Multi-domain and Sub-role Oriented Software Architecture for Managing Scientific Big Data

Qi Sun, Yue Liu(✉), Wenjie Tian, Yike Guo, and Jiawei Lu

School of Computer Engineering and Science, Shanghai University, Shanghai, China
sunqichn@163.com, yliu@staff.shu.edu.cn, tianwenjie1997@163.com,
y.guo@imperial.ac.uk, jiaweifirst@gmail.com

Abstract. The existing Scientific Data Management Systems (SDMSs) usually focus on a single domain and the interaction pattern of each subsystem is complex. What's more, the heterogeneity and multi-source of Scientific Big Data (SBD), resulting in a wide variety of databases, scientific devices and functional areas, make the incompatibility and conflict between system modules inevitable. In this context, the paper focuses on the design and technology requirements of a multi-domain and sub-role oriented software architecture. Through integrating multiple databases, third-party systems and related tools, this architecture realizes both the storage and the sharing of multi-domain and multi-type SBD. Particularly, this architecture is divided into four independent functional areas and corresponding roles are designed, which enhances the decoupling and extensibility of the architecture. In addition, this paper has a formal description of the partition design from the perspective of role. On this basis, this paper also shows the typical application scenarios under different roles. The rationality and comprehensiveness of the proposed architecture are proved by describing the architectures design and technology.

Keywords: Software architecture · Role · REST · Scientific big data

1 Introduction

The development of instruments and computing facilities leads to the geometric increase of scientific data, for example, the Large Synoptic Survey Telescope (LSST) in the field of astronomy generates 15–30 TB raw data every night [9]; the Large Hadron Collider (LHC) in the field of high energy physics generates 40PB of experimental data each year [1]. Similar trends have also been observed in life sciences [3], earth sciences [11], biology [2], and so on. At the same time, due to the complexity of scientific experiments, SBD often presents the characteristics of high dimension and complex structure.

In recent years, many institutions have developed varied SDMS for managing SBD. Sequoia 2000 is a system for studying global change information such as

R. Ren et al. (Eds.): SDBA 2018, CCIS 911, pp. 111–122, 2019.
https://doi.org/10.1007/978-981-13-5910-1_10

environmental pollution and global warming [5], its large capacity, fast storage engine and visualization tools meet the practical needs of Geoscience; NASA and IBM have developed Paradise for managing large-scale geographic information data [4,15], focusing on data storage and processing technologies rather than data modeling or query; Gray developed SkyServer, a SDMS for SDSS data, to manage TB scale astronomical images and process data [8]. StoneBraker et al developed a data management and analysis software system, SciDB, which focus on the analysis and processing of SBD [12]. Aiming at the complexity of SBD computation, Apach presents a big data high-performance computing framework, Hama [13], which provides a global synchronous parallel computing model and a graph model for scientific computing. In China, especially during the 12th Five-Year Plan, the application of SBD has also made a lot of achievements, such as the neutrino experimental database and the animal subject database. In addition, there are a series of big data benchmarks for data testing to better understand the nature of scientific big data [6,7,10,16].

These SDMSs have played a great role in the management of SBD, but most of them are oriented to a single field and do not have access to third-party management systems, which means that there is no common management of multi-domain data and cross-domain data sharing. In addition, most of these systems are based on certain types of databases, such as relational databases, which are easy to use and maintain but cannot satisfy the requirement of storage and analysis, while non-relational databases are easy to extend, however, when storing and processing SBD, the natural model structure of the data is often changed, which makes subsequent data management and the development of corresponding analysis software too complicated. Finally, because of the complexity of the SBD, the interaction pattern of each module is very intricate, which results in the poor decoupling and expansibility of system, and the modification of one module may leads to the modification of several corresponding modules. These deficiencies create obstacles to the wider and more efficient use of SBD [14].

