A Service-Oriented Infrastructure for Teaching Big Data Technologies

Oleg Sukhoroslov ✉

[1] Institute for Information Transmission Problems
of the Russian Academy of Sciences (Kharkevich Institute), Moscow, Russia
[2] Yandex School of Data Analysis, Moscow, Russia
[3] Higher School of Economics, Moscow, Russia
sukhoroslov@iitp.ru

Abstract. The paper presents an experience in incorporating Big Data technologies into introductory parallel and distributed computing courses and building a service-oriented infrastructure to support practical exercises involving these technologies. The presented approach helped to provide a smooth practical experience for students with different technical background by enabling them to run and test their MapReduce and Spark programs on a provided Hadoop cluster via convenient web interfaces. This approach also enabled automation of routine actions related to submission of programs to a cluster and evaluation of programming assignments.

Keywords: Big data · Parallel programming · Distributed computing · Hadoop · MapReduce · Spark · Web-based interfaces · Web services

1 Introduction

The explosive growth of data observed in a variety of areas from research to commerce, commonly referred to as the Big Data phenomenon, 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 models such as MapReduce [1] and technologies like Hadoop [2] and Spark [3] are emerged. Modern HPC systems such as clusters are being increasingly used for running data-intensive applications in science and technology. Therefore there is a growing demand to incorporate relevant programming models and technologies into a parallel and distributed computing (PDC) teaching curriculum.

The introduction of Big Data technologies in a PDC course brings a number of challenges. First, these technologies are noticeably different from traditional parallel programming technologies (e.g., MPI), by using other programming languages (e.g, Java, Scala or Python) and computing models (e.g., MapReduce or Spark RDD). The Big Data applications are also quite different from the traditional HPC applications, which often motivates the development of specialized courses. Second, currently it is not possible to easily collocate Big Data

and HPC applications on a single computing cluster due to incompatible resource managers and resource allocation policies. This necessitates the provision of dedicated computing infrastructure for such applications, e.g., Hadoop cluster. Third, the implementation of practical exercises is challenging due to the inherent complexity of involved systems and user interfaces. This is particularly true for undergraduate or non-technical students without prior Linux background. While the similar problem exists for traditional HPC systems, Big Data systems have specific interfaces that should be taken into account.

This paper reports an experience on solving the mentioned challenges while teaching two introductory PDC courses at the Yandex School of Data Analysis (YSDA) and the Higher School of Economics (HSE). The Parallel and Distributed Computing course at YSDA is an introductory PDC course for MSc students that features the following topics: concurrency, parallel programming and distributed data processing. The similar course in HSE is for BSc students from the Faculty of Computer Science. Both courses consider distributed computing models and platforms for processing of large data sets.

In particular, the paper describes the software infrastructure and high-level web services implemented in order to support practical exercises involving Big Data technologies. The presented service-based approach helped to provide a smooth practical experience for students with different technical background by enabling them to run and test their programs on a Hadoop cluster via convenient web interfaces. This approach also enabled automation of routine actions related to submission of programs to a cluster and evaluation of homework solutions.

The paper is structured as follows. Section 2 discusses related work. Section 3 provides an overview of the developed infrastructure. Section 4 describes the computing infrastructure and how it was adapted to accommodate both HPC and Big Data applications. Section 5 provides an overview of Everest, a web-based distributed computing platform used for building the presented services. Section 6 describes the generic services for running MapReduce and Spark programs and the problem-specific services for evaluating solutions of related programming assignments. Section 7 concludes and discusses future work.

2 Related Work

The use of web technologies for building convenient interfaces to HPC systems has been exploited since the emergence of the World Wide Web. For example, in [4] authors describe several prototypes of web-based parallel programming environments, including the Virtual Programming Laboratory (VPL) used for teaching parallel programming. The emergence of grid computing and the web portal technology enabled development of grid portals facilitating access to distributed computing facilities. For example, [5] describes an experience of building a grid portal to support an undergraduate parallel programming course.

