Journal of
open research software

Gebbie-Rayet, J et al 2016 Longbow: A Lightweight Remote Job Submission Tool.
Journal of Open Research Software, 4: e1, DOI: http://dx.doi.org/10.5334/jors.95

SOFTWARE METAPAPER

Longbow: A Lightweight Remote Job Submission Tool

James Gebbie-Rayet[1,*], Gareth Shannon[2,*], Hannes H. Loeffler[1] and Charles A. Laughton[2]

[1] Scientific Computing Department, STFC Daresbury Laboratory, Warrington WA4 4AD, UK
james.gebbie@stfc.ac.uk, hannes.loeffler@stfc.ac.uk

[2] School of Pharmacy and Centre for Biomolecular Sciences, University of Nottingham, University Park, Nottingham NG7 2RD, UK
gareth.shannon@nottingham.ac.uk, charles.laughton@nottingham.ac.uk

* James Gebbie-Rayet and Gareth Shannon contributed equally to this work

We present Longbow, a lightweight console-based remote job submission tool and library. Longbow allows the user to quickly and simply run jobs on high performance computing facilities without leaving their familiar desktop environment. Not only does Longbow greatly simplify the management of compute-intensive jobs for experienced researchers, it also lowers the technical barriers surrounding high performance computation for the next generation of scientists and engineers. Longbow has already been used to remotely submit jobs in a number of projects and has the potential to redefine the manner in which high performance computers are used.

Keywords: Remote job submission; high performance computing; HPC; molecular dynamics; MD; biosimulation; biomolecular simulation; longbow; hecbiosim; job management; automated job submission
Funding statement: Longbow was developed under EPSRC grant EP/L000253/1 as part of the HECBioSim project (http://www.hecbiosim.ac.uk/).

(1) Overview

Introduction

Much of modern computational science and engineering requires access to high performance compute resources. Notable examples include climate modelling [1] which underpins government environmental policy, jet engine development [2] which increases passenger safety and reduces environmental impact, and molecular dynamics simulations which aid the understanding and treatment of disease [3].

With computer power remaining faithful to Moore's law [4], the games industry delivering massively parallel GPU processors [5] and quantum computing making positive strides [6], society's demand for computationally expensive calculations can only be expected to escalate with time.

Despite the ever-increasing importance of high performance resources, usability remains an issue. HPC resources are typically a) remote; b) require familiarity with an operating environment that may differ significantly from that found on the user's desktop; and c) have job schedulers that differ significantly from one to another. To use such facilities, the average user currently has little option but to, interactively and step-by-step, get through a workflow whose key steps are i) logging in to the remote resource from their desktop, ii) transferring the required input files to the remote resource, iii) writing a job submission script, iv) submitting the job, v) monitoring the job and vi) finally transferring files back to their local workstation to analyse the results. For experienced practitioners this is time-consuming, but for potential new adopters it may be a severe barrier to engagement with HPC-enabled science altogether.

Here we present a software package, Longbow, as a solution to this problem. Longbow is a lightweight console-based remote job submission tool and library that greatly simplifies the process of running a compute job on a high performance resource. It does so by automating the laborious steps outlined above, without requiring the user to leave their familiar desktop environment. Longbow even writes the job submission script and retrieves the results on behalf of the user.

There are a number of benefits of this simplification. First of all, it reduces the time a researcher spends on job managment. Secondly, it allows inexperienced computational researchers to access high performance computers with ease and become productive quickly. Thirdly, as results files are automatically brought back to the local resource, researchers using a number of different remote resources within a single project can effortlessly centralise their results. Finally, operating details of the remote resource are hidden away through a common interface.

The philosophy behind the development of Longbow has been to make the software as simple as possible to use, install and extend. It has been written in "vanilla" python (2.6, 2.7, 3.x) and can be installed without administrative privileges *via* the pip management system (pip install longbow). Furthermore, it has no dependencies on

other packages besides the standard Unix utilities ssh and rsync, making installation much easier for the non-expert. Although "out of the box", Longbow currently only supports Molecular Dynamics packages. New software packages from any branch of computational science can be simply incorporated into the Python code by creating a single short script file as a plug-in. The result is a lightweight, simple and intuitive tool that can be extended in minutes to support the running of new software packages and executables on remote resources.