Therefore, the paper presents a multi-domain and sub-role oriented architecture, and analyses the architecture framework and technology selection from the technical perspective. In the second section of this paper, the design scheme of architecture will be put forward. Third section will formalize the role division of architecture, and describe the communication between roles in the architecture. The deployment of architecture and the selection of its technical requirements will be presented in section four. The fifth section will show the typical application scenarios of the architecture under different roles. The conclusion and future work will be given in seventh six.

2 Architecture of Scientific Big Data Management System

The architecture is consisted of four independent areas (see Fig. 1): Storage and Access Function Area (SAA), Analysis Function Area (AFA), Query Function

Area (QFA), and Basic Service Function Area (BSFA). The design and function of the four areas are independent of each other, and the decoupling of architecture is achieved by using RESTFul interfaces to integrate these areas.

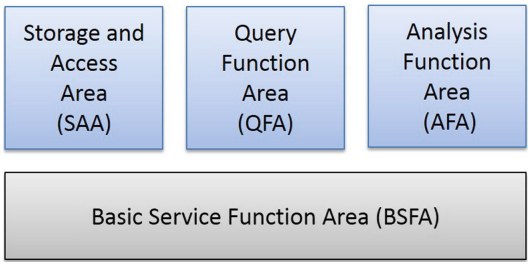

Fig. 1. Four areas of the architecture

Each areas has its own sub-modules and corresponding components. Figure 2 is the software architecture diagram, showing the internal composition of each area and interaction with other areas through interfaces. SAA is responsible for storing various types of SBD to solve the heterogeneous problem. AFA is composed of different scientific experimental tools, which are responsible for carrying out scientific experiments in various fields. The data in SAA and experiments in AFA can be queried and visualized in QFA. BSFA provides related functions to the above areas to maintain the operation of the entire architecture.

SAA is built on various types of databases and distributed file systems to store assets, including relational and non-relational SBD, algorithms such as data mining and domain methods, as well as External Asset Management System (EAMS). In addition, metadata and user information are stored here.

QFA plays a "bridge" role in the whole architecture. In this area, user can query and visualize the assets. It should be noted that QFA is composed of two sub-modules: an Asset Manager (AM) and a Visualizer. AM can be viewed as the "front face" of architecture, it supports a range of asset management operations, such as asset upload and download, quality testing and sharing setting, etc. The Visualizer supports the further display of assets through related devices.

Integration between these two sub-modules is accomplished in the following ways: in order to display asset information, the Visualizer requests the service "AM-Visualizer" through an interface, which is provided by AM, to get the assets and import them into chart generator.

AFA focuses on the scientific experimental analysis of assets by using the integrated methods in the system. The methods here include the special methods in various fields as well as the general machine learning algorithms. Scientists should choose the corresponding methods according to their needs to construct unique scientific experiments.

The fourth area is BSFA. As the base area of the whole architecture, BSFA provides system basic services and resource management functions to the other three areas, and maintains the operation of the architecture.

In the proposed architecture, the interoperability between all four areas is achieved by using RESTFul Interfaces and related services. AM is the requester of SAA service "SAA-AM", which can transfer assets stored in SAA to AM, while AM integrates operations related to asset management. The assets received by using service "SAA-AM" can be sent to AFA via the service "AM-AFA". AFA is not directly connected to SAA because all necessary assets from SAA needed for AFA are provided by AM. AFA can also use this service to transfer experimental results to QFA's Visualizer for visualization. On the other hand, SAA can obtain assets from EAMS through the service "EAMS", and these assets can also be transferred to AM and AFA through the corresponding services. It should be noted that only users with the same account number in this architecture and EAMS can share assets between EAMS and SAA. BSFA uses "BSFA-SAA","BSFA-QFA" and "BSFA-AFA" services to provide system services and resource management capabilities to the other three areas.

