Scalable Epidemiological Workflows to Support COVID-19 Planning and Response

Dustin Machi*, Parantapa Bhattacharya*, Stefan Hoops*, Jiangzhuo Chen*, Henning Mortveit*¶, Srinivasan Venkatramanan*, Bryan Lewis*, Mandy Wilson*, Arindam Fadikar†, Tom Maiden‡, Christopher L. Barrett*§ and Madhav V. Marathe*§
* *Biocomplexity Institute and Initiative, University of Virginia;* † *Argonne National Laboratory;*
‡ *Pittsburgh Supercomputing Center;* § *Department of Computer Science, University of Virginia;*
¶ *Department of Engineering Systems and Environment, University of Virginia*

2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS) | 978-1-6654-4066-0/21/$31.00 USG | DOI: 10.1109/IPDPS49936.2021.00072

***Abstract*—The COVID-19 global outbreak represents the most significant epidemic event since the 1918 influenza pandemic. Simulations have played a crucial role in supporting COVID-19 planning and response efforts. Developing scalable workflows to provide policymakers quick responses to important questions pertaining to logistics, resource allocation, epidemic forecasts and intervention analysis remains a challenging computational problem. In this work, we present scalable high performance computing-enabled workflows for COVID-19 pandemic planning and response. The scalability of our methodology allows us to run fine-grained simulations daily, and to generate county-level forecasts and other counterfactual analysis for each of the 50 states (and DC), 3140 counties across the USA. Our workflows use a hybrid cloud/cluster system utilizing a combination of local and remote cluster computing facilities, and using over 20,000 CPU cores running for 6–9 hours every day to meet this objective. Our state (Virginia), state hospital network, our university, the DOD and the CDC use our models to guide their COVID-19 planning and response efforts. We began executing these pipelines March 25, 2020, and have delivered and briefed weekly updates to these stakeholders for over 30 weeks without interruption.**

***Keywords*-COVID-19, Epidemic Modeling, HPC Workflow Development**

I. Introduction

COVID-19 represents the first pandemic since the 2009 H1N1 outbreak and is the worst pandemic on record since the 1918 pandemic. Since February 2020, the pandemic has had a severe economic, social, and health impact. According to the International Monetary Fund (IMF), the global economic burden for COVID-19 will likely be 9+ trillion US dollars. More than 113 million confirmed infections and 2.5 million deaths have been reported globally, with very different epidemic dynamic trajectories and mortality witnessed across various countries. Europe and the United States (US) are seeing a resurgence of cases and the situation is unlikely to get better anytime soon.

Epidemiological models and workflows comprising of these models can help provide insight into the spatiotemporal dynamics of epidemics by: (i) forecasting the epidemic's future course, (ii) guiding allocation of scarce resources and assessing depletion of current resources, (iii) inferring disease parameters that allow researchers to make better recommendations and (iv) providing insight into the effectiveness of different interventions. Individual behavior and public policies are critical influencers for controlling epidemics, and computational simulations can be powerful tools for understanding which behaviors and policies are likely to be effective. Our studies have used meta-population models, as well as detailed agent-based models. The network-based models consider epidemic spread on an undirected social interaction network $G(V, E)$ over a population V, where each edge $e = (u, v) \in E$ implies that individuals (also referred to as nodes) $u, v \in V$ interact [12], [26][1].

Our contributions and significance. In this paper, we describe a novel high performance computing (HPC) approach for executing epidemiological workflows that can support planning and response to pandemics such as COVID-19. Our approach is unique: (i) it uses detailed agent-based models as well as meta-population models to simulate epidemic dynamics over realistic representations of national-scale social contact networks, (ii) it splits the workflow across two supercomputing clusters due to resource constraints, and (iii) it is used to support near real-time response efforts. The workflows are comprised of a complex series of data ingestion, simulation and analytics steps. Details of how EpiHiper, the agent-based discrete time simulator for infectious disease spread used in this work, and other such networked agent-based modeling frameworks work are described in companion publications and are not the focus of this paper. However, the basic approach presented here can be used for other agent models and other synthetic social contact networks. We focus here on three epidemic workflows: (i) calibration of the models using county-level incidence data, (ii) predicting daily county-level incidence values for time periods covering two weeks to a few months and (iii) counter-factual analysis of various policy decisions during the ongoing pandemic. Key steps in all of the workflows include (i) a data-driven algorithm that integrates county-level incidence data, as well as individual behavioral representations and public policies, to calibrate the models and project incidence going forward; (ii) realistic individual-level social contact networks and HPC agent-based models

[1] An extended version of this paper is available at https://www.medrxiv.org/content/10.1101/2021.02.23.21252325v1

DOI 10.1109/IPDPS49936.2021.00072

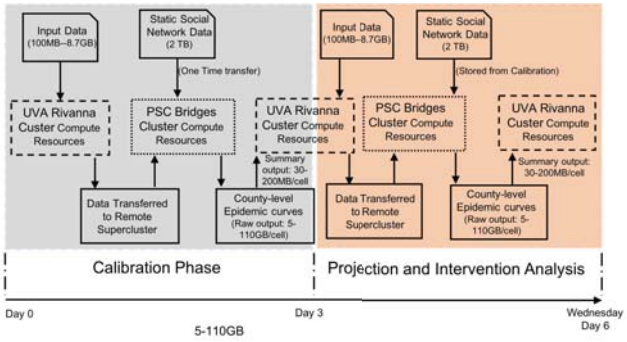

Figure 1. Combined workflow: This diagram illustrates the complete timeline of our process, from model configuration through intervention analysis.

to produce highly resolved outcomes (at the individual and family levels); and (iii) analytics that combine the simulation output, surveillance data and detailed synthetic data to support policy assessment. The workflows are executed in real time, meaning that the pipeline produces epidemic predictions every week that we share with federal and state authorities. Splitting and orchestration of workflows across two supercomputing systems that are geographically separated requires careful analysis of the workload and practical constraints.

We demonstrate our results by showing how our epidemiological workflows can be used to support a national COVID-19 response. Our pipeline typically runs 5,000–17,900 simulations per night, covering the entire US network which is comprised of about 300 million nodes and 7.9 billion edges partitioned across all 50 states and Washington DC. The simulations yield ensemble models for prediction of epidemic incidence curves at the US county level (3140 counties). These results are the first of their kind reported in the literature for national-scale US networks. The workflow is orchestrated between University of Virginia's Rivanna cluster and Pittsburgh Supercomputing Center's Bridges cluster; 20,000 cores of the Bridges cluster are dedicated each night for completing our complex calibration and prediction tasks.

We are the lead modeling group supporting our state's (Virginia) COVID-19 response. We have provided uninterrupted weekly projections and analytical products to the analysts and senior officials of the state hospital referral regions (HRR) and local universities (including our university) since March 25, 2020.[2]. We also provide our weekly forecasts to the Centers for Disease Control and Prevention (CDC), and our analytical products to the Department of Defense (DoD). Our results demonstrate that *real-time, data-driven high resolution epidemics science at a national scale is indeed possible*[3].

