Converging HPC, Big Data and Cloud Technologies for Precision Agriculture Data Analytics on Supercomputers

Chapter · October 2020

DOI: 10.1007/978-3-030-59851-8_25

13 authors, including:

Li Zhong
Universität Stuttgart
13 PUBLICATIONS **61** CITATIONS

SEE PROFILE

Dennis Hoppe
High-Performance Computing Center Stuttgart (HLRS)
34 PUBLICATIONS **132** CITATIONS

SEE PROFILE

Marcin Pospieszny
Poznańskie Centrum Superkomputerowo-Sieciowe
11 PUBLICATIONS **86** CITATIONS

SEE PROFILE

Nikela Papadopoulou
Chalmers University of Technology
34 PUBLICATIONS **110** CITATIONS

SEE PROFILE

Converging HPC, Big Data and Cloud technologies for precision agriculture data analytics on supercomputers

Yiannis Georgiou[1], Naweiluo Zhou[2], Li Zhong[2], Dennis Hoppe[2] Marcin Pospieszny[3], Nikela Papadopoulou[4], Kostis Nikas[4] Orestis Lagkas Nikolos[4], Pavlos Kranas[5], Sophia Karagiorgou[6], Eric Pascolo[7], Michael Mercier[1], and Pedro Velho[1]

[1] Ryax Technologies name.surname@ryax.tech
[2] HLRS name.surname@hlrs.de
[3] PSNC
[4] ICCS
[5] Leanxscale
[6] Ubitech
[7] CINECA

Abstract. The convergence of HPC and Big Data along with the influence of Cloud are playing an important role in the democratization of HPC. The increasing needs of Data Analytics in computational power has added new fields of interest for the HPC facilities but also new problematics such as interoperability with Cloud and ease of use. Besides the typical HPC applications, these infrastructures are now asked to handle more complex workflows combining Machine Learning, Big Data and HPC. This brings challenges on the resource management, scheduling and environment deployment layers. Hence, enhancements are needed to allow multiple frameworks to be deployed under common system management while providing the right abstraction to facilitate adoption.
This paper presents the architecture adopted for the parallel and distributed execution management software stack of Cybele EU funded project which is put in place on production HPC centers to execute hybrid data analytics workflows in the context of precision agriculture and livestock farming applications. The design is based on: Kubernetes as a higher level orchestrator of Big Data components, hybrid workflows and a common interface to submit HPC or Big Data jobs; Slurm or Torque for HPC resource management; and Singularity containerization platform for the dynamic deployment of the different Data Analytics frameworks on HPC. The paper showcases precision agriculture workflows being executed upon the architecture and provides some initial performance evaluation results and insights for the whole prototype design.

1 Introduction

High Performance Computing has been traditionally used for scientific computing to solve complex problems which require extreme amounts of computation.

HPC is designed with performance as principal focus, leveraging on supercomputers along with parallel and distributed processing techniques. The rise of Big Data came with an increasing adoption of data analytics and Artificial Intelligence in modern applications that make use of data-driven models and analysis engines to facilitate the extraction of valuable insights. Big Data Analytics utilize Cloud data-centers which provide elastic environments based on commodity hardware and adapted software; while instead of performance they focus on flexibility and programming simplicity. Containerization, based on Docker, has greatly improved the productivity and simplicity of Cloud technologies; and together with the advanced orchestration, introduced through systems such as Kubernetes, enabled the adoption of Big Data software by a large community.

Today, Big Data Analytics are becoming more compute-intensive, mainly due to AI and in particular Deep Learning, while needing extremely-fast knowledge extraction for rapid and accurate decisions. The convergence of HPC and Big Data, especially regarding systems software, resource management and programming, is an important concern which appears as top research objective in the Strategic Research Agenda (SRA4) of HPC in Europe as published by ETP4HPC [1].

