2017

Blockchains For Publicizing Available Scientific Datasets

Shirishkumar Patel
University of Mississippi

UNIVERSITY OF MISSISSIPPI

BLOCKCHAINS FOR PUBLICIZING AVAILABLE SCIENTIFIC DATASETS

by

Shirish Patel

A thesis submitted in partial fulfillment for the
master's degree

in the Computer Science
Dr. Philip Rhodes
Department of Computer and Information Science

May 2017

ABSTRACT

This thesis explores the effectiveness of blockchain technology for advertisement of Scientific Data. Recently the advancement in hardware and software for data processing increases the supply and demand for huge data sets. Such data may be widely distributed, and not immediately available to the scientists who need it. We need a method of advertising available datasets to interested parties.

Blockchains are a recent innovation developed by the cryptocurrency community, but are increasingly applied to other problem domains. Due to their currency heritage, however, the properties of blockchains do not always lend themselves to new applications.

We have developed a prototype DAPP (Distributed Application) to advertise available scientific datasets using metadata. A private Ethereum blockchain is used to distribute metadata to users, while an Ethereum contract matches dataset providers with consumers. Lastly, a conceptual currency is used to incentivize efficient resource selection.

ACKNOWLEDGEMENT

I would first like to thanks my thesis advisor Dr. Philip Rhodes of the Department of Computer Science at The University of Mississippi. Dr. Rhodes was always there whenever I ran into some trouble spot or whenever I needed suggestions on my research or writing. He guided me in the right directions whenever needed throughout this thesis.

I also would like to thank Dr. Feng Wang and Dr. Byunghyun Jang for being the members of my thesis validation committee and also helping me making my work and my write-up better.

Finally I would like to thank my family for their constant support and faith in me. This accomplishment would not have been possible without them. . .

CONTENTS

ABSTRACT ii

ACKNOWLEDGEMENTS iii

CONTENTS iv

ABBREVIATIONS vi

1. Introduction 1

1.1 The Problem 1

1.2 Motivation 2

1.3 Key Components 3

1.4 Challenges 3

1.5 Summary of Contributions 3

2. Background

2.1 Ethereum 5

2.2 Blockchains. 6

2.3 Solidity 7

2.4 Swarm and Whisper 9

2.4.1 Swarm 9

2.4.2 Whisper. 9

2.5 Related Work 10

2.5.1 Grid Computing. 10

2.5.1.1 Resource Discovery 10

2.5.2 Cloud Computing 11

3. Advertising Resources with a Blockchain

3.1 Mappings 12

3.2 Contract 13

3.2.1 Provider 14

3.2.2 Advertising Datasets 14

3.2.3 Consumer 15

3.2.4 Reputation 15

3.2.5 Example Transaction 16

4. Results
4.1 Testing Environment 20
4.2 Basic Functionality 20
4.3 Performance 24
4.3.1 Publishing 24
4.3.2 Matchmaking 27
5. Conclusion and Future Work
5.1 Conclusion 28
5.2 Future Work 28

Bibliography 29
Vita 32

ABBREVIATIONS

DAPP	**D**istributed **A**pplications
DDOS	**D**istributed **D**enial **O**f **S**ervice
EVM	**E**thereum **V**irtual **M**achine
Class**A**ds	**C**lassified **A**dvertisements
VM	**V**irtual **M**achine

CHAPTER 1: INTRODUCTION

“The blockchain is an incorruptible digital ledger of economic transactions that can be programmed to record not just financial transactions but virtually everything of value."

Don & Alex Tapscott, authors of Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World [1]

The “3 V's" (Volume, Variety and Velocity) are three defining properties of “Big Data" [2]. While Volume represents the amount of data, Velocity represents the speed of data processing. Variety refers to the diversity of data types used in different fields. Recent advancements in hardware and software have caused a data explosion. This resulted in a rapid increase in the amount of produced and published data and hence even further worsened the problems presented by the 3 V's. As these challenges increased, processing and managing datasets became more important.

In distributed environments, networked computers can communicate with each other by passing messages. Nodes in a distributed environment need a way to advertise the availability and costs of the resources in order to access the resources on the network [3]. The conventional way to do this is to have one master who controls and publishes data to all other nodes in the network, or to have peer to peer network in which all peers share information with each other.

1.1 The Problem

The Data explosion introduces new problems of managing huge datasets in an effective way. These datasets need to be processed in order to extract human understanding from the raw data. Even though one might have enough processing power to process these datasets, there remains a problem to find appropriate dataset for particular application. Datasets may be widely distributed across a network and have many replicas available, also different applications need datasets that are specially formatted or preprocessed. Satisfying these

needs underlines the necessity for a common platform which advertises available datasets in terms of metadata. Such metadata helps data consumers to make better choices from among the available replicas. Indeed, research into resource management and replica selection has been an ongoing topic of research for decades [4]. This thesis investigates whether the arrival of an exciting new technology, the blockchain is a viable approach to this important problem [5].

1.2 The Motivation

The notoriety of Blockchain Technology is largely due to the rise of BitCoin, as a solution for the problems of “double spending" and similar attacks that plaqued previous attempts at digital currencies. All the processing on a blockchain is recorded as a transaction which shows the proof of transaction processing. Once the transaction is completed it is stored in permanently in the blockchain. Blockchains are (generally speaking) immutable, using cryptography to prevent tampering with the contents of blocks. Nodes participating in the blockchain each have a complete copy of the entire blockchain, all the way back to the beginning of the chain. Newly joined nodes must sync the full blockchain content in order to start making transactions. As each node in the blockchain technology has a full history, it is almost impossible to have DDOS attack because there is no single point of failure.

In addition to this resilience, the fact that each node has a complete history makes blockchains well suited to the task of advertising available resources. Rather than taking a traditional peer-to-peer approach [6] that requires repeating historical information for new nodes, blockchains handle this problem naturally, since each node has a full copy of the blockchain history.

Even though the main application domain for blockchains remains digital currencies, it had been applied to different domains like Internet of Things, networking, publishing medical records, online library system etc. also in current year some of the multinational companies are making their own version of blockchains for private use.

The problem of advertising metadata about datasets or resources in distributed environments have been studied for years [7]. However, these e orts largely predate the invention of the blockchain. A major motivation of this thesis is to investigate the feasibility of using a blockchain for replica selection in a distributed context.

1.3 Key Components

We chose the Ethereum Blockchain to implement the prototype, which is an open source, distributed blockchain based computing platform featuring smart contracts: software running on a blockchain over the network. Smart contracts are account holding objects on the Ethereum blockchain, and are capable of interacting with other contracts, storing data, and making decisions. Solidity is the programming language with which smart contracts are written. A JavaScript API officially published by Ethereum developers is used to communicate with Ethereum node.

1.4 Challenges

Blockchain technology is new and fast evolving but right now it still has some limitations. For example it doesn't allow users to choose which data to be replicated on each node and which data to be stored in distributed manner, which makes direct representation of datasets unattractive and limits to advertise only metadata as an advertisement.

Blockchain technology was originally designed for managing digital currency, and inherits properties designed for this purpose. It therefore raises complexities when being applied to different domains. For advertising metadata about the scientific datasets using blockchain technology, we had to deal with multiple layers of technology inside the blockchain itself to make it work. As blockchains are new technology, certain conveniences (such as floating point types and a library of ready-made data structures) are simply not yet available.

