Blockchain Machine

Accelerating Validation Bottlenecks in Hyperledger Fabric

Haris Javaid

27 July 2021

XILINX

Agenda

- Xilinx at a Glance
- Hyperledger Fabric: Overview
- Hyperledger Fabric: Performance Bottlenecks
- Blockchain Machine: Networking and Compute Accelerators
- Summarized Results
- Concluding Remarks

- Acknowledgements: Ji Yang, Nathania Santoso, Mohit Upadhyay, S Mohan, Chengchen Hu, Gordon Brebner

XILINX

Xilinx at a Glance

- Inventor of the FPGA (Field Programmable Gate Array)
- Over 4,000 patents held

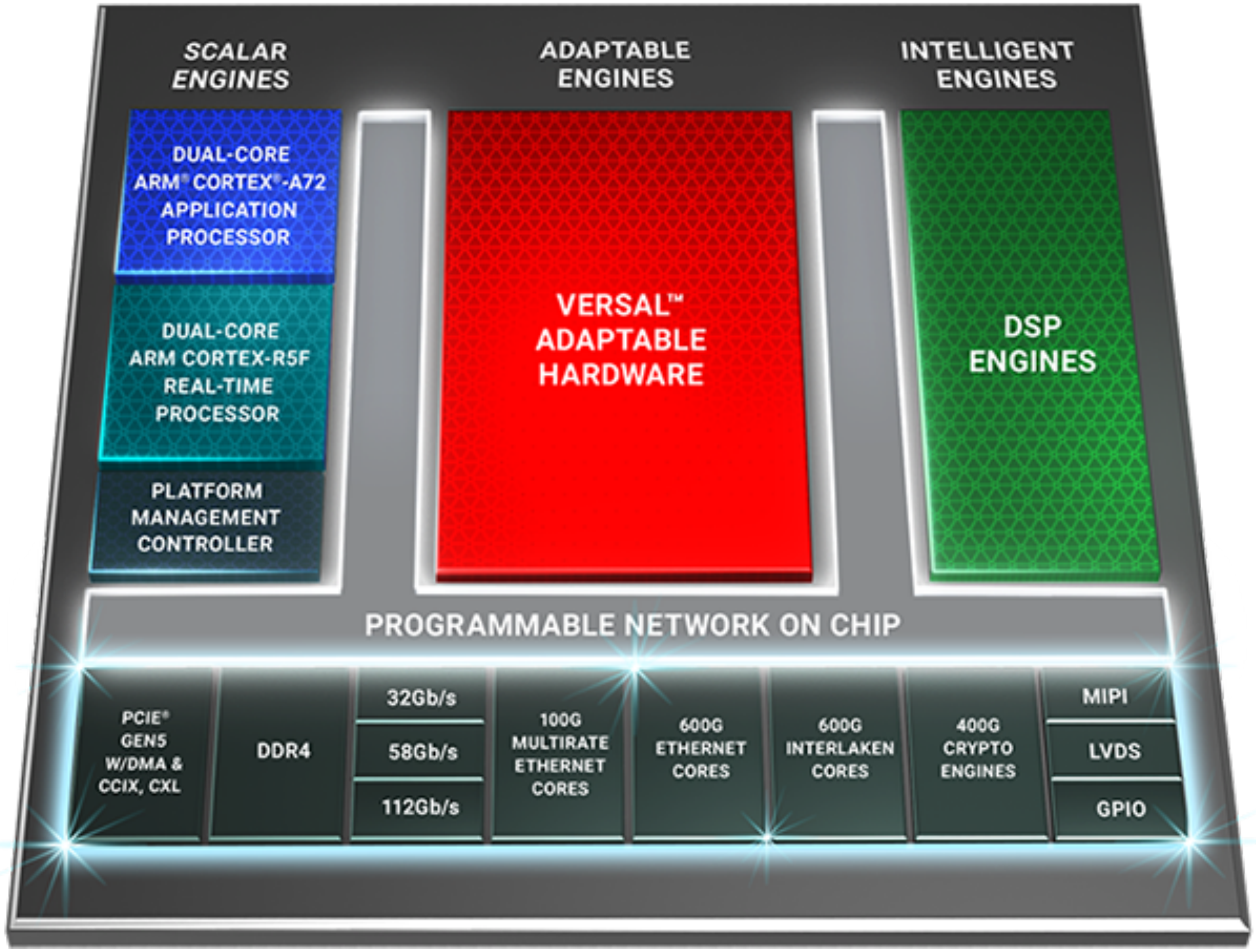
- Founded: 1984; Public: 1990; NASDAQ: XLNX

- Corporate headquarters in San Jose, USA
- Regional headquarters in Ireland and Singapore

- Around 4,900 employees worldwide
- More than 20,000 customers worldwide

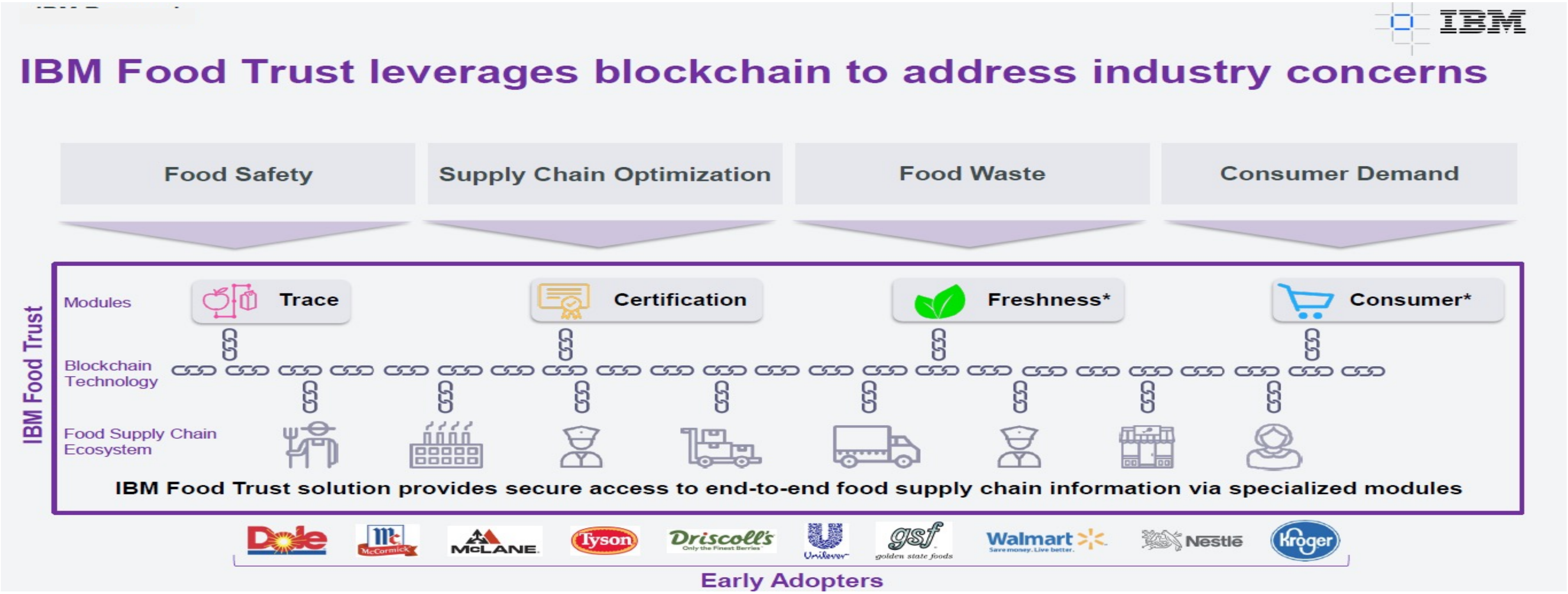
XILINX

Xilinx Programmable Acceleration Platform



XILINX

Blockchain Application Scalability



> **Example scaling up:**
1. One retailer, selected products
2. One retailer, all products
3. Major retailers, all products
4. All retailers, all products

> Estimated transactions per second:
1. 40,000
2. 400,000
3. 2,500,000
4. 250,000,000

Software-only wall: 400K, based on today's best plus 10x future optimization

Another 10x-100x needed from acceleration

Why Hyperledger Fabric?

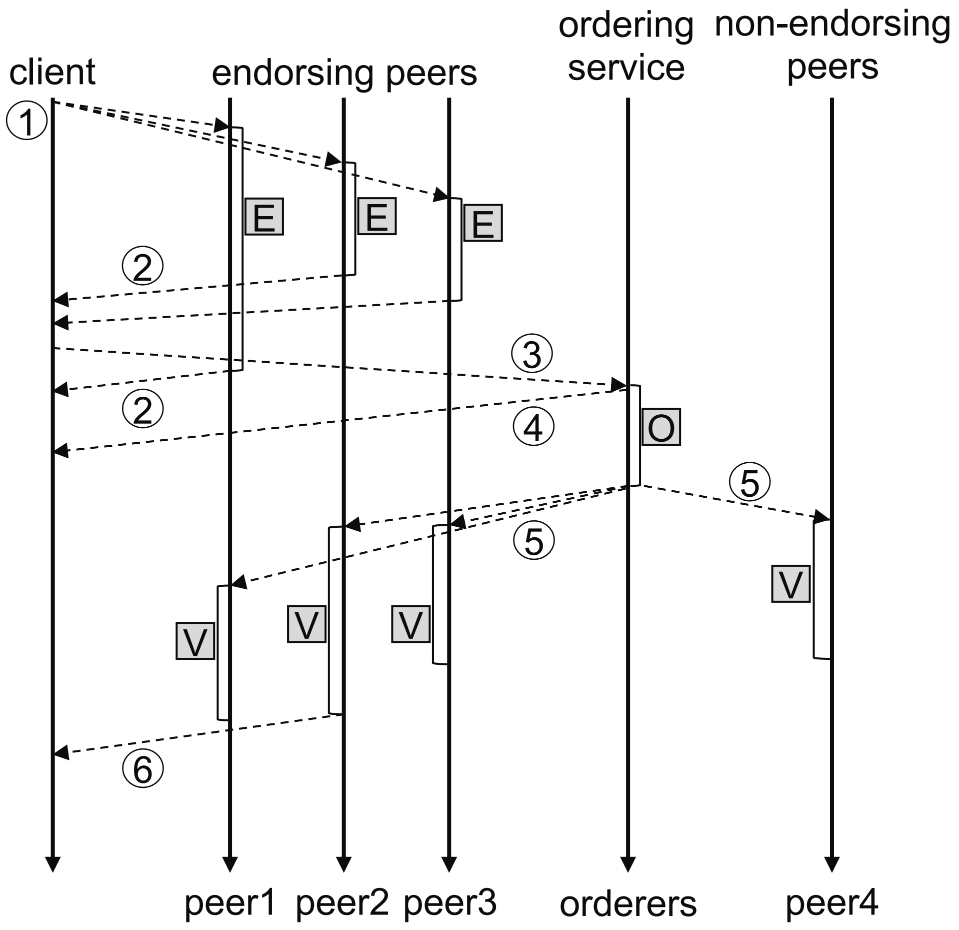
- Enterprise-grade implementation of a permissioned blockchain
  - Only authorized nodes can be part of or interact with the blockchain network
  - Consensus is delegated to a few nodes (unlike power-hungry proof-of-work mining in public blockchains such as Bitcoin)
  - Not associated with a cryptocurrency

- Open-sourced under Linux Foundation
  - IBM contributed the initial implementation, and sells a blockchain product based on this open-source code

- Enterprise applications in a wide range of industries
  - Banking, finance, supply chain, transportation, telecom, etc.