There are other utilities that facilitate the submission of compute-intensive jobs remotely, however none has quite the same focus as Longbow. For example, the Radical Cybertools suite [7] provides a very flexible and feature-rich framework for running jobs on distributed resources. However, this is best suited to advanced users with technical skills to install the sizeable package and its significant dependencies, and to adapt the relatively low-level interface the suite provides to the needs of their specific research problem. Though Longbow is a simpler product, its reduced learning curve means that for common usage patterns, a user can become productive more quickly.

There are also web-based solutions such as the bioinformatics-focussed Galaxy [8] that allow the submission of jobs through a web interface. Though such approaches can provide a particularly rich and supportive user interface, the challenges of integrating the web portal with a particular HPC resource will typically require the active support of the relevant computing center system administration team, and may well require a relaxation of the facility's security policy that could be difficult to negotiate.

To summarise, to the authors' knowledge Longbow is the only widely accessible console-based tool that allows the remote submission of jobs in a simple fashion.

Use of Longbow in research

Longbow is a new tool that was only released as stable software on 21/06/2015. Nevertheless this has not been an obstacle to the uptake of Longbow by researchers interested in utilising it in their research. Amongst the most notable projects that have developed beyond a scoping phase and into formal development are the following.

Use of Longbow in benchmarking HPCs

One of the functions of the HECBioSim consortium is to provide assistance to members of the bio-molecular simulation community with gaining access to HPC facilities of national level (presently the UK national supercomputer ARCHER). To encourage the most efficient use of this massively parallel resource, the consortium has generated and maintains a detailed database of the weak and strong scaling behavior of the most commonly-used Molecular Dynamics (MD) packages (AMBER [9], CHARMM [10], GROMACS [11], LAMMPS [12] and NAMD [13]).

The database is populated with the results from a wide-ranging set of benchmarking simulations that cover the range of most commonly encountered problem sizes and types (refer to http://purl.org/net/epubs/work/50963 and http://www.hecbiosim.ac.uk/benchmark). Each simulation must be run on a range of core counts, and using each of the supported MD packages. Running this suite is extremely time consuming and editing job submission scripts each time software versions change was a laborious task. Applying Longbow to this task has removed the need to continuously update hundreds of job submission scripts when testing new software. The Longbow-based version of the benchmark suite is also completely portable and can be used to comprehensively profile any supported HPC facility, with any release of supported software, in little more than the run time required for the simulations.

Use of Longbow in FLEX-EM

Longbow has been incorporated into the developing CCP-EM software suite for biological electron cryo-microscopy [14]. Several steps in the interpretation of micrographs require significant compute resources, such as recent Bayesian methods for the 3D reconstruction of molecular volumes, and Longbow provides the ability to submit computationally intensive jobs to remote clusters from a local instance of the CCP-EM graphical user interface. Longbow has been incorporated as a generic utility, but in the first instance has been tested on the FlexEM application [15] which flexibly fits atomic models into low resolution volumes.

Implementation and architecture

Longbow consists of a standalone application and a generically written core library. Here we will give an overview of the basic features and functionality of the application, followed by a description of what is available within the core library and plug-ins.

Longbow application

Executing Longbow

To run jobs on a high performance remote resource, a Longbow job is run either interactively or in the background on the user's desktop resource, or is submitted to the user's local batch queue system.

As mentioned previously, the Longbow application has been designed to be as simple to run as possible. In many instances, by simply placing the word "longbow" in front of a command string that would run the desired job on the local resource, the same job will execute on the remote resource. The full format of the Longbow command is displayed in **Figure 1**.

Once executed, Longbow will create a job submission script, copy this script along with required input files to the remote resource, submit the job, keep track of the progress of the job and periodically bring the output files back to the local machine.