The web-based interfaces have also been exploited to support submission and automated evaluation of programming assignments in PDC courses. For example, in [6] authors describe a framework enabling implementation of web portals

for automated testing of student programming assignments in distributed programming courses. Among the recent works, [7] describes a web-based application for automated assessment and evaluation of source code in the field of parallel programming. In [8] authors present a similar web-based system for running and validating parallel programs written in different programming paradigms.

The web technologies are also being actively used nowadays for supporting Massive Open Online Courses (MOOC) with a large number of attendees. For example, WebGPU is a web-based system developed to support GPU programming assignments in the Heterogeneous Parallel Programming course [9]. In [10] authors describe the "Introduction to Parallel Computing" course that is developed on the base of Moodle learning management system and supports automatic evaluation of parallel programs.

While the previously mentioned systems support teaching traditional PDC topics, currently there exists only a few web-based environments focused on teaching Big Data technologies. The only similar project is the WebMapReduce (WMR) [11], which provides a simplified web interface to Hadoop designed for teaching the MapReduce computing model. The WMR portal allows students to write mappers and reducers in a variety of languages. The programs are executed on a Hadoop cluster or in a testing environment that mimics the behavior of Hadoop while running within a single thread. In contrast to WMR, the presented infrastructure is more generic by supporting other technologies and computing models beyond MapReduce, e.g., Spark, and addressing additional challenges such as automated evaluation of homework assignments.

In addition, a variety of open source and commercial systems are currently emerging that provide convenient web interfaces for working with Big Data technologies [12–14], including interactive notebooks and dashboards. While not specifically designed for teaching, these systems can also be used in educational activities. The presented infrastructure relies on one of such interfaces, namely Hue [12], for browsing the data stored on a Hadoop cluster.

3 Infrastructure Overview

A high-level overview of the infrastructure used to support practical exercises in the mentioned courses is presented on Figure 1.

The computing infrastructure consists of a dedicated cluster with 20 nodes which is split into two partitions for running HPC and Big Data workloads. The students can optionally request a direct access to the cluster command line via SSH. However, the default way to access the cluster is via a set of provided web services that automate submission and execution of parallel programs on the cluster. There are two main types of such services. The so called generic services can be used to run arbitrary programs for some technology, e.g., MPI or MapReduce. There are also problem-specific services that can be used for submission and evaluation of solutions for homework assignments. The services are developed and deployed on Everest, a web-based distributed computing platform [15, 16] which supports integration with computing resources via special software

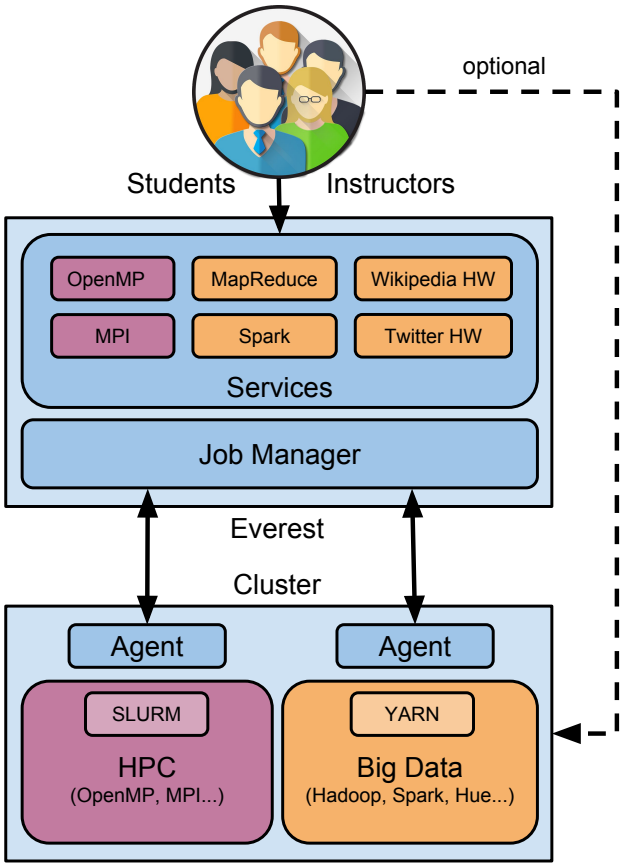

Fig. 1. Architecture of supporting computing and software infrastructure

agents. These agents are deployed on the cluster and are used by the Everest job manager for execution of programs submitted via the services.

