Towards Data Interoperability of Cloud Infrastructures using Cloud Storage Services

Tamas Pflanzner[1] and Attila Kertesz[2,1]

[1] University of Szeged, Department of Software Engineering
H-6720 Szeged, Dugonics ter 13, Hungary
tamas.pflanzner@gmail.com
[2] MTA SZTAKI Computer and Automation Research Institute
H-1518 Budapest, P.O. Box 63, Hungary
kertesz.attila@sztaki.mta.hu

Abstract. Cloud Computing is becoming more and more popular, and various cloud services have appeared to make our lives easier. Mobile devices can also benefit from Cloud services: the huge data users produce with these devices are continuously posted to online services, which may require the modification of these data. Using cloud storage services together with computation-intensive infrastructure services can provide a suitable solution for these needs. In this paper we address the open issue of data interoperability in clouds, and propose an approach to manage and share user data produced by mobile devices in different IaaS clouds. The approach is exemplified with an image generator application, and the performance of the application is evaluated with Android devices and a private IaaS cloud.

1 Introduction

Nowadays, as Cloud Computing is becoming more and more popular, and various cloud services have appeared to make our lives easier. These services are offered at different cloud deployment models ranging from the lowest infrastructure level to the highest software or application level. Within Infrastructure as a Service (IaaS) solutions we can differentiate public, private, hybrid and community clouds according to recent reports of standardization bodies. The previous two types may utilize more than one cloud system, which is also called as a cloud federation [6]. One of the open issues of such federations is the interoperable management of data among the participating systems. Another popular family of cloud services is called cloud storage services. With the help of such solutions, user data can be stored in a remote location – in the cloud. Mobile devices can also benefit from Cloud services: the enormous data users produce with these devices are continuously posted to online services, which may require the modification of these data. Nowadays more mobile devices are sold compared to traditional PCs [7], and Android devices are more and more popular [8]. The aim of our research is to develop a solution that uses cloud storage services together

with infrastructure services of cloud federations to enhance the capabilities of mobile devices.

Regarding related works, the need for data interoperability and the extensive use of cloud storage services have been identified by various research and expert groups (eg. [5, 3, 1]). Managing user data in the cloud also raises privacy issues [9, 4] that need to be taken into account during data processing. Nevertheless in this paper we refrain from legal issues and focus on interoperability problems. Dillon et. al [2] gathered several interoperability issues that need to be considered in cloud research, and named a new category called Data Storage as a Service to draw attention to the problem of data management in clouds.

In this paper we address the open issue of data interoperability in clouds, and propose an approach to manage and share user data produced by mobile devices in different IaaS clouds. Therefore the main contributions of this paper are: (i) envisioning a solution for the data interoperability problem of cloud federations, (ii) the development of an image generator application that interconnects mobile devices, IaaS services and cloud storage services, and (iii) the evaluation of our proposed approach using mobile devices and an IaaS Cloud.

The remainder of this paper is as follows: Section 2 presents a classification of cloud storage providers, and describes our approach for data interoperability; Section 3 introduces a scenario for managing user data in a cloud produced by mobile devices. Finally, Section 4 discusses the performed evaluations, and the contributions are summarized in Section 5.

2 An Approach for the Data Interoperability Problem in Cloud Federations

As we have seen in the introduction, retrieving and sharing user data and virtual images among different IaaS Clouds is an open issue. Without concerning data privacy issues, it is also not an easy task to move a user application from one Cloud infrastructure to another. Virtualization techniques and virtual image formats different providers support to run on their Virtual Machines (VMs) are usually incompatible. Retrieving a user's Virtual Appliance (VA, which is a specialized image hosting the user application) from an IaaS Cloud is impossible in most cases, not only in case of commercial providers, but also in academic solutions. Therefore we propose an approach to retrieve and share user application data among different providers with the help of Cloud Storage Services. In this way VAs running at different cloud infrastructures can manage the same data at the same time, and the users can access these data from their own local devices without the need for accessing any IaaS Clouds. This idea is demonstrated in Figure 1. A classification on cloud storage service providers (CSSP) is shown in Table 1. We compared 10 well-known providers focusing on their most important properties. From these providers we have chosen Dropbox [18] as the CSSP to demonstrate our approach, because it has the highest variety of development kits available and its API was the most reliable according to our evaluations. Regarding the IaaS middleware we have chosen OpenNebula.

