; \\ 0, & \text{otherwise.} \end{cases}$$

Based on this indicator, the number of replicas for data D_j can be given by the following equation:

$$N_r(D_j) = \sum_{k=1}^{N} \pi(D_j, M_k) \tag{3}$$

Besides, we define another indicator $isRack(M_m, M_n)$ to represent if two PMs belong to the same rack or not. $isRack(M_m, M_n)$ equals to 1 if the two PMs belong to the same rack; the value equals to 0, if not.

In the algorithm, the set of data having lost replica in the NVM failure is classified into multiple categories by their number of available replicas. *max* in lines 1-6 represents the number of categories. Then, the data with the same replica numbers are assigned to the same subset. Ω_n is the subset containing the data with n replicas. Finally, we call the *recovery* function to finish the data recovery procedure stage by stage. The details of the *recovery* is shown in Alg. 2.

Algorithm 2 Data Recovery Procedure: *recovery*(Ω)

Require: Ω: the set of data needs new replica.

1: **for** $D_j \in \Omega$ **do**
2: $src \leftarrow 0, dst \leftarrow 0, \omega \leftarrow 0, flag \leftarrow 0$;
3: **for** $k \leftarrow 1 \rightarrow N$ **do**
4: **if** $(R_{jk} \neq \emptyset)\&\&(\omega(M_k) > \omega)$ **then**
5: $src \leftarrow k$;
6: $\omega \leftarrow \omega(M_k)$;
7: **end if**
8: **end for**
9: *sort*(*racks*);
10: **for all** $rack \in racks$ **do**
11: **if** $isHR(D_j, rack)$ **then**
12: *sort*(*PMs*)
13: **for all** $M_k \in PMs$ **do**
14: **if** $place(D_j, M_k)$ **then**
15: $flag \leftarrow 1$;
16: $dst \leftarrow k$;
17: break;
18: **end if**
19: **end for**
20: **end if**
21: **if** *flag* **then**
22: break;
23: **end if**
24: **end for**
25: $src(D_j) \leftarrow src, dst(D_j) \leftarrow dst$;
26: **end for**
27: conduct data recovery;

For the data recovery procedure, we need to answer the following three questions. (1) When is it necessary to start the recovery procedure? (2) Where is the original data source if there are more than one available replica? (3) Where is the destination or host PM for the new replica?

For the first question, we start the recovery stage by stage as shown in Alg. 1. For the source selection problem for data migration from multiple replicas, the basic idea is to select the replica contained in the PM with minimal workload. It is shown in lines 3-8 in Alg. 2. The load of PM M_k is represented by $\omega(M_k)$, which is equal to the total load of replicas in PM M_k, i.e.:

$$\omega(M_k) = \sum_{j=1}^{|\mathcal{D}|} \pi(D_j, M_k) \cdot \omega_{jk}$$

src (in Alg. 2) is used to record the PM ID containing the source replica.

To decide the destination for the new replica, we need three steps: rack-level selection, PM-level selection, and slot-level selection. First, we sort the racks by their loads, which mean the total loads of the PMs hosted in the rack. This feature is implemented in function *sort*(*racks*) in line 9. The load of the rack is equal to the sum of PM loads in the rack. We will test each rack in *racks*. One of the necessary conditions for the selected rack is to satisfy the HR-rule, i.e. if a new replica is hosted in the selected rack. This peculiarity is shown in line 11. The function $isHR(D_j, rack)$ returns true if the new replica distribution of D_j fulfill the HR-rule when one new replica of D_j is placed in *rack*. Now, we have selected some *rack*. Then, for the PMs in the selected rack, we sort PMs by increasing the load order by the function *sort*(*PMs*). For all PM in the rack, we will also test if there is an available slot to place the new replica. This is executed by the function $place(D_j, M_k)$ in line 14. If the place function returns true, *dst* will record the hosted PM ID for a new replica. Until now, we know the source and destination for each data replica recovery.

