EpiFast: A fast algorithm for large scale realistic epidemic simulations on distributed memory systems

Conference Paper · January 2009

DOI: 10.1145/1542275.1542336 · Source: DBLP

5 authors, including:

Keith Bisset
Virginia Polytechnic Institute and State University
67 PUBLICATIONS **1,190** CITATIONS

SEE PROFILE

Xizhou Feng
Clemson University
36 PUBLICATIONS **1,875** CITATIONS

SEE PROFILE

Sritesh Kumar
Nagarjuna College
135 PUBLICATIONS **4,222** CITATIONS

SEE PROFILE

Madhav Marathe
University of Virginia
499 PUBLICATIONS **10,287** CITATIONS

SEE PROFILE

EpiFast: A Fast Algorithm for Large Scale Realistic Epidemic Simulations on Distributed Memory Systems*

Keith R. Bisset
kbisset@vbi.vt.edu

Jiangzhuo Chen †
chenj@vbi.vt.edu

Xizhou Feng
fengx@vbi.vt.edu

Virginia Bioinformatics Institute, Virginia Tech
Blacksburg, VA 24061, USA

V.S. Anil Kumar
akumar@vbi.vt.edu

Madhav V. Marathe
mmarathe@vbi.vt.edu

Department of Computer Science and Virginia Bioinformatics Institute
Virginia Tech, Blacksburg, VA 24061, USA

ABSTRACT

Large scale realistic epidemic simulations have recently become an increasingly important application of high-performance computing. We propose a parallel algorithm, EpiFast, based on a novel interpretation of the stochastic disease propagation in a contact network. We implement it using a master-slave computation model which allows scalability on distributed memory systems.

EpiFast runs extremely fast for realistic simulations that involve: (i) large populations consisting of millions of individuals and their heterogeneous details, (ii) dynamic interactions between the disease propagation, the individual behaviors, and the exogenous interventions, as well as (iii) large number of replicated runs necessary for statistically sound estimates about the stochastic epidemic evolution. We find that EpiFast runs several magnitude faster than another comparable simulation tool while delivering similar results.

EpiFast has been tested on commodity clusters as well as SGI shared memory machines. For a fixed experiment, if given more computing resources, it scales automatically and runs faster. Finally, EpiFast has been used as the major simulation engine in real studies with rather sophisticated settings to evaluate various dynamic interventions and to provide decision support for public health policy makers.

†Corresponding author. Email: chenj@vbi.vt.edu. Tel.: +1 703 518 8032. Fax: +1 703 518 5376.

*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 NSF Nets Grant CNS-0626964, NSF HSD Grant SES-0729441, CDC Center of Excellence in Public Health Informatics Grant 2506055-01, NIH-NIGMS MIDAS project 5 U01 GM070694-05, DTRA CNIMS Grant HDTRA1-07-C-0113 and NSF NETS CNS-0831633.

ICS'09, June 8–12, 2009, Yorktown Heights, New York, USA.

Categories and Subject Descriptors

I.6.3 [**Simulation and Modeling**]: Applications

General Terms

Algorithms, Experimentation, Human Factors, Performance

1. INTRODUCTION

Pandemics such as avian influenza pose a serious threat to global public health security. Pandemic influenza strains have demonstrated their ability to spread worldwide quickly and to cause infections in all age groups. According to a recent World Health Report 2007, every year human influenza results in an estimated three to five million cases of severe illness and between 250,000 and 500,000 deaths [23]. Although, the final number of infections, illnesses, and deaths is unpredictable, and could vary tremendously depending on multiple factors, it is certain that without adequate planning and preparations, an influenza pandemic in the 21st century has the potential to cause enough illnesses to overwhelm the public health system at all levels.

Computational epidemiology is the development and use of computer models to understand the spatio-temporal diffusion of disease through populations. An important factor that greatly influences an outbreak of an infectious disease is the structure of the interaction network across which it spreads. Aggregate or collective computational epidemiology models often assume that a population is partitioned into a few subpopulations (e.g. by age) with a regular interaction structure within and between subpopulations. However, this approach does not take the structure of underlying social network into account. Additionally, the number of different subpopulation types considered is small and parameters such as mixing rate and reproductive number are either unknown or hard to observe. Over the last few years, it has become increasingly clear that an interaction based approach to computational epidemiology is likely to play an increasingly prominent role in developing public health poli-

cies [17–19]. *Interaction based computational epidemiology* seeks to develop detailed representation of the underlying social contact networks, within and across disease progression and transmission models, as well as computational representations of implementable policies. These models are then appropriately composed to produce detailed computer simulations that can inform public health policies.

Computer simulations are an appropriate tool to study large interaction-based systems. However, developing such high resolution simulations tends to be computationally challenging for a number of reasons, including the following.

- Scale and heterogeneity of social contact networks. For example, a social contact network spanning the New York region is comprised of approximately 20 million individuals. The contact patterns and individual attributes are highly heterogeneous. As a result interprocess communication patterns and load balancing are non-trivial.
- The contact networks are dynamic and coevolve as a result of actions taken by individuals and interventions by public health officials. This implies that non-adaptive schemes for efficient partitioning will most likely not be efficient.
- Use of these simulations in practical settings requires execution of a large number of replicates for a given combination of independent parameters. More importantly, a typical case study involves searching an extremely large parameter space that is best searched using adaptive designs. Factorial designs are practically impossible in such situations.

The above discussion motivates the need for extremely fast HPC-based simulations that are capable of simulating disease dynamics over large dynamic social contact networks. Furthermore, these simulations should be able to represent complex interventions and public policies. This aspect is crucial if such simulations are to be used in practical environments.

Our contributions. In this paper, we present *EpiFast* — an extremely fast, scalable, high performance simulation tool. EpiFast can be used to study how infectious diseases spread through individual populations (in humans as well as animals) as well as the relative effectiveness of various interventions and public policies. It can also be used for answering general questions pertaining to network science, e.g. – which individual nodes are critical or vulnerable in a given population, how does the structure of the network affect disease dynamics, and how to quantify the uncertainty of the results.

