School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales

Analysing Data Quality with a Decentralised Blockchain System

by

Tomas Donovic

Report submitted as a requirement for the degree of

Batchelor of Software Engineering

Submitted: 4/12/2019

Supervisor: Dr Helen Paik & Dr Marco Comuzzi

Student ID: z5020506

Topic ID:

Abstract

Blockchain systems have been used widely to make processes that otherwise required large amounts of trust (movement of funds, etc.), trustless. Permissioned blockchains have meant that decentralisation can be applied to an extended set of use cases. Data quality analysis, a previously centralised process, can now be performed in a decentralised setting. This project implements a decentralised blockchain system to analyse data quality. The system implemented is far slower than a centralised solution. However, due to it's nature of being a decentralised solution offers privacy, finality guarantees, and much lower trust of third parties to use than a centralised solution.

Acknowledgements

I'd like to thank my supervisors Helen and Marco for their advice, patience and insight. Thanks to Juan Delard and his 2018 optimism for decentralisation and everything blockchain, for inspiring my interest in the topic. Thanks Matt Hollingdale for giving me the time and experience to make this reality.

Contents

Chapter 1

Introduction

Data is oft regarded as the new oil. The similarities are canny: both have pipelines, markets and growing global demand for both. Data is becoming increasingly commonly traded, exchange, bought or sold, however, an important function in the market is missing. While there are different designations of oil quality, assessing data quality for exchange is lacking.

To satiate the increased demand for data from both the commercial and academic sphere, many organisations are starting to exchange data with each-other. This need has arisen since they either need to enrich their own data in house, or they may even require the input of an external provider to help them to analyse the data they have, in order to remain competitive with other data rich organisations.

It is currently difficult for these organisations to perform these exchanges, because there is a lack of measure of the quality of the data they are exchanging. This problem will only become more prolific as autonomous systems attempt to source and exchange data without human intervention. This problem is compounded by the lack of trust: no participant will hand over their entire dataset to the purchasing party, or potentially a trusted third party, without payment. Additionally, sharing data with other parties is an increasingly large regulatory problem, with consumer data protection legislation such as GDPR (in Europe) and CCPA (in California).

Assessing data quality while dealing with this variety of issues is the motivation for this project.

With the advent and proliferation of decentralised, permissioned blockchain systems, the tools to solve this problem are now available. By constructing a network of co-operative, unaffiliated parties, it is possible to split the risk of misuse of the dataset by quality assuring parties by splitting the dataset amongst them. Additionally, by building a blockchain system, a robust audit trail can be built, and more importantly, trusted, due to the ledger's immutable nature, and the formation of consensus between nodes in the network. By limiting the amount of data released to any one party, both row and field wise, it is also possible to minimise reidentification risk of end users within a dataset.

This paper details the implementation of such a system, by performing some simple data quality analysis in a decentralised manner, to show the feasibility of such a system. We compare the performance of running such analysis on a decentralised system, compared to operating in a centralised manner, and examine the system through the lens of software architecture quality attributes.

Our results show much slower operation in a decentralised mode, however, this performance loss must be contrasted with the increased utility that a trustless system brings.

Chapter 2

Background

This chapter gives an overview of the three main concepts that this project synthesises. We discuss the concept of data quality, the proliferation of data exchange and finally decentralised blockchain systems.

2.1 Data Quality

Data Quality (DQ) is a measurement of various attributes of a dataset in order to gain understanding as to whether it is fit for purpose. It is important that data is of the expected quality, as critical decisions, be it in business, medical or academic fields, are made from it.

2.1.1 What's wrong with low quality data?

There are massive impacts to consuming data of different quality than expected: Gartner reports "poor data quality to be responsible for an average of $15 million per year in losses." [MMA12]. From a less financial point of view, Forbes reports "businesses can see loss of reputation, missed opportunities and higher-risk decision making as a result of low confidence in data." [Ins18]. Now, more than ever, whether in the academic or business spheres, organisations are looking externally to enrich their datasets with those from other providers or sectors. The risks of poor data quality are compounded when consuming this externally sourced data, not only may some attributes be misrepresented or poorly quantified, but that low quality data had some cost to acquire also. Such a situation illustrates how important characterising the quality of a dataset before consuming it is.

2.1.2 Measuring Data Quality

We consider three main ways that data can be stored: relationally, non-relationally and in a decentralised manner. While there are significant differences between how quality needs to be measured between these groups, there are also these in common:

Attribute	Description
Accuracy	How close is the value to the “true” value
Completeness	How many records are missing
Timeliness	Was data collected and/or delivered on time

DQ attributes and descriptions

Timeliness

Timeliness is an indication of whether the data was collected and or reported when it was expected to be [TB98]. In the geospatial field, Veregin (1999) [Ver99] details timeliness as a key measure as it provides some guarantee that comparing places and their state can be done with surety, since readings taken at the same time guarantees that external events haven't modified the state of the world. Similarly, when comparing the patient outcomes of cancer patients, Larsen et al (2009) [LSJ+09] took into account how early or late cancer was diagnosed. Without quantifying these differences, a correct conclusion could not be drawn between the outcomes, since another variable could confound their conclusions: time.

Consistency

Consistency can be divided into two main cases, reporting and statistical. When researchers in Finland analysed the quality of data residing in their population sized cancer registry, they found that doctors often reported that cancers were at a different stage than they actually were [TPL94]. This was due to the overcomplication of reporting these measures, ie. many different codes to describe a similar issue. These reporting consistency issues can, in some cases be alleviated if these values can be cross checked or verified by using calculations of other attributes related to that entry [Fan15]. Consistency can also refer to statistical measures of the deviation of a set, for example, the propensity of a set to contain outlier values that must be discarded. Last Kandel (2001) propose a method using a lookahead window and modelling the subsets distribution to see if a value is an outlier or not [LK01].

