MIPRO 2022, May 23 - 27, 2022, Opatija, Croatia

Practice and Experience using High Performance Computing and Quantum Computing to Speed-up Data Science Methods in Scientific Applications

M. Riedel[*†], M. Book[†], H. Neukirchen[*], G. Cavallaro[†]
A. Lintermann[†],
[*] School of Engineering and Natural Sciences, University of Iceland, Iceland
[†] Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany
morris@hi.is, book@hi.is, helmut@hi.is, g.cavallaro@fz-juelich.de, a.lintermann@fz-juelich.de

Abstract—**High-Performance Computing (HPC) can quickly process scientific data and perform complex calculations at extremely high speeds. A vast increase in HPC use across scientific communities is observed, especially in using parallel data science methods to speed-up scientific applications. HPC enables scaling up machine and deep learning algorithms that inherently solve optimization problems. More recently, the field of quantum machine learning evolved as another HPC related approach to speed-up data science methods. This paper will address primarily traditional HPC and partly the new quantum machine learning aspects, whereby the latter specifically focus on our experiences on using quantum annealing at the Juelich Supercomputing Centre (JSC). Quantum annealing is particularly effective for solving optimization problems like those that are inherent in machine learning methods. We contrast these new experiences with our lessons learned of using many parallel data science methods with a high number of Graphical Processing Units (GPUs). That includes modular supercomputers such as JUWELS, the fastest European supercomputer at the time of writing. Apart from practice and experience with HPC co-design applications, technical challenges and solutions are discussed, such as using interactive access via JupyterLab on typical batch-oriented HPC systems or enabling distributed training tools for deep learning on our HPC systems.**

Keywords—High-Performance Computing; Software Framework; Machine Learning; Deep Learning; Quantum Computing

I. Introduction

Many scientific and engineering user communities increasingly use High-Performance Computing (HPC) with parallel data science methods to speed-up their Artificial Intelligence (AI) applications worldwide in general and at our Juelich Supercomputing Centre (JSC)[1] in particular. Examples include the many AI user communities of our Helmholtz AI Unit[2] at JSC [1], or in specific EU research projects such as the European Centre of Excellence - Research on AI- and Simulation-Based Engineering at Exascale (CoE RAISE)[3]. In close collaboration with JSC, the Simuluation and Data Lab Remote Sensing[4] of the Icelandic HPC (IHPC) National Competence Center (NCC) for HPC and AI increasily use HPC resources at scale that contribute to the findings of this paper. Examples include using Generative Adversarial Networks (GANs) on HPC [2] or using Convolutional Neural Networks (CNNs) with a large number of Graphical Processing Units (GPUs) to solve complex remote sensing problems [3, 4, 5, 6].

This paper provides insights in how the above mentioned practices and experiences in using HPC to speed-up data science methods shaped the unique AI software framework layout design of the CoE RAISE by using application co-design[5]. It describes how the design of this framework is based on profound lessons learned gained in using cutting-edge HPC resources with parallel data science methods and AI algorithms over the last years. Adopting HPC at scale for AI researchers is still challenging (i.e., combination of AI/HPC components unclear, etc.), one goal of this framework is to provide a blueprint for new AI/HPC communities capturing our lessons learned. While this paper focuses on using traditional HPC resources with data science methods, we are not loosing sight of the latest developments of using Quantum Computing (QC) for AI. Hence, this paper also takes into account our recent experiences [7, 8, 9] of using novel quantum annealer systems to perform Quantum Machine Learning (QML).

The remainder of this paper is structured as follows. While the scene is set in Section I, Section II reviews related work and compares existing approaches to our approach of designing an AI software framework to foster existing experience and lessons learned using HPC. Section III then provides details about our gathered practice and experience of using AI models on cutting-edge HPC resources over the years, also including the latest findings of our CoE RAISE project. In Section IV, we further identify important functionalities of an AI software framework for Exascale by describing lessons learned from our application co-design activities at the JSC (in collaboration with the University of Iceland) in general, and CoE RAISE in particular. Section V then presents the core building blocks of our AI framework design based on our lessons learned as well as practice and experience scaling up towards Exascale over the last years. Since our experiences with QC are still relatively less compared to traditional HPC resources, but acknowledging that it is important for AI, we discuss the integration approaches of novel quantum annealing techniques in Section VI. This paper ends with some concluding remarks.

