



## D5.3: Final report on the VECMA infrastructure

<b>Due Date</b>	14 December 2021
<b>Delivery</b>	14 December 2021
<b>Submission of updated version</b>	N/A
<b>Lead Partner</b>	PSNC
<b>Dissemination Level</b>	Public
<b>Status</b>	Final
<b>Approved</b>	Executive Board
<b>Version</b>	V1.2



**DOCUMENT INFO**

<b>Date and version number</b>	<b>Author</b>	<b>Comments</b>
06.09.2021 v0.1	Bartosz Bosak, Tomasz Piontek	Skeleton for the document
19.09.2021 v0.2	Bartosz Bosak	Introduction for the Vecma Infrastructure section
25.09.2021 v0.3	Bartosz Bosak	Summary on the Infrastructure Status (introduction, diagram and table)
30.09.2021 v0.4	Bartosz Bosak, Tomasz Piontek	Added Altair supercomputer description
05.10.2021 v0.5	Bartosz Bosak, Piotr Kopta	Table with developed QCG-PilotJob functionality
14.10.2021 v0.6	Bartosz Bosak, Tomasz Kuczyński	VECMA Jupyter Notebook Platform
21.10.2021 v0.7	Bartosz Bosak, Michał Kulczewski, David Coster, Hamid Arabnejad, Derek Groen	Applications and Infrastructure
28.10.2021 v0.8	Bartosz Bosak, Piotr Kopta	QCG-PilotJob Common Scheduling Service
29.10.2021 v0.9	Bartosz Bosak, Michał Kulczewski	Dissemination activities
02.11.2021 v0.11	Bartosz Bosak, Derek Groen	Collaboration with external projects
03.11.2021 v0.12	Piotr Kopta	qcg-pm-report tool's usage reference
05.11.2021 v0.13	Nicholas Laver	Language corrections
05.11.2021 v0.14	Tomasz Piontek	General checking and adjustments
25.11.2021 v0.15	Bartosz Bosak, Piotr Kopta, Marco Verdicchio, Werner Muller, Erwan Raffin, Paul Karlshoefer	Performance evaluation
28.11.2021 v0.15b	Derek Groen, Erwan Raffin	Internal review
30.11.2021 v0.16	Bartosz Bosak, Piotr Kopta	Addressed the reviewers' comments
06.12.2021 v1.0	Nicholas Laver, Bartosz Bosak	Fine-tuning
13.12.2021 v1.1	Bartosz Bosak, Tomasz Piontek	Final editing after PI's review
14.12.2021 v1.2	Peter Coveney	Approval

## CONTRIBUTORS

- Bartosz Bosak, PSNC – main author
- Tomasz Piontek, PSNC - WP5 leader, author, supervision
- Piotr Kopta, PSNC - author
- Tomasz Kuczyński, PSNC - author
- Michał Kulczewski, PSNC - author
- David Coster, MPG - author
- Hamid Arabnejad, Brunel (UBRU) - author
- Derek Groen, Brunel (UBRU) - author
- Erwan Raffin, Bull - author
- Paul Karlshoefer, Bull - author
- Lourens Veen, eScience Center - author
- Marco Verdicchio, SURF - author
- Werner Muller, UCL – author
- Nicholas Laver, UCL – author, language corrections

## **Disclaimer**

This document's contents are not intended to replace consultation of any applicable legal sources or the necessary advice of a legal expert, where appropriate. All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user, therefore, uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability in respect to this document, which merely represents the authors' view.

## TABLE OF CONTENTS

1	Executive summary	8
2	VECMA Infrastructure	9
2.1	Final status of VECMA Infrastructure	10
2.1.1	Main Achievements of Work Package 5	10
2.1.2	New resources contributed to VECMA from M19	13
2.2	Pilot Job Mechanism	14
2.2.1	Developed functionality	14
	Scheduling Queue Service	16
	Performance Tracking and Analysis Suite	17
2.2.2	Performance evaluation	19
	Initial tests of EQI	19
	Synthetic Performance Tests on Altair	20
	LAMMPS-based performance tests	22
	○ ARCHER2	22
	○ Cartesius	23
	○ SuperMUC-NG	24
	Summary from the performance tests	25
2.2.3	Integration with User-level tools	26
	EasyVVUQ	26
	FabSim3	27
	MUSCLE3	27
2.3	VECMA Jupyter Notebook Platform	27
2.4	Applications and Infrastructure	28
2.5	Dissemination activities related to Work Package 5	29
2.6	Collaboration with external projects	30
3	Conclusions	31
4	References	32
•	Annex I – QCG-PilotJob Documentation	33
•	Annex II – VECMA Notebook Platform	34

## LIST OF FIGURES

Figure 1. Execution layer of VECMA architecture	9
Figure 2. Work Package 5 timeline with major accomplishments highlighted	10
Figure 3. High-level picture on the architecture of the QCG-PilotJob system with the QCG-PilotJob Scheduling Queue service employed	17
Figure 4. Two ways of measuring QCG-PilotJob performance. METHOD 1 is based on a comparison with a reference execution, METHOD 2 on measuring the resources idleness. The second method has been assumed as more realistic and therefore selected for the implementation	18
Figure 5. Performance tests of EQI on SuperMUC-NG	20
Figure 6. The results of QCG-PilotJob performance tests from Altair when a single instance of the service was used	21
Figure 7. The results of QCG-PilotJob performance tests from Altair in a partitioned execution model	21
Figure 8. The results of QCG-PilotJob performance tests on ARCHER2	23
Figure 9. The results of QCG-PilotJob performance tests on Cartesius	24
Figure 10. The results of QCG-PilotJob performance tests on SuperMUC-NG	25

## LIST OF TABLES

Table 1. A list of main achievements of Work Package 5	11
Table 2. Main developments in QCG-PilotJob during VECMA	14
Table 3. Overview on applications in the context of infrastructure usage	28
Table 4. The list of dissemination activities of Work Package 5	29
Table 5. The list of projects that expressed interest in usage of VECMA Work Package 5 outcomes for their needs	30

**ACRONYMS AND DEFINITIONS**

Acronym	Definition
API	Application Programming Interface
EQI	EasyVVUQ & QCG-PilotJob Integration API
HPC	High Performance Computing
HTC	High Throughput Computing
MPI	Message Passing Interface
QCG	Quality in Cloud and Grid
SA	Sensitivity Analysis
VECMAtk	VECMA Toolkit
VVUQ	Validation, Verification and Uncertainty Quantification

# 1 Executive summary

This document is a final report on the VECMA Infrastructure, established by Work Package 5. This work includes the collection of requirements and design of the overall architecture of the VECMA system (Task 5.1), the configuration and management of the project testbed, built upon several large-scale computing resources (Task 5.2), the adaptation of existing middleware technologies (Task 5.4) and the development of new user-level tools (Task 5.5) in accordance with the needs of VECMA users. This deliverable is a follow-up to *D5.1 Architecture of the VECMA system* and particularly *D5.2 First report on the VECMA infrastructure*. As such, it extends the information contained within those documents.

The focus of this deliverable is twofold: firstly, it outlines the main achievements of WP5 across the whole project lifetime and on all logical levels of the VECMA Infrastructure. Secondly, it provides more detailed information about the activities undertaken after the release of D5.2, between months 19 and 42 of the project. This period resulted in crystallization and slight amendment of the VECMA needs which subsequently influenced the priorities of Work Package 5. For example, as the project partners expected there to be particular importance attaching to the pilot job solutions for the VECMA VVUQ scenarios, considerable effort has been spent to enhance quality, robustness, performance and scalability of the QCG-PilotJob tool, as well as to make this tool more user-friendly and easily integrated with other VECMA toolkit components. Most notably, we made the following major improvements to QCG-PilotJob:

- We added a restoring mechanism which allows a workflow of tasks to be restarted from the point it was stopped (e.g. due to a failure or exceeding the wall-time limit).
- We performed a significant optimisation of the execution of iterative tasks.
- We streamlined the main QCG-PilotJob API and developed an alternative, simplified API based on conceptions of Python *Executor* and *Promise* classes (used for integration with EasyVVUQ).
- We prepared new documentation based on Sphinx and made it available on the readthedocs.io portal
- We developed Performance Tracking and Analysis Suite to provide an easy and portable way to monitor performance of the tool and enable users to discover bottlenecks in their scenarios
- We established a common Scheduling Queue service that allows users to dynamically integrate resources from many allocations

QCG-PilotJob has also been intensively tested in regard to portability and scalability across multiple high-end computing resources. These HPC facilities have been provided by the VECMA partners, but have also been offered through collaboration with external projects, such as CompBioMed2.

In addition, Work Package 5 has undertaken multiple activities that were important for the promotion of the project. Beside common dissemination activities such as preparation of presentations, creation of posters or participation in conferences and training events, efforts have also been expended on more specific tasks, i.e. configuration and deployment of the VECMA Notebook Platform (<https://jupyter.vecma.psnr.pl/>) on the PSNC's cloud infrastructure and making it available to all people within the VECMA consortium and partners interested in using the VECMA toolkit.

## 2 VECMA Infrastructure

This deliverable describes the final status of the VECMA infrastructure developed within Work Package 5 (WP5) of the VECMA project. This document summarises all achievements of WP5, but focuses mainly on the work realised during the period between Months 19 and 42 (formerly Month 36) of the project and in a natural way extends information provided in the analogous deliverable published in Month 18 (*D5.2: First report on the VECMA infrastructure*) and is based on the general architectural from the deliverable published in Month 9 (*D5.1: The Architecture of the VECMA system*).

Stated in previous deliverables, the term "*infrastructure*" within VECMA is exchanged with the term "*execution layer*" and should be understood as hardware and software systems offered to the VECMA project to enable and facilitate execution, as well as, verification, validation, and uncertainty quantification (VVUQ) of multiscale simulations and applications. This means that the VECMA infrastructure is a relatively broad concept and covers integrated automation and execution user-level tools, infrastructure services and distributed HPC/HTC machines. This is depicted in Figure 1 (first introduced in D5.1) which outlines the VECMA execution layer.