Table 1. Classification of Cloud Storage Service Providers

Provider	Initial Storage (GB)	Bonus (GB)	Max. Storage (GB)	Supported OS	Mobile Platforms
Google Drive [14]	5	-	5	Win, Mac	iOS, Android
Amazon CD [15]	5	-	5	Win, Mac	iOS, Android
SkyDrive [16]	7	-	7	Win, Mac, Linux	iOS, Android
SpiderOak [17]	2	1	10	Win, Mac, Linux	iOS, Android
Dropbox [18]	2	0.5	18	Win, Mac, Linux	iOS, Android
IDrive [19]	5	1	50	Win, Mac	iOS, Android
SugarSync [20]	5	0.5	32	Win, Mac	iOS, Android
Glide [21]	30	-	30	Win, Mac, Linux	iOS, Android
CX [22]	10	0.3	16	Win, Mac	iOS, Android, Kindle Fire
Memopal [23]	3	0.5	13	Win, Mac, Linux	iOS, Android, Blackberry

Provider	Version Control	Encryption	Num. of devices	API	SDK
Google Drive [14]	+	-	-	+	Java, Python, PHP, .NET, Ruby
Amazon CD [15]	-	-	8	-	-
SkyDrive [16]	+	-	-	+	Android, .NET, iOS
SpiderOak [17]	+	+	-	+	-
Dropbox [18]	+	+	-	+	iOS, Android, Python, Ruby, Java, OS X
IDrive [19]	+	+	1	+	-
SugarSync [20]	+	+	1	+	-
Glide [21]	+	-	6	-	-
CX [22]	+	+	-	+	-
Memopal [23]	+	+	10	+	-

In the next section we introduce a use case and a sample application that demonstrates the usability of this approach.

3 Managing mobile data in Clouds

As a use case, we have identified a real world scenario that requires interoperable data management among cloud infrastructures to manage user data produced by mobile devices. Though the computing capacity of mobile devices has rapidly increased recently, there are still numerous applications that cannot be solved with them in reasonable time. Our approach is to utilize cloud infrastructure services to execute such applications on mobile data stored in cloud storages. The basic concept of our solution is the following: services for data management

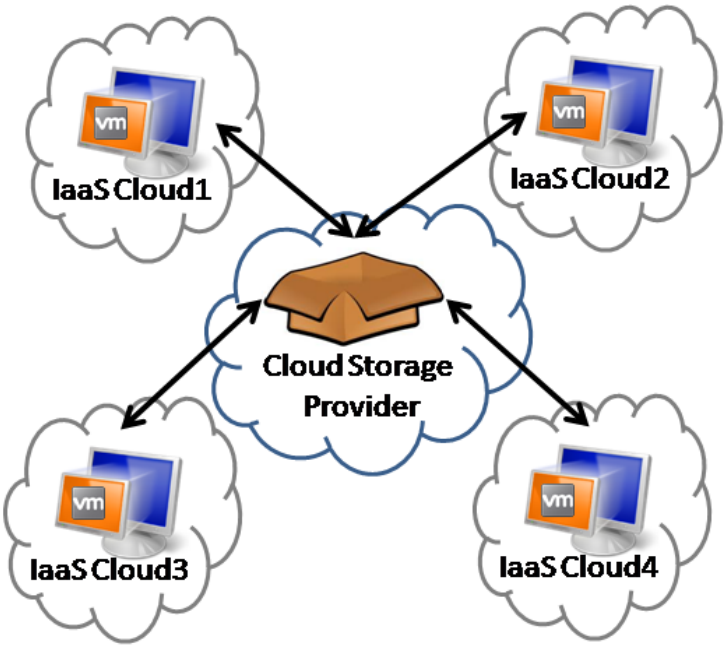

Fig. 1. An envisioned solution for data interoperability in Clouds

are running in one or more IaaS systems that keep tracking the cloud storage of a user, and execute data manipulation processes when new files appear in the storage (see Figure 2).

