International Journal of Cloud Computing

ISSN online: 2043-9997 - ISSN print: 2043-9989
https://www.inderscience.com/ijcc

Data consistency protocol for multicloud systems

Olga A. Kozina, Volodymyr I. Panchenko, Oleksii V. Kolomiitsev, Viktoriya V. Usik, Nataliia K. Stratiienko, Ludmila V. Safoshkina, Yurii F. Kucherenko

DOI: 10.1504/IJCC.2024.10061839

Article History:

Received:	26 April 2021
Last revised:	28 July 2021
Accepted:	14 August 2021
Published online:	26 January 2024

Data consistency protocol for multicloud systems

Olga A. Kozina*, Volodymyr I. Panchenko, Oleksii V. Kolomiitsev and Viktoriya V. Usik

Department of Computer Engineering and Programming,
National Technical University 'Kharkiv Polytechnic Institute'
(NTU 'KhPI'),
Kharkiv, Ukraine
Email: kozina4common@gmail.com
Email: vladimir.panchenko@gmail.com
Email: Kolomijcev.O.hnups@gmail.com
Email: usik.viktory@gmail.com
*Corresponding author

Nataliia K. Stratiienko

Software Engineering and Management
Information Technologies Department,
National Technical University 'Kharkiv Polytechnic Institute'
(NTU 'KhPI'),
Kharkiv, Ukraine
Email: strana.snk@gmail.com

Ludmila V. Safoshkina

Associate Research Centre,
National Academy of the National Guard of Ukraine,
Kharkiv, Ukraine
Email: saf0705@ukr.net

Yurii F. Kucherenko

Associate Scientific Centre of Air Force,
Kharkiv National Air Force University (KhNUPS),
Kharkiv, Ukraine
Email: KucherenkoYF@gmail.com

Abstract: Using the resources of several cloud service providers (CSPs) to store, serve, and access user's data can improve availability and reduce latency. However, the management of multicloud systems also poses an important challenge of how to guarantee that requests from any region to geo-distributed replicas of the database will content equivalent actual data, which is considered in this paper. The existing taxonomy of data consistency models allows choosing the required level of data consistency in cloud systems, however, the implementation of consistency protocols for multicloud systems requires a

reasonable choice of middleware architecture and compromise decisions between response time and other constraints required by clients requirements. We propose consistency protocol based on the geo-distributed architecture of multicloud middleware to assign the ordering of numbers in a global sequence for incoming writing.

Keywords: data consistency; consistency protocol; consistency model; multi clouds; cloud service providers; multicloud systems; latency; geo-distributed database; response time; middleware architecture.

Reference to this paper should be made as follows: Kozina, O.A., Panchenko, V.I., Kolomiitsev, O.V., Usik, V.V., Stratiienko, N.K., Safoshkina, L.V. and Kucherenko, Y.F. (2024) 'Data consistency protocol for multicloud systems', *Int. J. Cloud Computing*, Vol. 13, No. 1, pp.42–61.

Biographical notes: Olga A. Kozina is a Professor at the National Technical University 'Kharkiv Polytechnic Institute', Kharkiv, Ukraine. She received her PhD in Technical Sciences from Kharkiv National University of Radio Electronics, Ukraine in 2001. She has experience in area of computer science, system modelling and optimisation. Her research interests include cloud computing, biomedical information processing technologies, networking, cloud computing, distributed systems.

Volodymyr I. Panchenko is a Senior Lecturer at the National Technical University 'Kharkiv Polytechnic Institute', Kharkiv, Ukraine. He has experience in area of computer science, system programming and optimisation. His research interests include operating systems, software engineering, networking, web servers and distributed systems.

Oleksii V. Kolomiitsev is a Professor at the National Technical University 'Kharkiv Polytechnic Institute', Kharkiv, Ukraine. He received his Doctor of Engineering Sciences from Ivan Kozhedub Kharkiv National Air Force University, Ukraine, in 2017. He has experience in area of computer science, system modelling and optimisation the parameters of the multifunctional information-measuring system. His research interests include optimisation of the multifunctional information-measuring system, distributed systems, cloud computing.

Viktoriya V. Usik is a Professor at the National Technical University 'Kharkiv Polytechnic Institute', Kharkiv, Ukraine. She received her PhD in Technical Sciences from Kharkiv National University of Radio Electronics, Ukraine in 2005. She has experience in the field of informatics, modelling of acoustic fields in confined spaces. Her research interests include architectural acoustics, cloud computing, educational technologies, biomedical information processing technologies.

Nataliia K. Stratiienko is a Professor at the Software Engineering and Management Information Technologies Department in National Technical University 'Kharkiv Polytechnic Institute', Kharkiv, Ukraine. She received her PhD in Technical Sciences from National Technical University 'Kharkiv Polytechnic Institute', Ukraine in 2000. She has experience in area of computer science and IT-project management. Her research interests include analysis of algorithms; cloud computing, operations research, and information technologies.

Ludmila V. Safoshkina is a Senior Researcher at Associate Research Center of the National Academy of the National Guard of Ukraine, Kharkiv, Ukraine. She received her PhD in Technical Sciences, Kharkiv, Ukraine, in 2014. Her research interests include operations research, optimisation theory, information technologies, cloud computing.

Yurii F. Kucherenko is a Leading research at Associate Scientific Center of Air Force Ivan Kozhedub Kharkiv National Air Force University, Kharkiv, Ukraine, in 1993. He has experience in area of construction automated control systems and information systems. He has authored or co-authored more than 80 academic papers. He has experience in area of computer science, operations research and specialised software. Him research interests include optimisation theory, networks, and information technologies, cloud computing.

1 Introduction

Multicloud systems have many advantages (Rafique et al., 2015; Abualkishik et al., 2020; Kozina and Panchenko, 2018). Each organisation has a justified set of QoS requirements for its customers and expects to achieve them using multiple CSP resources (Ibrahim et al., 2016). Some organisations are using multicloud to obtain a high-power high-bandwidth computing environment, as discussed in Koulouzis et al. (2020). For other organisations, multicloud is primarily an opportunity to receive a reliable service for storing, processing and geo-distributed data access for their customers, as shown in Gudeme et al. (2019), García-Dorado (2015) and Jambunathan and Yoganathan (2018). Still, others require some combination of a powerful computing environment and data storage/processing. However, today, choosing the optimal architecture and organising effective management of multicloud systems remains a challenge with highly specialised ad hoc solutions (Bittencourt and Calheiros, 2017).

