Towards Full Stack Adaptivity in Permissioned Blockchains

Chenyuan Wu
University of Pennsylvania
wucy@seas.upenn.edu

Mohammad Javad Amiri
Stony Brook University
amiri@cs.stonybrook.edu

Haoyun Qin
University of Pennsylvania
qhy@seas.upenn.edu

Bhavana Mehta
University of Pennsylvania
bhavanam@seas.upenn.edu

Ryan Marcus
University of Pennsylvania
rcmarcus@seas.upenn.edu

Boon Thau Loo
University of Pennsylvania
boonloo@seas.upenn.edu

ABSTRACT

This paper articulates our vision for a learning-based untrustworthy distributed database. We focus on permissioned blockchain systems as an emerging instance of untrustworthy distributed databases and argue that as novel smart contracts, modern hardware, and new cloud platforms arise, future-proof permissioned blockchain systems need to be designed with *full-stack adaptivity* in mind. At the application level, a future-proof system must adaptively learn the best-performing transaction processing paradigm and quickly adapt to new hardware and unanticipated workload changes on the fly. Likewise, the Byzantine consensus layer must dynamically adjust itself to the workloads, faulty conditions, and network configuration while maintaining compatibility with the transaction processing paradigm. At the infrastructure level, cloud providers must enable cross-layer adaptation, which identifies performance bottlenecks and possible attacks, and determines at runtime the degree of resource disaggregation that best meets application requirements. Within this vision of the future, our paper outlines several research challenges together with some preliminary approaches.

PVLDB Reference Format:
Chenyuan Wu, Mohammad Javad Amiri, Haoyun Qin, Bhavana Mehta, Ryan Marcus, and Boon Thau Loo. Towards Full Stack Adaptivity in Permissioned Blockchains. PVLDB, 17(5): XXX-XXX, 2023.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Today's large-scale distributed data management systems need to deal with untrustworthy environments where multiple mutually distrustful entities communicate with each other, and maintain data on untrusted infrastructure. By relying on Byzantine fault-tolerant (BFT) protocols, untrustworthy distributed databases, in particular, permissioned blockchain systems, have enabled a large class of distributed applications ranging from contact tracing [63], crowdworking [8], supply chain assurance [9, 79], and federated learning [64]. In fact, the popularity of these services has motivated cloud providers (e.g., Amazon [1, 2], IBM [3], Oracle [4], Alibaba [83]) to offer Blockchains-as-a-Service (BaaS) [28]. However, there are two categories of outstanding challenges for today's permissioned blockchain systems:

Challenge 1 (Software): No one-size-fits-all transaction management paradigm. Different applications may exhibit different workload characteristics, such as read/write ratios, skewness of popular keys, and compute intensity. A proliferation of permissioned blockchain systems is proposed to address these workload variations, e.g., Tendermint [49], Fabric [11], Fabric++ [74], Fabric# [71], Streamchain [44], ParBlockchain [7], and BIDL [67]. These systems present different transaction management paradigms, including the sequence in which ordering, execution, and validation are performed, the number of transactions in a block, stream processing (with no blocks), and the use of reordering and early aborts.

Moreover, each such paradigm can be further customized with a broad spectrum of Byzantine fault-tolerant (BFT) protocols, such as PBFT [20], Q/U [5], FaB [56], Zyzzyva [48], Prime [6], CheapBFT [45], SBFT [40], HotStuff [85], Themis [46], PoE [41], and Kauri [60]. These BFT protocols also embody different design choices, including commitment strategy, number of communication phases, and leader replacement mechanism, etc. [10].

Past studies [22, 34] have shown that different systems are optimal for different workloads and faulty conditions. Unfortunately, current users have to choose a fixed paradigm with a predetermined BFT protocol, potentially resulting in poor performance, as *no single configuration provides dominant performance.* Even when the user has control over the entire blockchain stack, choosing the right configuration is not easy, given the large search space of paradigms and BFT protocols. Moreover, the workload may shift as different parties join or leave the network, and client requests fluctuate with different patterns throughout the day. Malicious participants might carry out dynamic multi-vector attacks as well. Of course, one could imagine building a static mapping from the workload and attack characteristics to optimal configurations – but this mapping would (1) be cumbersome to compute, (2) depend on the underlying hardware, (3) still be suboptimal for workloads that shift unexpectedly over time, and (4) necessitate recomputing the mapping each time a new paradigm or BFT protocol is proposed.

Challenge 2 (Hardware): No one-size-fits-all resource provisioning. Smart contracts have been traditionally used to perform memory-intensive operations such as retrieving customer profiles stored in large databases [30]. However, recent advancements in decentralized applications [27, 32, 37, 66, 86] have showcased compute-intensity in smart contracts, where machine learning algorithms are integrated on-chain. Cortex Blockchain [27], for instance, offers a decentralized lending platform that leverages machine learning algorithms to determine interest rates based on individual credit histories. This transparency fosters trust, prevents discrimination, and ensures fairness throughout the lending process. Moreover, a smart

Proceedings of the VLDB Endowment, Vol. 17, No. 5 ISSN 2150-8097.
doi:XX.XX/XXX.XX

contract can interchangeably be compute- and memory-intensive at different times and execution stages, exhibiting dynamic and heterogeneous resource requirements.

Current BaaS offers a convenient way for users to manage their resources. However, the current rigid BaaS infrastructure poorly supports such heterogeneous resource demands. First, to ensure high-throughput services over various workloads, servers are usually over-provisioned, leading to low utilization of certain or all types of resources as well as high cost for BaaS users. Second, as workloads evolve, it is hard to scale up/down resources independently and seamlessly, e.g., expanding/shrinking memory resources independent of CPUs without pausing transactions. These limitations stem from the fact that data center resources have been traditionally arranged in monolithic servers, which contain fixed compute, memory, and storage resources for processing jobs. Thus, deploying smart contracts with diverse resource demands requires a radical rethinking of existing BaaS infrastructure.

This paper articulates our vision for *FAB*, an adaptive framework for untrustworthy distributed databases, which leverages machine learning techniques to address the above challenges. The focus of this paper is mainly on permissioned blockchain systems, as it is the most notable instance of untrustworthy distributed databases. FAB is designed with *full-stack adaptivity* in mind. First, FAB not only needs to adaptively choose the best transaction management paradigm in order to maximize performance for dynamic workloads, but also quickly adapt to new hardware and unanticipated workload changes on-the-fly. Second, the BFT protocol chosen for consensus should also automatically adapt to the dynamic workloads, ever-changing faulty conditions, and network configuration while maintaining safety and liveness. Finally, at the infrastructure level, FAB leverages resource disaggregation [33, 73, 87, 88] whenever appropriate in order to adapt to diverse resource demands. Interestingly, some paradigms and workloads benefit immensely from a resource-disaggregated setting, while others may backfire – motivating the need for a cross-layer adaptation strategy.

