2019-06-12 Minutes

Hyperledger Caliper Project

Community Regular Meeting
June 12, 2019 (9AM – 10AM UTC)
via Zoom

Participants
Nick Lincoln, Qinghui Hou, Attila Klenik

Proposed agenda items
1. Discussion about the extent of SUT monitoring that Caliper should be responsible for
2. Monitor(s) => Caliper-core => Reporter(s) design discussion
3. Final additions prior to npm release
4. Updates and additional ideas

Agenda discussion summary

Monitoring discussion
1. Monitoring the SUT poses numerous challenges
   a. A heterogeneous backend infrastructure monitoring requires a wide range of support from Caliper
   b. A vast number of metrics are provided by monitoring agents
      i. Reporting all wouldn't be informative
      ii. Selecting the key metrics automatically is a complex problem
2. Performance reporting
   a. A report doesn't necessarily need to include SUT-side performance metrics
   b. SUT-specific metrics would make the results noncomparable
   c. The client-side, high-level metrics are common indicators across multiple platforms, as defined by the PSWG.
      i. These are directly observed by Caliper
3. Design for performance
   a. To help to design the capacity of a SUT, some common resource metrics could be gathered
      i. Basic CPU, memory, storage, network metrics
      ii. Maintaining multiple "collectors" (for sources like Docker, node_exporter, laaS monitoring solutions) for a small set of metrics could be feasible

Summary:
- Point 2. is a must-have even for reporting. Fortunately, everything is contained inside Caliper, there is no dependency on third-party tools.
- Point 3. is partially covered by Docker (and Node.JS process) monitoring. But we still need to collect the most popular monitoring agents and select the common key-metrics to support.

Monitor-reporter design discussion

A clean data model is necessary for flexible reporting and analysis. The well-known design pattern from monitoring tools is starting to emerge:

- A monitor subsystem, that collects observations from multiple sources:
  - Load generator clients (e.g., TX event observations)
  - Remote monitoring agents (e.g., resource metric time series)
- A reporter/exporter subsystem, that disseminates the gathered data to multiple targets:
  - Aggregator for an HTML reporter
  - Detailed data exporter for CSV format
  - Detailed data exporter for XY database
  - Detailed data exporter for remote service (like the Caliper UI)

Summary: we need to come up with a flexible data model, containing both metadata (measurement campaign ID, test round, data source, etc) and TX- or resource-related observations.

The targeted feature set for the NPM release

The following tasks must be completed before the NPM release:
1. Refactoring the Fabric CCP adapter. Corresponding issue: https://github.com/hyperledger/caliper/issues/467
2. Add the Fabric SDK's fabric-network (evaluate/submit) approach to the Fabric CCP adapter (as a complementary run mode for 1.4.X networks)
3. Rename the Fabric adapters, which will essentially deprecate the older adapter.

Every other feature will be added incrementally after NPM publishing.
Additional ideas

The gist of some other ideas that occurred during the call:

1. Adding discovery support for Fabric 1.4.X networks
   a. The SDK makes this relatively painless
   b. It opens up the way to deal with evolving systems (adding channels, chaincodes or users even during the test run, not just during the init phase)

2. Support arbitrary user behavior profiles in the test round configuration
   a. Specify a client profile by setting the workload aspects: rate and user test module
   b. When putting together a test round configuration, set the participating clients explicitly
      i. E.g., use 5 clients with profile A, 10 with profile B, etc.

3. Ask the community for feedback:
   a. Feature requests
   b. Usability requests