Hyperledger Caliper
A performance benchmark framework for blockchain
Contents

- What’s Caliper
- Architecture & Design
- Roadmap
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
What’s Caliper

Target users and typical scenarios (1)

For decision makers who choose blockchain system for their business, Caliper can help:

- Test performance with specific test cases to find out which one best meet their needs
- Guarantee fairness for various systems
- Learn resource (CPU, Memory,...) requirements and estimate costs to set up the system
- ......
What’s Caliper

Target users and typical scenarios (2)

For system operators, Caliper can help:
- Evaluate performance of multiple blockchain configuration schemes and choose the best one
- Learn how network condition would affect the performance
- Find out the hardware requirements for specific SLA
- ……
For developers, Caliper can be used as an internal tool to:

- Qualify the performance improvement of new version
- Assess the impact of new features on performance
- Compare with other blockchain systems
- ......
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

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

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
  - …..

Scripts which define interactions with the system under test
- Use Caliper’s NBIs to define common scripts for multiple blockchain systems
How it works

- 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

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

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
  - ……

Scripts which define interactions with the system under test

- Use Caliper’s NBIs to define common script for multiple blockchain systems
Test configuration file is used to specify test flow and workloads, as well as other global configuration items.

```json
{
    "blockchain": {
        "type": "fabric",
        "config": ".fabric.json"
    },
    "command": {
        "start": "docker-compose -f ../../network/fabric/simplenetwork/docker-compose.yaml up -d",
        "end": "docker-compose -f ../../network/fabric/simplenetwork/docker-compose.yaml down; docker rm $(docker ps -aq)"
    },
    "test": {
        "clients": {
            "type": "local",
            "number": 5
        },
        "rounds": [
            {
                "label": "open",
                "txnNumber": [5000, 10000],
                "rateControl": [{"type": "fixed-rate", "opts": {"tps": 200}}, {"type": "fixed-rate", "opts": {"tps": 300}}],
                "arguments": {"money": 10000},
                "callback": "benchmark/simple/open.js"
            },
            {
                "label": "query",
                "txnNumber": [5000],
                "rateControl": [{"type": "fixed-rate", "opts": {"tps": 300}}],
                "arguments": {},
                "callback": "benchmark/simple/query.js"
            }
        ],
        "monitor": {
            "type": "docker",
            "docker": {
                "name": "peer0.org1.example.com", "http://192.168.1.100:2375/orderer.example.com"
            },
            "interval": 1
        }
    }
}
```

- **Specify the location of blockchain network configuration file**
- **User defined commands which are called before/after test**
- **Specify type and number of clients used for the test**
- **Specify test rounds**
  - **txnNumber**: defines an array of sub-rounds with number based test runs
  - **rateControl**: defines how to control the txns submitting
  - **arguments**: user defined arguments which are passed directly to the specified test script
  - **callback**: location of the test script
- **Resource monitor**
  - **docker**: local/remote containers which will be watched
Network configuration file is used to specify access points, as well as other necessary informations such as cryptographic materials required to interact with the SUT.

Informations of orderer and peers which can be used to submit transactions to Fabric.

Informations of fabric channels.

Informations of fabric chaincodes.
Example: test case

A typical directory of test case: https://github.com/hyperledger/caliper/tree/master/benchmark/simple

- config-fabric.json
- config-iroha.json
- config-sawtooth.json
- config-zookeeper.json
- config.json
- fabric-remote.json
- fabric.json
- iroha.json
- main.js
- open.js
- query.js

Test & network configuration files for various configuration options as well as specific systems under test

Startup parameters `-c` `-n` are used to specify config files for the test

Bootstrap script, the test can be started by running `node main.js`

Default script is implemented using benchmark engineer, the script can be used for various test cases. However, developers can also implement their own bootstrap script.

Test scripts which defines the actual blockchain operations using caliper NBIs
Test results are outputted to the console in real time, and a HTML format report will be generated after the test.
Performance and Scale Working Group is a cross project forum of Hyperledger for architects and technologists to discuss, research, and identify key metrics that relate to the performance and scalability of a blockchain and blockchain related technologies.

https://wiki.hyperledger.org/groups/pswg/performance-and-scale-wg
Roadmap

- Fabric v1.1 & Sawtooth v1.0 & Iroha
- Performance metrics
  - Success Rate
  - Throughput
  - Latency
- Resource Monitor
  - Docker Container
  - Local Process
- Simple sample test cases

- Other Hyperledger blockchain (and Non-Hyperledger System?)
- Keep up with PSWG metrics
- GUI & Dashboard support
- Add more sophisticated test cases
- Stable version supports long-term & large-scale testing
- Integrate with blockchain & network operation tools

Jun 2018

Dec 2018
Contact

• Join the discussion of performance metrics and benchmark requirements
  https://wiki.hyperledger.org/groups/pswg/performance-and-scale-wg

• Submit issues or PRs to contribute directly
  https://github.com/Huawei-OSG/caliper

• Any questions / suggestions about Caliper
  https://chat.hyperledger.org/channel/caliper
  zhouhaojun@huawei.com
  huruifeng@huawei.com