2 BACKGROUND

Distributed systems rely on fault-tolerant protocols to provide robustness and high availability. While cloud systems [18, 26, 29] rely on crash fault-tolerant protocols [17, 51–53, 61], to establish consensus, a Byzantine fault-tolerant (BFT) protocol is a key ingredient in distributed systems with untrustworthy infrastructures [10]. A BFT protocol runs on a network consisting of a set of nodes that may exhibit arbitrary, potentially malicious behavior. BFT protocols use the State Machine Replication (SMR) algorithm [50, 72] where the system provides a replicated service whose state is mirrored across different deterministic replicas. At a high level, the goal of a BFT SMR protocol is to assign each client request an order in the global service history and execute it in that order [75].

The performance of a given distributed system component depends on the user's workload and the underlying hardware. Since choosing the correct configuration of components for a given workload and hardware is non-trivial [10], several works have turned to machine learning techniques [14, 54, 82]. This turn is parallel to several other fields [36], such as programming languages [21, 42], image processing [68], and relational databases [80]. Many of these approaches follow a simple pattern: collect a training set of features/configuration/performance triplets, and train a supervised ML model to predict the performance of a given configuration given some features. While this approach is powerful, it requires generating an exhaustive training set, which is both expensive and must be redone whenever new configurations are invented. Thus, some recent work has shifted towards using *reinforcement learning (RL)* [12, 77] in order to learn a policy while the system is running, in an online fashion [54, 55, 82, 84]. While these RL techniques avoid costly training set generation, they must balance the exploration of new configurations (e.g., trying a new configuration with unknown performance) and the exploitation of prior knowledge (e.g., picking a configuration that worked well in the past).

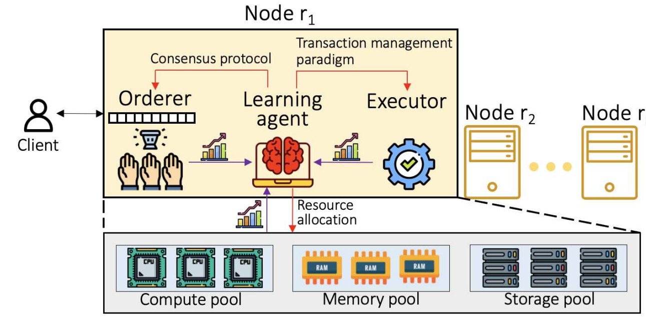

Figure 1: FAB framework overview.

3 FAB OVERVIEW

In this section, we first provide a running example that motivates FAB, and then introduce FAB's system model. Our running example below is based on supply chain management, which is a cross-enterprise application that includes untrustworthy entities.

Varying workloads. First, supply chain management involves different types of transactions, exhibiting different workload characteristics. For instance, when merchants want to check the temperature of certain containers, they issue inspection transactions which are more *read-heavy*. When clients purchase some items or merchants push shipping notices, they issue inventory transactions which are more *write-heavy*. Moreover, since some items are more popular than others, transactions are *skewed* based on what keys they accessed. Past studies [22, 34, 82] have shown that depending on workload and hardware characteristics, the performance of a given transaction management paradigm can vary drastically. For example, as shown in Table 1, under a highly compute-intensive workload A with high skewness but low write ratio, an Execute-Order-Validate (XOV) with reordering paradigm [74] provides the best throughput, outperforming the next best by 15%; whereas under a contentious workload D with high write ratio, the same paradigm performs worst with near-zero throughput.

Presence of faults. Second, supply chain systems suffer from different attacks and malicious behaviors [19, 69, 76]. For example, in the context of COVID-19 vaccine supply chain, attacks on 44 companies involved in the vaccine distribution in 14 countries [62] are reported. While the safety and liveness of the system are guaranteed by the BFT consensus, adversaries such as a competing manufacturer could still slow down the system significantly if a

Table 1: Effective throughput (tps) for each paradigm in the last 20 episodes of each workload and the convergence time (minutes) of AdaChain [82].

Workload	Effective Throughput					AdaChain's Conv. Time
	XOV+reorder	XOV	OXII	OX	AdaChain	
A	**1532**	1415	968	194	1425	2.48
B	897	866	**1545**	861	1426	0.62
C	3228	**3235**	940	98	3153	0.48
D	1	272	1494	**1498**	1447	0.43
Average	1414	1447	1237	663	**1862**	1.00
Worst	1	272	940	98	**1425**	2.48

fixed BFT protocol is used consistently. For instance, Zyzzyva [48] performs well in normal cases, but even with benign crash faults, it significantly falls behind pessimistic protocols like PBFT [20]. As another example, if PBFT's leader is hijacked to broadcast proposals slowly, the end-to-end system performs poorly, whereas switching to a robust protocol like Prime [6] avoids such an issue.

Diverse application resource needs. Last but not least, supply chain management systems demonstrate diverse resource demands overtime. For instance, once merchants have ordered some items from the manufacturer, they might frequently query the predicted delivery time of their items in order to make business planning accordingly. Such predictions incur regression tasks on-chain, resulting in a higher demand for computation resources than usual. This case study of supply chain management suggests adaptivity is not only required, but also needs to be tackled in a *full-stack* manner. Given the exponentially large state space and the intricate interference between actions, we argue that instead of taking on the Sisyphean task of manually crafting heuristics, machine learning is a promising solution to achieve such full-stack adaptivity.

System model. Figure 1 presents an overview of FAB, which comprises a fixed set of *nodes* and a finite number of *clients*. FAB follows the Byzantine failure model, where up to f nodes and any number of clients may exhibit arbitrary, potentially malicious behavior. We assume the Byzantine failure model, as it encompasses the crash failure model, and the non-trustworthiness of nodes is a widely accepted assumption in most blockchain environments. Clients submit requests to nodes and await responses from $f + 1$ nodes. Each node fulfills multiple roles concurrently: *orderer*, *executor*, and *learning agent*. The orderer ensures the total ordering of blocks, while the executor executes transactions, updates the ledger, and responds to clients with the results. The learning agent gathers data continuously, trains a machine learning model, and dynamically instructs the accompanying orderer and executor to replace current transaction management and consensus protocol with more effective alternatives. Whenever appropriate, the learning agent also maps the node to resource pools in a disaggregated infrastructure. In FAB, a faulty node may act arbitrarily in any of its roles.

