Understanding Cloud-Based Visual System Architectures

Jeanette Ling
Rockwell Collins
Salt Lake City, Utah
jeanette.ling@rockwellcollins.com

ABSTRACT

The Government's "Cloud First" policy of 2011 set an accelerated course of government technology migration to cloud resources. The benefits of cloud services and infrastructure are appealing for use in simulation and training for many reasons, including the ability to provide point-of-need (PoN) simulation, freedom from hardware maintenance and upgrades, reduction of capital expenditure and hardware footprint, and practically limitless resources that allow ease of scalability. Evaluation of the fitness of visual system services for migration to the cloud as per the cloud-first guidance of readiness and value is highly dependent on the intended use case and architecture of a cloud-based simulator.

While attractive in concept, serious limitations in training quality and effectiveness can exist depending on the implementation strategy of a cloud-based visual system. This paper explores the technical challenges and functional ramifications of distributing visual system components across the cloud compared to on-premises resources. Topics include latency, performance, distributed visual system architectures, latency tolerance of basic visual system components, and edge device computing.

A wide spectrum of use cases exists within the simulation and training realm, and cloud-based visual systems must provide a flexible and adaptable hybrid cloud architecture to achieve required goals across very diverse training needs and physical infrastructure.

ABOUT THE AUTHOR

Jeanette Ling is a Principal Software Engineer in the Visual Systems group of Rockwell Collins with 30 years of experience in the industry. Her career has focused on personal computer (PC)-based visual systems that use consumer-off-the-shelf (COTS) graphics cards, and includes pioneering work on some of the first true 3D graphics cards available for PCs as an engineer with Evans & Sutherland. Ms. Ling has held primary engineering roles in producing visual systems for key military programs that span the range from fast-jet to ground vehicles, including full mission simulator (FMS) dome systems for the F-35 Joint Strike Fighter (JSF), helicopter simulators for Aviation Combined Arms Tactical Trainer (AVCATT), and ground warfare training systems for Close Combat Tactical Trainer (CCTT). Ms. Ling holds a B.S. in Computer Science and has a patent pending related to visual system render performance improvements.

Understanding Cloud-Based Visual System Architectures

Jeanette Ling
Rockwell Collins
Salt Lake City, Utah
jeanette.ling@rockwellcollins.com

INTRODUCTION

The Government's "Cloud First" policy (Kundra, 2011) set an accelerated course of government technology migration to cloud resources. Traditional on-premises visual systems seem ripe for cloud migration because the systems are expensive, dedicated, compute-intensive machines with large physical footprints and requirements for regular maintenance, upgrades, and environment control. The practically limitless resources of the cloud are highly appealing for hosting visual systems to eliminate or reduce many of these negatives while providing new benefits such as ease of scalability and the opportunity for point-of-need (PoN) simulation.

Although attractive in concept, serious limitations in training quality and effectiveness can exist depending on the implementation of a cloud-based visual system. This paper explores the technical challenges and functional ramifications of distributing visual system components across the cloud compared to on-premises resources. We begin by describing cloud options that exist for hosting visual system components. (Morris, 2011) (Abacus Next, 2017).

Public Cloud

Public cloud service platforms are those owned and operated by a third-party company such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform. These cloud service providers build and maintain huge data centers around the world with pay-as-you-go public access to a variety of resources including central processing unit (CPU) cores, random access memory (RAM), graphics processing unit (GPU) cores, GPU memory, and disk storage. They provide building blocks of various compute, storage, and other capabilities, allowing users to assemble systems using cloud resources that fit the tasks, but within the constraints of the provider's available building blocks. An important attribute of public clouds is the ability to quickly and easily scale up or down to accommodate varying workloads. The cloud provider handles the physical infrastructure, thus freeing the user from buying and maintaining hardware. Users connect to the abundant resources of the public cloud via a network.

Private Cloud/Data Center

Private clouds are compute resources in data centers that are for sole use by one entity, such as a company or organization, often housed in the same facility as the users. These data centers are capable of providing the same types of resources and virtual work environments as in a public cloud, but offer the ability to customize resources to fit requirements, including the private network over which users connect. Unless hosted by a third-party provider, scalability and maintainability may not be as easy as on public clouds such as AWS, since provisioning resources may require Information Technology (IT) involvement, as well as capital expense for increased capacity. Security can be an important advantage of private clouds over public clouds.

Hybrid Cloud

A hybrid cloud uses a combination of public cloud and private cloud resources, exploiting the advantages of each type to the benefit of the overall application.

Edge Device and Edge Computing

Users are at the "edge" of the network, interacting on an edge device such as a smartphone, tablet, or personal computer (PC). Edge computing performs processing of time-critical data closer to the data's source, rather than sending it across the network to potentially distant public cloud or private data center resources. (Butler, 2017).

VISUAL SYSTEMS AND THE CLOUD

A visual system comprises hardware and software that renders, displays, and controls a virtual scene for a training scenario in real time. Visual systems are typically compute and GPU-intensive applications and require large storage capacity for scene files storage. The massive resources of the cloud make it attractive for hosting visual systems, especially when there is a huge demand for PoN delivery of simulation and training.

Visual systems typically have high expectations for performance. They are expected to run deterministically at the required system update rate, 60 hertz (Hz) typical, which means they draw a new frame for the scene at the constant system update rate. Any deviations to that constant rate are perceived as “stutters” or “hitches” to an otherwise smoothly-updating scene. Visual systems also require low latency such that the system feels immediately responsive to the user.

