INFRAEDI-02-2018

www.bioexcel.eu

BioExcel-2 Project Number 823830

D2.1 – State of the Art and Initial Roadmap

WP2: Convergence of HPC/HPDA and Improved Usability

Document Information

Deliverable Number	D2.1
Deliverable Name	State of the Art and Initial Roadmap
Due Date	2019-06-30 (PM6)
Deliverable Lead	IRB
Authors	Adam Hospital (IRB) Stian Soiland-Reyes (UNIMAN) Josep Lluís Gelpí (BSC) Pau Andrio (BSC) Daniele Lezzi (BSC) Sarah Butcher (EMBL-EBI) Ania Niewielska (EMBL-EBI) Yvonne Westermaier (NBD)
Keywords	Tools, Workflows, Workflow managers, Hardware infrastructures, Container technologies
WP	WP2
Nature	Report
Dissemination Level	Public
Final Version Date	2019-06-28
Reviewed by	Mark Abraham (KTH) Alexander Bonvin (UU)
MGT Board Approval	2019-06-29

Document History

Partner	Date	Comments	Version
IRB, BSC, UNIMAN, EMBL-EBI, NBD	2019-06-03	First draft	0.1
IRB	2019-06-18	Second draft, comments and feedbacks from EB review addressed	0.2
IRB, EMBL-EBI	2019-06-19	EMBL-EBI reviews and comments addressed	0.3
UNIMAN	2019-06-19	Addressing comments	0.3.1
IRB, BSC	2019-06-20	Minor fixes	0.3.2
KTH	2019-06-28	Final edits	0.4
UNIMAN	2021-03-15	DOIs	0.4.1

Executive Summary

This deliverable is presented as a follow-up to the BioExcel-1 D2.1 deliverable, describing the new ***state of the art*** **of technologies, methods and tools** applicable to the computational biomolecular field, and presenting a *roadmap* for WP2 in the first period (18 months) of the BioExcel-2 project.

A *current situation* section briefly summarizes the work done in BioExcel-1 on the development of the BioExcel building blocks (biobb) library following a collection of software development best practices, success stories reached for a set of workflows built using this library, and finding and using BioExcel workflows through the BioExcel Cloud Portal.

The state of the art of technologies, tools and methods presented in the BioExcel-1 D2.1 is reviewed, briefly describing new hardware and software infrastructures that have been gaining popularity in the recent years, such as software containers or Jupyter notebooks. Workflow managers, especially the ones focused on the HPC and exascale supercomputers are also reviewed, presenting a collaboration with the Molecular Science Software Institute (MolSSI). New, still ongoing initiatives closely related to the BioExcel expertise are introduced, including workflows and MD trajectories data repositories. The set of available modules used on the solution-oriented workflows built for the first period of the project will be updated in this new period. A new list of proposed tools to be wrapped into the BioExcel building block library is presented, including advanced and modern functionalities such as biased MD simulations and High Performance Data Analytics (HPDA).

The final section of the deliverable describes the immediate future *roadmap* for the WP2, divided in the different tasks presented in the DoA:

I. Application building blocks for computational biomolecular simulations
II. Definition, development and specification of workflow prototypes and demonstrators
III. Optimization of Workflows for Exascale computing
IV. Convergence of HPC and HPDA
V. Provisioning and maintaining a workflow environment
VI. Retaining usability, interoperability and reproducibility in Exascale workflows

For each of the tasks, a set of objectives and corresponding milestones for the first period of the project (18 months) are introduced. An extension and update on this roadmap will be presented in the D2.3 - First release of demonstration workflows in PM18.

Contents

1 INTRODUCTION 6

2 CURRENT SITUATION 7

3 STATE-OF-THE-ART OF TECHNOLOGIES, METHODS AND TOOLS 9

3.1 HARDWARE AND SOFTWARE INFRASTRUCTURES 10
3.1.1 INFRASTRUCTURE AUTOMATION TOOLS WITH VMS 10
3.1.2 SINGULARITY CONTAINERS, KUBERNETES ORCHESTRATION 11
3.1.3 CONDA PACKAGES 12
3.1.4 JUPYTER NOTEBOOKS & BINDER 13
3.1.5 EUROPEAN OPEN SCIENCE CLOUD 14
3.2 WORKFLOW MANAGERS 15
3.2.1 COMMON WORKFLOW LANGUAGE 16
3.2.2 RADICAL-CYBERTOOLS (RCT) 17
3.2.3 PARSL 18
3.2.4 GMXAPI: GROMACS API 19
3.2.5 CROSSBOW AND CROSSFLOW 19
3.2.6 ADAPTIVEMD 20
3.2.7 FIREWORKS 21
3.3 ONGOING INITIATIVES 21
3.3.1 PROTEIN DATA BANK IN EUROPE (PDBE) GRAPH DATABASE 21
3.3.2 IRB'S MD SIMULATION DATABASE 21
3.3.3 ONE-STOP-SHOP WORKFLOW REPOSITORY 22
3.4 APPLICATION BUILDING BLOCKS 23
3.4.1 CHEMISTRY 23
3.4.2 DATA ANALYTICS 24
3.4.3 BIASED & ADVANCED MOLECULAR DYNAMICS 25
3.4.4 MD ANALYSIS 25
3.4.5 DATA RETRIEVAL & MANIPULATION 26

4 INITIAL ROADMAP PROPOSAL 28

4.1 APPLICATION BUILDING BLOCKS FOR COMPUTATIONAL BIOMOLECULAR SIMULATIONS (T2.1) 29
4.2 DEFINITION, DEVELOPMENT AND SPECIFICATION OF WORKFLOW PROTOTYPES AND DEMONSTRATORS (T2.2) 30
4.3 OPTIMIZATION OF WORKFLOWS FOR EXASCALE COMPUTING (T2.3) 32
4.4 CONVERGENCE OF HPC AND HPDA (T2.4) 34
4.5 PROVISIONING AND MAINTAINING A WORKFLOW ENVIRONMENT (T2.5) 34
4.6 RETAINING USABILITY, INTEROPERABILITY AND REPRODUCIBILITY IN EXASCALE WORKFLOWS (T2.6) 36

5 CONCLUSIONS 39

6 REFERENCES 40

APPENDIX 42

1 Introduction

Computational biomolecular research is undergoing a breakthrough period. Recent advances in computational power coupled to new parallelization strategies and computer accelerators have allowed simulations of molecular processes to reach real biological timescales. The drug discovery field is routinely using large computational power in critical steps of its pipelines, such as in the well-known Virtual Screening process, where millions of compounds are filtered out to identify a small list of candidates most likely to bind to a drug target.

New supercomputers with millions of available cores are being built and will be ready soon for carrying out research. However, there is currently no biomolecular tool able to scale up to this large number of cores. The most appropriate and suitable alternative to optimize use of this computational power is to launch a large number of parallel (independent) executions in one single job. This is possible with the help of workflows and the Workflow Management Systems controlling them.

Workflows are extremely useful in computational biomolecular research for many additional reasons: *i)* complex scientific studies usually require elaborate software pipelines, *ii)* these pipelines are commonly built from a number of different tools, *iii)* steps of the pipeline can be interconnected with dependencies, and *iv)* steps of the pipeline can have different computational time and storage size requirements.

Joining both these worlds together - exascale biomolecular workflows controlled by workflow managers - will be able to use hundreds of thousands of cores in parallel in one single job. Concepts such as elasticity, flexibility, fault tolerance, or adaptiveness are being studied and tested in a broad number of HPC-focused workflow managers. The BioExcel use cases are clear examples of complex scientific studies that will benefit from these systems.

Complex biomolecular workflows, and the possibility to run them in supercomputers, should not be incompatible with their usability. BioExcel-1 demonstrated with the BioExcel building blocks library that the same workflow can be used in desktops, virtual machines, graphical user interfaces, and supercomputers. BioExcel-2 will continue this work, extending the library, developing new and more ambitious biomolecular workflows (use-case driven), and integrating HPC calculations with data analytics on the produced results. This document contains an initial roadmap presenting the newly proposed developments **using the state-of-the-art technologies, methods and tools** analysed during the first months of the project.

2 Current Situation

Work done by the WP2 in the first BioExcel period was extensively presented in the BioExcel-1 D2.4 deliverable. This *current situation* section aims to briefly summarise and highlight the main results obtained in the first period, and provide the starting point for the second period.

The work on BioExcel-1's *portable environments for computing and data resources* focused on the setup of biomolecular workflows and modular tools, easily deployable and operational in a wide range of computational infrastructures. With a particular interest towards interoperability and reproducibility, the software development process chosen for the generation of the BioExcel building blocks and workflows followed the recommendations of ELIXIR project and the FAIR principles. As a result, a set of interoperable units based on a collection of Python wrappers encapsulating software components were developed (BioExcel building blocks - biobb's) and made freely accessible from GitHub, BioConda packages, DockerHub, and SingularityHub repositories. The software development process has been described in detail in the submitted paper "*Building of an interoperable workflow ecosystem for Biomolecular simulations within ELIXIR*" (Andrio, P. et al, Nature Scientific Data, just accepted). The current set of available BioExcel building blocks can be found in the Appendix table A1.

The newly developed BioExcel building blocks were used to assemble a set of biomolecular simulation workflows. A remarkable technical outcome was obtained with a pipeline to study flexibility changes in protein mutations (pymdsetup), which was successfully run as one single job using 38,400 cores in parallel (BSC Marenostrum 4). BioExcel-2 will extend this work towards exascale, trying to efficiently exploit large HPC systems.

Thanks to the workflow manager-agnostic design of the biobb's, different systems could be used to build and/or run the generated workflows. HPC-focused workflow managers (PyCOMPSs, Toil) were used in massively parallel runs. Workflow systems with Graphical User Interfaces (GUIs) (KNIME, Galaxy) were used to build workflows with a user-friendly drag & drop approach. Docker-ready workflow descriptions using Common Workflow Language were created to further increase interoperability and reproducibility, which execution was tested with the CWL reference implementation workflow manager cwltool.

Workflows generated were packaged inside *Virtual Machines* (VMs) to ease their use. Two different approaches were followed to make these VMs available to the users. The first one was through a Virtual Organization in the European Grid Infrastructure (EGI). VMs could be directly deployed in one of the available VO-endorsed infrastructures. The second one was through the BioExcel Cloud Portal, a web portal enabling easy access to BioExcel applications and workflows packaged in VMs. The portal allows these key applications to be deployed on demand onto the cloud providers the user is able to access, without the need of installing any kind of software. The cloud portal was successfully used in training events (in collaboration with WP4), where users were presented with everything

they needed to follow the course already installed in VMs. Users simply deployed the VMs in a BioExcel-provided cloud environment (EMBL-EBI Embassy Cloud).

While we have demonstrated the power of the BioExcel building blocks library, workflows and VMs, as well as the possibility to run them in a broad range of infrastructures; their *visibility* and *uptake* within the wider biomolecular simulation field is still in need of improvement. This is thus one of the main WP2 focus areas for the new BioExcel-2 period.

Thinking on this visibility and usability, the application of a second software packaging approach to the building blocks and workflows was started in the last year of the BioExcel project. *Conda* packaging helps ease the distribution and installation processes, taking care of the software dependencies. BioExcel building blocks are being uploaded to a particular Conda channel (repository of packages) specialized in bioinformatics software: BioConda, which is coordinated with BioContainers that directly converts each of the Conda packages to Docker containers.

As a result of a focus group meeting with pharmaceutical companies in collaboration with WP5, new mini-projects were started in the last months of the BioExcel-1 project. These included a GUI to run workflows built from the biobb's library, with a possible connection to HPC supercomputers, and the porting of the building blocks to a set of KNIME nodes. These two projects are of direct interest for the pharma industry and will be continued in the current BioExcel-2 period (T2.6).

In summary, BioExcel-1 project's results show the broad range of technologies, methods and tools used in a serious effort to reach a broad number of biomolecular simulation users, providing a workflow software library designed to be easily used at different levels: i) expert users who want to get out the most from biomolecular simulations, primarily working using supercomputers from HPC centers; ii) intermediate users who are interested in biomolecular simulations and know how to use the associated technology, but do not want to spend time installing, configuring or tuning the tools, preferring the ease of use to performance; and finally iii) entry level users who are interested in the field, but are scared (or just unable) to start using the available tools.

