Information **2015**, *6*, 228-245; doi:10.3390/info6020228

OPEN ACCESS

information

ISSN 2078-2489

www.mdpi.com/journal/information

Article

CIMS: A Context-Based Intelligent Multimedia System for Ubiquitous Cloud Computing

Abhilash Sreeramaneni [1], Hyungjin Im [1], Won Min Kang [1], Chan Koh [1] and Jong Hyuk Park [1,2,*]

[1] Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 139-743, Korea; E-Mails: abhilashsreeramaneni@gmail.com (A.S.); imhj9121@seoultech.ac.kr (H.I.); wkaqhdsk0@seoultech.ac.kr (W.M.K.); chankoh@seoultech.ac.kr (C.K.)

[2] Department of Interdisciplinary Bio IT Materials, Seoul National University of Science and Technology, Seoul 139-743, Korea

* Author to whom correspondence should be addressed; E-Mail: jhpark1@seoultech.ac.kr; Tel.: +82-2-970-6702; Fax: +82-2-977-9441.

Academic Editors: Young-Sik Jeong and Jianhua Ma

Received: 4 March 2015 / Accepted: 6 May 2015 / Published: 4 June 2015

Abstract: Mobile users spend a tremendous amount of time surfing multimedia contents over the Internet to pursue their interests. A resource-constrained smart device demands more intensive computing tasks and lessens the battery life. To address the resource limitations (*i.e.*, memory, lower maintenance cost, easier access, computing tasks) in mobile devices, mobile cloud computing is needed. Several approaches have been proposed to confront the challenges of mobile cloud computing, but difficulties still remain. However, in the coming years, context collecting, processing, and interchanging the results on a heavy network will cause vast computations and reduce the battery life in mobiles. In this paper, we propose a "context-based intelligent multimedia system" (CIMS) for ubiquitous cloud computing. The main goal of this research is to lessen the computing percentage, storage complexities, and battery life for mobile users by using pervasive cloud computing. Moreover, to reduce the computing and storage concerns in mobiles, the cloud server collects several groups of user profiles with similarities by executing K-means clustering on users' data (context and multimedia contents). The distribution process conveys real-time notifications to smartphone users, according to what is stated in his/her profile. We considered a mobile cloud offloading system, which decides the offloading actions to/from cloud servers. Context-aware decision-making (CAD) customizes the mobile device

performance with different specifications such as short response time and lesser energy consumption. The analysis says that our CIMS takes advantage of cost-effective features to produce high-quality information for mobile (or smart device) users in real time. Moreover, our CIMS lessens the computation and storage complexities for mobile users, as well as cloud servers. Simulation analysis suggests that our approach is more efficient than existing domains.

Keywords: context-based intelligent multimedia system; mobile users; ubiquitous cloud computing; K-means clustering; mobile-cloud offloading system; context-aware decision; pervasive cloud computing

1. Introduction

Over the last decade, Internet usage around the world has been increasing day by day. Internet users daily browse billions of multimedia webpages and face difficulties in finding the content in which they are interested. Ubiquitous computing plays a vital role in smart devices (*i.e.*, smartphones, iPads, tablets, *etc.*) by providing strong data networks (3G/4G) for mobile users to access fast Internet anywhere at any time. As mentioned above, mobile cloud computing has grown in popularity for many reasons, such that today smart devices are hampered by limitations, e.g., memory space, CPU usage, and battery life, when compared to notebooks and desktops [1]. A question thus arises: How can we diminish the above challenges for mobile users so that they can acquire their desired contents from several multimedia applications [2]? To date, recommended lists have been provided only for videos [3], to list the video's classification, description tags, watching history, and clips sharing in websites from end-users [4]. This paper concentrates on providing real-time notifications for mobile user applications about their desired multimedia contents. Moreover, these recommendation results are not implicit and stable for mobile users. Many websites are commercially driven because search engines based on keywords remember users' favorite videos on mobile applications. Several recommendation algorithms are successfully developed and exploited only for videos. For example, Google and AdWords services are based on context-based filtering (CB), and in recent years Amazon and Taboo have attained success by introducing collaborative filtering (CF) recommender systems to make browsing more user-friendly. The fundamental idea is to ensure that highly popular contents are distributed across a large group of common users, but that less popular contents will never be recommended to the users. One of the world-famous social networking websites, Facebook, has a recommendation system based on social network filtering (SNF) according to space links, content forwards, user concerns, and user interactions. Almost all of the existing recommendation algorithms consider two essential components: content recommender and contextual information, (e.g., time and location). Maximum contextual information is collected in a ubiquitous environment, according to the user interests. Due to inexorable computations, a real-time recommendation cannot be guaranteed in the existing algorithms. In this paper "a context-based intelligent multimedia system (CIMS) for ubiquitous cloud computing" is proposed, with cost-effective mechanisms (*i.e.*, recommendation, distribution, and offloading). The main idea of this paper is to reduce the computing percentage and storage complexities and extend the battery life for smartphones through a mobile cloud offloading system, which decides whether to offload actions to/from the cloud servers.

Context-aware decision-making (CAD) takes advantage of different historical benchmarks, based on context, to produce offloading decisions for lessening energy consumption, response time, and other criteria [1].

The rest of the paper is structured as follows. Section 2 discuss the related works, and Section 3 illustrates our proposed framework, "context-based intelligent multimedia system (CIMS) for ubiquitous cloud computing" and its design. Finally, we conclude and outline future work in Section 4.

2. Related Work

In this section, we discuss the core technologies for CIMS, existing research, and system considerations. The core technologies include a recommendation system, mobile cloud offloading systems, and mobile energy modeling.

