Using HPC as a Service for Remote Parallel Processing on the Fiji Platform

Jan Kožusznik[1,2], Petr Bainar[2], Jana Klímová[2], Michal Krumnikl[1,2], Pavel Moravec[1,2], Václav Svatoň[2], and Pavel Tomančák[2,3]

[1] Department of Computer Science, FEECS VŠB – Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
[2] IT4Innovations, VŠB – Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
[3] Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstrasse 108, 01307 Dresden, Germany

{jan.kozusznik, petr.bainar, jana.klimova, michal.krumnikl, pavel.moravec, vaclav.svaton, pavel.tomancak}@vsb.cz

Abstract. Tackling current biomedical challenges calls for in-depth understanding of biological systems, particularly their structures, functions, and interactions on both the molecular and the cellular level. Biological imaging constitutes an important field of scientific investigation and one of its most valuable techniques is fluorescence microscopy. State-of-the-art imaging devices, such as light sheet microscopes, produce data sets so large that they can only be effectively analyzed by employing methods of image processing on high-performance computing (HPC) clusters. To address this issue, an HPC plugin for Fiji, one of the most popular open-source software tools for image processing, has been developed. The plugin enables end users to make use of HPC clusters to analyze large scale image data remotely and via the standard Fiji user interface. Seamless interaction between the remote HPC infrastructure and the user is substantially facilitated by the HPC as a Service (HaaS) middleware. To demonstrate the performance of the plugin it has been benchmarked on a Snakemake pipeline, performing complex registration and fusion tasks on sizable Selective Plane Illumination Microscopy (SPIM) time-lapse in toto recordings of developing embryos. The presented plugin offers a graphical user interface which allows the user to smoothly define task parameters, start execution, monitor progress, download results, and debug errors of the SPIM image processing pipeline. The presented framework will form a foundation for parallel deployment of any Fiji/ImageJ2 command on a remote HPC resource, greatly facilitating big data analysis.

Keywords: Selective Plain Illumination Microscopy (SPIM); Image Analysis; Big Data; High Performance Computing (HPC); ImageJ2; Fiji; HPC as a Service; Snakemake

1 Introduction

Modern microscopes generate vast amounts of image data that often have to be computationally processed before meaningful biological insights can be gained. The main culprit responsible for the data explosion is light sheet fluorescence microscopy [**?**,**?**,**?**]. One of its techniques, Selective Plane Illumination Microscopy (SPIM) [2] is particularly demanding in terms of data pre-processing, and yet SPIM has become very popular in biological studies since it allows imaging of cellular and developmental processes over long periods of time at high spatial and temporal resolution and across relatively large samples [1]. Such experiments can produce terabytes of multidimensional image data, and researchers in biology are often poorly equipped to deal with such a massive influx of imagery [**?**].

Fiji ("Fiji Is Just ImageJ"), a popular distribution of the open source platform for biological image analysis ImageJ, has emerged as a tool of choice for SPIM data processing (so called SPIMage processing) [9,**?**]. Plugins for multi-view reconstruction, time-lapse registration, fusion, deconvolution, and visualization are integrated into a comprehensive software solution under the familiar graphical user interface (GUI) of ImageJ [4,5,6,7,8]. Nevertheless, the sheer size of SPIM data makes deployment on a single computer impractical. To address this problem, Schmied et al. introduced a Snakemake based pipeline for parallel processing of SPIM data on a high-performance cluster (HPC) [10]. This approach however requires considerable expertise in command line operation as well as direct login access to an HPC cluster, two pre-requisites which may be unavailable to a typical researcher in life sciences.

In the ideal scenario, any remote HPC resource should be accessible directly through a Fiji GUI, for user-friendly big data SPIMage processing. An early attempt to achieve this was the Archipelago project, which targeted particularly ad hoc clusters of idle computers on a shared network in an academic institution. However, the set-up of Archipelago required profound programming experience and its development became inactive several years ago [REF footnote Github]. HPC as a Service (HPCaaS) constitutes a convenient solution to the problem of lack of access to suitable HPC hardware. HPCaaS enables users to access an HPC infrastructure without the need to purchase and manage physical servers, therefore effectively lowering the entry barrier to massively parallel processing.

