Dynamic Job Allocation on Federated Cloud-HPC Environments

Dynamic job allocation on federated Cloud-HPC environments

Giacomo Vitali, Alberto Scionti, Paolo Viviani, Chiara Vercellino, Olivier Terzo

Abstract With the growing complexity of workflows brought by the recent integration of machine learning, deep learning and big data analytics techniques, there is an ever increasing demand for compute, network and storage resources which require innovative approaches to their management, as well as their easy access and use (including the cloud model). Although there is an abundance of resources in today's HPC infrastructures, they remained shared across the users, and for certain use cases (*e.g.*, urgent computing applications) they may be still not enough to fulfil the workflow requirements. Also, specific computing resources (*e.g.*, hardware accelerators) may be accessible only within certain datacentres. To cope with these challenges, a secure interconnection among multiple HPC datacentres that allows mutual access to their resources (federation) is considered. This paper focuses on the extension of the *SimGrid* software library, a C++ based simulation framework, for evaluating the jobs allocation strategies that lay at the core of a federated execution platform. A greedy-based allocation strategy has been evaluated against random and round-robin approaches; then, this greedy allocation strategy has been integrated within the main orchestration service developed in the context of the LEXIS federated execution platform. Tests with real workflows showed the capability of this greedy allocation strategy to dynamically select the best suitable execution cluster for different jobs.

Giacomo Vitali
LINKS Foundation, Torino e-mail: giacomo.vitali@linksfoundation.com

Alberto Scionti
LINKS Foundation, Torino e-mail: alberto.scionti@linksfoundation.com

Paolo Viviani
LINKS Foundation, Torino e-mail: paolo.viviani@linksfoundation.com

Chiara Vercellino
LINKS Foundation, Torino e-mail: chiara.vercellino@linksfoundation.com

Olivier Terzo
LINKS Foundation, Torino e-mail: olivier.terzo@linksfoundation.com

1 Introduction

The ever-growing of workflows complexity is putting more pressure on HPC infrastructures in providing adequate resources for executing different jobs demanding very diverse computing, storage and networking resources. To address this challenge, the federation of HPC infrastructures and combination of cloud services provisioning has been proposed as an effective way of providing such resources, as well as to support applications with peculiar requirements such as the cases found in the urgent computing domain. In this regard, the orchestration of complex scientific and industrial workflows on distributed resources presents a large number of challenges, such as the management of data dependencies, automatic job scheduling, results retrieval and visualization among others. Moreover, moving towards such federated execution environments, composed of HPC datacentres and cloud services, adds another layer of complexity to it, as data and jobs have to be orchestrated between different and often distant locations.

In this context, the LEXIS (Large-scale EXecution for Industry & Society) project proposed a solution [1] that leverages the flexibility of the Yorc orchestrator[1] supported by the common Distributed Data Infrastructure (DDI)[2], which provides data availability on multiple centres, and a newly developed module –named Dynamic Allocation Module (DAM), which is written in Python language. The latter allows the orchestrator to dynamically allocate jobs at any available location and at any given moment, by retrieving all the necessary information for each federated HPC datacentre before the actual job submission. The gathered information involves mainly the HPC and cloud resource status; gathering operation is accomplished by means of REST APIs requests to the HEAppE middleware [2][3] and OpenStack respectively, while programmed maintenance and networking performances are periodically updated through dedicated REST APIs and stored in a database. Then, the preferred location is chosen using a greedy strategy, which, for each location, computes the weighted mean of the scores associated with the gathered information.

To validate this type of approach, a C++ based HPC-federation simulator has been developed, leveraging the *SimGrid*[3] software library. The simulator allows to define a number of workflows composed of any number of jobs, as well as the resources each centre dispose of, such as clusters and cloud partitions. Moreover, it can simulate data transmission, as well as queue-like behaviour and machines occupancy, by using a parametrized Monte Carlo approach. The preliminary results of this simulator demonstrate that the implemented greedy strategy performs better in terms of overall time-to-solution and job distribution than random and round-robin approaches, which are typically used by meta-orchestrators to fairly distribute the workload on available resources. This activity resulted in the implementation of the DAM[4] and its full integration within the LEXIS orchestration architecture.

