BoR: Toward High-Performance Permissioned Blockchain in RDMA-Enabled Network

Bobo Huang, *Student Member, IEEE*, Li Jin, ZhiHui Lu, *Member, IEEE*, Xin Zhou, Jie Wu, *Member, IEEE*, Qifeng Tang, and Patrick C. K. Hung, *Member, IEEE*

Abstract—Known as a distributed ledger, blockchain is becoming prevalent due to its decentralization, traceability and tamper resistance. Particularly, permissioned blockchain such as Hyperledger Fabric shows great application prospects as the infrastructure of IoT security, credit management, etc. Many cloud platforms like AWS, Azure, Oracle and IBM cloud currently provide blockchain as a service, in which tenants can quickly build permissioned blockchain and run smart contract based applications. However, the transactions throughput and scalability in the permissioned blockchain are not ideal, despite many optimization efforts in consensus protocol and parallel chain. Existing solutions still reveals some limitations like excessive CPU scheduling, inefficient block broadcast and high latency of initial blocks synchronization when new nodes join blockchain network. Inspired by the emerging RDMA (Remote Direct Memory Access) network, we propose BoR, an RDMA-based permissioned blockchain framework. By offloading the block transfer transaction into RDMA NICs, it can increase block broadcast speed and reduce block sync delay. We exploit the RDMA primitives to redesign the block synchronization protocol and accelerate DPoS (Delegated Proof of Stake) consensus process for higher throughput and lower latency in kernel-bypass manner. As demonstrated in our evaluation with different workloads, BoR with lower CPU utilization significantly outperforms the state-of-the-art EoS blockchain.

Index Terms—Blockchain-as-a-service, RDMA, permissioned blockchain, one-sided communication, RoCE, kernel-bypass network

1 Introduction

Blockchain as a Service (BaaS) is an emerging powerful model that simplifies the deployment of distributed blockchain systems and provides an efficient tradeoff among transparency, efficiency and cost. On one hand, BaaS cloud providers concentrate on maintaining the important blockchain-related artifacts and infrastructures for their tenants. The complex back-end activities supported by BaaS include blockchain deployment, suitable resources allocation, security feature, and hosting requirements for their tenants. On the other hand, the tenants can focus on various blockchain-based applications by using BaaS cloud services without worrying about performance related issues. BaaS bridges the huge gap between flexibility and complexity. Many cloud providers including AWS Cloud [1], Microsoft Azure [2], Oracle and IBM cloud [3] have deployed cloud-based BaaS into their data centers.

With the increase of blockchain data volume and service request rate, BaaS needs the support of high-throughput and low-latency blockchain systems. In response, many research works have been devoted to blockchain performance analysis [8], [9], [10], [11] and consensus protocol [6], [7], [12], [13], [14], [15], [16]. The early PoW-driven blockchains such as Bitcoin [17] and Ethereum [5] require a large amount of computing power for hash calculation, which results in very low transaction throughput, high CPU consumption and large block interval. Thanks to the great efforts on consensus algorithms [6], [7], [12], [13], [14], [15], [16], the block internal in blockchain systems has been reduced from minutes to seconds, even milliseconds as shown in Table 1. PBFT [12] consensus has been adopted for Hyperledger Fabric [6] in which the block internal is 3.6 s, while DPoS [13] has been used for EoS [7] where the block internal is 0.5 s. Such efficient consensus protocols remove meaningless computing power loss and lead to higher transactions throughput. Hence, the performance bottleneck in blockchains has shifted to communication overhead including blocks broadcast delay and bootstrapping time of new nodes.

Moreover, with a larger-scale blockchain-as-a-service cloud, the latency blocks transmission will lead to more forks (Fig. 1) with higher probability of being attacked. Additionally, with the rapid evolvement of blockchain network (as shown in Table 2), the latency for bootstrapping new nodes increases linearly, which severely reduces the scalability of blockchain systems. Despite many research efforts, the traditional TCP-based blockchain is confronted with lots of challenges in terms of transaction throughput, block broadcast delay, CPU overhead and bootstrapping time of new nodes [18]. Therefore, how to reduce meaningless CPU involvement,

- *B. Huang, L. Jin, Z. Lu, X. Zhou, and J. Wu are with the School of Computer Science, Fudan University, Shanghai 200433, China, and also with the Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, Shanghai 200433, China. E-mail: {huangbb16, ljin18, lzh, xzhou18, jwu}@fudan.edu.cn.*
- *Q. Tang is with the Shanghai Data Exchange Corporation and with National Engineering Laboratory for Big Data Distribution and Exchange Technologies, Shanghai 200436, China. E-mail: keven@chinadep.com.*
- *P.C.K. Hung is with the Faculty of Business and IT, University of Ontario Institute of Technology, Oshawa, ON L1G 0C5, Canada. E-mail: patrick.hung@uoit.ca.*

Manuscript received 31 Mar. 2019; revised 3 Sept. 2019; accepted 10 Oct. 2019. Date of publication 21 Oct. 2019; date of current version 15 Apr. 2020. (Corresponding author: Zhihui Lu.)
Digital Object Identifier no. 10.1109/TSC.2019.2948009

TABLE 1
Comparison of Four Platforms of Blockchain

Platform	Block Interval	Consensus	TPS
Bitcoin	10 minutes	PoW [4]	about 7
Ethereum	10 to 20 seconds	PoS [5]	< 100
Fabric	3 to 6 seconds	PBFT [6]	> 1000
EoS	0.5 seconds	DPoS [7]	million

TABLE 2
Amount of Data and Synchronization Time

Platform	Total	Growth rate	Time
Bitcoin	170 GB	50 GB per year	2-3 days
Ethereum	430 GB	270 GB per year*	> 2 days
EoS	330 GB	12 GB per month**	1-2 days

** From 2016 to 2017*
***The first month of EoS*

how to minimize the block broadcast delay and the bootstrapping time of new nodes, are significant challenges for the permissioned blockchain in data center network.

Meanwhile, emerging advanced hardware features like high-speed interconnect with RDMA [19] pose new opportunities for high-performance quality of data-intensive applications [20], [21], [22], [23], [24], [25]: the RDMA hardware support for kernel bypassing and TCP/IP stack stateless offloads properties makes it very promising to offload data transmissions into RDMA NIC (RNIC) Host Channel Adapters (HCAs). Furthermore, RDMA over Converged Ethernet (RoCE) [24] has been widely deployed in many data centers [8], [26], [27] to accelerate big data transfers. Many applications in data centers exploit RDMA primitives (such as RDMA WRITE/READ/SEND) [28] to overwrite the underlying communication mechanisms for higher throughput, lower latency and less CPU involvement, which is called RDMA-enhanced paradigm. For instance, RDMA-driven graph-based RDF store system for low-latency and highly concurrent query service over large-scale datasets, was put forward by Wukong [20]. RDMA_Mongo [29] employs one-sided RDMA primitives to redesign the oplogs synchronization for faster large-scale data processing in document-based NoSQLs. Octopus [30] leverages RDMA and NVM to implement low-latency persistent memory file system. Other RDMA-enabled systems focus on key-value store [27], [27], [31], [32], [33], distributed file system [22], [34] .etc. Therefore, can we implement permissioned blockchain over RDMA to boost higher performance of blockchain as a service in cloud data centers?

Inspired by RDMA-enhanced paradigm, we propose a high-performance permissioned blockchain over RDMA named BoR, which is based on open-source EoS framework [7]. First, we design a RDMA context detection method when a local node establishes connections with its peer nodes. If the local and remote nodes are all in the interconnected RDMA network, the RDMA QP metadata including RNIC gid index and device port number is exchanged through TCP connections. The RDMA channels between them will be created and pushed into local channel list. Otherwise, the TCP socket-based connection will be used for communication. Second, the local node sends handshake message to all TCP connections and RDMA channels for state synchronization. Then it receives a notice message including the state of blocks and transactions on remote peer nodes. Based on such notice message, the local node determines whether block data synchronization action is required or not. Transmission data can be divided into ACK, control message, transactions and blocks. During the communication process, RDMA WRITE primitive with immediate data is exploited to send the transmission data. Then RDMA completion event channel is exploited to monitor and process the incoming messages. Eventually, we redesign the blocks/transactions synchronization protocol with RDMA primitives. Considering the overhead of registering and deregistering RDMA Memory Region (MR), we register sufficient MRs for each RDMA QP-based channel and introduce memory pool to manage MR. Whenever a message needs to be sent or received, the RDMA channel only needs to apply for one MR fragment and the corresponding memory address with the remote key.

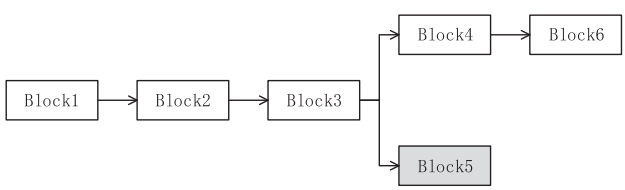


Fig. 1. The fork of a blockchain.

