The rise of the cloud computing paradigm realized through hyperscale data centers has rapidly transformed data management infrastructures from centralized databases to distributed systems. Cloud-based systems typically assume trusted infrastructures and tolerate only crash failures. Increasing the number of malicious attacks on the cloud, on one hand, and the rapid popularity of blockchain, on the other hand, have shifted the focus of fault tolerance from tolerating crash failures to tolerating malicious failures, making untrusted infrastructures more prevalent and commonplace for storing data. Tackling practical large-scale problems on untrusted infrastructures requires addressing the natural tension between scalability, fault tolerance, and trustworthiness, which can be made possible through leveraging scalable data management solutions as well as cryptographic and distributed computing approaches developed to restrict malicious behavior.

My research aims to bridge large-scale data management and distributed fault-tolerant systems in the presence of untrustworthy infrastructures. My long-term goal is to build a complete ecosystem that efficiently manages large-scale data on distributed untrusted infrastructures. Since the current data management systems are mainly built for trusted infrastructures, a significant redesign and reconsideration of the fundamentals and principles of distributed data management are needed. To realize this vision, I study the fundamental distributed data management functionalities, such as transaction processing and consensus. Specifically, the focus of my research, both in my dissertation research [1] and subsequent postdoc work, has been on three main directions and associated daunting challenges as follows.

- First, designing database-enabled techniques to address performance and scalability requirements of managing large-scale data in untrusted infrastructures [3, 4, 7, 11, 14, 15, 17, 19] (Thrusts 1 and 2).
- Second, leveraging privacy and cryptography techniques to support confidentiality and verifiability in the context of cooperative data management [2, 10, 12, 13] (Thrusts 3 and 4).
- Third, addressing the adaptivity requirement of large-scale systems in all infrastructure, transaction processing, and consensus layers by relying on resource desegregation and machine learning approaches [18, 22, 23] (Thrust 5).

Figure 1 presents a summary of my research. In the rest of this statement, I discuss the five requirements, performance, scalability, confidentiality, verifiability, and adaptivity and how my research addresses them in different systems. Some of the systems I have built are designed for specific environments, e.g., cloud, and specific applications, e.g., supply chain. Other than the main research papers and software systems, the figure also categorizes my conference tutorials [5, 6, 16, 21], and published book [8] under the main requirements. The statement ends with three future directions of my research.

Figure 1: A summary of my research.
1 Efficient execution of transactions: performance

The main stages of processing transactions in fault-tolerant systems are order and execution. The order stage uses a consensus protocol to assign an order to a transaction in the global service history, while the execution stage executes the transaction in that order. In this line of work, I focus on the execution stage to improve performance.

1.1 Graph-based concurrency control. Traditionally transactions are executed sequentially to ensure state consistency across all replicas. Sequential execution of transactions, however, limits performance, especially when parallel ordering is not an option due to resource limitations. This is particularly challenging in blockchain systems that are supposed to execute complex compute-intensive smart contracts. In an effort to address this challenge, we designed ParBlockchain [4]. ParBlockchain uses a dependency-graph-based concurrency control mechanism to execute non-conflicting transactions in parallel. ParBlockchain targets blockchain environments and shows significant performance improvement in typical workloads with low to moderate degrees of contention, e.g., 5.7 times higher throughput and 84% less latency compared to the sequential execution paradigm and 2.4 times higher throughput and 87% less latency compared to the state-of-the-art permissioned blockchain system, Hyperledger Fabric.

1.2 Data-dependent order-fairness. Dependency graphs can also be used to address the performance challenge of fair ordering. Existing consensus protocols typically rely on a designated leader to receive transactions from clients, assign order to each transaction, and initiate agreement among all replicas. Nevertheless, a malicious leader can control transactions’ inclusion and final ordering without violating safety or liveness. Existing Byzantine fault-tolerant (BFT) protocols either allow the adversarial manipulation of the actual ordering of transactions or incur high-performance overhead. We observed that fair ordering is essential when transactions access the same data objects. Based on this observation, we defined the notion of data-dependent order-fairness to ensure fair ordering of data-dependent transactions and developed a high-performance fair ordering protocol, Rashnu [17]. Rashnu demonstrated up to 233% throughput improvement compared with the state-of-the-art order-fairness protocol, Themis.

