International Journal of Swarm Intelligence and Evolutionary Computation

Ascó A, J Swarm Intel Evol Comput 2018, 7:3
DOI: 10.4172/2090-4908.1000171

Research Article Open Access

Optimising Data Access with an Adaptive Geo – Replication Strategy

Amadeo Ascó*

Trifork Leeds, Leeds, UK

***Corresponding author:** Amadeo Ascó, Trifork Leeds, Leeds, UK, Tel: +442071180242; E-mail: a.asco@bocaditos.co.uk

Received date: September 14, 2018; **Accepted date:** October 15, 2018; **Published date:** October 25, 2018

Abstract

The amount of data being processed in Data Centres (DCs) keeps growing at an enormous rate so that full replication may start being impractical. One way to increase data availability can be accomplished using replication between DCs so data may be accessed locally, if possible, which allows to recover in the presence of site failures and reduce access costs. This means that replicating the data only in some of the DCs is becoming more critical to reduce the costs of keeping the data consistent or eventually consistent and still maintaining a high availability (scalability) and low access costs. The data locations overall the DCs must be determined dynamically giving the changing patterns of read and write requests for the data to be replicated. Given that the problem of finding an optimal replication schema in a general network has been shown to be NP-complete for the static case, it is unlikely to be able to generate a general algorithm to find efficient solutions to the dynamic problem.

An adaptive bio-inspired replication strategy is presented here, which is completely decentralised, adaptive, inspired on the Ant Colony algorithm, and event-driven. Also, the replication protocol is independent of the strategy implemented but it is guided by the strategy.

Keywords Optimisation; Methaheuristics; Evolutionary algorithms; Adaptive geo-replication; Data centres

Introduction

The amount of data being processed in DCs keeps growing at enormous rate [1-3]. Some of the areas where the amount of stored data already reaches Terabytes (TBs) and even Petabytess (PBs) are data mining, particle physics, climate modeling, high energy physics and astrophysics, to site few, data which needs to be shared and analysed [4-6]. DCs are able to ensure that stored data is highly accessible and scalable. But the location of a DC in respect of the client accessing the data has an impact on availability, access times (latency-accessibility) and costs derived from providing the data. Replicating some of the data at multiple sites is a possible solution to reduce some of these undesirable effects [7-9]. An increase in the number of replications may result in a large bandwidth saving and lead to a reduction in user response time on reads or writes depending on the replication type, i.e. replication on read or write. But keeping too many replicas of the data incurs extra costs, such as extra replication traffic to keep all versions of the data coherent, extra required storage and extra computational power [10-11]. Also, the distance between the client and where the accessed data is located has an impact on the client experience. So, the reduction of unnecessary replication will decrease the cost of providing the service, the extra replication traffic and storage. This approach was considered by Pan et al. [12-15], where Mansouri [13] use some indicatives, such as last time the replica was requested, number of accesses, and the size of replica, to decide which data will be replicated, so reducing the number of replicas, whereas [14] use data mining techniques with the objective to extract meaningful information, which may be used in enhancing data replication and the selection of the replication strategies. All these approaches correspond to the area of Adaptive Geo-Replication Approach's (AGRAs).

Only replicating certain data is another area of study, which may also be combined with the one proposed by Asco et al. [15,16]. Under this category, Liu et al. [17] propose a selective data replication mechanism with the aim to reduce inter data center communication while still achieving low service latency by selecting user data for replication in Online Social Networks (OSNs).

The replications may be grouped into two types; static replication where a replica persist until it is deleted by a user or its duration expires, and dynamic replication where the creation and deletion of a replica are managed automatically and normally directed by the access pattern of the data used by the users [18]. In static replication the major drawback is an inability to adapt to changes in the access pattern of the data used by the users. Also, there are two types of replication based on their effect on the data; partial replication is concerned with the number of parts the full data is composed of, all of which may be located in different parts of the overall system, i.e. DCs, within a DC in different nodes or at the client-side [7,10,19]. A genuine partial data replication protocol for transactional systems is provided by Peluso et al. [19,20] and Schiper et al. [21] study the problem of partial database replication protocols and it compares favourably to existing solutions both in terms of number of messages and communication steps. Whereas adaptive geo-replication is concerned in what data and where the data or part of the data is located within the overall system of DCs and how many replicas exist simultaneously [16,22-26].

The problem of finding an optimal geo-replication schema in a general network has been shown to be NP-complete for the static case [27-29] so there is no known efficient geo-replication algorithm to locate a convergent optimal solution dynamically. Given this it is proposed an algorithm base on Wolfson's algorithm [30], which proposes an adaptive algorithm for replicated data between processors which considers the changes in the read-write pattern of the processors in the network, and it is also based on the principles of the Ant Colony Optimisation algorithms, which are inspired in the behaviour of ant

colonies when deciding which path to follow when foraging [31,32]. It should be noted that the main purpose of the replication is not to recover from disasters, as this is the responsibility of the recovery centres, which are data centres sufficiently close to the operational DC, they are associated to, so copies can be processed quickly enough but at the same time sufficiently far apart as to avoid any potential geographical issues that may happen to the DCs, i.e. earthquakes, fire that destroys the DC, the shutting down of main power plant which provides energy to the DC. Neither is the DC responsibility to provide analytical services as these are provided by the data warehouse (s). The main propose of the considered DCs is to provide operational access to clients (operational DCs), which corresponds to intensive read/write operations to the client's most recent data. The following references to DCs in this document referrer to operational data centres.

The adaptive bio-inspired replication strategy originally presented by Asco et al. [15] which is completely decentralised, adaptive, inspired on the Ant Colony algorithm, event-driven, and part of the Adaptive Geo-Replication Approaches is further studied here. The replication protocol is independent of the strategy implemented but it is guided by the strategy. This adaptive replication strategy dynamically establishes the location of the replicas, number of replicas, and when replicas are removed from a DC based on a pre-defined set of parameters. In the following sections a model of the work load of the problem is presented together with the replication strategy, followed by the algorithm, and then the replication strategy is compared with the performance of both the full replication and no replication.