Completeness

Completeness refers to how many, if any, records from a dataset are missing [WW96]. This is a key measure, since missing data can have just as profound an impact on actions as incorrect data. Take for example, a dataset consists of time series entries to a log file from a software application. However, entries between two points in time are completely missing. If a consumer of this data is not aware of this limitation, they could conclude that the application had a longer uptime than it actually did, since there were no records to explicitly say that the application was not running. Completeness can be computed without specialist external data, since only the interval of recording or how many entities are expected per batch are required for its calculation.

Accuracy

Accuracy can be described as whether a reported value is close enough to its actual value in the real world [MMA12]. This sets it aside from the other indicators of DQ, since it requires external data, other than that contained within the dataset to determine its quality. However, accuracy can also be defined as how close elements of a dataset are relative to each other [BCFM09], and this would not require external data to asses. For example, if sensor values from an outdoor thermometer have fluctuated between 0^{o}c and 30^{o}c in a day, but external sources suggest that the range for that day was only 15^{o}c and 25^{o}c, that day's datapoint could be considered inaccuracy by the first definition of inaccuracy. If, on the other hand, instead of being compared to an external source, the day's data-points were compared to the previous and following week's record and both fell out of the range reported on other days, this would be inaccurate in the second sense.

In this project, we group the second type of accuracy in with consistency, as a method that only relies on statistics as a proxy for quality. Accuracy is an incredibly important measure of quality, as all the other metrics of DQ may paint a positive picture of a dataset, but if those values have no resemblance to the real world, that data cannot be used to make effective decisions. Whilst these measures broadly apply to each of the categories, there are significant differences in how these are measured in each category. Additionally, decision rules must be applied to the metrics that are gained back from analysis, for example, a data consumer may not care if data was written in a timely manner, but in order for the data to be useful to them must have less than a certain variability.

2.1.3 Differences measuring DQ in different systems

While we have defined various attributes of data quality, implementing the measurements of these attributes can vary greatly depending on which type of datastore they must be measured on.

Relational

Relational data-stores, such as RDBMS (Relational Database Management Systems) are the first stage in the evolution of datastore technologies. These databases have strong controls for consistency control as the table schemas are commonly enforced on write. This means that consistency can be guaranteed to some degree if correct controls are put into table design, as inconsistent values cannot be written to a table if they do not conform to the schema as defined.

Non-Relational

The biggest difference between non-relational data sets and those stored in traditional RDBMSs has to do with schema. Whilst RDBMSs enforce schema on read, non-relational data stored in data lakes like Apache Hive, Facebook's Presto or Amazon's Glue and Athena enforce their schema on read. Cursorily, this seems like non issue: data is consistently written to the store, why would there be variation on write? However, with changing data sources, different versions of software writing to the store, or bugs that made it through QA, the data on disk could be quite varied. This problem is further compounded: only some subset is analysed at the time of each query to detect the schema, schema shifts may not be detected over time. Merino et al (2016) proposes a model that prioritises contextual, temporal and operational adequacy in order to address these concerns [MCR+16].

Decentralised Data

Dealing with decentralised data is even more difficult than the previous categories. Not only may datasets have properties of either relational or non-relational datasets, the data is also being contributed by many potentially untrustworthy sources. Complicating matters further, each contributor may not trust the consumer with all the data before the transaction has taken place. Additionally, transactions of data are made autonomously by interacting systems, and that same data may be shared with multiple other parties. Current solutions like that posed by the Ocean Protocol use an economic incentives to signal DQ [Fou19]. Participants in the network can "stake", or buy shares in a dataset or service, and reap the rewards if and when they are increasingly consumed. Whilst this is a successful mechanism for indicating dataset popularity, it only signifies DQ by proxy, and doesn't solve the fundamental problem of how to actually assess the quality of data, objectively, before it is consumed. This forms the part of the problem that this thesis project aims to solve.

When implementing these quality measures in a decentralised system, broadly the different strategies described fall into two categories, those measures that can be computed without external data (rule based) and those that require external information (knowledge based). Rule based measures include completeness, consistency and timeliness, since these measures can be determined entirely algorithmically, for example, if a time series database is missing some entries in the sequence, or if the standard deviation of a set of records falls outside what is considered adequate. On the other hand, knowledge based measures are usually some form of accuracy, for example, it can only be determined if a value is completely inaccurate, an outlier, say, if a single temperature measurement falls outside the all time high for that region. In this case, the data needed to determine this outlier needs to be gathered externally, and compared to the value in question to form an opinion on quality.

2.2 Blockchain Systems

At its most basic level, a blockchain is a linked list, containing the hash of the previous entry. This idea was enhanced by Haber and Scott (1990) [HS90] to create a timestamping notarisation service, whereby entries in the chain of blocks could not be rearranged or otherwise tampered with once they had been committed to the chain. Whilst other projects like b-money and bit-gold [Pec12] were established earlier, the concept of blockchain gained notoriety when used to create Bitcoin, the first widely adopted decentralised digital currency. Satoshi Nakamoto modified the chain to become a Merkel tree of blocks, and perhaps more importantly, a proof of work system was described [N+08] that would ensure that the system could continue to update its distributed ledger, without any one central party controlling the who chose which participant would be allowed to create the next block. Systems like this would come to be known as decentralised systems, ie. those that function without a centralised authority through the use of cryptography and mathematical proofs. In 2015, the second generation of blockchain systems became widely available and made significant changes compared to the currency-centric first generation. Specifically, Ethereum introduced the a decentralised compute platform [W+14] that allowed scripts to be run in a decentralised manner across the network. This meant that applications or even other currencies could be built atop the Ethereum platform. More broadly, this idea became known as the programmable blockchain [XPZ+16].