1.3 Hybrid clouds. While large enterprises might have their own geo-replicated fault-tolerant cloud storage, smaller enterprises may only have a local private cloud that is lacking in resources to guarantee fault tolerance. The trustworthiness of a private cloud allows an enterprise to build services that can utilize crash fault-tolerant protocols. Nevertheless, due to a lack of private resources, if a third-party public cloud is used, the nodes of the public cloud may behave maliciously, in which case the enterprise is forced to use Byzantine fault-tolerant protocols. To tackle this challenge, we developed SeeMoRe [14], a hybrid consensus protocol that uses the knowledge of where crash and malicious failures may occur in a public/private cloud environment to improve overall performance. SeeMoRe reduces the number of communication phases, and messages exchanged and also requires fewer replicas to tolerate a given number of malicious and crash failures, reducing an application’s overall deployment cost.

1.4 Declarative logic of transactions. Finally, to improve the execution stage’s performance, we also focused on the execution logic of transactions. Specifically, we considered smart contracts, programs stored and executed on blockchains, due to their complex logic. The complexity of writing and analyzing smart contracts leads to programs with high execution costs, though avoidable. In this line of work, we focus on designing an easy-to-use declarative programming language, DeCon [19], for implementing smart contracts and specifying contract-level properties. Driven by the observation that smart contract operations and contract-level properties can be naturally expressed as relational constraints, we modeled each smart contract as a set of relational tables that store transaction records. This relational representation of smart contracts enables convenient specification of contract properties, facilitates run-time monitoring of potential property violations, and enables debugging via data provenance. The follow-up work to DeCon has been submitted to the NSF SaTC program as a proposal.

2 Coordination on untrusted infrastructures: scalability

Partitioning the data into multiple shards that are maintained by different clusters of (crash-only) nodes is a proven approach for improving the scalability of distributed databases. In the presence of Byzantine nodes, however, traditional (coordinator-based) sharding mechanisms are not applicable. The aim of my work in this thrust is to design scalable Byzantine fault-tolerant data management systems.

2.1 Flattened sharding. Using the sharding mechanism, different clusters process disjoint sets of transactions in parallel. Sharded systems rely on a coordinator node to process cross-shard transactions, which access multiple shards, using a commitment protocol, e.g., two-phase commit. In the presence of malicious nodes, however, the system cannot rely on a single coordinator, as the coordinator node itself might be malicious. Moreover, relying on a single coordinator (or even a Byzantine fault-tolerant cluster of coordinators) limits the ability of the system to process cross-shard transactions in parallel. To address these challenges, we built a scalable distributed transaction processing protocol, SharPer [3,7]. Unlike traditional single-leader consensus protocols, in SharPer, multiple clusters,
each with its own leader, compete with each other to order cross-shard transactions. SharPer processes cross-shard transactions in a flattened manner by running agreements among the nodes of involved shards without requiring a coordinator node. The flattened nature of the cross-shard consensus in SharPer enables parallel processing of transactions with non-overlapping clusters. This setting has been encountered neither in traditional consensus protocols nor in coordinator-based sharded systems, leading us to resolve challenges such as conflicting transactions, deadlock situations, as well as the simultaneous failure of leader nodes across different replicated domains.