Scale-Out

In a system comprising of n nodes, each node i has a capacity, C_i, which is composed of local work, `i, and remote work, R_i, where node i ϵ {1,…,n}, Equation 1. It is considered that the proportion of local load is part reads and the other writes, $l_i=R_{ei}+W_i$. The presence of the data in a DC i is expressed by X_i, with value of 1 if DC i has a replica of the data or 0 otherwise. If w_i is the part of the full load that are writes, then $W_i=w_i*l_i$ are writes, and $R_{ei}=(1-w_i)*l_i$ are reads, seen Table 1. In full replication any direct write to node i must be propagated to the other nodes with replicas, Equation 2.

$$C_i = l_i + R_i \quad (1)$$

$$C_i = l_i + X_i * \sum_{j=1, j!=i}^{n} (w_j * l_j) + \sum_{j=1, j!=i}^{n} X_j * w_i * l_i \quad (2)$$

Parameter	Description
n	The total number of nodes.
C_i	The capacity for node i.
l_i	The local work for node i.
R_i	The remote work for node i, $C_i = l_i+R_i$.
w_i	The percentage of the local work which is reads for node i, wi ϵ (0, 1).
Re_i	The part of the local work which is reads for node i, $Re_i=(1-w_i)*l_i$.
W_i	The part of the local work which is reads for node i, $W_i=w_i*l_i$.
X_i	Indicate if the data is replicated in node i, with value of 1 if it is replicated, and 0 if it is not replicated, X_i ϵ {0,1}.

Table 1: Scale-out parameters.

In full replication, $X_i = 1, \forall_i \in \{1, \ldots, n\}$, such that local received writes are to be send to all the other nodes, 1 n nodes, and to receive all the writes from the other nodes, Equation 3.

$$C_i = l_i + \sum_{j=1, j!=i}^{n} (w_j * l_j) + (n-1) * w_i * l_i \quad (3)$$

If it is considered a balanced process so all nodes have the same load, i.e. wi=w, li=l, and $C_i = C \forall_i \in \{1, \ldots, n\}$

.The capacity can be expressed as Equation 4 and the local work as Equation 5. In the case of only one node, with no replication, then l=C as expected, which means all the capacity is used in local work. The maximum capacity is dependent of the node. If all the nodes are balanced then the scale-out is expressed by Equation 6, which values are shown in Figure 1 for full replication, different number of nodes n and write load w of 0% (no writes), 5%, 30%, and 50%.

$$C_i = l + (n-1) * w * l + (n-1) * w * l = (1 + 2 * w * (n-1) * l) \quad (4)$$

$$l = \frac{C}{1 + 2 * w * (n-1)} \quad (5)$$

$$scale - out = \frac{\sum_{i}^{n} = 1^{li}}{c} = \frac{(n*l)}{c} = \frac{n * (\frac{C}{1 + 2 * w * (n-1)})}{C} = \frac{n}{1 + 2 * w * (n-1)} = \frac{1}{\frac{1}{n} + 2 * w - \frac{2 * w}{n}} = \frac{1}{2 * w + \frac{1 - 2 * w}{n}} \quad (6)$$

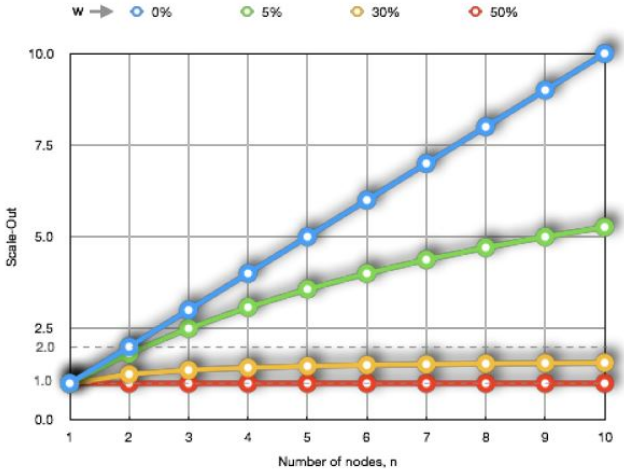

Figure 1: Scale-out for full replication.

If n=1, with no replication, then all capacity is used in local work and any increase in the number of nodes correlates to a linear increase in total local work capacity, which it is also corroborated by the scale-out, blue line on Figure 1. But it also corresponds to losing the recovery capability provided with replication, which may reduce availability, and increase access times.

As an increase in the replication means a decrease in the local work, so it is advisable to choose a level of replication more suited to the specific requirements, with full replication being more unsuitable as the number of nodes increases since more of the capacity of the nodes is used for remote work not local work. So the replication could be restricted only to n_0 of the nodes, where $1 \leq n_0 \leq n$, Equation 7 and Figure 2.

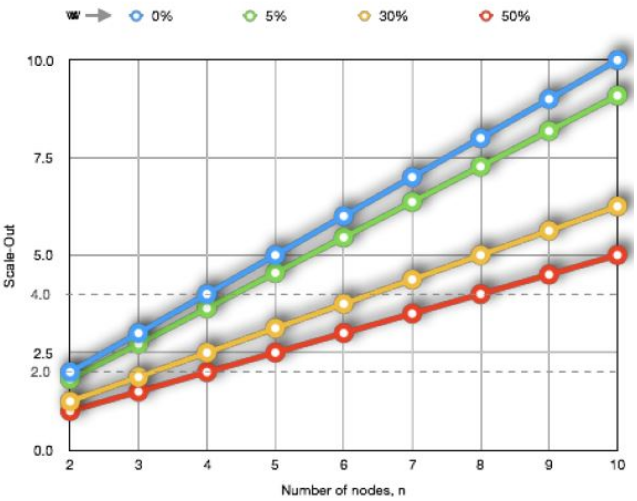

Figure 2: Scale-out for fixed replication with n_0=2.

$$scale-out = \frac{n}{1+2*w*(n_0-1)} (7)$$

All this means that ending an optimal replication distribution that minimises the amount of network traffic given certain read and write frequencies for various objects should alleviate these extra costs when replicating [7,33]. Given the volume of operations considered, which are predominately higher in reads than writes, and the speed of the access expected then any algorithm suitable to be applied to this constraint optimisation problem must have a very fast execution time or it will have a detrimental effect in the latency.