2.2.1 Permissionless Blockchains

Permission-less, public blockchains like Bitcoin or Ethereum have an important place in the world of decentralisation. They provide wide reaching guarantees into finality and fairness despite requiring no trust except the integrity of the network itself. This is really worthwhile for applications like digital currencies like Bitcoin, where very little computation is performed to move assets from one entity to another.

Not all cases are so computationally minimal though. Ethereum allows its blockchain to be programmed, but requires some tradeoffs in order to implement this. In order to maintain consensus, a smart contract is executed by at a minimum, 51% of the nodes in the network. As a network size grows, the cost of using said network also grows with it. Even in cases where the network is used for computationally simple transactions, as networks' usage increases, this approach isn't efficient enough or fast enough. Increased load on the network means transactions take longer and longer to finalise, and require more incentive to notarise, making them more expensive on two axis. If attempting to assess the data quality of a multi gigabyte dataset, smart contracts will be an incredibly expensive way to do this processing.

Operating in a trustless, permissionless mode also means that for state to be persisted in the network, a majority of nodes need to store that state, to come to a consensus. Currently, the two major public blockchains, Bitcoin and Ethereum, utilise proof of work to form consensus (POW) [XPZ$^+$16]. Whilst POW is a proven and secure method for forming consensus, it is also incredible wasteful: Vranken (2017) estimates the Bitcoin network consumes 100-500MW of energy [Vra17], equivalent to the power consumption of a small country.

Additionally, whether these public networks are actually truly decentralised is also an issue: 75% of the Bitcoin network is controlled by six main groups of miners [GKCC14]. This means that just two or three of these groups would need to collude and would be able to form consensus without the rest of the network, validating whichever blocks they chose, and potentially re-writing history. In the case of executing smart contracts on the Ethereum Virtual Machine (EVM) this means that every node in the network must perform the computation of every smart contract. This process means that the integrity of the network can be preserved, ie. any node can check to see if any other node has actually completed the computation, rather than returning a false value. However, this also means that as the network grows, and until measures like sharding have been put in place, the cost and number of parties that execution data is shared with will continue to rise. This makes computation on the EVM and other decentralised computers incredibly expensive, and incentivises doing work that does absolutely need to be trust-less elsewhere.

As of recently, public blockchain platforms have fallen out of favour. Many projects built on top of Ethereum and other programmable blockchains offered tokens to the public for purchase, as a means to raise funds to build the project that would eventually use said tokens as a currency, often referred to as an initial coin offering (ICO), or more recently, initial token offering (ITO). While there have been some variations on how such offerings have taken place, specifically trying to skirt the United States' Securities and Exchange Commision (SEC) regulations around what defines a security [Lev19], even the most compliant offerings have failed to appease regulators. Telegram, a widely used end to end encrypted chat app, launched an ITO to raise funds to build a decentralised platform for payments within its app. However, despite its best efforts at following SEC guidelines, it ultimately looks unlikely to succeed, with actions pending against it [Lev19].

Another limitation of public blockchains is around privacy. Since any member of the network can see a global view of state in the network, no data written to the chain is private. In cases where sensitive data must be processed, another solution must be found.

2.2.2 Permissioned Blockchains

Permissioned blockchains differ from permission-less ones by requiring some trust of a central entity, rather than in the integrity of the system as a whole. Here, permissioned systems rely on forming a consortium of trusted entities, rather than relying on trusting that the system keeps itself completely honest. In permissionless systems, consensus protocols are still required, but are not required to be Byzantine fault tolerant, and thus don't require the massive computational overheads of POW. This means that work can be distributed across multiple nodes and but doesn't need to be run on every node. This means that consensus can be formed quickly and cheaply. The most well developed permissioned networks is Hyperledger Fabric (HLF, or Fabric). Here, trust in the network is established by explicitly trusting some central entities. Thes entities orchestrate the network, and issue x509 certificates to each participant in network. Additionally, the central entity is able to institute access controls (ACLs). These ACLs mean that smaller, private partitioned networks are able to be established within the main network, and data can be shared between nodes in a network without others necessarily being able to access it. These permissioned components mean that Fabric can be used to not only build a mostly decentralised network, but a distributed system too.

Since data can be selectively shared with specific nodes, using the Fabric's sideDB component, a sidecar database that can have ACLs applied to its contents. As a result, each node can do a subset of work, not duplicating all the work other nodes have done and importantly not seeing any of the data that other nodes are working on. Additionally, certain smart contracts (known as chaincode in HLF) can be deployed onto different nodes in the network, meaning that specific parties don't necessarily have all the same tasks to complete as others. In this way, it is possible to create a system where less trust is required than asking a single third party to perform some process, but that work can also split work amongst workers, in a distributed manner, and finally using built in consensus mechanisms, form a final consensus on the process that was split across those workers, in a decentralised manner. Results from this smaller, semi-trustless network could be written back to a broader system like that proposed by Ocean Protocol, providing extra functionality, while reducing complexity compared to a completely trust-less network. The proposed system would also mean that in more traditional data exchanges, such as between two organisations, no central party would need to be trusted to determine quality. Instead, the contributor could write their data to this permissioned system, and have the quality processed in many individual, small pieces. As a result, no single party has enough raw data to be of concern, but when combining their findings together, can provide useful and holistic quality analysis. The exchange could then go on with confidence that the contributor hasn't misrepresented the dataset they are providing.

Chapter 3

System Design and Implementation

In order to introduce the system design, it is useful to present a motivating example, where such a system could be used, such as the case of data exchange between a distributed sensor network data provider, and an outdoor media company.

