CloudFog: Towards High Quality of Experience in Cloud Gaming

Yuhua Lin and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University, Clemson, South Carolina 29634
Email: {yuhual, shenh}@clemson.edu

***Abstract*—With the increasing popularity of Massively Multiplayer Online Game (MMOG) and fast growth of mobile gaming, cloud gaming exhibits great promises over the conventional MMOG gaming model as it frees players from the requirement of hardware and game installation on their local computers. However, as the graphics rendering is offloaded to the cloud, the data transmission between the end-users and the cloud significantly increases the response latency and limits the user coverage, thus preventing cloud gaming to achieve high user Quality of Experience (QoE). To solve this problem, previous research suggested deploying more datacenters, but it comes at a prohibitive cost. We propose a lightweight system called *CloudFog*, which incorporates "fog" consisting of supernodes that are responsible for rendering game videos and streaming them to their nearby players. Fog enables the cloud to be only responsible for the intensive game state computation and sending update information to supernodes, which significantly reduce the traffic hence the latency and bandwidth consumption. To further enhance QoE, we propose the receiver-driven encoding rate adaptation strategy to increase the playback continuity and the deadline-driven sender buffer scheduling strategy to ensure that the segments arrive at the players within their response latency. Experimental results from PeerSim and PlanetLab show the effectiveness and efficiency of CloudFog and our individual strategies in increasing user coverage, reducing response latency and bandwidth consumption.**

***Keywords*-Cloud gaming; P2P network; Online gaming; Quality of experience**

I. Introduction

Massively Multiplayer Online Game (MMOG) (e.g., World of Warcraft, Second Life) allows users to inhabit in the same virtual world and interact with each other. It is characterized by a huge number of simultaneous players. Indeed, its customer base has grown from a few thousands to tens of millions of players. At the height of its popularity, World of Warcraft currently has over 12 million users. Generally, MMOG uses the centralized client/server infrastructure, in which players need to install games, receive the game information from the servers, update game status and render new game videos [1]. MMOG requires players to have sufficiently powerful computers, which excludes users with thin clients such as tablets and smartphones. Buying and maintaining servers to support the tremendous number of players is cost-prohibitive to the game service provider. Nowadays, the number of smartphone users has been increasing rapidly and mobile gaming is also seeing the fastest growth among the gaming models [2]. The number of smartphones in use worldwide reached one billion during the third quarter of 2012, and will increase by another billion by 2015 [3]. Thus, thin-client MMOG is an inevitable trend of current MMOG and a cost-efficient system to support thin-client MMOG is desirable for game service providers.

Cloud gaming, as a flourishing gaming model, is a solution for thin-client MMOG, which frees players (we use players, clients, nodes and users interchangeably in this paper) from the requirement of hardware and game installation on their local computers. Nowadays, the cloud gaming is becoming a flourishing gaming model, with OnLive [4] and Gaikai [5] as two pioneers in cloud gaming. In cloud gaming, games are stored and run on remote servers, and game videos are streamed to end-users through broadband Internet connections. Cloud gaming also saves the cost of game service providers. They can buy cloud resource based on the actual demands in the large-scale system. Also, game service providers do not have to develop multiple versions of the same game to meet different operating systems (e.g., Linux, Windows, Mac), and spend money on software piracy protection.

Mobile gaming is seeing the fastest growth [2] in the gaming area. The advantages of cloud gaming makes it a very promising model to cater to the dramatically rapid growth of MMOG and online mobile gaming considering their very large user scale and thin clients. Though the advantages of cloud gaming makes it a very promising model to cater to thin-client MMOG, it currently faces severe challenges (i.e., latency, network connection, user coverage and bandwidth cost) that prevent it from becoming a leading gaming model. First, response latency is a critical factor in user quality of experience (QoE). By offloading computation to a remote host, cloud gaming suffers from long *response latency*; the delay in sending the user action information and game video between the end-user and the cloud. Second, cloud gaming services post a strict requirements of high-speed network connection for a relatively high constant downlink bandwidth (e.g., 5Mbit/s recommended by OnLive). Third, the shortage of datacenters limits user coverage. Players begin to notice a response delay of 100ms [6]; 20ms attributed to playout delay on client side and processing delay on the cloud, 80ms attributed to the network latency. The playout delay of a client includes the time to send action information, receive and play the

game video. Choy *et al.* [7] found that Amazon's EC2 (with 13 datacenters) can provide a median latency of 80ms or less to only fewer than 70% of their 2500 tested end-users in the US. They also found that substantial increase in the total number of datacenters is required to significantly increase user coverage. Existing cloud infrastructure is not sufficient for hosting cloud gaming, as a sizeable portion of the population would experience significantly degraded QoE. Fourth, besides server time, bandwidth costs represent a major expense when renting on-demand resources. An average traffic of 27TB per 12 hours leads to about $130k monthly fee for bandwidth with Amazon EC2's price (i.e., $0.085 per GB) [8]. Considering the MMOG's huge user scale, these costs can significantly affect the feasibility of thin-client MMOG [9] on the cloud.

The great promises of cloud gaming and the obstacles it faces motivate us to explore approaches to efficiently handle the challenges. Though previous study suggested deploying more datacenters [6], building and maintaining a large number of datacenters is cost-prohibitive. In this paper, we propose a lightweight system called *CloudFog*. We introduce a concept called "fog", formed by powerful supernodes, that are close to end-users and connect them with the cloud. Considering that desktop systems, being idle around 95% of the time [10], are underutilized in most organizations, the supernodes can be from these idle resources or from players' computers. In CloudFog, the intensive computation [1] of the new game state of the virtual world is conducted in the cloud. The cloud sends update messages to supernodes, the supernodes update the virtual world, render game videos for different players and stream videos to the players. Thus, users without high speed network connection to cloud or out of the coverage of the cloud can be supported by nearby supernodes, and the cloud does not need to transmit entire game videos to far-away users. This strategy can increase user coverage, shorten response latency, ensure relatively high-speed network connection for high QoE and reduce bandwidth cost. Specifically, CloudFog incorporates the following strategies to handle the challenges and enhance QoE. Strategies (2) and (3) are based on the phenomenon that different games have different tolerance on packet loss rate and response delay [11].