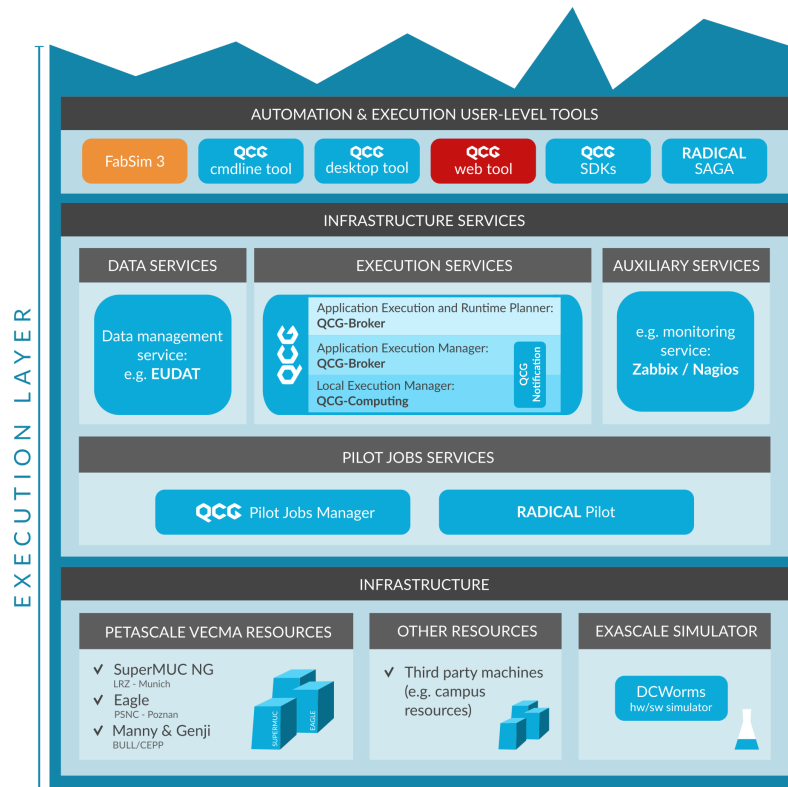


Figure 1. Execution layer of VECMA architecture

With this understanding of the infrastructure term, the following sections provide information about tasks realised and accomplished under the umbrella of Work Package 5.

In order to avoid duplication, whenever it is not essential, this document doesn't repeat messages already included in the previous deliverables of WP5. Thus, only the next section can be recognised as a classical summary of the work done in WP5, while later sections concentrate mostly on recent achievements and those aspects that should be explained in more detail.

## 2.1 Final status of VECMA Infrastructure

The final status of the VECMA infrastructure at the end of the project, is a result of the realisation of four tasks, namely:

- *Task 5.1 Requirements, analysis and design of the VECMA architecture;*
- *Task 5.2 Creation and management of the project testbed and requirements to structure the work across different teams in VECMA;*
- *Task 5.4 Enabling middleware technologies for efficient execution of the VECMA simulations;*
- *Task 5.5 Development of intuitive user-level tools to facilitate the usage of the VECMA e-Infrastructure environment.*

Please note that, due to the common agreement between partners being a result of the performed requirements analysis "*Task 5.3 Application of simulation frameworks to simulate execution of VECMA codes on forthcoming, exascale computational resources*" - which was initially planned for realisation in the second part of the project - has been cancelled, and the efforts initially assigned to it have been consequently moved to other tasks, namely *Task 5.4* and *Task 5.5*. [5, 4]

Please also note that, by common agreement in the consortium, the project has been extended by 6 months from the initial period of 36 months. Consequently, we've decided to extend the timeframe of *Task 5.2*, *Task 5.4* and *Task 5.5* accordingly, and finish all these tasks at the newly defined end of the project, in month 42. This decision allowed not only the spread of development work in time, but more importantly, to steer that work based on emerging, and more well defined, needs of VECMA users. Not without significance for the WP5, there was also the possibility to schedule dissemination activities more flexibly.

### 2.1.1 Main Achievements of Work Package 5

The main achievements of WP5 are presented visually in Figure 2 and described in more detail in Table 1. Let's firstly concentrate on the first representation.



Figure 2. Work Package 5 timeline with major accomplishments highlighted

Figure 2 provides an overview of the work done within the work package, broken down into individual tasks according to the predefined schedule. Each of the tasks has its own lane where the main outcomes of the task are graphically depicted. Although it is not strict, the order of elements on the timelines provides an estimation of when resource and software items were made available to the consortium for the first time. Arrows on the diagram outline the main relations between tasks within WP5 and the connections to other work packages. As such, it can be noted that there was a close cooperation and merge between Task 5.4 and Task 5.3 conducted by WP5, and associated tasks in WP3 and WP4. This joint work has allowed the VECMA project to develop the VECMA Toolkit [7] according to the scientific expectations of researchers on one hand, and to efficiently leverage capabilities of the computing resources on the other.

It can also be noted that QCG-PilotJob, Jupyter and QCG-Monitoring icons are included in both lanes, for T5.4 and T5.5 tasks, indicating that there was joint development of these software components in both tasks.

We next move to the stricter overview of the main achievements of the WP5, which is presented in Table 1. Each row in the table addresses a specific item that can be recognised as an achievement. For each of these items, the following data is provided:

- Item: the name of the item,
- Task: a task or tasks that have been directly involved in the development of the item,
- Time: the period in which there was work done on the item, or the item was in an "active" state,
- Results: descriptive information about the result of the work,
- Comments.

Table 1. A list of main achievements of Work Package 5

Item	Task	Time	Results	Comments
------	------	------	---------	----------

Requirements analysis	T1	M1-M9	The set of requirements for architecture design as an input for the architecture of VECMA system (D5.1).	Input from WP3 and WP4
Architecture	T1	M1-M9	Three-layered architecture of the VECMA system, consisting of: Applications, VVUQ and Infrastructure layers (D5.1).	
Testbed: Instructions and procedures	T2	M3-M18	A set of instructions and procedures, in a form of dedicated guidance for different VECMA stakeholders, available on a wiki page (D5.2).	
Testbed: Access to Eagle	T2	M3-M30	The access to Eagle supercomputer hosted at PSNC was possible for all partners starting from M3 to M30 (D5.2), then Eagle was replaced by Altair. The overall budget provided to VECMA was 9M Core Hours.	around 75% of the budget has been consumed
Testbed: Access to SuperMUC-NG	T2	M3-M42	The access to the SuperMUC-NG supercomputer hosted at LRZ is possible for all partners starting from M18 (D5.2). The annual budget provided to VECMA partners on SuperMUC-NG oscillated around 100K Core Hours.	
Testbed: Access to Altair	T2	M31-M42	PSNC's Altair supercomputer launched at the beginning of 2021 is a successor of Eagle. It is available on the same terms as Eagle to all VECMA partners. The available budget for VECMA for 2021 is equal to 5M Core Hours.	
Testbed: Access to Many and Genji	T2	M3-M42	The allocation on the Bull's resources has been given to the Bull partner for the works on scalability and efficiency of VECMA software (D5.2).	
Testbed: Third-party resources	T2	M3-M42	A set of additional resources has been given to VECMA partners on an individual basis. This set, among others includes: ARCHER, Cartesius, Marconi, Cobra, Draco, Hazel Hen, Prometheus, Tryton (D5.2), ARCHER2.	The access to ARCHER2 and Cartesius was possible thanks to the collaboration with CompBioMed project
QCG middleware	T4	M9-M42	QCG middleware services have been deployed on Eagle, Altair, and several resources belonging to PLGrid Infrastructure and available to PSNC (D5.2).	Due to procedural restrictions, deployment of QCG services on SuperMUC-NG has been abandoned.
QCG-PilotJob	T4, T5	M9-M42	QCG-PilotJob has been variously extended and improved. The updates range from development of new functionality, through improved portability and scalability to preparation of extensive documentation and tutorials.	<a href="https://qcg-pilotjob.readthedocs.io">https://qcg-pilotjob.readthedocs.io</a>
VECMA	T4,	M15-	A self-hosted JupyterLab platform has been	<a href="https://jupyter.vecma.">https://jupyter.vecma.</a>

Jupyter Notebook Platform	T5	M42	deployed on PSNC's Cloud infrastructure and adjusted to VECMA needs. It has been made available to both VECMA partners and external users.	<a href="https://psnc.pl">psnc.pl</a>
QCG-Monitoring	T4, T5	M15-M42	An instance of QCG-Monitoring service has been integrated with the QCG middleware services for VECMA and offered to users on a pre-production basis (D5.2). A dedicated monitoring scheme for tracking progress of QCG-PilotJob has been already constructed, while other schemes, especially for individual applications, can be added to the service depending on demands.	
QCG-Client	T5	M9-M42	QCG-Client has been offered to users along with the QCG middleware. A remote instance of the tool has been deployed on a dedicated machine being in administration of PSNC. During the course of the project, QCG-Client was also containerised and employed as a backend for FabSim3 (D5.2).	
FabSim3	T5	M9-M42	Under WP5 FabSim3 has been integrated with QCG middleware services (to this end, a containerised version of QCG-Client has been used [4]) and QCG-PilotJob. This integration has enabled several FabSim user communities (e.g. in migration, climate and pandemic modelling) to routinely use QCG-PilotJob for their work.	FabSim3 has been developed mainly in Work Package 3 as a part of VECMA toolkit.
EasyVVUQ-QCGPJ (EQI)	T5	M9-M42	EasyVVUQ-QCGPJ (EQI) is a lightweight library that integrates early versions of EasyVVUQ with QCG-PilotJob. EQI is not compatible with EasyVVUQ versions starting from 1.0.	EasyVVUQ from version 1.0 provides a built-in QCG-PilotJob executor that replaces EQI
QCG-Now	T5	M15-M42	QCG-Now has been offered to VECMA end-users as an intuitive GUI for submission of VVUQ tasks for execution on QCG-managed resources. To this end a dedicated QCG-Now domain for VECMA has been created and the tool has been integrated with VECMA middleware services, including QCG-Monitoring (D5.2).	

### 2.1.2 New resources contributed to VECMA from M19

The extensive information about the resources provided to VECMA in the first 18 months of the project was presented in D5.2 [4]. Here we extend this list with a description of new and largely modified resources that are currently available to VECMA project participants.

#### Altair

The PSNC Altair supercomputer [23], which was commissioned in January/February 2021, is an extension of the Eagle supercomputer. Thanks to its 1320 nodes equipped with two Intel Xeon Platinum 8268 processors each, it reaches a total performance of 5.9 Pflop/s. From the perspective of users, Altair is a separate partition of Eagle, so is available in the same manner as the old resource. All VECMA participants have benefited from the 5M Core Hours budget on Altair reserved for 2021.

**Marconi**

The Marconi supercomputer at CINECA is available for the MPG partner, where a grant of 4.6M Core Hours was allocated to the VECMA project in 2021. The resource consists of traditional Skylake processors and recently commissioned Nvidia V100 accelerators. They offer 8 Pflop/s and 21 Pflop/s of computing power respectively.