2.2 Hierarchical edge networks. SharPer is able to outperform existing scalability solutions by processing cross-transactions in parallel without relying on a coordinator node. However, its performance is still inefficient when the system is deployed over wide-area networks, and the involved shards are far apart due to several messages crisscrossing over high-latency low bandwidth links on the Internet. Coordinator-based approaches do not fare much better than the flattened approach of SharPer in such a scenario, as the coordinator node is either close to clients or the data shards, which will not avoid slow network links when cross-shard transactions take place. This is especially important in delay-sensitive edge applications deployed on untrustworthy edge computing infrastructures where establishing consensus among edge servers requires Byzantine fault-tolerant protocols. In edge infrastructures, nodes are placed in a hierarchical structure on the spectrum between edge devices and hyperscale clouds, with edge and fog servers in between. We focus on leveraging the hierarchical structure of edge infrastructures to reduce wide-area communication overhead by localizing network traffic for consensus and replication within local networks. This resulted in Saguaro [11], a scalable system that can process cross-shard transactions in edge networks by relying on the lowest common ancestor of all involved domains. Saguaro also uses the hierarchical structure of edge networks for data aggregation across different shards as well as optimistic processing of cross-shard transactions. The design and development of Saguaro was the focus of our CISE/CNS proposal funded by NSF.

2.3 Confining maliciousness within zones. Scalable geo-distributed systems might also need to enforce network-wide policies on all zones (clusters). In particular, in an edge application, we might want to ensure that a zone cannot host more than 10000 edge devices or that a mobile client can migrate at most 10 times a year. The standard way to ensure this is to maintain global system meta-data on all nodes, including the required data for enforcing policies. However, the global synchronization among all zones in order to update the global system meta-data is expensive, as it requires consensus among all zones. To address this challenge, we developed Ziziphus [15], a scalable geo-distributed system that supports edge computing applications with mobile edge clients. Ziziphus performs global synchronization using a lightweight Paxos-style protocol among the leader nodes of different zones by confining the maliciousness of Byzantine servers within each zone.

3 Towards collaborative environments: confidentiality

Moving towards collaborative environments with multiple enterprises, data confidentiality becomes paramount. In distributed collaborative environments, multiple mutually distrustful enterprises collaborate to provide different services. As a result, we need to address untrustworthiness in both enterprises’ infrastructures and their collaboration.

3.1 View-based confidentiality. In collaborative environments, the common public data accessed by global transactions across all enterprises, e.g., a place order transaction in a supply chain workflow, must be visible to all enterprises enabling them to verify transactions. However, enterprises want to keep their internal data, which are accessed by internal transactions, confidential. The existing cryptography-based techniques, which prevent the access of irrelevant parties to data, e.g., using encryption, mainly suffer from the computation and communication overhead. We took the first step towards a scalable, confidential cross-enterprise solution by developing CAPER [2]. CAPER presents a view-based confidentiality technique that maintains different local views of data on different parties while guaranteeing data consistency across views using DAG-structured logs (ledgers). CAPER further introduces different consensus protocols to globally order cross-enterprise transactions. CAPER processes low-contention workloads with 12 times higher throughput at the same latency compared to Hyperledger Fabric.

3.2 Hierarchical data model. In addition to local and global transactions, any subset of enterprises involved in a collaboration workflow might want to keep their collaboration private from other enterprises. For example, a supplier might want to make private transactions with a manufacturer to keep some terms of trade confidential from other enterprises. Moreover, even if the access of irrelevant parties to confidential data is prevented, an attacker might compromise some nodes within an enterprise’s untrusted infrastructure, resulting in confidential data leakage. We developed Qanaat [13] to address these two challenges. Qanaat proposes a novel hierarchical data model consisting of a set of data collections to support confidential collaboration (i.e., data sharing) among any subset of enterprises.

---

1 CNS Core: Small: Towards Internet-scale Permissioned Blockchain Infrastructure for the Mobile Internet, Boon Thau Loo (PI), Mohammad Javad Amiri (co-PI), https://nsf.gov/awardsearch/showAward?AWD_ID=2104882
To prevent confidential data leakage despite the Byzantine failure of nodes, Qanaat utilizes the privacy firewall technique and (1) separates ordering nodes that agree on the order of transactions from execution nodes that execute transactions and maintain the ledger and (2) uses privacy filters between execution nodes and ordering nodes.

4 Tension between privacy and transparency: verifiability

In many cross-enterprise environments, participants need to verify transactions that are initiated by other enterprises to ensure the satisfaction of some constraints in a privacy-preserving manner.