(1) **Fog-assisted cloud gaming infrastructure.** We leverage the hardware and bandwidth capacity of some idle machines from players and organizations, and deploy them as supernodes. These supernodes constitutes the "fog", which are responsible to stream game videos for nearby players.
(2) **Receiver-driven encoding rate adaptation.** In order to ensure the playback continuity even in network congestion, when a supernode streams a game video to a player, it adaptively changes the encoding rate of the video based on the segment size in the player's buffer according to the game's tolerance on delay and packet loss.
(3) **Deadline-driven sender buffer scheduling.** To meet response latency requirement of each game, supernodes give higher priority to lower delay-tolerant game videos to send out, and drop packets from different game videos based on their packet loss tolerant degree.

This is the first work that uses lightweight approaches to handle the aforementioned challenges for cloud gaming to support thin-client MMOG. The remainder of the paper is organized as follows. Section II presents an overview on the related work. Section III describes the detailed design of CloudFog. The performance evaluation is presented in Section IV. Section V concludes this paper with remarks on our future work.

II. Related work

MMOG on the client-server architectures has gained much attention in the research communities in recent years. Common approaches of MMOG divide the virtual environment into regions and assign each region to different servers [1]. Bezerra *et al.* [12] proposed a kd-tree mechanism to partition the game environment into regions, and perform load balancing among multiple servers based on the distribution of avatars in the virtual world. Many works proposed to leverage the bandwidth contribution of peer-to-peer (P2P) networks to reduce server load [13]. Ahmad *et al.* [13] presents a P2P live video system to help players share screen-captured video of their games. Chen *et al.* [14] proposed a content-oriented pub/sub system that exploits the network condition and end-systems to enable efficient player management and decentralized information dissemination. These P2P and information dissemination methods cannot be directly applied to the context of cloud gaming, in which each player receives its own game video that cannot be shared with other players. Also, the players with thin clients may not be able to conduct rendering, computation and storage [7], which are offloaded to the cloud.

Previous works developed different cloud gaming systems. GamingAnywhere [15] is the first open cloud gaming system with high extensibility, portability, and reconfigurability. Zhao *et al.* [16] designed a game cloud with a visualized cluster of CPU/GPU servers to reduce game latency of thin computers. Wang *et al.* [17] proposed to shift the burden of executing gaming engine from mobile devices to cloud servers, and developed a mobile gaming user experience model to characterize user experience. Hemmati *et al.* [18] presented a content adaptation encoding scheme, in which only the most important objects from the perspective of the player's activity are encoded in the scene and irrelevant or less important objects are omitted. EdgeCloud [19] augments the cloud infrastructure with a number of servers with specialized resources located near end-users to increase user coverage, these servers are responsible for

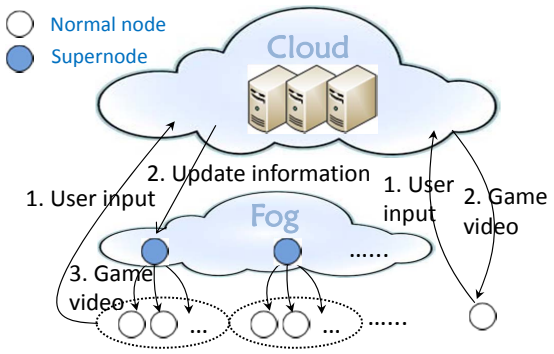


Figure 1: Fog-assisted cloud gaming infrastructure.

computing new game state and rendering game video for players. The difference between EdgeCloud and CloudFog lies in the responsibility of newly added servers. In EdgeCloud, the addition of a small number of servers are used to store and compute game status and render new game videos; while in CloudFog, the storage and computation are carried out on the cloud, servers are only used to render new game videos and stream them back to the players. As the rendering work does not require high hardware configuration, given the same amount of revenue, *CloudFog* can deploy more servers than EdgeCloud by using proper incentives to motivate players or organizations to contribute their spare machines.

In spite of the previous research efforts on cloud gaming, except deploying more datacenters which is costly, no other approaches have been proposed to handle its critical challenges. We propose light-weight strategies to tackle the challenges to support thin-client MMOG.

III. System design of CloudFog

A. Fog-assisted Cloud Gaming Infrastructure

Previous studies [6], [7] revealed that the uploading from the players to the cloud does not seriously affect the response latency, and downstream latency is an important factor for QoE [6], which is affected by the game video streaming rate. Thus, we aim to reduce the downstream latency by reducing the traffic transmitted from the cloud. In our design, game videos are streamed from nearby supernodes to players, instead of from remote game servers. As the computation of a virtual world for MMOG has a very high demand on server capacities [1], cloud is responsible for this task. Figure 1 shows our fog-assisted cloud gaming infrastructure. The fog is formed by supernodes, and normal nodes are connecting to their nearby supernodes. The normal nodes that cannot find nearby supernodes directly connect to the cloud.

We use n_i to denote a normal node, and sn_j to denote a supernode in the system. When each supernode is initially deployed, it is pre-installed with the *game client*. During the game playing, when node n_i makes an action (e.g., launching a strike or moving to a new place), this information is sent to the cloud server. The server collects action information from all involved players in the system and performs the computation of the new game state of the virtual world (including the new shape and position of objects and states of avatars). The cloud then sends the update information to the supernode of n_i (sn_j), which updates its virtual world accordingly. sn_j then renders game video for n_i based on $n_i's$ viewing position and angle. sn_j finally encodes the game video and stream it to n_i. As a player is close to its supernode in network distance, and the traffic from the cloud is significantly reduced, so the game video transmission delay is much shorter than that of downloading game video directly from the cloud as in the current cloud computing systems.

