

# Gateway-based Access Interface Management in Big Data Platform

Minh Chau Nguyen\*, Hee Sun Won\*

\* Big Data SW Research Department, Electronics and Telecommunication Research Institute, South Korea

chau@etri.re.kr, hswon@etri.re.kr

**Abstract**— Nowadays, there has been a massive data explosion coming from various devices sensors, social networks and IoT services. Due to big data analytics platforms, users can store, organize, and process these large sets of data to solve different issues in different domains. However, the current big data platforms still have many drawbacks. Among the limitations, managing access interfaces, an important process of analytic service development, needs to be improved significantly. The main reason is that the emergence of too many systems recently has been making the process become more and more complicated and costly. Therefore, we propose here a system related to the field of big data management, in particular to interface management to allow end-users to use easily their desired functions including metadata and data accessing. It also helps platform managers to extend and modify effortlessly the access interfaces. A case study on log analytic service is conducted to verify the validation and practice use of our system.

**Keywords**— Analytic, Big data, Gateway, Interface Management, Metadata

## I. INTRODUCTION

Nowadays, big data era has been becoming more popular and helpful due to the contributions of different sources: theorists, system builders, scientists or application designers [1]. In the meanwhile, more and more new big data systems have been invented to satisfy the new demands on storing, analyzing and processing different kinds of big data. Each of these systems has specific goals, and it is adopted on one or few specific usage patterns [2]. As a result, each of them often exposes totally different interfaces, hence causing a great concern for users who have an interest in adopting any of these systems. Additionally, the interfaces become more difficult to manage and access when the number of systems increases. In other hands, the big data platform often aims to offer many kinds of data services on big data to a large amount of users who have different demands on data processing, accessing and storing [3]. Obviously, without a flexible system for interface access and management, the applicability of the platform will be considerably reduced.

To solve the above problem, a recently promising approach is to create a middle layer for each system to provide a unified interface [4]-[6]. However, this approach has some limitations as follows in big data platforms. First, each big data system

often has totally different interfaces for its own metadata and data accessing. It means that it is almost impossible for defining a unified interface which can satisfy all existing systems. Second, users need access rights to interact directly with each system which can be a big problem for closecluster-based platforms. Finally, it is difficult to extend or add new different systems because of the limited number of supportable systems and security problems. It means that this approach is not really suitable for big data field in which more and more systems have been developed.

We propose a new architecture for interface access and management via an integrated gateway in the big data platform that aims to solve the limitations of the approach described above. By improving these disadvantages, we intend to provide a flexible system with standard mechanisms for data and metadata accessing as well as interface management.

## II. COLLABORATIVE ANALYTIC BIG DATA PLATFORM

Figure 1 shows the general concept of collaborative analytic big data platform for Big Data as a Service. Multiple users who have different demands and different specialized knowledge should be supported by various services in the big data platform. Particularly, data owners need a data-oriented environment for collecting, importing, and sharing their own data in different forms (text, video, etc.), while data scientists need an analytic environment to conduct experiments for extracting value from data. Service developers need a serviceoriented environment for developing different services based on analytic results. On other hands, business users should perform easily sandbox-based cluster deployment and platform manager should easily monitor all activities in the platform.

In order to realize the above concept, the platform should have multiple layers including many components as shown in Figure 2. There are eight big blocks in this architecture which take in charge of data storage management (File System, Data System, Data Collector), data processing management (Execution Management, Computing Framework, Processing Language, Analysis Engine) and other functionality management (System Management, Multi-tenancy Management, Interface Management, Portal Management, etc.) for multitenancy support. Moreover, one of the most important features in this design is that the architecture of each block should be flexible and based on plug-in mechanism for inserting, editing or removing components inside without any negative effects on the running system. For example, YARN-based execution management component allows being added new computing

frameworks such as Storm, Giraph or MPI. In addition, the emergence of too many components with many different interfaces in the close-cluster-based platform can result in the confusions and difficulties in application management and development, so it is significant to develop a flexible system for access interface management in the platform.

