Annotated Bibliography on Transactions

Hengfeng Wei
hfwei@nju.edu.cn

1 Books

Bernstein, P. A., Hadzilacos, V., and Goodman, N. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Longman Publishing Co., Inc., USA, 1987

Weikum, G., and Vossen, G. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001

It is researchers-oriented. Highly recommended.

2 Transactional Consistency Models

2 Frameworks

Gray, J. N., Lorie, R. A., Putzolu, G. R., and Traiger, I. L. *Granularity of Locks and Degrees of Consistency in a Shared Data Base*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994, 181–208

This paper defines four degrees of consistency, attempting to show the equivalence of locking, dependency, and anomaly-based characterizations. The anomaly definitions were too vague. The authors continue to get criticism for that aspect of the definitions. Only the more mathematical definitions in terms of histories and dependency graphs or locking have stood the test of time.

This is a book chapter version. Originally published in 1977. See also [37].

Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E., and O'Neil, P. A critique of ansi sql isolation levels. *SIGMOD Rec. 24*, 2 (May 1995), 1–10

Defines Isolation Levels in terms of phenomena; Introduces new phenomena; Define Snapshot Isolation.

Atluri, V., Bertino, E., and Jajodia, S. A theoretical formulation for degrees of isolation in databases. *Information and Software Technology 39*, 1 (1997), 47 – 53

This paper formulates these different degrees of isolation in terms of histories, as in the case of the usual serialization theory and proposes timestamp-based protocols for different degrees of isolation.

Adya, A., and Liskov, B. H. *Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions*. PhD thesis, USA, 1999. AAI0800775

This thesis presents the first implementation-independent specifications of existing ANSI isolation levels and a number of levels that are widely used in commercial systems, e.g., Cursor Stability, Snapshot Isolation.

We use a graph-based approach to define different isolation levels in a simple and intuitive manner.

The thesis describes new implementation techniques for supporting different weak consistency levels in distributed client-server environments.

Generalized isolation level definitions. In *Proceedings of the 16th International Conference on Data Engineering*, ICDE '00, IEEE Computer Society (USA, 2000), 67

Our specifications are portable; they apply not only to locking implementations, but also to optimistic and multi-version concurrency control schemes. Furthermore, unlike earlier definitions, our new specifications handle predicates in a correct and flexible manner at all levels.

It also discusses "Mixing of Isolation Levels".

Bailis, P., Davidson, A., Fekete, A., Ghodsi, A., Hellerstein, J. M., and Stoica, I. Highly available transactions: Virtues and limitations. *Proc. VLDB Endow. 7*, 3 (Nov. 2013), 181–192

In this work, we consider the problem of providing Highly Available Transactions (HATs): transactional guarantees that do not suffer unavailability during system partitions or incur high network latency. We introduce a taxonomy of highly available systems and analyze existing ACID isolation and distributed data consistency guarantees to identify which can and cannot be achieved in HAT systems. This unifies the literature on weak transactional isolation, replica consistency, and highly available systems.

Koskinen, E., and Parkinson, M. The push/pull model of transactions. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '15, Association for Computing Machinery (New York, NY, USA, 2015), 186–195

We present a general theory of serializability, unifying a wide range of transactional algorithms, including some that are yet to come. To this end, we provide a compact semantics in which concurrent transactions PUSH their effects into the shared view (or UNPUSH to recall effects) and PULL the effects of potentially uncommitted concurrent transactions into their local view (or UNPULL to detangle).

Cerone, A., Bernardi, G., and Gotsman, A. A Framework for Transactional Consistency Models with Atomic Visibility. In *26th International Conference on Concurrency Theory (CONCUR 2015)*, L. Aceto and D. de Frutos Escrig, Eds., vol. 42 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (Dagstuhl, Germany, 2015), 58–71 *we propose a framework for specifying a variety of consistency models for transactions uniformly and declaratively. Our specifications are given in the style of weak memory models, using structures of events and relations on them. The specifications are particularly concise because they exploit the property of atomic visibility guaranteed by many consistency models: either all or none of the updates by a transaction can be visible to another one. This allows the specifications to abstract from individual events inside transactions.*

Crooks, N., Pu, Y., Alvisi, L., and Clement, A. Seeing is believing: A client-centric specification of database isolation. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, Association for Computing Machinery (New York, NY, USA, 2017), 73–82

This paper introduces the first state-based formalization of isolation guarantees.

Crooks, N. *A Client-Centric Approach to Transactional Datastores*. PhD thesis, The University of Texas at Austin, 2019

The PhD Thesis version of [25].

2 Serializability

Weikum, G., and Vossen, G. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001

The "TIS" book contains several variants of serializability: Final State Serializability, View Serializability, Conflict Serializability, Commit Serializability, Multiversion Serializability, Global Serializability, Quasi Serializability

Eswaran, K. P., Gray, J. N., Lorie, R. A., and Traiger, I. L. The notions of consistency and predicate locks in a database system. *Commun. ACM 19*, 11 (Nov. 1976), 624–633

This paper defines the concepts of transaction, consistency and schedule and shows that consistency requires that a transaction cannot request new locks after releasing a lock. Then it is argued that a transaction needs to lock a logical rather than a physical subset of the database. These subsets may be specified by predicates. An implementation of predicate locks which satisfies the consistency condition is suggested.

