Towards Performant Workflows, Monitoring and Measuring

(Invited Paper)

Jeanette Sperhac
Center for Computational Research
University at Buffalo
Buffalo, NY 14203
Email: jsperhac@buffalo.edu

Robert L. DeLeon
Center for Computational Research
University at Buffalo
Buffalo, NY 14203
Email: rldeleon@buffalo.edu

Joseph P. White
Center for Computational Research
University at Buffalo
Buffalo, NY 14203
Email: jpwhite4@buffalo.edu

Matthew Jones
Center for Computational Research
University at Buffalo
Buffalo, NY 14203
Email: jonesm@buffalo.edu

Andrew E. Bruno
Center for Computational Research
University at Buffalo
Buffalo, NY 14203
Email: aebruno2@buffalo.edu

Renette Jones-Ivey
Institute for Computational and Data Sciences
University at Buffalo
Buffalo, NY 14203
Email: renettej@buffalo.edu

Thomas R. Furlani
Roswell Park
Comprehensive Cancer Center
Buffalo, NY 14203
Email: Thomas.Furlani@roswellpark.org

Jonathan E. Bard
Genomics and Bioinformatics Core
University at Buffalo
Buffalo, NY 14203
Email: jbard@buffalo.edu

Vipin Chaudhary
Computer Science and Engineering
University at Buffalo
Buffalo, NY 14203
Email: vipin@buffalo.edu

***Abstract*—As part of the U.S. National Science Foundation (NSF) funded XD Metrics Service project, we are developing tools and techniques for the audit and analysis of High Performance Computing (HPC) and Cloud infrastructure. This includes a suite of tools for the analysis of HPC jobs, based on performance metrics collected from compute nodes. To date, we have developed two closely related utilities: XDMoD, which was designed to monitor usage and performance of NSF's innovative HPC resources (known as XSEDE), and Open XDMoD, which was designed to monitor usage and performance in academic, governmental or commercial cyberinfrastructures. Considerable effort has been made to continually improve XDMoD, in order to capture the most important aspects of modern research computing.**

One area in which XDMoD is lacking is in tracking workflows, which are broadly designated as containing the elements of data transfer/input and one to many computational steps. As data sets have become larger, data movement has become more time and resource intensive, and hence more important to characterize. In addition, multiple step workflows, in which one input spawns a complex series of processes, are becoming more common. Although XDMoD currently captures some of the information required to properly track complex workflows, there are clearly some key data that are missing. In this paper, we discuss the existing state of workflow monitoring, and suggest strategies to improve on the information captured.

***Index Terms*—Computer Performance, Data Processing**

I. Introduction

The XD Metrics on Demand (XDMoD) tool provides stakeholders with ready access to data about utilization, performance, and quality of service for High Performance Computing (HPC) and cloud resources [1], [2], [3], [4], [5]. This comprehensive tool was originally developed to support resources for the National Science Foundation (NSF) innovative HPC program (known as XSEDE), and has been open-sourced and made available to general application at universities, government laboratories, and commercial entities [6]. XDMoD enables users, managers, and operations staff to monitor, assess and maintain quality of service for their computational, storage, and networking resources. To do this, XDMoD harvests data from the various resources and displays the resulting job, usage, and accounting metrics over any desired timeframe, using the XDMoD web interface and its array of visual analysis and charting tools. See Figure 1 for a view of the XDMoD user interface.

Scientific workflows, here called simply *workflows*, execute a series of computational or data manipulation steps. Workflows may consist of automated software pipelines, or loosely associated tasks executed one after the other. Workflows typically entail the use of widely distributed computing resources ranging from storage to compute nodes, and may enable the processing and analysis of data from large-scale scientific experiments. Workflows can help to organize, streamline, and document the scientific process, and can be shared along with datasets, assisting with reproducibility.

XDMoD is instrumented to collect metrics on many aspects of high-performance computing, but it does not comprehensively collect and associate data on scientific workflows. One of the obstacles is that any individual workflow may perform many disparate tasks on a variety of resources. Desirable

978-1-7281-6607-0/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: University at Buffalo Libraries. Downloaded on March 20,2023 at 14:11:10 UTC from IEEE Xplore. Restrictions apply.

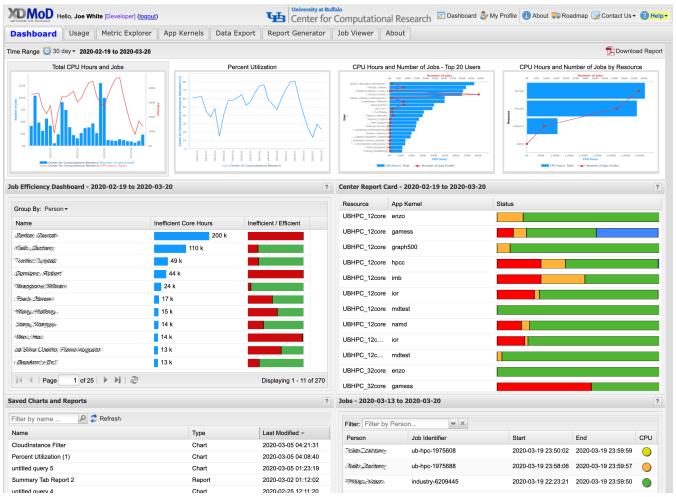

Fig. 1. Example screenshot of the XDMoD web interface showing usage, performance data and quality-of-service metrics for an academic HPC resource.

data on workflows should describe numerous facets of these computations, such as network usage incurred by transferring large data sets, network usage outside of HPC resources, and resource usage on non-HPC computational infrastructure, such as webservers and databases. Standardizing instrumentation across resources and connecting the dots between workflow components and tasks present a challenge to workflow monitoring. A goal is to instrument computing resources so that we can better understand network usage by even informal workflow jobs that involve substantial data transfer in addition to computation. However, substantial information about several aspects of workflow usage, such as network usage within the HPC stage of workflow jobs, can be gleaned from XDMoD now. With this paper we explore the monitoring already in place, as well as the components still needed, to achieve monitoring of workflows at this level.

II. Prior Work

A. System monitoring

A number of monitoring tools can be used to track HPC performance; a few open-source examples include Ganglia [7], Lightweight Distributed Metric Service (LDMS) [8], [9], [10], Performance Co-Pilot (PCP) [11], TACC_Stats [12], and XALT [13]. Others are discussed elsewhere [14]. Most of these tools track hardware and OS performance counters, enabling determination of computational nodes' performance. Indeed, several of these tools (PCP [11], TACC_Stats [12]) provide raw system-level performance data to the XDMoD processing pipeline. Network monitoring of data transfers, and large scale file transfer monitoring, has been addressed by Globus and others [15], [16], [17].