4 ADAPTING TRANSACTION MANAGEMENT PARADIGM

As our first step towards adaptive transaction management paradigms, we have developed AdaChain [82], a strawman solution capable of selecting the best-performing paradigm at run-time according to the workload, in the context of permissioned blockchains. Table 1 shows how AdaChain performs compared to other fixed paradigms when running four different workloads. For each paradigm, we measured its average and worst throughput across all workloads. The worst-case throughput metric is useful for understanding how robustly a paradigm performs when the setup changes. AdaChain achieved the highest average throughput and the highest worst-case throughput, demonstrating preliminary adaptivity. We next outline AdaChain's limitations and the specific research agenda required to fully realize FAB's vision at the transaction layer.

4.1 Learning Framework

Learning Strategies. AdaChain utilizes a reinforcement learning (RL) approach to provide significant operational benefits: it learns from its mistakes and optimizes long-term rewards through its trials, without requiring a separate training data collection process prior to deployment. This would allow the system to adapt to whatever new hardware, unseen workloads, and novel paradigms at hand. However, inside the RL approach, different problem formulations exist, which also result in different algorithms. For instance, a *contextual multi-armed bandit* (CMAB) problem which assumes episodes are independent of each other can be tackled using Thompson sampling [78]. In contrast, a *full RL* problem where the current action affects the future state requires DQN [43, 59], A3C [58], etc. Interestingly, the independence assumption is closely related to the design of the paradigm-switching mechanism. For example, the CMAB problem used by AdaChain requires pending blocks of each episode to be early aborted, which potentially leads to loss of liveness on a slow client [39] and censorship in blockchains [57].

Moreover, even for the same problem formulation, different types of predictive models could be used. For instance, a *value based model* takes the state (i.e., workload) concatenated with action (i.e., paradigm choice) as input, and outputs the predicted performance. On the other hand, a *policy based model* predicts simultaneously the probability of each action being optimal, requiring minimal or no featurization of the specific action. Therefore, open questions remain on how different problem formulations and RL algorithms behave, especially in the novel context of permissioned blockchains.

Featurization. The ledger (e.g., database log) is naturally decentralized across nodes, and it contains rich information about past transactions. Thus, unless the workload changes at an extremely rapid rate, each learning agent can extract features from its most recent ledger to approximate the incoming workload with minimal overhead. To featurize the system's state, FAB could perform traditional feature engineering similar to AdaChain, e.g., derive the write ratio, skewness of keys, and compute-intensity of the workloads from on-chain data, and use them as input to the predictive model. Alternatively, FAB could also explore *automatic* state extraction: deserialize the ledger to form the conflict graph, where each edge is annotated with the submission and commit timestamp, and use graph convolutional networks [47] to extract the state *automatically*. Further, if using value-based predictive models that take both the state and action as the input in order to predict the performance, FAB needs to featurize the design space of paradigms.

Adversarial machine learning. AdaChain's usage of machine learning introduces a new performance attack vector: feature data might be manipulated to cause the learning agent to always pick a bad paradigm. To carry out this attack, malicious nodes might propose adversarial feature values to misguide the honest learning agents. There are at least two ways such adversarial features could negatively impact performance: (1) *decision attacks* that target the inference phase, where an adversary reports false observations of its own features in order to push the global feature in one direction or another; and (2) *poisoning attacks* that target the training phase, where an adversary reports carefully selected feature values and labels to cause the next trained model to be inaccurate. To address both challenges, FAB could utilize *randomized smoothing* [25, 70], which recently has been demonstrated effective in the ML community. Specifically, instead of using a single reward predictor R, each learning agent utilizes a set of predictor $\{R_i\}$, where the input to each predictor is the original feature x plus a small perturbation δ_i, where δ_i is sampled from certain distribution depending on the norm our defense belongs to. The final prediction result is then voted by $\{R_i\}$. This way, FAB has provably consistent output even when feature x is manipulated by adversary within some ranges.

Uncovering new transaction processing paradigms. Another intriguing research is to figure out whether the learning framework can uncover new effective paradigms that are not previously explored by human experts. For example, FAB can mix and match design attributes, combining Order-Parallel Execute (i.e., OXII [7]) together with reordering and early aborts to generate new paradigms. Specifically, consider three totally ordered transactions and their accessed keys: $T_1(A)$, $T_2(A, B)$, $T_3(B)$. If no reordering happens, these transactions need to be executed sequentially even in OXII, whereas if T_2 is reordered as the first transaction, T_1 and T_3 can be executed in parallel. Thus, such reordering also requires developing algorithms that prune and minimize the depth of a graph while not skewing the transaction distribution (i.e., fairness).

4.2 Switching between Paradigms

Finer-grained adaptation. AdaChain's switching mechanism works at the granularity of episodes, where each episode is a constant number of blocks. While this approach is able to select the optimal paradigm among existing ones during an episode, its coarse-grained operation limits its ability to fully exploit workload characteristics for performance. For example, it only chooses between enabling/disabling the reordering algorithms used by Fabric++ but fails to directly learn the final order of each transaction. As another example, it does not allow a certain transaction to go through the OX pipeline while sending the subsequent one into the XOV pipeline. In FAB, an interesting direction is to explore adaptation *on a per-transaction basis*. At such fine granularity, the featurizer first needs to be redesigned to encode information about each incoming transaction. Second, a policy-based predictive model will be favored over a value-based one, given the exponentially large action space. Finally, a special block cut phase could be added to the adaptation protocol to ensure the learning is consistent for the same transaction across honest peers. FAB should also study how to minimize the adaptation overhead at such fine granularity.

5 ADAPTING BFT PROTOCOL

Another aspect of adaptivity is choosing the best-performing BFT protocol. Previous studies [15, 39] at this layer demonstrate the potential of adaptive BFT protocols in enhancing systems performance under dynamic environments and workloads. Specifically, Abstract [13, 39] proposed a switching framework for BFT protocols, where the system can switch protocols in a predefined order when a certain progress condition is not met, e.g., a client did not receive enough matching replies. Once the original protocol is aborted by clients, the next predefined protocol is invoked. Despite ensuring safety and liveness, such a predefined switching order lacks intelligence and flexibility. This requires careful tuning of switching heuristics, and achieves even worse performance in many common scenarios. Based on the same switching mechanism as Abstract, ADAPT [15] takes one step further, where supervised learning is utilized to decide the next promising protocol, based on the impact factors that its Event System (ES) monitors.

