The HemeLB Offloader

Nash, R.W.[1], Bernabeu, M.O.[2], Illingworth[1], M., Sloan, T.M. [1]

[1]EPCC, The University of Edinburgh.

[2]Centre for Medical Informatics, The University of Edinburgh

1. Introduction

This paper outlines an approach for enabling access to HPC applications as Software as a Service (SaaS) on conventional high-end HPC hosts such as EPCC's Cirrus and cloud providers such as Microsoft's Azure service. The focus for the approach is enabling access to the HemeLB application with the Polnet biomedical workflow. The paper reports on an implementation of this approach that allows Polnet workflows to run on the Cirrus and LISA supercomputing services at EPCC and SURFsara respectively.

2. Context

PolNet [1] is an open source software tool for the study of blood flow and biological activity at the single cell level during vessel morphogenesis. It provides an image processing and analysis workflow for entire in vivo vascular networks that provides quantitative estimates of the endothelial cell polarity and haemodynamic forces due to blood flow. The tool enables, for the first time, network-level statistical analysis of polarity and flow for individual endothelial cells, enabling study of endothelial cell polarisation and migration during vascular patterning, as demonstrated by recent publications [2], [3]. See Figure 1 for an example of analysis.

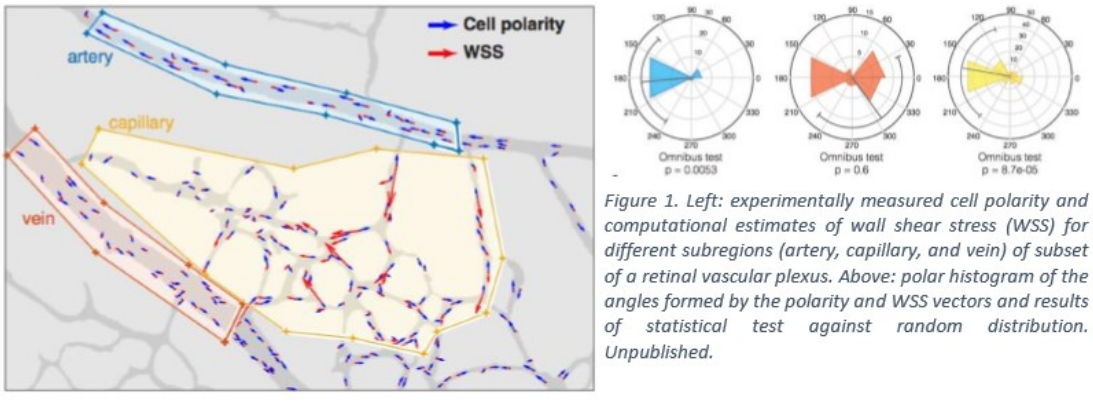

Figure 1. Left: experimentally measured cell polarity and computational estimates of wall shear stress (WSS) for different subregions (artery, capillary, and vein) of subset of a retinal vascular plexus. Above: polar histogram of the angles formed by the polarity and WSS vectors and results of statistical test against random distribution. Unpublished.

This software currently runs within a Docker container, which can be easily run by our biologist collaborators, who do not have to manage the burden of installing the many dependencies required. The software is relatively easy to use, a GUI hiding the command-line based interface of the components.

3. Aim

PolNet runs on a single workstation, which, for much of the analysis, is an appropriate platform. However, estimating the forces due to the blood flow requires running a computational fluid dynamics (CFD) simulation, which on a single workstation, can take several days to complete. We use the HemeLB (https://github.com/UCL/hemelb) application for this work, which can take full advantage of HPC systems. HemeLB scales

up to thousands of nodes on leadership-class machines such as ARCHER or up to machine size on a Tier-2 machine like Cirrus, thus reducing the wall clock time for a simulation to tens of minutes. The goal is to offload the simulation execution part of PolNet to an HPC system such as Cirrus at EPCC or Lisa at SURFsara, and, if practical later, to a dynamically allocated HPC cluster on the Azure public cloud. The offload service will be accessed via a simple RESTful API in order to hide the details of accessing the HPC cluster and allow us to control access credentials to the systems.

4. Implementation – The HemeLB Offload Service – The Hoff

The service architecture implemented in Figure 2 to meet this aim focused on supporting a PolNet client workflow to run HemelB simulations but it is sufficiently general to support other applications and potentially other execution hosts. The service has successfully enabled PolNet users to exploit supercomputing resources at EPCC in the UK and SURFsara in The Netherlands.

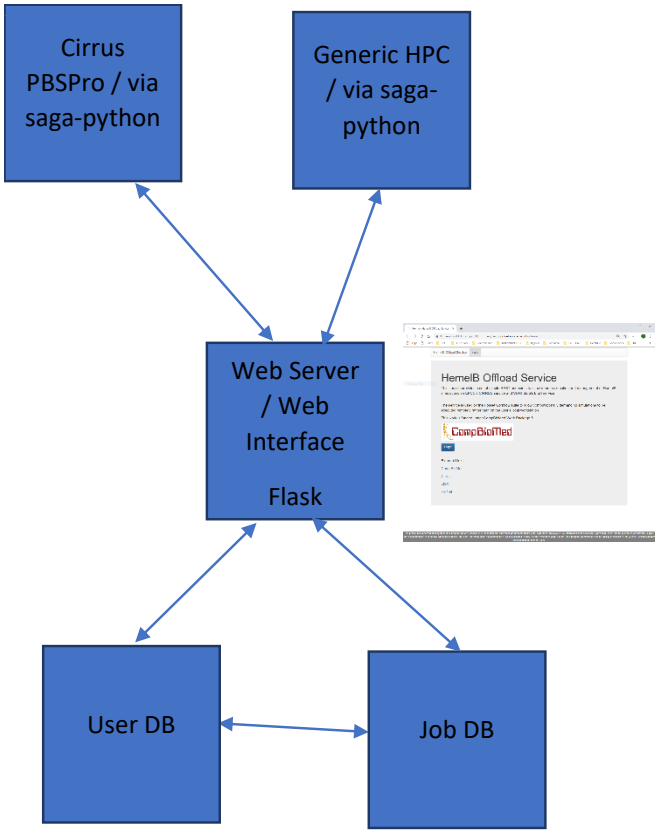

***Figure 2** Architecture*

The web service exposes the following RESTful endpoints to clients.

