Instant Detection of Virtual Devices

Ajay Kaher (akaher@vmware.com)
Alexey Makhalov (amakhalov@vmware.com)
Ashwin Dayanand Kamat (kashwindayan@vmware.com)

vmware

Agenda

- Importance of Boot-time for Virtual Machines
- Initialization time of Virtual Devices
- MMIO Direct Read, Skip-write, Pre-configured PCIe config
- Improvement and Suggestions

Importance of Boot-time for Virtual Machines

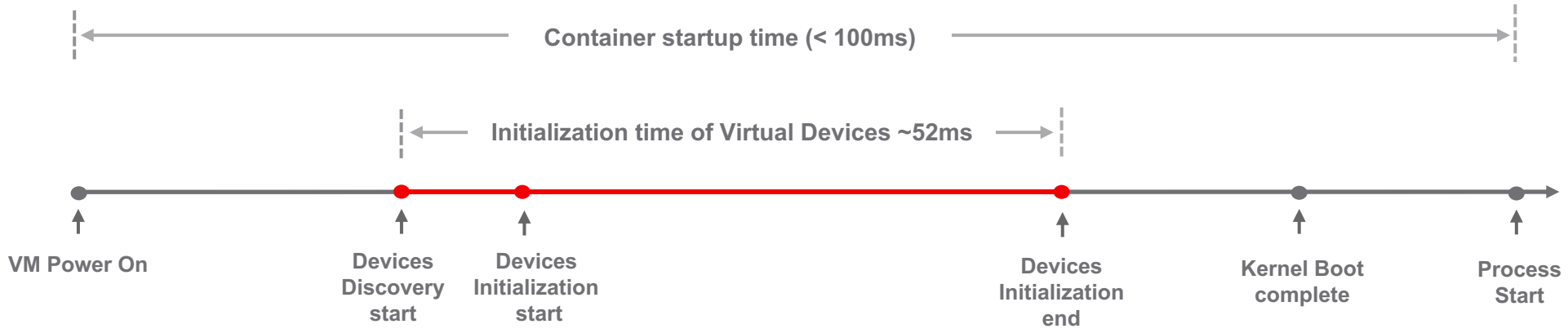
- CRX (Container Runtime for ESXi), it’s a VM based Secure Container
- Kata containers is a secure container runtime with lightweight virtual machines
- Faster boot is a critical feature for CRX, Kata containers which compliments it to behave like a container

Importance of Boot-time for Virtual Machines

- CRX (Container Runtime for ESXi), it’s a VM based Secure Container
- Kata containers is a secure container runtime with lightweight virtual machines
- Faster boot is a critical feature for CRX, Kata containers which compliments it to behave like a container

Problem Statement: Initialization time of Virtual Devices (>50% of Kernel boot time)

