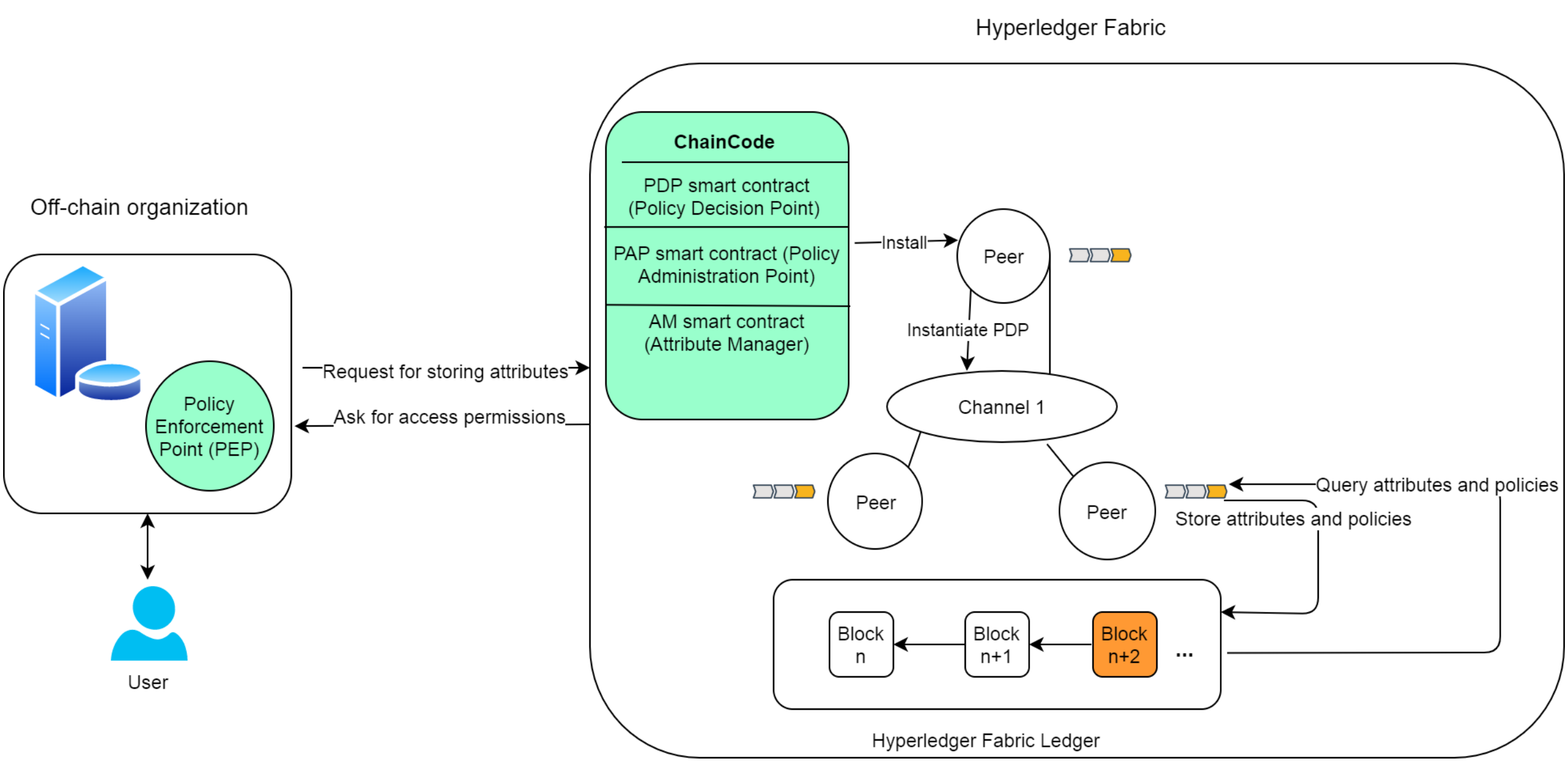
Hyperledger Fabric Based Access Control 2019