[1] https://github.com/ystia

[2] https://heappe.eu/web/

[3] https://simgrid.org

[4] https://github.com/lexis-project/orch-service-dynamic-allocator-module

2 Related Work

Dynamic resource allocation is a research topic well studied in the literature, with approaches based on well defined optimality criteria being proposed both in the cloud and Grid computing domains. Similarly, simulation frameworks have been proposed to model large computing infrastructures with different degrees of detail.

The resource allocation task resembles a scheduling problem and, thus, can be tackled in many different ways; among the others, the mapping of the resource allocation problem as a combinatorial optimization one has been well studied. For instance the (online) bin packing problem [4] is well-known to be NP-hard to solve; thus, effective solvers use heuristics to approximate the optimal solution within an acceptable time window. To mention a few, in [5], the authors proposed to hybridize the cuckoo search heuristic with a gradient descent technique, to speed up the algorithm convergence towards the overall workload execution time reduction, while in [6], particle swarm optimization (PSO) based heuristic was used to optimally schedule the predicted workloads. An overview of various approaches for allocating cloud resources have been surveyed in [7], which include bio-inspired approaches. A scheduling approach oriented to cover specific requirements (*e.g.*, urgent computing) that are common in the weather forecast workflows is reported in [9]. Finding a suitable allocation of the computing resources has been studied to contribute in reducing the energy consumed by the infrastructures. As such, the energy minimization has been tackled by solving an associated mixed integer linear programming (MILP) problem [8]. Also, a deep learning (DL) approach has been used by Google[5] to reduce the power usage effectiveness (PUE) of their datacentres by 15%.

Unlike the cloud, in Grid computing the assignment of resources is done through a "pull" mechanism, *i.e.*, each resource signals its availability to the orchestrator and requests a new task/job to run. Examples of such pulling approaches can be found in the SETI@Home and BOINC [10] world-scale projects, as well as in the DIRAC management system [11], which has been developed in the context of the LHCb experiment at CERN. With respect to the cloud and Grid computing models, the LEXIS platform lies in between, (HPC/cloud partitions on the clusters) served through a queuing system (similarly to the case of Grid's nodes).

Simulation frameworks have been used in all the scientific and technical domains to evaluate the behavior of a system without the need of performing complex operations on the real target system. As such, simulation frameworks provide means for assessing the behavior of a given resource allocation strategy and to compare it against the others. Several frameworks have been proposed to quickly and accurately simulate distributed computing infrastructures. Worth to mention are WRENCH system [12], which is based on the SimGrid library, and GSSIM [13]. Mansouri et al. [15] surveyed other tools developed for research in the cloud domain. Compared to these works, we provided extensions to the *SimGrid* library for modelling and simulating heterogeneous clusters exposing different types of computing resources.

[5] https://www.deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-by-40

3 Simulator Overview

The multi-centre federated context for which the LEXIS platform has been developed does not allow for the straightforward creation of a separate test environment where the functionality of every single module, as well as their interactions, can be properly and safely assessed before moving to production. Therefore, in order to minimize any possible detrimental effects on the overall system, the development phase must follow a strict procedure. Moreover, in the specific case of the creation of a module for the dynamic allocation of workflow jobs, assessing the functionality of the envisioned allocation algorithm was a sensible prerequisite. These considerations led to the idea of developing a software library for the simulation of a federated HPC-centre environment where we could safely make initial tests on the module's behaviour and evaluate the effect on the system.

It is important to note that, while it is very challenging to precisely model a "digital twin" of the whole LEXIS federation, its simulation still remains an interesting and useful topic to investigate, as it allows, among other things, to perform a rough performance evaluation of the allocation strategy before even starting the development of the real module.

In the following sections, we describe in detail the simulator's architecture and design, as well as report the results obtained from the performed tests.

