Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

EFFICIENT IMPLEMENTATION OF COMPLEX INTERVENTIONS IN LARGE SCALE EPIDEMIC SIMULATIONS

Yifei Ma
Keith Bisset
Jiangzhuo Chen
Suruchi Deodhar
Madhav Marathe

Network Dynamics and Simulation Science Laboratory, Virginia Tech
1880 Pratt Drive
Blacksburg, VA 24061, USA

ABSTRACT

Realistic agent-based epidemic simulations usually involve a large scale social network containing individual details. The co-evolution of epidemic dynamics and human behavior requires the simulation systems to compute complex real-world interventions. Calls from public health policy makers for executing such simulation studies during a pandemic typically have tight deadlines. It is highly desirable to implement new interventions in existing high-performance epidemic simulations, with minimum development effort and limited performance degradation. Indemics is a database supported high-performance epidemic simulation framework, which enables complex intervention studies to be designed and executed within a short time. Unlike earlier approaches that implement new interventions inside the simulation engine, Indemics utilizes DBMS and reduces implementation effort from weeks to days. In this paper, we propose a methodology for modeling and predicting performance of Indemics-supported intervention studies. We demonstrate our methodology with experimental results.

1 INTRODUCTION

Recently, high-performance agent-based epidemic simulations have been applied to study disease dynamics in large scale populations and to evaluate the effectiveness of various intervention strategies (Ferguson et al. 2006, Parker 2007, Barrett et al. 2008, Chao et al. 2010). For example, during the recent H1N1 pandemic, we executed several case studies, sponsored by U.S. federal agencies like CDC, DHS, DoD, and DHHS, using EpiFast (Bisset et al. 2009) and Indemics (Bisset et al. 2010) simulation systems. These case studies demonstrated the criticality of determining how effective human interventions are during an on-going epidemic. Such human interventions are often evaluated using simulations to study their effects on epidemic propagation before being applied in the real world. Unfortunately, intervention strategies are often unknown to the simulation software developers until the request for the case study comes. Furthermore, these case study simulations typically demand simulation results within a short period of time. In a real world scenario, the execution of strategy simulations must be accomplished within a short window, or the results will be obsolete. Due to these factors, there are unique challenges posed on how to efficiently implement interventions in epidemic simulations. Some of these challenges include reducing the human effort involved in implementing new types of interventions, in addition to optimizing the running time of the simulation.

978-1-4577-2109-0/11/$26.00 ©2011 IEEE

1.1 Background

Large scale agent-based epidemic simulations. Realistic simulations of disease propagation in social contact networks usually involve city, state, or even national level populations. These networks often consist of millions, sometimes billions, of agents, and hundreds of millions of inter-agent connections. For example, the New York City contact network used in our simulations for the 2009 H1N1 pandemic has 18 million nodes and almost 1 billion edges. To generate realistic results and to provide meaningful and implementable mitigation strategies for public health policy makers, individual level heterogeneous details, including demographic data, activity schedule, health state, as well as behavioral response to the ongoing epidemic, are considered and computed in these simulations. Furthermore, epidemic simulations are often applied to explore a large space of possible intervention policies and have factorial experiment designs. Sometimes, the simulations are re-run repeatedly to adapt to the surveillance data. Achieving good performance of the simulation tool for large problem size is a major challenge.

Complex interventions. Following are some examples of epidemic interventions:

1. Vaccinate randomly chosen people.
2. Vaccinate people (nodes) with high degrees in the contact network.
3. Keep all school age children home for 2 weeks.
4. Each county decides to close its schools if the number of diagnosed students in the county exceeds its preset threshold. Students from closed schools will stay home.
5. Same as the last intervention, plus an additional constraint that for each student that stays home, if the student's age is less than 12, then there must be a guardian staying at home too.

Real-world interventions can be even more complicated than these examples. Oftentimes the interventions are unknown until the diffusion process of the simulation has begun. Many existing simulation tools can handle only the first two examples cited above. To implement a completely new intervention type, significant changes have to be made in the simulation engine code, which is time consuming and requires expertise in high performance computing (HPC). Our experience in implementing different types of interventions has shown that modeling the intervention strategies by coding them completely within the simulation engine like EpiFast (Bisset et al. 2009) may be possible, but this approach does not scale well. This motivated us to develop a platform called Indemics (Bisset et al. 2010), that supports quick re-deployment of interventions, reduces cumulative complexity and risk of errors in the intervention codes and potentially handles limitless intervention types. Our experiences with Indemics are discussed in Section 1.2.

Efficient implementation. The simulation tool users that need to implement new interventions are often domain experts with limited knowledge of HPC systems. Hence expert programmers are needed to make changes to the simulation engine code, based on the requirements specified by the epidemiologists. This introduces communication delays along with an extended software development life cycle, which is a flip side of implementing interventions inside the HPC-based simulation engine. On the other hand however, interventions implemented inside the simulation engine typically run faster. This presents a trade-off between performance and capability. In this paper, we study the performance of Indemics (Bisset et al. 2010) and compare the performance of implementing interventions using Indemics to that of implementing interventions within EpiFast simulation engine. To the best of our knowledge, the other existing simulation systems do not handle all intervention types exhaustively. The capability of Indemics mainly comes from using a DBMS to compute interventions for the HPC-based simulation engine. We propose a model for analyzing performance of the database queries related to computing interventions and a methodology for predicting the running time of any given intervention case study.