EpiFast brings in various novel contributions. Below we highlight several of them. First, EpiFast represents a novel interpretation of epidemic simulations. It is significantly different from the usual multi-agent based discrete event simulations. By using a simpler (yet realistic) disease model and restricting the set of interventions that can applied, EpiFast can use a pre-constructed contact network and avoid scanning agent activities in each step while delivering similar results with comparable details.

Second, the EpiFast algorithm is scalable on almost any distributed memory system. EpiFast incurs low communication cost and tolerates high communication latency. Thus it is able to run on almost any distributed memory system including commodity clusters. To our best knowledge, EpiFast is one of the few epidemic simulations (the other being EpiSimdemics [3]) that does not rely on large shared memory machines.

Third, EpiFast runs extremely fast. For example, a 180-day simulation of the greater Los Angeles metropolitan area, whose population is about 16 million, on a cluster using 96 Power5 processors, takes less than 5 minutes (ignoring the network data load cost). Such high performance comes from a combinatorial interpretation of the stochastic process of disease propagation in a population, as well as an elegant yet efficient parallel algorithm that allows most of the computations to be parallelized in a scalable way. The EpiFast code runs at least several magnitudes faster than other tools while delivering similar results.

Fourth, EpiFast supports representation and reasoning about complex interventions and public policies. It can handle sophisticated interventions to allow realistic case studies. This distinguishes EpiFast from classical simulations based on percolation theory [17, 21]. See Section 2.4 for details.

Last, but also most important, EpiFast demonstrates very good usability in practical applications. We have recently developed a web-based tool called *Didactic* that is integrated with EpiFast so that epidemiologists and public health analysts can easily use EpiFast. Didactic allows users to specify an experimental design using a web based interface. The detailed dynamic information produced by EpiFast is then analyzed with ease by using preconfigured analytical tools.

Organization of this paper. In Section 2, we formally describe the epidemic simulation problem. After presenting the sequential and parallel versions of the EpiFast algorithm in Section 3, we evaluate the performance of an MPI/C++ based EpiFast implementation in Section 4, followed by overviews of two practical applications of EpiFast in Section 5. Related work is described in Section 6.

2. THE EPIDEMIC SIMULATION PROBLEM

In the *epidemic simulation problem*, we are given (i) a population with detailed schedules of each person on each day, which describe at any moment where the person is and what he is doing; (ii) a disease and the characteristics of each person with respect to the disease, which describe how likely the person will be infected, once infected how long he takes to recover, and how likely he will infect other people; (iii) self interventions and public health interventions during an epidemic, which may change the infectivity of the involved people or their schedules; and (iv) initial conditions, which describe how the disease starts with which people. With each simulation run, which represents one possible instance of the stochastic disease propagation, we want to know on each day the health state of each person, and if anyone is infected, who possibly transmits the disease to him.

We assume that we do not care where the disease transmissions occur. The disease transmits through people to people contacts. Therefore, we only need to know the people-people contact graph at any moment. We also assume that in the simulation the discrete time increases by *day*. That is, there is nothing happening between any day and its next day. Without loss of generality, we assume everything occurs at the beginning moment of each day. So in the simulation results, we only care about that a person is healthy on day

t and becomes infected on day $t+1$; we do not care exactly when during day $t+1$ he is infected. Furthermore, we assume that the schedules of each person is known at the beginning of the day, and do not change day to day, unless interventions are applied to the person. Now it's easy to see that if we know the people-people contacts for each day, it is sufficient to simulate the disease propagation.

2.1 Contact network

The people-people contacts form a network, called *social contact network*. A contact network $G(V, E, w)$, is a directed, edge-weighted network. Nodes correspond to individuals in the population, edges represent the contact between two end nodes during the day, and edge weights are contact durations. Edge (u, v) with weight $w(u, v)$ represents that node u has contact of duration $w(u, v)$ with node v on the day, during which the disease may transmit from node u to node v with probability $p(w(u, v))$. Function $p(\cdot)$ is non-decreasing and depends on the disease infectivity. In most cases the edges are bi-directional with equal weights. For each day t, there is a contact network G_t. Since the people-people contacts do not change very often, and the average number of contacts of each person is usually upper-bounded by a constant, we can pre-compute a base network $G = G_0$, and update it when there are interventions.

2.2 SEIR model for within host disease progression

We adopt the standard SEIR disease model which is widely used in mathematical epidemiology literatures [2, 15, 20]. Each person is in one of the following four health states at any time : *susceptible*, *exposed*, *infectious*, and *removed*. Let $\theta(v, t)$ denote the health state of person v on day t. Associated with each person v are an incubation period $\Delta t_E(v)$ and and an infectious period $\Delta t_I(v)$.

A person is in the susceptible state until he becomes exposed. If person v becomes exposed, he remains so for $\Delta t_E(v)$ days, during which he is not infectious. Then he becomes infectious and remains so for $\Delta t_I(v)$ days. Finally he becomes removed (or recovered) and remains so permanently.

The health states of person v during the epidemic period, denoted by $\theta(v)$, can be equivalently represented by $\tau(v) = (t_{S\to E}(v), t_{E\to I}(v), t_{I\to R}(v))$, where $t_{S\to E}(v)$ is the day when v becomes exposed, $t_{E\to I}(v) = t_{S\to E}(v) + \Delta t_E(v)$ is the day when v becomes infectious, and $t_{I\to R}(v) = t_{E\to I}(v) + \Delta t_I(v)$ is the day when v becomes removed. We assume that we know $\Delta t_E(v)$ and $\Delta t_I(v)$ for each person v at the beginning of the simulation. The SEIR model is illustrated in Figure 1.

2.3 SEIR model for between host disease progression

With the SEIR model, the disease spreads in a population in the following way. It can only be transmitted from an infectious node to a susceptible node. On any day, if node u is infectious and v is susceptible, disease transmission from u to v occurs with probability $p(w(u, v)) = 1-(1-r)^{w(u,v)}$ where r is the probability of disease transmission for a contact of one unit time. So the disease propagates probabilistically along the edges of the contact network. Figure 2 shows how the disease spreads via the network and how the SEIR states of the nodes change accordingly.