Our primary contributions are summarized as follows:

- We conduct a detailed analysis of the network communication protocol of the EoS blockchain, then determine the design goals and challenges of BoR, and propose the overall architecture of Blockchain over RDMA called BoR.
- We propose an RDMA-enabled sync manager for managing RDMA QP channel, Memory Region, RDMA adapter and block sync state.
- We redesign a new node bootstrapping and blocks/transactions broadcast protocol based on RDMA primitives with immediate data. And completion event channel is employed to guarantee low latency, high throughput and less CPU involvement.
- We implement the BoR prototype and evaluate it with increasing scale of block datasets. A comparison with the state-of-the-art EoS blockchain system shows that BoR achieves the reduced consuming time by up to *20.2 percent* and lower CPU overhead by *26.4*33.9 *percent* for the initial block synchronization when a new node joins the blockchain network.

The remainder in this paper is organized as follows. Section 2 introduces the key concepts about blockchains, RDMA and RDMA-driven system, then explains the motivation. Section 3 demonstrates the overview of BoR architecture, then discusses BoR design goals and challenges. In Section 4, we describe the key ideas of BoR, then RDMA-enabled initial synchronization and block broadcast. In Section 5, we evaluate the BoR system and compare it with the original EoS blockchain. In Section 6, discussions and related works on Blockchain and RDMA are illustrated. Section 7 presents the conclusion and future work.

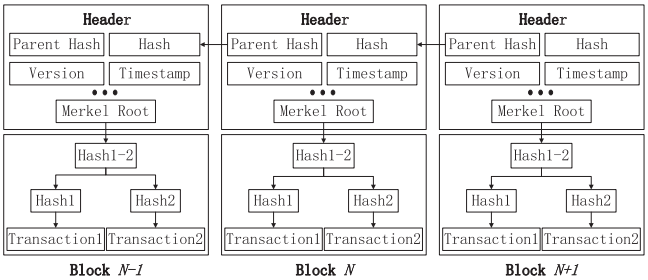


Fig. 2. Blockchain data structure.

2 Background and Motivation

2.1 Blockchain

In recent years, with the booming of emerging cryptocurrency such as Bitcoin [17] and Ethereum [5], the underlying blockchain technology has raised universal attention in academia and industry. Blockchain is a decentralized distributed ledger or database [35] which consists of a chained data structure based on block unit [18]. It is designed to enable transparent multi-party transactions services without third-party trust guarantee. Byzantine fault-tolerant consensus protocols [36] (such as provable voting mechanism [37]) are leveraged to ensure the consistency among blockchain nodes, and each joined node has a complete blocks data copy. Scalable P2P network [38] is exploited to realize decentralization of blockchain. Digital signatures [39] and Merkle Tree [40] are used to prevent transaction records on the blockchain from being maliciously tampered. Smart contracts provide the programmability of blockchain platform and every data record maintained within blockchain can only be accessed and traced by running smart contracts [41].

The following three key concepts are the basis of blockchain:

Transaction. As a carrier of assets or value, a transaction can be broadcast to the blockchain network and collected into blocks. Generally, previous transaction outputs are referenced as new transaction inputs by a transaction. A transaction can only be regarded as an irreversible one when a sufficient number of trusted peer nodes reach a consensus on a block being valid. All transactions which have been aggregated into a block can be browsed as transactions are unencrypted in blockchain.

Block. A block can be treated as a batch of transactions. Every transaction is permanently recorded in blocks files. All blocks are organized into a linear chain-based structure over time, as shown in Fig. 2. Each block is composed of block header and body. The block header contains the metadata including parent hash, local hash, timestamp and Merkel root while the block body consists of transactions which organized into a Merkel tree. New transactions processed by miners are aggregated into the Merkel tree in new blocks which are appended to the end of blockchain.

Consensus. Consensus protocols are leveraged by a blockchain context to guarantee the integrity of the blockchain. The critical objectives of the consensus are to prevent unwanted forks or double spending.

Basically, blockchain can be divided into public, private or permissioned chain. In public blockchains (such as Bitcoin [17] and Ethereum [5]), anyone can be allowed to participate in, leave, audit, access, and write the ongoing activities. That helps to maintain the self-governed features. Compared with the public blockchain, private or permissioned blockchain like Hyperledger Fabric [42] and EoS [7] is generally deployed in data centers and designed to build a platform with high credibility where each participant is aware of others. This paper focuses on the private/permissioned blockchain.

However, with the increasing scale of historical block datasets and the rapid evolution of blockchain architecture with faster consensus, TCP-based communication layer in blockchain cannot satisfy the requirements of low latency, high throughput under high concurrency level. The underlying network stack for traditional blockchains is a potential bottleneck, especially for the initial blocks synchronization when a new node joins a blockchain network.

2.2 Blockchain as a Service

Blockchain as a Service (BaaS) is a cloud-based solution which provides a distributed blockchain runtime for high flexibility and availability demanded by tenants. The cloud providers focus on managing the blockchain-related backend services, components, and resources allocation while the tenants focus on developing functionalities, smart contracts, and blockchain applications [11], [35], [39], [43], [44] in a straightforward manner. Hence, BaaS model gained great traction recently in academia and industry. For instance, Blockstack [35] is a Bitcoin-based naming and storage system in which minimal metadata is stored in Bitcoin. The prior work [44] proposes the blockchain-based searching scheme over encrypted data in cloud to resist malicious attacks.

However, with the growing number of blockchain applications and the increasing scale of users of blockchain applications, the high request rate of BaaS in cloud data centers generates a strong demand for high-performance distributed blockchain systems. While prior systems [11], [14], [15] have proposed some efficient optimization strategies and algorithms to deal with the performance gap, they fall short in leveraging the rich modern hardware like RDMA devices in data centers to achieve higher quality of BaaS.

2.3 EoS and Graphene Technology

This section describes EoS and its underlying technology. The prototype implementation of the RDMA-driven permissioned Blockchain in this paper is based on the EoS system.

EoS. EoS is an emerging blockchain platform developed by Blockone. The objective of EoS is to implement a blockchain architecture that supports the functionalities and applications like an operating system [7]. Due to the limitations of high latency and low throughput in common blockchain applications, EoS employs an efficient parallel chain and DPoS consensus to boost a faster blockchain. Unlike the PoW and PoS consensus mechanisms, the DPoS nodes need to participate in the witness election. Only the nodes (at least 21 nodes) who win the election can be entitled to the generation of blocks. In addition, another 100 candidate nodes are regarded as witness candidates. They are reckoned as substitutes once the 21

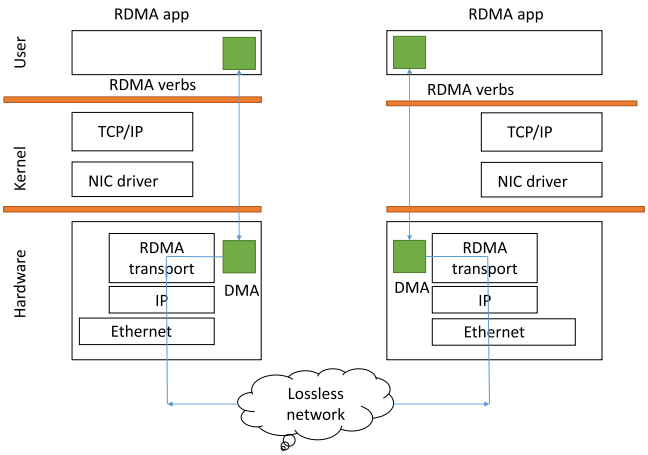


Fig. 3. The overview of RDMA enabled communication.

witness nodes have problems. The current block interval of EoS is about 0.5 seconds [7].

There are three main components in EoS: nodeos, cleos and keosd. Nodeos acts as a server node component. Cleos is a command line interface that is responsible for interacting with blockchains, managing wallets and accounts, and calling smart contracts on blockchains. Keosd is exploited to manage the wallets in EOSIO.

Graphene Technology. Graphene is a blockchain toolkit developed by the BitShares team which can enable higher concurrency level. It is an open-source blockchain underlying library which is based on DPoS consensus and has been leveraged in EoS. With Graphene technology, EoS can reach millions of Transactions Per Second (TPS) through the parallel chain. The local parallel chain in EoS can even reach the millisecond-level confirmation speed. Due to its high performance in consensus and TPS, EoS is employed by this paper as the basis of our BoR prototype.

2.4 RDMA and Its Characteristics

Remote Direct Memory Access (RDMA) is a completely kernel-bypassing and high-speed interconnect technique with memory semantics by offload the entire TCP/IP protocol stack into dedicated RDMA NICs (RNICs). As shown in Fig. 3, lossless network is leveraged by RDMA to achieve high bandwidth while zero copy technology is employed to reduce the number of system interrupts and memory copies for ultra-low latency. Two kinds of message-oriented primitives are provided by RDMA: two-sided message passing interfaces such as SEND/RECV Verbs and one-sided primitives such as WRITE, READ and atomic operations (Compare-and-Swap and Fetch-and-Add). Compared to one-sided RDMA verbs, two-sided operations still have CPU involvement to remote host and user-space memory copies, which introduces an unavoidable overhead [45], [46]. To minimize the unnecessary overhead during the initial block synchronization, we leverage one-sided RDMA WRITE with immediate for bandwidth-sensitive data messages (such as blocks and transactions).

Current RDMA provides three transport modes which contains Reliable Connection (RC), Unreliable Connection (UC) and Unreliable Datagram (UD). The available operations in each RDMA mode are different. For instance, the connection-oriented RC mode supports all RDMA verbs while the stateless UD mode only supports RDMA SEND/RECV verbs. Moreover, only RC mode can guarantee the sequential delivery of packets. To guarantee the high reliability of the blocks/transactions transfer in BoR, we focus on the connection-oriented RC transport in this paper.