1.2 Motivation

Scenario 1. EpiFast is a fast agent-based epidemic simulation tool developed in our group (Bisset et al. 2009). After its initial release in 2008, users found that while it could handle these interventions: (a) antiviral prophylaxis to randomly chosen people and (b) keep all primary school students home (school closure version 1); it could not handle the more realistic versions: (a') antiviral treatment to currently sick people and (b') keep all primary school students home and for each of them let a guardian stay home too (school closure version 2). The intervention (a') could be implemented only after months of code development within EpiFast, and became available over a year later. We have not yet implemented (b') in EpiFast due to its complicated character and estimated coding effort. After we developed the Indemics system in 2009, however, it took only a few hours to implement either (a') or (b') with Indemics scripts. These scripts were written in *Indemics intervention language*, a controlled natural language that is easy to master even without any programming experience. The experience with interventions (a') and (b') convinced us about the wide applicability of Indemics for applying realistic interventions, which was earlier very difficult or almost infeasible using EpiFast.

The Indemics framework incurs limited performance overhead in executing some intervention strategies as described in (Bisset et al. 2010). The small overhead incurred in simulation running time, however, is easily offset by the huge savings of human effort in implementing new types of interventions.

Scenario 2. After the recent H1N1 pandemic, our group was requested to run a case study with a large factorial experiment design and involving several new types of interventions. The simulation results were to be presented in a CDC conference within two weeks. We decided to use Indemics to run the study. After the epidemiologist in our group specified the study design details, we examined the intervention scripts and based on our prediction model, inferred that if the current scripts were used, the design would take far more than two weeks to complete. The epidemiologist modified the study design himself and the new enhanced script was estimated to be finished in about one week. Our prediction was accurate and so the epidemiologist was able to complete the study execution in one week, with a few days left for analyzing simulation output and preparing the conference presentation. Thus, the prediction mechanism for estimating running time of the case study helped us keep the tight deadline of the case study simulation.

In this paper, we propose a formal model for Indemics performance analysis and present a methodology for predicting performance of Indemics in running given case studies. We will support our performance predicting methodology by presenting a comparison of the predicted running time and actual running time of a realistic intervention case study.

1.3 Major Contributions

In this paper we focus on performance of intervention implementation for epidemic simulations. Our major contributions are:

- We present our observations on the performance gain/loss in using Indemics to simulate epidemic interventions instead of computing everything within the simulation engine.
- We address the issue of reducing development time for new intervention strategies. With Indemics interventions are composed in an easy-to-learn controlled natural language (CNL) (Bisset et al. 2010). We demonstrate how this improves the productivity of simulation users.
- We introduce a detailed methodology for performance modeling and prediction of running intervention case studies using Indemics. Using this prediction model, the simulation designers can estimate the overall running time of the simulations before actually executing them. This gives the user reasonable confidence of completing the simulation studies in time.

2 INTERVENTION IMPLEMENTATION AND PERFORMANCE

We have two ways to simulate epidemic interventions: within the high-performance epidemic simulation engine or using external database management systems to compute interventions and apply them to the epidemic simulation engine. In what follows, we use EpiFast (Bisset et al. 2009) to represent the former and the Indemics framework (Bisset et al. 2010) to represent the latter. Also, we assume that the simulation engine used in the Indemics framework is EpiFast. The main factor for users to choose one execution method over the other should be the ability to complete end to end implementation and obtain simulation results within the time constraint of the case study. This puts a priority on short development time supported by efficient implementation, as well as reduced running time of intervention simulations. In this section we compare the efficiency and performance of different ways of implementing epidemic interventions.

2.1 Efficient Epidemic Intervention Implementation

A typical epidemic simulation consists of two co-evolving parts: disease diffusion and human intervention. Correspondingly, an epidemic simulation system usually includes diffusion code and intervention code. E.g.

$$\begin{aligned}\text{EpiFast} &= \text{diffusion code (C++)} + \text{intervention code (C++)}\\ \text{Indemics} &= \text{EpiFast diffusion code (C++)} + \text{framework code (Java)} + \text{DBMS} + \text{intervention script (CNL)}\end{aligned}$$

where CNL means a controlled natural language.

Computation of most interventions requires both dynamic data generated within the diffusion computation (e.g. current diagnosed cases) and data not directly relevant to disease diffusion (e.g. household income). Some examples of interventions include:

- **Triggered intervention**: In this intervention type, a threshold is evaluated every day. When it is exceeded, then the predefined actions are taken. e.g. *when fraction of diagnosed school-age children exceeds 20%, close all schools*.
- **Targeted intervention**: In this type, intervention action is applied to people in a certain health state i.e. a target population. e.g. *administer antiviral on diagnosed school-age children*.
- **Local triggered intervention**. In this intervention type, many subpopulations are predefined and when any subpopulation incurs a local outbreak the intervention action is applied to the whole subpopulation, e.g. *close any school where more than 5% students show symptoms* (called “school intervention”); *vaccinate all people in any census block group if more than 1% in that block group are diagnosed* (called “block intervention”).

