2023 IEEE 16th International Conference on Cloud Computing (CLOUD)

2023 IEEE 16th International Conference on Cloud Computing (CLOUD) | 979-8-3503-0481-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/CLOUD60044.2023.00052

Enabling Scalability in the Cloud for Scientific Workflows: An Earth Science Use Case

Paula Olaya
University of Tennessee
Knoxville, USA
polaya@vols.utk.edu

Jakob Luettgau
University of Tennessee
Knoxville, USA
jluettga@utk.edu

Camila Roa
University of Tennessee
Knoxville, USA
mroa@vols.utk.edu

Ricardo Llamas
University of Delaware
Newark, USA
rllamas@udel.edu

Rodrigo Vargas
University of Delaware
Newark, USA
rvargas@udel.edu

Sophia Wen
IBM Research
Yorktown Heights, USA
hfwen@us.ibm.com

I-Hsin Chung
IBM Research
Yorktown Heights, USA
ihchung@us.ibm.com

Seetharami Seelam
IBM Research
Yorktown Heights, USA
sseelam@us.ibm.com

Yoonho Park
IBM Research
Yorktown Heights, USA
yoonho@us.ibm.com

Jay Lofstead
Sandia National Laboratories
Albuquerque, USA
gflofst@sandia.gov

Michela Taufer
University of Tennessee
Knoxville, USA
mtaufer@utk.edu

***Abstract*—Scientific discovery increasingly relies on interoperable, multimodular workflows generating intermediate data. The complexity of managing intermediate data may cause performance losses or unexpected costs. This paper defines an approach to composing these scientific workflows on cloud services, focusing on workflow data orchestration, management, and scalability. We demonstrate the effectiveness of our approach with the SOMOSPIE scientific workflow that deploys machine learning (ML) models to predict high-resolution soil moisture using an HPC service (LSF) and an open-source cloud-native service (K8s) and object storage. Our approach enables scientists to scale from coarse-grained to fine-grained resolution and from a small to a larger region of interest. Using our empirical observations, we generate a cost model for the execution of workflows with hidden intermediate data on cloud services.**

***Index Terms*—Intermediate data, Workflows, Cloud service, High-performance computing, Object storage, Cost model**

I. Introduction

Modern scientific workflows are growing in complexity, comprising multiple interacting blocks that generate, preprocess, and analyze large datasets. These workflows can transform data using four modalities (Fig. 1). In the first modality, workflows produce large output data from small input data. In the second, workflows reduce data via processing a large input and generating a small output. In the third, workflows process the same input data with large data reuse. Last, workflows in the fourth modality produce significant intermediate data that another application reuses, resulting in smaller output data products. Multistage workflows characterized by the fourth modality hide the complexity of large intermediate data, and their overall execution can be significantly affected by the underlying computational infrastructure. The I/O bandwidth of writing and reading that large intermediate data is a key contributor to a long makespan for workflows in the fourth modality.

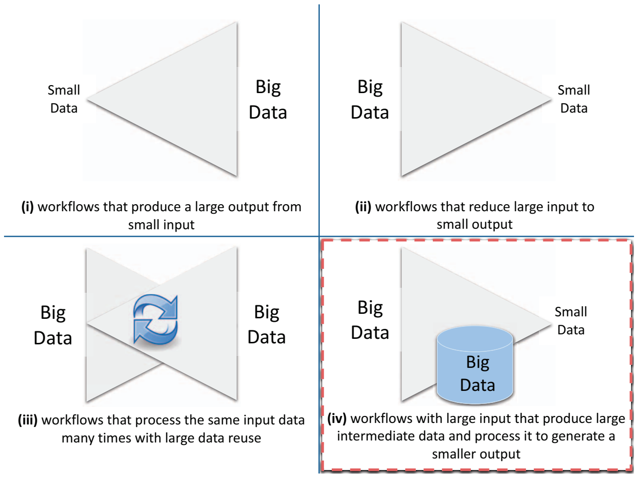

Small Data
Big Data
(i) workflows that produce a large output from small input
Big Data
Small Data
(ii) workflows that reduce large input to small output
Big Data
Big Data
(iii) workflows that process the same input data many times with large data reuse
Big Data
Small Data
Big Data
(iv) workflows with large input that produce large intermediate data and process it to generate a smaller output

Fig. 1: Modalities of data used in scientific workflows (Courtesy of Frank Wuerthwein, SDSC).

To study how the cloud can better serve workflows with large intermediate data, we select two cloud services: HPC as a service with IBM Spectrum LSF (LSF) and cloud-native with Kubernetes (K8s). LSF is an HPC cluster in the cloud close to on-premise HPC but on top of an IaaS platform. K8s is a container-based PaaS platform; we can containerize and execute HPC workflows using K8s. When scientists compose their workflows for these two cloud services, they must answer critical questions regarding (i) compute orchestration: type and number of compute instances required by the workflow and the interaction between them; (ii) data management: RAM size, storage technology, and its connection to the compute instances; and (iii) scalability: automatic allocation of resources as the workflow and its data grow. This paper addresses these questions using an earth science workflow called SOMOSPIE on the two cloud services (i.e., LSF and K8s). SOMOSPIE [1] uses machine learning (ML) models to predict intermediate soil moisture data from low-resolution satellite data to high-resolution values necessary for practical use in earth sciences, including precision forestry and agriculture, hydrology for landscape ecology, and regeneration dynamics [2], [3]. Mainly, we use SOMOSPIE to replicate the studies of scientists who either scale the resolution of a region of interest up from coarse-grained to fine-grained or scale out from a small to a larger region of interest with a fixed resolution. In both cases, the process translates into large intermediate data that must be managed by the cloud services efficiently and may result in additional costs (i.e., performance degradation and monetary invoicing). Our contributions are:

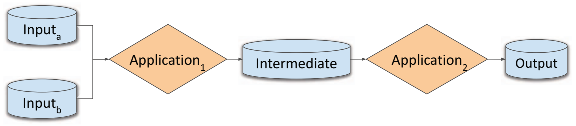
- A description of data and scalability complexity in scientific workflows with large intermediate data through the ML-based SOMOSPIE workflow.
- A methodology to integrate scientific workflows in two cloud services: HPC as a service with IBM Spectrum LSF and cloud-native with Kubernetes.
- A cost model and projections based on empirical observations for workflows with hidden intermediate data through scalability studies.

II. Workflows, Resolutions, and Partitioning

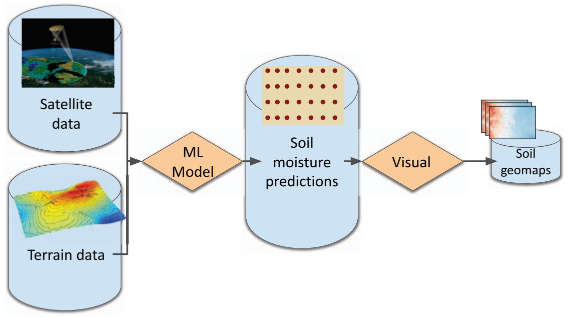
A. Composable ML-based Workflows

Workflows have taken center stage in many domains of science [4]–[8]. They allow scientists to compose complex applications by combining heterogeneous codes; defining parameters; managing and generating input, intermediate, and output data; and controlling dependencies. A scientific workflow consists of one or multiple interoperable applications with their software stack and data (i.e., input, intermediate, and output) that scientists can compose and run to study a scientific problem in a well-defined execution environment. We present an abstraction of a general workflow in Figure 2a with two applications with input, intermediate, and output data.

