Consistency Models for Global Scalable Data Access Services

Michał Wrzeszcz[1,2(✉)], Darin Nikolow[2], Tomasz Lichoń[1,2], Rafał Słota[2], Łukasz Dutka[1], Renata G. Słota[2], and Jacek Kitowski[1,2]

[1] Academic Computer Centre Cyfronet AGH, AGH University of Science and Technology, Krakow, Poland
dutka@cyfronet.pl

[2] Department of Computer Science, Faculty of Computer Science, Electronics and Telecommunications, AGH University of Science and Technology, Krakow, Poland
{wrzeszcz,darin,tlichon,slota,rena,kito}@agh.edu.pl

Abstract. Developing and deploying a global and scalable data access service is a challenging task. We assume that the globalization is achieved by creating and maintaining appropriate metadata while the scalability is achieved by limiting the number of entities taking part in keeping the metadata consistency. In this paper, we present different consistency and synchronization models for various metadata types chosen for implementation of global and scalable data access service.

Keywords: Data consistency model · Global data access
Distributed data storage

1 Introduction

At some time in the past due to the rapid developments in computer science and technology, the idea of distributed systems has come to life bringing new research challenges. One of those challenges is keeping distributed data consistent when the data is accessed by more than one independent client/process running asynchronously or when the data is replicated [8].

Cloud storage [2] is a popular data storage service for data accessing from anywhere – but it is bound to a single data storage provider, although there are needs from various communities to have global unified access to data distributed among independent providers. Data-driven scientific research using data intensive distributed applications is an example where efficient and scalable global data access services are needed [1].

The reported study is a continuation of our previous work [3,15] in which we defined globally distributed data access model for provision of transparent data access with quality. The model, based on our previous experience [7,9], covers problems with different data formats, storage systems and data locations, using

R. Wyrzykowski et al. (Eds.): PPAM 2017, LNCS 10777, pp. 471–480, 2018.
https://doi.org/10.1007/978-3-319-78024-5_41

context information represented by metadata. In this article we focus on different kinds of metadata consistency with working hypothesis that different consistency and synchronization models for various metadata types are required for provisioning global and scalable data access. We assume that the globalization is achieved by creating and maintaining appropriate metadata (with different consistency models) while the scalability is achieved by limiting the number of entities taking part in keeping the metadata consistency (different synchronization models).

The rest of the paper is organized as follows. The state of the art is presented in Sect. 2. Section 3 outlines description and classification of metadata for global data access with useful consistency models according to our vision. Section 4 presents details of metadata management for a global scalable data access service which realizes our vision. Test results of consistency and performance benchmarking on the implemented service are given in Sect. 5 and the last section concludes the paper.

2 State of the Art

The consistency of shared data has popped up as a research topic with the advent of parallel and concurrent processing. The problem concerns the assertion of the correctness of subsequent updates or writes and reads of shared or distributed data. Consistency model describes the rules for ordering and visibility of distributed memory updates. Various consistency models allow achieving various goals so we have analyzed which models are used by systems with different properties.

Distributed storage systems are ubiquitous in modern computing environments allowing for better performance and data availability. Viotti and Vukolić in [11] presented a survey of consistency models for distributed storage systems. Strong consistency is mostly wanted but it comes with the price of sacrificing the performance. Hence, many data access services implement some weaker sort of consistency like the eventual consistency. Some popular solution implementing distributed storage are briefly overviewed below in the context of data consistency. We have chosen to present only selected examples of related work in the field of distributed storage systems. Each of presented solutions uses certain consistency to achieve desired features. Our goal is to depict those features and open a discussion whether it is possible to propose a model of metadata management for global and scalable data access service, that combines them altogether.

Ceph [13] is a distributed object-based storage solution implementing the CRUSH [14] ring hashing algorithm for distributing data among storage nodes. Ceph by default implements strong consistency by updating all the replicas of an object and the object itself before allowing further data access operation on that object. A weak consistency implementations for Ceph, name PROAR, has been proposed in [16].

