KVM Enhancements for OPNFV

Jun Nakajima

Intel Corporation

Contributors: https://wiki.opnfv.org/nfv-kvm

LINUX FOUNDATION
COLLABORATIVE PROJECTS

Project: NFV Hypervisors-KVM

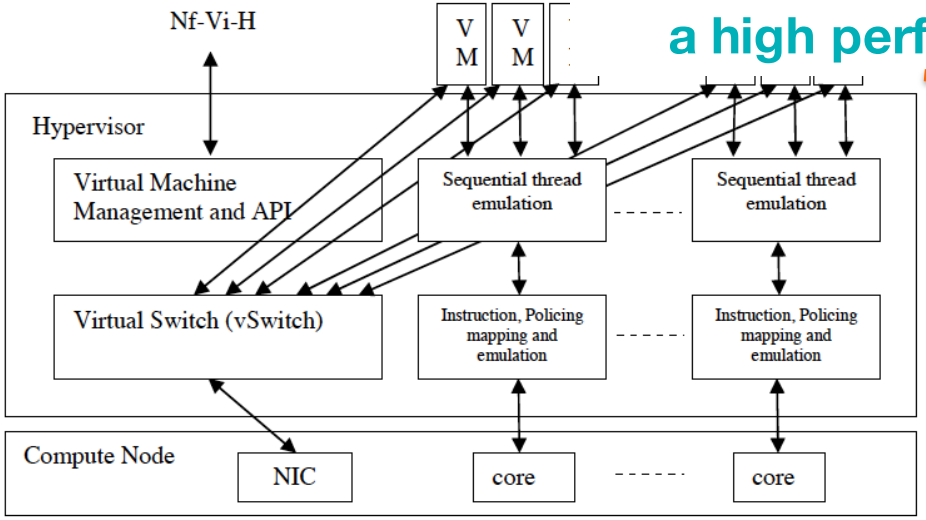
https://wiki.opnfv.org/nfv-kvm

1. Minimal Interrupt latency variation for data plane VNFs
2. Inter-VM Communication
3. Fast Live Migration

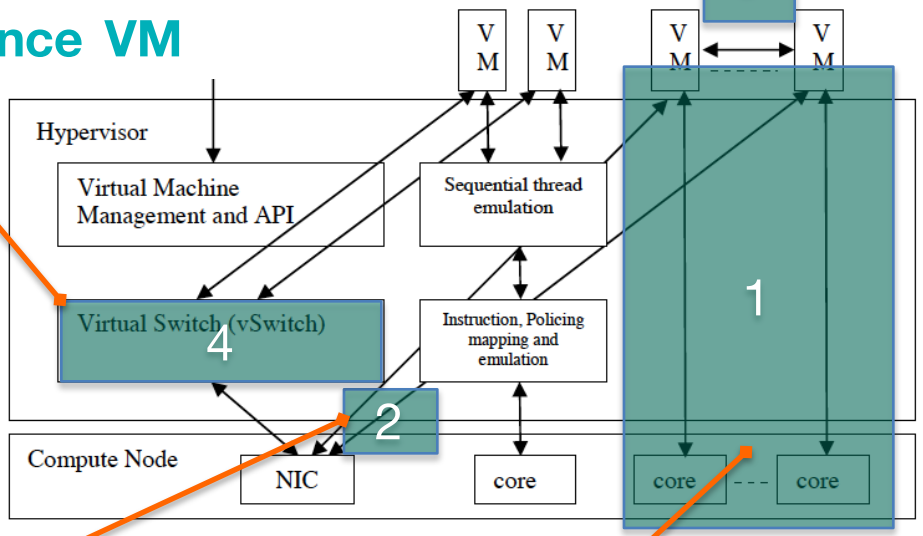
Enhancements for NFV Hypervisor

3. **Inter-VM Communication (direct-memory mapped)**

4. **vSwitch implementation as a high performance VM**



General public and enterprise cloud Hypervisor Architecture



NFV Hypervisor Architecture

2. **Direct I/O (e.g. SR-IOV)**

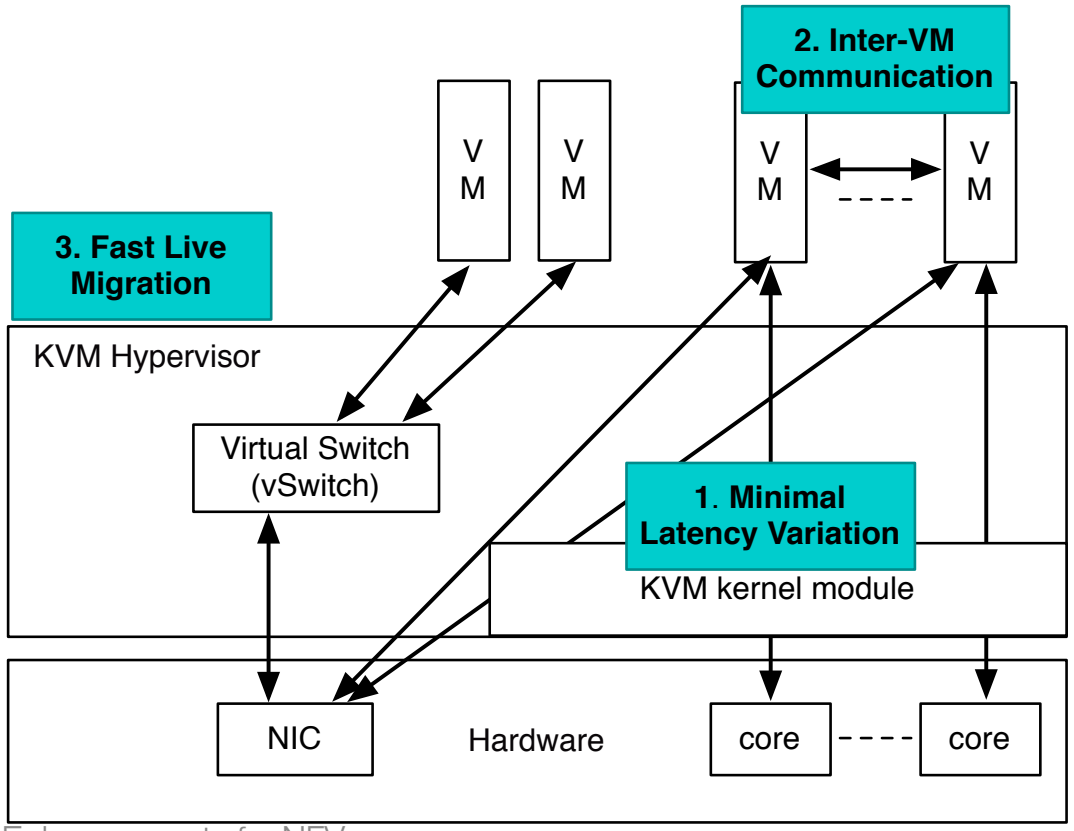
1. **Exclusive allocation of whole CPU cores to VMs**

From ETSI
"Network Functions Virtualization (NFV); Infrastructure; Hypervisor Domain"

Scope of the Project

System Configuration

- Inter-VM Config. Information
- Live Migration Config. Information
- Configuration Information (BIOS, boot, kernel)



Testing and Perf Tools

- **Inter-VM Comm. Perf. Tools**
- **Live Migration Perf. Tools**
- **Latency Variation Perf. Tools**

Minimal Interrupt Latency Variation

Goals of Minimal Interrupt Latency Variation

- Timer latency => AVG 5 us, MAX 15 us
- Interrupt latency => MAX 20 us (Future target: 5 us)
- Maximum guest vCPU pre-emption period => 10 us (Future target: 2 us)
- Cumulative pre-emption within 1 ms window => X < 20 us (Future target: X < 4 us)