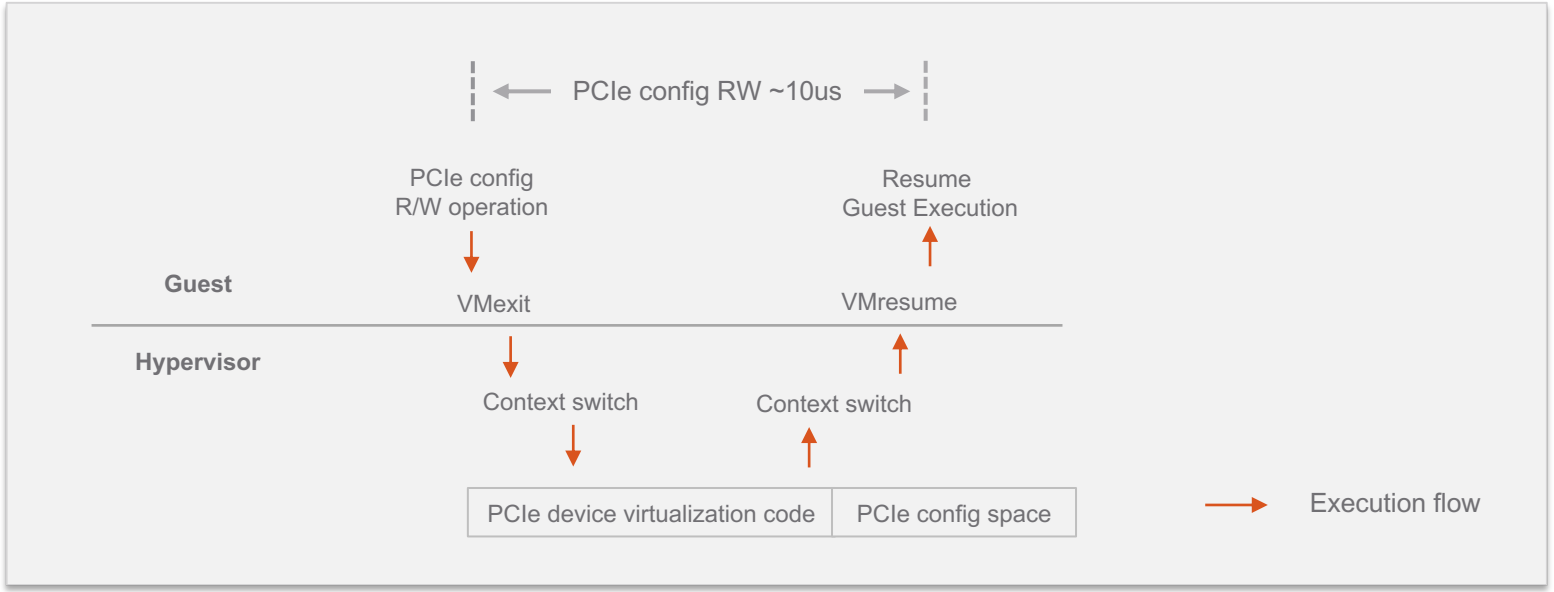
- Significant amount of time is taken in Devices Detection and Initialization, as highlighted with red line
- CRX boot time is ~100ms and ~52ms is to initialize the virtual devices (>50% of Kernel boot time)
- ~52ms is because of the 'PCIe config Read/Write' calls from guest.



Goal: Reduce Initialization time of Virtual Devices

PCIe config Read/Write operations

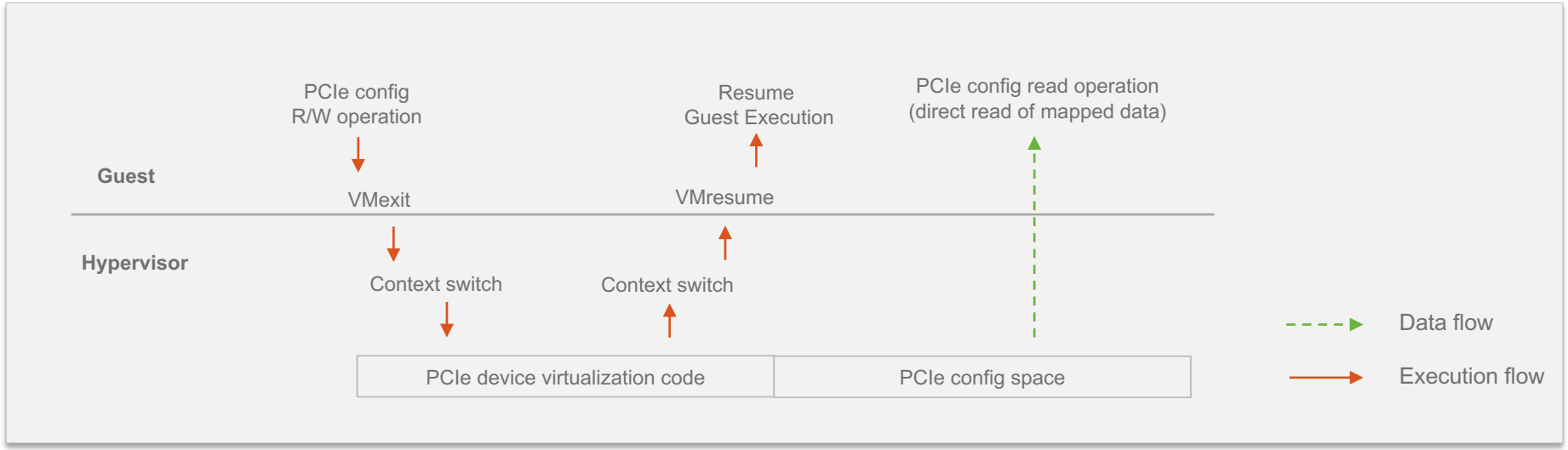
- Each PCIe config R/W operation takes ~10us
- Guest executes VM exit/resume commands to communicate with virtual devices in the host
- CRX with average configuration performs ~3250 PCIe config Read/Write during boot.



