

Using Everest Platform for Teaching Parallel and Distributed Computing

Oleg Sukhoroslov^{1,2}(✉)

¹ Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute), Moscow, Russia
`sukhoroslov@iitp.ru`

² Higher School of Economics, Moscow, Russia

Abstract. The paper presents a practical approach for building high-level services for teaching parallel and distributed computing based on Everest platform. Originally designed for publication of computing applications, the platform is suitable for rapid development of services for running different types of parallel programs on high-performance resources, as well as services for evaluation of practical assignments. As was demonstrated by using Everest for teaching two introductory PDC courses, the proposed approach helps to enhance students' practical experience while avoiding low-level interfaces and providing a level of automation necessary for scaling the course to a large number of students. In contrast to other solutions, the exploited Platform as a Service model provides the ability to quickly reuse this approach by other PDC educators without installation of the platform.

Keywords: Parallel programming · Distributed computing · Web-based interfaces · Web services · Platform as a Service

1 Introduction

The teaching of parallel and distributed computing (PDC) has increasingly gained importance during the last decade due to the ubiquity of multi-core architectures, graphical processors, cloud computing services and the need to process vast amounts of data. Aside from the theoretical foundations, practical programming exercises form an integral part of any PDC course aimed at mastering domain knowledge and developing relevant skills by working with different classes of computational systems and programming technologies.

However, providing a practical experience to the students of PDC course is challenging due to the inherent complexity of involved systems, user interfaces and technologies. A typical example is arranging practical exercises and homework assignments on a compute cluster. A common approach is to provide remote logins for each student, and then train students to use cluster command line environment to compile and submit their programs. This approach suffers from several problems. First, it introduces additional administration, teaching

and support overheads for the course staff. Second, it requires a considerable effort for running programs by students who are often unfamiliar with Unix environment, etc. As a result, the additional time and effort are spent by both instructors and students instead of focusing on the essential parts of the course. Due to the limited human resources the traditional approaches to teaching also do not scale well to a large number of students. Therefore new approaches are needed to make teaching PDC more efficient and reach a wider audience.

Web-based environments provide a convenient alternative for accessing parallel computing systems and supporting practical assignments on such systems. While being successfully used in teaching some PDC courses, the development of such environments represents a substantial cost to many educators. While some of existing solutions can be reused, the additional costs related to deployment, customization and administration of such systems in-house can also be significant. Therefore there is a need in generic platforms that can be reused with a minimal effort, possibly without installation as modern cloud platforms, while being flexible and supporting all common use cases.

In this paper we present an approach to automation of practical programming exercises in PDC based on using Everest platform [1, 9]. While originally designed for building computational web services, the platform proved to be extremely useful for supporting educational activities as well. Everest has a number of unique features in comparison to related solutions. It implements the Platform as a Service (PaaS) model by supporting multiple users and providing all its functionality via remote interfaces. The platform is not tied to a predefined computing infrastructure by enabling users to attach external resources and bind them to applications. This makes it possible to immediately start using Everest without installation.

In order to simplify access to computing resources a number of generic services have been developed on Everest for running various types of concurrent, parallel and distributed programs. Also a number of problem-specific services has been created for each homework assignment in order to automate evaluation and provide immediate feedback to a student. The presented approach has been successfully used since 2014 for teaching two introductory PDC courses.

The paper is organized as follows: Sect. 2 discusses related work and compares the presented approach and available solutions. Section 3 provides technical details on the Everest platform and describes its use for development of services supporting teaching activities on HPC resources. Section 4 presents several use cases from different PDC topics and describes the experience gained from using Everest in two PDC courses. Section 5 concludes and discusses future work.

2 Related Work

The use of web technologies for building convenient interfaces for accessing high-performance resources has been exploited since the emergence of the World Wide Web. In [3] 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. [10] describes an experience of building a grid portal to support an undergraduate parallel programming course.

