A General Architecture for Monitoring Data Storage with OpenStack Cloud Storage and RDBMS

Bin Hu
College of Engineering
Bo Hai University
Jinzhou, China
hubinustb@gmail.com

Abstract—**The process monitoring and fault diagnosis is one of the important problems in the process industry. Archived monitoring data is valuable for long-term analysis and decision making. In this paper, we propose a general architecture to manage and storage archived monitoring data. It has been implemented within an archiving terabytes of monitoring data in metallurgy industry. It is an effective way to solve the problems of the efficiency and expandability via practical applications.**

Keywords- Cloud Storage; Massive Data Storage; OpenStack; Data Monitoring

I. INTRODUCTION

With the development of industrial technology, the status monitoring system for equipment also got the very great development, and playing an important role in ensuring the safety, stable, reliable of equipment. Taking metallurgy for example, we need to monitor the temperature, voltage, amplitude and other equipment status. How to store and deal with such massive archived data has become a problem.

Large scale metallurgy enterprise systems make their administrator a very complex task. To give administrators an on-line view of the system health, many monitoring platforms have been developed. Meanwhile, archived monitoring data is valuable for long-term analysis and decision making. Consequently, we designed and developed the general platform for monitoring data in metallurgy industry.

Cloud storage system is a cooperation storage service system with multiple devices, many application domains, and many service forms [1]. The development of cloud storage system is benefit from the broadband network, Web 2.0, storage virtualization, storage network, application storage integrated with servers and storage devices, cluster technology, grid computing, distributed file system, content delivery network, peer-to-peer, data compression, data encryption, etc.

We propose a low-cost, flexible, high-performance solution for a large scale data management of archiving data in metallurgy industry. Scale is too large for most commercial databases. Even if it weren't, cost would be very high. Specifically, we present our benchmarking effort on open source object storage (Swift) and RDBMS. We compare the throughput of Swift and MySQL Cluster. Although, there would have been other candidates for the performance comparison, these systems cover a broad area of modern storage architectures.

Our contributions are threefold.

(1) We present the use case and massive data challenge of application performance management with taking into account the time characteristics of the industrial monitoring data for sharding;

(2) We present a performance comparison of two different data store architectures on a same structured compute cluster;

(3) Finally, we report on details of our experiences with these systems from an industry perspective.

The rest of this paper is arranged as follows. Section 2 introduces related work. Section 3 presents the system architecture and model. Section 4 proposes experiment analyses. Section 5 makes a conclusion.

II. RELATED WORK

In addition to traditional enterprise-class storage technology, many organizations now have a variety of storage needs with varying performance and price requirements. In recent years, many cloud storage models and platforms proposed and implemented [2-5]. Most currently available cloud data management systems such as Amazon S3, Google GFS[6], Bigtable[7], Apache Cassandra[8] and Dynamo[9] have been designed to provide good scale-out strategy for mass storage. However, they can not efficiently supporting complicated queries and their data retrieval API is narrow and restrictive. In contrast, traditional RDBMS can implement the multidimensional data model and operations, which supports SQL is an excellent tool for this type of task. However, it has relatively little or no ability to scale-out. In metallurgy industry, in view of data complex retrieval and scalability requirements, a more thorough view of the monitored system is needed.

In [10], it proposed a Relational Database and Key-Value store combined Cloud Data management ("RDB-KV CloudDB") framework. In [11], it introduced LogBase – a scalable log-structured database system that adopts log-only storage for removing the write bottleneck and supporting fast system recovery. It is designed to create a scalable archival storage system named PRESIDIO that efficiently stores diverse data in [12]. It had implemented an approach for archiving data provenance and quality information produced by the workflow populating the data archive, which was the use of domain-specific and generated metadata in [13].

However, these store systems designed for supporting data scalability and improving system performance, with little care for the characteristic of archived data.

Allow for time characteristics of monitoring data, this paper proposes a general architecture for massive monitoring data storage, which is designed to provide a solution for storage and management metallurgy data more availably.

