Stochastic Configuration Networks Based Adaptive Storage Replica Management for Power Big Data Processing

Changqin Huang, Member, IEEE, Qionghao Huang, Student Member, IEEE, Dianhui Wang*, Senior Member, IEEE

Abstract—**In the power industry, processing business big data from geographically distributed locations, such as online line-loss analysis, has emerged as an important application. How to achieve highly efficient big data storage to meet the requirements of low latency processing applications is quite challenging. In this paper, we propose a novel adaptive power storage replica management system, named *PARMS*, based on stochastic configuration networks (SCNs), in which the network traffic and the data center (DC) geo-distribution are taken into consideration to improve data real-time processing. First, as a fast learning model with less computation burden and sound prediction performance, the SCN model is employed to estimate the traffic state of power data networks. Then, a series of data replica management algorithms is proposed to lower the effects of limited bandwidths and a fixed underlying infrastructure. Last, the proposed PARMS is implemented using data-parallel computing frameworks (DCFs) for the power industry. Experiments are carried out in an electric power corporation of 230 million users, *CSG*, and the results show that our proposed solution can deal with power big data storage efficiently and the job completion times across geo-distributed DCs are reduced by 12.19% on average.**

Index Terms—**Power big data processing, stochastic configuration networks, geo-distributed, cloud storage, data replica optimization.**

I. INTRODUCTION

As a key national infrastructure, the power grid is rapidly moving towards automation and intelligence, and with the widespread use of IoT technologies, power grid data takes on typical big data features, such as the large volume of exponential growth [1]. The processing and utilization of big data plays an important role in smart power grid development. In many big data applications, near real-time data processing, such as online power line-loss calculation, is the most frequent and challenging. In addition, the underlying network architecture can not be casually changed for a relatively fixed infrastructure [2]. Therefore, based on the existing data storage infrastructure, how to meet the requirements of low latency data processing becomes increasingly significant to intelligent power data applications. In general, there are two major strategies to solve the aforementioned issue originating from limited bandwidths and datacenter geo-distribution, i.e. task scheduling and data replica management [3]. Currently, a large body of research has been devoted to the task scheduling, however, the performance of these works greatly varies because of the difference in application types, so the former is not fully appropriate for a variety of power data applications [4], [5]. Nevertheless, the data replica strategy is emerging as a promising approach to provide underlying support for a diverse array of applications [6]. When utilizing cloud computing with data replica, some solutions for data storage achieve a certain performance [7], [8], however, they are designed in a centralized fashion, in which all data is pulled to a single cluster for processing, rending them inefficient in many power data applications due to inadequate bandwidths or long latency [5]. Thereby, under the data-parallel computing frameworks (DCFs, *e.g.* MapReduce, Dryad, Spark), developing adaptive storage replica management over geo-distributed data centers(DCs) is a practical and preferable solution for power big data processing issues [9].

Traffic state prediction is critical to optimize data storage over distributed networks. A newly developed randomized learner model, termed stochastic configuration networks (SCNs) [10], can learn complex mappings from sampling data quickly with reasonably good modelling performance, compared to neural networks built using the well-known error back-propagation learning algorithm. Therefore, we adopt SCNs to implement this highly time-sensitive network traffic state prediction in a real-time fashion. Based on the fast learning benefits of SCNs, we develop *PARMS* by taking some power data processing characteristics (e.g. cyclicity, instantaneity) into account. Our proposed *PARMS* has been successfully applied to a real-world application in the *China Southern Power Grid (CSG)* - one of the two biggest power corporations in China, providing electric power supplies for *230* million people from *5* provinces over one million square kilometers.

Overall, this paper makes three main contributions:

Changqin Huang is with Department of Educational Technology, Zhejiang Normal Univerity, Jinhua, China.

Qionghao Huang is with School of Information Technology in Education, South China Normal University, Guangzhou, China.

Dianhui Wang is with Department of Computer Science and Information Technology, La Trobe University, Melbourne, Australia.

* Corresponding author (dh.wang@latrobe.edu.au).

1) Practical, near real-time, SCN-based network traffic prediction is addressed for a time-sensitive scheduling situation in data-parallel computing.
2) Adaptive replica management approaches integrating traffic prediction are proposed to lower the consequence of limited bandwidth and an underlying infrastructure for power big data processing.
3) A systematic, efficient cloud storage framework is designed and implemented to support power big data applications across geo-distributed DCs using DCFs in the power industry.

The remainder of this paper is organized as follows: Section III reviews the related work, Section IV details the proposed algorithms for *PARMS*, Section V describes the system implementation, and Section VI reports the results of the experiments, followed by the conclusion in Section VII.

II. Related Work

With high performance and lower cost, cloud storage is widely used in various fields, including the power industry[2], [7], [8]. Using core technologies in cloud computing, researchers have proposed a storage architecture to monitor data management and a free table storage model in the NoSQL-based LaUD-MS system [11]. Jun Zhu et al. [12] proposed a framework-based software approach to address key utility big data storage and processing issues. Yongheng Li et al. [7] designed a Hadoop and *HDFS* based big data system. In addition, cloud storage technologies have been applied into some specific scenarios in the power industry [13], [14], [1].

Most of the research in this area is designed for a centralized computing system, which results in poor performance when tasks of DCFs [15] dispatch across geo-distributed DCs [5], which are common scenarios in the power industry. To lower the consequences of limited bandwidth and to achieve low latency of data access in such scenarios, some job scheduling algorithms are proposed, such as WANalytics [16], Pixida [17], DESRP [18], Gaia [4], Flutter [5], *etc*. Pixida [17] employs a generalized min-k-cut algorithm to cut the DAG into several parts for execution, and each part is executed in one data center. Gaia [4] adapts a new synchronization model to avoid unnecessary global synchronization, and thus preserves the traffic across geo-distributed data centers.

In addition to job scheduling, optimizing replica management in distributed DCs is an important solution [3], and widely used in various domains, such as mobile networks [19], energy-saving management [20], video services [21], social networks [22], virtual machine management [23]. To address the important placement problem in replica technology, Xili Dai et al. [21] proposed a scheduling mechanism based on a tripartite graph and k-list algorithm, which optimizes access latencies while maintaining the benefit of low storage cost. Conducting analytical studies and experiments that identify the performance issues of MapReduce in data centers, based on a topology-aware heuristic algorithm, Incheon Paik et al. [24] presented an optimal replica data placement, minimizing global data access costs. Yi Wang et al. [25] proposed a K-SVD sparse representation technique to improve the smart meter data compression ratio and classification accuracy in the electricity industry. In relation to another important problem in replica technology, using the multithreaded and integrated maximum flow, Nihat Altiparmak et al. [26] present an optimal replica selection algorithm to handle heterogeneous storage architectures, and compared with a maximum flow algorithm in a black-box manner, this approach reduces massive amount of unnecessary flow calculations, achieving less latency in response. To reduce the data availability time, and data access time, Nagarajan et al. [27] developed the replication algorithm that makes decisions on the selection and placement of replica using multiple criteria. This algorithm considers multiple parameters such as storage capacity, bandwidth, and the communication cost of geo-distributed sites.

However, most of these works focus on the optimization of general domains, with only a few focusing on storage scheduling optimization for geo-distributed power big data systems, and furthermore the characteristics of the power industry received little attention in their research.

III. PARMS: Motivation and Solution

In this section, we introduce the motivation of *PARMS*, then give an overview of the proposed solution.

A. Characteristics of the Power Industry

Taking *CSG* for example, here we describe some distinct characteristics of big data processing in power industries, which serve as our motivation for the proposed algorithms or developed applications.