3.1 Simulator Architecture and Design

To create a simulation library for a complex multi-centre federated environment, composed of both HPC and cloud partitions, connection between datacentres and clusters, as well as supporting the execution of heterogeneous multi-job workflows, we performed a survey of the existing frameworks and we decided to use the *SimGrid*[6] as the main engine of the simulator. In fact, *SimGrid* is a C/C++ based framework for distributed computer systems' simulation, which has been used by researchers in many fields of application to prototype, evaluate and compare relevant platform configurations, system designs, and algorithmic approaches. It provides models and API to simulate a wide range of distributed computing platforms (such as commodity clusters, wide-area and local-area networks, peers over DSL connections, datacentres, etc.), which we used as fundamental bricks to build our specific HPC-federation simulator.

Following the *S4U* (*SimGrid for you*) interface's formalism, we created a set of **actors**, which represent the independent streams of execution in the application, *i.e.*, the *orchestrator* (master actor), the HPC resources composed of the *receiver* and the *executor* actors, and the *cloud* actor. Each actor associates a function describing the task the specific actor has to execute, with a **host**, that represents the "physical" HPC/cloud resources on which the actor's function is performed. A set

[6] https://simgrid.org/

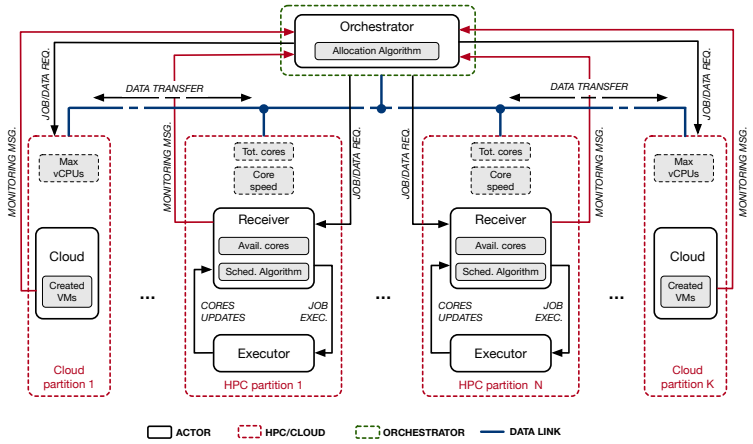

Fig. 1: Simulator architecture with extension provided to properly represent the HPC/cloud federation.

of parameters can be set for the orchestrator actor (*i.e.*, the name of the allocation strategy to be used, the number of workflows to be executed, etc.) and the hosts (*i.e.*, the max core Flop/s, the total number of cores, network performance, presence of specific hardware, such as Burst Buffers, GPU, etc.) in two dedicated configuration XML files. The modular approach adopted by the developed solution allows for the creation of specific classes for any allocation algorithm to be tested, which then can be easily selected for a simulation run through the specific configuration XML file. Figure 1 shows the described simulator architecture, detailing the interactions between *hosts* (also named as partitions in the figure) and *actors*.

The workflows to be simulated have to be described in text files following a format specifically created for this simulator. This format allows for the definition of HPC and cloud jobs' parameters, such as Flop/s and number of cores required, and the size of input data, as well as the dependencies between them. These templates are then used by the simulator to launch workflow execution instances. During the simulation run, the status of the resources is emulated using a Monte Carlo approach. In detail, queue waiting time and CPU occupancy follow a normal distribution, while their update is triggered following a Poisson distribution. The parameters, such as mean, standard deviation, update frequency, are defined in the configuration files mentioned above. Moreover, the distribution types can be customized by users, but contrary to the allocation algorithm's case the change should be done in the code (this could be improved in a future version of the simulator). One important remark is that, while computing the Flop/s needed by a specific job (which approximates the average amount of operations performed in the time unit) is unrealistic in real cases, a reasonable value can be computed using the computational time for the job in a real machine whose technical specifications are known.

3.2 Simulator Assessment

The simulator has been used to benchmark the performance of a greedy allocation strategy (developed in the context of LEXIS project) with respect to simple allocation policies, such as purely *random* allocation and *round robin*. The algorithm computes the weighted mean value $s_{mean,j}$ of a set of scores s_{ij} assigned for specific metrics i for each available location j, then chooses the one with the highest mean score:

$$s_{mean,j} = \frac{\sum_i w_i s_{ij}}{\sum_i w_i},$$

where w_i is the weight assigned to the i-th score. The values of these scores range from 0 to 1, where the former indicates a non compatible platform, and the latter the best one.

