Enhancing Usability of Cloud Storage Clients with Dew Computing

Tushar S Mane.
TASM2M,
Total Automation Solutions,
Pune, India.
tushar.mane@tasind.com

Himanshu Agrawal.
Department of Computer Science,
Symbiosis Institute of Technology,
Pune, India.
himanshu.agrawal@sitpue.edu.in

Gurmeet Singh Gill.
Brand Manager,
Delicious Tiffin Pvt. Ltd.,
Pune, India.
gurmeet@deliciousstiffin.com

***Abstract*— Cloud storage services like Dropbox, Google Drive, Microsoft One Drive have been active and improving unceasingly since 2007. Introduction of desktop/ mobile clients, for example, 'Dropbox Desktop' are the cherry on top, as the users could lever their cloud space from their desktops or mobiles, giving them elegant feel with the file system and storage present on their device. Whatever happens on the device reflects back on cloud space with the corresponding linked account, and vice versa. We can 'partially' say that the computation has been brought down to the ground by such clients. Though this has extremely augmented the user experience, there are still some parts which need to be put in place to complete the picture. This paper highlights some of the missing pieces (offline version management, offline file sharing. and access control list, file URL generation etc.) and their resolutions, which can further improve the usability of the cloud storage clients taking, them towards the completeness. As this is a seamless demonstration of computation happening at device level when there is an intermittent internet connectivity, and then handshaking back with the cloud as it gets connected back to the internet, i.e. Dew Computing, we are referring it as Dewbox. This is one of the many, yet enclosed features of on-device computing, and hence we attempt to encourage researchers to expose possible mechanisms which would utilize the device's potentials to its best. As per the knowledge of authors, this is the first ever attempt till date which traces the missing features of cloud storage clients.**

Keywords—Cloud Computing, Fog Computing, Dew Computing, Cloud Storage, Version Management, Dropbox, Google Drive, One Drive.

I. INTRODUCTION

Storage was the fundamental aspect of Cloud Computing [1] that made it so widespread and beneficial, as it came up with the notions such as Horizontal and Vertical Scaling [2]. Compute services followed the storage services and so do the other services. Latency, security, and privacy are still the foremost concerns of Cloud Computing [3]. To address these issues, Fog Computing [4], which is proximal to devices is presented. Developments in embedded computing [5] have now made devices remarkably powerful. Quantum computing is already knocking the doors [6]. Dew Computing [7] focuses on utilizing device capabilities, especially when those are offline. Dew Computing can be defined as on-device computing, which is not only independent of Cloud Computing but also collaborative with it. Independent means device should work in the absence of internet connectivity, while collaborative meaning whatever happened in absence of internet connectivity is synced appropriately and in order with the cloud as soon as the device gets connected back to the internet. While investigating Storage in Dew (STiD) category [8], we found one of the important missing functionalities, offline version management. Almost 15 GB space is allocated for an unlimited period in free tier. So, students, freelancers, small-scaled or medium scaled organization's developers prefer it as a workspace for their source codes. The purpose is to have cloud as well as a local copy of workspace, so that developers can code online, offline, and at the same time they don't have to worry about version control, which is default feature of cloud storage services. Version management [9] allows one to work freely, without worrying about possible mistakes. One can just switch back to the previous or next version as and when it is necessary. It also offers huge relief on maintenance/ reuse side, one can just restore an applicable version, modify it and get objectives completed. Apart from developers, there are other users (who don't care about versions/ maybe sometimes they do) of storage clients too, who are interested in storing documents, sharing those with any person of choice, on the move. These users can also get benefited by having Dew Computing as an add-on in their storage clients. The sole purpose of the paper is to demonstrate, how missing functionalities of any software client or tool, which are collaborated with the cloud, can be discovered to take them towards completeness, and then unleash the power of devices to achieve the necessary computing i.e. Dew Computing. This is just one of the million possible compute examples to showcase the influence of Dew Computing, and attempt to promote the immense scope in Dew Computing Research.

