Empowering Agroecosystem Modeling with HTC Scientific Workflows: The Cycles Model Use Case

Rafael Ferreira da Silva*, Rajiv Mayani*, Yuning Shi†, Armen R. Kemanian†, Mats Rynge*, Ewa Deelman*
*Information Sciences Institute, University of Southern California, Marina Del Rey, CA, USA
†Department of Plant Science, The Pennsylvania State University, University Park, PA, USA
{rafsilva,mayani,rynge,deelman}@isi.edu, {yshi,kxa15}@psu.edu

***Abstract*—Scientific workflows have enabled large-scale scientific computations and data analysis, and lowered the entry barrier for performing computations in distributed heterogeneous platforms (e.g., HTC and HPC). In spite of impressive achievements to date, large-scale modeling, simulation, and data analytics in the long-tail still face several challenges such as efficient scheduling and execution of large-scale workflows (O(10^6)) with very short-running tasks (few seconds). While the current trend to support next-generation workflows on leadership class machines have gained much attention in the past years, at the other end of the spectrum scientific workflows from the long-tail science have become larger and require processing massive volumes of data. In this paper, we report on our experience in designing and implementing an HTC workflow for agroecosystem modeling. We leverage well-known (task clustering and co-scheduling) and emerging (hierarchical workflows and containers) workflow optimization techniques to make the workflow planning problem tractable, and maximize resource utilization and the degree of task parallelism. Experimental results, via the implementation of a use case, show that by strategically combining the above strategies and defining an appropriate set of optimization parameters, the overall workflow makespan can be improved by 3.5 orders of magnitude when compared to a regular (non-optimized) execution of the workflow.**

***Index Terms*—Scientific Workflows, High-Throughput Computing, Agroecosystem Modeling.**

I. Introduction

Scientific workflows are mainstream for enabling computational science progress in several scientific domains [1], [2]. Notable contributions include the first detection of gravitational waves from colliding black holes [3], wildfire resilience [4], among others. In recent years, there is a shift toward developing systems that can support workflow applications that consume and produce large amounts of data or that require high-performance computing, a situation commonly termed as extreme-scale computing [5]. To address such demand, solutions such as in transit (e.g., burst buffers [6], [7]) and in situ processing [8] have emerged as centerpiece technologies for supporting the next-generation of workflow applications. Although supporting this new class of next-generation complex applications is key for unleashing breakthrough discoveries, computational science in the long tail still imposes challenges when, for example, addressing complex research questions [9], [10] such as: *How the average crop production would respond to different levels of soil fertilization? Which fraction of weed would harm crop production for different crops on distinct regions? How do fertilizer level and weed pressure interact across regions, years or planting dates within a growing season?* The number of questions, and the demands on a modeling system, can grow exponentially.

Advances in scientific computing in the long tail has produced technologies to automate the execution of scientific code and data processing on distributed resources. In the context of scientific workflows, software elements targeted robust and efficient execution of computing jobs on campus clusters, grid and volunteer computing, and more recently clouds [1]. Despite the fact that most scientific domains are increasingly producing/consuming large, heterogeneous datasets, only a small number of them in the long tail has developed or amended software for parallel processing – this is mostly due to the complexity inherent to modeling variables, equations, iterations, etc. Therefore, with the advent of increasing heterogeneity, volumes of data, and data science and machine learning techniques, there is still a need for solutions to address these needs for scientific computing in the long tail.

In this paper, we report on our experience in designing and implementing a high-throughput scientific workflow for agroecosystems modeling. More specifically, we present the challenges to handle the massive number of embarrassingly parallel computing jobs (O(10^6)), and how we leverage a set of well-known (job clustering and co-scheduling) and emerging (hierarchical workflows and containers) technologies for addressing such challenges. We implement the workflow with the state-of-the-art Pegasus workflow management system (WMS) [3]; we execute and characterize the workflow profile on a local cluster with different optimization techniques; and we draw conclusions on performance characteristics related to the workflow structure. To make the planning and scheduling problem tractable, we partition the workflow DAG into sub-workflows. Experimental results show that by applying task clustering, the overall workflow makespan is improved by about three orders of magnitude when compared to a standard non-optimized execution; and by allowing task-resource co-allocation, such improvement is augmented up to 3.5 orders of magnitude.

This paper is organized as follows. Section II presents an overview of scientific workflows in the long tail, and a brief description of the state-of-the-art Pegasus WMS. Section III introduces the Cycles simulation software, and presents the designed abstract workflow. Section IV presents our use case implementation and experimental results. Finally, Section V concludes with a brief summary of results and a discussion of future research directions.

II. Background and Related Work

A. Scientific Workflows in the Long Tail

Today's computational and data science applications process vast amounts of data (from remote sensors, instruments, etc.) for conducting large-scale simulations of underlying science phenomena. These applications may comprise millions of computational tasks and process large datasets, which are often distributed and stored on heterogeneous resources. Scientific workflows have emerged as a flexible representation to declaratively express complex such applications with data and control dependencies, and have become mainstream in domains such as astronomy, physics, environmental sciences, biology, and others [1], [2], [11], [12]. With the advent of complex modeling and increased volumes of data, some scientific domains pursued the development of novel code that required high-performance computing (HPC) platforms. For instance, HPC-enabled workflows have supported hydrology [13], [14], bioinformatics [15], among others. At the long tail, scientific workflows have enabled the execution of high-throughput computing (HTC) applications on grids and clouds — e.g., the Open Science Grid (OSG) [16] project and more recently the upcoming NSF-funded cyberinfrastucture HPC systems (Bridges-2 and Expanse) that will also provide specialized support for HTC applications [17].

