D2.1

Exascale workflow design

Alberto Garcia, Nicola Marzari, Ada Böhm, Pietro Delugas, Carlo Cavazzoni, Andrea Ferretti, Luigi Genovese, Davide Sangalli, Daniel Wortmann, Matteo Vandelli, Daniele Varsano

Due date of deliverable	31/12/2023 (**month 12**)
Actual submission date	27/12/2023
Final version	27/12/2023
Lead beneficiary	CSIC (participant number 5)
Dissemination level	PU - Public

Document information

Project acronym	MAX
Project full title	Materials Design at the Exascale
Research Action Project type	Centres of Excellence for HPC applications
EuroHPC Grant agreement no.	101093374
Project starting/end date	01/01/2023 (month 1) / 31/12/2026 (month 48)
Website	http://www.max-centre.eu
Deliverable no.	D2.1
Authors	Alberto Garcia, Nicola Marzari, Ada Böhm, Pietro Delugas, Carlo Cavazzoni, Andrea Ferretti, Luigi Genovese, Davide Sangalli, Daniel Wortmann, Matteo Vandelli, Daniele Varsano
To be cited as	Garcia et al. (2023): Exascale Workflow Design Deliverable D2.1 of the HORIZON-EUROHPC-JU-2021-COE-01 project MAX (final version as of 27/12/2023). EC grant agreement no: 101093374, CSIC, Barcelona, Spain.

Disclaimer

Contents

Executive Summary

The present deliverable reports on the design, architecture, implementation, and development plan for exascale workflows, according to Task T2.1 of WP2.

Workflows are the top-level layer in the practical use of a computing resource. Lower-level layers are composed of system software, compilers, libraries, and individual codes, and are by now reasonably automatised and/or packaged. In contrast, workflows retain an artisan character as they must be designed for specific use cases, handling the coordination of many tasks involving a variety of applications and possibly heterogeneous resources, and organising the flow of data. While the lower-level computing layers are basically standardised, there is a large degree of fragmentation in the community regarding the tools used for workflow design and orchestration.

In this document we provide an overview and a plan for the design and implementation of several exascale workflows, inspired by and targeted to specific scientific showcases. We will not aim at this stage at the creation of a general "workflow development kit", but rather at the combination of a set of tools and ideas to make it easier to handle the scientific demands. We identify common patterns for workflow design among our scientific showcases, and frame the design within the long-running MAX theme of modularity, and exploiting common ideas at different levels of the computational hierarchy.

Exascale workflow design is influenced markedly by the boundary conditions imposed on the actual use of the HPC machine(s): the degree of congestion (i.e., whether one can assume that resources are available within reasonable limits) and the needed support for fallback measures in case of system failures and shutdowns. A first design within "ideal conditions" can serve to structure the main ideas and techniques. Further refinement can introduce the needed recovery measures, within limits dictated by a balance between robustness, complexity of design, and the cost involved.

This report is organised as follows. In Section 2 we review the selected scientific use cases, highlighting background and scientific motivations, the involved computational tools, and the relevance for exascale. Then, in Section 3 we present a number of available tools (partly developed within MAX , but not necessarily) for workflow encoding, orchestration, and scheduling as well as for data-exchange and code-interoperability. Finally, in Section 4 we rationalise and present the different types of workflows targeted by MAX WP2 activities, and discuss the current status of development of the selected workflows presented in Sec. 2.

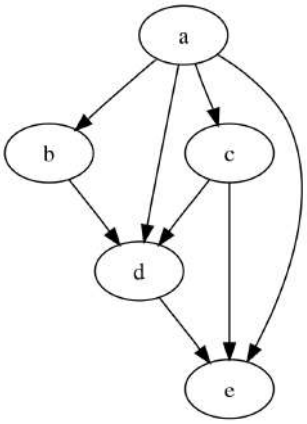

Figure 1: Sketch of a Directed Acyclic Graph which can represent a workflow if the nodes are considered as tasks and the edges as dependency conditions.

1 Introduction

Formally, a workflow is just a collection of interdependent tasks, and is thus a very general concept, which could apply to various levels of computing operations. Hence, to focus on a particular domain, one must specify the level of granularity of the tasks involved. For example, the tasks pertaining to "scientific workflows" are typically executions of simulation codes or of significant modules within them, and are interdependent because of the associated data flow. To continue with the granularity argument, the codes themselves can be considered as "pre-packaged workflows", and for the purposes of this work-package assumed to have been optimised and deployed, so they act (almost, as we will see below) as black boxes of functionality with specific data inputs and outputs.

The task interdependence in a workflow can be represented by a directed acyclic graph (DAG), see Fig. 1, but this is only the topological side of things. To actually deploy and run a workflow one needs to supplement this, in a particular computing system, with means for:

- Task distribution;
- Task allocation and execution;
- Proper data handling.

In HPC systems, and in particular in (pre)exascale machines, the allocation of resources is typically handled by a system scheduler, which implements policies for managing the resources, which might be large, but finite. Execution of each task takes place on a set of computing elements (compute nodes, disks, network interfaces) which have a (hopefully small) probability of failure. A robust workflow design, therefore, must account properly for a possible scarcity of resources, and has to be resilient to face system failures.

While aiming of course for robust workflow designs, our plans include a first stage of work under what we call "ideal conditions": (i) the execution of the workflow takes place with no congestion in the machine (e.g. extra resources, including data bandwidth, can be obtained within reasonable limits at any point); (ii) there are no external (hardware)

failures. To these we can add another one which is really code-specific, but relevant in the overall performance of workflows: (*iii*) there are no internal code failures (such as, e.g., lack of convergence of a specific step in a calculation). Designing under ideal conditions helps to focus on the basic elements and strategies of the workflow. When non-ideality is accounted for, some of the elements might have to be revised, but it is hoped that this can be done incrementally without changing the overall design.

It is here that a second pillar of our design plan enters: we will have a repertoire of basic tools and strategies for task distribution, communication, and execution, that are modular and very specific to the role at hand, and that interface with (but do not aim at substituting) the system-provided tools. This is very important for portability, in addition to the proven benefits of modular design (a long-running theme in MAX). Some of our "Swiss-army-knife" tools contrast in their simplicity with those involved in other approaches to workflow design. We will have more to say about them in specific sections below, but here we might mention the lean quality of `HyperQueue` in its role of wrapping the functionality of the system scheduler, versus, e.g., the redesign of the latter in the `Flux` package of the US Exascale Computing Project.

At this point, it is fit to discuss why workflows are important in the context of exascale computing. Exascale machines offer extraordinary capabilities, but it is seldom that they can be exploited by stand-alone calculations, except in specific fields, such as weather forecasting, whose simulations can scale rather easily (and meaningfully) by reducing some grid spacing or similar quality parameters. In the field of materials, such a scaling is harder (even if growing steadily thanks to efforts like MAX; see e.g. report D1.2 for some recent technical achievements concerning scalability and deploy). Besides, often a single simulation is not nearly enough to shed light on a scientific problem, and explorations (of chemical space, of external conditions, of structural details) are essential. As shown in the scientific cases that are surveyed below, property calculations and simulations of complex phenomena are naturally framed in workflows involving up to very many, possibly large, tasks.

This would apply equally to non-exascale systems, but a second very important point is that exascale machines offer relief for the resource scarcity problem that is a key concern in workflow operation. Increased resources also mean the ability to hold a coherent store of data related to the workflows in a single place (thereby avoiding the need to move large amounts of data). When every aspect involved in computational science is taken into account, from design, to programming, to actual execution, the global performance (according e.g. to the "time-to-science" meta-metric) grows with the size and capabilities of the machine.

In the sections that follow, we will discuss the workflow-design tools and strategies we plan to rely on, and outline specific design patterns inspired by the scientific showcases. It should be stressed that the actual design work will be tightly coupled with developments in other work-packages: new code capabilities (in property calculation and in interfaces related to workflow operation) will be implemented in the context of WP1. WP3 will monitor and advise on developments in system software and perform benchmarks, and WP4 can use our experience in the co-design of system architecture and in the energy-optimisation of our workflows.

2 Review of scientific showcases

In the following, we provide a description of the main scientific themes addressed by the exascale workflows under development within the activities of WP2. In doing so, we stress both the scientific motivations and the technical requirements of the workflows.

2.1 Thermal conductivities

Understanding thermal transport is of fundamental interest in many areas of science, and also key in nanotechnology applications, including not only electronics but thermal insulation and energy harvesting. The main contribution to thermal conductivity in non-metals is due to the phonons.

- **Background and Motivation:**

  The two main approaches to the computation of thermal conductivity are based on molecular dynamics (either directly to obtain and analyse temperature profiles, or through the study of correlations), and the solution of the Boltzmann transport equation for phonons, using higher order force constants.

  After the seminal work by the SISSA group [1], the Green-Kubo method can be used with first-principles simulations: Coordinate and velocity snapshot data are used to compute thermal fluxes, whose autocorrelation can be analysed to compute the thermal conductivity.

  TDEP (Temperature-Dependent Effective Potential) is a method (and code implementation) to compute the thermal properties of materials, including the thermal conductivity, taking into account all the anharmonic effects that renormalise the force constants (including the higher-order ones) as a function of temperature.

- **Computational tools:**

  For the Green-Kubo method:

  – The first piece of the workflow is the generation of snapshots from a MD Siesta/QE run at a given temperature. The other piece involves a second code (QEHeat) that computes the thermal fluxes and stores their time series.

  – The data to be passed on to the code that computes the thermal fluxes can be simply the coordinates and velocities of each snapshot, but for maximum efficiency, electronic-structure information (ideally the wavefunctions) should be handed over as well. This would avoid the wasteful recomputation of the data needed. Wavefunction datasets are potentially huge. Ideally, they should be exchanged without using the (slow) filesystem. This could be possible through the use of a generalised master-slave setting, but this approach demands resource allocation and management features that might go beyond the capabilities of the scheduler. In addition, a coarser level of electronic-structure information (e.g. the charge density) can be used instead of wavefunctions.

  The TDEP method is based on the generation via molecular dynamics simulations at a given temperature of coordinate-force snapshots. (Alternatively, the structures

to be used can be generated iteratively through an stochastic procedure that mimics the canonical ensemble.) This information is used to fit an effective lattice potential in the form of a Taylor expansion in atomic displacements. The coefficients are then the renormalised force constants. The solution of the Boltzmann equation gives the desired properties.

  - The generation of configuration and force snapshots can be done with several independent SIESTA runs. It is particularly convenient to use the stochastic method to increase the parallelization possibilities.
  - The data exchange needs in this workflow are minimal: just the coordinates and forces of each snapshot.
  - Further parallelisation opportunities appear through the consideration of different temperatures, and of different compositions, defect conditions, and phases of the materials studied.

- **Relevance for exascale:**

  In the Green-Kubo method, system sizes (supercells) need to be large for the adequate convergence of the fluxes, and simulation times relatively long to extract the appropriate autocorrelation information.

  For the TDEP/Boltzmann workflow, the key is to guarantee convergence with a number of parameters: the quality of the effective Hamiltonian depends on the adequate size of the supercell used to compute the structure-force snapshots, and also on the range of force constants used in its construction. Both the fitting of the effective Hamiltonian and the computation of the thermal conductivity incur a substantial computational load when the number of force constants increases (this is particularly the case for low-symmetry systems). The quality of the Hamiltonian also crucially depends on the quality and number of snapshots used in the fit.