3.1 A Motivating Example

CityWifi is a municipal wifi provider that with their city wide network of Wifi access points. These APs record which devices are connected to them at any point in time. When mapped and examined in a timewise fashion, this data can be used to map congestion, as well as the flow of people through the city, as well as link people to the identity they used to sign up for the free wifi with. An advertising company AdCo, wants to place ads more intelligently on digital billboards, and show ads that are relevant, down to the five minute block that they expect certain people will walk by, in order to raise the efficacy of their advertising.

To achieve this, AdCo approaches CityWifi to purchase their dataset. However, AdCo is far from comfortable with paying for the data without some guarantee as to its quality, since they may not be able to run their personalisation program without a certainty that only some small amount of data points are missing. Conversely, CityWifi won't provide the full data set to AdCo upfront, rather only a small synthetic sample, which doesn't give them the confidence they need to sign on to the deal. Clearly an impasse has been reached. Traditional data quality assurances do not work in this case, because AdCo have no guarantee that the statistics on the data provided are representative of the set as a whole, and the CityWifi don't feel comfortable using a third party to determine the data quality either, since they must provide them the whole dataset also. Whilst there are legals in place to mitigate all of these factors, still neither party feels confident enough to go through with the deal, as they would feel more comfortable with some technical controls put in place as mitigation in addition to the protections provided in the legals. Additionally, neither company wants to fall foul of their responsibilities regarding GDPR or CCPA, which could leave them with fines in the millions if data is handled incorrectly.

In this scenario, a system as described in this paper would solve this problem. City-Wifi's concerns would be resolved, as their risk of their dataset being exposed in entirety is completely mitigated, since it would be split into n parts and distributed to unrelated actors in the network. Additionally, the dataset could be stripped of all other components than location and timescale, and these attributes could be split into two different groups. Further, the location data could be randomly chunked and allocated from the original set, reducing the re-identification risk, as well as its usefulness if it leaked.

In this way, it is possible to achieve the aims of both CityWifi and AdCo, while mitigating the risk they both posed to each other.

Given the challenges explored in the Background and materialised by a such a sample problem, we designed a blockchain based, distributed system for assessing data quality. By leveraging the decentralised nature of a blockchain system, but the inherent trust in a permissioned blockchain system, we can distribute computation amongst many peers, while maintaining trust in the results, and the integrity of the system.

3.2 Application Flow

In terms of application flow, users can define how they want their data quality to be tested (create a data quality pipeline). Then, they can add data to that pipeline, they have reached their defined number of records. Then, the dataset is broken into a user defined number of pieces, and written to private storage on the blockchain. Once these partitions are available, workers on the blockchain network read these partitions, compute their data quality as defined in the original contract, and write their results back to the blockchain. As a final step, once the results are aggregated and written back to the chain, and the original client is notified. In this way, a trustworthy attestation of the quality of some dataset can be provided.

Specifically, we split the workload up and allocate the work to different, disparate nodes. In this way, we can both split the workload, and also the risk: no one client will get a partition of data large enough to compromise the dataset.

While the whole system could be implemented as chaincode, running decentralised across the blockchain network, we instead pursued a hybrid approach. This meant that operations didn't need, and were detrimental to performance when performed blockchain system component, weren't. In this way, we formed a separation of the system.

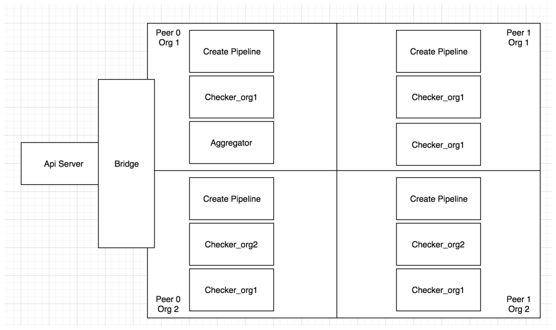

Figure 3.1: Block Diagram

The system consists of two main components, a local client, which interacts with, chaincode running on the HLF network, instantiated on a single node per organisation, of which there are two.

3.3 Local Client

The local client presents functionality required to create a data quality pipeline, partition the dataset, and make that dataset available to be written to the blockchain. While HLF is versatile, and these operations could have been implemented in chaincode, there were only drawbacks of taking this method. Execution times are limited with block times, which would have made partitioning large datasets difficult, and executing this chaincode on only one node, the contributing org's node, would have added no real audit, or security benefits. As a result, we implemented this functionality in a self contained, local Golang binary. The local application also makes the necessary calls to the chaincode running on HLF.

To ease integration with other systems, the binary starts a HTTP server, allowing other applications to interface with it, using its RESTful API.

3.3.1 Managing Data

Managing the data added to the system was attempted a few ways. The first naive approach was to simply store the data as raw CSVs, and manipulate those CSVs manually, in memory. Whilst this solution was simple to begin with, it became increasingly complicated since the data needed to be marshalled and unmarshalled continually, and each time using my own methods, which were likely slow and definitely cumbersome. Instead of manipulating these CSVs directly, it made more sense to encapsulate them in some other database to leverage their methods for manipulating this data. On this front, there were several options to choose from. Both LeverDB and CouchDB are implemented by HLF for use in private data storage, so reusing one of these may have eased development between the chaincode and off chain software. CouchDB was more complicated and way larger than we required for its uses in the project, and LevelDB is explicitly just a key/value store, making it difficult to use for partitioning datasets, however, its form as an embeddable library was really enticing. As a library, the functionality could be compiled into the main application binary, not requiring running a separate server. Similar in this regard was SQLite, a module providing SQL functionality, running in memory or on disk. Furthermore, compared to a naive implementation, SQLite is faster and less work to implement, and likely much better written given its age and following. As a result, this was a perfect fit for the needs of this project. Rather than sending JSON or raw CSVs, the system transacts using SQLite databases. Additionally, this means it is trivial to implement extra data quality checking rules. Whereas in the inception of the project, we defined some rules that could be reused, but were ultimately hard coded into the system, now rules can be expressed as SQLite compliant SQL, with the addition of some keyword extensions.