One of XDMoD's strengths lies in its modular design, which permits incorporation of new data sources, enables configuration for use on different systems, and provides views tailored to a range of user roles, from center directors to application developers. As architectures and practices evolve, XDMoD can be adapted to monitor them. The present effort to improve monitoring of workflows is just one example of efforts to adapt XDMoD to changing computational paradigms.

B. Workflows

Workflows are computational tasks that involve multiple distinct steps. For example, a simple workflow could involve copying data from a database to a shared storage area, running computational tasks on the data to produce some output, and then storing the output in a destination database. Each individual task may itself involve multiple stages. The computational task could involve, for instance, a serial pre-processing step, compute-intensive parallel processing, and finally, a serial post-processing task.

Web-based science gateways are an example of workflows. For the I-TASSER (Zhanglab) Gateway [18] workflow, the process begins when a user accesses the webserver and submits input data, which is validated and then copied over a network for further processing on a compute cluster. Next, the data are prepared and an HPC job is submitted to the XSEDE Comet resource. Once the HPC job is complete, the results are copied back to the gateway cluster, and final processing generates web page output. Finally, the user is sent an informational email, and can then access the webserver and download the results. Each step summarized here itself consists of multiple computational and data processing tasks.

Workflows can take on several different forms. They can be based on formal workflow utilities such as Pegasus [19] and Apache Taverna [20], or specifications such as the Common Workflow Language (CWL) [21] and Open Workflow Definition Language (OWDL) [22]. They can be produced by specific procedures encoded on gateways such as Zhanglab [18], as described above, and many others in XSEDE [23]. Probably most commonly, workflows can be informal, custom procedures to transfer data and then operate on it, which are developed to address some specific processing requirement.

High-level monitoring of workflows is critical to ensuring access to resources and planning for future acquisition, among other concerns. Workflow monitoring is provided by a number of utilities and gateway frameworks, including Apache Airavata [24], PIM [25], and others. For example, Apache Airavata XBaya, which supports various workflow runtimes, is capable of tracking workflow progress either synchronously or asynchronously; similarly, Pegasus and others offer extensive workflow monitoring capabilities.

Much workflow monitoring is understandably focused on development-time tracking, error detection and recovery, establishing correctness and performance, and verifying provenance [26], [27]. This focus is needed to produce useful, extensible workflows and advance scientific results. However, monitoring that takes a longer view will help to establish trends in the demand for workflows. Accessing and cataloging the metadata collected by workflow runtimes will be key to establishing these pipelines and ultimately providing these metrics. In this paper we pursue some higher-level aspects of workflow monitoring to enable a better understanding of

Authorized licensed use limited to: University at Buffalo Libraries. Downloaded on March 20,2023 at 14:11:10 UTC from IEEE Xplore. Restrictions apply.

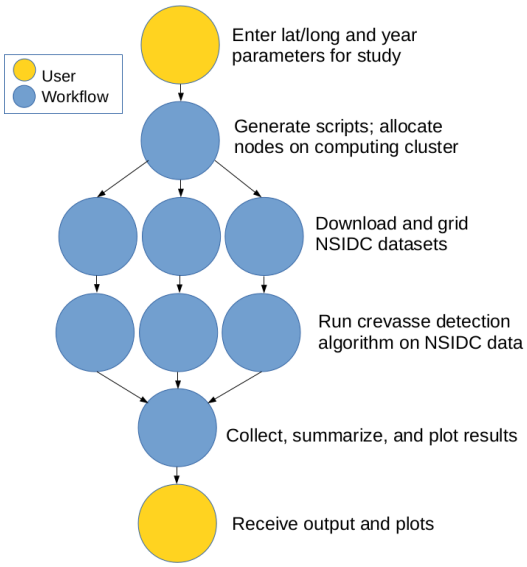

Fig. 2. GHub crevasse detection workflow schematic. Yellow=user tasks; blue=tasks automated in software, illustrates how the workflow handles data wrangling and HPC job submission. Previously, scientists performed all steps of the workflow manually.

workflow demand and the corresponding computational and infrastructure needs of these useful tools.

III. Workflow Case Studies

To illustrate the issues presented by data-intensive workflows, we present two case studies that show different types of issues from a monitoring perspective.

A. Crevasse detection workflow

An example of a workflow that encompasses data acquisition and processing followed by computation is a crevasse detection workflow that runs on the Greenland Hub Gateway [28] hosted at the University at Buffalo (UB) Center for Computational Research (CCR) [29]. In Southeast Greenland, cold air temperatures and high snowfall rates give rise to firn aquifers, vast stores of liquid water beneath the ice-sheet surface [30]. Though invisible from the surface, firn aquifers drain through crevasses to the glacier bed, influencing the flow of the ice above [31]. The crevasses, which are variable from one year to the next, are visible from the surface. Better information on crevasse location is needed to inform the glacier melting models prepared by ice-sheet scientists.

The crevasse detection workflow provides a direct way to infer the drainage patterns of the firn aquifers around Greenland in space and time. Figure 2 shows the workflow consisting of multiple, formerly user-intensive processes that are pipelined together.

A user submits the year range, crevasse detection tolerances, and the latitude/longitude bounds of interest, using a Jupyter Notebook hosted on the GHub gateway [32]. The Pegasus workflow then allocates compute nodes on the CCR academic HPC cluster, from which it accesses, processes, and grids the appropriate IceBridge altimetry datasets from the National Snow and Ice Data Center (NSIDC) [33]. It then runs a crevasse-identifying algorithm on the gridded data [34]. Finally, the workflow returns the results for the user to view and download. Results include plots and text files of crevasse size, shape, depth, and locations. Refer to Figure 3 for an example output plot from the workflow showing identified crevasse features.

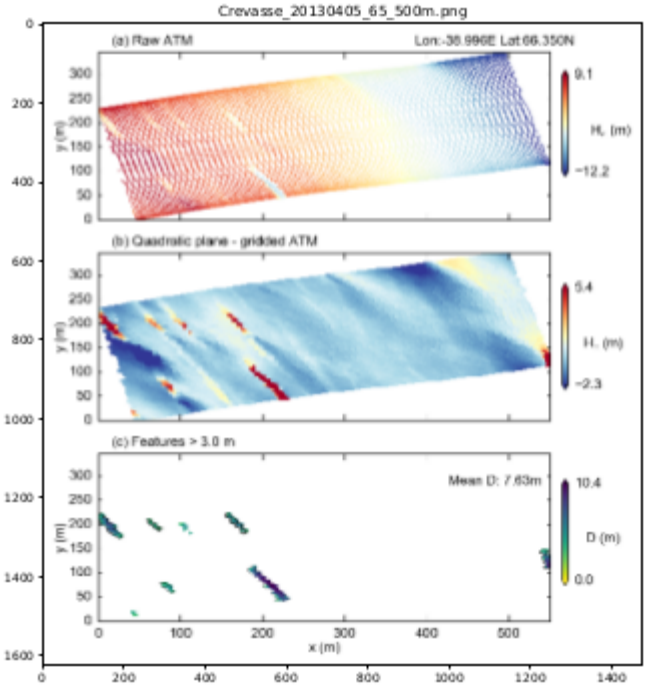

Fig. 3. GHub crevasse detection workflow output. Features depicted in the bottom plot signify crevasses identified according to user-supplied tolerances.

