Hyperledger Caliper

Benchmark framework for blockchain
Content

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

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

- Integrated 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
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
- …..
For system operators, Caliper can help:
- Evaluate performance with 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
- ......
Architecture

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 as configured
- 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 workload
- Transaction count or duration based test
- Pluggable rate controller
  - Fixed submitting rate
  - Dynamic submitting rate based on specific schema
  - ......

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

```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",
      "txNumber": [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",
      "txNumber": [5000],
      "rateControl": [{ "type": "fixed-rate", "opts": { "tps": 300 }}],
      "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
}
```

- **Config path of SUT configuration file**
- **User defined commands which are called before/after test**
- **Config 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
  - arguments: user defined arguments which are passed directly to the specified test script
  - callback: location of the test case script
- **Resource monitor**
  - docker: local/remote containers which will be watched
Example: SUT configuration

```
{
  "fabric": {
    "cryptodir": "network/fabric/simplenetwork/crypto-config",
    "network": {
      "orderer": {
        "url": "grpcs://localhost:7050",
        "mspid": "OrdererMSP",
        "user": {
          "key": "network/fabric/……/keystore/be595…57cd_sk",
          "cert": "network/fabric/……/Admin@example.com-cert.pem"
        },
        "server-hostname": "orderer.example.com",
        "tls_cacerts": "network/fabric/……/tls/ca.crt"
      },
      "org1": {
        "name": "peerOrg1",
        "mspid": "Org1MSP",
        "user": {...},
        "peer1": {...},
        "peer2": {...}
      },
      "org2": {...}
    },
    "channel": [{
      "name": "mychannel",
      "config": "network/fabric/simplenetwork/mychannel.tx",
      "organizations": ["org1", "org2"],
      "deployed": false
    }],
    "chaincodes": [{"id": "simple", "path": "contract/fabric/simple", "language": "golang", "version": "v0", "channel": "mychannel"}]
  }
}
```

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

Informations of fabric channels

Informations of fabric chaincodes
A typical test case:

https://github.com/hyperledger/caliper/tree/master/benchmark/simple

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 engine, 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.
Example: Test report

Test results are outputted to the console in real time, and a HTML format report will be generated after the test.
Roadmap

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

- Non-Hyperledger Systems
- Keep up with PSWG metrics
- Enhance GUI & Dashboard support
- Add more sophisticated test cases
- Stable version supports long-term & large-scale testing
- Network emulators
- Integrate with blockchain & network operation tools

Jun 2018 | Dec 2018
Resources

- Performance and Scale Working Group (PSWG)

- Contributors
  - From Huawei, IBM, Intel, Soramitsu, Budapest University of Technology and Economics and more

- Users
  - PersistentSystems
  - Trusted Blockchain Alliance
Thank you!

• **Visit Huawei Booth 501** for Caliper demo and more information

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

• Try out Caliper and submit issues or PRs
  https://github.com/hyperledger/caliper

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