A Cloud Gaming Framework for Dynamic Graphical Rendering Towards Achieving Distributed Game Engines

James Bulman, Peter Garraghan

Evolving Distributed Systems Lab

Lancaster University, UK

Lancaster University

DS-LAB

THE DISTRIBUTED SYSTEMS LAB AT LANCASTER

Lancaster University

Video Games

World's largest entertainment sector

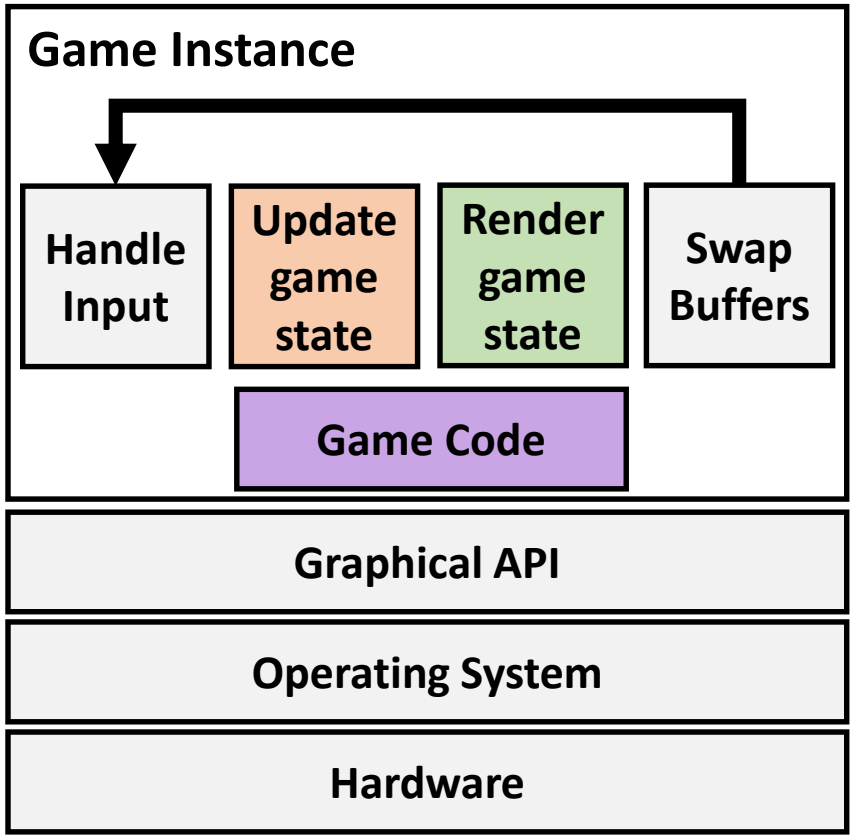
- $143.5 billion in 2020

Game engines use tightly-coupled subsystems

- Graphics, physics, AI, lighting, audio, etc…
- Synchronize within the game loop