1.5 Summery of Contributions

- In Chapter 2 we've given basic introduction on Ethereum and Blockchain technology. Also Chapter 2 includes brief information on Swarm and Whisper protocols, which are complementary technologies to Blockchain.
- Chapter 3 talks about the work we've done. It mainly focuses on the detailed explanation of the DAPP (Distributed Application) we've developed. Also it includes run through transition from an instance of a system, which will help you better understand how the DAPP works.
- Chapter 4 discusses about the tests: why they are needed and what the results are.

- Chapter 5 includes conclusion of our work and future work possible in this area.

CHAPTER 2: BACKGROUND

Bitcoin is a cryptocurrency and a payment system invented by an unidentified programmer, or group of programmers, under the name of Satoshi Nakamoto. Bitcoin uses peer-to-peer technology with no central authority like banks i.e. transactions take place between users directly. Bitcoin is open-source; nobody owns it or controls it and anyone can take part. Ethereum uses the same concept as Bitcoin but offers more functionalities to its user.

2.1 Ethereum

Ethereum is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third party interference. These apps run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property. This enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a futures contract) and many other things that have not been invented yet, all without a middle man or counterparty risk. Ethereum pro-vides an Ethereum Virtual Machine (EVM) which can execute Turing-complete scripts using an international network of public nodes and a token called ether. Ethereum was initially proposed in late 2013 by Vitalik Buterin, a cryptocurrency researcher and programmer. Development was funded by an online crowd sale during July August 2014. To complete transaction one must pay price in terms of Gas (an internal pricing mechanism for Ethereum) which also helps prevent spamming the network. Ethereum and Bitcoin differs in many terms but the most important distinction to note is that Bitcoin offers one particular application of blockchain technology, a peer-to-peer electronic cash system that enables online Bitcoin payment. While the bitcoin blockchain is used to track the ownership of digital currency (bitcoins), the Ethereum blockchain focuses on running the programming code of any decentralized application.

2.2 Blockchains

A Blockchain is a distributed database that maintains a continually growing list of ordered records called blocks, designed to hold the records of the transactions that have taken place. Each block contains a timestamp and a link to previous block. A genesis block is the starting block of blockchain which defines the characteristics of the resulting blockchain. Blockchain uses a peer-to-peer network and a distributed timestamp server to maintain database autonomous. Blockchains are by default made resistant to the modification of data; i.e. once recorded, data in the block cannot be altered. If needed the current state of a contract can be updated in blockchain, but this updated data will not replace the old data. Instead a new branch will be made for this updated data and a reference to this branched data will be given at the old location of data i.e. old data will remain there as long as blockchain exists.

Blockchains provide secure online transactions and store these transactions across many computers in such a way that makes alteration of these records almost impossible. Transactions are authenticated by mass collaboration powered by collective self-interests a.k.a. miners, which allows participants to access and verify the transactions in an inexpensive manner. For a trans-action to be stored into a block, has to be authenticated by miners. Miners play important role in the security of blockchain as it won't allow spam blocks to be added to blockchain.

A Blockchain stores the data across its network, which eliminates the problems that comes with storing data in a single place like single point of failure. A Blockchain uses a public key (a long string of hexadecimal numbers) as an address on the blockchain for encrypting data on the blockchain. Each user will have own private key using which it can access the contents of blockchain. As whole content of blockchain will be replicated on each node by blockchain technology and each user will have their own private key, each user will have full access to the content of blockchain. With Each node having full copy of blockchain it is safe to say that there is no "official" copy exist or no user is important than other user.

Blockchains are databases used for storing data, but to populate those blockchains uses different software, Ethereum and BitCoin are examples. For serializing changes blockchain uses different types of time-stamping schemes like proof-of-work, proof-of-stake etc. Blockchain technology was originally developed for keeping track of transactions performed by BitCoin but as time passed and the evolution of blockchain took place, managing

transactions for cryptocurrencies like BitCoin became merely one of the many possible applications of blockchain technology.

Proof-of-Work time stamping technology is currently used by BitCoin, meaning certain amount of work has to be done in order to complete the transaction. This results in making process of mining in BitCoin centralized because of the most of the companies ended up purchasing hardware dedicated for mining process. Ethereum in other hand uses the Proof-of-stake time stamping technology, which focuses on validating a block instead of mining it. Means blocks still need to be created by someone, and who gets to create the next block depends on the specific Proof of Stake algorithm, but the selection process have randomness in contrary with of Proof-of-Work algorithm used by BitCoin.

2.3 Solidity

As mentioned earlier the main difference between BitCoin and Ethereum is that BitCoin uses a blockchain for recording and maintaining online transactions where as Ethereum uses blockchain to run smart contract all over the network. Smart Contracts are small programs which governs the behavior of accounts within the Ethereum state. These programs operate within the context of the Ethereum environment. Solidity is the programming language for writing smart contracts. It is perhaps the first example of a contract-oriented programming language; a slight tweak on the notion of object-oriented programming language. While closely related to object-oriented languages, this is a language designed specifically to help express agreements that must encode ideas and relationships relevant to Real Life, or some formal model thereof. As such we see notions such as ownership, identity, protections and restrictions forming a core part of the vocabulary and idiomatic grammar.

Example of HelloWorld.sol a simple contract:

```
contract mortal  {
    /* define variable owner of the type address */ address owner;
    /* this function is executed at initialization and sets the owner of the contract */
    function mortal () { owner = msg . sender ; }
    /*    Function to recover the funds on  the contract */
    function kill ()  {  if  ( msg . sender == owner )  selfdestruct ( owner );  }
}

Contract greeter is mortal {
    /* define variable greeting of the type string */
    string greeting ;
```

```
    /* this runs when the contract is executed */
    function greeter ( string _ greeting ) public {
        greeting  =  _ gr e et in g ;
    }
    /*  main function */
    function greet () constant returns ( string ) {
    return greeting ;
    }
}
```

Explanation:

As you can see there are two different contracts in the code: greeter and mortal. Solidity provides inheritance just like other languages; in the code above the greeter contract inherits the properties of mortal contract by "is mortal". You can declare global variables and structures and can access those using functions. Once you deploy a contract on blockchain, the same bytecode will be replicated on each instance of the blockchain giving access to every node with the full blockchain copy. There is no central identity who manages the code, meaning each node will have equal priority by default but you can give the owner of contract or some other node a little more privileges than other nodes as in the code above only the owner i.e. the creator of the contract will be able to call the function kill(). Upon deploying the contract as a return value you will get an address (string of hexadecimals) at which contract is deployed. In order to access the same contract from other nodes, one will need this address and a ABI (application binary interface). The ABI defines different things such as how parameters are passed to the functions, how application interacts with itself, where the return value is placed for return etc.

Once the contract is deployed on the blockchain then it is supposed to run for infinite time as long as there is even a single node active on the network. But if you want the contract to be inaccessible and retrieve all the resources contract is holding you can do it so by the use of kill() function as shown above in the code. The data of the contract will be still there in the blockchain as the data in the blockchain is supposed to be immutable but it won't be accessible. After deploying the above contract on blockchain you first need to set value of variable greeting of type string by calling function greeter using the contract object created using the deployed address and ABI of the contract. After setting the value of greeter if you call function greet using the same contract object, you will get the value you have settled before for greeting.