The paper is structured as follows, second section illuminates Dew Computing to its state of art in a comprehensive manner. The third section enlightens the issues in the current cloud storage clients with the example of Dropbox, while the fourth section proposes the extension to 'Storage in Dew (STiD)' category of Dew Computing to resolve the issues in the current model of cloud storage clients. In the same section, we essentially propose the architecture of Dewbox. Please note implementation guidelines of only the missing features are given. We conclude the paper in the fifth section along with future directions.

II. DEW COMPUTING: STATE OF ART

Current Dew Computing Research revolves around four principal visions by various researchers involved in Dew Computing. However, all forks joining at one common feature, on-device computing which is collaborative with upper layers of computing. Associate Professor, Dr. Yingwei Wang, School of Mathematical and Computational Sciences, University of Prince Edward Island, Canada, states it as a computing residing on the 'on premise' computer, which is independent of cloud in offline mode, while collaborative with the cloud in case of online mode [8]. Whatever happened during offline mode would be synchronized and correlated back with the cloud in the subsequent online

mode. The following diagram (fig. 1) depicts the proposed Cloud-Dew Architecture, wherein any device in the local network will be served by corresponding Dew Server. Devices can still avail minimal set of services or frequently used functions from the Dew Server for unbroken computing.

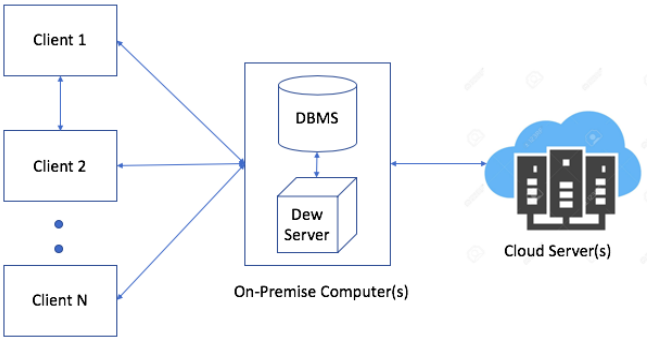

Fig. 1. Cloud- Dew Architecture.

The key objective is to facilitate with the services to users even if there is no internet connection. The use case has been classily demonstrated with the category Web in Dew (WiD) [10], wherein user can still access the website in absence of network connectivity. Key set of minimal functions are still served by 'on premise' server with the help of application, web, and database server running on it. Example, you can browse your Facebook posts and pictures in your spare time even if you don't have an active internet connection. Further, Dr. Yingwei Wang has categorized this generic architecture into several type of services, so that parallel research work can be started for rapid growth of Dew Computing. In this paper we explore Storage in Dew (STiD).

Second research involvement on Dew Computing is directed by Dr. Karolj Skala, Professor at Rudjer Boskovic Institute, Zagreb, Croatia. He proposes Dew Computing to be Context as a Service (CaaS) to offload Cloud Computing [11]. Context as a Service (CaaS) involves processing data at ground, and provide a meaningful context to the cloud. This will surely be a helping hand to the cloud servers. This would scale computing power drastically (fig. 2) and will open doors to solutions to the various computing problems which were considered to be hard to solve till now.

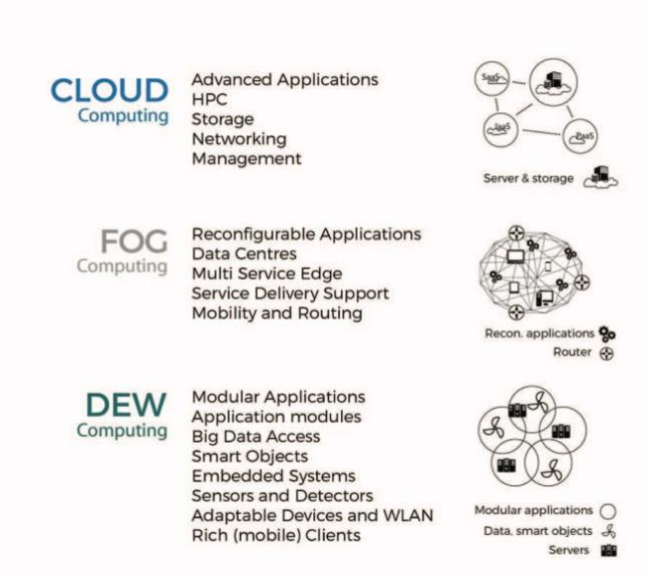

Fig. 2. Scalable Computing Hierarchy [11].