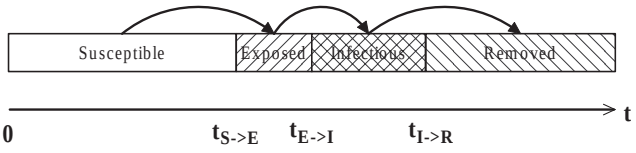


Figure 1: The $S \to E \to I \to R$ disease model of health state transitions. A person in the susceptible state can become exposed. A person in the exposed state will become infectious. A person in the infectious state will become removed. The state transitions are one-way and there are no other possible transitions.

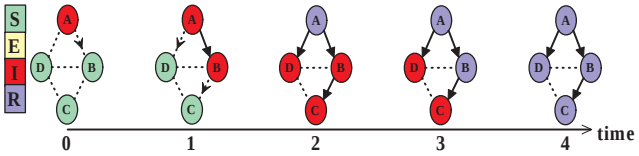


Figure 2: Example of SEIR model for between host disease progression. The contact network has four nodes, and five contact edges. Suppose each node has $\Delta t_E = 0, \Delta t_I = 2$, and each edge represents a transmission probability of 0.5 for each day on which the tail node is infectious (I – red) and the head node is susceptible (S – green). Node A is infectious at start. On day 0, node A transmits the disease to node B but not node D. On day 1, both nodes A and B are infectious. Node A transmits the disease to node D; node B infects node C but not node D. On day 2, node A is removed, all other nodes are infectious but there are no more susceptible nodes. So the nodes are removed gradually and on day 4, the system enters a fixed point status and stops evolving.

A crucial assumption made in almost all epidemic models is that of *independence*: we assume that the spread of infection from a node u to node v is completely independent of the infection from a node u' to node v. Similarly, an infected node u spreads the infection to each neighbor v, independent of the other neighbors of u. This is a central assumption in almost all the epidemic models and the analytical results based on percolation.

2.4 Interventions

Interventions can be broadly classified using the following attributes: (i) Pharmaceutical (PI) versus Non-pharmaceutical (NPI), (ii) non-adaptive/adaptive.

PI interventions include administering antivirals, antibiotics and vaccines. NPI interventions refer to actions that effectively change the social network without administering any drugs. This includes: various kinds of social distancing (closure of various facilities such as schools, etc) quarantine and sequestration. It is now well accepted that NPI measures, especially social distancing can be extremely beneficial in controlling disease spread; see the recent federal pandemic influenza plan [6] for additional discussion.

Non-adaptive interventions occur before the start of the epidemic. The non-adaptive interventions unrealistically assume people's behaviors do not change with the course of the

epidemic. They are limited to studying treatments that have a permanent effect, like vaccination. The adaptive strategies on the other hand, incorporate changes in the movement of the people, treatments that have only temporary effects (antiviral medications are only effective when being taken), and wholesale changes to the interactions within the population (like school closure). We can differentiate various strategies by how frequently interventions are applied and triggering conditions are checked. This can be viewed as degree of adaptation.

At implementation level, an intervention is specified as either changing the infectivity/vulnerability of the intervened person, or changing the behaviors of the intervened person. In the former case, the intervention does not change the people-people interactions, so the disease may transmit through the contact network via the same edges. But the transmissions are hindered at the intervened nodes. This class of interventions include vaccination and antiviral administration. A vaccinated person still goes to the same places, meets the same people, as usual. But now the disease cannot propagate via him. If we apply antiviral on a person, he will become less likely to get infected and even if he is infected he will be less likely to infect other people. The PI interventions are implemented in EpiFast under this class.

The latter class of interventions do change the contact network, so the disease cannot transmit along the same network edges. Examples include social distancing measures. If we apply school closure intervention on a person, his contact edges that take place in the school will be removed, so the disease cannot transmit to him or from him along these edges. The NPI interventions are implemented in EpiFast under this class.

Non-adaptive interventions are implemented as node or edge changes at the beginning of the simulation. Adaptive interventions are implemented using *trigger conditions*. Generally speaking, a trigger condition is a boolean function of the states of all people on all days from the start of the simulation until current simulation day. In EpiFast, this function is computed on every simulation day; if it have value *true* the associated intervention will be executed. Note that this is a stochastic function which can take different values in different simulation instances. So the intervention is adaptive to the simulation progression.

Our EpiFast implementation can handle interventions that have predetermined target subpopulation and change the network edges associated with the intervenees or their infectivity/vulnerability, the interventions can be either non-adaptive, or adaptive so long as the function form of the trigger condition is predetermined.

The following is an example to show what kind of interventions EpiFast are used to evaluate in real studies.

> *When there are more than 100 infectious people in this population, apply antiviral to critical workers for two weeks. If among the school age children, more than 0.1% are infectious, immediately close all public schools. If more than 5% of the population are infected (really major outbreak), 10% people choose to stay home.*

Note that the current implementation of EpiFast cannot handle trigger conditions with function form, or interventions with adaptive targets. For example, a trigger like: "*20% of the contacts of the infectious people become exposed*"; or an intervention like: "*all contacts of the currently infectious people*".

Problem statement. Now we formally state the epidemic simulation problem. Let $G = (V, E, w)$ be the base contact network. Let $\Delta\tau = \{\Delta t_E(v), \Delta t_I(v)\}_{v \in V}$ be the incubation and infectious period durations of the people. Let $A = \{A_t\}_{t \in [0..T]}$ be the interventions during the epidemic. The interventions in set A_t can be predetermined ahead of the simulation, or dynamic based on $\{G_0, G_1, \ldots, G_t, \theta_0, \theta_1, \ldots, \theta_t\}$ where θ_t represents the health state of all people on day t. Let Λ denote the initial conditions, including the initial set of infected individuals. Let $\tau = \{\tau(v)\}_{v \in V}$ be the health states of all people on all days. Let $\pi(v)$ be the set of people that transmit the disease to person v and $\Pi = \{\pi(v)\}_{v \in V}$. This structure is most naturally represented as a *transmission network* TN: it is a directed acyclic graph, the set of initial infections are at level 0. Level i contains the set of nodes (individuals) that become infected on day i. a directed edge from u to v represents the fact that u transmitted the disease to v. A single execution of a discrete event simulation produces a *random* instance of TN; the probability of occurrence of TN is dependent on the various input parameters.

PROBLEM 2.1. *Given* $(G, \Delta\tau, A, \Lambda)$*, the* epidemic simulation *problem is to compute a random instance of the transmission network* TN *and the tuple* (τ, Π).