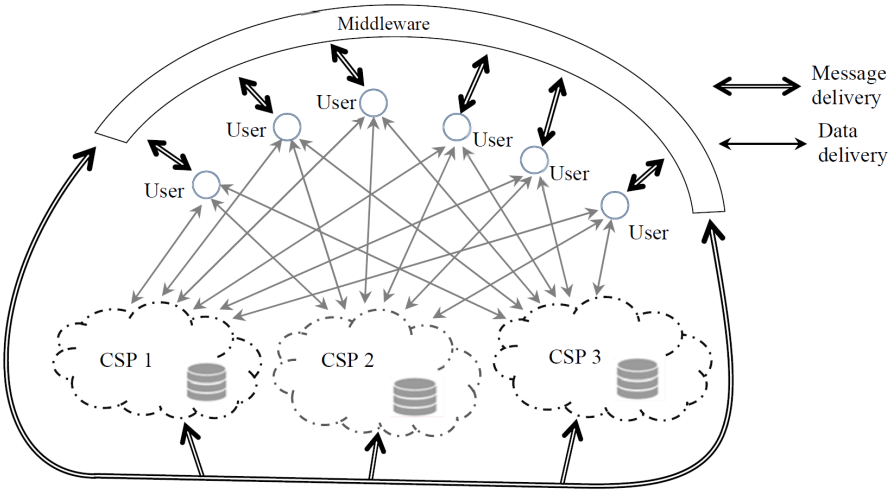
In this paper, we discuss the issues of ensuring data consistency in multicloud systems designed primarily for storing and accessing data. A prime example of this type of system is healthcare record systems (Chandavale et al., 2019; Rezaeibagha and Mu, 2016). The problem of data consistency does not arise when in all data storage and processing centres the data itself, which the user can access at any moment, is identical, in other words, there is no situation when at the same time in some data centres (DC) there is new data, while others do not yet. How to organise replicas of data within cloud resources is a task for each CSP that can be addressed in different ways. But the task of replicating data between several CSPs should be solved by middleware developers (Viotti and Vukolić, 2016), which is directly related to the choice of both the architecture and the multicloud system consistency protocol.

This work proposes the protocol of monotonic write consistency VIP-Grab for multicloud systems. The latency of user data writing can be used for numerical comparison and selection of the optimal multicloud architecture. This paper discusses the results of an experiment to determine the latency of writing in replicas of a real database located in Azure, AWS and Google Cloud Platform (GCP) for a geo-distributed middleware architecture.

2 Problem statement

Applying at least three replicas in different storage centres into different CSPs minimises the risk of failure data loss, thereby ensuring that users' data can be recovered in the event of a failure, as discussed at work Alshammari et al. (2020). In our work, the structure of a multicloud system is considered as a resource group of three different CSPs with a centralised control geo-distributed middleware with high availability (Figure 1).

Figure 1 General scheme of a multicloud system with geo-distributed middleware



In common case, a multicloud middleware can perform different types of tasks depending on the purpose of use and multicloud' architecture, for example, combining heterogeneous resources from different administrative domains; providing interoperability including managing data locality and latency; incentivising resource sharing; implementing easy-to-use authentication and authorisation mechanisms; analysing and combining a variety of billing models at different service as studied at works (Bittencourt and Calheiros, 2017; Koulouzis et al., 2020; Wang et al., 2020; Mealha et al., 2019). In this work, we assume that a multicloud middleware has a central role in the management of resources from different cloud providers by enabling a distributed system platform that can efficiently coordinate users traffic, route users' data read/write operations and also, as a service broker, collect real-time data.

Let the databases of each cloud contain identical data, i.e. are replicas, and the user is guaranteed that a read request sent to the database of any CSP will return up-to-date and identical data. We classify as a write (or update) any operation that modifies the value of an object in the database, while, conversely, reads return to the caller the current value held by the object's replica without causing any change to it.

The middleware architecture significantly affects the final level of data consistency for multiclouds, since the level of consistency implemented in the middleware additionally superimposed on the level of consistency implemented by each CSPs separately. In this regard, for numerical comparison of the final levels of data consistency in multicloud, it is advisable to use the numerical values of data reading latency, which in

turn depend on the consistency protocol implemented in the middleware. The main goal of the work is to develop a data consistency protocol for the middleware that would provide the minimal dynamically adjustable response from replicas in a multicloud with guaranteed identical up-to-date data when clients contact from any location.

3 Related work

Several works in the literature deal with different types of consistency from traditional to novel consistency models, such as Aldin et al. (2019) and Viotti and Vukolić (2016). Consistency means that each process knows of the other processes have access to the resource, whether they read or write, and also know what to expect. The authors demonstrate that characteristics of the distributed systems such as reliability, availability, latency, energy consumption, scalability, cost/monetary cost and others depend on the correct choice and effective implementation of the consistency model. Although they focus on more than 50 different consistency semantics that applies to distributed web services and distributed storage systems, none of them is related to the consistency model for multicloud systems with database replicas. Such research influenced the development of this work.

In Ardekani and Terry (2014), the Tuba – a geo-replicated key-value store – was implemented as middleware on top of Microsoft Azure Storage which provides geo-replication consistency-based service level agreements and automatic reconfiguration. Six levels of consistency – strong, eventual, read-my-writes (RMW), monotonic reads, bounded, and causal – are implemented in the proposed cloud storage system. Depending on the desired consistency/latency combination, the network delays between clients and various DCs, and cost constraints, new configurations of replicas executed. Authors experimentally demonstrated that cloud storage system reconfiguration completed automatically every two hours can increases read with guaranteeing strong consistency by 18% but the possibility of implementation of such an approach to multicloud systems should be discussed.

Lots of works, like Crain and Shapiro (2016), Ren et al. (2019) and Mealha et al. (2019), are focused on different types of algorithms and protocols for enforcing different consistency, for example, causally consistent protocol, SLOG to ensure strict serialisability guarantees, a weak consistency model protocols based on the operational transformation or conflict-free replicated data types (CRDT).

Kakwani and Nasre (2020) proposed a protocol supported read-only transactions (ROT), named Orion. Implementation of this protocol for enforcing causally consistent in distributed multi-version key-value store with full replication across three Google Cloud Platform DCs located in Iowa, Finland and Taiwan has demonstrated increasing up to 1.7× higher throughput against the existing protocol CausalSpartanX.

In some works, replication algorithms basing on various parameters combinations that reflect the functioning efficiency of distributed systems are proposed. In Wang et al. (2012), a dynamic data replication strategy based on a trade-off between reading access time and write updating cost using historical access records and proactive deletion is proposed. The authors note that the method of replica placement and weight of historical records are essential to achieve the required dynamic replication performance in distributed systems.

In Zawirski et al. (2013), latency by geo-replicating data in several DCs across the world was estimated. Proposed client-assisted failover protocol preserves causality cheaply and in addition to its updates, a client may observe a causally-consistent view of stable (i.e., stored at multiple servers) updates from other users. Experiment with DCs in three Amazon EC2 availability zones had shown that in the cloud-no fault-tolerant configuration, transaction latency is proportional to the client-DC RTT, and cloud-fault-tolerant suffers additional latency for writing to a quorum. Besides, SwiftCloud protocol has based on using a cache containing replicas of a subset of objects (partial replication), so this protocol is difficult to use for multicloud systems.

In Wu and Madhyastha (2013), the potential latency benefits of deploying webservices across three popular cloud infrastructure services – Amazon EC2, Google Compute Engine (GCE) and Microsoft Azure are estimated. The authors examine reasons for the potential latency benefits of web service deployments spanning multiple cloud services but the latency of read/write operations in a multicloud architecture with database replicas is not analysed.

In Eischer et al. (2020), a novel architecture of cloud-based geo-replication applications with the protocol to perform guaranteed writes are proposed. The protocol named Weave is a Paxos-based multi-leader state-machine replication protocol that optimised quorum sizes to provide guaranteed writes locally within each replica group hosted in separate cloud-provided fault domains. The experiment with replicas spread across the three Amazon EC2 regions in Ohio, Frankfurt and Sydney, demonstrate that Weave's response time is more than 78% lower than the Paxos broadcast optimisation protocol required two wide-area communication steps. Thus, this protocol can improve the quality of service and reduce latency in cloud systems of one CSP, but the techniques studied here are others. In this paper, we propose the VIP-Grab protocol of monotonic write consistency for multicloud systems.