2.4 Swarm and Whisper

Both Swarm and Whisper are complementary technologies contributing to the vision of Ethereum as \World Computer". Imagining Ethereum as a metaphor for shared computer, it should be noted that a computation alone is not enough. For a computer to be fully useful, it also needs storage to remember things and bandwidth for communication. Keeping in mind the vision of Ethereum World Computer smart contracts can be seen as distributed logic, Swarm can be seen as decentralized storage and Whisper can be seen as decentralized messaging.

2.4.1 Swarm

Swarm is a distributed storage platform and content distribution service, a native base layer service of the ethereum web 3 stack. Swarm is being designed as an accounting protocol that benefits from the self-execution of "smart contracts" on Ethereum Virtual Machine. The primary objective of Swarm is to provide a sufficiently decentralized and redundant store of Ethereum’s public record, in particular to store and distribute app code and data as well as block chain data.

From the end user's perspective, Swarm is not that different from WWW, except that uploads are not to a specific server. The objective is to peer-to-peer storage and serving solution that is DDOS-resistant, zero-downtime, fault-tolerant and censorship-resistant as well as self-sustaining due to a built-in incentive system which uses peer to peer accounting and allows trading resources for payment. Swarm is designed to deeply integrate with the devp2p multiprotocol network layer of Ethereum as well as with the Ethereum blockchain for domain name resolution, service payments and content availability insurance.

2.4.2 Whisper

Whisper is a pure identity based messaging protocol for Ethereum which allows messaging between using within the same network. Whisper is separate from Ethereum so smart contracts have not (yet) access to it. Whisper is not designed to provide a connection oriented system, nor for simply delivering data between a pair of particular network endpoints. Whisper is a new protocol designed expressly for a new paradigm of application development. It is

designed from the ground up for easy and efficient multi-casting and broadcasting, and also for low-level partially-asynchronous communications.

Whisper is not designed to transfer huge amount of data between nodes, but only to exchange messages of information between nodes. Like whenever DApps needs to publish small amount of information to each other, DApps that needs to signal each other in order to ultimately collaborate on transaction or whenever DApps needs to provide non-real-time hinting or general communication between each other. Whisper works on very low level, API is only exposed to DApps and never to the end users.

2.5 Related Work

Distributed Computing where the resources are distributed over network and are managed, used, owned by organizations and the fact that partners can join or leave organization at any time makes it dynamic. Resource discovery in such dynamic network is tedious and very important [8].

2.5.1 Grid Computing

A simple way to think about Grid Computing is that it is the way of taking advantage of unused CPU-cycles of a pool of interconnected computers. According to Ian Foster, “Grid Computing refers to the large-scale integration of computer systems (via high-speed networks) to provide on-demand access to data-crunching capabilities and functions not available to one individual or group of machines"[9]. Grid Computing removes the need to “provision" for peak load. Instead, organizations share resources with other organizations, borrowing from other organizations during their own peak periods and vice versa.

Condor is a software system developed by Miron Livny's group at the University of Wisconsin and made available to the public in 2003[10]. Condor is a specialized batch processing system that manages the jobs for Grid Computing system. Condor places the job in queue until the resources required for that job are available and then starts the job on available grid machine.

2.5.1.1 Resource Discovery

As mentioned above one of the tasks of Condor is to schedule jobs to available Grid Machine, as the efficiency of a scheduling algorithm is bounded by the resource discovery algorithm makes the problem of resource discovery in Grid Computing very important.

Classified Advertisements (ClassAds) are the lingua franca of Condor. They are used for describing jobs, workstation and other resources. They are exchanged by Condor processes to schedule jobs. They are logged les for statistical analysis and debugging purpose. They are used to inquire about the current state of problem.

Match making is a resource management software for High Throughput Computing. Most of the problems while deploying Condor in Grid are handled by this [11]. As the machines being located at different physical locations, selection of a right replica of dataset from other replicas becomes an important issue [12].

2.5.2 Cloud Computing

Unlike Grid Computing where, more than one computer coordinates to solve a problem together and process will have direct access to the resources required, Cloud Computing is where application doesn't have direct access to the resource required. Rather it accesses them through something like a service. So instead of talking to specific hard drive to store the data or CPU for computation it talks to the service that provides these resources. Another difference between Grid Computing and Cloud Computing is that in Cloud Computing virtualization of resources is possible of which end user is unaware. Virtualization makes little difference to the customer experience, but they are only charged for the time they actually use. If they aren't using any VMs, they are not charged. By giving access as VM's comes elasticity, means the cloud provider can run only the number of physical machines that are necessary at a given time.

In theory, our model could support either grid or cloud computing. Grid Computing complements the vision of Ethereum as a World Computer. Ethereum allows creating "tokens" that functions as currencies. One way our system could support cloud computing is via a token 1that can be bought with “real world" cash.

CHAPTER 3: PUBLICIZING RESOURCES WITH BLOCKCHAIN

Data in the blockchain is complete, consistent, timely and widely available. Due to the decentralized network, blockchains do not have single point of failure and that makes them better able to withstand malicious attacks. As there is no central authority to control the transactions users are responsible for all of their information and transactions. Changes to the blockchains are viewable by all parties, creating transparency between users. All the data in the blockchain are by default immutable, meaning they cannot be altered or changed. The fact that having a distributed database all over network minimize the risk of single point of failure also leads to have each node having a full copy of blockchain which increases traffic because of message passing system of blockchain and memory occupied by each to hold blockchain data. Immutability of blockchain provides integrity but also makes it di cult to handle dynamic data.

Blockchain works as underlying infrastructure for Ethereum allowing smart contracts to run on it, solidity is used for developing smart contracts. Once deployed on blockchain smart contracts are called DAPPS (Distributed Applications) and are supposed to run for infinite time. Solidity offers many functionalities for developing smart contracts.

3.1 Mappings

Mappings can be seen as Hashtable which are virtually initialized such that every possible key exists and is mapped to a value byte whose byte-representation is all zeros: a type's default value. It consists of two main parts: a Key-Part and a Value-Part. Syntax for declaring a mapping is:

```
mapping ( _KeyType => _ValueType ) mapName ;
```

One can think of KeyValue as the key that has to pass to the function to get the ValueType associated with it. Here the KeyValue can be any type except a mapping, a dynamic size array, an enum or a struct where as a ValueType can be anything also another mapping which

is called mapping of mapping. Mappings are only allowed for state variables (or as storage reference type in internal functions).

Unlike traditional hashtables in key data is not stored in mapping, only its keccak256 hash to look up value. Because of this mappings do not have a length or concept of key or value being set. Also this makes mapping non iterable.

An example of mapping:

```
struct sellerAccounts {
        string sellerName;
        address sellerAddress;
        uint accountBalance;
        bool doesExist ;
        int sellerReputation ;
        uint numberOfFeeds;
    }

mapping ( address => sellerAccounts) list_Accounts;
```

As you can see in the code above there is structure sellerAccounts with multiple members. Here the KeyValue is of type address and ValueType i.e. return type is of type sellerAccount. Here values using mapping can be set as:

```
list _ Accounts [0 x 9 c 7 c a a 8 4 c f 6 2 e a 7 3 9 8 8 7 d 3 3 8 3 a c b 8 d 8 0 c 5 c 2 2 d 2 e ]. sellerName = " xyz ";
list_Accounts [0 x 9 c 7 c a a 8 4 c f 6 2 e a 7 3 9 8 8 7 d 3 3 8 3 a c b 8 d 8 0 c 5 c 2 2 d 2 e ].accountBalance = 0;
list_Accounts [0 x 9 c 7 c a a 8 4 c f 6 2 e a 7 3 9 8 8 7 d 3 3 8 3 a c b 8 d 8 0 c 5 c 2 2 d 2 e ].doeExist = false ;
```

3.2 Contract

We developed a smart contract that advertises the available datasets to the scientists who might use them. Customers can query a dataset type and will get list of sellers who provides that particular type of dataset then it's up to customer to choose appropriate dataset from the list. Whenever a seller uploads any new advertisement he/she will get some points as a reward (token) and can use it later on for buying (or accessing!!) resources on the same network. Also buyers can give feedbacks on the datasets that will help other buyers making decision and choosing the right dataset from list.