We use an example of ML-based earth science workflow (SOMOSPIE) that follows the same dataflow structure in Figure 2b. Our SOMOSPIE workflow is composed of two applications. The first uses ML modeling techniques to downscale the 27 km resolution satellite data from the ESA-CCI soil moisture database [9] to higher resolutions necessary for practical use in earth sciences, including precision forestry and agriculture, hydrology for landscape ecology, and regeneration dynamics [2], [3]. The second application performs analytics (e.g., time series, statistical analysis, data-pattern findings) and visualization. SOMOSPIE follows a data transformation in the fourth modality, where it has large input data (i.e., low resolution, satellite-generated soil moisture values and terrain parameters) and produces significant intermediate data (i.e., high-resolution soil moisture predictions) that then is processed, resulting in a smaller output (i.e., images and statistics). We focus on ML modeling of the first application as the large intermediate data brings challenges in terms of resources and performance, resulting in sudden system crashes or unexpected costs. In the first application, the satellite and terrain parameter data are used to train and test an ML model with K-nearest neighbors, random forest, surrogate-based modeling, and other hybrid methods. The generated soil moisture predictions have high resolution (up to 1 m) and are fed into visualization tools to create geographic soil moisture maps and statistics.



(a) General scientific workflow



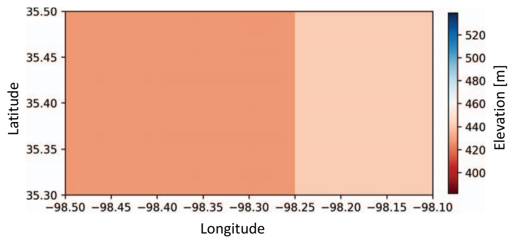
(b) SOMOSPIE workflow

Fig. 2: Structure of (a) a general scientific workflow with one or multiple interoperable applications with input, intermediate, and output data; and (b) the mapping structure of the SOMOSPIE workflow that follows the fourth modality with large input and intermediate data.

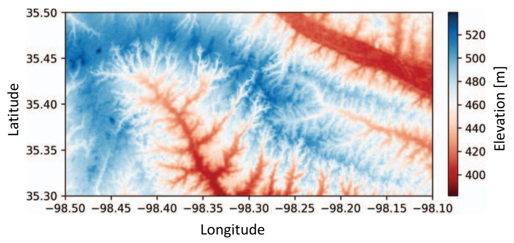
B. Scaling Resolutions and Regions

For ML-based workflows, data is crucial in obtaining better predictions. As data scales, scientists can test the limits of their scientific discovery. We present two ways in which data can scale. The first deals with data that grows because we move from low to high resolutions in a given region (scale up); the second is where data expands as we cover a larger region (scale out). We investigate these scalability scenarios for our SOMOSPIE workflow.

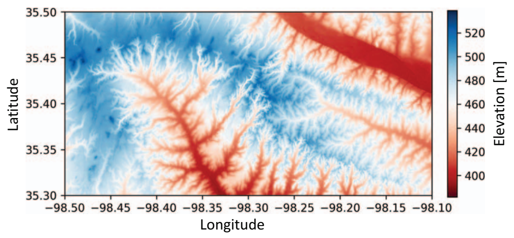
Satellite soil moisture data is collected daily at 0.25 degrees spatial resolution, approximately 27 km [9]. This ESA-CCI database includes records of soil moisture from 1996 until 2020. Scientists input soil moisture and terrain parameter data combinations into SOMOSPIE across time to create different models. Depending on the target resolution of predictions (i.e., from 27 km satellite soil moisture data to 10 m soil moisture predictions) and the region of interest (i.e., from a state to a continent or worldwide), the intermediate data (i.e., soil moisture points on the map) generated by the ML model may



(a) Midwest at 27 km (satellite resolution), 2 × 1 data points grid in the selected area



(b) Midwest at 90 m, 453 × 227 data points grid in the selected area



(c) Midwest at 10 m, 8926 × 4539 data points grid in the selected area

Fig. 3: Example of satellite resolution at (a) 27 km, (b) low resolution of 90 m, and (c) a lower resolution at 10 m for a selected area in the Midwest region. (Scalability: Same region but higher resolution.)

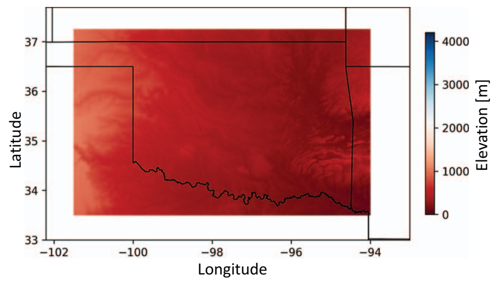
increase exponentially. Scalability studies using SOMOSPIE or similar workflows in earth sciences target two dimensions: resolution and region scalability. When scaling the resolution of soil moisture, scientists define a region of interest and scale the resolution up to generate finer-grained soil moisture values, resulting in more points of soil moisture values over the same region. For example, Figure 3 shows (a) an example of a satellite resolution of 27 km, (b) a higher resolution of 90 m, and (c) a finest-granularity resolution of 10 m for a region centered around Oklahoma (the Midwest region). We move from 450 to 36 M to 2.9 B data points as we increase the resolution at which we aim to predict soil moisture. When scaling the region, scientists define a resolution and scale out on the map to select a larger area. In Figure 4, we scale from the Midwest region at 10 m resolution with an area of 283,499 km^2 to a much larger region, such as the Contiguous United States (CONUS) at the same resolution with an area of 8,080,464 km^2. In this specific scenario, we move from 2.9 B to 167 B soil moisture data points.

We demonstrate the exponential growth of intermediate data for resolution scalability in the Midwest region when scientists predict soil moisture within the same region but scale from a lower resolution at 90 m with 4.25 GB of total data to a higher resolution of 10 m that increases the total data size to 420.76 GB, increasing the data by 100×. We also document the data growth for the region scalability, where scientists scale from the Midwest region (420.76 GB total data size) to the whole CONUS, expanding the data to 14.93 TB while preserving the same resolution. Table I shows three data scenarios for SOMOSPIE that define a region of interest and a resolution at which the scientist aims to predict the soil moisture. Each row represents one of these scenarios: (i) Midwest region at 90 m resolution, (ii) Midwest region at 10 m, and (iii) CONUS at 10 m. For each scenario, we break down the data transformations (i.e., input, intermediate, and output data) and describe the data type, number of points, and the size of each dataset. We observe how input and intermediate data encompass most of that total data size for all scenarios. For all data scenarios, we train our model by using the satellite soil moisture data at 27 km. We average the soil moisture values for the month of January 2010 to ensure the time scalability factor is constant. The time scalability for the same regions across different months is studied in [10].

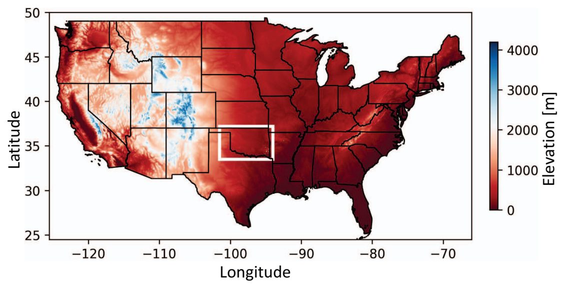
C. Data Partitioning

Scientists apply data partitioning to process temporal and spatial data at different scales. The partitioning enables data parallelism, executing the same application concurrently to different data partitions. Temporal data often exhibit dependencies (e.g., predictions for one year may depend on observations of previous years), making any partitioning along the time dimension often impossible without compromising the accuracy of the ML prediction model. On the other hand, partitioning spatial data is often feasible by considering continuity in a region of interest; scientists can either define a buffer to automatically add information around each partition or use partitions with integrated overlapping neighboring information generated during a pre-processing phase.

We leverage the second type of data partitioning for SOMOSPIE's spatial data. We pre-process the USGS digital elevation models (DEM) to generate terrain parameters of a region of interest that can be partitioned into tiles and deployed for predictions without needing neighbor buffers. Consequently, given a region, we partition it into the number of tiles (tiles$_{total}$). The tiles are independent of each other, as they already include neighboring information in the terrain parameters and thus can be fed independently into the ML model. Table II presents the number of total tiles and the size of the tiles for the Midwest region and CONUS at the two resolutions (i.e., 90 m and 10 m). When we scale up for the Midwest region, we use up to 225 tiles and predict at a higher resolution such as 10 m. The data processed in the execution scale is up to 388.98 GB. Similar data growth occurs when we scale out up to 1,156 tiles, meaning that we go into a larger area, such as CONUS, with the same 10 m resolution, where



(a) Midwest at 10 m



(b) CONUS at 10 m

Fig. 4: Example of (a) a state region, Midwest at 10 m and (b) whole country region, Contiguous United States (CONUS) at 10 m. (Scalability: Same resolution but a larger region.)

TABLE I: Data scenarios for SOMOSPIE. Resolution scalability: from the Midwest region at 90 m to the Midwest region at 10 m. Region scalability: from the Midwest region at 10 m to CONUS at 10 m.

Region Resolution	Data Type	Data Description	Data Characterization Points [#]	Size [B]
Midwest, 90 m x 90 m	Input	Satellite data, 27 km x 27 km	450	44,198
	Input	Terrain params (4 params), 90 m x 90 m	36,073,181	2.43 G
	Intermediate	1 prediction, Midwest, 90 m x 90 m	36,073,181	1.42 G
	Output	1 visualization, Midwest, 90 m x 90 m	-	438,126
Total Midwest 90 m x 90 m				**4.25 G**
Midwest, 10 m x 10 m	Input	Satellite data, 27 km x 27 km	450	44,198
	Input	Terrain params (4 params), 10 m x 10 m	2,921,927,661	248.51 G
	Intermediate	1 prediction, Midwest, 10 m x 10 m	2,921,927,661	135.90 G
	Output	1 visualization, Midwest, 10 m x 10 m	-	4.57 M
Total Midwest 10 m x 10 m				**388.98 G**
CONUS, 10 m x 10 m	Input	Satellite data, 27 km x 27 km	8,214	985,600
	Input	Terrain params (4 params), 10 m x 10 m	99,925,046,500	9.00 T
	Intermediate	1 prediction, CONUS, 10 m x 10 m	99,925,046,500	5.12 T
	Output	1 visualization, CONUS, 10 m x 10 m	-	21.14 M
Total CONUS 10 m x 10 m				**14,934.00 G**

TABLE II: Data partitioning for SOMOSPIE scenarios.

Region, Resolution	Data Type	Data Description	Tiles$_{total}$	Size per tile [B]
Midwest, 90 m x 90 m	Input	Terrain params	1	2.43 G
	Intermediate	1 prediction	1	1.42 G
Midwest, 10 m x 10 m	Input	Terrain params	225	900 M - 1.6 G
	Predictions	1 prediction	225	350 M - 800 MB
CONUS, 10 m x 10 m	Input	Terrain params	1,156	1.1 G - 11 G
	Predictions	1 prediction	1,156	650 MB - 6.1 G

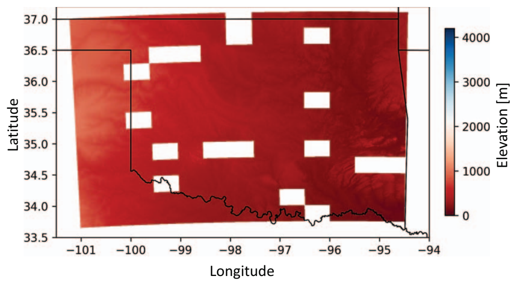
the workflow encompasses 14.9 TB of input, intermediate, and output data. Figure 5 shows an example of tile distribution for 152 tiles used for the Midwest region and CONUS at 10 m resolutions in our experiments. Each tile has the same number of points.

III. Integrating Workflows in HPC on Cloud

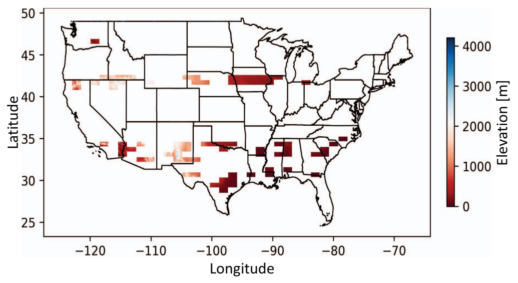
A. HPC on Cloud Services

Traditionally, scientists deploy workflows with large data transformations in HPC systems. However, as these workflows have evolved and the focus has shifted toward scalability, accessibility, and flexibility, the HPC systems show some limitations. For example, the required resources to execute large workflows exceed all resources available in most institutional HPC settings, limiting scientific discovery. To overcome these limitations, we explore two cloud services. These two services aim to bridge the gap between HPC and cloud systems.

Our first service is an HPC service on the cloud. We select the IBM Spectrum load sharing facility (LSF) cluster close to on-premises HPC but on top of an IaaS platform. It provides a fully automated and configurable deployment of LSF-based HPC clusters in the cloud. Our second service is a cloud-native HPC service using Kubernetes (K8s), where we have a container-based PaaS platform on top of which we can containerize and execute HPC workflows. K8s provides an orchestration solution for scheduling and automating containerized applications' deployment, management, and scaling. We execute the same application on both platforms (LSF and K8s). On K8s, the application is containerized. LSF and K8s run on top of a Virtual Private Cloud (VPC) with a secure software-defined network on which scientists can request, configure, and deploy resources on demand. For LSF, virtual machine (VM) instances (nodes) are directly configured inside the VPC. For K8s, Kubernetes runs on all VM instances (nodes) and includes the control plane responsible for managing the Kubernetes objects (e.g., VM instances, pods, PVCs, and PV). A VM instance hosts one or more pods, the most straightforward unit Kubernetes object model. We run one pod per VM instance. LSF has two VM instances: worker and admin (i.e., login



(a) 152 tiles of the same size for the Midwest at 10 m



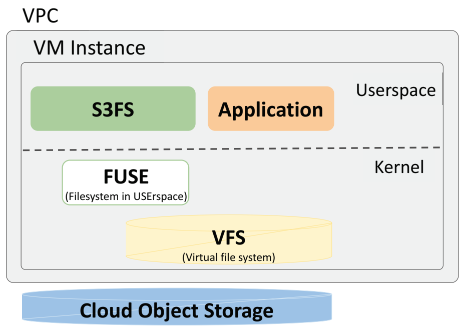
(b) 152 tiles of the same size for CONUS at 10 m

Fig. 5: Tile distribution for Midwest at 10 m and the Contiguous United States (CONUS) at 10 m.

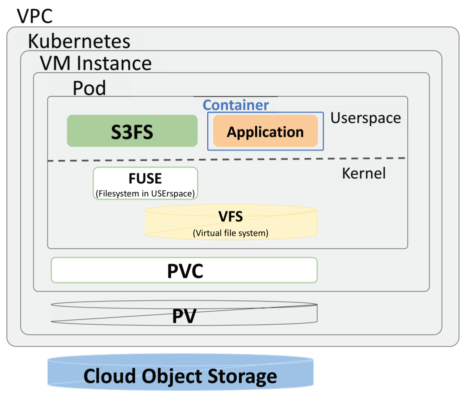
instance, LSF management instances, and storage instances). On the other hand, K8s has worker instances only. In LSF, we use a conventional LSF batch system to schedule our jobs; in K8s, we use the standard Kubernetes scheduler. The standard Kubernetes scheduler ensures pods are attached to worker nodes given the resource limitations (i.e., one pod per worker node) but does not support batch scheduling as LSF does.

B. Data Transformations with Cloud Technology Solutions

In traditional HPC systems, the large intermediate data generated by workflows is typically pushed to parallel file systems or scratch space. When dealing with this data in cloud environments, cloud technology solutions can be used to support the data transformations. We choose the cloud object storage (COS) service, which like a distributed file system, also grants data distribution over multiple instances to provide proportional capacity and throughput scalability. Both LSF and K8s operate with the same COS solution, and we leverage its distributed property to host the input, intermediate, and output data of our workflow into independent COS buckets. There are different packages for mapping object storage into POSIX namespaces, as studied in [11]. We use S3FS [12] for LSF and K8s to map the data in object storage as a file system. S3FS is a FUSE-based (Filesystem in USErspace, white box) file system that enables the users to read and write data in object storage as if they were from local or HPC file systems. When the application calls the virtual file system (VFS), the VFS invokes the FUSE kernel module that maps the COS bucket data into the VM instance's underlying file system. In LSF, we mount the COS bucket directly to the VM instances through S3FS. In K8s, we use Kubernetes objects, such as a PVC (Persistent Volume Claim) and a PV (Persistent Volume) with an S3FS storage class underneath, to mount the COS bucket into the VM instances. We put all the computation and storage layers together and present the architecture for LSF and K8s with COS for a single VM instance in Figure 6.



(a) IBM Spectrum LSF (LSF)



(b) Kubernetes (K8s)

Fig. 6: Architectures of the LSF (HPC) and K8s (cloud-native) services with object storage to handle the data transformations for a single VM instance.

HPC on the cloud and cloud-native services come with default settings that scientists use in good faith, assuming the infrastructure is tuned for their workflow. Unfortunately, these services are tuned for different workflows (i.e., HPC for compute-intensive applications and the cloud for web services). Out-of-the-box settings are not optimal for scientific workflows but tuning the platform's I/O parameters can optimize scientific workflows performance. We recommend tuning the I/O parameters at a single instance level to ensure performance at a large scale. I/O parameters exposed by S3FS include parallel count, multisize part, and caching. The parallel count refers to the number of concurrent threads requesting the object storage. The multisize part is the size in MB of the chunks transferred from and to the object storage. The cache has two options: location for LSF (i.e., off-instance block storage, RAM) and retention policy for K8s (i.e., auto-cache and kernel-cache).

C. *Scalability in the Cloud*

We select cloud services built with the scalability needs of workflows in mind. We leverage the scalability in our cloud infrastructure configuration to map the parallel data nature of a workflow (Sec. II-C). We map each data partition into an independent VM instance to obtain ideal performance when we have the same number of VM instances as the number of data partitions. Figure 7 illustrates mapping each input data partition to an independent VM instance and writing the intermediate data into a separate COS bucket. We model the trade-off between the number of VM instances vs. the total execution time of the application vs. cost.

We schedule our parallel jobs using the LSF batch system for LSF. However, the standard scheduler in K8s does not support traditional batch scheduling. To address this, we schedule the parallel jobs based on a common template, and by using expansions, we define which partition of the data each job processes. As we increase the number of VM instances reading and writing data, we leverage COS's proportional capacity and throughput scalability to host our input, intermediate, and output data. In Figure 7, we present the composition of a workflow in the fourth modality (large intermediate data). We make a representation at the VM instance level connected to three COS buckets for the input, intermediate, and output data, which are common resources for LSF and K8s. We provide insights about the I/O scalability as more VM instances read and write concurrently into independent COS buckets.

IV. Scalability and Cost on the Cloud

A. *Tuning I/O Parameters*

We integrate SOMOSPIE into two cloud services: an HPC service in the cloud (i.e., IBM Spectrum LSF) and an open-source cloud-native (K8s) service. We study the scalability and the cost of running the SOMOSPIE workflow on the two cloud services. In particular, we want to understand the effects of scaling up the resolution (i.e., low to high) and scaling out the region (i.e., from a state to the entire CONUS).

We tune I/O parameters for a single tile execution because the tile is the basic data unit we process in each independent VM instance. We define the number of cores in a VM instance and its RAM size based on the SOMOSPIE requirements (i.e., data and application). For the Midwest region at 90 m, we require a RAM size of a minimum of 32 GB, and we select a VM type of 4 cores and 32 GB. The default settings for the S3FS I/O parameters: parallel count is set to 5 threads, multisize part is 10 MB, and caching is off-instance for LSF and auto-cache for K8s respectively. We tune the performance on a single instance using the Midwest region at a 90 m resolution because the region has a single tile for reading (2.43 GB) and writing (1.42 GB), fitting in a single node. We use FIO (Flexible IO tester) [13] to explore the parametrical space quickly and inexpensively. We run the FIO benchmark for the write operation because it is the most time-consuming I/O operation. We set the FIO jobs to have one sequential write file of 1.4 GB with a block size of 52 MB to match the I/O operation of our application.

We run the benchmark 10 times on a large VM instance with a profile of 48 cores and 192 GB with an Intel Xeon Processor (Skylake, IBRS). We perform an exhaustive search with different combinations of the three S3FS parameters. The parallel count ranges from 5 to 20 threads, the multisize part ranges from 5 MB to 54 MB in size for the transferred chunks to the COS, and the caching considers both a and b for LSF and c and d for K8s. Table III presents the write bandwidth obtained by the FIO benchmark when using the default parameters (i.e., five threads, 10 MB chunk size for both services, and caching off-instance for LSF and auto-cache for K8s) and for the tuned S3FS parameters generating the best empirically observed I/O performance (i.e., 12 threads, 10 MB chunk size, and caching in RAM for LSF; 20 threads, 40 MB chunk size, and kernel cache retention policy for K8s).

TABLE III: Write bandwidth statistics obtained when running the FIO benchmark ten times with the default and the tuned S3FS parameters generating the best empirically observed I/O performance. PC: Parallel count. MP: Multisize part.

Cluster	**Default Parameters**	**Write Bandwidth [MB/s]**	**Tuned Parameters**	**Write Bandwidth [MB/s]**

LSF	PC = 5
	MP = 10 MB
	Off-instance cache
		median=255
		mean=252
		stdev=13
			PC = 12
			MP = 10 MB
			Cache in RAM
				median=423
				mean=406
				stdev=54

K8s	PC = 5
	MP = 10 MB
	Auto cache
		median=200
		mean=196
		stdev=14
			PC = 20
			MP= 40 MB
			Kernel cache
				median=350
				mean=338
				stdev=39

Based on the results from the FIO benchmark, we make an informed decision regarding the S3FS parameters for executing our SOMOSPIE workflow on the Midwest region at the 90 m resolution on both clusters (i.e., LSF and K8s). We use the same instance type used for the benchmark [i.e., 48 cores and 192 GB with an Intel Xeon Processor (Skylake, IBRS)] to run the workflow. We run our application 10 times, measuring the read and write bandwidth.

Figure 8 presents a boxplot of the measured bandwidth over the 10 runs. We observe an improvement in write bandwidth of 25% for LSF when using the S3FS tuned parameters (Fig. 8a), and 1.3% for K8s (Fig. 8b). As a consequence of tuning the write operation, we also improve the read bandwidth of 17.8% for LSF (Fig. 8c) and 28.4% for K8s (Fig. 8d). With the parameter tuning, we can reduce the read and write bandwidth variability for LSF and K8s and bring the read and write performance closer to each other, closing the gap between platforms. With the distributed nature of our predictions, in which regions are cut in tiles (Sec II-C), we can use the tuned I/O parameters to study predictions at a large scale.

B. *Soil Moisture Predictions at the Large Scale*

Predicting soil moisture at high resolutions or for large regions requires scalable platforms to enable data growth. Cloud platforms allow scientists to scale resources to support their workflow requirements. However, the scalability consequences in terms of performance for a given platform are often

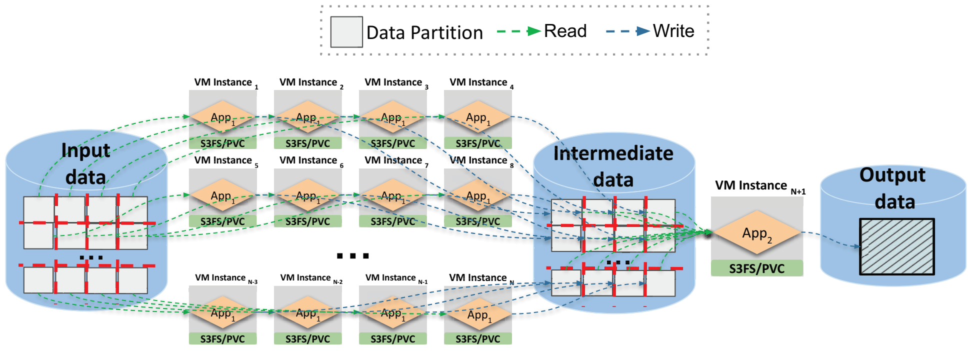
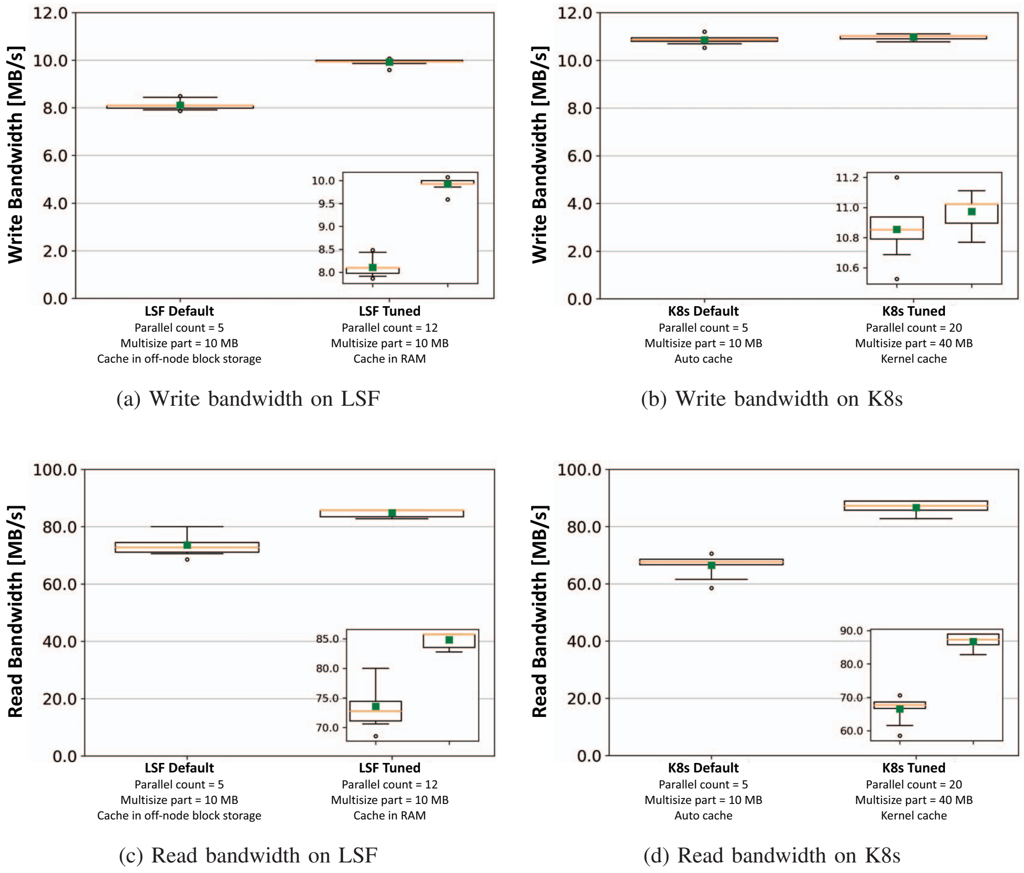


Fig. 7: Composition of a workflow with large intermediate data (fourth modality) using the cloud services (i.e., VM instances from each HPC on cloud service connected to three different COS buckets through S3FS).



(a) Write bandwidth on LSF

(b) Write bandwidth on K8s

(c) Read bandwidth on LSF

(d) Read bandwidth on K8s

Fig. 8: SOMOSPIE write and read bandwidth comparison on the LSF and K8s deployment clusters before and after tuning the s3fs parameters when executing for the Midwest region at 90 m resolution. The smaller images zoom into the boxplots to outline the difference in I/O performance.

unknown to application teams. We study the I/O performance of the object storage as we scale the soil moisture predictions in resolution (from 90 m to 10 m in the Midwest region) and region (from Midwest to CONUS at 10 m).

We decompose the data into tiles, 225 for Midwest at 10 m and 1,156 for CONUS at 10 m (Table II). We process each tile in an independent VM instance. We use 2 cores and 16 GB for Midwest at 10 m, and 8 cores and 64 GB VM type for CONUS at 10 m. Figure 9 presents the multi-instances configuration.

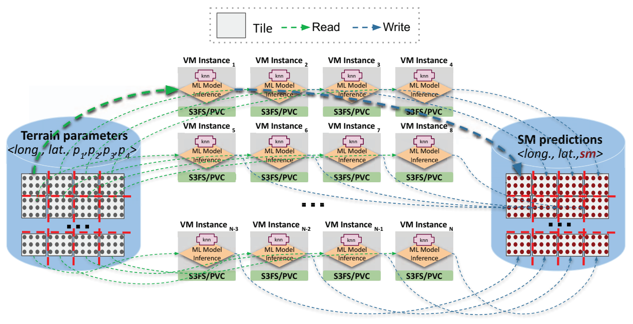


Fig. 9: Multiple VM instances configuration for the Midwest region and CONUS at 10 m resolution.

As we scale the number of VM instances to match the number of tiles to process them in parallel, we increase the number of I/O requests to the two COS buckets. We measure the I/O performance as we read multiple tiles at the time from the terrain parameters COS bucket and write multiple tiles with the soil moisture predictions to the SM predictions COS bucket. We do a weak scaling test for the object storage solution. In other words, we measure how the read and write bandwidth varies with the number of VM instances processing a tile with the same size per instance.

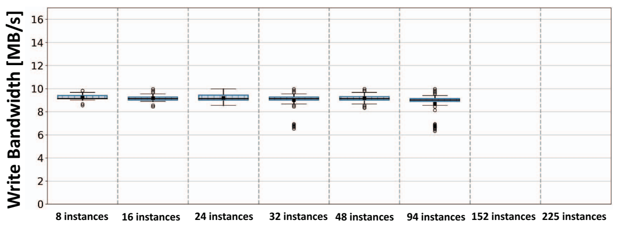
We present the weak scaling results for the Midwest region at 10 m resolution in Figure 10. We consider up to 225 tiles that read 1.2 GB of data per tile and write 685 MB (Figure 5a). The number of tiles, and the associated number of VM instances, range as follows: 8, 16, 24, 32, 48, 94, 152, and 225 (the entire Midwest region). We apply some resource constraints on the HPC service in the cloud to mimic the scientists having access to on-premise HPC systems (i.e., limited allocations and computational nodes). On LSF, our experiments are constrained to 200 cores that can be split between the worker and admin instances. We use a worker VM instance of 2 cores and 16 GB; by eliminating the cores for the admin VM instances, we scale up to 94 VM instances in LSF. Instead, in the cloud-native service, K8s, we do not have any restrictions in terms of computational resources, so we can scale up to the number of VM instances. For LSF, the COS bucket maintains I/O performance when we write 64.4 G (Figure 10a) and read 112.8 GB (Figure 10c) of data in parallel from 94 nodes for LSF for the Midwest region at 10 m resolution. For K8s, we observe that the I/O performance of the object storage scales as we write 135.9 G (Figure 10b) and read 248 GB (Figure 10d) of data in parallel from 225 nodes. We translate the performance into accumulated bandwidth across all VM instances. For LSF, we reach 864.8 MB/s write bandwidth and 5.6 GB/s read bandwidth, having 94 VM instances. For K8s, we reach 2.4 GB/s write bandwidth and 11.2 GB/s read bandwidth having 225 VM instances. Overall, we observe no I/O performance degradation in the object storage as we increase the number of instances of reading and writing in parallel for LSF and K8s. We note higher bandwidth variability (Figures 10b and 10d) in the K8s service that is attributed to the additional layers in the cloud-native service architecture and the virtualization of resources.

We increase the tile size to demonstrate I/O performance with a similar number of VM instances (i.e., 8, 16, 24, 32, 48, 94, 152) but a larger problem size (larger tiles) using the CONUS at 10 m scenario. We increase the tile size by 4 and now read tiles of size 5.1 GB and write 2.8 GB soil moisture predictions for each tile. We use a worker VM instance of 8 cores and 64 GB for this scenario. By eliminating the cores for the admin VM instances, we can scale up to 24 VM instances in LSF before hitting the platform's resource constraints. We scale to 152 for the CONUS and stop because of budget allocation limits. We observe that the I/O performance of the object storage scales up when writing 425.6 GB of SM data point in parallel with 152 VM instances in K8s (Figure 11b). For LSF, the object storage I/O performance scales as we write 67.2 GB (Figure 11a) and read 122.4 GB (Figure 11c) of data in parallel from 24 nodes for CONUS at 10 m resolution. The reading in K8s (Figure 11d) scales up to 94 VM instances (479.4 GB); when we reach 152 VM instances (775.2 GB), the performance drops from 80 MB/s to 65 MB/s. This is an empirical trend that can be further investigated in future work. We translate the performance into accumulated bandwidth across all VM instances. For LSF, we reach around 240 MB/s write bandwidth and 1.8 GB/s read bandwidth, having 24 VM instances. For K8s, we reach 1.6 GB/s write bandwidth and 10.6 GB/s read bandwidth having 152 VM instances. We obtain limited scalability performance results when we limit the number of VM instances for LSF, mimicking the limited access to educational resource allocations. Alternatively, we demonstrate scalability with K8s (cloud-native service) as we stretch the resources under stress conditions.

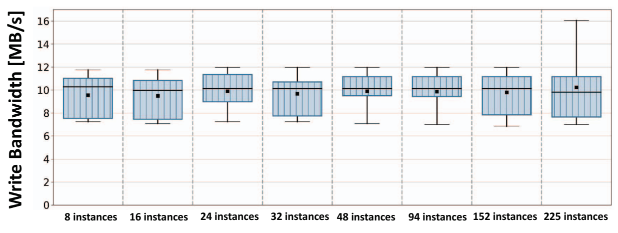
C. *Modeling Costs for Soil Moisture Prediction*

As we scale the data in the workflow, the prediction costs grow. When scaling to higher resolutions or larger regions, a workflow like SOMOSPIE allows for saturating all resources available in most institutional settings. The consequence of this limit is that we have to preprocess the tiles in multiple batches one after another. A benefit of cloud environments is that we can assume virtually infinite resources, allowing us to preprocess all tiles in parallel in constant time. We aim to understand the tradeoff between time and cost for all the resources. We develop a model for calculating the costs of predicting soil moisture in LSF and K8s. Our model measures the total cost for the computational, I/O, and storage resources. The storage cost for the object storage technology is $cost_{storage} = \frac{\$USD}{hour} * capacity$, where $capacity$ is the total data stored in the bucket. The storage cost is the same for LSF and K8s.

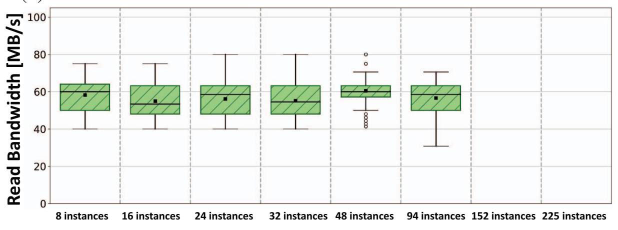
We establish our computational and I/O cost model with t_{tile} as our basic unit. This is the time in hours that it takes to



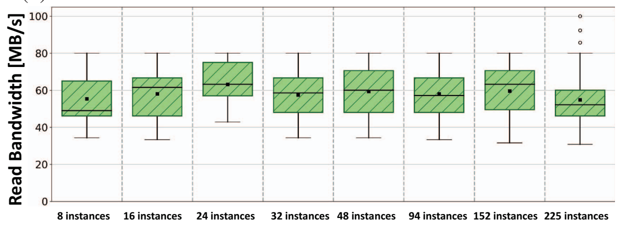
(a) Weak scale write bandwidth for Midwest at 10 m on LSF.



(b) Weak scale write bandwidth for Midwest at 10 m on K8s.

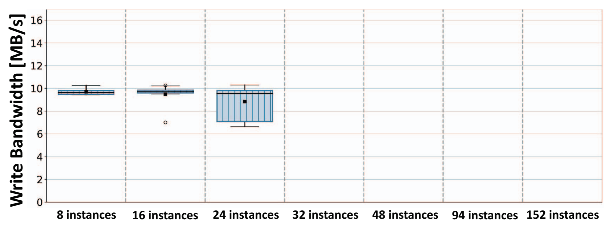


(c) Weak scale read bandwidth for Midwest at 10 m on LSF.

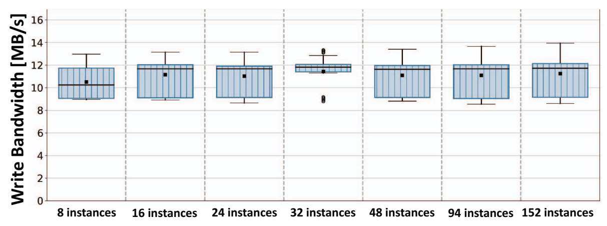


(d) Weak scale read bandwidth for Midwest at 10 m on K8s.

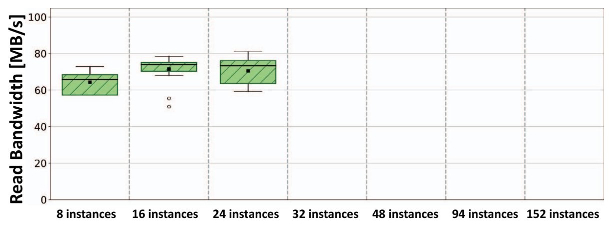
Fig. 10: I/O bandwidth scalability as we increase the number of VM instances (from 8 to 225) for reading a size $1.2GB$ tile and writing $685MB$ soil moisture predictions for each tile for the Midwest region at 10 m resolution.



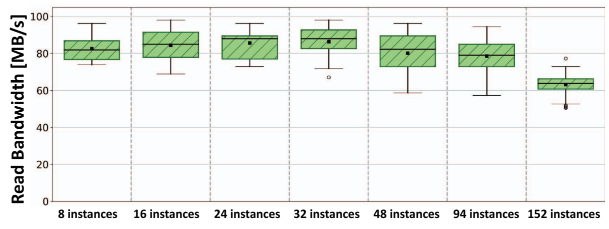
(a) Weak scale write bandwidth for CONUS at 10 m on LSF.



(b) Weak scale write bandwidth for CONUS at 10 m on K8s.



(c) Weak scale read bandwidth for CONUS at 10 m on LSF.



(d) Weak scale read bandwidth for CONUS at 10 m on K8s.

Fig. 11: I/O bandwidth scalability as we increase the number of VM instances (from 8 to 152) for reading a tile of size 5.1 GB and writing 2.8 GB soil moisture predictions for each tile for the CONUS at 10 m resolution.

TABLE IV: Modeling costs apply to our three SOMOSPIE data scenarios: Midwest 90 m, Midwest 10 m, and CONUS 10m on the two deployment clusters: LSF and K8s. We list the times in seconds for readability purposes, but when calculating the cost, we convert them into hours.

Cluster	Data Scenario	$tiles_{total}$	$N_{workerVMs}$	N_{cores}	RAM_{size}	$t_{tile}[s]$	$t_{total}[s]$	$cost_{total}[\$]$
LSF	Midwest 90m	1	1	4	32	331.5	331.5	0.5
	Midwest 10 m	225	94	2	16	155.7	466.8	16.0
	CONUS 10 m	1156	24	8	64	868.4	42483.0	1484.1
K8s	Midwest 90m	1	1	4	32	330.9	330.6	0.2
	Midwest 10 m	225	225	2	16	153.5	153.5	12.3
	CONUS 10 m	1156	1156	8	64	984.8	984.8	1638.7

read, infer high-resolution soil moisture values (computation), and write the predictions for a single tile (Eq. 1).

$$t_{tile} = t_{read} + t_{compute} + t_{write} \quad (1)$$

We establish the cost of a VM as in Eq. 2. Depending on the size of the tile, we define a VM type for our worker instances in terms of the number of cores (N_{cores}) and the

RAM capacity (RAM_{size}). Each cloud vendor has a $\frac{\$USD}{hour}$ rate for the CPU and memory in the VM instance.

$$cost_{VM} = (\frac{\$USD_{cpu}}{hour} * N_{cores} + \frac{\$USD_{mem}}{hour} * RAM_{size}) \tag{2}$$

When we scale to all the tiles for a region of interest, we consider the number of tiles in the region ($tiles_{total}$) and the number of worker VM instances available in the LSF and K8s clusters ($N_{workerVMs}$). With these two variables, we can calculate the ratio of batches of tiles that will be processed in parallel as $\left\lceil \frac{tiles_{total}}{N_{workerVMs}} \right\rceil$. We multiply this ratio by the t_{tile} to obtain the total time (t_{total}) that it takes all worker VM instances in the cluster to execute the application. We determine the computational cost once we have the time for all tiles. To this end, we use the cost for the VM type we define for our worker instances ($cost_{workerVM}$) and the number of worker instances available ($N_{workerVMs}$). Combining the time it takes to execute all tiles, the cost of a worker VM, and the number of worker instances, we define the computational cost for all tiles in a region as in Eq. 3.

$$\begin{aligned} cost_{compu} =& N_{workerVMs} * cost_{workerVM} \\ & * \left\lceil \frac{tiles_{total}}{N_{workerVMs}} \right\rceil * t_{tile} \end{aligned} \tag{3}$$

The computational cost in K8s for all tiles is the same as $K8scost_{total} = cost_{total}$ because the K8s cluster has only worker instances. However, the LSF cluster and the worker instances include the admin instances. Our LSF cluster has one login, two management, and one storage instance. We have the same VM type for all of them and refer to its cost as $cost_{adminVM}$. Therefore, we define the computational cost for LSF as the sum of the worker instances and the LSF admin VMs, as in Eq. 4.

$$\begin{aligned} LSFcost_{total} =& (4 * cost_{adminVM} \\ & + N_{workerVMs} * cost_{workerVM}) \\ & * \left\lceil \frac{tiles_{total}}{N_{workerVMs}} \right\rceil * t_{tile} \end{aligned} \tag{4}$$

We apply our model to the three data scenarios and present the computational costs for LSF and K8s in Table IV. Each cloud vendor provides a rate for their instances in terms of CPU ($\frac{\$USD_{cpu}}{hour}$) and memory ($\frac{\$USD_{mem}}{hour}$). After checking different cloud vendors rates, we define $\frac{\$USD_{cpu}}{hour} = 8cents/h$ and $\frac{\$USD_{mem}}{hour} = 7cents/h$. For LSF, we establish the same type of VM for our admin VM instances with two cores and 8 GB of RAM. We calculate the t_{tile} by taking the geometric mean across all tiles and different runs for each stage (i.e., read, compute, and write) and data scenario. We observe that given the worker VM instances in the clusters, the total time processing all the tiles is higher in LSF than in K8s since in K8s we can run all tiles in parallel. A second remark is that the computational cost for K8s is lower than LSF for Midwest at 90m and 10 m. As there are more tiles, like in the CONUS at 10 m example, two factors seem to penalize the cost for K8s, making it higher than for LSF. The first factor is the time per tile because 1156 instances in parallel in the cloud augment the variability and decrease the compute performance. The second factor is the number of VM instances; even though in LSF the 24 worker VM instances would take around 11.8 hours, there is a tax for the amount of worker VM instances in K8s.

V. Related Work

Our work is part of a broader effort to develop tools and best practices for supporting HPC and cloud convergence for scientific workflows. Several efforts have addressed porting scientific workflows traditionally run in HPC and HTC (high-throughput computing) systems into the cloud across domains such as astronomy [14], bioinformatics [15], biology [16], and engineering [17]. In earth sciences, Pangeo [18] prototypes an architecture composed of a cloud object storage and compute cluster. The external cloud storage is dedicated to the easy retrieval of large-volume datasets. K8s provisions the compute cluster, and Dask enables computation parallelism. As cloud technology evolves, new services in the cloud (e.g., HPC services and cloud-native services) and storage technologies (e.g., block storage, object storage) emerge to overcome past challenges. One of these challenges is leveraging those services originally tailored for applications far from those in scientific domains. Our work studies the issue of data movement and the associated parameter tuning. Although our study is applied to an earth science application, the proposed approach and model can be transferred to other applications. In other words, our work complements theoretical studies [19], [20] addressing significant data transformation in the cloud by offering a practical approach for scientists to explore and tune the cloud for their workflows. There is a growing need for developing persistent scientific workflows to seamlessly connect and integrate software stacks and data services across cloud platforms supported by virtualization and data provenance [21]. Containerization of scientific workflow enables reusability, portability, and reproducibility of results [4], [22] as well as ease of system maintenance efforts [23]–[26]. Our work supports these efforts as the containers can be integrated into any cloud-native services. As cloud-native services, such as K8s, gain popularity, our work joins other approaches [27], [28] that enable distributed computation of scientific workflows. We emphasize tuning the default settings and parameters for scalability improvements and cost mitigation.

VI. Conclusions

In this paper, we study the composition of large-scale scientific workflows that deal with large intermediate data on two HPC on cloud services. The first one is LSF, an HPC as a service on cloud resources, close to on-premise HPC but on top of an IaaS platform. The second is K8s, a container-based PaaS platform where workflows are containerized and executed. We provide best practices on cloud infrastructure to enable the shareability and scalability of scientific workflows, increase the productivity of scientists, and accelerate scientific discovery. We demonstrate the scalability of the cloud infrastructure for an exemplary ML-based scientific workflow in earth sciences called SOMOSPIE. SOMOSPIE uses ML models to predict satellite soil moisture data to a resolution necessary for policymaking and precision agriculture. Specifically, we measure performance when scaling up the resolution (i.e., low to high) and scaling out the region (i.e., from a state to the entire CONUS). We reach an accumulated write bandwidth of 864.8 MB/s and 5.6 GB/s accumulated read bandwidth having 94 VM instances in LSF. We obtain 2.4 GB/s write bandwidth and 11.2 GB/s read bandwidth having 225 VM instances in K8s. Future work includes continuing the HPC and cloud convergence initiative by focusing on scheduling policies on Kubernetes and offering operators that facilitate the automatic composition of scientific workflows.

ACKNOWLEDGMENT

The authors acknowledge the support of IBM through a Shared University Research Award; Sandia National Laboratories; the National Science Foundation through the awards #2028923, #2103845, #2103836, and #2138811; and the Access program through the NSF grant #2138296.

