Cluster Comput (2016) 19:1801–1818
DOI 10.1007/s10586-016-0637-y

Adaptive measurement method for data popularity in distributed systems

C. Hamdeni[1] · T. Hamrouni[1] · F. Ben Charrada[1]

Received: 25 April 2016 / Revised: 17 August 2016 / Accepted: 6 September 2016 / Published online: 17 September 2016

Abstract Distributed systems provide geographically distributed resources for large-scale applications while managing large volumes of data. In this context, replication of data in several sites of the system is an effective solution for achieving interesting performances. A number of data replication strategies have been proposed in the literature. Data popularity is one of the most important parameters taken into consideration by these strategies. It analyzes the historic of the data access pattern, and provides predictions for future data requests. However, measuring data popularity is a challenging task because there are several factors that contribute to the evaluation of data popularity. In this paper, a new adaptive measurement for data popularity in distributed systems is proposed. The proposed measurement covers all factors taken into consideration by previous work of the literature. It also takes into consideration new factors to deal with the dynamic nature of the system so it can adapt to any access pattern. We show that the exploitation of our measurement improves the performances of replication strategies, while offering the possibility to use the data popularity parameter in new contexts in replication management.

Keywords Distributed system · Replication strategy · Data popularity · Access pattern · Temporal locality

✉ T. Hamrouni
tarek.hamrouni@fst.rnu.tn

C. Hamdeni
hamdeni.chamseddine@gmail.com

F. Ben Charrada
f.charrada@gnet.tn

[1] Computer Science Department, Faculty of Sciences of Tunis, Tunis El Manar University, University Campus, Tunis, Tunisia

1 Introduction

1.1 Background and motivations

The huge increase in data storage and computing requirements has led to Big Data, for which several distributed systems are being designed and implemented. Distributed systems handle extremely large volumes of data while requiring fast processing time with minimal possible cost. They represent an efficient solution to deal with these related challenges by offering great promise to programmers interested in developing applications that serve large volumes of data management.

Data replication, a well-known technique in distributed systems, is a practical and effective approach to face these challenges [2]. It consists in storing multiple copies of the same data at multiple sites. If one of the sites is not accessible then the data can be accessed from a different site [23]. This technique has been widely used to reduce data access time and network traffic, while increasing data availability, accessibility, and fault tolerance. It is becoming a popular approach in many distributed systems [13] such as Data Grid systems [12,16,31], Cloud systems [27,32,55], P2P systems [28,38,47], and CDN systems [26,37].

Replication in distributed systems has several problems to solve [15], like when to do replication? Where to place a new replica? How to maintain data integrity and consistency? How to reduce job execution time, job scheduling time, access latency, resource consumption and maintenance overhead? to quote but a few. Many replication strategies have then been proposed trying to answer these questions optimally. These strategies use various parameters to make the decision that deems appropriate for them. Such parameters include data popularity, resources consumption, response time, resources availability, latency, workload, energy consumption, security, storage capacities, number of replicas, etc [32]. They differ from one strategy to another according to the objectives of each strategy.

Data popularity is one of the most common parameters taken into consideration by these strategies [8,18,45,51,54]. It consists in measuring how much a given piece of data is requested by the system sites. This constitutes key information since it gives an indication of the importance of this data, allowing as a consequence a more intelligent data placement and a large optimization in the storage utilization. That is why much research in distributed systems mainly focuses on data popularity [4,7,14,24,41].

In this respect, *temporal locality* represents an important notion that must be taken into consideration when assessing data popularity in distributed systems [21]. It consists in considering that recently requested data are likely to be requested again in the near future [1]. That is why the parameter *data popularity*, in all its manifestations in distributed systems, dominates the other parameters in replication strategies. It allows indeed to obtain an indication of the probability of requesting data again.

It is worth noting that data in distributed systems may be a set of files, a file or a file part. It may also be a database, a database table or an object of a database table. All these possibilities will be covered by using hereafter the term *dataset*. Moreover, in our case, the definition of popularity is based on the popularity of certain datasets among sites.

1.2 Contributions

In this paper, a new method to measure dataset popularity is proposed. This method offers several advantages over existing methods. They are as follows:

- It avoids all the drawbacks experienced by existing measurements and provides more accurate prediction for the next dataset popularity. Indeed, the experiments show that the new measurement can reach up to 38 % improvement even comparing it with the best result among existing measurements. Obtained results also prove that almost 50 % of the performed replications are more effective due to deploying the proposed strategy.
- It is a generalization of the other popularity measurements of the literature with more considerations taken into account.
- It can be instantiated according to the application requirements and offers the opportunity to control the tradeoff between the calculation cost and the result accuracy.
- It can adapt the dynamic nature of distributed systems and can handle with any access pattern, even when the access pattern does not support the temporal locality.
- It allows the usage of the popularity parameter in new contexts that were never used before.

1.3 Paper organization

This paper is organized as follows: In Sect. 2, we analyze previous works, identify the factors considered by the existing popularity measurements and show the drawbacks experienced by each one. In Sect. 3, we propose a new measurement method to assess dataset popularity. In Sect. 4, we highlight usages of our measurement in variety of contexts. In Sect. 5, we discuss the obtained experimental results. The last section summarizes our contributions and depicts future work.

2 Analysis of previous works

2.1 Importance of data popularity parameter in replication management

To highlight the importance of the data popularity parameter in replication management, some simulations are performed using the OptorSim simulator [5,9] applied on the CMS testbed configuration [10]. 6000 requests for various datasets are analyzed. These requests are generated by jobs executions during the simulation and captured randomly from some actual CMS runs. After doing statistics on the generated requests, we observe that more than the half of the requests (55 % of the total requests) is interested in only a small portion of the datasets (more precisely, 10 %). Accordingly, any action that will happen to these 10 % datasets will directly affect jobs execution. Therefore, the manner of managing the most popular datasets influences significantly the performance of replication strategies. This highlights the importance of promoting the most popular datasets when designing replication strategies.

To show the influence of the data popularity parameter on performances of replication strategies, three replication strategies were also tested in [18], namely: Periodic Optimiser [6], DR2 [49] and PDDRA [40]. In this respect, we compare the original version of each of the three aforementioned strategies, in which the popularity parameter is considered, with its popularity-unaware version. That is to say that the comparison is carried out with a modified version in which all datasets are considered as having the same popularity. We notice that the removal of the popularity parameter causes significant losses in the three strategies performances. According to the effective network usage (ENU) metric, the loss reaches 87.10 %. Also, the loss in terms of response time reaches 41.30 %.

This underlines the importance of considering the data popularity parameter, and justifies the reliance of several replication strategies on data popularity parameter to predict future requests whether in Grid [30,42–44,50,53], in Cloud [22,36,52,55], and in P2P systems [24,28,38,47].

Given the importance of data popularity in replication management, much research in distributed systems has been mainly focused on the popularity parameter [14,24,41]. However, these works did not take the popularity issue in all its aspects. Some important factors that will be highlighted in this work were indeed neglected in the literature. For example, the degree of the stability in the historic data accesses and the duality between calculation cost and result accuracy are of paramount importance in the correct assessment of the popularity parameter. They also did not show some imperfections that are experienced by the existing data popularity measurements. For example, the weight values that are affected to requests are not justified, and the tradeoff between the calculation cost and the precision degree cannot be controlled.

2.2 Considered factors in existing data popularity measurements

For high replication strategies performances, knowledge of future data popularity is of paramount importance. This is indeed crucial to decide which datasets have to be requested, replicated, or even deleted. However, the manner how the popularity is assessed varies from one strategy to another. The analysis of the existing data popularity measurements allows to identify the factors taken into consideration by each one.

In the general case, there are three main factors that contribute to the evaluation of data popularity that were highlighted in [18] and which are:

- *The number of dataset requests*: allows to identify how many times the dataset was requested.
- *The dataset lifetime*: allows to quantify the mean of the number of requests since the creation of the dataset.
- *The requests distribution over time*: allows to distinguish for a given dataset old requests from recent ones.

In fact, all the existing measurements consider the number of requests. However, the two other factors are considered by some measurements while neglected by others. Accordingly, measurements can then be classified into four categories as depicted in Table 1.

Among the measurements of the first category, we may refer to the measurement indicating the number of accesses to each dataset (denoted *#Requests*) [39]. It is among the most easy and natural metric to be used in order to quantify the actual popularity of a dataset. An example of the measurements of the second category is proposed by Al Mistarihi and Yong [3] (denoted RRD as acronym for Replica Request Demand) in which the number of requests is divided by the lifetime of the dataset. Mansouri and Asadi [33] proposed a measurement (denoted VSE as acronym for Value Storage Element) that belongs to the third category. The calculation of VSE is based on the number of requests and the timestamp of the last request, while the dataset lifetime is neglected. Among the measurements of the fourth category, we can cite the one proposed in [11] (denoted AF as acronym for Access Frequency). In this measurement, they consider the total number of requests while assigning a coefficient for each request so that the recent requests will have higher weights than the old ones. They also guarantee that the dataset lifetime will not be a reason for increasing the popularity through averaging the obtained value by dividing by the total number of periods.

It is in this respect important to mention that the efficiency of each measurement w.r.t. the calculation cost and the result accuracy closely depends on the category to which it belongs. Simple calculations are cheap in both space and time, while more complex (and hence more accurate) calculations required more space and calculation cost. The choice of the appropriate category is then subject to a tradeoff between result accuracy and calculation cost. This generally has the form shown in Fig. 1.

