BTEST
A PERFORMANCE ANALYSIS TOOL FOR SDN CONTROLLERS: ONOS versus OPENDAYLIGHT comparison

Authors: Manuel Cadenas, Luis Sciacca, Rodolfo Alvizu, Guido Maier

ONOS/CORD Workshop September 14th, 2016, Turin (Italy)
AGENDA

- Introduction
- Environment Setup
- Btest Development
- ONOS vs OpenDaylight
- Demo
- Conclusions
Scope of this work

1. Create a performance assessment tool for the main carrier-grade SDN network controllers
2. Compare the performance of available open source carrier-grade controller

Open Source

- Based on Open source tools

Easy to use

- Current performance tools are difficult to use, Btest is an easy to use and understandable tool.

Graphical Results

- Numerical and graphical representation to have a better understanding of the controller performance

Support for Carrier Grade Controllers

- ODL and ONOS
ONOS and ODL are the biggest and widest Open source SDN controllers and both are carrier-grade oriented
Considered SDN Controllers Performance Metrics

**Throughput**
- Number of responses per second that the controller can handle

**Latency**
- The time that takes to send and receive an answer to a packet_in message

**Sojourn Time**
- Time that the packet-in message last inside the controller

**Topology Discovery Time**
- Time that takes to the controller to discover all the network components
The communication between switches and controller is done with Packet_in messages.

The asynchronous messages are the one initialized by the switch to send any update to the controller.
BTEST DEVELOPMENT AND ENVIRONMENT SETUP
Btest Development

TEST COMPILER

Puts together all the pieces, the open source software used was TestON.

LATENCY AND THROUGHPUT CALCULATOR

Calculates the Throughput and the Latency of the SDN controller, the open source software used was Cbench.

Btest

- ODL support
- Btest Integration
- Plotly Integration
- Cbench Integration
- Latency and Throughput modules
- Sojourn time Module

GRAPHIC GENERATOR

Allows to generate a graphical result of the test execution, the open source used was Plotly

SOJOURN TIME

Calculates the Estimated Sojourn time based on a M/M/1 queuing model.

TEST COMPILER

Puts together all the pieces, the open source software used was TestON.

LATENCY AND THROUGHPUT CALCULATOR

Calculates the Throughput and the Latency of the SDN controller, the open source software used was Cbench.

Btest Integration

- ODL support
- Btest Integration

Plotly Integration

- Cbench Integration
- Latency and Throughput modules

Sojourn time Module

Cbench Integration

Calculates the Throughput and the Latency of the SDN controller, the open source software used was Cbench.

Graphical result of the test execution, the open source used was Plotly

Calculates the Estimated Sojourn time based on a M/M/1 queuing model.
In Latency mode, each configured switch sends a single packet_in message to the controller and waits for a flow_mod message, then repeats this process. The total number of responses received during test period is then used to calculate the average processing latency.

Throughput mode:
Each configured switch constantly sends as many packet_in messages as possible, in order to measure the maximum capacity of the controller.
Sojourn Time

Queuing Model M/M/1

\[ \lambda = \sum_{i=1}^{k} \lambda_i \]

\[ \bar{N} = E[N] = \sum_{j=1}^{\infty} j p_j = \frac{\rho}{1 - \rho}. \]

\[ \bar{N}_q = \sum_{j=1}^{\infty} j p_{j+1} = \frac{\rho^2}{1 - \rho}. \]

\[ \bar{W}_q = E[W_q] = \frac{\rho}{\mu(1 - \rho)}. \]

\[ \bar{W} = \bar{W}_q + \frac{1}{\mu} = \frac{1}{\mu - \lambda}. \]

Arrival rate of packet_in messages

Controller Processing rate

Average queue length

Average length of waiting queue

Average waiting time packet_in message

Sojourn time

Btest Architecture

ONOS and OpenDaylight drivers that allow the communication between Btest and the SDN Controllers.

In this section are located all the python files.

Core of the Btest in which everything is processed.

.topo file must be configure with data regarding the topology ip and passwords while .params file with parameters regarding to the test.
Latency and Throughput
Establish a connection with the controller and executes the cbench command with the configured parameters.

Sojourn Time
For the measured, a subtraction of the incoming and exit time of the packet is performed. For the estimated Btest runs in sequence the Latency and Throughput test and calculates the estimated through the formula.

Comparison Test
Latency or Throughput test was run in both controllers returning which controller had a better performance.
Network Configuration

ONOS NETWORK
ONOS Network is composed of ONOS instance, Btest, and Mininet.

ODL NETWORK
The ODL network is composed of ODL, Btest, and Mininet.

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestON</td>
<td>2.0.0</td>
<td>Test Compiler</td>
</tr>
<tr>
<td>Mininet</td>
<td>2.2.1</td>
<td>Network simulator</td>
</tr>
<tr>
<td>Ubuntu Server</td>
<td>16.04</td>
<td>OS</td>
</tr>
<tr>
<td>Cbench</td>
<td>1.0.0</td>
<td>Testing Framework</td>
</tr>
<tr>
<td>OpenDaylight</td>
<td>Helium SR3</td>
<td>Network Controller</td>
</tr>
<tr>
<td>ONOS</td>
<td>Falcon</td>
<td>Network Controller</td>
</tr>
<tr>
<td>Plotly</td>
<td>1.12.0</td>
<td>Python Graphic generator</td>
</tr>
</tbody>
</table>
RESULTS
The Throughput decrease while the number of switches increases.

