Computer Engineering Department

Implementing Memory Ballooning Concept in OpenBSD Hypervisor
Project Advisor: Mike Larkin

Ruchika, Hazariwal
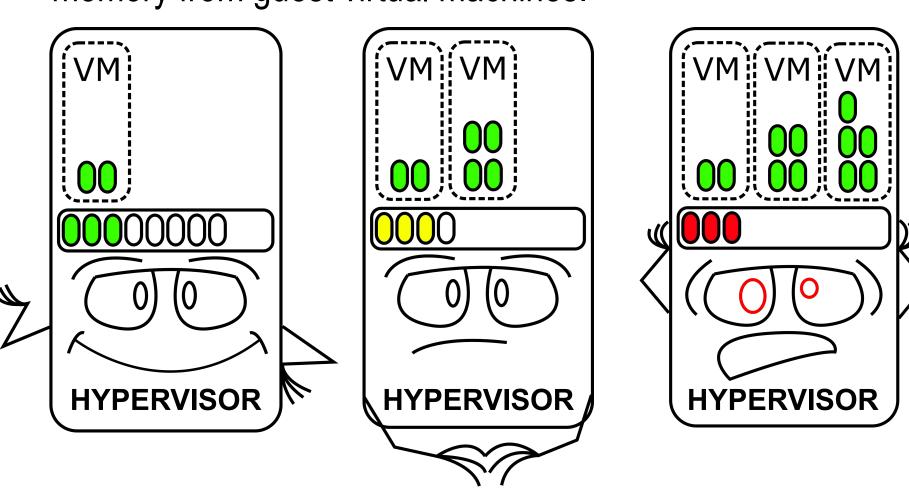(MS Software Engineering)
Hemaprasanthi, Mutyala
(MS Software Engineering)
David, Montes (MS Software Engineering)
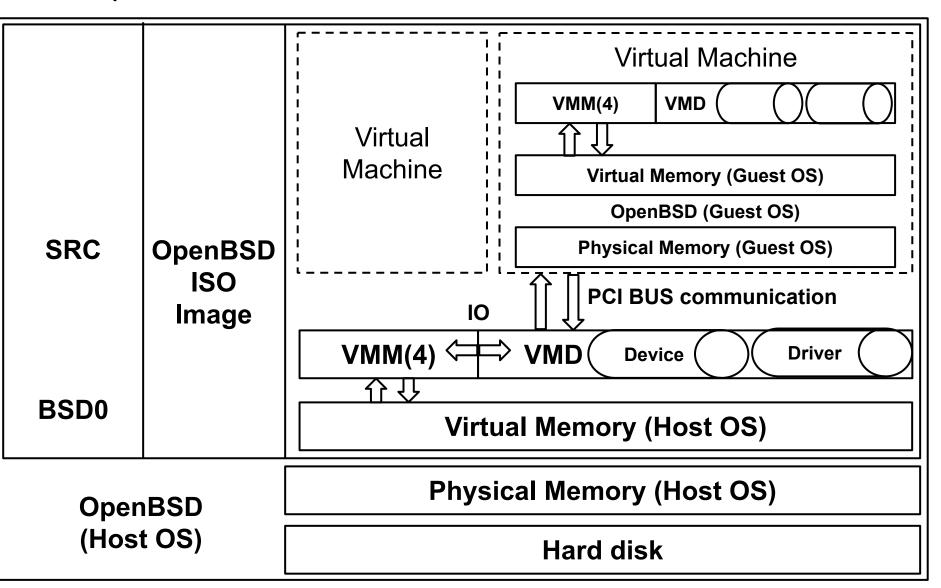**Anjala, Thulasiram (MS Software Engineering)**

Introduction

The purpose of this project is to develop a dynamic memory management technique that enables a hypervisor to reclaim unused memory from virtual machines(VM). A hypervisor is a host to guest virtual machines and each guest is assigned a portion of the hypervisor's memory. The hypervisor over-commits its memory resources when assigning memory to the guest VMs in order to maximize the use of its available memory. When a hypervisor is low on memory, it needs a mechanism to retrieve unused memory from guest virtual machines.
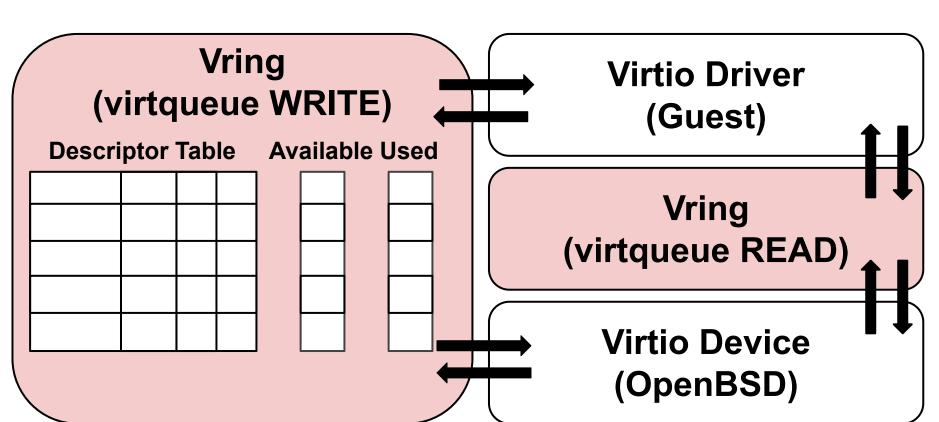

Methodology

The major components for the OpenBSD OS environment are shown in the following architecture diagram. Our work was focused on the VMM(4), VMD, driver and device components.

