2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

Machine Learning-assisted Computational Steering of Large-scale Scientific Simulations

Wuji Liu*, Qianwen Ye*, Chase Q. Wu*, Yangang Liu†, Xin Zhou† and Yunpeng Shan†
*Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA
†Environmental & Climate Sciences Department, Brookhaven National Laboratory, Upton, NY 11973, USA
Email: {wl87,qy57,chase.wu}@njit.edu, {lyg,xzhou,yshan}@bnl.gov

***Abstract*—Next-generation scientific applications in various fields are experiencing a rapid transition from traditional experiment-based methodologies to large-scale computation-intensive simulations featuring complex numerical modeling with a large number of tunable parameters. Such model-based simulations generate colossal amounts of data, which are then processed and analyzed against experimental or observation data for parameter calibration and model validation. The sheer volume and complexity of such data, the large model-parameter space, and the intensive computation make it practically infeasible for domain experts to manually configure and tune hyperparameters for accurate modeling in complex and distributed computing environments. This calls for an online computational steering service to enable real-time multi-user interaction and automatic parameter tuning. Towards this goal, we design and develop a generic steering framework based on Bayesian Optimization (BO) and conduct theoretical performance analysis of the steering service. We present a case study with the Weather Research and Forecast (WRF) model, which illustrates the performance superiority of the BO-based tuning over other heuristic methods and manual settings of domain experts using regret analysis.**

***Index Terms*—Computational steering; parameter tuning; bayesian optimization; machine learning.**

I. Introduction

The advance in supercomputing technology is expediting the transition in various basic and applied sciences from traditional laboratory-controlled experimental methodologies to modern computational paradigms involving complex numerical modeling and extreme-scale simulations of physical phenomena, chemical reactions, climatic changes, and biological processes. These model-based simulations, which have become an essential component in next-generation scientific applications, typically generate a large volume of model outputs, which must be processed and analyzed by domain experts in a timely manner using various computing techniques for knowledge discovery and scientific innovation. Increasingly, such simulation-computing processes are represented, executed, managed, and orchestrated by workflow technologies. Particularly, real-time user interaction and online cooperative steering of remote simulation form a closed control loop of the workflow-based research process. In fact, parameter tuning and data analysis for model exploration and evaluation constitute a crucial part of these mission- and time-critical research processes [1]–[4].

One typical scientific application of such types is the research on the Earth's weather and climate system, which involves multiple physical processes acting over a wide range of scales spanning from microphysics at the level of individual cloud droplets to cloud systems at regional and global scales as shown in Fig. 1 [5]. Limited by computational resources and incomplete physical understanding, most of these models contain approximate representations of processes that occur at the spatiotemporal scales smaller than model grid spacing. Such subgrid parameterizations often contain empirical parameters that need to be validated or tuned against measurements. Depending on the subgrid processes in question, the number of tunable parameters can range from several up to hundreds, and the specific values of these parameters likely vary with weather and cloud regimes. Thus, the process of "objective tuning" poses a great challenge to the computational communities as well as the Earth Science community including forecasting of climate, weather, and renewable energies such as wind and solar.

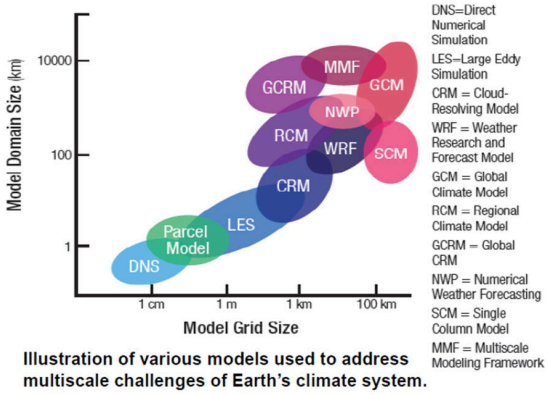

Fig. 1. Widely-used multi-scale, multi-physics models for the Earth's weather and climate system.

To instantiate our discussion, we shall take one specific domain of applications - renewable energy as an example. Renewal energy generation is different from many traditional energy sources in that its availability highly depends on time-varying atmospheric phenomena such as solar irradiance, winds, clouds, and precipitations. For example, a small improvement (around 10% - 20% improvement in the mean absolute error) in wind forecast can lead to millions of dollars in cost savings. Since the total electric supply and demand

978-0-7381-2646-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00138

must always remain in synchronism, a cost-efficient integration of such renewables in the electric grid ultimately depends on our ability to accurately forecast related key meteorological variables - solar irradiance for solar farms and winds for wind farms - over various time horizons (e.g., minutes, hours, and days ahead) and at high space-time resolutions. The physical forecasting model of WRF-Solar is built on the widely used community Weather Research and Forecast (WRF) model [6] with an emphasis on forecasting solar and wind energies, and contains numerous parameterized subgrid processes that affect model performance, including cloud and aerosol microphysics, radiative transfer, planetary boundary layer, turbulence, convection, and land surface.

The development of such models must iterate through complex parameterization, which, for scientists, is a labor-intensive, time-consuming process requiring careful considerations of various trade-offs and calibrations with observations/experiments or domain knowledge. The combination of high-resolution measurements and high-resolution physical modeling in dynamic tuning of high-dimensional parameter space (up to several hundred, depending on the science question) with multiple objective functions generates an explosive growth of data characterized by large volume, variety, velocity, and veracity, and presents a paramount need for effectively and efficiently steering the "tuning" of parameters [7], [8]. The challenges for such steering processes mainly arise from numerous model options, vast parameter spaces, complex inter-parameter coupling effects, and almost infinite continuous value ranges. In some sense, the tasks such as selecting the best model, identifying critical parameters, and setting appropriate values in the enormous model and parameter space are just as challenging as looking for a needle in a haystack.

