A Cognitive Platform for Mobile Cloud Gaming

Wei Cai[1], Conghui Zhou[2], Victor C.M. Leung[1], Min Chen[3]
[1]Department of Electrical and Computer Engineering, The University of British Columbia, Canada
[2] We Software Limited, Hong Kong
[3] School of Computer Science and Technology, Huazhong University of Science and Technology, China
[1] {weicai, vleung}@ece.ubc.ca, [2] neio.zhou@gmail.com, [3] minchen2012@hust.edu.cn

***Abstract*—Mobile cloud gaming provides a whole new service model for the video game industry to overcome the intrinsic restrictions of mobile devices and piracy issues. However, the diversity of end-user devices and frequent changes in network quality of service and cloud responses result in unstable Quality of Experience (QoE) for game players. A cognitive cloud gaming platform, which could overcome the above problem by learning about the game players environment and adapting the cloud gaming service accordingly, does not currently exist. To fill this void, we design and implement a component-based gaming platform that supports click-and-play, intelligent resource allocation and partial offline execution, to provide cognitive capabilities across the cloud gaming system. Extensive experiments have been performed to show that intelligent partitioning leads to better system performance, such as overall latency.**

***Index Terms*—cognitive; platform; mobile; cloud; game**

I. Introduction

Well recognized as the next generation computing infrastructure, the cloud is considered as a provider of scalable storage and computational resources, which supports various types of online services for end users. Researchers have shown great interest in migrating all kinds of applications from fixed servers to cloud platforms, so called Everything as a Service. In this specific environment, cloud gaming is attracting remarkable attention in both industry and academia recently.

Hosting games in the cloud brings many advantages. It is maintenance free and has nominal (almost negligible) costs for service provisioning compared to the costs of the hardware and gaming software that one has to pay for in personal computer (PC) or console gaming. Since cloud-based services support cross-platform solutions, cloud gaming can replace those PlayStations or Xboxes with its ability to stream games to all browsers regardless of whether it is running in a mobile device or a PC. Moreover, the game developing companies like the idea of cloud games, since it is the best solution for anti-piracy. The cloud-based gaming model changes the distributions of the games into providing gaming services, which creates continuous profits.

The situation will be even more interesting when the user terminals are mobile devices. With the approach of offloading [1], the mobile terminal utilizes the rich resources from the cloud to enhance its functionality and prolong the battery lifetime through better energy efficiency, and therefore, to overcome the intrinsic constraints of mobile devices, such as incompatible operating system, limited storage, insufficient computational capacity and battery drain problem. For instance, you can now play World of Warcraft on an Apple iPad!

OnLive[1], Gaikai[2] and G-Cluster[3] are the most famous commercial providers of gaming services on-demand. Their cloud gaming model, so called cloud video gaming systems, takes advantage of cloud computing to overcome the limitations of mobile devices. Video games are hosted in their private cloud servers and the gaming video frames are encoded by the streaming server before being transmitted over the Internet to the clients, such as interactive televisions, desktop PCs, smartphones, etc. In reverse, the players' inputs are delivered to a cloud server and accepted by the game content server directly [2]. In this context, the cloud is intrinsically an interactive video generator and streaming server, while the mobile devices function as the event controllers and video receivers that can run sophisticated games despite their restricted hardware. This results in a longer battery life for the device and longer gaming times for the user at the expense of higher consumption of communication resources. Nevertheless, those cloud-based video games still suffered from the bandwidth-bottleneck of Internet access. The bandwidth constraints restrict the bit rate of gaming videos, while the jitter and delay affect the quality of experience (QoE) for the players. Therefore, technologies regarding real-time video rendering, compressing, and transmission quality of service (QoS) control become the most critical issues for system design [3][4]. Another approach to provide cloud gaming services is browser game [5], which always relies on online social network sites with a massive number of users (e.g., FarmVille on Facebook). In a typical browser game, the gaming contents, including data and all of the gaming procedures, are stored and executed within the cloud, while the gaming graphics and videos are rendered by the browser, instructed by the returns from the cloud server. Compared to the normal solution of cloud-based video games, browser games leave the presentation functionalities to the browsers, in order to eliminate the high bandwidth consumption for gaming video transmission. According to the study of above two types of cloud gaming, we identify that browser game is more efficient in the use of communication resources, at the expense of a heavier computation load in the

[1]http://www.onlive.com/
[2]http://www.gaikai.com/
[3]http://www.g-cluster.com/

978-0-7695-5095-4/13 $31.00 © 2013 IEEE
DOI 10.1109/CloudCom.2013.17

IEEE computer society

user device.

In other words, the main distinction of the two existing cloud-based gaming model is the proportion of offloading. However, both of them are still of insufficient flexibility, given the various scenarios of playing cloud games on mobile devices. In this work, we investigate and develop a cognitive and flexible cloud gaming platform, which learns about the game players environment (i.e., the combination of terminal and access network) and adapts the cloud gaming service to these evaluations. With the proposed platform, we explore a potential solution that enables players to continue their gaming sessions while the mobile terminal is temporarily disconnected from the cloud. The outline of the paper is as follows. We review related work in Section II. We then discuss the design objectives of the cognitive platform for mobile cloud games in Section III. The design and implementation of the cognitive platform are described in Section IV and V, respectively. Experiments on optimizing the overall latency and enabling partial offline execution are conducted in Section VI. Section VII concludes the paper.

