Data Integration Methodologies and Services for Evaluation and Forecasting of Epidemics

Suruchi Deodhar

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

June 17, 2015
Blacksburg, Virginia

Keywords: Computational Epidemiology, Data Integration, Big Data Analytics, Epidemic Forecasting, Web Services, Service Oriented Architecture, Predictive Analytics

Data Integration Methodologies and Services for Evaluation and Forecasting of Epidemics

Suruchi Deodhar

ABSTRACT

Most epidemiological systems described in the literature are built for evaluation and analysis of specific diseases, such as Influenza-like-illness. The modeling environments that support these systems are implemented for specific diseases and epidemiological models. Hence they are not reusable or extendable.

This thesis focuses on the design and development of an integrated analytical environment with flexible data integration methodologies and multi-level web services for evaluation and forecasting of various epidemics in different regions of the world. The environment supports analysis of epidemics based on any combination of disease, surveillance sources, epidemiological models, geographic regions and demographic factors. The environment also supports evaluation and forecasting of epidemics when various policy-level and behavioral interventions are applied, that may inhibit the spread of an epidemic.

First, we describe data integration methodologies and schema design, for flexible experiment design, storage and query retrieval mechanisms related to large scale epidemic data. We describe novel techniques for data transformation, optimization, pre-computation and automation that enable flexibility, extendibility and efficiency required in different categories of query processing. Second, we describe the design and engineering of adaptable middleware platforms based on service-oriented paradigms for interactive workflow, communication, and decoupled integration. This supports large-scale multi-user applications with provision for online analysis of interventions as well as analytical processing of forecast computations. Using a service-oriented architecture, we have provided a platform-as-a-service representation for evaluation and forecasting of epidemics.

We demonstrate the applicability of our integrated environment through development of the applications, DISIMS and EpiCaster. DISIMS is an interactive web-based system for evaluating the effects of dynamic intervention strategies on epidemic propagation. EpiCaster is a situation assessment and forecasting tool for projecting the state of evolving epidemics such as flu and Ebola in different regions of the world. We discuss how our platform uses existing technologies to solve a novel problem in epidemiology, and provides a unique solution on which different applications can be built for analyzing epidemic containment strategies.

Acknowledgments

I would like to sincerely thank my advisor, Dr. Madhav Marathe for being my academic mentor and guide during the PhD program. I have gained a great deal of technical knowledge on various aspects of distributed computing, mathematical modeling and big data through my regular interactions with him. My conversations with him have always been a great source of inspiration and novel research ideas. In addition to technical insights, I have gained invaluable lessons in leadership, management and work ethic by observing his interactions with students, faculty and staff, and I hope to carry that in my professional career as well.

I would like to thank my committee members Dr. Jiangzhuo Chen and Dr. Keith Bisset for mentoring me during my time at NDSSL. Their insights and guidance on various topics related to distributed systems, databases, high-performance computing and epidemiology have been instrumental in shaping my research work. I am especially thankful to Dr. Chen for giving me several valuable comments and suggestions on my dissertation.

I would like to thank Dr. Chris Barrett for his unique strategic insights in the context of the overall research work in computational epidemiology performed at VBI and its applicability in other domains. I extend my sincere gratitude to Dr. Eli Tilevich and Dr. Naren Ramakrishnan for their inputs based on data science aspects and middleware, which are relevant to my thesis. In particular, I would like to thank Dr. Tilevich, from whom I was fortunate to learn the art of writing good research papers early on in my PhD study and who has always been very motivating during our technical interactions.

I am thankful to Dr. Nathaniel Osgood for accepting to be a part of my committee as an external committee member. His insights into my work on analytics based on his experience in computational epidemiology have given a new perspective to my thesis work.

I have been very fortunate to be a part of the Network Dynamics and Simulation Science Laboratory (NDSSL) at Virginia Bioinformatics Institute (VBI) during my research work. The wide and diverse group of individuals at NDSSL have all been very helpful and collaborative throughout my PhD work. I am not sure if I will find another work place with such talented, dedicated and helpful bunch of people anywhere else. I would like to sincerely thank all the members of NDSSL and in particular, Yifei Ma and Jiangzhuo Chen, with whom I collaborated closely during the initial part of my thesis related to Indemics, and Mandy Wilson for her timely help and support in the last leg of my research work. Mandy's knowledge and experience with databases has helped me tremendously in shaping my work on EpiCaster.

My research work would not have been complete without the support of my close friends in Blacksburg. In particular, I would like to sincerely thank Sapna Kaul, Saurabh Bisht, Balaji Subramanian and Astha for being my family away from home and for making my time in Blacksburg a memorable one. I cherish my time spent in Blacksburg because of them.

Last but definitely not the least, I would like to thank my family without whom this thesis would not have been possible. My sincerest gratitude to my mother, Supriya Deodhar, who has been a pillar of strength and motivation throughout my life and to my father, Vikas Deodhar, who has been a great source of inspiration. I am eternally grateful to my elder brother, Santosh Deodhar, who has been my teacher, my guide and my guru throughout my life and for motivating me to pursue a degree in STEM and instilling the confidence in me to pursue a doctorate in Computer Science. The technical discussions that I have had with him have always uplifted me and have often given me a new direction whenever I was stuck.

Finally, I would like to thank my husband, Krishnan for being my partner and friend over so many years and and for supporting me in this long journey. I am grateful to him for his constant encouragement and for being an equal shareholder in our responsibilities. This thesis would not have been possible without his presence in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computational epidemiology involves the development and use of computer models to simulate the changes in spatio-temporal patterns of health among populations of different regions, leading to infections. It also involves evaluating various methods for controlling the propagation of such patterns, commonly referred to as epidemics.

Computational modeling environments incorporate various computational models and methods, and allow analysts and epidemiologists to understand disease propagation dynamics and undertake counter-factual experiments for studying the efficacy of various intervention strategies. Potential interventions for controlling infectious diseases include pharmaceutical interventions, such as vaccinations and distribution of anti-virals, as well as non-pharmaceutical interventions, including social-distancing mechanisms, such as school closures to reduce interactions between individuals. Non-pharmaceutical interventions for vector-borne diseases also include eradication of vectors like mosquitoes that cause disease spread. Computational models can be used to assess the feasibility and effectiveness of such proposed interventions. Computational models can also be used to identify critical subpopulations, which when intervened can inhibit the spread of an epidemic.

In traditional modeling environments, epidemiological models and simulations are stored as high-performance computational code, and require programming expertise to edit the code for trying out experiments with different sets of parameters or intervention strategies. This makes the computational environment not only difficult for epidemiologists to use, but also inefficient with respect to the human effort involved and time to study new epidemic scenarios. Computational modeling environments must be accessible through a web-based interface for ease of use and adoption by a large community of health experts and epidemiologists, who may not be computational experts. The environment should also be interactive, so that users can pause the system for analysis of the current extent of an epidemic, then continue with a new set of parameters for execution. Such interactivity and flexibility through dynamic parameters for executing various propagation and intervention simulation scenarios is an important requirement of an epidemic modeling environment. This is lacking in many of the existing computational epidemiological systems.

Another limitation of the current epidemiological modeling environments is the lack of flexibility and adaptability for the quick incorporation of real-time data into the system for evaluation of ongoing and evolving epidemics of various diseases such as Ebola. Data about ongoing epidemics can be obtained from traditional sources like the Centers for Disease Control and Prevention (CDC), patient data from hospitals, and so on. This process of collecting surveillance information is relatively slow. Recent advances in social media have facilitated collection of near-real-time data on disease outbreaks. Google Flu Trends (GFT), based on keyword searches of flu using the Google Search Engine, is a notable example of this approach. Real-time information integration is an important requirement of modeling environments for quickly adapting the computational experiments to real-world data so that ongoing epidemics, like Ebola, MERS (Middle East respiratory syndrome) etc., and their effects can be studied by public health decision makers in a short period of time.

In order to be used as a tool for real-time situation assessment and information dissemination, an epidemiological modeling environment should provide web-based access for reaching a large number of people, not just policy analysts and epidemiologists, so that they can take precautionary measures. Use of computational models and simulations is not limited to evaluating and assessing the current state of an epidemic alone. There is increasing research in the computing world for using models and simulations for predictive analysis based on real-world data, in addition to using them for descriptive analysis. Scientists are advocating the use of data for predictive analysis. Some researchers including Haas et al. [51] have argued that static data, which only provides information about the current state of the system, is dead or has no value, since it does not provide any new insights into the system dynamics. In the field of epidemiology, computational models implemented using epidemiological simulations can be combined with surveillance data obtained from the real world to forecast the future state of epidemics. Such predictive analysis can then be used by public health decision makers to undertake targeted preventive measures for containing epidemics, and to plan for the allocation of limited resources. There are several types of forecasting models that can be used for making predictions. Nsoesie et al. [72] provide a comprehensive literature review on the different techniques and models that have been used by different groups for forecasting epidemics. A modeling environment should be flexible to support these different types of forecasting models without the need for a major reengineering effort.

We have designed and developed an integrated epidemic modeling environment that incorporates all the important features listed above, including interactivity, ease-of-use, real-time information integration, flexibility, adaptability, extendibility, scalability, usability and accessibility for epidemic evaluation and forecasting. This environment provides services that allow for the development of different web and social applications around it. Using these services, users can evaluate the current state of epidemics around the world, and request epidemic forecasts for the future at fine-grained levels. The environment is based on a decoupled and modular architecture, and builds on the basic concept introduced in INDEMICS [18], where the intervention implementation is decoupled from the propagation simulation execution in a relational database.

Figure 1.1 shows the functionality of the modeling environment as a coordinating mechanism between various user applications, computing resources, and data repositories, as well as models,

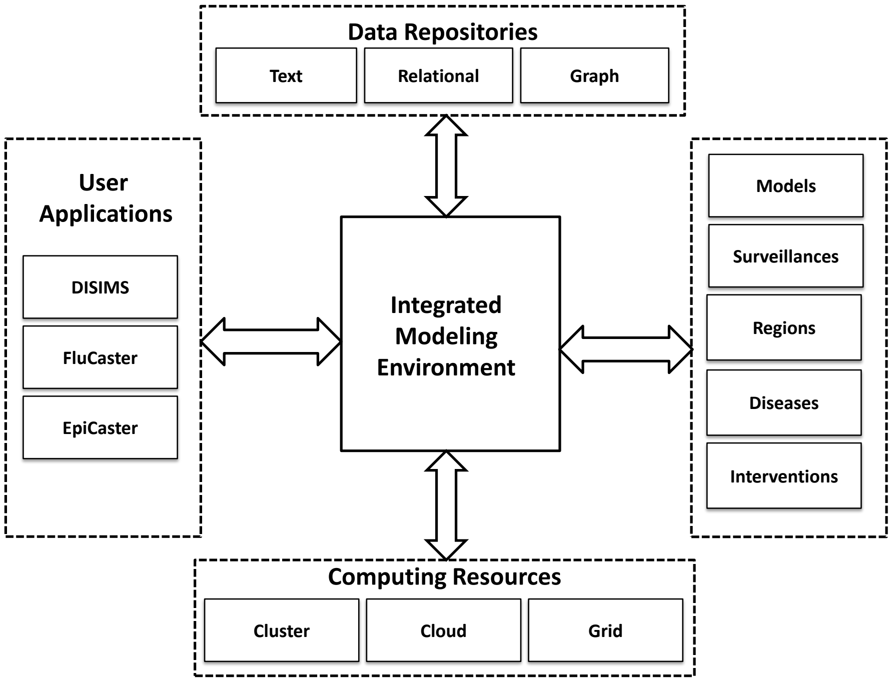


Figure 1.1: An integrated modeling environment that supports a variety of web applications, data sets, computing resources, models, measures and diseases for evaluation and forecasting of epidemics

measures and analyses. The main components of this integrated modeling and analytics environment include the backend infrastructure, databases and middleware platform. The backend infrastructure handles the execution of different epidemiological models through simulations. The databases store all types of data related to epidemics, such as disease surveillance data, regional demographics data and interventions data. The middleware handles the flow of data and context between different component systems.

This thesis is based on two main components that form the backbone of the integrated analytical environment - data and middleware services. We have extended the use of relational databases to add flexibility for the swift integration of real-time information into the simulation system, in addition to using it for intervention execution. We demonstrate how the data representation techniques that we have implemented allow new information on evolving epidemics to be integrated into the system quickly so that it can be made available for analysis. Relational databases also facilitate storage of intermediate information into the forecasting pipeline for regional and individual-level epidemic forecasting. We have developed REST-based web services, that are flexible and extensible, for providing access to information related to epidemic situation assessment and forecasting.

We demonstrate the use of our techniques through the development of two web and social applications, DISIMS and EpiCaster, that are built on top of the analytical environment.

DISIMS is a planning and evaluation tool for policy analysts and epidemiologists that allows users to execute complex epidemic simulations with dynamic intervention strategies. The tool provides interactivity by allowing users to start, stop, pause, resume and roll back previously applied intervention strategies and disease propagation processes. Users can ask complicated spatio-temporal queries in support of situation assessment, and submit new execution parameters. Unlike traditional modeling tools, which require access to computational resources for execution and programming knowledge to modify the parameters of execution, the web application we have developed provides an interactive web-based interface to the simulation system. This web interface provides all the capabilities for executing complex intervention experiments without the need for computing expertise, thereby improving user productivity.

EpiCaster is a social application for fine-grained situation assessment and forecasting of infectious diseases. The backend of EpiCaster is comprised of: (i) a high performance computing based simulation of disease progression (ii) a forecasting module that combines a digital library of precomputed forecast results, with a non-linear optimization process to infer disease propagation parameters for non-matching inputs, and iii) a set of databases that are used to store both the detailed assessments of ongoing epidemics, as well as forecasts. Using EpiCaster, users can view the current state of different epidemics around the world, such as Ebola and flu, through heat maps and graphical plots. Using this information, users can make appropriate behavioral adaptations and/or clinical interventions to mitigate the risks of the epidemic spread.

Going forward, the goal of EpiCaster is to collect information from registered users about their health states, combined with their behavioral adaptations in response to ongoing epidemics. Using this information, targeted individualized forecasts can be produced for registered users. Such individual-level forecasting of epidemics is a complex problem and is a part of ongoing research work. The process involves mapping of users in the modeling environment to their synthetic representation, that can capture their social, behavioral, demographic and geographic attributes. This can be done using machine learning techniques such as clustering to enable large-scale data processing. The process also involves devising techniques to compute individual-level forecasts based on the large scale pre-processing. We describe some of the ongoing work in this area in the last Chapter.

Overall, this dissertation focuses on the data integration methodologies and services as part of an interactive, flexible, and scalable epidemic modeling environment for the evaluation and forecasting of a range of epidemics in different regions of the world. The thesis describes the design, architecture and modeling extensions used for supporting these features. The main contribution of this thesis is the development of a flexible and extensible big data architectural platform for analysis and forecasting of various epidemics, using some of the established technologies, such as relational databases and RESTful web services.

Chapter 2

Background and Related Work

2.1 Computational Models for Epidemiology

Computational models used in the study of epidemics are comprised of two broad categories – aggregate-based models and individual-based network models.

Aggregate-based models partition individuals into separate classes depending on the model of infection dynamics chosen. There are several models of infection dynamics defined in the literature, including SEIR model, SIR model, SIS model and so on. S, E, I and R correspond to Susceptible, Exposed, Infectious, and Removed states respectively. These models assume homogeneous mixing of populations and use differential equations to study the changes in epidemic states of populations over time.

Aggregate-based models of epidemic simulation are implemented using ordinary differential equations (ODE). Hence they are also referred to as ODE-based models. In an ODE-based SEIR model, the population (N) is partitioned into four compartments: Susceptible, Exposed, Infectious, and Removed. Within each compartment, people are homogeneous and fully mixed. If β is the transmission rate, α is the rate at which an exposed person becomes infectious, and ν is the removal rate, then the following equations describe the dynamics of the epidemic in the population. An ODE-based simulation based on these equations computes the number of Susceptible, Exposed, Infectious, and Removed individuals at each time step (typically represented as a day). So, the model is able to produce aggregated counts of infected individuals in different regions every day. The model is also able to assimilate forecasting in the presence of interventions applied on a subset of the population, given by a certain percentage of that population.

$$\frac{dS}{dt} = -\beta\frac{I}{N}S \tag{2.1}$$
$$\frac{dE}{dt} = \beta\frac{I}{N}S - \alpha E \tag{2.2}$$
$$\frac{dI}{dt} = \alpha E - \nu I \tag{2.3}$$
$$\frac{dR}{dt} = \nu I \tag{2.4}$$
$$S + E + I + R = N \tag{2.5}$$

Some examples of aggregate differential equation based models used for studying disease propagation include epidemiological models developed by Longini et al. [56,86]. Another tool based on this differential equation based approach is the open source tool Spatio-Temporal Epidemiological Modeler- STEM [39], developed by IBM in collaboration with Eclipse, Johns Hopkins University and others.

Individual-based network models explicitly represent individuals within a population as autonomous agents. Hence, they are also referred to as agent-based models (ABM models). The individuals are connected to each other, and therefore exposed to each other via an interaction network, which form the edges of the graph. Each node has different demographic, geographic, and economic attributes. Hence the nodes are heterogeneous. The edges represent contact through physical proximity, and have durations associated with them. The disease transmits from node u to v only if u is infectious, v is susceptible, and u comes in contact with v. The probability of transmission is given by

$$p(u, v) = 1 - (1 - \tau)^{d(u,v)}$$

where τ is the probability of transmission in every unit time of contact, and $d(u, v)$ is the contact duration. An ABM simulation computes the disease transmission from node to node based on this probability. The above equation represents ABM model as a discrete-time simulation; however, there may be ABM models that use continuous-time simulations, which represent probability of transmission over a continuous time period.

Recent research in the area of ABM models includes work by Myers et al. [67], Dimitrov et al. [68], Meyers et al. [78], Barrat et al. [10], and Newman et al. [69]. This work involves deriving closed form analytical results on random graphs for finding epidemiological patterns of interest. Another type of individual-based model uses important statistics of a region, such as density of individuals in a region using land scan data and basic census information, to get the demographic distribution of individuals in a region in order to model epidemic propagation. Research in this area includes work by Germann et al. [49] and Ferguson et al. [44, 45]. Some researchers have also explored a hybrid approach where counties are represented as nodes, and the movement of individuals represent the edges. Coupled rate simulations are used for propagation simulation between counties. Example of a high-performance agent-based simulation based on this approach is the Global-Scale Agent Model (GSAM) [77], which simulates propagation of epidemics over billions of agents.

The last category of individual-based models incorporate individual level interactions between people based on their day-to-day activities, and generate a graph of social contact networks over which epidemics propagate. Research in this area includes work by Keeling et al. [59] and Meyers et al. [67]. Some of the earlier research performed in our lab, including the work on EpiSimdemics [12], EpiSim [41], EpiFast [19] and INDEMICS [18] also broadly fits in this category. EpiFast [19] uses high-performance computational code to implement individual-level interactions affecting disease propagation and intervention strategies for disease containment. INDEMICS [18] uses relational databases for intervention implementation on individuals and high-performance computational code for propagation simulation.

There are multiple advantages of using ABM models over ODE-based models. First, the ABM approach is able to represent fine-grained population characteristics of people within a region without assuming homogeneous mixing of the population as is done in the ODE-based approach. Secondly, ABM simulations allow representation of interventions for subsets of populations based on demographic characteristics, while ODE-based models can examine propagation based on a certain percentage of the population. Since each individual in an ABM is represented as a node in the network, pharmaceutical and non-pharmaceutical interventions can be applied based on individual characteristics, and may result in a change in the health state and behavioral attributes of individuals. For example, pharmaceutical interventions such as vaccination decrease the probability of transmission on the nodes being intervened, while non-pharmaceutical interventions such as school closures remove the contact edges between the nodes being intervened. This allows representation of targeted interventions, such as vaccination of adults in a certain zip code, and school-closures in a particular block group of a county. Such fine-grained interventions are not possible in the ODE-based approach since there is an assumption of homogeneous mixing of populations. Thirdly, the type of metrics that can be evaluated using the ABM approach are much more detailed than the ODE-based approach. For instance, the ABM models not only support forecast metrics such as aggregated infection counts, but also infection counts by demographic characteristics, such as age-groups, gender and so on. This allows analysis of forecast results, such as projecting the infections among females in a population, or forecasting the effects of an ongoing epidemic on school-aged children.

The difficulty associated with the ABM simulation, however, is that it requires a realistic representation of the contact network for every region. The scale of data involved to represent all the individuals of a region, along with their health, geographical, and behavioral characteristics, is much higher than storing aggregated information on infections in ODE-based simulations. We have constructed synthetic populations and synthetic contact networks for several countries of the world, including USA, Mexico, Liberia, Sierra Leone, Peru, India and Israel. The synthetic populations that we have developed, provide a realistic representation of the actual global populations. We execute ABM simulations over these synthetic populations representing all the individuals of that region. For countries where we have not built synthetic contact networks due to lack of information availability, we rely on ODE-based simulations to compute epidemic curves from models.

2.2 Web and User Interfaces for Epidemic Modeling

Recent work in computing technologies and ubiquitous computing has enabled relatively easy access to remote computing resources, including grids, clouds, or clusters. One such recent work by Parashar et al. [8], also discusses providing access to high performance computing resources as a service. There has been significant research in the area of accessing models and simulations through web-based systems, so that simulation complexity is hidden from users. Research has also been conducted in the broader area of applying different visualization techniques for representing high-end simulations on web-based user interfaces. Some examples of graph visualization techniques and tools developed include Gephi [13] and Pajek [14]. Easy accessibility through visualization is particularly important in public health epidemiology because domain specialists have limited technical expertise to execute and analyze complex simulations on high end computing platforms like clusters and grids.

Some examples of research conducted in developing web-based systems and visualization platforms for epidemic simulations include Epinome [60], Gryphon [104], FRED (Framework for Reconstructing Epidemiological Dynamics) [57], GLEaMviz [23] and SIBEL (formerly known as ISIS – Interface to Synthetic Information Systems. ISIS was recently renamed to SIBEL, and will be referred to as SIBEL for the rest of this document) [16]. Epinome is a user-centric system with visual analytics support for epidemiology that helps users evaluate intervention strategies based on available information. Gryphon presents a modeling environment to represent the geographic spread of the SARS (Severe Acute Respiratory Syndrome) outbreak based on published data, but it supports relatively smaller scale models. FRED is an open source modeling system developed collaboratively by University of Pittsburgh and Carnegie Mellon University that captures interacting effects of mitigation strategies, behavioral changes of people, and the evolution of the virus. GLEaMviz is a desktop-based visualization and analytical application that simulates disease propagation based on integration of data at three levels - data on global population, data on population mobility, and a model of the infection based on disease dynamics. Neither FRED nor GLEaMviz, however, provide a web-based interface for easy adoption by epidemiologists and health professionals.

SIBEL [16], one of the web interface tools developed by our lab, allows execution of a limited set of high-performance simulation models through a user-friendly web-based interface. We have developed a modeling environment that not only embeds the features of SIBEL for visualization, but also provides additional interactive features, such as pause-resume-roll and access to improved simulation models, including those for epidemic forecasting at a population and individual level. DISIMS is a web application that we have developed on top of the modeling environment, for evaluation of dynamic interventions in epidemic propagation. DISIMS also provides interactive computations and analytical visualizations, thereby improving the productivity of analysts.

Table 2.1 compares the features of existing epidemic systems with DISIMS based on several design principles. As far as we know based on published literature, DISIMS is the only system that can support and execute large scale individual-based epidemic simulations interactively, through a

System Name	**Web interface**	**User interactivity**	**Scalability**
FRED [57]	No	No	Support for large scale epidemic models
EPINOME [60]	Yes	Partial	Pre-run simulated disease outbreak models
Gryphon [104]	Yes	Yes	Relatively smaller scale models
GSAM [77]	No	No	Highly scalable with high resolution models
Flute [28]	No	No	Large scale individual-based models
GLEaMviz [23]	Desktop based visualizations	Yes	Large scale hybrid models
DISIMS	Yes	Yes	Large scale high resolution individual-based models

Table 2.1: Comparison of features of DISIMS with other UI-based epidemiological tools

web-based interface.

2.3 Research in Social Computing for Situation Assessment

Recent advances in social media have facilitated collection of near-real-time intelligence on epidemic outbreaks. Research in this area includes real-time epidemic surveillance through processing and analyzing of online news sites, discussion forums, Twitter data and so on for situation assessment. Published research in this space includes work by Salathe et al. [87], Chunara et al. [31] and Brownstein et al. [24], [25].

The models and techniques used in the design of socially-enabled epidemiological modeling environments derive several concepts from the emerging fields of social computing, human computations and crowd-sourcing. The term “human computation”, in the context of modern computing, was introduced by Luis von Ahn in his doctoral dissertation. He largely describes human computation as a technique that allows humans to solve tasks which cannot be solved by computers alone. The term “crowd-sourcing”, coined by Jeff Howe [54], involves using a large group of people to perform a particular task, which is traditionally performed by an expert or a designated agent. Technology platforms like wikis, blogs and social media websites have popularized the term “social computing” for dealing with collective action and social interaction. In the context of epidemic surveillance, these technology platforms can be major sources of data gathering.

Some of the collective intelligence techniques gathered from large populations can be combined with data mining techniques to find patterns and make sense of the data. For example, Kamar et al. [58] have explored use of machine learning tasks along with human computations supported by large-scale crowd-sourcing in the classification of celestial bodies for a science project called Galaxy Zoo. Vijayanarasimhan et al. [97] have applied active learning along with large-scale crowd-sourcing for building training data for image recognition.

In the context of using social data for epidemic evaluation and forecasting, several models have been developed for modeling and representation of flu outbreaks in different regions of the world. Some notable examples include Google Flu Trends (GFT), developed by Larry Brilliant et al., that uses search terms entered by users to develop statistical models for influenza forecasts [50]. Researchers like Butler et a. [26] have explained some of the shortcomings of GFT-based predictions by analyzing the results produced by Google Flu Trends and explaining why the results overestimate the actual flu counts in populations.

Research in flu forecasting includes work by Shaman et al. [88], who have developed real-time forecasting models of seasonal influenza outbreaks, and applied it to derive retrospective ensemble forecasts using techniques which are common in numerical weather predictions. Similar to weather-driven forecasts, work is also presented by Yang et al. studying the relationship between humidity and influenza [100]. Yang et al. [102] have also described inference systems for influenza that can handle the observational biases and errors arising from the epidemic surveillance data gathered from the Centers for Disease Control and Prevention (CDC) and Google Flu Trends. They have used Bayesian Inference techniques to derive such forecasts. Osgood et al. [74] have demonstrated the use of particle filtering techniques to robustly tolerate the noise in the epidemiological data feeds used in forecasting simulations, without making significant assumptions on the epidemiological processes used. Work by Vespignani et al. [95] includes using Monte Carlo Maximum Likelihood analysis for generating ensemble forecasts for different regions in the northern hemisphere.

There are also other models using data sources like Twitter and Wikipedia for gathering health-related data. Paul et al. [79] show the use of social media messages for predicting a variety of ailments and conditions, including obesity, allergies, and other ailments. Sugumaran et al. [91] have used Twitter data for spatio-temporal analysis of West Nile virus. These models have shown that health and epidemiological events can be forecasted using a combination of mathematical models and diverse data sources. There has been a flurry of activity in this field, including work by Tamerius et al. [93], Yuan et al. [105], Nsoesie et al. [73], Chakraborty et al. [30], and so on. Recently, Paul et al. [80] have shown that using Twitter data actually improves the flu forecast, reducing forecasting error by 17-30 percent over baseline models using only historical data. They have also demonstrated that using Twitter data provides better forecasts compared to GFT data. Hickmann et al. [53] have demonstrated the use of Wikipedia access logs together with the weekly CDC reports to derive forecast results for flu ahead of the peak flu season. There are several such flu forecasting models developed by different groups. Yang et al. [101] have provided a summary of the comparison of forecast results obtained using six different filtering techniques for making retrospective flu predictions, including the maximum likelihood estimation via iterated filtering (MIF), ensemble filters such as the ensemble Kalman filter (EnKF), the ensemble adjustment

Kalman filter (EAKF), and the rank histogram filter (RHF).

Nsoesie et al. [72] provide a comprehensive literature review on forecasting techniques used for studying the dynamics of influenza outbreaks. Most of the work so far has focused on using passive data for forecasting. Furthermore, the forecasts produced use fairly simple statistical models to achieve the objective.

Even though there has been substantial research for collection of epidemic data and using it for visualization for nowcasting of current epidemics, the research on using such information from diverse sources for generating high-resolution forecasts is still at a nascent stage.

Most of the epidemic forecasting models so far have focused on coarse-grained forecasting, such as predicting the epidemic peak time, intensity, attack rate, and so on. In addition, a majority of the epidemic forecasting models existing today are based on retrospective data, where the predictions are made retrospectively on past outbreaks. For instance, Nishiura et al. [70] have described a technique for real-time forecasting of the 2009 H1N1 epidemic using a discrete time stochastic model.

Some of the methodologies published in the literature for forecasting in related contexts include the method of analogues, the Bayesian method of predictive analysis, and the Extended Kalman Filter (EKF) approach.

The method of analogues for forecasting was first introduced in meteorology and is widely used in weather forecasting. This is a non-parametric approach to forecasting where the forecast of the future is computed based on finding an analogous match in the time series in the past. The method of analogues for epidemic forecasting was described in the paper by Viboud et al. [96]. In this methodology, the best match of the current epidemic time series is found by matching the time series of epidemics observed in the past for a given region by using a distance function. The distance function is given by $dist(X(T), X(t)) = \sum_{j=0}^{p}(I(t-j) - I(T-j))^2$ where p is a time-step in the past for which epidemic data is available; T is the current time step; and I(T) represents the current time-step time series. After a set of best matches of time steps are found from the past, the algorithm simply finds the weighted average of the future time series for the selected time steps, and assigns it to the forecast.

Osgood et al. [82] have described an Extended Kalman Filter (EKF) approach for regrounding of epidemic simulations based on empirical data about ongoing epidemics. Empirical data about an outbreak is generated through agent-based simulations over empirical microcontact networks, and is used as input to revise the results of an EKF-corrected aggregate model. Bayesian ensemble modeling approach has been applied by Shaman et al. [88] using a data assimilation method called the ensemble adjustment Kalman filter (EAKF) for epidemic forecasting. GFT data is used to train the model. A temporally evolving ensemble of model simulations is created and a posterior estimate of the model is derived using the EAKF technique.

Research in epidemic forecasting models is not limited to flu alone. Several models have also been developed for other types of infectious diseases, such as Ebola and Malaria. The recent outbreak of Ebola in West Africa made headlines for its large number of casualties. Public policy makers needed tools and techniques to forecast the epidemic spread in the affected countries and other regions of the world. Epidemiological models were developed to assist decision makers for planning purposes. Some researchers including Lofgren et al. [61] and Rivers et al. [84] described how mathematical models are the key tools for an effective Ebola epidemic response. Rivers et al. [85] modeled the effects of intervention strategies in containing the spread of Ebola in Sierra Leone and Liberia. Researchers including Merler et al. [66] studied the effectiveness of non-pharmaceutical interventions in controlling spatio-temporal propagation of Ebola. Researchers also used dynamic modeling and Bayesian inference methods based on observations to generate weekly forecasts of the Ebola outbreaks in Guinea, Liberia and Sierra Leone. This includes the work by Shaman et al. [90]. Use of species distribution models for mapping the zoonotic niche of the Ebola virus in Africa was demonstrated by Pigott et al. [81]. Extensive research on Ebola epidemic modeling and mitigation strategies was also conducted by the Network Dynamics and Simulation Science Laboratory at Virginia Tech [6]. Several of these models have been used to study the Ebola outbreak in different regions of West Africa, under different circumstances.

2.4 Web Tools for Epidemic Situation Assessment and Forecasting

In addition to epidemic models and simulations, it is important for researchers to develop tools and techniques that can be used by public health decision makers and epidemiologists for situation assessment of the current extent of epidemics and view trends of the epidemic spread on maps and graphical plots. These tools might use data from different surveillance sources and epidemiological forecasting models, explained in the previous sections.

We list some of the tools and websites that have been developed in recent years that use social media and other related technologies for studying disease propagation. Some of these tools present visual dynamics of epidemics, including situation assessment (also referred to as nowcasting) and forecasting.

- Google Flu Trends: Google Flu Trends (GFT) uses the search terms entered by users through the Google search engine to make predictions about the outbreak of flu at different regional levels. GFT can provide estimates of aggregate flu trends at the state and city level for the USA, and at the country level for several other nations. GFT provides weekly counts of flu forecasts in aggregated form in a csv file format. [This service has been recently discontinued by Google.]

- Flu Near You: Flu Near You [2] is an Open Surveillance System to collect and disseminate disease information. Information is collected from users about local flu activity through weekly surveys. The application also uses flu information published by the CDC and Google Flu Trends to show disease spread over time in the form of maps and charts.

