Accelerating complex modeling workflows in CyberWater using on-demand HPC/Cloud resources

Feng Li, Ranran Chen, Yuankun Fu, Fengguang Song, Yao Liang
Indiana University-Purdue University
Indianapolis, United States
{lifen, ranrchen, fuyuan, fgsong, yaoliang}@iupui.edu

Isuru Ranawaka, Sudhakar Pamidighantam
Indiana University
Bloomington, United States
{isjarana,pamidigs}@iu.edu

Daniel Luna, Xu Liang
University of Pittsburgh
Pittsburgh, United States
{del47,xuliang}@pitt.edu

***Abstract*—Workflow management systems (WMSs) are commonly used to organize/automate sequences of tasks as workflows to accelerate scientific discoveries. During complex workflow modeling, a local interactive workflow environment is desirable, as users usually rely on their rich, local environments for fast prototyping and refinements before they consider using more powerful computing resources. However, existing WMSs do not simultaneously support local interactive workflow environments and HPC resources. In this paper, we present an on-demand access mechanism to remote HPC resources from desktop/laptop-based workflow management software to compose, monitor and analyze scientific workflows in the CyberWater project. CyberWater is an open-data and open-modeling software framework for environmental and water communities. In this work, we extend the open-model, open-data design of CyberWater with on-demand HPC accessing capacity. In particular, we design and implement the LaunchAgent library, which can be integrated into the local desktop environment to allow on-demand usage of remote resources for hydrology-related workflows. LaunchAgent manages authentication to remote resources, prepares the computationally-intensive or data-intensive tasks as batch jobs, submits jobs to remote resources, and monitors the quality of services for the users. LaunchAgent interacts seamlessly with other existing components in CyberWater, which is now able to provide advantages of both feature-rich desktop software experience and increased computation power through on-demand HPC/Cloud usage. In our evaluations, we demonstrate how a hydrology workflow that consists of both local and remote tasks can be constructed and show that the added on-demand HPC/Cloud usage helps speeding up hydrology workflows while allowing intuitive workflow configurations and execution using a desktop graphical user interface.**

***Index Terms*—scientific workflow, hydrologic modeling, on-demand HPC**

I. Introduction

Scientific discovery often requires the execution of various coupled computational tasks using diverse data from local and remote resources. These tasks can be organized into stages, based on their data dependencies. A workflow management system (WMS) is a type of software system where an end user can describe the data dependencies of tasks, compose workflows, and launch such workflows for execution in designated computing environments. Nowadays, different types of computing environments are supported by popular WMSs. In a typical WMS, workflows are described as directed acyclic graphs (DAG) where each vertex is a computation task and the edges describe the data dependency between tasks. Such DAGs are then submitted to an execution environment, which can be a Cloud system, an HPC system, or a local computer. For example, in Pegasus WMS [1], one can provide the abstract workflow as a "DAX" file, and Pegasus translates it to an "execution workflow", which is then submitted to one of the supported execution environments. For a simple, small-sized workflow, a local computer can be used as the execution environment. For larger workloads, an HTCondor [2] pool of worker nodes can be used instead, in which case the tasks are mapped to a collection of worker nodes.

Although popular WMSs such as Pegasus allow users to utilize various types of execution environments to launch computation tasks, we find there are two limitations in practice. Firstly, the choice between local and remote execution sites is not flexible: workflows are typically only allowed to run in their entirety in local or remote environments. For workflows running in a local environment, computation power is limited; for workflows running remotely, it can take much longer to prototype, develop and debug. Secondly, from a workflow user's perspective, correctly preparing an abstract workflow can require a lot of effort for large workflows. For example, Pegasus WMS users must either manually create the DAX file, or use one of the supported programming interfaces to generate the DAX file. In contrast, desktop-based WMSs such as VisTrails [3], provide a feature-rich GUI-based frontend, which allows users to drag and drop widget boxes to form a complex workflow, and also gives comprehensive and timely information such as execution provenance.

To address these two practical issues, we present LaunchAgent, which provides desktop-based workflows with a mechanism of on-demand access to remote computing resources, so that rich configurations and trivial computation tasks can be done in the local environment, and only computation-

* Accepted by eScience '21 (2021 IEEE 17th International Conference on e-Science, Sept. 20-23, virtual).

ally expensive tasks are offloaded to powerful HPC/Cloud resources as needed. LaunchAgent began as a part of the CyberWater project [4], which aims to create an open-model and open-data framework to accelerate collaborative water research. VisTrails, a Python-based desktop workflow management software program, is currently adopted in the CyberWater project to support tasks such as provenance management and reproducible computing for exploratory computation tasks. Salas et al. [5] show that, using the VisTrails WMS, one can compose workflows consisting of different hydrological data sources and computational models and enable model coupling through its desktop graphical user interface (GUI). However, the previous integration of VisTrails and the CyberWater software framework only supports model execution on desktop computers, which usually have limited computation power and storage space. The proposed LaunchAgent library extends the CyberWater software framework so that users can select out specific computationally expensive tasks from the entire workflow during the GUI workflow configuration, and the chosen tasks are offloaded to HPC/Cloud resources automatically.

LaunchAgent supports both direct Slurm-based [6] access and Airavata gateway [7] access to remote computing resources. Such a design allows us to use both mid-size campus-based clusters and large-size grid computing resources (e.g., from XSEDE [8]). To illustrate the integration of LaunchAgent along with the CyberWater software framework and its VisTrails desktop interface, we run a real-world hydrological modeling workflow that has an HPC-enabled Variable Inflitration Capacity (VIC) model [9]. Our experiments show that by utilizing various types of HPC/Cloud resources, LaunchAgent is able to accelerate the VIC model significantly, with the convenience of intuitive user interactions.

In the rest of the paper, we begin by introducing background information on the CyberWater project and the Airavata gateway framework in Section II. Then, in Section III, we describe the design of the extended CyberWater software framework with the new on-demand HPC/Cloud mechanism. We present our experimentation results in Section IV. Related work is reviewed in Section V. Finally, we conclude the paper in Section VI.

