Collaborative Visual Analytics Using Blockchain

Darius Coelho[1], Rubin Trailor[2], Daniel Sill[3], Sophie Engle[2], Alark Joshi[2], Serge Mankovskii[3], Maria Velez-Rojas[3], Steven Greenspan[3], and Klaus Mueller[1]

[1] Stony Brook University, Stony Brook NY 11790, USA
[2] University of San Fransisco, San Francisco, CA 94117, USA
[3] Independent Researcher**

Abstract. Blockchain, a decentralized, distributed and encrypted ledger, was created to eliminate the need for a central trusted entity. Blockchains provide users with a secure, trusted, auditable, and immutable record of transactions and are applicable to systems that require a trustworthy record of information. Our work explores the use of blockchain in collaborative visual analytics systems where users share and store a record of the visual analysis of some data. We built *Share.va*, a framework that allows users to store and share the states of visual analytics dashboards through a blockchain. We apply *Share.va* to an existing visual analytics dashboard and conduct a pilot study to understand the effectiveness and limitations of blockchain in collaborative visual analytics.

Keywords: Collaborative Visualization · Provenance · Blockchain.

1 Introduction

In the collaborative analysis of data, analysts need to share data analysis tools, visualizations, and maintain a shareable history of their explorations. Such an auditable record is essential to shared decision making [5]. At the same time, to ensure trust among collaborators, their content must remain private and secure when being shared [8]. For example, in the collaborative analysis of medical records, multiple doctors and medical technicians may need to analyze a patient's sensitive data in order to diagnose the patient. Here, the patient needs to ensure that only trusted doctors and technicians analyze and share the data. The inherent properties of blockchain make it a viable candidate to support such scenarios. It's encryption based distributed consensus mechanism ensures that the blockchain only logs trusted updates and the log is authentic and unchangeable.

In our work, we aimed to explore the viability of a blockchain back-end for collaborative visual analytics (C-VA) systems. Most literature on C-VA systems do not address the need for security and trust among collaborators. Using blockchain can address such issues, however, CV-A systems are highly interactive and require low latency communication and can be affected by the established performance issues with blockchain [10]. Our work explores the limits of blockchain in such systems.

** This research was conducted while the authors were at CA Technologies, USA

We built *Share.va*; a framework that uses blockchain to log changes and insights derived through visual analytics dashboards. We also designed a visual metaphor to represent each collaboratively created state stored on the blockchain thus aiding the navigation of the blockchain. We demonstrate and evaluate the capabilities of our framework through a case study. Here, we retrofit an existing visual analytics dashboard used by social scientists with *Share.va*. We then observed the scientists collaboratively analyze their data both synchronously and asynchronously as well as record the analysis on the blockchain. By observing the scientists collaborate with the dashboard, we learned about the limitations of blockchain during real-time collaboration and how users worked around them.

2 Background

Blockchain was originally introduced to support the bitcoin cryptocurrency [7], but its ability to establish trust in a decentralized environment has led to many applications in the finance, health, and education sectors [3]. As explained by Swan [9], it is fundamentally a distributed ledger database that shares and synchronizes all records of digital transactions across multiple entities. Transactions are recorded into a block and added in a sequential manner so every block contains a hash to the previous block. This means every ledger entry is re-traceable across the full history of the blockchain. Before a transaction is added to the blockchain, all relevant parties need to agree that the block is valid. Once there is agreement on the validity of a block and it is recorded, it cannot be changed. These properties make blockchain a suitable technology that can be applied to any system that requires a trustworthy record of information [11].

Prior to implementing our framework, we first ensured that blockchain is suitable for our application by addressing the five key questions to determine the appropriateness of blockchain use [3]. First, users are required to have a shared database of datasets and insights in collaborative visual analytics. Next, they must be able to write to this database to add transformed data or insights. Additionally, in scenarios such as the medical analysis example, the patients need to trust that their information was only reviewed by approved doctors and other related individuals. In such cases, blockchain's disintermediation protocol ensures trust among collaborators. It does so by removing the need to trust a central entity as in traditional databases managed on a central server. Finally, the blockchain's ability to store an ordered, immutable and auditable record of the analysis inherently provides provenance of the analysis.