HDFS [6] is a filesystem with centralized metadata management. The consistency of data is realized with the WORM model of data management which means that once created (open, write, close) a file cannot be modified.

Lustre [12] is a cluster filesystem which does not provide replication capabilities but rather relies on the underlying storage layers for data availability.

GlobalFS [5] is a multi-site filesystem implementing strong data consistency. The sites, however, are single owned implementing the same policy for data management.

iRODS [4] is a storage solution which can integrate distributed storage resources and manage them based on user-defined rules implemented as micro-services. The WORM data management is used to keep the data consistent. User-defined data replication is possible realizing a user-designed consistency model.

Parrot [10] creates a kind of virtual filesystem which integrates remote I/O resources. While this approach maintains application mobility it does not provide itself any data consistency mechanisms other than the ones provided by the remote I/O resources.

As we can see there are different approaches to data consistency used by the presented solutions. Metadata consistency is essential for providing correct data access. Depending on the scale and purpose of the system different consistency models are used. By taking into account the requirements for consistent anytime/anywhere view of data we believe that our vision of metadata consistency management allows for high system scalability.

3 Metadata for Global Data Access Services

Our vision of globally distributed data access service assumes that in the general case the storage resources for such service are owned by a group of providers (for more details see [15]). The providers apply independent access policies to own resources according to their business or scientific objectives. The users are granted access by one or more providers; they can access their data by connecting to any of the providers via various data access interfaces and have always the same uniform view of their data without regard to the providers. Such assumptions impose relevant metadata organization and management which is described below.

3.1 Metadata Description

The metadata provides the necessary information for the proper working of storage service concerning various aspects. The following types of metadata have been identified:

- *user metadata* - it provides information about the users on the highest level. That information is either provided during registration or obtained from third party authentication services.
- *storage provider metadata* - it contains information about the given storage provider like its name, location, supported authorization methods and storage that they possess.

- *storage metadata* - it is composed of information about its provider, storage type, interfaces suitable for accessing it, its restrictions and capabilities such as its availability, latency, and throughput.
- *dataset metadata* - it provides information about the stored data. The data is stored within some dataset, which usually has a name, access control list, and is linked with storage provider in which data is located.
- *namespace metadata* - it holds identifiers for stored entities and their relations between each other. Users store their data within a dataset and create entities under some identifiers. Entities may be different depending on the type of dataset, e.g. for file system we have files, for noSQL databases - documents, for other structures - it may be a row in a table or a data stream. The common thing is that there is a namespace defined, allowing to find those entities within a dataset. It may be flat, like in a key-value store, or hierarchical, like in a file system.
- *administrative metadata* - the metadata about the entities stored within a dataset. There are a few predefined administrative pieces of information about a creator, owner, access times, permissions or size, which are typical metadata seen in many traditional filesystems.
- *custom metadata* - it is defined by the users and may have an arbitrary structure.
- *location metadata* - it provides a mapping between logical entities created by users in a dataset, into the location of the actual data. So we may see it as a link between user created entity and storage, together with information about identifiers of data within that storage. As the entity may be stored on multiple storage systems, this type of metadata must hold also information on how to compose data from multiple storage systems into one coherent entity that user has stored.
- *runtime metadata* - it is used to track current activity. This type of metadata is connected with a specific node in the system and includes information about user sessions, handles for open resources, usage statistics, or measures from monitoring of hardware resources.

Due to the fact that there are so many various types of metadata, we state that appropriate metadata structure and metadata management model is essential to achieve certain required characteristics of a data access service. In order to simplify the management of those metadata, we classify them further.

3.2 Metadata Classification

We have identified three main classes of metadata which are distinguished by their locality, availability requirements, updating policy:

1. cooperation metadata,
2. stored data metadata,
3. environment state metadata.