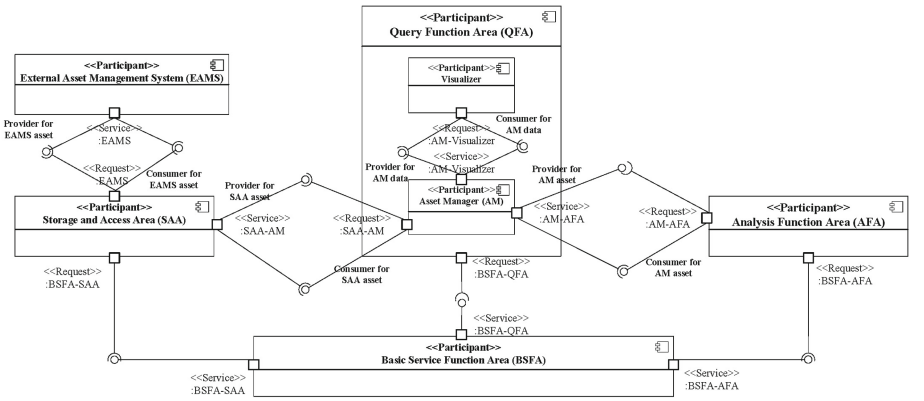

Fig. 2. Software architecture diagram shows the internal composition of each area and interaction with other areas through interfaces.

3 Formal Description of Architecture

The description of existing software architecture design is usually non-formal. In order to have a deeper understanding of architecture, this chapter formalizes the proposed architecture design from the perspective of role.

3.1 Asset

In the discussion of the architecture design, we will refer to scientific data, algorithm and external system collectively as assets and define them in the form of triples.

Definition 1. *Asset*

$$A = (data, algorithm, system) \tag{1}$$

3.2 Component

The concept of a component is defined on top of asset. Component is the basic unit of the architecture, the system developers usually don't need to understand the internal structure of the component, but only focus on the interaction between components. Through components, the architecture can be divided into multiple interactive subsystems, which enhance the independence, scalability and decoupling of the system.

Definition 2. *Component*

$$C = (A_1, \cdots, A_n, S(A_i)) \tag{2}$$

where $i \in [1, n]$, and $S(A_i)$ represents the structures or combinations of assets.

3.3 Interface

The interaction between components is achieved through interface that define the input and output rules of the component and treat the component as a blank box, so that users don't have to care about internal structure of the component.

Definition 3. *Interface*

$$I_{c_p, c_q} = (c_p, c_q), c_q, c_q \in C \tag{3}$$

where c_p represents the request input component, c_q represents the object component.

3.4 Role

For the four areas in the architecture described above, each area contains several components that are functionally similar or grouped together to accomplish a business logic, and the components within the area can interact with each other. The interface of a component which is open to outside is called the interface of that area. This chapter abstracts these areas into four roles, as shown in Table 1.

Definition 4. *Role*

$$R = (c_1, \cdots, c_n, S(c_j), i_{in}, i_{out}), j \in [1, n] \tag{4}$$

where i_{in} are the interfaces between components within the area, i_{out} are the interfaces for area interaction outside, also known as the interfaces for that role.

Table 1. Correspondence of areas and roles

Area	Role	Function
SAA, QFA	Asset owner	Uploading and managing assets
AFA, QFA	Scientist	Conducting scientific experiments
BSFA	System operators	Providing system basic services
ALL	System development	Providing development services

3.5 Architecture

Following the design of the architecture in Sect. 2 and the correspondence of areas and roles, the architecture's formal description is as follows.

Definition 5. *Asset Owner*

$$R(AssetOwner) = (c_R, c_{NR}, c_F, S(c_R, c_{NR}, c_F), i_{in}, i_{out}) \tag{5}$$

where c_R, c_{NR}, c_F represent relational databases, non-relational databases and file systems, which cover most forms of SBD, and

$$i_{out} = (Request : EAMS, Service : SAA - AM, Request : BSFA - SAA) \tag{6}$$

represents externally displayed interfaces.