- HealthMap: HealthMap [4] uses information gathering techniques about infectious diseases in different global regions to map the evolving epidemic status. It collects data from multiple sources and provides a comprehensive view of the disease propagation in a region. This information is used for monitoring an outbreak and for real-time surveillance of emerging disease threats. HealthMap is also available as a smart phone application that can be used by a large group of people for reporting and tracking diseases.

- Columbia Prediction of Infectious Diseases: [1] This Web application, which is based on research by Shaman et al. [89], shows current ILI counts in certain select cities of the US on a map, and displays seasonal flu forecasts as an epidemic curve.

- FluOutlook: FluOutlook [3], developed by Vespignani et al., provides a Web interface to view forecasts of the current influenza season in North America and Europe through maps and charts. The underlying model of disease propagation is based on aggregate-based models of epidemic simulation.

- MappyHealth: MappyHealth [5] tracks disease trends using keywords on Twitter for tracking and prediction of disease propagations. Trends on different disease conditions are collected along with location information. The application also ranks the top five trending illnesses in a specified region.

- FluPhone: The FluPhone project by Yoneki et al. at Cambridge [103] involves the study of infectious diseases and behavioral responses by using mobile phones to derive contact information and interpret how it affects the spread of flu. The smart phone application collects information on flu symptoms as well as person-to-person contact information.

- iepi: Hashemian, Osgood et al. have explored the use of cell phones to derive contact networks by developing an application called iepi [52]. iepi collects infection information through its sensors based on physical proximity. In this work, the authors have also explored complementing the epidemic computational models with the collected infection information, so that the two can yield powerful decision-making tools.

- EpiCollect: EpiCollect [7] is a Web application that uses features such as GPS and Google Maps available on smart phones to send epidemic-related information to a centralized web database. This project has created an ecology of community data collection for gathering epidemic data.

The applications that we have developed in our research including FluCaster and EpiCaster share some common features with the applications listed above, such as, visualization of flu magnitude through heat maps and plots and count of flu infections in specific regions.

The distinguishing feature of our platform and the forecasting tools built on top of them is that the platform is not tied to a single data source or forecasting model. The platform supports use of agent-based models for forecasting in addition to ODE-based models, producing fine-grained results of epidemic forecasts for different regions of the world. This allows analysts and epidemiologists to analyze the forecast results at multiple spatial, temporal and social resolutions. This is a big advantage in comparison with most of the above applications that use compartmentalized models of forecasting producing only aggregate results of infections.

In addition, using our platform, it is possible to analyze forecast results filtered by specific demographic sub-populations, over a defined period of time. The underlying forecasting models, data schema and architecture provide the ability to produce such analytical forecasts. Our platform also produces forecasts in the presence of targeted intervention strategies applied to subpopulations of a region, thus allowing advanced predictive analytics, which is lacking in the above epidemiological forecasting tools.

Chapter 3

Experimental Design and Set-up for Large Scale Epidemiology

As described in Chapter 2, the main types of epidemiological models used in the study of epidemics include network-based epidemic models and aggregate-based epidemic models. These models help in mathematical formulation of the propagation diffusion problem and allow design of experiments around it. These models also help in the study of mitigation strategies in disease containment. Aggregate-based epidemic models assume homogeneous mixing of populations and hence provide coarse-grained analysis in the study of epidemics. Individual-based models, on the other hand, can model individual-level interactions in a social contact network over which the diseases propagate. The underlying network that forms the fabric of disease propagation is different for different diseases. For example, for a disease like flu, the contact network is derived from physical proximity, whereas for a vector-borne disease like malaria, the contact network is based on exposure to vectors such as mosquitoes. In addition, individual-based models also allow representation of behavioral interventions for disease containment. These may include individual behavioral interventions such as use of face-masks or policy level interventions such as a county-level school closure policy enforced by the Government.

The propagation dynamics of every disease is different and is based on the vector of disease transmission. It is also based on the presence of different mitigation strategies for inhibiting disease spread. Hence, most of the propagation models studied in the literature focus on modeling the propagation of a single infectious disease. Even though from a modeling standpoint, every disease has different parameters of execution for evaluation of propagation dynamics and intervention strategies, different epidemics of infectious diseases are governed by similar set of rules and modeling principles such as propagation dynamics through physical proximity or through vectors, and health, demographic and behavioral characteristics of populations. The experimental design and set-up required for studying epidemics of various contagious diseases is also similar. It involves analysis of prevalence of incident cases and infection-causing pathogens, number of epidemic-related deaths and so on.

To be able to study the propagation dynamics and intervention strategies of different contagious diseases through a common platform, we abstract out the common features in all the diseases related to epidemic evaluation. In the subsequent section, we describe the common terminologies and terms that are used in the context of experimental set-up for epidemic simulations.

3.1 Experimental Set-up for Epidemic Analysis

Epidemiological models and simulations are based on stochastic processes due to the probabilistic nature of epidemic propagation dynamics. To minimize the errors involved in such stochastic process representation, an epidemiological experiment is set-up as a factorial design with several parameters of execution.

A single instance representing a combination of parameters is referred as a scenario. For a factorial design, an experiment is conducted over several scenarios. A complete experimental study involves several experiments.

We describe the terms involved in the study of an epidemiological experiment as follows:

Cell: A cell represents a single configuration with a combination of values of parameters. There may be several cells corresponding to a particular scenario based on the valid values of parameters.

Replicate: A replicate is the number of times an experiment execution is repeated to reduce variability in the results. Each execution corresponding to a replicate executes with a different random seed.

Intervention: Intervention refers to the mitigation policy applied to a subpopulation at a policy or individual level to contain an epidemic from propagation. Intervention is identified by other parameters, such as subpopulation, efficacy, compliance, trigger and delay, which are described below.

Subpopulation: Subpopulation refers to a specific subpopulation of a geographical region, where an epidemic is propagating, on which an intervention is applied.

Efficacy: Efficacy is given by the effectiveness of an intervention strategy to contain an epidemic. Efficacy is measured by a percentage value that describes to what extent an intervention strategy is effective in containing an epidemic.

Compliance Rate: Compliance Rate is given by the percentage of individuals in a subpopulation of a geographic region, that comply with an intervention strategy. For example, a 40 percent compliance rate of vaccination in Montgomery county of Virginia implies that 40 percent of individuals in Montgomery county of Virginia have been vaccinated.

Trigger: Trigger refers to the type of event that triggers an intervention strategy. Trigger is represented by a trigger event, subpopulation and delay period. For example, a trigger can be described by an event, where the percentage of a subpopulation of a county becomes infected, triggering the need for an intervention. Delay period refers to a delay, given by the number of days, in implementing an intervention. Trigger is either a timed trigger or is driven by a threshold. An example of a timed trigger is application of intervention after a specified number of days have passed. A trigger driven by threshold is applied when certain epidemiological measures such as the infection levels among subpopulations cross a particular threshold.

A single factorial experiment design for studying a particular intervention strategy in a region for a specific disease involves several cells corresponding to a range of values for the parameter types described above, such as sub populations, triggers, intervention types and so on. The number of replicates determine the number of times the experiment would be repeated every time with random "seeds" or infected population to reduce the variability in the results. The scale of computations involved for a single experiment is thus very large. Since a complete experimental study can have several such experiments, the scale of data and computations can increase exponentially.

The experiments are both data and compute intensive and require high performance computing resources to be able to generate results of the simulation in real time.

In our research, we have used synthetic information systems as the backbone of demographic information representation for different regions. This synthetic data is built using census data as described in [11]. Using synthetic information systems in study of epidemic propagation allows fine-grained analysis at an individual level.

We describe the data formats and storage of synthetic information systems in Chapter 4.

3.2 Experimental Set-up for Forecasting

System analysts and epidemiologists need access to epidemiological environments not only for studying different mitigation strategies, but also to get an insight into how the epidemic might unfold in the future, with or without any interventions.

This is crucial for diseases like Ebola, where the analysts have to work with a limited amount of information, and need real-time forecasts to implement feasible intervention strategies.

The basic requirements for experimental set-up for forecasting are similar to that for epidemic evaluation as explained in Section 3.1. However, the data requirements for forecasting are much more than those involved in epidemic evaluation. Firstly, forecasting requires up-to-date information on current state of epidemic to be able to forecast its future direction. So, in addition to the static data about demographics of a region, which may be derived from synthetic information systems, the forecasting simulation needs up-to-date information on current extent of epidemics.

We propose using synthetic information systems to model populations of different regions and use it together with real-time epidemic information for generating epidemic forecasts. To accomplish this, we have implemented a mapping mechanism to translate the real-time surveillance data to synthetic information systems. We first describe the epidemic modeling and generation process and then explain the experimental parameters used for forecasting.

3.2.1 Epidemic Forecast Modeling and Generation

Epidemic forecast modeling is a wide area of research. The main objectives of forecasting in the context of epidemics involves estimation of epidemiological measures, such as peak timing, peak number of infections, total number of infections, start and end of epidemic season and so on. Estimation of these epidemiological measures allows analysts and epidemiologists to take precautionary measures ahead of the epidemic reaching peak levels. As described in the literature, following are the main types of epidemic forecasting models that have been used for forecasting of epidemics. These models have also been described in detail in the paper by Nsoeie et al. [72].

1. Time-series models: These models predict future forecast values based on past observations. Typically, autoregressive moving average models are used as time-series models.
2. Method of Analogs: This approach involves matching current infection patterns to historically observed patterns, without making any assumptions on the underlying distributions. This is a non-parametric approach.
3. Compartmental models: These models assume homogeneous mixing of populations where populations are divided into compartments to introduce subpopulations, such as Susceptible, Infectious and Recovered.
4. Agent-based models: These models simulate the behavior of individuals of a population, represented as agents, which interact with each other based on predefined rules.
5. Metapopulation models: These models represent populations as discrete patches, and the subpopulations interact through migration. The epidemic states of populations are described similar to compartmental models.

3.2.2 Epidemic Forecasting Pipeline

The epidemic forecasting pipeline that we have implemented is built to support a variety of epidemiological models and techniques that can derive epidemic forecasts based on advanced stochastic computations and optimizations. We describe these computations and optimizations in the following sections.

The epidemiological models and simulations that generate forecasts are the main building blocks of the pipeline. The pipeline is developed in such a way that it provides several techniques and methods for translations and transformations to handle the discrepancies in data generated from various models.

We use a simulation-driven optimization approach for the forecasting pipeline. From a modeling standpoint, this method has been described in detail by Nsoesie et al [71]. Our implementation of this approach has three main components:

1. models of infectious diseases and their corresponding simulation engines that produce highly resolved epidemic information. The models may be individual-based or aggregate-based simulations.
2. a stochastic programming-based method for parameter matching and searching based on input epidemic surveillance data, and
3. a model library comprised of a large number of plausible precomputed outcomes of epidemic forecasts using the first two steps.

Epidemic Models

Our forecasting pipeline implementation supports both aggregate-based models, implemented using ordinary differential equations (ODE); and agent based models (ABM) which represent disease propagation in a population through contact networks. The aggregate-based models are usually called compartmental models.

In an ODE-based model, the population (N) is partitioned into different compartments. For instance, in the SEIR model, there are four compartments: Susceptible (S), Exposed (E), Infectious (I), and Removed (R). Within each compartment, people are homogeneous and fully mixed.

In the ABM approach, the population is represented by a network where the nodes are represented as individuals and the edges are the contacts between the individuals. The nodes are heterogeneous and have demographic, geographic, and economic attributes.

Details of both these models have been provided earlier in the Background section.

Model Library

The simulation optimization process is a massive large-scale computation process. Hence, it is not just time-consuming, but also challenging to produce forecasts using this process in near-real time when surveillance data is available. To make the process more efficient, we have developed a pre-computed model library that produces a large set of possible forecast outcomes considering a range of epidemic parameters along with a variety of interventions. This pre-computation is done using EpiFast [19], a high-performance modeling environment developed for pandemic planning that is an implementation of ABM approach using synthetic data on regions. When new surveillance data comes in, this model library of pre-computed forecast outcomes is searched, and the best match is used for further computations, as described in the following subsection.

Parameter Matching and Search

We use a Bayesian approach for deriving the results of situation assessment and forecasting in different regions. Our model implementation is based on the simulation optimization approach to epidemic forecasting, described in [71].

With the time-series data obtained from surveillance about infection counts periodically (typically on a weekly basis), the pipeline either searches the precomputed epidemic curves in the model library (called the DL approach), or calls a derivative free optimization algorithm to search the model space; the purpose is to find a model that generates epidemic curves that are the most *similar* to the given surveillance curve.

Suppose $M \in \mathcal{M}$ is a model in a set of epidemiological models $\mathcal{M}$. Let M be given by $M = \{(p_1, p_2, \ldots, p_k) \in \mathbb{R}^k\}$, where p_i represents either epidemic parameters such as transmission rate and removal rate in the classical SEIR model that affect disease transmission, or intervention parameters such as compliance rate, efficacy of pharmaceutical interventions, duration of non-pharmaceutical interventions, timing of the interventions and so on. If $S \in \mathcal{S}$ represents an epidemic simulation, which takes a model $M \in \mathcal{M}$ as input and produces an epidemic curve or time-series of infection counts as output, then the output epidemic curve is given by $S(M) = X = (x_1, x_2, \ldots, x_T)$, where x_i represents the number of infections in week i.

The epidemic simulation is a stochastic process. Hence we have several possible epidemic curves from the simulation output, that represents the probabilistic nature of the output. Hence, $S(M)$ is computed across several replicates given by $X_1, \ldots, X_R$. Each output X represents infection counts computed at different geographical levels, per week in a single replicate. Currently, we use averaging techniques by finding a mean of the infection counts derived from all the replicates to arrive at the final forecast outcome. This is stored as an aggregated count of infections at different geographical levels for every region for which forecast is sought.

Let $y = (y_1, y_2, \ldots, y_T)$ be the observed epidemic curve from surveillance which represents weekly infection counts in a region. To compute situation assessment and forecast of the infection counts in the region based on surveillance, the first step is to find a matching set of models and corresponding epidemic parameters that represent the current epidemic. To achieve this for finding a matching model, the algorithm computes a distance measure between y and x. We have implemented Euclidean distance as a simple distance measure calculation between y and x in our current version, but the system supports a variety of distance measures to be used for this purpose. Based on the distance measure, if a matching model that represents the current epidemic time-series is found, then the output infection time-series curve for the corresponding model stored in the library is used as the forecast outcome. If there is no matching model, then we use an optimization process such as the Nelder-Mead optimization algorithm to derive epidemic parameters such as transmission rate and removal rate in the classical SEIR model. Using the parameters derived from the optimization process, a simulation is executed to derive the forecast results.

Figure 3.1 shows the forecasting pipeline of our platform from a process-oriented viewpoint.

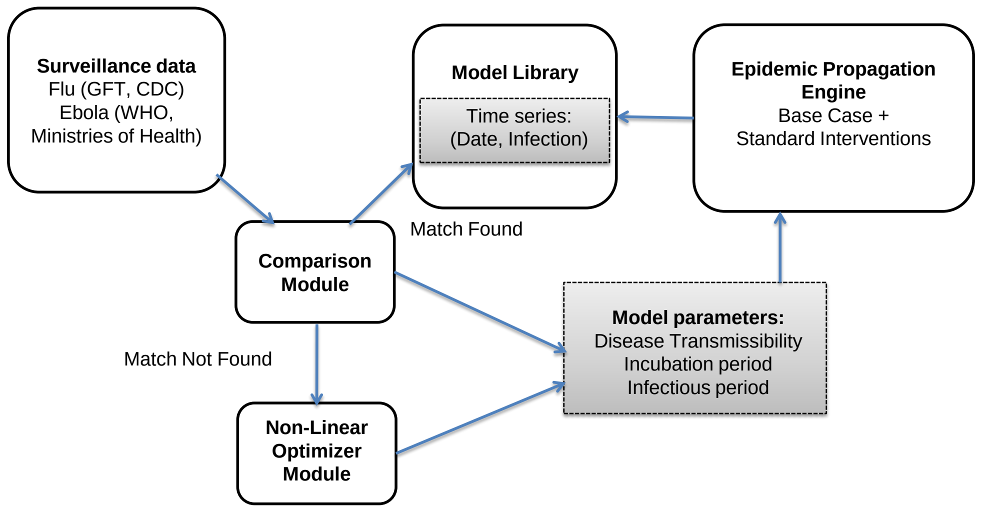


Figure 3.1: EpiCaster forecasting pipeline : Processing of epidemic data from surveillance data collection to epidemic forecast generation

The epidemic propagation engine executes epidemic simulations based on a range of epidemic time-series data used as input, and stores the forecast outcomes in the model library along with their model parameters, including disease transmissibility, incubation period and infectious period. Since we use synthetic population data for ABM simulations, we have detailed demographic information of every individual in each region. With the granularity of demographic information available, the forecast epidemic simulations can be executed with not just the base case of 'no interventions', but also with fine-grained interventions applied to certain sub-populations of the region. Some of the intervention strategies that can be supported for epidemic evaluation and forecasting include vaccination of a sub set of individuals of a population or evaluating behavioral changes by certain high-risk individuals. For instance, types of intervention strategies that can be analyzed include, finding infection counts when a certain percentage of school-age population is vaccinated, school-closures in a particular county of a region, anti-viral distribution among high-risk populations etc. The corresponding results of the epidemic simulation are also stored in the model library and used for parameter matching when intervention information is available from surveillance. This is a distinguishing feature of our system, that supports forecasting in the presence of mitigation strategies.

We describe the data model for supporting the forecasting pipeline in Chapter 4 and the architectural details of the pipeline in Chapter 5.

3.2.3 Epidemiological Forecast Measures for Experimental Analysis

The modeling environment that we have designed supports fine-grained forecasting. The main objective of our research is to not only provide key epidemiological measures such as epidemic peak timing and peak number of infections, but also to provide fine grained assessment based on demographics and effects of mitigation strategies. Using our approach, we are able to find out effects of epidemics based on population demographics, geographic spread and time-lines. Our approach is able to produce forecasts at a spatial, temporal and social granularity.

For instance, using our approach, analysts can analyze the effect of a current epidemic on school-age children or forecast the extent of an epidemic among female population. In addition, the system also is able to support analytics in the presence of fine-grained interventions such as vaccinations applied to senior citizens in a particular county with 30 percent compliance rate and so on.

To achieve this, we propose using agent-based models for fine-grained forecasting of epidemic infections along with aggregate ODE-based models. We use synthetic information systems in addition to real-time data as the backbone of our forecasting models.

In addition, for the system to support study of multiple diseases across various regions of the world at different levels of granularity and mitigation strategies poses a new set of computational and data requirements that needs to be handled through appropriate data representation and management techniques that we describe in Section 4.

Our forecast approach is based on epidemic simulations. Hence the terms involved in the study of an epidemiological experiment including the cell, replicate, efficacy, compliance and so on described before are also relevant to an epidemic forecasting experiment. The use of agent-based modeling to generate fine-grained forecasts is data and compute intensive similar to a single experiment described before for running intervention evaluations.

The main difference for a forecasting experiment from a simple epidemiological simulation is the presence of actual environmental information related to epidemic surveillance and the mapping required to map to an experimental evaluation along with the scale and magnitude of forecast data processing involved. An epidemic simulation experiment is typically executed online at run-time based on the parameters chosen for evaluation. A forecast experiment is based on real world data and hence can only be executed when surveillance information is available. Also, since the scale of data is very large based on time scale and regional demographics, forecasting experiments are usually pre-computed and then analysis can be carried out at run-time by analysts.

Firstly, since we use agent-based modeling using synthetic information systems, the mapping from the environment which is in an aggregated form to the synthetic information systems is critical. In the ABM approach, we use the aggregate count of infections obtained for a particular region from surveillance (for example, GFT data for flu) and randomly assign the same count of agents in the synthetic network as being in the "Infected" state and use the agents as random seends of infection. The ABM model is then executed using an HPC-based simulation such as EpiFast and forecast results are computed.

Secondly, we have developed a model on top of the forecast results to aggregate and present the forecast results to the users. The data representation of the forecast output is explained in detail in the Section 4. From an experimental set-up standpoint, aggregate-based models produce infection counts for different geographical regions at multiple levels.

In our approach, we have created separate data abstractions using relational database at a national level. Simulation output data generated at any level can be represented inside the table created for each nation. Typically, the simulations are run at either national level or at state level. For a large country like USA, the simulation is run at a state level (for example, California, Virginia, New York etc.) and the data from multiple tables corresponding to each state, is aggregated together.

In our current implementation, for aggregate epidemic counts, we average the counts generated across multiple replicates and derive the final forecast outcome. For individual-level forecasts, we use a model on top of the individual-level epidemic forecasts obtained using ABM forecasting models, that involves some averaging over the replicates. This fine-grained individual level forecast is explained in detail in the last Chapter of the thesis.

Chapter 4

Data Representation for Large-scale Epidemiology

Over the years, there have been a range of epidemiological models and tools that have been built by computational scientists to closely capture the dynamics of various epidemics in different regions of the world and to project or forecast epidemic propagation in the near future. Most of these analytical models are built for situation assessment of specific diseases and are able to capture disease propagation dynamics in specific regions, where the disease is prevalent. In addition, other factors that affect disease propagation dynamics, such as individual-level and policy-level mitigation strategies applied to contain the epidemic, are included as part of the model in many cases.

Hence, in computational systems built around such epidemiological models, the data is closely integrated with the model and there is limited flexibility to reuse the model for any other disease or region. Moreover, integrating policy-level or behavioral interventions with the model requires changes to the model, which is a time-consuming process. This imposes limitations on the number of epidemic propagation scenarios that can studied using the system.

We have built an end-to-end computational system that can allow public health decision makers and epidemiologists to study epidemics of different diseases that may be prevalent in different regions of the world at multiple geographical granularities by using a single web-based platform. Our platform provides flexibility to the users to request forecasts with or without application of dynamic intervention strategies and to view the forecasts through time-varying heat maps and graphical plots for ease of analysis.

The data involved in building this integrated platform comes from multiple sources and with diverse formats, such as time-series data of infections gathered from surveillance sources, graph-based data of the social-contact network over which the epidemic spreads, GIS data for map displays and relational data about population demographics in different regions.

In this chapter, we describe the data representation techniques and methodologies that we have implemented for building an end-to-end data analytics platform with diverse data sets, for evaluation and forecasting of epidemics.

4.1 Problem Definition

Our aim is to separate the implementation of the platform from the implementation of the model and provide the users flexibility to select any model of epidemic forecasting through a single web-based platform. The models may be large scale network-based models or differential equation based models based on simulations of disease propagation or causal models of epidemic forecasting. Most of the traditional epidemiological systems have the data completely embedded inside the model. This tight-coupling of data with the model and the simulation system makes the system inflexible to any changes or extensions in the type and format of data.

There is a need to build a platform where the users can request forecasts based on different types of forecasting models. The forecasts may be produced at multiple geographical granularities based on types of models used. The system should not only be able to handle the current set of diseases, such as such as Flu, Ebola, Malaria etc., but also be flexible enough to incorporate models for new diseases as they appear. The system must also be extendable to add new global regions for which infection data becomes available. In addition, the system needs to support collection of surveillance data through several different sources, such as the CDC, WHO, Ministries of Health, social media, and so on.

The data involved in building such an integrated platform comes from diverse sources and with diverse formats. The first set of data consists of the models used for epidemic propagation. These include models for epidemic situation assessment and forecasting, which in turn may be based on epidemic propagation models such as aggregate-based models and network-based models. The model data is closely tied with the simulations that execute the models on computing platforms. The second set of data consists of the surveillance data gathered from several sources of data collection, such as the CDC, Ministries of Health and so on, as a time-series of infection counts in different regions. The third set of data is the demographic data and contact network data associated with the regions under consideration, that affect the disease propagation dynamics, and hence, used as input to the simulation model. The last data set involves the output of the epidemic forecasting simulations and other associated data for displaying the results on the User Interface (UI). These include the GIS (Geographic Information Systems) data consisting of maps and geographic locations.

Here, we have described the data representation techniques and methodologies that we have implemented for building an end-to-end data analytics platform with diverse data sets, for evaluation and forecasting of epidemics. The main contribution of our work is the design and development of various data integration methodologies for representing and processing the "big data" associated with large scale epidemic situation assessment and forecasting.

Region Name	Population (in millions)	Network size (in millions)	Storage size
Virginia, USA	7.25	202.5	5.4 GB
Wyoming, USA	0.56	12.08	340 MB
California, USA	33.6	947.9	26GB
Liberia	4.2	84.8	2GB

Table 4.1: Population in million, contact network size and storage size of different regions used in the epidemiological experimental analysis

The accuracy of the epidemic forecasts is dependent on the veracity of the data and trends collected from various surveillance sources. The rate of data ingestion or the "velocity" of input data from surveillance sources such as the CDC, WHO and so on, adds more complexity to the scale of data in the forecasting pipeline. For instance, one of the epidemic surveillance sources that we use is the Google Flu Trends. This data is obtained on a weekly basis for different regions in a comma separated file format. The typical size of each file is around 500 KB and stores aggregated counts of infections.

In addition to surveillance data, we use synthetic data to represent population demographics of different regions under consideration. We also use a synthetic contact network data to represent contact patterns between individuals of a region. Table 4.1. shows the population and contact network sizes of Liberia and 3 regions of the USA, namely Virginia, California and Wyoming.

The storage and representation of the input surveillance and demographic data is critical for efficient execution of forecast epidemic simulations. For instance, to support different geographical and social resolutions of the forecasts, the regional populations need to be represented and stored at fine-grained levels, such as state, county, block group and so on, which makes the scale of data involved very large. Additionally, the data is replicated in multiple formats for ease of processing and retrieval. As seen in Table 4.1, the storage size for a contact network for Virginia is 5.4 GB and for a big state like California, the contact network occupies 26 GB of space, when stored in a relational format. The same data is also stored in files to be used during simulations.

We have designed the schema of our databases to accommodate the volume, variety and velocity of the input data so that it can be processed in the forecasting simulations. In addition to the input data, it is also important to devise techniques for optimal representation of intermediate data and output forecast data so that it can be presented to the users. Our goal is to provide the users with the flexibility to evaluate the forecast results for a range of diseases, models, regions, surveillance sources and interventions.

To achieve this, the schema that we have developed produces a unique forecast output for each

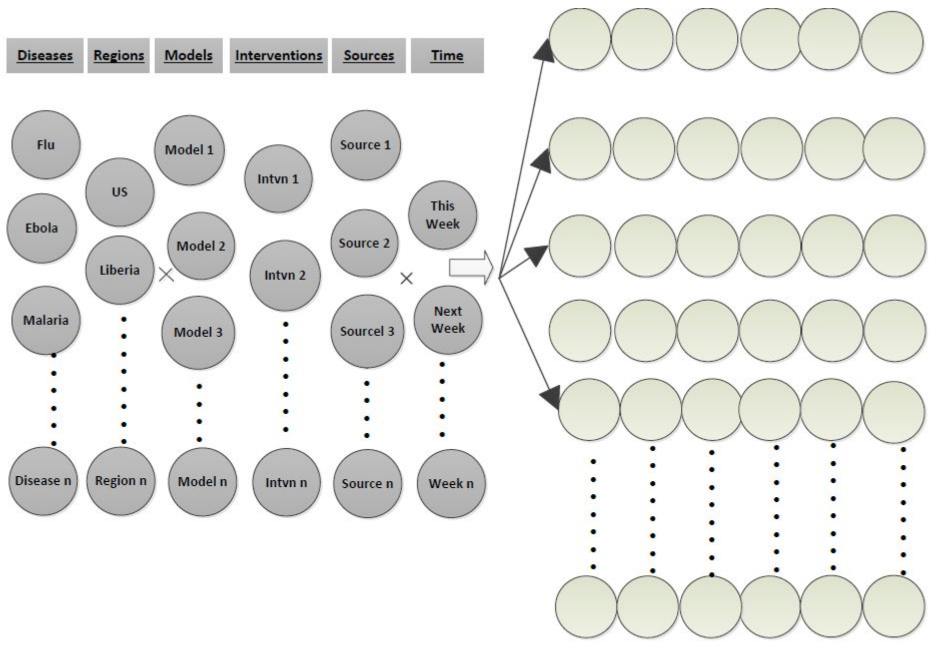


Figure 4.1: Epidemic forecasting output generated by the integrated analytics platform, represented as a Cartesian product of input parameters such as forecasting model, disease, region etc., showing the scale of data processed by the system

combination of diseases, models, regions, surveillance sources and interventions, which represent the dimensions of the forecast output. This ensures fast retrieval of forecast data so that it can be rendered using heat maps and plots.

The interesting aspect that summarizes the scale of data that is processed as output by our analytics platform can be explained as follows. If D is the set of diseases in the system, M is the set of epidemiological forecasting models, S is the set of surveillance sources , R is the set of the regions of the world for which epidemic forecast is sought, T is the forecast output period, and I is the set of interventions relevant to a disease, then the derived forecast output is a Cartesian product of D, M, R and I given by $Forecast\_Output$ = D X M X S X R X T X I. Hence, the volume of data can grow quickly with the addition of new entries to the dimension sets, such as addition of new diseases, regions, models and so on.

Figure 4.1 shows the overall scale of data that needs to be supported by the platform when there are multiple diseases, regions, models, surveillance sources and interventions.

4.2 Data Architecture

Most of the epidemiological models described in the literature for epidemic evaluation and forecasting, study specific epidemic scenarios in specific regions of the world for the benefit of health experts and epidemiologists. For instance, there are models that can forecast the impact of Ebola in West Africa based on surveillance data gathered from the ministries of health in the West Africa region. Modelers also use many different sources of surveillance data such as social media data, data gathered from the health ministries, hospital data etc. to compare the forecasting results based on various surveillance sources.

Due to the complex nature of the epidemiological models, the model output is a function of the parameters such as the surveillance source, region demographics, disease type, application of intervention strategies and so on. Hence, the computational systems that implement these epidemiological models are closely integrated with the data.

Our objective is to build a common epidemiological platform for evaluation and forecasting of a range of diseases so that they can be studied together. The platform should allow reuse and re-application of the models along with the ability to incorporate different model parameters.

To achieve this goal, we have developed a decoupled architecture where data is separated from the processing. Figure 4.2 shows the high-level data architecture of the analytics platform. All the data related to epidemics is collected in a single data store, which forms a common repository of information. Epidemic surveillance data is collected from multiple surveillance sources, such as, the CDC, hospital data, Ministries of Health, etc. and ingested into the central data store. The data store also ingests demographic data about populations in different regions of the world and the social-contact network between people that forms the fabric over which an epidemic propagates. The network structure may be different based on different diseases. For instance, the contact network for Flu is based on the physical proximity of people in a region, whereas for a vector-borne disease like malaria, the structure of the contact network is based on the presence of the vectors i.e. mosquitoes that affect the propagation of malaria in the region.

The epidemic simulations that implement the forecasting epidemiological models read this surveillance and population data as input from the data store and write back the output of the situation assessment and forecasting into the data store. Since the data is stored in a common place, any data visualization application can access this data from the central data store through use of appropriate object mapping techniques and display it through maps or other graphs on Web-based systems.

There are three main considerations with respect to the data stored in the platform.

- Data Model and Schema : This relates to the type of data-sets involved in the epidemic evaluation and forecasting process.
- Data Storage : This involves evaluation of the format of data storage, given requirements of latency, efficiency and throughput.
- Data Retrieval : This involves evaluation of the data formats used for data storage so that a range of flexible queries are supported for retrieving situation assessment and forecast data.

We describe these in detail in the next few sections.

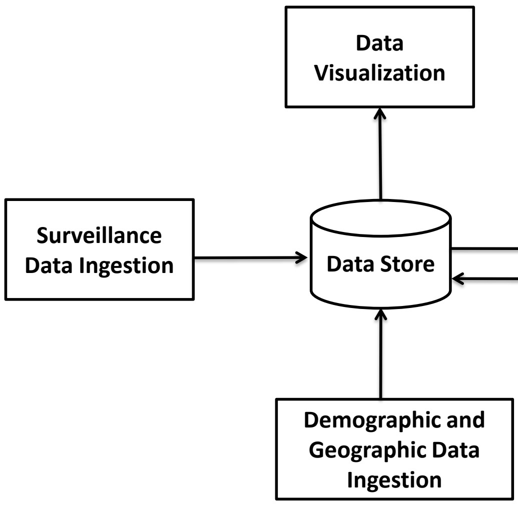

Data Visualization
Surveillance Data Ingestion
Data Store
Simulation Data Processing
Demographic and Geographic Data Ingestion

Figure 4.2: Decoupled data architecture of analytics platform, where the central data store is the common repository for all epidemic related information.

4.3 Data Model and Schema

The epidemic evaluation and forecasting models use a variety of data sets as input and produce forecasting data as output during the processing phase. In this section, we describe the various types of input and intermediate data-sets involved in the epidemic forecasting process along with the data schema.