II. Background

A. Cyberwater project

CyberWater is a collaborative project for creating a new infrastructure with an open-data and open-modeling software framework [4]. The CyberWater project aims at reducing user time and effort needed for hydrologic modeling studies by enabling flexible integration of diverse data sources and user models needed for executing complex workflows with on-demand remote HPC resources. It utilizes the Meta-Scientific-Modeling (MSM) framework [5] to address challenges of accessing heterogeneous data sources and integrating individual models. The MSM framework consists of four parts: a core (the MSM core), an interface with the Workflow engine, Data Agents, and Model Agents, as shown in the dashed box in Figure 1. The Data Agents are dynamically loaded components that describe how to connect to and retrieve data from different external data providers through the internet. The Model Agents are dynamically loaded components that describe the input/output and execution specifications of different hydrological models. The Core interacts with the Workflow Engine through the Workflow Interface to prepare and trigger individual tasks (e.g., data retrieving tasks and model execution tasks) specified in workflows.

VisTrails [3], a Python-based graphical science workflow system running in desktop environments, is currently adopted in the CyberWater project as the workflow management system, and provides the workflow engine that MSM core interacts with. Users can compose complex hydrology workflows using VisTrails graphic user interfaces. Then, the tasks defined in the workflow are captured by the MSM Core, which triggers actions such as data fetching, model execution, data processing/transformation, by means of Model Agents and Data Agents.

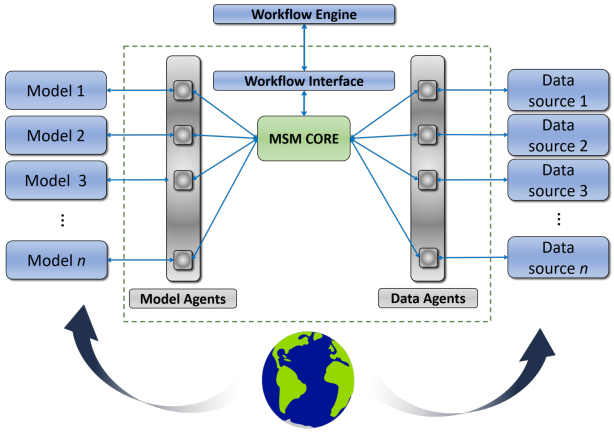

Fig. 1: Broad Scheme of CyberWater.

An in-depth comparison between the CyberWater MSM framework and other model/integration systems has been demonstrated in [5]. Previously, the CyberWater MSM framework allowed models to be executed in users' local computing environment (e.g., desktops) only. In this paper, we focus on the infrastructure design and support that enable the on-demand access to HPC/Cloud resources for better execution efficiency and for saving end-to-end workflow time. In order to integrate HPC/Cloud systems with the current CyberWater MSM framework, the Apache Airavata science gateway framework described below is adopted.

B. Airavata gateway framework

Apache Airavata [7] is a science gateway software framework to compose, execute, and monitor distributed applications from local clusters to computational grids and clouds. The Airavata framework is a collection of distributed micro service components of identity management, application and experiment management, job and workflow management, and digital object sharing management. Science Gateway Platform

(SciGaP) [10] provides Apache Airavata software as a hosted middleware service on Indiana University (IU) Intelligent Infrastructure systems[1].

SciGaP exposes public APIs that science gateways can use to outsource those general capabilities, as shown in Figure 2. Through the API services, researchers can register an application by specifying information such as executable script path, environment variables, input/output arguments and data files. The API services also allow the gateway administrator to add computing resources (clusters) so that when a SciGaP gateway user requests an experiment execution, the corresponding jobs will be created, launched at a designated HPC cluster, and monitored by the job management services.

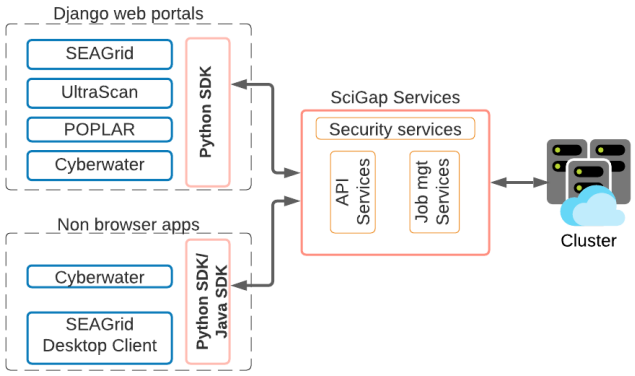

Fig. 2: SciGaP Integration Overview. SciGaP can expose API services to both browser-based and non-browser applications.

As illustrated in Figure 2, Science Gateways may have web portals, non-browser desktop/device based apps, or a combination thereof for their end-user researchers. To this end, the Airavata framework provides software development kits (SDKs) to connect with SciGaP and Apache Airavata services. The API services provided by SciGaP have been successfully used in different domains[2]. Web browser-based interfaces using Django web framework have been developed and provided as a reference to enable users to configure, launch, and monitor jobs/workflows. However, the browser-style integration is not always suitable for certain scientific workflow applications such as CyberWater that require feature-rich desktop based VisTrails workflow management tool. On one hand, CyberWater uses the Django web interfaces for tasks such as user registration, computing resource management, but on the other hand, it utilizes the Python SDK to configure, launch, and monitor applications. Integration of the SciGaP API services into the CyberWater system is described in Section III-B.

III. Methodology

In the CyberWater MSM framework, users' models can be plugged in easily, and without coding, using Generic Model Agent Tools, in which the model's execution environment is local, as shown in the center part of Figure 3. The VisTrails workflow system is currently adopted in the CyberWater framework, with which users can simply drag and drop component modules in their rich desktop environments to compose complex workflows. When the workflow is launched, MSM Core captures the computation tasks defined in the workflow, and then executes the tasks locally, which only utilizes a local desktop's computation power to carry out all the tasks.

The new on-demand HPC/Cloud mechanism extends the CyberWater software framework, which now includes a new remote HPC/Cloud execution environment provided by the LaunchAgent library to offload models specified in the Model Agents to HPC/Cloud resources on demand. The newly implemented LaunchAgent library, described in the following section, advances the Generic Model Agent Tool even further: it allows a model user to deploy the same models easily to different remote computing environments for faster task execution. Although we chose VisTrails as our current workflow engine, the MSM framework described in Figure 3 has a generic design that can be adapted to other workflow management systems.

