Metadata search on large-scale distributed storage systems

Dimitrios Vasilas[1, 2] and Marc Shapiro[2]

[1]Scality
[2]Sorbonne Universités-UPMC-LIP6 & Inria
dimitrios.vasilas@lip6.fr, marc.shapiro@acm.org

1 Problem Statement

The research problem addressed in this PhD is the locating and management of data stored in object storage systems. Our goal is to design and implement a scalable, geo-distributed search system, focused on queries on metadata.

Today's object storage systems store billions of objects and petabytes of content and are highly updated. This poses significant challenges to a search subsystem:

- Enabling fast queries on a very large, evolving collection of data.
- Support queries on metadata attributes containing a mix of data types, including text, integers and complex structures such as access control lists.
- Tracking content updates incrementally as they occur, without incuring overhead for the data store.
- Geo-distributing the index. The search subsystem should support concurrent updates and queries originating from clients located in different geographic locations, and should remain available in the presence of network partition.

2 Background

Object stores identify and locate objects based on globally unique identifiers (keys). Although key-based access is scalable, it is only useful when the keys of objects that need to be located are known. Various use cases, however, require the ability to locate an object using metadata attributes, without knowing the exact object key. One example is data lifecycle management which involves applying policies to the backup, archival and migration of data, based on their size and access patterns.

Metadata consist of attributes generated by the storage system (content size, timestamp of last modification, author, access control lists), as well as custom, user-defined attributes, represented as arbitrary key-value pairs.

The use case of data lifecycle management includes queries with some common characteristics; Searches often contain more than one metadata attributes, and use both exact match and range predicates as well as logical operators.

We model a geo-distributed data store as a set of storage processes running on servers that are placed at different geographic sites (data centers). Storage processes act as ingest points for content updates from clients, and replicate data among them. The search system should receive client updates, and enable clients located at any site to search the entire data store.

3 Literature Review

Some indexing algorithms support keyword search, thanks to a distributed inverted index implemented atop a Distributed Hash Tables (DHT) [1]. Although a DHT provides efficient key lookup, this approach is not adapted to searching for ordered data, such as strings and dates, which involves implementing range queries.

Other approaches use skip-lists, a multi-level indexing mechanism based on hierarchies of lists [2]. This is more adapted to range queries, since skip lists are ordered.

Another class of indexing systems implement metadata search for large-scale file systems [3, 4]. These approaches are not suitable for the flat namespace of object storage systems, as they assume the hierarchical structure of traditional file systems in order to leverage namespace locality.

4 Approach

We have implemented a prototype metadata search system, which addresses some of the above challenges.

Our system maintains a distributed inverted index. It consists of a set of indexing and search proceses. An indexing process receives client updates, updates its local index, and propagates updates to other indexing processes. A search process receives query op-

erations, contacts the appropriate indexing processes, and merges the retrieved results. Indexing processes are organised in groups, and each group is assigned a set of metadata attributes to index [5]. Each process of a group is located in a different geographic site. It receives local updates for the assigned attributes, and propagates results to the other sites.

The system's inverted index maps metadata attribute values to sets of object keys, indicating which objects cocntain the particular attribute values. Metadata attributes of different data types are encoded as text using the same representation, and are stored in lexicographic order. The system can thus use a uniform search interface, and gsupport range queries on ordered data types.

Our inverted index implementation relies on Conflict-Free Replicated Data Types (CRDTs), replicated data types that guarantee convergence of conflicting operations without the need for application conflict handling [6]. The use of CRDTs enables indexing processes of the same group to propagate and merge their local index, without the need for central synchronization, despite messages being duplicated and reordred.

Indexing processes store the inverted index persistently in AntidoteDB [7], a highly-available, geo-distributed key-value store. We use Antidote's geo-replication mechanism to propagate local index updates between indexing processes of the same group.

5 Discussion

Indexing processes update the inverted index and propagate index updates between site asynchronously, in order to avoid degrading the performance of the storage system. Inherently, this approach causes the index to be stale relative to the data store. We plan to introduce probes that estimate this statelness [8]. If staleness increases beyond a threshold specified by the application, a feedback loop will attempt to reduce it, by using techniques such as addapting the update propagation algorithm, or throttling the rate of data updates. [9].

We plan to evaluate our systems efficiency in addressing the challenges posed by the target use case. Our metrics can be the query answering latency and throughput. We will evaluate our system's scalability in indexing large datasets, and experiment with queries containing a mix of data type, various search types and logical operators. We also plan to evaluate the impact of our design decision, such as the use of CRDT's and Antidote, on the system's performance. Finally, as this PhD is prepared in collaboration with Scality, a data storage company. we plan to integrate our prototype to Scalitys storage system. This will allow us to perform further testing in a real-world environment, in order to validate and refine approach.

References

[1] Lintao Liu, Kvung Dong Ryu, and Kang-Won Lee. Supporting efficient keyword-based file search in peer-to-peer file sharing systems. In *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, volume 2, pages 1259–1265 Vol.2, Nov 2004.

[2] Shuming Shi, Guangwen Yang, Dingxing Wang, Jin Yu, Shaogang Qu, and Ming Chen. *Making Peer-to-Peer Keyword Searching Feasible Using Multi-level Partitioning*, pages 151–161. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[3] Andrew Leung, Minglong Shao, Timothy Bisson, Shankar Pasupathy, and Ethan L. Miller. Spyglass: Fast, scalable metadata search for large-scale storage systems. In *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST '09)*, February 2009.

[4] Y. Hua, H. Jiang, Y. Zhu, D. Feng, and L. Tian. Smartstore: a new metadata organization paradigm with semantic-awareness for next-generation file systems. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, Nov 2009.

[5] Amy Tai, Michael Wei, Michael J. Freedman, Ittai Abraham, and Dahlia Malkhi. Replex: A scalable, highly available multi-index data store. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 337–350, Denver, CO, 2016. USENIX Association.

[6] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. volume 6976, pages 386–400, Grenoble, France, October 2011.

[7] Antidote reference platform. `https://github.com/SyncFree/antidote`.

[8] Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, and Ion Stoica. Probabilistically bounded staleness for practical partial quorums. *Proc. VLDB Endow.*, 5(8):776–787, April 2012.

[9] Haifeng Yu and Amin Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Trans. Comput. Syst.*, 20(3):239–282, August 2002.