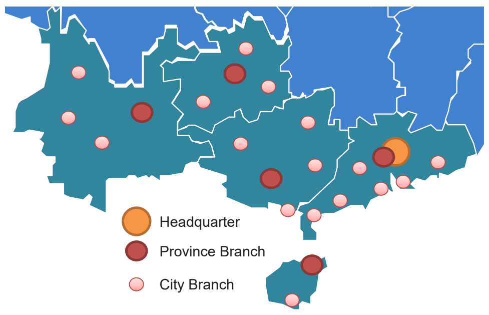


Fig. 1. A brief view of the hierarchical division of *CSG*.

*a) **Hierarchicality of Data Centers**:* As Fig. 1 shows, *CSG*[1] has strict hierarchical administrative and geographic divisions, which consists of the headquarter, province branches, and city branches. Thus, as Fig. 2 shows, *CSG* establishes relatively fixed geo-dispersed DCs with the same hierarchy as the geographic division,

[1]http://www.csg.cn

which cannot casually alter the network topology due to security or other concerns.

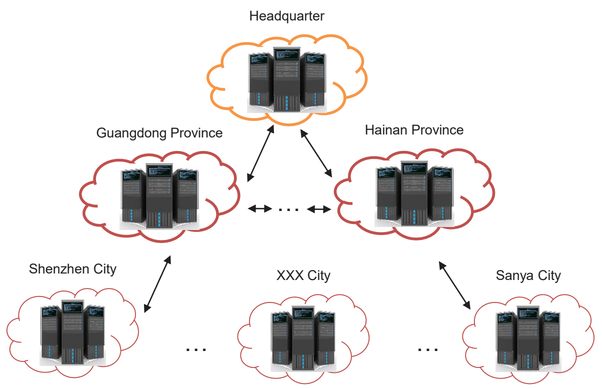


Fig. 2. The geo-distributed DCs are connected with private commercial high-speed data links, the topology and bandwidth of network between DCs is relatively stable.

*b) **Instantaneity of Data Processing**:* Processing data is cpu-intensive, memory-intensive, and cpu-memory-intensive. However, lots of power big data-based applications require real-time operations with strict demands on low latency across geo-distributed DCs. Limited bandwidths are likely to cause a bottleneck for the low latency demand of big data processing applications across geo-distributed DCs. Thus, with such a strictly stable structure of DCs, how to develop a practical solution to lower the implications caused by limited bandwidths and geo-distributed DCs is a very urgent and challenging problem to be solved.

*c) **Cyclicity or Stability of Data Processing**:* The locations of sensors, the volume of data collected, and the time to transfer the collected data are relatively stable. Taking a city named Zhongshan in Guangdong Province for example, smart meters and other sensors transfer collected data to DCs every *15mins* and about *2-3 Terabytes* of textual data is collected every month. So, there is an obvious cyclicity of the time to collect data from sensors. With such potential cyclicity or patterns in data processing, it is possible to develop a solution based on machine learning or artificial intelligence to minimize data motion across geo-distributed DCs.

*d) **Heterogeneity of Computation and Storage Resources**:* Due to the high computation demand and storage needed for power big data processing, many devices with different capacities (*e.g. CPU* speed, IOPS) in computation or storage are continuously deployed to the centers, resulting in a significant heterogeneity to the performance of computing or storage servers, and the inter-datacenter bandwidths are also heterogeneous. Such a heterogeneity results in some scalability and extendibility problems for the systems.

B. Overview of PARMS

To achieve the low latency processing of power big data across the relatively fixed geo-distributed DCs, we design a traffic prediction-based adaptive replica management system, *PARMS*, for power big data processing.

Fig. 3 illustrates the four main components of the architecture of *PARMS*. The tracing daemons and threads in the clusters monitor the system and collect running information for *GaExUnit*; *GaExUnit* reprocesses the logs and forwards them to *Intelligent Analysis System* for analysis. With the output from *Intelligent Analysis System*, the replica management component in *GaExUnit* runs the algorithms to optimize the placement and selection of replica, and *Optimizer Daemon* executes the optimal instructions.

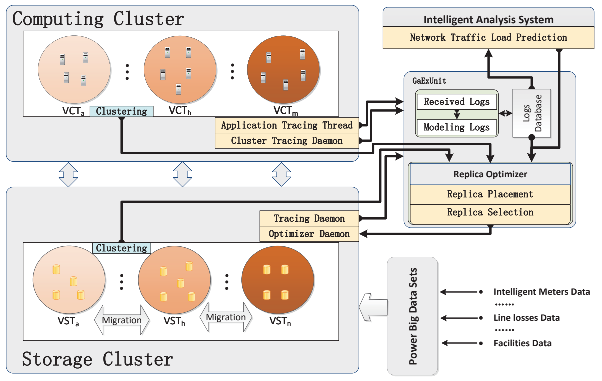


Fig. 3. The architecture of *PARMS*: A network traffic prediction-based adaptive replica management system for a geo-distributed cloud storage system in the power industry.

A summary of our solution, *PARMS*, is as follows:

1) We adapt a fuzzy C-means-based device clustering algorithm to deal with the scalability and extendibility problem brought by the heterogeneity in computing or the storage servers for the power industry: (Section IV-A).
2) We propose a practical, near real-time SCN-based network traffic prediction mechanism with deep consideration characteristics (such as, hierarchicality, cyclicity, stability) in the power industry (Section IV-B).
3) We design power big data processing-oriented replica placement and selection algorithms to accelerate data access across geo-distributed DCs, and satisfy the low latency demand of data processing in the power industry (Section IV-D and IV-E).

IV. ALGORITHMS FOR *PARMS*

We introduce clustering algorithms based on fuzzy C-means to *classify* the capacity of underlying devices, which provides valuable indexes of underlying devices for replica management or system management. Based on a deep learning model and the application-level information of the computing tasks, we employ a network traffic load prediction framework to provide possible network traffic in the near future for replica management.

In the following, we detail the related algorithms for *PARMS*.

A. Classifying Underlying Devices

To deal with the stability and extendibility problem caused by the heterogenous performance of computing or storage servers, we categorize computing and storage servers into different logical groups by applying a clustering algorithm based on fuzzy *C*-means [28].

The power big data system structure can be simply described as a directed graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ where the set of vertices $\boldsymbol{V} = \boldsymbol{CN} \bigcup \boldsymbol{SN}$, $\boldsymbol{CN} = \{cn_1, ..., cn_i, ..., cn_{n_c}\}$ indicates computing nodes, $\boldsymbol{SN} = \{sn_1, ..., sn_j, ..., sn_{n_s}\}$ indicates data storage nodes (also called data nodes), and $\boldsymbol{E}$ represents the transmission network links among the nodes. Assume that there are n computing nodes or data nodes in a system, with each node having n_p attributes that determine the performance of the node (*e.g.* CPU speed, IPOS). We denote $pf_{i,k}$ $(1 \leq k \leq n_p, k \in \mathbb{N})$ as the k^{th} attribute of the i^{th} node. Thus, all attributes of the i^{th} node can be represented as a vector:

$$\boldsymbol{PF}_i = (\lambda_1 pf_{i,1}, ..., \lambda_k pf_{i,k}, ..., \lambda_{n_p} pf_{i,n_p}), \boldsymbol{PF}_i \in \mathbb{R}^{n_p}, \quad (1)$$

where λ_k is a coefficient of the j^{th} attribute that normalizes the various ranges of attribute values into *0-1*.

Stacking all the attributes of n computation or storage nodes, we have a matrix:

$$\boldsymbol{PF} = (\lambda_k pf_{i,k})_{n \times n_p}, \mathbf{PF}_i \in \mathbb{R}^{n \times n_p}, \quad (2)$$

$\boldsymbol{PF}$ acts as the input for the clustering algorithm. Its output is

$$\boldsymbol{LST} = \{\boldsymbol{LST}_l | l \in [1, c]\}, \boldsymbol{LCT} = \{\boldsymbol{LCT}_l | l \in [1, c]\}, \quad (3)$$

where a subscript of $\boldsymbol{LCT}$ or $\boldsymbol{LST}$ represents a node cluster, as well as being an indicator of its capacity to deal with data.