II. Related Work

A. QoE for Cloud Gaming

Maintaining an acceptable QoE is the main criteria of the proposed cognitive platform for mobile cloud gaming. Various subjective user studies have been conducted to demonstrate the relationship between cloud gaming QoE and QoS, including game genres, video encoding factors, conditions of the wireless network [2], CPU load, memory usage, and link bandwidth utilization [6], response latency and the game's real-time strictness [7], network characteristics (bit rates, packet sizes, and inter-packet times) [8], number of users [9], and an empirical network traffic analysis of On-Live and Gaikai [10].

B. Cloud Gaming Architecture

Adaptive mobile video streaming architectures have been widely studied for mobile cloud games. The framework in [11] monitors the end-to-end network status and alters the video encoding parameters for cloud games accordingly. This work considers dynamic video encoding adaptation focusing on improving the user-perceived quality of a Gaming-on-Demand service, which is a highly demanding interactive multimedia application based on a client-server infrastructure and video streaming. Similarly, a recent work [4][12] proposes a Remote Server Based Mobile Gaming (RSBMG) architecture and develops a set of application layer optimization techniques to ensure acceptable gaming response time and video quality in the remote server based approach. The techniques include downlink gaming video rate adaptation, uplink delay optimization, and client play-out delay adaptation.

C. Partitioning Solution

To facilitate intelligent resource allocation, the cloud games should support dynamic partitioning between cloud and mobile terminal. There has been some work on partitioning of mobile applications. [13] first introduces a K-step algorithm as a dynamic solution where the partition is calculated on-the-fly, once a mobile connects and communicates its resources. Furthermore, according to [14], there is no single partitioning that fits all due to environment heterogeneity (device, network, and cloud) and workload. Consequently, they proposed a system that can seamlessly adapt to different environments and workloads by dynamically instantiating what partitioning to use between weak devices and clouds. An implementation [15] called CloneCloud is a flexible application partitioner and execution runtime that enables unmodified mobile applications running in an application-level virtual machine to seamlessly off-load part of their execution from mobile devices onto device clones operating in a computational cloud. However, these work requires the application to be completely installed in both the mobile terminal and the virtual machine residing in the cloud.

III. Objectives of the Proposed Platform

Cognitive systems [16] are attractive for the heterogeneous environment we are considering. The situation-aware architecture of a cognitive system monitors and assesses the working environment to predict and make decisions for the providing services, and refine future decisions by learning from the achieved results. To the best of our knowledge, a QoE-oriented cognitive system is not available for cloud gaming systems. To provide cognitive capabilities across the cloud gaming system, we need to overcome the diversity of user-end devices and frequent changes in network QoS and cloud responses. We use the concept of cognitive system design (i.e., act, learn, adapt) to realize our proposed situation-aware cloud gaming platform. Our objective is to develop an architectural framework that is cognitive of resources and characteristics of the cloud, the access network, and the end-user devices, to enable dynamic utilization of these resources.

As shown in Fig. 1, we envision a cognitive platform with a novel capability to learn about the game players environment (i.e., the combination of terminal and access network) and adapt the running of the game to maintain an acceptable QoE. The proposed platform should be able to:

- *support click-and-play without game installation:* in contrary to most of the partitioning and offloading work for mobile cloud applications [13][14][15], the game applications designed for the proposed platform do not need to be installed in the mobile devices in advance. The players are able to play their favorite games immediately when they connect to the gaming platform.
- *support intelligent resource allocation:* the platform is able to measure, evaluate and predict the overall system performance, including CPU load, memory usage, link bandwidth utilization, and specialized application-layer metrics, such as the number of players, spatial distribution of the player population, or game state computation delay; afterwards, the platform is able to intelligently adapt its gaming services to system performance, such as changing network conditions (QoS), to keep user QoE above a prescribed threshold.

