ICIT 2015 The 7th International Conference on Information Technology
doi:10.15849/icit.2015.0031

Consistency tradeoffs on distributed multi-datacentric systems

SCOLCH theorems

Balla W. Diack and Samba Ndiaye
Dept. of Mathematic and Informatics
Cheikh Anta Diop University of Dakar
Dakar, Senegal 18522
balla.diack@ucad.edu.sn
samba.ndiaye@ucad.edu.sn

Yahya Slimani
Department of Computer Sciences
ISAMM, University of Manuba
2060 Manuba, Tunisia

***Abstract*—Distributed systems in the cloud computing context spread data across geographically remote datacenters to ensure always availability, scalability, and a best reactivity. Choosing latter properties in these systems leads to consistency issues (version conflicts, obsolete data, etc.); besides, most analytical solutions suggested for these issues are incomplete. SCOLCH proposes tradeoffs to achieve the required properties for service level agreement in cloud computing.**

Keywords— multi-datacentric systems; consistency; convergence; latency; availability

I. INTRODUCTION

Data explosion on datawarehouses referred to as BigData, has completely shaken the modern distributed systems and has led to the cloud computing which was impulsed since the mid-2000s by Amazon, Google, Salesforce.com, etc. The most of cloud service providers have set new levels of consistency in their distributed databases (Dynamo [15], PNUTS [13], BigTable [14], Cassandra [19]) to ensure better performance and to keep their databases always available. These actors claimed that the eventual consistency [29] should not be overcome by distributed systems in the cloud. This statement is based on the CAP theorem (Consistency, Availability, Partition tolerance), also known as Brewer's theorem [10]. Nonetheless, number of researchers have criticized this theorem and have showed its limitations [27, 28].

In section 2 of this paper, we give mathematical formalizations of basic concepts in multi-datacentric systems. In section 3, we give new theorems associated with tradeoffs between strong consistency, convergence, high availability, low latency and causal consistency. In section 4, we remind the related works on these issues. In the last section, we conclude and we give some opened issues.

II. WIDELY DISTRIBUTED SYSTEMS ON THE CLOUD

In this section, we outline basic concepts which are often used but rarely explained by the authors and which are subject to a lot of confusions. Cloud computing consists essentially of a set of datacenters (thousands of servers per datacenter) and services provided to ubiquitous clients across the internet. A datacenter allows to house computer systems and their associated components.

A. Basic concepts

1) Unreliable and asynchronous systems: A system is said to be unreliable whether the messages between nodes may be reordered, dropped, or delayed for an arbitrary but finite duration [23]. A distributed system is asynchronous if its logical local clocks run at different speeds, i.e two operations that are performed simultaneously at two different nodes may appear to be executed at different logical time.

*2) Safety and liveness:*Safety in distributed systems means that some bad thing doesn't happen during execution; liveness means that a good thing happens eventually [2, 20].

3) Operation: An operation *u* is either a read operation or a write operation; it is characterized by two timestamps: the beginning of its performing on a node *start(u)*, and the deliverance of a response which indicate the termination of the operation *resp(u)*. An operation belongs to an execution (process), deals with an item on a node in a datacenter.

Notations: We use the following notations which are equivalent: $(op_i(O,x))_{do/Nj}$, $((op_i(O,x))_{do})$, $((op_i(O,x))_{Nj})$, $((op_i(x))_{do})$, $((op_i(x))_{Nj})$, $(op_i(x))$

doi:10.15849/icit.2015.0031

Whether op_i (O; x))d=N_j is an operation performed on the node $N_j \epsilon$ d a datacenter in the system. O is the item involved in op and x is the value modified or read in O. These depend on whether you point out the datacenter and the node where the operation is performed.

4) Execution: An execution (process) is a series of operations performed by a user.

5) Availability: It's a liveness property that means that all operations issued to the datacenter complete successfully. No operation can block indefinitely. The high availability means that any operation to a relevant node should result a response.