Definition 6. *Scientist*

$$R(Scientist) = (C_s, S(C_s), i_{in}, i_{out}) \tag{7}$$

where

$$i_{out} = (Request : AM - AFA, Request : BSFA - AFA) \tag{8}$$

contains all requested services of AFA.

Definition 7. *System Operator*

$$R(SystemOperator) = (c_{BS}, c_{RM}, S(c_{BS}, c_{RM}), i_{in}, i_{out}) \tag{9}$$

where c_{BS}, c_{RM} are the related components of system basic services and resource management, and

$$i_{out} = (Service : (BSFA - SAA, BSFA - QFA, BSFA - AFA)) \tag{10}$$

represents its externally displayed interfaces.

Definition 8. *System Developer*

$$R(SystemDeveloper) = (R(AssetOwner, Scientist, SystemOperator)) \tag{11}$$

it shows that the System Developer is a union of three other roles.

Figure 3 shows the interaction between the various roles in this architecture: asset owners are responsible for providing data, algorithms, and other assets in various fields, then assets can be scientifically experimented by scientists and saved in the databases owned by the asset owners; the system operators are responsible for providing resource management functions, monitoring and security measures during this process; the above behaviours are implemented by system developers using a series of tools, and they could perform activities of the other three roles.

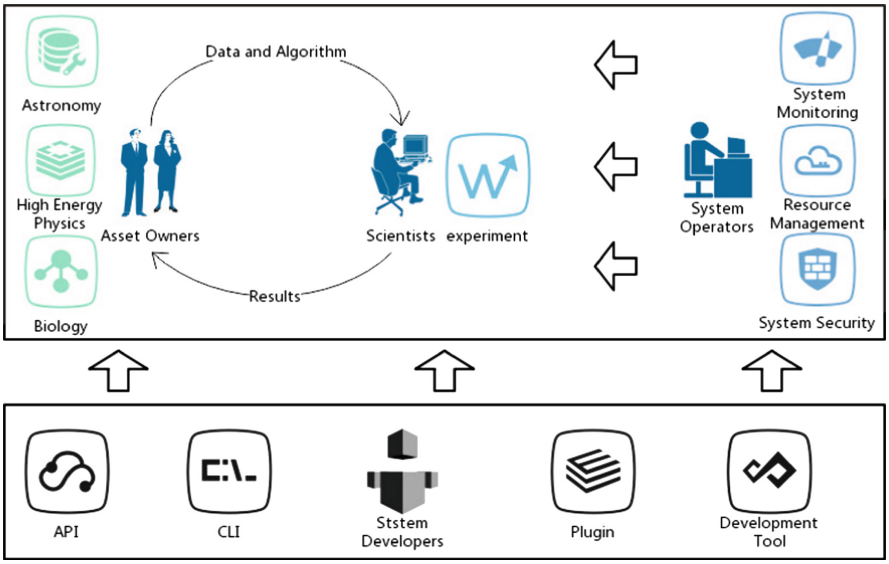

Fig. 3. Assets and information exchange are achieved through communication among various roles. This interactive form promotes the operation of the system.

4 Deployment Architecture as an Implementation of Design

This section describes the integration architecture for the previously defined architecture. The UML deployment diagram is shown in Fig. 4, which presents the main components and communication channels between four areas, along with elements that are used to facilitate communication between these areas.

SAA consists of three different types of databases and one file management system, all run in Linux OS: Mysql as a storage of structured data, MongoDB as a storage of unstructured data, gStore as a storage of RDF graph data and HDFS as a distributed file system. These four components cover almost all kinds of SBD. Since different types of data are stored in different databases or file

systems, an adapter named "SAA-QFA Adapter" is required to aggregate the data so that only one unified access interface is displayed externally, reducing the inconsistency caused by accessing different types of data.

SAA is also responsible for connecting with EAMS, in this architecture, EAMS is defined as a material machine learning platform, which uses machine learning methods to analysis material data. The platform is also connected to the SAA-QFA Adapter through an interface, so that the assets of EAMS are consistent with assets in SAA.