Going forward, the professor has coined the terms Distributed Information Service Environment (DISE), Global Information Processing Environment (GIPE) and Low Power Low Information Processing (LPLIP) as agents of massively distributed and connected physical things.

Dr. Sasko Ristov, Institute of Computer Science, University of Innsbruckalso, has similar vision of utilizing maximum resources at the roots, for information processing, before computation is handed over to the cloud. Researcher proposed 'computation scalability mechanism and its load balancing' in his research work [12].

Recently, Mr. Partha Ray extended Dr. Yingwei Wang's Cloud-Dew Architecture and introduced some terms to support the extended model [13]. He aims to have lightweight 'Dew Server' on client itself, which should serve one client at a time, and store most frequently used functions. In case of data loss, the 'Dew Server' should be able to recover from cloud server from the last checkpoint. Local copy of data should be as small as possible which is referred as 'Dew Site'. The refinement consists of how 'Dew Site' can be modified by 'Dew Client'. 'Dew Script' is a web script file, which will be used for modification of 'Dew Site'. These modifications will be supervised by a 'light weight web come application controller' called 'Dew Analyzer', which will be responsible for maintaining the 'Dew Site' state in respective database(s) present in Database Management System on the 'Dew Server'. The operational details are made clear by 1:1 and 1: N mappings between 'Dew Server' and 'Dew Site' as shown in fig. 3.

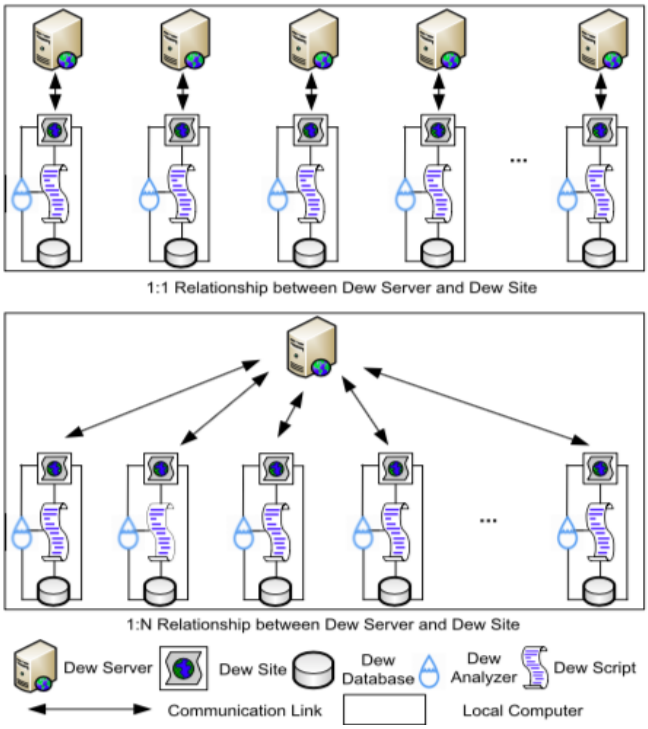

Fig. 3. Mappings Between Dew Server(s) and Dew Site(s) [13].

Last but not the least, in our (me and Dr. Himanshu Agrawal) recent paper [14], which clubs Cloud-Fog-Dew Computing paradigms in to one Service Computing Ecosystem, we shown how Dew Computing can considerably reduce the computational latency. One of the key properties of Fog Computing is its heavy geographical distribution to support the scalability [15]. This geo-graphical distribution comes with the maintenance overload

and hence the possible computing outage. So we put forward a ‘Dew Node Architecture’, which enables ‘Dew Node’ to be a Service Provider or Consumer. Fig. 4 shows architecture of ‘Dew Node’.