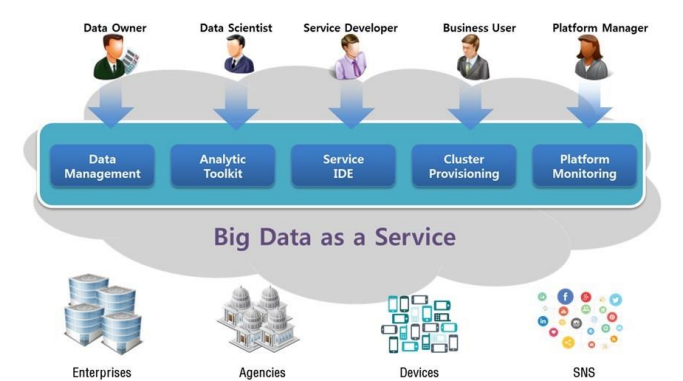


Figure 1. Collaborative Analytic Big Data Platform Concept

| Interface Management |                                                                                                                                                                                                                                                                                                                                                         |                       |                      |                               |                   |                                 |                                                             |  |     |     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|----------------------|-------------------------------|-------------------|---------------------------------|-------------------------------------------------------------|--|-----|-----|
| Data Collector       | Structured data (Sqoop)                                                                                                                                                                                                                                                                                                                                 |                       | General data (Flume) |                               | Log data (Chukwa) |                                 | Specialized data (Mobigen)                                  |  | ... |     |
| Analysis Engine      | pig                                                                                                                                                                                                                                                                                                                                                     | hive                  | ...                  | shark                         | stream            | ...                             | Presto-SQL<br>HBase Connector    Cassandra Connector    ... |  | ... |     |
| Computing Framework  | Batching (Hadoop Map Reduce)                                                                                                                                                                                                                                                                                                                            |                       |                      | Interactive-Streaming (Spark) |                   |                                 | Interactive (Presto)                                        |  | ... |     |
| Execution Management | Multi-tenant Execution (Hadoop YARN)                                                                                                                                                                                                                                                                                                                    |                       |                      |                               |                   | Single-tenant Execution (Mesos) |                                                             |  |     | ... |
| Data System          | Column (Hbase)                                                                                                                                                                                                                                                                                                                                          | Key-value (Cassandra) |                      | In-Memory (Redis)             |                   | Relational (MariaDB)            | Document (MongoDB)                                          |  | ... |     |
| File System          | Distributed FS (HDFS)                                                                                                                                                                                                                                                                                                                                   |                       |                      | Network FS (NFS)              |                   |                                 | Journaling FS (JFS)                                         |  | ... |     |
| Resources            | <div></div> |                       |                      |                               |                   |                                 |                                                             |  |     |     |

Figure 2. Software Stacks in Big Data Platform.

III. GATEWAY-BASED ACCESS INTERFACE MANAGEMENT

The overall architecture of our proposed system is presented in Figure 3. App is a set of applications including web application, mobile application, etc. which are developed by different developers to provide different services to endusers. Authentication Server is responsible for authenticating users and services based on different protocols such as Kerberos, OpenID and LDAP. Authorization Server is responsible for authorizing users and services to decide which interfaces they can access via Gateway. Gateway is used for forwarding requests from App to MetaManager and PlatformComponent. MetaManager which consists of many manager servers corresponding with applications in App takes charge of handling directly the valid requests and update metadata in Metadatabase. Metadatabase stores all metadata in the cluster such as user, group, quota, resource, application and cluster configuration. PlatformComponent includes different components from different big data systems such as HDFS,

Spark and Flume, which are responsible for collecting, analyzing and visualizing data.

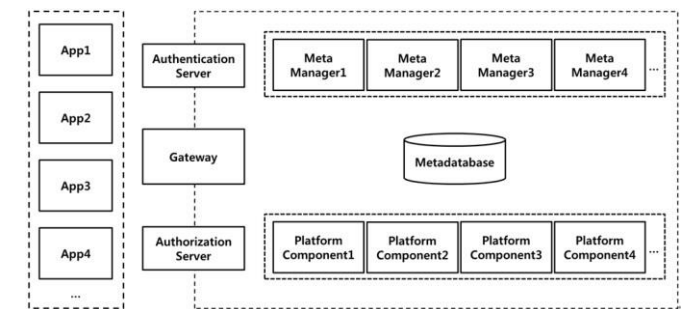


Figure 3. System Architecture of Gateway-based Access Interface Management Concept.

