Abstract: Cloud gaming provides game-on-demand (GoD) services over the Internet cloud. The goal is to achieve faster response time and higher QoS. The video game is rendered remotely on the game cloud and decoded on thin client devices such as tablet computer or smartphone. We design a game cloud with a virtualized cluster of CPU/GPU servers at USC GamePipe Laboratory. We enable interactive gaming by taking full advantage of the cloud and local resources for high quality of experience (QoE) gaming.

We report preliminary performance results on the game latency and frame rate. We find 109 ~ 131 ms latency in using the game cloud, which is 14% ~ 38% lower than 200 ms latency experienced on a thin local computer. Moreover, the frame rate from the cloud is 25% ~ 35% higher than that of using a client computer alone. Base on these findings, we anticipate game cloud to have a performance gain or QoS improvement of 14% ~ 38% over video gaming on a thin client device such as a smartphone or a tablet computer.

Keywords: Game cloud, virtual machine, video-on-demand, image compression, remote rendering, latency analysis, and game benchmarking.

1. Introduction

Cloud gaming posts an attractive domain for the entertainment industry. Game players do not have to purchase and update expensive gaming consoles such as Xbox360 and PS3 to play top-tier video games. The video game is rendered remotely on the game cloud and decoded on thin client device such as IPTV, tablet computer, and smartphones. After assessing some commercial game clouds, namely Gaikai [5], OnLive [13], and Otoy [14], we propose a GamePipe cloud to support interactive gaming, adaptively.

Our cloud platform provisions virtual machines (VMs) to control multi-party gaming operations. The game cloud distributes game tasks to VMs for frame encoding, 3D graphics rendering, artificial intelligence, physics simulation, animation and audio/video playback to minimize the response time. GamePipe cloud is built with a hybrid CPU/GPU cluster of multi-core CPU servers attached with an array of many-core GPU servers.

Our primary cloud design goal is to achieve interactive gaming by taking full advantage of computing resource on both the cloud and clients. Online gaming has evolved over the years. Cloud gaming has diverse applications in education, health care, and social networking etc. With widespread use of smartphones and tablet computers, gaming becomes much more popular in our daily life [2, 4, 6, 16].

For instance, the top 10 applications on Apple’s app store are related to gaming. A large share of popular social website like Facebook support playing social games among friends. The concept of using cloud resources for remote rendering of gameplay is illustrated in Figure 1. Emerging game cloud platforms also create new challenges to both game developers and publishers [1, 3, 10].

Figure 1. The concept of using a cloud to provide Game-as-a-Service (GaaS)
The higher is the quality-of-service (QoS) level in game graphics, physics simulation, and artificial intelligence, the higher is the demand of computing performance and reduced latency or response time. The popularity of some high-end games is constrained by limited graphics capability and battery life of mobile devices. In addition, game developers have to spend more effort to tune their games for different platforms and make tradeoff between quality and performance.

The most important measures of the QoS in online gaming are the downlink and uplink latencies and the frame rate in graphics processing over the cloud [2, 3]. Other important issues include the scalability over casual and serious games, monetization of the core games, and connectivity at the client end [6]. Cloud gaming demand both software-as-a-service (SaaS) and platform-as-a-service (PaaS) on the cloud [7].

This paper presents the architecture of the GamePipe cloud, the game VM designs and reports some preliminary results on a selected suite of gaming scenarios. In Section 2, we review some related work and identify our unique contributions in building the game cloud. The USC game cloud architecture is presented in Section 3. We characterize typical cloud gaming operations in Section 4.

We review various game VMs for implementing major gaming algorithms for frame encoding, 3D graphics rendering, animation, physics simulation, artificial intelligence and audio/video facilities. Then, we report the cloud experimental results about cloud latency and frame rate in Section 5. Finally, we summarize the major research findings and suggest further research directions.

## 2. Cloud Gaming and Our Approach

Cloud gaming is similar to video-on-demand (VoD) except game software is retrieved, executed on datacenter and streamed to thin client devices through the Internet. The core value of cloud gaming is to alleviate the players from resource management for video and graphics processing. However, game-on-demand (GoD) demands interactivity than VoD since control commands need to send back by players to the datacenter to update the game objects. Therefore, cloud gaming is very sensitive to the response latency.

With cloud gaming, players just pay for the time of playing multiple games and do not need to worry about minimum hardware requirements of each game. The mobile players can also demand high-quality video games on their handsets on the way. Today, the network bandwidth is capable to support cloud gaming online, but the latency is still a critical concern on the QoS in cloud gaming industry.