As demonstrated in prior works [47], proper combination of different RDMA primitives (two-sided or one-sided) can promote higher performance for RDMA enhanced system. Therefore, two-sided RDMA SEND/RECV verbs are exploited for exchanging registered message buffers among BoR nodes, while one-sided RDMA WRITE with immediate is leveraged for control and data messages during block broadcast and synchronization in BoR.

2.5 Motivation

Blockchain as a Service (BaaS) has been widely deployed into many cloud data centers [1], [2], [3] and exposes the blockchain-related programming interfaces to tenants and developers. A large amount of smart contract-based applications employ cloud-based BaaS as their underlying blockchain runtimes. With the great popularity of BaaS and the increasing number of smart contracts over BaaS, the growing request rate of BaaS requires a faster and higher concurrent permissioned blockchain. However, due to the unavoidable overhead introduced by redundant memory copies, frequent CPU context switching and interruptions in the transport layer of traditional TCP-based blockchains, existing solutions [7], [11], [14], [42] cannot achieve the desired quality of BaaS like high throughput and low latency. On the other hand, more faster and efficient consensus algorithms make blocks synchronization and broadcast potential bottlenecks. Meanwhile, modern RDMA hardware in data centers has been widely leveraged to enhance data-intensive applications such as NoSQL systems [20], [27], [27], [29], [31], [32], [33] and distributed file systems [22], [34] due to its ultra-low latency and high throughput with remote CPU/OS bypassing. The kernel bypassing feature supported by RDMA reduces the number of CPU context switching and system interruptions while the TCP/IP stack offload and zero copy technique drastically reduce the number of payload copies to achieve low latency, high message rate and low memory bus contention. Inspired by this, we attempt to exploit hybrid RDMA primitives (RDMA WRITE and SEND/RECV) to accelerate the initial block synchronization and blocks/transactions propagation in BaaS. To the best of our knowledge, BoR is the first attempt to apply RDMA to boost high-performance permissioned blockchain for cloud-based BaaS.

3 Overview

3.1 Design Goals

More Efficient Synchronization. When a new node joins a blockchain network, it will first synchronize all historical blocks in this blockchain. It takes 1 to 3 days to finish this operation (as shown in Table 2), which is significantly inefficient. To be worse, during the initial block synchronization of the new nodes, the producers in this blockchain continue to generate new blocks. Thus, the new nodes have to take a longer time to synchronize the new generated blocks. In recent years, the block interval has reduced obviously. As

shown in Table 1, the shortest block interval is 0.5 second in EoS. Therefore, if the synchronization speed is lower than the generation speed, for instance, 1 second per block for synchronization while 0.5 seconds per block for generation, the new nodes will always catch up with the latest block and never finish synchronization. Since the network transmission will greatly affect the synchronization speed, our first design goal is to propose an efficient approach to synchronize the blocks with kernel-bypass RDMA technology.

Lower Broadcast Latency. When a producer generates a new block, it will broadcast it to all the other peers immediately for consensus-based confirmation. Other peers who received the block will verify it through the consensus protocol (such as PoW or DPoS) and then broadcast the validation result (valid or invalid block) to the whole blockchain network. Due to the shorter block internal (0.5 s), current EoS system requires more faster and efficient confirmation. That means, if a new block does not be verified by other peers for a long time due to network latency or other issues, the next block may be generated before the previous one being confirmed. The accumulated unconfirmed blocks may significantly affect the overall performance of blockchain system in BaaS cloud. Hence, reducing the broadcast latency is imperative and can obviously boost a high-performance blockchain. Therefore, the second design goal for BoR is to introduce an RDMA-based broadcast method in RoCE to accelerate the blocks broadcast. Further, we provide a lightweight RDMA-enhanced consensus mechanism with lower broadcast latency.

Higher Throughput and Lower CPU Consumption. Besides the optimization on business level of a blockchain, it is also critical to optimize the blockchain from system perspective. Owing to the demand for computing power in consensus-based hash calculations and high CPU utilization introduced by initial block synchronization, existing resource allocation for blockchain system is insufficient and inefficient. The transport layer under traditional TCP-based blockchain systems occupies the CPU resources due to unavoidable payload copies, frequent CPU context switching, and other OS-involved operations. To minimize the above overhead and decrease the CPU utilization, BoR leverages zero-copy feature and hybrid RDMA primitives (SEND/RECV and RDMA WRITE with immediate) to reduce the consumption of the system resource. (i.e., lower CPU overhead), and thus to improve the throughput of the blockchain network. Further, the resources saved in blocks/transactions transfer can be exploited by other blockchain operation like consensus. Therefore, we redesign a more efficient blocks synchronization over RDMA-capable network.

3.2 BoR Architecture: Blockchain over RDMA

BoR is a RDMA-enhanced distributed ledger with a distributed database structure. It consists of three primary components: nodeos, keosd and cleos. Nodeos is a core process running on the BoR server which is composed of chain manager layer, blocks store and transport layer. It carries the basic functions of interacting with the blockchain network like configuring system parameters, managing blockchain state, processing transaction/block requests, managing local block records, etc. Keosd is a

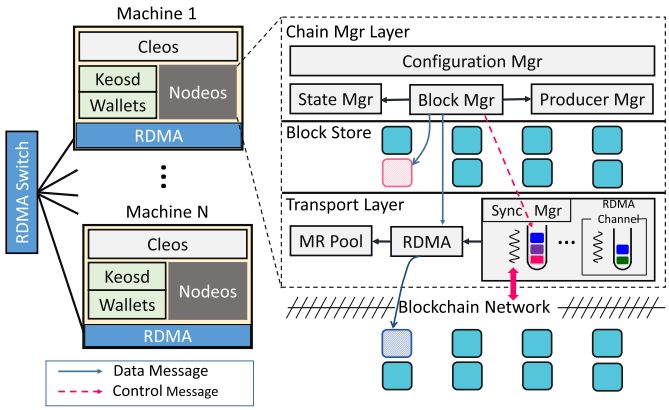


Fig. 4. The architecture overview of BoR.

wallet management client which is primarily used to manage public/private key and wallet information. Cleos is an interactive command line tool which bridges users and keosd/nodeos. This paper focuses on the nodeos component. We implement a RDMA-enabled transport layer for nodeos which can cooperate with chain manager layer and blocks store in a RDMA-friendly manner, as shown in Fig. 4.

Blocks store is a storage layer which is leveraged to store blocks/transactions in the local machine. It can provide the query/update/delete operations over blocks/transactions to the block manager of chain manager layer.

Chain manager layer called *Chain Mgr Layer* is composed of configuration manager (*Configuration Mgr*), state manager (*State Mgr*), blocks manager (*Block Mgr*) and producer manager (*Producer Mgr*). Configuration manager is mainly responsible for reading the BoR configuration file (like *config.ini* and *genesis.json*) and providing corresponding configuration parameters to the state manager, blocks manager and producer manager. State manager is employed to manage the state of local blocks/transactions. For example, blocks/transactions which are confirmed by the consensus protocol are regarded as irreversible blocks/transactions, while the blocks/transactions which are not confirmed are regarded as reversible blocks or transactions. Producer manager is leveraged to produce/sign new blocks and verify the legitimacy of blocks, signatures or transactions.

The block manager can receive or send control/data messages from/to the transport layer. To be specific, when the block manager receives a request from cleos, it generates a corresponding control message and pushes it to the send queue of the corresponding RDMA channel which is maintained by the sync manager in transport layer, as shown in Fig. 4. When receiving the control message from the transport layer, the block manager will take the actions corresponding to different message types (like handshake and request messages). The actions include requesting the blockchain status from the state manager or requesting signatures from the producer manager. For instance, when receiving a *sync request message*, block manager resolves the block head num or transaction id in the payload and query related blocks/transactions in blocks store. Then the result is packaged into data messages and sent back to the target node through one-sided RDMA primitives.

Transport layer consists of three primary components: Memory Region (MR) pool, RDMA and sync manager. The pre-allocated MR pool is exploited to mitigate the overhead from MR (de-/)registration at runtime. RDMA component with one-sided verbs provides efficient transmission for bandwidth-sensitive data messages like blocks/transactions. Sync manager manages a set of RDMA channels and every RDMA channel represents a session or connection between local and peer nodes. The RDMA channel employs an event channel-based worker thread to asynchronously receive control/data messages and callback the corresponding handlers. Additionally, sync manager can also receive messages from the block manager in chain manager layer and send them to the blockchain network.

3.3 Challenges

Multiple Message Type versus Unified Message Type in Permissioned Blockchain. In EoS, there are totally 9 message types. Commonly used messages are *handshake_message*, *notice_message*, *request_message* and *sync_request_message*. These four types of messages act as the control message between different peer nodes, which involve different fields to implement different features. There are two other types of messages, *signed_block* and *pack_transaction*, which transfer the blocks or transactions generated by producing nodes. The receiver handles the messages via an overloaded function *handle_message()*. The challenge is how to manage such different message types efficiently. BoR exploits one-sided RDMA WRITE with immediate to boost latency-sensitive control messages transfer as well as bandwidth-sensitive data messages transfer. However, the transport with one-sided primitives requires pre-registered memory regions (MRs) for sending and receiving message. This results in great difficulties to allocate a properly sized message buffer for RDMA memory semantics transfer with an unknown upcoming message type. The receiver cannot predict which type of message is upcoming while different message type has different message size. Although the message type is defined in message structure by the sender and then notified to the receiver, the introduced overhead will reduce overall performance. Therefore, we use a unified message definition which includes all types of message in BoR to simplify the allocation of memory buffers, the management of memory pool as well as the serialization and deserialization of messages. That means only one message abstraction is needed to represent different message types.

