Understanding Infrastructure Dependency in Blockchain Ecosystem

PhD Qualification Exam

Wei Xu

Committee: Mai Zheng, Henry J Duwe, Goce Trajcevski

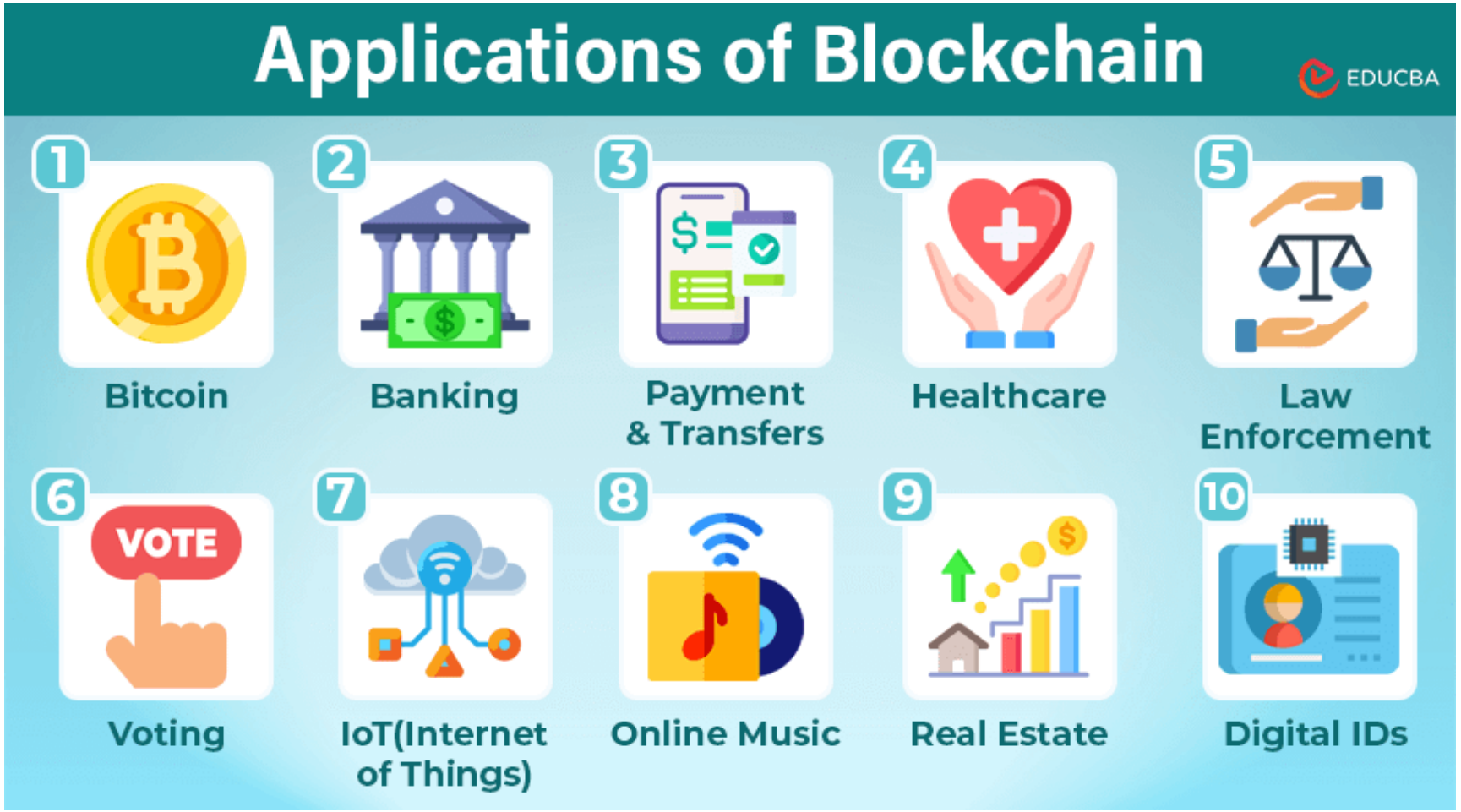
Department of Electrical Engineering

Outline

- Motivation
- Background
- Methodology
- Prototype and preliminary results
- Discussion & Future work

Motivation

Blockchain is promising in many use cases:



https://www.educba.com/applications-of-blockchain/

Blockchain is popular among enterprises:

COMPANY	TECHNOLOGIES USED *(Blockchain networks / DLT frameworks / SC language/software)*
Microsoft	bitcoin, ethereum, CONSENSYS Quorum, daml
Alphabet	THETA, HYPERLEDGER FABRIC, ethereum, Hedera Hashgraph
Alibaba Group 阿里巴巴集团	ANTCHAIN, HYPERLEDGER FABRIC, CONSENSYS Quorum
Coca-Cola	HYPERLEDGER FABRIC, baseline, ethereum
accenture	HYPERLEDGER FABRIC, corda, CONSENSYS Quorum, daml
TCS TATA CONSULTANCY SERVICES	HYPERLEDGER FABRIC, ethereum, corda
SAP	CONSENSYS Quorum, ethereum, MultiChain, corda, HYPERLEDGER FABRIC
Shell	energy web CHAIN, CONSENSYS Quorum, ethereum
citi	corda, AXONI AxCore, ripple
Goldman Sachs	corda, CONSENSYS Quorum, AXONI AxCore
SAMSUNG	Nexledger, ethereum, HYPERLEDGER FABRIC

https://changelly.com/blog/companies-that-use-blockchain/

Motivation

- Blockchain is robust thanks to the following features:
  - Immutable blocks stored in chain
  - Decentralized data stores
  - Consensus mechanisms
  - Transactions protected by cryptography

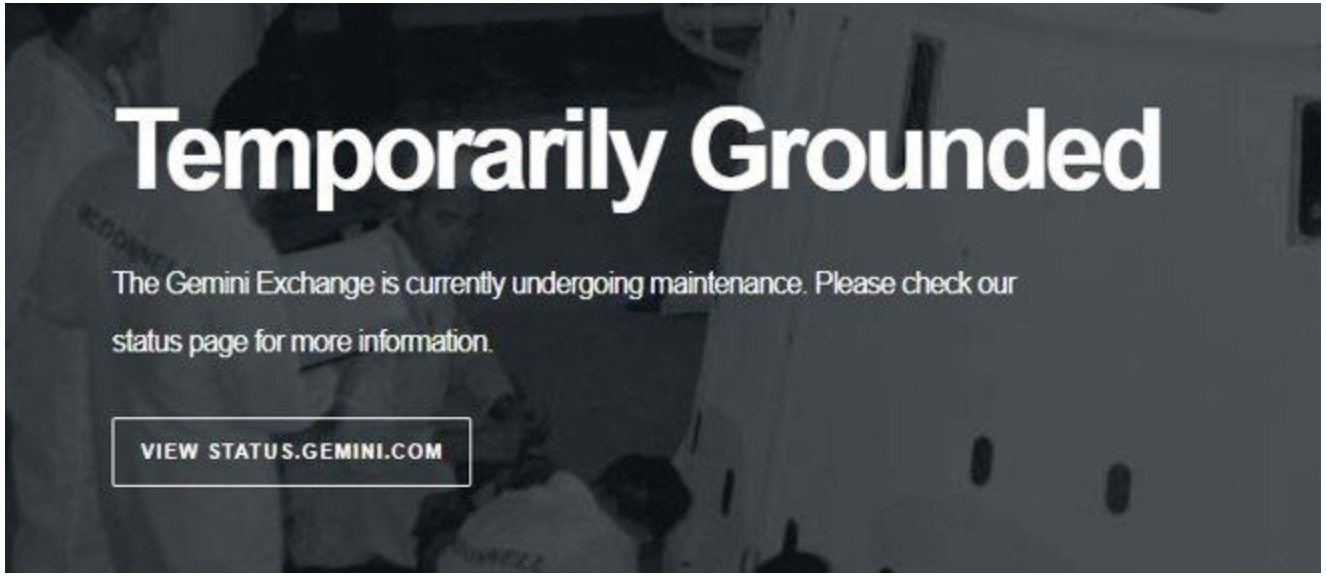
Motivation

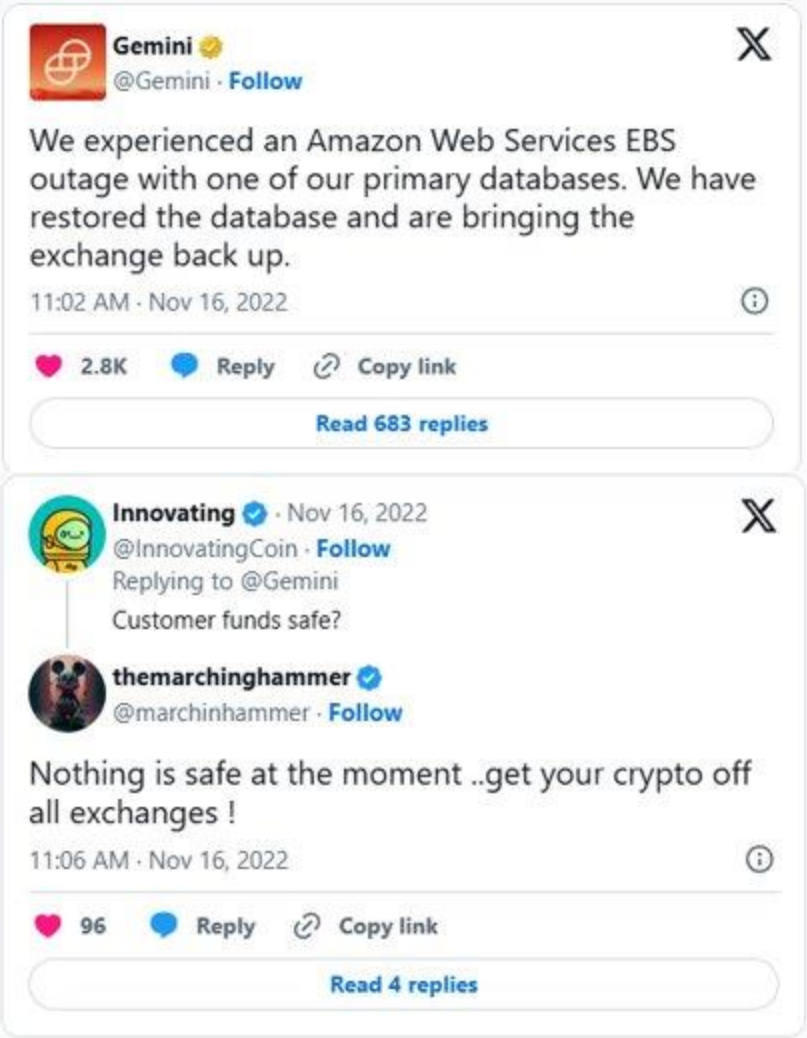
- Blockchain is robust thanks to the following features:

But are they really this robust in practice?

Motivation

- Examples of real-world problems
  - Case 1: Crypto exchange Gemini down in November 2022
    - Caused by AWS EBS outage with one of the primary databases



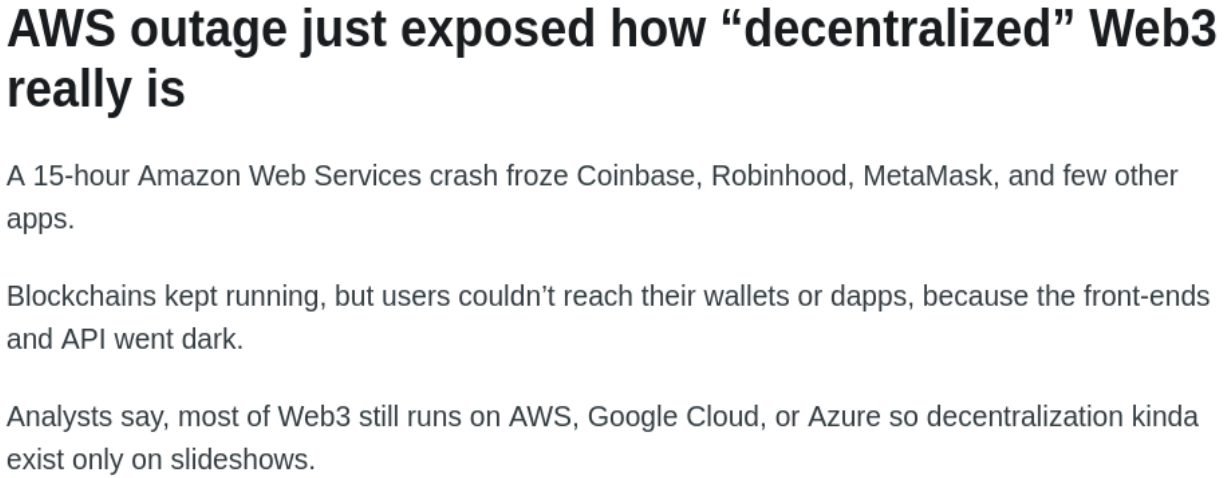


Motivation

- Examples of real-world problems
  - Case 2: AWS meltdown in Oct 2025 took down several major crypto companies/networks
    - Caused by DNS misconfigurations and DynamoDB failure, then propagated to dependent services

Crypto's 'Decentralized' Illusion Shattered Again by Another AWS Meltdown

The October AWS outage took down some of crypto's most prominent companies and networks. Many in the community pointed out their lack of decentralization.



