PUBLICATION AND REUSE OF BIG DATA APPLICATIONS AS SERVICES

Oleg Sukhoroslov

Institute for Information Transmission Problems of the Russian Academy of Sciences

Abstract: The report considers development of domain-specific web services for processing of large volumes of data on high-performance computing resources. The development of these services is associated with a number of challenges, such as integration with external data repositories, implementation of efficient data transfer, management of user data stored on the resource, execution of data processing jobs and provision of remote access to the data. An approach for building big data processing services on the base of Everest platform is presented. The proposed approach takes into account the characteristic features and supports rapid deployment of these services on the base of existing computing infrastructure.

Keywords: big data, web services, data transfer, data management, distributed data processing, Hadoop, MapReduce

The explosive growth of data, observed in a variety of areas from research to commerce, requires the use of high-performance resources and efficient means for storing and processing large amounts of data. During the last decade, the distributed data processing technologies like Hadoop and Spark are emerged. However, the complexity of the hardware and software infrastructure prevents its direct use by non-specialists, and requires the creation of user-friendly tools to solve particular classes of problems. One way of implementing such tools is the creation of domain-specific services based on the Software as a Service model.

Data-intensive services (DIS), in comparison to conventional computational services with a small amount of data, started to develop recently, so the principles and variants of implementation of these services are poorly understood. There is a lack of best practices for implementation of DIS on the basis of the existing infrastructure for big data processing such as a cluster running Hadoop or Spark platforms which are increasingly used for the analysis of scientific data. Also, little attention is paid to the integration of DIS with existing repositories and data warehouses, including the cloud-based ones, as well as other services. Finally, there is a lack of platforms for implementation and

deployment of DIS that would provide ready-made solutions of typical problems encountered when creating this kind of services.

Consider typical requirements to DIS that represent remotely available services for solving a certain class of problems with a large amount of input data. Such services should provide remote interfaces, usually in the form of a web user interface and application programming interface (API). The interface must allow the user to specify the input datasets and parameters of the problem being solved in terms of subject area.

DIS must use high-performance and scalable (normally distributed) implementations of data analysis algorithms, requiring appropriate computing infrastructure for data processing and storage. Such infrastructure is generally represented by one or more computing clusters running Hadoop platform or a similar technology. DIS must translate the user request into one or more computing jobs that are submitted on a cluster and use scalable implementations (e.g., based on MapReduce) of perspective algorithms.

The user must be able to pass arbitrary input data to DIS. If the data is initially located on the user's computer or external storage resource (e.g., a data repository) DIS must implement the transfer of data over a network to the used cluster. When transferring large amounts of data it is important to ensure the maximum transfer rate and automatic failover. Since the process of working with big data is often exploratory, requiring multiple invocations of DIS, the service should support reuse of data loaded to the cluster. In order to optimize the use of network DIS must also cache frequently used datasets on the cluster. Data transfer functions can also be implemented as separate auxiliary services.

Importantly, DIS may operate separately from computing resources used for real data processing. DIS can use multiple resources, that can be situated at different locations. It is also possible that the service uses the resources provided by the user. In such cases it is important for reasons of efficiency to avoid passing the input data from the user to the resource through the service and to transmit the data directly.

In practice, the data analysis is often a multi-step process that requires performing different tasks at different stages of the analysis. In such cases, the results produced by one DIS can be passed as the input to another service. If these services use different resources, there also arises a problem of data transmission between resources. In general DIS should allow the user to download the output to his computer or an external resource, as well as to transfer the data directly to another service. In addition, DIS may

provide additional functionality for remote data preview and visualization. These functions may also be implemented as separate auxiliary services.

DIS must support the simultaneous use by multiple users. This requires the protection of user data, resource distribution between users and isolation of computational processes. In the case of cloud infrastructure, DIS must also manage dynamic allocation and deallocation of resources in the cloud, according to the current load.

Everest [1] is a web-based distributed computing platform. It provides users with tools to quickly publish and share computing applications as services. The platform also manages execution of applications on external computing resources attached by users. In contrast to traditional distributed computing platforms, Everest implements the Platform as a Service (PaaS) model by providing its functionality via remote web and programming interfaces. A single instance of the platform can be accessed by many users in order to create, run and share applications with each other. The platform implements integration with servers and computing clusters using an agent that runs on the resource side and plays the role of mediator between the platform and resources. The platform is publicly available online to interested users [2].