The main advantage of the service-based approach is the ease of use and ubiquity in comparison to the command line environment. Such environment and queuing systems used on the cluster are unfamiliar and too low-level for many students. The execution of programs of the cluster also implies manual copying of required files that can be automated by the services, which is very convenient for quick demonstrations in class. Another advantage of the presented approach is the reduced administration overhead, since it does not require creation of cluster accounts for each student. The management of students in Everest can be automated by creating a dedicated user group and configuring a secret code for self-registration by the students. Finally, the use of special services for evaluation of homework assignments can provide an instant feedback for the students which enhances the learning experience.

4 Computing Infrastructure

As was previously mentioned, Big Data and HPC technologies use different resource managers and resource allocation policies. For example, while the execution of MPI applications on a cluster is usually managed by the batch system such as SLURM or PBS, the execution of MapReduce programs is managed by the YARN service, which is a part of Apache Hadoop platform. Also, while MPI programs allocate and use a fixed subset of cluster resources, MapReduce programs can dynamically allocate and release resources during their execution. Therefore it is very hard to use a single resource manager for both types of workloads.

To accommodate both HPC and Big Data applications the cluster was split into two separate partitions. The first partition, managed by the SLURM batch system and using the NFS file system, is dedicated for running HPC applications such as MPI and OpenMP programs. The second partition, managed by the YARN service and using the HDFS file system (also a part of Hadoop platform), is dedicated for running Big Data applications such as Hadoop MapReduce and Spark programs. The second partition also has a number of other Big Data technologies installed such as Hive, HBase and Kafka.

Having two separate cluster partitions brought an issue of efficient cluster utilization when one of the partitions is underutilized, for example when the students study MPI programming and use only the HPC partition. Currently the size of each partition can be changed by the administrator by manually stopping and starting the SLURM and YARN daemons on the cluster nodes. Given the known schedule of practical exercises by different courses using the cluster, the manual tuning of partition sizes proved to be sufficient. However, a more sophisticated automated tuning based on a current load can also be implemented in the future.

Both partitions have configured limits of resource usage per program which is essential in order to avoid the excessive use of cluster resources by inefficient or misbehaving programs. The HPC partition imposes a limit on the wall clock time used by a program. However, it is not possible to use a similar metric for Big Data applications since their run time can depend on the current cluster load. Therefore an alternative metric of consumed core-seconds was used to limit the resource consumption for the second partition. Since Hadoop YARN doesn't support enforcement of resource usage limits, a special script was developed that periodically checks the current resource consumption of running programs and kills those that exceeded the configured limits. The preemption in YARN scheduler is turned off in order to ensure stable execution and measurements, especially for Spark programs.

5 Everest Overview

Everest [15] is a web-based distributed computing platform used for building the services of the described infrastructure. In this section we provide a brief overview of this platform.

Everest provides users with tools to quickly publish and share computing applications as web 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 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 is available online to all interested users [16].

Everest supports development and execution of computing applications following a common model. An application has a number of inputs that constitute a valid request to the application and a number of outputs that constitute a result of computation corresponding to some request. Upon each request Everest creates a new job consisting of one or more computational tasks generated by the application according to the job inputs. The tasks are executed by the platform on computing resources specified by a user.