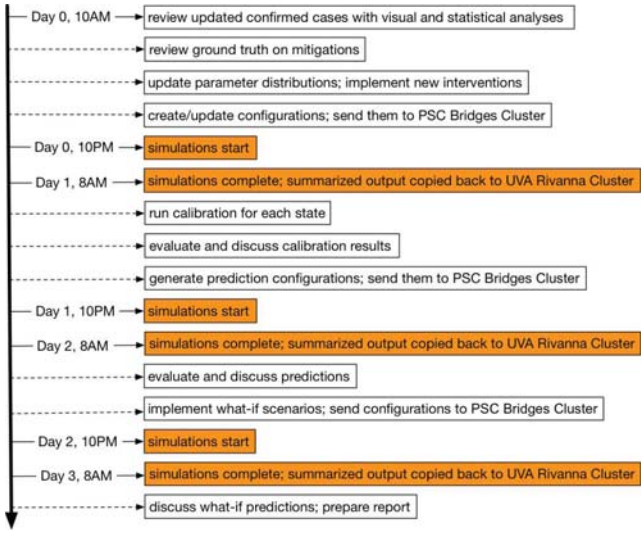

Figure 2. Timeline of tasks involving human efforts. It shows the schedule of the sequence of tasks over multiple days for a complete calibration-prediction cycle. The orange boxes are automated.

Overview. The epidemiological pipeline workflows are shown in Figures 1, 3, 5 and 4.

Each workflow is split between a local cluster (Rivanna) and a remote super-computing cluster (Bridges). Our workflows support our vision of *HPC-oriented, real-time epidemic science*, and are carefully organized to conform to the following set of practical constraints. (i) Our access to the Bridges cluster is limited, so it is not available to us 24/7. We note that the level of access provided to us is very generous — we have had exclusive access to the cluster, with over 20,000 cores, for 10 hours a day (from 10 pm to 8 am) for over 4 months. (ii) Purchased datasets and tools are maintained on Rivanna cluster; these items are not ported to the Bridges cluster due to time and licensing requirements. (iii) Analysts have more consistent access to and control over the Rivanna cluster, so the workflows are split between the two sites. The workflows are also designed to provide a level of resiliency and task parallelization: we use the Rivanna cluster during the day, and use the Bridges cluster at night. Figure 2 shows the timeline of tasks involving human efforts. The overall workflow differs based on the specific kind of problem addressed, but all of them consist of the following significant sub-components: (i) generation of national-scale synthetic social contact networks, (ii) agent-based and meta-population models that can scale to large systems, (iii) methods for calibrating and producing ensemble models, (iv) tools for assembling the input data and distributing this dataset on cluster nodes, and (v) tools

[2]More details on how our models are used can be found at: https://www.vdh.virginia.gov/coronavirus/category/covid-19/model

[3]More information about our work can be found at https://biocomplexity.virginia.edu/project/covid-19-pandemic-response

Workflow	# Cells	# States	# Replicates	# Simulations	Raw Output	Summ. Output
Economic	12	51	15	9180	3.0TB	5.0GB
Prediction	12	51	15	9180	1.0TB	2.5GB
Calibration	300	51	1	15300	5.0TB	4.0GB

Table I
REPRESENTATIVE EXAMPLES OF INDIVIDUAL WORKFLOWS, THEIR SCALE, AND SIZE OF RAW AND SUMMARIZED DATA GENERATED BY THEM.

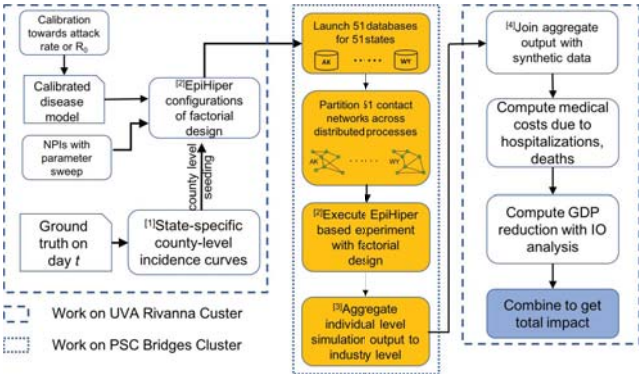

Figure 3. Economic workflow: Economic workflow is used for computing the medical costs incurred due to the pandemic. The models and details of this can be found in [6]. [1] Incidence data includes about 3000 counties × over 200 days of entries. [2] An example factorial design has (2 VHI compliances × 3 lockdown durations × 2 lockdown compliances) × 51 states × 15 replicates = 9180 simulation instances. [3] Size of individual level output data: 12 cells × 51 states × 15 replicates × multi-million state transitions = multi-billion entries, about 3TB. [4] Size of aggregate output data: 12 cells × 51 states × 15 replicates × 365 days × 90 health states × 3 counts = about 1 billion entries, 2.5GB. Size of synthetic data: 300 million × 8 = 2.5 billion entries.

for post-processing the output data so that summary data can be sent back to the home cluster for further analysis.

Each of our workflows represent an interesting mix of data and compute intensive steps and thus crucially need HPC resources. Table I summarizes some of the key numbers for case studies we have described to illustrate the workflows. The partitioning of tasks and specific computation is carefully managed to reduce the amount of data transfer between the two clusters and achieve a near real-time response. Our paper advances the use of parallel and distributed computing in this important area – to the best of our knowledge this is the first time two HPC resources have been used in this manner to support near real-time epidemic planning and response.

II. DESCRIPTION OF THE WORKFLOWS

Figure 1 describes the overall workflow and how it is orchestrated across the two systems. Here we discuss three specific epidemiological workflows that are described in Figures 3, 4 and 5. Figure 2 shows the timeline of tasks involving human effort.

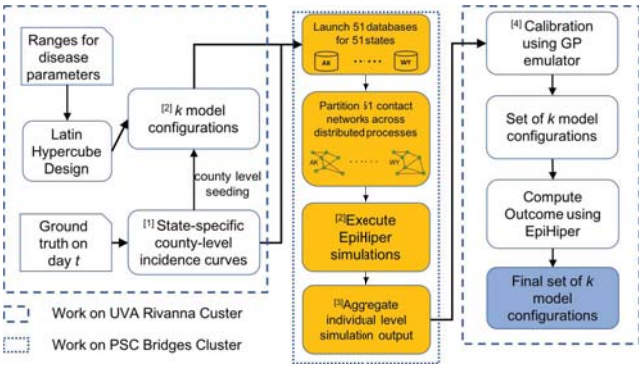

Figure 4. Calibration workflow: [1] Incidence data includes about 3000 counties × over 200 days of entries. [2] An example of calibration design has 300 cells × 51 states × 1 replicates = 15300 simulation instances. [3] Size of individual level output data: 300 cells × 51 states × 1 replicates × multi-million state transitions = multi-billion entries, about 5TB. [4] Calibration uses aggregate data of size: 300 cells × 51 states × 1 replicates × 365 days × 90 health states × 3 counts = about 1.5 billion entries, 4GB.

Counter-factual analysis workflow (Figure 3). Counter-factual analysis refers to the study of outcomes under various posted scenarios. The range of scenarios considered reflect the possible trajectory of the epidemic and is not known in advance. Our counter-factual analysis usually comprises various lockdown policies, compliances, and non-pharmaceutical interventions. The system is calibrated to reflect the current conditions on the ground. Usually such an analysis entails running a large factorial design and then computing certain outcomes that combine the output of the simulations and detailed synthetic social network, demographic and socio-economic data.