The metadata access procedure in our invented system can be explained as follows. In the first step, user sends an authentication request including user identity and password to Authentication Server through App. Based on the user information in Metadatabase, the Authentication Server checks whether there are any errors in the request information for authentication or not. If the answer is “Yes”, the user needs to perform again the first step. If the answer is “No” then the user now can send a metadata access request to a specific App. App then queries the access right information storing in Metadatabase by sending an authorization request to Authorization Server. The Authorization Server checks whether there are any access right violations or not. If the answer is “Yes”, the user needs to check and perform again. If the answer is “No”, App then sends the request to Gateway. After that, based on interface mapping list in Metadatabase, Gateway forwards the request to the correct MetaManager. Here, MetaManager accesses the metadata information in Metadatabase including reading/updating metadata according to the user request and returns the result to Gateway. Gateway then returns the result to App and finally the user can receive the result from App.

Similarly, data access procedure in our invented system includes the following steps. In the first step, user sends an authentication request including user identity and password to Authentication Server through App. Based on the user information in Metadatabase, the Authentication Server checks whether there are any errors in the request information for authentication or not. If the answer is “Yes”, the user needs to perform again the authentication. If the answer is “No” then the user now can send a data access request to a specific App. App then queries the access right information storing in Metadatabase by sending an authorization request to Authorization Server. The Authorization Server checks whether there are any access right violations or not. If the answer is “Yes”, the user needs to check and perform again the data access request. If the answer is “No”, App then sends the request to Gateway. After that, based on interface mapping list in Metadatabase, Gateway forwards the request to the correct PlatformComponent. Here, PlatformComponent performs the users’ requests such as data collecting, data processing and returns the result to Gateway. Gateway then returns the result to App and finally the user can receive the result from App.

On other hands, interface management procedure is elaborated as follows. In the first step, platform manager sends an authentication request including identity and password to Authentication Server through Gateway. Based on the platform manager information in Metadatabase, the Authentication Server checks whether there are any errors in the request information for authentication or not. If the answer is “Yes”, the platform manager needs to perform again the first step. If the answer is “No” then the platform manager now can send an interface update request to Gateway.

Gateway then queries the access right information storing in Metadatabase by sending an authorization request to Authorization Server. The Authorization Server checks whether there are any access right violations or not. If the answer is “Yes”, the platform manager needs to check and perform again the interface update request. If the answer is “No”, Gateway updates the interface mapping between frontend and back-end information storing in Metadatabase according to the Platform Manager request. Finally, Gateway then returns the result to the Platform Manager.

#### IV. CASE STUDY

In ETRI big data platform, the implementation of gatewaybased access interface management system is illustrated in Figure 4. We use KDC (Key Distributed Center) as Kerberos authentication server for both user and service authentication. We modified LDAP server to query user information directly from Metadatabase servers. All requests from users via application servers including ETRI Hue, Workflow, Portal, Monitoring and Provisioning are sent to Gateway which is responsible for filtering, authorizing and forwarding requests. Metadata or data accessing requests then are sent to correct components. Note that during operation, platform servers also need to interact with Metadata Management servers to update and retrieve metadata information in Metadatabase servers.

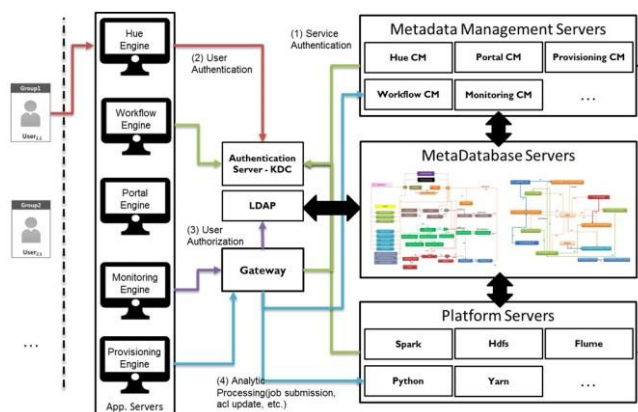


Figure 4. The implementation of Gateway-based Access Interface Management in ETRI Big Data Platform.

