SAMSUNG Best Paper Award 2019

Multi-batch Scheduling for Improving Performance of Hyperledger Fabric based IoT Applications

Hyojung Lee, Changsuk Yoon, Kyusang Lee, Sangji Bae,
Sangwon Lee, Sangjun Kang, Kiwoon Sung, and Seungjai Min

(Abstract) Blockchain technology has been regarded as a paradigm that enables enterprises to deploy decentralized Internet of things (IoT) applications, which require security, mutual truthfulness, privacy, and data reliability. However, the adoption of blockchain technology on IoT applications has been limited due to the lack of blockchain's capability to process a huge amount of transaction requests from IoT devices. In this paper, we propose a new transaction processing mechanism, called *Accelerator*, which improves the performance of blockchain-based IoT applications in terms of transaction throughput and latency. Two key design ideas of Accelerator lie in: (i) independent modular structure that can be implemented without changing the incumbent blockchain network and (ii) adaptive multi-batch scheduling engine for robust performance. We first provide a mathematical analysis that is used to choose the parameters of Accelerator to maximize the performance. Moreover, we implement Accelerator on a decentralized blockchain network using Hyperledger Fabric and perform the extensive test using Hyperledger Caliper. We show that Accelerator achieves 8x transaction throughput improvement without sacrificing latency.

1. Introduction

1.1. Motivation

With the widespread adoption of blockchain technology, which comes from Bitcoin's architecture [1], a number of decentralized applications based on blockchain have arisen. One of the promising usage of blockchain is to adopt Internet of Things (IoT), where the IoT applications require to guarantee security, mutual truthfulness, privacy, and data reliability without a traditional centralizer. The viability of such blockchain-based IoT applications has been actively investigated in recent years, as described in [2, 3, 4, 5, 6, 7, 8]. However, the lack of blockchain's capability caused by time-consuming consensus process still remains as a limitation of the adoption of blockchain on IoT applications. Moreover, IoT applications have various patterns on generating requests, where some applications might request over thousands of transactions per second (TPS).

The main interest of this paper lies in providing an adaptive structure to enhance the performance of blockchain-based IoT applications. After Bitcoin, a number of new blockchain technologies are emerged [9, 10, 11], aiming at improving transaction throughput by redesigning the consensus algorithm. However, the consensus algorithms have inherent structural challenges in achieving high transaction throughput, since it should fulfill blockchain's promise such as security and immutability. In this paper, we design a new transaction processing engine, called *Accelerator*, which highly improves the transaction throughput of blockchain without any change of the blockchain's consensus algorithm. To this end, we develop the adaptive multi-batch scheduling engine that classifies and aggregates the submitted transactions as batched transactions.

1.2. Main Contribution

As the first step of our approach, we adapt Accelerator to the Hyperledger Fabric [11], which is one of popular permissioned blockchain technology and being studied in depth [12, 13, 14]. To take full advantage of Hyperledger Fabric, two key design features of Accelerator are summarized in what follows:

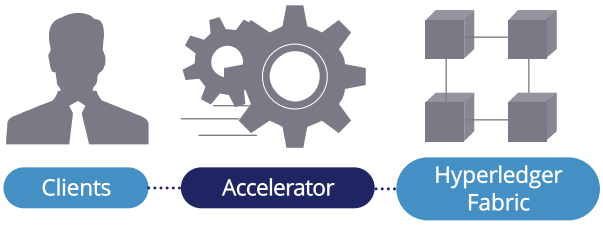


Fig. 1. Overall structure of Accelerator

(a) ***Independent and modular structure.*** Accelerator has an independent and modular structure, thus is implemented as a form of intermediary running between the clients and the blockchain network as depicted in Fig. 1. Such a feature is suitable for adopting IoT applications, since IoT devices may not have enough computing power to run an additional process for improving blockchain performance.

(b) ***Adaptive multi-batch scheduling engine.*** Accelerator provides a simple, but powerful transaction processing algorithm inspired by batch scheduling. Accelerator classifies the incoming transactions into the transaction processing policy of Fabric and subsequently makes a batched transaction. To this end, Accelerator is carefully designed to choose the adaptive batch size, depending on the characteristic of requested transactions and remaining computational resource of blockchain network.

We first provide a rigorous mathematical analysis to choose the parameters of Accelerator. Subsequently, an extensive performance evaluation is conducted to understand the impact of Accelerator on the performance on Fabric network. Moreover, we use Hyperledger Caliper [15], which is a test harness

All authors are working in Blockchain Research Lab, Samsung SDS R&D Center, 56 Seongchon-gil, Seocho-gu, Seoul, South Korea