Moreover, cloud gaming can prevent software piracy, since game is serviced on demand after authorization, instead of as a packaged software product. As observed by Ojala and Tyrvainen [12], executing the games on a cloud server makes illegal copying practically impossible. Jarschel et al [8] use an emulation technique to evaluate the user-perceived quality-of-experience (QoE) of cloud gaming. They find that downstream packet loss and with a latency of 80 ms are perceivable in most fast-paced games.

In 2009, AMD proposed a supercomputer called Fusion Cloud to specifically handle gaming graphics [19]. Barboza, et al [1] have proposed a VM approach to provide GoD services on the cloud platform. They proposed to use three levels of managers for the cloud, hosts and clients. A cloud manager monitors the host managers’ activities on provisioning VMs for video streaming to clients.


New business models for cloud gaming were developed to face the rapid changes in SaaS for video gaming. Online game industry wants to use cloud over the Internet instead of using packaged CDs or DVDs. A game cloud provider licenses game source code from the studio or publishers. The cloud provider may modify the source code to enable games to be executed on its cloud platform.

The modifications include the user interface and networking protocols. For instance, a game originally developed for console has to include the driver to control the keyboard input on a PC. Most games are not optimized on the cloud platform. The cloud provider initially reaches the customers through VoD and set-top box suppliers or some middleware vendors.

With the competition of PC gaming market [2], cloud gaming companies transform their business model to directly reach customer through the network provider. In addition, a cloud gaming company also offers the SDK to facilitate the construction of game
for cloud platform. However, cloud gaming companies still have limited access to game source code.

Game deployed on the cloud is still based on the model of one virtual console per player. That is, the game is executed on one VM per player. Although this model can be scaled up or down when the number of online players increases or decreases, one VM is heavily loaded with data-intensive tasks while other VMs could be idle. Therefore, we need load balancing among the provisioned VMs. Other concerns may include tradeoff between latency and screen resolution.

In addition, client game devices may be limited in resources to interact with cloud. To make virtualized game engine more efficient on the cloud platform, we propose a new architecture of cloud gaming platform. To amend the aforementioned shortcomings in today’s game clouds, we choose the following approach to developing a cost-effective game cloud platform with assured QoS in practical applications.

(1). We propose a new cloud architecture largely using virtualized game engines. This cloud platform can distribute many games for parallel execution on multiple VMs, which will shorten the game processing latency and thus raise the QoS to many players concurrently.

(2). We develop front-end hosts to adaptively provision VMs for remote-rendering or remote-update to satisfy different levels of computing capabilities of user devices in playing the game over the cloud.

(3). Back-end VMs are developed for game graphics, animation, and audio are shared by all client players. This will improve the resource utilization and load balance in the game cloud.

3. Architecture of GamePipe Cloud

The architecture of the USC game cloud, namely the GamePipe cloud, is depicted in Fig. 2. This cloud is built with a cluster of CPU and GPU servers. Each CPU server has multicore processor, which are used to host the execution of many games simultaneously. These CPU servers are connected via an Ethernet to the network disk arrays and a workstation that hosts the VM manager.

The GPUs are attached to the CPU servers via PCIe busses. The GPUs have many-core architecture with a high degree of parallelism. These GPU are used for data-intensive parallel execution of game graphics, animation, and physics simulation.

The snapshot of the prototype game cloud, which is still under construction at USC, is shown in Fig. 3. The prototype cloud is built with three Intel QSSC-S4R servers, namely CPU server1, server2, and server3. Each CPU server has four 10-core Xeon E7-8870 CPUs running at 2.4 GHz with 64 GB memory. The data storage is on a separate iSCSI enabled network storage device connected via a Gbit Ethernet. Each server installs an OS of Windows Server 2008 R2 Datacenter edition.

To implement remote rendering, the system has one GPU server vCore Express 2070Q from NextIO. The GPU server includes four NVIDIA Tesla M2070Q GPU, each of which has 448 CUDA cores, 6 GB video memory, and 150 GB/s memory bandwidth. Each GPU supports up to 25 VMs. CPU server1 and server2 connect to GPU to host VM for graphics processing. The CPU server3 is used to host the VMs for CPU only processing.

![Figure 2. The architecture of a game cloud built with a cluster of CPU/GPU servers](image-url)
Based on above hardware organization, we propose to virtualize the platform for game cloud services as shown in Fig. 4. We apply the System Center 2012 from Microsoft to build up a private cloud. Each server use Hyper-V hypervisor to host VM such as Windows 7 Enterprise edition. All VMs on CPU server1 and server2 have RemoteFX enabled to communicate with GPU server.