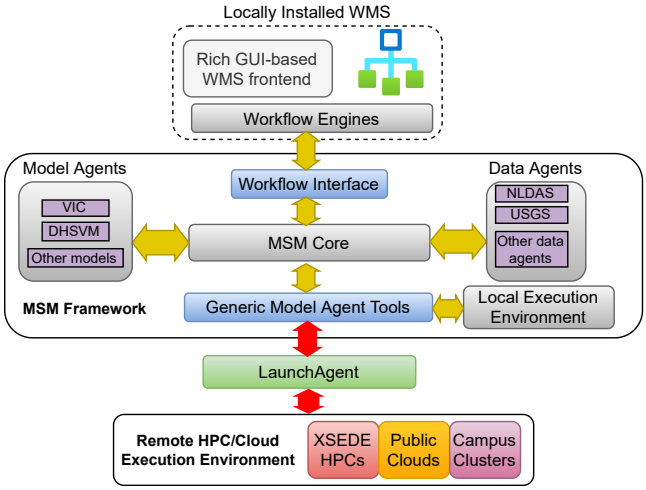

Fig. 3: The new HPC/Cloud-enabled CyberWater software framework with LaunchAgent integration. LaunchAgent extends the open-model and open-data design of the CyberWater MSM framework, and allows selection of computationally expensive tasks to be launched as remote jobs in different remote computing resources.

A. Design of LaunchAgent

LaunchAgent is a core component in providing on-demand HPC/Cloud access in the CyberWater software framework. LaunchAgent is designed to help workflow engines offload computationally intensive tasks to remote HPC/Cloud resources on demand through Python programming APIs. As shown in Figure 3, a locally installed workflow engine (e.g., VisTrails) on a desktop can manage a workflow as a graph of computational tasks. For the default local-computation setup,

[1]https://uits.iu.edu/services/intelligent-infrastructure
[2]SciGaP collaborators and clients: https://scigap.org/pages/collaborations.

tasks are scheduled by the local computer's operating system scheduler, and communication between tasks happens in the form of memory objects/local files. However, the computational power on local computers is usually limited.

With the help of LaunchAgent, specific tasks along with their input files can be offloaded to remote computing resources to accelerate the workflow. LaunchAgent manages user authorization so that users have proper access to remote resources. It also composes, submits, and monitors the computation tasks for the users. Through the design of a universal Python API, a local workflow engine can periodically check the tasks' status in remote sites, and download output files when the tasks are finished.

Listing 1 below shows how a VIC5 (VIC version 5.0) application [11] can be deployed in the IU BigRed3 cluster using the LaunchAgent interface. Firstly, a user can register as a Cyberwater gateway user through the gateway web portal[3]. Then, LaunchAgent can authorize the user based on gateway credentials and grant permission for the requested HPC resources (in the example of Listing 1, the IU BigRed3 system is requested from the user). The running environment of VIC5 (including input files and configuration files) is stored in the "vic" folder. The `run_monitor_job` function is a blocking call and the control returns until the job is finished on the HPC/Cloud side. The corresponding job's exit status is also returned and kept locally for provenance purposes.

```
# 1. Gateway authentication.
agent = GatewayAgent(gateway_username, gateway_passwd,
    exp_name = "test-gateway-cyberwater",
    site_name= "bigred3")

# 2. Upload local folder.
agent.upload_folder("vic")

# 3. Configure experiment, the run.sh file in the
# uploaded folder shall define how the job will be run.
agent.configure_exp(nodes = 1, ntasks_per_node = 2,
    email='somemail', walltime_in_mins= 5)

# 4. Run the job remotely and wait until job finishes.
agent.run_monitor_job()

# 5. Download all results to local directory.
agent.download_folder("./results_vic_gateway")
```

Listing 1: Example LaunchAgent usage with VIC hydrological model.

Typically, researchers have access to two types of cluster resources: on-campus clusters and remote clusters accessible through services such as XSEDE. At IU, there are high-performance/high-throughput clusters such as BigRed 3, Karst, et al [12]. Those resources typically require University IDs to operate and usually have limited computing power. On the other hand, extreme-scale computing infrastructure provided by XSEDE, such as TACC Stampede2 [13] and PSC Bridges-2 [14] usually provide significantly higher computing power. To this end, LaunchAgent has been designed to suit both settings through two channels: a direct Slurm-based channel, and an Airavata gateway based channel, both of which will be discussed in detail next.

[3]https://cyberwater.scigap.org/

B. Gateway-based channel

The gateway-based channel in LaunchAgent utilizes both Web-browser-based and non-browser-based interfaces provided by the SciGaP framework. First, the gateway administrator needs to configure the desired CyberWater application for the gateway created on SciGaP. This step is done through a web browser only once for all CyberWater gateway users, and is used to initialize the gateway environment and to prepare common metadata for all tasks submitted through the non-browser Python SDK interfaces. Figure 4 illustrates the main components of a gateway application and their dependencies. The gateway administrator needs to define the application inputs, outputs, and deployments. The application input describes the required inputs to run experiments and the output defines the experiment output results. The application deployment refers to the metadata related to connecting to the HPC clusters such as SSH Keys, job submission protocols, job queue information, and login account details. This information is configured in the **Group Resource Profile (GRP)** of the SciGaP gateway. Moreover, GRP is a collection of computing resources metadata and common SSH Keys to access computing resources (hosts). The gateway administrator can share GRPs with specific users or user groups to give access to computing hosts for their experiments. Data handling and transfer information are bundled in the **Gateway Resource Profile (GwRP)**, which contains storage access information for end users to upload and download experiment inputs and outputs. Furthermore, users can share GRP with other users or user groups to provide access for the data. Hence, GRP and GwRP create another layer of abstraction to manage access and data sharing easily with users and groups.

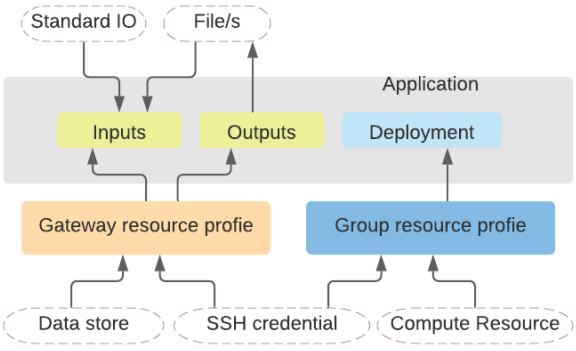

Fig. 4: Core components in an Airavata gateway application and their dependencies.