There is another issue that requires more investigation. When we place the new replica in the selected PM, there may be no idle slot to store it. In this case, we need to replace some existing replica in the destination PM. This action is also the slot-level selection as mentioned above. To select the target replica, we still need to practice the HR-rule. A consequence of this indication is that although the selected replica is removed its data replica distribution still meet HR-rule. In our implementation, let M_k be the selected PM. We select the data replica R_{jk} with largest $N_r(D_j)$ value.

V. RARITY-AWARE DATA REPLICA RECOVERY

We reconsider the problem addressed in this paper and find that the data replica distribution should match the service requirement. This indication is also the reason to place multiple data copies in a social network system. To describe the relationship between data and service requirement, we define *hotness* for each data as follows:

$$\theta_j = \sum_{k=1}^{N} \omega_{jk}$$

where N is the number of PMs in data center, and ω_{jk} is defined in Eq. 1.

For each data, hotness indicates the demands on bandwidth for all associated services. However, it still cannot reflect the requirement for the scale of data replicas. Consequently, we define *rarity* for the data D_j, i.e., the average load for replicas to satisfy the service requirement, as follows:

$$\gamma_j = f(N_r(D_j), \theta_j) = N_r(D_j) + \beta \cdot \lg \theta_j$$

where N_r is defined in Eq. 3, β is a coefficient to adjust the θ_j with various definition. According to the experiments,

we find that the data needs more replica if $\gamma_j < 0$. This conclusion is because the data is very hot. The average load for each replica is heavy, even if there are more than three replicas in the system. We should be aware that *rarity* and *hotness* will work with the above mentioned HR-rule. So far, we have combined the data hotness and replica distribution.

Based on the above analysis, we propose a Rarity-Aware Data replica Recovery (RADR) algorithm as shown in Alg. 3. In this algorithm, the function $isHR(D_j)$ in line 3 returns true if the replica distribution of data D_j practices the HR-rule, or return false if not. Based on the two conditions, the item in set Ω is assigned to subset Ω_A and Ω_B. There may be some items in Ω that do not belong to Ω_A or Ω_B. This feature is because we take data rarity into account. We think that if there are enough replicas of the data, it is unnecessary to recover the lost replica. Formally, the RADR algorithm is also a staged algorithm, but with different data set from the SDR algorithm.

Algorithm 3 Rarity-Aware Data Replica Recovery: $radr(\Omega)$

Require: Ω: the set of data having lost replica.

1: $\Omega_A \leftarrow \Omega_B \leftarrow \emptyset$;
2: **for all** $D_j \in \Omega$ **do**
3: **if** $isHR(D_j)$ **then**
4: $\Omega_A \leftarrow \Omega_A \cup \{D_j\}$;
5: **else if** $\gamma_j < 0$ **then**
6: $\Omega_B \leftarrow \Omega_B \cup \{D_j\}$;
7: **end if**
8: **end for**
9: $recovery(\Omega_A)$;
10: $recovery(\Omega_B)$;

For the *recovery* function in Alg. 3 (RADR Algorithm), it is also a bit different from the function in Alg. 1 (SDR Algorithm). More in detail, the sub-function $place(D_j, M_k)$ is different. For the RADR algorithm, the replaced replica is the one with minimal load (refer Eq. 1), and the remaining replica distribution still satisfy the HR-rule and positive rarity value. For instance, let R_{jk} be the selected replica to be replaced. The new replica distribution for data D_j still meet the HR-rule, and we have $\gamma_j' = f(N_{r_j} - 1, \theta_j) = N_r(D_j) - 1 + \beta \cdot \lg \theta_j > 0$.

VI. PERFORMANCE EVALUATION

In this section, we will evaluate our algorithms by simulations. As discussed above, the QoS is the crucial factor to measure the algorithm performance, and the allocated bandwidth decides the QoS directly. Hence, we will evaluate the bandwidth allocation model first.

