Recent years have seen the rise of large-scale distributed systems with stringent performance requirements. Applications such as Facebook, Twitter, and Amazon store large amounts of data which must remain accessible with low latency, regardless of the geographical location of users or the number of faulty machines [6]. Data is replicated geographically to minimise latency and multiple copies of the data are placed on separate servers to concurrently process requests. In this environment, developers must choose between consistency through slow lock-step coordination, or performance/availability through independently and concurrently operating replicas. The former is untenable financially because of the latency overhead. The latter leads to confusing application behaviours and a lack of coherent semantics. These two extremes highlight an increasing tension between performance, correctness and complexity [13]. Understanding this tension is what drives my research. I hope to develop the foundations and system support for reasoning about such systems, whose executions are fundamentally concurrent, and cannot be easily reduced to a sequential execution. In that context, my main interests are three-fold: 1) consistency and concurrency-control in databases and distributed systems 2) data-centric and declarative distributed programming, 3) distributed computing. I believe that concurrency and distribution should be treated as first class primitives and I would argue that leveraging the semantics of programs through declarative, state-based, specifications is the best way to reconcile performance and correctness in distributed systems.

Many have complained about the lack of coherent semantics in modern distributed systems and databases [39]. Existing approaches cluster around two extremes: at one end of the spectrum, eventual consistency equates consistency to convergence. At the other, strong synchronisation reduces consistency to serializability [4], but very little exists in between. I would argue that this lack of coherent semantics stems from a fundamental mismatch between the centralised, sequential execution model, captured by a total order of read-write operations, to which developers are accustomed, and the concurrent reality, represented by a partial order of semantically meaningful operations. In that context, the core of my dissertation focuses on a project called TARDiS [10]. This project observes that most approaches to eventual consistency make a fundamental assumption about the underlying storage system: that it tracks the linear evolution of a single site. The project questions that assumption. It instead advocates that the storage should explicitly track the independent executions that arise in loosely synchronised systems, modeling conflicting executions as separate branches. Intuitively, the *TARDiS* database branches on conflict instead of aborting. Consistency is then maintained on a per branch basis with explicit merge points. We found that forking on conflict improved overall performance whilst keeping track of independent executions simplified conflict resolution significantly. In other words, exposing distribution to the programmer improved performance and, paradoxically, reduced complexity.

By working on TARDiS, we realised that the read-write definition of consistency that many use is insufficient [3]. It is impossible for a database to merge conflicting operations in a way that always makes sense to the application: defining conflict as strictly linked to competing writes, restricts consistency guarantees to operation ordering and convergence. Instead, I would argue that consistency is strictly application-specific: users expect the system to guarantee a small number of application-specific properties. In line with existing systems that leverage application-specific properties to improve performance [17, 20, 34, 31], I believe that a two-level approach to designing and programming distributed systems is necessary, where the interface (policy) is separate from the execution (mechanism). *TARDiS*, for example, separated the mechanism of tracking and executing operations from the policy of merging conflicting executions, providing more flexible semantics to the application. But the reverse approach can also be taken: tuning the implementation to best match a workload whilst keeping the same semantics. As a postgraduate student at Cambridge in 2012, I co-started Musketeer [14], a workflow manager for large-scale distributed processing which leverages this insight. Musketeer takes jobs written in existing workflow specification languages (Hive [38], Lindi [24], GraphLinq [24], etc), maps them to a common intermediate abstraction based on a direct acyclic graph (DAG) of data-flow operators (loosely based on the relational algebra), which it then compiles dynamically to the best data-processing system(s). By separating policy from mechanism, Musketeer makes it easy for programmers to write jobs in a single high-level language, but leverages the performance benefits of highly specialised implementations. More recently, my colleages and I at UT Austin begun the process of reformulating consistency and isolation in a state-based, client-centric fashion, which is better aligned with how users perceive database isolation and consistency [9]. Informed by TARDiS and Muskeeter, our approach is premised on a simple observation: applications view storage systems as black-boxes that transition through a series of states, a subset of which are observed by applications. Separating the interface from the low level mechanisms that implements it allows us to define isolation guarantees in terms of high-level states that frees definitions from implementation. Doing so helps make immediately clears

