Uncertainty Quantification-as-a-Service

Małgorzata Zimoń
IBM Research
Warrington, Cheshire, United Kingdom
Malgorzata.Zimon@uk.ibm.com

Vadim Elisseev
IBM Research
Warrington, Cheshire, United Kingdom
vadim.v.elisseev@ibm.com

Robert Sawko
IBM Research
Warrington, Cheshire, United Kingdom
rsawko@uk.ibm.com

Samuel Antão
IBM Research
Warrington, Cheshire, United Kingdom
samuel.antao@ibm.com

Kirk Jordan
Data Centric Solutions, IBM T.J. Watson Research
Cambridge, MA, USA
kjordan@us.ibm.com

ABSTRACT

Uncertainty Quantification (UQ), which enables non-destructive virtual testing, is the fast growing area of modern computational science. UQ methods are computationally intensive and require construction of complex work-flows, which rely on a number of different software components often coming from different projects. Therefore, there is a need for developing a portable and scalable UQ pipeline that will enable efficient stochastic modelling. Our paper introduces a strategy for UQ as a Service (UQaaS) using High Performance Computing (HPC) and Hybrid Cloud infrastructures and presents its application to a heat transfer study in nuclear reactors simulation and modelling of tsunami events.

KEYWORDS

Uncertainty Quantification, Microservices, Cloud, High Performance Computing, Modelling

ACM Reference Format:
Małgorzata Zimoń, Vadim Elisseev, Robert Sawko, Samuel Antão, and Kirk Jordan. 2019. Uncertainty Quantification-as-a-Service. In *Proceedings of CASCON (CASCON'18)*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 MOTIVATION

Uncertainty quantification (UQ) can be defined as the end-to-end study of the reliability of scientific inferences [4, 23].

There are two common categories of discrepancies identified both in physical and numerical experiments: *aleatoric* and *epistemic*. The first type refers to uncertainty due to the physical variability of a system (irreducible), while the latter arises from a lack of knowledge (reducible). UQ in computer experiments comprises rigorous analysis of the epistemic errors that can classified as uncertainty-model related, where the correctness of the strategy is questioned, and uncertainty-parameters related caused by inadequately chosen values for representing certain properties.

Of particular interest in modelling of real-world problems is parametric UQ, which is identified as a crucial component of simulation-based reliability analysis. To increase confidence in modelling predictions, we need to account for discrepancies which stem naturally from our limitations in measuring, manufacturing, and replacing deterministic simulations with stochastic framework. Therefore, there is a growing demand in strategies that enable solving large-scale uncertainty quantification problems without being affected by poor performance, scalability and resilience.

Conventional non-intrusive approaches for UQ are based around Monte Carlo (MC) analysis which requires a large number of simulations for different sets of input parameters. An alternative strategy uses a smaller number of realisations to build a surrogate model which approximates the response of the original simulator. A common surrogate-based methods are polynomial chaos and Gaussian process emulation [24].

Notwithstanding there are approaches that address UQ objectives at a fraction of the original cost, the analysis is still expensive and requires significant computational resources, particularly for high-dimensional problems. In addition, it is difficult to assess *a priori* how many calculations will be needed to obtain a converged stochastic response of a system. Moreover, the study relies on large numbers of complex software packages and modules to generate the appropriate input files, run the simulations and perform the statistical analysis. Therefore, creating portable and scalable uncertainty-quantification pipelines that enable efficient stochastic modelling is a must.

The aim of this project is to address challenges of the UQ modelling by developing the **Uncertainty Quantification as a Service (UQaaS)** framework, which can take advantage of modern HPC and Cloud technologies. As proof-of-concept we plan to demonstrate UQaaS for nuclear and tsunami events modelling, but our proposed approach is generic and can lead to improved reliability analysis for any computational simulations, regardless of its domain.

The rest of the paper is organised as follows. Section 2 describes the multi-level Monte Carlo (MLMC) method, which is a suitable UQ strategy for complex stochastic systems. Section 3 provides examples of application work-flows, which we intend to use in our studies. Section 5 outlines the proposed design of the UQ as a service framework. Finally, Section 6 provides conclusions and plans for future work.