The advantage of using Everest platform to create DIS is the availability of ready-made tools for rapid deployment of computational services and integration with computing resources that do not require a separate installation of the platform. At the same time, since the platform was originally created to support services with a small amount of data, the effective implementation of DIS on the base of Everest requires a number of improvements. In particular, it is necessary to implement support of direct data transfers from external storage to the resource and vice versa, bypassing the platform. In addition, it is required to implement the integration of the agent with the components of Hadoop platform platform or similar technology used for data storage and processing on the cluster.

Figure 1 presents the proposed scheme of implementation of DIS on the base of Everest platform and existing Hadoop cluster. We plan to add to the agent support for loading data from major types of repositories and storage. Currently the basic support for downloading files via HTTP and FTP protocols, as well as an experimental integration with Dropbox and Dataverse repository are implemented. The downloaded input data is placed in Hadoop Distributed File System (HDFS). The submission of data processing jobs is performed via the cluster resource manager such as Yet Another Resource

Negotiator (YARN). A special adapter was implemented in order to support interaction of Everest agent with YARN, similar in function to the previously created adapters for integration with HPC batch schedulers. Upon a job submission the agent monitors the state of the job and transmits the progress information to the service. On job completion the agent transmits to the service the output files (of small size) and the final status of the job. If the output data produced by the job is of big size, the agent should support direct upload of such data over the network to a specified external storage. The necessary information should be provided by the user when sending the request to the service. At the moment, the upload of output data to the specified FTP server or Dropbox folder is implemented.

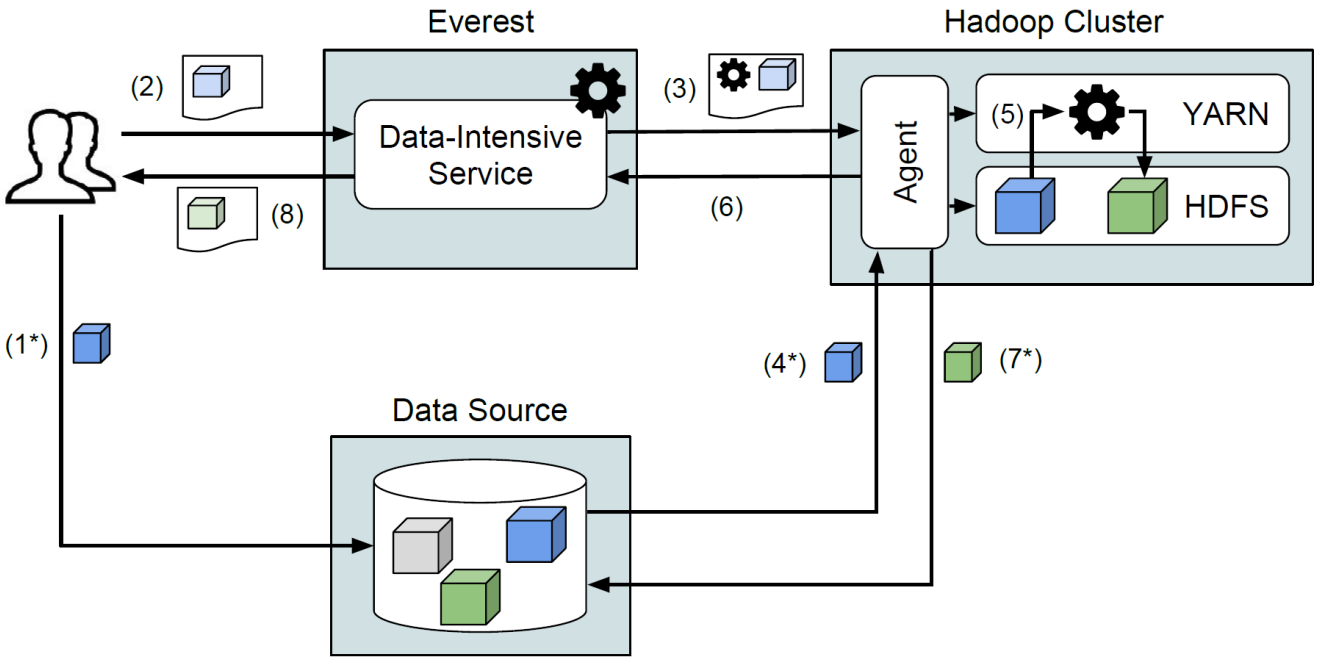

Figure 1. Implementation of DIS on the base of Everest platform.

References

[1] Sukhoroslov O., Volkov S., Afanasiev A. A Web-Based Platform for Publication and Distributed Execution of Computing Applications // 14th International Symposium on Parallel and Distributed Computing (ISPDC). IEEE, 2015, pp. 175-184.

[2] https://everest.distcomp.org/