EdgeCloud: A New Hybrid Platform for On-Demand Gaming

Sharon Choy
University of Waterloo
s2choy@uwaterloo.ca

Bernard Wong
University of Waterloo
bernard@uwaterloo.ca

Gwendal Simon
Telecom Bretagne
gwendal.simon@telecom-bretagne.eu

Catherine Rosenberg
University of Waterloo
cath@uwaterloo.ca

Technical Report CS-2012-19

ABSTRACT

Cloud computing has been a revolutionary force in changing the way organizations deploy web applications and services. However, many of cloud computing's core design tenets, such as consolidating resources into a small number of datacenters and fine-grain partitioning of general purpose computing resources, conflict with an emerging class of multimedia applications that is highly latency sensitive and requires specialized hardware, such as graphic processing units (GPUs) and fast memory.

In this paper, we look closely at one such application, namely, on-demand gaming (also known as cloud gaming), that has the potential to radically change the multi-billion dollar video game industry. We demonstrate through a large-scale measurement study that the current cloud computing infrastructure is unable to meet the strict latency requirements necessary for acceptable game play for many end-users. In response to these results, we propose augmenting the existing cloud infrastructure with an EdgeCloud, consisting of end-hosts that are more geographically diverse than datacenters and also more likely to have specialized resources. We detail the challenges introduced by such a hybrid design and our solutions to these challenges, and demonstrate through a large-scale simulation that the addition of even a small number of end-hosts significantly reduces latency and improves client coverage for on-demand gaming.

1. INTRODUCTION

Cloud computing has, in the past few years, become the predominant environment for hosting web applications and services. Its rapid adoption largely stems from the intrinsic benefits of resource consolidation offered by the cloud, such as higher resource utilization resulting in lower costs, and more elastic resource acquisition diminishing the need for accurate growth forecasting. However, in order to maximize the benefits of resource consolidation, most cloud providers only offer general purpose computing resources that are located in a relatively small number of large datacenters. Furthermore, many cloud datacenter locations are chosen to minimize cooling and electricity costs, rather than to minimize latency to end-users.

Unfortunately, these architectural decisions are in conflict with the needs of an emerging class of multimedia applications that is interactive and therefore highly latency-sensitive, and requires specialized hardware resources, such as GPU and fast memory. Hosting these applications requires that the application providers either deploy their own infrastructure (*i.e.*, become a private cloud provider), or restrict their deployment to the very limited number of public cloud datacenters with the necessary equipment. Unfortunately, only the most profitable applications can command dedicated resources. The remaining applications must therefore limit their user-base to those who are geographically close to the specialized hardware-equipped datacenters, hence limiting user coverage. Such user-base restrictions are particularly damaging for applications with strong network effects that must reach a critical mass of users to become profitable.

In this paper, we select on-demand gaming as a representative case-study for this new class of applications. On-demand gaming, also known as cloud gaming, is a new video gaming application/platform that represents a giant shift in the multi-billion dollar gaming industry. Rather than requiring that end-users have sufficiently powerful computers to run modern games, on-demand gaming instead performs the intensive game computations, including the game graphics generation, remotely with the resulting output streamed as a video back to the end-users. Current on-demand gaming companies, such as Gaikai [6] and On-Live [13], perform the computation using cloud datacenter servers equipped with GPUs. In this platform, the client-side device only serves as a dumb-terminal, which enables games to run on any device that can maintain a high-speed network connection, such as smartphones and tablets in addition to computers, and thus greatly increasing the potential user-base.

However, unlike streaming movies and music, the interactive nature of games requires that the video stream of the game output be highly responsive to user input. Unfortunately, by offloading the computation to a remote host, this architecture suffers encoding latency, that is, the time to compress the video output, and network latency, which is the delay in sending the user input and video output back and forth between the end-user and the cloud. Although the video encoding latency will likely fall with faster hardware encoders, a significant portion of network latency is unavoidable as it is bounded by the speed of light in fiber. Past studies have found that a network latency above 80 ms appreciably affects the user-experience even for game genres that are relatively insensitive to input latency. Moreover, fast-pace multiplayer games are significantly more affected by network latency, as these games must already contend with coordination latency necessary to synchronize all of the game participants.

To quantify the additional network latency introduced by offloading computation to the cloud, we performed a large-scale measurement study consisting of latency measurements from PlanetLab and EC2 to more than 2500 end-users in the United States. We found that Amazon's EC2 infrastructure is capable of providing a median latency of 80 ms or less to fewer than 70% of our measured end-hosts. We also found that it requires a substantial increase in the total number of datacenters to significantly increase user coverage. Adding datacenters specifically for on-demand gaming is therefore prohibitively expensive. These results suggest that the cloud alone is a poor platform for hosting highly latency-sensitive applications, as a sizeable portion of the population would experience significantly degraded quality of service.

In response to the results of our measurement study, we propose a new cloud architecture, which we call *EdgeCloud*, that augments the existing cloud infrastructure with a centrally managed set of peers to help serve highly latency-sensitive applications such as on-demand gaming. With EdgeCloud, peers offer their resources in exchange for some incentives, and these resources can be used to run applications and serve content to other nearby users that are geographically distant from the cloud datacenters. An added benefit of incorporating peer resources is that many workstations are equipped with GPUs, which is a scarce resource in datacenters. For on-demand gaming, peers can also include unused gaming consoles, such as the Sony Playstation 3 or Microsoft Xbox 360, which further increases the pool of available peer candidates.

However, the EdgeCloud architecture also introduces a number of new challenges. For many applications, there is some fixed cost in incorporating a new peer into the system. For example, in on-demand gaming, the games must be distributed to the peers before they can be used to serve users, and given the size of modern video games, distributing the games to a large-number of peers can incur a non-trivial bandwidth cost. Therefore, selecting which games to distribute, at what quantity, and to which peers, can greatly affect the contribution from each peer and the total cost in providing the on-demand gaming service. We tackle this problem by modeling the costs and introducing a model-driven content distribution protocol to determine a game to peer mapping that is user-coverage-maximizing and cost-efficient.

Other pragmatic challenges introduced by the EdgeCloud architecture include ensuring that there is connectivity between the user and the peer by offering some form of NAT traversal, and matching users with peers that will be available for the duration of the user-session. We can leverage past work to perform NAT traversal [26] and address the latter problem by using a combination of policies that stipulate requirements for availability, and application-specific migration of saved state, which for gaming would involve periodically saving game state to the cloud and restarting the game from another peer.

We evaluate the EdgeCloud architecture through simulations parameterized by our large-scale measurement dataset. Our simulation results show that the EdgeCloud architecture can increase user coverage by 20%. With the same cost, our system achieves 10% more coverage than the naïve approach.

Overall, this paper makes three contributions. First, it identifies, through a large-scale measurement study, the cloud's inability to serve as a platform for hosting an emerging class of highly latency-sensitive multimedia applications. Second, it introduces the EdgeCloud infrastructure that augments the existing cloud infrastructure with end-hosts that are situated nearby the application users in order to significantly reduce network latencies. As part of the description of the EdgeCloud infrastructure, the paper identifies the challenges of using EdgeCloud for on-demand gaming, and introduces a model-driven content distribution protocol to maximize the benefits of this new infrastructure. Finally, this paper provides empirical results from a large-scale simulation that demonstrates the effectiveness of incorporating end-hosts into the cloud. By offering a low-cost mechanism to increase the number of potential customers for latency-sensitive applications, EdgeCloud offers a new, attractive alternative for hosting and serving a new emerging class of applications.

2. BACKGROUND

In this section, we begin by defining response time targets for cloud gaming. Afterwards, we analyze the response time of the cloud and characterize measured latency as client, network, and server latency. Lastly, we investigate the components of network latency, define the entities that are responsible for each component, and illustrate the latencies that we cannot reduce.

2.1 Response Time Targets

The interactive *response time* is defined as the elapsed time between when an action of the user is captured by the system and when the result of this trigger can be perceived by the user. For example, the *response time* is the amount of time it takes for a user action to be displayed on screen. We note that response time includes delays in addition to network latency.

