Poster Abstract: EdgeStore: Integrating Edge Computing into Cloud-Based Storage Systems

Zijiang Hao, Qun Li
College of William and Mary, Williamsburg, VA, USA
{hebo, liqun}@cs.wm.edu

***Abstract*—We present EdgeStore, a cloud-based storage system that integrates edge computing for better performance. Preliminary results on our prototype system demonstrate the efficiency of EdgeStore when working in an edge computing environment. We believe that EdgeStore provides a new perspective on how to exploit the potential of edge computing in cloud-based systems.**

I. Introduction

Edge computing has emerged over the last several years as a new computing paradigm that extends cloud computing [1], [2], [3], [4]. By executing computational tasks on the edge of the network, edge computing establishes an environment that enjoys better network conditions, including shorter network latency, higher network bandwidth, and more stable network connections. All of these are critical for QoS-sensitive applications, such as mobile-cloud applications, IoT applications, big data applications with real-time constraints, and so on.

As edge computing is an extension of cloud computing, it is a natural idea to integrate edge computing into cloud-based storage systems for better performance. In light of this, we propose EdgeStore, a cloud-based storage system enhanced by edge computing. The design of EdgeStore can be summarized into the following aspects.

(1) In EdgeStore, every data object consists of one or more *data field*s, and every data field has a *synchronization policy* attached. EdgeStore provides a *default* synchronization policy for all data fields, but a developer could *override* this default synchronization policy by attaching customized synchronization policies to the data fields she has defined. The goal of the default synchronization policy is to help the system achieve reasonable performance in an edge computing environment in most cases. When a data field needs to be specially handled to fit the edge computing environment better or to achieve some particular goals, we believe it is the developer who has defined the data field has the knowledge of how to properly handle it. In light of this, EdgeStore provides a programming interface, through which developers could customize the synchronization policies of data fields, helping the system make wiser decisions when synchronizing the data fields for write operations.

(2) EdgeStore provides *location-based* data accessing services. More specifically, it tracks the movement of any client device that is reading (a large amount of) data from the system, and predicts the next edge node that the client device will switch to. By doing this, EdgeStore can prefetch the data to the next edge node, such that the client device could enjoy seamless edge node switches during the read operation, improving both read performance and user experience.

II. Data Object Examples

As mentioned, an EdgeStore data object consists of one or more data fields. Each data field has a synchronization policy attached. The default synchronization policy is to eagerly synchronize to the cloud as much data as possible from the beginning of the data field, based on the currently available network resources. The remaining part of the data field could be lazily synchronized to the cloud when the system has extra network resources.

videoData
textData
videoData

(a) The Video object. (b) The SNSPost object.

Fig. 1: Data object examples.

Fig. 1(a) illustrates a simple EdgeStore data object, i.e., the Video object, which contains only one data field, i.e., videoData. The developer may override the synchronization policy of videoData, specifying that at least the first 20 MB data of the data field be eagerly synchronized to the cloud. By doing this, the data field can be immediately read after it has been written, and the system has a good change to provide seamless data accessing services by eagerly synchronizing the remaining part of the data field when it is being read.

Fig. 1(b) illustrates another EdgeStore data object, i.e., the SNSPost object, which contains two data fields, i.e., textData and videoData. The developer may override the synchronization policies of the two data fields, specifying that all the data of textData and at least the first 20 MB data of videoData be eagerly synchronized to the cloud.

III. System Design

Fig. 2 illustrates the software stack of EdgeStore. As shown in the figure, each data object instance exposes a *Data Access Interface* to user applications, through which user applications can read/write its data fields. Each data object instance also contains a *Synchronization Proxy*, through which it can communicate with the *Synchronization Manager*. Developers could

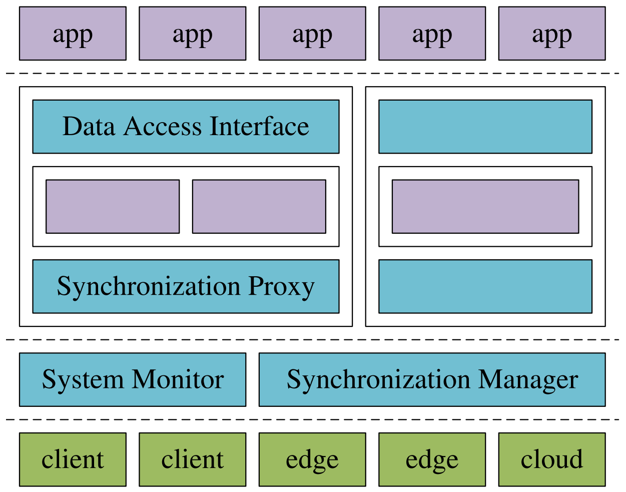

Fig. 2: EdgeStore software stack.

implement customized synchronization policies for data fields, by programming their *callback function*s. More specifically, they could give suggestions to the Synchronization Manager through the Synchronization Proxy in the callback functions, helping the Synchronization Manager make wiser decisions on how to synchronize the data fields.

