On Achieving High Data Availability in Heterogeneous Cloud Storage Systems

Mouhamad Dieye*, Mohamed Faten Zhani†, Halima Elbiaze*
*Université du Québec À Montréal, Montreal, Quebec, Canada
†École de Technologie Supérieure, Montreal, Quebec, Canada

***Abstract*—In the era of Big data, cloud storage services have become the option of choice to store and share data thanks to their cost-effectiveness and seemingly limitless capacity. The increasing success of these services is driving cloud providers to further improve their storage management systems in order to offer more stringent guarantees on data availability and access time. However, despite recent efforts towards this goal, existing solutions have largely overlooked the heterogeneity of the workloads and the underlying storage components in terms of failure rates, capacity and I/O speed. To fill this gap, we present in this paper a heterogeneity-aware data management scheme (dubbed Heron) based on a genetic algorithm that takes into consideration disk heterogeneity to satisfy SLA requirements in terms of access time and availability and minimizes costs in terms of data migration, storage and energy consumption. Through realistic simulations, we show that Heron significantly improves data availability and access time and ensures minimal storage costs and data migration overhead compared to heterogeneity-oblivious solutions.**

***Index Terms*—Cloud Storage Systems, Data Availability, Fault tolerance, Genetic algorithms**

I. Introduction

With the unprecedented growth of large-scale Internet applications, cloud storage services have become the top platform for hosting and delivering data, providing instantaneous and easy access to seemingly limitless storage resources. As a result, Cloud Storage Providers (CSPs) are compelled to further increase their capacity and to provide more stringent guarantees on data availability and access time. Recent reports estimated the cost of critical business applications downtime between $84,000 and $500,000 per hour [1, 2], making the lack of guarantees on data availability a major barrier to a wider adoption of the cloud.

Typically, cloud providers resort to data replication as an effective technique to meet Service Level Agreement (SLA) requirements in terms of data availability and access time. Replicas are then used as a mean for data recovery in case of disk failure, thereby increasing data availability. They can also be leveraged to serve incoming data requests and alleviate the workload burden on some disk drives in order to improve the overall data access time. For CSPs, the main challenge related to data replication is to find the optimal number of replicas for each data block and to find the optimal placement of these replicas in the cluster in order to satisfy SLA requirements in terms of data availability and access time.

In recent years, a large body of work has been devoted to the design of replica management schemes [3–8]. Despite these efforts, the inherent heterogeneity within cloud storage systems have remained largely overlooked. Indeed, cloud storage systems are typically clusters of thousands of machines, containing disk drives with various characteristics in terms of capacity, I/O speed, energy consumption, model [9]. Consequently, in such heterogeneous environments, disk failures are very common with failure rates significantly varying over space and time from one disk to another and from one year to another [10, 11].

The aforementioned considerations have deep implications on the performance of replica management systems. First, overlooking the heterogeneity of failure rates and their variability over time may lead to unpredictable data availability which might severely decrease over time. The number of replicas and their placement should ideally be decided based on the availability of the hosting disks and should be adjusted dynamically taking into account the variability of disk failures over time. Second, in order to ensure that access delay requirements are satisfied, data requests should be directed to different replica locations based on their heterogeneous processing capacities. A heterogeneity-oblivious scheme may distribute requests evenly across replica locations; However, as disk drives have different I/O processing speed, some replicas may not be able to handle the requests within the targeted access delay. Finally, one must also consider management costs incurred when replicas are created, migrated or deleted. Such operations consume disk I/O, computing and bandwidth resources at both the source and destination and, of course, lead to increased energy consumption in the cluster. It is therefore necessary to minimize such costs and ensure the overall system stability.

In this paper, we aim at addressing replica management problem in a heterogeneous storage cluster while taking into account all the aforementioned considerations. We hence propose Heron[1], a HEteROgeneity-aware replica management framework that relies on a genetic algorithm. Heron takes into account the variability of workloads (i.e., access rates) and disk characteristics (i.e., model, capacity, failure rates, power consumption) over time and space in order to satisfy SLA

[1]Heron: a bird known for its long legs with more than 64 different species worldwide.

978-3-901882-89-0 @2017 Crown

requirements (in terms of data availability and access time) and minimize energy costs. To the best of our knowledge, accounting for the heterogeneous characteristics of disk drives and their variations over time has yet to be considered in the context of cloud storage management.

The rest of the paper is organized as follows. Section II reviews the related work. section III describes the architecture of the proposed storage management system. In Section IV, we mathematically formulate the replica management problem. Section V presents the proposed replica management algorithm. We finally present our simulation results in Section VI and conclude in Section VII.

II. Related Work

In this section, we briefly overview recent work on disk failure characterization and replica management in the cloud.

A. Disk failure characterization

Disk drive failures have been reported as one of the biggest cause of data unavailability and, in worst cases, permanent data loss [10, 11]. As a result, characterizing disk drive failure behaviour has been much investigated in recent years [10–13]. For instance, Schroeder et al. [11] observed increasing disk failure rates over time with annual replacement rates commonly ranging from 2 to 4%, reaching up to 13% in some systems. Moreover, they observed that the failure behaviour of disk drives vary even for disks from the same model depending on the operating conditions such as temperature, workload. Vishwanath et al. [14] reported that disk drive failures represented 78% of the total failures behind service disruption and server failures. In addition, they found that servers having experienced prior failures are more prone to fail in the near future. Pinheiro et al. reported that the annual failure rates of the disks grouped by age ranged from 1.7% to 8.6% [10]. They indicated that the values of S.M.A.R.T attributes are highly correlated with disk failures. Leveraging this observation, recent work have proposed disk prediction models based on S.M.A.R.T attributes that are able to achieve high accuracy and low false rates [12, 13]. In [12], Li et al. presented an approach based on classification and regression trees that is able to predict disk drive failures with a 95% accuracy and false alarm rate under 0.1%. Leveraging back propagation neural networks, Zhu et al. [13] proposed a prediction model with an accuracy reaching 95%.