3.2.1 Provider

Whenever a provider has a dataset he/she wants to sell. He/She must create an account, upon creation of account successfully a token will be associated to that particular address and will be used as a reference in all further transactions. Once token is assigned with account, seller can upload the advertisement about dataset. According the credentials specified by seller for dataset, seller will get points in his/her account that can be used later for accessing other resources. Each token will have a sellerName, sellerAddress, accountBalance which are visible to all members and sellerReputation which is not visible to anyone but helps improving the system. SellerReputation will be directly affected by the feed backs from the datasets that were uploaded by that particular account and in turn will affect the feed backs given by this seller. We can say the older system gets the more stable it becomes. Once advertisement is successfully added to the contract seller will be rewarded with points in accountBalance.

3.2.2 Advertising Datasets

Advertisement have information specific to particular dataset. Solidity offers structure type for having different data types. Keeping in mind the further upgrades on solidity we decided to store information about dataset into two different structures datasetNode info and datasetNode qos contains the basic information about the dataset and quality of service respectively. Structure datasetNode info have variables:

IP: IP address of the server where the data is stored

datasetLocation: Geological location where the dataset is being stored

datasetDescription: Free form metadata

encryptedURL: An URL that will be given in return when buyer buys the dataset. Could be the url to access the dataset

cost: Cost to buy the dataset

doesExist: Use to prevent duplication of datasets and also can be used when dataset is in maintenance and seller temporary wants to make it unavailable

volume: Volume of the dataset (in Kb)

owner: Owner of the dataset (account address of the dataset owner)

All the information in the datasetNode info will be visible to the buyer on query. datasetNode qos have only a single variable that wil be visible to the user:

datasetFeeds: An integer showing the feedback received for the dataset

3.2.3 Consumer

For accessing the datasets consumers can either directly go to the address of Owner and dataset type (if he/she already knows exactly what dataset he/she needs) or can get the list of providers who have that type of datasets and can decide what dataset is perfect for the application. For accessing any dataset the buyer must have enough points in his/her account Balance. Once consumer accesses the dataset he/she can give feedback about the dataset. Feedback given by a particular customer will be modulated by the personal reputation of that consumer and then only will be added to that dataset. We can say that in order to access any resource on the network one first should share something on the network.

3.2.4 Reputation

Reputation of an account holder plays an important part in maintaining the integrity of system. Function giveFeedBack takes sellerReputations as a reference to count the final feedback given to the dataset. Even though the feedback of the owners who provide good datasets count as more important than those who doesn't, we made sure that one individual's feedback doesn't affect dataset's feedback more than a certain limit.

Each account holder at a time can be in either of the following category. Consider the code below:

```
Function calcMul () returns  ( int ){
        if ( list_Accounts [ msg.sender ].numberOfFeeds >10){
            if ( list_Accounts[ msg .sender ].sellerReputation == 5){ return 1;
            }
            else if ( list_Accounts[ msg .sender ]. sellerReputation == 10){ return 2;
            }
            else if ( list_Accounts [ msg .sender ]. sellerReputation == -5){ return -1;
            }
```

```
            else if ( list_Accounts [ msg.sender ]. sellerReputation == -10){ return -2;
            }
            else {
                return      0;
            }
        }
        else {
            return      0;
        }
    }
```

Here msg.sender is the address of the node originating transaction i.e. in this case buyer, who is giving feedback. list_Accounts[msg.sender].numberOfFeeds is sum of all feedbacks received by the datasets uploaded by the feedback giver i.e. msg.sender with this case. It is important to check since only after getting certain number of feedbacks, feedbacks given by buyer should affect more or less or else should be considered as neutral. As you can see in the code instead of relating sellerReputation directly with the return value we used it for categorizing owner that bounds the effect of feedbacks given by an individual buyer. You can tune the final effect on feedbacks by individual buyer by changing return value here. As this return value will directly affects the final feedbacks.

3.2.5 Example Transaction

Following is a state of account with an address: "0x003cc32e6f95748ebf31809a2803dcfda2fe7526" at some time. Let's give this account an alias Owner1.

	Owner1
sellerName :	University of Mississippi
sellerAddress :	0x003cc32e6f95748ebf31809a2803dcfda2fe7526
accountBalance :	1550
numberOfFeeds :	26
ownerReputations :	-6

Interpretation of the information in the table above can be done as: Owner1 owns multiple datasets (no need to know exact number of datasets owned). Right now Owner1 has 1550 points which it got from uploading datasets that it can use for accessing other resources. For all the datasets owned by Owner1 it received a total of 26 feedbacks (this includes positive and negative feedbacks). According to the feedbacks received to its datasets Owner1's reputation is -6, we will see how it affects the feedback given by Owner1. numberOfFeeds and ownerReputations are not visible to any user in the system but are used for maintaining the QoS of system.

Another account with address "20df3558eee9fbc4a3bfc8ddb82fd8b63c4ef49e" is created with initial credentials and no datasets are uploaded from this account yet, let's give this account an alias Owner2. At current time the state of Owner2 will be as follows:

	Owner2
sellerName :	Mississippi State University
sellerAddress :	20df3558eee9fbc4a3bfc8ddb82fd8b63c4ef49e
accountBalance :	0
numberOfFeeds :	0
ownerReputations :	0

Owner2 uploads dataset with the following credentials, let's give this dataset an alias dataset2:

	Dataset2
IP :	130.74.96.151
datasetLocation :	Starkville, MS
datasetDescription:	Some string
encryptedURL :	www.urltothedataset.com

datasetCost :	700
datasetVolume :	1024KB
datasetOwner :	20df3558eee9fbc4a3bfc8ddb82fd8b63c4ef49e
datasetFeedBack :	0

It is important to note here is that encryptedURL will not be visible to any buyer when it gets the list of sellers, it will be returned only when buyer buys the dataset and after points are credited into seller's accountBalance. As Owner2 uploads dataset advertisements its state changes but this does not affect the state of Owner1, now the states of two accounts Owner1 and Owner2 will be as follows:

	Owner1	Owner2
sellerName :	University of Mississippi	Mississippi State University
sellerAddress :	0x003cc32e6f95748ebf31809a2803dcfda2fe7526	0x003cc32e6f95748ebf31809a2803dcfda2fe7526
accountBalance :	1550	1536
numberOfFeeds :	26	0
ownerReputations :	-6	0

