Performance study of Blockstack decentralized storage system: Gaia

Mustafa Alam, Syed Ali Umair Tirmizi, Mesum Raza
Department of Computer Science SBASSE, LUMS

***Abstract*—Blockstacks presents a new decentralized free network based om blockchain that allows values of arbitrary size and allows for a variety of cloud storage back-ends (Amazon S3 and Azure Blob Storage) to be used to store encrypted user data for only storage purposes giving user full control over his data. By treating these cloud silos as dumb drive, cloud providers cannot tamper with the stored data as they can only see the encrypted blocks and associated public keys for decryption are only discover able through the stacks blockchain. Blockstack uses Gaia hub for decentralized storage however little work has been done to look into efficiency of gaiahub across different cloud network storage. In our work, we tested and measure read and write latency across 2 major cloud storage provider (AWS S3 , Azure Blob) and also looked into performance from user's perspective deploying and load testing a boiler plate javascript project offered by Blockstacks. We found our numbers to be higher for larger file size when replicated on AWS with AWS performance slightly better than Azure. The write latency also increase exponentially for encrypted file leading to system crash. Our application level testing revealed the concurrency filter on POST call was making application crash for higher number of clients making updates to same call leading to poorer performance while GET call was faster and comparable to traditional networks. We presents our finding and possible culprits for downgrade performance for larger size file which could be underlying encryption algorithm , difference of hardware at cloud storage end etc.**

***Index Terms*—Blockstack, decentralized storage, Blockchain, cloud storage, latency**

I. Introduction

Blockchain is a promising technology that replaces a trust-based model of carrying out transactions with an electronic system based on cryptographic proof of work that allows two parties to conduct transactions without the need for a trusted third party. Blockchain technology has seen a high adoption rate after the Bitcoin whitepaper was published in 2009. By 2018, 15% financial companies use blockchain and 95% of companies were willing to invest in blockchain. Since 2017, the bitcoin blockchain ledger size has grown from 98.65 GB to 331.07GB and it is anticipated that by 2030 the blockchain ledger size will cross more than 1TB easily [1][2].

Blockchain is an immutable, shared digital ledger technology that keeps track of the transactions in a public or private peer to peer network. The ledger is distributed to all members in the network. The ledger permanently records, in blocks of information the history of asset exchanges that take place between two parties. All these validated and confirmed blocks of information are linked and chained from the very first transaction block to the current block. This makes the blockchain immutable, thus it acts as a single source of truth.

Blockstack uses the bitcoin blockchain. Bitcoin blockchain uses a block size of 1MB to store transactions [3], hence limiting the information that can be written at a time against a block. It is not meant to be used as a general-purpose database [4]. In blockchains, the number of Reads and Writes are very different from conventional databases. There is very little bandwidth and a very high latency cost (of the orders of hours) associated with storing data. That is due to the fact that blockchains have a very different use case, as they are meant to be used as either a secure communication channel or when we need agreement on a highly valuable small piece of data. Hence, blockchains cannot be used to communicate large amounts of data.

To work around this challenge, it is not necessary to store all information related to a transaction in the blockchain, instead supporting files and sensitive data are kept in what is called "off-chain" storage. Off-chain data can either be stored locally on-premises or in a public cloud. The performance of cloud storage systems is also better than peer to peer networks. The challenge in storing data on the public cloud is to ensure the data is authentic i.e. is it the same data that was written by the actual writer? Adding the blockstack peer to peer layer on it helps Blockstacks to achieve the "Decentralized" property that is missing from the existing cloud storage system.

In our work we look at Blockstacks decentralized storage Gaiahubs performance across different cloud storage providers for read and write latency with different encrypted / non-encrypted file sizes. We present key finding with Gaiahub taking larger time for signed file and leading to crashes as encryption increase file size by factor of two . We also found AWS to perform slightly better then Azure while still not able to achieve reported results earlier by Blockstack. We also found a conflict error in concurrent calls for POST by different clients in boiler plate application which is server limitation. We present some optimizations such as using memory efficient encryption and handling concurrency asynchronous at server side.

II. Background

Ever since with the introduction of blockchain and distributed computing more and more research has been done for improving performance, reducing cost and providing a distributed and scalable architecture.