Virtqueues:
Our device, along with all VIRTIO devices, use one or more virtqueues as their primary method of bulk transportation of data. A virtqueue is comprised of a descriptor table, available ring and used ring. Our device uses three virtqueues: stats, inflate and deflate.

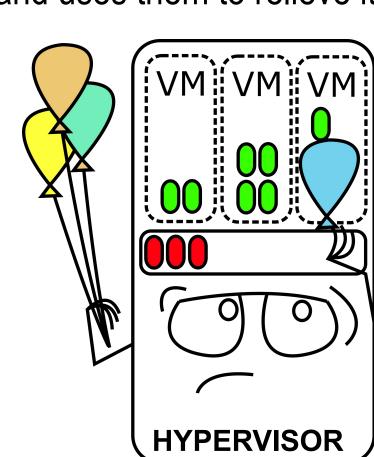Each virtqueue can be implemented to accomplish
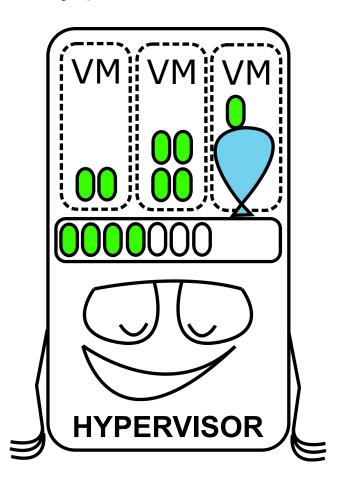
Methodology

different tasks but the underlying functionality is the same. Let us examine one of these implementations in depth. The stats queue is used to gather memory statistics from the guest. The communication in the stats queue is initiated by the driver. When the guest VM is initialized, the balloon driver places an empty buffer in the descriptor and notifies the device.

When the host is low on memory, it sends a used buffer notification to the guest. The guest answers the request by adding the required statistics into the descriptor, adding the index into the available ring and notifying the device. The device will view the statistics and use that information to determine how much memory it should reclaim.
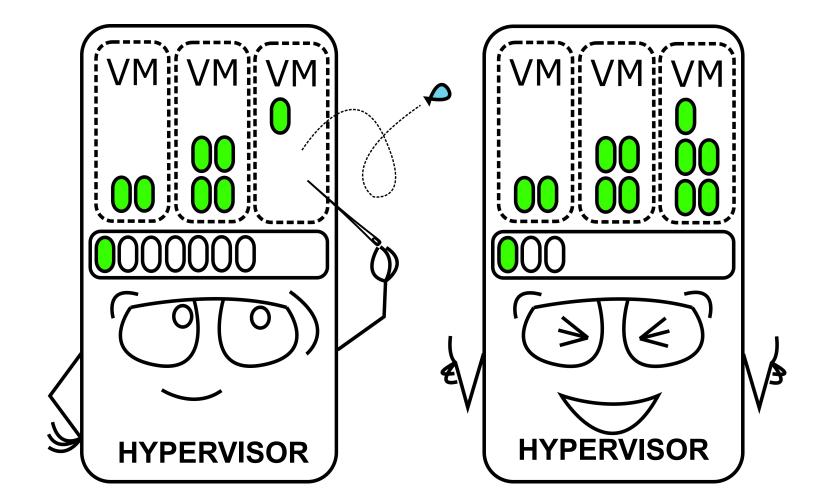
Balloon Inflation Concept:
The hypervisor initiates balloon inflation with a configuration change. When the driver receives that interrupt, it compares the desired pages to the actual pages. The driver calculates the difference between the total desired pages and the actual pages needed by the hypervisor. It allocates those pages and returns them in the inflate queue. The hypervisor receives these pages and uses them to relieve its memory pressure.
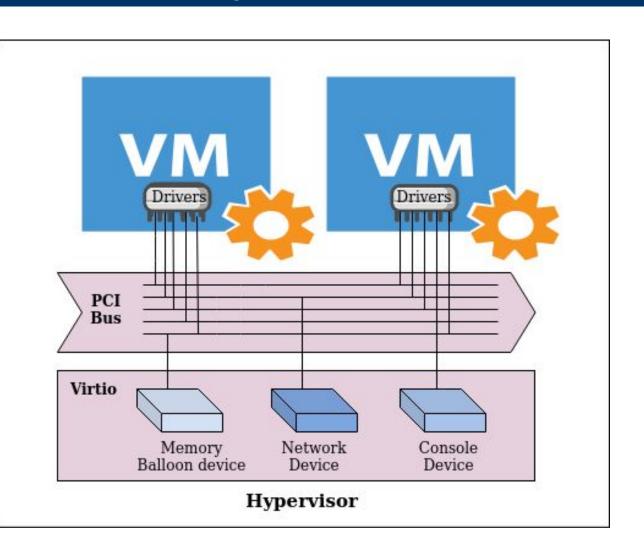