Unfortunately, ADAPT still suffers from several major drawbacks that make it impractical in Byzantine settings. First, it relies on a single trusted replica to collect data, train the machine learning model, and then distribute the model to all other replicas. In a Byzantine environment, such an assumption is not realistic, since an honest/malicious replica is hard to identify. A malicious replica could disrupt the training data and training process, resulting in an inaccurate model and coercing the system to always choose the protocols with lower performance. It could even "equivocate" when distributing pre-trained models, resulting in a violation of liveness. Second, ADAPT requires a cumbersome data collection process prior to deployment, and assumes training data is complete. Consequently, ADAPT is unable to adapt to unseen hardware or workloads. Third, ADAPT is not aware of failures or adversarial behaviors in the system, and its featurizer (i.e., the Event System) is marked as future work. Last but not least, ADAPT only covers a small set of BFT protocols that were proposed more than a decade ago. Below, we describe our initial solutions as well as future research agenda in FAB to address the limitations above.

5.1 Learning Framework

Problem formulation. Similar to the transaction management layer, FAB formulates the selection of the best performing BFT protocol as a reinforcement learning problem, whose objective function is maximizing the accumulated reward (i.e., certain user-defined performance metrics) over time. This brings significant operational benefits to FAB and is capable of solving trust issues.

Specifically, the *state* space in the RL problem consists of four categories of factors: workloads, faults, hardware, and system configuration. Workloads are impacted by clients and the specific content of their requests, whereas faults can be caused by either network partition, crashed replicas, dishonest replicas, or even dishonest clients. Hardware-level factors include network latency and bandwidth, as well as machine-level setups like CPU frequency and the number of cores. System-level configurations include the number of replicas and their geo-distribution. The interaction of these four categories of factors constitutes a large state space, which renders manual crafting of heuristics extremely hard, if not impossible.

The *action* space includes the choice of candidate protocol, batch size, as well as the length of the next episode (i.e., the length k of backup instance in Abstract). In FAB, all candidate protocols are of $n = 3f + 1$ network size, including PBFT [20], Zyzzyva [48], RePBFT [31], Prime [6], HotStuff [85], PoE [41], SBFT [40], and Kauri [60], which embody a wide range of design principles. For instance, optimistic protocols (e.g., Zyzzyva, PoE, SBFT, and Kauri) achieve outstanding performance when the system is fault-free, but could suffer even under benign crash failures. One can transition to pessimistic protocols (e.g., PBFT and HotStuff), when an optimistic assumption is violated. On the contrary, robust protocols (e.g., Prime) perform well even under performance attacks, such as pre-prepare delay and timeout manipulation, but have limited performance in normal case operations. Moreover, each of the candidate protocols has a distinct communication pattern, resulting in different message complexity and number of phases.

As an initial design, the consensus layer in FAB still proceeds episode by episode, where each episode is marked by k *committed* requests. During episode n, each replica i exchanges their locally observed state s^i_{n+1} in order to obtain an agreed global state s_{n+1} for the next episode, and the reward r_n for the current episode will be collected as well. Once agreement is reached, the (s_n, a_n, r_n) triplet will be added to RL's experience buffer for future retraining, and each replica's deterministic learning agent (i.e., with the same random seed) would decide the promising protocol a_{n+1} to be invoked in the next episode. When episode n finishes, the switching subroutine will be called, and episode $n+1$ begins. Note that the learning overhead is masked in FAB, since the learning agent is invoked during the episode n when protocol a_n keeps processing requests without being interrupted.

Robust online data collection. When achieving adaptivity at the protocol layer, it is also imperative to ensure both safety and liveness. While FAB's safety is guaranteed by the composability theorem of Abstract [13] and the safety of each candidate protocol, its liveness relies on the data collection and consequent decision-making process. Further, the data collected needs to be robust, preventing malicious replicas from misguiding honest learning agents to derive poor decisions consistently. Thus, FAB proposes the following properties that its data collection mechanism needs to guarantee: (1) consistency – the state s_n is identical across all replicas; (2) robustness – the collected data s_n, r_n are within a reasonable range from what an honest node reports.

To guarantee *consistency*, a straightforward solution is to run two BFT protocols in parallel, i.e., one for ordering actual client requests, namely a_n, and the other for data collection, denoted by dc. During episode n, each replica reports its locally observed state and reward to the leader of dc, once a certain number of requests have been executed in n. After the leader gathers a quorum certificate qc of reports, dc guarantees each replica receives the same qc using consensus. Thus, the state s_n would be identical across all replicas, once certain deterministic policy p is used to obtain s_n from qc.

In a Byzantine network, since at most f replicas may refuse to report their local observations, an honest leader of dc is guaranteed to receive reports from no less than $2f+1$ replicas, whereas a dishonest leader will be replaced by dc itself. Inside a collected qc of size $2f+1$, since at most f arbitrary values could exist due to Byzantine reporters, the median value of qc (i.e., represented by policy p) can be easily proved to fall between the minimum and maximum values reported by honest replicas. Therefore, *robustness* is guaranteed. While the above solution is effective in most cases, a malicious leader of a_n could still violate robustness by issuing an in-dark attack. Specifically, in-dark attacks are common in leader-based protocols where the malicious leader refuses to send proposals to as many as f honest backups. This would prevent those honest replicas from being involved in the consensus a_n, and thus they would report zero or meaningless local observations to the leader of dc. The median value of a qc of size $2f+1$ would no longer be robust, if f of them are zero and f of them are arbitrary. FAB addresses such issue as described below: whenever an honest replica detects itself to be in-dark attacked, it will refuse to report its own features to the leader of dc, where there is a timer for proposing qc; therefore, if a replica in dc commits a qc whose size is less than $2f+1$, it will *complain* to a_n and initiate a view change as suspecting the leader of a_n to be faulty, and a_{n+1} will remain the same as a_n. Since view change routines can guarantee an honest leader to be chosen in no more than f rounds of view changes, FAB can thus ensure that the problem can be solved within f episodes.

Featurizing faults and protocols. Having discussed how to collect data, the next challenge at the protocol layer is to figure out what data to collect, i.e., featurization of system state. Among the four categories of factors in the state space, workloads can be featurized similar to the transaction management layer. Hardware and system configurations are fairly static during the FAB deployment, and therefore do not need explicit featurization. Thus, the challenge is about how to featurize faulty behaviors in the system.

To address this challenge, FAB replies on the following insight: instead of designing a Byzantine fault detector and characterizing the fault itself, it is more important to identify *how the candidate protocols are affected by the faults*. Specifically, FAB captures the impact of faults using two features: (1) slow path probability; (2) timer elapsed values. Modern BFT protocols often utilize dual-path designs, where a fast path is used for handling fault-free conditions, while an expensive slower path is triggered when fault/attacks occur. Thus, the slow path probability r_{slow} can be derived by measuring the percentages (probability) of requests committed in the slow path during a certain window. The fast path probability is then derived using $1 - r_{slow}$. For non-optimistic protocols, e.g., PBFT and Prime, all requests are considered to be in the slow path. Timer elapsed values represent the gap between timers are initiated and just before they are reset. This feature contains rich information about timing-based attacks [6, 24, 65], where a malicious leader deliberately delays requests without triggering view change.

