A System Infrastructure for Strongly Consistent Transactions on Globally-Replicated Data

Faisal Nawab Vaibhav Arora Victor Zakhary Divyakant Agrawal Amr El Abbadi

Department of Computer Science, University of California, Santa Barbara
Santa Barbara, CA 93106
{nawab,vaibhavarora,victorzakhary,agrawal,amr}@cs.ucsb.edu

Abstract

Global-scale data management (GSDM) empowers systems by providing higher levels of fault-tolerance, read availability, and efficiency in utilizing cloud resources. This has led to the emergence of global-scale data management and event processing. However, the Wide-Area Network (WAN) latency separating datacenters is orders of magnitude larger than typical network latencies, and this requires a reevaluation of many of the traditional design trade-offs of data management systems. Therefore, data management problems must be revisited to account for the new design space. In this paper, we revisit the problem of supporting strongly-consistent transaction processing for GSDM. This includes providing an understanding of the limits imposed by the WAN latency on transaction latency in addition to a design of a system framework that aims to reduce response time and increase scalability. This infrastructure includes a transaction processing component, a fault-tolerance component, a communication component, and a placement component. Finally, we discuss the current challenges and future directions of transaction processing in GSDM.

1 Introduction

The cloud computing paradigm promises high-performance 24/7 service to users dispersed around the world for cloud applications. Achieving this is threatened by complete datacenter outages and the physical limitations of both the datacenter infrastructure and wide-area communication. To overcome these challenges, systems are increasingly being deployed on multiple datacenters spanning large geographic regions. The replication of data across datacenters (geo-replication) allows requests to be served even in the event of complete datacenter-scale outages. Likewise, distributing the processing and storage across datacenters brings the application closer to users and sources of data, enabling higher levels of availability and performance. Moving to global-scale data management (GSDM), despite its benefits, raises many novel challenges. One of the main sources of these challenges is the large WAN communication latency, which is orders of magnitude larger than traditional communication latency (See Figure 1). This invalidates the traditional space of design trade-offs and makes the communication latency the dominating bottleneck.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

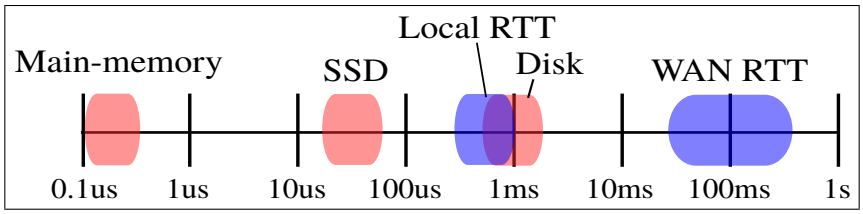

Figure 1: Latency of the Wide-Area Network Round-Trip Time communication (WAN RTT) compared to memory access latency [27] and network latency within the datacenter (local RTT)

When coordination is involved in the data management task, the effect of WAN communication latency is amplified. This makes transaction processing—a coordination-intensive task—a victim of geo-replication's large communication latency. The significance of transaction processing to data management systems and the seriousness of the WAN communication challenge have led many researchers from both academia and industry to design new and improved transaction processing protocols specifically for geo-replication [2, 3, 6–10, 13, 15, 16, 18–20, 22–25, 28, 31, 35]. Many of these efforts explored weakening consistency guarantees to lower the coordination demands [7–9, 13, 15, 16, 18, 19, 22, 24, 31] or explored coordination-free approaches to avoid the cost of wide-area coordination [2, 28, 35]. However, a large-body of work tackled the problem of global-scale transaction processing while maintaining the stringent consistency guarantees of traditional data management systems [3, 6, 10, 20, 23, 25]. The argument for maintaining stringent guarantees is that they manifest easy-to-use abstractions for concurrent access to data. Serializability, for example, ensures an outcome equivalent to a serial execution [4], ridding the program developers from worrying about concurrency anomalies.

In this paper, we present a system infrastructure design for global-scale transaction processing with serializability guarantees. We adopt a holistic approach where in addition to the transaction processing engine, we improve the design of the surrounding system components that affects the performance of transactions. We begin the paper by discussing the fundamental limits of transaction processing in geo-replication and develop a theoretical framework of the optimality of transaction latency. Then, we present transaction processing protocols that leverage the newfound knowledge about transaction latency optimality to achieve better performance in geo-replication. Then, we present two components outside of the transaction processing engine that are essential for transactions processing efficiency in geo-replicated systems. The first is global-scale communication and the second is data placement. We conclude the paper with an outlook on the future of geo-replicated transaction processing in the context of the advancements in global-scale data management systems.