Calibration workflow (Figure 4). Calibration refers to finding plausible configuration(s) that produce simulation output similar to observed ground truth. Generally, such parameter searches are carried out by first defining a parameter space consisting of plausible parameter values, then evaluating the *closeness* of the simulation output to the ground truth at various points in that parameter space. However, when running the simulation is expensive, an emulator can be used in place of the actual simulation inside a calibration loop. An emulator is a statistical model that maps the input to the output of the simulation; it is cheap to run, and offers a way to quantify uncertainty for a deterministic system. To calibrate EpiHiper, a Gaussian Process [31], [34] emulator is used inside a Bayesian calibration framework for multivariate output [13], [22] to produce a set of plausible parameter configurations conditioned on the ground truth and associated uncertainty on the future predictions. The calibration task is carried out using the GPMSA framework [18] in Matlab. The calibration workflow typically resumes when ground truth data is updated or when we want to improve our predictions with a more appropriate parameter space or better-modeled mitigations. The calibration may

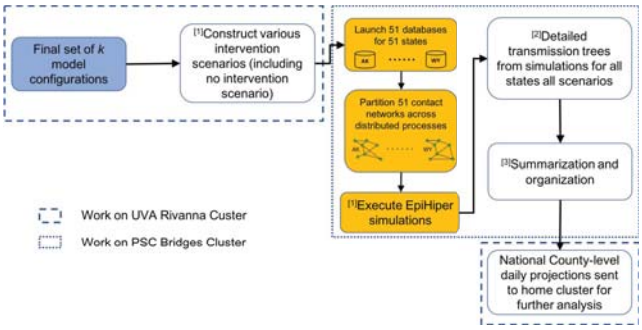

Figure 5. Prediction workflow: [1] An example design has (3 partial reopening levels $\times$ 4 contact tracing compliances) $\times$ 51 states $\times$ 15 replicates = 9180 simulation instances. [2] Size of transmission tree data: 12 cells $\times$ 51 states $\times$ 15 replicates $\times$ 1 million transmissions = 9 billion entries, about 1TB. [3] Size of summary data: 12 cells $\times$ 51 states $\times$ 15 replicates $\times$ 365 days $\times$ 90 health states $\times$ 3 counts = about 1 billion entries, 2.5GB.

reuse the existing model configurations, or generate new configurations as simulated by EpiHiper. This can vary by state. After simulation and aggregation, the time series of simulated case counts is compared to the ground truth with the aforementioned Bayesian approach to generate configurations for the prediction workflow.

Prediction workflow (Figure 5). To make predictions, we run simulations using the model configurations generated from the calibration workflow, and aggregate individual-level output to obtain future counts for various forecasting targets (e.g. confirmed cases, hospitalizations, deaths) at various spatial resolution (state or county level) with different temporal horizons (from one week to five months ahead) depending on the objective. The ensemble of the model configurations and the simulation output provides uncertainty quantification on the predictions. The prediction workflow typically resumes when the calibration workflow generates a set of model configurations, which are simulated by EpiHiper. The output is aggregated and analyzed by public health domain experts to identify inconsistencies (which may then trigger the calibration workflow again). If the predictions are deemed reasonable, we expand the configurations with a few possible future *what-if* scenarios (e.g. what if the stay-at-home order is lifted earlier; what if the mitigation compliance rate increases; what if testing and contact tracing are improved). Then simulations are run for the expanded configurations, and the results are combined with the as-is predictions.

III. Description of Hardware, Software and Data

In this section, we describe the individual components of the overall workflow: (i) the underlying hardware, (ii) the software components used, and (iii) data used as input and generated as output. As mentioned earlier, the hardware consists of a home cluster (Rivanna) and a remote super-computing cluster (Bridges). Our typical workflow depicting the sequence of computations and data transfers between the two clusters are described in Figure 1, and described in more detail in the following section. The workflow relies on two important datasets as inputs: (i) the synthetic population and associated social contact network for the US, and (ii) COVID-19 specific disease parameters. Ranges for these parameters are based on best estimates from COVID-19 literature. The final component of the pipeline is the simulation-based models. Although we use meta-population models in addition to agent-based models, we will focus here on the agent-based models due to the significant computing challenges they pose.

Home cluster and remote super-computing cluster. Our methodology makes use of two computing clusters, which we refer to as the *home cluster* and the *remote super-computing cluster*. The home cluster refers to the Rivanna computing cluster available at University of Virginia, the author's home institution. The Rivanna cluster is modest-sized relative to the significantly more powerful remote super-computing cluster, the Bridges at Pittsburgh Super-computing Center. The actual simulation runs are executed on the Bridges cluster. We find this distinction to be fairly typical and important, as most institutions do not have a super-computing facility available on their local premises, and researchers/practitioners often run the less compute-intensive parts of their workflows on their local systems, while running more computationally heavy tasks at dedicated super-computing facilities. Making this distinction explicit allows us to formally take into consideration issues arising from these kind of setups.

Note that commercial cloud computing platforms, such as Amazon EC2 and Google Cloud Platform, also provide services to make it relatively easy to set up computing clusters with software stacks mimicking those of HPC and super-computing facilities. Hence, this logical separation of "home cluster" and "remote super-computing cluster" is also relevant for institutions making use of hybrid cloud infrastructures, where a small local compute cluster is used alongside off-premises, cloud-based systems.

Input Data: Synthetic populations and contact networks. Our epidemic computational models depend upon *detailed synthetic populations* and *contact networks* to support accurate and realistic simulations. Such data is prepared for each state, see Figure 6 for a summary of node and edge counts by US state.

For each population, data is supplied as a comma-separated values (CSV) file containing the *traits* of each synthetic person. Whereas particular sets of traits may vary across simulations, typical choices for the US include household ID, age and age group, gender, county code, and the latitude and longitude of home locations. For design

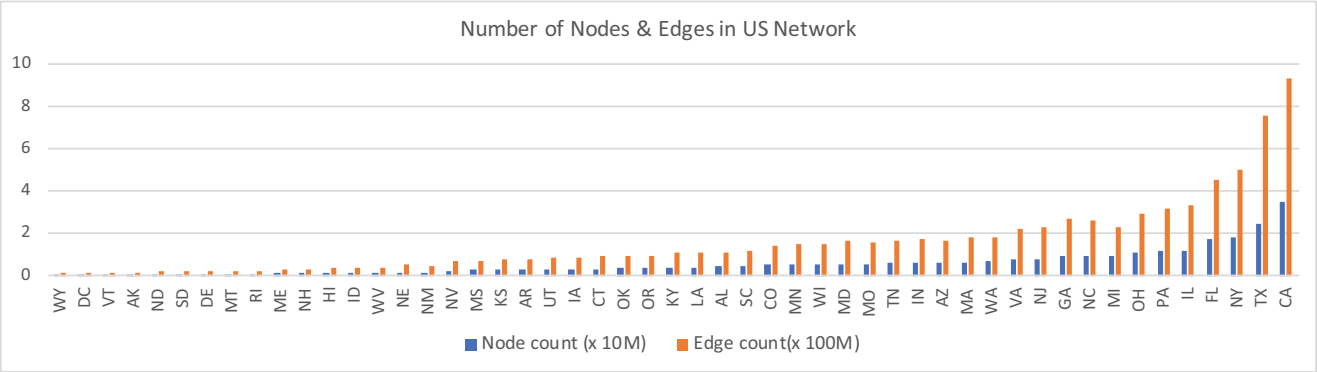

Figure 6. The diagram shows the number of nodes and edges in the contact network for each U.S. state as used in the simulations.

reasons, but also to avoid the cost of parsing and reading files from the file system during simulations, the population data is loaded into a PostgreSQL[4] database server. All simulations access the population data by communicating with the database server at run-time.

