We Need Volunteer Spirit! for Speeding Up Consolidated DPDK Applications

Yuki Tsujimoto, Kenta Ishiguro, Kenichi Yasukata, Kenji Kono
Keio University

In a cloud computing context, oversubscribing hardware resources by consolidating multiple virtual machines (VMs) on a single physical server is a promising approach to improve hardware utilization. However, we observed that network-intensive workloads, one of the most important workloads in cloud environments, do not always benefit from CPU oversubscription.

Problem. The two traditional mechanisms, *interrupt* and *polling*, are used to invoke the execution of networked applications. Both mechanisms do not fit well with CPU oversubscription because networked applications on the consolidated VMs suffer from additional virtualization overhead or low resource utilization when using these mechanisms.

Using hardware *interrupts* triggered by a NIC at a packet reception is the most widely applied mechanism to invoke the execution of networked applications. Fig. 1 shows how two networked applications with each mechanism handle five packets when they are consolidated on the same CPU. In Fig. 1, the interrupt-based mechanism suffers from frequent context switches from hardware interrupts at each packet arriving. This issue gets worse when receiving packets at a high rate because frequent context switches caused by interrupts consume many CPU cycles.

A *polling*-based mechanism is applied in some systems like DPDK to mitigate the overhead highlighted by the interrupt-based mechanism. This mechanism turns off interrupts and monitors NIC registers in the context of networked applications. While this mechanism eliminates the performance overhead caused by handling interrupts, it lowers effective resource utilization by wasting CPU cycles. For example, as shown in Fig. 1, vCPU 0 wastes CPU cycles by monitoring NIC registers after processing three packets, even though vCPU 1 is waiting to process a packet. Consequently, the overall performance of consolidated networked applications with the polling-based mechanism is substantially limited.

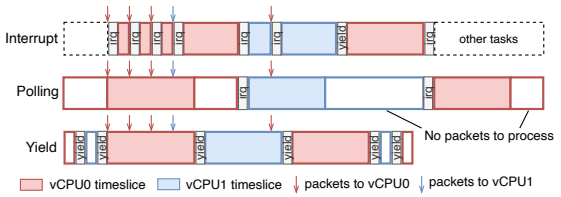

Figure 1. Timeline of two networked applications on a single physical CPU handling packets. The uncolored boxes depict the CPU cycles that do not contribute to processing packets.

Approach. To mitigate the resource inefficiency, we propose a voluntary yield mechanism by modifying DPDK applications. In our approach, DPDK applications use the traditional polling-based approach, but they yield up their CPU cycles with a yield hypercall once they observe an empty NIC queue, even if they do not use up their allocated CPU cycles. For example, as shown in Fig. 1, vCPU 0 processes three incoming packets and then yields up its CPU cycles before exhausting the remaining timeslice. By doing this, the voluntary yield approach can eliminate meaningless monitoring of DPDK applications and improve resource efficiency.

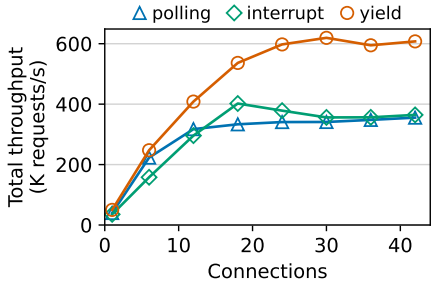

Figure 2. Aggregated throughput of two DPDK-applied HTTP servers running on a single physical CPU.

Experiments. Fig. 2 shows the aggregated throughput of two DPDK-applied HTTP servers with different numbers of TCP connections[1][2][3]. Each HTTP server runs inside a 1-vCPU VM[4] on Xen 4.12 and those two VMs are consolidated on a single physical CPU.

The proposed trick exhibits the best performance among the three schemes. These results demonstrate that 1) frequent interrupts invoked by the network interfaces diminish the performance of networked applications running on VMs, 2) the polling mode, where VMs do not voluntarily yield the CPU time, negates the cumulative performance because the CPU cycles are wasted just for monitoring the NIC registers, and 3) the proposed trick, disabling interrupts while yielding CPU cycles accordingly, complements the limitations of the existing approaches, consequently, achieves high performance.

[1]We conducted the experiment on one server machine and one client machine. Both machines contain a 3.80 GHz 6-core Intel Xeon E-2276G processor with 32 GB of RAM and a 10 Gb SR-IOV capable Intel X540 NIC.
[2]The interrupt mode uses the DPDK's epoll facility.
[3]The client runs wrk2 to generate HTTP requests.
[4]We dedicated a virtual function of the SR-IOV NIC to each VM.