Adaptable Location of Replicas Based on Ant Colony Algorithm

Amadeo Ascó

24^{th} September 2014

1 Introduction

The amount of data being processed in Data Centres (DCs) keeps growing at enormous rate [15, 4, 3]. Some of the areas where the amount of stored data already reach terabytes (TBs) and even petabytess (PBs) are data mining, particle physics, climate modelling, high energy physics and astrophysics, to site few, data which needs to be shared and analysed [10, 12, 13]. 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, [2, 1, 16]. An increase in the number of replications may result in a large bandwidth savings and lead to a reduction in user response time on reads. But keeping too many replicas of the data incurs in extra costs, such as extra replication traffic to keep all versions of the data coherent (writes), extra required storage and extra computational power [7].

This means that finding 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, [18, 2]. 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 constrain optimisation problem must have a very fast execution time.

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 [5]. In static replication the major drawback is their inability to adapt to changes in the access pattern of the data used by the users.

The proposed algorithm is base on Wolfson's algorithm, [18], which proposes an adaptive algorithm for replicated data between processors which takes into account the changes in the read-write pattern of the processors in the network.

Also the proposed algorithm is 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, [6].

It should be noted that the main propose 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 it 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.

2 Algorithm

The general idea of this algorithm is to decide without the need of human intervention where and when to replicate with the main objective of reducing the latency and network traffic (reduce usage of bandwidth).

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, but write operations 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 one of the DCs. This algorithm is further explained below together with a mathematical representation. The variables and constants used in the algorithm are summarised in Table 1.

Equation 1 represents the existence of a replica of data k in DC d, $c_{dk} = 1$, or its absence $c_{dk} = 0$. R_k is the number of replicas of data k, as expressed in Equation 2.

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

$$R_k = \sum_{d=1}^{|DC|} c_{dk} \tag{2}$$

F_{kd} represents the full strength of the replication of the data k in DC d, which it is expressed in Equation 3.

$$F_{kd} = \max(0, \min(L_k, r_{kd} * \Delta r_k + w_{kd} * \Delta w_k - \sum_{i=1, i \neq d}^{|DC|} w_{kd} * \Delta w_{kdi} - X_{kd} * \Gamma)) \tag{3}$$

Variable	Description	Type
DC	It is the set of all DCs. d identifies one of the DCs, $d \in \{1, \ldots, \lvert DC \rvert\}$. DC_d represents the DC d which holds some data, D_d.	$d \in \mathbb{N}^+$
D_{kd}	It represents the replica of data k in DC d, $k \in \{1, \ldots, \lvert D \rvert\}$.	$k \in \mathbb{N}^+$
r_{kd}	It is the number of reads for data k requested on DC d.	$r_{kd} \in \mathbb{N}_0$
Δr_k	It represents the strengthening of the replication in the DC used to execute one read.	$\Delta r_k \in \mathbb{R}^+$
w_{kd}	It is the number of writes for data k directly requested on DC d.	$w_{kd} \in \mathbb{N}_0$
Δw_k	It represents the strengthening of the replication in the DC directly used to execute a write.	$\Delta w_k \in \mathbb{R}^+$
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.	$w_{kdi} \in \mathbb{R}^+$
Γ	It is the decay of the replication strength with time. A simple example corresponds to a constant decay (τ) of the replication strength from the time the replica was created, $\Gamma = \Delta t * \tau$.	$\Gamma \in \mathbb{R}^+$
N_k	Minimum number of replicas of data k with $1 \leq N_k \leq \lvert DC \rvert$, default $N_k = 1$.	$N_k \in \mathbb{N}^+$
T_k^+	It is the replication strength required to start the replication of data k in a DC which does not currently contain a replica of the data.	$T_k^- \in \mathbb{R}^+$
T_k^-	It is the replication strength from where the replication of the data is removed k in a DC which contains currently a replica of the data, default $T_k^- = 0$.	$T_k^- \in \mathbb{R}^+$
L_k	The maximum replication strength for data k.	$L_k \in \mathbb{R}^+$
F_k	The replication strength for data k, $F_k \leq L_k$.	$F_k \in \mathbb{R}^0$
R_k	It is the number of replicas for data k, D_k, with $1 \leq N_k \leq R_k \leq \lvert DC \rvert$.	$R_k \in \mathbb{N}^+$
X_{kd}	It refers to the existence of a replica of data k in DC d, with value 1 if the replicas exists or 0 otherwise.	$X_{kd} \in (0, 1)$

Table 1: List of variables and constants.

