Space Weather Data Management System and Monitoring in Decentralized Storage Environment

Yoga Andrian
Department of Computer Engineering
Keimyung University
Daegu, South Korea
yoga.andrian@lapan.go.id

Essaid Maryam
Department of Computer Engineering
Keimyung University
Daegu, South Korea
maryama.essaid@gmail.com

Kim DaeYong
Department of Computer Engineering
Keimyung University
Daegu, South Korea
imdy@stu.kmu.ac.kr

Soo Hoon Maeng
Department of Computer Engineering
Keimyung University
Daegu, South Korea
tngns2004@stu.kmu.ac.kr

Hongtaek Ju
Department of Computer Engineering
Keimyung University
Daegu, South Korea
juht@kmu.ac.kr

***Abstract*— Indonesian National Institute of Aeronautics and Space (LAPAN) concerns to develop a system that provides actual information and prediction related to space weather activities called Space Weather Information and Forecast Services (SWIFtS). SWIFtS is supported by a data storage system that serves data near real-time. Because the data is collected on one single server and is served by a centralized model, problems emerge when the researchers need the server for data processing and making forecasts to update the content on the SWIFtS website. The system is incapable of providing the latest data when the server is down. Therefore, we propose a new system that utilizes the decentralized model for storing data using the Inter Planetary File System (IPFS). Our proposed method focuses on the background process, and its scheme will increase the data availability and throughput by spreading it into nodes through a peer-to-peer connection. Other unused resources would be useful and no single point of failure. For monitoring, we develop a real-time data flow from each node and information of status nodes. As our expected, performance shown that our proposed system has better throughput than the existing system.**

Keywords*— Decentralized Storage, Data Management, Peer-to-Peer Networks, Distributed File Technology, Automatically Data Storing, Data Monitoring

I. Introduction

In general, space weather term refers to physical processes that occur in the space environment that can affect human activities on earth and space. Represented by the National Institute of Aeronautics and Space (LAPAN; https://lapan.go.id), Indonesia is actively developing a system that providing information and forecast related to space weather activities for ASEAN region called Space Weather Information and Forecast Services (SWIFtS; http://swifts.sains.lapan.go.id). SWIFtS researchers and forecasters conduct daily data analysis and processing to provide accurate and actual information on the SWIFtS website, especially for the users who use the application leveraging space technology (satellite or radio wave technology). For this reason, the data availability from space weather observation instruments is continuously updated to maintain information. At the background process, SWIFtS is supported by the data storage system that collects the data from various instruments installed on observatory stations spread throughout areas in Indonesia. The daily information provided by SWIFtS requires the data storage system to be continuously running well and providing real-time data to fulfill researchers' needs.

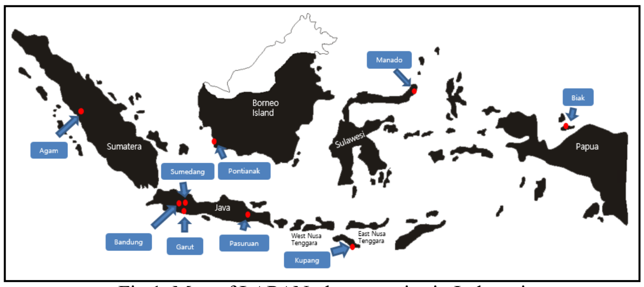


Fig 1. Map of LAPAN observatories in Indonesia

There are eight observatory stations (hereafter referred to as sites) spread in different islands and one central data center in Bandung, Java Island (Figure 1). The existing system implements a client/server hierarchy based on the centralized model. We indicate that the centralized method of the current system has several drawbacks, such as low data availability, low throughput, and increased time for data updates. Currently, the main server in Bandung is the central destination for storing data. A large number of connection to the Bandung server can create a high network load on the server. The current system employs two crontabs (cron tables), which are configuration files that specify shell commands for running a script program on a schedule. All synchronization processes run at one time to synchronize all data through default SSH port utilizing the Rsync tool. Thus, this process creates a queue of jobs that have to be executed while the various data from each site and instrument should be stored safely in real-time and without damage. Low throughput affects the duration of data synchronization due to the high network load; thus, synchronization takes longer.

The decentralized storage system is a method of storing data by encrypting and distributing the data across a decentralized network. It does not rely on a central service provider for data storage [1]. Decentralized storage can increase the percentage of data availability by spreading files to be stored on the hosts that are connected peer-to-peer. Consequently, each host can exchange files directly as well as have roles as a client and server, would reduce user dependency on a single server [2]. The IPFS provides a high data throughput with a content-addressed block storage model, requires no central server, and distributes and stores the data in spread locations [3, 4]. In this paper, a new system is proposed for data management using a decentralized storage model leveraging the IPFS distributed system for storing and sharing the space weather observation data across the peer-to-peer network. The novelty points of this paper are we used the directory watcher method for realtime data update and peers monitoring in the cluster.