A. *Decentralized Storage*

Storing data away from a central entity to a more localised storage provides more data protection from data leaks and as well as maintaining data localized to the user. With the introduction of blockchain more and more decentralized storage appeared. Protocol Labs, in 2014 introduced IPFS [5], a protocol which was based on peer to peer networking allowing users to connect computing devices with the same system of files for sharing within a single git repository keeping routing in hand of the user. Another decentralized network storage based on Ethereum was Storj [6] which used file-sharding and markel tree to give end-to-end encrypted data transfer without a central point of storage. Storj also provided MetaDisk [7] which was an application allowing user-interface based storage using crypto currency. Like IPFS, SSB [8] was another storage network using Git which provided a social network which replicated messages across users connected to the network including offline updates using git. Similar approach was used in DAT Project [9] which allowed file sharing in the P2P network by assigning a unique id to a folder which can be tracked throughout even with dynamic content inside while also providing access log and history to the creator. Another decentralized storage derived from bitcoin was Sia [10] which leveraged contracting between peers for storage space which was backed by storing entities providing regular updates until the contract expires with storage hosts getting incentive. Blockstack [11] similar introduced another Decentralized Storage called Gaia which used existing cloud providers silos but treating them as dumb drives which store only encrypted/signed data blob not visible to these cloud provider. It uses a discovery layer allowing URI record for data pointing. The paper gives a micro-benchmark of Gaia by showing the latency (in milliseconds) of reading and writing 1M, 10M, and 100M files onto the Amazon S3 as a storage backend. Gaia hub not only provides user's privacy of data , it also allows read/write/delete along with browser capability and guarantees lookup performance with globally addressable data which has a lot of decentralized storage in [5][6][9][10].

B. *Performance of Blockstack*

Performance of reads and writes are evaluated in the paper Blockstack: A Global Naming and Storage System Secured by Blockchains [11]. Evaluating cloud storage benchmarks has been a well-studied topic. There are numerous studies done on popular public clouds like Amazon, Azure, Google, etc. Other work such as [12] has been done to compared cloud storage with respect to QoE . Another research introduces Bitforest [13] that combines centralized yet only partially trusted name servers with efficiently query-able verification data embedded in a novel data structure inside a cryptocurrency blockchain. In [14] the paper experiments in lookup performance against Blockstack with Blockstack lookup growing slower with larger chains due to iterative walk as compared to Bitforest.

While separate work has been done in [15] to judge efficiency of decentralized storage and compare different P2P networks . There is still a need to look into performance figures across different cloud storage platforms for the BlockStack network in depth as cloud platforms can vary across region , hardware and organization size.

III. Blockstack Gaia Storage System

Blockstack provides to its user a decentralized storage system, called Gaia. It has been designed on the philosophy that users do not need to trust any underlying storage cloud providers. Gaia has the capability to provide comparable performance similar to the performance of centralized cloud providers. Users having a blockchain-based decentralized identity can use this storage through their apps and services. Blockstack treats the storage back ends (like Amazon S3, Microsoft Azure) as dumb drives and can store data on them. The data can be stored in the form of un-encrypted, encrypted, and un-encrypted signed or encrypted signed formats. Stored data is constructed in such a way that it must contain the signature from the user's private key. User's zone file contains a URI record which points to the data record stored in Gaia [11]. Reading and writing data involves a separate sequence of operations in the Gaia. Applications can read data to Gaia hub by fetching the zone file and the data. However, writing data involves signing and replicating the data to the storage. Cloud providers cannot tamper with the stored data as they can only see the encrypted blocks and associated public keys for decryption are only discover able through the stacks blockchain.

A. *User control on the data*

The decentralization approach taken by Gaia is to give users the control of their data and storage. In blockstack, users can decide which Gaia hub and which cloud storage will be used to store their data. The choice of selecting a storage backend may be influenced by high performance and high availability for data reads and writes. Blockstack applications can store their data on Gaia on behalf of their users. Applications require the URL of the Gaia hub and the authorization token generated by the user's private key to perform writes [11]. Once an application knows the URL of the Gaia hub, it can use the standard HTTP request to get the file.