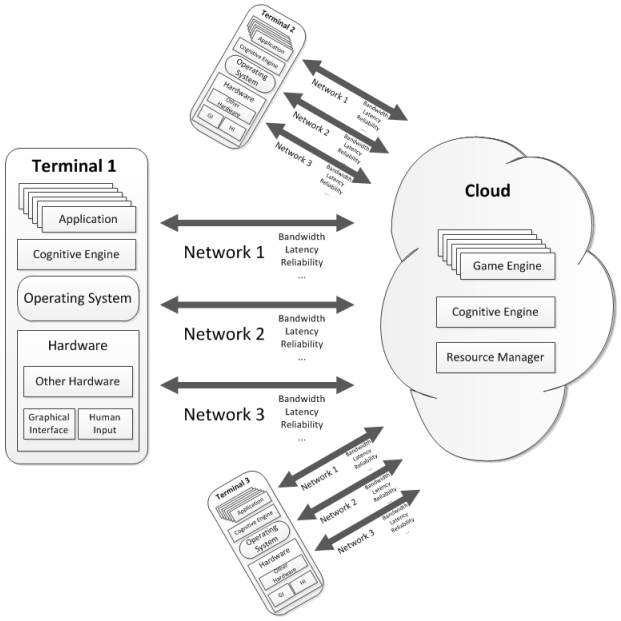

Fig. 1. Architecture Framework for Cognitive Mobile Cloud Gaming

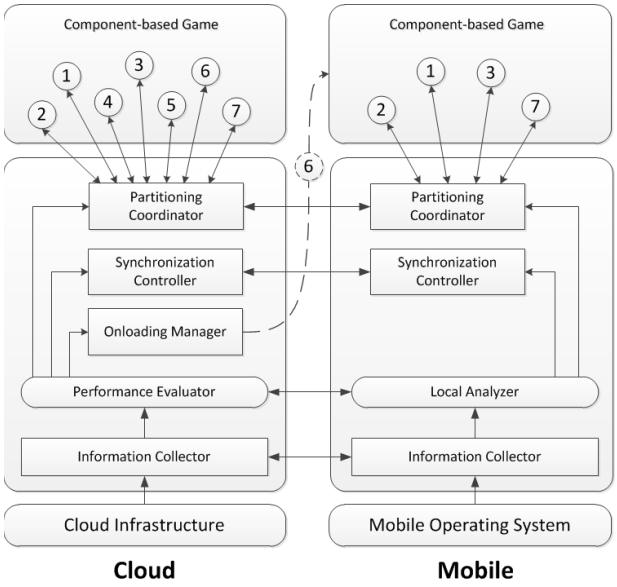

Fig. 2. Cognitive Platform for Mobile Cloud Gaming

- *support gaming without network connection for special cases:* strong dependency on network connection is the intrinsic problem of cloud gaming system. Nevertheless, even in multi-player games, players still spend a significant number of time to interact with Non-Player Characters (NPCs) by themselves, according to the categorization of gaming scenes in [17]. The platform is able to support offline gaming for these scenes, in order to eliminate the occasional disconnection problem for mobile devices.

IV. Design of Cognitive Platform

A. Elements of Cognitive Platform

Given the above requirements, we define the main building blocks of the architectural framework on both the cloud-side and terminal-side, and identify the prevalent standards that are applicable to the interfaces between these building blocks.

Fig. 2 illustrates the elements of the proposed cognitive platform and their relationships. *Information Collector*s on both the cloud and mobile sides monitor resource usage at cloud, access network and mobile terminal. These surveillance data are reported to the *Performance Evaluator* and *Local Analyzer*. Note that the *Performance Evaluator* is able to guide the procedure in *Local Analyzer*, while the *Local Analyzer* reports its results to *Performance Evaluator* periodically for further evaluations. Games designed for the cognitive platform consist of a number of inter-dependent game components. These components are able to migrate from the cloud to the mobile terminal via a network, under the instruction of the *Onloading Manager*. As the message gateway between components, the *Partitioning Coordinator* intelligently selects destination components, locally or remotely, to achieve dynamic resource allocations. The *Synchronization Controller* is designed to guarantee the synchronization of data in identical components distributed in the cloud and mobile terminals. Note that the *Onloading Manager*, *Partitioning Coordinator* and *Synchronization Controller* are manipulated by the *Performance Evaluator* and *Local Analyzer* for the purpose of maintaining an acceptable QoE for players.

B. Onloading

Since the cognitive platform supports *click-and-play*, none of the game component exists in the mobile client at the beginning of a gaming session. In this case, the cloud server shall be capable to transmit executable components to the mobile terminal, in order to enable dynamic resource allocation. We employ the concept of mobile agent [18] to realize this process. A mobile agent is a composition of computer software and data, which is able to migrate (move) from one computer to another autonomously and continue its execution in the destination computer. In this context, the game components are encapsulated as mobile agents and dispatched from the cloud to mobile devices.

The onloading process could either be performed before gaming session starts or be running in the background during the gaming session. It is scheduled by the *Onloading Manager*, which assigns each game component a priority based on the overall assessment on the particular component. Similar to application's consumption graph in [13], Fig. 3 denotes the dependency of game components as a directed graph $G = \{C, E\}$, where every vertex in C is a component c_i and every edge $e_{i,j}$ in E is a dependency between c_i and c_j . Each component c_i is characterized by following parameters:

- r_i: the resource consumption of c_i on a mobile device

- s_i: the size of the compiled code of c_i
- $in_{j,i}$: the amount of data that c_i takes in input from c_j
- $out_{j,i}$: the amount of data that c_i sends in output to c_j
- $fin_{j,i}$: the frequency that c_i takes data input from c_j
- $fout_{j,i}$: the frequency that c_i sends data output to c_j

In this context, the priority p_i for the ith game component is modeled by the following function:

$$p_i = f(r_i, c_i, \sum_j fin_{j,i} \cdot in_{j,i}, \sum_j fout_{i,j} \cdot out_{i,j}) \quad (1)$$