2.1. Core Technologies

- Recommendation System

Recommendation systems concentrate on particular domains, e.g., Google News, Amazon and YouTube users. CB recommendation [5] considers similarities based on content titles, tags, or descriptions, while CF-based recommendation [6,7] focuses on user proceedings such as content browsing history and prominence. A CF system requires ample information about the history and feedback from users. Context-aware recommendation furnishes constant information without examining the user context (*i.e.*, location, time, and network). As a matter of fact, user interests differ depending on context (*i.e.*, location, time, and emotion), so context-aware recommendation conveys the sensed information to smartphones and takes a long time to aid the user and select his/her contents of interest [3]. Graph-based recommendation [8] is mostly used to reduce the memory usage in cloud servers by mapping similarities between a group of users' data (context and multimedia contents); these systems are important algorithms for a recommender system.

- Mobile Cloud Offloading Systems

In the early years, mobile cloud offloading became popular [1,9–11]. MAUI [9] is designed by Microsoft for Windows phones; it recognizes that the best way to offload is by stating the current context (*i.e.*, energy consumption, CPU utilization, and network conditions) of a mobile at runtime. ThinkAir [10] is like MAUI, but the script of offloading methods is added by the builder. Clone cloud [11] is specially designed for the Android offloading system. As a matter of fact, mobile applications offload information continuously if a network is available, and this does not check the energy consumption. Kwon and Tilevich [12] found that offloading minimizes higher energy consumption by using checkpoints.

- Energy Model

For a decade numerous energy consumption models have been presented for mobile devices [13–17]. Three important energy consumption components are categorized: (i) computation; (ii) communications; and (iii) display. In this paper, we contemplate the multimedia contents' energy consumption, because it is directly affected by contexts and its remaining behaviors are static. Many energy models consider

the different states of network interface and utilize the energy consumption levels [13,14]. This serves a useful purpose as an adjustable power model, because it considers the dependence between mobile hardware and power models [15]. These energy models never take different network conditions into consideration. Rahmati and Zhong consider the benefits of retransmission when any transferring data has failed over the same network [16]. Mittal *et al.* examine signal strength, which is a key function for communication energy [17].

2.2. System Considerations

This subsection explains the system considerations of ubiquitous computing.

- Computing Management:

To address the resource constraints of smartphones, mobile cloud computing has emerged to meet the demands for intensive computing tasks. In the coming years immense usage of cloud servers will lead to computation and database concerns; to overcome these issues, our research uses K-means clustering in our system.

- Real-Time Recommendation:

Generally mobile users expect to find exact information in real time. User context and content data interests vary day by day. A cloud server executes a distribution process to control users' data similarities without collisions and provide accurate notifications to mobile user applications in real time.

- Energy Saving:

Most smartphones continuously offload data when they are connected to a network; this process leads to high energy consumption. We consider a mobile cloud offloading system for mobile devices, with the aims of a short response time and lower energy consumption with the aid of a customized CAD. Using real-time notification is another way to lessen the energy consumption and computing costs for mobile users.

3. CIMS: Context-Based Intelligent Multimedia System

In this section, we present an architecture of CIMS with mechanisms such as recommendation, distribution, offloading, and design of the systems. We discuss a service scenario and efficiency analyses for CIMS in detail.

3.1. Architecture of CIMS

Figure 1 represents our proposed CIMS Architecture in Mobile Cloud Computing. As aforementioned, smart devices with limited resources acquire needed information from a cloud via the Internet. To reduce computing and storage concerns for mobile users, we perform computation in cloud servers and send notifications to user applications in real time. We consider user specifications in order to collect user context and multimedia contents of interest, and to identify similarities between users. After installing the application, users should perform authorized authentication (*i.e.*, email, password)

and fill in his/her content fields of interest; when the user starts browsing, all this data is stored in a cloud database. By collecting a user's information (as stated in his/her profile) in a cloud server, it is easy to match up similarities between a user's context and content data. The following sections explain the recommendation system, distribution process, and offloading mechanisms to lessen the computation cost, energy consumption, and memory size for mobile users and cloud servers.

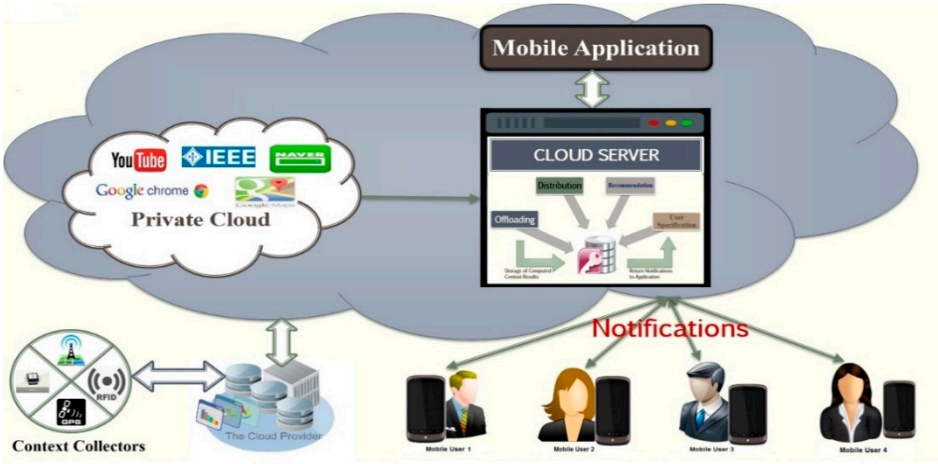

Figure 1. CIMS in Mobile Cloud Computing Architecture.

