Verification, Validation and Uncertainty Quantification of large-scale applications with QCG-PilotJob

Bartosz Bosak[1], Tomasz Piontek[1], Paul Karlshoefer[2], Erwan Raffin[2], Jalal Lakhlili[3], and Piotr Kopta[1]

[1] Poznań Supercomputing and Networking Center, Poznań, Poland
[2] CEPP – Center for Excellence in Performance Programming, Atos, France
[3] Max-Planck Institute for Plasma Physics - Garching, Munich, Germany

Abstract

Efficient execution of large-scale and extremely demanding computational scenarios is a challenge for both the infrastructure providers and end-users, usually scientists, that need to develop highly scalable computational codes. Nevertheless, at this time, on the eve of exa-scale supercomputers, the particular role has to be given also to the intermediate software that can help in the preparation of applications so they can be efficiently executed on the emerging HPC systems. The efficiency and scalability of such software can be seen as priorities, however, these are not the only elements that should be addressed. Equally important is to offer software that is elastic, portable between platforms of different sizes, and easy to use. Trying to fulfill all the above needs we present QCG-PilotJob, a tool designed to enable flexible execution of numerous potentially dynamic and interdependent computing tasks in a single allocation on a computing cluster. QCG-PilotJob is built on many years of collaboration with computational scientists representing various domains and it responses to the practical requirements of real scientific use-cases. In this paper, we focus on the recent integration of QCG-PilotJob with the EasyVVUQ library and its successful use for Uncertainty Quantification workflows of several complex multiscale applications being developed within the VECMA project. However, we believe that with a well-thought-out design that allows for fully user-space execution and straightforward installation, QCG-PilotJob may be easily exploited in many other application scenarios, even by inexperienced users.

1 Introduction

The success of scientific research can be evaluated based on its applicability for solving real-world problems. Not surprisingly, before computational simulation codes are used in production, their robustness needs to be strictly proven. This applies to both, the quality of the code itself and, even more importantly, the reliability of the generated results. To this end, scientists employ VVUQ procedures to verify, validate, and precisely quantify the uncertainty of calculations.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_39

The inherent characteristic of the majority of available techniques is multiple evaluations of models using different input parameters selected from the space of possible values. This is a computationally demanding scenario even for evaluations of traditional single-scale applications, but in the case of multiscale simulations that consist of many coupled single-scale models, the problem becomes a real challenge. There is a need to support simultaneous execution of single-scale models that may have extremely large and different resource requirements, some single-scale models may need to be attached dynamically, the number of evaluations of single-scale models may not be known in advance, to name just a few difficulties. Within the VECMA project[4] we are trying to resolve all these issues with efficient and flexible tools. One of key elements in VECMA toolkit[5] [4] is EasyVVUQ[6] [8], the user-facing library that brings VVUQ methods to many different use-cases. However, EasyVVUQ itself abstracts from the aspects of execution of model evaluations on computational resources and outsources this topic to the external tools. One of them is QCG-PilotJob[7] (QCG-PJ) developed by Poznan Supercomputing and Networking Center as part of QCG (Quality in Cloud and Grid) middleware [7]. Its functionality, based on the idea of so called *Pilot Job*, which manages a number of subordinate tasks, is essential for the flexible and efficient execution of VVUQ scenarios that inherently consists of many tasks, from which some may be relatively small.

The rest of this paper is structured as follows. In Section 2 we present a brief overview of related work. In Section 3 we describe basic objectives and functionality of QCG-PJ. Next, in Section 4, we introduce a few schemes of integration between EasyVVUQ and QCG-PJ that have been developed in the VECMA project and then we present a range of application use-cases that already use QCG-PJ. The results of performance tests conducted so far are outlined in Section 5. Finally, in Section 6, we conclude and share main plans for the future.

2 Related work

The problem of efficient and automated execution of a large-number of tasks on computing clusters managed by queuing systems is known from decades. One of the most recognized systems that deal particularly with the pilot job style of execution on computing resources is RADICAL-Pilot[6], being developed as part of the RADICAL-Cybertools suite. In contrast to QCG-PJ, which can be easily installed in a user's home directory, RADICAL-Pilot is not a self-contained component and needs to be integrated with external services. There are also several solutions having some commonalities with the QCG-PJ idea, but they are focused primarily on the workflow orchestration rather than on the efficiency and flexibility of computing on HPC machines. An example of a mature system is here Kepler[2], which addresses the need for the HTC execution of parts of

[4] https//www.vecma.eu
[5] https://www.vecma-toolkit.eu
[6] https://github.com/UCL-CCS/EasyVVUQ
[7] https://github.com/vecma-project/QCG-PilotJob

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_39

the workflows on clusters. Further examples are Swift-T[9] and Parsl[1], which share a common goal to effectively support data-oriented workflows, or Dask[8], which aims to enable parallel processing for analytical workflows.