Algorithm

The general idea of this algorithm is to decide without the need of human intervention where and when to replicate or remove replicas with the main objective of reducing the latency and network traffic (reduce usage of bandwidth). This algorithm was previously presented in [16].

In general terms any read operation in a DC reinforces the need for a replica of the data in such DC, similarly but perhaps with a different degree it happens with the write operations, which also decrease the need for a replica of the data in the other DCs with replicas of the data, so eventually these DCs will not have any replica of the data. Given that we do not want to keep replicas, if it is not necessary, then the need for such replica will decay as time pass, but always making sure that the data is present (replicated) at least in as many DCs as the stated minimum number of DCs. This algorithm is further explained below together with a mathematical representation. The variables and constants used in the algorithm are summarised in Table 2.

Equation 8 represents the existence of a replica of data k in DC d, X_{kd}=1, or its absence X_{kd}=0, where D_k is the set of replicas of data k. n_k is the number of replicas of data k, as expressed in Equation 9, which must be bigger or equal to the minimum number of replicas for data k (n_k^-) and smaller or equal to the total number of DCs expressed by n as represented in Inequality 10.

$$X_{kd} = \begin{cases} 1 if D_k \in DC_d (\exists D_{kd}) \\ 0 otherwise \end{cases} (8)$$

$$n_k = \sum_{d=1}^{n} X_{kd} (9)$$

$$n_k^- \leq n_k \leq n (10)$$

Equation 11 shows that the replication strength for the data k in DC d is increased by the reads and writes requested through DC d, with intensities r_{kd} and w_{kd} respectively, and weakened by the writes requested through other DCs than DC d, with intensity w_{kdi}, and it is furthermore weakened by a temporal decay on the replication strength (last term in the equation). Where Equation 11 states that only DCs with a replica of the data, D_k, will be penalised by writes and time, which in practice it is achieved by not sending the write to those DCs without a replica so it does not incur in any extra cost when implementing it. Similarly, the decay in time, k_d, will not be applied to DC d without replicas of the data with the exceptions of those candidate DCs where the data may later be replicated. Reads may increase the number of replicas where writes may strengthen the replication in a DC but may also potentially remove a replica from one

of the other DCs, effect that it is strengthen by the temporal decay of the replication strength.

$$F_{kd} = \max\left(0, \min\left(L_{kd}, T_{kd} * \Delta w_{kd} - \sum_{\substack{i=1, i \neq d}}^{n} X_{ki} * w_{kd} * \Delta w_{kdi} - X_{kd} * T_{kd}\right)\right) \quad (11)$$

T_{kd}^{+} is the threshold of the replication strength of data k which determines when to keep an existing replica in DC d, as expressed in Inequality 12.

$$F_{kd} \leq T_{kd}^{+}, \exists D_{kd}, t < t_0 \wedge F_{kd} > T_{kd}^{+}, t_0 \Rightarrow \exists D_{kd}, t_0 (D_{kd} \in DC_d) \quad (12)$$

Variable	Description	Type
DC	It is the set of all DCs. d identifies one of the DCs, d ϵ {1,….,n}. DC_d represents the DC d which holds some data.	d ϵ N^+
D	It is the set of all the data. k identifies one of the data within the data set D, k ϵ {1,….,|D|}. D_k represents the set of replicas of data k.	k ϵ N^+
n	It is the total number of DCs, n=|Dc|.	N^+
n_k^-	It is the minimum number of replicas of data k, with 1≤ n_k^-≤|D_k|≤n, default n_k^-=1. The number of DCs that can be safely removed from the system without information loss is n_k^-−1.	N^+
n_k^+	It is the maximum number of replicas of data k with n_k^-≤|D_k|≤n_k^+≤n, default n_k^-=1.	N^+
n_k	It is the number of replicas for data k, D_k, with n_k = |Dk|.	N^+
r_{kd}	It is the number of reads for data k requested on DC d.	N_0
Δr_{kd}	It represents the strengthening of the replication of data k in the DC d used to execute one read.	R^+
w_{kd}	It is the number of writes for data k directly requested on DC d.	N_0
Δw_{kd}	It represents the strengthening of the replication of data k in the DC d directly used to execute a write.	R^+
w_{kdi}	It represents the decay of the replication in the DC i consequence of the write request in DC d. This value will depend of both DCs d and i and may also depend on the time of the day or other useful information available at the time it is used.	R^+
r_{kd}	It is the decay of the replication strength of data k in DC d with time. A simple example corresponds to a constant decay (T_{kd}) of the replication strength from the time the replica was created, r_{kd} =Δt*T_{kd}.	R^+
T_{kd}^+	It is the replication strength of data d in DC d required to start the replication of data k in a DC which does not currently contain a replica of the data.	R^+
T_{kd}^-	It is the replication strength of data d in DC d from where the replication of the data is removed, remove threshold for k in a DC which contains currently a replica of the data, default T_{kd}^-=0.	R^+
L_{kd}	The maximum replication strength for data k in DC d.	R^+
F_{kd}	The replication strength for data k in DC d, $F_k \leq L_k$.	R^0
X_{kd}	It refers to the existence of a replica of data k in DC d, with value 1 if the replica k exists in DC d or 0 otherwise.	{0,1}

Table 2: List of variables and constants.

T_{kd} is the threshold of the replication strength of data k which determines when to remove an existing replica in DC d where the other constrains is still kept. i.e., minimum number of replicas, as expressed in Inequality 13. Also a simple view of the thresholds is shown in Figure 3.

$$F_{kd} > T_{kd}^{-} \wedge \exists D_{kd}, t < t_0 \wedge F_{kd} > T_{kd}^{-}, t_0 \Rightarrow \exists D_{kd}, t_0 (D_{kd} \in DC_d) \quad (13)$$

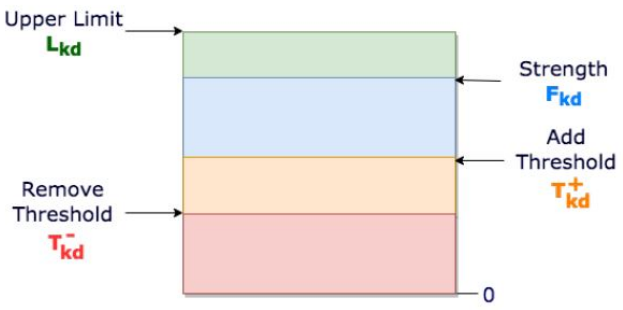

Figure 3: Constraints.

