1 Summary

In some cases customers have experienced increased network latency and other network performance issues when using Open vSwitch (OvS).

OVS is the default network stack on XenServer.

This article discusses the OvS architecture, potential bottlenecks, and how to identify and address OvS related issues occurring in an environment.

2 Background

Introduced in version 6.0, XenServer uses Open vSwitch (OvS) as the default network stack. This allows the use of OpenFlow controllers to control XenServer hosts, and access to the Software Defined Networking (SDN) market.

However, it required architectural changes to the way in which a XenServer host handles network connections.

As OvS is a software switch, it creates flow entries for every new connection to maintain the current status of network traffic. The creation of these flow entries has a cost in terms of performance, and if flow entries are created too frequently, in some use cases, customers can notice measurably worse performance.
Newer versions of OvS (1.11 and upwards) introduce new ways to overcome these problems and will be made available in releases following 6.2. However, XenServer v6.2 or before, make use of the versions of OvS (1.0 and 1.4) and can be adversely affected by frequent flow creations.

There are several ways in which OvS, and the XenServer system, can be configured to minimize or avoid these problems altogether. These are discussed in more detail in the following three sections.

3 Avoiding physical switch flooding

**Problem 1** - A frequently observed problem can occur if the physical switches in the network start to flood packets on their ports. It can happen in overloaded situations, for example when guest VMs can't reply to the traffic they receive in a timely fashion, and therefore the upstream switches times out their MAC address entry in the MAC learning table. As the switches no longer have a record of where the guest is, they begin to broadcast all traffic to these MAC addresses.

**Solution** –
The first thing to overcome this situation is to make sure the guest will reply to its incoming traffic, at least to some part of it. For example, by traffic shaping or bandwidth limitation to prevent overload and see if this reduces the flooding.

**Problem 2** - It is also possible that another node(s) on the same segment triggers this flooding. If the host has multiple network cards connected to the same Ethernet segment, they will also receive this broadcasted traffic, and the flow creation burden will be multiplied.
Solution - One approach to handle this situation is to try to isolate the traffic flowing towards the XenServer host from other network traffic.

Problem 3 - Some switches implement Network Loop Protection (NLP) that performs similar flooding. Flooding will always have a performance impact on the whole network, even if other nodes may be not that sensitive to this kind of problem.

Solution – Disabling NLP improves the situation and often the number of persistent flow entries can drop from several 1000’s+ to 100s.

Problem 4 - Another common issue is when the switch ports connected to the XenServer host are in VLAN trunking mode. This will deliver every VLAN’s traffic to the host, even though this is not necessary. There are some physical switches which are sending probing packets on every possible VLAN id, and that can cause 4094 new flow entries every time such a probe occurs.

Solution – Tag the VLAN to the ports on the physical switch which are necessary. This will depend on the physical switch configuration.

4 Increasing the flow-eviction threshold

The threshold is a type of limit on the number of flows that are cached in the kernel. OVS will handle as many flows as can be processed through ovs-vswitchd.

If the number of newly created connections reaches this limit, in a 5 second window, OvS attempts to keep the memory consumption under the limit by evicting older flow entries.

It is possible to increase this flow-eviction threshold. This is the recommended first step to resolve issues related to the number and frequency of flow entries being created,

To set the flow-eviction-threshold enter the following:

ovs-vsctl set bridge <bridgename> other-config:flow-eviction-threshold=<new value>

Notes:
- The maximum possible value is 65535 (2^16-1).
- Recommended values are in the range 1,000 to 10,000
- The setting is not persistent after host reboot but the command can be added to the end of the /etc/rc.local file
- If an upgrade or update changes the rc.local file then the commands may not persist so – this needs to be checked
- Please note that on XS v6.0 the option to modify the value is not available, where it is on 6.0.2, 6.1 and 6.2.
Increasing the setting will increase the maximum OvS memory consumption by a small amount and, under heavy network loads, this represents a trade-off of kernel memory consumption versus userspace processing. This is explained in more detail in section 6.3. Hence – tuning for the given workload and system implementation details may be required.

The details are explained later in the section, ‘Considerations when adjusting the flow-eviction-threshold and impact to Dom0 memory and CPU consumption’.

5 Switching to Linux Bridge

If flooding is not the problem, and tuning the flow-eviction threshold is not effective, then switching back to Linux bridge mode is a valid option.

This can be achieved by running the following command on each host in the pool –

`xe-switch-network-backend bridge`

Note – to become effective – this requires a host reboot.