- Most widely used permissioned blockchain platform under the Hyperledger umbrella
  - Other blockchains include Sawtooth (Intel), Besu (Consensys), etc.

Hyperledger Fabric Transactions



1. Client invokes a transaction (by sending it to endorsers)

2. Peer sends endorsement (ECDSA signature, and database read/write sets) to the client, which collects endorsements from relevant peers

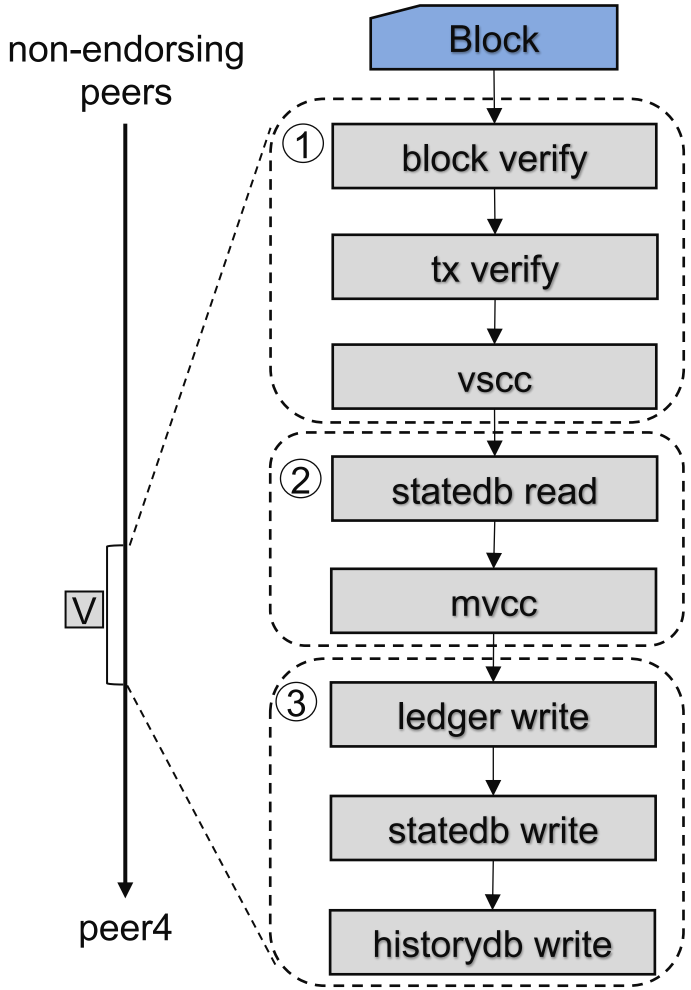
3. Client submits the transaction to ordering service (for being ordered and included in a block)

4. Orderer sends confirmation to client

5. Orderer broadcasts the block to peers (after a timeout or block has reached its limit)

6. Each peer validates the block and sends commit notification to client

Hyperledger Fabric (Validator)



Validator peer (V):

- Verifies the orderer signature
- Verifies each transaction's syntax and creator signature
- Runs validation system chaincode (vscc) on each transaction
  - Validates each endorsement of the transaction
  - Ensures endorsements satisfy the endorsement policy

- Reads from state database to create read sets of all the transactions
- Runs multi-version concurrency control (mvcc) to check read-write conflicts across the transactions
  - Compares current read sets with those from the endorsement phase

- Commits block to ledger
- Updates state database with the write sets of valid transactions

Validation phase is one of the major bottlenecks [1, 2, 3]

[1] P. Thakkar, S. Nathan, and B. Vishwanathan, "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform," in *MASCOTS*, 2018.
[2] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second," in *ICBC*, 2019.
[3] P. Thakkar and S. Nathan. 2021. "Scaling Hyperledger Fabric Using Pipelined Execution and Sparse Peers," arXiv:2003.05113, 2020.

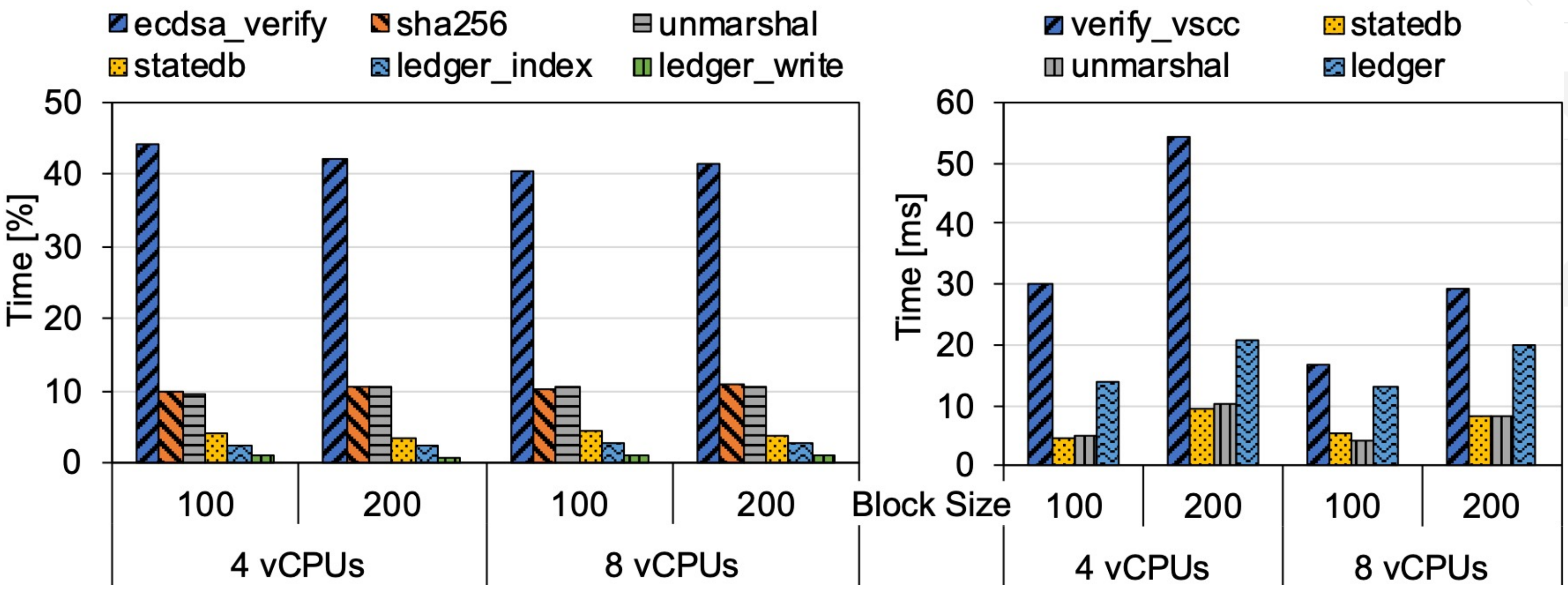
Bottlenecks in Validator Peer

> Retrieving block and transaction data involves unmarshalling of many protocol buffers

> Validation of a block involves verification of many ECDSA signatures, which becomes the critical path

> State database accesses are typically slow

> Ledger write takes longer for larger block sizes, and is an I/O bound operation



XILINX

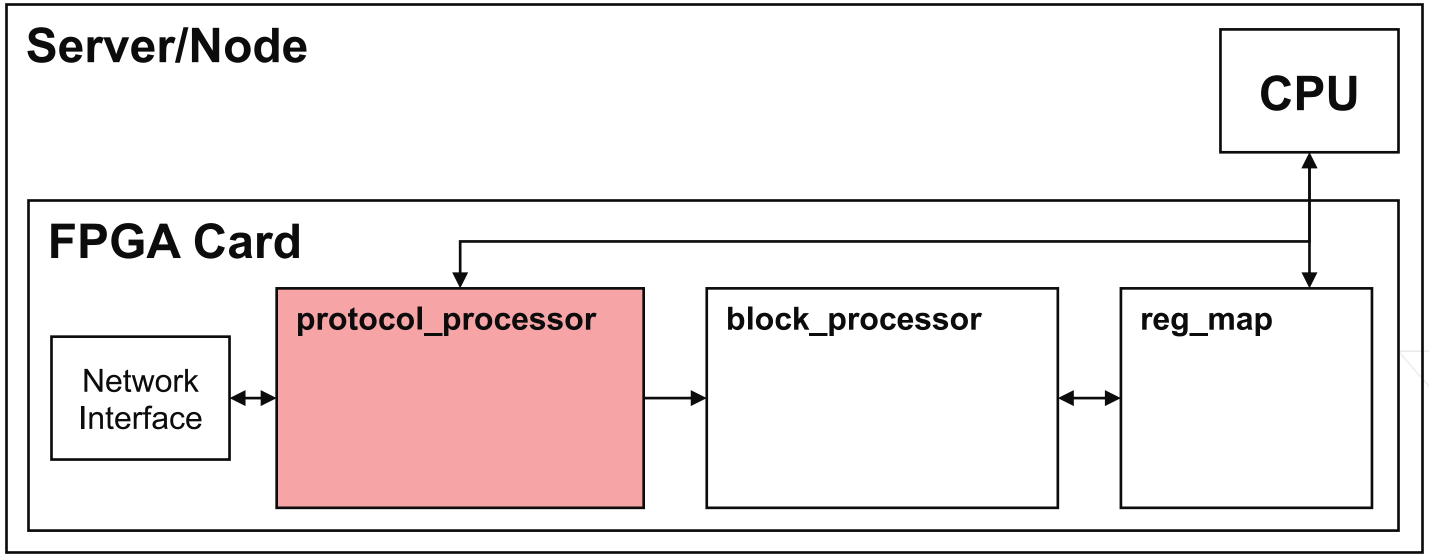
Blockchain Machine (1)

- **Goals**
  - Build FPGA based hardware accelerators for Hyperledger Fabric
  - Hardware/software co-design setup
  - Improve performance metrics: transaction throughput, confirmation time, etc.