Source of the simplicity

A major reason submitting jobs is so simple is that much of the extra information Longbow requires to describe how and where to run the remote jobs is stored in one, or optionally two, configuration files in .ini file format: the hosts configuration file and the job configuration file. The hosts configuration file is used primarily to store information about the remote resource such as username and

```
% longbow [longbow arguments] executable [executable arguments]
```

Figure 1: The simple format of the Longbow command.

account details, but can also provide default values for parameters such as numbers of cores to be used or wall-time limit. The job configuration file provides a mechanism to override the default values of any parameter for the current project, while these again may be overridden by arguments specified on the command line to give job-specific control.

Another reason submitting jobs is so simple is because Longbow will automatically detect input files that should be copied to the remote resource to be used by the executable. The user does not have to provide a list.

Job types

The Longbow application supports three types of jobs to be run on the remote resource: single, replicate and multijobs.

Single jobs involve just one calculation: a single executable, a single set of input data, and all running on a single compute resource. Replicate jobs involve the more-or-less synchronous running of a single executable against a variety of related input data sets, all on a single resource.

A distinguishing feature of Longbow is its ability to simultaneously initiate several jobs on different remote resources at a single keystroke by running what is known as a multijob. Multijobs can be easily set up by providing details of all the jobs to be run in separate sections of a job configuration file. A good example of when this would be useful is the aforementioned FLEX-EM project. In this case, the multijobs feature allowed the simultaneous submission of many jobs that utilised several different codes on several different resources. Without Longbow, achieving this would have required a more complex solution.

Longbow architecture

Longbow core library

The Longbow core library (corelibs) contains generically written methods that provide the functionality behind Longbow. The core library consists of procedurally written code, which is categorised into distinct python modules based upon the nature of the method. This ensures that source files stay relatively short in length and that developers wishing to incorporate Longbow can choose which aspects of the core library they wish to use.

The methods within the core library have been engineered in such a way that only two main data structures are required for them to operate. These main data structures consist of python dictionaries, which form a logical distinction between data that describes a job and that which describes a host (HPC resource). In the Longbow application these structures are initialised within the configuration methods of the core library. It is here that developers can find the template structures and logic for setting up such structures should they wish to produce a custom solution.

The following outline gives a brief overview of which methods can be found within the core library. However, more in depth information can be gleaned from either comments within the code or from the developer's documentation on the HECBiosim wiki [16].

applications.py

testapp()	Checks that executables exist on the HPC machine
processjobs()	Process jobs to extract a list of files for staging and an execution command-line. This method contains hooks to call methods that are provided via the plug-in framework (such as custom input file parsers).

configuration.py

processconfigs()	Processes configuration information from sources such as command-line and configuration files into the two main Longbow data structures.
loadconfigs()	A method for loading configuration files (ini format)
saveconfigs()	A method for saving configuration files (ini format)

exceptions.py Contains various custom exception classes which are used in the error handling within Longbow methods.

logger.py

setuplogger()	A method that sets the correct logging mode.
standardlogger()	The standard logger method, configures python logger to log standard events to specified file only.
verboselogger()	The verbose logger method, configures python logger to log standard events to the specified file and console output.
debuglogger()	The debug logger method, configures python logger to log an enhanced set of log messages to the log file and console output.

scheduling.py

testenv()	This method will check the scheduling environment for all hosts referenced in jobs if not already set by user, and make an attempt to auto determine if it is not.
delete()	A generic job delete method, this method makes a call to the environment specific delete method that are provided via plug-ins.
monitor()	The generic method for monitoring jobs after submission, this method

	contains a loop that continues until all jobs have ended. This method contains the logic for persistent staging should this have been configured. Calls are made to environment specific methods that are provided via plug-ins.
prepare()	The generic method for writing the job submission file, this method makes calls to the environment specific method provided via the plug-in framework.
submit()	The generic method for submitting jobs, this method makes calls to environment specific submit methods provided via the plug-in framework.