3 Objectives

The access to HPC systems is regulated by the policies of resource providers and restricted by local resource management system configurations as well as their implementations. For example, the policy at SuperMUC-NG cluster installed at the Leibniz Supercomputing Centre[9], in order to promote large-scale computing, allows users to submit and run only a small number of jobs at the same time. Smaller HPC installations may be less restrictive, but in general, the large tasks have a priority over small tasks, and the rule remains the same: there is no way to flexibly schedule many jobs with basic mechanisms. If users want to efficiently run a huge number of conceptually different tasks they need to employ solutions that can mitigate the regulations on a level of single allocation. One of the possible approaches is to define a processing scheme in a scripting language, but this is neither generic nor flexible and possibly prone to many bugs and inefficiency. The other, recommended approach is the utilisation of specialised software, like QCG-PJ.

Functionality

The basic idea of QCG-PJ is to bring an additional tasks management level within the already created allocation. As it is presented in Figure 1, from the queuing system's perspective, QCG-PJ is only a single regular task, but for a user, it is a second-level lightweight resource management system that can be administered and used on an exclusive basis.

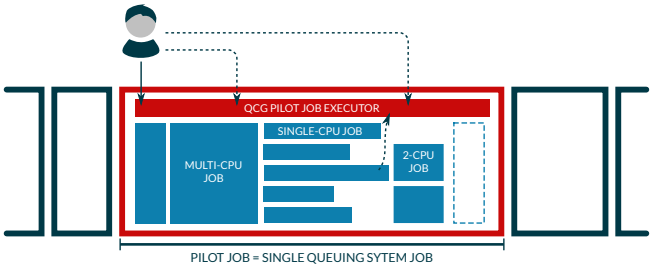

Fig. 1. The general computation scheme in QCG-PilotJob

As SLURM manages resources of a cluster, QCG-PJ manages resources of an allocation and ensures that tasks are scheduled efficiently. That being said,

[8] https://dask.org
[9] https://doku.lrz.de/display/PUBLIC/Job+Processing+with+SLURM+on+SuperMUC-NG

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_39

users or alternatively client software components can interact with it in a similar way as with any queuing system. Through a dedicated lightweight service called QCG-PJ Scheduler, they can submit new tasks with specific requirements, list submitted tasks or cancel them.

In order to enhance usability, QCG-PJ offers two ways of submission of tasks: firstly, it is possible to provide task definitions in a form of JSON file and submit this file from a command line at startup of the QCG-PJ, and secondly, it is possible to use the provided python API for dynamic creation and management of jobs directly from a running python program.

Moreover, the tool provides a few supplementary built-in features that can be recognised as particularly useful for selected scenarios. Among others, it allows defining dependencies between tasks as well as it offers resume mechanism to support fault tolerance at a workflow level.

Architecture towards exascale

One of the biggest challenges at the very beginning of QCG-PJ development was to ensure its ability to meet requirements defined by extremely demanding multiscale applications. In order to reach the performance of hundreds of petaflops or even higher, these applications ultimately call for, it was particularly important to design an appropriate architecture. First of all, such architecture should be scalable: the system should be easy to use, even on a laptop, but also easily extendable, portable and efficient once the use-cases grow up to require HPC resources. Consequently, the natural choice was to propose a hierarchical structure of components, where top-level services are released from the high-intensive processing that can be performed in a distributed way by low-level services. In consequence, the QCG-PJ architecture includes a concept of partition, which reflects a subset of resources that are managed separately and can be dynamically attached to the optional top-level QCG-PJ Scheduling Queue service. This is presented in Figure 2.

In the presented full-scale deployment scenario, QCG-PJ Scheduling Queue is an entry point to the system and keeps global information about all tasks that should be processed. One or multiple QCG-PJ Scheduler services, associated with the elementary partitions, can request Scheduling Queue for a portion of tasks to execute. Once the tasks are completed, the schedulers report this information back to the central service. Consequently, resources coming from a single or many allocations, also from a single or many HPC clusters, can be robustly integrated and offered as a single logical concept, while the communication overhead is minimised.