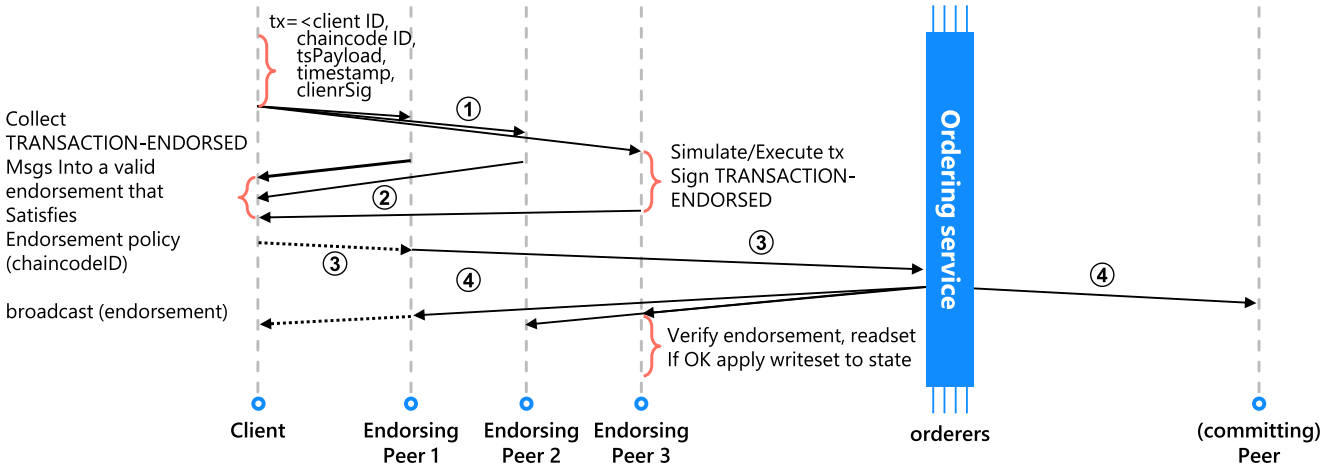


Fig. 2. Transaction flow of Hyperledger Fabric

provided by Hyperledger, applying some modifications in order to make proper simulation scenarios for IoT applications. Finally, we conclude that Accelerator enables Hyperledger Fabric based IoT applications to finalize more than 12,000 transactions per second, which achieves 8x improvement compared to the pure Hyperledger Fabric network.

1.3. Related Work

In this paper, we consider a blockchain network based on Hyperledger Fabric, which is suitable to show the performance improvement by Accelerator due to a modular consensus process. The related work on the performance of Hyperledger Fabric includes [12, 13, 14]. The authors in [12] study how the configuration parameters impact on its performance and find out where the bottleneck is. The paper [13] also contains the results from adjusting parameters of Fabric and further optimizing on the data structure for efficient usage of hash table. The authors [14] suggest a performance model using stochastic reward nets, and provide performance evaluation results of Fabric. Our work also studies and improves the performance of Fabric, however, we do not change any part of Fabric but provide an additional independent module, Accelerator, which is inspired by batch scheduling. Accelerator has a pluggable physical structure as well as is able to choose adaptive parameters according to the remaining computing resource of participants on Fabric.

The related work on blockchain-based IoT applications includes [2, 3, 4, 5, 6, 7, 8, 16, 17]. The papers in [2, 3, 4] introduce the various usage of blockchain technology on IoT as well as the challenges and the opportunities. In [5], the authors implement Ethereum-based network among IoT devices and show how to manage them using smart contract. The papers [6, 7, 8] introduce a design of IoT networks based on Hyperledger Fabric. The authors [6] demonstrate the applicability of Fabric to IoT applications and data management that aims at providing end-to-end trustiness. The paper [7] and [8] propose hierarchical architectures for securing sensor data acquisition and edge computing, respectively. But, these works differ from ours in that they focus on how to implement a blockchain-based IoT application rather than its performance. To the best of our knowledge, this paper is the first that studies the performance improvement of blockchain-based IoT applications.

Organization. The rest of this paper is organized as follows. Section 2 presents a brief introduction of Hyperledger Fabric. In Section 3 and Section 4, we describe how is Accelerator designed combining with the incumbent Hyperledger Fabric network, and show its impacts on the Fabric by extensive performance evaluations, respectively. In Section 5, we finally conclude the paper.

2. Background: Hyperledger Fabric

Fabric is one of open-source Hyperledger project hosted by the Linux Foundation. Since the detailed description of Fabric is in [18], here we provide brief explanations to understand how Accelerator runs with Fabric.

Chaincode, ledger and state. Chaincode, which corresponds to *smart contracts* of other blockchain technologies such as Ethereum [9] and Corda [10], refers to a program code containing functions that can be executed by some peers when a client submits a transaction proposal. Ledger is an append-only data structure where all transactions are recorded in a hashed chain. State is a concise form of the latest ledger written in a versioned key-value store.

Clients, peers and orderers. Transaction proposals are generated by clients and delivered to selected peers, referred to as *endorsers* or *endorsing peers*, specified by the *endorsement policy* of a chaincode. All peers validate transactions as well as maintain the blockchain ledger and the state at the latest version. Orderers altogether determine the order of all transactions and make a block. After the ordering, the orderers send the block to all peers for validation.

Transaction flow. The consensus of transactions in Fabric is established by *execute-order-validate* architecture as shown in Fig. 2.

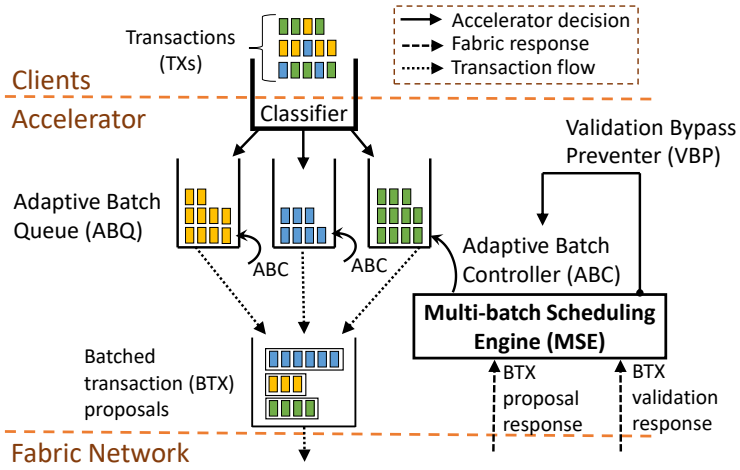


Fig. 3. Architecture of Accelerator

○ *Execution:* A client submits a transaction proposal to endorsers for the proposal execution. After the execution, each of the endorsers sends back a proposal response, called *endorsement*, that is cryptographically signed and includes a *writeset* consisting of key-value updates, which is a result of the execution, and a *readset* containing the versioned keys read in the execution.