3 State-of-the-Art of Technologies, Methods and Tools

Coinciding with the BioExcel-1 project, there was a spike in the number of new software deployment technologies. Software containers (e.g. Docker, Singularity), are quickly gaining popularity thanks to their lower overhead in comparison to Virtual Machines (VMs). The previous security issues raised by HPC centers have been addressed, and now containers are present in most of the supercomputers [1] around the world. The way in which VMs are being deployed is also changing. VM images, typically in the range of tens of GBs, are being transformed into a single system-configuration file, which is being interpreted and used by software configuration managers such as Ansible to deploy new, on-the-fly generated VMs, improving portability and maintainability. Applications and libraries installation has also changed significantly with the appearance of software packaging technologies (e.g Conda, Environment Modules), providing system-agnostic executable environments. HPC systems are also evolving with time, being currently approaching the exascale era, with millions of cores available in one single supercomputer (Sunway TaihuLight, China, 10,649,600 cores), and heterogeneous architectures (Summit, US). Efficient scaling codes (WP1) and massively parallelizable workflows (WP2) able to run on this kind of infrastructures will be essential in the coming years.

The field of bioinformatics, and in particular the biomolecular simulation tools, are evolving in parallel with the technology. A specific category (channel) of Conda packages was created less than four years ago (BioConda, 2015), specialized for bioinformatics software and currently containing more than 6,600 packages (accessed May 2019). Available Conda packages include tools to manipulate 3D structures (OpenBabel, ACPype), to view 3D structures and trajectories (VMD, NGLview) and to run and analyse Molecular Dynamics (MD) simulations (GROMACS, AmberTools, BioExcel building blocks). MD simulation code teams have been working for years now trying to scale efficiently with an increasing number of cores (see previous paragraph), and also adapting their codes to new hardware accelerators (e.g. GPUs, FPGAs). In recent years, data analytic tools, especially machine learning and deep learning (e.g. Tensorflow) have been used together with the huge amount of data being generated in the bioinformatics and biomolecular simulation fields to train prediction models and advance drug discovery [2, 3] (see also the special issues in "*Frontiers in" journal* [1] and in "*MDPI biomolecules" journal* [2] *).* The logical convergence between data analytics and HPC, known as High Performance Data Analytics (HPDA), is one of the most popular hot-topics of the field in the present times (see Top500 article[3]).

Workflow managers, controlling complex execution pipelines, are also quickly evolving. The number of available systems is increasing everyday (see list collected by CWL project), with their visibility now reaching the whole scientific community. The usefulness and advantages of using them to manage scientific

[1] *Machine Learning in Biomolecular simulations*

[2] *Machine Learning for Molecular Modelling in Drug Design*

[3] *The Intersection of AI, HPC and HPDA: How Next-Generation Workflows Will Drive Tomorrow's Breakthroughs*

pipelines (reproducibility, portability, provenance, etc.) is currently beyond doubt [4]. New approaches such as the Common Workflow Language (CWL), a specification for describing analysis workflows and tools, are helping in sharing workflows, making them portable and scalable across a variety of software and hardware environments. From the available frameworks, HPC-focused workflow managers, able to run advanced workflows (adaptive, elastic), are of particular interest for BioExcel (e.g. PyCOMPSs, Parsl, RADICAL-Cybertools).

This deliverable is presented as a follow-up of the BioExcel-1 D2.1 deliverable, where state of the art of infrastructures and workflow managers for the biomolecular simulation field were listed and described. Here, 3 years after the submission of the above deliverable, a list of new technologies identified as useful for the BioExcel-2 roadmap is herein presented, divided into 4 main sections: Infrastructures, workflow managers, ongoing initiatives, and application building blocks.

3.1 Hardware and Software Infrastructures

Tools, workflows and web portals offered by BioExcel will be deployed and run in the most appropriate software environments, depending on their specific requirements. In the first period of BioExcel, as mentioned in section 2, the chosen software environments were mainly VMs and HPC supercomputers. In the new period, exploitation of new, state of the art technologies will be studied. The proposed technologies are described in detail in the following sections.

3.1.1 Infrastructure automation tools with VMs

Virtual Machines configuration and deployment processes have moved recently from a static mechanism (first deploy, then configure) to a more flexible one (first configure, then deploy)[4]. The latter approach, similar to the one used by container applications (see section 3.1.2), is accomplished thanks to the configuration management software. When using configuration management, all of the steps needed to go from a base image to a fully configurable and usable image are defined in an ASCII file. For example, instead of logging into a VM and manually executing a set of command line tools to set up a particular workflow environment, one would declare this set of tools using a configuration management tool such as Ansible, Salt Stack, Puppet or Chef.

Ansible is an IT automation engine, which allows open source software and cloud provisioning, configuration management, application deployment and intra-service orchestration[5]. It uses a very simple language (YAML, in the form of Ansible Playbooks) to describe system configuration.

[4] https://itnext.io/immutable-infrastructure-using-packer-ansible-and-terraform-7ca6f79582b8?gi=315e4cca280

[5] https://www.ansible.com/overview/how-ansible-works

In an analogous way, the processes of creation, update and deletion of cloud resources are now automated through the description of the target state of a desired infrastructure (Infrastructure-as-Code). CloudFormation (Amazon Web Services - AWS) and Terraform are two of the most popular Infrastructure-as-Code tools.

Terraform is an open source infrastructure as code software tool enabling definition and provisioning of data-center infrastructures using a high-level configuration language. Terraform supports a number of cloud infrastructure providers such as OpenStack, Amazon Web Services, Microsoft Azure or Google Cloud Platform.

Ansible and Terraform are complimentary solutions, each addressing a key area of application/environment and infrastructure management. Terraform provides infrastructure management, whereas Ansible helps in provisioning and configuring the software environment and applications. VMs provisioned using Terraform with base OS images and Ansible from a software requirement description will improve portability of the VMs between different cloud providers, as well as make them easier to adapt and maintain.

3.1.2 Singularity Containers, Kubernetes Orchestration

Container virtualization platforms, such as Docker or Singularity, allow software to be installed and executed under an isolated and controlled environment. They have rapidly gained popularity, especially in the *DevOps* movement, as they provide an easy way to install dependencies for software development and deployment. They mainly differ from VMs as they do not have the hardware virtualization overhead.

Docker containers (see BioExcel-1 D2.1 deliverable) are extremely popular nowadays in all the technology related fields, including computational biology. Biocontainers.pro[4], an open source and community-driven framework which provides platform independent executable environments for bioinformatics software, has more than 7,000 registered tools in their registry, summing up more than 53,000 software containers (accessed in May 2019). However, Docker containers were not as popular in the HPC community, mainly due to these issues: (1) security - Docker gives superuser privileges to the user running them and (2) scheduling - lack of communication between job scheduler system requirements and the Docker daemon. Different approaches have recently appeared to tackle these problems (e.g. uDocker[5], Singularity, Shifter).

Singularity containers were designed to bring containers and reproducibility to scientific and HPC fields[6]. It was designed with the ability to run Docker containers, to avoid fragmentation in user space. Singularity has native support for HPC features such as interconnection networks (e.g. InfiniBand, Intel Omni-

Path Architecture), message passing interfaces (OpenMPI), and job scheduler systems (e.g. SLURM, TORQUE, Oracle Grid Engine); thus, many HPC centers have adopted it.

Kubernetes is an open source container orchestration system for automating deployment, scaling, and management of application containers across clusters of hosts. It works with a range of different software container systems, including Docker and Singularity[6]. It is the perfect mechanism for using containers in public or private cloud environments, offering a really simple way to create and deploy functional services from software containers[7].

These latest container technologies offer new opportunities to explore portability, efficiency, reproducibility, and visibility of BioExcel workflows and tools.

3.1.3 Conda Packages

Conda is a package and environment management system that quickly installs, runs, and updates packages and their dependencies. Conda easily creates, saves, loads, and switches between environments on a computer. Initially created for Python programs, it is now able to package and distribute software for any language.

Anaconda and Miniconda are software package distributions, containing a collection of pre-built and pre-configured packages that can be installed and used on a system.

Anaconda is a free and open source distribution of Python and R packages for scientific computing, containing more than 1,700 popular packages (accessed May in 2019), including data science, machine learning, deep learning applications, data processing and analytics, and data visualizers. A large number of channels with their own associated packages are also available. A channel is a repository of packages for a specific theme: a science field, a particular software program framework, an individual programmer, etc. BioConda, for example, is an Anaconda channel for bioinformatics software, which currently contains more than 6,600 packages (accessed in May 2019), including tools to manipulate and view 3D structures, trajectories and tools to run and analyse MD simulations (BioExcel-1 work, see BioExcel-1 D2.4 deliverable).

[6] https://www.sylabs.io/2019/04/the-singularity-kubernetes-integration-from-a-deep-learning-use-case-to-the-technical-specifics/

[7] https://opennebula.org/news-from-the-marketplace-kubernetes-appliance/

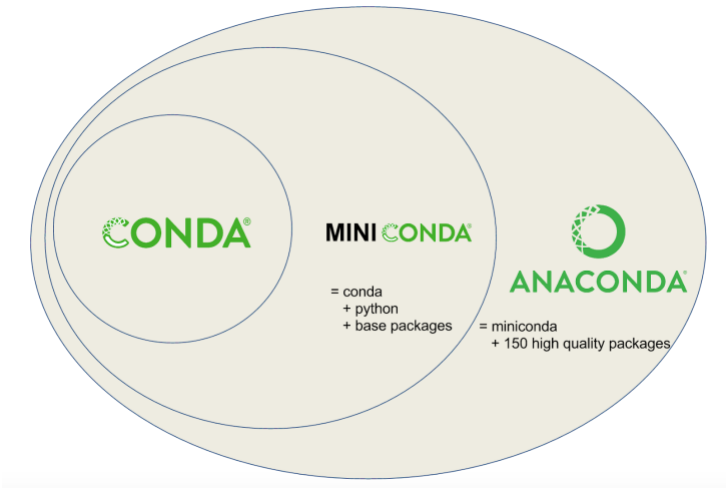

Fig.1: Conda, Miniconda and Anaconda distributions.

Miniconda is a reduced distribution, which can be seen as a subset of Anaconda (see Fig.1), including only the Conda environment manager, the Python programming language, and an smaller number of base packages. Miniconda is more suitable for production deployments, where only a stable, and well-defined set of dependencies is required.

3.1.4 Jupyter Notebooks & Binder

Jupyter Notebooks are (2015, v1.0) are an extension of the iPython (interactive Python) initiative, a command shell for interactive computing in multiple programming languages (initially developed for Python). Jupyter Notebooks extend the console-based interactive shell providing a web-based application with which programmers can develop, document, execute code, and share results. The notebooks' computational interface (Read–Eval–Print Loop, REPL) replicates the one issued in the late 1990s by other programs such as Maple or Mathematica.

Jupyter Notebooks have steadily gained traction since their official presentation 4 years ago, especially in the scientific community. This popularity is mainly due to a set of useful functionalities:

- Presentation and visualization features: charts, interactivity, sortable tables, selectors, molecular visualizers, and in general any kind of integration offered by the compatible libraries (IPython, jQuery, Bootstrap, MathJax, etc.).

- Possibility to explore scientific or mathematical ideas thanks to the executable cells and their interactivity. In the GUI, variables can be changed and cells executed as many times as needed to investigate particular algorithms or methods.

- Possibility to directly convert the notebook to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python).

- Suitability for educational purposes: Tutorials can be built using notebooks containing programming code accompanied by text information and/or documentation, versatile graphical charts and data visualization, and can also be easily exported to printed material. Easy access through a web-browser is provided. The document can be separated from the computational core using an external server to supply students with lectures, slides, or interactive tutorials.