Visual System Latency

Visual system latency for typical ground and helicopter systems is required to not exceed 100ms and 70ms, respectively, with update rates of 30Hz and 60Hz. Typical fast-jet trainers are even more demanding with visual system latency of less than 50ms and update rates of 60Hz. Visual system latency is defined as the time a visual system receives eye point position data from a user action such as a joystick movement, to completing display of the first field based on that data.

User Latency

User latency is defined as the time from when a user performs a local control action, such as a joystick operation, to when the user sees the effect of that action displayed in the scene. Visual system latency is a subset of overall user latency. Higher user latency times make the system seem sluggish or “laggy.” Grigorik (2013) states that human perception of delays is relatively constant; humans perceive delays of less than 100ms as instant, and delays of 100 to 300ms as noticeable, while delays of above 300ms are perceived as the machine not keeping up with the workload. Delays of 1000ms or more cause a user to lose engagement with the task.

The Obvious Cloud-Based Visual System

When considering a cloud-based visual system, the case typically envisioned is one where all components, other than user display and user control devices, are in the cloud (Figure 1). The visual system application makes use of the abundant cloud-based resources, including CPU cores, RAM, disk storage, GPU cores and GPU memory, to retrieve scene files from cloud storage, then build and render a scene relative to the user’s eye point. A rendered frame is compressed and streamed across the network to the user’s edge device, where it is decoded and displayed.

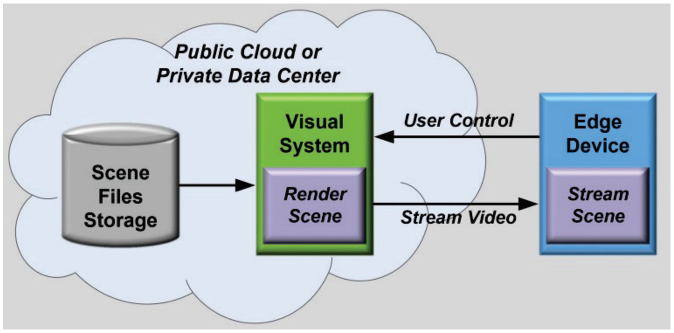

Figure 1. Visual System in Cloud with Streaming to Edge

The Problem: Latency and the Cloud

Round-trip times between a user’s edge device and the cloud vary widely, depending on connection type, geographical locations of source and destination, route, security, and other factors. Any visual system design that uses the cloud must be cognizant of the inherent added latency associated with network transmissions to and from the cloud.

Network Latency

Network latency is the round-trip network transport time from a user to and from a network-connected device, and generally equates to ping time. Connectivity requirements from streaming game providers give insight into acceptable network latency for game play, and hence what may be acceptable for streaming visual system scenes. NVIDIA’s game streaming service, GeForce NOW, requires less than 60ms ping time to an NVIDIA data center hosting the game (NVIDIA, 2018). If visual systems are to be hosted in the cloud, the network latency must be considered a portion of the visual system latency. A 60ms ping time exceeds fast-jet visual system latency requirements, consumes nearly 86% of the helicopter latency budget, and is 60% of ground-based latency requirements.

Amazon builds data centers (AWS regions) around the world in an effort to minimize transfer times and hence network latency between the user's location and the data center chosen to host an application. Table 1 shows the results from round-trip ping tests to various AWS regions. For the Los Angeles user location, four separate network latency round-trip ping tests were done by the same person in the same place, but using different connection types and devices, showing that round-trip ping times can vary widely for someone sitting in the same chair. Ping times that meet the NVIDIA GeForce NOW requirement are highlighted in green.

Table 1. Network Latency: Ping Times to AWS Regions

	AWS Round Trip Ping Time in milliseconds (http://www.cloudping.info/)								
User Location	Los Angeles, CA				Orlando, FL		Tokyo, Japan		Airborne in Alaska
Connection	wired, No VPN	wired, VPN	wireless	LTE	wired	wireless	wireless	4G	in-flight wireless
Device	PC	PC	phone	phone	PC	phone	PC	phone	PC
AWS Region									
US-East (Virginia)	97	122	204	159	57	131	181	246	750
US-East (Ohio)	105	128	115	147	72	132	185	231	873
US-West (California)	22	160	30	95	129	138	295	177	691
Europe (London)	154	186	182	259	157	171	295	321	892
Asia Pacific (Tokyo)	133	225	153	200	214	331	25	59	992
AWS GovCloud (US)	46	159	56	131	145	172	146	204	838

Network latency is a difficult obstacle to overcome because unlike network bandwidth, memory capacity, CPU clock rates, GPU cores and other system resources that generally improve exponentially on a Moore's Law-like curve, network latency is constrained by the laws of physics (Grigorik, 2013), and is worsened as layers of routing, security and virtualization are added. Figure 2 illustrates major functional blocks that contribute to the overall user latency experienced by on-premises vs. cloud-based visual system users. Visual systems that are fully hosted in the cloud incur additional latency due to network transport, video encoding, and video decoding when compared to on-premises systems. It also shows the differences between "user latency", "network latency", and traditional on-premises "visual system latency". Cloud-based visual system user latency is the sum of all delays incurred from the moment of user input (e.g. joystick movement) to seeing that input's effect in the user's display. An inspection of the ping times of Table 1 *plus* visual system latency of 50ms or more, shows that very few cases fall under the "instant" human perception user latency time of 100ms; most are in the "perceptible delay" range of 100 to 300ms.