This is the first paper to formalize mathematically the concurrency control problem. It also defines "conflict serializability", which is termed DSR in [55].

Papadimitriou, C. H. The serializability of concurrent database updates. *J. ACM 26*, 4 (Oct. 1979), 631–653

It is shown that recognizing the transaction histories that are serializable is an NP-complete problem. Several efficiently recognizable subclasses are introduced.

Kanellakis, P. C., and Papadimitriou, C. H. Is distributed locking harder? In *Proceedings of the 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, PODS '82, Association for Computing Machinery (New York, NY, USA, 1982), 98–107

We examine the problem of determining whether a set of locked transactions, accessing a distributed database, is guaranteed to produce only serializable schedules. For a pair of transactions we prove that this concurrency control problem (which is polynomially solvable for centralized databases) is in general coNP-complete.

Yannakakis, M. Serializability by locking. *J. ACM 31*, 2 (Mar. 1984), 227–244

It is shown that locking cannot achieve the full power of serializability. An exact characterization of the schedules that can be produced if locking is used to control concurrency is given for two versions of serializability: state serializability and view serializability.

See also its STOC conference version [71].

Attar, R., Bernstein, P. A., and Goodman, N. Site initialization, recovery, and backup in a distributed database system. *IEEE Trans. Softw. Eng. 10*, 6 (Nov. 1984), 645–650

Introduce One Copy Serializability (1SR), as a distributed/replicated counterpart of Serializability in a single-server system.

Kanellakis, P. C., and Papadimitriou, C. H. The complexity of distributed concurrency control. *SIAM J. Comput. 14*, 1 (Feb. 1985), 52–74

We present a formal framework for distributed databases, and we study the complexity of the concurrency control problem in this framework. Our transactions are partially ordered sets of actions, as opposed to the straight-line programs of the centralized case. The concurrency control algorithm, or scheduler, is itself a distributed program.

Hadzilacos, T., and Papadimitriou, C. H. Algorithmic aspects of multiversion concurrency control. In *Proceedings of the Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, PODS '85, Association for Computing Machinery (New York, NY, USA, 1985), 96–104

In this paper we introduce a new notion of multiversion serializability (MVSR) based on conflicts (MVCSR), and discuss its relation with the well known single version conflict serializability (CSR).

Ports, D. R. K., and Grittner, K. Serializable snapshot isolation in postgresql. *Proc. VLDB Endow. 5*, 12 (Aug. 2012), 1850–1861

This paper describes our experience implementing PostgreSQL's new serializable isolation level. It is based on the recently-developed Serializable Snapshot Isolation (SSI) technique. This is the first implementation of SSI in a production database release as well as the first in a database that did not previously have a lock-based serializable isolation level.

Raynal, M., Thia-Kime, G., and Ahamad, M. From Serializable to Causal Transactions for Collaborative Applications. Research Report RR-2802, INRIA, 1996

It defines Causal Serializability.

2 Snapshot Isolation

Fekete, A., O'Neil, E., and O'Neil, P. A read-only transaction anomaly under snapshot isolation. *SIGMOD Rec. 33*, 3 (Sept. 2004), 12–14

It has been widely assumed that, under SI, read-only transactions always execute serializably provided the concurrent update transactions are serializable. The reason for this is that all SI reads return values from a single instant of time when all committed transactions have completed their writes and no writes of non-committed transactions are visible. This seems to imply that read-only transactions will not read anomalous results so long as the update transactions with which they execute do not write such results. In the current note, however, we exhibit an example contradicting these assumptions: it is possible for an SI history to be non-serializable while the sub-history containing all update transactions is serializable.

Elnikety, S., Zwaenepoel, W., and Pedone, F. Database replication using generalized snapshot isolation. In *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems*, SRDS '05, IEEE Computer Society (USA, 2005), 73–84

It defines GSI (Generalized Snapshot Isolation) and PSI (Perfix-Consistent Snapshot Isolation).

While (conventional) snapshot isolation requires that transactions observe the "latest" *snapshot of the database, generalized snapshot isolation allows the use of "older" snapshots, facilitating a replicated implementation.*

Yabandeh, M., and Gómez Ferro, D. A critique of snapshot isolation. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, Association for Computing Machinery (New York, NY, USA, 2012), 155–168

We introduce write-snapshot isolation, a novel isolation level that has a performance comparable with that of snapshot isolation, and yet provides serializability. The main insight in write-snapshot isolation is to prevent read-write conflicts in contrast to write-write conflicts that are prevented by snapshot isolation.

Du, J., Elnikety, S., and Zwaenepoel, W. 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, IEEE Computer Society (USA, 2013), 173–184

Clock-SI is a fully distributed protocol that implements snapshot isolation (SI) for partitioned data stores. It derives snapshot and commit timestamps from loosely synchronized clocks, rather than from a centralized timestamp authority as used in current systems.