Execution flow of PCIe config R/W operation

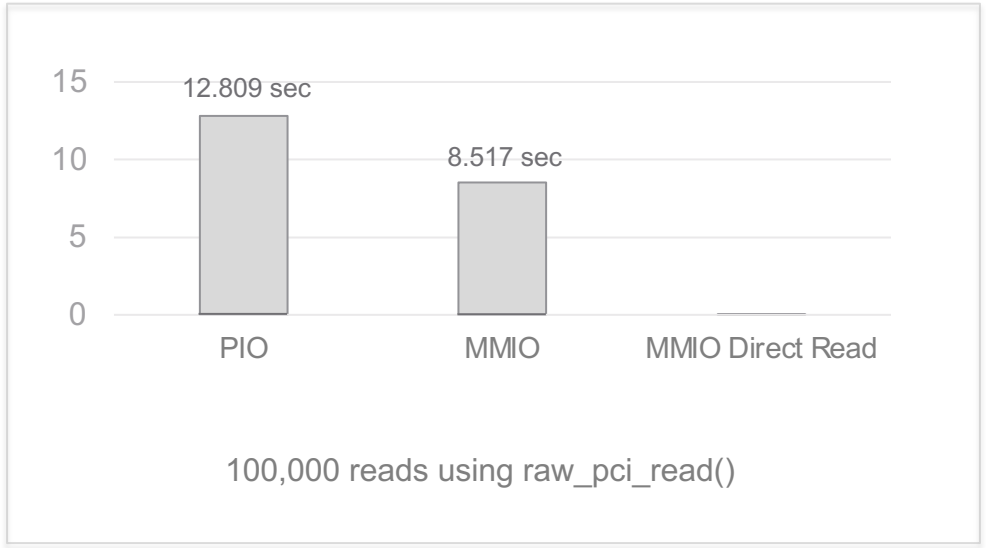
Solution: MMIO Direct Read

- Map virtual device PCIe mmconfig structure to MMIO region of the Guest
- The memory region is mapped as "read-only”
- Writes would still be trapped and handle by hypervisor
- No need to map complete 256MB of Mmconfig, only necessary page(s) per device/bridge will be mapped
- Side effect: If any action requires it hypervisor end while reading that will be skipped.



PIO, MMIO vs MMIO Direct Read

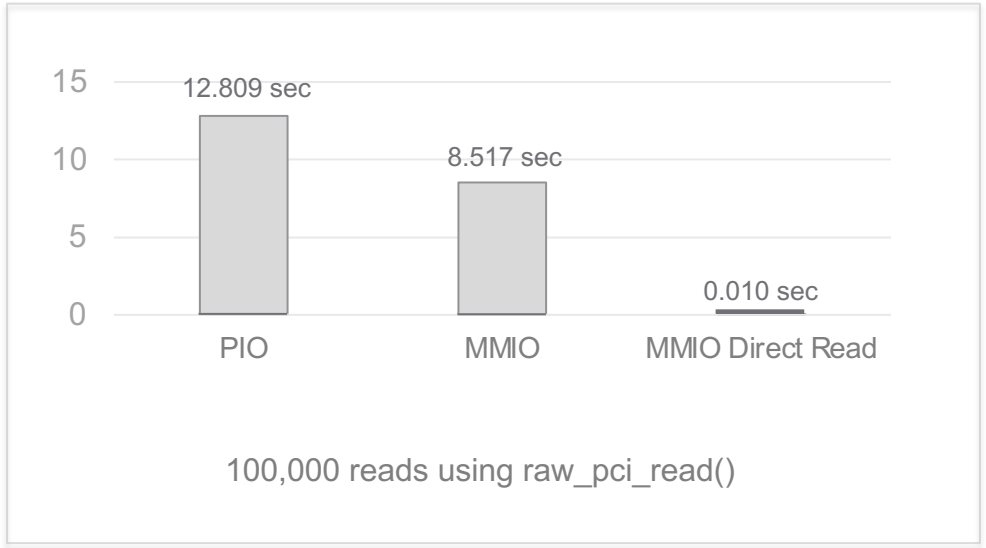
Improvement: MMIO Direct Read



- These readings are from Linux Kernel v5.10, on VMware hypervisor.