We would also like to point out that many computational science applications such as supernova and combustion research follow a similar modeling-computing paradigm as in climate and weather research. The large model-parameter space in the simulation, the colossal amount of generated data, the intensive computing needed for data processing and analysis, the high complexity of user interactions, and the increasing dynamics of computer and network systems make it practically infeasible for domain experts to manually deploy and optimize such scientific workflows for predictive modeling and computational steering. Especially, many scientists are isolated from each other in using computing facilities and accessing simulation or experimental data produced by individual modeling processes or scientific instruments, inevitably leading to high cost and low efficiency in replicating results and sharing information and resources, which significantly impede the pace of knowledge discovery and constrain the productivity of scientists.

In this work, we aim to provide the capability of collaborative computational steering (Co^2Steer) as a service to automate and accelerate parameter tuning of scientific simulations. We develop a Co^2Steer framework, which employs Bayesian Optimization to guide the parameter tuning process and identify the best parameter values for guaranteed quality of simulations. Our contributions to the field are summarized as follows:

- We design a framework of Co^2Steer for steering as a service with generic models of simulations and extensible Application Programming Interfaces (APIs) to interact with and guide model-based simulations towards user-preferred directions.
- We design a data-driven approach using Bayesian Optimization to perform incremental parameter tuning and identify optimal model configurations quickly to accelerate scientific simulations.
- We theoretically analyze the steering performance of Co^2Steer, and test it with real-life climate modeling workloads to illustrate its performance superiority compared with the default setting provided by domain experts.

The rest of the paper is organized as follows. In Section II, we define the computational steering objective, conduct a brief survey of existing work in related fields, and introduce background knowledge of Bayesian Optimization. The design of Co^2Steer is detailed in Section III. In Section IV, we conduct theoretical analysis of the proposed steering approach. In Section V, we present a case study of computational steering with a real-life scientific application and conduct a comparative performance evaluation of our framework. We conclude our work and sketch a research plan in Section VI.

II. Preliminaries

A. Steering Objectives and Effects of Hyperparameters

In scientific simulations, there are three main goals of computational steering: performance optimization, algorithm experimentation, and model exploration. In performance optimization, steering is used to improve an application's performance, e.g., by balancing workload in parallel applications. In algorithm experimentation, it allows the user to adapt program algorithms at run time, e.g., to experiment with different numerical solving methods. In this work, we focus on model exploration, where computational steering is used to explore parameter spaces and simulation processes to gain additional insights into the model behaviors.

Without loss of generality, we define the steering objective of model exploration as

$$\underset{\vec{x}}{\operatorname{argmin}}\, f(\mathcal{S}(\vec{x}), obs), \tag{1}$$

where $\mathcal{S}$ is a model of steering with tunable parameters $\vec{x}$ of interest and f is a user-specified objective function, e.g., the Mean Square Error (MSE) between the output (simulation data) of the steering model and the observation data obs.

To illustrate the effects of hyperparameters on the steering objective f, we simulate the WRF-Solar model with various hyperparameter settings, and then fit the simulation trails and plot the response surface of MSE using Gaussian Regression Model.

As shown in Fig 2, the response surface of f is significantly affected by the intricate interplay of tunable parameters such as relative dispersion and condensation rate. Although one ideal

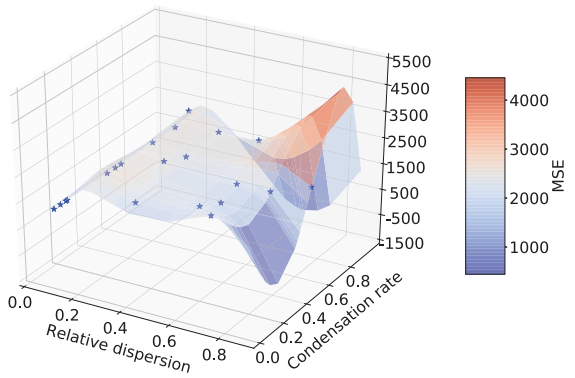

Fig. 2. Response surface of Mean Squared Error fitted by Gaussian regression.

approach would be to check all possible parameter settings, it is practically infeasible to do so because the number of possible parameter settings in a given model-based simulation typically grows exponentially with the number of parameters. For example, if a simulation involves the interaction of n binary parameters (that is, each has one of the two states), this leads to checking and understanding 2^n possible parameter settings with respect to the underlying goal [9]–[11]. An additional level of complexity arises from the underlying model that dictates how these parameters contribute to the goal. In reality, most parameters are continuous and represent an even much larger number of states or values than binary ones.

B. Existing Computational Steering Systems

Computational steering enables users to interact online with simulations or computing procedures, and its importance has been well recognized by the research community [12]–[21]. There exist a plethora of systems and frameworks that support computational steering. Examples include SCIRun [22], CUMULVS [23], VIPER [24], ParaView Catalyst Live [25], GRASPARC [26], Cactus [27], RealityGrid [28], RICSA [29], and many others [16]–[18], [30], [31]. Similar to our approach, these systems also require code instrumentation through libraries or API calls, as commonly adopted in computational sciences [13], [32]. However, most of them may suffer from a high learning curve and require the installation of various software packages such as Globus, SOAP, PVM [33] and, AVS [34] on the user end to realize their full benefits, which often places an undue burden on end users, who are typically domain experts, to spend a significant amount of time in configuring and learning a new system. Furthermore, some of them are platform dependent and hence are not universally supported in diverse environments. Also, to the best of our knowledge, these systems currently do not perform the tracking of steering actions.

There also exist several application-specific steering solutions, for example, in the oil and gas industry [35] and in the study of computational fluid dynamics [36]. One representative system is BSIT [37], which is tailored for seismic applications to support adaptations in parameters, programs, and datasets. The steering capability in such systems is built upon a fixed scheme and is generally not adaptable or extendable to other applications.