Inequality 14 shows, that for the data k in DC d, the removed threshold of the replica, T_{kd}^{-}, is positive and smaller or equal to the add threshold of the replica, T_{kd}^{+}, which it is smaller or equal to the maximum replicate strength, L_{kd}. The Equations 15 and 16 show the cases where a data k is not replicated in a DC d. A replica in a DC, with replication strength not bigger than the remove threshold, T_{kd}^{-}, will only be destroyed if the number of replicas is bigger than a minimum, n_k^{-} as represented in Inequality 15. Also for a DC which does not have a replica of the data the replication strength must be higher than a pre-set threshold T_{kd}^{+} to create a new replica, as represented in Inequality 16.

$$0 \le T_{kd}^{-} \le T_{kd}^{+} \le L_{kd} \quad (14)$$

$$\exists D_{kd}, t < t_0, n_k > n_k^{-}, F_{kd} > T_{kd}^{-} \equiv X_{kd} = 1 \Rightarrow \quad (15)$$
$$\exists D_{kd}, \quad t_0, F_{kd} \le T_{kd}^{-} \equiv X_{kd} = 0$$

$$\exists D_{kd}, t < t_0, F_{kd} \le T_{kd}^{+} \equiv X_{kd} = 0 \Rightarrow \quad (16)$$
$$\exists D_{kd}, \quad t_0, F_{kd} \le T_{kd}^{+} \equiv X_{kd} = 0$$

Each DC with a replica of the data must know about the other DCs that have replicas of the same data in order to manage the replication of such data.

If data k is only replicated in one DC d and a write is requested using a different DC j than the one it is currently replicated in, so that its replication strength in DC d is reduced to zero or under ($F_{kd}<0$), then the data will continue to be replicated in that DC d until a replica is place in another DC or its replication is increased over T_{kd}^{-}.

For the time being, it is considered the case where w_{kdi} increases with the distance (network distance) between both DCs d and i. Also it is assumed that the value is symmetrical, such that $w_{kdi}=w_{kid}$, so there is the same cost of transferring the data from DC d to DC i than from DC i to DC d. If to transfer the data between both DCs d and i requires the use of an intermediate DC j then $w_{kdi} \ge w_{kdj}+w_{kji}$, similarly if many intermediate nodes are used the decay will be at least the sum of the intermediate decays. The effect of Γk_d is required to ensure that in absence of writes some replications will still vanish and if the reads are concentrated in a few DCs, n_k^{-}, then the replicas in other DCs will be eventually removed. There is an extra requirement in the case that the data only exists in n_k^{-} DCs, in which case, the temporal effect should be ignored, so the data exists at least in n_k^{-} DCs. A read request to a DC, which does not have a replica of the data, will be forwarded to the closest DC with a replica. The DC with a replica will not gain strength from this read operation as the read was not initiated directly from itself. This DC will then have knowledge of the data but not a replica,

So this knowledge will be used in subsequent reads/writes to the DC which eventually may keep a replica. Once the replication strength is higher than the threshold (T_k^{+}) and the data is not already replicated on the maximum number of DCs ($n_k<n_k^{+}$), this DC will notify all the DCs with a replica of the existence of the new DC with a replica. It may also be required to use another temporal effect, the Time To Live (TTL), to make sure that eventually the data will be fully removed. This value may not be applied to data stored in recovery centres and data warehouses which may have their own TTL. Also it would be desirable for data that expires to be copied into those data centres before it is removed from all the DCs.

Overall it is assumed that reads and writes are not treated differently (not using Command Query Responsibility Segregation (CQRS)), so they are not directed to different DCs, otherwise it may be needed some adjustment to the approach presented here or may even invalidate it. The algorithm is optimal in the sense that when the replication scheme stabilises, the total number of replicas required for the reads and writes is minimal. The read sequence diagrams for this algorithm are shown in Figures 4, and the write ones are shown in Figures 5. The Figure 4(b) may also have to notify all DCs with a replica of the data of the new replica, which it will be sent by the direct DC k for when the threshold has been passed and a replica is created in the direct DC k.

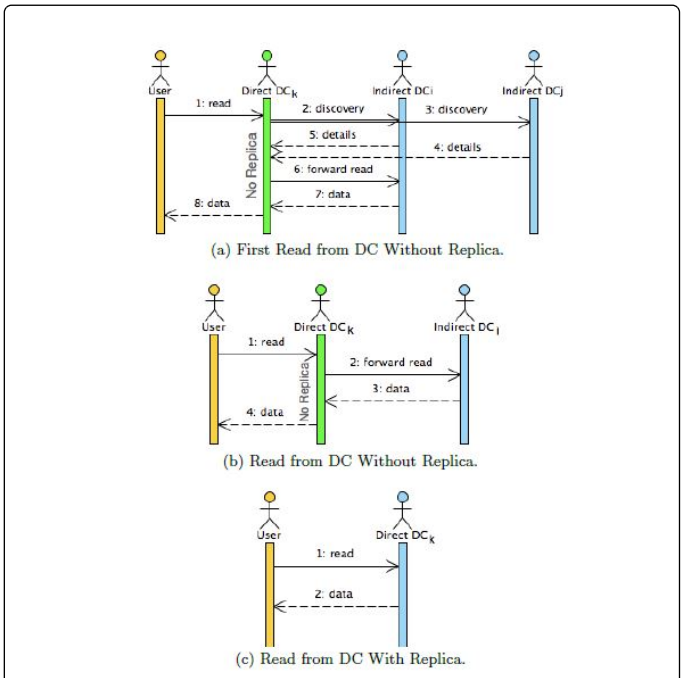

Figure 4: Read sequence diagrams on write replication.

