Dropbox Client

ASOIS Project

Necula Cristina-Suzana
Faculty of Automatic Control and Computer Science
SSA

Problem Definition

Cloud services are becoming more and more popular as they provide a way to offload local computers computational resources usage. By offering file hosting services, cloud computing attracts a big number of users, as one user can access files stored on a remote server from almost any device that has an active internet connection. More and more users are considering to utilize cloud-storage services, particularly for depositing large amounts of data. Cloud-services are provided to users as commercial services, with a paid subscription, or as consumer services, which are free but impose certain limitations. A series of companies – Google, Dropbox, Amazon or Microsoft – offer storage services. While Google and Dropbox are perceived as consumer service providers, even though they have support for business solutions, Amazon and Microsoft are targeted as business service providers.

Having a big range of services from where to choose, and considering the fact that file storage is the only interest of a user in the services provided listed before, one and an important aspect that has to be considered when using such system, is the latency of uploading or downloading files. Timing needed for uploading or downloading a file is influenced by internet speed connection, position of data centers, devices from which the requests are being performed but also services implementation. This project aims to measure latency for Dropbox for writing operations when uploading and downloading files of different sizes. To achieve this goal, we aim to implement a Dropbox client that uploads a set of files on a consumer Dropbox account and then downloads the uploaded files. Time is to be measured for both operations on every file.

Dropbox is a file hosting service that offers cloud storage, file synchronization, personal cloud and client software. It is available for a large series of operating systems: Windows, Mac OS, Linux Based, Android, iOS, Blackberry and Windows phone. Dropbox provides an application that, when downloaded and registered, creates a folder inside the device file system that is synchronized on all the devices that use the same account to login.

Design

The proposed software solution for solving the problem stated in the previous chapter, we implement a Dropbox Client Application. Client Application has the following features:

1. Generate files of various dimensions
2. Upload content of given directory to Dropbox
3. Download content of given directory from Dropbox to local machine
4. Measure timing results for upload and download and corresponding file size

The client receives as an input parameter a file name representing the folder in which files will be generated for future upload. It then creates a new directory and generates an arbitrary number of files with different dimensions. The created directory content is covered and all the

files are uploaded to Dropbox, measuring file size and the time needed for the actual upload. After all the files are uploaded, client application downloads them, also measuring every file size and time needed for download. All writing operations have to be sequential as we want to measure the latency of those operations.

For the purpose of this project, file sizes are to vary and the folder should not contain two files with same dimension. In order to be able to compare time needed for upload and download, same files that are uploaded to Dropbox have to be downloaded afterwards.

Dropbox has a special section for developers that want to include cloud sharing in their applications. In order to accomplish this feature, an SDK is provided under an open source license. The SDK is available for download for different programming languages: Swift, Python, .NET, Java, JavaScript, PHP and Ruby. To accomplish what we have proposed, Dropbox SDK has to be included in the client application.

Implementation and Experimentation

In the implementation and experimentation phase, the design, or some aspect of the design is implemented to demonstrate feasibility. The experimentation involves developing a reasonable hypothesis and designing and performing an experiment to test this hypothesis.

For Dropbox Client Application implementation, we used Python 2.7.5+ and Python SDK for Dropbox API v2. The SDK was downloaded via github.com and installed using the provided setup script.

In order to create an application that is able to upload and download files to a Dropbox folder, one has to register a Dropbox API application. This is accomplished using App console. Developer has to login before creating an app. First step when creating the application on Dropbox developers page, is to select the API that is going to be used. There are two available options: Dropbox API and Dropbox Business API. Since we want to measure latency using a regular Dropbox account, we chosen Dropbox API. Step 2 is to choose the type of access one needs: access to a single folder specifically for the current app and access to all files and folders in the connected user Dropbox folder. For this application, we used app folder access. Step 3 and the last step is to choose a name for the application. After application creation, we generated an access token that is going to be used inside the implementation of the client application so we can access API v2. The account token can only be used to access the account of the user registered when creating the application.

After creating the application, a new folder, named Apps, is added in the Dropbox folder of the registered user. Inside it, there is another directory that has the name user introduced when creating the application. This is the folder that is going to be used when interacting with the application we implement.