1) Requirements and Incentives for Supernodes: Rendering game video is relatively less hardware demanding than computation and communication in MMOG [19]; most modern computers with discrete graphics cards are sufficient to meet the rendering requirement. The nodes with sufficient hardware are chosen as supernodes. As shown in [10], most desktop systems are idle around 95% of the time in most organizations. So the supernodes can be contributed by different organizations that have idle computer resources, and game players that have powerful computers can also be selected as supernode candidates. Besides, game service providers can deploy their own supernodes by placing servers in different areas to support players. A game client of MMOG usually takes about 5-6GB, and it is pre-installed in the supernode. The supernodes are required to be: 1) reliable, as malicious supernodes may distribute spam or virus that may degrade player experience or harm players' machines; and 2) stable, supernodes need to provide stable support and notify the central server of game service providers before leaving the system. To satisfy these requirements, organizations and individual players need to provide credentials to game service providers, game service providers will verify the information of supernode contributors and have contracts with them. Contributing a machine as a supernode generates costs of running the machine (e.g., electricity and maintenance costs). Therefore, to incentivize other organizations and players to contribute supernodes, an incentive mechanism is needed to reward supernodes based on the amount of upload bandwidth they contribute. The reward can be in the form of real money or virtual money for online games, and we use c_s to denote the reward for each bandwidth unit contributed by a supernode. An organization or a player considers to contribute a supernode only when it brings about certain profit, which is calculated by subtracting its running costs from its earned rewards. We use $P_s(j)$ to denote the profit gained by supernode sn_j:

$$P_s(j) = c_s \times c_j \times u_j - cost_j, \quad (1)$$

where c_j represents sn_j's upload capacity, u_j denotes sn_j's bandwidth utilization, and $cost_j$ denotes the cost paid by sn_j's contributor in the same unit of c_s. $P_s(j)$ quantifies the profit of contributing a supernode. Contributing a supernode is lucrative when $P_s(j)$ is greater than a certain threshold (different contributors set their own thresholds based on their expectations on profits). Then, the supernode's owner is motivated to contribute this supernode. We will evaluate the effectiveness of this incentive mechanism in Section IV.

2) Economic Benefits for Game Service Providers: The game service provider needs to guarantee that the money spent on rewarding supernodes is smaller than the bandwidth costs saved by the contribution of supernodes. We use $N(t)$ to denote the number of existing users at time t. For simplicity, we omit t in the notations. Given the streaming rate of game video R, the total system demand for bandwidth equals $N \times R$. Suppose there are m supernodes, each having c_j upload capacity with utilization u_j. Then, supernode bandwidth contribution equals $B_s = \sum_{j=1}^{m} c_j \times u_j$. We use Λ to denote the bandwidth usage for the cloud to send update information to one supernode, and use n to denote the number of users that supernodes support. Then, in CloudFog, the bandwidth consumption for one player action for nodes connecting to supernodes equals $\Lambda \times m$, and that for users directly connecting to the cloud equals $(N - n)R$. The bandwidth reduction (B_r^-) of CloudFog compared to current cloud computing system equals:

$$\begin{aligned} B_r^- &= N \times R - \Lambda \times m - (N - n)R \\ &= n \times R - \Lambda \times m \end{aligned} \tag{2}$$

Suppose c_c is the revenue gained by saving each server bandwidth unit, the goal of the game service provider is to maximize the saved cost by leveraging supernode bandwidth contribution, which can be formulated as below.

$$\begin{aligned} C_g &= \max(c_c \times B_r^- - c_s \times B_s) \\ &= \max\{c_c[n \times R - \Lambda \times m] - c_s \times B_s\} \end{aligned} \tag{3}$$

$$s.t. \quad \sum_{j=1}^{m} c_j \times u_j \geq n \times R \tag{4}$$

$$u_j \leq 1, \ \forall j \in \{1, 2, ...m\} \tag{5}$$

Equation (4) guarantees that the total supernode bandwidth contribution must reach the required node support bandwidth, while Equation (5) restricts the utilization of a supernode's upload bandwidth within its bandwidth capacity. In Equation (3), we see that given a specific number of n (*i.e.* the coverage of normal nodes is determined), saved cost C_g increases when m decreases; that is, a smaller number of supernodes lead to higher cost saving. For the game service provider, it should consider the pay and gain before deploying a supernode. Suppose a new supernode sn_j is deployed in an area; as a result, the coverage of players supported by supernodes is increased by ν new players. We use $G_s(j)$ to denote the game service provider's revenue gain by deploying sn_j, and $G_s(j)$ is estimated by:

$$G_s(j) = c_c[\nu \times R - \Lambda] - c_s \times c_j \times u_j. \tag{6}$$

If $G_s(j) > 0$, the cost of deploying supernode sn_j is surpassed by the benefit of bandwidth saved from the ν new players supported by sn_j. After supernodes are determined, we need to map players to their closest supernodes. We present the details in the next section.

3) Supernode Assignment: We use C_i to denote supernode sn_j's capacity, which is defined as the maximum number of normal nodes sn_j can support. The cloud stores the information of supernodes in the system in a table including their IP addresses and available capacities. Since node locations and coordinates can be determined by IP addresses [20], [21], the distance between a user and supernode can be calculated using the coordinate information. When a player joins the system, the cloud finds its physically close supernodes by referring to the table and calculating the distance between the player and all supernode candidates. Then the close supernodes of a user can be located based on their IP addresses.

After newly joined node n_i receives its close supernode candidates from the cloud, it tests the transmission delay to all of them, then removes candidates with transmission delay greater than its threshold L_{max_i}. The node can measure its response latency requirement based on the genre of its game [22], and then determines its L_{max_i}accordingly. This threshold is used to ensure that its supernode is capable of providing quick streaming support to it. Node n_i then chooses the supernode with the shortest transmission delay and available capacity as its supernode, and records other supernodes as its backups. If n_i cannot find qualified supernode measured by L_{max_i} and C_i, n_i directly connects to the cloud.

B. Receiver-driven Encoding Rate Adaptation

A player stores its received segments into its buffer while playing the game video. To guarantee the continuity of the video playback, the player needs to continuously fetch segments from the buffer and play. Game video bitrate affects the number of video segments received by a player during a unit time period, hence the player's playback continuity. Thus, we can adjust game video bitrate based on the size of buffered segments. The game video can be encoded to different bitrates based on the requirements on pixel size (resolution), hence the video quality level. A video segment with a higher quality level (i.e., a higher bitrate) leads to longer transmission latency. As shown in Figure 2, 500kbps corresponds to 384x216 resolution, and such a segment leads to 50ms latency. We use q_i ($i \in [1, ..., Q]$)

Quality level	Video resolution	Video bitrate	Latency requirement	Latency tolerance degree
5	1280×720	1800 kbps	110 ms	1
4	720×486	1200 kbps	90 ms	0.9
3	640×480	800 kbps	70 ms	0.8
2	384×216	500 kbps	50 ms	0.7
1	288×216	300 kbps	30 ms	0.6

Figure 2: Video parameters for different quality levels.