○ *Ordering:* When the endorsements are sent back to the submitting client, the client submits the transaction with its endorsements to orderers. Then, the orderers decide the order of all transactions and place them into blocks according to its policy, e.g., the maximum number of transactions and timeout. The generated blocks are delivered to all peers.

○ *Validation:* When a peer receives a block, the peer verifies the endorsement of each transaction within the block. If the policy is satisfied, the peer sequentially checks whether the readset version of each transaction is valid against current state or not. After validation, the block is appended to the ledger and the state is updated. The submitting client can receive an event about the appended block from peers so that the client can confirm whether the submitted transaction has been committed.

3. Accelerator Design

In this section, we present our software module, Accelerator, for enhancing the consensus process of blockchain. We start by an overview of Accelerator architecture.

3.1. Overview

Overall structure of blockchain network. Accelerator is a transaction scheduling module that is placed between clients and Fabric network as depicted in Fig. 3. Note that Fabric network is composed of the nodes for establishing consensus, e.g., endorsers, orderers, and peers. Accelerator communicates with the nodes and the clients where all connections can be established in wired/wireless manners. To focus on the performance of blockchain with Accelerator, we assume that the network throughput is large enough thus the transmission delay is negligible.

Accelerator architecture. When the transactions are submitted by clients, Accelerator decides how aggressively the transactions should be batched in an endorsement-dependent manner. To this end, the transactions are classified according to the chaincode and its endorsement policy and sequentially enqueued into Adaptive Batch Queue (ABQ). The classified transactions in ABQ are regulated by Multi-batch Scheduling Engine (MSE), which is composed of Adaptive Batch Control (ABC) and Validation Bypass Preventer (VBP). ABC is a key component determining the aggressiveness of batch scheduling, controls how many transactions in ABQ are grouped as a batch as well as when to dequeue the Batched Transaction (BTX) proposal. ABC determines the size of BTX depending on the Fabric network responses, e.g., the BTX proposal response and the validation response. In brief, the classified transactions in the same ABQ are grouped as a BTX according to ABC, and the BTX is sequentially delivered to endorsers at once. Moreover, ABC is regulated by VBP to avoid a bypass of the validation procedure.

Batched transaction flow with Accelerator. Fig. 4 depicts a simple example of BTX flow with Accelerator. As we explained, Accelerator acts as an intermediary between Fabric network composed of nodes (e.g., endorser, orderer, and peer) and clients. On behalf of clients, Accelerator submits BTX proposal made of the transaction requests from clients to Fabric network. We introduce two key features of Accelerator in order to accomplish the successful BTXs' commitment to Fabric.

○ *Execution of BTX with chaincode-agnostic module:* Even though Accelerator submits the transactions as a form of batch, the individual transactions should be executed separately by each chaincode. Therefore, we develop a *chaincode-agnostic module* that decomposes a BTX to its original transactions and returns the total results as an endorsement. This module can be implemented in any chaincode running on Fabric.

○ *Validation bypass preventer (VBP):* Without VBP, using

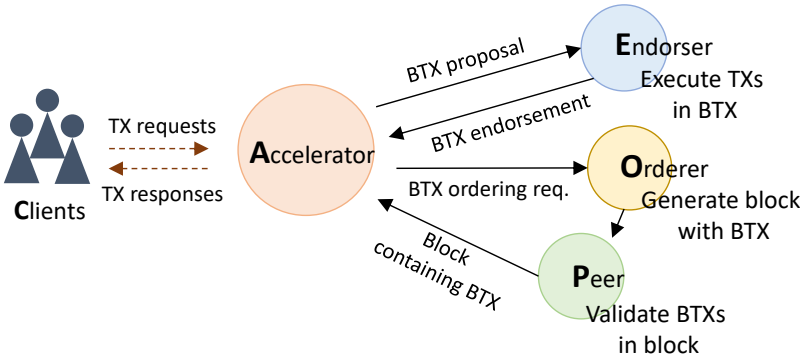


Fig. 4. Simple example of BTX flow with Accelerator

Accelerator may cause a problem that Fabric cannot detect a validation failure resulted in disordered transaction proposals. Therefore, VBP solves such a non-detection problem by preventing transactions disordering before endorsing. By regulating ABC, VBP forces a BTX not to include the transactions that try to update (i.e., write and read) the same keys. The detailed algorithm of VBP is described in Section 3-3.

3.2. Performance of Blockchain Network with Accelerator

In this subsection, we introduce how we define the performance of Accelerator based on Fabric network in terms of transaction throughput and transaction latency.

Transaction throughput. A blockchain network's transaction throughput, simply throughput, is defined as the *maximum* number of successful transactions per second, where a transaction is regarded as successful when it establishes consensus so that it is committed to the ledger. The throughput depends on the complexity of consensus procedure and the computing power of Fabric nodes (i.e., endorsers, orderers, and peers). We denote by $\mu_q(\beta_q)$ the throughput of transactions classified in ABQ $q \in \mathcal{Q}$, where we let $\mathcal{Q}$ be a set of ABQ and the batch size of BTX is defined as β_q. The batch size β_q is assumed to be a natural number, i.e., $\beta_q \in \mathbb{N}$.