The service running in the cloud can download the user data files from the cloud storage, execute the necessary application on these files, and upload the modified data to the storage service. Such files can be for example a photo or video made by the user with his/her mobile phone to be processed by an application unsuitable for mobile devices. In our solution currently developed for Android devices, there is a possibility to configure the processes to be performed on the data with a separate configuration file, which is automatically created and managed by a mobile application running on the users device. This application is also responsible for communicating with the cloud storage, which is Dropbox in our case. The file manipulation applications have been created as a virtual appliance, and have been pre-deployed in our SZTAKI cloud infrastructure based on OpenNebula, to perform our evaluations.

In order to exemplify the usability of this generic approach, we have created a concrete application called *FolderImage*, which can be used to manipulate pictures produced by mobile devices. This program creates thumbnails of each image of the appropriate folder then ensembles them into a single image (called as *folder* image) that represents the folder and gives an overview of its contents

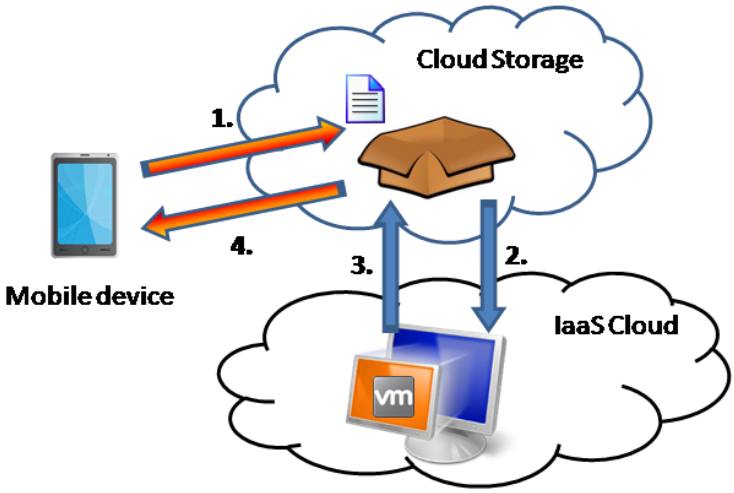

Fig. 2. Enhancing data management of mobile devices

to the user. This app can be really useful by providing a glimpse of a directory, when a user has thousands of pictures spread over numerous directories, and she is looking for a specific one.

We have developed an Android application that is able to perform this task locally on the user's device, and a Java application performing the same task encapsulated in a VA, deployed in a local cloud. In the following subsections we introduce these applications, then in Section 4 we compare the performance of the two approaches.

3.1 The FolderImage Android application

In this subsection we introduce in detail how the *FolderImage* Android application has been developed and can be used. This application can be used in two modes: (i) in local operation, when the folder image is generated by using the computing resources of the actual device, and (ii) in cloud operation, when the application communicates directly with Dropbox, in this case the pictures can be uploaded on demand, or synchronized continuously, and the folder image is generated in the cloud, which is downloaded to the device after completion. These modes can be triggered by clicking on the appropriate button in its graphical interface (see Figure 3).

The implementation of the application uses the Android [10] and Dropbox API [11]. It needs to read and write files to the local storage of the device, to access the internet, and to securely communicate with a Dropbox account storage. These capabilities are denoted in its 'AndroidManifest.xml' file. The application itself needs to be registered at the official Dropbox developer website [12]. The 'APP-KEY' retrieved during registration need to be set in the application to

allow secure communication and access to the user's storage. If no such key information is given, the application offers the possibility to log in through the device browser to a Dropbox account – in this case the permanently retrieved credentials will be used to access the remote storage.