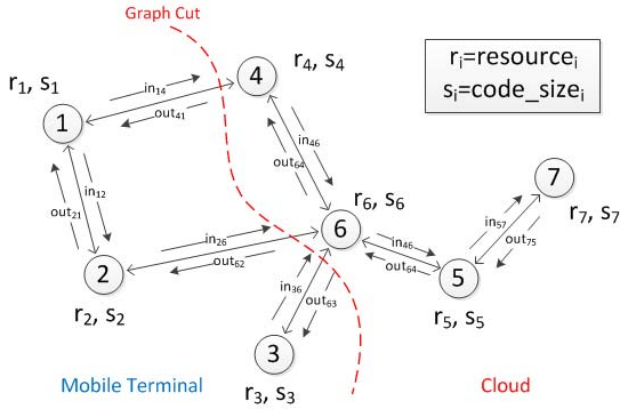

Fig. 3. Components Partitioning for Proposed Cognitive Platform

Nevertheless, the priority of a game component is also associated with its function. Some key components should have a higher priority in the onloading process, since they provide featured benefits in the mobile terminal. We will discuss more details on this topic in Section IV-E.

C. Partitioning

Once the mobile terminal fetches the game components from the cloud, *Partitioning Coordinator* should work with *Performance Evaluator* and *Local Analyzer* to solve the dynamic partitioning problem, in order to provide a QoE-oriented resource optimization. In our proposed framework, all input and output data from the components are sent to the *Partitioning Coordinator*, which provides a routing service for invoke messages by intelligently selecting the destination components when an application cycle is determined.

As depicted in Fig. 3, the partitioning problem intrinsically seeks to find a cut in the consumption graph such that some components of the game execute on the client side and the remaining ones on the cloud side. The optimal cut maximizes or minimizes an objective function O, which expresses the general goal of a partition, e.g., minimizing the end-to-end interaction time between the mobile terminal and the cloud, minimizing the amount of exchanged data, or minimizing the latency in a gaming session.

In designing the objective function O, we need to satisfy the user's QoE requirement, including resource constraints in mobile devices and tolerable latency for each interaction cycle:

$$\sum_{k \in mobile} r_k \leq R_{Mobile} \quad (2)$$

Denote components set S_c involved in each gaming interaction cycle, for all $i, j \in S_c$,

$$\sum_{i \in mobile, j \in cloud} \frac{(fin_{j,i} \cdot in_{j,i} + fout_{i,j} \cdot out_{i,j})}{B} + T_{S_c} \leq T_m \quad (3)$$

where R_{Mobile} represents the available resource in the mobile device, T_{S_c} denotes the transaction delay of the components, B denotes the average bandwidth, and T_m denotes the average maximum delay that the players can tolerate.

D. Synchronization

The remote distribution of game components results in the data asynchronization problem. To address this problem, the *Synchronization Controller* is employed to update all parameters in the gaming environment. However, the synchronization process also introduces a non-negligible network overhead. Consequently, we design the synchronizing mechanism following the principle of "Sync-Only-If-Necessary" to minimize the transmission cost.

E. Partial Offline Execution

One of the most critical problems for mobile cloud gaming is the conflict between strong network dependency and unstable network connectivity. In all existing cloud gaming frameworks, the gaming session will be suspended or even destroyed, once the mobile device loses its network connection to the cloud. In this work, we explore a novel solution for the temporal disconnection issues in special cases.

As a matter of fact, players are not interacting with each other all the time during the gaming session, even in those multi-player games. In a typical Massively Multiplayer Online Role-Playing (MMORPG) game, the avatar spends a remarkable amount of time in monster hunting by him/her-self, for the sake of level-up and outfit gathering [17]. The players will never be happy if they lose valuable items in the battlefield due to the network access problem. Our proposed platform focus on the solution to these scenarios.

Fig. 4 demonstrates a case of redirection service provided by the *Partitioning Coordinator*: the mobile-executed component 7 is trying to activate component 3 in the cloud with an output message, while the network connection to the cloud is temporarily lost. Rather than suspending the gaming session, the *Partitioning Coordinator* locates component 3 on the mobile device and redirects the output message to this local copy. Thus, the player will not be disturbed by the temporary disconnection of Internet. This approach is called "Partial Offline Execution".

Note that once the mobile device recovers its network access, the *Synchronization Controller*, which is aware of the data modifications, will perform data synchronization with the cloud server.

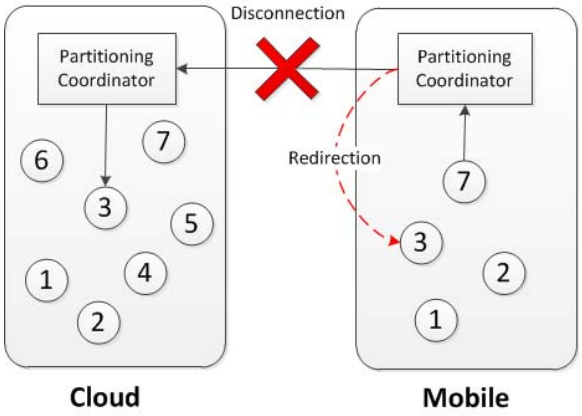

Fig. 4. Redirection Service in Disconnection to the Cloud

As has been discussed in Section IV-B, the components that support partial offline execution will be assigned higher priority levels in the onloading procedure.