AWS outage just exposed how "decentralized" Web3 really is

A 15-hour Amazon Web Services crash froze Coinbase, Robinhood, MetaMask, and few other apps.

Blockchains kept running, but users couldn't reach their wallets or dapps, because the front-ends and API went dark.

Analysts say, most of Web3 still runs on AWS, Google Cloud, or Azure so decentralization kinda exist only on slideshows.

Motivation

- Examples of real-world problems
  - Case 3: Solana blockchain went offline in Sept 2021 after a surge of transactions
    - Caused by validators overloaded by too heavy workload

Solana Says Bots Generated Transactions That Flooded the Network on Sept. 14

SECURITY

Solana attributed the 17-hour blackout to a denial-of-service attack targeting decentralized exchange (DEX) on Sep. 14.

According to the Solana Foundation, bots affected the Grape Protocol IDO on Raydium with around 400,000 transactions per second (TPS), causing the latest blackouts.

Solana explained the cause of the latest blackout in a blog post on Sept. 21. The team addressed its community. They noted that bots spammed the network as Grape launched its IDO on DEX Raydium last Tuesday at 12:00 UTC.

Source : https://coinquora.com/solana-says-bots-generated-transactions-that-flooded-the-network-on-sept-14/

Solana NurPhoto

- The Solana team announced on Twitter at around 02:00 ET that they were able to restart the network.
- The network crashed the day before after high transaction volumes forced a restart.
- The blockchain's sol token was down by around 5% at $160.50, having hit a record above $200 last week.

Motivation

- Examples of real-world problems
  - Case 3: Solana blockchain went

Blockchains are highly dependent on their underlying infrastructures!

We need to understand the blockchain dependency to ensure E2E robustness and performance implication

According to the Solana Foundation, bots affected the Grape Protocol IDO on Raydium with around 400,000 transactions per second (TPS), causing the latest blackouts.

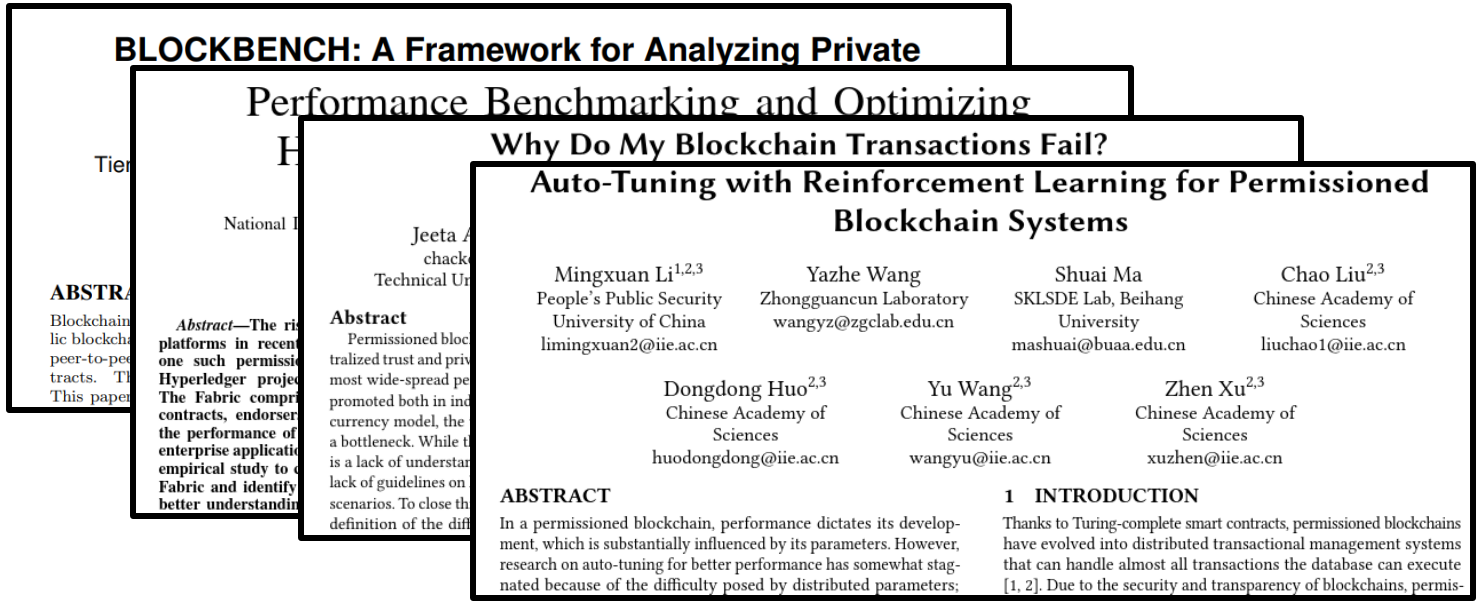
Solana explained the cause of the latest blackout in a blog post on Sept. 21. The team addressed its community. They noted that bots spammed the network as Grape launched its IDO on DEX Raydium last Tuesday at 12:00 UTC.

Source : https://coinquora.com/solana-says-bots-generated-transactions-that-flooded-the-network-on-sept-14/

- The Solana team announced on Twitter at around 02:00 ET that they were able to restart the network.
- The network crashed the day before after high transaction volumes forced a restart.
- The blockchain's sol token was down by around 5% at $160.50, having hit a record above $200 last week.

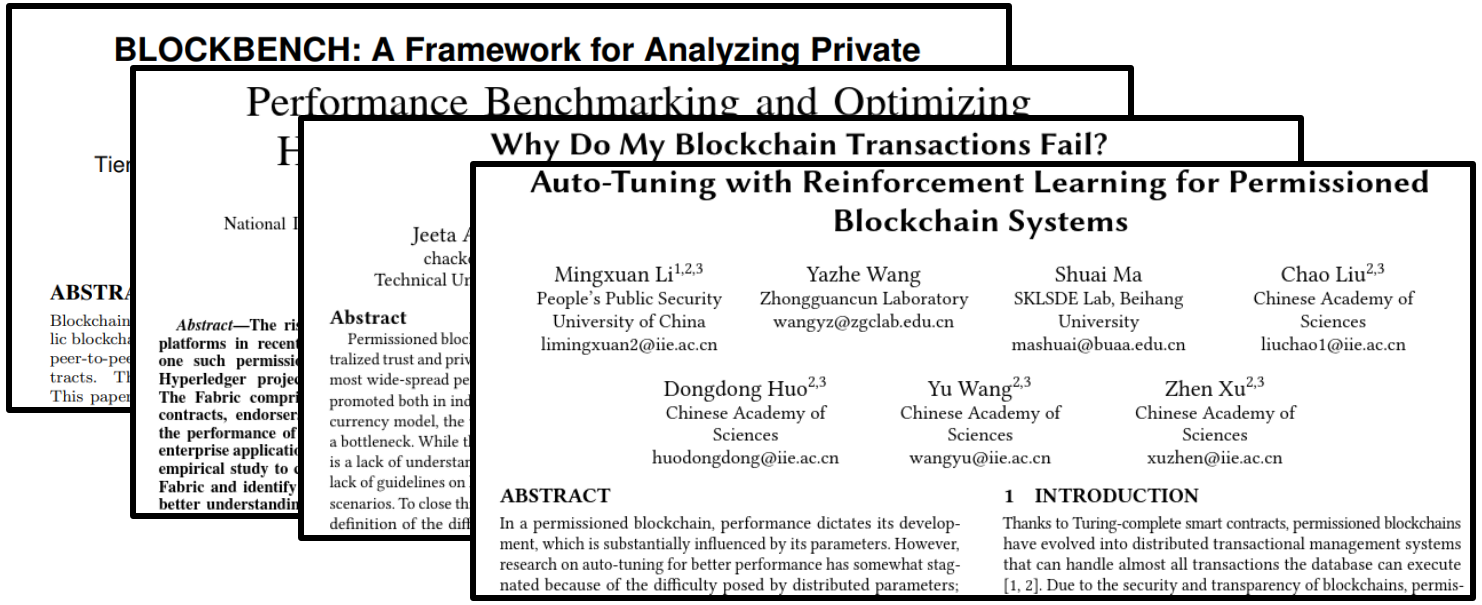
Limitations of the state of the art

- Related works:
  - Proposed the first benchmarking systems for private blockchains (Tien et al.@SIGMOD'17)
  - Observed some configurable parameters and introduced simple optimizations. (Thakkar et al.@MASCOTS'18)
  - Classified transaction failures, developed a benchmarking system, and observed dependency between block size and failures. (Chacko et al.@SIGMOD'21)
  - Proposed an auto-tuning system for performance optimization and filtered out some important configuration parameters for auto-tuning efficiency. (Li et al.@VLDB'23)



Limitations of the state of the art

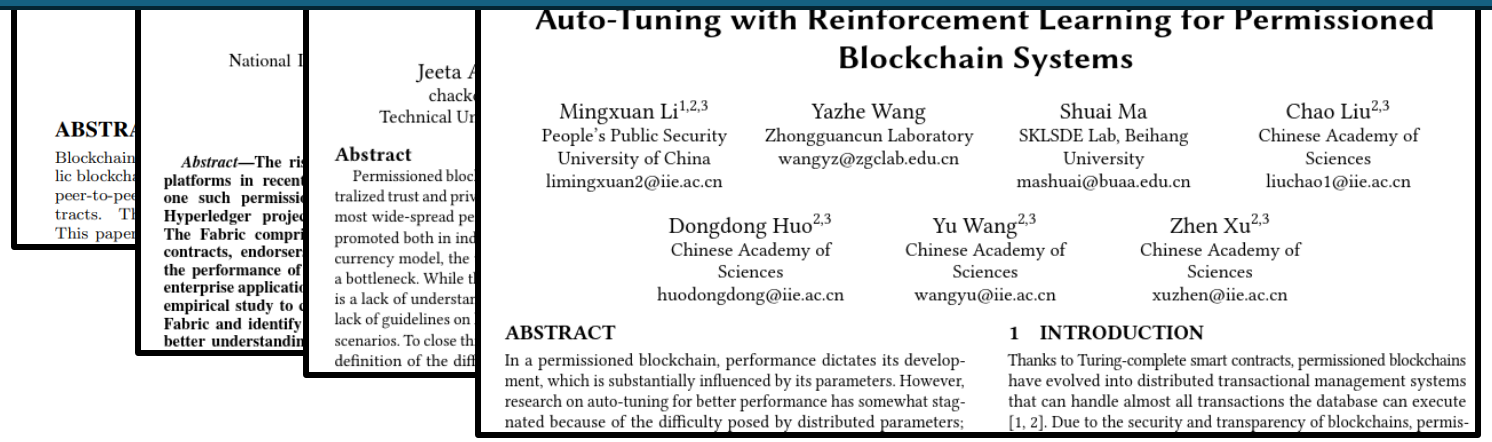
- Do not consider cross layer dependencies
  - Only focused on a limited number of configurable options in an application/blockchain layer
  - Did not consider system behavior under abnormal scenarios



Limitations of the state of the art

- Do not consider cross layer dependencies

We need a framework to study the blockchain dependency on infrastructure conveniently and thoroughly



Auto-Tuning with Reinforcement Learning for Permissioned Blockchain Systems

Mingxuan Li[1,2,3]
People's Public Security University of China
limingxuan2@iie.ac.cn

Yazhe Wang
Zhongguancun Laboratory
wangyz@zgclab.edu.cn

Shuai Ma
SKLSDE Lab, Beihang University
mashuai@buaa.edu.cn

Chao Liu[2,3]
Chinese Academy of Sciences
liuchao1@iie.ac.cn

Dongdong Huo[2,3]
Chinese Academy of Sciences
huodongdong@iie.ac.cn

Yu Wang[2,3]
Chinese Academy of Sciences
wangyu@iie.ac.cn

Zhen Xu[2,3]
Chinese Academy of Sciences
xuzhen@iie.ac.cn

ABSTRACT

In a permissioned blockchain, performance dictates its development, which is substantially influenced by its parameters. However, research on auto-tuning for better performance has somewhat stagnated because of the difficulty posed by distributed parameters;