**Raven**

Raven is a new HPC system installed in Garching and is available for the MPG partner. It is a heterogeneous machine consisting of traditional nodes equipped with Xeon IceLake-SP processors, which offer a power of 8.8 Pflop/s, and nodes with Nvidia V100 accelerators, which provide power of 16 Pflop/s.

**ARCHER2**

ARCHER2 is the latest UK national supercomputer built using nodes equipped with dual AMD EPYC Zen2 (Rome) 64 core processors at 2.2GHz. As we submit this report, the full 23-cabinet 750,000 core machine is now entering production, and should reach a peak performance of around 28 Pflops/s. The VECMA project was able to benefit from the initially installed 4-cabinet ARCHER2 system available since Q2 2021, consisting of 1024 nodes, thanks to the collaboration with CompBioMed2 [19].

**2.2 Pilot Job Mechanism**

The standard mechanisms available on large-scale HPC computing resources are geared towards execution of computational tasks which are large and relatively long, and submitted occasionally. However, in the context of VVUQ, there is an inherited need to run numerous, relatively short and small simulations required for reliability of results. In the VECMA project, to support such scenarios in an efficient and consistent manner, we considered employment of RADICAL-Pilot and QCG-PilotJob. After the analysis of VECMA requirements and the proffered functionality of both tools, we found both solutions suitable, but ultimately we decided on broader usage of QCG-PilotJob [8]. This decision was motivated not only by purely technical aspects, such as simplicity of deployment or easiness of integration with other tools, but also because the author of QCG-PilotJob, i.e. PSNC, as a VECMA partner, could offer the required assistance throughout the VECMA project.

QCG-PilotJob overcomes restrictions present on HPC resources by provisioning a second-level scheduling mechanism, which is available from an intuitive Python API, and can be easily employed to build workflows consisting of numerous tasks. In order to provide sufficient level of support to VECMA application scenarios, the tool has been extensively tested, then extended and optimised during the timeframe of the project. It has also been integrated with several VECMAtk components, as well as, intensively tested on large-scale computing resources, which we report in the late part of this deliverable. Finally, it has been successfully employed for executions of complex application use-cases on a regular basis.

**2.2.1 Developed functionality**

Over the course of VECMA, guided by both the generic circumstances of implemented VVUQ scenarios and requirements of individual applications, the functionality of QCG-PilotJob has been extended. An overview of the main developments is presented in Table 2. Please note that this table [D5.3 Final report on the VECMA infrastructure]

outlines only the major new features, while there have been multiple other developments aimed at improving flexibility, scalability, efficiency as well as portability of the tool.

Table 2. Main developments in QCG-PilotJob during VECMA

Feature	Release	Description	Comments
LocalManager	0.4.1 (M12)	Enables starting QCG-PilotJob Manager directly from a python code	
Virtualized/Test mode of execution	0.4.1 (M12)	Possibility to define virtual resources for QCG-PilotJob Manager	The functionality allows testing a scenario on a local computer before it goes to an HPC machine.
Node Launcher	0.5 (M15)	A dedicated lightweight service for more efficient starting tasks on nodes	Current implementation supports single-core tasks; multi-core tasks are started with <i>srn</i> .
Reservation of a CPU core for QCG-PilotJob Manager	0.5 (M15)	In order to separate the management of tasks from their computation, a dedicated CPU core can be reserved exclusively for QCG-PilotJobManager.	This capability is particularly useful in case of demanding workflows, where scheduling of tasks may be CPU intensive.
Tasks CPU binding	0.5 (M15)	Binding of tasks to CPU's cores allows to ensure correct assignment of tasks to resources.	Without this feature many tasks could run on the same cores, resulting in inefficiencies.
Automatic binding to free TCP ports	0.5 (M15)	Releases QCG-PilotJob from running on a predefined TCP port and allows it to run on the free one.	
Passing modules and virtual environment to tasks	0.6 (M18)	Information about modules and python's virtual environments can be passed from the parent script to tasks.	
API refinement	0.8 (M24)	The API has been largely modified in order to make it more consistent and simpler to use.	Among others change of the main module name from <i>qcg.appscheduler</i> to <i>qcg.pilotjob</i> .
New form of documentation based on Sphinx	0.8 (M24)	The tool's documentation has been transferred to *.rst files, refined, extended and then placed on the readthedocs.io portal.	Documentation link: <a href="https://qcg-pilotjob.readthedocs.io">qcg-pilotjob.readthedocs.io</a> See also: <i>Annex I – QCG-PilotJob Documentation</i>
Extended support for iterative jobs	0.8 (M24)	Refined configuration of iterative tasks and provided a new optimised mechanism of launching such tasks	
Execution models for MPI tasks	0.9 (M27)	Developed new execution models designed to launch MPI applications with different implementations of MPI and Slurm configurations	New models: <i>intelmpi</i> , <i>openmpi</i> , <i>srnmpi</i> .
Support for hyper-threading	0.9 (M27)	Automatic discovering of hyper-threading configurations and binding of more than a single CPU to	

		process	
Restoring mechanism	0.10 (M30)	Support for restoring execution of workflow only for the tasks that have not been executed, e.g. due to the exceeding walltime limit.	Provides means of fault tolerance: not completed tasks can be rerun, without losing results of other tasks.
Collection of metrics from executions	0.11 (M33)	Support for gathering basic information about tasks executions.	This is the first module of the Performance Tracking and Analysis Suite.
Performance Tracking and Analysis Suite	0.11.1	A wrapper has been provided to automatically collect detailed timings from jobs and the <i>qcg-pm-report</i> tool to generate various reports.	The suite has been employed to ensure consistency of performance tests of the toolkit, which were executed on different resources and by different people.
Executor-like interface	0.12.1 (M36)	Developed an alternative API that in some aspects mimics <i>concurrent.futures</i> API provided with Python.	The API is suitable for integration with tools already using Python Executor and Future concepts. In VECMA allowed to employ QCG-PilotJob as an executor engine for EasyVVUQ and thus improve its scalability
Breakdown of the tool into components	0.12.1 (M36)	QCG-PilotJob has been split into functionally-diversified components, currently: <i>core</i> , <i>cmds</i> and <i>executor-api</i> .	Only the <i>core</i> component is essential from the users' perspective. Other components are auxiliary.
Enumeration values in iterative tasks	0.12.3 (M39)	Improved functionality of iterative tasks: <i>iteration</i> element can take a list of enumeration values as an input	
Minimisation of the <i>srun</i> command usage	0.13.0 (M42*)	The basic mechanism that allowed to start QCG-PilotJob agents only with <i>srun</i> has been generalised and allows to select different mechanisms, e.g. <i>ssh</i>	The aim is to reduce the number of uses of the <i>srun</i> command, which for demanding workflows can affect the stability of SLURM. Motivated by the tests on SuperMUC-ng
Pub/Sub task status queries	0.13.0 (M42*)	Provided an asynchronous notification mechanism on a level of QCG-PilotJob manager so the client (via API) can trace the task status changes without periodic querying the service	
Scheduling Queue service	0.14.0 (M42*)	The initial version of the Scheduling Queue service that allows to instantiate QCG-PilotJob across many allocations.	

\* The item not yet released

## Scheduling Queue Service

The realisation of the Pilot Job mechanism on a level of single allocation can be seen as a main building block to create a more sophisticated solution that allows resources from multiple allocations, even from multiple physical computing machines, to be combined and served as a large virtual

resource. In order to realise this conception, during the final year of the project many efforts have been spent to develop the QCG-PilotJob Scheduling Queue service, which can act as a common source of information for the execution of QCG-PilotJob tasks across many allocations (the baseline architecture of this conception is presented in Figure 3). Consequently, an initial version of the service has been provided and tested with the selected usage scenarios. At the time of writing, the process of fine-tuning this service is still ongoing, and we are expecting to perform more tests in the near future. Once ready, the common Scheduling Queue service can be used to integrate many computing resources together, as well as, to automatically and dynamically extend (or limit) a set of available resources depending on current needs.

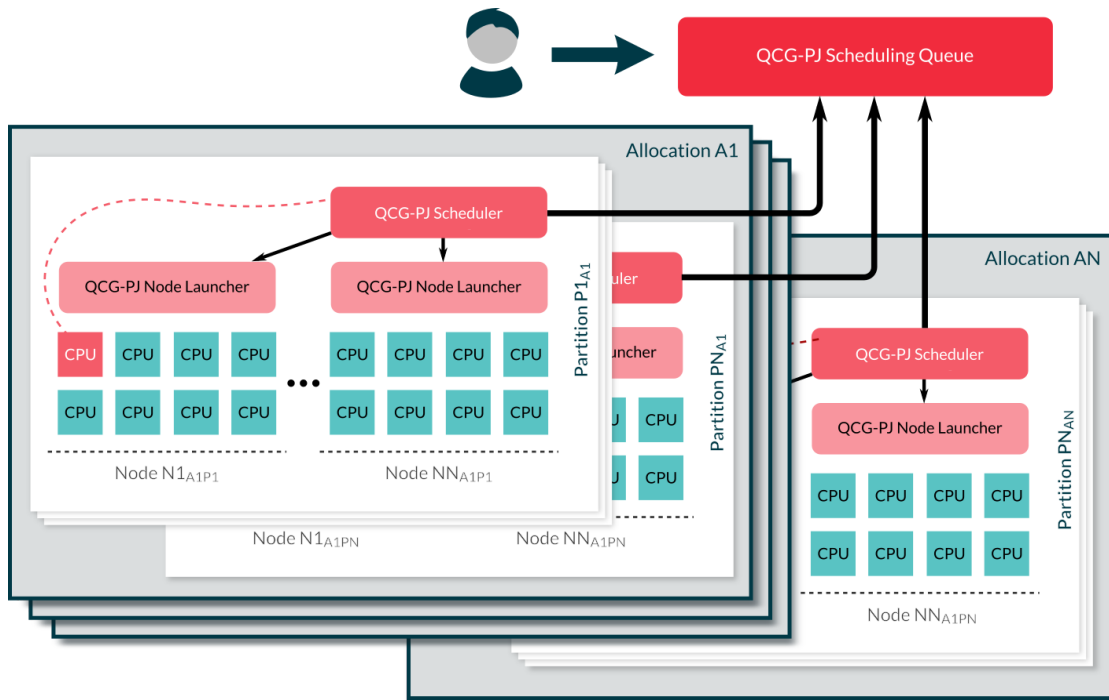


Figure 3. High-level picture on the architecture of the QCG-PilotJob system with the QCG-PilotJob Scheduling Queue service employed

### Performance Tracking and Analysis Suite

As already stated, the scalability and efficiency of QCG-PilotJob plays a key role for the execution of VECMA VVUQ scenarios on peta- and exa-scale machines. Therefore, it is important to determine the limits of the software in general, but also to figure out if a given application use case can be efficiently handled by the software. To this end, the possibility to both measure the overheads of QCG-PilotJob itself, and to easily discover potential optimisation points in the aligning application scenario to the mechanisms of the tool are essential.