B. Happen-before relation:

We recall the classical happen-before relation (HBR noted →) with adjustments. HBR is defined on a graph (G) which vertices reflect all operations performed by a relevant node in the system. HBR satisfies the following assertions:

- If *a* and *b* are two operations on a same execution, then $a \rightarrow b$ if there's an oriented edge from v_a to v_b *(with v_a and v_b the respective vertices corresponding to a and b).*
- If *w* is an update and *r* is a read that returns the value written by *w*, then $w \rightarrow r$.
- Let a, b, and c are three operations in an execution, if $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$.
- Two operations u and v are concurrent if $u \rightarrow v$ and $v \rightarrow u$ are not verified.

C. Levels of consistency

1) Causal Consistency: For any execution: $U=\{u_1 \ldots u_k\}_{k\geq 1}$, *U* is causally consistent if and only if:

- There is a serial order of the operations of *U* at each node, i.e., HBR is verified on the operations in *U*.
- Any read operation *r* in *U* at a node N_i on an item *o* returns the latest concurrent write at this node on *o*. *r(o):* $x = x_o$ such as $w_k(o, x_o) \Rightarrow \forall j \neq k \ \ w_j(o, x) \rightarrow w_k$.

Causal consistency means that an operation $op_i(o,x_o)_{dk/Nj}$ completes if and only if: $\forall w_i$ such as $w_i(x_o)_{dk/\ Nj} \rightarrow op_i(x_o)_{dk/Nj}$ then w_i is completed.

2) Linearizability or strong consistency: An execution U is said to be linearizable if its operations appear to take effect across the entire system at a single instance in time between the invocation and the completion (delivrance of response) of the operation.

$$\forall\ w_i\ ,\ (w_i(o, x_0)) \in U \Rightarrow (r_i(o){:}x_0)_{dh}\ \forall\ d_h \in D$$

D is the set of datacenters $\{d_1 \ldots d_m\}_{m\geq 1}$; w_i the write number i of U; d_h is a random datacenter; $x_o$is the latest updated value of item o returned w_i.

3) The window of inconsistency is the duration in which an item is not up to date at a node.

D. Latency and convergence:

1) Latency: It is the delay between a request starts and its completion; particularly, the low latency is the latency which does not exceed few tens of milliseconds.

2) Convergence: A system is strongly convergent if any set of relevant and connected nodes that have received, performed and propagated the same updates will have equivalent state, i.e., all the reads on these nodes will return the same result.

$$\sum_{i=0}^{m} w_i = \sum_{j=0}^{m} w_j \Longrightarrow \forall\ \mathrm{r}\ ,\ (\mathrm{r(o,x)})_{Ni} = r(o,x))_{Nj}$$

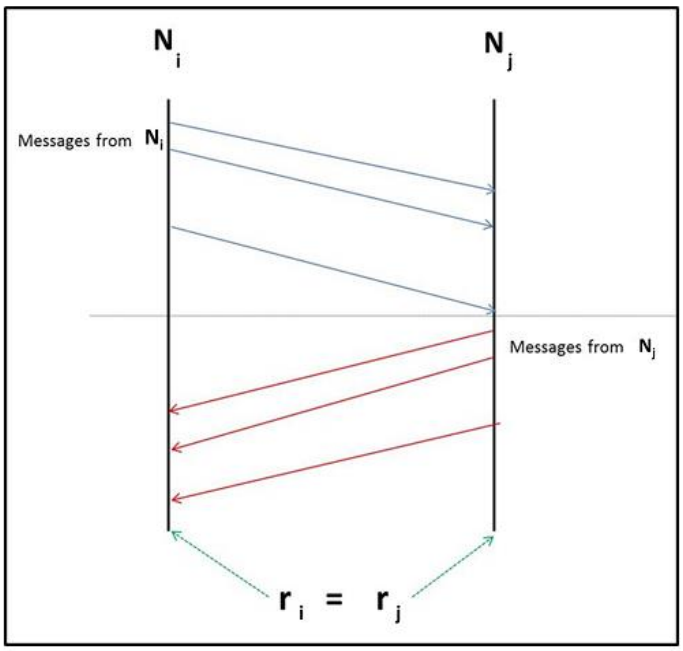

Fig. 1. Illustration of the one way convergence

In this section, we've outlined important concepts, what should be used in the next section to prove our choices.