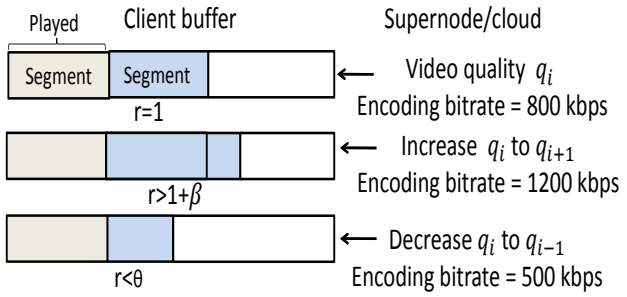


Figure 3: Receiver-driven encoding rate adaptation.

to denote the video quality for quality level i and use b_{q_i} to denote the corresponding bitrate. Different genres of games have different requirements on response latency [11]. Based on Figure 2, if a game video has a latency requirement of 90ms, the supernode should use 1200kbps encoding bitrate, corresponding to a quality level of 4. To reduce the latency of the game video, the supernode can choose encoding bitrates corresponding to quality level lower than 4; that is, sacrificing quality for lower latency. Due to unexpected network condition (e.g., network congestion), packets may actually be transmitted at a lower speed. In this case, a game video can reduce video quality (decrease bitrate of the video in encoding) in order to reach its latency requirement.

We aim to ensure that the playback rate is always lower than or equal to the segment downloading rate. When this condition cannot be satisfied, the video quality needs to be reduced by one level. When the size of buffered video at current quality level q_i is expected to reach the size of buffered video at quality level q_{i+1} (i.e., the downloading rate is faster than the playback rate), the current encoding bitrate b_{q_i} can be increased to $b_{q_{i+1}}$ to increase the video quality to q_{i+1}. Below, we explain the details of the adjustment operation. The estimated size of the video buffered at time t_k (denoted by $s(t_k)$) is calculated by:

$$s(t_k) = s(t_{k-1}) + (t_k - t_{k-1})(d(t_k) - b_p(t_k)), \quad (7)$$

where $d(t_k)$ and $b_p(t_k)$ denotes the downloading rate and video playback rate at time t_k. We use r to denote the number of segments in the buffer:

$$r = \frac{s(t_k)}{\tau} = \frac{s(t_{k-1}) + (t_k - t_{k-1})(d(t_k) - b_q(t_k))}{\tau}, \quad (8)$$

where τ denotes video segment size. If

$$r > 1 + \beta, \quad (9)$$

the video bitrate adjusts up. β is an adjust-up factor,and

$$\beta = max\{(b_{q_{i+1}} - b_{q_i})/b_{q_i}, \forall i \in [1, 2, ..., Q]\}. \quad (10)$$

β guarantees that the size of the buffered segments reaches that of the incremented quality level. When the video bitrate adjusts up, the user will not suffer from playback delay during the game. The adjust-down operation is performed if

$$r < \theta \ (\theta \leq 1), \quad (11)$$

where θ denotes adjust-down threshold. Formula (11) enables to proactively adjust down video bitrate to ensure the playback continuity in network congestion, in which the segment transmission time is typically much longer than usual. In order to prevent the fluctuation of the video bitrate for a client, the client can conduct the calculations of r for a number of times consecutively. The video bitrate is adjusted only when all results satisfy Formula (9) or Formula (11). Figure 3 shows an example of the encoding rate adaptation. When $r > 1 + \beta$ for several consecutive estimations, the supernode increases the video encoding quality by one level; from 800kbps to 1200kbps for the player. When $r < \theta$, the supernode decreases the video quality by one level; from 800kbps to 500kbps.

Different games have different latency-tolerant degree [11]. We consider this property to further enhancing the probability of meeting the response latency requirement for different games. Specifically, we require higher latency-sensitive games to have larger buffered video size for the encoding rate adjustment. We use $\rho \in [0, 1]$ to denote the latency tolerance degree; higher ρ means higher latency tolerance degree. We then change Formula (9) to $r > (1+\beta)/\rho$ and change Formula (11) to $r < \theta/\rho$ for triggering encoding bitrate adjustment. As a result, latency-sensitive (lower latency-tolerant) games have a higher r threshold while latency-tolerant games have a lower threshold for adjusting the encoding bitrate.

C. Deadline-driven Sender Buffer Scheduling

After a supernode encodes the game video, it puts the video segments into its buffer to send to its supported normal nodes. Each supernode has a single queuing buffer to send out video segments [23]. In a supernode's buffer, the segments for different game videos have different packet loss tolerance and delay tolerance degrees [11]. We take advantage of this property to improve overall QoE of players across different games. Basically, segments from packet loss tolerant game can drop packets while still meet their packet

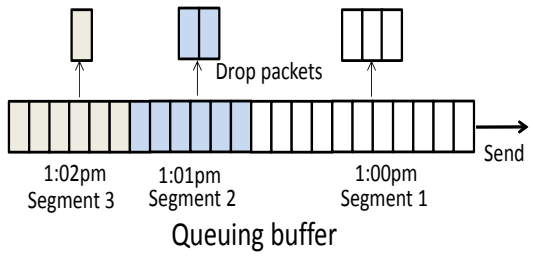


Figure 4: Deadline-driven buffer scheduling.

loss rate requirements; and segments from delay tolerant game can delay transmission while still meet their latency requirements. For video segment i, we use $\tilde{L}_{t_i}$ and $\tilde{L}_{r_i}$ to denote its packet loss tolerance rate and response latency requirement of its game. $\tilde{L}_{r_i}$ can be evaluated based the genre of game [22]. The expected arrival time for segment i is calculated by $t_{a_i} = t_{m_i} + \tilde{L}_{r_i}$, where t_{m_i} denotes the time that the player makes an action. When a new segment is generated, the supernode inserts it into its queuing buffer in ascending order of the expected arrival times; segments with earlier arrival times have higher priorities to send out earlier, as shown in Figure 4.