In this paper we introduce a Fiji plugin which simplifies the tasks of remote deployment and parallel execution of jobs by incorporating the HaaS middleware and enhancing it with a graphical user interface. Consequently, the user can manage remote task execution, monitor progress, and debug errors without any interaction via the command line. We use a complex, multi-step workflow for large multi-view SPIM data sets as an application example for the proposed Fiji parallel processing framework.

2 Methods

2.1 HPC as a Service

To provide this simple and intuitive access to the supercomputing infrastructure an application framework called HPC as a Service Middleware (from now on referred to as HaaS middleware) has been developed at the IT4Innovation supercomputing center in Ostrava Czech Republic [1]. As the name suggests, the implementation is utilizing a

[1]http://www.it4i.cz/?lang=en

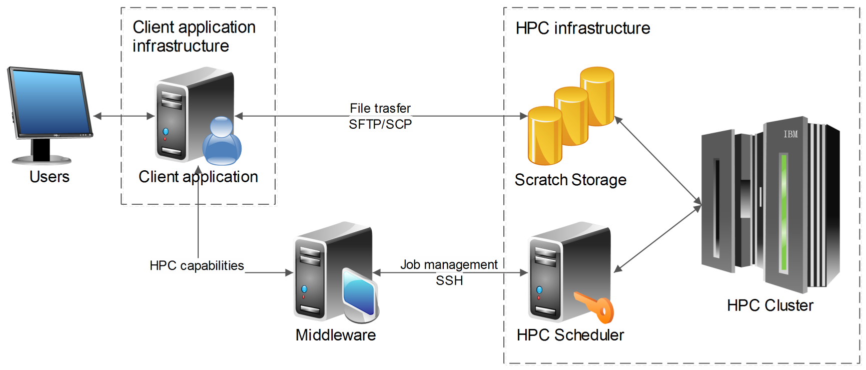


Fig. 1. Accessing the HPC environment via HPC as a Service Middleware

mid-layer that manages and provides information about submitted and running jobs and their data between the client application and the HPC infrastructure (Figure 1). HaaS middleware is able to submit required computation to HPC infrastructure, monitor the progress and notify the user should the need arise. It provides necessary functions for job management, monitoring and reporting, user authentication and authorization, file transfer, encryption, and various notification mechanisms.

HaaS middleware is a universally designed software architecture that enables unified access to different HPC systems through a simple object-oriented client-server interface using standard web services. In this way, it provides HPC capabilities to the users without the need to manage the running jobs from the command-line interface of the HPC scheduler on the cluster. It simplifies the access to the computation resources from the security and administrative point of view. Users are no longer authenticated via their IT4I's cluster credentials because they do not need direct cluster access but are authenticated via IT4I's HaaS middleware. HaaS middleware also provides the mapping between the external users and internal cluster service accounts that are being used for the actual job submission to the cluster.

Therefore, for the security purposes HaaS middleware enables the users to run only pre-prepared set of so-called Command Templates. Each template defines arbitrary script or executable file that will be executed on the cluster, any dependencies or third-party software it might require and the type of queue that should be used for the processing. The Command Template also contains the set of input parameters that will be passed to the executable script during run-time. Thus, the users are only able to execute pre-prepared command templates with the pre-defined set of input parameters, however the actual value of each parameter can be changed by the user for each job submission.

HaaS middleware has already been successfully used in several projects; for example, providing What-If analysis in a crisis decision support system Floreon+ [11], satellite image data analysis via ESA's Urban TEP portal [1] or in the area of molecu-

[1]https://urban-tep.eo.esa.int

lar diagnostics and personalized medicine. At IT4Innovations national supercomputing center, the usual workflow with HaaS middleware consists of a one-time Command Template preparation through cooperation of HPC center specialist and the end-user. After that the computations can be routinely executed by the end-user.