3.2. Design of Recommendation System

The proposed system, CIMS, is composed of a recommendation system. The dominant role behind the recommendation system is to reduce computing tasks in the mobile framework by using pervasive cloud computing. Usually mobile users depend on implicit and complete behavior models; for this we obtain a recommendation system, which includes four functions as illustrated in Figure 2: (1) User's Context and Content Data Collector, (2) Context Data Clustering, (3) Content Data Clustering, and (4) Clique Profiles Collector. All these functions are processed in the cloud server. A cloud server has several user profiles, which manage the user's privacy and data (*i.e.*, context and multimedia contents). The user's Context and Content Data Collector collects data from the database and divides the user's data (context and content) for clustering evaluation process to finally form several groups of user profiles according to their similarities (between context and content) (Figure 2). The cloud server performs recommendation processing by collecting data stated in user profiles and executing a clustering function according to the users' relationships and similarities in context and content data. These processes minimize the storage constraints for the cloud server, offer updates to a user profile if such data is available, and impart notifications without a mobile user's interactions, according to context entities (*i.e.*, location, time, network type, and user's interests). Finally, the recommendation system forms several groups of user profiles based on their similarities in context and multimedia contents and transmits the results to a distribution process to give real-time notifications for mobile user applications. The four major recommendation system functions are described as follows:

- User's Context and Content Data Collector

The user's context collectors (*i.e.*, sensors, General packet radio service (GPRS), Radio-frequency identification (RFID), *etc.*) gathers the user information and transmits the context data (*i.e.*, network types, time, and location) to the cloud server. The user's contents of interest (*i.e.*, audio, video files, documents) are collected from user profiles (*i.e.*, Gmail, Yahoo, Hotmail, *etc.*) at the initial stage of application and continue to be updated according to the user's browsing history. The recommendation system collects context and content data from a highly secure cloud database according to user profiles. We think it is worthwhile to develop a notification technique for mobile user applications because users spent a lot of time surfing for their interests on the Internet, and this leads to high computation and energy concerns. Our goal is to reduce computation and energy consumption for smart devices; we achieve these constraints by sending updated notifications to a mobile user's application in real time according to his/her profile interests. Due to the daily increase of similarities in a user's interest and context types, computing will lessen quickly. To minimize the diverse challenges for recommendation systems, we perform a clustering procedure to reduce the computing load at cloud servers. User profiles and their data (*i.e.*, context and multimedia contents) collection are updated simultaneously to keep the recommendation process fresh at the server side.

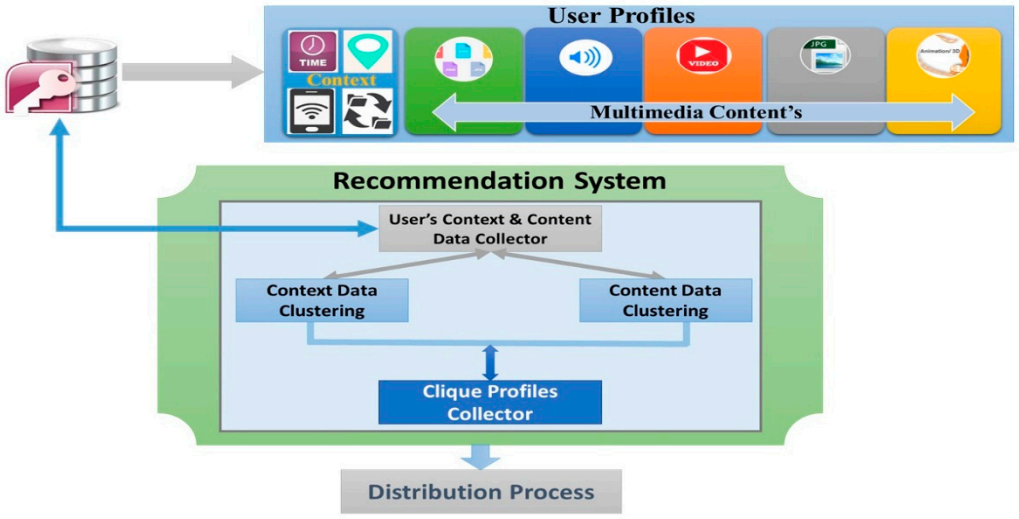

Figure 2. Design of Recommendation System.

- Context Data Clustering

As aforementioned, ubiquitous computing relies on precise context data to provide an accurate service to users. Studying users' context (*i.e.*, network types, time and location) is significant. For example: consider YouTube, a famous entertainment application with billions of users enjoying their favorite music and videos at any one time. In this context several users have similarities in accessing the same video clip or music album on either a particular network (3G/4G, Wi-Fi), or at a particular time or location. We use these context similarities to cluster and form several groups of users based on their similarities. Through research studies considering examples like the above, we perform a clustering method on context data to notify the user automatically only if there is any update relevant to the user's interests, according to their context entities. As said before, to obtain an exact recommendation system,

we use cluster algorithms for user satisfaction. In our paper we perform the K-means clustering operation shown in Figure 3:

(1) User profiles are divided into s chunks; each truck includes the profile of different users. We compare user profile context data (time, location, and network type) to find parallel context between profiles.
(2) K-means clustering algorithm collects the user's content information (browsed content (BC), keywords, and content-type) and again matches the user profiles to obtain content similarities.
(3) Next it will match user profiles by both context and content; finally, the results are stored in clique profiles with their similarities.
(4) If any user does not find similarities, his/her profile will be stored in clique profiles for future use.

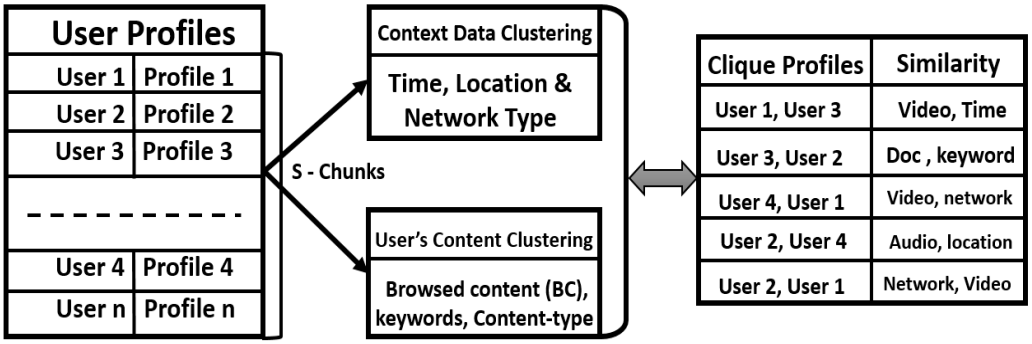

Figure 3. User's Context & Content Clustering.