CASCON'18, October 2018, Toronto, Ontario Canada
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 MULTI-LEVEL MONTE CARLO

When the dimensionality of the uncertainty (number of varying input parameters) is high, estimation of an expected value arising from a stochastic system is a very expensive task, particularly in the domain of engineering applications. Many sophisticated surrogate-based techniques fail to provide a feasible solution and Monte Carlo methods are preferred due to their simplicity and independence from stochastic dimension.

The cost of classical random sampling is related to the required estimation confidence levels. For classic Monte Carlo (MC), confidence intervals scale inversely with the square root of the number of realisations. Many approaches seek to improve this sublinear scaling or the overall efficiency of the sampling method. For instance, the combination of MC and multi-grid strategies, referred to as the multi-level Monte Carlo (MLMC), enables reducing the variance in MC statistics. This method employs a nested sequence of grids, each typically twice as fine as the previous, to recover the accuracy of the finest mesh at minimal computational cost.

MLMC decomposes the output random variable into a sequence of differences between realisations on two consecutive resolution levels and performs sampling on these differences. Intuitively, the cost reduction is achieved because at high resolution a converging numerical solution to a partial differential equation should exhibit vanishing differences and therefore the variance of MC estimations will decrease with each level. If q is the output variable and l is the spatial resolution level the MLMC constructs an estimate of the following expected value:

$$\mathbb{E}[q] \approx \mathbb{E}[q_L] = \mathbb{E}[q_0] + \sum_{\ell=1}^{L} (\mathbb{E}[q_\ell] - \mathbb{E}[q_{\ell-1}]) \qquad (1)$$

where $\mathbb{E}[\cdot]$ is the expected value and L is the total number of resolution levels. The MLMC estimator is then:

$$\mathbb{E}[q]_L \approx E^{\text{MLMC}}[q_L] = \frac{1}{M_0}\sum_{i=1}^{M_0} q_0^{(i)} + \sum_{\ell=1}^{L} \frac{1}{M_\ell}\sum_{i=1}^{M_\ell} (q_\ell^{(i)} - q_{\ell-1}^{(i)}), \qquad (2)$$

where M_ℓ for $\ell = 0, 1, \ldots L$ is the number of samples at each level. Then the notion of decreasing variance can be expressed as follows:

$$\epsilon^2 = \mathbb{E}\left[\left(E^{\text{MLMC}}[q_L] - \mathbb{E}[q_L]\right)^2\right] = \frac{\mathbb{V}[q]_0}{M_0} + \sum_{\ell=1}^{L} \frac{\mathbb{V}[q_\ell - q_{\ell-1}]}{M_l} \qquad (3)$$

and the overall reduction of the computational cost is obtained if the variance function $\mathbb{V}[q_\ell - q_{\ell-1}]$ is decreasing sufficiently fast.

In order to obtain an optimal number of samples at each level ℓ the classic MLMC solves a constrained minimization problem requiring tolerance ϵ to be below a given threshold and minimizing the total work function:

$$W = \sum_{\ell=0}^{L} M_\ell W_\ell \qquad (4)$$

where W is the total work and W_ℓ is the computational effort required at each level. More detailed description of the method can be found in [11]. It is worth noting that neither the work function W_ℓ nor the $\mathbb{V}[q_\ell]$ functions are known *a priori*. They need to be estimated on the fly and existing variants of MLMC make various assumptions about the form of these functions to arrive at an analytical solution of the optimisation problem. Because of this, implementations of MLMC method become iterative procedures in which we draw a new set of samples, correct the information about W_ℓ and $\mathbb{V}[q_\ell]$ until convergence is reached. The convergence of this procedure is case-dependent and relies on the validity on the assumptions we make about the work and variance functions.

There are several reasons why this non-intrusive UQ approach can benefit from "as a service" framework, which uses HPC cloud resources augmented by Artificial Intelligence (AI). On one hand, the increase of resolution means that the cost of performing a simulation at a given level may increase rapidly. On the other hand, weak-scaling has been demonstrated for many engineering work-flows meaning that a flexible HPC system could keep the computation time bounded. Moreover, because of lack of *a priori* knowledge about model sensitivity (variance), computation effort as well as parallel-performance characteristics of the simulation code, the approach can potentially gain from adaptation offered by machine learning (ML) techniques. An ML model could be employed to make input dependant run-time predictions so as to optimise allocation of HPC resources and reducing the time to solution of the entire UQ work-flow.