The real world case studies for ongoing epidemics often come with a tight deadline on the delivery of study results. From our experiences with various federal agencies, we were often requested to complete a study within a few weeks, which included running a whole set of simulations for several rounds, each round having a time budget of only a few hours.

In Table 1 we present our development experiences in implementing several interventions in EpiFast vs. in Indemics. Clearly Indemics reduces development time by at least one order of magnitude. This can shorten the overall study life cycle significantly. This also largely improves the productivity of the simulation tool users by allowing them to explore larger classes of intervention strategies.

2.2 Performance Comparison

In this section, we choose a few interventions that are already implemented both for Indemics and for EpiFast, and compare the performance of simulating them in the two systems. To make an impartial comparison, simulations with EpiFast and with Indemics are the same, both for the diffusion part and for the intervention part, and produce exactly the same outcome. The only difference is that the intervention is computed using C++ code in the former solution and java plus Oracle in the latter one.

Table 1: Development cost: implementing interventions in EpiFast vs. with Indemics scripts.

intervention	EpiFast implementation		Indemics implementation	
	lines of code	effort	lines of script	effort
school/block	500	8 weeks	< 100	8 hours
triggered	150	1 week	< 100	1 hour
targeted	600	12 weeks	< 100	1 hour

Note that although we spent months to implement the two interventions in EpiFast, it only took us a few hours to write Indemics scripts for them. In this experiment, however, we are mainly concerned about simulation execution time. For EpiFast simulations we collect the total running time. For Indemics simulations we collect the total running time as well as the execution time of database queries related to intervention computation.

Triggered intervention. In this experiment we are mainly concerned about the performance of threshold evaluation. We plot database query time (for counting diagnosed people) and the total execution time of Indemics simulations along with the total running time of EpiFast simulations in Figure 1. We simulate the triggered intervention on four US urban regions: Miami, Seattle, Boston and Chicago, with increasing population sizes. Figure 1 shows that, using Indemics, the intervention simulations incur a reasonable performance loss which is easily outweighed by the benefits derived from ease of query composition.

Targeted intervention. The targeted intervention is simulated on the Miami population, under disease models with different infectivities. The moderate disease model has a lower attack rate, where attack rate is the fraction of population getting infected during the epidemic; the strong disease model has a higher attack rate. A higher attack rate leads to larger set of people to be intervened. From Figure 2 we observe that Indemics simulations perform better than EpiFast simulations for this targeted intervention.

The experiment suggests that Indemics may improve performance in some cases and incur a reasonable overhead in others. The ratio of total execution time using Indemics simulations to that using EpiFast simulations is about 2 for the trigger intervention (100% performance overhead) and about 0.5 for the targeted intervention (Indemics improves the total running time). Even in the case of the trigger intervention, Indemics is still preferable, since the benefits of Indemics derived from quick re-deployment of interventions, decrease in the risk of code errors and reduction of human effort can easily outweigh reasonable performance losses.

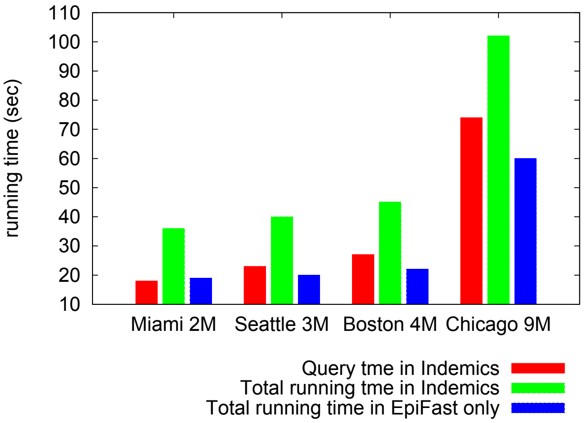

Figure 1: Triggered intervention: Indemics simulation and EpiFast simulation.

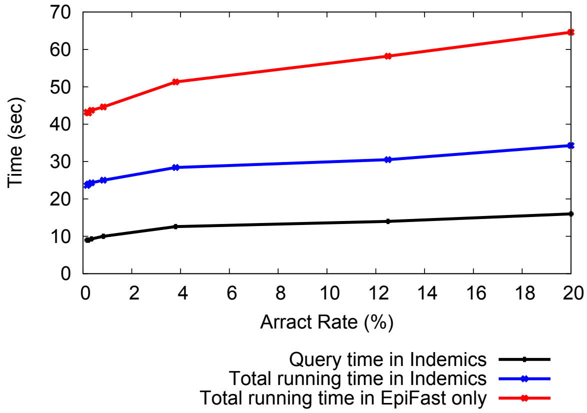

Figure 2: Targeted interventions: Indemics simulation vs. EpiFast simulation.