IV. Design and Implementation of Gaia Storage System for measuring performance

Our goal is to measure the Read and Write latency in milliseconds of the Gaia storage system with Amazon s3 and Azure Blob storage backends and gather insight into the performance parameters and bottlenecks associated with the gaia storage system. The reference implementation consists of three components i.e. decentralized application (dApp), Gaia-hub and Data Storage backend. Below is an overview of these components:

1) **Blockstack applications** use the Gaia storage system to store data on behalf of a user. When a user logs in to an application, it authenticates and grants the URL of a Gaia-hub, which performs writes on behalf of that user. The Gaia hub authenticates writes to a location

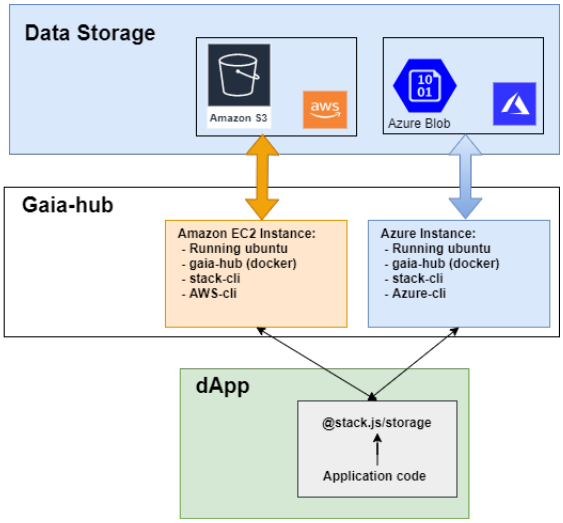

Fig. 1. Gaia storage system architecture diagram

by requiring a valid authentication token, generated by a private key authorized to write at that location.

2) **Gaia-hub** is an interface between decentralized applications and backend storage. It can either be hosted locally, with a public routable address or can be hosted on a virtual machine on a public cloud. We have adopted the more popular deployment approach of hosting the gaia-hub on the cloud as it saves time and efforts related to routing and port-forwarding.

3) **Storage backend** A free tier storage backend of 5GB storage capacity is used in our implementation. The storage backend and the gaia-hub instance are located in the same region to minimize network delays. The data stored in the storage bucket is in the form of objects [16].

Performance testing has been conducted in two scenarios:

1) **Gaia-hub based Testing**
In this approach we use stacks-cli located on the gaia-hub to initiate Read and Write requests along with the measurement framework in place. We have access to gaia-hub's syslog messages to monitor OS related events during test runs.

2) **App-based Testing** In this approach, a sample dApp is used with some code modifications to initiate Read and Write requests along with the measurement framework in place. The application code has been modified to include request and response log timing as well as enable developer tools to monitor network and console logs. Logging of timestamps was added on saveTasks and fetchTasks function on storage.js which is main class that handles requests to gaia-hub using console API times function. We also added a bypass to upload json string resulting in large content-length header of size 1 MB, 10 MB and 20 MB. The todo app was executed on local machine running mac OS with network running 50 Mbps upstream and downstream. The application when executed connects to public gaia-hub hosted by Stacks. Our code for App-based test execution is available on github [19].
We also wanted to observe the application to look for any errors in console as well see user experience in conditions of load and concurrency. For this, which we decided to run the request of GET and POST via k6 open source load testing tool [17] against $c \geq 1$ where $c = clientrunning$.

V. Evaluation

In this section we show the following:

- **Storage performance of Gaia on AWS platform** The latencies reported for 1M file are higher those reported for 10M and 100M are lower than the ones reported in the paper [11].
- **Storage performance of Gaia on Azure platform** The latencies reported for 1M file size are almost comparable to the ones reported for AWS backend and higher than the ones reported in the paper [11]. Similarly the latencies reported for 10M and 100M are higher than the ones reported for AWS backend and lower than the ones reported in the paper
- **Write Latency Trend with increasing filesize** With increase in filesize, the latencies for un-encrypted writes increase gradually. However, for encrypted case the increase in latencies is exponential. For larger file sizes, the test even failed to execute due to an out-of-memory event reported in the syslogs.
- **Challenges and limitation of Gaia** The challenges incurred in both testing scenarios are listed here.

