Untangling Blockchain: A Data Processing View of Blockchain Systems

Instructor: Wei Zheng

Reporter: Xinbo Zhang

Author : Tien Tuan Anh Dinh , Rui Liu, Meihui Zhang , Member, IEEE,
Gang Chen, Member, IEEE, Beng Chin Ooi , Fellow, IEEE, and Ji Wang

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

Outline:

- Blockchains
  - Private & Public
- Key concepts
  - Distributed Ledger
  - Consensus
  - Cryptography
  - Smart Contrast
- BLOCKBENCH
  - Layers
  - Implementation
  - Workloads
- Evaluation
  - Macro benchmarks
  - Micro benchmarks
- Recap

Comparison of Blockchain Systems

	Application	Smart contract execution	Smart contract language	Data model	Consensus
Hyperledger v0.6.0 [32]	General applications	Dockers	Golang, Java	Key-value	PBFT
Hyperledger v1.0.0 [33]	General applications	Dockers	Golang, Java	Key-value	Ordering service (Kafka)
Bitcoin	Crypto-currency	Native	Golang, C++	Transaction-based	PoW
Litecoin [34]	Crypto-currency	Native	Golang, C++	Transaction-based	PoW (memory)
ZCash [35]	Crypto-currency	Native	C++	Transaction-based	PoW (memory)
Ethereum [5]	General applications	EVM	Solidity, Serpent, LLL	Account-based	PoW
Multichain [36]	Digital assets	Native	C++	Transaction-based	Trusted validators (round robin)
Quorum [37]	General applications	EVM	Golang	Account-based	Raft
HydraChain [38]	*General applications*	*Python, EVM*	*Solidity, Serpent, LLL*	*Account-based*	*Trusted validators (majority)*
OpenChain [39]	*Digital assets*	-	-	*Transaction-based*	*Single validator*
IOTA [40]	*Digital assets*	-	-	*Account-based*	*IOTA's Tangle Consensus*
BigchainDB [31]	Digital assets	Native	Python, crypto-conditions	Transaction based	Trusted validators (majority)
Monax [24]	General applications	EVM	Solidity	Account-based	Tendermint [41]
Ripple [6]	Digital assets	-	-	Account-based	Ripple consensus
Kadena [30]	*Pact applications*	*Native*	*Pact*	*Table*	*ScalableBFT [42]*
Stellar [29]	Digital assets	-	-	Account-based	Stellar consensus
Dfinity [43]	*General applications*	*EVM*	*Solidity, Serpent, LLL*	*Account-based*	*Threshold relay*
Parity [11]	General applications	EVM	Solidity, Serpent, LLL	Account-based	Trusted validators (round robin)
Tezos [44]	Michaleson applications	Native	Michaleson	Account-based	Proof of Stake
Corda [45]	Digital assets	JVM	Kotlin, Java	Transaction-based	Raft
Sawtooth Lake [46]	General applications	Native	Python	Key-value	Proof of elapsed time

Ones in italics are deemed inactive or at early phases of development.

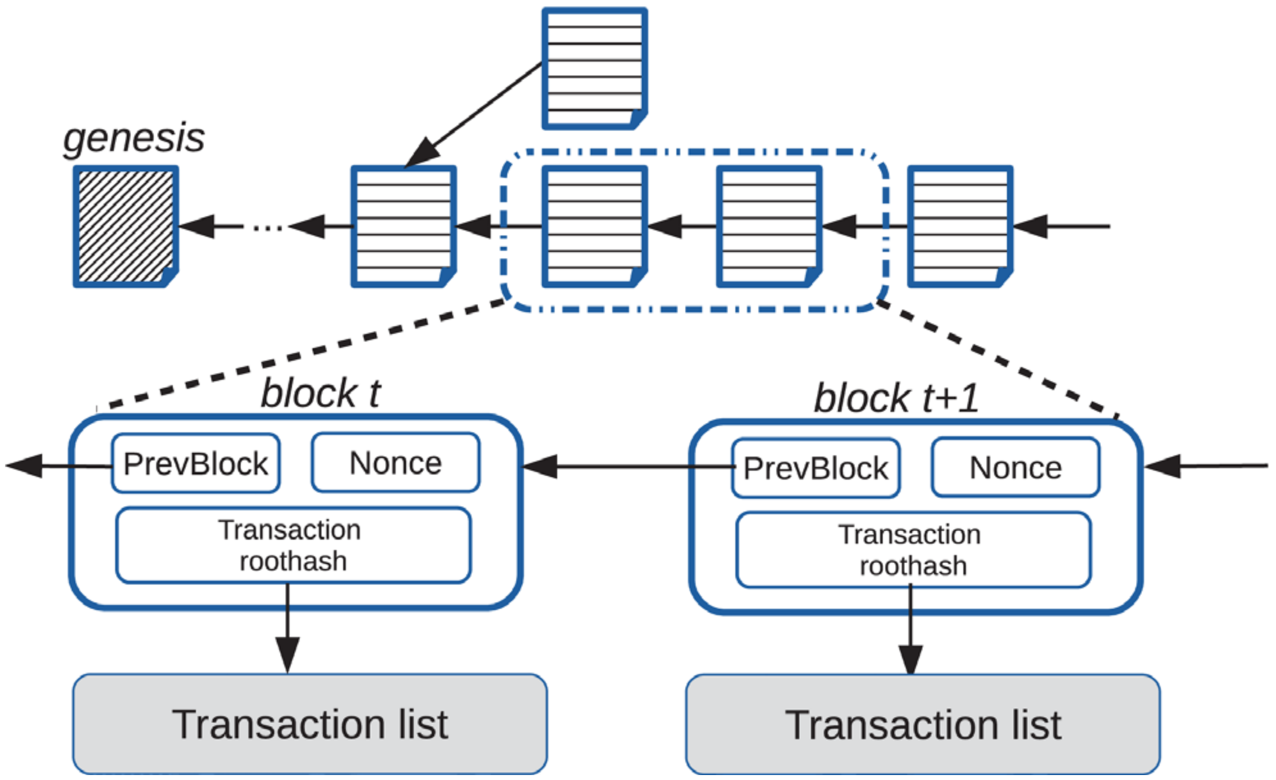
Outline:

- Blockchains
  - Private & Public
- Key concepts
  - Distributed Ledger
  - Consensus
  - Cryptography
  - Smart Contrast
- BLOCKBENCH
  - Layers
  - Implementation
  - Workloads
- Evaluation
  - Macro benchmarks
  - Micro benchmarks
- Recap

	Application	Smart contract execution	Smart contract language	Data model	Consensus
Hyperledger v0.6.0 [32]	General Applications	Dockers	Golang, Java	Key-value	PBFT
Hyperledger v1.0.0 [33]					Ordering service (Kafka)
Ethereum [5]		EVM	Solidity, Serpent, LLL	Account-based	PoW
Parity [11]					PoA

Blockchains:

- A typical blockchain system consists of multiple nodes which do not fully trust each other.
- The nodes maintain a set of shared, global states and perform transactions modifying the states.
- Blockchain is a special data structure which stores historical states and transactions.



Blockchains: Private & Public

- Private:
  - Blockchain platforms designed for private settings where participants are authenticated are called private (or permissioned)
- Public:
  - As opposed to the early systems operating in public environments (or permissionless) where anyone can join and leave.

Key Concepts:

- Distributed Ledger
- Consensus
- Cryptography
- Smart Contracts

Key Concepts: Distributed Ledger

- A ledger is a data structure that consists of an ordered list of transactions.
  - e.g, a ledger may record monetary transactions between multiple banks, or goods exchanged among known parties.
- In blockchains, the ledger is replicated over all the nodes.
  - transactions are grouped into blocks which are then chained together.
- The distributed ledger is essentially a replicated append-only data structure.
- A blockchain starts with some initial states, and the ledger records entire history of update operations made to the states.

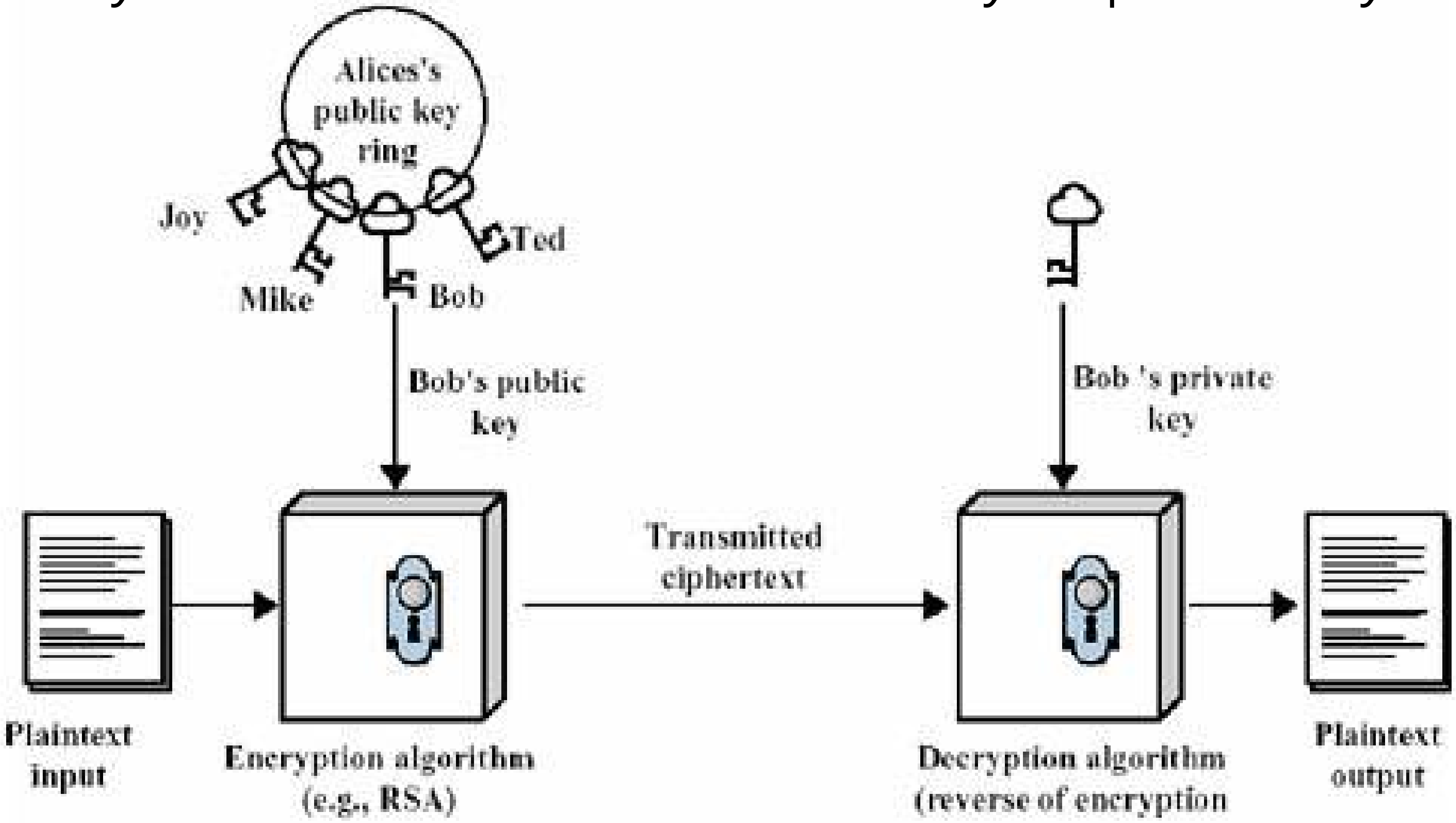
Key Concepts: Consensus

- Being replicated, updates to the ledger must be agreed on by all parties.
- The consensus protocol must tolerate Byzantine failures.

- PBFT(Practical Byzantine Fault Tolarant):
  - purely communication based protocols in which nodes have equal votes and go through multiple rounds of communication to reach consensus.
  - PoA(Proof of Authority), used in private blockchains where they improve PBFT by executing consensus in smaller networks called federates.
- PoW(Proof of Work):
  - use proof of computation to randomly select a node which single-handedly decides the next operation.

Key Concepts: Cryptography

- To ensure integrity of the ledgers
- Blockchain's security model assumes the availability of public key cryptography.



Key Concepts: Smart Contrast

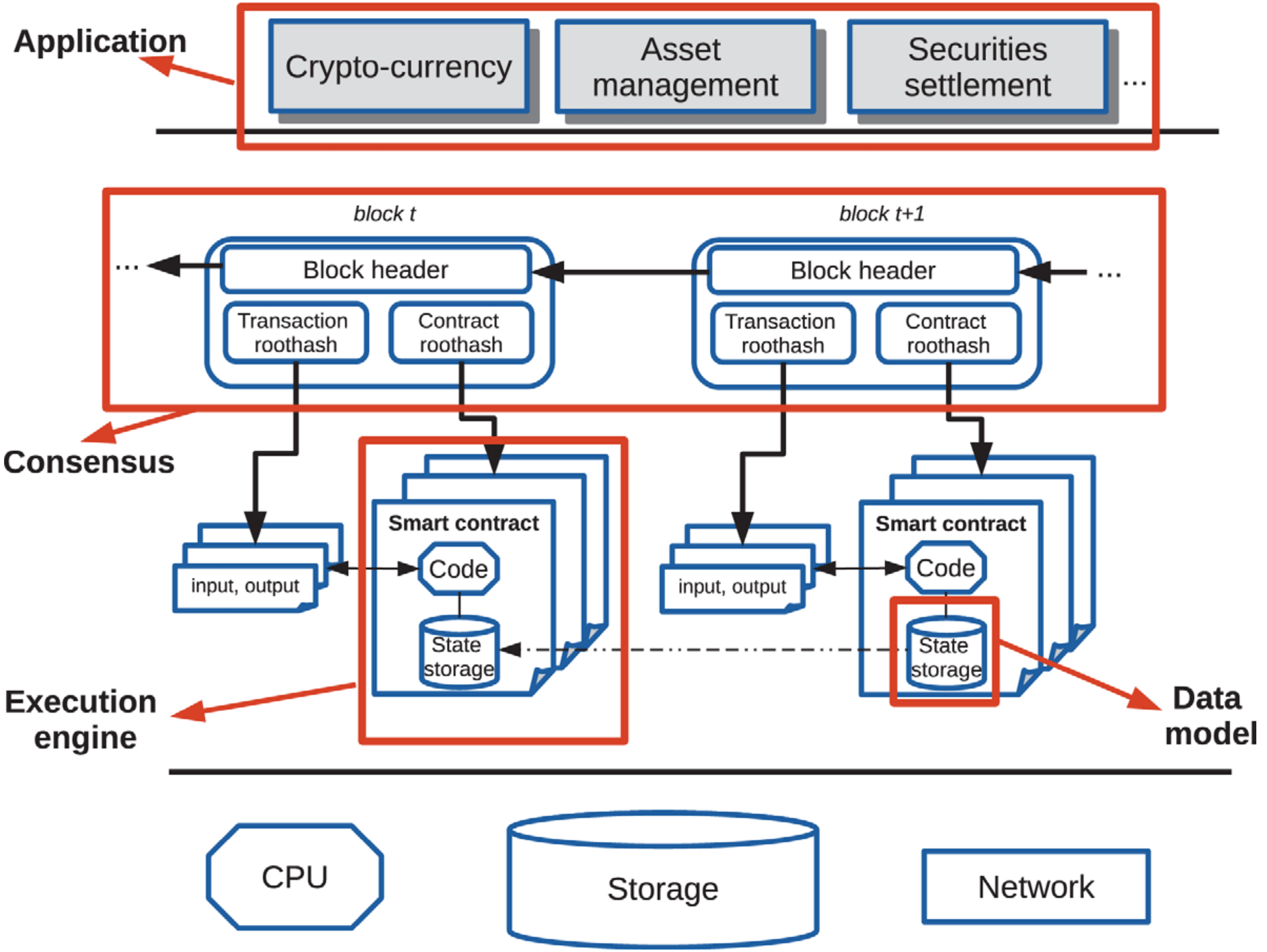
- The computation executed when a transaction is performed.

```
contract Doubler{
    struct Participant{
        address etherAddress;
        unit amount;
    }
    Participant[] public participants;
    unit public balance = 0;
    /** ... */
    function enter(){
        /** ... */
        balance += msg.value;
        /** ... */
        if(balance > 2*participants[payoutIdx.amount]){
            transantionAmount = /** ... */;
            participants[payoutIdx].etherAddress.send(transactionAmount);
        }
    }
}
```

BLOCKBENCH:

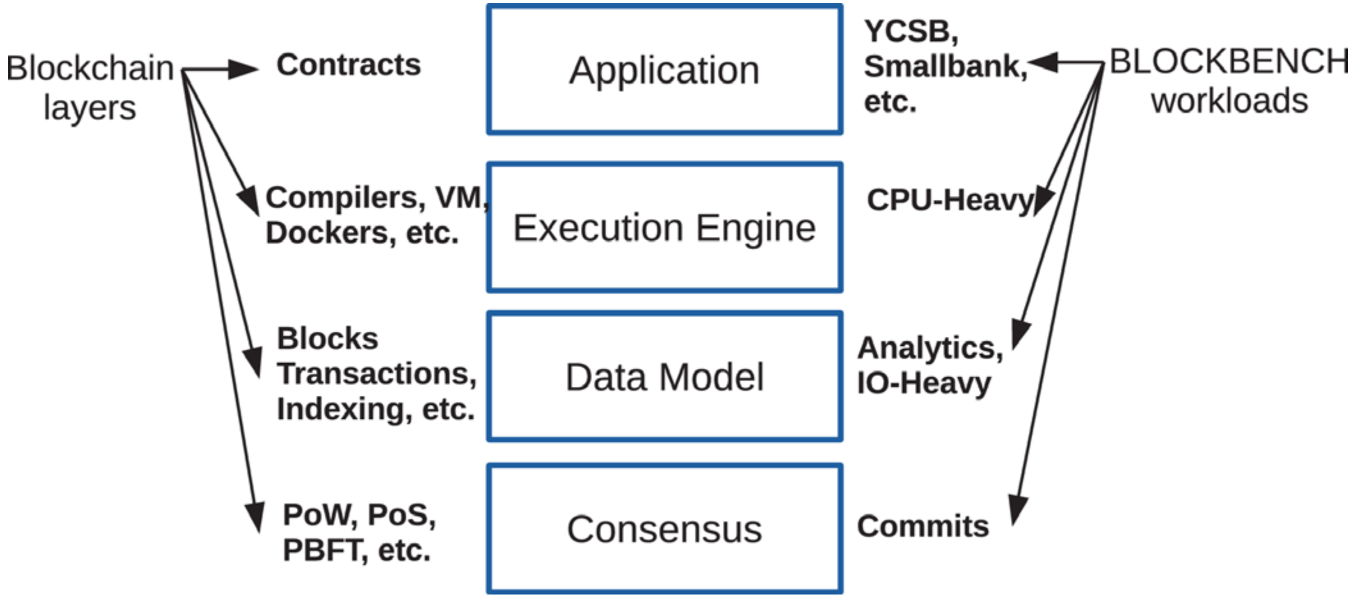
The framework targets private blockchains with Turing-complete smart contracts.

BLOCKBENCH is open source and contains data processing workloads commonly found in database benchmarks



BLOCKBENCH: Layers

- The consensus layer implements the consensus protocol.
- The data model layer contains the structure, content and operations on the blockchain data.
- The execution layer includes details of the runtime environment for executing smart contracts.
- The application layer includes class of blockchain applications

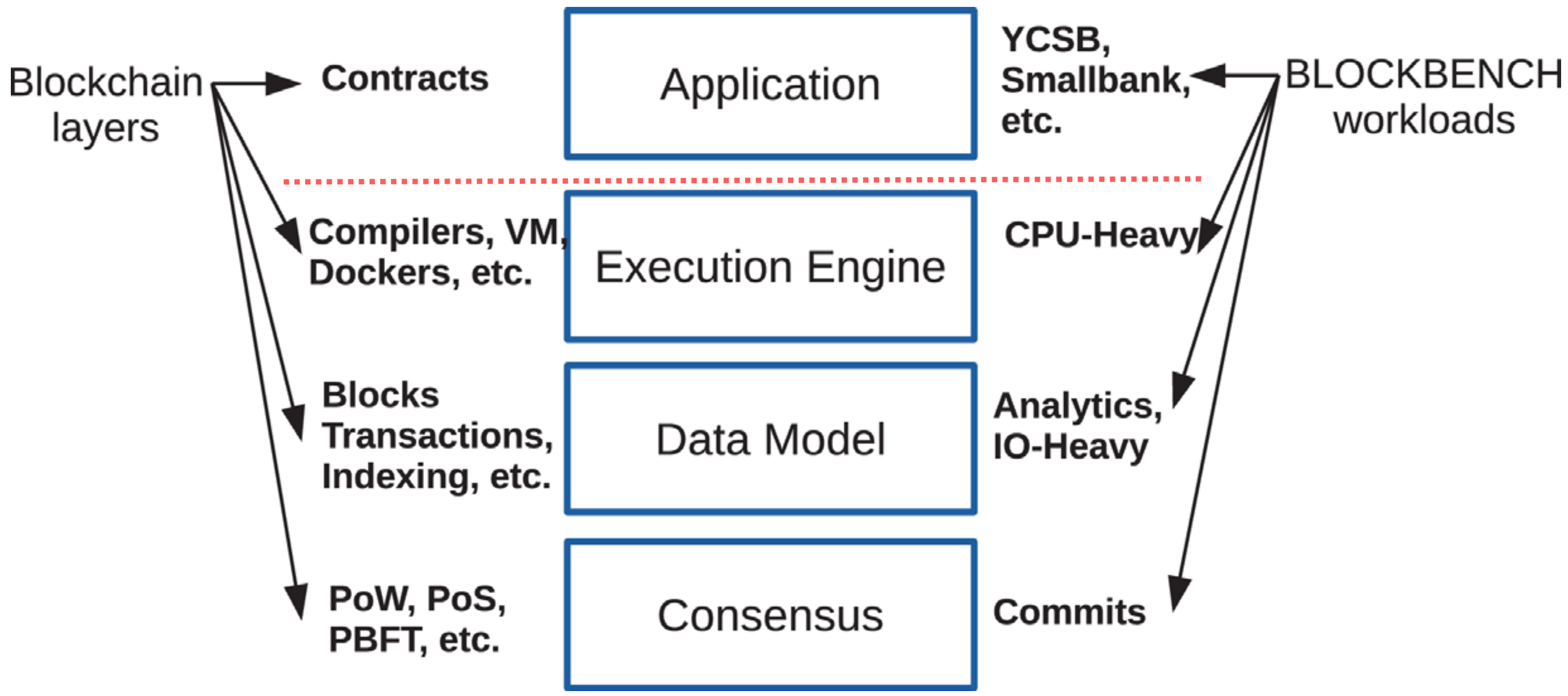


BLOCKBENCH: Implementaion

- BLOCKBENCH stack consists of
  - a frontend interface for integrating new benchmark workloads
  - a backend interface for integrating new blockchains
  - a driver for driving the workloads.
- It collects runtime statistics which are used to compute five important metrics.
  - Throughput: the number of successful transactions per second.
  - Latency: the response time per transaction.
  - Scalability: the changes in throughput and latency when increasing the number of nodes and number of concurrent workloads.
  - Fault tolerance: the changes in throughput and latency during node failure.
  - Security metrics: the ratio between the total number of blocks included in the main branch and the total number of confirmed blocks.