After the successful configuration of the gateway application, gateway end-users can submit experiments to the gateway under the created application. Figure 5 illustrates the sequence of operations supported by the Airavata Python SDK to execute an experiment through SciGaP services. Firstly, when a user initializes the LaunchAgent instance with a Gateway

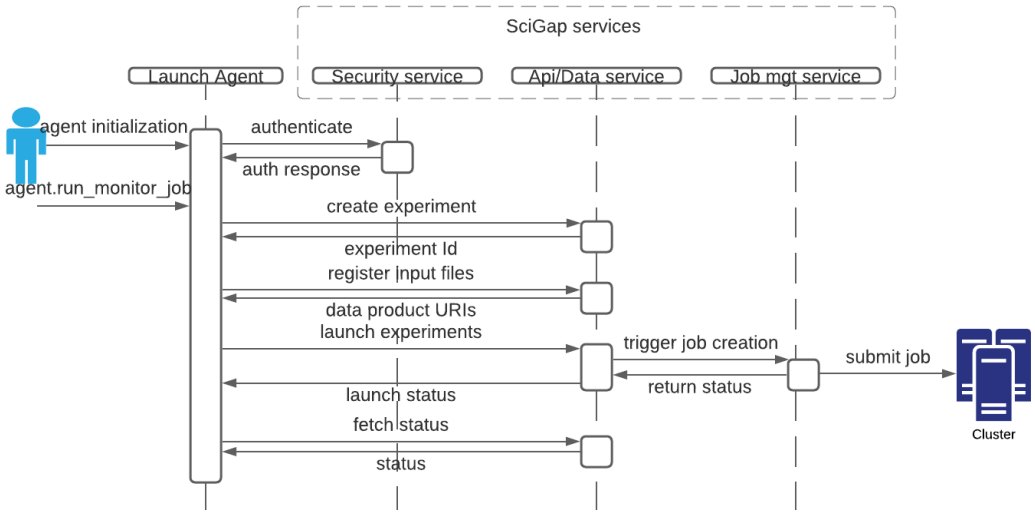

Fig. 5: LaunchAgent Operation Sequence with SciGaP services.

channel, LaunchAgent uses the Airavata Python SDK to authenticate the user based on the user-provided credentials with the SciGaP security service. Then an OAuth token is returned by the security service, and this token is used to access each subsequent API of SciGaP services via Python SDK. Secondly, during the $run\_monitor\_job$ call, an Airavata "experiment" will be created and the required files will be uploaded and launched into the job scheduler at the remote resource. The $agent.run\_monitor\_job$ function finishes when an exit status is returned while querying SciGaP API/Data service.

Currently, we use several computing resources from XSEDE, and the CyberWater gateway is registered in the XSEDE science gateways listing[4]. An XSEDE community account is set up and used to access the allocations on systems like IU Jetstream [15] and Pittsburgh Supercomputing Center (PSC) Bridges-2. We pre-configured those computing sites in the Airavata web portal as a group resource profile. Specifically, we allocate an SSH key-pair for this community account and copy the public key to the selected remote sites so that the gateway service can authenticate itself in order to access the registered HPC/Cloud resources.

Once a user joins the CyberWater gateway and is approved by the administrator, he/she gains access to the group of pre-configured HPC systems defined by the group resource profile. By using the CyberWater gateway username/password during the agent initialization function (see Listing 1), the LaunchAgent can use the selected site in the group resource profile to launch computation tasks.

C. Direct Slurm-based channel

We initially developed the direct Slurm-based channel for LaunchAgent using Paramiko SSH2 Python library[5]. The direct Slurm-based LaunchAgent has an operation sequence similar to the gateway-based method, as shown in Figure 5. In the gateway environment, the gateway usually utilizes a community account for all users, and works as middleware that submits jobs to remote sites for each registered user. Also, currently the gateway approach requires that computing resources be validated and registered in SciGap, before gateway users can deploy applications on them. In contrast, with the direct Slurm-based method, each user has his/her own login credentials to remote resources, which gives direct access to the HPC/Cloud systems without going through the gateway. A typical use case of the Slurm-based LaunchAgent is campus-based clusters, to which university students/faculties have direct access.

Similar to the gateway programming interface example shown at Listing 1, the user needs to configure the folder to be uploaded, which includes input data and running configurations. LaunchAgent automatically archives the user's folder, submits it to remote computing resources, and retrieves output data once the job finishes. For authentication, Slurm-based LaunchAgent allows users to authenticate themselves using their HPC logins, with either passwords or SSH key pairs. Here, each user provides his/her personal login credentials, and such credentials are not saved in the Slurm-based LaunchAgent.

Initially we developed our prototype using IU BigRed3 supercomputer. Using the generic Slurm-based agent, we were able to launch parallel programs from personal computers. To accommodate larger computation, we also added support for XSEDE supercomputers, such as PSC Bridges-2 and TACC Stampede2. We realize that different systems have specific requirements for user authorization and authentication, and have dealt with them in the direct Slurm-based LaunchAgent implementation. For example, PSC Bridges-2 requires a user to upload his/her ssh login public key through a specific key management web page (operated by PSC), and Stampede2 requires multi-factor-authentication (MFA) for each ssh session.

[4]https://www.xsede.org/web/site/ecosystem/science-gateways/gateways-listing

[5]https://www.paramiko.org/

Apart from the XSEDE HPC resources, the direct Slurm-based LaunchAgent also supports cloud computing sources such as Google Cloud Platform and JetStream Cloud. To initialize the Slurm clusters from those Cloud providers, we utilize the Slurm-GCP tool[6] for Google Cloud Platform, and JetStream Elastic Slurm Cluster tool[7] for JetStream Cloud. Both Slurm-GCP and JetStream Elastic Cluster allow dynamic resizing of clusters by allocating more cloud virtual machines on demand.

D. Integration with Cyberwater frontend

To incorporate the LaunchAgent component into the CyberWater system, we have created an "HPC" module, which can be dragged and dropped in the VisTrails WMS desktop frontend. The HPC module extends the functionality of the Generic Model Agent Tools (GT) to provide on-demand HPC usage in the CyberWater system. Figure 6 shows the HPC module and its configuration window in the CyberWater-VisTrails GUI interface.