Additionally, a higher degree of flexibility can be achieved as we lift the assumptions related with the work and variance functions. As MLCM is dependent on a number of parameters, such as discretisation hierarchy, the number of levels (mesh resolutions) and the number of samples per level, a pre-processing step can determine optimal settings, minimising the cost or the error of the study.

3 APPLICATIONS

UQ analysis is crucial in several application domains and is usually employed in risk analysis. In this section we describe how UQ is applied to two different real-world cases: carbon deposition on fuel pins and extreme flooding. Both problems possess similar features: relative scarcity of available data, reliance on a computational model, large number of uncertain parameters and the need for adaptive methods.

3.1 Carbon Deposit on Fuel Pins

Advanced gas-cooled reactors (AGRs) are currently the main source of nuclear energy in the UK. The fission fuel is delivered to the reactor in so called fuel assemblies and the lifetime of these objects depends partly on the ability to keep them at sufficiently low temperatures. Fuel assemblies are composed of fuel pins with helical ribs that are meant to increase the efficiency of heat transfer. Finally, the gas-coolant is CO_2.

In the extreme environment with high temperatures and pressures, layers of carbon deposit have been observed to form on fuel pin surface (see Figure 1). The presence of these layers affects the heat transfer due to the insulating properties of carbon and changes to the effective height of the ribs, deteriorating the efficiency of the cooling process. This leads to increased temperature in the steel component, which will result in thermal fatigue and increased pin failures [12].

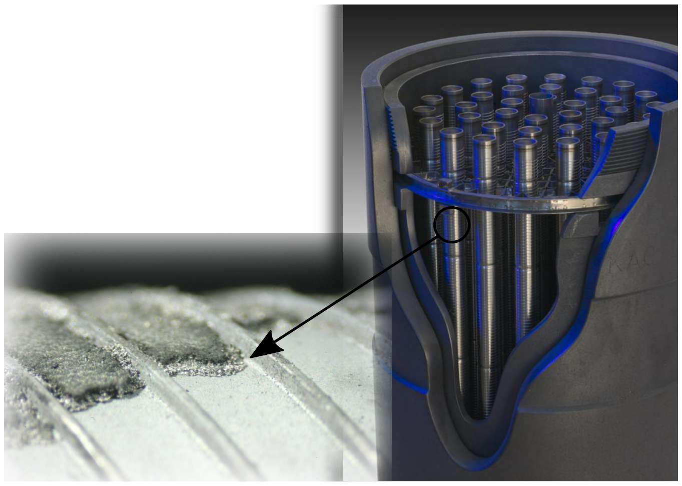
Figure 1: Right: AGR fuel pin assembly. Left: carbon deposit.

In order to understand these failures, engineers study heat transfer impairment (HIT) which is obtained from computational fluid dynamics models that describe heat transfer of the coolant and the solid components on the pin surface including steal and deposit part. HIT is later used in process models that represent the bulk behaviour of the plant and are used in optimisation and design.

One of the current main limitations is the inability to properly describe the effect of heterogeneity in the carbon deposit structure and uncertainties related to its morphology. Lifting this limitation requires introducing a stochastic input capturing changing material properties (e.g., thermal conductivity). The stochastic structure can be captured by a real valued Gaussian random field with fixed first and second order moments. This in turn leads to a large number of simulation inputs making it infeasible to tackle it with standard surrogate-based methods. To estimate moments and other probabilistic properties of HIT, the MLMC technique described in Sec. 2 is a much more appealing solution.

3.2 Modelling Tsunamis

Rare catastrophic events such as the Sumatra tsunami of 26 December 2004 and the disastrous flooding of Fukushima Daichi Nuclear Power Plant of March 2011 remain a threat to life, property and infrastructure located along the coast. According to some estimates since 1850 tsunamis resulted in the loss of approximate 420,000 lives and cost billions of dollars in economical terms [3].

In response to these events several tsunami warning centres have been established and continue to operate around the world. These centres regularly run computer codes based on the shallow-water equations to simulate various coastal flooding scenarios. The methodologies for these computational models have also been progressing [2] and there are now several well-established operational programs as well as new promising research software projects. The parallel performance of these codes is still a subject of active development but, more often than not, the packages are used in the context of high-throughput computing.