The cooperation metadata contains information about users and providers like credentials and users relations (e.g., description of users groups). It is allowed to have a delayed propagation of changes to those metadata assuming that all changes will be applied. This type of metadata is changed rarely but is frequently read.

The metadata about stored data contains the necessary information for managing data distributed among providers like the locations of file fragments, attributes of files and user-defined metadata. This type of data is updated frequently which may provide write conflicts, which should be resolved automatically.

The environment state metadata describes the state of resources and data usage which is used for load balancing. We assume local management of resources done by their provider. Those metadata are not visible for the other providers.

By taking into account the big scale of environment it should be noted that there are relatively little cooperation metadata but all providers are interested in those metadata (global data access). There is a huge amount of metadata of the other two classes, but the number of providers which are interested in accessing relevant metadata about the stored data is relatively low.

4 Metadata Management for Scalable Data Access Service

The mentioned metadata classes impose different data consistency. Caching of metadata is used to increase performance which introduces additional metadata copies kept more or less consistent according to the design requirements.

In order to create a scalable and functional solution for global data access, it is important to choose a relevant data consistency model for the presented metadata classes. The metadata classes and their consistency models according to our vision are presented in Table 1. Each metadata class has a different scope of metadata. The cooperation metadata are needed by every provider, so they have a global scope. The metadata about stored data are needed only by the providers which support the given dataset. Environment state metadata are only needed locally.

Table 1. Metadata classes and consistency models

No	Metadata class	Metadata	Consistency model	Scope
1	Cooperation	User, provider, dataset, namespace	Sequential	Global
2	Stored data	Administrative, custom, location	Eventual	Relevant providers for a dataset
3	Environment state	Runtime	Eventual	One provider

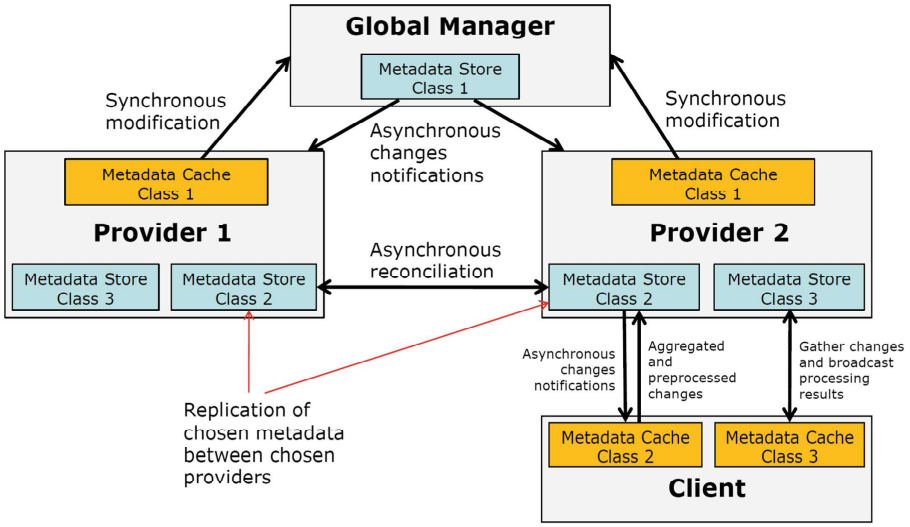

Fig. 1. Global data access entities

Our vision of metadata management for scalable data access service is depicted in Fig. 1.

In our approach datasets are called spaces. A space may be supported by one storage provider, which means that the space data is stored on its storage, or by many providers. Each space has the same access methods defined, whereas its data may be possibly stored on different storage systems.

The sequential consistency is needed for the cooperation metadata. Such metadata is stored by Global Manager. The providers cache the necessary metadata to cope with the performance issues. Thus, it is acceptable to read outdated data from cache but all updates have to be executed at the latest version of metadata to maintain the sequential consistency of metadata store. The modifications of cooperation metadata (Class 1) are done as follows:

1. the provider synchronously modifies metadata (using API of Global Manager) to avoid conflicts,
2. if modified metadata is cached by any provider, the relevant providers are notified asynchronously, which results in lazy replication.