The workflow efficiency evaluation is a complex aspect dependent on many factors and sensitive to even small differences in the scenarios being executed, such as resource allocation, as well as task assignment or scheduling. Due to the main design assumption, which is simplicity of deployment and use, QCG-PilotJob doesn't contain any advanced scheduling policies. Instead, jobs in QCG-PilotJob are considered, and if possible, executed in a FIFO manner (first in, first out) with elements of back-filling (i.e. tasks further-down in a scheduling queue might be executed earlier if they fit into currently available resources). The user has an influence on the efficiency of scheduling operation, only by

skilful adjustment of the allocation size and tasks to be run. Therefore, in order to measure the performance of QCG-PilotJob reliably, it is extremely important to look at the effectiveness of the tool, taking into account the unavoidable influences of unbalanced allocation of tasks to resources.

For the sake of performing QCG-PilotJob efficiency tests in a repeatable and unambiguous manner, a dedicated Performance Tracking and Analysis Suite has been developed. The suit consists of two main elements:

- a lightweight wrapper on tasks started by QCG-PilotJob which collects data on the start-time and end-time of each task;
- a command-line *qcg-pm-report* tool that allows the production of various reports based on the collected timings.

To determine QCG-PilotJob execution efficiency, i.e. to measure the time spent on assigning jobs to available resources, as well as, launching and managing tasks, we considered implementation of one of two methods. The first method (METHOD 1) was to measure the theoretical minimum time of running specific tasks on available resources without QCG-PilotJob and compare it with the time of running the same scenario with QCG-PilotJob. The alternative (METHOD 2) was to measure and sum-up the time intervals when available resources have not been utilized, that is the intervals between one process finish time and the next process start time with regard to the scheduling plan. For an overview of these two methods please see Figure 4. Note that the presented scenarios are greatly simplified and we assume here only a serial execution of pilot job tasks. Since the performance of the former method is conditioned on the theoretical assumptions that the exact runtime of each of the tasks launched by QCG-PilotJob is known in advance, as well as that there are no external factors, such as temporary I/O performance of the file system or network connections influencing the computations, we decided on the implementation of the latter method, which is free of such drawbacks.

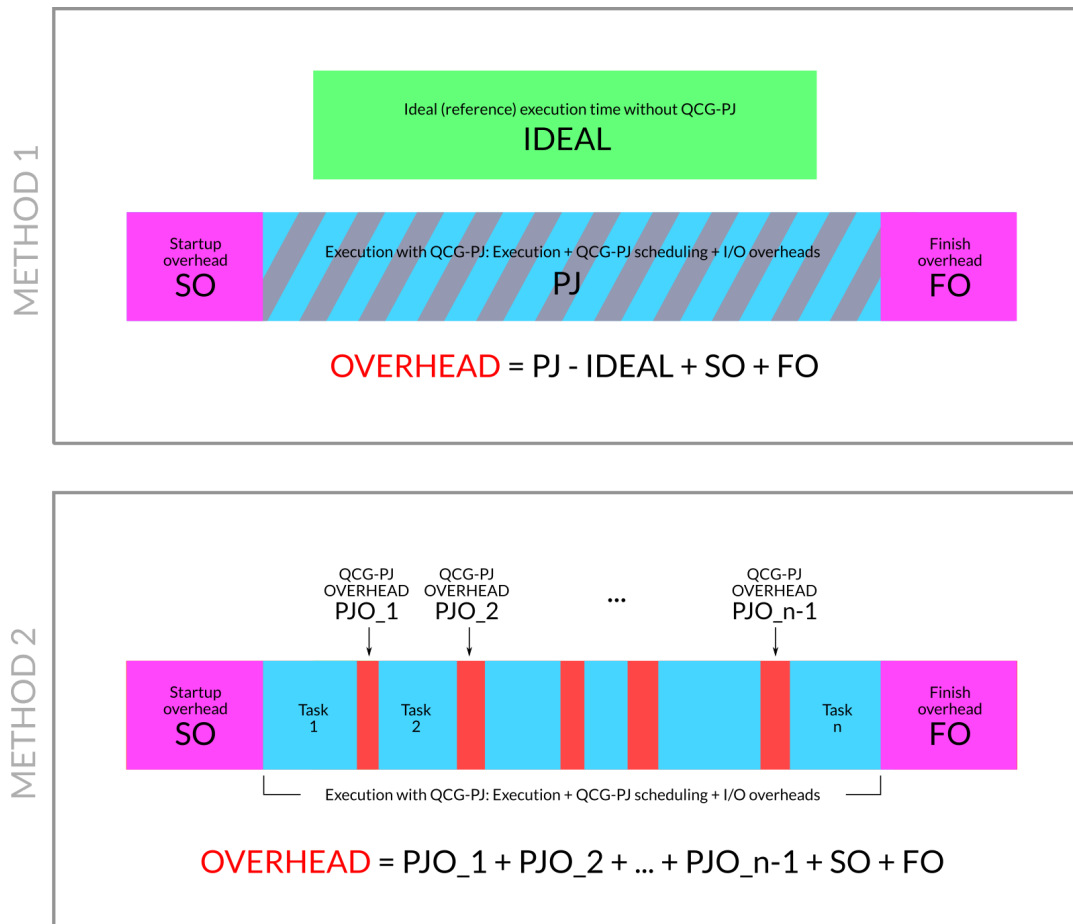


Figure 4. Two ways of measuring QCG-PilotJob performance. METHOD 1 is based on a comparison with a reference execution, METHOD 2 on measuring the resources idleness. The second method has been assumed as more realistic and therefore selected for the implementation

QCG-PilotJob provides a set of tools that not only shows the metrics describing the performance of the computation, but also helps to diagnose potential scheduling imbalance problems. All these tools are available with the `qcg-pm-report` command by passing the name of the tool as the first argument and the location of the working directory where QCG-PilotJob was executed as the second. Let us briefly introduce each of the currently available tools:

- **stats:** displays basic metrics such as allocation size, number of jobs (including those terminated with an error), service initialization, run and termination times, initialisation and service termination overhead expressed as both percentage of the total runtime and core-hours.
- **rusage:** generates metrics related to resource usage. It displays the number of cores used in the computation (which may be lower than the total number of allocation cores) as well as their usage percentage. The utilization rate is calculated as the ratio of the time during which any job was running on each core to the total runtime of the QCG-PilotJob service. The optional **--details** argument allows you to generate an individual utilization percentage for each core included in the allocation.
- **efficiency:** shows the resource usage as a percentage if we exclude the time when the resource was inactive due to a scheduling plan. Resource usage time is assumed to be the time when a task was running or when a task was waiting for another task(s) to free up resources. Thus, the efficiency metric only takes into account delays due to QCG-PilotJob's job launching and termination handling.

- **launch-stats:** provides metrics showing the delay of the QCG-PilotJob service in starting and recording the completion of a job.
- **gantt / gantt-gaps:** depending on a specified command, this generates the timeline plot with marked runtimes of individual tasks on allocated resources or a timeline plot with marked moments when resources were not used. These plots are the easiest way to spot scheduling imbalance problems and mismatches between selected task scheduling and available resources.

For the full reference please see the QCG-PilotJob documentation available at *readthedocs* [8].

## 2.2.2 Performance evaluation

To measure the actual performance of QCG-PilotJob in a real environment and with realistic scenarios, WP5 has organised and led software testing activities on several large-scale HPC machines across Europe. Testing began in the middle 2020 on SuperMUC-ng by testing the performance of integrated EasyVVUQ with QCG-PilotJob through EQI, however the actual testing of QCG-PilotJob intensified in the period April 2021 to November 2021, after the release of the QCG-PilotJob Performance Tracking and Analysis Suite which makes the collection of performance statistics and analysis of results easier.

### Initial tests of EQI

EasyVVUQ library has been designed to allow the exchange of execution backends depending on the complexity and demands of the VVUQ scenarios. From the very beginning of the project, QCG-PilotJob has been considered as a prospective software that could run a huge number of ensembles efficiently, and what's more, could be easily incorporated into EasyVVUQ as an execution engine. In order to verify the actual readiness of QCG-PilotJob to meet these objectives, a set of initial performance tests has been performed over the EQI tool on the SuperMUC-NG supercomputer. Since EQI, being the lightweight and non-intrusive integrator between EasyVVUQ and QCG-PilotJob, imposed only a marginal overhead on the operation of QCG-PilotJob, the tests, in practice, were a reliable measurement of the performance of QCG-PilotJob on its own. Figure 5 presents the operation of EQI for one of the test use-cases, where 50 dual-socket nodes of SuperMUC-NG were used. The crosses indicate the start of new UQ tasks. It can be noted that the shifts between these crosses, which reflects delays associated with performance of QCG-PilotJob, are negligible, which means that QCG-PilotJob remained reliable for the whole test lifetime. These initial performance verification tests have confirmed the applicability of QCG-PilotJob to the needs of EasyVVUQ and motivated the consortium to more closely integrate EasyVVUQ and QCG-PilotJob. Consequently, EQI has been abandoned and QCG-PilotJob has been incorporated into EasyVVUQ as an execution engine.