  The application of these methods to realistic systems demands very large computational resources.

2.2 Dynamics of nanoscale magnetism

Research in magnetism and magnetic materials nowadays if strongly focused on complex quasi-particle like magnetic objects like skyrmions. These objects are believed to form the basis of new paradigms in information storage and processing, able to revolutionise computing by enabling e.g. neuromorphic or reservoir computing.

- **Background and Motivation:** The simulation of such large magnetic objects is typically performed in a multi-scale fashion: first DFT simulations performed to evaluate interaction parameters that can then be fed into, e.g., an atomistic spin-lattice model to simulate the full system and also to model its dynamics. While this approach has demonstrated its usefulness, it has a serious limitation. The interplay between the magnetic and the electronic structure of the system is not captured in all cases correctly. In particular, if the magnitude of the magnetic moment changes significantly, the standard approaches are expected to fail.

A solution for such challenges is the use of a full ab-initio spin-dynamics. Here, the mapping of the DFT results is omitted, and the different spin configurations are directly evaluated in the DFT step. However, this approach faces two main obstacles in practice: (i) the phase space of magnetic configurations can be very large and hence many simulations have to be performed. As the magnetisation is a vector quantity, this problem is more pronounced than e.g. in a typical structural exploration. (ii) The magnetisation density and the charge density have to be converged together and as the energy scale associated with different magnetisation densities is much smaller than the energy changes originating from charge density rearrangement, the convergence behaviour of charge and magnetisation densities is very different.

- **Description of the scientific use-case:** We plan to simulate the stability and dynamical properties of skyrmions and similar magnetic objects. Starting from the structures obtained by atomistic spin models or continuum models, we will verify these results and investigate e.g. special points like Bloch-points which represent singularities in the continuum model. The details of the behaviour of the magnetisation at such points will play a crucial role in the stability of the magnetic objects, and hence we expect our simulations to provide valuable input with respect to the plans to use complex magnetic objects in future devices.

- **Computational tools:** Key to the simulations will be an automatic workflow that interlinks DFT simulations using the `FLEUR` code with atomistic models. In contrast to the standard approach, in which such a link is already established by the initial calculation of model parameters from `FLEUR`, we now need to augment the spin dynamics simulation with more complex results, like e.g. magnetic torques continuously during the spin-dynamics run. This will require a constant stream of DFT simulations on different magnetic structures to obtain adjusted results during the course of the spin dynamics run.

- **Relevance for exascale:** So far, the approach outlined here was hardly possible due to its significant computational requirements. The simple DFT simulations using `FLEUR` of the magnetic structures proposed here already required computational resources in the order of a GSC large scale project, as demonstrated in the past phase of the MAX CoE. Hence, the repeated simulations arising from the updates of the magnetic configuration as proposed in this use-case easily require the use of exascale machines.

2.3 Replacement of noble metals with cheaper materials for the catalysis of the (de-)hydrogenation reaction based on formic acid

In this industrial demonstrator, we propose a high-throughput study of alloys to discover candidate materials for the replacement of noble-metal nanoparticles in the catalysis of the (de-)hydrogenation reaction.

- **Background and Motivation**: Hydrogen-based fuel cells are currently one of the most promising candidates to replace fossil fuels as an energy vector, in order

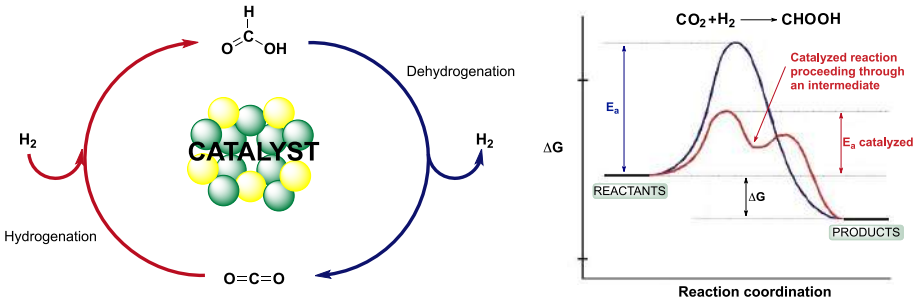

Figure 2: Scheme of the (de-)hydrogenation cycle with formic acid as a liquid organic hydrogen carrier (LOHC), left panel. Illustration of the energy barrier of a reaction in the absence (black line) and in the presence (red line) of a catalyst (right).

to meet the current targets in terms of reduction of CO_2 emissions and sustainability. Since pure molecular hydrogen requires high pressures and low temperatures to be stored safely, a strategy is the use of liquid organic hydrogen carriers (LOHC). These inert organic compounds, stable under ambient conditions, can bind molecules of H_2 and release them in a controlled manner. This process requires several cycles of hydrogenation and dehydrogenation reactions. From a scientific point of view, this investigation can shed light on the microscopic mechanisms of heterogeneous catalysis and allow us to obtain a "periodic table" of the catalytic elements for the (de-)hydrogenation reaction. From an industrial point of view, it enables the design and the production of new cheaper fuel-cells, based on commonly available materials, opening a new promising route for the decarbonisation of energy-intensive industries, such as those operating in the aerospace sector.

- **Description of the scientific use-case**: In this project, we focus on the use of formic acid as LOHC. Current technologies based on the use of formic acid and CO_2 for hydrogenation have low reaction rates for practical applications, unless a properly selected catalyst is used to speed up the reaction. High reaction rates were found using nanoparticles of noble metals to catalyse the reaction, such as silver (Ag), palladium (Pd) or a combination of the two [2]. However, these metals are extremely rare and expensive, so in practical applications it is desirable to replace them with nanoparticles made of more commonly available elements, such as common metals. In addition to its cost, a suitable catalyst should be stable, non-toxic, easily synthesised in the laboratory and efficient in decreasing the energy barrier of the rate-determining step. The proposed metals for this replacement are manganese, iron, nickel, cobalt, copper and molybdenum combined in different alloys. As a first proxy for the nanoparticles used in experiments, we will investigate binary surface alloys made of the aforementioned elements at different concentrations.

- **Computational tools**: As a first approach, we will develop a workflow that uses density functional calculations (DFT) to compute the total energy of the system of formic acid or CO_2+H_2 in the presence of each kind of catalyst. To this aim, we will use the `pw.x` code of the QUANTUM ESPRESSO distribution. First, we

will investigate the dynamical and thermodynamic stability of the catalytic materials at different concentrations and geometries, which will allow us to identify the possible configurations of the catalysts during an initial screening step. After that, for each candidate material, we will use the nudged elastic band (NEB) method to evaluate the reaction pathways and reaction yield. Additional approaches could then be used to accurately determine the reaction dynamics of the most promising candidates. For instance, we could use molecular dynamics to compute quantities commonly measured in experiments, such as the turnover number (TON) and turnover frequency (TOF).

- **Relevance for exascale**: The use of exascale machines for this use-case enables the study of a large space of possible compositions of the catalyst. Each simulation for the different compositions involves hundreds of atoms only to describe the surface or the nanoparticle. Additionally, each NEB simulation requires loosely coupled calculations that can be partially parallelized. The specific requirements in terms of number of calculations and computational effort for each of them is still under evaluation. The details of the high-throughput workflow and the specific requirements in terms of computing nodes are still under evaluation and discussion.

2.4 Electronic and optical properties of complex systems from Green's function methods

- **Background and Motivation:** Doped, defected and functionalised semiconductor surfaces are at the basis of the optimisation of photovoltaic and photocatalytic devices. For instance, it has been shown that the insertion of substitutional defects and the presence of oxygen vacancies in TiO_2 surfaces allows one to tune the gap of the material in order to optimise the absorption of the solar spectrum and increase its photocatalytic properties [3, 4]. Similarly, semiconductor surfaces functionalised with molecular dyes are the basis of the operation of dye-sensitised solar cells (DSSCs) [5]. Accurate and reliable simulations of the impact of doping and functionalization on the electronic and optical properties of materials are thus of great relevance. The modelling of materials in presence of defects or dopants requires the use of very large super-cells containing hundreds of atoms and thousands of electrons. This complexity makes first principles calculation of their electronic and optical properties, relying on many-body perturbation theory (MBPT) an exceptionally challenging task.

- **Description of the scientific use-case:** In this scientific demonstrator, we plan to study the structural and optical properties of hydroxylated $TiO_2(110)$ surfaces combining density functional theory (DFT) and many-body perturbation theory (MBPT) via the GW approximation and the solution of the Bethe-Salpeter equation (BSE) to include ladder diagrams. Titanium dioxide (TiO_2) is one of the most thoroughly investigated metal oxides, due to its broad range of uses in several key technologies including heterogeneous catalysis, photocatalysis, and solar energy conversion. All these applications are based on the generation of charge carriers by optical absorption, followed by their diffusion to the surface and transfer to adsorbed species. Recently, state selective studies using two-photon photoemission

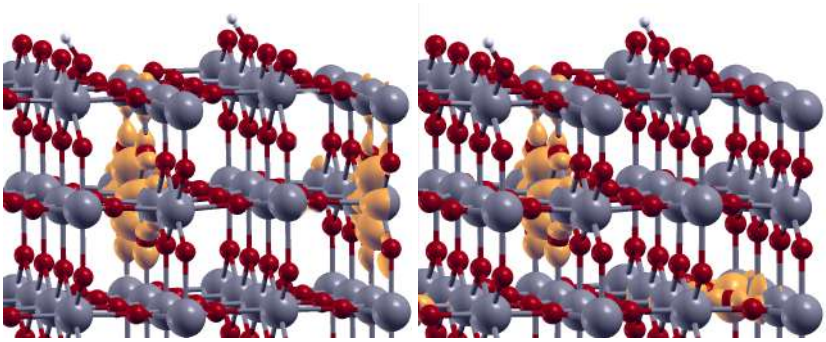

Figure 3: Ball-and-stick view of two unit cells of hydroxylated TiO_2(110) surfaces, each showing two polarons (here displayed according to their electronic isosurfaces) with different relative localisation. These cells are built as 4×2 reconstructions of TiO_2(110) with 4 trilayers, and contain 210 atoms (including passivating hydrogens at the bottom layer), and 1554 electrons.

(2PPE) spectroscopy have identified an additional photoexcitation mechanism at the surface (110) of rutile TiO_2. This process involves the excitation of polaronic excess electrons to conduction band states. We plan to study these excitations performing several BSE simulations exploring different structural configurations of the -OH groups at the surface and, importantly, different localization of the polarons.

- **Computational tools:** For each different polaron and dopant position, the simulation of optical properties can be divided into four steps: (i) calculation of the structural and electronic properties using DFT; (ii) calculation of the electronic screening; (iii) evaluation of the quasi-particle (QP) energies, and (iv) computation of the optical properties via the solution of the BSE. We plan to exploit the `pw.x` code of QUANTUM ESPRESSO to complete the step i, while all the remaining steps will be performed using YAMBO. Remarkably, we will make use of the `aiida-yambo` plugin implementing the data exchange from QE to YAMBO and the restarting logic in case of crash (see e.g. Refs. within [6]), that has been developed during the previous phases of MAX .