3.3.2 Partitioning Work

Blockchain systems establish trust through consensus: multiple entities do the same job, and we trust that it happened when most of the participants agree that it happened. Simultaneously, in the case of analysing data quality, we don't want to give any one entity in the network too large of a partition of the dataset, since it may be sensitive, or compromise the value of the set. As a result, in this system we use both consensus and partitioning to ensure trust in the system. A dataset can be partitioned into as small partitions as possible, both by excluding some fields, and simply by only including some subset of records.

table

At the discretion of the contributing party, partitions will contain a large enough amount of data to gain some insight into data quality, but also too little data to be revealing on their own. In this way, we can minimise the amount of data loss to each analysing party, but still retain being able to analyse the whole dataset.

3.3.3 Interacting with Hyperledger Fabric

Typically the HLF network, and the node running for the contributing organisation would be potentially run on a different host to the one running the contribution software. In a setup like this, HLF provides some functionality to create a GRPC gateway, allowing external applications to interface with the software running on chain. However, in our case, the whole network runs inside docker, on the same host as the client. While implementing the gateway would have been beneficial from a versatility standpoint, the lack of documentation for the Golang SDK made it difficult to implement this. Instead, we simply execute a command to tell Docker to run some executable within the container, with the parameters we pass. In this way, we are easily able to interface with the HLF network, without developing the gateway.

3.4 The Blockchain System (and Chaincode)

The remaining functionality of the system is implemented as chaincode on top of Hyperledger Fabric, a decentralised blockchain system. Where the implementation of this functionality varies from a more traditional, permissionless blockchain system is in the ability to choose which nodes and organisations execute which instances of chaincode. The chaincode has functionality that is required by the initiating party, like instantiating a data quality pipeline, writing SQLite databases to the SideDB, and aggregating the results of the analysis, as well as the analysis functionality must take place on all orgs. The network consists of two organisations, each running two nodes.

3.5 Network Organisation

The blockchain system was implemented into three main components. That is, Lodger, Checker and Aggregator.

3.5.1 Lodger

This component runs on all four nodes, and requires each org to have reached the same result for the orderer to accept its transaction. This component creates a data quality pipeline inside Fabric. The requirement for all orgs to agree on pipelines created is to ensure that all entities understand what their job is upfront. This chaincode also exposes functionality to get pipeline specifications for other chaincodes.

3.5.2 Checker

While this component runs on all four nodes, it runs with separate consensus between the orgs. Since different orgs can be passed separate partitions of data, it makes sense that consensus should only need to be formed within an org in this case. This chaincode can ingest data from the API server, and run the analysis prescribed by the pipeline. It exposes its results with a getter method.

3.5.3 Aggregator

This component again (as with Lodger) must form consensus across all orgs. It reads the pipeline defintion from Lodger, and results from Checkers, and aggregates those results, and finally writes them back to the chain, exposing them again with a getter.

3.5.4 SideDB

SideDB One of the biggest advantages of using Hyperledger Fabric, is SideDB and private data. In traditional, permissionless blockchain system, all data is shared either by writing it on the chain, or by providing a reference of some external place to download that data. In the case of this project, writing all the data straight to the chain has two issues, speed and security. Firstly, there is a definite limit to how quickly blocks (testing the speed) can be written to the chain. In the case of providing the data to various nodes in the network, we don't really want the data to live on the chain forever. We especially don't want the data to be persisted for all nodes forever, that would go directly against what we were trying to achieve with the project. Instead, Fabric allows us to specify that certain data should only be accessible to certain orgs or nodes in the network. Additionally, this data is never written to the main blockchain, rather, only to this sideDB.

Chapter 4

Results

To qualify the success of our system, we evaluate it quantitatively by comparing its performance running in a standalone, centralised mode, to running in decentralised blockchain mode. Additionally, we assess its use qualitatively when applied to a variety of business cases, examining both the costs and benefits of using the system in such context. Finally, we assess the system in comparison to a both a traditional distributed system, as well as a traditional decentralised system.

4.1 Quantitative Analysis

To characterise the performance of the system, as well as the overhead that operation with Hyperledger Fabric introduces, we profiled system performance running both in standalone, and decentralised mode. Additionally, we modified the dataset size as well as batch size in order to more thoroughly understand how the system performs. We analysed if there were any missing fields in some BPI data. These were partitioned by BPI cases, with each case representing around 5000 records.

We compared the average performance of creating a job, loading data, partitioning, analysing and aggregating on different dataset sizes, as well as when running in a distributed and standalone mode.

The performance when running in decentralised mode was dramatically slower, nearly 200x slower in some cases. The performance difference is stark, given a linear scale, and the inability to even see the off chain bars on the chart. What is clear here, though, is most of the time taken in processing, regardless of the dataset size, was the loading data part of the process. Here, chaincode makes a request to the api server and gets back the requested SQLite db as a base64 string, and writes it to SideDB. It seems that writing this data to SideDB has massive overheads, as this process (which happens on disk, and already not the fastest way), as seen in the diagram below, happens in tenths of a second. The low when writing to SideDB could be to do with Docker utilising

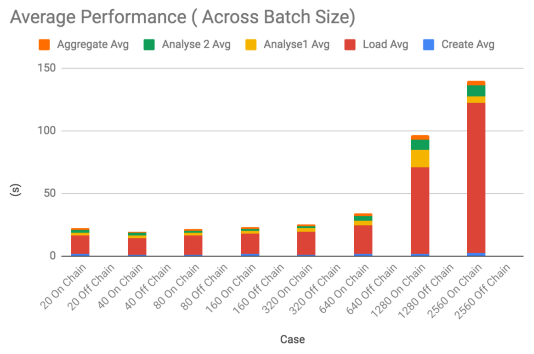

Figure 4.1: Average Performance, On Chain vs Off Chain

OverlayFS, which is known to slow down write operations. However, this should still only be small, given the maximum dataset size written was in the order of several MB.