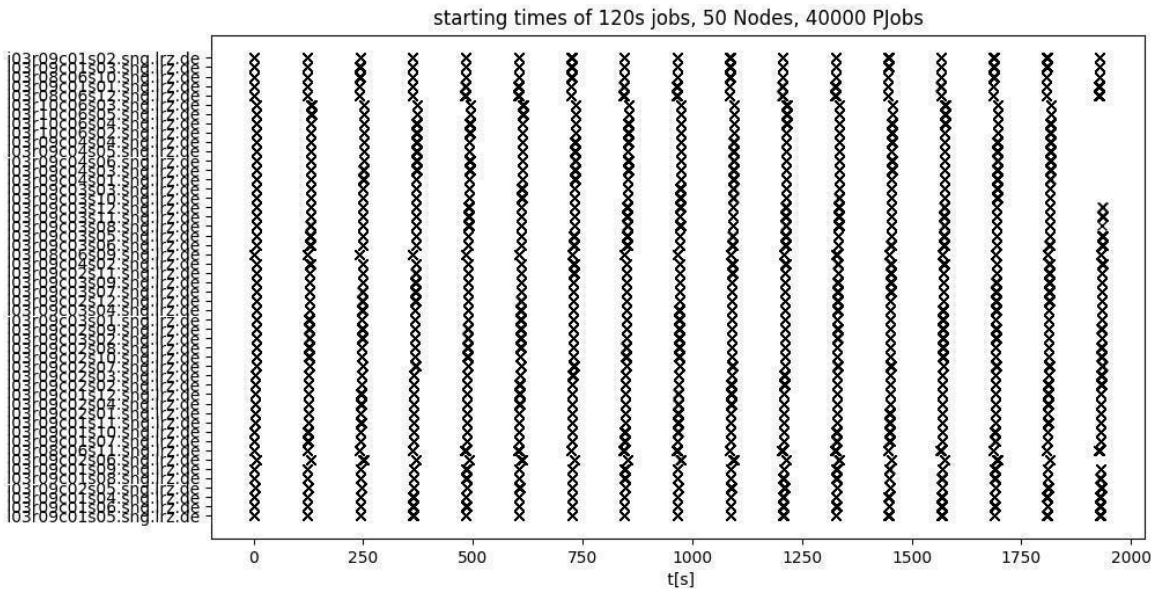


Figure 5. Performance tests of EQI on SuperMUC-NG

### Synthetic Performance Tests on Altair

To analyse the performance of QCG-PilotJob more deeply and search for optimisation points which may be critical for the VECMA's aspiration to reach the exa-scale, several further large-scale test campaigns have been established. The first of these was run on the Altair supercomputer before it was officially commissioned. With access to an allocation of 800 computing nodes equipped with 2 Xeon Platinum 8268 24C 2.9GHz processors each and InfiniBand EDR interconnect, we were able to instantiate test scenarios that run on up to 38,400 cores. The exclusive access to the resources was a great opportunity to perform reliable and extensive tests that were not affected by the execution of competing tasks. Consequently, we were able to define demanding HTC-like testing scenarios that were characterised by execution of many relatively short (in terms of walltime) and small (in terms of number of used cores) tasks. The tasks themselves were simple “sleeps”, so did not influence the operation of QCG-PilotJob. Importantly, we took care to properly assign tasks to resources to ensure that situations where resources were not used were held to a minimum. The scenarios were executed for the first time with the support of the newly implemented Performance Tracking and Analysis Suite, in two different manners:

- with a single instance of QCG-PilotJob Manager (single partition),
- with many instances of QCG-PilotJob Manager (many partitions).

The test results for these two schemes are shown in Figure 6 and Figure 7 respectively. Yellow points show the resource usage, grey bars represent the overhead induced by both the initialisation and finalisation of QCG-PilotJob services, and the blue bars show the execution efficiency. It is important to note that the presented efficiency shouldn't be considered as the actual efficiency of the operation of QCG-PilotJob. The values presented in these two charts are calculated in a simplified manner as the resource-usage during the task execution phase. Since this measure is skewed by scheduling unalignment i.e. all gaps when tasks are awaiting resources are considered as inefficiencies, the real efficiency of operation of QCG-PilotJob during execution of tasks is higher. Note that in the case of this scenario the bias was mitigated with the carefully selected workflow of tasks, so that gaps were minimised, but it would have a greater impact in cases of unaligned workflows.

Synthetic tests of QCG-PilotJob on ALTAIR (single instance)  
48 cores / node; single task size: 24 cores, task average runtimes 30s and 60s

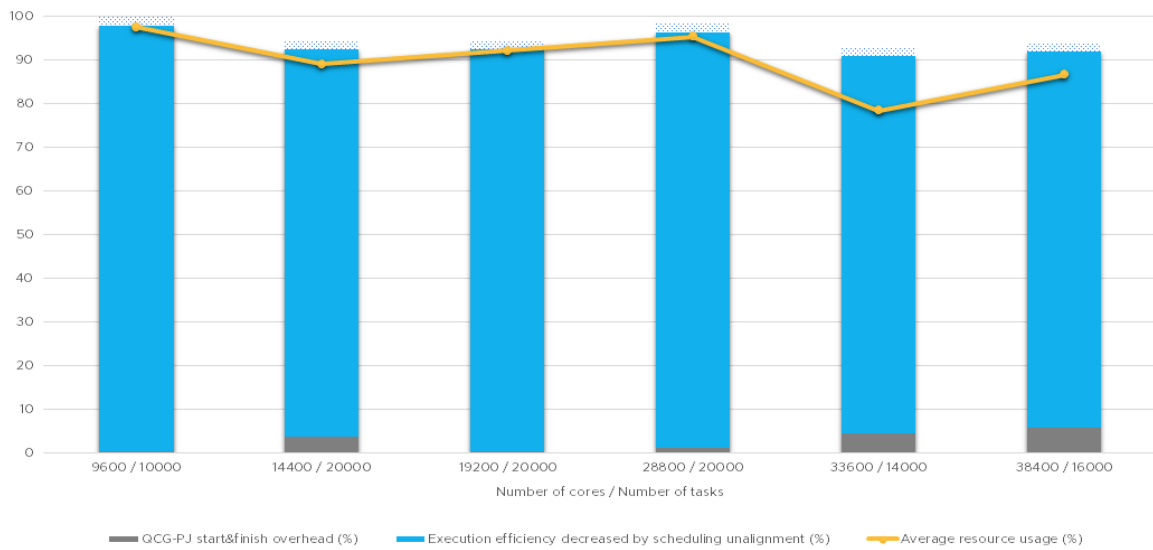


Figure 6. The results of QCG-PilotJob performance tests from Altair when a single instance of the service was used

Synthetic tests of QCG-PilotJob on ALTAIR (partitions)  
48 cores / node; single task size: 8 cores, task average runtimes 180s; partition size: 50 nodes

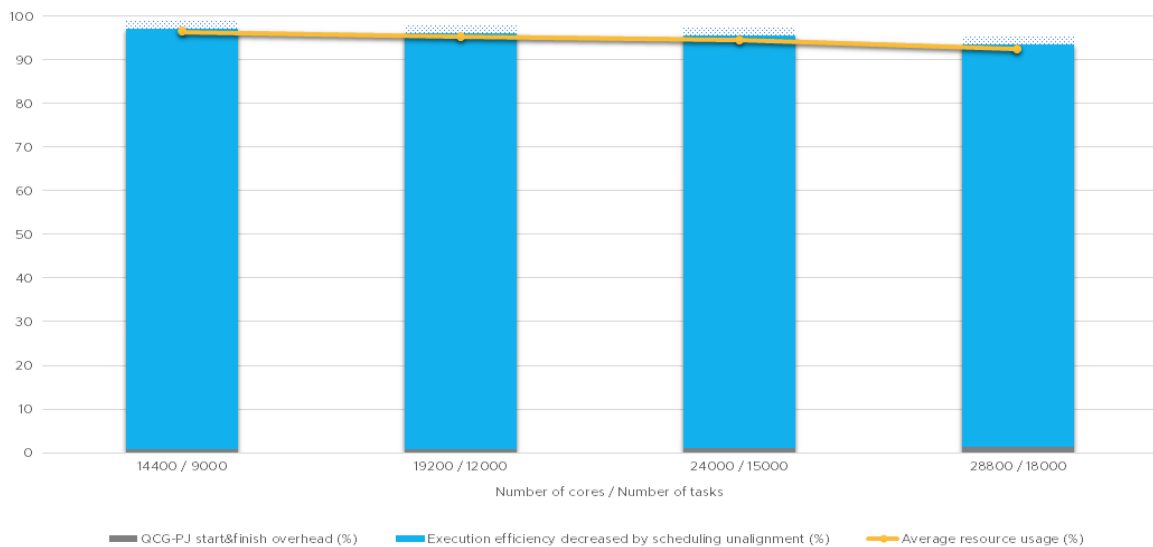


Figure 7. The results of QCG-PilotJob performance tests from Altair in a partitioned execution model

It is found that the performance of QCG-PilotJob is better in partitioned mode, but in general it is relatively high regardless of the use of only a single instance of QCG-PilotJob manager or the partitions. Both the resource utilization and efficiency metrics show a slight tendency towards decreasing as the number of resources increases, but remains relatively high even for 38,400 cores. In case of single instance tests, there are visible fluctuations which result from some random conditions on the machine (e.g. a delay in launching a node), but the overall evaluation is not affected.

## LAMMPS-based performance tests

Having the basic performance of QCG-PilotJob validated on Altair, the strategy of testing moved from the insertion of "sleep" commands towards the execution of more realistic computational tasks that make use of MPI. Consequently, a relatively simple LAMMPS [3] use-case has been selected as the basic unit of tests performed across several large-scale HPC machines.

The LAMMPS benchmark consists of an all-atom molecular dynamics simulation modelling an infinite sheet of the 2D material graphene oxide together with a polymerised monomer parylene-C. The total number of atoms is 85,884 and the simulation is run under an NPT ensemble where the pressure and temperature are kept constant whilst volume of the simulation box is varied.

In contrast to the tests carried out on Altair, during the tests with LAMMPS we focused only on executions with a single instance of QCG-PilotJob (without partitions), which provides a lower-bound estimation for the possible performance. It should also be noted that during these tests we were able to use the new version QCG-PilotJob Performance Tracking and Analysis Suite, which allowed us to get the efficiency metric, which is no longer dependent on gaps in a scheduling plan, and thus present more reliable values to discover upper limits of QCG-PilotJob.

### ○ ARCHER2

The first resource where the LAMMPS scenario has been executed was the preview installation of ARCHER2. This machine, available to the project thanks to the collaboration between VECMA and the CompBioMed Centre of Excellence [18], offered 1,000s of computing nodes. We utilised up to 800 nodes which equals 102,400 CPU cores.