In addition to steering commonly applied to computations, some workflow management systems (WMS) also provide such capability to computing modules, including [18]–[21], [38]. Pegasus [38] is a widely used WMS that provides a database with execution data for debugging. As an extension to Pegasus, Lee *et al.* [18] enabled adaptive execution of scientific workflows through the analysis of this database. OpenMole [19] is another WMS that allows users to replace programs in the workflow at runtime. FireWorks WMS [20] uses a DBMS-driven workflow engine and takes a JSON-based approach for state management and workflow execution monitoring, but it does not support other steering actions. Copernicus WMS [21] also allows for dynamic workflow steering via parameter tuning to steer exploration towards undiscovered regions of a solution space. WorkWays [15], which uses Nimrod/K, an extension of the Kepler workflow system [39], as its workflow engine, enables users to dynamically adapt the workflow by reducing the range of some parameters. However, these WMS do not consider the dynamics caused by various steering actions in job scheduling and resource allocation in big data systems deployed on clouds.

Different from the aforementioned work, we propose Steering as a Service that provides the capability of automatic parameter tuning based on machine learning techniques.

C. Bayesian Optimization for Computational Steering

Bayesian Optimization (BO) is a powerful data-driven technique for solving hyperparameter optimization problems and has been extensively studied in the field of automated machine learning [40], [41]. Under the standard settings of BO, the steering objective y is denoted as a function f of a vector of hyperparameters $\vec{x}$, i.e., $y = f(\vec{x})$. The steering problem is then transformed into an optimization problem, i.e., $\text{argmin}_x f(\vec{x})$.

BO first makes an assumption of f with an approximate function/model $\mathcal{M}$, i.e., $\mathcal{M}(\vec{x}) \approx f(\vec{x})$, and then solves this problem in an iterative way: in each iteration, it queries f using the contemporary decision $\tilde{x}$ by optimizing an acquisition function derived from the surrogate model $\mathcal{M}$ and maintains a dataset D consisting of query pairs $\{\vec{x}, f(\vec{x})\}$. It then updates $\mathcal{M}$ using the updated historical dataset D.

In the view of Bayesian, given a dataset of n steering trails $D = \{(\vec{x}_i, y_i)|i = 1, ..., n\}$, the posterior probability of the assumption f is calculated as: $P(f|D) = \frac{P(D|f)P(f)}{P(D)}$. As $P(D)$ is irrelevant to f, the posterior is proportional to the likelihood of steering trails D, i.e., $P(f|D) \propto P(D|f)P(f)$. Placing a Gaussian Process (GP) prior [42] on f, the distribution of f is assumed to be a GP, i.e., $f(\vec{x}) \sim GP(\mu(\vec{x}), k(\vec{x}, \vec{x}'))$, where $\mu(\cdot)$ and $k(\cdot)$ are the mean function and kernel function of GP,

respectively. In each iteration of GP-based BO, the posterior of f for a new configuration setting $\vec{x}'$ is described as:

$$P(f(\vec{x}')|D, \vec{x}') = \mathcal{N}(\mu(\vec{x}'), \sigma^2(\vec{x}')), \tag{2}$$

where

$$\mu(\vec{x}') = \mathbf{k}^T\mathbf{K}^{-1}\mathbf{Y}, \tag{3}$$

$$\sigma^2(\vec{x}') = k(\vec{x}', \vec{x}') - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}, \tag{4}$$

where

$$\mathbf{K} = \begin{bmatrix} k(\vec{x}_1, \vec{x}_1) & \cdots & k(\vec{x}_1, \vec{x}_n) \\ \vdots & \ddots & \vdots \\ k(\vec{x}_n, \vec{x}_1) & \cdots & k(\vec{x}_n, \vec{x}_n) \end{bmatrix}, \tag{5}$$

$$\mathbf{k} = [k(\vec{x}', \vec{x}_1) \quad k(\vec{x}', \vec{x}_2) \quad \cdots \quad k(\vec{x}', \vec{x}_n)], \tag{6}$$

and $\mathbf{Y}$ is the vector representation of $\{y_i\}_{i=1}^n$.

Eqs. 3 and 4 are then plugged into an acquisition function to generate the next query point that yields the extremum. One of the widely used acquisition functions is the expected improvement (EI) function, i.e.,

$$EI(\tilde{x}) = \begin{cases} (\mu(\vec{x}') - f(\vec{x}^+))\Phi(Z) + \ \sigma(\vec{x})\phi(Z), & \text{if } \sigma(\vec{x}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where

$$Z = \frac{\mu(\vec{x}) - f(\vec{x}^+)}{\sigma(\vec{x})}, \tag{8}$$

where $f(\vec{x}^+)$ is the contemporary extremum in D, and $\phi(\cdot)$ and $\Phi(\cdot)$ are the PDF and CDF of the normal distribution, respectively.

III. A Machine learning-assisted Framework to Optimize Steering-driven scientific simulation

To bring optimization-guided *autonomy* to the generic steering process in various scientific applications, we propose a framework of Bayesian Optimization-assisted Steering as a Service for collaborative computational steering, Co^2Steer, in support of large-scale simulation-based computational sciences.

As shown in Fig. 3, the overarching framework of Co^2Steer integrates the following technical components: i) transport method for steering, ii) machine learning-based automatic parameter tuning, iii) front-end dashboard, and vi) provenance tracking. The web-based dashboard provides a user interface through a web browser for users to access Co^2Steer, which is hosted as a service. A scientific workflow that constitutes a model-based simulation process and various post-data processing jobs are executed in a specific computing system designated by the user. The interactions between the steering engine and the scientific workflow are carried out over a unified communication channel that enables the delivery of various steering commands from multiple users to different simulation processes being executed concurrently. The output data are analyzed for model validation with visual feedback provided to the user through the dashboard. The entire steering process, and model validation are managed, tracked and recorded in a provenance database.

A. Steering with Bayesian Optimization

Scientific simulations often encompass a large parameter space that constitutes various of computation-intensive processes. The computational steering problem is to determine a specific configuration of model parameters to produce a desired output with a limited number of trails.

Our approach consists of two steps: i) build a prior belief model (i.e., a Gaussian regression model $\mathcal{M}$) to describe the relationship between the parameters $\vec{x}$ and the steering objective y using existing historical trails, and ii) automate the steering process based on the underlying model $\mathcal{M}$ and expert guidance. The second step continues in an iterative manner until the goal is achieved with a certain error tolerance. Our approach could be broadly classified as fully automated steering with supervised algorithms and automated steering with hybrid algorithms. These solutions are objective-driven and follow the aforementioned two-step approach. We design the tuning process as an iterative stationary process and propose to use Gaussian process to explore the unknown mapping f that captures the interplay between the parameters and the objective.