III. Cloud Storage System Architecture Model

A. Architecture

Compared with traditional storage, cloud storage can be any place to provide different data storage and business access services [14]. The problem of architecture design should rekindle interest in the well-known problems of index selection, data partitioning and the selection of materialized views. However, while revisiting these problems, it is important to recognize the special role played by sharding. Our solution can efficiently manage archived monitoring data which is effectively stored and retrieved from Swift Objects and MySQL with data sharding.

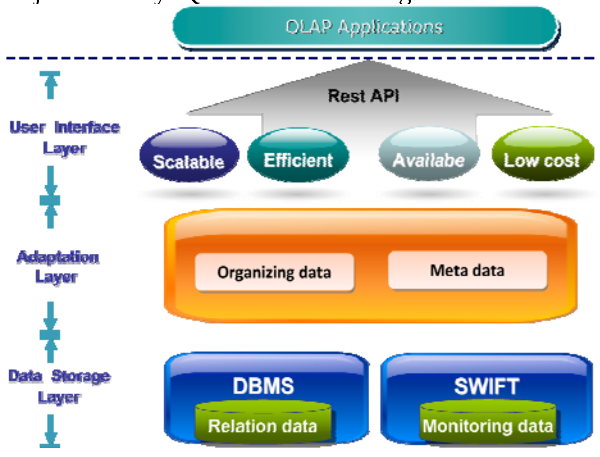

Figure 1. System architecture

Fig.1 depicts the architecture. The system architecture model consists of three layers.

1) Data Storage Layer

Storage Layer is the basic part of the system. Cloud storage consists of hundreds of thousands of servers through wide area network. The traditional single-server based data organization difficult to meet the wide area network for multi-user throughput performance and storage capacity requirements. In contrast, the data storage systems based distributed environment to better meet the needs of the application of online storage services [15]. It includes two parts. One is MySQL, which stores relation data about information of monitoring node, workshop and factory. The other is Swift, which provides redundant, scalable object storage using clusters of standardized servers capable of storing petabytes of data for monitoring. We split monitoring data file into blocks stored in Swift by timestamp of data.

2) Adaptation Layer

Adaptation Layer consists of two tools. For transforming and integrating the data, "Organizing data" component is the core of our AL, which can split and combine blocks. At the same time there is a repository for storing and managing metadata of files sharding. There are two metadata types: location information of blocks and name spaces of blocks.

3) User Interface Layer

User Interface Layer: Interface Layer provides a RESTful JSON API that can be accessed from any environment that allows HTTP requests. The HTTP methods (e.g., GET, PUT, POST, or DELETE) are typically used to implement a web service.

B. Massive Data Model

In this subsection, we describe what massive data model is. We use the standard data types, space-time data types and their related data types have already been designed and implemented as shown in Fig.2.

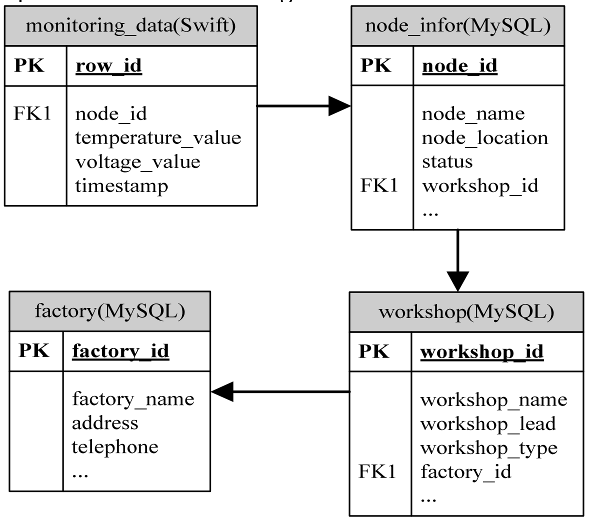

Figure 2. Massive data model

Definition 1 (*MonitoringData*): a monitoring value, denoted as *MonitoringData*, which is defined as follows.

MonitoringData=(row_id, node_id,temperature_value,voltage_value,timestamp)

In Definition 1, the *row_id* is the identifier of table row, which is offset of records in table. The *node_id* is the identifier of sensor node. The *temperature_value* is the value of temperature that sensor node monitored. And the *voltage_value* is the value of voltage that sensor node monitored. The timestamp is the time of sensor node monitoring. With the timestamp, we split the records of *monitoring_data* table into blocks every day.