Equation 3 shows that the replication strength for the data k in DC d is strengthened by the reads and writes requested through DC d, with intensities Δr_k and Δw_k respectively, and weakened by the writes requested through other DCs than DC d, with intensity Δw_{kdi}, and it is furthermore weakened by a temporal decay on the replication strength (last term in the equation). Where

Equation 3 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 so it does not incurred in any extra cost when implementing it. Similarly Γ will not be applied to DCs 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.

T_k^+ is the threshold of the replication strength of data k which determines when to keep an existing replica in a DC, as expressed in Equation 4.

$$F_{kd} \leq T_k^+, \nexists D_{kd}, t < t_0 \wedge F_{kd} > T_k^+, t_0 \Longrightarrow \exists D_{kd}, t_0 \ (D_{kd} \in DC_d) \quad (4)$$

T_k^- is the threshold of the replication strength of data k which determines when to remove an existing replica in a DC when the other constrains are still kept. i.e. minimum number of replicas, as expressed in Equation 5. Also a simple view of the thresholds is shown in Figure 1.

$$F_{kd} > T_k^- \wedge \exists D_{kd}, t < t_0 \wedge F_{kd} \leq T_k^-, t_0 \Longrightarrow \nexists D_{kd}, t_0 \ (D_{kd} \in DC_d) \quad (5)$$

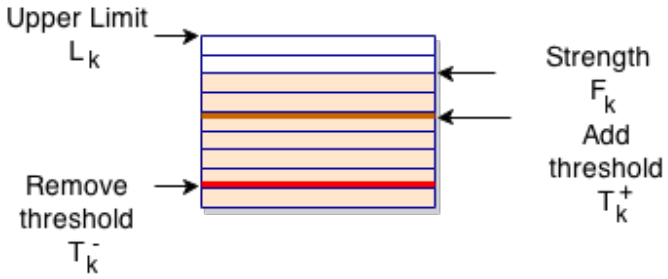

Figure 1: Constraints.

The Equations 6 and 7 show the cases where a data k is not replicated in a DC d. A replica in a DC, with a replication strength not bigger than the reduction replication strength ,T_k^-, will only be destroyed if the number of replicas is bigger than a minimum, N_k, as represented in Equation 6. 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_k^+ to create a new replica, as represented in Equation 7.

$$\exists D_{kd}, t < t_0, R_k > N_k, F_{kd} > T_k^- \Longrightarrow \nexists D_{kd}, t_0, F_{kd} \leq T_k^- \quad (6)$$

$$\nexists D_{kd}, t < t_0, F_{kd} \leq T_k^+ \Longrightarrow \nexists D_{kd}, t_0, F_{kd} \leq T_k^+ \quad (7)$$

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 correctly 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 until a replica is place in another DC or its replication is increased over zero.

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 $w_{kdi} \geq w_{kdj} + w_{kji}$, similarly if many intermediate nodes are used the decay will be at least the sum of the intermediate decays.

The effect of Γ is required to ensure that in absence of writes some replications will still vanish and if the reads are concentrated in a few DCs then the replicas in other DCs will be eventually removed. There is an extra requirement in the case that the data only exists in a DC, in which case, the temporal effect should be ignored, so the data exists at least in one DC.

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 a 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^+), 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 Table 2 with its flowchart presented in Figure 2, and the write ones are shown in Table 3 with its flowchart presented in Figure 3. The second figure in Table 2 may also have a notification for all DCs with a replica of the data to notify of the new replica which will be sent by the direct DC for when the threshold has been passed and a replica is created in the direct DC.

The creation of data will generate a replica in the directly accessed DC, as the number of replicas must be at least one ($M_k \geq 1$). If the minimum number