Hyperledger Fabric Based Access Control

- **Introduction**
  - **Name**: Sara Rouhani
  - **Location**: Saskatoon, Canada
  - **University**: University of Saskatchewan
  - **Mentor(s):** Prof Rui Cruz, Rafael Belchior
  - **Hyperledger project**: **Hyperledger Fabric Based Access Control**

Hyperledger Fabric Based Access Control

- **Project Description**:
- Attribute based access control
- Hyperledger Fabric
- ArchiMate (Architecture Modeling)
- NodeJS API
- Hyperledger Caliper
- Matplot and Python
- HTML / CSS/ Express/ JavaScript


Hyperledger Fabric
Off-chain organization
ChainCode
PDP smart contract (Policy Decision Point)
PAP smart contract (Policy Administration Point)
AM smart contract (Attribute Manager)
Install
Peer
Instantiate PDP
Channel 1
Peer
Peer
Request for storing attributes
Ask for access permissions
Policy Enforcement Point (PEP)
Query attributes and policies
Store attributes and policies
Block n
Block n+1
Block n+2
...
User
Hyperledger Fabric Ledger

Hyperledger Fabric Based Access Control

- **Project Objectives**:
  - Obj 1: Designing the system architecture and application components
  - Obj 2: Configuring Hyperledger Fabric
  - Obj 3: Implementing the ChainCode
  - Obj 4: Testing and analyzing system
  - Obj 5: Writing the paper
  - Obj 6: Throughout the documentation

Hyperledger Fabric Based Access Control

- **Project Deliverables:**
  - Deliverable 1: System architecture (document)
  - Deliverable 2: ChainCode and application (code)
  - Deliverable 3: Performance data based on Hyperledger Caliper (Data)
  - Deliverable 4: Analyzing data and creating graphs
  - Deliverable 4: Academic paper
  - Deliverable 5: Documentation