Unfortunately, the models as well as computer codes suffer from a large degree of uncertainty. On the modelling side the issues are related to tsunami sources which can come from earthquakes, submarine landslides, rock slides and volcanic eruptions, each requiring its own specific modelling and its own set of uncertain parameters. Another aspect are model parameters required for wave propagation such as Manning's coefficients, which tie in space the empirical roughness coefficient with the friction factor. These parameters are spatially heterogeneous and non-isotropic. Combined, these issues quickly result in a high dimensional problem that requires a large number of simulations.

Finally, a lot of work has been done in recent years in statistical processing including Bayesian inversion procedures. There has been progress in the development of sampling techniques aiming to reduce the number of numerical experiments by selecting a smaller but representative sample. Also, static experiment design is being progressively replaced by "active" design, also known as sequential, which employs machine learning techniques to select better future query points.

4 UQ WORK-FLOW

In this section, we introduce the typical work-flow that is used for UQ studies in computational fluid dynamics. Figure 2 depicts two major components of the procedure and their interrelationships: UQ module and a user-supplied simulation code.

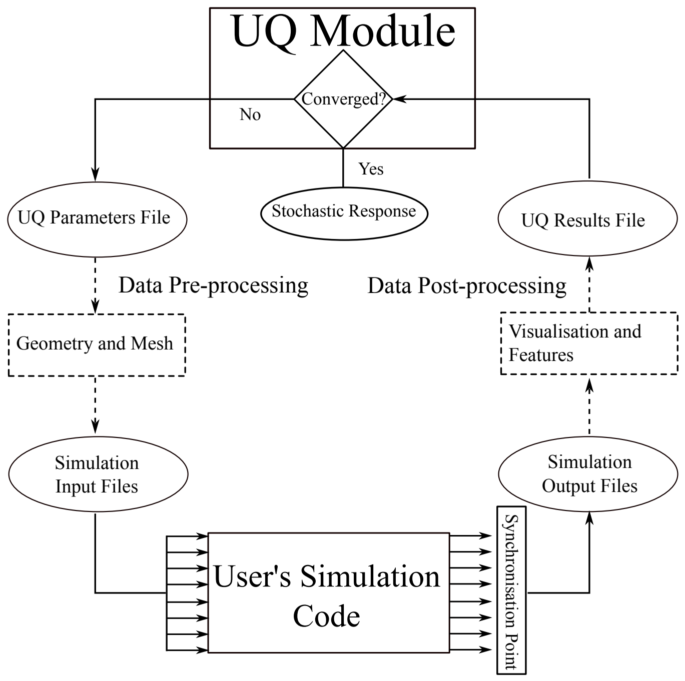

Figure 2: High-level representation of non-intrusive UQ work-flow. Multiple arrows indicate high-throughput calculations.

UQ Module, which is a newly developed software component, generates values that can be either transferred to simulation input files or used in the pre-processing stage to obtain suitable inputs for the solver. Each set of values is a different realisation of the process modelled in the uncertainty analysis (e.g., shape of the fuel pin deposit or frictional effects between the water column and the seabed). Realisations form a part of computational experiment design and can be chosen: *(*i)) uniformly, *(*ii))through a requirement of specific optimality condition, *(*iii)) through a quadrature rule in surrogate model projection methods, *(*iv)) expected improvement method or, *(*v)) randomly in Monte-Carlo fashion.

In case of geometric uncertainty, a realisation describes a shape of the modelled system. Subsequently, a mesh module, such as Gmsh [10] or Smesh [8], will use these values to produce the corresponding grid file for the simulations. This stage can be computationally expensive, in particular for high-resolution grids and thus require HPC resources.

Once all input files are ready, the simulations are run on an HPC cluster or cloud. Due to the complexity of the equations, numerical algorithms and any data storage operations, it can take a substantial amount of time (a couple of days or even weeks) to complete the simulations.

After the simulation stage is completed, the results are collected and considered for analysis. The post-processing of the data might require an additional visualisation application for extracting quantities of interest, for example ParaView [22]. In case of large data sets, support for distributed computations is required. A UQ module will read the modified results and evaluate the response statistics. If the convergence of response surface in the space of random parameters is not achieved, the whole work-flow is repeated with more sample points. Before the experiment is executed, the cost of such a numerical study is not known.

For each new set of inputs, the full simulation flow have to be repeated. Because of this, the complete UQ work-flow for the MLMC method described in Section 2 comprises three levels of parallelism [7]:

- level-parallelism,
- sample-parallelism and
- solver-parallelism.