4.1 Anonymous tokens. Recently, there has been increasing interest by governmental, legal and social institutions to enforce regulations, such as minimal and maximal work hours, on crowdfunding platforms. Platforms within multi-platform crowdfunding systems, therefore, need to collaborate to enforce cross-platform regulations. For example, the total work hours of a worker, who might work for multiple crowdfunding platforms, per week may not exceed 40 hours to follow Fair Labor Standards Act\(^2\) (FLSA). While collaborating to enforce global regulations requires the transparent sharing of information about tasks and their participants, the privacy of all participants needs to be preserved. To address the tension between privacy and transparency in the context of multi-enterprise environments, we developed Separ [10]. Separ enforces privacy using lightweight and anonymous tokens, while transparency is achieved using fault-tolerant blockchain ledgers shared among multiple platforms.

4.2 Zero-knowledge proofs. While Separ guarantees verifiability, it still relies on a centralized trusted authority to model global regulations using anonymous tokens and distributes them to participants. Other than enforcing regulations, verifiability might be needed to prove the validity of transactions, e.g., privacy-preserving data provenance. The privacy-preserving verifiability can be supported in a decentralized manner by leveraging zero-knowledge proofs, as we demonstrated in PriProVer [12].

5 Tackling diverse and dynamic workloads: adaptivity

Data management service providers need to guarantee high-throughput services over various transaction workloads. In this line of research, I present different techniques at infrastructure, transaction processing architecture, and the consensus protocol level to deal with diverse and dynamic workloads.

5.1 Resource disaggregation. Transactions might perform memory-intensive operations, compute-intensive operations, or a mix of both at different times and execution stages. These diverse hardware demands require rethinking existing system architectures to enable the flexibility of scaling compute and memory resources independently and elastically. Recently, resource disaggregation has become a trend in data center design. This looming resource disaggregation model will manifest itself by reorganizing resources from servers to physically distinct pools that are dedicated to processing, memory, or storage. To demonstrate the effectiveness of resource disaggregation in addressing the diverse resource requirements of workloads on untrusted infrastructures, we, for the first time, developed a novel disaggregated permissioned blockchain system, FlexChain [22]. FlexChain comprises a tiered key-value store based on disaggregated memory and storage that elastically scales the state.

5.2 Adaptive blockchains using reinforcement learning. Other than the underlying hardware resources, the choice of transaction processing architecture significantly impacts system performance. Distributed systems present significant variations in architectural design, including the sequence in which ordering and execution are performed, the number of transactions in a batch, stream processing (with no batch), and the use of reordering and early aborts. Depending on the workload characteristics, different architectures exhibit various performances. This demonstrates the need for a framework that adaptively chooses the best architecture in order to optimize throughput for dynamic transaction workloads. AdaChain [23] addresses this challenge by automatically adapting to an underlying, dynamically changing workload through the use of reinforcement learning. AdaChain switches from the current architecture to a promising one at runtime in a way that respects correctness and security concerns.

5.3 Unified platform for BFT protocols. BFT protocols are different along several dimensions, including the number of replicas, processing strategy (i.e., optimistic, pessimistic, or robust), supporting load balancing, etc. While dependencies and trade-offs among these dimensions lead to several design choices, there is currently no unifying tool that provides the foundations for studying and analyzing BFT protocols’ design dimensions and their trade-offs. We envision that such a unifying foundation will provide an in-depth understanding of existing BFT protocols, highlight the trade-offs among dimensions, and enable protocol designers to choose among several dimensions to find the protocol that best fits the characteristics of their applications. This, however, requires us to extract essential elements of the agreement and define atoms that connect these elements [20]. In an effort to address this challenge, we developed Bedrock [18] that can be used to analyze and navigate the evergrowing BFT landscape. Designing such a platform is the first step toward developing an adaptive BFT protocol selection approach.

\(^2\)https://www.dol.gov/agencies/whd/flsa
6 Future Directions

My plans for future work center around my long-term career goal to build a complete ecosystem to enable large-scale data management in untrusted environments.