- **Relevance for exascale:** GW+BSE simulations for systems as large as those considered in this grand challenge (see figure above for examples of the unit cells to be adopted, each of them including 2 polarons) are currently extremely demanding. Moreover, they require converging a very large set of parameters, typically leading to multiple convergence runs to be performed to sample the accuracy of the results. However, this also means that there is a large phase space of parameters to be tuned and exploited. Following the experience gathered in recent years [6], this phase space can be optimised within a workflow, also taking care of sampling the different polaronic and dopant configurations. Overall, given the number of configurations to be sampled, and the large computational requirements of each MBPT simulation, in order to design this workflow we definitely need to target large scale, exascale-class runs.

2.5 Manipulation and control of coherent quantum states for quantum computing and photovoltaic applications

The goal of this scientific grand challenge is to describe the coupling of excitons and phonons (exc-ph) in complex materials in the non-equilibrium regime.

- **Background and Motivation:** Ultrafast non-equilibrium dynamics of matter and its constituents is at the basis of many technological applications. The manipulation of coherent manipulation of quantum states, e.g. **coherent excitons**, is a key requirement for quantum computing, which may revolutionise digital technologies. The interaction of excitons with other degrees of freedom, in particular **exciton-phonon coupling**, which might lead to exciton self-trapping, plays a fundamental role in the efficiency of many opto-electronic devices. Achieving control of these phenomena, however, requires a detailed understanding of the couplings between light, excitons, and phonons in many-body systems in- and out-of-equilibrium.

  First principles methods based on density functional theory (DFT), many-body perturbation theory (MBPT), and their time-dependent extensions have evolved into accurate, precise, and reliable tools for modelling optoelectronic and non-equilibrium phenomena. The ab-initio description of the coupling between excitons and phonons in systems in- and out-of-equilibrium, however, remains a major shortcoming of these techniques, which hinders their application to describe complex phenomena in ultrafast science as, e.g., light-induced phase transitions.

- **Description of the scientific use-case:** 2D materials based on transition metal dichalcogenides (TMDCs), such as MoS_2, are promising candidates for various optoelectronics devices, such as high-efficiency quantum emitters [7], based on injected excitonic states. At low injected density, exc-ph coupling is the key dissipation mechanism for exciton dynamics [8], and it constitutes the key to the formulation of novel technology concepts based on quantum materials [9]. In this workflow, we will compute exciton-phonon coupling in TMDCs, starting from monolayer MoS_2. Once the demonstration is ready, the same approach can be used to study more challenging and technologically relevant systems, such as heterostructures based on TMDCs or perovskites for solar cell application.

- **Computational tools:** The workflow is based on running finite momentum Bethe-Salpeter (BSE) and density functional perturbation theory (DFPT) simulations. This approach has the extra advantage of capturing the coupling between phonons and excitons at all finite momenta. For a more detailed discussion on different approaches to exc-ph see Sec. 4.4.8. The BSE at finite momentum is implemented in YAMBO, and has undergone extensive optimisation in previous rounds of MAX, and has proven suitable for exploiting large scale HPC systems. The DFTP scheme is implemented in `ph.x` of the QE suite, and it has also been optimised in the last two years. The coupling of the output of the two runs is performed via a novel algorithm recently implemented in the YAMBO code.

- **Relevance for exascale:** The calculation of exc-ph couplings and lifetimes requires a very large set of both BSE and DFPT calculations, fitting naturally into a complex exascale workflow (see Fig. 4). Moreover, all these calculations needs

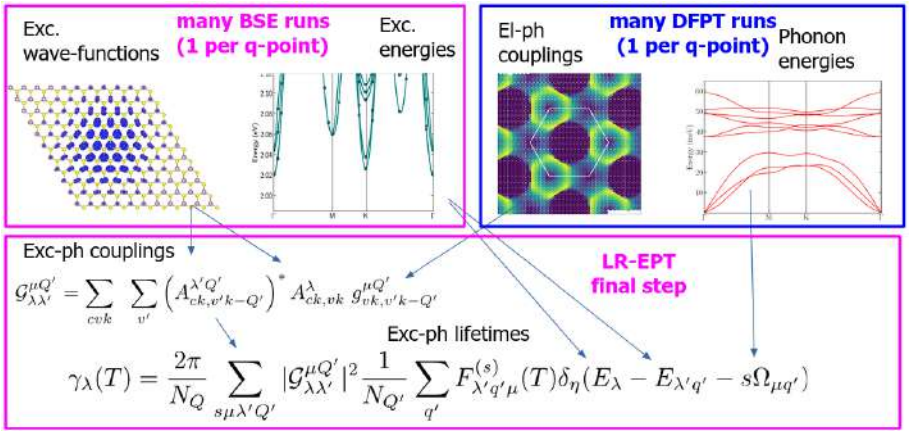

Figure 4: Representation of the different calculations involved in the evaluation of the exciton-phonon lifetimes from first-principles. Namely, exciton dispersions, phonons, and electron-phonon coupling matrix elements needs to be combined to compute the ultimate exciton-phonon lifetime.

to be combined in a complex workflow which involves (at least) two codes, thus also requiring to address non-trivial interoperability issues and data exchange. For further details on the exc-ph workflow, see Sec. 4.4.8.

2.6 Quantum-Mechanics derived indicators to characterise Protein-Protein binding in Antibody/Antigen (AA) assemblies

Our project aims at identifying the microscopic interactions that favour/disfavour the stability of AA assemblies by means of a hybrid (sequential) multi-scale Molecular Modelling/Quantum Mechanical (mMM/QM) simulation protocol.

- **Background and Motivation:** Monoclonal antibodies targeting specific antigens are promising biochemical macro-molecules for therapeutic purposes. However, whereas targeting specific monoclonal antibodies can be routinely achieved, increasing the antigen affinity as much as desired is still a challenging task. Most of the available theoretical methods devoted to that task mostly focus on investigating close contact antibody/antigen (AA) local regions and usually ignore the effect on affinity of more distant domains. Because of the atomic size of AA complexes (from 500 to 1000 protein residues and above) only standard pairwise molecular modelling force fields and more likely empirical cost functions are used to quantify the strength of the interactions within such complexes. However, these theoretical approaches are known to be based on crude approximations, preventing to reach a high level of accuracy in estimating the stability of AA assemblies. Note also that despite the recent huge progress in predicting 3D protein structures at the atomic level using AI techniques, accurate investigations of the origin of the stability of protein assemblies like AAs is still an ambitious goal.

- **Computational tools:** Our multi-method workflow relies on a QM method based on the BIGDFT code and a mMM approach (based on polarizable force fields cou-

pled with an O(N) polarizable multi-scale coarse grained water model[1]) that are able to efficiently simulate on modern supercomputing systems microscopic systems up to tens of thousands atoms, and at the million atom scale, respectively. The mMM approach is used to sample the microscopic potential energy surface (PES) of the system (using a Hamiltonian Replica Exchange, HRE, see e.g. Ref. [10]).

Instead, QM is used to further analyse the microscopic factors modulating the strength of the assembly interaction by investigating a set of representative assembly structures identified from the mMM step. The latter structure set is built by clustering the main mMM(HRE) simulation and the main (more abundant) clusters are then post processed using QM, in particular to identify the pairs of assembly fragments (usually a residue or a small set of residue) that have a strong effect on the assembly stability. The final quantum data from the QM fragmentation phase correspond thus to averages of data from the most abundant clusters. We refer to this approach as the QM/CR approach, which stands for "Complexity Reduction" framework.

- **Description of the scientific use-case:** During the last year, we have systematically applied our mMM/QM approach to a portion of protein/protein assemblies (about 3 thousand structures presently) whose dissociation constants KD's and three-dimensional atomic structures are available at the PDBbind database[2] and at the Protein Data Bank (PDB), respectively. The aim is to build a database of interaction data (termed as Interaction Network Plots, INP's, see below) to understand the microscopic factors modulating the strength of the association of proteins and from which to provide to experimentalists reliable solutions to optimise new protein/protein assemblies, like AA ones. Precisely our goal is to add two new steps, the mMM/HRE + QM/CR methods as described above, to standard computational protocols used to model AA complexes, in order to further evaluate and refine their solutions.

- **Relevance for exascale:** This workflow is, essentially, a on-the-fly curated approach for High Throughput Calculations of very large DFT systems, where the workflow orchestrator (in our case the `remotemanager` approach, see below) has a prominent role. The investigation of the PDBbind dataset which has been stating this year has enabled us to elaborate a reliable full performance model for each of the PDBbind items. The investigation time can be estimated, on a x86 architecture (we used the CEA Irene machine for the purpose) to be about 25k CPU hours per protein-protein assembly. Stated otherwise, the investigation of some 3 hundred AA structures would cost about 15 Million hours, which can be consumed quite rapidly as each of the structure can be processed independently (yet, for each structure an mMM/HRE model has to be implemented). In a exascale machine, this would enable us to extract relevant QM/CR information for novel AA structures, including potential candidates for optimisation (mutations).

A demonstrator of this protocol has been published this year in a paper [11] to investigate the microscopic factors at the origin of the stability of SARS-COV-2 main protease/-

[1] For further details see http://biodev.cea.fr/polaris/
[2] http://www.pdbbind.org.cn

ketoamide inhibitor complexes. Work is ongoing to relate the QM/CR results to the experimental dissociation constants.

2.7 High throughput screening and characterisation of Li-ion batteries

- **Background and Motivations:** Lithium-ion batteries have emerged as pivotal components in modern technology, playing a critical role in powering a myriad of devices, ranging from smartphones and laptops to electric vehicles. As the world increasingly shifts towards renewable energy and electric mobility, the demand for lithium-ion batteries continues to grow, underscoring their crucial role in shaping the future of clean and efficient energy storage.

- **Description of the scientific use-case:** We propose important scientific investigations for advancing new and sophisticated battery materials for potential industrial applications, having as primary objectives: (i) the discovery of new cathode materials and (ii) the exploration of their operational mechanisms (e.g. calculation of charge/discharge rates).

  The most important outcomes will include: (i) a database of accurate Hubbard parameters, (ii) a database of innovative Li-ion cathode materials characterised by enhanced electrochemical properties with potential for commercialisation, (iii) new equivariant machine learning (ML) models for predicting HPs and oxidation states of transition-metal elements.

- **Computational tools:** The workflows of interest will be devoted to the automatic calculation of HPs (AiiDA-HP), management of the molecular dynamics simulations, and interface with state-of-the-art machine learning techniques. AiiDA will be used to orchestrate the calculations with Quantum ESPRESSO (`pw.x` and `hp.x` for the AIMD and HP calculations). The AiiDA-HP workflow is under actual development and some of them will be defined and discussed based on the actual scientific need.

- **Relevance for exascale:** The material discovery phase involves the screening of thousands of lithium-containing materials and the calculations of the relevant parameters for accurate modelling, i.e. the Hubbard parameters (HP) for transition-metals. On the other hand, the study of charge/discharge rates needs the use of ab-initio molecular dynamics (AIMD) simulations. The extensive scope of materials screening, the high computational cost of computing high-accuracy HPs and the AIMD require the utmost computational power and parallel processing capabilities, ideally offered by exascale systems.