Past studies have found Cloud Gaming to be highly demanding with respect to response time [28]. The work in [29] measures quality of experience perceived by gamers in various networking settings and several game genres. This study demonstrates that a latency around 100 ms is highly recommended for dynamic, action games; however, response time of 150 ms is required for slower-paced games.

2.2 Cloud Response Time Analysis

Cloud response time consists of latencies from various components, which are depicted in Figure 1. Additionally, various entities are responsible for the total latency; these are indicated by the lines on the bottom of Figure 1. The overall interactive response time T of an application is as follows:

$$T = t_{client} + \overbrace{t_{access} + t_{isp} + t_{transit} + t_{datacenter}}^{t_{network}} + t_{server}$$

In this section, we provide a breakdown of the latencies that form the total network latency.

We define t_{client} as the *playout delay*, which is the time spent by the client to (i) send action information (for example: initiating character movement in a game) and (ii) receive and play the video. Only the client's hardware is responsible for t_{client}.

Additionally, we define t_{server} as the *processing delay*, which refers to the time spent by the server to process the incoming information from the client, to generate corresponding video information, and to transmit the information back to the client. For the purposes of gaming, the work in [16] shows that *processing delay* is affected by the amount of computational resources provisioned by the cloud provider, and this delay ranges from 10 ms to more than 30 ms. Performance significantly degrades when several VMs share the same machine. The cloud provider is responsible for the *processing delay*.

Both playout and processing delays can only be reduced with significant hardware changes. For our purposes, we optimistically estimate that playout and processing combine amount to 20 ms of delay, although we recognize that this value can vary especially when we consider end-hosts as servers in Section 4. The remaining contribution of total latency comes from the network. We further divide the network latency into four components: t_{access}, t_{isp}, $t_{transit}$, and $t_{datacenter}$.

Firstly, t_{access} is the data transmission time between the client's device and the first Internet-connected router. The access network of residential homes is usually known as the bandwidth bottleneck of the end-to-end path; however, it is generally not attributed as a large source of network latency. Nevertheless, studies have shown that access network delay can be significant. Three quarters of end-users, who are equipped with a DSL connection, experience a t_{access} greater than 10 ms when the network is idle [24], and the average access delay exceeds 40 ms when the link is loaded [41]. The behavior of different network access technologies can greatly vary, as the latency of the access network can differ by a factor of three between different Internet Service Providers (ISP) [41]. Additionally, the home network configuration and the number of concurrent active computers per network access can cause the latency of the access link to double [23].

The second component of network delay is t_{isp}, which corresponds to the transmission time between the access router and the peering point connecting the ISP's network to the next hop transit network. During this phase, the data travels exclusively within the ISP's network. Although ISP networks are suppose to be fast and reliable, some major ISPs have indicated that they are overwhelmed by the traffic generated by new multimedia services [27].

The third component is $t_{transit}$, which is defined as the delay from the first peering point to the front-end server of the datacenter. Figure 1 indicates that the ISP and cloud provider are responsible for $t_{transit}$; however, the networks along the path are often owned by third-party network providers. Nonetheless, it is the responsibility of the ISP and the cloud provider to ensure good network connectivity for their clients. Therefore, the major cloud providers have developed their own Autonomous Systems (AS) and set peering agreements with major ISPs in order to obtain better control for this portion of this end-to-end network path.

Lastly, $t_{datacenter}$ is defined as the transmission delay between the front-end server of the datacenter and the hosting server for the client. The cloud provider is responsible for $t_{datacenter}$. The latency of the datacenter network has been a major concern for cloud providers [42]; however, typical network latencies in modern datacenters are below 1 ms [34].

3. A REALITY CHECK ON INFRASTRUCTURES FOR CLOUD GAMING

In this section, we study the ability of today's cloud to offer cloud gaming services. We focus on the network latency since the other latencies, especially the generation of game videos, have been studied in previous works [20, 30].

We conduct two different measurements to evaluate the performance and latency of cloud gaming services on existing cloud infrastructures in the US. Firstly, we perform a measurement study on the Amazon EC2 infrastructure during May 2012. Although EC2 is considered to be today's largest cloud provider, our measurements show that it has some performance limitations. Secondly, we use PlanetLab [17] nodes to serve as additional datacenters in order to estimate the behavior of a larger, more geographically diverse cloud infrastructure.

3.1 Measurement Settings

As emphasized in previous network measurement papers [24,

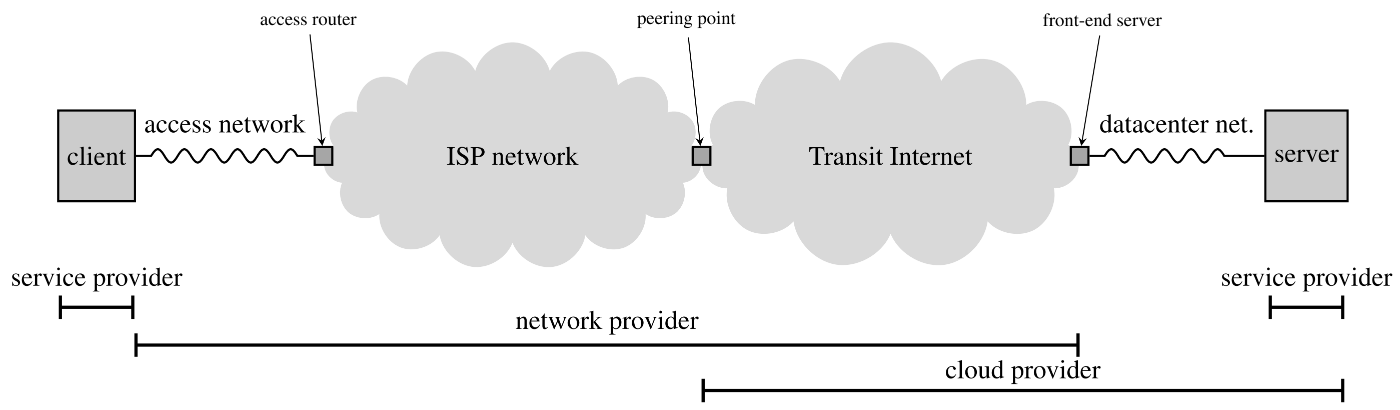

Figure 1: Cloud Response Time Decomposition

41], it is challenging to determine a representative population of real clients that are involved in extensive measurement experiments. For our measurements, we utilize a set of $2{,}504$ IP addresses, which were collected from twelve different BitTorrent [3] swarms. We implement a modified BitTorrent client that fetches the IP addresses of peers from public BitTorrent trackers. We then use the *GeoIP* [11] geolocalization system to filter out end-users located outside of the US. The geographic distribution of the end-users, whom we refer to as *clients*, is illustrated in Figure 2.

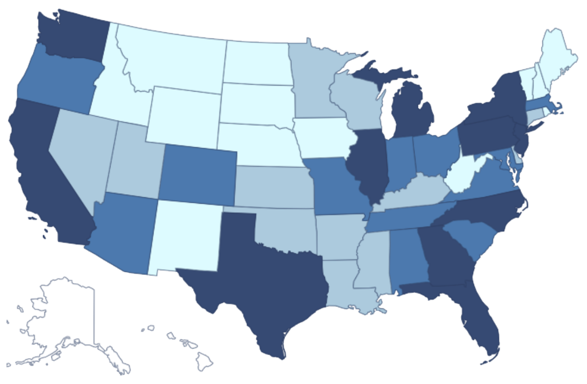

Figure 2: Geographic distribution of clients. Light blue, azur, blue and dark blue stand for states having respectively less than 14, between 15 and 27, between 28 and 71, and more than 71 clients.

One main advantage of retrieving IP addresses from a BitTorrent system is that BitTorrent tracker provides both an IP address and an open TCP port. By connecting to an open TCP port, we can measure the round-trip-time from the initial TCP handshake, which is more reliable than a traditional *ping*, since the *pings* are frequently filtered by network operators.

3.2 Case study: Amazon EC2

The Amazon EC2 cloud offers three datacenters in the US to its customers. We obtain a virtual machine instances in each of the three datacenters. Every 30 minutes, we execute a program that measures the latency between each datacenter to all the $2{,}504$ clients. We choose the median latency from the set of collected measurements.