1) Fully Automated Steering with Supervised Algorithms: The steering objective of interest is determined by the configuration of a set of control parameters $\vec{x}$ (e.g., relative dispersion and condensation rate in a weather forecast model) and the observation $f(\cdot)$ is corrupted with independent, identically distributed noise $\epsilon = N(0, \sigma^2)$, i.e., $y = f(\cdot) + \epsilon$. Due to the high computation overhead for simulating a large number of complex scientific processes, the evaluation of each configuration could be prohibitively time-consuming. Even with common model complexity and parameter space, it may oftentimes take half a day to complete.

The second step of our iterative approach selects the next configuration of the parameters for validation. Hence, it is important to exploit the historical validations and explore the unknown configurations. BO offers desired properties to balance exploration and exploitation. In each iteration, BO fits the existing dataset using a machine learning model $\mathcal{M}$ (e.g., Gaussian Process Regression), selects the next query point by maximizing an acquisition function, and updates the dataset. The implementation details are shown in Alg. 1.

Algorithm 1 Bayesian Optimization

Input: surrogate model $\mathcal{M}$, simulation model $\mathbb{S}$, historical dataset $\mathcal{D}$ and acquisition function $\mathcal{L}$
Output: the best configuration $\vec{x}^*$

1: **for** $t_i = 1, 2, \ldots, T$ **do**
2: fit $\mathcal{M}$ with $\mathcal{D}$;
3: obtain the next query point $\vec{x}^{'}$ = $\text{argmax}_x\, \mathcal{L}$;
4: simulate with a new configuration: $y^{'} = \mathcal{S}(\vec{x}^{'})$;
5: consolidate the data: $D = D \cup \{\vec{x}^{'}, y^{'}\}$;
6: query $\mathcal{D}$ to obtain the best configuration $\vec{x}^*$ that results in the minimum y;
7: **return** $\vec{x}^*$;

2) Semi Automated Steering: The use of automated methods does not, however, obviate the need for subjective judg-

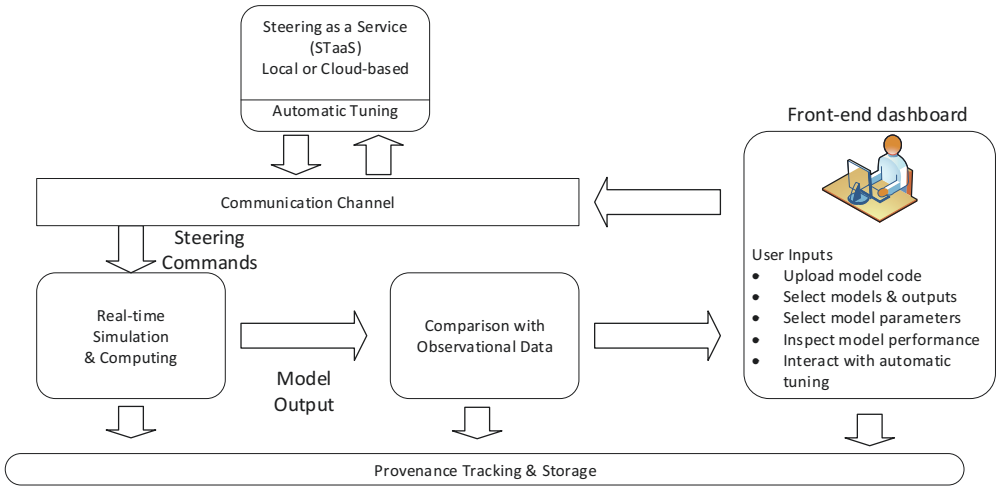

Fig. 3. Architecture of the Co^2Steer framework.

ment concerning the priorities and targets of the steering process. Moreover, BO is shown to be overconfident in searching some unexplored boundary region and may lead to unnecessary cost.

To address the overconfidence problem, we propose a hybrid approach to involve humans in the steering loop, but only in a strategic manner. We present the dynamic steering procedure to domain experts, who could provide a wide range of feedbacks - from simple binary feedbacks (e.g., labeling a parameter as important vs. unimportant) to non-binary feedbacks (e.g., iteratively shrinking the search boundary), or a combination of both. Based on such feedbacks, we update model M and repeat the overall computational loop until certain accuracy is achieved.

B. API Design and Provenance Tracking

One essential function of Co^2Steer is to enable computational steering by a group of remote collaborative users who wish to keep track of the simulation process SIM and communicate and share simulation/analysis results DS_{out} or DS_{final} with peers while the simulation is being steered in batch mode. To make this possible, the simulation steering can no longer remain closed and needs to open up channels to intercept and distinguish steering commands from different users at runtime. Towards this goal, we define the syntax and semantics of a set of generic core APIs, and provide steering capability through automatic mode: Users upload their codes through the dashboard and Co^2Steer identifies the locations in the codes for taking appropriate steering actions at the entry or exit of a loop that implements the body of a simulation process.

Fig. 4 illustrates the code skeleton of typical model-based simulation programs that call a set of essential API functions for computational steering by multiple users simultaneously. Among them, $Co2Steer\_init()$ initializes the steering process by subscribing to the communication channel and registering with the steering service and provenance database. A steering action SA is captured by $Co2Steer\_recvMsg()$ in the beginning of each iteration to update the parameter setting used in the simulation process SIM. All changes and corresponding results are recorded in the provenance database and also sent back to a group of participating users for analysis and examination.

Provenance is a key part of the architecture and service in the common computing infrastructure for tracking processes and analyzing results. Particularly, in a model-based simulation process, it is critical to keep track of all configuration options, model versions, parameter choices, etc., all of which have an impact on the outcome of the model simulation. Such provenance information provides a complete view of the derivation process from original sources to final results, and enables scientists to verify the correctness of their simulations and reproduce them if necessary. We design a provenance component using script and integrate it into the Co^2Steer steering engine to provide complete provenance information related to the execution of the simulation for post-data processing and analysis. The provenance component automatically records all the information about the simulation process such as the execution time and parameter settings, and tracks the history and evolution of all trials.

IV. Convergence Rate Analysis

The steering objective of our Bayesian Optimization-based framework for large-scale scientific simulations is to find the optimal hyperparameter setting $\vec{x}'$ with the least number of iterations such that the error between simulation result $\mathcal{S}(\vec{x}')$ is arbitrarily close to the observation obs, i.e.,

$$f(\mathcal{S}(\vec{x}'), obs) - f(\mathcal{S}(\vec{x}^*), obs) < \delta, \tag{9}$$

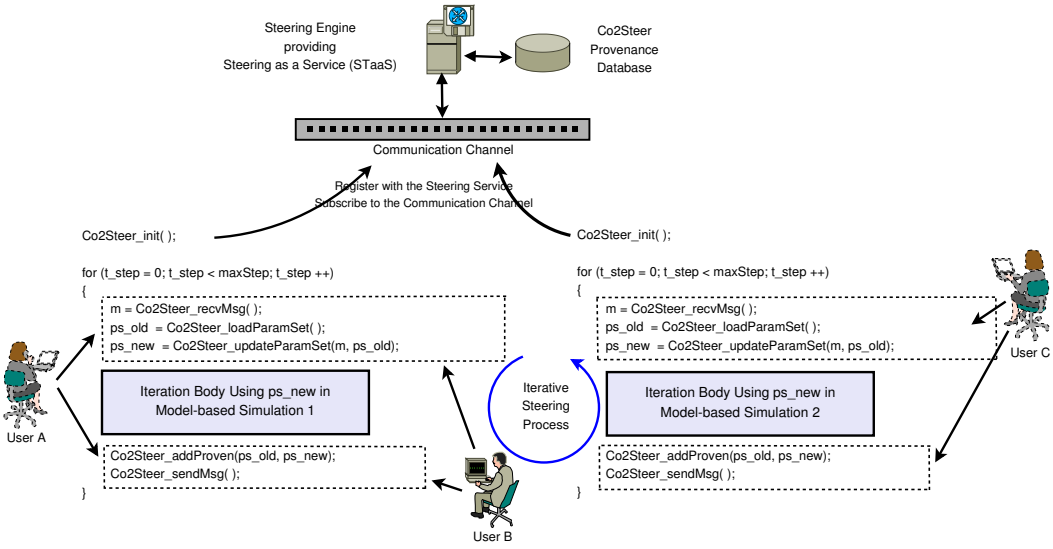

Fig. 4. A code skeleton of typical model-based simulations that make steering API calls for computational steering engine and communication channel.

where $\vec{x}'$ is the best guess of our optimization method, $\vec{x}^*$ is the unknown global optimal setting of $\mathcal{S}$, and δ is a positive constant.

To determine the convergence speed of computational steering, we need to estimate the total number of iterations required to achieves δ accuracy. Moreover, the general global optimization problem is theoretically intractable without making any assumptions to function f [43]. Without loss of generality, we consider f to be Lipschitz continuous, i.e., the simulation error $f(\cdot)$ cannot vary arbitrarily fast as we change $\vec{x}$. The convergence rate of our approach is dominated by the theoretical bound of the Bayesian Optimization technique, which contains two intertwined components: the surrogate component for exploitation and the acquisition component for exploration. Our approach follows the standard setting, where the Gaussian process regression model is selected as the surrogate model and the expected improvement (EI) function is used as the acquisition model [44]. The convergence speed of BO has been widely investigated [45], [46]. As stated by Bull in [45], under certain hypothesis, the expected improvement-based BO is shown to converge to the minimum of any function on its Reproducing-Kernel Hilbert Space (RKHS) with rate $\mathcal{O}(\frac{1}{\delta^{v/d}})$, where v is a smooth measure of f and d is the number of parameters to optimize over. Note that the smooth measure v of f is assumed to be greater than one to make BO better than random guesses. The time complexity of BO is $\mathcal{O}(n^3)$, where n is the number of historical trails, because BO solves Cholesky decomposition problem for each iteration.

V. Validation with Real-World Applications

We develop a prototyped Co^2Steer service that enables, optimizes, and tracks steering-driven simulation-oriented scientific workflows for model exploration and evaluation. This prototype steering service is deployed on a virtual machine (VM) instance equipped with eight processors and 20GB memory. We test Co^2Steer for model-based simulation in a real-life large-scale scientific application, i.e., a WRF model for climate research.

A. Case Study: Weather Research and Forecasting

The physical forecasting model of WRF-Solar is built on the widely used community Weather Research and Forecast (WRF) model [47] with an emphasis on forecasting solar and wind energies, and contains numerous parameterized subgrid processes that affect model performance, including cloud and aerosol microphysics, radiative transfer, planetary boundary layer, turbulence, convection, and land surface. The tunable parameters involved in the model are shown in Table I. We first introduce the interested parameters, and then illustrate the tuning effects of such parameters on the simulation quality. We further perform a point-wise comparison between our work and default settings and verify the convergence speed of our solution.

1) Description of Physical Parameterizations and Important Tunable Parameters: The WRF-Solar model is implemented based on Thomas scheme [48], which contains various sub-processes that simulate aerosol process, cloud droplet, liquid water, etc,.

Particularly, in this case study, we investigate two parameters, namely, relative dispersion and condensation rate, which are accessible and tunable in representative processes that simulate the cloud-to-rain autoconversion and effective radius in liquid water clouds. The final output of this WRF model emulates the evolution of solar irradiance volume in 90 days.

The effective radius r_e is simulated based on a Gamma distribution, which contains two degrees of freedom, i.e., shape parameter and slope parameter.

TABLE I
LIST OF MODEL PARAMETERS

Symbol	Description	Default Value	Range	Location/Variable Name
ρ_s	Density of snow	100 kg/m^3	50-200 kg/m^3	*rho_s* Line 88
ρ_g	Density of graupel	500 kg/m^3	[450, 700] kg/m^3	*rho_g* Line 89
ρ_i	Density of ice	890 kg/m^3	[700, 900] kg/m^3	*rho_i* Line 90
a	Mass power-law constant	0.069	0.0185-0.176	*am_s* Line 139
b	Fall speed power-law constant	2	[1.9, 2.2]	*bm_s* Line 140
α	Fall speed power-law constant	Rain: 4854 Ice:1847.5 Snow:40 Graupel:442	0.15 336-1847.5 129.6-40 351.2-442	R Line 149 I Line 157 S I
β	Fall speed power-law constant	Rain: 1 Ice:1 Snow:0.55 Graupel:0.89	Fixed 0.6635 - 1 0.42- 0.55 0.37 - 0.89	R Line 149 bv_i Line 157 bv_s bv_g
f	Fall speed power-law constant	Rain: 195 Ice: 0 Snow: 125 Graupel: 0	Fixed Fixed 100-125 Fixed	fv_s Line 154
C	Capacitance of hydrometeor	Sphere: 0.5 Plates/aggregates: 0.15	15% 15%	C_cube C_sqrd
E_{yx}	Collection efficiencies	si: 0.05 rs: 0.95 rg: 0.75 ri: 0.95	15% within 0 - 1 1 15% within 0 - 1 15% within 0 - 1	Ef_si Ef_rs Ef_rg Ef_ri
D_0	Lower limit of hydrometeor diameter	Cloud:$1E^{-6}$ Rain: $50E^{-6}$ Snow: $200E^{-6}$, m Graupel:$250E^{-6}$, m	C: [$0.5E^{-6}$, $2E^{-6}$] R: [$50E^{-6}$, $100E^{-6}$] S: [$150E^{-6}$, $250E^{-6}$] [$200E^{-6}$, $300E^{-6}$]	Line 220 Line 221 Line 222 Line 223
β_{con}	Condensation rate constant	$1.15E^{23}$	[$1.02E^{20}$, $1.67E^{24}$]	Line 78
ϵ	Relative dispersion of cloud droplet spectrum	0.1	0.01 to 1.4	Line 69

The relative dispersion ϵ affects r_e through the product of a dimensionless parameter β and the mean volume radius v, i.e.,

$$r = \beta \cdot v. \tag{10}$$

For cloud droplets, rain, cloud ice and snow, β is calculated as:

$$\beta = \frac{(1 + 2 \times \epsilon^2)^{2/3}}{(1 + \epsilon^2)^{1/3}}, \tag{11}$$

where ϵ is the relative dispersion. In consideration of all unknown effects (e.g., turbulence-related processes), an empirical condensation rate constant β_{con} is defined to emulate the turbulence effect.

2) Tuning Effects on Simulation Quality: The simulation result of the WRF model is a time-series sequence that represents solar irradiance change in 90 days. The tuning effects of the relative dispersion on the volume of solar irradiance are plotted in Fig 5. We observe that the simulation result approaches the observation data as the value of dispersion decreases. However, further decreasing the value of dispersion does not bring performance improvement. This is because some of the simulation processes are skipped due to a negligible value of relative dispersion. We would like to point out that the impact of relative dispersion is complicated, which strongly suggests the use of machine learning algorithms for parameter tuning.

3) Methods in Comparison: We compare BO with another heuristic method using random walk [49]. Instead of exploring the search space as a compact real realm, random walk first

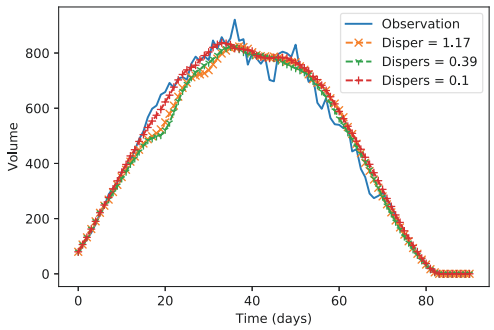

Fig. 5. Illustration of tuning effects of dispersion on simulation quality.

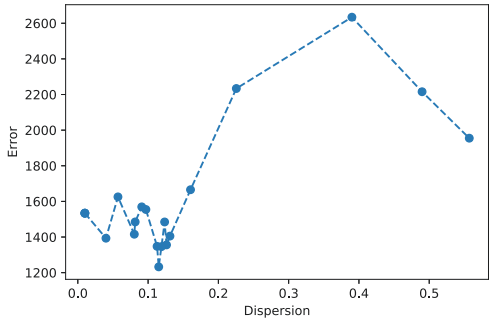

Fig. 6. Illustration of effects of dispersion on the Mean Squared Error.

transforms the searching space into grids, and then iteratively explores the search space by moving a fixed length at a randomly generated angle. We also compare BO and random walk with the default setting to show the performance improvement over manual tuning recommended by domain experts.

4) Evaluation Metrics: The execution of Co^2Steer in climate research involves both model-based simulations and real-world observations. To evaluate the effectiveness of tuning, we calculate Mean Squared Error (MSE), and consider average cumulative regret that measures the convergence rate of steering. In each iteration of online steering, the instantaneous "regret" at the i-th iteration is defined as the distance from the current evaluated value $f(\mathcal{S}(\vec{x}_i), obs)$ to the optima $f(\mathcal{S}(\vec{x}^*), obs)$, i.e.,

$$r = f(\mathcal{S}(\vec{x}_i), obs) - f(\mathcal{S}(\vec{x}^*), obs). \quad (12)$$

The average accumulative regret over t time-steps is calculated as $\frac{1}{t}\sum_i r_i$.

5) Experimental Setting: As BO-based tuning relies on historical trails, we randomly generate two data points to initialize the tuning process. For random walk, we select the center point of the search area as the start point and set the step size to be $\pm$ 0.014 for relative dispersion and $\pm$ $1.67E^{22}$ for condensation rate, respectively. In each iteration, the random walk algorithm moves a single mosquito step for each dimension with equal probability.