2.8 Modeling non-adiabatic transitions in molecular systems

- **Background and Motivations:** Non-adiabatic molecular dynamics (NAMD) plays a crucial role in understanding and simulating complex chemical processes that involve rapid transitions between electronic states. The Tully surface hopping algorithm [12, 13, 14] provides an effective framework for modelling non-adiabatic transitions in molecular systems. This algorithm allows for the simulation of electronic state dynamics by incorporating quantum mechanical effects, such as

non-adiabatic transitions between different potential energy surfaces. An exascale-capable NAMD framework can have many diverse applications [15]:

  - **Photochemical Reactions:** Investigating processes such as photoisomerisation, photodissociation, and other light-induced reactions to gain insights into energy transfer mechanisms and reaction pathways.
  - **Charge Transfer Processes:** Examining electron or hole transfer dynamics in donor-acceptor systems is crucial for designing organic electronics, solar cells, and other optoelectronic devices.
  - **Chemical Reactions in Solutions:** Simulating reactions in solvents to understand the impact of the surrounding environment on reaction rates, pathways, and product distributions.
  - **Excited State Dynamics:** exploring the dynamics of excited states, including non-radiative decay processes, internal conversion, and intersystem crossing, essential for understanding fluorescence, phosphorescence, and other photophysical phenomena.
  - **Catalysis and reaction mechanisms:** Studying how catalysts facilitate chemical reactions by examining the involvement of multiple electronic states, providing insights into reaction pathways, intermediates, and rate-determining steps.
  - **Quantum Computing and Information Processing:** Investigating coherence and entanglement in molecular systems as potential components for quantum computing and quantum information processing.

- **Description of the scientific use-case:** Our demonstrative application will aim at the development of an exascale workflow for simulating large supramolecular bio-aggregates, relevant for research on **neurodegenerative diseases such as Alzheimer's and Parkinson** [16, 17]. In particular, one of the grand challenges in imaging biological molecules is devising non-invasive and non-toxic probes that allow for identifying molecular interactions. In the last decade, the use of non-aromatic fluorescence has emerged as a potential candidate in this regard, where it has been observed that proteins lacking any aromatic or conjugated groups display weak fluorescence upon forming aggregated states.

  The photochemistry and photophysics underlying these phenomena remain poorly understood and are critical information needed to realise better probes for these neurodegenerative diseases.

- **Computational tools:** NAMD computations involve several steps: for a given system geometry, the electronic ground-state needs to be calculated. In a second step, a number of excited electronic states are needed as well as the so-called non-adiabatic couplings between those states and the derivative of their energies with respect to nuclear coordinates. Given this information, the (excited-state) forces as well as the hopping probabilities can be calculated. With this information, the dynamics can be advanced by one time-step.

  The `pw.x` program of the QUANTUM ESPRESSO suite can handle the ground-state part of the calculations "out of the box", even with state-of-the-art hybrid

functionals and in large systems, being already adapted to work on massively parallel computer architectures. The second step, involving excited states calculations, can be addressed within the framework of time-dependent DFT (TD-DFT), implemented in the `turbo-davidson.x` code of QUANTUM ESPRESSO. Alternatively, the calculation of the potential energy surfaces for excited states can be performed using the GW+Bethe-Salpeter approach of YAMBO.

- **Relevance for exascale:** The size of the molecular system under study is relevant because excitation and fluorescence properties depend on the whole supramolecular aggregate rather than on the single individual components. Moreover, since the time evolution of the species can split into different branches (e.g. a photoexcited intermediate can either detach or react with a catalyst surface), and the splitting might occur, in principle, at any point of each trajectory, the simulation runs cannot be just computed completely independently of each other, but rather they must communicate data at runtime. In addition, several such surface-hopping trajectories are needed in parallel also to account for the statistical nature of the transfer processes.

  Exascale-class computational power is then expected to be a crucial asset, as it allows for the computational treatment of the entire system, including all relevant chemical interactions, using accurate methods.

3 Tools for workflow deployment

This section provides first summary descriptions of those workflow tools (some but not all developed within the MAX consortium) that we find particularly useful to the project. These are a very small fraction of those available in the community, but their modularity and flexibility are specially desirable characteristics for our design work. The last subsection mentions some other tools which might bring other functionality or inspiration to our work.

3.1 Meta-schedulers — HyperQueue

`HyperQueue`[3] (HQ) is a specialised job scheduler tailored for high-performance computing (HPC) environments that enhances the efficiency, execution, and scalability of computational tasks. HQ integrates smoothly with prevalent workload management systems, such as SLURM and PBS, facilitating distributed task execution without compatibility issues.

A prominent advantage of `HyperQueue` is its resource management prowess. HQ offers dynamic load balancing, which streamlines the distribution of tasks. Users are not required to pre-configure resources (e.g., allocating 16 CPUs) for specific tasks or manually bundle tasks into larger computational units. Instead, users simply initiate a worker node furnished with requisite resources, submit tasks, and HQ dynamically assigns those resources as needed. Additionally, workers need not be launched by hand; HQ possesses the capability to autonomously initiate SLURM or PBS jobs on demand.

[3] https://github.com/It4innovations/hyperqueue/

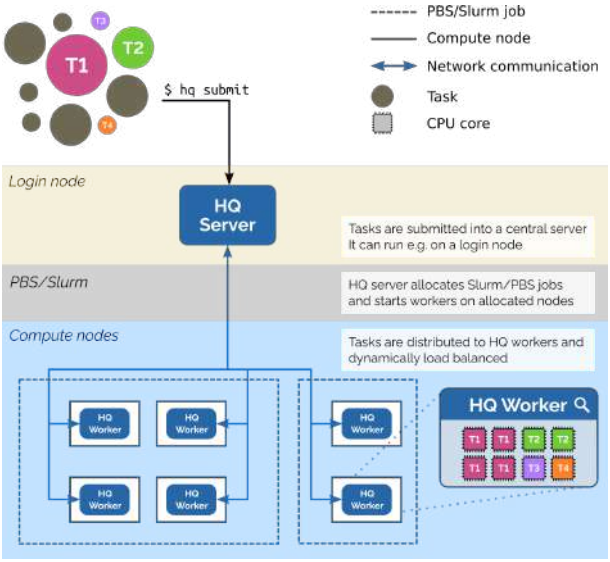

Figure 5: Sketch of the main elements and operation of `HyperQueue`.

Performance-wise, HQ is adept at managing sizeable HPC infrastructures, scaling across potentially hundreds of nodes and workers while imposing a negligible overhead, maintained below 0.1 milliseconds per task. Moreover, HQ is easy to deploy, as it comes as a single, statically linked binary, eliminating complex dependencies other than `libc`. This approach simplifies the installation process. HQ does not need administrative access to the cluster, thus making it accessible for a broader range of use cases. HQ use has been successfully demonstrated at large facilities such as LUMI [4] and is being deployed in other EuroHPC facilities.[5] In context of MAX, HQ offers the following features:

- Workers can be dynamically created and shutdown, thus allowing for efficient allocations.
- Task execution can be optimised based on resources, including virtual resources that assign specific kinds of workers to appropriately tagged tasks.
- Dependencies: Tasks may have defined dependencies that are essential for computing DAG workflows

Further developments envisaged:

- Check-pointing of server state to increase resiliency of server and allow for continuity of context in long-running workflows.
- Data-handling layer that leverages node resources to offer an abstraction of a unified data space, with transparent migration when needed.
- A more fine-grained dependency model that allows for signalling that "data is available" even when the task itself is still running and may potentially produce more data.

[4] See: https://www.lumi-supercomputer.eu/hyperqueue-facilitates-bett...
[5] See e.g. https://docs.csc.fi/apps/hyperqueue/ for another showcase of deployment.

3.2 Workflow managers and deployment interfaces

3.2.1 AiiDA

As stated in the proposal, a very successful platform for workflow development, based on the AiiDA engine and code-specific plugins, was developed and used extensively during MAX -2, mostly for high-throughput computing and the associated high-performance data analytics. The platform proved its scalability by a run on the entire LUMI-C partition of 196608 AMD Epyc cores, coordinating 55704 Quantum ESPRESSO calculations for 15324 inorganic compounds, with essentially 100% utilisation over 12 hours.

AiiDA workflows exploit the core features of the platform, plus specific code plugins that encapsulate the functionality of the codes and expose their basic input/output in the form of AiiDA-compatible data structures (e.g. a structure object, a molecular-dynamics data object, etc). In addition, AiiDA maintains full provenance of the calculations carried out by storing all relevant nodes in a dedicated database.

Beyond high-throughput computing of relatively small calculations which depend on one another only through relatively modest data dependencies, AiiDA can still be useful as an orchestration tool for exascale workflows with some relatively minor extensions and modifications:

- Addition of an interface to code performance models to inject resource options at each relevant step, currently specified manually.
- Extension of the launching semantics, leveraging the already existing HyperQueue interface, to provide the abstraction of "worker" spawning, in addition to and matched to "job launching".
- Refocus of the provenance tracking to consider only the higher level of the workflow(s) (e.g. top-level inputs and summary outputs).
- Refactoring of code plugins to enable the option of "task/data offloading" (i.e. by hiding from the interface those IO channels that are internally handled by tools like ZeroMQ).
- Extension of the "RemoteData" abstraction to allow for data migration among computing resources (using a planned data-handling feature of HyperQueue).

The underlying idea is to enhance the flexibility of the platform and reduce the overhead associated to the provenance handling, while maintaining the very useful plugin concept as a key feature for extensibility and modularity. AiiDA's built in workflow-specification language will retain its basic constructs and idioms. These have been exercised in production by a number of workflows involving QUANTUM ESPRESSO, YAMBO, FLEUR, and SIESTA.

An issue that should be discussed in this context is the practical problem of execution of AiiDA workflows. In some computer centres (e.g. Leonardo) two-factor-authentication schemes complicate the mechanics of remote execution of AiiDA workflow tasks. When offered the alternative of running the AiiDA instance on the actual HPC machine, its workings need to be "certified" for compliance with security measures. While we understand the need to maintain an adequate level of security, we encourage EuroHPC to help us find a balance with usability.

3.2.2 RemoteManager

This tool is currently used by the BIGDFT team. It is nevertheless a general utility tool that focuses on the deployment and (remote) execution aspects of the workflow, as a lighter alternative to the corresponding AiiDA module. It also features "front-end" capabilities linked to notebooks.

At the beginning of the MAX-2 project, a significant weakness of the initial `PyBigDFT` implementation was that the writing of workflows would be interrupted by the need to run computationally heavy calculations on remote machines. If a workflow was being written in a Jupyter notebook, one would need to convert that notebook to a script, submit it through the job system, wait for the calculations to complete, copy data to the machine with the Jupyter server, and then restart the notebook (exploiting lazy calculations). Computationally demanding calculations include not just standard BIGDFT runs, but a wide variety of pre- and post-processing operations. These requirements necessitate a strategy for driving the execution of *arbitrary* Python routines on (potentially multiple) remote machines.