2.3 Drawbacks of existing measurements

According to the aforementioned factors, the popularity measurements may suffer from two kinds of drawbacks:

- Neglecting the dataset lifetime factor. Indeed, an old dataset may be favored when it is compared to a new one. This unfortunately gives a wrong indication of the popularity. It is worth mentioning that the first and the third categories suffer from this drawback.

Table 1 Considered factors by each category

	Number of requests	Dataset lifetime	Requests distribution over time	Example of measurement
First category	Considered	Neglected	Neglected	Ranganathan and Foster [39]
Second category	Considered	Considered	Neglected	Al Mistarihi and Yong [3]
Third category	Considered	Neglected	Considered	Mansouri and Asadi [33]
Fourth category	Considered	Considered	Considered	Chang and Chang [11]

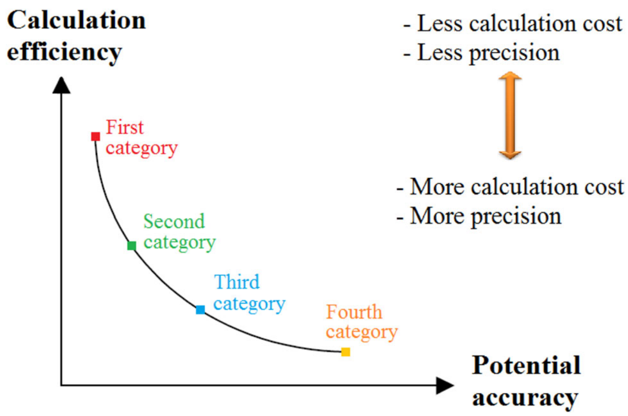


Fig. 1 Tradeoff between calculation efficiency and potential accuracy

– Neglecting the requests distribution over time, *i.e.*, the timestamp of each request. This does not allow to differentiate between old requests and recent ones, which is inconsistent with the temporal locality notion [1,21]. Noteworthily, the first and the second categories suffer from this drawback.

The fourth category measurements do not suffer from any kind of the two aforementioned drawbacks. This happens thanks to the fact that they consider the three aforementioned factors. Unfortunately, the fourth category measurements do not necessary lead to the best result compared to the other categories. Indeed, there is no need to consider the dataset lifetime factor when the dataset lifetimes are equal. Also, the requests distribution over time factor can be neglected when the temporal locality is not effective in such access pattern, knowing that the effectiveness of the temporal locality differs from one access pattern to another [35]. The experiments presented in Sect. 5 will further confirm this important fact.

Based on this, there are some situations when using one of the other categories is more appropriate than using the fourth category:

– The first situation occurs when the dataset lifetimes are equal, and the temporal locality is not effective. In this case, the first category deems more appropriate because it allows to take advantage of its low calculation cost.
– The second situation happens when the dataset lifetimes are not equal, and the temporal locality is not effective. In this case, privileging recent requests will not be necessary. The second category is then more efficient in this situation.
– The third situation arises when the dataset lifetimes are equal and the temporal locality is effective. In this case, the third category is more appropriate.

When considering the other situations, the fourth category is more appropriate because it considers the dataset lifetime and the temporal locality notion. However, the measurements of this category do not distinguish between high temporal locality degree and low temporal locality degree for a given access pattern. This drawback is not overcome by any existing measurement, and its consideration will constitute one of the key properties of the measurement proposed in the present work.

Figure 2 presents the appropriate category for each situation starting from the most particular case that suits the first category, to the general case which encompasses the first three particular cases and requires using the fourth category. The situation when the temporal locality degrees are different covers all the existing categories.

3 New method to measure the data popularity

In this section, we will propose a new data popularity measurement which will overcome all the discussed drawbacks by considering the three aforementioned factors in an effective manner. Indeed, our measurement will analyze the historic of each dataset access pattern, identify its situation and consider only the appropriate factors to this situation. It can hence join the appropriate category when there is a particular case. Otherwise, it will consider all factors at the same time. In addition, new factors will be taken into consideration which offers new advantages in comparison with existing measurements. The proposed measurement allows mainly to:

– take into consideration the dynamic nature of the system w.r.t. the changes in the dataset access pattern. In fact, the existing measurements that put greater weight on recent requests than older ones use the same weights every time. These weights are predefined and not justified. In our case, a function that reflects the behavior of the historic and scales automatically the weight values will be used. Indeed, this function will analyze the historic and identify how much the temporal locality is effective (*cf.* Sect. 3.4). The obtained temporal locality degree will give an indication of the probability of maintaining the same access pattern in the near future. In this way, we can assess whether or not recent historic requests represent a good indication of future requests, and hence design a series of weights to be applied to each request (*cf.* Sect. 3.5).
– offer the possibility to control the tradeoff between calculation cost and result accuracy. In some cases, one may prefer good accuracy while accepting high calculation cost while in another it may be more beneficial to have low calculation cost and tolerate less accuracy. Existing measurement techniques do not consider an approach to deal with this duality. In this work, a dynamic period

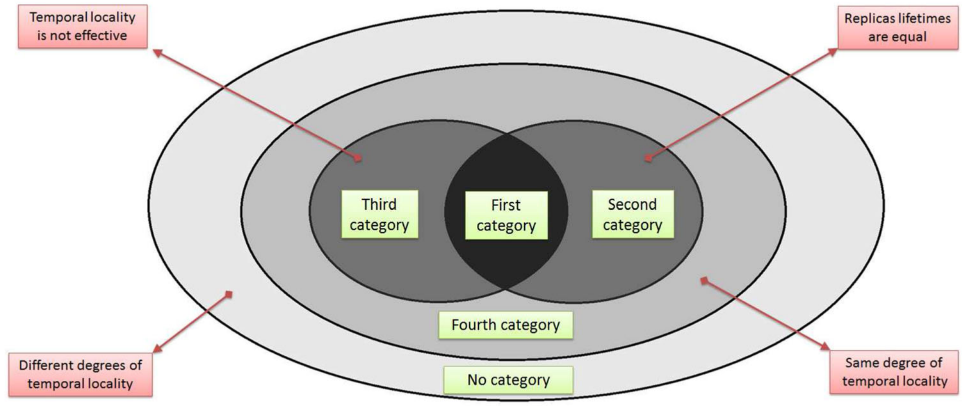


Fig. 2 Appropriate category for each situation

of requests counting will be used. The time for such a period can be shortened if the target is the accuracy of results, while it can be extended if the purpose is to obtain low calculation costs. A generic formula will be proposed and the system has to instantiate it according to its requirements by choosing dynamically the appropriate granularity level of the calculation process (*cf.* Sect. 3.2).

– to be at the basis of the employment of the popularity parameter in new usages in replication management. The popularity assessment will indeed be able to cover new objects, like the popularity of a given site. In this way, the popularity parameter will be able to contribute to some strategies that never used it before (*cf.* Sect. 4).

In the design of the proposed measurement, we begin by evaluating a specific set of requests issued from one given site to one given replica of a given dataset. Then, the evaluation is generalized in several ways to obtain several popularity measurements, like a measurement to quantify the popularity of a given replica, of a given dataset, of a given site, etc.

The evaluation process will pass through two main steps: initially, the time segment will be split into periods and the number of requests in each period will be counted. Then, each period will have a specific weight which will be multiplied by the number of requests in this period. Each weight will be dynamically computed according to the stability of the access pattern. The obtained total sum will be divided by the sum of the weights. The functions that will be used in this measurement are described in the following paragraphs.

3.1 T: timestamp determination function

The analysis of the distribution over time of the requests performed by a given site for a given replica requires the determination of the timestamp of each request in the access historic of the replica. In this regard, since the creation of the replica, the timestamp T_j of each performed request Req_i is recorded. At a given point in time, the function $T(Req_i)$ takes as parameter a request identified by its rank i with $1 \leq i \leq n$ and n is the total number of requests for the concerned replica issued by a given site. The function $T(Req_i)$ gives as a result the corresponding timestamp of the request Req_i. The timestamp determination function is then as follows:

$$T(Req_i) = T_j \tag{1}$$

To give the corresponding timestamp, this function uses T_0 to refer to the starting time when the concerned replica was created, and T_c to refer to the replica lifetime. Each replica has its proper T_0 and T_c. Each request Req_i, with $1 \leq i \leq n$, will then be associated to its timestamp T_j with $T_0 \leq T_j < T_c$.

Figure 3 shows an access pattern example, with $n = 7$ and $T_c = 30$. For example, the timestamp of the request Req_7 is T_{26}.

3.2 #PN: periods number determination function

The purpose is to split the time segment into periods in order to apply a specific weight to each period. The obtained number of periods is the factor that can control the trade-off between the calculation cost and the precision degree. Indeed, on the one hand, many periods means many weights that will be applied to the requests. This will increase the precision of the result, while increasing the calculation cost. On the other hand, a low number of periods will decrease the calculation cost since a small set of weights will be used in the computation of the replica popularity. This will however decrease the result precision. Therefore, we face two contradictory objectives: decreasing the calculation cost or increasing the result precision.