III. NEW TRADEOFFS ON MULTIDATACENTRIC SYSTEMS

In this section, we are proving the incompatibility between strong consistency in a side and high availability, low latency and convergence in another side. Afterwards, we prove that we can guarantee the latter three properties if the causal consistency is ensured.

A. Strong consistency in multi-datacentric systems

1) Proposition 1: "Any multi-datacentric system is unreliable and asynchronous."

doi:10.15849/icit.2015.0031

The unreliability is intrinsic to widely extended networks as reported by Peter Deutsch [16].

Synchronization means that all nodes in the system have the same clocks (a clock is a function in each node which returns a real number for any operation performed on that node). Ensuring the synchronization requires an enormous overhead, furthermore, in actual multi-datacentric systems; synchronizing clocks on the nodes is not so relevant if we want to guarantee always availability..

2) Theorem 1: "Any multi-datacentric system, which guarantees strong consistency, cannot be always available."

We assume an asynchronous and unreliable distributed system with n servers $(S_1 \ldots S_n)_{n\geq 2}$ allocated to *m* datacenters $(d_1 \ldots d_m)$ and we assume that the strong consistency is guaranteed.

$d_1=\{S_1 \ldots S_i\} \ldots d_m=\{S_{j+1} \ldots S_n\} \quad i \leq j$

A break on the network between two datacenters $d_p=\{S_i \ldots S_{i+k}\}$ and $d_q=\{S_j \ldots S_{j+k}\}$ partitions their set of servers. All messages between the two datacenters are lost or delayed until the recovery of the partitions. We suppose by contradiction that the system provide high availability.

Let u_1 be an update on two datacenters (d_p and d_q); $u_1(o,x)_{dp}$ is performed on a server in d_p and is propagated through the others replicas of *o*. The high availability implies that the update is committed at all the replicas. Knowing that d_q is unreachable; any read $r_1(o)_{dq}$ to a replicas in d_q during the inconsistency window will necessarily return a wrong response $x_q \neq x$; which violate the strong consistency. Therefore we came across a contradiction. Consequently, the system will not be always-available if it is strongly consistent.

B. Latency tradeoffs

Although some works had highlighted the latency tradeoff on distributed systems [6,22,23], neither of them has mathematically proven the relationship between the latency and consistency level. In this section we try to prove that.

1) Theorem 2: "Any multi-datacentric system that ensures strong consistency will see its latency increasing dramatically."

We suppose a multi-datacentric system ($\dot{S}$), which is strongly consistent. For any execution $U \in \dot{S}$, and for any update $u_i(x) \in U$ on two nodes belonging to any pair of datacenters $(d_j, d_k) \in \dot{S}$, $u_i(x)_{dj} = u_i(x)_{dk}$. The latency of $u_i(x)$ is:

$$L_G = \alpha \times (\lambda/c) + L_1 \quad (1)$$

λ: the distance between d_j and d_k
c: the speed of the light, $c \approx 3^8$
L_1: latency at the datacenter hosting the receiver node.
α: see lemma 1
λ/c is bounded and rarely exceeds 0.1 therefore latency depends essentially on α factor.

2) Lemma 1: In unreliable and asynchronous distributed systems, the more the system is spread, the more **α** is increasing.

Linearizability implies that item replicated to multiple geographically different sites must be up to date at any moment.

$$u(x)_{dk} = x_0 \ \forall (d_k)_{k\geq 1} \in \dot{S} \quad (2)$$

Where x_0 is the latest value up to date of x

Linearizability requires a lot of message sending between datacenters of $\dot{S}$, hence we'll have:

$$\alpha = \sigma \times MSG\_OP \times IMPL\_DTC \times RF \ (3)$$

MSG_OP: average number of messages per operation
$IMPL\_DTC$: number of datacenters involved in U
RF: A redundancy factory defined to offset the lost messages
σ: a likelihood which increases with the distance between nodes, the number of routers, the number of messages.