Having a closer look at the logic present within a single partition, two elements should be noted. The first of them is the possibility to reserve a core for QCG-PJ Scheduler. This option is useful when processing done by the Scheduler service significantly influences the actual computations. The second element is the Node Launcher service. This component is designed to improve the startup efficiency of single-core tasks.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_39

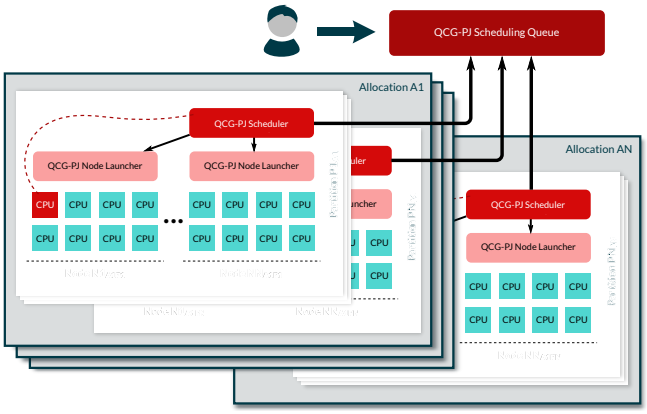

Fig. 2. QCG-PilotJob Architecture Overview

4 Use cases

Integration with EasyVVUQ

EasyVVUQ is a tool for domain experts who work on concrete VVUQ scenarios related to their applications. We argue that these experts shouldn't spend their valuable time to set-up the logic of execution of EasyVVUQ workflows on computing resources. Rather, they should focus on purely scientific or engineering aspects. To this end, VECMA toolkit provides a few approaches for the integration of EasyVVUQ with QCG-PJ so it can be efficient and natural for the domain scientist. Currently, there are the following possibilities:

Direct integration with EasyVVUQ: It is the most straightforward type of integration, where QCG-PJ is transparently employed in EasyVVUQ as one of its internal execution engines. Although at the moment of writing, this type of integration is not yet completed and doesn't benefit from more advanced features offered by QCG-PJ, e.g. iterative tasks, it is expected to be the preferred one at some point in the future.

Integration through the EQI library: EQI[10], which stands for EasyVVUQ-QCGPilotJob Integrator, is a lightweight library designed to bring optimised processing schemes to selected types of highly-demanding EasyVVUQ workflows. It makes use of advanced functionalities of QCG-PJ, like resume mechanism and iterative jobs.

Integration through FabSim3: QCG-PJ has been integrated with the FabSim3 automation toolkit[11] in order to support demanding application campaigns. Since FabSim3 internally uses EasyVVUQ, the combined execution of EasyVVUQ and QCG-PJ is also possible.

[10] https://github.com/vecma-project/EasyVVUQ-QCGPJ
[11] https://github.com/djgroen/FabSim3

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_39

Applications

At the moment of writing, there are already several application teams from VECMA that use QCG-PJ for their professional research. For instance, scientists from Max-Planck Institute for Plasma Physics use EasyVVUQ to quantify the propagation of uncertainty in a fusion turbulence model which is computationally expensive. It is a 3D parallel code and needs 512 to 16384 MPI cores [5]. Running a UQ campaign on such models require a very large number of jobs. Thanks to EQI and QCG-PJ, it was possible to execute the required simulations in a single batch allocation. In a similar way, the QCG-PJ tool has been employed for UQ of the UrbanAir application developed by PSNC [10]. It is also worth mentioning recent studies, where FabSim3 and QCG-PJ have been employed for UQ performed on the CovidSim epidemiological code [3].

5 Performance evaluation

Ultimately, the performance of QCG-PJ will be the most determining factor for its usability. Since the early days of its development, scalability and accessibility are evaluated repeatedly on large European supercomputers such as SuperMUC-NG at LRZ and Eagle/Altair at PSNC.
Thus far, test runs involving 100 dual-socket nodes, which equates to around 5000 CPU cores, showed very promising results, with more than 99% of time spent in the user-defined pilot jobs. More specifically, 20.000 pilot jobs with a runtime of five minutes each kept 99.2% of the available resources occupied.
With these promising results, we plan on conducting experiments with actual scientific applications which involve much larger node counts, alongside an exhaustive scalability study. Additionally, these tests were conducted by users which are not directly involved in the development of QCG-PJ, which in turn further contributed to the accessibility of the API.