V. Implementation

A. Enabling Technologies

To implement the proposed cognitive platform for mobile cloud games, we seek enabling technologies that facilitate the migration and partitioning of game components. JavaScript is adopted as the programming language, which is originally implemented as a part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed. More recently, however, its use has become common in both game development and the creation of desktop applications.

Node.js[4] is a server-side software system designed for writing scalable Internet applications, notably web servers. Programs on the server side are written in JavaScript, which enables web developers to create an entire web application in JavaScript, both server-side and client-side. This feature facilitates the game components, JavaScript gaming code in this context, to migrate from the cloud to user-end, and to be executed on cloud server and client as a mobile agent.

For the mobile client, we embed a WebKit-based browser into the cognitive engine for parsing and executing the JavaScript mobile agent from the cloud server. In our implementation, the WebKit browser is built on Android smartphone. However, all mobile operating systems supporting browsers are able to run our cognitive platform after a small number of modification. We are also looking for alternative solutions to implement the mobile client as native applications on JavaScript. As the state-of-the-art, Microsoft already supports native application development with JavaScript on its metro-style interface.

[4]http://nodejs.org/

B. Application Programming Interface

As a game developing platform, our cognitive platform provides a set of application programming interfaces (APIs), which facilitates the game developers to create their game applications without the knowledge of intelligent onloading and partitioning schemes. To trigger the partitioning, the developers only have to add a "$$" mark before the name of the components when these components are invoked in the code (e.g. $value = \$\$component_X(\{msg : msg\});$). Our cognitive platform then recognizes these components and performs dynamic partitioning as predetermined.

C. Onloading Solution

Our implementation enables three types of onloading: i) the system administrator is able to onload specific components to the client manually from the Cloud Configuration Center (as shown in Fig.5); ii) the platform is able to randomly dispatch selected components to the client, when the network connection between the cloud and mobile devices are idle; iii) the client is able to request and fetch specific components from the cloud, once the optimal solution is determined and some of the components required in the client are still missing. In our experiments, since the code length of the components are short, their transmission cost is negligible in the experimental results, we adopt the third mode.

D. Adaptive Partitioning Solution

For the applications running on the cognitive engine, we need to measure the computational cost of each component and transmission cost between the components. Based on the measurement, once the client request an interaction, the platform enumerates all possible partitioning solutions for the involving components to estimate the overall resource cost for all solutions. Based on the estimation, the platform creates a cost table of candidate solutions, from which the optimal solution is determined. Table I shows an example of cost table of candidate solutions for 7 components.

TABLE I
Cost Table of Candidate Solutions

Solution	On Cloud	On Client	Transmission	Cloud	Client
1	1,2,3	4,5,6,7	30bps	22MB	32MB
2	1,2,4,5	3,6,7	60bps	30Mb	24MB
3	1,3,4,5,6,7	2	10bps	50Mb	4MB
...	...	...	...	...	...

E. Implementation Screenshots

Fig. 5 illustrates a screenshot of Cloud Configuration Center. The configuration center provides a graphical interface for the system administrator to monitor the system performance of each cloud-client pair, such as network bandwidth, usage of client CPU, memory, battery, etc. In addition, the configuration center also visualizes the real-time onloading and partitioning status and enables the administrator to manipulate these actions by simple clicks.