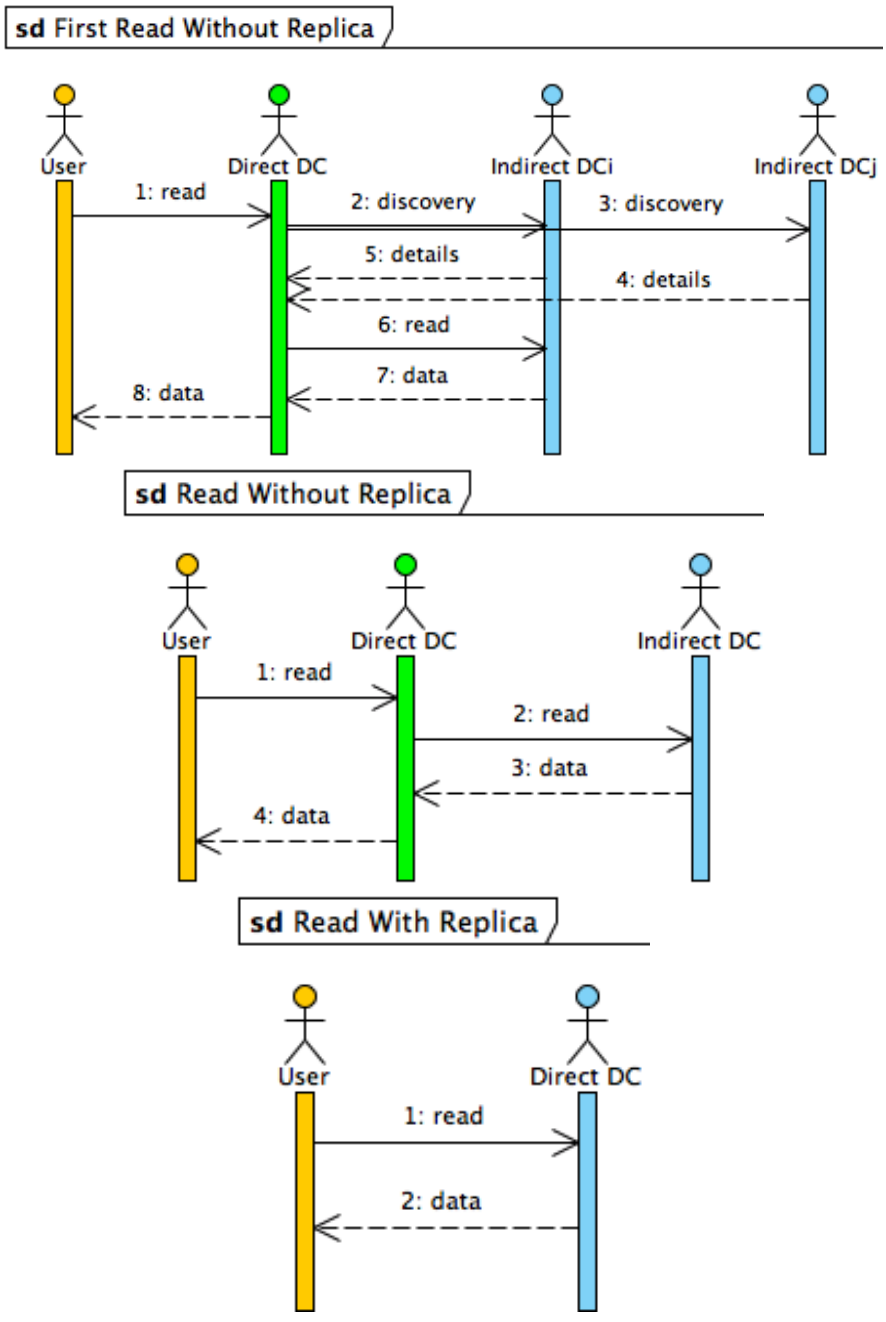

Table 2: Read Sequence Diagrams.