The eventual consistency is used when synchronizing metadata about stored data (Class 2) between providers. This is done in this way because the metadata traffic between providers can be high and keeping strong consistency would be too costly. The users use the client entity to access data stored in the resources of supporting providers. As the user can be supported by more than one provider it is possible to have the same data replicated ('closer' to the user) to achieve higher transfers to the client. The providers need to have consistent metadata of Class 2 to allow a unified view of the users' data regardless of the provider

to which the client is connected. The client caches relevant metadata of Class 2 and the update runs as follows:

1. The client sends aggregated metadata changes to the provider,
2. The provider sends the changes to all connected clients. The newer change always wins in case of conflicts between clients changes. The client applies the change suggested by the provider to its cache,
3. The provider sends the changes to the providers that are interested in particular metadata (concerning a supported space),
4. The providers that receive the changes apply them using a conflict resolving algorithm that is based on revisions of metadata. The algorithm guarantees that all providers resolve a conflict in the same way,
5. The providers send the changes to all clients connected to them. These changes already include the result of the eventual conflict resolution.

The metadata of Class 3 is stored and managed by a single provider. Although the clients can propagate changes of such metadata with a delay, the version of particular metadata in metadata store of the provider is always considered as current and the client cache is considered as incoherent until the provider approves the change. Thus, the consistency of metadata of Class 3 can be considered as eventual because of incoherent caches. The update of metadata of Class 3 is as follows:

1. The client sends aggregated metadata changes to the provider,
2. The provider processes changes and sends the result of the processing to chosen connected clients if needed, e.g., changes in monitoring data can result in a recommendation of reconfiguration for the clients that work with particular storage system.

The proposed solution allows efficient metadata processing due to its use of eventual consistency and lazy replication. Although it can result in loss of some metadata changes in the extreme case, the dedicated update mechanism allows conflict avoidance and changes loss applying sequential consistency for the most important metadata. As a result, the proposed solution simultaneously provides scalability and reliability needed to coordinate the global cooperation of users and providers.

5 Tests

To verify our working hypothesis the proposed metadata management has been implemented in the project called Onedata, targeted at inventing a global data access solution for scientific and business communities (roughly outlined in [15]).

Performance tests of the proposed metadata management have been conducted and presented in Fig. 2. We have measured the transfer rates of accessing data stored in: (1) a space supported by only one provider (denoted local space), and (2) a space supported by more than one providers (denoted shared space) for which metadata synchronization between the providers occurs. Presented