https://etherpad.opnfv.org/p/nfv-kvm

Summary Of Solutions

Excusive/Static Allocation

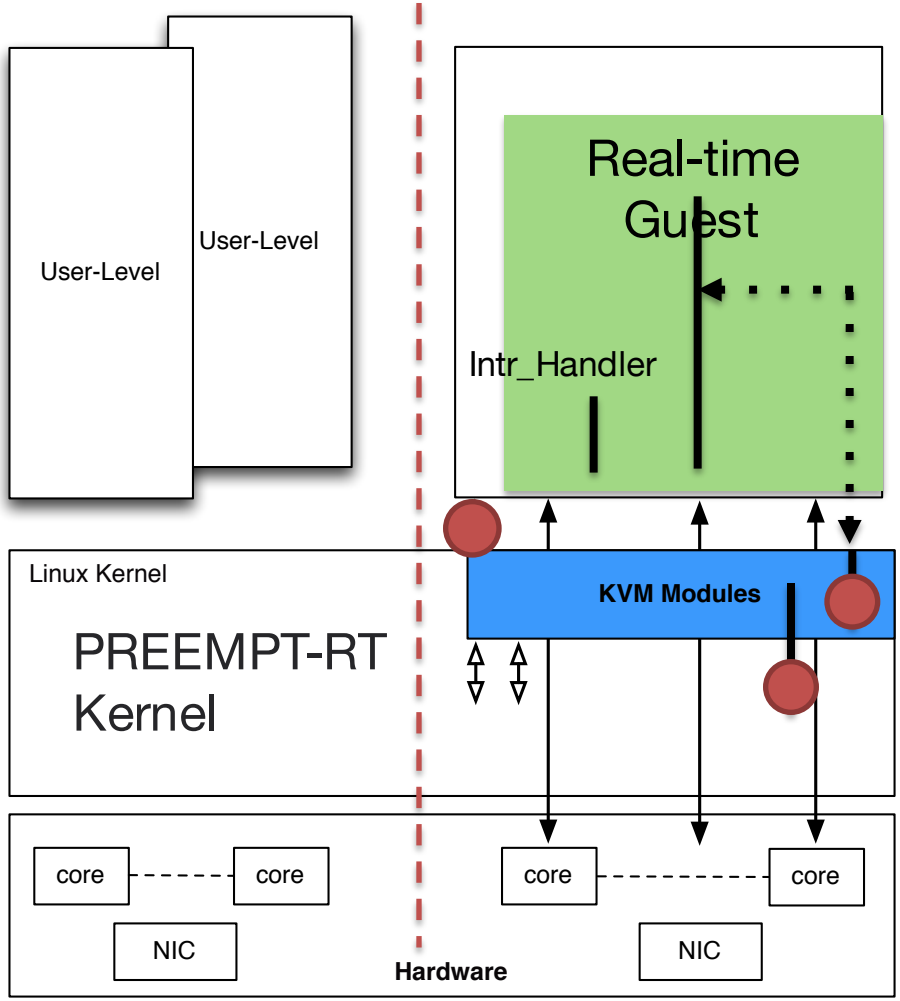
Soft “Partitioning”, CPU Binding, Huge Pages