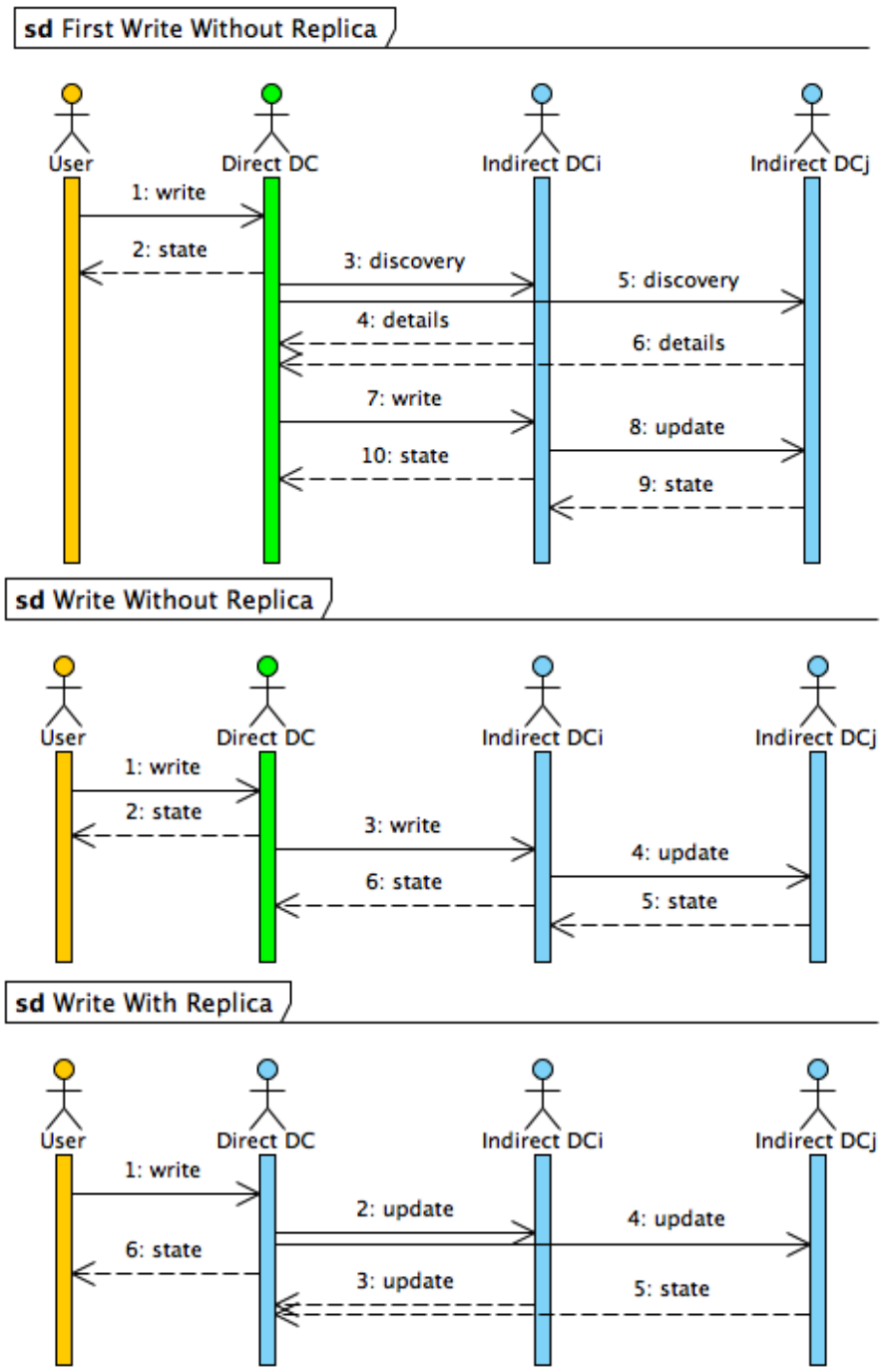

Table 3: Write Sequence Diagrams.

