# Enabling Ubiquitous Application Access for IoT

Injung Kim
Computer Science
Columbia University
NY, U.S.A.
injungkim@caa.columbia.edu

Sambit Sahu
Cloud and Big Data Analytics
IBM T.J. Watson Research Center
NY, U.S.A.
sambits@us.ibm.com

***Abstract*— Ubiquitous and efficient information access is key in IoT environment. For such enablement, Cloud is leveraged for ubiquitous content access (anywhere and anytime) where some example services are Dropbox, iCloud, SkyDrive, and Google Drive. An eco-system of applications and services has been made possible by Cloud based platforms that were not feasible that easily. These use cases have successfully allowed users to access their content ubiquitously and seamlessly from a wide variety of desktop as well as mobile devices of their choice.**

**Extending this abstraction to personalized applications stack, we propose and build a personalized application access leveraging Internet of Thing (IoT) framework to support access to personalized applications from anywhere on any device. First, we develop a framework for application virtualization that decouples an application from any specific platform or device and enables access to personalized application access at per user level. Next we provide an implementation of this framework for applications on Windows platform leveraging Amazon S3 cloud storage although our approach can be implemented for any platform with any other cloud systems.**

***Keywords*— cloud, ubiquitous application access, application virtualization**

## I. INTRODUCTION

The Internet of Things (IoT) has revolutionized the cyber-physical space with the information gathering, processing and digital information access. Combined with Cloud computing resources at the backend, together these have enabled a wide range of cyber-physical services and applications that were not feasible before. These include a wide range of services such as healthcare, smart energy, transportation, tourism, daily public services, entertainment industries, or emergency services.

Cloud computing with its resource level virtualization supports on-demand and elastic access to IT computes resources. Such ease of elastic and on-demand resource access has spurred a wide variety of enterprise grade applications that were otherwise very difficult to provide without Cloud based resource. One such application category is the ubiquitous access to data anytime anywhere from any platform or device. Example services include Dropbox [1], OneDrive [2], iCloud [3], and Google Drive [4]. Storage cloud is leveraged by these services to allow users access to their data in a truly ubiquitous manner.

The focus of our paper is to extend this ubiquitous data access to ubiquitous application access in an IoT environment so that seamlessly access to customized applications from anywhere on any type of platforms and devices can be supported. The benefit of ubiquitous application access in an IoT environment in addition to data access is *to empower services that need such personalized application accesses at these end points at any time.*

We propose Ubiquitous Application Access (UAA) - that leverages cloud to allow access to applications ubiquitously as mentioned above. The key component of UAA is an application virtualization framework that allows UAA to enable platform independent access to personalized applications.

The application virtualization framework consists of two components, i.e., (i) ability to support platform independent access to applications, and (ii) ability to support user specified application personalization in an application independent manner. While the first requirement is relatively easier and feasible to support with simple modifications of existing capabilities such as AppStore [5] and Google Play [6], the second capability requires one to automatically translate the user initiated personalization of an application in one platform to other platforms - which is the core of our proposed UAA application virtualization framework. The implementation of our UAA framework consists of two key components: one is to package all the application and to deploy the applications seamlessly on any computer, and the other is to compose the user profile.

While AppStore [5] with iCloud and Google Play[6] with Google Cloud could be leveraged to allow ubiquitous application access, these are neither platform agnostic nor support per user application customization. Desktop cloud is significantly different in that user is provided access to a virtualized platform with remote access to a set of application through remote desktop. We provide access to applications on the device natively that has significant advantage both in terms of performance and functionality.

In order to illustrate the feasibility of our approach, we provide a prototype implementation for Windows based platform with the right level of abstraction leveraging Amazon S3 storage service [7]. Windows registry is a special mechanism to maintain application and desktop configurations, and our prototype implementation also controls the registries to support seamless application and user profile. We demonstrate our prototype by comparing device based and cloud based compositions of UAA framework. In addition, we discuss areas of future research to extend the UAA framework to the diverse platforms and devices.

In Section 2 we motivate UAA capability and illustrate usage scenarios. Section 3 describes the UAA framework with application virtualization and functional architecture. Section 4 describes our end-to-end prototype for Windows based platform. Section 5 summarizes our current capability and discusses the future direction to truly achieve application level ubiquity leveraging cloud.