Typically, substantial user interaction would be required to complete each of the steps in this computation. To run this workflow, however, the user need only interact with a GUI running in a web browser. This workflow pushes both the management of the job and the data selection, transfer, and conversion into the background, allowing the scientist to focus on their analysis. The workflow can be readily shared along with datasets, peer reviewed papers, and conference presentations, for reproducible and shared research outcomes.

XDMoD can provide a time evolution view of the HPC portion of formal workflows of this sort. The two timeseries plots in Figure 4 illustrate a crevasse workflow job with its initial data transfer step that ran within the HPC job portion of the workflow. In the bottom plot, data staging can be seen as two spikes that occur in the data transfer rate; once the data have been programmatically selected on NSIDC's web portal [35] and downloaded to the HPC resource's scratch space, the CPU User % rises, as shown in the top plot, indicating that the staged data are being processed.

Figure 5 shows average CPU User % for selected workflow jobs. These jobs were submitted to UB CCR HPC resources by

Authorized licensed use limited to: University at Buffalo Libraries. Downloaded on March 20,2023 at 14:11:10 UTC from IEEE Xplore. Restrictions apply.

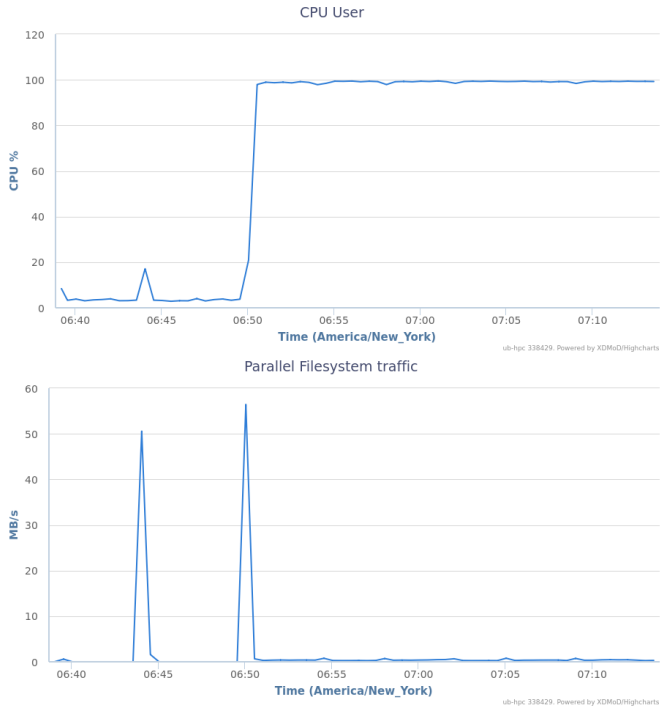

Fig. 4. XDMoD timeseries plots from an individual HPC job show the effect of workflow data staging. Top plot shows the CPU User % on the compute nodes; bottom plot shows data transfer rates between the compute nodes and the parallel filesystem (MiB/second). Note the peaks corresponding to fetching the dataset (bottom) precede the high CPU usage (top) which indicates actual processing.

two HUBzero-based gateways, VHub[1] and GHub,[2] from 2019-03-01 to 2020-03-31. Most are workflow jobs run via Pegasus. This figure illustrates the conundrum posed by these jobs: While we can fully characterize them on the system side, we have little to no insight into how the workflow was developed and deployed. By making performance profiles available to workflow developers and end users, real gains can be made in understanding overall workflow efficiency.

B. Informal genomics workflow

Recent technological advances in the biological sciences field of next-generation sequencing has led to a dramatic increase in data production and demand for computational power. Each data point produced by such sequencing platforms consists of strings of "A", "G", "C", and "T" characters, fifty to three hundred elements long, representing the sequenced DNA, along with associated data quality scores. Illumina-based sequencing platforms, such as the NovaSeq, are capable of producing 10 billion data points in the course of 48 hours. The task of routinely analyzing this large quantity of complex data requires scientific workflows.

Once sequenced on the Illumina platform, the data undergo primary and secondary analysis using CCR's compute clusters. Figure 6 shows the CPU User % and data transfer rate for

[1]https://vhub.org

[2]https://vhub.org/groups/ghub/

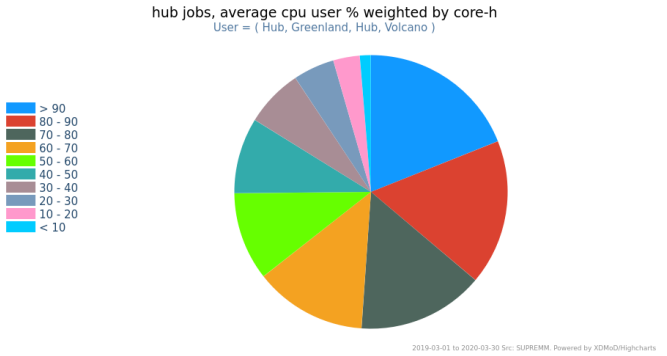

Fig. 5. Workflow jobs submitted from the GHub and VHub gateways, 2019-03-01 to 2020-03-31, are characterized by their CPU user time efficiency in this XDMoD plot (100% is full CPU usage by the user application). Here, fully 50% of the jobs show CPU User % of 70% or better (Reading clockwise from the top of the chart, medium blue, >90%; red, 80–90%, and dark grey, 70–80%).

a genomics workflow. This workflow uses the HISAT2 [36] alignment program to map next-generation sequencing reads. For RNA sequencing experiments, raw sequencing data is submitted for transcriptomic alignment to standard reference genomes using HISAT2. During alignment, each character string of DNA is compared against a reference genome containing the sequence for the appropriate organism. Each Illumina-based sequencing experiment produces data for multiple biological samples (typically between 6 and 48), each requiring alignment. Using a custom batch script, samples and their reference data are submitted to the HPC resource, each sample as a separate job. HISAT2 then processes the sequence data, and maps each data point to the reference genome. Once each data point has been associated to the correct position in the reference, the data set is sorted along a linear representation of the specified reference using a coordinate based system. This final step, in which each data point is coordinate-sorted and written out to the final output file, is typically I/O bound.