shellwrappers.py

testconnections()	A method for testing that SSH connections can be established with each HPC host listed under jobs.
sendtoshell()	A method for assembling commands to be sent to the shell via python subprocess calls.
sendtossh()	A method for assembling commands to be sent to the shell which will utilise SSH.
sendtorsync()	A method for assembling commands to be sent to the shell which will utilise rsync.
localcopy()	A method for copying files/directories between local paths.
localdelete()	A method for deleting files/directories from local paths.
locallist()	A method for listing the contents of local paths.
remotecopy()	A method for copying files/directories between locations on a remote host.
remotedelete()	A method for deleting files/directories from a remote host.
remotelist()	A method for listing the contents of paths on a remote host.
upload()	A method for uploading files from a local path to a path on a remote host.
download()	A method for downloading files from a path on a remote host to a local path.

staging.py

stage_upstream()	Method for staging all files from a set of jobs to their respective HPC hosts.
stage_downstream()	Method for staging all files for a particular job from the HPC host back to the local machine.
cleanup()	Method for cleaning up the working directories of each HPC host listed in completed jobs.

An example of how the core library should be utilised can be seen in **Figure 2**, this diagram outlines how the core library is used within the Longbow application executable.

Longbow plug-ins

Longbow plug-ins contain code that interfaces with the core library to support both the executables to be run on the remote resource and the schedulers to which the jobs are submitted. As such the code is housed in two libraries: apps and schedulers.

plugins.schedulers

Each supported scheduler is defined in a python file named after the scheduler. For example, PBS is supported by Longbow in file pbs.py.

The following outline gives a brief overview of which methods can be found within each supported scheduler file. In every case the plug-ins methods are called by the methods of the same name in scheduling.py in the core library. More in depth information can be gleaned from either comments within the code or from the developer's documentation on the HECBiosim wiki [16].

prepare()	This method writes the job submission file to be submitted to the scheduler.
delete()	This method will delete a job that has been submitted to the scheduler in question on the remote resource.
submit()	This method submits the job submission file to the scheduler in question on the remote resource.
status()	This method will query the status of jobs that have been submitted to the scheduler.

plugins.apps

Each supported app (executable) is defined in a python file named after the software. For example, molecular dynamics software CHARMM is supported by Longbow in file charmm.py.

The following outline gives a brief overview of which methods and dictionaries that may be found within each the supported app file. Those that are required vary depending on the requirements of the software package. More in depth information can be gleaned from either comments within the code or from the developer's documentation on the HECBiosim wiki [16].

EXECDATA	This dictionary defines the names of the supported executables for the package and the command line flags the software requires. This dictionary is required in all app files.
file_parser()	This method recursively searches through input files to the executable for references to other required input files. All filenames found are added to the list of files to be staged to the