3 PERFORMANCE MODELING

The experiment running time of Indemics is critical for real time epidemic planning and it should be estimated during the experiment design stage. In this section, we introduce a methodology of performance modeling and prediction for Indemics. Performance prediction during experiment design can guide the user to adjust the experiment design so that the most comprehensive study can be completed within the allowable time frame.

In the Indemics framework, interventions are computed within the database. Intervention instance has two pieces of information, intervention type and intervened subpopulation. Intervention type defines the action and its parameters, such as vaccination with its efficacy. Intervened subpopulation defines the agents that will take the intervention action. Computation of interventions is mainly for the selection of the intervened subpopulation, which usually involves dynamic data from the simulation and has to be determined at run time. The selection of intervened subpopulations is written in an Indemics script, which is then translated into database queries; at run time these queries are executed in each time step to select the desired subpopulation; then the intervention type and subpopulation are sent to the simulation engine for the diffusion computation in the next time step. The following is an example of Indemics script for the *school intervention*. SQL statements that implement this intervention are presented under the controlled natural language.

```
Initialization;

Define School_Trigger as SCHOOL_DIAGNOSED_TOTAL.persons > 0.05 *
SCHOOL_INTERVENED.schoolsize
Reset table SCHOOL_INTERVENED intervened_day = -1 :
update SCHOOL_INTERVENED set intervened_day = -1;
For Day from 1 to 300
  1. Count newly diagnosed students in each school, save to SCHOOL_DIAGNOSED_TODAY :
  insert into SCHOOL_DIAGNOSED_TODAY
    select school, count(pid) as persons, Day as diag_time from STUDENT, DIAGNOSED
    where pid = diagnosed_pid and diagnosed_time = Day;

  2. Count diagnosed students in each school, save to SCHOOL_DIAGNOSED_TOTAL :
  insert into SCHOOL_DIAGNOSE_TOTAL select
    school, sum(persons) as persons, diag_time from SCHOOL_DIAGNOSED_TODAY where
    diag_time between Day and Day - 6 group by school;

  3. Set SCHOOL_INTERVENED intervened_day = Day
    if school is not intervened and School_Trigger is triggered:
  update SCHOOL_INTERVENED set intervened_day = Day
    where SCHOOL_INTERVENED.school = SCHOOL_DIAGNOSED_TOTAL.school and diag_time = Day
    and SCHOOL_DIAGNOSED_TOTAL.persons > 0.05 * SCHOOL_INTERVENED.schoolsize;

  4. Apply vaccination to schools in SCHOOL_INTERVENE with intervened_day = Day:
  select pid from STUDENT, SCHOOL_INTERVENED where
    STUDENT.school = SCHOOL_INTERVENED.school and intervened_day = Day;
```

As illustrated in the Indemics script example, the query statements specify how to manipulate dynamic data and select the intervention subpopulations from the database. The Indemics intervention script is translated into SQL statements to query the relational DBMS. SQL statements consist of SQL atomic statements. A legal SQL statement is either just an atomic statement or a composition of atomic statements. Table 2 lists the atomic SQL statements supported in the Indemics intervention language. To model the performance of a given intervention scenario, the corresponding intervention script is translated into SQL queries and each statement is decomposed into query atoms. The performance modeling is based on these query atoms. Typical query statements and their atoms are given in Table 3.

The selection of the intervened subpopulations during the intervention phase is data-intensive, and the execution time of manipulating and selecting data from database is a considerable part of the whole intervention study. It is essential for Indemics to model its performance in the intervention phase and use this performance model to predict the execution time in the study before actually performing the study. This is important because a case study is usually associated with a tight deadline. By using the performance model, the study designer can estimate execution time directly from a given intervention scenario.

Table 2: Atomic statement examples in Indemics.

Atom	Symbol	Algebra	SQL Example
Projection	π	$\pi_\alpha R$	select pid from persons
Predicate	θ	$\alpha = \beta$	age = 18
Selection	σ	$\sigma_\theta R$	select from persons where age = 10
Natural Join	$\bowtie$	$P \bowtie I$	select from persons P, infections I where P.pid = I.pid
Insert	ϕ	$\phi_S R$	Insert into S from R
Update	μ	$\mu_{\alpha=x} R$	update R set $\alpha = x$
Group By	Σ	$\Sigma_\alpha R$	group by α
Index	Ψ	$\Psi_\alpha R$	create index on $R(\alpha)$

Table 3: SQL statement examples in Indemics.

Statement	Algebra
Insert new infection cases R into table INFECTION	$\phi_{\text{INFECTION}}R$
Update status to x for schools in R	$\mu_{status=x}R$
Search household with infections	$\pi_{hid}(\text{HOUSEHOLD} \bowtie_{pid} \text{INFECTION})$
Count new infection cases in each census block	$\pi_{block,count(pid)}\Sigma_{block}\text{BLOCK} \bowtie_{pid} \text{INFECTION}$

We model Indemics performance based on query statement atoms. Given the script in Indemics language for a case study, we decompose the query statements in the script into atoms, and estimate the configurations of these atoms. The atom configuration includes the size of queried table, the size of query results and other parameters about the query. In order to predict the data query performance, we profile the performance of query atoms in the database employed by Indemics in advance. This database profile forms a performance lookup library which collects the performance of each statement atom under different configurations in the database system. The performance of query atoms can be referred to from this performance lookup library.

