Research Statement

Geoffrey Ramseyer

My research bridges the design of high-performance computer systems and the theoretical study of the computational and economic tradeoffs implicit in system architectures. I redesign system interfaces to guarantee linear scalability, often through commutativity, which I find provides additional surprising benefits. Conversely, I study the tradeoffs in existing architectures, in order to enable users to make informed decisions and to guide my design of new architectures. Constructing these new interfaces often requires translating theoretical computer science and economics into practice, and in doing so I am guided by my own theoretical analyses.

1 Overview

Too many applications cannot scale to a global deployment, both in terms of computational performance and in terms of support for heterogeneous requirements. Computatinally, countless systems, such as those built in the classic replicated state machine architecture [2], execute operations one by one, by design. This design prevents most or all implementation parallelism on crucial workloads, and thus prevents applications from exploiting the power of modern many-core datacenter hardware. Furthermore, limited application interfaces constrain user choices and give suboptimal economic outcomes.

For example, policymakers are interested in developing new financial infrastructure [9]. Cryptocurrency companies have spent billions building this infrastructure at so-called "scale." Yet these systems typically reach only dozens to hundreds of transactions per second, require layers of complex architecture, and, at best, only scale in narrow settings. These limitations force platforms to charge exorbitant fees. This is absurd—payments should be too cheap to bother metering—and yet these tradeoffs are accepted in the industry as inherent. Realizing the benefits of new financial infrastructure, such as lower transaction costs, improved interoperability, competition between disparate systems, and improved access to financial services, requires high-performance infrastructure that efficiently supports heteregeneous economic tradeoffs.

I approach these problems by re-examining a system's overarching goals. Several of my projects redesign application interfaces both to add commutativity (and thus parallelism) as a first-class feature, and also to provide the flexibility for users to choose their own tradeoffs. These designs are not the standard solutions, but often give surprising benefits beyond scalability, such as novel forms of economic efficiency, fairness, and privacy—all while simplifying a distributed implementation.

Conversely, I also examine the mathematical tradeoffs inherent in existing architectures and interfaces. Many systems give users the agency to make their own well-informed cost-benefit analyses. Yet precise mathematical understanding is required to make a cost-benefit analysis; furthermore, some designs constrain users into suboptimal tradeoffs. Theoretical understanding of the underlying problems leads to improved system designs.

2 SPEEDEX [12]

Interoperability between currencies requires an efficient exchange between currencies. The standard design is a continuous double-auction; that is, users submit offers to trade, and the exchange sequentially either matches a new offer against an earlier one or adds it to a list of open offers. In the classic architecture [2], a fault-tolerant protocol replicates a totally ordered log of new offers, and replicas of this state machine consume log entries sequentially.

This architecture prevents computational scalability. A continuous double-auction is the worst-case workload for optimistic parallelism. Every operation modifies a single data structure, and the order between all operations affects the exchange rate of each trade.

Furthermore, the architecture causes negative economic externalities. Well-connected actors can manipulate offer ordering (so-called "front-running," similar to "high-frequency trading"), extracting risk-free profit at other users' expense. This front-running is especially prevalent on existing decentralized exchanges. This architecture also explicitly matches offers pairwise, implicitly grouping activity by the pair of traded assets. Users therefore face the challenge of routing a trade through intermediate assets, as trading pairs are not uniformly liquid, and the most liquid trading pairs generally involve a reserve currency, like the US Dollar. This challenge makes the overall market less liquid than it could be and creates risk-free arbitrage opportunities, which act as a traffic multiplier and, in practice, have overloaded decentralized exchanges.

Instead, my system, SPEEDEX, leverages the theoretical literature on exchange markets to build a

currency exchange with near-linear computational scalability (despite the worst-case workload), while simultaneously addressing these economic challenges. Instead of processing offers sequentially, the SPEEDEX replicated state machine consumes batches of offers simultaneously. Batches execute in parallel, with minimal thread synchronization. Execution traces interleave nondeterministically, but the end result of applying a batch is deterministic—because SPEEDEX uses a market mechanism that makes offers in a batch *commute*.

SPEEDEX achieves commutativity by settling all trades in a batch at the *same* exchange rate, instead of at different rates. SPEEDEX also processes trades between many currencies simultaneously, not just between pairs of assets separately. Designing this mechanism means translating a batch of offers into a model from the theoretical literature, the Arrow-Debreu exchange market [1]. Exchange rates are quotients of asset "valuations" computed on each batch. Conveniently, in any batch, the valuations that clear the batch (each offer trades iff its limit price is below the batch rate) are unique.[1]

Because every trade in the batch clears at the same rate, this mechanism eliminates the type of front-running (through order manipulation) that is especially prevalent on decentralized exchanges. Additionally, because rates are quotients of valuations, SPEEDEX simplifies trading in any pair of currencies, and eliminates the trade routing problem and the resulting class risk-free arbitrage opportunities.

