Caliper is a performance benchmark framework for blockchain and one of the Hyperledger projects hosted by the Linux Foundation

- Integrate with multiple existing DLTs (Distributed Ledger Technology)
- Measure the performance of specific blockchain systems with predefined test cases
- Reports containing standard performance indicators defined by Hyperledger Performance and Scale WG
- Provide abstract NBIs (Northbound Interface) to help extend test cases

https://github.com/hyperledger/caliper
Target Users and Typical Scenarios

Decision makers
- Test Case
- Test Case
- Test Case

System operators
- Test Case

Developers
- Test Case

- Blockchain No1
- Blockchain No2
- Configuration 1
- Configuration 2
- Old Version
- New Version

Caliper

Performance Report
Performance Report
Performance Report
Performance Report
Performance Report
Performance Report
Node.js based, 3 layers from top to bottom

- **Benchmark Layer**
  - Predefined benchmark test cases
  - Pluggable & configurable benchmark engine

- **Interface & Core Layer**
  - Blockchain NBIs – install, invoke, query……
  - Resource Monitor – memory, cpu, network io ……
  - Performance Analyzer – latency, throughput ……
  - Report Generator – HTML format test report

- **Adaptation Layer**
  - Translate NBIs into DLT protocols
How it works

Preparation
- Initialize blockchain
- Deploy smart contracts
- Start resource monitor

Tests Execution
- Create context
- Generate & submit tx
- Release context
- Return statistics

Performance Analysis

Tests Report

Caliper

Config files

Blockchain

Master
- Execute the test flow according to the configuration
  - Preparation: prepare the test context, e.g. installing smart contracts
  - Test Execution: assign tasks to clients to run the test
  - Performance Analysis: gather test results & generate report

Clients
- Run test case according to the specific workload
  - Transaction count based test or duration based test
  - Pluggable rate controller: Fixed submitting rate, Dynamic submitting rate based on specific schema, ...

Test Case
- Scripts which define interactions with the system under test
  - Use Caliper’s NBIs to define common script for multiple blockchain systems
Example: test configuration file

config.yaml

```yaml
specify type and number of clients used for the test

specify test rounds
- txNumber: defines an array of sub-rounds with number based test runs
- rateControl: defines how to control the txns submitting rate
- arguments: user defined arguments which are passed directly to the specified user test module
- callback: location of the user test module

resource monitor
- docker: local/remote containers which will be watched
```
**Example: network configuration file**

```yaml
name: Fabric
version: "1.0.0"
mutual-ifs: false
caliper:
  blockchain: fabric-cpp
command:
  start: docker-compose f network/fabric-v1.4.2org [peer1] docker-compose.yml up -d sleep 3s
  end: docker-compose f network/fabric-v1.4.2org [peer2] docker-compose.yml down
info:
  Version: 1.4.0
  Size: 2 Orgs with 1 Peer
  Orderer: Solo
  Distribution: Single Host
  StateDB: CouchDB
clients:
  - client0.org1.example.com:
    client:
      organization: Org1
      credentialStore:
        path: /tmp/hc-kvs/arg1
      cryptoStore:
        path: /tmp/hc-crypto/arg1
      clientPrivateKey:
        path: network/fabric-v1.4/config/crypto-config/seer/Organizations/arg1.example.com/users/User1@org1.example.com/msp/keystore/key.pem
      clientSignedCert:
        path: network/fabric-v1.4/config/crypto-config/seer/Organizations/arg1.example.com/users/User1@org1.example.com/msp/signcerts/User1@org1.example.com-cert.pem
  - client0.org2.example.com:
    client:
      organization: Org2
```