Transaction latency. A transaction's latency, simply latency, is defined as the sojourn time of a transaction at the blockchain network. The latency is the sum of the transmission delay at the network links between nodes, the transaction computing delay at each node, and the queueing delay which occurs the batching processes at ABQ generating BTX as well as the orderers making a block. However, the transmission delay and the transaction computing delay are negligible compared to the queueing delay, thus we assume that the transaction latency is equal to the queueing delay. We denote by $\omega_q(\beta_q)$ the average latency of transactions classified in ABQ $q \in \mathcal{Q}$, where the batch size of BTX is β_q.

Performance of Accelerator. Denote $f_q(\theta_q, \mu_q(\beta_q), \omega_q(\beta_q))$ be the performance of ABQ $q \in \mathcal{Q}$, where $\theta_q \in [0, 1]$ refers to the heterogeneous preference of the transactions dedicated in an ABQ $q \in \mathcal{Q}$ on the high transaction throughput rather than the low latency. For example, if $\theta_q = 1$, then $f_q(\cdot) = \mu_q(\beta_q)$ and if $\theta_q = 0$, then $f_q(\cdot) = \omega_q(\beta_q)$. The value of θ_q depends on the characteristic of transactions in the ABQ q.

3.3. Multi-Batch Scheduling Engine of Accelerator

We aim at understanding how to control ABC and VBP, which are the key components of MSE, determining the adaptive batch size to maximize the performance of Accelerator.

Multi-Batch Scheduling Engine of Accelerator

Adaptive batch controller (ABC). ABC mainly controls how many transactions in each ABQ are grouped as a BTX, and when to dequeue the BTX proposal to endorsers. ABC determines the maximum batch size $\bar{\beta}_q$ for an ABQ $q \in \mathcal{Q}$, to maximize the blockchain performance, i.e.,

$$\bar{\beta}_q \quad = \quad \arg\max_{\beta_q \in \mathbb{N}} f_q(\theta_q, \mu_q(\beta_q), \omega_q(\beta_q)), \ \forall q \in \mathcal{Q},$$

where θ_q is given. Moreover, to avoid an excessive queueing delay at the ABQ, we set the maximum waiting time τ_q. Then, the upper bound of batch size becomes $\lfloor \tau_q \lambda_q \rfloor$, where the average transaction arrival rate at q is denoted by λ_q. Consequently, we define the optimal batch size $\beta_q^\star$ for the ABQ $q \in \mathcal{Q}$ as follows:

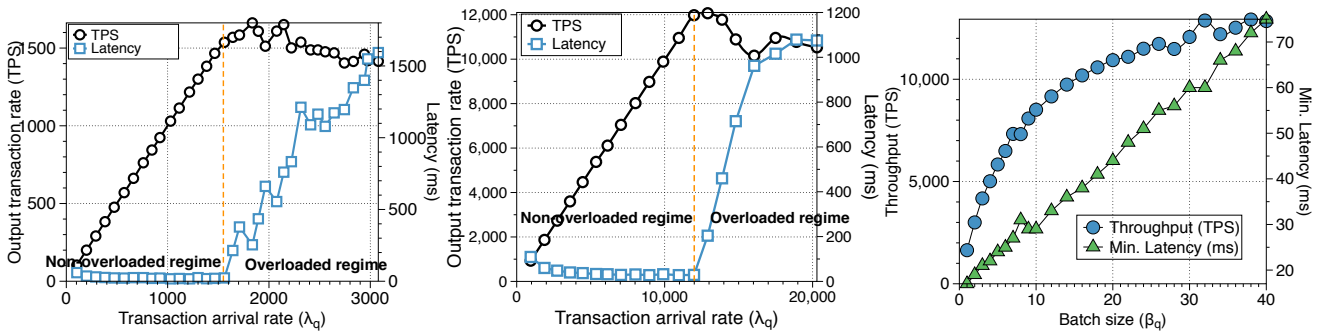
$$\beta_q^\star = \min\left(\bar{\beta}_q, \lfloor \tau_q \lambda_q \rfloor\right), \quad \forall q \in \mathcal{Q}. \tag{1}$$

Note that $\beta_q^\star$ is adaptively determined by the values μ_q, ω_q and λ_q, which are observed by MSE.

Validation bypass preventer (VBP). VBP ensures the correctness of ABC. If the b-th incoming transaction into ABQ $q \in \mathcal{Q}$ causes a readset version conflict with transactions in the ABQ q, then VBP forces ABC to immediately generate a BTX with all existing transactions in the ABQ except the b-th transaction, i.e., $\beta_q^\star = b - 1$.

3.4. Rationale of Accelerator

We now present a rationale of *Accelerator*. The main contribution of Accelerator is to determine how to choose the batch size β_q using ABC, as shown in equation (1). The consensus



(a) Throughput and latency ($\beta_q = 1$) (b) Throughput and latency ($\beta_q = 20$) (c) Throughput and latency

Fig. 5. Impact of Accelerator on throughput and latency

procedure of Fabric, which is established by execute-order-validation architecture, has been performed for every single transaction. However, we focus on the fact that a set of transactions using the same chaincode under the same endorsement policy is sequentially executable, moreover, can be cryptographically signed at once. Therefore, we design the ABQ, which is assigned to such a set of transactions, controlled by ABC and VBP. This simple, yet powerful idea of Accelerator helps a lot to improve the transaction throughput of Fabric, since the computing power consumption caused by cryptographic operations during endorsement/validation phases are in proportion to the number of transactions. Therefore, Accelerator can significantly reduce the computing delay of each node in Fabric by submitting BTX proposals rather than individual transaction proposals.