- Content Data Clustering

User's profiles are exploited by the recommendation system to find user content similarities. Multimedia content applications retrieve the user's favorite contents by browsing history, audio, video files, documents, keywords, *etc.* In addition, several user groups are formed based on similarities with content illustrations (browsed content, keywords, content type). For instance, in Figure 3, the second row shows similarities between user 3 and 2 with keywords. This means users 2 and 3 are searching the same document in a multimedia application. Our system considers the similarities in a set of user profiles and sends notifications automatically if any updates are available for their related research. This feature lessens the storage concerns for cloud servers (by not saving an individual profile with his/her contents of interest). Relying on content illustration, the recommendation system requires little computation and provides exact information. Naturally, a user's interests change according to context and old profiles produce a lot of noise throughout the recommendation process; to control these concerns we merge the context clustering (Figure 3) to get exact and complete information according to the user profile. We do this by placing the group of users in one truck and performing K-means clustering on user's data (context and content). Based on clustering, the data cloud server collects a group of users' information with shared interests or other features in common; this method helps the application providers to make implicit recommendations for users.

- Clique Profile Collector

Users usually browse multimedia content applications (*i.e.*, YouTube, IEEE, Naver, *etc.*) for their favorite content; many users have highly similar interests in the resources. By considering similar resources (*i.e.*, browsing history, audio video files, documents, *etc.*), research studies find users with highly similar behavior. The recommendation system uses a Clique Profiles Collector to achieve better performance and reduce the storage and computing complexities for the cloud server, based on the similarities between several users. Moreover, users' clustering data (context and content) is replaced by the Clique Profiles Collector to remove the collisions between several groups of user profiles and also give exact results for individuals as well as for groups of users. For example, a person is searching for a job opportunities with his/her user ID and his friend (or another authorized user with similar interests) is browsing the same website. First the recommendation system considers individual histories (context and content data) and performs matching with several user profiles; if any match exists it will store the similar data (content and context) in a group with several users, and send notifications only if there is any update regarding the user's interests. In cases where the user does not match with another user, his/her profile will be stored in clique profiles for future use. This approach reduces the computing cost for the cloud server. After performing the recommendation process, results are directed towards the distribution process, which performs real-time recommendation process.

3.3. Service Scenario of CIMS

This section explains the general process and core mechanisms of our proposed system (CIMS) with defined functions shown in Table 1.

Table 1. Explanation of terms.

Terms	Explanation
MU	Mobile User
APP	Application
CS	Cloud Server
OFF	Offloading
DB	Data Base
RS	Recommendation System
DP	Distribution Process
RR	Real-Time Recommendation

- CIMS General Process

Figure 4 illustrates the general process of CIMS.

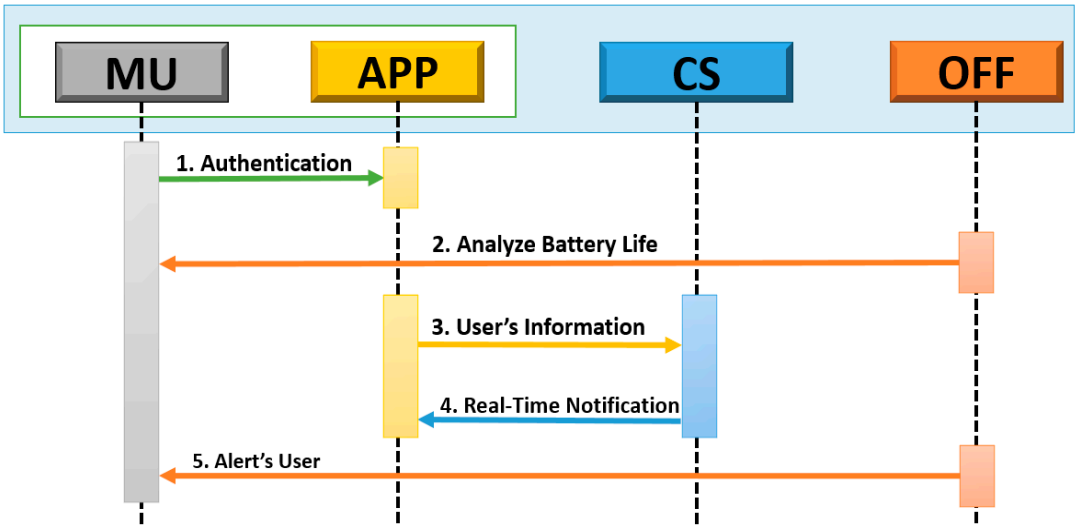

Figure 4. General process of CIMS.

Step 1 MU→APP: Authentication
After installation of the application, the mobile user performs authentication with his/her personal username and password.

Step 2 OFF→MU: Analyze the Battery Life
An offloading process analyzes the battery life of the user's mobile by using an energy model and allows the application to store user information to the cloud only if battery life is efficient.

Step 3 APP→CS: User Information
The user application collects information (*i.e.*, context and multimedia contents) and stores it on the cloud server database.

Step 4 CS→APP: Real-Time Notification
With all users' data, the cloud server performs its core operations to post updated notifications to a mobile user's application in real time.

Step 5 OFF→MU: Alerts the User
The offloading system alerts the user while downloading large contents, to lessen the energy consumption based on the user context.

- Core Mechanisms of CUM System

Figure 5 demonstrates the core mechanisms of our system (CIMS). After obtaining the user's information, the cloud server uses the recommendation system to generalize about users' similarities based on context and content information. The distribution system obtains the similarities between attributes to provide real-time notifications to mobile users.