- **Current design**
  - Implements validator peer (validation phase of Fabric) on a network-attached Xilinx Alveo card
    - Hardware-friendly protocol to send blocks
    - Block/transaction data is retrieved in hardware directly from the network interface
    - Configurable and efficient block-level and transaction-level pipeline in hardware
  - Integrates with Fabric v1.4 LTS

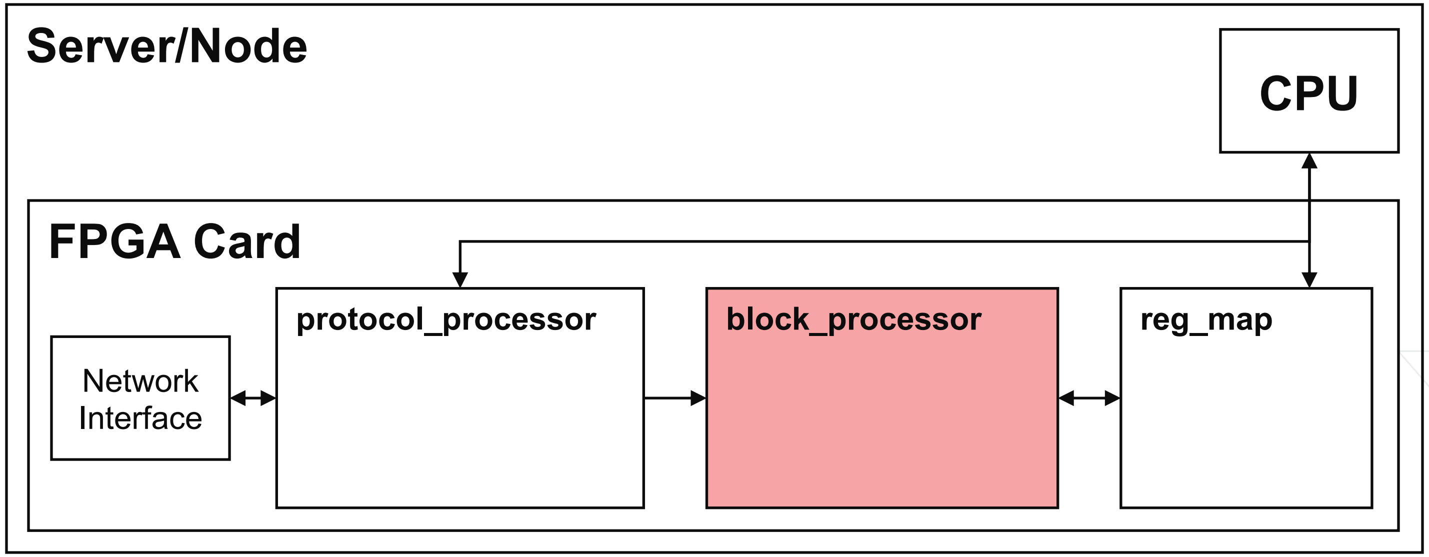
XILINX

Blockchain Machine (2)



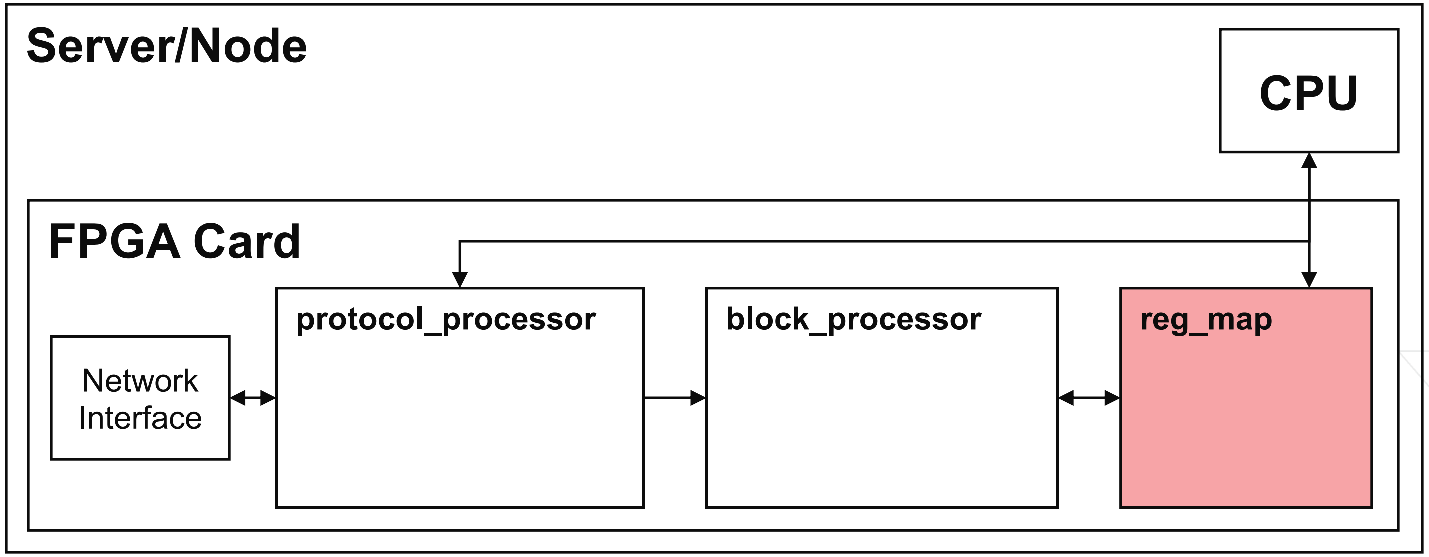
- All the network traffic goes through the FPGA card with integrated network interface
  - CPU NIC is not used
- The protocol_processor
  - filters blockchain machine related packets, and forwards relevant data to block_processor
  - forwards other packets (not intended for blockchain machine) to/from the CPU
  - Block data streams in from the FPGA card network interface to block_processor

Blockchain Machine (3)



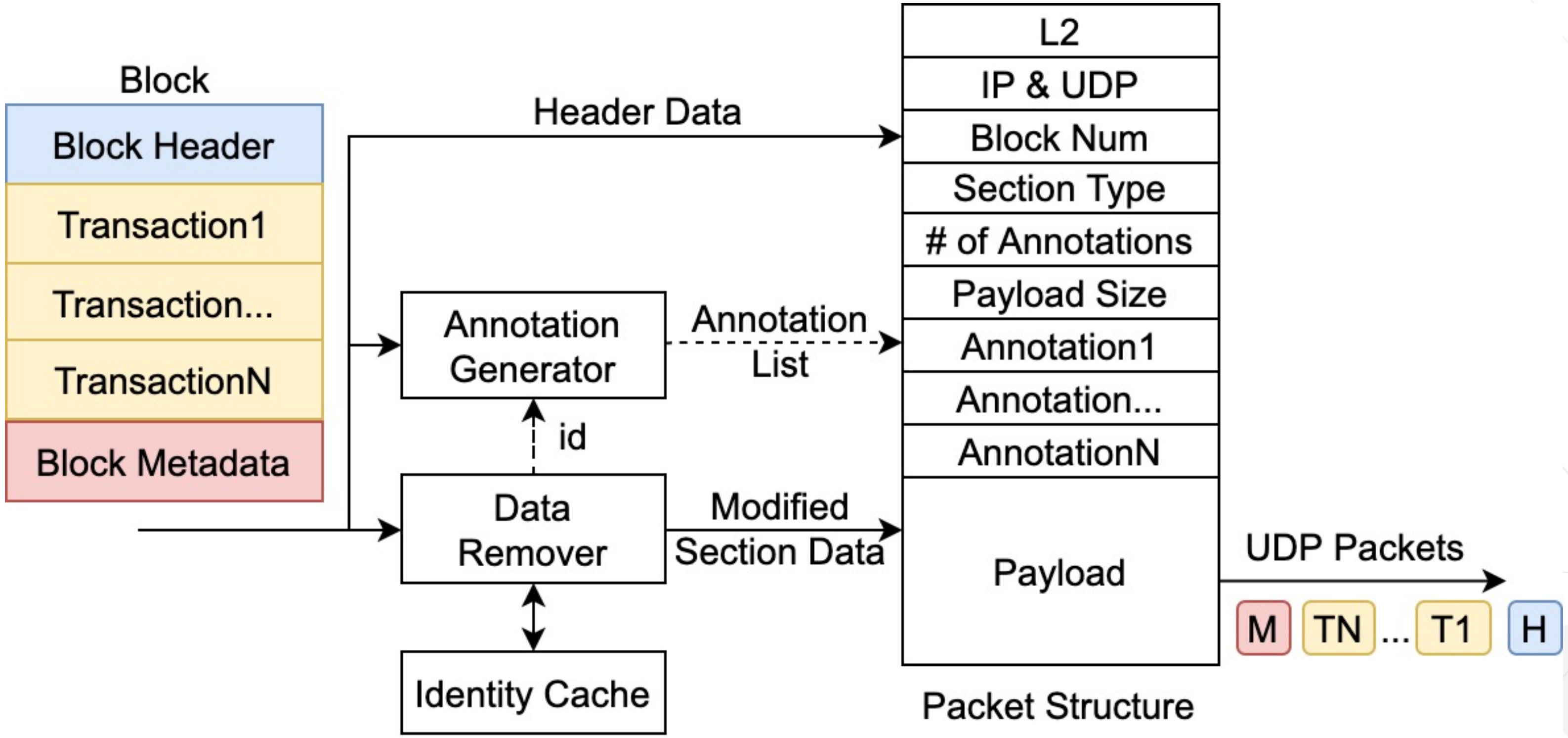
- The block_processor contains
  - Block-level and transaction-level pipeline to process the block and its transactions
  - Parallel-pipelined architecture for high throughput
  - ECDSA verification engines, key-value based database, control/status registers, etc.