Big Data analytics are applied extensively, under the digitalization efforts, in various industries such as pharmaceutics, construction, automotive but also agriculture and farming. Supercomputers and HPC can be of great benefit to Big Data Applications since large datasets can be processed in timely manner. But the steep learning curve of HPC systems software and parallel programming techniques along with the rigid environment deployment and resource management remain an important obstacle towards the usage of HPC for Big Data analytics. In addition, the usage of classic Cloud and Big Data tools for containerization and orchestration cannot be applied directly on the HPC systems because of security and performance drawbacks. Hence workflows mixing HPC and Big Data executions cannot be yet combined intelligently using off-the-shelf software.

CYBELE [2] is an EU funded project which aims to provide solutions to the above issues. It brings a prototype architecture combining HPC and Big Data hardware and software tools to enable the deployment of data analytics workflows, in the context of precision agriculture and livestock farming. CYBELE proposes a suite of Cloud-level tools combined with Big Data and HPC systems software and adapted techniques to bring the right abstractions to data scientists with non-HPC systems expertise to optimally leverage HPC platforms. CYBELE disposes four production HPC platforms across Europe upon which a complete set of demonstrators[8] will be rolled-out, covering 9 topics in total: from protein-content prediction in organic soya yields, to climate smart predictive models, to autonomous robotic systems, to crop yield forecasting, down to sustainable livestock production, aquaculture and open sea fishing.

This paper focuses on the systems software layer and in particular on the basic building blocks of the parallel and distributed execution management tools

[8] https://www.cybele-project.eu/demonstrators

used in CYBELE. However, the described tools and techniques can be used in any case where Big Data Analytics need to be executed on HPC platforms. We consider an architecture featuring one Big Data partition composed of VMs managed by Kubernetes, using a mix of Docker and Singularity runtimes for containerization, along with one HPC partition, as the typical HPC production system, composed of baremetal machines managed by Slurm or Torque, using only Singularity containerization. The contributions of this paper are the following:

- Meta-scheduling and resource abstraction techniques enabling first the execution of Big Data Analytics as batch jobs on Slurm or Torque managed HPC partitions, through a Kubernetes micro-service submission based on singularity containers and wlm-operator software adapted for multi-user support; and second the possibility to deploy Big Data Analytics and Cloud-level tools such as workflow managers and databases on the VMs of the Big Data partitions, using the typical Kubernetes API. The deployed Cloud-level tools will provide the needed abstractions to non-HPC experts for the Data Analytics execution on the underlying HPC-Big Data hybrid system.
- An Environment deployment tool for the creation of customizable environments based on singularity containerization and a specialized repository with pre-built images featuring Big Data and AI frameworks (such as Pytorch, Tensorflow and Horovod) for specific HPC resources (such as GPUs and Infiniband).

The reminder of the paper goes as follows: section 2 provides the related work, section 3 describes the Meta-scheduling and resource abstraction techniques, Section 4 presents the Environment Deployment tool, section 5 the validation using precision agriculture data analytics and finally section 6 the Conclusions and Future Works.

2 Related Work

2.1 Resource Management and Orchestration

Older state-of-the-art HPC resource managers such as Slurm[12] and Torque do not provide integrated support for environment provisioning and hence no orchestration[4] is feasible. However, newer resource managers such as Mesos[9] and Kubernetes [10] enable the deployment of containers and allow the applications' lifecycle management. Another widely used orchestrator with limited capabilities but simplicity in usage is Docker Swarm[11]. Kubernetes[5] is the de-facto standard for Cloud and Big Data orchestration, it has a rapidly growing community and ecosystem with plenty of platforms being developed upon it. Kubernetes simplifies the deployment and management of containerized applications. It is based on

[9] https://github.com/apache/mesos
[10] https://github.com/kubernetes/kubernetes
[11] https://github.com/docker/classicswarm

a highly modular architecture which abstracts the underlying infrastructure and allows internal customizations such as deployment of different software defined networking or storage solutions. It supports various Big Data frameworks such as Hadoop MapReduce, Spark and Kafka and has a powerful set of tools to express the application lifecycle considering parameterized redeployment in case of failures, auto-scaling, state management, etc. Furthermore, it provides advanced scheduling capabilities and the possibility to express different schedulers per job.