Similarly, QFA also consists of two subsystems: Bootstrap as AM, and "Chinese VisCloud" as Visualizer. Bootstrap use a series tools in its framework to operate assets; "Chinese VisCloud", a visual cloud platform of Shanghai University, consists of 48 screens on which multiple browser windows can be opened for cross-screen presentations. Because the data formats between the two components are not consistent, it is necessary to configure an adapter "Visualization Adapter" to exchange data over their respective interfaces.

AFA mainly uses Spark for SBD analysis. As a new generation of distributed processing framework, Spark's memory-based computing can speed up the execution of the algorithms, and it has an excellent machine learning library MLlib, which can be used as an auxiliary tool for data analysis. In addition, "QFA-AFA Adapter" can convert data passed from QFA to RDD format for Spark. Of course, scientists could also use the system integration algorithm to perform scientific experiments without using Spark.

From the deployment diagram, we can see that QFA connects to the "SAA-QFA adapter" in SAA through the interface"SAA-QFA Adapter API", which eliminates the differences between various types of databases when accessing assets. On the other hand, AFA connects to Bootstrap through "QFA-AFA Adapter" and interface"asset provider", so that AFA can get the asset stored in SAA. The visualization function of experiments can be directly implemented through the interface and adapter to show the experimental results to the "Chinese VisCloud" or Bootstrap.

Besides the above three areas, BSFA consists of three components: a Java security framework, Apache Shiro, which provides security features such as identity privilege verification and single sign-on for the architecture; a common resource management system, Apache YARN, which can provide unified resource management and scheduling for upper-layer applications, its introduction brings huge benefits to clusters in terms of utilization, unified resource management, and data sharing; a web-based tool, Apache Ambari, which supports cluster provisioning, management, and monitoring. These functions and services are supported through the "BSFA Adapter" and"Basic service provider" for other areas.

5 Architecture Application Scenario

This section will demonstrate and illustrate the typical application scenario of the above architecture under different roles.

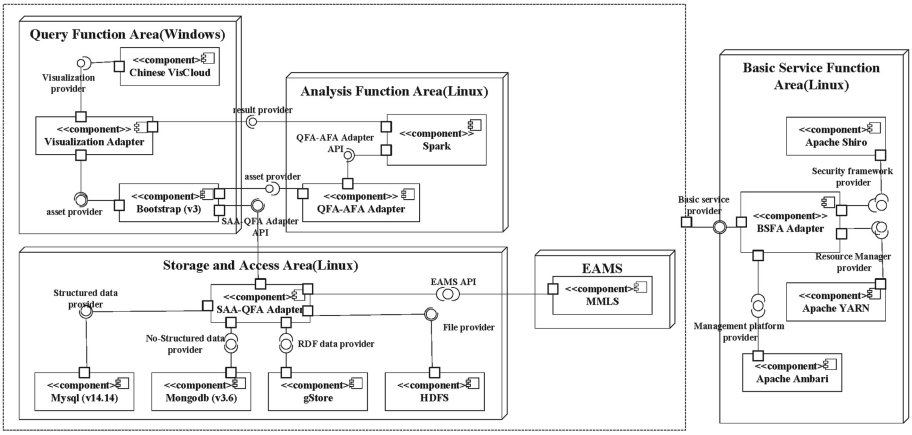

Fig. 4. The UML deployment diagram presents the main components, communication channels and technological requirements of four areas.

5.1 Conducting Scientific Experiments by Scientists

The main goal of scientists conducting scientific experiments is to make a good decision by constantly changing the combination of data and algorithms and observing the experimental results. Scientists often run different script files on the local machine to combine data and algorithms to form different scientific experiments. They may also be necessary to obtain corresponding data from different databases, and it will make the preparation of scientific experiments very tedious. The architecture integrates various types of data and algorithms in different fields. From the scientists' point of view, these assets are the same, and there is no difference due to storage in different databases. At the same time, the scientists only need to select the data and algorithms, and fill in the necessary parameters. It is not necessary to modify and debug the script as before, which greatly enhances the efficiency and user experience of scientists in conducting scientific experiments.

