Preserving State for Fast Hypervisor Update
Pasha Tatashin, Google / Microsoft
Outline

- Fast Hypervisor Update Introduction
- Hot-patching and Live Migration vs. Fast Local Update
- Fast Local Update
- Recent Publications
- Fast Local Update: Building Blocks
- Fast Local Update: Demo Cloud Hypervisor
- Fast Local Update: Data for QEMU
- Discussion topics:
  - Emulated PMEM Interface
  - Preserving device state across reboot
  - Support for Virtual Functions
- Future: Multi-root update
Fast Hypervisor Update Introduction

- Cloud relies on virtualization
- Hypervisors must be provisioned with minimal disruption for the tenants
- Shutting down VMs is not an option, therefore a fast traditional reboot is not a solution
Hot-patching and Live Migration vs. Fast Local Update

- Hot Patching - Excellent for quick security fixes, not an update solution.
  - Extra development expenses
  - Maintenance matrix complexity
  - Limited inline functions and macros support
  - Limited data structures modification
  - Not a software update
  - Slowdown
  - Extra testing
  - Kernel rejuvenation

- Live Migration - Excellent for hardware provisioning, and load balancing.
  - Additional network traffic
  - Keeping device state challenges
  - Virtual machine utilization dependence
  - Extra hardware availability
Fast Local Update

- Being able to update all parts of virtualization stack without terminating virtual machines locally on the host
- The software parts may include: VMM, Driver Modules, Kernel, and Firmware
Recent Publications

Fast Local Update: Building Blocks

- **Kexec**
  - Fast reboot
  - Skip firmware reset
  - Enables memory state preservation

- **Local Live Migration**
  - Migration to a file, or snapshotting
  - Preserves the VM state across reboot, so hypervisor can be rebooted

- **Emulated PMEM + DAX + EXT4**
  - Allows using VM memory as a file on a RAM based filesystem without pagecache
  - Enables skipping memory copy during local live migration
  - Enables preserving VM state across reboot
Fast Hypervisor Update: Demo Cloud Hypervisor

- Link to the demo
  https://youtu.be/fRLYCeyQLcI
Fast Hypervisor Update: Data for QEMU

<table>
<thead>
<tr>
<th>Machine</th>
<th>VM Suspend</th>
<th>VM Save</th>
<th>QEMU Shutdown</th>
<th>Userland Shutdown</th>
<th>Kernel Shutdown</th>
<th>Kernel Startup</th>
<th>Userland Startup</th>
<th>QEMU Startup</th>
<th>VM Restore</th>
<th>VM Resume</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD 3950x</td>
<td>0.0021</td>
<td>0.282</td>
<td>0.042</td>
<td>0.1051</td>
<td>0.5405</td>
<td>1.3324</td>
<td>0.1166</td>
<td>0.1368</td>
<td>0.0021</td>
<td>0.0157</td>
<td>2.5753</td>
</tr>
<tr>
<td>Xeon 8272CL</td>
<td>0.0297</td>
<td>0.3537</td>
<td>0.0656</td>
<td>0.2857</td>
<td>1.1296</td>
<td>2.9323</td>
<td>0.3571</td>
<td>0.2898</td>
<td>0.0045</td>
<td>0.0309</td>
<td>5.4789</td>
</tr>
</tbody>
</table>

---

**Diagram:**

- **AM 3950x:**
  - VM Suspend: Pale gray
  - VM Save: Medium gray
  - QEMU Shutdown: Light gray
  - Userland Shutdown: Very light gray
  - Kernel Shutdown: Black
  - Kernel Startup: Medium gray
  - Userland Startup: Very light gray
  - QEMU Startup: Light gray
  - VM Restore: Dark gray
  - VM Resume: Dark gray

- **Xeon 8272CL:**
  - VM Suspend: Pale gray
  - VM Save: Medium gray
  - QEMU Shutdown: Light gray
  - Userland Shutdown: Very light gray
  - Kernel Shutdown: Black
  - Kernel Startup: Medium gray
  - Userland Startup: Very light gray
  - QEMU Startup: Light gray
  - VM Restore: Dark gray
  - VM Resume: Dark gray
Discussion topics: Emulated PMEM Interface

- Emulated PMEM can be created in three ways
  - UEFI marks part of regular memory as *Type 7*
  - Add `pmem` node under root node in the device tree
  - Kernel parameter `memmap=nn[KMG]!ss[KMG]`

- The admin must know the intimate knowledge about the physical memory layout.

- Not portable

- “Legacy” PMEM?
For Intel IOMMU based device the preserved state includes saving the following data across reboot:

- VMX posted interrupts **tables** and soft data structures
- IOMMU **page tables** and soft data structures
- Interrupt Remappings soft data structures
- PCI soft data structures
- VFIO PCI soft data structures

The soft data structures are relatively small in size and can be portably passed from one kernel to another, but interrupt tables, and page tables which are configured for the devices must stay in place.

Can we use the same memory where the rest of VM memory is preserved to create those data structures?
Discussion topics: Support for Virtual Functions

VM1  VM2  I/O Domain  Driver for Network Card  Fake VF1  Fake VF2

Hypervisor  

VF 1  VF 2  Network Card

PCI BUS

Host System
Future: Multi-root update

- Dual boot can be achieved with a modified kexec call that starts new kernel on offline CPUs and with specifically configured memmap parameter without shutting down the current kernel.

- Pass the VM handling from one hypervisor to another on the fly by using local live migration, and PMEM.

- Pass the passthrough device states by passing the preserved soft data structures, and also the device state that was also allocated on PMEM with VMs.
Thank you