B. SCN-Based Network Traffic Prediction

Recently, some researchers used application-level data access patterns to predict traffic flow in data-parallel computing-based big data processing systems, and achieved better performance when predicting flow size than traditional prediction algorithms [15], [29]. As discussed in Section III-A, compared with other big data processing platforms, there are more constraints on the architecture of DCs, the data collection methods, and the execution of power big data processing tasks, such as hierarchicality, cyclicity and stability. These characteristics enable us to develop a mechanism to learn *priori knowledge* of task execution times in a *DAG* provision. Therefore, we propose a practical mechanism to predict network traffic over a period time in the future.

As Fig. 4 shows, the mechanism consists of three parts:

1) An operator execution time fitting model-based stochastic configuration network [10].
2) A mechanism to extract DAG information from data-parallel computing applications and calculate flow size information out of each stage.
3) A time series analysis of job execution logs to find certain patterns in the job execution order.

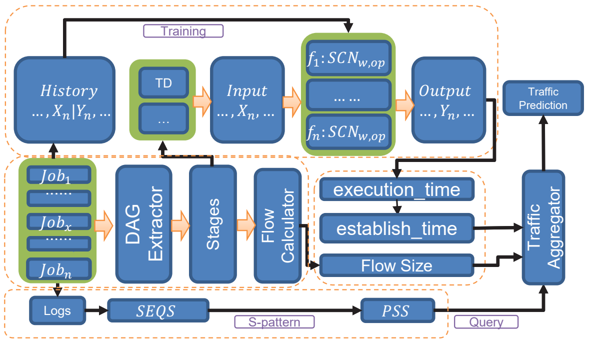


Fig. 4. The mechanism of network traffic prediction.

In the following, we detail our framework and provide definitions and relevant algorithms.

Definition 1. *TD (Task Descriptor), A descriptor of a computing task executed by a worker node, is denoted as:*

$$TRC = \langle IS, DT, Pri, WCID, JCID, CPU, Mem, OP \rangle \quad (4)$$

where IS is an integer indicating the input size of a computing task, DT is an integer indicating the type of collected data categorized by CSG, Pri is an integer for scheduling , CPU and Mem are computing resources assigned by the scheduler, $WCID$ is the clustering number from the clustering algorithm, $JCID$ is an integer indicating whether a computing task is processor-intensive, memory-intensive, or I/O-intensive, and OP is a code number representing operator (*e.g. map(), union()*) of data.

Definition 2. *Task Event, a descriptor of an operator and the volume of data to be processed, denoted as en_k. The set of task events with respect to operators provided by DCFs as an event set of $\boldsymbol{\varepsilon} = \{en_k\}$*, where we have $k = 1, .., n_e$.

The en_k can be represented by tuple (OP, IS), which consists of an operator and the size of the input data. Then the state of a task event at one time running in the worker denoted as $R_{TE}(\boldsymbol{EN})$, where $\boldsymbol{EN}$ consists of the task event and its duration is denoted as $\boldsymbol{EN} = \{(en_k, t_k)\}$. In particular, the value t_k is zero when the task event is complete or null. The state of work nodes cn_i at time t can thus be denoted as:

$$\boldsymbol{S}_{i,t} = R_{TE}(i, (en_k, t_{k,i}), ..., (en_{n_e}, t_{n_e,i})). \quad (5)$$

The fitting model accepts the changes in these states that cause the fluctuation in task execution time. The changes of variable values in Eqn. (5) over a time interval is calculated as follows:

$$\begin{aligned} \Delta\boldsymbol{S}_{i,\Delta t} &= ((\boldsymbol{S}_{i,t} \ominus \boldsymbol{S}_{i,t'}), t) \\ &= (R_{TE}(i, (en_1, \Delta t_{i,1}), ..., (en_{n_e}, \Delta t_{i,n_e})), t). \end{aligned} \quad (6)$$

where Δt means the time left to finish a task event.

With the above definitions and formulation, we now introduce our fitting model based on SCN. Compared with traditional machine learning methods (*e.g.* BP, RBF neural networks, SVM) [30] or other deep learning models (*e.g.* LSTM [31]), SCN introduces little overhead to the system while achieving reliable prediction results, which we demonstrate in the experiment. The input and output of the fitting model f_i are simplified as:

$$\boldsymbol{X}_{i,t} = (t, TE, \Delta \boldsymbol{S}_{i,\Delta t}), \boldsymbol{Y}_{i,t} = t_{TE}, \tag{7}$$

We use a similar mechanism in [15] which serves as the second part of our prediction framework. In addition to providing operator logs for the fitting model, it outputs a tri-tuple (*source, destination, flow_size*). With this information, we use the prediction results from the SCN model to predict the time for flow dumping into the real-world network.

To mine certain or potential cyclicity in the power industry as discussed in Section III-A, we employ simple but efficient sequence pattern mining algorithms which serve as the third part of our network traffic prediction framework.

C. Replica Management

As an essential part of software systems for cloud storage, replica technologies play a decisive role in improving concurrent access, the reliability and availability of data [32].

Replica management consists of replica generation, replica deletion, replica placement and replica selection. Several important factors exert significant influence on the performance of replica management, such as the capacity of replica hosts, the network state of replica locations, and the characteristics of data-related applications (such as data queries and data analysis).

In this part, we give a brief introduction of the main components of replica management in *PARMS*. In this paper, we detail the relevant algorithms or mechanism for replica placement(Section IV-D) and replica selection (Section IV-E), which are specially designed for the power industry. As for replica generation and replica deletion, based on our previous work [33], we make some alternations to use in this work.

a) Replica Generation: Power cloud storage systems are highly dynamic computing environments, and data blocks of different data vary greatly in data access *patterns* or *hotness*. Based on the replica generation mechanism in our work [33], we update the relevant definition of *ideal access popularity* and *actual access popularity* as well as their computing methods. After such an alteration, this component can provide the others with API to generate appropriate data blocks with new-coming or existing data in power cloud storage systems.

b) Replica Deletion: After running for a certain amount of time, some *outdated* replica may exist in the system which should be set free to optimize data storage and network transmission. There are several ways to delete *outdated* replica, such as delayed deletion, offline deletion, and marked deletion [34]. Before deletion, the system needs to design a mechanism to decide what kinds of replica are *outdated*, hence we have slightly modified our previous work in [33] to achieve replica deletion in *PARMS*.

c) Replica Placement: Based on the results of SCN-based network traffic prediction (Section IV-B), the data access popularity calculations and other important factors, we design a dynamic replica placement algorithm to optimize data storage and network transmission in the power industry. The relevant definitions and algorithms are detailed in Section IV-D.

d) Replica Selection: Running computing tasks on power big data processing platforms is a typical multi-task scenario over geo-distributed DCs, and the preferences of power big processing tasks differ greatly from each other. We develop a practical framework to transform this into a multi-objective optimization problem. The relevant definitions and algorithms are presented in Section IV-E.

D. Replica Placement

As mentioned in Section III-A, there are some potential patterns or a certain cyclicity in data parallel computing across the relatively fixed geo-distributed DCs in a power big data system. The frequency of data access differs greatly from different applications, and results in degrees of *hotness* or *coldness* for data blocks. Therefore, we need to make optimal decisions on where to place replica and how to select replica, by considering replica factors and storage locations.