Software

Real-Time Linux, Code inspection, testing/measurements

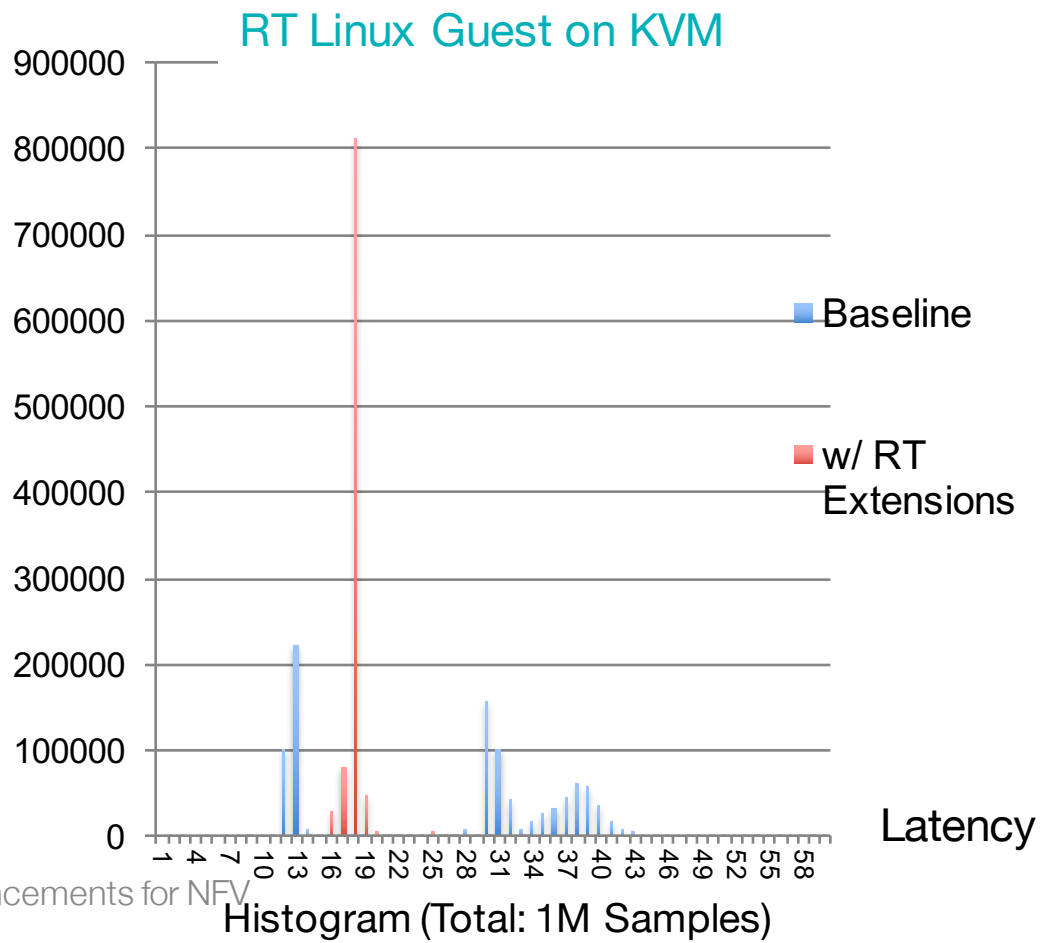
Hardware Technologies

Cache Allocation Technology, Advanced VT features



Update on Enhancements:

Cyclictest (Initial)



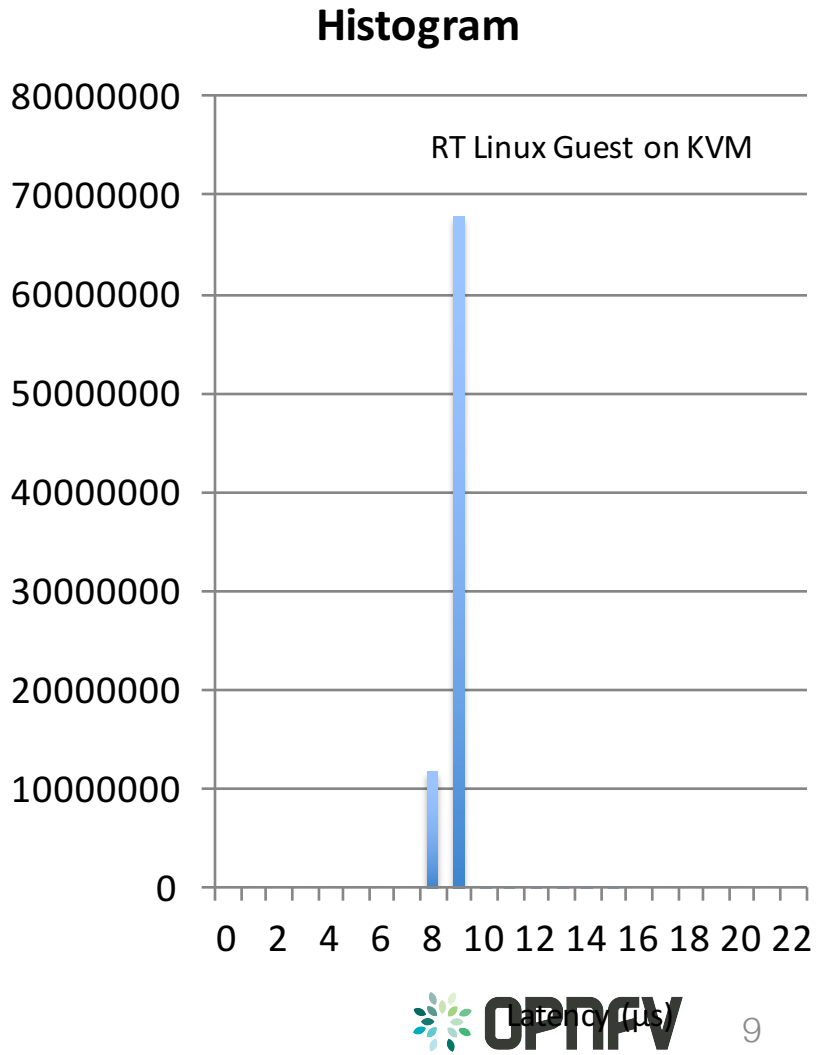
Update on Enhancements:
Cyclictest (After)