- This helps to reduce virtual machine PCI scan and initialization time by ~65% (52ms to 19ms)

Improvement: MMIO Direct Read

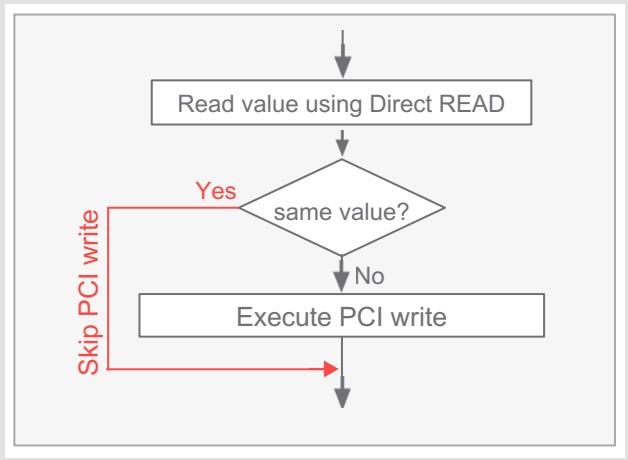


- These readings are from Linux Kernel v5.10, on VMware hypervisor.

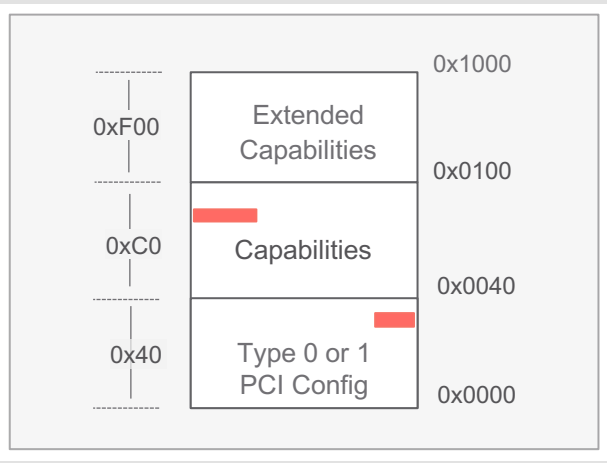
- This helps to reduce virtual machine PCI scan and initialization time by ~65% (52ms to 19ms)

Solution: Skip write, Pre-initialized (cont.)