endpoint	method	Description
/jobs	GET	Lists all jobs known by the web service
/jobs	POST	Create a new job and return its identifier
/jobs/<job_id>/files	POST	Upload one or more input

		files for a job
/jobs/<job_id>/submit	POST	Submit a job for execution
/jobs/<job_id>/state	GET	Get the job's current state
/jobs/<job_id>/retrieved	GET	Returns a flag indicating if the job's output files have been retrieved from the execution host
/jobs/<job_id>/files	GET	Return a list of output files produced by the job
/jobs/<job_id>/filename	GET	Download a specific output file
/jobs/<job_id>	DELETE	Kill the job if in a running state, delete any resources held by the job and any files
/services	GET	Return a list of known execution hosts/services
/inputsets	POST	Create a set of related input files, return and id for the set
/inputsets/<id>	POST	Upload a file to an input set
/inputsets/<id>/hash	GET	Return a hash calculated from the inputset files
/templates	GET	Return a list of job template names

Jobs are specified by posting a job description as a JSON payload, following the definitions provided within the SAGA specification. A client creates a new job by posting a payload containing the job description. The job description contains details of the resource on which to execute the application and information such as reservation details and wallclock time. In response the web service creates a new job locally in the job database with status "NEW" and returns the job's identifier to the client. A staging area for the job is created on the web server. The client may then use the returned job identifier to post input files to the web service. Once file upload is complete, the client requests submission of the job by sending a POST to the job identifier. In response the web service returns a response of "SUBMITTED" and begins the workflow of creating an execution resource and executing the job. A service thread creates a working directory on the execution host, then stages over the input files and/or contents of an input set if specified. The job is then submitted on the remote host.

A separate service thread within the web service periodically monitors the remote job states of any jobs submitted via the web interface and updates the local job database accordingly. When a job has reached a "final" state, e.g. "Done", "Failed", output files are automatically retrieved from the execution resource to the webserver and any remote

resources released. Output files can subsequently be downloaded by the client from the web service via GET requests. The job and any output files / resources can be deleted by issuing a DELETE request using the web server job identifier. This will cancel the job if it is running, and free any resource and file space. The details of the job and its final state (DELETED) will remain in the web server job database. To enforce fair usage, a limit is applied to the number of active jobs which any user may have. Any job which is in a state other than 'DELETED' is considered as active. Deleted jobs are not counted towards a user's limit.

5. Impact

Providing scientific analysis and/or simulation services via online interfaces is commonplace in some fields and at an early stage more widely. These services come under a variety of names, such as "science-as-aservice", "science gateways", "workflows", or simply "our services". However these platforms are typically very heavyweight and require users to fully commit to their way of doing things. Here we are attempting a straightforward single use-case solution to solve real users' immediate problem. Longer term, however, there is no reason this API could not be accessed from within one of these systems should it be desired. Cloud computing has started to make inroads into the academic computing market in high-throughput application domains, such as genomics, particle physics analysis, and image processing. However as most public cloud instances do not include the high bandwidth, low latency networking needed for multi-node, tightly coupled simulations, "traditional" HPC workloads (CFD, molecular dynamics, weather/climate modelling, etc) have remained on institutional or national services. The Azure nodes equipped with InfiniBand offer a unique opportunity to enable these workloads, should we be able to obtain some credit for Microsoft's services. Returning to PolNet's particular domain, understanding the roles of flow in vascular development is important from a basic science point of view since the molecular regulation of angiogenesis remains incompletely understood. Furthermore, the translation of these results hold the key to the development of novel therapeutic approaches for vascular normalisation in the context of e.g. retinopathies or cancer.

6. References

[1] M. O. Bernabeu, M. L. Jones, R. W. Nash, A. Pezzarossa, P. V. Coveney, H. Gerhardt, and C. A. Franco, "PolNet: A Tool to Quantify Network-Level Cell Polarity and Blood Flow in Vascular Remodeling," Biophys. J., vol. 114, no. 9, pp. 2052–2058, May 2018.

[2] C. A. Franco, M. L. Jones, M. O. Bernabeu, I. Geudens, T. Mathivet, A. Rosa, F. M. Lopes, A. P. Lima, A. Ragab, R. T. Collins, L.-K. Phng, P. V. Coveney, and H. Gerhardt, "Dynamic Endothelial Cell Rearrangements Drive Developmental Vessel Regression," PLoS Biol, vol. 13, no. 4, p. e1002125, Apr. 2015.

[3] C. A. Franco, M. L. Jones, M. O. Bernabeu, A.-C. Vion, P. Barbacena, J. Fan, T. Mathivet, C. G. Fonseca, A. Ragab, T. P. Yamaguchi, P. V. Coveney, R. A. Lang, and H. Gerhardt, "Non-canonical Wnt signalling modulates the endothelial shear stress flow sensor in vascular remodelling," eLife, vol. 5, p. 1024, Feb. 2016.