A key technical challenge is in computing clearing valuations at scale. Binary search solves the case where a batch trades between two assets. However, when many currencies are traded simultaneously, the problem is equivalent to the much harder problem of computing equilibria in linear Arrow-Debreu exchange markets.

Existing theoretical algorithms feature (1) poor asymptotic scaling, (2) unacceptable approximation error, and (3) unworkable empirical performance. SPEEDEX starts with an iterative method [3], but scaling to tens of millions of open offers and hundreds of thousands of trades per second requires new ideas. I leverage market structure not present in the theory to (1) improve the asymptotics to a logarithmic runtime per iteration. I then use a linear program to (2) shape approximation error into acceptable forms, like trade fees. Finally, I achieve good empirical performance through (3) close attention to cache performance while parallelizing each iteration.

There are many natural extensions to SPEEDEX's mechanism, so I also study their economic and computational tradeoffs [13]. I show that several natural economic desiderata are incompatible, so a deployment of a batch trading mechanism must consider its surrounding context.

SPEEDEX is available open-source, both as a standalone implementation and as a prototype component of a public blockchain.

Another surprising benefit of SPEEDEX's commutative design is that it simplifies the process of deploying the new mechanism. The prototype component uses only 10% of the code of the standalone implementation. Changing a blockchains's interface requires a complex, coordinated upgrade, but the hard-to-audit parallel code—which does not change the interface—can be deferred to uncoordinated future upgrades.

3 GROUNDHOG [14]

SPEEDEX's architecture, of blocks of commutative transactions, generalizes to diverse applications far outside of currency exchange.

Programmability enables self-serve innovation and seamless extensibility, and is critical if a system is to support a global population of heterogeneous users and applications. In a replicated state machine, programmability, taken to its limit, means letting each transaction run arbitrary code ("smart contracts") to modify, e.g., a key-value store. Yet the standard sequential architecture gives each transaction a global lock, and arbitrary code can force sequential execution. Sequential execution inherently limits system throughput, and is the root cause of the absurdly high transaction fees in existing blockchain infrastructure. How can a deterministic state machine allow unrestricted programmability and linear scalability through parallelism?

Through my system GROUNDHOG, I tackle this challenge with a novel set of semantics through which arbitrary programs modify a state machine (a key-value store). Like SPEEDEX, it concurrently executes blocks of transactions, while the semantics guarantee transactions in a block *commute*. Transactions read from a snapshot taken at the start of the block, and write typed *modifications* to typed values. The key contribution is the design of a minimal set of sufficiently powerful types.

For example, GROUNDHOG maintains an "integer" type, which is modified by addition and subtraction. Addition and subtraction commute, so the runtime deterministically resolves concurrent modifications to the same key by computing the net change.

Applications also need constraints, such as nonnegativity in account balances. GROUNDHOG maintains constraints through precise assembly of transaction blocks, using a reserve-commit system. Block assembly is as scalable (i.e., linearly) as the process of executing a block of transactions and checking for constraint violations.

Surprisingly, our nonnegative integer primitive, a natural fit for account balances, also implements a semaphore. Applications therefore can implement only the locks

[1] Unique up to rescaling, in nonempty markets.

they need, instead of having to pay for a global lock.

GROUNDHOG demonstrates that this nonnegative integer primitive, along with a bytestring type and a sorted set type, suffice to implement a wide variety of applications. Importantly, they implement, as a fallback, the message-passing "actor" model. Thus, although GROUNDHOG's design differs from the standard, it is strictly more expressive than several production systems, and lets each application choose its own locks and throughput. Applications written to be scalable run at high throughput, and low-throughput applications run slowly, but slow applications cannot slow down fast applications. In other words, applications built in GROUNDHOG could implement individual fee mechanisms, but only if necessary, and users would only pay for the costs they incur—and never pay the absurd cost of a global lock.

My open-source implementation handles more than half a million payments per second. Unlike all prior systems, my implementation is linearly scalable and reaches this throughput regardless of how often transactions contend on the same account.

3.1 Privacy Through Commutativity

A surprising benefit of GROUNDHOG's commutativity is that it enables efficient account audits (and publicly-verifiable proofs of fraud) in the traditional privacy model of financial institutions, without expensive cryptography [15]. This fills a crucial gap in the literature for proving institutional solvency [4]: namely, proving correctness of the movement of funds, not just correctness at snapshots.

