Authentication and Integration of Cloud Storage Services

Mr.V.Manickavasagan[1] K.C.Soujanya[2] J.Keerthana[3] V.Gayathri[4]
[1]Assistant Professor [2,3,4]Student
[1,2,3,4]Department of Computer Science Engineering
[1,2,3,4]Velammal Institute of Technology, Chennai, Tamil Nadu, India

Abstract— In this Information age, the volume of data dealt everyday is enormous already, and is still growing. This has led to the development of various cloud storage services that help users store and manage the data with ease. So, it is natural that a user's data stored across multiple accounts in various cloud services, bringing with it the overhead of maintaining these accounts. Therefore, there is a need for unifying these accounts into a single module to help the user access them easily. The paper aims to solve this problem by integrating many of these services into a single cloud service. The service can be authenticated by the user only once. To access the files a tabbed view of all the services is being provided. The use of REST (Representational State Transfer) APIs of all the services to retrieve and list the files, is integral to the working of the module. The OAuth 2.0 protocol, which is the current industry standard, is followed to authenticate all of the services. The idea helps in managing various cloud platforms within a single unit. The access and management of projects is easier, since the integration of various cloud services is scalable.

Key words: REST (Representational State Transfer), OAuth 2.0, Authentication. Integration, Cloud Services

I. INTRODUCTION

The huge volume of information that is growing everyday requires proper storage and maintenance. The need for storage space led to the development of online cloud storage services that enable user to store their documents online and retrieve whenever needed. It is possible that the user maintains account with several services to obtain more storage. This leads to the overhead of managing different accounts. This problem of the user can be addressed by integrating these services in a single web portal. This project aims to solve this problem by creating a web portal consists of various online file storage and cloud services. The user has to authenticate these services. Once authenticated, the user can view the list of files in their account. The REST (REpresentational State Transfer) [1] APIs are mainly used to retrieve the files from the respective services. This way of listing files across all the authenticated services in a single web portal enable users to manage several accounts at once with ease. The authentication is performed using the OAuth 2.0 which is an open standard authorization protocol. The OAuth 2.0 protocol, which is the current industry standard, is followed to authenticate all of the services. This facilitates the user to allow or revoke access to the module whenever required. Various additional features such as searching across all files, and automatic suggestion of files, are also added to further enhance the experience of the user.

II. LITERATURE REVIEW

Chris Borckholder [2] has presented a framework that provides a solution for biomedical data integration needs. The built infrastructure consists of different biomedical service types providing an integrated view. It was developed to help the biomedical research community to query different services for clinical analysis. The services are integrated in a single framework named Vienna Cloud Environment. The application uses the client API, which interacts with data services through SOAP or RESTful web services. This framework enables unified access to data resources. However, necessary improvements have to be made to increase the performance. The REST API calls may lead to increase response time. Hence appropriate methodology has to be designed to reduce the RESTful API calls thus reducing the traffic.

Konstantinos [3] proposed a methodology for integrating and using geospatial services in the cloud. Data semantics play an extremely significant role in spatial data infrastructures by providing semantic specifications to geospatial data and enabling in this way data sharing and interoperability. Cloud computing may enable geospatial processing since it refers to efficient computing resources providing on demand processing services. This paper proposes a design and architectural framework for web applications based on open geospatial standards. The approach also includes data acquisition services that are essential in case of obtaining satellite images and remote sensing areas. Thus the framework integrates appropriate services in the cloud to combine and produce a solution for a specific need. However, the data acquisition services accessing data across distributed resources have to be handled.

Ghada ElSheikh [4] has proposed a data integration system called Service Oriented Data Integration based on Map Reduce (SODIM). This system is designed pool the resources of various cloud services and allow easy accessibility to them. On the hosting level, new architecture has emerged providing computing resources and storage on demand and adopts pay as you go pattern in charging users known as cloud computing. The development of cloud has lead to the development of other frameworks like map reduce for orchestrating the large number of machines and improving their performance. This paper focuses on an integration system that allows integration of web services and use map reduce technology to make the response time faster. The Map reduce Framework is mainly used for handling the heterogeneous data only. An existing application which is designed as a plug-in allows a user to manage their files hosted in single web service. For example, the user can authenticate their Google account and access their files in Google drive, Gmail, Google cloud, etc. However this application does not integrate web services of different platform. This will not suffice the need of the user because user may have many cloud storage account and they would like to use it all in a single place.