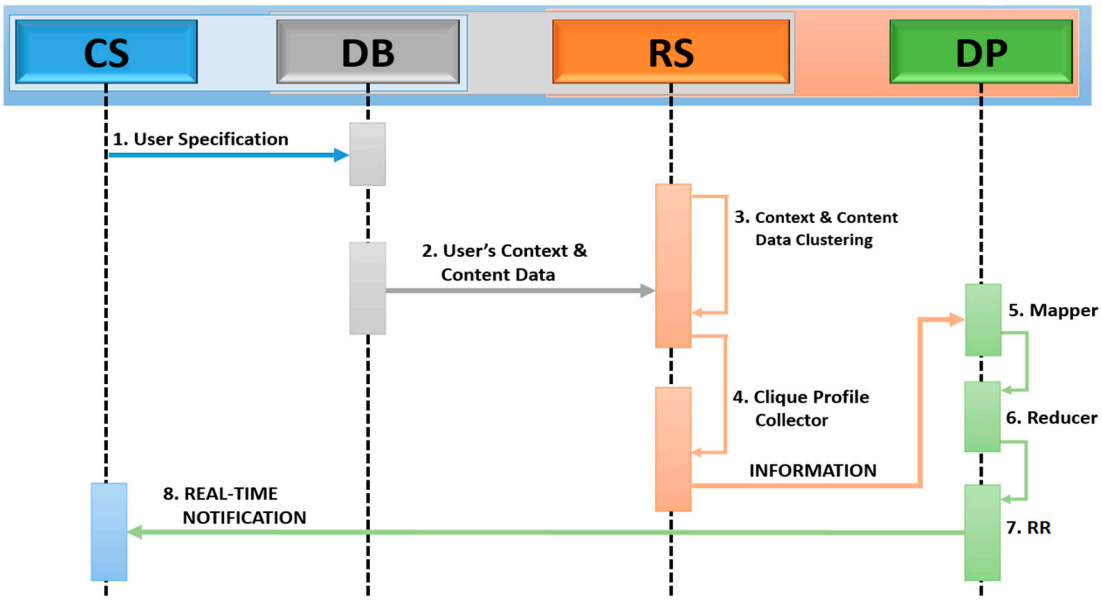

Figure 5. Core Mechanisms of CIMS.

Step 1 CS→DB: User Specification

The application requests the user to enter fields of interests, because without basic information it is very difficult for the server to analyze the user's interests at this initial stage. Along with the user's interests, his/her browsing data (*i.e.*, context, multimedia contents) are stored to the cloud database.

Step 2 DB→RS: User's Context and Content Data

A cloud database containing many users' profiles with their context data and multimedia contents of interest is processed by the recommendation system.

Step 3 RS: Context and Content Data Clustering

K-means clustering first finds similarities in context and then proceeds for content information between many users.

Step 4 RS: Clique Profile Collector

A clustering process is performed between the context and multimedia contents of users' data to form several groups of users based on their similarities.

Step 5 RS→DP: Mapper

Obtain Clique Profile information from RS for cluster mapping. Then the mapper computes user similarities based on attributes, arranges the user profiles according to high similarities, and organizes them into one cluster.

Step 6 DP: Reducer

The reducer performs similarly to the mapper, but to deepen the search it considers K clusters by mixing the KS profiles. If any similarities are found it will merge them; otherwise it updates the cluster.

Step 7 DP: Real-Time Recommendation (RR)

This can be obtained by analyzing user behavior in cloud-based clustering.

Step 8 DP→CS: Real-Time Notification

Recommendation rules (Figure 8) help the cloud server to convey notifications to mobile users' applications in real time by matching the user context in order.

3.3.1. Distribution Process for CIMS Service

After obtaining recommendation results, the distribution process performs real-time recommendation (*i.e.*, time, location, and network) and automatically returns favorite notifications to the applications according to user profiles, as shown in Figure 6. By having these recommendation rules, we ensure that user needs are met in real time. To reduce the burden for the recommendation system we include a profile partition using cluster mapping and a reducer to merge user profiles with similar interests in both context and multimedia contents. These functions reduce the computation and give real-time notifications to the mobile user. Clustering mappers are prompted as in Figure 7, to process the partition profiles.

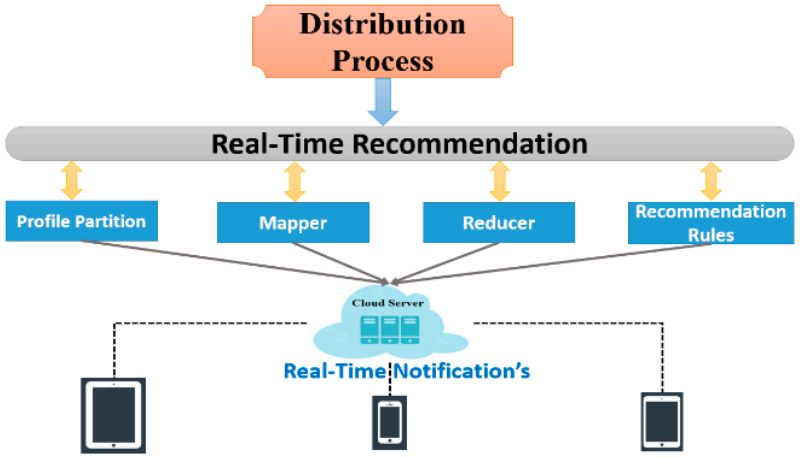

Figure 6. Distribution Process Design.