Here rewards will be credited to accountBalance which is directly dependent on volume of the dataset but the cost of dataset do not depend upon any other variables but is totally up to the account holder. Owner1 wants to access the Dataset2 owned by Owner2, accountBalance of Owner1 which is 1550, is greater than the cost of dataset which is 700 so Owner1 can access Dataset2. If everything is alright and after points transferred from Owner1's accountBalance to Owner2's accountBalance, encryptedURL will be returned to buyer, from where buyer can access the dataset. After buying Dataset2 states of Owner1 and Owner2 will be as follows:

	Owner1	Owner2
sellerName :	University of Mississippi	Mississippi State University
sellerAddress :	0x003cc32e6f95748ebf31809a2803dcfda2fe7526	0x003cc32e6f95748ebf31809a2803dcfda2fe7526
accountBalance :	850	2236
numberOfFeeds :	26	0
ownerReputations :	-6	0

After accessing Dataset2 Owner1 can give feedback for Dataset2 but as we know Owner1 has -6 ownerReputations, meaning Owner1's datasets have not received very good feedbacks. This means Owner1's feedbacks should not be affecting more, but it should affect a little. So according to logic for whatever feedback Owner1 gives, less than one will be implied to dataset means if Owner1 gives positive 3 feedback to Dataset2 only positive 2 will be added to Dataset2. After Owner1 gives feedbacks for Dataset2 states of Owner1, Owner2 and Dataset2 will be as follows:

	Owner1	Owner2
sellerName :	University of Mississippi	Mississippi State University
sellerAddress :	0x003cc32e6f95748ebf31809a2803dcfda2fe7526	0x003cc32e6f95748ebf31809a2803dcfda2fe7526
accountBalance :	850	2236
numberOfFeeds :	26	1
ownerReputations :	-6	1

Here datasetFeedBack will be visible to all buyers who query datasetList:

	Dataset2
IP :	130.74.96.151
datasetLocation :	Oxford, MS
datasetDescription:	Some string
encryptedURL :	www.urltothedataset.com
datasetCost :	700
datasetVolume :	1024KB
datasetOwner :	20df3558eee9fbc4a3bfc8ddb82fd8b63c4ef49e
datasetFeedBack :	2

At current time if Owner2 wants to give feedback to other datasets, feedback will be given to dataset as they are provided by Owner2 as right now Owner2's dataset only have received 1 feed from other users. We cannot determine whether Owner2 is increasing Qos or decreasing QoS of system.

CHAPTER 4: RESULTS

Ethereum is already successfully running over the world on huge number of nodes, and was designed to be scalable. So it would be redundant for us to evaluate scalability here. Also Ethereum's whisper protocol for exchanging data between nodes makes it much efficient in terms of propagating transactions from one node to another. The main purpose of this tests is to check the functionalities of the DAPP we've developed and make sure it is functioning as it should be.

The domain of Ethereum is to manage cryptocurrency, so with ethereum comes data integrity, data security, data availability and other advantages like resistance to DOS attacks, fast propagation of data between nodes etc., making ethereum a suitable choice for our application.

4.1 Testing Environment

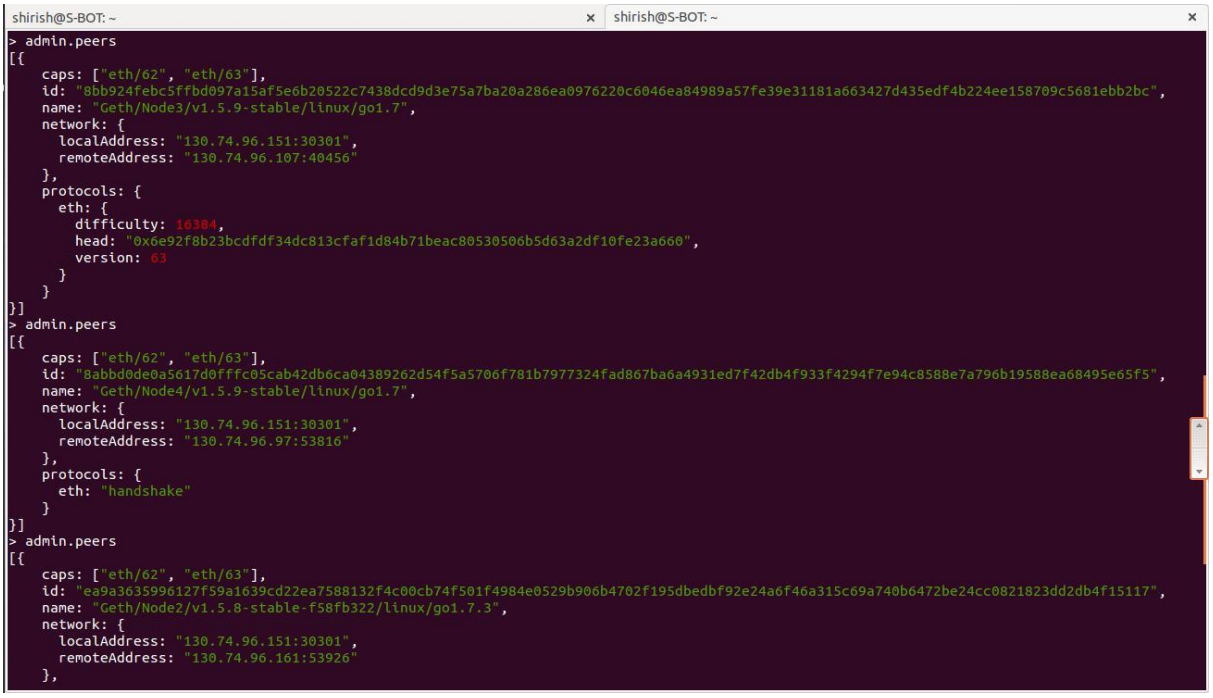
As scalability is not an issue for Ethereum, instead of going for large number of nodes we have tested our DApp on 6 nodes placed within 2miles of radius.

For this testing we used three nodes with intel i7 3.40 GHz quad core processors with each having 16 GB of RAM, two nodes with intel Xenon 2.40 GHz quad core processors with each node having 24GB of ram, one node with AMD A8 quad core processor with 6GB of RAM. All these nodes were connected wireless and were located in 1000meters of range as latency because of distance is negligible for our application.