Definition 1 The predicted running time of a query atom α is $AP(\alpha)$. The value of $AP(\alpha)$ under different configurations can be found in the performance lookup library.

Since each query statement consists of query atoms and the composite query executes its atomic queries sequentially, the predicted performance of a given query can be calculated by summing up the performance of its atomic queries.

Definition 2 The predicted running time $QP(Q)$ for a query statement Q is given by:

$QP(Q) = AP(a_1) + AP(a_2) + ... + AP(a_n)$

where Q can be decomposed into $AP(a_1), AP(a_2), ..., AP(a_n)$.

The Indemics intervention script specifies the data manipulation and subpopulation selection on each simulation day, therefore, the predicted running time of the script is the total of the predicted running time of all query statements on each day.

Definition 3 The predicted running time $SP(S)$ of a script S implementing the intervention study is given by:

$SP(S) = \Sigma_{day=0}^{T} QP(Q_1^{day}) + QP(Q_2^{day}) + ... + QP(Q_i^{day})$.

where on each simulation "*day*", the queries $Q_1^{day}, Q_2^{day}, ..., Q_i^{day}$ are executed.

In order to predict the performance of a case study, we use the atomic query performance lookup library to calculate $SP()$ as the predicted performance for the study.

4 PERFORMANCE PREDICTION

In this section, we introduce our performance prediction method for Indemics. This prediction method is based on the performance model in Section 3. Intervention computation time is predicted by analyzing query atoms in the intervention scenario. The School Intervention is used as an example to demonstrate our prediction method.

The query execution time varies remarkably on different database management systems. It is necessary to first collect query performance data for the specific DBMS under the specific machine configuration. For example, we have collected the performance profiles of each query atom relevant to Indemics on our database management system, Oracle 10g, under typical query configurations, and have organized the performance profiles as a lookup library. This library is as comprehensive as possible to cover all relevant configurations. But due to space limit, we only show a segment of the library in Table 4.

Table 4: A segment of performance lookup library.

Statement	Algebra	Configuration	Time (seconds)
Insert	$\phi_S R$	S: 1K	0.005
Insert	$\phi_S R$	S: 100K	0.1
Update	$\mu_{S.\alpha=x} R$	R: 1K	0.002
Update	$\mu_{S.\alpha=x} R$	R: 5K	0.01
Select	$\sigma_\theta R$	R: 10K	0.001
Select	$\sigma_\theta \Psi_\alpha R$	$\sigma_\theta \Psi_\alpha R$: 2M	0.02
Nature join	$P \bowtie I$	P: 0.5M I: 1K	0.06
Nature join	$P \bowtie I$	P: 0.5M I: 100K	0.5
Group by	$\Sigma_\alpha R$	R: 10K	0.3

Table 5: Table statistics for the performance prediction experiments.

Table	Region	Size
STUDENT	Chicago	2.2M
STUDENT	Boston	0.9M
STUDENT	Seattle	0.7M
STUDENT	Miami	0.5M
SCHOOL_INTERVENED	Chicago	8K
SCHOOL_INTERVENED	Boston	6K
SCHOOL_INTERVENED	Seattle	3K
SCHOOL_INTERVENED	Miami	2K
PERSON_BLOCK	Chicago	9M
PERSON_BLOCK	Miami	2M

To predict the performance of an intervention scenario never studied before, its performance is estimated by analyzing the corresponding Indemics script s and computing $SP(s)$ as in Definition 3. Now we demonstrate how to analyze an Indemics script and calculate $SP(s)$ with a concrete case study.

4.1 Scenario of Case Study

In the School Intervention, any school where the number of students diagnosed with flu exceeds a given threshold (such as 5% of enrolled students), then vaccines will be offered to the whole school. The School Intervention will be evaluated for each simulation day. It has four steps to manipulate disease diffusion data and demographic data, and these data manipulations are abstracted into query algebra using the query atoms. The query algebra for School Intervention is illustrated in Algorithm 1. Finally, the query algebra is translated into an Indemics intervention script, as illustrated by the example in Section 3, to simulate the School Intervention strategy under the Indemics framework.

4.2 Analysis of Query Atoms

Now we present the analysis of the School Intervention study in Miami. As illustrated in Algorithm 1, our Indemics script manipulates database for School Intervention in four steps: (i) counting newly diagnosed students today in each school, (ii) counting the total diagnosed students in each school, (iii) identifying schools that have exceeded the intervention threshold, and (iv) applying the pharmaceutical treatment in the identified schools. Their algebra expressions are given in Algorithm 1.