Ambitious creativity -> Quality of Experience

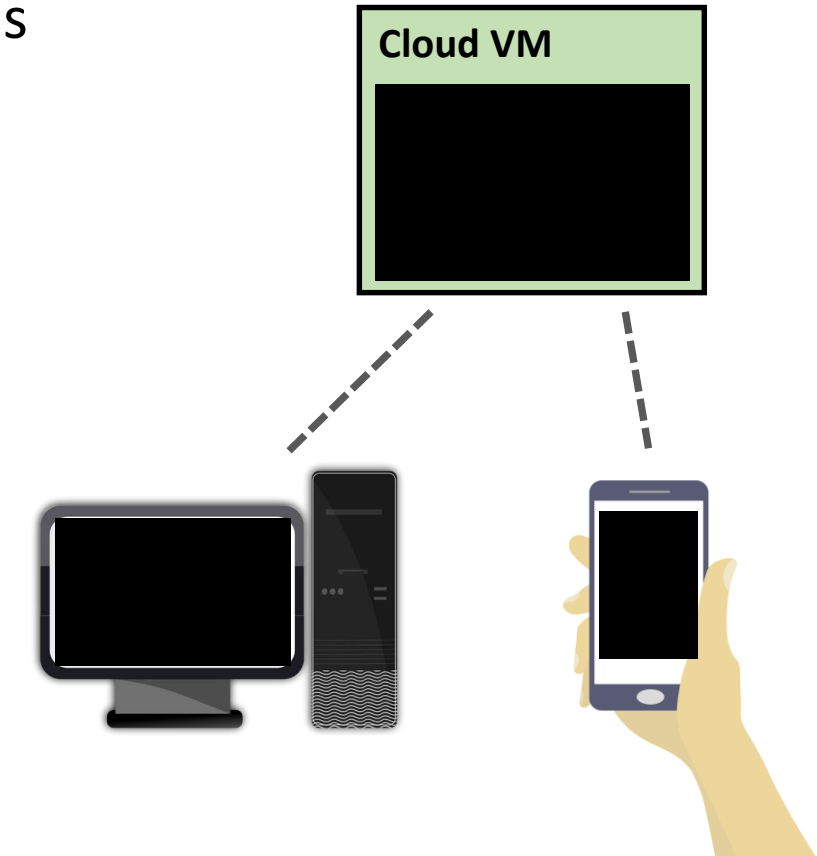
- Higher computation & hardware demands
- Game consoles, GPUs, VR, etc.



Cloud Gaming

Cloud gaming exploits cloud resources providing Games as a Service

- Google Stadia, Microsoft Project xCloud
- Game executed in cloud VM with minor* modifications



Benefits

- Instant access to games without downloads
- Powerful hardware -> higher quality experiences
- Cross-device access

Lancaster University

Research Problem

Reliance on Cloud performance & dependability

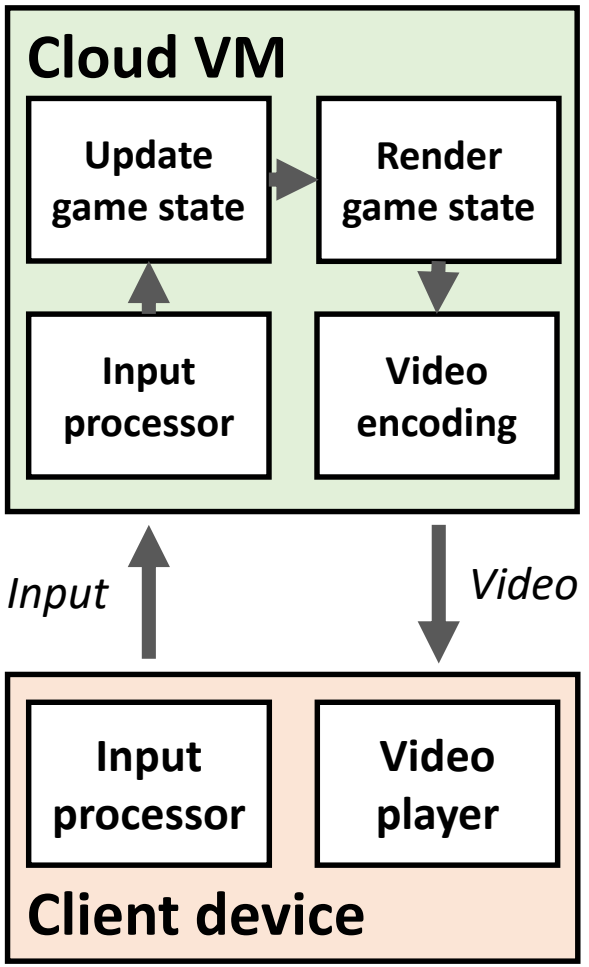
- VM interference or failure
- Network volatility
- Inconsistent framerate, total service loss