The runtime performance of the full genomics workflow is dictated by two primary factors, depth-of-sequencing (number of data points per sample), and the size of the reference genome. For instance, the human reference genome has more than three-billion nucleotides, whereas the *E. coli* bacterium has just over five million nucleotides. This dramatic difference in reference genome size directly impacts the time for alignment and final sorting. This HISAT2-based workflow is the most common sequencing pipeline used by the UB Genomics and Bioinformatics Core facility, with slight variations dependent on the biological questions being addressed.

Due to the size and number of input files, data staging is a critically important element of genomics workflows. The sequencing instruments used by UB's Genomics Core are directly connected to CCR's HPC storage systems, and typically produce several TiB per run. Despite the direct connection, data transfers from the sequencers to the HPC storage are not currently tracked, making it difficult to associate data transfers

Authorized licensed use limited to: University at Buffalo Libraries. Downloaded on March 20,2023 at 14:11:10 UTC from IEEE Xplore. Restrictions apply.

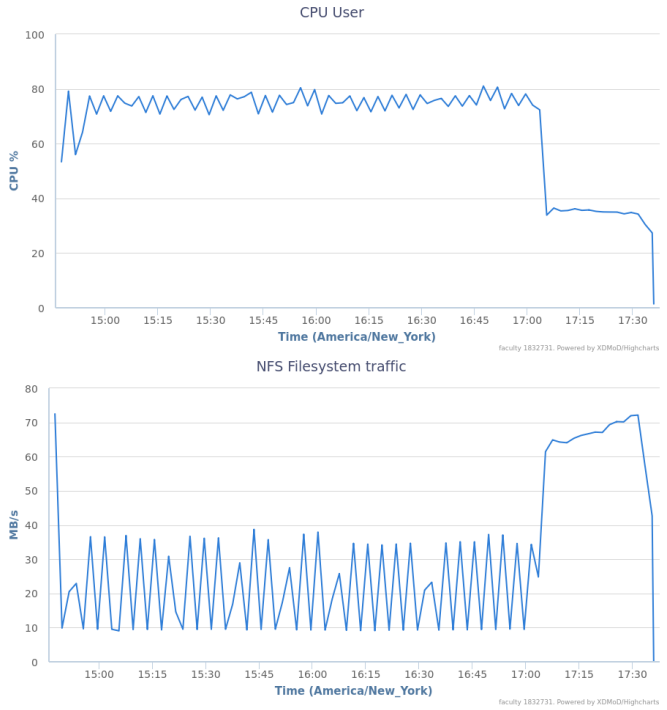

Fig. 6. XDMoD timeseries plots from an individual HPC job show a workflow pipeline in which the data was pre-staged. Top plot shows the CPU User % on the compute nodes; bottom plot shows data transfer rates between the compute nodes and the parallel filesystem (MiB/second). Note the initial CPU intensive stage (top), followed by post processing writing out the results (bottom).

with the corresponding downstream HISAT2 alignment jobs. However, even if the resources were instrumented to monitor the data transfer, linking the data staging to the actual informal workflow job would still be challenging.

As a result, the XDMoD timeseries plots illustrating the workflow fail to capture the initial data transfer step, as shown in Figure 6. As before, the top plot shows CPU User %, and the bottom shows data transfer rate, but the dataset is already staged, in contrast to that shown in Figure 4. The workflow itself is constrained by the underlying filesystem, resulting in the tight correlation seen between CPU and I/O load in Figure 6. However, this workflow is able to take good advantage of processing power by ensuring that the necessary data are close to the processors. The job illustrated here was also memory intensive, hence the filesystem performance was exposed due to the lack of I/O caching in memory.

In the case of the genomics workflow, it is less clear that any inefficiency is incurred by the data transfer time itself, given that data transfer happens asynchronously with the computing portion of the work. Certainly there is a time cost to the data transfer, however, in that the compute portion must wait until the initial transfer is complete.

In contrast to the Pegasus workflow described in Section III-A above, many data-intensive scientific workflows are ad-hoc, requiring user intervention for data transfer and processing, and complicating metric collection on these jobs. In order to determine data transfer information for these informal workflows, it may be necessary to secure cooperation or data sharing from the submitting user, in addition to instrumenting the transfer or compute nodes.

IV. Analysis

A. Computational signatures of workflows

Though there is great variation among workflows, both formal and informal, a common factor is the substantial use and transfer of datasets. Indeed, workflows exist because they can smoothly incorporate the handling, transfer, validation and conversion of data together with computation steps, simplifying the execution of these processes. The movement of large datasets for custom user workflows is not specifically tracked in XDMoD. To demonstrate that this hampers our understanding of the true resource use of workflows, we investigated ways that large data transfer workloads differ from routine HPC jobs. To do so, we compared the XSEDE jobs that most heavily used the Lustre parallel filesystem with those that showed average Lustre usage. The data are from the XDMoD database for XSEDE from 2019–2020; analysis includes approximately 15M jobs; the results are shown in Figure 7. Not surprisingly, for the large data transfer jobs, the CPU spends a higher fraction of its time in system mode. This time spent in system mode represents the system overhead on handling and transferring datasets of substantial size.

Unfortunately, the main effects of transferring large data sets in preparation for computation is the time spent on transfer nodes, which delays the time to science and exaggerates bottlenecks in the parallel filesystems. These drawbacks manifest in increased job wall time, which decreases resource efficiency and throughput. However, these effects are not well captured to date.

For example, most if not all large filesystems provide monitoring tools that can be used to track usage due to various activities. Figure 8 shows one such annual plot for a local HPC resource [29] running IBM's Spectrum Scale (formerly known as GPFS). While the overall traffic shown can be traced to individual users, the filesystem reporting provides no insight into per-job data movement. One can subtract the job contribution (measured by XDMoD and its supporting toolchains) from the filesystem view, but that is still not sufficient to identify the actual workflow contributions (some contributions can instead be due to completely unrelated lower priority tasks such as remote backup, synchronization, etc.).

B. Workflows and gateways

In recent years, there has been a substantial gain in science gateways usage as a percentage of NSF XSEDE usage [37], [38]. Gateway jobs that run on computing resources external to the gateways themselves can be considered to be workflows, at least nominally; such jobs must stage and transfer data, queue and run remote computations, then collect, transfer, and represent the results for the gateway enduser to collect. Figure 9 shows the growth of gateway usage (in CPU Hours) on the SDSC's Comet resource over the five year period 2015–2020. Note the log scale on the y-axis and the *order of magnitude*

Authorized licensed use limited to: University at Buffalo Libraries. Downloaded on March 20,2023 at 14:11:10 UTC from IEEE Xplore. Restrictions apply.