To fulfil this requirement, we have developed a new library called `remotemanager` (List. 1). This library is able to serialise arbitrary Python routines and arguments that are sent to a remote machine and executed through a job system, after which the resulting data are synchronised to the local machine. For simple data-types, serialisation is done with YAML. For more complex data-types, external libraries like Dill or JSONPickle can be selected. To use `remotemanager`, a user defines the function they want to run remotely, and uses it to create a `Dataset`. Sets of arguments are then added to the `Dataset` and the whole set can be run asynchronously. If auxiliary functions are required, they can be defined with the `@RemoteFunction` decorator. The `remotemanager` package follows the model of BigDFT's `SystemCalculator` in that a `skip` argument will check if the calculation has already completed, and if so, no calculation is performed. We also note that `remotemanager` has been proven to be completely compatible with two-factor authentication methods, which become mandatory in the emerging platforms.

The `remotemanager` library does not use a daemon on the remote machine and only needs to be installed on the user's workstation. Collections of remote calls can be bundled together into the `Dataset` type, which is optimised to minimise commands sent to the remote machine (a feature that helps in high latency environments). Connections are defined (List. 2) via the `URL` module: a Python wrapper around the Secure Shell Protocol (SSH). On top of the `URL` class is the `Computer` derived type and associated derived types for various specific machines. Each computer type is defined with information about how to generate job scripts and the associated options (number of nodes, maximum wall time, etc). Front end versions of a machine can also be defined for low-cost operations. A user planning to access a new machine can either define a new `Computer` derived type or define one using a YAML configure file.

Code development inside virtual notebooks often differs substantially from best practices in software engineering. As noted above, developers frequently use notebooks for exploratory calculations instead of building well-structured workflows. The `remotemanager` library can work well under these circumstances (List. 3). Entire Jupyter cells can be designated to run remotely using the "cell magics" included with `remotemanager` (keyword `sanzu`). When other literate programming environments are used, we also provide

```python
# Function to run on the remote machine
def remote_function(sys, inp, run_name):
    from BigDFT.Calculators import SystemCalculator
    calc = SystemCalculator()
    log = calc.run(sys=sys, input=inp, name=run_name)
    return log.energy

# Create a Dataset to store the function
remote = Dataset(function = remote_function,
                 url = url,
                 serialiser = serialdill())

# Append a run with data
args = {"sys": sys, "inp": inp, "run_name": "remote"}
remote.append_run(arguments = args)

remote.run()  # submit the calculation

# Wait for job completion, checking every 10 seconds
remote.wait(interval=10, timeout=3600)

# Fetch and display the results
remote.fetch_results()
print(f'energy: {remote.results}')
```

Listing 1: An example of a remotely submitted calculation. First, we define a function that we wish to run remotely. The function is given to the `Dataset` object along with a `URL` object that describes the remote computer. The arguments to the function are given to the `append_run` method. Finally, the run method serialises everything, submits a calculation asynchronously, waits for the calculation to complete, and fetches the result.

```python
# Basic Url
url = URL(user="username", host="remote.host.address")
url.test_connection()

# YAML Configured
url = BaseComputer.from_yaml("summer.yaml", user="username")

# Or Derived Type Computer
url = Fugaku(user="username")

# Computer Configuration
url.mpi = 48
url.omp = 12
url.nodeshape = "torus"  # Machine Specific
```

Listing 2: Definition of the `URL` base class and predefined `Computer` types. A `URL` can be defined as a simple remote address, with a configuration file, or as a derived type when complex logic is required for job script generation and submission. After a `URL` object is constructed, global job parameters can be set.

```
[Cell 1: Run Dataset]
for name, sys in systems.items():
    args = {"sys": sys, "inp": inp, "run_name": name}
    remote.append_run(arguments = args)
remote.run(); remote.wait() ; remote.fetch_results()

[Cell 2: Process The Results]
energies = {k: v for k, v in zip(systems, remote.results)}
mink = min(energies, key=energies.get)
minsys = systems[mink]

[Cell 3: Magic Single Job]
%%sanzu url=url, serialiser=sdill
%%sargs sys=minsys, inp=inp, run_name=mink
from BigDFT.Calculators import SystemCalculator
inp.read_orbitals_from_disk()
inp.write_cubefiles_around_fermi_level()
calc = SystemCalculator()
log = calc.run(sys=sys, input=inp, name=run_name)
log.energy
```

Listing 3: Jupyter cell executed remotely with cell magics (`sanzu`). After a set of systems is processed, a single post-processing operation is defined, which can be run without the `Dataset` boilerplate. The `sargs` delimiter is used to indicate arguments to the function executed remotely. Return values, printed data, and errors appear in the notebook as if the cell was run locally.

the `@SanzuFunction` decorator to transform a function such that it is always executed remotely and synchronously. Thus, we envision `remotemanager` as a general library for performing Remote Procedure Call on HPC systems, which enables explorations that may have large computational requirements.

The intermingling of text and code in literate programming make notebooks a perfect tool for collaborative efforts. The `remotemanager` framework can enable such collaborative use and "Computational Continuity": allowing users to hand off notebooks and serialised results to their coworkers, who can then add to the notebook and generate further results without having to redo computationally demanding portions. Another notion that follow the API definition of those two libraries, is the concept of *Intentional Programming*; the reader of the notebook should have no difficulty in understanding what each programming line stands for, and the resulting readability of the notebook source code would then ease the concept of continuity. In other terms, those libraries are standing between the "scientist ideas" and the actual computational implementation of them, hiding the somehow tedious and contentless layers which have to be often explicitly manipulated by the researchers.

3.3 Communications and data-handling libraries and tools

3.3.1 ZeroMQ as a tool for asynchronous task distribution and data handling

`ZeroMQ` [18] is a communications library built on the familiar socket concept but focused on robustness and flexibility. It provides strong guarantees (such as atomicity of message delivery) that make it a good foundation for the design of general distributed systems. In the context of MAX it can be used, among other things, for task distribution

and asynchronous data handling. As an example, consider the following pseudo-code for a process ("producer") executing a molecular dynamics (MD) simulation:

```
MD: do
    compute forces
    move atoms
    ! data for calculation "downstream"
    send_packet(data=x,v)
enddo
```

and for another process ("analyser/consumer") computing a certain property using the data generated by the first process:

```
PROP: do
     read_packet(data=x,v)
     compute_property(x,v, prop)
enddo
```

The `send_packet` and `read_packet` operations are (wrapped) calls to the `ZeroMQ` library, which by means of a proxy process connects both processes. The data is transferred asynchronously, so that the producer process can continue the simulation in the next iteration of the loop, and the consumer process keeps receiving data and computing with it. This is just (very useful) data exchange for different processes, which might be running on different nodes, with different, tailored architectures or computational parameters, according to the characteristics of the tasks involved. Note also that the changes to the producer code to enable this feature are minimal. The same paradigm, with a further very minor change, turns the mainly serial workflow above into a distributed, parallel, dynamic workflow. Consider a "one-off" consumer process:

```
program consumer
     read_packet(data=x,v)
     compute_property(x,v, prop)
end program
```

We have just eliminated the previous loop. Now, imagine that (perhaps using `HyperQueue`) we launch several copies of the consumer program, all connected to the same proxy that is handling the data stream from the producer process. Then, transparently, consumers will (atomically) receive full packets from the proxy and work on them, in parallel. By tuning the number of consumers according to the throughput of the producer and the computation time of the property, we can get optimum performance. Since the processes are all independent, they can be launched with the appropriate number of nodes, GPUs, and other computational parameters to maximise the return (in terms of time-to-solution, or allocation cost, or energy).

This setup uses one of the built-in communication patterns in `ZeroMQ`. The library can be used through a Fortran interface (`fzmq`), on top of which Riccardo Bertossa from SISSA added wrappers to enable the sending of Fortran arrays and more general data packets. This approach was demonstrated as a proof-of-concept in the context of one of the demonstrators of the second phase of MAX.[6] Its strengths are: simplicity, since almost no code changes are needed, just a basic interface to the library (see section on code

[6] See Deliverable D6.3 of MAX phase-2.

hooks); flexibility, as codes can be hooked together in various ways to enable appropriate workflows; leanness: there is in principle no need to move data through the file system.

As mentioned in the Introduction, these strengths are predicated on the execution under ideal conditions. In particular, the last one (data is exchanged once through the network and the packet can be discarded once the handshake is over) does not play well in case of consumer failure: that particular task and its data would be lost to the workflow. However `ZeroMQ` makes it very easy to add further elements in the distributed workflow (such as a holding "post-office") that can serve to cover for certain kinds of failures (and congestion).

One additional note: the interface to `ZeroMQ` in the code is still roughly semantically equivalent to a "file interface". This further exemplifies the modularity of our approach, highlighted in the Introduction. If and when the demands of a workflow preclude its use, the same abstractions could be employed with the kinds of tools described in the next section.

3.3.2 Other tools with similar semantics.

While the data exchange via the filesystem is still the easiest to understand and use, special demands might make it necessary to resort to other specialised tools, and eventually prepare interfaces which are semantically similar. Here we provide a tentative list of tools that could be of use within MAX :

- A *burst buffer* serves as an intermediary storage layer positioned between the front-end computing processes and the back-end storage systems within high-performance computing environments. Its primary function is to bridge the performance disparity that often exists between the processing speed of compute nodes and the input/output (I/O) bandwidth of storage systems. Typically composed of high-performance storage devices like NVRAM and SSD, a burst buffer delivers I/O bandwidth one to two orders of magnitude higher than that of the back-end storage systems. By buffering data, especially through the use of NVRAM and SSD arrays, burst buffers significantly accelerate scientific data movement on supercomputers. This buffering approach provides applications with opportunities to optimise data traffic to the back-end storage systems, thereby ensuring efficient utilisation of storage system bandwidth. Furthermore, by bypassing the traditional storage systems, applications can maximize the performance benefits offered by the burst buffer.

- `BeeOND` (BeeGFS On Demand) [19] is a very simple and flexible ad-hoc filesystem that is active for the duration of a simulation, and can be leveraged as a fast-store for initial staging of data.

- `Apache Cassandra` [20] is a free and open-source, distributed, wide-column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. This is a relatively sophisticated tool that must be launched and active during workflow operation, thereby complicating the workflow, but it is very useful when data is not produced and accessed in "stream mode" from different parts of the workflow (e.g. pieces might need to be processed out-of-order).

3.4 Code-level interoperability hooks

If we consider the codes (in particular our flagship codes) as the basis elements of a workflow, it is clear that there must be mechanisms and special features for their integration: for data exchange, flow of control, and signals and message passing to log their operation and initiate recovery actions if needed. Some of the mechanisms are dull and tiresome (e.g. checkpointing), others are more attractive (e.g. `ZeroMQ` and similar interfaces), but all are needed to deploy robust and flexible workflows. This section describes the basic needs that have been identified, and their implementation in codes is expected to be worked on in collaboration with WP1.

- **Checkpointing**. This is primarily employed to furnish fault tolerance to applications. In this scenario, the entirety of the application's state is routinely stored in a stable medium, such as a disk, and can be retrieved if the original application crashes due to a failure in the underlying system. Subsequently, the application is restarted (or recovered) from the latest checkpoint, minimising the time lost as a result of the failure.