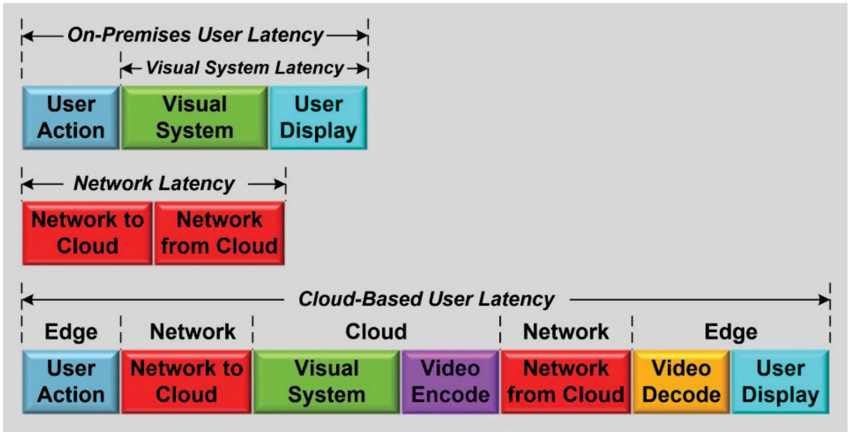

Figure 2. On-Premises vs. Cloud-Based Visual System Latencies

EXPLORING OTHER CLOUD-HOSTING STRATEGIES

Trends in other industries show that hybrid cloud architectures, along with edge computing for time-sensitive operations (Butler, 2017), (Janakiram, 2017), (Baker, 2017), are increasingly being used to achieve performance goals, in particular for latency-intolerant information processing. This paper now explores how some of these emerging concepts may be applied to visual system architectures in an effort to strike a balance between efficient use of cloud resources and acceptable user experience.

We begin with an analysis of four basic visual system components for their latency tolerance, requirements for compute, memory, and GPU resources, and data flow between components. By understanding these attributes, one may begin to envision other visual system cloud-hosting architectures that may work better for particular use cases. It is important to remember that visual system design will vary based on supplier; the components listed here are generally applicable to any visual system, but the latency tolerance, resource requirements and data flows will be implementation dependent. The component analysis methodology can be applied regardless of implementation.

Scene Files Storage (SFS)

Visual system scenes are constructed from a variety of sources that may include satellite imagery, terrain grid-post data, model files, geo-typical textures, detailed inset models for areas of interest (AOI) and other data. The amount and type of data is dependent on the use case; fast-jet simulation may need to span the whole Earth, but with high fidelity AOIs, whereas ground-based training may require scene files defining just a few city blocks. The required storage capacity can range from megabytes to terabytes or more. SFS appears well-suited to cloud storage solutions and can drastically simplify or eliminate the configuration management logistics of updating and distributing the data to many widely-dispersed users.

Scene Creation (SC)

A scene must be created relative to the user's current eye point. Scene files in the vicinity of the eye point are read from SFS, and then processed into a form that can be ingested by the rendering hardware. SC operations can entail terrain skin generation, texture synthesis for terrain, placement of 3D cultural features to populate the terrain, and incorporation of detailed model files for AOIs. The operation is compute and memory intensive, and sufficient resources must exist on the hosting device to support multi-threaded SC. The size of the geometric data created by the SC component can range from just a couple megabytes for terrain with no features, to 20MB or more for scenes with high-density content. Texture data handled by the SC varies widely depending on variables such as resolutions of texture images and vehicle speed. The initial loading of a scene may require handling of more than a GB of texture for a complex scene, but once a scene is loaded, textures may not change at all if paging is not needed. Cloud resources are abundant and scalable for both compute and memory, making SC a good candidate for cloud hosting.

Executive (EXEC)

Typical visual systems operate at 60Hz, meaning that a visual frame is produced at the rate of 60 times per second, equating to a frame length of 16.667ms. One of the important tasks of the executive is to handle the per-frame real-time aspect of the visual system, including update of the user's current eye point position and triggering frame rendering. The executive operation is not particularly compute or memory intensive, but it is highly time sensitive and requires determinism. The shared resource model of the cloud virtual machine environment may have unique challenges to determinism that are not an issue on a dedicated system. Data flow to the EXEC is minimal; per frame control data specifying the user's current eye point position can be less than 100 bytes.

Renderer (REN)

At the beginning of each frame, the EXEC orders the REN to draw the scene at the user's current eye position and for the channel's field of view. The frame may be displayed to the user via a direct video connection for on-premises rendering, or via video streaming for cloud rendering. The REN requires sufficient GPU memory to support frame buffers of the required resolution and samples per pixel, in addition to caching of scene geometry, textures, and state data. The GPU cores and clock rate must be adequate to produce the scene pixels at the necessary resolution and scene depth complexity. GPU memory and cores are abundant and scalable on cloud resources, and thus seem well-suited to hosting the rendering component of a visual system. The per-frame aspect of the REN requires determinism that must be achievable on a shared cloud resource.

LATENCY TOLERANCE OF VISUAL SYSTEM COMPONENTS

Visual system components have different tolerances to latency depending on their function. The basic components are grouped into three latency sensitivity categories, described below.

Per Frame Operations

Per frame operations are those that must occur each time a frame is rendered for a visual simulation. For the components under analysis, the EXEC and REN are per frame. Since the EXEC directs the REN to draw a frame at the current eye point, the REN is directly dependent on the EXEC. Per frame operations are latency intolerant, as any delays to their timely execution directly affect the overall user latency. An example of the time-critical nature of these per frame operations is the timely handling of ownship position updates and subsequent frame rendering that is required by a fast-jet pilot practicing intricate mid-air refueling maneuvers, where visual system delays to joystick movements can cause negative training effects (Jenkins, J. C. and Havig, P. R., 2015).

