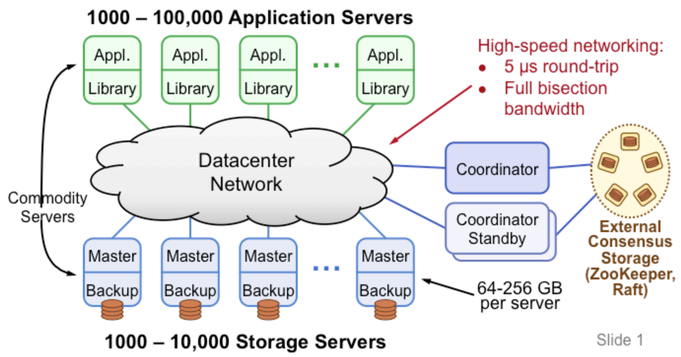
RAMCloud Latency over SRIOV/KVM on Xeon DPDK

Satoshi Matsushita[1,2], Yoshihiro Hasebe[2], Kazuo Kuwata[3]
1) Stanford University, 2) NEC Corporation 3) NEC Solution Innovators, Ltd.

2015 SEDCL Retreat Poster

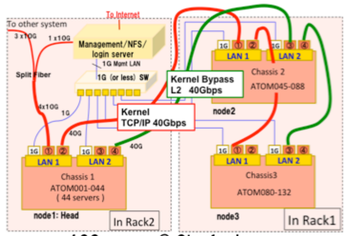
RAMCloud Architecture



Slide 1

RAMCloud in a Box - Atom CPU

- **NEC Microserver - 16 chassis in a 19 inch rack**
- **2U Chassis consists of:**
  - Up to 46 Modular Servers: 1.7GHz/2.4GHz Atom C27x0, 32GB DRAM, 128GB SSD
    - Chassis total: 1.4TB DRAM, 5.8TB SSD
  - 2x 230Gbps switch (FM5224), 4x 40Gbps uplinks
  - Redundancy (Power supply, Networks, Fans) - Hot Swappable



132 server @ Stanford
- No expensive 40G/10G switches

Slide 2

Atom or Xeon

1) **Atom:Embedded NIC** or **Xeon:CPU speed**
2) **Features**
   - **Atom** and Xeon: **Intel VT-c** "virtual NICs": SRIOV, VMDq
   - **Xeon**: **Intel VT-d** "Direct DMA" : reduce VM overhead

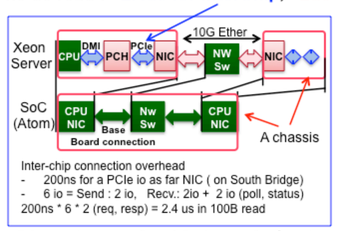
Result: RAMCloud 100B Read/Write Latency & Ratio → **Details in 3)**

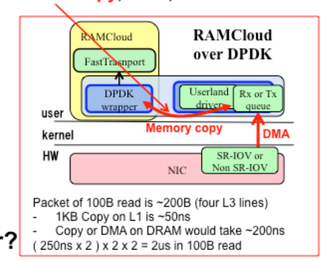
System	Transport	Features	Latency (us)		
			Rd	Wr	W/R
Stanford Xeon	40Gb Infiniband	Verb, RDMA	4.7	14.7	3.1
Solarflare Xeon	10Gb Ether+Fast	Kernel bypass, SRIOV	9.5	23.0	2.4
RAMCloud in a Box (Atom)	2.5Gb Ethernet+ FastTransport	Intel DPDK, Kernel bypass	11.6	34.7	3.0
Xeon + Intel 10G	10Gb Ethernet + FastTransport	**Bare-metal**, Kernel bypass	14.5	33.2	2.3
		SRIOV (VT-c)	14.6	33.6	2.3
		KVM Guest OS (VT-d)	20.3	48.6	2.4

1) Why Xeon+10G is Slower than Atom?

- **(Wr / Rd ~= 2) implies that NW overhead dominates because Wr's hops are twice of Rd's**
- **Why Xeon's Network is slow?**

14.5 us read : ~2.4us for InterChip, ~2us for Mem copy, NIC , Network-Wire



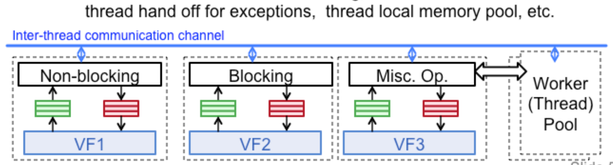


- **How to make Network faster?**
  1. SoC: XeonD
  2. Hardware solution: Verbs/RDMA/QP, RoCE
  3. Clean slate NIC architecture

