Management of provenance metadata for large scientific experiments based on the distributed consensus algorithm*

A. Demichev and A. Kryukov

Skobeltsyn Institute of Nuclear Physics, Lomonosov Moscow State University

Abstract. We suggest a new approach to management of provenance metadata for data generated by large scientific experiments. Although a number of projects have been fulfilled in recent years to create management systems for such metadata, but the vast majority of implemented solutions is centralized. This is inappropriate for using by organizationally unrelated or loosely coupled communities of researchers. In this paper we propose to create a distributed registry of provenance metadata on the basis of the newly developed hashgraph consensus algorithm, which has several advantages and does not require a lot of resources for its maintaining.

Keywords: Provenance metadata · Consensus algorithms · Hashgraphs.

1 Introduction

Metadata describing data, provide context and are vital for the accurate interpretation and use of data by both humans and machines. One of the most important types of metadata is provenance (lineage, pedigree) metadata [1]. Provenance from the point of view of computer science is a meta-information related to the history of obtaining and processing data, starting from the source. Metadata of this type is intended for tracking the stages at which data were obtained, determining their origin, ensuring their correct storage and reproduction, interpreting and confirming the correctness of the scientific results obtained on their basis. Thus, provenance metadata (PMD) are key for organizing a research workflow to obtain reliable results. The need for PMD is especially topical when large data are jointly processed by several research teams, which is currently a very common practice in many scientific areas. This requires a wide and intensive exchange of data and programs for their processing and analysis (the open science model [2]) during long periods of time when data and algorithms of their processing may transform.

This work is aimed at developing an approach and principles for the creation, storage and management of provenance metadata for data generated by large-scale scientific experiments. A vivid example of large installations for which this

* This work was funded by the Russian Science Foundation (grant No 18-11-00075).

is very important is the Large Hadron Collider (LHC, CERN, Geneva), the time of active work of which and, accordingly, the generation of large scientific data, is several tens of years, and the processing time of the data will be at least twice as much. Without detailed and correct provenance metadata, comparing the results obtained with an interval, for example, in a few years, will be simply impossible. Although a number of projects have been implemented in recent years to create systems for the support and management of metadata, including the data provenance, but the vast majority of implemented solutions are centralized [3, 4], which is inadequate for the use in distributed environments, the open science model and the possibility of using metadata by organizationally unrelated or loosely related research communities. The use of centralized solutions in a distributed environment that includes different administrative domains is always associated with the problems of organizing the management of the central service, as well as with ensuring the trust in such a service from the side of participants of the distributed system. In addition, any central service is a bottleneck and a point of failure of the system. Federated databases (FDB) have too complex structure to use them in the case of metadata. In particular, to search for information in FDB, it is needed a general rather sophisticated catalog (in this case, metadata for metadata). In this case also, there is the problem of ensuring the trust of one administrative domain to a part of the FDB managed by another administrative domain. Hierarchical systems, in addition to having common disadvantages of centralized systems, do not correspond to the single-layered structure of metadata. Besides, these approaches do not have built-in mechanisms for monitoring the integrity of metadata and their traceability. On the other hand, in recent years, distributed registries based on blockchain technology [5] have become very popular in various applied areas thanks to a number of important advantages. In the most recent time, on the basis of the blockchains, there have also been developments for the PMD management systems [6, 7]. Currently, these developments are in the testing phase, but the analysis of the proposed solutions shows that they are designed for cloud storage environments, are quite heavyweight and resource-intensive. The latter is related to the peculiarities of realization of the distributed registries as blockchains (necessity for block mining). This makes doubtful the prospects for the successful use of these solutions for the storage and management of provenance metadata generated by large scientific experiments in distributed environments.