Binnig, C., Hildenbrand, S., Färber, F., Kossmann, D., Lee, J., and May, N. Distributed snapshot isolation: Global transactions pay globally, local transactions pay locally. *The VLDB Journal 23*, 6 (Dec. 2014), 987–1011

This paper revisits the problem of implementing Snapshot Isolation in a distributed database system and makes three important contributions. First, a complete definition of Distributed Snapshot Isolation is given, thereby extending existing definitions from the literature. Based on this definition, a set of criteria is proposed to efficiently implement Snapshot Isolation in a distributed system. Second, the design space of alternative methods to implement Distributed Snapshot Isolation is presented based on this set of criteria. Third, a new approach to implement Distributed Snapshot Isolation is devised; we refer *to this approach as Incremental.*

Cerone, A., and Gotsman, A. Analysing snapshot isolation. *J. ACM 65*, 2 (Jan. 2018)

We give an alternative specification to SI that characterises it in terms of transactional dependency graphs of Adya et al., generalising serialisation graphs.

We then exploit our specification to obtain two kinds of static analyses. The first one checks when a set of transactions running under SI can be chopped into smaller pieces without introducing new behaviours, to improve performance. The other analysis checks whether a set of transactions running under a weakening of SI behaves the same as when running under SI.

2 Mixed Transactional Consistency Models

Fekete, A. Allocating isolation levels to transactions. In *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, Association for Computing Machinery (New York, NY, USA, 2005), 206–215

In this paper, we discuss the problem of taking a collection of transactions, and allocating each to run at an appropriate isolation level (and thus use a particular concurrency control mechanism), while still ensuring that every execution will be conflict serializable. When each transaction can use either S2PL, or snapshot isolation, we characterize exactly the acceptable allocations, and provide a simple graph-based algorithm which determines the weakest acceptable allocation.

Milano, M., and Myers, A. C. Mixt: A language for mixing consistency in geodistributed transactions. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2018, Association for Computing Machinery (New York, NY, USA, 2018), 226–241

To manipulate both weakly and strongly consistent data in a single transaction, we introduce a new abstraction: mixed-consistency transactions, embodied in a new embedded language, MixT. Programmers explicitly associate consistency models with remote storage sites; each atomic, isolated transaction can access a mixture of data with different consistency models.

de Régil Basáñez, B. A. A fast implementation of parallel snapshot isolation. Master's thesis, Universidad Complutense de Madrid, 2020

This work shows an approach to bridge the gap between PSI, NMSI and strong consistency models like serialisability. It introduces and implements fastPSI, a consistency protocol that allows the user to selectively enforce serialisability for certain executions, while retaining the scalability properties of weaker consistency models like PSI and NMSI.

3 Robustness (and Dependency Graphs)

Fekete, A., Liarokapis, D., O'Neil, E., O'Neil, P., and Shasha, D. Making snapshot isolation serializable. *ACM Trans. Database Syst. 30*, 2 (June 2005), 492–528

This article develops a theory that characterizes when nonserializable executions of applications can occur under SI.

Jorwekar, S., Fekete, A., Ramamritham, K., and Sudarshan, S. Automating the detection of snapshot isolation anomalies. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, VLDB Endowment (2007), 1263–1274

In this paper we address several issues in extending the earlier theory to practical detection/correction of anomalies. We first show how to mechanically find a set of programs which is large enough so that we ensure that all executions will be free of SI anomalies, by modifying these programs appropriately. We then address the problem of false positives, i.e., transaction programs wrongly identified as possibly leading to anomalies, and present techniques that can significantly reduce such false positives. Unlike earlier *work, our techniques are designed to be automated, rather than manually carried out. We describe a tool which we are developing to carry out this task.*

Cahill, M. J., Röhm, U., and Fekete, A. D. Serializable isolation for snapshot databases. *ACM Trans. Database Syst. 34*, 4 (Dec. 2009)

This article describes a modification to the concurrency control algorithm of a database management system that automatically detects and prevents snapshot isolation anomalies at runtime for arbitrary applications, thus providing serializable isolation.

While it is fairly well known how to identify qualitatively the anomalies that are possible under a certain isolation level, it is much more difficult to detect and quantify such anomalies during run-time of a given application. In this paper, we present a new approach to detect and quantify consistency anomalies for arbitrary multi-tier application running under any isolation levels ensuring at least read committed.

Shang, Z., Yu, J. X., and Elmore, A. J. Rushmon: Real-time isolation anomalies monitoring. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, Association for Computing Machinery (New York, NY, USA, 2018), 647–662

To resolve these problems, these systems need an indicator to report the number of bad events caused by out-of-order executions. In this paper, we tackle this problem. Based on transaction processing theory, we find the number of cycles in the dependency graph, and demonstrate it is a good indicator. With this observation, we propose the first real-time isolation anomalies monitor.

Beillahi, S. M., Bouajjani, A., and Enea, C. Robustness Against Transactional Causal Consistency. In *30th International Conference on Concurrency Theory (CONCUR 2019)*, W. Fokkink and R. van Glabbeek, Eds., vol. 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik

(Dagstuhl, Germany, 2019), 30:1–30:18

In this paper, we investigate application-specific relationships between several variations of causal consistency and we address the issue of verifying automatically if a given transactional program is robust against causal consistency, i.e., all its behaviors when executed over an arbitrary causally consistent database are serializable.