A. BANDWIDTH ALLOCATION MODEL

We conducted extensive experiments on a real system to explore the bandwidth allocation model for data migration tasks. We allowed the data transfers between PMs in the same rack. First, we transferred 10 equal-size data. The size of the data has been set to 64MB, 128MB, 256MB, 512MB, and 1024MB respectively. The data migration time for each data is shown in Table 1. From the result, we know that the 10 data have the similar migration time under any data size setting. This result indicates that the 10 data migration tasks share the bandwidth equitably.

TABLE 1. Data migration time - 10 data with same size (second).

Data Size	64MB	128MB	256MB	512MB	1024MB
Data 1	6.09	11.76	23.19	46.03	91.68
Data 2	6.09	11.80	23.20	46.04	91.68
Data 3	6.09	11.80	23.20	46.05	91.70
Data 4	6.10	11.80	23.23	46.07	91.70
Data 5	6.10	11.81	23.23	46.07	91.70
Data 6	6.10	11.81	23.23	46.07	91.73
Data 7	6.10	11.81	23.23	46.07	91.73
Data 8	6.11	11.81	23.23	46.07	91.73
Data 9	6.11	11.81	23.23	46.07	91.73
Data 10	6.11	11.81	23.23	46.07	91.73

TABLE 2. Data migration time - 5 data with various size (second).

Data Size (MB)	64	128	256	512	1024
Migration Time	3.00	5.29	8.57	13.41	18.09

We also managed the experiments in another scenario. We transferred 5 data with different size simultaneously, including 64MB, 128MB, 256MB, 512MB, and 1024MB. The data migration time is shown in Table 2. We can also conclude that the data migration tasks share the bandwidth reasonably. In detail, five tasks share the bandwidth at first, and then four tasks after the 64MB data are moved. Similarly, the data with a more substantial size have greater average bandwidth since fewer tasks share the resource gradually.

Based on the above result analysis, we have employed the average distribution model for bandwidth allocation in the following simulations.

B. SIMULATION ANALYSIS

We performed simulations to evaluate our algorithms under various settings. As the case in our data center, the link capacity has been set to 1000 Mbps. In the data center, the racks are connected by one core switch. There is one key factor to represent the initial system state, system load, which indicates the average load for all the racks. We used u to represent the system load. The value is the ratio of the average load to the link capacity.

We implemented three algorithms in our simulation. Besides, the SDR and RADR algorithms, we also realized the typical direct data replica recovery (DDR) algorithm. The latter recovers all the lost replicas immediately when an NVM failure occurs. This feature has been the baseline for our algorithm performance. For the *recovery* function, the bandwidth is assigned to the IoT services and data migration tasks equitably. We also have made the following settings.

- the system load varied from 40% to 80%;
- the data size has been set to $1GB$;
- the NVM capacity has been set to $1TB$.

Therefore, there are 1000 slots to store the data. There are 2000 data in the system. For the replica distribution, the number of replicas for data follows normal distribution between 2 and 5. Consequently, there are some idle slots in the NVM. For each service, the load accepts normal distribution

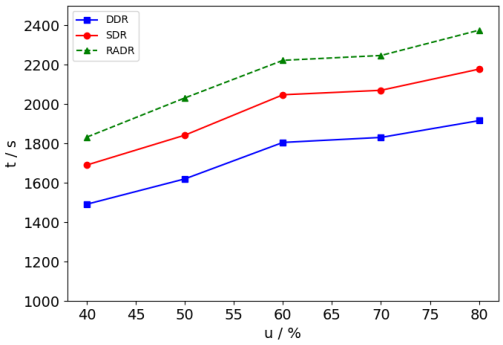


FIGURE 1. Data Recover Time for DDR, SDR, and RADR (second).

between 50 and 200. At the same time, the original replica distribution practices the HR-rule for all data. We used the average distribution model for bandwidth allocation as mentioned above.

