Real Time Rendering and Raytracing on the Edge

July 2019
Building with 5G and Edge Compute
5G: Latency and Bandwidth

Traditional computing especially on mobile has left the heavy lifting to the hardware onboard devices. With larger screens and thinner form factors the difficulty to balance performance with quality and battery life has been a challenge for developers.

With the development of the 5G network latency has been reduced and bandwidth has been expanded to the point that many computing tasks can be offloaded.
Mobile Edge Computing

The development of Edge Computing has allowed for even quicker response times than from traditional public clouds. By placing computing devices and GPU’s right in the last leg of the network we allow for super quick round trip times and rapid options for offload and virtualization.

Edge compute is however an expensive resource with many unique characteristics so designing for the edge requires the utmost care.

Benefits Include:
• Processing complex workloads on limited hardware
• Reduce heat on a user endpoint
• Extend battery life on a constrained device
GPU Services on the Edge
Development Roadmap

Create an Edge Native Platform and API Services that allows for new classes of experiences on all devices:

- Develop easy to access API’s for Rendering, Computer Vision, XR Lighting, Raytracing, Spatial Audio and more
- Create an easy to use platform that lets anyone create and deploy Edge Services
- Connect Edge Services to client platforms with simple plugins for Unreal Engine and native platforms (Windows, Android, iOS)
- Enable developers to test and build!
**Edge XR Lighting API**

Benefits of Edge Based Lighting

- Real-time lighting / reflections on mobile devices for XR applications
- Low latency makes real-time reflections in XR possible
- End user setup of live AR portals and subscription could support new types of shopping / tourist / entertainment experiences
- Could also be used for generating in game lighting / reflections, instead of camera, imagery can come from virtual world to provide higher fidelity mobile experience than is available on current hardware
Real-Time Rendering API

- Platform to offload rendering to the edge
- Streams results to client devices
- Distributed render graph framework
- Supports compositing, compression, asynchronous GPU read and streams to client devices
- Client SDK features a mobile friendly forward renderer which composites server data in real-time

Benefits of MEC

- End user can make their own rendering requests
- End user gets dedicated container for custom rendering
- Offloads graphics processing
  - Saves device power
  - Allows you to do things that cannot be done on mobile hardware
  - Supports MEC thin client and path for cheaper phones
Raytracing API

Benefits of Edge Raytracing

• Supports high end desktop quality graphics on mobile devices
• End user can make their own rendering requests
• End user gets dedicated container for custom rendering
• Vulkan + RTX pipeline
• Lightmap atlas stream combined with local rendering pipeline for hybrid lighting effects
Computer Vision API for XR

Benefits of Edge Based CV

- Real-time horizontal scaling computer vision for 2D and 3D recognition and tracking
- Supports full or hybrid render
- Fully featured webGUI for designing and training targets
- Supports occlusion, classification, and segmentation
MEC Deployment Requirements

- GPU & CPU architecture variants
- Requires a dedicated container per application
  - Each container capable of supporting multiple concurrent users
- Resource requirements determined by application
- Container launched upon application request and delete on disconnect.

Benefits of MEC

- Ability to offload expensive audio processing to free client resources & enable thin clients
- 15 – 50ms total RTT target, including DSP
- Supports high-end, desktop-quality immersive & social audio experiences, otherwise impossible on mobile
- Supports large number of concurrent OTT voice chat & spatial audio users per application
- Able to do machine learning on voice for captioning, translation, & other data analyses
Verizon GPU Edge Platform
GPU Based Orchestration Platform

Verizon XR Team has been building an independent GPU Based Orchestration system specifically aimed at solving the following:

- GPU slicing and management of vGPU/vCPU slices for multi-tenant applications
- Load balancing of distributed clusters of GPU blades
- Remote provision, deploy and ingress for Docker-based GPU Applications
- Rapid MEC development pipeline for mobile device to compute backend using Unreal Engine or Native Applications
- Providing integration and API test bed for Edge Native API's
## Platform Stack

**Verizon XR GPU Orchestration Platform:**

Shared cluster Paas. Orchestration and federation of Kubernetes clusters within Edge Locations and across the whole network.

