Achieving a fair distribution of the processing of guest network traffic over available physical CPUs

January 2011 — Version 1.1

CITRIX®

1 Summary

Static allocation of virtual network interfaces (VIFs) and interrupt request (IRQ) queues among available CPUs can lead to non-optimal network throughput. This article explains the details of this problem, and how best to avoid it.

2 Requirements

This article applies to a host with XenServer 5.6 FP1 (XS 5.6 FP1) that is experiencing heavy network traffic on virtual machines.

3 Background

All guest network traffic is processed by `netback` processes in the control domain on the guest's host. There is a fixed number of `netback` processes per host (four by default in XS 5.6 FP1), where each one is pinned to a specific control domain virtual CPU (VCPU).[1] All traffic for a specific VIF of a guest is processed by a statically-allocated `netback` process. The static allocation of VIFs to `netback` processes is done in a *round robin* fashion when virtual machines (VMs) are first started after XenServer restart. The allocation is not dynamic, since that would require further synchronisation mechanisms, and would close any active connection on every rebalance.

For example, suppose we have a host with four CPUs, with four `netback` processes (running on four VCPUs in the control domain), and two physical network interfaces (PIFs). Furthermore, suppose the host has four VMs, where each VM has two VIFs that correspond to PIFs (in the same order for all VMs), and that VMs are started one after another (which is normally the case). Then, the first VIF of all four VMs will be allocated to odd-numbered `netback` processes, while the second VIF of all four VMs will be allocated to even-numbered `netback` processes. Therefore, if VMs send network traffic only on their first VIF, then only two of the available four CPUs will be utilised. See Figure 1.

With heavy network traffic on fast network interfaces, CPU can quickly become the bottleneck, so being able to utilise all available CPUs is crucial.

4 Procedure

The goal is to achieve a distribution of VIFs on the available `netback` processes which distributes the processing of network traffic to desired physical CPUs; in most cases, we aim to evenly distribute over all available physical CPUs.

By starting VMs in a predictable manner, we know which VIF is allocated to which `netback` process (and therefore to which virtual CPU in the control domain) — see §3 for detailed information.

By either analysing XenCenter networking performance for VMs, or by using various networking tools within the VMs, we can determine relative throughput requirements for each VIF.

[1] In XS 5.6 FP1, the control domain has four VCPUs by default, as opposed to one VCPU in all previous version of XenServer.

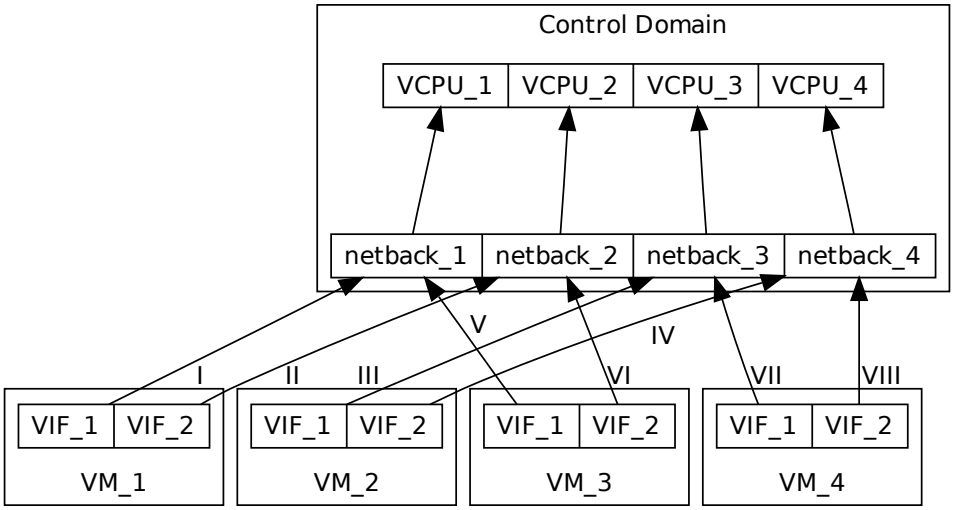

Figure 1: Example distribution of VIFs on the available `netback` threads. *The roman numbers indicate the order in which VIFs are allocated to* `netback` *threads.*

By observing (with `top`; press "1" in `top` to see distribution for each VCPU) relative VCPU usage of `netback` processes within the control domain, and by using the information obtained above, we can determine the optimal allocation of VIFs on available `netback` processes.

There are at least three ways to modify the allocation of VIFs on `netback` processes:

- modify the order in which VMs are first started;
- modify the order of declared VIFs (on VMs); and,
- add dummy VIFs to VMs.

Continuing with the working example from §3, suppose we find that all guest-level network traffic is going through the second VIF of each VM, and that all VMs require roughly the same network throughput. There are two simple solutions to achieving optimal network throughput here:

1. Add a VIF to each VM, restart host, start VMs in order — the second VIFs are allocated to `netback` processes (in this order) 2, 1, 4, and 3.
2. Switch the order of the two VIFs on the third and the fourth VMs, restart host, start VMs in order — the relevant VIFs (those that were previously the second VIFs) are allocated to `netback` processes (in this order) 2, 4, 1, and 3.

5 Interrupt Queues

When a network package arrives on a specific NIC on a host, it creates an *interrupt request* (IRQ), which is placed into one of the NIC's IRQ queues. To find out more about the host's current IRQ queues and VCPU affinities, see

`/proc/interrupts` on the control domain. These queues are regularly processed by the control domain's VCPUs.

In XenServer 5.6 FP1, the control domain has four VCPUs; however, by default, all IRQ queues are processed by VCPU-0. When the host experiences heavy network traffic, VCPU-0 can become a bottleneck.

The simplest way to optimise this is to install and start a daemon called `irqbalance`[2] It will dynamically allocate IRQ queues across all control domain's VCPUs. Note that some time is normally required for `irqbalance` to adapt to the network traffic patterns.

Therefore, to effectively distribute IRQs for a single NIC across multiple VCPUs, it is essential that the NIC exposes multiple IRQ queues. Ideally, the number of IRQ queues of a NIC equals the number of VCPUs.

To manually set VCPU affinity of an IRQ queue *irqn*, write an appropriate VCPU mask to `/proc/irq/`*irqn*`/smp_affinity`. For example, if you want the IRQ queue `1274` to be processed by the third VCPU, set the contents of `/proc/irq/1274/smp_affinity` to `100`.

6 Version History

Version 1.1

Author(s) Rok Strniša.

Reviewer(s) Jonathan Davies, Matthias Görgens.

Date January 2011.

Comment Added a section about interrupt queues.

Version 1.0

Author(s) Rok Strniša.

Reviewer(s) Jonathan Davies, Matthias Görgens.

Date January 2011.

Comment Initial version.

[2] In future releases of XenServer, `irqbalance` daemon is expected to be running by default.