[1]https://www.fz-juelich.de/ias/jsc/EN/Home/home_node.html
[2]https://www.helmholtz.ai/themenmenue/our-research/consultant-teams/index.html
[3]https://www.coe-raise.eu/
[4]https://ihpc.is/simulation-and-data-lab-remote-sensing/
[5]https://www.coe-raise.eu/news-2021-05

II. Related Work

In the context of our CoE RAISE that focus on data science methods towards Exascale using HPC, there is the European Center of Excellence in Combustion (CoEC)[6] that also works on similar engineering-oriented use cases. In contrast to CoE RAISE however, the CoEC focus on combustion with physic use cases and thus does not include very much data-driven use cases and additionally does also not primarily focus on AI methods towards Exascale.

In more detail, our identified data science methods in the use cases as shown in Table I have been used in HPC with other use cases and thus we survey this related work here. Auto-Encoder (AE) [10] and Convolutional Auto-Encoder (CAE) have been used with HPC in computational fluid dynamics (CFD) as described by Glaws et al. in [11], but not including elements of a larger framework as shown in this paper. The same is true for Physics-Informed Neural Networks (PINNs) that are a specific form of a new general approach in AI called Physics-Informed Machines Learning (PIML) [12].

Feed-forward Artificial Neural Networks (ANNs) and its specific variants like Radial Basis Function Artificial Neural Networks (RBF-ANN) are considered rather shallow learners and as such do not require much HPC resources. As a consequence, scaling up ANNs and integrating them into a software framework towards Exascale is rather unique in CoE RAISE. But real DL models like Convolutional Neural Networks (CNNs) or its specific variants like U-Nets [13] or Residual Networks (RES-NET) are used with HPC as benchmarks, e.g. by Jiant et al. in [14]. In contrast, our framework design is influenced by using RES-NET with applications as shown in [6] using more than 128 GPUs in parallel with pyTorch-DDG or Horovod, including hyper-parameter tuning with Raytune.

Neural Operators (NOs) and specific variants like Fourier NOs (FNOs) are a very particular ML-based methods to model 2D turbulent flows [15]. Some related work of NOs and FNOs in data and physical sciences using HPC are described in the special issue by Xue et al. in [16], but do not consider usually the larger software framework blueprint of the HPC ecosystem that is the focus in this paper. The same is true for using very specific Statistical Methods (SMs) such as Auto-Regressive models (ARs), AR Moving Average models (ARMAs), or AR Integrated Average models (ARIMAs) [17]. Also we found no evidence of co-designing larger software framework aspects of HPC or using HPC at scale with the very specific Graph Neural Networks (GNNs) used at CERN before CoE RAISE such as ML Particle Flow (MLPF) [18] or JEDI-NETs [19].

Recurrent Neural Networks (RNNs) like Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRUs) have been used with HPC, but are mostly applied to language datasets and not as much to physical time series as within CoE RAISE. Finally, Generative Adversarial Networks (GANs) and its specific variants Wasserstein GANs (WGANs) [20] are also used by others, but not in the direct contact of large-scale HPC software framework designs.

[6]https://coec-project.eu/

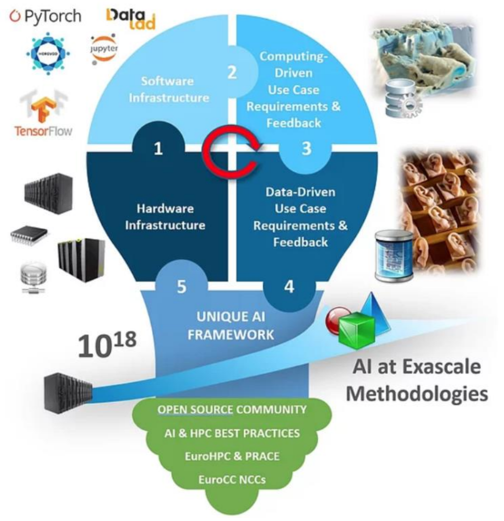

Figure 1. Relevant HPC ecosystem factors influencing the unique AI framework layout design of CoE-RAISE.