Data blocks with the same access frequency may have different popularities, which vary according to different computing tasks. Each data block and its replicas are associated with a queue of timestamps that record the values of their access popularity. Specifically, we calculate the popularity:

$$\begin{aligned} heat_0(b_i) &= 0, rf = k \times R_t + f \times F_t, \\ heat_{t+1}(b_i) &= \frac{heat_t(b_i) \times log_2(e^{\lambda(T_{t+1}-T_t)})^{-2} + rf}{Z}, \end{aligned} \tag{8}$$

where $heat_{t+1}(b_i)$ is the updated value of the access popularity of data block b_i at time $t+1$, the attenuation function $(log_2(e^{\lambda(T_{t+1}-T_t)})^{-2}$ deteriorates the replica's access popularity over time, with a cooling coefficient λ , k and f consonant coefficients with $k \in (0,1)$ and $f \in (-1,1)$, R_t the number of accesses at time t, F_t the number of accesses probably occurring in the forecasting sequence of I/O events over a period from $\boldsymbol{SEQS}$, and Z is a normalization factor.

A replication factor is assigned to each data block in the history logs of $\boldsymbol{SEQS}$ by using maximum likelihood estimation. After the access popularity (also called *hotness*) of the data blocks is calculated, their relationship is given as follows:

$$Req(b_i) = heat_T(b_i|\hat{\theta}), \tag{9}$$

where $Rep(b_i)$ is the replication factor of b_i, and $heat_T(b_i)$ is the access popularity of b_i at time T. We use the maximum likelihood to estimate parameter $\theta \in \Theta$:

$$\hat{\theta}_{MLE} = arg \max_{\theta \in \Theta} \sum_{i=1}^{n} ln(heat_T(b_i|\theta)), \tag{10}$$

Therefore, the replication factor of each data block in a big data cloud storage system can be specified in a granular way.

Algorithm 1 Dynamic Replica Placement

```
Input: $\boldsymbol{LST}$, $\boldsymbol{SN}$]/*A set of n data nodes*/, $\boldsymbol{B}$/*A set of m data blocks*/, $\boldsymbol{CN}$/*A set of g computing nodes*/
Output: $\boldsymbol{RP}[n][m]$ //The matrix represents the layout of $m$ data blocks on $n$ data nodes.
1: for $l$ from 1 to $c$ do
2:     $\boldsymbol{LST}_l = \{sn_j^l | i \in [i, Size(\boldsymbol{LST}_l)\} \leftarrow SBI(\boldsymbol{LST}_l);$
3:     $S(\boldsymbol{LST}_l) \leftarrow SumFreeSpace(all\ sn_j^l \in \boldsymbol{LST}_l)$
4: end for//Sort the data nodes within the same cluster, and sum their free space.
5: for $i$ from 1 to $m$ do
6:     $\boldsymbol{DS}_{pi} = \{b_i^{pi} | pi \in [1, p]\} \leftarrow SortByTaskPref(b_i);$
7: end for// Categorize data blocks by the preference of geo-distributed computing tasks.
8: for $pi$ from 1 to $p$ do
9:     $\boldsymbol{H}_{pi} = \{Heat(b_i) | b_i \in \boldsymbol{DS}_{pi},\ i \in [1, Size(\boldsymbol{DS}_{pi}]\} \leftarrow GetHotness(\boldsymbol{DS}_{pi});$
10:    $\boldsymbol{DS}_{pi} = \{b_i | pi \in [1, Size(\boldsymbol{DS}_{pi}]\} \leftarrow SBH(\boldsymbol{H}_{pi});$
11: end for//The sort of data blocks with the same $\boldsymbol{DS}_{pi}$ by their hotness.
12: $\boldsymbol{RF} = \{Rep(b_i) | i \in [1, m]\} \leftarrow (\boldsymbol{H}, \boldsymbol{B})$
13: $\boldsymbol{L} = \{\boldsymbol{L}_{\tau,t} | t \in [1, T]\} \leftarrow DoForcast(f_\tau);$// Forecast the network traffic load
14: for $pi$ from 1 to $p$ do
15:     for $i$ from 1 to $Size(\boldsymbol{DS}_{pi})$ do
16:         $\boldsymbol{CanN} \leftarrow SelectNode(\boldsymbol{SN}, \boldsymbol{LST});$//Obtain the set of candidate data nodes
17:         $\boldsymbol{FNode} \leftarrow EstimateNetLoad(\boldsymbol{CanN}, \boldsymbol{L}_{\tau,t}, Failover);$
18:         $Update(\boldsymbol{RP}[n][m], \boldsymbol{FNode})$
19:     end for
20: end for
21: return $\boldsymbol{RP}[n][m]$
```

Thus, we have described our prediction model, the logical groups of storage servers, the calculations of access popularity and the replication factors. Now, we present a dynamic replica placement algorithm that increases the system throughput and data transfer rate by optimizing the usage of network transmission across geo-distributed DCs. Algorithm 1 is the algorithm for replica placement.

E. Replica Selection

After replica placement, selecting the best replica to satisfy the instantaneity of data processing demands as discussed in Section III-A is a challenging problem in a multi-task scenario. In order to measure the serviceability of the replica, we select three important metrics, *response time, network traffic load, and reliability*. They are weighted according to the different QoS requirements of data access for a given computing task, that is, $\boldsymbol{w} = (w_1, w_2, w_3)$, where $w_1 + w_2 + w_3 = 1$, $(0 < w_i < 1, i = 1, 2, 3)$.

a) ***Matrix of Selection:*** a possibility of replica selection, denoted as $\boldsymbol{PM}$. Assume that n computing nodes of a given computing task that requests the same replica as a set $\boldsymbol{RC} = \{rc_1, rc_2, ..., rc_{n_{rc}}\}$ and m data nodes that keep the replica as a set $\boldsymbol{RS} = \{rs_1, rs_2, ..., rs_{n_{rs}}\}$. The $\boldsymbol{PM}$ of n computing nodes and m replicas of the data block can be described as follows:

$$\boldsymbol{PM} = \boldsymbol{RC}^T \boldsymbol{RS} = (pm_{i,j})_{n_{rc} \times n_{rs}}. \tag{11}$$

where $pm_{i,j} = 1$ means that computing node rc_i requests replica rs_j through data node j, and $pm_{i,j} = 0 (1 \leq i \leq n_{rc}, 1 \leq j \leq n_{rs})$, otherwise. Since there is only one replica for a request at one time, it is obvious that we have $\sum_{j=1}^{n_{rs}} pm_{i,j}$,where$1 \leq i \leq n_{rc}$.

b) ***QoS_1 of response time:*** The performance of data transmission between nodes is mainly determined by the capacity of transmission network between them. $v_{i,j}$ is calculated by the historical network throughout NT, the state of running network NV, and the IPOS of storage servers L:

$$v_{i,j} = \frac{1}{\alpha' \times NT + \beta' \times NV + \gamma' \times L'}, \tag{12}$$

where coefficients α', β', and γ' are obtained by measuring the correlations between NT, NV, and L from history logs and current running information. Thus, the metric matrix of $\boldsymbol{QoS}_1$ can be written as:

$$\boldsymbol{QoS}_1 = (v_{i,j})_{n_{rc} \times n_{rs}}. \tag{13}$$

c) ***QoS_2 of network traffic load:*** The network traffic load between the nodes across geo-distributed DCs is also an important factor for selecting a replica. $nl_{i,j}$ is an indicator of the network traffic load determined by the current traffic load KNL and the predicted FNL from the fitting model f_τ:

$$nl_{i,j} = \frac{1}{\mu \times KNL + (1 - \mu) \times FNL'}, \tag{14}$$

where $\mu (0 \leq \mu \leq 1)$ is set by examining history logs. Therefore, the metric matrix $\boldsymbol{QoS}_2$ is as follows:

$$\boldsymbol{QoS}_2 = (nl_{i,j})_{n_{rc} \times n_{rs}} \tag{15}$$

d) ***QoS_3 of reliability:*** The high reliability of data access is essential to success in completing computing tasks, especially for those with high priority and high frequency. The rs_j^r measures the reliability that may involve the quantitative stability of the hosting node denoted as HP, the duration t_{srv}, the creation time t_{gen} of the replica, and the current time of system t_{now}. It is calculated as:

$$rs_j^r = \rho \times HP + w \times \frac{t_{srv}}{t_{now} - t_{gen}}. \tag{16}$$