Along these past two decades, several workflow management systems have developed techniques to optimize workflow executions for the long tail [1], [2], [12], [18]. For instance, task clustering has been used for overcoming system overheads when running very short-running tasks [19]. HPC-based solutions were developed for wrapping entire workflows of embarrassingly parallel tasks as MPI jobs [3], [20]. Container support has been enabled for fostering reproducible and self-contained execution environments [21], [22]. Nevertheless, most of these works focus on a single or a couple of optimizations for orchestrating and executing workflows. In this paper, we review some of these techniques and combine them with more advanced and recent approaches for enacting efficient execution of a large-scale agroecosystem modeling workflow, while improving resource utilization.

B. Pegasus WMS

Pegasus is being used in production to execute workflows for dozens of high-profile applications in a wide range of scientific domains [3]. Pegasus provides the necessary abstractions for scientists to create workflows and allows for transparent execution of these workflows on computer clusters, cloud services, national cyberinfrastructures, and other platforms. During execution, Pegasus translates an abstract resource-independent workflow into an executable workflow, determining the specific executables, data, and computational resources required for the execution. Workflow execution with Pegasus includes data management, monitoring, and failure handling, and is managed by HTCondor DAGMan. Individual

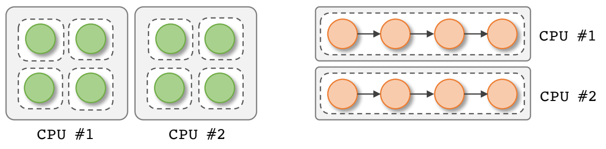

Fig. 1. Example of task co-allocation (*left*)) and clustering (*right*). When co-allocating tasks, a single CPU may run multiple tasks concurrently; while in clustering, tasks are typically executed sequentially in a pipeline.

workflow tasks are managed by a workload management framework, HTCondor [23], which supervises task executions on local and remote resources.

During the past two decades, the Pegasus software co-evolved alongside ever growing scientific needs and applications and novel technologies [24]. In order to support the development of the Agroecosystem modeling workflow (Section III), and more specifically to support the execution of the large-scale workflow execution, described in Section IV, we have leveraged well-known Pegasus features as well as more recently released capabilities as follows:

Task-resource co-allocation. Resource co-allocation enables concurrent execution of multiple computational tasks within a single CPU core (Fig. 1-*left*) [25]. Co-allocation is typically implemented for maximizing resource utilization while increasing parallelism. However, it should be used sparingly since an excessive degree of parallelism may lead to significant slowdown of the application execution.

Task clustering. Grouping sets of tasks reduces execution overhead and increases the computational granularity of scientific workflow tasks executing on distributed resources [19]. Traditionally, tasks are clustered together based on reasoning regarding, for example, the workflow structure, tasks characteristics, or user-based labelling. Clustered tasks are typically mapped into a pipeline and executed sequentially (Fig. 1-*right*) – although parallel execution can also be performed if running in a multicore node. Notice that the dependency between tasks in a pipeline may or not involve data dependency. We leverage task co-allocation and clustering for diminishing scheduling overheads, and therefore improving workflow makespan.

Hierarchical workflows. Workflows are in most cases described as static DAGs, where the nodes represent computing tasks and the edges dependencies between tasks (either data or control dependencies). With the advent of extreme-scale computing, workflows became ever larger (O(10^6) tasks), which led the workflow planning process (creation and scheduling) intractable. Hierarchical workflows emerged then as a solution for mitigating this issue by allowing nodes of the DAG to represent entire sub-workflows – in which a hierarchy of sub-workflows can be defined (Fig. 3). This way, such large-scale workflows can be partitioned into moderate-scale, tractable sub-workflows. An alternative use of hierarchical workflows is for enabling conditions and late binding. By capitalizing on the ability to define sub-workflows as nodes in the DAG, planning only occurs once the sub-workflow node will be executed, i.e.

once all its dependencies have been satisfied. In this paper, we leverage late binding and sub-workflow partitioning, thus enabling the use case presented in Section IV.

Containers. Container technologies have become keystone for fostering reproducibility in scientific computing. They provide a flexible, custom, user-controlled environment for executing applications seamlessly – containers provide self-contained execution environments where all dependencies of the software stack have been resolved. We leverage the recently released Pegasus' approach [21] on efficiently managing automatic container deployment to mitigate application execution errors due to misconfigurations or missing dependencies.

By combining the four key features above, planning and executing high-throughput large-scale workflows become practicable, while yielding better resource utilization and improved efficiency, thereby improved workflow turnaround time.

III. The Cycles High-Throughput Workflow

In this section, we present a brief description of the Cycles simulation model and its main capabilities. We also present the abstract (i.e., platform and scenario agnostic) implementation of the workflow to support large-scale simulation executions.

A. *Cycles Simulation Model*

Cycles is a user-friendly, multi-crop, multi-year, process-based Agroecosystem model with daily time step simulations of crop production and the water, carbon (C) and nitrogen (N) cycles in the soil-plant-atmosphere continuum. The model is an evolution of C-FARM [26] and is closely related to CropSyst [27]. The hydrology is simulated with an adaptive sub-daily time step. The algorithms of heat and water transport were adapted from [28]. The reference evapotranspiration is calculated using the Penman-Monteith equation as formulated in [29]. Daily plant growth is based either on the radiation capture (light limited) and or on the realized transpiration (water limited), an approach that surrogates for a coupled transpiration and photosynthesis model [30]. In Cycles, the stomatal conductance is determined by temperature and the leaf water potential, with the latter depending on the balance of the transpirational demand, the soil water supply and plant hydraulic properties [31], [32]. Crop development is calculated using thermal time, and grain yield is calculated using the biomass accrued and the harvest index [33]. Soil organic C and N cycling is based on saturation theory [34]. The minimum inputs to the model are: latitude, daily weather (minimum and maximum temperature and relative humidity, precipitation, solar radiation, and wind speed), soil description (layer thickness, clay, sand and organic matter content), cropping sequence, and management information.