III. Practice and Experience with relevant Technologies to Speed-Up Data Science Methods

Based on our profound practice and experience, this section provides identified technologies that are relevant for the AI framework without loosing sight of the larger ecosystem, such as HPC systems and relevant standards. As this paper focuses on the framework, official sources are referenced as much as possible without losing sight of the essential elements of each topic. Figure 1[7] shows the relevant HPC ecosystem factors influencing the unique AI framework layout design of CoE RAISE. As shown in Figure 1, this ecosystem represents an unprecedented Exascale hardware infrastructure (cf. Figure 1, item 1). Based on this solid foundation, a seamlessly usable and versatile software infrastructure (cf. Figure 1, item 2) is critical for accelerating convergence through new AI tool sets that are ready to scale for enormous quantities of data. CoE RAISE is performing application co-design based on a variety of use cases[8]. It considers AI requirements of compute-driven use cases (cf. Figure 1, item 3) using numerical methods based on known physical laws. CoE RAISE also addresses AI requirements of data-driven use cases (cf. Figure 1, item 4) with large measurement device datasets and focuses on AI methods. All these four main key factors and their requirements shape the design of the unique AI framework (cf. Figure 1, item 5) to be usable by many other use case applications outside the CoE RAISE consortium.

A wide variety of AI technologies influence the unique AI framework, while CoE RAISE emphasizes those relevant for Exascale and can leverage HPC system environments. In the following, these technologies are briefly described with a focus on their unique selling propositions for CoE RAISE. A general observation in RAISE in particular and in the AI community is that Python is the preferred language for Machine Learning (ML) and Deep Learning (DL) using AI technologies. It

[7]https://www.coe-raise.eu/ai-exascale
[8]https://www.coe-raise.eu/use-cases

TABLE I. Specifications of the DEEP-EST DAM Prototype

Use Cases	**AE**	**PINN**	**ANNs**		**CNNs**		**NO**	**SMs**			**GNNs**		**RNN**		**GAN**
Details	**CAE**		**ANN**	**RBF-ANN**	**U-Net**	**RES-NET**	**FNO**	**AR**	**ARMA**	**ARIMA**	**MLPF**	**JEDI-NET**	**LSTM**	**GRU**	**WGAN**
AI for turbulent boundary layers	X														
AI for wind farm layout optimization				X				X	X	X					
AI for data-driven models in reacting flows					X										
Smart models for next generation aircraft engine design					X										
AI for wetting hydrodynamics							X								
Event reconstruction and classification at the CERN HL-LHC											X	X			
Seismic imaging with remote sensing for energy applications	X	X				X							X	X	
Detect-free metal additive manufacturing	X		X												X
Sound Engineering	X		X												

also works seamlessly with C and C++ code modules that are relevant in CoE RAISE when intertwining AI methods with traditional simulation science codes often written in these languages. From a license perspective, many of the AI technologies are free and open-source, being community-friendly and guaranteeing improvements in the long run. Also, already existing libraries that often enable solutions for given AI problems are also briefly listed in the next sections.

A. Basic AI frameworks, tools, and libraries

All use cases in CoE RAISE use innovative approaches of DL networks. Therefore, frameworks, tools, and libraries supporting the use cases are of primary interest to RAISE. One of the most popular open-source libraries is TensorFlow[9] (originally developed by Google) that enables the development and training of DL models. More recently, it also comes directly packaged with the intuitive high-level Application Programming Interface (API) called Keras[10] that enables easy model building, rapid prototyping, fast model iteration, and easy debugging. TensorFlow (with Keras) is available on almost all HPC machines in the community and is adopted in many commercial cloud platforms today. An increasing uptake of the PyTorch DL library[11] for Python (originally developed by Facebook) is observed in CoE RAISE. For CoE RAISE, the relevance lies in its strong support for C++, e.g., through C++ interfaces. This library is also customizable to develop and train a wide variety of DL models, and is already widely used in the HPC community. There are many other basic DL frameworks, tools, and libraries available such as Theano[12] (recently renamed to Aesara because Theano development stopped) or the more broadly used MXNet[13]. As both are not used in any CoE RAISE co-design applications, they are not influencing the design of the CoE RAISE AI framework.