The rationale behind this approach is to try to find the best compromise between the various metrics, excluding the incompatible locations ($s_{ij} \leq 0$). This simple approach is easily extensible in the number of metrics considered and provides enough flexibility in terms of expected behaviour through the customization of the scores' weights. In the implemented code, 4 metrics and the relative scores have been taken into account for each location j:

$$s_{1,j} = \frac{1}{1 + \frac{c_{used,j}}{c_{tot,j}}}; \qquad s_{2,j} = \frac{Flops_{core,j}}{Flops_{max}}; \qquad s_{3,j} = \frac{t_{sim,j}}{t_{sim,j} + t_{queue,j}}; \qquad s_{4,j} = e^{-x_j},$$

where $c_{used,j}$ is the number of used cores at the allocation time, $c_{tot,j}$ is the total number of cores, $Flops_{core,j}$ is the Flop/s per core, $Flops_{max}$ is the max Flop7s per core between all the available resources of the same type (HPC or cloud), $t_{sim,j}$ is the estimated simulation time, $t_{queue,j}$ is the estimated queue waiting time (only for HPC resources) and x_j is a discrete variable whose value is 0 if the job's input data are in the same location j, 1 if they are in the same datacentre and 2 otherwise; the rationale being data-transfer over long distances should be penalized. To evaluate the functionality of the envisioned *greedy* strategy, two simulated experiments have been performed using a common system platform which is shown in Figure 2.

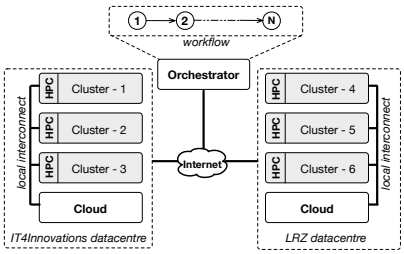

Fig. 2: The simulated LEXIS federated platform.

	Alloc. Strategy	μ $[10^3 s]$	σ $[10^3 s]$
Setup 1	*greedy*	93.602	0.049
	random	196.496	91.354
	round robin	192.351	96.866
Setup 2	*greedy*	134.294	2.472
	random	248.927	85.807
	round robin	246.124	84.648

Table 1: Timing results for the experiment with setup 1 and 2.

Table 1 shows the results obtained in terms of simulated workflow execution time during the two experiments with different setups. They consisted of the execution of 100 identical single-branch DAG workflows, each composed of 3 jobs with linear dependency to emulate pre-processing, simulation and post-processing. The two setups differ in the clusters' occupancy parameters and in the weights' setting. For the first setup, the occupancy has been set at around half capacity with a small variance ($\mu \simeq 50\%$ and $\sigma \simeq 5\%$) for all the clusters and $w_i = 1$. In the second case, instead, the occupancy of the most frequently chosen location during the first experiment, and the weight assigned to the score related to the CPU core performance has been lowered from 1 to 0.25 as the difference between each cluster was impacting too much the choice of the allocation in the first experiment. As expected, the greedy strategy performed better than *random* and *round-robin* in both cases, as the workflows are statistically executed in a shorter time and with a much smaller variance. These simple tests demonstrate the functionality of this approach in a simulated and, therefore, controlled environment, which of course may differ from the complex and unpredictable real case scenarios, but provides nonetheless a first indication of the expected behaviour. In the next section, we describe the developed solution for the LEXIS orchestrator and its implementation in the real platform.

4 Service Implementation

Figure 3 depicts the main structure and the components involved in the orchestration process, as designed in the context of the LEXIS platform. The services that lay behind the LEXIS platform allow the provision of computing and storage resources by federating many datacentres into a single managed platform. For this purpose, each datacentre may expose the capabilities of different machines in terms of computing power and storage space; as such, the access to the resources is provided via cloud or HPC model. To make the federation work, a network connection among the datacentres exists. Furthermore, the access to computing, cloud and storage resources is abstracted through two specific software components: (*i*) the HEAppE middleware instances govern the actual job submission process to the HPC resources; (*ii*) the Distributed Data Infrastructure (DDI) component abstracts and eases the access to the datasets used by running workflows. The access to OpenStack cloud resources, instead, is provided through YORC plugin using HEAppE authentication support.