Therefore, the $\boldsymbol{QoS}_3$ of reliability can be formed as a vector

$$\boldsymbol{QoS}_3 = (rs_1^r, rs_2^r, ..., rs_{n_{rs}}^r). \tag{17}$$

*e) **Objective functions** F_1, F_2, **and** F_3:* The different values of $\boldsymbol{PM}_{n_{rc}\times n_{rs}}$ can be regarded as different possibilities of replica selection. The values of $\boldsymbol{QoS}_1$, $\boldsymbol{QoS}_2$, and $\boldsymbol{QoS}_3$ for each $\boldsymbol{PM}_{n_{rc}\times n_{rs}}$ are represented by

$$\begin{cases} F_1(\boldsymbol{PM}) = \boldsymbol{e}\ (\boldsymbol{PM} \odot \boldsymbol{QoS}_1)\ \boldsymbol{e}^T \\ F_2(\boldsymbol{PM}) = \boldsymbol{e}\ (\boldsymbol{PM} \odot \boldsymbol{QoS}_2)\ \boldsymbol{e}^T \\ F_3(\boldsymbol{PM}) = \boldsymbol{e}\ (\boldsymbol{PM} \odot \boldsymbol{QoS}_3)\ \boldsymbol{e}^T \end{cases} \tag{18}$$

where $\boldsymbol{e}$ is a vector of all ones, $\boldsymbol{PM}$ is equivalent to $\boldsymbol{PM}_{n_{rc}\times n_{rs}}$, $F_1(\boldsymbol{PM})$ is the value of $\boldsymbol{QoS}_1$ for $\boldsymbol{PM}_{n_{rc}\times n_{rs}}$, So are $F_2(\boldsymbol{PM})$ and $F_3(\boldsymbol{PM})$. The operator $\odot$ is *Hadamard product*.

Algorithm 2 Replica Selection for Power Big Data Cloud Storage

```
Input: $w$
Output: $\boldsymbol{PM}_{optimal}$
1: $(\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3) \leftarrow \boldsymbol{W}$;
2: $\boldsymbol{PM} \leftarrow \boldsymbol{RC}^T\boldsymbol{RS}$//Initiate $\boldsymbol{PM}$ randomly.
3: $\boldsymbol{QoS}_1 \leftarrow (v_{i,j})$;//The configuration of $\boldsymbol{QoS}_1$.
4: $\boldsymbol{QoS}_2 \leftarrow (nl_{i,j})$; //The configuration of $\boldsymbol{QoS}_2$.
5: $\boldsymbol{QoS}_3 \leftarrow (rs_j^r)$ ;//The configuration of $\boldsymbol{QoS}_3$.
6: $F(\boldsymbol{PM}) \leftarrow (F_1, F_2, F_3) \odot \boldsymbol{W}(\boldsymbol{PM})$;
7: $\boldsymbol{PM}_{optimal} \leftarrow Solve(max\ F(\boldsymbol{PM}))$;
8: return $\boldsymbol{PM}_{optimal}$
```

It is desirable to select replica when each objective function reaches the maximum with a certain value of $\boldsymbol{PM}$. Further, different applications can set different preferences for $\boldsymbol{QoS}_i (i = 1, 2, 3)$ by specifying $\boldsymbol{w}_i$. Thus, the overall objective function for replica selection is obtained as:

$$F(\boldsymbol{PM}) = F_1(\boldsymbol{w_1} \odot \boldsymbol{PM}) + F_2(\boldsymbol{w_2} \odot \boldsymbol{PM}) + F_3(\boldsymbol{w_3} \odot \boldsymbol{PM}). \tag{19}$$

where we have $\boldsymbol{W} = (w_{i,j})_{n_{rc}\times 3}$, and $\boldsymbol{w}_j = (w_{i,j})_{n_{rc}\times 1}$, $(j = 1, 2, 3)$.

The maximum of $F(\boldsymbol{PM})$ is to find the best selection matrix $\boldsymbol{PM}_{optimal}$ for the objective function. Thus, the solution to replica selection is to find the approximate optimal solution to $max\ F(\boldsymbol{PM})$, and we implement a parallel genetic algorithm (*kGA*) on the CUDA architecture described in Petr's work [35] to solve the problem. The whole algorithm is given in Algorithm 2.

V. System Implementation

We now describe the implementation of the main modules of *PARMS*. We employ Ganglia [36] for system monitoring, which can run customized scripts with very low per-node overheads and high concurrence, and integrate this with Spark or Hadoop monitoring APIs. We also develop customized scripts to collect monitoring data as the inputs to our algorithms, such as I/O event logs, and the QoS preferences of computing tasks. The collected information is managed by PostgreSQL (9.4).

The implementation of replica management policies comprises mainly of two parts: replica placement and replica selection. We incorporate our replica placement and selection inside the Hadoop Distributed File System (HDFS 2.6.0). Table I shows the main APIs of replica management policies, the Java version is 1.7.0_55. The APIs also work in the latter version such as 2.8.0, 2.7.4 of Hadoop HDFS and Kudu 1.4.0 Client. The Python based client scanner API will be implemented soon.

TABLE I
REPLICA MANAGEMENT POLICIES APIS

Method	Caller of Setter
public class DynamicBlockPlacementPolicy extends BlockPlacementPolicyDefault	dfs.block.replicator.classname
public static final ReplicaSelection QOS_PREF	client.scanner

For the convenience of the operators, we adapt both the C/S and B/S manner to develop a GUI-based client (Microsoft Windows platform, Java, Web) and a CMD-Line based client (mainly used in Linux, shell). The GUI-based client provides operators with a friendly, graphical interface to manage the *PARMS*. Fig. 5 shows a screenshot of the management client of the *PARMS* in Windows 7.

(a) C/S (b) B/S

Fig. 5. Screenshots of *PARMS*.

VI. Performance Evaluation

This section describes the experimental platform, the configurations of our experiments, and reports the experimental results.

A. Experiment Platform

We conduct the experiments on the geo-distributed power big data processing system of CSG. For the implementation and specification of the power big processing platform in *CSG*, readers can refer to our published paper [37], and a common latency-aware task scheduling strategy is employed to dispatch data-parallel computing tasks across geo-distributed DCs [3]. Table II shows the available bandwidths between nodes across geographically dispersed DCs in the experiments.

The computing tasks in the experiments are routine tasks or data mining programs in power big data systems of CSG. Examples are the line losses calculation in real time, the analysis of the power consumption behavior of consumers, and the abnormality monitoring and alarm in power consumption. The volume of data

TABLE II
GEO-DISTRIBUTED DATACENTERS OF *CSG* FOR *PARMS* EVALUATION, L1 REPRESENTS FOR THE HEADQUARTER, L2 FOR PROVINCE BRANCHES, AND L3 FOR CITY BRANCHES

	HQ(L1)	**GD Province(L2)**	**Zhongshan City(L3)**	**Foshan City(L3)**
HQ(L1)	1.5Gpbs	136Mbps	-	-
GD(L2)	-	1.2Gbps	-	-
Zhongshan City(L3)	-	85Mbps	1.01Gbps	43.5Mbps
FoShan City (L3)	-	78 Mbps	-	1.1Gbps

TABLE III
CONFIGURATION OF GEO-DISTRIBUTED TASKS

Item	**Application**	**Configuration**
Task A	Line-loss Calculation	Spark, 102GB
Task B	Power Consumption Analysis	Hadoop, 482GB
Task C	WordCount	Spark, 40GB
Task D	GrapX	Spark, 4847571 nodes, 68993773 edges

for computing tasks on the experimental platform is about *550G* from different geo-distributed cloud system in CSG, and some open datasets (WordCount, GraphX) are also employed to test the overhead of *PARMS* introduced to the systems (as Table III shown).