All VMs are managed by the VM Manager on a standalone workstation in the LAN. Due to constraint of CPU and memory resources on the prototype cloud system, at most 50 VMs can be hosted on each CPU server for remote cloud services. The cloud architecture divides the VM’s work into front-end and back-end. Front-end VMs directly interact with clients while back-end VMs process game-related computing. As illustrated in Fig. 4, N clients are provisioned with N host VMs in the front-end to interact with.

In the backend, VMs are clustered based on types of functionality. Note that some VM clusters are used for graphics or physics simulations on GPUs. The rest are VM clusters deployed on the CPU servers for processing artificial intelligence, animation, audio and frame encoding.

To deal with different computing capability of client, different size of virtual cluster is provisioned in the back-end. For instance, a VM for remote-update is provisioned for client, which has sufficient local graphics processing capability. For extreme thin client, all computing are migrated to the cloud.
4. Interactive Cloud Gaming

All requests from the clients are handled by multiple host VMs. Table 1 summarizes provisioned VM for three different clients. The host manager provision remote-render VM, remote-update VM, or remote-update-render VM based on the computing capability of client. For instance, a player uses a very thin client machine with limited graphics and computing resources such as smartphones which will need both remote-update and remote-render. The thin client receives, decompress, and decode the frame. All heavy-lifting works are taken care of by the back-end VMs in the cloud.

On the other hand, if the customer uses a client machine such as tablet computer with rich graphics processing capability, the host VM will only provision remote-update VM migrating graphics computing to client, since the local graphics computing resource is fast and should not be wasted. For a customer with a PC with limited graphics but rich computing resources, only remote rendering is necessary.

In summary, the host VM adaptively provision different types of VM to satisfy different client requirements. At the back-end, the architecture distributes the tasks of game engine into multiple VMs to work in parallel. Some VMs are dedicated to graphics rendering. Some VMs process physics or AI simulation during the gameplay. Based on the different types of tasks, back-end VMs may or may not access GPU server.

For instance, graphics rendering and physics simulation can be accelerated by GPUs while AI simulation can be executed on the CPU server. The computing results from multiple back-end VMs are sent to the front-end VMs through MPI. The front-end VMs combine the distributed results.

Figure 5 shows all steps involved in a complete gameplay cycle. After a player issue a gameplay request to the cloud, the cloud must check with the archive in the database. The host VM evaluates the computing capability of the user devices. This examines the client CPU/GPU power, memory capacity, and network bandwidth. For sufficient resources in a rich client without graphics, a few VMs are provisioned for remote render.

With a thin client with limited resource, more VMs may be provisioned for performing remote render and update functions as shown on the lower right side of the flowchart. When all VM workers finish their assigned tasks, the cloud wakes up a specific VM for encoding and sending back frames to the players. Cycle repeats itself until the entire game is executed or player quits the game. The sequence is summarized below in 8 steps.

1. Client login using different devices.
2. Login VM check DB for authorization.
3. Host VM is provisioned for each client.
4. Host VM evaluates user's computing capability.
5. Host VM notify VM scheduler to provision remote-render, remote-update or remote-render-update VM.
6. Multiple VMs in the back-end are provisioned to process the game session.
7. Client issue end of gameplay.
8. Host manager save game status and sleep corresponding VM in the back-end.

Table 1. Worker VMs Deployed to Perform Various Game Functions for Three Gaming Scenarios.

<table>
<thead>
<tr>
<th>Game Function</th>
<th>Remote update</th>
<th>Remote rendering</th>
<th>Remote update/ rendering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target user device</td>
<td>Tablet with thin CPU</td>
<td>PC with thin GPU</td>
<td>Smartphone with thin CPU &amp; GPU</td>
</tr>
<tr>
<td>Compress or Decompress</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Graphics</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Physical simulation</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine intelligence</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Animation</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio/video playback</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. Cloud Latency and Frame Rate

We conducted three experiments for remote-update, remote-render, and remote-update-render using GamePipe game engine. The remote-update implementation is a client server based design. The game objects are created in the VM on the server and replicated on all the clients that connect to the VM. Figure 6 shows a typical screenshot of the “Destroy the Castle” game executed on the USC GamePipe cloud.

The VM on server maintain multiple clients by assigning unique GUID as key for each object. The update loop is continuously transmitting attributes of objects such as position, orientation or animation to clients. However, all rendering is executed on client.