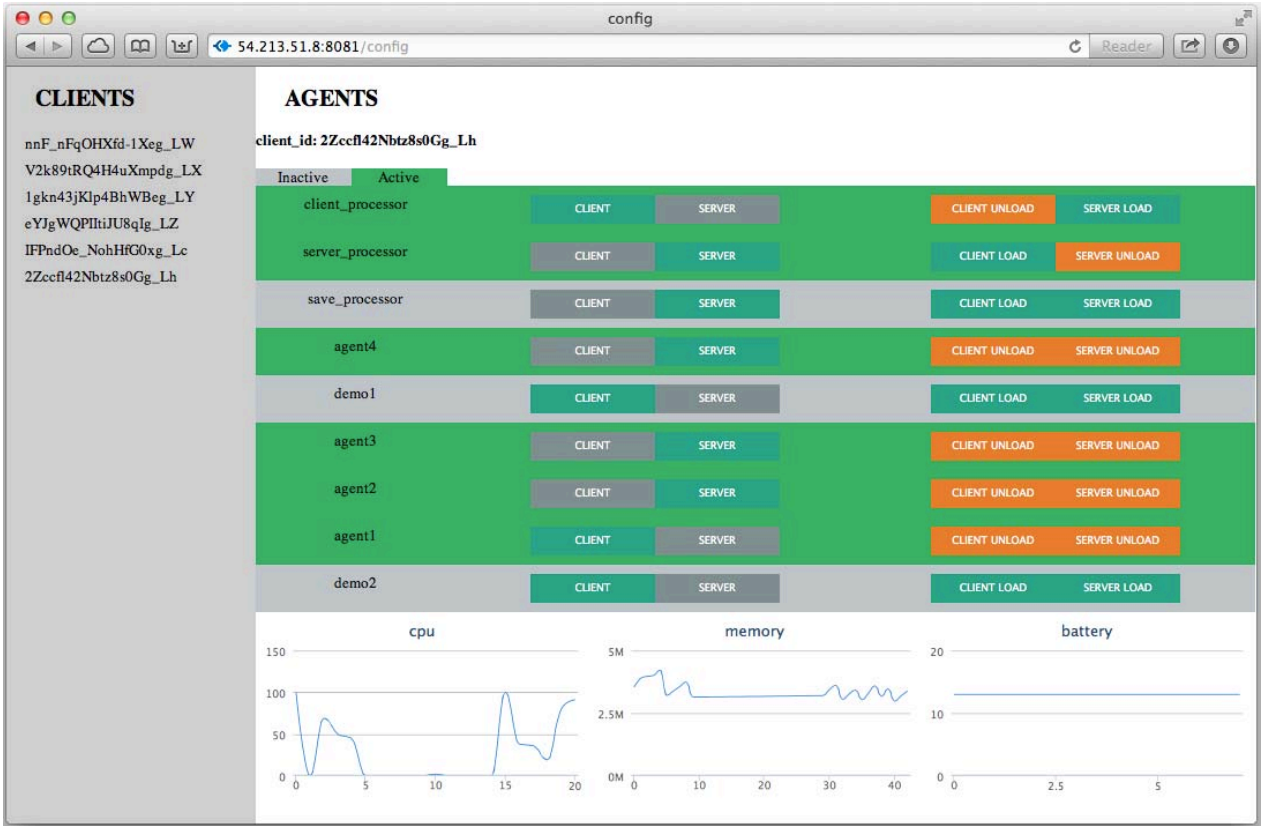

Fig. 5. Screenshot of Cloud Configuration Center

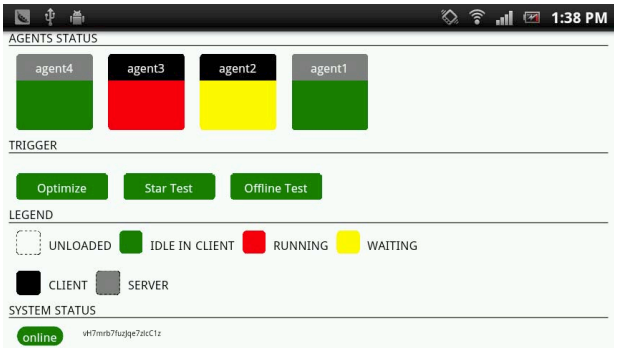

Fig. 6. Screenshot of Monitoring Application on Android

Fig. 6 depicts a demo of component monitoring application on Android[5] smartphone. From the monitoring interace, we are able to check the real-time status of each component, including their running environment (cloud or client), onloading state (unloaded or onloaded), execution status (idle, running, or waiting), and network connectivity (online or offline). Trigger buttons are implemented to start our designed experiments.

[5]http://www.android.com/

VI. Experiments

A. Experimental Setup

To validate the performance of our cognitive platform, we set up the following experiments. For cloud side, we choose Amazon Elastic Compute Cloud (Amazon EC2)[6] as cloud host, which is a web service that provides resizable compute capacity in the cloud. Ubuntu[7] Server 13.04 is set up as operating system on the cloud and node.js version 0.8.9 is installed as the server engine. For mobile client, we utilize a LG Revolution smartphone with Android 2.3.4 as operating system. All experiments are conducted in the Vancouver Campus of The University of British Columbia (UBC), through the WiFi connection provided by UBC Information Technology.

B. Latency-Oriented Optimization

Latency is one of the most importance impacting factors for user experience in gaming. In this experiment, we simulate the interaction procedure between mobile client and the cloud, exploring a optimization solution to minimize the latency for each interaction.

To facilitate the experiments, we design a sequence of 4 components, which conducts simple loops and calls each