For simplicity, the project contains only one file, Main.py. We implemented the functionalities stated in Design chapter pursuing the following workflow:

- Instantiate Dropbox
- Create new folder that will contain the files to be uploaded
- Generate 10 binary files inside the folder with sizes from 14.3 MB to roughly 143.2 MB
- For each file in the created folder, print size and upload it to Dropbox; measure time needed only for upload
- For each file previously uploaded, print size and download it to local directory; measure time needed for download

To make calls to the API, a Dropbox instance is required. When instantiating Dropbox, the access token previously generated has to be passed as an argument:

dbx = dropbox.Dropbox(TOKEN)

To test if the right account was linked and if the instantiation has not failed, we printed the linked Dropbox account email:

print 'Linked Dropbox account: ', dbx.users_get_current_account().email

Dropbox SDK exposes functions for both upload and download. For upload, we used files_upload function:

files_upload(*f*, *path*, *mode=dropbox.files.WriteMode.overwrite*, *autorename=True*, *client_modified=None*, *mute=True*)

Data represents the content of the file to be uploaded and is obtained using read () method. file_path represents the path in the user's Dropbox to save the file. mode selects what to do if the file already exists. If there's a conflict, as determined by **mode**, have the Dropbox server try to autorename the file to avoid conflict by setting autorename parameter to true. client_modified records timestamps of user requests. Mute parameter configures whether or not a notification should be sent to the user when modifications are performed.

For download, the exposed function is more straight-forward:

md, res = dbx.files_download(file_path)

The file_path is the path of the document to be downloaded. The metod returns downloaded file metadata, md, and a response, res. We use response to get the size of the downloaded file:

len(res.content)

Time is measured using time.time() method before and after we call files_upload and files_download. The actual time for performing an operation is calculated as the difference between start time and end time.

The application was tested on a machine running Ubuntu 13.10 32-bit operating system, 4GB RAM, Intel® Core™ i5 CPU M 480 @ 2.67GHz × 4 Processor and 83.3GB disk space. An internet connection speed test was performed before testing the application, using sppedtest.com, showing a download speed of 26.87Mbps and 2.40Mbps upload speed.

Figure 1 shows running time for uploading generated files with different sizes. As it was expected, time necessary for uploading a single file increases as the file size increases. Time varies from 8.914 seconds when uploading a file of 14.32 Megabytes, up to 101.268 seconds when uploading a file of 143.17 Megabytes. We can notice a strange behavior for the file with size 100.22 Megabytes. Time should have increase linearly with file size, but, due to variations in internet connection speed, strange behaviors like the one previously stated, can appear when uploading and downloading files.

Figure 2 shows running time for downloading previously uploaded files from Dropbox. One can see that time varies between 0.528 seconds to 0.867 but, different from upload, when time increases almost linearly with the file size, download does not stay on the same pattern. Dropbox was installed as application on the device on which we performed the tests, so automatic syncing was being performed after every file upload. This is one of the causes of timing results for download: when calling *files_download* method, Dropbox syncs only differences between files stored in Dropbox folder and files stored on Dropbox servers. Since files were not downloaded in the same order they were uploaded, syncing is performed starting from the first files uploaded.

Average download time is 0.659 seconds, while average upload time is 55.542 seconds. This is caused by the difference between internet connection upload and download speed.

