DPDK: Accelerate Remote Rendering of Cloud Gaming

Jingjing Wu & Owen Zhang - Intel
DPDK Summit - China - 2019
Agenda

- Cloud Gaming Background
- Data Path for remote rendering
- Solution & work status
- Future work
Background - Cloud Gaming

- Streamed frames, files or commands from cloud/edge to device.
- $1B business in 2017, projected to grow at 26%

Google’s Project Stream is a working preview of the future of game streaming

Microsoft’s xCloud service streams Xbox games to PCs, consoles, and mobile devices

Here’s the evidence Amazon is building a cloud gaming service

1 Zion Market Research, “Cloud Gaming Market by Cloud Type (Public, Private, and Hybrid), by Streaming Type (Video and File), and by Device (Smart Phones, Tablets, Gaming Consoles, and PCs): Global Industry Perspective, Comprehensive Analysis, and Forecast, 2018—2026”
Intel® VCA2 (Visual Compute Accelerator)
Delivering the Visual Cloud. Faster.

- Powered by the Intel® Xeon Processor E3-1500 v5 with Intel® Iris Pro Graphics P580 and Intel® Quick Sync Video
- Outstanding TCO for media transcoding & rendering applications.
- Learn more: intel.com/accelerators

**Broadcast:** Ultra-high channel density, with high Visual Quality

**Virtual Reality:** Ultra-dense transcode enables truly immersive User Experiences

**Cloud Gaming:** Iris™ Pro graphics delivers richly rendered games, on any device, anywhere

**Multi-Party Communications:** B2B, C2C communications with massive scaling
Android Cloud Gaming Overview

Cloud Gaming Services deployed in Data Center or Edge Server

- Video Stream
- User Input

E5 Server + VCA2 (3x E3 SKL) or Future GPU card

Communication between game clients and servers

Game Server in Data Center

Operator: Easy to gain more users

End User: Easy to play new game

Developer: Easy to make better game

Operator: Easy to gain more users
Software Stack

Virtual machine
- Android In Container (AIC)
- Game App
- Android Framework
- Remote Render frontend

Stream server
- Input
- Video

Remote Render backend
- App
- UMD
- Mesa (GLX + OpenGL)

Linux Kernel
- VCA driver
- drm_drv
- Intel I915 KMD

Client Device
- Client App
  - Input
  - Media Player

E5 Server in DataCenter or Edge
- VCA driver

Visual Cloud Acceleration Card - VCA2
Characteristics of remote rendering data path

- Game frame from Server to Accelerator Card
- Video stream from Accelerator Card to Server
- Stream-based socket-like interface
- Isolate flow transaction between Server and Accelerator from data center networking
- Scale to support multi-VM
- Last but not least - Performance obsessed
Stream type socket w/o IP

<table>
<thead>
<tr>
<th>Socket Family</th>
<th>Device</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF_INET</td>
<td>PF passthrough</td>
<td>N/A</td>
</tr>
<tr>
<td>AF_INET</td>
<td>virtio_net</td>
<td>Yes</td>
</tr>
<tr>
<td>AF_VSOCK</td>
<td>virtio_vsock</td>
<td>No</td>
</tr>
</tbody>
</table>
## Scale for multiple VMs

### Socket Family | Device | IP | Multi-VM
---|---|---|---
AF_INET | PF passthrough | N/A | No
AF_INET | virtio_net | Yes | Yes (Switch/Router)
AF_VSOCK | virtio_vsock | No | Yes
Data Path solution

- AF_VSOCK
  - Classic sockets API
  - QEMU+KVM compatible (virtio-vsock device)
  - Bi-directional between hypervisor and VMs (context id + port)
  - Lightweight transport layer

How to talk with accelerator?
DPDK
Data Path Traffic Flow

Game frame->video stream data path

1. IRR client receives game frame and push to VM kernel vsock to transmit.

2. User space driver who emulates virtio backend ring Rx/Tx for virtio vsock, receives packet from VM vsock device.

3. Forwarding traffic between vhost user device and virtio vsock backend driver for VCA VOP vsock device.

4. User space driver who uses NTB to emulate virtio backend ring Rx/Tx for vca virtio vsock, sends packet to VOP device.

5. IRR server receives the render the frame and encoded into video streams using OpenGL, UMD and so on.

[VOP vsock control NTB and map remote resource according to designed ring format (virtio likely).]
1. Bring up VCA2 card, and configure the context ID for the node on card.

2. Set up DPDK environment as usual.

3. Start DPDK applications with two ports: ./examples/vsock_fwd -l 21-24 -n 4 --socket-mem 1024,1024 -- vdev="net_vsock0,iface=/tmp/dpdk-vca0.sock,dequeue-zero-copy=1" --vdev="vop_user0,path=/dev/vop_virtio00,iface=vop"

4. Bring up VM with virtio vsock user: -chardev socket,id=vus0,path=/tmp/dpdk-vca.sock -device vhost-user-vsock-pci,chardev=vus0,id=vsock-pci0,guest-cid=8

5. Run applications/Iperf in VM and accelerator.
- 15x Games @ one node run successfully as expected
Future Work

- Further Cloud Gaming stack integration and tuning
- Optimization
  - Remote memory access optimization
  - Enlarger buffer to improve efficiency
  - Enable DMA/CBDMA for buffer moving
  - Zero-copy in receive side
Thanks

Jingjing Wu
jingjing.wu@intel.com

Owen Zhang
owen.zhang@intel.com
Backup
- Qemu - vhost vsock user support.
- DPDK - Polling mode driver of vhost vsock ring.
- Tools - Enable AF_VSOCK on Iperf
- DPDK app: Fwd without dropping
Components (Host <-> Accelerator)

- DPDK
  - Polling mode driver of vop vssock ring based on NTB.
- VCA kernel driver
  - Virtio vssock driver based on NTB (VCA2 side)
  - Interface provided user space to map NTB BAR and trigger event.