4 Concurrency Control Protocols

4 Theory

Kung, H. T., and Papadimitriou, C. H. An optimality theory of concurrency control for databases. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, SIGMOD '79, Association for Computing Machinery (New York, NY, USA, 1979), 116–126

There is a growing body of literature on various solutions to the concurrency control problem. This paper gives a uniform framework for evaluating these solutions, and, in many cases, for establishing their optimality. We point out a trade-off between the performance of a scheduler and the information that it uses. We show that most of the existing work on concurrency control is concerned with specific points of this fundamental trade-off. For example, our framework allows us to formally show that the popular approach of Serialization is the best one can hope for when only syntactic information is available.

Boksenbaum, C., Cart, M., Ferrié, J., and Pons, J.-F. Certification by intervals of timestamps in distributed database systems. In *Proceedings of the 10th International Conference on Very Large Data Bases*, VLDB '84, Morgan Kaufmann Publishers Inc. (San Francisco, CA, USA, 1984), 377–387

Certification by Intervals of Timestamps in Distributed Database Systems.

4 Overview

Bernstein, P. A., and Goodman, N. Concurrency control in distributed database systems. *ACM Comput. Surv. 13*, 2 (June 1981), 185–221

In this paper we survey, consolidate, and present the state of the art in distributed database concurrency control. The heart of our analysts is a decomposition of the concurrency control problem into two major subproblems: read-write and write-write synchronization. We describe a series of synchronization techniques for solving each subproblem and show how to combine these techniques into algorithms for solving the entire concurrency control problem. Such algorithms are called "concurrency control methods." We describe 48 principal methods, including all practical algorithms that have appeared m the literature plus several new ones. We concentrate on the structure and correctness of concurrency control algorithms. Issues of performance are given only secondary treatment.

4 Locking

Gray, J. N., Lorie, R. A., and Putzolu, G. R. Granularity of locks in a shared data base. In *Proceedings of the 1st International Conference on Very Large Data Bases*, VLDB '75, Association for Computing Machinery (New York, NY, USA, 1975), 428–451

This paper proposes a locking protocol which associates locks with sets of resources. This protocol allows simultaneous locking at various granularities by different transactions. It is based on the introduction of additional lock modes besides the conventional share mode and exclusive mode. The protocol is generalized from simple hierarchies of locks to directed acyclic graphs of locks and to dynamic graphs of locks. The issues of scheduling and granting conflicting requests for the same resource are then discussed. Lastly, these ideas are compared with the lock mechanisms provided by existing data management systems.

It introduces intention locks.

Yannakakis, M. A theory of safe locking policies in database systems. *J. ACM 29*, 3 (July 1982), 718–740

Necessary and sufficient conditions are found for a locking policy to be safe, but it is shown that in general it is NP-complete to test for these conditions. However, when the database has a given structure, a simple set of rules which is sufficient for safety and, moreover, necessary for a wide class of natural locking policies is developed.

Buckley, G. N., and Silberschatz, A. Beyond two-phase locking. *J. ACM 32*, 2 (Apr. 1985), 314–343

Graph protocols.

Thomasian, A. Concurrency control: Methods, performance, and analysis. *ACM Comput. Surv. 30*, 1 (Mar. 1998), 70–119

This tutorial reviews CC methods based on standard locking, restart-oriented locking methods, two-phase processing methods including optimistic CC, and hybrid methods (combining optimistic CC and locking) in centralized systems.

4 Hybrid Concurrency Control

Xie, C., Su, C., Littley, C., Alvisi, L., Kapritsos, M., and Wang, Y. High-performance acid via modular concurrency control. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, Association for Computing Machinery (New York, NY, USA, 2015), 279–294

This paper describes the design, implementation, and evaluation of Callas, a distributed database system that offers to unmodified, transactional ACID applications the opportunity to achieve a level of performance that can currently only be reached by rewriting all or part of the application in a BASE/NoSQL style. The key to combining performance and ease of programming is to decouple the ACID abstraction, which Callas offers identically for all transactions, from the mechanism used to support it. MCC, the new Modular approach to Concurrency Control at the core of Callas, makes it possible to partition transactions in groups with the guarantee that, as long as the concurrency control *mechanism within each group upholds a given isolation property, that property will also hold among transactions in different groups.*

Ziv, O., Aiken, A., Golan-Gueta, G., Ramalingam, G., and Sagiv, M. Composing concurrency control. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '15, Association for Computing Machinery (New York, NY, USA, 2015), 240–249

Concurrency control poses significant challenges when composing computations over multiple data-structures (objects) with different concurrency-control implementations. We formalize the usually desired requirements (serializability, abort-safety, deadlock-safety, and opacity) as well as stronger versions of these properties that enable composition. We show how to compose protocols satisfying these properties so that the resulting combined protocol also satisfies these properties. Our approach generalizes well-known protocols (such as two-phase-locking and two-phase-commit) and leads to new protocols. We apply this theory to show how we can safely compose optimistic and pessimistic concurrency control. For example, we show how we can execute a transaction that accesses two objects, one controlled by an STM and another by locking.