[9]https://www.tensorflow.org/
[10]https://keras.io/
[11]https://pytorch.org/
[12]https://github.com/aesara-devs/aesara
[13]https://mxnet.incubator.apache.org/versions/1.8.0/

B. Distributed Deep Learning towards Exascale

One of the key concerns of CoE RAISE is the scaling-up of distributed DL techniques, which is necessary due to CoE RAISE's continuously increasing volume and complexity requirements demanding Exascale resources. In addition, hyper-parameter optimization and AutoML techniques execute the same basic models with different hyper-parameter configurations that add to the demand for computational Exascale resources. A thorough overview of low-level distributed DL training techniques leveraging parallel and scalable methods on HPC systems is given by Tal Ben-Nun et al. in [21]. An open-source library for distributed DL for CoE RAISE is Horovod [22] (originally developed by Uber for TensorFlow), which abstracts the complex TensorFlow functionalities. It thus supports other basic DL libraries mentioned above such as PyTorch or MXNet. Horovod offers quite good usability in the sense that adding/changing just a few lines of code to enable the scale-up of DL models. Horovod is often installed on European HPC systems and is used by many scientific communities to include multi-GPU computing in parallel DL model training. For CoE RAISE, the PyTorch Distributed Data Parallel (DDP) module [23] that is a built-in approach to parallelize the training of DL models across multiple GPUs is also of interest. In other words, PyTorch does not require explicitly written lines of code for scaling-up like Horovod does. This framework is also very popular in our experience, given its built-in feature for scaling-up among many scientific and engineering communities.

IV. Application Co-Design Lessons Learned

This section informs about our application co-design process insights, and the identified concrete use case AI requirements (cf. Table I), and lessons learned when working with the range of technologies identified of being relevant in Section III. The lessons learned that contribute to a unique AI framework design in CoE RAISE are driven by four different factors (see Figure 1): (1) hardware infrastructure, (2) software infrastructure, (3) compute-intensive use cases, and (4) data-intensive use cases. As a whole, they contribute to HPC and AI

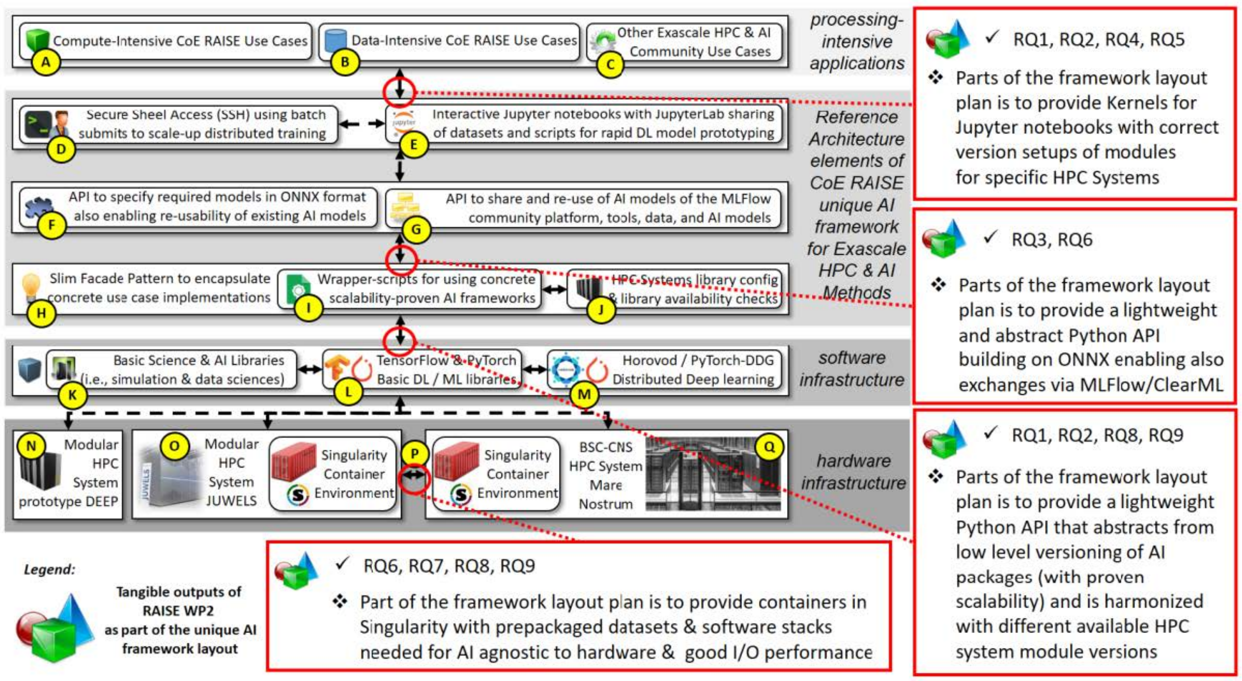
Compute-Intensive CoE RAISE Use Cases
A
Data-Intensive CoE RAISE Use Cases
B
Other Exascale HPC & AI Community Use Cases
C
processing-intensive applications
Secure Sheel Access (SSH) using batch submits to scale-up distributed training
D
Interactive Jupyter notebooks with JupyterLab sharing of datasets and scripts for rapid DL model prototyping
E
API to specify required models in ONNX format also enabling re-usability of existing AI models
F
API to share and re-use of AI models of the MLFlow community platform, tools, data, and AI models
G
Slim Facade Pattern to encapsulate concrete use case implementations
H
Wrapper-scripts for using concrete scalability-proven AI frameworks
I
HPC-Systems library config & library availability checks
J
Reference Architecture elements of CoE RAISE unique AI framework for Exascale HPC & AI Methods
Basic Science & AI Libraries (i.e., simulation & data sciences)
K
TensorFlow & PyTorch Basic DL / ML libraries
L
Horovod / PyTorch-DDG Distributed Deep learning
M
software infrastructure
Modular HPC System prototype DEEP
N
Modular HPC System JUWELS
O
Singularity Container Environment
P
Singularity Container Environment
BSC-CNS HPC System Mare Nostrum
Q
hardware infrastructure
✓ RQ1, RQ2, RQ4, RQ5
❖ Parts of the framework layout plan is to provide Kernels for Jupyter notebooks with correct version setups of modules for specific HPC Systems
✓ RQ3, RQ6
❖ Parts of the framework layout plan is to provide a lightweight and abstract Python API building on ONNX enabling also exchanges via MLFlow/ClearML
✓ RQ1, RQ2, RQ8, RQ9
❖ Parts of the framework layout plan is to provide a lightweight Python API that abstracts from low level versioning of AI packages (with proven scalability) and is harmonized with different available HPC system module versions
Legend:
Tangible outputs of RAISE WP2 as part of the unique AI framework layout
✓ RQ6, RQ7, RQ8, RQ9
❖ Part of the framework layout plan is to provide containers in Singularity with prepackaged datasets & software stacks needed for AI agnostic to hardware & good I/O performance

