Practical challenges for biomedical modeling using HPC.

David W. Wright[1], Robin A. Richardson[1] and Peter V. Coveney[1]

Abstract— **The concept underlying precision medicine is that prevention, diagnosis and treatment of pathologies such as cancer can be improved through an understanding of the influence of individual patient characteristics. Predictive medicine seeks to derive this understanding through mechanistic models of the causes and (potential) progression of diseases within a given individual. This represents a grand challenge for computational biomedicine as it requires the integration of highly varied (and potentially vast) quantitative experimental datasets into models of complex biological systems. It is becoming increasingly clear that this challenge can only be answered through the use of complex workflows that combine diverse analyses and whose design is informed by an understanding of how predictions must be accompanied by estimates of uncertainty. Each stage in such a workflow can, in general, have very different computational requirements. If funding bodies and the HPC community are serious about the desire to support such approaches, they must consider the need for portable, persistent and stable tools designed to promote extensive long term development and testing of these workflows. From the perspective of model developers (and with even greater relevance to potential clinical or experimental collaborators) the enormous diversity of interfaces and supercomputer policies, frequently designed with monolithic applications in mind, can represent a serious barrier to innovation. Here we use experiences from work on two very different biomedical modeling scenarios - brain bloodflow and small molecule drug selection - to highlight issues with the current programming and execution environments and suggest potential solutions.**

I. INTRODUCTION

The goal of computational biomedicine is to generate insights from complex mathematical models of human biology in order to influence the development of new and existing therapies. This endeavor naturally encompasses the integration of data and understanding of phenomena from a huge range of temporal and spatial scales (from chemical reactions to the whole human and even up to the level of populations). The application of such modeling in the context of precision medicine requires that sufficient detail is included in the models to differentiate individual patients. The source of differentiation is determined by both the disease of interest and the type of intervention that is being considered. For example, the mutations present in a brain tumour are highly likely to influence the selection of a drug while having less impact on the efficacy of surgery to remove the tumour. Here we argue that developing simulation and modeling in this varied and challenging field is about more than method development and improving code performance and scalability. Focusing on optimizing *time-to-completion*, while undoubtedly useful, nevertheless risks ignoring the *time-to-insight* which is of far greater interest to scientists. We start by outlining specific issues facing computational biomedicine and putting them in the context of the HPC ecosystem illustrated with descriptions of the impact on our own work (in brain bloodflow and small molecule drug selection).

Large and diverse input data - The data required to specify a system of interest can span many orders of magnitude in size, and parameter dimensionality can be equally varied. This naturally results in very different computational requirements between preprocessing of the input, and subsequent simulation. It is a dirty secret of many computational modeling workflows that the preprocessing of input data can be as, or more, computationally demanding as simulations themselves. Furthermore, automation requires the reliable application of data scrubbing/sanitisation, and the ability to fail hard and early when "bad" data is detected (ideally with error messages understandable to a clinical or experimental user).

Uncertainty quantification - If the aim is to influence decision-making processes at the clinical or industrial level, it is paramount that the "trust" one can place in a given set of results is quantified, and its sensitivity to different input parameters explored in a systematic and communicable fashion. Validation, verification and uncertainty quantification (UQ) of computational models is increasingly recognized as essential to producing "certified" and therefore actionable results.

Complex workflows - A typical biomedical simulation will not be a single step but usually a workflow, ingesting data and preprocessing to specify the system of interest, followed by large scale simulation(s) and then analysis. Such directed acyclic workflows are common as the base description of a calculation, but dealing with uncertainty quantification, adaptive sampling or model failure in a robust fashion can lead to far more complex workflows, containing conditional steps or other decision making.

The advent of exascale facilities is often discussed in terms of monolithic applications. This focus tends to encourage large, one-off simulations rather than workflows or multi-simulation sampling approaches (where statistical ensembles are used to ensure more complete model exploration). By definition such calculations are expensive and hard to repeat, which can lead them to contribute to the "reproducibility crisis" [1]. Furthermore, at larger core counts high communication costs mean that ensemble simulations (necessary for UQ) often represent a more efficient use of compute time.

In this context, the ecosystem of software for workflow

*This work was supported by the EU projects EXDCI-2, CompBioMed, COMPAT & EOSC-hub

[1]Centre for Computational Science, Department of Chemistry, UCL, 20 Gower Street, London, WC1H 0AJ. dave.wright at ucl.ac.uk

development and execution can be of greater importance to the general exploitation of computational precision medicine than the scaling of monolithic applications. For example, the transfer of data between two stages of the workflow may be better achieved through a persistent memory object than through interaction with the I/O system, particularly in the case of enormous files or datasets. The focus on headline grabbing core counts exacerbates the existing tendency for funding bodies to focus on capital expenses (i.e. hardware) and to neglect integration and services.