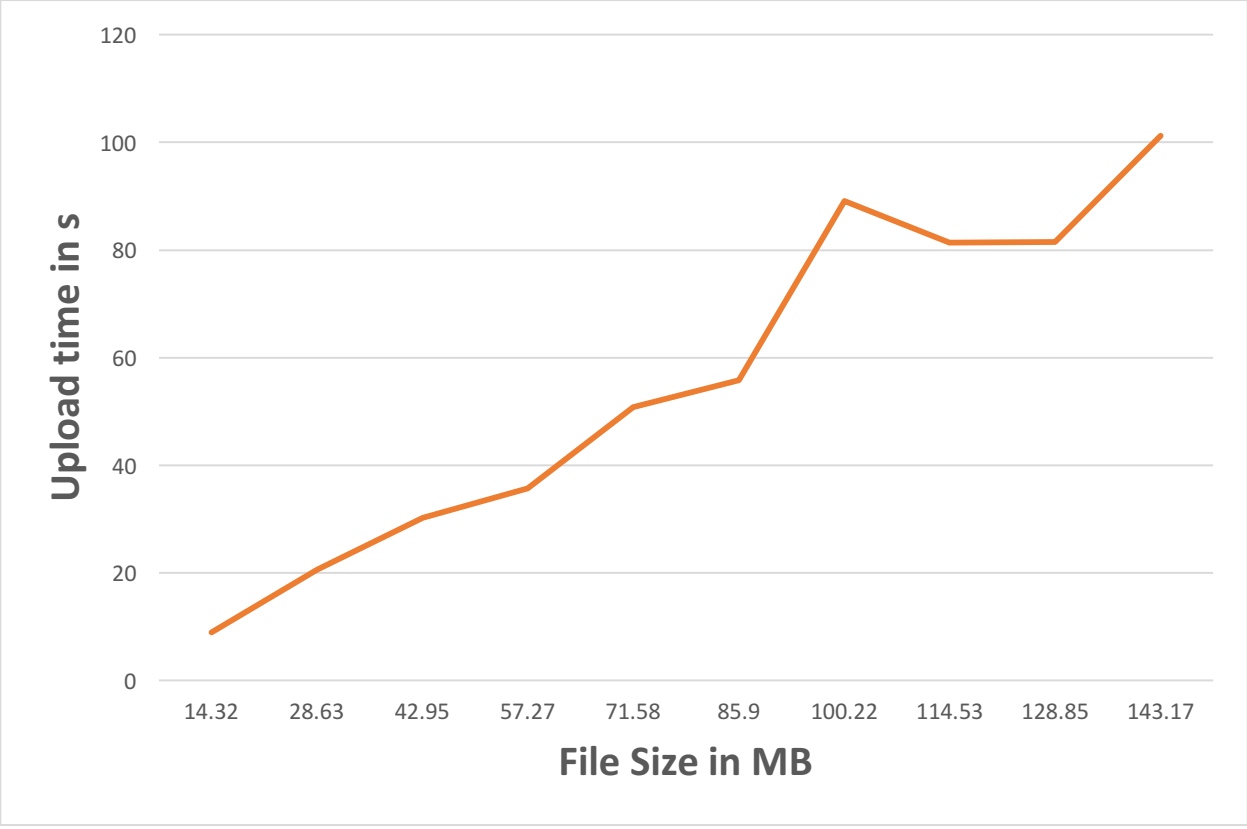

Figure 1

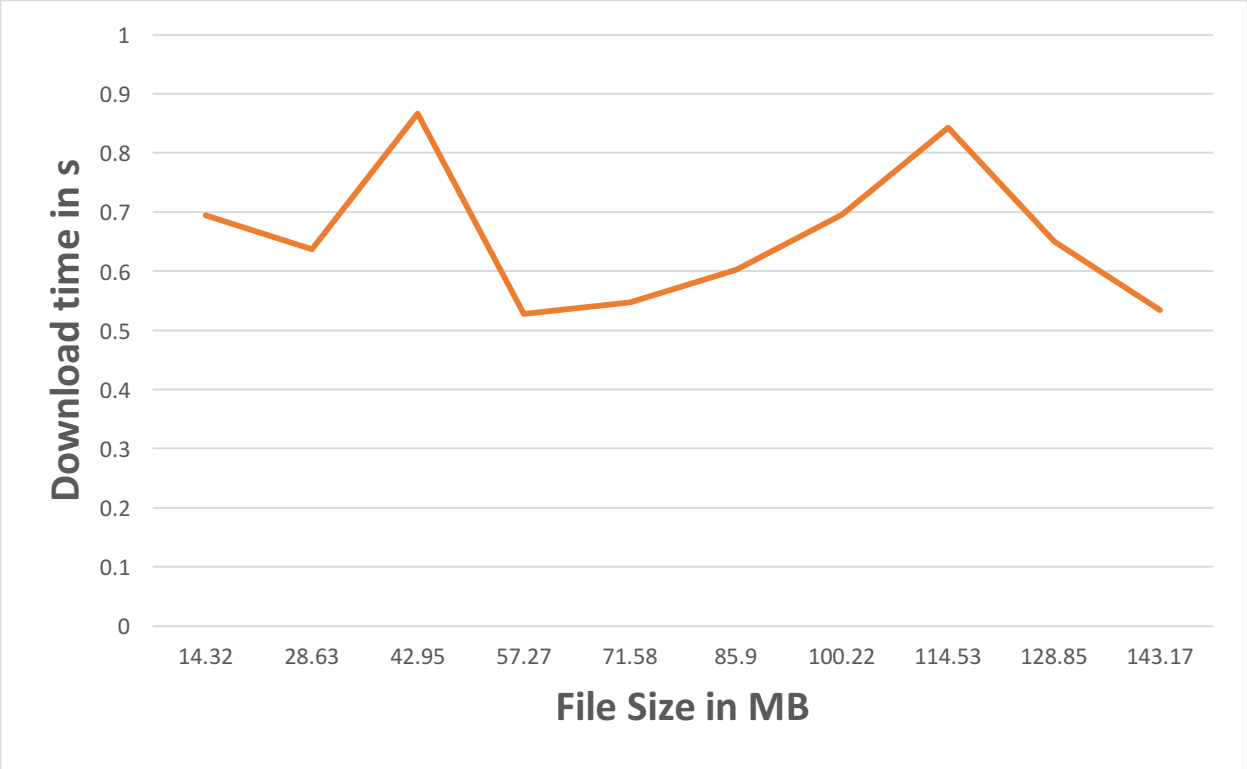
Download time in s
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
14.32
28.63
42.95
57.27
71.58
85.9
100.22
114.53
128.85
143.17
File Size in MB

Figure 2