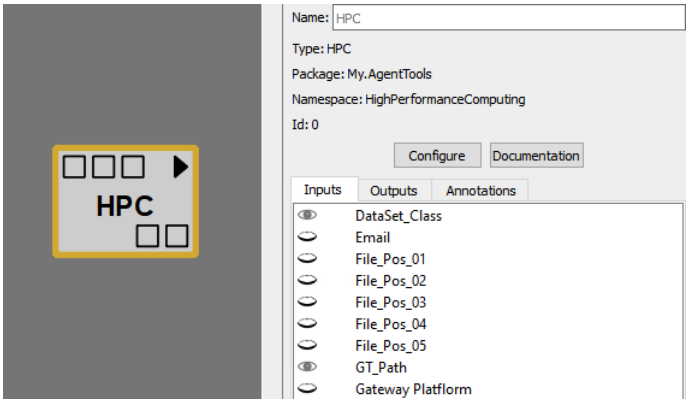

Fig. 6: The HPC Module and its configuration window.

In Figure 6, only part of the supported input configurations (ports) are showed for demonstration purposes. We also provide a more complete list of the configurable ports in Table I. A real-world example using the HPC module is described in Section IV-A, where we configure the VIC5 executable for remote execution at the PSC Bridges-2 supercomputer.

IV. Usecases and performance evaluation

In this section, we demonstrate through a real-world hydrology workflow example how LaunchAgent can be used in the CyberWater project to provide convenient, on-demand access to HPC/Cloud systems. We then focus on the the most computationally expensive VIC5 module in the workflow, analyzing and comparing its performance when launching it using LaunchAgent in different execution environments.

Input Port Specification	
Port name	Explanation
DataSet_Class	To get DataSet_Class brought in from MainGenerator.
Email	The email address for job notifications.
Execution_File	The path of the executable program file.
Execution_Folder	The path of the source code folder, which needs to be compiled in HPC.
File_Name_Prefix	The prefix of the output result files.
File_Pos_01-05	To set the output dataset in the corresponding column of output files.
GT_Path	To get GT_Path brought in from MainGenerator (the working directory where simulation files are saved).
Gateway_Platform	To choose which HPC/Cloud platform to use for the gateway-based channel, including IU Karst and PSC Bridges-2.
Output_Name_01-05	To output the dataset needed from these ports.
Project_Name	The name of project running in HPC.
Password	The password used for logging on the specifed platform.
Ready_List	To connect the output of ForcingDataFileGenerator, AreaWiseParamGenerator and InitialStateFileGenerator if it exists.
Runtime	Estimated duration of the task needed from the platform to which user wants to apply, in "hh:mm:ss" format.
SSH_Platform	To choose the HPC/Cloud platform to use with the direct Slurm-based channel, including BigRed3 supercomputer at IU, PSC Bridges-2 (both extreme-memory and regular-memory queues), TACC Stampede2, XSEDE JetStream cloud and GCP (Google Cloud Platform) cloud.
Username	The username used for logging on the specified platform.

Output Port Specification	
Port Name	Explanation
Output01-05	To output the Dataset needed from these ports.

TABLE I: The input/ouput port specifications of the HPC module configurable through the CyberWater GUI frontend.

A. HPC-enabled workflow with LaunchAgent

To demonstrate that LaunchAgent integrates seamlessly with the current CyberWater framework, we use the CyberWater-VisTrails graphical user interface (GUI) to compose a hydrology workflow. This workflow uses the VIC5 model to study the West Branch Susquehanna[8] river basin for the period 1995-1996. This study area covers more than 17,700 square kilometers.

As shown in Figure 7, we define the time/space range of the studied problem by configuring the **TimeRange** and **WBSusquehanna (SpaceRange)** module. Then, from the NLDAS (North American Assessment-Land Data Assimililation System [16], [17]), we use the **Hourly NLDAS Forcing for VIC5** group module, which internally contains 7 instances of **NLDASAgent** data agents (temperature, longwave radiation, shortwave radiation, precipitation, pressure, water vapor pressure and wind speed) and the corresponding unit conversions. This way, CyberWater will pull the seven chosen types of datasets of the specified time/space range from the NLDAS site to the local cache directories. After that, we use several

[6]https://cloud.google.com/solutions/deploying-slurm-cluster-compute-engine
[7]https://github.com/XSEDE/CRI_Jetstream_Cluster

[8]USGS information used: https://waterdata.usgs.gov/pa/nwis/uv?site_no=01553500

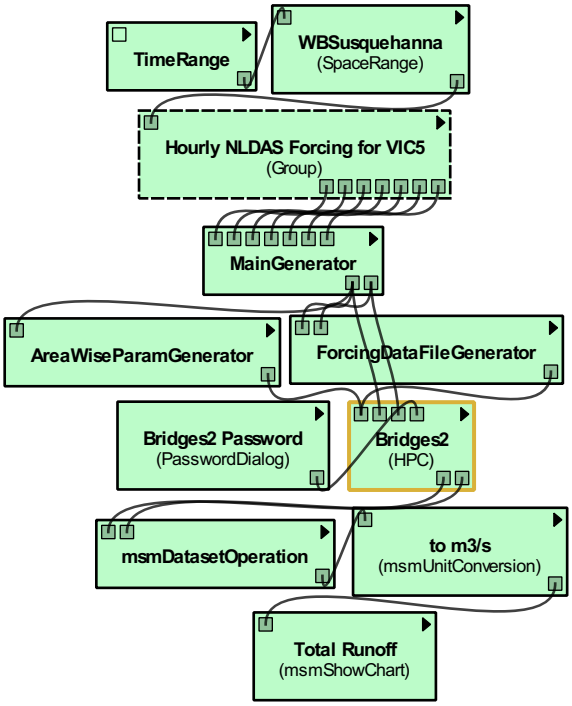

Fig. 7: An example of remote computing with HPC module.

"Generator" modules to prepare the forcing data and parameter files for the VIC5 execution. Then, the **Bridges-2 (HPC)** module is used to launch the VIC5 hydrological model to the PSC Bridges-2 system. During the remote launch, all prepared data are sent to the remote HPC/Cloud system; the VIC5 model is then executed in the remote HPC/Cloud environment; and, finally, all output results are downloaded to the specified local output directory. This entire launch process is conducted automatically by the execution of HPC module in the workflow.

CyberWater also provides post-analysis and visualization modules, such as the **msmShowChart** module. The msmShowChart module is used to display useful information such as surface runoff and baseflow for view. For example, Figure 8 shows the the total runoff, which is the sum of baseflow and surface runoff time series of the studied example workflow. The "baseflow" means the portion of the streamflow that is sustained between precipitation events, and it is contributed by slowly moving water within the porous media due to soil moisture or groundwater. "Surface runoff" describes the excess amount of water from rain, snowmelt or other resources that moves over the land surface. The **msmDatasetOperation** module in Figure 7 sums the baseflow and surface runoff computed from VIC5 to obtain the total runoff.