A screenshot of the graphical user interface (GUI) of the application is shown in Figure 3. After installing and starting the FolderImage application, the user will see a similar look with three buttons. Clicking on the first, top button, it will automatically log in to Dropbox, and after the successful login, the name of the account holder is displayed above the button. The second button can be used to trigger a folder image creation in the cloud, and the third one to perform the folder image generation locally. For simplicity, this prototype uses the 'Photos' directory, and creates a file called 'folder_image_Photos.jpg'. (This prototype can be easily extended to handle any directories on the device storage.) This generated image is also shown in the application GUI, under the buttons. Status messages and updates on image up-, downloads and generation are also shown between the buttons and the folder image. Though this GUI is relatively simple, it is not an easy task to show proper look on devices having different display resolutions.

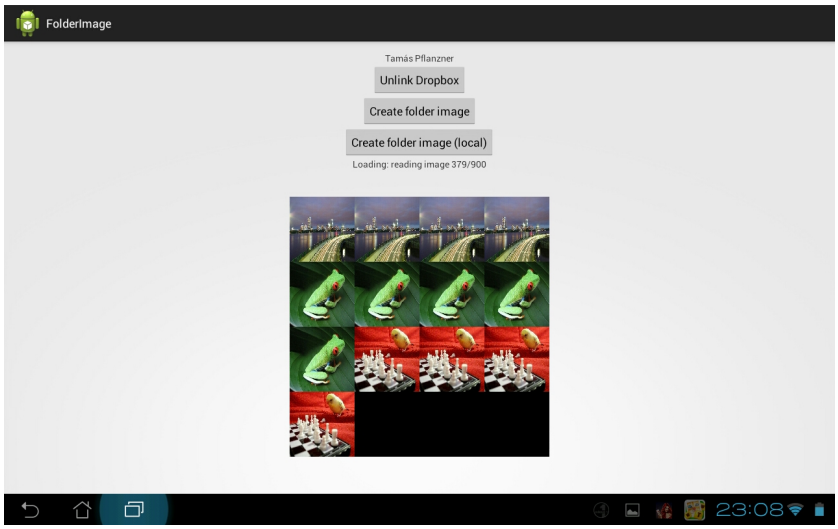

Fig. 3. A screenshot of the FolderImage Android application

As we mentioned, the second and third buttons can be used to generate the folder image. These events trigger the 'createFolderImage' and 'createFolderImageLocal' methods respectively. These methods have five similar steps:

1. list: to generate a list of the images the actual folder contains;

2. download: to access the images of the folder;
3. resize: to generate thumbnails of the images;
4. create: to ensemble the thumbnails;
5. upload: to save the created folder image.

These steps are timestamped, therefore comparison between the local and cloud computation can be easily done. The second step of the local computation (when each image is loaded to the memory) had to be optimized for the local version, because most Android device has a relatively small internal memory for this purpose (eg. 32 MBs). Therefore in this step it is not the transfer time that needs to be considered, but the loading of the pictures to the internal memory one by one to generate separate thumbnails (on the contrary to the cloud version, here the memory is too small to keep all pictures in the memory at the same time). For the cloud version, the images are downloaded from the Dropbox folder to the VA, where the computation is performed. In step 3 thumbnails with 50x50 pixels are generated, then in step 4 they are organized into a matrix that will represent the folder image. Finally, in step 5 the local version only saves the folder image file, while the Java application in the cloud uploads the generated folder image file to the Dropbox storage, which will be retrieved by the device application.