REFERENCES

[1] D. Rorabaugh, M. Guevara, R. Llamas, J. Kitson, R. Vargas, and M. Taufer, "SOMOSPIE: A Modular SOil MOisture SPatial Inference Engine Based on Data-Driven Decisions," in *Proc. of the 15th International Conference on eScience (eScience)*, pp. 1–10, IEEE, 2019.

[2] R. M. Llamas, M. Guevara, D. Rorabaugh, M. Taufer, and R. Vargas, "Spatial Gap-Filling of ESA CCI Satellite-Derived Soil Moisture Based on Geostatistical Techniques and Multiple Regression," *Remote. Sens.*, vol. 12, no. 4, p. 665, 2020.

[3] M. Guevara, M. Taufer, and R. Vargas, "Gap-free global annual soil moisture: 15 km grids for 1991–2018," *Earth System Science Data*, vol. 13, no. 4, pp. 1711–1735, 2021.

[4] P. Olaya, D. Kennedy, R. Llamas, L. Valera, R. Vargas, J. Lofstead, and M. Taufer, "Building trust in earth science findings through data traceability and results explainability," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 704–717, 2023.

[5] T. Coleman, H. Casanova, K. Maheshwari, L. Pottier, S. R. Wilkinson, J. Wozniak, F. Suter, M. Shankar, and R. F. Da Silva, "WfBench: Automated Generation of Scientific Workflow Benchmarks," in *Proc. of the International Workshop on Performance Modeling, Benchmarking and Simulation of High-Performance Computer Systems (PMBS)*, pp. 100–111, ACM/IEEE, 2022.

[6] M. Wolf, J. Logan, K. Mehta, D. Jacobson, M. Cashman, A. M. Walker, G. Eisenhauer, P. Widener, and A. Cliff, "Reusability First: Toward FAIR Workflows," in *Proc. of the International Conference on Cluster Computing (CLUSTER)*, pp. 444–455, IEEE, 2021.

[7] K. Tran, A. Palizhati, S. Back, and Z. W. Ulissi, "Dynamic workflows for routine materials discovery in surface science," *Chemical Information and Modeling*, vol. 58, no. 12, pp. 2392–2400, 2018.

[8] H. S. Stein and J. M. Gregoire, "Progress and prospects for accelerating materials science with automated and autonomous workflows," *Chemical Science*, vol. 10, pp. 9640–9649, 2019.

[9] "Soil Moisture CCI." Available at https://www.esa-soilmoisture-cci.org [Online; accessed 02-25-2023].

[10] R. M. Llamas, L. Valera, P. Olaya, M. Taufer, and R. Vargas, "Downscaling Satellite Soil Moisture Using a Modular Spatial Inference Framework," *Remote Sensing*, vol. 14, no. 13, 2022.

[11] P. Olaya, J. Luettgau, N. Zhou, J. Lofstead, G. Scorzelli, V. Pascucci, and M. Taufer, "NSDF-FUSE: A Testbed for Studying Object Storage via FUSE File Systems," in *Proc. of the 31st International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, p. 277–278, ACM, 2022.

[12] s3fs-fuse, "s3fs." Available at https://github.com/s3fs-fuse/s3fs-fuse, version 1.91 [Online; accessed 02-25-2023].

[13] J. Axboe, "fio: Flexible I/O ." Available at https://github.com/axboe/fio, version 3.33 [Online; accessed 02-25-2023].

[14] Q. Jiang, Y. C. Lee, M. Arenaz, L. M. Leslie, and A. Y. Zomaya, "Optimizing Scientific Workflows in the Cloud: A Montage Example," in *Proc. of the 7th International Conference on Utility and Cloud Computing (UCC)*, pp. 517–522, IEEE, 2014.

[15] M. J. Rosa, C. G. Ralha, M. Holanda, and A. P. Araujo, "Computational Resource and Cost prediction Service for Scientific Workflows in Federated Clouds," *Future Generation Computer Systems*, vol. 125, pp. 844–858, 2021.

[16] T. Reiter, P. T. Brooks†, L. Irber†, S. E. K. Joslin†, C. M. Reid†, C. Scott†, C. T. Brown, and N. T. Pierce-Ward, "Streamlining data-intensive biology with workflow systems," *GigaScience*, vol. 10, no. 1, 2021. giaa140.

[17] M. Krämer, H. M. Würz, and C. Altenhofen, "Executing cyclic scientific workflows in the cloud," *Journal of Cloud Computing*, vol. 10, p. 25, Apr 2021.

[18] R. P. Abernathey, T. Augspurger, A. Banihirwe, C. C. Blackmon-Luca, T. J. Crone, C. L. Gentemann, J. J. Hamman, N. Henderson, C. Lepore, T. A. McCaie, N. H. Robinson, and R. P. Signell, "Cloud-Native Repositories for Big Scientific Data," *Computing in Science and Engineering*, vol. 23, no. 2, pp. 26–35, 2021.

[19] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li, "Big Data Processing in Cloud Computing Environments," in *Proc. of the 12th International Symposium on Pervasive Systems, Algorithms, and Networks*, pp. 17–23, IEEE, 2012.

[20] M. Barika, S. Garg, A. Y. Zomaya, L. Wang, A. V. Moorsel, and R. Ranjan, "Orchestrating Big Data Analysis Workflows in the Cloud: Research Challenges, Survey, and Future Directions," *ACM Comput. Surv.*, vol. 52, no. 5, 2019.

[21] H. Bhatia, F. Di Natale, J. Y. Moon, X. Zhang, J. R. Chavez, F. Aydin, C. Stanley, T. Oppelstrup, C. Neale, S. K. Schumacher, D. H. Ahn, S. Herbein, T. S. Carpenter, S. Gnanakaran, P.-T. Bremer, J. N. Glosli, F. C. Lightstone, and H. I. Ingólfsson, "Generalizable Coordination of Large Multiscale Workflows: Challenges and Learnings at Scale," in *Proc. of the International Conference for High-Performance Computing, Networking, Storage and Analysis (SC)*, p. 1–16, ACM/IEEE, 2021.

[22] D. Kennedy, P. Olaya, J. Lofstead, R. Vargas, and M. Taufer, "Augmenting Singularity to Generate Fine-grained Workflows, Record Trails, and Data Provenance," in *Proc. of the 18th International Conference on e-Science (e-Science)*, pp. 403–404, IEEE, 2022.

[23] A. Dusia, Y. Yang, and M. Taufer, "Network Quality of Service in Docker Containers," in *Proc. of the International Conference on Cluster Computing (CLUSTER)*, pp. 527–528, IEEE, 2015.

[24] S. McDaniel, S. Herbein, and M. Taufer, "A Two-Tiered Approach to I/O Quality of Service in Docker Containers," in *Proc. of the International Conference on Cluster Computing (CLUSTER)*, pp. 490–491, IEEE, 2015.

[25] J. Monsalve, A. Landwehr, and M. Taufer, "Dynamic CPU Resource Allocation in Containerized Cloud Environments," in *Proc. of the International Conference on Cluster Computing (CLUSTER)*, pp. 535–536, IEEE, 2015.

[26] S. Herbein, A. Dusia, Ayushvand Landwehr, J. McDaniel, Seanvand Monsalve, S. R. Yang, Yangvand Seelam, and M. Taufer, "Resource Management for Running HPC Applications in Container Clouds," in *Proc. of the International Supercomputing Conference (ISC)*, pp. 261–278, Springer, 2016.

[27] B. Baliś, A. Broński, and M. Szarek, "Auto-scaling of Scientific Workflows in Kubernetes," in *Proc. of the International Conference on Computational Science (ICCS)*, pp. 33–40, Springer, 2022.

[28] Z. Yang, P. Nguyen, H. Jin, and K. Nahrstedt, "MIRAS: Model-based Reinforcement Learning for Microservice Resource Allocation over Scientific Workflows," in *Proc. of the 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 122–132, IEEE, 2019.