The central data store collects and stores the following types of data sets: population demographic data, contact network data, surveillance data, model data and forecast data.

For individual-based epidemic models, the population data is represented as individuals or agents with demographic and geographic attributes. Most of the individual-based epidemiological models such as EpiFast [19] and Indemics [18] use synthetic information systems to represent individuals of a region. The population data of the synthetic information system is generated based on Census data [11], and hence can be assumed to be more or less static for the duration of the simulation.

The individual-level data is given by the tuple R = (person id, age, gender, home location id). In addition to individual-level information, epidemiological systems need i) social contact network information describing which individual is connected to which other individual and what is the type of contact, and ii) geographical data related to regions to find aggregate infection counts at different geographical levels.

The social contact network information is a graph $G(V, E)$ of a region, where V represents an individual of the region and E represents social proximity. The proximity may be defined differently based on different disease types. For instance, for studying a disease like Flu, the edges of the graph G represent physical contacts, whereas for a vector-borne disease like malaria, the contact network edges are based on the presence of vectors like mosquitoes in close proximity to be able to transmit the disease. Representation of these social contact network graphs G is an important consideration for executing epidemiological forecasting simulations. The details of the formal representation are available in the research on Indemics [18] [62].

Formal representation: An epidemic simulation system for situation assessment and forecasting can be represented as an extended Co-evolving Graphical Discrete Dynamical System (CGDDS). An extended CGDDS is represented by symbol $\mathcal{S}$ over a given domain $\mathbb{D}$ of state values and a domain $\mathbb{L}$ of label values, and is a triple $(G, \mathcal{F}, W)$, whose components are as follows.

The social contact network representing each region for which an epidemic needs to be computed is given by Graph $G(V, E)$ where vertex set $V = \{v_1, v_2, \dots v_n\}$ represents the set of agents (individuals). For each vertex v_i, let vector s_i denote its states $s_i = (s_i^1, s_i^2, \dots s_i^k) \in \mathbb{D} = (\mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_k)$, where s_i^j corresponds to state j. The states comprise of the agent's health state (Susceptible, Infectious, Recovered etc.), behavioral state (e.g. level of fear, risk aversion, etc.) as well as static demographic attributes (e.g. age, gender, household income etc.)

The edge set $E = \{e_1, e_2, \dots, e_m\} \subseteq (V \times V)$ represents the contacts between agents. For any edge $e \in E$, let vector ℓ_e denote its labels $\ell_e = (\ell_e^1, \ell_e^2, \dots, \ell_e^h) \in \mathbb{L} = (\mathbb{L}_1 \times \mathbb{L}_2 \times \cdots \times \mathbb{L}_h)$. In our social contact network, the edge labels include the contact duration and the contact type (home, school, work, shopping, or others).

Functions $\mathcal{F} = (f, g^V, g^E)$, where f is a set of local-transition functions; g^V is a set of vertex modification functions; and g^E is a set of edge modification functions.

For each vertex v_i, $f_i : \mathbb{D} \times \mathbb{D}^{V_i} \times \mathbb{L}^{E_i} \mapsto \mathbb{D}$ represents its local state transition function, where V_i and E_i are the neighboring vertices and edges of v_i. Normally V_i are vertices adjacent to v_i and E_i are edges incident on v_i. The function f_i corresponds to the epidemic propagation process which changes an agent's states based on the current states of the agent as well as that of all its neighboring agents and current labels on the contact edges with its neighboring agents.

$g^V = \{g_1^V, g_2^V, \dots g_{k_V}^V\}$ represents a set of vertex modification functions, where each $g_j^V : \mathbb{D}^V \times \mathbb{L}^E \mapsto \mathbb{D}^V$ directly changes states of vertices based on the current state of the whole graph. We assume V is constant.

$g^E = \{g_1^E, g_2^E, \dots g_{k_E}^E\}$ represents a set of edge modification functions, where each $g_j^E : \mathbb{D}^V \times \mathbb{L}^E \mapsto \mathbb{L}^{V \times V}$ changes the set of edges and the edge labels based on the current state of the whole graph. Note that g^E functions may add new edges to G; and for vertex-pair u, v with no contact, the labels on them are null.

The change in state of the vertices, representing individuals of a population, is represented by the schedule string W over the alphabet $g^V \cup g^E$. The schedule of modifications on graph G is given by $W = w_1^1 w_1^2 \ldots w_1^{j_1} \ldots w_t^1 w_t^2 \ldots w_t^{j_t} \ldots w_T^1 w_T^2 \ldots w_T^{j_T}$, where T represents the number of time steps. The t^{th} substring of W, $w_t^1 w_t^2 \ldots w_t^{j_t}$, denotes the updates on vertices and edges at time step t.

In an extended CGDDS representing epidemic dynamics in a social contact network, G is the contact network, f functions correspond to between-host disease progression, g^V functions correspond to within-host disease progression, pharmaceutical interventions (**PI**'s, e.g. antiviral, vaccination), and behavioral adaptations that directly change people's states (e.g., increase of fear level as the epidemic takes off, use of face masks), and g^E functions correspond to non-pharmaceutical interventions (**NPI**'s, e.g., school closure, quarantine and social distancing) that change the graph structure.

We have currently created such synthetic networks for different regions of the world including USA, India, Israel etc. For instance, the synthetic contact network of the USA is comprised of approximately 300 million nodes and 15 billion edges. The synthetic network generation process has been validated over the last 10 years through various quality control methods. Refer to [22], [42] for how such a network can be constructed rigorously.

We store the graph representation of different regions of the world in flat files on the high-performance computing clusters and read it in memory while executing the forecast simulations. The social contact network data N is also stored as a tuple $N = (pid\_1, pid\_2)$, where pid_1 and pid_2 represent the end points of an edge in the social contact network. It is stored in the RDBMS so that interventions based on the social contact network structure can be formulated.

Geographical information is stored at a fine-grained level for every nation of the world. The population information is given by the tuple $P = (pid, state, county, blockgroup)$. This is stored for every individual of the region. The geographical location information for each region is given by the tuple L= (region_id, level1_area_id, level2_area_id, level3_area_id), where region_id indicates the unique identifier of the region, and each subsequent level indicates a finer gradation of the geographical region. Typically, each nation is represented as a separate geographical region so that various geographical breakdowns per country are represented in a uniform way. For example, for USA, level1_area_id represents the state, level2_area_id represents the county, and level3_area_id the block group identifier; for Liberia, there is only a level1_area_id that represents the county.

The last set of input data needed for epidemiological models and the most important one is the up-to-date surveillance data about infected individuals in the affected regions. This dynamically evolving epidemic data from surveillance sources is represented as time-series data of infected individual counts, and is represented by tuple E = (date,region,count). E is updated periodically based on availability of surveillance. For instance data from CDC and GFT for flu is available on a weekly basis, whereas data for Ebola is collected from sources such as WHO, ministries of Health (MoH) and so on whenever it is published by those sources.

The output of the forecasting simulations that implement the forecasting models is given by the situation assessment and epidemic forecasts for different regions for which specific epidemic information is available. Based on the type of epidemiological models used, the output is different. For instance, if individual-based model like EpiFast [19] is used, then the forecast is represented as individual-level infections and is given by the tuple F = (person id, replicate, infection date). The forecasting simulations are executed when the epidemic surveillance data is available i.e. when data-set E is updated. Note that the epidemic simulation is a stochastic computation, and usually runs for multiple replicates to account for the variability in the simulations. Therefore the output data set F of infected individuals, computed for each region (typically at national level) for a given disease, stores *replicate* information. This forecast information is indexed by both the infection date and the identifiers of the infected individuals. This indexing allows fast and flexible retrieval of data for studying the forecasts at fine-grained individual levels. The forecast data across multiple replicates is averaged to arrive at the aggregate forecast counts at different area levels by using individual forecast information together with location information. This aggregate-level data set from individual-based epidemic models is given by A = (infection date, infection count, region_id, level1_area_id, level2_area_id, level3_area_id). The output from the aggregate-based epidemiological models can also be described by the data set A, since it directly produces aggregate infection information for each region.

4.4 Data Storage

The main considerations for data storage of epidemic situation assessment and forecast data include efficiency with respect to space and time for data access and manipulation. The volume of data that we deal with in the epidemic forecasting domain makes data storage and retrieval an interesting problem. As part of the data storage requirements, we also need to consider whether data needs to be replicated across different machines for ease of reliable processing and availability.

As described in Section 4.3, for executing individual-based epidemic forecasting simulations, we store the population of each region by representing each individual of the population with demographic and geographic attributes. For instance, the population of the USA is around 300 million, the population of Liberia is around 4.2 million, and that of India is around 1.25 billion. Within the USA, the population needs to be stored and queried at multiple levels, including state, county and block group. For instance, the state population sizes range from 0.563 million (Wyoming) to 33.6 million (California).

The population information is critical for running forecast simulations. Hence this data is replicated across 2 places - (i) The data is stored as flat files on the high performance computing clusters that simulate epidemic propagation, and (ii) The data is stored in relational format for querying purposes.

Other than the population tables, we store surveillance data for each disease in relational format so that it can be easily accessed. The surveillance data is also stored as flat files and read in memory while executing forecast simulations.

We chose relational databases for storing the demographic and population data because the data model fits well with the relational format, where information needs to be stored for each person of a region, and their locations. The relational database format is also used for storage of forecasts, since it provides the flexibility to query the system in multiple ways, without any restrictions on the format and structure of the data to be fetched. The synthetic contact network data is stored as flat files and is fetched during the simulation execution phase of the forecasting pipeline. This allows the contact network to be different for different diseases based on whether it is air-borne or vector-borne, and separate from the other population demographics, which remain the same for every type of forecast.

The main drawback of NoSQL data stores like Cassandra, neo4j etc. with respect to our requirements is that most of the NoSQL data stores do not provide the flexibility of query mechanism. The schema of such databases has to be designed during the database design phase itself with the knowledge of typical queries to be executed later, thus limiting the flexibility of query execution at run-time. For instance, the design of the neo4j database is optimized for storing graph queries. For any other types of queries, the database might have significant performance issues compared to relational databases. Similarly, data stores such as Cassandra can optimize storage of time-series data pretty well, but the schema design is not flexible and does not support join operations with other data sets. This limits the type of queries that can be executed using such databases. Hence, the relational database model and structure was chosen as the format for data storage and retrieval for storing the data related to situation assessment and forecasting in different regions.

Scale of Forecasting Data

We store the forecast outputs of each region in relational format in separate tables, based on source of surveillance, forecasting model and type of intervention. The forecasts are generated periodically for each disease type for each region, after receiving input surveillance data from different data sources. Note that, if a model uses multiple sources in its forecasting computation, then the combination surveillance source is stored as a separate entry in the database table. Forecasting output derived from individual-based models, the size of the tables can contain up to several thousand rows, depending on infection rates and number of replicates used for stochastic computations. This use of stochastic computations increases the scale of storage several folds to improve the accuracy of the forecast. For instance, based on GFT data, the forecast simulations produce individual outputs for each state of the US. The typical size of these tables for WV, VA and CA are 2 MB, 46MB and 240 MB respectively, generated every week. The total size of data for US is around 3-5 GB every week. Similarly, several MBs of data is generated for other nations of the world based on availability of surveillance. This data needs to be purged periodically after completion of analysis and aggregation, so that it does not keep growing exponentially.

In our platform, we also create aggregate tables, to store aggregate counts of infections at different geographic levels for each region. Some of these tables are large, but not excessively so; for example, each summary table for the US is about 582 MB. However, when one considers that there could be more than four interventions, at least three or four data sources, and several models, the system has to generate and support at least 18GB of forecast output data per country per week.

This is in addition to the raw surveillance data gathered from several sources that is only a few GBs in size, and the synthetic demographic and contact data stored for each region, with storage sizes ranging from a few MBs for smaller regions to several GBs for bigger regions.

In short, the volume and velocity of information gathering leads to generation and storage of several gigabytes of epidemic surveillance and forecast data every week, and the scale can grow into several terabytes as more replicated data is stored over a period of time.

To summarize (the scale of data), the surveillance data is obtained periodically (typically on a weekly basis) and the forecast needs to be computed as soon as the surveillance is available. With the pre-computations, aggregations and transformations from different formats, the scale of data produced is very large. For instance, in the current implementation of our platform, we compute flu forecasts for the US and India, and Ebola forecasts for Liberia and Sierra Leone. The scale of demographic and contact network data for these regions is in the range of around 250 GB. We obtain surveillance data every week for flu and Ebola in different formats. Considering intermediate data involved in transformations and manipulations, the scale of this surveillance data is around 10 GB per week. We have ABM-based and ODE-based simulations that produce forecast outcomes. This output forecast data is in the range of 15-20 GB. Overall, the current platform implementation stores around 500 GB of data, with around 20-30 GB of data getting added every week for flu and Ebola. With addition, of new regions, models and diseases, there will be a corresponding increase in the scale of this data.

4.5 Data Retrieval and Analytics

The main objective of building an integrated platform for evaluation and forecasting of epidemics is to be able to retrieve large scale analytical data in a flexible way. In our relational model, we have extended the concepts from data warehousing and relational databases to efficiently retrieve flexible analytical queries at transactional speeds.

The queries that need to be supported for fine-grained situation assessment and forecasting within EpiCaster are mostly analytical in nature. The typical type of data that needs to be retrieved using queries includes finding aggregated counts at regional, national and world levels per disease, and ranking of infection counts by region per disease. For instance, sample queries might include “Find the overall infection counts of Ebola in the world this week”, “Find the top 3 regions in USA that are forecast to have highest counts of population affected by flu”, and “Find the flu infection counts in Montgomery County, Virginia in the next week”.

We have applied several mechanisms for improving the speed of query execution to retreive forecast results faster from a schema standpoint. Some of the optimizations that we have implemented to streamline the epidemic data collection process as well as the forecast data generation and retrieval include:

1. Precomputation: Our analytics platform processes epidemic forecasts on multiple geograph-

ical levels based on several inputs including type of model, surveillance source, disease, intervention types and region. The forecast simulation that produces the forecasts is a detailed simulation that may take several minutes to hours depending on the computation and parameters of execution. Hence, it is difficult to execute a simulation when the forecast data is requested. We use precomputations of several different combinations of execution parameters and store the results in a way that can be accessed when queried.

As a special case of precomputations, we perform summarization of data that is requested often for faster response times. For instance, we have created a summary table containing precomputed data for the retrieval of aggregate-level counts of infected individuals in different regions derived from individual-level tables generated from individual based models.

2. Lookup Registry: We have created a look-up registry for ease of access of precomputed forecast data from the database. The look-up registry that we have developed is an extension of the star schema from the data warehousing literature. The star schema that we have designed is based on different dimensions that relate to an epidemic forecast, such as region, disease, intervention, surveillance source and forecast model. We describe the star schema architecture that we have designed for the lookup registry in the following sections. The lookup table is an extension of the fact table in a star schema. However, instead of directly storing the facts like in a star schema, the lookup table stores a pointer to different tables that stores forecasts for a combination of dimension attributes. For example, the lookup registry stores a pointer to the USA_FLU_M3_S1_SUMMARY table for the row with attributes region = USA, disease =flu, surveillance_source= GFT and model= EpiFast and so on. We describe our design of star schema in the following sub sections.

3. Denormalization: We have denormalized the data in the database wherever possible for faster response times; with denormalization, response times have dropped from minutes to seconds. For instance, all the location data tables have been denormalized for faster data retrieval at multiple geographical levels.

4. Transformations: We apply transformation techniques to store data in a compatible format throughout the forecasting pipeline. For instance, the demography information related to individuals in a region is transformed into population tables that store the geographic levels in different regions of the world and their corresponding populations. This also speeds up the computation while computing aggregate infection counts at different geographical levels.

5. Automations: To avoid any manual interventions in the data collection and manipulation processes, we have automated the creation of forecasts and the generation of table summaries; these scripts are run weekly to populate the forecast tables and make data available to the end-users.

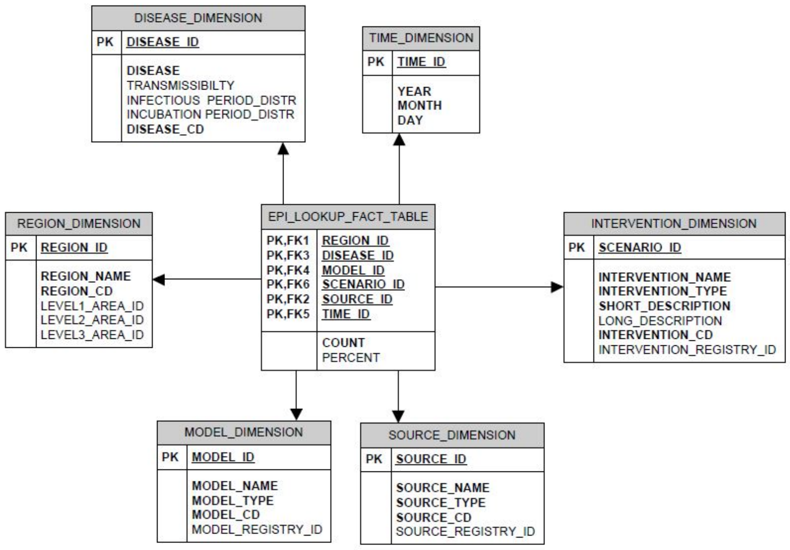


Figure 4.3: Traditional star schema model design for analyzing epidemic forecasting results across multiple dimensions where the fact table is represented by a central look-up registry of forecast counts

4.5.1 Star Schema Design of the Analytics Platform

The main motivation for designing our analytical platform is to be able to analyze a large range of analytical queries in a flexible way. We have extended concepts from big data processing and traditional data warehousing.

In particular, we have used the star schema design from the data warehousing literature for the purpose of storing forecasting data. The dimensions that relate to an epidemic forecast include region, disease, intervention, surveillance source, forecast model and time. Time is an important dimension for analysis of evaluation and particularly forecast data. However, there are multiple ways to represent the time dimension, which changes the flexibility of querying. Time dimension is dependent on the frequency of executing forecast simulations, which in turn is affected by the frequency of collecting surveillance data from different surveillance sources.

We evaluated three different ways of representing the forecast data in a star schema format. Each format has its advantages and disadvantages and we discuss each of them below.

Standard Star Schema Design

The first approach is the classical star schema design used in traditional data warehousing approach, as shown in Figure 4.3. In this approach, time is considered as one of the dimensions along with disease, region, model, source and intervention. The time dimension is represented by day, month and year. All the aggregated forecast output data related to all the diseases in different regions of the world, is stored in a single fact table that stores the forecasts as “facts”. Since the fact table stores all the forecast data, the scale of data in this table is very large.

The advantages of using this approach are as follows:

- The data related to all the epidemics is stored in a denormalized form in a single table. Hence a single table needs to be loaded while fetching data related to any disease.
- Since the analytical data is denormalized, this approach is efficient when the types of queries from forecast outcome are known in advance. In this way, the fact table structure can be optimally designed for fast query retireval.

There are also several disadvantages of using this approach as follows:

- Inflexibility: Since the data related to all the epidemics is stored in a single table, the structure of the table needs to be set. If a new type of parameter needs to be added, then the structure of the table might need to be changed and the forecast output data for all regions and diseases using all models has to be reloaded again.
- It is imperative to load the data related to all the diseases, regions, models, sources and interventions into the main memory, during query processing, irrespective of the type of query, which is memory inefficient.
- This approach is not able to handle the differences in the number of geographical levels for different regions of the world. For instance, USA has three main non-overlapping contained geographical levels, namely state, county and block group. Liberia, on the other hand, has only one geographical level given by counties. However, using the traditional star schema design approach, it is imperative to represent all the regions with the maximum number of levels possible, i.e. in this case three levels(level1_area_id, level2_area_id and level3_area_id) as shown in the Region_Dimension table in the Figure 4.3. This increases the scale of data to be stored and retrieved from the system.

Modified Star Schema Design for Flexibility

The second approach to represent data for the analytical platform involves a modified star schema design as shown in Figure 4.4. In this approach, time is considered as a separate dimension, similar to the traditional star schema design. The main difference in this approach compared to the traditional star schema approach is that instead of storing the data related to epidemic forecasting output in a central fact table, the fact table is only used as a lookup table to point to other tables that store the forecast outcome data. There is a table storing aggregate data for each combination of region (national level), disease, surveillance source, forecasting model, intervention and time. The time dimension is used to represent the time at which the data is gathered from the surveillance source and when the forecast is generated using simulations. For instance, Google Flu Trends is used as the surveillance source for forecasting of flu in the USA. GFT data is gathered on a

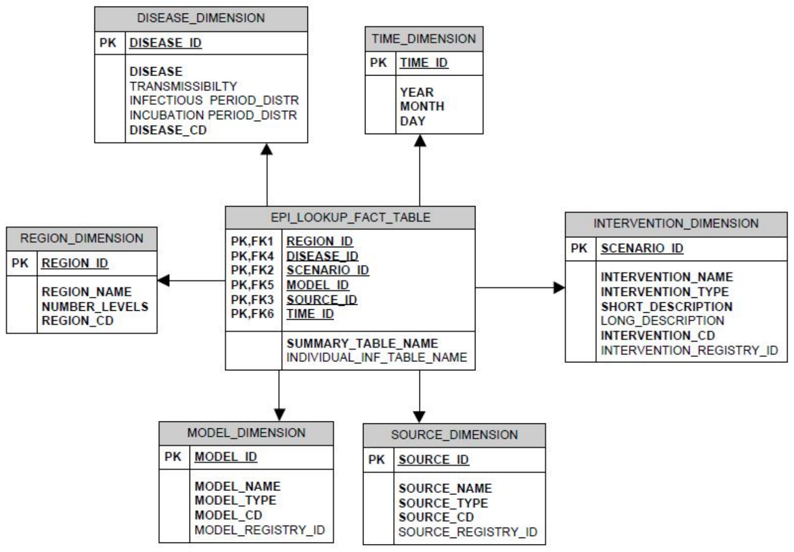


Figure 4.4: Modified star schema design for adding flexibility and extendibility in analysis of epidemic forecast results across multiple dimensions, by having the central fact table point to a table storing forecast results

weekly basis. The forecast pipeline generates the forecast output as soon as GFT surveillance data is available every week. This forecast output data is stored in a new output table every week and referenced by the central look-up fact table. Similarly, a new aggregate data table is generated every time period for every combination of region, disease, surveillance source, forecasting model and intervention.

The advantages of this approach are as follows:

- Flexibility: The data related to each combination of region (national level), disease, surveillance source, forecasting model and intervention is stored in a separate table. So when query results are to be retrieved for a particular disease or region, only a subset of the data needs to be loaded in the main memory, which is more efficient than loading the entire data into the main memory.

- If a new parameter related to a particular disease in a particular region needs to be added, then the change is localized, and only the relevant table need to be modified, instead of modifying or changing data related to all the diseases.

- Since the features related to a particular region can be localized to specific tables, this approach can effectively handle the differences in the number of geographical levels for different countries of the world.

- This approach can store retrospective forecasts derived at different time-stamps in the past

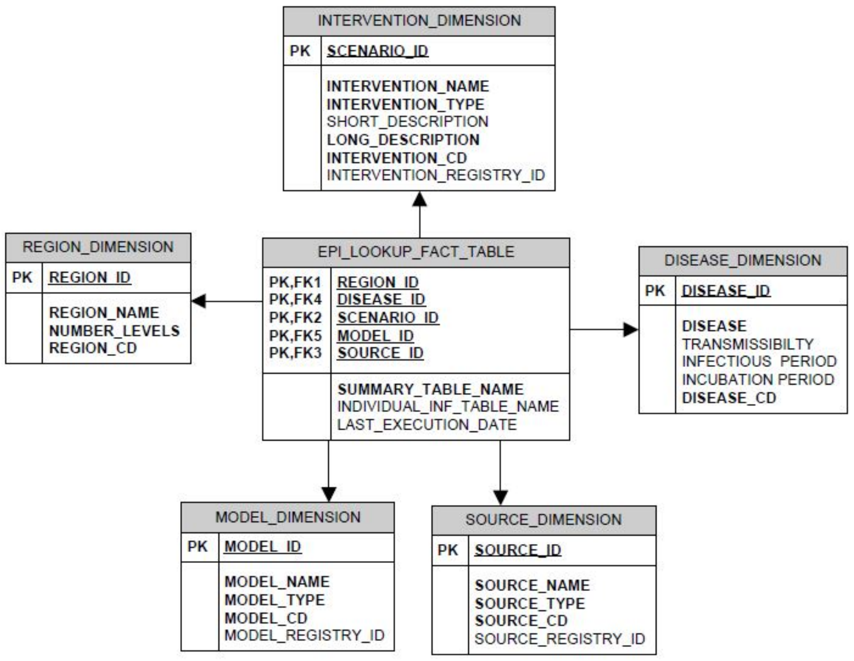


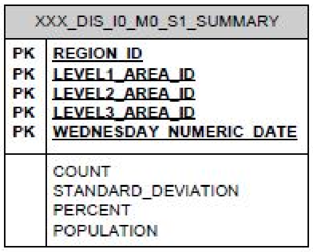
Figure 4.5: Modified star schema model design with flexibility and storage optimization by storing forecast results across different time periods together in a single table

with reference to the corresponding surveillance data. Such retrospective data may be required for in-depth analysis of how the forecast accuracy has evolved over a period of time or to analyze how the quality of forecasts may affect the forecast outcome.

The disadvantages of this approach are as follows:

- This approach produces several small tables at regular intervals based on the rate of gathering of the surveillance data (typically weekly). Creation of many such small tables may lead to database fragmentation issues.
- This approach stores forecast data in separate tables, based on the rate of gathering of surveillance data. So the forecast outcome generated for the current time period using surveillance data from previous time period is stored in a separate table from the one storing the forecast output from the previous time period. When the surveillance data is obtained for the current time period, then the forecast data for the current time period becomes irrelevant. However, in this approach all this additional data is stored in separate tables, increasing the overall scale of data. However, such data may be needed for analysis of retrospective forecasts as described above.

Modified Star Schema Design for Flexibility and Storage Optimization



XXX_DIS_I0_M0_S1_SUMMARY	
PK	REGION_ID
PK	LEVEL1_AREA_ID
PK	LEVEL2_AREA_ID
PK	LEVEL3_AREA_ID
PK	WEDNESDAY_NUMERIC_DATE
	COUNT
	STANDARD_DEVIATION
	PERCENT
	POPULATION

Figure 4.6: Structure of the summary table of infections referenced by the central fact table in the modified star schema for looking up infection forecast results for a given region

The last and final approach to represent data for the analytical platform extends the modified star schema design and takes care of some of its inefficiencies with respect to the scale of data and storage. We have implemented this design in our analytical platform. The star schema design of this approach is shown in Figure 4.5.

In this approach, time is not considered as a separate dimension in the design of the star schema. The dimensions of this star schema include region (national level), disease, surveillance source, forecasting model and intervention. The approach is very similar to the modified star schema design for flexibility described in the previous sub section, where the fact table is only used as a lookup table to point to other tables that store the data. There is a table storing aggregate data for each combination of region (national level), disease, surveillance source, forecasting model and intervention. Instead of having separate tables for the time dimension based on the time of gathering the surveillance data, we append the new forecasting output data into the same table based on a combination of other 5 dimensions. For example, when new data on Ebola epidemic is gathered from the World Health Organization (WHO), a simulation is executed based on the ABM-approach without any interventions. The simulation output is generated, and the aggregated summary data is uploaded in the table "LBR_EBL_I0_M1_S2_SUMMARY" , storing region_id=430 (for Liberia), disease_id=2 (for Ebola), scenario_id=0 (for no intervention scenario), model_id=1 (for EpiFast forecast), and source_id=2 (for WHO). The structure of this table is of the format given by Figure 4.6. When new surveillance data is gathered, the forecast outcome from the same time period is over written with the surveillance data.

This approach has almost all the advantages of the modified star schema approach mentioned above such as memory efficiency, flexibility and localized changes. It also has some additional advantages as below:

- This approach stores all the forecast information related to the five dimensions in a single forecast table. This prevents database fragmentation due to creation of several small tables.
- Since all the data across time is stored in a single table, data can be retrieved across a larger time frame by accessing a single table, thus making the retrieval process more efficient.

- This approach can be extended for implementation of newer models of infection forecasting such as individual-level infection forecasting. This can be done by having a pointer to the individual-based infection tables. In the traditional approach, this is not possible without altering the structure and schema of the fact table. Also, using this approach, new forecasting measures such as “Transmission Rate”, “Recovery Rate” etc. can be studied, that may be produced by only certain forecasting models, without having to change the storage formats of all forecast outputs.

There are also some disadvantages of this approach as follows, that stem from the storage optimizations that it achieves:

- This approach stores surveillance and forecast outcome data across time periods in a single table and overwrites the forecast data produced from the previous time periods when new surveillance data is gathered. Sometimes, this forecast data generated from past time periods may be needed for retrospective analysis, for comparing how the output of the forecasting model has evolved. Since this approach overwrites past data, retrospective analysis is not possible.
- Using this approach, the volume of data that needs to be loaded into the main memory when the database is queried, is larger than that in the second approach. This is not a significant problem over a short duration of time, but over a large time-period ranging over several epidemic seasons, the total volume of data stored in each table may be very large. It may be inefficient to load the entire data across time periods in the main memory.

We experimented with all the 3 approaches above for storing analytical forecasting data on flu and Ebola in our system. Currently, we have used the last approach described above in our platform implementation because of all the advantages that it offers.

4.5.2 Scope of Data and Underlying Assumptions

We have described the data architecture and schema for supporting a range of diseases, regions, models and surveillance sources in our platform. The current implementation of the platform has been applied to the forecasting approach using EpiFast, which is an Agent Based Model implementation for flu and Ebola. We have also demonstrated our platform for use in forecasting that has been derived using Aggregate based models for flu. The underlying model for epidemic propagation was based on the SEIR model.

Forecasting models that use other epidemic propagation models such as SIR model or SIS model can also be incorporated into the system. Based on the type of forecasting model used, the forecast outcomes will differ. Hence different data transformation techniques will have to be applied to transform the output to a common format. The advantage of our Modified Star Schema design is that the forecast model output is stored in a table separate from the Fact table of star schema. Hence any new infection characteristics other than only "Infected counts" such as "Recovered" counts or "Transmission" counts can also be included for analysis.

The underlying assumption for using our platform is that the forecast outcomes derived using any new forecasting model should adhere to the following guidelines, with or without data transformations:

- The forecast output for any epidemic can be produced at any geographical level by the forecasting algorithm. However, while storing the data, we store the aggregated counts at a country level. For instance, the forecast output of USA is produced by aggregating the forecast outcomes of its 50 states, for which the forecasts are produced separately. While evaluating the forecast outcome at any level, the assumption is that if the forecast data for the next level is available for even one sub-region, then data for all sub-regions within it is available and hence the infection count is a summation of all infection counts across all sub-regions.
- Our approach assumes regional hierarchies within a national level. Hence for evaluation of forecasts by geographies, there is an assumption of disjoint regions at every level. Hence, the system cannot handle non-disjoint overlapping set of regions. If a forecasting model produces forecasts for such non-disjoint sets, then the model has to transform the data to produce non-overlapping disjoint sets.
- Flexibility to analyze the forecast outcome is limited by the availability of demographic and geographic information for each region for which forecast output is computed. Since we use relational databases for data representation, the flexibility is also dependent on the limitations imposed by the relational algebra on the data schema.

4.6 Summary of Technical Contributions

To summarize, following are our technical contributions with respect to the data integration methodologies:

1. Design and implementation of decoupled data architecture for providing flexibility and extendibility with respect to multiple diseases, geographical regions, surveillance sources, forecasting models and mitigation strategies.
2. Design and extension of star schema for large scale data analytics related to epidemic forecasting where the central fact table is used as a lookup table, instead of storing facts.
3. Creation of flexible data storage and retrieval mechanism for optimal query execution in epidemic forecasting models. Our data representation approach simplifies the process of adding a new region, disease, model or surveillance source into the system. The total time

for adding new data sets for new diseases, regions etc. is reduced from several days of development effort to a few hours of data manipulation and population in appropriate database tables.

4. Support for flexible queries related to epidemic computations across several different regions of the world at multiple geographical levels for evaluating and forecasting of a range of epidemics with and without interventions.

Chapter 5

Architecture and Systems Aspects in Analysis and Forecasting of Epidemics

Epidemiological modeling systems involve large scale data related to population demographics, surveillance, and model parameters, as explained in Chapter 4. Moreover, given the stochastic nature of the computations, the experiment has to be executed over several replicates, further increasing the overall volume of data. An ABM-approach represents populations and propagation dynamics at a fine-grained individual level. Hence, the scale of data is large in the ABM approach as compared to ODE-based models.

In addition to data volume, the scale of the computations involved in epidemiological systems is large, given that the computations need to be executed in a reasonable amount of time so that the public health decision makers can set up experiments, analyze the results of the simulations, and implement concrete measures for epidemic containment.

