COVID-19 Joint Pandemic Modeling and Analysis Platform

Gautam Thakur, Kevin Sparks, Anne Berres, Varisara Tansakul, Supriya Chinthavali, Matthew Whitehead, Erik Schmidt, Haowen Xu, Junchuan Fan, Dustin Spears, Elton Cranfill
Oak Ridge National Laboratory
Oak Ridge, Tennessee
{thakurg,sparksa,berresas,tansakulv,chinthavalis,whiteheadmc,schmidteh,xuh4,fanj,Spears,spearsdl,cranfillej}@ornl.gov

ABSTRACT

The non-pharmaceutical intervention to reduce the impact and spread of COVID-19 requires the development of policies and guidance through a collaborative effort among government, academia, medicine, and citizens. To operationalize this effort, we have developed an all-encompassing situational awareness platform that can process multi-modal and multi-source data allowing informed decision making. Besides, showing the current spread of infection, the platform also captures the impact of human dynamics on the infection spread, location, and availability of critical infrastructure, prediction, and high-performance computing driven simulation. The platform is extensible, allowing third-party integration and services to consume the curated data and analytics in near real-time. We believe the platform will augment critical decision making for reducing the impact and spread of the pandemic.

CCS CONCEPTS

• **Computer systems organization → Real-time system architecture**.

KEYWORDS

real-time architectures, visualization, distributed systems

ACM Reference Format:
Gautam Thakur, Kevin Sparks, Anne Berres, Varisara Tansakul, Supriya Chinthavali, Matthew Whitehead, Erik Schmidt, Haowen Xu, Junchuan Fan, Dustin Spears, Elton Cranfill. 2020. COVID-19 Joint Pandemic Modeling and Analysis Platform. In *1st ACM SIGSPATIAL International Workshop on Modeling and Understanding the Spread of COVID-19 (COVID-19), November 3, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3423459.3430760

1 INTRODUCTION

In January 2020, the Director-General of the World Health Organization (WHO) declared the novel coronavirus (COVID-19) outbreak a Public Health Emergency of International Concern (PHEIC), WHO's highest level of alarm. Since then, both pharmaceutical and non-pharmaceutical interventions sprung to action in an attempt to staunch the spread of infection. In the latter case, healthcare agencies, volunteers, non-profit organizations, and several others have put forward an effort of epic proportions to curate high-quality datasets on cases and available healthcare resources. Prediction and simulation models are built to demonstrate the likelihood of infection and its impact. However, to a large extent, these efforts are isolated and focus solely on one thing; for example, Johns Hopkins University (JHU) reports cases at the province level in China; at the city level in the USA, Australia, and Canada; and at the country level otherwise[7]. Policymakers need an all-encompassing view of the situation to make an informed and rational decision for maximum impact. This involves quick access to current situational awareness, future prediction, and ancillary information such as the location of critical infrastructure. Until recently, no such mechanism exists that provide everything under a single umbrella. There could be several reasons: i) it is difficult to conflate multi-variate data with varying spatial and temporal granularity, ii) in a continuously evolving situation, data variables and schema also evolves, making it impossible to converge on a stable data format, iii) the volume and velocity of data is enormous, requiring specialized data and compute architecture. Also, in a dynamic environment like this, the currency of the data and analytics is paramount. Thus, Real-time (RT) architectures capable of stream processing are vital to address the challenge arising from the currency of data insights. Developing scalable and operational RT architectures have a high upfront cost, sometimes making it difficult to build and run.

The objective of this work is to develop an integrated COVID-19 pandemic monitoring, modeling, and analysis capability that will include, - i) historical and current spatio-temporal trends of disease spread, ii) estimates of required hospital beds, ICU units, ventilators, etc., iii) needed testing capacity and where iv) quantify the effectiveness of implemented interventions and mitigation strategies. To address these challenges, we developed and operationalized an agile, online COVID-19 platform for integrating, synthesizing, analyzing, and visualizing geographically resolved data (collected as part of this effort) as well as conveying modeling and simulation results that anticipate future COVID-19 transmission dynamics.

The proposed platform is built by hybridizing the concepts of Lambda architecture and Hyperscaling to achieve real-time analytics and visualization of spatiotemporally disparate datasets through load-aware vertical and horizontal scaling of available infrastructure with zero downtime. Besides the architecture, the proposed platform offers two key application-level functions:

Multisource data integration data store: A critical component of the online platform is the ability to ingest and merge structured and unstructured data sources curated in support of the COVID-19 platform from multiple sources that include hospitals, regional governments, social media, and other crowd-sourced outlets related to COVID-19 infectious diseases spread.

Interactive analytics dashboard: A substantive system for merging and scientifically analyzing multiple, disparate open-source data

COVID-19, November 3, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-8168-0/20/11…$15.00
https://doi.org/10.1145/3423459.3430760

streams (e.g. COVID-19 cases, twitter content, quarantine maps, demographic context, and news feeds), physical data (e.g. temperature, precipitation) and modeling and simulation outcomes. The work leverages hyperscale architecture for data curation, analysis, and visualization of a range of curated and modeled data.

The remainder of the paper is structured as follows: section 2 modestly discusses related work and relevant background, the proposed architecture is discussed in section 3, while the operational workflow release plan is discussed in section 4. The end product of the platform is a suite of interactive dashboards, their design and development are discussed in section 5. Hotspot detection is discussed in section 6 and extending the platform to connect with third-party applications is discussed in section 7. Finally, the paper concludes in section 8 with a discussion on stress testing.

2 BACKGROUND

Ever since the onset of the COVID-19 pandemic, countless web-based dashboards have been developed to track the development of COVID-19 at the local, national or global level. Most governmental agencies (e.g., Centers for Disease Control and Prevention (CDC), state or county health agency) use web-based dashboards to release the latest information about COVID-19 to the public. It does not take long for researchers and decision makers to realize the need for a more comprehensive platform that can collect, aggregate, visualize, and predict the dynamics of COVID-19 using various scattered data sources.

One of the most widely referenced platforms is the web-based dashboard from Johns Hopkins University[8], which tracks the latest number of COVID-19 cases and deaths around the world, at different spatial granularities for different countries. It functions both as an authoritative COVID-19 data curator and a basic tracking tool with visualizations from ESRI. The New York Times[25] also developed a dashboard that offers similar functions. There are many other web-based platforms that visualize and track different aspects of the pandemic, e.g., the spread and evolution of different strains of SARS-CoV-2[14], real-time symptom and public behavior survey around the world[6, 11], online conversation and information spread[22], healthcare system capacity[2], and human mobility[9]. Most of these tools focus on a relatively narrow aspect of a large system, and lack a predictive analysis capability. In this work, we have developed a geospatial platform that provides real-time situational awareness, prediction, and simulation results generated by different laboratories. More importantly, it conflates disparate data sources to depict a story with context rather than mere statistics.