Blockchain Machine (4)



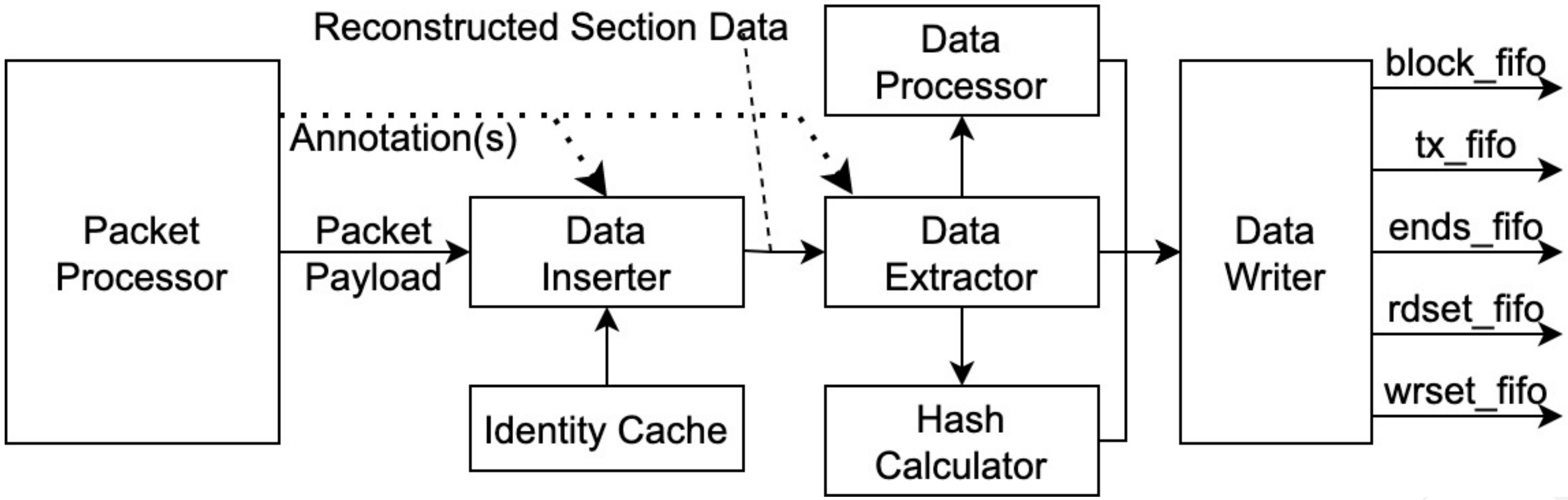
- The reg_map serves as the hardware/software interface
  - Fabric peer software on CPU gathers block validation results using an API
  - Commits the block to ledger just like the software-only validator peer

Hardware-friendly Protocol



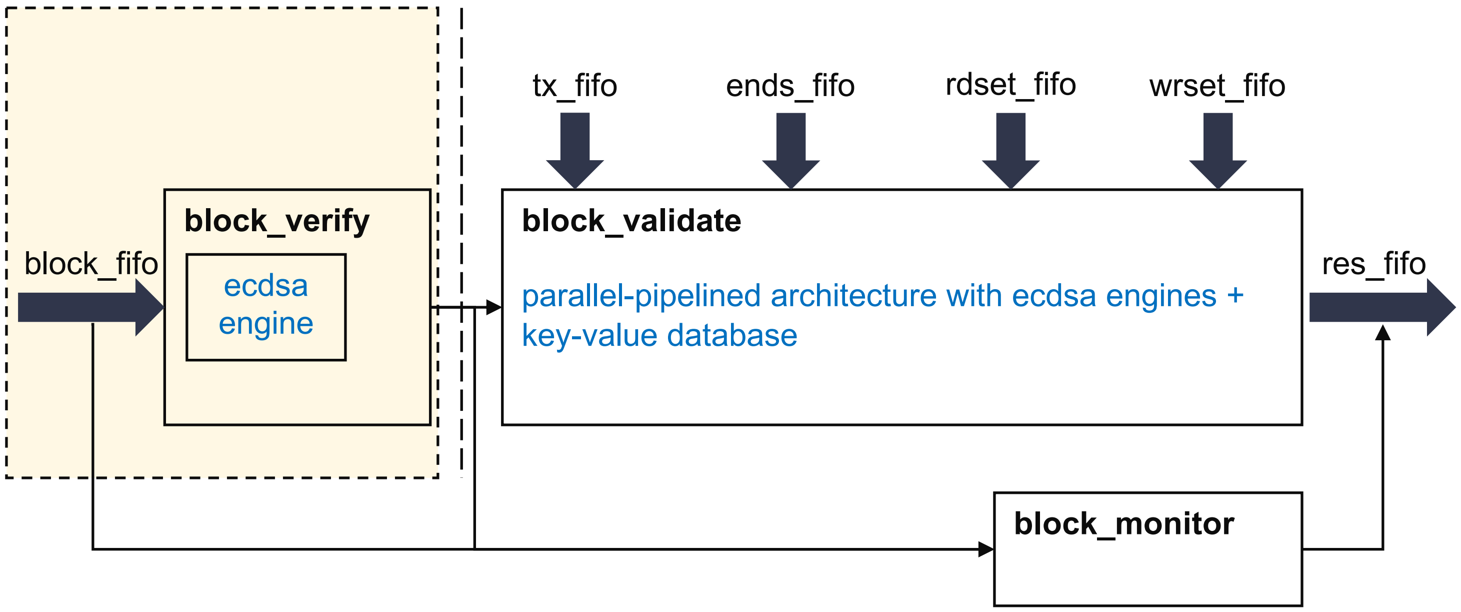
- > Each block is broken down into three sections: header, transactions, metadata
  - >> Each section is sent separately in its own packet
  - >> Repetitive data (such as identity certificates) are replaced with encoded ids
- > Self-contained UDP packets
  - >> Each packet contains annotations in its header for efficient retrieval of data from its payload, without waiting for other packets

Protocol Processor



- The protocol_processor acts as the hardware-friendly receiver
  - Parses each packet and uses annotations embedded in its header to extract the relevant data
  - Transforms extracted data for block_processor (e.g. calculates hash of the block, transaction, etc.)

Block Processor Architecture – Stage 1



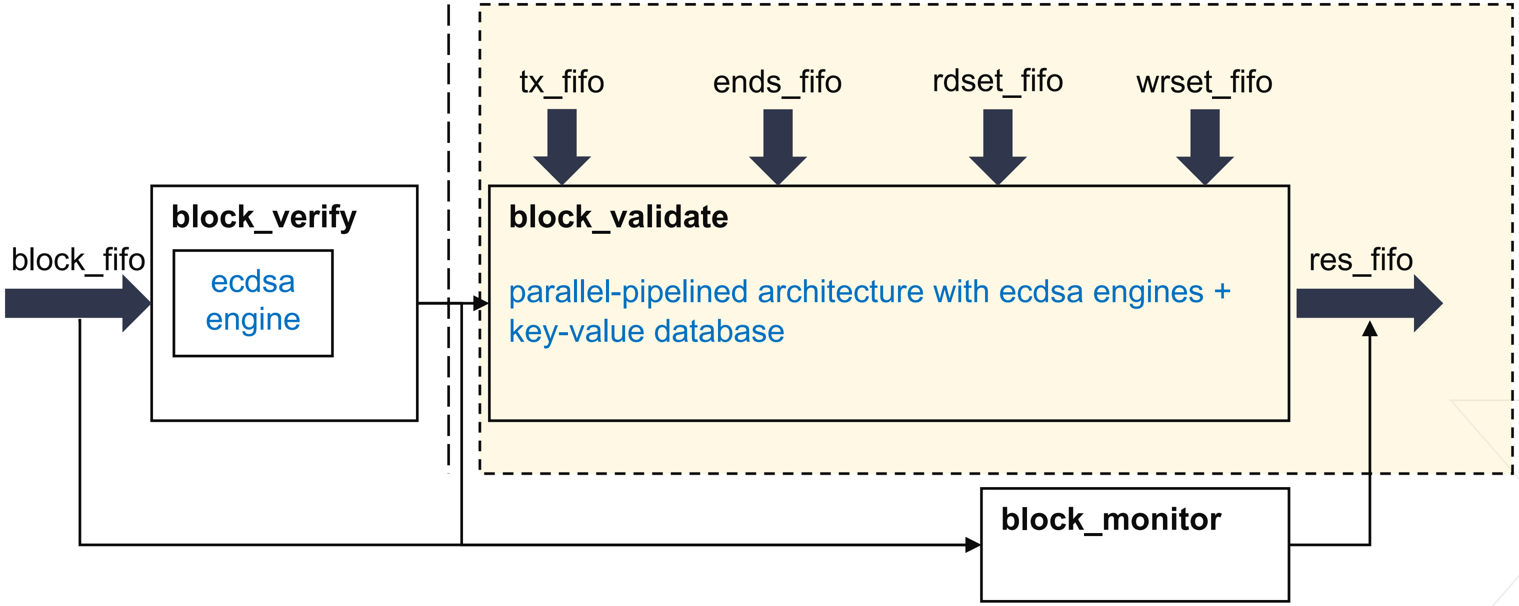
> Two stage block-level pipeline (two blocks processed in a pipelined fashion)

> block_verify verifies the orderer signature on the block

  >> block_fifo: each element contains block-level data like block number, number of transactions, orderer signature, etc.

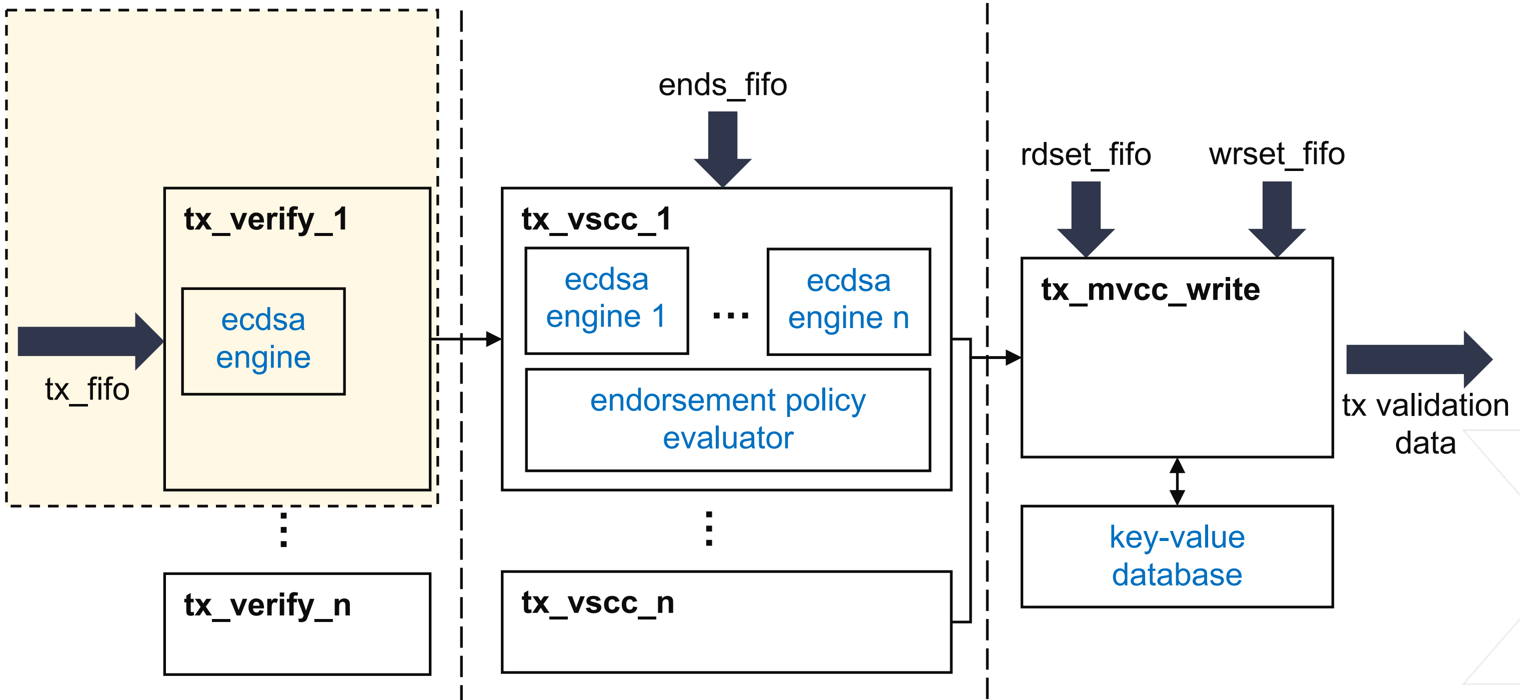
  >> One ecdsa engine is dedicated to this stage.