2.2 SPIM image processing pipeline

SPIM typically images living biological samples from multiple angles (views) collecting several 3D image stacks to cover the entire biological specimen. The 3D image stacks, representing one time point in a long-term time-lapse acquisition, need to be registered to each other which is typically achieved using fluorescent beads as fiduciary markers [6]. After the registration, the individual views within one time-point need to be combined into a single output image in a process referred to as fusion. Fiji offers two multi-view fusion strategies: content-based fusion [**?**] and multi-view deconvolution [5], the latter of which is a computationally demanding iterative process frequently requiring GPU acceleration.

In the context of long-term time-lapse imaging, the living specimen can move during acquisition, necessitating an intermediate step of time-lapse registration achieved usually with the same fiduciary beads as in multi-view registration [6]. Whereas parallel processing of individual time points has proven to be beneficial, the time-lapse registration takes only a few seconds and can therefore be performed on a single computing node without a need for parallelization.

The sheer size of the SPIM data requires conversion from raw microscopy data to Hierarchical Data Format (HDF5) for efficient input/output access and visualization through the BigDataViewer (BDV) Fiji plugin [4]. BDV uses an HDF5 container accompanied with an XML file containing experiment metadata (i.e. number of angles, time points, channels etc.). Although the conversion to HDF5 is a parallelizable procedure, further updating the XML file downstream the pipeline is not and per-timepoint XML files have to be created and then merged after completion of the registration and fusion steps. Consequently, the parallel processing of individual time points on a HPC resource (conversion to HDF5, registration, fusion and deconvolution) is interrupted by non-parallelizable steps (time-lapse registration and XML merging) performed typically on the cluster head node.

Schmied et al. introduced a Snakemake pipeline for parallel processing of individual time points [**?**,**?**]. Input parameters for the pipeline are defined by the user on an exemplary time-point of the SPIM acquisition using GUI-reliant Fiji SPIMage processing plugins. The parameters are then entered into a *config.yaml* configuration file. Snakemake workflow engine is capable of resolving dependencies between subsequent steps and thereby executing in parallel any tasks appearing to be independent. We use the complex SPIMage processing via Snakemake as a test case to demonstrate HaaS-mediated remote cluster operation directly from Fiji GUI.

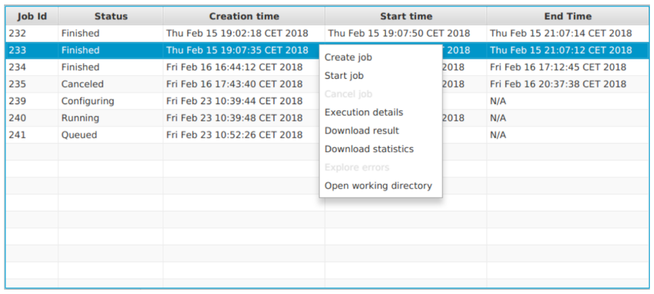
2.3 Interfacing Fiji SPIMage processing with HaaS

In order to provide the user with an intuitive interface for managing pipeline jobs[1], we created a Fiji plugin invokable from the application menu. This plugin communicates with the HPC as a Service (HaaS) middleware that submits a request for pipeline job execution to the cluster.

In the specific case of executing the SPIMage processing pipeline, the Snakemake engine is launched on the cluster and starts performing pipeline steps with parameters defined in the corresponding *config.yaml* file [10]. Consecutive steps identified by the engine as independent are executed in parallel as separate computational tasks using an instance of Fiji installed on the cluster.

Upon plugin (available from [3]) invocation, an initial window requiring HaaS user credentials[2] is launched. Apart from user name, password and email address, a local working directory needs to be defined. This directory contains a subdirectory for each created job and is named identically to the job ID. In addition, job subdirectories serve as staging areas for *config.yaml* configuration files specifying pipeline parameters. The user can modify the job parameters by editing the corresponding *config.yaml* in a common text editor and restart an interrupted, finished, or failed job.

Following a successful login, the main window providing overview about all jobs is displayed. The jobs are arranged in a table enabling the user to instantly check desired information (Figure 2).