Due to their popularity, Jupyter Notebooks have recently reached Cloud computing, with important cloud providers adopting them as a front-end interface for cloud users (Amazon, Google, Microsoft's Azure). Using these cloud resources, Jupyter notebooks can be produced and shared with their fully interactive power, without the need of installing anything locally.

Binder was the first dynamic environment to share Jupyter Notebooks not as static pages but as interactive, live notebooks, with all necessary libraries. Binder allows the creation of custom computing environments able to execute Jupyter Notebooks that can then be shared and used by many remote users. Binder is powered by BinderHub, which is an open source tool that deploys the Binder service in the cloud. Mybinder is a free example of a Jupyter Notebooks deployment using BinderHub. Mybinder is able to build a Docker container (image) of the repository, and create a live environment using just a URL linking a GitHub repository containing one (or more) notebooks, and an environment settings dependency file. BinderHub can be installed locally in a private cloud environment.

3.1.5 European Open Science Cloud

The European Open Science Cloud (EOSC) fosters open science and open innovation: It is a network of organizations and infrastructures from various countries and communities that supports the open creation and dissemination of knowledge as well as scientific data. EOSC is a virtual environment with open and seamless services for storage, management, analysis and re-use of research data. It runs across borders and scientific disciplines by federating existing scientific data infrastructures, currently dispersed across disciplines and the EU Member States.

The EOSC Portal is the entry point to the EOSC initiative, providing access to data, services and resources. It contains up-to-date information about the EOSC, including best practices, governance and user stories. The European Grid Infrastructure (EGI) is one of the main contributors to the EOSC Portal. It offers advanced computing (Cloud Compute and High-Throughput Compute) and data services from publicly funded and commercial organizations, and operates

federated identity provisioning, authentication and authorization services for the EOSC users and service providers.

In the EOSC-Life project, where several BioExcel and ELIXIR partners take part, there is a particular focus on providing FAIR tool and workflow definitions that are also portably executable across the EOSC infrastructure. Key aspects of this approach are packaging a large set of life science tools using BioConda or Docker containers, describing their invocation using CWL, executing them on distributed compute nodes using the GA4GH workflow and task execution APIs, annotating their functionality and origin using Research Objects, and composing them using multiple workflow managers including Galaxy, KNIME, Nextflow and CWL (Fig. 2).

The EOSC-Life federated *workflow registry* is currently being developed, using the BioExcel-1 *workflow repository* prototype as a starting point, but extending it to be backed by a model of distributed CWL descriptions maintained in multiple GitHub repositories. In BioExcel-2 we will join this effort as a co-development, adding the *biobb* building blocks (which are already largely described in CWL) as an early case of federated tool descriptions for the workflow registry, as well as influencing further EOSC development by contributing with our knowledge and particular needs from the point of view of scalable workflow execution for biomolecular simulations.

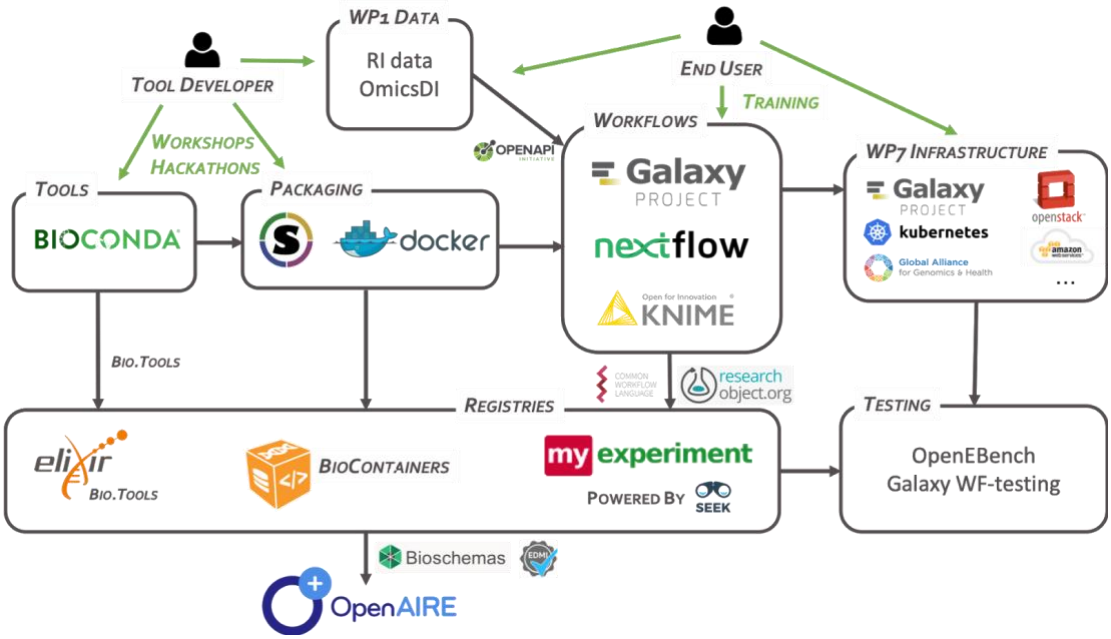

Fig.2: Proposed EOSC-Life structure.
(source: https://github.com/eosc-life/)

3.2 Workflow Managers

Complex scientific workflows developed by BioExcel will be controlled at run-time using workflow managers. At the beginning of BioExcel-1, the focus was on 5 platforms: Copernicus, KNIME, Galaxy, Apache Taverna, and COMPSs/PyCOMPSs.

Common Workflow Language (CWL) emerged as a good candidate to support, for its capabilities to describe, specify and facilitate the executions of scientific workflows across a variety of software and hardware environments. Due to the collaboration with the Molecular Sciences Software Institute (MolSSI), our main focus in Bioexcel-2 will be around the Biomolecular Simulation community workflow tools and system development projects. The aim of the collaboration is to reach a joint agreement on which systems are most relevant to the researchers, and thus have the best chance of being long-term sustainable products. The technologies studied so far were presented in a white paper based on the first joint BioExcel-MolSSI workshop[7] and are described in detail in the next sections (except COMPSs/PyCOMPSs, already extensively introduced in the BioExcel-1 D2.1 deliverable). A summary table containing all the tools presented in the following sections is available in the appendix of this document (Table A2).

3.2.1 Common Workflow Language

The Common Workflow Language (CWL)[8] is a specification for describing workflows and tools that are portable and scalable for execution across a variety of software and hardware environments, from workstations to cluster, cloud, and high performance computing (HPC) environments. CWL has been adopted by the ELIXIR project as the recommended standard language to describe workflows and has seen a particular growth in uptake across the bioinformatics community, although it is designed to meet the needs of any data-intensive science, also including Medical Imaging, Astronomy, Physics, and Chemistry.

The core concept of CWL is the common workflow engine denominator of pipelines of command line tools executed on cloud and cluster infrastructures, but ensuring reproducibility and tool portability using technologies like containers (Docker, Singularity Conda) and explicit annotations of tool parameters, versions and file handling.

Multiple workflow engine vendors already support executing CWL, including Toil, Arvados, Airflow, IBM CWLExec, REANA, Cromwell. In addition the reference implementation *cwltool* is frequently used for testing and development of CWL workflows, although the other engines provide more scalable and reliable execution across a wider variety of compute infrastructures, including cloud (AWS, Azure), schedulers (SLURM, LSF, PBS-Torque, SGE, HTCondor) and higher level frameworks like Kubernetes and Mesos.

Workflows described in CWL are frequently maintained in GitHub, and can be directly transformed into a graphical diagram. This was made possible by the BioExcel-1 supported CWL Viewer, a richly featured web visualization suite, which graphically presents and lists the details of CWL workflows with their inputs, outputs, and steps (Fig.3). It also packages the CWL files into a downloadable Research Object Bundle including attribution, versioning and dependency metadata in the manifest, allowing CWL workflows to be shared and archived using versioned permalinks.

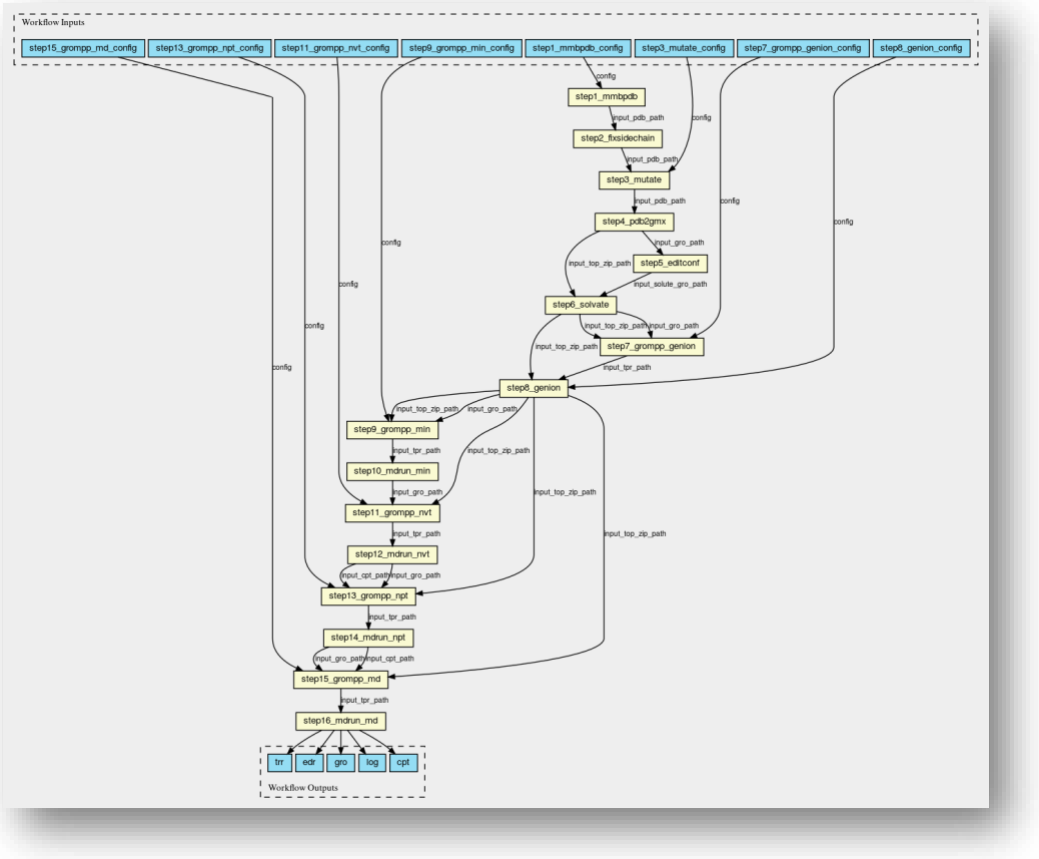

Fig.3: CWL diagram generated with CWLViewer of the pymdsetup BioExcel workflow, updated version (2019).

3.2.2 RADICAL-Cybertools (RCT)

RADICAL-Cybertools[9] enable the execution of ensemble-based applications on a variety of high performance computing infrastructures. An increasing number of scientific domains is adopting and benefiting from ensemble-based applications. Most notably, MD simulations are nowadays rather executed as many parallel jobs of short simulations instead of a single, long and very large MPI job. There are both scientific and algorithmic reasons for this. It is more likely to have a better conformational sampling with many short MD runs than just one long MD (although a good representative set of starting conformations needs to be generated previously). In addition, many short MDs are a form of ensemble scaling. In contrast, a long simulation needs high performance, which requires strong scaling, which is at odds with the restriction of having to use a single core.

RADICAL-Cybertools (RCT) consist of 4 independent software systems: RADICAL-SAGA (RS), RADICAL-Pilot (RP), RADICAL-Ensemble Toolkit (EnTK), and RADICAL-Analytics (RA). RS enables interoperability across job schedulers, file transfer and resource provisioning services via a unified, high-level API. RP implements the pilot paradigm and architectural pattern offering concurrent execution of heterogeneous tasks on the same pilot and the support of more than

twelve methods to launch tasks. EnTK allows users to develop ensemble-based applications in terms of static or adaptive pipelines, stages and tasks, abstracting away the complexities of resource and distributed execution management. Finally, RA offers a low-level API to profile traces of other RCT tools, enabling time-series analysis of each component behaviour. RCT successfully enabled biological, earth and climate science simulations from more than twenty research groups on leadership-class and large HPC machines in the United States, Europe and Japan.