3 The Share.va Framework

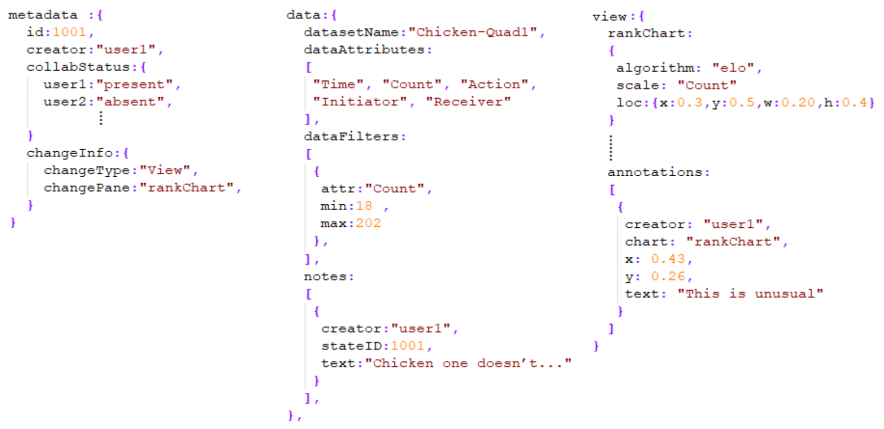
Share.va is designed for web-based collaborative visual analytics dashboards and addresses the general requirements of or most C-VA systems[4]. It enables users to share interactions, data filters, annotations, and notes. It also allows users to make use of heterogeneous dashboards, i.e. dashboards that have different visual configurations but analyze the same data. Every user interaction leads to a change in the dashboard state which our framework logs to a back-end blockchain thus storing the provenance of an analysis. The blockchain can later be polled for prior dashboard states.

3.1 Representing Dashboard States

To represent a dashboard's shared state, we designed a specific data structure that supports the use of heterogeneous dashboards as well as sensemaking tools. The structure is concise due to the size limitation of blockchain and consists of three main subparts - data, view, and metadata. As we target web-based dashboards, the state is stored as a JSON with each subpart being stored as a field in this JSON. Each subpart is explained below:

Data: The data subpart of the state stores the data (or a pointer to it), and any functions that are used to manipulate the data along with the function parameters. These functions are often triggered by user interactions. They can be as simple as a list of filters over each data attribute or they could be more complex such as a clustering or layout algorithm. Changing a function's parameters can trigger a change in the visualizations, for example it may cause points to be highlighted or repositioned. Additionally, user-generated notes are stored in this subpart. Notes are the insights collaborators have gained at a particular dashboard state during their analysis. Although collaborators can have different dashboard views (with heterogeneous dashboards) the data insights would be the same as the functions applied to the data are always shared. Notes are essential to asynchronous collaboration as they communicate a user's thought process to his or her collaborators.

View: The view subpart stores information about the visual properties of the dashboard. The view is stored as a list of visual components, their locations



```
metadata :{
  id:1001,
  creator:"user1",
  collabStatus:{
    user1:"present",
    user2:"absent",
          ⋮
  }
  changeInfo:{
    changeType:"View",
    changePane:"rankChart",
  }
}
```

(a) Meta-data

```
data:{
  datasetName:"Chicken-Quad1",
  dataAttributes:
  [
   "Time", "Count", "Action",
   "Initiator", "Receiver"
  ],
  dataFilters:
  [
   {
    attr:"Count",
    min:18 ,
    max:202
   },
  ],
  notes:
  [
   {
    creator:"user1",
    stateID:1001,
    text:"Chicken one doesn't..."
   }
  ],
},
```

(b) Data

```
view:{
  rankChart:
  {
   algorithm: "elo",
   scale: "Count"
   loc:{x:0.3,y:0.5,w:0.20,h:0.4}
  }
  ⋮
  annotations:
  [
   {
    creator: "user1",
    chart: "rankChart",
    x: 0.43,
    y: 0.26,
    text: "This is unusual"
   }
  ]
}
```

(c) View

Fig. 1: An example of the three dashboard state components from our case study.

in the dashboard, and annotations associated with them. The annotations, like the notes, are used for sensemaking but are more specific to the chart and often point to particular data points. Additionally, transforms that are chart specific are not essential for data sharing can also be maintained as part of the view, for example zoom levels or axis scales. When using heterogeneous or personalized views, the user can choose to ignore this view subpart of a shared state and use a personalized view. For example, a visual component that shows an overview of the data may either use a scatter plot or a parallel coordinate plot. By ignoring the view component one user could use a parallel coordinate plot and the other a scatter plot to visualize the same data, they could even use the scatter plot at different zoom levels.