Monolithic game engine architectures

- Tightly-coupled subsystems
- Limited portability (OS, graphics API)
- Game per VM: *"Equiv. game experience + latency"*

Allow cloud-client execution

Distributed game subsystems



Lancaster University

Objectives

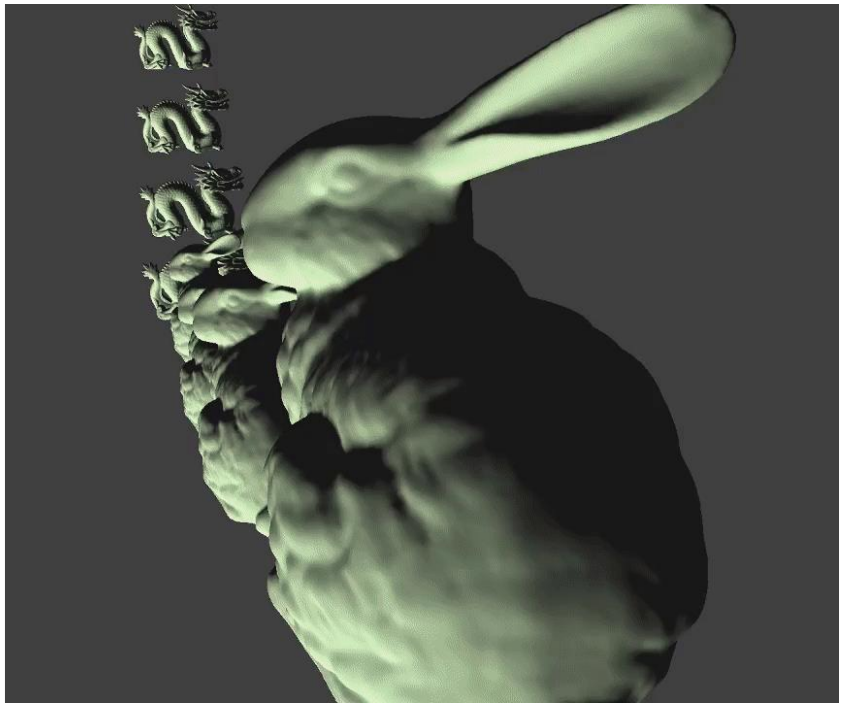
A Cloud Gaming Framework for Distributed Game Engines

Features

- Loose-coupling graphical renderer and game engine
- Dynamic cloud-client frame interlacing
- Graphical API hot-swapping

Advantages

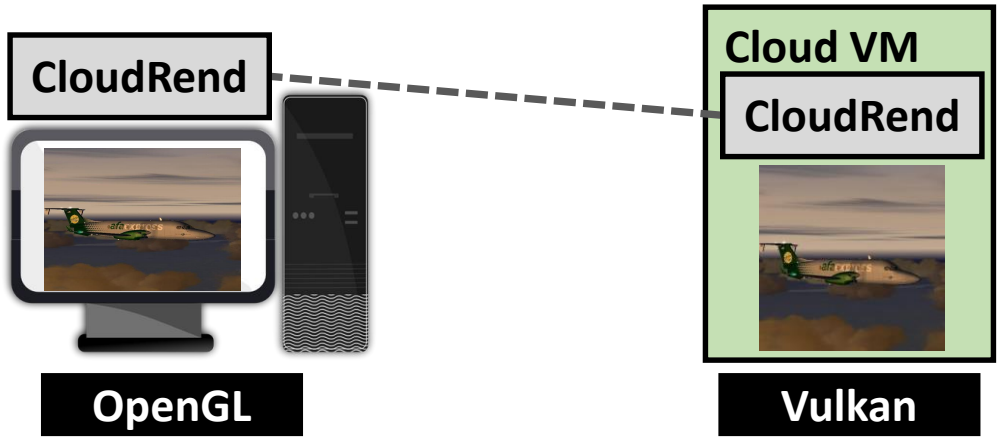
- Platform and graphics API independence
- Tolerate against Cloud failure and network loss
- First distributed game engine designed for the cloud

