ASIC: An Architecture for Scalable Intra-domain Control in OpenFlow

Pingping Lin
Tsinghua University
Outline

- The understanding of OpenFlow and SDN
- Scalability problem in OpenFlow control plane
- Related solutions
- The ASIC architecture
- Experimental results and analysis
Outline

- The understanding of OpenFlow and SDN
- Scalability problem in OpenFlow control plane
- Related solutions
- The ASIC architecture
- Experimental results and analysis
Software Defined Networking (SDN)

- Software defined networking (SDN)
  - separates the control and data plane
  - moves the network intelligence in control plane to a logically centralized controller
  - opens up the control plane and its protocol implementation.

- Architecture in network device is disclosed.
- The idea is well received by academic and industry researchers.
- OpenFlow, the most popular instance, is deployed by many universities and research institutions.
Outline

- The understanding of OpenFlow and SDN
- Scalability problem in OpenFlow control plane
- Related solutions
- The ASIC architecture
- Experimental results and analysis
The Scalability Problem

- OpenFlow is a centralized control model.
- All the routes are determined by the controller.
- The first packet (initialization request) of each data flow is sent to the central controller.
- Controller computes the routing path and install it to the related OpenFlow switches.
The Scalability Problem

- The request processing capability of a single controller is limited:
  - NOX could process about 30K requests/s;
  - Maestro could process about 600K requests/s.

- Large-scale network environments always have vast amounts of data flows:
  - 1) a 1500 server cluster might generate 100K requests per second;
  - 2) a 100 switch data center might generate 10000K requests per second.
The Scalability Problem

- With the increasing scale of deployment, the scalability becomes obvious.
Outline

- The understanding of OpenFlow and SDN
- Scalability problem in OpenFlow control plane
- Related solutions
- The ASIC architecture
- Experimental results and analysis
## Related Solutions

<table>
<thead>
<tr>
<th>Name</th>
<th>Method</th>
<th>Shortcomings</th>
</tr>
</thead>
<tbody>
<tr>
<td>HyperFlow</td>
<td>Multiple controllers; Each handles local OF switches; Sharing the global information by WheelFS.</td>
<td>can only deal with not occur frequently events</td>
</tr>
<tr>
<td>DevoFlow</td>
<td>clustering or fuzzy matching technology; each of the controller processes is a class of data flows;</td>
<td>classification number $\leftarrow$ compromise $\rightarrow$ control burden</td>
</tr>
<tr>
<td>DIFANE</td>
<td>returns some control right to the OpenFlow switch; pre-installing several forwarding rules.</td>
<td>sacrifices real-time visibility of the global data flow status</td>
</tr>
<tr>
<td>ONIX</td>
<td>treats each partition network as a logical node when make global routing decisions; each partition network makes local routing decisions.</td>
<td>sacrifices real-time visibility of the global data flow status</td>
</tr>
</tbody>
</table>
Outline

- The understanding of OpenFlow and SDN
- Scalability problem in OpenFlow control plane
- Related solutions
- The ASIC architecture
- Experimental results and analysis
Load Balancing to Initialization Requests

- Initialization requests are distributed (varieties of algorithms) to different controllers by load balancing equipment.
- All the physical controllers are equivalent.
- Load balancer can be: router, OF switch, professional Web load balancer, professional Web load balancer...
To provide a consistent global view for each controller, this paper suggests adopting a mature data storage system which must include at least two parts:

1) persistent storage (support transaction);
2) caching storage.

Controllers directly deal with the caching storage in both reading and writing.

The persistent storage could be used for data analysis or maintain network status during a reboot process.
Storage Cluster for Data Sharing

- To give a concrete picture, this paper takes the MySQL database for example.

- **Step 1:**
- Writing and reading mechanism to the distributed storage of the shared data:
Storage Cluster for Data Sharing

- **Step 2:**
  - The data cluster in ASIC further adds a memory caching pool to the data sharing mechanism:
  - Memcached pool and the persistent storage should synchronize the data from time to time.
Scalability Analysis of the ASIC

Each of the three levels in ASIC has its own scalable solution and could be extended to meet the requirements of large-scale networks:

- The selection of the load balancing equipment can range from the software balancer to the current Internet backbone router.
- ASIC can also increase the corresponding number of the physical controllers. (Different controllers could run different controller software)
- The number of databases should also increase correspondingly.
Application Deployment in Control Plane

- Developers selectively install their applications according to their preference of developing language or to the functions of different controllers.
Outline

- The understanding of OpenFlow and SDN
- Scalability problem in OpenFlow control plane
- Related solutions
- The ASIC architecture
- Experimental results and analysis
Five requests generating hosts are connected to five different ports of the router.

The router cascades five controllers in the right direction.
Throughput

- Performance metrics:
  - request processing throughput, data flow setup latency.

- Throughput

<table>
<thead>
<tr>
<th>Controller</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>414,598</td>
<td>403,470</td>
<td>425,553</td>
<td>383,376</td>
<td>394,511</td>
</tr>
</tbody>
</table>

Single controller is about 400,000 request/s.

5 controllers (X > 5): 400,000 × 5 = 2,000,000.
Latency Analysis

- The time delayed in ASIC is $O(h) + O(f(n))$

1) $h$ is the hops of balancers in the balancing system.
   - The time for traversing a balancer is equal to traversing an ordinary switch (acceptable).

2) $f(n)$ represents the time consumed to fetch the global network view data from the shared data storage.
   - $n$ means the number of nodes where the data segments located.
   - Storage is cached in the memory
   - Data can be fetched by multi-thread in a parallel
   - Drops to $f(n/\text{num(threads)})$ (negligible).
Conclusion and Future Work

- The computer field adopted a common underlying hardware (x86 instruction set).
- OpenFlow can be considered as “the x86 instruction set of the network”.
- This paper adopts the idea of load balancing, parallel processing, data sharing and clustering to solve the scalability problem in the OpenFlow control plane.
- Future work focuses on deploying the ASIC to the actual OpenFlow network and monitoring its behavior.
Thank you!

Questions & Comments?