2.2 Containerization in HPC

Containers have recently started to be applied on HPC clusters. HPC applications are hardware specific, and their applications are often specifically optimized for the nodes. Considering that performance is the focus for HPC applications, it poses the key question for massive usage of containerized applications on HPC cluster [6], [7], [8]. Nevertheless, the flexibility of containerization principles and the productivity advantages makes them very interesting for HPC. Singularity[12] [13] is a technology that bears all the benefits of Bring-Your-Own-Environment, composability and portability, also matching the security requirements in HPC environments. While Docker[11] is the popular approach for containerization in cloud environments, it poses security implications when it comes to HPC centers: Docker allows root user operation, which can lead to privilege escalation. In addition, Docker containers rely on Docker daemon, which requires root access. Rootless mode for the Docker daemon is still experimental. On the other hand, Singularity is a container technology that has been designed for use on HPC systems [14]. Singularity containers do not rely on a daemon for execution and are executed as child processes. Moreover, the user within a Singularity container is the same user as the user of the host system who executes the container, with the same privileges, thus preventing privilege escalation. Regarding the transparent use of resources, Singularity also provides native support for MPI and GPUs. Udocker[13] is another basic user tool (written in Python) to execute simple Docker containers in user space without requiring root privileges.

3 Orchestration and Resources Abstraction in HPC

For the execution of Data Analytics on supercomputers we propose a combination of Big Data, HPC and Cloud tools. The meta-scheduling and orchestration tasks are based upon Kubernetes. The first role of Kubernetes, in that context, is to allow the deployment of either Big Data (ML, DL, etc) or HPC (MPI-based) workloads upon Big Data or HPC platforms through the same command line interface (kubectl) or API (Kubernetes API) making use of a common representation using YAML language. The deployment on the Big Data platforms can be done directly through this API. The deployment on the HPC platforms pass

[12] https://github.com/sylabs/singularity
[13] https://github.com/indigo-dc/udocker

through the integration of a specific existing open-source software named wlm-operator[14] which has been adopted and extended to fit our needs. The software wlm-operator, allows the submission of a job on the dedicated HPC resource manager (Slurm or Torque) by using the Singularity containerization. Kubernetes will also perform resource management and containers orchestration on the Big Data platforms of the supercomputing centers, enabling typical widely used cloud-native software, to be introduced to supercomputers. This is the case of the various Cloud services for which, as we can see on figure 1, the resource abstraction is provided through the help of Kubernetes.