Balloon Deflation Concept:
The hypervisor keeps a record of the pages that it reclaims. After the memory pressure is gone, the hypervisor sends a configuration change notification to the driver and returns the pages via the deflate queue.

Device – Driver Attachment:
The virtual machine must successfully negotiate the feature bits to attach the memory balloon driver to the device via the PCI bus.

Analysis and Results

The memory ballooning device can be seen successfully attached to the pci bus in the pcidump command output.

```
inner# pcidump
Domain /dev/pci0:
    0:0:0: OpenBSD VMM Host
    0:1:0: Qumranet Virtio RNG
    0:2:0: Qumranet Virtio Network
    0:3:0: Qumranet Virtio Storage
    0:4:0: OpenBSD VMM Control
    0:5:0: Qumranet Virtio Memory Balloon
```

NOTE: The device can be operated by the hypervisor based on its memory condition or it can be manually run by VMCTL.

Memory Statistics:
The VMCTL stats command directs the hypervisor to collect memory statistics from the guests. The hypervisor chooses to collect six major statistics that it will used to determine its reclamation strategy. Those six statistics along with their tags and values are displayed below.

```
viombh_notifyq: stats[0] for Swap pages in use: tag=0x0 val=0x0
viombh_notifyq: stats[1] for Pages swapped out: tag=0x1 val=0x0
viombh_notifyq: stats[2] for Faults: tag=0x2 val=0x1cc11
viombh_notifyq: stats[3] for Free pages:  tag=0x4 val=0x1566f000
viombh_notifyq: stats[4] for Total Number of Pages: tag=0x5
val=0x1dd0f000
viombh_notifyq: stats[5] for Number of Buffered Pages: tag=0x7
val=0x47f400
viombh_notifyq: leaving
```

Inflation:
The VMCTL inflate command directs the hypervisor to reclaim a specific amount of memory from the guests. The guest reads the request and inflates the balloon for that specific memory size. The following example specifies 8192 bytes which is converted to two pages that are 4096 bytes each.

```
outer# vmctl balloon -m 8192m test
vmctl: balloon adjusted on vm 1 successfully
```

Deflation:
The VMCTL deflate command directs the hypervisor to deflate the guest balloons. The guest will be notified of the deflation and will be given back all of its original pages. The following example shows the deflate command being successfully executed.

```
outer# vmctl balloon -d test
vmctl: balloon adjusted on vm 1 successfully
```

Summary/Conclusions

This project adds a new feature to the open source OpenBSD project. It depends on the functionality provided by OpenBSD and the VMM(4) hypervisor in order to be used. It is not portable to other operating systems or hypervisors as it belongs to the VIRTIO family of devices.

The memory ballooning device enables each OpenBSD hypervisor to communicate with its guest VMs to reclaim their unused memory. When the hypervisor needs more memory, each guest is able to allocate its unused pages. These pages are returned to the hypervisor, thus inflating the balloon. The hypervisor is able to return these same pages back to the guests by deflating the balloons.

This memory balloon concept in OpenBSD enables the hypervisor to maximize its limited memory resources to support multiple guest virtual machines by dynamically reallocating memory as needed between the guest and hypervisor.

Key References

[1] Waldspurger, C. A. (2002). Memory resource management in VMware ESX server. ACM SIGOPS Operating Systems Review, 36(SI), 181. doi: 10.1145/844128.844146.

[2] Wang, J., & Wang, B. (2016). A hybrid main memory applied in virtualization environments. 2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI). doi:10.1109/cci.2016.7778955.

[3] Moniruzzaman, A. B. M. (2014). Analysis of Memory Ballooning Technique for Dynamic Memory Management of Virtual Machines (VMs). International Journal of Grid and Distributed Computing, 7(6), 81–90. doi: 10.14257/ijgdc.2014.7.6.07

[4] VMware Virtualization. (2011). SciVee. doi: 10.4016/33276.01]

[5] Virtual I/O Device (VIRTIO) Version 1.1. Edited by Michael S. Tsirkin and Cornelia Huck. 20 December 2018. OASIS Committee Specification Draft 01 / Public Review https://docs.oasis-open.org/virtio/virtio/v1.1/csprd01/virtio-v1.1-csprd01.html.

Acknowledgements