3.2 Image generation in the cloud

As we have depicted in Figure 2, the idea of our scenario is to move computation-intensive tasks from a mobile device to the cloud. Therefore we have also created a Java application called *ImageConverter* that is capable of performing the 5 steps discussed in the previous subsection. We have encapsulated this application to a VA, deployed, and started it as a web service running in a local cloud. It also has a direct connection with the user's Dropbox storage. It can continuously synchronize the image directory, and perform the folder image generation once a new image is added to the folder. It can also be set to listen to a specific setup file that instructs it to execute certain methods (eg. performing different kinds of image manipulation processes). Once the 'createFolderImage' method is called from the Android application, the setup file is refreshed, and the web service running in the cloud is notified about this change, which triggers the folder image generating and uploading processes.

If we deployed web services of similar VAs into different IaaS providers, we could execute a set of cloud services that can access the same data storage under a Dropbox account. In this way we could handle and manage data in an interoperable way among different IaaS solutions.

4 Evaluation

As we have already mentioned, we performed our evaluations on a private IaaS Cloud. It has been developed by a national project called SZTAKI Cloud [13],

Table 2. Capabilities of the used devices in the evaluation

	Samsung Galaxy Mini (phone)	**Asus Slider SL101 (tablet)**	**Cloud VM1**	**Cloud VM2**
OS	Android 2.2	Android 4.0	Ubuntu 12.04 64bit	Ubuntu 12.04 64bit
CPU	600 MHz	1 GHz (dual-core)	1 CPU	4 CPUs
RAM	384 MBs	1 GB	1 GB	4 GBs

which was initiated to perform research in clouds, and to create an institutional cloud infrastructure for the Computer and Automation Research Institute of the Hungarian Academy of Sciences based on the latest research results to enable a significant improvement of the local IT infrastructure. This new cloud-based infrastructure is more effective and cost efficient than the former physical infrastructure built on traditional concepts and individual specific servers. This improved institutional infrastructure could even be the first milestone in a global academic paradigm shift, and will serve as a base for further research investigations.

To perform the measurements, we deployed the *ImageConverter* VA to the SZTAKI Cloud. We have deployed them to two different types of virtual images: one having one processor and one GB memory (VM1), and the other 4 processors with 4 GBs of memory (VM2). Meanwhile we have also tested the *FolderImage* application on two different Android devices: on a phone and a tablet. Table 2 summarizes the used resources during the evaluations. Tables 3 and 4 show the evaluation results corresponding to the 5 steps of the executed methods introduced in the previous section. In the first round of measurements we considered a folder containing 450 images, while in the second round we manipulated 900 images.

Table 3. Evaluation results for 450 images

Device	**1. list (ms)**	**2. download (ms)**	**3. resize (ms)**	**4. create (ms)**	**5. upload (ms)**	**Sum (ms)**
Android phone	879	199738	872	10631	1587	213707
Android tablet	304	68334	286	3480	491	72895
Cloud VM1	20	173	135	277	326	931
Cloud VM2	14	189	68	203	181	655

The evaluation results clearly show the differences among the different types of executions. Regarding the Android devices, the tablet performed the generation 3 times faster then the phone in both rounds of experiments. The web service running in VM2 type virtual machine performed two times faster then the other deployment at VM1. The local execution on the Android devices are significantly slower (more then 100 times) then the image generations performed in the cloud. These measurements fulfil our expectations, therefore it is worth

Table 4. Evaluation results for 900 images

Device	1. list (ms)	2. download (ms)	3. resize (ms)	4. create (ms)	5. upload (ms)	Sum (ms)
Android phone	2972	401312	1496	21643	3173	430596
Android tablet	971	133575	509	6957	998	143010
Cloud VM1	44	220	300	575	702	1841
Cloud VM2	24	191	73	541	239	1068

both in terms of computation time and energy efficiency to move computation-intensive tasks to clouds from mobile devices.

Nevertheless we have to mention that file transfer times can also affect these execution times. When a significant amount of data should be moved from a mobile device with slow internet connection may result in much worse performance, but if the new images are occasionally uploaded to Dropbox by using high speed wifi connection, still good results can be achieved.