Figure 2. Lessons learned of application co-design creating a unique AI software framework blueprint towards Exascale.

technical requirements with respect to software and hardware of a unique AI framework, ready for Exascale.

One of the most important goals of CoE RAISE is to support AI towards Exascale using cutting-edge HPC systems and as such building, training, and evaluating ML and DL models is considered most important. The design and discussion sessions in the continuously accessible virtual CoE RAISE Interaction Rooms [24] enabled the identification of an initial set of relevant AI/HPC methods shown in Table I. A key requirement of the CoE RAISE AI framework is thus to support the design, development, and use of AI models of this matrix. In other words, Table I represents the requirements analysis and lessons learned with respect to the model side of the AI framework. Its DL libraries, packages, and tools need to support these models and enable thus a speed-up of data science methods. We reviewed similiar HPC approaches of our identified data science methods already in Section II, and all of them share distinct properties that are important to consider when designing the blueprint for the software framework. As a consequence, they are adopted quite differently in all the CoE RAISE co-design applications shown in Table I.

Apart from the above listed AI/HPC methods and model requirements per use case, there exist further software and hardware infrastructure requirements that have been identified during the Interaction Room process and that are based on profound lessons learned. They are rather seen as general design requirements for the RAISE AI framework as shown in Figure 2. The requirements are numbered as Requirement #X (RQX) to reference them in Section V of this paper, where the overall AI framework design is described, and to explain how CoE RAISE aims to address those requirements.