Sets with more than 2560 cases were tried, but all ultimately failed. This was due to the limited amount of time that a transaction can take in this implementation of Fabric: 300 seconds. When trying to load datasets with these increased number of records, the process took too long, and the transaction timed out. For this reason, we are limited to datasets with 2560 cases, or around 50000 records.

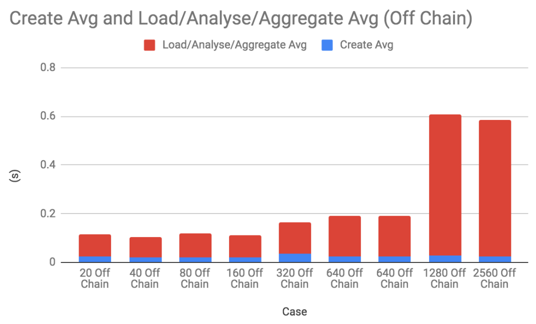

Figure 4.2: Average Performance, Off Chain

When running in standalone, or off chain mode, performance was consistent regardless of dataset size. However, this performance degraded massively when moving to and a 1280 case dataset. This may have happened due to running out of memory on the test machine, and needing to write that data to disk, rather than keeping it in memory.

We also modified the number of partitions made in each analysis, to see what impact this would have on performance. In the off chain example, in most cases, partitioning the dataset into smaller partitions clearly made performance worse. This is most likely due to the overhead that opening files with SQLite introduces. While in the case of on chain analysis, we see this trend for the most part, apart from when the number of records in each partition gets large. In the 1280 and 2560 case runs, four partitions performed the best. This was likely due to the balance between adding latency by

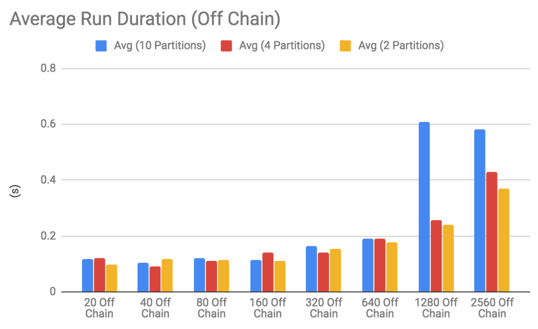

Figure 4.3: Average Performance, Modifying Partitions, Off Chain

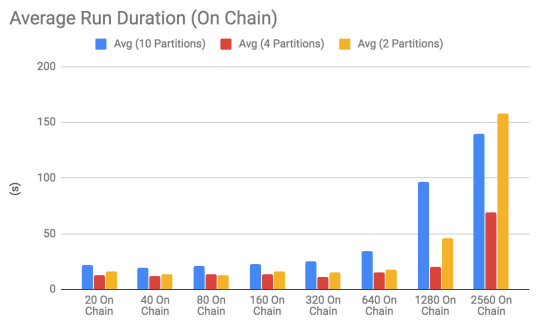

Figure 4.4: Average Performance, Modifying Partitions, On Chain

making more requests for records and how potentially the same OverlayFS write issue described before.

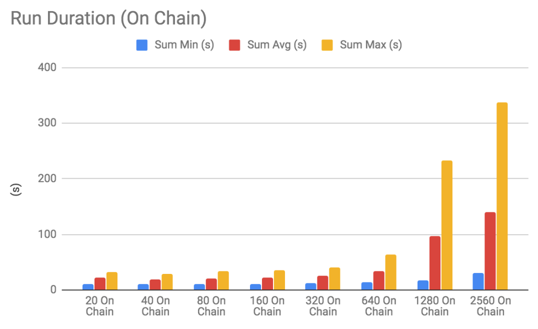

Figure 4.5: Average Performance, Modifying Partitions, Off Chain

The final set of analysis was to look at how much variance there was in different runs of the same record volume. There was far more variance in the blockchain implementation, as a post to running off chain, which was exaggerated the most when dealing with larger sets. This increased variance was due to the variance that Hyperledger Fabric introduced to the system, as there was comparatively less difference in the on chain example.

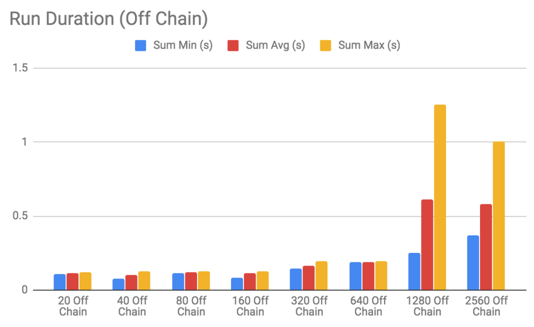

Figure 4.6: Average Performance, Modifying Partitions, On Chain

Overall, we can see that running analysis, regardless of batch size or partition size, runs both slower, and with less consistency than when running in off chain mode.

4.2 Qualitative Analysis

While running the system in blockchain mode is slower, and less consistent than running off chain. However, there are many other ways to evaluate the systems performance and many ways that the system is better than an off chain implementation.