Web-based interfaces have also been exploited to support submission and automated evaluation of programming assignments in PDC courses. In [5] authors describe a framework enabling implementation of web portals for automated testing of student programming assignments in distributed programming courses. Among the recent works, [8] describes a web-based application for automated assessment and evaluation of source code in the field of parallel programming. In [6] authors present a similar web-based system for running and validating parallel programs written in different programming paradigms.

Finally, a few web-based environments emerged to support recent PDC topics such as big data processing and general-purpose computing on GPU. WebMapReduce [4] is a web interface for Hadoop designed for teaching MapReduce programming. WebGPU is a web-based system developed to support GPU programming assignments in the Heterogeneous Parallel Programming course [2].

In comparison to existing solutions built from scratch for teaching purposes, the presented approach is based on reusing a general-purpose web platform for building computational web services. The ability to quickly build custom services and connect them to computing resources helps to significantly reduce development time. The flexibility of service-oriented approach enables development of different types of services targeting various use cases and application areas. Finally, the exploited PaaS model provides the ability to reuse this approach by other educators without installation of the platform. To our best knowledge, there are no similar attempts were previously made. These features make Everest to stand out in cases where educators need an easy to use yet flexible solution, but lack the resources needed to deploy and maintain such system in-house.

3 Technical Aspects

3.1 Everest Overview

Everest [1,9] is a web-based platform enabling publication, sharing and execution of scientific applications across distributed computing resources. In this section we provide a brief overview of this platform.

Figure 1 shows the high-level architecture and some of the key concepts of Everest. In contrast to traditional distributed computing platforms, Everest implements the Platform as a Service 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. An application added to Everest is automatically published as a web form and a web service. The latter enables programmatic access to applications, integration with third-party tools and composition of applications into

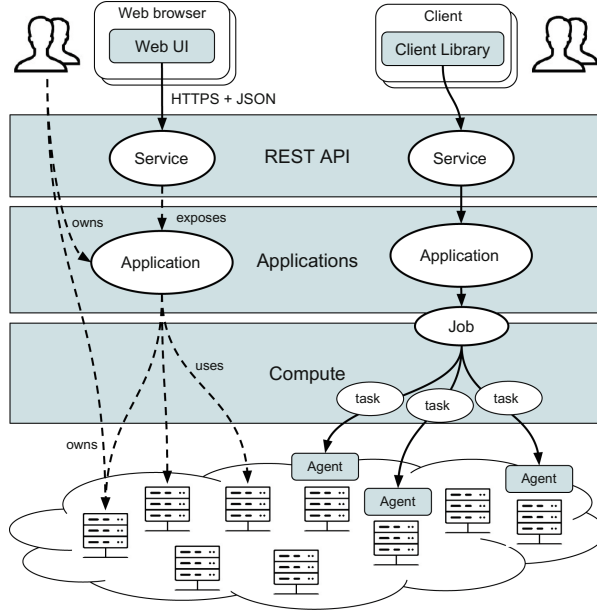


Fig. 1. High-level architecture of Everest

workflows. Another distinct feature of Everest is that it allows users to attach their computing resources and flexibly bind them to applications.

The server-side part of the platform is composed of three main layers: REST API, Applications layer and Compute layer. The client-side part includes the web user interface (Web UI) and client libraries.

REST API implements the remote programming interface providing access to all platform’s capabilities. It serves as a single entry point for all clients, including Web UI and client libraries, and is implemented as a set of web services following the Representational State Transfer (REST) architectural style [7]. The API specification is open and allows implementation of third-party clients.

Applications layer implements a hosting environment for applications created by users. Applications are the core entities in Everest that represent reusable computational units that follow a well-defined 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. Each application is automatically exposed as a web service via the REST API. This enables remote access to the application via Web UI and client libraries.

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.

Compute layer manages execution of applications on computing resources. When an application is invoked via REST API it generates a *job* consisting of one or more computational *tasks*. Compute layer manages execution of these tasks on remote resources and performs all routine actions related to staging of task input files, submitting a task, monitoring a task state and downloading task results.