5 Conclusion

In this paper we addressed the open issue of data interoperability in clouds, and proposed an approach to manage and share user data produced by mobile devices in different IaaS clouds. We have introduced a solution for solving the data interoperability problem of cloud federations by using cloud storage services. We have shown how to develop an image generator application that interconnects mobile devices, IaaS services and cloud storage services, and evaluated the prototype application using mobile devices and a private IaaS cloud.

Our future work aims at extending the functionalities of the designed mobile application, and supporting additional IaaS and cloud storage providers, and mobile platforms.

6 Acknowledgment

The research leading to these results has received funding from the CloudSME FP7 project under grant agreement 608886, and it was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TAMOP 4.2.4. A/2-11-1-2012-0001 'National Excellence Program', and by the European Union and the European Social Fund through project FuturICT.hu (TAMOP-4.2.2.C-11/1/KONV-2012-0013).

References

1. J. Bozman. Cloud Computing: The Need for Portability and Interoperability. IDC Executive Insights, August 2010.

2. T. Dillon, C. Wu, and E. Chang. Cloud Computing: Issues and Challenges. In proc. of the 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 27–33, 2010.
3. Fraunhofer Institute for Secure Information Technology. On THE Security of Cloud Storage Services, SIT Technical reports, March 2012. Online: http://www.sit.fraunhofer.de/content/dam/sit/en/documents/Cloud-Storage-Security_a4.pdf
4. F. Gagliardi, S. Muscella. Cloud Computing – Data Confidentiality and Interoperability Challenges. In book: Cloud Computing, Computer Communications and Networks, Springer-Verlag London, pp. 257–270, 2010.
5. K. Jeffery, and B. Neidecker-Lutz. The Future of Cloud Computing, Opportunities for European Cloud Computing beyond 2010. Expert Group Report, January 2010.
6. G. Kecskemeti, A. Kertesz, A. Marosi, P. Kacsuk. Interoperable Resource Management for establishing Federated Clouds. In book: Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice, IGI Global (USA), pp. 18-35, 2012.
7. Mobile and PC shipments. Online: http://blog.modernmobileapps.com/post/2011/05/03/Qualcomm-A-Hand-in-Everything.aspx, March 2011.
8. Mobile OS statistics. Online: http://www.globalnerdy.com/2013/02/14/idcs-smartphone-stats-for-4q-2012-and-a-review-of-their-mobile-os-share-prediction-for-2015/, Feb. 2013.
9. Sz. Varadi, A. Kertesz, M. Parkin. The Necessity of Legally Compliant Data Management in European Cloud Architectures. Computer law and security review (CLSR), vol. 28, issue 5, pp. 577-586, Elsevier, 2012.
10. Android website. Online:http://www.android.com/, April 2013.
11. Dropbox API website. Online:https://www.dropbox.com/developers/reference/api, April 2013.
12. Dropbox developer website. Online:https://www.dropbox.com/developers, April 2013.
13. The SZTAKI Cloud project website. http://cloud.sztaki.hu/en/home, 2013.
14. Google Drive. Online: https://drive.google.com/, May 2013.
15. Amazon Cloud Drive. Online: http://www.amazon.com/clouddrive/, May 2013.
16. Microsoft SkyDrive. Online: http://windows.microsoft.com/skydrive/, May 2013.
17. SpiderOak. Online: https://spideroak.com/, May 2013.
18. Dropbox. Online: https://www.dropbox.com/, May 2013.
19. IDrive. Online: https://www.idrive.com/, May 2013.
20. SugarSync. Online: https://www.sugarsync.com/, May 2013.
21. TransMedia Glide. Online: http://www.glideconnect.com/, May 2013.
22. CX. Online: https://www.cx.com/, May 2013.
23. Memopal. Online: http://www.memopal.com/, May 2013.