Besides attacks, for value-based RL models [43, 59], it is beneficial to featurize the action space (i.e., candidate BFT protocols). FAB does so using four dimensions, and each can be represented by a categorical variable: (1) communication topology – clique (n to n), star (n to 1 to n), or tree (only between parent and children); (2) processing strategy – optimistic, pessimistic, or robust; (3) the number of phases and (4) leader rotation – true or false.

Although FAB adopts the Byzantine failure model to handle non-trustworthiness, we expect the same learning algorithms and featurization to be effective for crash fault-tolerant (CFT) protocols as well, since the dimensions in CFT protocols' design space could be seen as a subset of dimensions in BFT protocols' design space. For example, similar to optimistic dual-path protocols in the context of BFT (e.g., Zyzzyva, SBFT), a CFT protocol can reach consensus in one round optimistically assuming all nodes are non-faulty, and if the assumption does not hold, it runs consensus in two rounds by switching to its slow path [23].

Discovery of novel protocols. Similar to the transaction management layer, an interesting direction is to discover new BFT protocols that fit new environments or meet new application requirements through the learning framework. This requires fine-grained actions, not at the level of choosing among different existing BFT protocols, but rather to take actions to change attributes (i.e., dimensions) within the protocol design space, e.g., changing a processing strategy or modifying the number of phases required for consensus. Although automated Byzantine attack generators (e.g., Twins [16]) can be used, open questions remain on how to systematically ensure and prove the correctness of the newly discovered protocols.

5.2 Switching between BFT Protocols

Optimizing protocol switching. FAB's switching mechanism can also be implemented over Abstract [13], where an episode of FAB is equivalent to a *Backup* instance in Abstract, utilizing its established proof system. Specifically, in terms of *safety*, Abstract's idempotency theorem specifies that if individual BFT instances are correct, irrespectively of each other, then the system composed through switching is also correct. In terms of *liveness*, Abstract guarantees liveness if a request is not aborted by all instances.

Unlike Abstract, each episode in FAB is designed to run on the same cluster of machines. Thus, its switching mechanism could be further optimized. First, instead of relying on the client to panic, each replica can decide whether to switch or not. Once executing k requests, each replica multicasts an init history of executed requests (i.e., checkpoint) to other replicas. Second, an honest replica does not need to wait for $f+1$ matching init history and execute it before starting the new episode, since the init history is already reflected in its local service state. Once k requests are committed in episode n and the BFT protocol for $n+1$ is derived by the learning agent, episode $n+1$ is invoked. In other words, the protocol switching can be asynchronous inside FAB with low overhead.

6 CROSS-LAYER ADAPTATION

Optimizing blockchain software layers alone without consideration of the underlying hardware infrastructure will hit a performance wall. In this section, we discuss how this wall can be surmounted through cross-layer optimizations.

Identifying performance bottlenecks. One crucial step in supporting cross-layer adaptation is to identify performance bottlenecks in the end-to-end system, so as to avoid unnecessary configuration switching or resource over-provisioning. For example, when FAB is under timing attack, the BFT protocol likely becomes the bottleneck, even when the best-performing protocol is chosen (e.g., Prime [6]). Under such conditions, adjusting other layers does not improve the performance, but only incurs additional switching overhead or waste of resources [81]. On the contrary, under a fault-free condition, blockchains are usually bottlenecked by an inefficient transaction processing paradigm or inadequate hardware resources, which should be adjusted via the learning agent. Adaptation should focus on the bottleneck layers until the best action is chosen. Identifying the bottleneck in FAB at run-time is challenging since it depends on the system state (e.g., client workloads, faulty conditions, and hardware) as well as the currently selected configurations at each layer, and hence shifts over time.

Disaggregation or not? Our prior work FlexChain [81] proposed a disaggregated BaaS infrastructure for blockchains with an XOV-style paradigm, demonstrating efficient resource utilization through independent scaling of different resource types. The elasticity of disaggregated data center (DDC) infrastructure [38, 73, 88, 89] also improves end-to-end performance when cloud resources (either compute, memory or storage) are the bottleneck of BaaS. However, DDC incurs at least 12.8% overhead in using remote memory. Thus, when the workload (or other factor in the state space) shifts and hardware resources are no longer the bottleneck, it is beneficial to use a non-DDC traditional setup. FAB envisions a hybrid cloud deployment where DDC and non-DDC hardware can co-exist, and it should seamlessly determine which infrastructure should be adopted. A promising solution is to first view the blockchain ledger as multi-channel time series data, and forecast the workload changes that should lead to a transition in infrastructure.

Unlike paradigm or BFT protocol switching, this infrastructure-level transition has higher migration costs, e.g., copying remote memory into main memory. The overhead of switching from DDC to non-DDC infrastructure can be first modeled according to the transition protocol, and vice versa. If the overhead of the transition is less than the overhead of disaggregation, the transition can be carried out. Next, incremental transition algorithms for switching between a DDC and non-DDC setup will be designed. Such an incremental approach is made possible by the abstraction of running FAB as a collection of virtual peers, i.e., some peers can be disaggregated while others are not.

BFT compatibility. A BFT protocol can only be used in a subset of paradigms, depending on the protocol's underlying assumptions. For example, the Zyzzyva protocol [48] is incompatible with the XOV-style paradigm [11, 35, 71, 74] since Zyzzyva requires clients to actively participate in the protocol in order to detect failures and change the leader. Thus, FAB needs to identify the permissible BFT protocol and paradigm pairs based on their design principles. Since the number of protocols and paradigms is not fixed (i.e., new ones keep emerging), it is interesting to study whether FAB can learn the permissible combinations on the fly after some exploration without human interpretation. Finally, FAB should extend the episode-based switching mechanism to coordinate simultaneous paradigm and BFT protocol switching in an end-to-end system.

7 CONCLUSION

This paper outlines our vision for adaptivity in untrustworthy distributed databases. Being the most notable instance, we mainly focus on permissioned blockchains and demonstrate that a one-size-fits-all architecture is not future-proof as novel smart contracts, modern hardware, and new Byzantine consensus protocols keep emerging. To address the adaptivity challenge, we lay out a vision of FAB that advocates machine learning at run-time across all layers of the untrustworthy distributed databases stack.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful feedback and suggestions. This work is funded by NSF grants CNS-2104882 and CNS-2107147.

REFERENCES

[1] [n. d.]. AWS. Amazon quantum ledger database (QLDB). https://aws.amazon.com/qldb/.

[2] [n. d.]. Blockchain on AWS Enterprise blockchain made real. https://aws.amazon.com/blockchain/.