Everest does not provide a computing infrastructure and instead relies on external resources attached by users. The platform implements integration with standalone machines and clusters by using 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. The platform also supports integration with the European Grid Infrastructure.

Web UI provides a convenient graphical interface for interaction with the platform. It is implemented as a JavaScript application that can run in a modern web browser without installation of additional software on the user's machine.

Client libraries simplify programmatic access to Everest via REST API and enable users to write programs that access applications and compose them in workflows. At the moment, a client library for Python language is implemented.

3.2 Generic Services for Running Parallel Programs

A common challenge for students learning PDC is working on computing resources in order to run their programs on scale. The command line environment and queuing systems used on such resources are unfamiliar and too low-level for many students. Everest can be used to remove these technical barriers by creating web-based services for running parallel programs on a compute cluster. Such services are implemented as Everest applications linked to the provided resource. Since different programming models and technologies use different languages and runtime parameters it is convenient to create multiple generic applications with relevant parameters. In this section we outline steps required in order to create such applications. The complete description of these steps along with technical details can be found in user tutorial on the Everest website [1].

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 program 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.

In order to link the application to a compute cluster used in the course the instructor should attach the cluster to Everest by installing and starting the agent. This step usually does not require much effort since the agent is easy to install under non-root user, provides integration with common batch

systems and does not need inbound connectivity. It is convenient to create a dedicated account on the cluster for running the agent and student submissions from Everest. This approach also avoids creation of personal accounts for each student and associated management overheads.

Once the application is tested and ready to be used by the students, the instructor configures access to the application by specifying users and groups allowed to run it. Everest supports creation of arbitrary user groups. For teaching activities it is convenient to create two groups for students and instructors respectively and configure application to allow submissions from both groups. The students' group can be configured to allow self-registration by providing a secret code to avoid manually adding students to the group. After all is set up it is sufficient to ask students to sign up in Everest, add themselves to the required group and check the applications list.

An example of generic application for running MPI programs created on Everest is presented in Fig. 2. The submit form shown on the figure includes input parameters that should be specified by a student for submitting a job. It is also possible to specify custom job name and enable email notification when the job completes which is convenient for long-running jobs or jobs waiting in a queue. Note that this example allows only a single source file to be submitted by a user, which is often adequate for teaching purposes. However it can be easily modified to support cases that require submitting multiple source files.

Upon the job submission the student is redirected to the job page that displays dynamically updated information about the job state. Figure 3 contains a screenshot of completed job for the MPI application. The opened Outputs section provides access to output parameters produced by the job. The job page also includes sections containing general information about the job and all input parameters specified by the student. By default Everest job is accessible only by its owner. For teaching purposes it is possible 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.

The described approach have been used to implement a number of generic services for running different types of programs described in Sect. 4.

3.3 Problem-Specific Services for Programming Assignments

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

The automated evaluation of assignments requires development of problem-specific services that run the program against the custom test suite. Such services can be implemented on Everest using the same approach as the previously

MPI

[About](#)[Parameters](#)[Submit Job](#)[Discussion](#)

Job Name

MPI

Program

+ Add file...

MPI program as a single *.c or *.cpp file

Arguments

Command line arguments to pass to the program

Files

+ Add item

Additional input files that are used by the program (optional)

Nodes

2

Number of cluster nodes to run the program on (maximum is 12)

Processes per Node

2

Number of MPI processes to run on each node (maximum is 12)

Wall Time

60

Required wall-clock time in seconds (current limit is 3 minutes)

Email Notification

☐

 Send me email when the job completes

Your job will be automatically shared with: @pdc-instructors

Request JSON

▶ Submit

Fig. 2. Submit form of generic application for running MPI programs

MPI Broadcast Example

[Job Info](#)[Inputs](#)[Outputs](#)[Share](#)

Compiler Output	compiler.log
Program Stdout	stdout.txt
Program Stderr	stderr.txt
Wrapper Log	mpirun.log

↺ Resubmit

🗑 Delete

Fig. 3. Results of completed MPI job

discussed generic services. However, in this case the wrapper is replaced by a test suite for the given assignment that can execute a program multiple times with different runtime parameters and performs additional actions such as result validation and performance measurements. The outputs of problem-specific applications can include validation results, performance metrics and scores, etc.

