Adaptable Location of Replicas Based on Ant Colony Algorithm

Amadeo Ascó

24th September 2014

1 Introduction

The amount of data being processed in Data Centres (DCs) keeps growing at enormous rate [? ? ?]. 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 [? ? ?]. 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, [? ? ?]. 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 [?].

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, [? ?]. 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 [?]. 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, [?], 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, [?].

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 **??**.

Equation **??** 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 **??**.

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

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

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

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

Variable	**Description**	**Type**
DC	It is the set of all DCs. d identifies one of the DCs, $d \in \{1, \ldots, \vert DC\vert\}$. 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, $k \in \{1, \ldots, \vert D\vert\}$.	$k \in \mathbb{N}^+$
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.	$w_{kdi} \in \mathbb{R}^+$
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 $T_k^- = 0$.	$T_k^- \in \mathbb{R}^+$
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, $\Gamma = \Delta t * \tau$.	$\Gamma \in \mathbb{R}^+$
R_k	It is the number of replicas for data k, D_k.	$R_k \in \mathbb{N}_0$
M_k	Minimum number of replicas of data k, default $M_k = 1$.	$M_k \in \mathbb{N}_0$
F_k	The replication strength for data k, $F_k \leq L_k$.	$F_k \in \mathbb{R}^+$
L_k	The maximum replication strength for data k, $F_k \leq L_k$.	$L_k \in \mathbb{R}^+$

Table 1: List of variables and constants.

Equation **??** 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 **??** 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 **??**.

$$\exists D_{kd},\, t_0 \ (D_{kd} \in DC_d) \text{ if } F_{kd} \leq T_k^+ \ \text{ and } \nexists D_{kd},\, t < t_0 \text{ and } F_{kd} > T_k^+ \text{ at } t_0 \tag{4}$$

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 **??**. Also a simple view of the thresholds is shown in Figure **??**.

$$\nexists D_{kd},\, t_0 \ (D_{kd} \in DC_d) \text{ if } F_{kd} > T_k^- \ \text{ and } \exists D_{kd},\, t < t_0 \text{ and } F_{kd} \leq T_k^- \text{ at } t_0 \tag{5}$$

Figure 1: Limit constraints.

The Equations **??** and **??** show the cases where a data k is not replicated in a DC d. A replica in a DC, with no replication strength, will only be destroyed if the number of replicas is bigger than a minimum, M_k, 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 **??**.

$$\nexists D_{kd},\, t > t_0 \text{ if } \exists D_{kd} \text{ for } t < t_0 \text{ and } F_{kd} \leq 0,\, R_k > M_k,\, t_0 \tag{6}$$

$$\nexists D_{kd},\, t_0 \text{ if } \nexists D_{kd},\, t' \leq t_0 \text{ and } F_{kd} \leq T_k^+,\, t_0 \tag{7}$$

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, 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 **??** with its flowchart presented in Figure **??**, and the write ones are shown in Table **??**

with its flowchart presented in Figure ??. The second figure in Table ?? 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.

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 use and current storage capacity available in the DC, or new terms could be added to take account of those new factors.