3.2.3 Parsl

Parsl[10] is a Python library for programming and executing data-oriented workflows (dataflows) in parallel. Parsl scripts allow selected Python functions and external applications (called apps) to be connected by shared input/output data objects into flexible parallel workflows. Rather than explicitly defining a dependency graph and/or modifying data structures, developers simply annotate Python functions instead. Parsl constructs a dynamic, parallel execution graph derived from the implicit linkage between apps based on shared input/output data objects. Parsl then executes apps when dependencies are met. Parsl is resource-independent, that is, the same Parsl script can be executed on a laptop, cluster, cloud or supercomputer.

Parsl is used for a variety of data-oriented workflows ranging from traditional many task computing workflows to newer online, machine learning, and interactive computing models. In each case the underlying workflows share similar structures that include pipeline (series of connected applications) and more complex conditional/loop-based workflows (e.g. simulation-analysis loop). These workflows arise in many scientific domains including biology, physics, cosmology, chemistry, and social sciences.

Parsl and its predecessors, Swift/K and Swift/T, have enabled a wide variety of applications: simulations (supercooled glass materials, protein and biomolecular structures and interactions), climate model analyses, decision making for global food production and supply, material science studies at the Advanced Photon Source, multiscale subsurface flow modelling, power grid modelling, high-resolution surface modelling of the arctic, large-scale neural network hyperparameter optimization for cancer research, understanding the physics of overtaking manoeuvre in Indy Car racing, and high-resolution modelling of urban airflow, using machine learning to predict stopping power in materials, studying ionic liquids and obtaining information from deep eutectic solvents, and the simulation of cosmic ray showers for high school students.

Recently Parsl has been used to simulate images to be obtained from the Large Synoptic Survey Telescope (LSST). This simulation is crucial for developing the workflows needed to analyze LSST data with the aim of measuring how Dark Energy behaves over time. Members of the Dark Energy Science Collaboration have created a Parsl-based workflow that executes a variety of image reconstruction steps based on simulated instance catalogues. The workflow uses Singularity containers with the primarily Python-based simulation code. Parsl then orchestrates the execution of a series of commands by first bundling tasks

into appropriate sizes for nodes (considering available resources). It then runs over 10,000s of instance catalogues, each with 189 simulated sensors to create images. The workflow has been run on Argonne's Theta supercomputer and NERSC's Cori supercomputer. In one run, the workflow used 4,000 Theta nodes (256K cores) for 72 hours.

3.2.4 gmxapi: GROMACS API

The gmxapi[11] is a Python wrapper which links to GROMACS for all cases where people would otherwise write scripts to call GROMACS binaries and functions. This includes users who are scripting GROMACS jobs for HPC, method developers who are wrapping GROMACS calls in their methods, or methods/software developers who are currently doing custom hooks to GROMACS. The users should have some scripting and programming skills, but do not need to be advanced users of GROMACS or MD and workflow software. gmxapi library is currently in development, but is already available through a GitHub repository which includes a modified version of GROMACS that supports the latest gmxapi features not yet available through an official GROMACS distribution.

The gmxapi's interface is itself not limited to a specific job size. The job start and stop latencies are not the limiting factors. The individual nature of the jobs and ensemble execution back-ends however are. These limits will be depending on the user and the scientific problem at hand. The current back-end task runner operates on MPI. It is desirable to have additional back-ends for larger scope tasks. Users should also be able to develop their own back-ends. The wider user community of GROMACS will have access to the initial versions of gmxapi as the tool set will be integrated within GROMACS. This community is self-sustaining, and will help gmxapi grow through feedback and development.

3.2.5 Crossbow and Crossflow

Crossbow is a Python-based toolkit for workflow construction and execution, aimed particularly at Crossbow clusters but more generally at distributed computing environments. It provides an easy entry to cloud-based computing for biomolecular simulation scientists. It is particularly aimed at end users with limited expertise in system administration. Crossbow provides a "one click" deployment of computer clusters consisting of a head node and a variable number of worker nodes, all linked to a shared file system.

Crossflow shares many of its design aspects with Parsl. It provides tools to wrap Python functions and external applications (e.g. legacy MD simulation codes), in such a way that they can be combined into workflows using a task-based paradigm. Crossflow uses Dask Distributed as the task scheduling and execution layer. This handles the construction of the task graph, optimal scheduling of tasks on resources (workers), data distribution, and resilience. Though Crossflow has so far mainly been tested on Crossbow distributed clusters, it will run on any set of resources on which Dask Distributed can be deployed, which includes traditional HPC type systems.

Crossbow/Crossflow is a relatively new project, but demonstrator versions of a wide range of workflows relevant to the biomolecular simulation community have been developed. These include workflows for system preparation (high throughput, standardized workflows for e.g. system parameterization, solvation and equilibration), enhanced sampling of conformational space (e.g. simulation/loop analysis) and replica exchange methods (both temperature replica exchange and more general Hamiltonian replica exchange MDs). The molecular modelling and simulation tools that have been interfaced with Crossflow include a variety of standard MD codes (Amber, GROMACS, and NAMD), validation tools (Whatcheck), analysis tools (ProPka and FPocket), and docking tools (AutoDock Vina).

3.2.6 AdaptiveMD

AdaptiveMD is a Python package designed to create HPC-scale workflows (parallel tasks) for adaptive sampling of biomolecular MD simulations. This method seeks to improve sampling of the slowest processes, i.e. those on biologically relevant timescales, with unbiased MD replicas. To run reliably and independently over several months, AdaptiveMD was designed as a distributed application that can be launched from a laptop or directly on an HPC resource and automate asynchronous workflow creation and execution. Multiple adaptive sampling algorithms are fully automated with minimal user input, while advanced users can easily make modifications to workflow parameters and logic through the Python API. Users can prototype and utilize sampling algorithms via their own (potentially very small) analysis and frame selection scripts with a simple interface to the MD trajectory data structure. Our current out-of-the-box restart state adaptation can be expanded to use interim data from the workflow (or arbitrary logic) for making runtime adaptations to: (1) other task properties such as analysis types or parameters, (2) workload properties such as task count or (3) convergence criteria.

Validation cases are currently running on the Oak Ridge Leadership Computing Facility (OLCF) Titan to compare adaptive sampling workflows with very long trajectories run on Anton, a specialized MD supercomputer. AdaptiveMD is also currently being used on OLCF's Summit and Titan to elucidate mechanisms of lignocellulosic biomass decomposition in various solvent conditions, and to characterize medically relevant differences in disease-implicated protein mutants for the MHC class II antigen presentation complex and the IMPDH protein. AdaptiveMD can address the entire chain from a workflow-generating instance to task execution when using its native worker class. This paradigm is sufficient for rapid deployment on small-scale or lab specific resources, where dedicated workflow management software is not available or otherwise accessible to a user. To provide robust workflow management, AdaptiveMD is also integrated with the RADICAL Cybertools stack, which greatly enhances the runtime error detection and correction functionality, but has a much higher installation and configuration overhead. To keep configuration and installation simple while providing reliable workflow execution on a wide range of computational resources, further development efforts will focus on (1) isolating the workflow-generating

functionality and (2) integrating with multiple back-ends that can be seamlessly activated for task execution on a wide variety of resources.

3.2.7 FireWorks

FireWorks[12] is a free, open source code for defining, managing, and executing workflows. Complex workflows can be defined via Python, JSON, or YAML, are stored using MongoDB, and can be monitored through a built-in web interface. Workflow execution can be automated over arbitrary computing resources, including those that have a queueing system. FireWorks has been used to run millions of workflows encompassing tens of millions of CPU-hours across diverse application areas and in long-term production projects over many years. In particular, it has been used in the fields of material science, chemistry, and catalysis research. It is also used for graphics processing, machine learning, multi-scale modelling, and document processing. FireWorks has a very active support on its forum from developers and expert users. The documentation is user-friendly; therefore, new users and experts can easily use it.

3.3 Ongoing Initiatives

3.3.1 Protein Data Bank in Europe (PDBe) Graph Database

The PDBe, the European resource for the collection, organisation and dissemination of data on biological macromolecular structures, is working on a new noSQL graph-based database, powered by neo4j. A graph-based database uses graph structures for semantic queries with nodes, edges, and properties to represent and store data. The nodes represent any kind of unit (for example a PDB ID or a chain in a protein structure), whereas the edges connect the nodes according with their interactions or relationships (for example a given PDB ID containing a certain ligand). Graph-based databases, and in particular neo4j, can be queried using a specific query language known as Cypher, that allows for expressive and efficient querying to access the database in terms of the relationships between the nodes. Storing information about the macromolecular structures and their connections using this technology allows new calls that were unavailable before due to their complexity, and the ability to use Cypher speeds up many of the current REST API calls.

The PDBe initiative is already working with a development prototype of the database and an associated REST API. BioExcel joined for a Hackathon, presenting scientific use cases that could be solved using the new technology. Use cases were explored using the new database. Strengths and weaknesses of the database structure and query language were identified, generating a useful feedback for the project. BioExcel participants found that the new PDBe infrastructure is a technology with a bright future in biological sciences. It will be integrated into our workflows once it will be available.

3.3.2 IRB's MD simulation database

Despite the level of maturity reached by the biomolecular simulations field, only a few public repositories for simulation trajectories exist. Over the past 10 years, the BioExcel partner IRB published two of them: MoDEL was protein-centered with a relational mySQL database storing metadata and analyses, whereas trajectories were saved to regular storage systems; BigNASim focused on nucleic acids, and had noSQL databases storing metadata and analyses (mongoDB) as well as trajectory coordinates (Apache Cassandra). Building on the expertise gained with these two projects, IRB is working on a new repository, able to store any kind of macromolecular simulation trajectory (protein, nucleic acid or membrane-bound), and allowing uploads from the biomolecular simulation community. Trajectories resulting from the BioExcel studies (e.g. use cases) will be the first ones to be stored and made available in the repository prototype.

Trajectory data and metadata storage and backup will be provided by the IRB-BSC long term archive file systems and databases via the Starlife infrastructure. mongoDB noSQL databases will be used to store metadata, and the gridFS file storage utility (also from mongoDB) will be used to efficiently store large data files such as trajectory coordinates and analyses. A REST API programmatic interface will be exposed to query (and possibly upload) information from (and to) the database.

The database will follow the FAIR Guiding Principles, with a globally unique and persistent identifier, with rich metadata stored and retrievable by this identifier, built using a formal language for knowledge representation (EDAM ontology, with the new structural terms added by BioExcel), and maintaining a provenance information for the data and metadata. At the same time, the repository will follow the "Ten simple rules on how to create open access and reproducible molecular simulations of biological systems"[13], storing and offering the simulation protocol used to generate the trajectories, as well as all the files needed to reproduce them.

3.3.3 One-stop-shop workflow repository

During the last months of BioExcel-1 project, a set of small-scale exploratory projects were defined and started. One of them was a workflow repository, led by the University of Manchester BioExcel partner (UNIMAN). The new tool started as an extension of the SEEK platform (structured digital asset repository for projects and collaborations) to add workflow support, and was based on the myExperiment web site. The new platform, however, works more like a catalogue (e.g. with links to GitHub and Docker Hub) rather than a repository where workflow definition files are deposited. This new catalogue will allow the presentation of the BioExcel workflow building blocks and its examples, with multiple platform representations of the same workflow.

This *workflow repository* prototype has been also used as a starting point for the EOSC-Life federated *workflow registry* (see section 3.1.5). This new registry, backed by a model of distributed CWL descriptions maintained in multiple GitHub repositories, is being co-developed by UNIMAN BioExcel partners. All the BioExcel

building blocks and workflows, already described in CWL, will be used as the first entries to test the platform.