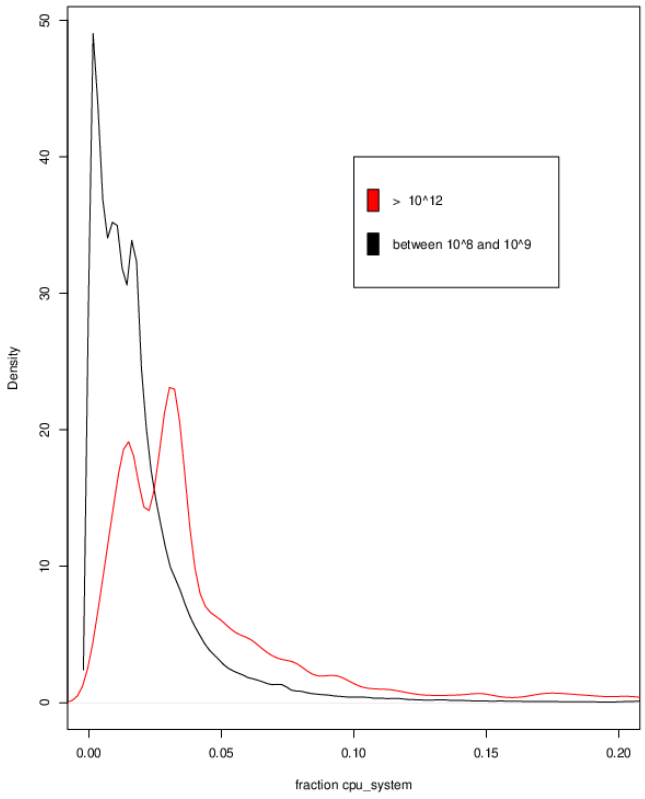

Fig. 7. Density plot of the number of jobs of HPC jobs running on XSEDE resources broken down by the fraction of time the O/S on the compute nodes ran in system mode. For jobs that transfer more data to the Lustre parallel filesystem, the CPU spends a much higher fraction of its time in the system mode than do jobs with typical Lustre usage. Red line: jobs with Lustre transmit rates greater than 10^{12} Bytes; Black line: jobs with Lustre transmit rates between 10^8 and 10^9 Bytes. The data are for XSEDE in the last year and the analysis includes 15M jobs.

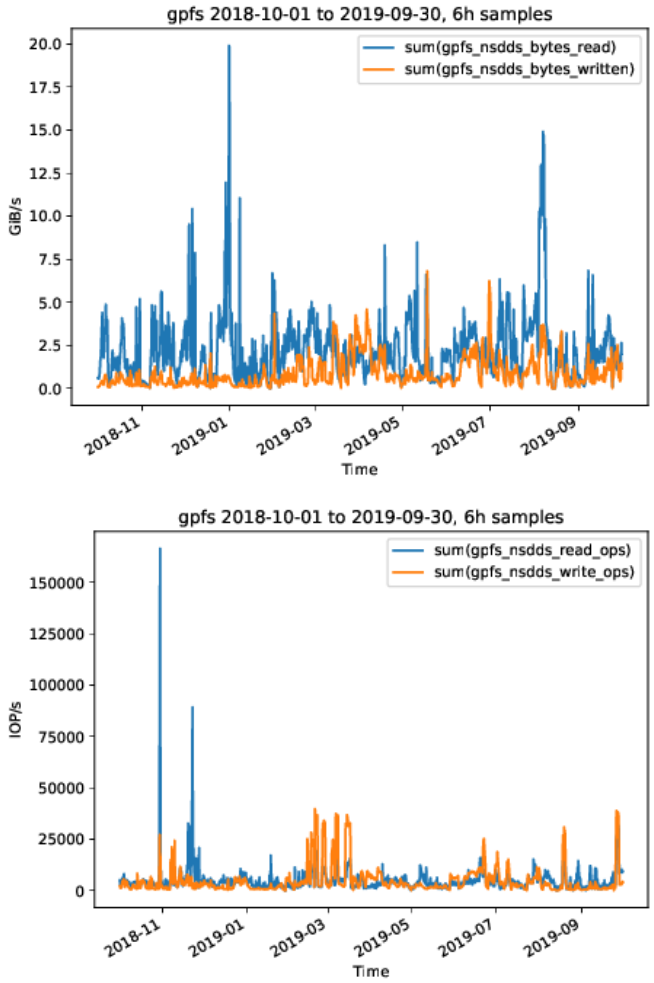

Fig. 8. Overall throughput to UB CCR's scratch filesystem over a one year period. Top plot shows read/write bandwidth, bottom plot shows I/O operations.

growth in the relative gateway usage, and thus, workflow usage.

Many of these gateways jobs are large bioinformatics workflows. Zhanglab [18] and Cipres [39] are two such bioinformatics gateways, and the two most heavily used in XSEDE. Figure 10 shows a hint of how such gateway workflows are different from other jobs on the same resource. The data plotted in the figure were gathered by XDMoD on SDSC Comet, an XSEDE resource, during 2019–2020. The analysis includes 4.5M jobs. As for the large parallel file system usage jobs, the CPU spends more time in the system mode on these gateway jobs than on the majority of other Comet jobs, again suggesting that substantial file transfer may be contributing to this overhead.

V. Discussion

Since workflows are prominently featured in bioinformatics and other data-intensive computations, we seek to position XDMoD to better monitor workflows. Labeling jobs as being or belonging to workflows, capturing the metadata that describes these jobs, assembling the data from various sources, determining appropriate metrics, and then designing means of reporting this information to users must all be considered.

Numerous pitfalls complicate monitoring data transfers in these informal workflows. These include: time lag between receiving data from different resources using the workflow, tracking extent of data transfer, and inferring network speeds. Additionally, collecting workflow information may require cooperation (data sharing) from the submitting user, gateway, and/or workflow framework.

A. Labeling workflow components

For informal workflows, the smaller jobs that form workflow components must be identified with labels or names. In the simplest case, consistent job naming schemes could help identify workflow related jobs so that their metadata could be processed together. This approach has the advantage of being simple and robust to the choice of resource manager. Alternately, something akin to Slurm workload characterization

Authorized licensed use limited to: University at Buffalo Libraries. Downloaded on March 20,2023 at 14:11:10 UTC from IEEE Xplore. Restrictions apply.