The agent-based models use dynamic contact networks to encode interactions between persons during simulations. The initial dynamic contact network in EpiHiper is generated statically. However, during the course of the simulation, each edge in the contact network can be turned on and off dynamically as required in response to, for example, social distancing interventions. Like the person data, the contact network of each population is supplied to the simulations as one CSV file. Each edge in the contact network includes the identifiers of the two persons in contact, and is annotated by the start time and duration of the interaction, in addition to the context in which the persons meet (home, work, shopping, other, school, college, and religion). These contexts may not be the same for both persons, however; for example, if one person is at the store, their context may be shopping, while the grocer they came in contact with would be working. Due to the large size of the contact networks, the network is partitioned between different MPI processes at the beginning of the simulation run. The overall objective is to split the contact network such that each partition contains approximately the same number of edges, while, at the same time, ensuring that all incoming edges of any given node are in the same partition. In the current implementation, we utilize a simple algorithm to partition edges: given a partition, continue to allocate nodes to that partition until the number of incoming edges is greater than a threshold $(E/P + \epsilon)$ where E is the number of edges, P is the number of partitions, and ϵ is the tolerance factor. Note that even a simple partitioning scheme (such as the one described) takes a significant amount of compute time. This is why we use our current (simple) algorithm rather than one that is more sophisticated or optimal. We can also cache the result of the partitioning computation on disk, which saves time on future runs.

Input Data to simulation: Disease progression parameters and parameter configurations. The disease model used for this work is shown in Figure **??** in the extended version of this paper (see footnote 2, page 1) and depicts the transmission of COVID-19 through interactions between individuals, and the subsequent disease progression of an infected individual. As shown in Figures 4 and 5, both calibration and prediction workflows start by generating simulation configurations, also known as *cells*. For calibration workflows, a larger number of cells are created, each with smaller numbers of replicates relative to routine prediction workflows, in order to explore the model configuration state space. For prediction workflows, however, a much smaller number of cells are generated which are based on the most likely model configurations from the calibration phase, each with a relatively larger number of replicates. The model configurations specify which populations and contact networks to use, as well as the disease parameters, interventions, initializations, and the number of days to simulate.

Input data to calibration. For calibrations, we use confirmed cases from multiple data sources[5] [6] [7] as our ground truth dataset. The ground truth data has county-level daily confirmed case counts starting from January 21, 2020, for over 3000 counties (as of April 22, 2020, there were 2772 counties with case counts greater than zero).

Simulation-based models. *EpiHiper* is an agent-based discrete time simulation model for infectious disease spread in a social contact network. It is implemented as a parallel codeset in C++/MPI. It computes probabilistic disease transmission between nodes (representing individuals) in a network of edges (representing interactions between individuals), as well as the disease progression within each infected

[4]https://www.postgresql.org/

[5]https://github.com/nytimes/covid-19-data
[6]https://nssac.bii.virginia.edu/covid-19/dashboard/
[7]https://coronavirus.jhu.edu/map.html

individual. It is based on the synthetic populations, accessible to the simulations via a database launched at run-time, and the synthetic contact network, partitioned pre-simulation and loaded into memory of the allocated processing units in order to support scalability. The simulation keeps track of the health state of each individual at each tick (the temporal resolution, set to one day in this case).

Output data: dendograms and summary information. EpiHiper produces state transitions of all persons during the simulation. Each line of the output file written by EpiHiper includes the tick of the transition event, the identifier of the person, their exit state, and the identifier of the person causing the state transition in the case of disease transmission. The size of the output depends on the total number of ticks, overall epidemic size (number of infected persons), as well as the complexity of the finite state machine. Dendograms are part of this output, which are transmission trees rooted at initial infections.

From the individual-level output data, we can aggregate simulation results to the county level for different health states, and use the summary data for calibration and prediction. For example, the time series of daily cumulative counts of symptomatic cases at the state or county level are compared to the ground truth data as part of our calibration, and daily counts of symptomatic cases, hospitalizations, ventilations, and deaths are used in our predictions.

IV. Orchestration of the workflows

Structure of simulation jobs. The software stack on the Bridges cluster uses the Slurm scheduler for scheduling jobs, and Intel MPI for distributed communication. PostgreSQL servers are utilized to run the population databases. The number of processes to use per compute node is predetermined statically based on the configuration of individual compute nodes on the cluster. Furthermore, as described earlier, the population networks are partitioned statically beforehand, and they also determine the number of compute nodes/processes that will be utilized when running simulation jobs that use them. For simulations sharing a given user population, a single PostgreSQL server is started on a compute node and made available. The simulations use them to load population information at run-time. The data transfer between the Rivanna cluster and the Bridges cluster utilizes the Globus platform[8].

Every 24 hours, simulations are generated and executed to support the decision-making processes of policymakers. The process begins with the generation of the simulation configurations. The nature of the configurations generated depends on whether the calibration or prediction workflows are to be executed. Calibration workflows typically generate a large number of different model configurations to explore the space of the configurations. Prediction workflows, however, typically have a smaller set of model configurations, each replicated multiple times.

Once the configurations are generated, their transfer from the Rivanna cluster to the Bridges cluster is started manually using the Globus platform. Once the configurations are copied over, the population databases are started, one per population. To speed up the start of the population databases, snapshots of the databases are generated when the populations are initially created, and these snapshots are instantiated at run-time. Next, scripts are used to submit Slurm job arrays, which are scheduled to run using the heuristic scheduling strategy discussed above. Once simulation jobs have completed, the summary of simulation outputs are generated and transferred back to the Rivanna cluster using the Globus platform.

V. Mapping and scheduling jobs on PSC Machines

Mapping our workflows on the Bridges cluster is an important component. First, recall that the overall efficiency of the workflow is measured as time to complete the workflow rather than a single replicate of a single cell. Abstractly, our workflows can be thought of as large scale hierarchical *statistical experimental designs*. Each workflow is comprised of 51 regions (50 states and DC), and each region is then comprised of a number of cells that each denotes one combination of various parameters used to study a given problem. Each cell is further comprised of a number of replicates. Together, this represents a 3-level hierarchy: *regions-cells-replicates*. Each cell for a given region uses exactly the same input data; thus, we view our atomic jobs as $\langle cell, region \rangle$. For certain workflows, it is more convenient and efficient from a scheduling perspective to group several cells into one to create jobs of appropriate sizes.

In general, the running time for a single replicate for $\langle cell, region \rangle$ is not fixed; this is due to (i) randomness within the computation, (ii) triggered interventions that can, at certain times, cause new calculations to be spawned based on the epidemic, (iii) number of processors assigned to the replicate and (iv) machine-specific randomness due to processors' computation, access to the database etc. Nevertheless, by running the replicate several times we can obtain a reasonable bound on these times. For the workflows considered, we fixed the number of processors assigned to each $\langle cell, region \rangle$. We state the mapping problem in two stages:

The workflow mapping problem (WMP). We are given a set of $\langle cell, region \rangle$ tasks, denoted by task $T[r, c]$. We assume that we know a bound $t_l(T[c, r])$ and $t_r(T[c, r])$ denoting the lower and upper bound on the time to complete task $T[c, r]$ using $p(T[c, r])$ processing units. We use $\overline{t(T[c, r,])}$ to denote the empirical mean running time obtained by running the computation several times and will use this for the rest of the paper. We assume $p(T[c, r])$ is

[8]https://www.globus.org/

known a priori. The problem is assigning an order to these tasks, then supplying this ordered set to the Slurm scheduler in such a way that minimizes the overall completion time of all tasks.

WMP is **NP**-hard. This can be seen by reducing the 2D Bin packing problem to the WMP problem: Rectangles become tasks: their width becomes $p(T[c,r])$ and their height is running time $\overline{t(T[c,r])}$. This reduction is useful and this correspondence also leads to natural heuristics for the problem discussed later in this section.