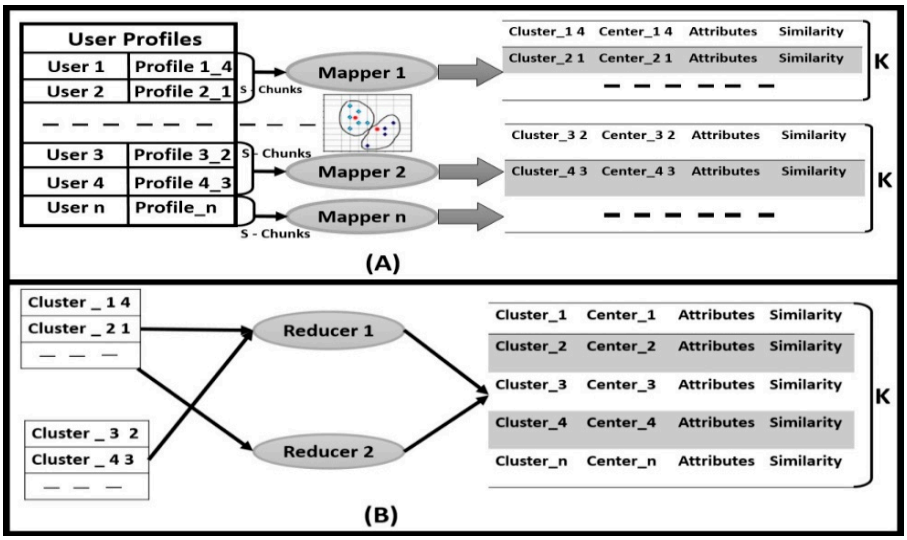

Figure 7. K-Means Clustering Mapper & Reducer.

- Mapper

To compute similarities in attributes between user profiles at chunk-s, we employ a mapper in the distribution system to find the central point of *K* clusters, as shown in Figure 7A. For example, if many users in YouTube search for their favorite music, based on attributes (*i.e.*, key words, browsing history, location, time and network type) application stores the information on the cloud server in the user's profile. The mapper computes the similarities between users based on attributes and arranges the user profiles linked by close similarities into one cluster. To minimize the memory size requirements, we re-choose the cluster central point *K* and compare it with old clusters; if no matches exist, the mapper ends the task. Otherwise it updates the central point *K* cluster to give surety and stability to the cluster.

- Reducer

The reducer is the final phase for lessening memory size in cloud servers. A mapper with a central profile acquires the K-clustering means of similarities by using profile attributes (*i.e.*, key words, browsing history, location, time, and network type). In the distribution system, after processing with the mapper, the reducer intermixes the KS profiles into single *K* clusters, as shown in Figure 7B. The reducer performs similarly to the mapper; taking a step in advance, it builds *K* clusters by mixing the KS profiles, because user profile attributes with similarities are merged directly. Otherwise the reducer does the same job as the mapper, stabilizing the K-means clustering through iteration.

- Real-Time Recommendation

Real-time recommendation can be obtained by analyzing user behavior in cloud-based clustering. Example recommendation rules are shown in Figure 8. The cloud-server notifies the user when new updates are available. For example, users often access the Google map application to reach their office, but the next day bus timings might have changed. Our real-time recommendation helps the user to get the notification in real time and thus choose another way to reach the office. The rules help the application server to convey the information at the right time by achieving the rules one by one in order by matching the user contexts. As we discussed before, the user context and interests change from day to day, so we should keep on updating recommendation rules to improve the real-time process.

Recommendation Rules:

Context Rule:

Rule-cxt: context(?s)^hasaccessnetwork(?s, ?net-type)^Swrld:equal(?net, "wifi")^hasdevicetype(?s,?dt)^swrld(?dt, "smartphone")^hasaccesstime(?s, ?t)^swrld:greaterthan(?t, "15:00")^swrld:greaterthan(?t, "24:00") ^haslocation (?s, ?loc) ^swrld:greaterthan(?loc, "1m")->viewcluster(?s,cxt)

Content Rule:

Rule-ctnt: content(?s)^hastype (?s, ?type)^Swrld:frequency(?s, "audio")^haspixels (?s, ?pi) ^Swrld:scale(?size, "width,height") hasresolution (?s, ?res) ^Swrld:greaterorequal(^res, "360p")^hasalphabets(?s, ?alpha) ^Swrld:type(?s, "doc,pdf")->viewcluster(?s,ctnt)

Recommending Rule:

Rule-recc: recommending(?s)^hascontext(?s.?context)^swrld:similar(?context,m)^haskeywords(?s,?key) ^haskey_1(?key,?key_1)^swrld:similar(?Key_1, "cloud") ^haskey_2(?key,?key_2)^swrld:similar(?Key_1, "IOS")^haskey_3(?key,?key_3) ^swrld:similar(?Key_3, "push")^hassimilar(?s,?sim)^swrlb:similar(?sim,1) ->recommendation(?s,recc)

Figure 8. Recommendation Rules.

3.3.2. Offloading System

Mobile cloud computing appears to lessen the resource constrains of smart devices, but it suffers from continuously offloading actions as long as network is available. The process of offloading between the smart phone and the cloud server is shown in Figure 9, which consists of three components: energy model, user profiler, and context-aware decision (CAD). We detail each component in the following subsections.

- User's Profiles

Several aspects underlying user profiles are required to process the offloading decisions. We assess four important contexts: (1) Network Signal—periodically collects the latest network signal (*i.e.*, Wi-Fi, 3G/4G); (2) Transmission rate—by transmitting dummy data to/from cloud server, we measure the transmission time and energy estimation; (3) Time—by partitioning the 24 hours into time slots and recording the opportunities for offloading; (4) Location—obtaining a user's geographical location by GPS every half an hour.