V. AI Software Framework Layout Design

Figure 2 shows the AI software framework layout that captures our lessons learned over the years and that satisfies CoE RAISE use cases. As a software framework, still taking into account hardware infrastructure constraints, it provides a ‘standard way’ to build and deploy AI applications in RAISE and is a universal, reusable software environment that provides particular functionality as part of a larger software platform, e.g., including specific HPC system module deployments, to facilitate the development of AI use cases in CoE RAISE. Based on the requirements, the software framework includes supporting programs, code libraries, toolsets, and APIs that bring together all the different components to enable development of AI models of Compute-Intensive CoE RAISE Use Cases (cf. Figure 2, item A) and Data-Intensive CoE RAISE Use Cases (cf. Figure 2, item B). One of the goals of this reference architecture is also to enable other Exascale HPC and AI Community Use Cases (cf. Figure 2, item C) such as those driven by other CoEs or future Digital Twins such as Destination Earth[14].

To address the requirements stated in Figure 2, the reference architecture needs to support low-level access such as SSH (RQ5) and high-level access methods such as Jupyter notebooks (RQ4). In this initial blueprint, the core building block to enable low-level access is via SSH protocols using batch scheduler scripts for automation (cf. Figure 2, item D). The requirement analysis of the CoE RAISE use cases revealed that an interactive access via Jupyter notebooks is also required (cf. Figure 2, item E) to enable quick and rapid prototyping

[14]https://digital-strategy.ec.europa.eu/en/policies/destination-earth

of DL algorithms. In this context, it is possible to create SSH sessions out of Jupyter notebooks on HPC systems.

The framework needs to be hardware-agnostic with respect to accelerators (RQ8) and requires a high I/O performance capable of working with large quantities of data (RQ9). To enable fast portability between different DL frameworks and reproducibility (RQ6), the unique AI framework needs to abstract the specification of specific tasks (RQ2) by using the ONNX format (cf. Figure 2, item F). A more comprehensive view on use cases reveals the requirements that platforms like MLFlow (cf. Figure 2, item G) should be supported to share and re-use existing AI models among the broader AI and HPC community (RQ6).

To map the above rather abstract specifications to specific software and hardware infrastructure via the Facade pattern (cf. Figure 2, item H), the unique AI framework layout design employs an abstract wrapper functionality that maps the abstract specifications to concrete software and hardware configurations (RQ1) of available HPC systems (cf. Figure 2, item I), encapsulating users from the need to know about low-level version details. In this context, it is important to check what versions are available in what modules on the specific HPC systems (cf. Figure 2, item J) as CoE RAISE's use cases considered this a major obstacle for using AI technologies on HPC systems today.

The reference architecture in Figure 2 includes specific versions of packages within its software infrastructure provided by HPC centers. Those software packages are basic science libraries (cf. Figure 2, item K) as mentioned above. The RAISE reference architecture also leverages specific harmonized versions of the DL packages TensorFlow and PyTorch (cf. Figure 2, item L), as well as PyTorch-DDP and Horovod for scaling towards Exascale (cf. Figure 2, item M). At the time of writing, the complementary benchmark activities of CoE RAISE investigate scaling and its limits when using these packages particularly with respect to their scalability towards Exascale.

The reference architecture in Figure 2 also includes a hardware infrastructure that deploys the software infrastructure above and that is accessible to CoE RAISE use case members. Among the HPC systems available to RAISE are the DEEP prototype (cf. Figure 2, item N), the JUWELS system at JSC (cf. Figure 2, item O), the MareNostrum system at BSC-CNS (cf. Figure 2, item Q), and others. To enable portability, users of the AI framework for RAISE based on this reference architecture are able to use specially prepared containers with use-case specific software stacks prepared based on Singularity (cf. Figure 2, item P).

We above listed all major components of the framework, while it still lacks to address novel emerging types of HPC-related computing techniques such as Neuromorphic computing or Quantum Computing (QC). While we report in the next section briefly about our experience in QC, we observe in CoE RAISE and other applications that more research is needed still to fully integrate those into our software framework design today.

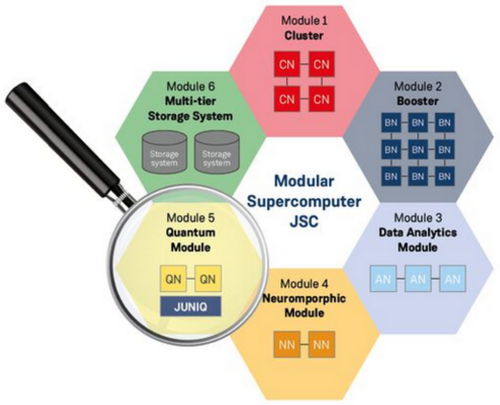

Figure 3. Integration of QC in the MSA connecting to our software framework on network level.