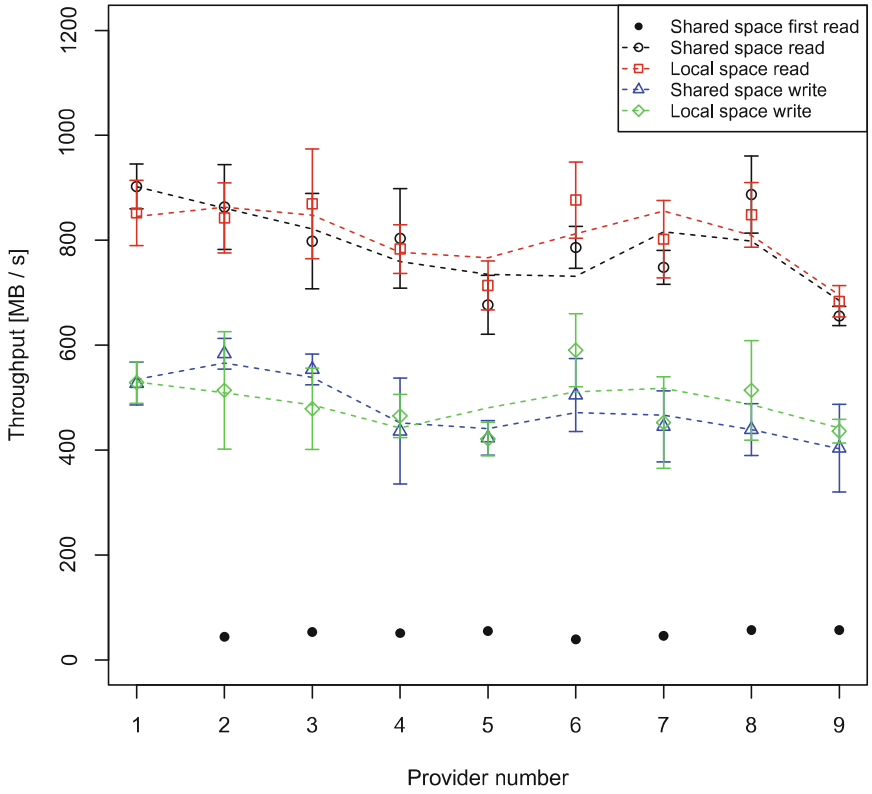

Fig. 2. Data access throughput for local and shared Onedata spaces

load corresponds to typical user behavior of creating a space and extending its support to multiple providers.

The test environment has been set up using 10 virtual machines with public IP addresses. Each of them is a part of the same local network. The virtual machines have 4 CPUs of 2.6 GHz and 8 GB of RAM memory each. The first machine is running the module that implements the Global Manager functionality inside a docker container, and the other nine are running modules implementing storage provider functionality and client functionality, both in separate docker containers.

The test procedure is as follows. A file is written to the shared space via the first provider and then the file is read and written via the rest of providers. The first read for providers 2–9 is presented separately since it requires inter-provider data transfers which are limited by the network link. Additionally, each provider supports its own local space for which similar writes and reads are executed. The file size is 1 GB.

The transfer rates for local and shared spaces are similar which means that the metadata synchronization does not essentially influence the performance.

The average transfer rate for the first reads is 50 MB/s while the average transfer rate between the providers measured by using scp is 75 MB/s. It results from the behavior of FUSE which reads data blocks sequentially waiting for data arrival before sending the next request. The client partially compensates this by prefetching which by default is tuned for accessing fragments of big files. If we use the option of using more aggressive prefetching which requests all file blocks at once than the transfer rate for the first read is 81 MB/s.

6 Conclusion

In this paper, we have presented our vision of metadata management for a global and scalable storage service in which special attention has been paid to efficiency aspects regarding metadata consistency requirements. We have presented an implementation of our vision as a global data access service in which role-dependent consistency models allowing for high scalability and efficiency are applied. The globalization of the service is achieved by introducing relevant metadata. Moreover, the introduction of different metadata classes with different consistency models and scopes allows for low overhead of metadata synchronization and for high scalability. The conducted tests have shown that our approach is efficient and indeed introduces very little overhead.

Acknowledgements. This research is supported partly by the European Regional Development Fund program no. POIG.02.03.00-12-137/13 as part of the PLGrid Core. R. G. Słota, D. Nikolow and J. Kitowski acknowledge AGH-UST statutory Grant no. 11.11.230.337. Support by IndigoDC project no. RIA 653549 is also acknowledged by Ł. Dutka, M. Wrzeszcz. T. Lichoń acknowledges Polish Ministry of Science and Higher Education under AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications statutory project no. 15.11.230.317, and owes special thanks to ACK Cyfronet AGH computing center for providing computing and storage infrastructure for this research.