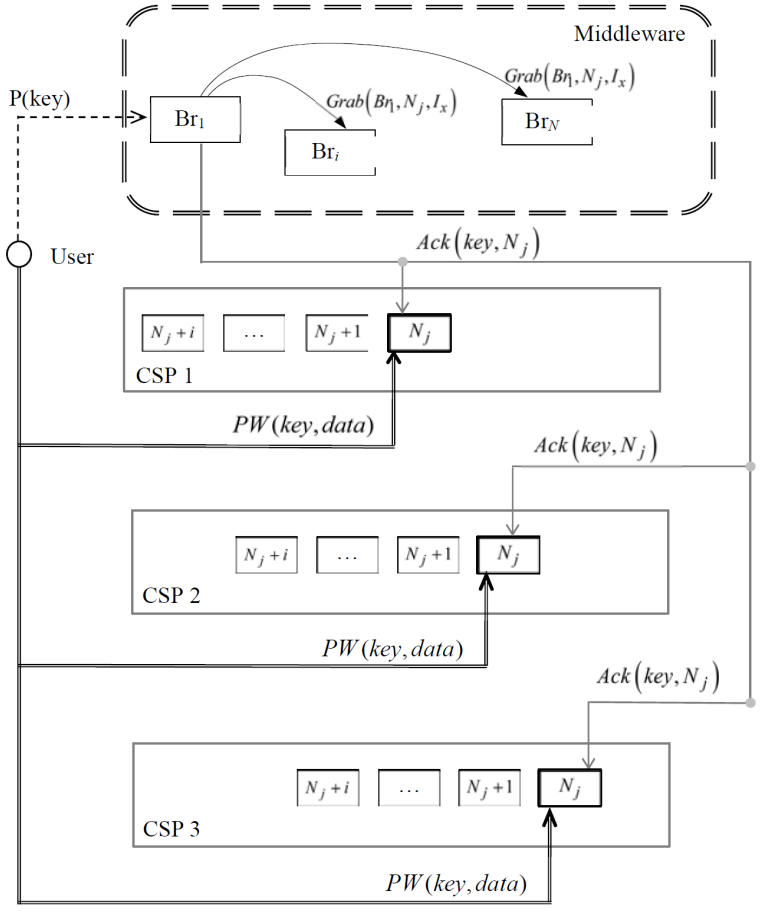
4 Protocol of consistency VIP-Grab

The basis of the algorithm for setting the order of data writing to databases on different CSPs is a mechanism in which the write operation does not automatically synchronise the replicas of databases located at different cloud providers. This is applying an approach similar to that used for active replication with a fully ordered global sequence of data write.

As a basic model of data consistency in a multicloud with middleware, the monotonic write consistency model is used. Data transfers between resources of different CSPs tend to have a longer WAN latency, so the proposed VIP-Grab protocol of consistency uses as little data and message transfers between clouds as possible.

In the VIP-Grab protocol, the formation of message flows is based on the developed data preparation algorithm, the input data for which is information about the data transmission intervals between various parts of the entire system. The use of the cumulative results of this algorithm allows for flexible adjustment of the response of the multicloud system both in the long and short term. It is also proposed to use various mechanisms for users data read and write operations processing.

Figure 2 Flows of packages for writing



The scheme of data flow control in the VIP-Grab protocol is shown in Figure 2. The client side of the cloud application sends a data packet for writing *PW*(*key*, *data*) with a unique *key* to the databases of all cloud providers. At the same time, the client side of the cloud application sends a message with this same *key* to the nearest middleware node. When the *key* is received by the middleware the matching node sends a notification about grab the current free number *N* in the global sequence for writing to the other nodes.

The proposed algorithm allows setting the correct order of increasing numbers in a global sequence for writing during the adjustment interval for all data packets received by all geo-distributed middleware nodes. At the end of the adjustment interval, each node that received a unique *key* send to all cloud databases the message *Ack*(*key*, N^*) containing this *key* and the corrected number in the global sequence N^* received after sorted such numbers.

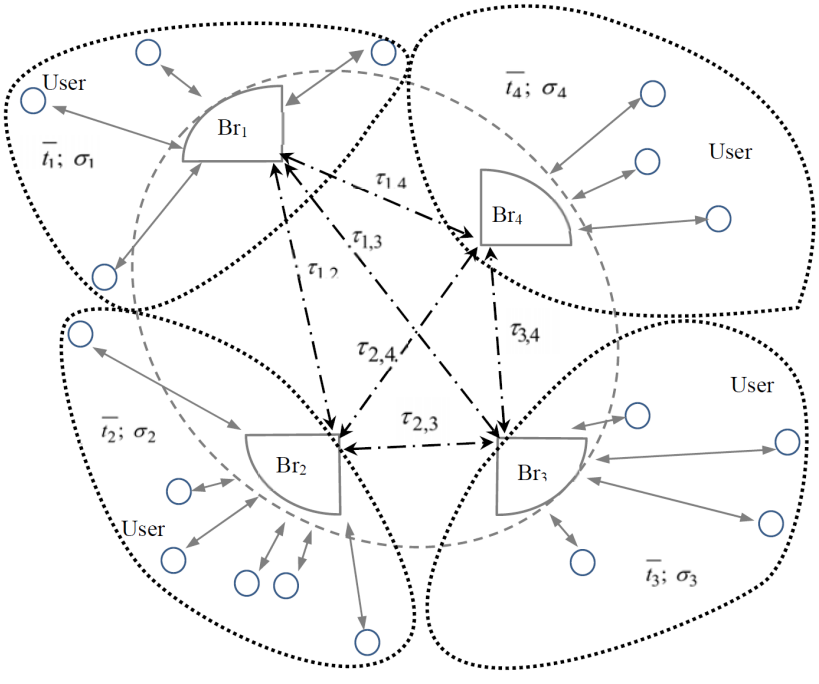
The package with user data is written to the database of each CSP only when from the middleware the sorted number N^* from the global sequence arrives issued for the corresponding *key* and only after the successful execution of writing procedure with the previous number $N^* - 1$ package. Acknowledgements about the successful write *Ack*(*w*, *key*) and the successful read *Ack*(*r*, *key*) go to the middleware.

Such consistency protocol allows processing read operations separately from data writing operations and directing read requests from users to the nearest CSP, thereby reducing response time, but it requires a conscious choice of localisation of middleware nodes. The middleware should have nodes located as close as possible to geographic clusters of users. It means that, on the one hand, it is possible to arrange the middleware at the CSP whose resources is going to use by the customer of the multicloud system for access and process data. However, on the other hand, if the main goal of organising a multicloud system is to avoid dependence on single cloud data storage, then middleware nodes should be located at an additional cloud resource provider that has the lowest round-trip time (RTT) from areas of the largest concentration of users.

4.1 Algorithm for preparing data for ordering numbers in a global sequence for writing in multicloud replicas

A justified choice of middleware nodes localisation allows us to consider the directions of data transfer between its nodes Br_i as a complete graph. Moreover, we assume that each middleware node serves its own assigned region of users. The directions of data transfer between four nodes Br_i with assigned regions of users are shown on the service region map in Figure 3. When a new user is connected, depending on its geographic location, the middleware determines which node it will be served by.