QCG-PilotJob tests results with LAMMPS on ARCHER2

128 cores / node; single task size: 64 cores, single task average runtime ~ 12.5 s;

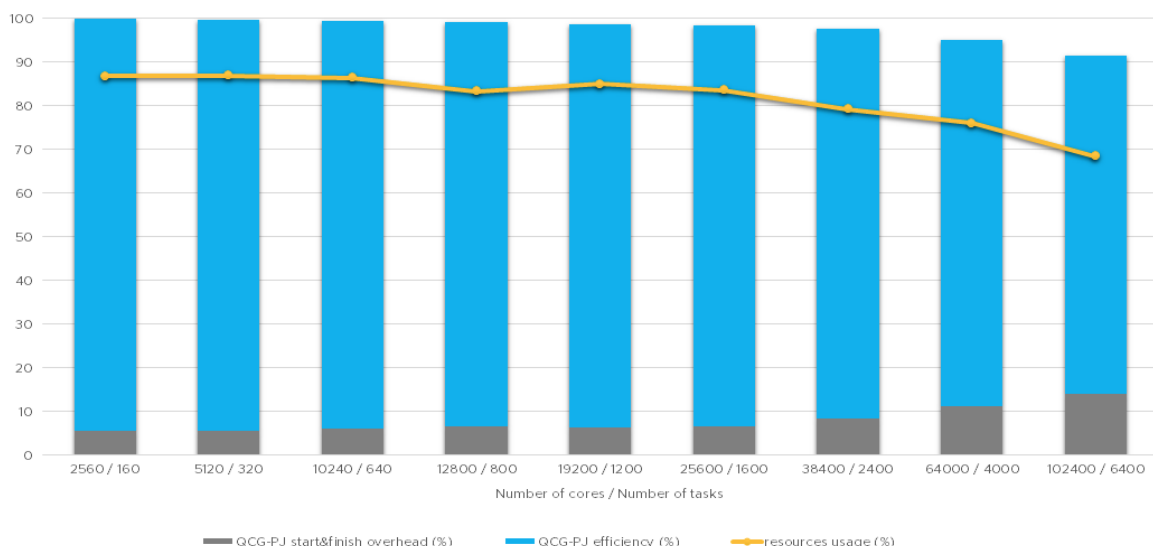


Figure 8. The results of QCG-PilotJob performance tests on ARCHER2

The test results are shown in Figure 8. Similarly, to the previously presented charts, resource usage is indicated with yellow points, and the start and finish overhead with grey bars, blue bars show the newly introduced efficiency metric.

The test results confirm relatively good performance of QCG-PilotJob, however they quite precisely display the tendency of QCG-PilotJob to slowly drop the performance metrics for a given scenario starting from 25,600 CPU Cores. To the largest extent, the resource usage is downgraded by the overhead related to the start and finish of QCG-PilotJob, but also the efficiency of processing tasks within QCG-PilotJob decreases to 91,4% for the largest use-case. Nevertheless, it should be noted that the scenario was executed in a basic configuration with only a single instance of QCG-PilotJob Manager. Undoubtedly, the performance would improve in case of partitioned mode of execution, as demonstrated by the tests carried out on Altair.

As a side note, we would like to highlight that further testing of QCG-PilotJob is planned on the full-scale 23-cabinet ARCHER2 machine that was recently officially commissioned (see above).

#### ○ *Cartesius*

Within a collaboration between the VECMA project and the CompBioMed Centre of Excellence, we deployed and tested the QCG-Pilotjob middleware on the Cartesius HPC system available at SURF until October 2021 (the system has been recently replaced by Snellius, the current Tier 1 HPC system at SURF [22]). The tests were executed on the “Haswell” nodes within the “thin” partition of Cartesius, equipped with 2 × 12-core 2.6 GHz Intel Xeon E5-2690 v3 (Haswell) CPUs and 64 GB of memory per node. To understand the performances of the workflow, we varied the number of total jobs per node, with the size of the allocation. We tested on 50, 100, 200 and 400 compute nodes for a total of 1200, 2400, 4800 and 9600 CPU cores jobs respectively, executing from 200 to up to 10,000 LAMMPS 6 cores runs. The test results are shown in Figure 9.

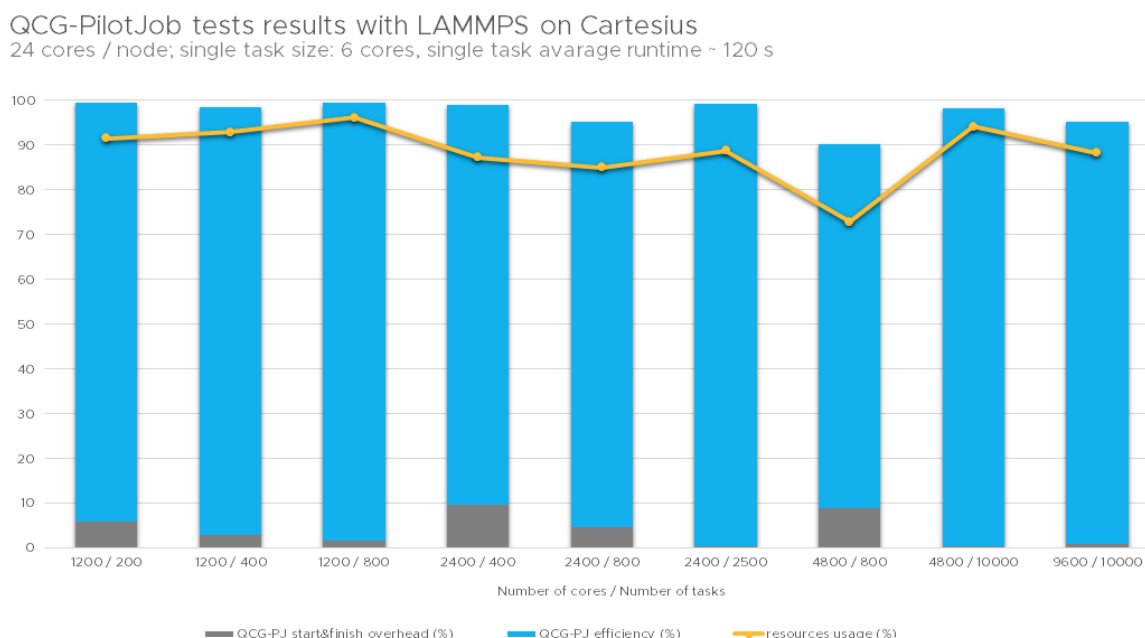


Figure 9. The results of QCG-PilotJob performance tests on Cartesius

All test scenarios show a low overhead from QCG-PilotJob. Looking at the variation of service runtime with the total number jobs for a fixed number of total nodes, almost linear scaling is observed, confirming that the middleware can efficiently handle the increased amount of work. Interestingly, at

Cartesius we noted fluctuations in performance that were greater than those observed on Altair. It is assumed from this that this is a common characteristic of many HPC systems, possibly due to the temporary load, however since it causes significant inefficiencies, it should be analysed further in the future research.

There is ongoing work to deploy QCG-PilotJob on the replacement of Cartesius, i.e. Snellius [23]. This required preparation of the EasyBuild [14] recipe for QCG-PilotJob, which was recently completed by SURF. After completion of other required works, we will perform tests of QCG-PilotJob also on this new resource at SURF.

#### ○ **SuperMUC-NG**

Testing of QCG-PilotJob on SuperMUC-NG has been performed iteratively through the project, but was intensified during the block operation of the machine, which took place in mid-November 2021 and allowed us to run large test-cases without long queues. Eventually we were able to run the LAMMPS use cases on up to 1000 nodes, which corresponds to 48,000 CPU cores. As it is presented in Figure 10, for the tested scenarios we observed relatively large overhead caused by start and finish of QCG-PilotJob, but the overall efficiency of the execution has remained high.

QCG-PilotJob tests results with LAMMPS on SuperMUC-NG  
48 cores / node; single task size: 24 cores, single task average runtime ~ 36 s

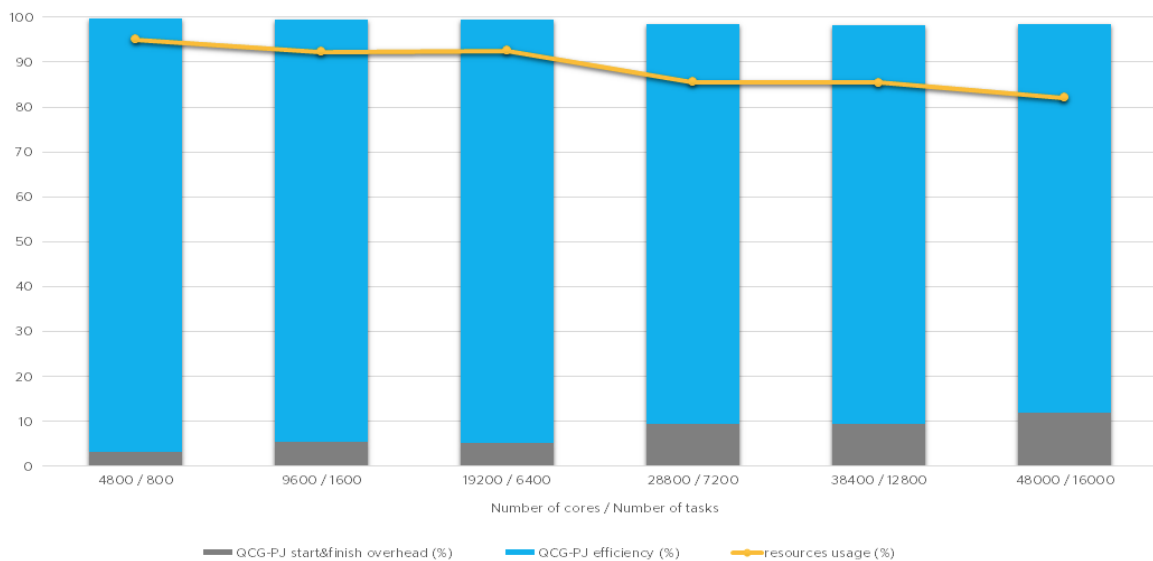


Figure 10. The results of QCG-PilotJob performance tests on SuperMUC-NG

Unfortunately, attempts to run larger test scenarios were blocked by SuperMUC-NG administrators due to the observed instability of the SLURM system installed at the resource when it has been subjected to many invocations of the *srun* command by QCG-PilotJob. Although expected to occur at some scale, this limitation has not been discovered in the previous tests as a real issue. Therefore we were not able to act in advance to mitigate the problem. We are currently investigating different options of replacing the basic mechanism used by QCG-PilotJob for launching tasks and MPI applications. So far we proposed a relatively simple fix which replaces invocations of *srun* with compatible invocations of *mpirun* via *ssh* and we are now repeating the tests with this new mechanism on SuperMUC-NG, but sadly outside of block operation. Based on the test results we will

decide whether other options, such as those presented in a recently published paper [1] for RADICAL-Pilot, should be incorporated into QCG-PilotJob.