Note that all user interactions in this subsection happen in graphical user interfaces in the local desktop/laptop workflow environment, where the users have no need to login to remote HPCs. By utilizing the "HPC" module in the CyberWater GUI, we are able to launch the selected tasks remotely using more powerful computing systems. Importantly, CyberWater allows researchers to integrate the input/output of such remote execution seamlessly with local workflow management systems. Also, it is convenient to substitute the HPC module with the local execution module for VIC5 if the user's workflow does not require much computation power during the model prototyping and debugging stages.

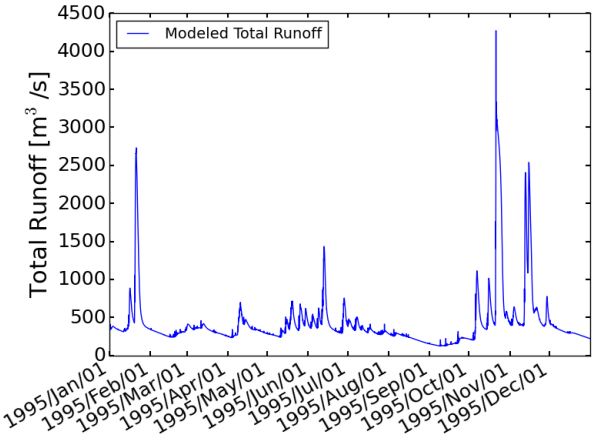

Fig. 8: VIC5 simulated total runoff from the workflow of Figure 7 for the West-Branch Susquehanna river basin, displayed by the msmShowChart module in CyberWater-VisTrails GUI interface.

B. Performance evaluation

We evaluated the performance gain of applying the HPC resources over the default local execution by comparing the elapsed time of the VIC5 computation module when we use local desktop environment versus when we use different remote execution sites. In this part of the experiments, we use the same study area and time period as in Subsection IV-A, to showcase the performance gains that CyberWater on-demand HPC access can provide.

For both local and remote execution configurations, the VIC5 workflow is launched from the same Windows 10 laptop system, which is equipped with 2 CPU cores and 16GB RAM. This laptop is connected to the IU campus network through a wired Gigabit Ethernet switch installed in a Computer Science research lab at the Indiana University-Purdue University Indianapolis (IUPUI) campus.

In the local run, we use the VIC5 windows binary currently shipped in the CyberWater installation[9]. For this local setup, all software and data are self-contained: the VIC5 executable reads data cached locally, and computation happens solely on the local computer.

In a different setup with the proposed LaunchAgent tools for on-demand access to HPC and Cloud, we use the same laptop system, and change the default local execution module for the LaunchAgent module. For the HPC computing

[9]This executable uses the "classic" VIC driver, and was originally built with the Cygwin POSIX-compatible environment. For more details, please refer to https://vic.readthedocs.io/en/vic.5.0.1/Documentation/Drivers/Classic/ClassicDriver/.

environments, we use the SKX partition in the Stampede2 system and the Regular Memory (RM) partition for the Bridges-2 system, which has 48 and 128 CPU cores on each compute node, respectively. For the (IU) Jetstream Cloud, we use a dynamically-sizing Slurm cluster created using Jetstream Elastic Cluster toolkit. We include a copy of the VIC5 source code and the corresponding build script during the *agent.upload_folder* step, so that the VIC5 executable is built on the remote system at the beginning of each remote execution. Note that LaunchAgent also allows the use of pre-built packages, in which case the application executable and its dependencies are pre-configured in the HPC system by the administrators before the initialization of LaunchAgent.

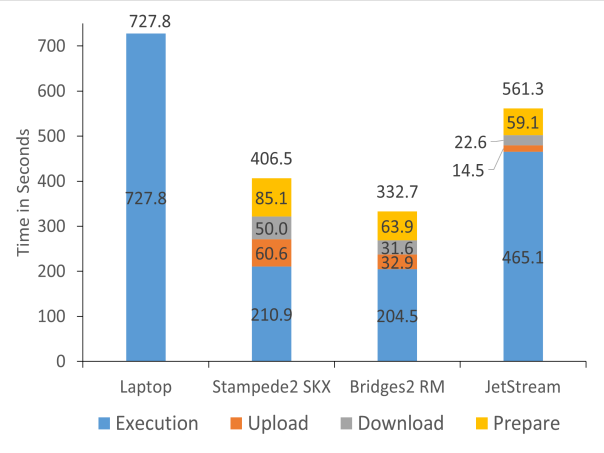

Fig. 9: Time breakdown of running VIC5 module with LaunchAgent in different environments.

Figure 9 shows the comparison of the elapsed time of the VIC5 module. While the VIC5 module takes over 12 minutes in a local execution environment, we observe 1.8, 2.2 and 1.3 times speedup when we use LaunchAgent to launch the same model to Stampede2, Bridges-2 HPC, and the Jetstream cloud system, respectively. For each remote launch, the elapsed time shown in the figure includes the time for uploading input, waiting for remote execution and downloading output[10]. The figure also shows the remote "prepare" time, which includes the job queue waiting time and other work such as file-system operations before and after the VIC5 execution. We have used optimization techniques for different HPC systems, such as AVX512 vectorization in Stampede2 and vendor-optimized libraries for AMD CPUs in Bridges-2. From the figure, it's clear that HPC systems such as Bridges-2 and Stampede2 can greatly reduce the VIC5 model execution time, however, they require longer preparation time due to shared job queues. The Jetstream Cloud provides limited computation power for model execution, but has advantages in data transfer since the cloud instances are physically located closer to the end-user (also hosted in the IU network). Jetstream also requires less preparation time because resources (Cloud virtual machines) can be allocated on demand. Overall, the results suggest that for tasks with different characteristics (e.g., computation-bound or communication-bound), LaunchAgent provides the flexibility to utilize different types of resources for achieving the best possible performance.

[10]In all remote-launching experiments, a total of 70MB of compressed input data are uploaded and 600MB of result data are downloaded.