  While checkpointing is advantageous for various computations, it assumes a distinct significance for parallel applications and workflows at large scales. As the number of processors used increases, the overall system availability diminishes, underscoring the importance of reliable fault tolerance mechanisms. The challenge is generating globally consistent checkpoints throughout the entire application or workflow. At the application level, hierarchical protocols must be used to balance the cost of checkpointing with the cost of restarting from scratch. A typical example is the saving of some data structure representative of the current state of the electronic-structure computation (i.e. the density matrix or the charge density, or (more expensively) wavefunctions). Workflow-level checkpointing is not managed by the codes, but these must be able to provide adequate signals to the workflow orchestrator.

- **Performance models.** Task allocation and execution would be improved if a good prediction of the resources needed, and of the execution time, is available. For example, `HyperQueue` could be able to pack extra tasks in a worker, or a workflow-level decision could be made regarding the optimal ratio of producers and consumers. The creation of a performance model for a given code (or a module within a code) is a non-trivial task, but one that can repay handsomely.

- **ZeroMQ and other interfaces.** We have already mentioned that a Fortran interface to `ZeroMQ` is available. Further wrapping for data encapsulation is straightforward. At the same time, a higher-level abstraction is possible, to include similar functionality with "file interfaces". In this case, it is important to ensure that file system operations are atomic, i.e. they cannot be interrupted or "partially" performed. Either the entire operation is performed or the operation fails. The final objective is to provide an abstraction for data handling (including in some cases, as we have seen, task distribution) that can be ported to various systems. A library including these interfaces can be written once and for all and be reused by various codes.

- **Logs and actionable error reporting.** These are essential for proper recovery. Since reliability must be implemented with reference to specific failure cases, it is important to categorise these and report them (when possible). Obviously, in this category we can cover the failure cases which pertain to the code itself (lack of convergence of some step, insufficient memory, etc). Failures in system software and hardware need to be detected at the workflow level.

- **Other ways to distribute and execute tasks.** APIs can be used to expose the functionality of a code to other applications running on the same computer or on other computers over a network. The MaX flagship codes already offer APIs with various degrees of sophistication. For example, Siesta can be run in "server mode" using a socket-based protocol. It can also be treated like a "subroutine" of another program, via the linking of a library and the use of MPI communicators to separate the data spaces of different instances. The latter restriction is typical of codes with a long history that might still have a significant number of global variables. By removing these variables and keeping the "state" of the computation hidden behind an abstract handle, a more flexible API can be designed. This is a long-term effort.

  AiiDA plugins and file interfaces (as used, for example, by the Atomic Simulation Environment (ASE)) are other ways to expose the functionality of a code, and have been designed with workflows in mind.

  In all these cases, a higher-level script, notebook, or master MPI-enabled application takes care of the orchestration of the tasks.

Another classification of the above points is afforded by considering some key elements of workflow operation:

- **Task distribution.** Here we have already seen the example of use of `ZeroMQ` or similar schemes to asynchronously hand out tasks and the associated data. This is arguably a very flexible approach that, when coupled with the dynamic allocation and execution facilitated by `HyperQueue`, can maximise throughput and, in harder conditions, adapt better to the constraints.

  Other options include the explicit handling of the tasks within an overarching MPI framework. The best-known example is the master-slave model, in which certain ranks are devoted to coordination and data channelling, while others are assigned to computational tasks. This is a reasonable model for fixed-structure and near-fixed-demands workflows, but not for more dynamic workflows. It needs a pre-allocation of resources, which might prove insufficient, or get wasted, depending on the stage of the workflow and the actual computational load. Heuristics and performance models can only get so far in this case. Moreover, from the point of view of a code, this model demands substantial changes in the exposed interfaces: it must be able to be run as a subordinate process in a split MPI communicator. While some codes offer this kind of API, this is not yet widespread.

  In the `AiiDA` paradigm, codes are wrapped as plugins (a one-time effort), and a workflow manager (a "spec") follows the DAG, creating tasks and launching them asynchronously. This is ideally suited to high-throughput computing, notably when using `HyperQueue` as a backend handler to pack jobs into workers and reduce

the load on the scheduler. A downside for workflows with larger data demands is that the `RemoteData` mechanism is fragile in the face of ephemeral working directories and node reallocation.

- **Composability** is another key aspect of the workflow concept. Existing workflows can (and typically should be) combined into larger workflows. As it would be impractical to demand a unified task distribution model for all the pieces, it is clear that extra glue will be needed to operate the larger workflow. (The `AiiDA` model, to some extent, is an attempt to formalise a unified scheme, although plugins can encapsulate arbitrary policies). For example, `AiiDA` might be used in some specific parts of a workflow, but others might use the socket-based scheme, or the master-slave concept. A scripting language or notebook interface can provide the extra glue.

3.5 Other tools from the wider workflow community

As mentioned in the Introduction, the landscape of workflow tools is exceptionally diverse, reflecting the multifaceted nature of scientific research and its distinct needs, ranging from data preprocessing to complex simulations and analyses. Researchers can choose from an array of workflow management systems, each designed to streamline and optimise specific aspects of computational workflows. There are tools specifically tailored to high-performance computing environments, ensuring efficient utilisation of resources for computationally intensive tasks.

It is not possible to provide an overview of workflow tools in this section. Instead, we mention a few that are potentially relevant to our own work, either as part of our toolbox or as inspiration:

- The ExaWorks initiative [21], funded by the Exascale Computing Project (ECP) in the U.S., is spearheading a collaborative design process to develop a Software Development Toolkit (SDK) for workflows. This toolkit encompasses a diverse array of workflow management tools that can be combined and interact through common interfaces. Among them: `Flux` (a more flexible scheduler), `Swift/T` (for dataflow computing over MPI), `Parsl` (for parallelization of python scripts), `PSI/J` (a wrapper for submission to schedulers).

- `FireWorks` [22] is a workflow manager similar to `AiiDA` in scope, and is used by the Materials Project for their high-throughput calculations.

- The Common Workflow Language (CWL) [23] is an open standard for describing and sharing workflows and tools used in computational science. It serves as a specification that provides a common way to describe the steps of a data analysis or computational pipeline in a portable and platform-agnostic manner. CWL is designed to be simple, human-readable, and flexible. It does not deal with the workflow execution, which other engines must handle.

- `Plumbum` [24] is primarily a Python library for shell script-like functionality, providing a convenient and expressive way to interact with shell commands and create subprocesses. While `Plumbum` itself is not designed explicitly as a workflow management tool, it can be used as a building block within Python scripts to create and

manage workflows. `Plumbum` can be helpful for tasks such as running external commands, capturing their output, handling input and output files, and managing dependencies between different steps in a workflow.

4 Workflow design and implementation plan

4.1 Outline of ideas

We list first some key ideas and boundary conditions for the design phase of MAX exascale workflows within WP2:

- How the analysis of specific scientific showcases and broader use cases determines workflow archetypes leading to design ideas.
- Combine lightweight tools and strategies in ad-hoc solutions, rather than attempting to develop a common workflow framework.
- Aim at reasonable reliability while considering the design and deployment costs involved.
- Focus on generality and portability (do not over-engineer a design for a specific system).
- The ideas behind workflow design can also benefit code optimization, thus feeding into work by WP1.

In addition to these, we would like to raise some comments addressed to the access and users' policies of the EuroHPC JU machines:

- We need access to machine resources (in the form of project allocations that ideally should not be constrained by the rigidity of formal calls) to try the ideas and strategies, and to deploy demonstrators for workflows.
- We need cooperation from sys-admins in order to deploy the workflows in a way that is reasonably compliant with security measures but not overly encumbered by them. This applies to flexibility in the connection to the machines and to the use of techniques such as sockets and proxy servers.

A theme running through the whole document is that the boundaries between "workflows" and "codes" are blurred. This has several implications, as shown in the sections below, naively titled according to the monolithic vs. multi-step or static vs. dynamic aspects. But a more profound side to the theme is that it gives a more unified outlook to the MAX effort. It is foreseen that there will be a downward percolation in workflow "offloading" techniques towards codes, as well as a more modular and smarter approach to workflow design based on code performance models, fallback measures, and other features beside pure performance.

Another issue is that most of the scientific showcases demand some extra implementation work on code "physics" features, or a refactoring of existing functionality to develop better workflows. This aspect is not covered in detail in this report, but this implementation work should be considered in the overall planning, notably in respect to workflow demonstrators.

4.2 Notes on types of exascale workflows

4.2.1 Near-monolithic workflows

These workflows feature the execution of a "code". The quotes call attention to the fact that a code is really a pre-packaged workflow, with individual sub-tasks and data flow. While in principle one could say that there is nothing to be done regarding workflow design in this case, it is worth pointing out that the viewpoint can turn inward and ask whether the operation of the code itself could be improved (in scalability, overall throughput, and reliability) by applying some of the concepts developed for "bona fide workflows".

This is an illustration of the general principle that all computational processes are workflows at a certain level. For example, consider this snippet to implement the "parallel over $\mathbf{k}$" idiom in SIESTA, in which the diagonalisation of the $H(\mathbf{k}), S(\mathbf{k})$ matrices is done in parallel over the $\mathbf{k}$-points. It should be possible to do:

```
do concurrent
   get k-point
   [build and diagonalize H(k),S(k)]
enddo
[check for completion]
```

where the bracketed line in the loop "offloads" an independent process (a la `ZeroMQ`, for example) that performs the setup and diagonalisation. Since the instrumentation for this kind of asynchronous offloading (or out-handing) is almost trivial, it should be possible to identify spots in the code that could benefit from the added flexibility (e.g., use the right number of MPI ranks in the diagonalisation process, and a separate one in the main code, without the need for explicit handling of communicators).

More generally, near-monolithic workflows still need to request appropriate resources, and these could be more efficiently used if different parts of the code are segmented. Alas, a complete code refactoring at this level might not always be possible, so one-off allocation of resources might be needed. Here it is essential to have a good performance model to help in the matching of the request to the actual needs. Internal resource balancing could be achieved by not trying to extract the last drop of time-to-solution (TTS) optimisation in some sections of the code that might need a relatively large resource allocation, trying to focus instead on a reasonable TTS looking at the overall needs.

4.2.2 Static-DAG workflows

These are workflows in which all the relevant steps are specified at the outset. In principle, all tasks are known and perfectly determined by a DAG, and one can write a "script" (or "spec" in `AiiDA` parlance) for the workflow which includes the appropriate launching of tasks at specific points. Very few useful workflows are actually static in this way. Even without taking into consideration possible failures that need to be planned for (thus involving more tasks), most workflows feature some kind of "while loop" (for convergence of a particular feature, or for gathering appropriate statistics). Thus, the "scripting" has been extended to cover iteration and testing constructs.

4.2.3 Dynamic workflows

The showcase on the computation of thermal conductivity with the Green-Kubo method matches the general idea of combination of a producer process (typically running a molecular-dynamics simulation) with the need to compute a time-series of a linked specific property (in this case thermal fluxes used for the determination of the thermal conductivity). More generally, the producer might actually be a series of processes generating data snapshots that are to be passed on downstream. This pattern applies to machine-learning, structure prediction, free-energy calculations, and many others. It is a most general pattern, and the techniques and strategies involved will be of general use.

One of the differentiating aspects is that there might be an "internal" production of new tasks (as in the producer model) that cannot feature explicitly in a static DAG but need to be accounted for anyway. Workflow "scripting" is not directly usable without extending the meaning of some of the constructs. Rather than packaged workflow steps, there are "events" (e.g. "a new snapshot is available") that need to be processed. A proof-of-concept of the event processing in a simple toy workflow is available online (See Ref. [25]).