RDMA Channel Management in BoR. Another challenge is how to manage the RDMA channels efficiently. As mentioned in Section 2, the RDMA-driven paradigm requires dedicated RDMA-capable NICs and only machines with interconnected RNICs can enjoy the benefits of communication over powerful RDMA primitives. However, we cannot guarantee that every node in a blockchain network supports RDMA. Without the deployment of RNICs, machines cannot use RDMA channel to communicate with other nodes. Therefore, these nodes without RNICs will be isolated. To resolve this issue, we propose a hybrid solution: reserving both TCP channel and RDMA channel. A novel RDMA NIC detection algorithm is employed to determine whether the communication over RDMA between different machines is available or not. If available, the two peer nodes will leverage RDMA-enabled transport for blocks/transactions synchronization. Otherwise, the TCP-based transport layer is adopted for BoR. The RDMA context detection is executed on TCP connections during the startup of BoR.

Asynchronously Request Handling over RDMA-Driven Blockchain. The arrival of requests from BoR clients usually requires corresponding responses message produced by BoR server. In the case of high concurrency, client request messages should be handled asynchronously to reduce the blocking time. During the execution of BoR, the asynchronous handling can release resources like occupied threads to avoid blocking, wait until the response messages are generated, and then re-acquire one available thread for request processing. The challenge is how to use RDMA for asynchronous processing. RDMA has not provided already wrapped asynchronous handlers yet. Therefore, we create a listening thread which leverages RDMA completion event channel to receive the messages asynchronously which cached in RDMA Completion Queue (CQ). Then the messages attached to Completion Queue Element (CQE) are parsed and distributed into the corresponding message handlers while the event channel in the listening thread is continuously listening. Within message handler, the opcode field in CQE is used to identify *IBV_WC_RECV_RDMA_WITH_IMM* type from different primitive operations while the immediate data in CQE is used to identify different request types including *RDMA_IMM_DATA_ACK* and *RDMA_IMM_DATA_MESSAGE* type. After generating the response message, the message handler will leverage one-sided primitives (i.e., RDMA WRITE verbs) to send it.

4 BoR Design

In this section, we introduce the detailed implementation of RDMA-enabled sync manager, new node bootstrapping protocol and block broadcast mechanism.

4.1 The Definition of Blockchain Message over RDMA

The transmission data on BoR can be grouped into four types: *ACK*, *Control Message*, *block* and *transaction*. The control message consists of *handshake* message, *go away* message, *time* message, *notice* message, *request* message and *sync request* message. The handshake message, go away message and notice message are used to do handshake request/response for synchronizing the block status between receiver and sender nodes. Time Message is used for heartbeat detection between nodes. Request message and sync request message are used to synchronize reversible blocks/transactions and irreversible blocks/transactions, respectively. In RDMA-enabled message handling mechanism, to avoid the overhead of frequently registering and deregistering RDMA Memory Regions (MRs), memory pool is introduced to manage RDMA MRs. In order to improve the efficiency of allocating fixed-size MRs for sending/receiving control messages and corresponding remote addresses/keys from the MR pool, multiple different types of messages are merged into the same message definition named as *RdmaMessage*. We use the field *RdmaMessageType* in *RdmaMssage* structure to distinguish different types of

messages. The definition of *RdmaMessageType* is shown as Table 3.

Algorithm 1. Async Messages Handling with CQ Event Channel

Require: *eventChannel*, *pd*, *wc*
eventChannel: Completion event channel, to wait for work completions.
pd: RDMA protection domain
wc: Pre-allocated work completions array used for polling
Ensure: *None*

```
1: isRun ⇐ true
2: cq ⇐ Completion queue, to poll on work completions
3: cqContext ⇐ CQ Context
4: while isRun is true do
5:   ibv_get_cq_event(eventChannel, cq, cqContext)
6:   ibv_ack_cq_event(eventChannel, cq, cqContext)
7:   ibv_req_notify_cq(cq, 0)
8:   ne ⇐ ibv_poll_cq(cq, sizeof(wc), wc)
9:   for i in range(0, ne) do
10:    if wc[i].opcode is IBV_WC_RECV_RDMA_WITH _IMM then
11:      rdmaChannel ⇐ wc[i].wr_id
12:      rdmaChannel.Recv()
13:      imm ⇐ wc[i].imm_data
14:      if imm is RDMA_IMM_DATA_ACK then
15:        receiveAckMsg()
16:        sendNextItem()
17:        continue
18:    else if imm <=RDMA_IMM_MAX_REQUEST_ID then
19:        recvBlockTransactionContent()
20:        continue
21:      end if
22:      rdmaMsg ⇐ parseMsgFromMR()
23:      type ⇐ rdmaMsg.msgType
24:      if type is RDMA_MSG_HANDSHAKE_REQ then
25:        handleHandshakeMsg()
26:      else if type is RDMA_MSG_GO_AWAY_RES then
27:        handleGoAwayMsg()
28:      else if type is RDMA_MSG_NOTICE_RES then
29:        handleNoticeMsg()
30:      else if type is RDMA_MSG_SYNC_REQ then
31:        queryLocalIrreversibleBlock()
32:        rdmaWriteIrreversibleBlock()
33:      else if type is RDMA_MSG_VERIFY_CATCHUP then
34:        queryReversibleBlockOrTxn()
35:        rdmaWriteReversibleBlockOrTxn()
36:      end if
37:    else if wc[i].opcode is IBV_WC_RDMA_WRITE then
38:      wrId ⇐ wc[i].wr_id
39:      writeType ⇐ wrId.write_type
40:      if writeType is RDMA_WRITE_ID_MSG then
41:        sendNextItem()
42:     else if writeType is RDMA_WRITE_ID_ BLOCK_WRITE then
43:        popResponse()
44:      end if
45:    end if
46:  end for
47: end while
```

TABLE 3
The Definition of RdmaMessageType

Message Type	Enum Value
time message	RM_TIME
notice message	RM_NOTICE_RESPONSE
go away message	RM_GO_AWAY_RESPONSE
handshake message	RM_HANDSHAKE_REQUEST
request message	RM_CATCHUP_REQUEST
sync request message	RM_SYNC_REQUEST

4.2 RDMA-Enabled Sync Manager

RDMA-enabled sync manager is a very significant component which is responsible for maintaining the RDMA channel list, RDMA message buffers, RDMA completion event channel based asynchronous message handling, blocks/transactions synchronization and so on. The blocks/transactions sync processes are triggered by the handshake negotiation among peer nodes in RDMA-based P2P network. We assume that node A is the receiver and B is the sender. Then the detail of handshake negotiation between node A and B is shown in Fig. 5. If the received handshake request in B is invalid, B will exploit one-sided RDMA WRITE with immediate to send back go away message with corresponding reason to node A. Otherwise, related notice message will be sent back by B with the comparison on block head id, block head number, last irreversible block number between A and B.

Algorithm 1 demonstrates the asynchronous Completion Queue (CQ) handling with RDMA event channel within BoR server. Completion event channel *eventChannel* is employed to deliver the notification about the upcoming work completion in CQs, which can significantly reduce the CPU utilization. The protection domain *pd* is used to guarantee resource isolation (like address handles and memory regions) between different RDMA channels while the input *wc* is a pre-allocated work completions array which can cache multiple work completions in one call *ibv_poll_cq()*.

Once the connection-oriented RDMA channel between two peer BoR nodes is established, the BoR server in each node will start a *PollCQ* thread to asynchronously listen for event notifications attached to upcoming work requests from completion queue. Specifically, within the loop body of the *PollCQ* thread, *ibv_get_cq_event()* method attached to *eventChannel* is invoked and be block until the events of incoming work completions (such as completed send and receive requests) are trigged. Then the events are acknowledged by invoking *ibv_ack_cq_event()* method while the notification for the next completions is requested by *ibv_req_notify_cq()* method. Through calling *ibv_poll_cq()* method, a batch of work completions for RDMA operations have been cached into pre-allocated *wc* array, which can significantly increase the concurrency level of BoR. For each work completion *wc[i]* in *wc* array, different operation codes opcode represent the corresponding RDMA primitives. For instance, the opcode *IBV_WC_RECV_RDMA_WITH_IMM* means the remote RDMA WRITE operations with immediate data has been finished and the payload has been pushed into local receive queue, while the opcode *IBV_WC_RDMA_WRITE* means local RDMA WRITE operation has been finished. When *wc*