Users typically expect their data to be private to only them, their financial institution, and, in certain instances, the government. Yet public blockchains require all transaction history to be public. This requirement is inherent to sequential systems, because verifying one transaction (verifying correctness of reads and writes) requires replaying the entire history.

In contrast, in GROUNDHOG, all transactions in one block read from the same snapshot, and modifications are applied at the end of the block. Thus, it suffices for an institution to publish at each block a commitment to ledger state, to the set of transactions, and to an index of state modifications. A user can verify their own account history with appropriate commitment openings.

Crucially, a user only accesses their own subset of the ledger. Not only does this enable standard access controls, but also means that a user's computational requirements are proportional only to their own usage. For comparison, in a sequential system, each new user imposes additional costs on every other user. Those costs inherently limit a sequential system's ability to scale to many users.

These efficient audits also allow limited end-user hardware to meaningfully interact with a high-performance ledger on datacenter hardware. Equivalently, by publishing a limited set of commitment openings, GROUNDHOG can efficiently prove to external systems that an event (or fraud) happened.

4 Theoretical Tradeoffs in Systems Design

The other side of my research studies the computational and economic tradeoffs inherent in system architectures. Understanding these tradeoffs both enables users to make informed cost-benefit tradeoffs, and guides my research into new system designs, like SPEEDEX, that bypass these tradeoffs.

4.1 Market-Making

As an example, recent years have seen interest in market-making through simple, automated strategies, parametrized by a "trading curve."[2] How can market-makers who use these systems pick one curve over another? I give an explicit transformation [7], through the KKT conditions of an optimization problem, that maps implicit beliefs on future prices and volatility into an optimal curve. The simplicity of these strategies is an artefact of the computational constraints of public blockchains—but my analysis shows the baseline performance of market-making under these limits.

Additionally, users trading with these passive market-makers may want to hide their opinions on asset prices (implicitly, to obscure the size of their trades). Yet adding noise to trades, as in differential privacy, perturbs the market-maker's price quotes, which creates arbitrage opportunities. We characterize [6] the noise distributions that can be appropriately priced, so that privacy-conscious traders pay for exactly the arbitrage they create. Importantly, the resulting mechanism allows each user to choose their own personalized noise distribution based on their own privacy requirements.

4.2 Payment Channel Networks

Payment channels give another tool for accelerating computationally-limited blockchains. In a payment channel, two users deposit collateral and transact by privately changing ownership records of the collateral. Bilateral channels can link together into a network, with payments routed across the graph. Yet not every payment can be routed safely, if, for example, one side of a channel owns all of the collateral. A payment channel network multiplies blockchain throughput, but not infinitely.

Users face a complex tradeoff between network structure, capital costs, and network performance. We

[2] These are often referred to as "Constant-Function Market Makers," and are closely related to prediction markets.[8]

show [5] how to understand this tradeoff, and how a user's individual choices relate to the network's geometry. But we also show that this tradeoff is inherently suboptimal [11]. The fact that a user allocates capital to specific pairwise channels, instead of to all of their channels simultaneously, causes an at-least quadratic reduction in asymptotic performance.

4.3 Distributed Sequencing

Distributed systems need a principled way to sequence operations, especially when ordering carries economic consequences. Yet different parts of a georeplicated system may have different local views on transaction arrival order. The overall system ordering should take each individual view into account.

I argue that this problem is a streaming, unbounded analogue of a classic social choice problem [10]. Adapting a classical algorithm to this setting gives a mathematically well-founded method for reconciling individual preferences on ordering. In fact, the resulting algorithm has strictly stronger fairness and liveness guarantees than the state of the art solutions to this problem in the blockchain literature.

5 Future Directions

From the systems-building side of my research, my implementations empirically support many millions of users, but horizontal scaling to the billions of people in the world requires horizontally scaling across many machines, not just one. Dividing work over physical machines raises the cost of any cross-thread synchronization, but commutativity means that this cost should be marginal in a natural extension of my designs.

One already identifiable challenge will be managing a distributed database tuned for GROUNDHOG and SPEEDEX's workloads. Unlike most applications, my systems make very few (atomic) database transactions, but each transaction is massive and constructed in parallel. Additionally, public-facing infrastructure should hide the layout of data between machines, to help guard against denial of service attacks against a single machine. Yet layout randomization appears in tension with the data locality needed to quickly compute commitments to a key-value store, such as Merkle-Patricia trie root hashes.