BLOCKBENCH: Workloads

- Macro benchmark workloads for evaluating the application layer,
- Micro benchmark workloads for analyzing the lower layers.
- Deployed on Ethereum, Parity and Hyperledger

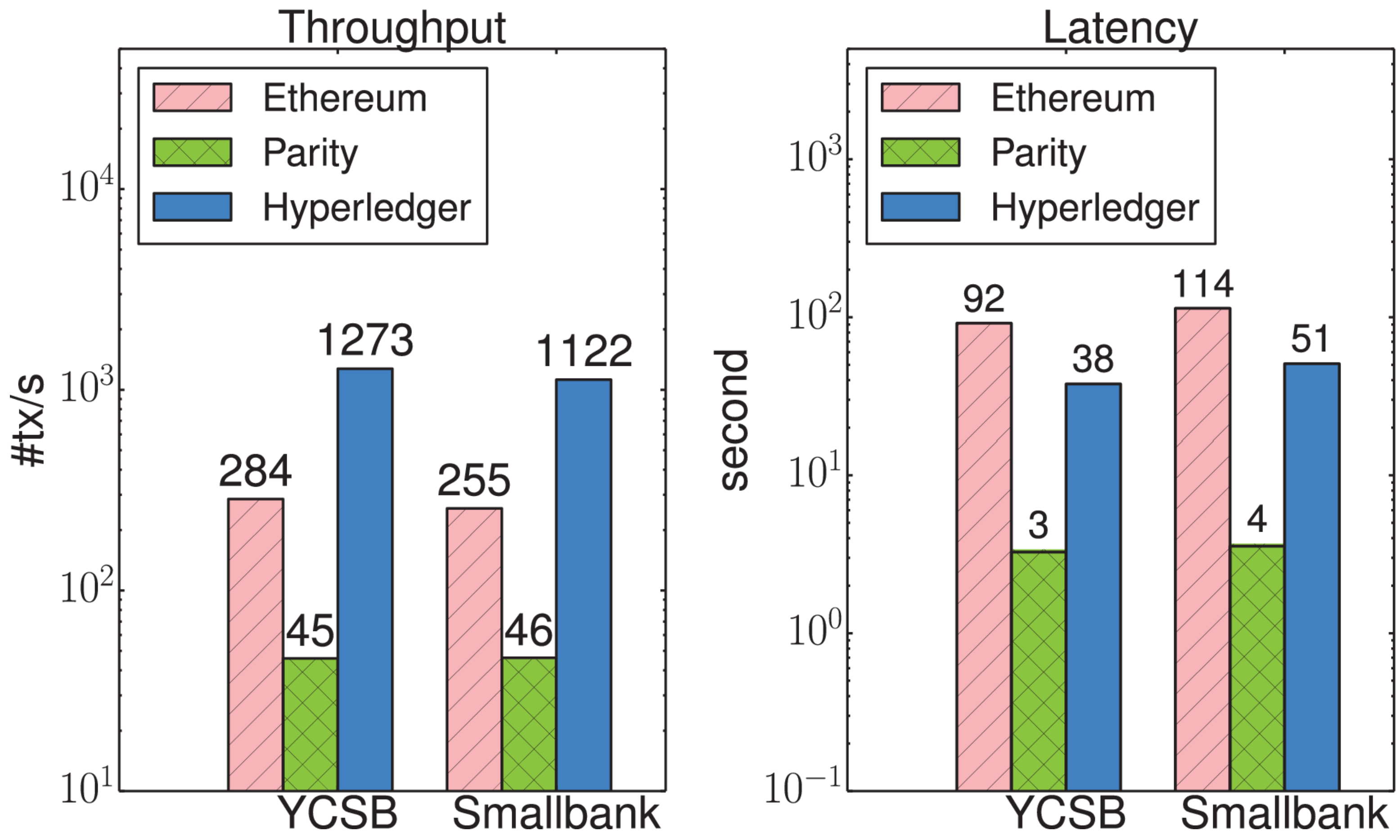


Evaluation:

- Selected Ethereum, Parity and Hyperledger for a comparative study using BLOCKBENCH
  - Go implementation
  - the Ethereum, geth v1.4.18
  - the Parity release v1.6.0
  - the Hyperledger version is v0.6.0-preview.
- The experiments were run on a 48-node commodity cluster
- Each node has an E5-1650 3.5 GHz CPU, 32 GB RAM, 2 TB hard drive, running Ubuntu 14.04 Trusty, and connected to the other nodes via 1 GB switch