Note that in experimental results from Figure 9, each individual remote execution only utilizes one process. To examine the performance behavior of LaunchAgent with different resource allocation sizes, we ran a similar workflow with various numbers of processes on the Bridges-2 system. In this experiment, we use the same West-Branch Susquehanna river basin study area, but with a longer 5-year simulation period, which takes around 3440 seconds for local execution. The uploaded/downloaded file archives for remote executions are 352.8 MB and 1.17 GB, respectively. The time breakdown results are shown in Figure 10, where we show the average time breakdown for 3 runs. Figure 10 shows that we can achieve at most 10.6 times speed up when 64 processes are used, compared with the local execution. However, using a process number larger than 32 does not bring much performance benefit: although the VIC5 simulation scales well, the overhead of upload, preparation, and download time becomes more significant. We are currently working on adding heuristics so that CyberWater users can get a suggested or preferred number of processes for their applications. The suggested number of processes can be estimated using information such as the current occupation of job queues in the remote systems, the I/O and scaling patterns of the applications.

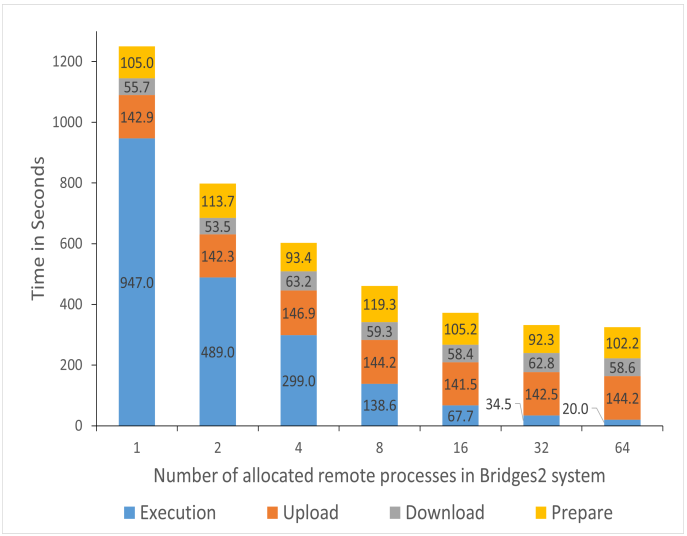

Fig. 10: Time breakdown of running VIC5 module with LaunchAgent on PSC Bridges-2 system with different numbers of remote processes.

V. Discussion and Related Work

Apart from the VIC5 model, the CyberWater software framework currently also supports other hydrological models such as DHSVM (Distributed Hydrology Soil Vegetation Model), VIC4 (an older version of the VIC model), and a

routing model. Researchers are encouraged to bring in their own hydrological models and datasets. To serve such needs more efficiently, we have prepared detailed manual documentation so that collaborating researchers can follow the step-by-step examples to integrate their own models and data. Also, the CyberWater project team regularly holds study groups and user workshop sessions, and recordings of such sessions are also archived and openly available at the CyberWater project page at CUAHSI website[11].

Several water-science related collaborative information systems have been developed in recent years. Futurewater [18] is an ongoing effort at Indiana University to answer crucial questions such as the climate change effects on Indiana's water resources. WaterHub [19] uses a GIS-enabled model sharing platform to allow users not only to run simulations online but also to publish and share model results. Those methods typically utilize browser-based Web access, which is convenient but can be less feature-rich than a local desktop solution such as VisTrails. In comparison, the proposed HPC/Cloud enabled CyberWater software framework combines the benefits of the rich local environment and powerful remote executions environments, so that trivial tasks such as format transformation, GUI operation can be efficiently done locally, with the computationally expensive components offloaded to remote sites.

Task-based workflow systems such as PyCOMPSs [20] and Pegasus [1] provide Python-based interfaces for users to define workflows containing multi-task dependencies. Parsl [21] (Pervasive Parallel Programming in Python) is a Python-based parallel scripting library; it allows developers to express parallelism in Python code. The advantages of those workflow systems are perpendicular to the current CyberWater system design. Although VisTrails is currently adopted in CyberWater as the workflow management system, the on-demand HPC/Cloud computing capacity can be adapted to other workflow systems too, under the generic CyberWater MSM workflow interface design.

The on-demand usage of remote computing resources has been previously explored in works such as KNIME [22] and Taverna [23], [24]. KNIME is a workflow management tool largely used in the cheminformatics domain and it allows users to compose data analytical pipelines through its GUI frontend. KNIME allows selective components submitted to remote clusters for more efficient executions. However, such support is only available through its commercial KNIME Cluster Executor extension[12]. In comparison, the on-demand HPC/Cloud support in the CyberWater software framework is freely available to all CyberWater users. Taverna allows users to compose workflows from a mixture of distributed web services, local scripts, and other service types [24]. Computationally intensive operations, such as genome-scale analyses, can be performed remotely regardless of local infrastructure. This approach brings challenges in reliability: the externally-hosted services may not function correctly due to factors such as service maintenance, outage, or interface upgrades. By contrast, CyberWater with HPC/Cloud support does not rely on designated remote service providers or computing environments, and as a result, computationally intensive tasks can be offloaded to various resources more flexibly and interchangeably.

The CyberGIS-Jupyter framework [25] integrates cloud-based Jupyter notebooks with HPC resources to form a hybrid computing environment. CyberGIS-Jupyter uses the centralized JupyterHUB service to handle authentication and scheduling Jupyter servers as containers in different virtual machines. It requires users to write their workflows as Jupyter notebooks (in Python language). In comparison, CyberWater allows users to configure complex workflows all from the intuitive, feature-rich VisTrails front-end. NASA's NEX project [26] allows researchers to design workflows in VisTrails and then launch them to remote HPC resources. In NEX, workflows generated from VisTrails are submitted as jobs to HPC. In comparison, the CyberWater system lets researchers identify computationally expensive tasks, and utilizes the power of HPC/Cloud systems only for those selected tasks.

VI. Conclusion and future work

This work introduces the efforts to extend the CyberWater software framework with on-demand HPC/Cloud access. To this end, we design and implement LaunchAgent in CyberWater, which utilizes either a SciGaP channel or a direct Slurm-based channel to offload computational/data-intensive tasks to different computing environments such as campus-based clusters and XSEDE/Grid computing systems. The CyberWater system extended with LaunchAgent not only allows the open-data and open-modeling framework to continue using graphical-based workflows with rich GUI-based interactions, but also enables on-demand access to HPC resources. By selectively offloading computationally and data expensive tasks rather than entire workflows, CyberWater provides a more scalable and effective way to use HPC/Cloud resources. We have demonstrated the expressiveness of our design through a VIC5 based workflow constructed from CyberWater. Our experiments show that the new HPC/Cloud enabled CyberWater allows users to launch expensive computing tasks to remote resources conveniently, gaining significant speed. We are currently experimenting with more complex large-scale tasks and adding support to various computationally expensive models. We also plan to research the scheduling perspectives of on-demand HPC/Cloud usage, intelligently scheduling tasks and utilization across multiple types of resources.