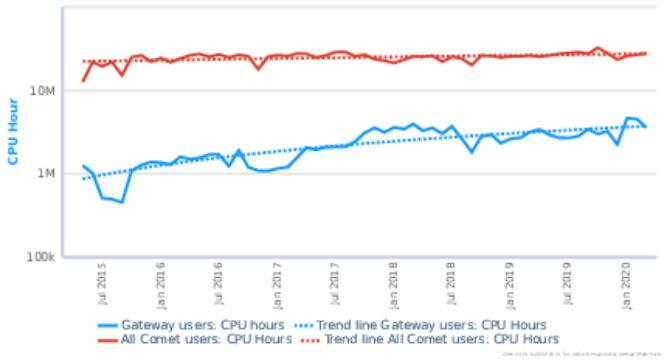

Fig. 9. XDMoD plot of the growth of gateway usage (CPU Hours) on SDSC Comet from 2015 to 2020. y axis is log scale. Note the order of magnitude growth in the relative gateway usage.

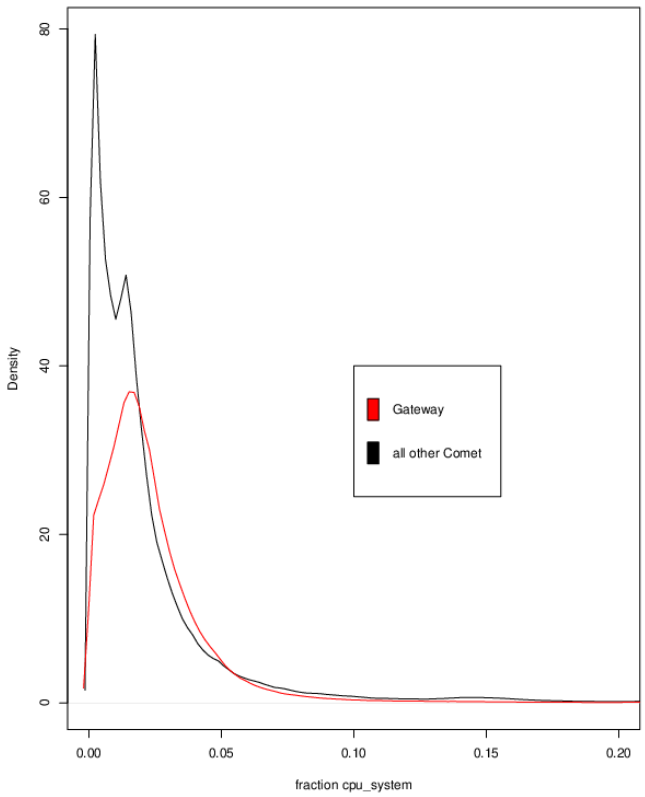

Fig. 10. Density plot of the number of jobs comparing CPU System % for Gateway jobs against other jobs, all run on the SDSC Comet XSEDE resource. Note the the larger fraction of time that the CPU spends in the System mode on the gateway jobs. The data, from 2019–2020, encompass 4.5M jobs.

keys (WCKey) could be used to label workflow component parts [40]. Such keys could be set up for tasks such as the genomics alignment described in Section III-B. Then, using the key, Slurm could tag the job. Subsequent processing of job metadata by XDMoD could then associate job elements labeled in this way. The disadvantage of such an approach is that it is specific to the Slurm resource manager.

Many apparently informal workflows consist of computational tools and software components, strung together to form workflows. Some such tools are quite sophisticated. Partnerships with tool developers could help encourage them to place the right hooks in their codes so that they are chained together to provide tracking metadata as well as accomplishing the needed computational work.

In the case of "formal" workflows such as the one described in Section III-A, some information about the workflow is known. Even then, however, some workflows are missed. Classification of the job software may be incorrect, or job names may be misleading. Even for these workflows, labeling and naming can be useful.

B. Job characteristics of interest

In order to monitor and characterize workflows, numerous types of data should be gathered, and the relevant metrics offered, for example:

- data transfer rates and network usage;
- CPU usage;
- CPU counters;
- network usage outside of HPC resources;
- and resource usage on non-HPC infrastructure, such as webservers and databases.

This list is not intended to be comprehensive, but rather to illustrate the current set of metrics useful for characterizing workloads. The onus is then on the ability to systematically harvest these metrics for workflows.

C. Capturing workflow information

Once workflows are labeled and identified, the next step is capturing their information. In the case of formal workflows, numerous frameworks such as Pegasus and others provide extensive means for monitoring different workflow stages and components. The challenge for a system such as XDMoD would lie in collecting, associating, and aggregating these data, in a fashion sufficiently general for different workflow specification standards to make use of it. Ideally, user communities could assist by developing and contributing adapters for different workflow specifications' data, enabling unified data collection. Additionally, for formal workflows, it should be possible to capture the workflow data transfers per job, since file staging is controlled within the workflow.

Informal workflows, which are not organized using workflow specifications or software, make information capture much more difficult. The data collection options presented by workflow frameworks are absent in this case. For informal workflows, associating data such as network transfers with a given workflow is also more of a challenge. These could

Authorized licensed use limited to: University at Buffalo Libraries. Downloaded on March 20,2023 at 14:11:10 UTC from IEEE Xplore. Restrictions apply.

be inferred based on time, however, this will result in false positives. Liu et al. [16] developed a system for automatically identifying I/O intensive jobs from the logs of the filesystem I/O nodes. They were able to mine the correlation between I/O traffic and application executions to obtain information on application I/O patterns. This data mining relies on the assumption that a given application has a consistent I/O pattern that remains the same between different HPC jobs. Such an approach might be useful here.

Once we know the data transfer is occurring inside the workflow, we can make inferences about network speed, etc. Monitoring information collected from gateway frameworks will be useful, in the case of formal workflows. Instrumentation of transfer nodes or logs from, e.g., Globus can also be put to use. It may be possible to note where low CPU User figures correspond to data fetch. Furthermore, known workflow jobs (e.g.) could be represented in a way that highlights the different aspects of the workflow, time taken in different inferred tasks, for instance. In addition to metrics currently collected by XDMoD, it will be important to tie into both additional monitoring tools, like perfSONAR[17] for networking and Globus[15] for large scale file transfer, as well as formal workflow tools like Pegasus and similar frameworks.

Although there are commercial products [41] that do infrastructure and application monitoring on clouds, including hybrid clouds, these products do not work with applications involving complex workflows or require large effort for every specific workflow. However, these products have sophisticated methods for doing synthetic (active) monitoring to simulate a path that a customer would take to execute their application. These paths can then be continuously monitored for availability, response time and functionality. Combining synthetic and passive monitoring (as in current verion of XDMoD) will allow for quantifying user experience.

VI. Conclusion & Future Work

Workflows will continue to grow in importance in computational science and engineering, particularly as more researchers engage from diverse fields of study. As more computational resources are located off premise (e.g., clouds), how do we ensure that these workflows achieve maximum efficiency in computation, storage, and network usage?