4.3 Putting everything together in a design plan

In the "ideal conditions" model, and in collaboration with WP1 in those aspects that concern code functionalities, a design plan for exascale workflows can be devised as follows:

- Implement the needed "physical" features in codes for property calculation or simulation capability;
- Develop performance models for the relevant functionalities of the codes;
- Refine communication interface libraries (to `ZeroMQ` and to file-oriented stores);
- Enable "workflow task/data offloading" asynchronous/non-blocking calls at relevant places in codes;
- Streamline the `AiiDA` plugins for codes to offer the possibility to elide certain inputs/outputs and to deactivate provenance handling;
- Leverage `HyperQueue` to maintain a proper cohort of workers, matched to specific workflow sections by resource tags, and to pack, load-balance, and execute the tasks interfacing with the system's scheduler.
- Supplement workflow scripts with a monitoring thread that interfaces with `HyperQueue` to manage workers.
- When using a file-oriented data-exchange interface, leverage `HyperQueue`'s data layer and `AiiDA`'s extensions to provide referential transparency for data.

Workflow scripting in the above refers to the specification of the operations involved. It can be done by `AiiDA`'s spec tool when appropriate, but also by other tools. A more

general term that includes this specification and the management is "workflow orchestration". An orchestrator can be built by combining our tools with a general-purpose scripting language.

A second phase will extend reliability and robustness. As mentioned earlier, cost-effectiveness demands to focus on the most probable failure modes, which are those associated with code operation, followed by system errors. As for dealing with congestion in resource allocation, our hope in the first place is that appropriate partitions be made available for workflow tests, deployment, and production, but some measures can be taken if this is not possible. A list follows:

- Design auxiliary distributed storage elements for the socket interface to recover the data associated to failed executions. This will not be needed in the case of file-oriented stores, except in the presence of congestion;
- Formalise error codes and return values from codes to take proper recovery actions;
- Checkpoint those parts of the workflow that cannot be re-executed (e.g. the state of any producer processes);
- Implement heuristics to throttle specific sections of the workflow in the presence of congestion.

The timeline for these actions can only be approximate. Those in the first group can be worked-on mostly simultaneously, and workflow demonstration can proceed in stages as they complete.

In the next sections we provide more details about the work to be done for the design of workflows for the scientific showcases presented in Sec. 2. It will be seen that they are at various stages of conceptual design and implementation. For some, the necessary physical-property implementation parts are already in place. Some other showcases demand the implementation of new functionality in the codes. Such is the case for example of the NAMD workflow, which needs the excited states' gradients in `turbo_Davidson`, as well as the non-adiabatic couplings' calculator, also within the QUANTUM ESPRESSO suite. In these cases workflow design can proceed abstracting the missing functionality, but obviously testing and demonstration must be put on hold.

It should also be mentioned here that the focus of WP2 is mainly on the design and demonstration of workflows, and not as much on the ulterior scientific use, although the demonstration must involve the application to realistic systems. We thus need a balance between sound workflow design and the implementation and verification of the many scientific-level details that must feature in a complete research campaign.

4.4 Design patterns and specific issues for scientific showcases

In the following the order of the showcases has been altered in some cases to better group the different workflow patterns, and some titles have been changed.

4.4.1 Thermal conductivity

The showcase on thermal properties deals with two different use cases. The first, on the computation of the thermal conductivity through the Green-Kubo scheme, is a paradig-

matic producer-consumer workflow. In the last stages of MAX-2 the initial proof-of-concept of task distribution was already applied to the case in which only coordinates and velocities were passed from QE to QEHeat (See report D6.3). The next stage is to increase the performance of the workflow by considering restart data for the thermal flux calculation (either the wavefunctions for the appropriate snapshot or, less optimally, the charge density). Here the `ZeroMQ` approach to task distribution can in principle still be used for data handling, but the demands of large datasets, and/or the work under non-ideal conditions will necessitate fallback mechanisms for the data layer.

The second use case is the computation of properties (conductivities and other thermal coefficients) by the use of a model which is fitted to ab-initio data. This model is typically an "effective Hamiltonian" (H_{eff}) that includes higher-order force constants and/or electron-phonon coupling constants. The model can then be used in a Boltzmann Transport Equation (BTE) setting to compute the desired properties. The essential elements in this case are the production of appropriate first-principles data for the fit, and a mechanism to decide when the fit is sufficiently good. In the SIESTA-TDEP workflow, an iterative scheme is followed: an initial trial H_{eff} is used to generate a collection of "canonically distributed" coordinate snapshots. These are passed to SIESTA (or to any DFT program) to generate the coordinate-force information. A new, improved H_{eff} is fitted with these data, and the procedure continues until convergence is achieved.

From the point of view of workflow design, this is again a producer/worker paradigm, although inverted in the sense that the first-principles code is the worker. Depending on the size and symmetry of the system, the fitting step could also be a non-trivial part of the workflow. The workflow pattern is ideally a "streaming interface" for the snapshot-calculation tasks issued by TDEP, which can be received by one or more Siesta workers. This can use the same elements for task distribution already discussed, and in this case there are no large data demands. Optionally, for performance enhancement, one can add a step of prediction of the initial density-matrix by interpolation over the snapshot collection. This will accelerate the convergence of the force calculations. The rest of the workflow logic is being implemented leveraging Plumbum (see tools section) to wrap the TDEP utilities (to extract force constants, compute phonon frequencies, etc) and to handle their input and output streams.

4.4.2 Dynamics of nanoscale magnetism

The showcase on magnetic skyrmions, despite belonging to a completely different domain, actually falls within the same workflow archetype: a simpler model (an atomistic spin H_{eff}) is used to explore magnetic configurations that later are studied at a higher level of theory by `FLEUR`. Here we can follow the same pattern as above, modulated by the cost (substantially larger in general for a magnetic calculation than for a simpler force calculation) of the analysis step. Thus, typically a smaller number of workers will be employed.

4.4.3 Photocatalysis with TiO_2

This workflow has a-priori a fixed structure, with a first phase of computation of the electronic structure of sample configurations by QE, and a further computation of the optical properties by YAMBO. However, the Green's function techniques used by the

latter necessitate an extra loop of convergence checking (which was already implemented during the second phase of MAX and can act as a sub-workflow, straightforward but demanding in terms of resources). In addition, the consideration of candidate polaron configurations in the presence of defects introduces a new layer of complexity due to the number of possibilities. Hence, it has been suggested to employ an approximation to the potential energy surface (PES) for configurations, based not on actual coordinates but on a simpler lattice models featuring defect sites and potential polaron localisation regions. To obtain this PES model, machine-learning techniques can be used [26], with a training set made of sample configurations whose energy is computed by DFT calculations. This sub-workflow will produce a set of relevant configurations for the second phase of the workflow. Wave-function data from QE needs to be efficiently passed to YAMBO. This has been implemented as an `AiiDA` plugin.

4.4.4 Catalysis of (de)-hydrogenation reactions

This showcase can be seen to share a number of features with the previous one. A large phase space of possible configurations needs to be explored to identify the most promising candidates for catalytic action. The selection of configurations is better done with a surrogate model (featuring possible binding sites as well as catalyst characteristics) trained on DFT calculations. The second part of the workflow is then in principle a fixed-DAG one, with the calculation of energy barriers and other reaction data. The data-exchange needs for this workflow are modest.

4.4.5 Li-ion battery materials

The workflow for this showcase has several pieces, some of them already functional. The first is a preliminary step of quality assurance for the DFT calculations used: Hubbard parameters for a "DFT+U+V" model are obtained with machine-learning tools. `AiiDA` is used to drive the needed calculations. A second piece dealing with the exploration of the chemical and structural space of potential battery materials is in development. In principle, archetypical structures are used, so the problem seems just combinatorial, although ML techniques could be profitably used at a later stage. The diagnostic phase (based on the pinball model for conductivity and on other MD simulations for charge/discharge) can be run with any orchestrator, and `AiiDA` is the natural choice due to the previous experience.

4.4.6 Binding in Antibody/Antigen (AA) assemblies

In this workflow, the BIGDFT code is used to generate electronic-structure data (namely the density matrix (DM)) for a set of representative configurations of the AA proteins. In a code using localised basis sets such as BIGDFT, the DM can be analysed for relevant sections that corresponds to particularly strong interactions of atomic-or-residue-level moieties in the proteins. Hence, one obtains a much more compact view of the interaction modes. The point of departure is also a huge configuration space, but instead of direct surrogate models generated by machine-learning techniques, the relevant configurations to analyse are generated by molecular dynamics with a force field, in a "Hamiltonian

Replica Exchange" mode (mMM(HRE)) which serves as a sampling engine. The workflow, for a given pair of biomolecules is thus composed of a piece that carries out the mMM(HRE) calculations (several in parallel to enable the "exchange" feature), followed by BIGDFT calculations to generate the density matrices, and a "fragment analysis" to identify the most active regions. The workflow has been proven already, orchestrated by a combination of a notebook interface, python wrappers for BIGDFT and the force-field-based MD simulator, and the `remotemanager` dispatcher described in Sec. 3.2.2. Further work will focus on robustness and ease-of-use features, such as the automatic selection of Hamiltonian features to exchange among the replicas, and restarts.

4.4.7 Non-adiabatic transitions in molecular systems

The main workflow will be based on a versatile molecular dynamics framework that enables the integration of Tully surface hopping algorithms and the usage of external modules for calculating energies, forces, excited states and electronic states couplings. The workflow will leverage the libraries, modules, data structures and applications developed in the QUANTUM ESPRESSO suite, notably `pw.x` and `cp.x` quantum engines, the excited state calculator based on TDDFpT `turbo_Davidson` and additional property calculators that will be developed in MAX WP1: the excited state gradients and the excited state non-adiabatic couplings. We also plan to integrate other MAX components (e.g. the excited state gradients calculator from YAMBO). The analysis of the system's evolution during the MD run will be handled by appropriate tools depending on the use case. The overall top-level workflow is thus of a producer-consumer kind, with a producer stage that will demand substantial resources and consumers that might also be computationally expensive if they have to process large electronic-structure data.

4.4.8 Coherent quantum states for quantum computing and photovoltaic applications

The design phase for this workflow started with an exploration of the feasibility, as well as the "pros and cons" of different approaches to exciton-phonon (EP) coupling. We started with the initial plan of using the real-time propagation scheme embedded in the YAMBO code (**TD-HSEX**) [27, 28]. The interaction of excitons with phonon (e.g. atomic motion) would be achieved by including Ehrenfest classical ions dynamics (E-TD-HSEX), and in a second step via quantum electron-boson dynamics (EB-TD-HSEX) [29]. Both approaches require non-trivial interoperability between the YAMBO and QUANTUM ESPRESSO codes. This challenge was addressed by (i) interfacing YAMBO to the QEpseudo library, and (ii) enhancing the YAMBO interface with the QE phonon module, `ph.x`. The QEpseudo library reconstructs the classical ion potential, while `ph.x` computes the quantum electron-phonon couplings via density functional perturbation theory (DFPT).