For all emergency response analytic platforms, context is a critical component of communication. As stated in [26], "Geography and history offer unique perspectives on context through study of the interconnectedness of phenomena, events, and places across multiple spatial and temporal scales through which situations are understood." For COVID-19, this means effectively communicating information beyond case counts and deaths. Providing geographic and historical information relating to the spread of the pandemic is important. Furthermore, context surrounding COVID-19 extends to supporting information like hospitalizations, where and how much people are traveling to public/commercial spaces, school closures, and more. To that end, the front-end visual portion of many COVID-19 dashboards contextualize the pandemic by including map views with multiple layers, supported by various graphs, charts, and tables all with historical and current data, with which users can interact.

The proposed platform is built on lambda architecture[17] allowing a way of processing massive quantities of data that provides access to batch-processing and stream-processing methods with a hybrid approach. The lambda architecture itself is composed of 3 layers: batch, serving, and stream. The platform benefits from this approach, when combining archive data with streaming data that the platform collects. In addition, the platform incorporates the features of hyper-scaling architecture[4] that can benefit from expanding both compute and storage power as required. Besides these, there exists a plethora of architectures such as kappa[18], derived from lambda architecture and less complex to deploy in real-world, Apache Hadoop[15], Apache Spark[27], among others. There are more architectures, we keep the details limited, and for more information review[21, 24].

Besides high-availability, another important measure of success for any scalable architecture is its ability to maintain low latency during I/O intensive operations. Distributed Caching Mechanisms (DCM) that stores large amounts of data in the memory of more than one machine offers to bridge the gap and improves the latency. Information-centric networking[28] is one of the best examples of a distributed cache implementation that focuses on location-independent content sharing across the planet. There are four types of DCMs, that includes cache aside, read-through cache, write-through cache and write-back. These approaches are application and scenario dependent on maximizing the application throughput. Open source implementation of DCMs are widely available, such as Hazelcast, Memcache, and Redis, among others[23]. These are data agnostic and their effectiveness depends on the implementation. The proposed platform uses a combination of write through caching for improving the performance of temporally located data.

3 ARCHITECTURE

This platform is built on the principles of lambda[17] and hyperscale[4] architecture to address the challenges of combining disparate data sources and dynamically scale to address computational challenges. The architecture benefits from the use of widely available off the shelf servers and computational equipment. The biggest benefit lies in the ability to scale the platform as a function of load and latency to accommodate additional users and requests. The architecture can be scaled both vertically and horizontally, maximizing in-built fault tolerance and cost-effectiveness.

The proposed architecture is shown in Figure-1 that includes data collection and processing, distributed data grid to expedite the data transfer and reduce latency, application server interface, machine learning, and data quality evaluation. The remainder of this section discusses these in more detail.

3.1 Data Collection

The first step in release planning is the collection and curation of high-quality authoritative data. This involves discovering relevant data source(s), sanitizing and transforming the new data, de-duplicating, and semantically conflating it with other existing data sources. The data collection's geographic coverage is the entire planet and the spatial granularity goes to the county or even census block group level. Besides, data gathering should be done in

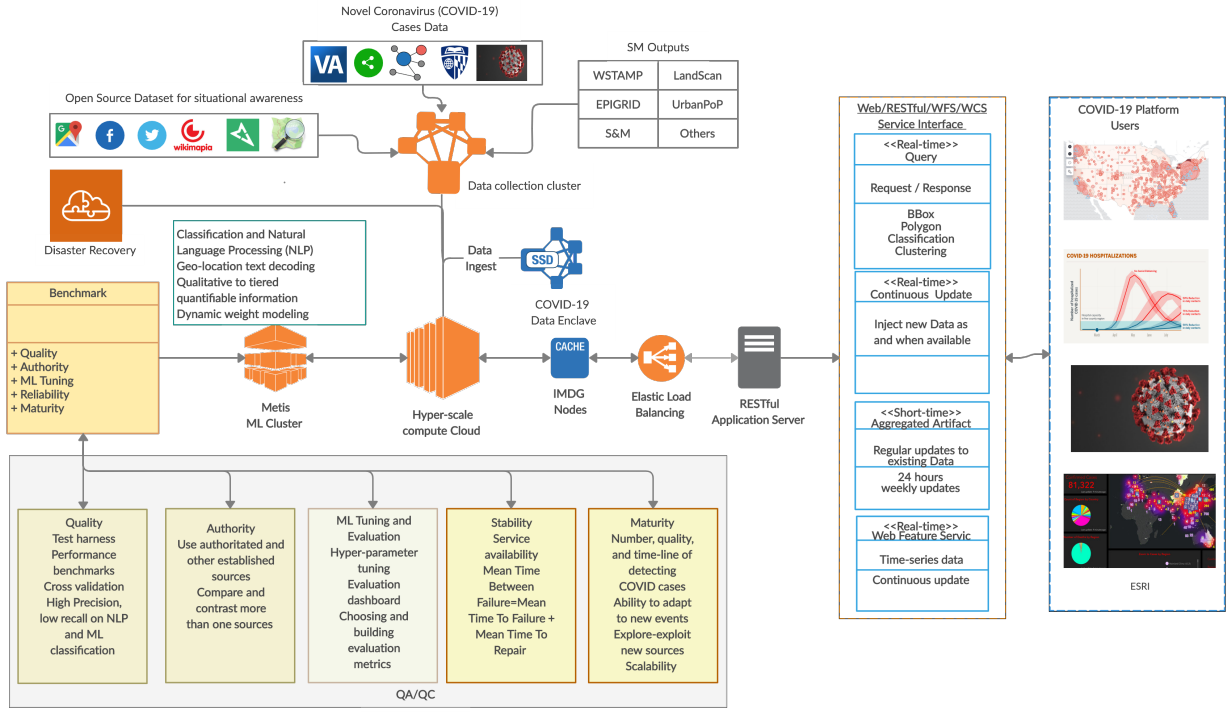


Figure 1: COVID 19 platform architecture

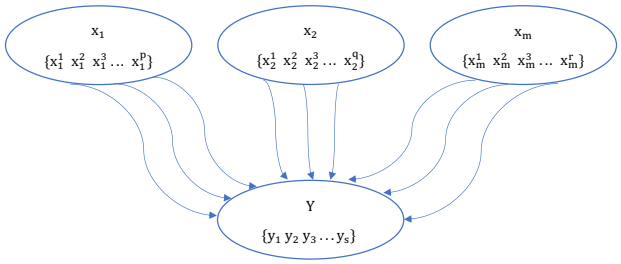


Figure 2: Unified mapping format for disparate data sources

real-time and continuously for the currency of insights. On average, over 100 data sources were searched daily and several million records were collected. At this scale, data curation efforts must be automated with a human in the loop only when necessary.