4.2 Basic Functionality

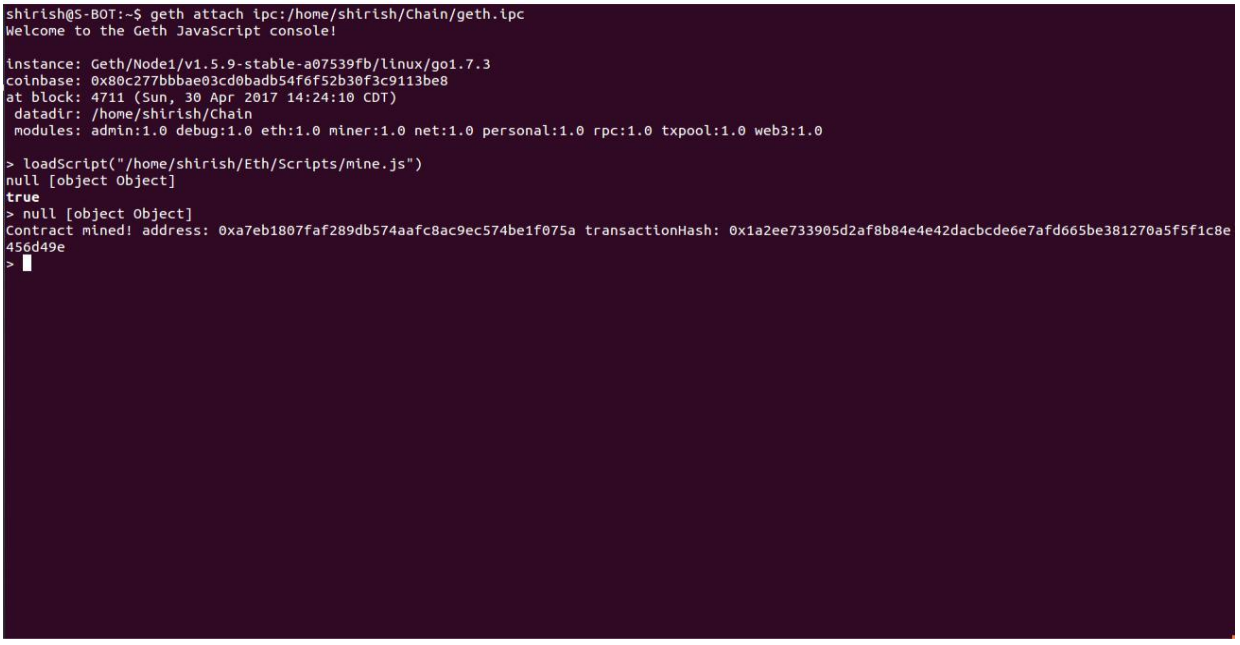
This section will guide you through the basic functionalities Following are the screen shots from actual testing phase at particular state of system, which will give you an idea how the system works:

First of all all the nodes were initialized with the same genesis block and are connected with each other making a cluster of nodes. Once cluster is up and running you can check it from any node as shown in image below:



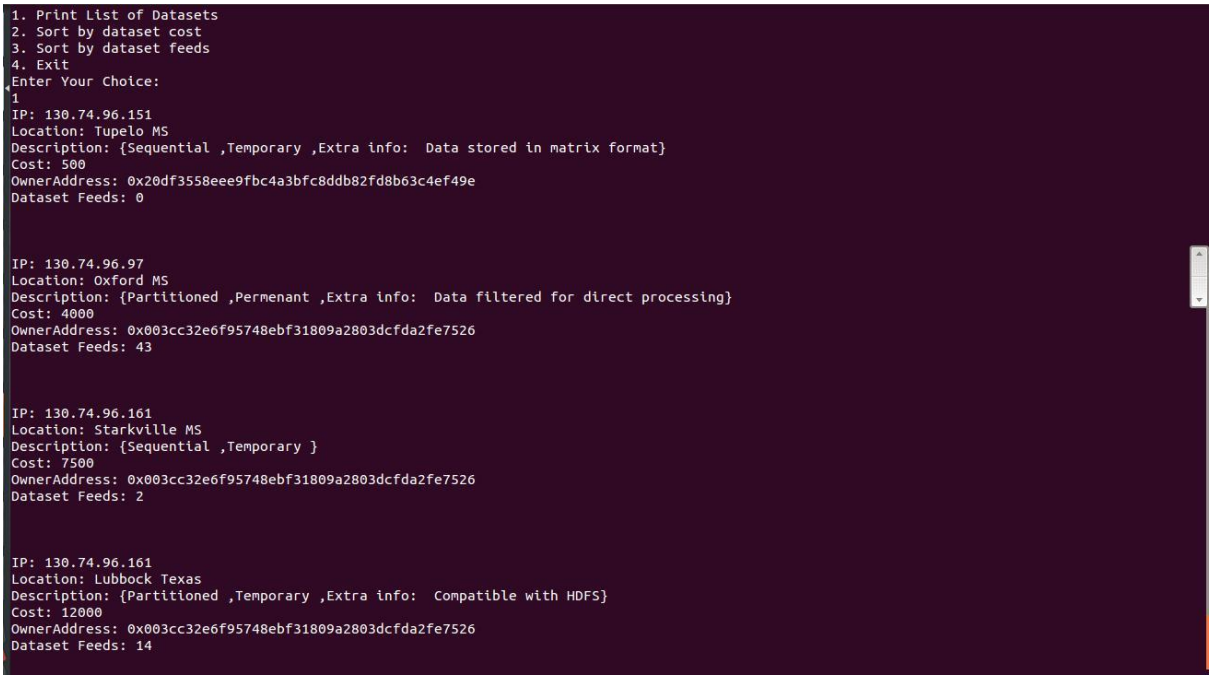
After cluster is ready and each node it connected, contract can be deployed on any on of the node. In the following screen shot you can see contract being deployed on any one of the machine

---



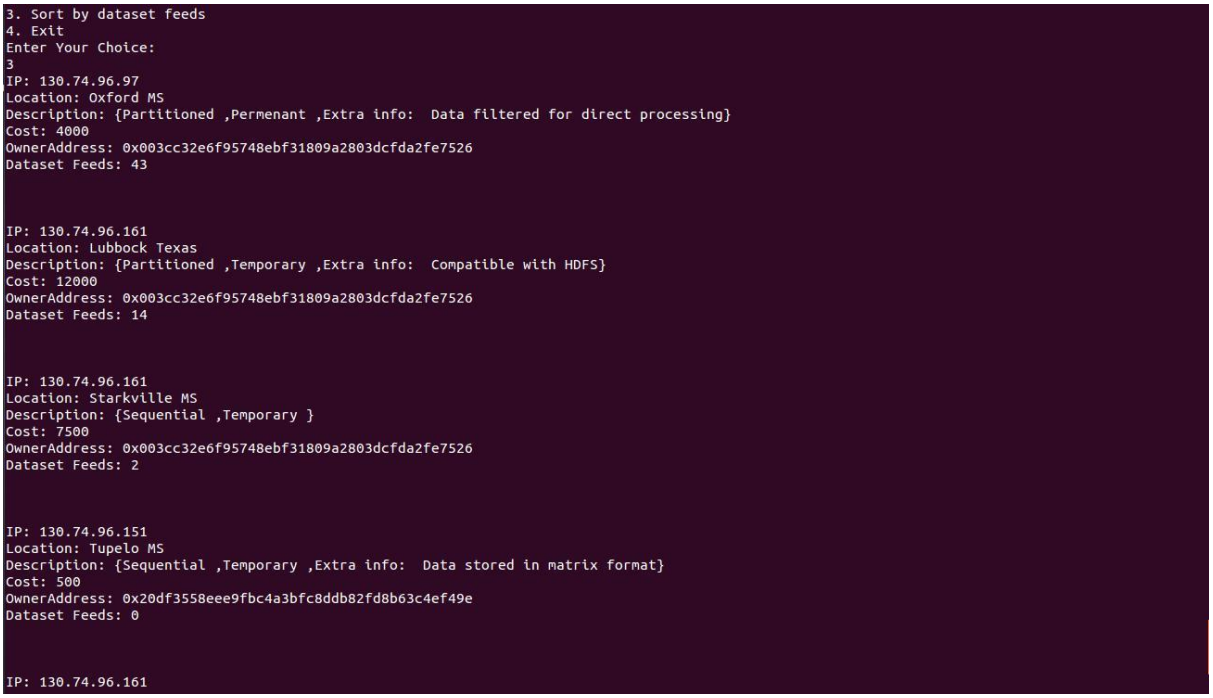
After the contract is deployed on blockchain after some time (exact timing in upcoming sections) you can retrieve data from any other node in cluster. You can see this in the screen shot below. Thing to be noted here is that data was uploaded from other node and was retrieved from other node.