Notes:
- Citrix fully supports both the Linux bridge and OvS on XenServer (6.X).
- New networking features will be introduced to the default OvS configuration.
- Customers are encouraged to contact support for assistance to resolve issues related to the OvS.

Customers can safely revert back to the Linux bridge, with the knowledge that it is fully supported with its existing functionality.

If it is necessary to revert back again to the OvS backend, run the following command on each host to reverse the setting: (Requires reboot to become effective)

`xe-switch-network-backend openvswitch`

6 Identifying the problem and potential solutions

6.1 Using a status report

Upload a status report to the Citrix Predictive Support site – [https://taas.citrix.com](https://taas.citrix.com)

Issues will be identified in the analysis output.
The status report can be manually examined for the following files in the bug report folder to determine if the flow-eviction-threshold has been exceeded:

**ovs-vsctl-list-bridge.out** – Lists the virtual switches and flow-eviction-threshold if manually adjusted

**ovs-dpctl-show.out** – Lists the flow values per virtual switch at the time that the status report was collected.

### 6.2 Using the command line on a running host

To query whether the flow-eviction-threshold was set explicitly:

```
ovs-vsctl get bridge <bridgename> other-config:flow-eviction-threshold
```

If the value has been manually configured, the command will return the value, otherwise a failure message will display, and then the default value would apply.

The default settings for different XenServer versions are as follows:

- **6.0 default value = 1000; customers cannot modify this value, even if the above command appears to succeed.**
- **6.0.2 default value = 1000; installing Hotfix XS602E024 allows customers to modify the value.**
- **6.1 default value = 1000; customers can modify the value without applying hotfixes. Note: Hotfix XS61E035 increases the default value to 2500**
- **6.2 default value = 2500**

Customers can use the active flow-eviction-threshold value to evaluate the current number of active flows by using the following command:

```
watch -n 1 ovs-dpctl show
```

This command prints out the bridge statistics every second, until the process is ended.

The flows counter displays the actual number of current flow entries. If the number of flows approaches the flow-eviction-threshold, OvS has to deal with a lot of flow creations and deletions. In this case increasing the flow-eviction-threshold should help address the issue.

The lookups line contains the following information:

- **Hit**: These are the packets that hit an existing entry in the kernel hash lookup table
- **Missed**: The number of packets sent to userspace and were not found in the kernel hash lookup table.
• **Lost**: The number of packets lost before arriving at the userspace module

Lost packets in **ovs-dpctl-show** usually denotes netlink queue overrun.

### Are the OVS processes overloaded?

```
# ovs-dpctl show
# Shows Flow Statistics
```

```
[root@xs-baaxs01 ~]# ovs-dpctl show
system@xenbr0:
  lookups: hit:481193439 missed:417542450 lost:0
  flows: 250
  port 0: xenbr0 (internal)
  port 1: eth0
  port 103: vif58.0
  port 104: tap58.0
system@xapi1:
  lookups: hit:1170068531 missed:848755070 lost:0
  flows: 518
  port 0: xapi1 (internal)
  port 1: eth1
  port 2: eth2
  port 8: vif52.0
```

Use the **missed** counter, to check the rate at which new flows are created. This counter counts every occasion when a new flow has to be created. If the pace of increment on this counter exceeds a few hundred per second, check the source of the new connections, as they are probably causing a noticeable performance degradation. Especially if the virtual switch doesn't have an active VM on it, rapid increase means unwanted traffic keeps the OvS busy.

The following command creates a detailed list about currently active flows, and can be useful to verify the types of traffic the OvS sees:

```
ovs-dpctl dump-flows <bridgename>
```

### 6.3 Considerations when adjusting the flow-eviction-threshold and impact to dom0 memory and CPU consumption

Increasing the flow-eviction-threshold will increase the CPU utilization of the openvswitch userspace daemon (ovs-vswitch) which can slightly also decrease the amount of ‘low’ memory available in the Control Doman (dom0) which may have a slight performance impact.

The follow discussion explains the relative importance of these 2 effects – memory consumption and cpu utilization.
**Memory impacts** — Dom0 memory can be loosely described as being partitioned into *low memory* and *high memory*. The low memory region is used for a variety of performance critical operations and is limited in size.

Increasing the flow-eviction-threshold does not automatically increase the low memory used for storing the datapath flows. Datapath flows are stored in the slab cache (sw_flow). Each flow object (or a flow in the slab cache) is 164B.

Therefore if there are 10K flows in the kernel datapath the low memory used would be 10K*164 => 1.6MB  This does not include the size used by actions for each flow (size of actions are quite small).