We present in Figure 3 the main results of our measurements. Figure 3 depicts the ratio of covered end-users that have at least one network connection with a latency below x ms. Two observations can be made from this graph:

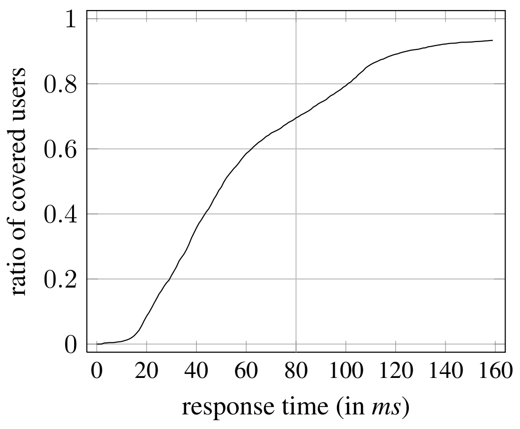

Figure 3: Population covered by EC2 cloud infrastructure

- *More than one quarter of the population cannot play games from an EC2-powered cloud gaming platform.* The thin, vertical gray line in Figure 3 represents the 80 ms threshold network latency yielding a 70% coverage, which we deem unacceptable.
- *Almost one tenth of the potential clients cannot be reached at all.* Although we filter out the IP addresses that experienced inconsistent latency results, we still observe that a significant proportion of the clients have a network latency over 160 ms. This result confirms the measurements made by previous works, which determine that home gateways can introduce a significant delay on data transmission [41].

3.3 Effects of a larger cloud infrastructure

The multi-billion dollar gaming industry has the potential to bring new cloud operators into the cloud computing market. An alternative to deploying a small number of large datacenters is to instead use a large number of smaller datacenters. Providers such as Gaikai [6] or Onlive [13] claim to possess up to a dozen of datacenters within the US [7, 9] in order to guarantee a better coverage of the population. Since a large datacenter is economically more efficient than a small data center, cloud providers should carefully determine if the initial investment to build a new datacenter is worth the price. In this section, we investigate the gain in population coverage when new datacenters are added into existing infrastructure

We utilize 44 PlanetLab nodes as possible locations for installing datacenters.We consider a cloud provider that can choose from the 44 locations to deploy a k-datacenter cloud infrastructure. We design two strategies for deciding the location of datacenters:

- **Latency-based strategy**: the cloud provider wants to build a dedicated cloud infrastructure for interactive multimedia services. The network latency is the *only* driving criteria for the choice of the datacenter locations. For a given number k, the cloud provider wants to place k datacenters such that the number of covered end-users is maximal. When k is greater than four, we approximate the optimal results by taking the best k-subset out of five thousand randomly generated subsets.

- **Region-based strategy**: the cloud provider tries to distribute datacenters over an area. However, it takes into account various criteria for the location of its datacenters (for example: electricity cost, workforce, infrastructure quality and natural risks). We divide the US into four regions as set forth by the US Census Bureau: Northeast, Midwest, South, and West. Every datacenter is associated with its region. In every region, the cloud provider chooses *random* datacenter locations. Thus there is at least one datacenter in every region, and no region has more than two datacenters than another region.

For cloud providers, the main concern is determining the minimum number of datacenters to cover a significant portion of the target population. We present Figure 4, which depicts the ratio of covered populations for two targets network latencies: 80 ms, which enable good response times for action games, and 40 ms for ultra-fast response times.

The main observation is that a large number of datacenters is required if one wants to cover a significant proportion of the population. Typically, a cloud provider, which gives priority to latency, reaches a disappointing coverage ratio of 0.85 with ten datacenters. Using the region-based strategy requires nine datacenters to reach a (poor) 0.8 ratio. In all cases, a 0.9 coverage ratio with a 80 ms response time is not

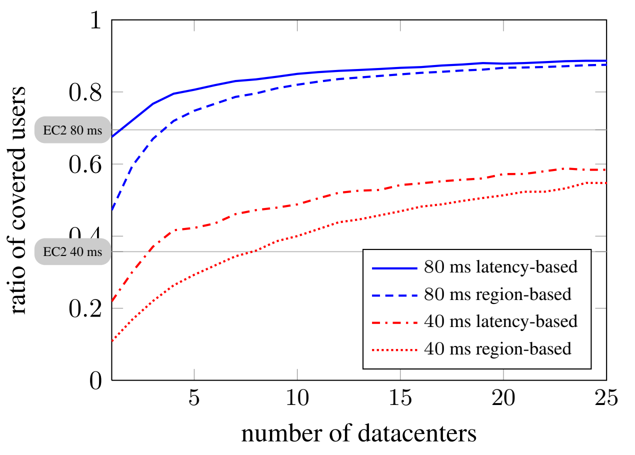

Figure 4: Coverage vs. the number of deployed datacenters

achievable without a significant increase in the number of datacenters (around 20 datacenters). Similarly, one cannot expect the majority of the population to have ultra-fast response time. Even if 20 datacenters are deployed, less than half of the population would have ultra-fast response time.

We also emphasize that EC2 is a reasonable 3-datacenter deployment. It is in particular as good as a latency-based 3-datacenter for the 40 ms target response time. The performances of EC2 are also close to the ones of a region-based 3-datacenter for the 80 ms response time. The similarity beween the our measurements from PlanetLab and EC2 suggests that PlanetLab nodes can simulate datacenter sites.

We then focus on the performances of two typical cloud infrastructures: a 5-datacenter, and a 20-datacenter infrastructure. We assume a region-based location strategy since it provides a good trade-off between costs and performances. We present in Figure 5 the ratio of covered populations for both infrastructures.

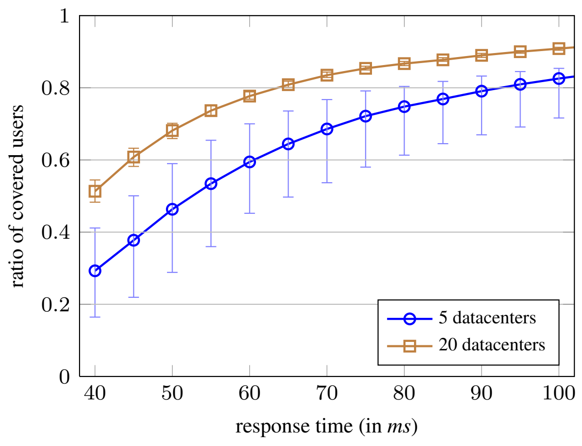

Figure 5: User coverage for a region-based datacenter location strategy (average with min and max from every possible set of locations)

We observe that there can be significant performance gaps between a 5-datacenter deployment and a 20-datacenter deployment. Moreover, five datacenters do not guarantee good performances, despite we assume a region-based location strategy that is expected to provide a good coverage of the area. Typically, 80% coverage for 80 ms is achievable with five datacenters; however, a poorly chosen 5-datacenter deployment can result in a disastrous 0.6 coverage ratio. In contrast, a 20-datacenter deployment exhibits insignificant variances in the coverage ratio.

4. EDGECLOUD

The performance limitations of existing datacenter-based cloud architectures restrict their viability as platforms for latency-sensitive multimedia applications. To address these performance limitations, we introduce EdgeCloud, a new hybrid architecture that augments the existing cloud infrastructure with resources from a small body of participating peers. Although peer resources are generally less reliable and more difficult to manage, peers have two important characteristics that public cloud datacenters do not have: locality and access to specialized hardware. In this section, we describe the EdgeCloud architecture, explore some of the key challenges behind using peers to serve clients, and describe how we address these challenges.

4.1 Architecture

The EdgeCloud architecture combines resources from existing public cloud datacenters with resources from peers in order to increase client coverage; a client is covered by EdgeCloud if it can access the application within the latency threshold that has been defined by the application provider. EdgeCloud is designed as a shared platform for simultaneously serving multiple latency-sensitive multimedia applications. In the context of on-demand gaming, each application is a different game, and each game can have different latency requirements.