2 Transaction Processing

In this section, we present the development of transaction processing protocols that target reducing transaction latency on geo-replicated data. This begins by understanding the source of the latency inefficiency in traditional transaction processing systems (Section 2.1). Then, we develop a lower-bound formulation for transaction latency on geo-replicated data (Section 2.2) and use this newfound understanding to develop Helios [23], a transaction processing protocol that achieves the lower-bound (Section 2.3).

2.1 The Latency Limit of the Request-Response Paradigm

There is a fundamental coordination latency limit due to the polling nature of traditional protocols that we call the Request-Response paradigm. In the Request-Response paradigm, the coordination for a request starts *after* the request is made, where the node making the request polls other nodes to inquire about their state and detect conflicts. The request is served only after receiving a *response* from other replicas. This makes a Round-Trip

Time (RTT) of communication inevitable—an expensive cost in GSDM. This applies to both centralized and quorum-based protocols.

The limit of the Request-Response paradigm leads to the question: *Is it possible to avoid the Request-Response paradigm limit on coordination or is it a fundamental limit on performance?* Our work Message Futures [21] demonstrates the possibility of breaking the limit of the Request-Response paradigm by an observation that coordination of future requests can start before they arrive. As requests arrive, they are assigned to a predetermined future coordination point. We call this approach *Proactive Coordination*. Coordination points are judiciously calculated to ensure conflicts are detected. A coordination point still needs an RTT for coordination. However, because a request is assigned to a coordination point that already started, the request's observable latency is less than RTT. Message Futures is the first protocol that shows the possibility of faster-than-RTT coordination for all the replicas of a distributed system. Also, it introduces Proactive Coordination, a new approach to coordination that overcomes the limitations of the Request-Response paradigm.

2.2 Theoretical Lower-Bound on Coordination Latency

Breaking the RTT latency barrier via the Proactive Coordination paradigm invalidates the previously held convention that coordination cannot be performed faster than the RTT latency. Thus, it opens the question: *What is the lower-bound on coordination latency?* This is a fundamental question in understanding the extent of the effect of the wide-area latency limit on coordination latency. Such a lower-bound, if proven, will provide system designers and researchers with a theoretical foundation on what is achievable by current and future systems.

Our objective now is to develop a lower-bound on commit latency of transactions on replicated data stores *while maintaining serializability* [4]. Maintaining serializability requires coordination between replicas (datacenters in our case). The communication latency necessary for this coordination imposes a limit on *commit latency*, which is the time duration to decide whether a transaction commits or aborts. Achieving low commit latency is the focus of this study. Consider two datacenters A and B with unique commit latencies L_A and L_B, respectively. We show in this section that the *summation* of L_A and L_B must be at least the Round-Trip Time (RTT) between A and B. Note that this is a summation which means that the commit latency of a datacenter can be lower than RTT.

The lower-bound result extends to larger groups of datacenters by applying the lower-bound to all pairs in the group. This will allow us to judge whether the group of datacenters can commit with a certain set of commit latency values. We are particularly interested in minimizing the average commit latency of all datacenters. We call the minimum average latency a *Minimum Average Optimal* (MAO) latency or optimal latency for short.

2.2.1 Theoretical model and assumptions

We consider a theoretical model that consists of datacenters with communication links connecting them. Each transaction undergoes two phases. First, the transaction is issued and it becomes visible to the datacenter. At that stage it is called a *preparing* transaction. Then, at a later time the datacenter decides whether it commits or aborts and it becomes a *finished* transaction. The time spent as a preparing transaction is the *commit latency*.

The following are the assumptions on communication and computation for this model.

- *Compute power:* Infinite compute power is assumed in the model. The datacenter does not experience any overhead in processing and storing transactions. We make this assumption to focus our attention on communication overhead.

- *Communication links:* Sending a message through a link takes a specific latency to be delivered to the other end. Links are symmetric and take the same amount of time in both directions. Note that different links could have different latencies. However, triangle inequality must hold.