Also note that the number of flow slab objects might be greater than the number of flows in the system. When a kernel flow is deleted from the flow table, the kernel might decide not to delete the slab object. Unless there is memory pressure in the system, some of slab objects are retained in the slab cache (warm cache) even though the flow is deleted.

For example, in the dump shown below, there are 18 (for xenbr0) + 16 (for xenbr1) = 34 flow in the system, but the number of active_objects in sw_flow slab cache is 127. Memory utilized by flow in the kernel is equivalent to the size of sw_flow slab cache.

```
cat /proc/slabinfo | grep sw_flow
# name  <active_objs> <num_objs> <objsize> <objperslab> <pagesizeperlab>  : tunables <limit> <batchcount>
<sharedfactor>  : slabdata <active_slabs> <num_slabs> <sharedavail>
sw_flow 127    336    164   24    1 : tunables 120   60
          8 : slabdata 14  14      6
```

```
ovs-dpctl show
system@xenbr0:
  lookups: hit:165504 missed:77724 lost:0
  flows: 18
  port 0: xenbr0 (internal)
  port 1: eth0
system@xenbr1:
  lookups: hit:38981 missed:76583 lost:0
  flows: 16
  port 0: xenbr1 (internal)
  port 1: eth1
```

**CPU utilization impacts** - Userspace (ovs-vswitchd) requires the datapath flow statistics (from the kernel) to decide which datapath flows to evict (time out). The ovs-vswitch does this by requesting the kernel (every second, in some scenarios it could be even less than a second) to send the datapath flow statistics. Based on the statistics received, the switch can decide which flows to evict. ovs-vswitchd needs to process statistics update, flow miss, flow eviction, bond rebalancing and so on, in a single thread. If the flow-eviction-threshold is increased to e.g. 65K and there are 60K flows in the kernel datapath, then the openvswitch kernel would need to send information about all these 60K flows to userspace.
Userspace processes the statistics update for 60K in a linear fashion (which involves copying kernel stats netlink message and updating stats for each flow) and this is a lengthy process.

Testing results have shown that it can take around 500ms to process a 60K flow. Spending 500ms in 1 second stats interval, for a statistics update, wastes valuable CPU resource. Higher flows in the kernel, increase the time spent by ovs-vswitchd to manipulate statistics.

**Summary when adjusting the flow-eviction threshold** – As per the above – the dominant effect is on CPU utilization and not on memory (as the memory impact is small and the system has methods for reducing and handling this impact further). The CPU utilization can be monitored using top in dom0 to ensure that ovs-vswitchd is not using large amounts of CPU and starving the rest of the system of resources.

7  **Summary of resolution options**

The options available at this point – for users of XenServer 6.2 or before which make use of the versions of OvS (1.0 and 1.4) – are summarized as follows:

**Option 1**
Investigate and remove any potential sources of flooding.

**Option 2**
Change, and tune as necessary, the flow eviction threshold where possible to adjust the number of flows – whilst monitoring the CPU utilization of ovs-vswitchd in dom0.

**Option 3**
Switch the networking backend from OvS to Linux bridge and investigate whether this resolves the issues seen.

8  **More Information**

We are aware of the impact this issue has and we are constantly working to improve the XenServer product with each updated release and plan to address this condition in the future.
9 Disclaimer

The above mentioned sample code is provided to you AS IS with no representations, warranties or conditions of any kind. You may use, modify and distribute it at your own risk. CITRIX DISCLAIMS ALL WARRANTIES WHATSOEVER, EXPRESS, IMPLIED, WRITTEN, ORAL OR STATUTORY, INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NONINFRINGEMENT. Without limiting the generality of the foregoing, you acknowledge and agree that (a) the sample code may exhibit errors, design flaws or other problems, possibly resulting in loss of data or damage to property; (b) it may not be possible to make the sample code fully functional; and (c) Citrix may, without notice or liability to you, cease to make available the current version and/or any future versions of the sample code. In no event should the code be used to support of ultra-hazardous activities, including but not limited to life support or blasting activities. NEITHER CITRIX NOR ITS AFFILIATES OR AGENTS WILL BE LIABLE, UNDER BREACH OF CONTRACT OR ANY OTHER THEORY OF LIABILITY, FOR ANY DAMAGES WHATSOEVER ARISING FROM USE OF THE SAMPLE CODE, INCLUDING WITHOUT LIMITATION DIRECT, SPECIAL, INCIDENTAL, PUNITIVE, CONSEQUENTIAL OR OTHER DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Although the copyright in the code belongs to Citrix, any distribution of the code should include only your own standard copyright attribution, and not that of Citrix. You agree to indemnify and defend Citrix against any and all claims arising from your use, modification or distribution of the code.