EdgeCloud provides its clients with streaming-based remote access to their applications. This is the same type of access that existing Cloud Gaming providers offer, and is conceptually similar to remotely accessing an application over Citrix [35], X11 [15] or VNC [14]. In streaming-based remote access, a client's input is sent to a remote server that is running the application, and the application's output is continuously sent as a compressed stream to the client. The video and audio output can be captured either through modified drivers at the server, or a external hardware device such as SpawnBox [10]. The key advantage to streaming-based remote access is that it supports unmodified applications, and therefore, provides the EdgeCloud platform with a large corpus of existing multimedia applications and games.

EdgeCloud's primary goal is to maximize client coverage. As such, with the assumption that the public datacenters are sufficiently provisioned to handle nearly any volume of requests, EdgeCloud preferentially serves clients from data-

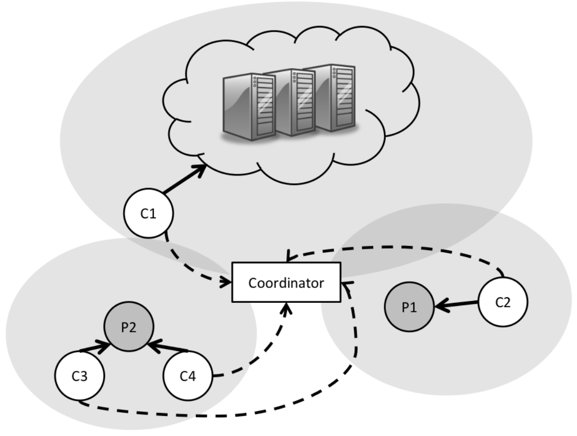

Figure 6: The architecture of EdgeCloud consists of a datacenter, peers, and a coordinator to assign clients to peers. Clients make a request to the coordinator (dotted line), and the coordinator determines if the client is to be served by the datacenter or a peer (solid line)

centers when the datacenter servers can meet the application-specific latency requirements. For clients that cannot be served from datacenters, EdgeCloud will select peers with a copy of the application pool that meet the latency requirements. Figure 6 illustrates the EdgeCloud architecture.

4.1.1 Hardware Requirements

An EdgeCloud server, located either at a datacenter or nearby peer, must have sufficient computational power and the necessary specialized resources, such as a GPU and fast memory, to run the application. Most modern home computers with a discrete graphics card can satisfy these requirements. Popular game consoles, such as the Sony Playstation 3 and Microsoft Xbox 360, can also serve as EdgeCloud servers especially in the context of providing a cloud gaming service. With more than 80 million game consoles sold in the US since 2005 [8], including under-utilized game consoles significantly increases the number of peers that can serve as EdgeCloud servers.

In addition to meeting the computational requirements, a server must also have sufficient downstream bandwidth to receive the client input, which is generally negligible, and upstream bandwidth to send the compressed video and audio output. Past studies [25] have found that the YouTube video service streams standard definition videos at 0.6 Mbps, and high definition videos up to 3 Mbps. We expect compressed application outputs to have similar bitrate demands. Although this bandwidth requirement currently precludes a non-trivial percentage of US homes from serving as peers, with the average upload speed in the US being 435 kbps [1], there is a strong upward trend in the number of peers with high upstream bandwidth that will likely eliminate this barrier in the near future. Upstream bandwidth of 10 Mbps or more is already commonplace in many countries in Europe

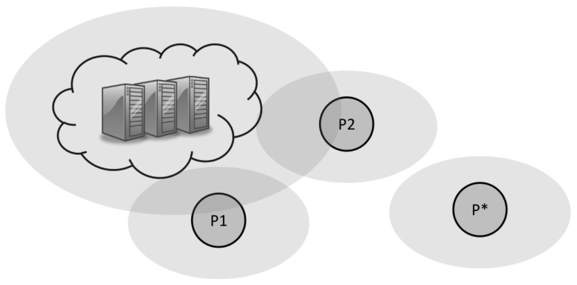

Figure 7: Assuming that client distribution is uniform in this figure, P1 and P2 are not as valuable as P*, since P1 and P2 have overlapping coverage with the cloud datacenter

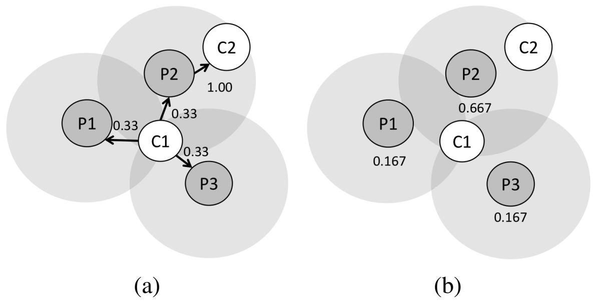

Figure 8: In 8(a), each client assigns a fraction of its vote to prospective peers. Afterwards, in 8(b), the probability of a peer becoming a candidate is determined by its total vote and the number of clients

and Asia [2] .

An EdgeCloud provider manages peers through a logically centralized coordinator located at one or more of its datacenters. As part of the registration and login process, a peer indicates to the coordinator what time periods it is available to provide service for clients. Incentives to the peers are determined via an automated reverse auction; we will detail EdgeCloud's resource auctioning scheme in Section 4.4. Because an application's size may be on the order of gigabytes, EdgeCloud assumes that applications cannot be loaded on peers on demand. The coordinator must therefore determine which applications are preloaded at each peer, with peers indicating to the coordinator the amount of disk space they have available for storing applications. Finally, EdgeCloud assumes that a peer can only serve one client at a time. In our model, well provisioned peers that are capable of running multiple applications concurrently and have sufficient upstream bandwidth to stream to multiple clients are treated as multiple independent peers. Given that we envision many of these applications are sufficiently resource-hungry that they require dedicated access to most modern GPUs, we believe that the one client per peer constraint is a reasonable simplifying assumption.

4.2 Candidate Distribution

Supporting applications with large footprints requires EdgeCloud preloads applications on peers. However, not all peers that are preloaded with applications will be selected to serve a client, as peer selection is subject to varying client demand. We call a peer with any preloaded application a *candidate*, with all candidates collectively forming a single candidate pool. A simple approach to creating the candidate pool is to accept all peers that meet the minimum hardware requirements. This would provide the greatest flexibility in assigning peers to clients, but assuming that the size of the candidate pool is substantially larger than the number of selected peers, the EdgeCloud provider would incur significant overhead in disseminating large applications to peers that are never selected.

EdgeCloud reduces dissemination overhead by tracking P, the peak number of clients per day that cannot be served from the cloud datacenters, and then organizing a candidate pool that is k percent larger than P from the previous day, where k is an EdgeCloud provider specified parameter. Increasing k improves client coverage, at the cost of additional dissemination overhead.

Determining which peer to include in the candidate pool can also affect client coverage. Selecting only peers from one geographic region will, not surprisingly, yield a poor candidate pool; clients from other parts of the country would be unable to find a nearby peer. A candidate pool consisting of a geographic distribution of peers that follows the geographic distribution of clients provides significantly higher client coverage. However, the quality of this candidate pool is still limited as the peer selection is oblivious to the location of the cloud datacenters. As datacenter servers that meet the application's latency requirements are selected preferentially over peers to serve clients, candidates that are situated near datacenters are much less likely to be selected than those that are further away.

EdgeCloud accounts for cloud datacenters in its candidate selection by selecting candidates proportional to the amount of additional client coverage that the candidates provide. Using a representative sample of clients, each client that cannot be serviced by a cloud datacenter divides a vote across all peers that can service it. For example, out of m peers that can service a client, the client awards $\frac{1}{m}$ votes to each of these peers. Given there are n clients in the representative sample that cannot be served by datacenters, each peer divides its total vote by n to determine the probability that it should be selected as a candidate. This algorithm biases candidate selection to increase the client coverage by recognizing that certain peers are more valuable than others based on its relative location to the cloud datacenters.

4.3 Application Placement

As the number of applications that can be placed at a candidate is generally much less than the total number of applications that the EdgeCloud provider is serving, it is important to distribute the games to peers in sufficient volume to serve all clients and following a distribution that maximizes coverage. Specifically, the application placement algorithm should minimize the occurrence of candidates being available, but none of the candidates are hosting the applications that can satisfy the client requests.