However, the TD-HSEX scheme suffers from the inherent serial nature associated with the real-time propagation, and its extension to finite momenta was not straightforward. Hence, the focus has shifted to a linear-response approach to exciton-phonon (**LR-EP**), while the implementation of the TD-HSEX workflow will be postponed at a later stage of MAX. The logic for the LR-EP workflow is discussed in sec. 2.5. In this initial stage has been implemented using the `yambopy` scripting tool, which still needs manual

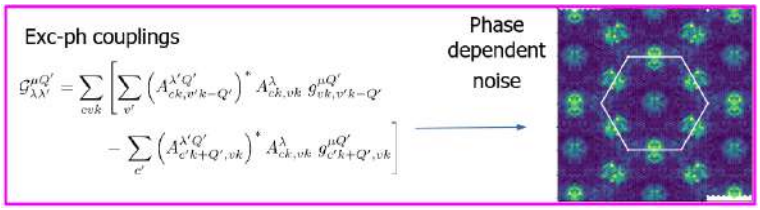

Figure 6: Exciton-phonon coupling obtained via preliminary runs of the workflow suffer from phase dependent noise which we are currently addressing via re-designing the interface between the YAMBO and the QE codes

intervention. As a test run, the LR-EP in 2D MoS_2 has been computed using the following parameters: **k**-grid 24×24×1 for DFT and GW steps, while using a 42×42×1 grid for the `ph.x` and BSE steps; interpolated grids up to 600×600×1 were used for exciton and phonon energies. The runs were performed on the local HPC cluster at ISM-CNR in Rome-Montelibretti (Intel Xeon CPU E5-2683 v4 @ 2.10GHz, 32 cpus per node). Time-to-solution: 1 week (scf, nscf and nscf on double grid on 32 cores), 1 months (el-ph on 64 cores), 2 weeks (screening on 32 cores), 2 weeks (QP+BSE on 32 cores), 1 week (data checks and restart due to different reasons; e.g. code crash, cluster shut-downs), 2 weeks (exc-ph couplings and lifetimes). Total: *3 months*. The data exchange has been done via files in the HDF5 format. Dataset sizes were: 6 GB (Kohn-Sham wave-functions), 126 GB (BSE matrices), 244 GB (el-ph and dvscf), 370 GB (dielectric screening at 10 Ry), 1 GB (other). Total: *0.75 TB*.

The key findings:

- *Technical challenges.* LR-EPT without an automated workflow proved to be cumbersome. The manual setup required the users to configure many calculations, and the subsequent combination of results involved storing large datasets to disk, also resulting in an error-prone procedure. Moreover, solutions adopted so far in the literature [30, 31, 32] require the use of extra codes (often not released), beyond YAMBO and QUANTUM ESPRESSO.

  We are now testing an automatic workflow based on `yambopy` which orchestrates the needed steps, also via a scheduler management. More advanced solutions for handling run failures, database quality checks, or data exchange on the fly, will be considered at a later stage (e.g. via AIIDA, HYPERQUEUE, and/or `ZeroMQ`).

- *Interoperability Hurdles.* Working on the numerical approach, we identified a non-trivial interoperability challenge linked to the random phases (see fig. 6) of electron-phonon matrix elements generated by `ph.x` [33]. Addressing this issue involves considering alternatives such as exchanging not only matrix elements, but also the convention adopted for using symmetry operations. Alternatively, we are exploring an approach based on the direct loading of the variations of the Kohn-Sham potential (which is phase-independent). To this end, the interface with the QEpseudo library can be effectively leveraged.

- *Physics challenges.* Most exciton-phonon quantities require very dense discrete grids in reciprocal space. So far, we have used a double grid approach with interpolated exciton and phonon energies. A better and more accurate approach,

based either on the tetrahedron method (currently implemented in QE) or on other numerical integration methods, is foreseen. In addition, accurate calculations of linewidths and luminescence peak shapes requires the inclusion of the off-diagonal elements and frequency dependence of the exc-ph self-energy, which needs to be implemented and goes beyond the calculation of just the exc-ph matrix elements [31].

5 Conclusions

We have laid the conceptual foundation for the design of workflows for a set of exascale-worthy scientific showcases, and identified some tools that provide the needed flexibility and modularity for the task. We have also identified a few common "workflow design patterns" among our showcases. A fixed "workflow development framework" is not appropriate for our exascale workflows, and ad-hoc solutions can be implemented by the combination of simpler but targeted tools. This is actually a natural extension of the foundational idea of the MAX project, which puts forward modularity and "open innovation" at the center. Another key insight is that the frontier between "code" and "workflow" is a diffuse one. Techniques for workflow design can be profitably used in code optimisation, and, conversely, code improvements can streamline workflow design.

The tasks of design, testing, and demonstration depend also on the boundary conditions for execution. First, on how far from "ideality" the target system is (in terms of the degree of congestion and the system failure rate), and second, but related, on the mostly administrative conditions for allocation of project resources, physical access to the machine(s), and exploitation of some techniques that might be seen in some circles as non-standard.

Importantly, for the continuation of WP2 activities, we call for a more streamlined project allocation process for the testing and deployment of workflows on EuroHPC machines, and for a balanced consideration of some of the techniques we plan to use, which might be seen by some system administrators as security risks.

References

[1] Marcolongo, A., Umari, P. & Baroni, S. Microscopic theory and quantum simulation of atomic heat transport. *Nature Physics* **12**, 80–84 (2016).

[2] Masuda, S., Mori, K., Futamura, Y. & Yamashita, H. Pdag nanoparticles supported on functionalized mesoporous carbon: Promotional effect of surface amine groups in reversible hydrogen delivery/storage mediated by formic acid/co2. *ACS Catalysis* **8**, 2277–2285 (2018).

[3] Kumar, S. G. & Devi, L. G. Review on modified tio2 photocatalysis under uv/visible light: selected results and related mechanisms on interfacial charge carrier transfer dynamics. *J. Phys. Chem. A* **115**, 13211–13241 (2011).

[4] Pan, X., Yang, M.-Q., Fu, X., Zhang, N. & Xu, Y.-J. Defective tio 2 with oxygen vacancies: synthesis, properties and photocatalytic applications. *Nanoscale* **5**, 3601–3614 (2013).

[5] O'regan, B. & Grätzel, M. A low-cost, high-efficiency solar cell based on dye-sensitized colloidal tio2 films. *Nature* **353**, 737–740 (1991).

[6] Bonacci, M. *et al.* Towards high-throughput many-body perturbation theory: efficient algorithms and automated workflows. *npj Comput. Mater.* **9** (2023).

[7] Amani, M. *et al.* Near-unity photoluminescence quantum yield in MoS_2. *Science* **350**, 1065–1068 (2015).

[8] Mueller, T. & Malic, E. Exciton physics and device application of two-dimensional transition metal dichalcogenide semiconductors. *npj 2D Materials and Applications* **2**, 29 (2018).

[9] Niehues, I. *et al.* Strain control of exciton–phonon coupling in atomically thin semiconductors. *Nano Lett.* **18**, 1751–1757 (2018).

[10] Cancès, E., Legoll, F. & Stoltz, G. Theoretical and numerical comparison of some sampling methods for molecular dynamics. *ESAIM: Mathematical Modelling and Numerical Analysis* **41**, 351–389 (2007).

[11] Genovese, L. *et al.* Protein–ligand interactions from a quantum fragmentation perspective: The case of the sars-cov-2 main protease interacting with α-ketoamide inhibitors. *J. Chem. Phys.* **158** (2023).

[12] Tully, J. C. Perspective: Nonadiabatic dynamics theory. *J. Chem. Phys.* **137**, 22A301 (2012).

[13] Tully, J. C. Dynamics with electronic transitions. *J. Chem. Phys.* **93**, 1061–1071 (1990).

[14] Tully, J. C. & Preston, R. K. Trajectory surface hopping approach to nonadiabatic molecular collisions: The reaction of H^1 with D_2. *J. Chem. Phys.* **55**, 562–572 (1971).

[15] Nelson, T. R. *et al.* Non-adiabatic excited-state molecular dynamics: Theory and applications for modeling photophysics in extended molecular materials. *Chem. Rev.* **120**, 2215–2287 (2020).

[16] Mirón, G. D. *et al.* The carbonyl-lock mechanism underlying non-aromatic fluorescence in biological matter. *Nature Communications* **14(1)**, 1–13 (2023).

[17] Chung, C. W. *et al.* Intracellular $A\beta42$ aggregation leads to cellular thermogenesis. *J. Am. Chem. Soc.* **144**, 10034–10041 (2022).

[18] Hintjens, P. Ømq - the guide. https://zguide.zeromq.org/ (2019).

[19] BeeGFS. Beeond: Burst buffer on demand. https://www.beegfs.io/wiki/BeeOND (2018).

[20] Apache cassandra. https://cassandra.apache.org/ (2023). Accessed: Dec. 19, 2023.

[21] Bosilca, G. *et al.* Exaworks: Workflows for exascale (2021). URL https://arxiv.org/abs/2108.13521. Accessed: Dec. 19, 2023, 2108.13521.

[22] Jain, A. *et al.* Fireworks: A dynamic workflow system designed for high-throughput applications (2015). URL https://arxiv.org/abs/1501.04690. Accessed: Dec. 19, 2023, 1501.04690.

[23] Common workflow language. https://www.commonwl.org/ (2023). Accessed: Dec. 19, 2023.

[24] Filiba, T. Plumbum: Shell combinators. https://pypi.org/project/plumbum/ (2023). Accessed: Dec. 19, 2023.

[25] Producer-consumer toy workflow. https://github.com/albgar/fortran_cloud/tree/cloud-ag/workflows/producer-consumer (2023).

[26] Birschitzky, V. C., Ellinger, F., Diebold, U., Reticcioli, M. & Franchini, C. Machine learning for exploring small polaron configurational space. *npj Comput. Mater.* **8** (2022).

[27] Attaccalite, C., Grüning, M. & Marini, A. Real-time approach to the optical properties of solids and nanostructures: Time-dependent bethe-salpeter equation. *Phys. Rev. B* **84**, 245110 (2011).

[28] Sangalli, D. *et al.* Many-body perturbation theory calculations using the yambo code. *J. Phys.: Condens. Matter* **31**, 325902 (2019).

[29] Karlsson, D., van Leeuwen, R., Pavlyukh, Y., Perfetto, E. & Stefanucci, G. Fast green's function method for ultrafast electron-boson dynamics. *Phys. Rev. Lett.* **127**, 036402 (2021).

[30] Chen, H.-Y., Sangalli, D. & Bernardi, M. Exciton-phonon interaction and relaxation times from first principles. *Phys. Rev. Lett.* **125**, 107401 (2020).

[31] Chan, Y.-h. *et al.* Exciton lifetime and optical line width profile via exciton–phonon interactions: Theory and first-principles calculations for monolayer MoS_2. *Nano Lett.* **23**, 3971–3977 (2023).

[32] Zanfrognini, M. *et al.* Distinguishing different stackings in layered materials via luminescence spectroscopy. *Phys. Rev. Lett.* **131**, 206902 (2023).

[33] Lechifflart, P., Paleari, F., Sangalli, D. & Attaccalite, C. First-principles study of luminescence in hexagonal boron nitride single layer: Exciton-phonon coupling and the role of substrate. *Phys. Rev. Mater.* **7**, 024006 (2023).