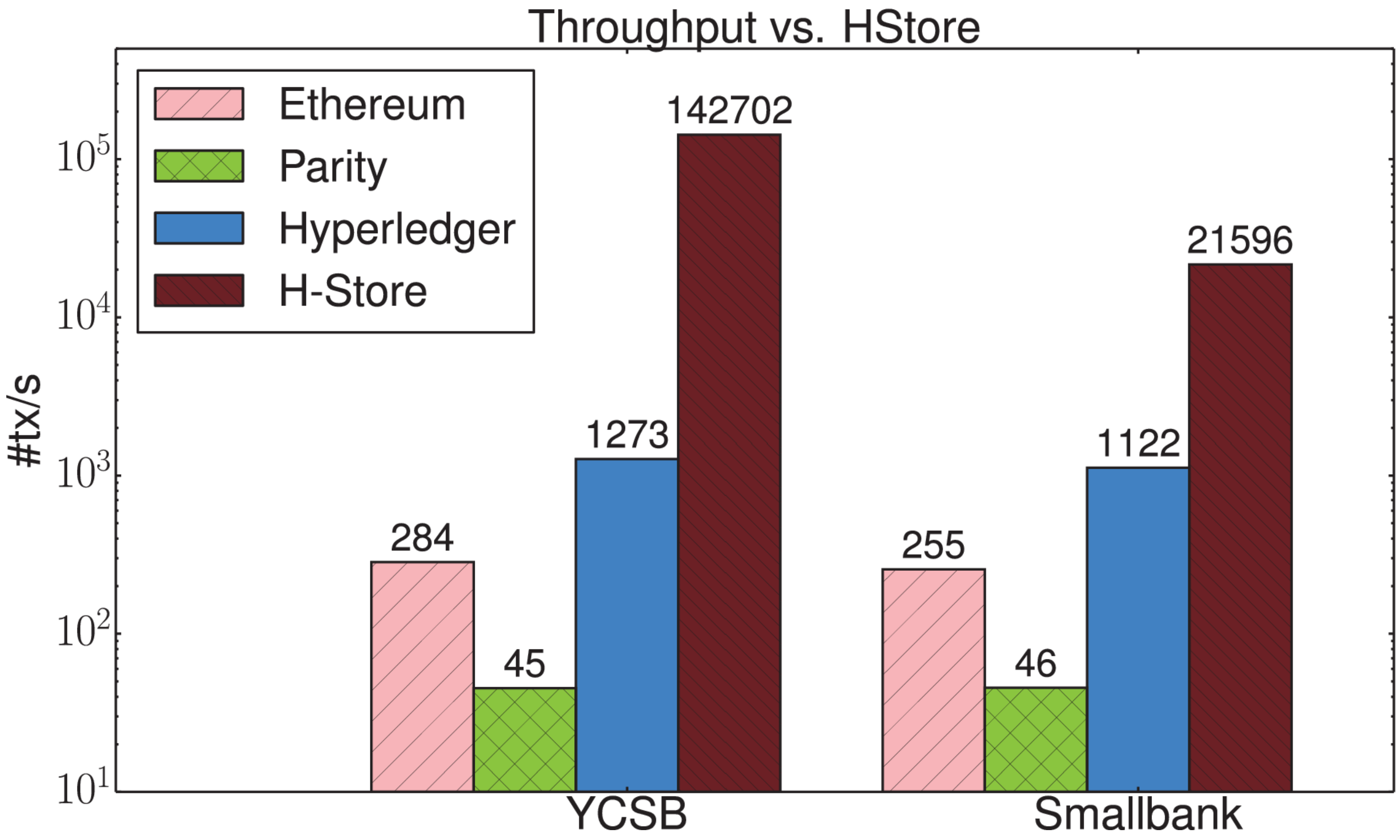
Evaluation: Macro Benchmarks

This section discusses the performance of the blockchains at the application layer, using YCSB and Smallbank benchmarks



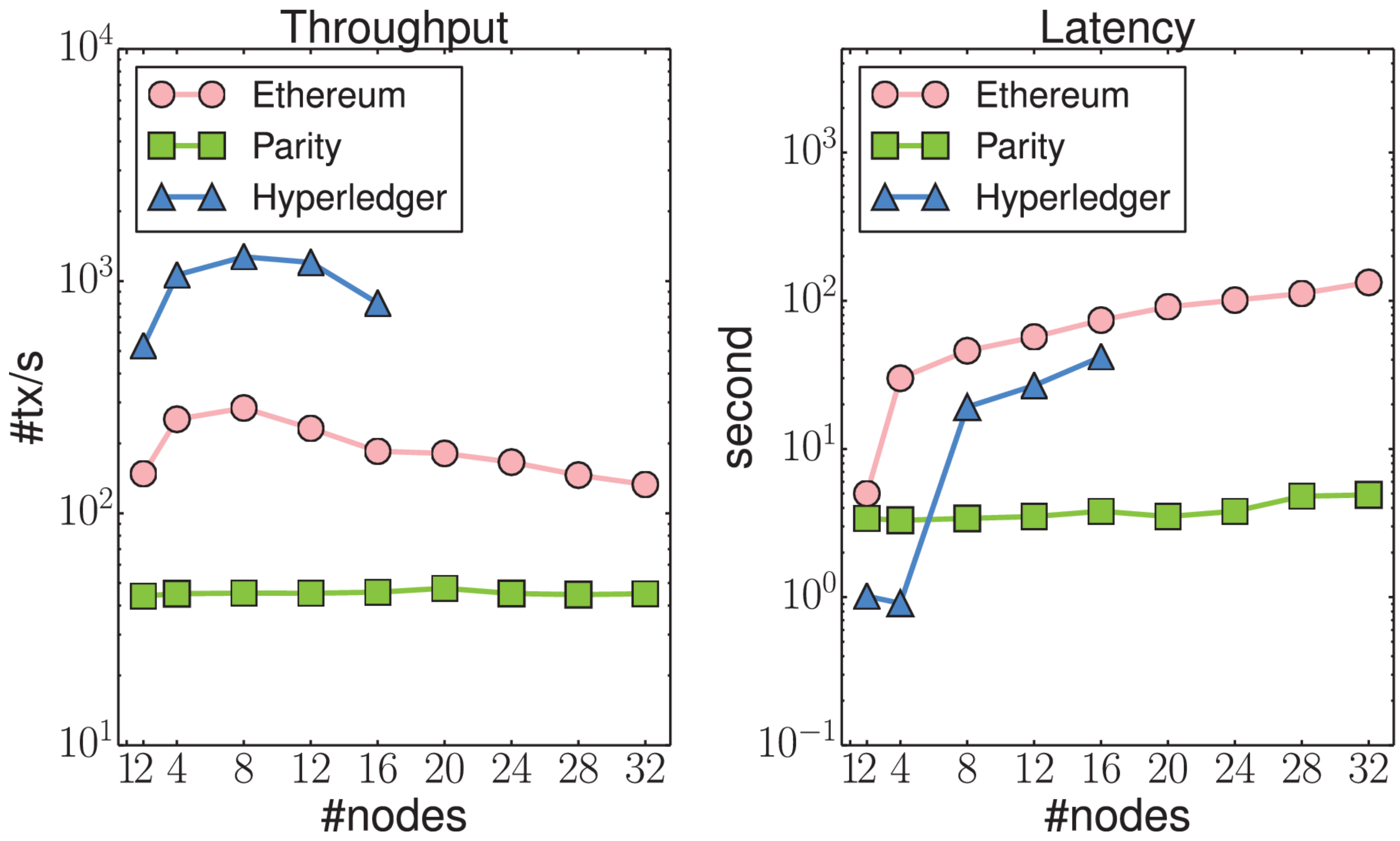
Evaluation: Macro Benchmarks

Compare the three blockchains against a popular in-memory database system, namely H-Store, using the YCSB and Smallbank workload