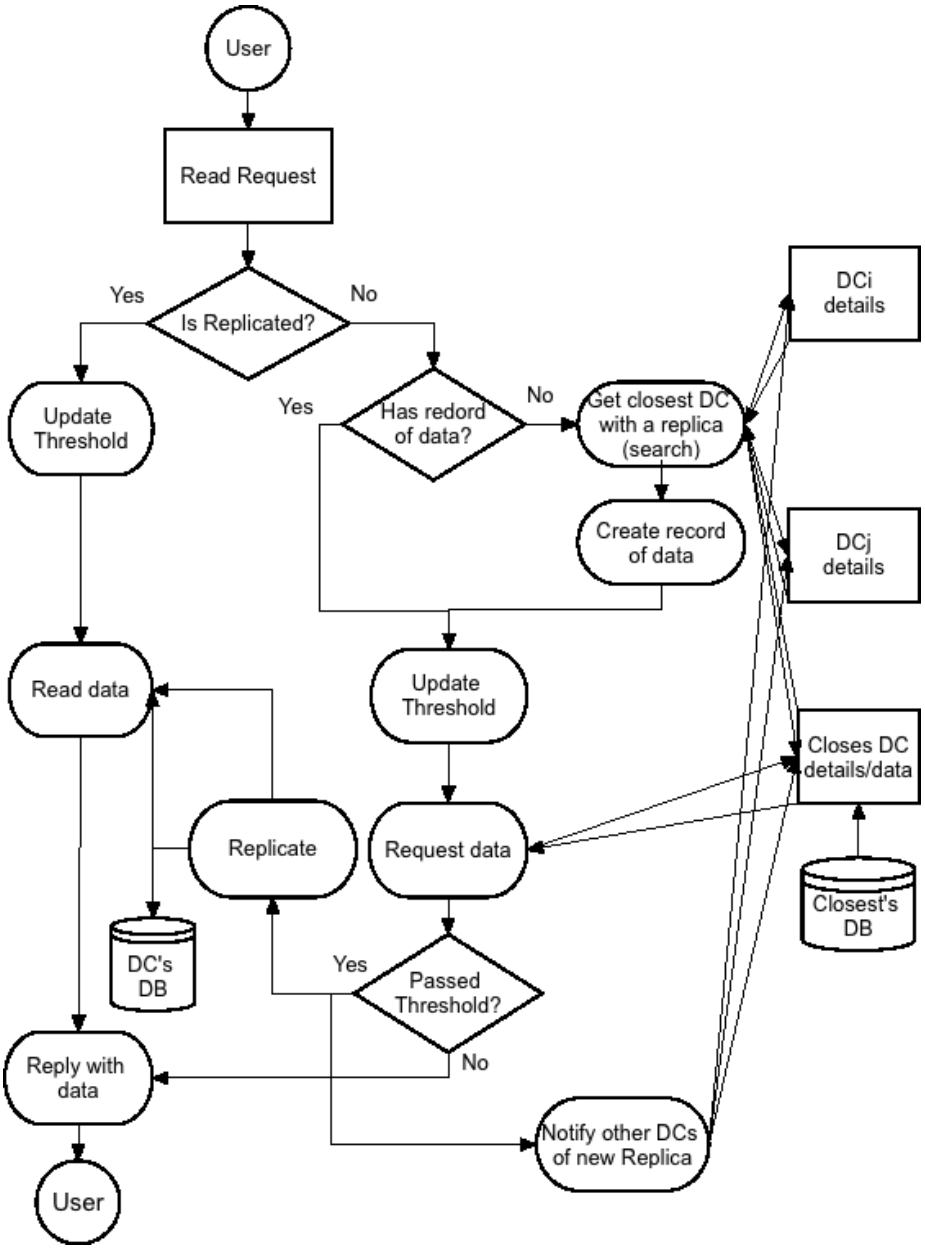

Figure 2: Flowchart for read requests.

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 to distribute created replicas to be implemented as a pluggable distribution

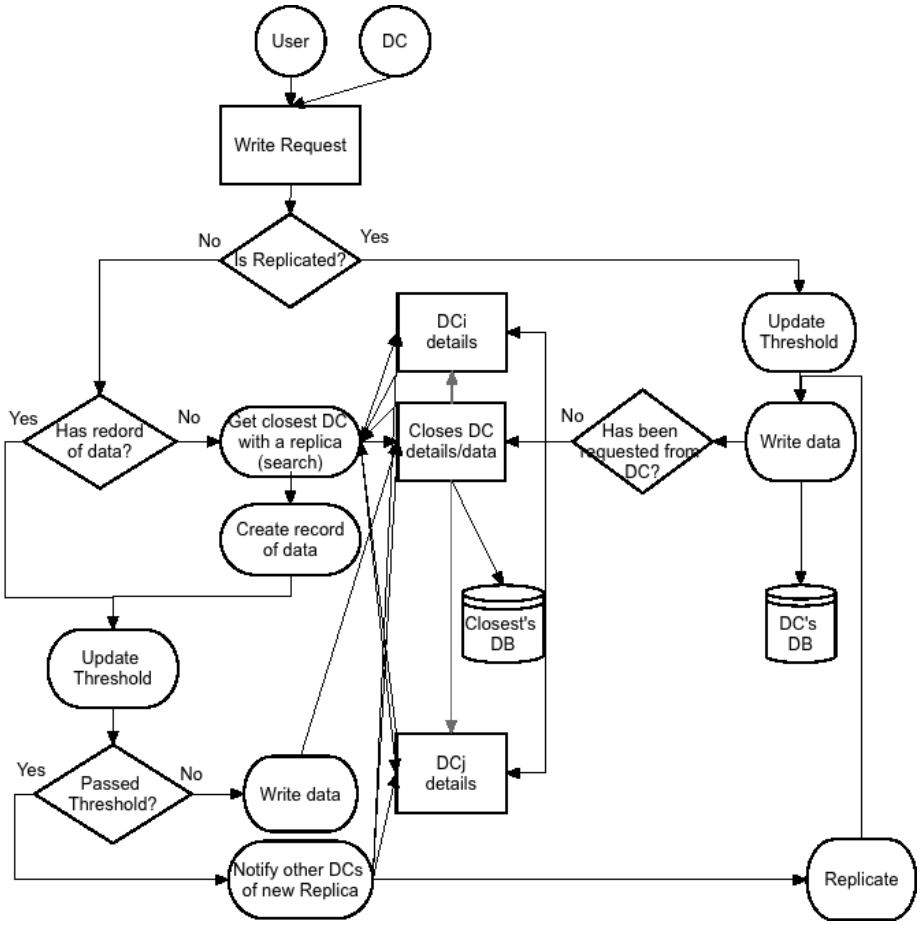

Figure 3: Flowchart for write requests.

approach, i.e. distribute replicas within the vicinity. The creation of the data will require to check if the data exists already, i.e. in a key value tuple that the key does not al;ready 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.