6) Results: The performance evaluation includes two parts: i) evaluate the effectiveness of our approach in terms of MSE, and ii) evaluate the efficiency of our approach in terms of average accumulative regret.

We first plot the smallest MSE among historical tuning trails in terms of iterations, as shown in Fig. 7. The performance of the default setting forms a straight line and can be used as the baseline. The random walk algorithm performs poorly due to the limitation of iterations. The simulation error of our approach drastically decreases with more iterations of steering, and achieves a plateau much lower than random walk and the default setting recommended by domain experts.

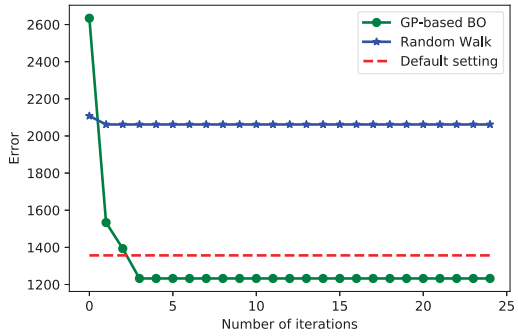

Fig. 7. Performance comparison in terms of MSE.

We further compare the convergence rate of our approach with that of random walk in terms of average accumulative regret. Fig. 8 shows that our approach converges faster than random walk.

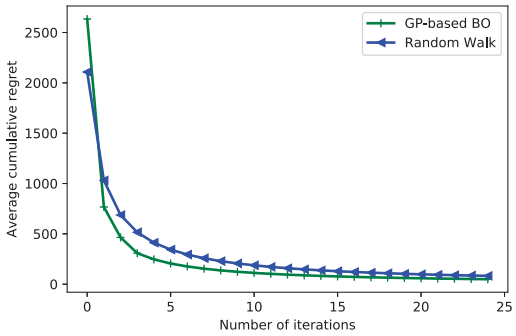

Fig. 8. Convergence rate comparison in terms of average accumulative regret.

VI. Conclusion

In this paper, we developed a prototype steering as a service auto-tuning framework, which consists of a set of APIs for domain experts to execute, monitor, and interact with model-based simulations. We conducted extensive experiments to evaluate the efficiency and effectiveness of our parameter tuning method with a real-life WRF model.

We plan to integrate other optimization methods into this steering framework to address different hyperparameter tuning problems. For scalability evaluation, we will integrate and vary the scale of computing systems, the complexity of big data workflow structures, the number of steerable parameters, and the range of parameter values, and measure the corresponding end-to-end workflow performance, execution time of automated steering algorithms, and other system-specific optimization objectives such as speedup, efficiency, and workload balancing.

Acknowledgments

This research is sponsored by U.S. National Science Foundation under Grant No. CNS-1828123 with New Jersey Institute of Technology.

References

[1] DOE. Report of the DOE Office of Science Workshop, 2011.
[2] DOE. Report of the DOE Office of Science Data-Management Workshop, 2004.
[3] NSF. http://www2.evl.uic.edu/NSF/index.html, 2001.
[4] A. Mezzacappa. SciDAC 2005: scientific discovery through advanced computing. *J. of Physics: Conf. Series*, 16, 2005.
[5] Y. Liu. Introduction to the special section on fast physics in climate models: Parameterization, evaluation, and observation. *Journal of Geophysical Research: Atmospheres*, 124(15):8631–8644, 2019.
[6] P. Jimenez and J. Hacker and J. Dudhia and S. Haupt and J. Ruiz-Arias and C. Gueymard and G. Thompson and T. Eidhammer and A. Deng. WRF-Solar: Description and Clear-Sky Assessment of an Augmented NWP Model for Solar Power Prediction. *Bulletin of the American Meteorological Society*, 97(7):1249–1264, 2016.
[7] L. Wu and T. Zhang and Y. Qin and W. Xue. An effective parameter optimization with radiation balance constraint in cam5 (version 5.3). *Geoscientific Model Development*, 13(1):41–53, 2020.

[8] T. Zhang and M. Zhang and W. Lin and Y. Lin and W. Xue and H. Yu and J. He and X. Xin and H. Ma and S. Xie and W. Zheng. Automatic tuning of the community atmospheric model (cam5) by using short-term hindcasts with an improved downhill simplex optimization method. *Geoscientific Model Development*, 11(12):5189–5201, 2018.

[9] R. Brownlie and J. Prowse and M. S. Phadke. Robust testing of at&t pmx/starmail using oats. *Bell Labs Technical Journal*, 71(3):41–47, 1992.

[10] M. B. Cohen and P. B. Gibbons and W. B. Mugridge and C. J. Colbourn. Constructing test suites for interaction testing. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 38–48. IEEE, 2003.

[11] C. Yilmaz and M. B .Cohen and A. A. Porter. Covering arrays for efficient fault characterization in complex configuration spaces. *IEEE Transactions on Software Engineering*, 32(1):20–34, 2006.

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

[13] A. C. Bauer and H. Abbasi and J. Ahrens and H. Childs and B. Geveci and S. Klasky and K. Moreland and P. O'Leary and V. Vishwanath and B. Whitlock and E. W. Bethel. In situ methods, infrastructures, and applications on high performance computing platforms. In *Computer Graphics Forum*, 2016.

[14] R. Souza and M. Mattoso. Provenance of dynamic adaptations in user-steered dataflows. In *International Provenance and Annotation Workshop*, pages 16–29, 2018.

[15] H. A. Nguyen and D. Abramson and T. Kipouros and A. Janke and G. Galloway. Workways: interacting with scientific workflows. *Concurrency and Computation: Practice and Experience, Special Issue: Workflow in Grid Systems*, 27(16):4377–4397, 2015.

[16] J. D. Mulder and J. J. V. Wijk and R. V. Liere. A survey of computational steering environments. *Future generation computer systems*, 15(1):503–523, 1999.

[17] R. V. Liere and J. D. Mulder and J. J. V. Wijk. Computational steering. *Future generation computer systems*, 12(5):441–450, 1997.