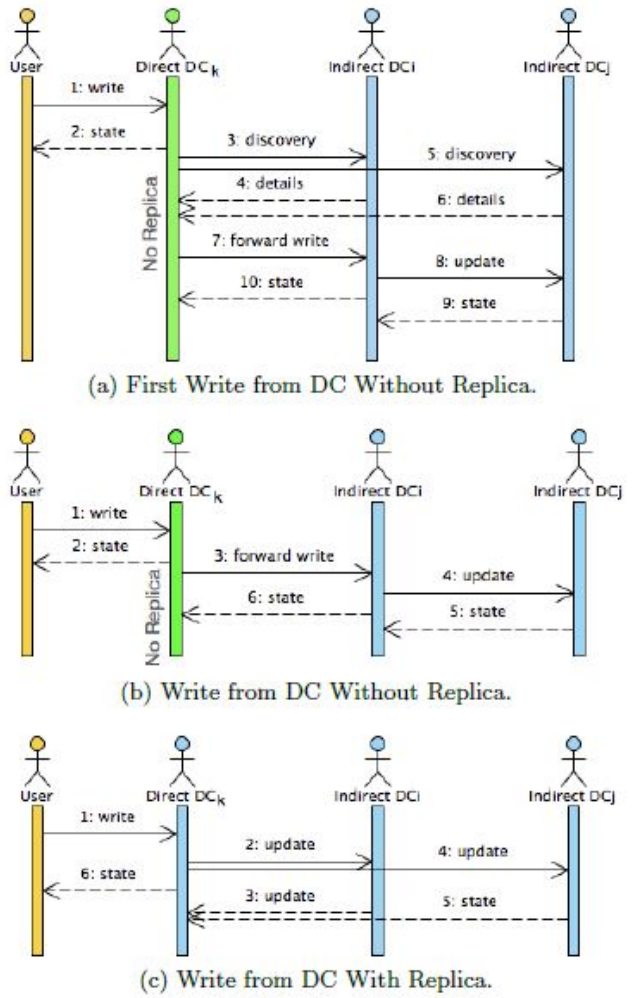

Figure 5: Write sequence diagrams on write replication.

The creation of data will generate a replica in the directly accessed DC, as the number of replicas must be at least one ($n_k^- \geq 1$). If the minimum number of replicas is bigger than 1 then the algorithm should generate and distribute the other replicas between the vicinity DCs. It is proposed that the approach distributes the created replicas to be implemented as a pluggable distribution approach, i.e. distribute replicas within the vicinity. The creation of the data will require checking if the data exists already, i.e. in a key value tuple that the key does not already exist in any of the other DCs. Then only in the case where the data does not already exist in any of the DCs the data will be replicated in the direct accessed DC and distributed replicas within other of the DCs based on the current creation approach and the minimum number of replicas.

Given that the number of reads is expected to be higher than the number of writes, and normally a read will be executed before a write, then it may make sense for Δw_{kd} to be smaller than Δr_{kd} ($\Delta w_{kd} < \Delta r_{kd}$) so that more writes would be required to maintain or create a replica in DC d than when using reads. Some of the parameters may be generalised even further by allowing them to depend on the DC where the calculation is executed in. Even the terms $r_{kd}{}^*\Delta r_{kd}$ and $w_{kd}{}^*\Delta w_{kd}$ could be generalise to consider other factors like bandwidth used and current storage capacity available in the DC d, or new terms could be added to take account of those new factors. Paiva et al. [32] use a very simple representation of the available storage capacity, which is taken into account in the replication of the data. We have assumed that the capacity in a DC is sufficient and when more storage capacity is required then extra storage is added to the DC. If this is not the case and S^d is the total storage capacity in DC d, and s_k is the size of the data to store then Equation 17 provides the free storage available in DC d, $S_{free}{}^d$.

$$S_{free}^{d} = S^{d} - \sum_{k=1}^{|D|} s_k * X_{kd} \quad (17)$$

As it has been seen previously having full replication introduces some disadvantages, which may be desirable to limit. With this propose in mind, it could also be introduced a maximum number of replicas, $n_k^- \leq n_k^+ \leq n$. This would require that the rules to identify when a new replica is placed is updated by having into account the replication threshold, T_{kd}^+, for the DC d where to place a replica, and the minimum strength between all the DCs, F_k^-, where there are already replicas, Equation 18, so it is possible to ascertain if the new replica will be placed on DC d and identify which DC j will lose the replica from the DCs with replicas for when the maximum number of replicas has already been archived, as stated in Equations 19 and 20.

$$F_k^- = \min_{i=1, X_{ki}=1}^{n} F_{ki} \quad (18)$$

$$X_{kj} = \begin{cases} 1 \begin{cases} if n_k < n_k^+ and F_{kj} > T_{kj}^+ \geq F_k^- \\ if n_k = n_k^+, F_{kj} = F_k^- and F_{kd} \leq F_k^- \end{cases} \\ 0 \begin{cases} if n_k = n_k^+, F_{kj} = F_K^- and F_{kd} > F_k^- \\ otherwise \end{cases} \end{cases} \quad (19)$$

$$X_{kj} = \begin{cases} 1 \begin{cases} if n_k < n_k^+ and F_{kd} > T_{kd}^+ or \\ if n_k = n_k^+ and F_{kj} = F_{kd} > F_k^- \end{cases} \\ 0 otherwise \end{cases} \quad (20)$$

Accessibility in general is increased for adaptive replication when compared to no replication. But also accessibility in general decreases for adaptive replication when compared to full replication, given that when fewer replicas exist there are more chances that, on partition, a user is in the side of the partition where there is not any replica. Of course, this can be reduced by increasing the minimum number of replicas that must exist at each time and even adding extra constraints in relation to where those replicas must be in relation to each other.

Mathematical cost representation

It is considered that there is a direct cost associated to the reads (crkd) and writes (cwkd) executed in the system for the data k in the DC d. Also, there is a cost associate to the reads (crkdj, crkdd=0) and writes (cwkdj, cwkdd=0) between DCs d and j for data k, shown in Table 3. There is a cost incurred when a replica is placed in a new DC and when the data is removed from a DC which is not present. If X'kd is the new replication state after all the current operations have been executed with a cost per notification of cr*kdj then the cost of a new

replica can be represented as in Equation 21, and the cost of removing a replica from a DC, with cost per notification of c^{r}_{*kdj}, can be represented as in Equation 22.