what anomalies applications can expect to observe, thus bridging the gap between how isolation guarantees are defined and how they are perceived. Using this formalization, we find that several well-known guarantees, previously thought to be distinct, are in fact equivalent, and that many previously incomparable flavors of snapshot isolation can be organized in a clean hierarchy. Moreover, we observe that adopting a state-based view of isolation opens up new opportunities for implementations of popular isolation levels, making them resilient to slowdown cascade, a common phenomenon in datacenters that has inhibited the adoption of stronger isolation and consistency levels at scale. As I finish my PhD, I hope to continue developing systems in which consistency is better aligned with modern requirements: consistency should be client-centric, state-based, branching and adaptive. Our most recent work (under submission), for instance, focuses on extending our state-based formulation in our PODC work to expressing consistency and isolation as a series of concurrent branches, strengthening the theoretical foundations on which the TARDiS project stands.

More recently, I have become interested in a separate aspect of consistency at scale: security. Cloud storage services are stocking increasingly sensitive data that can reveal personal information about individual users, ranging from their political leanings to their sexual orientation [26, 33, 25]. Though the majority of cloud storage systems store the data encrypted, evidence suggests that it is insufficient to successfully hide all private information, as access patterns alone often reveal sensitive information. For instance, the frequency at which certain objects are updated can reveal a patient's cancer type [21, 22]. The desire to hide access patterns from cloud storage systems is not new: it has long been studied in the context of private information retrieval [16, 1, 7, 18, 27, 12, 40, 15], oblivious ram [8, 36, 37, 30, 5, 21, 11, 32, 35] and more recently, in systems like CryptDB [29]. These systems, however, suffer from two main drawbacks, which makes them difficult to implement in modern cloud storage systems: they provide no or little [5] support for concurrent writes, and do not support transactions. Concurrent writes are essential for scalability in scale-out systems like S3, while increasingly many storage systems are adding transactional support to facilitate application development [28, 2, 19, 23]. I recently started a project at Cornell that proposes a new design point in the realm of private cloud-based transactional storage systems and asks a question: is it possible to build a transactional cloud-based storage system that both provides transactions and provably hides both the content of the data, and how it is accessed? Though the work is just beginning, it is a promising avenue of research, and I look forward to continuing this line of work, especially extending our initial ideas to set-ups with trusted hardware and hardware-based cryptography primitives.

References

[1] Carlos Aguilar-Melchor et al. *XPIR: Private Information Retrieval for Everyone*. Cryptology ePrint Archive, Report 2014/1025. 2014.

[2] Amazon. *Simple DB*. `https://aws.amazon.com/simpledb/`.

[3] Peter Bailis et al. "The potential dangers of causal consistency and an explicit solution". In: *Proceedings of the Third ACM Symposium on Cloud Computing*. SoCC '12. San Jose, California: ACM, 2012, 22:1–22:7. ISBN: 978-1-4503-1761-0. DOI: 10.1145/2391229.2391251. URL: http://doi.acm.org/10.1145/2391229.2391251.

[4] Philip A Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency control and recovery in database systems*. 1987.

[5] Elette Boyle, Kai-Min Chung, and Rafael Pass. "Oblivious Parallel RAM and Applications". In: *Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*. Ed. by Eyal Kushilevitz and Tal Malkin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 175–204. ISBN: 978-3-662-49099-0. DOI: 10.1007/978-3-662-49099-0_7. URL: http://dx.doi.org/10.1007/978-3-662-49099-0_7.

[6] J. Brutlag. *Speed Matters for Google Web Search*. 22 June 2009. URL: http://http://code.google.com/speed/files/delayexp.pdf.

[7] Benny Chor et al. "Private Information Retrieval". In: *J. ACM* 45.6 (Nov. 1998), pp. 965–981. ISSN: 0004-5411. DOI: 10.1145/293347.293350. URL: http://doi.acm.org/10.1145/293347.293350.