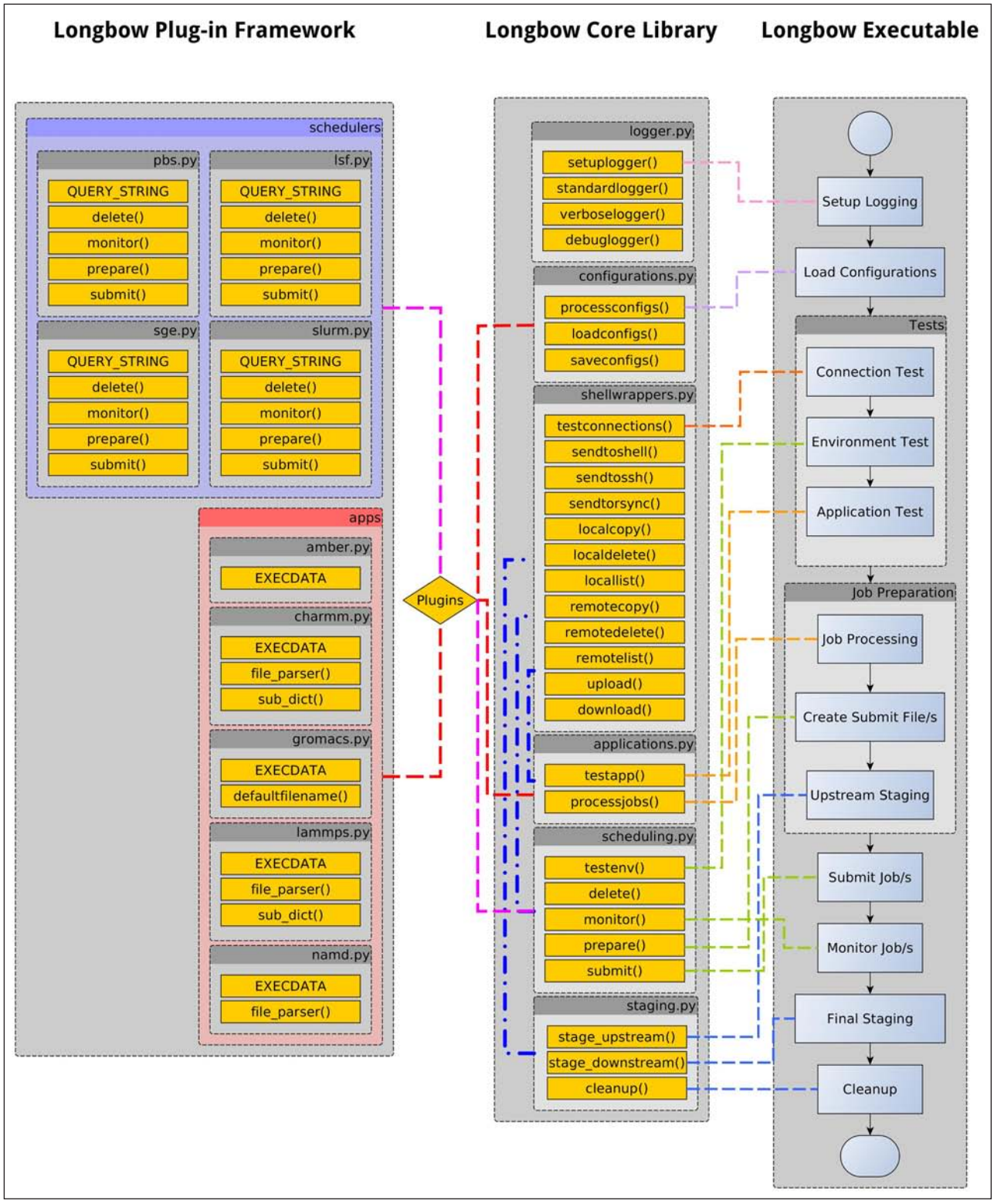

Figure 2: A schematic showing how the flowchart for the Longbow executable (right) maps onto the relationship diagram of the core library and plug-in framework. The colour coding is simply for clarity.

remote resource. Only executables that can depend on input files that are not explicitly provided on the command line require this method.

sub_dict() This method detects command-line parameter substitutions to be applied in input files. Only packages that support such substitutions and users that wish to implement such a feature require this method.

defaultfilename() This method will automatically add the file extension specified in the method onto the name of an input file provided without the extension. This method is to support the atypical case that a package might expect

the name of an input file to be provided without the extension.

For Longbow to support a new package, often just a python dictionary needs to be provided which highlights the extensibility of the software. If necessary or desired, the methods outlined above can also be provided to use a given software package in a more complex fashion.

Quality control

To ensure that Longbow conformed to good quality control standards we conducted alpha and beta testing as well as developing a suite of tests.

Alpha and beta testing

Alpha and beta testing: Alpha testing was done in-house using colleagues ranging from junior PhD students to experienced postdoctoral workers with a range of academic backgrounds and technical expertise. Both ease of installation and use of the Longbow package were tested. Following this round of testing, an open beta release was made available to the UK academic community and feedback was collected over a period of 3 months.