4.2.1 Data Privacy

One of the main reasons to engineer and adopt a system as described here is to preserve the privacy and ownership utility of data. Data privacy is important to preserve in the context of increasing regulation around the collection and dissemination of user data, such as GDPR and CCPA. Entities collecting this data face large fines under these regulations if they use the personal data they collected without gaining prior consent from their users. Particularly, if personal data is transmitted to a third party without consent of the user, huge fines could be faced. An example of where this could be an issue is when sharing user data with a single third party for data quality checking: since entire entries about a user are shared, and considered Personally Identifiable Information (PII), such a transaction could be in breach of GDPR or similar.

Specifically, PII is defined as a set of data that makes it possible to identify a person given that data. For example, a date of birth, postcode and gender would be considered PII since these data points are unique enough to reidentify an individual. However, just providing postcode and gender, for example, would not be considered PII, since the data is not granular enough to reidentify the individual without further enrichment of the data.

This problem is explicitly solved by the system implemented by this project, specifically via two means, firstly, by field level control of dispersion, and by dataset partitioning.

Regardless of which method is used, regulatory, privacy and performance perspective, transmitting the smallest amount of data to any one party is the most conservative and best.

Field Level Control

In a traditional system, the entire dataset would be transmitted to the DQ analysing party, and then various analysis would be conducted on that dataset. If that party is trusted and there is consent from the consumer over that data, this method is fine. However, in many cases, neither of these bars have been met. The system in this paper removes these barriers by field level controls over transmission. This means that for a given data quality analysis job, only the fields that are required for some analysis are transmitted to that party. This reduces the reidentification risk of individuals to an acceptable level, such that if an entire partition of data was compromised, not enough information about any single user would be exposed to reidentify them.

Dataset Partitioning

Again, where previous techniques would have transmitted an entire dataset to a third party for analysis, this implementation takes a more risk averse approach. For many measures of data quality, every record of the dataset does not need to be present in order to derive these statistics. For example, checking for missing fields, or malformed values can be done row by row, with the results aggregated finally. This technique can be extended to more rigorous statistical measures also, for example the standard deviation for some measure can calculated over many partitions of a dataset, and then aggregated finally using the square root of those pooled variances. In its current implementation, the system does not support calculating these values, as SQLite does not provide a standard deviation function. However, with further development, these measures could be added as further modules to the analysis chaincode.

4.2.2 Security

Unlike in permissionless blockchains, Hyperledger Fabric deals with the idea of identity. Each organisation in the network has a unique set of cryptographic material, that when leveraged to provide identity to nodes in the network, is known as a Membership Service Provider (or MSP). In order for the network to function, each party operating in the network must trust the others identity is who they say they are. At a basic level, as long as the ownership of the cryptographic material is proven to the other organisations in the network, its use thereon can be guaranteed to be by that party.

In addition to this basic concept, HLF leverages x.509 hierarchical trust, so that sub organisations can be vouched for by their originally trusted parents. This functions in the same way as any other x.509 system, for example the issuance of SSL certificates for websites. In the same way, that browsers trust a root CA and its issuance of certificates, so do do orgs within HLF. In addition to this trust model, SSL is utlised for all communication within the network to keep all data on the wire safe from interception. Data at rest, stored either on the shared ledger or in private data is not encrypted and HLF does not support a transparent way to encrypt this data. Further work could be done to implement per organisation encryption here. This would greatly increase the security of the system and is a necessity if handling any real sensitive data.

4.2.3 Usability

The system as it stands is not designed to be consumed by a non technical end user. The system has been built around providing a Restful API, with the backed boilerplate code being generated from an OpenAPI spec. This guarantees the API contracts are kept, and ensures it is far easier to integrate.

4.2.4 Scalability

This project runs a prototype network running two organisations, with a node in each. This two node network is obviously capacity limited: there is a limit to how much data can be written to SideDB per transaction, and how long a transaction can take, meaning that a quality analysis job has a time limit. These limits mean that as a dataset gets larger, or the computation to be made about quality becomes more complicated, the number of records per transaction needs to be smaller.

The system supports manual modification of partition size as is, which goes some way to address these concerns. However, it also poses a big limitation: in order to discover the optimal partition size, users must do their own profiling of the system. This functionality should be included in a more production ready system, as it is unreasonable to expect users to be able to determine partition sizes accurately and quickly. Such an extension would run a predefined workload to find a benchmark performance value, which could be used to properly tune the system for performance.

Scalability is also constrained by how many nodes operate in the network. With the current two node setup, running in docker on a single host, there are definite limitations to throughput, as seen in the quantitative results section. Additional nodes could be added to the network (specifically, if it wasn't running in docker on a single host), to increase performance. However, the overhead of registering another org and actually setting up the infrastructure is not straightforward. To make this project truly scalable, additional work would need to be done to make provisioning new organisations seamless.

SideDB also poses a scalability concern, due to transaction duration and block size limits. This could be mitigated by not using SideDB, and rather storing data partitions on the contributors local node, and encrypting each partition with the consuming org's keys. In this case, the partition metadata and location could be written on chain, and processing could happen asynchronously off chain. This would relegate HLF to just coordinating the processing, rather than actually doing the processing. While it would reduce some of the guarantees around what code is executing on the partition provided, it creates opportunity for higher performance processing than just Golang in Docker, for example, by using Spark or similar methods.

4.2.5 Availability

The system built in this project is highly available, with room to become more high available. Organisations in HLF can run multiple nodes. In the implementation used in this project, each organisation runs two nodes, or peers. The endorsement policy required by the system states that at least one peer in each organisation must agree on a transaction. Since there are two peers per org, if one peer is unavailable in each organisation, the network still functions as normal.

In its current implementation, all peers run on the same host, just in different docker containers. For a more production ready implementation, running each peer on a separate host, and, for example in a cloud environment, running no two peers in the same org on hosts in the same availability zone, would go to lengths to ensure that the system is more highly available.