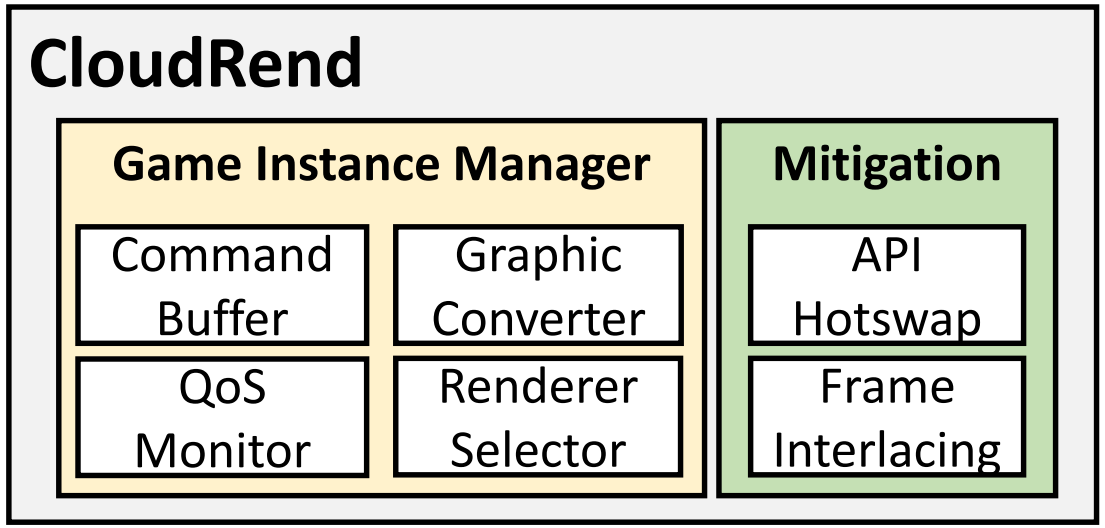


Overview

CloudRend

- Leverages generic graphical commands
- Converts these commands to API specific calls
- Cloud-client rendering
- Supports Vulkan, OpenGL