Su, C., Crooks, N., Ding, C., Alvisi, L., and Xie, C. Bringing modular concurrency control to the next level. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, Association for Computing Machinery (New York, NY, USA, 2017), 283–297

This paper presents Tebaldi, a distributed key-value store that explores new ways to harness the performance opportunity of combining different specialized concurrency control mechanisms (CCs) within the same database. Tebaldi partitions conflicts at a fine granularity and matches them to specialized CCs within a hierarchical framework that is modular, extensible, and able to support a wide variety of concurrency control techniques, from single-version to multiversion and from lock-based to timestamp-based.

4 Atomic Commitment and Consensus

Hadzilacos, V. A knowledge-theoretic analysis of atomic commitment protocols. In *Proceedings of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '87, Association for Computing Machinery (New York, NY, USA, 1987), 129–134

A knowledge-theoretic analysis of atomic commitment protocols.

Gray, J., and Lamport, L. Consensus on transaction commit. *ACM Trans. Database Syst. 31*, 1 (Mar. 2006), 133–160

The Paxos Commit algorithm runs a Paxos consensus algorithm on the commit/abort decision of each participant to obtain a transaction commit protocol that uses $2F + 1$ coordinators and makes progress if at least $F + 1$ of them are working properly.

Rao, J., Shekita, E. J., and Tata, S. Using paxos to build a scalable, consistent, and highly available datastore. *Proc. VLDB Endow. 4*, 4 (Jan. 2011), 243–254

Spinnaker is an experimental datastore that is designed to run on a large cluster of commodity servers in a single data center. It features key-based range partitioning, 3-way replication, and a transactional get-put API with the option to choose either strong or timeline consistency on reads. This paper describes Spinnaker's Paxos-based replication protocol.

Baker, J., Bond, C., Corbett, J. C., Furman, J., Khorlin, A., Larson, J., Leon, J.-M., Li, Y., Lloyd, A., and Yushprakh, V. Megastore: Providing scalable, highly available storage for interactive services. In *Proceedings of the 5th Biennial Conference on Innovative Data Systems Research (CIDR' 11)* (January 2011), 223–234

Megastore is a storage system developed to meet the requirements of today's interactive online services. Megastore blends the scalability of a NoSQL datastore with the convenience of a traditional RDBMS in a novel way, and provides both strong consistency guarantees and high availability. We provide fully serializable ACID semantics within fine-grained partitions of data. This partitioning allows us to synchronously replicate each write across a wide area network with reasonable latency and support seamless failover between dat-

acenters. This paper describes Megastore's semantics and replication algorithm.

While many systems use Paxos solely for locking, master election, or replication of metadata and configurations, we believe that Megastore is the largest system deployed that uses Paxos to replicate primary user data across datacenters on every write.

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

We first develop and analyze a transaction management and replication protocol based on a straightforward implementation of the Paxos algorithm. Our investigation reveals that this protocol acts as a concurrency prevention mechanism rather than a concurrency control mechanism. We then propose an enhanced protocol called Paxos with Combination and Promotion (Paxos-CP) that provides true transaction concurrency while requiring the same per instance message complexity as the basic Paxos protocol. Finally, we compare the performance of Paxos and Paxos-CP in a multi-datacenter experimental study, and we demonstrate that Paxos-CP results in significantly fewer aborted transactions than basic Paxos.

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

Spanner uses Two-Phase Commit and Two-Phase Locking to provide atomicity and isolation for globally distributed data, running on top of Paxos to provide fault-tolerant log replication. We show in this paper that it is possible to provide the same ACID transactional guarantees for multi-datacenter databases with fewer cross-datacenter communication trips, compared to replicated logging. Instead of replicating the transactional log, we replicate the commit operation itself, by running Two-Phase Commit multiple times in different datacenters and using Paxos to reach consensus among datacenters as to whether the transaction should commit. Doing so not only replaces several inter-datacenter communication trips with intra-datacenter communication trips, but also allows us to integrate atomic commitment and isolation protocols with consistent replication protocols to further reduce the number of cross-datacenter communication trips needed for consistent *replication; for example, by eliminating the need for an election phase in Paxos.*

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

MDCC (Multi-Data Center Consistency) is an optimistic commit protocol for geo-replicated transactions, that does not require a master or static partitioning, and is strongly consistent at a cost similar to eventually consistent protocols. MDCC takes advantage of Generalized Paxos for transaction processing and exploits commutative updates with value constraints in a quorum-based system. Our experiments show that MDCC outperforms existing synchronous transactional replication protocols, such as Megastore, by requiring only a single message round-trip in the normal operational case independent of the master-location and by scaling linearly with the number of machines as long as transaction conflict rates permit.

Mu, S., Nelson, L., Lloyd, W., and Li, J. Consolidating concurrency control and consensus for commits under conflicts. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, USENIX Association (USA, 2016), 517–532

In this paper, we make the key observation that the coordination required for concurrency control and consensus is highly similar. Specifically, each tries to ensure the serialization graph of transactions is acyclic. We exploit this insight in the design of Janus, a unified concurrency control and consensus protocol.

Like MDCC and TAPIR, Janus can commit unconflicted transactions in this class in one round-trip. Unlike MDCC and TAPIR, Janus avoids aborts due to contention: it commits conflicted transactions in this class in at most two round-trips as long as the network is well behaved and a majority of each server replica is alive.