After a segment is put in the buffer, the supernode estimates the arrival times of this segment and its succeeding segments. If the estimated arrival time is later than its expected arrival time, the supernode drops some packets from the preceding segments based on their packet loss tolerance rates in order to meet latency requirement. The supernode first estimates the response latency L_{r_i} for segment i:

$$L_{r_i} = l_{r_i} + l_{s_i} + l_{q_i} + l_{t_i} + l_{p_i}, \tag{12}$$

where l_{r_i} is server receiving delay from a player makes an action until its supernode receives its game information from the cloud; l_{s_i} is server processing time for a supernode to render game videos; l_{q_i} is queuing delay from the time when the video segment enters the buffer until it leaves the buffer; l_{t_i} is transmission time from the first packet until the last packet of segment i has left the supernode; and l_{p_i} is the propagation delay of transmitting the video segment from the supernode to the player. l_{r_i} and l_{s_i} are known. l_{q_i}, l_{t_i} and l_{p_i} can be estimated as below. Suppose np_i is the total size of segments preceding segment i, then $l_{q_i} = np_i/\lambda_r$, where λ_r is bandwidth of the supernode. $l_{t_i} = s_i/\lambda_r$, where s_i is the size of the segment. Different game videos are sent to different players, and the players have different distances from the supernode. The supernode records the propagation delay of m recently sent packets for each player, and uses their average value to estimate the propagation delay of segment i in Equation (13):

$$l_{p_i} = (l_{p_1} + ... + l_{p_m})/m \tag{13}$$

If $L_{r_i} > \tilde{L}_{r_i}$, segment i is likely to be delayed, and user who needs this segment would suffer from response delay during the game. In this case, the supernode drops packets from segment i and its preceding segments. The number of packets to drop (denoted by D_i) is calculated by: $D_i = (L_{r_i} - \tilde{L}_{r_i})/\sigma$, where σ is the average amount of latency reduced by dropping one packet in the buffer. The number of packets needed to drop for each segment is determined based on its packet loss tolerance rate. Also, in order to prevent the preceding segments from dropping excessive number of packets, we apply an exponential decay [24] factor $\phi_i \in [0, 1]$ for segment i. ϕ_i is initialized to 1 and decreases as time elapses: $\phi_i = e^{-\lambda t_i}$, where t_i is the time period that segment i waits in the queue. We then use d_k to denote the number of packets needed to drop in each segment k.

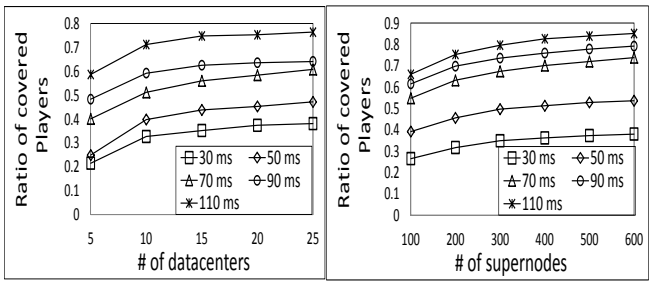
$$d_k = \frac{\tilde{L}_{t_k} \times \phi_k}{\sum_{j=0}^{k} \tilde{L}_{t_j} \times \phi_j} \times D_i \tag{14}$$

As shown in Figure 4, suppose 6 packets need to be dropped in order to make segment 3 meet its latency requirement. Segments 1, 2, 3 have packet loss tolerance rates $L_{t_1} = 0.6$, L_{t_2}=0.2, $L_{t_3} = 0.5$, and decay factors $\phi_1 = 0.5$, $\phi_2 = 0.1$, $\phi_3 = 0.2$, respectively. Based on Equation (14), the numbers of dropped packets for different segments are calculated as $d_1 = 3$, $d_2 = 2$ and $d_3 = 1$, respectively.

IV. Performance Evaluation

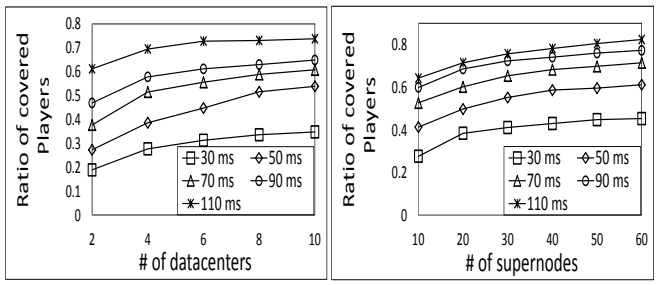
We conducted experiments on the PeerSim [25] simulator and the PlanetLab [26] real-world testbed to evaluate the performance of CloudFog in comparison with other systems. We measured the performance in response latency, playback continuity and user coverage. *Basic CloudFog (CloudFog/B)* denotes the fog-assisted cloud gaming infrastructure without applying our proposed strategies; *Advanced CloudFog (CloudFog/A)* denotes our system with all proposed strategies. We compared *CloudFog* with the current cloud gaming model [15] (denoted by *Cloud*) and *EdgeCloud* [19]. *EdgeCloud* deploys a number of powerful servers to increase user coverage. The difference between *EdgeCloud* and *CloudFog* lies in the responsibility of newly added servers. *EdgeCloud* simply adds powerful servers to takeover all the cloud's tasks (including storing and computing game status and rendering new game videos), while in *CloudFog*, the supernodes only need to receive updates from the cloud to render new game videos and stream them to the players. A user is covered by datacenter if the response latency is no more than the latency requirement of the user's game. The default number of main datacenters is 5 and 2 for all systems in simulation and PlanetLab, respectively. *EdgeCloud* has additionally 45 and 8 randomly distributed servers in simulation and PlanetLab, respectively. Other default settings are: $\theta = 0.5$, $\lambda = 1$, $h_1 = 100$ and $h_2 = 10$.

In the simulation, there were 10,000 game players (including online and offline players), 10% of which have



(a) User coverage VS # of datacenters. (b) User coverage VS # of super nodes.

Figure 5: Impact of # of datacenters and supernodes on PeerSim.



(a) User coverage VS # of datacenters. (b) User coverage VS # of supernodes.

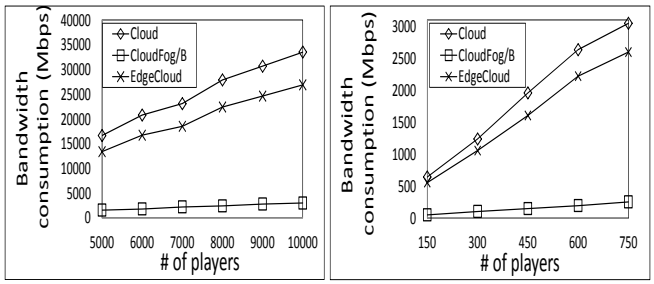
Figure 6: Impact of # of datacenters and supernodes on PlanetLab.

the capacity to be supernodes. We randomly selected 600 supernodes for *CloudFog*. This is reasonable as the hardware requirement of servers in *EdgeCloud* is much more demanded than that of supernodes in *CloudFog*, thus, given the same amount of revenue, *CloudFog* can deploy more supernodes than the number of servers. The number of friends for each player follows power-law distribution with skew factor of 0.5 [27]. In order to simulate the dynamics of supernodes and players, the players join the system following the Poison distribution with an average rate of 5 players per second [28]. Each node leaves the system after it finishes playing and joins the system for the next session. As in [29]–[32], the capacities of nodes follow Pareto distribution with a mean of 5 and shape parameter $\alpha = 1$. According to studies in [33], we randomly selected 50% nodes and 30% nodes to play for a period randomly selected from (0, 2] and (2, 5] hours a day, and let the remaining 20% nodes to play for a period randomly selected from (5, 24] hours a day.

