Emerging Frameworks for Advancing Scientific Workflows Research, Development, and Education

Henri Casanova[‡], Ewa Deelman[§], Sandra Gesing[¶], Michael Hildreth[‖], Stephen Hudson[**], William Koch[‡], Jeffrey Larson[**], Mary Ann McDowell[‖], Natalie Meyers[‖], John-Luke Navarro[**], George Papadimitriou[§], Ryan Tanaka[§], Ian Taylor[‖], Douglas Thain[‖], Stefan M. Wild[**], Rosa Filgueira[†], Rafael Ferreira da Silva[*]
[‡]Information and Computer Sciences, University of Hawaii, Honolulu, HI, USA
[§]Information Sciences Institute, University of Southern California, Marina Del Rey, CA, USA
[¶]Discovery Partner Institute, University of Illinois Chicago, Chicago, IL, USA
[‖]University of Notre Dame, Notre Dame, IN, USA
[**]Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA
[†]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
[*]National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN, USA

2021 IEEE Workshop on Workflows in Support of Large-Scale Science (WORKS) | 978-1-6654-1136-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/WORKS54523.2021.00015

***Abstract*—Lightning talks of the Workflows in Support of Large-Scale Science (WORKS) workshop are a venue where the workflow community (researchers, developers, and users) can discuss work in progress, emerging technologies and frameworks, and training and education materials. This paper summarizes the WORKS 2021 lightning talks, which cover four broad topics: (i) libEnsemble, a Python library to coordinate the concurrent evaluation of dynamic ensembles of calculations; (ii) EduWRENCH, a set of online pedagogic modules that provides simulation-driven hands-on activity in the browser; (iii) VisDict, an envisioned visual dictionary framework that will translate terms, jargon, and concepts between research domains and workflow providers; and (iv) Pegasus Kickstart, a lightweight tool for capturing workflow tasks' performance, including performance metrics from Nvidia GPUs.**

***Index Terms*—Scientific workflows, training and education, ensembles, python, concurrent computing, numerical optimization, simulation, Nvidia, GPU, monitoring**

I. Introduction

Scientific workflows have proved to be an excellent medium for representing scientific methods and for enhancing the efficiency and reproducibility of computational tasks. In the meantime, modern scientific applications are becoming more complex, run on heterogeneous resources, and require increasingly computational power, storage capacity, and network and I/O bandwidths. Workflows provide the necessary abstractions and mechanisms for efficiently automating the management of these applications across computing resources (e.g., clouds, HPC, edge, etc.), while providing fault-tolerance, capturing provenance records, and enabling reproducible results. Despite the impressive results to date, workflow research and development is still ad-hoc [1]–[3]. As a result, most workflow systems do not share common and interoperable interfaces, and therefore workflow applications are locked to a specific system; there is a steep learning curve for adopting a workflow system, but limited training materials are available. Furthermore, there is a strong need for close collaboration and intensive communication with domain scientists to successfully translate high-impact scientific methods into workflows. While there is a trend to make workflow editors and workflow dashboards as intuitive as possible, there is a lack of tools that support direct communication between scientists and workflow providers.

In this context, the workshop on Workflows in Support of Large-Scale Science (WORKS) has positioned itself as the primary venue for workflow researchers and developers to share and discuss innovative ideas to enhance the current workflow research and development landscape. Specifically, WORKS' lightning talks provide a venue where members of the community can introduce short talks on works in progress, emerging technologies and frameworks, and training and education materials to lower the entry barrier and thus increase adoption. This paper provides overviews of the four lightning talks from the 16th edition of the workshop (WORKS 2021):

***libEnsemble* (Section II) –** A Python library to coordinate the concurrent evaluation of dynamic ensembles of calculations. The library is developed to use massively parallel resources to accelerate the solution of design, decision, and inference problems and to expand the class of problems that can benefit from increased concurrency levels. This talk gives an overview of the libEnsemble package, highlighting the unique front-end, modular design, and the capability to run on a large range of platforms from laptops to thousands of compute nodes on supercomputers.

***EduWRENCH* (Section III) –** An online educational portal which emphasizes simulation-based pedagogy for teaching parallel and distributed computing concepts. This talk gives an overview of EduWRENCH's module about workflow concepts divided into 5 sections. Each section includes: a pedagogic narrative; a simulation-driven hands-on activity in the browser; a set of practice questions that students answer using simulation and reasoning, and whose answers can be revealed at will; and a set of open questions for instructors to be used as homework or exam questions.