A. Testbed and Methodology

Our testbed consists of a storage bucket and a gaia-hub instance running on AWS and Azure cloud platforms as showing in Figure (1). The gaia-hub and storage buckets are kept in the same region to minimize latencies incurred due to network. Each gaia-hub instance has 1 x CPU core and 1 GB RAM with a non-encrypted disk volume attached. The network connection speed is close to 70Mbit/s. Each gaia-hub runs an nginx server with TLS security. In order to support large files we have made the following changes:

1) "$maxFileUploadSize$" : "1000" was added in the gaia-hub config file to support files that are larger than 20MB (default)
2) $client\_max\_body\_size1000M$; was added in the nginx.conf file.

B. Baseline

The results published in the Blockstack whitepaper [11] are taken as baseline for this study. To results are as follow:

filesizes	Read Latency (msec)		filesizes	Write Latency (msec)	
	un-encrypted	encrypted		un-encrypted	encrypted
1M	374	527	1M	662	1330
10M	841	1726	10M	1850	3289
100M	5956	12886	100M	8938	16852

Fig. 2. Baseline:Read and Write Latencies in milliseconds

C. Results

This section presents the latencies incurred in the **Gaia-hub based Testing scenario** and **App-based Testing scenario**. Ten iterations of each test-case have been executed to get an average time. For **Gaia-hub based Testing scenario** scenario we have observed the following as shown in Figure 3, 4 5. System-level resource usage is also captured during the resource intensive encryption tests in the Appendix.

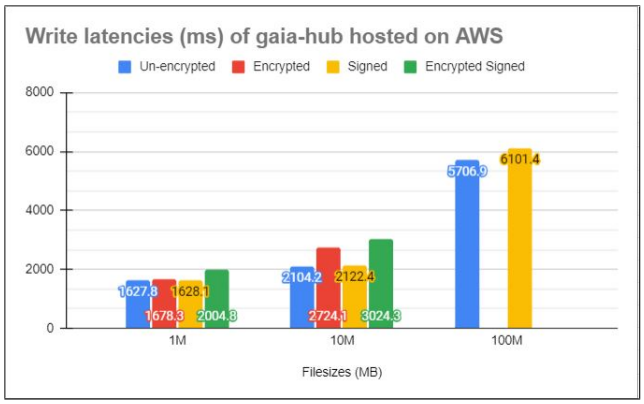

Fig. 3. Write Latencies on AWS platform

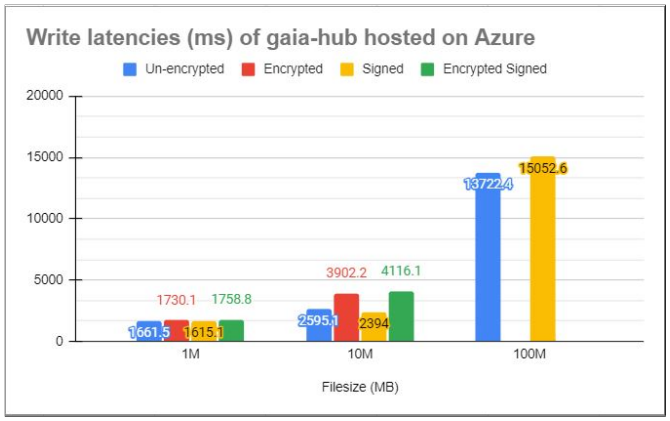

Fig. 4. Write Latencies on Azure platform

For the **App-based Testing scenario** our objective was to record time taken for GET and POST request done by regular application over 10 iterations. To test load and concurrency , we wrote a k6 script with virtual users (vu) with value vu = 1..100 which was executed by k6 collecting http logs each time. We have observed the following results:

- Average GET time on public gaia-hub for 1 MB file was about 440 ms while for 10 MB file it took 980ms and for 20 MB file, the GET call took average 1.42 seconds as shown in Figure 6.
- For POST call the average time for 1 MB was relatively fast at 812 ms while 10 MB and 20 MB file took larger time than expected with 9.12 second and 12.13 seconds respectively as shown in Figure 7. For JSON files that exceeded 20 MB the application received error after 12 seconds of content-length cannot exceed above 20 MB.