Database Access Constraints. There is one additional constraint that needs to be taken care of which makes the problem computationally challenging. The constraint relates to database access. Recall that each task needs access to the input synthetic network. The number of simultaneous connections to the database are upper bounded for technology and efficiency reasons. We can capture this by using a compatibility graph. Usually compatibility constraints for tasks are captured as a coloring problem: we have a node for each task, and two tasks u and v have an edge iff they cannot be scheduled at the same time. A valid coloring captures a feasible schedule. In our case, the problem is more challenging and can be best described as a *new* kind of vertex coloring problem, which we will call a *relaxed coloring problem* (r-relaxed-coloring): We are given a graph $G(V,E)$. Edges represent conflicts, and vertices represent tasks. We are given a number r. The (r-relaxed-coloring) is to assign a color to each node in the graph (such a graph would be constructed for each region separately) such that if a node v gets color $c[v]$ then no more than r of its neighbors can get the color $c[v]$. If $r = 1$, we get the classical coloring problem and thus all the hardness results hold for the relaxed coloring problem as well.

The DB-access constrained workflow mapping problem (DB-WMP). DB-WMP is a constrained version of the WMP wherein the number of tasks that can be scheduled simultaneously is bounded. Thus the general DB-WMP problem can be thought of as 2D Bin packing with an interesting compatibility constraint.

Our Mapping heuristic (MAP). Our mapping heuristic is based on a few simplifying assumptions and exploiting the problem structure. *Assumption 1:* We assume that all tasks for a given region take the same amount of time which is $\overline{t(T[c,r])}$, in other words $\forall c_i, t(T[c_i,r]) = \overline{t(T[c,r])}$. *Assumption 2:* All tasks have to be scheduled non-preemptively. *Assumption 3:* The number of connections that can be made by tasks corresponding to a region r is bounded by $B(T[r])$ (i.e. it is not dependent on the cell). *Assumption 4:* For each region, all the tasks $T[c_i,r]$ require the same number of threads for simplicity and are denoted by $dt(T[c,r])$, thus $\sum_c dt(T[c,r]) > B(T[r])$. Our heuristic is motivated by the non-decreasing first fit heuristic. Recall the task of this heuristic is to provide the Slurm scheduler an ordering and chunking of tasks. Slurm further does a certain amount of real-time optimization. It comprises of the following steps:

Step 1. Split the overall database so that we have one database per region. For various system-level reasons and from the standpoint of human productivity, each such database occupies one node of the system. Thus, all tasks corresponding to a given region can access the region-specific database. Access by each region can now be done in parallel with no constraints beyond the fact that we have a constraint on the total number of processors. Let $T[r] = \cup_c T[c,r]$ denote the set of tasks for region r. The above decomposition makes the coloring problem easy. We now have r subsets — one subset per region. There is no edge between the subset, and the graph within each subset is a complete graph. All tasks for a given region r thus belong to a Region set $RS(r)$.

Step 2. Organize the tasks in non-increasing order by time needed to complete the computation. The time is directly correlated with the size of the network for each cell. Using an idea motivated by the 2D Bin packing methods, we use a level-oriented approach [8], [9], [35], [38]. Think of processors on the X-axis and time on the Y-axis. The tasks are mapped from left to right (in terms of available processors), in rows forming levels. Within the same level, all tasks are packed so that their bottoms align. The first level is the bottom of the strip and subsequent levels are defined by the time taken of the slowest task on the previous level.

Step 3. We considered two different mapping algorithms: The Next-Fit Decreasing time with database constraints (NFDT-DC) algorithm assigns the next task $T[c,r]$ (in non-increasing time) on the current level if $T[c,r]$ fits and database access constraints are satisfied. Otherwise, the current level is "closed" and a new level is created. The First fit decreasing time with database access constraint (FFDT-DC) algorithm schedules the next task in non-increasing order of time, until either the database access constraint for the region is violated, or, if no level can accommodate the task, a new level is started.

Without the database constraints, the NFDT-DC and FFDT-DC algorithms have worst-case performance guarantees of 2 and $17/10$ respectively. Let EC denote the empirical efficiency of our method. This is computed as the ratio of the total time used by all processors as they were computing divided by the product of the total processors and the time when the last task was completed. As the next section discusses, our algorithms do quite well; the FFDT-DC ordering achieves a very high system utilization.

VI. Performance Analysis

Runtime performance of EpiHiper. Figure 7 (top) shows that EpiHiper's running time increases linearly with its input size. On the other hand, Figure 7 (middle) shows how increasing the number of processing units for three

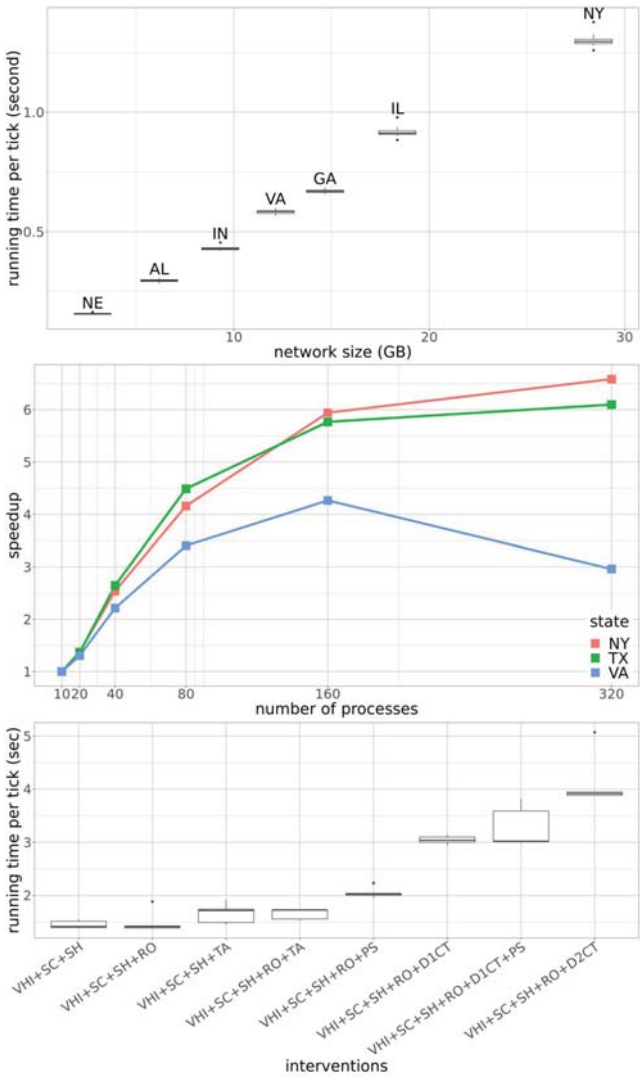

Figure 7. (top) Running time of EpiHiper on networks of different sizes given 40 processing units. (middle) As the number of processing units increase, the corresponding improvement in the performance of the simulations illustrates the strong scaling results of EpiHiper. Beyond some point, which varies with the problem size, the benefit of using more resources starts to diminish. (bottom) Running time of EpiHiper varies with different interventions in the simulation. Simulations with more interventions, or with more complex interventions, take more time.

medium-to-large networks can significantly improve simulation performance. The improvement in the performance, however, starts to decrease beyond a certain number of processing units due to increasing communication costs between processes. It may even become slower with too many processes. In Figure 7 (bottom), we show that EpiHiper's running time depends also on the interventions implemented in the simulation. In the base case, the simulation has implemented VHI (voluntary home isolation), SC (school closure), and SH (stay-at-home). When we add more interventions to the simulation, the running time increases. The

	Bridges Cluster	**Rivanna** Cluster
# Allocated nodes	720	50
# CPUs/node	2	2
# Cores/CPU	14	20
RAM per node	128GB (DDR4)	384GB (DDR4)
CPU	Intel Haswell E5-2695 v3	Intel Xeon Gold 6148
Network	Intel Omnipath-1	Mellanox ConnectX-5
Filesystem	Lustre	Lustre

Size of user traits and contact networks	2TB (one time)
Size of daily simulation configurations	100MB–8.7GB (per day)
Size of raw simulation outputs generated	20GB–3.5TB (per day)
Size of summarized outputs	120MB–70GB (per day)

Table II
CONFIGURATION OF THE BRIDGES CLUSTER AT PITTSBURGH SUPERCOMPUTING CENTER AND THE RIVANNA CLUSTER AT UNIVERSITY OF VIRGINIA, ALONG WITH DATA GENERATED AND MOVED ACROSS THEM.

simpler interventions RO (partial reopening), which extends SH, and TA (testing and isolating asymptomatic cases), which extends VHI, increase running time marginally. The more complex interventions PS (pulsing shutdown), which repeatedly alternates SH and RO, and D1CT (distance-1 contact tracing and isolating), which affects many more nodes and edges, significantly increase the running time. The most complex intervention we have implemented so far, D2CT (distance-2 contact tracing and isolating), increases the running time by almost 300% from the base case.