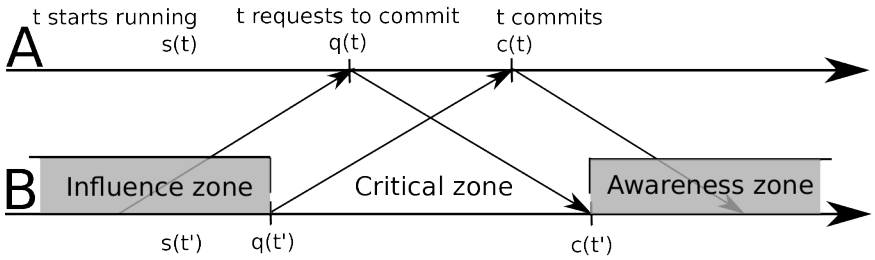

Figure 2: Two transactions, t and t', executing in a scenario with two datacenters

- *Arbitrary read-write transactions:* All datacenters have no restrictions on their choice or order of objects to be read or written in a transaction. Additionally, each transaction must have at least a single write operation. Thus, the model does not apply to optimizations for read-only transactions and disjoint data manipulation techniques. Also, transactions must try to commit, hence aborting all transactions is not allowed.

- *Knowledge:* Each datacenter A knows precisely every preparing and finished transaction that exists at another datacenter B up to the current time minus half the RTT between them, *i.e.*, now - $\frac{RTT(A,B)}{2}$. This reflects the fastest time a datacenter knows about any event in another datacenter. In a realistic setting this is a lower bound of such knowledge.

- *Commit latency:* We assume that the commit latency at each datacenter is fixed. This assumption simplifies the presentation. The discussions can be extended to the general case by taking each point in time in isolation.

2.2.2 Lower-bound proof

Intuition. For any two concurrent conflicting transactions, at least one of them must be able to detect the other before committing. Otherwise, both transactions will commit, which could result in incorrect executions. Here, we show that there is a lower-bound on commit latency. If the commit latency is lower than the lower-bound, then two conflicting transactions could commit without detecting each other, thus possibly violating correctness.

Formulation. Consider two datacenters A and B and a transaction t executing at datacenter A and transaction t' executing at datacenter B that could be conflicting with t. Figure 2 shows these transactions. In the figure, $s(t)$ is the transaction's start time, $q(t)$ is the commit request time, and $c(t)$ is the commit time. Transaction t's read and write-set are visible at the commit request time. Given the knowledge assumption, B knows about t starting from time $q(t) + \frac{RTT(A,B)}{2}$. Transaction t is preparing from time $q(t)$ until time $c(t)$ when it is committed. Three *zones* are defined at datacenter B with respect to t: (1) The *awareness zone* where B can possibly know about t, (2) The *influence zone* where B's transactions can be known to t, and (3) the *critical zone* where B is neither in the awareness nor influence zone.

Lemma 1: The sum of the commit latencies of two datacenters is greater than or equal to the RTT between them, *i.e.*, $L_A + L_B \geq RTT(A, B)$, where L_X is the commit latency at datacenter X.

Now, we present a description of the intuition behind this lemma (based on the proof in the original paper [23]). Consider the scenario in Figure 2 . The time when t requests to commit is $q(t)$ and the time it commits is $c(t)$, *i.e.*, $c(t) - q(t) = L(t)$. Based on transaction t, we can divide the timeline in B to three regions: (1) *Awareness zone:* this zone contains the events at B that can be affected by t. This zone starts from the earliest point of time

Protocol	L_A	L_B	L_C	Average
Leader-based (A leader)	0	30	20	16.67
Leader-based (C leader)	20	40	0	20
Majority	20	30	20	23.33
Optimal (MAO)	5	25	15	15

Table 1: Possible commit latencies, L_A, L_B and L_C, for three datacenters with Round-Trip Times $RTT(A, B) = 30$, $RTT(A, C) = 20$, and $RTT(B, C) = 40$.

when B can receive t, which is $q(t) + \frac{RTT(A,B)}{2}$. (2) *Influence zone:* this zone contains the events that can affect the outcome of t. An event can affect the outcome of t if it is received before the commit point, $c(t)$. Events at B that can arrive by the commit point are ones with a timestamp lower than $c(t) - \frac{RTT(A,B)}{2}$. Events at B with a higher timestamp cannot be received at A by time $c(t)$ because half an RTT is required to communicate. (3) *Critical zone:* this zone represents the time duration that is neither in the awareness zone nor in the influence zone. We postulate that it is impossible for a transaction to have requested to commit and commits in the critical zone. To show this, consider a transaction t' at B that requests to commit and commits in the critical zone. Transaction t' will not affect the outcome of t, since t' is not in the influence zone. Also, t will not affect t', since t' is not in the awareness zone. However, t' can conflict with t. Since t' is not aware of t and t is, likewise, not aware of t', both transactions successfully commit. This potentially results in an inconsistency, and is thus not allowed. To summarize, a transaction that starts at the beginning of the critical zone at B cannot commit with a commit latency smaller than the duration of the critical zone. This duration is equal to $RTT(A, B) - L(t)$. Thus, *the sum of L_A and L_B must be greater than or equal to $RTT(A, B)$.*

The lower-bound shows a direct trade-off between the commit latencies of two datacenters. Given this lower-bound we are now able to judge whether a set of commit latencies are *achievable* or violates the lower-bound for scenarios with more than two datacenters by applying the lower-bound to each pair of datacenters.