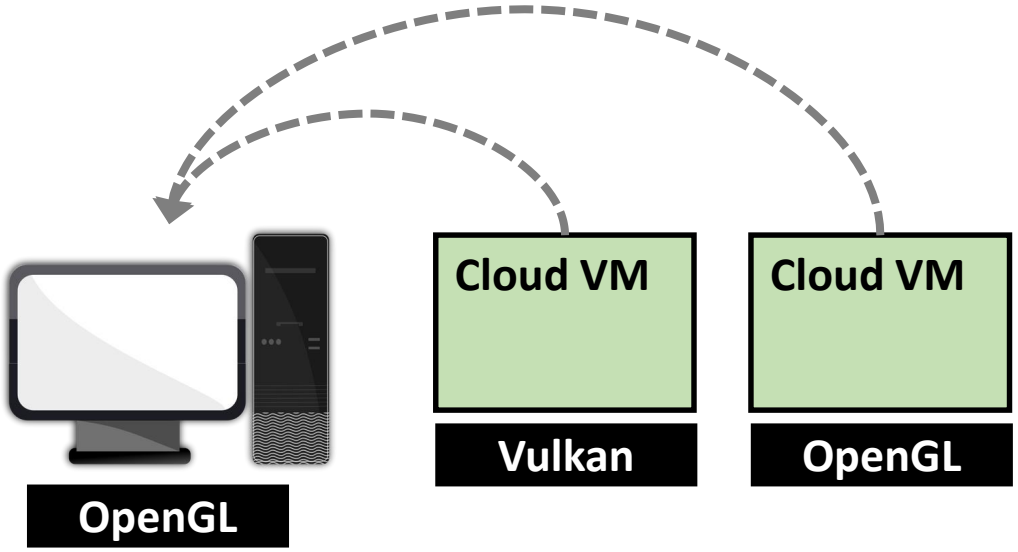




Mitigation

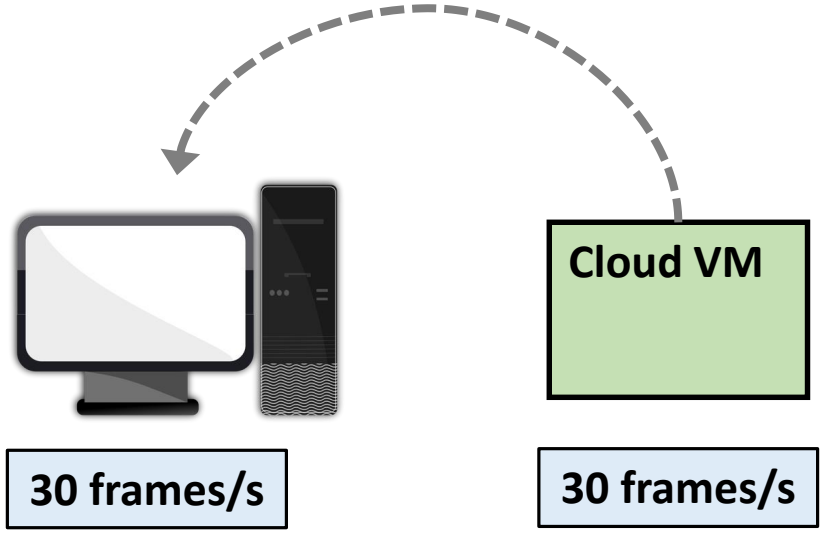
Hotswap

Run-time switching of Graphical APIs (Vulkan, OpenGL)