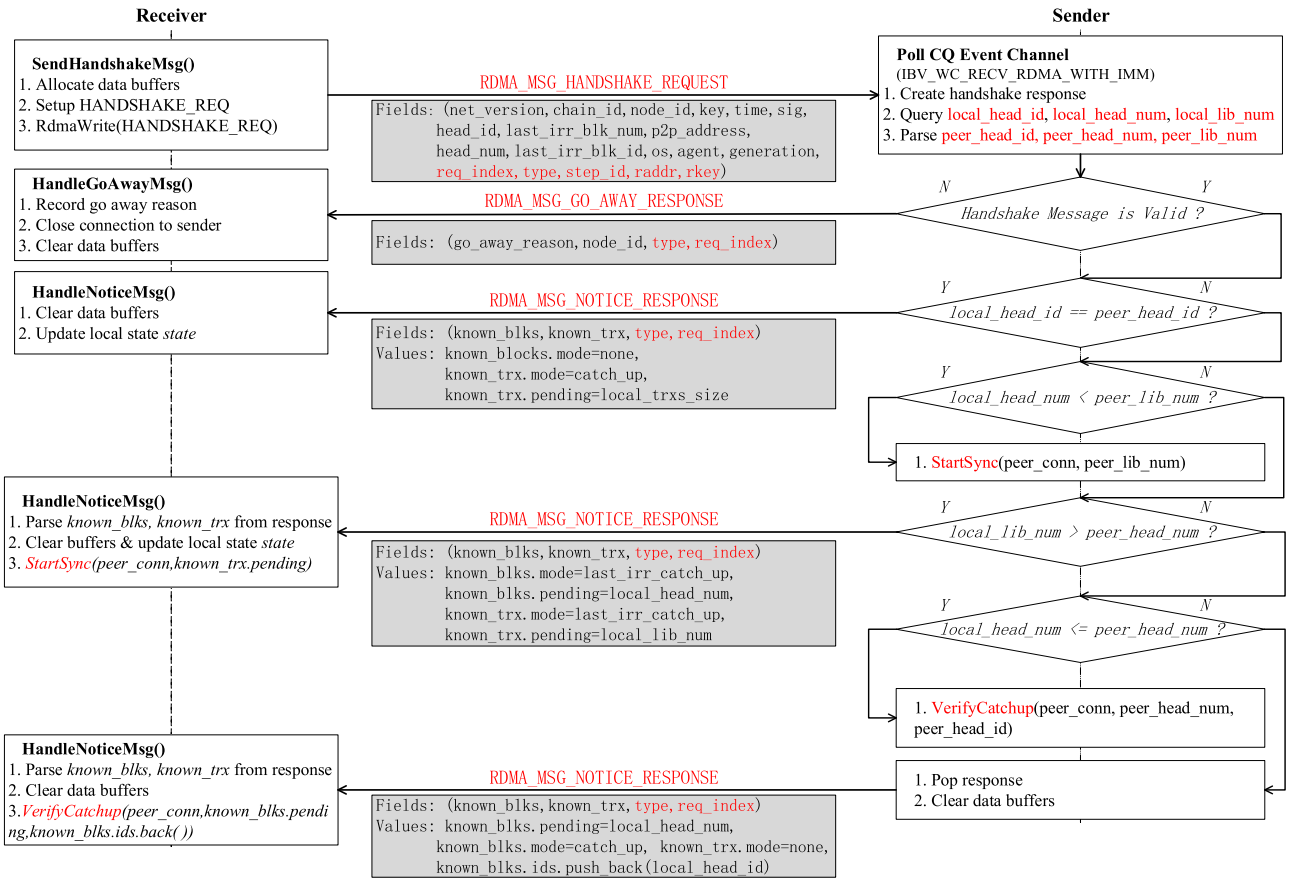


Fig. 5. RDMA-enabled communication protocol for BoR.

[i].opcode is equal to *IBV_WC_RECV_RDMA_WITH_IMM*, immediate data *imm* and related RDMA channel context are extracted. If the *imm* is *RDMA_IMM_DATA_ACK*, receive an ACK message and send the next RDMA message from send queue. If the *imm* is less than or equal to *RDMA_IMM_MAX_REQUEST_ID*, receive one block or transaction data. Otherwise, we parse the control message *rdmaMsg* from the specified receive MR. According to the *type* field in *rdmaMsg*, we can identify different types of control messages like handshake and notice message and take the corresponding message handler. When the *wc[i].opcode* is equal to *IBV_WC_RDMA_WRITE*, the *writeType* flag is extracted and used to check if the RDMA WRITE operation on control or data message is completed.

4.3 RDMA-Enabled New Node Bootstrapping Protocol

When referring to the blocks/transactions synchronization, the sync state can be divided into three types: *catch_up*, *last_irr_catch_up* and *normal*. For blocks, *catch_up* state means sender B packages all local reversible blocks and sends to receiver A; *last_irr_catch_up* state means sender B packages all local irreversible blocks and sends to receiver A; normal state means the irreversible block is searched in sender B depending on the block id vector in receiver A and the found block is sent to A. While for transactions, *catch_up* state means sender B takes the reversible transaction id required by node A as input and sends the found transactions to A; *normal* state means sender B takes the irreversible transaction id required by node A as input and sends the found transactions to A.

When the new node N has just joined the blockchain network, local block head number in node N is less than the last irreversible block head number in its peer node (Node N is in *last_irr_catch_up* state). This drives node N to start synchronizing irreversible blocks from the peer node, as shown in Fig. 7. Once local block header number in node N is greater than the peer latest irreversible head number and less than peer block number in the peer node (Node N is in *catch_up* state), node N will start synchronizing reversible blocks from target peer node until local head number is greater than or equal to the head number in the peer node (Node N is in normal state). The blocks/transactions synchronization process is enabled by one-sided RDMA Write verb with immediate data as shown in Figs. 6 and 7.

4.4 Other Optimization Considerations

This section describes additional optimization considerations for BoR including the use of hybrid RDMA primitives and efficient RDMA resources allocation.

Hybrid RDMA Primitives for BoR. As noted in prior work [47], the use of hybrid RDMA primitives can bring better performance gain for RDMA-based distributed transaction. Learning from this, we leverage two-sided RDMA SEND/RECV operations to exchange the pre-registered TX/RX MRs between interconnected nodes, and employ one-sided RDMA WRITE verb with immediate to boost faster transport for latency-oriented control messages and bandwidth-oriented data messages.

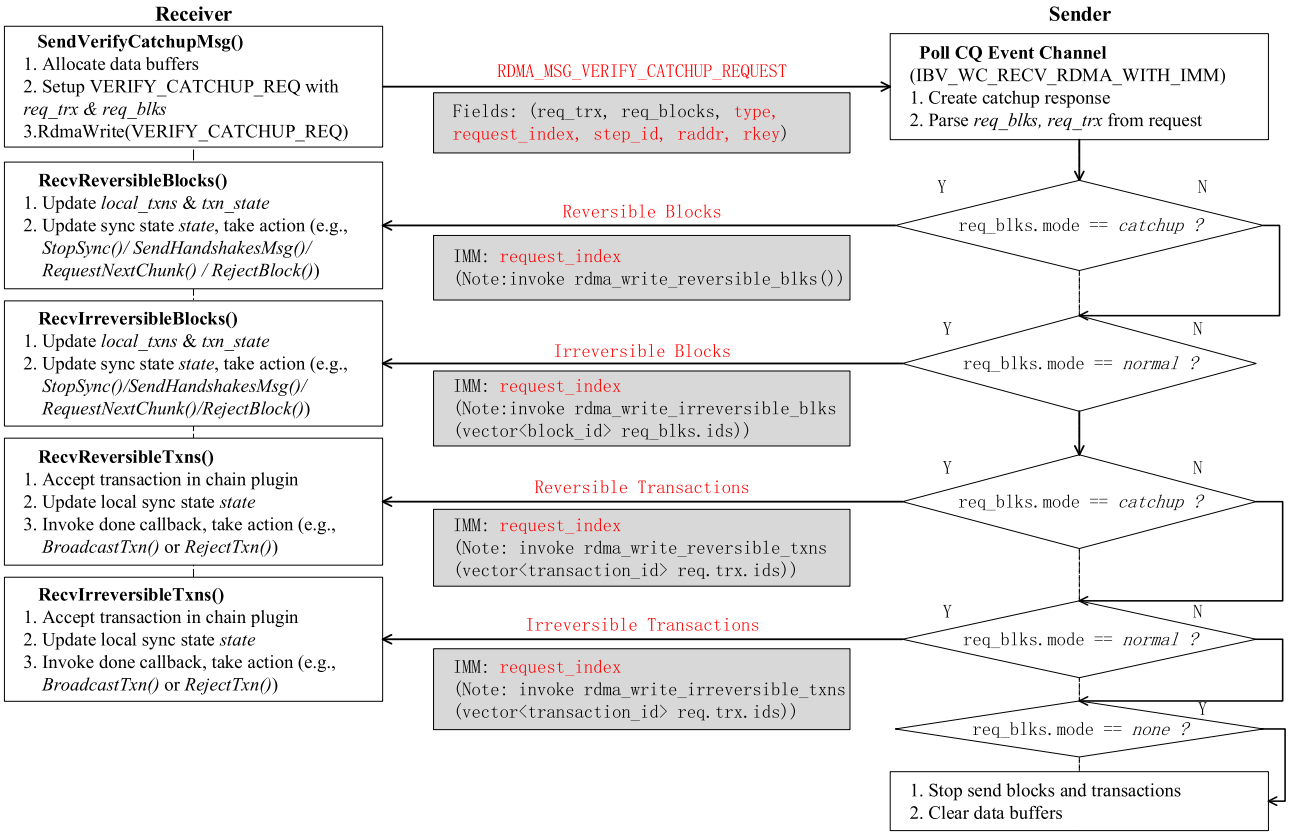


Fig. 6. Rdma-enabled verify_catchup() process for BoR.