VI. Towards integrating Quantum Computing

QC includes a broad range of approaches (e.g., gate-based approaches, or quantum annealing, etc.), but paves the way for an enhancement of computing capabilities beyond traditional HPC resources addressed above. Being a very novel research topic in the context of data sciences, QML aims at developing ML models specifically designed for quantum computers. The interest towards QML has increased, also due to the development of the first prototypes of quantum computers.

Our experience is based on using prototypes of the D-Wave 2000Q Quantum Annealer (QA) as we described in [9] using traditional Support Vector Machines (SVMs). More recent experience [7, 8] that we also discuss in this paper is based on using D-Wave's next-generation quantum processor 5000+ qubit Advantage that is now part of our Juelich UNified Infrastructure for QC (JUNIQ)[15]. Our experience, however, in the light of this paper reveals, that the software framework environment and operation of the systems is partly different. Examples include libraries such as the Ocean Software Development Kit (SDK)[16] from D-Wave for programming QAs with Python that is basically similar like using other Python libraries (e.g., TensorFlow, PyTorch, NumPy, etc.) and could be integrated into our framework.

But the way of how we integrate quantum annealers into larger HPC systems and its software ecosystem is rather similar like using accelerators with Graphical Processing Units (GPUs) from NVidia with data science methods. In other words, to connect seamlessly from cutting-edge HPC systems like JUWELS that adopt the Modular Supercomputer Architecture (MSA) [25] shown in Figure 3, we use Quantum Modules with Quantum Nodes (QN) for very specific problems only (e.g., solving optimization problems in ML algorithms) connecting rather on the networking level. While the integration into our framework can be done, we also observe limits (e.g., only small number of datasets to be used by QA currently compared to 'big data') that is our rationale to not include it already in the software framework today.

[15]https://www.fz-juelich.de/ias/jsc/EN/Expertise/JUNIQ/_node.html
[16]https://docs.ocean.dwavesys.com/en/stable/

VII. Conclusion

We can conclude that our framework provides kernels for Jupyter notebooks with correct version setups of modules for specific HPC systems addressing RQ1, RQ2, RQ4, and RQ5. Furthermore the approach os using a lightweight and abstract Python API building on ONNX to enable easy exchange with MLFlow/ClearML addressing RQ3 and RQ6 seams feasible too when talking with end user communities. That also includes abstracting from low level versioning of AI packages (with proven scalability) on HPC resources and is harmonized with different available HPC system module versions addressing RQ1, RQ2, RQ8, and RQ9. Finally, a promising approach is to provide containers in Singularity with prepackaged datasets and required software stacks needed for AI models addressing RQ6 and RQ7. This environment needs to be able to work with large-scale datasets and having good I/O performance addressing RQ9. Additionally, those environments are underlying hardware-agnostic addressing RQ8. Finally, future work will integrate QC and QA more seamlessly to use quantum modules as 'unique accelerators'.

Acknowledgements

This work was performed in the Center of Excellence (CoE) Research on AI- and Simulation-Based Engineering at Exascale (RAISE) receiving funding from EU's Horizon 2020 Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733. Icelandic HPC National Competence Center is funded by the EuroCC project that has received funding from the EU HPC Joint Undertaking (JU) under grant agreement No 951732.