The ultimate goal of Accelerator lies in maximizing the performance of blockchain ledger by adjusting the batch size. The batch size β_q of ABQ $q \in \mathcal{Q}$ is a crucial factor that trades off the transaction throughput and the queueing latency in each ABQ. For example, if β_q grows, each peer (either endorser or not endorsing peer) consumes less computing power per a single transaction so that the total transaction throughput of blockchain network is augmented, but the queueing delay increases at the ABQ $q \in \mathcal{Q}$. Therefore, we suitably choose the parameter β_q, which depends on the preference on throughput or that on latency of the grouped transactions in the same ABQ $q \in \mathcal{Q}$. In this paper, we design the performance $f_q(\cdot)$ of an ABQ $q \in \mathcal{Q}$ as follows:

$$f_q(\theta_q, \beta_q) = \theta_q \mu_q(\beta_q) - (1 - \theta_q)\omega_q(\beta_q), \forall q \in \mathcal{Q}, \qquad (2)$$

where we remind that $\mu_q(\beta_q)$ and $\omega_q(\beta_q)$ are the throughput and the latency, and $\theta_q \in [0, 1]$ refers to the heterogeneous preference of the transactions in an ABQ q. The function $f_q(\cdot)$ models how the performance of Accelerator reflects the tradeoff between throughput and latency as a concavity with respect to the batch size β_q. To this end, we assume that $f_q(\cdot)$ linearly increases with $\mu_q(\beta_q)$ but decreases with $\omega_q(\beta_q)$, in order to take a balanced weight between throughput and latency, where the throughput $\mu_q(\beta_q)$ increases with β_q due to saving computing power of each peer while the latency $\omega_q(\beta_q)$ increases in proportion to the size of β_q. In the following section, we provide extensive evaluation results that reveal the impact of Accelerator on the performance of Fabric network.

4. Performance Evaluation

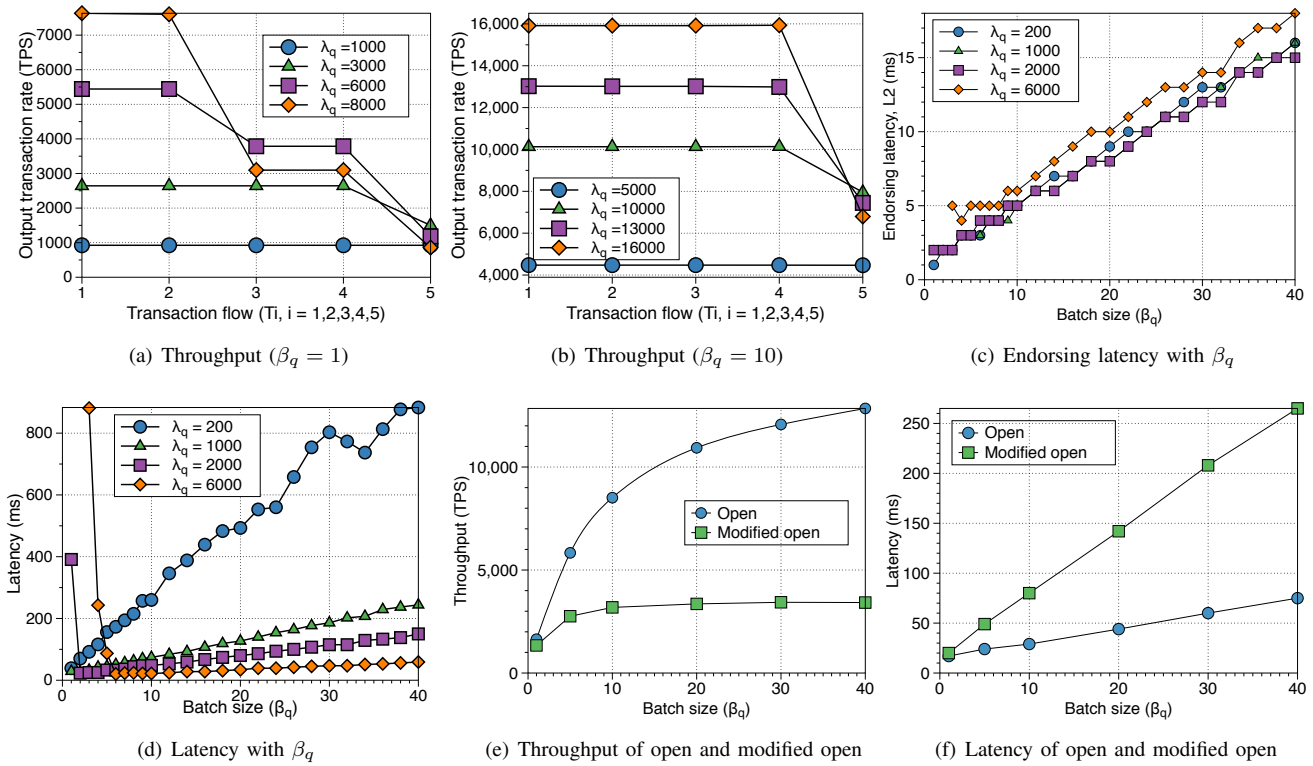
Our performance evaluation aims at understanding (i) how the nodes in Fabric network actually work for transaction flow and (ii) how to attain the controllability on Fabric's performance using Accelerator.

