Performance Benchmarking & Optimizing Hyperledger Fabric Blockchain Platform

Parth Thakkar, **Senthil Nathan N**, Balaji Vishwanathan

IBM Research, India

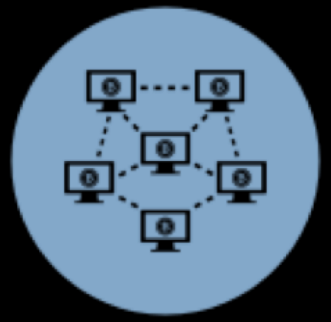
Outline

- Background
  - Blockchain & Hyperledger Fabric
- Motivation for performance benchmarking
- Experimentation methodology
  - Performance **metrics**
  - **Parameters** which Impact the Peer Components
  - **Setup** and **workload**
- Results: Impact of various configurable parameters on the performance
  - **Guidelines** on parameter configuration & **Bottlenecks**
  - **Research opportunities**
- Optimizations (improved the overall throughput by **16x** -- from **140 tps to 2250 tps**)
  1. Cache to minimize crypto operations
  2. Parallel validation
  3. Bulk read/write

Background

Blockchain & Hyperledger Fabric

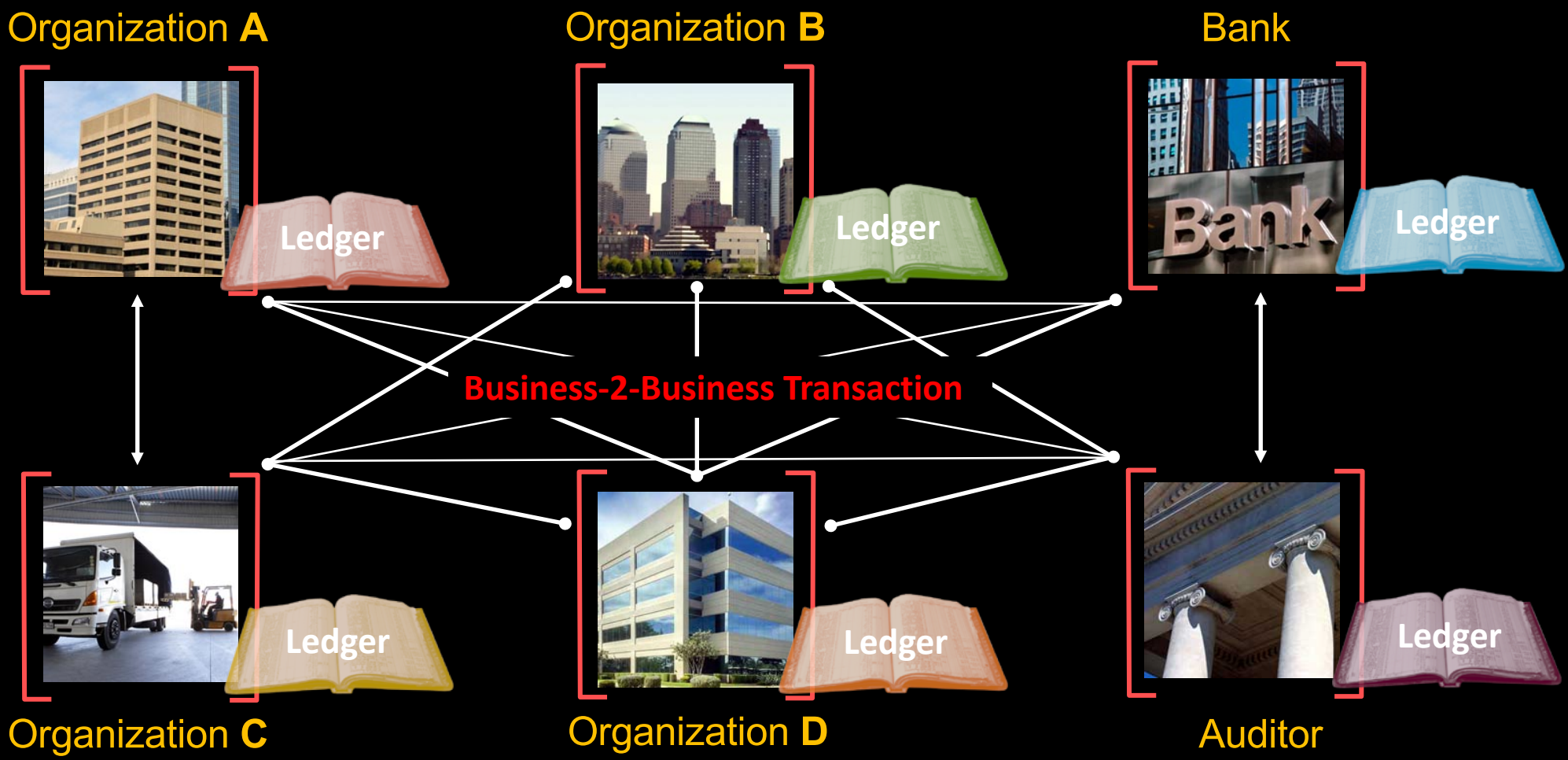
Blockchain Defined



Blockchain is a design pattern made famous by its use in Bitcoin. But its uses go *far beyond*.

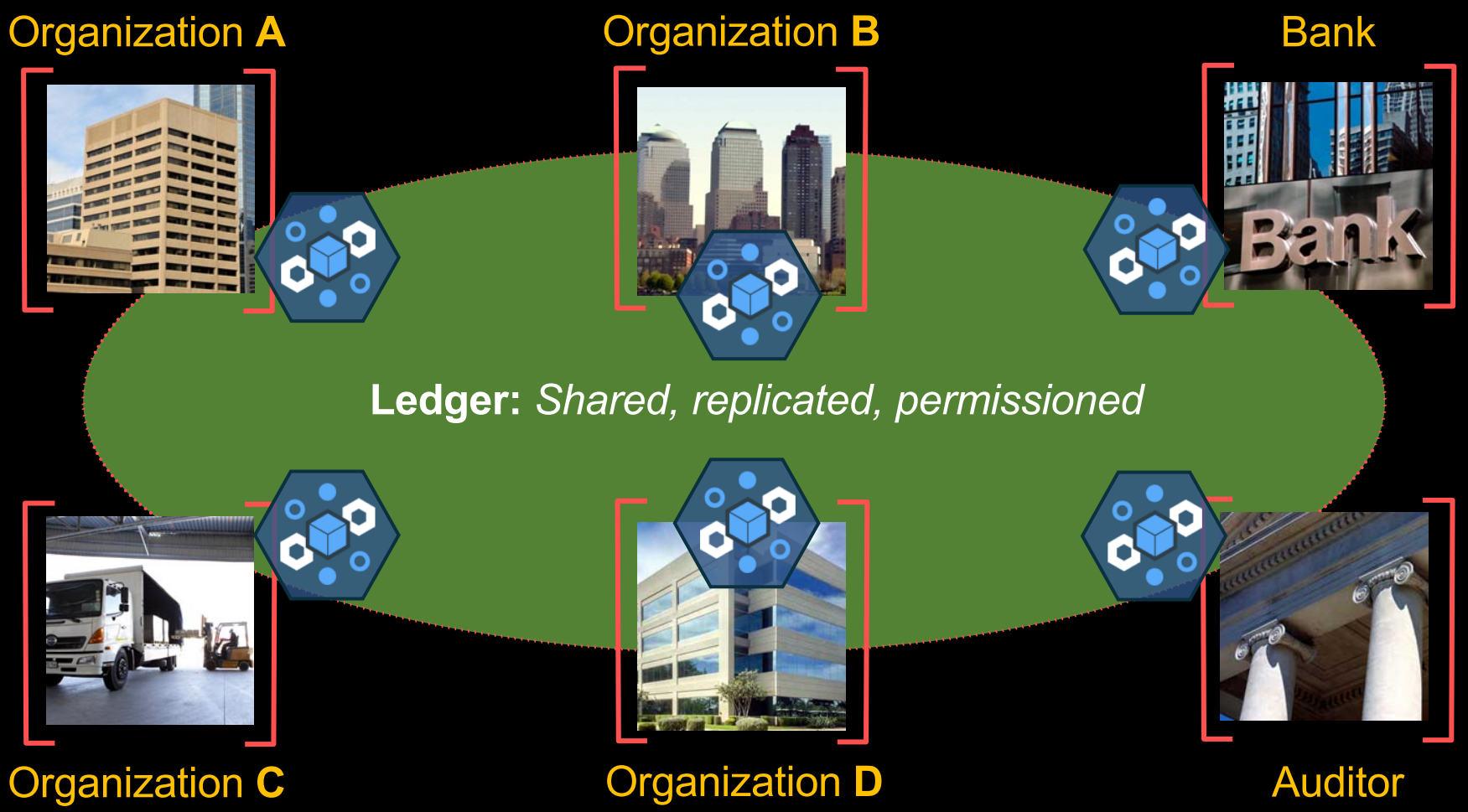
Blockchain can reimagine the world's most fundamental business interactions and open the door to invent new styles of digital interactions.