Example. Consider an example of three datacenters, A, B, and C. The RTTs between the datacenters are: $RTT(A, B) = 30$, $RTT(A, C) = 20$, and $RTT(B, C) = 40$. Table 1 shows four achievable commit latencies and the average commit latency of the datacenters. The first two represent a leader-based replication approach, where a single leader is responsible for committing transactions. In this approach, the leader commits immediately, and the other datacenters commit latencies are the RTT to the leader. Note how each pair of datacenters satisfies the lower-bound, *e.g.*, when A is the leader $L_A + L_B = 30 = RTT(A, B)$. The third row represents a majority replication approach. For the case of three datacenters, the commit latency of a datacenter is the RTT to the nearest datacenter. These replication protocols experience different average commit latencies: 16.67, 20, and 23.33. However, the minimum average commit latency (MAO) that is achievable for this scenario is 15. The fourth row in the figure show the commit latencies, L_A, L_B, and L_C, that achieve an average commit latency of 15 while not violating the lower-bound.

MAO solutions, such as the one in the previous example, can be derived using the following linear programming formulation:

Definition 2: (Minimum Average Optimal)
The Minimum Average Optimal commit latencies for n datacenters is derived using a linear program with the following objective and constraints:

$$\begin{array}{lll} \text{Minimize} & \sum_{A \in R} & L_A \\ \text{subject to} & \forall_{A,B \in R} & L_A + L_B \geq RTT(A, B) \\ \text{and} & \forall_{A \in R} & L_A \geq 0 \end{array}$$

where R is the set of datacenters. This formulation follows directly from Lemma 1. Minimizing the latency

is our objective and the constraints are the correctness conditions that commit latencies are not negative and Lemma 1 is satisfied. We will use this methodology to derive the commit latency values used with the Helios commit protocol. This linear program can be adapted to objectives other than average latency [23].

2.2.3 Summary

The lower-bound result shows that the coordination latency can be faster than what is previously achieved by traditional protocols and even faster than what is achieved by Message Futures. The model of coordination, in addition to being essential for deriving the lower-bound, advances our understanding of the cost of global-scale coordination. It also brings a newfound understanding of the latency characteristics of traditional and Proactive Coordination protocols.

2.3 Achieving the Lower-Bound Transaction Latency

The lower-bound model inspired a protocol based on Proactive Coordination, called **Helios** [23] that theoretically achieves the lower-bound, thus proving that the lower-bound is tight. Experimental evaluation shows that Helios approaches the lower-bound in a real global-scale deployment. Next, we provide an intuition of how Helios achieves the lower-bound and refer the interested reader to the full paper [23].

To provide an intuition of the Helios commit protocol, consider the scenario in Figure 2. The figure shows the timeline of two datacenters, A and B. At A, a transaction t is issued at time $q(t)$ and committed at time $c(t)$. Transaction t commits immediately after receiving a log of events from B that is shown as an arrow going from B to A. This log carries transactions that were issued up to the time of sending the log, including transaction t' (assume t' is issued at the time of log transmission). The time the log was sent from B is $q(t')$. $q(t')$ is also the commit request time of t'. Helios receives the log in order (via a FIFO channel), meaning that all transactions, preparing or finished, at B prior to or at time $q(t')$ are known to A at time $c(t)$.

Transactions at B must not conflict with t. The approach to avoid conflicts is influenced by the way the lower-bound latency was developed in Section 2.2. However, here we do not make any assumptions regarding clock synchronization or communication. Rather, we rely on the exchanged event logs and received transaction timestamps. A transaction, t', at B is either issued during the influence zone, critical zone, or awareness zone. If t' starts during the influence zone, then transaction t will detect it because the log will contain a record of t'. If t' starts in the awareness zone, then it will detect t. Thus, for these two cases, conflicts will be detected. An undetected conflict can arise only if t' starts *and* commits within the critical zone. Thus, if t' is issued in the critical zone, Helios must ensure that it does not commit until it is in the awareness zone, which means that B will detect the conflict between t and t'.