We consider a function *#PN(GL)* that takes as parameter an integer number representing the *Granularity Level* which is noted GL. This function gives as a result an integer the number of periods that will be obtained according to GL. The function is as follows:

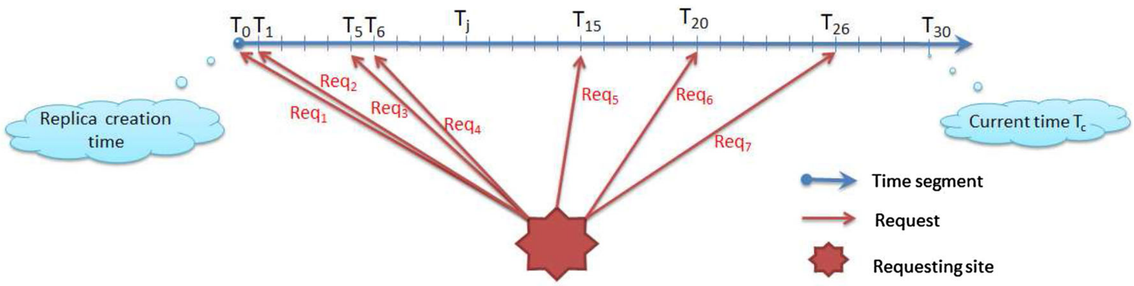


Fig. 3 Example of an access pattern

$$\#PN(GL) = \begin{cases} \lfloor \frac{T_c}{GL} \rfloor & \text{when } GL \in]0, T_c] \\ 1 & \text{when } GL \geq T_c \end{cases}$$

This means that the time segment will be split into $\#PN(GL)$ periods with $1 \leq \#PN(GL) \leq T_c$. The extreme case when $GL = 1$ will give $\#PN(1) = \lfloor \frac{T_c}{1} \rfloor = T_c$. In this case, the time segment will be split into T_c periods. There are hence T_c different weights that can be applied to the requests. This is the highest precision degree which will give the most accurate result. The other extreme case when $GL = T_c$ will give $\#PN(T_c) = \lfloor \frac{T_c}{T_c} \rfloor = 1$. So, the time segment will not be split into periods. All the requests will then have the same weight. Therefore, we will take advantage of its low calculation cost. In fact, this particular case meets the first category measurements, *i.e.*, considering only the number of requests while neglecting the replica lifetime and the requests distribution over time.

Let us take the same example of Fig. 3, in which $T_c = 30$. A fine granularity $GL = 1$ will give #PN(1) = 30, so the time segment will be split into 30 periods. A medium granularity $GL = 3$ will split the time segment into 10 periods. A coarse granularity $GL = 10$ will give 3 periods. While the maximum coarse granularity $GL = T_c$ will give #PN(T_c) = #PN(30) = 1.

3.3 P: corresponding period determination function

We consider a function $P(T_j)$ which takes as a parameter the timestamp T_j of the request Req_i, given by the function $T(Req_i)$, and returns its adequate period k with $1 \leq k \leq$ #PN(GL). The function $P(T_j)$ is calculated as follows:

$$P(T_j) = \left\lfloor \frac{T_j}{GL} \right\rfloor + 1 \tag{2}$$

Let us take the same example of Fig. 3. The corresponding period of each request according to different GL values is shown in Table 2.

After determining the corresponding period of each request, the total number of requests in each period k is counted and is denoted $\#Requests_k$. Therefore, an access pattern X will represent the partition of the different requests into the associated periods. It is then equal to: $X = \{\#Requests_1, \#Requests_2, \ldots, \#Requests_k, \ldots, \#Requests_{\#PN(GL)}\}$.

3.4 TLD: temporal locality degree function

3.4.1 TLD utility

Exploiting temporal locality of data has been a common idea for replication in distributed systems [21]. It is actually used for designing replication strategies in Data Grid [29], Cloud Storage [48] and other data storage systems [1]. It refers to the reuse of specific data within relatively small time duration. If at one point in time a replica is requested, then it is likely that the same replica will be requested again in the near future [1].

In data popularity assessment, this notion is specially exploited by the fourth category measurements. Indeed, these latter promote recent periods and give them higher weights than old periods. However, the weights they use are constant, predefined, and not justified since not depending on the access pattern of the considered dataset. For example, Chang and Chang [11] give a weight equal to 1 to the last period, 0.5 to the period before last, then 0.25, etc. They estimated that the last period deserves the double of the weight of the period before last, and they fixed the weights based on this estimation. This rigid estimation of the temporal locality of a dataset access may be accurate for some access patterns but inaccurate for others. Thus, the temporal locality is not guaranteed for all access patterns with the same degree. For example, the case where $X = \{6;25;2;27;3\}$, the last period is not even an indication of the future because the temporal locality is not effective. So, the fact of giving a double weight to the last period is inappropriate in this case. Likewise, when $X = \{8;8;9;9;9\}$, the temporal locality is highly effective. Then, the last period requests are likely to be repeated in the future and therefore they represent a very good indication of the future. As a consequence, they deserve a very higher weight compared to old requests.

Table 2 Corresponding period with different GL values

Request		Corresponding period			
Request Req_i	Timestamp T_j	GL = 1	GL = 3	GL = 6	GL = 15
Req_1	T_0	1	1	1	1
Req_2	T_1	2	1	1	1
Req_3	T_5	6	2	1	1
Req_4	T_6	7	3	2	1
Req_5	T_{15}	16	6	3	2
Req_6	T_{20}	21	7	4	2
Req_7	T_{26}	27	9	5	2

In our contribution, we will quantify the temporal locality effectiveness using a function, called *TLD*, which takes as a parameter the access pattern of a given replica w.r.t. to a given site and gives the *Temporal Locality Degree* of this access pattern. *TLD* analyzes the historic of the access pattern and gives the probability of maintaining the same access pattern in the near future. The weights that will be applied to each period will be scaled based on the obtained *TLD* value.

3.4.2 TLD calculation

Many mathematical functions can help us to assess how stable the access pattern is, *i.e.*, how strongly it likely supports a high degree of current temporal locality. We cite, for example, the *variance*, the *standard deviation*, and the *expected value*. In our contribution, we will use the *variance* function because it deems the most appropriate to our purposes. Indeed, the *variance* measures how far a set of numbers is spread out within a set of sample values [25].

The variance of an access pattern X is given by the following formula:

$$V(X) = \frac{\sum_{k=1}^{\#PN(GL)} (\#Requests_k - Avg\_Requests)^2}{\#PN(GL)} \quad (3)$$

where $Avg\_Requests = \frac{n}{\#PN(GL)}$ and n is the total number of requests in X.

In general, a low value of $V(X)$ means that the temporal locality is effective, while a high value means that the access pattern does not support the temporal locality notion. Based on this, the *variance* is a decreasing function of the temporal locality degree. Therefore, the temporal locality degree can be obtained from the inverse of the variance as follows:

$$TLD(X) = \frac{1}{V(X)} \quad (4)$$

In this way, as much as the temporal locality notion is effective, as much as *TLD(X)* increases. Table 3 shows some examples of *TLD(X)* values according to different access patterns.

Table 3 *TLD(X)* values with different access patterns

Access pattern X	Variance V(X)	TLD(X)
{8;7;8;9;8}	0.50	2.00
{7;8;6;8;9}	1.30	0.77
{9;4;10;4;6}	7.80	0.13

3.5 Weight function

3.5.1 Weight function properties

Firstly, we need a general function $f(x)$ allowing to affect an accurate weight to each period starting from the last one and going back to old periods. The function that will be used to play the role of $f(x)$ must verify three properties:

- $f(x)$ should be positive. As a consequence, the affected weights to the periods will be positive
- $f(x)$ should be decreasing. In this way, the most recent period will have the highest weight. Then, the weights will decrease with the going back to the past.
- $\lim_{x \to +\infty} f'(x) = 0$. The decrease of the weights should be quicker at the beginning in order to give more importance to recent periods. Then, the decrease should start to decline towards the stability, *i.e.*, towards $f'(x) = 0$, in order to not totally exclude old periods but only give them less importance.

Any instance of $f(x)$, verifying the aforementioned three properties, can guarantee what we target to reach through our proposal.

3.5.2 Weight function proposed instance

Many instances of $f(x)$ can be proposed. We cite, for example, $\frac{1}{x}$, $\frac{1}{e^x}$, $\frac{1}{ln(x)}$, etc. All these functions verify the three aforementioned properties. In this work, we will use the function $\frac{1}{ln(x)}$ in the interval $[2, +\infty[$. In fact, $\frac{1}{ln(x)} > \frac{1}{x} > \frac{1}{e^x}$ which allows to obtain higher weight values. Therefore, the differentiation between the weights of successive periods will be clearer.

In our evaluation process, the function f will take as a parameter the index of the period for which the associated weight will be calculated. Since $f(x)$ is decreasing, a period k will have the index $\#PN(GL) - k + 2$.

The weight of each period will be obtained from a combination of $f(x)$ and $TLD(X)$ so that the temporal locality degree will contribute in scaling the weights. In the general case, the combination must ensure the fact that as much as $TLD(X)$ is high, as much as the weights decrease rapidly. In other words, the difference between the weights of recent periods and those of old ones should expand. This allows us to exploit the effectiveness of the temporal locality in the concerned access pattern by promoting recent periods.

