ORACLE

Hot-resizing of Windows VMs via Hyper-V Dynamic Memory protocol

Oracle

Maciej S. Szmigiero
Linux Virtualization and Security
September 13, 2022

Problem statement

- This talk describes a current attempt to add hot memory (RAM) resizing capability to QEMU Windows VMs, via their native Hyper-V Dynamic Memory protocol
- Why support Windows at all in Open Source / Free Software ecosystem?

Problem statement

- This talk describes a current attempt to add hot memory (RAM) resizing capability to QEMU Windows VMs, via their native Hyper-V Dynamic Memory protocol
- Why support Windows at all in Open Source / Free Software ecosystem?
- The same reason for other Windows-specific stuff like Samba: customers want it to not have to switch to 100% Windows-based solutions

Existing solutions

Why hot-resizable VMs?

- Hard to forecast memory requirements with 100% accuracy
  - Best if VM size grows smoothly together with its workload
    - At the same time nobody wants to pay for unused capacity
  - Cloud customers don't like to reboot their VMs, for various reasons
    - Best if we can adjust the size without a guest reboot
  - Potentially offer a "budget" hosting where customer VMs can get dynamically resized depending on their host occupacy

 | |

Existing solutions

Why hot-resizable VMs?

- Hard to forecast memory requirements with 100% accuracy
  - Best if VM size grows smoothly together with its workload
    - At the same time nobody wants to pay for unused capacity
  - Cloud customers don't like to reboot their VMs, for various reasons
    - Best if we can adjust the size without a guest reboot
  - Potentially offer a "budget" hosting where customer VMs can get dynamically resized depending on their host occupacy
- Let's review the existing VM resize solutions

 | |

Existing solutions

ACPI-based PC DIMM hotplug

- Slot limit
  - Only 256 memory slots in QEMU
  - Just blindly increasing that limit probably won't scale too well
    - There are problems already with ACPI table size in some configurations
- Very high granularity
  - Side effect of the slot limit to reach the required guest size dynamic range
  - Makes removal of a module difficult since a single busy page can prevent it
- Ripple effect on removal possible

 | |

Ripple effect on removal

- There are 3 extra DIMMs plugged into the guest: A, B, C.
  - A and B are nearly empty, but C is nearly full.
- The host does not know anything which DIMM is empty and which is full

Ripple effect on removal

- There are 3 extra DIMMs plugged into the guest: A, B, C.
  - A and B are nearly empty, but C is nearly full.
- The host does not know anything which DIMM is empty and which is full
- It requests the guest to unplug the stick C
  - The guest copies the content of the stick C to the stick B

 | |

Ripple effect on removal

- There are 3 extra DIMMs plugged into the guest: A, B, C.
  - A and B are nearly empty, but C is nearly full.
- The host does not know anything which DIMM is empty and which is full
- It requests the guest to unplug the stick C
  - The guest copies the content of the stick C to the stick B
- Again, the host does not know anything which DIMM is empty and which is full, so it requests the guest to unplug the stick B
  - The guest now has to copy the same data from the stick B to the stick A, again

 | |

virtio-mem

- Large block size
  - Even 1 MiB (the current minimum) is 256 pages
    - Same problem as with DIMM hotplug, just not as severe
- No native Windows driver
  - There was some attempt presented at last year's DevConf.CZ by Marek Kędzierski,
    see: `https://www.youtube.com/watch?v=I_XvMBHzmDw`
  - But it's still rather challenging

 | |

virtio-mem

- Large block size
  - Even 1 MiB (the current minimum) is 256 pages
    - Same problem as with DIMM hotplug, just not as severe
- No native Windows driver
  - There was some attempt presented at last year's DevConf.CZ by Marek Kędzierski,
    see: `https://www.youtube.com/watch?v=I_XvMBHzmDw`
  - But it's still rather challenging
- Some of these issues are theoretically fixable
  - It's closed-source Windows we are talking about, not Linux
  - Would be much more invasive
    - This QEMU host driver is (almost) a self-contained solution

Ballooning

- Required to be able to hot-remove memory from guest with high granularity
  - Ballooning usually operates with single page (minimum) granularity
- Ballooning↔hot add driver integration desirable
  - This way balloon gets deflated first when increasing the guest size before attempting hot-add