6 Summary and future work

In this paper, we shortly introduced the concepts and features of QCG-PilotJob system and depicted how it is used by VECMA project to support demanding VVUQ scenarios. The progress made to several large-scale applications, when they successfully employed QCG-PJ, allows us to rank the current usability of QCG-PJ relatively high. Nevertheless, there are still ongoing works aimed to enhance the quality and functionality of the software. In regards to the former, since individual SLURM configurations can pose challenges for QCG-PJ and require its adaptation and customization, we are in the process of extensive tests of the tool on high-end European clusters. For instance, PSNC is in the process of deploying QCG-PJ to SURF (Amsterdam) and ARCHER2 (Edinburgh). Ultimately, we want to ensure that QCG-PJ is easily deployable and works efficiently on a large variety of machines and configurations. In terms of new functionality, our aim is to complete the implementation of the global Scheduling Queue service as well as to provide a dedicated monitoring solution.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_39

Acknowledgments This work received funding from the VECMA project realised under grant agreement 800925 of the European Union's Horizon 2020 research and innovation programme. We are thankful to the Poznan Supercomputing and Networking Center for providing its computational infrastructure. We are also grateful to the VECMA partners for the invaluable motivation.

References

1. Babuji, Y., Woodard, A., Li, Z., Katz, D.S., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J.M., Foster, I., Wilde, M., Chard, K.: Parsl: Pervasive parallel programming in python. In: Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing. p. 25–36. HPDC '19, New York, NY, USA (2019). https://doi.org/10.1145/3307681.3325400
2. Cabellos, L., Campos, I., del Castillo, E.F., Owsiak, M., Palak, B., Płóciennik, M.: Scientific workflow orchestration interoperating htc and hpc resources. Computer Physics Communications **182**(4), 890–897 (2011). https://doi.org/10.1016/j.cpc.2010.12.020
3. Edeling, W., Arabnejad, H., Sinclair, R., Suleimenova, D., Gopalakrishnan, K., Bosak, B., Groen, D., Mahmood, I., Crommelin, D., Coveney, P.V.: The impact of uncertainty on predictions of the covidsim epidemiological code. Nature Computational Science **1**(2), 128–135 (2021). https://doi.org/10.1038/s43588-021-00028-9
4. Groen, D., Arabnejad, H., Jancauskas, V., Edeling, W.N., Jansson, F., Richardson, R.A., Lakhlili, J., Veen, L., Bosak, B., Kopta, P., Wright, D.W., Monnier, N., Karlshoefer, P., Suleimenova, D., Sinclair, R., Vassaux, M., Nikishova, A., Bieniek, M., Luk, O.O., Kulczewski, M., Raffin, E., Crommelin, D., Hoenen, O., Coster, D.P., Piontek, T., Coveney, P.V.: Vecmatk: a scalable verification, validation and uncertainty quantification toolkit for scientific simulations. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **379**(2197), 20200221 (2021). https://doi.org/10.1098/rsta.2020.0221
5. Luk, O., Hoenen, O., Bottino, A., Scott, B., Coster, D.: Compat framework for multiscale simulations applied to fusion plasmas. Computer Physics Communications (2019). https://doi.org/10.1016/j.cpc.2018.12.021
6. Merzky, A., Turilli, M., Titov, M., Al-Saadi, A., Jha, S.: Design and performance characterization of radical-pilot on leadership-class platforms (2021)
7. Piontek, T., Bosak, B., Ciznicki, M., Grabowski, P., Kopta, P., Kulczewski, M., Szejnfeld, D., Kurowski, K.: Development of science gateways using qcg — lessons learned from the deployment on large scale distributed and hpc infrastructures. Journal of Grid Computing **14**, 559–573 (2016)
8. Richardson, R., Wright, D., Edeling, W., Jancauskas, V., Lakhlili, J., Coveney, P.: Easyvvuq: A library for verification, validation and uncertainty quantification in high performance computing. Journal of open research software **8**, 1–8 (2020)
9. Wozniak, J.M., Armstrong, T.G., Wilde, M., Katz, D.S., Lusk, E., Foster, I.T.: Swift/t: Large-scale application composition via distributed-memory dataflow processing. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. pp. 95–102 (2013). https://doi.org/10.1109/CCGrid.2013.99
10. Wright, D.W., Richardson, R.A., Edeling, W., Lakhlili, J., Sinclair, R.C., Jancauskas, V., Suleimenova, D., Bosak, B., Kulczewski, M., Piontek, T., Kopta, P., Chirca, I., Arabnejad, H., Luk, O.O., Hoenen, O., Weglarz, J., Crommelin, D., Groen, D., Coveney, P.V.: Building confidence in simulation: Applications of easyvvuq. Advanced Theory and Simulations **3**(8), 1900246 (2020)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_39