For an access pattern X, the weight W_k of the period k will then be as follows:

$$W_k = \left(\frac{1}{ln(\#PN(GL) - k + 2)}\right)^{TLD(X)} = ln^{-TLD(X)}(\#PN(GL) - k + 2) \quad (5)$$

In this way, the value of $TLD(X)$ determines the rate of decay of the weights when going back to the past. The higher the value of $TLD(X)$ is, the more recent periods will be favored over old periods. Besides, the decrease of the value of $TLD(X)$, which indicates that the temporal locality is not effective, will reduce the difference between the weights.

The impact of the value of *TLD(X)* on the weights is illustrated in Table 4 for different access patterns. We can note that the more the temporal locality is effective, the more recent periods are favored over old periods. Note that the difference between the weights is significant when $TLD(X)$ is high.

It is worth mentioning that two extreme cases exist, which are as follows:

- The first occurs when the same number of requests is repeated over time, *i.e.*, $V(X) = 0$. This represents the maximum temporal locality degree. The last period will be considered, in this situation, as a sufficient indication of future popularity. Therefore, our measurement will rejoin the third category measurements where the dataset lifetime factor is neglected.

Table 4 Impact of the temporal locality degree on the weights

Access pattern		Weight of each period				
X	TLD(X)	W_5	W_4	W_3	W_2	W_1
{8;7;8;9;8}	2.00	2.08	0.83	0.52	0.39	0.31
{7;8;6;8;9}	0.77	1.33	0.93	0.78	0.69	0.64
{9;4;10;4;6}	0.13	1.05	0.99	0.96	0.94	0.93

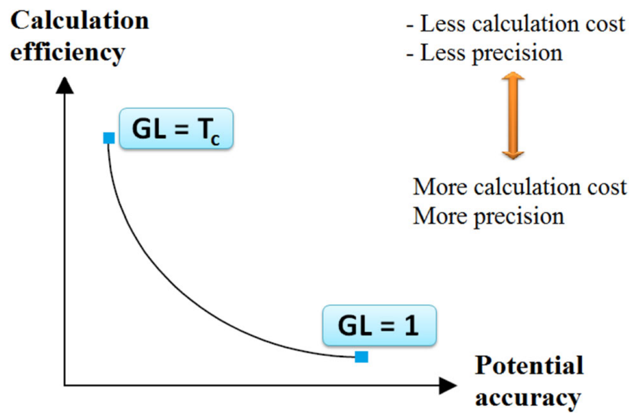


Fig. 4 Accuracy versus calculation efficiency tradeoff

- Whereas the second case arises when there is no temporal locality at all, *i.e.*, $TLD(X) \approx 0$. In this situation, all the weights will be almost equal because there is no need to differentiate between the requests on the basis of their timestamps. The calculation process will behave like it is calculating the mean of the number of requests throughout replica lifetime. Indeed, all the requests will have the weight value equal to 1. Our measurement meets then the measurements of the second category, *i.e.*, those considering the number of requests and the replica lifetime while neglecting the requests distribution over time.

3.6 RID: the proposed measurement

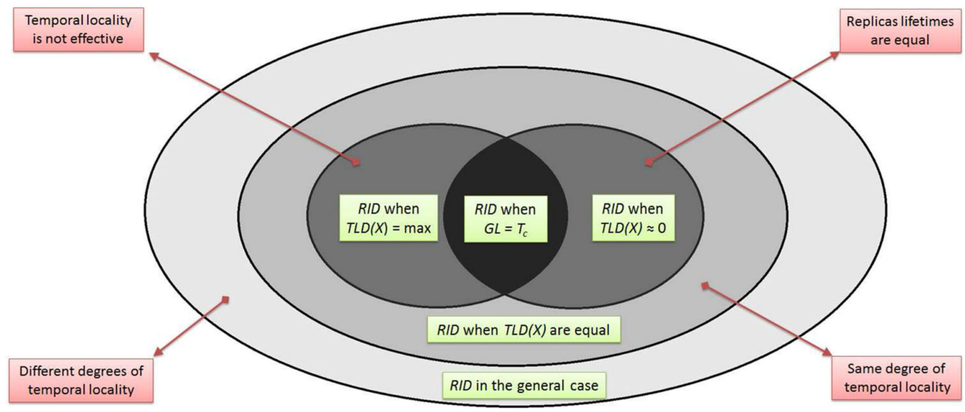
As highlighted above, our purpose is to quantify as a preliminary stage the intensity of the requests between one given site S and one given replica R. The measurement is called RID as an abbreviation of Requesting Intensity Degree. It calculates the sum of weights of all requests divided by the sum of weights of all periods as shown in the following formula:

$$RID(S, R) = \frac{\sum_{k=1}^{\#PN(GL)}(W_k \times \#Requests_k)}{\sum_{k=1}^{\#PN(GL)} W_k} \quad (6)$$

According to the value of GL, RID may move towards the accuracy target, as it may move towards the calculation cost diminution. The result of RID is then subject to a curve (potential accuracy versus calculation efficiency) which generally has the form shown in Fig. 4.

A comparison between this figure and Fig. 1 allows to note that RID can join the first category measurements when $GL = T_c$, so it will be dedicated specially for the diminution of the calculation cost, while it can join the fourth category measurements when $GL = 1$, so it will focus on the accuracy of the results even at the expense of the calculation cost. RID is then a generalization of the existing measurements and it

Fig. 5 RID covering of the other categories



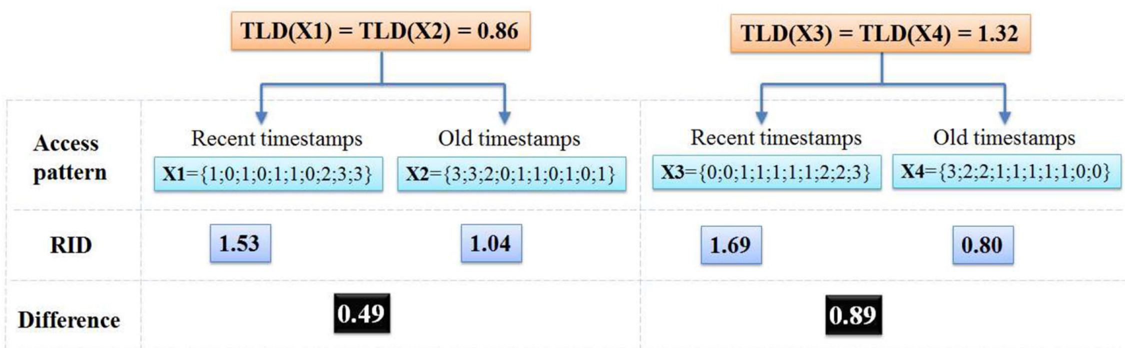


Fig. 6 Impact of the requests distribution over time factor

can cover all the categories. Figure 5 illustrates the different cases when RID joins the other categories.

3.7 Illustrative examples

3.7.1 Impact of the requests distribution over time

The impact of the requests distribution over time on RID is shown by some illustrative examples in Fig. 6.

A first access pattern $X1$ = {1;0;1;0;1;1;0;2;3;3} represents the partition over 10 periods of 12 requests associated to a replica and coming from a given site. In this case, *TLD(X1)* = 0.86 which gives RID = **1.53**. For the same number of requests and the same temporal locality degree, timestamps of requests are varied in order to note the influence on the results of the requests distribution over time. The access pattern after modifications is then $X2$ = {3;3;2;0;1;1;0;1;0;1} in which the majority of the requests have old timestamps. $X2$ gives RID = **1.04**.

Noteworthily that while having the same TLD value, the RID value of $X1$ is higher than that of $X2$. The difference reaches 0.49, although we maintained the same number of requests and the same temporal locality degree. This is due to the recency of the requests in $X1$.

Let us now consider $X3$ and $X4$ in which we varied the requests timestamps, as we did earlier, while this time under a higher temporal locality degree. We then have $X3$ = {0;0;1;1;1;1;1;2;2;3} and $X4$ = {3;2;2;1;1;1;1;1;0;0} which gives *TLD(X3)* = *TLD(X4)* = 1.32. The RID of $X3$ is equal to **1.69**, while for $X4$, RID = **0.80**.

The advantage of $X3$ is due to the recency of its requests. Furthermore, the difference in this case is expanded (equal to 0.89) because the comparison is made under a higher temporal locality degree. This offers more importance to the requests distribution over time. Likewise, low temporal locality degree will reduce the difference between the weights of old and recent requests which downgrades the impact of the requests distribution over time.

3.7.2 Impact of the number of requests factor

The number of requests issued by a given site towards a given replica is the parameter which has the most influence on the results. Indeed, any variation in the number of requests will cause a significant change in the result. However, this change is subject to the timestamps of the added/removed requests. In this respect, changes in old requests have less impact than those in recent requests. Some examples are shown in Fig. 7.