4.2.6 Testability

Testability is a strong point of the system. Since the system as a POC was designed to run entirely on a single host, in Docker, it is relatively portable. This means that no additional configuration needs to be added when compared to if the system was only running on a remote host, deployed to via CI/CD. Functionality is broken into sensible chunks, and all non HLF specific code is shared between both the local and chaincode specific builds of the application. These shared functions are unit tested for increased surety of their correctness.

4.2.7 Working with Hyperledger Fabric

HLF has been used at scale by companies as large as Maersk and much of it has been developed by IBM. Despite this, the documentation for developing for the system is lacking. For instance, the golang SDK is missing most of the documentation that the Node.JS contains, which makes development difficult. Even with hurdles like this cleared, it is still difficult. This system was built on top of the "build your first network" (BYFN) that the documentation encourages you to start with, which uses docker-compose to spawn a network and associated hosts as docker containers on your system. However, many modifications needed to be made out of the box to the configuration of the network to have non tutorial chaincode running. Debugging is difficult, as integral features such as sideDB were not available in the development network. When running against BYFN, chaincode would sometimes crash a node, and the whole network would need to be torn down and recreated to restore the functionality.

The lack of documentation continued to be an issue.

Despite these issues, HLF was not chosen by chance: when this project was started it presented the most viable way to construct a system as described here. As time has passed, Hyperledger's Sawtooth project has gained more momentum than Fabric, notably being used as the backbone for Salesforce's data sharing platform. Sawtooth also has the benefit of being EVM compatible, making smart contract development potentially easier than that on Fabric. However, despite these advantages, it's capabilities around privately sharing data between entities in the network is underdeveloped. Put simply, at this stage, this project could not be implemented on Sawtooth.

Chapter 5

Conclusion & Future Work

This thesis aimed to prove the feasibility of implementing a hybrid distributed/decentralised system for analysing data quality. This thesis showed that

1. Such a system can be constructed and poses significant mitigations to the risks of exposure when analysing the quality of a third party dataset.
2. Performance of such a system when implemented on Hyperledger Fabric although feasibly usable on small datasets, is likely too slow for real world user

5.1 Future Work

Whilst this project proves the feasibility of such a system and implement it, some further work could be done:

5.1.1 Further Hybridisation

DQ analysis could be coordinated by a similar system as implemented here, but with compute and storage provided by an Apache Spark cluster, instead of the nodes running Fabric. This would leverage the benefits of the using the permissioned blockchain, but the computational power of a much more adequate

5.1.2 Implementation on a Public Blockchain

If rearchitected, it would be feasible to use a public blockchain system to coordinate tasks, and using encryption, store the data off chain and have only registered nodes be able to read their partition of the dataset. It would be interesting to compare performance, and the complexity of the implementation.

Bibliography

[BCFM09] Carlo Batini, Cinzia Cappiello, Chiara Francalanci, and Andrea Maurino. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)*, 41(3):16, 2009.

[Fan15] Wenfei Fan. Data quality: From theory to practice. *Acm Sigmod Record*, 44(3):7–18, 2015.

[Fou19] Ocean Protocol Foundation. Ocean protocol: A decentralized substrate for ai data & services technical whitepaper. 2019.

[GKCC14] Arthur Gervais, Ghassan O Karame, Vedran Capkun, and Srdjan Capkun. Is bitcoin a decentralized currency? *IEEE security & privacy*, 12(3):54–60, 2014.

[HS90] Stuart Haber and W Scott Stornetta. How to time-stamp a digital document. In *Conference on the Theory and Application of Cryptography*, pages 437–455. Springer, 1990.

[Ins18] Forbes Insight. The data differentiator how improving data quality improves business. Whitepaper, Forbes, 2018.

[Lev19] Matt Levine. The sec really doesn't like icos. Newspaper, October 2019.

[LK01] Mark Last and Abraham Kandel. Automated detection of outliers in real-world data. In *Proceedings of the second international conference on intelligent technologies*, pages 292–301, 2001.

[LSJ$^+$09] Inger Kristin Larsen, Milada Småstuen, Tom Børge Johannesen, Frøydis Langmark, Donald Maxwell Parkin, Freddie Bray, and Bjørn Møller. Data quality at the cancer registry of norway: an overview of comparability, completeness, validity and timeliness. *European journal of cancer*, 45(7):1218–1231, 2009.

[MCR$^+$16] Jorge Merino, Ismael Caballero, Bibiano Rivas, Manuel Serrano, and Mario Piattini. A data quality in use model for big data. *Future Generation Computer Systems*, 63:123–130, 2016.

[MMA12] S Moossavizadeh, Mehran Mohsenzadeh, and Nasrin Arshadi. A new approach to measure believability dimension of data quality. *Management Science Letters*, 2(7):2565–2570, 2012.

[Pec12] Morgen E Peck. The cryptoanarchists' answer to cash. *IEEE Spectrum*, 49(6):50–56, 2012.

[TB98] Giri Kumar Tayi and Donald P Ballou. Examining data quality. *Communications of the ACM*, 41(2):54–57, 1998.

[Ver99] Howard Veregin. Data quality parameters. *Geographical information systems*, 1:177–189, 1999.

[Vra17] Harald Vranken. Sustainability of bitcoin and blockchains. *Current opinion in environmental sustainability*, 28:1–9, 2017.

[W$^+$14] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[WW96] Bo Pedersen Weidema and Marianne Suhr Wesnaes. Data quality management for life cycle inventories—an example of using data quality indicators. *Journal of cleaner production*, 4(3-4):167–174, 1996.

[XPZ$^+$16] Xiwei Xu, Cesare Pautasso, Liming Zhu, Vincent Gramoli, Alexander Ponomarev, An Binh Tran, and Shiping Chen. The blockchain as a software connector. In *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 182–191. IEEE, 2016.