Thesis: Performance Testing and Analysis of Qpid-Dispatch
Date: 2018-02-20
Time: 14:00-15:00
Attendance: jstejska, opiske, zkraus, fnigro, rkubis

Meeting objects:

1. How are notes from Maestro-Clients evaluated/executed in back-end?
2. Agent component structure
3. Agent component responsibilities
4. Performance testing metrics for thesis
5. Plan for next steps
6. Ideas for testing
7. FAQ

1. How are a notes from Maestro-Clients evaluated/executed in back-end?

Note are serialized in the Maestro-Client and sent to the Maestro-Broker's specific topic (/maestro/daemon/sender for example). Back-end peers listening on this topics and when they receive note they do data deserialization. After it the peer create command for execution based on note (run, stop) or just send another note.

2. Agent component structure

The agent will be new module in Maestro-Java project divided into three parts:

- **Router Management** - Manage the router itself (start, stop…).
- **Basic Metrics Measurement** - Part which should measure basic performance metrics.
- **Router Specific Metrics Measurement** - Part for gathering information about internal metrics of the router.

This module will inherit as much as possible from another Maestro modules (MaestroWorker, MaestroClient, etc.). Basically it will extend current peers functionality, add some new notes for router testing control (Peer is any Maestro component connected to the Maestro-Broker).

3. Agent component responsibility

The agent will have some responsibility during the testing. There should be some basic responsibilities and also some responsibilities for manipulating with topology for some experimental cases (for thesis):

- Start/Stop/Restart router/broker
- Update topology during test (Add new router, add redundant paths, remove paths/routers from current topology)
- Close links between topology devices (messaging router, messaging broker, messaging clients)
- Shut down receiver (monitor behavior of router during this case)
- TODO (feel free to add another responsibilities)

4. Performance testing metrics for thesis

For whole thesis there should be two types of metrics. The first type are **basic** metrics which can be already measured by current version of Inspector, Sender and Receiver. THe second type should be some **behavioral** metrics:

- **Basic metrics** - Rate, Throughput, Latency

- **Behavioral metrics**
  - Router recovery (how long, how it influence the other metrics and the load…)
  - Topology convergence (during route-link disable, or router disable)
  - Influence of disabling broker connected to the router
  - TODO

5. Plan for next steps

The work on the thesis should be divided into few steps:

1. Basic interconnect performance testing
   a. Current version of Maestro can do that
   b. Just specific multiple scenarios via groovy scripts and run the test (it should show router capabilities in thesis)
2. Behavioral performance testing
   a. Implementation of the Agent module
   b. Create multiple scenarios for testing
   c. Run the testing and collect the results
   d. Write conclusion about collected results
   e. Suggest some performance improvements
3. Future work
   a. Implement extensions of Maestro for future use
   b. Prepare Maestro for performance testing which will use metadata from router itself (during the test)
   c. Anything else?

6. Ideas for future testing

There are some ideas about implement testing methods which are based on metadata from router itself. For example tagged packet which will contains timestamps from routers - each router add timestamp when proceed this packet. However, this testing types needs update of Qpid-Dispatch itself so it should be only mentioned and described as an idea for the future development of the Maestro.

7. FAQ

It's necessary to update testing scripts for demonstrate maestro testing with docker containers?

No, the tests using docker images add all address and other data based on docker image name.

Where are created all instances during the testing?

Groovy scripts contains all data for test such as test execution time and so on. Here is also created Maestro instance which create all necessary instances of the Maestro modules.