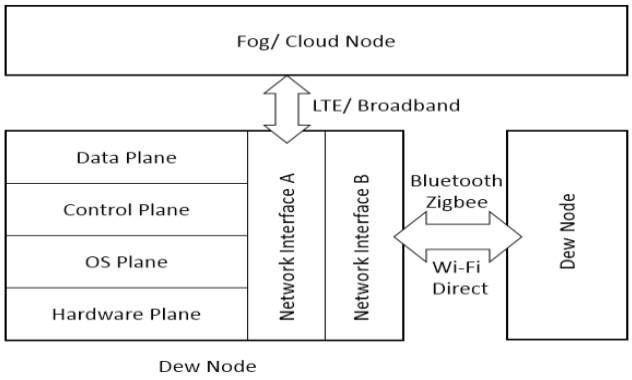

Fig. 4. Dew Node Architecture [14].

So, in case Fog Computing layer fails to provide service to the end devices, one end node, which now becomes ‘Dew Node’ will provides service to other end node i.e. ‘Dew Node’ with contextual intelligence embedded in them.

III. Cloud Storage Clinets: The Missing Elements

As stated before, this paper emphasis on Storage in Dew (STiD) class of Dew Computing. Existing functions of cloud storage clients are well versed, well established, and stable. But, there are some major missing features, which we trace in this section. Please note very carefully, that we have chosen Dropbox for demonstration purpose, as Dropbox is implemented entirely using open source technologies. However, same issue can be reproduced on Google’s backup and Sync (Previously Google Drive Desktop) and Microsoft’s One Drive and any other cloud storage client.

A. Version Management

We illustrate the conflict between local and cloud versions lucidly with the help of small experiment done on Dropbox. Please note the folder structure, text editor and symbols. File is stored in Dropbox folder, the special space created on user’s device after installing Dropbox client. We have used ‘nano’ editor, but any text editor would do. Green tick indicates local copy is in sync with cloud, while cycle symbol indicates, it not synced yet (or syncing under progress).

User creates file ‘DewBox.c’ on his device (in Dropbox folder, as shown in fig.5) and it is synced with its cloud space as there is active internet connection. (Version 1, On-device)

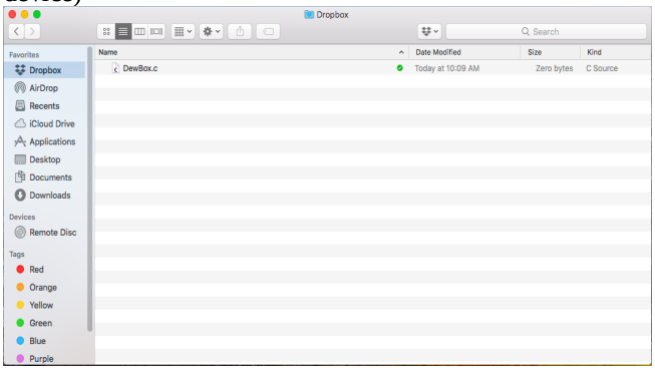

Fig. 5. File is Created (Online, On-device, Version 1).

Instantly, cloning happens on Dropbox cloud with your linked account (Version 1- Cloud), fig. 6.

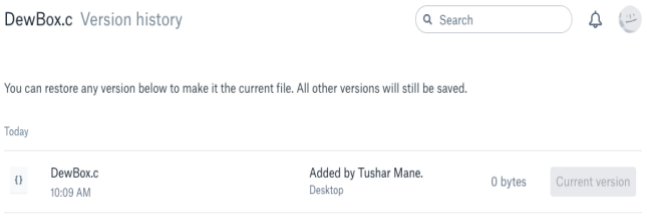

Fig. 6. File Create Operation Clonned (Cloud, Version 1).

Program description, Author info, and Date is added in program while online, file is saved and closed (Meta Data Added- Version 2, On-device), fig. 7.