Scheduling and partitioning simulation jobs. The primary purpose of the workflow presented in this paper is to serve the needs of policymakers by providing them with timely predictions of disease progression that incorporates the most recent data. To serve this purpose, we face a high throughput problem where we have to maximize the number of simulation jobs we can execute in order to generate calibration and projection results. We are given two constraints (i) limited compute time (10:00pm - 8:00am), and (ii) limited number of compute nodes as described in (Table II). The job scheduling strategy presented in the previous section focuses on timeliness, that is, reducing the time span required to execute a given set of jobs on the compute cluster.

The minimal memory requirement per job is given by the size of the contact network which is stored in memory during runtime. Furthermore, the memory requirements may increase due to the complexity of interventions performed in a scenario. Our experience is that in nearly all cases, the additional memory is proportional to the network size. For simplicity, we therefore divided the 51 regions (networks) into 3 categories: small (2 compute nodes), medium (4), and large (6). With these assignments, we were able to guarantee that the jobs have sufficient memory to complete even the complex intervention scenarios. We intentionally avoided using partial nodes in order to limit problems caused by competing memory requirements of different jobs running

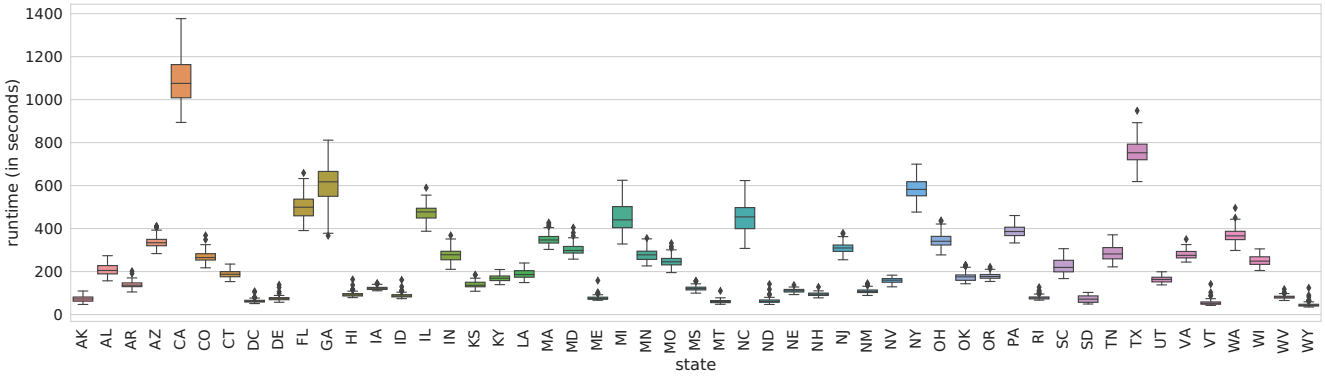

Figure 8. Variance in runtime for EpiHiper simulations for different US states, across different cells or simulation configurations.

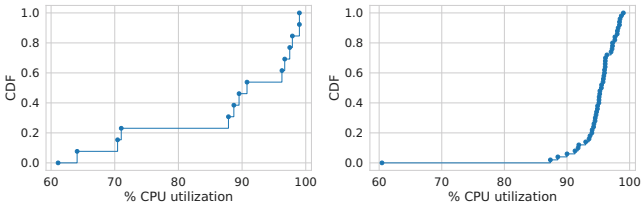

Figure 9. Utilization of compute resources on Bridges cluster for different days of workflows. The *left* figure shows utilization for the days when all 50 states and DC were simulated, while the *right* figure shows utilization for days when only different cells for the state of Virginia were simulated.

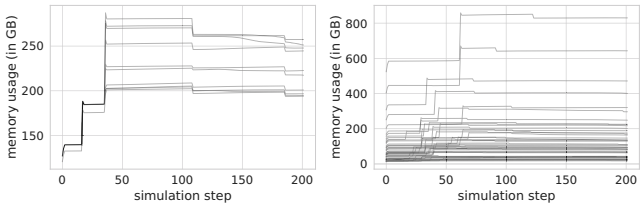

Figure 10. Changes in memory required for different cells for VA (left) and US states (right) at different timesteps. In the left figure, every line corresponds to a different cell or simulation configuration generated for the state of Virginia, and shows the mean memory required (across replicates). In the right figure, every line corresponds to a different US state, and shows shows the mean memory required (across cells).

on the same node. By consistently using the maximum number of cores available per node, we ensure that the available compute resources are fully utilized.

Furthermore, we chose to create static network partitions in order to save compute time, since partitioning the network to binary chunks for California alone would take over one hour. The time for partitioning a network is larger than the typical run time for a simulation run, which usually requires between 100 to 300 time steps of about 3 seconds each for a network the size of California (Figure 7 (left)). We chose not to assign additional resources, since Figure 7 (right) shows that increasing the number of compute nodes for a single simulation gives diminishing returns in terms of runtime; due to that, the cost of messaging negates any gains obtained from using more compute power. After the general categorization of all jobs into the 3 categories above, we are faced with maximizing the number of simulations we are able to run within our time window. Every night, we have a varying number N jobs to run, and face the challenge of scheduling them efficiently.

While supercomputing facilities can grant access to a large amount of resources, access to these resources come at a large cost, either for the users who are paying for the resources directly or for the taxpayer in case of publicly funded research. Thus one important metric to consider for the scheduling problem is the issue of resource utilization. Figure 9 shows the resource utilizations for our workflows, in terms of percent of CPU hours allocated that were actually used. The Figure 9 (left) shows the distribution of utilizations of 9 workflow runs which simulated all 51 regions, while Figure 9 (right) shows the same for 24 workflow runs that simulated multiple configurations for the state of Virginia. In the case of all state workflows, we have a median utilization of 96.698% while the same for Virginia-only workflows is 95.534%. Note that the above results are for the scheduling configuration (FFDT-DC) where the largest jobs were scheduled first. Our initial workflow runs without this scheduling scheme (NFDT-DC) led to utilization numbers between 44.237% and 55.579% for the all-state case.

Runtime performance of simulation ensembles on remote system. Here we present the runtime characteristics of the simulations. Figure 8 shows the variance in runtime (across compute nodes) for the 50 US states and DC for a single representative day of simulation. To understand the dynamic nature of the total memory required for the different EpiHiper simulations, we plot the total memory required for different cells, and US states in Figure 10. Figure 8 shows that runtimes of simulations are dependent on intervention scenarios and is strongly correlated to the network size. The memory increase during the simulation in Figure 10 is due to the intervention scheduled at fixed time points. Figure 10 (left) shows that memory requirements in

the same scenario may depend on the compliance of nodes with the interventions, i.e., higher compliance and, therefore, more scheduled changes to the system state require more memory. Finally, Figure 10 (right) shows that the final memory requirements are strongly correlated with the initial requirements, i.e., the network size.

