Seamless HPC Integration of Data-Intensive KNIME Workflows via UNICORE

Richard Grunzke[1](✉), Florian Jug[2](✉), Bernd Schuller[3], René Jäkel[1], Gene Myers[2], and Wolfgang E. Nagel[1]

[1] Technische Universität Dresden, Dresden, Germany
richard.grunzke@tu-dresden.de
[2] Max Planck Institute for Cell Biology and Genetics, Dresden, Germany
jug@mpi-cbg.de
[3] Forschungszentrum Jülich, Jülich, Germany

Abstract. Biological research is increasingly dependent on analyzing vast amounts of microscopy datasets. Technologies such as Fiji/ImageJ2 and KNIME support knowledge extraction from biological data by providing a large set of configurable algorithms and an intuitive pipeline creation and execution interface. The increasing complexity of required analysis pipelines and the growing amounts of data to be processed nurture the desire to run existing pipelines on HPC (High Performance Computing) systems. Here, we propose a solution to this challenge by presenting a new HPC integration method for KNIME (Konstanz Information Miner) using the UNICORE middleware (Uniform Interface to Computing Resources) and its automated data processing feature. We designed the integration to be efficient in processing large data workloads on the server side. On the client side it is seamless and lightweight to only minimally increase the complexity for the users. We describe our novel approach and evaluate it using an image processing pipeline that could previously not be executed on an HPC system. The evaluation includes a performance study of the induced overhead of the submission process and of the integrated image processing pipeline based on a large amount of data. This demonstrates how our solution enables scientists to transparently benefit from vast HPC resources without the need to migrate existing algorithms and pipelines.

Keywords: KNIME · UNICORE · High Performance Computing · Integration

1 Introduction

BioImage Computing is central for many biological research projects [3,4,11], and many such projects share two common problems: (*i*) Data is large and analyzing it is time consuming, and (*ii*) both, data as well as the required data

R. Grunzke and F. Jug—These authors contributed equally.

analysis chain, changes in response to progress in ongoing research projects. Therefore, automated or semi-automated solutions are aiming at minimizing manual user intervention and overall runtime in order to solve these problems.

Dynamic research environments require short execution times of automated analysis pipelines and constant revisions of these pipelines to meet the projects needs. This poses a challenge for smaller research groups or institutes that simply cannot effort to employ dedicated HPC experts. It is desirable therefore to find solutions where the same person who implemented the initial analysis pipeline can also deploy it to available cluster hardware.

As delimitation to previous methods that enable the execution of KNIME workflows on HPC systems, the following methods are mentioned. One solution, the KNIME Cluster Execution module [13], is limited as it is proprietary and only enables access to cluster resources in conjunction with the Oracle Grid Engine. Another solution that is generic in scope converts KNIME workflows to gUSE workflows [5] to execute them on HPC resources. gUSE/WS-PGRADE [12] is a framework to build advanced HPC-enabled science gateways such as MoSGrid [14]. Due to the technological requirements of such science gateways and that KNIME is our target platform, this second alternative approach is also unsuitable in our use case. KNIME is also capable of executing jobs via Hadoop-like frameworks. Instead of our focus on arbitrary analysis workloads, this approach focuses on sub-workflows with problem structures that needs to fit the framework.

In this article we present a novel HPC access method that allows biological researchers with no HPC infrastructure knowledge to deploy and execute their analysis pipelines to a broad range of existing HPC systems. Users of the proposed setup are not required to learn any new language or HPC concept - they do, in fact, not even have to leave their local every day analysis environment (ImageJ2/KNIME). Regarding the technical realization we base our distribution method on UNICORE, which can be installed on any cluster or HPC resource. We have extended UNICORE's existing Data Oriented Processing [21] module with a parameter sweep feature to be capable of starting multiple parameterized jobs (see Sect. 2.3). This is utilized within the rule we designed that triggers the KNIME workflow execution on the HPC resources (see Sect. 2.3). For KNIME we developed all necessary modules for receiving and interpreting parameters for loading and fractionating the data (see Sect. 2.1). We provide various convenience functions that aim at making the entire process as intuitive and generic as possible. In Sect. 3 we present an evaluation of our approach including a thorough performance analysis.