4 Use Cases

The described approach and Everest have been successfully used since 2014 to support two PDC courses for students of various levels.

The Parallel and Distributed Computing course at The Yandex School of Data Analysis (YSDA, Moscow, Russia) is an introductory PDC course for MSc students that features the following topics: concurrency, parallel programming and distributed data processing. The course grade is based on the results of homework assignments implying writing parallel and distributed programs that are executed on a dedicated compute cluster.

The use of Everest in the YSDA course started in 2014 by development of services for evaluation of homework assignments. These services have been improved and are actively used in this course since that time. Until 2016 no generic services for running parallel programs were used, and students had direct access to the YSDA cluster via personal SSH accounts.

In 2015 a similar approach has been applied for teaching High Performance Computing course for BSc students from the Faculty of Computer Science at the Higher School of Economics (HSE, Moscow, Russia). In order to simplify working with the cluster, in addition to problem-specific services, new generic services for running different types of parallel programs have been introduced. Such services also helped to arrange more practical exercises and demos in the class. The students had no problem with accessing Everest via a web browser and were able to quickly learn and successfully use the provided services for submitting their programs both in class and while working on homework assignments.

In 2016 the complete approach using both generic and problem-specific services was used during teaching YSDA course. This time the students were able to perform all practical activities via Everest without directly accessing the cluster. As an option, it was possible to request a cluster account as in previous years to get an additional practical experience. However, only a few students used this option during the course and none of them has completely switched from using Everest to the cluster command line. The decreased support overhead and increased level of automation helped to scale the course to a larger number of students (118 enrolled students in comparison to 80 in 2015 and 48 in 2014).

According to existing experience the development of a service takes from an hour to several days depending on case. The majority of the time is consumed by implementing and debugging a wrapper or a test suite written in Python or Bash. The development of test suites and corresponding services is usually more time consuming than generic services where some previous code can be reused.

In the rest of this section we provide an overview of various services used in the mentioned courses grouped by the core subjects.

4.1 Multi-threaded Programming

Both courses include introduction to concurrency and multi-threaded programming. The C++ programming language and the standard thread support library are used for writing concurrent programs. During this part students examine various pitfalls of concurrent programming, learn how to avoid them and perform coordination between threads.

Since the students usually have no problem with compiling and running multi-threaded programs on their machines, there were no need in development of generic services for running such programs on a cluster. However, the development of problem-specific services for testing homework assignments proved to be extremely useful.

For example, in the Dining Philosophers task the students should solve the well-known problem by meeting the basic safety and liveness guarantees while also ensuring fairness, performance and scalability of their solution. The students are provided with an initial implementation that is not safe and serves as a template for a student's solution. A test suite for evaluation of solutions has been developed that performs checking of all requirements by running a program under various conditions. Besides checking safety and liveness, the test suite also measures fairness, performance and scalability of the solution. The key metrics used are min-max ratio of eat counts and mean wait time in hungry state. The scalability is evaluated by increasing the number of philosophers up to 5000. The test suite prints results of each test and overall summary including scores for all requirements and the total grade.

The developed test suite was provided to the students as a service that takes a program and runs the tests against it on a cluster. To ensure the reliable and reproducible evaluation each test run was configured to use a whole cluster node.

4.2 Parallel Programming

The parallel programming part considers OpenMP and MPI, the two most popular technologies used for shared and distributed memory systems respectively. The generic services are developed for running both kinds of parallel programs on the cluster.

While it is quite easy to compile and run OpenMP programs on students' machines, the OpenMP service provided the students with the ability to run a program on a high-end server with 12 processor cores. The input parameters include the program, command arguments, additional files and number of threads to use, while the output parameters include compiler and program outputs.