A Business Network Without Blockchain



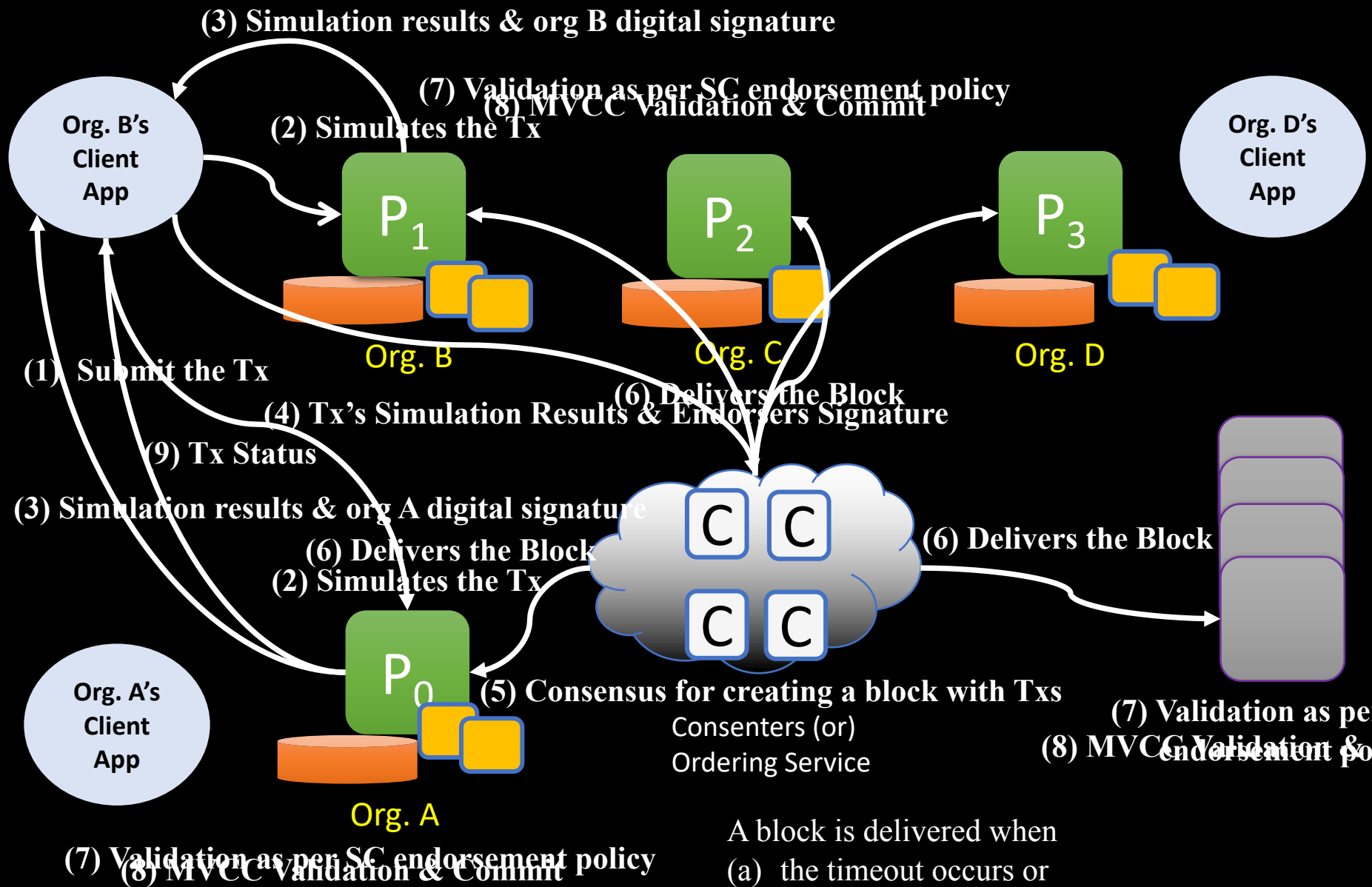
Without Blockchain: Inefficient, Expensive, Vulnerable, Conflicts

A Business Network With Blockchain



With Blockchain: Consensus, Provenance, Immutability

Hyperledger Fabric: Permissioned Blockchain Platform



State DB

Smart Contract

Endorsing & Committing Peer

Committing Peer

Consenters

A block is delivered when
(a) the timeout occurs or
(b) the block size limit is reached

Steps for Executing a Tx

(1) Submits the Tx
(2) Simulates the Tx
(3) Simulation results & digital signature
(4) Submit results & endorsers signatures to consenter
(5) Consensus
(6) Delivers the block
(7) Policy validation
(8) MVCC & commit
(9) Send Tx's status

Our Focus: Three Sub-Components in a Peer

1. Endorser
   - Invoke the chaincode and **simulate the transaction** to collect read/write set
   - Sign the transaction response using **ESCC** (endorser system chaincode code)

2. Validator: On receiving a block
   - Validator validates all transactions in a block by invoking **VSCC** (validator system chaincode)
   - Any **two organization** endorsement signature
     - Or (And(A, B), And(A, C), And(A, D), And(B, C), And(B, D), And(C, D))

3. Committer & Ledger
   - Committer **calls the ledger** to commit the received block
   - Performs **MVCC validation** on read/write set of each transaction
   - Commit the block to **block storage** and valid transactions to **state/history DB**

Motivation for Performance Benchmarking

Motivation for Performance Benchmarking

1. To find out **bottlenecks** and gaps for improvement

2. To propose and implement **optimizations** to improve the performance
   - blockchain solutions' throughput requirement is between 100 tps and 20,000 tps (for RTGS)

3. To provide **guidelines** on **parameters configurations**
   - 100s of questions on parameters configuration on Hyperledger Fabric mailing-list

4. To identify **research opportunities**
   - A new research topic and a lot of potential for enhancement
   - Increase research activities in Blockchain domain

Experimental Methodology

Performance Metrics, Parameters, Setup and Workload

Performance Metrics

- Transaction **Throughput**
- Transaction **Latency**

Transaction **Latency**

1. **Endorsement** Latency
   Client – Endorser
2. **Broadcast** Latency
   Client – Orderer
3. **Ordering** Latency
   Orderer – Peer
4. **Commit** Latency
   Peer – Client

Block-Level Latency

1. **VSCC** Validation Latency
2. **MVCC** Validation Latency
3. **Ledger** Update Latency

Parameters which Impact Peer Components

Parameters	Endorsers	Validator	Committer & Ledger
(1) Block Size		✓	✓
(2) Endorsement Policy		✓	
(3) Number of Channels	✓	✓	✓
(4) Resource Allocation	✓	✓	✓
(5) State DB Choice (CouchDB vs GolevelDB)	✓	✓	✓
(6) Transaction Size (number of keys read/write, size of key/value)	✓		✓

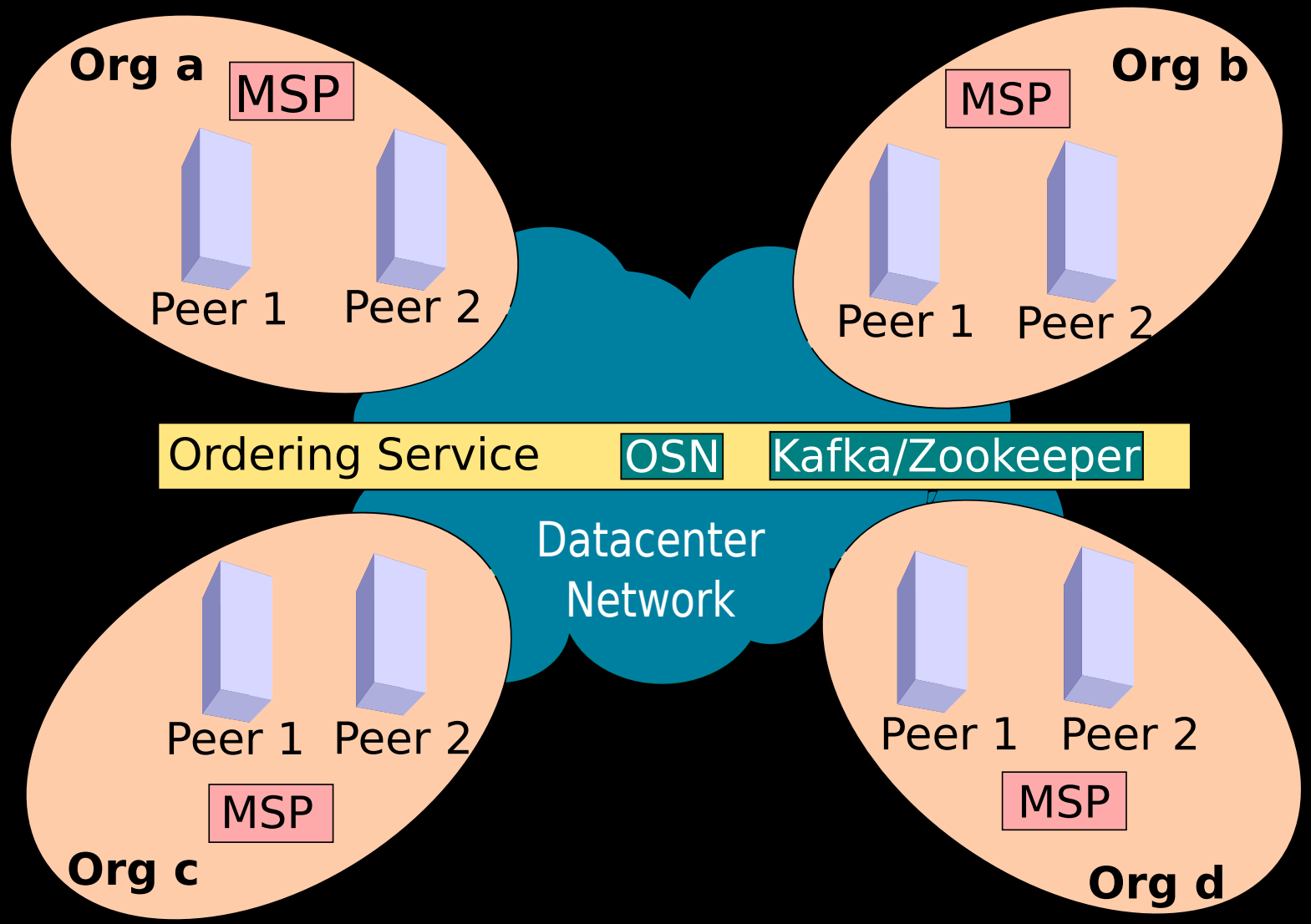
Experimental Methodology

- Perform experiments by **varying values set to configurable parameters** to understand

1. Impact of various **load generation rate** and **block size** on the performance.
2. Impact of **endorsement policy** on the performance.
3. Impact of **number of channels & resource allocation** on the performance.
4. Impact of **transaction size** and **state DB choice** on the performance.

Setup & Workload

Setup



Workload

Built our own workload by surveying around 12 internal customer solutions

Write-only transactions

- 1 KV write
- 3 KV writes
- 5 KV writes

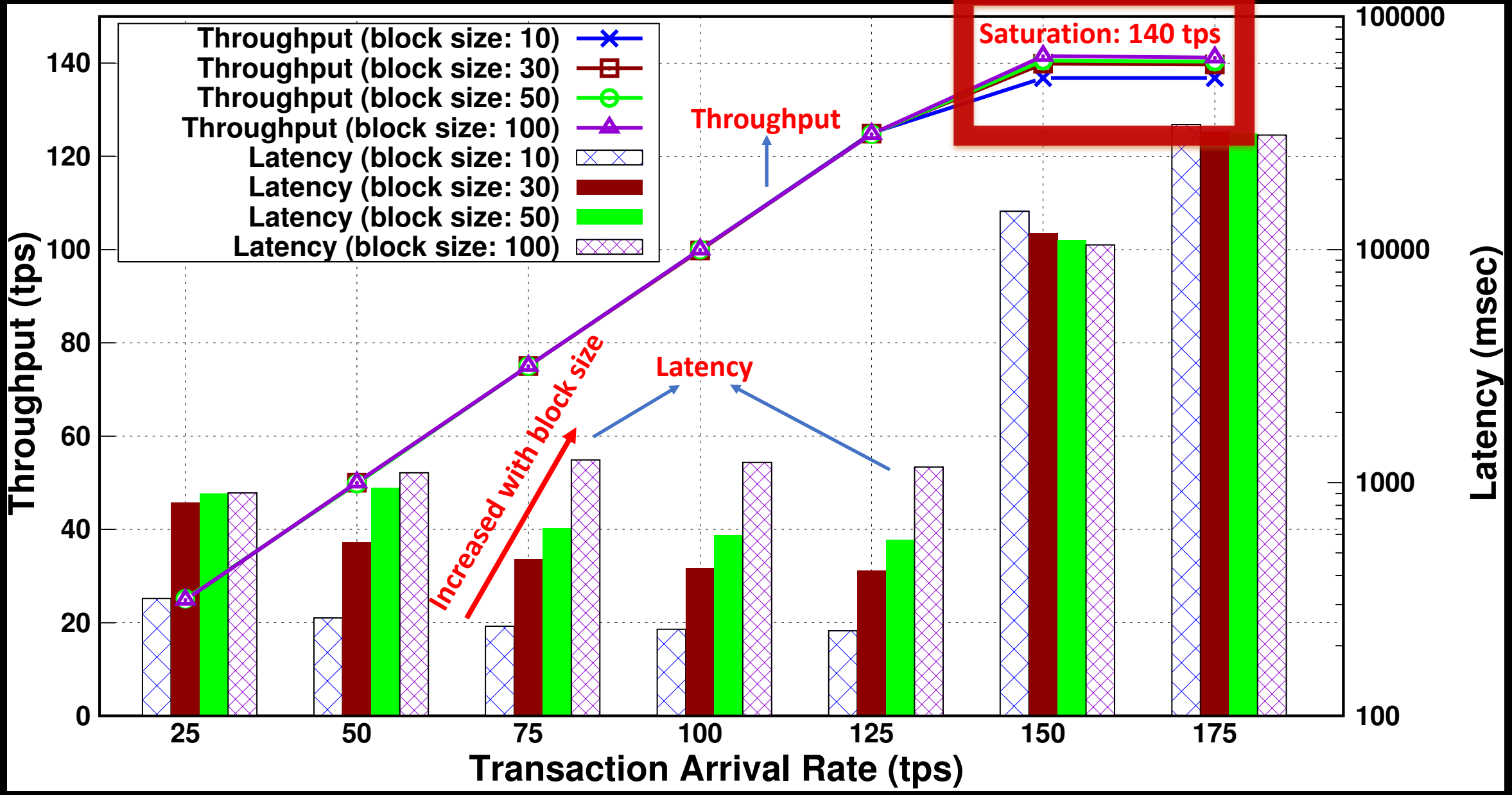
Read-Write transactions

- 1 KV read write
- 3 KV reads writes
- 5 KV reads writes

Results: Impact of various configurable parameters on the performance

Guidelines, Bottlenecks, & Research Opportunities

(1) Impact of Transaction Arrival Rate & Block Size



(1) Impact of Transaction Arrival Rate & Block Size

- Guidelines on configuring the block size:
  - transaction arrival rate < saturation point → **lower block size**
  - transaction arrival rate >= saturation point → **higher block size**

- Bottleneck:
  - CPU resources are under-utilized (~7% utilization) - **Serial validation of transactions by VSCC**
    - Parallel validation has some challenges but could improve the performance

- **Research opportunity:**
  - On higher transaction arrival rate, the **latency increased by >10x**
  - **Admission control** for transactions is necessary to get lower latency
    - very challenging in an **untrusted and distributed system**

(2) Impact of Endorsement Policy

And/Or

1 Endorser : Or (A, B, C, D)
2 Endorsers: Or (And(A, B), And(A, C), And(A, D), And(B, C), And(B, D), And(C, D))
3 Endorsers: Or (And(A, B, C), And(A, B, D), And(B, C, D), And(A, C, D))
4 Endorsers: And (A, B, C, D)

NOutOf

1 Endorser : 1-out-of (A, B, C, D)
2 Endorsers: 2-out-of (A, B, C, D)
3 Endorsers: 3-out-of (A, B, C, D)
4 Endorsers: 4-out-of (A, B, C, D)

- Guideline on configuring the endorsement policy
  - To achieve higher performance, define endorsement policy with lesser #sub-policies and #endorsers
  - NOutOf policy is better performant than And/Or but less powerful

(2) Impact of Endorsement Policy

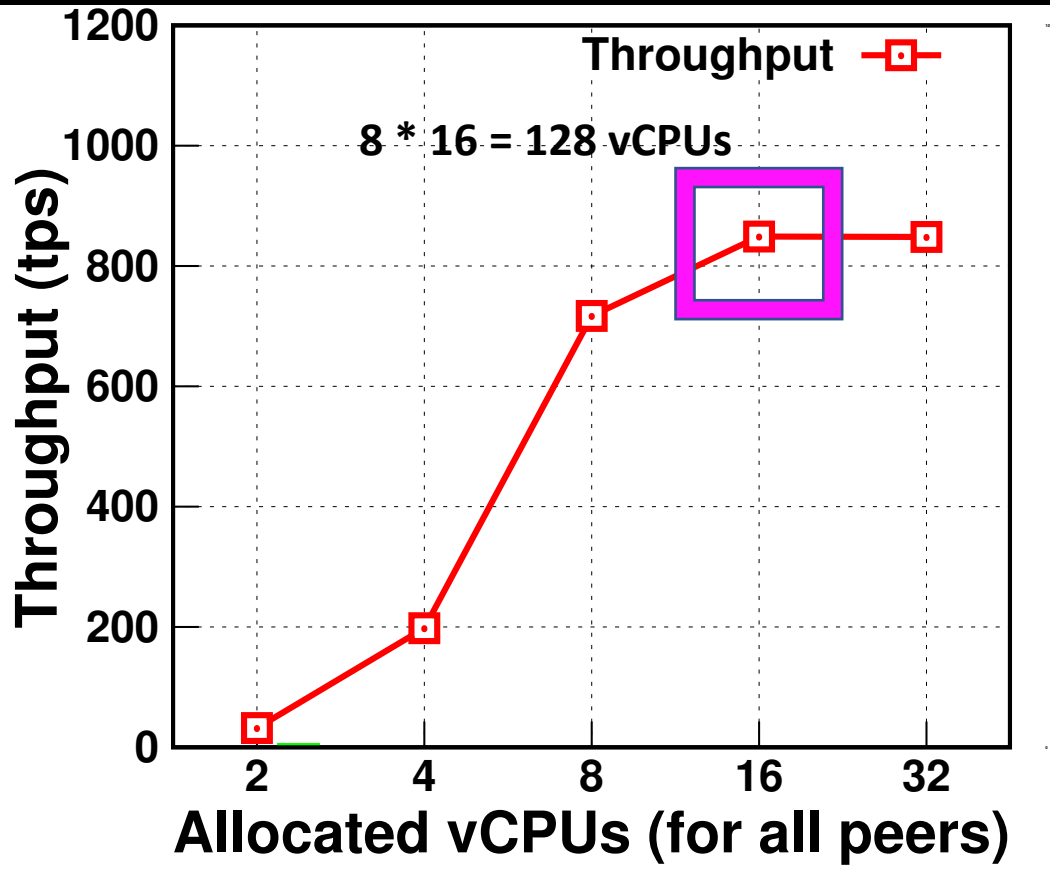
- Bottleneck:
  - Validation of endorsers signature against the policy
    - Repetitive crypto operation -- usage of cache could improve the performance

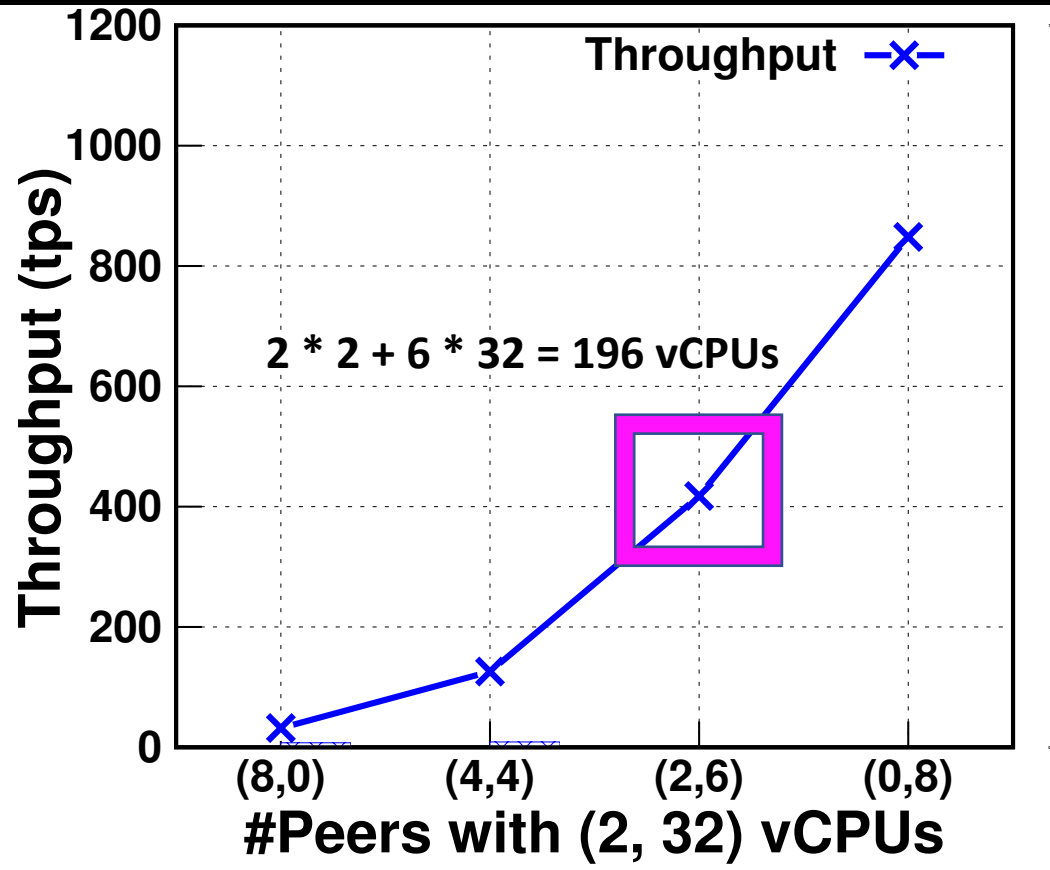
- **Research opportunity:**
  - Matching digital signature set against the endorsement policy is a NP complete problem
  - In a 5 minutes run, 96K out of 220K crypto operations performed were useless – due to greedy way of matching the signature set against the policy.
  - An algorithm that can reduce/avoid such unnecessary crypto operations is needed.