EdgeCloud uses a greedy-approach to application placement that aims to maximize client coverage for each application. The number of replicas to create for each application is based on client demands from the previous day; additional manual provisions can be made to accommodate new applications. The first replica for each application is placed on a random candidate. Additional replicas are placed one at a time on candidates that, together with the previously selected replicas, maximize client coverage. This discourages selecting the replicas with significant client overlap with the existing replicas, as they would have little impact on the total client coverage from the replica set. To estimate the coverage of a subset of peers, EdgeCloud leverages on a representative sample of clients, in the same manner as for the voting candidate selection.

When the increase in coverage is equivalent across multiple candidates, EdgeCloud aims to balance the popularity distribution of applications on the candidates in its candidate selection. For example, when deciding on which candidate to place an unpopular application out of candidates that offer equivalent client coverage, EdgeCloud will choose the candidate that is hosting mostly popular applications over the one that is hosting mostly unpopular applications. Ideally, each candidate should host applications with a popularity distribution that mimics the popularity distribution of the entire application library.

4.4 Peer Selection

EdgeCloud's peer selection criteria, that is, how it selects which peer to serve a client request, is a function of several different parameters that affect both the clients and the EdgeCloud provider. These parameters include: the cost of employing the peer, the reliability of the peer, the reduction in coverage from removing the peer from the candidate pool, and the absolute latency between the peer and client. In this section, we look at the reasons for including each of these parameters in turn, and then describe the peer utility metric that EdgeCloud uses to select peers.

Pricing. An EdgeCloud provider must provide incentives to attract peers to join its network and provide services to its users. As we assume that there are no prior business relationships between the provider and the peers, the existence of which would allow for some other goods and services to be exchanged in return for providing service to a client, EdgeCloud relies on monetary payments to compensate peers. The payment amount must be fluid with respect to user demand in order for EdgeCloud to ensure that it does not have to refuse service to its users due to having insufficient peers.

To this end, EdgeCloud uses an automated reverse auction to determine peer pricing. In a reverse auction, an agent on each peer submits a price for the peer's service with the expectation that the EdgeCloud provider is incentivized to select lower price peers. The goal of an agent is to maximize the peer's profit by determining the maximum price that the EdgeCloud provider is willing to pay, while minimizing the unpaid price exploration time. A peer can also specify the minimum price it is willing to accept to ensure that the compensation covers the opportunity cost lost from renting out its computational resources.

Reliability. As with any for-profit business, it is in the EdgeCloud provider's best interest to minimize operating costs. However, serving a client from an unreliable peer can lead to a very poor user experience when serving a long-lived application, as is the case for many games, which can negatively affect user adoption. The EdgeCloud service provider should therefore account for the value of consistent reliability from a peer in selecting peers for clients.

The reliability of a peer can be automatically tracked by ensuring the client software sends back a report to the coordinator at the end of each session. A reliability score is assigned to each peer reflecting the percentage of sessions that it successfully served. For cloud gaming, as different game genres have different average session lengths, the coordinator maintains for each peer a separate reliability metric for each game genre. The peer selection function uses the reliability specific to the game genre that the client game selection falls under.

Application Coverage. In EdgeCloud, each peer can only service one client at a time. By selecting a peer to serve a client request, the peer is removed from the candidate pool and the coverage for the applications that the peer is hosted are reduced for subsequent clients. This might have a dramatic effect for the (unpopular) applications that are rarely hosted by the peer. Therefore EdgeCloud aims to minimize the cases where a peer is selected among a set of eligible peers although this peer hosts a rare application and some other eligible peers store only well-replicated applications. EdgeCloud preferentially selects the peer that stores the most popular applications out of the set of peers that respect the target application response time to the client.

Latency. Although reducing latency beyond the application's specified threshold will generally provide diminishing returns, previous studies have shown that even a small amount of additional latency is noticeable to most game players [19, 21, 22, 30]. We therefore include this additional parameter to help steer the selection process towards a better user experience when all else are essentially equal.

Peer Utility Function. Our utility function for a peer is a linear combination of these four parameters:

$$u = \begin{cases} 0, & \text{if } l > T. \\ \alpha c + \beta r + \gamma(1 - v) + \delta(T - l)/T, & \text{otherwise.} \end{cases} \tag{1}$$

where c is the cost, r is the reliability index from 0 to 1, v is the vote value from 0 to 1, and T is the latency threshold, and l is the latency to the client. The constants, represented by the Greek symbols, are parameters specified by the EdgeCloud provider and reflect the provider's business focus. EdgeCloud selects the peer with the highest utility value to serve each client.

5. EDGECLOUD EVALUATION

In this section, we explore the effectiveness of using peers in addition to using cloud resources. We demonstrate the potential of EdgeCloud, and our experiments show that EdgeCloud increases the ratio of covered users. We determine the critical ratio of peers to clients, and we show that we can achieve a better coverage ratio if peers can host more games.

5.1 Settings

Out of the 2,504 IP addresses collected in our measurement study, unless otherwise specified, we select 1500 clients, 330 candidate peers, and 300 peers. Each peer stores five applications. We consider that a peer is actually located in its closest PlanetLab node, and we added an extra latency (randomly chosen between 0 to 15 ms) to the latency between servers and clients. The extra latency models servers that are located in a residential network, which suffers from additional last-mile latencies.

5.2 Potentials of EdgeCloud

We first explore the potentials of the EdgeCloud infrastructure. Figure 9 presents the ratio of covered users by both EC2 and the combination of EC2 and peers. We consider here an idealized system without constraints on client-peer matching. In this scenario, there is only one game, and each peer can serve an unlimited number of clients.

These results demonstrate the significant gains that can be achieved by the EdgeCloud infrastructure. EdgeCloud nearly doubles the ratio of covered users who's target response time is below 45 ms and achieves a 28% increase in the ratio of users for a 80 ms target response time. Our results in Section 3.3 show that more than 20 datacenters are required to achieve a similar improvement.

5.3 The critical ratio of peers to clients

We now focus on the 80 ms target response time, and we consider the factors that affect the performance of EdgeCloud. A closer look at the system lets us make a clear distinction whether clients are covered by EC2 or not. The *EC2-uncovered clients* can then also be distinguished between those who may be covered by a peer and those who are unreachable for a given response time.

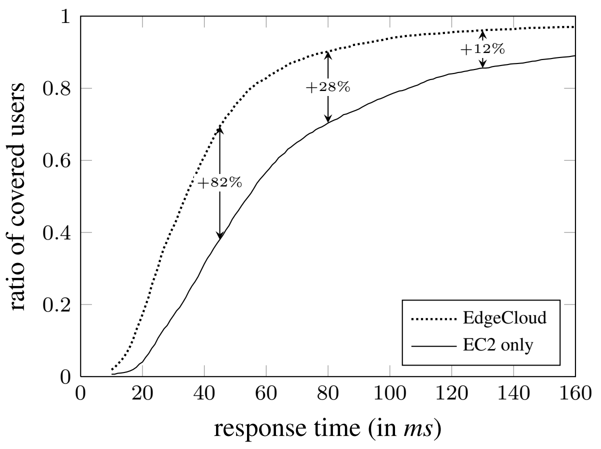

Figure 9: Maximum achievable performances for an EdgeCloud infrastructure

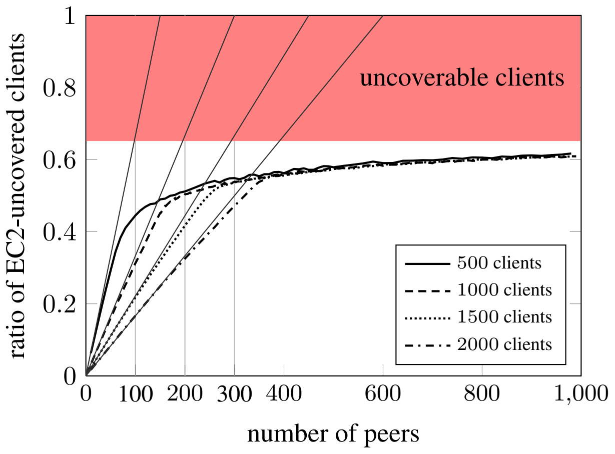

Figure 10: Ratio of covered clients among the EC2-uncovered clients (for a catalog of 100 games and each edge server stores 5 games)

Figure 10 illustrates the number of peers required to serve a given ratio of the population. We emphasize hereafter that EdgeCloud is naturally bounded in two aspects.