III. PROPOSED SYSTEM

A single web portal is designed to accommodate different cloud services that enables user to store their files online. The different cloud services that are taken for integration are Google drive, SkyDrive, Box and Dropbox. In addition to these cloud services, Zoho Docs is also integrated in the portal. The time taken to access, authenticate and download files from different storage services one at a time, will be very high. The proposed solution relieves the user of this delay by integrating the most frequently used file storage services in a single web portal. This portal is opened whenever the user wants to upload files from online cloud services like gdrive, box, dropbox, etc. The proposed methodology makes use of open standard protocol names OAuth 2.0 for authorization. Based on this protocol, an access token is generated while granting access. The grant of access token, checks for various credentials like client id and client secret in the developer's database. The generated access token is used to access and retrieve the resources. The access token may expire after one API call, or after an hour, or after a specific time as defined by the web service that generates them. Authentication is followed by listing of the files stored in that cloud service by that user. The response is then parsed using JSON(JavaScript Object Notation) parser and displayed. The user can select the files from different services and click on the "attach" button to upload the selected files. The selected files are stored in an array along with their unique id and an identifier to indicate the name of the cloud service. This array is then parsed and separate HTTP requests are initiated to the respective REST API.

IV. PROPOSED SYSTEM ARCHITECTURE DIAGRAM

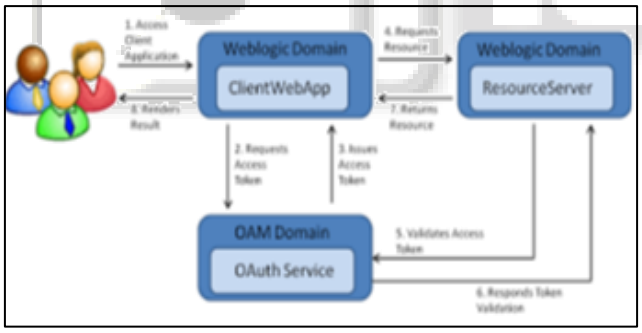


Fig. 1: Integrating Various Cloud Services

A. Integration of ZOHO Docs

Zoho [5] is an ideal solution for users who work remotely and dealwith a large number of files. It allows you to store a large number of files in all formats such as all file formats like images, music files, videos, zip, pdf, etc., and keep them organized in folders. This makes it easier to locate and access them. To start with, Zoho Docs offers 5GB of free space for every user under free plan. Zoho Docs also acts as a backup for all the files saved on your computer or other data storage gadgets. Being an online service, the files are well protected from the risk of getting lost or damaged. Files once lost, are difficult to recover in the digital form. Hence Zoho Docs offer safe and secure storage of files online.

B. ZOHO Docs API:

Zoho Docs API provides their partners and organizations with a more tightly integrated means of interacting with Zoho's applications for secure storage and document management infrastructure. Zoho Docs API can be integrated into an existing document system extending the range of functionality. Partners can build their own applications and leverage the collaboration and document sharing capabilities of the integrated Zoho services. The Zoho Docs APIs also provides a means for document storage so that partners can focus on their core applications and services rather than build and manage a storage solution.