It is easy to produce an instance of TN in polynomial time. This is because the disease dynamics always reach a fixed point in which the set of individuals are partitioned into two classes, namely, recovered and susceptible. Our goal is to develop a fast parallel algorithm for the epidemic simulation problem.

3. EPIFAST ALGORITHM

A sequential version of the EpiFast algorithm is described in Algorithm 1. Now we show how to parallelize most of the computation using a master-slave model and present the parallel version of the EpiFast algorithm.

To parallelize the EpiFast algorithm, we need to divide the data and computations. The contact network is the major data so we first consider partitioning the network. There are several natural ways to divide a contact network. One is based on that the network consists of a list of undirected edges. So we divide the edges into groups. With this approach, however, it is difficult to keep track of status of each person. For example, if person v is infected, this information has to be synchronized in all groups since we don't know which group does or does not contain edges incident on v. Therefore, any event needs to be synchronized among all processes, which leads to large communication costs. Another approach is to divide the people according to their geographic locations. This relies on that we have their location information and since the population is never evenly distributed, it is difficult to achieve load balancing without complicated partitioning algorithms.

Our network partitioning method is simple yet efficient. We adopt an adjacency list representation of the network, where the directed edges are grouped by the tail nodes. The network naturally consists of $|V|$ components, each corresponding to a node and containing the node and all outgo-

Algorithm 1: Sequential EpiFast algorithm

Input: $(G, \Delta\tau, A, \Lambda)$ where G is the contact network, $\Delta\tau$ is the incubation and infectious period durations, A is the interventions, and Λ is the initial conditions.

Output: (τ, Π) where τ is the health state transition times of all people, and Π is the infectors who transmit the disease to the infected people.

```
initialization
for t = 0 to T do
    foreach person v ∈ V do
        update his health state according to τ(v)
    get interventions A_t and update G accordingly
    foreach infectious node u ∈ V do
        foreach contact v of node u do
            if v is susceptible then
                let transmission u → v occur with
                probability p(w(u, v))
                if u → v does occur then
                    update τ(v) and π(v)
output (τ, Π)
```

ing edges from the node. The $|V|$ components are assigned arbitrarily to processors, each processor containing approximately the same number of edges. Figure 3 is an example of our network partitioning approach. Suppose the component corresponding to v is assigned to processor γ. We say that γ is the owner of v and that v is local to γ. It is obvious that each person is local to exactly one processor and has exactly one owner. All information about person v will be kept on his owner processor. Note that the head node of the edges store on a processor may be local or non-local to the processor.

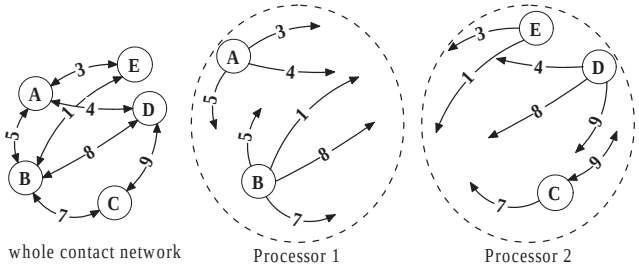


Figure 3: Network partitioning example. A 5-node network is partitioned and assigned to two processors. Each node together with its outgoing edges form a component. Grouping components A, B together and C, D, E together makes the two processors contain the same number of edges (7). This approach achieves approximate load balancing.

Figure 4 depicts the executions and communications in the EpiFast algorithm with a symmetric parallel computation model. The communication complexity is quadratic in the number of processors. So the algorithm is not scalable to many processors. Our solution to this problem is a master-slave computation model, where the single master processor knows the owner processor of each node, while each of the

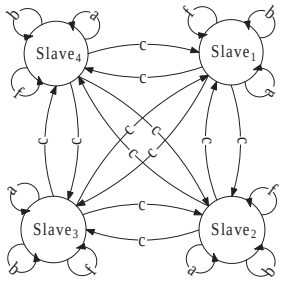


Figure 4: A symmetric model for parallelization of EpiFast algorithm. Sequence of operations:
a: compute probabilistic transmissions from local nodes
b: update $\tau(v)$ of local nodes
c: send transmission $(u \to v)$ where v is non-local to owner of v
f: update $\tau(v)$ of received transmission $(u \to v)$

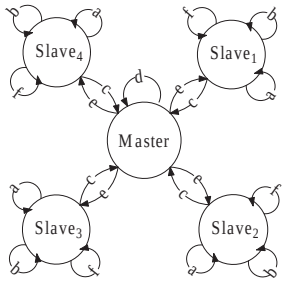


Figure 5: A master-slave model for parallelization of EpiFast algorithm. Sequence of operations:
a: slave computes probabilistic transmissions from local nodes
b: slave updates $\tau(v)$ of local nodes
c: slave sends transmission $(u \to v)$ where v is non-local to master
d: master looks up owners of received messages
e: master sends transmission (u, v) to owner of v
f: slave updates $\tau(v)$ of received transmission $(u \to v)$

slave processors knows for each node whether it is local. The slave processor sends operation requests concerning any non-local nodes to the master processor. The master processor does not do any computation rather than looking up the destination of each request and forward the requests to their owner processors. The master-slave model is described in Figure 5.

Based on Figure 5 we parallelize the EpiFast algorithm as in Algorithm 2. Its computation complexity is linear in the number of infected nodes, and depends on the interventions. About the communication cost we have the following observation.

PROPOSITION 3.1. *The communication cost of Algorithm 2 is upper-bounded by a linear function in the number of infected nodes which does not depend on the number of processors.*

PROOF. Suppose the number of infected node during the epidemic is k. Each infected nodes is infected by a node that has either the same owner or a different owner. The total number of transmissions that need to be forwarded is upper-bounded by k. Therefore the total communication cost is ck where c is the constant cost for forwarding one transmission. □

Our parallel EpiFast algorithm is implemented in C++/MPI. The code is portable and tested on commodity clusters consisting of x86 processors or PowerPC processors, as well as SGI Altix systems.