Several previous works have also attempted to quantify data availability. Generally speaking, it is easy to devise a mathematical expression for data availability under the assumption of constant failure rate [15]. As a result, many studies have considered the steady state availability defined as the proportion of time for which the data is available after a long run [16, 17]. However, this assumption has been repeatedly contested for electronic components such as disk drives [10, 11] as it oversimplifies the availability analysis, potentially leading into false conclusions. In practice, disk drives are in use for a limited time period during which the steady state availability may not be reached. Consequently, the instantaneous availability is more critical when time-varying failure rate are considered although it is mathematically more tedious to compute [15, 18].

B. Replica management in the cloud

Replica management in cloud computing systems have been studied in many contexts, typically focusing on determining the minimal number of replicas and their placement in the system depending on various goals such as minimizing costs or space consumption, maximizing data availability and reducing data access latency. For instance, Abad et al. [6] proposed DARE, a data replication mechanism for MapReduce clusters which adapts to file popularity change. Bin et al. [19] presented DRDS, a dynamic replica management strategy which aims to reduce storage and maintenance by deleting non-performance crucial replicas while ensuring QoS requirements. Rahman et al. [20] consider a p-median model where the replica hosts are determined based on the response time. However, all of these studies overlook the underlying heterogeneity of storage components.

In [3], the authors proposed a dynamic replica strategy which triggers replication as the popularity of a data file passes a dynamic threshold. Wei et al. proposed CDRM [4], a dynamic replica management scheme implemented in HDFS which aims to satisfy availability requirements by maintaining a minimum number of replicas and dynamically placing them based on the blocking probability of storage nodes in order to distribute the workload. While both solutions take into account node heterogeneity in terms of capacity and speed, they do not consider time-varying failure rates in their availability analysis.

Different from previous work, this paper considers the heterogeneity of the nodes and their time-varying failure rates to dynamically adjust the number of replicas and their placement in order to satisfy the desired data availability and access time.

III. System Overview

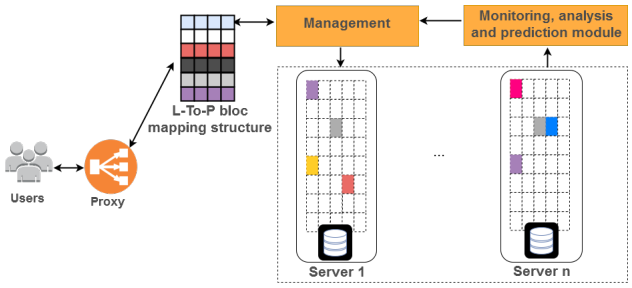

Fig. 1: HERON storage management system

In this section, we provide an overview of HERON, the storage management system we are proposing in this paper. Traditionally, storage systems aim at storing/retrieving files in/from a large pool of servers. Usually, each file is divided into multiple blocks of data, called hereafter logical blocks. The available physical disks are also divided into physical

blocks of data. Logical and physical blocks are assumed to have the same size. Each logical block is then mapped onto one physical block and its content will physically be stored there. A physical block content might be replicated in other physical blocks. The content of a logical block can hence be retrieved from any of the existing replicas.

Such a storage system requires a management framework to efficiently manage the data placement and replication to satisfy SLA requirements in heterogeneous cloud storage environment. In this context, HERON is proposed to provide a storage service with stringent guarantees on data access time and availability by taking into account the heterogeneous nature of disk drives (in terms of capacity, I/O speed, time-varying failure rate) available in data centers.

Fig. 1 shows the proposed storage management system which is built up from the following components:

• ***L-To-P block mapping structure:*** This data structure is a table which contains information about the mapping between Logical blocs and the Physical blocs (L-To-P) and the placement of the physical blocs (including block replicas) in the infrastructure. This table is maintained by the management module and will be frequently used by the Proxy module.

• ***Proxy module:*** this module serves as an interface between the storage system and the end users. Hence, it mainly receives file upload/download requests. When a file is requested, the proxy checks the L-To-P block mapping structure in order to retrieve the placement of physical blocs composing the file. The proxy fetches those blocks and send them back to to the user. Conversely, when a file is uploaded, the proxy communicates with the management module to inquire where the file blocs should be placed in the physical infrastructure. The proxy then uploads the file blocs to their corresponding physical placement in the cluster.

• ***Monitoring, analysis and prediction module:*** this module is in charge of monitoring and collecting information about the I/O requests, access latency and the state of each disk drive. This data is used afterwards to forecast future data access requests and disk failures. In this work, we leverage classification and regression trees to predict disk failures [12] in data centers as it was shown that they provide low false alarm rates [12, 13]. To forecast the request rate, we use the Auto-Regressive Integrated Moving Average (ARIMA) model as it has been shown to provide acceptable accuracy when predicting cloud workloads [9, 21, 22]. However, the proposed system can be easily extended to use other prediction models.

• ***Management module:*** this module decides of the physical blocs placement in the infrastructure. In other words, maintains the L-To-P block mapping structure. Furthermore, based on the measurements and disk failure prediction provided by the monitoring module, the manager takes appropriate measures to ensure SLA requirements in terms of data access time and data availability. To achieve this goal, the manager dynamically migrate and replicate the blocs of data over time while minimizing replica migration and creation costs as well as energy costs.