3 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

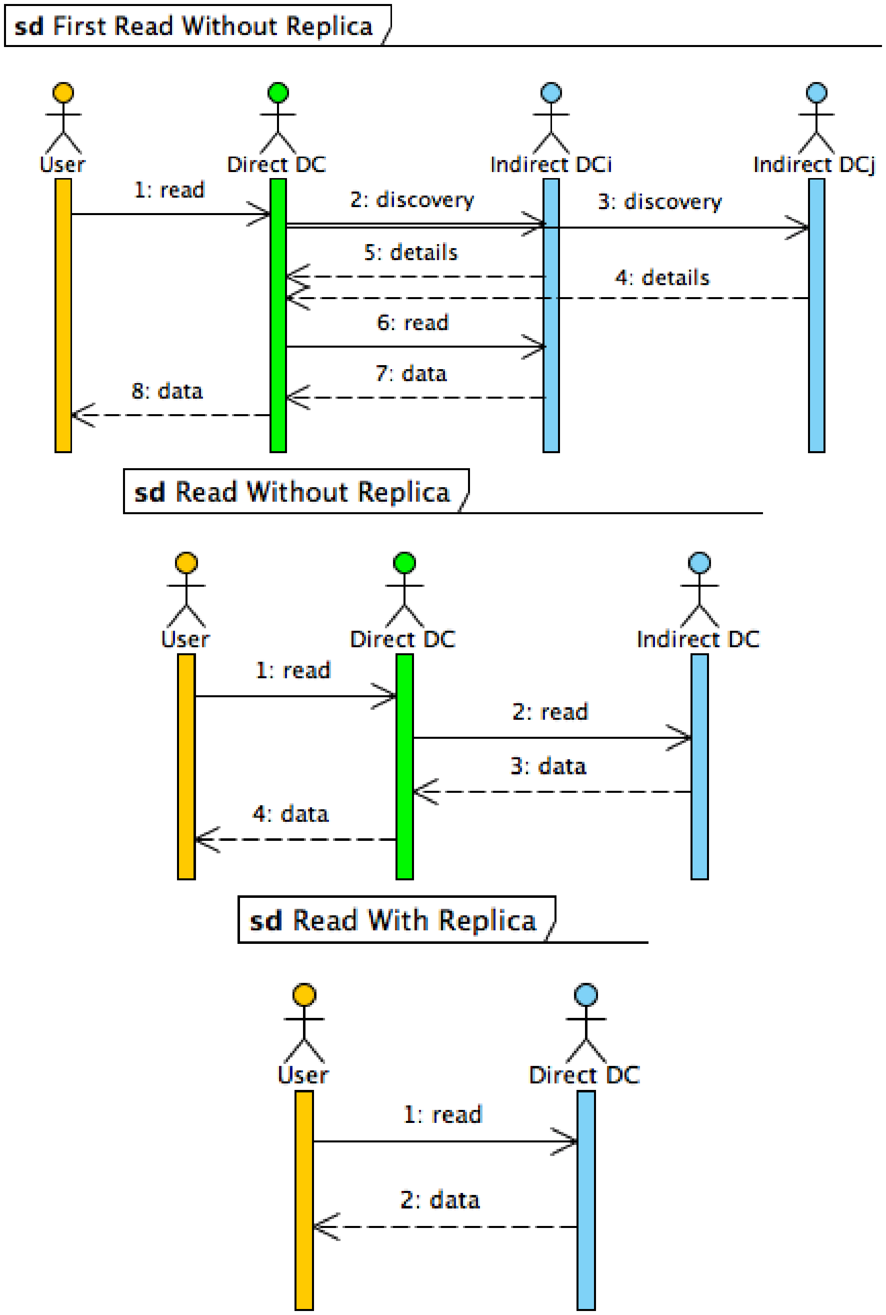

Table 2: Read Sequence Diagrams.