Moreover, a technique for monitoring the system through active and passive approaches is developed. The passive approach is intended to measure the flow of data between nodes and the active approach determines the status of each node (e.g., whether online or not). Afterward, system performance is evaluated and analyzed by comparing the performance of the proposed system with that of the existing one. The tested parameters are the mean time of replication of files and the mean throughput from a node.

II. Related Works

This section reviews related works that leverage decentralized network approaches to store the data; however, this approach needs the user as an actor to upload or download the data manually. Most of the papers explained the utilization of the IPFS as their storage system combined with blockchain. However, they did not disclose detailed information on how the IPFS works.

Sia is a simple decentralized storage system proposed by Vorick, et al. [5]. They need clients as users to upload and download files in the Sia network; there is no automated process involved with getting or putting data. Likewise, there are Storj [6] and Maidsafe [7] which also need actors to store and retrieve files manually from the system. Moreover, those systems rely on the blockchain network and have a more commercial focus, needing cryptocurrency to use their service. In 2018, Nygaard [8] leveraged the IPFS-Cluster to limit the replication of data only for peer members. His proposed system also needs the client as actors for storing data manually. However, our proposed system does not need cryptocurrency or tokens because there is no commercial orientation in this system. Moreover, this system also requires no manual process to store the data because various data are produced by the instruments continuously, such as every 5 min, 15 min, hourly, and daily. It would be impossible for a human to conduct this as a manual process.

The building of a scaled-out Network Attached Storage (NAS) and the IPFS to store an object spread through the Fog/Edge infrastructure has been proposed by Confais, et al. [4] and also Brisbane [9], who proposed decentralization for big data by He leverages the IPFS by changing the Hadoop Distributed File System (HDFS) as the file system. However, this system did not provide a detailed explanation of how the IPFS works, stores, and retrieves the data. In this thesis, a detailed explanation is provided of how the IPFS works as a distributed file system using a P2P network.

Indeed, the IPFS provides a web user interface on each node connected to the network. One of the features displayed in all global peerIDs that are connected to the IPFS. However, this system only needs status information from all nodes in the cluster and data flow from each node added to the IPFS. In addition, we need a centralized monitoring system that can determine the current condition of all peers in our cluster that the IPFS does not have today.

III. Proposed System Architecture

All Peers are restricted to a private IPFS network that has been prepared for the process of distributing files between peers automatically. For this reason, another scheme is used to do automatic pinning for every file added to the IPFS. IPFS-Cluster is a tool used to coordinate between IPFS daemons running on different hosts [10]. If one peer disconnected from the network or delete that object, it still provided by others. In addition, once peer distributes an object, it's no more duplication in the network and still be an object with the same hash or called deduplication. Thus, this scenario may not significantly increase the storage space of the nodes since each file with identical content will not be stored.

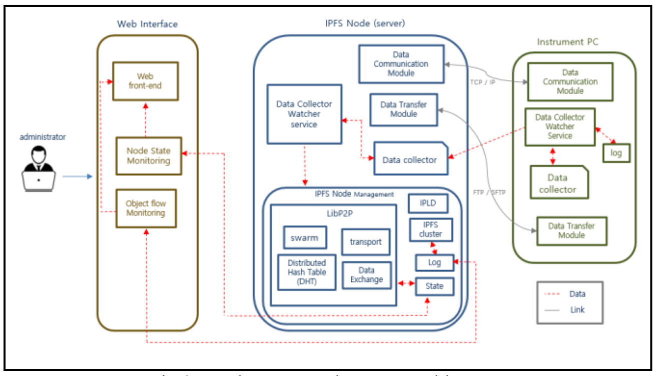


Fig 2. The proposed system architecture

The instrument PCs have two types of operating systems installed, Windows and Unix. Figure 2 shows the architecture of the proposed system. The data communication module between server and PC is using TCP/IP. For the data transfer module, FTP is used for data transfer to the Windows OS, whereas SFTP is used for the PC which has the Unix OS installed. Each of them has a data collector as a place where data is stored for sending the latest data automatically to the data collector on the site server (IPFS node) and then generates a log file containing its metadata. This watcher is built with the watchdog library from python.