Figure 3 Service region map for middleware nodes



For each region, it is necessary to determine $\overline{t_i}$ – the average time of package delivery from users to node Br_i; σ_i^{Br} – the standard deviation (SD) of the delivery time of packets from users to node Br_i.

The choice of the sample duration for calculating such averages will be a very important step. The accumulation of statistically reliable data will make it possible to rank the necessary average response values from nodes depending on the day of the week, time of day, season or other parameters that are significant for the operation of the entire multicloud system, which means dynamically reconfigure data flows from users to middleware nodes.

The set of SDs $G = \{\sigma_i^{Br} \mid i \in (1 \dots M)\}$ is sorted in descending order and the position numbers of the elements in such a sorted set correspond to the service priority numbers by the SD $\Pr(\sigma_{Br})$ for the corresponding node, for example, for eight middleware nodes, i.e., for M = 8, priority table matching SDs might look like this:

Table 1 Priorities of nodes by SD

σ_i^{Br}	0.065	0.064	0.057	0.057	0.033	0.031	0.029	0.026
Br_i	5	3	4	7	8	2	6	1
$\Pr(\sigma_{Br})$	1	2	3	4	5	6	7	8

If the values σ_i are equal, the priority numbers are sorted by the node index of the distributed middleware.

The intervals during which the messages from one middleware nodes can reach other nodes are random values, therefore, as input values for the considered algorithm for generating numbers in the global sequence for writing, it is necessary to use a set of a priori obtained mean values and SDs of such message delivery intervals $\tau_{i,j}$ from each node to all others pairwise. For example, let for multicloud system with M = 8 the obtained symmetric matrix of pairwise message delivery intervals $\tau_{i,j}$ look like this:

$$\tau_{i,j} = \begin{array}{c|cccccccc} & Br_1 & Br_2 & Br_3 & Br_4 & Br_5 & Br_6 & Br_7 & Br_8 \\ \hline Br_1 & - & 251 & 80 & 62 & 148 & 135 & 90 & 51 \\ Br_2 & 251 & - & 129 & 359 & 41 & 28 & 29 & 28 \\ Br_3 & 80 & 129 & - & 31 & 132 & 98 & 71 & 62 \\ Br_4 & 62 & 359 & 31 & - & 31 & 149 & 30 & 47 \\ Br_5 & 148 & 41 & 132 & 31 & - & 31 & 36 & 134 \\ Br_6 & 135 & 28 & 98 & 149 & 31 & - & 143 & 80 \\ Br_7 & 90 & 29 & 71 & 30 & 36 & 143 & - & 143 \\ Br_8 & 51 & 28 & 62 & 49 & 134 & 80 & 143 & - \end{array} \quad (1)$$

Then the set F of maximum values for each row $\{\max \tau_{i,j}\}$ for this matrix will be such:

$$F = \{251, 359, 142, 359, 148, 149, 143, 143\} \quad (2)$$

The set F is sorted in descending order. In this case, the position numbers in such a sorted set correspond to the levels of service priorities for pairwise message delivery intervals, for example, for M = 8 according to the set F, the priority table for pairwise message delivery intervals $\Pr(\tau_{Br})$ may look like in Table 2.

Table 2 Priorities of nodes by pairwise message delivery intervals

$max\ \tau_{i,j}$	359	359	251	149	148	143	143	132
Br_i	4	2	1	6	5	7	8	3
$\Pr(\tau_{Br})$	1	2	3	4	5	6	7	8

In the case of equality of values max $\tau_{i,j}$, the priority levels are set following the priority levels by SDs for the same nodes, for example, according to the data in Table 2, max $\tau_{4,2}$ = max $\tau_{2,4}$ = 359, however, the priority level is $\Pr(\tau_{Br4}) > \Pr(\tau_{Br2})$ because $\Pr(\tau_{Br4}) > \Pr(\tau_{Br2})$. Thus, the priority levels by SDs are the second rule for ranking middleware nodes when prioritising nodes by pairwise message delivery intervals.

The average value of the longest pairwise message delivery interval between nodes max τ, taking into account the SD of this random variable, is the adjustment interval T for numbers in the global sequence, i.e., T = max $\tau + \sigma$. Thus, the adjustment interval is longer than any length of message delivery interval between all nodes in the middleware.

During the adjustment interval T, the message with the key from the user may arrive at the moment when the notification about the grab of the number, sent by this node, does not have time to reach all nodes. The 'availability window' W_i of a node Br_i is a certain interval from the beginning of the adjustment interval T, during which all messages sent by the node Br_i will have, on average, sufficient time to reach all other middleware nodes

$$W_i = T - \max_{j=1\ldots M} \tau_{i,j} \tag{3}$$

4.2 Algorithm for ordering numbers in a global sequence for writing in replicas of a multicloud

The main mechanism for numbers ordering in the global sequence for writing in the databases of various CSPs implies that a notification about sending data to cloud storages with a unique *key* from the user is sent to one middleware node Br_i only. During the adjustment interval, each middleware node Br_i receiving *key* must send to the rest of the nodes a message about number grab $Grab(Br_i, N_j, I_x)$, which contains:

- the identifier of the node Br_i that received *key*
- the primary version of the number in the sequence, N_j
- the identifier of the adjustment interval, I_x.

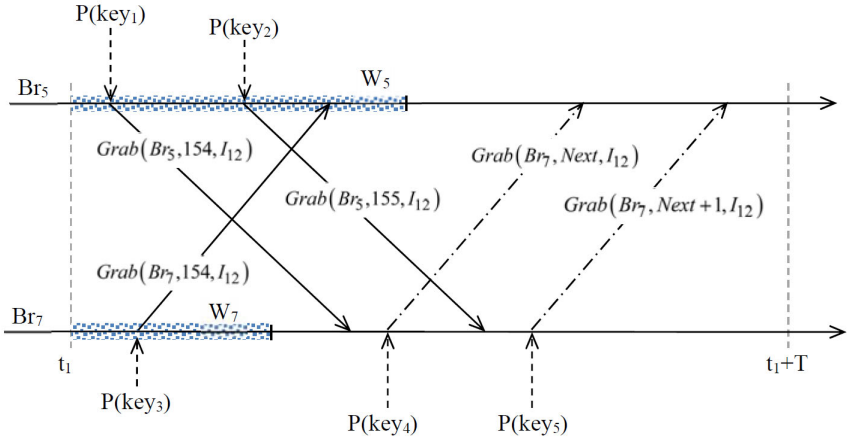
Each middleware node, having received a notification $Grab(Br_i, N_j, I_x)$ from any other node, the so-called 'foreign Grab', according to the rules for managing numbers, can correct the numbers of 'own Grabs' only, i.e., those capture notifications, the numbers for which it generated. The following rules are formulated for managing numbers in the global sequence for writing in each node:

1 Each node Br_i in its 'own Grab', which is generated in response to the first message from the user with a *key* in the 'availability window' W_i, uses the starting free number $N_f(Br_i, I_x)$ of the node as the number in the sequence. $N_f(Br_i, I_x)$ then increases by 1 and becomes the current sequence number for the current 'availability window'. When forming all subsequent 'own Grabs' from the 'availability window', the current free number $N_y(Br_i, I_x)$ is used, after which 1 is also added to it.