Chockler, G., and Gotsman, A. Multi-Shot Distributed Transaction Commit. In *32nd International Symposium on Distributed Computing (DISC 2018)*, U. Schmid and J. Widder, Eds., vol. 121 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (Dagstuhl, Germany, 2018), 14:1–14:18

We introduce Transaction Certification Service (TCS), a new formal problem that captures safety guarantees of multi-shot transaction commit protocols with integrated concurrency control. TCS is parameterized by a certification function that can be instantiated to support common isolation levels, such as serializability and snapshot isolation. We then derive a provably correct crash-resilient protocol for implementing TCS through successive refinement.

Zhang, I., Sharma, N. K., Szekeres, A., Krishnamurthy, A., and Ports, D. R. K. Building consistent transactions with inconsistent replication. *ACM Trans. Comput. Syst. 35*, 4 (Dec. 2018)

We present the Transactional Application Protocol for Inconsistent Replication (TAPIR), the first transaction protocol to use a novel replication protocol, called inconsistent replication, that provides fault tolerance without consistency. By enforcing strong consistency only in the transaction protocol, TAPIR can commit transactions in a single round-trip and order distributed transactions without centralized coordination.

This article demonstrates that it is possible to build distributed transactions with better performance and strong consistency semantics by building on a replication protocol with no consistency. We present inconsistent replication, a new replication protocol that provides fault tolerance without consistency, and TAPIR, a new distributed transaction protocol that provides linearizable transactions using IR.

(Fast path vs. Slow path)

Fan, H., and Golab, W. Ocean vista: Gossip-based visibility control for speedy geo-distributed transactions. *Proc. VLDB Endow. 12*, 11 (July 2019), 1471–1484

We introduce Ocean Vista – a novel distributed protocol that guarantees strict serializability. We observe that concurrency control and replication address different aspects of resolving the visibility of transactions, and we address both concerns using a multi-version protocol that tracks visibility using version watermarks and arrives at correct visibility decisions using efficient gossip. Gossiping the watermarks enables asynchronous transaction processing and acknowledging transaction visibility in batches in the concurrency control and replication protocols, which improves efficiency under high cross-datacenter network delays.

Maiyya, S., Nawab, F., Agrawal, D., and Abbadi, A. E. Unifying consensus and atomic commitment for effective cloud data management. *Proc. VLDB Endow. 12*, 5 (Jan. 2019), 611–623

Data storage in the Cloud needs to be scalable and fault-tolerant. Atomic commitment protocols such as Two Phase Commit (2PC) provide ACID guarantees for transactional access to sharded data and help in achieving scalability. Whereas consensus protocols such as Paxos consistently replicate data across different servers and provide fault tolerance. Cloud based datacenters today typically treat the problems of scalability and fault-tolerance disjointedly. In this work, we propose a unification of these two different paradigms into one framework called Consensus and Commitment (C&C) framework. The C&C framework can model existing and well known data management protocols as well as propose new ones.

4 Implementations

Jung, H., Han, H., Fekete, A., and Röhm, U. Serializable snapshot isolation for replicated databases in high-update scenarios. *Proc. VLDB Endow. 4*, 11 (Aug. 2011), 783–794

We propose a new algorithm Replicated Serializable Snapshot Isolation (RSSI) that uses SI at each site, and combines this with a certification algorithm to guarantee 1-copy serializable global execution. Management of ww-conflicts is similar to what is done in 1-copy SI. But unlike previous designs for 1-copy serializable systems, we do not need to prevent all rw-conflicts among concurrent transactions. We formalize this in a theorem that *shows that many rw-conflicts are indeed false positives that do not risk non-serializable behavior. Our proposed RSSI algorithm will only abort a transaction when it detects a well-defined pattern of two consecutive rw-edges in the serialization graph.*

Mahmoud, H. A., Arora, V., Nawab, F., Agrawal, D., and El Abbadi, A. Maat: Effective and scalable coordination of distributed transactions in the cloud. *Proc. VLDB Endow. 7*, 5 (Jan. 2014), 329–340

Very little theoretical work has been done to entirely eliminate the need for locking in distributed transactions, including locks acquired during two-phase commit. In this paper, we re-design optimistic concurrency control to eliminate any need for locking even for atomic commitment, while handling the practical issues in earlier theoretical work related to this problem.

Chairunnanda, P., Daudjee, K., and Özsu, M. T. Confluxdb: Multi-master replication for partitioned snapshot isolation databases. *Proc. VLDB Endow. 7*, 11 (July 2014), 947–958

We propose a set of techniques that support update transaction execution over multiple partitioned sites, thereby allowing the master to scale. Our techniques determine a total SI order for update transactions over multiple master sites without requiring global coordination in the distributed system, and ensure that updates are installed in this order at all sites to provide consistent and scalable replication with SI.

Didona, D., Guerraoui, R., Wang, J., and Zwaenepoel, W. Causal consistency and latency optimality: Friend or foe? *Proc. VLDB Endow. 11*, 11 (July 2018), 1618–1632

In this paper, we show that such "latency-optimal" ROTs induce an extra overhead on writes that is so high that it actually jeopardizes performance even in read-dominated workloads. We show this result from a practical as well as from a theoretical angle.