### Summary from the performance tests

The assembly of test results shows good performance of QCG-PilotJob in execution of demanding HTC-like test cases on up to 100,000 CPU cores. The tests have generally been successfully processed on all resources, with the exception of SuperMUC-NG, where tests on more than 1,000 of nodes have been declined due to the high influence of the test scenarios on SLURM. Following modification of QCG-PilotJob, we are repeating the tests on this machine. There were also visible fluctuations of the performance metrics on selected machines, which are likely due to the random conditions on these resources, but should be a matter of further analysis in order to check possibilities of adjusting QCG-PilotJob to counter such specific circumstances. We have tested QCG-PilotJob mostly in a non-partitioned mode since it was sufficiently objective, but also because the partitioned mode is going to be replaced soon by the Scheduling Queue service. Undoubtedly, we are looking forward to making the performance tests of the scenarios employing this new mechanism.

### 2.2.3 Integration with User-level tools

The basic way of using QCG-PilotJob assumes its direct use to help and automate the execution of a large number of users' computing tasks on HPC resources. An alternative approach is to integrate QCG-PilotJob as an execution backend into user-facing tools. This second approach is essential in the case of VECMA, where the EasyVVUQ [9], FabSim3 [11] and MUSCLE3 [13] tools all leverage QCG-PilotJob to run computing tasks. Within this section we discuss how the integration has been realised for each of these tools.

#### EasyVVUQ

The wide support of sampling and UQ algorithms by EasyVVUQ and its intuitive usage have resulted in high uptake by diversified application teams. EasyVVUQ may be used on its own to perform UQ or SA of simple simulations on a users' laptop, but it is most powerful when combined with other tools, such as FabSim3, Dask or QCG-PilotJob, and applied to execute large use cases on large-scale machines.

Before EasyVVUQ gained maturity, the intensive cross work package activities between WP3 and WP5 had been undertaken to enable execution of EasyVVUQ campaigns with QCG-PilotJob. In the next section we will describe the integration of QCG-PilotJob with FabSim3, where both QCG-PilotJob and EasyVVUQ can be combined together into the FabSim3 workflow. Here we would like to focus on the direct integration between QCG-PilotJob and EasyVVUQ that allows the leveraging of HPC machines for efficient and scalable execution of ensembles of tasks instantiated on a level of EasyVVUQ itself. Due to the ongoing development of both EasyVVUQ and QCG-PilotJob, which resulted in constantly appearing changes in the APIs or internal logic, the process of integration was iterative and divided into the preparation of three subsequent solutions:

1. Preparation of an example workflow based on a low-level QCG-PilotJob API.
  - adaptation of an example is complicated and error-prone
2. Development of EasyVVUQ-QCGPJ (EQI) [10] as an external library integrating both tools.
  - possibility to predefine a set of optimised workflows for common scenarios

- lack of support for more sophisticated VVUQ algorithms (e.g. Markov Chain Monte Carlo)
  - additional efforts required to install and configure EQI
  - changes in QCG-PilotJob or EasyVVUQ entail adaptation of EQI (lack of support starting from EasyVVUQ 1.0)
3. Built-in integration of QCG-PilotJob in EasyVVUQ based on QCG-PilotJob Executor API:
- easy installation and usage (all dependencies already available in EasyVVUQ)
  - compliance with all EasyVVUQ algorithms
  - for some scenarios possible lower efficiency in comparison to EQI

Undoubtedly, the last solution should be considered as the final one. Its realisation has been partially grounded in the previously developed EQI library, but additional efforts were needed to make the integration homogeneous with the concepts of pools and executors employed in EasyVVUQ from version 1.0. Thus, a new Executor API on the QCG-PilotJob side, and a dedicated executor adapter on the EasyVVUQ side have been developed for the VECMA's M36 release. These updates allowed QCG-PilotJob to be transparently and interchangeably used with other executors already available in EasyVVUQ.

### **FabSim3**

FabSim3 is a software toolkit for automating computational research activities that rely (mainly) on the use of remote resources. It is currently in use by a wide range of application communities, particularly in migration and epidemiological modelling. FabSim3 integrates with QCG-Client [12], as well as with QCG-PilotJob. Although the QCG-Client integration has found little uptake during the project, the QCG-PilotJob integration has been used extensively by a diverse user communities. This includes communities in VECMA, as well as those in other EU projects (e.g. HiDALGO [15], STAMINA [16]), national projects (SEAVEA [17]) as well as those further afield (e.g. by research groups at Columbia University in the US). FabSim3 users can enable the use of QCG-PilotJob to more rapidly allocate and execute their ensembles, simply by using a `'PJ=true'` flag in their command-line syntax when submitting jobs.

FabSim3 also contains installation commands that automate the installation of QCG-PilotJob on a number of remote machines in the VECMA testbed, such as Altair. Moving forward, we are planning to maintain, enhance, and promote uptake of the integration of FabSim3 and QCG-PilotJob for the duration of the newly started SEAVEA project (which runs from August 2021 until October 2024).

### **MUSCLE3**

MUSCLE3 is the third incarnation of the MultiScale Coupling Library and Environment, developed independently by the University of Amsterdam and the Netherlands eScience Center (associate partner of VECMA and contributor to the VECMA toolkit). It connects multiple submodels together into a multiscale simulation in a flexible way and supports submodel ensembles, thus making it possible to implement advanced Uncertainty Quantification Patterns [1]. MUSCLE3 is used in the VECMA applications ISR3D, Fusion and Flee.

MUSCLE3 can be used in combination with EasyVVUQ for non-intrusive Uncertainty Quantification, EasyVVUQ running multiple instances of the full MUSCLE3 simulation. This was applied in VECMA to develop subgrid-scale surrogate models for a 2D Gray-Scott model [1]. Furthermore, a MUSCLE3

extension has been developed that allows it to use QCG-PilotJob to orchestrate the many submodel instances making up a multiscale simulation within an HPC resource allocation. Together with an in-development resource scheduling model for multiscale simulations this will relieve the user of the burden of manually sub-allocating resources and allow MUSCLE3 to perform optimisations, thus making it both easier to use and more performant.

## 2.3 VECMA Jupyter Notebook Platform

Jupyter Notebook is a popular and grounded technology for interactive computing. The possibility to mix documentation and live code is particularly useful for the preparation of tutorials to enable trainees to easily familiarise themselves with a given library or API. To implement this solution in VECMA, WP5 has configured a dedicated JupyterLab<sup>1</sup> platform on the PSNC PaaS infrastructure. The VECMA Notebook Platform (see *Annex II – VECMA Notebook Platform*) [21] has been armed with a set of functionalities and settings adjusted to the VECMA project's needs, namely:

- capability to run multiple instances of JupyterLab in parallel
- assignment of multiple CPU cores for a single JupyterLab instance
- user access based on locally created accounts, as well as using google credentials
- automatic mounting of the repository with tutorial materials
- styling of the interface in accordance with the VECMA branding

The provided platform has been used during different events organised by the VECMA project. The platform together with a set of predefined notebook tutorials for EasyVVUQ, FabSim3, EasySurrogate, EQI and QCG-PilotJob have been employed during all VECMA training events, during VECMA hackathons and also individually by internal and external users, depending on their needs. As of November 30 2021, there were 98 separate user accounts activated on the platform.

## 2.4 Applications and Infrastructure

The ultimate target of all actions undertaken in WP5 was to efficiently support VVUQ of multiscale and multiphysics applications by simplification of the access to large-scale hardware resources and optimisation of the code execution by means of user-friendly and scalable software components. The information presented in Table 3 is a summary showing which Infrastructure elements have been employed by particular applications, including both computing resources and software components. For each application it also provides information about the resource requirements of the largest problem executed to date and an estimation of resource requirements of a problem the scenario is aiming to solve in the foreseen future, both expressed in Core Hours. For the detailed science-oriented descriptions of VECMA applications we encourage the reader to review the WP4 documents, in particular Deliverable D4.3 [6].

*Table 3. Overview on applications in the context of infrastructure usage*

Name	Description	Resources used	WP5 tools and services used	Resource requirements (Core Hours)		Notes (eg. information about scalability tests)
				Current	Aiming at	
UrbanAIR	Simulation of air pollution in	Eagle / Altair,	QCG-Broker, QCG-PilotJob,	36,864	534,528	Smooth and efficient

<sup>1</sup> <https://jupyter.org/>

	urbanised areas	Tryton, Prometheus	QCG-Client, QCG-Now, QCG-Monitoring			execution with EasyVVUQ and QCG-PJ on Altair.
CovidSim	Model of Imperial College for estimating effect of Non Pharmaceutical Interventions	Eagle	Fabsim3, QCG-PilotJob	381,000	381,000	No issues with scalability. Combination Fabsim + QCG-PJ + Eagle worked well.
Multiscale Fusion Workflow (MFW)	Model fusion devices on the slow (transport) time-scale based on fast (turbulence) processes	Marconi, Cobra, Draco, Raven	EasyVVUQ, MUSCLE2, MUSCLE3	100,000	100,000,0 00	Scalability of the components has been demonstrated.
ISR2D/3D	In-stent restenosis model simulating post stenting tissue growth	Eagle, Cartesius	MUSCLE2, MUSCLE3	300,000	10,000,00 0	
Multiscale Migration Prediction	An agent-based simulation tool predicting forced displacement movements	Eagle/Altair	FabSim3, FabFlee plugin, EasyVVUQ, QCG-PilotJob	20,480	250,000*	No issues with scalability. Uses FabSim + QCG-PJ. Occasional stability issues with Python module loads.
BAC	A workflow to compute binding energies to determine molecules efficiency to bind to specific proteins	SuperMUC	EasyVVUQ, QCG-PilotJob	2,000,000	2,000,000	
FACS	An agent-based simulation tool modelling the spread of COVID-19 in local areas	Eagle/Altair	FabSim3	8,192	1,000,000 **	No issues with scalability. Uses FabSim + QCG-PJ. Occasional stability issues with Python module loads.
Materials	Atomistic simulations of the material properties of	Eagle/Altair	EasyVVUQ, QCG-PilotJob	700,000	700,000	

	graphene					
--	----------	--	--	--	--	--

\* Due to the urgent nature of conflicts, Flee simulations should be able to be run very quickly. So the (modest) 250,000 target should involve a large ensemble that can be turned around in hours.

\*\* For FACS we are working towards a national-level model, which will rely on the newly developed parallel code.

## 2.5 Dissemination activities related to Work Package 5

The final 2 years of the project have allowed us to organise numerous dissemination activities through WP5. Detailed description of these activities is one of the topics of D6.5, but we briefly outline them here, in Table 4, as they reflect and show the direction of all activities realised under WP5.