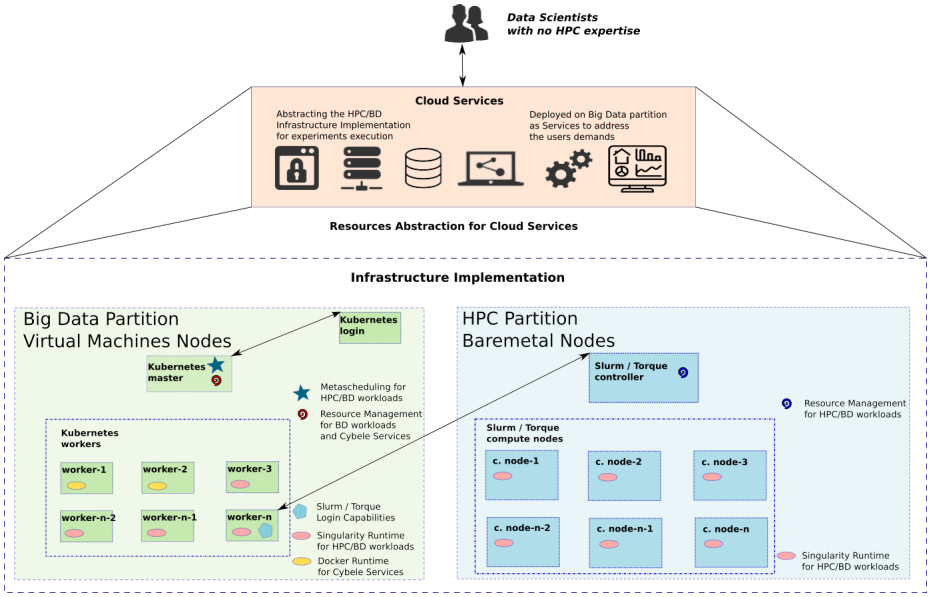

Fig. 1. Architecture for Big Data Analytics on hybrid HPC-Big Data platforms

3.1 Kubernetes and Container Runtimes

Kubernetes controls the deployment lifecycle of containerized applications while managing distributed systems resiliently. Another important part is the resources abstraction through the containerization platform used. As the matter of fact, Kubernetes introduced pods, which specify the resources utilized by a group of application containers. At run time, a pod's containers are instances of container images, packaged and distributed through container image registries. The usage of a containerization platform on Kubernetes goes through the support of specialized Container Runtime Interface (CRI) which has to comply on the Open

[14] https://sylabs.io/guides/cri/1.0/user-guide/k8s.html

Container Interface (OCI) standards. Kubernetes uses by default Docker and its specialized CRI, which is the traditional choice and supported out-of-the box by most cloud-Native software. Docker runtime will be used on Big Data partitions to deploy Cloud services which will then be used to abstract the complexity of deploying experiments on hybrid HPC/BD platforms. For the actual execution of the Big Data Analytics which can take place either on the Big Data or the HPC partition, we adopt Singularity platform. The maintainers of Singularity have proposed Singularity CRI [15] to allow the usage of Singularity for the pods runtime within Kubernetes and the particular mechanism wlm-operator[16] which can enable the direct connection between Kubernetes pods and execution on HPC partition through Singularity containers. .

In the installation and configuration phase of each worker we need to distinguish the nodes that will use Singularity or Docker as runtime. To perform a rightly matched scheduling we need to define specific label per node to show which runtime is used and then on the application submission (yaml of the pod) we need to provide the right node-selector to determine the need in terms of runtime and divert the pod to be scheduled on the right node. As shown in figure 1 the workers with Docker runtime will be used to deploy the Cloud services while those with Singularity will allow the deployment of BD/HPC workloads.

3.2 Integration of Kubernetes and Slurm/Torque with multi-user support

At least one of the Kubernetes workers that will be deployed with Singularity runtime will also need to have Slurm or Torque login nodes capabilities, which will enable the connection with the HPC cluster. This means that from that worker node we should have the capability to run Slurm or Torque commands and in particular job submissions. Based on that we have installed and deployed the wlm-operator software which will open a communication protocol with the Slurm or Torque Resource Manager to submit and monitor containerized HPC jobs through Kubernetes API. The wlm-operator integrates with Slurm by default but in the context of our project and to respect the needs of a particular testbed we have extended it to also support Torque. Furthermore we have enhanced the mechanism of wlm-operator with multi-user support. The prerequisites of wlm-operator software are to have Singularity-CRI runtime on at least the Kubernetes worker node with the Slurm (or Torque) login capabilities and Singularity software installed on all HPC compute nodes to manage containerization on the HPC side.

The wlm-operator software can automatically discover Slurm partition resources (CPUs, memory, nodes, wall-time) and propagates them as node labels to Kubernetes by creating one virtual node (virtual-kubelet) per partition. For this the virtual-kubelet technique is used internally [17]. Similar procedure is followed for Torque queues. Each Slurm partition (or Torque queue) is represented

[15] https://github.com/sylabs/singularity-cri
[16] https://github.com/sylabs/wlm-operator
[17] https://github.com/virtual-kubelet/virtual-kubelet

as a dedicated virtual node in Kubernetes. Those node labels will be respected during Slurm job scheduling so that a job will appear only on a suitable partition with enough resources. The communication protocol between Kubernetes and Slurm (or Torque) is based upon a gRPC proxy, named red-box, which takes place on the worker node that operates the Slurm (or Torque) login binaries. Furthermore, on that Kubernetes worker the user to submit jobs will have to be created on the HPC site and have the rights to submit and monitor jobs.