[8] Kai-Min Chung and Rafael Pass. *A simple oram*. Tech. rep. DTIC Document, 2013.

[9] Natacha Crooks et al. "Seeing is Believing: A Client-Centric Specification of Database Isolation". In: *PODC*. 2017.

[10] Natacha Crooks et al. "TARDiS: A Branch-and-Merge Approach To Weak Consistency". In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. San Francisco, California, USA: ACM, 2016, pp. 1615–1628. ISBN: 978-1-4503-3531-7. DOI: 10.1145/2882903.2882951. URL: http://doi.acm.org/10.1145/2882903.2882951.

[11] Jonathan Dautrich, Emil Stefanov, and Elaine Shi. "Burst ORAM: Minimizing ORAM Response Times for Bursty Access Patterns". In: *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, 2014, pp. 749–764. ISBN: 978-1-931971-15-7. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/dautrich.

[12] William Gasarch. "A survey on private information retrieval". In: *Bulletin of the European Association for Theoretical Computer Science (EATCS)* 82 (2004), pp. 72–107.

[13] Seth Gilbert and Nancy Lynch. "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services". In: *SIGACT News* 33.2 (June 2002), pp. 51–59. ISSN: 0163-5700. DOI: 10.1145/564585.564601. URL: http://doi.acm.org/10.1145/564585.564601.

[14] Ionel Gog et al. "Musketeer: All for One, One for All in Data Processing Systems". In: *Proceedings of the Tenth European Conference on Computer Systems*. EuroSys '15. Bordeaux, France: ACM, 2015, 2:1–2:16. ISBN: 978-1-4503-3238-5. DOI: 10.1145/2741948.2741968. URL: http://doi.acm.org/10.1145/2741948.2741968.

[15] I Goldberg. *Percy++ project on SourceForge*. http://percy.sourceforge.net/.

[16] Trinabh Gupta et al. "Scalable and Private Media Consumption with Popcorn". In: *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*. NSDI'16. Santa Clara, CA: USENIX Association, 2016, pp. 91–107. ISBN: 978-1-931971-29-4. URL: http://dl.acm.org/citation.cfm?id=2930611.2930618.

[17] Tim Kraska et al. "MDCC: multi-data center consistency". In: *Proceedings of the 8th ACM European Conference on Computer Systems*. EuroSys '13. Prague, Czech Republic: ACM, 2013, pp. 113–126. ISBN: 978-1-4503-1994-2. DOI: 10.1145/2465351.2465363. URL: http://doi.acm.org/10.1145/2465351.2465363.

[18] Eyal Kushilevitz and Rafail Ostrovsky. "Replication is not needed: Single database, computationally-private information retrieval". In: 1997.

[19] Avinash Lakshman and Prashant Malik. "Cassandra: structured storage system on a P2P network". In: *Proceedings of the 28th ACM symposium on Principles of distributed computing*. PODC '09. Calgary, AB, Canada: ACM, 2009, pp. 5–5. ISBN: 978-1-60558-396-9. DOI: 10.1145/1582716.1582722. URL: http://doi.acm.org/10.1145/1582716.1582722.

[20] Cheng Li et al. "Making geo-replicated systems fast as possible, consistent when necessary". In: *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation*. OSDI'12. Hollywood, CA, USA: USENIX Association, 2012, pp. 265–278. ISBN: 978-1-931971-96-6. URL: http://dl.acm.org/citation.cfm?id=2387880.2387906.

[21] Travis Mayberry, Erik oliver Blass, and Agnes Hui Chan. *Efficient Private File Retrieval by Combining ORAM and PIR*.

[22] Travis Mayberry, Erik-Oliver Blass, and Agnes Hui Chan. "PIRMAP: Efficient private information retrieval for MapReduce". In: 2013.

[23] Microsoft. *Azure Tables*. https://azure.microsoft.com/en-us/services/storage/tables/.