Algorithm 1 School Intervention in Algebra

```
  Initialization;
  Reset table SCHOOL_INTERVENED intervened_day = -1 :
  $\mu_{intervened\_day=-1}$SCHOOL_INTERVENED;
  for simulation day $\alpha$ from 1 to 300 do
    1. Count newly diagnosed students today in each school, save the results in SCHOOL_DIAGNOSED_TODAY :
    $\phi_{\text{SCHOOL\_DIAGNOSED\_TODAY}}\pi_{school,persons=count(pid),diag\_time}\Sigma_{school}\Psi_{pid}$STUDENT $\bowtie_{pid}$ $\sigma_{diagnosed\_time=\alpha}$DIAGNOSED;
    2. Count diagnosed students in each school still in infectious state, save the results in SCHOOL_DIAGNOSED_TOTAL:
    $\phi_{\text{SCHOOL\_DIAGNOSED\_TOTAL}}\pi_{school,persons=sum(persons),diag\_time}$ $\Sigma_{school}\sigma_{\alpha-6<diag\_time<=\alpha}$(SCHOOL_DIAGNOSED_TODAY);
    3. Set SCHOOL_INTERVENED intervened_day = $\alpha$ if this school has not been intervened and diagnosed students cross the threshold :
    $\mu_{intervened\_day=\alpha}\sigma_{persons>0.05*schoolsize}$(SCHOOL_INTERVENED $\ltimes_{school}$ $\sigma_{diag\_time=\alpha}$SCHOOL_DIAGNOSED_TOTAL);
    4. Apply vaccination treatment on the target schools:
    $\pi_{pid}(\sigma_{intervened\_day=\alpha}$SCHOOL_INTERVENED $\ltimes_{school}$ STUDENT);
  end for
```

1. The algebra for step (i) can be decomposed into atoms $\sigma_{diag_time=\alpha}$DIAGNOSED, Ψ_{pid}STUDENT $\bowtie_{pid}$ $\sigma_{diag_time=\alpha}$DIAGNOSED , Σ_{school}, π and ϕ. The estimated average daily diagnosed cases will not exceed 1000 under the epidemic model of this experiment. Since the sizes of Ψ_{pid}STUDENT and $\sigma_{diag_time=\alpha}$DIAGNOSED are approximately 500K rows (Table 5) and 1K rows for Miami, from the performance lookup library (Table 4), we know that the performance dominating atom in these four query atoms is the join query Ψ_{pid}STUDENT $\bowtie_{pid}$ $\sigma_{diag_time=\alpha}$DIAGNOSED; and that the running time of this join atom and step (i) is around 0.06 second.
2. The algebra for step (ii) consists of the atoms $\sigma_{\alpha-6<diag\_time<=\alpha}$(SCHOOL_DIAGNOSED_TODAY), Σ, π and ϕ. Because of the estimation that the average daily diagnosed cases will not exceed 1000, the size of table $\sigma_{\alpha-6<diag\_time<=\alpha}$(SCHOOL_DIAGNOSED_TODAY) will not exceed 1k rows. From the performance lookup library, we conclude that all the atoms in step (ii) have negligible performance cost, so the running time are ignored in step (ii).
3. The algebra for step (iii) contains the performance dominating atoms, SCHOOL_INTERVENED $\ltimes_{school}$ $\sigma_{diag_time=\alpha}$SCHOOL_DIAGNOSED_TOTAL, $\sigma_{persons>0.05*schoolsize}$ and μ. The size of the table SCHOOL_INTERVENED is only 2K rows and the number of schools tagged to be intervened generally is within 1K, so we conclude that the atoms in step (iii) are also negligible.
4. The last step is $\pi_{pid}(\sigma_{intervened_day=\alpha}$SCHOOL_INTERVENED $\ltimes_{school}$ STUDENT) and the performance dominating atom is the join atom. The size of STUDENT is around 500K rows and the number of schools to be intervened is 1000 at most. The execution time of this query statement is estimated to be 0.06 seconds based on the performance lookup library.

4.3 Calculation of Predicted Performance

After the analysis of query atoms in the given Indemics script, we are able to predict its performance $SP()$. We only need to consider the query atoms with significant costs. Denote the four query statements in the School Intervention by Q_1, Q_2, Q_3 and Q_4. We can ignore $QP(Q_2)$ and $QP(Q_3)$ in $SP()$, based on the previous analysis. Eventually, $SP()$ is calculated as $\Sigma_{day=0}^{300}(QP(Q_1)+QP(Q_4)) = 300*2*0.06 = 36$ seconds.

The performance predictions for the regions other than Miami can also follow the same procedures. From the analysis on the School Intervention script, we know that the performance dominating atoms are the join atoms with table STUDENT, and the size of table STUDENT for Seattle is very close to Miami, therefore the estimated execution time for Seattle is also about 36 seconds. Chicago has twice as many students as Miami, and the School Intervention for Chicago is estimated to take 72 seconds based on the performance lookup library for the join performance.