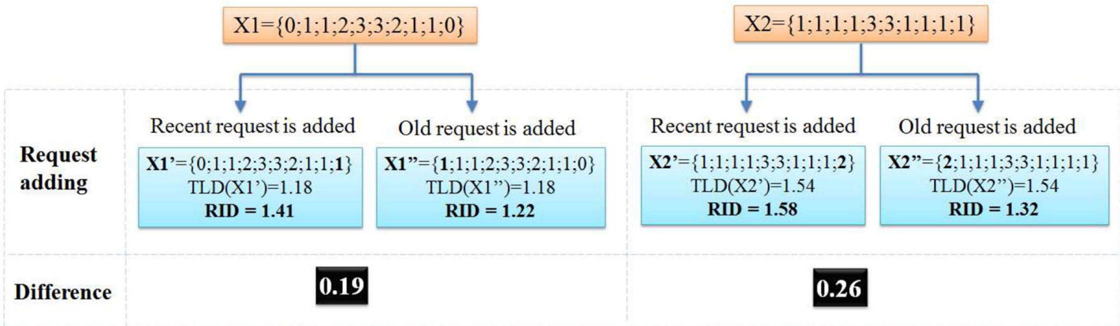


Fig. 7 Impact of the number of requests factor

We consider an access pattern $X1 = \{0;1;1;2;3;3;2;1;1;0\}$ with *TLD(X1)* = 0.96. On the one hand, a new request is added to the last period. The access pattern is then $X1' = \{0;1;1;2;3;3;2;1;1;\mathbf{1}\}$, which gives *TLD(X1')* = 1.18 and therefore RID = **1.41**. On the other hand, a new request is added to the first period. We then obtain $X1'' = \{\mathbf{1};1;1;2;3;3;2;1;1;0\}$ which gives RID = **1.22**, under the same temporal locality degree equal to 1.18. The difference between $X1'$ and $X1''$ is then equal to 0.19.

Now, we will remake the same process by adding a new request as we did earlier, but this time under higher temporal locality degree. Indeed, we take an access pattern $X2 = \{1;1;1;1;3;3;1;1;1;1\}$ having *TLD(X2)* = 1.56. A new request is added to the last period. So, we obtain a new access pattern $X2' = \{1;1;1;1;3;3;1;1;1;\mathbf{2}\}$, which gives *TLD(X2')* = 1.54, and therefore RID = **1.58**. Then, a new request is added to the first period. We then obtain $X2'' = \{\mathbf{2};1;1;1;3;3;1;1;1;1\}$ which gives, under the same temporal locality degree, RID = **1.32**. The difference between $X2'$ and $X2''$ is equal to 0.26.

The impact of adding a new request is subject to its timestamp. A recent request causes an increase in the value of RID more than an old one. Moreover, the impact of a new request on RID increases with the augmentation of the temporal locality degree. The same deductions are obtained when we remove a request. However, in this case, the value of RID will decrease according to the timestamp of the removed request.

4 Usages of the requesting intensity degree

In the following paragraphs, some usages of RID are presented and can be used as a basis for the design of new replication strategies. These strategies necessarily rely on some other parameters, in addition to RID, to take their decisions. However, we will focus here mainly on the manners that illustrate how RID can be exploited into these strategies.

Firstly, in order to model the treatment of data in the distributed system, we will consider that the system stores several datasets and each dataset has multiple replicas. These replicas are distributed through the sites of the system. We consider that there are p distinct datasets $\{D_1, D_2, \ldots, D_i, \ldots, D_p\}$. For each dataset D_i, there are m_i replicas. The replicas of D_i are then denoted $D_{i1}, D_{i2}, \ldots, D_{ij}, \ldots$, and D_{im_i}. Each replica D_{ij} is requested by $n_{D_{ij}}$ different sites $\{S_1, S_2, \ldots, S_k, \ldots, S_{n_{D_{ij}}}\}$.

Once we obtain an evaluation of the requesting intensity degree between a site S_k and a replica D_{ij}, *i.e.*, $RID(S_k, D_{ij})$, we can employ this information in multiple uses. This is detailed in the following paragraphs.

4.1 Measuring replica popularity

If we browse all the $n_{D_{ij}}$ requesting sites for one given replica D_{ij}, we can obtain a useful parameter that will be denoted *Replica_Popularity* and calculated as follows:

$$Replica\_Popularity(D_{ij}) = \sum_{k=1}^{n_{D_{ij}}} RID(S_k, D_{ij}) \qquad (7)$$

This parameter can be used in evaluating the placement of a replica w.r.t. its requesting sites. It can underpin solutions for the following problems:

- Which replicas should be removed from a storage element if there is not enough space to accommodate new ones? We can exploit the replica popularity to determine the set of replicas that should be removed while taking into consideration other important parameters, like the storage capacity and the distribution of the replicas of the same dataset among sites. From a stand-alone popularity view, replicas stored in the storage element can be ranked based on their popularity. Least popular ones have priority to be removed.
- Are the replicas well distributed in the system? The replicas distribution quality is an important parameter that must be considered in replication management [17]. Indeed, a replication strategy can be evaluated based on

the distribution quality pre-existing before the strategy is invoked and the distribution quality generated after [19]. Also, other evaluation metrics, such as response time and ENU, can be corrected to make the evaluation more objective by considering the distribution quality pre-existing before launching the strategy [20]. The quantification of the distribution quality is based on the evaluation of the placement of all the replicas of the system. As an improvement of the distribution quality assessment, the process can rely on the replica popularity, in addition to other parameters, to evaluate the placement of each replica.
– Which replica should be updated? There are two kinds of update propagation strategies, namely eager and lazy [34,46]. In eager strategies, all replicas are updated at the same time. However, in lazy strategies, replicas are updated progressively (one after another) until covering all replicas. In the case of lazy replication strategies, the choice of the replica to be updated first is an important issue. In this respect, measuring each replica popularity could serve for obtaining a replicas update order. When a dataset is updated, its replicas can be ranked based on their popularity. The most popular replica will receive the update before the others and so on.

4.2 Measuring dataset popularity

4.2.1 Dataset popularity w.r.t. the entire system

If we browse all the $n_{D_{ij}}$ requesting sites for all the m_i replicas of a given dataset D_i, we can obtain the parameter $Dataset\_Popularity$. This parameter measures the popularity of a given dataset relatively to the entire system. The dataset popularity of D_i is given by the following formula:

$$Dataset\_Popularity(D_i) = \sum_{j=1}^{m_i} Replica\_Popularity(D_{ij}) \quad (8)$$

This parameter is useful in solving several problems in replication strategies. We mention for example:

– When to do replication? $Dataset\_Popularity$ can help to decide the best time to do replication. For example, when $Dataset\_Popularity(D_i)$ exceeds a certain threshold, the dataset D_i deserves to be replicated since considered as popular.
– Any dataset should be replicated? The datasets can be ranked based on their popularity using the parameter $Dataset\_Popularity$. The most popular datasets will have the priority for replication.

4.2.2 Dataset popularity w.r.t. one given site

A site S not having a replica of the dataset D_i carries out remote accesses to read D_i from other sites. So, if we browse all the requests carried out by the site S for the remote replicas of D_i, we will obtain how much this site needs a replica of D_i. This is done as follows:

$$Site\_Dataset\_Popularity(S, D_i) = \sum_{j=1}^{m_i} RID(S, D_{ij}) \quad (9)$$

This parameter can underpin some replication strategies:

– It can help to decide where to place a new replica of a given dataset D_i. Indeed, the most needing site for this replica according to $Site\_Dataset\_Popularity$ will have the priority to obtain a new replica of D_i.
– For the strategies that use optimal number of replica (*ONR*) to decide how many replicas must exist in the system, they have to create new replicas if the current number of replicas is lower than (*ONR*). In this case, the parameter $Site\_Dataset\_Popularity$ can be used to decide which replicas should be created as well as their locations.

4.3 Measuring site popularity

Let us consider a given site S which stores n different replicas denoted as follows: $\{R_1, \ldots, R_i, \ldots, R_n\}$. If we browse all the n replicas of S and calculate their popularity using $Replica\_Popularity(R_i)$, we can quantify how much S is popular. For this purpose, the following formula is used:

$$Site\_Popularity(S) = \sum_{i=1}^{n} Replica\_Popularity(R_i) \quad (10)$$

This parameter can be used to predict the workload of a given site in the near future based on the popularity of the replicas it contains. A site having a large set of popular replicas will be classified as overloaded. We can use this key information in load balancing strategies by deciding which sites should be accessed in such a way the system will not be overloaded. In addition, a site which will perform a remote access to get data has, in general, an alternative choice among several replicas situated in several other sites. To avoid the system overload, it is better to access sites that have low *Site_Popularity* values since those sites are not overloaded by requests for the replicas they contain.

5 Experimental study

We will test the performances of RID experimentally using the OptorSim simulator [5,9] applied on the CMS testbed configuration [10]. OptorSim was developed by European data grid projects and is written in Java. It provides a framework to simulate the real grid environment. It consists in several sites and a Resource Broker. Each site may contain a Computing Element and/or Storage Element. The simulation parameter values and the workload characteristics are given in Table 5.

We aim to show that RID is accurate and putting it in use for replication management is beneficial. Two kinds of experimental study are then carried out. The first is an evaluation of the accuracy of RID predictions and its degree of conformity to the reality. The second is a substitution of an existing popularity measurement by RID within a replication strategy to highlight the added value of our proposal.

5.1 Accuracy of RID predictions

The performances of RID are compared to those of a measurement from each category of the four measurements categories (*cf.* Table 1). Our aim is to verify which one gives the closest prediction to the reality. In this regard, we choose $\#Requests$ [39] from the first category, RRD [3] from the second, VSE [33] from the third, and AF [11] as a representative of the fourth category. Moreover, two comparison methods will be used to evaluate the accuracy of RID.