***VisDict* (Section IV) –** A science gateway to enhance the communication between domain researchers and workflow

978-1-6654-1136-3/21/$31.00 ©2021 IEEE
DOI 10.1109/WORKS54523.2021.00015

providers by implementing a visual dictionary which translates terms, jargon, and concepts between both communities. VisDict will also define a methodology to represent and map different definitions of knowledge by using semantic representations (e.g., knowledge graphs using ontology).

***Pegasus Kickstart* (Section V) –** A tool for launching computing tasks, monitoring the behavior of tasks, and capturing information about tasks' performance and provenance data. This talk presents a lightweight tool, designed as an extension to Pegasus Kickstart, to capture monitoring information from Nvidia GPUs.

II. Coordinating Dynamic Ensemble Calculations with libEnsemble

By: Stephen Hudson, Jeffrey Larson, John-Luke Navarro, and Stefan M. Wild

libEnsemble [4], [5] is one of a number of extreme-scale workflow software packages, and primarily distinguishes itself via its generator-simulator function paradigm that sidesteps requiring users to define task dependencies in favor of data dependencies between configurable Python user functions. This allows the user to focus their attention on function logic.

libEnsemble is quick to install and get started with, requiring minimal dependencies. User functions are accessible, being written in Python, a language already familiar to a multitude of researchers. User applications can be launched with no modification via the supplied executors.

This composable design also lends itself to exploiting the large library of example user functions that are provided with libEnsemble, maximizing code re-use. For example, users can easily select an existing generator function while modifying a simulator function for their specific use case.

While libEnsemble provides a complete ensemble toolkit, including a task executor interface, its modular design also allows users to plug in components from other workflow packages. For example, in scenarios where the direct launching of MPI applications from the workers is infeasible, libEnsemble can use the Balsam workflow manager by swapping to the Balsam Executor.

libEnsemble coordinates its computations via a simple *manager-workers* paradigm as shown in Figure 1. Workers call the Python *generator* and *simulator* functions to perform any type of computation, then exchange data with the manager to determine and initiate future ensemble members. The manager and workers can run on one of three communication mediums: MPI (via `mpi4py`), multiprocessing (via Python's built-in module), and TCP (for distributed/cloud-based environments).

libEnsemble can dynamically change whether workers are used for simulators or generators and, as of version 0.8, can re-allocate resources available to each worker, even at a sub-node level. Again, this helps users avoid manually specifying directed task dependencies in a DAG or another paradigm in favor of focusing on data-flow between generator and simulator function instances.

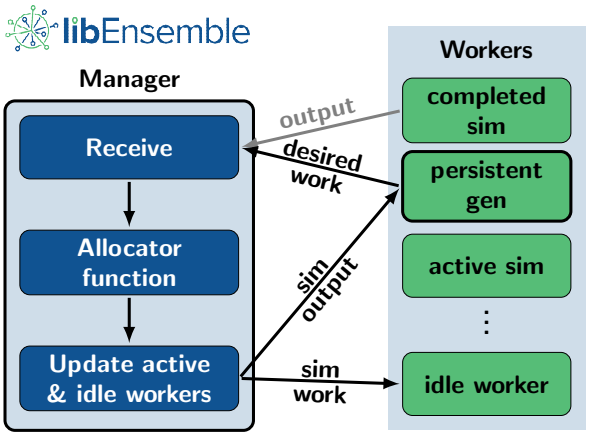

Fig. 1. Example of data movement between libEnsemble's manager and workers. Here, the manager receives output from a completed simulation and also some other work requested by a persistent generator. This is given to the allocation function, along with information about what workers are active or idle. The allocation function determines what work should be done and with what resources.

Some of our current use-cases take advantage of libEnsemble's advanced features. These include multi-fidelity studies that require dynamic scheduling of resources, generator-directed cancellation of previously issued simulations (includes killing of running simulations and retrieving partial results), and restarting ensembles.

We give a demonstration of one generator from the libEnsemble library. APOSMM, one of the motivating examples for libEnsemble, implements a parallel multistart optimization algorithm (Figure 2). APOSMM accepts or performs an initial sample of the parameter space and starts local optimization runs from points that do not have better points in their respective neighborhood. This neighborhood is adjusted dynamically as simulation outputs are observed from APOSMM's requested candidate points distributed to workers for evaluation via the simulation function.

libEnsemble is in active development in close collaboration with researchers from a variety of fields. Other example generator functions include optimization routines, machine learning, parameter estimation, or sensitivity analysis. Example simulator functions include particle accelerator simulations, subsurface flow, and applications using PETSc/Tao. libEnsemble has been run on many clusters and HPC systems, including thousand workers runs on ALCF's Theta and OLCF's Summit.