- Cyclic Test in Guest: Latency (in µs)
  - Min: 7
  - Avg: 9
  - Max: 16

Latency	Occurrences
000007	000003
000008	11757562
000009	67812652
000010	159222
000011	069100
000012	011004
000013	000379
000014	000207
000015	000049
000016	000005

99.69% (Total #: 79,810,183)

Host: Linux with RT patches



Update on Enhancements:
Cyclictest (Latest)

- Cyclic Test in Guest: Latency (in µs)
  - Min: 6 ~~7~~
  - Avg: 6 ~~9~~
  - Max: 11 (9*) ~~16~~
- Host
  - 4.1.0 + PREEMPT-RT (upstream)

*: w/ 1Hz tick disabled

Update on Enhancements:

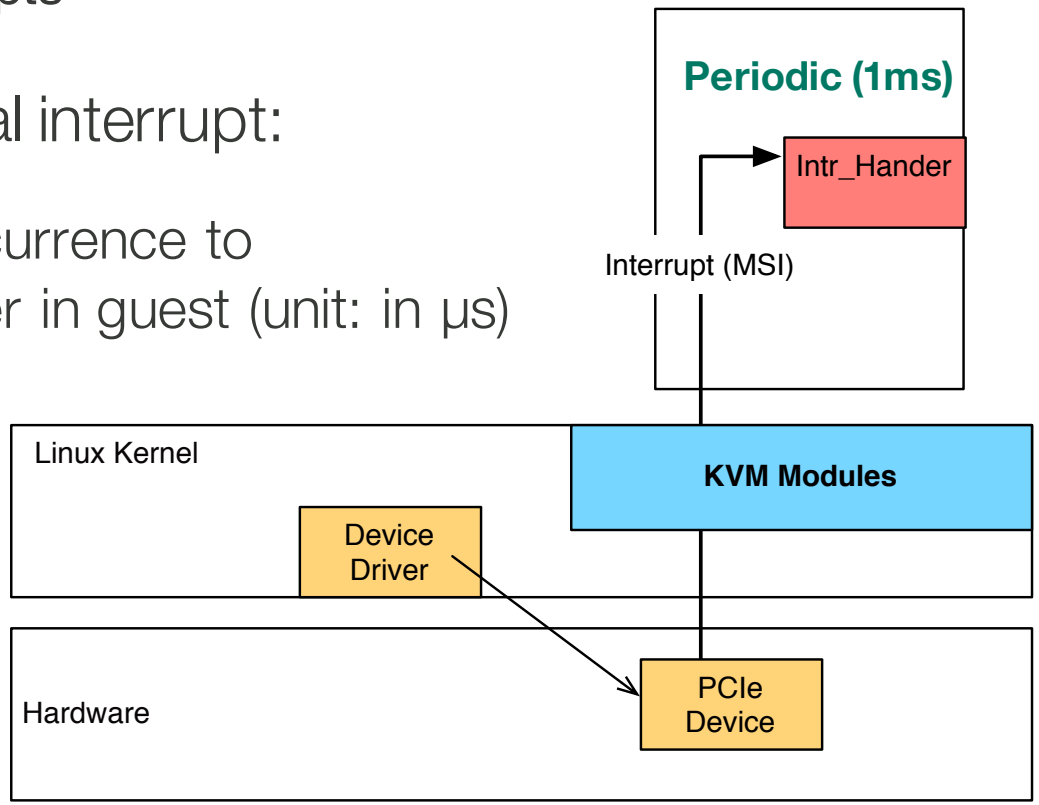
Latency from Periodic External Interrupts

- Latency from periodic external interrupt:
  - Time delta from interrupt occurrence to invocation of interrupt handler in guest (unit: in µs)