3 Global-Scale Data Communication

In the previous section, we have tackled the problem of coordination latency and building transaction commit protocols that ameliorate the impact of the large WAN communication latency. However, large applications that receive large amounts of requests may face the problem of *scaling* inter-datacenter communication. Typical communication protocols used by data management systems are not built for WAN environments and large communication demands. This leads to the increase in the demand and stress on the network I/O, which translates into significant communication latency overhead. To combat this challenge, we developed Chariots [22], a scalable inter-datacenter communication platform. Chariots implement a communication layer to be used by transaction processing engines such as Message Futures and Helios that we presented earlier. Chariots maintains the causal order of the communicated events. This is because Message Futures and Helios rely on a causally consistent communication solution called the Replicated Dictionary [33]. However, these communication solutions cannot scale, and Chariots offer a scalable alternative for causally-consistent communication systems.

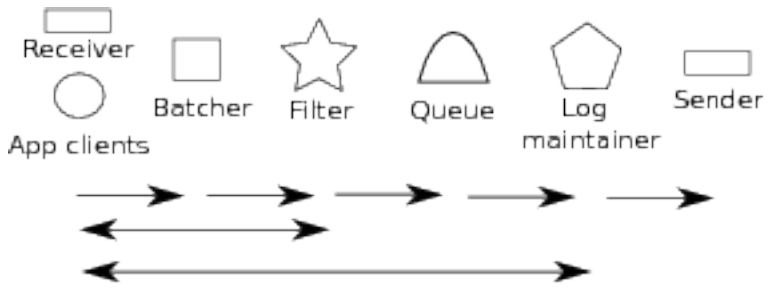

Figure 3: The components of the multi-data center shared log. Arrows denote communication pathways in the pipeline.

To enable higher levels of scalability, Chariots adapts the following design features:

- (1) Highly-available stateless control. This approach has been identified as the most suitable to scale out in cloud environments [11]. Communicated data is represented in the form of immutable records in a distributed shared log. The architecture of Chariots consists of three types of machines: *Log maintainers* are responsible for persisting the log's records and serving read requests. *Indexers* are responsible of access to log maintainers. Finally, control and meta-data management is the responsibility of a highly-available cluster called the *Controller*.

- (2) Fast data ingestion. To achieve this, Chariots implements a distributed log that allows inserting records arbitrarily to any log node without coordination with other log nodes. All coordination is done lazily in the background, in summarized batches, to reduce the impact on ingesting new records.

- (3) Efficient causal-dependency enforcement. To achieve this, a pipeline design is adopted to process inter-datacenter communication. Chariots pipeline consists of six stages depicted in Figure 3. The first stage contains nodes that are generating records. These are Application clients and machines receiving the records sent from other datacenters. These records are sent to the next stage in the pipeline, *Batchers*, to batch records to be sent collectively to the next stage. *Filters* receive the batches and ensure the uniqueness of records. Records are then forwarded to Queues where they are ordered. After it is ordered, a record is forwarded to a log node that constitutes the Log maintainers stage. The local records in the log are read from the log nodes and sent to other datacenters via the *Senders*.

  The arrows in Figure 3 represent the flow of records. Generally, records are passed from one stage to the next. However, there is an exception. Application clients can request to read records from the Log maintainers. Chariots support elastic expansion of each stage to accommodate increasing demand. Thus, each stage can consist of more than one machine, *e.g.*, five machines acting as Queues and four acting as Batchers.

4 Global-Scale Data Placement

The success of the cloud model has led to the continuing increase of the number of public cloud providers in addition to the increase in the amount of resources offered by these providers. This includes an increase in the number of available physical datacenters, which raises the question: *at which datacenters should data be placed?* This is called the Global-Scale Data placement problem, or placement problem for short. The challenging aspect of the placement problem, is that developers want to optimize a diverse set of objectives that are sometimes in conflict, such as monetary cost and performance. This problem has been tackled by providing frameworks to reason about placement decisions [1, 17, 26, 30, 32]. However, these solutions are not suitable for

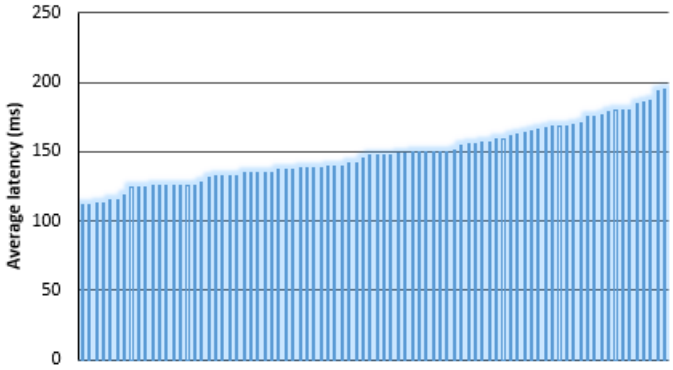

Figure 4: The average latency, of all the clients in 9 datacenters, to reach the closest quorum (2 out 3) for all the possible $\binom{9}{3} = 84$ different placements sorted by latency.

our system infrastructure because they target systems with relaxed access abstractions [1, 17, 26, 32] or systems with different coordination patterns [30].