It's clear that **α** increases seriously whether the number of involved datacenters grows (IMPL_DTC and **σ** are increasing).

C. Best level of consistency in multi-datacentric systems

Most cloud providers grant just the eventual consistency and claim that it should not be overtaken [15]. But some recent works proved that eventual consistency should be overcome [7,23]. Here we are showing that the causal consistency which is better than eventual consistency can be guaranteed in highly available systems.

1) Lemma 2:
"Any asynchronous and unreliable distributed system can ensure high availability."

When we have an asynchronous system, termination of operations is not affected because there's different timestamps for each node. Therefore, the availability can be guaranteed since operations should not be blocked. Whether a system is unreliable, messages may be dropped, reordered, or delayed in an undefined but finite duration. This may affect the correctness of the results (safety) but not their outcome (liveness). Furthermore, dropping messages does not preclude the eventual completion of the queries accordingly any operation should terminate. From the above remarks, we can conclude that the high availability can be implemented in such unreliable and asynchronous systems.

2) Proposition 2: "Any system which guarantees the causal consistency can accept concurrent writes."

It follows that for any execution *U* and *G* its HBR graph, all the write operations $(w_a, w_b) \in U$ such as $w_a \rightarrow w_b$ and $w_b \rightarrow w_a$ are not verified in *G*, can be concurrent.

doi:10.15849/icit.2015.0031

3) Theorem 3: "Any asynchronous and unreliable distributed system, which guarantees high availability, can ensure at most causal consistency."

We suppose a highly available system which ensures a consistency stronger than the causal consistency. For an execution in such systems $U=\{u_1 \ldots u_k\}_{k\geq l}$ and G its associated HBR graph; we proceed as follows:

We construct another execution $U'=\{u_1, \ldots u_n\}_{n>k}$ from U by joining it with $R=\{u_k \ldots u_n\}$ a set of read operations which control the implementation of U. We add the vertices corresponding to these reads to G and we obtain a new graph G_1; then we construct edges on G_1 in the following way: for each read in R we build an edge between it and all the non-local write operations in which it depends in G_1. The idea is to come establish a contradiction by processing a large number of operations; either we'll have concurrent write operations necessarily (no operation blocks due to the high availability). This means that:

- $\exists(w_i, w_j) \in (U')$, such as w_i and w_j are concurrent so neither $w_i \rightarrow w_j$ nor $w_i \rightarrow w_j$ should be verified in G_1. Therefore U' does not exceed the causal consistency which tolerate the concurrency between any pair of write operations which are not dependent.

- Finally based upon proposition 2, we can assert that strong consistency cannot be achieved in multi-datacentric systems.

D. Latency, convergence and high availability on multi-datacentric systems

1) Corollary 1 of lemma 2: "Any asynchronous and unreliable system which provides causal consistency can achieve a high availability."

2) Lemma 3: "Any multi-datacentric system which achieves causal consistency can guarantee strong convergence."

- Let $U=u_1, \ldots, u_k$ be an execution in such system and (w) an update of U on two replicas (R_1, R_2). Write operations are performed if for any operations (op_i) such as: $op_i \rightarrow w$, then op_i is performed before w at each replica. The idea is to commit entirely updates involving items on R_1 and R_2 $(u_i) \in U$. And thereafter R_1 exchange messages involving all the updates of U with R_2.

- Lemma 2 and theorem 3 allows us to assert that finally any read on the items updated on R_1 and R_2 will give the same result. Hence R_1 and R_2 are convergent, we can generalize this on any connected replica R_i and R_j of the system, $i, j \in \mathbb{N}$.

3) Lemma 4: "If we've causal consistency in an asynchronous and unreliable distributed system, we can improve drastically the latency."

(1) and (3) give:

$$L_G = \sigma \times (\lambda/c) \times MSG\_OP \times IMPL\_DTC + L_1 \quad (4)$$

RF can be ignored here; the messages are eventually delivered even if there's a failure or if messages are delayed for a finite duration. The causal consistency should be guaranteed at each node by his cache memory.