We hope to expand libEnsemble's user-base, support the needs of scientists performing ensemble computations, and continue to demonstrate the capabilities of Python in high-performance computing.

Acknowledgments. libEnsemble was developed as part of the software ecosystem for the U.S. Department of Energy exascale computing project (ECP).

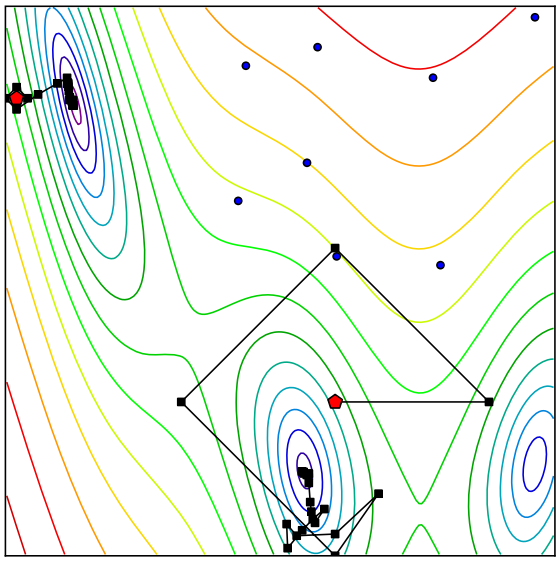

Fig. 2. Example of the APOSMM generator. Random samples with better points in their neighborhood are marked as blue circles; red pentagons show random points that start runs; and black squares are points arising from local optimization runs.

III. Learning Fundamental Workflow Concepts with EduWRENCH

By: Henri Casanova, Ryan Tanaka, William Koch, and Rafael Ferreira da Silva

Education and training in the field of Parallel and Distributed Computing (PDC) is known to be challenging (a testimony to this is the establishment of the EduHPC and EduPar workshop series, as well as the NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing). One of the challenges is that many relevant learning objectives are better achieved by providing students with hands-on active learning opportunities. This requires providing students with access to compute platforms and application workloads, which is (i) not feasible at many institutions; (ii) less feasible as platform and application configurations of interest are more complex and distributed; and (iii) requires that students be trained for several technologies and usage policies, which requires time and effort, and which can get in the way of achieving basic learning objectives.

The EduWRENCH project (https://eduwrench.org) aims at providing teaching and training material for fundamental PDC topics. The material is organized in modules. Each module, or subset thereof, can be used to enhance/complement the pedagogic content in existing courses and/or to provide training on particular topics. Independent learners can also complete the modules as a sequence. The main innovation is that each module provides several *in-the-browser hands-on activities*. These activities are in *simulation*, meaning that they do not require access to any particular hardware or software besides a Web browser. Students can achieve hands-on learning: they can explore various questions and develop answers to these questions by running simulations with different input parameters and by analyzing simulation output. Simulators are implemented using the WRENCH [6] and SimGrid [7] simulation frameworks.

The EduWRENCH site provides several modules, some introductory and others more advanced, such as the module dedicated to *workflows*.

A. Overview

The Workflow EduWRENCH module is available at https://eduwrench.org/pedagogic_modules/workflows/ and comprises five sections. Each section includes: a pedagogic narrative; a simulation-driven hands-on activity in the browser; a set of practice questions that students answer using simulation and reasoning, and whose answers can be revealed at will; and a set of open questions for instructors to be used as homework or exam questions. The sections in the module are as follow:

Fundamentals – The concept of a workflow as a task-graph with data dependencies is presented to the students. Then, using simulation, students are able to observe and reason about a workflow's execution on a single multi-core host with a given RAM capacity and a single disk.

Distributed Execution – The workflow execution is now distributed on the network, using a remote cluster of multi-core hosts and a remote data store. Students then observe and reason about the workflow execution on this platform.

Data Locality – A "cache" data store is now co-located with the cluster. Using simulation, students can then observe and reason about the effect of data locality on the execution.

Mixed Parallelism – In this section, workflow tasks are data-parallel programs, so that the workflow exhibits both task- and data-parallelism. Students are introduced to these concepts, and once again are able to understand and experiment with them hands-on via simulation for a workflow execution.

Capstone – This last section does not include any simulation activity. Instead, it is a small case-study in which students apply everything that they have learned in the module to make decisions regarding resource provisioning (given a fixed budget) to optimize the execution of a particular workflow.