Block Processor Architecture – Stage 2



- block_validate validates all the transactions in a block
  - Multiple transactions can be processed in parallel, and tx validation results are collected.
  - res_fifo: each element contains validation data like block number, valid/invalid tx flags, latency, etc.

Block Validate Architecture – Stage 1



> Parallel-pipelined architecture with three transaction-level stages

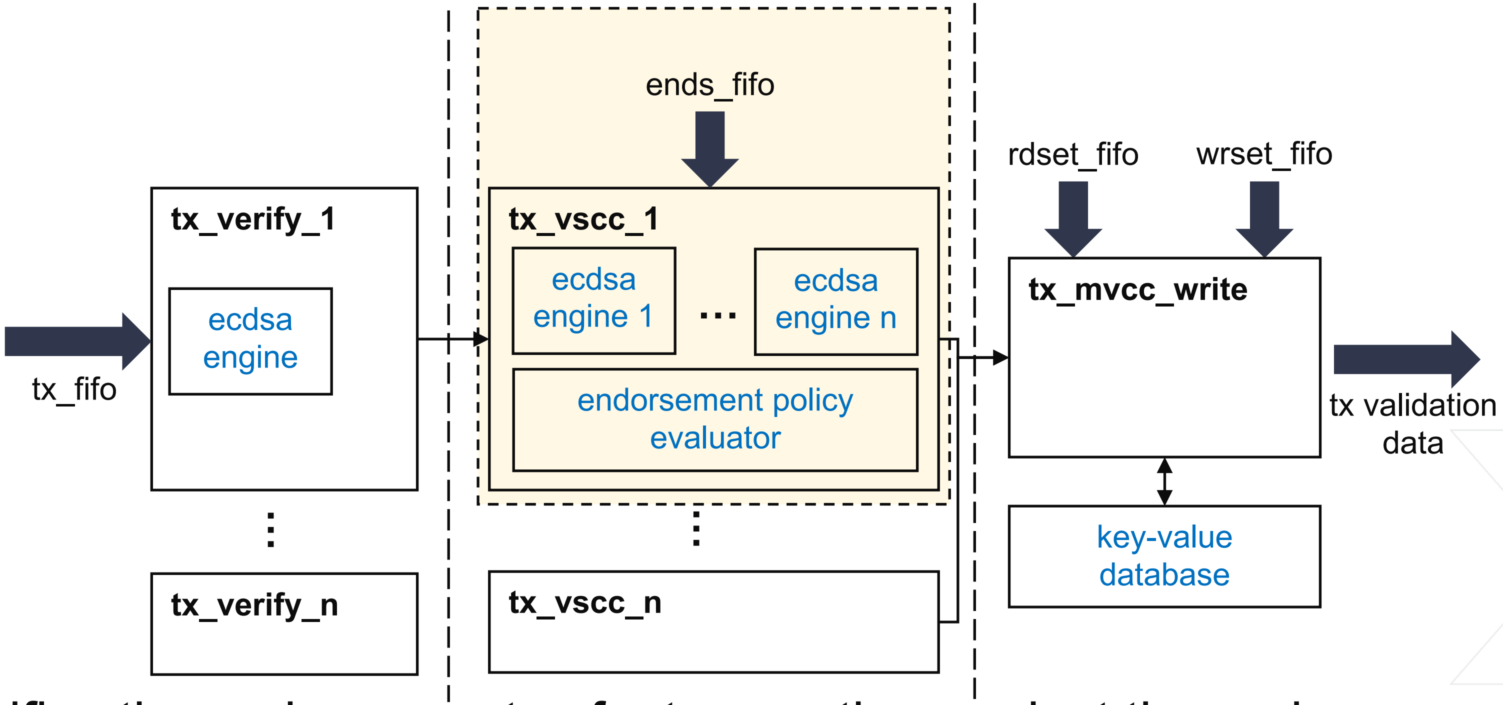
> tx_verify verifies the client/creator signature on the transaction

>> tx_fifo: each element contains transaction-level data like tx number, client/creator signature, number of endorsements, number of read and write keys, etc.

>> One ecdsa engine is used in each tx_verify instance.

> Multiple tx_verify instances can be included in the design, which operate in parallel.

Block Validate Architecture – Stage 2

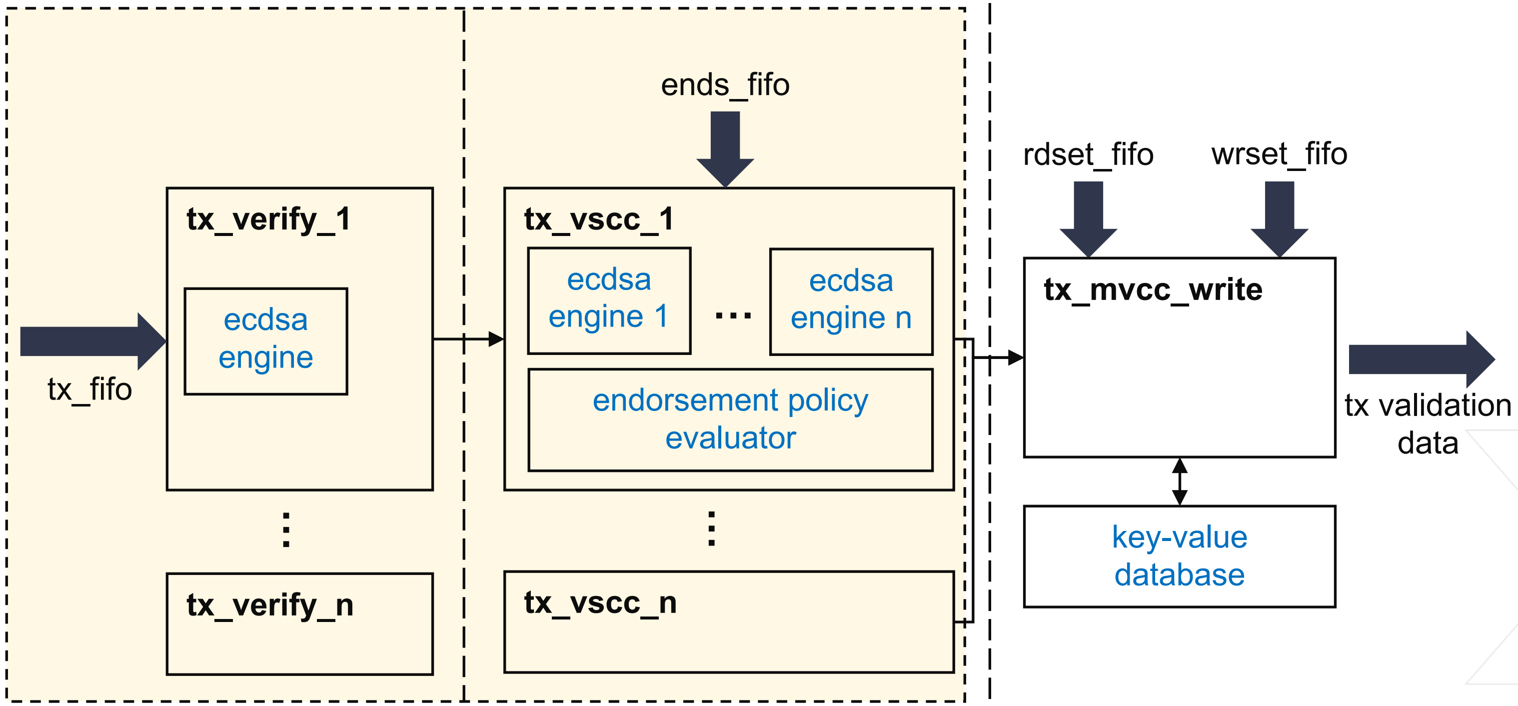


> tx_vscc verifies the endorsements of a transaction against the endorsement policy
  - ends_fifo: each element contains endorser id and endorsement data (e.g. signature, etc.)
  - Endorsements are scheduled across multiple ecdsa engines
  - Policy evaluator keeps endorsement policies on a per-chaincode basis
  - Configurable number of ecdsa engines (based on application requirements) in each tx_vscc instance

> Multiple tx_vscc instances can be included in the design, which operate in parallel.

XILINX

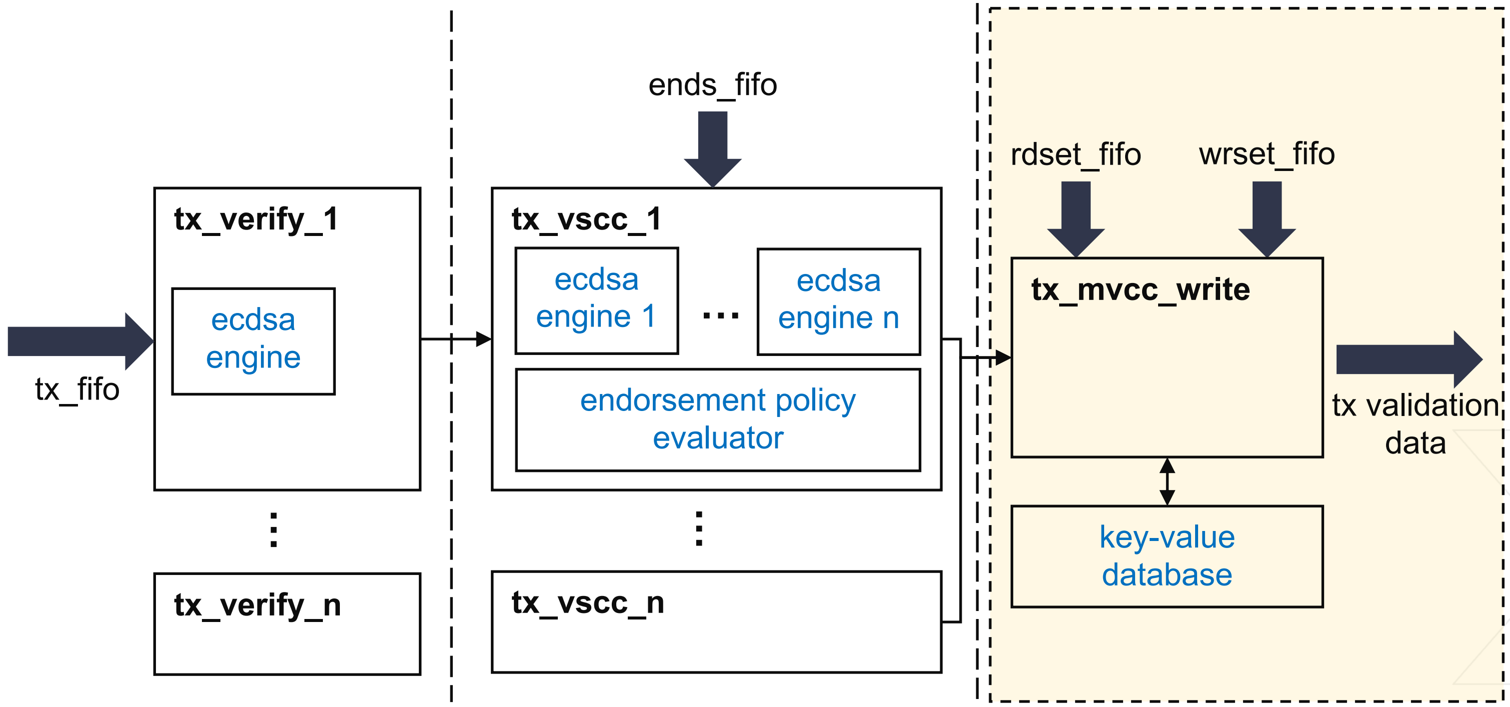
Block Validate Architecture – Stage 1 & 2