TABLE IV
TRAINING PARAMETERS OF *LSTM* IN KERAS

Para.	**Conf.**	**Para.**	**Conf.**
model	Sequential	add1	LSTM
add2	Dense(1)	dropout	0.2
batchsize	86	epochs	50
loss	mae	optimizer	adam

TABLE V
TRAINING PARAMETERS OF *SCN*

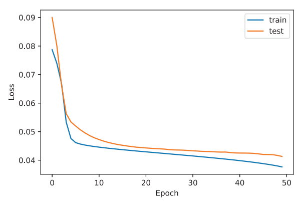
Para.	**Conf.**	**Para.**	**Conf.**
L_max	250	tol	0.001
T_max	600	Lambdas	[0.5, 1, ..., 250]
batchsize	1	r	[0.9, 0.99, ..., 0.999999]

B. Evaluation of SCN-Based Fitting Model

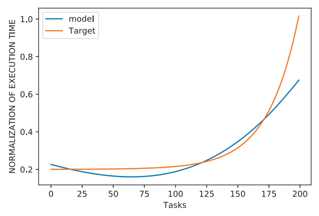
A practical traffic prediction algorithm is critical to implement replica management policies. As discussed in Section IV-B, the key to predicting traffic in DCFs is the information of the execution times of tasks. An experiment is conducted to evaluate the performance of the proposed SCN-based fitting model. The comparison is a competitive model based on LSTM. The configuration of the training parameters is shown in Tables IV and V. As shown in Fig. 6, the predicted results from the SCN-based fitting model are better than the LSTM-based one under the same computing resources and time limit. Even if the fitting model malfunctions due to significant changes in the fitting mappings, it costs little to rebuild a functional one.

C. Evaluations on Read Latency by Comparing Different Policies

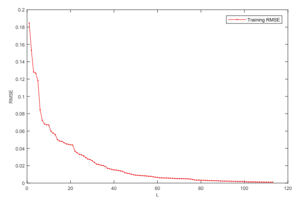
The experiment evaluates the read latency over geo-distributed DCs reduced by the proposed replica management policies. Fig. 7 plots the reading times for different sizes of data with respect to three kinds of replica



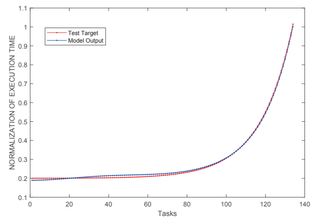
(a) The loss of LSTM-based fitting model when training.



(b) The predicted execution times from the LSTM-based.



(c) The loss of SCN-based fitting model when training.



(d) The predicted execution times from the SCN-based.

Fig. 6. The performance of the prediction of execution times of tasks by different fitting models.

management policies. From the measurement data of latency among nodes, both the default policy of *HDFS* and our proposed policy outperform the one without any policies. Our replica management policies achieve better performance, though the reading time still increases at a linear rate. This is, however, inevitable for data transfer, as it is limited by the network bandwidth and disk transfer rate. Overall, our algorithm is more effective in data access in the geo-distributed DCs. Therefore, it is more suitable for power big data processing across geo-distributed DCs.

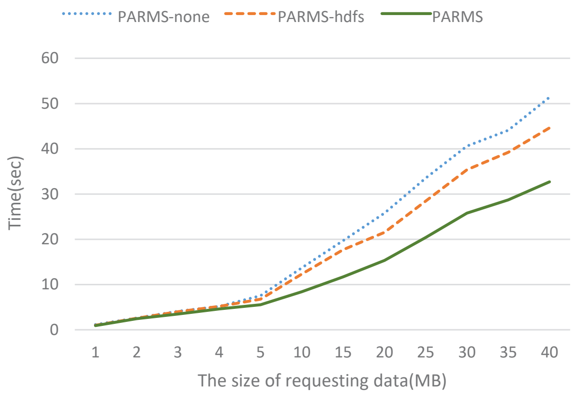
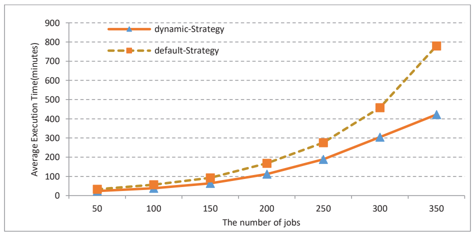


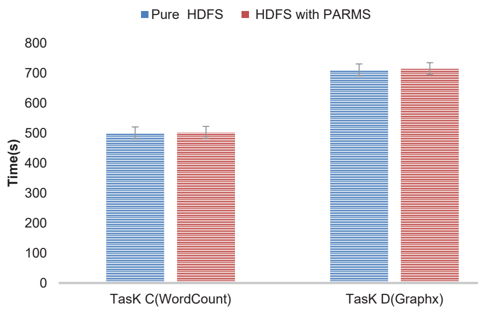
Fig. 7. The reading time for data of different sizes for three kinds of replica management policies (where the lower, the better). *PARMS-none* is denoted as the system without any replica management policies, *PARMS-hdfs* is the default policy, and *PARMS* is our approach.

D. Performance Comparisons of Dynamic Replica Placement Strategies

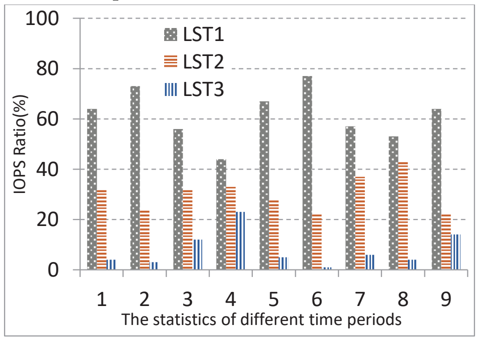
This experiment aims to evaluate the resultant enhancement of the execution of geo-distributed computing tasks by using the replica placement strategy and



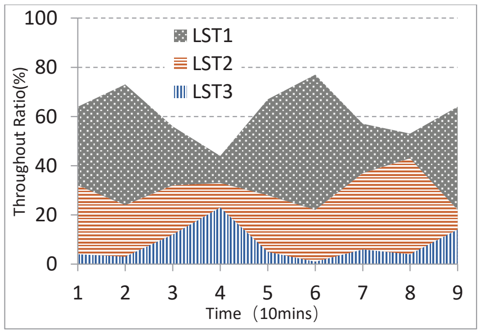
(a) The average execution time of jobs.



(b) Job completion time on DCFs (overhead test).



(c) The IOPS ratio of *LSTs*.



(d) The throughout ratio of *LSTs*.

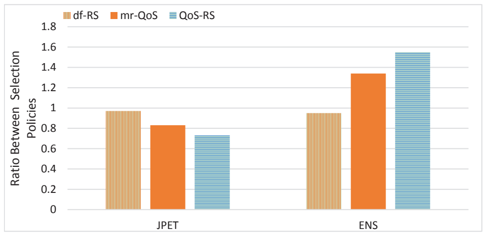
Fig. 8. The performance of replica placement strategies. Our approach is denoted as *dynamic-Strategy* and the default as *default-Strategy*.

the classification of data nodes. The performance is measured by the average execution time of jobs. In the experiment, the data nodes are categorized into three logical storage areas, LST_1, LST_2, and LST_3 (the smaller value of subscript, the better performance its associated node is). As shown in Fig. 8a, the *dynamic-Strategy* takes less than the average running time of executing jobs than *default-Strategy*. From Fig. 8b, we find out that the extra time introduced by *PARMS* is negligible.