We provide here an overview of the "Mixed Parallelism" section in which students first go through a pedagogic narrative that introduces relevant concepts driven by an example workflow. Students are then presented with a simulation scenario for that same example workflow, as depicted in Figure 3. A 5-task diamond workflow, in which three tasks are data-parallel with a different Amdahl's law parameter α, is to be executed on two compute nodes, each with 3 cores. The goal is to understand how using different numbers of cores for the data-parallel tasks impacts workflow execution and the performance thereof. To this end, the students can provide parameters to the simulation, as depicted in Figure 4. The simulation produces different kinds of output, including a Gantt chart of task executions (Figure 5) as well as a view of host/core utilization (Figure 6).

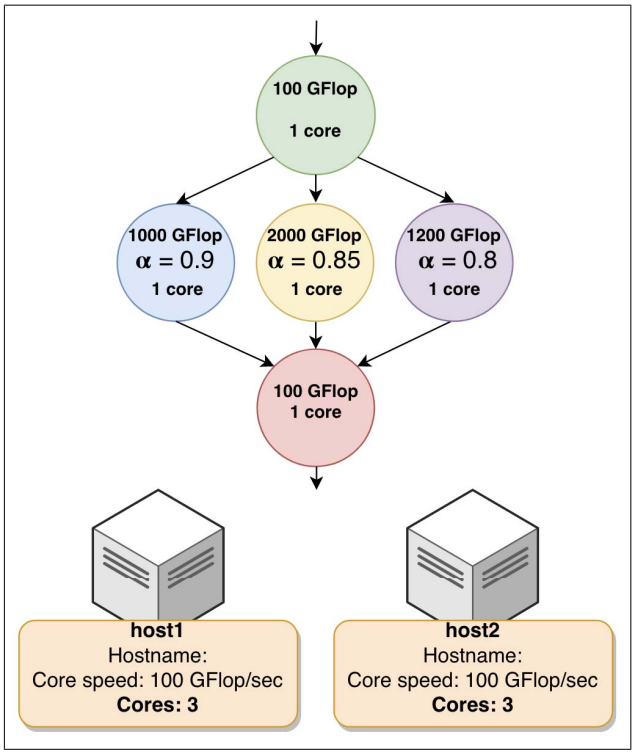

Fig. 3. Mixed parallelism simulation scenario.

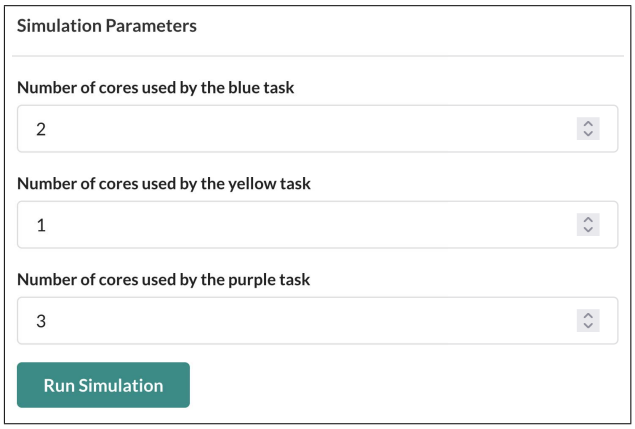

Fig. 4. Mixed parallelism simulation input form.

Students are then asked to answer several practice questions, using the above simulation to determine or double-check answers. For instance, one question is "Say that you must configure two of the data-parallel tasks to use 1 core, and the third one to use 3 cores. Which task should use 3 cores to achieve the shortest execution time? Come up with an answer based on reasoning and then check your intuition in simulation." When observing students going through this and other simulation activities, we find that the vast majority of

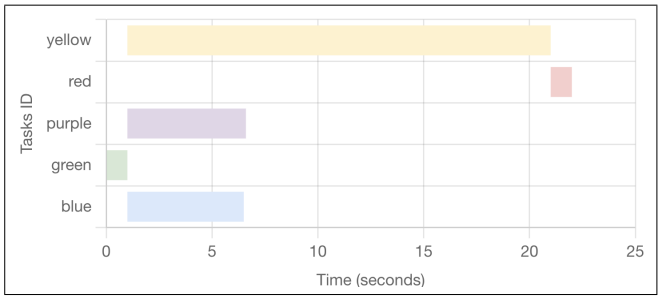

Fig. 5. Mixed parallelism simulation output: Gantt chart of task execution.