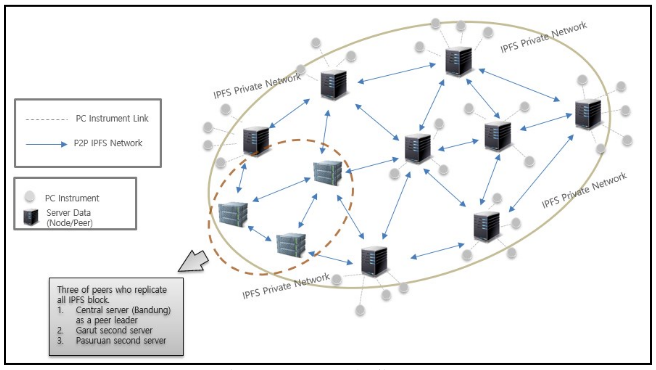


Fig 3. Network diagram

On the IPFS node, a data collector watcher service was built using the *pyInotify* library from python and a client library of the IPFS API. Besides, LibP2P is the major part of our system that requires some mandatory tasks such as allowing for data and communication transport, creation of a distributed hash table, and file exchange in the system. For the data communication module, the link between the PC and the server is connected by TCP/IP. Each server has a node ID and uses a multiaddr-formatted byte string to communicate among nodes in the overlay network. When nodes need to exchange files, a peer-to-peer connection is built between them, so a node can connect to another directly (see Figure 3). Our proposed system does not replicate the file to all peers in consideration of network and storage cost. So, this system only replicates the data to the other three nodes for our default setting similar to the replication factor in the Hadoop Distributed File System (HDFS) [11].

TABLE I. OBSERVATORY STATIONS STATUS IN Q1 OF 2019

No.	**Observatory Stations**	**Total Amount of Daily Data (MB)**	**Bandwidth (Mbps)**	**Internet Connection Link Type**	**Frequent of Server Down (Jan – Apr 2019)**		
					Network Offline (FO Cut)	***Power Outages***	***OS Failures***
1	Garut (Java Island)	21	5	Fiber Optic	0	0	0
2	Pasuruan (Java Island)	39	7	Fiber Optic	4	0	0
3	Biak (Papua Island)	46	5	Wireless	1	30	0
4	Agam (Sumatera Island)	76	4	Wireless	2	0	0
5	Kupang (Nusa Tenggara Island)	106	5	Wireless	No data	No data	No data
6	Pontianak (Borneo Island)	110	7	Fiber Optic	3	4	0
7	Manado (Sulawesi Island)	151	1	Fiber Optic	No data	No data	No data
8	Sumedang (Java Island)	175	4	Fiber Optic	4	0	0

According to table 1, we consider selecting other nodes for storing the data based on the same geographical location with Bandung in Java Island, the nodes which have the lowest total size in collecting data for daily, nodes with higher bandwidth capacity than others and prefer to choose nodes that connected by fiber optic link. So, we select the node in Garut and Pasuruan as the second and third center of data replication. In figure 3, our proposed scheme add the second server in Garut and Pasuruan that has a role only receive the data replication from others together with Bandung. This approach inspired by hosting role in Sia project [5].