IV. Problem formulation

In this section, we provide a formulation of the heterogeneous replica management problem. Assume a discrete time system $T = \{0, 1, 2, ..., |T|\}$ with $t \in T$. At the end of each time period, replica creation or migration decisions are taken by the management module in order to guarantee SLA requirements. Given a data center consisting of a pool of disk drives $H = \{0, 1, 2..., |H|\}$, taken from a mixture of M disk models. Each model $m \in M$ is characterized by a set of specifications: capacity $c^m \in \mathbb{N}$, an active and idle power consumption represented respectively by $p^{m,act} \in \mathbb{R}^+$ and $p^{m,idle} \in \mathbb{R}^+$, and a disk model service time l^m defined as the time necessary to complete a single I/O request by a disk drive. Define a matrix $Q_{|M|\times|H|}$, where $q^{m,h} = 1$ denotes a disk h belonging to model m and 0 otherwise. We derive the following relations:

$$s^h = \sum_{m \in M} q^{m,h} \cdot l^m, \quad \forall h \in H \tag{1}$$

where s^h expresses the disk service time of a disk h. Further let $\lambda^{m,h} = \{\lambda_1^{m,h}, \lambda_2^{m,h}, ..., \lambda_{|T|}^{m,h}\}$ represent a set of time varying failure rates where $\lambda_t^{m,h}$ designates the observed failure rate at time t for a hard drive h of model m. Given that user data are partitioned into data blocks, let $B = \{0, 1, 2..., |B|\}$ denote the set of data blocks stored across disk drives where each block b has a size $\bar{b} \in \mathbb{N}^+$, a minimum availability required A_{thr}^b and maximum data access response time W_{thr}^b. Consider a matrix $R_{|B|\times|H|\times|T|} = (r_t^{b,h})$ mapping the locations at time t of each replica. $r_t^{b,h}$ is equal to 1 when a replica of b is placed into h and to 0 otherwise. We infer the following constraints:

$$\sum_{b \in B} r_t^{b,h} \cdot \bar{b} \leq \sum_{m \in M} q^{m,h} \cdot c^m \quad \forall h \in H, \ \forall t \in T \tag{2}$$

By holding constraint (2), we ensure enough space is available prior to assigning a replica to a location and that multiple replicas are not lost through a single disk failure.

A. Modeling the availability constraint

Disk drives are the most replaced component and leading reason behind server downtime in a data center [10, 11], we therefore simplify our model by assuming the availability experienced by a block b to be identical to that of the disk on which it is stored. We thus model the time-variation of disk drives failure rate using a stair-step approximation algorithm [15] based on the observation that the time-varying property for the failure rate is mainly exhibited at a large time scale such as a year or month. Given that disk failures are proactively predicted in HERON, the repair time for each disk drive is reasonably assumed to be constant. Therefore,

the instantaneous availability $A_t^{m,h}$ of a single disk drive h is obtained through [15] :

$$A_t^{m,h} = \begin{cases} \text{if } t \in [0,1] \\ \frac{\mu}{\lambda_0^{m,h}+\mu} + \frac{\lambda_0^{m,h}}{\lambda_0^{m,h}+\mu} e^{-(\lambda_0^{m,h}+\mu)t} \\ \\ \text{if } t, i \in [2,3,...,|T|] \\ \frac{\mu}{\lambda_i^{m,h}+\mu} + [A_t^{m,h} - \frac{\mu}{\lambda_i^{m,h}+\mu} e^{-(\lambda_i^{m,h}+\mu)(i-t)}] \end{cases} \tag{3}$$

with $\lambda_t^{m,h}$ and μ, respectively denoting the failure rate of disk h of model m and the repair rate at time $t \in T$ and $i \in T$, the next time slot to t. Keeping in mind that a data block is available when at least one replica is accessible, the instantaneous availability A_t^b of a data block b at time t is then given by:

$$A_t^b = 1 - \prod_{h \in H} (1 - (A_t^{m,h} \cdot r_t^{h,b})), \quad \forall b, \forall h, \forall t \tag{4}$$

Finally, we enforce the availability requirement for b through the constraint below:

$$A_t^b \geq A_{thr}^b, \quad \forall b, \forall t \tag{5}$$

B. Modeling the processing delay constraint and demand assignment

The heterogeneity of disk drives in a data center in terms of I/O speed naturally leads to discrepancies with regard to the processing delay. Thus, define D_t^b as the access demand at time t for a data block b expressed in terms of I/O requests per second (IOPS). The demand is then split amongst replicas with a processing delay constraint W_{thr}^b. Using a M/M/1/K queue, we model a single disk drive where I/O requests are held until they are serviced with the queuing delay w_t^h computed by:

$$w_t^h = \rho_t^h d_t^{b,h} \left(\frac{1}{1-\rho_t^h} - \frac{(k+1)(\rho_t^h)^{k+1}}{1-(\rho_t^h)^{k+1}} \right) \leq W_{thr}^b \tag{6}$$

with ρ_t^h and $d_t^{b,h}$ respectively denoting the utilization and arrival rate of I/O requests at disk h. From equation (6), it is easy to derive, for each replica location, the following relation:

$$(d_t^{b,h})^{-1} \geq \frac{\rho_t^h}{W_{thr}^b} \left(\frac{1}{1-\rho_t^h} - \frac{(k+1) \cdot (\rho_t^h)^{k+1}}{1-(\rho_t^h)^{k+1}} \right) \tag{7}$$

with the following constraint to ensure all data access demands are treated:

$$D_t^b = \sum_{h \in H} r_t^{hb} \cdot d_t^{b,h} \quad \forall t, \forall b \tag{8}$$