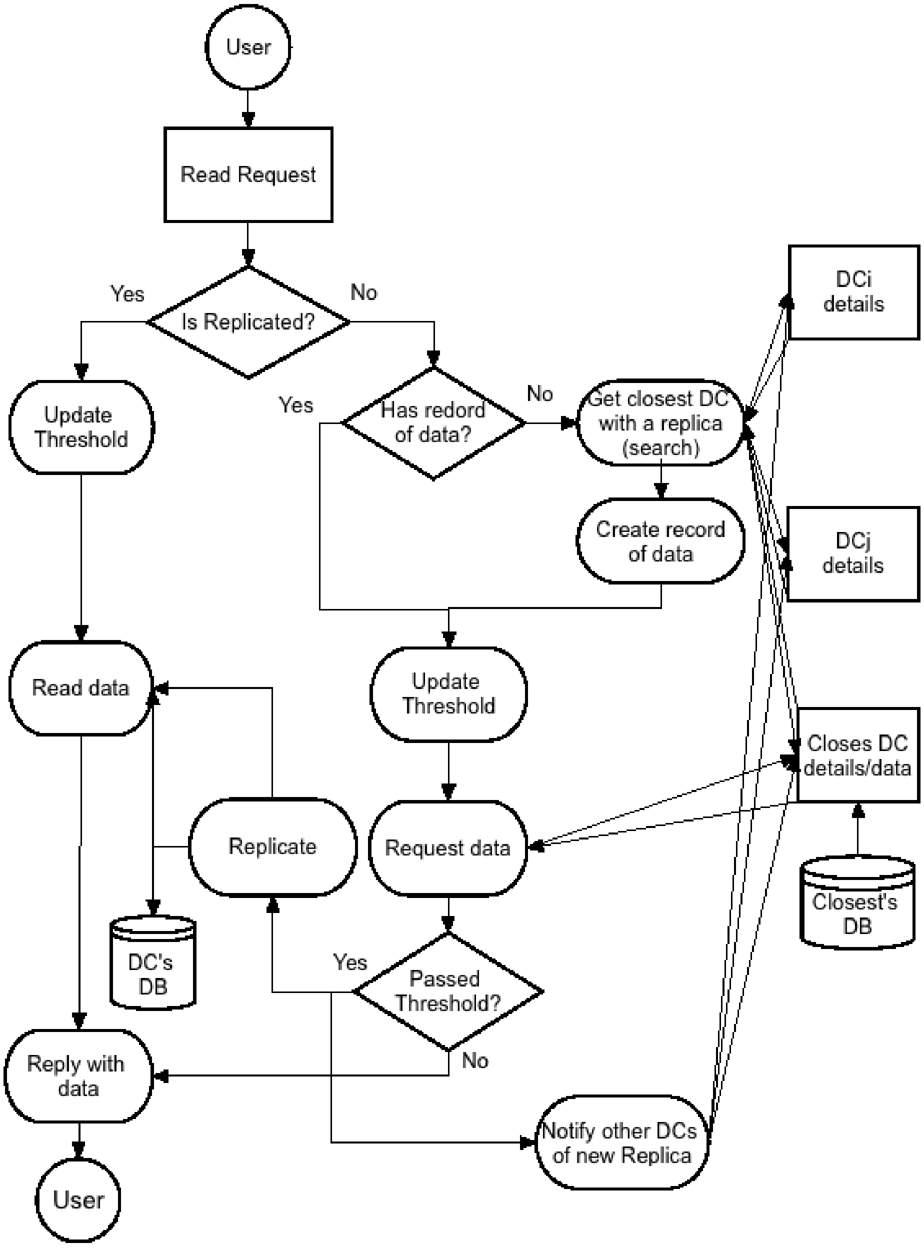

Figure 2: Flowchart for read requests.

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.

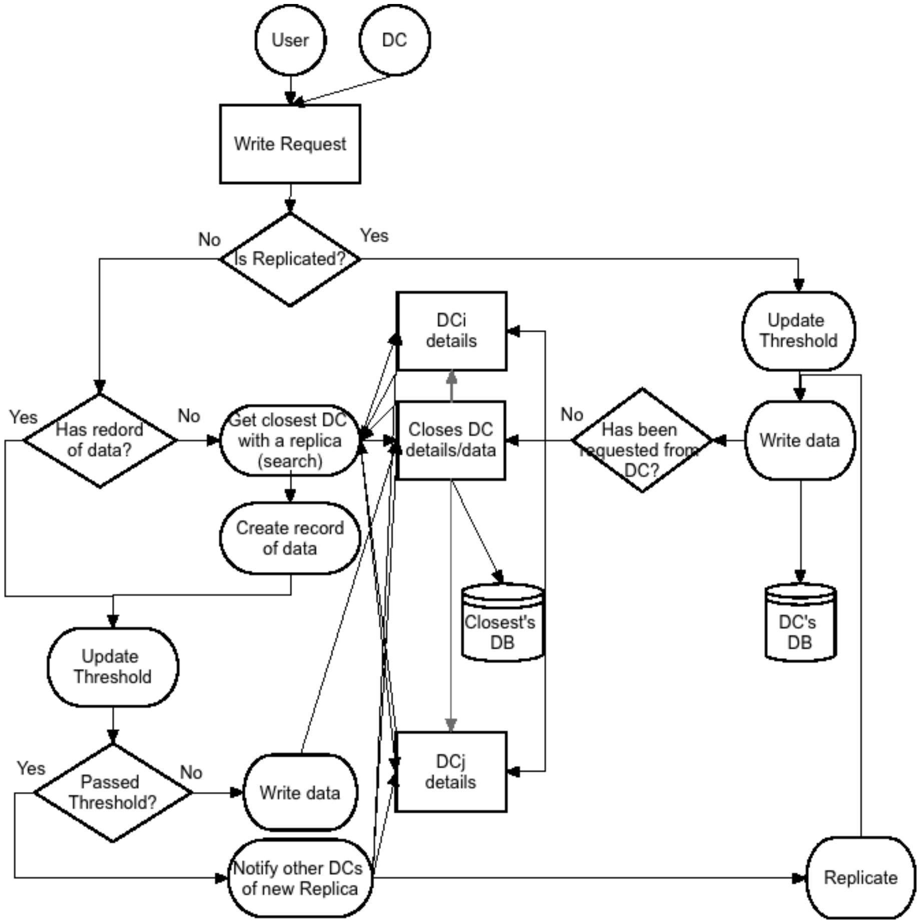

Figure 3: Flowchart for write requests.