Meta-data: While the data and view subparts reflect changes made to the content, an essential requirement for collaborative sense-making is knowing who made that change and when it was made. The meta-data section of the state is used to store this information. Additional information about the state is also maintained here such as which collaborators were present during the creation of the state and the type of change that led to the creation of the state (a data or view change). This meta-data is primarily used by history glyphs we designed to help users navigate states on the blockchain. The glyph is shown in figure 2 It is based on a dining table metaphor. A wireframe of the dashboard in the centre is the "dining table" and the arc segments along the circumference are the "seats" around this table. The arcs represent collaborators that were present during a state's creation and the circle behind the wireframe is colored and connected to the creator's arc. Panels in the wire-frame are highlighted to indicate in which panel a change occurred

An example of the state of a dashboard used in our case study is shown in figure 1. Here, the meta-data section (figure 1a) of the state stores the state's identifier in the *id* field, its creator in the *creator* field, the status of collaborators (present or absent) during the state's creation in the *collabStatus* field, and the type of change the state caused and the visualization it affected in *changeInfo* field. The data section (figure 1b) of the state stores a pointer to the dataset being analyzed in the *datasetName* field. The list of attributes in the dataset, the filters applied to them, and user notes are also stored in the *dataAttributes*, *dataFilters*, and *notes* fields of the data section. Adding or modifying a filter will cause a data change that is reflected across all visualizations. The view section (figure 1c) of the state has a list of fields that store the location and visual properties of the dashboard's visual elements. For example, the *rankingChart* field stores the normalized location and size of a line chart used to represent ranks of animals in the *loc* sub-field. It also has an *algorithm* sub-field which allows users to select a ranking algorithm and a *scale* sub-field to select the kind of scale, in this dashboard the user could choose between item count and clock time. Since the view section can be ignored by collaborators, they could use different ranking algorithms and scales for this chart while collaborating with each other. The view section also stores annotations made to the dashboard in the *annotations* field.

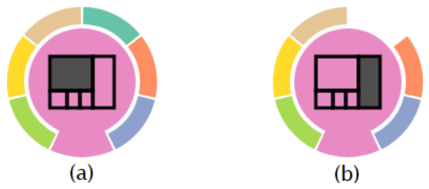


Fig. 2: The history glyph which help users navigate states on the blockchain. In (a) all seven collaborators were present when a change was made to the top left panel (dark gray highlight) while in (b) one collaborators was absent (green segment) when a change was made in the panel on the right (dark gray highlight).

Blockchain Implementation Due to the complexity of our framework's data structure as well as other constraints and criteria, it was necessary to seek out the appropriate blockchain implementation from the many emerging options. We reviewed the available blockchain implementations (MultiChain, OpenChain and ScaleChain) to find one that best aligned with our needs. It should be noted that at the time of evaluation, promising candidates such as Hyperledger were rejected based on their maturity level but these may now be equally suitable for use in the framework. We had two primary design goals: (1) store data in a JSON object and (2) process client requests asynchronously.

We selected MultiChain, an open source system that stood out since it fulfilled our requirements as well as allowed us to store the largest size of data in the blocks. It provides a stream feature that allows raw text data to be written to the blockchain. This gave us the ability to encode our application data in JSON format, while still leveraging the auditable features of the blockchain. Additionally, MultiChain provides a permission management system that allows admins to grant read or write permissions to the blockchain. In our use case, the creator of a collaborative session is the admin who can then add collaborators to that session. Based on the permission a collaborator has, he or she can either contribute (write) to an analysis or only review (read) an analysis. The assignment of permissions is tracked by the blockchain ledger, much like a coin would be. This means that the assignment of read and write permissions would benefit from being traceable and auditable as well. A downside is that MultiChain uses JSON-RPC which requires synchronous communication with a client thereby making the client unresponsive until MultiChain replies to it. To overcome this issue, we implemented a Java program that acts as an interface between MultiChain and the client. Here we use the REST architecture to communicate with the client asynchronously while communicating MultiChain synchronously.