The controller needs to handle a higher packet-in messages traffic.

ONOS outperforms ODL in this test for all the number of switches. 
Both controllers start getting unstable while the number of switches increases.

ODL is gets more unstable with a higher number of switches compared with ONOS.

**ONOS vs ODL - Round 2: Throughput’s Standard Deviation**

- **Responses per second**
- **Switches**
  - ONOS vs ODL Standard Deviation

![Graph showing the comparison between ONOS and ODL in terms of throughput's standard deviation.](image-url)
The Latency performance for both controller as we can see depicts a constant behavior.

This performance is correct, since we know that in Latency mode the controller do not buffer the packet in messages.

ONOS had a better latency performance compared with ODL controller.
ONOS vs ODL - Round 4: Stress Test

**Throughput Stress Test**

- Throughput decreases while the number of switches increases.

**Latency Stress Test**

- Delay remains stable.
The Sojourn time results are derivate from the Throughput and Latency, so the behavior of these results is proportional to both of them.

For both controllers is possible to see how the Sojourn time increases while the number of switches also increases.

The packet in message should last less in the queue if there is less traffic and the packet out response message is generated in a faster way.
Conclusions

After 5 rounds of Test cases: ONOS provided a better performance than ODL.

In ODL and ONOS:
- Throughput decreases as number of switches increases.
- Standard deviation increases as number of switches increases.

ODL is built on multi-core processing model and needs a large amount of CPU resources.

Btest is in an initial development stage, we expect more features and tests to be included.
THANK YOU

Contacts
luis.sciacca@polimi.it
rodolfoenrique.alvizu@polimi.it
guido.maier@polimi.it
## Related Works

<table>
<thead>
<tr>
<th>Conference/Event</th>
<th>Title</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE, Cyprus Section</td>
<td>SDN Controllers: A Comparative Study</td>
<td>Superficial comparison between the controllers in a theoretical level, performance analysis using open source performance tool (cbench)</td>
</tr>
<tr>
<td>MELECON 2016</td>
<td>Qualitative comparison of open-source SDN controllers</td>
<td>Comparison of different SDN controllers using cbench under poor network resources</td>
</tr>
<tr>
<td>IEEE TELSIKS '15</td>
<td>A comparison between several SDN controllers</td>
<td>Comparison of several software defined network controllers, using Mininet as a measurement tool.</td>
</tr>
<tr>
<td>NOMS-2016</td>
<td>On the performance of SDN controllers: A reality check</td>
<td>Reality check on the current performance achieved by mainstream open source controllers</td>
</tr>
<tr>
<td>UPV-SDN 2015</td>
<td>Development of a performance measurement tool for SDN</td>
<td>Design and development of a measurement tool for testing POX Controller using Mininet and python script.</td>
</tr>
<tr>
<td>NFV-SDN 2015</td>
<td>Performance Comparison Of the state of the art Openflow Controllers</td>
<td>State of the art comparison of several controllers</td>
</tr>
</tbody>
</table>

Luis Sciacca & Manuel Cadenas Master Thesis
Future Work

Add new metrics and tests
Test clustering capabilities. Topology changes, Northbound metrics, and other test cases

Open source Btest
Make the code public

Better compute resources
Run Btest performance assessment using better compute resources.
## SDN Main characteristics

<table>
<thead>
<tr>
<th>Directly Programmable</th>
<th>Agile</th>
<th>Centrally Managed</th>
<th>Open Standard and Vendor neutral</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network control is directly programmable</td>
<td>Abstracting control from forwarding layer</td>
<td>Network intelligence is centralized</td>
<td>When implemented through open standards, SDN simplifies network design and operation.</td>
</tr>
</tbody>
</table>
To install Plotly's python package, it was needed to use the package manager `pip` inside the terminal. Once the packet was installed, it was needed to change the user name and the `api_key`.
OpenDaylight Driver Development

Code
Many python and bash lines code were necessary to establish a connection with the Open Daylight controller.

ODL Variable export
The environment variables for ODL are export, then the ODL directory is located in order to get to work the controller.

Handler
This driver handles all the configurations needed for a particular test.
ENVIROMENT SETTINGS

SSH CONFIGURATION
Connection between different machines was established using the standard port 22 through SSH (Secure Shell) interface for securely getting access as is done in a real network environment.

ODL CONFIGURATION
Much easier than ONOS to configure and install. Only a few steps for configuring the apps that work together with ODL as Java, Maven and Karaf.

ONOS CONFIGURATION
ONOS works properly as multi-instance controller with an odd number of nodes, so we created one node cluster to compare against ODL in same situation.

BTEST CONFIGURATION
Based on the TestON initial configuration files, the user needs to change the 2 main input files: topo, param
Performance Test Implementation

### Latency and Throughput
When a user selects this test a script is executed establishing a connection with the controller using cbench command tool to perform the execution.

### Sojourn Time
The script runs first the throughput test and then latency test, all the data is grab from both test and using the model explained the estimated sojourn time is calculated.

### Comparison Test
Script is executed performing a comparison between both controllers in latency or throughput mode using a sequence run test for each one, giving them a fair environment conditions.

### Topology Discovery
A connection between Mininet and the controller involved is established, through port 6633 or 6653. Then, it configures the features in the controller if necessary and network discovery starts using pingall command from Mininet machine.
ONOS and ODL enable SDN through a combination of components.

- Open Source
- The installing documentation is up-to-date and detailed
- New releases every 3 months
- Support of OpenFlow 1.0, OF 1.3.