Ballooning - virtio-balloon

- virtio-balloon is the current *preferred* QEMU ballooning solution
- The Windows driver actually marks the pages that were ballooned out as *in use* inside the guest
  - Prevents removing of the DIMM stick backing them
- This is a pure ballooning driver / protocol
  - No way to resize the guest up past its boot size
- Low performance due to operating on single pages
  - Probably fixable if it turns out to be a real performance bottleneck

 | |

Hyper-V Dynamic Memory protocol

- Uses Hyper-V VMBus
  - Neither bus nor protocol are documented in the Hyper-V Top Level Functional Specification
  - QEMU host support already developed by Virtuozzo - thanks guys!
- Linux kernel (client) drivers are a huge help
  - Don’t document Windows-specific details
- Built-in support for this protocol in Windows Server 2012 R2, Windows Server 2016 and Windows Server 2019
- Hot-add support needs S4 disabled
  - Incompatible with some “VM freezing” solutions via its hibernation

 | |

Hyper-V Dynamic Memory protocol

- Good news: Windows seems to be rather determined to free the requested number of pages: waiting for the guest to reply to a 2 GiB balloon request sometimes takes 2-3 seconds
  - Possibly doing some kind of memory compaction during that time
  - Enforcing the balloon floor is best left to the guest
    - It knows best its internal memory requirements
- Somehow related to *Dynamic Memory* VM setting in Hyper-V
  - This only controls *automatic* VM size management
  - User requests will use the DM protocol regardless of this setting

hv-balloon driver

- A new QEMU VMBus driver was developed
  - Named hv-balloon after the Linux kernel client driver for the Dynamic Memory Protocol
    - And to follow the naming pattern established by the virtio-balloon driver
- This driver supports both memory hot-add / hot-removal and ballooning
  - Resizing currently plugged into QEMU ballooning commands (*balloon* and *info balloon*)
- What this driver is NOT
  - A cross-platform VM resizing driver
    - Currently x86 only, ARM64 maybe in a distant future
  - Linux guests aren't in scope

First implementation

- Using an universal backing devices for memory hot-add protocols
  - “haprots"
  - DM protocol registers as a provider for these
- Allows removing memory from guest in single page (4k) units
  - Via built-in ballooning operations support
- Careful not to trust the guest
  - Can maliciously report returning pages outside its current address space
    - These can later clash with the address range of hot-added memory

First implementation

Range trees

- Guest-released memory is tracked in range trees
  - As a series of (start, count) ranges
- Required a few new *GTree* operations
  - These were released as a part of Glib 2.68
  - Presence detected at configure time to allow building QEMU with older Glib versions

 | |

First implementation

Range trees

- Guest-released memory is tracked in range trees
  - As a series of (start, count) ranges
- Required a few new *GTree* operations
  - These were released as a part of Glib 2.68
  - Presence detected at configure time to allow building QEMU with older Glib versions
- This gave much better ballooning performance than virtio-balloon
  - 230 GB / minute versus 70 GB / minute on Xeon E5-2699

 | |

Haprot

- A haprot device works like a virtual DIMM stick
  - Allows inserting extra RAM into the guest at run time
- Some differences from the ACPI-based PC DIMM hotplug
  - Notifying the guest about the new memory range done via a protocol handler that registers with the haprot framework
  - The ACPI DIMM slot limit does not apply
  - Virtual DIMM size determined at insertion time
  - Protocol handler can inform the guest about removal of a haprot device and / or do its own cleanup

 | |

Scalable memslots

- Required since each hot-added memory region needs a new memslot
  - Doesn’t apply to un-ballooned regions of course
- Operates on "pay as you go" basis, that is, that the user only pays the price of the memslot count that is actually used, not of the maximum count allowed
  - The previous implementation had quite a few linear scans
  - Allowed reasonable performance with the maximum memslot count increased to 32k
- Operation semantics were carefully matched to the original implementation
  - The outside-visible behavior did not change
- Makes lookup and memslot management operations O(*log*(*n*))

Scalable memslots

- Required since each hot-added memory region needs a new memslot
  - Doesn’t apply to un-ballooned regions of course
- Operates on "pay as you go" basis, that is, that the user only pays the price of the memslot count that is actually used, not of the maximum count allowed
  - The previous implementation had quite a few linear scans
  - Allowed reasonable performance with the maximum memslot count increased to 32k
- Operation semantics were carefully matched to the original implementation
  - The outside-visible behavior did not change
- Makes lookup and memslot management operations O(*log*(*n*))
- Available in kernel 5.17

Summary

- This was just the first attempt / MVP
  - Still a lot of work to do
- Figure out what to do on guest reboot
  - Hyper-V seems to resize the boot memory to match the current guest size in this case
    - May need memslot resize operation… or not
  - We try to avoid re-launching QEMU
  - Currently, the virtual DIMM sticks are re-inserted after the guest reconnects to the DM interface
    - Marked *not in use* at reboot before re-insertion

 | |

Summary

- Avoid using virtual DIMM sticks as an unit of memory management
  - Will make it easier to directly control guest size from HMP
  - Avoids having to control QEMU via libvirt-like external entity
    - To listen for virtual DIMM becoming not in use and remove them
- Make NUMA aware
  - DM protocol hot remove request has a *virtual_node* member
  - Need to investigate how to tell Windows about the node number of hot added memory
- Allow hv-balloon + virtio-balloon coexistence
  - Covers both Linux and Windows guests
  - Maybe even integrate them as backends of a common user interface?

Demo

Laptop-scale live demonstration

Q & A

Questions?