4.1. Implementation, Test scenarios, and Measurements

Implementation. For the evaluation, we first construct a Fabric network with version 1.4, which is composed of three peers, one endorser, and one orderer [1]. We use Hyplerledger Caliper [15], which is a test harness provided by Hyperledger, applying some modifications in order to make proper simulation scenarios for IoT applications. We exploit two Calipers, each of which is composed of eight clients who randomly generate a massive amount of transactions. Accelerator is implemented as a stand-alone server that mediates the Calipers and Fabric network using gRPC call [20]. Each node, each Caliper, and Accelerator run on the x86_64 virtual machines in a Samsung SDS cloud, where each virtual machine is allocated 8 vCPUs of Intel Xeon CPU E5-2690 v4 @ 2.60 GHz and 32 GB of memory. Moreover, all nodes are communicated with 2 Gbps network.

Test scenarios. As we mentioned, we modified the Caliper to evaluate Accelerator in order to make a virtual simulation case where a lot of transactions using IoT applications are randomly generated. We assume that the transaction proposals are independently submitted and follow the Poisson process, in which the inter-arrival time between transactions is a random variable following the exponential distribution with parameter λ_q. We exploit the *open* function in a *simple* chaincode provided by the Caliper. Evaluations are conducted in two

[1] In detail, we assume that a single organization with a single channel where the peers, the endorser, and the orderer exist. We use solo type orderer with block size ten. Due to the space limitation, we did not mention a part of Fabric structure such as organization and channel, but we believe that it does not disrupt understanding the feature of Accelerator. We provide the evaluation environment in Nexledger Accelerator github [19].



(a) Throughput ($\beta_q = 1$) (b) Throughput ($\beta_q = 10$) (c) Endorsing latency with β_q

(d) Latency with β_q (e) Throughput of open and modified open (f) Latency of open and modified open

Fig. 6. Impact of Accelerator on throughput and latency

scenarios. One is a plain scenario using open function that operates read and write once, and another is a modified open that repeats read operation *three* or *ten* times and write once. Under those scenarios, we perform extensive simulations to know how to choose the batch size for the best performance of blockchain-based IoT applications. Moreover, to focus on the impact of batch size, we consider the cases where the transactions have non-conflicted key-value pairs to avoid the key duplications and the invalidations.

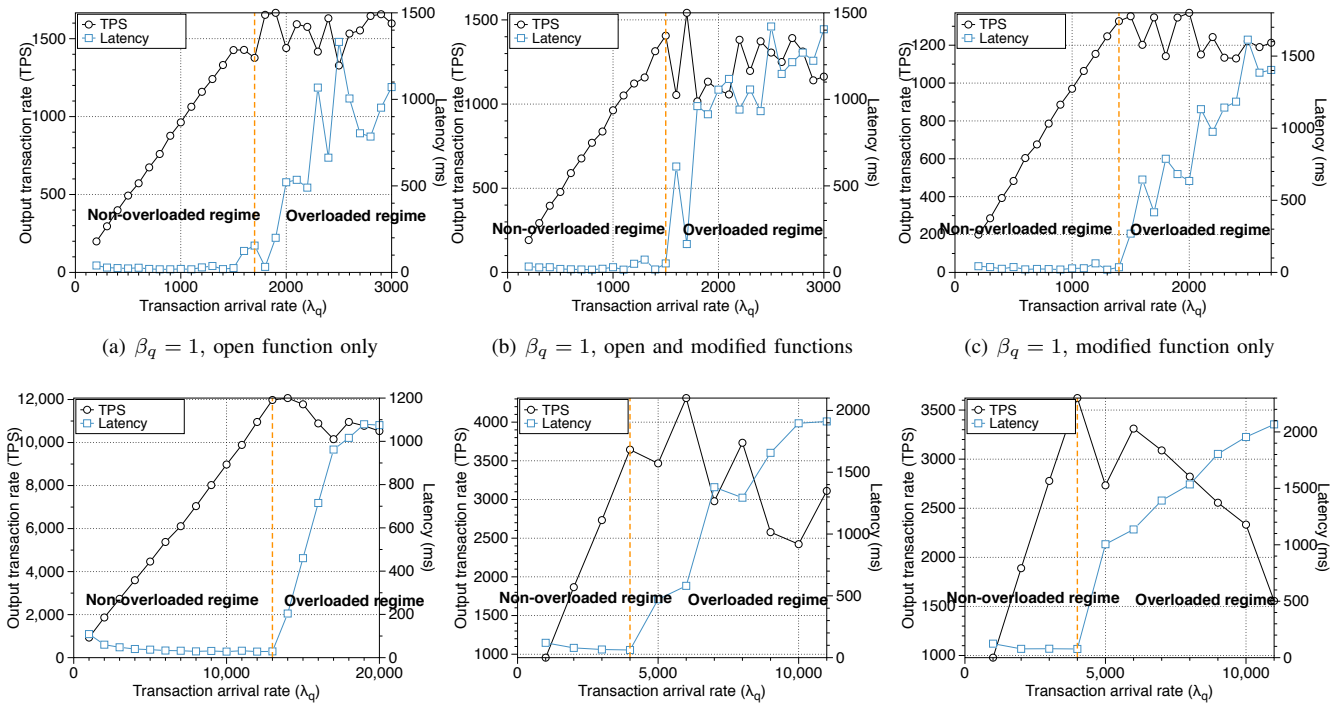
Measurements. We carefully deploy the measurement module on Accelerator, which aims at understanding the impacts on the detailed transaction flow. By doing so, we can operate ABC and VBP properly. First, we measure all transaction input/output rates passing through Accelerator as shown in Fig. 4. The rate `T1` is from Caliper to Accelerator, `T2` is from Accelerator to endorser, `T3` is from endorsers to Accelerator, `T4` is from Accelerator to orderer, and `T5` is from validating peer to Accelerator. The transaction rates `T1`-`T4` are used as a proof of sound transaction flow and the final output transaction rate from Fabric network `T5`, shortly *output transaction rate*, is used for finding the throughput of ABQ $q \in \mathcal{Q}$. Moreover, the latency $\omega_q(\beta_q)$ is observed at a cyclic path through Accelerator from the transaction request to the transaction response at Caliper. Additionally, we measure the endorsing latency `L2` which occurs during endorsement procedure.