Interaction with non-typical HPC users - Unlike many areas of, for example, physics and engineering, the utility of biomedical computing comes through its integration with the expertise of clinicians and biologists, who typically not only lack HPC knowledge but may have no familiarity with modeling and simulation at all. For such communities the scientific applicability and robustness of output are more important than the innovation of methodology. This innovation, when resulting in frequent changes to existing workflow tools, may in fact hamper such a community's efforts in this domain.

II. DIVERSITY OF CHALLENGES EXPERIENCED

As members of CompBioMed, an EU funded Centre of Excellence in HPC, we have been exposed to the wide range of codes and modeling approaches currently being employed in computational biomedicine. Part of the project remit is to understand the diverse needs of the community and engage with HPC providers to best accommodate them. Here we give details of two exemplar applications from the project with different use cases and our experience in developing, using and porting them to a variety of supercomputers.

A. *Binding Affinity Calculator*

The Binding Affinity Calculator (BAC) [2] automates system building, execution and analysis of molecular dynamics (MD) simulations in order to compute the strength with which drugs bind to their target protein. Whilst BAC supports a range of simulation protocols and analysis methods, the workflow is common to all. A number of 'replica' simulations of the same protein-drug complex are initiated from a single input structure, each consisting of a number of consecutive steps; minimization, equilibration and production. Once the simulation is complete an analysis step is executed (typically much less computationally demanding than the MD). The use of multiple replica simulations is used as both a sampling strategy and to provide uncertainty estimates.

BAC can be used to provide personalized estimates of drug binding based on the genetic sequence of proteins within a patient (or pathogen), and also in drug discovery scenarios. In both cases large numbers of runs may be required, either to untangle the interactions of mutations or to scan a large chemical space. Typically BAC based MD simulations use less than a few hundred cores (or a node's worth plus a GPU) and require 6 to 12 hours to complete. Policies at most supercomputers (we have direct recent experience of ARCHER in the UK, SuperMUC in Germany, and Titan and Bluewaters in the US) force such jobs to be bundled together in order to run at the required scale. This is due to either limitations on the number of jobs allowed per user in the queue (this is mitigated to a degree through array jobs in some instances) or a requirement for a particular job size to allow runs of sufficient duration for the completion of the simulations. In order to facilitate these runs we have recently developed HTBAC [3], [4], a middleware solution based on RADICAL CyberTools [5] that standardizes the way we manage replicas across the supercomputers to which we have access. Furthermore, by using a workflow middleware we have the potential to use adaptive execution patterns (for example terminating simulations once convergence is reached and using the freed cores for other systems). Some of the challenges in creating this flexible middleware have been finding a solution to having a message broker accessible to compute nodes, deletion of required stack elements and unexpected updates of system libraries.

B. *HemeLB*

HemeLB [6], [7] is a 3D computational fluid dynamics solver that employs the lattice-Boltzmann method (LBM), optimized for the highly sparse geometries of the cerebral vasculature. Images taken of a patient's brain during X-ray CT scans (or, less commonly, MRI scans) are segmented and combined to form an estimate of the 3D surface of the blood vessel network. Artifact identification and removal, and surface smoothing, are performed at this stage, often with partial human intervention. A flow-diverting stent may also be introduced into this mesh. In preparation for simulation with LBM, the interior of the surface is discretized at the desired resolution (often 10 μm or lower). This meshing process can carry a high computational cost, often at a significant fraction of the simulation phase cost. Load-balancing during the LBM simulation phase is complex for a sparse system (especially if magnetic drug particles are introduced [8]), and a simpler block decomposition scheme is often preferred for large systems due to the relative costs of calculating more optimal load distributions. Finally, the flow-field and vessel wall shear stress patterns are analyzed, with visualization playing an important part of communicating the result to, say, a clinical user.

Each step in the workflow can and does have very different resource requirements, and data transfer between the steps can be costly if done via I/O (in the case of the very high resolution models that may be needed for clinical accuracy). These differing computational demands, particularly in terms of core counts, mean that it is not always possible or practical to launch a given workflow in a single job submission, and instead extra queuing time is incurred via the scheduling of multiple sequential jobs.