Cost	Description	Type
$X_{kd}(t)$	It refers to the existence of a replica of data k in DC d for time t, with value 1 if the replicas exists or 0 otherwise.	{0,1}
$X`_{kd}(t)$	The new replication state of a replica of data k from DC d for time t after all the current operations have been executed.	{0,1}
$r_{kd}(t; t+dt)$	It is the number of reads for data k requested on DC d from time t to t + dt. It could be also represented as $r_{kd}(t; t+dt)=r_{kd}(t + dt)-r_{kd}(t)$, where $r_{kd}(t)$ is the number of reads from the beginning up to time t.	N0
$w_{kd}(t; t+dt)$	It is the number of writes for data k directly requested on DC d from time t to t+dt. It could be also represented as $w_{kd}(t; t+dt)=w_{kd}(t+dt)-w_{kd}(t)$, where $w_{kd}(t)$ is the number of writes from the beginning up to time t.	N0
ρkd(t; t+dt)	The existence of an operation from time t to t+dt, which is 1 if $(r(t; t+dt)_{kd}+w(t; t+dt)_{kd})>0$ or zero otherwise.	{0,1}
$c_{kd}{}^{r}(t)$	The cost of a read request of the data k to DC d for time t.	R^+
$c_{kd}{}^{w}(t)$	The cost of a write request of the data k to DC d for time t.	R^+
$c_{kdj}{}^{r}(t)$	The cost of a read request of data k from DC d to DC j, and $c_{kdd}{}^{r}=0$ for time t.	R^+
$c^{r}_{*dj}(t)$	The cost of a search request from DC d to DC j, and $c^{r}_{*dd}=0$ for time t.	R^+
$c_{kdj}{}^{w}(t)$	The cost of a read request of data k from DC d to DC j, and $c_{kdd}{}^{w}=0$ for time t.	R^+
$C^{r}_{*kdj}(t)$	The cost per noti cation of data k from DC d to DC j for time t.	R^+

Table 3: Costs and variables time dependent.

$$\sum_{j=1, X_{kj}(t_i)=1}^{n} X'_{kd}(t_i) * (1 - X_{kd}(t_i)) * c^{r}_{*kdj}(t_i) \quad (21)$$

$$\sum_{j=1, j\neq d, X_{kj}(t_i)=1}^{n} (1 - X'_{kd}(t_i)) * X_{kd}(t_i) * c^{r}_{*kdj}(t_i) \quad (22)$$

The cost between time t_i and $t_i + t_{i+1}$, where i 2 (0; : : : ; m(with t_0=0 and t_m=T , is expressed by Equation 23 and the minimum number of replicas is represented by Inequality 24 for time t. The cost expressed in Equation 23 corresponds to the cost of the operations for the reads and writes of data k in DC d, `direct cost', plus the cost of propagating the updates, `update propagation cost', plus the cost of redirecting the reads to the less costly DC with a replica, `redirect reads cost', and finally plus the cost of searching for a DC with a replica of the data k to forward the request if the directly access DC does not have a replica of such data, `search cost', plus the cost of creating a new replica `new replica cost', plus the cost of removing a replica `remove replica cost', from time t_i to time t_{i+1}. The Equation 24 has a quadratic term that corresponds to the `search cost'. So the total cost up to a time T which it is divided into m sections is expressed in Equation 25.

$$\begin{aligned}
Cost_k(t_i, t_{i+1}) &= \sum_{d=1}^{n} r_{kd}(t_i, t_{i+1}) * w_{kd}(t_i, t_{i+1}) * c^{w}_{kd}(t_i) \\
&+ \sum_{j=1}^{n} w_{kd}(t_i, t_{i+1}) * c^{w}_{kdj}(t_i) * X_{kj}(t_i) \\
&+ (1 - X_{kd}(t_i)) * r_{kd}(t_i, t_{i+1}) * \min_{j \in n_k} (c^{r}_{kdj}(t_i)) \\
&+ (1 - X_{kd}(t_i)) * w_{kd}(t_i, t_{i+1}) * \min_{j \in n_k} (c^{r}_{kdj}(t_i)) \\
&+ \sum_{i=1, j\neq 1}^{n} (1 - X_{kd}(t_i)) * X_{kj}(t_i) * pkd(t_i, t_{i+1}) (c^{r}_{*dj}(t_i)) \\
&+ \sum_{j=1, X_{KJ}(t_i)=1}^{n} X'_{kd}(t_i) * (1 - X_{kd}(t - i)) * (c^{r}_{*kdj}(t_i) \\
&+ \sum_{i=1, j\neq 1}^{n} (1 - X_{kd}(t_i)) * X_{kj}(t_i) * pkd(t_i, t_{i+1}) (c^{r}_{*dj}(t_i)) \\
&+ \sum_{i=1, j\neq d, X_{kj}(t_i)=1}^{n} 1 - X'_{kd}(t_i)) * X_{kd}(t_i) * c^{r}_{*kdj}(t_i)
\end{aligned} \quad (23)$$

$$n_k^- \leq \sum_{d=1}^{n} X_{kd}(t) \leq n_k^+ \quad (24)$$

$$Cost_k = \sum_{i=1}^{m-1} Cost_k(t_i, t_i + 1) \quad (25)$$

The algorithm proposed does not incur the full `search cost' as only the first reply is processed and DCs with no replica of the data do not reply to the search request. Also all the reads received could be combined into one to the closes DC with a replica. Equation 23 could be used by another adaptive algorithm to set the parameters of this algorithm at running time in a dynamic way, instead of using fixed parameter values.

Comparisons with Full and No Replication

Following the proposed algorithm is compare with only one fixed copy approach, so no replication, and full replication, where there is a copy of the data in each of the DCs.