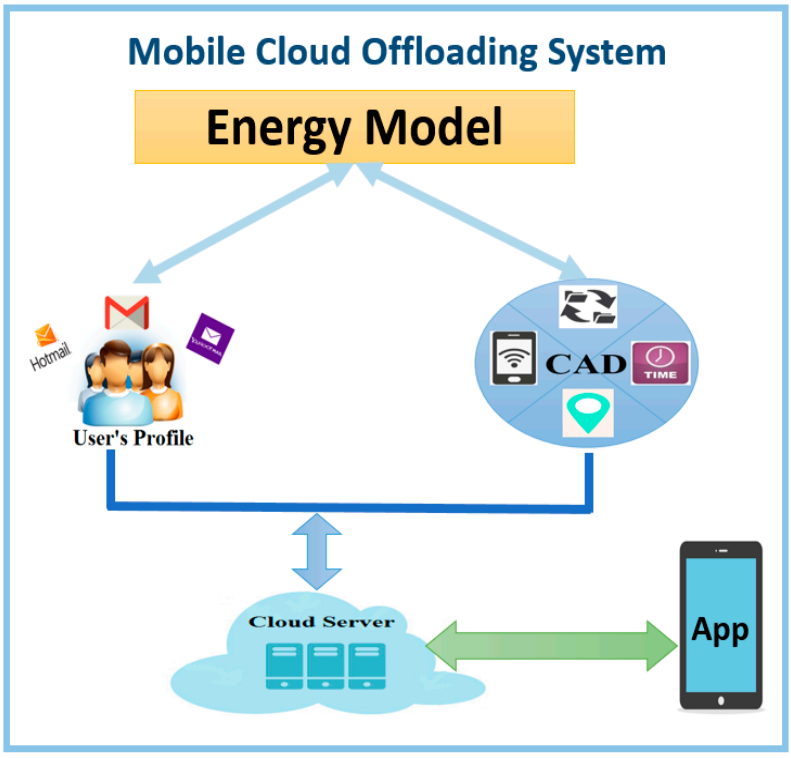

Figure 9. Mobile Cloud Offloading System Design.

- Context-Aware Decision (CAD)

Considering the existing research on offloading systems, we strongly believe that power saving does not always happen while offloading to the cloud. For example, if a mobile performs offload computation, especially on a 3G/4G network, the transmission energy may be greater than consumption energy. However, a context-aware decision engine is deployed to lessen the energy consumption. We assess four important contexts—network signal, transmission rate, time, and location—in a CAD engine to measure

a user's regular behaviors, *i.e.* the places a user regularly visits at the same time, particularly on weekdays, thus accessing the same network conditions and applications. For example, user might go to the office Monday–Friday, 9 a.m. to 5 p.m., and go to the park on weekends from 5 p.m. to 8 p.m. Generally the wireless network (Wi-Fi) in an office will be sufficient for offloading computation, and power consumption does not matter. However, for a user in the park, a wireless network (3G/4G) is not as good as an office network. Based on users' past context data, CAD executes the offloading process. Before calling up users' data, CAD checks the present user's context. If it is good enough, it will carry on; otherwise, it utilizes past execution costs.

- Energy Model

Once the process of recommendation system, distribution process, and offloading mechanisms is executed, the costs including computing time, energy consumption, and time are recorded in a secure database in the cloud server. Our energy model estimates energy consumption using the above costs based on network signal, transmission rate, location, and time. We store all these context data in the user profile on the cloud sever. A CAD engine with context rules (Figure 8) uses all this contextual information to perform offloading computation on a mobile device and returns the standard performance to the cloud. Another important point to bear in mind so as to avoid null results is that CAD requests information from the user at the initial stage.

3.4. Efficiency Analyses of CIMS

This section presents the efficiency of our proposed system, CIMS, and presents a comparative analysis with existing research models. We compare and analyze the recommendation and offloading systems, presented in Section 2.1 [1,3,6] with our proposed system. We obtain the following considerations for comparison analysis: computing process, data-base management, real-time processing, and power consumption. Table 2 shows the analysis results.

Table 2. Analyses of performance.

Performance	CIMS	[1]	[3]	[6]
Computing Process	⊙	X	Δ	Δ
Data-Base Management	⊙	X	Δ	X
Real-time Processing	⊙	⊙	⊙	X
Power Consumption	⊙	Δ	X	X

(⊙: Good, Δ: Moderate, X: Weak.)

- Computing Process

A resource-constrained mobile device refers to a cloud server over the Internet that is used to simplify the computing process. An offloading engine [1] deploys a mobile and uses the CAD algorithm for offloading to cloud servers. The existing literature only considers video applications [3,6]. The recommendation system [3] considers video sharing websites that collect user context, relationship, and profiles to generate multimedia recommendation rules. The recommendation system created by Go *et al.* studies three user behaviors: favorites, uploads, and viewing of videos in YouTube [6]. The proposed

system conveys real time notifications to mobile user by obtaining the users' profile, contexts, multimedia contents of interest, and attributes. The offloading engine in the cloud server performs offloading actions to/from the mobile user, based on the user's context. The above factors are included in our system without user interaction via cloud computing, which reduces the high computation task for a smartphone.

- Database Management

Database management refers to how to reduce the database concerns in cloud servers in the upcoming years. Mo *et al*. [3] utilized user profiles to find content similarities between users; several communities were formed based on access patterns and content descriptions were mapped and clustering algorithms performed to obtain users' similarities in content. Existing literature does not consider database management in the cloud server [1,6]. Our system performs K-means clustering on user profiles and procures similarities between user contexts and multimedia contents, and then we use a clique profile collector to acquire groups of user profiles with features in common. We use a reducer to deepen the search based on arranging highly similar user profiles into one cluster to reduce storage concerns for cloud servers.

- Real-Time Processing

Real-time processing is a must and should guarantee the quality of the user experience. Our proposed system and mobile cloud offloading engine [1] perform similar actions, by notifying the user according to their contexts. A new user request is accepted by the real-time component and returns recommendation lists to the user [3]. The component translates the request into recommendation rules on the basis of keywords and user context and then searches for user favorites according to those rules. By contrast, the proposed system makes it easier for the user, by collecting the user's information and sending absolute notifications related to the mobile user's profile to his/her applications in real time, from the cloud server. We employ the reducer in a distribution process to avoid collisions between several user profiles and impart absolute information in real time, based on the recommendation rules in Figure 8.

- Power Consumption

Most smartphones with many multimedia applications continuously perform offloading actions when connected to a network, which leads to high power consumption. A mobile cloud offloading system [1], on the other hand, decides whether to offload from cloud servers and evaluates the CAD algorithm for low power consumption. The existing literature does not review power consumption in smartphones [3,6]. However, in our system downloading as well as offloading are performed via the cloud server, relying on CAD with a user's context to reduce the power consumption for mobile users. Therefore, the proposed system has better availability in ubiquitous cloud computing than the previously proposed schemes.