Min: 3.98

Avg: 4.42

Max: 9.10

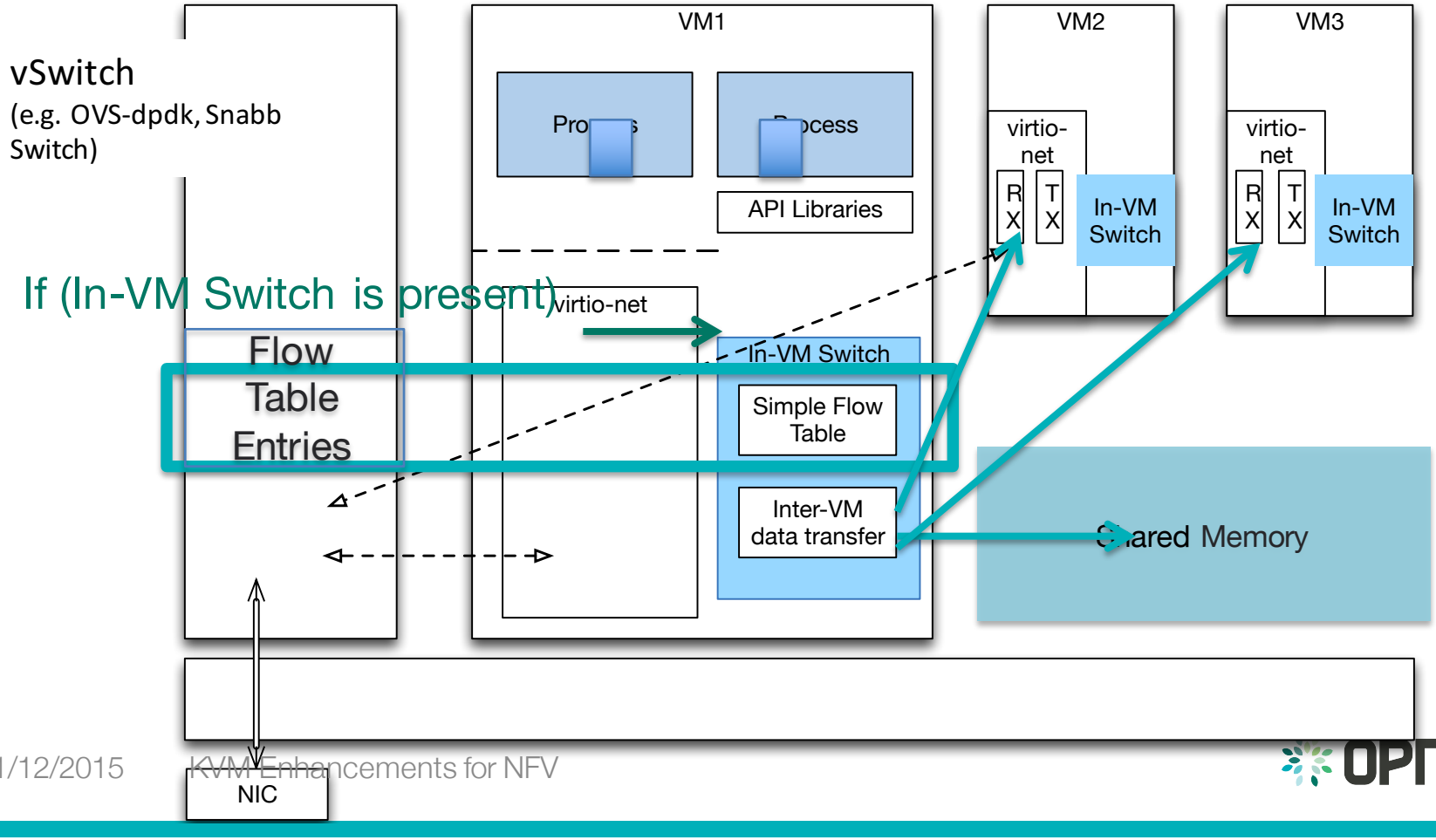


Inter-VM Communication

Goals of Inter-VM Communication

- Add fast-paths in VMs as optimized inter-VM communication
- Maintain consistent flow table entries in VMs
- Enable protected access to the destination VM or shared memory
  - Open the Window when needed
  - Close it immediately when done