2 Utilized Methods and Implementation

In this section we first describe the BioImage tools Fiji/ImageJ2 and the KNIME workflow application. Then, the UNICORE HPC middleware is described that greatly facilitates the utilization of HPC systems. We conclude with a description of how we integrated KNIME with HPC resources via UNICORE. This integration enables applying scientists to easily and efficiently utilize vast HPC resources from within their accustomed KNIME workflow application.

2.1 Fiji/ImageJ2 and KNIME

Image analysis tools in biological research are diverse and heterogeneous [3]. Although our proposed method is applicable for most if not for all available tools, here we constrain us to examples using Fiji/ImageJ2 [17] and KNIME (Konstanz Information Miner) [2]. While Fiji/ImageJ2 is the de-facto standard for working with microscopic datasets in biological research [17,18,20], using KNIME for image analysis applications is a more recent trend we believe to become increasingly popular in the near future.

Fiji, by itself, is a distribution of plugins and features for ImageJ [20] and ImageJ2. KNIME, initially a dedicated data mining and workflow management system [2], also developed into a potent image processing tool [3]. The developers of both systems, Fiji/ImageJ2 and KNIME, use a common image processing and analysis library for storing and operating on image data, the ImgLib2 [15]. This enables developers to write plugins that can be used in Fiji as well as in KNIME without additional work or code. Today, a plethora of image analysis methods via plugins are available.

The workflow we use throughout this paper is a data-preprocessing pipeline. It loads microscope images sequences acquired using a microfluidic device called ‘Mother Machine’ [9,22]. Each image sequence contains multiple experimental setups and the main task of the preprocessing is to enhance image quality and automatically find all regions of interest for extracting each individual experiment for further processing [10].

Parallelization of the overall computation is performed along two axes. Our proposed system submits multiple jobs and assigns equal amounts of data to be processed to each such job. Each image sequence contains multiple experimental setups and the main task of the preprocessing is to enhance image quality and automatically find all regions of interest for extracting each individual experiment for further processing. On top of that, each job can make use of multiple concurrently running threads to further split the data into smaller junks. The latter is implemented directly from within the KNIME workflow by using the ‘Parallel Chunk Loop’ construction that is freely available in the ‘Virtual Nodes’ package on the KNIME update site. While a regular loop executes its body for each dataset given, a ‘Parallel Chunk Loop’ performs for-loop-unrolling using a thread pool of configurable size.

Like many tasks in BioImage Computing, parallelization can be achieved by independently operating on individual images of a given dataset. A system that allows users, without previous exposure to cluster computing, to easily design such a processing pipeline in Fiji/ImageJ2 or KNIME was never done before. Our proposed system achieves this via the HPC middleware UNICORE. This constitutes an important step in unleashing the huge potential of HPC in various use cases in BioImaging and beyond with many potential users.

2.2 HPC Middleware UNICORE

UNICORE (Uniform Interface to Computing Resources) [1] is a middleware for building federated computing solutions. It focuses on providing seamless and secure access to heterogeneous resources such as high-performance computers and compute clusters, remote data storage and file systems. UNICORE is deployed and used in a variety of settings, from small projects to large, multi-site infrastructures involving HPC resources of the highest category. The latter category includes the European PRACE research infrastructure [16], the US XSEDE research infrastructure [23] and the EU flagship Human Brain Project [7].

UNICORE comprises the full middleware software stack from clients to various server components for accessing compute or data resources. Its basic principles are abstraction of resource-specific details, openness, interoperability, operating system independence, security, and autonomy of resource providers. In addition, it is easy to install, configure, administrate and available under a free and open source BSD license.

UNICORE offers services such as job submission and job management, data access and file transfer, metadata management and workflows. It abstracts the details of job submission, batch system commands, heterogeneous cluster properties and much more, allowing for a much simpler user interaction with HPC and data resources. Users typically use one of the UNICORE or custom clients to interact with the system to create and submit jobs. For the present use case, a different, data-driven way of interaction with UNICORE is employed and extended, which is described in detail in the next section.