Scientists' process of conducting scientific experiments in the architecture is as follows: first of all, scientists should register the experimental information, including experiment name, type, number of steps, running environment, and so on; afterwards, scientists could freely select the required data and algorithms on the page, no matter what domains of these assets, and no matter whether stored in a relational or non-relational database, scientists just need to click the select and search box; finally, the experiment is started by clicking the "Start Analysis" button; the results of the experiment will be displayed on the page, scientists can choose to view and save it. Figure 5 shows scientists' activity model and the steps that AFA take in the above process.

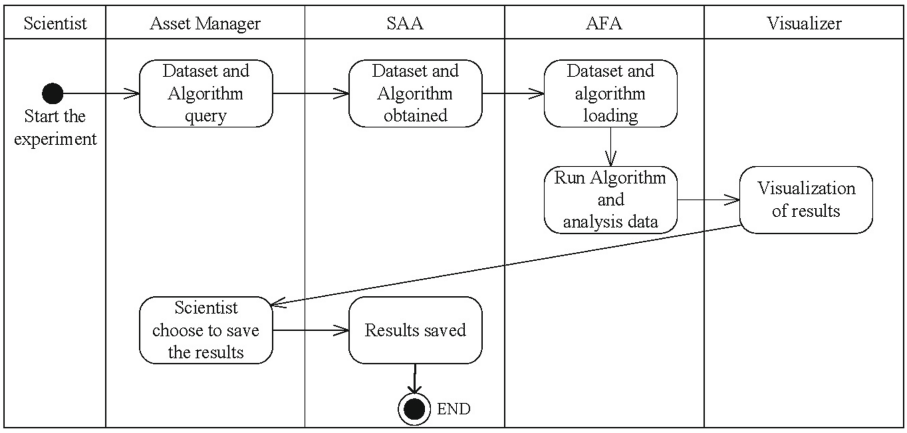

Fig. 5. Scientists' activity model related to conducting scientific experiments, and the steps that each area takes in the model.

5.2 Data Quality Detection by Asset Owner

Data is the carrier of information, and the quality of data is of great significance to correctly reflect the scientific significance of it and to effectively support decision-making. On the basis of the above research, this paper adopts the following data quality detection methods for scientific data: field incompleteness detection, numerical field detection and data set sampling survey.

The first on is field incomplete detection. The incompleteness of a field is divided into two cases: unassigned and undefined. Unassigned corresponding field content is empty, and undefined indicates that the field value is unknown (NULL). In fact, an empty field is a valid character, and NULL is a special kind of value, unlike a zero-length string. In this architecture, the above two cases are treated as incomplete fields.

The second one is numeric field detection. Because most of the scientific data are numerical, a series of statistical methods are used to check whether the data are within a reasonable range and the distribution of the data, etc. These methods include average, inter-quartile, standard deviation, skewness, and upper and low limit method.

The last one is data set sampling survey. A sample survey takes a small number of samples, conducts an actual investigation of them, and verifies the authenticity of the data. Sampling includes simple random sampling, stratified random sampling, cluster sampling, system sampling, etc. Considering the differences between SBD, simple random sampling is chosen as the sampling method.

5.3 Providing System Services by System Operators

While the system operators are responsible for system basic services and resource management functions, they can view various information during the current system runtime, such as system environment variable information, system configuration attributes, operating system information, and file system information. The system performance information, like CPU utilization (user utilization, system utilization, and total utilization), memory and swap usage, and network situation can also be observed. Through the above information, the system operators could clearly understand the operation of the current system, and could timely capture the abnormal information for processing.

5.4 Super Administrator for System Developers

In this architecture, the system developers are similar to a super administrator and have all rights of scientists, asset owners, and system operators. They have all functions and can perform user management, rights configuration, and other activities.