- Given our dataset, out of the clients who cannot be served by a datacenter, only 65% can be covered by peers. The remaining clients exhibit excessive delay to all peers and datacenters, which is likely due to non-network delays that are outside of EdgeCloud's control. Thus, EdgeCloud's performance with respect to the ratio of covered clients is limited by this "ceiling".

- All clients that cannot be served by datacenters must be served by peers. As we assume that a peer can serve only one client, there must be at least as many peers as there are clients that are not covered by datacenters. The vertical grey lines in Figure 10 represent the number of clients that are not covered by datacenters.

Ideally, EdgeCloud would be able to achieve a perfect matching between the clients, which are not covered by EC2, and peers. However, peers host only a subset of games, and they are geographically distributed, hence not all peers can serve all clients. We next look at how different parameters affect EdgeCloud's coverage.

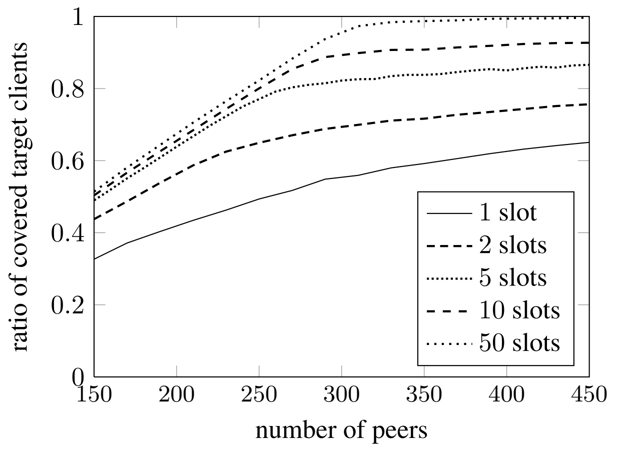

Figure 11: Impact of the number of games that the system pushes into the candidate peers (1,500 **clients, all candidate peers utilized,** 100 **games in catalog)**

In Figure 11, we demonstrate that by increasing the number of allowed games per machine, which we call *slots*, we increase EdgeCloud's client coverage. By allowing peers to host more games, there is increased flexibility when matching clients with peers. Given a catalogue of 100 games, we approach the theoretical maximum coverage by having 50 slots per peer. However, Figure 11 demonstrates that increasing slots has diminishing returns. If more slots are used, bandwidth usage is increased as more applications need to be pushed to the peers. Depending on the budget of the cloud provider, it would be reasonable to limit the number of slots for each peer. Furthermore, not all peers may be able to host a significant number of slots.

In addition to increasing the number of slots to improve client coverage, Figure 12 shows that increasing the candidate pool leads to better client coverage. A larger candidate pool results in more flexibility for matching clients with peers. Similar to increasing the number of slots, there is more bandwidth usage as the candidate pool becomes larger. Furthermore, depending on the business model, candidates may be paid, and this adds to the costs of the cloud provider.

We analyze the effectiveness of EdgeCloud's algorithm. Firstly, we consider a randomized algorithm for candidate selection, application placement, and peer selection. Thus, peers are randomly selected from the same distribution as users, candidates are selected randomly from these peers, and applications are placed randomly on these peers according to the application's popularity. As demonstrated in Figure 13, random selection provides good user coverage. We

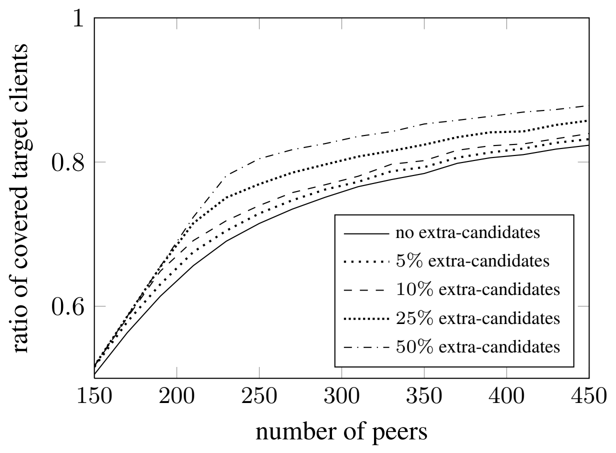

Figure 12: Expected performances improvement by pushing games to more candidate peers than actually used peers

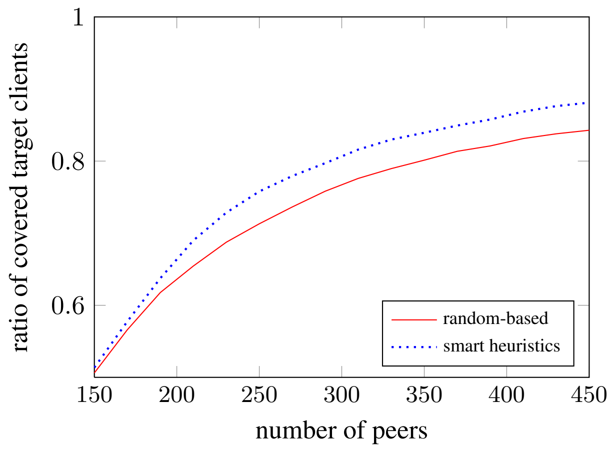

Figure 13: Comparison: naive random-based heuristic vs. the combination of heuristics

compare random selection to the heuristics presented in Section 4. That is, peers are selected by a voting-based selection, games are pushed according to the coverage of peers, and clients are assigned to peers so that the peers hosting unpopular rare games are not preferentially selected. We see that these additional heuristics provide a 10% increase in coverage. Since using such heuristics do not increase the cost for the cloud provider, they should be used in place of a more basic random-based algorithm.

Finally, we consider different application response times. We previously argued that 80 ms is a widely accepted target. However, fast-paced multiplayer games may require even faster response times. In Figure 14, we represent the overall ratio of covered clients by EC2 only, and by EdgeCloud with various number of peers. For a 30 ms target response time, if EdgeCloud is given 450 peers, the ratio of covered clients is more than twice what EC2 can achieve.

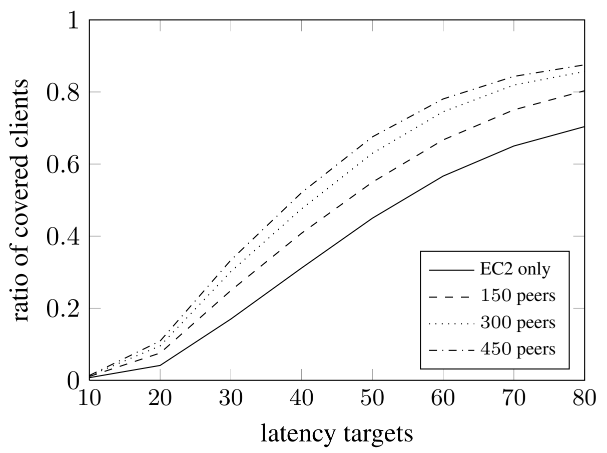

Figure 14: Improvement achieved by EdgeCloud for various latency targets ($1,500$ clients, 100 games and five slots per peers)

6. DISCUSSION

Business Viability for EdgeCloud Providers. In order to determine the profitability of on-demand gaming using EdgeCloud, EdgeCloud providers need to consider the demand to play computationally-intensive games on non-specialized hardware, the supply of prospective peers, and the appropriate amount of compensation for participating peers. EdgeCloud providers will also need to consider the marginal return of using peers in order to determine when it would be worthwhile to build a new datacenter altogether. Although a detailed market analysis is outside the scope of this paper, we believe that by increasing user coverage by 20%, EdgeCloud would increase the profitability of many applications that are aimed at mass markets, which includes on-demand gaming, by expanding the user base.

Obstacles for Adoption. Since EdgeCloud relies on participating peers to provide the GPU and computational power for games, having an insufficient number of peers can be an obstacle for EdgeCloud. Furthermore, previous work [37] has found that the median session lengths for peers in peer-to-peer file sharing systems varies from one minute to one hour. Therefore, it is critical to ensure that we have a sufficient number of peers, these peers have reasonable availability, and there are mechanisms to cope with disconnecting peers.