## II. Motivation and Usage Scenario

In this section, we describe UAA - a ubiquitous personalized application access in IoT environment that allows access to applications from anywhere on any platform. First, we motivate our proposed solution and compare it with related work. Next we illustrate UAA usage and its enabling key elements.

### *A. Motivation*

In a truly emerging ubiquitous computing era, one should be able to access ones personalized application besides the data from anywhere on any platform. Cloud storage based services such as Amazon S3 [7], Google Drive [4], Dropbox [1], or Microsoft OneDrive [2] allows ubiquitous data access, and it has been extensively studied in the several ways; access control for multi-authority cloud storage[8], reducing the frequency of data loss[9], consistency checking[10], minimizing trust[11], and so on. However, these do not consider how access to ones personalized applications.

One may argue that a user may access to his/her applications remotely, i.e., remote desktop - but this has clear disadvantages which are network dependency, incompatibility with some operating systems, downtime, or bottlenecks. Many usage scenarios and applications would not meet the desired performance or the constraints.

While services like iCloud [3] a step in this direction that could be leveraged to build the capability that being advocated by our UAA solution, it is not platform independent and also does not support application personalization. Our previous work [12] supports the portability of application and user profile level based on the USB hard drives, but it has limited storage capacity, as well as exposes itself to the security threat like a lost device, and could only deploy platform dependent applications.

Our proposed solution allows one to access personalized applications on any platform - a capability that extends the ubiquitous data access. UAA allows users access to their personalized applications thereby providing a seamless connected experience in a platform independent manner.

### *B. UAA Usage Phases*

Let us illustrate how UAA facilitates the capability stated above. Figure 1 illustrates the initialization phase, i.e., the very first time a user accesses an application using UAA from a device. This phase is similar to installing and accessing an application from AppStore [5] for iOS, Google Play [6] for Android platform. But UAA supports this in a platform

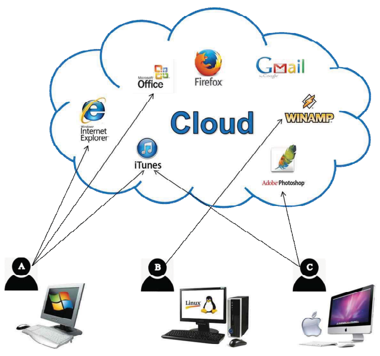

Fig. 1. Users might choose applications they want to use on any platform

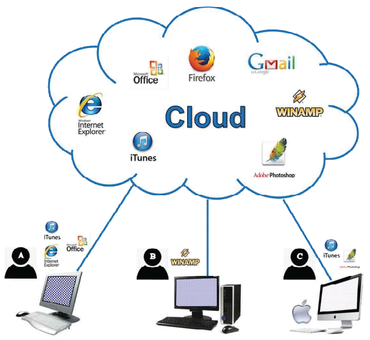

Fig. 2. The user could use the application with their uer profile.

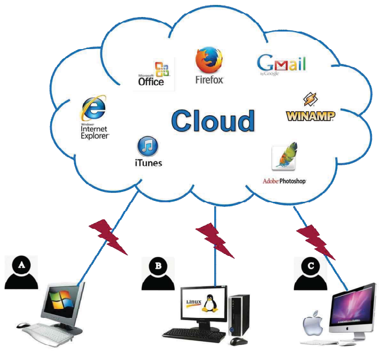

Fig. 3. As soon as the user is disconnected to the UAA framework, the public computer will restore to the original status.

agnostic manner by automatically selecting appropriate application version and installation process. The user is then allowed to use these applications with ubiquitous data access. The user may choose to personalize the applications thereby changing application configurations.

Figure 2 illustrates the capability to personalize and customize the applications. For example, one may choose to change ones screen wall paper. It is feasible as UAA framework maintains application customizations by keeping track of changes to appropriate configuration files. Thus next time a user accesses the same application, the saved configuration files are used instead of initial configuration files.

Figure 3 illustrates the cleanup phase that is activated as soon as user wants to end the session. This is a critical phase for privacy concern as user could be accessing his/her personalized applications along with data from a public computer.

As illustrated, users may easily access their data and any application with a simple connection to the UAA framework that is enabled on top of cloud storage. As UAA restores applications using saved per application configurations in a platform independent manner, users are provided a truly ubiquitous application level access.