Development of the simulation code as well as the workflow presents its own challenges, with respect to the portability and installation across multiple supercomputing platforms, and the different ways in which errors are caught or communicated by the local machine during runtime.

III. WORKFLOW REQUIREMENTS

The development, validation and verification of computational biomedicine workflows represents a significant investment. The portability and repeatability of the workflow is therefore essential to keeping costs manageably low. Unfortunately, developers currently face portability issues across both space (running on different computers with different architecture and policies) and time (system updates on the same computer, affecting repeatability). The spatial aspect is of particular importance in decisions of whether to rely on particular middleware tools which may prove time-saving on a given supercomputer, while being completely or partially prohibited by security policies on another. The lack of standards in this area adds to reticence among application developers to use certain types of middleware, such as workflow engines. On a related note, the third-party nature of middleware means that the responsibility for support lies neither with the user nor with the supercomputing staff. One may not wish to invest in development of a workflow that depends on one or more third party tools which may or may not still be maintained in two or three years time. Unfortunately, the time scale for development of such workflows is at least of order several years.

A. *Repeatability*

For any software solution to be deemed production ready it must at the very least meet the criteria that it can be run more than once using valid inputs. This may seem like an absurdly low bar, but for many scientific code bases on HPC resources if there is a significant time delay between two runs then it may not be reached. One of the major factors of this is the recklessness of user-developers who may update their software without having performed due diligence (for example running integration tests). Thankfully, the increasing awareness of best practices (driven by initiatives like Software Carpentry [9]) and the wider use of version control, this is both becoming less common and more easily recoverable. There are nevertheless still many parts of the software stack that are beyond the user's control, the most obvious of which is the set of libraries maintained by target HPC machines. It is unrealistic to ask either for HPC administrators never to update their modules or for users to track all changes, however it should always be possible for users to find out what has changed and when the change occurred, a requirement which is not generally met. This lack of information is even more of a problem for the developers and maintainers of lower level tools such as middleware.

A second policy decision which frequently impedes repeatability is the combination of having only a subset of storage locations available to compute nodes, and of these being regularly cleared. Whilst this is a sensible (probably necessary) arrangement, complex workflows often require persistent, compute accessible, storage for management software and the option to ask for limited space for this would greatly aid software stability and repeatability.

B. *Reproducibility*

While workflow engines (and associated middleware) can be an important step towards systematically reproducible computational science, the problems with portability across systems, and (more critically) the need for technical support from third party developers (with no guarantee of future long-term development) make their use a less appealing investment than might otherwise be assumed. Other researchers are not likely to test out a published workflow, even if all files and data are provided, if their experience tells them that a time-consuming amount of bug-hunting and tuning will be needed to make it work on their chosen HPC platform.

The systematic calculation of uncertainties in measured values is vital in order to ascertain the reproducibility of any given result. The flow of uncertainty through complex workflows is not easy to track, and the full understanding of this process represents an entire field of research. Consequently portable UQ toolkits are required that make state of the art techniques available to scientists on any platform they run their code.

Key to the ability to understand the causes of seemingly non-reproducible results is knowledge of exactly what was run in the first place. Most non-developers are unaware of which versions of which libraries are being called by the applications they use. It is currently the responsibility of the application developer (often delegated to the workflow tool authors) to augment results directories with some form of metadata capturing the system environment and run parameters at the moment of execution. Doing so in a standard manner across different systems can be painstaking and prone to instabilities resulting from system updates, amongst others. Automatic production of this metadata in a standard (or as close as practicable) form across HPC sites would aid (and thus promote) better reproducibility practices in computational science.

IV. PORTABILITY

The experience of running software on HPC is generally that each supercomputer will require some adaptation of the work to be performed, due to both technical and policy differences. Such changes represent both a barrier to adoption and a possible point of failure. As such, for applications where robustness is paramount, there is a need for standardization of tools and convergence of policies wherever possible to enable as-frictionless-as-possible porting between systems.

A. *Software Environment*

Standardization of workflow tools - A large and diverse range of tools exist for workflow management and code coupling in the scientific domain and beyond [10], [5], [11], [12], [13], [14], [15], [16]. The level of choice can be unhelpful to application and workflow developers, and result in a lot of duplicated effort on the side of the tool designer. A minimal set of standards (such as an API analogous to OpenMPI) could help settle this next layer of abstraction, and give application developers the solid, stable ground on which to build their workflows to be lasting and low-maintenance,

and in the process aid repeatability and reproducibility in the domain of computational science.