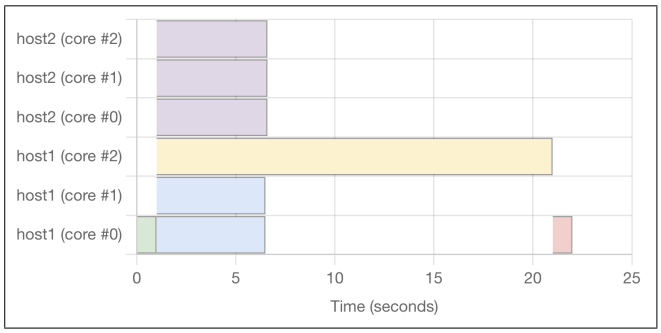

Fig. 6. Mixed parallelism simulation output: host/core utilization throughput time.

them use the simulation to explore scenarios/options that are not part of the specific questions of them, but often merely to satisfy their own curiosity.

B. Usage

The effectiveness of the simulation-based pedagogic approach used in EduWRENCH has been demonstrated via qualitative and quantitative user-studies in the classroom. In terms of the specific workflow module, it has been used successfully to date in more than four undergraduate university courses at our institutions. Early evaluation results obtained in the classroom have shown the effectiveness of the simulation-driven pedagogic approach, and student feedback has been used to improve the pedagogic content and its delivery [8]. Usage logs show that the module has been used by users worldwide as part of other university courses. Also, this module has been used for other purposes, such as for training incoming Masters students that join the SciTech research group at the USC Information Sciences Institute.

Acknowledgments. This work is funded by NSF contracts #1923539 and #1923621; and partly funded by NSF contracts #2103489, and #2103508. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

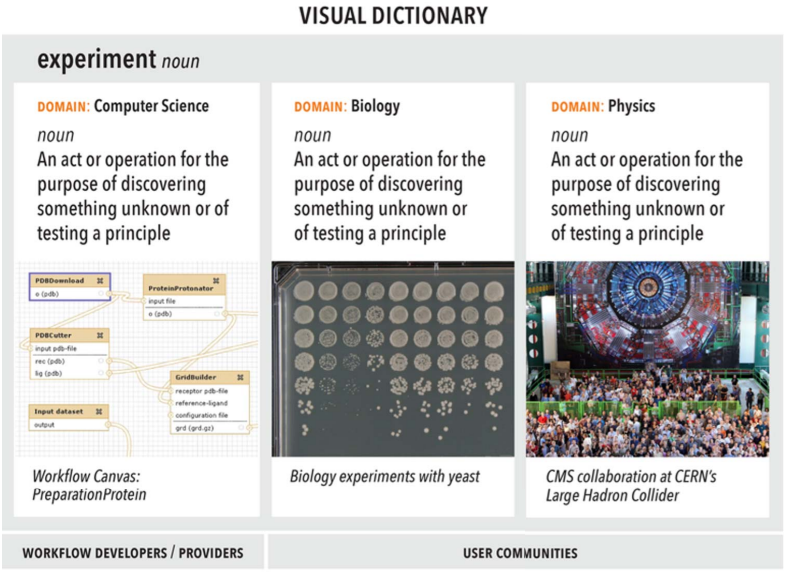

Fig. 7. An example for an entry for the visual dictionary where the definition is the same while the illustration illuminates the different aspects of an experiment per domain.

IV. VisDict: Enhancing the Communication between Workflow Providers and User Communities via a Visual Dictionary

By: Sandra Gesing, Rafael Ferreira da Silva, Ewa Deelman, Michael Hildreth, Mary Ann McDowell, Natalie Meyers, Ian Taylor, and Douglas Thain

Thousands of researchers [9] rely on scientific workflows for managing analyses, simulations, and other computations in almost every scientific domain [10], [11]. Scientific workflows have underpinned some of the most significant discoveries of the last decade, including the first detection of gravitational waves from colliding black holes [12]. The creation of workflows supporting a research topic requires an understanding of the targeted problem and can be a labor-intensive and error-prone process. One source of errors is the communication between domain researchers and workflow providers. While the agreement on one natural language for communication - or involving a translator - is the typical set up communication, there is a lack of tools or translators for communication between research domains and computer science. The project VisDict will fill this gap by providing a set of vocabularies in a science gateway to enhance communication. The goal is to present a definition for different domains and workflow providers and thus, serve as source for terms. Adding visualization will lower the communication barrier further following the saying "A picture is worth a thousand words."