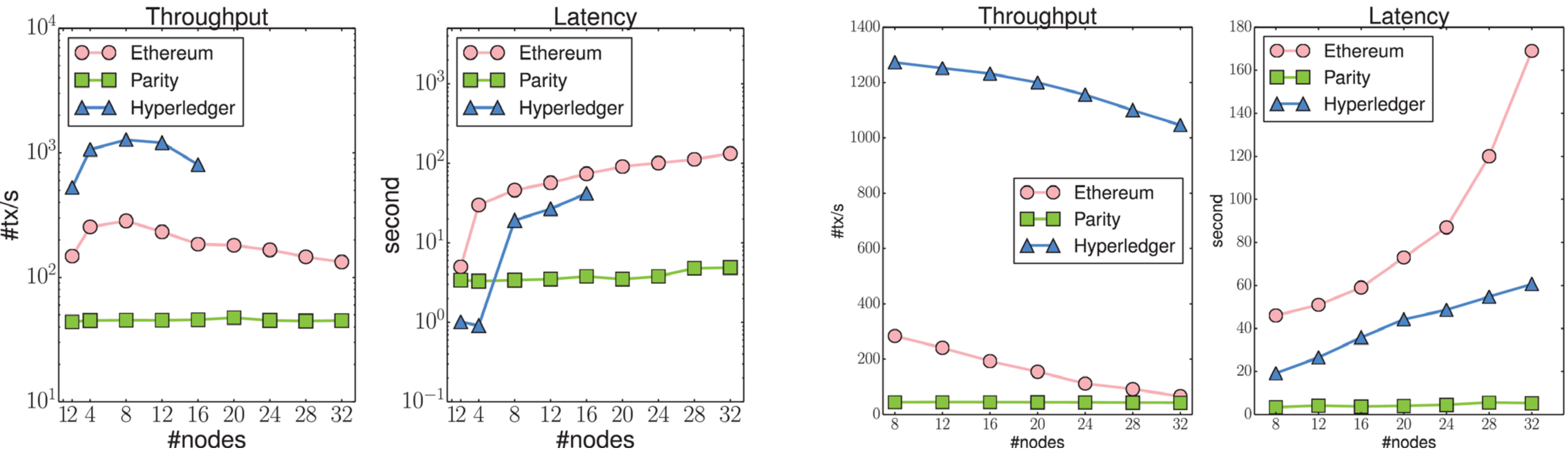
Evaluation: Macro Benchmarks

Fixed the client request rate (320 requests per second for Hyperledger, 160 requests per second for Ethereum and Parity) and increased both the number of clients and the number of servers.