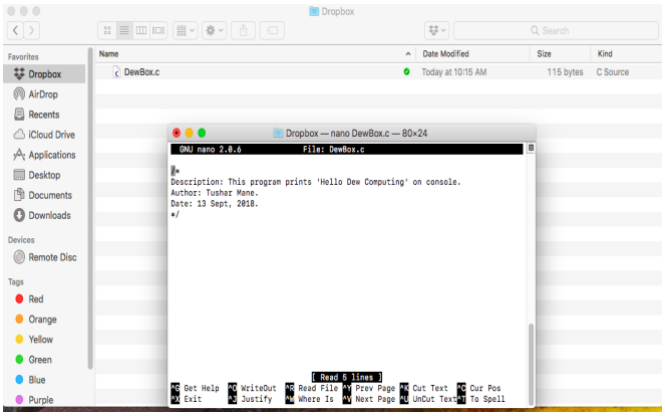

Fig. 7. Meta Data is Added in File (Online, On-device ,Vesrion 2).

With internet connection available, changes are reflected on respective cloud space (Version 2- Cloud)- Fig. 8

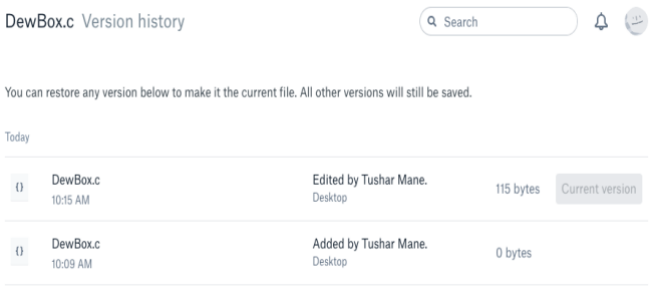

Fig. 8. Meta Data Reflected (Cloud, Version 2).

Now, internet connection is disconnected deliberately and local file is added with some statements and saved, fig. 9. (Version 3, On-device)

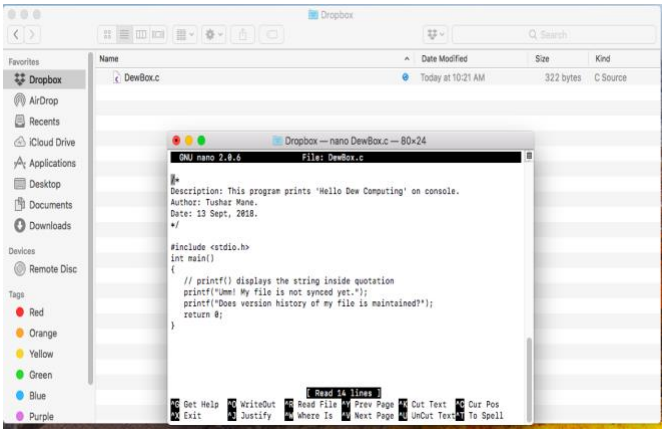

Fig. 9. Statements are Added in File (Offline, On Device, Version 3).

Again, without internet connection, statements in the file are modified and changes are saved, fig. 10. (Version 4, On-device)

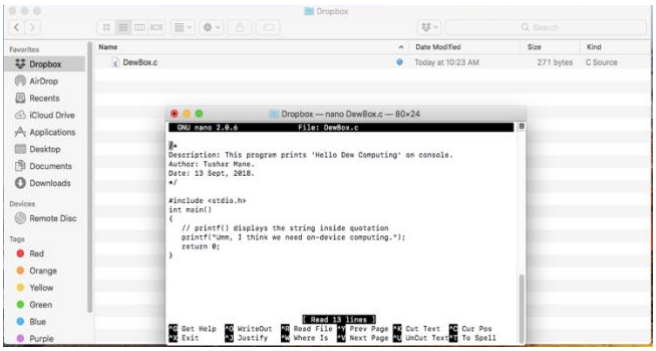

Fig. 10. Some more Modifications in file (Offline, On-device, Version 4).

Now, file is modified again and internet is enabled. (Version 5, On-device)- fig. 11.