VisDict just started and the project plans a series of surveys and interactions with the workflow and domain science communities. Surveys and focus groups are an integral part of the project, and will be crucial for identifying the terms that will compose the base of common knowledge between the communities. Another goal is to define a methodology to represent and map different definitions of knowledge. This goal will be accomplished through the use of semantic representations (e.g., knowledge graphs using ontology) in which computer- or domain-specific terms will be related to definition of terms in each domain. A science gateway will convey knowledge maps of the domain-specific terms in the form of visual dictionaries. The key for an excellent dictionary is to understand the pain points of user communities and workflow providers in the communication. The goal is to integrate definitions of the terms each group uses and they are familiar with and extend definitions via visual presentations. Figure 7 shows the example of the term "experiment". While the definition is the same for computer science, biology and physics, the illustration reveals the different aspects the researchers might have in mind when talking about experiments. Another use case for the visual dictionary would be a term like library. The definitions per domain would be different as well as the visualization per domain.

The vision is that the VisDict framework can be used as DaaS (dictionary-as-a-service) for many research domains and more IT-related domains beyond workflows.

Acknowledgments. This work is funded by NSF contracts #2100561 and #2100636.

V. A Lightweight GPU Monitoring Extension for Pegasus Kickstart

By: George Papadimitriou and Ewa Deelman

Compute jobs in the Pegasus workflow management system (WMS) [12] are wrapped using a lightweight C executable called "pegasus-kickstart" (Kickstart) [13], [14] that captures runtime job performance and provenance data. The toolkit

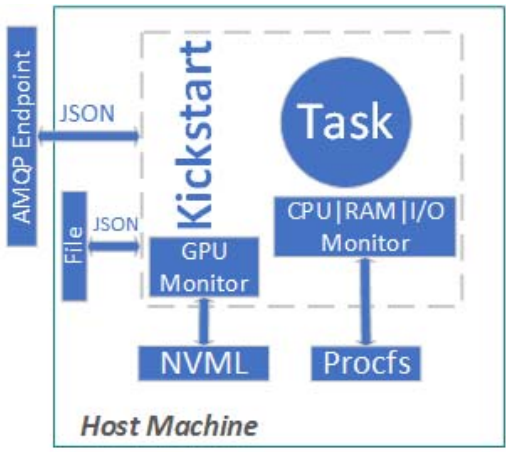

Fig. 8. Kickstart online monitoring.

collects useful information about the execution of the wrapped task such as the environment setup, performance data and output logs. Kickstart is a very important component of the Panorama data collection architecture [15]. In Pegasus' Panorama branch [16], Kickstart has been extended to include fine-grained monitoring capabilities that can pull resource usage statistics of workflow running tasks within a user-defined time interval. This information is then published to an AMQP [17] endpoint in JavaScript Object Notation (JSON) format so it can be ingested into a repository, saved to a storage system, or uploaded to an analysis framework (e.g., Elasticsearch [18]). Until now though, Kickstart could only collect statistics available in Linux's procfs [19], ignoring other subsystems, such as graphics processing units (GPUs).

A. Approach

To extend Kickstart's capabilities with monitoring support for Nvidia GPUs, we are leveraging Nvidia's monitoring library (NVML) [20]. NVML offers a C-based API for monitoring and managing various states of Nvidia GPU devices. We have extended Kickstart (Figure 8) with a lightweight C wrapper for the NVML library that queries the state of all the GPU devices available on an execution host machine. Kickstart polls for new GPU statistics on a user-defined interval and populates JSON formatted events. Kickstart GPU polling supports multithreading and creates a new polling thread for each GPU device, which is essential when sampling the PCI-Express bus utilization.

Events. During a job's execution there are three events produced by Kickstart containing information about the Nvidia GPUs.

- **kickstart.inv.online.gpu.env:** This event is produced once at the beginning of the job and it contains information about the GPU environment (e.g., number of GPUs, driver version, type of GPUs etc.)
- **kickstart.inv.online.gpu.stats**: This event is produced throughout the execution and it contains a snapshot of the GPU counters at that given time (e.g., GPU utilization, memory usage, power consumption etc.)
- **kickstart.inv.online.gpu.stats.max**: This event is produced at the end of the execution, and it contains max values observed during the run (e.g., max GPU utilization, max GPU temperature etc.)

All of the events are easily correlated with workflow runs and their respective jobs, since they are annotated with workflow related attributes (e.g., workflow uuid, dag job id). In the case of the GPU statistics event, there are some optional fields that are controlled via environment variables. The full list and description of the fields available in the produced events can be found on GitHub [21].