- Provide quantitative feedback on performance and utilization to both developers and users,
- Expose the underlying bottlenecks in all aspects of the environment,
- Gather information regarding workflows (formal and informal) across various disciplines and resources for a holistic view.

It will be increasingly important to tie together the disparate sources of metrics, including everything from workflow framework monitoring outputs, and network information, to provide a comprehensive view of overall workflow performance.

Acknowledgement

The authors would like to thank CCR Systems Administration staff member Sam Guercio and the members of the XDMoD development team, including Cynthia Cornelius, Steven M. Gallo, Greg Dean, Jeffrey T. Palmer, Benjamin Plessinger, Ryan Rathsam, Nikolay Simakov, and former members Rudra Chakraborty, Martins Innus, Thomas Yearke, Amin Ghadersohi, and Ryan Gentner.

This work was performed in part at the University at Buffalos Center for Computational Research [29]. We also acknowledge use of resources procured under NIH award S10OD024973 ("High Performance Data and Computing Infrastructure"), and under NSF award 1724891 (MRI: Acquisition of High Performance Computing Infrastructure to Support Computational and Data-Enabled Science and Engineering). This work was sponsored by the National Science Foundation under award ACI 1445806 for the XD Metrics Service (XMS). Vipin Chaudhary was funded by an NSF IPA grant. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Center for Computational Research, University at Buffalo, "XD Metrics on Demand (XDMoD)," https://xdmod.ccr.buffalo.edu, 2017.

[2] J. C. Browne, R. L. DeLeon, A. K. Patra, W. L. Barth, J. Hammond, M. D. Jones, T. R. Furlani, B. I. Schneider, S. M. Gallo, A. Ghadersohi, R. J. Gentner, J. T. Palmer, N. Simakov, M. Innus, A. E. Bruno, J. P. White, C. D. Cornelius, T. Yearke, K. Marcus, G. von Laszewski, and F. Wang, "Comprehensive, open-source resource usage measurement and analysis for HPC systems," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 13, pp. 2191–2209, 2014, cPE-14-0027.R1. [Online]. Available: https://dx.doi.org/10.1002/cpe.3245

[3] T. R. Furlani, M. D. Jones, S. M. Gallo, A. E. Bruno, C.-D. Lu, A. Ghadersohi, R. J. Gentner, A. K. Patra, R. L. DeLeon, G. von Laszewski, L. Wang, and A. Zimmerman, "Performance metrics and auditing framework using applications kernels for high performance computer systems," *Concurrency and Computation*, 2012.

[4] T. R. Furlani, B. I. Schneider, M. D. Jones, J. Towns, D. L. Hart, S. M. Gallo, R. L. DeLeon, C. Lu, A. Ghadersohi, R. J. Gentner, A. K. Patra, G. Laszewski, F. Wang, J. T. Palmer, and N. Simakov, "Using XDMoD to facilitate XSEDE operations, planning and analysis," in *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery (XSEDE '13)*. ACM, 2013, p. 8.

[5] J. C. Browne, R. L. DeLeon, C. Lu, M. D. Jones, S. M. Gallo, A. Ghadersohi, A. K. Patra, W. L. Barth, J. Hammond, T. R. Furlani, and R. T. McLay, "Enabling comprehensive data-driven system management for large computational facilities," in *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis (SC '13)*. ACM, 2013, p. 11.

[6] J. T. Palmer, S. M. Gallo, T. R. Furlani, M. D. Jones, R. L. DeLeon, J. P. White, N. Simakov, A. K. Patra, J. M. Sperhac, T. Yearke, R. Rathsam, M. Innus, C. D. Cornelius, J. C. Browne, W. L. Barth, and R. T. Evans, "Open XDMoD: A tool for the comprehensive management of high-performance computing resources," *Computing in Science and Engineering*, vol. 17, no. 4, pp. 52–62, 2015. [Online]. Available: https://dx.doi.org/10.1109/MCSE.2015.68

[7] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817 – 840, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819104000535

[8] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker, "The lightweight distributed metric service: A scalable infrastructure for continuous monitoring of

Authorized licensed use limited to: University at Buffalo Libraries. Downloaded on March 20,2023 at 14:11:10 UTC from IEEE Xplore. Restrictions apply.

large scale computing systems and applications," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2014, pp. 154–165.

[9] J. M. Brandt, A. C. Gentile, D. J. Hale, and P. P. Pebay, "Ovis: a tool for intelligent, real-time monitoring of computational clusters," in *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*. IEEE Computer Society, April 2006, pp. 8 pp.–.

[10] S. Feldman, D. Zhang, D. Dechev, and J. Brandt, "Extending ldms to enable performance monitoring in multi-core applications," in *Proceedings of the 2015 IEEE International Conference on Cluster Computing*, ser. CLUSTER '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 717–720. [Online]. Available: https://doi.org/10.1109/CLUSTER.2015.125

[11] Silicon Graphics Inc, Aconex, and Red Hat, "Performance Co-Pilot (pcp)," https://pcp.io, 2000.

[12] T. Evans, W. L. Barth, J. C. Browne, R. L. DeLeon, T. R. Furlani, S. M. Gallo, M. D. Jones, and A. K. Patra, "Comprehensive resource use monitoring for HPC systems with TACC_Stats," in *Proceedings of the First International Workshop on HPC User Support Tools*, ser. HUST '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 13–21. [Online]. Available: https://dx.doi.org/10.1109/HUST.2014.7

[13] K. Agrawal, M. R. Fahey, R. McLay, and D. James, "User environment tracking and problem detection with XALT," in *Proceedings of the First International Workshop on HPC User Support Tools*, ser. HUST '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 32–40. [Online]. Available: https://dx.doi.org/10.1109/HUST.2014.6

[14] T. Furlani, M. Jones, S. Gallo, A. Bruno, C. Lu, A. Ghadersohi, R. Gentner, A. Patra, R. DeLeon, G. Laszewski, L. Wang, and A. Zimmerman, "Performance metrics and auditing framework using application kernels for high-performance computer systems," *Concurrency and Computation: Practice and Experience*, vol. 25, p. 918, 2013.

[15] I. Foster, "Globus online: Accelerating and democratizing science through cloud-based services," *IEEE Internet Computing*, vol. 15, no. 3, pp. 70–73, 2011.

[16] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai, "Server-side log data analytics for i/o workload characterization and coordination on large shared storage systems," in *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2016, pp. 819–829.

[17] T. perfSONAR Project and contributors, "performance service-oriented network monitoring architecture (perfsonar)," https://perfsonar.net, 2005.