Multiple Frame Operations

A number of components within a visual system can tolerate multiple 60Hz frames to complete. These operations are latency tolerant compared to per frame operations, because their duration and completion do not immediately affect the current frame's rendering. Components in this category are those that are performed in anticipation of future frames' positions and rendering, but the current frame is not dependent on it. The SC component fits this category, taking multiple frames or even several seconds to complete. As the eye point approaches a new, non-moving object such as a building, the scene files associated with the upcoming object are paged from SFS, and the scene incorporating these new files is built well before the eye point reaches the new object. The latency tolerance of this component requires that the scene being actively created is independent of the scene geometry and texture necessary to render the frame at the user's current eye point, implying a double-buffered scheme.

Static or Rarely Changing

SFS is largely static, or changed on relatively rare occasion. This information may need updating in response to new data obtained from the field such as updates via drone reconnaissance, but its frequency of change is on the order of hours, days, or longer, and may not change at all during a simulation exercise.

USE CASE ANALYSIS

Four example use cases are described below, spanning the range from a simple ground-based walk-through on a hand-held mobile device, to a multi-channel fast-jet trainer. They are used to illustrate a wide variety of latency tolerance, resource requirements, and network connectivity that suggest different levels of value and readiness for cloud hosting. Hypothetical visual system component distribution models are given for each use case.

Use Case 1: PoN Reconnaissance Data Inspection

In this use case, a user does a first-person, ground-based walk-through of an AOI, analyzing the latest reconnaissance data supplied from drones. Users can be widely dispersed, and are often using mobile devices such as tablets or cell phones. The user will frequently stop and inspect areas, so higher user latency can be tolerated. Scenes are dense and complex and can cover a variety of gaming area sizes.

This use case is likely ready to be fully-hosted in a public cloud or private data center, coupled with streaming video to a range of devices (Figure 1), provided the network connection between the user and hosting resource is sufficient. Its relative tolerance to longer user latency due to stopping and inspecting an area, rather than requiring continuous or intricate maneuvers, makes it more accommodating to inevitable network latency. User latency, however, should not exceed one second, beyond which a user's flow and engagement is typically broken (Grigorik, 2013). Ample storage, CPU, RAM and GPU cloud resources are available for SFS, building the scene, and rendering large, dense gaming areas. Streaming support is a ubiquitous feature available on a wide variety of mobile handheld devices.

Use Case 2: Ground-Speed Trainer on Mobile Devices
This use case is a ground-based driver trainer in a training center. The users' edge devices may be mobile devices or laptops, and they will be driving vehicles at ground speeds through small gaming areas with moderate scene content.

Ground-speed trainers still require visual system latency under 100ms, so any network latency between the user and a cloud-hosted visual system needs to be minimized, perhaps via a private data center located in the training facility. If the network latency is unacceptable to host the full visual system in the cloud, a hybrid option could use the cloud resource to host the latency tolerant SFS and SC components, and then transfer the generated scene to the user's edge device. The edge device would then perform EXEC eye point control and rendering (Figure 3).

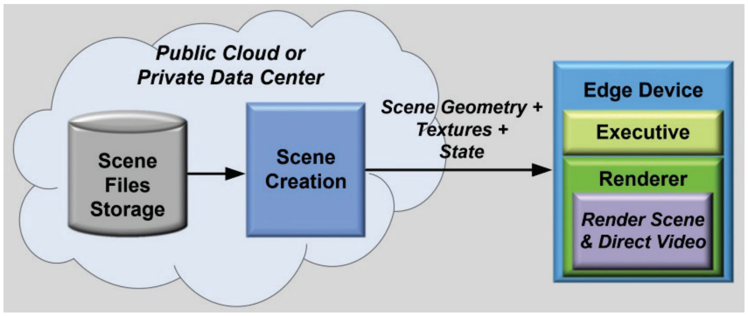

Figure 3. Latency Tolerant Components in Cloud, Latency Intolerant Components at Edge

This configuration could offer the lower user latency required for this trainer, while still benefiting from targeted use of cloud resources. The need to do rendering and user control on a variety of edge devices suggest a common solution is necessary to abstract the wide range of edge device hardware, such as a web browser-based thin client (Dukstein, 2017) or game engine. The edge device must have adequate storage for the small gaming area of moderate scene content, and have sufficient CPU, RAM, and GPU resources to render the scene at ground-based rates of 30Hz.

Use Case 3: Multi-User Helicopter Trainers
This use case considers a multi-user, multi-channel helicopter trainer at a military base. The trainer will have relatively high performance constraints of sub-100ms latency and 60Hz update rates. It will support multiple synchronized displays to provide a complete out-the-window view to a pilot and co-pilot. Furthermore, this case includes consideration for hardware virtualization, where trainers are required in several different rooms of the facility, but only a few of them are ever in use at any given time.

The value of a cloud-based visual system for this use case is high, since it allows allocation of available data center resources to only those trainers that are in active use. A private data center located on-base with low network latency to the trainers' edge devices (Figure 4) is likely ready to supply hardware-encoded video streams to the in-use trainers.

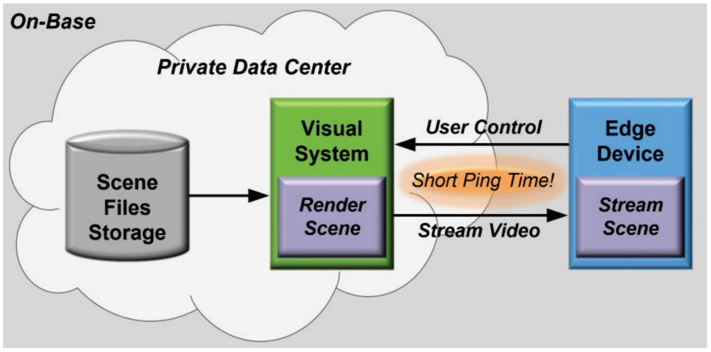

Figure 4. Private Data Center Hosting Visual System with Fast Network to User

Careful management of data center resources will be essential to achieve the 60Hz frame determinism required for helicopter training with virtual machines on a shared server. Otherwise, users may experience distracting or unacceptable performance stutters. Given the requirements for multi-channel synchronization, the display technology of the trainers must support synchronization and the cloud-hosted visual system must be able to produce multiple channels in a synchronous fashion. Channel synchronization of a cloud-based visual system is expected to be more difficult than a traditional visual system, since the variability in video frame transport time via a network is higher when compared to the direct video connections of a dedicated visual system.