[3] [n. d.]. IBM Blockchain Platform. https://www.ibm.com/cloud/blockchain-platform.

[4] [n. d.]. Oracle Blockchain. https://www.oracle.com/blockchain/.

[5] Michael Abd-El-Malek, Gregory R Ganger, Garth R Goodson, Michael K Reiter, and Jay J Wylie. 2005. Fault-scalable Byzantine fault-tolerant services. *Operating Systems Review (OSR)* 39, 5 (2005), 59–74.

[6] Yair Amir, Brian Coan, Jonathan Kirsch, and John Lane. 2011. Prime: Byzantine replication under attack. *Transactions on Dependable and Secure Computing* 8, 4 (2011), 564–577.

[7] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. 2019. ParBlockchain: Leveraging Transaction Parallelism in Permissioned Blockchain Systems. In *Int. Conf. on Distributed Computing Systems (ICDCS)*. IEEE, 1337–1347.

[8] Mohammad Javad Amiri, Joris Duguépéroux, Tristan Allard, Divyakant Agrawal, and Amr El Abbadi. 2021. SEPAR: Towards Regulating Future of Work Multi-Platform Crowdworking Environments with Privacy Guarantees. In *Proceedings of The Web Conf. (WWW)*. 1891–1903.

[9] Mohammad Javad Amiri, Boon Thau Loo, Divyakant Agrawal, and Amr El Abbadi. 2022. Qanaat: A Scalable Multi-Enterprise Permissioned Blockchain System with Confidentiality Guarantees. *Proc. of the VLDB Endowment* 15, 11 (2022), 2839–2852.

[10] Mohammad Javad Amiri, Chenyuan Wu, Divyakant Agrawal, Amr El Abbadi, Boon Thau Loo, and Mohammad Sadoghi. 2024. The Bedrock of Byzantine Fault Tolerance: A Unified Platform for BFT Protocol Design and Implementation. In *Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association.

[11] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, and Yacov Manevich. 2018. Hyperledger Fabric: a distributed operating system for permissioned blockchains. In *European Conf. on Computer Systems (EuroSys)*. ACM, 30:1–30:15.

[12] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. [n. d.]. A Brief Survey of Deep Reinforcement Learning. 34, 6 ([n. d.]), 26–38. https://doi.org/10.1109/MSP.2017.2743240 arXiv:1708.05866

[13] Pierre-Louis Aublin, Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. 2015. The next 700 BFT protocols. *Transactions on Computer Systems (TOCS)* 32, 4 (2015), 12.

[14] Jean-Paul Bahsoun, Rachid Guerraoui, and Ali Shoker. [n. d.]. Making BFT Protocols Really Adaptive. In *2015 IEEE International Parallel and Distributed Processing Symposium* (2015-05) *(IPDPS '15)*. 904–913. https://doi.org/10.1109/IPDPS.2015.21 ISSN: 1530-2075.

[15] Jean-Paul Bahsoun, Rachid Guerraoui, and Ali Shoker. 2015. Making BFT protocols really adaptive. In *Int. Parallel and Distributed Processing Symposium*. IEEE, 904–913.

[16] Shehar Bano, Alberto Sonnino, Andrey Chursin, Dmitri Perelman, Zekun Li, Avery Ching, and Dahlia Malkhi. 2022. Twins: Bft systems made robust. In *Int. Conf. on Principles of Distributed Systems (OPODIS)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[17] F. Brasileiro, F. Greve, A. Mostéfaoui, and M. Raynal. 2001. Consensus in one communication step. In *Int. Conf. on Parallel Computing Technologies (PaCT)*. Springer, 42–50.

[18] Nathan Bronson, Zach Amsden, George Cabrera, Prasad Chakka, Peter Dimov, Hui Ding, Jack Ferris, Anthony Giardullo, Sachin Kulkarni, and Harry Li. 2013. TAO: Facebook's Distributed Data Store for the Social Graph. In *Annual Technical Conf. (ATC)*. USENIX Association, 49–60.

[19] Lucien Bruggeman and Sasha Pezenik. 2022. Emergent BioSolutions discarded ingredients for 400 million COVID-19 vaccines, probe finds. https://abcnews.go.com/US/emergent-biosolutions-discarded-ingredients-400-million-covid-19/story?id=84604285.

[20] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine fault tolerance. In *Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association, 173–186.

[21] Lujing Cen, Ryan Marcus, Hongzi Mao, Justin Gottschlich, Mohammad Alizadeh, and Tim Kraska. [n. d.]. Learned Garbage Collection. In *Proceedings of the 4th ACM SIGPLAN International Workshop on Machine Learning and Programming Languages* (2020) *(MAPL @ PLDI '20)*. ACM. https://doi.org/10.1145/3394450.3397469

[22] Jeeta Ann Chacko, Ruben Mayer, and Hans-Arno Jacobsen. 2021. Why Do My Blockchain Transactions Fail? A Study of Hyperledger Fabric. In *SIGMOD Int. Conf. on Management of Data*. ACM, 221–234.

[23] Bernadette Charron-Bost and André Schiper. 2006. Improving fast paxos: being optimistic with no overhead. In *Pacific Rim Int. Symposium on Dependable Computing (PRDC)*. 287–295.

[24] Allen Clement, Edmund L Wong, Lorenzo Alvisi, Michael Dahlin, and Mirco Marchetti. 2009. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults.. In *Symposium on Networked Systems Design and Implementation (NSDI)*, Vol. 9. USENIX Association, 153–168.

[25] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *Int. Conf. on Machine Learning (ICML)*. PMLR, 1310–1320.

[26] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, and Peter Hochschild. 2013. Spanner: Google's globally distributed database. *Transactions on Computer Systems (TOCS)* 31, 3 (2013), 8.

[27] CortexFoundation. 2020. Cortex Overview. https://github.com/CortexFoundation/tech-doc/blob/master/cortex-details.md.

[28] Sam Daley. 2021. 18 Blockchain-as-a-Service Companies Making the DLT More Accessible. https://builtin.com/blockchain/blockchain-as-a-service-companies.

[29] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: Amazon's highly available key-value store. *Operating Systems Review (OSR)* 41, 6 (2007), 205–220.

[30] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. 2017. Blockbench: A framework for analyzing private blockchains. In *SIGMOD Int. Conf. on Management of Data*. ACM, 1085–1100.

[31] Tobias Distler, Christian Cachin, and Rüdiger Kapitza. 2016. Resource-efficient Byzantine fault tolerance. *Transactions on Computers* 65, 9 (2016), 2807–2819.