Writing to the Blockchain Having selected an appropriate blockchain implementation, we explored the possible methods of using it to store and share states among collaborators. In *Share.va*, a dashboard state is stored as a blockchain transaction and we employed two methods of storing this state - appending states and updating states - which are explained as follows:

Appending States: Appending a state is straightforward. For every user action, the current state of the user's dashboard is generated; this is essentially the entire JSON state described previously. This state is then sent to the blockchain and stored as a single transaction. As users generate annotations and notes, the number of items stored in the annotation and notes fields of the state increases. This increases the packet size of items being stored to the blockchain which can degrade the overall performance.

Updating States: Updating a state is a more involved process; for every user action, only the fields that have changed in the JSON are dispatched. The blockchain, however, is append-only and so does not allow stored states to be modified. Therefore, the latest state is duplicated and the changes are merged. This modified state is then appended to the blockchain.

Both methods are supported in the current implementation of *Share.va*. However, we recommend using the update method when one collaborator's interface is lagging. This causes less disruption for collaborators whose interfaces are not lagging. For example, if a user's interface is lagging, he would essentially be working on an older state of the dashboard. Now if he is is five updates behind his collaborators, then using the append state method would cause collaborators to receive a state that has the five changes undone and a new change added which is effectively six changes. On the other hand, when using the update state method, the collaborators would receive just a single change that may not make sense to the collaborators since it was based on an older version of the state. While both methods cause disruptions, using updates is not as jarring as appending states with a lagging user. This is because with the update method the collaborators would always receive a single incorrect change while the append method can cause the collaborators to receive multiple incorrect changes.

4 Blockchain Performance

Before conducting our case study, we tested the response time of the blockchain while varying the number of users. The results are shown in table 1. These response times are much slower than traditional databases and can cause problems when the frequency of user interactions is very high. Specifically, states can appear to be rejected when the blockchain receives too many requests before it finishes creating a new block. Consider the case of one or more users performing two or more interactions in under one second. As shown in the update method, it takes the blockchain 1.35 seconds at the very least to create a block. Thus only the first interaction in the one second period is logged to the blockchain with the others timing out. Such high response times are unacceptable in highly interactive environments such as multiplayer games, but as we show in our case study visual analytics tasks have a significantly higher threshold. Additionally, asynchronous collaboration does not suffer from these issues as requests to the blockchain would be infrequent.

Table 1: The time taken to read and write (with the append and update methods) a block to the blockchain as the number of collaborators increase.

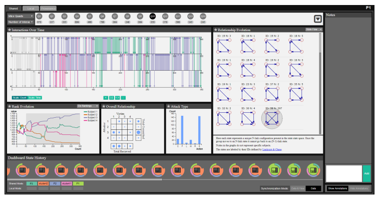
No. of Users	1	2	3	4	5	6	7	8	9
Read Time (sec)	0.85	0.86	1.21	1.41	1.58	1.75	2.43	13.43	28.23
Append Time (sec)	0.58	0.58	0.59	0.59	0.63	0.63	1.78	7.72	20.49
Update Time (sec)	1.35	1.35	1.35	1.35	1.45	1.47	3.06	17.89	54.12

5 Case Study

To demonstrate and evaluate Share.va, we tested it on a real-world application. We took an existing dashboard used by social scientists [2] and augmented it with Share.va. The Share.va enabled dashboard, shown in figure 3, allowed scientists to load one of many datasets that contain an interaction record of an animal group and analyze it collaboratively.