Traditional epidemiological systems have used high performance computing resources to execute simulations. In the past, grid and cluster computing were the preferred ways of executing large scale epidemic simulations. Recently, cloud computing and other citizen science projects like BOINC have also gained popularity for executing large scale scientific computations.

The middleware for supporting communication and data transfer in an analytics platform should be able to handle the scale and complexity of high-performance computational systems. In addition, the middleware has to be adaptable to the changing models and methodologies used in the computations. A tightly coupled system, where the middleware and data are intertwined will require major reengineering if there are any changes or updates in the data or the models. Hence, a loosely coupled service-oriented abstraction with modular architecture is better suited for the design of such an environment.

Use of service-oriented architecture in high performance computing systems is an emerging field. In particular, there has been some research in the field of distributed systems to expose high-performance computing resources as services that can be consumed by other systems [8]. This new area of innovation is called HPCaaS (HPC as a Service), which derives from the positives of both cloud computing and high-end cluster computing.

In computational epidemiology, the main goal of a middleware platform is not just to develop higher level services that are static and publish them for consumption, but also to provide services that are dynamic and interactive. The middleware should be able to support large scale analytics and computations that can present dynamic results at transactional speeds. The necessary speed and scale of the computations make the design of the middleware and back-end platforms an interesting problem.

We have built our analytics platform for the following main purposes:

- Evaluation of dynamic intervention strategies for epidemic containment in near real-time, and supporting accessibility of the system through Web interfaces.
- Forecasting of epidemics of different contagious diseases in different regions of the world, computed using diverse forecasting models, based on a range of surveillance data.

From a systems architecture standpoint, the requirements of both are vastly different. In the first goal, we need to have a system that can support interactive computations and the ability to pause and analyze partial simulation results through Web interfaces. The second goal is dependent on the availability of surveillance information from the environment, and, hence, the computations are tied to the availability of data.

We describe our main approaches for handling these vastly different requirements - 1. of interactive computations via the Web, and 2. of asynchronous communication with multiple data types for forecasting computations. Both of these approaches need flexibility to query the system in multiple ways for evaluation. The two approaches that we have developed to support these requirements are both based on service-oriented architectural paradigm, but the scale of computations and the speed of data passed back and forth through the system differ vastly.

5.1 Middleware Platform for Interactive Computations in Intervention Evaluation

Achieving interactivity through a middleware platform is a challenging problem, since the platform needs to store the current “state” of the system to be able to resume the ongoing computations. Achieving interactivity through a Web-based platform is more challenging, since the parameters of computation have to be passed efficiently from the web browser to the web server and further to the high performance computational resources, where the epidemic simulations execute.

We have developed and extended two main programming abstractions - Simfrastructure and Indemics Middleware Platform - to achieve our goals. Interactivity is achieved through the Indemics

Middleware Platform, whereas asynchronous communication through message passing to start, stop, pause and resume a simulation are achieved through the use of Simfrastructure. We briefly describe the details of these two middleware platforms in the following sections and describe how we have extended both together in the middleware architecture to achieve our goals.

5.1.1 Simfrastructure

Bisset et al. have developed Simfrastructure [21], a flexible coordination middleware that supports high performance computing oriented decision and analytics environments to study socially coupled systems. Simfrastructure provides a multiplexing mechanism by which user interfaces can be plugged in as front-end systems, and high-end computing resources, including clusters, clouds, or grids, can be plugged in as back-end systems for execution. Simfrastructure is based on the concept of Service brokers that can handle the request for a particular service and return back the results to the blackboard. It uses a distributed, coordinated blackboard mechanism for message passing between various system components

Related Research

There have been many recent advancements in the field of high performance computing systems and middleware platforms to support coordination between the components of such systems. However, to the best of our knowledge, our work on Simfrastructure is the first effort of its kind in developing a middleware platform that aims at providing seamless access to powerful computational models and resources for use by subject matter experts. In this section, we list some of the recent research relevant to our work.

Parashar et al. [8] have argued the need for delivering high performance computing resources as a service to scientists and domain specialists. The authors provide a prototype implementation of an elastic cloud that provides high performance computing infrastructure as a service using the IBM BlueGene Supercomputer. This new innovation, HPCaaS, derives from the positives of both cloud computing and high-end cluster computing. In our approach, we deliver high performance computing services as well, but as an indirect consequence. Our focus is primarily on delivering higher level services, in particular analytics and modeling for socially coupled systems. We also differ in the types of computational resources we address, including clusters, clouds and grids. On top of providing an easy coordination mechanism between user interfaces, data stores and computing resources, our space-based architecture is also used to provide resource allocation and management, data transfer, and digital library services.

Some of the other related work in this domain includes the work on Narada brokering architecture and Granules. The Narada brokering architecture by Pallickara et al. [76] provides a distributed brokering system with a middleware that combines a hybrid environment of peer-to-peer systems and grids to provide web services. “Granules” [75], a streaming-based runtime environment, extends the basic brokering architecture from “Narada” to execute complex scientific applications on the cloud using a peer-to-peer system of communication between its broker components. It also supports the existing MapReduce framework developed by Dean et al. [33] and variants of it.

Our approach greatly differs from these other applications in that our work does not focus on any particular programming model such as MapReduce to provide services. Instead, we provide a platform for large scale, socially coupled systems to provide and access services, without having any constraints on the models used, type of data required, or computing resources available. For instance, our approach does not require the simulations to follow certain norms of execution parallelism, whereas MapReduce operations typically need the datasets to be highly parallel in nature. Hence our approach can be used in cases where large scale sequential execution has to be supported, along with cases where massively parallel operations are to be executed on the grid. Moreover, our approach uses tuple-space based architecture, instead of earlier approaches used in the literature, such as peer-to-peer systems.

Space-based architectures have gained popularity and applicability in various domains ever since the work of Gelernter et al. [48]. The papers by Docan et al. [37], Atkinson et al. [9], Engelhardtsen et al. [40] and Batheja et al. [15] explore space-based architectures for building adaptive distributed systems. The novelty of our approach lies in bringing together a comprehensive set of services involving simulations, data stores, computing resources, and user interfaces required for automating socially coupled systems. Resource management, job monitoring, data transfer management, and digital library services are a few examples of the services we can provide. Our approach also makes it possible to provide a powerful analytical platform for domain-experts.

Space-based architectures provide a natural way for coordination in a spatially, temporally and logically decoupled manner. However, space-based architectures are known to have certain limitations, some of which are thoroughly studied and addressed in the literature [9, 17, 46].

Simfrastructure Architecture

Simfrastructure is a distributed middleware platform providing a set of well-defined services. The services are provided by processes called brokers that coordinate over a shared associative memory space called a blackboard to serve a given request. Figure 5.1 depicts a typical set up of Simfrastructure, outlining its key components which are described in greater detail in the following sections.

Blackboard

The blackboard is the central communication and coordination mechanism of Simfrastructure. It is a shared, persistent memory space, commonly called the tuple space, that holds the requests for services. Requests are typed objects with attributes where the type signifies the service requested, and attributes represent the parameters required for the service. The requests are posted to and retrieved from the blackboard to request and fulfill services respectively. They can be read, retrieved and posted to the blackboard using the synchronous or asynchronous variations of “read”, “take” and "write" operations respectively. Requests can be read and retrieved with the use of a template request, with desired type and attributes, and then matched against the existing requests on the blackboard. The blackboard operations are delivered through a transaction mechanism to preserve the integrity of the system.

Requests are the primary means by which brokers coordinate by exchanging information. Typical information contained in requests are:

- **Parameters:** The parameters required to deliver a particular service. A simulation service may require parameters such as population, simulation model, disease model, infection seeds and interventions etc.
- **State:** A request can be in any one of the following states during its life cycle: new, posted, running, deleted, successful and failed. The state information is used extensively by service providers and consumers to indicate success or failure of a service fulfillment.
- **Workflow:** A workflow contains the details about how a service is to be fulfilled. For instance, a simulation request may contain the workflow to pre-process the input data, locate and fetch the required datasets, run the simulation, and validate the produced output. The workflow is typically specified in the form of an embedded object called the runner. The runners are described in greater detail in Section 5.1.1.

The blackboard is currently implemented as a JavaSpace [92] holding the service requests as standard Java objects.

Brokers

Brokers are the components of Simfrastructure responsible for delivering a particular service. They monitor the blackboard for specific requests and deliver the appropriate services. Additionally, brokers post requests to the blackboard to consume any intermediate services required to fulfill a service. A broker essentially comprises of a set of conditions and actions. The conditions, when satisfied, trigger the actions to deliver a service. The conditions embody the matching criterion specified over the type and attributes of the requests. The actions, however, vary greatly in complexity and nature based on the service they extend. Some services can be delivered through a simple set of actions, while others may need an elaborate workflow to be employed. For instance, a logging service would require actions to simply read the requests on the blackboard and log appropriate attributes as and when they are posted or removed. However, a simulation service would require an elaborate workflow of actions to pre-process the input data, locate and fetch the required datasets, run the simulation on computing resources, monitor the simulation execution, and validate the output produced.

In Simfrastructure, workflows are encoded in separate objects called runners. The runners are embedded in requests and are executed by brokers to deliver a service as can be seen in the Figure 5.2. The runners perform various actions, including consuming services from other brokers, in order to deliver a service. Runners are typified by short compute portions followed by long idle times as

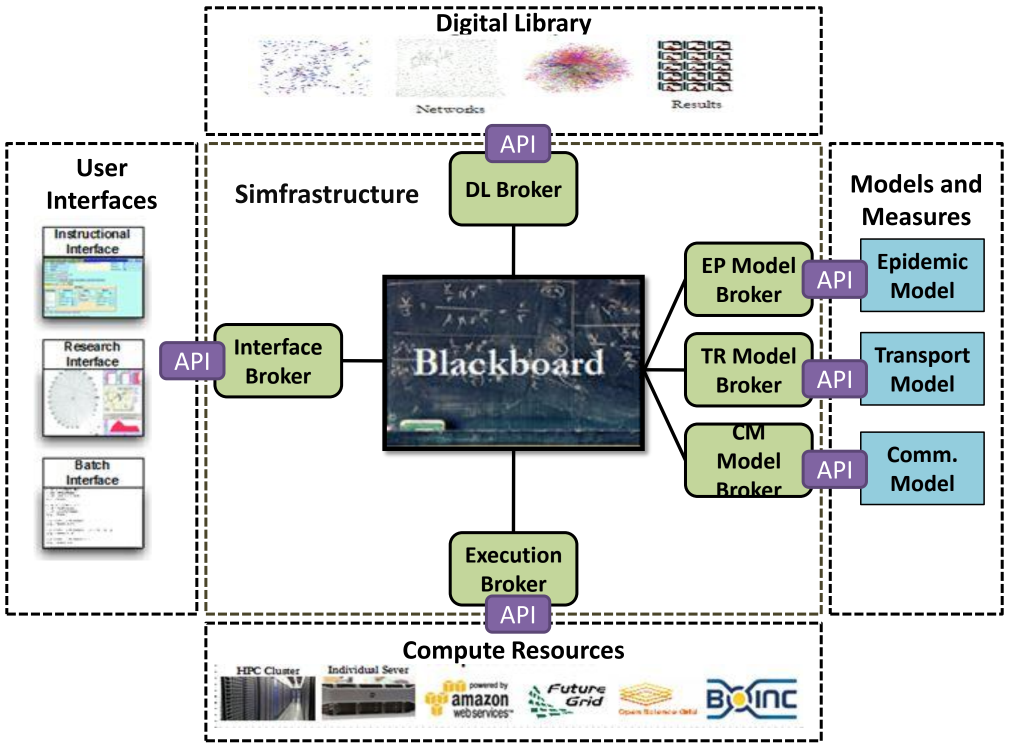


Figure 5.1: A typical setup of the modeling and simulation environment using Simfrastructure as the central communication and coordination mechanism. Simfrastructure coordinates data and context flow between user interfaces, the digital library, computational resources, and simulation models and measures. Reference: Simfrastructure, Bisset et al. [21]

requested services are fulfilled. For instance, the runner for executing a simulation model may use the request parameters to process the input and create a configuration file, determine the datasets needed, spawn new requests to find the datasets, and make them available on the local system if not already present. Furthermore, it may spawn a new request to execute the simulation on available computing resources, and, finally, validate the output before converting it into a common format as part of post-processing. In Simfrastructure, runners offer modularity that simplifies addition, update, and removal of workflows. Adding a new workflow requires developing a new runner and embedding it in the corresponding request.

Based on their actions, Simfrastructure brokers can be classified into Standalone and Workflow brokers. Standalone brokers deliver the service by executing the actions hard-coded within them. Standalone brokers are typically used to provide simple, well-defined and specific services. On the other hand, Workflow brokers deliver the service by executing an embedded runner. A workflow broker consists of three main parts: the broker context, the runner embedded in the request, and the data contained in the request. The broker context is a generic component that contains APIs that the runners use to interface with the middleware system, such as access to the blackboard and the logging system. As depicted in Figure 5.2, the broker context is always running and monitoring the

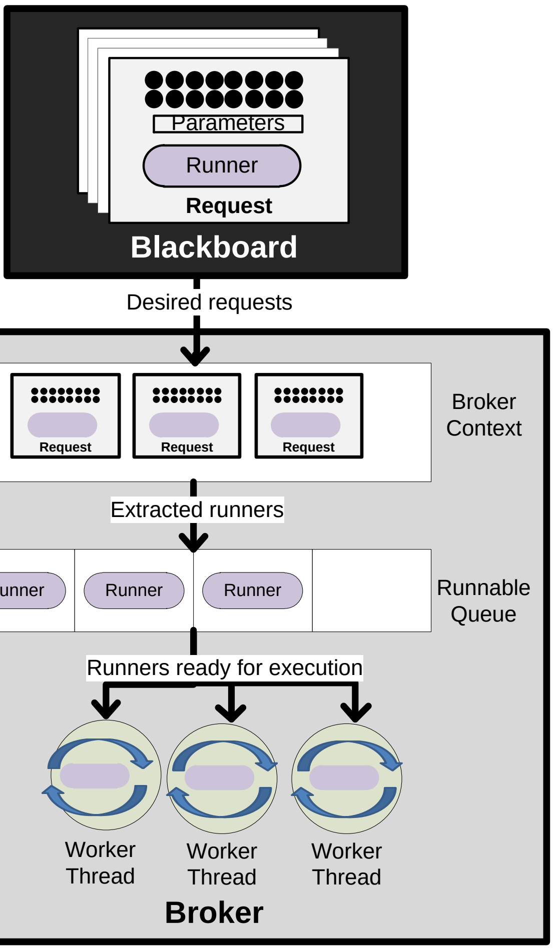


Figure 5.2: A generic depiction of actions in a workflow broker: The broker context fetches the desired requests from the blackboard and adds the embedded runners to the runnable queue. Worker threads retrieve runners from the runnable queue and execute them.

blackboard for suitable requests. It also controls the execution of the runners. The broker context maintains a set of worker threads and runners. When a runner is ready to continue execution, it is placed in a runnable queue. The idle worker threads remove runners from the queue and execute them. The workflow brokers are useful when providing complex and generic services that can be delivered through one or more workflows.

5.1.2 Indemics Middleware Platform

Indemics [18] introduced the novel concept of separating the intervention execution from propagation simulation. Databases were introduced for simulating a wide range of interventions and propagation simulation was executed using high performance computing resources. We extended the design of the Indemics Middleware Platform to achieve Web-based interactivity.

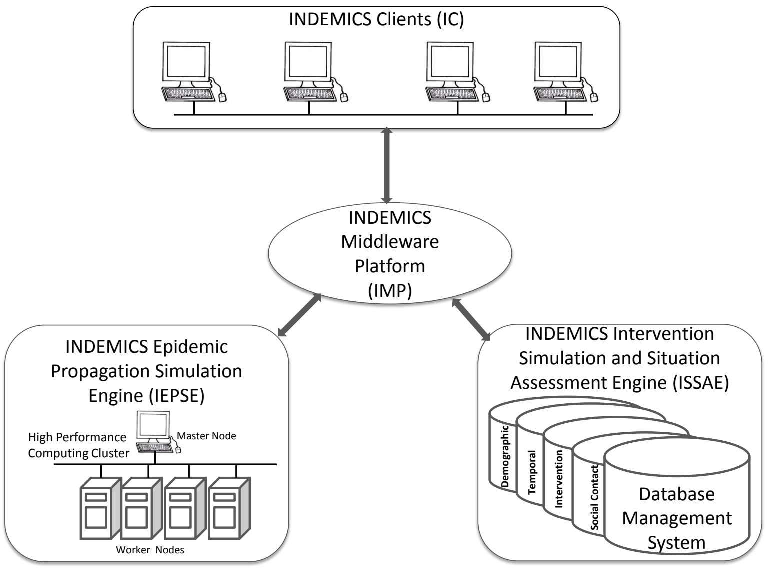

INDEMICS Clients (IC)
INDEMICS Middleware Platform (IMP)
INDEMICS Epidemic Propagation Simulation Engine (IEPSE)
High Performance Computing Cluster
Master Node
Worker Nodes
INDEMICS Intervention Simulation and Situation Assessment Engine (ISSAE)
Demographic
Temporal
Intervention
Social Contact
Database Management System

Figure 5.3: The high level architecture of INDEMICS. The INDEMICS middleware platform coordinates and synchronizes the communication between the IEPSE(INDEMICS Epidemic Propagation Simulation Engine), the IC(INDEMICS Clients), and the ISSAE (INDEMICS Intervention Simulation and Situation Assessment Engine).

The INDEMICS Middleware Platform (IMP) is the central hub in the INDEMICS framework, and is responsible for synchronizing and coordinating the interactions between the Indemics Client (IC implemented using Indemics Query Language Scripts), the ISSAE (Intervention Simulation and Situation Assessment Engine, implemented using relational databases), and the IEPSE (Indemics Epidemics Propagation Simulation Engine, implemented using EpiFast) in a distributed environment. All database accesses from the IEPSE or the IC go through the IMP.

To account for the differences in data formats across different modules, the IMP is responsible for appropriate data transformations to facilitate communication. Also, to make the INDEMICS framework independent of the specific implementations of its components and hide the implementation details of the message communication layer, INDEMICS abstracts the interactions between the IMP and other components and wraps them with a set of APIs (application programming interfaces), which are part of the IMP.

The implementation of the IMP has been designed to provide interfaces that hide low level socket communication and allow higher level abstractions for structured data to support communication with diverse modules with different data transfer and storage formats.

5.1.3 Architectural Abstraction for Web-Based Interactivity

We extended both the Simfrastructure and the Indemics Middleware Platform to achieve the architectural goals of supporting interactivity to study dynamic intervention strategies in ABM simulations and EpiFast over the Web. The extensions applied include

1. Abstracting the complexity of writing query scripts in INDEMICS Query Language (IQL), behind a web-based user interface, so that epidemiologists and public health decision makers can select epidemic parameters through the UI, rather than writing scripts.
2. Extending the user interface of SIBEL (previously called ISIS - Interface to Synthetic Information Systems) so that it is able to provide analytics on partial simulation results, allowing users to integrate real-time information in the analysis.
3. New methods for analyzing the current state of an epidemic during a simulation run, and new program abstractions to support rollback of previously applied intervention strategies without adversely affecting system response time and efficiency.
4. Design and development of the INDEMICS broker as an important backend component to facilitate interactivity. The broker plays a crucial role in orchestrating the interaction between the database and the diffusion simulator to support start, stop, pause and resume operations.

The key distinguishing factor of our implementation is that it doesn't support only a particular disease like Flu, but it can be easily extended to support other diseases, like Ebola or Malaria, if the surveillance data and the contact network pattern is available for executing the simulation. The platform also can easily support different modeling algorithms.

The overall architecture of the platform is designed to balance conflicting system requirements, for instance, speed of simulation versus usability and user experience. An epidemic simulation has to process large scale data ranging in several gigabytes at very high speeds (several GBs per second) to produce results in a timely manner. However, the information has to be presented to the users in a coherent way so users can consume the information and make decisions for further simulation execution. Hence, usability considerations often conflict with high speed processing goals, since the former requires slower processing for convenience of users, while the latter requires fast processing of large scale data. Since the requirement is for the platform to support multiple users submitting requests for simulation execution at the same time, the architecture has to cache the output for different users separately. The middleware architecture handles communication between the components, so optimal rate of interaction is supported, and the data is presented in an aggregated form to every user.

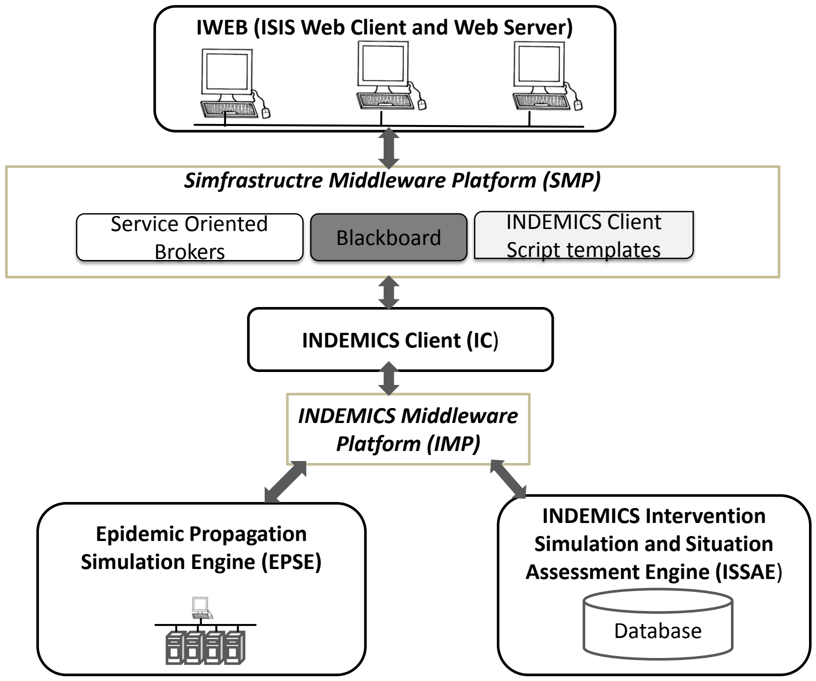

IWEB (ISIS Web Client and Web Server)
Simfrastructre Middleware Platform (SMP)
Service Oriented Brokers
Blackboard
INDEMICS Client Script templates
INDEMICS Client (IC)
INDEMICS Middleware Platform (IMP)
Epidemic Propagation Simulation Engine (EPSE)
INDEMICS Intervention Simulation and Situation Assessment Engine (ISSAE)
Database

Figure 5.4: High level architecture to support interactivity through web-based systems for evaluation of dynamic intervention strategies in ABM systems

Similarly, high-performance analytics requirements conflict with real-time information integrations. Real-time information becomes available at different rates from the environment based on availability of surveillance data. A tightly-coupled analytics environment requires complete information during processing in order to provide accurate analytical results. However, real-time information is obtained at a much slower rate; hence, integrating it into a high performance analytics environment is a non-trivial problem. To address these issues, we have designed a loosely-coupled architecture where information can be integrated when it becomes available to the relational database component, and analytical results can be updated based on new information. This ensures that the performance of the analytics environment does not degrade, while integrating real-time data from different sources.

Figure 5.4 shows the high level architecture of the analytics platform that supports the interactivity through the Web-based systems for the evaluation of dynamic intervention strategies in ABM simulations, using EpiFast as the propagation engine, Simfrastructure, and the Indemics Middleware Platform. The functional components are represented by black boxes, whereas the middleware platforms are represented by brown boxes.

5.2 Asynchronous Mechanism for Forecast Execution Using Pre-Computations

Epidemic forecasting is a very detailed process that involves large scale data in multiple formats and execution of simulations to derive the forecast outcome. The main requirement of the analytics platform that we wanted to develop is that it should be able to support forecasting of multiple diseases in different regions of the world, using several types of forecasting models.

As explained in Chapter 4, the scale of data involved in developing this integrated environment is very large. The modular database schema that we have designed based on relational paradigms allows new data sets to be added dynamically from surveillance data or from the output of epidemic simulations.

The output produced by the forecasting pipeline and stored in the database needs to be published in order to be consumed by other applications and displayed to the end users through a User Interface. The data stored in the back end cannot be tightly coupled with the UI applications, because any change to the pipeline or the forecast generation process would affect the user interactions. To avoid such tight-coupling, it is essential to develop a middleware layer that can handle the interactions between the back-end infrastructure and the UI. Moreover, the middleware has to support movement of large scale data, so that efficiency of the computations is maintained.

The main requirements of the middleware for the platform include

- the ability to provide access to epidemic data for flexible input types, including region, disease, intervention, model, and source.
- the flexibility to be agnostic to back-end data changes and execution. The execution may either be on HPC resources or pre-computed.
- the ability to add/modify/remove back-end data without affecting service guarantees or requiring platform reengineering.
- the ability to be accessible by any consumer system, or to be displayed on the Web or on social platforms.

To support these requirements, we have developed a Web services-based middleware architecture that supports communication and interaction between its component systems, mainly the Web-enabled User Interface and the Back-end Infrastructure. We use a RESTful [47] implementation of web services. The web services-based architecture is implemented using the Model View Controller (MVC) framework, where the controller accesses appropriate models, represented using DAOs (Data Access Objects). Data is passed back and forth from the web UI to the back end using JSON (Java Script Object Notation) format, which is a lightweight data interchange format. This design decouples the front-end user interface from the back-end infrastructure, and enables more flexibility and adaptability. In this way, the Web UI can be modified without affecting the back-end

Ramp-up time for 100 users	**Avg Response Time**	**Median Response Time**	**Throughput (requests/sec)**
20 sec	53 ms	28 ms	19.9
10 sec	99 ms	33 ms	10.1
5 sec	199 ms	35 ms	19.7
2 sec	1515 ms	1511 ms	38.9
1 sec	1952 ms	1928 ms	39.8

Table 5.1: Response times and throughput of the API *getOverallInfection()* when 100 users access the API over the period of 1 to 20 seconds

system logic. Similarly, the API implementation may be modified at anytime without affecting the UI, as long as the JSON output published for communication remains the same.

5.2.1 REST-Based Middleware APIs

The REST-based APIs that we have developed fall under the following categories:

- Query API - This API type provides access to static queries that get the state of the system. The consumer application requesting these services can learn about the possible forecast options that are available. For instance, *getDiseases()* returns all the diseases for which a forecast is currently available, *getRegions()* returns all the regions for which a forecast is available along with all its sub-regions, and so on.

- Push API - This API type is used to collect and push the information from the back end to the requesting system based on input parameters provided. It is also used for achieving interactive computations. For instance, *getInfectionPlot(region, timeRange)* passes back a time series-based data structure in JSON format that shows infection counts in the region over the specified time range, and *getTop3InfRegions(region, disease, intervention, week)* returns the top 3 infected subregions in the input region for a given week when the intervention type passed as input was applied on the subpopulation.

- Pull API - This API type is used to pass data into the database for storage. Pull APIs are typically accessed by systems that provide infection-related data to the back end. For instance, the information collected from users through questionnaires can provide data using this API type. For instance, *addQuestionnaireData(question, answer)* is a Pull API.

The API-based web services provided above hide the complexity of the back-end infrastructure and provide easy access to its services.

5.2.2 Web-Service Performance Evaluation

The main criteria for evaluating the performance of web services include Quality Of Service (QoS) parameters such as response time, availability, throughput, and scalability.

Currently, the REST-based web services that we have developed have been deployed on a virtual machine with a 2.93 GHz Intel Xeon processor. The virtual machine has 1 GB RAM and 122 MB reserved for cache space. The VM has 120 GB of disk space allocated to it.

Out of the total main memory available on the VM, the web server hosted on Tomcat uses 256 MB of RAM. The caching on the web server is handled through the Java internal caching mechanism.

With this configuration of web services, we expect the current deployment to support about a 100 users using the system simultaneously (ramp-up period = 10 seconds). We used Apache JMeter, an open source tool for conducting load tests, to evaluate the performance of the web services.

The performance can be improved further by changing the current configuration specified above. Since most of the computations involved are data intensive, a scale-up approach would be preferred over a scale-out approach of adding more resources, since it would be more efficient to receive results from a single VM resource than having an additional step of aggregating results gathered from multiple VM resources.

Experimental Setup

We conducted two sets of experiments. 1. To find out the number of users that the system can support simultaneously through the current deployment without degradation, and 2. To find the effects of the size of the region on the performance of epidemic forecast output through the APIs.

For the first set of experiments, we used a Push API called *getOverallInfection()*, which retrieves the overall infections in different regions of the world based on the input disease type. This API is used for fetching information on the map after the user selects the disease to be studied. Table 5.1 summarizes the results of this experiment. The number of users accessing this API is kept constant at 100, but the ramp-up time is varied from 1 to 20 seconds. The results show the response times when 100 users access the API over the specified ramp-up time. As the ramp-up time is progressively reduced, the performance of the API decreases and the response time degrades. As can be seen in the Table 5.1, the current deployment can support 100 users accessing the system within 5-10 seconds with no visible lags.

For the second set of experiments, we used 2 APIs - *getInfectionPlot()* and *getTop3InfectedRegions()*, which are used extensively to fetch the plot of the epidemic forecast, and for getting the top 3 infected subregions in the given region respectively. We conducted experiments on the regions California (Population: 33.6 million), Virginia (Population: 7.25 million) and Wyoming (Population: 0.56 million) in the USA; and on Delhi, India(Population: 13.9 million) for flu. We also studied the APIs for Ebola for the Liberia region (Population: 4.2 million). For this experiment set, we used 100 users over the period of 10 seconds for each API test.

API Name	**Region**	**Average Response Time**	**Median Response Time**	**Throughput (requests/second)**
getInfectionPlot()	Virginia	88 ms	34 ms	10
	Wyoming	90 ms	45 ms	10
	California	99 ms	33 ms	10.1
	Delhi	57 ms	33 ms	1.2
	Liberia	89 ms	31 ms	10.1
getTop3InfRegions()	Virginia	154 ms	132 ms	11.2
	Wyoming	254 ms	156 ms	10
	California	190 ms	133 ms	10.1
	Delhi	111 ms	54 ms	10
	Liberia	76 ms	44 ms	10

Table 5.2: Response times and throughput of 2 different APIs accessed with different regions and diseases

Table 5.2 summarizes the results of our experiments. From the experiment results, it is evident that the API response time and throughput of the API is independent of the size of the input region and the disease for which the forecast is sought. This shows the stability of our application to support the addition of any new region or disease without degradation in performance of the system.

5.2.3 Multi-level Web Services

An ideal middleware system should be completely decoupled from the data. However, in an ORM (Object-Relational Mapping) based approach, the service accessing the data from a database has to embed the code for database connectivity, and needs to support a particular structure of the database. For example, the code for JDBC connectivity is required to connect to an Oracle database from Java. Hence, if there is any change in the type of database used or the underlying database structure, then the entire code base needs to be changed to handle the change in the database type or structure.

In order to decouple the database from the web service code (partially), we have developed a multi-level Web services design. In this design, the web services are developed at multiple levels, where only the lowest level APIs are directly connected to the database. These APIs are closely tied to the database structure and embed the code for database connectivity. The next level APIs are abstracted from the database and are built so they can call multiple lower level APIs to achieve their goal. At the highest level, the APIs can call the APIs at the level directly below them or even at the lower levels as can be seen in Figure 5.5.

Using this approach, the lowest level APIs can be connected to any type of database including

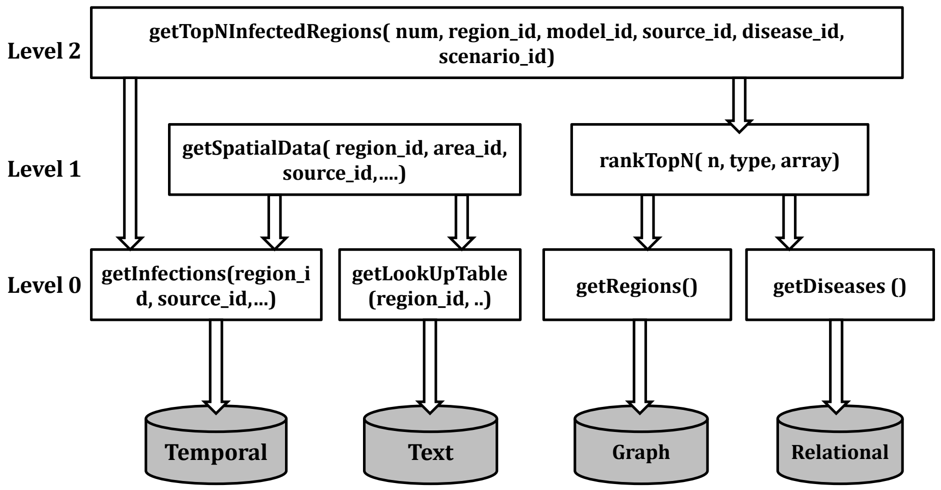

Level 2
getTopNInfectedRegions(num, region_id, model_id, source_id, disease_id, scenario_id)
Level 1
getSpatialData(region_id, area_id, source_id,....)
rankTopN(n, type, array)
Level 0
getInfections(region_id, source_id,...)
getLookUpTable (region_id, ..)
getRegions()
getDiseases ()
Temporal
Text
Graph
Relational

Figure 5.5: Multi-level API design in the development of the forecasting analytics platform

MySQL, Oracle, neo4j, flat files and so on. The higher level APIs are agnostic to the type of database used. If there is any change in the database, only the level 0 APIs need to be changed. Level 1 APIs can access only level 0 APIs. Level 2 APIs can access both level 0 and level 1 APIs. Level 0 APIs are used to pull atomic data. This includes APIs such as getRegions(), getDiseases() and so on. The higher level APIs have more specific information. For instance, rankTopN(n, type, array) ranks the type of array based on whether it is regions, diseases or infection counts. Level 2 API getTopNInfectedRegions() accesses the rankTopN() API along with the getInfections() API at level 0 to compute the top infected regions at any geographical level.