2.3 Workflow Integration on a Cluster

We performed the integration of KNIME along the workflow that is introduced in Sect. 2.1. We encapsulate the workflow logic for preprocessing the 'Mother Machine' procedure in a meta-Node called 'Workflow Logic'. We have then loaded another Meta-Node called 'Data Setup' (see Fig. 1). This node serves as a generic data loading and filtering module. The only input required from

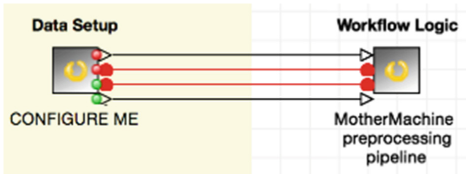

Fig. 1. Submission of a KNIME workflow. After the pipeline is assembled by the user he might choose to hide it in a so called Meta-Node, here called 'Workflow Logic'. After adding the predefined node 'Data Setup', the workflow is cluster ready in two simple steps: (1) the location of the data must be configured, and (2) the workflow must be exported to the submission directory via the standard KNIME export graphical dialog.

the user is the location of the data to be processed. In order to start the workflow on the cluster at hand it suffices to export it to the designated submission folder. This folder is exported from the cluster filesystem and mounted locally on the workstation of the user. The authentication of the user is done transparently by re-using the authentication already done during the local mounting of the network folder. Subsequently, UNICORE executes the user workflows on the cluster under the login of the specific user who submitted the workflow. Below we describe in detail how such a folder is to be configured.

Submitting and running Fiji/ImageJ2 workflows is achieved by very similar means. As mentioned in Sect. 2.1, every ImgeJ2 plugin is also a KNIME node. Therefore one can assemble a sequence of Fiji plugin calls from within KNIME and use the same procedure as described above. This possibility does, of course, not exist for all available tools and functions. Still, as long as the tool of choice is capable of storing a full processing pipeline for later execution our proposed UNICORE method applies.

To enable such easy job submissions and interact with the HPC cluster, the UNICORE feature called "Data Oriented Processing" [21] was used and extended for efficient parameter sweeps to enable the execution of multiple jobs per rule evaluation instead of just one. Here, the UNICORE server is set up to periodically scan a user's specific directory according to a set of user-defined rules. When new files are detected, the UNICORE server executes any matching rules, which lead to the automatic creation and submission of the HPC jobs. The rules are stored in a file in the submission folder. Here, we allow for two possible scenarios. The users themselves provide a rule for the execution of a given workflow. Otherwise, a system administrator provides a rule using specifications how to execute the workflow and how data handling and job creation needs to be performed by UNICORE.

During configuration of the rule, two opposing option have to be balanced. The first is to highly optimize the rule to get the best performance for a specific workflow. Such an optimization might to negatively impact the performance of other workflows and also decreases the usability as one optimized rule for every workflow has to be provided. The second option is find one or a few reasonable rule configurations that fit many workflows at once. This way the complexity for the user is minimized while a large performance increase by using HPC resources can be expected as compared to local workflow executions.

The code Listing 1.1 shows the unique part of such a UNICORE rule that is stored in a file named *.UNICORE_Rules*. The action within the rule is executed when a workflow in compressed form recognized by UNICORE. In our case the action defines a KNIME job with the workflow as import and parameters to steer the execution. The parameter k (line 14) defines that parameter sweep. In this example the job is created ten times (k from 0 to 9). The input dataset (parameter l in line 13) is preprocessed by ten KNIME instances processing independently one of the given chunks. Parameter n (line 15) declares how many instances are executed, whereas w (line 16) denotes the location of the imported input workflow from within the specific UNICORE job. Parameter *tr* (line 18) is the path and name of the workflow that triggered the action. Within the *Resources* (lines 19–22) section the specific job requirements are defined. According to these, UNICORE automatically requests fitting resources from the batch system. For the measurements in Sect. 3.4 up to 100 of such actions (lines 6–23) are activated to trigger KNIME instances.

Listing 1.1. The central section of the UNICORE rule is shown which, among other things, governs how an action is defined and triggered.

```
{                                                                  1
DirectoryScan: { IncludeDirs: ["."], "Interval": "30", },          2
Rules: [ {                                                         3
Name: BioHPCMeasurements_2880_1,                                   4
Match: ".*.zip",                                                   5
Action: { Type: BATCH, Job: {                                      6
  Name: knime_headless,                                            7
  Imports: [{ From: "file://${UC_FILE_PATH}",                      8
              To: workflow.zip },],                                9
  ApplicationName: knime,                                         10
  ApplicationVersion: 2.11.3_headless,                            11
  Parameters: {                                                   12
   l: "/lustre/ssd/grunzke/2880_1",                               13
   k: { From: 0, To: 9, Step: 1 },                                14
   n: "10",                                                       15
   w: "../workflow.zip",                                          16
   t: "8",                                                        17
   tr: "${UC_FILE_PATH}",},                                       18
  Resources: {                                                    19
   Memory: 20664M, CPUs: 8,                                       20
   Runtime: 1h, Queue: haswell,                                   21
   CPUsPerNode: 8,}},},                                           22
}, ... ],}                                                        23
```