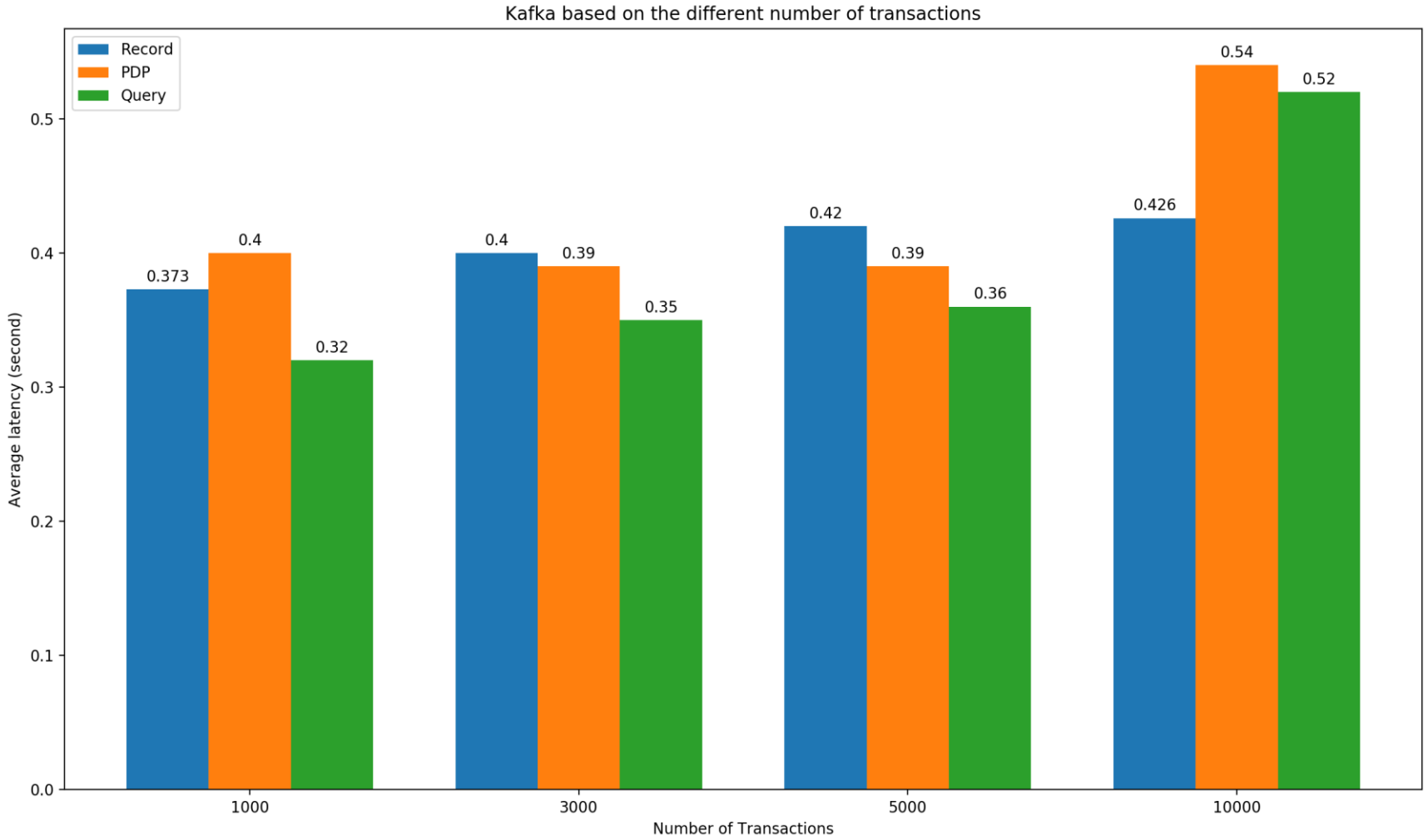
Hyperledger Fabric Based Access Control

› **Project Execution & Accomplishments**:

- ✓ ABAC application including ChainCode implementation and a simple web interface
- ✓ Collecting performance analysis data based on Hyperledger caliper and different configuration
- ✓ Data analysis and graphs design using Matplot library
- ✓ Academic paper
- ✓ github Wiki page