A universal attribution format was designed to unify disparate data sources ($X \in \{x_1, x_2, x_3, ...x_m\}$) and a translation engine (Θ) is developed to map attributes of disparate data sources to the universal format data ($Y \in \{y_1, y_2, y_3, ...y_s\}$). For each data source, a separate sequence of translations ($\Theta \in \{f_1, f_2, f_3, ...f_m\}$) were designed such that each uniquely maps to universal data format as,

$$f_i : x_i \rightarrow Y \tag{1}$$

where each data source x_i has k attributes, $x_i \in \{x_i^1, x_i^2, x_i^3, ...x_i^p\}$ that maps to Y s attributes, such that $p \leq q \leq r \leq s$, as shown in Figure 2. This was an important step to be performed so aggregate statistics and visualization from disparate data sources can be done seemliness manner.

3.2 Data Processing and Analytic

Developers use RESTful APIs to access the data for processing and generating analytics. Also, boilerplate templates are made available for developers to generate analytics via natural language processing and machine learning. Besides authoritative data, the platform also harvests social media data (e.g. twitter) to gather information about infection spread and response. Classification models can be built to evaluate communication and assess their impact on health. Rigorous benchmarks that include, quality control, tuning and maturity of results are evaluated before releasing the results.

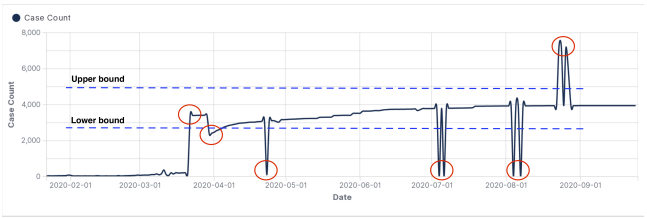


Figure 3: Missing data in time-series

3.3 Quality Assurance Quality Control

A suite of spatio-temporal and statistical data processing templates are developed to gather insights from the data. In addition, visualization packages such as Vega and custom ensemble visualization are integrated in the platform to allow the development of interactive dashboards and processing of reports. At times, the platform ingests direct results of simulation[10] or predictive models[16] for the purpose of visualization.

Algorithm 1: Algorithm for server-side HyperCache

Function `Remove`(*key*) *return value*
- **Data:** Distributed map
- **Result:** value of element to be removed
- *acquire_lock*();
- *value* ⟵ *remove*(*hmap*, *key*) ;
- **if** *value != null* **then**
  - return *value*;
  - *sync_cache*() ;
  - *wait*();
- **else**
  - return *null* ;
- *release_lock*();

Function `Update`(*key, value*) *return result*
- **Data:** Distributed map
- **Result:** value of element to be updated
- *acquire_lock*();
- *result* ⟵ *update*(*hmap*, *key*) ;
- **if** *result != 1* **then**
  - return *false*;
- **else**
  - return *true* ;
- *sync_cache*() ;
- *wait*() ;
- *release_lock*();

3.4 Distributed HyperCache

Developing elastic architectures that scale as a function of an evolving computing workload is essential for real-time applications. Significant advances have been made in hyper-scalable storage, data centers, and cloud computing infrastructure to accommodate the exponential increase of such workload. In this architecture, we utilize the distributed memory of nodes to improve storage latency that is processing and simultaneously retrieving a large amount of disparate data for real-time analysis. HyperCache is implemented in the form of an In-Memory Distributed Grid and is built on the top of the Direct-Attached Storage (DAS) computing cluster running simultaneous applications. These applications communicate with HyperCache via client-server architecture for maximum compatibility. A monitoring system is developed and deployed for performance bench-marking and providing essential support during exponential data compute workloads. A simple representation of adding/updating and removing the element in HyperCache is shown in Algorithm-1.

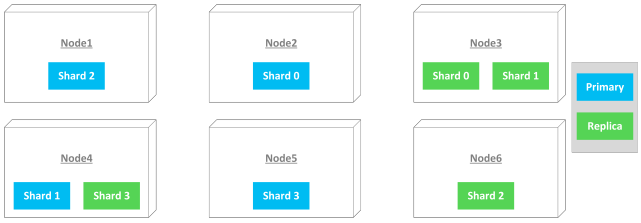


Figure 4: Replication of an index across six nodes.

3.5 Replicated Data Management

In this section, we discuss fault tolerance and approaches taken to ensure the platform remains accessible and robust when it is needed most. The platform manages its data integrity through replication across multiple servers. As shown in Figure 4, a database index is broken down in four primary shards (pieces) and four replicated shards. They are distributed across six nodes in such a way that if any machine is down or corruption, the database can still be rebuilt and no data loss occurs. The global consistency is maintained by frequently broadcasting updates and propagating them on all the nodes.

3.5.1 Serialization and Optimistic Concurrency. The management and consistency of aforementioned replicated data is achieved through serialization with optimistic concurrency control, such that the execution of a set of parallel data operations (transactions) must be equivalent to a serial execution of the same data operation[5]. Consider $\Gamma = \{t_0, t_1, ..., t_m\}$ is a set of parallel transactions. Then, for each transaction, t_i, let R_i is the read set and W_i is write set respectively for t_i. For example, in parallel when $t_i \rightarrow t_j$ occurs, then t_i must come before t_j equivalent in a serial transaction. Optimistic concurrency protocol ensures that any execution if not consistent is aborted based on the timestamp. A workspace is maintained for each transaction that later on executed to maintain long-term data consistency. This mean, sometimes user request response includes cached results that are not fully updated. Broadly, for three transactions t_0, t_1, t_2 such that their respective timestamps are $t_0 < t_1 < t_2$, the operations on a shared object occur in increasing order of the timestamp. Recent transactions (smaller timestamp) wait for older transaction (large timestamp) to finish to maintain data integrity. If an older transaction with larger timestamp (e.g. r_2) encounters a younger transaction (e.g. t_0), the previous dies and restarts with a smaller timestamp. This approach avoids potential deadlocks, as the execution of transactions are based on increasing order of timestamps.

The optimistic concurrency is broken down in three phases - i) execution phase, ii) validation phase, and iii) update phase. It begins by assigning each transaction t_i a timestamp TS_i at the start of the transaction and TV_i at the beginning of validation. It's read as assigned as R_i and written are assigned as W_i in this phase. In *execution phase*, a local workspace is created for each transaction with shadow copies of object to be updated. These objects are updated

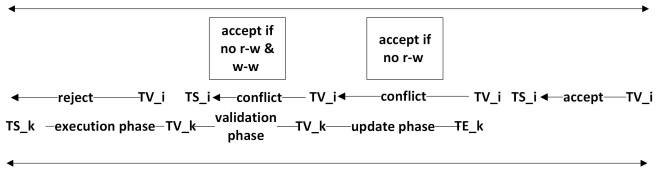


Figure 5: Optimistic Concurrency Control to store and visualize real-time data ingestion

locally and assigned a version number. In case of abort, the transaction is cancelled and the workspace is deleted. Otherwise, the transaction moves to next phase. In *validation phase*, mutual consistency among this and the distributed transactions is performed at remote locations to ensure serializability. The validation between two transactions (Say t_i and t_k) is achieved as follows-

- Validation of transaction t_i is not accepted if $TV_i < TV_k$.
- Validation of transaction t_i is accepted if it does not overlap with any t_k.
- The execution phase of t_i can overlap with update phase of t_k, given it completes its update phase before TV_i. Validation of t_i is accepted if $R_i \cap W_k = \emptyset$.
- The execution phase of t_i overlaps with the validation and update phase of t_k, and t_k completes its execution phase before TS_i. Validation of t_i is accepted if $R_i \cap W_k = \emptyset$ and $W_i \cap W_k = \emptyset$.