[6]http://aws.amazon.com/ec2/
[7]http://www.ubuntu.com/

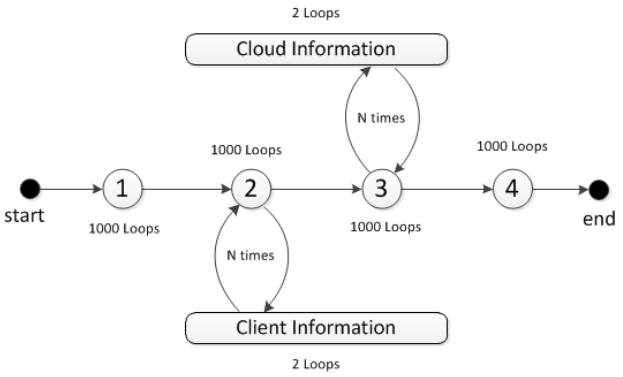

Fig. 7. A Cycle of Experiment Demo

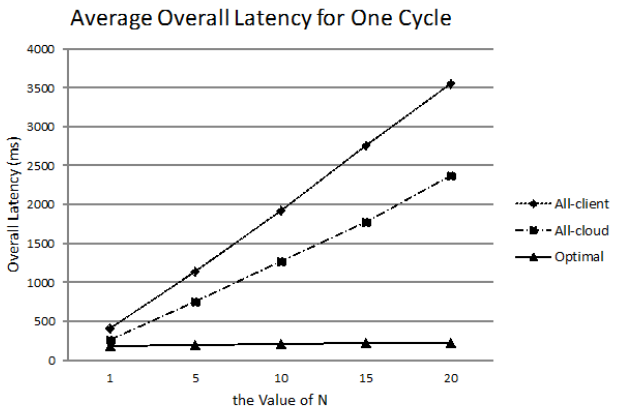

Fig. 8. Average Overall Latency for One Cycle

other as shown in Fig.7. We denote the procedure from component 1 to component 4 as an application cycle. In order to simulate the data collection from controllers and sensors in the mobile devices and information exchange with other users in the cloud, we design a Client Information module and a Cloud Information module, which are invoked by component 2 and component 3 for N times, respectively. Note that, if component 2 is executing on the cloud, the remote call to the Client Information module will leads to network transmissions. Otherwise, only local processing is required. In contrast, the case for component 3 is opposite.

In this specific task, the adaptive partitioning solution can be simplified: instead of measuring and estimating computational cost and transmission cost of each candidate solution, we use overall latency, including computation latency and transmission latency, as the unique criteria to select the optimal solution. Consequently, our cognitive platform scans the table and chooses the optimal solution with minimal overall latency. The simplified version of of cost table of candidate solutions is as follows:

TABLE II
LATENCY-ORIENTED COST TABLE OF CANDIDATE SOLUTIONS

Solution	On Cloud	On Client	Overall Latency
1	1,2,3	4,5,6,7	132ms
2	1,2,4,5	3,6,7	2400ms
3	1,3,4,5,6,7	2	1240ms
...	...	...	...

We compare the optimal solution selected by our cognitive platform with two conventional schemes: all-client and all-cloud, where all-client indicates all agents are executed on the mobile devices, and the all-cloud solution put all agents in the cloud. During the simulation, the parameter N, defined as the number of interactions with cloud and client information, is creased from 1 to 20 by the step size of 5. For each scenario of the particular schemes, extensive simulations are performed to retrieve the average value.

As shown in Fig. 8, even with a very small value of N the optimal solution achieves a significant gain in terms of overall latency. It is less than half of those all-client and about 67% of all-cloud when $N = 1$. More importantly, along with the increase of N, the overall latency for all-client and all-cloud mode go up to a very high value. For $N = 20$, all-client scheme suffers more than 3500 ms latency, and all-cloud mode also receives around 2500 ms delay. In contrast, the performance of optimal scheme remains stable. Even with $N = 20$, the latency only grows 25.3% (from 176.5 ms to 221.1 ms), which is 9.3% of all-cloud and 6.2% of all client.

C. Partial Offline Execution Example

To support partial offline execution is a highlight feature of the proposed cognitive platform. In this experiment, we simulate a network disconnection scenario to demonstrate the ability to support partial offline execution.

Note that, our implementation shall be able to handle the pending-disconnection case, which happens when a client component A is pending for a remote component B executing in the cloud. In this case, the lost of network connectivity will prevent the component B to invoke the modules in the mobile client, thus, makes the application procedure interrupted. To overcome this problem, our implementation creates a network detector to send disconnect notification to the pending component. As a reaction, the pending component rollback its status and send another invoke to the local version of destination component, if it exists. In addition, once the network connection resumes, our platform shall be able to recognize the status and switch back to the partitioning solutions predetermined by the platform.

Fig. 9 depicted an example of partial offline execution. We create a component named "Demo 1" to be executed in the cloud and a component name "Demo 2" running in the mobile devices. Demo 2 keeps invoking Demo 1 and receive returns from Demo 1. Fig. 9 a) shows the running status when the client is connecting to the cloud. We can see the Demo 1 on the mobile client is idle, since Demo 2 is calling the cloud copy of Demo 1. However, the mobile devices goes offline during another invoke process from Demo 2, as illustrated in Fig. 9 b). Our platform detects the connection lost and triggers

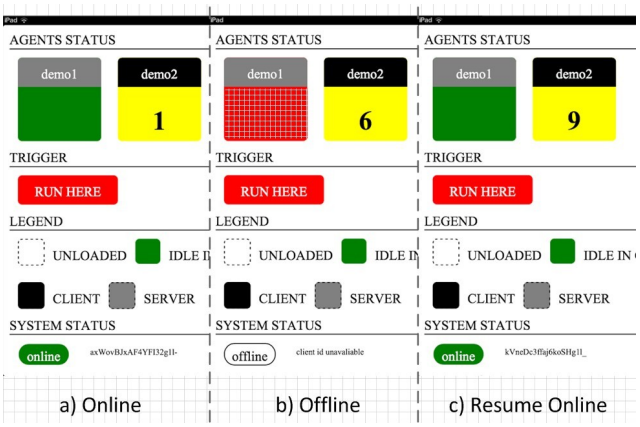

Fig. 9. Example of Partial Offline Execution

the client copy of Demo 1 to solve the problem. Note that, Demo 1 was onloaded to the client before the network breaks, otherwise, the invoke redirection can be conducted. Fig. 9 c) visualized that, when network recovers, the client reconnect to the cloud and Demo 1 is offloaded to the cloud as we specified in the platform again.