Workflows are formally described by the composition of TOSCA[7] software abstractions which represent either infrastructural elements (*e.g.*, a compute node with its CPU(s), memory and storage) or software elements (*e.g.*, the instantiation of a docker engine or a specific software). Alien4Cloud provides the mechanisms driving the user with the workflow description process; workflows and their building blocks components are stored in a catalogue which has been made accessible through the LEXIS portal (not represented in Figure 3). The actual workflow execution engine is

[7] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

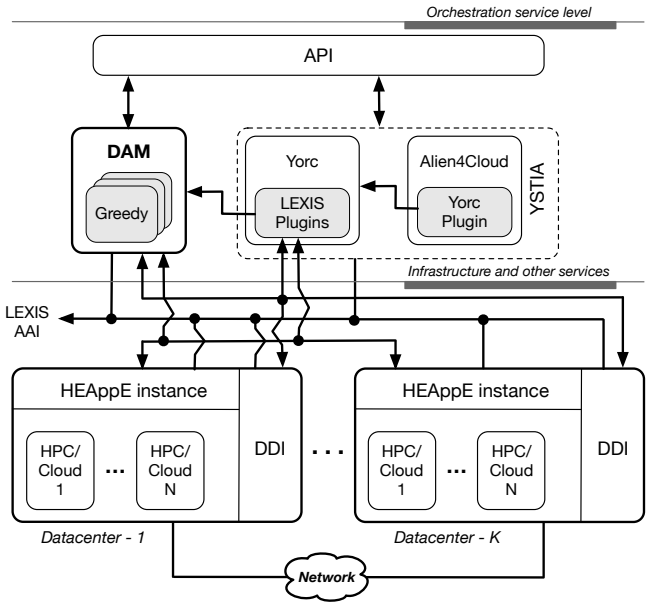

Fig. 3: Architecture of the LEXIS federated orchestration: Ystia toolbox is augmented with the DAM for dynamically selecting the jobs' execution locations.

Yorc orchestrator; it has been extended through specific plugins to interact with the HEAppE middleware instances and DDI, as well as with the Dynamic Allocation Module (DAM).

DAM allows Yorc engine to rely on the delegation model for selecting the proper HPC/cloud set of resources (location) to execute a given job (*i.e.*, by fulfilling the job requirements and all the infrastructural constraints). As such, the DAM has been implemented as a flexible and extensible service, whose purpose is to compute the most suitable location. To this end, like the case of the simulator (see Section 3), the DAM comes with different allocation strategies, which allow selecting time by time the most suitable location. Among the others, the *greedy* allocation strategy, as described in Section 3, allows combining different scores related to the amount of resources already in use at a given location, the total available resources, the availability of the input dataset and the data transfer speed between locations. All the scores are computed by dynamically querying the HEAppE middleware instances (one per datacentre) and the DDI subsystem. The averaged score can be eventually masked out if the specific location is planned to go into maintenance during the scheduled job execution time-frame. Averaged scores for the various locations are ranked and the one receiving the highest (averaged) score is selected and sent back to Yorc orchestrator to submit the job(s). Worth mentioning is that, through the DAM, the entire workflow execution becomes dynamic, *i.e.*, the allocation decisions are made during the workflow execution; this implies that different locations can be se-

lected between two different runs of the same workflow, and for each job contained in it. Such dynamicity allows the LEXIS platform to better exploit all the federated resources and to fulfil hard requirements such as in the case of urgent computing applications (where multiple locations can be selected to ensure the completion of the jobs within a given time window) or to overcome failures and other outage events.