*Table 4. The list of dissemination activities of Work Package 5*

Name of activity	Description
Preparation of the promotional animation titled "QCG tools and services for VECMA"	The 4 minutes long animation has been developed to advertise a family of QCG services and tools, with the particular focus on QCG-PilotJob usage in VECMA.
Preparation of the "Verification, Validation and Uncertainty Quantification of Large-Scale Applications with QCG-PilotJob" paper for MMS Workshop 2021	The paper describing usage of QCG-PilotJob in VVUQ scenarios has been prepared for the Multiscale Modeling and Simulation workshop collocated with the ICCS 2021 conference. The paper has been presented and included in the conference proceedings.
Creation of poster titled "Large-Scale Computations with QCG-PilotJob" for ISC 2021	Preparation of poster and associated short movie for the ISC 2021 Project Poster Reception with the QCG-PilotJob tools as a main topic.
Talk titled "Leverage large-scale HPC machines with QCG-PilotJob" at CompBioMed Conference 2021	Talk given during CompBioMed Conference 2021 with the goal to demonstrate applicability of QCG-PilotJob and VECMAtk to the CompBioMed project's use cases.
Organisation of Workshop titled "Overcome limitations of Scheduling Systems with QCG-PilotJob" SCFE conference	Preparation and conduction of Virtual Workshop and Training event devoted to the usage of QCG-PilotJob in diversified computational scenarios. The event took place as part of the Supercomputing Frontiers Europe 2021 conference.
Organisation and participation in VECMA training events and hackathons	Work Package 5 representatives have organised and participated in several VECMA trainings and hackathons where they had a role of experts regarding access to the computing Infrastructure and usage of WP5 software.

## 2.6 Collaboration with external projects

As the VECMA project progressed and the developed software matured, a number of external projects expressed interest in the usage of the VECMAtk components. In the context of the activities realised under the umbrella of WP5, particular popularity has been achieved by QCG-PilotJob and FabSim3 (the middleware part of the tool from the perspective of WP5). These tools have been recognised as being useful by such projects as CompBioMed2, SEAVEA, PIONIER-LAB and CovidSim, and thereby applied to their use-cases. Table 5 summarises all major collaborations.

Table 5. The list of projects that expressed interest in usage of VECMA Work Package 5 outcomes for their needs

Project	VECMA WP5 Components	Description of use-case
ComBioMed2	QCG-PilotJob	ComBioMed2 [18] seeks a tool that allows a large number of jobs to be executed in ensemble or replica computing scenarios. Thanks to this collaboration QCG-PilotJob has been tested on several large-scale HPC machines, including ARCHER2 and Cartesius. This collaboration resulted also in development of an EasyBuild [14] recipe for QCG-PilotJob.
HiDALGO	FabSim3, QCG-PilotJob	Both tools are extensively used by the HiDALGO's [15] migration pilot, in particular for the models of Tigrayan conflicts in Ethiopia, which is done in collaboration with two NGOs.
SEAVEA	FabSim3, QCG-PilotJob	SEAVEA [17] essentially continues the development and maintenance of VECMAtk, and will expand it also to include much more surrogate modelling approaches, as well as further improved scalability. It also includes a wide range of application use cases, ranging from fusion to migration and biomedicine, among many others.
PIONIER-LAB	QCG, QCG-PilotJob	PIONIER-LAB [19] is a Polish national project that constitutes a set of specialised laboratories. PSNC is leading the project and also the Multiscale Simulations Laboratory, where QCG and QCG-PilotJob are planned to be used in order to support computational use-cases.
CovidSim Analysis	FabSim3, QCG-PilotJob	Rapidly constructed uncertainty studies for CovidSim model [2] upon which the UK government relied in part to model various non-pharmaceutical intervention strategies, have been possible thanks to joint work of FabSim3 and QCG-PilotJob
STAMINA	FabSim3, (QCG-PilotJob)	STAMINA [15] focuses on modelling pandemic spread, among many other use cases. In the case of COVID-19 modelling, we rely on FabSim3 to perform ensemble simulations, and make use of QCG-PilotJob when the ensemble becomes particularly large.
ITFLOWS	FabSim3	ITFLOWS [20] focuses on migration modelling. Here the modelling of migration away from local conflicts is done using FabFlee. As part of the project, we also perform limited-scale uncertainty quantification.
Cloud UQ Resources	EasyVVUQ (QCG-PilotJob)	Utilising cloud resources to perform Monte-Carlo based Uncertainty Quantification of fusion simulations, which builds on the VECMA framework: EasyVVUQ with integrated QCG-PilotJob.

### 3 Conclusions


We have presented a summary of the activities we conducted within WP5 in the timeframe of the VECMA project. The works committed to the VECMA system architecture development, as well as, to [D5.3 Final report on the VECMA infrastructure]

enable access to the computing resources for the VECMA partners have been naturally driven by the needs of VECMA and has allowed for the progression of VECMA scientific scenarios. Many of the tasks performed in the WP5 resulted in the creation of long-lasting outcomes, which we expect to result in the continued uptake of the various components of VECMAtk long after the lifetime of the VECMA project. QCG-PilotJob and FabSim3 for example are already in use by user communities across at least half a dozen scientific disciplines, and across Europe, and beyond. It is all but certain that the results of WP5 will be used soon much more broadly by VECMA partners, and by external organisations. The already diverse uptake of the tools at the time of writing this deliverable, as well as the funding of a follow-up project (SEAVEA) can be seen as an early sustainability success of the project as a whole.

## 4 References

1. H. Lee, A. Merzky, L. Tan, M. Titov, M. Turilli, D. Alfe, A. Bhati, A. Brace, A. Clyde, P. Coveney, others, Scalable HPC & AI infrastructure for COVID-19 therapeutics, in: Proceedings of the Platform for Advanced Scientific Computing Conference, 2021: pp. 1–13.  
DOI:10.1145/3468267.3470573
2. D. Ye, L. Veen, A. Nikishova, J. Lakhilili, W. Edeling, O. O. Luk, V. V. Krzhizhanovskaya and A. G. Hoekstra, “Uncertainty quantification patterns for multiscale models”, Phil. Trans. R. Soc. A. 379, 20200072 (2021), DOI:10.1098/rsta.2020.0072
3. W. Edeling, H. Arabnejad, R. Sinclair, D. Suleimenova, K. Gopalakrishnan, B. Bosak, D. Groen, I. Mahmood, D. Crommelin, P. Coveney, “Model uncertainty and decision making: Predicting the Impact of COVID-19 Using the CovidSim Epidemiological Code”, Nat Comput Sci 1, 128–135 (2021), DOI:10.1038/s43588-021-00028-9
4. A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, S. J. Plimpton, “LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales”, Comp Phys Comm, 271 (2022) 10817.
5. [https://www.vecma.eu/wp-content/uploads/2019/12/VECMA\\_D5.2\\_First-Report-Infrastructure\\_PSNC\\_20191208.pdf](https://www.vecma.eu/wp-content/uploads/2019/12/VECMA_D5.2_First-Report-Infrastructure_PSNC_20191208.pdf)
6. [https://www.vecma.eu/wp-content/uploads/2019/03/VECMA\\_D5.1\\_Architecture\\_PSNC\\_v1.0\\_20190312.pdf](https://www.vecma.eu/wp-content/uploads/2019/03/VECMA_D5.1_Architecture_PSNC_v1.0_20190312.pdf)
7. [https://www.vecma.eu/wp-content/uploads/2021/06/VECMA\\_D4.3\\_On-the-Implementation-of-VVUQ-Techniques\\_MPG\\_v1.0\\_20210608.pdf](https://www.vecma.eu/wp-content/uploads/2021/06/VECMA_D4.3_On-the-Implementation-of-VVUQ-Techniques_MPG_v1.0_20210608.pdf)
8. VECMA toolkit: <https://www.vecma-toolkit.eu/>
9. QCG-PilotJob: <https://qcg-pilotjob.readthedocs.io>
10. EasyVVUQ: <https://easyvvuq.readthedocs.io>
11. EasyVVUQ-QCGPJ (EQI): <https://easyvvuq-qcgpi.readthedocs.io>
12. FabSim3: <https://fabsim3.readthedocs.io>
13. QCG: <http://qcg.psnc.pl>
14. MUSCLE3: <https://muscle3.readthedocs.io>
15. EasyBuild: <https://easybuild.io/>
16. HiDALGO: <https://hidalgo-project.eu/>
17. STAMINA: <https://stamina-project.eu>
18. SEAVEA: <https://gow.epsrc.ukri.org/NGBOViewGrant.aspx?GrantRef=EP/W007762/1>
19. CompBioMed2: <https://www.compbiomed.eu/>
20. PIONIER-LAB: <https://pionier-lab.pionier.net.pl/>
21. ITFLOWS: <https://www.itflows.eu>
22. VECMA Notebook Platform: <https://jupyter.vecma.psnc.pl>
23. Snellius supercomputer: <https://servicedesk.surfsara.nl/wiki/display/WIKI/Snellius>
24. Altair supercomputer: <https://www.psnc.pl/altair-supercomputer-is-fully-operational-and-ready-to-go/>

## ● Annex I – QCG-PilotJob Documentation



Search docs

- BASICS**
  - Overview
  - Installation
- USAGE**
  - Examples
  - Modes of execution
  - Parallelism
  - QCG-PilotJob Manager options
- FURTHER READING**
  - Key concepts
  - Execution environments
  - Execution models
  - File based interface
  - Executor API
  - Iteration resources schedulers
  - Resuming prematurely interrupted computations
  - Performance statistics
  - Performance tuning
  - Processes statistics
  - Log files
- MISCELLANEOUS**
  - FAQ

Docs » QCG-PilotJob [Edit on GitHub](#)

## QCG-PilotJob

A python service for easy execution of many tasks inside a single allocation.

### Basics

- [Overview](#)
- [Installation](#)

### Usage

- [Examples](#)
- [Modes of execution](#)
- [Parallelism](#)
- [QCG-PilotJob Manager options](#)

### Further reading

- [Key concepts](#)
- [Execution environments](#)
- [Execution models](#)
- [File based interface](#)
- [Executor API](#)
- [Iteration resources schedulers](#)
- [Resuming prematurely interrupted computations](#)
- [Performance statistics](#)
- [Performance tuning](#)
- [Processes statistics](#)
- [Log files](#)

### Miscellaneous

- [FAQ](#)
- [Dictionary](#)
- [License](#)

### API Docs

# ● Annex II – VECMA Notebook Platform