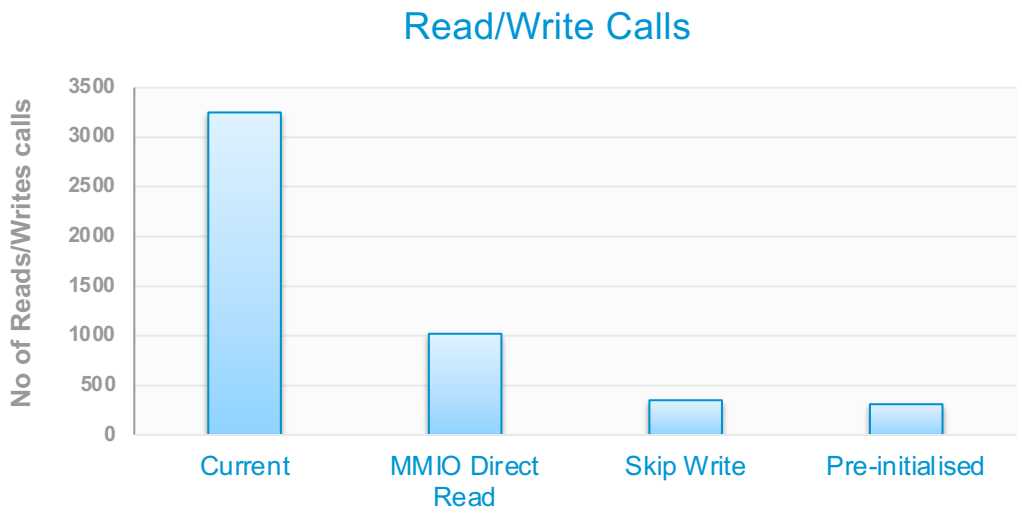
Skip Write: 12% Improvement

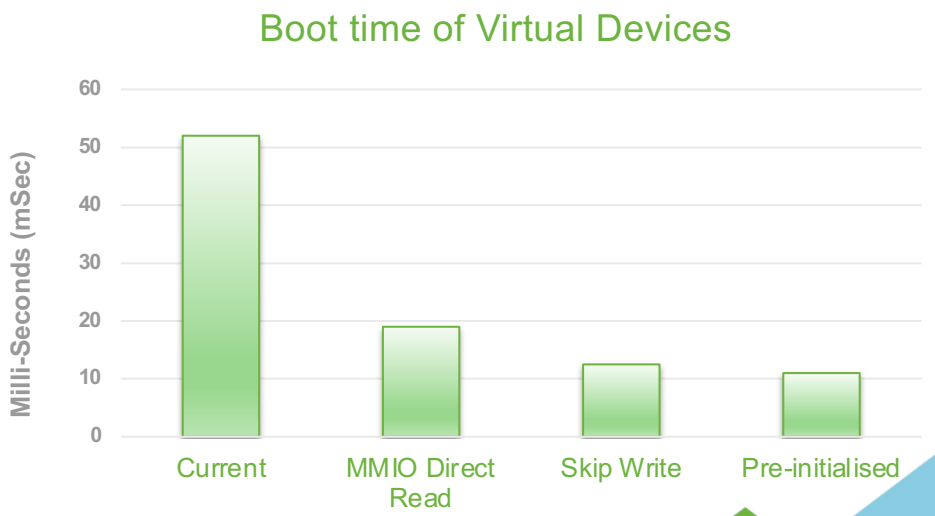


Pre-initialized: 2% Improvement and more possible

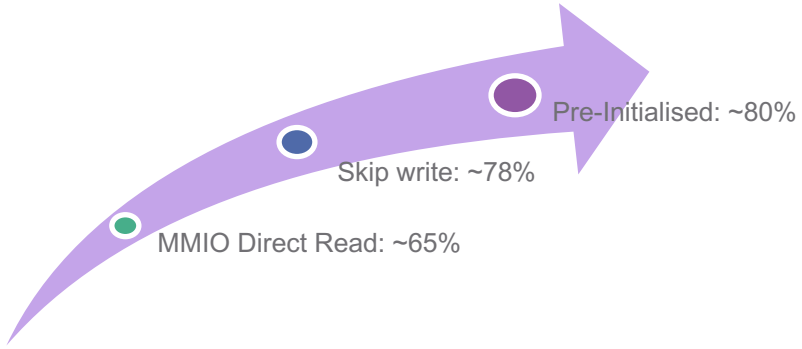


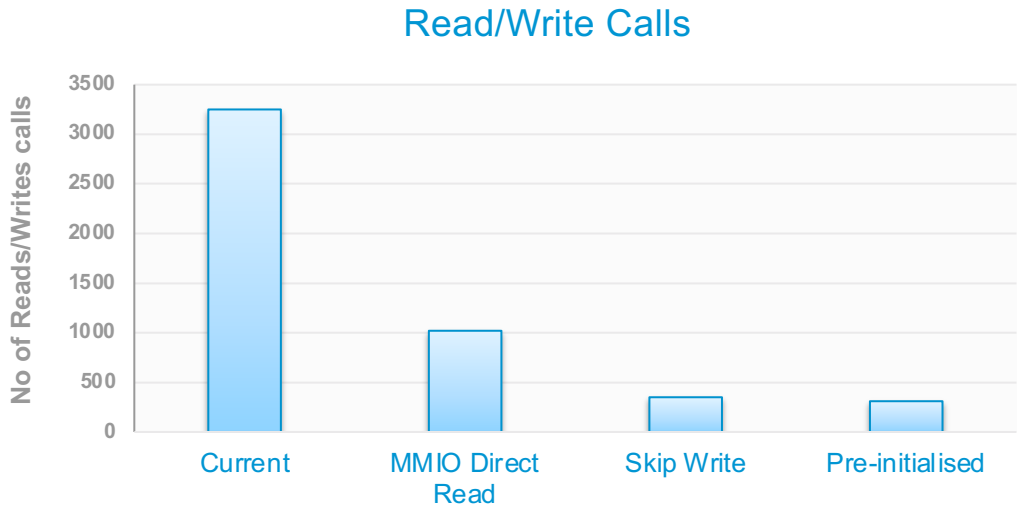
Improvement: Boot time of Virtual Devices