In order to bridge the communication between Torque and Kubernetes. Torque-Operator extends the wlm-operator with Torque support [3]. Both operators share similar mechanisms, nevertheless, their implementation varies significantly as Torque and Slurm have different structures and parameters. The Torque job script is encapsulated into a Kubernetes yaml job script. The yaml script is submitted from a Kubernetes login. The Torque script part is processed by the Toque-Operator. A dummy pod is generated to transfer the Torque job specification to a scheduling queue e.g. waiting queue, test queue (scheduling queue is a terminology of job scheduler). Torque-Operator invokes the Torque binary qsub which submits Torque job to the Torque cluster. When the Torque job completes, Torque-operator creates a Kubernetes pod which redirects the results to the directory that the user specifies in the yaml file.

By default, with the current version of wlm-operator, all submitted Slurm jobs will be executed on behalf of one user. This is very limiting in our context because we need multi-user support in order to enable individual monitoring, accounting, fairshare scheduling and other features per single user or group of users. For this, we provide a dynamic adaptation of the user context by automatically reconfiguring the virtual-kubelet and agent which is used as a pass-through for the Slurm or Torque job, along with the necessary red-box socket, on the node having the Slurm/Torque login capabilities, using the right user privileges. This gives us the ability to enable each user to use Kubernetes with her account and submit Big Data Analytics on the Slurm or Torque cluster through her account as well, hence removing the initially existing isolation and security barriers of wlm-operator. The mechanisms presented in this paper will be provided as open-source once they are considered more mature.

4 Environment deployment

In our hybrid Big Data-HPC context, the Environment Deployment tool is responsible for setting up the environment for the task execution and deployment. Based on our needs we sought for the following features:

1. the ability to define the environment for any application, without need for access to the underlying system, i.e. the ability to decouple development and deployment,
2. portability across different systems, including HPC and Big Data resource partitions and
3. the ability to transparently use all available system resources, i.e. accelerators, high-performance interconnects, storage.

We based our solution on Singularity containerization and pre-built images for specific AI and Big Data frameworks and HPC resources. For this we offer a repository of singularity images with Pytorch, Tensorflow, Keras, Horovod optimized for specific versions of GPUs, Infiniband and their adapted libraries CUDA, verbs, etc. Different combinations of the above results in a number of pre-defined images that can be used as the base to deploy a specific Big Data Analytics application.

4.1 Singularity Container creation and deployment

A Singularity container image is a single, immutable (read-only), SIF (Singularity Image Format) file. The environment is stored within the image and can include everything from the application code/executable to runtimes to system libraries. Using Singularity, a user can build an image from either a Singularity definition file or by downloading an existing image from a container library, or from Docker hub. In the latter case, the build process transforms the Docker image to a Singularity image. We highlight the following issues and our solutions, related to container creation in the context of its usage in hybrid Big Data - HPC platform:

1. A Singularity image may or may not contain the application itself. In the repository we prepare the images to be generic and we give the ability to the users to either built their application within a new image or just use the base image and deploy their application externally.
2. Although Singularity does not require elevated privileges to deploy an image, it still requires elevated privileges to build an image. Therefore, Singularity image files cannot be built directly on HPC systems, where user access rights are limited. We build all container images on external desktops/servers. The produced image can then be uploaded to the system where it will be deployed, or a shared repository from where it can be used directly by users.
3. To make transparent usage of the network and/or the GPUs, since containers do not virtualize the system, Singularity relies on the host environment as well. For example, for the case of MPI, Singularity partly uses the host runtime to manage MPI processes. Similarly, for the case of GPUs, it uses the host device drivers and related user-space driver libraries. This can cause compatibility issues, if the runtime encapsulated within the container image is not compatible with the runtime on the host. Therefore, even though we externally build Singularity images, we do take into account the related software versions of the underlying HPC systems, to ensure compatibility.