[18] W. Zheng, C. Zhang, E. W. Bell, and Y. Zhang, "I-tasser gateway: A protein structure and function prediction server powered by xsede," *Future Generation Computer Systems*, vol. 99, pp. 73 – 85, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18314705

[19] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. da Silva], M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17 – 35, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X14002015

[20] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva de la Hidalga, M. P. Balcazar Vargas, S. Sufi, and C. Goble, "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud," *Nucleic Acids Research*, vol. 41, no. W1, pp. W557–W561, 05 2013. [Online]. Available: https://doi.org/10.1093/nar/gkt328

[21] P. Amstutz, M. R. Crusoe, N. Tijani, (editors), B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Mnager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, and L. Stojanovic, "Common workflow language, v1.0. specification," Common Workflow Language working group, 2016, https://w3id.org/cwl/v1.0/ doi:10.6084/m9.figshare.3115156.v2.

[22] J. Gentry, C. Llanwarne, M. Lin, P. Magee, B. OConnor, O. Rodeh, and G. V. der Auwera, "Open workflow definition language (open wdl)," https://openwdl.org/.

[23] "XSEDE Science Gateway Catalog," https://www.xsede.org/ecosystem/science-gateways.

[24] S. Marru, L. Gunathilake, C. Herath, P. Tangchaisin, M. Pierce, C. Mattmann, R. Singh, T. Gunarathne, E. Chinthaka, R. Gardler, A. Slominski, A. Douma, S. Perera, and S. Weerawarana, "Apache airavata: A framework for distributed applications and computational workflows," in *Proceedings of the 2011 ACM Workshop on Gateway Computing Environments*, ser. GCE 11. New York, NY, USA: Association for Computing Machinery, 2011, p. 2128. [Online]. Available: https://doi.org/10.1145/2110486.2110490

[25] T. Kroes, H. Achterberg, M. Koek, A. Versteeg, W. Niessen, A. van der Lugt, P. van het Hof, B. van Lew, and B. Lelieveldt, "PIM: A visualization-oriented web application for monitoring and debugging of large-scale image processing studies," in *Medical Imaging 2020: Imaging Informatics for Healthcare, Research, and Applications*, P.-H. Chen and T. M. Deserno, Eds., vol. 11318, International Society for Optics and Photonics. SPIE, 2020, pp. 62 – 68. [Online]. Available: https://doi.org/10.1117/12.2541540

[26] M. D. Valerio, S. S. Sahoo, R. S. Barga, and J. J. Jackson, "Capturing workflow event data for monitoring, performance analysis, and management of scientific workflows," in *2008 IEEE Fourth International Conference on eScience*, 2008, pp. 626–633.

[27] E. Deelman, T. Peterka, I. Altintas, C. D. Carothers, K. K. van Dam, K. Moreland, M. Parashar, L. Ramakrishnan, M. Taufer, and J. Vetter, "The future of scientific workflows," *The International Journal of High Performance Computing Applications*, vol. 32, no. 1, pp. 159–175, 2018. [Online]. Available: https://doi.org/10.1177/1094342017704893

[28] J. Sperhac, K. Poinar, R. Jones-Ivey, J. Briner, B. Csatho, S. Nowicki, E. Simon, and A. Patra, "Ghub: Building a glaciology gateway to unify a community," in *Gateways '19 Proceedings*, September 2019, doi:10.17605/OSF.IO/JGHBZ.

[29] Center for Computational Research, University at Buffalo, "UB CCR Support Portfolio,," 2020, http://hdl.handle.net/10477/79221.

[30] Forster, R. R. and Box, J. E. and van den Broeke, M. R. and Miège, C. and Burgess, E. W. and van Angelen, J. H., "Extensive liquid meltwater storage in firn within the Greenland ice sheet," *Nature Geoscience*, vol. 7, pp. 95–98, 2014.

[31] Poinar, K and Joughin, I and Lilien, D and Brucker, L and Kehrl, L and Nowicki, S, "Drainage of Southeast Greenland Firn Aquifer Water through Crevasses to the Bed," *Front. Earth Sci.*, vol. 5, 2017.

[32] GHub project team, "GHub Gateway," https://vhub.org/groups/ghub/, 2019, accessed March 31, 2020.

[33] "National Snow and Ice Data Center (NSIDC), U.S." https://nsidc.org/. Accessed March 31, 2020.

[34] A. Petty, M. Tsamados, N. Kurtz, S. Farrell, T. Newman, J. Harbeck, D. Feltham, and J. Richter-Menge, "Characterizing arctic sea ice topography using high-resolution icebridge data," *The Cryosphere*, vol. 10, pp. 1161–1179, 2016.

[35] M. Studinger, "IceBridge ATM L1B elevation and return strength, version 2," Boulder, Colorado, USA. NASA National Snow and Ice Data Center Distributed Active Archive Center, 2013, updated 2020, doi: https://doi.org/10.5067/19SIM5TXKPGT. Accessed March 20, 2020.

[36] D. Kim, B. Langmead, and S. L. Salzberg, "Hisat: a fast spliced aligner with low memory requirements," *Nature Methods*, vol. 12, no. 4, pp. 357–360, 2015. [Online]. Available: https://doi.org/10.1038/nmeth.3317

[37] N. A. Simakov, J. P. White, R. L. DeLeon, S. M. Gallo, M. D. Jones, J. T. Palmer, B. D. Plessinger, and T. R. Furlani, "A Workload Analysis of NSF's Innovative HPC Resources Using XDMoD," *CoRR*, vol. abs/1801.04306, 2018. [Online]. Available: http://arxiv.org/abs/1801.04306

[38] J. M. Sperhac, R. L. DeLeon, T. R. Furlani, S. M. Gallo, M. Innus, M. D. Jones, J. T. Palmer, A. Patra, B. D. Plessinger, R. Rathsam, N. Simakov, J. P. White, R. Chakraborty, and G. Dean, "Managing computational gateway resources with xdmod," *Future Generation Computer Systems*, vol. 98, pp. 154 – 166, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18310732

[39] M. A. Miller, W. Pfeiffer, and T. Schwartz, "The cipres science gateway: A community resource for phylogenetic analyses," in *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, ser. TG 11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2016741.2016785

[40] A. Yoo, M. Jette, and M. Grondona, "SLURM: Simple Linux Utility for Resource Management," in *Lecture Notes in Computer Science*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin: Springer, 2003, vol. 2862, ch. Job Scheduling Strategies for Parallel Processing. JSSPP 2003.

[41] "Dynatrace," https://www.dynatrace.com.

Authorized licensed use limited to: University at Buffalo Libraries. Downloaded on March 20,2023 at 14:11:10 UTC from IEEE Xplore. Restrictions apply.