To validate our performance prediction model, the performance of different interventions in different regions has be estimated by analysis and evaluated by simulations. The actual execution time and the predicted execution time of these selected intervention studies are illustrated in Figure 3. Although the predicted time is not so accurate, our prediction still makes sense for the study planning. It is very typical that each intervention strategy will be simulated under tens of configurations for dozens of replicates, so the whole study can take days or even weeks. Figure 4 shows the estimated development costs, experiment costs, and the total times for the whole intervention studies, which have to be repeated for 50 iterations under 20 different disease models in four regions. This figure shows that Indemics can shorten the study periods and ensure that the study results are obtained in time.

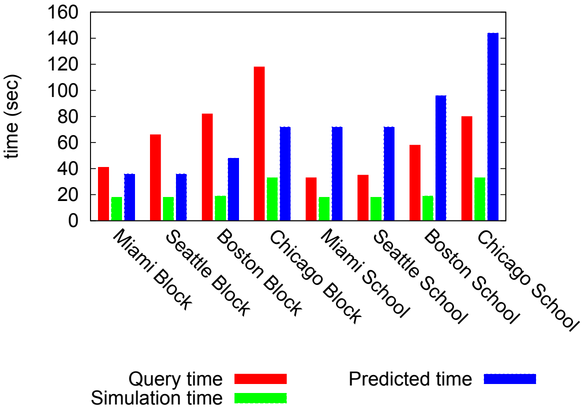

Figure 3: Performance by prediction and in experiments. Query time is actual query time for Indemics' database; simulation time is the actual diffusion simulation time; the predicted query time is our predicted query time for database.

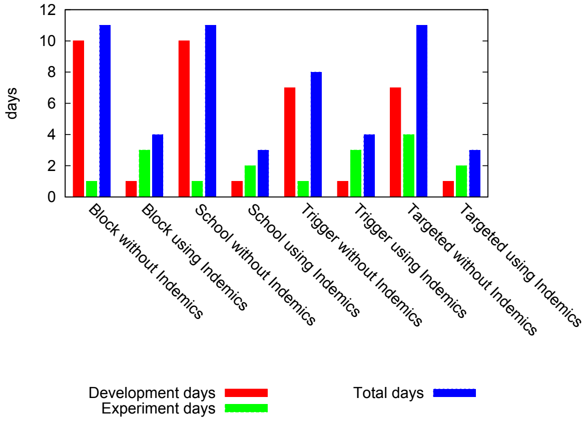

Figure 4: Comparisons of development days, experiment days and whole study periods between using simulation only and using Indemics. The experiment days of block and school interventions without using Indemics are estimated as 1 day, since they are difficult to implement without Indemics.

5 RELATED WORK

This paper extends our original work on the Indemics framework that seeks to build a high performance epidemic simulation to support public health epidemiology (Bisset et al. 2010); here we focus on performance modeling of complex interventions using DBMS. Recently there is much work on computational models for contact network epidemiology (Ferguson et al. 2006, Barrett et al. 2008, Bisset et al. 2009, Cauchemez et al. 2011). These high performance epidemic simulation tools allow public health policy makers to study intervention effectiveness during emergent epidemics, such as the 2009 H1N1 pandemic. It is still a challenge to represent complex interventions and the co-evolution of network, behavior, and epidemics. Indemics was inspired by the work of Gray et al. (Szalay and Blakeley 2009, Szalay and Gray 2006, Gray et al. 2005), which advocated the use of DBMS technology to support physical simulations. Work reported in White et al. (2007) and Wang et al. (2010) follows this work and proposes the use of DBMS to support games and social simulations. Our ongoing work on Indemics is most closely related in this line of work. In contrast to the work in Wang et al. (2010) that focuses on Hadoop and the MapReduce framework, we focus on traditional DBMS systems. We believe that Hadoop like frameworks are not suitable when dealing with complex interventions. Other related work on use of databases in related contexts can be found in (Christodoulakis 1984, Graefe 1993, Sevcik 1981, DeWitt and Hawthorn 1981).

6 CONCLUSION

We have described a systematic methodology to analyze and predict the performance of Indemics interventions. Our focus is on complex interventions that often arise during the course of analyzing an evolving epidemic. Representing and implementing the interventions is often a time consuming task. Indemics is developed to reduce the time from design to implementation of these complex interventions. The development period can be reduced from weeks to days, making the tool useful when supporting an evolving pandemic such as the recent H1N1 pandemic. A performance comparison between Indemics and previous simulation methods is presented and a methodology of performance modeling and prediction is proposed in this paper. These performance analysis and prediction results provide us with an analytical basis for evaluating whether Indemics can be used in direct support of a quick turnaround analysis request.

7 ACKNOWLEDGMENTS

We thank our external collaborators and members of the Network Dynamics and Simulation Science Laboratory (NDSSL) for their suggestions and comments. This work has been partially supported by NSF Nets Grant CNS- 0626964, NSF HSD Grant SES-0729441, NIH MIDAS project 2U01GM070694-7, NSF PetaApps Grant OCI-0904844, DTRA R&D Grant HDTRA1-0901-0017, DTRA CNIMS Grant HDTRA1-07-C-0113, NSF NETS CNS-0831633, DHS 4112-31805, DOE DE-SC0003957, NSF CNS-0845700, NSF Netse CNS-1011769 and NSF SDCI OCI-1032677.