(3) Impact of Channels & Resource Allocation

With increase in the number of channels → Throughput increases due to additional parallelism

Impact of Resource Allocation: Homogeneous & Heterogeneous Peers





(a) Homogeneous Peers, 16 Channels (b) Heterogeneous Peers, 16 Channels

(3) Impact of Channels & Resource Allocation

- Guideline on configuring resources for channels
  - **Dedicated vCPUs per channel** → for the maximum performance & avoid interference
  - **Homogeneous resource allocation** is important – means the **proportion** as per demand and not mean the same amount of resources. For e.g.,
    - If Peer 1 receives 1000 tps, it might require 2 vCPUs
    - If Peer 2 receives 500 tps, it might require only 1 vCPUs

- **Research opportunity**
  - Each channel would receive different load at different period
    - Need for **efficient resource sharing** among channels
  - Dynamic resource provisioning to **ensure homogeneous resource allocation** as per the demand

(4) Impact of State DB choice & Transaction Complexities

- Conducted experiments with different transactions complexities for both GolevelDB & CouchDB
  - Refer to the paper for results

- **Research opportunity:**
  - The usage of DB such as GoLevelDB and CouchDB, without snapshot isolation level, results in the whole database lock -- not good to achieve higher throughput
  - DB does not provide read & write set per transaction -- need to enhance DB
  - Need to make DB more aware of blockchain operations and aid blockchain transactions to improve performance

Fault Tolerance

- Peer failure does not affect the performance during non-overloaded scenario
  - App can collect endorsement from other available peers – application dependent

- Node rejoining after a failure need to sync up with other peers
  - Fetch missing blocks, validate and commit old transactions

- **During overloaded scenario, rejoining peer could not sync up**
  - Rate(new blocks added) = Rate(old blocks fetched by the rejoining peer) = peek rate

- **Research opportunity:**
  - protocol for rejoining peer's sync up so that with any block commit rate, peer should be able to sync
    - Adding a new peer is even worse -- need to fetch blocks from the beginning of the time, validate & commit

Proposed Optimizations & Performance Improvement

1. Introduce **crypto cache** to avoid repetitive crypto operations
   - **3X** improvement -- 140 tps to 520 tps

2. **Parallelly validate** multiple transactions in a block – in VSCC validation phase
   - **7X** improvement -- 140 tps to 980 tps

3. **Bulk read/write** during MVCC Validation and Commit
   - **2.5X** improvement -- 50 tps to 115 tps for CouchD

Performance Improvement from Hyperledger Fabric v1.0 to Fabric v1.1

Combining all three optimization resulted in **16X improvement -- 140 tps to 2250 tps**

Conclusion

- Conducted a comprehensive performance benchmarking to
  1. Provide guidelines on configuring parameters
  2. Improve the performance by 16X
  3. Identity research opportunity

- Future work -- to reach 20,000 tps to support RTGS systems
  - Sharding
  - Distributed validation

Thank You!

Questions?