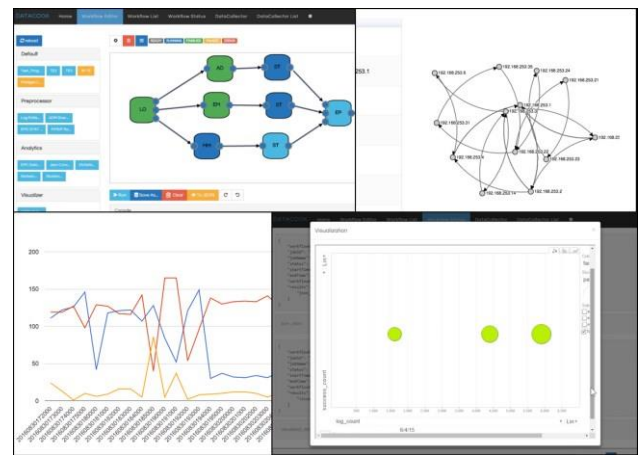


Figure 5. Analytic Results of Log Analytic Service.

Log analytic service is developed to detect abnormal activities in a cluster. The process of developing this service in our platform is as follows. Firstly, data owners need to import log data which are collected in real-time from all machines of their cluster into our platform. After that, they share this data with data scientists for researching how to extract abnormal activities in Portal Engine. Data scientists use Hue Engine to conduct their experiments and give results to service developers to develop different algorithm modules. After that, service developers use Workflow Engine to combine all modules and automate the service execution. Finally, Provisioning Engine can be used to support test, debug and deploy the log analytic service. Note that, due to gatewaybased access interface management system, all communications between the engines become easier and smoother. Figure 5 shows some analytic results of log analytic service.

#### V. CONCLUSIONS

In the paper, we introduced the system which can mitigate the aftermath of the heterogeneity of the interfaces provided by the several systems. The system not only can help users to obtain easily their desired functions but also can allow the platform managers to manage conveniently the interfaces in the platform. As the future work, we will conduct more experiments for the system evaluation and apply the optimization steps to enhance the performance of the proposed system.

#### ACKNOWLEDGMENT

This work was supported by ICT R&D program of MSIP/IITP. [B0101-16-0233, Smart Networking Core Technology Development]

#### REFERENCES

- [1] A. Labrinidis and H. V. Jagadish, “Challenges and opportunities with big data,” *Proc. VLDB Endowment*, 2012, vol. 5, no. 12, pp. 2032–2033.
- [2] Marcos D. Assunção et al., “Big Data computing and clouds: Trends and future directions,” *Journal of Parallel and Distributed Computing*, vol. 79, no. 80, pp. 3–15, May 2015.
- [3] Ibrahim Abaker Targio Hashem et al., “The rise of big data on cloud computing: Review and open research issues,” *Information Systems*, vol. 47, pp. 98–115, Jan 2015.

- [4] Yue-Shan Chang, Min-Huang Ho, Shyan-Ming Yuan, "A unified interface for integrating information retrieval," *Computer Standards & Interfaces*, 2001, vol. 23 no. 4, pp. 325-340..
- [5] Apache Camel. [Online]. Available: <http://camel.apache.org/>
- [6] MuleSoft. [Online]. Available: <http://www.mulesoft.com/>



**Minh Chau Nguyen** received the B.S. degree (2009) in Computer Science from the University of Sciences, Ho Chi Minh, Vietnam and the M.S. degree (2013) in Computer Science from the Korea Advanced Institute of Science and Technology (KAIST). He is currently a researcher of the Big Data Software Research Dept. at Electronics and Telecommunications Research Institute (ETRI), Daejeon, Rep. of Korea. His research interests include big data management, software architecture and distributed systems.



**Hee Sun Won** received the B.S. (1990) degree in Computer Science from Yonsei University, the M.S.(1992) and Ph.D (2016) degree in Computer Science from Korea Advanced Institute of Science and Technology (KAIST). From 1992 to 1999, she was a researcher in Korean Broadcasting System, where he participated in developing HDTV SW simulator and Data Broadcasting. She is currently a principal researcher of Big Data SW Research Dept. at

Electronics and Telecommunications Research Institute (ETRI). Her research interests include multi-tenant system, cloud resource management, big data analysis.