Summary Of Solutions

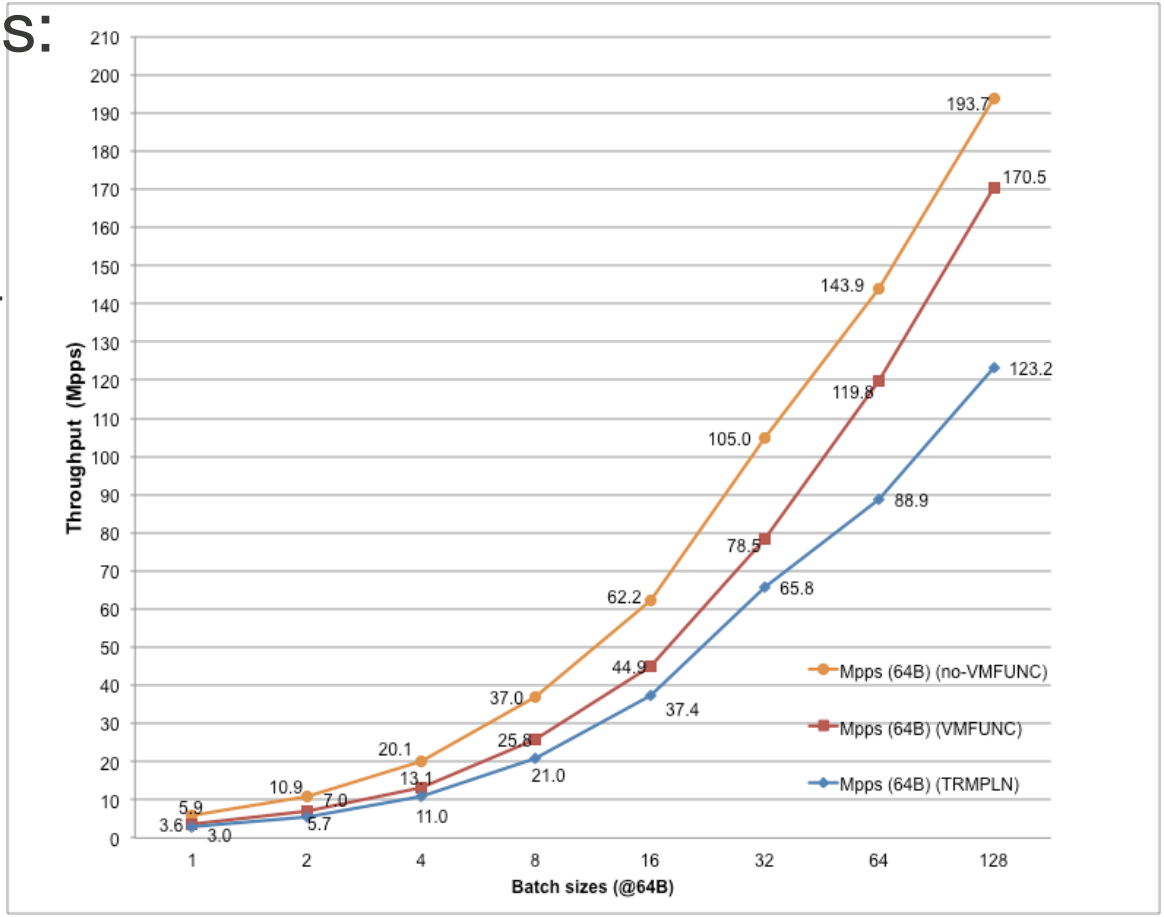


Update on Enhancements:
Upstream Project

- Proposal from Michael S. Tsirkin https://wiki.opnfv.org/vm2vm_mst
  - Inter-VM Communication for two VMs
  - Receiver allows one to access its own memory via existing IOMMU protection mechanism
- Our plan
  - Extend Michael's proposal to support multiple VMs
  - Work with him for implementation

Update on Enhancements:
PoC

- Transfer 64B packets from virtio-net to another VM (fast path)
  - **Memory Copy**
  - **VMFUNC**
  - **VMFUNC with Trampoline Code**
    - **65Mpps with 32-packet batching*:**



*Intel internal estimation

Fast Live Migration

Goals of Fast Live Migration

- Migration Time
  - < 2 sec.
- Down Time
  - Hypervisor downtime for IO intensive workload: 10 ms (future 3ms)
  - Hypervisor downtime for memory intensive workload: < 500 ms
  - Network downtime: 25 ms
- Configuration
  - Typical VM size: 8GB to 16GB
  - Network throughput: 5Gbps for 64bytes packets