The details of these stages are shown in Figure 5. In update phase, the changes to the data objects are made permanent and propagated across the cluster and in persistent memory and storage.

3.5.2 Disaster Recovery Plan. Besides replicating the data across the cluster onsite, a remote disaster recovery site was also deployed to keep operations running in case of major failures (such as natural disaster). We utilized Google Cloud Platform as Disaster Recovery as a Service (DRaaS) to snapshot and backup the status of current database every four hours on google cloud. Since, the snapshots are not instantaneous, take time to complete and asynchronous with respect to time and data integrity. This approach uses incremental backup strategies to save change quickly and efficiently. The cloud only serves as a remote location to store the data and serves as a backup. When snapshot is in-progress, it is still possible to add new data and make other requests to the cluster.

3.6 Data and Analytic Quality Control

Data quality and analytics are evaluated to ensure insights are scientifically accurate. The data curation task uses authoritative sources (such as CDC, JHU, etc.), reducing issues related to accuracy. However, curation issues occur when data attribution format changes (e.g. new attribute is added), network connectivity (intermittent disconnection, synchronization issue because of latency), among other issues. An example of Not missing at random (NMAR) data[20] in univariate time series of COVID-19 case counts are shown in Figure 3. In the figure, the probability for a missing observation depends on the value of the observation itself (the observations are not recorded because of a network error)[13]. If needed, linear interpolation or arithmetic smoothing is used to rectify the missing data in time series. Besides, the process also benefits from review by subject matter experts (epidemiologists, geographers, statisticians) from time to time.

3.7 Application Server and End User

The application server holds the core deployment of the application. We have used a load balancer with the multi-instance deployment of an application server for fail-over and load distribution. The platform is deployed at https://covid19.ornl.gov can be accessed via a web browser or through integrating RESTful services. A user authentication mechanism is implemented to secure and limit access to authorized users only. In section 7, we demonstrate an approach to extend the platform connecting ESRI services and the development of the story-telling feature.

4 RELEASE PLAN

The on-set and rapid spread of COVID-19 created an immediate need to deploy a reliable and stable situational awareness platform accommodating inputs generated by several research entities. It was critically important that this platform should display key scientific findings for policy guidance and informed decision making within a given schedule, quality, and effort constraints. This led us to formalize a systematic release plan workflow for the selection and development of features and their incremental release at a regular interval. Each release addresses the production of meaningful insights and new features developed by the participating research entities. This also allowed all the moving parts of the collaboration to work in sync, work towards a common goal, and for the end-user base to anticipate the changes and new updates. The release plan workflow is shown in Figure 6. The platform also monitors the usage of its services, detection of spurious activities, report generation, and allocating resources to allow a third-party application to utilize analytics and data stream. The post-deploy release step is useful toward extending the platform and for measuring the use.

5 VISUALIZATION

The visualization technology we use to display the data in this platform is Kibana, an open source data visualization platform from the Elastic stack. It provides a variety of standard charts, time series graphs, geospatial visualizations, and support for Vega visualizations. In addition, we have developed custom visualizations that we integrate as plug-ins. The user-facing part of this platform is organized as a series of dashboards.

5.1 Dashboards

In an effort to effectively organize, explore, and reflect the different uses of a large and varied volume of data ingested into the platform, we created several dashboards. These dashboards include a Situational Awareness dashboard (displaying current and historical data from global, to US state and county spatial resolutions), a Predictive Analytics dashboard (displaying multiple predictive models at national, state, and county levels), and more. Some of these dashboard provide high-level overviews, others provide a deep-dive into a particular aspect, or even one specific model or data feature.

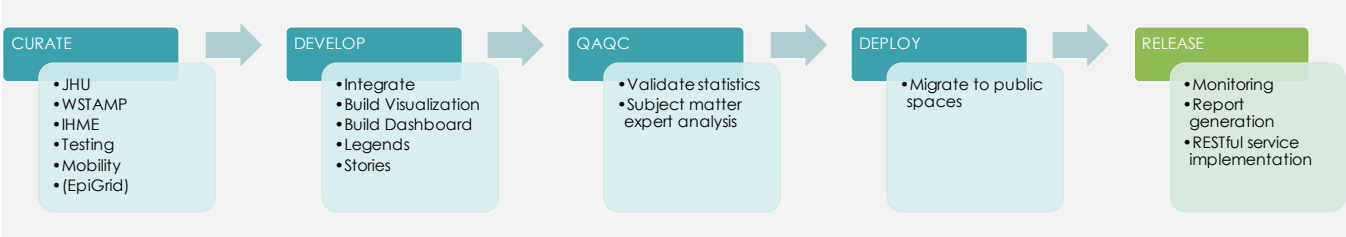


Figure 6: COVID 19 Release Plan Workflow

Incoming data's spatial resolution varies from national, state, county, and city. Temporal resolution ranges from daily, weekly, and monthly time steps. Data updates occur at a variety of times, from static-single upload data layers, to daily updates, to a few updates a month. As such, merging multiple data products into a simple, cohesive view was a challenge. Furthermore, designing visualizations to be responsive to global user input and querying when existing in different spatial and temporal resolutions required careful consideration during data processing, storage, and visualization.

When possible, we attempted to maintain thematic consistency for the purposes of intuitive user experience. Most dashboards consist of interactive maps, time series graphs, and basic charts. Map layers have zoom-dependent views from country, to state, to county. For each layer, we define an appropriate aggregation type (e.g. sum, maximum, etc) depending on the variable presented. To further create a sense of cohesion between dashboards, we attempt to use color as a guide: Similar data is shown in similar colors where possible. For categorical data, we use custom colormaps that represent the type of data appropriately.

In addition to the utilizing Kibana's standard visualizations, we also leverage its support for Vega, a declarative language for web-based visualizations. Vega allows us to create a wider variety of visualizations for the platform and gives us the flexibility to further customize visualizations. For example, in Figure 7 we use Vega to visualize R-naught estimates for the contiguous U.S., Alaska, and Hawaii at the county-level in a single map view. The diverging color scheme provides users with an at-a-glance view of counties experiencing reproduction rates above or below 1, and the hover-over tooltip functionality lets them quickly see the data behind the map. This and other Vega visualizations on the platform query the most recent data from Elasticsearch indices and are configured to respect global filters selected by the user, which allows for seamless and responsive integration with other charts in Kibana dashboards.