4. Conclusions

In this paper we proposed CIMS as a method of automatically sending notifications to mobile users according to their profiles in real time. We analyze three kinds of user behaviors—user contexts, contents of interest, and clique profiles in the recommendation system, to reduce the storage and computation complexities for mobile users. Our energy model, with context-aware decision (CAD), uses the transmission rate, network signal location, and time to make better mobile cloud offloading decisions. Existing research [3,6] does not consider real-time notifications and energy saving for smartphone users. Our system evaluates user profiles to obtain the similarities between users' contexts, contents of interest, and attributes, using K-means clustering to lessen the storage burdens in the cloud server. The distribution process controls users' data similarities without collisions and provides accurate notification to the mobile user in real time. Our mobile cloud offloading system is flexible and supports different optimization criteria, by minimizing the response time and energy consumption. Comparative analysis (Table 2) shows that our proposed system provides higher quality multimedia information to mobile users than present systems.

In future work, we will modify the pattern of clustering and distribution processes to reduce the storage constraints for the cloud server and develop efficiency in providing real-time notifications. Moreover, we are looking into additional user contexts that may further improve the decision accuracy of CAD. Another important goal is designing a cache for multimedia systems to reduce computation pressures in the cloud server.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning) of Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1014) supervised by the IITP (Institute for Information & Communications Technology Promotion).

Author Contributions

Abhilash Sreeramaneni: Main writing and also designing the total system; Won Min Kang and Hyungjin Im: Design of the total system; Chan Koh: Research on related works, analyzing and improving the proposed system; Jong Hyuk Park: Total supervision of the paperwork, review, and comments, *etc.* All authors have read and approved the final manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Lin, T.-Y.; Lin, T.-A.; Hsu, C.-H.; King, C.-T. Context-Aware Decision Engine for Mobile Cloud Offloading. In Proceedings of the 2013 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Shanghai, China, 7–10 April 2013; pp. 111–116.

2. Chang, K.-D.; Chen, C.-Y.; Chen, J.-L.; Chao, H.-C. Challenges to next generation services in IP multimedia subsystem. *J. Inf. Process. Syst.* **2010**, *6*, 129–146.
3. Mo, Y.; Chen, J.; Xie, X.; Luo, C.; Yang, L.T. Cloud-based Mobile multimedia recommendation system with user behavior information. *IEEE Syst. J.* **2014**, *8*, 184–193.
4. Lai, C.-F.; Huang, Y.-M.; Chao, H.-C. DLNA-based multimedia sharing system for OSGI framework with extension to P2P network. *IEEE Syst. J.* **2010**, *4*, 262–270.
5. Li, L.; Wang, D.; Li, T.; Knox, D.; Padmanabhan, B. SCENE: A scalable two-stage personalized news recommendation system. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'11, Beijing, China, 29–29 July 2011; pp. 125–134.
6. Go, G.; Yang, J.; Park, H.; Han, S. Using online media sharing behavior as implicit feedback for collaborative filtering. In Proceedings of the 2010 IEEE Second International Conference on Social Computing (SocialCom), Minneapolis, MN, USA, 20–22 August 2010; pp. 439–445.
7. Chan, N.N.; Gaaloul, W.; Tata, S. Collaborative filtering technique for web service recommendation based on user-operation combination. In *On the Move to Meaningful Internet Systems: OTM 2010*; Meersman, R., Dillon, T., Herrero, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Part 1, Volume 6426, pp. 222–239.
8. Baluja, S.; Seth, R.; Sivakumar, D.; Jing, Y.; Yagnik, J.; Kumar, S.; Ravichandran, D.; Aly, M. Video suggestion and discovery for YouTube: Taking random walks through the view graph. In Proceedings of the 17th International Conference on World Wide Web, Beijing, China, 21–25 April 2008; pp. 895–904.
9. Cuervo, E.; Balasubramanian, A.; Cho, D.; Wolman, A.; Saroiu, S.; Chandra, R.; Bahl, P. Maui: Making smartphones last longer with code offload. In Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys'10), San Francisco, CA, USA, 15–18 June 2010; pp. 49–62.
10. Kosta, S.; Aucinas, A.; Hui, P.; Mortier, R.; Zhang, X. ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In Proceedings of the IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 945–953.
11. Chun, B.; Naik, M.; Ihm, S.; Patti, A.; Maniatis, P. Clonecloud: Elastic execution between mobile device and cloud. In Proceedings of the 6th European Conference on Computer Systems (EuroSys'11), Salzburg, Austria, 10–13 April 2011; pp. 301–314.
12. Kwon, Y.; Tilevich, E. Energy-efficient and fault-tolerant distributed mobile execution. In Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS'12), Macau, China, 18–21 June 2012; pp. 586–595.
13. Perrucci, G.; Fitzek, F.; Widmer, J. Survey on energy consumption entities on the smartphone platform. In Proceedings of the 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring'11), Yokohama, Japan, 15–18 May 2011; pp. 1–6.
14. Zhang, L.; Tiwana, B.; Qian, Z.; Wang, Z.; Dick, R.; Mao, Z.; Yang, L. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS'10), Scottsdale, AZ, USA, 24–29 October 2010; pp. 105–114.

15. Dong, M.; Zhong, L. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys'11), Washington, DC, USA, 28 June–1 July 2011; pp. 335–348.
16. Rahmati, A.; Zhong, L. Context-based network estimation for energy-efficient ubiquitous wireless connectivity. *IEEE Trans. Mobile Comput.* **2011**, *10*, 54–66.
17. Mittal, R.; Kansal, A.; Chandra, R. Empowering developers to estimate app energy consumption. In Proceedings of the Annual International Conference on Mobile Computing and Networking (Mobicom'12), Istanbul, Turkey, 22–26 August 2012; pp. 317–328.