[24] Derek G. Murray et al. "Naiad: A Timely Dataflow System". In: *Proceedings of SOSP*. Farminton, Pennsylvania, 2013, pp. 439–455. ISBN: 978-1-4503-2388-8. DOI: 10.1145/2517349.2522738. URL: http://doi.acm.org/10.1145/2517349.2522738.

[25] Arvind Narayanan and Vitaly Shmatikov. "Myths and Fallacies of "Personally Identifiable Information"". In: 53.6 (June 2010), pp. 24–26.

[26] Arvind Narayanan and Vitaly Shmatikov. "Robust De-anonymization of Large Sparse Datasets". In: 2008.

[27] Rafail Ostrovsky and William E Skeith III. "A survey of single-database private information retrieval: Techniques and applications". In: 2007.

[28] Google Computing Platform. *Cloud Spanner*. http://cloud.google.com/spanner/.

[29] Raluca Ada Popa et al. "CryptDB: Protecting Confidentiality with Encrypted Query Processing". In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. SOSP '11. Cascais, Portugal: ACM, 2011, pp. 85–100. ISBN: 978-1-4503-0977-6. DOI: 10.1145/2043556.2043566. URL: http://doi.acm.org/10.1145/2043556.2043566.

[30] Ling Ren et al. "Constants Count: Practical Improvements to Oblivious RAM". In: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, 2015, pp. 415–430. ISBN: 978-1-931971-232. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/ren-ling.

[31] Marc Shapiro et al. "Conflict-free Replicated Data Types". In: ed. by Xavier Défago, Franck Petit, and V. Villain. Vol. 6976. Grenoble, France, Oct. 2011, pp. 386–400.

[32] Elaine Shi et al. "Oblivious RAM with O((Logn)3) Worst-case Cost". In: *Proceedings of the 17th International Conference on The Theory and Application of Cryptology and Information Security*. ASIACRYPT'11. Seoul, South Korea: Springer-Verlag, 2011, pp. 197–214. ISBN: 978-3-642-25384-3. DOI: 10.1007/978-3-642-25385-0_11. URL: http://dx.doi.org/10.1007/978-3-642-25385-0_11.

[33] Ryan Singel. "Netflix Spilled Your *Brokeback Mountain* Secret, Lawsuit Claims". In: *Wired* (Dec. 2009).

[34] Yair Sovran et al. "Transactional storage for geo-replicated systems". In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. SOSP '11. Cascais, Portugal: ACM, 2011, pp. 385–400. ISBN: 978-1-4503-0977-6. DOI: 10.1145/2043556.2043592. URL: http://doi.acm.org/10.1145/2043556.2043592.

[35] Emil Stefanov and Elaine Shi. "ObliviStore: High Performance Oblivious Distributed Cloud Data Store." In: *NDSS*. The Internet Society, 2013. URL: http://dblp.uni-trier.de/db/conf/ndss/ndss2013.html#StefanovS13.

[36] Emil Stefanov, Elaine Shi, and Dawn Song. "Towards practical oblivious RAM". In: *arXiv preprint arXiv:1106.3652* (2011).

[37] Emil Stefanov et al. "Path ORAM: An Extremely Simple Oblivious RAM Protocol". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13. Berlin, Germany: ACM, 2013, pp. 299–310. ISBN: 978-1-4503-2477-9. DOI: 10.1145/2508859.2516660. URL: http://doi.acm.org/10.1145/2508859.2516660.

[38] A. Thusoo et al. "Hive: a warehousing solution over a map-reduce framework". In: *Proc. VLDB Endowment* 2.2 (2009).

[39] Werner Vogels. "Eventually Consistent". In: *Queue* 6.6 (Oct. 2008), pp. 14–19. ISSN: 1542-7730. DOI: 10.1145/1466443.1466448. URL: http://doi.acm.org/10.1145/1466443.1466448.

[40] Sergey Yekhanin. "Private Information Retrieval". In: 53.4 (Apr. 2010), pp. 68–73.