5.2 Visualization for Situational Awareness

To illustrate situational awareness (Figure 8), we prioritized the display of current and historical data and analytics regarding case counts, deaths, testing, and various mobility metrics. For example, using daily case counts, we displayed up to date cumulative cases, new case rates, cases per capita, results from transmission rate models, and more. We provided supporting data to illustrate variables that likely influence case counts and deaths. These largely included various measurements of social reaction to the pandemic, including

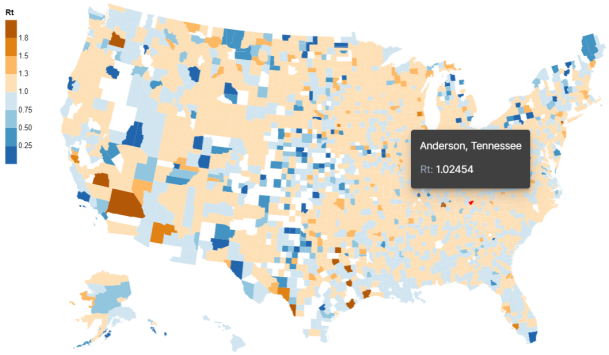


Figure 7: County-level map of R-naught estimates with tooltip functionality. Built with Vega.

dates of school closures, general mobility indices at the national, state and county scales, transportation intensity, and more. Possible user interactions include selectable map layers (country, state, and county scales), dynamically updating map layers based on zoom level, country/state/county term filters, and hover-over tooltips.

5.3 Forecasting and Predictive Visualization

We accumulated multiple predictive models from public (national laboratories) and private industries, that provided near future estimations on case counts, deaths, hospitalizations, intensive care patients, and more. Figure 9 shows filtered visualization of mechanistic model outputs [19] with different resolutions, including state and US metropolitan statistical area (MSA). For example, Figure 9a shows predictions for new cases in Alabama and Figure 9a shows predictions for new cases in Atlanta, Georgia. Note that Figure 9 show the results generated on September 6th, 2020. When the spatial scale and temporal resolution of predictive models were consistent with one another, attempts were made to group these model outputs into a single visualization. Otherwise, model outputs were placed side by side for comparison. All available model outputs were displayed in the dashboard, and users had the ability to filter results by state or county.

5.4 Visualization for Simulation Results

We created dashboards for EpiGrid and EpiCast [10] simulation results that were produced specifically in the context of this project. For each model, we developed a custom processing workflow which

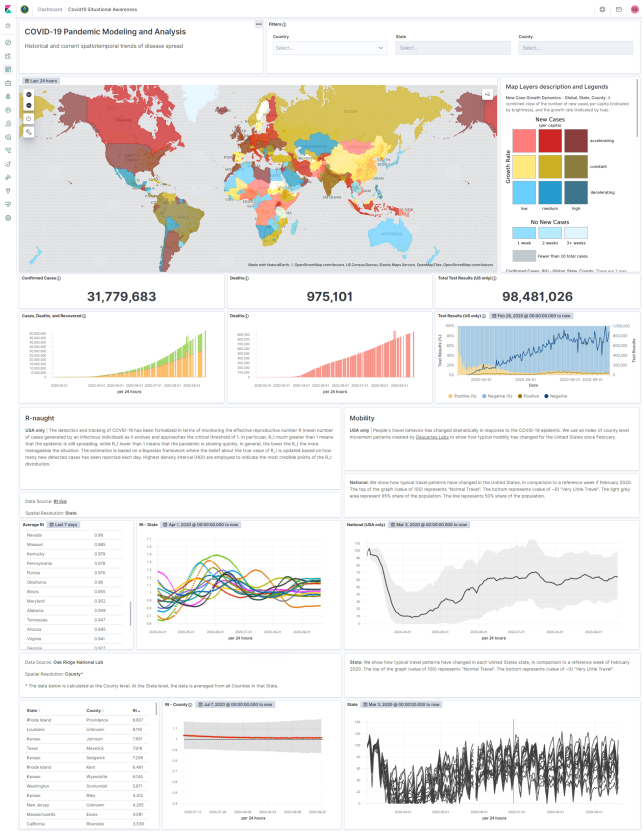
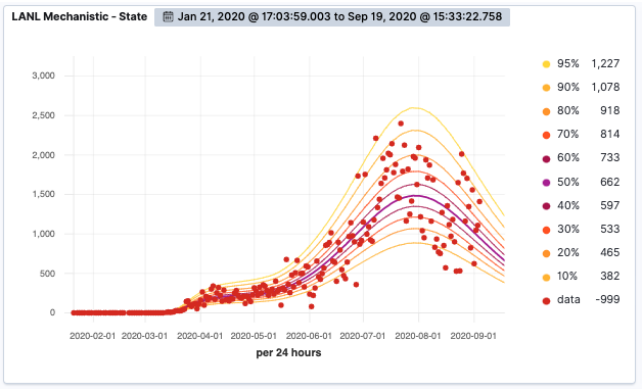
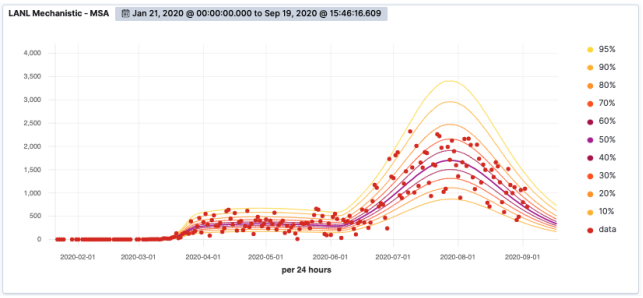


Figure 8: Situational Awareness Dashboard.



(a) New cases predictions for Alabama



(b) New cases predictions for Atlanta

Figure 9: Mechanistic model visualization.

converts the model outputs from their respective formats into a common format that is ingestible into ElasticSearch.

Since aggregations for map layers require incident data, but both models report cumulative infected individuals, we also computed the daily increase in cases as part of our workflow. This allows us to select a start date and end date and produce an accurate count of new cases that were predicted within the given timeframe.

EpiGrid focuses on different stages of infections from first exposure, infectiousness (with isolation status to account for quarantine), requiring hospitalization (with differentiation on whether or not they receive it to model healthcare availability), recoveries, and deaths. EpiCast focuses more on the severity of symptoms with more detailed outputs for hospitalized individuals. It provides outputs for symptomatic, hospitalized, in Intensive Care Unit (ICU), and requiring a ventilator. Unlike EpiGrid, it does not provide recoveries or deaths.