The level parallelism comes from solving the same problems at several different resolution levels. To compute the last term of Equation 2, we need to obtain the pair of solutions for the same input value at two consecutive levels for the total of $L-1$ levels. The sample-parallelism is a consequence of the Monte-Carlo method. Solver parallelism is exposed when the computation for a single sample is sufficiently large to be partitioned across HPC resources.

Finally, often the full experiment is preceded by an intensive benchmarking exercise in which the modeller adapts the batch execution parameters such as core counts, memory limitations, GPU allocations. The purpose of this is to evaluate the work functions of Equation 4 and find a combination which guarantees good performance for the complete experimental campaign. These choices are rarely optimal for the full domain of interest. With the help of modern ML methods, this step could be largely automated with model retraining using the evaluated data in each step of the MLMC.

5 UQ AS A SERVICE

Deployment of the UQ as a Service for work-flows described in previous sections presents a number of challenges from usability, infrastructure and operation perspectives. It needs to provide a friendly interface for domain science experts as well as less experienced users, who may be looking into integrating a UQ pipeline into a more complex work-flow.

It also needs to provide means to optimise performance of different components by utilising HPC resources in potentially Hybrid Cloud configurations. Modularity of the UQaaS pipeline is also important to allow for easy replacement of different simulation components based on requirements of particular applications. Last, but not least, the pipeline has to be portable to allow enable a straightforward deployment across different private and public environments.

Figure 3 depicts a most generic version of the UQ-as-a-Service design. Major pipeline components are divided into **Core Components** and **Simulation Components**. **Core Components** consist of the following modules:

- **Front End Module** provides graphical (GUI), command line (CLI) and application programming (API) user interfaces,
- **UQ Module** is responsible for generating UQ input parameters and work-flow graph using one of the work-flow languages, for example Common Workflow Language (CWL) [1] or Workflow Description Language(WDL) [20].
- **Work-Flow Engine**, translates work-flow graph into format understood by the **Resource Manager** module,
- **Resource Manager** module, is responsible for scheduling Simulation Components on HPC resources,
- **AI Module** has two functions:

(1) Selection of the cost function used by the UQ Module to generate a more optimal set of UQ Parameters.
(2) Assisting Resource Manager to analyse various metrics originating in the Simulation Components and provide run time predictions to the UQ Module and Resource Manager.

Simulation Components consist of applications required for a particular model and domain. Simulation Components can be executed as batch jobs on HPC resources managed by the Resource Manager. For our use cases, Simulation Components will consist of:

- Mesh generation components, for example Gmsh [10] or Smesh from the SALOME project [8]
- Simulation application.
- Post-processing application.

Core Components can be deployed as Microservices using Docker containers [6] on the Kubernetes-based platform [19]. Simulation Components may also be deployed in containers to address portability requirements.

5.1 UQaaS in Cloud

UQaaS can take advantage of Cloud computing by using its computational and storage resources. *"Pay as you go"* Cloud model [21] enables the optimisation of performance versus costs metrics. The UQaaS deployment can start with just Core Components using on-premises resources (a single server may be sufficient in many cases) and use off premises HPC resources to deploy Simulation Components and scale them up and down based on simulations requirements. Figure 4 shows how UQaaS can be used with a Hybrid Cloud by adding a **Cloud Broker** component to the Core Services.