To simplify creation of applications Everest provides a generic skeleton for command-line applications that makes it possible to avoid programming while adding an application. In addition to description of application inputs and outputs, the user should specify the command pattern parametrized by input values and describe the mappings between inputs/outputs and files read/produced by the application.

An application is automatically published as a RESTful web service with a unified interface. This enables programmatic access to applications, integration with third-party tools and composition of applications into workflows. The platform's web user interface also generates a web form for running the application via web browser. The application owner can manage the list of users that are allowed to run the application.

Instead of using a dedicated computing infrastructure, Everest performs execution of application tasks on external resources attached by users. The platform implements integration with standalone machines and clusters through a developed program called *agent*. The agent runs on the resource and acts as a mediator between it and Everest enabling the platform to submit and manage computations on the resource. Everest manages execution of tasks on remote resources and performs routine actions related to staging of input files, submitting a task, monitoring a task state and downloading task results.

6 Services

A number of web services have been developed using the Everest platform in order to simplify and automate execution of various types of parallel programs by the students on the cluster.

In order to create an application an instructor should specify via Everest Web UI application's metadata, input and output parameters, mapping of parameters to the executed command and files, etc. The core part of the application is a wrapper that takes input parameters and manages execution of a parallel pro-

gram on the cluster. The wrapper can be written in any programming language since Everest runs it via command line. It usually performs program compilation, preparing of execution environment, submitting the program via queuing system, etc. The development of such wrapper is currently the most difficult part of the process, however once implemented its parts can be reused for other applications.

6.1 Generic Execution Services

The following generic services have been developed for execution of different types of programs using Big Data technologies on the cluster. These services can be used to run an arbitrary program of some specific type.

Two generic services were implemented for running Hadoop MapReduce programs. The first service supports programs written in Python using the Hadoop Streaming interface, targeting students without Java skills. The submit form of this service is presented on Figure 2. The second service supports Java programs using the Hadoop Java API. Both services allow specifying program files, command line arguments, input and output paths in HDFS file system, number of reduce tasks and additional Hadoop options. The wrapper script performs submission of MapReduce job, monitors the job's state and updates status information displayed in Everest. When the job is running, a student is provided with a link to the job status page in the Hadoop web interface. After the job is completed the total resource usage in core-seconds is displayed along with a link to the job history interface with task logs. This provides enough information to troubleshoot failed programs or evaluate the program's efficiency.

Two similar services were implemented for running Apache Spark programs written in Python or Scala/Java on the cluster. In comparison to the MapReduce services, the Spark services have more sophisticated runtime parameters such as the number of executors, cores and memory per executor. It is also possible to specify the minimum ratio of registered executors to wait for before starting computations. This enables students to examine various trade-offs related to using different values of runtime parameters. The corresponding wrapper script is also more sophisticated. It allows to limit the maximum amount of physical resources requested by the program and the number of concurrent jobs per user. The wrapper script also computes the effective resource usage for a Spark program by excluding core-seconds spent while waiting for the executors.

Upon the program submission the student is redirected to the job page that displays dynamically updated information about the job state. The job page also includes sections containing general information about the job, inputs specified by the student and outputs produced by the job. For teaching purposes the services were configured to automatically share all jobs submitted by the students with the instructors group, so that in case of a problem a student can just send a link to a failed job to the instructor.

Due to the large size of input data and produced results, in addition to running programs on Hadoop cluster it was essential to provide a way to easily browse files stored in the HDFS file system without fully downloading them.