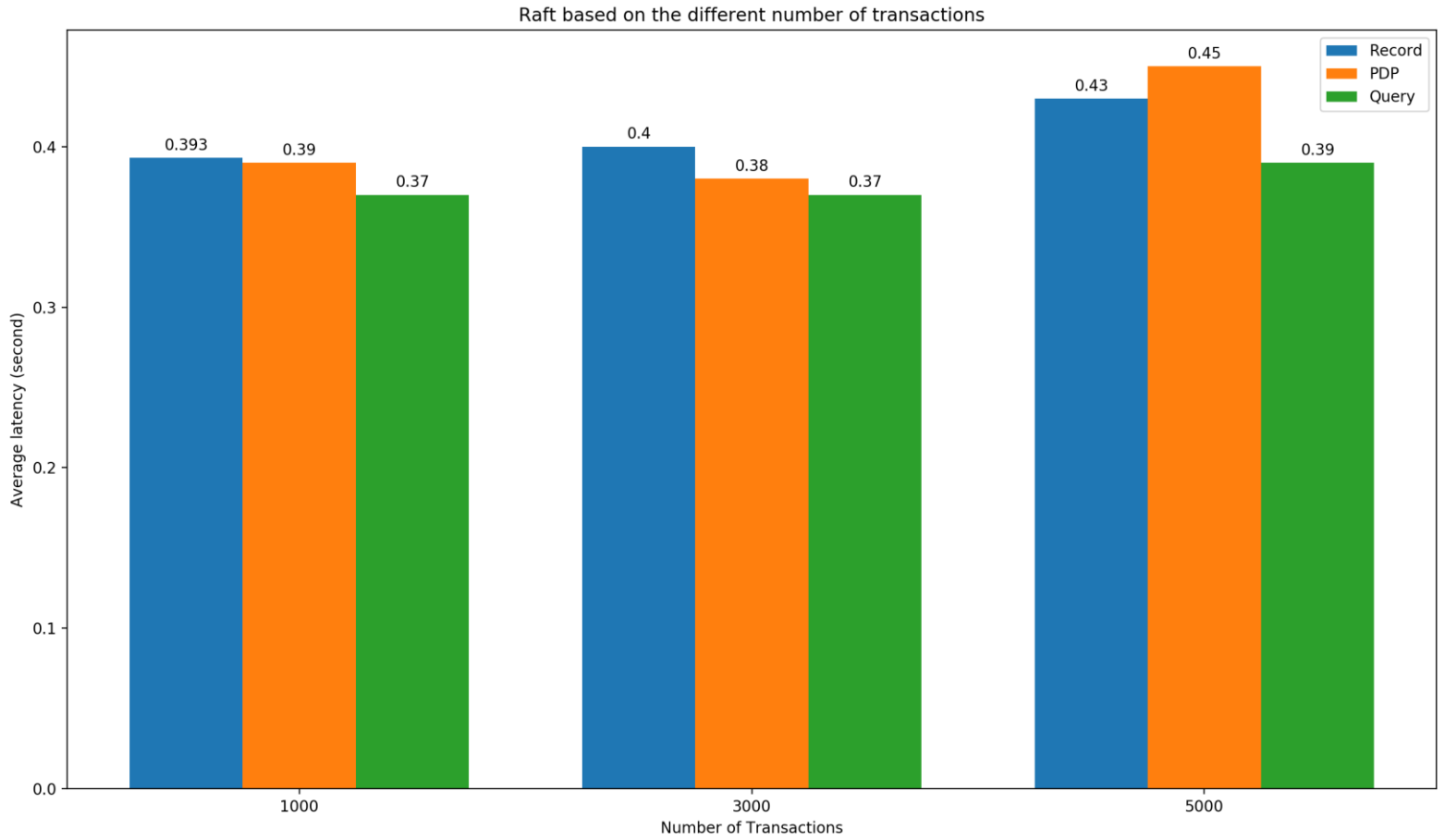
Performance Graphs

- Kafka based on the number of transactions



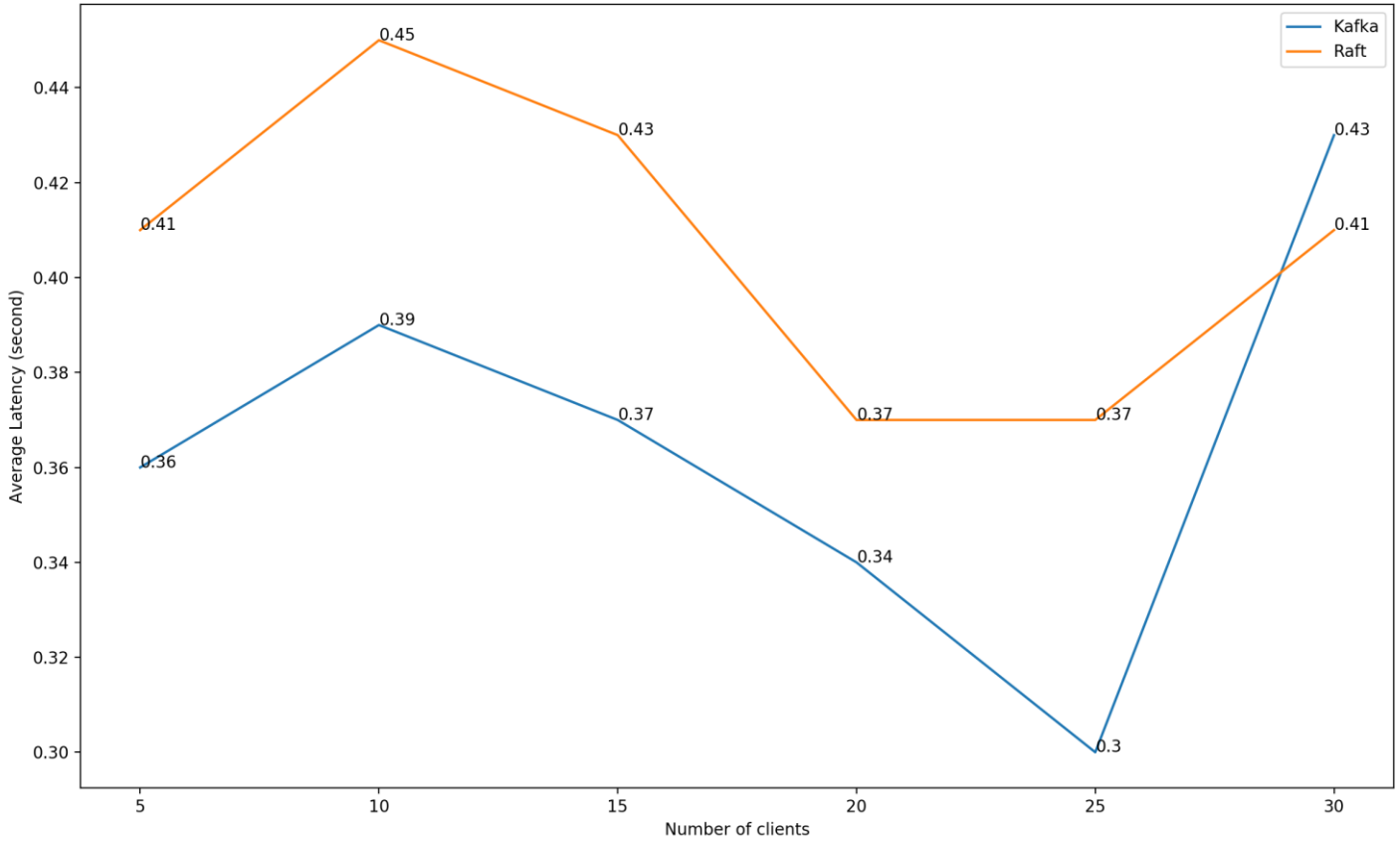
Performance Graphs

- Raft based on the number of transactions



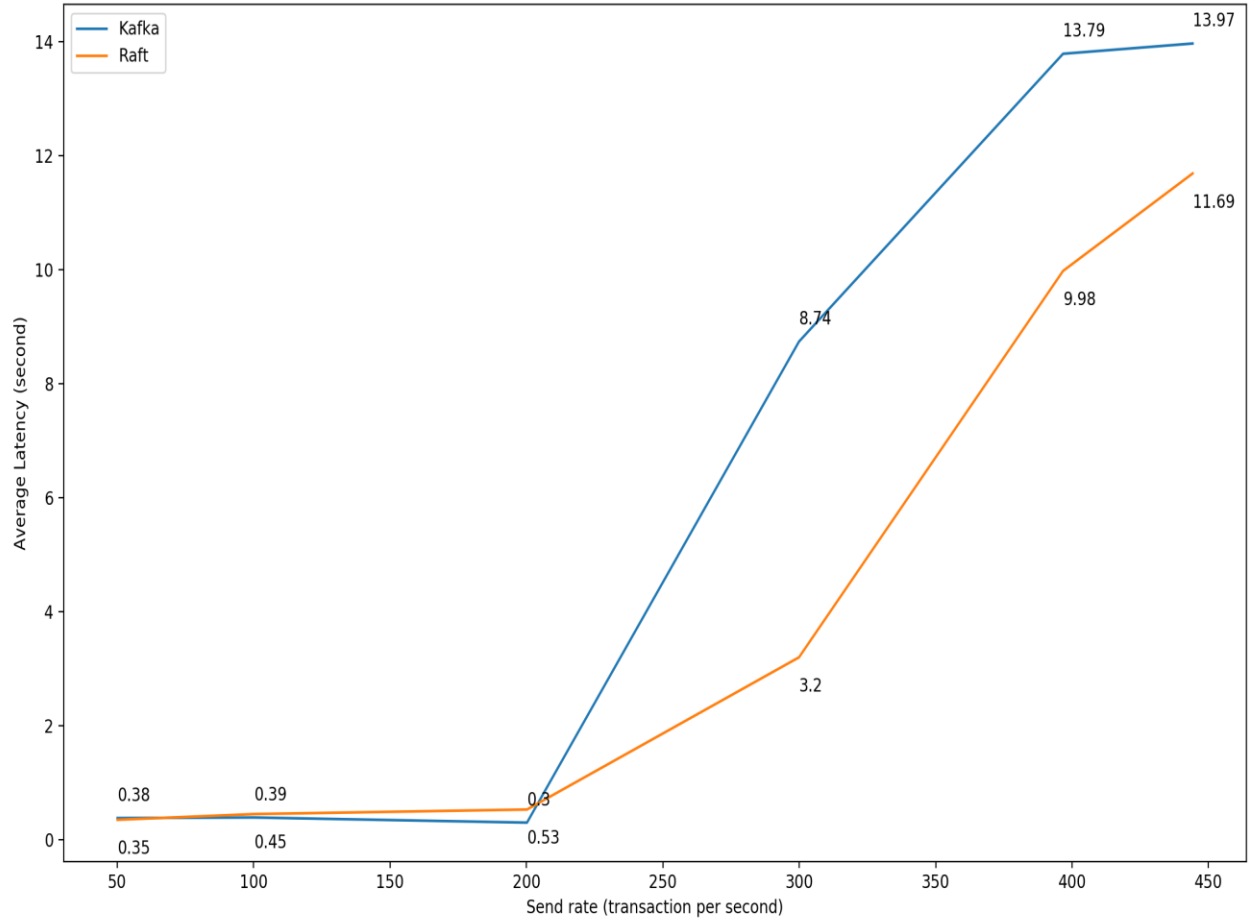
Performance Graphs

- Kafka and Raft based on clients and average latency



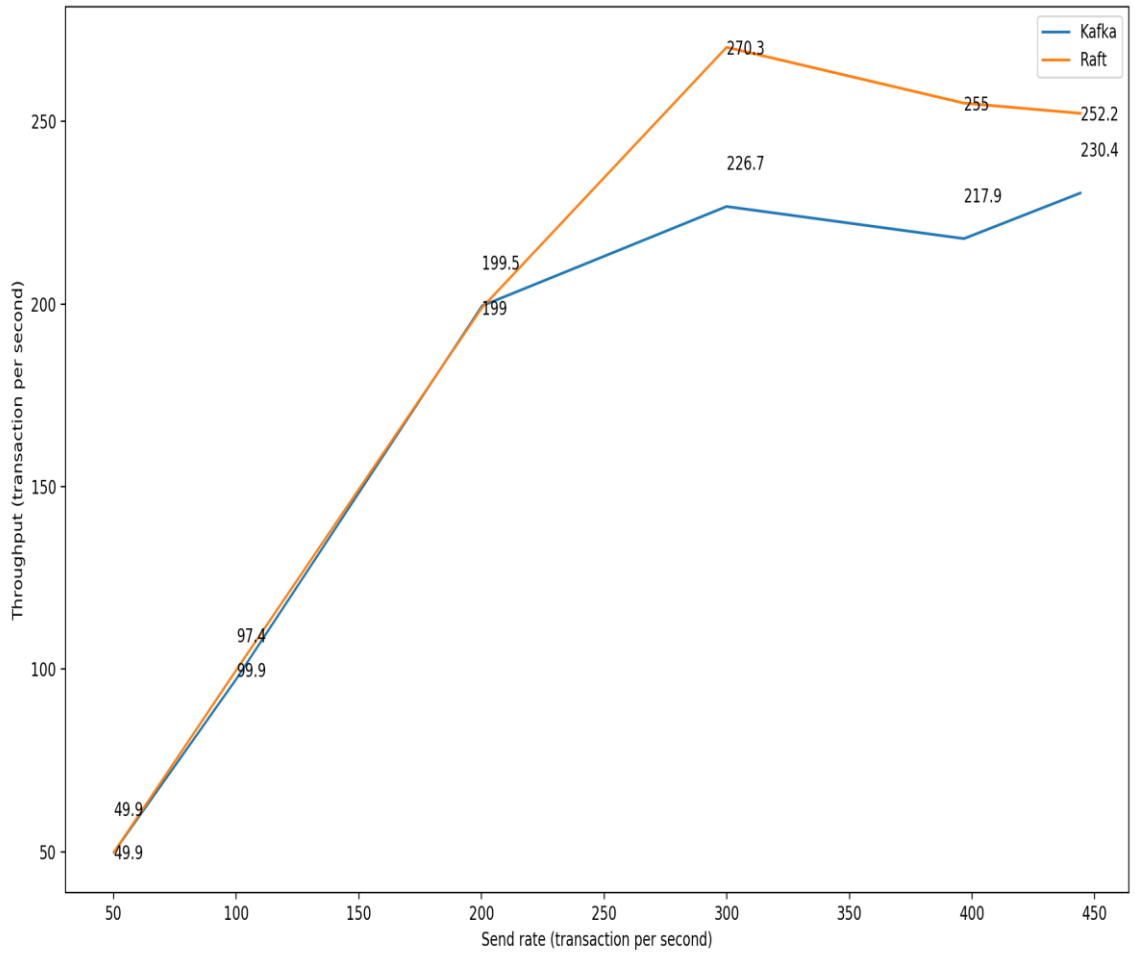