Job Id	Status	Creation time	Start time	End Time
232	Finished	Thu Feb 15 19:02:18 CET 2018	Thu Feb 15 19:07:50 CET 2018	Thu Feb 15 21:07:14 CET 2018
233	Finished	Thu Feb 15 19:07:35 CET 2018	2018	Thu Feb 15 21:07:12 CET 2018
234	Finished	Fri Feb 16 16:44:12 CET 2018	2018	Fri Feb 16 17:12:45 CET 2018
235	Canceled	Fri Feb 16 17:43:40 CET 2018	2018	Fri Feb 16 20:37:38 CET 2018
239	Configuring	Fri Feb 23 10:39:44 CET 2018		N/A
240	Running	Fri Feb 23 10:39:48 CET 2018	2018	N/A
241	Queued	Fri Feb 23 10:52:26 CET 2018		N/A

Fig. 2. Main window

The user can create a new job by selecting *Create job* in the context menu, popping up following a right-click on a table row. Upon job creation, the plugin obtains a new job ID from the HaaS middleware and sets up a corresponding job subdirectory in the working directory.

Similarly, the user can invoke a selected job by choosing *Start job* in the context menu. Subsequently, the job is sent to the cluster via the HaaS middleware, which is responsible for the job life cycle from this point on. The main window providing overview of all jobs periodically retrieves job state from HaaS and updates the table.

[1]In this context, the term *job* is used for a single pipeline run with specified parameters

[2]Credentials required for authentication can be applied for by contacting paper authors

In addition, by selecting *Execution details* in the context menu, the user can display a detailed progress dashboard showing current states of all individual computational tasks for this job (Figure 3). The dashboard also includes another pane for the Snakemake output, being a powerful tool for debugging any computational errors which may occur.

Following a successfully finished pipeline job, the user may download result data (*Download result*) as well as a summary file containing key information on the performed job such as average/maximal duration of a particular task or memory usage statistics (*Download statistics*).

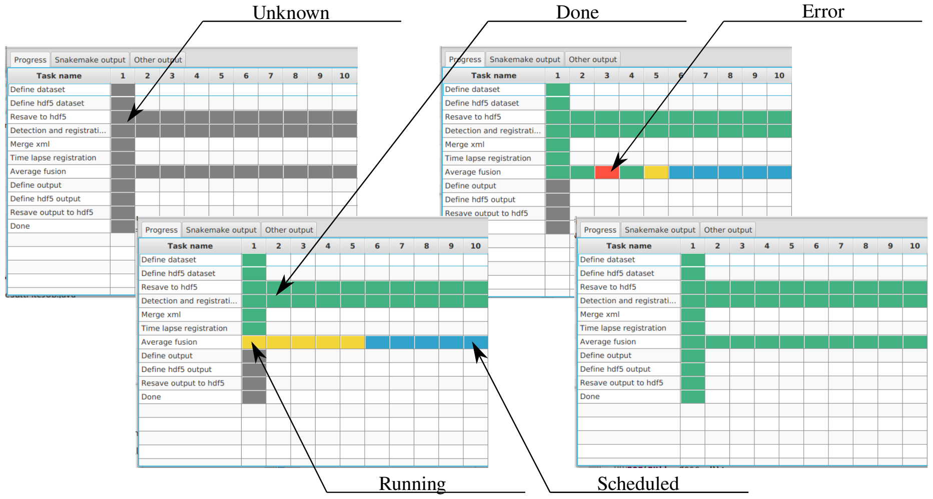


Fig. 3. Detailed dashboard for a selected job

2.4 Technical specification of used HPC resources

The execution of Snakemake pipeline from the Fiji plugin was tested on the Salomon supercomputer that consists of 1008 compute nodes, which in total provide 24 192 compute cores of x86-64 architecture and 129 TB RAM. 432 nodes are accelerated by two Intel Xeon Phi 7110P accelerators with 16 GB RAM, providing additional 52 704 cores and 15 TB RAM. The total theoretical peak performance reaches 2000 TFLOPS. Each node is equipped with 2x12 core Intel Haswell core processors and 128 GB RAM. The system is running Red Hat Linux.