As mentioned, the workflow is exported graphically from within KNIME to a given network folder from where it is picked up by the UNICORE submission system. All HPC resources that are attached via UNICORE are available. Depending on the rule, either workflows are submitted to a specific HPC system or UNICORE submits to an arbitrary HPC system that fits the workflow requirements configured in the rule file. In the code Listing 1.1, no specific system is configured otherwise the "Site" option would be defined. All this is transparent to KNIME. Following a specific rule defined in the submission folder, a large number of computing jobs can be seamlessly distributed over the cluster utilizing the batch system transparently.

3 Results

In the following we present the results of our research. First, we discuss how our approach is integrated from a user- and cluster-centric perspective. Second, the number of cores that can be efficiently utilized is identified. Then, the overhead induced be the UNICORE middleware is evaluated. Finally, the number of concurrently processed data is scaled up and evaluated.

3.1 Seamlessness Cluster Integration

A major motivation of our research is to enable the use of large computing infrastructures from within the workflow application KNIME. In the following, we discuss how we have achieved this from two different perspectives.

From a **user-centric perspective**, we achieved mainly two goals: (*i*) setting up a KNIME workflow submission system, which can be directly used from within KNIME, and (*ii*) the user does not require deeper knowledge of the HPC system at hand. Users are not obliged to use a specific job scheduling system or have to adopt to a different one used with the HPC infrastructure at hand. Support for various scheduling systems and arbitrary HPC infrastructure is an integral part of UNICORE. When a KNIME user finishes the design of a BioImage pipeline, the workflow can be run on a single workstation, but can now also easily be extended for cluster execution via our generic data handling node (see Fig. 1).

From a **cluster-centric perspective**, the UNICORE middleware monitors a directory for new submissions. The specified rule defines that when a valid KNIME workflows arrives, the submission and execution is automatically triggered. The rule defines the computing task that executes the KNIME workflow on the HPC resources. KNIME is required to be installed on the HPC system and configured in order to be executed in headless mode without graphical user interface. For a specific workflow representation the rule has to be written and tested just once. This enables users to continuously submit workflows for different input data sets to HPC systems and from within their working environment.

3.2 Evaluation of Job Scaling

In order to determine good parameters for the creation of individual jobs on the cluster the number of worker threads for the KNIME workflow is varied. The user can specify the location of the input images to be reconstructed and the number of threads to be used by the KNIME workflow engine.

As discussed in the previous Sect. 2.1, the user can graphically adjust the number of worker threads and therefore the degree of parallelization of individual jobs. The measurements for the runtime estimation use the dataset (2,880 input images and 70,000 output images with a total of 17 GB), a varying the number of worker threads and the overall job runtime was recorded. The HPC nodes used for this estimation are equipped with two Intel Xeon CPUs (E5-2680) with 12 cores each and a local SSD-based filesystem was used during measurements.

From the total run time, the mean processing time per image was calculated and is shown in Fig. 2 on the left side. As can be seen in the figure, the workflow can make use of additional worker threads in the reconstruction and can be reduced to slightly over one second per image (within statistical errors). Starting the given workflow with a larger number of worker threads does not further decrease the mean reconstruction time per image and a value of 8 worker threads seems to be a good trade-off value for minimal runtime per job and efficiency.

Figure 2 on the right side shows the speedup based on the runtime behavior shown on the left side. Also shown is the theoretical linear speedup with respect to the runtime (red line). In general the workload on the system induced by the workflow is rather I/O intense. Therefore, some deviation from the theoretical linear behavior is to be expected. To lower this impact we used local SSD-based storage at the compute node to minimize the I/O effect. As discussed, the increase in the number of threads for computation does not improve the runtime behaviour for values larger than 8 threads, and for the next measurement series we chose this value as default for this particular workflow.