- Parallel-pipelined architecture
  - Transactions are scheduled across the tx_verify/tx_vscc instances (transactions of a block can be validated in parallel)
  - Tx validation results are collected in-order from tx_vscc for the next stage
  - Configurable number of ecdsa engines (based on application requirements) in each tx_vscc instance
  - Configurable number of tx_verify/tx_vscc instances (based on application requirements)

XILINX

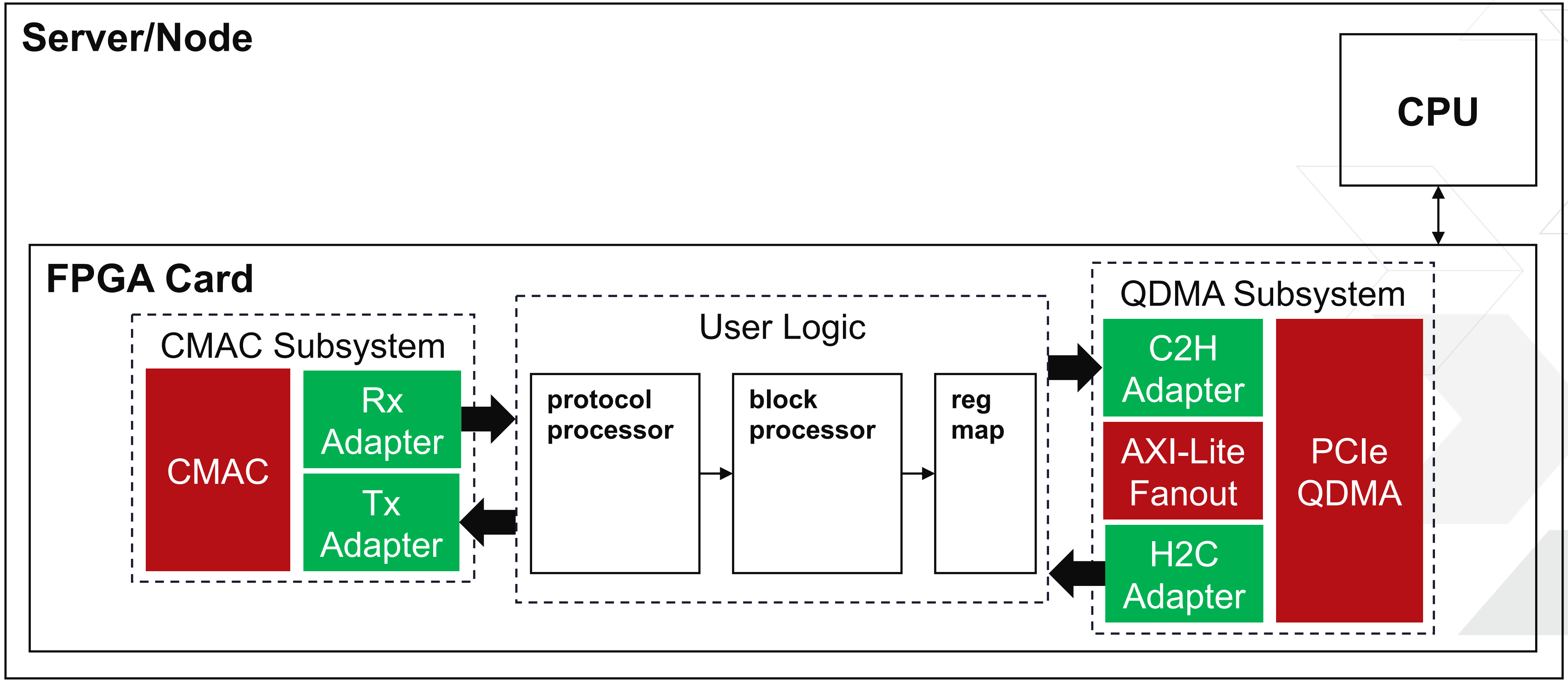
Block Validate Architecture – Stage 3



- tx_mvcc_write looks up read keys from database for version check, and commits write keys of valid transactions to database
  - rdset_fifo: each element contains a read key-version pair
  - wrset_fifo: each element contains a write key-value pair
  - key-value database: configurable key/value size, 1 cc read latency, 10-20 ccs write latency, internal lock mechanism to disallow reading of the key being written/updated

Putting It All Together

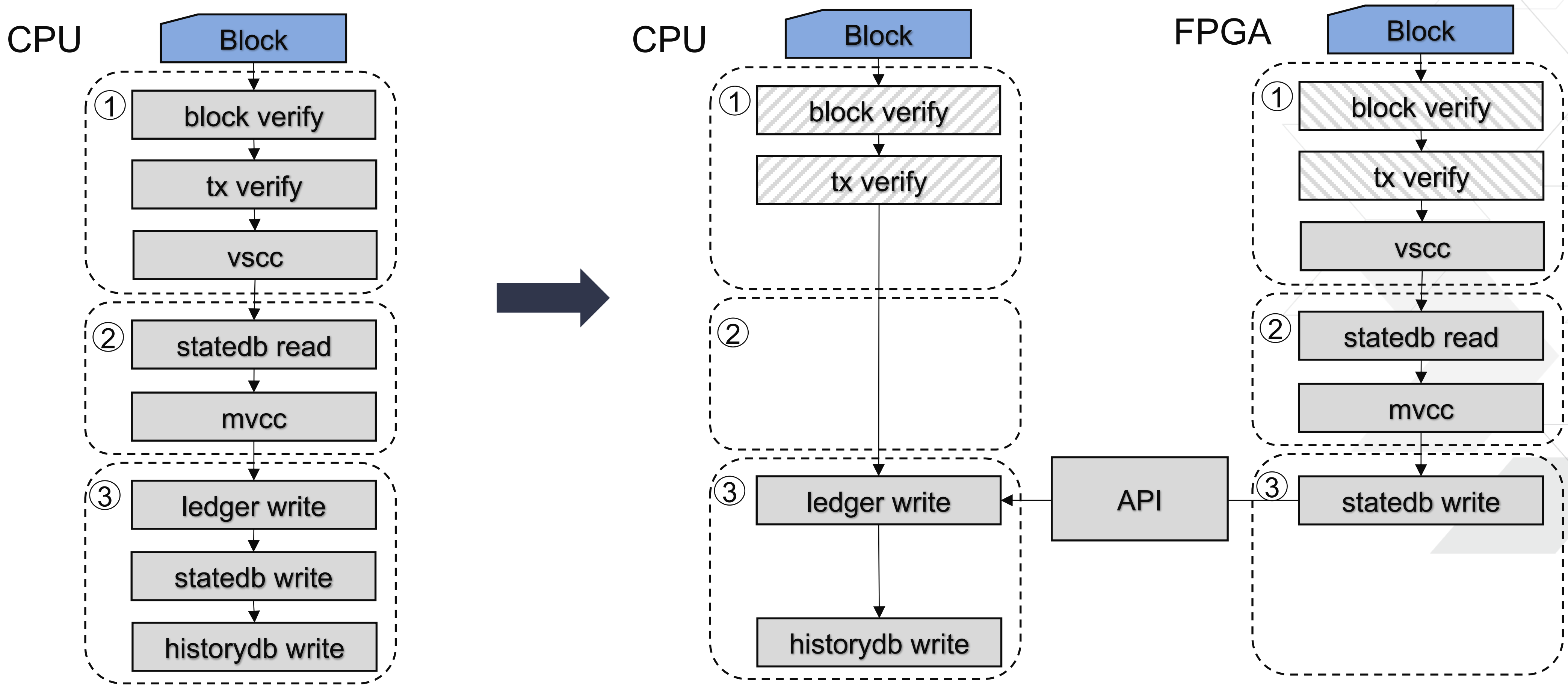
> Blockchain Machine implemented as the user logic block inside Xilinx OpenNIC [1]