Batch queue API standardization / middleware - The precise details of job submission varies considerably between supercomputers. Standardization of API access to batch queuing systems via DRMAA (Distributed Resource Management Application API, www.drmaa.org/) could help with this. A number of middleware tools, such as QCGBroker [10] aim to exploit this, but require installation on the HPC platform in use, and can often come into conflict with local security policies. Whilst the latter concern can be overcome through direct collaboration on a common project, this is not a scalable approach. Further development of common standards agreed between supercomputing centres may help here. Previous efforts at interoperability, for example those based on Globus and UNICORE, have largely been unsuccessful but the current landscape for HPC access is in stark contrast to the simplicity provided by Cloud APIs.

Programming environment - Despite our focus on production workflows, a large part of the computational biomedicine community is engaged in low level development of simulation codes. Of particular relevance to developer-oriented workflows are differences in the programming environment across HPC platforms. These can be partially (though not completely) alleviated through widely-used build systems such as cmake cmake.org, as well as package and environment management systems (such as conda for Python).

Tools for large datasets - A set of standard tools for the large datasets increasingly used in precision medicine workflows (as input, or in intermediary steps) may well become essential soon. Datasets from both next generation sequencing and medical imaging can result in exceptionally large files. For example within HemeLB input data files can be of the order of tens of terabytes. This has resulted in the creation of bespoke file reading code where these files are striped across multiple storage targets to increase bandwidth and allow simultaneous access to the same file for multiple processes. The creation of this code requires multiple workarounds for limitations of libraries such as MPI and system specific optimizations (e.g. of the stripe count). Maintaining this code represents a considerable overhead, taking effort away from scientific endeavours, as well as a source of additional code fragility. Standardized tools or libraries for such tasks would aid better and more robust simulation code development.

B. Platform Policy

Workflow software frequently requires persistent but low-resource-usage processes, for example databases or message brokers to help orchestrate job steps or to store information between them. This represents a challenge as processes on login nodes are terminated if they run for too long or use too much memory, yet connections to external databases can be blocked by firewalls. Availability of nodes for this type of job within the firewall, or less restrictive security policies, could both be of help in this area.

Workflow and middleware layers that enable the pipelining of steps with diverse computational requirements are currently prone to instability as the software and policies of supercomputers can change without adequate warning. Recognizing that supercomputing platforms often sit at the bottom of very complex software stacks deployed by users, and adopting suitable change management policies (for example governed by ITIL) taking this into account, would be invaluable here. Standardization of workflows could further minimize the work involved in terms of reducing the variety of dependencies that need to be maintained.

As supercomputer capacity grows, a wider diversity of queue functionality would help facilitate workflow building without the need for specific tools. Job chaining and array jobs can handle many tasks assuming that there are queues where jobs can be executed - many systems require that large numbers of cores/nodes are used if a step lasts any significant period of time. Whilst this is more relevant for development codes, it nonetheless provides a potentially light weight and sustainable solution.

V. FUTURE DIRECTIONS

Common and commonly used standards - It may also be possible to identify the features most in-demand by application developers through consideration of the plethora of existing workflow tools, and from there to develop a standard API or set of tools that would expose the desired functionality across different HPC sites through shifting responsibility from third party developers to the staff most intimately acquainted with their own machine. It is not sufficient for standards to be defined, they must be made accessible and supported in ways which encourage user (and developer) buy in.

Community managed modules - One potential solution is for community organizations, such as the EU's H2020 funded Centres of Excellence, to maintain central tools that enable workflows in ways appropriate to their user base. This would need to be accompanied by some links to specific HPC centres but not tying maintainers to a specific machine should reduce the "island effect" of each supercomputing centre providing only the environment it wants to users. It may also help middleware developers within the communities to understand the challenges from the point of view of supercomputer administrators. Such modules could be maintained through tools such as EasyBuilder (easybuilders.github.io/easybuild) and Spack (spack.io).

Containerization - Many of the portability issues may be solved (or at least alleviated) through the use of containers. Whilst the most popular containerization solution, Docker www.docker.com/, is not secure for use in multi-user environments, the same images can be deployed using tools such as Singularity (singularity.lbl.gov/) and Shifter (github.com/NERSC/shifter). It is, however, likely that this approach will present its own unique challenges when applied to extremely large or complex HPC systems and applications (as compared with, say, the typical jobs executed on Cloud services).