B. How to use

Installation. This tool is available under Pegasus Panorama branch [16] and can be used independently of Pegasus. Precompiled versions of this branch can also be found on the Pegasus download server [22]. Even though this tool is distributed with the Pegasus WMS it is a standalone tool and can be installed and used without using the rest of the system. On the Pegasus download server you will find the lightweight worker package that contains Kickstart and other essential Pegasus' tools (e.g., pegasus-transfer), which can be downloaded and installed independently.

Configuration. An example of using GPU-aware Kickstart with Pegasus is the "Predict Future Sales" workflow [23]. It has been configured to use the "–monitoring" flag during workflow generation. This flag instructs Pegasus Panorama to enable GPU monitoring for the jobs requesting GPUs, and orchestrates the data collection via an AMQP point.

To collect GPU traces using Kickstart as a standalone tool, one must set the following flags to "pegasus-kickstart":

- **-m** ¡interval¿: enables online monitoring and collects traces at every ¡interval¿
- **-G**: enables GPU monitoring (Note: this flag is considered only if the -m flag has been provided)

Finally, an environment variable sets the location where the statistics will be published (KICKSTART_MON_URL). Either file or AMQP endpoints can be specified. An example of a standalone invocation can be seen in Listing 1. For more we refer you to the "pegasus-kickstart" documentation [14].

C. Related and Future Work

Nvidia offers tools for detailed profiling and analysis (e.g., NVIDIA Nsight Tools), which provide in depth analysis of GPU kernels and can aid in debugging and performance optimizations. However, these tools add extra overhead that cannot be tolerated in production and they don't integrate well with other third party tools (e.g., monitoring tools of workflow management systems). Additionally, HTCondor [24] in version 8.8.9 introduced GPU monitoring, but it only offers statistics about the avg. GPU utilization and maximum memory usage, and no tracing is supported. With Kickstart we are able to

Listing 1 Example invocation of GPU monitoring

```
export KICKSTART_MON_URL = \
       rabbitmq://[USERNAME:PASSWORD]@hostname[:port]/api/exchanges/[VIRTUAL_HOST]/[EXCHANGE_NAME]/publish
or
export KICKSTART_MON_URL = file://filename
pegasus-kickstart <args> -G -m 10 ./exec
```

correlate GPU monitoring traces directly with workflow job executions.

We are currently working on extending the GPU monitoring feature to more devices such as AMD's ROCm GPUs.

Acknowledgments. This work is funded by DOE contract #DE-SC0012636 and NSF contract #1664162.

REFERENCES

[1] R. Ferreira da Silva, H. Casanova, K. Chard, D. Laney, D. Ahn, S. Jha, C. Goble, L. Ramakrishnan, L. Peterson, B. Enders, D. Thain, I. Altintas, Y. Babuji, R. Badia, V. Bonazzi, T. Coleman, M. Crusoe, E. Deelman, F. Di Natale, P. Di Tommaso, T. Fahringer, R. Filgueira, G. Fursin, A. Ganose, B. Gruning, D. S. Katz, O. Kuchar, A. Kupresanin, B. Ludascher, K. Maheshwari, M. Mattoso, K. Mehta, T. Munson, J. Ozik, T. Peterka, L. Pottier, T. Randles, S. Soiland-Reyes, B. Tovar, M. Turilli, T. Uram, K. Vahi, M. Wilde, M. Wolf, and J. Wozniak, "Workflows Community Summit: Bringing the Scientific Workflows Community Together," Mar. 2021.

[2] R. Ferreira da Silva, H. Casanova, K. Chard, T. a. Coleman, D. Laney, D. Ahn, S. Jha, D. Howell, S. Soiland-Reys, I. Altintas, D. Thain, R. Filgueira, Y. Babuji, R. M. Badia, B. Balis, S. Caino-Lores, S. Callaghan, F. Coppens, M. R. Crusoe, K. De, F. Di Natale, T. M. A. Do, B. Enders, T. Fahringer, A. Fouilloux, G. Fursin, A. Gaignard, A. Ganose, D. Garijo, S. Gesing, C. Goble, A. Hasan, S. Huber, D. S. Katz, U. Leser, D. Lowe, B. Ludaescher, K. Maheshwari, M. Malawski, R. Mayani, K. Mehta, A. Merzky, T. Munson, J. Ozik, L. Pottier, S. Ristov, M. Roozmeh, R. Souza, F. Suter, B. Tovar, M. Turilli, K. Vahi, A. Vidal-Torreira, W. Whitcup, M. Wilde, A. Williams, M. Wolf, and J. Wozniak, "Workflows Community Summit: Advancing the State-of-the-art of Scientific Workflows Management Systems Research and Development," Jun. 2021.