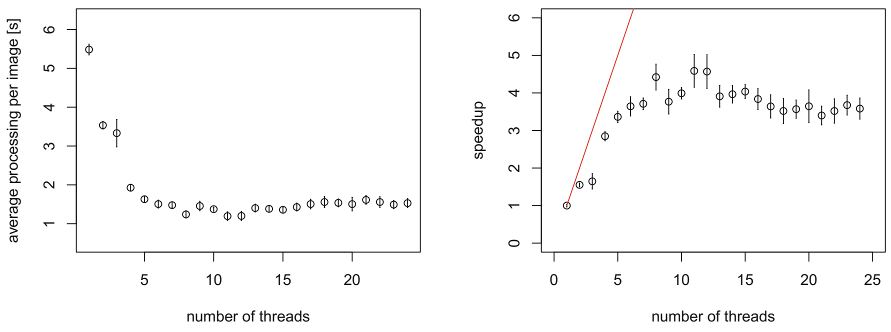

Fig. 2. Evaluation of the number of individual number of threads. The left plot displays the mean processing time per image for a given subset of image data with varying numbers of threads for the Fiji/ImageJ2 KNIME workflow including error bars. The right plot shows the calculated speedup including error bars of the Fiji/ImageJ2 KNIME workflow with a varying number of worker threads per job. The red line displays the theoretical linear speedup. (Color figure online)

3.3 Middleware Induced Overhead

The utilization of UNICORE with its triggering feature induces an overhead in the overall process. We estimate this overhead by measuring the time from which a workflow submission to the corresponding job submission time to the scheduler. One part of the overhead is the time needed for the middleware to register the new workflow in the directory. To reduce unnecessary communication overhead, a time interval for update checks is set to 30 s, which defines the maximum period for the middleware to recognize an new workflow. The second part of the overhead is the time period required by the middleware to execute methods to create compute jobs based on the workflow rule and to send those jobs to the scheduler. The first time measurement was triggered by exporting the workflow to the submission folder. Via the *scontrol* command the relevant submission time was obtained from the SLURM scheduler. The additional middleware overhead was therefore assessed by the difference between these two times. To estimate the variation in the process, the measurement was repeated ten times and in average a time of 27.2 s is the mean additional overhead induced by the UNICORE middleware.

3.4 Runtimes for Increasingly Large Datasets

This measurement series determines how our approach along the previously introduced BioImaging pipeline behaves at a large scale. The number of input and output files is scaled up to 7.488 million files with a total data emergence of up to 1.76 TB. The following table lists the configuration of the individual measurements which were repeated five times each. The size is the combined input and output data size. The second column displays the number of input and output files that are handled by the KNIME workflows during a measurement, whereas the third column denotes the number of datasets that are processed. Column four contains the number of chunks by which each dataset is partitioned for processing. The last column signifies that 800 cores in total are utilized, meaning 8 cores for every of the 100 KNIME instances (Table 1).

Table 1. Four measurement configurations with varying sizes and number of files, datasets, currently processed chunks per dataset, and utilized cores.

Size	Files	Datasets	Chunks	Cores
0.17 TB	0.7488 M	10	10	800
0.35 TB	1.4976 M	20	5	800
0.88 TB	3.744 M	50	2	800
1.76 TB	7.488 M	100	1	800

In our evaluation the number of concurrent KNIME instances is limited to 100. Beyond that, the error rate starts to significantly increase as KNIME is currently not built to handle a high number of parallel instances due to synchronization issues and relying on shared temporary files. A number of issues were solved by switching off OSGI locking and increasing the synchronization interval in knimi.ini[1]. As determined in Sect. 3.2, 8 cores are utilized as this is a sound number for this workflow. In these measurements, HPC nodes (each with two Intel Xeon CPUs (E5-2680) with 12 cores each) were utilized. A local SSD-based filesystem was used for providing the input data and storing the resulting data.

Figure 3 shows that the measurements with up to 1.76 TB of data have a runtime of 1259, 1955, 4318, and 7154 s respectively while utilizing 800 cores in each setting. The error bars show a significant standard deviation. This is due to the fact that the KNIME instances are scheduled as jobs by the HPC

[1] "-Dosgi.locking = none", "-Djava.util.prefs.syncInterval = 2000000".

batchsystem at varying points in time due to varying utilization levels of the HPC system. The measurements show that our approach can either process one dataset in parallel with many individual chunks, or it can process many datasets in parallel. Processing the datasets of the largest measurements (1.76 TB) on a local workstation with 4 cores would take about 17 days of continuous processing. With our easy-to-use and HPC-enabled approach the processing time is significantly reduced to just 2 h.