Use Case 4: Fast-Jet Dome Trainer
This use case is a multi-channel, high resolution fast-jet trainer on a projected, edge-blended, channel-synchronized dome display. Critical user interactions such as mid-air refueling and carrier landings demand low latency and deterministic performance. Gaming areas cover the entire world and include high resolution, high-detail AOIs.

The readiness of cloud hosting any portion of this high-performance system may be limited to SFS in a public cloud or private data center, provided it is tightly-coupled to the SC component through a controlled, fast network that is adequate to supply paging of the high-detail and high-resolution scene files at a fast-jet flight rate (Figure 5). Critical low-latency user interactions cannot tolerate much additional latency from sources such as the network without suffering negative training consequences (Jenkins, J. C. and Havig, P. R., 2015). Attempting to stream from a public cloud or private data center with no compression artifacts and synchronization of multiple high resolution channels for a projected, edge-blended, dome display requires lossless encoding, high bandwidth, and synchronization technology.

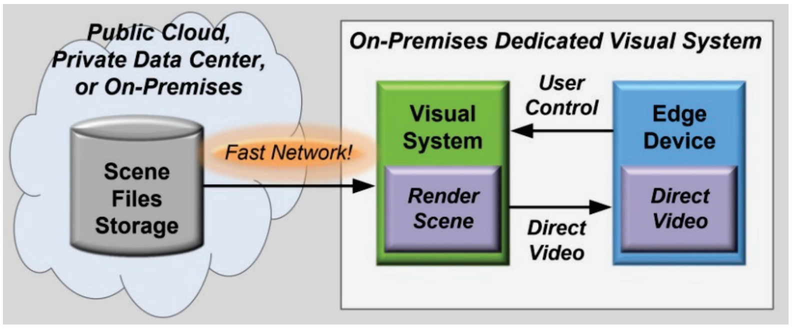

Figure 5. Dedicated, On-Premises Visual System with Fast Network Access to Scene Files Storage

DISTRIBUTED VISUAL SYSTEM EXPERIMENTS

A series of tests were conducted with prototype visual system components to explore different distributed cloud-based architectures and their suitability to a range of use cases. The experiments focused on measuring key performance parameters to understand the effects of moving the components between cloud and edge resources.

Method

The four basic visual system components were distributed across cloud and edge resources in various configurations with the goal of pushing the more latency tolerant operations further from the user, and more latency intolerant operations progressively closer to the user, and observing the effects. The following data were recorded:

- Round-trip latency as the time in milliseconds from a user control action until the effect of that action is seen by the user in the display (user latency)
- Ping time in milliseconds from user to applicable cloud location(s)
- Initial-position scene load time in seconds
- 10-mile reposition page-in time in seconds. Once the scene was fully loaded, reposition to another location 10 miles away, and measure the time to fully page-in the scene at the new position.
- Scene Creation time in milliseconds
- Executive time in milliseconds
- Renderer rate in Hz
- User's edge device display rate in Hz

Resolution of the rendered and displayed images for all tests was 1920x1080 (1080p). The test scene contains satellite imagery, geo-typical texture, airport inset, and 3D features (Figure 6). The scene is a 95x95 mile area surrounding the eye point consisting of 332,121 total triangles, of which 67,519 are drawn within the scene's field of view. Visual system component performance and latency timing were the only considerations during these experiments. Other visual system attributes such as video stream quality and security were not considered.

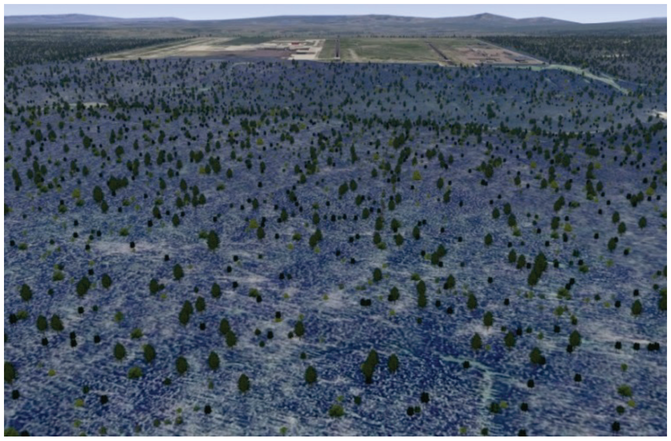

Figure 6. Test Scene

Apparatus

Two cloud hosts were used, an Elastic Compute Cloud (EC2) instance running a virtual machine on the AWS Ohio (OH) region, and a virtual machine running on a private data center in Cedar Rapids, Iowa (CRDC). They were provisioned similarly where possible, but differences were inevitable given limited available options (Table 2). The edge devices in all cases were PCs capable of running OpenGL 4.2, but with varying compute, memory and GPU resources.