Barthels, C., Müller, I., Taranov, K., Alonso, G., and Hoefler, T. Strong consistency is not hard to get: Two-phase locking and two-phase commit on thousands of cores. *Proc. VLDB Endow. 12*, 13 (Sept. 2019), 2325–2338

The goal of this paper is to establish a baseline for concurrency control mechanisms on thousands of cores connected through a low-latency network. We develop a distributed lock table supporting all the standard locking modes used in database engines. We focus on strong consistency in the form of strict serializability implemented through strict 2PL, but also explore read-committed and repeatable-read, two common isolation levels used in many systems.

The surprising result is that, for TPC-C, 2PL and 2PC can be made to scale to thousands of cores and hundreds of machines, reaching a throughput of over 21 million transactions per second with 9.5 million New Order operations per second.

To achieve these results, our implementation relies on Remote Direct Memory Access (RDMA).

Ren, K., Li, D., and Abadi, D. J. Slog: Serializable, low-latency, geo-replicated transactions. *Proc. VLDB Endow. 12*, 11 (July 2019), 1747–1761

In this paper we discuss SLOG: a system that avoids this tradeoff for workloads which contain physical region locality in data access. SLOG achieves high-throughput, strictly serializable ACID transactions at geo-replicated distance and scale for all transactions submitted across the world, all the while achieving low latency for transactions that initiate from a location close to the home region for data they access.

Kobus, T., Kokocinski, M., and Wojciechowski, P. T. Creek: a general mixed-consistency transactional replication scheme. *CoRR abs/1907.00748* (2019)

In this paper we introduce Creek, a low-latency, eventually consistent replication scheme that also enables execution of strongly consistent operations (akin to ACID transactions).

Creek totally-orders all operations, but does so using two different broadcast mechanisms: a timestamp-based one and our novel conditional atomic broadcast (CAB). The former is used to establish a tentative order of all operations for speculative execution, and it can *tolerate network partitions. On the other hand, CAB is only used to ensure linearizable execution of the strongly consistent operations, whenever distributed consensus can be solved. The execution of strongly consistent operations also stabilizes the execution order of the causally related weakly consistent operations. Creek uses multiversion concurrency control to efficiently handle operations' rollbacks and reexecutions resulting from the mismatch between the tentative and the final execution orders.*

Wu, C., Sreekanti, V., and Hellerstein, J. M. Transactional causal consistency for serverless computing. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, Association for Computing Machinery (New York, NY, USA, 2020), 83–97

This raises the challenge of Multisite Transactional Causal Consistency (MTCC): the ability to provide causal consistency for all I/Os within a given transaction even if it runs across multiple physical sites. We present protocols for MTCC implemented in a system called HYDROCACHE.

5 Theory: Complexity and Limitations

Didona, D., Fatourou, P., Guerraoui, R., Wang, J., and Zwaenepoel, W. Distributed transactional systems cannot be fast. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '19, Association for Computing Machinery (New York, NY, USA, 2019), 369–380

We prove that no fully transactional system can provide fast read transactions (including read-only ones that are considered the most frequent in practice). Specifically, to achieve fast read transactions, the system has to give up support of transactions that write more than one object. We prove this impossibility result for distributed storage systems that are causally consistent, i.e., they do not require to ensure any strong form of consistency. Therefore, our result holds also for any system that ensures a consistency level stronger than causal consistency, e.g., strict serializability. The impossibility result holds even for systems that store only two objects (and support at least two servers and at least four clients). It also holds for systems that are partially replicated.

6 Formal Methods

Pollak, D. H. Reasoning about two-phase locking concurrency control. Technical report, June 2017

We present a program logic for serializable transactions that are able to manipulate a shared storage.

We show this by providing the first application of our logic in terms of the Two-phase locking (2pl) protocol which ensures serializability.

Kaki, G., Nagar, K., Najafzadeh, M., and Jagannathan, S. Alone together: Compositional reasoning and inference for weak isolation. *Proc. ACM Program. Lang. 2*, POPL (Dec. 2017)

Unfortunately, the semantics of weak isolation is poorly understood, and usually explained only informally in terms of low-level implementation artifacts. Consequently, verifying high-level correctness properties in such environments remains a challenging problem. To address this issue, we present a novel program logic that enables compositional reasoning about the behavior of concurrently executing weakly-isolated transactions.

Milano, M., and Myers, A. C. Mixt: A language for mixing consistency in geodistributed transactions. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2018, Association for Computing Machinery (New York, NY, USA, 2018), 226–241

To manipulate both weakly and strongly consistent data in a single transaction, we introduce a new abstraction: mixed-consistency transactions, embodied in a new embedded language, MixT.

Whittaker, M., and Hellerstein, J. M. Interactive checks for coordination avoidance. *Proc. VLDB Endow. 12*, 1 (Sept. 2018), 14–27

In this paper, we establish conditions under which a commonly used sufficient condition for invariant confluence is both necessary and sufficient, and we use this condition to design (a) a general-purpose interactive invariant confluence decision procedure and (b) a novel sufficient condition that can be checked automatically. We then take a step beyond invariant confluence and introduce a generalization of invariant confluence, called segmented invariant confluence, that allows us to replicate non-invariant confluent objects with a small amount of coordination.