[3] R. Ferreira da Silva, H. Casanova, K. Chard, I. Altintas, R. M. Badia, B. Balis, T. a. Coleman, F. Coppens, F. D. Natale, T. Fahringer, R. Filgueira, G. Fursin, D. Garijo, C. Goble, D. Howell, S. Jha, D. S. Katz, D. Laney, U. Leser, M. Malawski, K. Mehta, L. Pottier, J. Ozik, J. L. Peterson, L. Ramakrishnan, S. Soiland-Reyes, D. Thain, and M. Wolf, "A community roadmap for scientific workflows research and development," *arXiv preprint arXiv:2110.02168*, 2021.

[4] S. Hudson, J. Larson, J.-L. Navarro, and S. M. Wild, "libEnsemble: A library to coordinate the concurrent evaluation of dynamic ensembles of calculations," *IEEE Transactions on Parallel and Distributed Systems*, 2021, to appear.

[5] S. Hudson, J. Larson, S. M. Wild, D. Bindel, and J.-L. Navarro, "libEnsemble user manual, version 0.7.1," Argonne, Tech report, 2021. [Online]. Available: https://libensemble.readthedocs.io

[6] H. Casanova, R. Ferreira da Silva, R. Tanaka, S. Pandey, G. Jethwani, W. Koch, S. Albrecht, J. Oeth, and F. Suter, "Developing accurate and scalable simulators of production workflow management systems with wrench," *Future Generation Computer Systems*, vol. 112, pp. 162–175, 2020.

[7] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, 2014.

[8] H. Casanova, R. Tanaka, W. Koch, and R. Ferreira da Silva, "Teaching Parallel and Distributed Computing Concepts in Simulation with WRENCH," *Journal of Parallel and Distributed Computing (JPDC)*, no. 156, pp. 53–63, 2021.

[9] E. Deelman, K. Vahi, M. Rynge, R. Mayani, R. Ferreira da Silva, G. Papadimitriou, and M. Livny, "The evolution of the pegasus workflow management software," *Computing in Science Engineering*, vol. 21, no. 4, pp. 22–36, 2019.

[10] I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields *et al.*, *Workflows for e-Science: scientific workflows for grids*. Springer, 2007, vol. 1.

[11] M. Atkinson, S. Gesing, J. Montagnat, and I. Taylor, "Scientific workflows: Past, present and future," 2017.

[12] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira 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.

[13] G. Juve, B. Tovar, R. Ferreira da Silva, D. Krol, D. Thain, E. Deelman, W. Allcock, and M. Livny, "Practical resource monitoring for robust high throughput computing," in *Workshop on Monitoring and Analysis for High Performance Computing Systems Plus Applications*, 2015.

[14] SciTech, "Pegasus Kickstart Documentation," https://pegasus.isi.edu/documentation/manpages/pegasus-kickstart.html?highlight=kickstart.

[15] G. Papadimitriou, C. Wang, K. Vahi, R. Ferreira da Silva, A. Mandal, L. Zhengchun, R. Mayani, M. Rynge, M. Kiran, V. E. Lynch, R. Kettimuthu, E. Deelman, J. S. Vetter, and I. Foster, "End-to-end online performance data capture and analysis for scientific workflows," *Future Generation Computer Systems*, vol. 117, pp. 387–400, 2021.

[16] SciTech, "Pegasus panorama," https://github.com/pegasus-isi/pegasus/tree/panorama.

[17] Pivotal, "Rabbitmq," https://www.rabbitmq.com/.

[18] "ELK stack," https://www.elastic.co/elk-stack, 2018.

[19] L. Foundation, "procfs," https://www.kernel.org/doc/html/latest/filesystems/proc.html.

[20] Nvidia, "NVML," https://docs.nvidia.com/deploy/nvml-api/index.html.

[21] SciTech, "Kickstart gpu events," https://github.com/pegasus-isi/pegasus/blob/panorama/src/tools/pegasus-kickstart/nvidia/README.md.

[22] ——, "Pegasus download server," http://download.pegasus.isi.edu/pegasus.

[23] ——, "Predict Future Sales Workflow," https://github.com/pegasus-isi/predict-future-sales-workflow.

[24] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience," *Concurrency and computation: practice and experience*, vol. 17, no. 2-4, pp. 323–356, 2005.