The generic MPI service enabled students to compile and run MPI programs with different runtime configurations on the cluster. The interface of this service was already presented in Sect. 3.

The YSDA course includes a programming assignment with two tasks covering both technologies. In the first task the students should implement a parallel version of the K-means method using OpenMP. A sequential implementation of

the method in C++ is provided as a starting point along with a generator of input data. The solution is required to produce the same result as the initial program on the same dataset. The second task considers a parallel implementation of the Game of Life using MPI. Similarly the students are provided with a reference sequential program in C and an input data generator.

A test suite and an Everest application is developed for each task. Both test suites have a similar structure. They compile a solution and perform multiple runs with different runtime configuration (number of threads or processes), input files and other parameters. The execution time, speedup and efficiency are measured for each run. The results of a run are compared with the reference values. This enables a complete evaluation of a solution including its’ correctness, performance, scalability, and dependence on input parameters. Figure 4 contains a screenshot of completed submission for the MPI assignment.

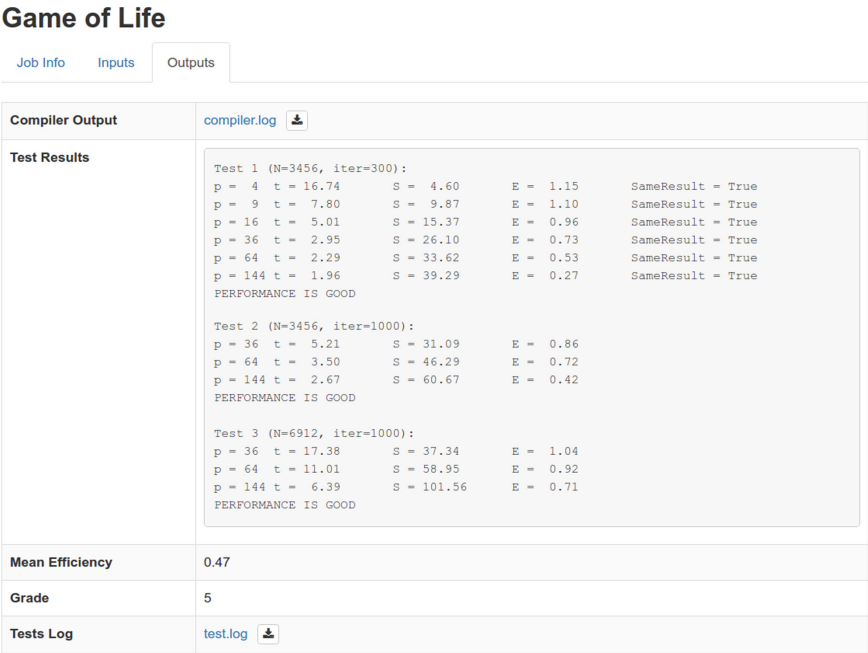


Fig. 4. Results of testing Game of Life assignment

4.3 Distributed Data Processing

This part considers distributed computing models and platforms for processing of large data sets, which are being actively developed during the last decade. Students learn the MapReduce programming model and its implementation in the Apache Hadoop platform. Another popular framework for distributed data processing considered in both courses is Apache Spark.

Two generic services were implemented for running MapReduce programs written in Python and Java respectively on the Hadoop cluster. Both services allow specifying program files, command line arguments, input and output paths in HDFS, 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 Spark programs written in Python or Scala/Java on the same cluster. In comparison to 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.

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. This was achieved by using Hue, a Web interface for Hadoop which includes a convenient HDFS file browser.

The homework assignments include building an inverted index of Wikipedia using MapReduce and analysis of Twitter graph using Spark. The corresponding services were created for each assignment. In contrast to previous assignments, these services do not run students' programs and only check the produced results. Therefore the students were asked to provide links to all submissions via generic services used to produce these results.