## III. FRAMEWORK AND ARCHITECTURE

In order to support ubiquitous application access, UAA provides two key enabling components: (i) platform and OS independent application access, (ii) application personalization. These are the core capabilities that UAA brings through application virtualization framework that we propose and provide an implementation. We first describe this proposed framework followed by functional description of UAA.

### *A. Application Virtualization Framework*

Supporting the first requirement is relatively easier. UAA needs to detect the platform and OS level details of the device that the user logs in to access UAA. Based on the platform details, it chooses the right application installation files and installer.

The second requirement of application personalization is tricky. UAA needs to capture the changes the user makes to the application configuration on the current platform and then derive the equivalent changes for the other platforms.

These two requirements are achieved by describing an application that captures platform dependent installation and configuration relationships across the platform. These together are our framework for application virtualization. A suggested implementation is a configuration specification of application defined in a manner that allows one to link configuration elements through a relationship. Once an element is changed in one platform configuration, through this relationship, this change is propagated to other platform configurations. In some sense this is similar to what an OVF format is for hypervisor independent virtual machine specification.

### *B. Functional Architecture*

Next we describe the functional aspect of the architecture. It could be divided into (i) preparing stage, (ii) deployment stage and (iii) composition phase.

*1) Preparing Stage (Packaging All The Applications)*: The UAA framework lets the users make the application packages they want to carry, and to make the application package, they need to have its general set up file. Figure 4 shows the key steps to make an application package. For example in the case of Windows platform, all the file actions are requested by I/O request packet (IRP) on the kernel level. Since it works on the kernel level, we call our functional module as a file I/O filter

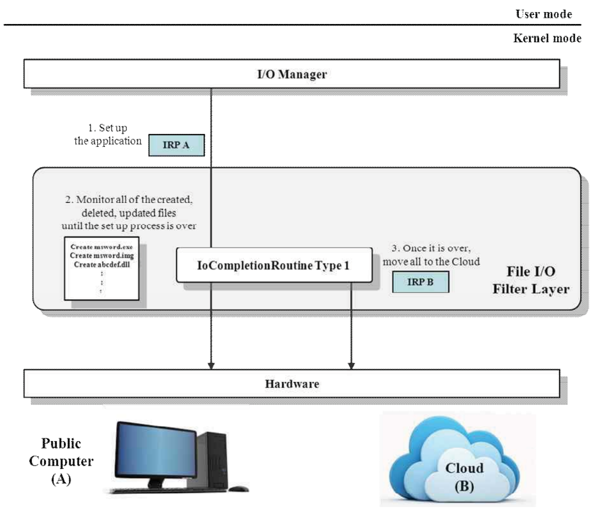
User mode
Kernel mode
I/O Manager
1. Set up the application
IRP A
2. Monitor all of the created, deleted, updated files until the set up process is over
Create msword.exe
Create msword.img
Create abcdef.dll
IoCompletionRoutine Type 1
3. Once it is over, move all to the Cloud
IRP B
File I/O Filter Layer
Hardware
Public Computer (A)
Cloud (B)

Fig. 4. Packaging all the applications

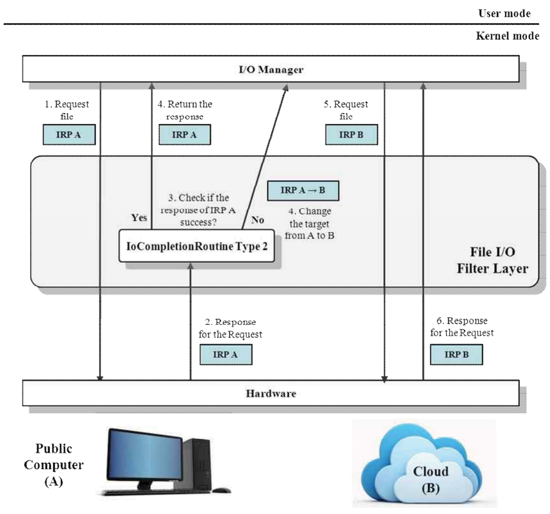

Fig. 5. Deploying applications seamlessly

layer, and the IoCompletionRoutine type1 function on the file I/O filter layer performs to make the application package.

For the first step, through the UAA user interface, they may choose any application's set up file, and then start it to let the IoCompletionRoutine function know which process is to be monitored. As soon as the set up process is started, the function also starts to monitor all the actions such as create, update, and delete files. And then, once the set up process is done, the function detects the end of the set up process and moves all the changed or created files to the cloud storage with file path, name, version, and platform information. In the Figure 4, we mark an original file request as IRP A which target is a public computer, and mark a copy request of application files as IRP B which target is cloud storage.