To solve this problem, in this paper we propose to use a new approach to the creation of distributed PMD registries, with the abandonment of the process of the block mining which is the resource-consuming and excessive in this area. The point is that the main property of the blockchain technology, on which its numerous applications are based, is that it allows reaching consensus among participants of a distributed network on the content and chronological order of transactions without the involvement of any third-party (central) trusted authorities or services. To reach a consensus is especially important for metadata of data obtained as a result of processing and analysis of primary experimental data. For brevity, we will call all such data secondary (although some of them can be obtained as a result of several processing steps). Indeed, the providers of primary data are a very limited set of experimental installations, and the corresponding provenance metadata is generated automatically. The security and integrity of the metadata database, including centralized one, for such providers can be achieved by standard methods (accounts with the appropriate rights, cryptographic keys, etc.). The situation with providers of secondary data (that is, researchers performing data processing and analysis) is significantly different. The number of such providers can be quite large and dynamically changing. Therefore, either the overhead of managing the access rights to the metadata base will be very large in the case of using the standard methods, or serious problems with accidental or malicious distortion of provenance metadata can occur. An example of motivation for intentional distortion of the provenance metadata may be the desire to get priority (for example, to get a fictitious priority in obtaining valuable results from the physical analysis of experimental data). The distributed registries based on consensus algorithms seems to be most suitable for solution of the PMD security and integrity issues in distributed computer systems. Besides the blockchain consensus mechanism there are other variants of consensus algorithms (see Section 2). We propose to use the newly developed algorithm based on hashgraphs [8] for constructing a distributed PMD register, which has a number of advantages and does not require large resources for its maintenance. The paper presents (Section 3) the general architecture and principles of operation of the provenance metadata management system based on the hashgraph algorithm intended for distributed systems for processing and analyzing large scientific data.

2 Selecting a consensus algorithm

In this section, the most popular and widely used consensus algorithms are briefly discussed and compared with the recently proposed algorithm based on hashgraphs.

2.1 The Paxos and Raft algorithms

One of the first and most well known consensus algorithms is the Paxos algorithm [9]. This algorithm is not designed to work in distributed systems with possible Byzantine errors. It is very difficult for understanding and implementing. In addition, Paxos uses an approach in which each consensus member interacts with each, so the complexity of the decision is $O(n^2)$, where n is the number of the members. As a result, practical implementations have little to do with Paxos. Each implementation begins with Paxos, detects difficulties in its implementation, and then significantly changes the architecture. It takes a lot of time and leads to errors. Because of these problems, Paxos is not a good choice for building real systems.

The Raft algorithm [10] realizes the consensus by choosing a single leader, giving it full responsibility for managing the replicated log. The leader accepts requests from clients, copies them to other servers, and tells the rest of the servers when it is safe to use log entries in their replicated state machines. The idea of having a special leader simplifies the management of the replicated log. If the leader for some reason stops working, the procedure for selecting a new leader begins. However, Raft is also not designed to work in distributed systems in which a Byzantine type of error is possible (malicious distortion of information by nodes). In addition, Raft systems are vulnerable to DoS-attacks on the current leader, which can lead to large delays in the system.

2.2 Blockchain consensus

Systems based on blockchain technology [5] have no shortcomings of Raft-type algorithms because they are not based on the choice of leader. However, network members never know for sure when consensus is reached. They are sure only that there is a probability of reaching a consensus, which grows with time elapsing. If two blocks are created simultaneously, then the chain forks until the community can agree on which branch will expand further. If blocks are added slowly, the community can always add blocks to a longer branch, and eventually the other branch will stop growing and can be pruned because it is "obsolete". This leads to inefficiency in the sense that some properly mined blocks are discarded. This also means that it is necessary to slow down the speed of forming blocks, so that the community can cut off branches more quickly than new branches sprout. This is the goal of proof-of-work (PoW): the requirement that the miners solve difficult computational problems for the formation of the block, provides a sufficiently long delay between the mining events. Another option is the proof-of-expired-time algorithm based on reliable hardware chips that provide a delay for a fairly long period, as if they were performing calculations in the spirit of PoW. However, this requires that all participants trust the company that produces such a chip.

There are other algorithms [11] for providing the Byzantine consensus, which avoid some of the problems inherent in blockchain technology. However, these algorithms involve the exchange of multiple messages between consensus participants. For reaching a consensus by n members on one question with a binary response, the system may need to send $O(n)$, $O(n^2)$ or even $O(n^3)$ messages. Then the algorithm for making a binary decision should be extended to the more general consensus on the order of transactions, which further increases the traffic.

2.3 The hashgraph algorithm

The hashgraph algorithm [8] for achieving distributed consensus does not use a leader and is resistant to DoS attacks on a limited subset of consensus members. The hashgraph is similar to the chain of blocks in the blockchain, which, however, is constantly branched, without any trimming. At the same time, none of the blocks ("events" in the terminology of the hashgraph) will never become obsolete, and each participant is allowed to form many new events per second, without overhead costs of the PoW type.