C. Modeling the management costs

In this section, we focus on expressing the costs incurred during the replica management process. We translate each replica operation (i.e, replica creation, migration, deletion, access) in terms of monetary costs. Assume that each disk model m has a storage space cost κ^m expressed in terms of \$/MB. We then derive the storage space cost κ_t^b for each block b as:

$$\kappa_t^b = \sum_{m \in M} \sum_{h \in H} q^{m,h} \cdot \kappa^m \cdot r_t^{b,h} \cdot \bar{b} \quad \forall b, \forall t \tag{9}$$

with $\bar{b}$, the size of data block b.
Thus the cost of replica creation can be deduced by:

$$z_t^b = \kappa_{t-1}^b - \kappa_t^b \quad \forall b, \forall t \tag{10}$$

Let us consider a binary variable $\gamma_t^{b,h} = \{0,1\}$ indicating whether a replica operation concerning a data block b is carried out on a disk h at time t. Inversely, let $\delta_t^h = 0,1$ denote whether the disk h is idle at time t. Each replica operation then induces an energy consumption cost E_t^b [23] :

$$E_t^b = (E_t^{b,act} + E_t^{b,idle}) \cdot \phi \quad \forall b, \forall t \tag{11}$$

where ϕ represents the power cost for a data center expressed in \$/kWh, $E_t^{b,act}$ the active power consumption (i.e, read or write operations) and $E_t^{b,act}$ the idle power consumption for a block b respectively defined by:

$$E_t^{b,act} = \sum_{m \in M} \sum_{h \in H} q^{m,h} \cdot p^{m,act} \cdot s^h \cdot \gamma_t^{b,h} \cdot \bar{b} \quad \forall b, \forall t \tag{12}$$

$$E_t^{b,idle} = \sum_{m \in M} \sum_{h \in H} q^{m,h} \cdot p^{m,idle} \cdot \delta_t^h \quad \forall b, \forall t \tag{13}$$

with $p^{m,act}$ and $p^{m,idle}$, respectively denote the power consumption for a disk model m when active and idle.

We further define a penalty ν_t^b to be paid by the CSP whenever unavailability occurs during replica migration process. To this effect, let $y_{b,t}^{h,u}$ denote the total downtime of a block b migrated from disks h to u at time t. If we consider a monetary penalty β expressed in \$/sec., we obtain the following relation:

$$\nu_t^b = y_{b,t}^{h,u} \cdot \beta \tag{14}$$

Combining 10, 11 and 14, we express the total replica management cost below by:

$$\mathcal{C}_t^b = z_t^b + E_t^b + \nu_t^b \quad \forall b, \forall t \tag{15}$$

Finally, to minimize replica synchronization management overhead, HERON must ensure that the chosen replica allocation is solid enough to endure the fluctuation of demand and disk drive failure rate based on the values forecasted by the monitoring, analysis and prediction module. However, we must also carefully calibrate the scope of predictions to avoid unnecessary costs due to over-provisioning. Thus, let $O = \{t, t+1, ..., t+|O|\}$, denote the time scope of the forecasts considered, we are then interested in allocating replication schemes that globally minimizes the aforementioned costs while ensuring the SLA requirements. Therefore, the overall objective of HERON is to minimize the amortized costs through the objective function below while respecting constraints (5), (7) and (8):

$$min \frac{1}{|O|} \sum_{t=0}^{T} \mathcal{C}_t^b \quad \forall b \in B \tag{16}$$

The problem described previously is NP-hard to solve as it generalizes the bin-packing problem with variable bin sizes and prices. Hence, we describe below our proposed heuristic.

V. Heuristic

In this section, we present the design of the Heron algorithm which aims to tackle the heterogeneous replica management problem. The Heron algorithm is based on Genetic Algorithms (GA) which are heuristic search algorithms built around the concept of population genetics, in which the process of natural evolution compels each individual to increase its fitness in order to survive throughout future generations. A key advantage of GAs is that they allow the use of historical information to efficiently direct the search towards better regions within a large search space allowing to yield near optimal solutions [24]. In Heron, we leverage the combined usage of workload and disk failure data obtained through data collection and prediction mechanisms along with an efficient characterization of the storage cluster heterogeneity in order to satisfy SLA requirements while minimizing reconfiguration costs.

A. Genetic algorithm encoding

Typically, each potential solution in GA is considered as an individual and represented by a chromosome. Each chromosome consists of genes which encode specific characteristics of the individual. In Heron, a chromosome is considered to be equivalent to a candidate replica allocation scheme with genes corresponding to a set of heterogeneous disk drives. In our work, we represent chromosomes using value encoding where each gene is identified by a unique disk identification, therefore allowing the retrieval of its heterogeneous parameters (e.g., model, active power consumption). Our goal is to find the optimal data block replica allocation scheme such that reconfigurations are minimized and system stability is ensured.

B. Search space reduction

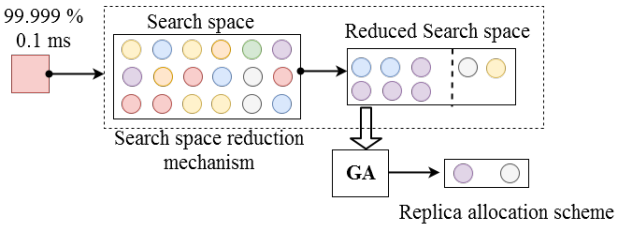

Fig. 2: Search space reduction

In order to increase Heron's efficiency and robustness, we introduce a search space reduction mechanism such that the replica location search is focused on promising areas in accordance with the SLA requirements. To this end, the first step is to evaluate the level of heterogeneity within the storage cluster with the objective of identifying which disk parameters should be emphasized during replica locations search. We hence define the premium set of disks as the the set of disks with low failure rates. In Heron, we use the k-means clustering [25] applied to the disk model failure rates to divide disks into 3 sets. The set with the lowest failure rates is considered as a the premium set of disks.