5.1.1 Difference between the ranking given by a prediction and the real ranking

We aim to determine how much the ranking of replicas popularity given by the prediction of each tested measurement is near to the real ranking. The choice of such evaluation process is argued by the fact that we cannot compare the popularity values obtained by each measurement since they have different units. For this purpose, we calculate the absolute value of the difference between the prediction rank and the real rank associated to each replica. Then, for each measurement, we assess the degree of conformity between its prediction and the reality based on the total sum of the differences for all replicas.

As an explanative example of the evaluation process, let us consider five replicas R_1, R_2, R_3, R_4 and R_5. The prediction ranking is as follows: R_2, R_4, R_3, R_1 then R_5. The real ranking is as follows: R_3, R_4, R_5, R_1 then R_2. Our comparison method for this example is illustrated in Table 6. A low value of the sum then indicates that the prediction is near to the reality. In particular, when the sum is equal to zero, we can deduce the conformity between the prediction and the reality.

We will put this method in use to compare the popularity prediction given by RID with the real popularity at the 240th millisecond. Note that we choose the 240th millisecond because it allows us to calculate the popularity using enough historic access patterns. In this regard, we ranked 20 replicas according to RID. These 20 replicas cover all possible situations, that is to say replicas of same lifetimes and those of different lifetimes, replicas of high TLD and those of low TLD, locally requested replicas and remotely requested replicas, etc. Then, the real rank is determined based on the

Table 5 Parameter configuration and workload characteristics

Parameter	Value
Number of datasets	97
Size of each dataset	1000 Mb
Number of sites	20
Storage size at each site	50,000 Mb
Minimum bandwidth	45 Mb/s
Maximum bandwidth	10,000 Mb/s
Maximum queue size	200 jobs the Job Handler can keep in its queue
Job delay	2500 ms between the Resource Broker submitting each job
Scheduling algorithm for the Resource Broker	Scheduling is done using a combination of the access cost for the datasets and the access cost for all the jobs in the queue at each computing element
Datasets access pattern	Datasets are accessed sequentially in the order stated in the job configuration file
Initial datasets distribution	The original datasets are distributed randomly to the storage elements

Table 6 Illustrative example of the first evaluation process

Replica	Prediction rank	Real rank	Absolute value of the difference
R_1	4th	4th	0
R_2	1st	5th	4
R_2	3rd	1st	2
R_4	2nd	2nd	0
R_5	5th	3rd	2
Total sum			8

Table 7 Obtained difference values between the rankings given by the prediction result of each measurement and the real ranking

	# Requests	VSE	AF	RRD	RID	Gain (in %)
240 ms	86	68	42	37	26	30
280 ms	86	72	40	33	25	24
320 ms	84	67	48	42	26	38
360 ms	82	66	48	40	26	35
400 ms	79	77	40	44	28	26

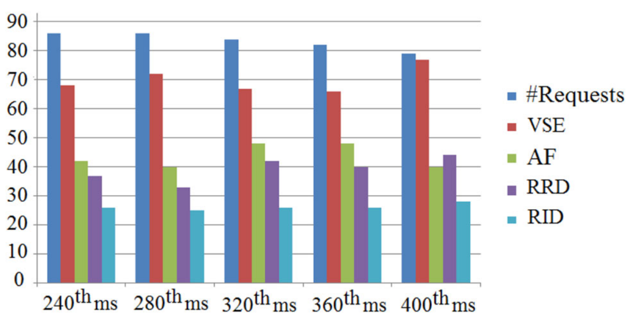


Fig. 8 Comparison of the performances of measurements w.r.t. the obtained difference values

rest of the execution, *i.e.*, the requests that will occur after the 240th millisecond. The sum of the absolute values of the differences between the two rankings is then equal to 26.

The evaluation performed for RID is also carried out for $\#Requests$, RRD, VSE and AF. In the same way, the predictions made in the 240th millisecond are also made in the 280th, 320th, 360th and 400th milliseconds. The results of these experiments are shown in Table 7 and illustrated in Fig. 8.

We can note that RID is more accurate than the other measurements. The gain reaches 38 % although we always compare RID with the best result among those of the four other measurements.

From another point of view, a comparison between RID and $\#Requests$, which gives the worst results, shows the degree of importance of the popularity parameter. Indeed, in the 280th millisecond, RID gives 25 and $\#Requests$ gives 86. The gain in this case is equal to 71 %. This significant difference between the results obtained through these two measurements highlights the importance of choosing a good measurement to assess the popularity.

5.1.2 Difference between the site needs given by a prediction and the real site needs

We present in the following paragraph a second evaluation process in order to compare the prediction result with the reality. In this regard, we focus on a set of replicas that a given site needs to execute its jobs. We then compare the prediction of future needs with the real future needs. Each site will be represented by a vector in the vector space generated by the replicas requested by this site. The coordinates of this vector represent the weights of each replica. This weight is calculated based on the rank of this replica in terms of popularity.

Let us take as an example a site S which requests for three replicas R_1, R_2 and R_3 with different requesting degrees. The popularity prediction gives the following ranking: R_2, R_3, R_1, while the real ranking is: R_3, R_2, R_1.

Each replica will have a weight according to its rank. The weight of a replica R is calculated in our case as follows: $Weight(R) = \frac{1}{rank(R)}$. Hence, the better the rank is the higher the weight is. Table 8 shows the weight of each replica. Then, the vector of the site S will be represented in the vector space as shown in Fig. 9 where:

$$\text{Prediction Vector}: \overrightarrow{P(S)} \begin{cases} 0.33 \\ 1.00 \\ 0.50 \end{cases} \qquad \text{Reality vector}: \overrightarrow{R(S)} \begin{cases} 0.33 \\ 0.50 \\ 1.00 \end{cases}$$

In the next step, we will calculate the cosine of the angle between the two vectors, $\overrightarrow{P(S)}$ and $\overrightarrow{R(S)}$, to determine the degree of conformity between the prediction and the reality. The more the cosine is close to 1 the more the angle is close to 0 which indicates the conformity between both vectors. The cosine in our example is then as follows:

$$cos(\overrightarrow{P(S)}, \overrightarrow{R(S)}) = 0.816 \tag{11}$$

In the general case, for a vector $\vec{V}$ with the coordinates $(V_1, V_2, \ldots, V_n)$ and a vector $\vec{W}$ with the coordinates $(W_1,$

Table 8 Weights calculation w.r.t. the rank of each replica

Replica	Prediction rank	Prediction weight	Real rank	Real weight
R_1	3rd	0.33	3rd	0.33
R_2	1st	1.00	2nd	0.50
R_3	2nd	0.50	1st	1.00

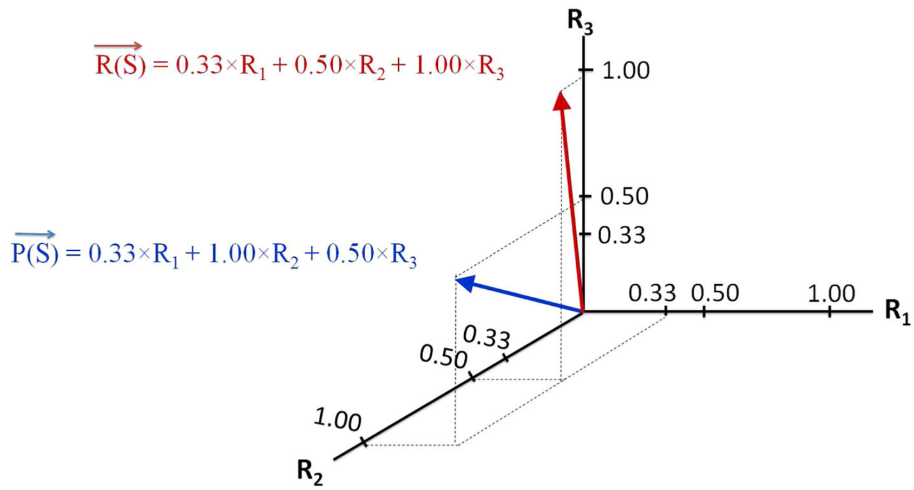


Fig. 9 Example of a site representation on a vector space generated by its requested replicas

Table 9 Obtained cosine values of the angle between the vector representing the real $Site16$ needs and the vectors representing its predicted needs

	# Requests	VSE	AF	RRD	RID
240 ms	0.798	0.796	0.767	0.850	0.875
280 ms	0.652	0.680	0.815	0.829	0.839
320 ms	0.672	0.670	0.714	0.768	0.840
360 ms	0.681	0.806	0.836	0.837	**0.914**
400 ms	0.674	0.589	0.826	0.833	**0.992**

Table 10 Obtained cosine values of the angle between the vector representing the real $Site17$ needs and the vectors representing its predicted needs

	# Requests	VSE	AF	RRD	RID
240 ms	0.620	0.542	0.823	0.831	**0.998**
280 ms	0.620	0.567	0.823	0.828	0.830
320 ms	0.622	0.622	0.963	0.831	**1.000**
360 ms	0.650	0.622	0.820	0.832	**0.998**
400 ms	0.734	0.734	0.890	0.758	**0.957**

$W_2, \ldots, W_n$), the cosine of the angle between $\vec{V}$ and $\vec{W}$ is calculated as follows:

$$cos(\vec{V}, \vec{W}) = \frac{\langle \vec{V}, \vec{W} \rangle}{||\vec{V}||.||\vec{W}||} = \frac{\sum_{i=1}^{n}(V_i \times W_i)}{\sqrt{\sum_{i=1}^{n}(V_i)^2 \times \sum_{i=1}^{n}(W_i)^2}} \tag{12}$$

This comparison model is exploited in our experiments to evaluate the prediction accuracy for two sites in the simulator: $Site16$ and $Site17$. We choose these sites because they are the most active ones during the experiments in terms of requests for various replicas either locally or remotely. The comparison is done at different timings and for the same five measurements cited above. The obtained results are shown in Tables 9 and 10, with their associated histograms in Fig. 10.