We defined 5 games, their quality levels and latency requirements are shown in Figure 2. When a player joins the system, if none of its friends is playing, it randomly chooses a game to play; otherwise, it chooses the game that has the largest number of its friends playing. OnLive provides gaming service at a frame rate of 30fps [4]. Thus, the frame rate of game videos in our experiment is set to 30fps. The communication latency between nodes in the simulation was set based on the trace from the PlanetLab. The time period for one experiment is 4 days. In the PlanetLab experiments, we used 750 distributed nodes nationwide, and 300 of them have the capacity to be supernodes. The nodes with IP 128.112.139.43 in Princeton University and IP 131.179.150.72 in the University of California, Los Angeles were set as cloud datacenters, due to their stable connection during the experiment. In the experiment, we used TCP protocol to transfer data between different nodes. All other settings are the same as in the simulation.

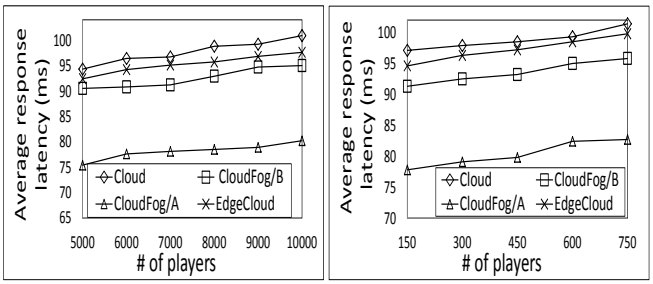
Experimental results. Recall that the general response latency requirement is 100ms [6]; 20ms is attributed to playout and processing delay and 80ms is the *network latency*. Figure 5(a) and Figure 6(a) show the ratio of covered players with different number of deployed datacenters and different network latency requirements of games on Peersim and PlanetLab, respectively. The figures illustrate that more datacenters lead to increased user coverage, as users are more likely to connect to close datacenters. Also, given a certain number of datacenters, stricter latency requirement leads to a smaller user coverage. In order to guarantee a better coverage of the user population, previous research suggested deploying more datacenters nationwide [19]. As building a medium size datacenter of approximately 300,000 gross square feet costs around 400 million dollars [34], [35], it would cost OnLive around 8 billion dollars to build 20 more datacenters; however, 25 datacenters can only cover 60% players with the general response latency requirement. Thus, increasing user coverage by deploying more datacenters is cost-prohibitive for game service providers. In *CloudFog*, a game service provider can offer a small amount of monetary rewards as incentives to encourage supernodes, and user coverage can be increased by deploying supernodes.

We then examine the effectiveness of supernodes in increasing user coverage under the current cloud infrastructure, that is, with 5 datacenters on PeerSim and 2 datacenters on PlanetLab. We see from Figure 5(a) that when the network latency requirement is 90ms, deploying 10 datacenters can increase about 10% user coverage than deploying 5 datacenters in PeerSim. Figure 6(a) reflects a similar trend as that in Figure 5(a), the two figures show that the effectiveness of increasing user coverage by deploying more datacenters weakens when the number of datacenters reaches a specific value. Figure 5(b) and Figure 6(b) show the ratio of covered players with different number of randomly selected supernodes and network latency requirements, Figure 5(b) shows that 100 supernodes can increase user coverage from 0.25 to 0.65 when the network latency requirement ranges from 110ms to 30ms. 200 supernodes can help achieve user coverage of deploying 25 datacenters. Figure 5(b) and Figure 6(b) show that instead of building datacenters, deploying



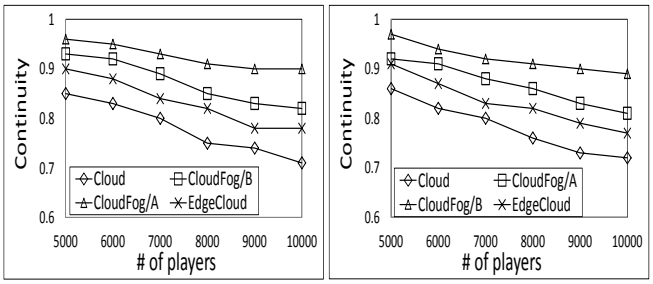
(a) The PeerSim simulator. (b) The PlanetLab real-world testbed.

Figure 7: Server bandwidth consumption.



(a) The PeerSim simulator. (b) The PlanetLab real-world testbed.

Figure 8: Response latency.



(a) The PeerSim simulator. (b) The PlanetLab real-world testbed.

Figure 9: Playback continuity.

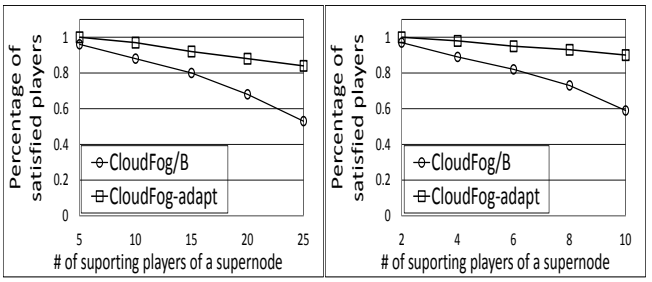
supernodes is an effective alternative in increasing user coverage.