First, for the traditional data recovery time, the obtained result is shown in Fig. 1. The x-coordinate is the system load, and the y-coordinate is the data recovery time. From this result, we know that the DDR algorithm has shortest recovery time. This conclusion is because of the DDR algorithm initials all replica recovery immediately when an NVM failure occurs, and it occupies more bandwidth to transfer the data. SDR and RARD algorithms have a more considerable recovery time since the recovered data is split into multiple groups (or stages) based on the replica distribution. According to the bandwidth allocation model, the data migration tasks have assigned less bandwidth to transfer data globally. As the system load increases, the total recovery time becomes more substantial, since there are more IoT services or data migration tasks to share the given bandwidth.

For the critical factor, i.e., the QoS, Fig. 2 shows the QoS given by the three algorithms. The x-coordinate is also the system load, and the y-coordinate represents the average QoS for services in the system. It is quantitatively calculated by Eq. 4:

$$QoS = \frac{\sum_{i=0}^{|\mathcal{S}|} Q_i}{|\mathcal{S}|} = \frac{\sum_{i=0}^{|\mathcal{S}|} \frac{\int_0^T A_i(t)dt}{B_i \cdot T}}{|\mathcal{S}|} \tag{4}$$

where T is the value of data recovery time, as shown in Fig. 1.

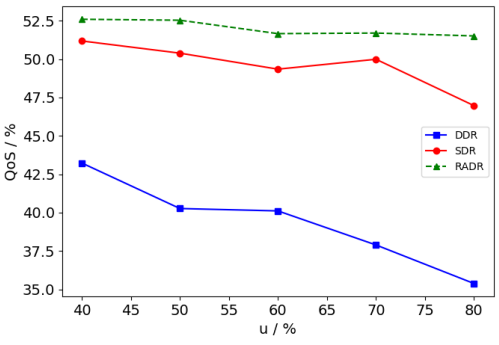


FIGURE 2. Average QoS given by DDR, SDR, and RADR.

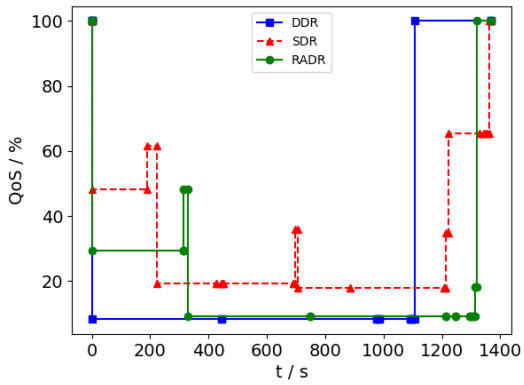


FIGURE 3. QoS values over time ($u = 30\%$).

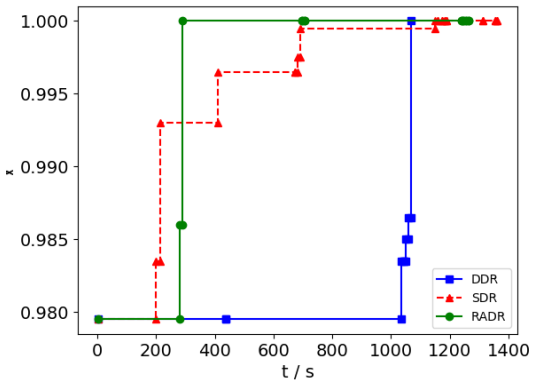


FIGURE 4. System Robustness under DDR, SDR, and RADR ($u = 30\%$).

From the obtained result, we know that our SDR algorithm and the RARD algorithm have significant QoS improvement than the DDR one. It is the similar reason for data recovery time. The IoT services occupied less bandwidth during data migration in the DDR algorithm. Subsequently, the QoS decreased significantly. We notice that the RARD algorithm offers the best QoS because it ignores many unnecessary replica recoveries based on data rarity. The principal difference between SDR and RADR lie in taking the data rarity into account or not. Hence, it is valuable to note that the combined consideration of the replica distribution and data hotness is beneficial.