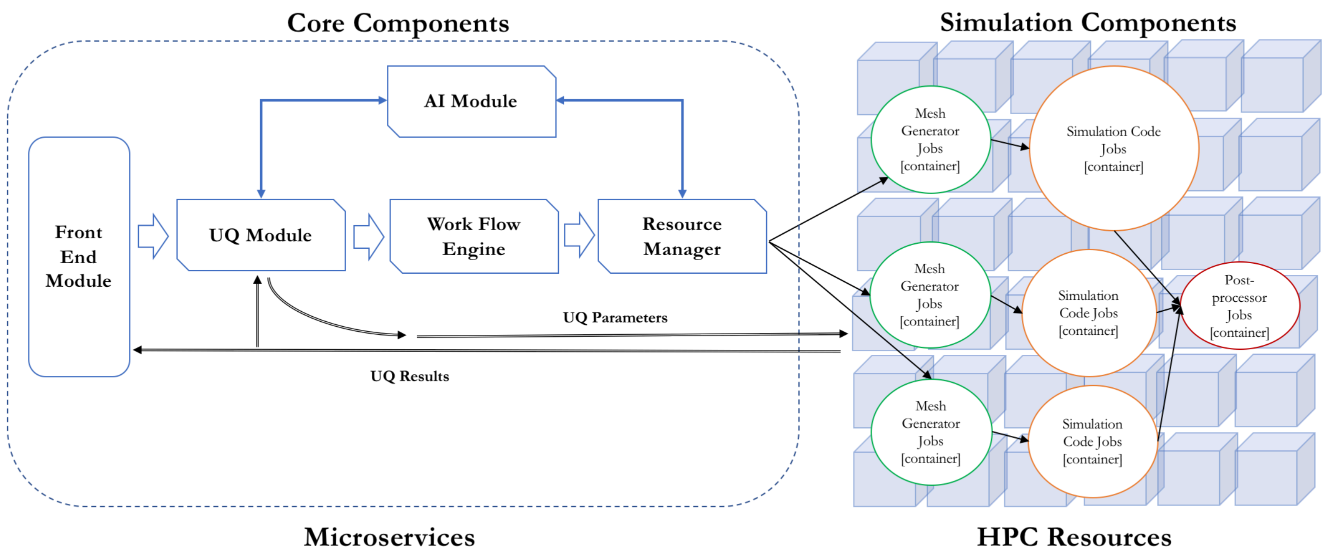

Figure 3: Proposed UQaaS framework: Core Components as micro-services, Simulation Components as potentially containerized HPC batch jobs.

The next section describes one particular implementation of such a Cloud Broker.

5.2 UQaaS Implementation

Previous sections described our vision of the UQaaS in general terms. In this paragraph we show how we plan to implement UQaaS using software and hardware that is already available today.

Figure 5 depicts a suggested Proof of Concept implementation of the UQaaS on OpenPOWER ™ [9] using a number of proprietary and Open Source components. **Core Components** can be deployed as Docker containers on the IBM Cloud Private [14] platform, which is based on Kubernetes. HPC sources can be organised into an IBM Spectrum LSF [13] cluster. UQ Module generates a CWL work-flow, which is translated by the CWLEXEC [5] or Cromwell [18]component into LSF submission scripts. LSF Resource Connector [17] can be used as Cloud Broker to enable the dynamic scaling of the LSF cluster based on required job resources. IBM Spectrum Scale provides high performance storage, which can be used in on-premises and in Hybrid Cloud configurations. The AI

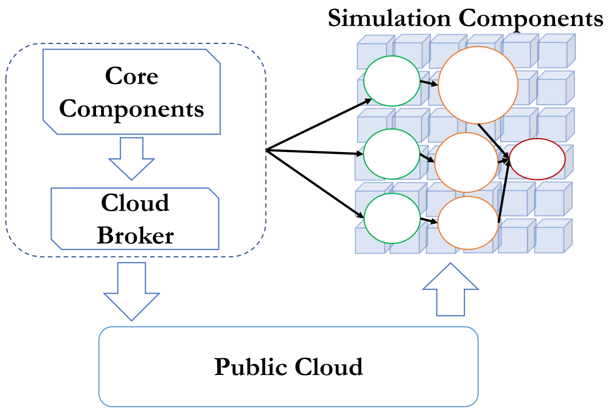

Figure 4: UQaaS in a hybrid cloud set-up.