4.2. Two-regimes: non-overloaded and overloaded

Before we study how to improve the performance of Fabric by Accelerator determining β_q, we first focus on the impact of increasing transaction arrival rate on the output transaction rate and the latency on Fabric network. Fig. 5(a) and Fig. 5(b) depict how the output transaction rate and the latency change with the transaction proposal arrival, λ_q, when $\beta_q = 1$ and $\beta_q = 20$, respectively. The behavior of Fabric is divided into two regimes. One regime is the *non-overloaded regime*, which occurs if the throughput is larger than the transaction arrival rate, i.e., $\lambda_q < \mu_q(\beta_q)$. Under the non-overloaded regime, the output transaction rate of Fabric increases with the arrival rate until the following condition is met:

$$\lambda_q = \mu_q(\beta_q).$$

Therefore, we can empirically find $\mu_q(\beta_q)$. For example, we observed that $\mu_q(1) \approx 1,500$ TPS and $\mu_q(20) \approx 12,000$ TPS, as depicted in Fig. 5(a) and Fig. 5(b). Also, if the transaction arrival rate grows over the throughput, i.e., $\lambda_q > \mu_q(\beta_q)$, Fabric network goes to the *overloaded regime*, then the submitted transactions might not be finalized. Fig. 5(a) also illustrates the latency $\omega_q(\beta_q)$. The latency under the non-overloaded regime decreases with the arrival transaction rate due to the decrease of queueing delay at ABQ q while the latency infinitely increases due to the growing queue under the overloaded regime.



(a) $\beta_q = 1$, open function only (b) $\beta_q = 1$, open and modified functions (c) $\beta_q = 1$, modified function only

(d) $\beta_q = 20$, open function only (e) $\beta_q = 20$, open and modified functions (f) $\beta_q = 20$, modified function only

Fig. 7. Impact of Accelerator on throughput and latency with multi-transaction requests

4.3. Impacts of Accelerator on Fabric network

Impacts of Accelerator and batch size decision. Fig. 5(c) describes how Accelerator impacts on the throughput $\mu_q(\beta_q)$ and the minimum latency for each fixed batch size β_q. We can find the value of $\mu_q(\beta_q)$ at the point satisfying $\lambda_q = \mu_q(\beta_q)$. As the batch size β_q grows, the throughput is logarithmically increasing while the latency is linearly increasing. In other words, $\mu_q(\beta_q)$ and $\omega_q(\beta_q)$ tend to be a logarithmically increasing function and a linearly increasing function, respectively. This phenomenon is very important not only to understand the behavior of Accelerator but also to determine the optimal batch size β_q. For example, if we assume that the performance function is simply defined as follows:

$$f_q(\theta_q, \beta_q) = \theta_q \log \beta_q + (1 - \theta_q)(-\beta_q),$$

then the $f_q(\theta_q, \beta_q)$ is a concave function with respect to β_q, thus we can find that $\beta_q^\star = \frac{\theta_q}{1-\theta_q}$ at the point where the first derivative becomes zero. Then, $\beta_q^\star$ is an increasing function with respect to $\theta_q \in [0, 1]$. This result is quite reasonable since it shows that a set of transactions who prefers a high throughput, i.e, high θ_q, tends to choose a large batch size β_q. In the following evaluations, we investigate the reason why $\mu_q(\beta_q)$ and $\omega_q(\beta_q)$ make such behaviors.

Impacts on throughput of each transaction flow. We now study which step becomes a bottleneck throughout the transaction flow with increasing transaction arrival rate and how Accelerator impacts on the bottleneck. Fig. 6(a) describes the transaction rates `T1`-`T5` with respect to various transaction arrival rate λ_q, when $\beta_q = 1$. Under the non-overloaded regime, e.g., $\lambda_q = 1,000$ TPS, there is no bottleneck phase thus the transaction rates `T1`-`T5` are equivalent to each other, because the computing power of each peer is enough to perform all transactions' consensus processes. When the Fabric network goes to the overloaded regime, e.g., $\lambda_q = 3,000$ TPS, the final output transaction rate `T5` decreases, it means that the validation procedure becomes a bottleneck. Furthermore, if $\lambda_q \geq 6,000$ TPS, `T5` severely falls, moreover, the transaction rate `T3` also decreases due to the lack of computing resource for endorsement. Such throughput degradation of Fabric network occurs because an endorsing peer suffers a lack of computing power since it conducts both endorsement (as an endorser) and validation (as a peer). Interestingly, Fig. 6(b), which illustrates the output transaction rates when $\beta_q = 10$, shows that the batch scheduling solves the throughput loss during the endorsement (see `T3`-`T4`). Therefore, we can conclude that the lack of computing power remains at the ordering and validation phase even though we adopt Accelerator, which causes the diminishing increase of throughput with respect to β_q, as shown in Fig. 5(c).