1 INTRODUCTION

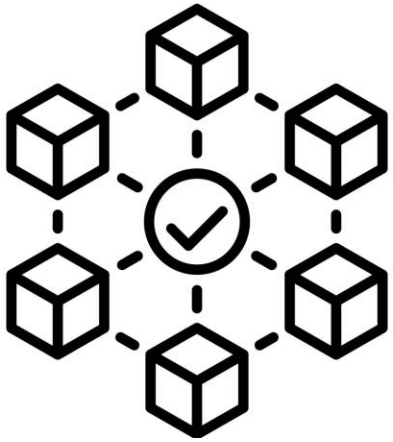
Thanks to Turing-complete smart contracts, permissioned blockchains have evolved into distributed transactional management systems that can handle almost all transactions the database can execute [1, 2]. Due to the security and transparency of blockchains, permis-

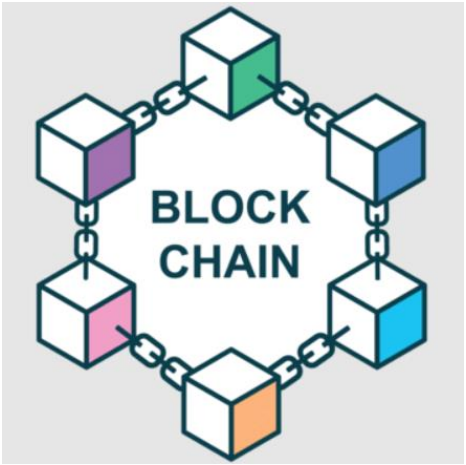
Outline

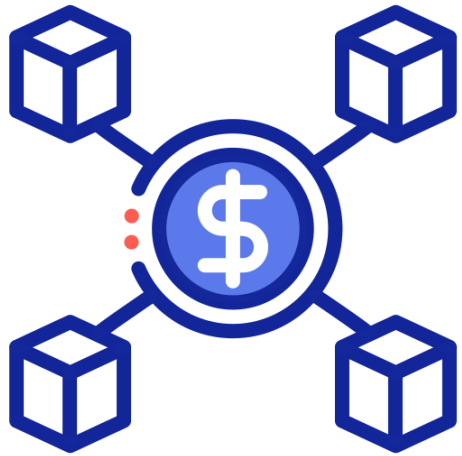
- ~~Motivation~~
- Background
- Methodology
- Prototype and preliminary results
- Discussion & Future work

What is blockchain

- An immutable transaction ledger, maintained within a distributed network of nodes
- How does blockchain works?
  - Nodes validate transactions by a consensus protocol
  - Transactions grouped into blocks connected as a chain by hash
  - Each node (peer) maintains a copy of the ledger and the blocks

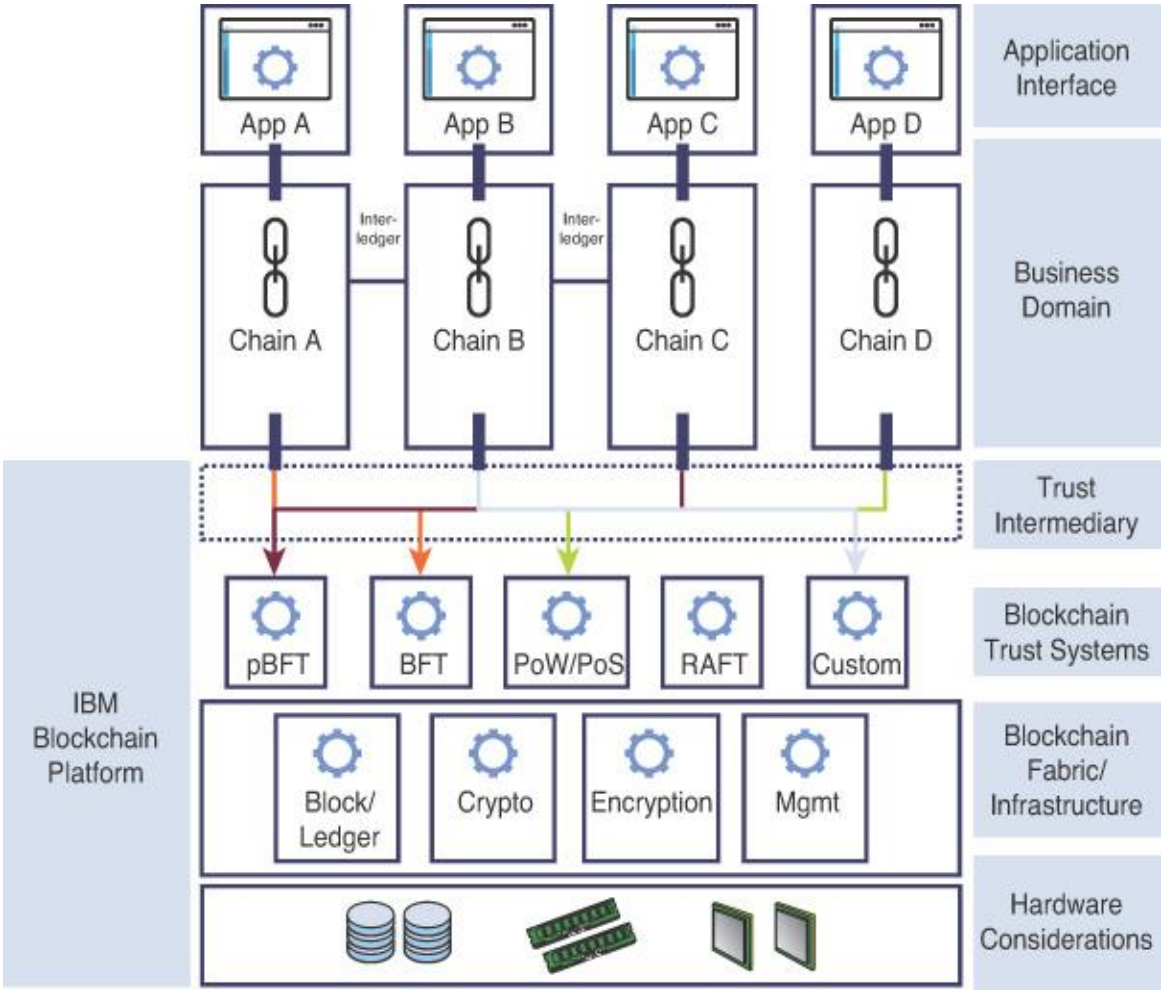






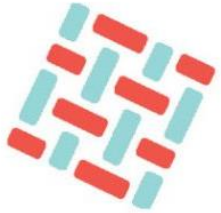
What is Blockchain ecosystem

- Blockchain is not a standalone application, relies on a complex infrastructure to run.
  - e.g., IBM blockchain infrastructure
  - Lots of config parameters
  - Configuration decides infrastructure
- Blockchain system behavior depends on ecosystem configurations



https://www.informit.com/articles/article.aspx?p=2955143

What is Hyperledger Fabric

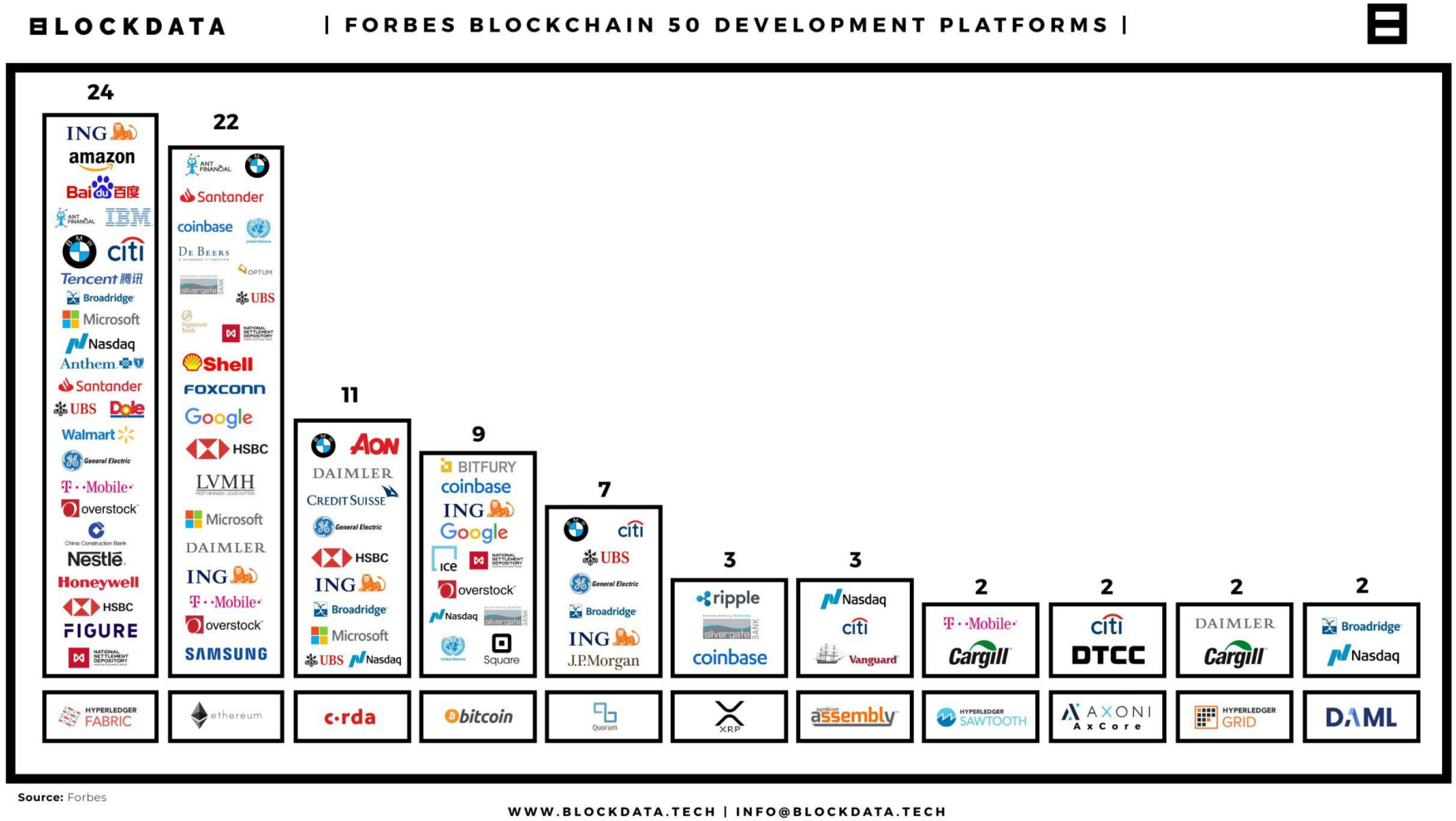


HYPERLEDGER
FABRIC

- Open-sourced, enterprise-grade distributed permissioned ledger platform
- Features:
  - E-O-V architecture that separates execution from consensus
  - No dependency on concurrencies
  - Policy-based endorsements
  - Extensible blockchain for running distributed applications