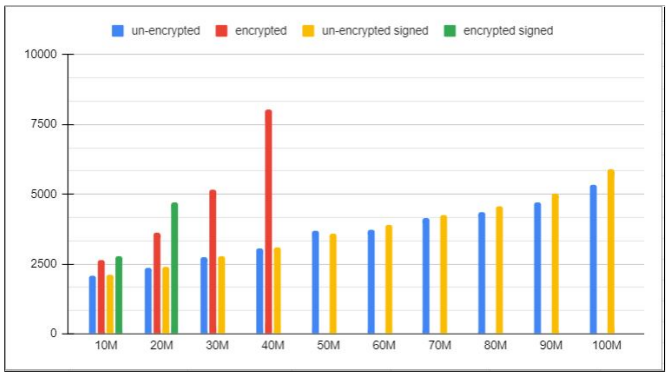

Fig. 5. Write latencies trend with increasing filesizes

To test User-Experience for single user we performed multiple post calls by rapidly inserting values in the App. As the application updates a value on insert we found **409 conflict error** being returned again and again by server causing application to completely crash. According to logs, the object of size 35 was sent to server however the application crash mid-way and would not load. This mean that network requires concurrent calls to be handled at client and concurrent updates needs to handle with retry or delay which raises the question of how an interactive application with frequent updates would behave. This was also applicable with k6 scripts as increasing number of vu to 2 on POST request leaded to conflict of 40% time and upon increase to vu = 10 number percentage reached 90%. The K6 results for a single todo item update for 10 iterations with 10 virtual users shows http_req_failed reached 9/10 times. A Summary of the test run is shown in Figure. 8.

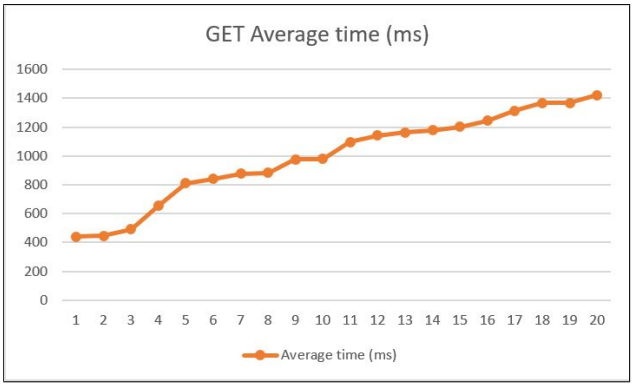

Fig. 6. GET Request Latencies with increasing filesizes

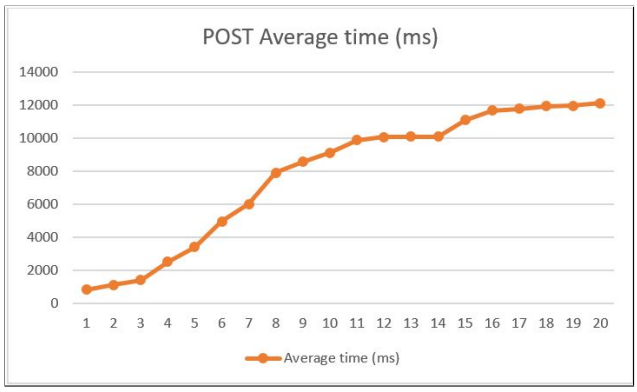

Fig. 7. POST Request Latencies with increasing filesizes

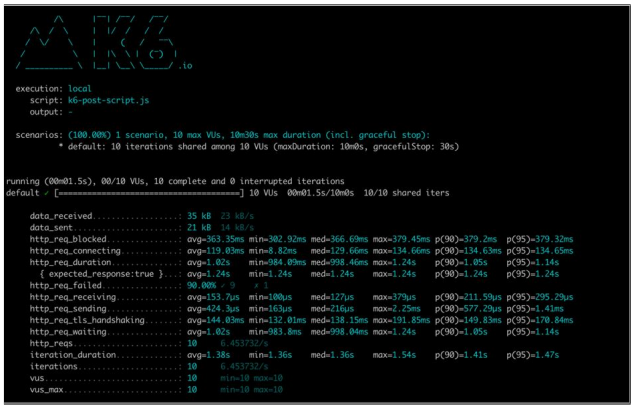

Fig. 8. Testing HTTP request on K6 emulator

D. Challenges and Limitations

Following challenges are observed during the performance test executions: In the **Gaia-hub based Testing scenario**:

1) There was a challenge in executing the **getfile** function from the command line. The **getfile** command line requires several arguments that have not yet been provisioned i.e. Blockstack_ID, Registrar service and a sub-Domain [18]. The documentation is still in Tech Preview status.
2) It has been observed that the maximum file size that is supported by gaia in the encrypted case is 34 MB on both Azure and AWS.
3) A general observation is that the current encryption algorithm used in this study is resource intensive, as the system resources get fully utilized during Write test for files greater than 34M.

In the **App-based Testing scenario**

1) The public gaia-hub has a file upload size limit set to 20 MB, hence it cannot be used to test files greater than 20 MB.
2) Regarding HTTP 409 Conflict error, Gaia hub release notes suggest gaia-hub currently does not support concurrent call for single file. They require the application developer to handle such retry scenarios. This limitation impose challenge on application developers that want to make frequent uploads on gaia storage i.e. live updates where a lot of ”onChange” Javascript functions are invoked.

3) We also observed while GET call response for public Gaia hub was coming directly from Azure Blob storage server (gaia.blockstack.org) while POST request response was coming from Cloudflare server which shows additional throttling might be done on POST call due to performance limitation and concurrency blocker.

Conclusion and Future Work

This independent performance study was first of kind to evaluate performance of decentralized storage offer by blockstack which encompassed different cloud storage provider as well as look at some of aspect such as application level performance and user experience. This performance study has uncovered several performance parameters and bottleneck that are critical for a fully functional Gaia storage system. In order to make it robust their are several streams of investigation that we propose, that might help make Gaia storage system more robust:

1) An alternate encryption algorithm is required that is less resource intensive but still provides the correctness guarantees that are critical to the blockstack design. This will enable encryption of files of larger sizes.
2) A concurrent request handler on the gaia-hub is required in order to cater for frequent requests. Gaiahub can use asynchronous approach and make use of data storages such as buffers to collect concurrent request and reorganize them rather than expecting clients to retry which puts strain on client as well as extra calls to network.
3) Deploy hub and backend storage bucket on other cloud platforms and extend results.
4) Capture read latency for Gaia-hub based testing scenario.

Future work can be done into more cloud platforms as exploring different encryption algorithms for data encryption before storage.

Acknowledgment

We thank our Professors Dr. Ihsan Ayyub Qazi, Dr. Zafar Ayyub Qazi and Dr. Zartash Afzal Uzmi for giving us the opportunity to work on this project.

References

[1] Blockchain Size: Everything You Need to Know — 101 Blockchains - https://101blockchains.com/blockchain-size/
[2] Bitcoin blockchain size 2009-2021 — Statista - https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/