REFERENCES

Barrett, C. L., K. R. Bisset, S. Eubank, X. Feng, and M. V. Marathe. 2008. "EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks". In *Proc. ACM/IEEE conference on Supercomputing*, 37.

Bisset, K., J. Chen, X. Feng, Y. Ma, and M. Marathe. 2010. "Indemics: an Interactive Data Intensive Framework for High Performance Epidemic Simulation". In *Proceedings of the 24th International Conference on Supercomputing*, edited by T. Boku, H. Nakashima, and A. Mendelson, 233–242.

Bisset, K. R., J. Chen, X. Feng, V. S. A. Kumar, and M. V. Marathe. 2009. "EpiFast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems". In *Proc. the 23rd International Conference on Supercomputing*, edited by M. Gschwind, A. Nicolau, V. Salapura, and J. E. Moreira, 430–439.

Cauchemez, S., A. Bhattarai, T. L. Marchbanks, R. P. Fagan, S. Ostroff, N. M. Ferguson, D. Swerdlow, and the Pennsylvania H1N1 working group. 2011, March. "Role of social networks in shaping disease transmission during a community outbreak of 2009 H1N1 pandemic influenza". In *PNAS*, edited by D. Cox, 2825–2830.

Chao, D. L., M. E. Halloran, V. Obenchain, and I. M. Longini Jr. 2010. "FluTE, a publicly available stochastic influenza epidemic simulation model". *PLoS Computational Biology* 6 (1).

Christodoulakis, S. 1984. “Implications of Certain Assumptions in Database Performance Evaluation”. *ACM Trans. Database Syst.* 9 (2): 163–186.

DeWitt, D. J., and P. B. Hawthorn. 1981. “A Performance Evaluation of Data Base Machine Architectures (Invited Paper)”. In *Proceedings of the 7th international conference on Very Large Data Bases*, 199–214.

Ferguson, N. M., D. A. T. Cummings, C. Fraser, J. C. Cajka, P. C. Cooley, and D. S. Burke. 2006. “Strategies for mitigating an influenza pandemic”. *Nature* 442:448–452.

Graefe, G. 1993. “Query evaluation techniques for large databases”. *ACM Comput. Surv.* 25:73–169.

Gray, J., D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber. 2005. “Scientific data management in the coming decade”. *ACM SIGMOD Record* 34:34–41.

Parker, J. 2007, December. “A Flexiable, Large-Scale, Distributed Agent Based Epidemic Model”. In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sevcik, K. C. 1981. “Data Base System Performance Prediction Using an Analytical Model (Invited Paper)”. In *Proceedings of the 7th international conference on Very Large Data Bases*, 182–198.

Szalay, A., and J. Gray. 2006. “Science in an exponential world”. *Nature* 7083:413–414.

Szalay, A. S., and J. A. Blakeley. 2009. *Gray’s Laws: Database-centric Computing in Science*. Microsoft Research.

Wang, G., M. A. V. Salles, B. Sowell, X. Wang, T. Cao, A. J. Demers, J. Gehrke, and W. M. White. 2010. “Behavioral Simulations in MapReduce”. *PVLDB* 3 (1): 952–963.

White, W., A. Demers, and C. Koch. 2007. “Scaling Games to Epic Proportions”. In *Proc. the 27th ACM SIGMOD International Conference on Management of Data*, 31–42.

AUTHOR BIOGRAPHIES

YIFEI MA is a PhD candidate in the Computer Science department at Virginia Tech. His research interests include high performance computing, parallel and distributed computing, agent based simulation, data management and interactive computing. He is a member of the ACM. His email address is yifeima@vt.edu.

KEITH R. BISSET is a Senior Research Scientist in the Network Dynamics and Simulation Science Laboratory at Virginia Tech. He received his PhD degree in Computer Science from New Mexico State University. His research interests include high performance computing, parallel and distributed simulation, and modeling complex networks. He is a member of the IEEE Computer Society and ACM. His email address is kbisset@vbi.vt.edu.

JIANGZHUO CHEN is a Senior Research Associate in the Network Dynamics and Simulation Science Laboratory of Virginia Bioinformatics Institute at Virginia Tech. He received Ph.D. in Computer Science from Northeastern University. His research interests are in algorithm design and analysis, agent based modeling and simulation, high performance computing, computational epidemiology, and computational social science. His email address is chenj@vbi.vt.edu.

SURUCHI DEODHAR is a Ph.D. student in the Department of Computer Science at Virginia Tech and working as a Graduate Research Assistant at the Network Dynamics and Simulation Science Laboratory. Her research interests include distributed systems, bioinformatics and modeling and simulations. Her email address is suruchi@vbi.vt.edu.

MADHAV MARATHE is a Professor of Computer Science, and Deputy Director of Network Dynamics and Simulation Science Laboratory at Virginia Tech. He received his Ph.D. in 1994 in Computer Science from the State University at Albany-SUNY. His email address is mmarathe@vbi.vt.edu.