Performance Graphs

- Kafka and raft based on send rate and average latency



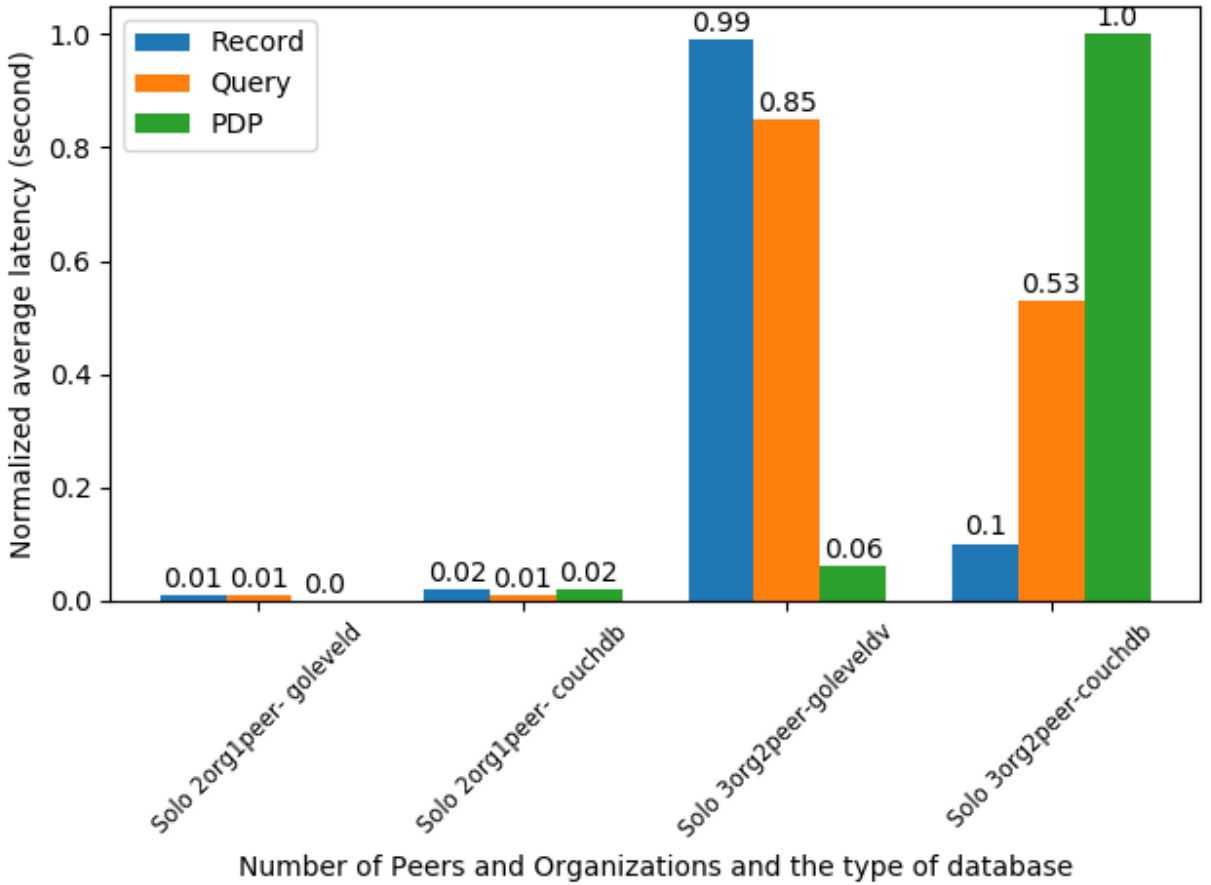
Performance Graphs

- Kafka and Raft based on send rate and throughput



Performance Graphs

The effect of increasing the number of peers and organizations and comparing two databases (CouchDB and goLevelDB)



Resource consumtion