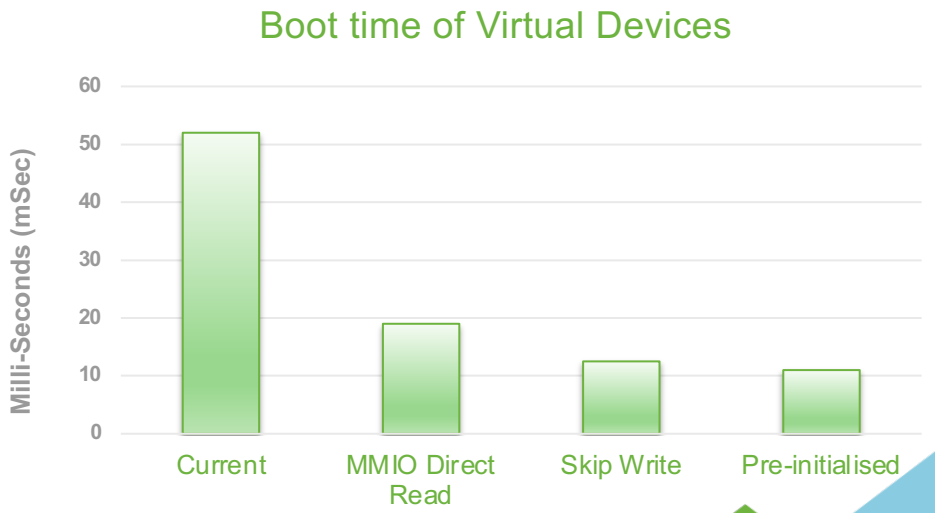




Improvement: Boot time of Virtual Devices







Following up work:

- Following patch in discussion:
  [PATCH v2] x86/PCI: Prefer MMIO over PIO on hypervisor

- KVM community also looking into KVM, QEMU to implement ‘PCIe MMIO Direct READ’:
- [PATCH v2 0/3] KVM: x86: KVM_MEM_PCI_HOLE memory

Following up work:

- Following patch in discussion:
  [PATCH v2] x86/PCI: Prefer MMIO over PIO on hypervisor

- KVM community also looking into KVM, QEMU to implement ‘PCIe MMIO Direct READ’:
- [PATCH v2 0/3] KVM: x86: KVM_MEM_PCI_HOLE memory

Looking for suggestion/feedback on:

- Are we creating a security loophole?

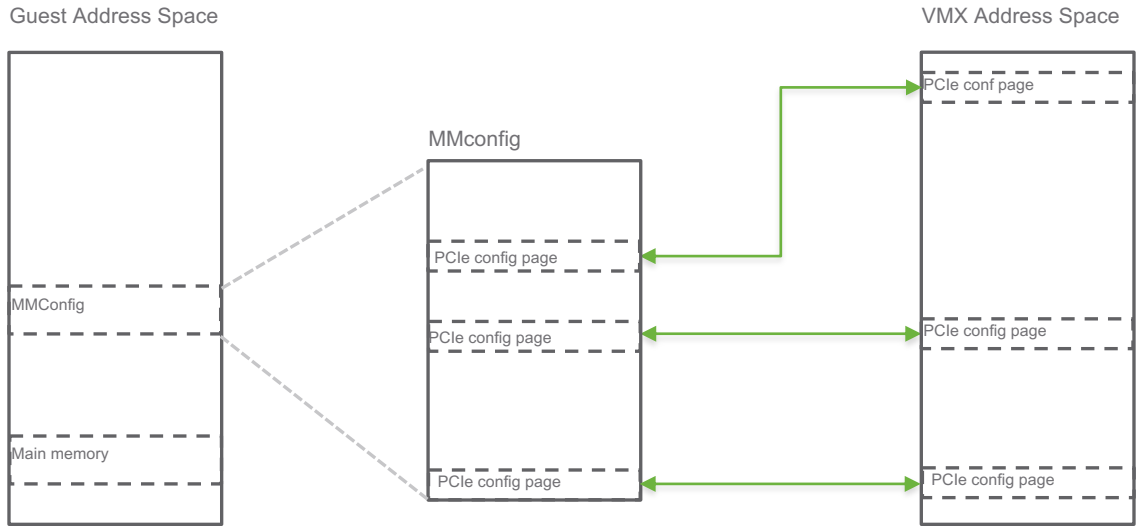
- If there is a mismatch in page size of the Host and the PCIe config page size, the solution does not work as intended.