Definition 2 (*Relation of data*): the relations, such as relation of node, workshop and factory denoted as *Relation of data* are shown as follows.

node_infor(node_id,node_name,node_location,status,workshop_id)

workshop(workshop_id,name,workshop_lead,workshop_type,factory_id)

factory(factory_id,factory_name,address,telephone,URL)

In definition 2, there are three entities: *node_infor*, *workshop* and *factory*. The *node_infor* table contains a foreign key to the workshop that contains a foreign key to the factory table.

C. The Key Paramter

The key parameter of the system depends on the data model and its characteristics. In a sensor-based monitoring system, there could be millions or more sensors of different kinds connected. They continuously sent monitoring data to the data center. In most cases, it also has timestamp characteristics of massive data. So we partition them by timestamp for supporting complicated query and analyzing of application. The size of sharding is a key parameter. According to actual demand, we spit the archived file into sharding by day.

IV. PERFORMANCE MEASUREMENTS AND ANALYSIS

The evaluation of the system focuses on the quantitative results from performance measurements, leaving formal monitoring data considerations and usability tests for future work.

The overall performance is influenced by many factors such as the degree of redundancy, the use of the cache for metadata, and the achievable network throughput and so on. The test setup consisted of two Ubuntu server 64 bit machines with 4GB of RAM using 4 Intel Core Xeon E5606 with 2.67GHz frequency each under the 10Mbps bandwidth. In a minimal redundancy setup of k = 3 blocks for the same data, which works faster than MySQL at the same environment.

TABLE I. EXAMPLE OF OPTIONS USED FOR PERFORMANCE COMPARISON

Symbol	DESCRIPTION	SQL Statement	Number of Records
Q1	Insertion operation	Insert into monitoring_data from test.txt	Number of records varying from one day to one year
Q2	Query operation	Select * from monitoring_data	
Q3	Ordering	Select * from monitoring_data order	

The performance of the storage system has been measured with test data for both writing and reading performance. The results are visualized in the Fig.3 and Fig.4, respectively.

Fig.3 shows performance of various insertions over two systems with varying number of records produced from one day to one year. In this case, multithreaded Swift performs better than MySQL and Single thread Swift. With Figure 3, we can conclude that the Swift is not optimal when archived data is less. For large amount of data, it is better performance relative to MySQL.

Fig.4 shows performance of various queries with varying the number of records. It also shows the multithreaded Swift query quicker than MySQL under amount of archived data. This is due to the fact that archived files are partitioned into blocks with timestamps, so the inquiries skip a lot of useless data.

Fig.5 shows performance of sorting the result of query. It also shows the multithreaded Swift sorting quicker than MySQL under amount of archived data. According to the data in block sorted by time, the sorting operation is faster than sorting the disordered data.