Algorithm 2: Parallel EpiFast algorithm

```
Input/Output: same as in Algorithm 1.
partition G
initialization
for t = 0 to T do
    if slave then
        foreach local node v do
            update his health state according to τ(v)
        get interventions A_t and update local partition
        of G accordingly
        foreach local infectious node u do
            foreach contact v of node u do
                a: let transmission u → v occur with
                probability p(w(u, v))
                if u → v does occur then
                    if v is local then
                        b: update τ(v) and π(v)
                    else
                        c: send transmission (u → v) to
                        master
    if master then
        receive transmissions from slaves
        d: look up the destination of the transmissions
        e: forward the transmissions to owner slaves
    if slave then
        f: update τ(v) and π(v) for each received
        (u → v)
output (τ, Π)
```

4. EXPERIMENTAL RESULTS

In this section, we describes some detailed performance profiling and analysis of our EpiFast implementation on a terascale system. This system consists of 1000+ nodes and each nodes has two 2.3 GHz PowerPC 970FX processors. The system uses InfiniBand interconnection. We insert timing functions into the EpiFast source code to instrument the running time of each computation phase on all participating processors.

The effects of interventions. Practical studies using EpiFast usually consist of a set of simulations and each of them simulates one or multiple intervention strategies. So it is useful to know the computational cost of interventions. In our performance analysis, we measure the running time without and with interventions. For the scenario without intervention, we use the following settings: The incubation period and infectious period of each person are sampled from distributions such that the former has a mean of 1.9 days and the latter has a mean of 4.1 days. Twenty people are randomly chosen as the seeds to initiate the epidemic. Under these settings, the epidemic has an attack rate[1] of about 42%. For the scenario with interventions, we consider the following two interventions.

1. Intervention `AV`. This is a non-adaptive antiviral intervention. From day 20 to day 39, 10% of the people will be applied antiviral for continuous 20 days. The antiviral drug can make the person 87% less likely to be infected or if he is already infected, make him 80% less likely to infect others. After day 39 the antiviral drug is not effective any more.

2. Intervention `GSD`. This is an adaptive social distancing intervention. When more than 0.1% of the population are infectious, 10% people will avoid unnecessary activities, so they will only go to work or school then come stay home.

[1]The attack rate of the an epidemic is the fraction of the population that ever get infected during the epidemic.

Figure 6 shows the running time of complete simulations (with 25 replicates and 180 days) for the above three scenarios for the Los Angeles area network. The LA network has 16 million people and nearly 900 million people-people contacts. This figure shows that the setting with interventions requires about 75% more computation time than that without interventions. However, the scaling trends are same for both cases, with or without interventions. This figure indicates that EpiFast can finish a complete simulation in about three hours for either scenario or less than 15 minutes for a simulation with only one replicate. This experimentally confirms that EpiFast is a very fast algorithm.

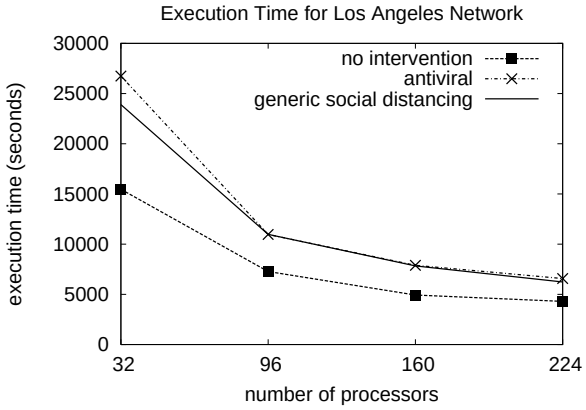


Figure 6: The execution time of EpiFast with and without interventions. There are three scenarios: (i) no intervention, (ii) non-adaptive antiviral intervention, (iii) adaptive social distancing intervention. The experimental settings are the same in all cases, except for the interventions.

The strong scaling of EpiFast. As described in Section 3, by splitting the network nodes across processors, EpiFast reduces the memory footprints of each processor as well as decreases the execution time for simulations. Figure 7 shows the execution times and relative speedups of EpiFast for four different networks: Miami, Boston, Chicago, and Los Angeles. The simulation was configured to use the same incubation and infectious period distributions as described above, plus same kinds of generic social distance interventions. As the simulations requires large memory footprints that can not fit in to a single compute node, we compute relative speedup using the execution time on 32 processors as the base. Figure 7 indicates that EpiFast scales reasonable well for fixed problem size, achieving 3.1 to 4.1 times speedup when using 6x more processors. The diminished performance gains of using more processors are largely caused by the initialization phase that read the network data from disk

Table 1: Experimental Results Under Weak Scaling. *Size* column is the number of nodes (in millions) in the network; *CPUs* column is the number of processors; *Time* column is the average running time (in seconds) per simulation day.

Network	Size	CPUs	Time
Miami	2.09	32	0.47
Boston	4.15	64	0.54
Chicago	9.05	128	0.54

and then distribute to all processors. Our profiling shows that the initialization phases accounts for more than 30% of the total execution time of a single replicate and that cost does not scales well with the number of processors.

The weak scaling of EpiFast. As it is difficult to obtain weak scaling data using realistic networks, we discuss it here empirically. Table 4 shows the experimental results of EpiFast for three networks under the weak scaling rule. As the network sizes are roughly doubled, we double the number of processors to keep the execution time constant. As shown in this table, the execution times of per day simulation for the three networks are indeed close enough. This confirms that the EpiFast algorithm has very good weak scaling properties and thus is feasible to simulate very large networks just by increasing the number of processors.

Better graph partitioning algorithm. EpiFast uses a simple and scalable algorithm to partition the network. Nodes are assigned in the order of their ids to the processors in the order of their ids. That is, there exist $n_0 = -\infty, n_1, n_2, \ldots, n_J$, where J is number of slave processors, such that ids of nodes assigned to processor j are all in interval $(n_{j-1}, n_j]$. Each processor only needs to store $\{n_1, n_2, \ldots, n_J\}$ to know owner of every node. This scheme appears adequate for dealing with networks we are currently working with.