  For example: if the Host is configured with a page size of 64 KB, and given that Guest PCIe config pages are 4 KB, it leads to inefficient use of Host memory or overlapping.

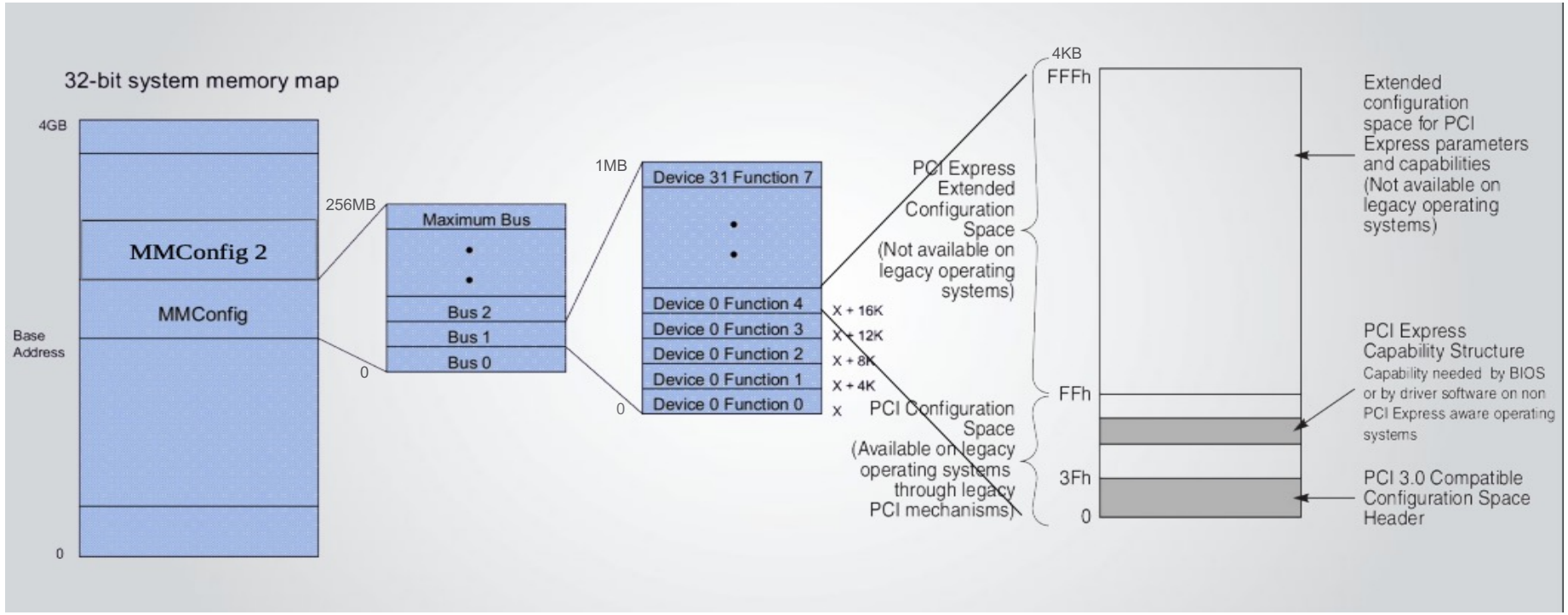
Thanks

Solution: MMIO Direct Read (cont.)

- No need to map complete 256MB of MMconfig
- Only necessary page(s) per device/bridge will be mapped



PCIe configuration space



PCIe config space per Function = 4 KB
PCIe config space per Device = 8*4 KB = 32 KB
PCIe config space per Bus = 32*32 KB = 1 MB
PCIe config space = 256*1 MB = 256 MB

Write technique