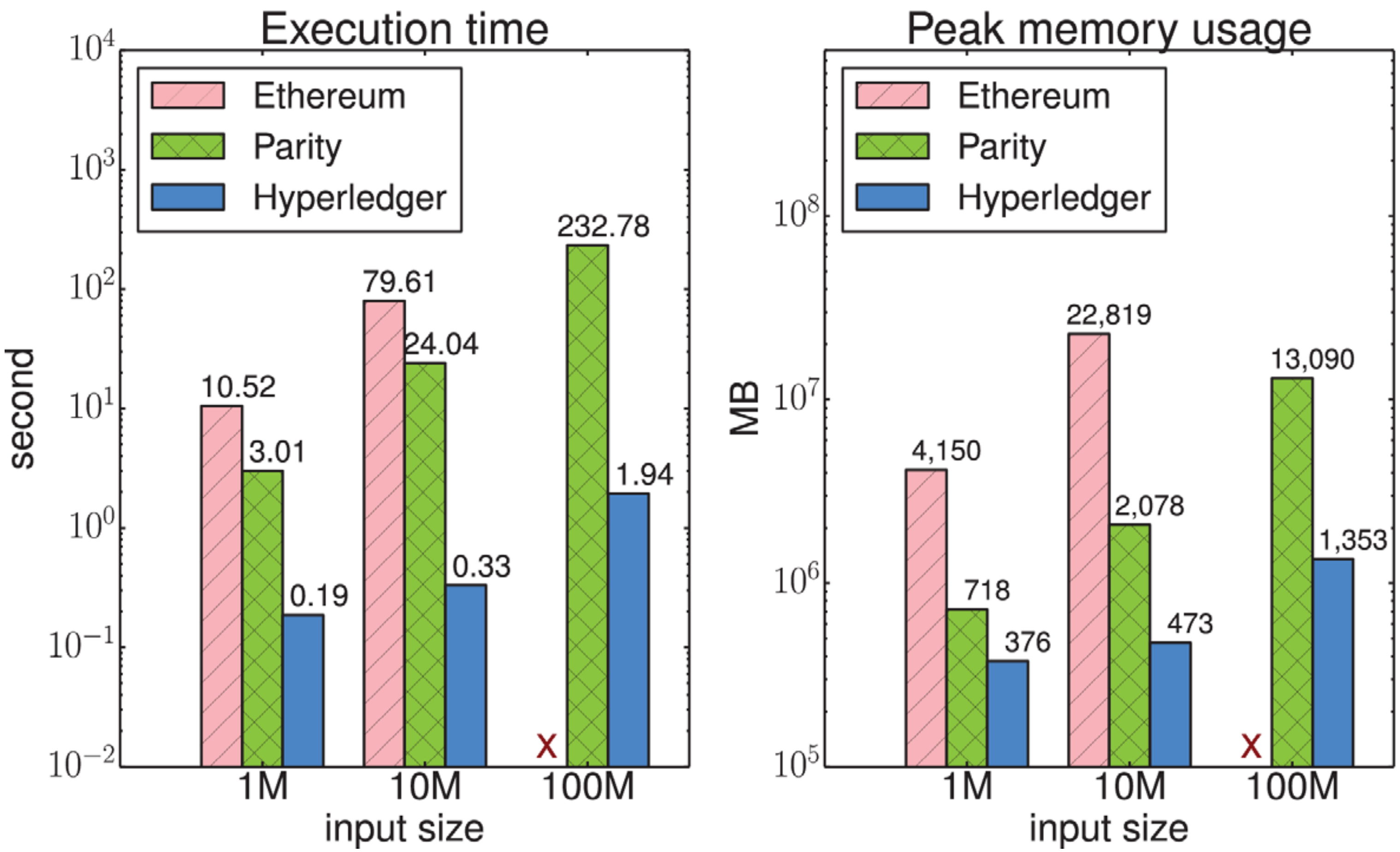
Evaluation: Macro Benchmarks

Since the original PBFT protocol guarantees both liveness and safety, we can attribute this failure to scale to Hyperledger’s implementation.



Evaluation: Micro Benchmarks

This section discusses the performance of the blockchains at the execution, data model and consensus layer. For the first two layers, the workloads were run with one client and one server. For the consensus layer, 8 clients and 8 servers were used.

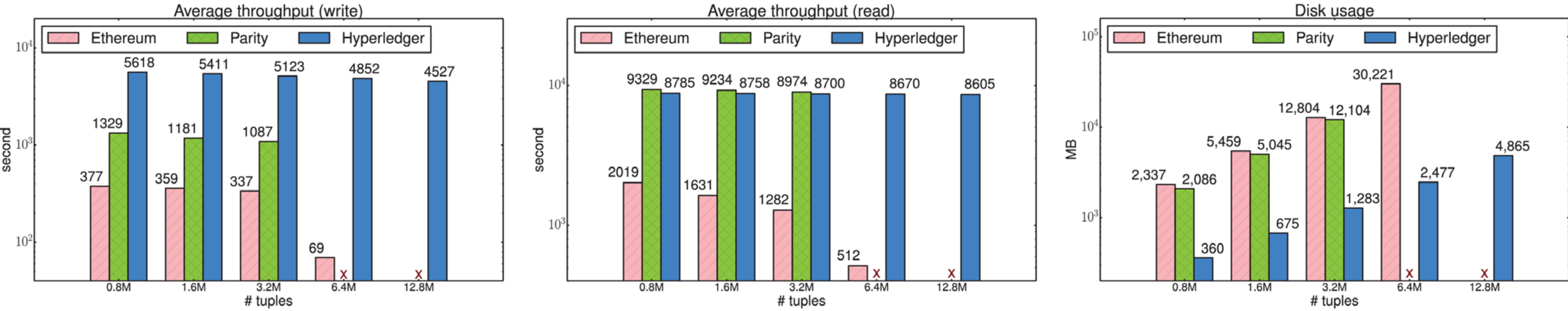


Evaluation: Micro Benchmarks

We deployed the IOHeavy smart contract that performs a number of read and write operations of key-value tuples

Ethereum and Parity use the same data model and internal index structure, therefore they incur similar space overheads

Both use an order of magnitude more storage space than Hyperledger which employs a simple key-value data model



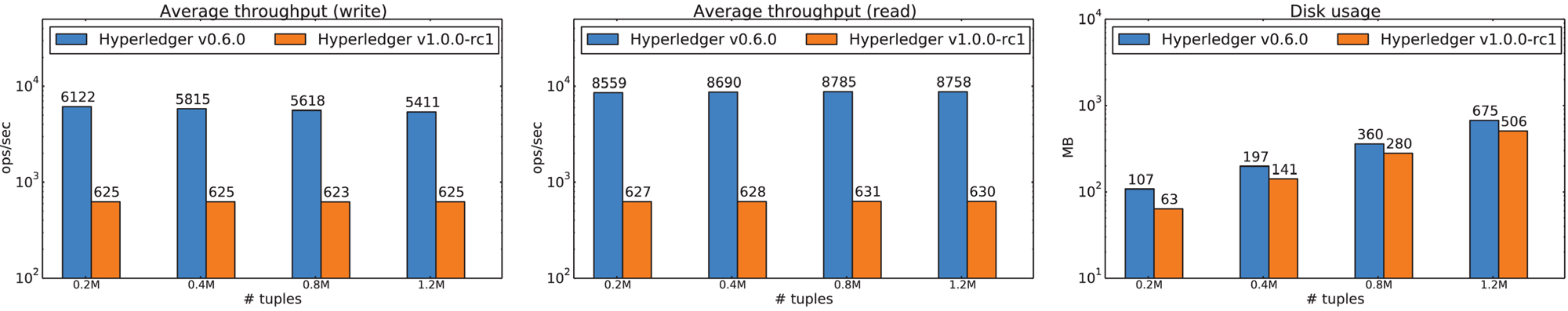
(a) Write (b) Read (c) Disk usage

Evaluation: Micro Benchmarks

For Hyperledger v1.0., using the same IOHeavy smart contract to compare I/O performance of Hyperledger version v1.0. with the older version v0.6.

With this new service, transactions in the IOHeavy workload now need to communicate with the orderer for them to be confirmed

This result suggests that replacing PBFT with a centralized service not only fails to protect the blockchain against Byzantine failures, but it may also impair the overall performance



(a) Write (b) Read (c) Read

Recap

- Provide an in-depth survey of blockchain systems. Categorize current systems along 4 dimensions:
    - Distributed ledger
    - Cryptography
    - Consensus protocol
    - Smart contract.
- Describe benchmarking framework, **BLOCKBENCH**, designed for understanding performance of private blockchains against data processing workloads.
- Present a comprehensive evaluation of
    - Ethereum, Parity and Hyperledger.
- The results show the limitation of blockchains as data processing platforms.
- Identify several performance bottlenecks, and therefore can serve as a baseline for future blockchain research and development.

Q&A

- Thanks for your listening!