We consider in our study five main disk parameters for the solution search space mechanism:

- the disk active power consumption $p^{m,act}$ to indicate potential energy consumption, particularly when the disk hosts heavily requested data blocks.
- the standard deviation σ^h of the disk model's typical failure rates as an indication of the fluctuation over time.
- the average of the disk failure rates $\overline{\lambda}^{m,h}$
- the disk service time as an indication of the its processing speed s^h
- the mean disk utilization π^h

Henceforth, depending on the proportion of the premium disks, the replica locations search will focus on disk parameters which tend to be harder to find within the storage cluster. For instance, in a storage cluster consisting of a large majority of premium disk drives, we believe it is wiser that the replica location search to be geared more towards optimizing the required number of replicas to satisfy access delay requirements. Therefore, we define the notions of primary set and secondary set of disk parameters that refer to disk parameters which are more or less critical towards satisfying the requirements of the concerned data block. Obviously, the disk parameters retained in each set depends on the severity of the data block's SLA requirements. The goal is to associate a profile to regions within the search space in order to reduce the dimension of the search space. This profile is defined based on the SLA requirements and the proportion of premium disks using machine learning techniques. In Heron, we use decision trees to determine primary and secondary disk parameters. For instance, in a storage cluster environment with a large set of premium disks and for data blocks with high availability requirements (e.g., above 99.99%), we consider as primary parameters the standard deviation σ^h and disk service time s^h while active disk power consumption is regarded as a secondary parameter.

Once the primary and secondary set of disk parameters are configured, we further define a reduced search space as shown in Fig. 2 which contains a main group and a diversity group which respectively cluster disks with regard to the primary and secondary parameters. The dimensions of each group vary in accordance with the severity of the data block requirements to avoid resource starvation for data blocks with higher SLA requirements. For each group, we define a reference disk with regard to the concerned parameters. We then apply a similarity score function defined below in order to bundle similar disks.

$$sim(h_i, h_j) = \frac{1}{1 + \sqrt{\sum_{k \in P} (h_i^k - h_j^k)^2}} \tag{17}$$

where $k \in P$ represents the concerned parameters and h_i the reference disk considered. Disks with the highest score

Algorithm 1 Search space reduction mechanism

Input: Storage cluster $H = \{1, 2, ..., |H|\}$, A^b_{thr}, $D^b = \{D^b_{t-i}, ..., D^b_{t-1}, D^b_t, D^b_{t+1}, ..., D^b_{t+o}\}$, decision tree P
Output: Reduced search space $I = \{1, 2, ..., |I|\}$ with $|I| < |H|$

$f^{b,thr}_t \leftarrow k - means(H, 3)$
$H^{'} = \{\forall h \in H \mid \lambda^{m,h}_t \leq f^{b,thr}_t\}$
if SLA violation predicted **then**
  $t^b_{dec} \leftarrow min(t_f, t_w)$
  // primary and secondary parameters : $p^{'}_b,\ p^{''}_b$
  $p^{'}_b,\ p^{''}_b \leftarrow \emptyset$
  $(p_b,\ p_b^{''}) \leftarrow$walk the decision tree P
  // main group, diversity group : $g^{'}_b,\ g^{''}_b$
  $g^{'}_b,\ g^{''}_b \leftarrow \emptyset$
  // reference disk main and diversity group: $r^{'}_b,\ r^{'}_b$
  $g^{'}_{temp} \leftarrow H,\ g^{''}_{temp} \leftarrow H$
  for $\forall g \in g^{'}_{temp}$ [alternatively $\forall g^{'} \in g^{''}_{temp}$] **do**
    Compute eq. (17) with $h_i = r^{'}_b$ and $h_j = g$ [alternatively $h_i = r^{''}_b$ and $h_j = g^{'}$]
  end for
  sort $g^{'}_{temp}$, $g^{''}_{temp}$
  // max. dimension main and diversity group : $dim(g^{'}_b),\ dim(g^{''}_b)$
  $i, j \leftarrow 0$
  while $i \leq dim(g^{'}_{temp})$ [alternatively $dim(g^{''}_{temp})$] **do**
    Take next element k in $g^{'}_{temp}$ and put in $g^{'}_b$ [alternatively $g^{''}_{temp}$ and put in $g^{''}_b$]
  end while
  $I \leftarrow g^{'}_b \cup g^{''}_b$
end if
return I

are selected in the group. If there are disks with similar similarity score, the mean disk utilization ϕ^h is used to separate them. We summarize the search space reduction mechanism in Algorithm 1.

C. Design of the Heron Algorithm

Algorithm 2 summarizes the proposed heuristic. Heron initially generates potential solutions, hereby called candidate allocation scheme, through random selection of disk drives obtained from the population obtained in the reduced search space such that constraints (2, 5, 7 and 8) are satisfied for each candidate replica allocation scheme. The complexity to generate a potential solution is $\mathcal{O}(n \cdot m)$ where n and m respectively denote the size of the reduced search space and the number of disks in the candidate allocation scheme.

The second step is the crossover of the replica allocation schemes in which we generate promising solutions through the combination of existing solutions. Our adopted strategy consists of retaining the previous replica allocation scheme as one parent and mate successively with existing candidate allocation schemes in order to create offspring solutions as illustrated in Fig. 3. Given that disk drives are

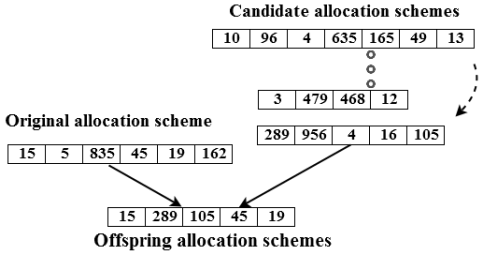

Fig. 3: Replica allocation scheme crossover

heterogeneous, the size of candidate allocation scheme may differ. In Heron, we use a uniform crossover [26] given that it allows information to be taken from one parent while inheriting the best information from the other parent. The goal is to make sure that, from each replica allocation scheme to another, there is still similarity to the original allocation in order to ensure system stability and avoid excessive operations over time. The complexity to produce an offspring is $\mathcal{O}(n)$. Note that offspring allocation schemes may not necessarily be a valid scheme.