RDMA Resources Management. The unjustified management of RDMA resources may result in cache misses within RNIC HCA, which significantly affects the overall performance of BoR. To maximize the performance benefits of RDMA, we adopt different tuning strategies for message buffer, send/receive queues (QP), and completion queue (CQ). For instance, the message buffers of BoR are aligned to cache line size to eliminate read-modify-write. In order to minimize Translation Lookaside Buffer (TLB) misses in RNIC and CPU, BoR employs hugepages with 2 MB page size to allocate message buffers on close NUMA node. Shared receive queue (SRQ) is leveraged to improve the scalability of BoR while single completion queue per thread/core is exploited to avoid polling on several memory segments. In order to achieve lower CPU consumption, BoR uses completion event channel and caches multiple work completions in one call. In addition, pre-registered MR pool is leveraged for the one-sided transfer of BoR to mitigate the delay overhead introduced by registering or deregistering MRs at runtime.

5 Evaluation

5.1 Experiment Setup

Hardware Configuration. We conducted the performance experiment between BoR and plain EoS on a RoCEv2-based

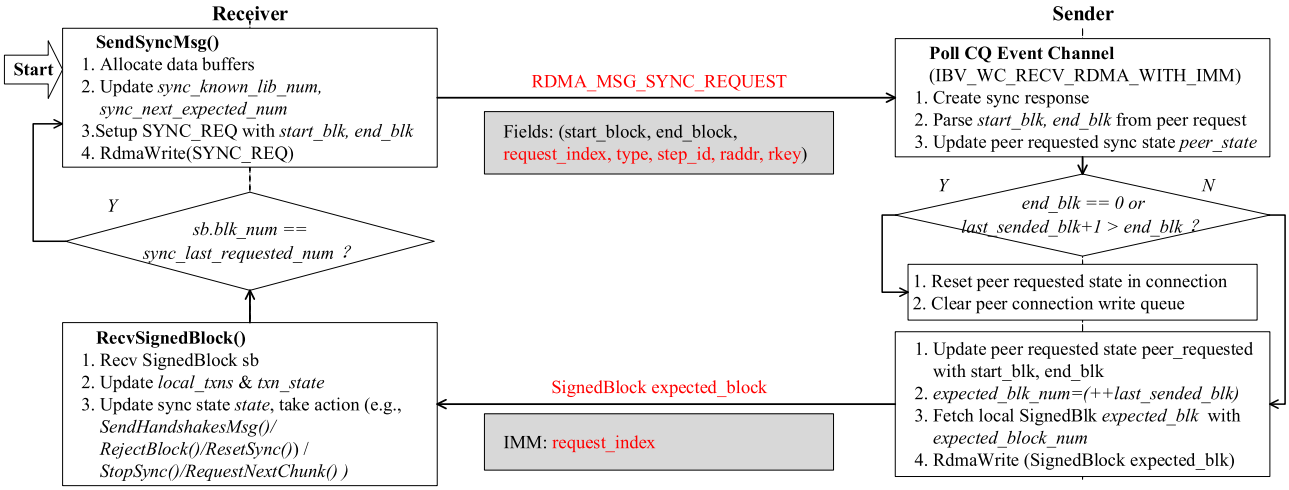


Fig. 7. Rdma-enabled start_sync() process for BoR.

TABLE 4
Configuration of Experiment

Number of Servers	4
Model Name	PowerEdge R730
Vendor	Dell
CPUs	10
CPU cores	48
RAM	128 GB DDR4
OS	CentOS 7.4.1708
RDMA Switch	Mellanox SN2100 (100 Gbps)
RDMA NIC	MCX516A-CDAT ConnectX-5 (100 Gbps)

testbed with 4 servers which inter-connected by a Mellanox SN2100 switch, as shown in Table 4. The DRAM size in each server is 128 GB with the type of DDR4, whose speed is 2,400 MHz. Each server is equipped with two E5-2687Wv4 CPU processors with 12 CPU cores and 3.00 GHz frequency. The L1 cache of each CPU core is 768 KB while the L2 cache is 3,072 KB. 12 cores on one CPU processor share the same 30 MB L3 cache. The RNIC in our servers is a ConnectX-5 MCX516A-CDAT 100 Gbps RoCE NIC and is connected to the SN2100 switch through PCIe 3.0x16. Nodeos and keosdare deployed in server 192.168.2.1 while three producers are deployed in server 192.168.2.2, 192.168.2.3 and 192.168.2.4. The OS release of all nodes is CentOS 7.4.1708 with OFED v4.5-1.0.1.

Experiment Design for Initial Block Synchronization. We respectively deploy BoR and original EoS blockchain network testbeds with three hosts named A, B, C. Then, 5 pieces of historical block datasets with different scales (10, 20, 30, 40, and 50 GB) are obtained from the EosNode test network [48]. We setup 5 cases based on 5 block files. In each case, the corresponding historical block files are replayed on BoR and plain EoS testbeds. We assume server D as a new node for the blockchain network.

5.2 Performance Analysis on Initial Block Sync

For the performance comparison between BoR and plain EoS, we center on two performance metrics including *consuming time* and *CPU usage*. *Consuming time* is defined as how long it takes to complete the initial block sync of a new node in blockchain network. The scale of target historical block datasets is regarded as an independent variable. *CPU usage* indicates how much CPU resources are used during the initial block sync. It contains three dimensions: the time-series distribution of CPU utilization, the average CPU usage with different scales of block datasets, and the peak of CPU usage differences between BoR and plain EoS. Linux performance monitoring tool *sar* is leveraged to collect the CPU utilization once per second in time-series manner.

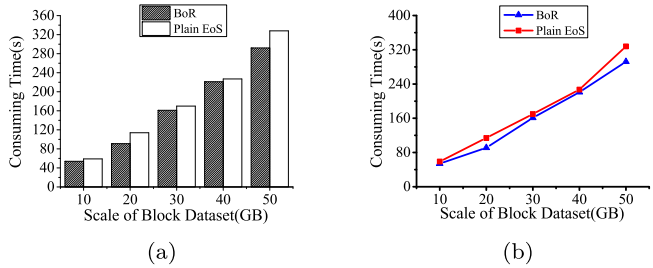


Fig. 9. The consuming time of initial blocks synchronization with different historical blocks datasets.

Average CPU utilization (referred to as *AVG CPU Usage*) is calculated by dividing the accumulated CPU usage value by the overall *consuming time*. The differences called *CPU D-value* refers to the gap of CPU usage at the relatively same time between BoR and plain EoS. The *peak* represents the maximum value of the *CPU D-value*. Each result in the experiment is the average of 12 runs.

Based on performance results, we observed that RDMA-enhanced paradigm can reduce the latency of initial block sync by up to 20.2 percent and decrease the average CPU overhead by 26.433.9 percent, which can boost the high-performance DPoS-based blockchain in BaaS cloud data centers.

5.2.1 Consuming Time for Initial Block Synchronization

Fig. 9 shows the distribution of consuming time under different-scale historical block datasets during the initial sync of new nodes in BoR and traditional EoS network. We can find that BoR with one-sided RDMA WRITE achieves much lower overall consuming time (up to 20.2 percent) than traditional TCP-based EoS with *max latency gap* of *36 s* and *min latency gap* of *5 s*. In the best case, the initialization block of BoR is synchronized. The overall time consumption is nearly *20.2 percent* better than native EoS (as shown in Fig. 9a). Where the consuming time of 50 GB-scale block records of BoR is *114 s* while that of native EoS is *91 s*. We can see that the gap of consuming time is significantly increased once the scale of block records is greater than 40 GB in Fig. 9b. This advantage benefits from less memory copies, higher message rate and significantly reduced CPU context switching in BoR with the use of hybrid RDMA primitives.

5.2.2 CPU Usage for Initial Block Synchronization

The time-series distribution of CPU utilization and average cpu usage are demonstrated in Figs. 8 and 10. Compared to plain EoS, BoR significantly reduces the CPU overhead. In terms of the distribution of CPU usage, Fig. 8 shows that the trend line of CPU usage occupied by BoR is generally below

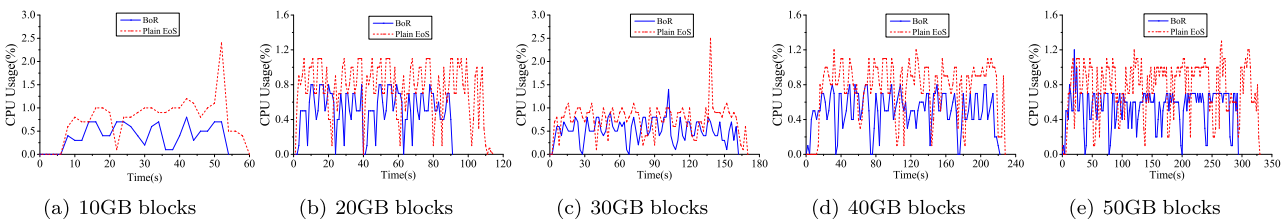


Fig. 8. The CPU utilization of initial blocks synchronization with different historical blocks datasets.