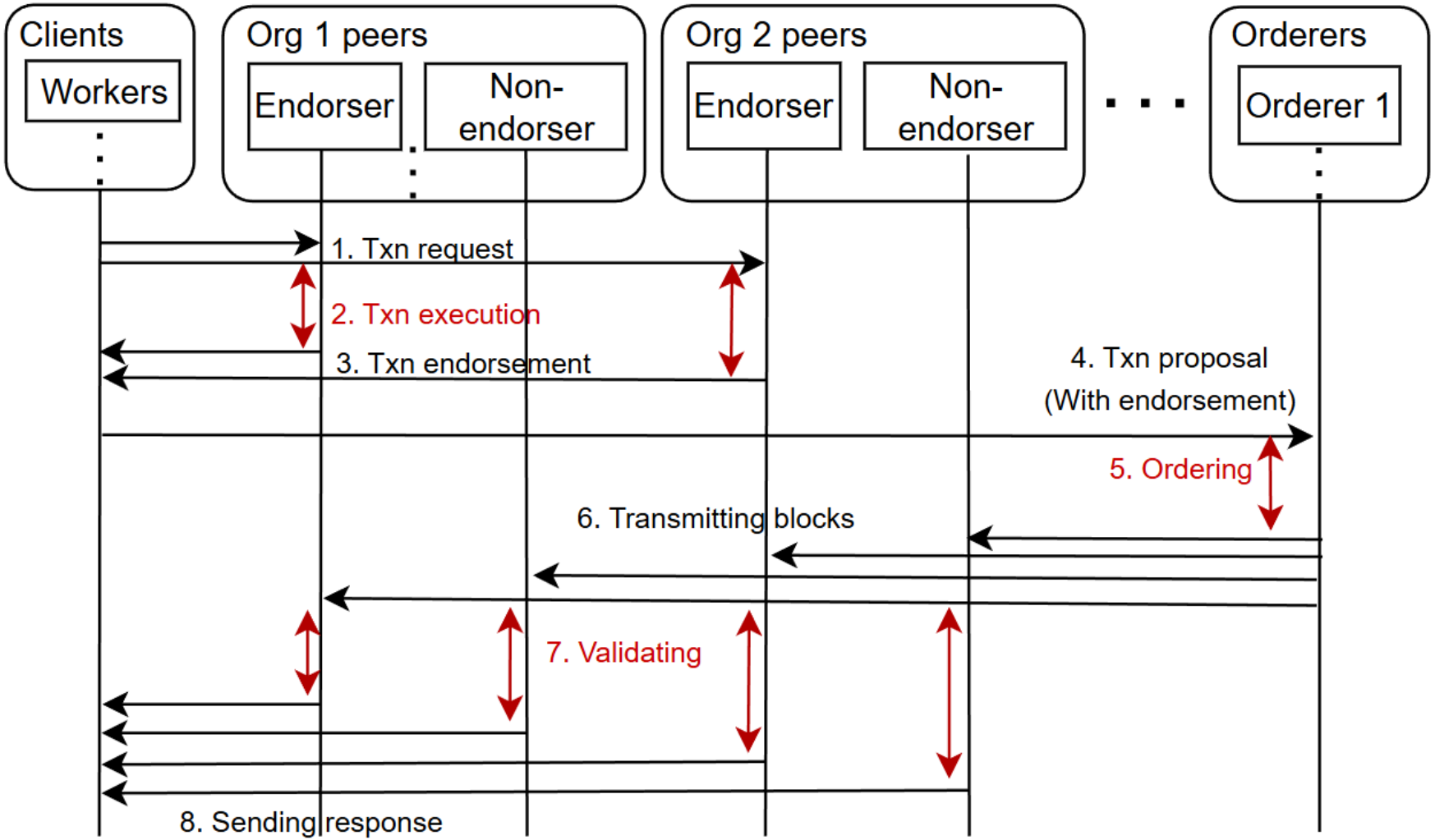
Why Hyperledger Fabric

- "Unofficial standard for enterprise blockchain platform" -IBM



https://medium.com/blockdata/forbes-blockchain-50-data-deep-dive-117fc230822f

Hyperledger Fabric Transaction workflow



Outline

- ~~Motivation~~
- ~~Background~~
- Methodology
- Prototype and preliminary results
- Discussion & Future work

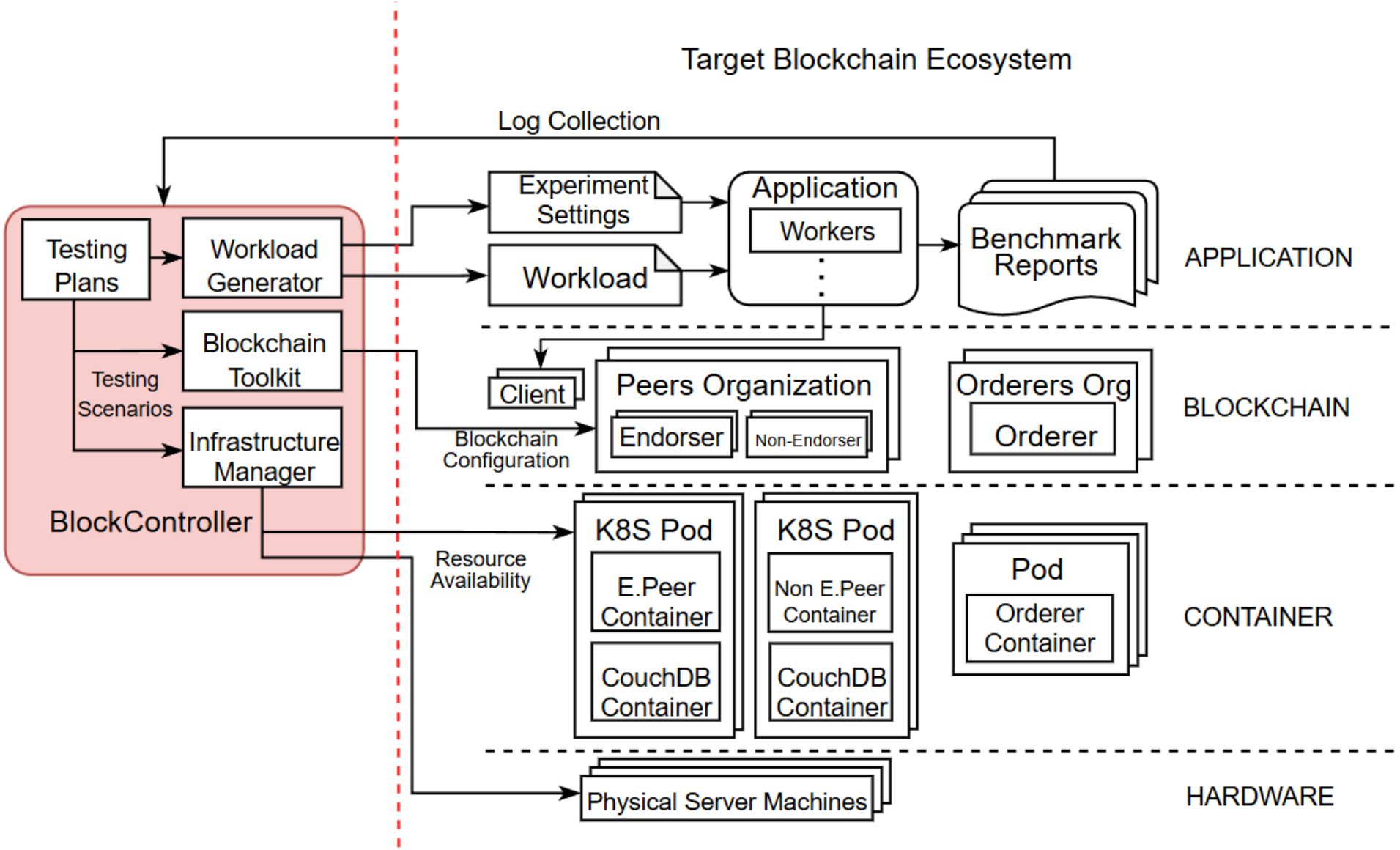
Methodology

- Research Goal:
  - Close the gap of comprehensive understanding about blockchain dependency on infrastructure
    - Dependency on infrastructure configuration
    - Performance metrics under different scenarios
      - Normal conditions with different configurations
      - Abnormal conditions with different faults

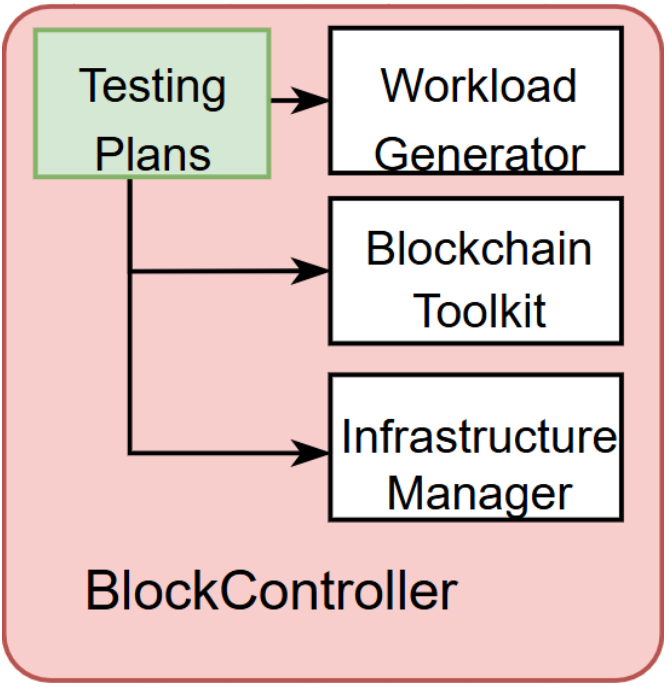
Methodology

- Research Goal:
  - Close the gap of comprehensive understanding about blockchain dependency on infrastructure
    - Dependency on infrastructure configuration
    - Performance metrics under different scenarios
      - Normal conditions with different configurations
      - Abnormal conditions with different faults
- We propose blockchain controller framework, **BlockController**
  - Automatic test network provisioning
  - Infrastructure configuration and refined benchmarking
  - Easy resource allocation

Architecture

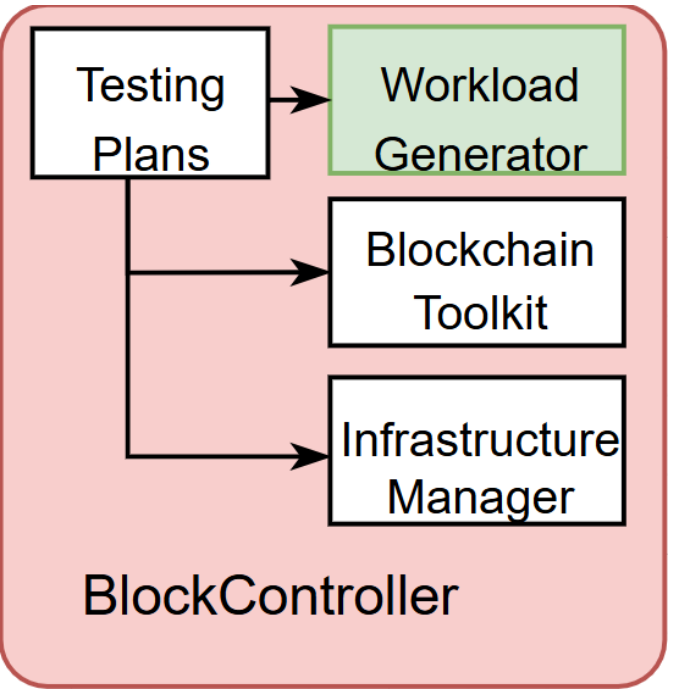


Testing Plans



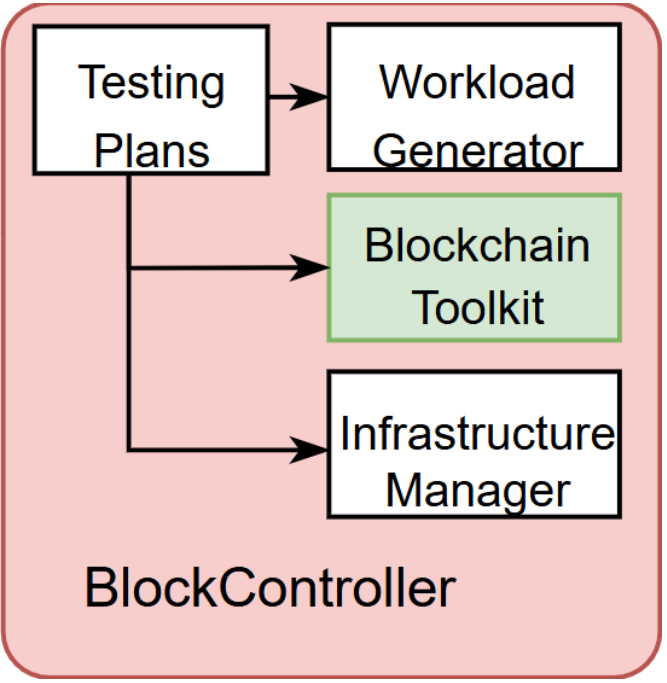
- Decide the direction of configurations
- Provide testing scenarios to simulate possible use cases:
  - Supply chain, healthcare, banking…
- Provide high level information like:
  - How many organizations/peers/endorsers… in the blockchain
  - What endorsement policy
  - Hardware resource assignment
  - What fault to inject (Extension of the current fault injection framework)
    - Not only statedb faults
    - Which layer has fault?
    - What component has fault?
    - Cross layer faults?
- The “Decision maker” of the framework