5.1 Procedure

In this study, two social science professors (P1 and P2) and their research assistant (A1) who are familiar with the original dashboard used the Share.va enabled version of the dashboard. They collaboratively explored a new unstudied dataset acquired by them. The study consisted of three sessions. In the first session participants P1 and P2 performed a joint exploration of the datasets. The second session was an individual analysis of the data by A1 that was unrestricted. This was a deeper analysis than in the first session. In the final session, P1 and P2 jointly reviewed the dashboard states and insights generated by A1 during session 2. The sessions were conducted over a period of one week with the participants being allowed to use the interface independently during the course of the week. The first session lasted for 28 minutes and the third for 42 minutes and an author moderated the sessions and provided minimal help to the users. The second session was an independent analysis in which the participant could take as much time as she wanted. This second session was not moderated but A1 was able to contact the author for technical support when needed and her interactions could be reviewed through the blockchain. Prior to the first session, participants were given a demo of the Share.va enabled dashboard. The demo showed the participants how to use the various features that support collaboration. After the demo the participants were asked to perform three practice tasks: change a visual representation, add an annotation and add a note. During the two sessions with P1 and P2 we recorded their screens and verbal communications. Additionally, we had a post session interview with the participants and asked them about their experience. All participants used desktop computers running with 27 inch screens at a resolution of 1920 by 1080 and ran our tool on a browser of their choice. Verbal communication was supported by a commercial VoIP service which is not part of our current system.


(a) P1's dashboard

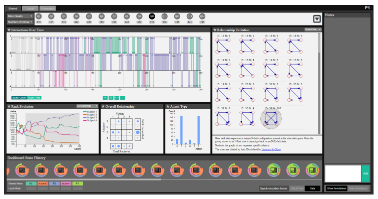

(b) P2's dashboard

Fig. 3: A snapshot of our participants' dashboards during the case study. Here the collaborators are synchronizing the data but not the view, this can be observed in the right panel where (a) shows P1 using the compressed state space and (b) using the full state space. The single item selected in (a) (gray circular highlight) maps to 9 items selected in (b).

5.2 Observations

During the first and third sessions which were synchronous collaborations between P1 and P2, we observed that right before or during an interaction with the dashboard they always verbally indicated which dashboard item they were looking at or interacting with. They did this because their mouse pointer's position was not shared due to the high-latency of the blockchain. We also observed that during the analysis, P1 and P2 would take turns performing a set of interactions and they would often have small discussions between interactions. The turn taking behavior was primarily due to each participant hypothesizing an insight and trying to verify it through a set of interactions while the other participant observed and discussed the changes in the visualization. The small discussions between interactions actually complimented the blockchain's high latency as they gave the blockchain enough time to record an interaction before processing the next request. Finally, we observed that P1 and P2 did not make use of notes they made handwritten notes instead. Additionally, they only used annotations when they could not verbally indicate an item's position.

In the second session, A1 performed an asynchronous analysis which was later reviewed by P1 and P2 in the third session. As A1 interacted with the dashboard alone, she did not face any latency issues. All her interactions were recorded and she did not have to undo or redo any part of her analysis. During her analysis, she applied multiple filters to the data and annotated visualizations with her observations at every stage of her analysis. She also made notes for P1 and P2. During session three, P1 and P2 spent the first half of the session reviewing A1's analysis. They first located the start of her analysis recording on the blockchain using the history glyphs. They then reviewed each of her actions by sequentially loading each state she created. Again, here one participant loaded the states while the other just viewed the dashboard and discussed A1 findings. Thus the blockchain never received two or more concurrent read requests. Also, since they spent time discussing each state, the blockchain had enough time to process each read request and was never overloaded.

6 Discussion

6.1 Latency

The main issue with blockchain is that it is slow due to its computational requirements. Our performance test showed that the blockchain could only allow a maximum of six users without rendering the system unfit for synchronous collaboration. The fastest time to read or write from the blockchain was just above half a second but on average it was longer. Most groupware applications would suffer from such a high response time. However, it might be acceptable for C-VA. Studies in information visualization have shown that most human cognitive tasks take one second or less [1]. On the other hand, Liu et al. [6] show that a 500ms delay in an interactive visualization has a significant effect on mouse movements and brushing but not in other tasks such as zooming and panning. Delays also cause users to change their interaction strategies. Our study participants adapted to delays by resorting to a turn taking behavior as well as filling time between interactions with discussions. The delays in a blockchain with a greater number of users are much larger but such delays have not been tested in C-VA. We hypothesize that users would also change strategies here and we plan to test this in future work. However, when using *Share.va* asynchronously, the study participants did not run into any performance related issues. This leads us to believe that blockchain might be more suitable for collaborative data analyses that are asynchronous or turn-based such as the case where a doctor and lab technician take turns analyzing a patient's data.