VII. Illustrative case studies

In this section, we discuss three case studies: (i) medical cost of the pandemic illustrated in Figure 3, (ii) forecasting workflow illustrated in Figure 5 and (iii) calibration workflow illustrated in Figure 4. The second and third workflows are discussed in the Appendix F in the extended version of this paper (see Footnote [1]).

The workflows selected illustrate a range of tasks undertaken using the two supercomputing clusters.

Case study: Medical costs of COVID-19. In this study, we estimate the medical costs of COVID-19 in the US. The overall impact also includes the cascading effect to the Gross Domestic Product (GDP), which can be analyzed by an input-output or general equilibrium model. Since the purpose is to demonstrate the workflow, we will focus on the medical cost estimating.

The medical costs include costs incurred by COVID-19 patients for medical attention, hospitalization, ventilator support, etc. For each patient, the total costs depend on the disease severity. We consider a calibrated (towards $R_0 = 2.5$) disease model with different scenarios with respect to NPI (non-pharmaceutical intervention) duration and compliance. For each scenario in our factorial design of 12 cells, we run simulations with 15 replicates for each of the 51 regions (50 states and DC), with county-level seeding derived from county-level confirmed case counts. The simulation outputs individual-level data on who are infected, receiving medical attention, hospitalized, and/or on ventilator support each day. The aggregate data is used to compute the total medical costs for each scenario. The details of the study are described in [6]. The workflow for our economic impact analysis consists of the following steps: *(i)* On Rivanna or Bridges cluster, calibrate the disease model towards $R_0 = 2.5$. *(ii)* On the Rivanna cluster, prepare simulation configuration files for a factorial design of different NPI durations and compliance; get the most recent county-level confirmed case counts and use them to prepare county-level seeding. *(iii)* Send the disease model, seeding, and configuration files to the Bridges cluster. *(iv)* On the Bridges cluster, create database jobs and simulation jobs, use our scheduling heuristic to submit jobs, and run post-simulation data aggregation. *(v)* Transfer aggregate simulation data to the Rivanna cluster. On the Rivanna cluster, run the economic impact model to estimate medical costs.

VIII. Related Work

Over the last decade, there has been substantial interest in developing scalable solutions to support various epidemiological tasks. This includes: planning and counterfactual analysis, forecasting, and various resource optimization problems. There has also been interest in developing web-based tools to support these tasks. The models used in these papers often range from simple statistical models to compartmental models. Due to space considerations, we only highlight a few important papers here.

Agent-based models in epidemic sciences can be traced back to the earlier work on human immunodeficiency virus (HIV), although the models were largely focused on the structural analysis of small networks; see [12], [16], [19]. The use of the models was largely restricted to modeling studies. Recent papers that aim to scale these simulations to the national level include [4], [30].

In [3], [10] the authors report on the development of web-based systems to carry out large computational experiments in support of epidemic planning. See [11], [20], [28] for other related efforts.

Researchers have also created data-driven pipelines to support epidemic forecasting. CDC runs an annual challenge in this area for studying influenza. Several important advances have been made to improve the overall forecasts; most of the work in this space is either statistical time series models or simple compartmental mass action models; see [32], [36]. Operational agent-based models for epidemic forecasting have not yet been reported on. Recently there have also been a lot of community-wide efforts related to COVID-19[9] [10] [11]; our group submits forecasts to a number of these efforts.

Developing scalable pipelines and workflows for HPC tasks involving large datasets has also been well-studied in literature [14], [21], [25], [29]. For example, the authors of [14] present a technique for building scalable workflows for analyzing large volumes of satellite imagery data, while [25] present a system for analyzing workflows related to weather-sensing data. Other studies have presented generalized methodologies for building scalable workflows for tasks requiring HPC platforms [5], [21].

Recently there has been a flurry of papers on developing agent-based and equation-based models for planning and response to the COVID-19 pandemic; see [1], [2], [7], [15], [17], [23], [24], [27], [33], [37] The present paper does not focus on our agent-based models — they are covered in a companion paper.

Our primary focus is on creating scalable HPC-oriented workflows to support a range of epidemiologically relevant tasks in real-time. Our work shows how two large

[9]https://github.com/ihmeuw/covid-model-seiir-pipeline
[10]https://covid-19.bsvgateway.org/
[11]https://reichlab.io/

supercomputing clusters have been used to meet that goal, and is a step towards demonstrating the use of hybrid supercomputing cloud technology for epidemic science. The resulting challenges are unique, and form an important data-driven simulation platform.

IX. Conclusion

We describe how we have developed high performance computing-oriented epidemic workflows in order to support the planning and response to pandemics such as COVID-19. Our workflows are unique in their use of two geographically separated supercomputing clusters. The workflows are also unique from the standpoint of executing large data-intensive steps that incorporate daily county-level surveillance and policy data, national and highly resolved agent-based simulations of epidemic processes, and post-simulation analytics for projections and counter-factual analysis. The work arose in response to requests from federal and state agencies to support their work on COVID-19 planning, and, using this approach, we have been able to provide uninterrupted support for over 30 weeks. This was accomplished in record time – we began this effort in early March after access to such machines was made possible by the HPC Consortium. We were provided with unprecedented support by Pittsburgh Supercomputing Center. Our results demonstrate that *real-time data-driven high resolution epidemics science at national scale is possible*. COVID-19 is not over; we are witnessing a second, or possibly third, wave. The tools we have developed will assist policymakers in developing and evaluating new intervention measures, and will hopefully help prevent COVID-19 from becoming an even larger-scale outbreak.

Acknowledgment

We thank the anonymous reviewers for their valuable suggestions.

This work was partially supported by National Institutes of Health (NIH) Grant 1R01GM109718, NSF BIG DATA Grant IIS-1633028, NSF Grant No.: OAC-1916805, NSF Expeditions in Computing Grant CCF-1918656, CCF-1917819, NSF RAPID CNS-2028004, NSF RAPID OAC-2027541, US Centers for Disease Control and Prevention 75D30119C05935, University of Virginia Strategic Investment Fund award number SIF160, Google Grant, and Defense Threat Reduction Agency (DTRA) under Contract No. HDTRA1-19-D-0007. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

The authors would like to thank members of the Biocomplexity Institute and Initiative for useful discussion and suggestions. We thank the staff members at University of Virginia's high performance computing center for their help in ensuring our jobs run smoothly every day on the Rivanna Cluster. We also thank staff members at the Pittsburgh Supercomputing Center, Dr. John Towns head of NSF XSEDE project and Dr. Shawn Brown, head of PSC for providing us the needed HPC resources at such a short notice; our work would not have been possible without their support. The timely and important resource provided by the NSF during this national crisis was crucial and is sincerely appreciated.

References

[1] D. Adam. Modelling the pandemic the simulations driving the world's response to covid-19. *Nature*, 580(7803):316–318, 2020.

[2] C. Avery, W. Bossert, A. Clark, G. Ellison, and S. F. Ellison. Policy implications of models of the spread of coronavirus: Perspectives and opportunities for economists. Technical report, National Bureau of Economic Research, 2020.

[3] R. Beckman, K. R. Bisset, J. Chen, B. Lewis, M. Marathe, and P. Stretz. Isis: A networked-epidemiology based pervasive web app for infectious disease pandemic planning and response. In *Proc. ACM SIGKDD*, pages 1847–1856, 2014.

[4] A. Bhatele, J.-S. Yeom, N. Jain, C. J. Kuhlman, Y. Livnat, K. R. Bisset, L. V. Kale, and M. V. Marathe. Massively parallel simulations of spread of infectious diseases over realistic social networks. In *Proc. IEEE/ACM CCGrid*, pages 689–694, 2017.