3.4 Application Building Blocks

The BioExcel-1 D2.1 deliverable introduced a set of available modules to be used as a basis for the solution-oriented workflows proposed in the first period of the project. Here, an extension and update of this list is presented, containing a new set of tools to be wrapped into the BioExcel building block library. The new list of tools, following the approach taken during BioExcel-1, is mainly use case-driven: Software needed for the use cases studies were identified and added to the list.

The complete list of tools is introduced in the following tables, divided into 5 main areas. The information displayed is divided into:

- **Tool**: Name of the tool and link to its web page, if available
- **Type**: How the tool is presented (web portal, software, etc.)
- **Platform/interface**: Where the tool is implemented and where it runs (VM, Web, HPC, etc.).
- **Dependencies**: Specific software dependencies
- **Potential users**: Potential users of the tool
- **Description**: Brief description of the tool

A summary of the whole set of tools can also be found in the Appendix table A3.

3.4.1 Chemistry

Chemistry software for small molecules will be needed in the proposed use case workflows. Several programs, packages and toolboxes for chemical data manipulation (format conversion, modelling, energy minimization, force field parameterization, etc.) exist, and some of them will be integrated in the biobb library.

Tool / portal / workflow	Type	Platform/interface	Dependencies	Potential Users
OpenBabel[14]	Software Suite	Command line	-	Users interested in small molecules, biochemistry, and pharmacology
ACPype[15]	Software (Python)	Command line	AmberTools, OpenBabel, Python	Users interested in Molecular Dynamics simulations with small molecules
CMIP[16]	Software	Command line	-	Users interested in molecular interaction potentials of macromolecules
RDKit	Software library (Python/C++)	Command line KNIME nodes	Python	Users interested in small molecules, chemoinformatics and machine learning

Tool / Portal / Workflow	Description
OpenBabel	Chemical toolbox designed to search, convert, analyze or store data from molecular modelling, chemistry, solid-state materials, biochemistry or related areas
ACPype	A Python-based tool for Antechamber to generate topologies for chemical compounds
CMIP	Package to compute classical molecular interaction potentials (electrostatic, VdW, solvation), perform protein-ligand docking simulations, and predict water and ion positions on the surface of macromolecules
RDKit	RDKit is a collection of cheminformatics and machine-learning software written in C++ and Python

Table 1.- Library of modules: Chemistry

3.4.2 Data analytics

Data analytics, and in particular High Performance Data Analytics (HPDA) coupled to High Performance Computing (HPC) is one of the main focuses of BioExcel-2. Large amounts of data produced by scientific workflows from use cases will be analyzed using Machine Learning and data analytics packages.

Tool/portal/ workflow	Type	Platform/ interface	Dependencies	Potential Users
dislib	Software	Command line	PyCOMPSs	Users interested in Machine Learning for HPC
TensorFlow[17]	Python Library	Command line	Python	Users interested in Machine Learning
scikit-learn[18]	Python Library	Command line	Python	Users interested in Machine Learning
pyTorch	Software library (Python/C++)	Command line	Python / C++	Users interested in Deep Learning
keras	Python Library	Command line	Python	Users interested in Deep Learning

Tool / portal / workflow	Description
dislib	dislib is a distributed computing library highly focused on machine learning on top of PyCOMPSs. Inspired by NumPy and scikit-learn, dislib provides various supervised and unsupervised learning algorithms through an easy-to-use API
TensorFlow	Free and open source software library for dataflow and differentiable programming to develop and train Machine Learning models

scikit-learn	Open source Python library for data mining, data analysis, and Machine Learning
pyTorch	Open source deep learning platform providing a seamless path from research prototyping to production deployment
keras	Open source Python library providing high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.

Table 2.- Library of modules: Data Analytics

3.4.3 Biased & Advanced Molecular Dynamics

Workflows for atomistic MD simulations were prepared and run using GROMACS and the biobb library in BioExcel-1. In Bioexcel-2, biased and advanced MD methods will be integrated in the building blocks library and workflows, increasing the simulation complexity.

Tool/portal/ workflow	Type	Platform/interface	Dependencies	Potential Users
pmx[19]	Software/ Web	Command line / Web	Python / -	Users interested in alchemical free energy simulations
plumed[20]	Software	Command line	GROMACS	Users interested in biased and advanced MD simulations

Tool / Portal / Workflow	Description
pmx	pmx is a Python-based software package providing utilities for handling biomolecular structure and topology files that are directly compatible with GROMACS, with hybrid structure and topology generation for alchemical single topology-based free energy simulations
plumed	Open source, community-developed library providing enhanced-sampling algorithms, free-energy methods, and MD trajectories analyses

Table 3.- Library of modules: Biased & Advanced Molecular Dynamics

3.4.4 MD Analysis

Trajectory analyses of BioExcel-1 were centred on GROMACS analysis tools. In Bioexcel-2, new tools will be explored, adding missing functionalities such as the ability to work with and convert different trajectory formats.

Tool / portal / workflow	Type	Platform/ interface	Dependencies	Potential Users

AmberTools	Software Suite	Command line	-	Users interested in Molecular Dynamics simulations analysis
Bio3D[21]	R library	Command line	R	Users interested in analyses of protein structure, sequence and trajectory data
MDAnalysis[22]	Python Library	Command line	Python	Users interested in Molecular Dynamics simulations analysis

Tool / portal / workflow	**Description**
AmberTools	Suite of AMBER free-of-charge utilities to work with molecular dynamics simulations: build molecules, setup systems, analyse trajectories, etc.
Bio3D	R package containing utilities for the analysis of protein structure, sequence and trajectory data
MDAnalysis	Object-oriented Python library to analyze trajectories from molecular dynamics (MD) simulations in many popular formats

Table 4.- Library of modules: Molecular Dynamics Analysis

3.4.5 Data retrieval & manipulation

New databases in the field of biomolecular simulations are being developed and made programmatically available programmatically via REST APIs interfaces. Examples are the newly developed databases from the IRB partner or the new graph-based PDBe database (see section 3.3). Tools to manipulate PDB structures, although straightforward, are really useful and are missing in our library.

Tool / portal / workflow	**Type**	**Platform/interface**	**Dependencies**	**Potential Users**
pdb-tools[23]	Software (Python)	Command line	Python	Users interested in macromolecular structures in PDB format
IRB DBs APIs (not available yet)	REST API	Command line	-	Users interested in biomolecular simulations
PDBe API	REST API	Command line	-	Users interested in data on biological macromolecular structures
biopython[24]	Python Library	Command line	Python	Users interested in computational molecular biology analysis
MODELLER[25]	Software (Python)	Command line	Python	Users interested in biological macromolecular structures

Tool / portal / workflow	Description
pdb-tools	A “swiss army knife” for the PDB format, including utilities to select/extract sections from a PDB file or renumber atoms/residues
IRB DBs APIs (not available yet)	REST API libraries for programmatic access to IRB databases offering atomistic MD trajectories (MoDEL v2.0), small molecule conformations (Bioactive Compounds), and protein conformational transitions trajectories (TransAtlas)
PDBe API	PDBe is the European resource for the collection, organisation and dissemination of data on biological macromolecular structures. This API is based on Neo4J
biopython	Biopython is a set of freely available tools for computational molecular biology written in Python
MODELLER	Program for comparative protein structure modeling by satisfaction of spatial restraints

Table 5.- Library of modules: Data Retrieval and Manipulation

4 Initial Roadmap Proposal

The main objective of BioExcel-2 WP2, as described in the DoA, is to ease the usability of biomolecular compute and data resources through a range of scientific workflows and associated deployment environments, with a focus on scalability aspects and the integration of HPC and HPDA. Work planned in this regard for the first period of BioExcel-2 is divided in two main fields: i) continue with the development of a programming framework allowing dynamic biomolecular simulation workflows with particular focus on HPC and HPDA integration, and ii) extend the number of packaged pipelines using virtualization and container technologies to ease the deployment of the software in both, cloud and HPC environments, and increase visibility and usage within the computational biomolecular field.

This section of the deliverable introduces the initial roadmap proposal for the first phase of the project divided in the 6 different tasks included in the work package (see Fig. 4). Each task is presented with a brief summary of partners involved, leader, and task length, followed by a number of objectives. Objectives are described, with the state of the art technologies associated to it (from the analysis presented in section 3) and a set of proposed milestones to be reached in this first period of BioExcel-2.

An analysis of the milestones reached and the objectives achieved from this initial roadmap will be prepared and presented in the project month 18 (1.5 years) with the deliverable *D2.3: First release of demonstration workflows*. This deliverable will also include an update of the tasks roadmap, running until the end of the project. According to this, milestones presented in this document refer to the first 1.5 years of the project, with the exception of T2.1: application building blocks for computational biomolecular simulations, which will be revisited and updated in the project month 12 with the submission of *D2.2: First release of workflow-ready building blocks library*.

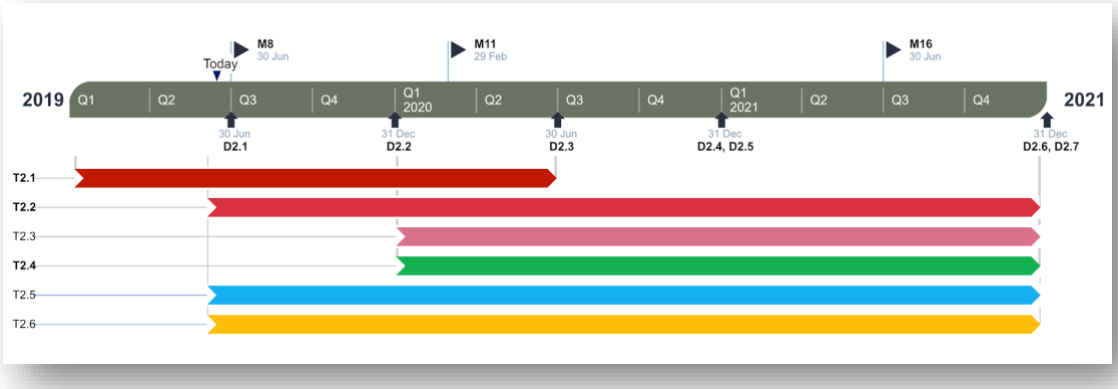

Fig.4: BioExcel-2 WP2 Timeline.

4.1 Application building blocks for computational biomolecular simulations (T2.1)

Summary

- Task leader: IRB
- Task partners: BSC, KTH, UU, UNIMAN, EMBL-EBI
- Task project months: 1-18

Objectives

- **(A) Building Blocks for the main codes: HADDOCK, pmx, and CP2K**. One of the main objectives for BioExcel-2 is to build workflows integrating the functionality and power of the main biomolecular simulation codes in BioExcel: GROMACS, HADDOCK, pmx, and CP2K. In Bioexcel-1, a set of tools from the GROMACS package was implemented as building blocks (biobb_md). Workflows using these building blocks and tackling scientific questions were presented (D2.4). In Bioexcel-2, building blocks wrapping the most important functionalities for the different main codes will be implemented. Demo workflows (T2.2) as well as workflows of interest to the pharmaceutical industry (Use case 3: Rational Drug Design; Objective B) will be built based on these building blocks. All building blocks will be generated following the best practices developed in collaboration with ELIXIR, introduced in D2.4 and presented in the submitted paper "***Building of an interoperable workflow ecosystem for Biomolecular simulations within ELIXIR***": documented using read the docs, specified with CWL and openAPI, registered in bio.tools and available from GitHub, BioConda, DockerHub, SingularityHub, and the BioExcel Cloud Portal.

    - Technology: Python, (Bio)Conda, Docker, Singularity, readthedocs, CWL

    - Milestones for year 1: Results will be presented in D2.2: First release of workflow-ready building blocks library (PM12)

        - (A1) Implement building blocks for pmx
        - (A2) Build a free-energy workflow integrating GROMACS & pmx
        - (A3) Make the workflow available through the GitHub repository and the BioExcel Cloud Portal

- **(B) Building Blocks for the development of the Rational Drug Design use case.** One of the proposed use cases for BioExcel-2, the Rational Drug Design use case, addresses studies of interest to the pharmaceutical industry. It is an ambitious project divided into four inter-dependent workflows, presented in order of increasing complexity (see D3.1, use case 3: Rational Drug Design). To exploit the power of the current supercomputer generation approaching the exascale era and following the massively parallel strategy presented in Bioexcel-1 with the Virtual Screening workflow (D2.4), a new set of building blocks for these new workflows will be implemented. New biobb categories will involve chemistry tools, biased MD methods (REMD, TMD, and SMD), or MD setup for membrane-embedded proteins.