C. Authorization in ZOHO Docs:

Zoho Docs API makes use of the generated auth token for authorization purpose. The developers have to generate the auth token and use them to access and retrieve the data from the Zoho Docs API service. To generate auth token, a request has to be sent to Zoho Accounts. The generated auth token can be removed or regenerated whenever needed. There are 2 ways of generating th auth token. They are as follows:

1) API Mode:

URL : https://accounts.zoho.com/apiauthtoken/nb/create
METHOD: POST
PARAMETERS: Scope, Username or E-mail ID, Password, Display name of the app.
RESPONSE: Auth Token

2) BROWSER Mode:

URL: https://accounts.zoho.com/apiauthtoken/create
METHOD: GET
PARAMETERS: Scope, Display name of the app
RESPONSE: Auth Token

D. Integration of Google Drive:

Google Drive [6] is a file sharing and synchronization service that allows Google users to store their documents in the online cloud storage. The files opened and viewed in any of the Google services can be stored in the Google drive with ease. The user files can be easily uploaded in Google drive and maintained safely. The files can also be collaboratively edited online. Google drive offers an initial storage of 15 GB free for all of its users. Google Drive supports larger file size when compared to other online file storing services. Files of size 1TB can be uploaded in Google drive. Google makes use of OAuth 2.0 open standard authentication protocol. Google comes up with APIs for different services which enables the developer to easily integrate and interact with the data and resources stored

E. Google API:

Google has developed their REST API to allow third party apps, client- side applications, and desktop applications to interact with the resources at the Google side. The developers have to register their app with the Google API console for enabling the access to their APIs. An application id is generated for every app which is the unique identifier of the app. Specific set of credentials is generated for every app while registering. The OAuth 2.0 credentials uniquely identify the appand its permission.

F. Authentication in Google Drive:

Google API can be accessed by an application using OAuth 2.0 by following the basic four steps:

1) Register the application: The application has to be registered with the Google Developer Console to obtain OAuth 2.0 credentials such as client ID and client secret. These credentials are known to both Google and your application.
2) Generate Access Token: The application has to obtain an access token before accessing the private data using Google API. The access token controls the degree of access to the various APIs. The access token is generated based on the scope parameter. The scope parameter sets the resources that have to be permitted by the access token. The user consent is got before sending the access token to the application.
3) Access the resource: The application sends the obtained access token to the Google API in the HTTP Authorization header. Access tokens limit the access to the specific set of APIs.
4) Refresh the access token: Access tokens have limited lifetime. They have to be obtained periodically. However, if the application needs long time access to the Google API, then refresh tokens can be generated.

Whenever the access tokens expire, refresh token can be used to regenerate them.

G. Integration of Sky Drive:

Sky Drive (now known as One Drive) [7] is a file hosting service that allows the users to upload and sync their files to a cloud storage. The users can access them from the web browser or any of the local devices. The user can also share their files publicly. This service offers 7GB of free space for users on their cloud storage. Sky Drive also makes use of OAuth 2.0 protocol for authorizing third party apps.

H. Sky Drive API:

Sky Drive [7] is REST-based and hence allows third party apps to integrate and fetch data after proper authorization using OAuth 2.0, an open standard protocol. The Live Connect Representational State Transfer (REST) API makes allows app developers to programmatically (HTTP Request) access the user information like web-activities, files, folders, contacts, calendar data, etc. from their media.

I. In Sky DRIVE:

The Live Connect API uses OAuth 2.0 protocol for authenticating its users. The developer has to register their app with the onedrive.live.com. The developer is provided with their unique credentials, namely client ID, client secret. These credentials are used when the app requests for access token from the live connect server. The Authorization code grant flow used in sky drive is depicted in the following figure 3. The steps in this flow of authorization are,

1) STEP 1: The client application redirects the resource owner (user) to the authorization endpoint. The parameters like client ID, client secret and scope are sent in the url.
2) STEP 2: The authorization server authorizes the resource owner through the user agent.
3) STEP 3: If the user has granted access, the user is redirected to the client application based on the redirect_URI parameter.
4) STEP 4: The User agent calls the client application which includes an authorization code and local state that was provided by the client.
5) STEP 5: The client requests for an access token by providing its credentials and authorization code to the Live Connect Authorization server.
6) STEP 6: If the credentials are valid, the access token is provided and communication proceeds.