The DAM service module has been implemented using Flask, a Python-based framework for developing web-based services, with less than 2000 lines of code and exposing a REST API. The service was supplemented by a time-series database (InfluxDB) to asynchronously collect and store values used to compute the scores, or the evaluated scores themselves. All the LEXIS orchestration service components have been designed to authenticate each time against the LEXIS Authentication and Authorization Infrastructure (AAI), thus following the zero-trust security principle. Also, these components have been mostly deployed on virtual machines which are able to connect to all the federated resources.

5 Experimental Tests

The LEXIS orchestration service along with the DAM has been tested in the field, by running application workflows defined by the LEXIS project's pilots and covering Aeronautics, Earthquake and Tsunami (also covering urgent computing use cases), and Weather and Climate application domains. These three pilots defined workflows involving the massive use of HPC resources, as well as the access to cloud virtual machines and datasets distributed across the federated datacentres.

To this purpose, the federation comprised the access to 3 supercomputers located in the Czech Republic –IT4Innovation HPC centre (*i.e.*, Salomon cluster – 2.0 PFlop/s, Barbora cluster –849.0 TFlop/s, and Karolina cluster –15.7 PFlop/s), and 3 supercomputers located in Germany –LRZ HPC centre (*i.e.*, SuperMUC (decommissioned) –3.0 PFlop/s, SuperMUC-NG –26.9 PFlop/s, and a generic Linux Cluster). Additionally, the access to cloud partitions and to smart buffer nodes for accelerating I/O intensive operations was provided by IT4Innovations and LRZ centres. A dedicated VPN connection between the two centers was set up, allowing a seamless access to the federated resources.

The DAM was tested in the context of the aforementioned application domains, by running different workflows and letting the module dynamically selecting the most suitable locations for running different jobs. To this end, and to demonstrate the limited amount of resources required by the DAM, a small virtual machine (2 vCPUs and 8GB of RAM) was used for the deployment. All the tests have been performed using the previously discussed greedy allocation strategy. Also, failure conditions have been extensively tested. As proof of the performed tests, we describe here those obtained in the context of the Weather and Climate domain.

LEXIS pilot covering this application area resulted in three workflows, all with a similar structure; they are namely: (*i*) RISICO –risks of wildland fires simulations over Italy [16]; (*ii*) Continuum –risks of flooding simulation over Italy [17]; and