The pipeline was tested on a dataset used in the experiments on the Madmax cluster at MPI-CBG [10]. Madmax cluster has 44 nodes with two Intel Xeon E5-2640, 2.5 GHz CPUs with 6 cores each (average CPU PassMark 9498). In comparison, Salomon nodes are equipped with two Intel Xeon E5-2680v3, 2.5 GHz CPU with 12 cores each (average CPU PassMark 18626). Salomon is running a newer generation of Xeon processors

(Haswell) providing two times higher performance than the Sandy Bridge architecture used on MadMax.

3 Results

In order to test the remote execution of the SPIMage processing pipeline through the developed Fiji plugin we uploaded a test dataset to the filesystem of the Salomon supercomputer at the National Supercomputing Center IT4Innovations in Ostrava, Czech Republic.

Table 1. Comparison of average calculation times – original and Salomon cluster

	Original cluster averages		Salomon cluster averages		
Summary	Memory [MB]	CPU time [s]	Memory [MB]	CPU time [s]	#jobs
Define dataset	2316	908	1448	256	1
Define hdf5 dataset	2158	39	219	23	1
Resave to hdf5	2827	530	2168	177	90
Detection and registration	7189	1388	5657	180	90
Merge xml	3	43	1	19	1
Time lapse registration	2534	953	1914	99	1
Average fusion	7761	3806	6765	280	90
Deconvolution GPU ǁ CPU	27171	7485	55040	2874	90
Define output	3	23	514	28	1
Define hdf5 output	2	32	885	29	1
Resave output to hdf5	4918	534	3922	89	90

This test dataset consists of 90 time-points of SPIM acquisition of a *Drosophila melanogaster* embryo expressing FlyFos fluorescent GFP fusion reporter for GENE X [**?**]. The embryo has been imaged with Lightsheet Z.1 SPIM microscope (Carl Zeiss Microscopy) from 5 views every 15 minutes from before cellularization until germband extension stage of embryogenesis. The raw data in the proprietary Zeiss *.czi* format comprised XXXGB.

In the first step the .czi raw data were resaved into the HDF5 container in parallel on the cluster. Next, the individual time-points were registered using fluorescent beads as fiduciary markers again in parallel on the cluster. Subsequently, a non-parallel job executed by Snakemake consolidated the registration XMLs into a single file followed by time-lapse registration using the beads segmented during the registration step. After this the pipeline diverged into either the parallel content based fusion or multi-view deconvolution. To achieve this divergence in practice, the Snakemake pipeline was launched from the Fiji plugin as two separate jobs using two different *config.yaml* files set to execute content based fusion and deconvolution respectively. In the final stage of the pipeline a new HDF5 container was defined and the fusion/deconvolution output was saved into it.

Table 2. Comparison of total calculation time per single node – original and Salomon cluster

	Original Cluster 90 TPs	Salomon cluster 90 TPs
Resave to hdf5	15	12
Detection and registration	15	7
Average fusion	47	6
Deconvolution GPU || CPU	740	57
Resave output	7	3
Total with average fusion	1h 31min	0h 35min
Total with deconvolution	13 h 9 min	2h 22min

The same dataset was used previously by Schmied et al. [10] to test an equivalent pipeline on a cluster at MPI-CBG in Dresden (for specifications see chapter 2.4). In order to compare the performance of the pipeline in the new HPC environment of the Salomon cluster in Ostrava we benchmarked the speed of the execution and memory consumption of individual steps of the pipeline, i.e. resaving, registration, fusion and deconvolution (Table 1) as well as the total duration of the processing of all 90 time-points (Table 2).

In addition, we examined the processed SPIM images in the HDF5 containers with the BigDataViewer (Figure **??**). The results show that entire pipeline was executed successfully and significantly faster on Salomon compared to Madmax cluster reflecting the difference in hardware performance.

Since the Snakemake pipeline on the Salomon cluster was launched through the Fiji plugin we conclude that the HaaS mediated remote execution is a viable option for processing of big SPIM datasets on a publicly accessible HPC resource.