VII. Conclusion and Future Work

In this paper, we have proposed a cognitive, flexible and promising gaming platform for mobile cloud gaming, which supports click-and-play, intelligent resource allocation and partial offline execution. Unlike previous work on cloud games, we have proposed a component-based game structure and designed specific mechanisms to facilitate the envisioned objectives, such as dynamic onloading process, partitioning, synchronization and redirection services for partial offline execution. We discussed the enabling technology and implemented the proposed platform as a pure JavaScript solution. Extensive experiments were performed to show that the adaptive partitioning is able to provide optimal solution in terms of overall latency. For the future work, we focus on the following directions: i) To better scheduling and resource management, we need to measure the computational performance of game components, both in cloud and mobile devices. In addition,the communication between these components shall also be measured. ii) The platform shall accurately, timely and efficiently measure, evaluate and predict the real-time system environment, to provide a reference for QoE-oriented adaption. iii) Instead of overall latency, we shall consider more sophistic models with more impact factors, such as computational performance, network bandwidth, battery percentage, etc.

Acknowledgement

This work is supported by a University of British Columbia Four Year Doctoral Fellowship and by funding from the Natural Sciences and Engineering Research Council.

References

[1] Kun Yang, S. Ou, and Hsiao-Hwa Chen, "On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications," *Communications Magazine, IEEE*, vol. 46, no. 1, pp. 56–63, 2008.

[2] Shaoxuan Wang and Sujit Dey, "Modeling and characterizing user experience in a cloud server based mobile gaming approach," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, 30 2009-dec. 4 2009, pp. 1 –7.

[3] Min Chen, "Amvsc: A framework of adaptive mobile video streaming in the cloud," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, 2012, pp. 2042–2047.

[4] Shaoxuan Wang and Sujit Dey, "Rendering adaptation to address communication and computation constraints in cloud mobile gaming," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010, pp. 1–6.

[5] Juha-Matti Vanhatupa, "Browser games: The new frontier of social gaming," in *Recent Trends in Wireless and Mobile Networks*. 2010, vol. 84 of *Communications in Computer and Information Science*, pp. 349–355, Springer Berlin Heidelberg, 10.1007/978-3-642-14171-3-30.

[6] Michael Jarschel, Daniel Schlosser, Sven Scheuring, and Tobias Hossfeld, "An evaluation of qoe in cloud gaming based on subjective tests," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, 30 2011-july 2 2011, pp. 330 –335.

[7] Yeng-Ting Lee, Kuan-Ta Chen, Han-I Su, and Chin-Laung Lei, "Are all games equally cloud-gaming-friendly? an electromyographic approach," in *Proceedings of IEEE/ACM NetGames 2012*, Oct 2012.

[8] Mark Claypool, David Finkel, Alexander Grant, and Michael Solano, "Thin to win? network performance analysis of the onlive thin client game system," in *Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on*, 2012, pp. 1–6.

[9] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on*, 2012, pp. 1–6.

[10] M. Manzano, J.A. Hernandez, M. Uruena, and E. Calle, "An empirical study of cloud gaming," in *Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on*, 2012, pp. 1–2.

[11] Sari Jarvinen, Jukka-Pekka Laulajainen, Tiia Sutinen, and Sami Sallinen, "Qos-aware real-time video encoding how to improve the user experience of a gaming-on-demand service," in *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, jan. 2006, vol. 2, pp. 994 – 997.

[12] Shaoxuan Wang and Sujit Dey, "Addressing response time and video quality in remote server based internet mobile gaming," in *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, 2010, pp. 1–6.

[13] Ioana Giurgiu, Oriana Riva, Dejan Juric, Ivan Krivulev, and Gustavo Alonso, "Calling the cloud: enabling mobile phones as interfaces to cloud applications," in *Proceedings of the ACM/IFIP/USENIX 10th international conference on Middleware*, Berlin, Heidelberg, 2009, Middleware'09, pp. 83–102, Springer-Verlag.

[14] Byung-Gon Chun and Petros Maniatis, "Dynamically partitioning applications between weak devices and clouds," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, New York, NY, USA, 2010, MCS '10, pp. 7:1–7:5.

[15] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*, New York, NY, USA, 2011, EuroSys '11, pp. 301–314, ACM.

[16] Pat Langley, John E. Laird, and Seth Rogers, "Cognitive architectures: Research issues and challenges," *Cognitive Systems Research*, vol. 10, no. 2, pp. 141 – 160, 2009.

[17] Wei Cai, Xiaofei Wang, Min Chen, and Yan Zhang, "Mmoprg traffic measurement, modeling and generator over wifi and wimax," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, dec. 2010, pp. 1 –5.

[18] Danny B. Lange and Oshima Mitsuru, *Programming and Deploying Java Mobile Agents Aglets*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1998.