Table 2. Public Cloud and Private Data Center Resource Provisioning

	AWS Ohio (OH) Region EC2 g3.4xlarge Instance	**Cedar Rapid Data Center (CRDC) Virtual Machine**
CPU	Intel(R) Xeon(R) CPU E5-2686 v4	Intel(R) Xeon(R) CPU E5-2697A
vCPU cores	16	14
vCPU clock (GHz)	2.30	2.60
RAM (GB)	122	32
GPU	NVIDIA Tesla M60	NVIDIA Quadro FX GRID M60-4Q
GPU memory (MB)	7680	4096

Experiment 1: Non-Distributed Visual System
Experiments were conducted for the non-distributed case where the full visual system (FVS) was hosted on each cloud resource and the user's edge device as shown in Figure 1. The FVS consists of the four basic visual system components, SFS+SC+EXEC+REN. Users in four locations participated, including Cedar Rapids, Iowa (CR), Orlando, Florida (ORL), Salt Lake City, Utah (SLC), and Burgess Hill, United Kingdom (UK). A thin client was used for the case where the FVS was hosted on the user's edge device, despite being unnecessary since the edge device has a direct video connection. This was for baseline comparison to the cloud-hosted cases.

Figure 7 illustrates the effect of network latency on the overall user experience. This experiment has the FVS residing on the hosting resource, including the latency-intolerant EXEC and REN components, so any network latency directly affects the overall user latency. The full stacked bar indicates the round-trip user latency. The segments shown within the round trip indicate duration, but do not indicate when a segment executes. For each of the four users, CR, ORL, SLC, and UK, the FVS was hosted in three locations, OH, CRDC, and on the user's edge device. Network ping times are shown for each user's edge device to the public and private cloud resources. Software video encoding is used on the FVS hosting device.

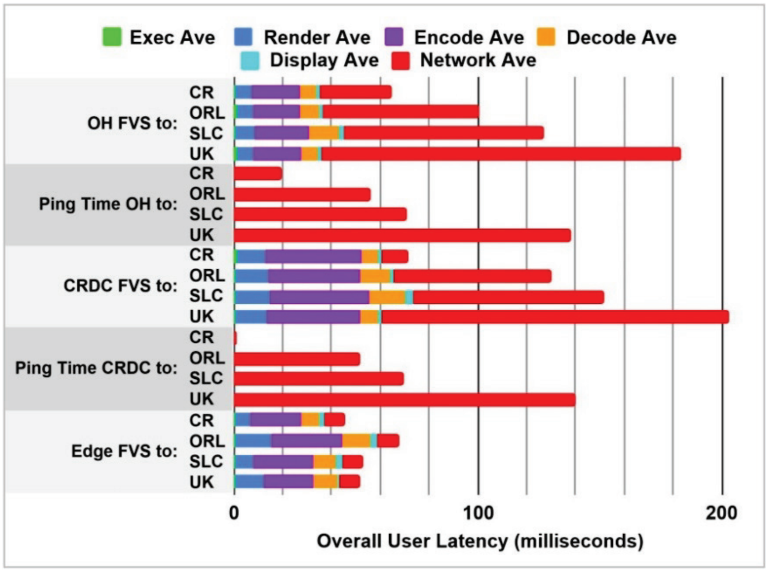

Figure 7. Effect of Network Latency on User Latency

Render Rate vs. Thin Client Display Rate
The rate at which the REN produces a frame may not be equal to the rate the thin client displays a frame for the non-distributed visual system, as shown in Figure 8. The main factor for the discrepancy between the two rates is the video encode and decode times, performed for these experiments via software. Hardware encoding and decoding can significantly reduce those times and will increase the thin client display rate. For these experiments, network latency does not affect the render rate or the thin client display rate; network latency only affects the overall user latency.

Variation in render times for the cloud resources compared to the SLC edge device is shown for a roughly five second duration in Figure 9. While the OH- and SLC-hosted REN are of similar performance and variability, the CRDC-based REN exhibited occasional spikes to ~200ms and an overall higher REN time.

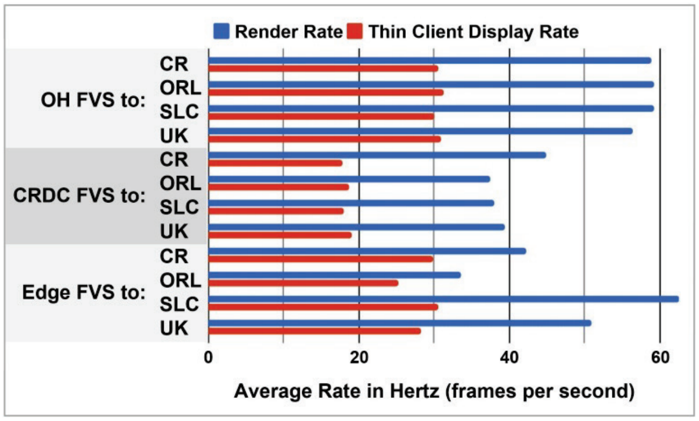

Figure 8. Render Rate vs. Thin Client Display Rate

The shared server resources of public and private clouds have unique challenges to determinism that are not found in traditional dedicated, on-premises visual systems. For deterministic systems, these performance variations and spikes must be understood and eliminated. Experience thus far is showing that performance lessons learned from dedicated systems regarding CPU cores and threads do not necessarily apply to virtual machines.