Acknowledgments

This work was supported by the U.S. National Science Foundation under [OAC-1835817] and [OAC-1835785] to Indiana University-Purdue University Indianapolis (IUPUI)/Indiana University (IU), and to University of Pittsburgh, respectively. This work also used HPC and Cloud

[11] https://www.cuahsi.org/projects/cyberwater/

[12] https://www.knime.com/sites/default/files/inline-images/KNIME_cluster-executor_productsheet_web.pdf

resources provided by the Extreme Science and Engineering Discovery Environment (XSEDE) under [TG-EAR200001], and XSEDE is supported by NSF grant ACI-1548562.

REFERENCES

[1] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. Pegasus: Mapping scientific workflows onto the grid. In *European Across Grids Conference*, pages 11–20. Springer, 2004.

[2] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: the Condor experience. *Concurrency and computation: practice and experience*, 17(2-4):323–356, 2005.

[3] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747, 2006.

[4] Daniel Luna, Ranran Chen, Cao Yuan, et al. Cyberwater—an open and sustainable framework for diverse data and model integration. In *American Geophysical Union Fall Meeting (AGU Fall Meeting 2019)*, San Francisco, CA, Dec. 9-13 2019.

[5] Daniel Salas, Xu Liang, Miguel Navarro, Yao Liang, and Daniel Luna. An open-data open-model framework for hydrological models' integration, evaluation and application. *Environmental Modelling & Software*, 126:104622, 2020.

[6] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

[7] Suresh Marru, Lahiru Gunathilake, Chathura Herath, et al. Apache airavata: a framework for distributed applications and computational workflows. In *Proceedings of the 2011 ACM workshop on Gateway computing environments*, pages 21–28, 2011.

[8] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, and Nancy Wilkins-Diehr. XSEDE: Accelerating Scientific Discovery. *Computing in Science Engineering*, 16(5):62–74, September 2014.

[9] Xu Liang, Dennis P Lettenmaier, Eric F Wood, and Stephen J Burges. A simple hydrologically based model of land surface water and energy fluxes for general circulation models. *Journal of Geophysical Research: Atmospheres*, 99(D7):14415–14428, 1994.

[10] Marlon Pierce, Suresh Marru, Eroma Abeysinghe, Sudhakar Pamidighantam, Marcus Christie, and Dimuthu Wannipurage. Supporting science gateways using apache airavata and scigap services. In *Proceedings of the Practice and Experience on Advanced Research Computing*, pages 1–4. 2018.

[11] Joseph J Hamman, Bart Nijssen, Theodore J Bohn, Diana R Gergel, and Yixin Mao. The variable infiltration capacity model version 5 (vic-5): Infrastructure improvements for new applications and reproducibility. *Geoscientific Model Development*, 11(8):3481–3496, 2018.

[12] Indiana University. Supercomputers for academic research at IU, 2021. https://kb.iu.edu/d/alde.

[13] Texas Advanced Computing Center. Stampede2 HPC system, 2021. https://www.tacc.utexas.edu/systems/stampede2.

[14] Pittsburgh Supercomputing Center. The Bridges-2 HPC System, 2021. https://www.psc.edu/resources/bridges-2.

[15] Craig A Stewart, Timothy M Cockerill, Ian Foster, et al. Jetstream: a self-provisioned, scalable science and engineering cloud environment. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, pages 1–8, 2015.

[16] The NASA Goddard Earth Sciences Data and Information Services Center (GES DISC). NLDAS-2 Model Data Description/Information, 2021. https://ldas.gsfc.nasa.gov/nldas/v2/models.

[17] Sujay V Kumar, Michael Jasinski, David M Mocko, et al. Nca-ldas land analysis: Development and performance of a multisensor, multivariate land data assimilation system for the national climate assessment. *Journal of Hydrometeorology*, 20(8):1571–1593, 2019.

[18] Indiana University. About FutureWater, 2021. https://futurewater.indiana.edu/about/index.html.

[19] Venkatesh Merwade, Wei Feng, Lan Zhao, and Carol X Song. Waterhub: a resource for students and educators for learning hydrology. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond*, pages 1–4, 2012.

[20] Enric Tejedor, Yolanda Becerra, Guillem Alomar, Anna Queralt, Rosa M Badia, Jordi Torres, Toni Cortes, and Jesús Labarta. Pycompss: Parallel computational workflows in python. *The International Journal of High Performance Computing Applications*, 31(1):66–82, 2017.

[21] Yadu Babuji, Anna Woodard, Zhuozhao Li, et al. Parsl: Pervasive parallel programming in python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 25–36, 2019.

[22] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. KNIME - the Konstanz information miner: Version 2.0 and beyond. *ACM SIGKDD Explorations Newsletter*, 11(1):26–31, November 2009.

[23] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R. Pocock, Peter Li, and Tom Oinn. Taverna: A tool for building and running workflows of services. *Nucleic Acids Research*, 34(suppl_2):W729–W732, July 2006.

[24] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall, Alex Hardisty, Abraham Nieva de la Hidalga, Maria P. Balcazar Vargas, Shoaib Sufi, and Carole Goble. The Taverna workflow suite: Designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(W1):W557–W561, July 2013.

[25] Dandong Yin, Yan Liu, Hao Hu, Jeff Terstriep, Xingchen Hong, Anand Padmanabhan, and Shaowen Wang. Cybergis-jupyter for reproducible and scalable geospatial analytics. *Concurrency and Computation: Practice and Experience*, 31(11):e5040, 2019.

[26] Jia Zhang, Petr Votava, Tsengdar J Lee, Owen Chu, Clyde Li, David Liu, Kate Liu, Norman Xin, and Ramakrishna Nemani. Bridging vistrails scientific workflow management system to high performance computing. In *2013 IEEE Ninth World Congress on Services*, pages 29–36. IEEE, 2013.