[5] V. G. Castellana, M. Drocco, J. Feo, J. Firoz, T. Kanewala, A. Lumsdaine, J. Manzano, A. Marquez, M. Minutoli, J. Suetterlein, A. Tumeo, and M. Zalewski. A parallel graph environment for real-world data analytics workflows. In *Proc. DATE*, pages 1313–1318, 2019.

[6] J. Chen, A. Vullikanti, S. Hoops, H. Mortveit, B. Lewis, S. Venkatramanan, W. You, S. Eubank, M. Marathe, C. Barrett, and A. Marathe. Medical costs of keeping the US economy open during COVID-19. *medRxiv*, 2020.

[7] M. Chinazzi, J. T. Davis, M. Ajelli, C. Gioannini, M. Litvinova, S. Merler, A. P. y Piontti, K. Mu, L. Rossi, K. Sun, et al. The effect of travel restrictions on the spread of the 2019 novel coronavirus (covid-19) outbreak. *Science*, 2020.

[8] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

[9] E. G. Coffman, Jr, M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

[10] S. Deodhar, K. R. Bisset, J. Chen, Y. Ma, and M. V. Marathe. An interactive, web-based high performance modeling environment for computational epidemiology. *ACM TMIS*, 5(2):1–27, 2014.

[11] A. Deshpande, K. Margevicius, E. Generous, K. Taylor-McCabe, L. Castro, J. Longo, and R. Priedhorsky. Tools and apps to enhance situational awareness for global disease surveillance. *Online journal of public health informatics*, 6(1), 2014.

[12] S. Eubank, H. Guclu, V. A. Kumar, M. V. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988):180–184, 2004.

[13] A. Fadikar, D. Higdon, J. Chen, B. Lewis, S. Venkatramanan, and M. Marathe. Calibrating a stochastic, agent-based model using quantile-based emulation. *SIAM/ASA Journal on Uncertainty Quantification*, 6(4):1685–1706, 2018.

[14] J. Farnes, B. Mort, F. Dulwich, S. Salvini, and W. Armour. Science pipelines for the square kilometre array. *Galaxies*, 6(4):120, 2018.

[15] N. Ferguson, D. Laydon, G. Nedjati Gilani, N. Imai, K. Ainslie, M. Baguelin, S. Bhatia, A. Boonyasiri, Z. Cucunuba Perez, G. Cuomo-Dannenburg, et al. Report 9: Impact of non-pharmaceutical interventions (npis) to reduce covid19 mortality and healthcare demand, 2020.

[16] N. M. Ferguson, D. A. Cummings, C. Fraser, J. C. Cajka, P. C. Cooley, and D. S. Burke. Strategies for mitigating an influenza pandemic. *Nature*, 442(7101):448–452, 2006.

[17] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall, and C. Fraser. Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. *Science*, 2020.

[18] J. Gattiker, K. Myers, B. Williams, D. Higdon, M. Carzolio, and A. Hoegh. Gaussian process-based sensitivity analysis and bayesian model calibration with gpmsa. In R. Ghanem, D. Higdon, and H. Owhadi, editors, *Handbook of Uncertainty Quantification*, pages 1867–1907. Springer, Switzerland, 2016.

[19] T. C. Germann, K. Kadau, I. M. Longini, and C. A. Macken. Mitigation strategies for pandemic influenza in the united states. *PNAS*, 103(15):5935–5940, 2006.

[20] J. J. Grefenstette, S. T. Brown, R. Rosenfeld, J. DePasse, N. T. Stone, P. C. Cooley, W. D. Wheaton, A. Fyshe, D. D. Galloway, A. Sriram, et al. Fred (a framework for reconstructing epidemic dynamics): an open-source software system for modeling infectious diseases and control strategies using census-based populations. *BMC public health*, 13(1):940, 2013.

[21] V. Hendrix, J. Fox, D. Ghoshal, and L. Ramakrishnan. Tigres workflow library: Supporting scientific pipelines on hpc systems. In *Proc. IEEE/ACM CCGrid*, pages 146–155, 2016.

[22] D. Higdon, J. Gattiker, B. Williams, and M. Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.

[23] S. C. Kamerlin and P. M. Kasson. Managing covid-19 spread with voluntary public-health measures: Sweden as a case study for pandemic control. *Clinical Infectious Diseases*, 2020.

[24] M. U. Kraemer, C.-H. Yang, B. Gutierrez, C.-H. Wu, B. Klein, D. M. Pigott, L. du Plessis, N. R. Faria, R. Li, W. P. Hanage, et al. The effect of human mobility and control measures on the covid-19 epidemic in china. *Science*, 2020.

[25] E. Lyons, G. Papadimitriou, C. Wang, K. Thareja, P. Ruth, J. J. Villalobos, I. Rodero, E. Deelman, M. Zink, and A. Mandal. Toward a dynamic network-centric distributed cloud platform for scientific workflows: A case study for adaptive weather sensing. In *Proc. eScience*, pages 67–76, 2019.

[26] M. Marathe and A. K. S. Vullikanti. Computational epidemiology. *Communications of the ACM*, 56(7):88–96, 2013.

[27] C. J. E. Metcalf, D. H. Morris, and S. W. Park. Mathematical models to guide pandemic response. *Science*, 369(6502):368–369, 2020.

[28] J. Paparian, S. Brown, D. Burke, and J. Grefenstette. Fred navigator: An interactive system for visualizing results from large-scale epidemic simulations. In *Proc. IEEE eScience*, pages 1–5, 2012.

[29] I. Paraskevakos, M. Turilli, B. C. Gonçalves, H. Lynch, and S. Jha. Workflow design analysis for high resolution satellite image analysis. In *Proc. eScience*, pages 47–56, 2019.

[30] K. S. Perumalla, A. J. Park, and V. Tipparaju. Discrete event execution with one-sided and two-sided gvt algorithms on 216,000 processor cores. *ACM Trans. Model. Comput. Simul.*, 24(3), June 2014.

[31] C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (gpml) toolbox. *JMLR*, 11(Nov):3011–3015, 2010.

[32] N. G. Reich, L. C. Brooks, S. J. Fox, S. Kandula, C. J. McGowan, E. Moore, D. Osthus, E. L. Ray, A. Tushar, T. K. Yamana, et al. A collaborative multiyear, multimodel assessment of seasonal influenza forecasting in the united states. *PNAS*, 116(8):3146–3154, 2019.

[33] K. Roosa, Y. Lee, R. Luo, A. Kirpich, R. Rothenberg, J. Hyman, P. Yan, and G. Chowell. Real-time forecasts of the covid-19 epidemic in china from february 5th to february 24th, 2020. *Infectious Disease Modelling*, 5:256–263, 2020.

[34] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

[35] M. Seizinger. The two dimensional bin packing problem with side constraints. In *Operations Research Proceedings 2017*, pages 45–50. Springer, 2018.

[36] J. Shaman and A. Karspeck. Forecasting seasonal outbreaks of influenza. *PNAS*, 109(50):20425–20430, 2012.

[37] R. Verity, L. C. Okell, I. Dorigatti, P. Winskill, C. Whittaker, N. Imai, G. Cuomo-Dannenburg, H. Thompson, P. G. Walker, H. Fu, et al. Estimates of the severity of coronavirus disease 2019: a model-based analysis. *The Lancet infectious diseases*, 2020.

[38] Wong and Jansen. Lanl covid-19 cases and deaths forecasts survey on 2-dimension bin packing. https://cgi.csc.liv.ac.uk/~epa/surveyhtml.html.