Workload generator



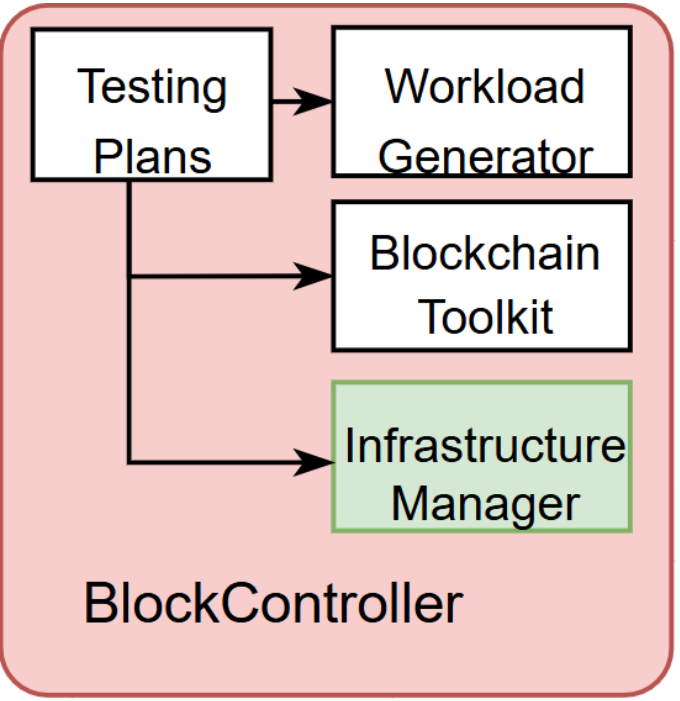
- Execute workload configurations
- Configurable parameters:
  - Number of workers
  - Number of assets
  - Transaction arrival rate
  - Read or Write operations
- Automatic log recording and better statistics

Blockchain Toolkit



- Easy blockchain network provisioning
- Automatic configuration (Decision made in Testing Plans) including the following:
  - Number of peers and their roles
  - Number of organizations
  - Endorsement policy
  - Block size
  - Consensus algorithm (RAFT, BFT)

Infrastructure Manager



- Ensure resource availability to both container layer and hardware layer
  - Automated resource allocation to each individual container and Kubernetes pod
  - Automatically assigning distinct roles to specific node
  - Straightforward application of container changes
  - Inject faults to different components

Outline

- ~~Motivation~~
- ~~Background~~
- ~~Methodology~~
- Prototype and preliminary results
- Discussion & Future work

Preliminary prototype setup

- 4 inter-connected Dell Poweredge d430 nodes on Cloudlab
  - Two 2.4 GHz 64-bit 8-core Xeon E5-2630v3 processors
  - 64 GB DDR4 RAM (8 x 8 GB modules)
  - 200 GB 6Gbps SATA SSD, 2 x 1 TB 7200 RPM 6 Gbps SATA disks
- Node list:

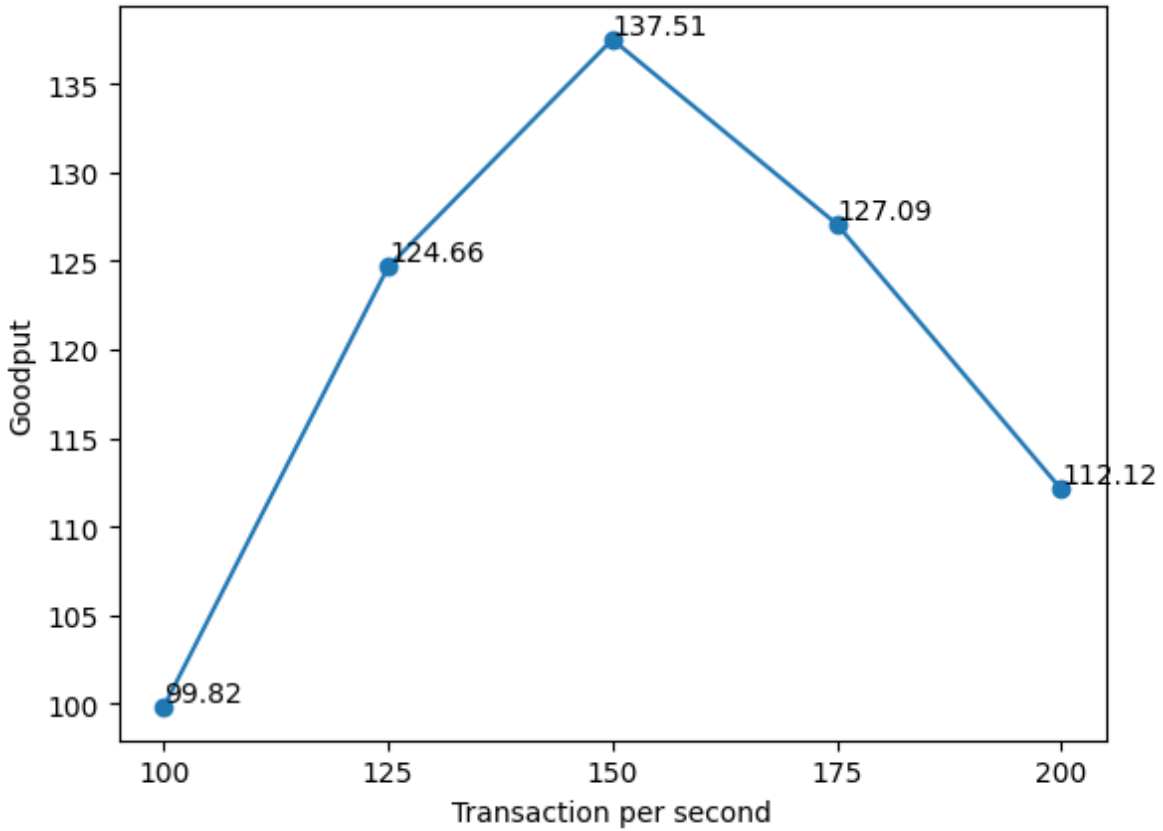
Node name	Role
node0.d430.isu-cloud.emulab.net	Control plane
node1.d430.isu-cloud.emulab.net	Org1 pods (Endorser, Non-endorser)
node2.d430.isu-cloud.emulab.net	Org2 pods (Endorser, Non-endorser)
node3.d430.isu-cloud.emulab.net	Org0 pods (Orderer)

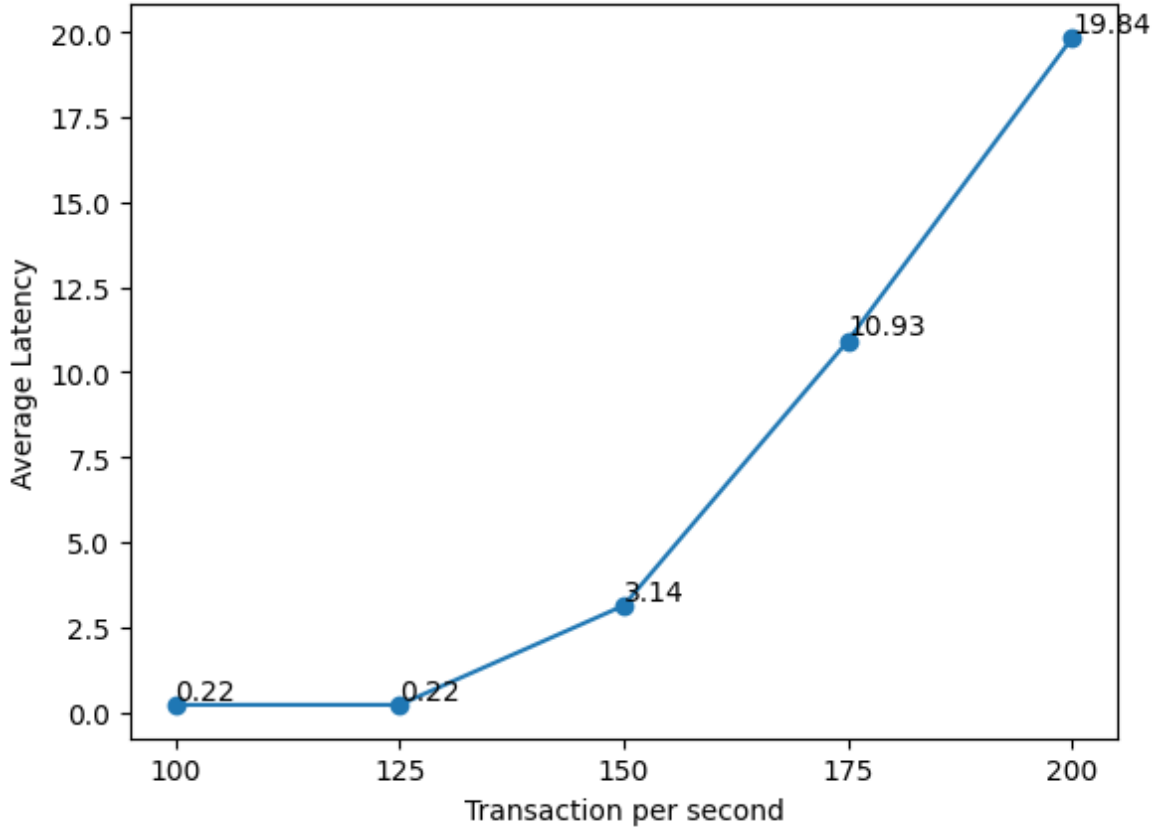
Preliminary prototype setup

- Implemented some functionalities in Blockchain Toolkit and Infrastructure Manager, including:
  - Automatic network provisioning
  - Endorser peer enrollment
  - Easy hardware resource allocation
  - Fault injection in container level
- Results: Performance under different scenarios

Preliminary experiments: Normal Scenario

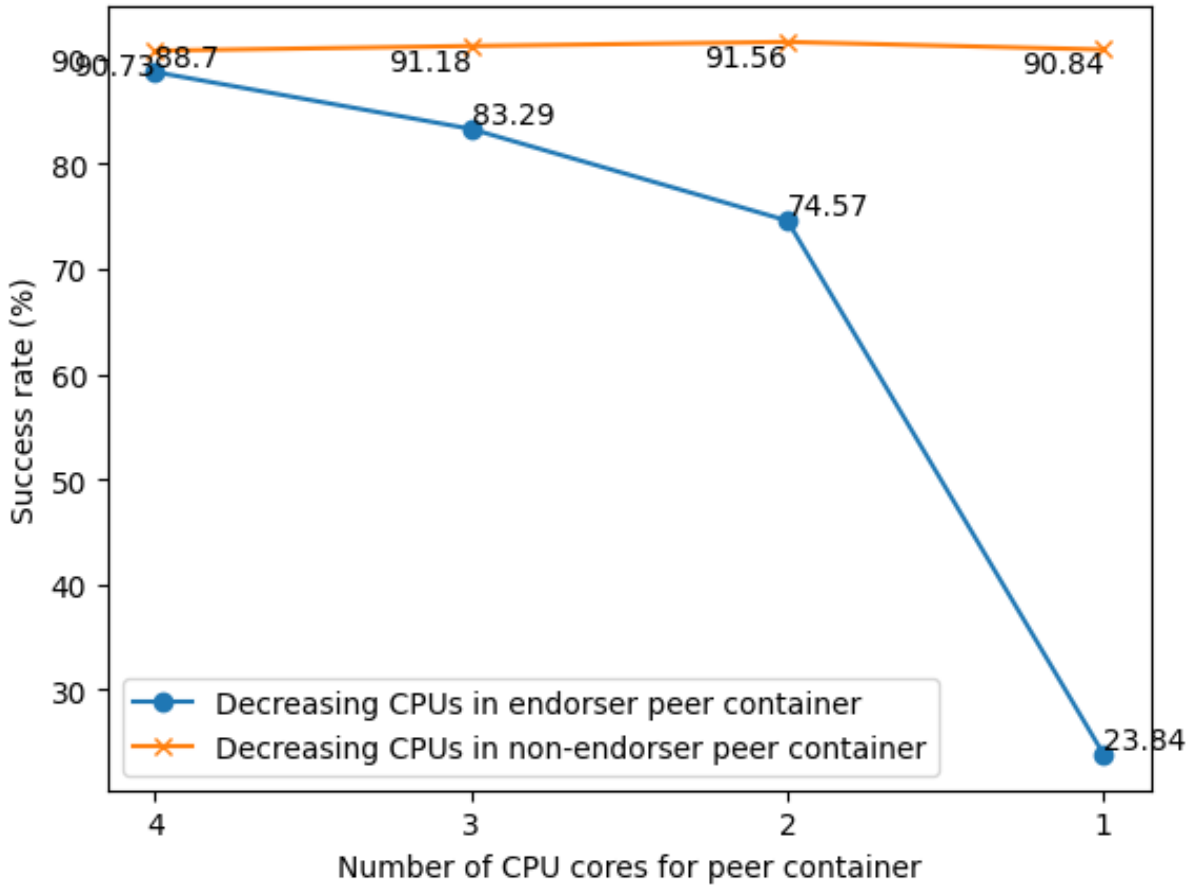
- Different configurations may lead to greatly changed performance
  - When changing the TPS, the throughput can increase by 37.76%
  - The performance change is not linear
    - Throughput decrease by 18.46%