The model can simulate perturbations of biogeochemical processes caused by agronomic practices such as tillage, irrigation, organic and inorganic nutrient applications, annual and perennial crops selection, grain and forage harvest, polycultures, relay cropping and grazing. Cycles allows unlimited crop species to be specified by the user. The current model set up has a spin up feature to allow a quasi-equilibrium in soil properties that effectively establish a baseline. The model is run repeatedly using the same climate until total soil organic carbon in the soil profile changes by less than 0.1% before and after a simulation cycle. All soil properties are stored and are used as initial condition for the remainder of the user-defined simulation. The user needs to decide which input files must be used to establish a baseline.

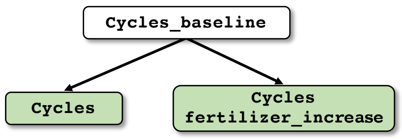

Fig. 2. Overview of a single configuration definition of a Cycles abstract (i.e., platform and scenario agnostic) workflow. The model output is evaluated against a reference or baseline.

B. *Cycles Abstract Workflow*

With the Cycles simulation model, a user can explore a domain that is much larger than what can be explored experimentally. To this end, the model needs to be used with a range of inputs compatible with the model calibration domain, which may limit model accuracy (e.g., actual soil descriptions, particularly in non-agricultural soils). The proposed Cycles workflow aims to aid researchers to answer questions such as: *What is the fraction of the years or fraction of an agricultural area in which crops establishment fails? What is the expected response to inputs (nitrogen fertilizer and other inputs) or long term effect of rotations?*. The workflow can also be used to estimate carbon dioxide, nitrous oxide emissions/uptake in agriculture systems, both of which are key component of the global warming potential of agroecosystems. For these purposes, we use a sequential approach, in which the model output is evaluated against a reference or baseline (`Cycles_baseline`) that is also obtained with the model and is based on realistic assumptions.

Fig. 2 shows a representation of the Pegasus Cycles abstract workflow for a single configuration run (e.g., one point of the grid cell, a single crop, and a particular set of unique parameters or system properties). Once the reference or baseline task is completed, the actual `Cycles` simulation is launched. We also perform an additional execution of Cycles with increased fertilizer rate of 10% (`Cycles_fertilizer_increase`) – such output is key to forecasting the economic impact of grain yields.

In order to answer research questions such as the ones illustrated above, one would have to execute Cycles simulation model for a range of system properties (e.g., nitrogen fertilization rates, weed fractions) in different locations (may involve several points of the spatial grid), and for different crops. Such set of executions, a.k.a. ensembles, may quickly scale up to hundreds of thousands of parameters combinations, and therefore up to O(10^6) computational tasks in a workflow. As discussed in Section II-B, static planning and scheduling such large-scale workflows is a cumbersome process and to

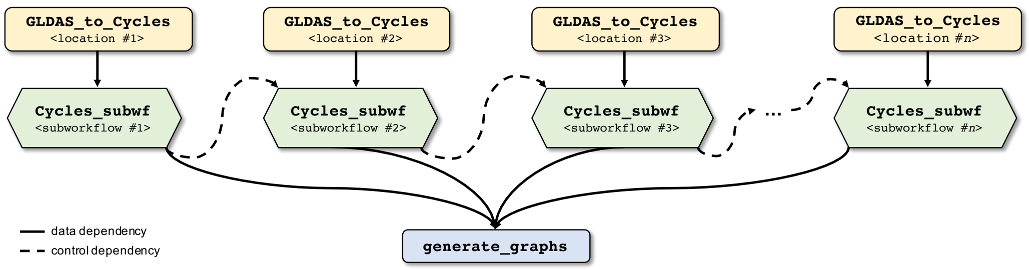

Fig. 3. Overview of the "top level" Cycles workflow structure. Weather transformation jobs (`GLDAS_to_Cycles`) are independent tasks and can run in parallel. Sub-workflows (`Cycles_subwf`) executions are defined by control dependencies – only a single sub-workflow execution is performed at a time. Each sub-workflow does Cycles simulations for individual points of the spatial grid. A `generate_graphs` tasks ensembles all summaries and produce visualization outputs.

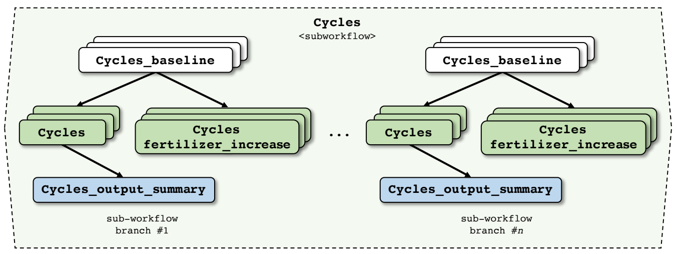

Fig. 4. Overview of a Cycles "sub-workflow" structure. The sub-workflow is composed of several branches, which are defined by the number of different crops. Each branch performs agroecosystem simulations for individual crops taking into consideration all combinations of the simulation matrix.

some extent an intractable problem. To address this challenge, we capitalize on Pegasus' hierarchical workflow feature to partition ensembles into sub-workflows to mitigate the planning and scheduling problem. Fig. 3 shows the "top level" structure of the Cycles abstract workflow. A sub-workflow (`Cycles_subwf`) represents an ensemble for a determined location (grid cell). To limit the number of concurrent executions and make scheduling more efficient, we define control dependencies between sub-workflows – a sub-workflow will only be planned and executed once the parent sub-workflow execution is completed.

Each Cycles simulation needs weather data for inferring weather conditions such as rain, air temperature, wind, etc. The `GLDAS_to_Cycles` task receives as input a set of GLDAS (Global Land Data Assimilation System) data products [35], and parses them into the Cycles required format for weather information. Although the number of locations is relatively small when compared to the number of parameters combinations, the input data size is rather large (O(10^2)GB). Thus, such tasks are set to run concurrently.

Fig. 4 shows the structure of a `Cycles_subwf`. The sub-workflow represents an ensemble of Cycles simulation runs for different crops (defined as "branches"). Each branch is composed of possibly several (O(10^3)) executions of the Cycles single configuration workflow (Fig. 2). In order to generate relevant and concise data products, e.g. for visualization, the `Cycles_output_summary` task aggregates and summarizes all outputs for a single crop.

IV. Use Case: Understanding Major Agricultural Outputs in South Sudan

Evaluating agricultural production in most regions of the world is still a challenging problem: (*i*) agricultural systems tend to be detail-rich, leading expert and non-expert users to focus in the noise rather than on the broad aspect of the problem; (*ii*) from the simulation perspective, specific information about management practices and actual output (e.g., yield, production areas) could be missing or incomplete to derive significant results (e.g., South Sudan), forcing analysts and modelers to either build simulations on too many assumptions or run large numbers of simulations that then become laborious to interpret. In short, (*iii*) simulating agricultural systems in ways that produce meaningful outputs requires quickly merging a biophysical understanding of different systems and landscapes with the human realities of a given location.

A. *Cycles Simulation Matrix*

To address the above issues, we designed an approach for generating a matrix of simulations that, using expert judgement, brackets the management practices in a given region. For example, one may not know the specifics about nitrogen fertilizer management in a region, but nothing prevents a user from running simulations that include a wide range of fertilizer rates that surely include the range of interest. This concept is generalized to other system properties including irrigation, weed control, crop species, and rotations. The more one knows, the more restricted the range, but limited information should not be an impediment for exploration.

Table I shows the simulation matrix for Cycles system properties used in this case study. The start planting date variable levels are in days of the year (January 1st = 1); planting dates are spaced out 7 days until the end of the planting date. The planting date can be fixed (i.e., in a given day) or conditional to soil moisture and temperature. Nitrogen fertilization rates are in kg/ha of nitrogen. The weed fraction is an abstraction to represent low to high weed pressure. The weeds are planted a week after the crop planting and left uncontrolled. Two weeds are planted, so when the weed fraction for a weed species is 0.4 and two different species are planted, there is a population of weeds equivalent to that of the crop. More severe weed pressures can be simulated.

TABLE I
CYCLES SIMULATION MATRIX.

Parameter	Values
Country	`South Sudan`
Crop	`Maize, Sorghum, Sesame, Peanut`
Start planting date	`100, 107, 114, 121, 128, 135, 142`
End planting date	`149`
Planting date fixed	`True, False`
Nitrogen rate	`0, 25, 50, 100, 200, 400`
Weed fraction	`0.0, 0.05, 0.1, 0.2, 0.4, 1.5, 2.0`

We consider a Cycles workflow instance composed of 2,352 properties combinations (cross product of all properties values) for a single point in a spatial grid, which resulted in 7,056 Cycles executions per sub-workflow (Fig. 4). In this paper, we consider GLDAS grids within South Sudan at 0.5 degree resolution, which resulted in 209 locations. As a result, the Cycles use case workflow considered here is composed of 209 sub-workflows. In total, the particular workflow instance studied here is composed of 1,475,122 computational tasks, which includes weather transformation and output summary tasks. Notice that given that peanut (a.k.a. groundnut) can obtain nitrogen from the atmosphere via biological fixation, we do not do Cycles simulations to obtain a response to nitrogen fertilization.

B. *Workflow Execution Profile*

The analysis presented in this work is based on the execution of an instance of the Cycles workflow derived from the simulation matrix shown in Table I. Workflow executions are performed on a local server, which is equipped with two 2.2GHz AMD EPYC 7601 32-Core Processors, 296GB of RAM, and standard magnetic hard drives in RAID configuration. Due to the large number of tasks involved in this instance of the workflow execution (O(10^6)), we must partition the workflow into sub-workflows to make the planning problem tractable. For instance, planning all task of the workflow would require O(10^2) GB of RAM and several hours to map the abstract workflow into an executable workflow. The workflow generator code including the simulation matrix generation are available online[1].

Table II shows the execution profile of Cycles workflow tasks, with one row per task type. These profiles were obtained from workflow executions with no performance optimizations, i.e., a single task was allocated to a single core. This way, we ensure task runtime and CPU utilization are not impacted by concurrent executions. The `GLDAS_to_Cycles` tasks reads over 6,000 weather files (∼1TB) and generates files in a format compatible with the Cycles model input requirements. These 209 tasks run over 6 h in our server and are an integral part of the DAG critical path. Cycles simulation model tasks (`Cycles_baseline`, `Cycles`, and `Cycles_fertilizer_increase`) run fast (under 10s). However, due to the massive number of tasks (derived from

[1]https://github.com/pegasus-isi/pegasus-cycles

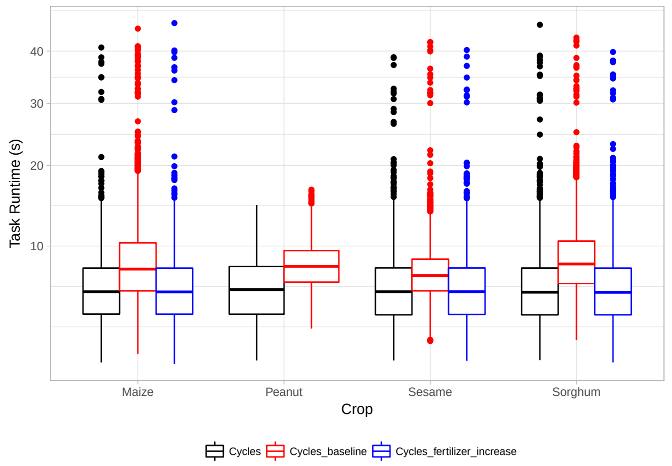

Fig. 5. Distribution of Cycles simulation task runtimes (in seconds) for the Cycles workflow. Simulations with increased fertilizer (nitrogen) are not performed for Peanut.

the simulation matrix shown in Table I), these tasks represent the bottleneck of the execution critical path. The `Cycles_output_summary` tasks are very short-running tasks, yet mostly data-intensive (perform over 4GB of I/O read operations).

Fig. 5 shows the distribution of Cycles simulation task runtimes for different crops. Although average tasks runtime and standard deviations are relatively similar (Table II), different configurations (crops, fertilizer rate, weed pressure, and others) cause different behaviors in execution, e.g. `Cycles_baseline` tasks, and whisker and outlier values (in particular for `Peanut`). Such small task runtime differences can become important during scheduling, and more specifically when considering system and workflow system overheads [36]. With the advent of using containers for fostering reproducible and self-contained execution environments, a new class of system overhead has been introduced to task executions. Building, distributing, and loading containers may lead to high overhead when running large-scale workflows [22]. Pegasus support for containers minimizes this effect (as discussed below), however for the Cycles simulation model tasks shown in Table II, container overhead may represent up to 22% of the task execution time (container overhead measurements vary between 0.8s and 1.4s for these runs). Despite the fact that this overhead interval is substantially small, when orchestrating a large number of very short-running workflow tasks this may become a significant factor of the workflow performance bottleneck. Overall, the workflow makespan (a.k.a. turnaround time) for this profiling run is ∼311h, being 2.1% performing weather transformations, 97.8% performing computations with the Cycles simulation model, and 0.1% summarizing and preparing data for visualization.

C. *Optimized Workflow Execution*

Orchestrating workflow executions from the long tail typically entails efficiently handling very large numbers of short-

TABLE II
EXECUTION PROFILES OF THE CYCLES WORKFLOW JOBS. RUNTIMES ARE SHOWN IN SECONDS, AND I/O OPERATIONS IN MB. (μ IS THE MEAN, AND σ THE STANDARD DEVIATION.)

Task	Count	Runtime		CPU Utilization		I/O Read		I/O Write	
		μ	σ	μ	σ	μ	σ	μ	σ
GLDAS_to_Cycles	209	22,486.3	142.7	84.1%	1.2%	1,001,690	0.0	0.5	0.0
Cycles_baseline	614,460	8.2	3.1	75.2%	1.4%	0.5	0.0	21.4	0.1
Cycles_fertilizer_increase	614,460	6.1	2.3	65.6%	1.8%	0.5	0.1	21.6	0.1
Cycles	614,460	6.2	2.4	59.7%	2.1%	0.5	0.1	21.6	0.1
Cycles_output_summary	1,045	3.6	1.1	72.3%	2.4%	4,514.4	0.1	2.7	0.0
generate_graphs	1	20.3	–	81.2%	–	2,821.5	–	11.3	–

running embarrassingly parallel tasks. As mentioned before, computing and workflow systems have inherent overheads that may harm the workflow execution (e.g., planning and queuing times, cleanup tasks, container spin up, etc.) [36]. More specifically, the overhead for scheduling and running Cycles simulation model tasks is significantly higher than the length of the task execution itself. This does not only incur in increased workflow makespan, but also in low resource utilization, and thereby low task throughput.

Task clustering. Given the above, we leverage task clustering to mitigate the overhead impact on the workflow execution. We also leverage Pegasus container support [21] for optimizing clustered jobs (sets of workflow clustered tasks) execution to overcome the container spin up and system overheads. In Pegasus, a set of clustered jobs share the same container during its execution, given that all tasks within the clustered job are of the same type. By using a single container for running a batch of grouped tasks (executed as a pipeline), the container spin-up overhead becomes negligible when compared to total cluster runtime (the larger the cluster, the smaller the overhead). In contrast, sharing containers instance requires additional effort for handling the execution directory (working directory) structure, e.g. files and folders names may be conflicted, etc.

For the optimized experimental executions presented in this Section, we limited clustering for Cycles simulation model tasks only. We arbitrarily define cluster sizes up to 10, 15, 20, 25, and 50 tasks, and compare the impact of different cluster sizes on overhead mitigation and ultimately workflow turnaround time. Table III shows the average task runtime for the clustered Cycles simulation executions. As expected, average clustered job runtime increases as the number tasks forming a cluster is higher. By using task clustering, system overheads become negligible and offers workflow makespan improvements up to three orders of magnitude, i.e. speedup of 3 (shown in Table IV). For larger cluster sizes ($\geq$20, i.e. coarse-grained clusters), workflow makespan has plateaued – mainly due to low degree of parallelism.

A more recent approach for improving container support in HTC is to use the CVMFS filesystem (CernVM File System) [37], an HTTP-based file distribution service, for staging in the container images once to the server, and distributing it to compute nodes. CVMFS is currently used by the Open Science Grid (OSG) project, and has demonstrated improved I/O throughput by constantly avoiding pulling the container image from an external endpoint (e.g., docker or singularity hubs) for every task execution, thus container overhead spin up is significantly reduced. On the other hand, by running on a container from a distributed filesystem incurs network delays. For the experiments conducted in this paper, we observed a critical slowdown (up to a factor of two orders of magnitude) when performing data integrity checks due to network delays when running OpenSSL. Such slowdown goes unnoticed for long-running tasks, thus the improved performance on OSG systems. Therefore, we claim that when designing a solution for running large-scale workflows with short-running tasks, network latency should be carefully considered.

Increased task and sub-workflow parallelism. Although system overheads are attenuated via task clustering, too-coarse clustered jobs may lead to low degree of parallelism, and thereby low resource utilization. To overcome this low performance while at the same time reducing the overall workflow turnaround time, we leverage task-resource co-allocation. Since Cycles simulation model tasks do not fully utilize the CPU capacity (see Table II) and since low degree of parallelism is observed for cluster sizes larger than 20, we configured the workload resource manager (HTCondor [23]) to allow task-resource co-allocation. We set the maximum allowed co-allocation to 2. We have also modified the workflow structure (Fig. 3) to permit up to four sub-workflows to run concurrently (depending on the degree of task parallelism). Table III also shows average runtime for Cycles simulation clustered jobs running with co-allocation. As expected, runtimes are slightly degraded due to increased concurrency on the CPU usage – recall that CPU utilization for Cycles simulation model tasks is above 50%. On the other hand, overall workflow makespan is improved by up to 3.5 orders of magnitude when compared to the non-optimized execution of the workflow, and up to an order of magnitude for smaller cluster sizes when compared to optimized executions, as shown in Table IV. Notice that due to increased degree of parallelism, the plateaued speed up for larger cluster sizes moderately increases.

D. *Output Products and Visualization*

Fig. 6 shows simulated grain yield response of maize for different fertilization rates and weed pressure levels for a single grid cell (identified as a cropland). The outcome of these simulations allow analysts to identify year to year variations

TABLE III
RUNTIME IN SECONDS FOR CYCLES SIMULATION CLUSTERED JOBS RUNNING ON A SINGLE CORE OR CO-ALLOCATED. CLUSTER SIZES RANGE BETWEEN 10 AND 50 TASKS.

Method	Cycles_baseline					Cycles_fertilizer_increase					Cycles				
	10	15	20	25	50	10	15	20	25	50	10	15	20	25	50
1 task per core	93.7 (±10.3)	138.6 (±18.2)	186.2 (±22.0)	219.8 (±22.4)	391.6 (±52.5)	61.9 (±5.6)	93.5 (±10.0)	124.4 (±14.3)	161.9 (±16.4)	355.2 (±72.6)	62.5 (±6.1)	93.6 (±8.3)	125.8 (±14.4)	163.8 (±14.9)	364.1 (±70.7)
Co-allocation	99.2 (±10.1)	141.7 (±14.3)	196.9 (±19.2)	242.3 (±30.2)	557.9 (±84.3)	64.3 (±7.3)	100.0 (±7.9)	129.9 (±19.9)	163.3 (±11.5)	360.1 (±46.5)	64.5 (±7.4)	100.3 (±10.1)	136.4 (±13.4)	165.4 (±14.5)	371.9 (±49.4)

TABLE IV
WORKFLOW MAKESPAN IN HOURS FOR EXECUTION ON A SINGLE CORE OR CO-ALLOCATED. CLUSTER SIZES RANGE BETWEEN 10 AND 50 TASKS.

Method	No clustering	Cluster Sizes				
		10	15	20	25	50
1 task per core	311.1	181.2	128.7	105.2	105.1	104.9
Co-allocation	237.4	128.9	114.1	99.2	93.8	86.6

in, for example, grain yield, but many other variables of production or environmental quality importance are available. By comparing the simulated grain yield production from 2017 to previous years, it is immediately clear that in that year there was a steep fall in grain yield and a weak response to fertilizer. Therefore, one can conclude that after several years of variable but never catastrophically low yields, a poor harvest can surprise local stakeholders and humanitarian agencies. Tactical adjustments like switching crops and obtaining new seeds are not instantaneous and requires months of preparation; so does aid distribution. Two years in a row of unfavorable weather and poor crop yields would likely turn into a disaster.

The modeled response to nitrogen fertilizer rates is substantial and requires three qualifications. First, it depends on weed pressure. Adding fertilizer without the ability of controlling weeds can lead to severe problems. Weed control can be limited by availability of labor, equipment or suitable agrochemicals, and by the weather. Many producers in areas of subsistence agriculture mix crops in small plots or even in the same row, which also complicates chemical weed control. Second, soil conditions can vary. Areas with shallower soils can have less soil water storage, a much subdued response to fertilizer and lower average yields. Third, even in this relatively narrow geographical area, there are clear variations among locations. Thus, while a smooth workflow can facilitate modeling and exploration, visualization procedures should aid the analyst understand the power and limitations of the outputs in the proper context.

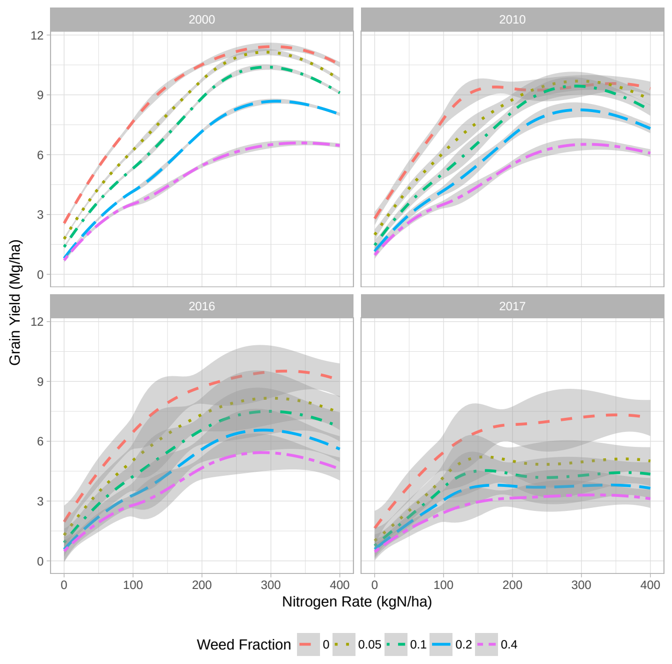

Fig. 6. Maize grain yield response to nitrogen fertilizer rate and weed pressure for a single location in four years.

V. CONCLUSION

In this paper, we have reported on our experience in designing and implementing a large-scale high-throughput workflow for agroecosystem modeling. By leveraging well-known optimization techniques and emerging technologies for workflow design and execution, we have made the workflow planning and scheduling problem tractable while improving resource utilization, and thereby workflow turnaround time. Our experimental use case, comprised of over 1.4M tasks, showed that by carefully strategizing and parameterizing the optimization methods, the overall workflow makespan could be improved by 3.5 orders of magnitude.

A short-term development direction is to perform a performance gain comparison between the presented approach and HPC-based solutions such as Pegasus PMC [3]. Another future work goal is to extend the synthetic workflow generator in [38], which produces realistic synthetic workflow configurations based on profiles extracted from workflow execution traces, to generate workflow traces based on the Cycles workflow. Such traces are key for supporting the development of new algorithms and solutions in workflow research [39].

ACKNOWLEDGMENTS

This work was funded by the Defense Advanced Research Projects Agency under award #W911NF-18-1-0027; and partly funded by the National Science Foundation under grant #1664162. We thank Deborah Khider and Karan Vahi for their valuable help.

References

[1] C. S. Liew, M. P. Atkinson, M. Galea, T. F. Ang, P. Martin, and J. I. V. Hemert, "Scientific workflows: Moving across paradigms," *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, p. 66, 2017.

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

[3] 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.

[4] D. Crawl, J. Block, K. Lin, and I. Altintas, "Firemap: A dynamic data-driven predictive wildfire modeling and visualization environment," *Procedia Computer Science*, vol. 108, pp. 2230–2239, 2017.

[5] R. Ferreira da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman, "A characterization of workflow management systems for extreme-scale applications," *Future Generation Computer Systems*, vol. 75, pp. 228–238, 2017.

[6] C. S. Daley, D. Ghoshal, G. K. Lockwood, S. Dosanjh, L. Ramakrishnan, and N. J. Wright, "Performance characterization of scientific workflows for the optimal use of burst buffers," *Future Generation Computer Systems*, 2017.

[7] R. Ferreira da Silva, S. Callaghan, T. M. A. Do, G. Papadimitriou, and E. Deelman, "Measuring the impact of burst buffers on data-intensive scientific workflows," *Future Generation Computer Systems*, vol. 101, pp. 208–220, 2019.

[8] F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki, and H. Abbasi, "Enabling in-situ execution of coupled scientific workflow on multi-core platform," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium*. IEEE, 2012, pp. 1352–1363.

[9] Y. Gil, K. Cobourn, E. Deelman, C. Duffy, R. Ferreira da Silva, A. Kemanian, C. Knoblock, V. Kumar, S. Peckham, L. Carvalho, Y.-Y. Chiang, D. Garijo, D. Khider, A. Khandelwal, M. Pahm, J. Pujara, V. Ratnakar, M. Stoica, and B. Vu, "Mint: Model integration through knowledge-powered data and process composition," in *9th International Congress on Environmental Modelling and Software*, 2018.

[10] R. Ferreira da Silva, D. Garijo, S. Peckham, Y. Gil, E. Deelman, and V. Ratnakar, "Towards model integration via abductive workflow composition and multi-method scalable model execution," in *9th International Congress on Environmental Modelling and Software*, 2018.

[11] R. Ferreira da Silva, E. Deelman, R. Filgueira, K. Vahi, M. Rynge, R. Mayani, and B. Mayer, "Automating environmental computing applications with scientific workflows," in *Environmental Computing Workshop, IEEE 12th International Conference on e-Science*, ser. ECW'16, 2016, pp. 400–406.

[12] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, "A survey of data-intensive scientific workflow management," *Journal of Grid Computing*, vol. 13, no. 4, pp. 457–493, 2015.

[13] L. Leonard and C. J. Duffy, "Automating data-model workflows at a level 12 huc scale: Watershed modeling in a distributed computing environment," *Environmental modelling & software*, vol. 61, pp. 174–190, 2014.