Regarding the algorithms with multiple stages, they introduced smooth QoS variation. Fig 3 can demonstrate this conclusion. In fact, it provides the QoS values over time for some specific service. In this figure, the QoS related to the DDR algorithm decreases suddenly, since all data recovery tasks start simultaneously. The value goes up and down, this indicates a new data migration stage. Compared to DDR, SDR and RADR algorithms have more data migration stages, and this massive transfer is distributed to multiple stages. These stages are unquestionably shown in the figure. For the SDR algorithm, it has four stages in this instance, while the RADR algorithm has two stages.

System robustness is also significant for data recovery. Fig. 4 presents the retrieved result regarding the robustness. The x-coordinate is the time, and y-coordinate is the value

of α for the system (see Eq. 2). By analyzing the results, we know that the DDR algorithm needs to face a long-time risk of losing data by another NVM failure or rack failure. Compared to the DDR algorithm, the SDR and RARD algorithms have a meaningful improvement on the system robustness. We can also find more stages in this figure. This result is because the allocated bandwidth for each data migration task is different. The value of α increases once some data migration tasks are finished.

VII. CONCLUSION

In this paper, we examined the data recovery problem for QoS assurance IC-IoT big data analysis. We have aimed to provide the QoS guarantee and robustness for the system during the data recovery. In particular, in the proposed solution, we jointly consider the QoS, recovery time and system robustness while presenting the HR-rule and rarity model to measure the replica distribution and service requirements comprehensively. Then, the rarity-aware data replica recovery algorithm has been presented, which not only gives the priority for lost data based on the HR-rule, but also explores the unnecessary replica by data rarity. The simulation results have shown that our RADR algorithm has significant performance improvement than a traditional direct data recovery method.

REFERENCES

[1] C. Zhao, M. Dong, K. Ota, J. Li, and J. Wu, "Edge-MapReduce-based intelligent information-centric IoV: Cognitive route planning," *IEEE Access*, vol. 7, pp. 50549–50560, 2019.

[2] H. Li, K. Ota, and M. Dong, "ECCN: Orchestration of edge-centric computing and content-centric networking in the 5G radio access network," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 88–93, Jun. 2018.

[3] H. Zhou, C. M. V. Leung, C. Zhu, S. Xu, and J. Fan, "Predicting temporal social contact patterns for data forwarding in opportunistic mobile networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10372–10383, Nov. 2017.

[4] C.-M. Huang, Y.-F. Chen, S. Xu, and H. Zhou, "The vehicular social network (VSN)-based sharing of downloaded GEO data using the credit-based clustering scheme," *IEEE Access*, vol. 6, pp. 58254–58271, 2018.

[5] Z. Cai, H. Yan, P. Li, Z.-A. Huang, and C. Gao, "Towards secure and flexible EHR sharing in mobile health cloud under static assumptions," *Cluster Comput.*, vol. 20, no. 3, pp. 2415–2422, Sep. 2017.

[6] W. Jiang, G. Wang, M. Z. A. Bhuiyan, and J. Wu, "Understanding graph-based trust evaluation in online social networks: Methodologies and challenges," *ACM Comput. Surv.*, vol. 49, no. 1, p. 10, 2016.

[7] W. Lin, Z. Wu, L. Lin, A. Wen, and J. Li, "An ensemble random forest algorithm for insurance big data analysis," *IEEE Access*, vol. 5, pp. 16568–16575, 2017.

[8] Z. Ding, K. Ota, Y. Liu, N. Zhang, M. Zhao, and H. Song, "Orchestrating data as a services-based computing and communication model for information-centric Internet of Things," *IEEE Access*, vol. 6, pp. 38900–38920, 2018.

[9] H. Zhou, H. Wang, X. Chen, X. Li, and S. Xu, "Data offloading techniques through vehicular ad hoc networks: A survey," *IEEE Access*, vol. 6, pp. 65250–65259, 2018.

[10] J. Wu, M. Dong, K. Ota, J. Li, and Z. Guan, "Big data analysis-based secure cluster management for optimized control plane in software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 27–38, Mar. 2018.