J. Integration of Dropbox:

Dropbox [8] is a file hosting service that allows the user to bring together all their files in one place and share them easily. Dropbox creates a special folder in each of the user's devices. Any file dropped in this folder is synchronized with all the other devices. Hence the user can drop the file in their desktop and open them in their ipad. This special feature of dropbox attracts users to share their folders and files in dropbox. Dropbox authenticates using OAuth 2.0 protocol for authentication. For Free accounts, dropbox offers a memory of 2 GB for free.

K. Dropbox API:

Dropbox provides REST API for the app developers to easily access the user files and folders. The REST API access is controlled by OAuth 2.0 authentication. The developers have to register with the Dropbox app console and get the necessary credentials for authenticating the app. While registering in the App console, the user has to set the proper permissions for the job. This permission determines the data that the app can access from the dropbox server.

L. Authorization in Dropbox:

The authorization flow begins with the web page that requests the user to login to their dropbox account and authorize the app. The OAuth 2.0 code flow returns a code via the redirect_uri callback which should then be converted into a bearer token using the oauth2/token call. This is the recommended code flow for apps running on the server.
URL: https://www.dropbox.com/1/2/authorize.
METHOD: GET
PARAMETERS: response_type, client_Id,redirect_uri

M. Integration of Box:

Box is an online content sharing service that enables users to store their content online and mange them online. Box also allows to manage the files on the cloud storage. Thus the core services offered by box are sharing, collaborating and working with files online. It is a secure content sharing service mainly used for business purposes. It offers upto 50GB of storage for personal accounts. Box uses REST API for platform developers with OAuth 2.0 authentication.

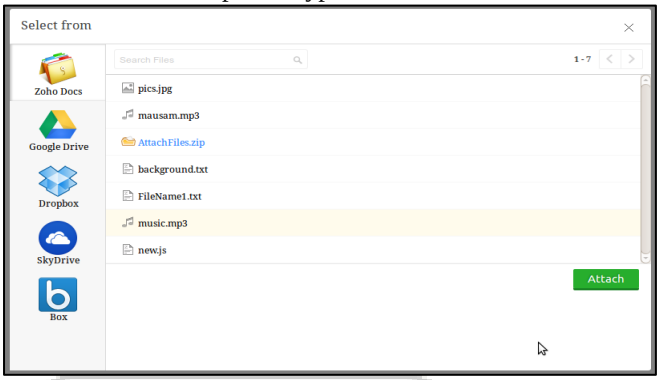
N. Box API:

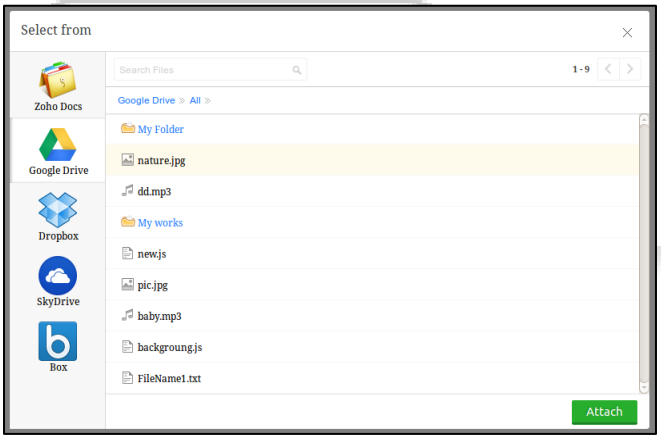
Box [9] offers the developers with the API to access the content stored on the box. It makes use of OAuth 2.0 authentication protocol. The Content APIallows proper management of the content and enables secure access to them. Box as a backend system provides data compliance features.

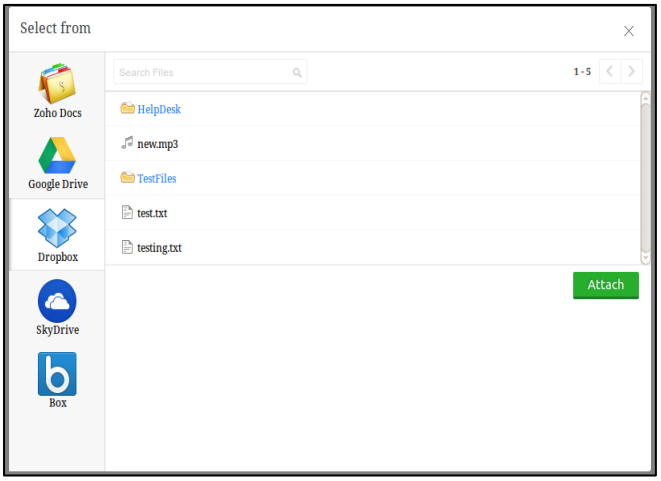
O. Authorization Flow in Box:

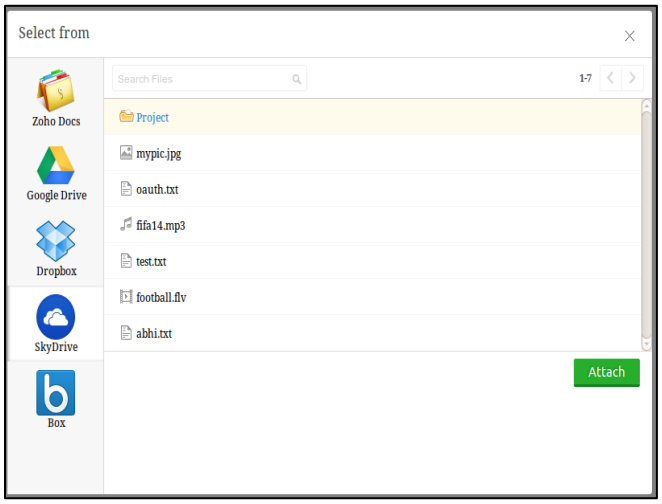
Apps connect to Box using OAuth 2.0 protocol. The authorization flow involves registering the app with the developers.box.com. While registering this app, Box service provides the app developer, with a client_id and a client_secret. The app developer must also specify the redirect_uri. The redirect_uri is the url within your application that receives the OAuth credentials. The scope specifies the limitation over the access to the data stored in the box server.

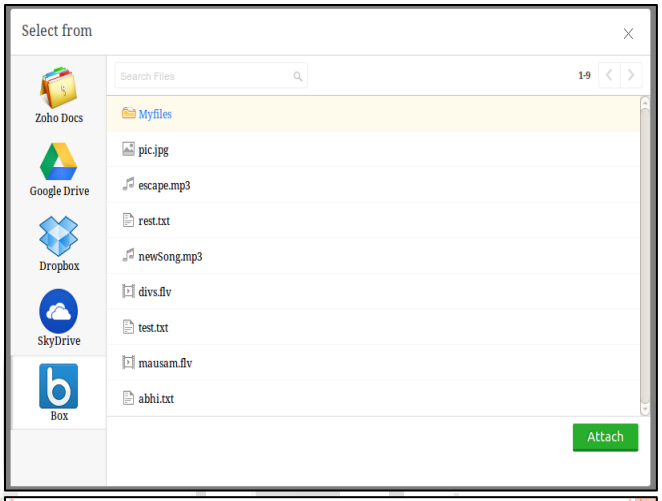
URL: https://www.box.com/api/oauth2/authorize
METHOD: GET or POST
PARAMETERS: response_type, client_id, redirect_uri, state

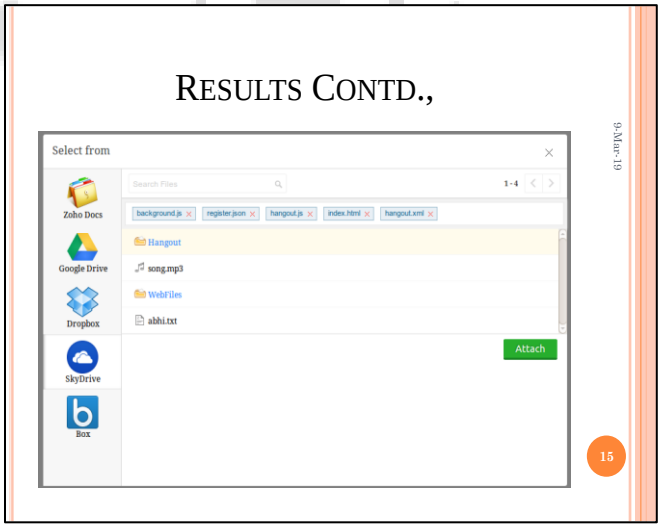












V. CONCLUSION

The modules Cloud Picker and Cloud Uploader makes access and management of files a lot easier. Even though the list of services integrated may seem little, the modules are highly scalable. There is a lot of scope for improvement in terms of features, and quality too. The ease at which the users can authenticate, download, upload, and managing their content, present across various services, makes the module highly efficient, and exhibits a lot of potential for shaping the end-user experience of our products. Learning about the sheer size

of the company, and the amount of hard work that goes into making each module, and thereby defining the experience of the end-user was truly informative.

VI. FUTURE WORK

The project is aimed at developing a dialog that displays the list of integrated online cloud storage and web services. The user can authenticated themselves in each of the services all in a single dialog that pops out. However, making frequent API calls will lead to an increase in the response time. Hence, frequently accessed data can be managed using local storage. At present, only five services are listed in the dialog. In future, various other online cloud and file storage services can also be integrated. There is also a good future scope for open sourcing this module so that any file attachment module can make use of it to enable users to easily upload files from across different cloud services like Google drive, One drive, Box, Dropbox, etc.

REFERENCES

[1] Fielding, Roy Thomas. "Architectural Styles and the Design of Network-based Software Architectures". Doctoral dissertation, University of California, Irvine, 2000.
[2] Chris Borckholder, Andreas Heinzel, Yuriy Kaniovskyi, Siegfried Benkner,Arno Lukas, Bernd Mayer (2013), "A generic, service-based data integration framework applied to linking drugs & clinical trials". Procedia Computer Science, Volume 23, Pages 24-35.
[3] Konstatntinos Evangelidis, Konstantinos Ntouros, Stathis Makridis, Constantine Papatheodorou (2104). "Geospatial services in cloud". Computers and Geosceinces., Volume 63, Pages 116-122.
[4] Ghada ElSheikh, Mustafa y. ElNainay, Saleh ElShehaby, Mohamed S. Abougabal (2013), "SODIM : Service Oriented Data Integration based on Map Reduce ". Alexandria Engineering Journal, Volume 52, Pages 313-318.
[5] https://www.zoho.com/docs
[6] https://www.developers.google.com
[7] https://www.msdn.microsoft.com
[8] https://www.dropbox.com/developers
[9] https://www.box.com
[10] Hardt, D. (Ed).: RFC 6749: The OAuth 2.0 Authorization Framework. Annalen der Physik (2012). Accessed 12 Dec 2016.
[11] Scott, C., Kemp, J., Philpott, R., Maler, E.: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 (2005), http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
[12] Lodderstedt, T., McGloin, M., Hunt, P.: OAuth 2.0 Threat Model and Security Considerations (2013), http://tools.ietf.org/html/rfc6749
[13] Slack, Q., Frostig, R.: Murphi Analysis of OAuth 2.0 Implicit Grant Flow (2011), http://www.stanford.edu/class/cs259/WWW11/
[14] Baidu Inc.: Baidu Open Connect (2014), http://developer.baidu.com/wiki/index.php?title=docs/oauth/authorization
[15] Wangyi Inc.: Wangyi Open Connect (2014), http://reg.163.com/help/help_oauth2.html cite as: Kevin Gibbons, John O'Raw, Kevin Curran (2014) Security Evaluation of the OAuth 2.0 Framework. Information Management and Computer Security, Vol. 22, No. 3, December 2014, ISSN: ISSN: 0968-5227 21. IETF, (2012). Web Authorization Protocol (OAuth) - Char
[16] Hardt, D.: The OAuth 2.0 authorization framework (2012), http://tools.ietf.org/html/rfc6819