For experiment of remote-update-render, the VM on the CPU server runs a game engine and update all states of game objects. Moreover, the VM communicates with GPU server for rendering. The rendered frames are transferred through UDP protocol to clients. Meanwhile, the client sent player input to server and display the frame received from server. For remote-render, only graphics are rendered on the cloud.

The measured delays in ms for remote-update, remote-render, and remote-update-render cases are shown in Fig.7. The round-trip latency on LAN is about 80 ms and display of 60Hz is used compared with Pohl’s setup [15]. The mouse and keyboard connected USB are polled at 125Hz. After the player generates input signal using keyboard and mouse, there is delay of processing and upload the data to server.

The delay of the uplink is 40 ms ~ 57 ms, mainly attributed to the local update overhead. The processing delay of the cloud servers varies with connected user devices. Remote render takes 17 ms to process compared with remote-update taking 12 ms. The encoded data or frame is sent back to the client. The thin client needs to decode and display the data to screen, which takes 57 ms ~ 102 ms, mainly attributed to the monitor delay.

To sum up, we find 109 ms ~ 131 ms latency in using the game cloud, which is 14% ~ 38% lower than the 200 ms latency experienced on a thin user computer. The cloud latency is mainly resulted from the upload and download links. The actual cloud processing time is only 11% ~ 23% of the total latency experienced.

The network latency on a thin notebook is about the same as the cloud. It is attributed more to upload delay than download delay. The game processing time on the notebook is about 8 times longer (about 200 ms) than the cloud processing time (25 ms). This is caused by the fact that cloud employs a server cluster to do parallel execution, while the end user computer executes the game program sequentially. Cloud gaming show strength in reducing the game execution time using many VMs in parallel.
Figure 7. Response latency in three remote cloud gaming experiments using the USC GamePipe cloud.

We compare in Fig.8 the frame rate of playing games over a local thin client with that experienced on the game cloud platform. The y-axis shows the frame rate in frames per second. The x-axis shows 3 game implementations, ranging from serial to parallel using Open MP and TBB libraries. In all cases, the cloud (white bars) performs better than the thin client computer (black bars).

Figure 8. Frame rate for playing games on a thin client computer compared with that on a game cloud like GamePipe at USC.

In summary, the frame rate of using the game cloud is 25% ~ 35% higher than that of using a thin client machine to play the same set of games locally. The thin client is a notebook with Intel Core 2 Duo 2.0 GHz CPU, 2GB memory, and NVIDIA GeForce 8400M GPU. The screen resolution is 1280x800 pixels. The parallel version of game has 35% higher frame rate than the serial version on the average.

6. Conclusions

The USC GamePipe cloud prototype design and operational experiences are summarized below in 5 important technical observations. We also discuss potential impacts to the game cloud industry. Some of these research findings echo what were asserted in [2, 4, 6, 16] with further evidences.

(1). Gaming Latency and Frame Rate: These are two crucial factors to assess the QoS of a game cloud. Our latency results (109 ms ~ 131 ms) are close to what can be tolerated in real-time for online gaming. The latency is mainly attributed to the network delays. The cloud latency and frame rate are 14% ~ 38% better than those experienced on a thin client computer. Further study will focus on mobile devices.

(2). Caching and Buffering Effects: One could consider caching or using buffering techniques to eliminate unnecessary uplink or downlink latencies to further improve the game cloud performance. The arrival rate of the frames must match with playback rate to avoid jitters [11].

(3). Scalability over Game Sizes: The game cloud must be designed to scale well with both casual games and data-intensive games. Further studies are needed to address the application scalability over games that require processing much greater dataset.

(4). New Game Cloud Business Models: The game cloud may demand a new business model. The traditional retail model may not be profitable. Monetization may demand a new model to release the core games in episodes to enhance the profit margins for game providers.

(5). Redundancy Elimination among Multi-Players: Game processing in the cloud is repeated for many players and thus results in resource waste at the datacenter. One could eliminate the redundant game processing among players to improve platform efficiency. In addition, peer-to-peer support could be also considered for future game clouds [17].
Acknowledgements

Our game cloud was built and tested at the USC GamePipe Laboratory. We thank the sponsorship from Intel, NVIDIA and Microsoft in providing hardware and software support.

We want to thank Prof. Michael Zyda and students in two USC game-related courses: EE452 Game Hardware Architecture and CSCI522 Game Engine Development, for their assistance to build the prototype cloud and perform game development.

Kai Hwang wants to acknowledge the support by China’s basic research project, under the 973 Grant No. 2011CB302505, on cloud software tools and application environment development at Tsinghua University, Beijing, China.

References