More sophisticated partitioners like Metis [13] can improve the communication complexity. The improvement, however, is limited because the running time of EpiFast is dominated by local computations. We use the k-way partitioning routine in Metis version 5.0 to partition the Miami contact network into 2^i parts, $i = 3, 4, 5, 6$, and measure the average running time of EpiFast for simulating non-intervention case, based on this partitioning. We do the same with our simple partitioning algorithm and compare the two methods. Note that Metis partitioning is done in a preprocessing step and its execution time is not counted in the simulation time, while our simple partitioning is done online during the simulation. Miami population size is about 2 million. The running time is per replicate, averaged over 20 replicates. The simulation is run on an IBM PowerPC-64 (Power4+) cluster, each node of which has eight 1.7 GHz Power5 processors and 16GB memory. The running times under two partitioning schemes are plotted in Figure 8.

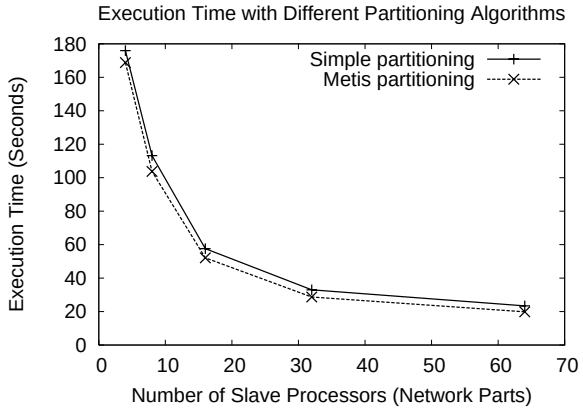


Figure 8: Metis graph partitioner slightly improves performance of EpiFast.

5. APPLICATIONS OF EPIFAST

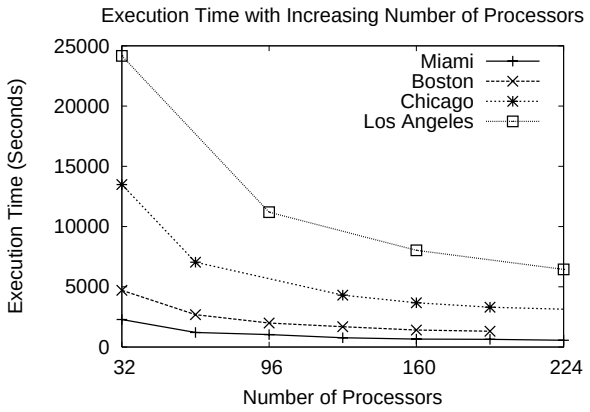
The high performance of the EpiFast algorithm does not sacrifice its usefulness. Our EpiFast implementation has started its applications by epidemiologists to various simulations that compute interesting measures that were previously computationally infeasible, as well as to real studies that aid the public health policy makers to in evaluations and decisions. In Section 5.1 we show a measure that probably only EpiFast can excel in computing. In Section 5.2, we present a real case study.

5.1 Variation in Temporal Vulnerability

We now discuss a setting which illustrates the need for a fast simulation tool such as EpiFast for computing epidemic properties. The vulnerability of a node $f(v)$ is defined as the probability that node v gets infected, starting at a random initial infection. The temporal vulnerability $f_t(v)$ is defined as the probability that node v gets infected during the first t time steps, starting at a random initial infection. The vulnerability measure is a basic property of disease propagation on a social network and its properties are therefore of interest from a public health perspective. It is related to classical bond percolation [21], which helps in fast estimation, but the temporal vulnerability cannot be computed directly by bond percolation techniques. We find that the temporal vulnerability can be estimated by Monte-Carlo simulations using EpiFast, illustrating its power and applicability.

In this experiment we use the contact network representing the Chicago population whose size is about 8.8 million. Regarding interventions we study the following four cases: (i) base case, no intervention; (ii) work closure on day 40 with 50% compliance; (iii) school closure on day 50 with 60% compliance; (iv) work closure on day 40 with 50% compliance and school closure on day 50 with 60% compliance. The attack rate is about 70% in case (i). The variation in the vulnerability distributions after 60 and 80 days are shown in Figures 9 and 10, respectively. The figures show that social distancing resulting from work or school closure has a very significant impact on the disease dynamics.

We run the experiment on an IBM PowerPC-64 (Power4+) cluster with EpiFast. We use four nodes, each with eight 1.7 GHz Power5 processors and 16GB memory. In each case the simulation runs for 500 replicates. The average running time per replicate is 2.35 minutes in case (i), 2.60 minutes in case (ii), 1.52 minutes in case (iii), and 2.03 minutes in case (iv). The difference in running time is mainly due to different interventions and different resultant attack rate: more interventions take more time; smaller attack rate results less running time.



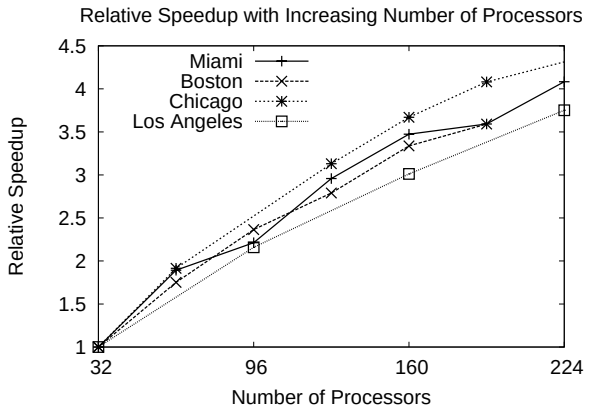


Figure 7: The strong scaling of EpiFast for four different networks. Each simulation includes 25 replicates of 180 days epidemic duration. All times are reported as wall clock time of the entire simulation runs.

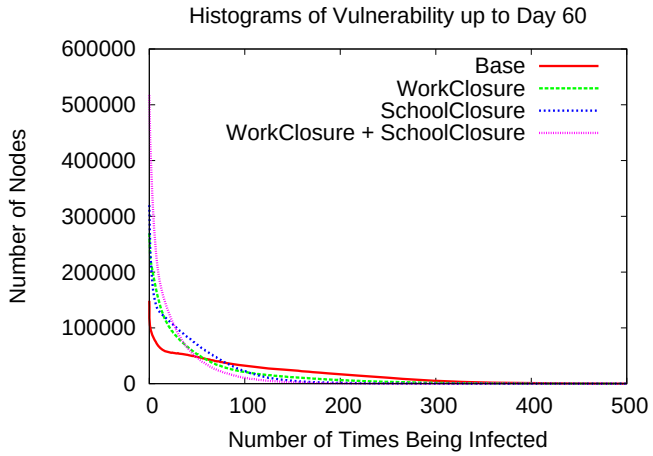


Figure 9: Vulnerability distribution after 60 days.

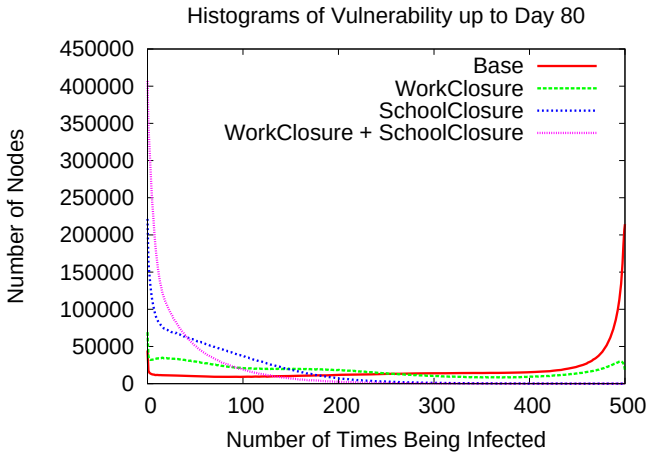


Figure 10: Vulnerability distribution after 80 days.

5.2 Evaluating the effectiveness of interventions

A typical case study is a full experimental design comparing the efficacy of different combinations of interventions. Each combination, called a cell of the experimental design, is run many times to quantify the random variability inherent in a stochastic simulation.

In the following simple study, four different interventions are combined. Each intervention has two levels, off and on, giving a total of sixteen combinations. Each cell is run 25 times with a different random number seed each time. The study was run on a social network representing the greater Los Angeles metropolitan region, specifically the counties of Imperial, Los Angeles, Orange, Riverside, San Bernardino, and Ventura, for a total population of 16 million people.

The interventions used are: a vaccine that has a 30% efficacy, a 40% compliance rate, administered on day 0 of the simulation; an antiviral with an 87% efficacy against infection and an 80% efficacy against transmission, a 60% compliance rate, administered when 1% of the population becomes infected; generic social distancing with a 50% compliance rate, put into effect when 0.5% of the population is infected; and school closure, with a 60% compliance rate, put into effect when 0.5% of the population becomes infected.

Figure 11 shows the results of the sixteen cells of the experimental design. Each point represents the proportion of the population infected for a particular cell, averaged over all 25 replicates of that cell. The left half of the figure shows the cells without school closure (the most effective intervention), and the right half with school closure. Within each half, the left side shows the cells without vaccination (the second most effective intervention), the right side shows the cells with vaccination. This pattern repeats for generic social distancing and antivirals (the least effective intervention). The simulations were run on 50 nodes (200 cores) of a commodity Linux cluster with 2.2GHz dual-socket, dual-core processors, 8 GB of main memory per node, and a Myrinet interconnect.

6. RELATED WORK

Mathematical models have been extensively used by epidemiologist to study the evolution of epidemics. Among them, the compartmental models (e.g., the classic Kermack-McKendrick model [1, 14, 20]), divide the population into *compartments* according to people's health state, and apply differential equations to study the evolution of epidemics in the system. They rely on an assumption of perfect mixing

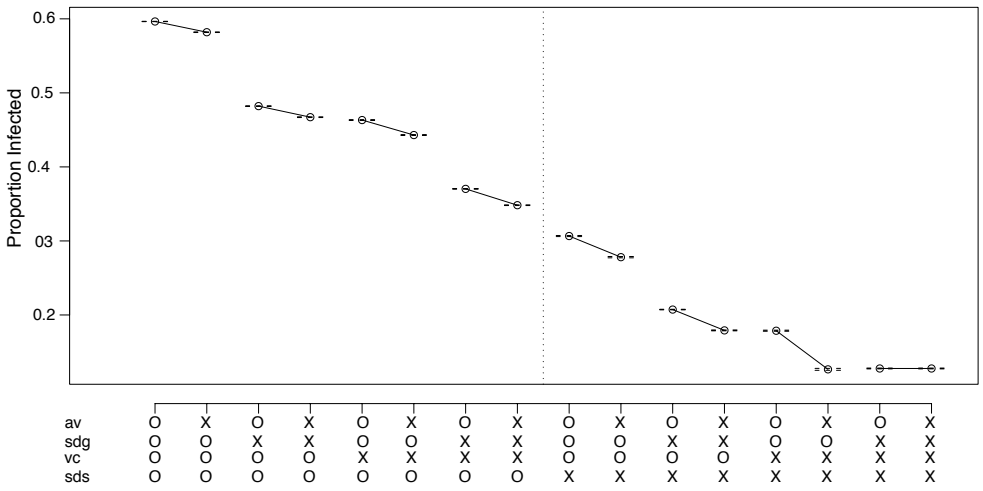


Figure 11: Summary of the results of an experimental design with 4 interventions: vaccination (v), antiviral (av), school closure (sds), and generic social distancing (sdg). An *X* indicates that the corresponding intervention was in effect for that cell, a *O* indicates that the intervention was not in effect. Each point represents the average infection rate across 25 replicates.

that is unrealistic. The contact network epidemiology [17–19] models epidemics as a random propagation process in a heterogeneous contact network and derives analytical results. The analyses in this area, however, often rely on assumptions about the network structure. They are unable to handle arbitrary, irregular contact networks. In addition, physicists have also used a percolation-based approach to simulate the spread of infectious diseases [21]. Such an approach is quite efficient but it, along with the previous approach, *only* yield the final outbreak size; the time varying information about disease dynamics (information about transients) cannot be obtained by such methods.