Balegas, V., Duarte, S., Ferreira, C., Rodrigues, R., and Preguiça, N. Ipa: Invariant-preserving applications for weakly consistent replicated databases. *Proc. VLDB Endow. 12*, 4 (Dec. 2018), 404–418

In this paper we propose a novel approach to preserve application invariants without coordinating the execution of operations. The approach consists of modifying operations in a way that application invariants are maintained in the presence of concurrent updates. When no conflicting updates occur, the modified operations present their original semantics. Otherwise, we use sensible and deterministic conflict resolution policies that preserve the invariants of the application. To implement this approach, we developed a static analysis, IPA, that identifies conflicting operations and proposes the necessary modifications to operations.

7 Systems

Calder, B., Wang, J., Ogus, A., Nilakantan, N., Skjolsvold, A., McKelvie, S., Xu, Y., Srivastav, S., Wu, J., Simitci, H., Haridas, J., Uddaraju, C., Khatri, H., Edwards, A., Bedekar, V., Mainali, S., Abbasi, R., Agarwal, A., Haq, M. F. u., Haq, M. I. u., Bhardwaj, D., Dayanand, S., Adusumilli, A., McNett, M., Sankaran, S., Manivannan, K., and Rigas, L. Windows azure storage: A highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operat-*

ing Systems Principles, SOSP '11, Association for Computing Machinery (New York, NY, USA, 2011), 143–157

Windows Azure Storage (WAS) is a cloud storage system that provides customers the ability to store seemingly limitless amounts of data for any duration of time. WAS customers have access to their data from anywhere at any time and only pay for what they use and store. In WAS, data is stored durably using both local and geographic replication to facilitate disaster recovery. Currently, WAS storage comes in the form of Blobs (files), Tables (structured storage), and Queues (message delivery). In this paper, we describe the WAS architecture, global namespace, and data model, as well as its resource provisioning, load balancing, and replication systems.

Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., Hsieh, W., Kanthak, S., Kogan, E., Li, H., Lloyd, A., Melnik, S., Mwaura, D., Nagle, D., Quinlan, S., Rao, R., Rolig, L., Saito, Y., Szymaniak, M., Taylor, C., Wang, R., and Woodford, D. Spanner: Google's globally distributed database. *ACM Trans. Comput. Syst. 31*, 3 (Aug. 2013)

Spanner is Google's scalable, multiversion, globally distributed, and synchronously replicated database. It is the first system to distribute data at global scale and support externally-consistent distributed transactions. This article describes how Spanner is structured, its feature set, the rationale underlying various design decisions, and a novel time API that exposes clock uncertainty. This API and its implementation are critical to supporting external consistency and a variety of powerful features: nonblocking reads in the past, lock-free snapshot transactions, and atomic schema changes, across all of Spanner.

7 New Hardwares

Zamanian, E., Yu, X., Stonebraker, M., and Kraska, T. Rethinking database high availability with rdma networks. *Proc. VLDB Endow. 12*, 11 (July 2019), 1637–1650

In this paper, we first make the case that in modern RDMA-enabled networks, the bottleneck has shifted to CPUs, and therefore the existing network-optimized replication techniques are no longer optimal.

We present Active-Memory Replication, a new high availability scheme that efficiently leverages RDMA to completely eliminate the processing redundancy in replication. Using Active-Memory, all replicas dedicate their processing power to executing new transactions, as opposed to performing redundant computation. Active-Memory maintains high availability and correctness in the presence of failures through an efficient RDMA-based undo-logging scheme.

8 Testing

Shang, Z., Yu, J. X., and Elmore, A. J. Rushmon: Real-time isolation anomalies monitoring. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, Association for Computing Machinery (New York, NY, USA, 2018), 647–662

. . . these systems need an indicator to report the number of "bad event" caused by "out-of-order" executions. In this paper, we tackle this problem. Based on transaction processing theory, we find the number of cycles in the dependency graph, and demonstrate it is a good indicator. With this observation, we propose the first real-time isolation anomalies monitor.

Rahmani, K., Nagar, K., Delaware, B., and Jagannathan, S. Clotho: Directed test generation for weakly consistent database systems. *Proc. ACM Program. Lang. 3*, OOPSLA (Oct. 2019)

This paper presents a novel testing framework for detecting serializability violations in (SQL) database-backed Java applications executing on weakly-consistent storage systems. We manifest our approach in a tool, CLOTHO, that combines a static analyzer and model checker to generate abstract executions, discover serializability violations in these executions, and translate them back into concrete test inputs suitable for deployment in a test *environment.*

Biswas, R., and Enea, C. On the complexity of checking transactional consistency. *Proc. ACM Program. Lang. 3*, OOPSLA (Oct. 2019)

In this work, we investigate the problem of checking whether a given execution of a transactional database adheres to some consistency model. We show that consistency models like read committed, read atomic, and causal consistency are polynomial-time checkable while prefix consistency and snapshot isolation are NP-complete in general.