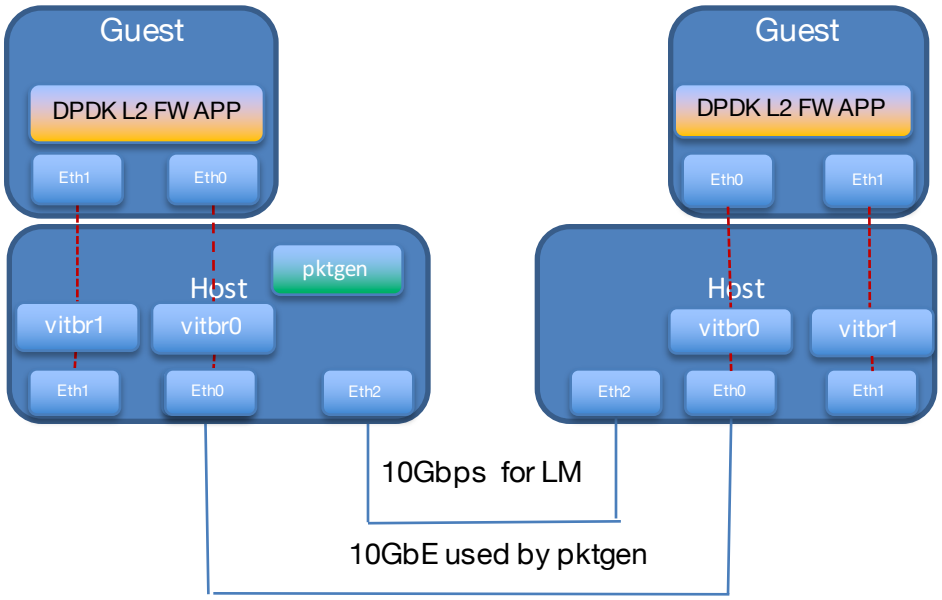
- SR-IOV Migration
  - Support NIC with kernel Land driver
  - Support NIC with DPDK driver

Summary Of Solutions

- Shorten VM Downtime
  - Clean up operation after completion of data transmission
  - Minimize cpu_synchronize_all_states() calls
- Shorten Total Migration Time
  - Optimize handling of unused or zero pages
  - Use CPU instructions to optimize zero pages checking
  - Use hardware accelerator to compress data before transmitting

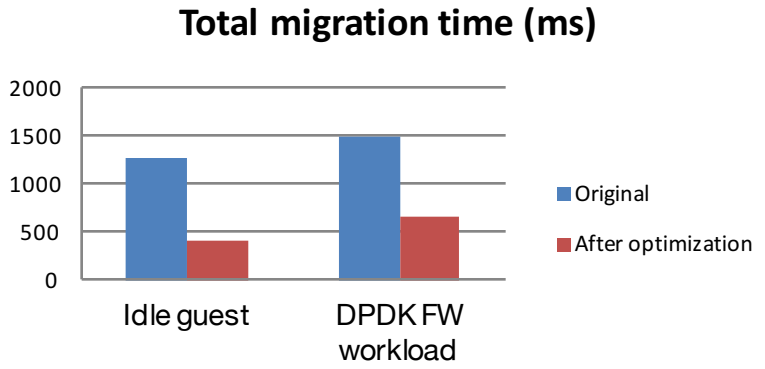
Update on Enhancements

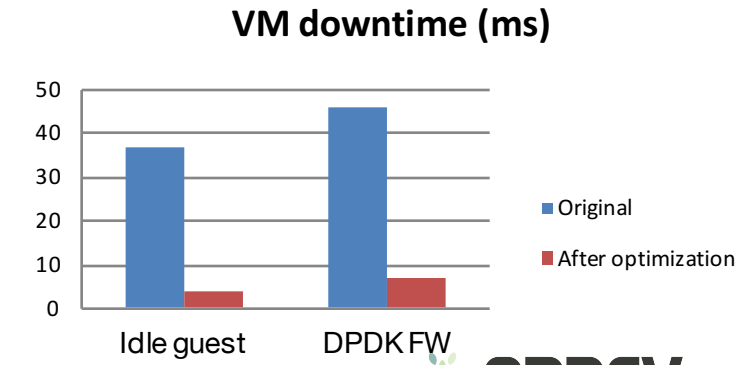
Test environment



Host CPU: Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz RAM: 64G
OS: RHEL 7.1, Kernel: 4.2-rc6 QEMU : 2.4.0
Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+

Performance data





Q & A

Join us!