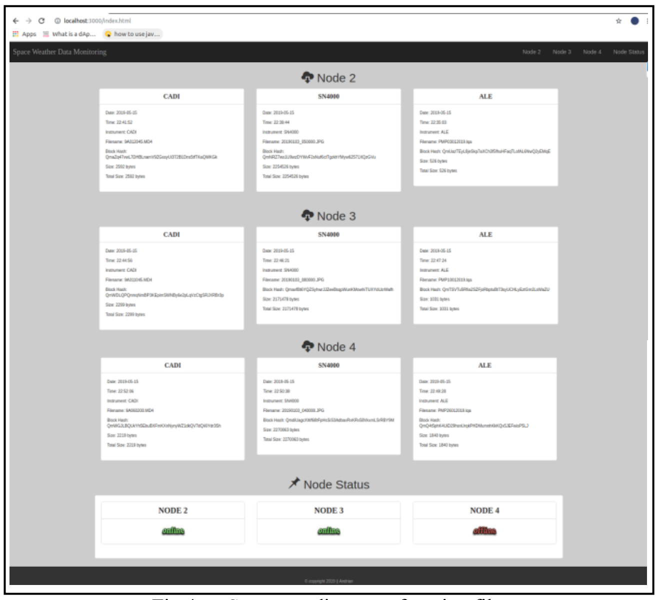


Fig 4. Sequence diagram of storing files

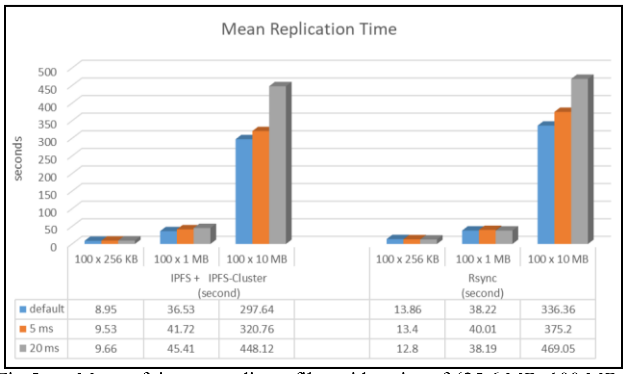
The IPFS relies on the BitSwap protocol to exchange blocks between nodes and a distributed hash table (DHT) to store the pointers of peers who have actual locations of files and node information [3, 12]. In the IPFS network, a file which is uploaded will map into an object with fixed size using a combination of a hash function and base encoding. An object might be several chunks that are called blocks. The SHA256 hash function has been applied by the IPFS and made the size of each block at most 256 KB because the IPFS uses the Rabin fingerprint method for chunking files [13].

A simple web interface has been implemented that is built with the Bootstrap 3.3 framework and JQuery. For data transfer, NodeJS version 10.15.3 was implemented as the HTTP server. Each node has several log files according to its instruments. Therefore, separate JQuery functions are used for reading and extracting the data from those files; one function is responsible only for one file. In Figure 4, we can see this monitoring that contains information on the data that has been successfully stored in the IPFS network and distributed to other nodes. In addition, the status of each node is also displayed to determine the current state of the nodes. The log files automatically update every minute. For data monitoring, logs were sent from all sites to the central server. Meanwhile, for node status monitoring, status information comes from files generated by the central server after finishing pings. In Figure 4, Nodes 2 and 3 succeed in showing their data because their state is online, contrary to Node 4, which has no data information due to its offline status.

IV. PEFORMANCE EVALUATION

This experiment is intended to evaluate the Rsync that has been implemented by the existing system compared to the IPFS coupled with the IPFS-Cluster. The tests were conducted using two PC machines. As for the parameters, the number of files (100) was varied with distinct sizes (256 KB, 1 MB, and 100 MB) and was generated randomly by the /dev/urandom interface for all tests. There were 100 files of each size with five trials for both Rsync and the IPFS + IPFS-Cluster. A total of 9000 datasets were tested for a total of 33,768 GB for its overall size. Moreover, the network latency between computer 1 and Computer 2 was set such at no latency (default), 5 ms and 20 ms latency. Latencies are emulated artificially using the Linux Traffic Control (tc) tool.

Figure 5 presents the replication time of files as a function of latency when all files are only replicated from computer 2 to computer 1. The main observation is that the latency has a greater impact when replicating large files. File size determines the duration of replication; the larger of file size, the longer it will take to replicate the file. The effect of latency also appears to be very significant on the replication time. It can be seen that high latency makes a considerable difference compared to low latency. It takes 297.64 s to replicate files of 1 GB (100 x 10 MB) when the default latency is considered, and 448.12 s when the latency of 20 ms is used for the IPFS+IPFS-Cluster. This happened similarly to Rsync which was 336.36 s at the default latency and 469.05 s when the latency was 20 ms.



	IPFS + IPFS-Cluster (second)			Rsync (second)		
	100 x 256 KB	100 x 1 MB	100 x 10 MB	100 x 256 KB	100 x 1 MB	100 x 10 MB
■ default	8.95	36.53	297.64	13.86	38.22	336.36
■ 5 ms	9.53	41.72	320.76	13.4	40.01	375.2
■ 20 ms	9.66	45.41	448.12	12.8	38.19	469.05

Fig 5. Mean of time to replicate files with a size of {25.6 MB, 100 MB, 1 GB} as a function of latency

Another interesting observation is that file replication by Rsync experiences a slight drop within small and large latencies where it takes 13.86 s and 12.8 s for file sizes of 25.6 MB (100 x 256 KB) and 38.22 s and 38.19 s, respectively, with a size of 100 MB (100 x 1 MB). This might be due to the use of more stable bandwidth or no usage sharing Wi-Fi with other users. On the other hand, file replication by the IPFS + IPFS-Cluster has a rising trend when latency increases. Replication takes 8.95 s to 9.66 s for the file size of 25.6 MB (100 x 256 kB) and 36.53 s to 45.41 s with the file size of 100 MB (100 x 1 MB). Overall, the results show that replication using the IPFS + IPFS-Cluster is faster than Rsync. This is because the IPFS replicates all files using chunks formed by the hash function which have a small and fixed size.