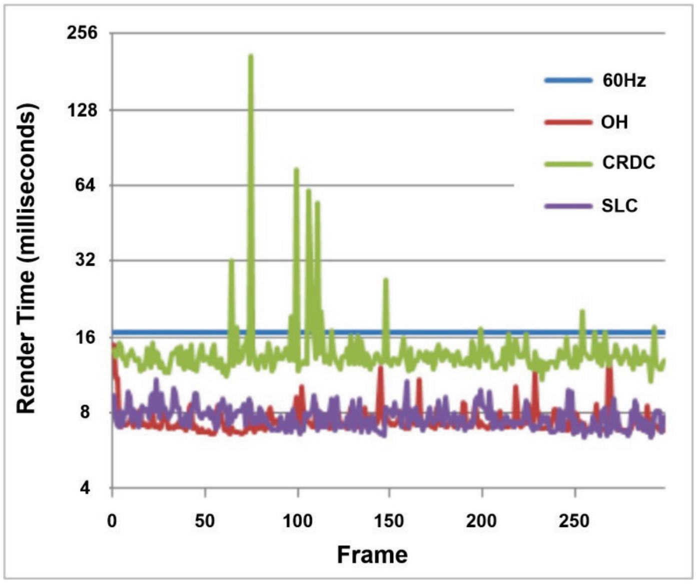

Figure 9. Variation in Render Times for Public, Private, and SLC Edge Hosted Visual Systems

Experiment 2: Distributed Scene Files Storage
The effect of network latency between the scene files storage and scene creation components is shown in Table 3. The 2.85GB of scene files were stored on solid state drives on the hosting machines. The drive was mapped via the network for access by the SC component. The visual system's SC component was running on the CRDC for each test; the only difference was the location of the SFS. The size of the generated scene geometry surrounding the eye point was 12.8MB, and the amount of texture loaded for the scene was 121.6MB. As network latency between the SFS and SC components increased, so did scene load times and page-in times.

The long load times observed with distributed scene files storage and scene creation components suggest excessive communication over the network may be occurring between these components. For use cases that cannot tolerate long delays in paging (fast-jet flight) or long scene load times, the delays must be understood and rectified before use of this distribution model is feasible. A slow ground vehicle use case in a constrained gaming area, however, may comfortably tolerate paging or scene load time delays.

Table 3. Effect of Distributed Scene Files Storage

	Visual System in CRDC for All Cases		
SFS location	**Round-Trip Ping Time, CRDC to SFS (ms)**	**10-Mile Reposition Page-In Time ave (sec)**	**Scene Load Time ave (seconds)**
CRDC	0	13	24
AWS OH	21	22	248
SLC	74	173	1710

Experiment 3: Only the Renderer on the Edge
The next experiment brought the latency-sensitive REN to the edge, but hosted the SFS, SC, and EXEC components together at varying locations with increasing ping times from the REN. The REN was always located at CR, and the SFS+SC+EXEC components were hosted in CRDC, OH, and SLC, respectively in three separate tests.

Despite having the latency-sensitive REN on the edge, Figure 10 shows the negative effects of network latency on average REN and frame times from distributing the EXEC and its directly dependent REN component on separate hosts. Average REN rates dropped as network latency and REN times increased. The CRDC, OH, and SLC hosting locations exhibited REN rates of 47Hz, 9Hz, and 3Hz, respectively.

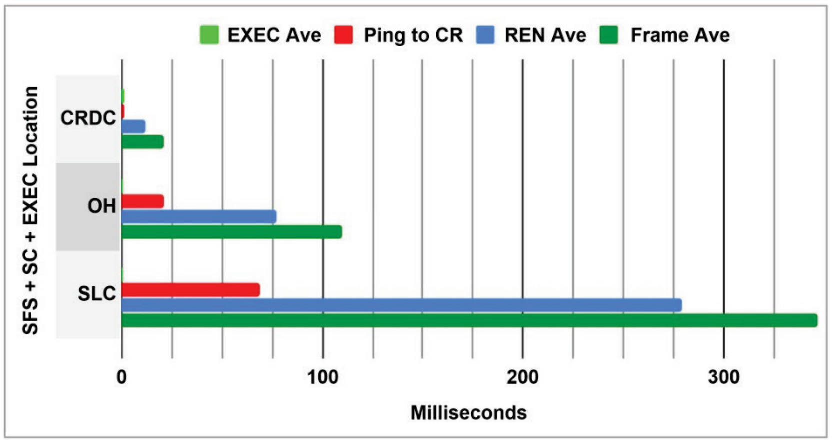

Figure 10. Effect on Render Times of Distributed EXEC and REN, with Dependencies

The average frame times for the OH and SLC cases were far more than expected given a naive belief that this distributed configuration would incur one round trip of network latency. In fact, the average frame time divided by the ping time suggests five round trips per frame for this test, and points out the handshaking that currently exists between the prototype EXEC and REN. This handshaking was observed in detailed frame timing analysis.

Experiment 4: Executive and Renderer on the Edge
The last experiment brought the latency-sensitive EXEC component and its equally latency-sensitive and dependent REN component to the edge, and hosted the SFS and SC components together (Figure 3) but at increasing ping times away from the edge. All tests for this component distribution model hosted the EXEC+REN components in CR; the increasing ping time locations of CRDC, OH, and SLC hosted the SFS+SC components.

The effect of increasing network latency on SC time is shown in Figure 11. The chart compares SC times of the non-distributed FVS case where all components are co-located against the SC times of the distributed configuration. The increase in SC times observed in the distributed cases is due to network transport of 12.8MB of scene data from the remote SFS+SC to the edge EXEC+REN. Given that the SC component is latency tolerant, the increase in SC time may be acceptable for some use cases.