5 Conclusion and Future Work

In this paper, we have presented a practical approach for building high-level services for teaching PDC based on Everest platform. Originally designed for publication of computing applications, the platform supports rapid development of various types of computational web services. In particular, as was demonstrated by using Everest for teaching introductory PDC courses, the platform is suitable for building services for running different types of parallel programs on HPC resources, as well as services for evaluation of practical assignments.

The use of discussed services helped to provide easy-to-use interfaces to students and to reduce administration overheads. The problem-specific services ensured reliable and reproducible execution of test suites against students' solutions while providing immediate feedback to students and assisting grading by instructors.

Everest has a number of unique features in comparison to related solutions. It implements the PaaS model by supporting multiple users and providing all

its functionality via remote web and programming interfaces. The latter enable integration of the platform and applications with external systems. Everest is not tied to a predefined computing infrastructure by enabling users to attach arbitrary resources and bind them to applications. This makes it possible to immediately start using Everest without installation. The platform is publicly available online to all interested users [1].

Being a general-purpose platform, Everest lacks a number of high-level features in comparison to specialized solutions. For example, in order to create a service an instructor should write a wrapper script or test suite implementing all necessary actions. While providing maximum flexibility, this approach often requires writing a boilerplate code dealing with cluster job submission or results checking. We plan to address these issues in future by implementing additional features in Everest and publishing ready-to-use blueprints to quickly reproduce the discussed services by other PDC educators.

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

References

1. Everest. <http://everest.distcomp.org/>
2. Heterogeneous Parallel Programming. <https://www.coursera.org/course/hetero>
3. Dincer, K., Fox, G.C.: Design issues in building web-based parallel programming environments. In: 1997 Proceedings of the Sixth IEEE International Symposium on High Performance Distributed Computing, pp. 283–292. IEEE (1997)
4. Garrity, P., Yates, T., Brown, R., Shoop, E.: Webmapreduce: an accessible and adaptable tool for teaching map-reduce computing. In: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, pp. 183–188. ACM (2011)
5. Maggi, P., Sisto, R.: A grid-powered framework to support courses on distributed programming. *IEEE Trans. Educ.* **50**(1), 27–33 (2007)
6. Nowicki, M., Marchwiany, M., Szpindler, M., Bała, P.: On-line service for teaching parallel programming. In: Hunold, S., et al. (eds.) Euro-Par 2015. LNCS, vol. 9523, pp. 78–89. Springer, Cham (2015). doi:[10.1007/978-3-319-27308-2_7](https://doi.org/10.1007/978-3-319-27308-2_7)
7. Richardson, L., Ruby, S.: RESTful web services. O'Reilly Media Inc., Sebastopol (2008)
8. Schlarb, M., Hundt, C., Schmidt, B.: SAUCE: a web-based automated assessment tool for teaching parallel programming. In: Hunold, S., et al. (eds.) Euro-Par 2015. LNCS, vol. 9523, pp. 54–65. Springer, Cham (2015). doi:[10.1007/978-3-319-27308-2_5](https://doi.org/10.1007/978-3-319-27308-2_5)
9. Sukhoroslov, O., Volkov, S., Afanasiev, A.: A web-based platform for publication and distributed execution of computing applications. In: 2015 14th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 175–184, June 2015
10. Touriño, J., Martín, M.J., Tarrío, J., Arenaz, M.: A grid portal for an undergraduate parallel programming course. *IEEE Trans. Educ.* **48**(3), 391–399 (2005)

Euro-Par 2016: Parallel Processing Workshops

Euro-Par 2016 International Workshops, Grenoble,
France, August 24-26, 2016, Revised Selected Papers

Desprez, F.; Dutot, P.-F.; Kaklamanis, C.; Marchal, L.;
Molitorisz, K.; Ricci, L.; Scarano, V.; Vega-Rodriguez,
M.A.; Varbanescu, A.L.; Hunold, S.; Scott, S.L.; Lankes,
S.; Weidendorfer, J. (Eds.)

2017, XXXIX, 829 p. 281 illus., Softcover

ISBN: 978-3-319-58942-8