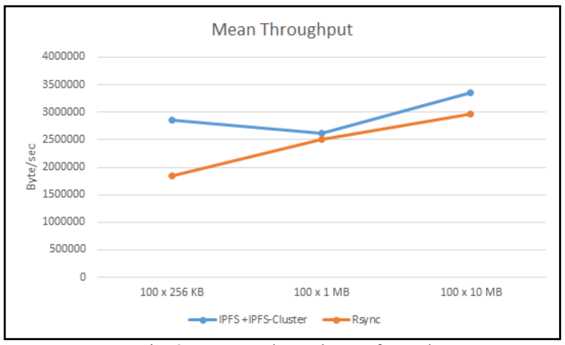


Fig 6. Mean throughput of a node

The mean node throughput is derived by dividing the average size of the replicated file by the transfer time. Figure 7 shows the results that the throughput by the IPFS + IPFS-cluster is greater than that of Rsync for each file size. The main observation is that the file size affects the level of throughput produced. Where the highest throughput generated is 3,359,763 B/s for the IPFS + IPFS-Cluster and 2,973,005 B/s for Rsync for a 1 GB (100 x 10 MB) file size. Meanwhile, a slight decrease in throughput of approximately 243,904 B/s occurs in the IPFS + IPFS-Cluster when replicating files with a size of 25.6 MB (100 x 256 KB) and 100 MB (100 x 1 MB). This is in contrast to the throughput generated by Rsync, which shows an increasing trend of approximately 18% to 35% for all tested files. Afterward, we also compared this data with network bandwidth measurement was supported by the *Iperf* for deriving the average network bandwidth between computer 2 and computer 1. The average bandwidth of the measurement was calculated at 26.4 Mbits/s. If we compare the highest node throughput score of the IPFS + IPFS-cluster meets the average throughput generated by the network with a rating (3,359,763 B/s * 8)/1,000,000 = 26.8 Mbits/s.

Conclusion

In this paper, we proposed a new system to increase data availability for supporting SWIFtS on IPFS node. A directory watcher is added forcing both instrument PCs to synchronize data and nodes to upload data automatically. Moreover, the implementation of the IPFS-Cluster is useful to replicate the data between peers as well as to limit the distribution process only in cluster peer members. Furthermore, a combination technique for system monitoring has been implemented and is seen to be useful in providing real-time data flow and node status. On the evaluation of the finished work, aiming to learn the proposed system performance compared with the existing system, the evaluation has shown that the IPFS-based solution is able to reduce the time of file replication and support high-throughput.

Acknowledgment

This work was supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00539, Development of Blockchain Transaction Monitoring and Analysis Technology).

References

[1] Wang, S., Zhang, Y., & Zhang, Y. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. IEEE Access, 6, 38437-38450. 2018.

[2] Wennergren, O., Vidhall, M., Sörensen, J., & Steinhauer, J. Transparency Analysis of Distributed File Systems: Bachelor Degree Project in Information Technology. University of Skövde, Sweden. 2018.

[3] Benet, J. IPFS-content addressed, versioned, P2P file system. arXiv preprint arXiv:1407.3561. 2014.

[4] Confais, B., Lebre, A., & Parrein, B. An object store service for a Fog/Edge Computing infrastructure based on IPFS and a scale-out NAS. In Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference, pp. 41-50. 2017.

[5] Vorick, D., & Champine, L. "Sia: Simple decentralized storage". 2014. [online]. Available: https://sia. tech/sia.pdf.

[6] Wilkinson, S., Boshevski, T., Brandoff, J., & Buterin, V. "Storj a peer-to-peer cloud storage network". 2014.

[7] Lambert, N., Ma, Q., Irvine, D. "Safecoin: The Decentralized Storage Token". 2015. [online]. Available: https://docs.maidsafe.net/Whitepapers/pdf/Safecoin.pdf.

[8] Nygaard, R. Distributed Storage with Strong Data Integrity based on Blockchain Mechanisms (Master's thesis, University of Stavanger, Norway). 2018.

[9] Brisbane, S. Decentralising Big Data Processing (Bachelor thesis, The University of New South Wales). 2016 .

[10] P. Labs. "IPFS Cluster". 2019. [online]. Available: https://cluster .ipfs.io/.

[11] Borthakur, D. HDFS Architecture Guide. Hadoop Apache Project, 53, 1-13. 2008.[online]. Available: https://docs.huihoo.com/apache/hadoop/1.0.4/hdfs_design.pdf.

[12] P. Labs. "IPLD". 2019. [online]. Available: https://ipld.io.

[13] Kaiser, J., Meister, D., Brinkmann, A., & Effert, S. Design of an exact data deduplication cluster. In 2012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1-12. 2012.