4 Discussion

Herein described Fiji plugin facilitates the deployment of time-consuming task to HPC resources by not only providing a graphical user interface for configuring and running jobs on remote HPC machine, but also by significantly simplifying the official registration process required for running tasks on HPC infrastructure. The plugin design is general and supports execution of any task that can be specified as a batch job not limiting itself to a Fiji environment.

A workflow for parallel processing of large multiview SPIM dataset was used as an testing example of compute-intensive task to present the features of the newly developed plugin with HPC as a Service middleware. The experiments on the SPIM pipeline showed that the newer CPU architecture and larger number of cluster nodes led to significantly shorter running times when compared to the original study. Newly developed plugin simplifies the deployment of the computational intensive tasks on the HPC infrastructure and thanks to the graphic interface may attract those users who were discouraged from using the HPC infrastructure by the necessity of manual deployment of tasks via the command line and remote shell.

Future improvements of the newly developed plugin should aim at running more general tasks, not only a batch pipeline. Eventually, we aim to develop a parallel processing plugin focused not only on a task parallelism but on more complex code-level parallelism.

5 Acknowledgement

This work was supported by the European Regional Development Fund in the IT4Innovations national supercomputing center – path to exascale project, project number CZ.02.1.01/0.0/0.0/16_013/0001791 within the Operational Programme Research, Development and Education.

References

1. Amat, F., Höckendorf, B., Wan, Y., Lemon, W.C., McDole, K., Keller, P.J.: Efficient processing and analysis of large-scale light-sheet microscopy data. Nature protocols 10(11), 1679 (2015)
2. Huisken, J., Stainier, D.Y.: Selective plane illumination microscopy techniques in developmental biology. Development 136(12), 1963–1975 (Jun 2009), `http://www.ncbi.nlm.nih.gov/pubmed/19465594`
3. Kozusznik, J.: Multiview reconstruction with hpcaaa. `https://github.com/kozusznik/multiview-reconstruction-hpcaas` (2018), `https://doi.org/10.5281/zenodo.1185278`
4. Pietzsch, T., Saalfeld, S., Preibisch, S., Tomancak, P.: Bigdataviewer: visualization and processing for large image data sets. Nature methods 12(6), 481 (2015)
5. Preibisch, S., Amat, F., Stamataki, E., Sarov, M., Singer, R.H., Myers, E., Tomancak, P.: Efficient bayesian-based multiview deconvolution. nature methods 11(6), 645 (2014)
6. Preibisch, S., Saalfeld, S., Schindelin, J., Tomancak, P.: Software for bead-based registration of selective plane illumination microscopy data. Nature methods 7(6), 418 (2010)
7. Rueden, C.T., Eliceiri, K.W.: The imagej ecosystem: An open and extensible platform for biomedical image analysis. Microscopy and Microanalysis 23(S1), 226–227 (2017)
8. Rueden, C.T., Schindelin, J., Hiner, M.C., DeZonia, B.E., Walter, A.E., Arena, E.T., Eliceiri, K.W.: Imagej2: Imagej for the next generation of scientific image data. BMC Bioinformatics 18(1), 529 (Nov 2017), `https://doi.org/10.1186/s12859-017-1934-z`
9. Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., et al.: Fiji: an open-source platform for biological-image analysis. Nature methods 9(7), 676 (2012)
10. Schmied, C., Steinbach, P., Pietzsch, T., Preibisch, S., Tomancak, P.: An automated workflow for parallel processing of large multiview spim recordings. Bioinformatics 32(7), 1112–1114 (2016), `+http://dx.doi.org/10.1093/bioinformatics/btv706`
11. Svatoň, V., Podhoranyi, M., Vavřík, R., Veteška, P., Szturcová, D., Vojtek, D., Martinovič, J., Vondrák, V.: Floreon+: A web-based platform for flood prediction, hydrologic modelling and dynamic data analysis. In: Ivan, I., Horák, J., Inspektor, T. (eds.) Dynamics in GIscience. pp. 409–422. Springer International Publishing, Cham (2018)