REFERENCES

[1] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature News*, vol. 533, no. 7604, p. 452, May 2016. [Online]. Available: http://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970

[2] S. K. Sadiq, D. Wright, S. J. Watson, S. J. Zasada, I. Stoica, and P. V. Coveney, "Automated Molecular Simulation Based Binding Affinity Calculator for Ligand-Bound HIV-1 Proteases," *Journal of Chemical Information and Modeling*, vol. 48, no. 9, pp. 1909–1919, Sep. 2008. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ci8000937

[3] J. Dakka, M. Turilli, D. W. Wright, S. J. Zasada, V. Balasubramanian, S. Wan, P. V. Coveney, and S. Jha, "High-throughput binding affinity calculations at extreme scales," *Computational Approaches for Cancer Workshop at SuperComputing17*, 2017. [Online]. Available: http://arxiv.org/abs/1712.09168

[4] J. Dakka, K. Farkas-Pall, V. Balasubramanian, M. Turilli, D. W. Wright, S. Wan, S. Zasada, P. V. Coveney, and S. Jha, "Rapid, concurrent and adaptive extreme scale binding free energy calculation," *arXiv:1801.01174 [cs]*, Jan. 2018, arXiv: 1801.01174. [Online]. Available: http://arxiv.org/abs/1801.01174

[5] A. Merzky, M. Santcroos, M. Turilli, and S. Jha, "Radical-pilot: Scalable execution of heterogeneous and dynamic workloads on supercomputers," *CoRR, abs/1512.08194*, 2015.

[6] M. Mazzeo and P. Coveney, "HemeLB: A high performance parallel lattice-Boltzmann code for large scale fluid flow in complex geometries," *Computer Physics Communications*, vol. 178, no. 12, pp. 894–914, Jun. 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0010465508000805

[7] D. Groen, J. Hetherington, H. B. Carver, R. W. Nash, M. O. Bernabeu, and P. V. Coveney, "Analysing and modelling the performance of the HemeLB lattice-Boltzmann simulation environment," *Journal of Computational Science*, vol. 4, no. 5, pp. 412–422, Sep. 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1877750313000240

[8] A. Patronis, R. A. Richardson, S. Schmieschek, B. J. N. Wylie, R. W. Nash, and P. V. Coveney, "Modeling Patient-Specific Magnetic Drug Targeting Within the Intracranial Vasculature," *Frontiers in Physiology*, vol. 9, Apr. 2018. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fphys.2018.00331/full

[9] G. Wilson, D. A. Aruliah, C. T. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, and P. Wilson, "Best Practices for Scientific Computing," *PLoS Biology*, vol. 12, no. 1, p. e1001745, Jan. 2014. [Online]. Available: http://dx.plos.org/10.1371/journal.pbio.1001745

[10] B. Bosak, J. Komasa, P. Kopta, K. Kurowski, M. Mamoski, and T. Piontek, "New Capabilities in QosCosGrid Middleware for Advanced Job Management, Advance Reservation and Co-allocation of Computing Resources Quantum Chemistry Application Use Case," ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2012, pp. 40–55. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-28267-6_4

[11] J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, E. Lusk, and I. T. Foster, "Swift/T: Large-Scale Application Composition via Distributed-Memory Dataflow Processing." IEEE, May 2013, pp. 95–102. [Online]. Available: http://ieeexplore.ieee.org/document/6546066/

[12] W. A. Warr, "Scientific workflow systems: Pipeline Pilot and KNIME," *Journal of Computer-Aided Molecular Design*, vol. 26, no. 7, pp. 801–804, Jul. 2012. [Online]. Available: https://link.springer.com/article/10.1007/s10822-012-9577-7

[13] D. K. Brown, D. L. Penkler, T. M. Musyoka, and . T. Bishop, "JMS: An Open Source Workflow Management System and Web-Based Cluster Front-End for High Performance Computing," *PLoS ONE*, vol. 10, no. 8, Aug. 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4539224/

[14] D. Groen, A. P. Bhati, J. Suter, J. Hetherington, S. J. Zasada, and P. V. Coveney, "FabSim: Facilitating computational research through automation on large-scale and distributed e-infrastructures," *Computer Physics Communications*, vol. 207, pp. 375–385, Oct. 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0010465516301448

[15] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services," *Nucleic Acids Research*, vol. 34, no. Web

Server issue, pp. W729–W732, Jul. 2006. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1538887/

[16] E. Bernhardsson, "Luigi presentation NYC Data Science," Dec. 2014. [Online]. Available: https://www.slideshare.net/erikbern/luigi-presentation-nyc-data-science