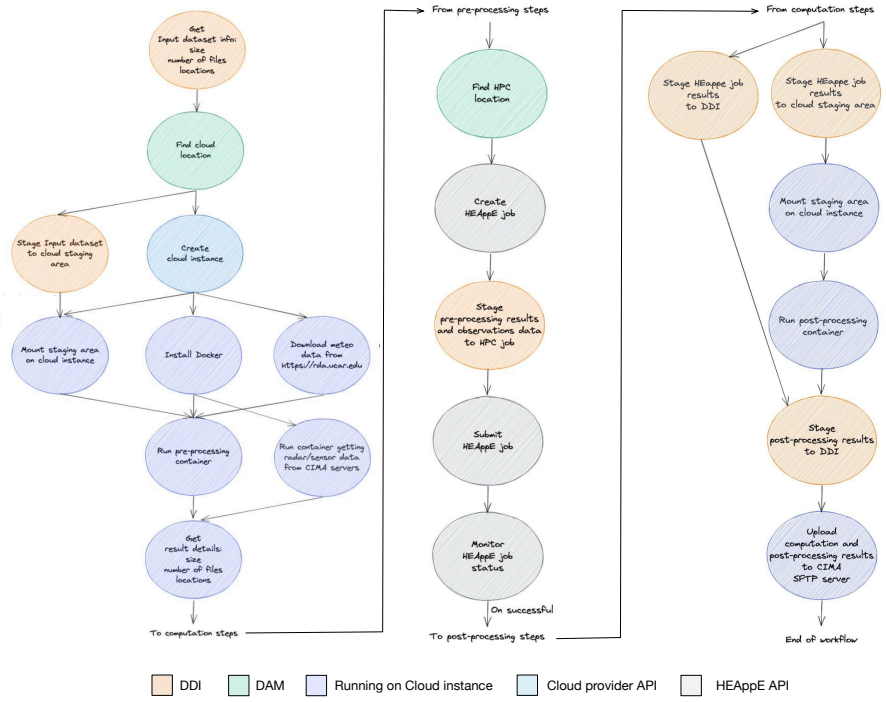

Fig. 4: Example of the Weather and Climate workflows' structure: pre-processing phase (left), heavy computing phase (center), post-processing phase (right).

```
Environment][Run][343cddd8-b369-4d1b-8b4a-3915b3812812][97f51658-775d-47f5-b21d-
e4a97cb25ccf][FindHPCLocationsJob][][tosca.interfaces.node.lifecycle.runnable][submit]
[]FindHPCLocationsJob submitting request to compute best location by
[2022-02-09T19:42:07.846470283+01:00][INFO]
[yorc-16d22cbf8f2d3ae813c66771ab0c9334_164442613279757809722469206131422763 07-
Environment][Run][343cddd8-b369-4d1b-8b4a-3915b3812812][97f51658-775d-47f5-b21d-
e4a97cb25ccf][FindHPCLocationsJob][][tosca.interfaces.node.lifecycle.runnable][submit][]Component
FindHPCLocationsJob submitting to Dynamic Allocator Module HPC placement request
{"number":1,"project":"5f248333-a8d7-62af-4484-
ef763ff2b331","max_walltime":27000,"max_cores":512,"taskName":"WRF Generic","storage_inputs":
[{"size":3756872,"locations":["lrz_iRODS"],"numberOfFiles":13}],"attempt":0,"original_request_id":""}
[2022-02-09T19:42:59.12451051+01:00][INFO]
[yorc-16d22cbf8f2d3ae813c66771ab0c9334_164442613279757809722469206131422763 07-
Environment][][][][FindHPCLocationsJob][0][][][]Attribute "request_id" value for node
"FindHPCLocationsJob", instance "0" changed to "50fb8eee-1c55-46c1-af7d-637c6b8cbf0a"
[2022-02-09T19:42:59.17961246+01:00][INFO]
[yorc-16d22cbf8f2d3ae813c66771ab0c9334_164442613279757809722469206131422763 07-
Environment][][][][FindHPCLocationsJob][0][][][]Attribute "request_type" value for node
"FindHPCLocationsJob", instance "0" changed to "hpc"
[2022-02-09T19:42:59.213413479+01:00][INFO]
[yorc-16d22cbf8f2d3ae813c66771ab0c9334_164442613279757809722469206131422763 07-
Environment][Run][343cddd8-b369-4d1b-8b4a-3915b3812812][97f51658-775d-47f5-b21d-
e4a97cb25ccf][FindHPCLocationsJob][][tosca.interfaces.node.lifecycle.runnable][submit][]Status for
workflow step "FindHPCLocationsJob_submit" changed to "DONE"
[2022-02-09T19:42:59.268024292+01:00][INFO]
[yorc-16d22cbf8f2d3ae813c66771ab0c9334_164442613279757809722469206131422763 07-
```

Fig. 5: Example of log captured during the execution of the Continuum workflow. The dynamic request to the DAM and its status change are highlighted in yellow.

(*iii*) ADMS –Air quality over France. Figure 4 shows the common structure of these workflows, highlighting how different steps involve the interaction with the components of the LEXIS orchestration service, including the dynamic location selection. All these workflows are characterized by three main phases, *i.e.*, the initial preprocessing of the input datasets, the heavy computational phase (where large simulations are involved, using the Weather Research and Forecasting Model –WRF), and the final step where post-processing of the simulation outputs is performed. These phases demanded accessing cloud resources (an instance with at least 10 vCPUs, 45 GB of memory and 150 GB of disk space), and the HPC clusters by specifying the number of cores and max wall time required. Figure 5 shows an example of the tracked LEXIS orchestration service logs related to the run of the Continuum workflow. Highlighted in yellow, there is the dynamic request done by Yorc orchestrator to the DAM (by defining the max wall time and cores as parameters) along with the detection of the request status change to "DONE", meaning that the DAM completed the location selection process. Indeed, the DAM REST API has been designed to work asynchronously, thus implying that Yorc orchestrator has to check whenever the request processing has been completed.