[18] K. Lee and N. W. Paton and R. Sakellariou and AA. A. Fernandes. Utility functions for adaptively executing concurrent workflows. *Concurrency and Computation: Practice and Experience*, 23(6):646–666, 2011.

[19] R. Reuillon and M. Leclaire and S. R. Coyrehourcq. Openmole, a workflow engine specifically tailored for the distributed exploration of simulation models. *Future Generation Computer Systems*, 29(8):1981–1990, 2013.

[20] A. Jain and S. P. Ong and W. Chen and B. Medasani and X. Qu and M. Kocher and M. Brafman and G. Petretto and G. Rignanese and G. Hautier D. Gunter and K. A. Persson. Fireworks: a dynamic workflow system designed for high-throughput applications. *Concurrency and Computation: Practice and Experience*, 27(17):5037–5059, 2015.

[21] I. Pouya and S. Pronk and M. Lundborg and E. Lindahl. Copernicus, a hybrid dataflow and peer-to-peer scientific computing platform for efficient large-scale ensemble sampling. *Future Generation Computer Systems*, 71(8):18–31, 2017.

[22] S. G. Parker and C. R. Johnson. Scirun: a scientific programming environment for computational steering. In *Proc. of Supercomputing Conference*, page 52, 1995.

[23] J. A. Kohl and T. Wilde and D. E. Bernholdt. Cumulvs: Interacting with high-performance scientific simulations, for visualization, steering and fault tolerance. *Int. J. of High Performance Computing Applications*, 20(2):255–285, 2006.

[24] M. Oberhuber and S. Rathmayer and A. Bode. Tuning parallel programs with computational steering and controlled execution. In *Proc. of the Thirty-First Hawaii Int. Conf. on System Sciences*, volume 7, pages 157–166, Kohala Coast, HI, 1998.

[25] U. Ayachit and A. Bauer and B. Geveci and P. O'Leary and K. Moreland and N. Fabian and J. Mauldin. Paraview catalyst: Enabling in situ data analysis and visualization. In *Proc. of Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, pages 25–29, 2015.

[26] K. Brodlie and A. Poon and H. Wright and L. Brankin and G. Banecki and A. Gay. Grasparc: a problem solving environment integrating computation and visualization. In *Proc. of Visualization*, pages 102–109, 1993.

[27] T. Goodale and G. Allen and G. Lanfermann and J. Masso and T. Radke and E. Seidel and J. Shalf. The cactus framework and toolkit: Design and applications. In *International Conference on High Performance Computing for Computational Science*, pages 197–227, 2002.

[28] S. Pickles and R. Haines and R. Pinning and A. Porter. A practical toolkit for computational steering. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 363(1833):1843–1853, 2005.

[29] Q. Wu and M. Zhu and Y. Gu and N. S. V. Rao. System design and algorithmic development for computational steering in distributed environments. *IEEE Trans. on Parallel and Distributed Systems*, 21(4):438–451, April 2010.

[30] S. M. Figueira and S. Bui. Cs-lite: A lightweight computational steering system. In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks*, pages 1–6, 2004.

[31] J. Knezevic and J. Frisch and R. Mundani and E. Rank. Interactive computing framework for engineering applications. *Journal of Computer Science*, 7(5):591, 2011.

[32] V. Silva and R. Souza and J. Camata and D. Oliveira and P. Valduriez and A. L. Coutinho and M. Mattoso. Capturing provenance for runtime data analysis in computational science and engineering applications. In *International Provenance and Annotation Workshop*, pages 183–187, 2018.

[33] PVM. http://www.csm.ornl.gov/pvm.

[34] AVS. http://www.avs.com.

[35] B. Swift and A. Sorensen and H. Gardner and P. Davis and V. K. Decyk. Live programming in scientific simulation. *Supercomputing frontiers and innovations*, 2(4):4–15, 2011.

[36] M. Garca and J. Duque and P. Boulanger and P. Figueroa. Computational steering of cfd simulations using a grid computing environment. *International Journal on Interactive Design and Manufacturing*, 9(3):235–245, 2015.

[37] M. Hanzich and J. Rodriguez and N. Gutierrez and J. Puente and J. Cela. Using hpc software frameworks for developing bsit: a geophysical imaging tool. In *Proceedings of WCCM XI*, pages 2019–2030, 2014.

[38] E. Deelman and G. Singh and M. Su and J. Blythe and A Gil and C. Kesselman and G. Mehta and K. Vahi and G. B. Berriman and J. Good and A. Laity and J. C. Jacob and D. S. Katz. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13:219–237, 2005.

[39] D. Abramson and C. Enticott and I. Altinas. Nimrod/K: towards massively parallel dynamic grid workflows. In *Proc. of Supercomputing Conference*, pages 1–11, 2008.

[40] V. Perrone and H. Shen and M. W. Seeger and C. Archambeau and R. Jenatton. Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[41] D. Salinas and H. Shen and V. Perrone. A quantile-based approach for hyperparameter transfer learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8438–8448. PMLR, 13–18 Jul 2020.

[42] C. E. Rasmussem and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005.

[43] L. C. W. Dixon. Global optima without convexity. Technical report, Numerical Optimisation Centre, 1978.

[44] I. Frazier. A tutorial on Bayesian optimization. Preprint 1807.02811, arXiv, 2018.

[45] A. D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(88):2879–2904, 2011.

[46] F. Berkenkamp and A. P. Schoellig and A. Krause. No-regret bayesian optimization with unknown hyperparameters. *Journal of Machine Learning Research*, 20(50):1–24, 2019.

[47] WRF. https://www.mmm.ucar.edu/weather-research-and-forecasting-model, 2020.

[48] G. Thompson and R. M. Rasmussen and K. Manning. Explicit forecasts of winter precipitation using an improved bulk microphysics scheme. part i: Description and sensitivity analysis. *Monthly Weather Review*, 132(2):519 – 542, 01 Feb. 2004.

[49] G. F. Lawler and V. Limic. *Random Walk: A Modern Introduction*. Cambridge University Press, 2010.