- Technology: Python, (Bio)Conda, Docker, Singularity, readthedocs, CWL

- Milestones for year 1: Results will be presented in D2.2: First release of workflow-ready building blocks library (PM12)

    - (B1) Implement building blocks for chemistry tools (ligand parameterization, small molecules formats conversion, 3D generation)
    - (B2) Build a ligand parameterization workflow using the new biobb category
    - (B3) Make the workflow available through the GitHub repository and the BioExcel Cloud Portal

- **(C) Building Blocks for the development of High Performance Data Analytics (HPDA).** BioExcel-2's WP2 is going to work in 2 main areas: usability improvement and convergence of HPC/HPDA. HPDA refers to the use of High Performance Computing (HPC) to analyze large datasets for patterns and insights. The use of HPC parallel processing to run powerful data analysis software tools opens the possibility to examine large (even huge) datasets within a reasonable time. A set of HPDA building blocks will be developed, starting with the Distributed Computing Library (dislib), integrated in the PyCOMPSs framework. Dislib is currently offering HPC-compatible clustering algorithms (K-means, DBScan), supervised learning models such as Support Vector Machines (SVM) and Random Forest (RF), or regression algorithms (k-nearest neighbours). The newly designed blocks will be tested in workflow prototypes and demonstrators (T2.2), and will be a crucial part of the Rational Drug Design use case, especially for the workflow on *Machine Learning for efficient drug design*.

    - Technology: Python, (Bio)Conda, Docker, Singularity, readthedocs, CWL, PyCOMPSs, dislib, scikit-learn.

    - Milestones for year 1: Results will be presented in D2.2: First release of workflow-ready building blocks library (PM12)

        - (C1) Implement building blocks for HPDA tools, wrapping dislib utilities
        - (C2) Build an HPDA workflow prototype using the new biobb category
        - (C3) Make the workflow available through the GitHub repository and the BioExcel Cloud Portal

4.2 Definition, development and specification of workflow prototypes and demonstrators (T2.2)

Summary

- Task leader: IRB
- Task partners: BSC, UNIMAN, NBD
- Task project months: 6-36

Objectives

- **(A) Workflow prototype: Structural conformations from a small molecule**. Workflow to generate an ensemble of different molecule conformations for a particular small molecule. The workflow starts with a small molecule in 2D or 3D, which could be represented in a variety of file formats (e.g. SMILES, mol2, SDF, PDB, etc.). The ligand will be prepared to be used with GROMACS, simulated using biased MD methods (REMD, Hamiltonian-REMD), and finally the resulting trajectory is going to be clustered, to extract the final ensemble of conformations. The demonstration workflow will use building blocks for data retrieval (biobb_io), ligand parameterization and format conversions (biobb_chemistry), as well as biased MD simulations (biobb_md) and trajectory clustering (biobb_analysis).

    - Technology: BioExcel Building Blocks (biobbs), Python, (Bio)Conda, Docker, Singularity, CWL, PyCOMPSs, Jupyter Notebooks

    - Milestones for year 1.5: Results will be presented in D2.3: First release of demonstration workflows (PM18)

        - (A1) Implement demonstration workflow as a Jupyter Notebook
        - (A2) Prepare the workflow to be used with PyCOMPSs and CWL
        - (A3) Make the workflow available through the GitHub repository and the BioExcel Cloud Portal

- **(B) Workflow prototype: Virtual screening with binding free energies**. Workflow to extract binding free energies from virtual screening results. Demonstration workflow using building blocks for data retrieval (biobb_io), ligand parameterization and format conversions (biobb_chemistry), MD simulations (biobb_md), trajectory clustering (biobb_analysis), protein-ligand docking methods (biobb_vs) and free energy calculations (biobb_pmx).

    - Technology: BioExcel Building Blocks (biobbs), Python, (Bio)Conda, Docker, Singularity, CWL, PyCOMPSs, Jupyter Notebooks

    - Milestones for year 1.5: Results will be presented in D2.3: First release of demonstration workflows (PM18)

        - (B1) Implement demonstration workflow as a Jupyter Notebook
        - (B2) Prepare the workflow to be used with PyCOMPSs and CWL
        - (B3) Make the workflow available through the GitHub repository and the BioExcel Cloud Portal

- **(C) Workflow prototype: HPDA methods combined with HPC calculations**. Workflow to train a supervised learning model (Machine Learning - ML) from a set of descriptors extracted from biased MD simulations of conformational transitions, and classify them in categories defined by dynamic properties. Demonstration workflow using building blocks for data retrieval (biobb_io), MD simulations (biobb_md), trajectory analysis (biobb_analysis), and HPDA methods (biobb_hpda).

- Technology: BioExcel Building Blocks (biobbs), Python, (Bio)Conda, Docker, Singularity, CWL, PyCOMPSs, dislib, Jupyter Notebooks

- Milestones for year 1.5: Results will be presented in D2.3: First release of demonstration workflows (PM18)

    - (C1) Implement demonstration workflow as a Jupyter Notebook
    - (C2) Prepare the workflow to be used with PyCOMPSs and CWL
    - (C3) Make the workflow available through the GitHub repository and the BioExcel Cloud Portal

4.3 Optimization of Workflows for Exascale computing (T2.3)

Summary

- Task leader: BSC
- Task partners: IRB, KTH
- Task project months: 12-36

Objectives

- **(A) Workflows dynamicity, flexibility and elasticity**. In order to deal with the requirement of porting the workflows to (pre)exascale facilities, programmatic interfaces for the definition of the workflows are needed. They allow us to express dynamicity and support the description of iterative constructions such as conditional loops, still offering the whole expressiveness of the programming language for complex algorithms, like optimization searches. Additionally, the workflow system will be enhanced for applications that accept streamed input data and streamed output data (visualization, monitoring, etc) and for controlling the execution of the workflows (allowing the cancellation of parts of the tasks based on specific conditions).

    - Technology: Python, PyCOMPSs

    - Milestones for year 1.5: Results will be presented in D2.3: First release of demonstration workflows (PM18)

        - (A1) Implement demonstration workflows showing dynamicity, flexibility, and elasticity
        - (A2) Make the workflow available through the GitHub repository and a BSC environment module

- **(B) Singularity Workflows**. Docker containers have rapidly gained popularity, offering the possibility for the software to be installed and executed in an isolated and controlled environment. However, Docker containers were not as popular in the HPC community, mainly due to security and scheduling issues (see section 3.1.2). Singularity containers were designed to solve these problems and bring containers to the HPC field. Many supercomputers nowadays are already compatible with Singularity (including BSC supercomputers). PyCOMPSs interoperability with container platforms, that

already included Docker and Mesos, have also been extended with the inclusion of Singularity, in order to support HPC clusters. Workflows built with Singularity containers (or Docker containers converted to Singularity containers) will be implemented. They will be tested thoroughly, exploring their efficiency, usability and parallelism as well as advantages and drawbacks.

  - Technology: Python, Singularity, PyCOMPSs

  - Milestones for year 1.5: Results will be presented in D2.3: First release of demonstration workflows (PM18)

    - (B1) Implement demonstration workflows with Singularity containers
    - (B2) Make the workflow available through the GitHub repository and a BSC environment module

- **(C) Performance analysis.** The implementations of the workflows will be assessed in representative HPC platforms in order to ensure that BioExcel software is compatible with different infrastructures and that the workflows managers used for their execution are able to scale and reach the performance adequate to (pre)exascale systems. PyCOMPSs applications can be executed on different platforms without the need to change the code but just configurations files according to the environment. Following the experience of BioExcel-1 PyCOMPSs will be used also to generate trace files, to analyze the behaviour of the applications, to verify that the tasks load is properly balanced and to tune and optimize the code.

  - Technology: Extrae and Paraver tools from BSC, PyCOMPSs

  - Milestones for year 1.5: Results will be presented in D2.3: First release of demonstration workflows (PM18)

    - (C1) Analysis report of the performance of the workflows in different HPC systems

- **(D) MolSSI collaboration**. At the BioExcel-MolSSI workshop on workflows in biomolecular simulations, some of the most important biomolecular simulation workflow tools and system development projects were presented. An agreement was reached on the most urgent needs, and collaborations have started (see section 3.2). BioExcel's BSC presented the COMPSs/PyCOMPSs tool, and committed to participate in all the discussions and event organizations, starting with a follow-up of the previously mentioned joint workshop.

  - Technology: COMPSs/PyCOMPSs, Parsl, AdaptiveMD, RADICAL-Cybertools, gmxapi, Fireworks, Crossbow, CWL

  - Milestones for year 1.5:

- (D1) Co-organize the second BioExcel-MolSSI workshop on workflows in biomolecular simulations (follow-up of the first workshop run in Barcelona in December 2018).
- (D2) Continue with an active participation in the collaboration established, creating synergies between global biomolecular simulation workflow managers.

4.4 Convergence of HPC and HPDA (T2.4)

Summary

- Task leader: BSC
- Task partners: IRB, BSC, UNIMAN, EMBL-EBI
- Task project months: 12-36

Objectives

- **(A) Workflows integrating HPC and HPDA**. An important trend is the convergence of HPC and HPDA by combining HPC simulations and data analytics tasks in complex workflows. The PyCOMPSs/COMPSs framework has recently been extended to support the HPC to HPDA convergence: a task in a COMPSs workflow can be a sequential task run in a core, a threaded task run in a node, or a complex simulation run in a set of nodes. This enables the development of complex and dynamic workflows, composed of computational and analytic parts. In BioExcel-2, we aim at implementing PyCOMPSs workflows which combine computational biomolecular workflows and HPDA algorithms. The work can leverage on the ongoing dislib library, which includes machine learning algorithms implemented in PyCOMPSs able to run on distributed computing platforms.

  - Technology: Python, PyCOMPSs, dislib

  - Milestones for year 1.5: Results will be presented in D2.3: First release of demonstration workflows (PM18)

    - (A1) Implement demonstration workflows showing HPC/HPDA convergence
    - (A2) Make the workflow available through the GitHub repository and a BSC environment module

4.5 Provisioning and maintaining a workflow environment (T2.5)

Summary

- Task leader: EMBL-EBI
- Task partners: IRB, BSC, UNIMAN
- Task project months: 6-36

Objectives

- **(A) Infrastructure-as-Code description for Workflows**. To improve portability of current workflows between different cloud providers, as well as to make them easier to adapt and maintain, workflows delivered for BioExcel-1 and new ones will be provisioned with Terraform using base OS images and software requirement will be described with Ansible. Each applications description will be packaged as EBI Cloud Portal application, published to GitHub and linked to the BioExcel Portal. Workflows described in this way will still (similarly to BioExcel-1) be provisioned in the form of Virtual Machines. As most of the BioExcel tools will be directly installable via (Bio)Conda, Ansible configuration should be straightforward and reusable for more workflows.

  - Technology: Terraform, Ansible, (Bio)Conda

  - Milestones for year 1.5:

    - (A1) Describe requirements for one existing workflow with Ansible
    - (A2) Integrate the workflow into BioExcel Portal
    - (A3) Create a new workflow or CWL training environment and integrate it into the portal

- **(B) Shared Infrastructure for Workflows**. At the moment, infrastructure for workflows is provisioned in the form of Virtual Machines, each of which is used exclusively by one user and destroyed after execution. There is a need to investigate a way to create infrastructure that can be shared more efficiently among users running their workloads. One idea is to provide a system, where users can run their workflows in the form of containers. As another objective is to provide remote environment for executing Jupyter notebooks, BinderHub and related tools have been identified as potentially interesting. We plan to investigate local installation of BinderHub and integrating the execution of workflows packaged as Jupyter Notebooks into the BioExcel Portal.