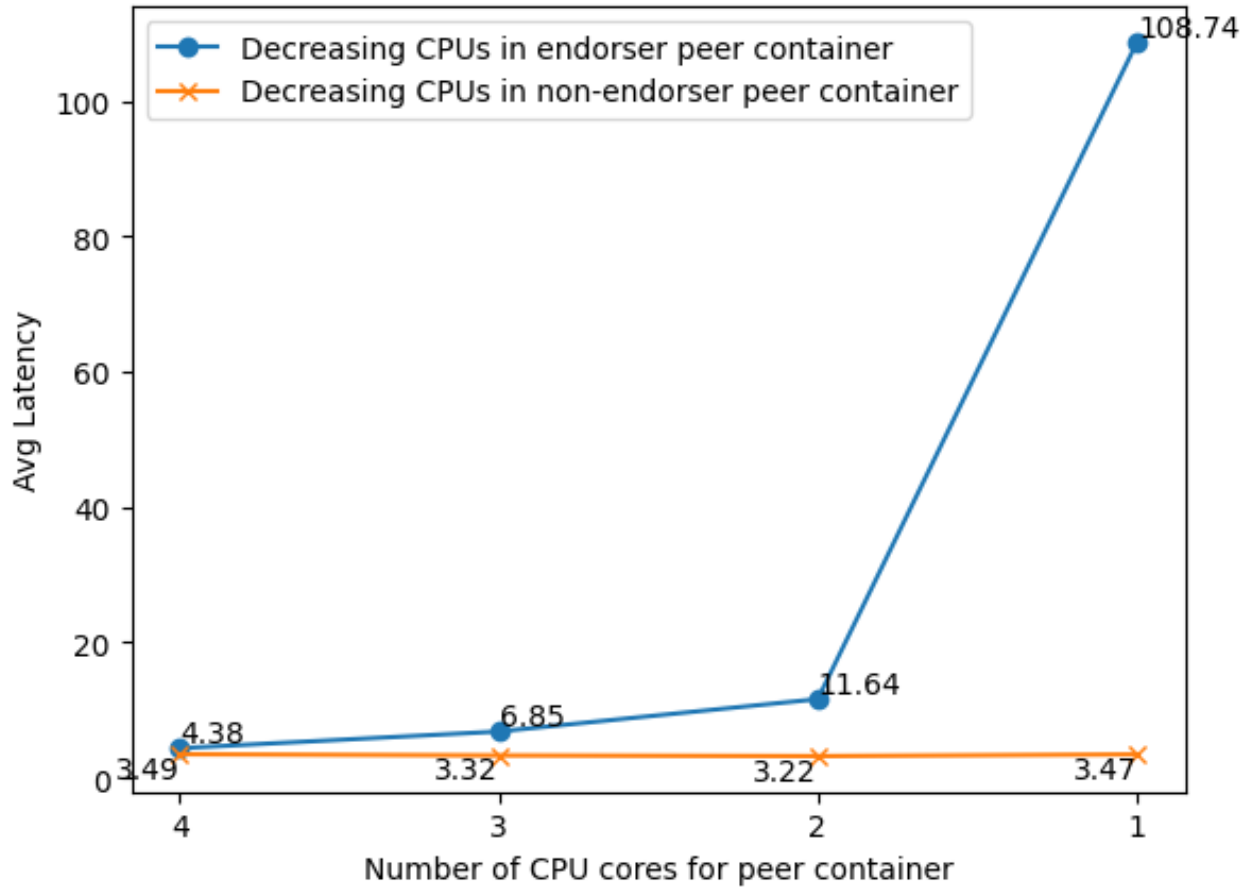




Preliminary experiments
Normal Scenario

- Different impacts when different amount of resource assigned to endorsers and non-endorsers

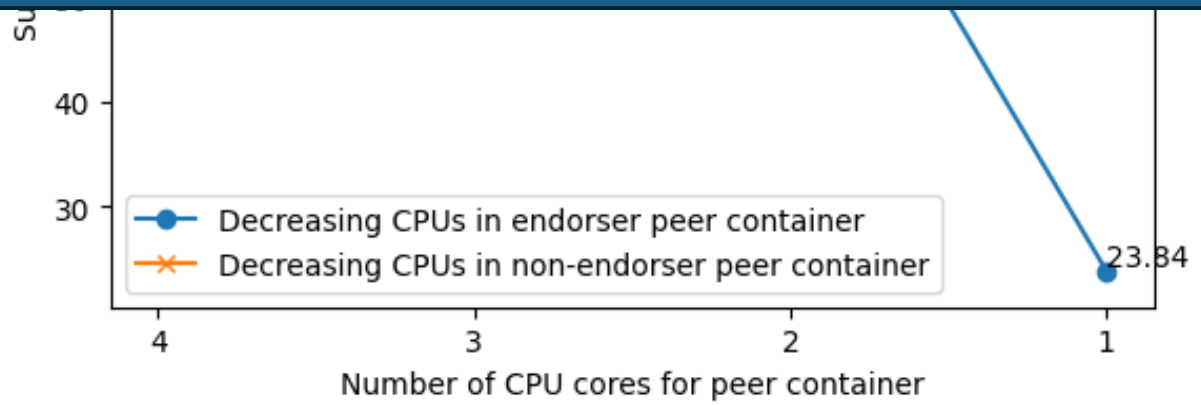


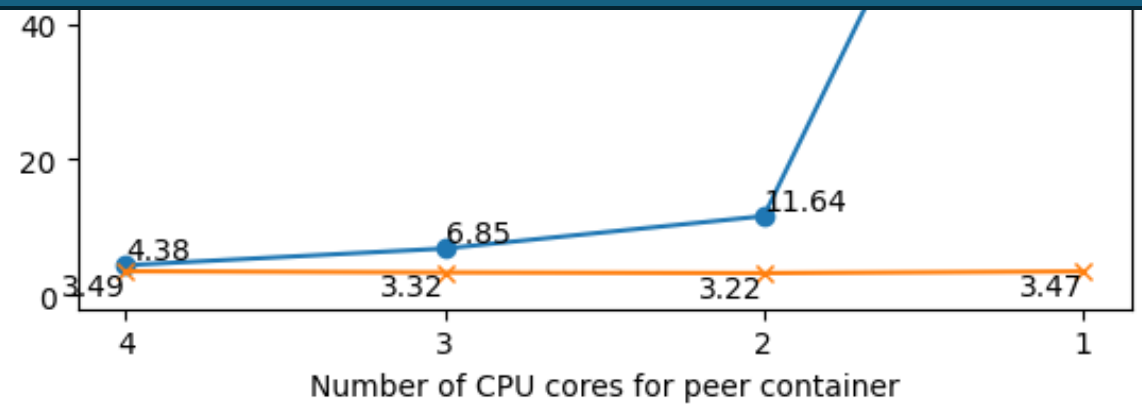


Preliminary experiments
Normal Scenario

- Different impacts when different amount of resource assigned to endorsers

Observation: Fabric transaction performance is very sensitive to endorser peers' computing power!

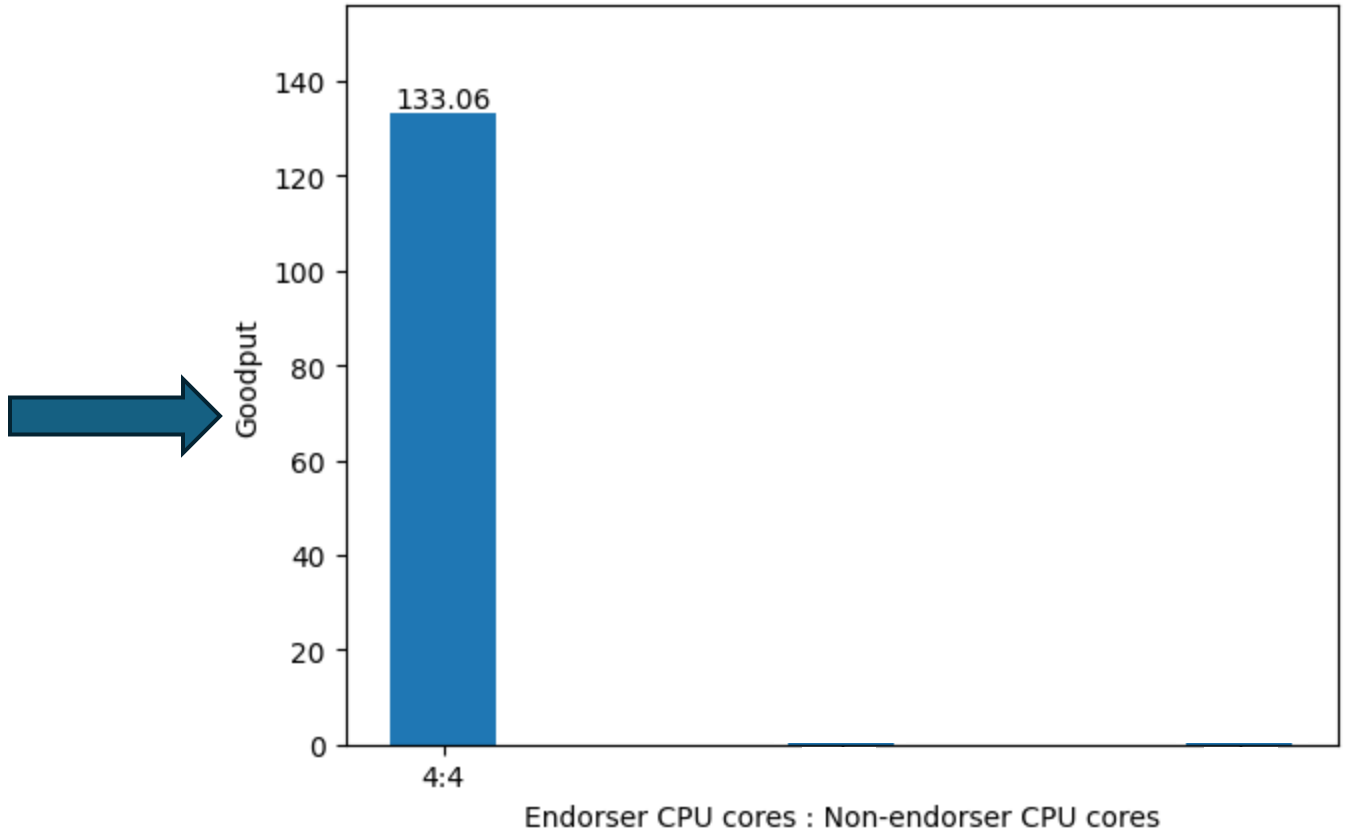




Preliminary experiments
Normal Scenario

- Keep overall CPU cores the same, distribute them unequally to endorsers and non-endorsers