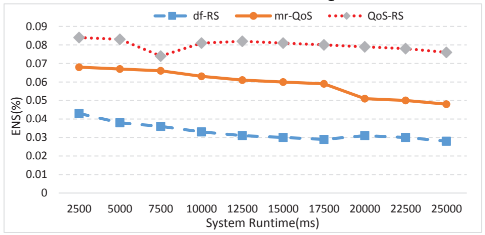
We also collected the IOPS and I/O throughput information of the logical storage areas. Fig. 8c and Fig. 8d show the statistical results. It is clear that LST_1 accounts for approximately 62%, LST_2 for 30%, and LST_3 for only 8% of both IPOS and I/O throughput on average.

E. Performance Comparisons of Replica Selection Strategies

This experiment validates whether the replica selection policy can satisfy the diversity of data access demands of data processing across geo-distributed DCs. The related QoS-based replica selection in [38] is employed for the comparison. To validate our proposed QoS-based replica selection policy, the performance of our algorithms is compared with *JPET* and *ENS* [38]. From Fig. 9a, *QoS-RS* is about three quarters of *default-RS* in the execution time, and achieves an obvious improvement of *ENS* over *default-RS*, achieving a better result than *mr-QoS*. Furthermore, we also report the fluctuation of *ENS* for the algorithms within a certain period of the system runtime, as shown in Fig. 9b. It is obvious that *QoS-RS* has a better utilization of network than *df-RS* and *mr-QoS*, though it may have an obvious jitter.



(a) The Ratio of selection policies.



(b) *ENS* of selection polices during the runtime.

Fig. 9. The performance of the replica selection strategies. The default policy is denoted as *df-RS*, the related QoS-based policy as *mr-QoS*, and our proposed QoS-based policy as *QoS-RS*.

F. Job Complete Time Improvement Using PARMS

With the predicted information on the network traffic, *PARMS* can optimize replica placement and selection to promote the efficiency of data access and reduce job complete time. Fig. 10 plots the completion time of jobs across geo-distributed DCs. Fig. 11 plots the completion time of Task C of different data sizes across geo-distributed DCs, which shows the proposed replica management system is more capable of dealing with power big data compared with the others.

VII. Conclusion

With the rapid development of smart power, the real-time processing of power big data is becoming increasingly important. To achieve low latency processing under the conditions of limited bandwidths and a relatively

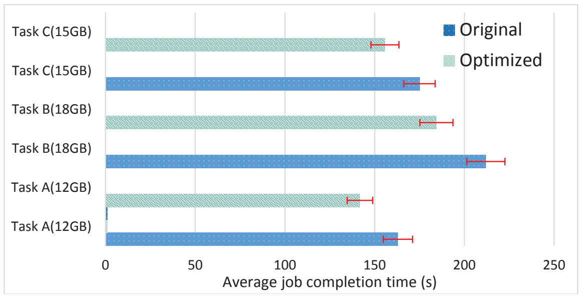


Fig. 10. The average completion times of three kinds of tasks within the system under the original setting and optimized setting of replica policies. The job completion times are reduced by 11.82% to 12.56% after conducting an optimization using *PARMS*.

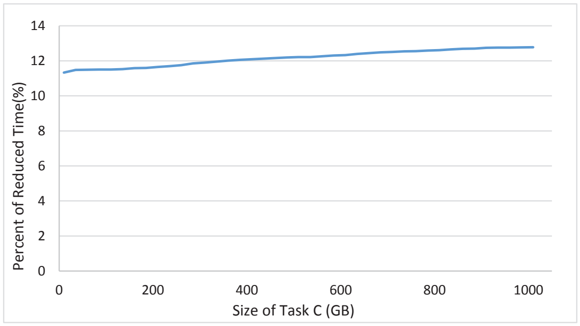


Fig. 11. The average completion times of Task C of different data size within the system under the original setting and optimized setting of replica policies. The job completion times are still reduced at a reasonable percentage when the data size of the tasks gets bigger after conducting an optimization using *PARMS*.

fixed underlying infrastructure, we designed and implemented an adaptive replica management system, *PARMS*, for geo-distributed power big data storage. Based on a near real-time SCN-based mechanism that is able to predict network traffic, efficient replica management approaches are designed to optimize the replica placement and selection. A series of experiments were conducted on the platform of an electric power corporation, *CSG*. The experiment results show that our replica management policies can solve the network transmission bottleneck to a certain degree, and increase the computing throughput of geo-distributed power big data systems. The job completion times across geo-distributed DCs are reduced by 12.19% on average when using *PARMS*.

Our future work will develop adaptive replica generation and deletion mechanisms for *PARMS*, and further integrate replica management policies with geo-distributed task scheduling.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No. 61877020), the Science and Technology Projects of Guangdong Province, China (No. 2014B010117007, 2018B010109002), and the Science and Technology Project of Guangzhou Municipality, China (No. 201904010393).

REFERENCES

[1] D. He, N. Kumar, S. Zeadally, and H. Wang, “Certificateless provable data possession scheme for cloud-based smart grid data management systems,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1232–1241, Mar. 2018.

[2] L. Jiang, L. D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, “An iot-oriented data storage framework in cloud computing platform,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1443–1451, Feb. 2014.

[3] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, “Low latency geo-distributed data analytics,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, Aug. 2015, pp. 421–434.

[4] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, “Gaia: Geo-distributed machine learning approaching LAN speeds,” in *Proceedings of 14th USENIX Symposium on Networked Systems Design and Implementation*, Mar. 2017, pp. 629–647.

[5] Z. Hu, B. Li, and J. Luo, “Time- and cost- efficient task scheduling across geo-distributed data centers,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 705–718, Mar. 2018.

[6] S. Sun, W. Yao, and X. Li, “DARS: A dynamic adaptive replica strategy under high load Cloud-P2P,” *Future Generation Computer Systems*, vol. 78, pp. 31–40, Jan. 2018.

[7] Y. Li, Y. Wang, and L. Jin, “Design of electric power data management system based on Hadoop,” in *Proceedings of 4th International Conference on Machinery, Materials and Information Technology Applications*. Atlantis Press, Jan. 2017, pp. 1090–1093.

[8] K. Jia, Y. Chen, T. Bi, Y. Lin, D. Thomas, and M. Sumner, “Historical-data-based energy management in a microgrid with a hybrid energy storage system,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2597–2605, Oct. 2017.

[9] W. Xiao, W. Bao, X. Zhu, and L. Liu, “Cost-aware big data processing across geo-distributed datacenters,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3114–3127, Nov. 2017.

[10] D. Wang and M. Li, “Stochastic Configuration Networks: Fundamentals and Algorithms,” *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3466–3479, Oct. 2017.

[11] X. D. Zhong, Yuand Huang and D. Liu, “NoSQL storage solution for massive equipment monitoring data management,” *Computer Integrated Manufacturing Systems*, vol. 12, pp. 3008–3016, Nov. 2013.

[12] J. Zhu, E. Zhuang, J. Fu, J. Baranowski, A. Ford, and J. Shen, “A framework-based approach to utility big data analytics,” *IEEE Transactions on Power Systems*, vol. 31, no. 3, pp. 2455–2462, Aug. 2016.