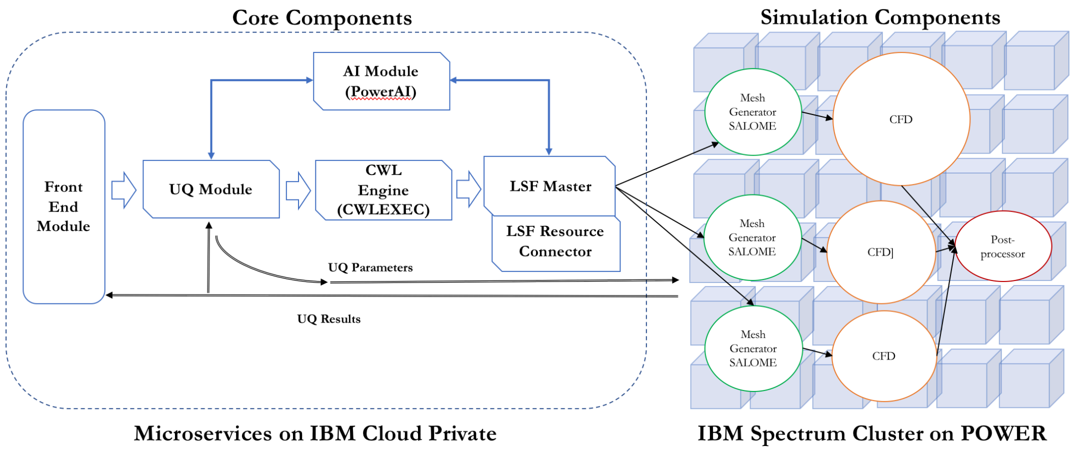

Figure 5: Envisioned proof of concept implementation of UQaaS

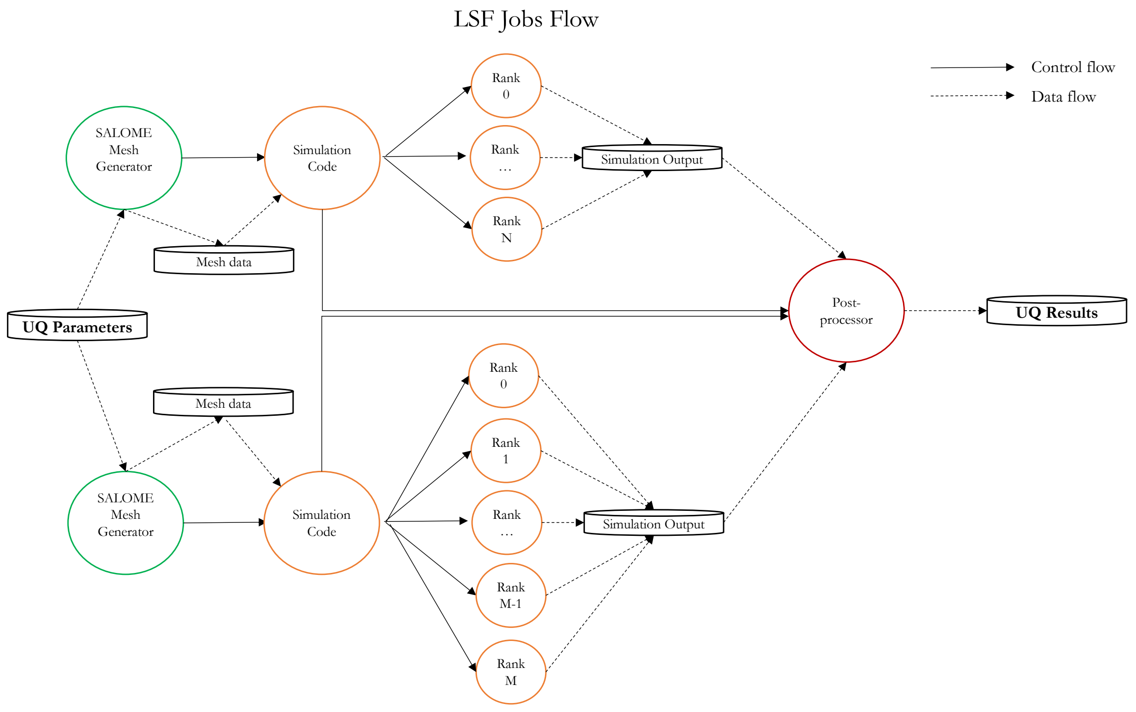

Figure 6: UqaaS simulation flow as LSF jobs.

Module can be implemented using PowerAI software stack [16]. Several options are available for the Front End Module, for example SALOME desktop [8]. OpenPOWER platform has been selected for its proven suitability for HPC, data centric and AI applications [15].

It is important to emphasise that UQaaS design aims at high utilisation rate of computational resources. The latter can be achieved by concurrent execution of multiple UQ pipelines, each corresponding to a different set of UQ parameters. Figure 6 shows Simulation Components as an LSF jobs flow. Simulation jobs depend on the successful completion of the respective mesh generating jobs and the availability of meshes. Final data post-processing job works as synchronisation point for multiple pipelines. Data movement throughout the UQaaS work-flow is handled by Spectrum Scale and LSF.