Now let's say after some time have passed and users have uploaded datasets and accessed datasets that changes the states of datasets as well as the reputations of the Owner of dataset(Section 3.2.5). Now if any user wants to get the list of all the producers who owns the the datasets of mobile data, The user can run a script and will get list of all producers in a le where user can process it, or it can be provided as an input to any application. Here in out example we have used this le as an input to a java program for further processing. Outputs of a Java program are presented in screen shots below:



We've processed data from a le and make Objects of each datasets so that we can do more with the data. Some of the examples are sorting and automatic dataset selection:

The dataset description has to be a patterned input to minimize the cost of upload on blockchain. But if any dataset have some special quality, Owner can specify it. The problem of a general metadata format is considered out of scope here.



4.3 Performance

For a transaction to be mined in Bit Coin takes about 6 to 10 minutes on average calculated around 25-30 seconds for Ethereum, but these times are not always exacts as final time will depend upon the special cation in block itself. For a transaction to become irreversible, the transaction has to be confirmed by certain number of miners which is specified inside Block.

4.3.1 Publishing

Publishing advertisements about datasets can differ from seller to seller as each dataset has different attributes meaning different size of data has to be stored inside blockchain. This test tests if the size of blockchain affects the time it takes for publishing the advertisement. We wrote a JavaScript that uploads random data on the blockchain to increase the number of blocks on the blockchain rapidly

In this section we have tested how the amount of data to be deployed on the blockchain affects the time taken for data deployment also how it changes with the increases with the change in size of blockchain. For testing we've used two functions from our code addDatasetType() and addDataset() which are as follows:

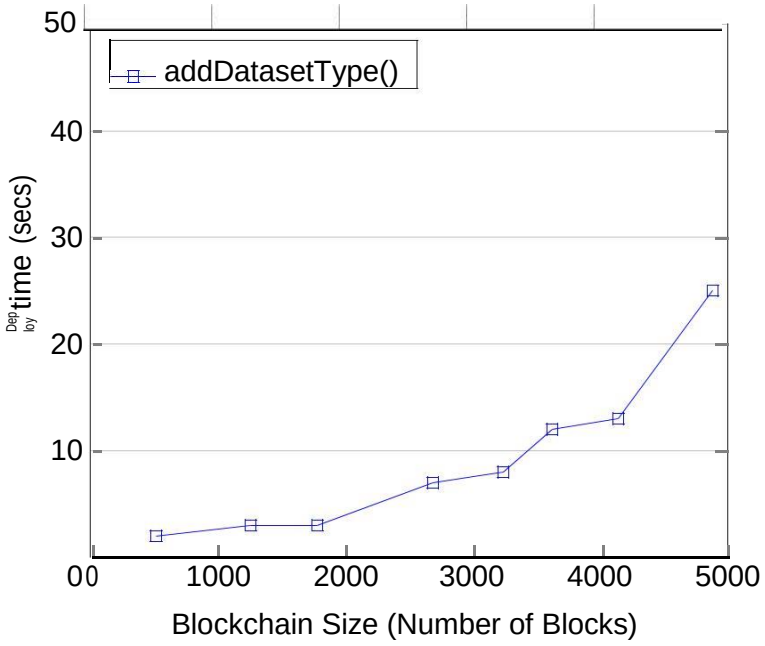
```
function addDatasetType ( string _datasetType ) returns ( string ){
    if ( list_datasetType [ _datasetType ]. doesExist != true ){
        list_datasetType [ _datasetType ]. datasetType = _ datasetType ;
        list_datasetType [ _datasetType ]. listHead = 0;
        list_datasetType [ _datasetType ]. doesExist = true ;
        return " new type success fully added ";
    }
    else {
        return " dataset type already exist you can add dataset of this type ";
    }
}
```

```
function addDatasetInfo ( string _datasetType , string _IP , string _datasetLocation , string
    _datasetDescription ,
string _encryptedURL , uint _cost ) {

list_datasetType [ _datasetType ] [ msg . sender ]. IP = _IP ;
list_datasetType [ _datasetType ] [ msg . sender ]. datasetLocation = _datasetLocation ;
list_datasetType [ _datasetType ] [ msg . sender ]. datasetDescription = _datasetDescription ;
list_datasetType [ _datasetType ] [ msg . sender ]. encryptedURL = _encryptedURL ;
list_datasetType [ _datasetType ] [ msg . sender ]. cost = _cost ;
```

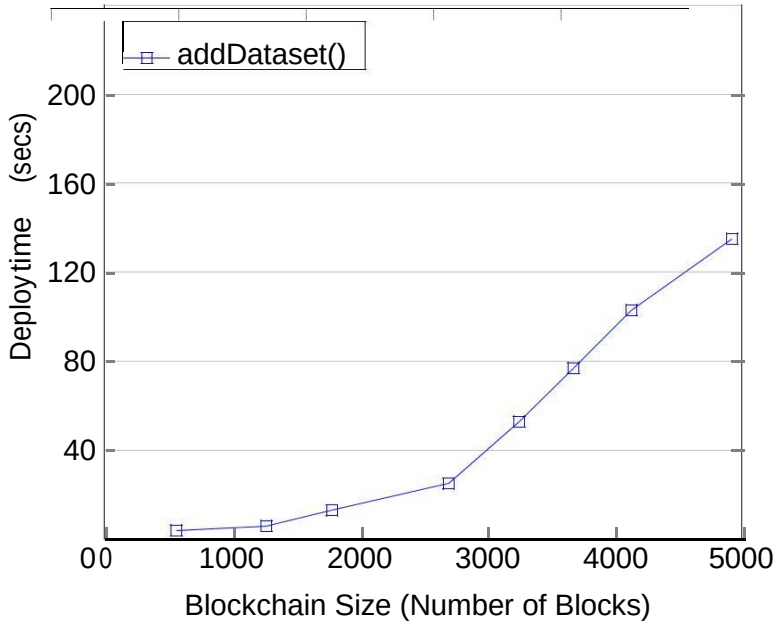
```
list_datasetInfo [ _ d a t a s e t T y p e ][ msg . sender ]. owner = msg . sender ;
list_datasetInfo [ _datasetType][ msg . sender ]. next = list_datasetType [_datasetType].
    listHead ;
list_datasetType [ _datasetType]. listHead = msg . sender ;

}


function initDatasetQos ( string datasetType ) {
    list_datasetQos [ _datasetType ][ msg . sender ]. datasetFeedBack = 0;
}

function addDataset ( string _datasetType , string _IP , string _datasetLocation , string
    _datasetDescription , string _encryptedURL ,
uint _cost , uint volume ) constant returns ( string ) {
    if (!list_datasetInfo [ _datasetType ][ msg . sender ] .doesExist ) {
        addDatasetInfo ( _datasetType , _IP , _datasetLocation , _datasetDescription ,
    _encryptedURL , _cost ) ;


        initDatasetQos ( _datasetType ) ;
        addUserReward ( volume ) ;
        return (" account successfully created ") ;
    }
    else {
        return " dataset already exist !!";
    }
}
```

From the code above it is clear that function addDataset() have more data to be deployed on blockchain than function addDatasetType().