Some notes on parameters:

Given that the number of read is expected to be higher than the number of writes, and normally a read will be executed before a write, it may make sense for Δw_k to be smaller than Δr_k ($\Delta w_k < \Delta r_k$) so that more writes would be required to maintain or create a replica in a DC than when using reads.

Some of the parameters may be generalised even further by allowing them to depend on the DC the calculation executed in, e.g. Δr_{kd} instead of Δr_k

and Δw_{kd} instead of Δw_k. Even the terms $r_k * \Delta r_k$ and $w_k * \Delta w_k$ could be generalise to consider other factors like bandwidth used and current storage capacity available in the DC, or new terms could be added to take account of those new factors.

[14] uses a very simple representation of the available storage capacity, which is taking 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 8 provides the free storage available in DC d, s^d.

$$s^d = S^d - \sum_{k=1}^{|D|} S_k * X_{kd} \tag{8}$$

2.1 Mathematical Cost Representation

It is considered that there is a direct cost associated to the reads (c^r_{kd}) and writes (c^w_{kd}) executed in the system for the data k in the DC d. Also there is a cost associate to the reads (c^r_{kdj}) and writes (c^w_{kdj}, $c^w_{kdd} = 0$) between DCs d and j for data k, shown in Table 4.

Cost	Description	Type
c^r_{kd}	The cost of a read request of the data k to DC d.	$\mathbb{R}^+$
c^w_{kd}	The cost of a write request of the data k to DC d.	$\mathbb{R}^+$
c^r_{kdj}	The cost of a read request of data k from DC d to DC j, and $c^r_{kdd} = 0$.	$\mathbb{R}^+$
c^r_{*dj}	The cost of a search request from DC d to DC j, and $c^r_{*dd} = 0$.	$\mathbb{R}^+$
c^w_{kdj}	The cost of a read request of data k from DC d to DC j, and $c^w_{kdd} = 0$.	$\mathbb{R}^+$

Table 4: Costs.

The total cost is expressed by Equation 9 and the minimum number of replicas is represented by Inequality 10.

$$Cost_k = \sum_{d=1}^{|DC|} \left(\overbrace{r_{kd} * c^r_{kd} + w_{kd} * c^w_{kd}}^{\text{direct cost}} + \overbrace{\sum_{j=1}^{|DC|} w_{kd} * c^w_{kdj} * X_{kj}}^{\text{update propagation cost}} \right.$$
$$\left. + \underbrace{(1 - X_{kd}) * r_{kd} * \min_{j \in R_k}(c^r_{kdj})}_{\text{redirect reads cost}} + \underbrace{\sum_{d=1}^{|DC|} \sum_{j=1, j \neq d}^{|DC|} (1 - X_{kd}) * X_{kj} * c^r_{*dj}}_{\text{search cost}} \right) \tag{9}$$

$$\sum_{d=1}^{|DC|} X_{kd} \geq N_k \tag{10}$$

The cost expresses in Equation 9 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'. The Equation 9 has a quadratic term that corresponds to the 'search cost'.

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.

There is a cost incurred when a replica is placed in a new DC and when the data is removed from a DC which are not present in Equation 9. If $X^{'}_{kd}$ is the new replication state after all the current operations have been executed with a cost per notification of c^r_{*kdj} then the cost of a new replica can be represented as in Equation 11, and the cost of removing a replica from a DC, with cost per notification of c^r_{*kdj}, can be represented as in Equation 12.

$$\sum_{j=1,X_{kj}=1}^{|DC|} X^{'}_{kd} * (1 - X_{kd}) * c^r_{*kdj} \tag{11}$$

$$\sum_{j=1,j\neq d,X_{kj}=1}^{|DC|} (1 - X^{'}_{kd}) * X_{kd} * c^r_{*kdj} \tag{12}$$

The extended cost is shown in Equation 13.

$$\begin{aligned} Cost_k = \sum_{d=1}^{|DC|} \Bigg(& \overbrace{r_{kd} * c^r_{kd} + w_{kd} * c^w_{kd}}^{\text{direct cost}} + \overbrace{\sum_{j=1}^{|DC|} w_{kd} * c^w_{kdj} * X_{kj}}^{\text{update propagation cost}} \\ & + \underbrace{(1 - X_{kd}) * r_{kd} * \min_{j \in R_k}(c^r_{kdj})}_{\text{redirect reads cost}} + \underbrace{\sum_{d=1}^{|DC|} \sum_{j=1,j\neq d}^{|DC|} (1 - X_{kd}) * X_{kj} * c^r_{*dj}}_{\text{search cost}} \\ & + \underbrace{\sum_{j=1,X_{kj}=1}^{|DC|} X^{'}_{kd} * (1 - X_{kd}) * c^r_{*kdj}}_{\text{new replica cost}} + \underbrace{\sum_{j=1,j\neq d,X_{kj}=1}^{|DC|} (1 - X^{'}_{kd}) * X_{kd} * c^r_{*kdj}}_{\text{remove replica cost}} \Bigg) \end{aligned} \tag{13}$$

Equation 13 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.