[32] Liming Fang, Bo Zhao, Yang Li, Zhe Liu, Chunpeng Ge, and Weizhi Meng. 2020. Countermeasure based on smart contracts and AI against DoS/DDoS attack in 5G circumstances. *IEEE Network* 34, 6 (2020), 54–61.

[33] Peter X Gao, Akshay Narayan, Sagar Karandikar, Joao Carreira, Sangjin Han, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. 2016. Network requirements for resource disaggregation. In *symposium on operating systems design and implementation (OSDI)*. USENIX Association, 249–264.

[34] Zerui Ge, Dumitrel Loghin, Beng Chin Ooi, Pingcheng Ruan, and Tianwen Wang. 2022. Hybrid blockchain database systems: design and performance. *Proceedings of the VLDB Endowment* 15, 5 (2022), 1092–1104.

[35] Christian Gorenflo, Lukasz Golab, and Srinivasan Keshav. 2020. XOX Fabric: A hybrid approach to transaction execution. In *Int. Conf. on Blockchain and Cryptocurrency (ICBC)*. IEEE, 1–9.

[36] Justin Gottschlich, Armando Solar-Lezama, Nesime Tatbul, Michael Carbin, Martin Rinard, Regina Barzilay, Saman Amarasinghe, Joshua B. Tenenbaum, and Tim Mattson. [n. d.]. The three pillars of machine programming. In *Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages* (Philadelphia, PA, USA, 2018-06-18) *(MAPL 2018)*. Association for Computing Machinery, 69–80. https://doi.org/10.1145/3211346.3211355

[37] Vincent Gramoli, Rachid Guerraoui, Andrei Lebedev, Chris Natoli, and Gauthier Voron. 2023. Diablo: A Benchmark Suite for Blockchains. In *European Conf. on Computer Systems (EuroSys)*. ACM.

[38] Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G Shin. 2017. Efficient memory disaggregation with infiniswap. In *Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 649–667.

[39] Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. 2010. The next 700 BFT protocols. In *European conf. on Computer systems (EuroSys)*. ACM, 363–376.

[40] Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. 2019. SBFT: a Scalable Decentralized Trust Infrastructure for Blockchains. In *Int. Conf. on Dependable Systems and Networks (DSN)*. IEEE/IFIP, 568–580.

[41] Suyash Gupta, Jelle Hellings, Sajjad Rahnama, and Mohammad Sadoghi. 2021. Proof-of-execution: Reaching consensus through fault-tolerant speculation. In *Int. Conf. on Extending Database Technology (EDBT)*. 301–312.

[42] Niranjan Hasabnis and Justin Gottschlich. [n. d.]. ControlFlag: a self-supervised idiosyncratic pattern detection system for software control structures. In *Proceedings of the 5th ACM SIGPLAN International Symposium on Machine Programming* (New York, NY, USA, 2021-06-20) *(MAPS '21)*. Association for Computing Machinery, 32–42. https://doi.org/10.1145/3460945.3464954

[43] Hado van Hasselt, Arthur Guez, and David Silver. [n. d.]. Deep Reinforcement Learning with Double Q-Learning. In *Thirtieth AAAI Conference on Artificial Intelligence* (2016-03-02) *(AAAI '16)*. https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389

[44] Zsolt István, Alessandro Sorniotti, and Marko Vukolić. 2018. StreamChain: Do Blockchains Need Blocks?. In *Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers (SERIAL)*. ACM, 1–6.

[45] Rüdiger Kapitza, Johannes Behl, Christian Cachin, Tobias Distler, Simon Kuhnle, Seyed Vahid Mohammadi, Wolfgang Schröder-Preikschat, and Klaus Stengel. 2012. CheapBFT: resource-efficient byzantine fault tolerance. In *European Conf. on Computer Systems (EuroSys)*. ACM, 295–308.

[46] Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. 2022. Themis: Fast, Strong Order-Fairness in Byzantine Consensus. *The Science*

of Blockchain Conf. (SBC) (2022).

[47] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[48] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. 2007. Zyzzyva: speculative byzantine fault tolerance. *Operating Systems Review (OSR)* 41, 6 (2007), 45–58.

[49] Jae Kwon. 2014. Tendermint: Consensus without mining. (2014).

[50] Leslie Lamport. 1978. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21, 7 (1978), 558–565.

[51] Leslie Lamport. 2001. Paxos made simple. *ACM Sigact News* 32, 4 (2001), 18–25.

[52] Leslie Lamport. 2006. Fast paxos. *Distributed Computing* 19, 2 (2006), 79–103.

[53] Leslie Lamport and Mike Massa. 2004. Cheap paxos. In *Int. Conf. on Dependable Systems and Networks (DSN)*. IEEE, 307–314.

[54] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. [n. d.]. Bao: Making Learned Query Optimization Practical. In *Proceedings of the 2021 International Conference on Management of Data* (China, 2021-06) *(SIGMOD '21)*. https://doi.org/10.1145/3448016.3452838 Award: 'best paper award'.

[55] Ryan Marcus and Olga Papaemmanouil. [n. d.]. Releasing Cloud Databases from the Chains of Performance Prediction Models. In *8th Biennial Conference on Innovative Data Systems Research* (San Jose, CA, 2017) *(CIDR '17)*. tex.authors= Ryan Marcus and Olga Papaemmanouil.

[56] J-P Martin and Lorenzo Alvisi. 2006. Fast byzantine consensus. *Transactions on Dependable and Secure Computing* 3, 3 (2006), 202–215.

[57] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. 2016. The honey badger of BFT protocols. In *Conf. on Computer and Communications Security (CCS)*. ACM, 31–42.

[58] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Int. Conf. on Machine Learning (ICML)*. PMLR, 1928–1937.

[59] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. [n. d.]. Human-level control through deep reinforcement learning. 518, 7540 ([n. d.]), 529–533.

[60] Ray Neiheiser, Miguel Matos, and Luís Rodrigues. 2021. Kauri: Scalable BFT Consensus with Pipelined Tree-Based Dissemination and Aggregation. In *Symposium on Operating Systems Principles (SOSP)*. ACM, 35–48.

[61] Diego Ongaro and John K Ousterhout. 2014. In search of an understandable consensus algorithm. In *Annual Technical Conf. (ATC)*. USENIX Association, 305–319.

[62] Dan Patterson. 2021. Hackers are attacking the COVID-19 vaccine supply chain. https://www.cbsnews.com/news/covid-19-vaccine-hackers-supply-chain/.

[63] Zhe Peng, Cheng Xu, Haixin Wang, Jinbin Huang, Jianliang Xu, and Xiaowen Chu. 2021. P2B-Trace: Privacy-Preserving Blockchain-based Contact Tracing to Combat Pandemics. In *SIGMOD Int. Conf. on Management of Data*. ACM, 2389–2393.