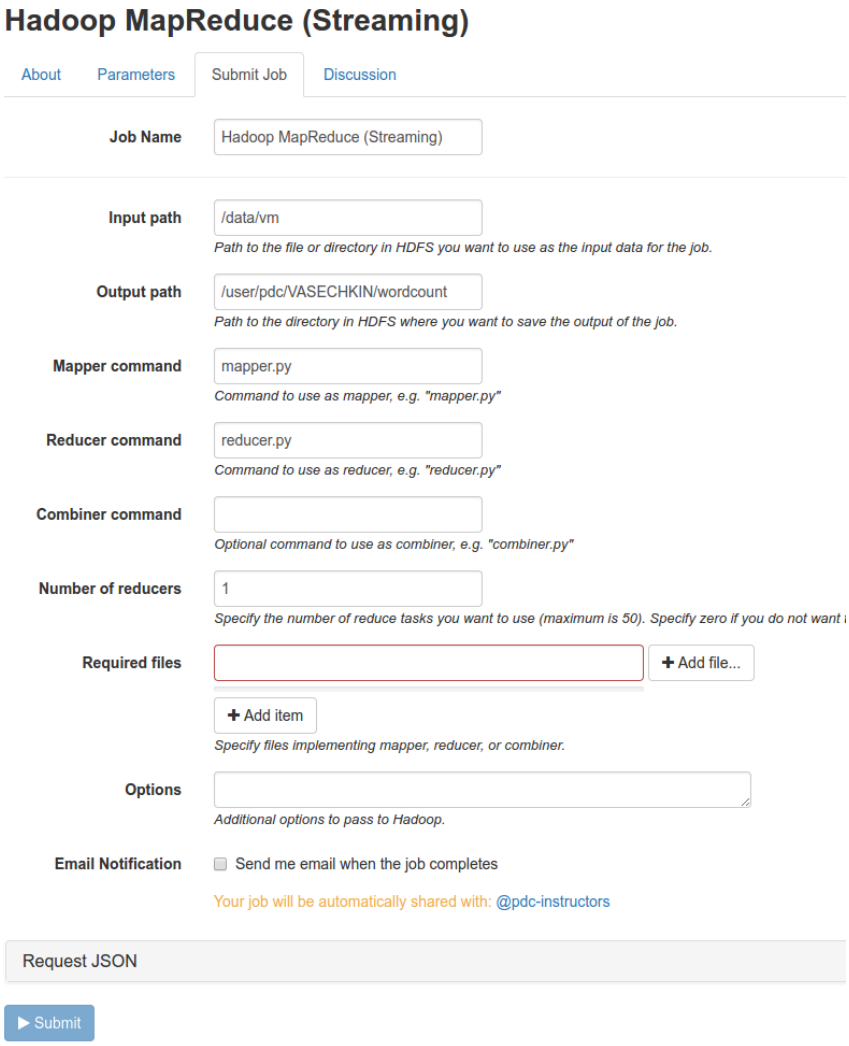

Fig. 2. The submit form of the service for running Hadoop MapReduce programs

This was achieved by using Hue [12], a web interface for Hadoop which includes a convenient HDFS file browser. Hue also provides a web interface for running jobs, however it is more complicated and low level in comparison to the developed services.

6.2 Services for Programming Assignments

The evaluation of programming assignments requires a significant effort and is one of the key scalability bottlenecks in terms of a number of students. The

generic services described above can be used for quick demonstrations, practical exercises and projects. However, they usually do not provide a feedback needed to validate solutions to programming assignments. For example, whether the program produced a correct result or has a good performance. Such immediate feedback is crucial for students since it helps to avoid manual validation and to focus on the solution. This feedback can also help instructors to reduce the time and effort needed to grade the solution.

A set of problem-specific services have been implemented for automated evaluation of homework assignments related to Big Data technologies. These services are implemented on Everest using the same approach as the previously discussed generic execution services. However, in this case the wrapper is replaced by a test suite for the given assignment.

The first assignment is dedicated to the MapReduce programming model and its implementation in Hadoop. The students should use MapReduce to build inverted index of the contents of Wikipedia pages. The solution of this assignment requires multiple MapReduce steps such as computing a list of frequent words excluded from the index and building an index itself for English and Russian versions of Wikipedia. Since it is difficult to implement an interface for specifying and running all these steps, the provided service doesn't perform the execution of solutions and only checks the provided results. The student should pass to the service the HDFS paths to the produced indexes. The service runs a script that checks that the index conforms to all requirements specified in the assignment. The students should include the link to test results in the homework report along with the links to all program runs via generic MapReduce services used to build the indexes. Instructors can view all programs created by a student by following these links in Everest. The generic services and job history web interfaces provide enough information to evaluate the efficiency of each program.

The second assignment is dedicated to using Apache Spark and its Resilient Distributed Datasets (RDD) programming model. The students should compute a number of results given a graph of follower relationships between Twitter users, such as the average count of followers, the most popular users and the number of users that can be reached by a tweet from popular users. The solution of this assignment also requires multiple steps, however, in contrast to MapReduce, these steps can be run as a single job in Spark. Nevertheless, to provide the maximum flexibility, the similar approach was used as in the previous assignment by implementing a service that only checks the produced results. This enabled students to incrementally compute and check different results. Again the students were asked to provide links to all submissions via generic services used to produce all results.

7 Conclusion

The paper presented an experience in incorporating Big Data technologies into introductory PDC courses and building a service-oriented infrastructure to support practical exercises involving these technologies. The presented approach

helped to provide a smooth practical experience for students with different technical background by enabling them to run and test their MapReduce and Spark programs on a provided Hadoop cluster via convenient web interfaces. This approach also enabled automation of routine actions related to submission of programs to a cluster and evaluation of programming assignments.

Future work will focus on improving the presented infrastructure and publishing the service implementations to enable other educators to reproduce the presented approach using the Everest platform.

Acknowledgments

This work is supported by the Russian Science Foundation (project No. 16-11-10352).

References

1. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
2. Tom White. *Hadoop: The definitive guide*. O'Reilly Media, Inc., 2012.
3. Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.
4. Kivanc Dincer and Geoffrey C Fox. Design issues in building web-based parallel programming environments. In *High Performance Distributed Computing, 1997. Proceedings. The Sixth IEEE International Symposium on*, pages 283–292. IEEE, 1997.
5. Juan Touriño, María J Martín, Jacobo Tarrío, and Manuel Arenaz. A grid portal for an undergraduate parallel programming course. *Education, IEEE Transactions on*, 48(3):391–399, 2005.
6. Paolo Maggi and Riccardo Sisto. A grid-powered framework to support courses on distributed programming. *Education, IEEE Transactions on*, 50(1):27–33, 2007.
7. Moritz Schlarb, Christian Hundt, and Bertil Schmidt. Sauce: A web-based automated assessment tool for teaching parallel programming. In *Euro-Par 2015: Parallel Processing Workshops*, pages 54–65. Springer, 2015.
8. Marek Nowicki, Maciej Marchwiany, Maciej Szpindler, and Piotr Bała. On-line service for teaching parallel programming. In *Euro-Par 2015: Parallel Processing Workshops*, pages 78–89. Springer, 2015.
9. Heterogeneous Parallel Programming. [online]. https://www.coursera.org/course/hetero.
10. Victor Gergel and Valentina Kustikova. Internet-oriented educational course introduction to parallel computing: A simple way to start. In *Russian Supercomputing Days*, pages 291–303. Springer, 2016.
11. Patrick Garrity, Timothy Yates, Richard Brown, and Elizabeth Shoop. Webmapreduce: an accessible and adaptable tool for teaching map-reduce computing. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 183–188. ACM, 2011.
12. Hue. [online]. http://gethue.com/.
13. Databricks Platform. [online]. https://databricks.com/product/databricks.

14. Cloudera Data Science Workbench. [online]. https://www.cloudera.com/products/data-science-and-engineering/data-science-workbench.html.
15. O. Sukhoroslov, S. Volkov, and A. Afanasiev. A web-based platform for publication and distributed execution of computing applications. In *Parallel and Distributed Computing (ISPDC), 2015 14th International Symposium on*, pages 175–184, June 2015.
16. Everest. [online]. http://everest.distcomp.org/.