Once a Singularity image is created, either uploaded to the target system or pulled onto the target system from the component library, it can be deployed to execute the corresponding application task. To guarantee an optimal and simplified deployment of singularity images the following parameters need to be configured correctly for the deployment process. These are the bindmounts, the environment variables and the possibility to use instances. Since these are very

closely related to the specific application to be deployed we do not provide any generic solution. Nevertheless some batch scripts examples featuring the usage of singularity deployments and best practices can still be helpful. The pre-built Big Data images along with their definition files will be provided as open-source once they have been sufficiently tested.

5 Multi-GPU scaling of sample precision agriculture application

In order to validate the orchestration and environment deployment layers of our hybrid Big Data/HPC solution we have performed some experiments using one real-life precision agriculture application. The aim of the application is to develop a framework for automatic identification and counting of wheat ears in fields by getting data from sensors on ground that will enable crop yield prediction at early stages and provide more informed decisions for sales planning. The application consists in training a deep learning algorithm written in Python and using Fastai/Pytorch framework based on a group of RGB images (initially 138 images). In particular we deployed the wheat ears counting application upon one single HPC node testing the scaling and parallelization of the code by increasing the number of GPUs.

The experiments have been performed on a dedicated testbed where the previously described architecture of Kubernetes orchestration on both Big Data and HPC partitions has been deployed, along with the integration to Slurm and Singularity for the execution on the HPC partition. The HPC testbed is part of BULL NOVA cluster and we made use of the following hardware:

- one HPC BareMetal node, featuring a Bull Sequana S800 machine, equipped with 4X2 Intel Xeon Platinum 8253 (256CPUs), 4 TB RAM and 4 GPUs NVIDIA GV100GL Tesla V100 PCIe 16GB,
- one Big Data VirtualMachine node, with 4 CPUs and 8GB RAM

The execution took advantage of the singularity pre-built image for fastai/pytorch and was triggered through the orchestration layer of Kubernetes abstraction using Kubectl command line utility. In the case of BareMetal node the execution is finally submitted by Slurm, whereas in the case of VM it is submitted directly by Kubernetes as batch job. The experiment was repeated 5 times for each case and the median value is shown in table 1.

The usage of BareMetal node may be misleading because the execution is not done literally as bare metal. Both cases use containerization with singularity for the executions with the difference that in the second case it is done on VM while in the first on bare-metal. The results in table 1 show the performance improvement of our application when using a powerful HPC BareMetal node with GPU instead of small VM, 100* orders of magnitude. Besides that it shows how the scaling of GPUs impacts the application performance: 10* orders of magnitude when using GPUs rather than only CPU. Our goal is to enable further performance optimizations by providing a distributed version selecting between

VirtualMachine(VM) or BareMetal(BM)	VM	BM	BM	BM	BM	BM
number of CPUs	4	256	256	256	256	256
number of GPUs	0	0	1	2	3	4
Execution Time (sec)	37008	7020	417	312	274	247

Table 1. Execution time of wheat-ears application on 1 VirtualMachine (4 CPUs) node or one BareMetal (256 CPUs) node scaling from 0 to 4 GPUs

pytorch.distributed or horovod. Furthermore and most importantly, these simple executions enabled us to validate the usability of our integrations combining Kubernetes, Slurm and Singularity in a hybrid Big Data/HPC environment for execution of Deep Learning training models.

6 Conclusions

This paper presents a prototype architecture to enable the execution of Big Data Analytics upon supercomputers using different Big Data and HPC hardware partitions and a converged Big Data-HPC-Cloud software stack. We proposed mechanisms that make use of Kubernetes as high-level orchestrator and common API to allow the deployment of Data Analytics as HPC jobs, through an integration with Slurm based on an a multi-user version of wlm-operator and Singularity containerization. Furthermore we proposed an environment deployment tool bringing pre-built images of Big Data and AI frameworks (Pytorch, Tensorflow, etc) specifically adapted to targeted HPC resources (GPUS, Infiniband, etc) which can be used as base to further built environments to be used for different types of Data Analytics on HPC.