6 CONCLUSIONS AND FUTURE WORK

We presented a blueprint of intelligent, modular, scalable and portable UQaaS infrastructure, which will provide easier access to HPC resources for domain scientists. The main advantage of our approach is the clear separation of roles: the domain expert delivers the computational model, the UQ and AI modules generate the work-flow and the hybrid cloud supplies scalable computational and storage resources in a cost effective manner. End-users with various backgrounds and technical skills interact with the system through a front-end that provides GUI, API and CLI endpoints. We believe that this solution will allow non-HPC experts to take advantage of HPC resources for tackling UQ problems.

We have described two examples to demonstrate how distinct application areas can generate UQ problems and the resulting workflows. These examples are not unique and there is a growing number of applications which will benefit from UQaaS.

AI is a key component of our approach and can be used for selection of UQ cost functions to reduce number of required simulation runs as well as to improve the overall efficiency in the utilisation of HPC resources, by predicting run times of UQ pipelines.

Our immediate next steps include proof of concept implementation and evaluation of the proposed UQaaS infrastructure on OpenPOWER platform using software components described in Section 5.2.

REFERENCES

[1] P. Amstutz, M. R. Crusoe, N. Tijani, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Mnager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, and L. Stojanovic. 2016. Common Workflow Language, v1.0. Specification, Common Workflow Language working group.

[2] J. Behrens and F. Dias. 2015. New computational methods in tsunami science. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 373, 2053 (2015).

[3] E. N. Bernard, H. O. Mofjeld, V. Titov, C. E. Synolakis, and F. I. González. 2006. Tsunami: scientific frontiers, mitigation, forecasting and policy implications. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 364, 1845 (2006), 1989–2007.

[4] A. Bishop and P. Messina. 2009. U.S. Department of Energy. Scientific Grand Challenges for National Security: The Role of Computing at the Extreme Scale. , 135 pages.

[5] IBM Spectrum Computing. 2018. CWLEXEC. https://github.com/IBMSpectrumComputing/cwlexec.

[6] Docker. 2018. IDocker. https://www.docker.com/

[7] D. Drzisga, B. Gmeiner, U. RÃijde, R. Scheichl, and B. Wohlmuth. 2017. Scheduling Massively Parallel Multigrid for Multilevel Monte Carlo Methods. *SIAM Journal on Scientific Computing* 39, 5 (2017), S873–S897.

[8] EDF. 2018. SALOME. https://www.salome-platform.org/

[9] OpenPOWER Foundation. 2011. OpenPOWER. https://openpowerfoundation.org/.

[10] C. Geuzaine and J.-F. Remacle. 2009. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering* 79, 11 (2009), 1309–1331.

[11] M. B. Giles. 2008. Multilevel Monte Carlo path simulation. *Operations Research* 56, 3 (2008), 607–617.

[12] C. Gras and S. J. Stanley. 2008. Post-irradiation examination of a fuel pin using a microscopic X-ray system: Measurement of carbon deposition and pin metrology. *Annals of Nuclear Energy* 35, 5 (2008), 829–837.

[13] IBM. 2017. IBM Spectrum LSF. https://developer.ibm.com/storage/products/ibm-spectrum-lsf/

[14] IBM. 2018. IBM Cloud Private. https://www.ibm.com/cloud/private

[15] IBM. 2018. IBM POWER9. https://www.ibm.com/support/knowledgecenter/en/POWER9/p9hdx/POWER9welcome.htm

[16] IBM. 2018. IBM PowerAI. https://developer.ibm.com/linuxonpower/deep-learning-powerai/.

[17] IBM. 2018. LSF Resource Connector. https://www.ibm.com/support/knowledgecenter/en/SSWRJV_10.1.0/lsf_welcome/lsf_kc_resource_connector.html.

[18] Broad Institute. 2018. Cromwell - Workflow Management System. https://github.com/broadinstitute/cromwell.

[19] Kubernetes. 2018. Kubernetes. https://kubernetes.io/

[20] OpenWDL. 2018. Workflow Description Language(WDL). http://www.openwdl.org/.

[21] T. Grance P. Mell. 2011. The NIST Definition of Cloud Computing. https://csrc.nist.gov/publications/detail/sp/800-145/final.

[22] ParaView. 2018. ParaView. https://www.paraview.org/

[23] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofar, W. Gropp, Lurie E., and D. Mavriplis. 2014. CFD vision 2030 study: a path to revolutionary computational aerosciences. , 51 pages.

[24] D. Xiu and J. Hesthaven. 2005. High-Order Collocation Methods for Differential Equations with Random Inputs. 27 (01 2005), 1118–1139.