3 Some Comparisons (draft)

Here the following implementations are compared:

A. Only one fixed replica, so no replication as such.

B. Proposed algorithm, adaptive location of replicas.

C. Full replication, so a replica in all DCs.

In the proposed algorithm, second implementation, by changing the minimum number of replicas we moved from the first case to the last one.

Table 5 shows the operations for each sequence diagram for the implementations considered. Looking at the storage requirements having replicates uses more storage than when the data is located only in one DC. Regarding the operations executed can be seen that for reads the existence of the data in many DCs reduces the total number of operation to execute as t is more likely that the accessed DC to get the data 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 technic to reduce the number of write executed from a DC with a replica to the other DCs with a replica is to combine multiple write in one before forwarding it to other DCs.

The algorithm proposed may behave as one fixed replica or as full replication by changing the value of some of its parameters, see below:

- **One fixed replica**: $N_k = 1, T_k^+ = \infty, T_k^-$.
- **Full replication**: $N_k = |DC|$.

This means that systems where the number of read is significantly higher that the number of writes will benefit from replicating the data in multiple DCs, whereas systems with similar of higher number of write will benefit from low or no replication (only one replica). But there are 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**: (A) does not provide scalability, where replication provides scalability. [9] provides 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). To reduce the 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

Sequence	**One fixed replica (A)**	**Algorithm (B)**	**Full replica (C)**
Storage	$1*$ data	R_k* (data + info) + Δinfo	$\lvert DC\rvert*$ data
First read without replica	$2*$ reads $+(\lvert DC\rvert-1)*$ discoveries	$2*$ reads $+(\lvert DC\rvert-1)*$ discoveries	$1*$ reads
Read without replica	$2*$ reads $+(\lvert DC\rvert-1)*$ discoveries	$2*$ reads	
Read with replica	$1*$ reads	$1*$ reads	
First write without replica	$2*$ writes $+(N_k-1)*$ discoveries	$(R_k+1)*$ writes $+(N_k-1)*$ discoveries	$\lvert DC\rvert*$ writes
Write without replica	$2*$ writes $+(N_k-1)*$ discoveries	$(R_k+1)*$ writes	
Write with replica	$1*$ write	R_k* writes	
Create	$1*$write	N_k* writes	$\lvert DC\rvert*$ writes

Table 5: Required operations by the different implementations for the specified sequences, $1 \leq N_k \leq R_k \leq |DC|$.

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.

- **Availability/Accessibility**: (B) and (C) improve accessibility, where (A) provides a limited accessibility, [11].
- **Latency**: 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.
- **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, [8, 17].

4 Summary

This algorithm implies:

1. 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 to forwarded the request 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.

2. 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-base approach, so it would be required to keep some track of the number of merged writes.

Also the merging the data should only be executed after the replication strength has been calculating and it is still higher than zero, which will reduce unnecessary operations.

3. On data without reads and writes concentrated on one DC, the number of replicas will be reduced as time passes by the temporal effect (Γ), until the data is only replicated in one DC.

This approach requires simple and fast operations to adapt to the changing conditions of the accessed 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 2.1.

5 Acknowledgements

This work is part of the European research project SyncFree.

References

[1] Cristina L. Abad, Yi Lu, and Roy H. Campbell. Dare: Adaptive data replication for efficient cluster scheduling. In *Proceedings of the 2011 IEEE International Conference on Cluster Computing*, CLUSTER '11, pages 159–168, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4516-5. doi: 10.1109/CLUSTER.2011.26. URL https://wiki.engr.illinois.edu/download/attachments/194990492/cluster11.pdf.

[2] Iwan Briquemont. Optimising client-side geo-replication with partially replicated data structures. Master's thesis, Louvain-la-Neuve, September 2014. URL http://www.info.ucl.ac.be/~pvr/MemoireIwanBriquemont.pdf.

[3] Aimee Chanthadavong. Internet of things to drive explosion of useful data: Emc. Technical report, ZDNet, April 2014. URL http://www.zdnet.com/internet-of-things-to-drive-explosion-of-useful-data-emc-7000028376.