The dashboards for both models are similar, with general information about the model, interactive elements for filtering, a map, and a side bar with instructions, information about layers, and legends. The blue section in Figure 10 shows the top section of the EpiCast dashboard with a view of predicted county-level data of individuals in the ICU on the map. The green section of Figure 10 shows a sample of visualizations from the EpiGrid dashboard for a subset of states (Georgia, Kansas, New Mexico, and New York), curves for each case type, and a comparison with ground truth data from the New York Times dataset [25]. In the pink section of Figure 10 we show a comparison of different model output parameters using the same colormaps as other dashboards (cases = yellow/red, recovered = green, death = black/gray) for some counties in New York. The data for this model run does not cover all counties of each state, which reflects in the comparison chart (top right in green section): the predicted case number (blue) is much lower than the actual case number (red).

6 HOTSPOT DETECTION

Spatial hotspot analysis can identify clustering areas of a spatial phenomenon. To help decision makers better understand the geographic patterns of the COVID-19 in the US, we have developed a hotspot detection module in our platform. For this initial version, we have selected two metrics (*cases per 100k population* and *deaths per 100k population*) that capture the prevalence and seriousness of COVID-19 in a region. In the future, the hotspot detection module can easily be extended to detect hotspots for other types of metrics (e.g, positive rate of testing, hospitalization). The raw data for hotspot detection is collected from Johns Hopkins University (JHU) Data Repository, which provides daily update of confirmed positive cases, deaths for the US at county level. The raw data is cleaned and then joined with census data to provide population data, with US county shape file to provide spatial information. For each county, we then calculated the *confirmed case per 100k population* and *deaths per 100k population*.

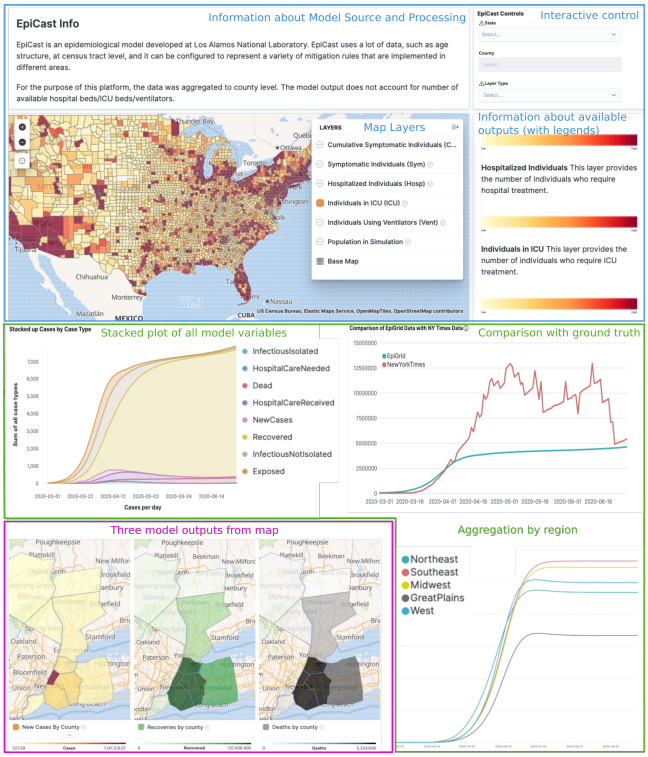


Figure 10: Simulation dashboard layout and some example visualizations from the EpiCast (blue section) and EpiGrid (green and pink sections) dashboards.

6.1 Hotspot detection algorithm

The hotspot detection algorithm uses the Getis-Ord Gi* statistic [12], which works by looking at each county within the context of neighboring counties as well as the national average, to determine whether a county is a hotspot or not.

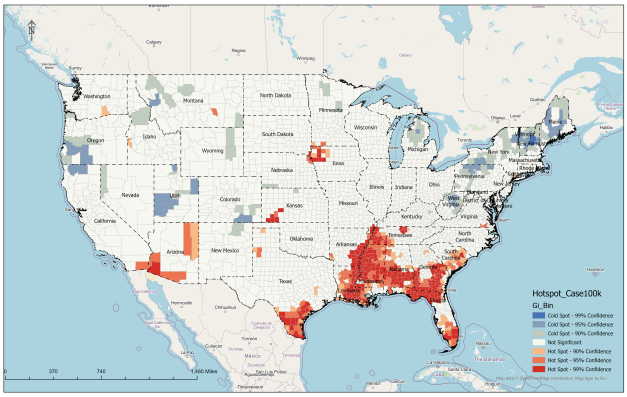
$$G_i^* = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \bar{X}\sum_{j=1}^{n} w_{i,j}}{S\sqrt{\frac{n\sum_{j=1}^{n} w_{i,j}^2 - \left(\sum_{j=1}^{n} w_{i,j}\right)^2}{n-1}}} \quad (2)$$

where $\bar{X} = \frac{\sum_{j=1}^{n} x_j}{n}$ and $S = \sqrt{\frac{\sum_{j=1}^{n} x_j^2}{n} - (\bar{X})^2}$

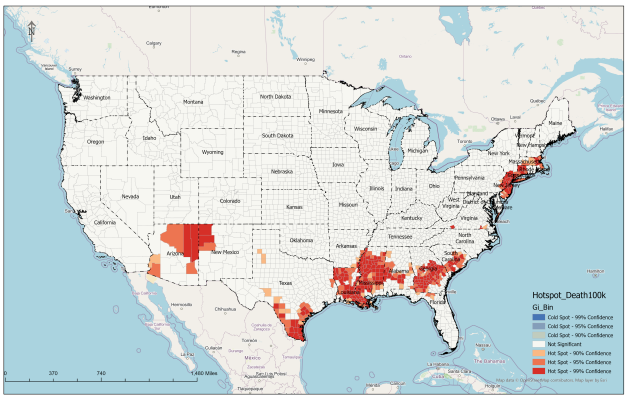
In equation 2, n is the total number of features, x_j is the attribute value of target feature and $w_{i,j}$ is the spatial weight between feature i and j. As we can see from equation 2, Gi* statistic accounts for both national average and neighborhood average. A county that has a high value and is surrounded by other counties with high values as well is a statistically significant hot spot.

6.2 Hotspot visualization

The Gi* statistic calculated for each county is a z-score. If the z-score is statistically significant, the larger the z-score is, the higher confidence we have about the clustering of high values (hot spot). Since the significance of each county is tested individually during the hotspot detection, there could be false positive due to multiple testing. Therefore, we have to calculate the corrected p-value cut-off to correct the bias of multiple testing [3]. The corrected p-value is used for hotspot visualization in our platform.



(a)



(b)

Figure 11: Hotspot visualization.(a) case per 100k population. (b) death per 100k population.

7 EXTENDING THE PLATFORM

In an effort to add extensibility to the platform and allow additional flexibility in web visualization, middle-ware was developed and added to the COVID-19 platform. The middle-ware is based off of the open source project Koop developed by Esri. Koop is a Node.js web server that translates GeoJSON stored in native formats and locations into RESTful Web services. Using the ElasticSearch plugin, Koop was integrated with the existing COVID-19 platform. The output from Koop is Web Feature Services (WFS) which are usable by many web mapping platforms and frameworks. In this case, the WFS generated from Koop are used in a deployment of ArcGIS Enterprise where visualization products including web maps and web mapping applications are created and deployed.