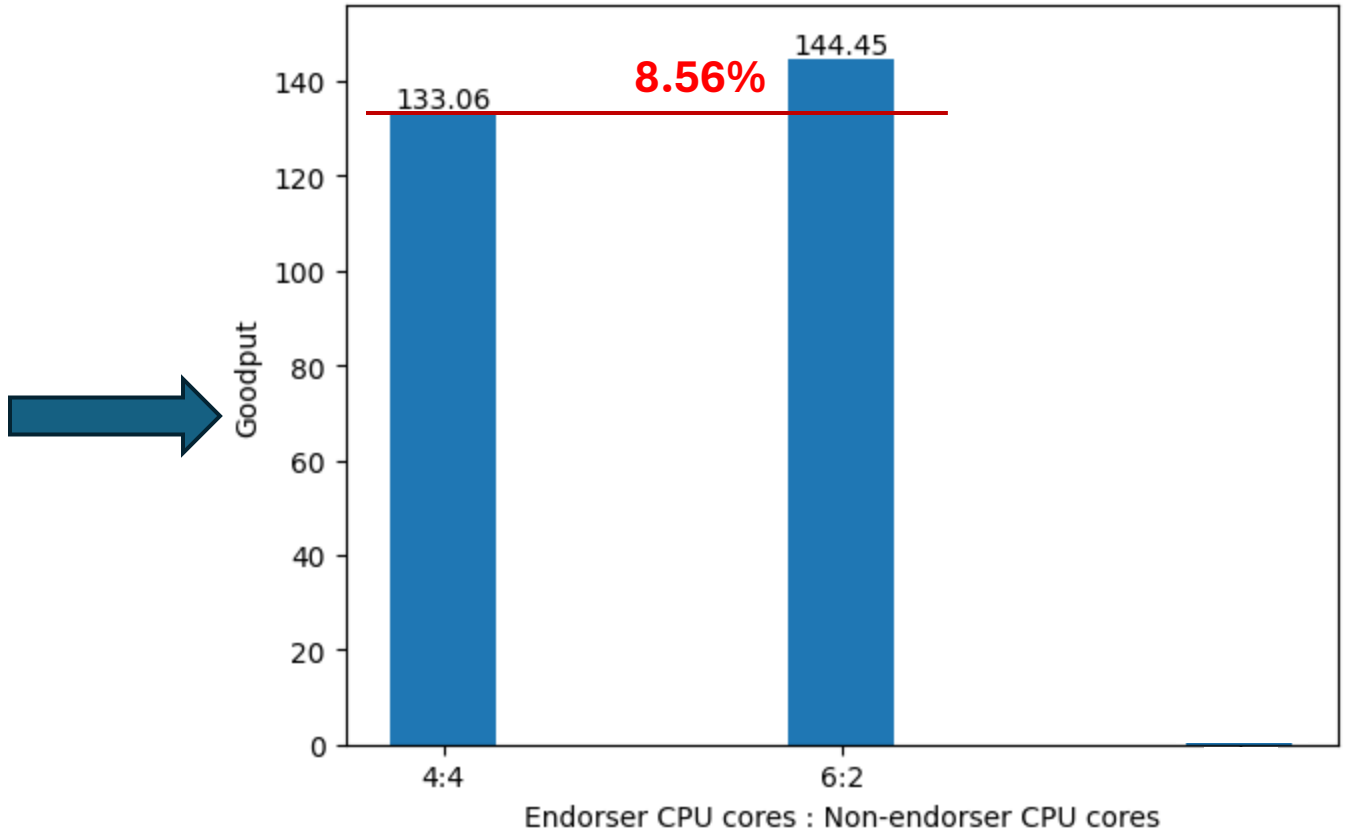
Peer container baseline configurations	
CPU cores (non-endorsing peer container)	**4**
CPU cores (endorsing peer container)	**4**
RAM (non-endorsing peer container)	**8 GB**
RAM (endorsing peer container)	**8 GB**



Preliminary experiments
Normal Scenario

- Assign 2 more cores to endorser, lead to 8.56% more goodput

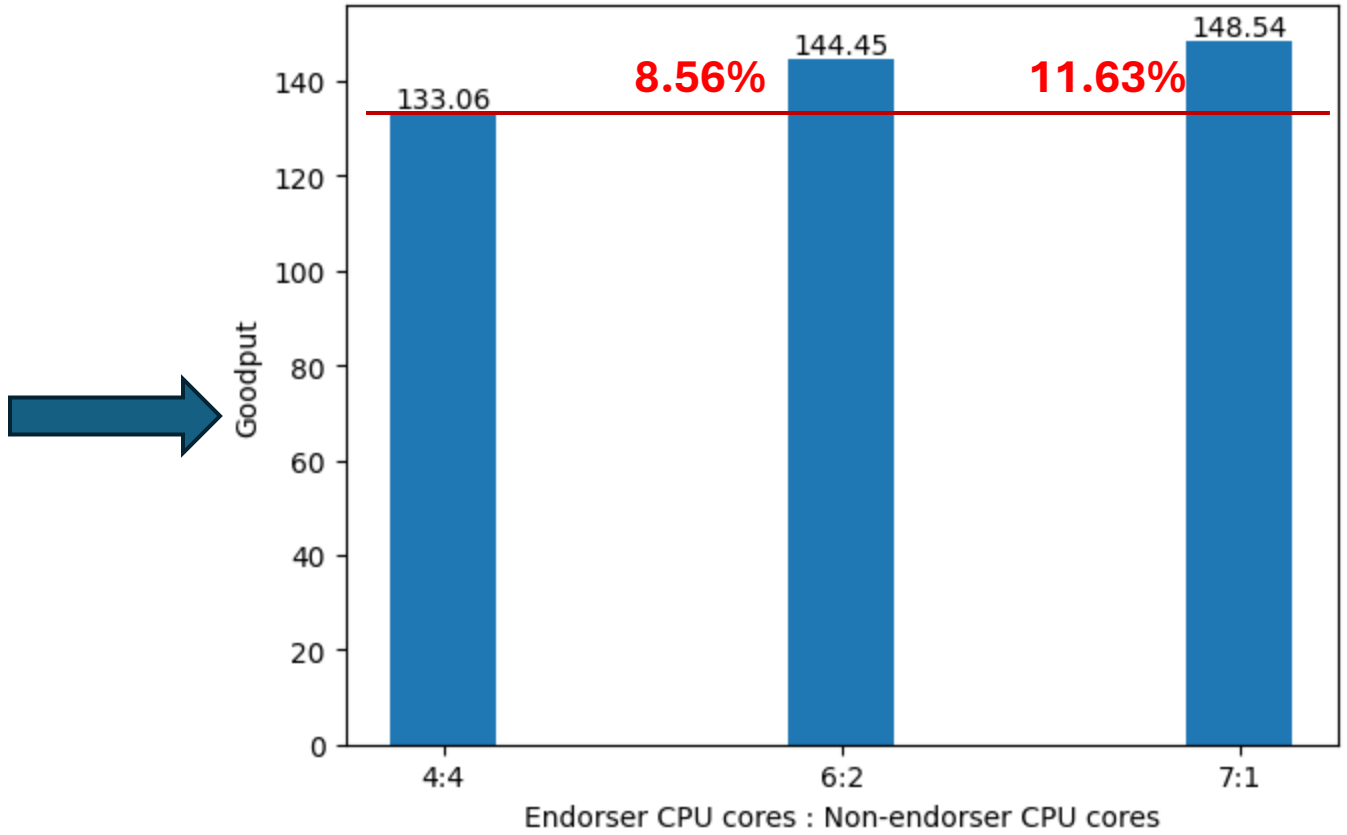
Peer container baseline configurations	
CPU cores (non-endorsing peer container)	2
CPU cores (endorsing peer container)	6
RAM (non-endorsing peer container)	8 GB
RAM (endorsing peer container)	8 GB



Preliminary experiments
Normal Scenario

- Assigning more cores to the endorser peer, the goodput almost reached the target TPS=150, with 11.63% more goodput

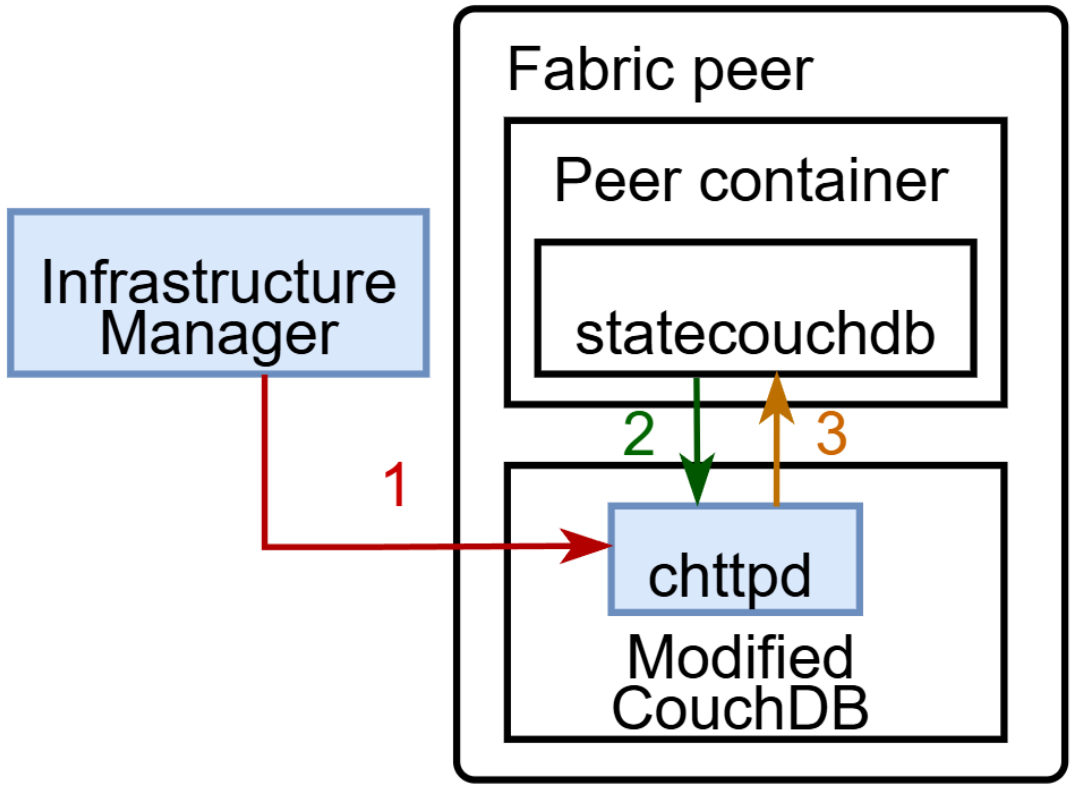
Peer container baseline configurations	
CPU cores (non-endorsing peer container)	1
CPU cores (endorsing peer container)	7
RAM (non-endorsing peer container)	8 GB
RAM (endorsing peer container)	8 GB



Preliminary experiments
Abnormal Scenario

- Inject a fault in the CouchDB container to measure performance metrics under abnormal scenarios

1. Send HTTP request to CouchDB to "switch on" the implemented faulty code.
2. CouchDB receives the request from peer.
3. CouchDB returns the implemented error message, instead of processing it correctly.



Preliminary experiments
Abnormal Scenario

- Two error codes tested: 404 and 500
- How to trigger:
  - Create the test-network using the modified CouchDB and Fabric container
  - Send HTTP request to CouchDB to toggle the error
  - Let CouchDB receive normal requests from the Fabric
- Result:
  - The peer which receives the injected fault from the CouchDB failed to join the blockchain network
  - Transactions cannot complete at all

Outline

- ~~Motivation~~
- ~~Background~~
- ~~Methodology~~
- ~~Prototype and preliminary results~~
- Discussion & Future work

Conclusion

- Designed the BlockController framework for blockchain dependency study
  - Enables fast multiple levels' configuration and integrated with benchmarking
  - Easy resource management

Future work

- Make the BlockController more "generic"
- Understand how each knob works, how these parameters change the performance
- Focus more on infrastructure
  - Reason: Infrastructure level change is less explored
- Investigate conflicts among different validators sharing hardware resource
  - Understand the conflicts → Mitigate the conflicts
- Start with existing setups (Related works)
  - See if their conclusion can still hold with changing configurations in infrastructures.

Future work (cont'd)

- Explore bottlenecks
  - e.g., What part of the infrastructure may become bottleneck, What case will exhaust hardware resource
  - Research questions like "When the infrastructure can be bottleneck/underutilized"
- Explore resource unavailability
  - Performance/System behavior under abnormal cases
  - Example: Network partition as an extreme case of availability issue
    - Need larger scaled testbeds

Outline

- ~~Motivation~~
- ~~Background~~
- ~~Methodology~~
- ~~Prototype and preliminary results~~
- ~~Discussion & Future work~~

THANK YOU! & QUESTIONS?