[4] Cisco. The zettabyte era-trends and analysis. Technical report, Cisco, June 2014. URL http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.pdf.

[5] Xiaohua Dong, Ji Li, Zhongfu Wu, Dacheng Zhang, and Jie Xu. On dynamic replication strategies in data service grids. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 155–161, May 2008. doi: 10.1109/ISORC.2008.66.

[6] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[7] Sushant Goel and Rajkumar Buyya. Data replication strategies in wide area distributed systems. In Robin G. Qiu, editor, *Enterprise Service Computing: From Concept to Deployment*, pages 211–241. Idea Group Inc, 2006. URL http://www.cloudbus.org/papers/DataReplicationInDSChapter2006.pdf.

[8] Rachid Guerraoui and André Schiper. Fault-tolerance by replication in distributed systems. In Alfred Strohmeier, editor, *Reliable Software Technologies – Ada Europe 96*, volume 1088 of *Lecture Notes in Computer Science*, pages 38–57. Springer Berlin Heidelberg, 1996. ISBN 978-3-540-61317-6. doi: 10.1007/BFb0013477. URL http://dx.doi.org/10.1007/BFb0013477.

[9] Ricardo Jiménez-Peris, M. Patiño Martínez, Gustavo Alonso, and Bettina Kemme. Are quorums an alternative for data replication? *ACM Trans. Database Syst.*, 28(3):257–294, September 2003. ISSN 0362-5915. doi: 10.1145/937598.937601. URL http://lsd.ls.fi.upm.es/lsd/papers/2003/tods03.pdf.

[10] R. Kingsy Grace and R. Manimegalai. Dynamic replica placement and selection strategies in data grids- a comprehensive survey. *J. Parallel Distrib. Comput.*, 74(2):2099–2108, February 2013. ISSN 0743-7315. doi: 10.1016/j.jpdc.2013.10.009. URL http://dx.doi.org/10.1016/j.jpdc.2013.10.009.

[11] Rivka Ladin, Barbara Liskov, Liuba Shrira, and Sanjay Ghemawat. Providing high availability using lazy replication. *ACM Trans. Comput. Syst.*, 10 (4):360–391, November 1992. ISSN 0734-2071. doi: 10.1145/138873.138877. URL http://doi.acm.org/10.1145/138873.138877.

[12] Noriyani Mohd. Zin, A. Noraziah, AinulAzila Che Fauzi, and Tutut Herawan. Replication techniques in data grid environments. In Jeng-Shyang Pan, Shyi-Ming Chen, and NgocThanh Nguyen, editors, *Intelligent Information and Database Systems*, volume 7197 of *Lecture Notes in Computer Science*, pages 549–559. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-28489-2. doi: 10.1007/978-3-642-28490-8_57. URL http://dx.doi.org/10.1007/978-3-642-28490-8_57.

[13] Shaik Naseera and K.V. Madhu Murthy. Agent based replica placement in a data grid environement. *Computational Intelligence, Communication Systems and Networks, International Conference on*, 0:426–430, 2009. doi: http://doi.ieeecomputersociety.org/10.1109/CICSYN.2009.77.

[14] João Paiva, Pedro Ruivo, Paolo Romano, and Luís Rodrigues. AutoPlacer: scalable self-tuning data placement in distributed key-value stores. In *Proceedings of the 10th International Conference on Autonomic Computing*, ICAC'13, San Jose, CA, USA, June 2013. USENIX. URL http://www.usenix.org/conference/icac13/autoplacer-scalable-self-tuning-data-placement-distributed-key-value-stores.

[15] K.M. Tolle, D. Tansley, and A.J.G. Hey. The fourth paradigm: Data-intensive scientific discovery [point of view]. *Proceedings of the IEEE*, 99(8):1334–1337, Aug 2011. ISSN 0018-9219. doi: 10.1109/JPROC.2011.2155130. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5958175.

[16] Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1), June 2006. ISSN 0360-0300. doi: 10.1145/1132952.1132955. URL http://www.cloudbus.org/reports/DataGridTaxonomy.pdf.

[17] J. von Neumann. *Automata Studies*, chapter Probabilistic Logics and Synthesis of Reliable Organisms from Unreliable Components, pages 43–98. Princeton University Press, 1956. URL https://ece.uwaterloo.ca/~ssundara/courses/prob_logics.pdf.

[18] Ouri Wolfson. A distributed algorithm for adaptive replication of data. Technical report, Department of Computer Science, Columbia University, 1990. URL http://hdl.handle.net/10022/AC:P:21285.