Figures 7(a) and 7(b) show the bandwidth consumption of the cloud versus the number of players in the system. As *CloudFog/A* does not influent the bandwidth consumption of *CloudFog*, thus we use *CloudFog/B* to represent the bandwidth consumption of both *CloudFog/A* and *CloudFog/B*. We see that the result follows *Cloud*>*EdgeCloud*>*CloudFog/B*. The bandwidth consumption of *EdgeCloud* does not include those of additional servers. If we include them, *EdgeCloud*'s bandwidth consumption is similar to that of *Cloud*'s. *CloudFog/B* saves significant bandwidth consumption cost due to its employment of supernodes to stream game videos to the players. The cloud only needs to send update information rather than the entire game video to the supernodes. We also see that as the number of players increases, the bandwidth consumption increases. The increase rate of *CloudFog/B* is smaller than those of other systems. This means that our system can save more bandwidth cost when there are a very large number of players.

Figures 8(a) and Figure 8(b) show the average response latency per player in different systems in PeerSim and PlanetLab, respectively. We see that *EdgeCloud* generates slight shorter response latency than *Cloud* due to the use of scattered servers, and users are more likely to connect to servers within a short distance. *CloudFog/B* shows a slightly reduce in response latency than that of *EdgeCloud*, which indicates the effectiveness of our fog-assisted infrastructure in reducing the latency. In *CloudFog*, users are supported by supernodes that are physically close to them. As the game video is streamed from supernodes to the users, instead of from servers that are physically far away. Thus, *CloudFog* is able to reduce the response latency for users. This result shows that our system not only reduces the response latency of the system of deploying many datacenters but also saves the prohibitive cost of building more datacenters. *CloudFog/A* further reduces the latency, which indicates the effectiveness of our proposed strategies in reducing response latency.

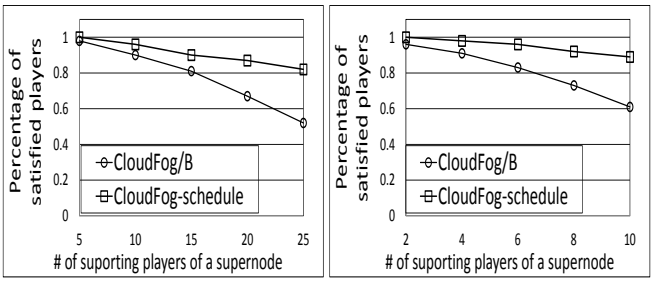
Video playback continuity is an important metric for QoE. When there are not enough packets in the cache, the player suffers from an playback interruption. We measured continuity by the proportion of packets arrived within the required response latency over all packets in a game video. Figures 9(a) and 9(b) show the average playback continuity of game videos when different number of players are playing games concurrently, which is a metric to measure weather a player can enjoy smooth video playback. We see that *Cloud* yields the lowest playback continuity because there are only a small number of cloud servers, which may locate far away from some players. So most game videos need to be transmitted from remote servers to clients, thus large portion of packets cannot be received within the required response latency. *EdgeCloud* produces higher continuity than *Cloud* because players in *EdgeCloud* are supported by their nearby servers. *EdgeCloud* generates smaller continuity than *CloudFog/B* and *CloudFog/A*, because not all users in *EdgeCloud* are able to connect to a nearby server due to the shortage of servers. So game video packets need to travel longer distance than that in *CloudFog*. *CloudFog/B* increases the continuity of *EdgeCloud* due to the effectiveness of the fog-assisted infrastructure, a large portion of users are supported by supernodes that are close to them. *CloudFog/A* provides an average of more than 90% continuity, with the contribution of all other proposed strategies.

In the following, we show the effectiveness of each of our proposed strategies: i) encoding rate adaptation and ii) deadline-driven buffer scheduling. Figure 10(a) and Figure



(a) The PeerSim simulator. (b) The PlanetLab real-world testbed.

Figure 10: Effectiveness of encoding rate adaptation.



(a) The PeerSim simulator. (b) The PlanetLab real-world testbed.

Figure 11: Effectiveness of deadline-driven buffer scheduling.

10(b) show the percentage of satisfied players with and without (denoted by *CloudFog-adapt* and *CloudFog/B*) the encoding rate adaptation strategy, in PeerSim and PlanetLab, respectively. We see that *CloudFog-adapt* increases the percentage of satisfied users in *CloudFog/B*. The increase rate reaches 27% when the number of supported players of a supernode is 25 in the simulation. When the network condition is not good enough to support high quality streaming of game videos, this strategy decreases the video quality level to meet the response latency based on loss rate tolerance, thus increasing the number of satisfied players.

QoE is determined by packet loss rate and response delay. Thus, if a user can receive 95% of its game packets within the game's response latency, we consider this user as a satisfied player, and this definition is adopted in all figures within the paper. Figures 11(a) and Figure 11(b) show the percentage of satisfied players with and without the deadline-driven buffer scheduling (denoted by *CloudFog-schedule* and *CloudFog/B*). We see that *CloudFog-schedule* increases the percentage of satisfied players, especially when a supernode is supporting a large number of players. This is due to the reason that in *CloudFog-schedule*, supernodes will assign higher priorities to packets with tight deadlines in the process of packet transmission, and drop packets from game videos based on the packet loss tolerance. Thus, *CloudFog-schedule* is able to ensure that packets arrive at the client within their latency requirement. The result confirms that this deadline-driven buffer scheduling strategy is effective in reducing the overall latency of the packets by globally considering their latency requirement and loss rate requirement of different games.

In Figures 11(a), 11(b), 10(a) and 10(b), as the number of supporting players of a supernode increases, the percent of satisfied players drops quickly in *CloudFog/B*, while that of *CloudFog-schedule* and *CloudFog-adapt* exhibits a moderate decrease. When a supernode supports more players, it needs to send out more packets at the same time, thus leading to high delays. Our strategies adjust the video quality and schedule sending queue based on different tolerant rates of packet loss and response delay of different games, thus resulting in a high percentage of satisfied players even in a traffic congestion. These results confirm the effectiveness of our proposed strategies in enhancing cloud game QoE.

V. Conclusions

The rapid growth of mobile users and high popularity of MMOG make thin-client MMOG an inevitable trend. However, conventional MMOG gaming model requires players to use powerful computers. Cloud gaming is a very promising model for thin-client MMOG since it frees players from this requirement, but it faces formidable challenges that prevents it from achieving high user QoE and low cost. We propose *CloudFog*, which leverages supernodes functioning as "fog" to connect the cloud to users. The cloud conducts the intensive computation for producing game state and sends update information to supernodes. The supernodes then generate game videos to stream to players. Considering that different games have different degrees of response latency tolerance and packet loss tolerance, we propose a receiver-driven encoding rate adaption and a deadline-driven buffer scheduling strategies to balance these two factors in guaranteeing user QoE. As a result, CloudFog reduces response latency and bandwidth consumption and increases user coverage. These advantages are verified by our experiments on the PeerSim simulator and the PlanetLab real-world testbed. In our future work, we will study the cooperation among supernodes in rendering and transmiting game videos to further reduce response latency. Also, we will study the security issues such as dealing with malicious supernodes and preventing cheating behaviors in *CloudFog*.