Table 4 shows the operations for each sequence diagram for the implementations considered. Looking at the storage requirements having more replicas uses more storage than when the data is located only in one DC. Regarding the operations executed, it can be seen that for reads the existence of the data in many DCs reduces the total number of operations to execute as it is more likely that the accessed DC to get the data from already have a replica of the data compared to when the data only exists in one DC. Whereas for writes having the data replicated in multiple DCs increases the total number of operations proportionally to the number of replicas. One technique to reduce the number of writes executed from a DC with a replica to the other DCs with a replica is to combine multiple write in one operation before forwarding it to other DCs. The algorithm proposed, represented in this Section as B, may behave as one fixed replica or as full replication by changing the value of some of its parameters, as presented below:

One fixed replica (A):

$$n_k^- = n_k^+ = 1, T_{kd}^+ = \infty, T_{kd}^- = -1, \Delta r_{kd} = 0, \Delta w_{kd} = 0, \Gamma_{kd} = 0,$$
$$\forall d \in \{1, ..., n\} and k \in \{1, ..., |D|\}$$

Full replication (C):

$$n_k^- = n_k^+ = n, T_{kd}^+ = 0, T_{kd}^- = -1, \Delta r_{kd} = 0, \Delta w_{kd} = 0, \Gamma_{kd} = 0,$$
$$\forall d \in \{1, ..., n\} and k \in \{1, ..., |D|\}$$

Sequence	One Fixed Replica	Adaptive Geo-replication	Full Replication
Storage	1 data	nk (data + info) + Δinfo	n data
First read on DC without replica	2 reads + (n−1) discoveries	2 reads + (n − 1) discoveries	1 reads
Read on DC without replica		2 reads	
Read on DC with replica	1 reads	1 reads	
First write on DC without replica	2 writes + (n−1) discoveries	(n_k+1) writes + (n−1) discoveries	n writes
Write on DC without replica		(n_k+1) writes	
Write on DC with replica	1 write	n_k writes	
Create data	1 write	(n_k+1) writes	n writes

Table 4: Maximum required operations by the different implementations for the specified sequences on write replication, n_k refers to the number of replicas before any due replication.

This means that systems where the number of read is significantly higher than the number of writes will benefit from replicating the data in multiple DCs, whereas systems with similar or higher number of write will benefit from low or no replication (only one replica). This will be true for write replication, but if the replication (update) is conducted on the reads then the increase in the number of replicas will be advantageous for writes and detrimental to reads. Future implementations of the proposed algorithm may take this into account so the replication type (on read or write) is also part of the adaptive mechanism. There are also many more requirements that have an important influence on the selection of the system configuration, i.e. number of DCs and replication. Some of the most common requirements are Scalability, Accessibility, Latency and Security.

Scalability:

It is the capability of a system to handle a growing amount of work, or its potential to be enlarged to accommodate that growth. (A) Does not provide scalability, where replication provides scalability. An analytical study which shows the scalability limits of full replication as updates have to be sent and executed at all replicated sites (symmetric processing) is provided in [15]. To reduce this, it can be used asymmetric processing where transactions are processed first at the originating site then collected and eventually propagated and applied to the other sites, which improves scalability. From the point of view of scalability, the processing power in a DC, when a write is received, is invested in processing the write and the updates. As the number of replicas increases, there is a point at which the increase on the number of DCs, so replicas, does not increase any more the system capacity. The main reason is that most of the system processing power is used in processing the updates. So (B) and (C) are preferable to (A) and at some point would be preferable to (C).

Availability/Accessibility:

The degree to which data can be accessed in a system (B) and (C) improve accessibility, where (A) provides a limited accessibility [34].

Latency:

It is a measure of the time it takes for some data to get to its destination across the network. Given that (B) and (C) provide replicas

in multiple DCs then the data can be accessed from any of those DCs and choosing the one with less latency will improve the latency seen by the customer, improve responsiveness. Even if in general (C) perform better regarding to this requirement, (B) will provide the same level of latency when the access pattern stabilises.

Security/Fault-tolerance:

Security normally is achieved by using recovery centres but it is also improved/achievable by the provision of replicas (redundancy) as if a DC suffer a catastrophic event where all data is lost if the data was replicated then the data can be retrieved from the other DC(s) where the data is also replicated [34,35]. So (B) and (C) would be preferred. (C) would not provide a significantly higher level on this requirement if the settings in (B) are appropriate, like the minimum number of DCs where to replicate. Other requirements could be considered, like the energy and storage, in which case, (A) improves on (B) and (C), but (B) improves on (C).

Conclusion

The presented algorithm is optimal in the sense that, when the access pattern stabilises, the total number of replicas required for the reads and writes is minimal with respect to the defined thresholds. The replication strategy can be further adapted to include other constraints and objectives, some of which have already been considered here.

For reads on a DC, the DC does not require to communicate any information to any of the other DCs. A read only needs to use resources in the DC where the read is initially requested from, so no extra network traffic is imposed on the systems. In the particular case where a read is executed on a DC, without a replica of the data, storage and processing power in the DC will be required and the request will be forwarded to its closest DC with a replica. This operation incurs in extra network traffic, but if it is repeated too often the extra network traffic will be removed by replicating the data in the requesting DC where the data was initially requested.

On a write the DC receiving the original request will (eventually) transmit it to the other DCs, which have a replica of the data, so no need to add extra network traffic as this is the normal approach. But extra data will be sent to the other DCs to notify them of the number of writes the changes refer to, which will depend of the type of Conflict-free Replicated Data Type (CRDT) approach used, i.e. op-based or state-based. In some cases not every write is transmitted to the other DCs, such is the case of the state-based approach, so it would be required to keep some track of the number of merged writes. Also the merging of the data should only be executed after the replication strength has been calculated and when it is still higher than zero, which will reduce unnecessary operations.

On data without any reads and writes on any DC, the number of replicas will be reduced as time passes by the temporal effect (and TTL), until the data is only replicated in the specified minimum number of DCs. This value may not be applied to data stored in recovery centres and data warehouses which may have their own TTL. The deletion of replicas requires strong consistency, but as it is not on any critical path, it will not incur latency increase. Also it would be desirable for data that expires to be copied into those data centres before it is removed from all the DCs. Adaptive replication t_s naturally into settings where eventual consistency is used. This approach uses simple and fast operations to adapt to the changing access pattern of the data. Furthermore, some or all of the parameters could be determined at run time by some other adaptive approach which has into account the cost function provided in Section 3.1.

The replication type (on read or on write) has an important influence in the costs of replicating data and should be incorporated into the strategy described above to even improve more its performance. The proposed algorithm also provides the potential to ad-just, reduce or extend the requirements taken into account to provide a good to the expectations and may also be used together with partial replication.

Acknowledgements

This work has been supported by the project SyncFree (co-financed by the European Commission through the grant agreement number 609551), and Trifork Leeds Ltd. (UK). The algorithm was previously presented in Asco, et al. [16] as part of the European research project SyncFree.