6 Conclusion

The ever-growing complexity of application workflows often demands accessing computing and storage resources that are not available at the single site (HPC datacentre), also by different access models (cloud vs. HPC batch schedulers). In this context, the HPC datacentre federation provides a practical approach to overcoming the limitations of the single HPC datacentre. The LEXIS project is building an advanced engineering platform at the confluence of HPC, cloud and Big Data, which leverages large-scale geographically-distributed resources from existing HPC infrastructure, employs Big Data analytics solutions, and augments them with cloud services; as such, the problem of dynamically selecting the most suitable compute structure (location) arose. To tackle this challenge, in this paper we described the design principle and implementation of a simulation framework we used to assess the behavior of a *greedy* allocation strategy in a real context. Given the positive feedback of the simulator, we implemented such a strategy as the main one for dynamically deciding the computing location to use for the job submission. To this end, we developed the DAM and integrated it within the LEXIS orchestration service. Finally, we provided a proof of the capability of the DAM in a real execution environment, through the LEXIS pilot workflows.

Acknowledgements

This work by the LEXIS project funded by the EU's Horizon 2020 research and innovation programme (2014-2020) under grant agreement No.825532.

References

1. Golasowski, M., et al. "The LEXIS Platform for Distributed Workflow Execution and Data Management." HPC, Big Data, and AI Convergence Towards Exascale. Taylor & Francis, 2022.
2. Golasowski, M., et al. "Data System and Data Management in a Federation of HPC/Cloud Centers." HPC, Big Data, and AI Convergence Towards Exascale. Taylor & Francis, 2022.
3. Svaton, V., et al., "HPC-as-a-Service via HEAppE Platform.", CISIS, Springer, Cham. (2020).
4. M. C. Cohen, P. W. Keller, M. Vahab and M. Zadimoghaddam, "Overcommitment in Cloud Services: Bin Packing with Chance Constraints," Management Science, p. 1–17, 2019.
5. S. H. Madni, et al., "Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment.", Cluster Computing, vol. 22, 2019.
6. M. Somnath, A. Scionti and A. S. Kumar, "Adaptive resource allocation for load balancing in cloud," in Cloud Computing, Springer, Cham, 2017, pp. 301-327.
7. S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," Journal of grid computing, vol. 14, no. 2, pp. 217-264, 2016.
8. M. Rahman, et al., "Adaptive workflow scheduling for dynamic grid and cloud computing environment," Concurrency and Computation: Practice and Experience, vol. 25, 2013.
9. A. Quarati and et al., "Scheduling strategies for enabling meteorological simulation on hybrid clouds," Journal of Computational and Applied Mathematics, vol. 273, pp. 438-451, 2015.
10. E. J. Korpela, "SETI@ home, BOINC, and volunteer distributed computing," Annual Review of Earth and Planetary Sciences, vol. 40, pp. 69-87, 2012.
11. A. Tsaregorodtsev et al., "DIRAC: a community grid solution," Journal of Physics: Conference Series, vol. 119, no. 6, p. 062048, 2008.
12. H. Casanova, et al., "Teaching parallel and distributed computing concepts in simulation with WRENCH", Journal of Parallel and Distributed Computing, Vol. 156 (2021).
13. Bak Slawomir, et al., "Gssim –a tool for distributed computing experiments." Scientific Programming 19.4 (2011): 231-251.
14. Buyya, Rajkumar, and Manzur Murshed. "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing." Concurrency and computation: practice and experience 14.13-15 (2002).
15. Mansouri, Najme, R. Ghafari, and B. Mohammad Hasani Zade. "Cloud computing simulators: A comprehensive review." Simulation Modelling Practice and Theory (2020).
16. Fiorucci P., et al., "Development and application of a system for dynamic wildfire risk assessment in Italy." Environmental Modelling & Software, 23(6), 690-702 (2008).
17. Silvestro, F., et al., "Exploiting remote sensing land surface temperature in distributed hydrological modelling: the example of the Continuum model." Hydrology and Earth System Sciences, 17(1), 39-62 (2013).