Acknowledgements

This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, Microsoft Research Faculty Fellowship 8300751.

References

[1] C. Bezerra and C. Geyer. A load balancing scheme for massively multiplayer online games. *Multimedia Tools Appl*, 2009.

[2] N. Bilton. Video Game Industry Continues Major Growth, Gartner Says. *The New York Times*, 2011.

[3] CBS News. Study: Number of smartphone users tops 1 billion. *http://www.cbsnews.com/8301-205_162-57534583/study-number-of-smartphone-users-tops-1-billion/*, 2012.

[4] Onlive. Inc. http://www.onlive.com/, [Accessed in Nov 2014].

[5] Gaikai. Inc. http://www.gaikai.com/, [accessed in nov 2014].

[6] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hobfeld. An Evaluation of QoE in Cloud Gaming Based on Subjective Tests. In *Proc. of IMIS*, 2011.

[7] S. Choy, B. Wong, G. Simon, and C. Rosenberg. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In *Proc. of NetGames*, 2012.

[8] K. Chen, P. Huang, and C. Lei. Game Traffic Analysis: An MMORPG Perspective. *Computer Networks*, 50(16):3002–3023, 2006.

[9] E. Carlini, M. Coppola, and L. Ricci. Integration of P2P and Clouds to support Massively Multiuser Virtual Environments. In *Proc. of NetGames*, 2010.

[10] P. Suresh Kumar, P. Sateesh Kumar, and S. Ramachandram. Recent Trust Models In Grid. *JATIT*, 2011.

[11] Y. Lee, K. Chen, H. Su, and C. Lei. Are all games equally cloud-gaming-friendly? An electromyographic approach. In *Proc. of NetGames*, 2012.

[12] C. Bezerra, J. Comba, and C. Geyer. Adaptive load-balancing for MMOG servers using KD-trees. *CIE*, 10(3):5, 2012.

[13] S. Ahmad, C. Bouras, E. Buyukkaya, R. Hamzaoui, A. Papazois, A. Shani, G. Simon, and F. Zhou. Peer-to-peer live streaming for Massively Multiplayer Online Games. In *Proc. of P2P*, 2012.

[14] J. Chen, M. Arumaithurai, X. Fu, and K. Ramakrishnan. Gaming over COPS: A Content Centric Communication Infrastructure for Gaming Applications. In *Proc. of ICDCS*, 2012.

[15] C. Huang, C. Hsu, Y. Chang, and K. Chen. GamingAnywhere: An Open Cloud Gaming System. In *Proc. of MMSys*, 2013.

[16] Z. Zhao, K. Hwang, and J. Villeta. Game cloud design with virtualized CPU/GPU servers and initial performance results. In *Proc. of ScienceCloud*, 2012.

[17] S. Wang and S. Dey. Cloud mobile gaming: modeling and measuring user experience in mobile wireless networks. In *Proc. of SIGMOBILE*, 2012.

[18] M. Hemmati, A. Javadtalab, A. Shirehjini, S. Shimohammadi, and T. Arici. Game as Video: Bit Rate Reduction through Adaptive Object Encoding. In *Proc. of NOSSDAV*, 2013.

[19] S. Choy, B. Wong, G. Simon, and C. Rosenberg. EdgeCloud: A New Hybrid Platform for On-Demand Gaming. Technical Report CS-2012-19, University of Waterloo, 2012.

[20] P. Salvador and A. Nogueira. Study on geographical distribution and availability of bittorrent peers sharing video files. In *Proc. of ISCE*, 2008.

[21] H. Shen and G. Liu. A lightweight and cooperative multi-factor considered file replication method in structured P2P systems. *TC*, 2012.

[22] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hobfeld. Gaming in the clouds: QoE and the users' perspective. *Mathematical and Computer Modelling*, 2011.

[23] H. Kanakia, P. Mishra, and A. Reibman. An adaptive congestion control scheme for real-time packet video transport. In *Proc. of SIGCOMM*, 1993.

[24] D. Gardner, J. Gardner, G. Laush, and W. Meinke. Method for the Analysis of Multicomponent Exponential Decay Curves. *The Journal of Chemical Physics*, 31(4):978–986, 1959.

[25] The PeerSim simulator. http://peersim.sf.net, [Accessed in Nov 2014].

[26] PlanetLab. http://www.planet-lab.org/, [Accessed in Nov 2014].

[27] S. Raza A. Nazir and C. Chuah. Unveiling dacebook: a measurement study of social network based applications. In *Proc. of SIGCOMM*, 2008.

[28] D. Wu, Y. Liu, and K. W. Ross. Modeling and Analysis of Multichannel P2P Live Video Systems. *TON*, 18(4):1248–1260, 2010.

[29] H. Shen and C. Xu. Locality-aware and churn-resilient load balancing algorithms in structured peer-to-peer networks. *TPDS*, 18(6):849–862, 2007.

[30] N. Bansal and M. Harchol-Balter. Analysis of srpt scheduling: investigating unfairness. In *Proc. of SIGMETRICS/Performance*, 2001.

[31] X. Zhang, Y. Qu, and L. Xiao. Improving distributed workload performance by sharing both cpu and memory resources. In *Proc. of ICDCS*, 2000.

[32] R. Subrata and A. Y. Zomaya. Game-theoretic approach for load balancing in computational grids. *TPDS*, 19(1):66–76, 2008.

[33] C. Hellstrom, K. Nilsson, J. Leppert, and C. Aslund. Influences of motives to play and time spent gaming on the negative consequences of adolescent online computer gaming. *Computers in Human Behavior*, 28(4):1379–1387, 2012.

[34] I. Goiri, J. Guitart, and J. Torres. Economic model of a Cloud provider operating in a federated Cloud. *Information Systems Frontiers*, 14(4):827–843, 2012.

[35] L. Barroso and U. Holzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Synthesis Lectures on Computer Architecture. 2009.