References

1. Tolle KM, Tansley D, Hey AJG (2011) The fourth paradigm: Data-intensive scientific discovery [point of view]. Proceedings of the IEEE 99: 1334-1337.
2. Cisco (2014) The Zettabyte Era: Trends and Analysis. Technical report, Cisco.
3. Chanthadavong A (2014) Internet of things to drive explosion of useful data: Emc. Technical report ZDNet.
4. Grace RK, Manimegalai R (2013) Dynamic replica placement and selection strategies in data grids- a com-prehensive survey. J Parallel Distrib Comput 74: 2099-2108.
5. Zin NM, Noraziah A, Fauzi AC, Herawan T (2012) Replication techniques in data grid environments. Asian Conference on Intelligent Information and Database Systems. Intelligent Information and Database Systems 7197: 549-559.
6. Naseera S, Murthy KVM (2009) Agent Based Replica Placement in a Data Grid Environement. Computational Intelligence, Communication Systems and Networks pp: 426-430.
7. Briquemont I, Bravo M, Li Z, Roy PV (2014) Optimising client-side geo-replication with partially replicated data structures. Master's thesis, Louvain-la-Neuve. Universite Catholique De Louvain.
8. Abad CL, Lu Y, Camp-bell RH (2011) Dare: Adaptive data replication for efficient cluster scheduling. In Proceedings of the 2011 IEEE International Conference on Cluster Computing, Washington, DC, USA. IEEE Computer Society pp: 159.
9. Venugopal S, Buyya R, Ramamohanarao K (2006) A taxonomy of data grids for dis-tributed data sharing, management, and processing. ACM Comput Surv 38: 3.
10. Serrano D, Patino-Martinez M, Jimenez-Peris R, Kemme B (2007) Boosting database replication scalability through partial replication and 1-copy-snapshot-isolation. In Dependable Computing, 2007. PRDC 2007. 13th Paci c Rim International Symposium pp: 290-297.
11. Goel S, Buyya R (2006) Data replication strategies in wide area distributed systems. In Robin G. Qiu, editor, Enterprise Service Computing: From Concept to Deployment pp: 31.
12. Pan S, Xiong L, Xu Z, Chong Y, Meng Q (2017) A dynamic replication management strategy in distributed GIS. Computers & Geosciences 112: 1-8.
13. Mansouri N (2016) Adaptive data replication strategy in cloud computing for performance improvement. Frontiers of Computer Science 10: 925-935.
14. Hamrouni T, Slimani S, Charrada FB (2016) A survey of dynamic replication and replica se-lection strategies based on data mining techniques in data grids. Engineering Applications of Artificial Intelligence 48: 140-158.

15. Asco A (2014) Adaptable location of replicas based on ant colony algorithm. Technical report, SyncFree WP1.
16. Asco A, Bieniusa A (2015) Adaptive strength geo-replication strategy. Proceedings of the First Workshop on Principles and Practice of Consistency for Distributed Data - PaPoC '15.
17. Liu G, Shen H, Chandler H (2016) Selective data replication for online social networks with distributed datacenters. IEEE Transactions on Parallel and Distributed Systems 27: 2377-2393.
18. Dong X, Li J, Wu Z, Zhang D, Xu J (2008) On Dynamic Replication Strategies in Data Service Grids. 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC) pp: 155.
19. Peluso PRS, Ruivo P (2015) Gmu: Genuine multiversion update-serializable partial data replication. IEEE Transactions on Parallel and Distributed Systems 27: 2911-2925.
20. Peluso S, Ruivo P, Romano P, Quaglia F, Rodrigues LS (2012) When Scalability Meets Consistency: Genuine Multiversion Update-Serializable Partial Data Replication. In In Proc. of International Conference on Distributed Systems.
21. Schiper N, Schmidt R, Pedone F (2006) Optimistic algorithms for partial database replication. Lecture Notes in Computer Science pp: 81-93.
22. Wang Z, Li T, Xiong N, Pan Y (2012) A novel dynamic network data replication scheme based on historical access record and proactive deletion. J Supercomput 62: 227-250.
23. Abdul-Wahid S, Andonie R, Lemley J, Schwing J, Widger J (2007) Adaptive distributed database replication through colonies of pogo ants. IEEE International Parallel and Distributed Processing Symposium pp: 1.
24. Loukopoulos T, Ahmad I (2004) Static and adaptive distributed data replication using genetic algorithms. J Parallel Distrib Comput 64: 1270-1285.
25. Peter M, Apers G (1988) Data allocation in distributed database systems. ACM Transactions on Database Systems 13: 263-304.
26. Wolfson O, Jajodia S, Huang Y (1997) An adaptive data replication algorithm. ACM Trans Database Syst 22: 255-314.
27. Wolfson O, Milo A (1991) The multicast pol-icy and its relationship to replicated data placement. ACM Trans. Database Syst 16: 181-205.
28. Wolfson O (1990) A distributed algorithm for adaptive replication of data. Technical report Department of Computer Science, Columbia University CCCS-057-90.
29. Dorigo M (1992) Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italy.
30. Jeason R, Ratnieks FLW, Jean-Louis D (2003) Pheromone trail decay rates on different substrates in the pharaohsant, monomorium pharaonis. Physiological Entomology 28: 192-198.
31. Jimenez-Peris R, Martnez MP, Alonso G, Kemme B (2003) Are quorums an alternative for data replication? ACM Trans Database Syst pp: 28: 257-294.
32. Paiva J, Ruivo P, Romano P, Rodrigues L (2013) Autoplacer: scalable self-tuning data placement in distributed key-value stores. In Proceedings of the 10th International Conference on Autonomic Computing, ICAC'13, SanJose, CA, USA.
33. Ladin R, Liskov B, Shrira L, Ghemawat S (1992) Providing high availability using lazy replication. ACM Trans Comput Syst 10: 360-391.
34. Guerraoui R, Andr Schiper A (1996) Fault-tolerance by replication in distributed systems. International Conference on Reliable Software Technologies, Reliable Software Technologies-Ada-Europe 96: 38-57.
35. Neumann JV (1956) Probabilistic Logics and Synthesis of Reliable Organisms from Unreliable Components. Automata Studies (AM-34) pp: 43-98.