To fill the gap in global-scale placement systems, we present DB-Risk [34], a data placement framework that targets multi-leader strongly-consistent transactional systems. DB-Risk embeds the commit protocol constraints into an optimization to derive both the data placement and the commit protocol configurations that minimize the overall transaction latency. Figure 4 shows the effect of only changing the placement on the average obtained latency for all the clients in a multi-leader protocol (*i.e.*, Paxos). Clients are equally distributed at nine of Amazon's datacenters in California, Oregon, Virginia, Sao Paulo, Ireland, Sydney, Singapore, Tokyo, and Seoul. As shown in Figure 4, changing only the placement while fixing all the other parameters (the protocol, the workload distribution, etc.) can lead to a significant change of 1.75x between the minimum and the maximum reported average latency.

In developing DB-Risk, we discovered counter-intuitive lessons about data placement and transaction execution practices. The most notable lesson is what we call *Request Handoff*, which is choosing the datacenter that will drive the execution of the transaction (the coordinator). The transaction latency is mainly affected by the distance between the client and the coordinator, the distance between the coordinator and the other participants, and finally the number of communication rounds required between the coordinator and the participants to serve the request. Different transaction management protocols choose the coordinator based on some intuitive heuristics, such as choosing the closest replica to the client. However, the choice of the coordinator can, in fact, drastically affect the request latency.

Request Handoff, in addition to other lessons, can be used to achieve better performance by being aware of the latency diversity and asymmetry of the WAN links. They are also applicable to both Paxos-based commitment protocols [3, 20, 25] and leader-based commitment protocols [6, 10]. DB-Risk incorporates these lessons, the commitment protocol constraints, and the application requirements in an optimization to find the placement that minimizes the average transaction latency.

5 The Future of Global-Scale Transaction Processing

In this section, we discuss some of the open challenges and future directions in the area of global-scale transaction processing.

Edge Resources. Emerging edge datacenter technologies, such as micro datacenters and cloudlets, have the potential to bring data even closer to end users. This has motivated many mobility and Internet of Things (IoT) systems to adopt the *edge computing model*, also called by other names such as fog computing [5]. However, *for*

data management tasks that require updating data, many edge computing systems still rely on storage solutions that do not utilize edge datacenters. We envision extending GSDM system beyond traditional datacenters to cover edge datacenters. Such a direction will complement and enhance existing edge computing frameworks by providing a unified storage abstraction at the edge that is globally synchronized and supports transaction processing. Edge data management enjoys the benefits—and faces the challenges—of edge computing [29]. For example, in this new model, the data management system can leverage edge datacenters for tasks such as maintaining copies for fault-tolerance and to offload smaller instances to serve requests at edge location close to users. The incorporation of edge datacenters changes the model of geo-replication. The number of replicas is no longer confined to a small number of datacenters, rather to a potentially large number of edge datacenters. Also, the Round Trip Time (RTT) between a datacenter and nearby edge datacenters is an order of magnitude lower than typical inter-datacenter RTT. Significant research and practical effort is needed to accommodate GSDM systems to this new environment. However, utilizing edge datacenters will improve the performance of GSDM systems and will enable emerging edge and mobile applications.

Flexible Fault-tolerance. Even with GSDM and efficient coordination, achieving a low end-to-end latency is a challenge. This is due to the needed latency to coordinate access between different replicas to maintain consistency, and the latency to replicate data across datacenters for fault-tolerance. Reducing and controlling the latency of coordination by relaxing consistency has been the topic of many research efforts including ours. *The remaining frontier is investigating relaxing fault-tolerance to reduce and control the need for synchronous WAN communication.* We argue that analogous to how some applications may have relaxed consistency requirements, some applications may have relaxed fault-tolerance requirements. We envision an exploration of the trade-off between fault-tolerance and latency in the context of edge data management, while preserving strongly consistent abstractions. Such exploration may lead to methods to control the fault-tolerance level in a way that result in achieving higher performance for weaker fault-tolerance levels. Also, there may be methods to relax the requirements of fault-tolerance and find a spectrum of durability guarantees with various performance characteristics.

Emerging WAN Techniques. Leveraging advances in WAN research from the networking community is an important step towards building efficient GSDM systems. Networking techniques, like Software-defined Networking (SDN), are now being applied to the context of WANs (*e.g.*, BwE [14] and B4 [12]). A promising opportunity is to develop GSDM systems that integrate these advances in networking.