The point is these packaged applications cannot be executed on other computers because all the application processes tries to find their files on the initial computer not on the cloud storage. Therefore, following stage will describe how to deploy application packages seamlessly.

*2) Deploying Stage (Deploying Applications Seamlessly):* Between two different stages; preparing stage and deploying stage, there is no need to work on the same public computer but need to connect to the user's cloud storage space. Once the user makes the application packages into the cloud storage, he or she can move all around the world, and then have a sit in front of any platform to keep going on his/her work.

Since we suppose that all of the application packages are already stored on the cloud storage by the preparing stage, there could be all the versions of diverse applications for all kinds of platform on the cloud storage. Due to the benefit of the cloud storage, there is no storage capacity limitation and the user may easily put any application package and their large data, either. However, the application package could not be executed itself because all the related application files do not exist on the public computer but on the cloud storage. Figure 5 explains how to redirect all the file requests from the public computer to the cloud storage. In this case, IoCompletionRoutine function on the file I/O filter layer performs to let the application package work seamlessly.

The concept is that the requests of the application will fail since the file does not exist on the public computer. Therefore, before the failed response is returned to the application itself in the user mode, the IoCompletionRoutine function captures the failed response, changes its target from the public computer to the cloud storage, and makes the request issue again. Then, I/O manager issues the request to the cloud storage without sending the failed response to the application, and the new response will be returned successfully. It is also possible that the IoCompetionRoutine bypasses the successful response when the requested file exists on the public computer. In the Figure 5, we mark an original file request as IRP A which target is a public computer, and mark a redirected request of application files as IRP B which target is cloud storage.

With the deploying stage, when the user connects the public computer to the UAA framework on the cloud storage, the applications stored on the cloud storage cloud be executed on any public computer.

*3) Composition (User Profile Automation):* Unlike application part of the UAA framework, composition of the user profile is intuitive. The user may choose any user profile they want to carry and store it into the cloud storage through the UAA framework. When the UAA framework is running, it will back up the user profile of the public computer, then change it to the user's one which is already stored on the cloud storage. For example, if a user stores his/her preferred wallpaper, screensaver, favorite links, or even internet cookies into the cloud storage, when the user is connected to the cloud storage from the public computer, the UAA framework will automatically compose the user profile into the public computer. Lastly, there is a cleanup process which restores the original user profile into the public computer right after the connection is dropped. The cleanup process also removes the file I/O filter layer from the public computer. With these automatic composing user profiles, the user could work on the ubiquitous application access cloud.

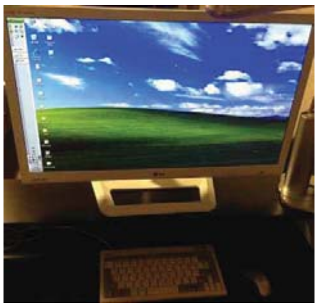
Fig. 6. Initial status of public computer will be restored when the connection to the UAA framework is dropped.

Fig. 7. When a user connects to the UAA framework, the user might choose any application with their user profile.

## IV. Prototype Implementation

This section shows a prototype on the Windows platform leveraging Amazon S3, and then presents performance metrics. This paper present an implementation of the UAA framework for applications and user profile on Windows platform leveraging Amazons S3 cloud storage [7]. Figure 6 shows the initial status of the public computer which has its own applications and default settings of profile, and in Figure 7, when an end-user connects to the UAA framework from the cloud storage, his/her ubiquitous application access including the user profile will be composed to the computer. Then, as soon as the connection to the framework is dropped, the original status of the public computer will be restored.

We have performance tests to show how different of application executions which is based on the cloud storage, the device, or the local computer. The experimental test-base is the Intel dual core 2.6GHz*2, 2.93GB RAM, Microsoft Windows XP SP3, with 144.0Mbps WIFI network. For UAA based on the cloud storage, we choose the Amazon S3 storage in Tokyo, Japan as the nearest cloud storage from Korea where the test was performed. In order to recognize the cloud storage as a typical disk partition, WebDrive 11.0 lets the user map a network drive to cloud services and use remote files like they right on the computer [13]. For the device based test, we use the SanDisk 1.0GB USB 2.0 device with a 480Mb/s bandwidth.