Slide 4

2) A Use Case of SRIOV

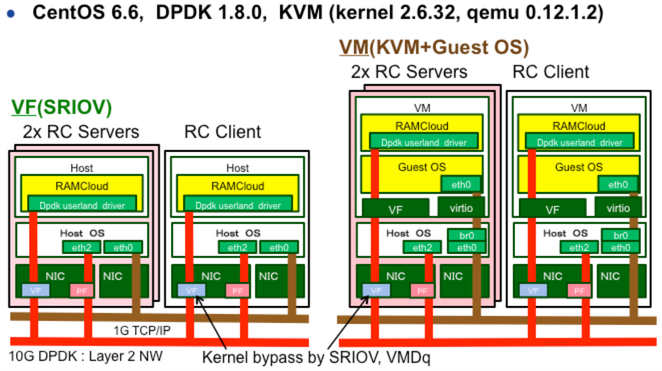
- **No latency penalty with SRIOV (Virtual NIC)**
- **Thread is expensive**
  - Latency for thread handoffs: 100ns in each direction
  - Total cost of threading: ~450 ns in latency (in RcCluster / ATOM)
  - But threading needed in Master for concurrent operation of Client request, Backup request for client write or recovery
- **Use VFs for different character of requests:**
  - VF1: Non-blocking: Read …. Save 450ns?
  - VF2: Blocking but serialization needed: Write, Delete …. Save 450 ns?
  - VF3: Misc. (server request): Backup, etc. – thread hand-off if blocks .. None
  - Consideration needed: Resource locking, thread hand off for exceptions, thread local memory pool, etc.



Slide 5

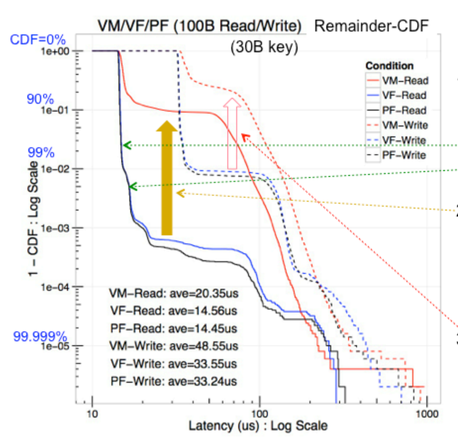
3) Xeon+10G: Evaluation Environment

- **Xeon E5504 (Nehalem-EP, 45nm) 2GHz, 4 core/4 thread**
  L1: 32KB/32KB, L2: 256KB, L3: 4MB (Shared)
- **CentOS 6.6, DPDK 1.8.0, KVM (kernel 2.6.32, qemu 0.12.1.2)**



3) Latency Distribution of 100B R/W

Xeon + 10G DPDK

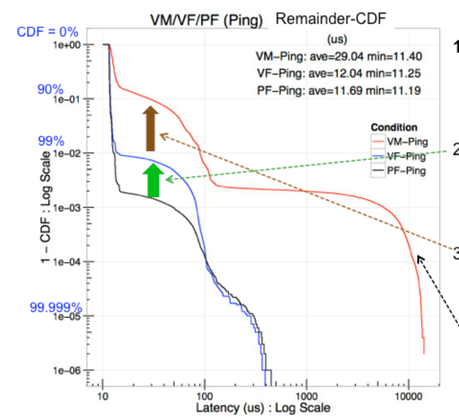


1. Almost no overhead with **SRIOV (VF)** against **Bare-metal (PF)**:
   99% in 14 us
   99.9% in 16 us
2. Slow fractions increases with **KVM (VM)**
   80% in 14 us
   95% in 16 us
   99% in 60 us
3. Similar shift in writes (1 replica, 2 servers)

Slide 7

3) Latency Distribution of Ping

Xeon + 10G DPDK



1. **Bare-metal (PF)**
   99% in 10.5 us
   99.8% in 10.8 us
   99.9% in 50 us
2. Small distribution shifts with **SRIOV (VF)**
   99% in 10.8 us
   99.5% in 50 us
3. Large shift with **KVM (VM)** + dull cliff
   90% in 11 us
   99% in 90 us

   99.9% in 5 ms (Context switch..)

Slide 8

3) Consideration on Latency Distribution

- **Cliffs**
  - {2.5us, 50-80us, 250us} slower than minimum in Ping/100B Rd. Similar to RcCluster Infiniband.
    - 2.5 us : L3 cache miss + CPU internal / Memory conflict?
    - 50-80 us : NIC or network?
    - 250 us : OS related?
  - 5 ms slower than minimum in VM ping – context switch, process in run queue with nanosleep()
- **SRIOV Overhead**
  - Distribution shift derived from NIC/Network
- **KVM Overhead**
  - Why all cliffs shift in the distribution?

Slide 9