[11] X. Li, L. Wang, Z. Lian, and X. Qin, "Migration-based online CPSCN big data analysis in data centers," *IEEE Access*, vol. 6, pp. 19270–19277, 2018.

[12] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Comput. Secur.*, vol. 72, pp. 1–12, Jan. 2018,

[13] T. Hirofuchi and R. Takano, "RAMinate: Hypervisor-based virtualization for hybrid main memory systems," in *Proc. ACM Symp. Cloud Comput. (SoCC)*, Oct. 2016, pp. 112–125.

[14] L. Wang, C.-H. Yang, and J. Wen, "Physical principles and current status of emerging non-volatile solid state memories," *Electron. Mater. Lett.*, vol. 11, no. 4, pp. 505–543, Jul. 2015.

[15] J. Xu, K. Ota, and M. Dong, "Real-time awareness scheduling for multi-media big data oriented in-memory computing," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3464–3473, Oct. 2018.

[16] Z. Shen, J. Shu, P. P. C. Lee, and Y. Fu, "Seek-efficient I/O optimization in single failure recovery for XOR-coded storage systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 877–890, Mar. 2017.

[17] H. Liu, Y. Chen, X. Liao, H. Jin, B. He, L. Zheng, and R. Guo, "Hardware/software cooperative caching for hybrid DRAM/NVM memory architectures," in *Proc. Int. Conf. Supercomput.*, Jun. 2017, Art. no. 26.

[18] J. Xu, D. Feng, Y. Hua, W. Tong, J. Liu, and C. Li, "Extending the lifetime of NVMs with compression," in *Proc. IEEE Design, Autom. Test Europe Conf. Exhib.*, Mar. 2018, pp. 1604–1609.

[19] B. Tian, W. Yan, L. Yan, C. Wang, F. Liu, Y. Tang, and S. Han, "A flexible dynamic migration strategy for cloud data replica," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jun. 2017, pp. 567–572.

[20] Z. Shen, J. Shu, and P. P. C. Lee, "Reconsidering single failure recovery in clustered file systems," in *Proc. Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun./Jul. 2016, pp. 323–334.

[21] J. Myint and T. T. Naing, "Management of data replication for PC cluster-based cloud storage system," *Int. J. Cloud Comput., Services Archit. (IJCCSA)*, vol. 1, no. 3, pp. 31–41, Dec. 2011.

[22] J. Kim, Y. Kim, and C. Jeon, "Real-time data replication strategy for data grids," *Cluster Comput.*, vol. 20, no. 3, pp. 2551–2562, Sep. 2017.

[23] Z.-W. He, J. Song, H. Chen, S.-H. Gao, X.-X. Chen, and Y. Zhang, "Framework of distributed storage system and dynamic load balance technology based on bandwidth condition," *Trans. Beijing Inst. Technol.*, vol. 37, no. 9, pp. 964–969, Sep. 2017.

[24] Z. Lian, X. Li, and X. Qin, "Topology-aware VM placement for network optimization in cloud data centers," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. (ISPA)*, Dec. 2017, pp. 558–565.

[25] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," *ACM SIGOPS Operating Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[26] H.-Y. Lin and W.-G. Tzeng, "A secure erasure code-based cloud storage system with secure data forwarding," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 6, pp. 995–1003, Jun. 2012.

[27] X. Luo and J. Shu, "Load-balanced recovery schemes for single-disk failure in storage systems with any erasure code," in *Proc. Int. Conf. Parallel Process.*, Oct. 2013, pp. 552–561.

[28] K. M. Greenan, X. Li, and J. J. Wylie, "Flat XOR-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST)*, May 2010, pp. 1–14.

[29] O. Khan, R. C. Burns, J. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads," in *Proc. FAST*, Feb. 2012, p. 20.

[30] Z. Shen, P. P. C. Lee, J. Shu, and W. Guo, "Cross-rack-aware single failure recovery for clustered file systems," *IEEE Trans. Dependable Secure Comput.*, to be published.