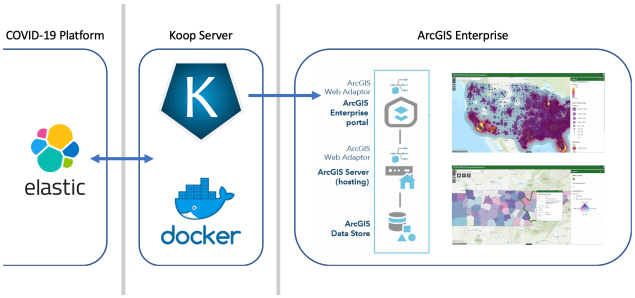


Figure 12: Koop architecture

7.1 Design: Extending Koop

As previously mentioned, Koop with the ElasticSearch plugin, was used to extend the COVID-19 platform, but additional functionality was also developed to customize Koop to fit specific needs within the platform. Koop expects the data being queried to follow the right-hand rule which is a GeoJSON standard that mandates coordinates of exterior rings of a polygon be formatted in a counterclockwise order. Some indices in the COVID-19 ElasticSearch do not follow this standard so functionality was added to reverse the order of the coordinates when necessary. In Figure 12, we show the architecture to connect with COVID-19 platform.

There was also an issue of indices having multiple documents representing the same geometry resulting in a WFS with stacked features. Functionality was developed to allow a service to be configured to only return a single geometry in cases of redundant geometries. The attribution to be returned is also configurable, but as it relates to the COVID-19 platform, the document with the most current date is returned in the case of stacked geometries.

Koop was also extended to account for cases where an index did not include and geometry but had geographic context (e.g county name, state name, FIPS). An additional feature was developed to allow joining the attribution of an index to the geometry of another index. The COVID-19 platform includes indices for both county and state geometries which were used to add geometry to non-spatial indices that had a geographic indicator, in most cases FIPS values.

7.2 Deployment

Because Koop is a simple and lightweight Node.js web server there are many options for deployment that offers flexibility and scalability. One limiting factor, however, is that Koop is single threaded and performance degrades as more services are being generated and used in front end applications. To counteract this, Docker was used to deploy multiple instances of Koop. The deployment strategy was to have a separate Docker container for each WFS output. In some cases, this included multiple containers and services for a single ElasticSearch index.

7.3 Use Case

The extended version of Koop deployed on an array of Docker containers was used to integrate the COVID-19 Platform to a deployment of ArcGIS Enterprise. ArcGIS Enterprise includes ArcGIS Portal which allows users to create and share maps and applications. The WFS generated from Koop that has been configured to the COVID-19 platform's ElasticSearch instance can be added directly into a web map or web mapping application in ArcGIS Portal seamlessly. An application was developed to give a national overview of the current state of the COVID-19 crisis. Also, a state level application was also created to provide a more detailed look at the situation. Both of these applications leverages openly available services, services generated from ArcGIS Enterprise, and services coming from Koop and the COVID-19 platform.

8 PERFORMANCE AND LOAD TESTING

The platform's stability and reliability is evaluated using a suite of non-functional tests that specifically evaluates the readiness of a system. Some examples include load testing, performance testing, availability testing, etc. This is achieved to determine a platform's behavior under both normal and at peak conditions. For this study, we perform load testing to evaluate simultaneous user access and measure network performance.

8.1 Basics

For load testing the COVID-19 platform, k6[1] was used, an open-source load testing tool. To visualize and track load testing metrics on the VM hosts, Prometheus and Grafana were deployed for collection and visualization of the metrics. To generate host metrics, Node Exporter provided a way to constantly expose metrics over a port number and Prometheus was then configured to scrape those metrics by providing the target IPs to Prometheus' configuration. The community dashboards available for Node Exporter provided visualization of the VM specific metrics. The official k6 Grafana dashboard provided details of the load test, including HTTP request durations, HTTP requests per second, etc. An InfluxDB instance was created as well, as k6 provides native support on sending metrics directly to InfluxDB.

8.2 Approach

To create the capable script to be used by k6, its recorder chrome plugin was used to record browser actions and convert them to the script. This included logging in and loading various layers in Kibana dashboards. After modifying the scripts to allow variable overrides, a Docker container was created that loaded the scripts vi CI/CD in GitLab. Then using a batch job deployment in Kubernetes, the load test could begin with various user count simulation and duration. Running this as a Kubernetes job offered the capability to scale the job, deploying multiple parallel running containers. This was used to generate load on the system with environment variable overrides selecting user count simulation and duration of the test. This also permitted the selection of various scripts available in the container for different dashboards within covid19.ornl.gov, without the need of multiple unique docker images.

8.3 Benchmark Results

Our test methodology was to generate load to certify that the COVID19 website was capable of supporting 1,000 simultaneous users. The load testing was run in four stages: 100, 250, 500, and 1,000 simultaneous users. In Figure-13, requests per second generated by 1,000 users is shown. The performance was as expected with low latency and higher throughput as show in Figure-14.

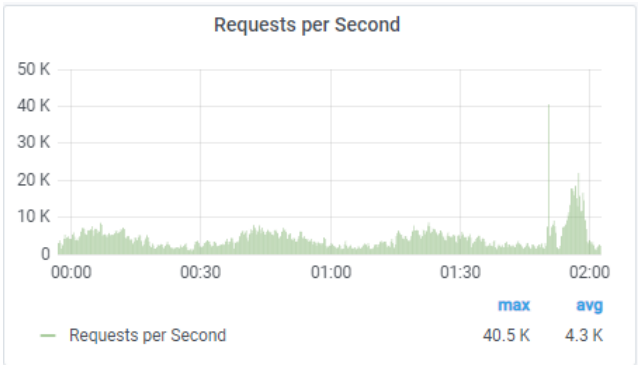


Figure 13: Loading testing for 1000 user shows number of request simulated per unit time.

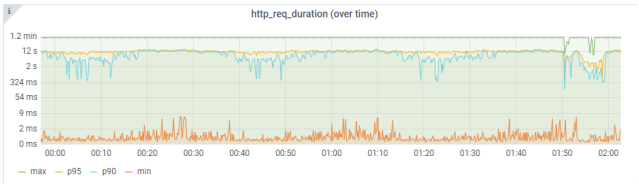


Figure 14: Request frequency and performance latency

We ran a peak test at 1500 users to determine how the system would behave and if performance would degrade beyond the 1000 user limit. Apache reverse proxy was unable to process more than 1,024 users and should be replaced with HAProxy or another proxy engine should the number of simultaneous users exceed 1000.