References

[1] S. Kesselheim, M. Riedel, et al., "Juwels booster–a supercomputer for large-scale ai research," in International Conference on High Performance Computing. Springer, 2021, pp. 453–468.
[2] R. Sedona, C. Paris, G. Cavallaro, L. Bruzzone, and M. Riedel, "A high-performance multispectral adaptation gan for harmonizing dense time series of landsat-8 and sentinel-2 images," IEEE journal of selected topics in applied earth observations and remote sensing, vol. 14, pp. 10134–10146, 2021.
[3] R. Sedona, G. Cavallaro, J. Jitsev, A. Strube, M. Riedel, and J. A. Benediktsson, "Remote sensing big data classification with high performance distributed deep learning," Remote Sensing, vol. 11, no. 24, pp. 3056, 2019.
[4] R. Sedona, G. Cavallaro, M. Riedel, and M. Book, "Enhancing large batch size training of deep models for remote sensing applications," in IEEE International Geoscience & Remote Sensing Symposium IGARSS, 2021, pp. 1583–1586.
[5] M. Riedel, R. Sedona, C. Barakat, P. Einarsson, R. Hassanian, G. Cavallaro, M. Book, H. Neukirchen, and A. Lintermann, "Practice and experience in using parallel and scalable machine learning with heterogenous modular supercomputing architectures," in IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2021, pp. 76–85.
[6] R. Sedona, G. Cavallaro, J. Jitsev, A. Strube, M. Riedel, and M. Book, "Scaling up a multispectral resnet-50 to 128 gpus," in IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium. IEEE, 2020, pp. 1058–1061.
[7] A. Delilbasic, G. Cavallaro, M. Willsch, F. Melgani, M. Riedel, and K. Michielsen, "Quantum support vector machine algorithms for remote sensing data classification," in 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS. IEEE, 2021, pp. 2608–2611.
[8] M. Riedel, G. Cavallaro, and J. A. Benediktsson, "Practice and experience in using parallel and scalable machine learning in remote sensing from hpc over cloud to quantum computing," in IEEE International Geoscience and Remote Sensing Symposium IGARSS. IEEE, 2021, pp. 1571–1574.
[9] G. Cavallaro, D. Willsch, M. Willsch, K. Michielsen, and M. Riedel, "Approaching remote sensing image classification with ensembles of support vector machines on the d-wave quantum annealer," in IGARSS IEEE International Geoscience and Remote Sensing Symposium, 2020, pp. 1973–1976.
[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," science, vol. 313, no. 5786, pp. 504–507, 2006.
[11] A. Glaws, R. King, and M. Sprague, "Deep learning for in situ data compression of large turbulent flow simulations," Physical Review Fluids, vol. 5, no. 11, pp. 114602, 2020.
[12] T. Kadeethum, T. M. Jørgensen, and H. M. Nick, "Physics-informed neural networks for solving nonlinear diffusivity and biot's equations," PloS one, vol. 15, no. 5, pp. e0232683, 2020.
[13] C. J. Lapeyre, A. Misdariis, et al., "Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates," Combustion and Flame, vol. 203, pp. 255–264, 2019.
[14] Z. Jiang, W. Gao, F. Tang, X. Xiong, L. Wang, C. Lan, et al., "Hpc ai500: Representative, repeatable and simple hpc ai benchmarking," arXiv preprint arXiv:2102.12848, 2021.
[15] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," arXiv preprint arXiv:2010.08895, 2020.
[16] W. Xue, H. Fu, et al., "Editorial for the special issue on large-scale ai in classical hpc environment and ai for science," 2021.
[17] Z. Zhang and J. Moore, "Chapter 8-autoregressive moving average models," Mathematical and Physical Fundamentals of Climate Change, pp. 239–290, 2015.
[18] J. Pata, J. Duarte, J.-R. Vlimant, M. Pierini, and M. Spiropulu, "Mlpf: efficient machine-learned particle-flow reconstruction using graph neural networks," The European Physical Journal C, vol. 81, no. 5, pp. 1–14, 2021.
[19] E. A. Moreno, O. Cerri, J. M. Duarte, H. B. Newman, et al., "Jedi-net: a jet identification algorithm based on interaction networks," The European Physical Journal C, vol. 80, no. 1, pp. 1–15, 2020.
[20] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in International conference on machine learning. PMLR, 2017, pp. 214–223.
[21] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," ACM Computing Surveys (CSUR), vol. 52, no. 4, pp. 1–43, 2019.
[22] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," arXiv preprint arXiv:1802.05799, 2018.
[23] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, et al., "Pytorch distributed: Experiences on accelerating data parallel training," arXiv preprint arXiv:2006.15704, 2020.
[24] M. Book, M. Riedel, H. Neukirchen, and M. Götz, "Facilitating collaboration in high-performance computing projects with an interaction room," in Proceedings of the 4th ACM SIGPLAN International Workshop on Software Engineering for Parallel Systems, 2017, pp. 46–47.
[25] E. Suarez, N. Eickert, and T. Lippert, "Modular Supercomputing architecture: from idea to production," in Contemporary High Performance Computing: From Petascale toward Exascale, vol. 3, pp. 223–251. FL, USA, 1 edition, 2019.