Orderer	Name	Memory(max)	Memory(avg)	CPU(max)	CPU(avg)	Traffic In	Traffic Out	Disc Read	Disc Write
Raft	dev-peer0.org1	74.4MB	71.MB	32.25%	28.21%	13.8MB	5.3MB	0B	0B
	dev-peer0.org2	73.3MB	69.7MB	33.23%	28.28%	13.8MB	5.3MB	0B	0B
	peer0.org1	379.3MB	369.4MB	67.21%	56.32%	31.1MB	23.7MB	0B	21.8MB
	peer0.org2	284.0MB	274,1MB	70.78%	55.66%	31.1MB	23.6MB	4.0KM	21.8MB
	orderer1	554.1MB	535.4MB	26.11%	15.41%	22.4MB	59.1MB	0B	37.2MB
	orderer2	525.3MB	506.5MB	18.78%	11.88%	27.6MB	28.7MB	0B	37.0MB
	orderer0	513.3MB	494.7MB	19.40%	11.63%	27.5MB	10.6MB	0B	37.2MB
Kafka	dev-peer0.org1	73.5MB	72.8MB	17.87%	15.26%	7.5MB	2.5MB	0B	0B
	dev-peer0.org2	64.5MB	62.8MB	18.73%	15.69%	7.5MB	2.5MB	0B	0B
	peer0.org1	295.9MB	286.2MB	52.08%	49.15%	27.1MB	17.0MB	368.0KB	21.2MB
	peer0.org2	294.1MB	282.7MB	51.54%	48.38%	27.1MB	17.1MB	152.0KB	21.2MB
	orderer0	121.1MB	113.1MB	25.80%	23.30%	29.6MB	11.1MB	4.0KB	18.4MB
	orderer1	124.1MB	115.2MB	21.87%	20.25%	29.7MB	46.5MB	276.0KB	18.4MB

Hyperledger Fabric Based Access Control

- **Recommendations for future work:**

- **Present the project as a module for Hyperledger Fabric**
- **Integration with Hyperledger Indy**