[1] Xilinx. OpenNIC Project. https://github.com/Xilinx/open-nic

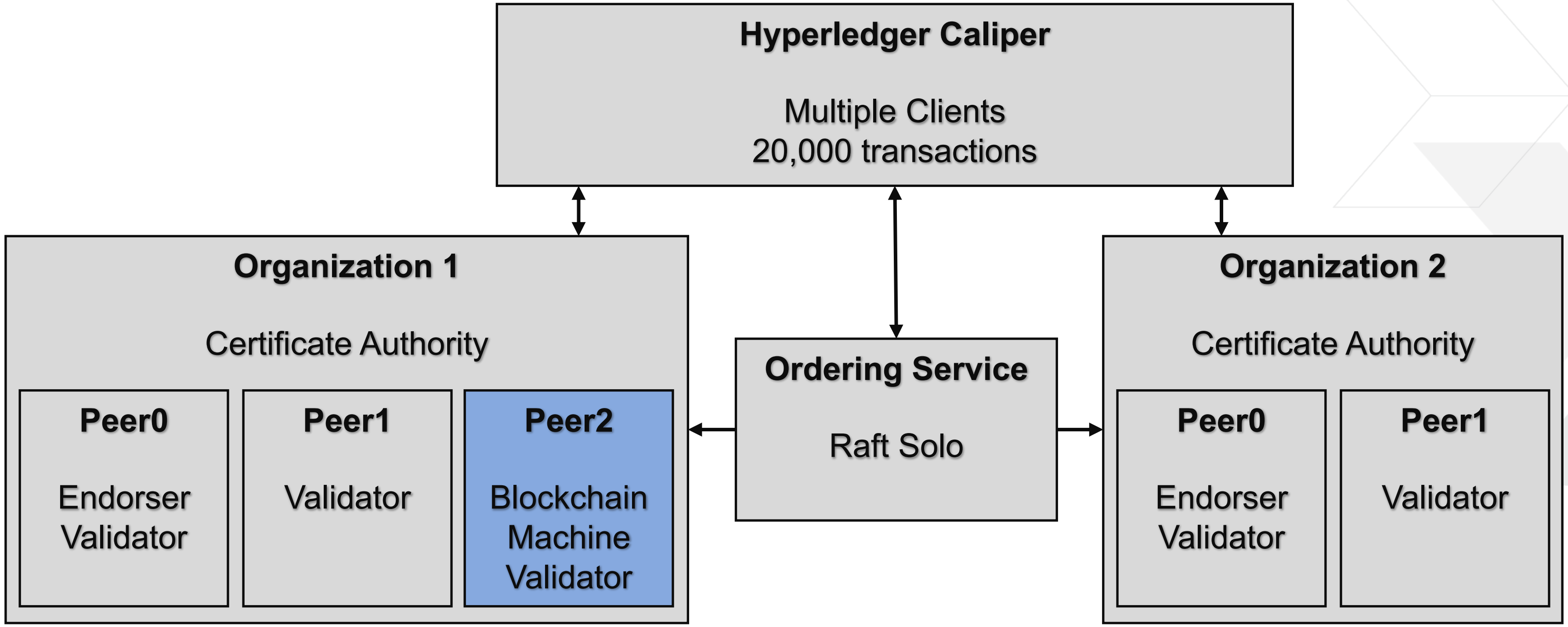
Putting It All Together

> [Hardware/Software partitioning] in Blockchain Machine
>> Data from hardware is read through a [Go language API]



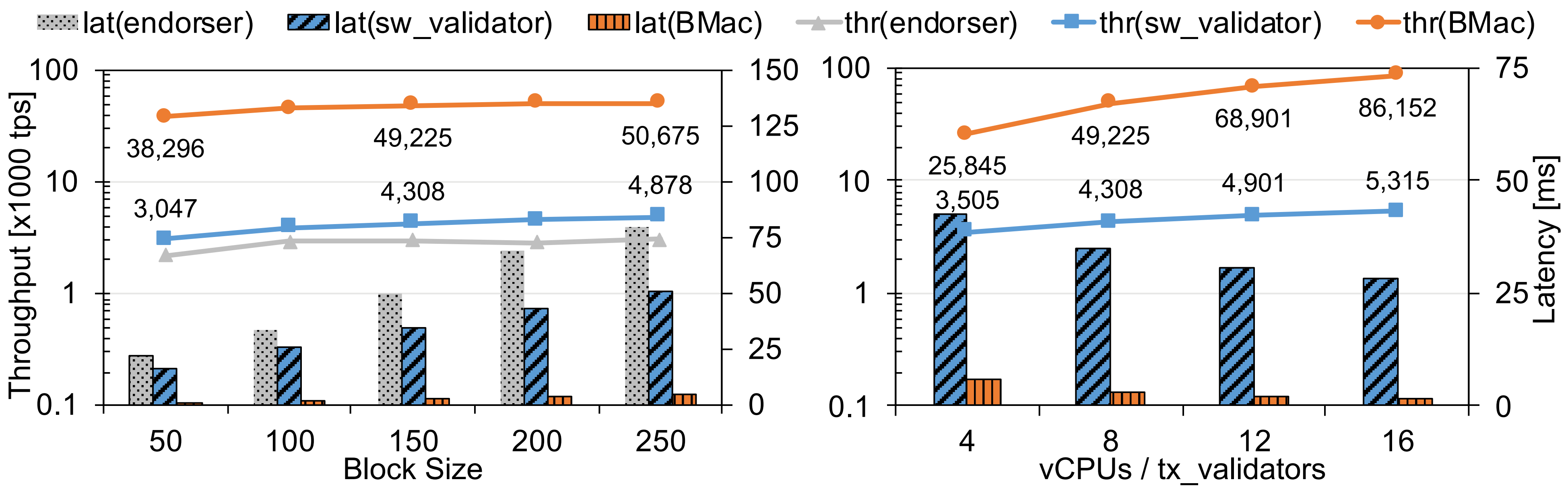
Evaluation Setup

- Hyperledger Fabric network
  - Caliper runs the smallbank benchmark (account creation, money transfers, etc.)
  - Orderer and peers are run in their own VMs (multiple vCPUs) -- number of vscc threads is the same as number of vCPUs
  - Blockchain Machine (BMac) peer is programmed on Alveo U250 board with multiple tx_validators (like vscc threads)



Results (1)

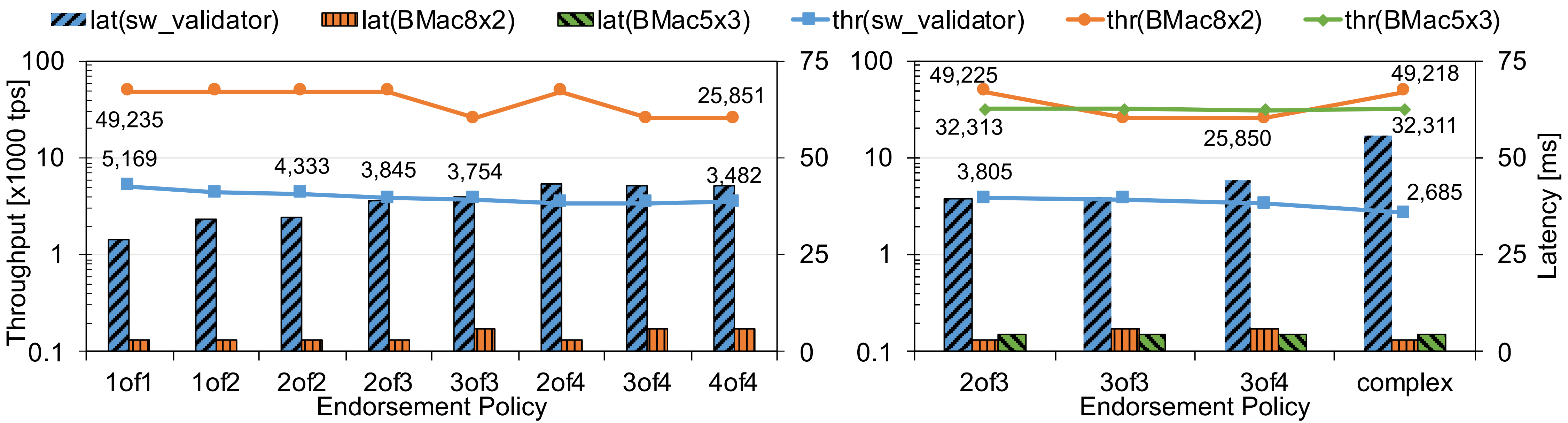
- > Block latency = time spent in validation phase
- > Commit throughput = transactions committed per second
- > The ledger write operation is excluded because it is executed on CPU in all cases
- > At least 10x speedup



Results (2)

> Blockchain Machine can be
  > Programmed with multiple endorsement policies/chaincodes
  > Adapted to endorsement policy (cryptographic workload)

> 4x2 means 4 parallel tx_validators each with 2 ecdsa engines per tx_vscc



XILINX

Results (3)

- Resource utilization on Alveo U250 board
- 4x2 means 4 parallel tx_validators each with 2 ecdsa engines per tx_vscc

Resource	4x2	5x3	8x2	12x2	16x2
LUT / LUTRAM	20.9%	25.4%	28.5%	35.8%	43.3%
FF	6.9%	6.9%	8.0%	9.1%	10.3%
BRAM / URAM	13.1%	13.1%	13.1%	13.1%	13.1%

Concluding Remarks

- Permissioned blockchains are well-suited for
  - Network-attached acceleration
  - Hardware/software co-design
  - FPGA programmability

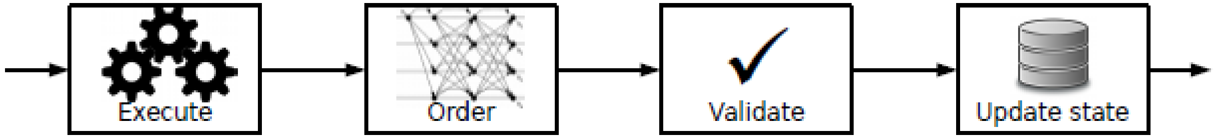
- Blockchain Machine proof-of-concept shows promising results

- Focusing on open-source contributions
  - Hyperledger Labs project (already launched, stay tuned for more updates soon!)
  - Setup in XACC NUS (just started)

Adaptable.
Intelligent.

XILINX®

Hyperledger Fabric Overview

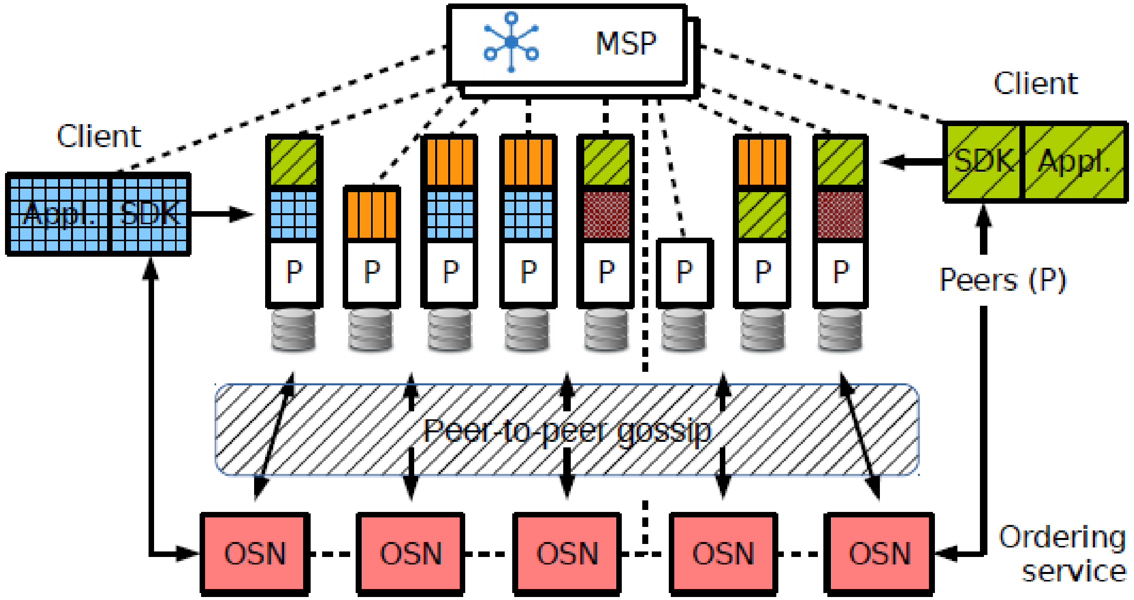


- Simulate trans. and endorse
- Create rw-set
- Collect endorse-ments

- Order rw-sets
- Atomic broadcast (consensus)
- Stateless ordering service

- Validate endorse-ments & rw-sets
- Eliminate invalid and conflicting trans.