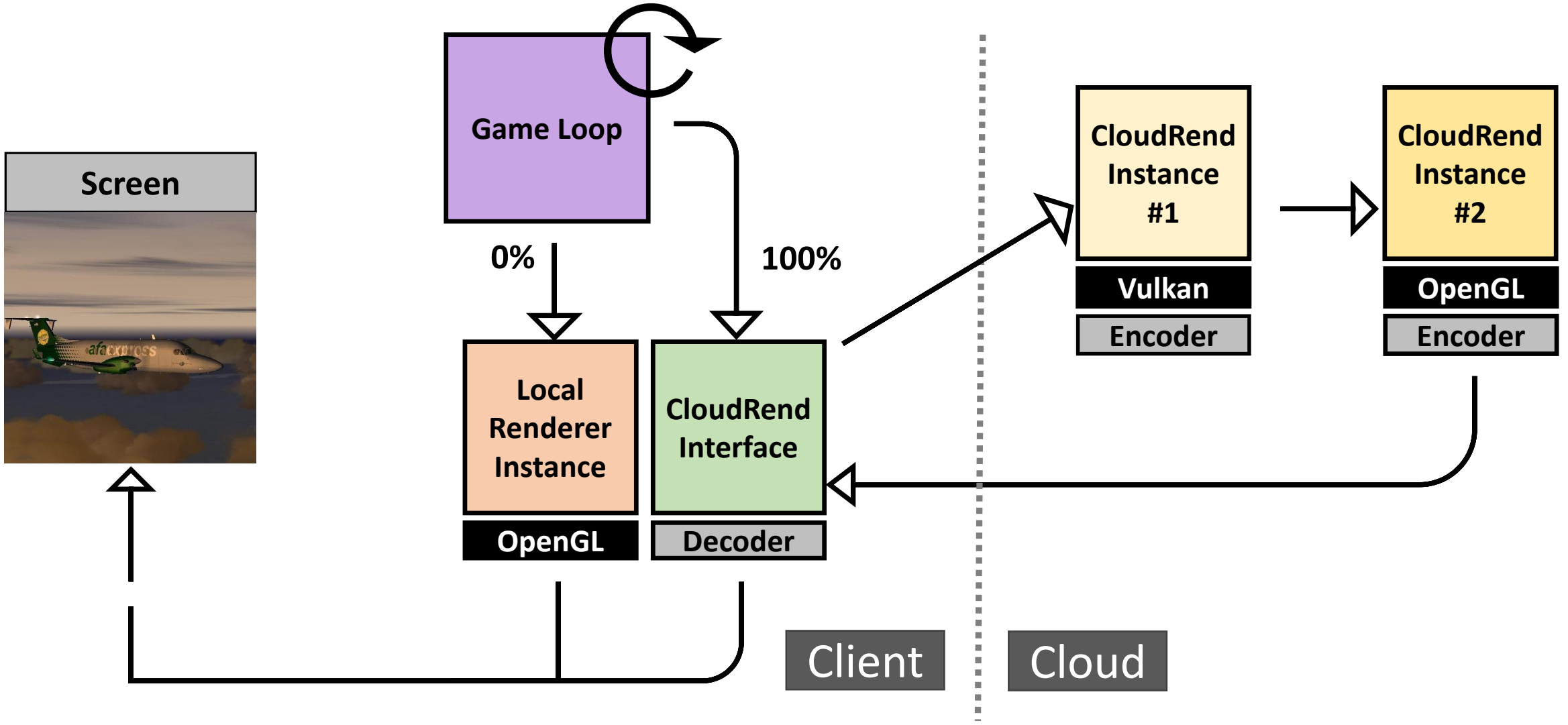


Frame Interlacing

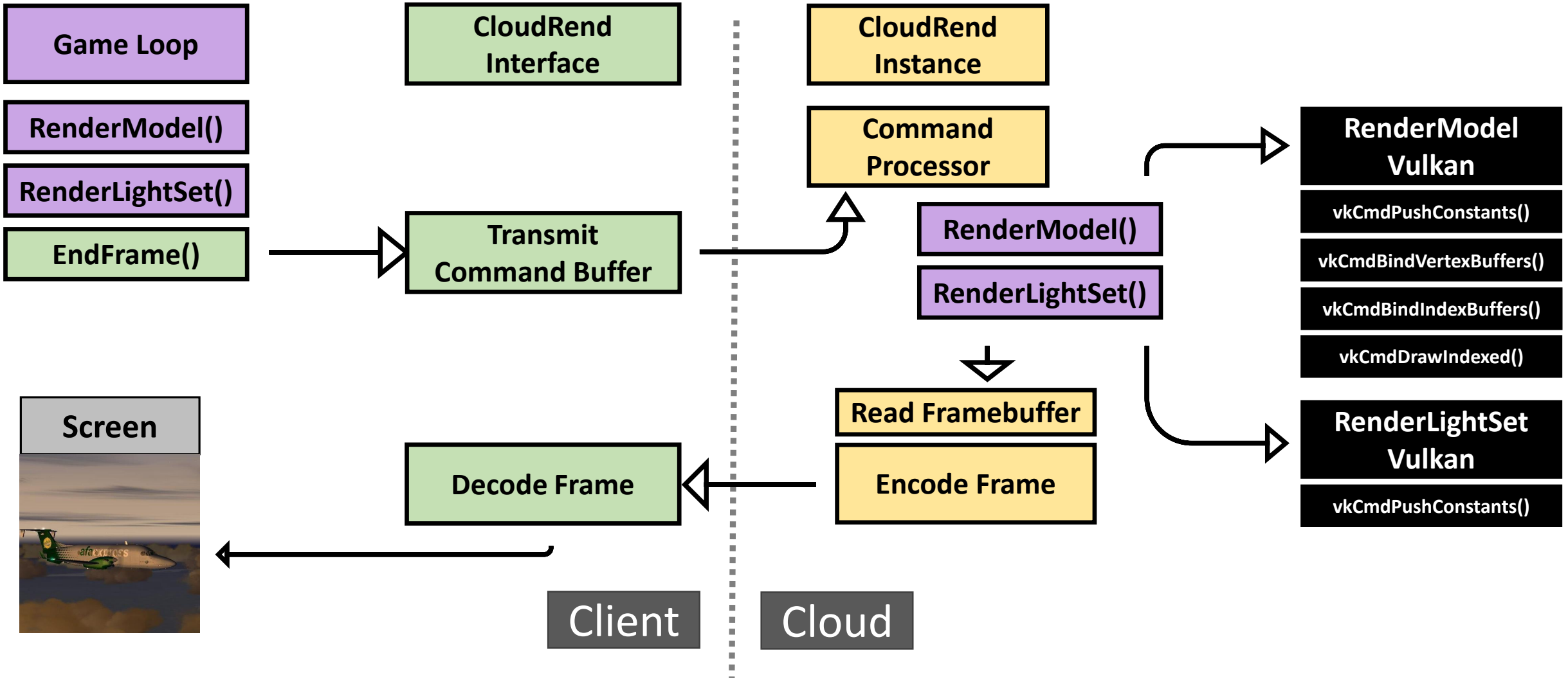
Collaborative cloud-client rendering (based on network)