6 Concluding Remarks

We believe that to make GSDM a reality, it is essential to provide intuitive and easy-to-use abstractions for application developers. Our goal is to enable web and cloud programmers to build their applications to benefit from the opportunities of GSDM. To achieve this, we focused on providing abstractions at the database layer that are intuitive by providing strong consistency, thus ridding the programmer from thinking about concurrency, replication, and other complexities related to the GSDM environment.

Each one of our proposed protocols treats wide-area coordination as the main bottleneck. This approach has turned to be rewarding in terms of novel designs that achieve a much higher performance than traditional counterparts in GSDM environments. We believe that the principles we propose in these protocols will impact the design of a wide-range of problems that share the aspect of wide-area coordination.

We handle strongly consistent transaction processing in GSDM. We have proposed a theoretical formulation of the performance limit imposed by wide-area latency, in addition to a transaction management paradigm called proactive coordination. The theoretical formulation enables finding a lower-bound transaction latency in global-scale environments. This lower-bound result provides a guide to system designers and researchers to reason about latency limits in multi-datacenter environments. Proactive coordination is a paradigm for transaction commit protocols where coordination for future transactions start before they are issued. We have shown how

this general concept can be used to implement two GSDM transaction commit protocols: (1) Message Futures, that breaks the RTT barrier for transaction latency, and (2) Helios, that is able to achieve transaction latency close to the theoretical lower bound. Message Futures and Helios combine traditional concurrency control approaches such as the use of timestamp ordering, log propagation, loose-time synchronization, and certification with the concepts that we propose in the paper such as proactive coordination and lower-bound latency.

In the course of developing global-scale systems, we encountered a common challenge in managing communication between globally-distributed nodes. This motivated Chariots, our work to provide a communication platform for GSDM systems. Chariots maintains a shared log abstraction between nodes in various datacenters. Chariots targets both scalability within the datacenter and across datacenters. Within the datacenter, Chariots includes a distributed shared log system, which removes coordination from the path of appending operations. Chariots then provides a framework to replicate distributed, shared logs across datacenters. Chariots guarantees causal order of events in the shared log, which is sufficient to implement strongly consistent protocols on top of it, including Message Futures and Helios. Also, we addressed the data placement problem of geo-replicated databases and find that a special treatment for strongly-consistent systems is needed. We envision that the presented designs will aid and inspire future protocols in global-scale data management.

References

[1] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan. Volley: Automated data placement for geo-distributed cloud services. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 2–2, Berkeley, CA, USA, 2010. USENIX Association.

[2] P. Bailis, A. Fekete, M. J. Franklin, A. Ghodsi, J. M. Hellerstein, and I. Stoica. Coordination avoidance in database systems. *Proc. VLDB Endow.*, 8(3):185–196, Nov. 2014.

[3] J. Baker, C. Bond, J. C. Corbett, J. J. Furman, A. Khorlin, J. Larson, J. Leon, Y. Li, A. Lloyd, and V. Yushprakh. Megastore: Providing scalable, highly available storage for interactive services. In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings*, pages 223–234, 2011.

[4] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[6] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford. Spanner: Google's globally-distributed database. pages 251–264, 2012.

[7] K. Daudjee and K. Salem. Lazy database replication with snapshot isolation. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 715–726. VLDB Endowment, 2006.

[8] J. Du, S. Elnikety, A. Roy, and W. Zwaenepoel. Orbe: Scalable causal consistency using dependency matrices and physical clocks. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 11:1–11:14, New York, NY, USA, 2013. ACM.

[9] J. Du, S. Elnikety, and W. Zwaenepoel. Clock-si: Snapshot isolation for partitioned data stores using loosely synchronized clocks. In *Proceedings of the 2013 IEEE 32Nd International Symposium on Reliable Distributed Systems*, SRDS '13, pages 173–184, Washington, DC, USA, 2013. IEEE Computer Society.

[10] L. Glendenning, I. Beschastnikh, A. Krishnamurthy, and T. Anderson. Scalable consistency in scatter. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 15–28, New York, NY, USA, 2011. ACM.

[11] A. Gupta, F. Yang, J. Govig, A. Kirsch, K. Chan, K. Lai, S. Wu, S. G. Dhoot, A. R. Kumar, A. Agiwal, S. Bhansali, M. Hong, J. Cameron, M. Siddiqi, D. Jones, J. Shute, A. Gubarev, S. Venkataraman, and D. Agrawal. Mesa: Geo-replicated, near real-time, scalable data warehousing. *Proc. VLDB Endow.*, 7(12):1259–1270, Aug. 2014.