References

1. Baud, J.P.B., Casey, J., Lemaitre, S., Nicholson, C., Smith, D., Stewart, G.: LCG data management: from EDG to EGEE. In: UK e-Science All Hands Meeting, Nottingham, UK (2005). http://www.allhands.org.uk/2005/proceedings/papers/475.pdf
2. Drago, I., Mellia, M., Munafo, M., Sperotto, A., Sadre, R., Pras, A.: Inside dropbox: understanding personal cloud storage services. In: Proceedings of the 2012 ACM Conference on Internet Measurement, IMC 2012, pp. 481–494. ACM, New York (2012)
3. Dutka, Ł., Wrzeszcz, M., Lichoń, T., Słota, R., Zemek, K., Trzepla, K., Opiola, Ł., Słota, R., Kitowski, J.: Onedata - a step forward towards globalization of data access for computing infrastructures. In: Proceedings of the International Conference on Computational Science, ICCS 2015, Computational Science at the Gates of Nature, Reykjavík, Iceland, pp. 2843–2847, 1–3 June 2015

4. Hünich, D., Müller-Pfefferkorn, R.: Managing large datasets with iRODS - a performance analysis. In: Proceedings of the 2010 International Multiconference on Computer Science and Information Technology (IMCSIT), pp. 647–654. IEEE (2010)
5. Pacheco, L., Halalai, R., Schiavoni, V., Pedone, F., Riviere, E., Felber, P.: GlobalFS: a strongly consistent multi-site file system. In: Proceedings of the 35th IEEE Symposium on Reliable Distributed Systems, SRDS 2016, Budapest, Hungary, 26–29 September 2016, pp. 147–156. IEEE Computer Society (2016). http://dblp.uni-trier.de/db/conf/srds/srds2016.html#PachecoHSPRF16
6. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), MSST 2010, pp. 1–10. IEEE Computer Society, Washington (2010). https://doi.org/10.1109/MSST.2010.5496972
7. Slota, R., Nikolow, D., Skalkowski, K., Kitowski, J.: Management of data access with quality of service in PL-Grid environment. Comput. Inf. **31**(2), 463–479 (2012). http://www.cai.sk/ojs/index.php/cai/article/view/950
8. Słota, R., Nikolow, D., Skitał, Ł., Kitowski, J.: Implementation of replication methods in the grid environment. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 474–484. Springer, Heidelberg (2005). https://doi.org/10.1007/11508380_49
9. Słota, R., Skitał, Ł., Nikolow, D., Kitowski, J.: Algorithms for automatic data replication in grid environment. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 707–714. Springer, Heidelberg (2006). https://doi.org/10.1007/11752578_85
10. Thain, D., Livny, M.: Parrot: an application environment for data-intensive computing. Scalable Comput. Pract. Exp. **6**(3), 9–18 (2005)
11. Viotti, P., Vukolić, M.: Consistency in non-transactional distributed storage systems. ACM Comput. Surv. **49**(1), 19:1–19:34 (2016). https://doi.org/10.1145/2926965
12. Wang, F., Oral, S., Shipman, G., Drokin, O., Wang, T., Huang, I.: Understanding Lustre Filesystem Internals. Technical report ORNL/TM-2009/117, Oak Ridge National Lab., National Center for Computational Sciences (2009)
13. Weil, S.A., Brandt, S.A., Miller, E.L., Long, D.D.E., Maltzahn, C.: Ceph: a scalable, high-performance distributed file system. In: Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI), pp. 307–320 (2006)
14. Weil, S.A., Brandt, S.A., Miller, E.L., Maltzahn, C.: CRUSH: controlled, scalable, decentralized placement of replicated data. In: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, SC 2006, Tampa. ACM, New York (2006). https://doi.org/10.1145/1188455.1188582
15. Wrzeszcz, M., Trzepla, K., Słota, R., Zemek, K., Lichoń, T., Opioła, Ł., Nikolow, D., Dutka, Ł., Słota, R., Kitowski, J.: Metadata organization and management for globalization of data access with Onedata. In: Wyrzykowski, R., Deelman, E., Dongarra, J., Karczewski, K., Kitowski, J., Wiatr, K. (eds.) PPAM 2015. LNCS, vol. 9573, pp. 312–321. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32149-3_30
16. Zhang, J., Wu, Y., Chung, Y.C.: PROAR: a weak consistency model for Ceph. In: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), pp. 347–353 (2016)