**Specify the name of the configuration file**

**Specify the YAML schema version**

**Indicate whether to use client-side TLS in addition to server-side TLS**

**Caliper information**: adaptor, user-defined commands which are called before/after test

**Network Information**: specify custom key-value pairs that will be included as-is in the generated report

**Client Information**: Specify organizations and their attributes
Example: network configuration file

channels:
  mychannel:
    configBlock: network/fabric-v1.4/config/mychannel.conf
  created: false
  peers:
    peer0.org1.example.com:
      eventSource: true
    peer0.org2.example.com:
      eventSource: true
    channels:
      - id: simple
        version: v0
        language: golang
        path: contract/face/simple/
  organizations:
    Org1:
     msp:
        Org1MSP:
          peer:
            peer0.org1.example.com:
          - certificateAuthorities:
            - ca.org1.example.com:
              adminPrivatekeyPath: network/fabric-v1.4/config/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp/private/Admin@org1.example.com MSP\_ca1.pem
              signerCertPath: network/fabric-v1.4/config/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp/signcerts/Admin@org1.example.com-1.example.com\_c1.pem
              Org2:
                peers:
                  peer0.org1.example.com:
                    url: grpc://localhost:7050
                    grpcOptions:
                      tlsCaName: overrides: peer0.org1.example.com
                  peer0.org2.example.com:
                    url: grpc://localhost:7051
                    grpcOptions:
                      tlsCaName: overrides: peer0.org2.example.com
                    grpc.baupalive_time_ms: 600000
                  peer0.org2.example.com:
                    certificateAuthorities:
                      ca.org1.example.com:
                        url: http://localhost:7054
                        httpOptions:
                          verify: true
                        registrar: null
                        - admin
                  certificateAuthorities:
                    ca.org2.example.com:

Certificates section contains one or more unique channel names as keys, and each key has a corresponding object (sub-keys) that describes the properties of the channel.

Organizations section contains one or more, arbitrary but unique organization names as keys, and each key has a corresponding object (sub-keys) that describes the properties of the organization.

Orders section contains one or more, arbitrary but unique orderer names as keys, and each key has a corresponding object (sub-keys) that describes the properties of the orderer.

Peers section contains one or more, arbitrary but unique peer names as keys, and each key has a corresponding object that describes the properties of that peer.

Certificate Authorities section contains one or more CA names as keys, and each key has a corresponding object (sub-keys) that describes the properties of that CA.
Demo(1)

- Pre-requisites
  - NodeJS 8 (LTS), 9, or 10 (LTS)
  - node-gyp
  - Docker(18.03.0-ce or above)
  - Docker-compose(1.21.0 or above)
- Run with benchmark sample
  - Build
  - Run with Caliper CLI
Run with your own blockchain system
  - Skip the start and end commands
  - publish the Caliper modules to the server
  - Modify network configuration file to connect with the remote booted network

Execute the run command
  - caliper benchmark run -w <path to workspace> -c <benchmark config> -n <blockchain config>
How to write your own adapter

- Use Blockchain NBI to write your own blockchain class
- Add your own blockchain type into the blockchain’s constructor function (caliper-core/lib/blockchain.js)

Nice to have

- Add predefined Network files into the dir "network/"
- Add your own network configuration file into the corresponding network folder
- Define your command which will be executed before and after the test
- Define your own smart contracts
- Define the installation script
What's new

- Adaptors:
  - Iroha
  - Ethereum
  - New Fabric-ccp adapter

- Code structure adjustment for NPM publishing
  - Decouple the samples, master flow and adaptors
  - New factory methods for adaptors
  - CLI package
Roadmap

**Jun 2019**
- Support of Fabric v1.1-1.4 (two adapters were provided: fabric and fabric-ccp) & Sawtooth v1.0 & Burrow & Iroha & Ethereum
- Added Fabric kafka example and raft example
- Provided file mode to support long time test
- Refactor code for npm package publishing
- Created CLI and sample package
- Improved documentation
- Enhanced abilities (configurable logging mechanism, fabric network creating detection, automatically generate crypto config, a license check process, Caliper running environment check)

**Sep 2019**
- Npm package
- API interface
- Docker image
- Upload/edit test configuration
- Real time charts
- Corda adaptor

**Dec 2019**
- Sufficient test coverage
- Quorum
- Sufficient use cases
- Metrics extension
  - Optional Metrics Framework
  - Costs for a certain TPS
Welcome to join Caliper team

- Questions and suggestions
  - Rocket chat (https://chat.hyperledger.org/channel/caliper)
  - Issues (https://github.com/hyperledger/caliper/issues)
- Regular meeting: UTC 3pm Wednesday (https://wiki.hyperledger.org/display/caliper)
Thank You!