5.2.4 Security in Exposing Web Services

Multi-level Web services can be used to provide security to the underlying data that is exposed to the external applications through the APIs. Using multi-level Web services, only the top-level Web services are exposed to the outside world. The lowest level Web services at level 0 that embed the code for database connectivity and the data structure are completely abstracted from external applications and are only available as internal web services.

The top level Web services can also implement a protocol such as the OAuth2, which is an open standard for authorization. Another similar security protocol that is used widely for user authentication is called OpenID. Using OpenID, the application allows access to the user because it trusts the OpenID Identity provider. On the other hand, OAuth2 allows the authorization servers to issue access tokens to third party applications with the approval of the resource owner, which can then be used for authentication. Hence OAuth2 can provide authorization so only a subset of the data is made accessible to the requesting applications.

5.3 Summary of Technical Contributions

1. Design of an asynchronous method for submitting execution requests for intervention evaluation in large scale epidemic studies using fine-grained agent-based simulation models, and interactivity through Web-based interfaces.
2. An asynchronous mechanism for publishing results of forecasts involving pre-computations using a REST-based API implementation and providing flexibility for the analysis of forecast results.
3. Design and implementation of multi-level Web services design to decouple the data and abstract the data definitions from requesting applications.

Chapter 6

Applications

In the previous chapters, we described the data methodologies and service oriented web services that we have implemented that are the key components of the integrated environment for epidemic evaluation and forecasting. We have used our analytical platform for building several applications related to epidemic evaluation and forecasting. The main applications that we have developed include DISIMS, FluCaster and EpiCaster. We describe these applications in detail in the following sections.

6.1 DISIMS

DISIMS (**D**istributed **I**nteractive **S**imulation **S**ystem) [34] is a flexible epidemiological modeling environment, which combines high resolution individual-based epidemic and intervention modeling environment with web-based user-friendly analytics. DISIMS can be used by policy makers and epidemiologists for undertaking a broad range of counterfactual computer experiments and for analyzing results through detailed graphs and plots inside the system. It also allows export of result data in standard formats for analysis using other tools. Additionally, the modeling environment can be used for training analysts in the use of complex epidemiological models.

DISIMS is an interactive modeling environment and interactivity is one of its key technical strengths. DISIMS allows an analyst to start, stop, pause, resume and roll back previously applied intervention strategies and disease propagation processes. Users can ask complicated spatio-temporal queries in support of situation assessment. DISIMS aids policy makers interested in developing dynamic health policies – policies that can adapt to new data that become available via surveillance. This is an important issue in epidemiology. See a recent paper by Yaesoubi and Cohen [99] for additional discussion. Developing such interactive simulations and computational steering environments, especially for parallel simulations, is a well known challenging problem. DISIMS achieves this by exploiting the problem specific semantics that allow one to achieve these features using a relatively small data footprint.

DISIMS uses existing software modules that are re-engineered appropriately to achieve the design goals. The data storage and communication mechanisms ensure that there is no bottleneck due to large scale data movement. The software subsystems that were part of the integration effort include EPIFAST [19] – an HPC-based simulation engine, that simulates disease propagation process over a given region; ISIS [16] – a web-based visual interface tool, that can be used for experiment set-up and analysis of the role of different parameters in disease propagation; and a database repository, storing and operating on the demographic and geographic information, extended from INDEMICS [18]. We analyzed feasibility of the existing middleware platforms such as Simfrastructure [21] and the middleware used in the implementation of INDEMICS, to support integration of these distinct system components and remodeled the middleware for supporting optimal data movement and interactivity.

DISIMS is specifically designed to improve user experience. This includes ease of use of the system and user productivity. Care has been taken so that a user can drive computational experiments by accessing complicated mathematical models without having to become a computing expert. For instance, the environment provides automated services for experiment set-up and management, thus reducing the overall time of conducting end-to-end experimental studies. It also allows reuse of past epidemiological experiments and their results, which avoids duplication of efforts. Also, using DISIMS, users can view and analyze partial and complete simulation results through graphs and plots without having to perform manual analysis. This automates the analysis to a large extent and leads to considerable gain in productivity of users.

6.1.1 Architecture and Implementation

DISIMS (**D**istributed **I**nteractive **S**imulation **S**ystem) is an interactive high-performance modeling environment for epidemiological simulations that integrates real-time information based on surveillance data into the simulation. DISIMS leverages three distinct modeling components built by our group: (*i*) EPIFAST [19] – an HPC-based simulation engine for epidemic propagation simulation; (*ii*) INDEMICS [20], an interactive modeling platform that provides a database repository for intervention selection and application, external to the propagation simulation engine. DISIMS also extends the interactive client and the loosely coupled Middleware Platform introduced in INDEMICS; (*iii*) and ISIS [16] – a web-based visual interface tool, to develop a truly interactive modeling environment. Each of the three modules was extended and re-engineered to support the functionality of DISIMS. In addition, the Simfrastructure Middleware Platform [21] was reengineered to support interactivity and communication between the component modules.

The DISIMS architecture has two parts: (*i*) Functional components of DISIMS architecture, and (*ii*) Middleware platforms to support the movement of data and control between components.

The high level architecture of DISIMS is based on the functional and middleware components shown in Figure5.4 in Chapter 5, to support interactivity through web-based systems. The functional components are represented by black boxes whereas the middleware platforms are represented with brown boxes.

In the subsequent sections, we describe these architectural components of DISIMS in detail.

6.1.2 Functional Components of DISIMS Architecture

The DISIMS architecture is comprised of three primary functional components:

- Epidemic Propagation Simulation Engine (EPSE) extended from EPIFAST
- INDEMICS Intervention Simulation and Situation Assessment Engine (ISSAE) and INDEMICS Client extended from INDEMICS
- ISIS Web Client and Web Server (IWEB) extended from ISIS

EPSE

EPSE is a high performance simulation engine that simulates the spread of epidemics through large scale populations, capturing the co-evolution of individual health, behavior, and disease transmission. It also has the capability to execute multiple replicates in order to capture the variability of the results due to the stochastic nature of the phenomena being modeled. In DISIMS, EPSE is implemented using an extended version of the propagation simulation engine, EPIFAST [19]. EPSE can execute the propagation process at a very rapid pace, simulating disease spread over multiple time-steps (i.e., simulated days). For instance, EPSE can execute epidemic simulation on a cluster of 10 nodes with eight cores each, for a city with a population of about two million, in less than 30 seconds. The speed of simulation can be further improved by adding more nodes to the cluster.

EPIFAST was initially designed to run to completion without stopping. To support interactivity, the EPSE component was engineered to support *pause*, *resume* and *roll-back* operations. Furthermore, when the system is paused, an analyst can examine the state of the system as well as modify the currently active set of interventions.

A single simulation run is performed for several simulation days or time-steps. Since the data between EPSE and ISSAE is passed back at every time-step and contains a set of PIDs, which may run into thousands of vertices, depending on the extent of the infections in the population, the scale and frequency of data transfer is large. For instance, a typical epidemic simulation is carried out for a period of 200-300 days, over which an analyst can study the effects of different mitigation strategies on epidemic spread in a particular region. If the simulation is run for a city with a population of say 2 million people, then it is possible that the number of infected individuals can run into several thousands on any particular day or days. This infection information is captured in EPSE and needs to be passed back to ISSAE within milliseconds if the entire simulation has to complete within a reasonable time of minutes. For simulations over larger regions such as state level, the scale of infections would be higher per day and hence the data to be transferred between EPSE and ISSAE modules would be higher. The typical amount of data transferred between EPSE and ISSAE modules for simulation at a city level for a city with a population of about 2-5 million individuals, is around 15-20 KB/time-step.

Handling this scale of data requires a specialized middleware platform implemented using the INDEMICS Middleware Platform (IMP) explained in Section 6.1.3.

Whenever a user requests a simulation to be paused after a certain duration to be able to analyze the results of the intervention strategy applied till that point, then EPSE holds the connection with the INDEMICS Middleware Platform just like it is expecting data for the next iteration. When a new INDEMICS client script is generated through the SMP it is passed to IMP and IMP makes the necessary data conversions to pass data to EPSE without having to make any other changes. We discuss the specialized middleware - INDEMICS Middleware Platform (IMP) and Simfrastructure Middleware Platform (SMP) in Section 6.1.3.

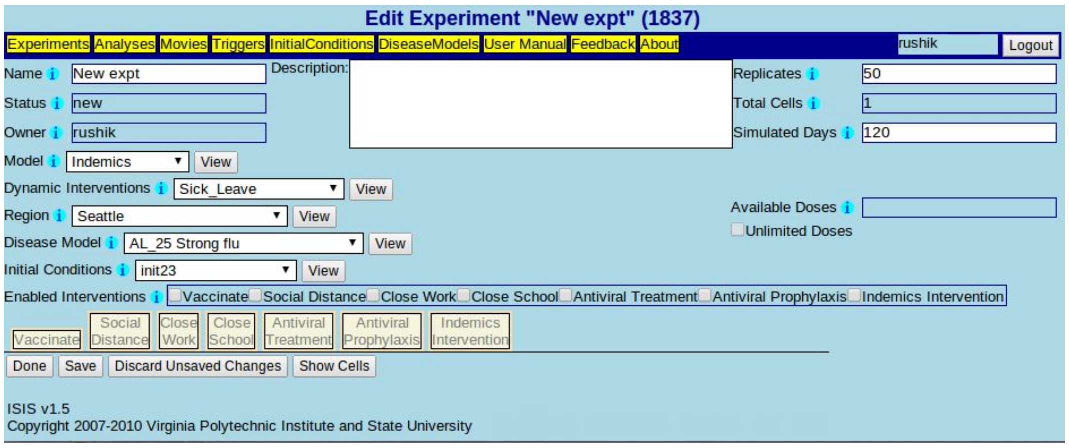
ISSAE and INDEMICS Client

ISSAE simulates application of intervention strategies during an ongoing epidemic simulation externally to the high-performance simulation engine. ISSAE component of DISIMS is implemented using a relational database management system, extended from the INDEMICS architecture. ISSAE is the main component that stores the data that we have stored in our analytical environment.

ISSAE stores demographic and social contact information along with time-varying infection data about individuals in relational format. Use of relational databases to represent ISSAE provides atomicity, consistency, isolation and durability to the data along with features such as indexing for fast retrieval. This allows real-time information to be incorporated quickly into the system so that the users can perform situation assessment. The ISSAE is updated with information of infected individuals at each time-step by EPSE. To get updated data on subpopulations to be intervened based on situation assessment, ISSAE can be queried using scripts written in INDEMICS query language (IQL), which is an extension of SQL.

The INDEMICS Client component of DISIMS is closely associated with ISSAE. This component reads the INDEMICS Client scripts or IQL scripts written in INDEMICS Query Language and feeds the queries to ISSAE through the middlware, IMP (INDEMICS Middleware Platform). To hide the complexity of writing and executing IQL scripts from users, the DISIMS architecture automates the creation of dynamic IQL scripts based on predefined INDEMICS client script templates. These templates are stored in a database of intervention scripts in the Simfrastructure middleware and invoked based on user selection dynamically. Also, the actual parameters of execution selected by users using the ISIS Web Client are substituted in the invoked script during run-time. With this feature, analysts and epidemiologists are freed from the burden of writing complicated IQL scripts and can focus on finding optimal intervention strategies for containing epidemics.

Separating the ISSAE component from the simulation engine provides users with the flexibility to choose different types of interventions dynamically and also to choose the subpopulation to apply


Edit Experiment "New expt" (1837)

Name	New expt
Description:	
Status	new
Owner	rushik
Model	Indemics
Dynamic Interventions	Sick_Leave
Region	Seattle
Disease Model	AL_25 Strong flu
Initial Conditions	init23
Replicates	50
Total Cells	1
Simulated Days	120
Available Doses	
Unlimited Doses	☐

Enabled Interventions ☐Vaccinate ☐Social Distance ☐Close Work ☐Close School ☐Antiviral Treatment ☐Antiviral Prophylaxis ☐Indemics Intervention

ISIS v1.5
Copyright 2007-2010 Virginia Polytechnic Institute and State University

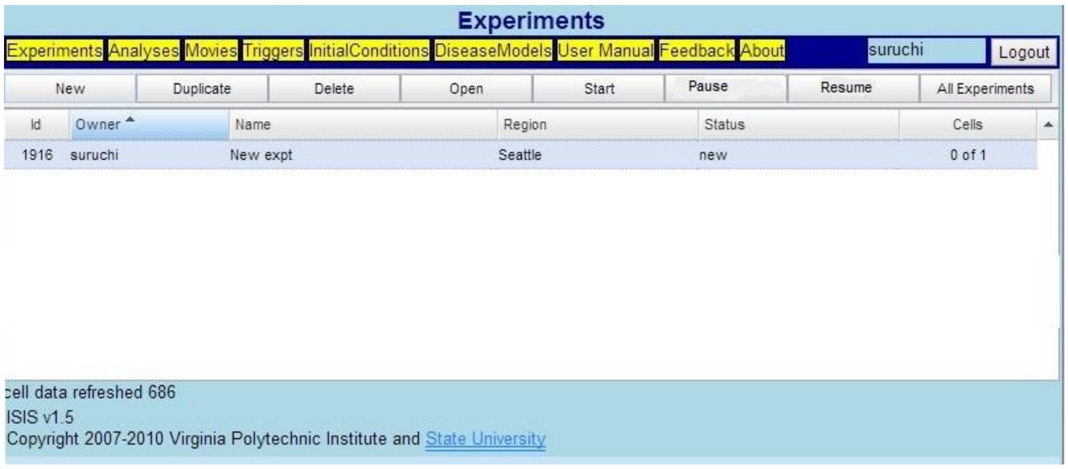
Figure 6.1: SIBEL-based UI for automated set-up and management of experiments in DISIMS.

interventions on.

ISIS (SIBEL) Web Client and Server (IWEB)

IWEB is the user-interface component of DISIMS which includes a combination of ISIS Web Client and Web Server. It is an extension of the original ISIS system developed by our lab based on several years of research and extensive interactions with policy analysts over a 10 year period. The ISIS system has been recently renamed to SIBEL. Our early work in [43] led to a realization that models can be made more useful only when they are easily accessible to end-users. SIBEL allows users access to models such as EPIFAST [19] and EPISIMDEMICS [12] through a web-based interface. It allows selection and analysis based on a range of parameters such as disease models, efficacy of interventions, compliance rate and so on, that play an important role in epidemic propagation. In addition, SIBEL provides embedded management and storage of experiments that saves time to set-up and manage epidemiological experiments. Figure 6.1 shows a screen shot of SIBEL with provision for selection of input parameters such as region, disease model, number of replicates, dynamic interventions and so on.

We extended the original ISIS Web client to support interactive features such as *pause resume* and *roll-back* operations. Earlier, users could only save an experiment and start the experiment by clicking the "Start" button. We added additional buttons on the user interface - "Pause" and "Resume", that provide the flexibility to make a single simulation run interactively. See Figure 6.2 for a snapshot of the available buttons on the UI. This figure shows the buttons that allows users to pause an ongoing simulation and resume with a different set of input parameters in a single experimental simulation run. Users can also specify running the simulation for a small duration by selecting the "Duration" parameter in the Interventions tab, after which the simulation pauses automatically. The constraint here is that the specified duration has to be smaller than the total number of simulated days. When the user clicks the "Resume" button, the simulation resumes



Id	Owner	Name	Region	Status	Cells
1916	suruchi	New expt	Seattle	new	0 of 1

cell data refreshed 686
ISIS v1.5
Copyright 2007-2010 Virginia Polytechnic Institute and State University

Figure 6.2: Start, pause and resume buttons in the SIBEL-based UI of DISIMS.

again with same or different set of parameters, selected by the user.

This added functionality allows users to analyze the effects of multiple intervention strategies on disease propagation in a single simulation run. Users can analyze partial simulation results by viewing analytical plots and graphs that are generated in the "Analyses" tab of ISIS. Based on the results, they can decide whether to continue the simulation with different parameters or roll back to a time-step in the past and resume with different set of interventions.

Analysis of partial simulation results also enables better situation assessment and real-time information integration. For instance, if it is known that a limited quantity of anti-virals would be available in the market only 15 days after the epidemic starts propagating, an analyst can pause the simulation on day 14 and analyze which subpopulation is the most affected and can benefit from the dose of anti-virals. The subpopulation may be based on age group, gender or other demographics. The simulation can then resume with a new set of anti-viral interventions applied to the particular subpopulation.

6.1.3 Middleware Platforms To Support Data Movement and Interactions

Design and development of middleware platforms is an important part of DISIMS architecture to support communication and movement of data between the functional components described in Section 6.1.2. The backend infrastructure, consisting of the EPSE, ISSAE and INDEMICS Client has extremely high data speed requirements to maintain simulation performance, whereas the front-end infrastructure, IWEB is used for interaction with users. These systems have varied performance requirements and hence a single middleware platform cannot be used as a common means of communication throughout the system.

At a macro level, there is need for data and control passing middleware mechanism between the front-end and the back-end infrastructure; and a platform to support high performance speed requirements between components of the back-end infrastructure.

The IWEB including the ISIS Web Client and ISIS Web Server interface represent data and operations at a higher level of abstraction for the convenience of end users. ISIS Web server is typically deployed on a single dedicated web server that hosts the ISIS application. EPSE, on the other hand, implements an MPI based algorithm operating at low levels of abstraction and requires high-performance computing resources such as grids or multi-node clusters for execution. Hence, the ISIS Web Server and the HPC-based EPSE simulation engine cannot be co-located on the same machine instance.

The input parameters selected by users through the ISIS Web Client and passed to the ISIS Web server, need to be relayed to the EPSE through some communication mechanism. The amount of data passed as parameters is usually small in scale. Once the propagation simulation starts at the EPSE, it may run for a long time depending on the parameters of execution or until it is paused or rolled-back. Hence, the nature of communication between the ISIS Web Server and EPSE is largely asynchronous. Moreover, even with interactions, the EPSE has to execute for some time-steps before the user can analyze the effects of any intervention. Hence the frequency of communication between the ISIS Web Server and EPSE is small.

We evaluated the applicability of "Simfrastructure" [21] for achieving communication between the front-end and back-end infrastructure of DISIMS. To enable communication between the ISIS Web Server and EPSE, we extended the blackboard and interface broker components of Simfrastructure. We engineered a new service broker component called the INDEMICS broker inside Simfrastructure. The INDEMICS broker is one of the most important components of DISIMS. It automates the process of starting, stopping or pausing the simulation. The INDEMICS broker maintains a database of several distinct client program templates written in IQL, corresponding to different intervention studies.

The INDEMICS broker continually monitors the blackboard for new simulation requests for DISIMS. When a new request is found on the blackboard, it invokes the appropriate template from the database and overwrites the actual parameters selected by users on the script template. When a user pauses, resumes or rolls-back a simulation, the INDEMICS broker interrupts the EPSE, which then handles the change in state as described in Section 6.1.2. The design of INDEMICS broker is one of the important contributions of DISIMS. Overall, with the extension of Simfrastructure, DISIMS is able to support variability in the implementation of its functional components and provide an asynchronous mode of communication between them.

The other aspect of the DISIMS architecture is to deal with large scale data communication between the EPSE and the ISSAE, implemented using the relational database. If ISSAE is directly connected to the EPSE, then any change in the implementation of ISSAE would need changes to the high performance code of EPSE. For making the system modular, flexible and adaptable, the EPSE and ISSAE have to be connected through an optimized middleware platform. Since EPSE executes a simulation over several time-steps (typically represented as days), data has to be retrieved from the database and passed to EPSE over many time-steps. This data passed back and forth can range up to several megabytes, as explained before. An orchestration mechanism of a service oriented architecture can slow the speeds of the EPSE. Also, a service oriented abstraction such as the one provided by Simfrastructure, where service requests are made asynchronously cannot be used since consistency of data has to be guaranteed in ISSAE.

INDEMICS [18] introduced the concept of using a high-performance middleware platform that is optimized for communication between the relational database component and high-performance simulation engine. We decided to reengineer the INDEMICS Middleware Platform for our communication needs between ISSAE and EPSE.

Using IMP as the additional middleware, large scale data volumes can be supported per time-step across multiple simulation runs. IMP has an optimized queuing mechanism to queue data to be passed back and forth between the EPSE and the ISSAE. IMP also has features for data interpretation to speed up data mapping and transfer process. This ensures optimal performance of the simulation engine within DISIMS. As can be seen in Figure 6.4, the presence of two middleware systems - SMP and IMP connected together in DISIMS, instead of a single generic middleware, allows the simulation system to accomplish its usability goals along with performance.

6.1.4 DISIMS User Workflow

In this section, we describe a detailed user workflow of DISIMS and its effect on the data and context flow within DISIMS. The DISIMS platform can be accessed by users using any standard web browser. The ISIS Web Client of DISIMS allows users to set-up and execute a simulation experiment. Figure 6.3 shows the sequence of events that take place inside DISIMS, when a user submits a request to start a new epidemiological simulation experiment. The sequence of events can be described as a series of steps 1 to 19 as follows:

- Step 1: As the first step to start a simulation experiment, a user has to select the parameters of simulation execution. Parameters of execution include the region of experimental study, number of days of simulation, number of replicates of simulation, disease model, initial conditions and so on. In addition, users can select intervention strategies to be applied to the propagation process such as vaccination, social distancing and so on. Requests can also be submitted to perform some complex epidemiological experiments by applying dynamic interventions such as the Block-based intervention strategy.

- Step 2 : Once the experiment is set-up and submitted using the ISIS Web-client, the request is received by the ISIS Web server to start a simulation with input parameters. The ISIS Web server passes this information as input to the interface broker of the Simfrastructure Middleware Platform (SMP).

- Step 3: The interface broker interprets the data, bundles the parameters as a 'Simulation request' and submits it on to the blackboard of the SMP.

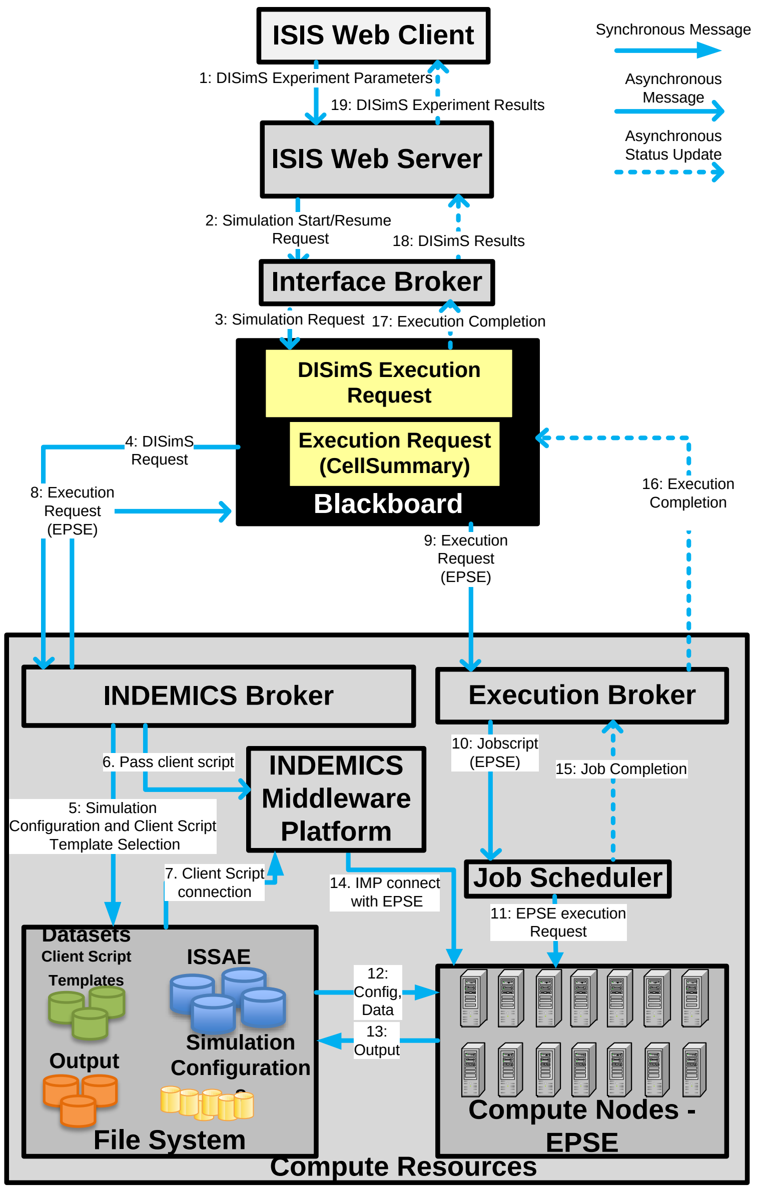


Figure 6.3: Workflow diagram showing the sequence of events that take place inside DISIMS when a request for simulation execution is submitted at the beginning of the simulation or resumed from Paused state. Reference : Deodhar et al. [34]

- Steps 4-5 : The INDEMICS broker, engineered as one of the sub-components of SMP, continually monitors the blackboard for new simulation requests for DISIMS. Once a service request embedded with the required parameters is found on the blackboard, it unpacks the request, invokes the correct INDEMICS script template from the SMP database and replaces the template parameters with the actual parameters selected by users.

- Steps 6-7 : During the same time, the INDEMICS broker hands over the created INDEMICS client script, written in IQL to the INDEMICS Middleware Platform (IMP) to communicate directly with ISSAE and EPSE. The dynamically generated IQL-based client script can be interpreted by the INDEMICS Middleware Platform.
- Step 8 : The INDEMICS broker creates a new bundled execution request for EPSE and places it on the blackboard for starting the propagation simulation on EPSE.
- Steps 9-11 : The execution broker monitors the blackboard for any new execution requests. When it finds a request, it starts execution of propagation simulation on the available compute resources such as a cluster or cloud, through a local job scheduler.
- Steps 12-13 : The required data sets and configuration parameters for job execution are read from the file system and the intermediate results are written back for further processing.
- Step 14 : The IMP is configured as a background process that is always in a running state. Once it receives an IQL script in Step 7, it monitors to see if the EPSE has started execution. If the EPSE has started execution, then IMP establishes connection with it. Based on the invoked intervention script, the IMP connects to appropriate database tables, retrieves intervened population data and passes it as an intervention to the EPSE. The EPSE continues to execute over several time-steps in this manner, receiving intervention data from IMP and executes until completion or until paused.
- Steps 15-19 : Once the simulation is completed, the results are written back on the blackboard as shown in Figure 6.3 and are displayed to the users through the ISIS Web Client.

DISIMS has a component called the “Analysis broker” that is configured to run analysis scripts based on the analysis request made by a user, similar to the execution broker,. The request for analysis of a particular experiment is made to an Analysis server that runs the R statistical software tool and follows similar workflow as above. The results of analysis are written on to the blackboard and consequently passed to the ISIS Web Client through the interface broker, where the corresponding graphs are plotted for analysis by the user. The user may also decide to use DISIMS interactively by pausing the simulation after a certain duration and running analysis on partial simulation execution. Based on results of the analysis, the user may decide to apply a different set of parameters including a new intervention strategy or continue with the same parameters for the rest of the simulation. A resume request issued by the user after pausing a simulation follows similar flow of data as given in Steps 1 through 19.

If a new dynamic intervention strategy is selected by the user, then the INDEMICS broker invokes a new client script starting from the day/time-step the simulation was paused. This new client script connects to the same session with the INDEMICS Middleware Platform. The IMP is capable of storing state information for each simulation session. It holds its connection with the EPSE corresponding to an ongoing simulation for a user and sends out new interventions to the EPSE by selecting new subpopulation data from ISSAE. In this way, an end-to-end interactive simulation can be executed using DISIMS.

6.2 FluCaster

We have developed FluCaster [36], a pervasive and scalable web application for situation assessment and forecasting of Influenza-like Illness (ILI), commonly referred to as the flu. Using FluCaster, one can assess the prevalence of ILI at highly resolved spatio-temporal levels. Importantly, FluCaster can also provide the user with short and long term forecasts, in the presence of interventions applied on specific sub-populations or without any mitigation strategies.

FluCaster is comprised of three basic components that are based on the predictive analytics platform described before (i) a web-enabled user-interface; (ii) a middleware that coordinates interactions between the UI components, the back end models and the data store; and (iii) a back end that is comprised of high resolution epidemic simulations, combined with optimization routines for forecasting as well as situation assessment. The back end also stores and operates on the GIS data consisting of maps and geographic locations, synthetic population data corresponding to population demographics, and synthetic contact network data for the different regions under consideration.

The underlying mathematical models that we have implemented involve highly resolved information on different regional demographics to compute the forecasting output and form the basis of the forecasting pipeline.

6.2.1 Architecture and Implementation of FluCaster

FluCaster is based on a modular architecture that allows it to be flexible with respect to the analytical queries that it can answer and scalable with respect to the volume of data that it supports.

Figure 6.4 shows the detailed architecture of FluCaster. The main building blocks of FluCaster can be categorized as follows:

- FluCaster Web-enabled User Interface
- FluCaster Back-end Infrastructure
- FluCaster Web Services-Based Middleware

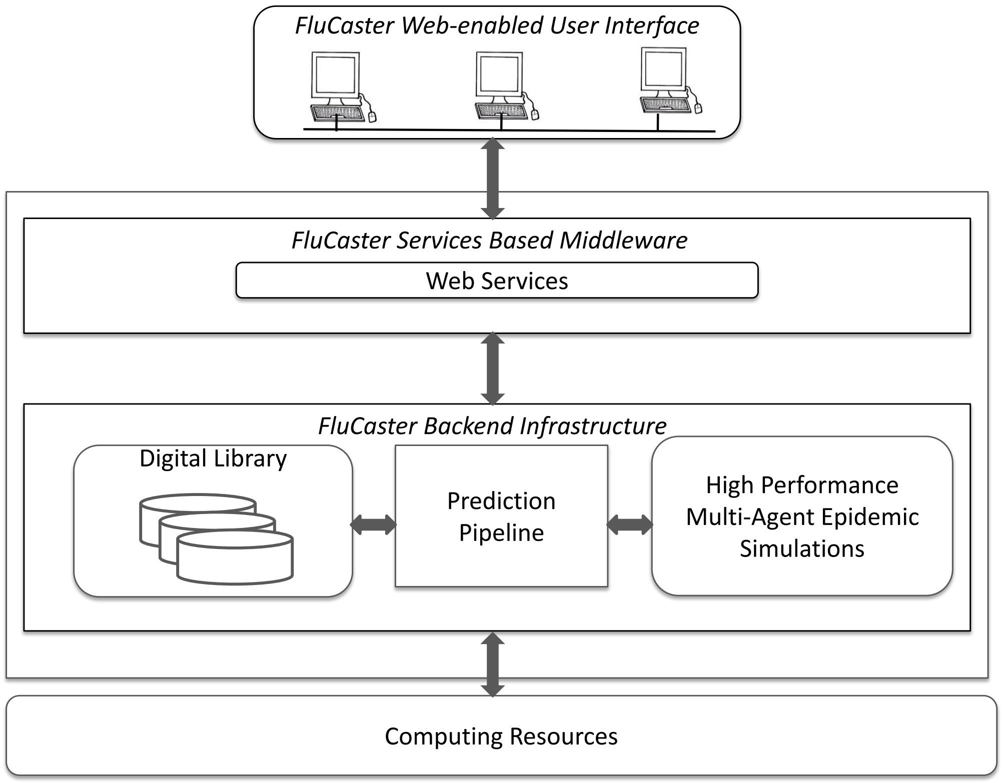

FluCaster Web-enabled User Interface
FluCaster Services Based Middleware
Web Services
FluCaster Backend Infrastructure
Digital Library
Prediction Pipeline
High Performance Multi-Agent Epidemic Simulations
Computing Resources

Figure 6.4: Detailed architecture of FluCaster - a web application for situation assessment and model driven forecasting of flu epidemics

6.2.2 FluCaster Web-Based User Interface

FluCaster is a Web-based system that can be accessed by end users through a Web browser to view trends in flu outbreaks in different regions of the world, and to access forecasts of the outbreak for the upcoming time period. FluCaster displays all the regions where the flu outbreak is prevalent on a time-varying heat map of the world. The user can then select a particular region on the map to get a fine-grained assessment of the current flu activity as well as forecasts within that region. Figure 6.5 shows a snapshot of FluCaster as a standalone web application. The application shows a heat map for the entire US region on the left, and on selecting the California region, the heat map shows infection counts in each county of California at a much granular level. As can be seen in the graphic on the right, users can also view the extent of the current flu epidemic and the forecast through the time series epidemic curves, commonly referred to as the "epicurves". The epicurves are available for both the surveillance data as well as the forecast data.