### In cluster container orchestration:

Kubernetes clusters with custom NVIDIA docker support.

### Virtualization:

- Red Hat Enterprise Linux® kernel, Kernel-based Virtual Machine (KVM) technology, and oVirt virtualization management projects.
- NVIDIA GRID virtualized GPUs

### Hardware:

- **1U Testbed**
  - 2x Xeon-S 4110, 128GB RAM, 2x 24GB NVIDIA RTX Quadro R8000

- **2U Testbed**
  - 2x Xeon-S 4110, 128GB RAM, 2x 24GB NVIDIA RTX Quadro R8000

- **4U Testbed - 2x Xeon Gold 6168, 128GB RAM, 8x 24GB NVIDIA RTX Quadro R8000**

Confidential and proprietary materials for authorized Verizon personnel and outside agencies only. Use, disclosure or distribution of this material is not permitted to any unauthorized persons or third parties except by written agreement.
Orchestration Platform - Functional Architecture

Cloud Hosted Layer
- Web Portal
- Admin Portal
- User Management
- Dev Portal
- Rancher
  - Deployment and Management
- JFrog Docker registry
- Federation API
- Analytics API

Test Bed + Edge
- Intelligent Routing
- CPU Masters
- GPU Node
- GPU Node
- GPU Node
- GPU Node
### Flat Assets

**Why would you want this?**
- FlatAsset Pipeline

<table>
<thead>
<tr>
<th>Accept all types of assets on the web.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process and convert on massively parallel GPU’s on the cloud</td>
</tr>
<tr>
<td>Keep only the data you want or need to use the file.</td>
</tr>
<tr>
<td>Generate new data not supported by current interchange formats</td>
</tr>
<tr>
<td>Collect information about the file so we can automate the remaining bits of the pipeline.</td>
</tr>
<tr>
<td>Package assets flat so they can be read with very low latency</td>
</tr>
</tbody>
</table>
CONTINUOUS INTEGRATION
EFFECTS ON PRODUCTION
WHY WOULD WE DO THIS?

- Tracking the users decisions on the cloud enables us to construct the game using their decisions in different scenarios
- Because we have already reduced lots of pipeline steps the game can be reassembled quickly
- Packaging assets into streaming formats and separating out LOD works really well
- Big data analysis can help optimize routines by comparing users decisions
- Players at home see the game updates live.
EDGE SERVICE CALL FLOW

User Endpoint → Public Cloud Platform → Orchestration Layer → Kubernetes → Docker → Edge Service

- Application Requests Edge Service
- Authentication
- Service init
- Provisioning and Utilization data
- Get Container Instance
- Cluster load analysis for Smart Routing
- Container Provisioning and Initialization
- Application Data Pipe
- Service Provisioning And Initialization
- Service Invocation
THANK YOU

Raheel Khalid - @rkhalid890
Edge/Cloud Vision Stack

Architecture Overview

get Target
Create Target (GPU)
Update Target (GPU)
Delete Target
Get DL data
Get TR data

AWS Vision Services – EKS (VPC)

Vision Publishing Clients (ARD, Oath Etc.)

AWS API Gateway

Get vision cluster

Client APP (MEC/Local)

Launch and connect to vision

Smart Routing

MEC Vision Kubernetes Cluster

Vision Docker Container

Web RTC
Object detection
Pose estimation
Rendering engine

Event Server

WebRTC
Object detection
Pose estimation
Rendering engine

Confidential and proprietary materials for authorized Verizon personnel and outside agencies only. Use, disclosure or distribution of this material is not permitted to any unauthorized persons or third parties except by written agreement.
THE PATH OF AN UPDATE

Virtualization – Best Case 3 frames, Worst Case 10+ frames

Hybrid Offloading – Best Case 3 frames  (Updates and Draws can occur Asynchronously)