[64] Zhe Peng, Jianliang Xu, Xiaowen Chu, Shang Gao, Yuan Yao, Rong Gu, and Yuzhe Tang. 2021. Vfchain: Enabling verifiable and auditable federated learning via blockchain systems. *IEEE Transactions on Network Science and Engineering* (2021).

[65] Marco Platania, Daniel Obenshain, Thomas Tantillo, Yair Amir, and Neeraj Suri. 2016. On choosing server-or client-side solutions for BFT. *ACM Computing Surveys (CSUR)* 48, 4 (2016), 1–30.

[66] World Food Programme. 2017. Blockchain Against Hunger: Harnessing Technology In Support Of Syrian Refugees. https://www.wfp.org/news/blockchain-against-hunger-harnessing-technology-support-syrian-refugees.

[67] Ji Qi, Xusheng Chen, Yunpeng Jiang, Jianyu Jiang, Tianxiang Shen, Shixiong Zhao, Sen Wang, Gong Zhang, Li Chen, Man Ho Au, et al. 2021. Bidl: A High-throughput, Low-latency Permissioned Blockchain Framework for Datacenter Networks. In *Symposium on Operating Systems Principles (SOSP)*. ACM, 18–34.

[68] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédo Durand, and Saman Amarasinghe. [n. d.]. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. 48, 6 ([n. d.]), 519–530. https://doi.org/10.1145/2499370.2462176

[69] Steve Reilly, Jason Paladino, Jonathan Lambert, and Matt Stiles. 2022. Fake vaccine cards are everywhere. It's a public health nightmare. https://www.grid.news/story/science/2022/01/25/fake-vaccine-cards-are-everywhere-its-a-public-health-nightmare/.

[70] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. 2020. Certified robustness to label-flipping attacks via randomized smoothing. In *Int. Conf. on Machine Learning (ICML)*. PMLR, 8230–8241.

[71] Pingcheng Ruan, Dumitrel Loghin, Quang-Trung Ta, Meihui Zhang, Gang Chen, and Beng Chin Ooi. 2020. A Transactional Perspective on Execute-order-validate Blockchains. In *SIGMOD Int. Conf. on Management of Data*. ACM, 543–557.

[72] Fred B Schneider. 1990. Implementing fault-tolerant services using the state machine approach: A tutorial. *Computing Surveys (CSUR)* 22, 4 (1990), 299–319.

[73] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. 2018. {LegoOS}: A Disseminated, Distributed {OS} for Hardware Resource Disaggregation. In *Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association, 69–87.

[74] Ankur Sharma, Felix Martin Schuhknecht, Divya Agrawal, and Jens Dittrich. 2019. Blurring the lines between blockchains and database systems: the case of hyperledger fabric. In *SIGMOD Int. Conf. on Management of Data*. ACM, 105–122.

[75] Atul Singh, Tathagata Das, Petros Maniatis, Peter Druschel, and Timothy Roscoe. 2008. BFT Protocols Under Fire.. In *Symposium on Networked Systems Design and Implementation (NSDI)*, Vol. 8. USENIX Association, 189–204.

[76] Judy Stone. 2021. How Counterfeit Covid-19 Vaccines And Vaccination Cards Endanger Us All. https://www.forbes.com/sites/judystone/2021/03/31/how-counterfeit-covid-19-vaccines-and-vaccination-cards-endanger-us-all/?sh=eaddb0e36495.

[77] Richard S. Sutton and Andrew G. Barto. [n. d.]. *Introduction to Reinforcement Learning* (1st ed.). MIT Press.

[78] William R. Thompson. [n. d.]. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. ([n. d.]).

[79] Feng Tian. 2017. A supply chain traceability system for food safety based on HACCP, blockchain & Internet of things. In *Int. Conf. on service systems and service management (ICSSSM)*. IEEE, 1–6.

[80] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. [n. d.]. Automatic Database Management System Tuning Through Large-scale Machine Learning. In *Proceedings of the 2017 ACM International Conference on Management of Data* (New York, NY, USA, 2017) *(SIGMOD '17)*. ACM, 1009–1024. https://doi.org/10.1145/3035918.3064029

[81] Chenyuan Wu, Mohammad Javad Amiri, Jared Asch, Heena Nagda, Qizhen Zhang, and Boon Thau Loo. 2022. FlexChain: An Elastic Disaggregated Blockchain. *Proc. of the VLDB Endowment* 16, 01 (2022), 23–36.

[82] Chenyuan Wu, Bhavana Mehta, Mohammad Javad Amiri, Ryan Marcus, and Boon Thau Loo. 2023. AdaChain: A Learned Adaptive Blockchain. *Proc. of the VLDB Endowment* 16, 8 (2023), 2033–2046.

[83] Xinying Yang, Yuan Zhang, Sheng Wang, Benquan Yu, Feifei Li, Yize Li, and Wenyuan Yan. 2020. LedgerDB: a centralized ledger database for universal audit and verification. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3138–3151.

[84] Zongheng Yang, Wei-Lin Chiang, Sifei Luan, Gautam Mittal, Michael Luo, and Ion Stoica. [n. d.]. Balsa: Learning a Query Optimizer Without Expert Demonstrations. In *Proceedings of the 2022 International Conference on Management of Data* (New York, NY, USA, 2022-06-10) *(SIGMOD '22)*. Association for Computing Machinery, 931–944. https://doi.org/10.1145/3514221.3517885

[85] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. 2019. HotStuff: BFT consensus with linearity and responsiveness. In *Symposium on Principles of Distributed Computing (PODC)*. ACM, 347–356.

[86] Raul Zambrano, Andrew Young, and Stefaan Verhulst. 2018. Connecting refugees to aid through blockchain-enabled ID management: world food programme's building blocks. *GovLab October* (2018).

[87] Qizhen Zhang, Yifan Cai, Sebastian Angel, Ang Chen, Vincent Liu, and Boon Thau Loo. 2020. Rethinking data management systems for disaggregated data centers. In *Conf. on Innovative Data Systems Research (CIDR)*.

[88] Qizhen Zhang, Yifan Cai, Xinyi Chen, Sebastian Angel, Ang Chen, Vincent Liu, and Boon Thau Loo. 2020. Understanding the effect of data center resource disaggregation on production DBMSs. *Proceedings of the VLDB Endowment* 13, 9 (2020).

[89] Qizhen Zhang, Xinyi Chen, Sidharth Sankhe, Zhilei Zheng, Ke Zhong, Sebastian Angel, Ang Chen, Vincent Liu, and Boon Thau Loo. 2022. Optimizing data-intensive systems in disaggregated data centers with teleport. In *Int. Conf. on Management of Data*. 1345–1359.