  - Technology: Jupyter Notebooks, JupyterHub, BinderHub, Docker, Kubernetes

  - Related to: Objective D

  - Milestones for year 1.5:

    - (B1) Investigate BinderHub and related technologies and try to install it/them locally on EBI infrastructure

- **(C) Improved UX in BioExcel Portal**. During Bioexcel-1 events, issues with UX were identified and improvements are needed. During BioExcel-2, the user onboarding process and sharing access to workflows and infrastructure with the user in the portal will be improved. The objective is to minimise the number of necessary steps to set up the workflow by the user. The process of sharing access to infrastructure should be rethought, so as to eliminate the need of the presence of a portal administrator during training events.

  - Technology: Elixir AAI, REST API, Spring Boot, UX design

  - Milestones for year 1.5:
    - (C1) Reimplement the portal so that sharing infrastructure and applications does not require the presence of a portal administrator during events

- **(D) Remote Environment for Jupyter Notebooks**. A remote environment for Jupyter Notebooks is needed, so that the user clicks a button in the Web UI next to a workflow packaged as Jupiter Notebook. This action creates a remote, cloud environment for this notebook. The user gets a URL in return, which can be opened in a browser. The user can then interact with the notebook without needing to install anything locally. This objective can be delivered using shared infrastructure (see Objective B) or user-exclusive (possibly as ECP application).
  - Technology: Jupyter Notebook, see also Objective B
  - Related to: Objective B
  - Milestones for year 1.5:
    - (D1) Investigate technologies involved. Try implementing ECP application that provides exclusive (VM) environment for Jupyter Notebook execution.

- **(E) Platform for executing CWL-described workflows**. CWL has to be addressed in Bioexcel-2. EBI has been developing a shared system (on top of Kubernetes), where users can execute their CWL workflows via GA4GH interfaces. The back-end of the system already exists as a prototype. Our BioExcel partners will be given access to this system. The workflows need to be described in CWL and all steps need to meet DockerRequirement (all tools have to be dockerized). If the feedback is positive, the GA4GH WES client will be integrated into the BioExcel Portal, so that the CWL workflows can be executed directly from the portal.
  - Technology: CWL, Docker, Kubernetes, GA4GH WES, REST API
  - Milestones for year 1.5:
    - (E1) Obtain (possibly help to create) a CWL workflow in the BioExcel Community and run it in the GA4GH system; adapt the workflow to the system needs, if necessary.

4.6 Retaining usability, interoperability and reproducibility in Exascale workflows (T2.6)

Summary

- Task leader: UNIMAN

- Task partners: IRB, BSC, EMBL-EBI
- Task project months: 6-36

Objectives

- **CWL task support in KNIME and Galaxy to enable scalable execution from graphical workflow managers**. One of the challenges in scaling up workflows is to retain the usability from graphical workflow managers like Galaxy and KNIME. Building on the BioExcel-1 prototype to wrap specific biobb building blocks for KNIME, we will here attempt to generalize to integrate any CWL-described tool as part of a graphical workflow environment, utilizing their interoperability and reproducibility aspects with containers, as well as distributed scalable task invocation beyond the workflow manager (e.g. using GA4GH APIs).

    - Technology: CWL, KNIME, Galaxy

    - Milestones for year 1.5:

        - (A1) Demonstrate biobb building blocks invoked from Galaxy as CWL tools
        - (A2) KNIME CWL node that executes containers remotely on multiple hosts

- **(B) HPC-specific annotations in CWL workflow to improve scheduling**. CWL workflows frequently rely on Docker containers for reproducibility and reliable software distribution in cloud environments, but for effective and scalable containerized workflows in an HPC setting, many additional settings should be appended to the workflow definition, e.g. Singularity options for OpenMPI, or job scheduler parameters for SLURM (e.g. cpu count, tasks per node). Here we will explore how to best extend the CWL specification with HPC-specific annotations, and research implementing these in one of the CWL engines (e.g. CWLEXEC on LSF, or Toil on Slurm) to help improve performance on BioExcel CWL workflows as have already been shown with PyCOMPs. The goal is for this user-extension to CWL to then mature in collaboration with the wider CWL community to become part of the next generation of the CWL specification language.

    - Technology: CWL, SLURM, Singularity

    - Milestones for year 1.5:

        - (B1) Extension of CWL engine using Singularity/SLURM-specific annotations
        - (B2) Draft of generalized HPC annotation extension for CWL

- **(C) Hardware-specific packaging of BioExcel-related tools in Conda / Docker**. Although technology like Docker and Conda largely remove the challenge of code binaries being compiled for particular distributions of operating systems and versions of libraries, they do this at the cost of making

generalized assumptions about the hardware. Computationally heavy codes like GROMACS frequently rely on compile-time optimizations based on hardware variants and instruction-set extensions (e.g. GPUs, AVX) which container images are poor at distinguishing. Here we will build on BioExcel-1 work to distribute optimized GROMACS Docker binaries, to find a more reusable approach that we aim to contribute back to the packaging system Conda for handling hardware variants.

- Technology: (Bio)Conda, Docker
- Milestones for year 1.5:
  - (C1) Demonstrate HW-optimized GROMACS in BioConda
  - (C2) Draft of Conda extension for hardware variant

- **(D) Ensuring reproducible container execution on HPC**. Here we will explore the challenges of reproducible workflow execution in HPC environments aided by containers, to attempt to find convergence between the more HPC-specific (B) and HW-specific (C) extensions and the general aim of software portability. For instance, GPU-optimized simulation code might sample the ensemble slightly differently, (depending on hardware or parallelism, as the results of floating-point arithmetic varies if the order of operations changes), and would necessarily be running differently coded algorithms than the pure CPU-bound variant, but the same HPC-annotated CWL workflow should ideally run equivalently on the different setups without needing modification, even if it effectively would use different container images or code binary variants. When recording *provenance* of such workflows for the purpose of reproducibility it is thus important to capture the choices made by the workflow engine in selecting the binaries, and where possible any self-diagnostics from the codes on hardware capabilities.
  - Technology: Docker, Singularity
  - Milestones for year 1.5:
    - (D1) Demonstrate provenance recording of container image variant selection
    - (D2) Capturing GROMACS self-diagnostics as JSON provenance by biobb/CWL to record hardware-dependent choices

5 Conclusions

During BioExcel-1, the foundations of the BioExcel building block library were laid and its potential was demonstrated with technical and scientific success stories. The library and the associated workflows were developed following a set of best practices aligned with the ELIXIR project; they were able to be deployed and run in a broad range of environments, from a local desktop or Virtual Machine, to an HPC supercomputer using thousands of cores. A web-based BioExcel Cloud Portal, offering the possibility to automatically deploy pre-configured Virtual Machines to test BioExcel tools and workflows was developed and was successfully used in training events.

In BioExcel-2, the work started three years ago will continue, extending the number of building blocks with new software allowing the development of more complex, use case-driven scientific workflows. New, **state-of-the-art technologies** will be explored in the project (e.g. BinderHub, Ansible, Singularity containers). Exascale workflows, able to run on hundreds of thousands of cores will be developed and optimized using HPC-focused workflow managers and profiling tools. Data analytics methods will be integrated in the workflows to mine the huge amounts of data produced by these workflows. The BioExcel Cloud Portal will be upgraded, improving the user experience and re-designing the backend infrastructure, offering new possibilities such as the direct execution of CWL-described workflows. The link with ELIXIR established in the first period of BioExcel will continue with the work of retaining usability, interoperability and reproducibility in our produced workflows. The new building blocks will be packaged in BioConda, Docker and Singularity containers. Jupyter notebooks will be produced with tutorials and examples on how to use the library. Workflows will be specified using CWL, and registered in a new workflow repository being implemented by the project partners.

The transition from BioExcel-1 to BioExcel-2 was a smooth process were the momentum was maintained. This is proven by the ambitious milestones presented in the initial roadmap for the Convergence of HPC/HPDA and Improved Usability Work Package. We are looking forward to present the first results from this roadmap in the coming deliverables *First release of workflow-ready building blocks library* (D2.2 - PM12) and *First release of demonstration workflows* (D2.3 - PM18).

6 References

1. Ramon-Cortes, C., et al., *Transparent Orchestration of Task-based Parallel Applications in Containers Platforms.* Journal of Grid Computing, 2018. **16**(1): p. 137-160. https://doi.org/10.1007/s10723-017-9425-z
2. Chen, H., et al., *The rise of deep learning in drug discovery.* Drug Discovery Today, 2018. **23**(6): p. 1241-1250. https://doi.org/10.1016/j.drudis.2018.01.039
3. Vamathevan, J., et al., *Applications of machine learning in drug discovery and development.* Nature Reviews Drug Discovery, 2019. https://doi.org/10.1038/s41573-019-0024-5
4. da Veiga Leprevost, F., et al., *BioContainers: an open-source and community-driven framework for software standardization.* Bioinformatics, 2017. **33**(16): p. 2580-2582. https://doi.org/10.1093/bioinformatics/btx192
5. Gomes, J., et al., *Enabling rootless Linux Containers in multi-user environments: The udocker tool.* Computer Physics Communications, 2018. **232**: p. 84-97. https://doi.org/10.1016/j.cpc.2018.05.021
6. Kurtzer, G.M., V. Sochat, and M.W. Bauer, *Singularity: Scientific containers for mobility of compute.* PLOS ONE, 2017. **12**(5): p. e0177459. https://doi.org/10.1371/journal.pone.0177459
7. Naden, L.N., S. Ellis, and S. Jha *MolSSI and BioExcel Workflow Workshop 2018 Report*. arXiv e-prints, 2019. https://arxiv.org/abs/1905.11863
8. Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić (editors), Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, Matt Scales, Stian Soiland-Reyes, Luka Stojanovic (2016): **Common Workflow Language, v1.0**. Specification, *Common Workflow Language working group*. https://w3id.org/cwl/v1.0/ https://doi.org/10.6084/m9.figshare.3115156.v2
9. Balasubramanian, V., et al. *RADICAL-Cybertools: Middleware Building Blocks for Scalable Science*. arXiv e-prints, 2019. https://arxiv.org/abs/1904.03085
10. Babuji, Y., et al. *Parsl: Pervasive Parallel Programming in Python*. *HPDC '19: Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, 2019. https://doi.org/10.1145/3307681.3325400
11. Irrgang, M.E., J.M. Hays, and P.M. Kasson, *gmxapi: a high-level interface for advanced control and extension of molecular dynamics simulations.* Bioinformatics, 2018. **34**(22): p. 3945-3947. https://doi.org/10.1093/bioinformatics/bty484
12. Jain, A., et al., *FireWorks: a dynamic workflow system designed for high-throughput applications.* Concurrency and Computation: Practice and Experience, 2015. **27**(17): p. 5037-5059. https://doi.org/10.1002/cpe.3505
13. Elofsson, A., et al., *Ten simple rules on how to create open access and reproducible molecular simulations of biological systems.* PLOS Computational Biology, 2019. **15**(1): p. e1006649. https://doi.org/10.1371/journal.pcbi.1006649