The hashgraph consensus algorithm uses the gossip protocol [12]. This means that one member of the network chooses at random another participant and informs him of all the information that he knows at the moment. Then both participants repeatedly do this by choosing other recipients, and all the other participants do the same. Thus, if one participant learns about new information, the information will exponentially spread to the entire community, until each participant learns about it. The history of any gossip protocol can be depicted as an oriented graph in which each vertex represents an event of the gossiping. An event can also contain a payload in the form of a record of any transactions that the participant wants to create at this point, and possibly a timestamp that corresponds to the creation time of the event, as claimed by the event's creator. Other ancestors of this event are connected with it by a set of cryptographic hashes hooking one another. Spreading the hashgraph by means of the gossip protocol gives participants a lot of information. If a new transaction is placed in the payload of an event, it quickly spreads to all participants until each member learns about it.

As the hashgraph grows, different participants may have slightly different subsets of new events, but the process quickly converges so that all participants have the same events in the overwhelmingly larger part of the hashgraph. In addition, if two participants are aware of an event, they are guaranteed to know about all of its ancestors with matching edges of the graph for these ancestors. This allows one to run algorithms for achieving Byzantine consensus locally, without exchanging by additional messages.

Of course, it is not enough to make sure that every participant knows every event (this is achieved with the help of the gossip protocol and the mesh of event hashes hooking each other). It is important to make agree the linear ordering of events and, consequently, of transactions recorded inside the events (regardless of what is stated in the participant timestamps, in fact some of them may be erroneous or malicious as it was explained in the Introduction). When running the hashgraph algorithm, no votes are sent across the network, because all voting is virtual and is based on local calculations [8]. This algorithm works, provided that the number of malicious participants is less than 1/3 of the total number of participants (in accordance with the general theorem on the Byzantine Generals' Problem [13]).

3 The architecture and basic principles of provenance metadata management based on hashgraphs

For some types of metadata (for example, bibliographic data of books), the temporal order of their records is insignificant. In the case of provenance metadata, this is not the case, since they describe the evolution of data states, which obviously can depend significantly on the temporal order of operations with these data. Within the framework of large scientific collaborations and, especially, in the open-science model, such data operations can be performed by many users of a distributed storage system independently from each other. The subject of a consensus that all users of a distributed system must achieve is the order of the PMD records. This ensures that the PMD database operates as a replicated state machine, which provides the correctness and reliability of the access service to this database. In the approach proposed in this paper for the management of the PMD, this problem is solved by a consensus algorithm based on hashgraphs.

Users of the PMD management system are divided into two types: creators (perform operations with data and make corresponding records of provenance metadata) and ordinary users (only request the PMD for reading). Creators can be subdivided into two types: sources of primary data (download data generated by experimental facilities, Monte Carlo simulators, etc., in the repository; create the corresponding PMD in the form of transaction records in the events of the hashgraph); data handlers (modify the existing data in the repository, primary or previously processed, and load the results back into the repository; create the corresponding PMD in the form of transaction records in the events of the hashgraph). To understand the operation of the system as a whole, we subdivided creators into the two above types, although it should be noted that, from the point of view of the PMD management, their functionality is very similar: the main thing is that the both types of creators produce events, that is nodes of the hashgraph with corresponding transaction records. However, it is worth mentioning that the number of the first type creators is limited and rather small so that security and integrity of the metadata database for such providers can be, generally speaking, achieved even for centralized architecture by usual methods such as providing appropriate access rights, cryptographic keys, etc. The number of creators of the second type can be quite large and dynamically changing. Therefore, the overhead of managing the access rights to the metadata base by the usual methods may be unreasonably large.

The architecture of the provenance metadata management system (PMD MS) based on hashgraph technology is shown in Figure 1. The PMD MS consists of the software components of creators and PMD request nodes. The latter respond to requests by providing users with PMD, selected in accordance with the filter specified in the query. Creators must register in the distributed consensus network before starting their work so that their Internet address be known to other participants. To do this, they only need to provide their address to one of the participants, the further distribution of the address occurs via the gossip protocol. Users should only get the right to read data from the PMD database. As a client software they can use a usual browser.