While not considering RID, the prediction accuracy of the other measurements varies from one moment to another. There is indeed no measurement that can be considered as being the best in all cases. This illustrates the fact that there is no measurement that is suitable for all situations.

On its side, RID offers better results than the other four measurements in all cases. This highlights its adaptivity to all situations. In addition, the prediction of RID was near to the reality several times. A cosine value greater than 0.9 is indeed obtained six times among ten. The prediction of RID even conforms to the reality in the 320th millisecond for $Site17$ which gives a cosine value equal to 1. In both Tables 9 and 10, bold values emphasize on the results ranging between 0.9 and 1 we obtained through the proposed measurement.

5.2 Benefits of RID in replication management

In this section, the effectiveness of RID is highlighted by employing it within a replication strategy. Indeed, we compared the original version of the Periodic Optimiser strategy

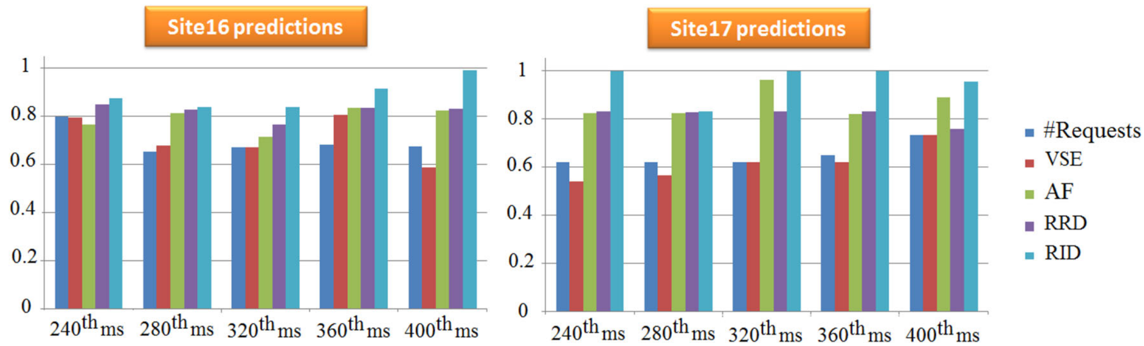


Fig. 10 Comparison of the performances of measurements w.r.t. the obtained cosine values

[6] with a modified version in which we used RID instead of the used original popularity measurement. RID is instantiated with a low value of GL equal to 1 for more accurate results. We then noticed the difference in terms of performances according to the effective network usage (ENU), the response time (RT) measured in *ms*, and the replication effect on the distribution (RED) [19]. The RED metric consists in evaluating the impact of the strategy on the replicas distribution quality. The results are shown in Tables 11, 12 and 13 respectively.

Obviously, the substitution of the original popularity measurement by RID has a positive impact on the strategy performances. This can be noted from the obtained gain which is proportional to the increase of the number of jobs. According to the ENU metric, 50 % of the performed replications have become more effective thanks to RID. In the same way, the results of the RED metric indicate that 40 % of the replicas become better placed when using RID.

Using an instance of RID which is dedicated for the accuracy goal allows then putting a large number of replicas in better placements. This will decrease remote accesses and offer further local ones. As a consequence, this allows also decreasing the response time. That is why we obtained better results also in term of response time. However, the gain w.r.t. this metric is not significant and does not exceed 5.47 %. This is due to the fact that we opted for an instance of RID with a

Table 11 Impact of RID on Periodic Optimiser from the perspective of the ENU metric

Number of jobs	Periodic Optimiser	Modified Periodic Optimiser	Gain (in %)
100	0.34	0.30	11.76
200	0.28	0.22	21.43
300	0.21	0.15	28.57
500	0.12	0.08	33.33
1000	0.08	0.04	50.00

Table 12 Impact of RID on Periodic Optimiser from the perspective of the RT metric

Number of jobs	Periodic Optimiser	Modified Periodic Optimiser	Gain (in %)
100	3410	3244	4.86
200	5266	5012	4.82
300	7023	6781	3.44
500	9408	8937	5.01
1000	14,471	13,679	5.47

Table 13 Impact of RID on Periodic Optimiser from the perspective of the RED metric

Number of jobs	Periodic Optimiser	Modified Periodic Optimiser	Gain (in %)
100	0.39	0.43	9.30
200	0.31	0.35	11.42
300	0.26	0.31	16.12
500	0.18	0.25	28.00
1000	0.06	0.10	40.00

low GL value which is dedicated for the accuracy goal more than for the diminution of the calculation cost.

6 Conclusion and future work

Correctly assessing data popularity is an important step towards the design of data replication strategies of high performance. In this work, we discussed many calculation methods of data popularity which exist in the literature. These measurements differ according to the factors taken into account by each one. A classification of these measurements into four categories is offered, while highlighting their main drawbacks. We then proposed a new measurement to assess data popularity. The proposed measurement considers all the highlighted factors as well as the temporal locality degree in order to adapt with any access pattern. In addition, it is generic so it can respond to the application requirements w.r.t. the tradeoff between calculation cost and result accuracy. This measurement is also useful in different contexts and can be exploited by several strategies towards multiple uses. Several experiment results were analyzed and allowed us to prove the effectiveness of the proposed measurement.

Future research work includes designing replication strategies for distributed systems based on the data popularity measurement proposed in this work in addition to other parameters. We also plan to design and evaluate a new instance of the proposed measurement which will be mainly dedicated for Read/Write systems. In this instance, another parameter will be added to analyze the quality of each request. It will distinguish between requests generating dirty reads and those requests generating proper reads. The proposal of an adaptive process for setting the value of GL according to the system/user requirements constitutes also an interesting issue.

References

1. Abad, C.L., Roberts, N., Lu, Y., Campbell, R.: A storage-centric analysis of MapReduce workloads: file popularity, temporal locality and arrival patterns. In: Proceedings of the 2012 IEEE International Symposium on Workload Characterization, pp. 100–109 (2012)
2. Aiqiang, G., Luhong, D.: Lazy update propagation for data replication in cloud computing. In: Proceedings of the 5th International Conference on Pervasive Computing and Applications, pp. 250–254 (2010)
3. Al Mistarihi, H.H.E., Yong, C.H.: Replica management in data grid. Int. J. Comput. Sci. Netw. Secur. **8**(6), 22–32 (2008)
4. Barrefors, B.: Dynamic data management in a data grid environment. Ph.D. thesis, University of Nebraska, USA (2015)
5. Bell, W.H., Cameron, D.G., Capozza, L., Millar, A.P., Stockinger, K., Zini, F.: OptorSim: a grid simulator for studying dynamic data replication strategies. Int. J. High Perform. Comput. Appl. **17**(4), 403–416 (2003)
6. Ben Charrada, F., Ounelli, H., Chettaoui, H.: An efficient replica placement strategy in highly dynamic data grids. Int. J. Grid Util. Comput. **2**(2), 156–163 (2011)
7. Bonacorsi, D., Boccali, T., Giordano, D., Girone, M., Neri, M., Magini, N., Kuznetsov, V., Wildish, T.: Exploiting CMS data popularity to model the evolution of data management for Run-2 and beyond. In: Proceeding of the 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP 2015), pp. 1–10 (2015)
8. Bsoul, M., Al-Khasawneh, A., Kilani, Y., Obeidat, I.: A threshold-based dynamic data replication strategy. J. Supercomput. **60**(3), 301–310 (2012)
9. Cameron, D.G., Carvajal-Schiaffino, R., Ferguson, J., Millar, A.P., Nicholson, C., Stockinger, K., Zini, F.: OptorSim v2.1 installation and user guide. Technical report, CERN (2006)
10. Cameron, D.G., Carvajal-Schiaffino, R., Millar, A.P., Nicholson, C., Stockinger, K., Zini, F.: Evaluating scheduling and replica optimisation strategies in OptorSim. In: Proceedings of the 4th International Workshop on Grid Computing, pp. 52–59 (2003)
11. Chang, R.-S., Chang, H.-P.: A dynamic data replication strategy using access-weights in data grids. J. Supercomput. **45**, 277–295 (2008)
12. Dayyani, S., Khayyambashi, M.: A comparative study of replication techniques in grid computing systems. Int. J. Comput. Sci. Inform. Secur. **11**(9), 64–73 (2013)
13. Dogra, N., Singh, S.: A survey of dynamic replication strategies in distributed systems. Int. J. Comput. Appl. **110**(11), 1–4 (2015)
14. Giommi, L.: Predicting CMS datasets popularity with machine learning. Master thesis, University of Bologna, Italy (2015)
15. Goel, S., Buyya, R.: Data replication strategies in wide area distributed systems. In: Enterprise Service Computing: From Concept to Deployment, pp. 211–241 (2006)
16. Grace, R.K., Manimegalai, R.: Dynamic replica placement and selection strategies in data grids: a comprehensive survey. J. Parallel Distrib. Comput. **74**(2), 2099–2108 (2014)
17. Hamdeni, C., Hamrouni, T., Ben Charrada, F.: New evaluation criterion of file replicas placement for replication strategies in data grids. In: Proceedings of the 9th IEEE International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 1–8 (2014)
18. Hamdeni, C., Hamrouni, T., Charrada, F.B.: Data popularity measurements in distributed systems: survey and design directions. J. Netw. Comput. Appl. **72**, 150–161 (2016)
19. Hamrouni, T., Hamdeni, C., Ben Charrada, F.: Impact of the distribution quality of file replicas on replication strategies. J. Netw. Comput. Appl. **56**, 60–76 (2015)
20. Hamrouni, T., Hamdeni, C., Ben Charrada, F.: Objective assessment of the performance of data grid replication strategies based on distribution quality. Int. J. Web Eng. Technol. **11**(1), 3–28 (2016)
21. Hockauf, R., Karl, W., Leberecht, M., Oberhuber, M., Wagner, M.: Exploiting spatial and temporal locality of accesses: a new hardware-based monitoring approach for DSM systems. In: Euro-Par98 Parallel Processing, Vol. 1470, pp. 206–215 (1998)
22. Hussein, M., Mousa, M.: A light-weight data replication for cloud data centers environment. Int. J. Innov. Res. Comput. Commun. Eng. **2**(6), 2392–2400 (2014)
23. Ikeda, T., Ohara, M., Fukumoto, S., Arai, M., Iwasaki, K.: A distributed data replication protocol for file versioning with optimal node assignments. In: Proceedings of the 16th IEEE Pacific Rim International Symposium on Dependable Computing, pp. 117–124 (2010)
24. Jacky, C., Kevin, L., Brian, N.L.: Availability and popularity measurements of peer-to-peer file systems. http://forensics.umass.edu/pubs/chu.labonte.p2pjournal.pdf. Accessed 1 Sept 2016
25. Kagan, A., Shepp, L.A.: Why the variance? Stat. Probab. Lett. **38**(4), 329–333 (1998)