Baseline Configurations – Application layer

Benchmark layer baseline configuration	
Number of workers	**10**
Number of assets each worker creates	**500**
Transaction Duration (sec)	**600**
Transaction send rate	**100 TPS**
Read:Update ratio in workload	**0% Read, 100% Update**

Baseline Configurations – Blockchain layer

Blockchain layer baseline configurations	
Fabric version	2.5
Chaincode	asset-transfer-basic
StateDB	CouchDB
totalQueryLimit	100000
MaxMessageCount	500
AbsoluteMaxBytes	10 MB
Number of Organizations	3
Number of peers per Organization	2
Number of endorsers per Organization	1

Conclusion & Future work

- Designed the BlockController framework for blockchain dependency study
  - Enables fast multiple levels' configuration and integrated with benchmarking
  - Easy resource management
- Timeline for the future works
  - Late Dec: Workload classification based on typical use cases
  - Jan 2026: Simplified fault injection process and corresponding error cases
  - Feb 2026: Better peer configuration/scaling
  - May 2026: Automated configuration optimization (Examples: LLM or other tools, etc)
  - And then: Extend the framework to other blockchains

Baseline Configurations – Kubernetes Layer

Kubernetes layer baseline configurations	
CPU cores assigned to non-endorsing peer container	5 cores
CPU cores assigned to endorsing peer container	5 cores
RAM assigned to non-endorsing peer container	5 GB
RAM assigned to endorsing peer container	5 GB
Total number of CPU cores assigned to endorsing peer	10 cores
Total size of RAM assigned to endorsing peer	10 GB
Total number of CPU cores assigned to non-endorsing peer	10 cores
Total size of RAM assigned to non-endorsing peer	10 GB

Baseline Configurations – Hardware layer

Hardware layer baseline configuration	
Total number of CPU cores per machine	16
CPU base speed	2.4 GHz
RAM size per machine	64 GB
Storage medium	1 SATA SSD, 2 HDDs
Total disk space	2200 GB
Number of drives	3
Bandwidth	941 Mbits/sec

- Initial testbed implementation
  - Preliminary controller module to provision resources
  - Policy is based on domain knowledge
  - Describe fault injection module (In Jobayer's project)
  - Performance measurement and fault injection experiments
    - Show different results in these two parts

Methodology

- Research Goal:
  - ~~For different configurations in different layers, examine what kind of impact that tuning each parameter will bring to the performance.~~
  - (Cover the dependency, refer to the SIGMOD paper, be more specific)
- How to measure the performance:
  - Metrics:
    - Goodput (Effective throughput)
    - Success rate
    - Latency (Max, Min, Avg)
    - Types of transaction errors and their percentage
- Solution: E2E testbed for better dependency studies

Motivation

- ~~What makes blockchain popular~~
- Just tell why ppl like it/Show blockchain is robust
- Don't ask question in the first point, show common beliefs that ppl believe blockchain is secured, immutable
  - Enhanced security
  - Increased efficiency and speed
  - ~~Automation~~ Don’t show irrelevant things here
  - Show ppl assume blockchain is stable
- But are they really robust enough?
  - Show a box like future slides

Preliminary experiments
~~Optimize~~ performance

- When the TPS=150, goodput reached max value 137.51
  - Strategy: Assign more CPU cores to endorser peer, less CPU cores to non-endorser peers

Peer container baseline configurations	
CPU cores (non-endorsing peer container)	4
CPU cores (endorsing peer container)	4
RAM (non-endorsing peer container)	8 GB
RAM (endorsing peer container)	8 GB

Resource Manager

Peer container baseline configurations	
CPU cores (non-endorsing peer container)	2
CPU cores (endorsing peer container)	6
RAM (non-endorsing peer container)	8 GB
RAM (endorsing peer container)	8 GB

Methodology

- How to do the experiments?
- What's the challenge in here?
  - Configuration space is huge (Put early in the motivation)
    - Identify some important configurations
  - Need to have justification
  - A problem to how to config the blockchain
  - Machine learned tuning
  - LLM AI agents to tune
    - Before doing this, we need some understanding on different impacts
    - Logic chain
    - Update the motivation part as well
  - Describe what we have done

Other existing challenges keep it backup slide

- Configuration space too huge
  - Hard to find the important configurations
  - "The performance boost observed by tuning Fabric parameters is merely the tip of the iceberg" (Li et al.@VLDB'23)
- Configurating parameters is inconvenient
  - Configuration files scattered at many places
  - Inefficient to configure manually
    - Existing auto-tuning solutions also cannot cover cross-layer dependencies

Other existing challenges

- Configuration space too huge

How to automatically apply configurations while cover the dependencies?

  - Existing auto-tuning solutions also cannot cover cross-layer dependencies

Discussions & Future work

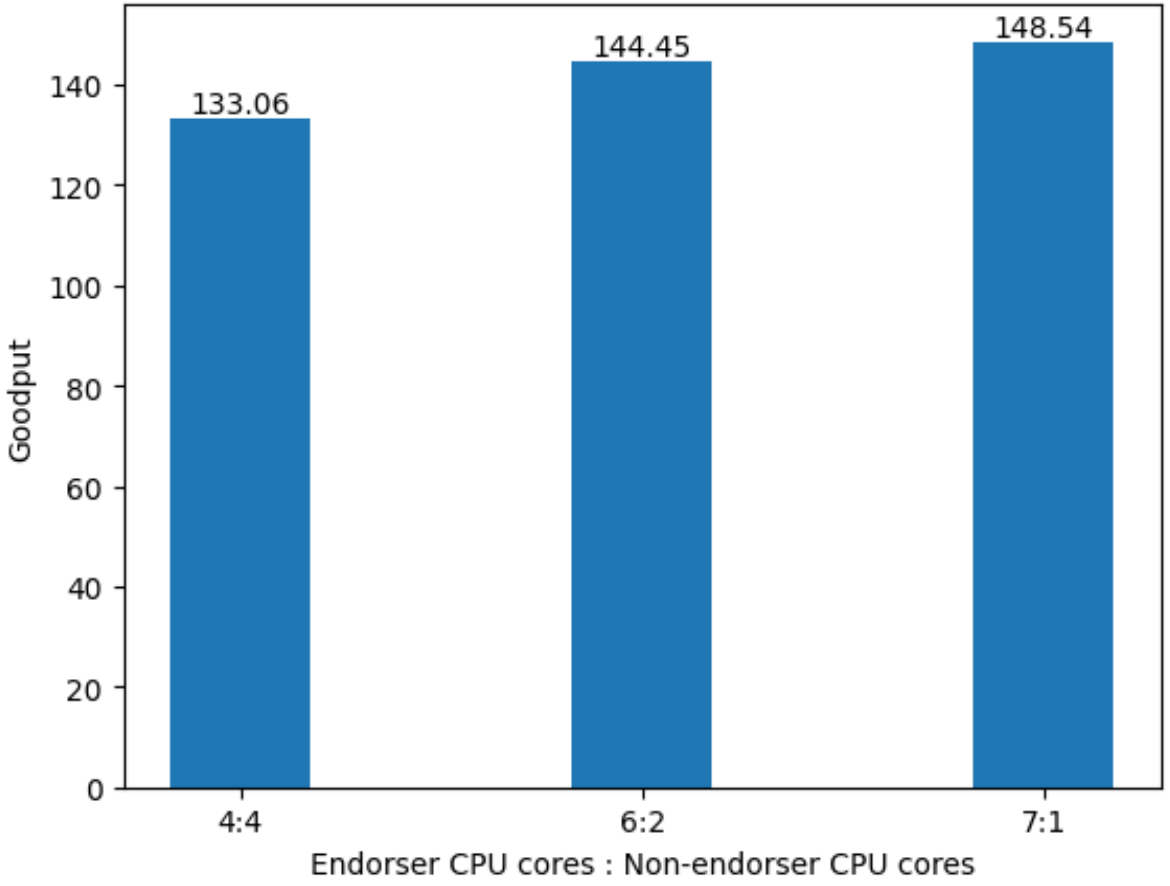
- Timeline for the future works
  - Some other configuration dependency works
  - Cite papers about hadoop, OpenStack, Tabassum's work (Check slack)
  - Complete implementation of the full FabricController framework
  - Simpler fault injection and corresponding
- What do expect to finish

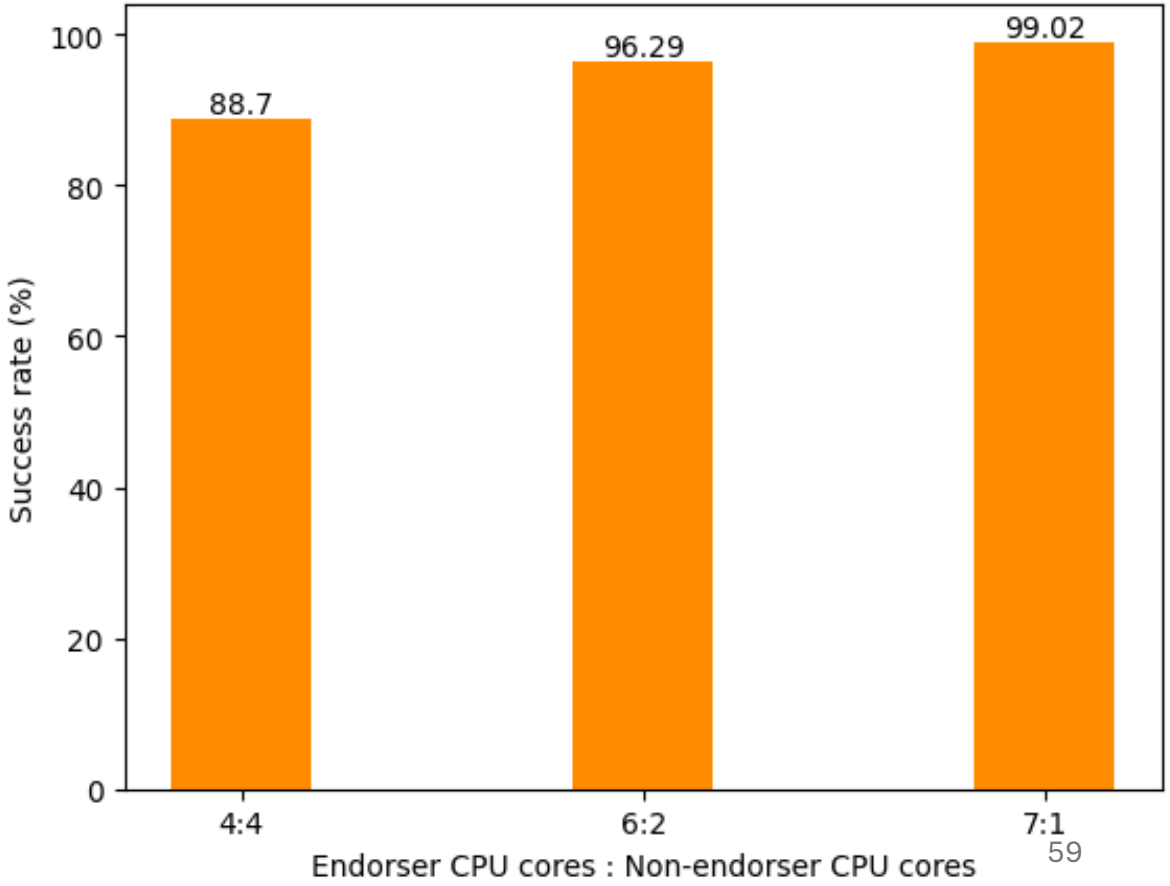
Methodology

- What is implemented in ~~Fabric~~BlockController:
  - Easy test network provisioning using ansible
  - Simplified test network recreation
  - Simplified resource managing for containers
  - Log recording for benchmarking results
  - Injected statedb fault
- Describe each component one by one
  - Checkout runzhou's IO slides
  - Merge fault injector with resource manager

Preliminary experiments
Hardware resource allocation

- Applying the resource manager module…
- Show the main message/finding here, summarize them here

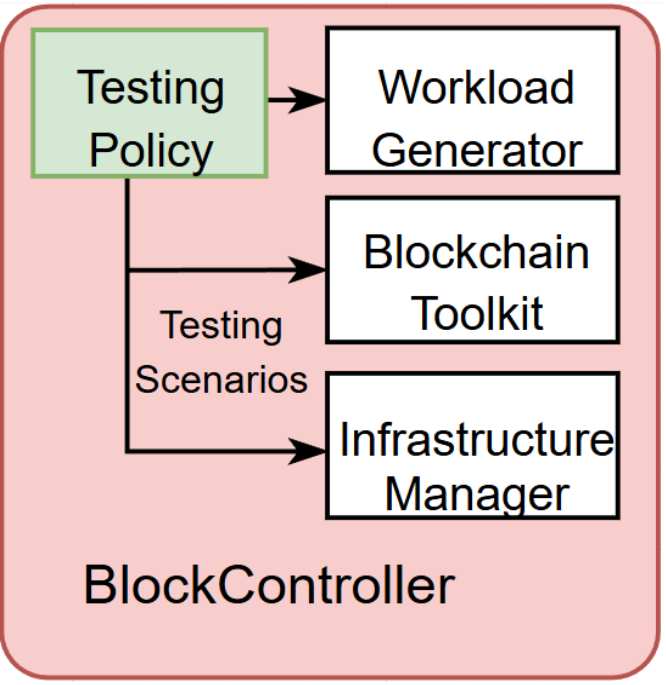




What is blockchain

- An immutable transaction ledger, maintained within a distributed network of nodes
- How does blockchain works?
  - Nodes validate transactions by a consensus protocol
  - Transactions grouped into blocks connected as a chain by hash
  - Each node (peer) maintains a copy of the ledger and the blocks
- Two kinds of blockchain: Permissionless and Permissioned
  - Permissionless: Everyone can join without permission
  - Permissioned: Every participant must be verified to join

Testing Plan decision made here



- Providing policies for blockchain/infrastructure configuration
- Different configuration strategies based on domain knowledge:
  - ~~Highest possible throughput~~
  - Balanced/Unbalanced blockchain
  - Balanced resource allocation by reassigning hardware
- Generate testing scenarios based on the strategies
- Apply configuration combination based on testing scenarios

- Simulate different resource, control resource/workload config to simulate different types of scenarios/different infra characteristics