Effect of Blockchain size on Deploy time for function addDatasetType



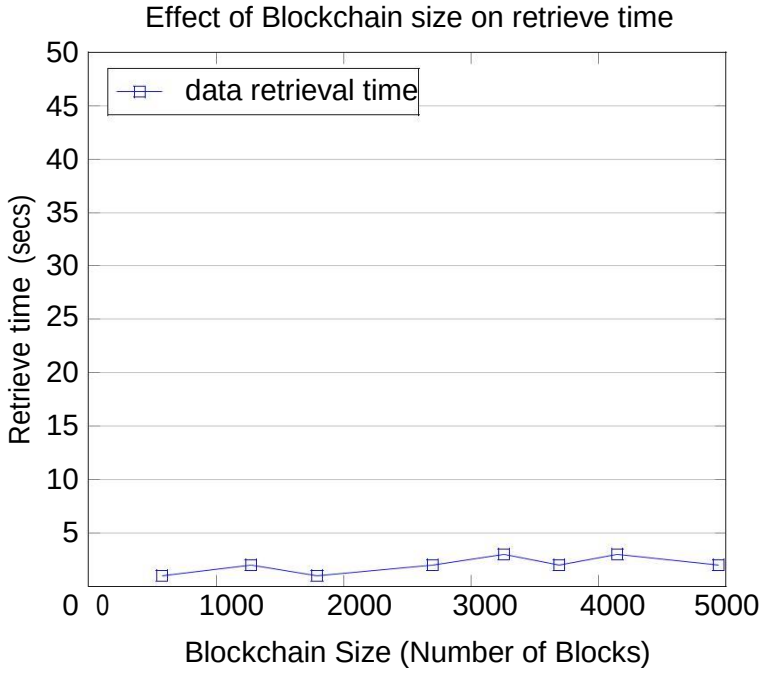
Effect of Blockchain size on Deploy time for function addDataset 240



From the graphs above it is clear that the time taken for deploying data on blockchain directly depends upon the size of the blockchain, leading us to believe that when size of blockchain will grow more than certain; the time for a transaction to complete will be unbearable. Though developers of Ethereum argues that because of the Ethereum's proof of stack algorithm [section 2.2] will not let the size of Blockchain affects the transition time to unbearable.

4.3.2 Matchmaking

The main feature of blockchain technology is that it replicates a whole copy of blockchain to each node, requires a public/private key to give access to data inside blockchain.



As you can see here because of a whole copy of blockchain is already available at each node, as the size of blockchain increases it does not affect the data retrieve time.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Conclusion

From the results from tests in section 4 it is clear that the time for deploying a data on the blockchain depends on both the size of metadata to be deployed and the size of the blockchain itself. Claim of the Ethereum developers about the Proof-of-Work concept for validating a block seems to be proven true as on an average with the current size of Ethereum blockchain it only takes about 25-30 seconds for a transaction to be mined. And also it doesn't matter how long it takes for a transaction to complete, as the whole copy of the blockchain available to each node locally retrieval of data from a blockchain is very fast.

The main purpose of this experiment was to determine if this newly arrived Blockchain technology is a viable approach for publishing metadata about scientific datasets, and from the results of our experiment it is clear that Blockchain technology is a viable approach for the given problem.

5.2 Future Work

Here the final output of list of datasets will be in a le. This means we can perform operations we need on the dataset advertisements and choose appropriate dataset for application, or application can choose whatever dataset will be suitable for itself. Replicas with the same cost can be differentiated by network cost. At the time when buyer buys dataset, a string with URL will be returned to buyer where it can access dataset, but once swarm developed and is fully functional will help integrate datasets with smart contracts. With the use of Whisper proto-col which is used for exchanging small data between nodes, swarm which provides distributed storage and holy grail protocol which will integrate hardware with smart contracts a power full sensor network can be deployed and monitored.

More QoS can be implemented once oats type will be available in future updates of Solidity. Right now it mainly have conditional variables to make decisions but once oats are available we can have exponential bounded by limits.

BIBLIOGRAPHY

[1] Don Tapscott and Alex Tapscott. Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World. Penguin, 2016.

[2] Diya Soubra. The 3vs that de ne big data. Data Science Central, 5, 2012.

[3] Rajesh Raman, Miron Livny, and Marvin Solomon. Matchmaking: Distributed resource management for high throughput computing. In High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on, pages 140{146. IEEE, 1998.

[4] Rajesh Raman, Marvin Solomon, Miron Livny, and Alain Roy. The classads language. In Grid resource management, pages 255{270. Kluwer Academic Publishers, 2004.

[5] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In Security and Privacy (SP), 2016 IEEE Symposium on, pages 839{858. IEEE, 2016.

[6] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. ACM computing surveys (CSUR), 36(4):335{371, 2004.

[7] Leili Mohammad Khanli and Morteza Analoui. Grid jqa: A qos guided scheduling algorithm for grid computing. In Parallel and Distributed Computing, 2007. ISPDC'07. Sixth International Symposium on, pages 34{34. IEEE, 2007.

[8] Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger. Economic models for resource management and scheduling in grid computing. Concurrency and computation: practice and experience, 14(13-15):1507{1542, 2002.

[9] Ian Foster and Carl Kesselman. What is the grid. A three point checklist, 20, 2003.

[10] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-g: a computation management agent for multi-institutional grids. In Proceedings 10th IEEE International Symposium on High Performance Distributed Computing, pages 55{63, 2001. doi: 10.1109/ HPDC.2001.945176.

[11] R. Raman, M. Livny, and M. Solomon. Matchmaking: distributed resource management for high throughput computing. In Proceedings. The Seventh International Symposium on High Performance Distributed Computing (Cat. No.98TB100244), pages 140{146, Jul 1998. doi: 10.1109/HPDC.1998.709966.

[12] S. Vazhkudai, S. Tuecke, and I. Foster. Replica selection in the globus data grid. In Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 106{113, 2001. doi: 10.1109/CCGRID.2001.923182.

VITA

Shirish Patel

University of Mississippi
Oxford, MS 38677 U.S.A.
Phone: 769-823-8953
email: srpatel3@go.olemiss.edu
DOB: August, 21 1994

Objective

To take a challenging and high performance oriented role in the field of Computer Science and implement the expertise and experience gained in this field to develop complex project with efficiency and quality.

Current Position

2015-presentGraduate Student
Department of Computer and Information Science
University of Mississippi
Current GPA-3.75

Education

2011-2015Under Graduate
Gujarat Technological University
GPA-3.7

Projects

Mobile VoIP - A call conferencing application in android with configured server on Ubuntu.

Artificial Intelligence course Project- An intelligent agent that can play Conniption game against human (purely coded in Java).

Parallel programming course project- Parallelized K-Means algorithm with increased performance and efficiency.

Operating System course project- Starvation free solution for banker's algorithm and dining philosopher problem.

Technical Skills

Languages Core Java (Simple Programming, Multithreading, Socket Programming), J2EE, C (Simple Programming, Multithreading), C++, HTML, CSS, JavaScript, UML.

Database MySQL

Tools GitHub, GDB, Make le, Shell Script

Experience

Jan 2016 - present Teaching Assistant for Java Advance, Computer Organization and Architecture at University of Mississippi, Oxford, MS

June 2016 – present Research member under Dr. Philip Rhodes, currently developing standalone application in C++ to write in block-chain bypassing the protocols of CPP-Ethereum.

Summer 2016 Student Worker at Dr. Coy Waller research laboratory for summer