[13] J. Chen, N. Liu, Y. Chen, and W. Qiu, “Power big data platform based on hadoop technology,” in *Proceedings of 6th International Conference on Machinery, Materials, Environment, Biotechnology and Computer*. Atlantis Press, Jan. 2016, pp. 571–576.

[14] M. H. Yaghmaee, M. Moghaddassian, and A. Leongarcia, “Autonomous two-tier cloud-based demand side management approach with microgrid,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1109–1120, Oct. 2017.

[15] H. Wang, L. Chen, K. Chen, Z. Li, Y. Zhang, H. Guan, Z. Qi, D. Li, and Y. Geng, “FLOWPROPHET: Generic and accurate traffic prediction for data-parallel cluster computing,” in *Proceedings of IEEE 35th International Conference on Distributed Computing Systems*, Jun. 2015, pp. 349–358.

[16] A. Vulimiri, C. Curino, P. B. Godfrey, T. Jungblut, K. Karanasos, J. Padhye, and G. Varghese, “WANalytics: Geo-distributed analytics for a data intensive world,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, May 2015, pp. 1087–1092.

[17] K. Kloudas, M. Mamede, N. M. Preguica, and R. Rodrigues, “Pixida: Optimizing data parallel jobs in wide-area data analytics,” *Very Large Data Bases*, vol. 9, no. 2, pp. 72–83, Oct. 2015.

[18] Y. Liu, W. Wei, and R. Zhang, “DESRP: An efficient differential evolution algorithm for stochastic demand-oriented resource placement in heterogeneous clouds,” *Future Generation Computer Systems*, vol. 88, pp. 234–242, May 2018.

[19] K. Muralidhar and N. P. Bharathi, “An optimal replica relocation scheme for improving service availability in mobile ad hoc networks,” in *Proceedings of 2014 IEEE International Conference on*

Computational Intelligence and Computing Research. IEEE, Dec. 2014, pp. 1–4.

[20] S. Long, Y. Zhao, and W. Chen, "A three-phase energy-saving strategy for cloud storage systems," *Journal of Systems and Software*, vol. 87, pp. 38–47, Jan. 2014.

[21] X. Dai, X. Wang, and N. Liu, "Optimal scheduling of data intensive applications in cloud based video distribution services," *IEEE Transactions on Circuits & Systems for Video Technology*, vol. 27, no. 99, pp. 73–83, May 2017.

[22] A. De Salve, B. Guidi, P. Mori, L. Ricci, and V. Ambriola, "Privacy and temporal aware allocation of data in decentralized online social networks," in *Proceedings of 12th International Conference on Green, Pervasive, and Cloud Computing*. Springer, May 2017, pp. 237–251.

[23] C. Guerrero, I. Lera, B. Bermejo, and C. Juiz, "Multi-objective optimization for virtual machine allocation and replica placement in virtualized Hadoop," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2568–2581, Nov. 2018.

[24] W. Chen, I. Paik, and Z. Li, "Tology-aware optimal data placement algorithm for network traffic optimization," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2603–2617, Aug. 2016.

[25] Y. Wang, Q. Chen, C. Kang, Q. Xia, and M. Luo, "Sparse and redundant representation-based smart meter data compression and pattern extraction," *IEEE Transactions on Power Systems*, vol. 32, no. 3, pp. 2142–2151, Aug. 2017.

[26] N. Altiparmak and A. Tosun, "Multithreaded maximum flow based optimal replica selection algorithm for heterogeneous storage architectures," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1543–1557, May 2016.

[27] V. Nagarajan and M. A. M. Mohamed, "A prediction-based dynamic replication strategy for data-intensive applications," *Computers & Electrical Engineering*, vol. 57, pp. 281–293, Jan. 2017.

[28] M. C. Hung and D. L. Yang, "An efficient fuzzy c-means clustering algorithm," in *Proceedings of 2001 IEEE International Conference on Data Mining*, Nov. 2001, pp. 225–232.

[29] Y. Peng, K. Chen, G. Wang, W. Bai, Y. Zhao, H. Wang, Y. Geng, Z. Ma, and L. Gu, "Towards comprehensive traffic forecasting in cloud computing: Design and application," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2210–2222, Aug. 2016.

[30] T. P. Oliveira, J. S. Barbar, and A. S. Soares, "Computer network traffic prediction: A comparison between traditional and deep learning neural networks," *International Journal of Big Data Intelligence*, vol. 3, no. 1, pp. 28–37, Sep. 2016.

[31] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. M. Liu, "LSTM network: A deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, pp. 68–75, Mar. 2017.

[32] S. Zaman and D. Grosu, "A distributed algorithm for the replica placement problem," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1455–1468, Sep. 2011.

[33] C. Q. Huang, Y. Li, Y. Tang, and R. H. Huang, "A research on replica management of cloud storage system for educational resources," *Journal of Beijing University of Posts and Telecommunications*, vol. 2, pp. 93–97, Feb. 2013.

[34] T. T. Liu, C. Li, Q. C. Hu, and G. G. Zhang, "Multiple-replicas management in the cloud environment," *Journal of Computer Research & Development*, vol. 48, no. S3, pp. 254–260, Sep. 2011.

[35] P. Pospichal, J. Jaros, and J. Schwarz, "Parallel genetic algorithm on the CUDA architecture," in *Proceedings of European conference on the applications of evolutionary computation*, Apr. 2010, pp. 442–451.

[36] M. L Massie, B. N Chun, and D. Culler, "The Ganglia distributed monitoring system: Design, implementation and experience," *Parallel Computing*, vol. 30, pp. 817–840, Jul. 2004.

[37] Q. Huang, J. Huang, X. Wang, C. Huang, and X. Heng, "Big data based service platform and its typical applications of electric power industries," in *Proceedings of 2nd International Conference on Energy, Power and Electrical Engineering*, Nov. 2017, pp. 184–191.

[38] A. Jaradat, A. H. Muhamad Amin, M. Zakaria, and K. Golden, "An enhanced grid performance data replica selection scheme satisfying user preferences quality of service," *European Journal of Scientific Research*, vol. 73, pp. 527–538, Mar. 2012.

Changqin Huang received his Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou in 2005. Currently, he is a Professor at Zhejiang Normal University, Jinhua, China. He was a Visiting Scientist at Zhejiang University, and an Honorary Visiting Professor at La Trobe University. His research interests include service computing, big data, and semantic information retrieval. He is an awardee of "Pearl River Scholar".

Dr. Huang is a Senior Member of the China Computer Federation, and a member of ACM and IEEE. He has served as a Reviewer for several conferences and journals.

Qionghao Huang received his master degree of Software Engineering in South China Normal University in 2018. And now he is a student in School of Information Technology in Education, South China Normal University, Guangzhou, China. His main research interests include cloud computing and deep learning methods. He is a student member of CCF and IEEE.

Dianhui Wang (M'03-SM'05) was awarded a Ph.D. from Northeastern University, Shenyang, China, in 1995. From 1995 to 2001, he worked as a Postdoctoral Fellow with Nanyang Technological University, Singapore, and a Researcher with The Hong Kong Polytechnic University, Hong Kong, China. He joined La Trobe University in July 2001 and is currently a Reader and Associate Professor with the Department of Computer Science and Information Technology, La Trobe University, Australia. Since 2010, he has been a visiting Professor at The State Key Laboratory of Synthetical Automation of Process Industries, Northeastern University, China. His current research interests include industrial artificial intelligence, industrial big data oriented machine learning theory (deep stochastic configuration networks, http://deepscn.com) and its applications in process industries, intelligent sensing technology and power systems.

Dr Wang is a Senior Member of IEEE, and serving as an Associate Editor for IEEE Transactions On Neural Networks and Learning Systems, IEEE Transactions On Cybernetics, Information Sciences, and WIREs Data Mining and Knowledge Discovery.