26. Kangasharju, J., Roberts, J., Ross, K.W.: Object replication strategies in content distribution networks. Comput. Commun. **25**(4), 376–383 (2002)
27. Kia, H.S., Khan, S.U.: Server replication in multicast networks. In: Proceeding of the 10th IEEE International Conference on Frontiers of Information Technology, pp. 337–341 (2012)
28. Knoll, M., Abbadi, H., Weis, T.: Replication in peer-to-peer systems. In: Self-Organizing Systems, Vol. 5343, pp. 35–46 (2008)
29. Kolodziej, J., Khan, S.U.: Data scheduling in data grids and data centers: a short taxonomy of problems and intelligent resolution techniques. In: Transactions on Computational Collective Intelligence X, Vol. 7776, pp. 103–119 (2013)
30. Leu, F.Y., Lee, M.C., Lin, J.C.: Improving data grids performance by using popular file replicate first algorithm. In: Proceedings of the IEEE International Conference on Broadband, Wireless Computing, Communication and Applications, pp. 416–421 (2011)
31. Ma, J., Liu, W., Glatard, T.: A classification of file placement and replication methods on grids. Future Gener. Comput. Syst. **29**(6), 1395–1406 (2013)
32. Malik, S.R., Khan, S.U., Ewen, S.J., Tziritas, N., Kolodziej, J., Zomaya, A.Y., Madani, S.A., Min-Allah, N., Wang, L., Xu, C., Malluhi, Q.M., Pecero, J.E., Balaji, P., Vishnu, A., Ranjan, R., Zeadally, S., Li, H.: Performance analysis of data intensive cloud systems based on data management and replication: a survey. Distrib. Parallel Datab. **34**, 179–215 (2016)
33. Mansouri, N., Asadi, A.: Weighted data replication strategy for data grid considering economic approach. Int. J. Comput. Control Quantum Inform. Eng. **8**(8), 47–56 (2014)
34. Manu, V., Shailendra, V., Priyank, B., Singh, K.D.: Eager computation and lazy propagation of modifications for reducing synchronization overhead in file replication system. In: Proceedings of the 3rd IEEE International Conference on Computer and Communication Technology, pp. 331–334 (2012)
35. McKinley, K.S., Temam, O.: A quantitative analysis of loop nest locality. In: Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 94–104 (1996)
36. Myint, J., Hunger, A.: Modeling a load-adaptive data replication in cloud environments. In: Proceedings of the 3rd International Conference on Cloud Computing and Services Science, pp. 511–514 (2013)
37. Passarella, A.: A survey on content-centric technologies for the current internet: CDN and P2P solutions. Comput. Commun. **35**(1), 1–32 (2012)
38. Rahmani, M., Benchaiba, M.: A comparative study of replication schemes for structured P2P networks. In: Proceedings of the 9th International Conference on Internet and Web Applications and Services, pp. 147–158 (2014)
39. Ranganathan, K., Foster, I.T.: Identifying dynamic replication strategies for a high-performance data grid. In: Proceedings of the Second International Workshop on Grid Computing, pp. 75–86 (2001)
40. Saadat, N., Rahmani, A.M.: PDDRA: a new pre-fetching based dynamic data replication algorithm in data grids. Future Gener. Comput. Syst. **28**(4), 666–681 (2012)
41. Seddiki, M., Benchaiba, M.: Toward a global file popularity estimation in unstructured P2P networks. In: Proceedings of the 8th International Conference on Systems and Networks Communications, pp. 77–81 (2013)
42. Shorfuzzaman, M., Graham, P., Eskicioglu, M.R.: Popularity-driven dynamic replica placement in hierarchical data grids. In: Proceedings of the 9th IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 524–531 (2008)
43. Shorfuzzaman, M., Graham, P., Eskicioglu, R.: Adaptive popularity-driven replica placement in hierarchical data grids. J. Supercomput. **51**(3), 374–392 (2010)
44. Singh, S.K., Prasad, A., Singh, P., Singh, R.: A replica placement and replacement algorithm for data-grid in DRTDBS. In: Proceedings of the IEEE International Conference on Electronics and Communication Systems, pp. 1–5 (2014)
45. Soosai, A.M., Abdullah, A., Othman, M., Latip, R., Sulaiman, M.N., Ibrahim, H.: Dynamic replica replacement strategy in data grid. In: Proceedings of the 8th International Conference on Computing Technology and Information Management, Vol. 2, pp. 578–584 (2012)
46. Souri, A., Pashazadeh, S., Navin, A.H.: Consistency of data replication protocols in database systems: a review. Int. J. Inform. Theory **3**(4), 19–32 (2014)
47. Spaho, E., Barolli, L., Xhafa, F.: Data replication strategies in P2P systems: a survey. In: Proceedings of the 17th International Conference on Network-Based Information Systems, pp. 302–309 (2014)
48. Sun, D., Chang, G., Gao, S., Jin, L., Wang, X.: Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. J. Comput. Sci. Technol. **27**(4), 256–272 (2012)
49. Suri, P.K., Singh, M.: DR2: a two-stage dynamic replication strategy for data grid. Int. J. Recent Trends Eng. **2**(4), 201–203 (2009)
50. Tang, M., Lee, B.S., Tang, X., Yeo, C.K.: The impact of data replication on job scheduling performance in the data grid. Future Gener. Comput. Syst. **22**(3), 254–268 (2006)
51. Thampi, S.M., Sekaran, K.C.: Review of replication schemes for unstructured P2P networks. In: Proceedings of IEEE International Advance Computing Conference, pp. 794–800 (2009)
52. Wang, X., Yang, S., Wang, S., Niu, X., Xu, J.: An application-based adaptive replica consistency for cloud storage. In: Proceedings of the 9th IEEE International Conference on Grid and Cloud Computing, pp. 13–17 (2010)
53. Wang, Z., Li, T., Xiong, N., Pan, Y.: A novel dynamic network data replication scheme based on historical access record and proactive deletion. J. Supercomput. **62**(1), 227–250 (2012)
54. Watanabe, T., Kanzaki, A., Hara, T., Nishio, S.: An update propagation strategy considering access frequency in peer-to-peer networks. In: Database Systems for Advanced Applications, Vol. 4947, pp. 661–669 (2008)
55. Ye, Z., Li, S., Zhou, J.: A two-layer geo-cloud based dynamic replica creation strategy. Appl. Math. Inform. Sci. **8**(1), 431–440 (2014)

C. Hamdeni is a Ph.D. student in Computer Science at the Faculty of Sciences of Tunis, Tunisia, where he received in 2014 his M.S. degree in Computer Science. His current research interests include cloud computing, data grid and data replication.

T. Hamrouni obtained his Ph.D. in Computer Science in 2009 from the Faculty of Sciences of Tunis (Tunisia) and the University of Artois (France). Since, he is working as an Assistant Professor in the Department of Computer Sciences and Multimedia at the Higher Institute of Multimedia Art of Manouba, Tunisia. His current research interests focus on cloud computing, data grid, data replication, data mining and knowledge extraction.

F. Ben Charrada is working as a Professor in the Department of Computer Sciences at the Faculty of Sciences of Tunis, Tunisia. His current research interests include cloud computing, grid, data replication and scheduling.