14. O'Boyle, N.M., et al., *Open Babel: An open chemical toolbox.* Journal of Cheminformatics, 2011. **3**(1): p. 33. https://doi.org/10.1186/1758-2946-3-33
15. Sousa da Silva, A.W. and W.F. Vranken, *ACPYPE - AnteChamber PYthon Parser interfacE.* BMC Research Notes, 2012. **5**: p. 367-367. https://doi.org/10.1186/1756-0500-5-367
16. Gelpí, J.L., et al., *Classical molecular interaction potentials: improved setup procedure in molecular dynamics simulations of proteins.* Proteins, 2001. **45**(4): p. 428-37. https://doi.org/10.1002/prot.1159
17. Mart, et al., *TensorFlow: a system for large-scale machine learning*, in *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*. 2016, USENIX Association: Savannah, GA, USA. p. 265-283. https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf
18. Pedregosa, F., et al., *Scikit-learn: Machine Learning in Python.* J. Mach. Learn. Res., 2011. **12**: p. 2825-2830. https://www.jmlr.org/papers/v12/pedregosa11a.html
19. Gapsys, V., et al., *pmx: Automated protein structure and topology generation for alchemical perturbations.* Journal of Computational Chemistry, 2015. **36**(5): p. 348-354. https://doi.org/10.1002/jcc.23804
20. Bonomi, M., et al., *PLUMED: A portable plugin for free-energy calculations with molecular dynamics.* Computer Physics Communications, 2009. **180**(10): p. 1961-1972. https://doi.org/10.1016/j.cpc.2009.05.011
21. Grant, B.J., et al., *Bio3d: an R package for the comparative analysis of protein structures.* Bioinformatics, 2006. **22**(21): p. 2695-2696. https://doi.org/10.1093/bioinformatics/btl461
22. Michaud-Agrawal, N., et al., *MDAnalysis: A toolkit for the analysis of molecular dynamics simulations.* Journal of Computational Chemistry, 2011. **32**(10): p. 2319-2327. https://doi.org/10.1002/jcc.21787
23. Rodrigues, J.a.T., JMC and Trellet, M and Bonvin, AMJJ, *pdb-tools: a swiss army knife for molecular structures [version 1; peer review: 2 approved].* F1000Research, 2018. **7**(1961). https://doi.org/10.12688/f1000research.17456.1
24. Cock, P.J.A., et al., *Biopython: freely available Python tools for computational molecular biology and bioinformatics.* Bioinformatics, 2009. **25**(11): p. 1422-1423. https://doi.org/10.1093/bioinformatics/btp163
25. Šali, A. and T.L. Blundell, *Comparative Protein Modelling by Satisfaction of Spatial Restraints.* Journal of Molecular Biology, 1993. **234**(3): p. 779-815. https://doi.org/10.1006/jmbi.1993.1626

Appendix

***Table A1**. List of currently available BioExcel Building blocks (June 2019 release). Building blocks developed during the first period of BioExcel-2 (PM1-6) are highlighted in blue.*

Block group	Block Id	Wrapped software	Functionality description
biobb_common		biobb Base structure & common elements	
biobb_template		Generic template to build new blocks	
biobb_io	Pdb	API Call	Downloads a PDB file from the RCSB or MMB REST APIs
	MmbPdbVariants	API Call	Creates a text file containing a list of all the variants mapped to a RSCB PDB code from the corresponding UNIPROT entries.
	MmbPdbClusterZip	API Call	Creates a zip file containing all the PDB files in the given sequence similarity cluster percentage of the given RSCB PDB code
biobb_model	FixSideChain	in house using biopython	Reconstructs the missing side chains and heavy atoms of the given PDB file
	Mutate	in house using biopython	Creates a new PDB file performing the mutations given in a list of amino acid mutations to the input PDB file.
biobb_md	Pdb2gmx	gmx pdb2gmx	Creates a compressed (ZIP) GROMACS topology (TOP and ITP files) from a given PDB file.
	Editconf	gmx editconf	Creates a GROMACS structure file (GRO) adding the information of the solvent box to the input structure file.
	Genion	gmx genion	Creates a new compressed GROMACS topology adding ions until reaching the desired concentration to the input compressed GROMACS topology.
	Genrestr	gmx genrestr	Creates a new GROMACS compressed topology applying the indicated force restrains to the given input compressed topology.
	Grompp	gmx grompp	Creates a GROMACS portable binary run input file (TPR) applying the desired properties from the input compressed GROMACS topology.
	Mdrun	gmx mdrun	Performs molecular dynamics simulations from an input GROMACS TPR file.
	MakeNdx	gmx make_ndx	Creates a GROMACS index file (NDX) from an input selection and an input GROMACS structure file.
	Solvate	gmx solvate	Creates a new compressed GROMACS topology file adding solvent molecules to a given input compressed GROMACS topology file.
	Ndx2resttop	in house	Creates a new GROMACS compressed topology applying the force restrains to the input groups in the input index file to the given input compressed topology.
biobb_analysis	GMXCluster	gmx cluster	Creates cluster structures from a given GROMACS compatible trajectory.
	GMXRms	gmx rms	Performs a Root Mean Square deviation (RMSd) analysis from a given GROMACS compatible trajectory.
	GMXRgyr	gmx gyrate	Computes the radius of gyration (Rgyr) of a molecule about the x-, y- and z-axes, as a function of time, from a given GROMACS compatible trajectory.
	GMXEnergy	gmx energy	Extracts energy components from a given GROMACS energy file.
	GMXImage	gmx trjconv	Corrects periodicity (image) from a given GROMACS compatible trajectory file.

biobb analysis	GMXTrjconvStr	gmx trjconv	Converts between GROMACS compatible structure file formats and/or extracts a selection of atoms.
	GMXTrjconvStrEns	gmx trjconv	Extracts an ensemble of frames containing a selection of atoms from GROMACS compatible trajectory files.
	GMXTrjconvTrj	gmx trjconv	Converts between GROMACS compatible trajectory file formats and/or extracts a selection of atoms.
	Average	AmberTools cpptraj	Calculates a structure average of a given cpptraj compatible trajectory.
	Bfactor	AmberTools cpptraj	Calculates the Bfactor fluctuations of a given cpptraj compatible trajectory.
	Rms	AmberTools Cpptraj	Calculates the Root Mean Square deviation (RMSd) of a given cpptraj compatible trajectory.
	Rmsf	AmberTools cpptraj	Calculates the Root Mean Square fluctuations (RMSf) of a given cpptraj compatible trajectory.
	Rgyr	AmberTools cpptraj	Computes the radius of gyration (Rgyr) from a given cpptraj compatible trajectory.
	Dry	AmberTools cpptraj	Dehydrates a given cpptraj compatible trajectory stripping out solvent molecules and ions.
	Strip	AmberTools cpptraj	Strips a defined set of atoms (mask) from a given cpptraj compatible trajectory.
	Snapshot	AmberTools cpptraj	Extracts a particular snapshot from a given cpptraj compatible trajectory.
	Slice	AmberTools cpptraj	Extracts a particular trajectory slice from a given cpptraj compatible trajectory.
	Convert	AmberTools cpptraj	Converts between cpptraj compatible trajectory file formats and/or extracts a selection of atoms or frames.
	Mask	AmberTools cpptraj	Extracts a selection of atoms from a given cpptraj compatible trajectory.
	Image	AmberTools cpptraj	Corrects periodicity (image) from a given cpptraj trajectory file.
biobb chemistry	AcpypeParamsAC	ACPype	Small molecule parameterization for AMBER MD package.
	AcpypeParamsCNS	ACPype	Small molecule parameterization for CNS/XPLOR MD package.
	AcpypeParamsGMX	ACPype	Small molecule parameterization for GROMACS MD package.
	AcpypeParamsGMXOPLS	ACPype	Small molecule parameterization for OPLS/AA MD package.
	BabelConvert	OpenBabel	Small molecule format conversion.
	BabelAddHydrogens	OpenBabel	Adds hydrogen atoms to small molecules.
	BabelRemoveHydrogens	OpenBabel	Removes hydrogen atoms to small molecules.
	BabelMinimize	OpenBabel	Energetically minimize small molecules.
	ReduceAddHydrogens	AmberTools Reduce	Adds hydrogen atoms to small molecules.
	ReduceRemoveHydrogens	AmberTools Reduce	Removes hydrogen atoms to small molecules.

***Table A2**. List of computational biomolecular simulation workflow managers presented in the state of the art analysis.*

Workflow Manager/Language	Main Infrastructure	Main functionality	Short Description
Common Workflow Language	Docker containers	Workflow description/ specification	Workflow specification language for describing workflows and tools that are portable and scalable for execution across a variety of software and hardware environments
RADICAL-Cybertools	HPC, non-MPI	Ensemble simulations	Workflow system that enables the execution of ensemble-based applications on a variety of high performance computing infrastructures
Parsl	HPC, MPI	Data-oriented workflows	Python library that allows external applications to be connected by shared input/output data objects into flexible parallel workflows
gmxapi	HPC, MPI	Ensemble runs with GROMACS engine modifications	Python wrapper which links to GROMACS that presents an abstract interface to building and executing computational graphs allowing transparent low-level optimization of data flow and task placement
Crossbow and Crossflow	Cloud environments, HPC	Cloud-based computing for biomolecular simulations	Python-based toolkit for workflow construction and execution, aimed mainly at distributed computing environments
AdaptiveMD	HPC, MPI	MD Adaptive sampling	Python package designed to create HPC-scale workflows (parallel tasks) for adaptive sampling of biomolecular MD simulations
FireWorks	HPC, MPI	GUI to execute, monitor and track provenance	Tool for defining, managing, and executing complex workflows via Python, JSON, or YAML, stored using MongoDB, and monitored through a built-in web interface

***Table A3**. Summary of proposed tools to be wrapped in new BioExcel Building blocks (bbs) during the BioExcel-2 project. Tools already used in new bbs during the first period of BioExcel-2 (PM1-6) are highlighted in blue.*

biobb_chemistry (new)	ACPype	Software Suite Command line	Chemical toolbox designed to search, convert, analyze or store data from molecular modelling, chemistry, solid-state materials, biochemistry or related areas.
	OpenBabel	Software (Python) Command line	A Python-based tool for Antechamber to generate topologies for chemical compounds.
	AmberTools Reduce	Software Suite	Reduce is a program for adding hydrogens to a Protein DataBank (PDB) molecular structure file, included in the AmberTools package.
	CMIP	Software (Fortran) Command line	Package to compute classical molecular interaction potentials (electrostatic, VdW, solvation), perform protein-ligand docking simulations, and predict water and ion positions on the surface of macromolecules.
	RDKit	Software library (Python/C++)	RDKit is a collection of cheminformatics and machine-learning software written in C++ and Python.
biobb_hpda (new)	dislib	Python Library	dislib is a distributed computing library highly focused on machine learning on top of PyCOMPSs. Inspired by NumPy and scikit-learn, dislib provides various supervised and unsupervised learning algorithms through an easy-to-use API.
	TensorFlow	Python Library	Free and open source software library for dataflow and differentiable programming to develop and train Machine Learning models.
	scikit-learn	Python Library	Open source Python library for data mining, data analysis, and Machine Learning.
	pyTorch	Software library (Python/C++)	Open source deep learning platform providing a seamless path from research prototyping to production deployment.
	keras	Python Library	Open source Python library providing high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
biobb_md (biased & advanced)	pmx	Software (Python) Command line	pmx is a Python-based software package providing utilities for handling biomolecular structure and topology files that are directly compatible with GROMACS, with hybrid structure and topology generation for alchemical single topology-based free energy simulations.
	plumed	Software Command line	Open source, community-developed library providing enhanced-sampling algorithms, free-energy methods, and MD trajectories analyses.
	gromacs	Software Command line	GROMACS is a molecular dynamics package mainly designed for simulations of proteins, lipids, and nucleic acids, and is one of the main codes in the BioExcel CoE.
biobb_analysis (extension)	AmberTools	Software Suite Command line	Suite of AMBER free-of-charge utilities to work with molecular dynamics simulations: build molecules, setup systems, analyse trajectories, etc.
	Bio3D	R library	R package containing utilities for the analysis of protein structure, sequence and trajectory data.
	MDAnalysis	Python Library	MDAnalysis is an object-oriented Python library to analyze trajectories from molecular dynamics (MD) simulations in many popular formats.

biobb_io (extension)	IRB DBs APIs (not available yet)	REST API	REST API libraries for programmatic access to IRB databases offering atomistic MD trajectories (MoDEL v2.0), small molecule conformations (Bioactive Compounds), and protein conformational transitions trajectories (TransAtlas).
	PDBe API	REST API	PDBe is the European resource for the collection, organisation and dissemination of data on biological macromolecular structures. This API is based on Neo4J.
Biobb_model (extension)	pdb-tools	Software (Python) Command line	A “swiss army knife” for the PDB format, including utilities to select/extract sections from a PDB file or renumber atoms/residues.
	biopython	Python Library	Biopython is a set of freely available tools for Computational Molecular Biology written in Python.
	MODELLER	Software (Python)	Program for comparative protein structure modeling by satisfaction of spatial restraints.