Architecture



CloudRend


Game Loop
RenderModel()
RenderLightSet()
EndFrame()
CloudRend Interface
Transmit Command Buffer
CloudRend Instance
Command Processor
RenderModel()
RenderLightSet()
RenderModel Vulkan
vkCmdPushConstants()
vkCmdBindVertexBuffers()
vkCmdBindIndexBuffers()
vkCmdDrawIndexed()
RenderLightSet Vulkan
vkCmdPushConstants()
Read Framebuffer
Encode Frame
Decode Frame
Screen
Client
Cloud

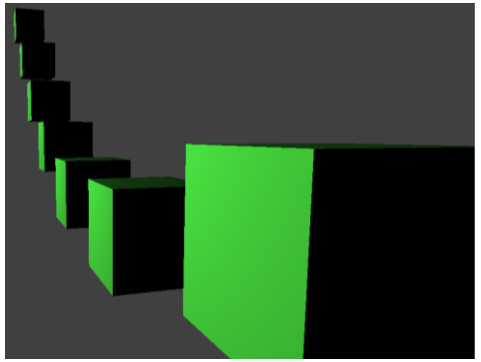
Setup

System

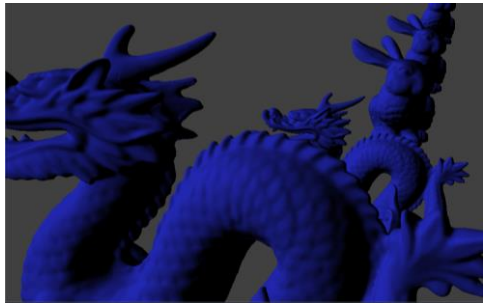
- **Cloud**: i7-7700HQ, GTX 1050 GPU, Vulkan/OpenGL
- **Client**: Raspberry Pi 4, Quad Core Cortex-A72, OpenGL ES 3.0
  Variable latency wireless network (16 – 100 ms)

Experiments

- Cloud-client frame interlacing (100-0% … 0%-100%)
- Graphical API hotswapping per frame
- 10,000 frame per run, frames per second (FPS), network