SPEEDEX and GROUNDHOG add a new design point to a long literature on deterministic databases and consistency in distributed systems. This architecture could apply to use cases beyond financial infrastructure, but may require new abstractions beyond GROUNDHOG's primitives, or new big-picture methods for clearing batches of operations, as in SPEEDEX. Additionally, because this architecture differs from the standard sequential model, developers may find it challenging to verify application code correctness. However, scalability presents new opportunities. For example, GROUNDHOG might run real-time security analyses, such as runtime checking of user-written security predicates.

And conversely, I want to continue to study the interaction of computation and economics in my new system interfaces. As a concrete example, SPEEDEX's ability to compute equilibria efficiently relies on the fact that all offers trade from one asset to one other asset. In the Arrow-Debreu exchange market model, this corresponds to agents with marginal utility for only two assets. We show this class of utilities can be generalized well beyond linear utilities [13] while retaining computational efficiency, but natural use-cases in the real world necessitate more complex utilities. For example, a trader might offer to sell stock in exchange for any combination of cash and government bonds, or might sell a bitcoin in exchange for any combination of USD-pegged tokens. Assets with varying fungibility, such as bonds with varying maturities, present new challenges for SPEEDEX's design.

I also suspect that continued study of the tradeoffs in existing systems will continue to point to interesting system designs. Preliminary results suggest that different ways of combining the peer-to-peer transaction systems can lead to qualitatively different system dynamics. novel combinations of some of the peer-to-peer transaction systems I have studied on their own qualitatively change the economic calculus of individual users. And my study [13] of the interaction between my systems' batch execution model and other economic primitives has only begun to scratch the surface of a fascinating research area.

The near future represents an opportunity for engineers and policymakers to build new digital infrastructure linking together all of the disparate systems that we have today. Secure, distributed transactions across systems that today remain locked in proprietary silos could provide seamless interoperability, increased competition, and accelerated innovation. But realizing these benefits requires infrastructure that is high-performance, extensible, and that supports customizable, flexible economic tradeoffs. However, the system interfaces we build we will be stuck with for decades, making good interface design crucial. No existing, sequential architecture is up to this task. Academia's neutral status between government and industry makes it the ideal place to do this research and to collaborate with policymakers, with the financial industry, and with the broader research community.

References

[1] Kenneth J Arrow and Gerard Debreu. Existence of an equilibrium for a competitive economy. *Econometrica: Journal of the Econometric Society*, pages 265–290, 1954.

[2] Miguel Castro, Barbara Liskov, et al. Practical

byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[3] Bruno Codenotti, Benton McCune, and Kasturi Varadarajan. Market equilibrium via the excess demand function. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 74–83, 2005.

[4] Gaby G Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 720–731, 2015.

[5] Ashish Goel and Geoffrey Ramseyer. Continuous credit networks and layer 2 blockchains: Monotonicity and sampling. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 613–635, 2020.

[6] Mohak Goyal and Geoffrey Ramseyer. Pricing personalized preferences for privacy protection in constant function market makers. In *Proceedings of the 2023 ACM CCS Workshop on Decentralized Finance and Security*, 2023.

[7] Mohak Goyal, Geoffrey Ramseyer, Ashish Goel, and David Mazières. Finding the right curve: Optimal design of constant function market makers. EC '23, page 783–812, New York, NY, USA, 2023. Association for Computing Machinery. DOI 10.1145/3580507.3597688.

[8] Robin Hanson. Logarithmic markets coring rules for modular combinatorial information aggregation. *The Journal of Prediction Markets*, 1(1):3–15, 2007.

[9] Board of Governors of the Federal Reserve System. Federal reserve announces that its new system for instant payments, the fednow® service, is now live. `https://web.archive.org/web/20230721012415/https://www.federalreserve.gov/newsevents/pressreleases/other20230720a.htm`, June 2023.

[10] Geoffrey Ramseyer and Ashish Goel. Fair ordering via streaming social choice theory. *arXiv preprint arXiv:2304.02730*, 2023.

[11] Geoffrey Ramseyer, Ashish Goel, and David Mazières. Liquidity in credit networks with constrained agents. In *Proceedings of The Web Conference 2020*, pages 2099–2108, 2020.

[12] Geoffrey Ramseyer, Ashish Goel, and David Mazières. SPEEDEX: A scalable, parallelizable, and economically efficient decentralized EXchange. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 849–875, Boston, MA, April 2023. USENIX Association.

[13] Geoffrey Ramseyer, Mohak Goyal, Ashish Goel, and David Mazières. Augmenting batch exchanges with constant function market makers. *arXiv preprint arXiv:2210.04929*, 2022.

[14] Geoffrey Ramseyer and David Mazières. Groundhog: Scaling smart contracting through commutative transaction semantics. 2023.

[15] Geoffrey Ramseyer and David Mazières. Heterogenous and efficient partial auditing of replicated state machines. 2023.