[12] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a globally-deployed software defined wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 3–14, New York, NY, USA, 2013. ACM.

[13] T. Kraska, G. Pang, M. J. Franklin, S. Madden, and A. Fekete. Mdcc: Multi-data center consistency. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 113–126, New York, NY, USA, 2013. ACM.

[14] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, M. Robin, A. Siganporia, S. Stuart, and A. Vahdat. Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 1–14, New York, NY, USA, 2015. ACM.

[15] Y. Lin, B. Kemme, M. Patiño Martínez, and R. Jiménez-Peris. Middleware based data replication providing snapshot isolation. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 419–430, New York, NY, USA, 2005. ACM.

[16] Y. Lin, B. Kemme, M. Patino-Martinez, and R. Jimenez-Peris. Enhancing edge computing with database replication. In *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems*, SRDS '07, pages 45–54, Washington, DC, USA, 2007. IEEE Computer Society.

[17] G. Liu and H. Shen. Minimum-cost cloud storage service across multiple cloud providers. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 129–138. IEEE, 2016.

[18] 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 *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 401–416, New York, NY, USA, 2011. ACM.

[19] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen. Stronger semantics for low-latency geo-replicated storage. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pages 313–328, Berkeley, CA, USA, 2013. USENIX Association.

[20] H. Mahmoud, F. Nawab, A. Pucher, D. Agrawal, and A. El Abbadi. Low-latency multi-datacenter databases using replicated commit. *Proc. VLDB Endow.*, 6(9):661–672, July 2013.

[21] F. Nawab, D. Agrawal, and A. El Abbadi. Message futures: Fast commitment of transactions in multi-datacenter environments. In *CIDR 2013, Sixth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*, 2013.

[22] F. Nawab, V. Arora, D. Agrawal, and A. El Abbadi. Chariots: A scalable shared log for data management in multi-datacenter cloud environments. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.*, pages 13–24, 2015.

[23] F. Nawab, V. Arora, D. Agrawal, and A. El Abbadi. Minimizing commit latency of transactions in geo-replicated data stores. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1279–1294, New York, NY, USA, 2015. ACM.

[24] G. Pang, T. Kraska, M. J. Franklin, and A. Fekete. Planet: Making progress with commit processing in unpredictable environments. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 3–14, New York, NY, USA, 2014. ACM.

[25] S. Patterson, A. J. Elmore, F. Nawab, D. Agrawal, and A. El Abbadi. Serializability, not serial: Concurrency control and availability in multi-datacenter datastores. *Proc. VLDB Endow.*, 5(11):1459–1470, July 2012.

[26] F. Ping, J.-H. Hwang, X. Li, C. McConnell, and R. Vabbalareddy. Wide area placement of data replicas for fast and highly available data access. In *Proceedings of the fourth international workshop on Data-intensive distributed computing*, pages 1–8. ACM, 2011.

[27] M. K. Qureshi, S. Gurumurthi, and B. Rajendran. *Phase Change Memory: From Devices to Systems*. Morgan & Claypool Publishers, 1st edition, 2011.

[28] S. Roy, L. Kot, G. Bender, B. Ding, H. Hojjat, C. Koch, N. Foster, and J. Gehrke. The homeostasis protocol: Avoiding transaction coordination through program analysis. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1311–1326, New York, NY, USA, 2015. ACM.

[29] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[30] A. Sharov, A. Shraer, A. Merchant, and M. Stokely. Take me to your leader!: Online optimization of distributed storage configurations. *Proc. VLDB Endow.*, 8(12):1490–1501, Aug. 2015.

[31] Y. Sovran, R. Power, M. K. Aguilera, and J. Li. Transactional storage for geo-replicated systems. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 385–400, New York, NY, USA, 2011. ACM.

[32] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha. Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 292–308, New York, NY, USA, 2013. ACM.

[33] G. T. Wuu and A. J. Bernstein. Efficient solutions to the replicated log and dictionary problems. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, PODC '84, pages 233–242, New York, NY, USA, 1984. ACM.

[34] V. Zakhary, F. Nawab, D. Agrawal, and A. El Abbadi. Db-risk: The game of global database placement. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 2185–2188, New York, NY, USA, 2016. ACM.

[35] Y. Zhang, R. Power, S. Zhou, Y. Sovran, M. K. Aguilera, and J. Li. Transaction chains: Achieving serializability with low latency in geo-distributed storage systems. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 276–291, New York, NY, USA, 2013. ACM.