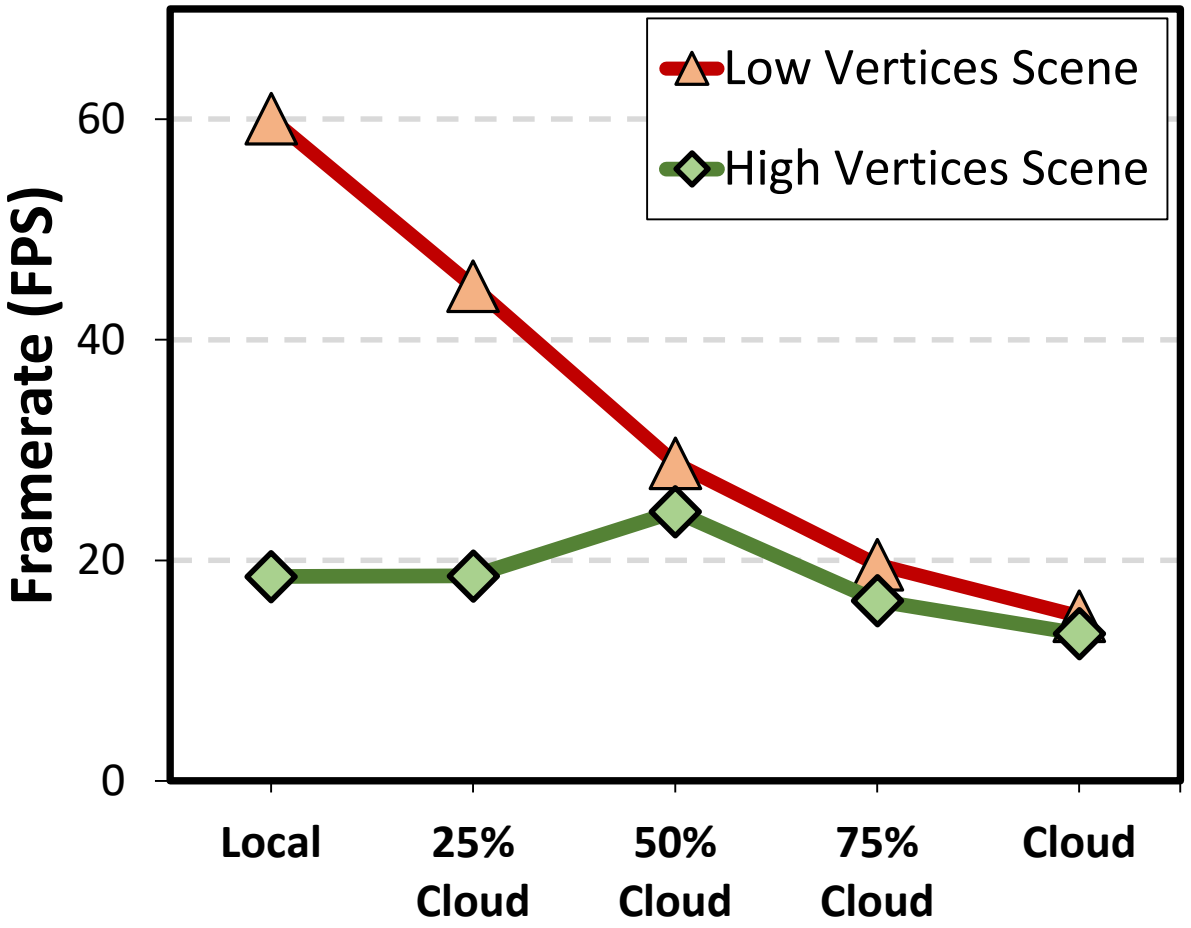


Low: 500 vertices



High: 200k vertices

Initial Results



Trade-off render complexity vs network latency

Higher consistent framerates when interlacing

Low network utilization
~4 Mbps

Lancaster University

Conclusions

Cloud gaming framework via distributed game engines

- Successful game distribution across cloud and client devices
- 33% performance gains, cloud latency vs. client computation

Vision

- Create fully-fledged cloud gaming framework and distributed game engines
- Cloud gaming in the home

Future Work

- Network-aware + power-aware frame interlacing
- System at scale – shared subsystems
- More decoupled subsystems – AI, collision detection, physics

Thanks

Contact

- **James Bulman** - j.bulman@lancaster.ac.uk
- **Peter Garraghan** - p.garraghan@lancaster.ac.uk

Evolving Distributed Systems Lab

School of Computing & Communications

Lancaster University, UK