2 Each node Br_i in its 'own Grabs' sent after the end of the 'availability window' in the current adjustment interval, uses the value 'Next' instead of the current number $N_f(Br_i, I_x)$ in the global sequence, i.e., it sends messages like $Grab(Br_i, Next, I_x)$. Each subsequent notification in the current adjustment interval is assigned the value 'Next + 1' instead of a number in the sequence.

An example of 'own Grabs' exchange between nodes Br_5 and Br_7 in one adjustment interval I_{12} if free start number $N_f = 154$ is demonstrated in Figure 4. Let packets $P(key_4)$ and $P(key_5)$ arrived at node Br_7 during the current adjustment interval $I_{12} = [t_1, t_1 + T]$ after the end of the 'availability window' W_7. In this case, node Br_7 should send to node Br_5 message $Grab(Br_7, Next, I_{12})$ about packet $P(key_4)$ and message $Grab(Br_7, Next + 1, I_{12})$ about packet $P(key_5)$ but not use numbers from the sequence, like 154 or 155.

Figure 4 An example of 'own Grabs' exchange between nodes Br_5 and Br_7 in one adjustment interval I_{12} if free start number $N_f = 154$ (see online version for colours)



3 Upon expiration of each adjustment interval at each node Br_i, a new free start number is calculated in the sequence $N_f(Br_i, I_x)$ as the sum of the values: free start number of the previous adjustment interval $N_f(Br_i, I_{x-1})$; the amount of received 'foreign Grabs' without the value 'Next' for the current adjustment interval; the amount of 'own Grabs' sent by this node during the 'availability window', i.e., amount of messages without 'Next'; the amount of received 'foreign Grabs' with the values 'Next' sent from the previous adjustment intervals:

$$N_f(Br_i, I_x) = N_f(Br_i, I_{x-1}) + \sum_{\substack{j=1\ldots M,\\ N^* \neq Next}} Grab(Br_j, N^*, I_x) + \sum_{\substack{j=1\ldots M,\\ j \neq i,}} Grab(Br_j, Next, I_{x-1}) \quad (4)$$

4 All notifications about grab of the number generated by the current node Br_i with the value 'Next' in the previous adjustment intervals I_{x-1}, i.e., 'own Grabs' of the type $Grab(Br_i, Next, I_{x-1})$, at the beginning of a new adjustment interval I_x have received

the replacement of the value 'Next' to the value of the new free start number $N_f(Br_i, I_x)$ and recalculated. By such way, there is, as it were, a movement of the generation time of such notifications from the interval I_{x-1} to the beginning I_x and, accordingly, an increasing of the free number in the sequence $N_f(Br_i, I_x)$ for all other 'own Grabs' or in other words 'late notifications are considered first in the Next interval'.

5 If the node Br_i during the adjustment interval I_x received a 'foreign Grab' of the type $Grab(Br_i, Next, I_x)$, sent also in I_x, then this does not change the values of numbers in the sequence for 'own Grabs' in this I_x.

6 If the node Br_i during the adjustment interval I_x received a 'foreign Grab' of the type $Grab(Br_j, Next, I_{x-1})$ sent in the previous adjustment interval I_{x-1}, then the free start number in the sequence $N_f(Br_i, I_x)$ for the current adjustment interval is increased by 1, and accordingly, all the numbers already used in the current I_x are recalculated, in other words, 'skip ahead late notification'.

7 If the node Br_i during the adjustment interval I_x received a 'foreign Grab' of the type $Grab(Br_j, N_y I_x)$ sent in the current interval from a node with a higher priority $\Pr(\tau_{Br_j}) > \Pr(\tau_{Br_i})$, then all 'own Grabs' generated up to this moment during the 'availability window' W_i increase by 1 their numbers in the sequence. This also means that for 'own Grabs' of the type $Grab(Br_i, Next, I_{x-1})$ from the previous adjustment interval, the new value instead of the 'Next' is not calculated.

At the finish of every adjustment interval I_x, each node Br_i sends acknowledgement messages of the type $Ack(key, N_y)$ to all cloud storages. These messages are containing the key of data *key* and the recalculated by this time numbers from the notifications $Grab(Br_i, N_y, I_x)$ that were adjusted during this interval I_x.

The proposed algorithm for ordering numbers in a global sequence for writing generates for data received from one region during one adjustment interval the order of writing to the storages of various cloud providers similar to the order of their keys entering in the middleware. Using the numbers formed by this algorithm allows writing data received during the 'availability windows' from different regions in the same order to entering of their keys in the middleware. Also, the data, whose keys arrived in the middleware in the current adjustment interval, will have numbers for writing less than the data, whose keys arrived in the subsequent intervals. Thus, the numbers obtained by the proposed algorithm do not violate the chronology of writing data received as within each interval so between intervals from one region, as well as do not violate the chronology of writing data received within the 'availability windows' from different regions.

5 The latency of writing

As shown upper, applying the VIP-Grab protocol in a multicloud with the middleware leads to general latency $L_{MultiCl}$ between the users sending of data to cloud storage, let it be t_0, and the moment when this data is written to all databases in a determined order $t_i^w \mid i = 1 \ldots S$, where S is the number of CSPs in whose resources users' data are storing. Let $S = 3$, then for each cloud database

$$L_{MultiCl_i} = t_i^w - t_0 \tag{5}$$

According to the VIP-Grab protocol, writing in the databases of each cloud occurs at the moments when both the message from the user containing the data and the key *PW*(*key*, *data*) and the message from the middleware node *Ack*(*key*, N^*) containing the same *key* and the final corrected number N^* in the global sequence already is collected.

Let us assume that the moment when the message *PW*(*key*, *data*) enters in i^{th} cloud depends primarily on the average delivery interval $\overline{t_{U,Cl_i}}$ between a user device and the i^{th} cloud database, implying that the time interval required for processing and routing the message inside the cloud is much less than data delivery to the cloud.

The message from the middleware node *Ack*(*key*, N^*) can arrive in the i^{th} cloud after processing the message *PW*(*key*, *data*) by nearest to user middleware node Br_i. The time of this occurring also depends on the average delivery interval $\overline{t_{Br_j,Cl_i}}$ of the message between the j^{th} middleware node and the i^{th} cloud.

Each middleware node generates a corrected number N^* during one adjustment interval T if the message from the user arrived during the 'availability window' W_j of this middleware node Br_i in the current adjustment interval, or during the next adjustment interval, if such message arrived to Br_i after finishing the 'availability window' W_j in the previous adjustment interval. This means that the minimum time for forming a corrected number N^* is the same for all middleware nodes and is equal to the adjustment interval T. The maximum time for forming the corrected number N^* depends on a node and for j^{th} node is $T + \max\limits_{i=1\ldots M} \tau_{j,i}$.

Considering that the adjustment interval depends on the maximum of inter-nodes message delivery intervals, for the furthest middleware node Br_A in terms of inter-nodes pairwise delivery time, the duration of forming N^* by VIP-Grab protocol T_s can be equal

$$T + \max_{i=1\ldots M} \tau_{Br_A,i} = \max_{i=1\ldots M} \tau_{Br_A,i} + \sigma_A + \max_{i=1\ldots M} \tau_{Br_A,i} = 2 \max_{i=1\ldots M} \tau_{Br_A,i} + \sigma_A \tag{6}$$

In the common case, the duration T_s of forming corrected number N^* for j^{th} middleware node take on value into

$$\left[(\max \tau + \sigma)\ldots\left((\max \tau + \sigma) + \max_{i=1\ldots M} \tau_{j,i}\right)\right] \tag{7}$$

Thus, in the absence of restrictions on writing associated with incomplete writing operations of users data with previous numbers in the global sequence, writing of users' data processed in j^{th} middleware node occurs in the i^{th} cloud at the moment t_i^w of the presence of both messages, i.e. at the moment when the next logical expression is equally *true*:

$$t_i^w = \left(t_0 + \overline{t_{U,Cl_i}}\right) \wedge \left(t_0 + \left(\overline{t_{U,Br_j}} + T_s\right) + \overline{t_{Br_j,Cl_i}}\right) \tag{8}$$

where $\overline{t_{U,Br_j}}$ is the average delivery interval between a user device and nearest to him/her middleware node Br_j. Then writing of users data in i^{th} cloud database can be carried out only after sending the data, i.e. after the duration of general latency $L_{MultiCl}$:

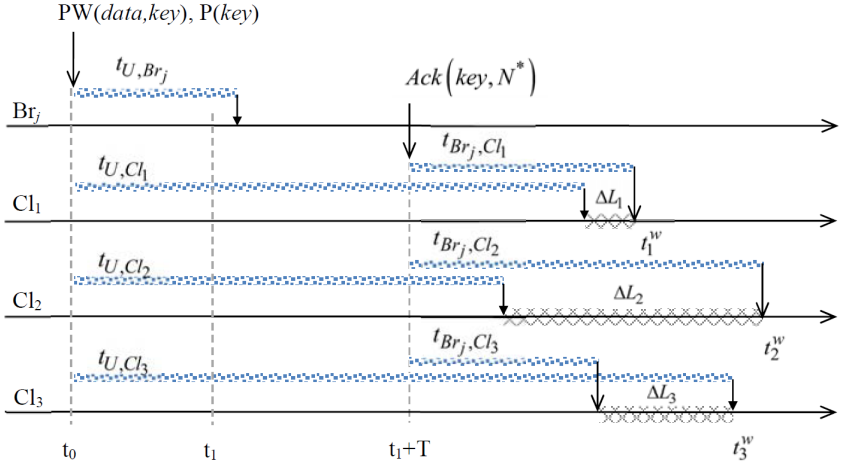
$$L_{MultiCl_i} = t_i^w - t_0 = \begin{cases} \left(t_0 + \overline{t_{U,Cl_i}}\right) - t_0 = \overline{t_{U,Cl_i}}, & \text{if } \Delta L_i < 0 \\ \left(t_0 + \left(\overline{t_{U,Br_j}} + T_s\right) + \overline{t_{Br_j,Cl_i}}\right) - t_0 = \left(\overline{t_{U,Br_j}} + T_s\right) + \overline{t_{Br_j,Cl_i}}, & \text{if } \Delta L_i > 0 \end{cases} \quad (9)$$

where ΔL_i is relative latency of writing determined as the difference between moments of receiving user data $PW(key, data)$ and the acknowledgement message $Ack(key, N^*)$ from j^{th} middleware node for i^{th} cloud, i.e.,

$$\Delta L_i = \left(\overline{t_{U,Br_j}} + T_s\right) + \overline{t_{Br_j,Cl_i}} - \overline{t_{U,Cl_i}} \quad (10)$$

Intervals determined relative latencies ΔL_i of writing data to clouds Cl_1, Cl_2 and Cl_3 are shown in Figure 5. Suppose a data packet $PW(key, data)$ sends by the client side at t_0 and the adjustment interval T for each node Br_j starts at t_1.

Figure 5 General latency $L_{MultiCl_i} = t_i^w - t_0$ and relative latencies ΔL_i of writing (see online version for colours)



The duration of forming corrected number N^* for j^{th} middleware node can take values from the interval (7), therefore the minimum relative latency ΔL_i of writing for a user data after arriving at the cloud:

$$\Delta L_i^{\min} = \left|\overline{t_{U,Br_j}} - \overline{t_{U,Cl_i}} + (\max \tau + \sigma) + \overline{t_{Br_j,Cl_i}}\right| \quad (11)$$

and maximum relative latency equal to

$$\Delta L_i^{\max} = \left|\overline{t_{U,Br_j}} - \overline{t_{U,Cl_i}} + 2\max \tau + \sigma + \overline{t_{Br_j,Cl_i}}\right| \quad (12)$$

and therefore the difference between the maximal and minimal data writing relative latency in a multicloud systems by using VIP-Grab protocol is determined by the duration of the longest pairwise message delivery interval between middleware nodes $\max \tau$.

6 Reading control

The existence of general latency $L_{MultiCl_i}$ of writing for replicas in all clouds at applying of VIP-Grab protocol of consistency data shows simultaneously requests to reading for all clouds will not give identical results, so reading control mechanism must be used.

The justified choice of the middleware nodes localisation, as described in Section 4, implies a priori analysis of pairwise message delivery intervals between its nodes, i.e. the minimum relative latency $\Delta L_i^{\min}$ for each cloud can be estimated in advance. In this case, during the interval $L_{MultiCl_i}$ ordered data will be written in all storages of the multicloud system, which means that to ensure the identity of responses to read requests, it is necessary to wait during an appropriate time before reading in every cloud.

The durations of pairwise message delivery intervals max τ between middleware nodes are random values, the values of which many CSPs tend to reduce to improve the quality of service to their customers, therefore, for the correct implementation of the selected consistency model, it is necessary to collect and analyse data which also makes it possible to dynamically correct the adjustment interval T in the middleware and reading latency.

7 Experiment

Three CSPs for data storage were chosen as a test environment for numerical estimation of data write latency: AWS, Azure and GCP. Copies of the relational database were hosted on DC located in Ohio (AWS), on Frankfurt DC (GCP) and Western Europe DC (Azure).

It is assumed that most users are located in Europe, but it is necessary to maintain accessibility to the data for users from all over the world. It is assumed that most data users of the selected multicloud system are located in Europe, but it is necessary to maintain the accessibility of data for users from all over the world. Each provider of the selected multi-cloud system has a data centre located in Frankfurt. To choose the provider to host the middleware for the tested multicloud system, we analysed the average of the data read intervals from the database copies for Frankfurt data centres of each provider.

The console application has been developed on .Net Core 2.2 to form a stream of requests to read data from databases in all CSPs. Every day in the morning, afternoon and evening for three weeks, our application sent 1,000 requests to select data from R rows using the instruction select top R. Thus, we received 63,000 intervals with response times from each database for every CSP. To reduce the risk of correlated results number of rows N in each query was changed randomly, and every request was sent after getting a reply to the previous one with a 20 ms delay. Our application has launched from the same desktop PC located in Kharkiv (Ukraine). The request payload is set to 140 bytes. Average values and SDs of the obtained response intervals from the database for each CSP in milliseconds are presented in columns 2, 3, 4 of Table 3.

Table 3 Response times from databases located at different CSPs

CSP (region)	*AWS (Frankfurt)*	*Azure (Frankfurt)*	*GCP (Frankfurt)*	*AWS (Ohio)*
1	2	3	4	5
Average time ms	101.08	105.17	108.87	208.47
SD, t_{U,Cl_i}, ms	0.0345	0.0249	0.0369	0.0385

AWS has the lowest average response time for reading requests at 101.08 ms, so we recommend choosing AWS to host the multicloud middleware for the tested multicloud system.

At the same time with using our application, the experiment of a similar form and content was conducted to collect and analyse the response times for reading requests to the database located on Ohio (AWS). Average values and SD of the obtained response intervals from the Ohio database are presented in column 4 of Table 3.

The average values of the time intervals obtained in this experiment can be considered approximately equal to twice the duration of the message delivery from a user device to those CSPs whose resources there will be no middleware, i.e., half values from columns 3, 4, 5 in Table 3 can be used as $\overline{t_{U,Cl_i}}$ in general latency $L_{MultiCl\,i}$ calculations.

Let us consider a case when a user, being in Kharkiv (Ukraine), writes data to AWS, Azure and GCP databases. To determine the latency of writing in each cloud $L_{MultiCl_i}$, it is necessary to compare the minimum relative latency $\Delta L_i^{\min}$ with zero. Assume that the nearest middleware node that process users requests from Kharkiv is located in Frankfurt (AWS), therefore, $\overline{t_{U,Br_{AWS,Frankfurt}}} = 50.54$ ms.

The minimum duration T_s of the formation of a number N^* in the global sequence due to VIP-Grab protocol, as shown in Section 4, is determined as $\max\tau+\sigma$ and is chosen as the maximum values of pairwise message delivery interval between all middleware nodes. According to data from AWS Latency Monitoring[1] the average intra-region latency for AWS nodes in December 2020 is 468.79 ms and we will consider this value as an approximate doubled value of the data transfer interval between its nodes. SD for average RTT was not found in the public sources, so we assume for AWS σ = 0.06 ms and max $\tau + \sigma \approx 234.39 + 0.06 = 234.45$ ms. Now, let's determine the general latency of writing in each cloud.

The minimum relative latency of writing for a user data after arriving at the Azure database is

$$\begin{aligned}\Delta L_1^{\min} &= \overline{t_{U,Br0}} - \overline{t_{U,Cl1}} + (\max\tau+\sigma) + \overline{t_{Br0,Cl1}} \\ &= 50.54 - 52.58 + 234.45 + \overline{t_{Br0,Cl1}} \\ &= 232.41 + \overline{t_{Br0,Cl1}} > 0\end{aligned} \tag{13}$$

where $\overline{t_{Br0,Cl1}}$ is the average message delivery interval between the nearest middleware node (AWS, Frankfurt) and cloud database in Azure (Frankfurt). According to round-trip latency Azure[2], the average intra-regional RTT between the closest West Europe Azure nodes is 11 ms, therefore, we will assume $\overline{t_{Br0,Cl1}} \approx 5$ ms. Then, $\Delta L_1^{\min} = 237.41$ ms and the general latency of writing data to the Azure database equal to

$$L_{MultiCl_1} = \left(\overline{t_{U,Br_0}} + T_s\right) + \overline{t_{Br_0,Cl_1}} = (50.54 + 234.45) + 5 = 289.99 \text{ ms} \quad (14)$$

The minimum relative latency of writing for a user data after arriving at the GCP (Frankfurt)

$$\begin{aligned} \Delta L_2^{\min} &= \overline{t_{U,Br_0}} - \overline{t_{U,Cl_2}} + (\max\tau + \sigma) + \overline{t_{Br_0,Cl_2}} \\ &= 50.54 - 54.43 + 234.45 + \overline{t_{Br_0,Cl_2}} \\ &= 230.56 + \overline{t_{Br_0,Cl_2}} > 0 \end{aligned} \quad (15)$$

According to Google Cloud latency between regions[3] the average intra-regional RTT between the nearest nodes of GCP in Europe is 7.42 ms, therefore, let message delivery interval between the middleware node (AWS, Frankfurt) and GCP (Frankfurt) is $\overline{t_{Br_0,Cl_2}} \approx 3$ ms. Then, the minimum relative latency equal to $\Delta L_2^{\min} = 233.56$ ms and general latency of writing data to the GCP database is

$$L_{MultiCl_2} = \left(\overline{t_{U,Br_0}} + T_s\right) + \overline{t_{Br_0,Cl_2}} = (50.54 + 234.45) + 3 = 287.99 \text{ ms} \quad (16)$$

The minimum relative latency of writing for a user data after arriving at the AWS (Ohio)

$$\begin{aligned} \Delta L_3^{\min} &= \overline{t_{U,Br_0}} - \overline{t_{U,Cl_3}} + (\max\tau + \sigma) + \overline{t_{Br_0,Cl_3}} \\ &= 50.54 - 104.23 + 234.45 + \overline{t_{Br_0,Cl_3}} \\ &= 180.76 + \overline{t_{Br_0,Cl_3}} > 0 \end{aligned} \quad (17)$$

According to 1 the average intra-regional RTT between AWS nodes in Ohio and Frankfurt determined due to 30-days observations is 102 ms, so assume that $\overline{t_{Br_0,Cl_2}} \approx 51$ ms. Then, minimum relative latency equal to $\Delta L_3^{\min} = 231.76$ ms and general latency of writing data to the AWS database is

$$L_{MultiCl_3} = \left(\overline{t_{U,Br_0}} + T_s\right) + \overline{t_{Br_0,Cl_3}} = (50.54 + 234.45) + 51 = 335.99 \text{ ms} \quad (18)$$

8 Discussion and conclusions

Analysis of the obtained values of the minimum relative latency $\Delta L_i^{\min}$ of writing in a multicloud system by using the VIP-Grab consistency protocol shows that the main contribution to them is made by the duration of the adjustment interval determined as max τ + σ, which in turn depends on the intra-region latency between the middleware nodes. On the other hand, the analysis of message delivery intervals between a geographic cluster of users $\overline{t_{U,Br_j}}$ and middleware nodes also plays a significant role in location choosing for middleware, as in the dynamical correction of the adjustment interval. The more values $\overline{t_{U,Br_j}}$ are collected for potential CSPs with middleware before deciding on its placement, the more optimal response of the entire multicloud system will be obtained, both at reading/writing for users.

For the cases of using a multicloud system to organise a highly reliable service with wide geo-distributed data availability, it should be noted the impact of the quality of a priory collection of message delivery intervals between all parts of the multicloud system: multicloud architects should pay attention to statistically significant sample size, the type of protocol in experiments – ICMP or TCP, as well as the dependence of latency on the time of day and seasons.

Thus, in this paper, we presented VIP-Grab protocol based on the geo-distributed architecture of middleware, which provides the ordering of numbers in a global sequence for writing operations in database replicas located in different CSPs. The duration of the interval $L_{MultiCl_i}$ during which the data ordered by VIP-Grab protocol will be written to the databases of all clouds can be used to numerically compare the optimality of multicloud systems architectures and select the appropriate CSPs configuration per the client's requirements. Determining the optimal latency of writing in each database of a multicloud system with middleware, and hence the values of deferred reading, is a multi-criteria task even from the point of view of estimating time parameters.

References

Abualkishik, A.Z., Alwan, A.A. and Gulzar, Y. (2020) 'Disaster recovery in cloud computing systems: an overview', *(IJACSA) International Journal of Advanced Computer Science and Applications*, Vol. 11, No. 9, pp.702–710, ISSN 2158-107X.

Aldin, H.N.S., Deldari, H., Moattar, M.H. and Ghods, M.R. (2019) *Consistency Models in Distributed Systems: A Survey on Definitions, Disciplines, Challenges and Applications* [online] https://arxiv.org/pdf/1902.03305v1.pdf (accessed 20 April 2021).

Alshammari, M.M., Nordin, A., Alwan, A.A. and Abualkishik, A.Z. (2020) 'Data backup and recovery with a minimum replica plan in a multi-cloud environment', *International Journal of Grid and High Performance Computing*, Vol. 12, No. 2, pp.102–120, ISSN 1938-0259; E-ISSN 1938-0267.

Ardekani, M.S. and Terry, D.B. (2014) 'A self-configurable geo-replicated cloud storage system', *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pp.367–381, ISBN 978-1-931971-16-4.

Bittencourt, L.F. and Calheiros, R.N. (2017) 'Middleware for multicloud', *IEEE Cloud Computing*, Vol. 4, No. 4, pp.22–25, E-ISSN 2325-6095 [online] https://www.computer.org/csdl/magazine/cd/2017/04/mcd2017040022/13rRUILLkxA (accessed 20 April 2021).

Crain, T. and Shapiro, M. (2016) 'Designing a causally consistent protocol for geo-distributed partial replication', *PaPoC '15: Proceedings of the First Workshop on Principles and Practice of Consistency for Distributed Data*, April, Article No. 6, pp.1–4 [online] https://doi.org/10.1145/2745947.2745953 (Accessed 22 July 2021).

Chandavale, A., Gade, A. and Dixit, A. (2019) 'Medical knowledge extraction scheme for cloudlet-based healthcare system to avoid malicious attacks', *International Journal of Cloud Computing*, Vol. 8, No. 4, pp.319–331, ISSN 2043-9989.

Eischer, M., Straßner, B. and Distler, T. (2020) 'Low-latency geo-replicated state machines with guaranteed writes', *PaPoC '20: Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*, April, Article No. 13, pp.1–9 [online] https://doi.org/10.1145/3380787.3393686 (Accessed 20 April 2021).

García-Dorado, J.L. (2015) *Bandwidth in the Cloud* [online] https://export.arxiv.org/ftp/arxiv/papers/1512/1512.01129.pdf (accessed 20 April 2021).

Gudeme, J.R., Pasupuleti, S.K. and Kandukuri, R. (2019) 'Review of remote data integrity auditing schemes in cloud computing: taxonomy, analysis, and open issues', *International Journal of Cloud Computing*, Vol. 8, No. 1, pp.20–49, ISSN 2043-9989.

Ibrahim, A.A.Z.A., Kliazovich, D. and Bouvry, P. (2016) 'Service level agreement assurance between cloud services providers and cloud customers', *CCGRID '16: Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pp.588–591, ISBN 978-1-5090-2452-0.

Jambunathan, B. and Yoganathan, K. (2018) 'Managing storage in multicloud environment', *International Journal of Pure and Applied Mathematics*, Vol. 118, No. 9, pp.617–623, ISSN 1311-8080.

Kakwani, D. and Nasre, R. (2020) 'Orion: time estimated causally consistent key-value store', *PaPoC '20: Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*, April, Article No. 3, pp.1–6 [online] https://doi.org/10.1145/3380787.3393676 (Accessed 20 April 2021).

Koulouzis, S., Martin, P. and Zhao, Z. (2020) 'Virtual infrastructure optimisation', in Zhao, Z. and Hellström, M. (Eds.): *Towards Interoperable Research Infrastructures for Environmental and Earth Sciences. Lecture Notes in Computer Science*, Vol. 12003, pp.192–207, Springer, Cham, ISBN 978-3-030-52828-7.

Kozina, O.A. and Panchenko, V.I. (2018) 'Karta nesuperechnosti danykh dlia bahatokorystuvatskykh onlain rolovykh ihor [Consistency data map for multiplayer online role games]', *Herald of the National Technical University 'KhPI'. Subject Issue: Information Science and Modelling*, No. 24(1300), pp.160–168, in Ukrainian, ISSN 2079-0031.

Mealha, D., Preguiça, N., Gomes, M.C. and Leitão, J. (2019) 'Data replication on the cloud/edge', *PaPoC '19: Proceedings of the 6th Workshop on Principles and Practice of Consistency for Distributed Data*, March, Article No. 7, pp.1–7 [online] https://doi.org/10.1145/3301419.3323973 (accessed 20 April 2021).

Rafique, A., Landuyt, D.V., Lagaisse, B. and Joosen, W. (2015) 'Policy-driven data management middleware for multi-cloud storage in multi-tenant SaaS', *2015 IEEE/ACM 2nd International Symposium on Big Data Computing*, Limassol, Cyprus, pp.78–84, ISBN 978-0-7695-5696-3.

Ren, K., Li, D. and Abadi, D.J. (2019) 'SLOG: serializable, low-latency, geo-replicated transactions', *Proceedings of the VLDB Endowment*, Vol. 12, No. 11, pp.1747–1761, ISSN 2150-8097.

Rezaeibagha, F. and Mu, Y. (2016) 'Distributed clinical data sharing via dynamic access-control policy transformation', *International Journal of Medical Informatics*, Vol. 89, pp.25–31, ISSN 1386-5056, http://dx.doi.org/10.1016/j.ijmedinf.2016.02.002.

Viotti, P. and Vukolić, M. (2016) 'Consistency in non-transactional distributed storage systems', *ACM Computing Surveys*, Vol. 49, No. 1 [online] https://doi.org/10.1145/2926965 (accessed 20 April 2021).

Wang, P., Zhao, C., Wei, Y., Wang, D. and Zhang, Z. (2020) 'An adaptive data placement architecture in multicloud environments', *Scientific Programming*, Vol. 2020 [online] https://www.hindawi.com/journals/sp/2020/1704258/ (accessed 20 April 2021).

Wang, Z., Li, T., Xiong, N. and Pan, Y. (2012) 'A novel dynamic network data replication scheme based on historical access record and proactive deletion', *J. Supercomput.*, Vol. 62, No. 1, pp.227–250 [online] https://doi.org/10.1007/s11227-011-0708-z (accessed 20 July 2021).

Wu, Z. and Madhyastha, H.V. (2013) 'Understanding the latency benefits of multi-cloud webservice deployments', *ACM SIGCOMM Computer Communication Review*, April, Vol. 43, No. 2, pp.13–20, ISSN 0146-4833.

Zawirski, M., Bieniusa, A., Balegas, V., Duarte, S., Baquero, C., Shapiro, M. and Preguiça, N. (2013) 'Fault-tolerant geo-replication integrated all the way to the client machine', *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, October [online] https://arxiv.org/pdf/1310.3107.pdf (accessed 20 April 2021).

Notes

1 *AWS Latency Monitoring* [online] https://www.cloudping.co/grid/p_98/timeframe/1M (Accessed 20 April 2021).

2 *December 2020 Round-trip Latency Azure* [online] https://docs.microsoft.com/en-us/azure/networking/azure-network-latency (Accessed 20 April 2021).

3 Kumar, C. (2020) *How much is Google Cloud Latency between Regions? Netsparker Web Application Security Scanner – The Only Solution that Delivers Automatic Verification of Vulnerabilities with Proof-Based Scanning* [online] https://geekflare.com/google-cloud-latency/ (accessed 20 April 2021).