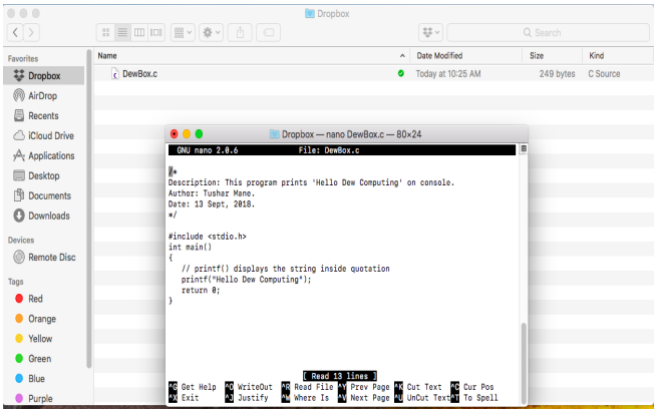

Fig. 11. Some Chnages are made and then Connected to Internet (Vesrion 5).

Immediately, file is synced with cloud and digital twin is created as shown in fig. 12 (Version 5).

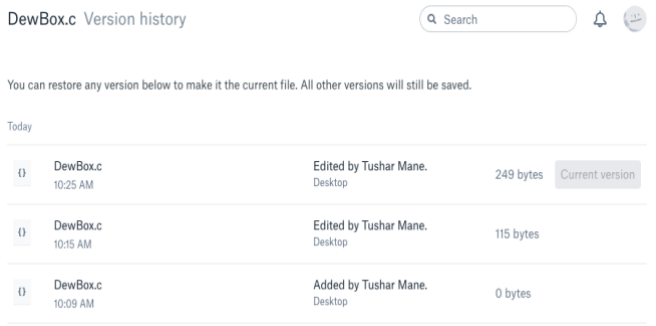

Fig. 12. File gets Synced (Cloud, Version 3).

Here's the serious problem. What about Version-3 and Version-4? If student/ developer/ freelancer relies on 'anytime & anywhere' feature of cloud, and does some work at home (or on another device). Next day he goes to office (or changes the device), and now wants to restore the program back to Version-3 or Version-4, how can it be done? Versions which were created, when he was unknowingly not connected to the internet are lost, right? Now, this is a very small and simple example (just for demonstration). In real world there are dependencies among files/ modules, critical functionalities, a lot of automatic documentation, and many more factors are involved. This is severe missing part of a system. Now let's have a look at possible add-ons.

B. *Enhanced File Operations (File sharing, File Link Generation, Access Control List etc.)*

Roaming people (sales and marketing, business development, site support, to name a few) or even other people/ students, who frequently need to share the documents, keep adding (or removing) people to (or from) shared list (Access Control List) or share a public file URL, might not always have an active internet connectivity, may be due to absence of network, low bandwidth or end of mobile data quota. In such situations most of the people generally tend to forget completing this to-do list. Fig. 13. shows current online operations users can do via client.

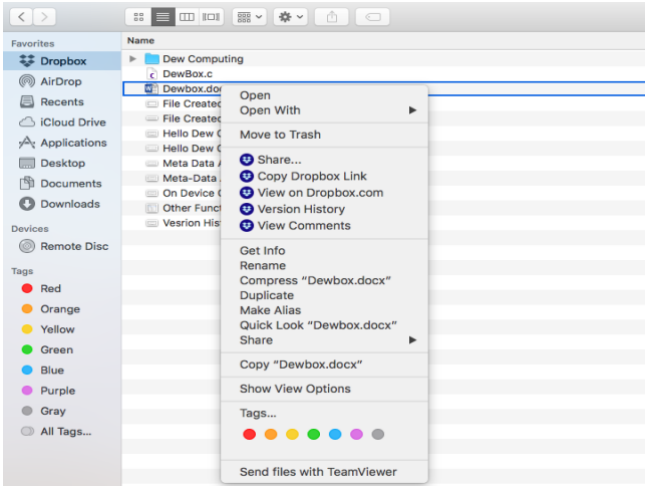

Fig. 13.Current Options in Cloud Storage Clients (e.g. Dropbox)

So, an add-on which enables these pending tasks to be queued in offline mode, so that they can just do these kinds of activities on the move, without having to maintain this to-do list in their mind or a diary, would remarkably increase the usability of cloud storage clients.
Next section explains, Dewbox architecture/ implementation guidelines and how it can overcome above issues.