9 CONCLUSION

In this work, we discussed the development of an all-encompassing operational platform for non-pharmaceutical interventions to the COVID-19 pandemic. The platform is deployed at https://covid19.ornl.gov/ and accessible to authorized users. The underlying scalable architecture supports an end to end workflow for joint pandemic modeling and analysis towards policy guidance and decision making. Custom visualizations are added to display a complex relationship among various data set and the user-facing part of this platform is organized as a series of dashboards. We hope the integration of various datasets, predictions, and simulation results will provide a complete picture to decision-makers for policy guidance.

ACKNOWLEDGMENTS

Research was supported by the DOE Office of Science through the National Virtual Biotechnology Laboratory, a consortium of DOE national laboratories focused on response to COVID-19, with funding provided by the Coronavirus CARES Act.

REFERENCES

[1] [n.d.]. GitHub - loadimpact/k6: A modern load testing tool, using Go and JavaScript - https://k6.io. https://github.com/loadimpact/k6 Accessed 2020-09-23.
[2] Azavea Aaron Su, Anthropocene Labs Dave Luo, Azavea Hector Castro, Azavea Jeff Frankl, Lauren Moos, Azavea Matt McFarland, Azavea Rob Emanuele, Azavea Simon Kassel, and Development Seed Zhuangfang NaNa Yi. 2020. CovidCareMap – Open geospatial work to support health systems' capacity to effectively care for rapidly growing COVID19 patient needs. https://www.covidcaremap.org/
[3] Marcia Caldas de Castro and Burton H. Singer. 2006. Controlling the False Discovery Rate: A New Application to Account for Multiple and Dependent Tests in Local Statistics of Spatial Association. *Geographical Analysis* 38, 2 (2006), 180–208. https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.0016-7363.2006.00682.x
[4] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger. 2016. A cloud-scale acceleration architecture. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–13.
[5] Randy Chow and Theodore Johnson. 1997. Distributed Operating Systems & Algorithms. , 569 pages. https://books.google.com/books?id=J4MZAQAAIAAJ&source=gbs_book_other_versions Accessed 2020-09-22.
[6] Delphi Research Group. 2020. COVIDcast - Real-time COVID-19 Indicators. , 10 pages.
[7] Ensheng Dong, Hongru Du, and Lauren Gardner. 2020. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet. Infectious diseases* 20, 5 (2020), 533–534. https://doi.org/10.1016/S1473-3099(20)30120-1
[8] Ensheng Dong, Hongru Du, and Lauren Gardner. 2020. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases* 20, 5 (2020), 533 – 534. https://doi.org/10.1016/S1473-3099(20)30120-1
[9] Facebook. 2020. COVID-19 Mobility Data Network. , 10 pages. https://visualization.covid19mobility.org/?date=2020-09-23&dates=2020-06-23_2020-09-23&region=WORLD Accessed 2020-09-25.
[10] Paul Fenimore, Benjamin McMahon, Nicolas Hengartner, Timothy Germann, and Judith Mourant. 2018. A Suite of Mechanistic Epidemiological Decision Support Tools. *Online Journal of Public Health Informatics* 10, 1 (may 2018), e1. https://doi.org/10.5210/ojphi.v10i1.8299
[11] Frauke Kreuter, Kathleen Stewart, Andres Garcia, Yao Li, Joe O'Brien, Junchuan Fan, Samantha Chiu, and Anil Kommareddy. 2020. COVID-19 World Symptom Survey. https://covidmap.umd.edu/ Accessed 2020-09-25.
[12] Arthur Getis and J. K. Ord. 1992. The Analysis of Spatial Association by Use of Distance Statistics. *Geographical Analysis* 24, 3 (1992), 189–206.
[13] John Graham. 2012. *Missing data: Analysis and design. New York, NY: Springer.*
[14] James Hadfield, Colin Megill, Sidney M. Bell, John Huddleston, Barney Potter, Charlton Callender, Pavel Sagulenko, Trevor Bedford, and Richard A. Neher. 2018. NextStrain: Real-time tracking of pathogen evolution. *Bioinformatics* 34, 23 (2018), 4121–4123. https://doi.org/10.1093/bioinformatics/bty407
[15] Apache Hadoop. 2011. Apache hadoop. *URL http://hadoop. apache. org* (2011).
[16] IHME COVID-19 Team and Simon I Hay. 2020. COVID-19 scenarios for the United States. *medRxiv* (2020). https://doi.org/10.1101/2020.07.12.20151191
[17] M. Kiran, P. Murphy, I. Monga, J. Dugan, and S. S. Baveja. 2015. Lambda architecture for cost-effective batch and speed big data processing. In *2015 IEEE International Conference on Big Data (Big Data)*. 2785–2792.
[18] Jay Kreps. 2014. Questioning the Lambda Architecture – O'Reilly. , 10 pages.
[19] Yen Ting Lin, Jacob Neumann, Ely Miller, Richard G Posner, Abhishek Mallela, Cosmin Safta, Jaideep Ray, Gautam Thakur, Supriya Chinthavali, and William S Hlavacek. 2020. Daily Forecasting of New Cases for Regional Epidemics of Coronavirus Disease 2019 with Bayesian Uncertainty Quantification. (2020).
[20] Steffen Moritz, Alexis Sardá, Thomas Bartz-Beielstein, Martin Zaefferer, and Jörg Stork. 2015. Comparison of different Methods for Univariate Time Series Imputation in R. arXiv:1510.03924 [stat.AP]
[21] Valerio Persico, Antonio Pescapé, Antonio Picariello, and Giancarlo Sperlí. 2018. Benchmarking big data architectures for social networks data processing using public cloud platforms. *Future Generation Computer Systems* 89 (2018), 98 – 109. http://www.sciencedirect.com/science/article/pii/S0167739X17328303
[22] FBK Researcher. 2020. Covid19 Infodemics Observatory. https://covid19obs.fbk.eu/. (Accessed on 09/25/2020).
[23] Haytham Salhi, Feras Odeh, Rabee Nasser, and Adel Taweel. 2018. Benchmarking and Performance Analysis for Distributed Cache Systems: A Comparative Case Study. In *Performance Evaluation and Benchmarking for the Analytics Era*, Raghunath Nambiar and Meikel Poess (Eds.). Springer International Publishing.
[24] A. Sanla and T. Numnonda. 2019. A Comparative Performance of Real-time Big Data Analytic Architectures. In *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*. 1–5.
[25] NY Times. 2020. Coronavirus in Texas: Map and Case Count - The New York Times. , 10 pages. https://www.nytimes.com/interactive/2020/us/texas-coronavirus-cases.html Accessed 2020-09-25.
[26] Brian Tomaszewski and Alan M MacEachren. 2012. Geovisual analytics to support crisis management: Information foraging for geo-historical context. *Information Visualization* 11, 4 (2012), 339–359.
[27] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, Ion Stoica, et al. 2010. Spark: Cluster computing with working sets. *HotCloud* (2010).
[28] Guoqiang Zhang, Yang Li, and Tao Lin. 2013. Caching in information centric networking: A survey. *Computer Networks* 57, 16 (2013), 3128–3141.