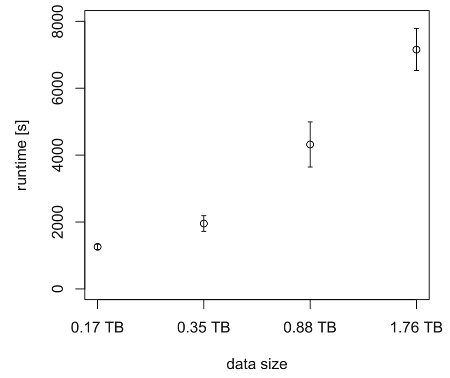

Fig. 3. Measurement results for concurrent processing of increasing number of datasets with an overall data size of up to 1.76 TB in 7.488 M files. The mean runtimes are 1259, 1955, 4318, and 7154 s respectively including error bars. These constitute a significant decrease in runtime compared to local processing on a workstation which would take about 17 days as compared to 2 h on the HPC system.

4 Conclusion

The presented work constitutes a novel approach to easily enable the use of large scale computing resources in the KNIME workflow application suite. Previously, KNIME users were very limited in using large computing infrastructures. Now, every major cluster scheduler is supported via the mature and widely used UNICORE HPC middleware. Enabling a high usability, users just need to graphically export a suitable KNIME workflow into a pre-configured directory and gets executed in parallel on the HPC cluster. To enable this, we extended the UNICORE middleware to support parameter sweeps via its data oriented processing feature, developed KNIME methods for interpreting parameters and fractionating input data, described how suitable workflows are created, and implemented it as a complete and generic approach along a concrete BioImaging use case. Performance measurements with up to 1.76 TB in 7.488 million files are presented that show highly favorable characteristics of our method. Trade-offs are required to either configure the method for high efficiency in a specific use case or for reasonable efficiency for multiple use cases. Alternatively, multiple directories can be configured for different use case scenarios offering more flexibility. Our approach can process either individual datasets in many parallel chunks, or many datasets in parallel, or a combination thereof. Now, large datasets in the range of terabytes can be processed in a matter of hours instead of weeks that were needed before.

5 Outlook

The preprocessing pipeline used throughout this work is a first example. We and our collaborators work on multiple other use-cases that perfectly fit our implemented system. As a rule of thumb, as soon as chunking the input in smaller pieces is a valid parallelization scheme, our method can be applied. We are planning to apply our method on even more ‘data-hungry’ tasks such as automated tracking pipelines for developing tissues (e.g. in *C. elegans*), or the segmentation, classification, and sorting of labeled neurons in the Drosophila fly brain. Other modes of operation are also feasible to enable novice users to use HPC resources. Currently, we work on fully automating an existing pre-processing pipeline [19] by utilizing the UNICORE data oriented processing approach and adding the advantage of enabling easy research data management via an integration with the KIT Data Manager [8] via the MASi project [6]. We also work on a method to offload individual KNIME workflow nodes to HPC resources.

Acknowledgment. The authors would like to thank the German DFG for the opportunity to do research in the MASi (NA711/9-1) project. Financial support by the German BMBF for the competence center for Big Data ScaDS Dresden/Leipzig and the 031A099 project is gratefully acknowledged. Furthermore, the research leading to these results has been supported by the LSDMA project of the German HGF and the Human Brain Project by the European Union.

References

1. Benedyczak, K., Schuller, B., Petrova, M., Rybicki, J., Grunzke, R.: UNICORE 7 - middleware services for distributed and federated computing. In: International Conference on High Performance Computing Simulation (HPCS) (2016, accepted)
2. Berthold, M.R., et al.: KNIME: the Konstanz information miner. In: Preisach, C., Burkhardt, H., Schmidt-Thieme, L., Decker, R. (eds.) Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization, pp. 319–326. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78246-9_38
3. Cardona, A., Tomancak, P.: Current challenges in open-source bioimage informatics. Nat. Methods **9**(7), 661–665 (2012)
4. Eliceiri, K.W., Berthold, M.R., Goldberg, I.G., Ibáñez, L., Manjunath, B.S., Martone, M.E., Murphy, R.F., Peng, H., Plant, A.L., Roysam, B., et al.: Biological imaging software tools. Nat. Methods **9**(7), 697–710 (2012)
5. de la Garza, L., Krüger, J., Schärfe, C., Röttig, M., Aiche, S., Reinert, K., Kohlbacher, O.: From the desktop to the grid: conversion of KNIME workflows to gUSE. In: Proceedings of the International Workshop on Scientific Gateways 2013 (IWSG) (2013)