These mechanisms are aimed to be used as the basic building blocks to provide supercomputers abstraction targeting data-scientists with no-HPC expertise. For this we aim to deploy Cloud-level software such as the Spring Cloud Dataflow workflow manager[18] and the LeanXscale database[19]. These tools can be used to create hybrid Big Data-HPC workflows, with the necessary data management, to be deployed transparently, for a data-scientist, through the common Kubernetes API and the aforementioned techniques to the underlying supercomputer. Further optimizations will be researched for the multi-user support of the Kubernetes integration with Slurm and for this we will explore the possibility to use the newly introduced Slurm REST-API[20] which will allow a more direct communication with Cloud services. The support of specialized Big Data frameworks such as Spark and Flink need the usage of a resource manager. This role can be played by Kubernetes [10] and this is another direction that we are exploring. In this context we are studying ways to allow the collocation of Big

[18] https://spring.io/projects/spring-cloud-dataflow
[19] https://www.leanxcale.com/
[20] https://slurm.schedmd.com/rest.html

Data and HPC jobs by making use of Kubernetes to deploy Spark applications on Slurm clusters through a non-interfering method of low-priority jobs[9].

7 Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement NO.825355.

References

1. ETP4HPC, "Strategic research agenda (SRA4) for HPC in Europe," March 2020. [Online].Available: https://www.etp4hpc.eu/pujades/files/ETP4HPC_SRA4_2020_web(1).pdf
2. Konstantinos Perakis, Fenareti Lampathaki, Konstantinos Nikas, Yiannis Georgiou, Oskar Marko, Jarissa Maselyne, CYBELE – Fostering precision agriculture & livestock farming through secure access to large-scale HPC enabled virtual industrial experimentation environments fostering scalable big data analytics, Computer Networks, Volume 168, 2020, 107035, ISSN 1389-1286
3. Naweiluo Zhou, Yiannis Georgiou, Li Zhong, Huan Zhou and Marcin Pospieszny. Container Orchestration on HPC Systems. To appear in IEEE CLOUD 2020
4. E. Casalicchio, Container Orchestration: A Survey, pp.221 235.Cham: Springer International Publishing, 2019
5. K. Hightower, B. Burns, and J. Beda, Kubernetes: Up and Running Dive into the Future of Infrastructure, OReilly Media,1sted., 2017
6. M. G. Xavier, M. V. Neves, F. D. Rossi, T. C Ferreto, T. Lange, and C. A. F. De Rose, Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments, 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 233–240, 2013
7. M. Plauth, L. Feinbube, and A. Polze, "A Performance Survey of Lightweight Virtualization Techniques," in Service-Oriented and Cloud Computing F. De Paoli, S. Schulte, and E. Broch Johnsen, pp. 34–48, Springer International Publishing, 2017
8. J. Zhang, X. Lu, and D. K. Panda, "Is Singularity-Based Container Technology Ready for Running MPI Applications on HPC Clouds?, in Proceedings of The10th International Conference on Utility and Cloud Computing, Association for Computing Machinery, 2017
9. Michael Mercier, David Glesser, Yiannis Georgiou, Olivier Richard: Big data and HPC collocation: Using HPC idle resources for Big Data analytics. BigData 2017: 347-352
10. Spark - Kubernetes integration: https://spark.apache.org/docs/latest/running-on-kubernetes.html
11. C. Boettiger, "An introduction to Docker for reproducible research," in ACM SIGOPS Operating Systems Review, 2015.
12. Andy B. Yoo, Morris A. Jette, Mark Grondona: SLURM: Simple Linux Utility for Resource Management. JSSPP 2003: 44-60
13. David Godlove: Singularity: Simple, secure containers for compute-driven workloads. PEARC 2019: 24:1-24:4
14. Giuseppa Muscianisi, Giuseppe Fiameni, Abdulrahman Azab: Singularity GPU Containers Execution on HPC Cluster. ISC Workshops 2019: 61-68