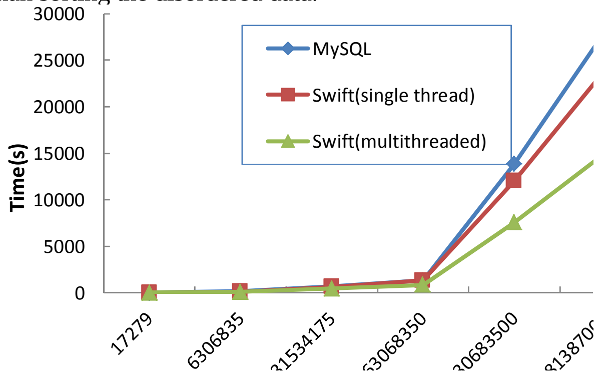

Figure 3. Performance comparison of Q1

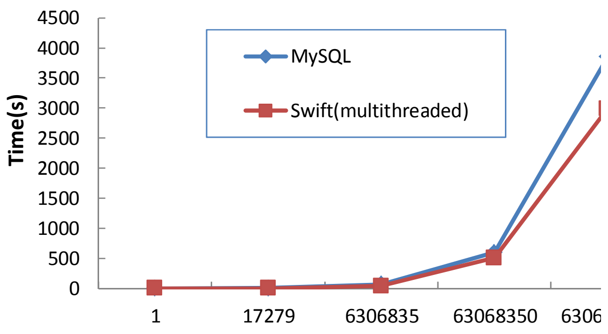

Figure 4. Performance comparison of Q2

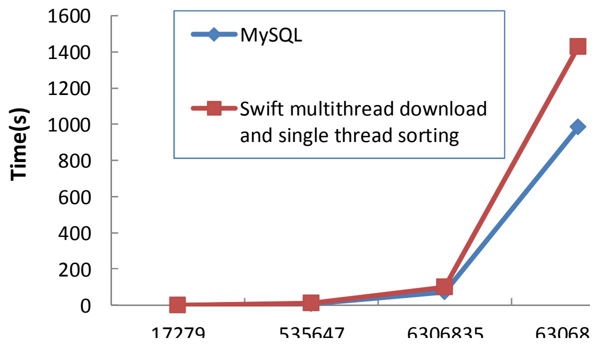

Figure 5. Performance comparison of Q3

V. CONCLUSION

The solution of massive data storage is designed for the field of industrial monitoring. This paper proposes the scalable and flexible measurement architecture. The core of the architecture consists of two novel mechanisms. One is the combination of object-oriented storage and RDBMS. The other is taking into account the time characteristics of the industrial monitoring data for sharding. In addition, it provides a generic query interface for monitoring data analysis. We believe that it is an effective massive data storage and management platform for industry monitoring data.

REFERENCES

[1] W. Zeng, Y. Zhao, K. Ou, and W. Song, "Research on cloud storage architecture and key technologies," in *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, Seoul, Korea, 2009, pp. 1044-1048.

[2] J. Zhou, N. Bruno, M. Wu, P. Larson, R. Chaiken, and D. Shakib, "SCOPE: parallel databases meet MapReduce," vol. 21, pp. 611-636, 2012.

[3] L. Kolb, A. Thor and E. Rahm, "Multi-pass sorted neighborhood blocking with MapReduce," vol. 27, pp. 45-63, 2012.

[4] S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi, "\mathcal{MD}-HBase: design and implementation of an elastic data infrastructure for cloud-scale location services," pp. 1-31, 2012.

[5] K. McKusick and S. Quinlan, "GFS: evolution on fast-forward," *Commun. ACM*, vol. 53, pp. 42-49, 2010.

[6] H. T. Vo, C. Chen and B. C. Ooi, "Towards elastic transactional cloud storage with range query support," Proc. VLDB Endow., vol.3, pp. 506-514, 2010.

[7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," *ACM Trans. Comput. Syst.*, vol. 26, pp. 1-26, 2008.

[8] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 35-40, 2010.

[9] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 205-220, 2007.

[10] D. Zhiming, G. Limin and Y. Qi, "RDB-KV: A Cloud Database Framework for Managing Massive Heterogeneous Sensor Stream Data," in *Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on*, 2012, pp. 653-656.

[11] H. T. Vo, S. Wang, D. Agrawal, G. Chen, and B. C. Ooi, "LogBase: a scalable log-structured database system in the cloud," *Proc. VLDB Endow.*, vol. 5, pp. 1004-1015, 2012.

[12] L. L. You, K. T. Pollack, D. D. E. Long, and K. Gopinath, "PRESIDIO: A Framework for Efficient Archival Data Storage," *Trans. Storage*, vol. 7, pp. 1-60, 2011.

[13] B. Cutt and R. Lawrence, "Managing data quality in a terabyte-scale sensor archive," in *Proceedings of the 2008 ACM symposium on Applied computing*, Fortaleza, Ceara, Brazil, 2008, pp. 982-986.

[14] D. Wang, "An Efficient Cloud Storage Model for Heterogeneous Cloud Infrastructures," vol. 23, pp. 510-515, 2011.

[15] T. Rabl, S. G, Mez-Villamor, M. Sadoghi, V. Munt, S-Mulero, H. Jacobsen, and S. Mankovskii, "Solving big data challenges for enterprise application performance management," *Proc. VLDB Endow.*, vol. 5, pp. 1724-1735, 2012.