6.1 One size does not fit all: be adaptive. Studying different transaction processing architectures showed us that depending on workload and hardware characteristics, the performance of a given architecture can vary drastically, hence, there is no one-size-fits-all architecture. In our experiments, we observed that not only is there no dominant architecture, but some architectures that perform well on one workload can perform quite poorly on another. I envision that adaptivity is the main feature of the next generation of transaction processing systems. On one hand, such systems need to support automatically adapting to an underlying, dynamic workload. On the other hand, they need to detect failures and attacks and take appropriate actions against them, e.g., by switching from an optimistic to a pessimistic BFT protocol. Adaptivity needs to be considered at three different levels. First, at the infrastructure level, the system should be able to use compute and memory resources independently and elastically. Second, at the transaction processing level, the system needs to switch between different architectures, and finally, at the consensus level, the system must choose the best protocol among dozens of BFT protocols. Putting all these three components together, an adaptive system must respond to the dynamicity of workloads, failures, and attacks by simultaneously choosing the best hardware, transaction processing architecture, and consensus protocol. This requires leveraging resource disaggregation and reinforcement learning approaches.

6.2 Dynamic data: privacy, regulations and verification. The privacy of stored data and the privacy of querying data at a large scale have been widely studied. However, databases are not solely query engines on static data; they must support updates on dynamically evolving datasets. Moreover, the underlying infrastructure might consist of a single database stored remotely on untrusted providers or multiple independent databases owned by different mutually distrustful enterprises. Furthermore, the updates need to be verified against internal constraints or global regulations before being executed on databases. This opens the space of privacy-preserving data management from the narrow perspective of private queries on static datasets to the larger space of private management of dynamic data. My goal is to design a universal framework for managing regulated dynamic data in a privacy-preserving manner. In a nutshell, the framework needs to support scalable, efficient, consistent, and verifiable execution of updates on data with regulations while preserving privacy. Designing this framework requires solving multiple challenges. First, the efficient verification of an update with respect to a given generally formulated regulation where the update and the regulation or both might have privacy constraints. Second, the efficient execution of an update on data where the update and the data or both might have privacy constraints. Finally, the scalability of these solutions with respect to the frequency of updates as well as the size of the data. We have taken the first step toward this goal on PReVer [9].

6.3 It’s time to substitute coordination with computation: be uncoordinated. The evolution of technology trends over the past four decades provides an abundance of networking, computing, memory, and storage resources. While in the early days of computer technology, CPU, storage, and memory were expensive, coordination was effectively free, coordination has now become a precious commodity as it is fundamentally limited by the speed of light4. This dramatic shift in technology trends drives us to think about new techniques that require less coordination in distributed settings. Permissionless blockchains can be seen as an example of substituting coordination with computation. While traditional consensus protocols rely on coordination to elect a leader and establish consensus among nodes, permissionless blockchains use computations, e.g., proof of work, for this purpose. The next generation of fault-tolerant protocols needs to tackle the high overhead of coordination. Directed acyclic graph (DAG)-based consensus protocols are one step towards this vision. DAG-based protocols support load balancing by separating transaction dissemination (by all replicas) from ordering and provide higher throughput.

7 Collaborations

Realizing the vision of managing large-scale data on distributed untrusted infrastructures requires expertise in many fields. During the course of my Ph.D. and postdoc, in addition to my Ph.D. and postdoc advisors, I have invested in collaborating with colleagues with similar interests from the University of Pennsylvania (Profs. Mayur Naik, Brett Falk, and Ryan Marcus), UC Santa Barbara (Prof. Cynthia Stohl), UC Davis (Prof. Mohammad Sadoghi), University Rennes (Prof. Tristan Allard), University of Waterloo (Prof. Sujaya Maiyya), CUHK (Prof. Eric Lo), Georgetown University (Prof. Wenchao Zhou), and Simon Fraser University (Prof. Yuepeng Wang) as well as 24 researchers and students across multiple institutes in academia and industry on a variety of topics. I have also collaborated with and mentored several undergraduate and master’s students in their research. See my teaching statement for more details.

4https://pathelland.substack.com/p/i-am-so-glad-im-uncorordinated
References