4 Acknowledgements

This work is part of the European research project SyncFree.

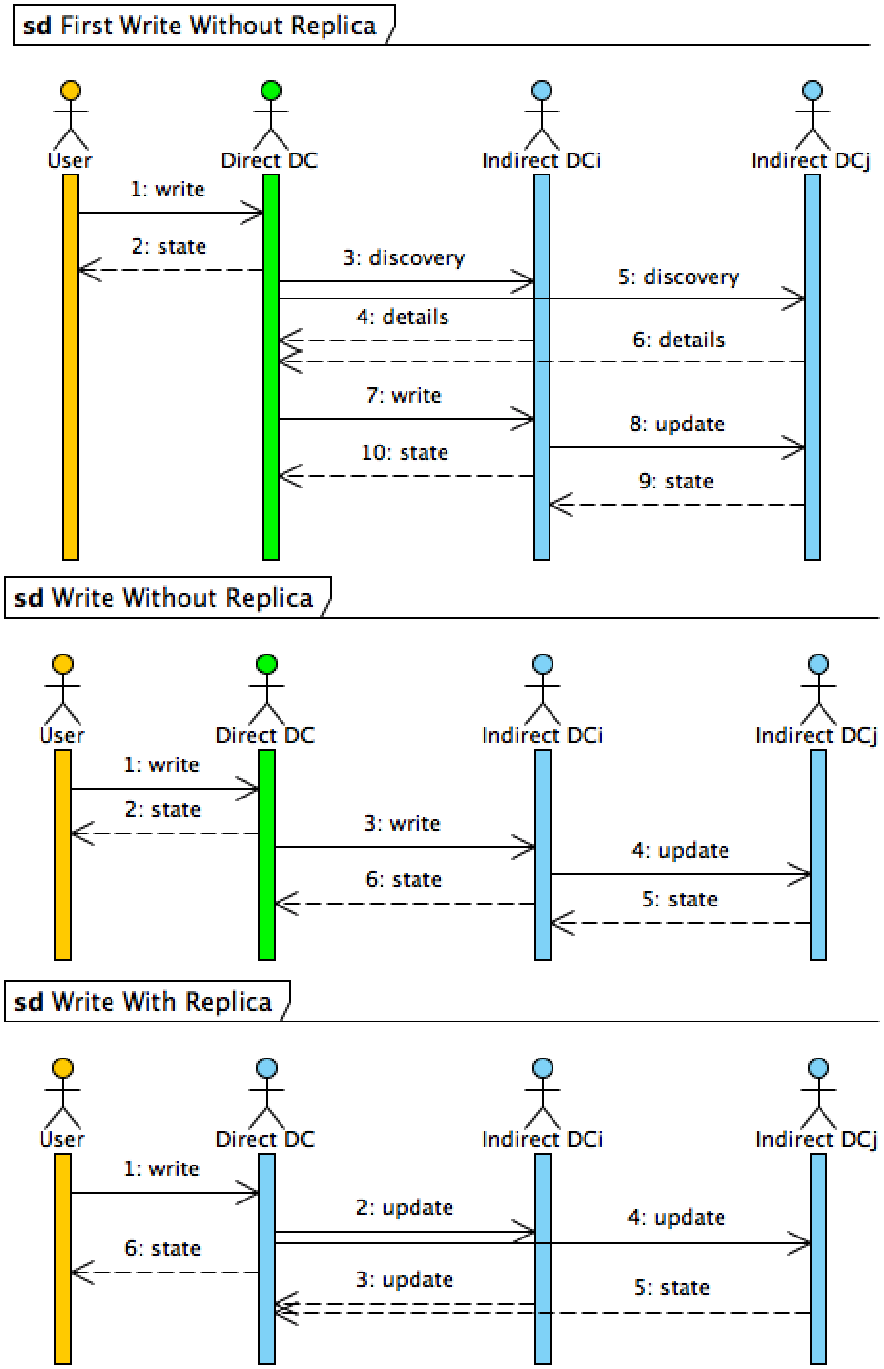

Table 3: Write Sequence Diagrams.