FluCaster's Web-enabled User Interface consists of the following main components.

- Interactive Timeline: FluCaster provides a timeline covering a range of weeks over which a user may view epidemic information. The available timeline options include the current

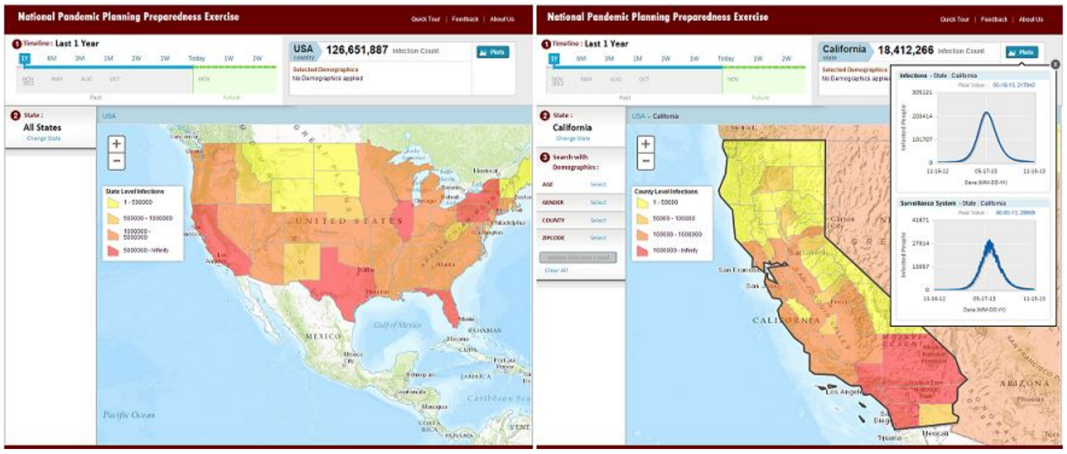

National Pandemic Planning Preparedness Exercise
Quick Tour | Feedback | About Us
Timeline : Last 1 Year
USA 126,651,887 Infection Count
California 18,412,266 Infection Count
Plots
State : All States
State : California
Search with Demographics :
AGE
GENDER
COUNTY
ZIPCODE
State Level Infections
County Level Infections
Infections - State : California
Surveillance System - State : California

Figure 6.5: Snapshot of FluCaster UI as a standalone web application.

week and any of the last four weeks for situation assessment; users can also select the timeline for up to two weeks in the future to view forecasts.

- Interactive heat map: FluCaster provides an epidemic heat map that shows infection levels aggregated at multiple geographical levels, such as state, county and block group. Data visualization on the map is provided using the Geographic Information Systems data or the GIS data. Once the timeline is selected, FluCaster displays epidemic activity in the selected region. The interactive heat map is displayed with different color schemes to allow the user to analyze the severity of the epidemic in the region. The user can also drill down to each subregion of the map to view infections at a granular level. The size of the population of different regions makes the computation and the resulting visualization complex. For instance, the average size of a United States Census block group is about 4,000 people. To retrieve and display the results of forecasts for flu for each block group, in each county, in each state of the USA is a challenging task. The data schema that we have implemented at the back end makes data retrieval and presentation possible in an efficient way.

- Plots (Epicurves): FluCaster allows the users to view epicurves showing infection trends in a region over a time period, as well as the peak infection count in the specified time period. Users can view the epicurves of any region for the past and current weeks based on surveillance sources such as GFT and CDC, and view forecast plots for the future weeks.

- Selection of Demographic Parameters: FluCaster allows the users to analyze the effects of an ongoing flu epidemic on specific subpopulations based on their demographic attributes. For instance, the current implementation of FluCaster allows users to filter the population and view infection counts by age, gender, zipcode and county.

Currently, FluCaster UI is hosted on a standalone Apache Tomcat Web server hosted on a single virtual machine, as can be seen in the Deployment diagram 6.6. This virtual machine also has a

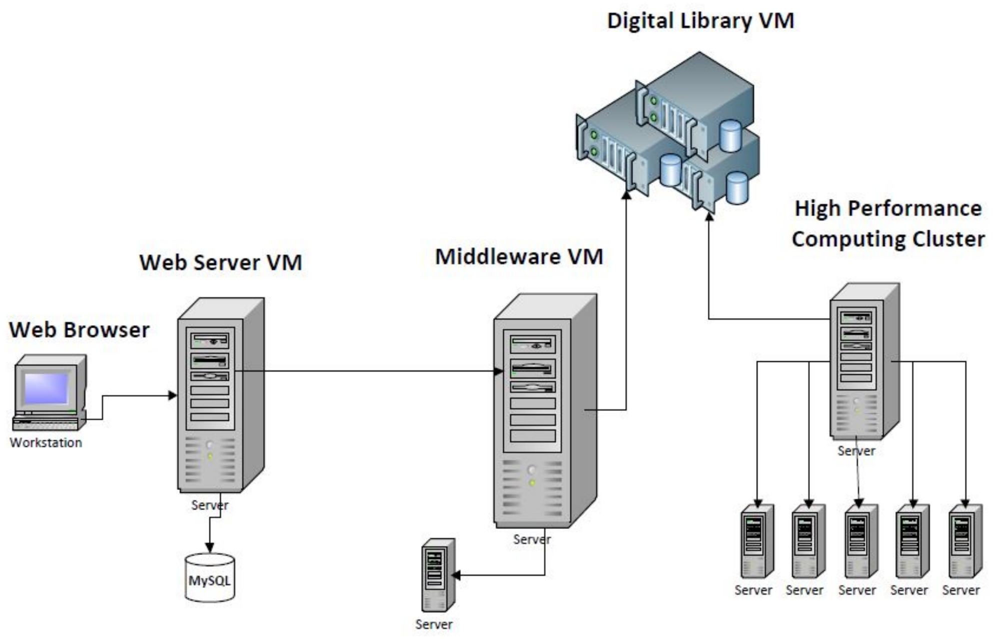


Figure 6.6: Deployment diagram of FluCaster/EpiCaster

light-weight relational MySQL database installed on it for storing user authentication information. This MySQL database may also be used for storing some pre-computed information for improving performance. Based on the selections made by the user through the UI, the web server decides whether to pass back pre-computed data from the local MySQL server or to invoke appropriate web services from the back-end infrastructure to fetch data.

The complexity of the FluCaster UI lies in the rendering of the dynamic map based on user selections without any visible delays or performance degradation. Many of the user selections require large-scale analytical queries to be computed at the back end. To avoid lags and delays in computing this information online, the web server is optimized to cache some of the static content displayed on the map as well as some large scale pre-computated output. The back-end web services are invoked only when dynamic information is sought; this ensures optimal performance of FluCaster operations.

6.2.3 FluCaster Back End Infrastructure

The back end infrastructure of FluCaster is the most important component of the FluCaster architecture and is part of the integrated modeling environment that we have developed for forecasting of various epidemics. The backend is responsible for generating forecasts for flu epidemics based on high-resolution epidemiological models, implemented using agent-based modeling techniques.

The back end not only handles collection of surveillance data from different surveillance sources, but also execution of large-scale simulations to generate and store flu forecasts.

Following are the main components of the back-end infrastructure of FluCaster:

1. An epidemic forecasting pipeline that handles surveillance data collection, executes multi-agent epidemic simulations, and uses a non-linear optimizer for forecast matching and generation. The forecasting pipeline is explained in detail earlier in Section 3.2.2.
2. A data repository storing complete information related to situation assessment and forecasting of flu epidemics. The repository also stores surveillance data in addition to forecast data.

The data stored in the database for flu forecasting is broadly classified into four main categories as follows. We have described the details of the data and storage in Section 4. (*i*) Social Contact Network data representing the set of proximity relationships for the given region, (*ii*) Demographic data about individuals in the given region, (*iii*)Intervention data about the mitigation strategies applied, and (*iv*) Temporal (time series) data about disease propagation from surveillance and derived as forecast from simulations.

6.2.4 FluCaster Middleware Services

The FluCaster middleware is based on the service oriented architecture and REST API based Web services described in the Section 5.2.1. Using the known semantics of the epidemic forecasting problem, we have optimized the "stateless" REST APIs to support interactivity. For instance, in case of a Push API requesting infection information of a region at a particular level based on input parameters, the performance of the system is improved by using the knowledge of the region's geographical levels. In this case, the API passes back data on all the sub-regions within the region. The data is then cached on the UI web server so that rendering time on the UI (e.g. maps) is reduced considerably. For instance, when forecasting information is requested using a PULL API for the state of New York, then the aggregate infection information about every county within New York state is also passed back, so that rendering time on the UI is significantly reduced when the user zooms-in on the map.

In addition, we have also used optimizations on the relational query execution plan for the queries accessed through REST services for fast retrieval through use of indexing, caching and other mechanisms. The web server hosting the web services and the web server hosting the user interface are both optimized for quicker response times.

We have used caching on the web server side for repetitive queries to avoid expensive back-end requests and avoid delays. For instance, queries for bigger states, like California, are cached on the Web server to avoid further queries to the back end database for the same state, thus avoiding delays. This also ensures scalability of the application, such that multiple requests to the database are not necessary for repetitive queries.

6.3 EpiCaster

We have developed EpiCaster [35], an integrated Web application for situation assessment and forecasting of various epidemics, such as flu and Ebola, that are prevalent in different regions of the world. Using EpiCaster, users can assess the magnitude and severity of different epidemics at highly resolved spatio-temporal levels. EpiCaster provides time-varying heat maps and graphical plots to view trends in the disease dynamics. EpiCaster also allows users to visualize data gathered through surveillance mechanisms such as Google Flu Trends (GFT) and the World Health Organization. The forecasts provided by EpiCaster are generated using different epidemiological models, and the users can select the models through the interface to filter the corresponding forecasts. EpiCaster also allows the users to study epidemic propagation in the presence of a number of intervention strategies specific to certain diseases.

With the design and development of EpiCaster, our goal is to leverage large and diverse sets of epidemiological models for generating epidemic forecasts for various diseases in different regions of the world, and make it available to the public health decision makers through an intuitive Web-based system. EpiCaster is based on our integrated platform that is not only able to handle models for current sets of diseases, such as flu and Ebola, but is also flexible enough to incorporate models for new diseases as they appear. The system is also extendable to add new global regions for which infection data becomes available. Through the user interface of EpiCaster, users can select the region, disease, models of computation, intervention strategies, and the source of surveillance data used in the forecast generation process. Our system supports collection of surveillance data through several different sources, such as the CDC (Centers for Disease Control and Prevention), WHO (World Health Organization), Ministries of Health, social media, and so on. Our modeling implementation allows use of causal models with detailed information, such as behavioral adaptations by individuals of a population and effects of policy-level interventions on disease propagation.

6.3.1 Evolution from FluCaster to EpiCaster

The conceptual design of EpiCaster has been developed over the course of several years. The first version of the application was designed specifically for supporting situation assessment and forecasting of flu, and was called “FluCaster”. FluCaster was developed to support a variety of forecast models at the back end for computing and generating flu forecast results. We leveraged some of the initial research work in the domain of epidemiological models for supporting Influenza, such as work by Shaman et al. [88], Nsoesie et al. [73], Tamerius et al. [93], work on Google Flu Trends [50], and so on.

The initial version of FluCaster was designed using individual-based models of flu forecasting. We implemented individual-based models for different states of the US. This version used synthetic population modeling, together with surveillance information obtained from the CDC. In this version, FluCaster was developed to have detailed features, such as having a number of options available to the end users to select demographic parameters like age and gender for evaluating the flu effects on certain subpopulations. This version of FluCaster also supported interventions at a very granular level. This version was successfully used for a number of experimental studies requested by CDC for forecasting epidemic parameters such as peak number of infections, peak time, and start and end of the flu season.

Over time, new studies were requested by IARPA and other organizations to conduct flu forecasting for Latin American countries. Since detailed demographic and geographic information was unavailable for some of the Latin American countries, it was difficult to deploy individual-based models. However, we could develop aggregate ODE-based models for Latin American countries to compute aggregate forecasts without requiring fine-grained demographic information. Even though the model of execution was different, these models could provide similar data output for visualization at an aggregate level. Hence, we decided to integrate and incorporate ODE-based models into the FluCaster back end so that users could select them while evaluating results for available regions in Latin America.

The main motivation to extend FluCaster to other diseases arose from the requirement last year to forecast the propagation of Ebola in West African nations during one of the worst epidemics in the world. Computational scientists developed detailed models of Ebola epidemic propagation based on disease dynamics, such as infection through bodily fluids, possible infections through deceased individuals, and other modes of transmission. The NDSSL at Virginia Bioinformatics Institute was a pioneer in developing several forecasting models for predicting the propagation of Ebola in West Africa. IARPA (Intelligence Advanced Research Projects Activity) requested to view forecasting results to see how the Ebola epidemic might unfold in West Africa. Using the surveillance data gathered from the WHO and Ministries of health, we produced forecast results every week, by providing it as input to our forecasting models.

From this experience of producing forecast results for Ebola, we realized that the forecasting pipeline of Ebola would work similarly to that of flu from a data representation and systems standpoint, producing some aggregate form of forecast output. The modeling of the disease dynamics and the network of Ebola is vastly different from that of flu, since the propagation dynamics are different. However, if the forecasts of flu can be made available to end users through a Web-based platform, then the forecasts of Ebola could be done in a similar way. This led to the conceptualization and design of “EpiCaster”, for supporting a range of global epidemics.

There have been some systems published in the literature that produce epidemic nowcasts and forecasts over the Web such as the systems developed by Columbia University’s Mailman School of Public Health [1] and HealthMap [4]. The distinguishing feature of EpiCaster is that it is not tied to a single data source or forecasting model. EpiCaster can support a variety of forecasting models of epidemic propagation, including fine grained models such as agent based models (ABM models). Hence, EpiCaster allows a detailed study of global epidemic forecasts for a range of diseases. EpiCaster also produces forecasts in the presence of targeted interventions strategies applied to subpopulations of a region, thus allowing advanced predictive analytics.

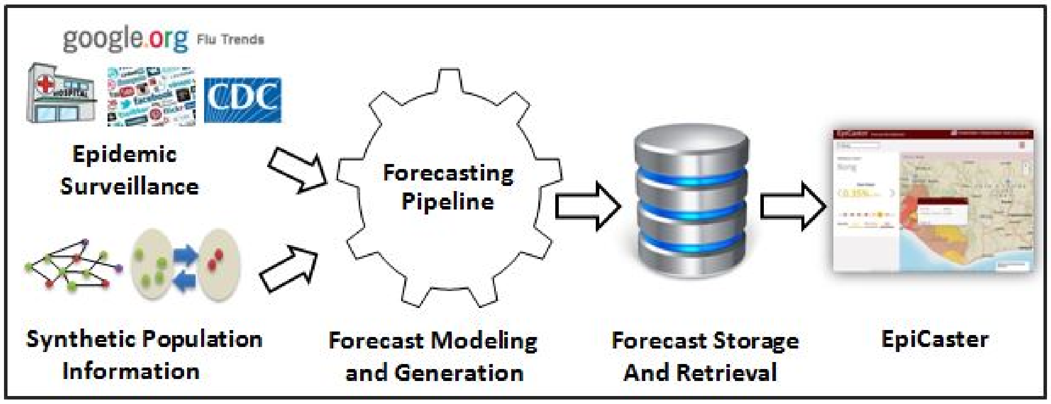

google.org Flu Trends
CDC
Epidemic Surveillance
Synthetic Population Information
Forecasting Pipeline
Forecast Modeling and Generation
Forecast Storage And Retrieval
EpiCaster

Figure 6.7: Conceptual architecture of EpiCaster from data collection to results display on the UI

6.3.2 Conceptual Architecture of EpiCaster

EpiCaster is built to support situation assessment and forecasting for a range of ongoing epidemics prevalent in the world that can also be viewed at multiple geographic, social and temporal resolutions. The architecture of EpiCaster has been designed to support scalability with respect to large volumes of data gathered from multiple surveillance sources and manipulated using epidemiological models to arrive at forecast results.

Figure 6.7 shows the conceptual architecture of EpiCaster. The backbone of EpiCaster's architecture can be described in four parts. First, information is gathered using various surveillance instruments, such as the CDC (Centers for Disease Control and Prevention), local hospitals, social media sources and so on. For a disease like Ebola, where surveillance information from affected countries in West Africa is very scarce, non-traditional surveillance methods can be used and incorporated into the system. For instance, for Ebola, we use information collected from the World Health Organization (WHO) and the Ministries of Health of the affected countries about current infection trends. EpiCaster builds on the premise that this collected information provides a good representation of the extent of the current epidemic. We use this information as input to the forecasting pipeline to produce forecast results. In addition, EpiCaster uses synthetic contact networks as the basis of population demographics of different regions and as a fabric of disease transmission, especially for individual-based epidemic models. These social contact networks are derived for each region using Census data as described in the research by Barrett et al [11].

The second part of the conceptual architecture involves forecast modeling and generation. We use complex epidemiological models, including individual-based models and aggregate models, as implemented using multi-agent simulations as the key components of the prediction pipeline.

The third component of the EpiCaster architecture involves storing the forecast output in a central data repository so that it can be retrieved in an effective way. The data store needs to handle the flexibility of querying the database to retrieve forecasts based on any combination of regions, diseases, interventions, models and data sources, while still supporting scalability. Hence, the data representation schema is critical.

Lastly, the forecast data retrieved from the data store needs to be passed back to the user interface (UI) so that it can be rendered on the user's web browser. The middleware that we have developed is able to handle flexible inputs from the users, including disease, region, models, data sources, and interventions. It also handles large scale data transfers to visualize current epidemic and forecast data on heat maps and plots.

The entire backend of EpiCaster is based on top of the integrated epidemic simulation environment that we have explained before including the data architecture and the multi-level Web services for flexible retrieval of forecasts on various diseases.

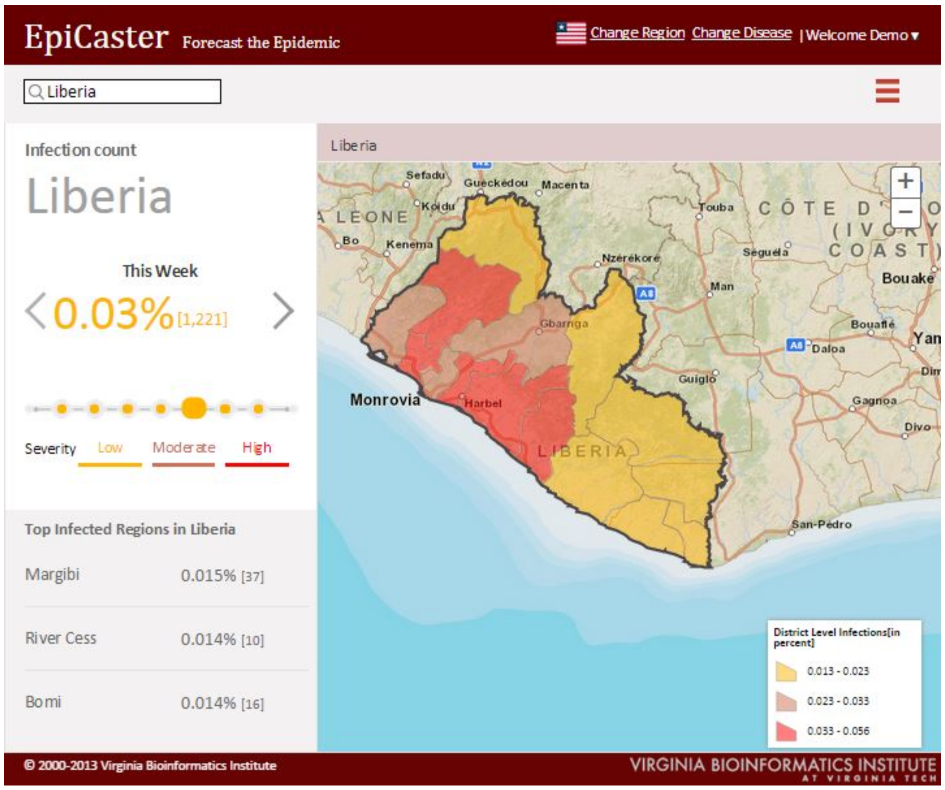
6.3.3 EpiCaster User Interface features

EpiCaster's Web-enabled User Interface allows users to select the disease for which the user seeks outbreak information; in the current version of the application, users can choose either flu or Ebola, but this is extendable. Once the disease is selected, EpiCaster displays all the regions where the disease outbreak is prevalent on a map of the world. The user can then select a particular region on the map to get a fine-grained assessment of disease activity as well as forecasts within that region. Figure 6.8 shows a snapshot of EpiCaster displaying infection counts for Ebola in Bong county of Liberia.

EpiCaster's Web-enabled User Interface consists of the following main components.

- Interactive Timeline: EpiCaster provides a timeline covering a range of weeks over which a user may view epidemic information. The available timeline options include the current week and any of the last four weeks for situation assessment; users can also select the timeline for up to two weeks in the future to view forecasts.

- Interactive heat map: EpiCaster provides an epidemic heat map that shows infection levels aggregated at multiple geographical levels, such as state, county and blockgroup. GIS data is used to provide interactive maps to improve data visualization. Once the timeline is selected, EpiCaster displays epidemic activity in the selected region. The interactive heat map is displayed with different color schemes to allow the user to analyze the severity of the epidemic in the region. The user can also drill down to each subregion of the map to view infections at a granular level. The varying number of geographical levels and naming conventions in different regions of the world makes this visualization complex. For instance, the geographical levels in US are divided into state, county and block group, whereas in a country like Liberia the geographical levels include only counties. Also, the size of the population in the region makes the computation complex. For instance, the average size of a United States Census block group is about 4,000 people. To retrieve and display the forecasting results of each block group, in each county, in each state of a nation is a challenging task. But us-



Top Infected Regions in Liberia

Margibi	0.015% [37]
River Cess	0.014% [10]
Bomi	0.014% [16]

Figure 6.8: Snapshot of EpiCaster as a standalone web application showing the top infected regions in Liberia and the infection counts and percentages in those regions

ing our optimization techniques, EpiCaster supports drilling down on the map to view such fine-grained forecast results.

- Plots (Epicurves): EpiCaster allows the users to view Epicurves showing infection trends in a region over a time period, as well as the peak infection count in the specified time period. Users can select the plots to be viewed for the past and current weeks based on surveillance sources such as GFT and CDC, and view forecast plots derived from a range of epidemic models for the future weeks.

6.3.4 Data Flow within EpiCaster

Figure 6.9 shows the flow of data within EpiCaster from surveillance to forecast representation. As seen in the figure, information about current infection trends is gathered from multiple surveillance sources such as the CDC, WHO etc. based on the disease type and epidemic information availability. This information is stored in various formats such as csv or text files. We have a module that converts this data into relational format and stores into the database.

The comparison module from the forecast pipeline, as explained before, matches the time-series

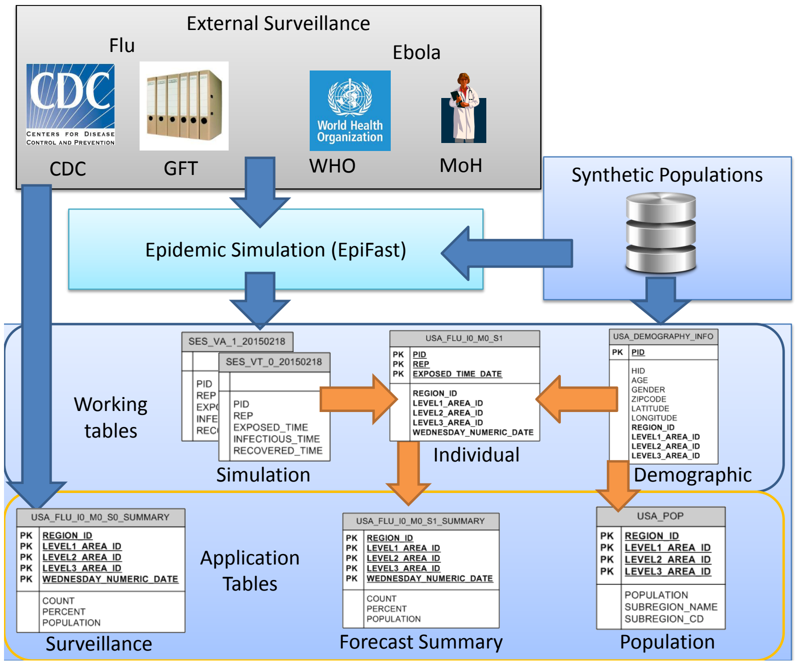


Figure 6.9: Data flow within EpiCaster

curve from surveillance with the precomputed model library. The pipeline is executed based on whether the match is found or not found and the forecast results are stored in the database. The synthetic population information is also stored in the same database. We use a bunch of working tables for data manipulations and transformations.

The final forecast outcomes are stored in aggregate format in forecast summary tables, from which the forecast can be retrieved, for each region, for each disease, with or without interventions.

The EpiCaster UI web server calls the appropriate EpiCaster APIs which in turn pick up the data related to epidemics from the correct tables based on the input parameters and the forecast is displayed on the UI in the form of heat maps and epicurves as explained in the Section 6.3.3.

Chapter 7

Performance Evaluation

In this section we present the performance evaluation of the system with respect to epidemic intervention strategies and forecasting. We first present the performance of intervention evaluation using DISIMS through two computational experiments. Next, we present the performance of various computational experiments performed using the web service APIs designed for epidemic forecasting, which form the backbone of EpiCaster. Lastly, we present a case-study on adding a new region, disease and intervention to the system and the total time taken for an end-to-end experimental analysis using our approach.

7.1 Experimental Evaluation of DISIMS

In this section we illustrate the capability of DISIMS with two computational experiments. The first one is a real world case study to evaluate the effectiveness of *school closure* intervention in containing an ongoing epidemic with DISIMS. The second one is to illustrate online optimization of intervention strategies along a decision tree via interactions with a DISIMS-run simulation.

7.1.1 Computational Experiment 1

In this computational experiment, we evaluated the school closure intervention strategy applied during an epidemic of catastrophic flu in a region. School closure has been deemed as an effective measure to contain a flu pandemic. For example, during the 2009 H1N1 flu outbreak, New York city officials ordered the closure of 30 schools "after an increase of reports of students with flu-like symptoms" [32]. School closure reduces the overall transmission within a school and is a well known and effective non-pharmaceutical intervention [27, 98].

For a school closure intervention, the potential risk of within-school flu outbreak if schools remain open has to be evaluated against the large social costs associated with school closures. The decision

System Name	**Development time**	**Analysis set-up time**	**Expt. set-up+analysis time**	**Expt. execution time**
EPIFAST	hard to implement	unknown	unknown	unknown
INDEMICS	1 day	0.5 day	1.5 days	3-4 min/iteration
DISIMS	5 minutes	20-25 mins	30 min	3-4 min/iteration*

Table 7.1: Comparison of efforts for school closure intervention case study on Miami city. Experiment execution is carried out over multiple iteration days. *For DISIMS, the expt. execution time excludes communication time between the web-server and INDEMICS Middleware Platform, which is negligible compared to actual expt. execution

to close schools is based on such evaluation. For example, the Centers for Disease Control and Prevention (CDC) revised its earlier recommendation of shutting down schools immediately when a few students became ill to keeping schools open even with flu outbreaks during the later period of the 2009 H1N1 pandemic [94].

The school closure policy works as follows. If in a school, the fraction of students diagnosed with flu exceeds a certain threshold then the school is closed for a certain number of days, and for each diagnosed student below a certain age, one parent or care giver must stay at home. School closure is not a binary decision, but based on a number of parameters. We analyzed an event-triggered school closure policy for an experimental study on a flu outbreak in Miami, to assist analysts in their decision making in the real world. We considered two parameters in this measure: threshold and duration. The former determines the severity of the epidemic to make it necessary to close schools; the latter determines how long schools need to be closed. We chose two values for each of the two parameters to form four configurations of the school closure intervention. In contrast to previous studies where we examined the course of epidemic dynamics, the main objective of this study was to study and provide an effective comparison between a variety of settings of an intervention policy. We point out that the study provides an illustration of possible interventions that can be easily simulated by DISIMS but difficult for either EPIFAST or INDEMICS.

The complexity of the school closure intervention comes from two aspects: each school is determined to be closed individually instead of universal closures; for the affected subpopulation (students of the closed school) we need to identify another subpopulation consisting of people of appropriate demographic properties (age and household), who represent the care givers of the children.

When we were requested to perform the school closure study, our simulation engines such as EPISIMDEMICS and EPIFAST did not have features to integrate supplemental information at that time and hence could only simulate simplified but less realistic versions of the school closure intervention. The code of these high performance simulation engines would have had to be modified to support such interventions. Our simulation engine developers and the intervention experiment strategy designers would have had to work together to precisely interpret the strategies and code them into the high performance engines. The estimated development time including requirements gathering, implementation, and testing would be several weeks, in contrast to the estimated experiment execution time of only one week. This approach would have been time-consuming and it would have been difficult to report simulation results and make policy recommendations in time.

To overcome this problem, we used INDEMICS, a database supported epidemic simulation framework, to run the study. In contrast to the epidemic simulation engine like EPIFAST, the implementation of interventions in INDEMICS is modeled by data query algebra, and the interventions are completely computed using the INDEMICS Query Language (IQL), as described before. Experiment strategy designers only need to describe their scenarios in IQL and submit the simulation jobs to INDEMICS for execution. The experiment development process of INDEMICS takes a few days to map the interventions into IQL. INDEMICS incurs marginal execution time overhead, but it needs no significant code development or testing for the HPC simulation engine. We adopted this solution to run the study and it greatly reduced the study period and saved a significant human effort.

Although the development time for implementing the intervention was shortened remarkably by INDEMICS, INDEMICS did not have a module to automatically set up experiments, monitor the state of an experiment and manage experimental inputs and results. There was no provision for re-usability and sharing by checking if an appropriate INDEMICS intervention script was previously written by some other user. Also, when the intervention had to be simulated with different parameter settings, using INDEMICS became cumbersome. For example, a script to run a factorial experiment by changing multiple parameter values had to be prepared manually, which was error-prone. The simulation inputs and outputs had to be well organized to avoid overwriting or misreading. The simulation jobs also had to be monitored by the experiment executors. Such tasks needed considerable manual effort. Reading and understanding raw simulation results was difficult since INDEMICS did not have statistical analysis or data visualization modules. We realized that when the user has minimum knowledge and experience in preparing INDEMICS scripts (in IQL) and running simulations in a high-performance computing environment, which is often true for public health domain experts, executing complex experimental studies such as the one to implement school closure intervention, is difficult even with INDEMICS.

From the experience of implementing the School closure intervention study, we realized that the usability of the simulation system had to be leveraged further. Hence we developed DISIMS, with features like user interactivity, simple interface, experiment data management, job monitoring and analysis in addition to the attributes that were already provided by INDEMICS and EPIFAST. Employing DISIMS for the school closure intervention study could have reduced the overall experiment set-up and management time and enhanced the human productivity considerably.

Using DISIMS, the users now only need to select the intervention scripts and parameters of execution using a intuitive web-based graphical interface. The data files for the factorial experiments are

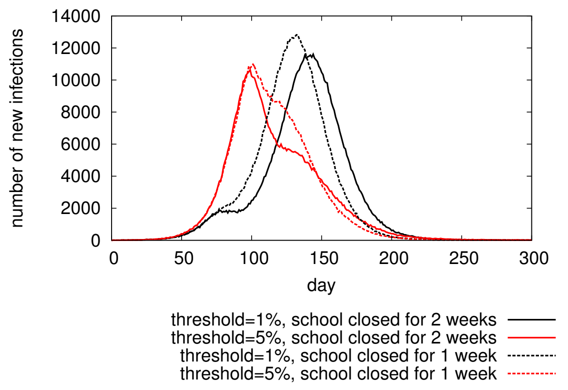


Figure 7.1: Example of DISimS generated plots: epicurves in a 2 by 2 experiment with different parameter settings of *school closure* intervention in Miami during a catastrophic flu.