The next step of the evolution process is mutation whose role is to introduce randomness, and thereby allow efficient exploration of new regions in the search space to avoid local optima. In Heron, the mutation process targets particularly invalid offspring allocation schemes in an attempt to transform them into valid ones. Offspring solutions are evaluated using equation (16) and sorted based on their scores. To avoid local optima, we temporarily place disks which regularly appear in low ranked offspring schemes into a blacklist in order to avoid them in the following solutions. Unfeasible offsprings are deleted while the highest ranking offsprings are taken into the next iteration of crossover and mutation.

The iterations are stopped when the decision time is reached. The offspring candidate scheme with the highest rank is then selected as the final solution.

VI. Experimental evaluation

A. Simulation setup

Heron is implemented using the CloudSim toolkit simulator [27]. All files in the cluster are mapped into fixed-size data blocks which are then assigned SLA requirements consisting of a minimal availability and a maximum access delay requirement respectively taken randomly from $[99.0, 99.9, 99.99, 99.999, 99.9999]$ and $[0.1, 0.25, 0.5, 1]$. Initially, the number of data blocks in the cluster is 100000. We model the file access patterns and size according to file access traces provided by Yahoo! [28] using the stratified sampling technique [29]. Workload fluctuation and application diversity are ensured by distributing the access requests arrival times using a mixture of exponential, pareto and weibull distributions.

We set the prediction horizon of the monitoring, analysis and prediction module to two times periods corresponding to around 6 days. We simulate a heterogeneous storage cluster

Algorithm 2 Heron algorithm

Input: Reduced search space $I = \{1, 2, ..., |I|\}$, Original allocation scheme S^b_{ori}, t^b_{dec}
Output: Alternative allocation scheme S^b_{alt}

$t_{cur} \leftarrow$ current time
// schemes inherited from previous generations : $P^b_{eli} \leftarrow \emptyset$
// max. dimension of $P^b_{eli} : |P^b_{eli}|$
while $t_{cur} \leq t^b_{dec}$ **do**
// Initial population: $P^b \leftarrow \emptyset$
// max dimension of $P^b : |P^b|$
Selection
  $j \leftarrow 0$
  $D^b_{aj} = max(\{D^b_{t+1}, D^b_{t+2}, ..., D^b_{t+o}\})$
  // Candidate allocation scheme: S^b_{can}
  $S^b_{can} \leftarrow \emptyset$
  while $j \leq |P^b|$ **do**
    while constraints (5), (7) and (8) not respected with $D^b_t = D^b_{aj}$ **do**
      $disqueTemp \leftarrow rand(i \mid i \in I)$
      if constraints (2) respected **then**
        $S^b_{can} \leftarrow S^b_{can} \cup disqueTemp$
      end if
    end while
    $P^b \leftarrow P^b \cup S^b_{can}$
    $S^b_{can} \leftarrow \emptyset$
    $j \leftarrow j + 1$
  end while
Crossover
  // Set of offspring schemes : $P^b_{off} \leftarrow \emptyset$
  for $p^b \in P^b$ **do**
  // Offspring allocation scheme : S^b_{enf}, S^b_{enf1}
    $(S^b_{off}, S^b_{off1}) \leftarrow$ uniformCrossover(S^b_{ori}, p^b)
    $P^b_{off} \leftarrow P^b_{off} \cup (S^b_{off}, S^b_{off1})$
  end for
  $P^b_{off} \leftarrow P^b_{off} \cup P^b_{eli}$
Mutation
  // Set of invalid offspring schemes : P^b_{inv}
  $P^b_{inv} \leftarrow P^b_{inv} \cup \{p^b \in P^b_{off} \mid p^b$ invalid $\}$
  // mutation probability : m
  According to m, modify $p' \in P^b_{inv}$
  Compute evaluation function in (16) for $\{p^b \in P^b_{off}\}$ and sort P^b_{off} $k \leftarrow \emptyset$
  while $k < |P^b_{eli}|$ **do**
    Take next element in sorted P^b_{off}
    $k \leftarrow k + 1$
  end while
  $t_{cur} \leftarrow$ current time
end while
$S^b \leftarrow min[\ eval(p^b)\ \forall p^b \in P^b_{off}]$
return S^b

consisting of 5000 disk drives from several manufacturers and models. Table I shows relevant information for each disk model as reported in [30]. The failure rate threshold used in the search space mechanism is set to 2.37% corresponding to the lowest centroid value of the k-means clustering ($k = 3$) applied on Table I. For realistic purposes, we set the annual failure rates of each disk in the cluster to be in a $\pm 20\%$ range of it's model estimated failure rate. In order to study the impact of heterogeneity, we consider in our simulations three different scenarios as shown in Table II. From a scenario to another, we reduce the proportion of premium disk models and increase the number of disks with high failure rates.

TABLE I: Disk models

Hd model #	Cap. TB	Act. Pow. (W)	Failure rate (%)		
			Year 1	Year 2	Year 3
WD10EADS	1	5.4	4.29	3.9	9.91
WD10EACS	1	7.5	0.01	5.21	0.01
ST31500541AS	1.5	5.8	10.52	9.52	12.07
ST31500341AS	1.5	11.6	23.29	23.53	26.29
HDS722020ALA330	2	9.4	1.03	1.07	2.81
DT01ACA300	3	6.4	6.93	3.68	2.8
WD30EFRX	3	4.4	3.79	6.94	8.79
ST33000651AS	3	9.3	6.91	4.8	3.55
HDS723030ALA640	3	8	10.35	43.08	30.94
HDS723030ALA640	3	8.6	1.01	2.27	2.12
HDS5C3030ALA630	3	6.4	0.99	0.59	1.31
HMS5C4040ALE640	4	6.2	3.85	1.41	0.7
HDS5C4040ALE630	4	6	1.65	0.91	0.86
ST4000DX000	4	7.5	1.12	1.12	3.73
ST4000DM000	4	7.5	4.17	2.58	3.31

In our simulations, each disk is prone to fail over the year in accordance with its respective annual failure rate. Prior to the failure of a disk, it is detected and a notification is sent by the monitoring, analysis and prediction module 3 days in advance. This is a reasonable assumption as 96% prediction accuracy can be maintained with a 352 hours advance notice according to [12]. The management is then tasked to find an alternative replica allocation scheme for the data block such that its SLA requirements are maintained.