The *System Monitor* keeps monitoring the states of system entities, including client devices, edge nodes, and the cloud. It provides necessary information to the Synchronization Manager, as well as to the data object instances to enable location-based data accessing services.

IV. Evaluation

We first build a testbed and deploy our preliminary implementation of EdgeStore on it. The testbed consists of four edge nodes and one cloud server. Each edge node is connected to the cloud server through a (10 ms, 40 Mbps) network link. Then we conduct experiments to evaluate the performance of using synchronization policies. Evaluation on location-based data accessing services is left for future work.

Table I: Benefits of involving edge computing.

	Throughput (tasks/min)	Network Usage (Mbps)
w/ edge sync	24.56	16.37
wo/ edge sync	5.99	39.94

The first group of experiments are designed to evaluate the benefits of involving edge computing in a cloud-based storage system. More specifically, we evaluate to what extent edge computing can help by applying synchronization policies to data fields. We simulate the scenario in which instances of the Video object shown in Fig. 1(a) are created and 200 MB data is written to each of them by client devices, with a data transmission rate of 8 Mbps. On each edge node, the arrival intervals of the Video instances follow an $N(10 \text{ sec}, 4 \text{ sec}^2)$ distribution. The synchronization policy that at least the first 20 MB data of the videoData field be eagerly synchronized to the cloud is attached to each Video instance. Table I shows the results. Clearly, when applying the synchronization policy via edge computing, the system could achieve higher throughput with lower network usage.

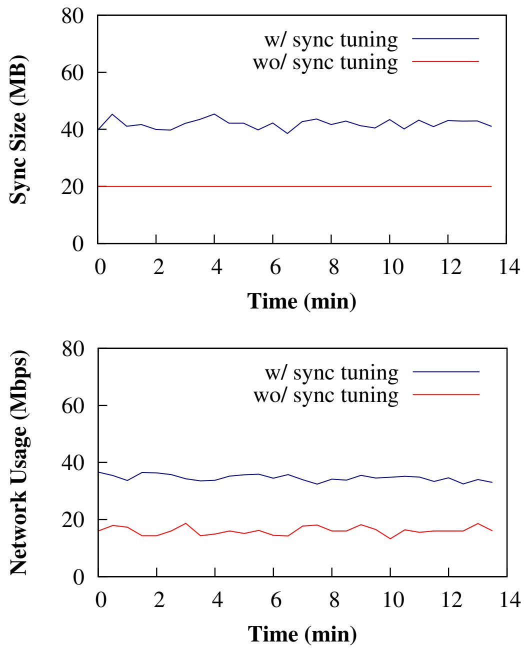

Fig. 3: Performance of EdgeStore.

The second group of experiments are designed to show how well EdgeStore can work in an edge computing environment. We only consider one edge node and the cloud server in these experiments. We still simulate the scenario in which Video instances are created and 200 MB data is written to each of them, with a data transmission rate of 8 Mbps. The arrival intervals of the Video instances follow an $N(10 \text{ sec}, 4 \text{ sec}^2)$ distribution, and the same synchronization policy as the one described above is attached to each Video instance. EdgeStore provides a feature called "Synchronization Tuning", i.e., the System Monitor monitors the network usage, and informs the Synchronization Manager if it finds that the network resources are not fully used. The Synchronization Manager will then try to synchronize more data for the active data fields. Fig. 3 compares the results of enabling Synchronization Tuning and those of not enabling it. Clearly, when Synchronization Tuning is enabled, the system could make better use of the network resources, and the system performance can hence be enhanced.

References

[1] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the internet of things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, 2015.
[2] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of MoBiData*, 2015, pp. 37–42.
[3] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *Proceedings of WASA*, 2015, pp. 685–695.
[4] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proceedings of HotWeb*, 2015, pp. 73–78.