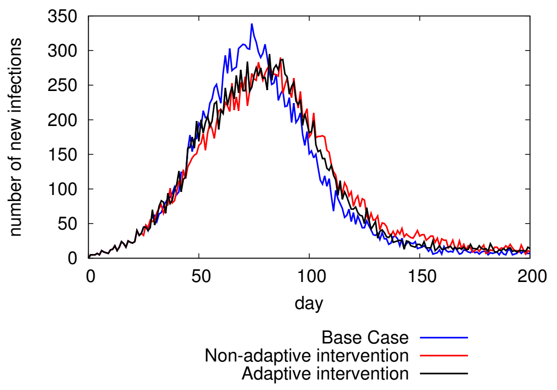


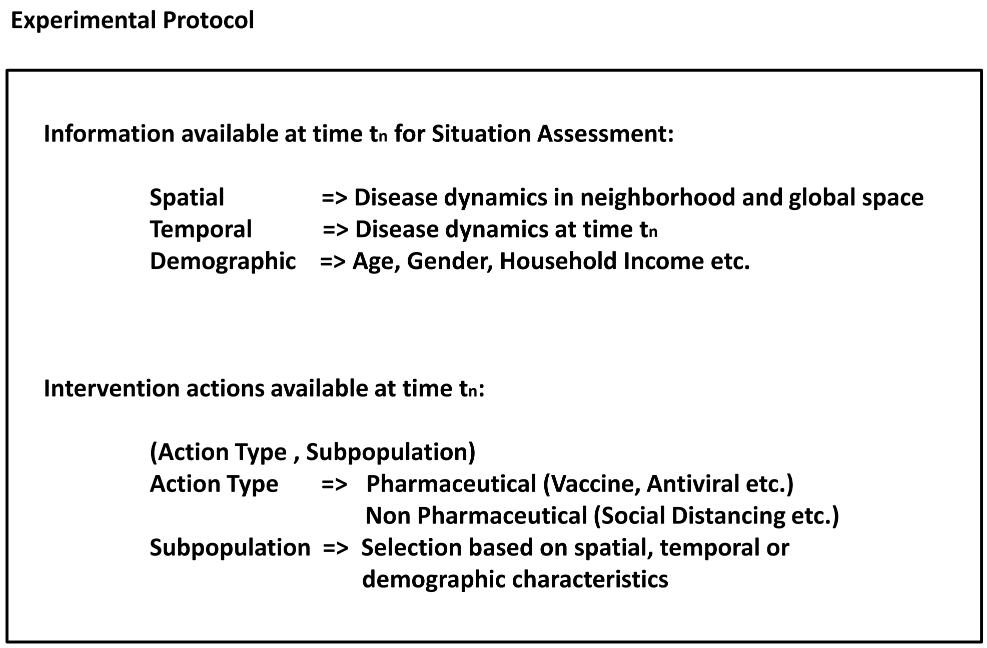
Figure 7.2: DISIMS supported optimization study: Comparison of epidemic curves obtained in non-adaptive vs. adaptive cases.

well-organized and well-archived and the simulation jobs are automatically monitored and scheduled. DISIMS introduces a marginal overhead of execution as compared to INDEMICS, which is equivalent to the communication time between the web-based front end to the INDEMICS server middleware. Table 7.1 shows the comparison of the efforts for the school closure intervention experiments using EPIFAST, INDEMICS and DISIMS on the city of Miami. As can be seen in the table, the total human effort for experiment design and analysis is reduced significantly by DISIMS compared to previous systems, and the total increase in the experiment execution time is negligible. This table shows the value of DISIMS for improving the productivity of epidemiologists and public policy decision makers. They can now set up, manage, and execute complex intervention case studies without much help from the computational scientists.

Figure 7.1 shows epicurves in different intervention settings. It is an example of visualization that user can obtain directly from the DISIMS system. In this plot, we can see that applying school closure too early (with 1% threshold) may suppress the disease outbreak temporarily but the epidemic takes off soon. But it indeed postpones the epidemic peak; and the gained time may be useful for taking other measures. Analysts are able to conduct such in-depth analysis using DISIMS and make decisions regarding the epidemic mitigation strategies to apply in real world.

DISIMS has been in use by the analysts in our lab to execute a number of simulation experiments with a variety of intervention strategies to contain epidemics. The analysts have reported that they are able to execute intervention strategies using DISIMS that were previously not possible to be executed within a stipulated time range. The analysts have also reported that DISIMS provides much greater capabilities in terms of analysis and range of experiments, that were not possible using any of the previous systems.

For instance, using DISIMS analysts can write Intervention script templates for new types of



Experimental Protocol

Information available at time t_n for Situation Assessment:

Spatial => Disease dynamics in neighborhood and global space
Temporal => Disease dynamics at time t_n
Demographic => Age, Gender, Household Income etc.

Intervention actions available at time t_n:

(Action Type , Subpopulation)
Action Type => Pharmaceutical (Vaccine, Antiviral etc.)
Non Pharmaceutical (Social Distancing etc.)
Subpopulation => Selection based on spatial, temporal or demographic characteristics

Figure 7.3: Experimental protocol of the information available to a public health analyst for situation assessment at any time-step when the simulation is paused and the range of options available for intervention application for the remainder of the simulation.

interventions and quickly incorporate them as part of the system. Other analysts can reuse the same scripts with a wide range of parameters such as number of replicates and different regions, and perform a factorial study design with increased accuracy. Such study was not possible before without several changes and manual analysis. DISIMS makes it easier to conduct complex experimental studies on high-end computational resources in a timely manner with graphical analytical results. DISIMS also frees the analysts from the burden of understanding details of the computing infrastructure or the need to write complicated simulation scripts.

7.1.2 Computational Experiment 2

In this computational experiment, we demonstrate the utility of DISIMS to train public health decision makers in evaluating the effectiveness of various intervention strategies. DISIMS provides a specific type of interactivity while executing realistic large scale simulations. This interactivity allows users to pause a simulation after a specific duration, analyze the evolution of the epidemic in the current scenario and then come up with a new strategy that can work best for containing the

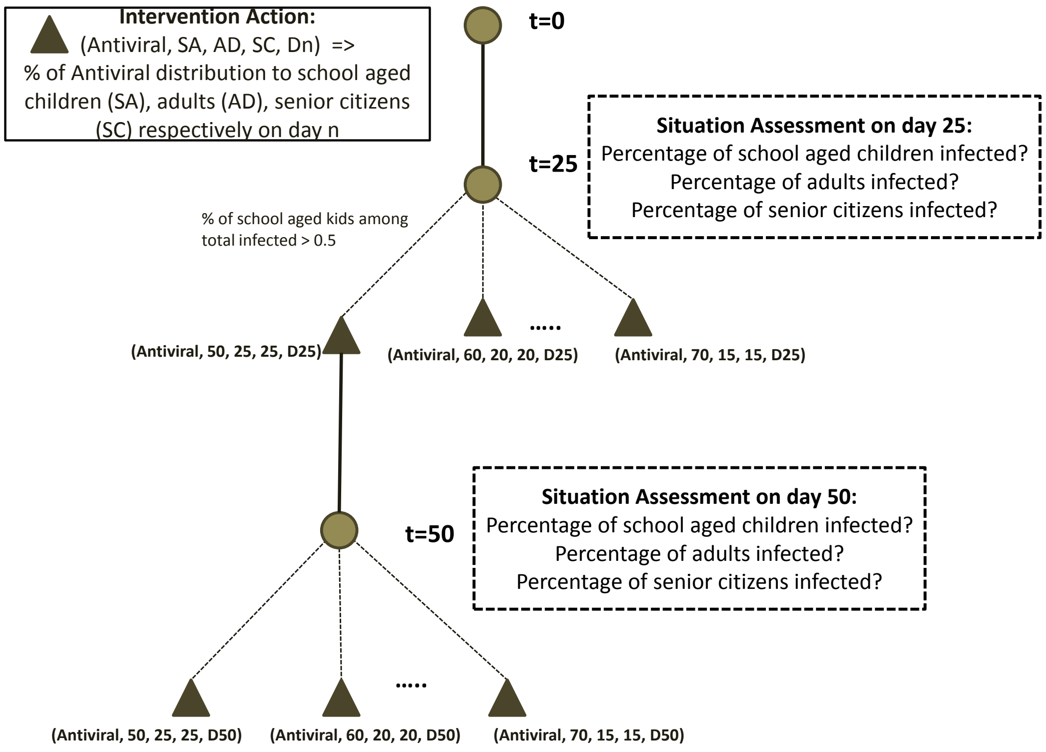

Intervention Action:
(Antiviral, SA, AD, SC, Dn) =>
% of Antiviral distribution to school aged children (SA), adults (AD), senior citizens (SC) respectively on day n
t=0
t=25
Situation Assessment on day 25:
Percentage of school aged children infected?
Percentage of adults infected?
Percentage of senior citizens infected?
% of school aged kids among total infected > 0.5
(Antiviral, 50, 25, 25, D25)
(Antiviral, 60, 20, 20, D25)
(Antiviral, 70, 15, 15, D25)
t=50
Situation Assessment on day 50:
Percentage of school aged children infected?
Percentage of adults infected?
Percentage of senior citizens infected?
(Antiviral, 50, 25, 25, D50)
(Antiviral, 60, 20, 20, D50)
(Antiviral, 70, 15, 15, D50)

Figure 7.4: Decision tree representation of the choices available to a public health analyst for applying interventions based on situation assessment. The analyst can pause the simulation periodically, analyze the situation and take actions that affect the course of the epidemic.

epidemic in the given scenario.

At every time-step at which the simulation is paused, the analyst has to make a decision about which intervention strategy to apply for the next duration ranging across several time-steps, so that the epidemic can be contained effectively. This decision is based on situation assessment of the epidemic dynamics at the current time-step, which is made possible because of the ISSAE component of DISIMS. The various choices available to the analyst for decision making and the actions taken by the analyst (that could affect availability of future choices), can be represented in the form of a decision tree. This decision tree is based on the experimental protocol of information available to users and the range of intervention actions that can be executed. The experimental protocol of an adaptive epidemiological experiment is shown in Figure 7.3.

Experiment Design. We have designed a computational experiment for training purposes to study a case of strong flu epidemic in the Montgomery County of Virginia. The county has a population of about 75,000 people. The public health decision maker has antivirals to distribute but the supply is limited. Starting from the beginning of the epidemic, every 25 days, 1,000 more units of antivirals become available. This is to address the limited pharmaceutical manufacturing capacities and latency of massive distribution of drugs. The antivirals are effective for 14 days once applied and they reduce probability of getting infected (for healthy people) or probability of infecting others (for ill people) by 80%.

A group of experts are chosen to participate in the experiment. Each expert can ask DISIMS questions about the current epidemic dynamics and can decide how to distribute the available antiviral supply based on the answers to the questions. For illustration, we pause the simulation every 25 time-steps (25 simulation days) to allow the experts to query and intervene based on age groups and current health state of each person in the Montgomery County. We point out, however, that the simulation can be paused at any time-step and the user interaction can be based on any data available from the simulation or in the database.

A public health analyst, at each decision point, can decide to do nothing, or apply K units of antivirals, where K is bounded by the current antiviral supply, to people randomly chosen in certain age groups. The possible decisions and the random epidemic trajectory formulates a stochastic decision tree, where at each decision point an analyst may decide to take different branches depending on the epidemic dynamics.

We ran this scenario with a public health analyst. Figure 7.4 shows the representation of the decision tree available to the analyst at the various decision points that he paused the simulation. In this case, the analyst paused the simulation on day 25 and day 50 respectively. The figure also shows the corresponding choices available for intervention application. For the sake of simplicity of the experiment, we made only limited choices available to the analyst for situation assessment and intervention application based on age groups. However, in reality the analyst can choose from a large range of information for situation assessment and intervention application, based on the Experiment protocol described in Figure 7.3.

Figure 7.2 shows epidemic curves generated from the DISIMS system for the experiment. In the non-Adaptive case, antivirals are evenly distributed among each age group - School age, Adults and Senior Citizens on day 25 and day 50. In the adaptive case, antivirals are distributed based on situation assessment on day 25 and day 50. Since it was observed by the analyst that higher percentage of school children were infected in the first few days compared to other age groups, greater percentage of AVs were distributed to school children. As can be seen in the Figure 7.2, the peak time of the epidemic shifted to the right in the non-adaptive and adaptive cases and the peak infection count also went down. There is not much difference observed in the epi curves plotted for non adaptive vs. adaptive cases, since there is only a small difference in the percentage distribution of antivirals in both cases. Hence, this may not cause a significant difference to the epidemic dynamics. However, if the adaptive technique is used for training the analysts such that they can infer more about the epidemic dynamics and apply intervention actions at the right decision points, then there is a possibility of seeing change in propagation dynamics of the epidemic.

This experiment illustrates the following capabilities of the DISIMS system: (i) A user can pause and resume a simulation at any time-step. (ii) A user can interact with the simulation online, query the system and make decisions accordingly. DISIMS enables adaptive interventions that address both uncertainty of epidemics and that of the human decision process. (iii) DISIMS supports

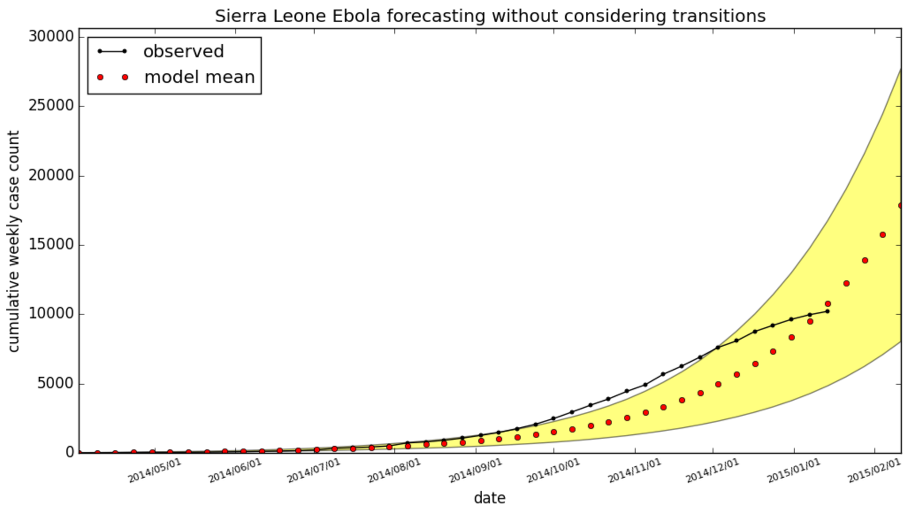


Figure 7.5: Forecasting of Ebola in Sierra Leone without considering a change in the transmissibility observed through surveillance

realistic scenarios of public health level decision making. (iv) DISIMS supports simulations of realistic, implementable but complex intervention strategies, based on dynamic epidemic data, as well as demographic and other data.

7.2 Experimental Evaluation of EpiCaster

We have used EpiCaster for performing several experimental case studies to evaluate the extent of ongoing epidemics in different regions of the world as well as to forecast future propagation. Initially, we started with using the Flucaster version for situation assessment and forecasting of flu in different states of USA, particularly to be able to come up with appropriate intervention strategies to contain the epidemic before the peak epidemic season.

One of the most notable experimental studies that we have used EpiCaster for recently is forecasting of the recent Ebola outbreak in West Africa. We explain one of the experimental study for Ebola that we executed using EpiCaster.

7.2.1 Computational Experimental Study Using EpiCaster for Ebola Forecasting

In the 2014-2015 Ebola pandemic, it was observed that in mid-October 2014 and mid-December 2014 there were significant changes on the transmissibility of the disease. Given the flexibility of our modeling design in EpiCaster, we were able to incorporate this epidemic parameter into the

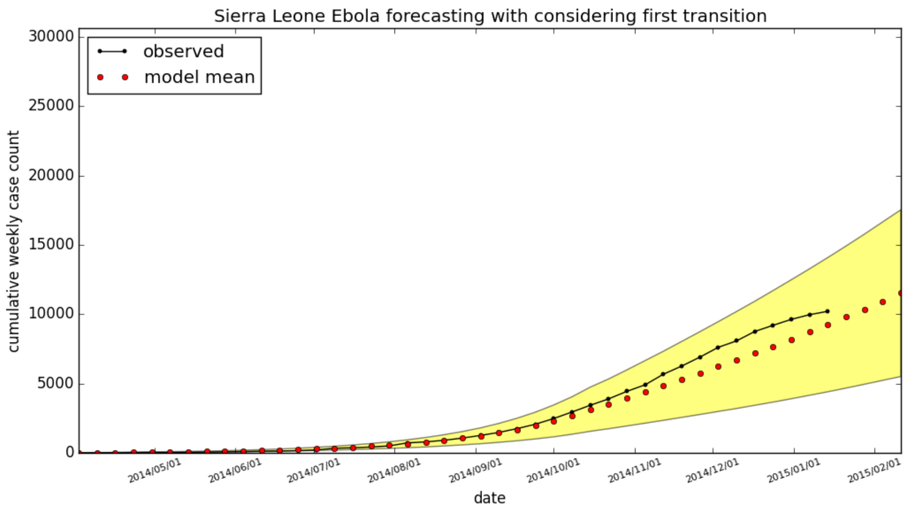


Figure 7.6: Forecasting of Ebola in Sierra Leone considering only a single transition in transmissibility observed through surveillance

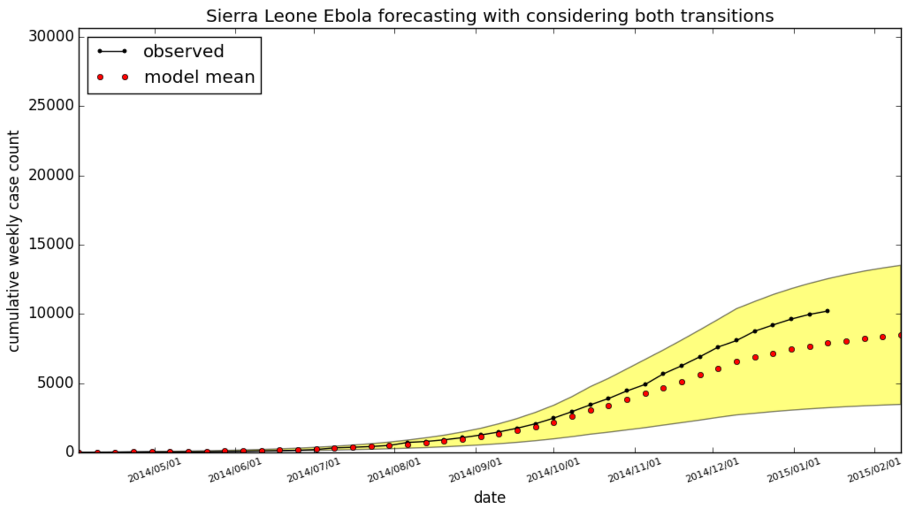


Figure 7.7: Forecasting of Ebola in Sierra Leone considering both transitions in the transmissibility observed through surveillance

system design and re-calibrate our model to study the effects of the pandemic propagation, taking into account this change.

We ran the EpiCaster forecasting pipeline for Sierra Leone in West Africa with and without the changes in transmissibility parameter and gathered the forecast results for analysis. We present our forecast results as follows in three plots: (i) in Figure 7.5 we made the prediction without considering either of the two transitions, so the transition parameters are not considered in the model searching; (ii) Figure 7.6 is the prediction made with the only mid-October transition being considered in the model searching; (iii) and Figure 7.7 is the prediction made with both transitions being taken into the account in the forecasting simulation.

As can be seen from the Figure 7.5, when the transition parameters are not considered the model tends to over-estimate the number of infections in the forecast period. The number of infections computed by the model tend to be much higher than the observed number of infections. Figure 7.6 shows that the model slightly underestimates the number of infections in the forecast period, when a single transition is considered in the simulation. The curve in Figure 7.7 shows that when the model takes into account both transitions in the forecasting simulation, the observed infection count and model infection count are very close. This shows that when the most updated information is available from the environment, the model outcome should be re-calibrated and the forecast outcome from EpiCaster can be improved to mirror the actual observed infections.

Chapter 8

Ongoing Work on Forecasting

In the previous chapters, we have explained the various components of our integrated analytics environment for evaluation and forecasting of different epidemics across multiple geographical regions. We have also presented the architectural details on how the environment has been made available as a platform-as-a-service offering through use of flexible data representation techniques and use of REST based web services.

The interesting feature of our platform is that it supports fine-grained assessment of epidemics by supporting execution of agent-based models in addition to aggregate-based models. This is made possible by the availability of fine grained information at an individual level by using synthetic information systems, built using census data to represent every unique individual of a region along with his/her activity patterns. This is one of the key factors of our analytical platform that allows computation of not just aggregate forecast counts of infections, but also fine-grained spatial forecasts at an individual level. This also allows analysis of forecasts based on demographic and behavioral characteristics. Some of the other factors include the ability to gather surveillance data from different surveillance sources, related to different types of epidemics, and use it within the forecast simulations. Moreover, both policy and behavioral interventions can be represented using our system, so that forecasting results can be computed in presence of mitigating factors.

Forecasting methods described in the literature for forecasting of infectious diseases like flu have focused on estimation of epidemic measures such as peak timing and intensity, peak height, attack rate, epidemic duration etc. Analyzing these epidemic measures is important for forecasting future trends of an epidemic so that appropriate control measures can be undertaken by public health decision makers. However, in addition to these epidemic measures, it is also essential to evaluate epidemics at a fine grained level including analysis of spatial, temporal, demographic and behavioral patterns. This involves analyzing demographic patterns of disease propagation such as prevalence of an epidemic among particular age groups or societies, analysis of infection vulnerability of individuals in a particular sub-region etc. Such level of analysis can facilitate implementation of targeted intervention strategies focused on particular demographic groups or individuals that are more susceptible to infections. Moreover, application of mitigation strategies through policy-based and individual-based interventions have shown to modify epidemic propagation dynamics. These may include individual level interventions such as the use of face masks, anti-viral and so on or policy level interventions such as regional school closures. It is essential to incorporate the effects of intervention strategies on propagation, while modeling epidemic forecasts.

There is not much published literature in the area of individual-level epidemic forecasting since research in this field is still at a very nascent stage. In this thesis, we propose a preliminary approach for fine-grained individual-level epidemic forecasting using concepts from machine learning. Individual-level epidemic forecasting in the context of this thesis refers to finding a probabilistic score that represents the vulnerability of an individual towards getting infected by an ongoing epidemic.

The techniques described here can be developed further to improve the quality and performance of the forecast computations. In this Chapter, we will focus on individual-level epidemic forecasting of flu outbreaks in different regions of the US, using fine-grained demographic, social and infection information gathered about individuals. The computational techniques described here can be extended for studying other epidemics as well.

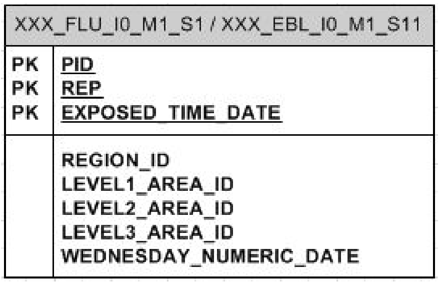
8.1 Individual-level Infection Computation

Evaluation of fine-grained epidemic forecast involves computation of different forecast measures based on various social, demographic, geographic and individual-level behavioral filters. Some examples of the measures involved in the fine-grained analysis include a) the number of infected individuals of a particular demographic (based on age-groups, gender, income-level and so on), b) the number of infected individuals belonging to a particular geographic locality (such as a county or a block group), and c) the number of infected individuals with certain common behavioral attributes (such as those using face masks, or those who may be vaccinated).

Modeling of an epidemic at a fine-grained level is a complex process, given the scale of data involved. Computation of epidemic forecasts at an individual level requires detailed information about all the individuals in a given region, along with their social, demographic and geographic attributes.

We have leveraged the use of synthetic information systems along with agent-based models of regional forecast computations, as described in the previous chapters, for building a fine-grained individual-level forecasting model. In our model, every individual of a population is represented as an agent in the epidemic simulation process.

The output of the epidemic forecasting process using agent-based simulations is a list of identifiers of individuals in a region who are exposed to the disease at a given time step of the simulation, which is typically represented as a day. This forecast output can also be aggregated to represent the total count of infected individuals at a particular regional level. Figure 8.1 shows the structure of a relational table that is populated by an agent-based flu forecasting model, executed for a particular



XXX_FLU_I0_M1_S1 / XXX_EBL_I0_M1_S11	
PK	PID
PK	REP
PK	EXPOSED_TIME_DATE
	REGION_ID
	LEVEL1_AREA_ID
	LEVEL2_AREA_ID
	LEVEL3_AREA_ID
	WEDNESDAY_NUMERIC_DATE

Figure 8.1: Structure of the individual level infection table created as an output of an ABM simulation executed using EpiFast simulation engine. For forecasting of flu across different regions in the United States, this data is generated for each of the 50 states within US, with level1 and level2 regions representing county and block group respectively.

region of the USA.

The forecasting output is typically produced every week, for every region (state) of the USA, once surveillance data is available for computation. As can be seen in the table, the forecast computation is fine-grained producing forecast counts at multiple geographical levels, such as county(level1), block group(level2) etc.

For fine grained forecasting, we use this data gathered from epidemic simulations together with data obtained from synthetic information systems. This includes individual level demographic and behavioral data about individuals and the synthetic social contact network representing contact patterns.

When a user requests his/her individual level forecast, or a measure of the vulnerability to an ongoing epidemic, the user needs to provide accurate information on his/her demographic, health and behavior attributes. The accuracy of the forecast computed by the algorithm is dependent on the validity of the information provided by the user. In our approach, we assume that the user provides up-to-date information about his/her demographics and health state before requesting a personal forecast for an ongoing epidemic and hence this information is available for computation.

Given this assumption, the individual level disease prediction problem can be divided into the following sub-problems:

- Finding matching individual/s in the synthetic information system, and
- Generating Individual-level epidemic forecast

Individual level epidemic forecast needs to be computed in real-time when a user requests for it. However, finding a matching individual in the synthetic information system is a complex process that is dependent on several factors. The matching process involves comparisons of the individual's demographic and social features with that of the entire synthetic population. Hence the process is both data and compute intensive. Considering the scale of data involved in the computations, we have devised an algorithm to speed-up the individual mapping process in the synthetic information system, so that user's infection vulnerability can be computed in near real-time. This process involves some data aggregations and pre-computations. We describe our approach below.

8.1.1 Finding Matching Individual/s in the Synthetic Information System

To forecast a user's vulnerability to ongoing epidemics, it is first essential to get information about the health, behavioral and social characteristics of the individual. We have created a synthetic representation of the entire population of the United States, such that the synthetic representation closely mirrors the actual characteristics of all individuals of the regions, without revealing their identities. We use this synthetic information system to find a matching individual or individuals that most closely mirrors the requesting user's demographic, geographic, social, health and behavioral characteristics in the synthetic population. The scale of this matching problem is dependent on the scale of the population under consideration. If more granular regional information is available, then the number of matching computations can be limited by the size of the regional population. For instance, if the user provides information that s/he resides in the Montgomery county of Virginia, instead of just stating that s/he resides in the state of Virginia, then the search space for comparison of user's characteristics with that of the synthetic population reduces from 8 million people to 100K individuals.

Given this assumption, the number of comparisons for matching the individual to an individual/s in the synthetic population, is equal to the population of the region that the individual belongs to. This matching can be an expensive operation especially for populations of densely populated regions like California and New York. In addition, the user may provide information about demographic characteristics such as age, gender, household income and so on, which can be used as features for matching with the synthetic population.

If there are m features provided by the user and s/he belongs to a region with a population of n individuals then the size of the matching problem for individual mapping in the synthetic population is in the order of $O(n \times m)$.

Computing this matching in near real-time is an expensive operation if user's geographic and demographic information is not available at a granular level. To make the matching process more efficient, we introduce pre-computations for improving the speed of the matching process. In particular, we propose using machine learning techniques for executing pre-computations on the synthetic information systems to increase the execution speed of the matching problem.

Clustering Algorithms for Mapping Individuals in the Synthetic Population

We analyzed the use of different machine learning techniques for applying precomputations on the synthetic information systems. Since the process is unsupervised, we evaluated using clustering algorithms to cluster sub-populations in a given region with similar features together. The main motivation behind doing this is to reduce the number of initial comparisons from the size of the entire population to the number of clusters. This process is an offline process and the results of the precomputation are stored in the system before any user can seek individual level forecast. We propose using clustering on every region under consideration (at the state-level for the United States) and using individual health, social and behavioral characteristics as features or dimensions of the clustering algorithm.

The clustering algorithm can be implemented using several types of clustering algorithms including k-means clustering, hierarchical clustering and so on. We have currently implemented the k-means clustering algorithm for the pre-computation process that classifies a population into k clusters. Our approach can however be implemented using any other clustering algorithms as well.

The features that we have used to cluster the population of a region can be broadly classified in the following categories:

- Geographical - These features are related to the geographical location of individuals. Examples include county, block group, zipcode etc.
- Demographic - These features are related to demographic characteristics of individuals. Examples include age, gender, household income etc.
- Social contact network - These features represent the characteristics related to the contact pattern between individuals such as number of proximity contacts, duration of contact etc.
- Health state - These features are related to the health state of the individuals of a population that directly affect the propagation of an infectious disease. Examples include state of being exposed to an epidemic, state of recovery from a disease etc.
- Behavioral - These features are related to the behavioral adaptations by individuals of a population in response to an epidemic. Examples include individual mitigation features such as the use of face masks etc.

The values of the above parameters/features can be either numerical or categorical. The numerical parameters are normalized before they are used in the clustering algorithm, while the categorical features are converted to nominal values, i.e. the value is set on the presence of the feature and not set in its absence. The approach we have used, of using binary values for categorical features in k-means clustering was first introduced by Ralambondrainy et al. [83]. For instance, absence of face mask can be represented by a nominal value 0, whereas presence of a face mask can be represented as a nominal value 1. Huang et al. [55] have described two different extensions to the k-mean algorithm, called the k-modes algorithm and the k-prototype algorithm, for clustering large data sets with categorical values in combination with numerical values. These algorithms can be used for clustering instead of the traditional k-means algorithm that we have used in our implementation.

There are 2 parts to our approach. One is the offline computation which involves creating clusters, and second is the online computation involving finding the closest cluster, based on the information provided by the user.

For the first step, we propose creating clusters of every regional population within the US for which epidemic forecasting is to be computed using the above features. Our current implementation, using k-means clustering algorithm creates k clusters with k centroids. These k centroids are the best representatives of each of the points withing the cluster. As an illustration, we have applied k-means clustering on the synthetic population derived for Montgomery county in Virginia.

For the second step, we gather information from the users on health and other parameters including social and behavioral characteristics, that can be converted to the feature-set as described above. After the information is gathered, the next step is to compute the matching cluster that the user belongs to, based on his/her characteristics. This is done using a distance metric. We have used simple Euclidean distance as the distance metric for computation after normalizing the numerical features and converting categorical features into binary values. However, other distance metric functions can also be used when the data is a mix of categorical and numerical features such as those explained by McCane et al. [65] in the context of classification and principal components analysis.

The Euclidean distance between two points X and Y is given by $d(X, Y) = \sqrt{\sum (x_i - y_i)^2}$, where X=$(x_1, x_2, x_3 ... x_r)$, Y=$(y_1, y_2, y_3 ... y_r)$ and r is the number of user characteristics or features used for the clustering computation. Thus, the distance metric for finding nearest centroid, is dependent on the type (numerical or categorical as described before) and number of user features. With this approach, the number of comparisons for matching an online user to the person identifiers in the synthetic information system reduces to comparisons with k centroids of clusters. The matching time complexity is in the order of $O(k)$.

The main assumption of our approach is that the clustering algorithm creates clusters in the synthetic population such that individuals represented by points in a single cluster are more alike than those outside of the cluster, and that the centroid is the best representative of all the points in a single cluster. Given these assumptions, the process of finding a matching individual from the synthetic information system can be converted to a process of finding the matching cluster.

Finding an optimal number of clusters such that each cluster can represent a specific subset of individuals of a population that distinguish them from those in other clusters is a challenging problem. This is because, if there are too many clusters then the matching problem remains the same in complexity as that of finding a matching individual from the synthetic information system. If there are too few clusters, then the cluster may not represent all the characteristics of the individual well, and hence the forecast may not be accurate. After applying the k-means clustering algorithm and finding the matching cluster that represents the individual, the epidemic forecast for the individual can be computed as described below, either as an aggregate forecast of the matching cluster, or randomly assigned from one of the representatives of the cluster.

8.1.2 Generating Individual-level Epidemic Forecast

The next step in the individual-level epidemic forecasting process is the process of generating the forecast, based on the identified cluster. As explained in the previous section, we assume that all individuals in the same cluster share similar features such as health state, age-group, geographic proximity, behavioral patterns and so on. So, the assumption is that they are also likely to have the same vulnerability towards infections and hence their epidemic forecast would be similar.