For comparison purposes, we have evaluated the effectiveness of Heron using a heterogeneity-oblivious 3-static and greedy replication scheme implementation.

- In the 3-static algorithm, we created for each data block three replicas which we then randomly place on a given disk. To maintain fairness between schemes, we ensure that two data block replicas are not placed at the same location.
- The greedy algorithm first selects a random group of disks which initially ensure the satisfaction of SLA requirements. Over each iteration, members of the replica allocation scheme are progressively replaced by selecting disks which provide the best trade-off between the mean failure rate of the replica allocation scheme and its mean service time. The number of iterations for the greedy algorithm is set to 50. Once an alternative allocation scheme is found, we proceed to first migrate replicas from the previous replica allocation scheme and create/delete replicas if needed.

B. Results

Fig. 4 shows the availability violation ratio which represents the percentage of requests that were not accommodated.

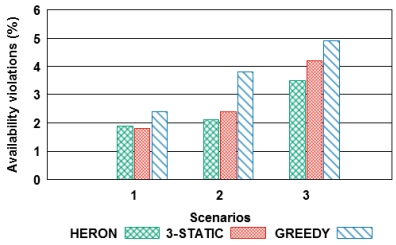

Fig. 4: Availability violation

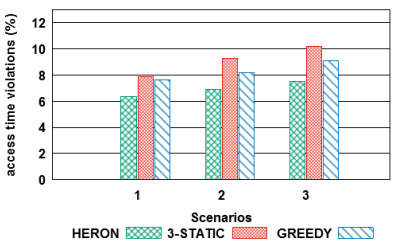

Fig. 5: Access time violation

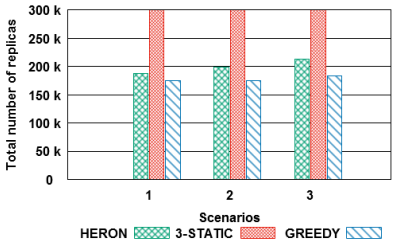

Fig. 6: Total number of replicas

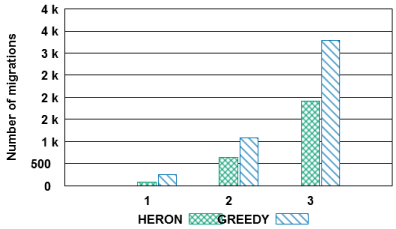

Fig. 7: Total number of migrations

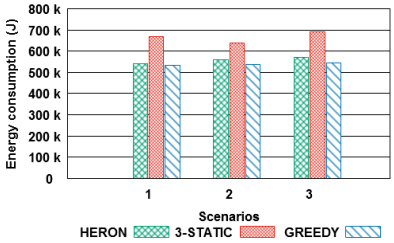

Fig. 8: Total energy consumption

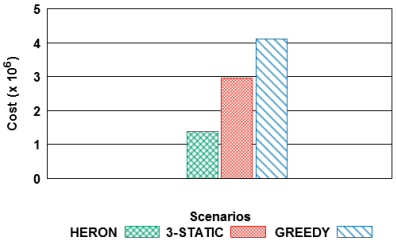

Fig. 9: Availability violation cost

TABLE II: Scenarios

Scenario 1

Hd model #	# of disks
WD10EADS	125
ST31500341AS	130
ST31500541AS	370
HDS722020ALA330	1500
ST33000651AS	375
HDS5C3030ALA630	2500

Scenario 2

Hd model #	# of disks
ST31500541AS	380
ST3000DM001	120
WD30EFRX	1500
HMS5C4040ALE640	2100
ST4000DM000	580
ST4000DX000	1400

Scenario 3

Hd model #	# of disks
ST31500341AS	630
HDS723030ALA640	1220
ST3000DM001	370
DT01ACA300	150
WD30EFRX	850
HMS5C4040ALE640	1780

An availability violation occurs when all replicas of the requested data are not available due to disk failures. It is clear from the figure that Heron induces less violations than the 3-static and the greedy algorithm. Indeed, Heron leverages the predicted failure rates to better distribute replicas among disks based on the SLA requirements and thereby ensure that there are always replicas that are available.

It can be seen also in Fig. 5 that Heron provides less data access time violations than the other two schemes. Here we are considering requests that were accomodated but for which data access time requirement was not satisfied. As Heron estimates the number of replicas based on workload prediction, it is able to deal with temporary workload fluctuation and to increase the number of replicas whenever necessary in order to meet the requested access time. This comes with the cost of a higher number of replicas in the system as shown in Fig. 6.

Fig. 8 shows also that the energy consumption of Heron is high compared to the greedy algorithm. However, this is normal because Heron dynamiclly adapts the number of replicas to the storage characteristics. It is also worth noting that the 3-static replication mechanism provided relatively good results in terms of availability and data access time requirements, particularly in simulation scenarios with a low number of disks with high failure rates. However, this approach comes at the expense of unnecessary replica creations and substantial energy consumption compared to Heron (Fig. 6 and Fig. 8).

In Fig. 7, we observe that Heron reduces the overall number of migrations. This is because it selects disks with failure rates that are low and stable over time to place the replicas. This allows to avoid subsequent migrations. In addition, Heron, through its mutation operator, ensures that from a replica allocation scheme to another, minimal migration is done by maintaining the same block locations when possible.