6. Grunzke, R., Hartmann, V., Jejkal, T., Herres-Pawlis, S., Hoffmann, A., Deicke, A., Schrade, T., Stotzka, R., Nagel, W.E.: Towards a metadata-driven multi-community research data management service. In: 2016 8th International Workshop on Science Gateways (IWSG) (2016, accepted)
7. HBP: The Human Brain Project - High Performance Computing Platform (2015). https://www.humanbrainproject.eu/high-performance-computing-platform1
8. Jejkal, T., Vondrous, A., Kopmann, A., Stotzka, R., Hartmann, V.: KIT data manager: the repository architecture enabling cross-disciplinary research. In: Large-Scale Data Management and Analysis - Big Data in Science, 1st edn (2014). http://digbib.ubka.uni-karlsruhe.de/volltexte/1000043270
9. Jug, F., Pietzsch, T., Kainmüller, D., Funke, J., Kaiser, M., van Nimwegen, E., Rother, C., Myers, G.: Optimal joint segmentation and tracking of *Escherichia Coli* in the mother machine. In: Cardoso, M.J., Simpson, I., Arbel, T., Precup, D., Ribbens, A. (eds.) BAMBI 2014. LNCS, vol. 8677, pp. 25–36. Springer, Cham (2014). doi:10.1007/978-3-319-12289-2_3
10. Jug, F., Pietzsch, T., Kainmüller, D., Myers, G.: Tracking by assignment facilitates data curation. In: IMIC Workshop, MICCAI, vol. 3 (2014)
11. Jug, F., Pietzsch, T., Preibisch, S., Tomancak, P.: Bioimage informatics in the context of Drosophila research. Methods **68**(1), 60–73 (2014)
12. Kacsuk, P., et al.: WS-PGRADE/gUSE generic DCI gateway framework for a large variety of user communities. J. Grid Comput. **10**(4), 601–630 (2012)
13. KNIME: KNIME Cluster Execution (2016). https://www.knime.org/cluster-execution/
14. Krüger, J., Grunzke, R., Gesing, S., Breuers, S., Brinkmann, A., de la Garza, L., Kohlbacher, O., Kruse, M., Nagel, W.E., Packschies, L., Müller-Pfefferkorn, R., Schäfer, P., Schärfe, C., Steinke, T., Schlemmer, T., Warzecha, K.D., Zink, A., Herres-Pawlis, S.: The MoSGrid science gateway - a complete solution for molecular simulations. J. Chem. Theory Comput. **10**, 2232–2245 (2014)
15. Pietzsch, T., Preibisch, S., Tomančák, P., Saalfeld, S.: ImgLib2 - generic image processing in Java. Bioinformatics **28**(22), 3009–3011 (2012)
16. PRACE: PRACE Research Infrastructure (2015). http://www.prace-ri.eu/
17. 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. Nat. Methods **9**(7), 676–682 (2012)
18. Schindelin, J., Rueden, C.T., Hiner, M.C., Eliceiri, K.W.: The ImageJ ecosystem: an open platform for biomedical image analysis. Mol. Reprod. Dev. **82**(7–8), 518–529 (2015)
19. Schmied, C., Steinbach, P., Pietzsch, T., Preibisch, S., Tomancak, P.: An automated workflow for parallel processing of large multiview SPIM recordings. Bioinformatics **32**, 1112–1114 (2015)
20. Schneider, C.A., Rasband, W.S., Eliceiri, K.W.: NIH image to ImageJ: 25 years of image analysis. Nat. Methods **9**(7), 671–675 (2012)
21. Schuller, B., Grunzke, R., Giesler, A.: Data oriented processing in UNICORE. In: UNICORE Summit 2013 Proceedings, IAS Series, vol. 21, pp. 1–6 (2013)
22. Wang, P., Robert, L., Pelletier, J., Dang, W.L., Taddei, F., Wright, A., Jun, S.: Robust growth of Escherichia coli. Curr. Biol. **20**(12), 1099–1103 (2010)
23. XSEDE: Extreme Science and Engineering Discovery Environment (2015). https://www.xsede.org