The number of messages across datacenters falls out in a spectacular way when we switch to the causal consistency. Consequently σ decreases drastically whether the number of message goes down; hence σ falls in whether we do not require higher than the causal consistency.

4) Theorem 4.4: "Any multi-datacentric distributed system which guarantees at most the causal consistency can provide the following properties: strong convergence, high availability and low latency."

This theorem is a corollary of the corollary 1 and the lemmas 4 and 5.

E. Corollary 5.1 (SCOLCH theorem)

Based on the foregoing; we state the following theorem: "On multi-datacentric systems we have an exclusive choice between the strong consistency in a side; and low latency, strong convergence, and high availability in another side."

IV. Related works

A little over a decade ago the CAP theorem [10] shook the world of distributed systems and has significantly influenced the actors of the cloud computing. Accordingly the consistency on distributed databases at a large scale aroused considerable interest recent years. Obviously a series of papers published about the CAP conjecture by Brewer, Gilbert and Lynch, Abadi, Ramakrishnan, Shim, and Birman [1,9,11,17,24,26] have reignited the debate on the tradeoffs in the distributed systems. Additionally, in earlier 2000's, Brzezinski has published a good work on the causal consistency. Afterwards, Mahajan [23] and Shapiro [25] works have highlighted the concept of convergence. Mahajan et al. [23] proved also that the real-time causal consistency was insurmountable in unreliable distributed systems. Bailis proposed highly available transactions (HAT) as an alternative for ACID databases in the cloud within a series of publications [4, 5, 6]. He proposed also the PBS (Probabilistically Bounded Staleness) which could minimize the staleness of data and guarantee a limited latency [7]; he discussed also the limitations of eventual consistency [4]. CALM conjecture (Consistency And Logical Monotonicity) was proven by Alvaro who proposed also 'coordination-free' distributed modals [3]. Furthermore, Lloyd et al.[21] presented the design and implementation of COPS, a key-value store that delivers a causal consistency model with convergent conflict handling; he proposed also a scalable, geo-replicated storage system that guarantees low latency [22]. Finally, T. Kraska et al. [18] have proposed MDCC: an optimistic commit protocol for geo-replicated transactions.

doi:10.15849/icit.2015.0031

V. CONCLUSION AND FUTURE WORK

The tradeoffs around the consistency property remain a nagging issue in multi-datacentric systems. In this paper we've lightened some basic concepts which are rarely explained by authors. Even if there are some theorems, the theoretical proofs are sorely lacking at this level, so we've given new theorems on these tradeoffs. Unlike the CAP theorem [10] which does not clearly take into account the latency and the convergence, our theorems prove that we can guarantee low latency, strong convergence and high availability in addition to the causal consistency which is proven to be better than the eventual consistency. In addition, our theorems prove that we cannot ensure the strong consistency without thereby sacrificing high availability and dealing with an unbridled growth of the latency.

In future work, it should be important clarify the fact that the very famous CAP theorem is not proved correctly and that CAP statement is out of context in actual distributed systems on the cloud.

ACKNOWLEDGMENT

We would like to thank the members of the database group of the doctoral school of mathematics and computer science of UCAD university of Dakar for their contributions to our research. We thank also the ICIT15 reviewers for their helpful feedback on this work.

REFERENCES

[1] D. J. Abadi, Consistency Tradeoffs in Modern Distributed Database System Design, IEEE Computer Society, vol. 45, no. 2, pp. 37-42. 2012.

[2] B., Alpern; F. B., Schneider. "Recognizing safety and liveness". Distributed Computing 2: pp. 117-126. (1987)

[3] P. Alvaro, N. Conway, J. M. Hellerstein, and W. R. Mar-czak. Consistency analysis in Bloom: a CALM and collected approach. In CIDR 2011.

[4] P. Bailis; A. Ghodsi. "Eventual Consistency Today: Limitations, Extensions, and Beyond". ACMQueue 11 : 20. April 2013.

[5] P. Bailis, A. Ghodsi, J. M. Hellerstein, and I. Stoica. Bolt-on causal consistency. SIGMOD 2013.