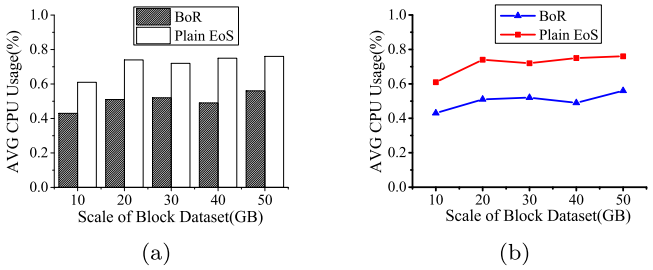


Fig. 10. The distribution of average CPU usage between BoR and plain EoS with the increasing scale of block datasets.

that of plain EoS under different block volumes. From Fig. 10a we can see that BoR achieves *26.433.9 percent* lower CPU utilization than plain EoS. In the best case, the average CPU usage of BoR is *0.49 percent* while that of EoS is *0.75 percent*. Moreover, with increasing scale of block records, the average CPU utilization of RDMA-driven BoR is more stable than TCP-based EoS (as shown in Fig. 10b, this can bring stable BaaS quality. The performance improvement achieved by BoR is due to the unique features of kernel bypassing and network protocol stack offloading in RDMA-capable network. The pre-allocated MR pool and asynchronous RDMA event channel also contribute to the lower CPU overhead of BoR.

6 Related Work

Blockchain Performance Analysis and Optimization. In recent years, many research works focus on how to optimize the performance of blockchain. In [49], the author first reconstructs the architecture of Hyperledger and Ethereum and then evaluates the modifiability, security and performance of these architectures. The author announces that these three metrics are the most important in blockchain evaluation. In [8], the authors first prove that the blockchain consensus mechanism satisfies a strong form of consistency and activity in asynchronous networks, with a priority-limited defensive delay, allowing adaptive damage and new player generation in formal models. In addition, the authors make a contribution that an abstract blockchain protocol has been proposed to identify the security. In [8], [9], [10], many other researches pay attention to performance analysis and gain great achievements. However, none of these realizes that the network performance is becoming more significant with the fast development of blockchain technology. Therefore, we propose BoR which focuses on the network layer optimization of blockchain systems with RDMA-driven paradigm.

Consensus Protocol. Since the consensus is one of the most important part of blockchain technology, many researches are interested in consensus protocol. Bitcoin-NG [14] proposes a new blockchain protocol as a next generation protocol on Bitcoin. In [15], the author proposes a new framework to monitor the PoW-based blockchain, as well as introduces various parameters of consensus and network. Real-world constraints (such as network propagation, different block sizes, block generation intervals, and information dissemination mechanisms) are taken into account when designing the blockchain framework. In [16], the author proposes a Proof-of-Trust blockchain named PoT to reduce the energy expenditure of PoW. With PoT, a peer node can do less work as long as it is proved to be more creditable. PoT is a typical solution for consensus protocol optimization based on a trust graph. With all above works, the previous inefficient consensus mechanism has been greatly improved. However, few research commits to improving the network performance in blockchain system. That's mainly because that with the underlying network an inefficient consensus protocol is not a bottleneck while the network transmission with an efficient consensus protocol becomes a potential performance bottleneck. Hence, kernel-bypassing RDMA network is employed in BoR to bridge the performance gap for a faster and more efficient blockchain.

RDMA-Driven Applications. RDMA-enabled network is widely deployed in data centers recently due to its high throughput, low latency and low CPU overhead. In [28], [50], an RDMA-driven B-tree store architecture is proposed to reduce the network overhead and improve the performance. In [20], Wukong provides a fast and concurrent solution on graph exploration using RDMA communication. It uses different RDMA primitives (i.e., RDMA READ/WRITE) based on different-scale RDF query. In [31], [51], [52], HydraDB, Nessie and InnerCache all propose an efficient key-value system based on RDMA. HydraDB provides an access service in a reliable way for distributed applications under high throughput and low latency. Nessie is an RDMA-enabled key-value system that provides a none-server solution in servicing requests and decouples the index and the data. InnerCache is another RDMA-based in-memory key-value store that provides a tactful cache mechanism. It leverages two-sided communication to improve the system performance. Other key-value researches [27], [32], [33] also achieve good results. In addition, RDMA-driven system optimization is also a research hotspot [30], [53], [54]. However, no research commits to designing an RDMA-enabled blockchain application so far. Therefore, BoR is the first attempt to accelerate the distributed blockchain using kernel-bypassing RDMA network.

7 Conclusion & Future Work

In this paper, we first analyze the bottleneck of the existing blockchain systems for emerging BaaS cloud. According to our investigation, consensus is the bottleneck of previous blockchain, such as Bitcoin and Ethereum. Fortunately, the consensus protocol has been improved significantly in recent years (i.e., EoS and its DPoS). However, with the faster consensus protocols, the bottleneck in blocks/transactions transmission over traditional TCP/IP network is gradually revealed. Therefore, we propose the design and implementation of BoR, a first RDMA-driven blockchain platform that leverages RDMA communication primitives to achieve high throughput and low latency without CPU intervention for the initial blocks synchronization and blocks/transactions propagation under high concurrency in BaaS cloud. Kernel-bypass and zero-copy technology in RDMA-enabled network are exploited by BoR to efficiently achieve higher performance. RDMA completion event channel is also leveraged to receive control message or blocks/transactions asynchronously while one-sided RDMA write with immediate data is employed to send control and data messages. To mitigate the high CPU overhead and long consuming time, BoR leverages RDMA in two paradigms: one is the initial synchronization when a new node joins the P2P network, and the other is the block broadcast when a producer generates a new block. Our evaluation shows that BoR

significantly outperforms the original EoS blockchain system by a wide margin. Compared with the state-of-the-art EoS blockchain, BoR obviously reduces the latency by up to 20.2 percent, and decreases the CPU utilization by 26.433.9 percent when a new node joins the blockchain network.

Our future work may extend BoR to support multi-path RDMA transport for initial blocks synchronization in BaaS cloud data center. We also plan to leverage the Reliable Datagram transport for RDMA communication to increase the scalability of BoR-based BaaS cloud. Apart from this, we aim to combine the emerging high-speed Non-Volatile Memory (NVM) with modern RDMA hardware to accelerate the persistence of metadata and blocks in BoR.

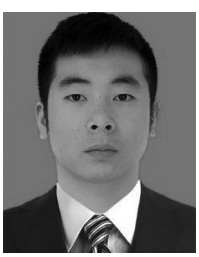
Acknowledgments

We would like to thank anonymous reviewers for their insightful feedback. We sincerely thank Qingchun Song, Gil Blooh and Pengzhi Zhu, researchers in Mellanox, for their helpful discussions and technology support. The work of this paper is supported by National Natural Science Foundation of China under Grant (No. 61873309, No. 61572137, and No.61728202), and Shanghai Innovation Action Plan Project under Grant (No. 19510710500, No. 18510760200, and No. 18510732000), and 2017 Research Projects of Shanghai Science and Technology Commission under Grant No. 17DZ1101000.

References

[1] Blockchain on aws, 2018. [Online]. Available: https://aws.amazon.com/blockchain/, Amazon Web Services, Inc
[2] Azure blockchain service, 2018. [Online]. Available: https://azure.microsoft.com/en-us/services/blockchain-service/, Microsoft Azure
[3] IBM blockchain platform, 2018. [Online]. Available: https://www.ibm.com/cloud/blockchain-platform, IBM Cloud Technologies
[4] Bitcoin-wikipedia, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Bitcoin
[5] Ethereum, 2015. [Online]. Available: https://www.ethereum.org/, Ethereum, Inc
[6] E. Androulaki et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, 2018, Art. no. 30.
[7] DanielLarimer, "EOSIO | Blockchain software architecture." 2018. [Online]. Available: https://eos.io/
[8] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2017, pp. 643–673.
[9] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *Proc. 40th Int. Conf. Softw. Eng.: Softw. Eng. Pract.*, 2018, pp. 134–143.
[10] N. Papadis, S. Borst, A. Walid, M. Grissa, and L. Tassiulas, "Stochastic models and wide-area network measurements for blockchain design and analysis," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 2546–2554.
[11] Z. Xu, S. Han, and L. Chen, "CUB, a consensus unit-based storage scheme for blockchain system," in *Proc. IEEE 34th Int. Conf. Data Eng.*, 2018, pp. 173–184.
[12] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric)," in *Proc. IEEE 36th Symp. Reliable Distrib. Syst.*, 2017, pp. 253–255.
[13] D. Larimer, "Delegated proof-of-stake (DPoS)," *Bitshare whitepaper*, 2014.
[14] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. USENIX Conf. Netw. Syst. Des. Implementation*, 2016, pp. 45–59.
[15] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 3–16.
[16] L. Bahri and S. Girdzijauskas, "When trust saves enegry: A reference framework for proof-of-trust (PoT) blockchains," in *Proc. Web Conf.*, 2018, pp. 1165–1169.
[17] Bitcoin.com | bitcoin news and technology source, 2019. [Online]. Available: https://www.bitcoin.com/
[18] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data*, 2017, pp. 557–564.
[19] M. Beck and M. Kagan, "Performance evaluation of the RDMA over ethernet (RoCE) standard in enterprise data centers infrastructure," in *Proc. 3rd Workshop Data Center-Converged Virtual Ethernet Switching*, 2011, pp. 9–15.
[20] J. Shi, Y. Yao, R. Chen, H. Chen, and F. Li, "Fast and concurrent RDF queries with RDMA-based distributed graph exploration," in *Proc. USENIX Conf. Operating Syst. Design Implementation*, 2016, pp. 317–332.
[21] Z. Wu, Z. Lu, P. C. Hung, S.-C. Huang, Y. Tong, and Z. Wang, "QaMeC: A QoS-driven IoVs application optimizing deployment scheme in multimedia edge clouds," *Future Gener. Comput. Syst.*, vol. 92, pp. 17–28, 2019.
[22] N. S. Islam, M. Wasi-ur Rahman, X. Lu, and D. K. Panda, "High performance design for HDFS with byte-addressability of NVM and RDMA," in *Proc. Int. Conf. Supercomput.*, 2016, Art. no. 8.
[23] X. Chen et al., "iDiSC: A new approach to IoT-data-intensive service components deployment in edge-cloud-hybrid system," *IEEE Access*, vol. 7, pp. 59 172–59 184, 2019.
[24] B. K. C. Zhang et al., "A deep reinforcement learning based approach for cost- and energy-aware multi-flow mobile data offloading," in *IEICE Trans. Commun.*, vol. E101.B(7), pp. 1625–1634, 2019.
[25] P. Wu, Z. Lu, Q. Zhou, Z. Lei, X. Li, M. Qiu, and P. C. Hung, "Bigdata logs analysis based on seq2seq networks for cognitive Internet of Things," *Future Gener. Comput. Syst.*, vol. 90, pp. 477–488, 2019.
[26] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "RDMA over commodity ethernet at scale," in *Proc. ACM SIGCOMM*, 2016, pp. 202–215.
[27] N. S. Islam, D. Shankar, X. Lu, M. Wasi-Ur-Rahman, and D. K. Panda, "Accelerating I/O performance of big data analytics on HPC clusters through RDMA-based key-value store," in *Proc. 44th Int. Conf. Parallel Process.*, 2015, pp. 280–289.
[28] F. Li, S. Das, M. Syamala, and V. R. Narasayya, "Accelerating relational databases by leveraging remote memory and RDMA," in *Proc. Int. Conf. Manage. Data*, 2016, pp. 355–370.
[29] B. Huang, L. Jin, Z. Lu, M. Yan, J. Wu, P. C. Hung, and Q. Tang, "RDMA-driven MongoDB: An approach of RDMA enhanced NoSQL paradigm for large-scale data processing," *Inf. Sci.*, vol. 502, pp. 376–393, 2019.
[30] Y. Lu, J. Shu, Y. Chen, and T. Li, "Octopus: An RDMA-enabled distributed persistent memory file system," in *Proc. USENIX Annu. Tech. Conf.*, 2017, pp. 773–785.
[31] B. Cassell, T. Szepesi, B. Wong, T. Brecht, J. Ma, and X. Liu, "Nessie: A decoupled, client-driven key-value store using RDMA," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3537–3552, Dec. 2017.
[32] A. K. M. Kaminsky and D. G. Andersen, "Using RDMA efficiently for key-value services," in *Proc. ACM SIGCOMM*, 2014, pp. 295–306.
[33] D. Shankar, X. Lu, N. Islam, M. Wasi-Ur-Rahman, and D. K. Panda, "High-performance hybrid key-value store on modern clusters with RDMA interconnects and SSDs: Non-blocking extensions, designs, and benefits," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2016, pp. 393–402.
[34] M. Wasi-ur Rahman, X. Lu, N. S. Islam, R. Rajachandrasekar, and D. K. Panda, "High-performance design of YARN MapReduce on modern HPC clusters with lustre and RDMA," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2015, pp. 291–300.
[35] M. Ali, J. C. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *Proc. USENIX Annu. Tech. Conf.*, 2016, pp. 181–194.
[36] M. Barborak, A. Dahbura, and M. Malek, "The consensus problem in fault-tolerant computing," *ACM Comput. Surv.*, vol. 25, no. 2, pp. 171–220, 1993.
[37] L. O. Taylor, M. McKee, S. K. Laury, and R. G. Cummings, "Induced-value tests of the referendum voting mechanism," *Econ. Lett.*, vol. 71, no. 1, pp. 61–65, 2001.
[38] J. Gattermayer and P. Tvrdik, "Blockchain-based multi-level scoring system for P2P clusters," in *Proc. 46th Int. Conf. Parallel Process. Workshops*, 2017, pp. 301–308.

[39] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 5, pp. 840–852, Sep./Oct. 2018.
[40] I.-C. Lin and T.-C. Liao, "A survey of blockchain security issues and challenges," *IJ Netw. Security*, vol. 19, no. 5, pp. 653–659, 2017.
[41] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[42] Hyperledger— Open source blockchain technologies, 2018. [Online]. Available: https://www.hyperledger.org/, The Linux Foundation
[43] A. Dua, N. Bulusu, W.-C. Feng, and W. Hu, "Towards trustworthy participatory sensing," in *Proc. 4th USENIX Conf. Hot Topics Security*, 2009, pp. 8–8.
[44] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 792–800.
[45] A. Dragojević, D. Narayanan, M. Castro, and O. Hodson, "FaRM: Fast remote memory," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implementation*, 2014, pp. 401–414.
[46] H. Chen et al., "Fast in-memory transaction processing using RDMA and HTM," *ACM Trans. Comput. Syst.*, vol. 35, no. 1, 2017, Art. no. 3.
[47] X. Wei, Z. Dong, R. Chen, and H. Chen, "Deconstructing RDMA-enabled distributed transactions: Hybrid is better!" in *Proc. 13th USENIX Symp. Operating Syst. Design Implementation*, 2018, pp. 233–251.
[48] B. Matrix, "EOS node tool." 2018. [Online]. Available: https://eosnode.tools/blocks
[49] J. Kim, S. Kang, H. Ahn, C. Keum, and C.-G. Lee, "Architecture reconstruction and evaluation of blockchain open source platform," in *Proc. 40th Int. Conf. Softw. Eng.: Companion Proc.*, 2018, pp. 185–186.
[50] C. M. K. Montgomery, L. Nelson, S. Sen, and J. Li, "Balancing CPU and network in the cell distributed B-Tree store," in *Proc. USENIX Annu. Tech. Conf.*, 2016, Art. no. 451.
[51] Y. Wang et al., "HydraDB: A resilient RDMA-driven key-value middleware for in-memory cluster computing," in *Proc. SC-Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2015, pp. 1–11.
[52] M. Yang, S. Yu, R. Yu, N. Xiao, F. Liu, and W. Chen, "InnerCache: A tactful cache mechanism for RDMA-based key-value store," in *Proc. IEEE Int. Conf. Web Services*, 2016, pp. 646–649.
[53] A. K. M. Kaminsky and D. G. Andersen, "Design guidelines for high performance RDMA systems," in *Proc. USENIX Annu. Tech. Conf.*, 2016, Art. no. 437.
[54] F. Liu, L. Yin, and S. Blanas, "Design and evaluation of an RDMA-aware data shuffling operator for parallel database systems," in *Proc. 12th Eur. Conf. Comput. Syst.*, 2017, pp. 48–63.



Bobo Huang is working toward the PhD degree in the School of Computer Science, Fudan University, Shanghai, China. His research interests include cloud computing, distributed system, data center networks, big data system, network functions virtualization, and network management driven by big data. He is a student member of the IEEE.

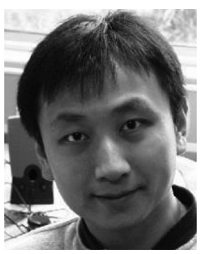
Li Jin is working toward the master's degree in the School of Computer Science, Fudan University, Shanghai, China. His research interests include networking, data center networks, cloud computing, big data distributed system, and network architecture.

Zhihui Lu received the PhD degree in computer science from Fudan University, Shanghai, China in 2004. He is an associate professor with the School of Computer Science, Fudan University. His research interests include big data architecture, cloud computing and service computing technology, edge computing, and blockchain distributed system. He is a member of the IEEE.

Xin Zhou is working toward the master's degree in the School of Computer Science, Fudan University, Shanghai, China. His research interests include data center network, operating systems, cloud computing, and big data distributed system.

Jie Wu received the PhD degree in computer science from Fudan University, Shanghai, China in 2008. He is a professor with the School of Computer Science, Fudan University. His research interests include internet technology, big data architecture, service computing, cloud computing, software defined network, and blockchain technology. He is a member of the IEEE.

Qifeng Tang received the graduate degree from EMBA, Beijing University of Aeronautics and Astronautics, Beijing, China. He is currently a CEO of Shanghai Data Exchange Corporation, chairman of the Board of Directors of National Engineering Laboratory, China Big Data Circulation and Transaction Technology, and chairman of BDU of China Enterprise Big Data Alliance. He has been engaged in the research of the big data application for a long time. He is a representative figure in the field of big data application in China and has many invention patents. He led the Shanghai Data Exchange Center to develop a data trading platform, to promote the interconnection and cooperation of Data Trading Institutions in the Pan-Yangtze River Delta region and even in the whole country, and to promote the circulation of commercial data assets.



Patrick C. K. Hung is a professor with the Faculty of Business and Information Technology, University of Ontario Institute of Technology, Canada. He has been working with Boeing Research and Technology in Seattle, Washington on aviation services-related research projects. He owns a U.S. patent on Mobile Network Dynamic Workflow Exception Handling System with Boeing. His research interests include services computing, big data architecture, business process, and security. He is a member of the IEEE.