Impacts on latency. Fig. 6(c) and Fig. 6(d) respectively depict the endorsing latency and the latency of transactions in an ABQ $q \in \mathcal{Q}$, with respect to the batch size β_q. Overall, the latency increases as the growth of batch size β_q, due to the queueing latency at the ABQ. However, if the transaction input is high (e.g., $\lambda_q \geq 2,000$ TPS), the Fabric network may fall into the overloaded regime with the small batch size (e.g., $\beta_q \leq 4$), where the queueing latency infinitely grows. Otherwise, under the non-overloaded regime, a transaction

having a higher transaction arrival rate has a lower latency when batch size is fixed.

Impacts of chaincode. Fig. 6(e) and Fig. 6(f) shows the impact of chaincode function on the performance improvement by Accelerator. We compare two simulation scenarios as follows: (i) using *open* function and (ii) using *modified open* function, which are explained in Section 4-1. We can achieve more improvement in the throughput and less increase on the latency when we use the open function. This stands for the suitability of Accelerator on the IoT applications which generate a massive number of simply executable transactions.

Impacts of Accelerator on multi-transaction requests. Fig. 7 shows the impacts of Accelerator on throughput and latency when the transactions are requested from multiple types of chaincode functions. Transactions are generated by two Calipers that can request the *open* function or the *modified open* with ten times read operations. Fig. 7(a) and Fig. 7(d) illustrate the case where two Calipers generate open function requests only. With Accelerator, as shown in Fig. 7(d), the Fabric network achieves 7-8 times higher throughput than the cases without Accelerator in Fig. 7(a). Compared to the single Caliper case in Fig. 5(b), the Accelerator works well and achieves high improvement on throughput. Fig. 7(b) and Fig. 7(e) depict the case where one Caliper generates open function requests and the other generates modified open function requests. Moreover, Fig. 7(c) and Fig. 7(f) show the case where all Calipers generate modified open function requests only. In those cases, the throughput improvement by Accelerator decreases as shown in Fig. 7(e) and Fig. 7(f). Therefore, the performance enhancement of Fabric using Accelerator under multi-chaincode case would be our main future work.

5. Conclusion

In this paper, we developed a new scheduling algorithm, called *Accelerator*, which improves the transaction throughput of Hyperledger Fabric based IoT applications. Using adaptive multi-batched scheduling method, we resolved a chronical problem of blockchain technology, which is an insufficient transaction throughput, without compromising the integrity of consensus algorithm of blockchain networks. Moreover, according to the type of transaction, Accelerator can adaptively determine the batch size, which is a crucial factor that trades off the transaction throughput and the queueing latency. This simple but powerful idea enables Fabric-based IoT application to finalize more than 12,000 transactions per second, which achieves 8x improvement compared to the pure Fabric network.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[2] Muhammad Salek Ali, Massimo Vecchio, Miguel Pincheira, Koustabh Dolui, Fabio Antonelli, and Mubashir Husain Rehmani. Applications of blockchains in the internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2018.

[3] Tiago M Fernández-Caramés and Paula Fraga-Lamas. A review on the use of blockchain for the internet of things. *IEEE Access*, 6:32979–33001, 2018.

[4] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Elsevier Future Generation Computer Systems*, 88:173–190, 2018.

[5] Seyoung Huh, Sangrae Cho, and Soohyung Kim. Managing IoT devices using blockchain platform. In *Proc. of IEEE ICACT*, 2017.

[6] Bin Yu, Jarod Wright, Surya Nepal, Liming Zhu, Joseph Liu, and Rajiv Ranjan. Trustchain: Establishing trust in the iot-based applications ecosystem using blockchain. *IEEE Cloud Computing*, 5(4):12–23, 2018.

[7] Juah C Song, Mevlut A Demir, John J Prevost, and Paul Rad. Blockchain design for trusted decentralized iot networks. In *Proc. of IEEE SoSE*, 2018.

[8] Alexandru Stanciu. Blockchain based distributed control system for edge computing. In *Proc. of IEEE CSCS*, 2017.

[9] Ethereum. https://ethereum.org, 2018.

[10] Corda. https://docs.corda.net, 2018.

[11] Hyperledger. https://www.hyperledger.org, 2018.

[12] Parth Thakkar, Senthil Nathan, and Balaji Viswanathan. Performance benchmarking and optimizing hyperledger fabric blockchain platform. In *Proc. of IEEE MASCOTS*, 2018.

[13] Christian Gorenflo, Stephen Lee, Lukasz Golab, and S Keshav. Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second. arXiv preprint arXiv:1901.00910, 2019.

[14] Harish Sukhwani, Nan Wang, Kishor S Trivedi, and Andy Rindos. Performance modeling of hyperledger fabric (permissioned blockchain network). In *Proc. of IEEE NCA*, 2018.

[15] Caliper. https://www.hyperledger.org/projects/caliper, 2019.

[16] Oscar Novo. Scalable access management in iot using blockchain: A performance evaluation. *IEEE Internet of Things Journal*, 6:4694–4701, 2019.

[17] Shitang Yu, Kun Lv, Zhou Shao, Yingcheng Guo, Jun Zuo, and Bo Zhang. A high performance blockchain platform for intelligent devices. In *Proc. of IEEE International Conference on Hot Information-Centric Networking*, 2018.

[18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proc. of ACM EuroSys*, 2018.

[19] Accelerator. https://github.com/nexledger/accelerator, 2019.

[20] gRPC. https://grpc.io/docs/guides/concepts/, 2019.