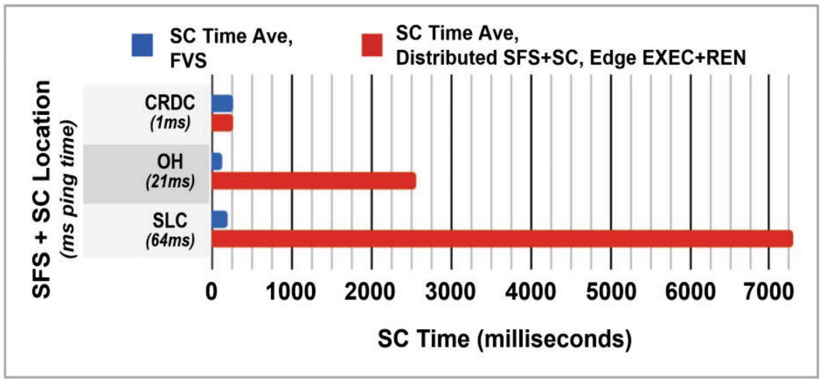

Figure 11. Effect of Network Latency on SC Time for Distributed SFS+SC and Edge EXEC+REN

This distribution model, however, provided a responsive user experience. The edge-hosted, latency intolerant EXEC and REN components averaged a rate of 60Hz regardless of ping time to the SFS+SC hosting location, and the round-trip user latency for all cases directly paralleled the average 16ms EXEC and REN times of the user's edge device. The prototype SC component ran on-demand, so the geometry and texture for the scene test location in Figure 6 was loaded once, then the eye point flown around the region. Since the remote SC component was not continually updating the edge EXEC+REN with new data, there may be unobserved negative effects to the 60Hz average EXEC and REN rates from additional threading and processing required to receive the scene data across the network from the remote SC.

CONCLUSIONS

The visual system needs to be decomposed to understand the latency, throughput, data use, and resource requirements of each major component. We showed that the per frame, multiple frame, and static categorization of components and operations each had different requirements, and components within these categories could be migrated between cloud and edge resources to optimize performance for a given use case.

Visual system components that are tightly and directly dependent on one another, such as the EXEC and REN, or SFS and SC, should be co-located to avoid excessive overhead. If there is a functional reason to distribute dependent components, their interface design must minimize handshaking. Public cloud hosting of SFS for ease of configuration management, and SC on a private data center is an example.

User latency for the prototype cloud-based FVS cases is currently dominated by encoding, decoding and network transport delays. Transport delays vary widely based on network topology, provider, virtual private network, user-to-cloud distance, and other factors. These can be improved by leveraging hardware-accelerated encoding and decoding, and optimizing network topology.

Shifting to a cloud-based computing model will require stringent provisioning of shared resources to provide the kind of performance and determinism guarantees users expect.

Different use cases lend themselves to different cloud-based architectures. Therefore, a flexible architecture that supports migrating visual system components to appropriate resources will facilitate adoption of the cloud for a broader range of use cases.

The value of using the cloud's massive compute, storage, and GPU resources seems high compared to using dedicated, on-premises equipment for visual systems, but its readiness is highly-dependent on use case requirements. As other industries trend toward hybrid cloud architectures, decentralization, and edge computing to solve their cloud performance and latency problems, tomorrow's visual system architectures must also consider these options for the same reasons. Visual systems will require flexibility to distribute their components across public cloud, private data center, and edge resources, exploiting the advantages of all to the overall benefit of the system and users, especially as those technologies, and the networks that connect them, evolve and advance in the future.

ACKNOWLEDGEMENTS

The author is indebted to Peter Bellows for his thorough reviews and keen suggestions that sharpened this paper. She also wishes to thank others who contributed their energy toward the paper's content and review, including Tadd Andrle, Layne Christensen, Carla Cropper, Paul French, Mike Genetti, Kevin Kovar, Ian McElhenny, Richard Rademaker, Larry Seppi, Don Seymour, Brad Southwick, and Jason Wenger.

REFERENCES

Kundra, V. (2011). Federal Cloud Computing Strategy. Retrieved from https://www.dhs.gov/sites/default/files/publications/digital-strategy/federal-cloud-computing-strategy.pdf

Morris, S. (2011). Cloud Types: Private, Public and Hybrid. Retrieved from https://www.asigra.com/blog/cloud-types-private-public-and-hybrid

Abacus Next (2017, January 5). What's the Difference between Public, Private, Hybrid, and Community Clouds? Retrieved from https://www.abacusnext.com/blog/whats-difference-between-public-private-hybrid-and-community-clouds

Butler, B. (2017, September 21). What is edge computing and how it's changing the network. Retrieved from https://www.networkworld.com/article/3224893/internet-of-things/what-is-edge-computing-and-how-it-s-changing-the-network.html

Grigorik, I. (2013). *High Performance Browser Networking.* O'Reilly Media. Retrieved from https://hpbn.co/primer-on-web-performance/#speed-performance-and-human-perception

NVIDIA (2018). Retrieved from https://shield.nvidia.com/support/geforce-now/system-requirements

Janakiram MSV. (2017, September 15). Demystifying Edge Computing -- Device Edge vs. Cloud Edge. Retrieved from https://www.forbes.com/sites/janakirammsv/2017/09/15/demystifying-edge-computing-device-edge-vs-cloud-edge/#7cf033053633

Baker, J. (2017, November 5). Edge Computing -- The New Frontier of the Web. Retrieved from https://hackernoon.com/edge-computing-a-beginners-guide-8976b6886481

Jenkins, J. C, Havig, P. R. (2015) An Evaluation of Latency in the F-35 Joint Strike Fighter Helmet Mounted Display. Retrieved from http://journals.sagepub.com/doi/pdf/10.1177/1541931215591005

Dukstein, G., Nielson, K., Dumanior, P. (2017). 3D Visualization for Point of Need and Cloud Based Training. Interservice/Industry Training, Simulation, and Education Conference.