Recently individual based network models have become a preferred approach for epidemic and social simulations [3, 5, 7, 8, 11, 12, 22]. Among existing simulators, EpiFast is closely related to EpiSims [8, 9] and EpiSimdemics [3]. All these three simulators are build upon the Simdemics framework [4] and provide a high fidelity environment for modeling disease spread across large realistic contact networks. Both EpiSims and EpiSimdemics aim to study very general contagion processes using a probabilistic timed transition system (PTTS) model [3]. They also use person-location network representation that includes detailed individual activities. These two simulators can be used to study very general disease models but requires significant computing resources. By incorporating disease semantics and the structure of social networks into algorithm design and implementation, EpiSimdemics gains much higher scalability than EpiSims that is based classical PDES approach. In contrast to these two simulators, EpiFast significantly speeds up the simulation process by using a simpler (yet realistic) disease model and restricting the set of applicable interventions. Our experimental results indicate that EpiFast usually runs 10 times faster than EpiSimdemics for equivalent networks, though we have to note that these two simulators are based on different simulation methodologies and thereby differ in modeling capability and flexibility.

EpiFast is also related to other parallel simulation systems developed by Longini et al [16], Parker [22] and Ferguson [10, 11]. The system implemented by Ferguson et al. runs on a shared memory platform; so the problem size it it can handle is constrained by the amount of available shared memory. The work of Longini et al. is indeed a parallel simulation like EpiSimdemics. But the locations in its underlying contact networks are not real but surrogates for simple location types such as school, home, etc. This results in a structured social contact network that is more amenable to efficient parallel computation, but which, arguably, is less representative of real-world social networks. Parker [22] has developed a parallel shared memory simulation that can scale to 300 million individuals. Like the work of Longini et al. and Ferguson et al., the underlying social contact networks are highly structured and randomly distributed.

7. CONCLUSIONS

This paper describes EpiFast, a novel parallel algorithm and implementation for simulating disease spread on large realistic social contact networks. By integrating the semantics of the SEIR disease models and using a pre-constructed social contact network, EpiFast reduces the SEIR epidemics simulation problem to a sequence of graph operations, and thereby significantly decreases the simulation cost. By formally incorporate interventions into the simulation problem itself, EpiFast has a wide range of practical applications in public health and social behaviors studies. Though in this paper we focus on our discussion on disease spread, it is worth pointing out that EpiFast is a general fast simulation algorithm that can be applied to other stochastic diffusion processes as well, such as virus propagation in computer networks and knowledge spreading in social networks.

The EpiFast algorithm is scalable on both shared memory systems and distributed memory systems. The algorithm incurs low communication costs and is tolerant to large communication overhead. Thus, it not only runs extremely fast for typical contact networks with several millions of population but also makes reality of simulating ex-

tremely large networks. In this paper, we describe the parallel performance of EpiFast based on a single simulation. However, for practical applications, we can further explore two higher-level parallelisms–replicate level parallelism and cell-level parallelism–to make full use of the computational capability provided by emerging petascale systems.

Our current EpiFast implementation is based on the MPI programming model. In the near future, we pan to investigate EpiFast implementations based on other programming models such as UPC and MPI+OpenMP. Considering some promising performance benefits of accelerator based HPC architecture, we will also explore the feasibility of porting EpiFast to GPGPU or Cell based clusters.

8. REFERENCES

[1] R. M. Anderson and R. M. May. Population biology of infectious diseases: Part I. *Nature*, 280:361–367, Aug. 1979.

[2] N. Bailey. *The Mathematical Theory of Infectious Diseases and Its Applications*. Hafner Press, New York, 1975.

[3] C. L. Barrett, K. R. Bisset, S. Eubank, X. Feng, and M. V. Marathe. Episimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In *Proceedings of the ACM/IEEE Conference on High Performance Computing (SC)*, page 37, 2008.

[4] C. L. Barrett, S. Eubank, and M. Marathe. An interaction based approach to computational epidemics. In *AAAI' 08: Proceedings of the Annual Conference of AAAI*, Chicago USA, 2008. AAAI Press.

[5] K. M. Carley, D. B. Fridsma, E. Casman, A. Yahja, N. Altman, L.-C. Chen, B. Kaminsky, and D. Nave. BioWar: scalable agent-based model of bioattacks. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 36(2):252–265, 2006.

[6] Dept. of Health and Human Services. HHS pandemic influenza plan, 2007.

[7] P. S. Dodds and D. J. Watts. A generalized model of social and biological contagion. *Journal of Theoretical Biology*, 232:587–604, 2005.

[8] S. Eubank. Scalable, efficient epidemiological simulation. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 139–145, New York, NY, USA, 2002. ACM.

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

[10] N. Ferguson, D. Cummings, S. Cauchemez, C. Fraser, S. Riley, A. Meeyai, S. Iamsirithaworn, and D. Burke. Strategies for containing an emerging influenza pandemic in Southeast Asia. *Nature*, 437:209–214, 2005.

[11] N. Ferguson *et al.* Strategies for mitigating an influenza pandemic. *Nature*, 442:448–452, 2006.

[12] T. C. Germann, K. Kadau, I. M. L. Jr., and C. A. Macken. Mitigation strategies for pandemic influenza in the United States. *Proc. of National Academy of Sciences*, 103(15):5935–5940, April 11 2006.

[13] G. Karypis and V. Kumar. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

[14] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, Aug. 1927.

[15] Y. Kuznetsov and C. Piccardi. Bifurcation analysis of periodic SEIR and SIR epidemic models. *Journal of Mathematical Biology*, 32:109–121, 1994.

[16] I. M. Longini, A. Nizam, S. Xu, K. Ungchusak, W. Hanshaoworakul, D. A. T. Cummings, and M. E. Halloran. Containing pandemic influenza at the source. *Science*, 309(5737):1083–1087, 2005.

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

[18] L. A. Meyers, M. E. J. Newman, and B. Pourbohloul. Predicting epidemics on directed contact networks. *Journal of Theoretical Biology*, 240(3):400–418, 2006.

[19] L. A. Meyers, B. Pourbohloul, M. E. J. Newman, D. M. Skowronskic, and R. C. Brunham. Network theory and SARS: predicting outbreak diversity. *Journal of Theoretical Biology*, 232(1):71–81, 2005.

[20] J. D. Murray. *Mathematical Biology I. An Introduction*, volume 19 of *Biomathematics*. Springer-Verlag, 3rd edition, 2002.

[21] M. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[22] J. Parker. A flexible, large-scale, distributed agent based epidemic model. In *Winter Simulation Conference*, 2007.

[23] World Health Organization. World health report 2007: A safe future: global public health security in the 21st century, 2007.