There are multiple approaches to computation of individual-level forecast based on the identified matching cluster. The first approach computes the infection vulnerability of the centroid of the matching cluster and assigns that as the infection vulnerability of the requesting individual. The second approach randomly selects a point/identifier from the matching cluster and assigns the infection vulnerability of the randomly selected point to the requesting individual. The third approach derives an average of the infection vulnerabilities of all the points in the cluster and assigns it to the requesting individual.

In our current version, we have implemented the second approach of individual level forecasting, where a random PID from the matching cluster is chosen as a representative of the user and it's infection vulnerability is assigned to the user.

The infection vulnerability of each member of the synthetic information system is calculated using our forecasting pipeline. We use EpiFast, an agent-based model of infection computation that is built on top of the synthetic information system. The input to the forecasting simulation includes surveillance data about infections gathered from various surveillance sources like CDC, along with demographic and geographic information about regions, which is part of the synthetic information system.

The computational forecast outcome from EpiFast is calculated over several replicates and is as shown in Figure 8.1, where PID represents the person identifier of an individual in the region, REP represents the replicate number in which the PID is exposed to an infection, Exposed_Time_Date is the day on which the PID is exposed to the infection, Region_Id is the identifier of the region in which the PID belongs (such as a US state) and Level_Area_ID represents the fine grained geographical region within the larger region (such as county and block group in a state). The field Wednesday_Numeric_Date is used for optimization of forecast retrieval over a week.

The overall infection vulnerability of an individual PID is based on the number of replicates across which the PID is marked as being in the "Exposed" state, on the same day, represented by Exposed_Time_Date. If a PID appears as being exposed to infection on day 'd' in 'p' replicates out of the total 'q' replicates used in forecast computation, then the infection vulnerability of the PID is calculated as $Infection\_Vulnerability = p/q$. For instance, if a PID is computed to be in the Exposed state on day 20 of the simulation in 6 out of 10 replicates, then the infection vulnerability of the PID to an ongoing epidemic is calculated as 0.6.

The infection vulnerability is pre-computed and stored for every person, represented by a PID of the synthetic population by executing EpiFast simulations over multiple replicates. This is done as soon as input surveillance data is available, so that the forecast related to every synthetic individual is up-to-date in the system.

As described before, when an individual requests for his/her infection vulnerability to an ongoing epidemic, then first a matching cluster is computed for the user and then the pre-computed infection vulnerability of a randomly chosen PID from the matching cluster is returned back to the user.

The overall algorithm for computing individual-level epidemic vulnerability to infections is as follows:

1. Step 1: Use an appropriate clustering algorithm such as k-means to cluster a population represented using synthetic data for all the regions under consideration.
2. Step 2: Pre-compute and store the list of person identifiers (PIDs) belonging to each cluster along with their centroids.
3. Step 3: Use an agent-based modeling simulation like EpiFast for pre-computing forecasts using synthetic information systems. The model should also be able to derive individual-level infection vulnerability over several replicates, for all the PIDs belonging to the region under consideration.
4. Step 4: During the online computation of individual-level epidemic forecast, collect all the information regarding health and behavioral characteristics from the user requesting infection forecast. These should be able to be mapped to the features used in the clustering algorithm.
5. Step 5: Find the matching cluster from the synthetic data that most closely matches the requesting user by computing the smallest Euclidean distance from the cluster centroids.
6. Step 6: Lastly, use an appropriate technique of computing actual forecast based on the matching cluster, such as aggregation of all forecast counts in the cluster, or random assignment of forecast vulnerability from the matching cluster, and use the precomputed forecast result from step 3 to derive the outcome.

8.1.3 Individual-level Forecast Availability over the Web

The individual-level epidemic forecast derived using the methodology described above, needs to be published as an API so that it can be consumed by Web UI based applications. Hence, similar to our web-services based implementation of publishing aggregate-level forecast of various epidemics across different regions described in Chapter 5, we have developed RESTful web services to expose individual-level epidemic forecast to external applications.

The web services we have developed fall in two categories - external web services and internal web services. For example, getInfectionVulnerability() is an external web service, that takes as input individual-level features such as age, gender, household income, health state etc. that are related to the clustering algorithm and sends back appropriate vulnerability score.

The internal web services are mainly used for doing certain pre-computations. We have developed two main types of internal web services that are used by the external web service, getInfection-Vulnerability() for online computation of individual forecast. Following are two main examples of internal web services that we have developed.

1. findMatchingIndividual() is an internal web service for finding matching individual in the synthetic system, that takes as input user parameters and returns the PID of the matching individual from the synthetic population. Euclidean distance is used as a measure to match the user with pre-computed cluster centroids. A random PID is chosen from the list of PIDs from the matching cluster.

2. findInfectionVulnerability() is an internal web service for infection vulnerability retrieval, that takes as input a person identifier (PID) from the synthetic population and sends back the associated infection vulnerability score that is stored in the database.

Using the above web services, new applications can be built for individual-level epidemic forecasting for a range of diseases and across multiple regions, without the need to develop forecasting models from scratch.

Summary of Contributions:

In this Chapter, we have introduced a basic approach for forecasting of epidemics at an individual level by presenting a vulnerability score associated with an individual's likelihood of getting infected. We have built our system using a combination of agent-based simulations and machine learning techniques for pre-computation of large scale forecast results. The first part of the process involving finding matching individuals in the synthetic population, has applications in several other domains that involve understanding of social, demographic or behavioral characteristics of individuals through use of synthetic information systems such as product marketing, genetic engineering and so on. For instance, given a synthetic information system of a region representing multiple individuals, it is critical to find a matching individual from the synthetic population that can closely represent the user, without disclosing any private information related to the user. The clustering approach that we have proposed provides a novel approach for pre-computations so that the size of the matching problem can be vastly reduced. Such optimizations can have applicability in other domains as well, where pre-computations can lead to run time efficiency.

The main contribution of our approach is the demonstration of the use of machine learning techniques like clustering to optimize the processing of large scale data through pre-computations. We have demonstrated how this approach can work for individual-level epidemic forecasting problem in near real time, which is both data and compute intensive. Going forward, more sophisticated techniques and algorithms can be developed on top of the basic approach that we have proposed here for individual matching in the synthetic population as well as for forecasting.

Chapter 9

Discussion and Conclusion

In this thesis, we have presented several data integration techniques and approaches that can be applied in the evaluation and forecasting of epidemics in different regions of the world. We have also presented a systems perspective for large scale computations and analysis using service oriented abstractions, that can be used for exposing the data related to epidemic forecasts in an interactive way.

Our approach provides a platform-as-a-service solution for evaluation of interventions in epidemic containment and forecasting. In addition, the applications that we have developed, including DISIMS, FluCaster and EpiCaster present a software-as-a-service solution for in-depth analysis of a range of epidemics. In the following section, we briefly discuss how the services and solutions that we have presented in this thesis can be priced for commercial use. Lastly, we present our concluding remarks by summarizing the thesis contributions.

9.1 Pricing of Web Services for Epidemic Evaluation and Forecasting

Pricing of web services is an important aspect of technical discussion, especially if the services need to be commercialized for use by third-party applications. The pricing plans are dependent on the level of information that needs to be exposed to the external world as well as the overhead of handling the privacy and security of data and systems.

Currently, our data is completely isolated from use by third party applications and is secured behind a protected firewall. The data in the database is available via RESTful web services, that have been published through a central web server, protected behind a firewall.

Currently, we have made our REST-based APIs available only to requesting applications. The requesting applications need to route the request through a machine located within the same domain where our REST-based APIs are deployed (vbi.vt.edu). In addition, the requesting applications need to be in the same Virtual Private Network (VPN) to access the services. The applications themselves can be made available to the general public over the Web for free.

As our platform becomes more mature, we may publish the Web services to third party applications for a fee, based on the requirement from the APIs. The pricing in this case will depend on the type of web service requested, along with the service guarantees that need to be fulfilled, including reliability and availability. A system with higher reliability requirement would also need to plan for redundancy, and back up in case of failures, along with appropriate recovery mechanisms. Hence, the deployment server of the system will have to be designed taking into consideration metrics such as Mean Time Between Failure (MTBF) and Mean Time To Recover(MTTR).

The overall pricing of the Web service can be thought of as a function of -

1. Value of the data (Based on pre-determined price-tag on data based on difficulty in gathering raw data and effort in transformation)
2. Lines of code implemented to provide the service
3. Man-hours required for design and implementation of the service algorithms
4. Man-hours required for routine maintenance
5. Cost of physical hardware used for data storage and computations. This is dependent on the redundancy that needs to be built-in to ensure availability, reliability etc.
6. Cost of supporting software based on licensing costs

We have currently used cluster based computing resources for data storage and computations in our implementation. Our approach provides access to high-performance computing resources as a service for epidemic analysis. For pricing of these service offerings, we can extend some of the pricing methodologies from the cloud computing literature. Since, cloud computing has evolved as a mature technology that has been widely accepted and used, research on business aspects of cloud computing such as Martson et al. [64], Chaisiri et al. [29], Macias et al. [63] and Durkee [38] can provide some useful perspective on pricing of our analytics platform.

9.2 Concluding Remarks

The field of epidemic forecasting is rapidly growing with the availability of more surveillance data on ongoing epidemics like flu and Ebola from multiple sources including social media. At the same time, there are new epidemics that are developing in different parts of the world all the time which have different patterns of propagation. The recent outbreak of MERS in South Korea is a notable example.

As an epidemic evolves, it is important for health analysts and policy makers to be equipped with the latest information on the epidemic for situation assessment, based on the latest surveillance data. It is also important for them to understand the projection of the epidemic in the future, with or without any policy level or behavioral interventions taken by individuals or enforced by policy, so that they can come up with appropriate mitigation strategies.

With the decoupled data architecture that we have designed, data about new diseases can be quickly incorporated into the system and can be made available through the Web for analysis. For example, the Web application that we have developed, EpiCaster, can gather and store surveillance information on new outbreaks into the system and execute forecasting simulations (depending on availability of surveillance data), without the need for re-engineering or code changes. EpiCaster can also incorporate new forecasting models for new diseases and execute them through a data pipeline, so that the situation assessment and forecast results can be viewed on heat maps and epicurves.

The analytics platform that we have developed allows detailed analysis of epidemics for public health decision makers through flexible query mechanism. Many different types of applications can be built on top of our platform for presenting insightful analysis on different epidemics. The de-coupled architecture that we have designed has significantly reduced the time for development and deployment of such new applications. The result of our approach is that health experts can spend more time doing data analysis on ongoing epidemics through user-friendly web UI, rather than having to develop computational systems from scratch.

The design and development of our analytical platform including our modified star schema architecture for flexibility has applications in several other domains outside of computational epidemiology. Our approach provides flexible query mechanism at run-time on large scale data, rather than having to operate on a fixed set of query outcomes. The modified star schema design that we have implemented can help in the development of analytical solutions in other domains including product marketing. In addition, our design and development of multi-level web services can be extended and used in several other fields of analytics where data security and data separation are critical.

Overall, this thesis provides a unique solution for integration of data from from diverse data sources and design of analytical web services, for evaluation and forecasting of global epidemics. The technical solutions proposed in this work have utility and applications in several other domains and can help in building flexible and scalable analytical environments.

Bibliography

[1] Columbia Prediction of Infectious Diseases. `http://cpid.iri.columbia.edu/flu.html`.

[2] FluNearYou. `https://flunearyou.org/`.

[3] FluOutlook. `http://fluoutlook.org`.

[4] HealthMap. `https://healthmap.org/`.

[5] MappyHealth. `http://mappyhealth.com/`.

[6] NDSSL: Ebola Research. `http://www.vbi.vt.edu/ndssl/ebola`.

[7] David M. Aanensen, Derek M. Huntley, Edward J. Feil, Fada'a al Own, and Brian G. Spratt. Epicollect: Linking smartphones to web applications for epidemiology, ecology and community data collection. *PLoS ONE*, 4(9):e6968, 09 2009.

[8] Moustafa AbdelBaky, Manish Parashar, Hyunjoo Kim, Kirk E. Jordan, Vipin Sachdeva, James Sexton, Hani Jamjoom, Zon-Yin Shae, Gergina Pencheva, Reza Tavakoli, and Mary F. Wheeler. Enabling high-performance computing as a service. *Computer*, 45:72–80, 2012.

[9] Alistair Kenneth Atkinson. Tupleware: A distributed tuple space for cluster computing. In *Proc. of the 9th IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 121–126, 2008.

[10] Alain Barrat, Marc Barthelemy, and Alessandro Vespignani. *Dynamical processes in complex networks*. Cambridge University Press, 2008.

[11] Christopher L. Barrett, Richard J. Beckman, Maleq Khan, V. S. Anil Kumar, Madhav V. Marathe, Paula E. Stretz, Tridib Dutta, and Bryan L. Lewis. Generation and analysis of large synthetic social contact networks. In *Winter Simulation Conference*, pages 1003–1014, 2009.

[12] Christopher L. Barrett, Keith R. Bisset, Stephen G. Eubank, Xizhou Feng, and Madhav V. Marathe. Episimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 37:1–37:12, Piscataway, NJ, USA, 2008. IEEE Press.

[13] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. 2009.

[14] Vladimir Batagelj and Andrej Mrvar. Pajek-program for large network analysis. *Connections*, 21(2):47–57, 1998.

[15] Jyoti Batheja and Manish Parashar. Adaptive cluster computing using javaspaces. In *Proc. of the 3rd IEEE International Conference on Cluster Computing*, pages 323–, 2001.

[16] Richard J. Beckman, Keith R. Bisset, Jiangzhuo Chen, Bryan L. Lewis, Madhav V. Marathe, and Paula E. Stretz. Isis: A networked-epidemiology based pervasive web app for infectious disease pandemic planning and response. *Public health*, 19:18.

[17] Alysson Neves Bessani. BTS: A Byzantine Fault-Tolerant Tuple Space. In *In Proceedings of the 21st ACM Symposium on Applied Computing - SAC 2006*, pages 429–433, 2006.

[18] Keith R. Bisset, Jiangzhuo Chen, Suruchi Deodhar, Xizhou Feng, Yifei Ma, and Madhav V. Marathe. Indemics: An interactive high-performance computing framework for data-intensive epidemic modeling. *ACM Trans. Model. Comput. Simul.*, 24(1):4:1–4:32, January 2014.

[19] Keith R. Bisset, Jiangzhuo Chen, Xizhou Feng, V. S. Anil Kumar, and Madhav V. Marathe. EpiFast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems. In *Proceedings of the 23rd International Conference on Supercomputing*, pages 430–439, June 2009.

[20] Keith R. Bisset, Jiangzhuo Chen, Xizhou Feng, Yifei Ma, and Madhav V. Marathe. Indemics: an interactive data intensive framework for high performance epidemic simulation. In *Proc. the 24rd international conference on Conference on Supercomputing*, pages 233–242, 2010.

[21] Keith R. Bisset, Suruchi Deodhar, Hemant Makkapati, Madhav V. Marathe, Paula E. Stretz, and Christopher L. Barrett. Simfrastructure: A flexible and adaptable middleware platform for modeling and analysis of socially coupled systems. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:506–513, 2013.

[22] Keith R. Bisset, Xizhou Feng, Madhav V. Marathe, and Shrirang M. Yardi. Modeling interaction between individuals, social networks and public policy to support public health epidemiology. In *Proceedings of the Winter Simulation Conference (WSC)*, pages 2020–2031, 2009.

[23] Wouter Broeck, Corrado Gioannini, Bruno Goncalves, Marco Quaggiotto, Vittoria Colizza, and Alessandro Vespignani. The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale. *BMC Infectious Diseases*, 11(1):37, 2011.

[24] John S. Brownstein, Clark C. Freifeld, and Lawrence C. Madoff. Digital disease detection - harnessing the web for public health surveillance. *New England Journal of Medicine*, 360(21):2153–2157, 2009.

[25] John S. Brownstein, Clark C. Freifeld, Ben Y. Reis, and Kenneth D. Mandl. Surveillance sans frontieres: Internet-based emerging infectious disease intelligence and the healthmap project. *PLoS Med*, 5(7):e151, 07 2008.

[26] Declan Butler. When google got flu wrong. *Nature*, 494(7436):155, 2013.

[27] Simon Cauchemez et al. Closure of schools during an influenza pandemic. *The Lancet Infectious Diseases*, 9(8):473–481, August 2009.

[28] Dennis L. Chai, M. Elizabeth Halloran, Valerie J. Obenchain, and Ira M. Longini Jr. Flute, a publicly available stochastic influenza epidemic simulation model. *PLoS computational biology*, 6(1):e1000656, 2010.

[29] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. Optimization of resource provisioning cost in cloud computing. *Services Computing, IEEE Transactions on*, 5(2):164–177, April 2012.

[30] Prithwish Chakraborty, Pejman Khadivi, Bryan L. Lewis, Aravindan Mahendiran, Jiangzhuo Chen, Patrick Butler, Elaine O. Nsoesie, Sumiko R. Mekaru, John S. Brownstein, Madhav V. Marathe, et al. Forecasting a moving target: Ensemble models for ili case count predictions. In *SIAM International Conference on Data Mining*, 2014.

[31] Rumi Chunara, Susan Aman, Mark Smolinski, and John S. Brownstein. Flu near you: an online self-reported influenza surveillance system in the usa. *Online Journal of Public Health Informatics*, 5(1), 2013.

[32] CNN. 31 New York schools closed as flu spreads. Available at http://www.cnn.com/2009/HEALTH/05/21/ny.flu.schools/, May 2009.

[33] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[34] Suruchi Deodhar, Keith R. Bisset, Jiangzhuo Chen, Yifei Ma, and Madhav V. Marathe. An interactive, web-based high performance modeling environment for computational epidemiology. *ACM Transactions on Management Information Systems (TMIS)*, 5(2):7, 2014.

[35] Suruchi Deodhar, Jiangzhuo Chen, Mandy Wilson, Keith R. Bisset, Christopher L. Barrett, and Madhav V. Marathe. EpiCaster: An integrated web application for situation assessment and forecasting of global epidemics. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, BCB '15, pages 156–165, New York, NY, USA, 2015. ACM.

[36] Suruchi Deodhar, Jiangzhuo Chen, Mandy Wilson, Manikandan Soundarapandian, Keith R. Bisset, Bryan Lewis, Christopher L. Barrett, and Madhav V. Marathe. FluCaster: A pervasive web application for high resolution situation assessment and forecasting of flu outbreaks. In *2015 International Conference on Healthcare Informatics, ICHI 2015, Dallas, TX, USA, October 21-23, 2015*, pages 105–114, 2015.

[37] Ciprian Docan, Manish Parashar, and Scott Klasky. Dataspaces: an interaction and coordination framework for coupled simulation workflows. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 25–36, 2010.

[38] Dave Durkee. Why cloud computing will never be free. *Queue*, 8(4):20, 2010.

[39] Edlund and Kaufman 2012. STEM: Spatio-temporal epidemiological modeler. http://www.eclipse.org/stem/.

[40] Fritjof Boger Engelhardtsen, Tommy Gagnes, Fritjof Boger, and Engelhardtsen Tommy Gagnes. Using javaspaces to create adaptive distributed systems, 2002.

[41] Stephen G. Eubank. Scalable, efficient epidemiological simulation. In *ACM Symposium on Applied Computing*, pages 139–145, Madrid, Spain, March 2002.

[42] Stephen G. Eubank, Hasan Guclu, V. S. Anil Kumar, Madhav V. Marathe, Aravind Srinivasan, Zoltan Toroczkai, and Nan Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429:180–184, 2004.

[43] Stephen G. Eubank, Hasan Guclu, V. S. Anil Kumar, Madhav V. Marathe, Aravind Srinivasan, Zoltan Toroczkai, and Nan Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 4:180–184, May 2004.

[44] Neil M. Ferguson, Derek A. T. Cummings, Christophe Fraser, James C. Cajka, Philip C. Cooley, and Donald S. Burke. Strategies for mitigating an influenza pandemic. *Nature*, 442:448–452, 2006.

[45] Neil M. Ferguson, Matt J. Keeling, W. John Edmunds, Raymond Gani, Bryan T. Grenfell, Roy M. Anderson, and Steve Leach. Planning for smallpox outbreaks. *Nature*, 425:681–685, 2003.

[46] Daniel Fiedler, Kristen Walcott, Thomas Richardson, Gregory M. Kapfhammer, Ahmed Amer, and Panos K. Chrysanthis. Towards the measurement of tuple space performance. *SIGMETRICS Perform. Eval. Rev.*, 33(3):51–62, December 2005.

[47] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000. AAI9980887.

[48] David Gelernter. Generative communication in linda. *ACM Trans. Program. Lang. Syst.*, 7(1):80–112, January 1985.

[49] Timothy C. Germann, Kai Kadau, Ira M. Longini, and Catherine A. Macken. Mitigation strategies for pandemic influenza in the United States. *Proceedings of the National Academy of Sciences*, 103(15):5935–5940, 2006.

[50] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2008.

[51] Peter J. Haas, Paul P. Maglio, Patricia G. Selinger, and Wang Chiew Tan. Data is dead... without what-if models. *PVLDB*, 4(12):1486–1489, 2011.

[52] Mohammad Hashemian, Dylan Knowles, Jonathan Calver, Weicheng Qian, Michael. C. Bullock, Scott Bell, Regan L. Mandryk, Nathaniel Osgood, and Kevin G. Stanley. iepi: an end to end solution for collecting, conditioning and utilizing epidemiologically relevant data. In *Proceedings of the 2nd ACM international workshop on Pervasive Wireless Healthcare*, MobileHealth '12, pages 3–8, New York, NY, USA, 2012. ACM.

[53] Kyle S. Hickmann, Geoffrey Fairchild, Reid Priedhorsky, Nicholas Generous, James M Hyman, Alina Deshpande, and Sara Y Del Valle. Forecasting the 2013–2014 influenza season using wikipedia. *arXiv preprint arXiv:1410.7716*, 2014.

[54] Jeff Howe. *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*. Crown Publishing Group, New York, NY, USA, 1 edition, 2008.

[55] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values, 1998.

[56] L. Hufnagel, D. Brockmann, and T. Geisel. Forecast and control of epidemics in a globalized world. *Proceedings of the National Academy of Sciences*, 101:15124–15129, 2004.

[57] John Grefenstette, Shawn Brown, Roni Rosenfeld, Jay DePasse, Nathan Stone, Phil Cooley, Bill Wheaton, Alona Fyshe, David Galloway, Anuroop Sriram, Hasan Guclu, Thomas Abraham, Don Burke. FRED: Framework for Reconstructing Epidemiological Dynamics. https://www.midas.pitt.edu/fred/.

[58] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *In AAMAS*, 2012.

[59] Matt J. Keeling and Ken T. D. Eames. Networks and epidemic models. *J. R. Soc. Interface*, 2:295, 2005.

[60] Yarden Livnat, Theresa-Marie Rhyne, and Matthew H. Samore. Epinome: A visual-analytics workbench for epidemiology data. *Computer Graphics and Applications, IEEE*, 32(2):89 –95, march-april 2012.

[61] Eric T. Lofgren, M. Elizabeth Halloran, Caitlin M. Rivers, John M. Drake, Travis C. Porco, Bryan L. Lewis, Wan Yang, Alessandro Vespignani, Jeffrey Shaman, Joseph NS Eisenberg, et al. Opinion: Mathematical models: A key tool for outbreak response. *Proceedings of the National Academy of Sciences*, 111(51):18095–18096, 2014.

[62] Yifei Ma, Keith R. Bisset, Jiangzhuo Chen, Suruchi Deodhar, and Madhav V. Marathe. Formal specification and experimental analysis of an interactive epidemic simulation framework. In *HPCC*, pages 790–795, 2011.

[63] Mario Macías and Jordi Guitart. A genetic model for pricing in cloud computing markets. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 113–118. ACM, 2011.

[64] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing - the business perspective. *Decision Support Systems*, 51(1):176 – 189, 2011.

[65] Brendan McCane and Michael Albert. Distance functions for categorical and mixed variables. *Pattern Recognition Letters*, 29(7):986 – 993, 2008.

[66] Stefano Merler, Marco Ajelli, Laura Fumanelli, Marcelo FC Gomes, Ana Pastore y Piontti, Luca Rossi, Dennis L Chao, Ira M Longini, M Elizabeth Halloran, and Alessandro Vespignani. Spatiotemporal spread of the 2014 outbreak of ebola virus disease in liberia and the effectiveness of non-pharmaceutical interventions: a computational modelling analysis. *The Lancet Infectious Diseases*, 2015.

[67] Lauren Ancel Meyers. Contact network epidemiology: Bond percolation applied to infectious disease prediction and control. *Bulletin of The American Mathematical Society*, 44:63–86, 2007.

[68] Lauren Ancel Meyers and Nedialko Dimitrov. Mathematical approaches to infectious disease prediction and control. 2010.

[69] M. Newman, I. Jensen, and R.M. Ziff. Percolation and epidemics in a two-dimensional small world. *Physical Review E*, 65:021904, 2002.

[70] Hiroshi Nishiura. Real-time forecasting of an epidemic using a discrete time stochastic model: a case study of pandemic influenza (h1n1-2009). *Biomed Eng Online*, 10:15, 2011.

[71] Elaine O. Nsoesie, Richard J. Beckman, Sara Shashaani, Kalyani S. Nagaraj, and Madhav V. Marathe. A simulation optimization approach to epidemic forecasting. *PLoS ONE*, 8(6):e67164, 06 2013.

[72] Elaine O. Nsoesie, John S. Brownstein, Naren Ramakrishnan, and Madhav V. Marathe. A systematic review of studies on forecasting the dynamics of influenza outbreaks. *Influenza and Other Respiratory Viruses*, 2013.

[73] Elaine O. Nsoesie, Madhav V. Mararthe, and John S. Brownstein. Forecasting peaks of seasonal influenza epidemics. *PLoS currents*, 5, 2013.

[74] Nathaniel Osgood and Juxin Liu. Towards closed loop modeling: Evaluatng the prospects for creating recurrently regrounded aggregate simulation models using particle filtering. In *Proceedings of the 2014 Winter Simulation Conference*, WSC '14, pages 829–841, Piscataway, NJ, USA, 2014. IEEE Press.

[75] Shrideep Pallickara, Jaliya Ekanayake, and Geoffrey Fox. Granules: A lightweight, streaming runtime for cloud computing with support, for map-reduce. In *CLUSTER*, pages 1–10. IEEE, 2009.

[76] Shrideep Pallickara and Geoffrey Fox. Naradabrokering: a distributed middleware framework and architecture for enabling durable peer-to-peer grids. In *Proc. of the ACM/IFIP/USENIX International Conference on Middleware*, pages 41–61, 2003.

[77] Jon Parker and Joshua M. Epstein. A distributed platform for global-scale agent-based models of disease transmission. *ACM Transactions on Modeling and Computer Simulation*, 22(1), 2012.

[78] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemics and immunization in scale-free networks. In S. Bornholdt and H. G. Schuster, editors, *Handbook of Graphs and Networks*. Wiley-VCH, Berlin, 2002.

[79] Michael J. Paul and Mark Dredze. You are what you tweet: Analyzing twitter for public health. In *ICWSM*, pages 265–272, 2011.

[80] Michael J. Paul, Mark Dredze, and David Broniatowski. Twitter improves influenza forecasting. *PLoS currents*, 6, 2013.

[81] David M. Pigott, Nick Golding, Adrian Mylne, Zhi Huang, Andrew J. Henry, Daniel J. Weiss, Oliver J. Brady, Moritz UG Kraemer, David L. Smith, Catherine L. Moyes, et al. Mapping the zoonotic niche of ebola virus disease in africa. *Elife*, 3:e04395, 2014.

[82] Weicheng Qian, NathanielD. Osgood, and KevinG. Stanley. Integrating epidemiological modeling and surveillance data feeds: A kalman filter based approach. In WilliamG. Kennedy, Nitin Agarwal, and ShanchiehJay Yang, editors, *Social Computing, Behavioral-Cultural Modeling and Prediction*, volume 8393 of *Lecture Notes in Computer Science*, pages 145–152. Springer International Publishing, 2014.

[83] H. Ralambondrainy. A conceptual version of the k-means algorithm. *Pattern Recogn. Lett.*, 16(11):1147–1157, November 1995.

[84] Caitlin M. Rivers. Ebola: models do more than forecast. *Nature*, 515(7528):492–492, 2014.

[85] Caitlin M. Rivers, Eric T. Lofgren, Madhav V. Marathe, Stephen G. Eubank, and Bryan L. Lewis. Modeling the Impact of Interventions on an Epidemic of Ebola in Sierra Leone and Liberia. *PLoS Currents*, 2014.

[86] Leonid A. Rvachev and Ira M. Longini. A mathematical model for the global spread of influenza. *Mathematical Biosciences*, 17:3–22, 1985.

[87] Marcel Salathe, Linus Bengtsson, Todd J. Bodnar, Devon D. Brewer, John S. Brownstein, Caroline Buckee, Ellsworth M. Campbell, Ciro Cattuto, Shashank Khandelwal, Patricia L. Mabry, et al. Digital epidemiology. *PLoS computational biology*, 8(7):e1002616, 2012.

[88] Jeffrey Shaman and Alicia Karspeck. Forecasting seasonal outbreaks of influenza. *Proceedings of the National Academy of Sciences*, 109(50):20425–20430, 2012.

[89] Jeffrey Shaman, Alicia Karspeck, Wan Yang, James Tamerius, and Marc Lipsitch. Real-time influenza forecasts during the 2012–2013 season. *Nature communications*, 4, 2013.

[90] Jeffrey Shaman, Wan Yang, and Sasikiran Kandula. Inference and forecast of the current west african ebola outbreak in guinea, sierra leone and liberia. *PLoS currents*, 6, 2014.

[91] Ramanathan Sugumaran and Jonathan Voss. Real-time spatio-temporal analysis of west nile virus using twitter data. In *Proceedings of the 3rd International Conference on Computing for Geospatial Research and Applications*, page 39. ACM, 2012.

[92] Sun Microsystems Inc. Javaspaces service specification version 2.2.

[93] James D Tamerius, Jeffrey Shaman, Wladmir J Alonso, Kimberly Bloom-Feshbach, Christopher K Uejio, Andrew Comrie, and Cécile Viboud. Environmental predictors of seasonal influenza epidemics across temperate and tropical climates. *PLoS pathogens*, 9(3):e1003194, 2013.

[94] Time. CDC says H1N1 outbreak shouldn't close schools. Available at http://www.time.com/time/health/article/0,8599,1915244,00.html, August 2009.

[95] Michele Tizzoni, Paolo Bajardi, Chiara Poletto, José J Ramasco, Duygu Balcan, Bruno Gonçalves, Nicola Perra, Vittoria Colizza, and Alessandro Vespignani. Real-time numerical forecast of global epidemic spreading: case study of 2009 a/h1n1pdm. *BMC medicine*, 10(1):165, 2012.

[96] Cécile Viboud, Pierre-Yves Boëlle, Fabrice Carrat, Alain-Jacques Valleron, and Antoine Flahault. Prediction of the spread of influenza epidemics by the method of analogues. *American Journal of Epidemiology*, 158(10):996–1006, 2003.

[97] Sudheendra Vijayanarasimhan and Kristen Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1449–1456, Washington, DC, USA, 2011. IEEE Computer Society.

[98] Joseph T. Wu et al. School closure and mitigation of pandemic (H1N1) 2009, Hong Kong. *Emerging Infectious Disease*, 16(3):538–541, March 2010.

[99] Reza Yaesoubi and Ted Cohen. Dynamic health policies for controlling the spread of emerging infections: Influenza as an example. *PLoS ONE*, 6(9), 09 2011.

[100] Wan Yang, Subbiah Elankumaran, and Linsey C Marr. Relationship between humidity and influenza a viability in droplets and implications for influenza's seasonality. *PloS one*, 7(10), 2012.

[101] Wan Yang, Alicia Karspeck, and Jeffrey Shaman. Comparison of filtering methods for the modeling and retrospective forecasting of influenza epidemics. *PLoS computational biology*, 10(4), 2014.

[102] Wan Yang, Marc Lipsitch, and Jeffrey Shaman. Inference of seasonal and pandemic influenza transmission dynamics. *Proceedings of the National Academy of Sciences*, 112(9):2723–2728, 2015.

[103] Eiko Yoneki. Fluphone study: Virtual disease spread using haggle. In *Proceedings of the 6th ACM Workshop on Challenged Networks*, CHANTS '11, pages 65–66, New York, NY, USA, 2011. ACM.

[104] Bin Yu, Jijun Wang, Michael McGowan, Ganesh Vaidyanathan, and Kristofer Younger. Gryphon: a hybrid agent-based modeling and simulation platform for infectious diseases. *Advances in Social Computing*, pages 199–207, 2010.

[105] Qingyu Yuan, Elaine O. Nsoesie, Benfu Lv, Geng Peng, Rumi Chunara, and John S. Brownstein. Monitoring influenza epidemics in china with search query from baidu. *PloS one*, 8(5), 2013.