[6] P. Bailis, A. Davidson, A. Fekete, A. Ghodsi, J. M. Hellerstein, and I. Stoica. "Highly Available Transactions: Virtues and Limitations". In VLDB 2014.

[7] Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, and Ion Stoica. "Quantifying Eventual Consistency with PBS". Communication of the ACM, vol. 57, n°8. August 2014.

[8] P. Bernstein and S. Das. "Rethinking eventual consistency". In SIGMOD, 2013.

[9] K. Birman and al., "Overcoming CAP with Consistent Soft-State Replication", IEEE Computer Society, vol. 45, Issue: 2 pp. 50- 58, February 2012.

[10] E. Brewer, Towards Robust Distributed Systems, Portland, Oregon, Keynote at the ACM Symposium on Principles of Distributed Computing (PODC). July 2000.

[11] E. Brewer, Pushing the CAP: Strategies for Consistency and Availability, IEEE Computer Society, pp. 23-29. February 2012.

[12] J. Brzezinski, C. Sobaniec, and D. Wawrzyniak, "From session causality to causal consistency, in Proc. of 12th Euromicro Conf. on Parallel", Distributed and Network-Based Processing. Citeseer, pp. 152–158. 2004.

[13] F. Chang et al, "BigTable: A Distributed Storage System for Structured Data", Seventh Symposium on Operating System Design and Implementation, November 2006.

[14] B. Cooper et al. PNUTS: Yahoo!'s hosted data serving platform. In VLDB 2008.

[15] B. DeCandia and al., "Dynamo: Amazon‟s Highly Available Key-Value Store", Proceedings 21st ACM SIGOPS Symposium on B. F. Cooper, PNUTS: Yahoo!'s Hosted Data Serving Platform, Proc. VLDB Endowment (VLDB 08), pp. 1277-1288. 2008.

[16] P. Deutsch. "The eight fallacies of distributed computing." http://tinyurl.com/c6vvtzg, 1994.

[17] S. Gilbert and N. Lynch, Perspectives on the CAP theorem, IEEE Computer Society, vol. 45, no. 2, pp. 30-36. 2012.

[18] T. Kraska, G. Pang, M. Franklin, and S. Madden. "Mdcc: Multi-data center consistency". In Eurosys 2013.

[19] A. Lakshman and P. Malik, Cassandra: a decentralized structured storage system, ACM SIGOPS Operating Systems Review, vol. 44, , pp. 35–40. 2010.

[20] L. Lamport, L.. "Proving the Correctness of Multiprocess Programs". IEEE Transactions on Software Engineering : (1977) march, pp. 125–143.

[21] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen. "Don't Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS". In SOSP 2011.

[22] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen. Stronger semantics for low-latency georeplicated storage. In NSDI 2013.

[23] P. Mahajan , L. Alvisi , M. Dahlin. Consistency, Availability, and Convergence. Department of Computer Science, University of Texas at Austin. Technical Report (UTCS TR-11-22).2012.

[24] R. Ramakrishnan, CAP and Cloud Data Management, IEEE Computer Society, vol. 45, Issue: 2 pp. 43 -49, February 2012.

[25] M. Shapiro, Convergent and Commutative Replicated Data Types, Bulletin of the EATCS, no. 104, pp. 67-88. June 2011.

[26] S. S. Y. Shim, CAP theorem's growing impact, IEEE Computer Society, vol. 45 , Issue: 2 pp 20 -21, February 2012.

[27] M. Stonebraker, Errors in Database Systems, Eventual Consistency, and the CAP Theorem, blog, Comm. ACM, http://cacm.acm.org/blogs/blog-cacm/83396-errors-in-database-systems-eventualconsistency-and-thecap-theorem. 2010.

[28] M. Stonebraker, Clarifications on the CAP Theorem and Data-Related Errors, VoltDB blog, http://blog.voltdb.com/clarifications-cap-theorem-and-data-related-error. 2010.

[29] W. Vogels, "Eventually Consistent", ACM Queue, vol. 6, n° 6, pp. 14-19. 2008.