Users willing to participate. A fundamental component of EdgeCloud is the remuneration of participating peers. By providing financial incentives, EdgeCloud can attract a sufficient number of prospective peers. There is evidence that a significant portion of the population is willing to provide computational resources for others. In particular, we reference distributed computing projects such as Folding@Home [5] and DNA@Home [4]. In these projects, peers provide computational power to fold proteins and analyze DNA for the purpose of discovering new pharmaceuticals or curing cancer. Therefore, this demonstrates that there are end-hosts who are willing to participate in a variety of distributed applications.

Moreover, we also need to encourage users, who are willing to sell time on their idle resources, to apply for the candidate pool. EdgeCloud provides a nominal payment to candidates in order to cover the bandwidth cost of uploading applications and to encourage prospective peers to join. However, the primary source of income for a candidate comes from performing a client's computation. In order to adapt to demand, we can increase the candidate payment in order to increase the number of prospective peers.

Ensuring Peer Participation and Reliability. In addition to encouraging peer participation, another concern is peer disconnection while serving clients, as it is inevitable that peers will periodically go offline. For example, the owner of the selected peer may want to use his or her hardware, or there may be a catastrophe such that the peer cannot run the application. In order to address this concern, peers can periodically send game state back to the datacenter. Therefore, if a client needs to be migrated to another machine, it is able to retrieve its last saved state. Additionally, if a peer wishes to disconnect, it can pre-emptively migrate the client's saved state to minimize client disruption.

Extending Applicability Beyond Games. In this paper, we motive EdgeCloud for the purposes of on-demand gaming by maximizing application coverage. However, remotely using the computational power of GPUs and other specialized hardware, while providing low-latency, is relevant to other applications as well.

Applications Requiring GPU. In addition to on-demand gaming, the EdgeCloud infrastructure can be used in various computing projects that require graphical rendering or intensive computation that cannot be done on the target (end-user) device. For example, in a 3D-modelling application, graphics rendering cannot be done locally on non-specialized hardware such as a cellphone or tablet. Nonetheless, artists may wish to render images or animations on non-specialized hardware, such as a tablet, which is better suited for completing their task (e.g. a tablet is easier to use for drawing and more portable).

Virtual Desktop. Although cellphones and tablets are becoming increasingly popular, their computational power is still limited when compared to a workstation. Nevertheless, these devices are far more portable, and it would be beneficial if they could use remote resources. By using an infrastructure such as EdgeCloud, these devices can allow the end-user to perform tasks that are normally done at a workstation. By saving state in the cloud, users can move freely between different input devices without having to manually transfer state. Although it is already possible to use the cloud for this purpose, EdgeCloud provides the user with much lower-latency access to his or her virtual environment.

Auctioning Process. As outlined in Section 4.4, one factor that maps a client to a peer is the peer's cost, which is determined by a reverse auction process. Given all other factors affecting the choice of peer are equal, the EdgeCloud coordinator chooses the candidate that is willing to provide its service at the lowest possible price in order to minimize the cost that the on-demand company incurs.

Collusion. A common issue when using auctions is that participants can collude. Since prospective peers need to inform the coordinator of their minimum required compensation when registering, EdgeCloud implements a sealed-bidding mechanism. Therefore, other prospective peers should not know their competitor's bidding value. Nonetheless, this does not prevent groups of prospective peers from submitting the same minimum required compensation to the EdgeCloud coordinator. However, given a large candidate pool, it is very difficult for all candidates to collude with each other in a manner that would cause the on-demand gaming provider to incur significant costs. To encourage truthful bidding, it is possible to use a reverse Vickrey auction [36] where the lowest bidder wins; however, the lowest bidder receives the payment of the second-lowest bid.

7. RELATED WORK

On-demand gaming is attractive to many end-users since it offers hardware independence by offloading computation to the cloud. Therefore, as on-demand gaming becomes more prevalent, systems such as EdgeCloud have increasing relevance as they reduce latency in order to provide end-users with a high-quality gaming experience. Studies such as [19, 21, 22, 30] demonstrate that short latency times are required in order to maintain an enjoyable user-experience. Although there are many technologies (such as video-on-demand applications, peer-assisted download services, and content distribution networks) that are similar to EdgeCloud, the goals of such technologies are not applicable to the gaming environment.

Claypool and Claypool [22] show that performances of various game genres degrade differently as latency increases. Depending on the genre, the performance of a game degrades by half after 100 ms, 500 ms, or 1000 ms for first-person avatar, third-person avatar, and omnipresent games respectively. Additionally, the work in [30] presents a measurement environment that emulates an on-demand gaming service and defines tests to gauge a user's reactions to various settings of propagation delay and packet loss. Although [22, 30] show that different genres of games degrade differently, the work in [30] argues that users begin to notice a delay of 80 ms. User-perception of gaming quality is crucial, and the results in [19] demonstrate that poor quality of service affects an end-user's willingness to continue the game. Therefore, to provide users with an enjoyable experience, low-latency is required, and this motivates the need for systems such as EdgeCloud.

Existing On-Demand Gaming Providers. On-demand gaming is attractive to game players since it allows them to instantly play games without having the required games locally, or possessing the necessary hardware to play the games. And as non-specialized gaming hardware such as tablets and cellphones are used to play games, on-demand gaming service providers such as On-live [13] and Gaikai [6] exist to provide game-users with hardware independence. Users pay a monthly subscription fee, rent, or buy games. Games are stored and run on remote servers, and then they are delivered to the end-user. However, on-demand gaming service providers rely on a small number of datacenters, and our simulations show that end-users located far away from these datacenters experience an unacceptable amount of latency. By leveraging the computational power of peers outside of the datacenter, EdgeCloud is able to reduce latency and enhance the user-experience.

Peer-to-peer support for online games. In addition to on-demand gaming, peer-to-peer technology is used to distribute game computation. The Peers@Play project [12] investigates how peer-to-peer technology can be used to create interactive multi-user virtual environments. The goal of the Peers@Play project is to effectively distribute computation for rendering massive multi-user environments. The work in [39] presents a peer-to-peer solution for low-latency gaming, and it argues that peer-to-peer requires less hops than running games in the cloud. This work uses peer-to-peer middleware to distribute server functionality among several servers. Additionally, Sueselbeck et al. [40] present a peer-based and server-based approach that is used to propagate updates for an area of interest (part of the virtual world). Their hybrid system adapts itself dynamically to available system resources. Furthermore, Sueselbeck et al. [38] address the challenge of selecting optimal coordinators in a P2P-MMVE environment and discuss utility parameters for optimal coordinator selection. Although these works demonstrate that peer-to-peer technology is effective for distributing computation of massive multi-user environments, using cloud computing resources has many advantages. Firstly, on-demand gaming requires minimal changes to the games, in contrast to the peer-to-peer based solutions that require significant modifications. Secondly, by relying on many peers to offer service, the peer-to-peer solutions have significantly worse reliability than a cloud-based approach. Finally, the Peers@Play project is only applicable to multi-user virtual environments (for massively multi-player role playing games), whereas EdgeCloud is agnostic to the game genre.

Peer-to-Peer Overlays. Knutsson et al. [18] propose the use of P2P overlays to support Massively Multiplayer Games (MMGs). This work uses the idea that players in games have a limited area of interest; therefore, parts of a game are distributed over a P2P overlay in order to improve game availability. However, EdgeCloud aims to have users play on one peer as opposed to traversing different peers for different components of the game.

Video on Demand Technologies. The framework and infrastructure of on-demand gaming consists of doing the computation in the cloud and sending information (video) back to the client; therefore, technologies for on-demand gaming are related to technologies for video streaming. Niu et al. [32] presents a bandwidth auto-scaling facility that dynamically reserves resources from multiple data centers by predicting and estimating bandwidth demand as it tracks usage history. Similarly, Wu and Lui [43] present effective replication algorithms to reduce a server's workload and improve streaming quality.

However, the goals of such systems are to effectively utilize bandwidth for video streaming applications so that service providers incur a lower bandwidth cost. Although cost-reduction is vital for the sustainability of a system, one must also consider latency beyond start-up latency when accessing video. The aforementioned mechanisms ensure that video is effectively distributed; however, these mechanisms may not be applicable for the gaming environment. Particularly, games are far more latency-sensitive than video, and EdgeCloud addresses this issue by leveraging the locality of the participating peers.