6 Conclusion

This paper proposes a software architecture model to manage multi-domain and heterogeneous SBD. Asset storage, analysis, visualization, and external systems access are integrated into one architecture through different services which are implemented by RESTFul interfaces. In addition, this paper formalizes the design of the architecture from the perspective of role to have a deeper understanding of the architecture in the future.

The highlights of the proposed architecture are: (1) Integration of multiple types of databases and provision of external system access capabilities to manage and share multi-source heterogeneous scientific data; (2) the role division of the architecture is proposed to describe the formal definition of each element in the architecture design; (3) this paper shows the architecture's typical application scenarios form the perspective of role, and solves the problems of management and utilization of SBD to some extent.

In future research, we will use formal, systematic, and standardized evaluation methods to evaluate the proposed architecture. The process of scientific experiment will be focused to further design and improve the four areas. In addition, based on the evolution of SBD, the origin and development of data will be further reflected in the architecture.

Acknowledgement. This work is supported by the National Key Research and Development Plan of China (Grant No. 2016YFB1000600 and 2016YFB1000601).

References

1. Andreeva, J., Campana, S., Fanzago, F., Herrala, J.: High-energy physics on the grid: the atlas and CMS experience. J. Grid Comput. **6**(1), 3–13 (2008)
2. Bengtssonpalme, J., et al.: Strategies to improve usability and preserve accuracy in biological sequence databases. Proteomics **16**(18), 2454–2460 (2016)
3. Cook, C.E., Bergman, M.T., Cochrane, G., Apweiler, R., Birney, E.: The European bioinformatics institute in 2017: data coordination and integration. Nucleic Acids Res. **46**(D1), D21 (2018)
4. Dewitt, D.J., Kabra, N., Luo, J., Patel, J.M., Yu, J.B.: Client–server paradise. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 558–569 (1994)
5. Dozier, J., Stonebraker, M., Frew, J.: Sequoia 2000: a next-generation information system for the study of global change. In: Proceedings Thirteenth IEEE Symposium on Mass Storage Systems. Towards Distributed Storage and Data Management Systems, pp. 47–53 (1994)
6. Gao, W., et al.: Data motif-based proxy benchmarks for big data and AI workloads. In: IISWC 2018 (2018)
7. Gao, W., et al.: Data motifs: a lens towards fully understanding big data and AI workloads. In: 27th International Conference on Parallel Architectures and Compilation Techniques, PACT 2018 (2018)
8. Ivanova, M., Nes, N., Goncalves, R., Kersten, M.: MonetDB/SQL meets skyserver: the challenges of a scientific database. In: International Conference on Scientific and Statistical Database Management, p. 13 (2007)
9. Ivezic, Z., et al.: LSST: from science drivers to reference design and anticipated data products. Am. Astron. Soc. **41**, 366 (2008)
10. Jia, Z., et al.: Understanding big data analytics workloads on modern processors. IEEE Trans. Parallel Distrib. Syst. **28**(6), 1797–1810 (2017)
11. Jun, C., Wen, W., Zi-yang, L., An, L.: Landsat 5 satellite overview. Remote Sens. Inf. **43**(3), 85–89 (2007)
12. Stonebraker, M.: Scientific data bases at scale and SciDB. Anal. Proc. **4**, 199–206 (2013)
13. Suchanek, F.M., Weikum, G.: Knowledge bases in the age of big data analytics. VLDB Endowment (2014)
14. Szalay, A.S., Gray, J., Fekete, G., Kunszt, P.Z., Kukol, P., Thakar, A.: Indexing the sphere with the hierarchical triangular mesh. Microsoft Research (2007)
15. Team, C.T.P.: Paradise: a database system for gis applications. In: ACM SIGMOD International Conference on Management of Data, p. 485 (1995)
16. Wang, L., et al.: Bigdatabench: a big data benchmark suite from internet services. In: IEEE International Symposium on High Performance Computer Architecture, HPCA 2014 (2014)