To further show the benefits of Heron compared to the other solutions, we considered a penalty model where the CSP pays a penalty that is proptional to the violation of the data availability requirement. Fig. 9 shows that the penalty (i.e., violation cost) incurred by Heron is less than those incurred by the two other schemes. As Heron reserves premium disks to data blocks with stringent requirements, the ratio of violations for these block is small compared to other schemes, and hence penalties are reduced.

VII. Conclusion

Despite recent efforts on data replica management in the cloud storage systems, existing solutions have largely overlooked the heterogeneity of the underlying storage components in terms of failure, capacity and I/O speed. In this paper, we proposed Heron, a heterogeneous-aware data management scheme based on a genetic algorithm that takes into account time-varying disk failure rates and heterogeneous disk capacities to satisfy SLA requirements while ensuring minimal storage and energy consumption compared to heterogeneous-oblivious solutions.

REFERENCES

[1] A. Sombers, "Let's get an availability benchmark," 2007. [Online]. Available: http://www.availabilitydigest.com/private/0206/benchmarking.pdf

[2] A. Arnold, "Assessing the financial impact of downtime - luminet," Apr 2010. [Online]. Available: http://luminet.co.uk/wp-content/uploads/2014/10/assessing-the-financial-impact-of-downtime-uk.pdf

[3] D.-W. Sun, G.-R. Chang, S. Gao, L.-Z. Jin, and X.-W. Wang, "Modeling a dynamic data replication strategy to increase system availability in cloud computing environments," *Journal of computer science and technology*, 2012.

[4] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, "Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster," in *IEEE Int. Conference on Cluster Computing (CLUSTER)*, 2010.

[5] J.-W. Lin, C.-H. Chen, and J. Chang, "Qos-aware data replication for data-intensive applications in cloud computing systems," *IEEE Transactions on Cloud Computing*, 2013.

[6] C. Abad, Y. Lu, and R. Campbell, "Dare: Adaptive data replication for efficient cluster scheduling," in *IEEE Int. Conference on Cluster Computing*, 2011.

[7] "Openstack storage object," http://swift.openstack.org/.

[8] "Hadoop distributed file system," https://hadoop.apache.org.

[9] Q. Zhang, M. Zhani, R. Boutaba, and J. Hellerstein, "Harmony: Dynamic heterogeneity-aware resource provisioning in the cloud," in *IEEE Int. Conference on Distributed Computing Systems*, 2013.

[10] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *USENIX Conference on File and Storage Technologies*, 2007.

[11] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an mttf of 1,000,000 hours mean to you?" in *USENIX Conference on File and Storage Technologies*, 2007.

[12] J. Li, X. Ji, Y. Jia, B. Zhu, G. Wang, Z. Li, and X. Liu, "Hard drive failure prediction using classification and regression trees," in *IEEE/IFIP Int. Conference on Dependable Systems and Networks*, 2014.

[13] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma, "Proactive drive failure prediction for large scale storage systems," in *IEEE Symposium Mass Storage Systems and Technologies*, May 2013.

[14] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *ACM symposium on Cloud computing*, 2010.

[15] H. Sun and J. Han, "Instantaneous availability and interval availability for systems with time-varying failure rate: stair-step approximation," in *Int. Symposium on Dependable Computing*, 2001.

[16] B. Sericola, "Interval-availability distribution of 2-state systems with exponential failures and phase-type repairs," *IEEE Transactions on Reliability*, 1994.

[17] J. Mi, "Limiting availability of system with non-identical lifetime distributions and non-identical repair time distributions," *Statistics & Probability Letters*, 2006.

[18] T. Hassett, D. Dietrich, and F. Szidarovszky, "Time-varying failure rates in the availability and reliability analysis of repairable systems," *IEEE Transactions on Reliability*, 1995.

[19] L. Bin, Y. Jiong, S. Hua, and N. Mei, "A qos-aware dynamic data replica deletion strategy for distributed storage systems under cloud computing environments," in *Int. Conference on Cloud and Green Computing*, 2012.

[20] R. Rahman, K. Barker, and R. Alhajj, "Replica placement design with static optimality and dynamic maintainability," in *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE Int. Symposium on*, vol. 1, May 2006.

[21] M. F. Zhani, H. Elbiaze, and F. Kamoun, "Analysis and prediction of real network traffic," *Journal of networks*, vol. 4, no. 9, pp. 855–865, 2009.

[22] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in *Proceedings of the 9th International Conference on Autonomic Computing*, ser. ICAC '12. New York, NY, USA: ACM, 2012, pp. 145–154. [Online]. Available: http://doi.acm.org/10.1145/2371536.2371562

[23] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang, "Modeling hard-disk power consumption," in *Proceedings of the 2Nd USENIX Conference on File and Storage Technologies*, ser. FAST '03. Berkeley, CA, USA: USENIX Association, 2003, pp. 217–230. [Online]. Available: http://dl.acm.org/citation.cfm?id=1090694.1090722

[24] Z. Ye, X. Zhou, and A. Bouguettaya, *Int. Conf. Database Systems for Advanced Applications*, 2011, ch. Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing.

[25] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[26] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 2–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=645512.657265

[27] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, 2011.

[28] "S2 - yahoo! statistical information regarding files and access pattern to files in one of yahoo's clusters," http://webscope.sandbox.yahoo.com/catalog.php?datatype=s, accessed: 2016-03-24.

[29] S. Thompson, *Sampling*, ser. CourseSmart. Wiley, 2012. [Online]. Available: https://books.google.ca/books?id=-sFtXLIdDiIC

[30] "Hard drive data sets," https://www.backblaze.com/hard-drive-test-data.html, accessed: 2016-03-24.