[3] What is the Block Size Limit - https://www.cryptocompare.com/coins/guides/what-is-the-block-size-limit/
[4] What are general misconceptions about blockchain use cases? - https://youtu.be/cXNhwFXwmR8
[5] Benet, Juan. "Ipfs-content addressed, versioned, p2p file system." arXiv preprint arXiv:1407.3561 (2014).
[6] Wilkinson, S., Boshevski, T., Brandoff, J. and Buterin, V., 2014. Storj a peer-to-peer cloud storage network
[7] Wilkinson, S., Boshevski, T., Brandoff, J. and Buterin, V., 2014. Storj a peer-to-peer cloud storage network
[8] Dominic Tarr, Erick Lavoie, Aljoscha Meyer, and Christian Tschudin. 2019. Secure Scuttlebutt: An Identity-Centric Protocol for Subjective and Decentralized Applications. In Proceedings of the 6th ACM Conference on Information-Centric Networking (ICN '19). Association for Computing Machinery, New York, NY, USA, 1–11
[9] Robinson, D.C., Hand, J.A., Madsen, M.B. and McKelvey, K.R., 2018. The Dat Project, an open and decentralized research data tool. Scientific data, 5(1), pp.1-4.
[10] Vorick, D. and Champine, L., 2014. Sia: Simple decentralized storage. Nebulous Inc.
[11] Ali, M., Shea, R., Nelson, J. and Freedman, M.J., 2017. Blockstack: A new decentralized internet. Whitepaper, May
[12] Ali, M., Nelson, J., Shea, R. and Freedman, M.J., 2016. Blockstack: A global naming and storage system secured by blockchains. In 2016 USENIX annual technical conference (USENIXATC 16) (pp. 181-194).
[13] Daher, Z. Hajjdiab, Hassan. (2018). Cloud storage comparative analysis amazon simple storage vs. microsoft azure blob storage. International Journal of Machine Learning and Computing. 8. 85-89. 10.18178/ijmlc.2018.8.1.668.
[14] Dong, Y., Kim, W. and Boutaba, R., 2018, November. Bitforest: a portable and efficient blockchain-based naming system. In 2018 14th International Conference on Network and Service Management (CNSM) (pp. 226-232). IEEE.
[15] Mirko Zichichi, Stefano Ferretti, Gabriele D'Angelo. On the Efficiency of Decentralized File Storage for Personal Information Management Systems
[16] Object storage - https://en.wikipedia.org/wiki/Object_storage
[17] K6 - developer load testing tool - https://k6.io/
[18] [Tech Preview] Using your own Gaia hub with the CLI - https://forum.stacks.org/t/tech-preview-using-your-own-gaia-hub-with-the-cli/6160
[19] Priavte repo for App-Based Performance testing https://github.com/syedtumair/blockstack-lums-project

APPENDIX

Below are System-level monitoring stats i.e. CPU, Memory, Context Switches, Memory Pressure collected from Prometheus node-exporter deployed on gaia-hub:

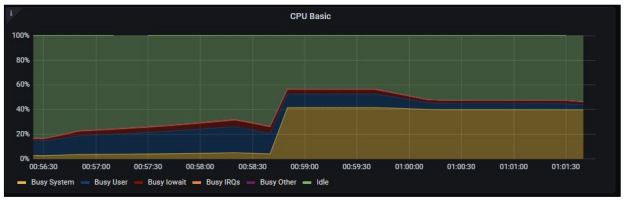

Fig. 9. CPU Usage during encryption test runs for large filesizes

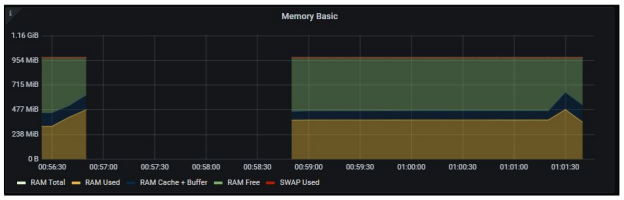

Fig. 10. Memory Usage during encryption tes truns for large filesizes. The area where data is missing shows the machine was hung

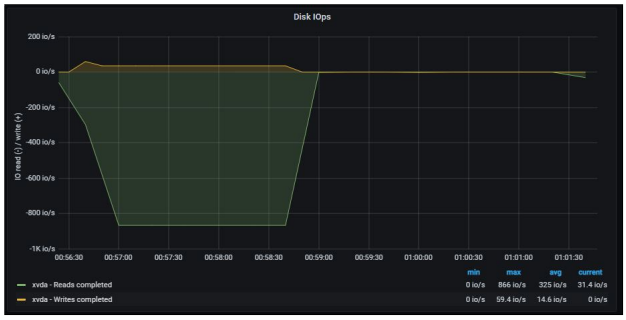

Fig. 11. Disk IOPs during encryption test runs for large filesizes

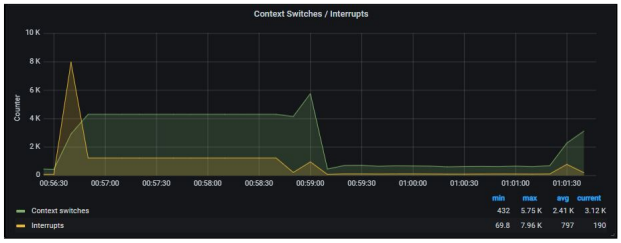

Fig. 12. Context Switches during encryption test runs for large filesizes

Fig. 13. High Memory Page faults during encryption test runs for large filesizes