Consideration of data management methods (including uploading or downloading data to/from distributed storage, checking their correctness, etc.) is beyond the scope of this paper. For the purposes of this work, it is only important that a creator, simultaneously with uploading a piece of data (file) to the distributed storage, creates a new event (the hashgraph node) and updates the hashgraph entry in his Hashgraph DB instance (graph database). A special REST API will be created, through which all data management and PMD col-

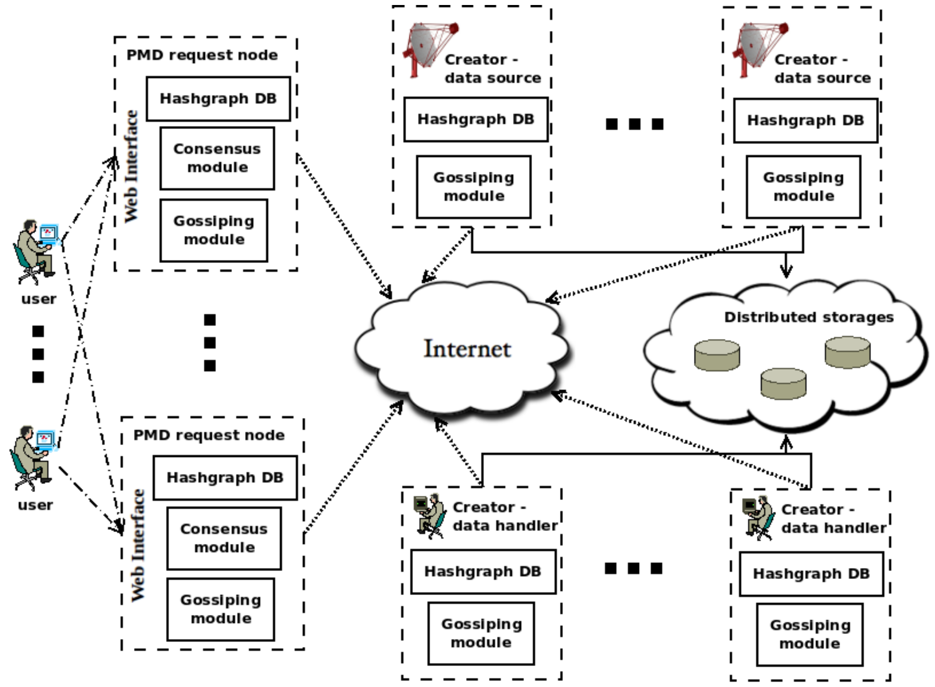

Fig. 1. The architecture of the provenance metadata management system based on hashgraph technology. Solid arrows designate lines for data loading (possibly within local networks or leased lines); dotted arrows correspond to the hashgraph gossiping; dash-dotted arrows stand for user requests for PMD (read-only mode).

lection will be performed. The hashgraph node contains a payload in the form of a PMD record corresponding to the transaction committed by the creator. PMD records have a key-value structure, with the keys corresponding to different data specifications (for example, upload date, source, calibration parameters of the experimental setup, etc.). To implement such records in the PMD management system, the JSON format is used. When the primary data is loaded, the PMD will be automatically generated, and when loading secondary PMD data, it will be generated semi-automatically with the creator's participation. The gossiping module sends this information to other consensus members. Thanks to this, the hashgraphs for both creators and the nodes of the PMD requests are identical up to the few last events (which have not yet spread to all the nodes of the system). In accordance with the hashgraph algorithm [8], the PMD request nodes compute the consensus order of transactions committed by creators (even if there is a certain number, less than 1/3, of failed or malicious nodes). Thus, a correctly ordered history of the experimental data generation and processing is stored in the PMD registry. Due to the presence of a set of PMD request nodes, synchronization of hashgraphs between them (by the gossip algorithm), and the hashgraph consensus algorithm, the PMD registry operates as a replicated state machine [12], which ensures its high fault tolerance.

Of course, the real PMD MS includes a number of additional components necessary for the operation of the system, including modules for controlling user access to the components of the system on the basis of Public Key Infrastructure (PKI), monitoring the presence of PMD records for each file in the distributed storages, etc. Due to lack of space, we do not describe these components here; details of their implementation will be presented elsewhere. Also they are not shown in Figure 1 to emphasize the main idea of constructing the PMD MS, namely, the use of the distributed consensus algorithm based on hashgraphs.