Test suite

The test suite has been designed to probe all of the functions within the core library as well as the performance of the standalone Longbow application. The tests are applied to the release code prior to tagging a new release version on the code repository and subsequently publishing a new release. The following types of tests are performed:

Unit Tests: These use the PyUnit testing framework, and are designed to test that changes to methods within the core library have not broken the basic functionality.

Functional Tests: These are designed to test that the standalone Longbow application functions for a standard set of configurations. There are also tests within this suite that force failure to make sure error handling occurs correctly.

Operation Tests: These are designed to probe the basic operation of Longbow. These tests are made up of jobs that run using different MD packages, job types, job configurations, scheduling environment and HPC facilities. These test that Longbow will run to completion full jobs across all supported schedulers and software packages (plug-ins supplied out of the box) and will perform all desired functions a user may request.

(2) Availability

Operating system

Longbow is designed to work best on Linux or Unix based operating systems. It is possible to run Longbow under the Windows operating system using a Unix emulation environment, such as Cygwin or MinGW to bring the SSH and rsync utilities to Windows.

Programming language

Longbow is written in Python and will run natively with versions 2.6, 2.7 and 3.x.

Additional system requirements

Longbow only requires that the machine it is run on has SSH-type connectivity to the HPC facility that is the intended target of use, and that password-less login to that resource can be established. The local machine must also have the standard unix rsync utility.

Dependencies

Longbow has no dependencies outside of the standard Python libraries.

List of contributors

James. T. Gebbie-Rayet, acted as co-principal developer having designed the software and contributed a substantial amount of code, documentation and user-support.

Gareth. Shannon, acted as co-principal developer having contributed a substantial amount of code, documentation and user-support.

Hannes H. Loeffler, designed the software, provided valuable technical guidance throughout the project and contributed many ideas.

Charles. A. Laughton, acted as principal investigator by steering the overall direction of the software and contributing numerous ideas as well as code to the codebase.

Software location

Archive (e.g. institutional repository, general repository) (required)

Name: pypi
Persistent identifier: http://dx.doi.org/10.6084/m9.figshare.1545562
Licence: GNU GPLv2
Publisher: Gareth Shannon
Date published: 21/06/2015

Code repository (e.g. SourceForge, GitHub etc.) (optional)

Name: Bitbucket
Identifier: https://bitbucket.org/jimboid/longbow
Licence: GNU GPLv2
Date published: 21/06/2015

Language

The language used throughout the documentation, code repository, support forums and naming and comments within the code-base is English.

(3) Reuse Potential

Longbow has been designed to be highly re-usable both as a standalone application or by directly integrating the Longbow core library into other software projects.

Users, groups or consortia can easily tailor Longbow for use with software specific to their field of interest simply by providing Longbow with plug-ins for the software they wish to support. Developers can make use of Longbow as a job submission layer embedded within their software. Developers have the freedom to do anything from simply wrapping Longbow as a standalone application through to incorporating the very core library into their own software.

Support

Support for users and developers of all abilities is provided through the forums within the HECBioSim website. We also encourage the reporting of bugs and feature requests through this channel.

Contributing to Longbow

We encourage contributions to the Longbow code base. However, developers wishing to contribute code should discuss this with the Longbow development team prior to submitting any source code. This is to ensure any code submitted conforms to the Longbow mission statement and code reviewing policies.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgements

The authors would like to acknowledge Martyn Winn who is head of Computational Biology at the Science and Technology Facilities Council, UK for his feedback and assistance in writing this manuscript.