IV. DEWBOX: TOWARDS COMPLETENESS OF CLOUD STORAGE CLIENTS

As mentioned before, all current features of cloud storage clients are well established, well versed, and stable. So architecture and implementation guidelines of Dewbox only focuses on offline version management, file sharing, URL generation and access control list.

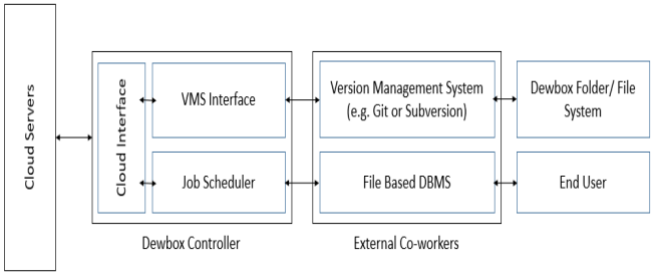

Fig. 14. Proposed Architecture for traced Missing Functionalities/ Add-ons

Architecture, fig. 14, mainly consists of two components, 'Dewbox Controller' and 'External Co-workers'. 'Dewbox controller' contains a component called 'VMS (Version Management System) Interface', which pulls out all the version history of any file from the external version

management system (e.g. Git or Subversion) [16] [17] and shares it with cloud storage server, with the help of 'Cloud Interface', fig. 15, whenever you get connected to the internet. So a fine grained version history of any file or set of files can be maintained. Open source version management systems like Git or Subversion would be installed along with Dewbox, and VMS interface will trigger repository initialization, querying the repository and deletion of the repository automatically in the background, based upon file operations done by user.

Fig. 15. Handshaking between Cloud Interface Module and Cloud Server.

Add-ons like 'Offline File Sharing, Link Generation and Access Control List' are taken care by the second component of 'Dew Controller', the 'Job Scheduler'. 'Job Scheduler' is Simple Queue Data Structure Implementation for which data elements are filled from the file-based Database Management System like SQLite [18]. Whatever operations are scheduled in offline mode by user, are put in a table called 'job_queue'. This table would be queried by 'Job Scheduler' to populate the job queue.

There would be three other tables in database, one for storing contact list of the linked account, other for list of files and their access control, and lastly for storing file URLs and access control. Whenever user says, I want file 'A' to be shared with person 'XYZ' with 'Read-Only' permission, a job will be created by selecting contact 'XYZ' from 'contact_list' table, file 'A' from 'files' table, inserting them in a 'job_queue' table with operation as 'share' and permission as 'Read Only'. 'Job scheduler' reads this table, put the jobs in queue and whenever device gets connected to the internet, it hands over this queue to the cloud server via 'Cloud Interface'. 'Cloud interface' packs these jobs in such a manner that those would be understandable by 'Cloud Server(s)'.

V. Conclusion and Future Directions

Cloud Storage has become prevalent with Cloud Storage Clients, which allows users to do file operations on their devices, irrespective of file system, even in absence of network connectivity. Operations done in offline mode are later get synced with the linked cloud storage. This phenomenon perfectly fit in with definition of Dew Computing, which states, computing which is independent and collaborative with cloud. Hence, Cloud Storage Clients such as Dropbox Desktop, Google Drive Desktop (Now Back up and Sync), Microsoft's One Drive, are categorized under Storage in Dew (STiD) class of Dew Computing. But the critical function of version management, which could lead to some serious issues, was still absent. Also, usability of such clients can also be increased by the offline extensions like File Sharing, URL Generation and Access Control List. In this paper, we attempt to trace these missing features and add-ons. We also suggest an architecture and implementation guidelines for the same. We name it as Dewbox, which would take Cloud Storage Clients towards completeness.