4 Conclusion

In this paper, we propose a new approach to the development of a distributed metadata management system for provenance metadata, which is fault-tolerant, safe, reliable from the point of view of the preservation and security of records from accidental or intentional distortions. This, in turn, allows to significantly improve the quality and reliability of scientific results obtained on the basis of processing and analysis of large scientific data in a distributed computer environment. This improvement is due to the fact that correct metadata containing, in particular, information about conditions of data acquisition are very important for high-quality data processing and correct interpretation of the results obtained. The main idea is to use modern distributed consensus algorithms, and their analysis led to the selection of the newly developed algorithm based on the use of hashgraphs as the basis for the construction of the provenance metadata management system. The proposed system is simpler and better suited for working with distributed storage than the previously developed centralized systems and systems for cloud data based on blockchain technology. As indicated in the Introduction, any central service is a bottleneck and a point of failure of the system as well as problems arise in organizing the management of the central service and ensuring trust in it. The previously proposed solutions based on the blockchain are very heavy and resource intensive due to the use of the proof-of-work consensus mechanism. On the contrary, the proposed in this work system is lightweight and provides fault tolerance due to the replication of PMD request nodes and the fact that the hashgraph algorithm provides a consensus even if there is a certain number, less than 1/3, of failed or malicious nodes of PMD creators. The storage reliability of the data itself is provided by the storage server (RAID, replicas). The results of this work become of particular importance due to the active development of such a modern trend as the open science model, which becomes especially actual in the big data era, when no one team can fulfill a full analysis of the results of large experiments, so that many independent teams take part in their analysis.

The suggested approach is currently under implementation in SINP MSU in the framework of the project supported by the Russian Science Foundation.

References

1. Hills, D., Downs, R.R., Duerr, R., Goldstein, J.C., Parsons, M.A. and Ramapriyan, H.K.: The importance of data set provenance for science. Eos, **96**, 10.1029 (2015).
2. David, P.A.: Understanding the emergence of open science institutions: functionalist economics in historical context, Industrial and Corporate Change, **13**(4), 571–589 (2004)
3. Zafar, F., Khan, A., Suhail, S., Ahmed, I., Hameed, K., Khan, H.M., Jabeen, F. and Anjum, A.: Trustworthy data: A survey, taxonomy and future trends of secure provenance schemes. Journal of Network and Computer Applications, **94**, 50–68 (2017)
4. da Cruz, S.M.S., Campos, M.L.M. and Mattoso, M.: Towards a taxonomy of provenance in scientific workflow management systems. In: Proceedings of the World Conference on Services-I, pp. 259–266. IEEE, N.Y. (2009)
5. Baliga, A.: Understanding blockchain consensus models. Tech. rep., Persistent Systems Ltd, https://pdfs.semanticscholar.org/da8a/37b10bc1521a4d3de925d7ebc44bb606d740.pdf (2017) Last accessed 12 Apr 2018
6. Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K. and Njilla, L.: Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 468–477. IEEE Press, N.Y. (2017)
7. Ramachandran, A. and Kantarcioglu, M.: SmartProvenance: A Distributed, Blockchain Based DataProvenance System. In: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, pp. 35-42. ACM, N.Y. (2018)
8. Baird, L.: The Swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance. Swirlds Tech Report SWIRLDS-TR-2016-01, http://www.swirlds.com/developer-resources/whitepapers (2016). Last accessed 12 Apr 2018
9. Lamport, L.: The Part-Time Parliament. ACM Transactions on Computer Systems **16**(2) 133–169 (1998)
10. Ongaro, D. and Ousterhout J.K.: In search of an understandable consensus algorithm. In: USENIX Annual Technical Conference, pp. 305–319. USENIX Association (2014)
11. Correia, M., Veronese, G.S., Neves, N.F. and Verissimo, P.: Byzantine consensus in asynchronous message-passing systems: a survey. International Journal of Critical Computer-Based Systems, **2**(2), 141–161 (2011)
12. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D. and Terry, D.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing, pp. 1–12. ACM, N.Y. (1987)
13. Lamport, L., Shostak, R. and Pease, M.: The Byzantine generals problem. ACM Transactions on Programming Languages and Systems (TOPLAS) **4**(3), 382–401 (1982)