- Persist state on all peers



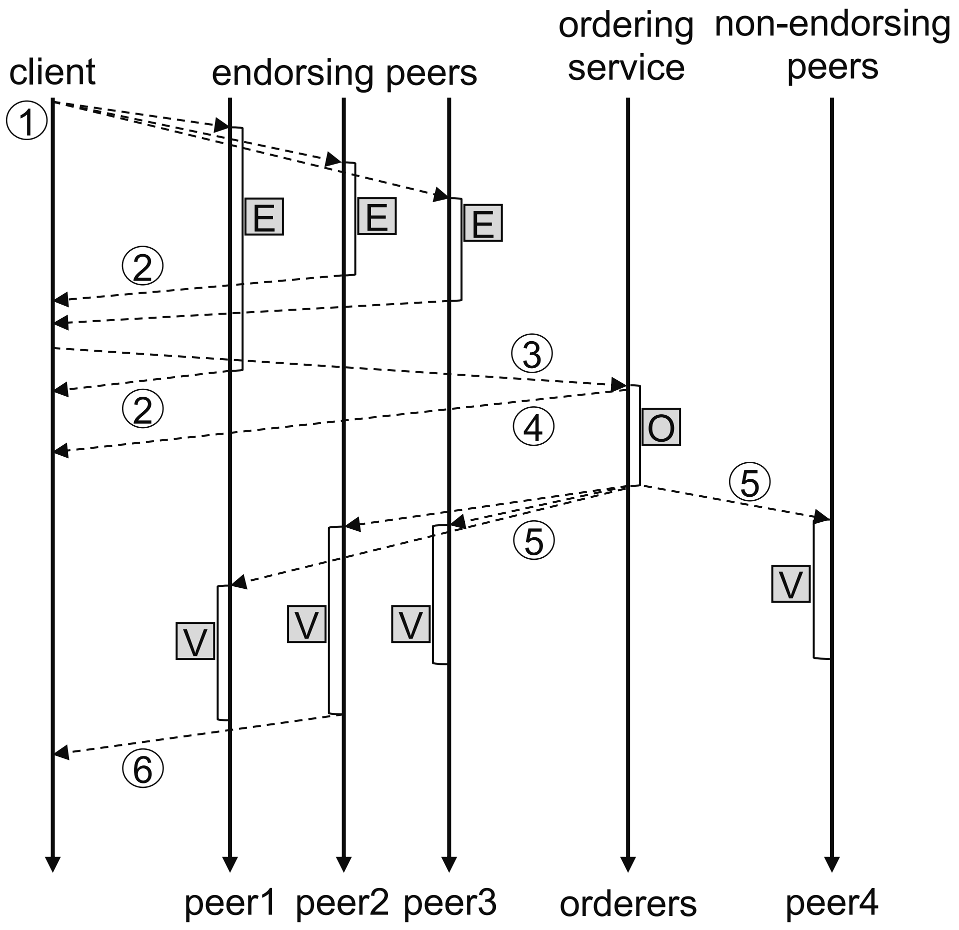
Peer

Endorser	Chaincode execution, containers
Committer	Validation, configuration, sys. chaincode
Ledger	Block store, peer trans. manager (PTM)
KVS	LevelDB, CouchDB
Gossip	Peer-to-peer block gossip

[1] E. Androulaki et al., “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains” in EuroSys, 2018.

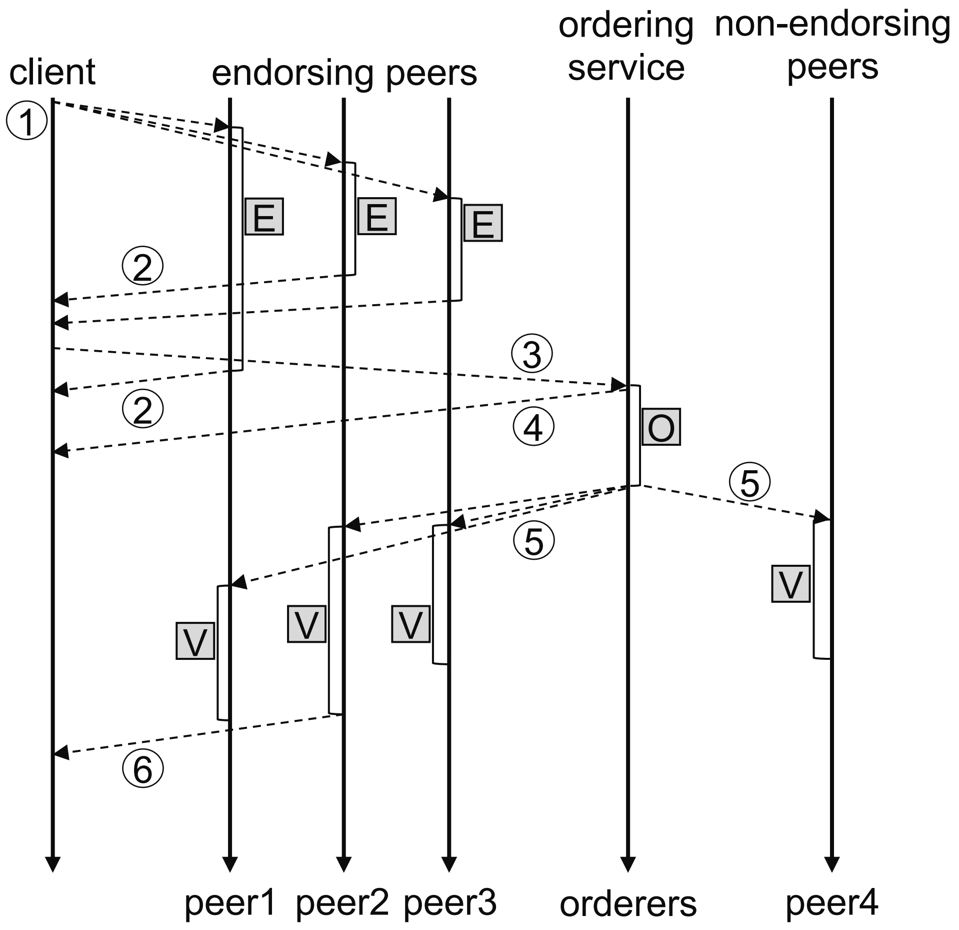
XILINX

Hyperledger Fabric Transactions



1. Client invokes a transaction (by sending it to endorsers)
2. Peer sends endorsement (client collects all endorsements)
3. Client submits the transaction to ordering service (for being ordered and included in a block)
4. Orderer sends confirmation to client
5. Orderer broadcasts the block to peers (after a timeout or block has reached its limit)
6. Peer sends commit notification to client

Hyperledger Fabric (Endorser + Orderer)



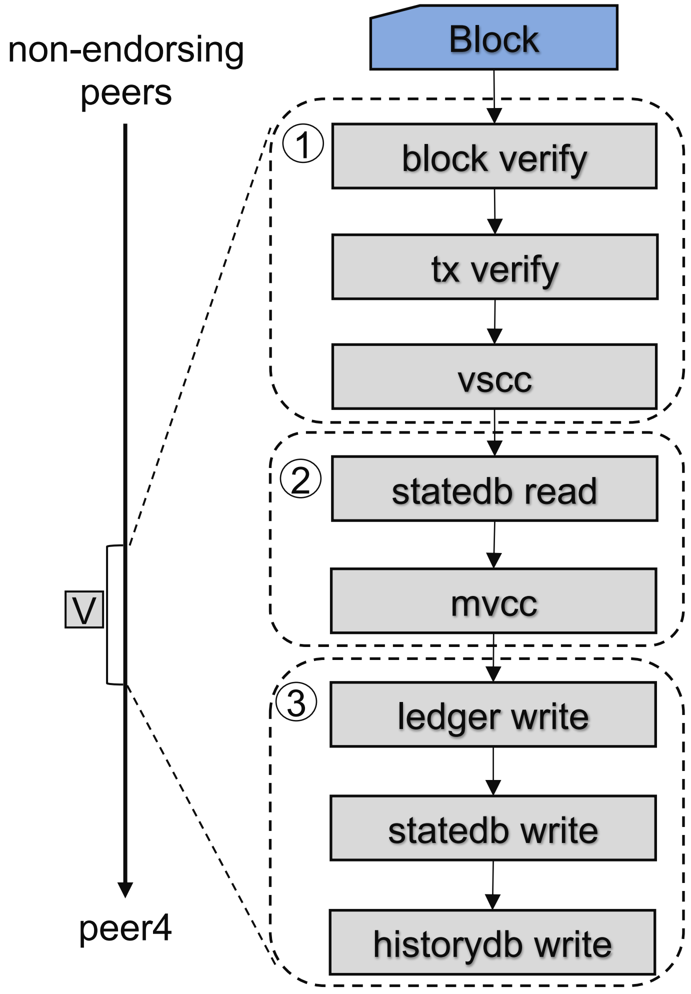
Endorsing peer (E):

- Verifies client's identity and checks whether it is authorized
- Simulates the execution of the chaincode (smart contract) to create the transaction's read and write sets against its local state database
- Runs endorsement system chaincode (escc) to sign the results of the transaction simulation
- Returns the result to the client

Ordering service (O):

- Enqueues the incoming transactions
- Creates a block of transactions after a timeout or enough transactions were available
- Signs the block
- Broadcasts the block to the peers through Gossip protocol

Hyperledger Fabric (Validator)



Validator peer (V):

- Verifies the orderer signature
- Verifies each transaction’s syntax and creator signature
- Runs validation system chaincode (vscc) on each transaction
  - Validates each endorsement of the transaction
  - Ensures endorsements satisfy the endorsement policy

- Reads from state database to create read sets of all the transactions
- Runs multi-version concurrency control (mvcc) to check read-write conflicts across the transactions
  - Compares current read sets with those from the endorsement phase

- Updates state database with the write sets of valid transactions
- Commits block to ledger

Validation phase is one of the major bottlenecks [1, 2, 3]

[1] P. Thakkar, S. Nathan, and B. Vishwanathan, “Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform,” in *MASCOTS*, 2018.
[2] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, “FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second,” in *ICBC*, 2019.
[3] P. Thakkar and S. Nathan. 2021. Scaling Hyperledger Fabric Using Pipelined Execution and Sparse Peers. arXiv:2003.05113.