Content distribution networks. The EdgeCloud architecture includes pushing games to various peers such that the peers can service end-users. One of the goals of EdgeCloud is to effectively distribute content (games) in such a manner that demand for all games is met. There have been various works in content distribution networks such as the ones mentioned in [33]. Previous related work also include research on Dynamic CDNs to address scalability problems (such as a large number of users wishing to access the same material) by changing the number, location and configuration of content hosts based on client demand, which is discussed in [33].

CYCLOPS [31] is a system that dynamically adjusts the bandwidth consumed by content servers by monitoring swarms of users. This work focuses on delivering a single piece of content to a set of clients, and its goal is to achieve cost savings for the content provider without noticeably impacting performance perceived by end users. This work also demonstrates that peer-assisted content distribution can be used to supplement the content provider's resources. In the same way, peers in EdgeCloud supplement computational resources in areas that are located away from datacenters.

8. CONCLUSIONS

In this paper, we demonstrated through a large scale measurement study that Amazon's EC2, the current market leader in cloud computing, is unable to meet the requirements of an emerging class of latency-sensitive multimedia applications. To address these requirements, we introduced EdgeCloud, a new hybrid platform that augments cloud datacenters with resources from end-hosts. The end-hosts provide additional geographic diversity that can be leveraged to significantly lower latency. These hosts are also equipped with specialized hardware, such as GPUs and fast memory, that are often in short supply in general-purpose datacenters.

We demonstrated the potentials of such architecture through measurements from PlanetLab nodes. We then described in detail the EdgeCloud architecture for managing the resources of both the datacenters and the peers. The architecture utilizes a logically centralized coordinator to determine which peers to include in its candidate pool, how the applications should be distributed, and which candidate should serve each client.

By incorporating resources from peers, EdgeCloud is able to serve around 90% of the poulation with 80 ms latency requirements although EC2 serves only 70% of the same population. EdgeCloud also authorizes the deployment of ultra-fast applications (50 ms response time) to two third of the population. We believe that by significantly increasing user coverage, EdgeCloud is an attractive alternative to cloud-only solutions for serving latency-sensitive multimedia applications.

9. REFERENCES

[1] Average u.s. upload speed: 435kbps. http://www.dslreports.com/shownews/Average-US-Upload-Speed-435kbps-96949.

[2] Bandwidth aid. http://www.networkedgraphics.org/blog/77-bandwidth.

[3] Bittorrent. http://www.bittorrent.com/.

[4] Dna@home. http://volunteer.cs.und.edu/dna/.

[5] Folding@home. http://folding.stanford.edu/English/HomePage.

[6] Gaikai open cloud gaming platform. http://www.gaikai.com.

[7] Gaikai will be fee-free, utilize 300 data centers in the us. http://www.joystiq.com/2010/03/11/gaikai-will-be-fee-free-utilize-300-data-centers-in-the-us/.

[8] Game consoles – september 2011 npd sales figure analysis. http://www.digital-digest.com/blog/DVDGuy/2011/10/15/game-consoles-september-2011-npd-sales-figure-analysis/.

[9] Gdc09 interview: Onlive founder steve perlman wants you to be skeptical. http://www.joystiq.com/2009/04/01/gdc09-interview-onlive-founder-steve-perlman-wants-you-to-be-sk.

[10] Hands-on: Spawn hd-720 brings your games to you. http://www.joystiq.com/2009/11/24/hands-on-spawn-hd-720-brings-your-games-to-you/.

[11] Maxmind - geoip python api. http://www.maxmind.com/app/python.

[12] The peers@play project. http://www.peers-at-play.org/.

[13] Play on-demand video games over the internet. http://www.onlive.com/.

[14] Realvnc - vnc® remote access and control technology. http://www.realvnc.com/.

[15] X.org foundation. http://www.x.org/.

[16] S. K. Barker and P. Shenoy. Empirical Evaluation of Latency-sensitive Application Performance in the Cloud. In *Proc. of ACM MMSys*, 2010.

[17] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. In *Networked Systems Design and Implementation*, San Francisco, CA, March 2004.

[18] H. L. Bjorn Knutsson, W. Xu, and B. Hopkins. Peer-to-Peer Support for Massively Multiplayer Games. In *IEEE INFOCOM*, volume 1, 2004.

[19] K. Chen, P. Huang, G. Wang, C. Huang, and C. Lei. On the Sensitivity of Online Game Playing Time to Network QoS. In *IEEE INFOCOM*, 2006.

[20] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, , and C.-L. Lei. Measuring the latency of cloud gaming systems. In *Proc. of ACM Multimedia*, 2011.

[21] M. Claypool and K. Claypool. Latency can kill: precision and deadline in online games. pages 215–222, 2010.

[22] M. Claypool and K. T. Claypool. Latency and player actions in online games. *Communications of The ACM*, 49:40–45, 2006.

[23] L. DiCioccio, R. Teixeira, and C. Rosenberg. Impact of home networks on end-to-end performance: controlled experiments. In *Proc. of ACM Sigcomm workshop on Home networks*, 2010.

[24] M. Dischinger, A. Haeberlen, P. K. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proc. of ACM Internet Measurement Conf. (IMC)*, 2007.

[25] A. Finamore, M. Mellia, M. Munafo, R. Torres, and S. G. Rao. YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience. In *Internet Measurement Conference*, 2011.

[26] S. Guha and P. Francis. Characterization and Measurement of TCP Traversal through NATs and Firewalls. In *Internet Measurement Conference*, 2005.

[27] S. Higginbotham. Smart TVs cause a net neutrality debate in S. Korea. Giga OM, Feb. 2012.

[28] T. Hoßfeld, R. Schatz, M. Varela, and C. Timmerer. Challenges of QoE Management for Cloud Applications. *IEEE Communications Magazine*, 50(4), Apr. 2012.

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

[30] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld. An Evaluation of QoE in Cloud Gaming Based on Subjective Tests. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2011.

[31] P. Michiardi, D. Carra, F. Albanese, and A. Bestavros. Peer-assisted content distribution on a budget. *Computer Networks*, 56(7), 2012.

[32] D. Niu, H. Xu, B. Li, and S. Zhao. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *INFOCOM*, 2012.

[33] A. Passarella. A survey on content-centric technologies for the current Internet: CDN and P2P solutions. *Computer Communications*.

[34] S. M. Rumble, D. Ongaro, R. Stutsman, M. Rosenblum, and J. K. Ousterhout. It's time for low latency. In *Proc. of HotOS*, 2011.

[35] D. Schlosser, B. Staehle, A. Binzenhöfer, and B. Boder. Improving the QoE of Citrix Thin Client Users. In *Proc. of IEEE ICC*, 2010.

[36] Y. Shoham and K. Leyton-Brown. *MULTIAGENT SYSTEMS Algorithmic, Game-Theoretic, and Logical Foundations*. Revision 1.1 edition, 2010.

[37] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *Internet Measurement Conference*, 2006.

[38] R. Sueselbeck, F. Heger, G. Schiele, and C. Becker. Challenges for Selecting Optimal Coordinators in Peer-to-Peer-Based Massively Multi-User Virtual Environments. In *International Conference on Computer Communications and Networks*, 2011.

[39] R. Sueselbeck, G. Schiele, and C. Becker. Peer-to-peer support for low-latency Massively Multiplayer Online Games in the cloud. In *Annual Workshop on Network and Systems Support for Games*, 2009.

[40] R. Sueselbeck, G. Schiele, S. Seitz, and C. Becker. Adaptive Update Propagation for Low-Latency Massively Multi-User Virtual Environments. In *International Conference on Computer Communications and Networks*, 2009.

[41] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. Broadband internet performance: a view from the gateway. In *Proc. of ACM Sigcomm*, 2011.

[42] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. Better never than late: meeting deadlines in datacenter networks. In *Proc. of ACM Sigcomm*, 2011.

[43] W. Wu and J. C. S. Lui. Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation. In *IEEE INFOCOM*, 2011.