We first examine the upload and download time of different size of files with cache or not in Table I. The small file is 1KB text, while the large file is 495MB movie, and after reconnection to the cloud storage or the device, accessing file

TABLE I. COMPARISON OF CLOUD STORAGE AND USB DEVICE

| **Check Lists** | | **Cloud Storage** | **Device** |
|---|---|---|---|
| Small file (1KB) | Upload | Less than 1sec | Less than 1sec |
| | Download | Less than 1sec | Less than 1sec |
| | Download after reconnection | Less than 1sec | Less than 1sec |
| Large file (495MB) | Upload | 7min 29sec | 1min 49sec |
| | Download | 19sec | 41sec |
| | Download after reconnection | 6min 22sec | 41sec |

TABLE II. PACKAGING AND EXECUTING THE MUSIC PROGRAM

| **Check Lists** | **Cloud Storage** | **Device** | **Public Computer** |
|---|---|---|---|
| Packaging or Setting up the program | 5min 19sec | 1min 3sec | 19sec |
| Executing the program | 15sec | 3sec | Less than 1sec |

does not have any cache. As a result, uploading and downloading of the small file are similar between cloud storage and the device, but the large file takes longer on the cloud storage. However, accessing to the cached file is even faster than the device since the WebDrive supports the cache itself. In this experiment, uploading and downloading any file into the cloud storage depend on several facts; where the cloud storage is, what kind of network the public computer uses, and how the WebDrive works internally. Therefore, even though using the cloud storage is slower than the device, it has more benefits that are the cloud storage's unlimited space and global cloud datacenter that make the user use the nearest one.

Table II shows the observations of the time for packaging the program into the cloud storage and the device, and setting it up on the public computer. The tested media program is Winamp 5.32 and its total size is 1.30MB. In addition, executing the program stored on the cloud storage, the device, and the public computer is also examined. The time depends on how many files the program has, and like above experimental, the place of cloud data center, network environments, and the WebDrive itself have a decisive effect on the result again.

## V. CONCLUSION

In this paper, we extend the ubiquitous data access to ubiquitous personalized application access leveraging cloud storage. Towards facilitating access to personalized access to applications natively from anywhere on any device, we proposed and built UAA - a ubiquitous application access leveraging cloud. First, we developed a framework for application virtualization that decouples an application from any specific platform or device and enables access to personalized application access at per user level. Next we provide an implementation of this framework for applications on Windows platform leveraging Amazon S3 cloud storage using application virtualization framework. Note that this same implementation framework can be extended to Linux, Mac iOS, Android based platforms - which is in our future implantation roadmap.

Future work for wider supports of large-scale ubiquitous application access cloud such as other operating systems, other cloud storages, or even other client machine like a smart phone is needed. It is also possible to improve the performance when the framework works on the remote storage by using cache or making full use of local applications first not to have much streaming requests.

## REFERENCES

[1] Dropbox, http://www.dropbox.com

[2] Microsoft OneDrive, https://onedrive.live.com/

[3] Apple iCloud, https://www.icloud.com

[4] Google Drive, http://drive.google.com

[5] Apple AppStore, http://store.apple.com

[6] Google Play, https://play.google.com/store

[7] Amazon Simple Storage Service(S3), http://aws.amazon.com/s3

[8] K. Yang, X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage", IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol 25, Issue 7, pp. 1735-1744, 2014

[9] A. Cidon, R. Stutsman, S. Rumble, S. Katti, et al., "Copysets: reducing the frequency of data loss in cloud storage", USENIX Annual Technical Conference (ATC) 2013, pp. 37-48

[10] D. B. Terry, V. Prabhakaran, R. Kotla, M. Balakrishnan, et al., "Consistency-based service level agreements for cloud storage", Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP), pp. 309-324, 2013

[11] P. Mahajan, S. Setty, S. Lee, A. Clement, et al., "Depot: cloud storage with minimal trust", ACM Transactions on Computer Systems, Volume 29, Number 4, Article 12, 2011

[12] I. Kim, M. Hwang, W. Lee, C. Park, "u-PC: personal workspace on a portable storage", The 4th Intionational Conference Mobile Technology, Application and Systems (Mobility Conference 2007), Singapore Chapter for ACM, 2007, pp. 228-233.

[13] Sound River Technology WebDrive, www.webdrive.com