[31] J. Hu, Q. Zhuge, C. J. Xue, W.-C. Tseng, and E. H.-M. Sha, "Software enabled wear-leveling for hybrid PCM main memory on embedded systems," in *Proc. Conf. Design, Automat. Test Eur.*, 2013.

[32] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2009, pp. 14–23.

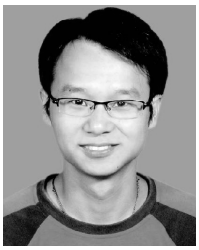
[33] S. K. Park, M. Maeng, K.-W. Park, and K. H. Park, "Adaptive wear-leveling algorithm for PRAM main memory with a DRAM buffer," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 4, Nov. 2014, Art. no. 88.

[34] L. Jiang, Y. Du, Y. Zhang, B. R. Childers, and J. Yang, "LLS: Cooperative integration of wear-leveling and salvaging for PCM main memory," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw.*, Jun. 2011, pp. 221–232.

[35] C.-H. Chen, P.-C. Hsiu, T.-W. Kuo, C.-L. Yang, and C.-Y. M. Wang, "Age-based PCM wear leveling with nearly zero search cost," in *Proc. Design Autom. Conf.*, Jun. 2012, pp. 453–458.
[36] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 24–33, 2009.
[37] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," *ACM SIGARCH Comput. Archit.*, vol. 38, no. 3, pp. 141–152, Jun. 2010.
[38] E. Ipek, J. Condit, E. B. Nightingale, D. Burger, and T. Moscibroda, "Dynamically replicated memory: Building reliable systems from nanoscale resistive memories," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 1, pp. 3–14, Mar. 2010.

SONGYUN WANG is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research interests include distributed systems and cloud computing.

JIABIN YUAN is currently a Professor with the Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. He is also a Research Fellow with the Laboratory for Grid and Cloud Computing. His research interests include cloud computing, cryptography, and distributed systems.



XIN LI received the B.S. and Ph.D. degrees from Nanjing University, in 2008 and 2014, respectively. He is currently an Associate Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. He is also with the State Key Laboratory for Novel Software Technology, Nanjing University. His research interest includes cloud computing, data management and analysis, distributed computing, and ubiquitous computing. He has published more than 20 research papers in related fields.

ZHUZHONG QIAN received Ph.D. degree in computer science from Nanjing University (NJU), in 2007, where he is currently an Associate Professor with the Department of Computer Science and Technology. He is also a Research Fellow of the State Key Laboratory for Novel Software Technology.

His research interests include cloud computing, distributed systems, and pervasive computing. He has published more than 80 research papers in related fields. He is the Chief Member of several national research projects on cloud computing and pervasive computing.

FABIO ARENA received the bachelor's and master's degrees in telecommunication engineering from the University of Catania, in 2006 and 2010, respectively. He is currently pursuing the Ph.D. degree with the Kore University of Enna. His current research interests include ITS, driverless vehicle, and network architecture.

ILSUN YOU received the M.S. and Ph.D. degrees in computer science from Dankook University, Seoul, South Korea, in 1997 and 2002, respectively, and the Ph.D. degree from Kyushu University, Japan, in 2012. From 1997 to 2004, he was with THINmultimedia, Inc., Internet Security Company Ltd., and Hanjo Engineering Company Ltd., as a Research Engineer. He is currently an Associate Professor with the Department of Information Security Engineering, Soonchunhyang University. His current research interests include internet security, authentication, access control, and formal security analysis. He is a Fellow of the IET. He has served or is currently serving as a main organizer of international conferences and workshops such as MIST, MobiSec, MobiWorld, and so forth. He is the Editor-in-Chief of the *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* (JoWUA). He is in the Editorial Board of *Information Sciences* (INS), the *Journal of Network and Computer Applications* (JNCA), IEEE Access, *Intelligent Automation and Soft Computing* (AutoSoft), the *International Journal of Ad Hoc and Ubiquitous Computing* (IJAHUC), *Computing and Informatics* (CAI), and the *Journal of High Speed Networks* (JHSN).