6.2 Space Limitations

The blockchain's storage constraint is another issue we had to deal with when designing *Share.va*. Most implementations support the storage of a few bytes of data which is insufficient for most C-VA systems. Today data being analyzed can exceed hundreds of gigabytes thus making it is impossible to store on a blockchain. In our case study, every user had a local copy of the dataset and only data transformation functions were shared and stored on the blockchain. Thus the security and authenticity of the dataset itself is not guaranteed by the blockchain; instead users need to rely on a third-party to secure the dataset.

6.3 Linearity

Another challenge with applying blockchain to C-VA is that it stores blocks in a linear manner and it does not support branching. On the other hand, in C-VA at a certain point analysts may choose to branch off into different paths of analysis. Due to the linear nature of blockchain only one branch can actively be shared and appended to the blockchain in real time. To support branching on a single chain, additional branches would need to be temporarily cached and appended to the blockchain sequentially at a later stage. This effectively makes the other branches asynchronous analyses. Thus blockchain is more suitable for collaborative analyses without branching or asynchronous turn-based collaboration.

7 Conclusion

In this paper we presented Share.va: a proof-of-concept framework that supports secure C-VA using a blockchain back-end. To the best of our knowledge, this is the first system to apply blockchain in this fashion. We tested our framework in a real world scenario by having a group of social scientists use our system to analyze their data collaboratively. While this study yielded insight in how our method and system is used in practice, it also revealed some of the system's limitations, and of blockchain technology applied for the purpose of C-VA overall. We found that while our use of blockchain in this way was viable for smaller numbers of users or for asynchronous collaboration.

Acknowledgments This work was supported in part by NSF I/UCRC 1650499: Center for Visual and Decision Informatics (CVDI) Site at SUNY Stony Brook, CA Technologies, a Broadcom Company, USA, NSF grant IIS 1527200, and MSIT (Ministry of Science and ICT), Korea, under the SUNY Korea's ICT Consilience Creative program (IITP-2020-2011-1-00783) supervised by the IITP.

References

1. Card, S.K., Mackinlay, J.D., Shneiderman, B. (eds.): Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
2. Coelho, D., Chase, I., Mueller, K.: Peckvis: A visual analytics tool to analyze dominance hierarchies in small groups. IEEE Transactions on Visualization and Computer Graphics **26**(4), 1650–1660 (2020)
3. Gatteschi, V., Lamberti, F., Demartini, C., Pranteda, C., Santamaria, V.: To blockchain or not to blockchain: That is the question. IT Professional **20**(2), 62–74 (Mar/Apr 2018)
4. Isenberg, P., Elmqvist, N., Scholtz, J., Cernea, D., Ma, K.L., Hagen, H.: Collaborative visualization: Definition, challenges, and research agenda. Information Visualization **10**(4), 310–326 (2011)
5. Kane, B.T., Toussaint, P.J., Luz, S.: Shared decision making needs a communication record. In: Proceedings of the 2013 Conference on Computer Supported Cooperative Work. pp. 79–90. CSCW '13, ACM, New York, NY, USA (2013)
6. Liu, Z., Heer, J.: The effects of interactive latency on exploratory visual analysis. IEEE Transactions on Visualization and Computer Graphics **20**(12), 2122–2131 (Dec 2014)
7. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
8. Shen, H., Dewan, P.: Access control for collaborative environments. In: Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work. pp. 51–58. CSCW '92, ACM, New York, NY, USA (1992)
9. Swan, M.: Blockchain: Blueprint for a new economy. O'Reilly Media, Inc. (2015)
10. Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C., Rimba, P.: A taxonomy of blockchain-based systems for architecture design. In: IEEE International Conference on Software Architecture. pp. 243–252 (April 2017)
11. Zyskind, G., Nathan, O., Pentland, A..: Decentralizing privacy: Using blockchain to protect personal data. In: 2015 IEEE Security and Privacy Workshops. pp. 180–184 (May 2015)