[14] L. Leonard and C. Duffy, "Visualization workflows for level-12 huc scales: Towards an expert system for watershed analysis in a distributed computing environment," *Environmental modelling & software*, vol. 78, pp. 163–178, 2016.

[15] M. Mattoso, J. Dias, K. A. Ocaña, E. Ogasawara, F. Costa, F. Horta, V. Silva, and D. de Oliveira, "Dynamic steering of hpc scientific workflows: A survey," *Future Generation Computer Systems*, vol. 46, pp. 100–113, 2015.

[16] "The Open Science Grid," https://opensciencegrid.org, 2019.

[17] "Upcoming NSF Cyberinfrastructure Projects to Support Long-Tail Users, AI and Big Data," https://www.hpcwire.com/2019/08/05/upcoming-nsf-cyberinfrastructure-projects-to-support-long-tail-users-ai-and-big-data, 2019.

[18] F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: a survey," *The Journal of Supercomputing*, vol. 71, no. 9, pp. 3373–3418, 2015.

[19] W. Chen, R. Ferreira da Silva, E. Deelman, and T. Fahringer, "Dynamic and fault-tolerant clustering for scientific workflows," *IEEE Transactions on Cloud Computing*, vol. 4, no. 1, pp. 49–62, 2016.

[20] J. Ozik, N. T. Collier, J. M. Wozniak, and C. Spagnuolo, "From desktop to large-scale model exploration with swift/t," in *2016 Winter Simulation Conference (WSC)*. IEEE, 2016, pp. 206–220.

[21] K. Vahi, M. Rynge, G. Papadimitriou, D. Brown, R. Mayani, R. Ferreira da Silva, E. Deelman, A. Mandal, E. Lyons, and M. Zink, "Custom execution environments with containers in pegasus-enabled scientific workflows," in *15th eScience Conference*, 2019.

[22] C. Zheng and D. Thain, "Integrating containers into workflows: A case study using makeflow, work queue, and docker," in *Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing*. ACM, 2015, pp. 31–38.

[23] 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.

[24] 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.

[25] K. Czajkowski, I. Foster, and C. Kesselman, "Resource co-allocation in computational grids," in *Proceedings. The Eighth International Symposium on High Performance Distributed Computing (Cat. No. 99TH8469)*. IEEE, 1999, pp. 219–228.

[26] A. R. Kemanian and C. O. Stöckle, "C-farm: A simple model to evaluate the carbon balance of soil profiles," *European Journal of Agronomy*, vol. 32, no. 1, pp. 22–29, 2010.

[27] C. O. Stöckle, A. R. Kemanian, R. L. Nelson, J. C. Adam, R. Sommer, and B. Carlson, "Cropsyst model evolution: From field to regional to global scales and from research to decision support systems," *Environmental modelling & software*, vol. 62, pp. 361–369, 2014.

[28] G. S. Campbell, *Soil physics with BASIC: transport models for soil-plant systems*. Elsevier, 1985, vol. 14.

[29] R. G. Allen, W. O. Pruitt, J. L. Wright, T. A. Howell, F. Ventura, R. Snyder, D. Itenfisu, P. Steduto, J. Berengena, J. B. Yrisarry *et al.*, "A recommendation on standardized surface resistance for hourly calculation of reference eto by the fao56 penman-monteith method," *Agricultural Water Management*, vol. 81, no. 1-2, pp. 1–22, 2006.

[30] C. Kremer, C. O. Stockle, A. R. Kemanian, and T. Howell, "A reference canopy transpiration and photosynthesis model for the evaluation of simple models of crop productivity," *by Agronomy Journal*, 2008.

[31] J. Jara and C. O. Stockle, "Simulation of water uptake in maize, using different levels of process detail," *Agronomy Journal*, vol. 91, no. 2, pp. 256–265, 1999.

[32] G. Camargo and A. R. Kemanian, "Six crop models differ in their simulation of water uptake," *Agricultural and forest meteorology*, vol. 220, pp. 116–129, 2016.

[33] A. R. Kemanian, C. O. Stöckle, D. R. Huggins, and L. M. Viega, "A simple method to estimate harvest index in grain crops," *Field Crops Research*, vol. 103, no. 3, pp. 208–216, 2007.

[34] C. M. White, A. R. Kemanian, and J. P. Kaye, "Implications of carbon saturation model structures for simulated nitrogen mineralization dynamics," *Biogeosciences*, vol. 11, no. 23, pp. 6725–6738, 2014.

[35] "GLDAS: Global Land Data Assimilation System," https://ldas.gsfc.nasa.gov/gldas, 2019.

[36] W. Chen and E. Deelman, "Workflow overhead analysis and optimizations," in *Proceedings of the 6th workshop on Workflows in support of large-scale science*. ACM, 2011, pp. 11–20.

[37] C. Aguado Sanchez, J. Bloomer, P. Buncic, L. Franco, S. Klemer, and P. Mato, "Cvmfs-a file system for the cernvm virtual appliance," in *Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research*, 2008.

[38] R. Ferreira da Silva, W. Chen, G. Juve, K. Vahi, and E. Deelman, "Community resources for enabling and evaluating research in distributed scientific workflows," in *10th IEEE International Conference on e-Science*, ser. eScience'14, 2014, pp. 177–184.

[39] H. Casanova, S. Pandey, J. Oeth, R. Tanaka, F. Suter, and R. Ferreira da Silva, "WRENCH: A Framework for Simulating Workflow Management Systems," in *13th Workshop on Workflows in Support of Large-Scale Science (WORKS'18)*, 2018, pp. 74–85.