The only purpose behind the paper is to demonstrate how various software or hardware clients or tools, which are connected to the cloud, can be made more independent and collaborative by utilizing resources present on them for increasing their usability. Dew Computing is surveyed in a broad manner to appeal researchers to come forward and contribute in this growing area, which will drastically offload cloud servers and provide seamless computing even in unexpected interruptions in Fog Computing layer.

We highly anticipate implementation of this paper as a foremost future work. We have not focused on security portion of Dewbox, which could mean, opening the Dewbox folder with credentials or securing file-based database system which we have recommended or more. We highly encourage to explore hardware clients like Arduino and other embedded boards for Dew Computing Research as Internet of Things, Quantum Computing are already here and we want Dew Computing to be essential bit of those.

References

[1] Yuhang Yang and Maode Ma, "A Survey of Cloud Computing," Proceedings of the 2nd International Conference on Green Communications and Networks 2012 (GCN 2012), Volume 3. pp 79-84. 2013.

[2] Chien-Yu Liu, Meng-Ru Shie , Yi-Fang Lee, Yu-Chun Lin and Kuan-Chou Lai, "Vertical/Horizontal Resource Scaling Mechanism for Federated Clouds," International Conference on Information Science & Applications (ICISA), 2014.

[3] Danilo Ardagna, "Cloud and Multi-cloud Computing: Current Challenges and Future Applications," IEEE/ACM 7th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems, 2015.

[4] Mohammad Aazam and Eui-Nam Huh, "Fog computing: The Cloud-IoT/IoE middleware paradigm," IEEE Potentials. May/June, 2016.

[5] Jing Huang, Renfa Li, Jiyao An, Derrick Ntalasha, Fan Yang and Keqin Li, "Energy-Efficient Resource Utilization for Heterogeneous Embedded Computing Systems," IEEE Transactions on Computers, 2017, Volume: 66, Issue: 9.

[6] Charles Day, "Quantum Computing Is Exciting and Important--Really!," Computing in Science & Engineering, 2007, Volume: 9, Issue: 2.

[7] Andy Rindos, Yingwei Wang, "Dew Computing: The Complementary Piece of Cloud Computing," IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), 2016.

[8] Yingwei Wang, "Definition and Categorization of Dew Computing," Open Journal of Cloud Computing (OJCC), Volume 3. Issue 1, 2016.

[9] B. Westfechtel, B.P. Munch and R. Conradi, "A layered architecture for uniform version management," IEEE Transactions on Software Engineering, 2001, Volume: 27, Issue: 12.

[10] Yingwei Wang and David Leblanc, "Integrating SaaS and SaaP with Dew Computing," 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), 2016.

[11] Karolj Skala and Davor Davidovic, "Scalable Distributed Computing Hierarchy: Cloud. Fog and Dew Computing," Open Journal of Cloud Computing (OJCC) Volume 2, Issue 1, 2015.

[12] Sasko Ristov, Kiril Cvetkov, and Marjan Gusev, "Implementation of a Horizontal Scalable Balancer for Dew Computing Services," Scalable Computing: Practice and Experience, 2016, Volume 17, Number 2, pp. 79-90.

[13] Partha Pratim Ray , “An Introduction to Dew Computing: Definition, Concept and Implications," IEEE Access, 2018, Volume: 6.

[14] Tushar S Mane and Himanshu Agrawal, “Cloud-fog-dew architecture for refined driving assistance: The complete service computing ecosystem”, IEEE 17th International Conference on Ubiquitous Wireless Broadband (ICUWB), 2017.

[15] Amir Vahid Dastjerdi and Rajkumar Buyya, “Fog Computing: Helping the Internet of Things Realize Its Potential,” IEEE Computer, 2016, Volume: 49, Issue: 8.

[16] https://git-scm.com/

[17] https://subversion.apache.org/

[18] https://www.sqlite.org/index.html