References

1. **Palmer, T** 2014 Build high-resolution global climate models. *Nature*, 515(7527): 338–9. DOI: http://dx.doi.org/10.1038/515338a
2. **Sanghi, V, Lakshmanan, B K** and **Sundararajan, V** 2000 Survey of advancements in jet-engine thermodynamic simulation. *Journal of Propulsion and Power*, 16(5): 797–807. DOI: http://dx.doi.org/10.2514/2.5644
3. **Karplus, M** and **McCammon, J A** 2002 Molecular dynamics simulations of biomolecules. *Nature Structural Biology*, 9(9): 646–52. DOI: http://dx.doi.org/10.1038/nsb0902-646
4. **Moore, G E** 1998 Cramming more components onto integrated circuits (Reprinted from Electronics, pp. 114–117, April 19, 1965). *Proceedings of the Ieee*, 86(1): 82–5. DOI: http://dx.doi.org/10.1109/JPROC.1998.658762
5. **Bernaschi, M, Bisson, M** and **Fatica, M** 2015 Colloquium: Large scale simulations on GPU clusters. *European Physical Journal B*, 88(6): 10. DOI: http://dx.doi.org/10.1140/epjb/e2015-60180-8
6. **Gibney, E** 2014 Quantum Computer Quest. *Nature*, 516(7529): 24–6. DOI: http://dx.doi.org/10.1038/516024a
7. **Goodale, T, Jha, S, Kaiser, H, Kielmann, T, Leijer, P K, von Laszewski, G, Lee, C, Merzky, A, Rajic, H** and **Shalf, J** 2006 SAGA: A Simple API for Grid applications, High-Level Application Programming on the Grid. *Computational Methods in Science and Technology*, 12(1).
8. **Goecks, J, Nekrutenko, A, Taylor, J** and **Galaxy, T** 2010 Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(R86). DOI: http://dx.doi.org/10.1186/gb-2010-11-8-r86
9. **Pearlman, D A, Case, D A, Caldwell, J W, Ross, W S, Cheatham, T E, Debolt, S**, et al. 1995 AMBER, a package of computer-programs for applying molecular mechanics, normal-mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Computer Physics Communications*, 91(1–3): 1–41. DOI: http://dx.doi.org/10.1016/0010-4655(95)00041-D
10. **Brooks, B R, Brooks, C L, Mackerell, A D, Nilsson, L, Petrella, R J, Roux, B,** et al. 2009 CHARMM: The Biomolecular Simulation Program. *Journal of Computational Chemistry*, 30(10): 1545–614. DOI: http://dx.doi.org/10.1002/jcc.21287
11. **Pronk, S, Pall, S, Schulz, R, Larsson, P, Bjelkmar, P, Apostolov, R,** et al. 2013 GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 29(7): 845–54. DOI: http://dx.doi.org/10.1093/bioinformatics/btt055
12. **Plimpton, S** 1995 Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1): 1–19. DOI: http://dx.doi.org/10.1006/jcph.1995.1039
13. **Phillips, JC, Braun, R, Wang, W, Gumbart, J, Tajkhorshid, E, Villa, E,** et al. 2005 Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 26(16): 1781–802. DOI: http://dx.doi.org/10.1002/jcc.20289
14. **Wood, C, Burnley, T, Patwardhan, A, Scheres, S, Topf, M, Roseman, A,** et al. 2015 Collaborative Computational Project for Electron cryo-Microscopy. *Acta Crystallographica Section D-Biological Crystallography*, 71: 123–6. DOI: http://dx.doi.org/10.1107/S1399004714018070
15. **Topf, M, Lasker, K, Webb, B, Wolfson, H, Chiu, W** and **Sali, A** 2008 Protein structure fitting and refinement guided by cryo-EM density. *Structure*, 16(2): 295–307. DOI: http://dx.doi.org/10.1016/j.str.2007.11.016
16. **HECBioSim wiki** 2015 Available at http://www.hecbiosim.ac.uk/wikis/index.php/Main_Page.

How to cite this article: Gebbie-Rayet, J, Shannon, G, Loeffler, H H and Laughton, C A 2016 Longbow: A Lightweight Remote Job Submission Tool. *Journal of Open Research Software,* 4: e1, DOI: http://dx.doi.org/10.5334/jors.95

Submitted: 16 September 2015 **Accecpted:** 06 January 2016 **Published:** 27 January 2016

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS